University of the Philippines Manila
College of Arts and Sciences
Department of Physical Sciences and Mathematics

Philippine Stroke Portal

A special problem in partial fulfillment
of the requirements for the degree of
Bachelor of Science in Computer Science

Submitted by:

John Rafael A. Ferrer

June 2016

Permission is given for the following people to have access to this SP:

Available to the general public	Yes
Available only after consultation with author/SP adviser	No
Available only to those bound by confidentiality agreement	No

ACCEPTANCE SHEET

The Special Problem entitled "Philippine Stroke Portal" prepared and submitted by John Rafael A. Ferrer in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science has been examined and is recommended for acceptance.

_____

Vincent Peter C. Magboo, M.D., M.Sc.
Adviser

EXAMINERS:

	Approved	**Disapproved**
1. Gregorio B. Baes, Ph.D. (*candidate*)	_____	_____
2. Avegail D. Carpio, M.Sc.	_____	_____
3. Richard Bryann L. Chua, M.Sc.	_____	_____
4. Perlita E. Gasmen, M.Sc. (*candidate*)	_____	_____
5. Marvin John C. Ignacio, M.Sc. (*cand.*)	_____	_____
6. Ma. Sheila A. Magboo, M.Sc.	_____	_____

Accepted and approved as partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science.

_____

Ma. Sheila A. Magboo, M.Sc.
Unit Head
Mathematical and Computing Sciences Unit
Department of Physical Sciences and Mathematics

_____

Marcelina B. Lirazan, Ph.D.
Chair
Department of Physical Sciences and Mathematics

_____

Leonardo R. Estacio Jr., Ph.D.
Dean
College of Arts and Sciences

Abstract

Patients with stroke and their caregivers need to be assisted for their coping mechanisms on the effects on stroke. It can be achieved through the use of a web portal that will aggregate various medical resources for the patients and caregivers. This study aims to create a stroke web portal in the Philippine setting. This approach will help the patients with their coping mechanisms and for the betterment of the quality of life. Additionally, this web portal will enable forum threads among medical professionals and patients or caregivers in order to discuss stroke, its types, symptoms, diagnosis, complications, initial treatment, and long-term treatment.

Keywords: stroke, web portal, medical professional, nutritionist, rehabilitation medicine doctor

Contents

Acceptance Sheet i

Abstract ii

List of Figures v

List of Tables vii

I. Introduction 1

A. Background of the Study 1

B. Statement of the Problem 3

C. Objectives of the Study 4

D. Significance of the Project 6

E. Scope and Limitations 7

F. Assumptions 8

II. Review of Related Literature 9

III. Theoretical Framework 16

A. Stroke 16

1. Types 16

2. Signs and Symptoms 17

3. Diagnosis 18

4. Complications 18

5. Initial Treatment 19

6. Long-Term Treatment 19

B. Philippine Neurological Association 21

C. Stroke Society of the Philippines 22

D. Web Portal 23

E. Patient Medical Records 23

IV. Design and Implementation **25**

A. Context Diagram 25

B. Use Case Diagram 26

C. Dataflow Diagram 27

D. Entity Relationship Diagram 31

E. Database Design 33

F. System Architecture 47

G. Technical Architecture 48

V. Results **49**

VI. Discussions **89**

VII. Conclusions **93**

VIII. Recommendations **94**

IX. Bibliography **96**

X. Appendix **101**

A. Source Codes 101

XI. Acknowledgement **327**

List of Figures

1 Context Diagram of PSP 25
2 Use Case Diagram of PSP 27
3 Top Level Data Flow Diagram of PSP 28
4 Sub-explosion for Login Function of PSP 29
5 Sub-explosion for Input Medical Condition, Record, and Regimen Function of PSP 29
6 Sub-explosion for Manage Visits Function of PSP 30
7 Sub-explosion for Forum Thread Function of PSP 30
8 Sub-explosion for Manage Online Medical Resources Function of PSP 31
9 Sub-explosion for Manage Accounts Function of PSP 31
10 Entity Relationship Diagram of PSP 32
11 System Architecture of PSP 48
12 Home page of PSP 49
13 Index page of announcements and events of PSP 50
14 Index page of medical resources of PSP 51
15 Index page of forum threads of PSP 52
16 Download a medical resource file of PSP 53
17 Login page of PSP 54
18 Forgot password page of PSP 54
19 Register page of PSP 55
20 Account profile of a user of PSP. 56
21 Edit page of account profile of a user of PSP. 57
22 Upload page of profile picture of a user of PSP. 58
23 Change password page of PSP. 58
24 View medication regimen page of PSP. 59
25 View nutrition regimen page of PSP. 60
26 View rehabilitation medicine regimen page of PSP. 61
27 Input daily medical condition page of PSP. 62

28 Input laboratory result page of PSP. 63
29 Uploading a scan file of laboratory result of PSP. 64
30 Setting a follow-up visit after inputting a medical condition of PSP. 65
31 Setting a follow-up visit directly to the doctor of PSP. 66
32 E-mail notification on the approved visit schedule of PSP. 66
33 Participate in a forum thread of PSP. 67
34 View page of the patients of the health professional of PSP. 68
35 View page of patient's daily medical conditions of PSP. 69
36 View page of patient's laboratory results of PSP. 70
37 Make a decision to a medical condition of a patient of PSP. 70
38 View page of medical records of a patient of PSP. 71
39 Update page of medical records of a patient of PSP. 72
40 Refer patient to another health professional of PSP. 73
41 View page of the medication regimens of PSP. 74
42 Update page of the medication regimens of PSP. 75
43 Approve of follow-up visit schedules page of PSP. 76
44 Upload medical resource page of PSP. 77
45 Notification of an approved medical resource page of PSP. 78
46 Notification of an uploaded medical resource page of PSP. 79
47 Approve medical resource page of PSP. 80
48 Moderate particular forum thread of PSP. 81
49 Approve a forum thread of PSP. 82
50 Update page of patient nutrition records of PSP. 83
51 Update page of patient rehabilitation medicine records of PSP. . . . 84
52 Activate/deactivate staff accounts of PSP. 85
53 Posting announcements and events page of PSP. 86
54 Manage supervisor accounts page of PSP. 87
55 Manage local hospital or clinic administrator page of PSP. 88

List of Tables

1 Users table . . . 33
2 Password resets table . . . 34
3 Patients table . . . 34
4 Medical professionals table . . . 34
5 Nutritionists table . . . 35
6 Physiatrists table . . . 35
7 Supervisors table . . . 35
8 Local admins table . . . 35
9 System admins table . . . 35
10 Hospitals table . . . 36
11 Hospital medical professions table . . . 36
12 Hospital nutritionist professions table . . . 37
13 Hospital physiatrist professions table . . . 37
14 Medical conditions table . . . 38
15 Laboratory results table . . . 38
16 Patient attributes table . . . 39
17 Patient attribute types table . . . 39
18 Medication records table . . . 39
19 Medical record attributes table . . . 40
20 Medical record attribute types table . . . 40
21 Nutritionist records table . . . 40
22 Nutritionist record attributes table . . . 41
23 Nutritionist record attribute types table . . . 41
24 Physiatrist records table . . . 41
25 Physiatrist record attributes table . . . 42
26 Physiatrist record attribute types table . . . 42
27 Medication regimens table . . . 42
28 Nutritionist regimens table . . . 43

29 Physiatrist regimens table 43
30 Forum threads table 44
31 Forum categories table 44
32 Forum replies table 44
33 Follow-up visits table 45
34 Decision on medical conditions table 45
35 Announcements and events table 46
36 Medical resources table 46
37 Files table 47
38 Notifications table 47

I. Introduction

A. Background of the Study

Stroke, which is also known as cerebrovascular accident, is caused by poor blood flow or blood supply interruption to the brain. This is effectuated by bursts of blood vessels or blocking of blood vessels due to clots. [1] Consequently, oxygen does not reach the brain cells and these cells begin to die. Hence, the person who is suffering from stroke is unable to retain memory, speak, or conduct other muscle controls when the corresponding brain cells that perform these abilities are affected. [2] The main symptoms of stroke include lowering of one side of the face; inability to raise one arm; numbness of arm or leg, specifically on one side of the body; and trouble in speaking and understanding. [3]

Two kinds of stroke can be caused by blood supply interruption to the brain. First, the ischemic stroke, which is the more usual kind, is caused by a blocking or plugging of a blood vessel in the brain due to a blood clot. The second is hemorrhagic stroke, which is attributed to the breaking and bleeding of a blood vessel in the brain. Additionally, transient ischemic attacks, which are also called mini-strokes, happen due to a brief interruption of the brains blood supply. [4]

The World Health Organization stated in 2014 that the deaths in the Philippines due to stroke reached 63,261 or 12.14% of the total deaths. [5] In addition, stroke is the second to cardiovascular system as leading cause of death in the country. The prevalence or the condition of being widespread of stroke in the Philippines is at 0-9%. The rate of stroke in the country is ever higher than the rate of myocardial infarction. Ischemic stroke is the most common type of stroke, which is 70% of the total stroke cases, while hemorrhagic stroke account for the remaining 30%. Actually, the percentages of ischemic stroke and hemorrhagic stroke caused by intracranial vascular disease are higher as compared to prevalence in the Western countries. [6]

With this in mind, some professional associations in the country aim to help

stroke patients and caregivers. The Philippine Neurological Association (PNA) is an organization internationally acclaimed as one of the leading neurological associations in Asia in implementing quality neurological care, education, and research. One of the councils that they constituted is the Stroke Council, which aims to coordinate research, advocacy, public information, service, and training on stroke with their accredited institutions and hospitals. They have also conducted Family and Caregivers Workshops in their annual convention to educate the caregivers. [7]

Another one is the Stroke Society of the Philippines (SSP), which is an association that targets the reduction the incidence of stroke, reduction morbidity and mortality from stroke, and improving the quality of life of those who suffered from stroke. They had published guidelines for the prevention, treatment, and rehabilitation of stroke intended for medical professionals and caregivers. [8] They also host the Brain Attack Awareness Week, an awareness campaign for stroke prevention, every third week of August every year. Likewise, they have launched a Stroke Research Contest, which aims to encourage doctors to study further treatment modalities and develop surgical methods to improve stroke treatment in the country. [9]

In order to supplement the coping mechanisms of stroke patients and caregivers, they must be assisted with information about the medication, treatment, rehabilitation, and other aspects of stroke. It can be achieved through the help of medical resources and information, which will educate and aid them in coping with the effects of stroke. Furthermore, these resources will be of much more assistance if these are validated by neurologists and other health professionals. Another way is to enable the patients and caregivers to communicate with medical professionals through forum threads in order to be more informed.

These can all be attained through the use of a web portal, which amasses these validated information and resources from various medical professionals in a single point of access and implements forum threads among its users to encourage

interaction between the medical professionals and patients or caregivers regarding stroke. A web portal is a web site that incorporates the information and resources from a computer information system in order to present and distribute them into a certain set of users. [10]

B. Statement of the Problem

However, there is a lack of web portal of stroke in the Philippines. Even though the PNA and SSP have implemented various advocacies in performing stroke health-care services, educating people regarding stroke, and raising public awareness in it, local stroke web portal has not yet been developed in aggregating validated online medical resources and presenting these information to patients and caregivers in a convenient way.

In addition, there is also much cost on the part of patients with stroke and caregivers when they frequently carry out face-to-face visits in neurological clinics and hospitals in particular scenarios for minor daily medical conditions. There has not been a way to enable the patients and caregivers to set visiting schedules in clinics and hospitals online. There is also a lack of way in notifying them which institutions within their vicinity or within the country have neurological clinics, neurologists, and other health professionals that cater to the needs and concerns of stroke patients.

Stroke patients and caregivers also do not have a way of discussing other non-medical remedies for stroke recovery. The aforementioned professional associations do not implement the use of forum threads in their websites for enabling stroke patients and caregivers to ask certain aspects of stroke, such as medication, lifestyle, diet, and rehabilitation. Also, the neurologists, nutritionists, and rehabilitation medicine doctors do not have a way of elaborating or explaining these aspects to the patients through a single website since there is no available web portal of stroke in the Philippines.

To address these problems, a free web portal for the Philippine setting is es-

sential, which will provide medically validated information regarding stroke, its symptoms, diagnosis, and treatments, rehabilitation, and other aspects. Likewise, the web portal will implement forum threads, which will enable stroke patients and caregivers to communicate with medical professionals to discuss about their condition.

C. Objectives of the Study

This project will create a Philippine Stroke Portal (PSP) with the following functionalities:

1. Allows patients and caregivers to:
    (a) Update account profile and password
    (b) Input minor or daily medical condition/s for probable online check-up
    (c) Input laboratory results done on outpatient basis
    (d) View daily medication regimen
    (e) View daily nutrtion regimen
    (f) View daily rehabilitation medicine regimen
    (g) Schedule for a follow-up visit to doctors, nutritionist clinic, and rehabilitation clinic
    (h) Receive notification of the set schedule through e-mail
    (i) View approved online medical resources about stroke
    (j) Save files of approved resources, like videos and images
    (k) Participate in forum threads on various topics on stroke and daily coping mechanism to it in order to highlight other non-medical remedies
2. Allows medical professionals to:
    (a) Register for an account and update account profile and password

(b) Register a patient for an account and update patient account profile

(c) View and update patient medical record/condition and laboratory tests

(d) Make decision in advising follow-up visit of a patient to a clinic or recommending immediate hospitalization

(e) Refer patient to another physician for other medical concerns

(f) Refer patient to a nutrition clinic

(g) Refer patient to a rehabilitation medicine clinic

(h) Approve patient request for scheduled visit

(i) Post online medical resources about stroke

(j) Receive notification through the web portal when a supervisor approves medical resource uploads

(k) Participate in forum threads

3. Allows supervising doctor or health professional to:

(a) Register for an account and update account profile and password

(b) Receive notification through the web portal when a medical resource needs approval

(c) Approve the posting of online medical resources of doctors, nutritionists, and rehabilitation medicine doctor

(d) Act as moderator in a forum thread

(e) Participate in forum threads

4. Allows rehabilitation medicine doctors to:

(a) Register for an account and update account profile and password

(b) View and update patient rehabilitation medicine record

(c) Approve patient request for scheduled visit

(d) Participate in forum threads

5. Allows nutritionists to:
    (a) Register for an account and update account profile and password
    (b) View and update patient nutritional record
    (c) Approve patient request for scheduled visit
    (d) Participate in forum threads
6. Allows local hospital or clinic administrators to:
    (a) Register for an account and update account profile and password
    (b) Manage user accounts of local staff
    (c) Post announcement on events and news of the local hospital or clinic
    (d) Participate in forum threads
7. Allows system administrators to:
    (a) Update account profile and password
    (b) Manage user accounts of local hospital or clinic administrators
    (c) Manage user accounts of supervising doctors or health professionals
    (d) Participate in forum threads
8. Allows guest users to:
    (a) View online medical resources about stroke
    (b) View forum threads

D. Significance of the Project

This web portal will be useful in educating people about stroke using validated online medical articles and resources posted by the medical professionals. Through these available resources, as well as the application of forum threads, users of the portal will be able to understand the types of stroke, its symptoms, diagnosis,

initial treatment, and long-term treatment, especially when they are at home. For that reason, the system will be helpful to patients and caregivers in their coping mechanism to stroke; the betterment of the quality of life will be achieved.

There will also be a cost reduction since it minimizes frequent face-to-face visit to hospitals and clinics. Besides, it will lessen the difficulty and time of hospital personnel in organizing and retrieving patient information compared to the usual paper-based method. The system will also serve as a way of health professionals in advertising their clinics and hospitals. Additionally, it will help other hospitals and clinics in promoting their news and events, especially in relation to stroke.

E. Scope and Limitations

1. Patient account will be made on the first consultation in the hospital or neurologist clinic.

2. The process of registering other accounts for medical professionals and assuring the veracity of their information is primarily done by the local administrator.

3. The process of registering a supervising medical professional account is done by the system administrator in compliance with the provided authority of the Philippine Neurological Association to register the user.

4. The moderation of the forum threads will be up to the supervisors in the portal; deciding to remove certain members in a forum thread or terminating a forum thread will be based on the judgment of the supervisor.

5. The file types of the files to be uploaded by the medical professionals to the portal are limited only to JPEG and PNG images, MP4 videos, and Portable Document Format files.

6. The system does not include the generation of reports of the cases of patients, like graphs and charts.

F. Assumptions

1. If a patient with stroke has difficulty in being a user of the portal and is represented by a caregiver, it is assumed that the caregiver is a trustworthy one and has a legal authority to represent the patient.

2. If a patient transfers to another doctor or hospital, he or she must be treated as a new patient and, hence, a new record.

3. If the doctor or health professional belongs to several hospitals or clinics, he or she still has one account only.

4. An account for the system administrator is already present in the web portal upon its deployment.

II. Review of Related Literature

Web portals for stroke have already been developed in other countries. The Cardiovascular Health Partners Web Portal is a portal that helps people in navigating through a multitude amount of information available from each of their involved partners that aim to provide awareness and prevention to stroke and heart diseases. These partners include American Heart Association, American Stroke Association, Centers for Medicare and Medicaid Services, National Institute of Neurological Disorders and Stroke of the National Institutes of Health, and other institutions. [11]

While this portal is very helpful in educating oneself about stroke, it lacks the implementation of presenting online medical resources on stroke and forum threads for communication between medical professionals and patients or caregivers. They just mainly help people in exploring different aspects of stroke, as well as heart diseases.

The use of information systems and electronic heath records have already been introduced before to handle, manage, and interpret stroke patient records and hospital information. One is the Electronic Stroke CarePath, which consists of technologically enabled care pathways. These care pathways include systematically collected physician outcome information and the outcome information based on the patients perspective. It was developed in Cleveland Clinic, a multispecialty academic hospital located in Cleveland, Ohio. It is implemented for patients with ischemic stroke and healthcare during their acute hospitalization and outpatient follow-up. [12]

It consolidated three strategies in improving the efficiency and quality of healthcare. The first was the standardization of care, which is accomplished through the use of care paths, or management plans that provide the sequence and timing of actions necessary for optimal efficiency of healthcare. The second one was the use of health information technology, which reconstructs clinical pathways to electronic workflow in electronic health records. The third was the systematic assessment

of patient-reported outcomes. Information regarding the patients symptoms and health status from the patients point of view were provided to the clinician; thus, healthcare providers would attain more thorough information for better decisions. [12]

Also, a Web-based user-interface for clinical decision support tool was developed to focus on secondary stroke prevention. A Self Management TO Prevent (STOP) Stroke tool was designed to improve the management of stroke risk factors, especially in susceptible people who were previously diagnosed with ischemic stroke or transient ischemic attack. It applied evidence-based recommendations for patient-specific problems. It was developed within the Computerized Patient Record System of the United States Department of Veterans Affairs. [13]

The system featured provider-to-patient communication and implementing patient-centred care using electronic health records. It provided access to printable patient-education materials and instruction for patient self-management action plans for secondary stroke prevention. Besides, clinical decision support tools, which were based on the routine workflow of clinicians, were incorporated in the system and in electronic health records to assist in decision-making process of healthcare providers. Also, an action-planning or goal-setting option was planned to be included in the system. This would demonstrate patient accountability; the patient would be involved in the decision-making process with his or her provider about how to reduce his or her stroke risk. The system was then perceived as helpful for communication between healthcare providers and patients about stroke risk-factor management. [13]

The use of telecommunication and information technology systems in delivering distant clinical health care have also extended in evaluating stroke patients. The AcuteCare Telemedicine (ACT) had delivered telemedicine services for stroke patients. Using telemedicine systems, four practice-based neurologists had administered evaluation for patients with neurologic emergency cases at remote hospitals. They were also augmenting access to the use of tissue-type plasminogen activator

(tPA), which boosts recovery of stroke patients. [14]

In the process of telestroke evaluation, emergency room physicians would first consult ACT neurologists. Then, the ACT neurologists would perform remote neurologic examination, which involved an assessment of National Institutes of Health Stroke Scale (NIHSS) from the American Heart Association, to patients that show symptoms of stroke. Telestroke video evaluations were also conducted to the patients. They would accordingly study the computed tomography brain scan and laboratory results. Treatment instructions, which comprised of the use of intravenous (IV) tPA or neurointerventional care referral, would be then conferred with the emergency room physician. The patient information and video evaluations would be stored in their database. [14]

In a recent study, the administering of telestroke had lead to its high demand in the hospital industry. With the use of telemedicine systems, the hospital network of ACT grew from 7 to 20 hospitals between November 1, 2012, and November 30, 2014. The network was within Georgia, Alabama, South Carolina, and Tennessee. The number of hospital beds in these hospitals ranged from 35 to 402. Besides, the study showed that implementing telestroke for stroke care can aid hospitals for the betterment of their processes and the outcomes of stroke cases. [15]

The researchers also noted the important factors in improving stroke care aside from having better telemedicine system and electronic health records. These were having better communication between hospital staff and telestroke neurologists, reinforcement in learning enhanced stroke care processes, engagement comprehensive stroke centers for regional expertise, and sharing of outcome measures among telestroke neurologists and healthcare providers. [15]

Furthermore, telemedicine systems were performed in the administration of thrombolytic therapy for hemorrhagic stroke patients in rural areas. Thrombolytic therapy includes the use of pharmacological drugs for clot busting to reduce continuing disability and lower mortality from stroke. However, the stroke equipment and technology, like computed tomography scanners, stroke emergency mobile

ambulances, and transcranial ultrasound, are only available within small radius of urban areas. Also, it is also necessary for people using transcranial ultrasound to have expert advice to administer accurate diagnosis; hence, paramedics in rural areas need support immediately from neurologists in urban areas to evaluate people with stroke. [16]

In light to this, the combination of ultrasound machines and an intelligent communication device, called Omni-Hub, was employed in Scotland. Omni-Hub is a universal sensor and communication hub that yields a long-range Wi-Fi hotspot. In this method, the clinician in the remote place would examine the patient for possible stroke symptoms. Then, the ultrasound operator would explore for the presence of intracranial haemorrhage. In real time, the ultrasound video and images would be streamed for review using the communication device. After receiving the streamed media, further treatment, like thrombolysis, would be made to the stroke patient after the assessment of the condition by the neurologist in the urban hospital. [16]

Aside from telemedicine systems, mobile technology is currently being implemented in examining patients with stroke. The mobile application IScore, or Ischemic Stroke outcome predictive risk score, was developed which takes various estimates from patients with acute ischemic stroke. The estimates are the risk of death at 30-day and 1-year period of the condition, functional outcomes, response to intravenous thrombolysis, risk of bleeding, and the institutionalization at discharge early after hospital admission or while assessing condition in the emergency department. [17]

The application was primarily used for patients from a Canadian population. Hence, a study was conducted in determine the usefulness of prediction of IScore about clinical response to thrombolysis of Korean patients with acute ischemic stroke. Indeed, the researchers discovered that the mobile application was reliable in predicting the clinical response after administering IV tPA during 3 months to stroke patients from a Korean population. It also reliably predicted other

stroke outcomes, which vary from short-term to long-term mortality and functional recovery before thrombolysis, during discharge, and during 30 days, 3 months, and 1 year of hospitalization. [18]

In another study, Ovbiagele proposed an integrated blood pressure self management intervention, which is called Phone-based Intervention under Nurse Guidance after Stroke (PINGS), for patients suffering from stroke in hospitals in countries in Sub-Saharan Africa. The researcher comprehended the intervention for hypertension because hypertension is a major risk factor for cardiovascular disease and is highly associated with stroke. It was comprised of blood pressure control clinics of nurses and the implementation of health technology, such as phone text messaging and monitoring devices of patients by healthcare providers. The tool would measure the level of hypertension of its patients and send the results to the healthcare provider. [19]

Techniques in implementing healthcare and rehabilitation for stroke patients using novel computer technologies are also being innovated and accomplished. In the research of Linden, Waights, Rogers, and Taylor in 2012, they discussed several approaches in the development of pervasive healthcare systems that will assist in stroke patient rehabilitation. First, they explained the user-centred approach, which takes the particular necessities and desires of users into consideration while initializing the system and during its development. An example was the SMART project, which had a sensor-based system for the arm to stretch. It was designed in order to support people recovering from stroke. The type of armband and type of fastener used were determined with coordination and satisfaction of the patients. [20]

They also mentioned technological inspirations for pervasive healthcare systems. It associates fabricating physical prototypes with various sensors that interact with the environment of the user. An example that employed prototype with electrical components and sensors was the MusicJacket: a wearable system or jacket that utilizes the sense of touch to support the teaching of good posture

and bowing technique to beginning violin players. The system was then integrated to assist in stroke rehabilitation by making use of the haptic feedback mechanism. Through vibration motors, the jacket would support stroke patients in their physiotherapy session by strengthening them to move their impaired limb or trunk during therapy exercises. [20]

Another software tool for rehabilitation was also developed. A flexible and fully customizable visualisation software tool was invented to allow the exploration of different visual techniques for stroke patients during the rehabilitation. It utilized three-dimensional motion capture technology that visualizes the movements of a patient. The flexibility of the software was attributed to the usage of appropriate motion capture data per patient to generate visualisations of data essential to their rehabilitation program. It was also flexible by considering the customizing the interface and visualisation options presented to the patient. This tool was effective in helping the patient recovering from stroke to visualize and accomplish exercises during rehabilitation. It was formed through multidisciplinary collaboration among the University of Strathclyde, The Glasgow School of Art, and Glasgow Caledonian University in Scotland. [21]

At present time, more ways of home-based telerehabilitation are being established and implemented for providing clinical assessment and clinical therapy for disabled people. The delivery of rehabilitation services through telecommunication networks and the internet contributed to the enhancement of the daily activities and independence of recovering stroke patients. [22]

In order to perceive the implications and enhancement made by telerehabilitation, a multitude of literature from electronic databases, such as Cochrane Library, AMED, Medline, and Embase, and from conferences were collected and analyzed. Out of the 2587 records, 11 studies, which met the predefined inclusion criteria, were considered. The results of the study indicated that telerehabilitation was efficient among stroke patients in recovery from stroke and in activities of their daily life. It also showed that telerehabilitation contributed the same with conven-

tional rehabilitation in terms of alleviating daily activities and performing motor functions. It also improved cost-effectiveness; the expenses of healthcare providers from the telerehabilitation were lower than the expenses in a conventional rehabilitation. [22]

In determining the use of health technology in the prescription of drugs for cardiovascular issues, a computerized decision support system was examined in Italy. It was developed for the betterment of pharmacological management in high-risk cardiovascular patients to type 2 diabetes mellitus, heart attack, and stroke. It assisted healthcare professionals in reducing cardiac and cerebral events, like myocardial infarction and stroke, with administration of antithrombotic, antihypertensive, and lipid-lowering drugs. This system provided further interventions in conducting healthcare practice and augmenting preventive measures. [23]

III. Theoretical Framework

A. Stroke

Stroke, or cerebrovascular accident, is a medical condition in the brain that occurs when a portion of the brain is blocked from receiving blood. Blood carries oxygen to different parts of the brain; hence, when a deprivation of oxygen happens, brain cells begin to die after a few minutes. Stroke also happens due to unexpected bleeding inside the brain. [3]

Stroke was first recognized over 2400 years ago. It was discovered by Hippocrates, the father of medicine, and it was called apoplexy at that time. [24] Apoplexy comes from the Ancient Greek apoplexia that means struck down by violence. [25] It was because people experiencing the condition were suddenly paralyzed and altered health and mind state.

Later on the mid-1600s, Jacob Wepfer revealed that bleeding inside the brain and blocking in one of the blood vessels in the brain caused apoplexy among patients. Then, in 1928, while further studying the condition, apoplexy was divided into different categories with reference to the problem with a blood vessel. Some of these were stroke or cerebrovascular accident, ovarian apoplexy, and pituitary apoplexy. [24]

Stroke is also called frequently called brain attack in similarity with heart attack, which is caused by the lack of oxygen-rich blood to the heart. Brain attack is also used to call immediate attention and emergency from the public in case it occurs to a person. [24]

1. Types

There are two main types of stroke. The first is the ischemic stroke, which is caused by a blood clot. A blood clot in the brain causes the blocking of its arteries and the starvation of blood flow. Ischemic stroke can occur as an embolic stroke or a thrombotic stroke. Embolic stroke occurs when a blood clot inside the body

travels through the blood flow in the brain. Then, it travels to a small blood vessel that it cannot get across to; hence, the clot gets stuck in it and impedes the blood flow. On the other hand, thrombotic stroke occurs when blood flow in the major arteries in the neck leaves cholesterol-laden plaques that affix in the inner wall of the artery. The size of the overall plaques eventually increases and stops the taking of the blood to the brain. [26]

The second is the hemorrhagic stroke, which is caused by a fissure in a blood vessel. This opening leaks blood in the brain and consequently impedes the flow of oxygen and nutrients. This type of stroke is distinguished based on its location. First, intracerebral haemorrhage happens when an artery inside the brain collapses and bleeds. Second, subarachnoid haemorrhage happens when bleeding is on the surface of the brain, particularly between the layer of membrane closest to the brain and the second layer of meninges. [26]

A transient ischemic attack, often called as mini-stroke, is also caused by a blood clot. The difference is that its symptoms occur rapidly and it only lasts for a minute. However, it should not be ignored and should be recognized as a warning sign for risk to stroke. [27]

2. Signs and Symptoms

As stroke happens instantaneously, its symptoms appear unexpectedly. Its symptoms include headache, possibly with changed consciousness or vomiting; trouble with speaking and understanding; numbness of the face, arm or leg, especially on one side of the body; difficulty in vision, and trouble with walking, with dizziness and lack of coordination. [28]

In addition, the warning signs may be different to each person. It is because these signs depend on which side and part of the brain that has been affected and the asperity of the injury. [28]

Furthermore, the National Stroke Foundation advises the use of the F. A. S. T. test to distinguish the sudden signs of stroke. To recognize the immediate

signs of stroke, one must know that a person, who is risk of stroke, will exhibit the following: Face drooping, Arm weakness, and Speech difficulty. Hence, they highly recommend that it is Time to call 911 to identify the condition. [29]

3. Diagnosis

Diagnosis of stroke involves the consideration of signs and symptoms, medical history, physical exam, and test results. A doctor will question about the risk factors, such as high blood pressure, smoking, heart disease, carotid artery disease, and previous cases of stroke in the patient or family. Accordingly, the doctor will check mental alertness, coordination, balance, and the existence of signs and symptoms in the physical exam. [30]

Succeeding diagnosis test will be performed. One is brain computed tomography (CT), which is the use of X-rays to capture detailed images of the brain. Another is magnetic resonance imaging, which is the use of magnets and radio waves to capture pictures of body organs and detect damage to brain cells and tissues. CT arteriogram and magnetic resonance arteriogram can also be performed to view large vessels of the brain.Carotid ultrasound, which is the use of sound waves; and carotid angiography, which is the use of dye and special X-rays; display the insides of carotid arteries; which supply oxygen-rich blood to the brain. [30]

4. Complications

Stroke may lead to temporary or permanent disabilities. Complications of stroke include increased pressure on the brain and brain swelling; fever, which may be a sign of infection, like pneumonia or urinary tract infection; high blood sugar, usually to people with diabetes; high blood pressure; gathering of spinal fluid within the brain; blood vessel spasms; pulmonary embolism or blood clot in the lungs, seizures; another stroke; and coma. [31]

5. Initial Treatment

Initial treatment varies on the type of stroke. In ischemic stroke, recovering the blood flow to the brain is the main focus. Initial treatment for this involves the medication of tissue plasminogen activator, which promotes clot-bursting. It is injected in the vein of ones arm. It is given within four hours when symptoms are noticed. Antiplatelet medicine, such as aspirin and clopidogrel, may also be given to prevent the clumping of platelets. Other procedures include carotid endarterectomy and carotid artery angioplasty, which open covered carotid arteries. [32]

In hemorrhagic stroke, the goal is to control the blood vessel bleeding. Appropriate treatment for this, such as decompression or hydrocephalus drainage, may be administered. [33] Surgeries include aneurysm clipping, which obstructs aneurysm from the blood vessels in the brain; coil embolization, which is the use of a catheter into an artery in the groin to yield blood clots that block the leaking blood flow; and arteriovenous malformation repair, which fixes malfunction arteries and veins in the brain. [32]

6. Long-Term Treatment

(a) **Medications**

The taking of prescribed medicines should be achieved and never be discontinued without consulting the appropriate doctor. Follow-up with the doctor is also advised in viewing the medication and its effects. Medicines that may be prescribed are anticoagulant medications, including heparin or warfarin and low-dose aspirin; citicoline, which treats ischemic stroke; epoetin, which is a synthetic version of human erythropoietin, for treating ischemic stroke; and the application of magnesium to serve as a potential neuroprotective agent. [33]

(b) **Stages of Stroke Recovery**

There are seven distinguished stages of stroke recovery. The transition of

stages is the progress that comes about for the patient. This framework, also called the Brunnstrom Approach, focuses on spastic and involuntary muscle movements of the patient. [34] These stages are enumerated as follows.

i. First, there will be flaccidity or no voluntary muscle movements.During this stage, the patient has no voluntary muscle movements in the areas affected by the stroke. Typically, a patient will lose motor function in their legs, arms, feet, or hands.

ii. Then, the patient retakes a small number of motor function but still no discernible synergy or muscle collaboration patterns. Muscles begin to make small, spastic, and abnormal movements during this stage.

iii. Third, spasticity in muscles increases and reaches its peak. Synergy patterns will appear, as well as voluntary movements.

iv. During stage four, spastic muscle movement begins to decline. The patient will also begin to regain a significant amount of motor control in the affected extremities. In addition, he/she will start making normal, controlled movements on a limited basis.

v. Spasticity continues to decline in stage five. Synergy patterns within the muscles also become more coordinated, and voluntary movements begin to become more complex.

vi. At stage six, spasticity in muscle movement disappears completely. The patient is now able to move individual joints, and synergy patterns become much more coordinated. Motor control is almost fully restored, and complex reaching movements in the affected extremities can now be coordinated.

vii. Lastly, the normal muscle functions arise. The patient is now able to move his/her arms, legs, hands, and feet in a controlled and voluntary manner.

(c) **Rehabilitation**

Rehabilitation for stroke patients involves physical, occupational, and speech therapy. In physical therapy, daily stretching exercises, muscle group strengthening exercises, and a sort of motion exercises are presented to the patient, with the help of physical therapists. In occupational therapy, helpful equipment and devices are used in splints, casts, or braces on the influenced arm or leg to allow proper limb positioning, avoid stiffness in joints, and maintain flexibility and range of motion. In speech therapy, the patient is assisted in his or her speaking, reading, writing, and understanding. [33]

(d) **Lifestyle Changes**

A doctor may provide changes in ones lifestyle. One way is to have a heart-healthy eating, which is comprised of fat-free or low-fat dairy products, fish rich in omega-3 fatty acids, fruits, vegetables, and whole grains. Other goals are lowering blood pressure, maintaining a healthy weight, managing stress, doing physical activity, and quitting smoking. [32]

B. Philippine Neurological Association

The Philippine Neurological Association (PNA) is an organization internationally acclaimed as one of the leading neurological associations in Asia in implementing quality neurological care, education, and research. It is committed to a progressive and proactive role in the promotion of health and well-being of the Filipino people, particularly in neurological care. It was founded by a group of neurologists, lead by Dr. Alfredo Bengzon, in 1973 and Dr. Gilberto Gamez was elected as its president. It was then accepted by the Philippine Medical Association as one of its component affiliate specialty societies in 1974. [7]

Since then, PNA has conducted research and teaching of neurology and represented the Philippines in the global community of neurological associations. Research has been administered in various accredited training institutions and different hospitals. They have studied neurological diseases, such as cerebrovas-

cular diseases, infectious diseases, neuromuscular diseases, and epilepsy. Indeed, they have organized 24 annual scientific conventions since 1978 and they had already hosted two congresses of Asian and Oceanic Congress of Neurology. They have also ensured excellence in the quality of neurological care and education and encouraged neurology practice in the country. [35]

PNA has also performed various services to the public. They have managed PNA Outreach Programs since 1982. In these programs, they have provided neurological services and education in different provinces every year. They have collaborated with provincial health offices to have regular neurology clinic for the public, teach healthcare workers every two months about the fundamental aspects and prevention of neurological diseases. They also have conducted conventions and raised awareness about different topics, such as acute stroke care, pharmacogenomics, and dementia. In addition, they had initiated Neuropathology Caravans, which aim to discuss a difficult neurologic case supported by pathologic diagnosis. [35]

The association also promote several advocacies. They had commenced and organized Neurology Week that focus on teaching people about neurology and improving healthcare. It was previously conducted in many SM Supermalls. Start Them Young is also a project initiated in 2010 that intends to educate people about the nervous system, its diseases, and the prevention of these diseases. [35]

C. Stroke Society of the Philippines

The Stroke Society of the Philippines (SSP) is an association that targets the reduction the incidence of stroke, reduction morbidity and mortality from stroke, and improving the quality of life of those who suffered from stroke. It was founded on July 13, 1995 and its Board of Directors, led by Dr. Joven Cuanang, was inaugurated at the EDSA Plaza Hotel on July 20, 1995. [8]

The main objectives of the SSP are to collect and disseminate information concerning the symptoms, diagnosis, treatment, management, and prevention of

stroke and other cerebrovascular conditions and diseases; to serve the public interest by undertaking and assisting in pure or basic and applied scientific research in the field of stroke and cerebrovascular diseases; and to initiate, stimulate, encourage, develop, support, assist and manage programs, projects and activities for the prevention of strokes and other related diseases. They also aim to undertake, support, or assist projects and community development work, training, and assistance in the prevention and treatment of stroke and the improvement of the quality of life of victims of stroke. [8]

They had published guidelines for the prevention, treatment, and rehabilitation of stroke intended for medical professionals and caregivers. [36] They also host the Brain Attack Awareness Week, an awareness campaign for stroke prevention, every third week of August every year. Likewise, they have launched a Stroke Research Contest, which aims to encourage doctors to study further treatment modalities and develop surgical methods to improve stroke treatment in the country. [9]

D. Web Portal

A web portal is a website specifically designed to gather information together from different sources and create a single point of access of these information. It also facilitates interaction among its users to share information through messages, forum threads, and discussion boards. Furthermore, users of the web portal can post or link information in various formats, like images, videos, and articles. Other users can also search for information contributed in the web portal. [37]

E. Patient Medical Records

Patient medical records are documentations of medical and healthcare services provided to patients during the supervision of healthcare providers. A medical record includes medical observations to the patient, recording observations and application of drugs and therapies, orders for the application of drugs and therapies, diagnosis test results, laboratory results, surgical results, x-rays, reports, and

other documentations. It allows physicians to provide quality health care to their patients by providing a detailed description of patients health status and other medical information. With this in mind, it is required for the healthcare professionals to keep complete and accurate medical records and preserve its veracity. It can also be in the form of paper records, electronic records, or both. [38]

IV. Design and Implementation

A. Context Diagram

The Philippine Stroke Portal (PSP) will be implemented as described in the succeeding diagrams. The web portal will have eight types of user, namely the patient or caregiver, medical professional, supervising medical professional, nutritionist, rehabilitation medicine doctor, local administrator, system administrator, and guest user. Each user will have different restrictions on accessibility and modification of information in the portal. The system administrator will be responsible for managing the portal and assuring that the portal and its components will be available for usage. Figure 1 shows the context diagram of the portal.

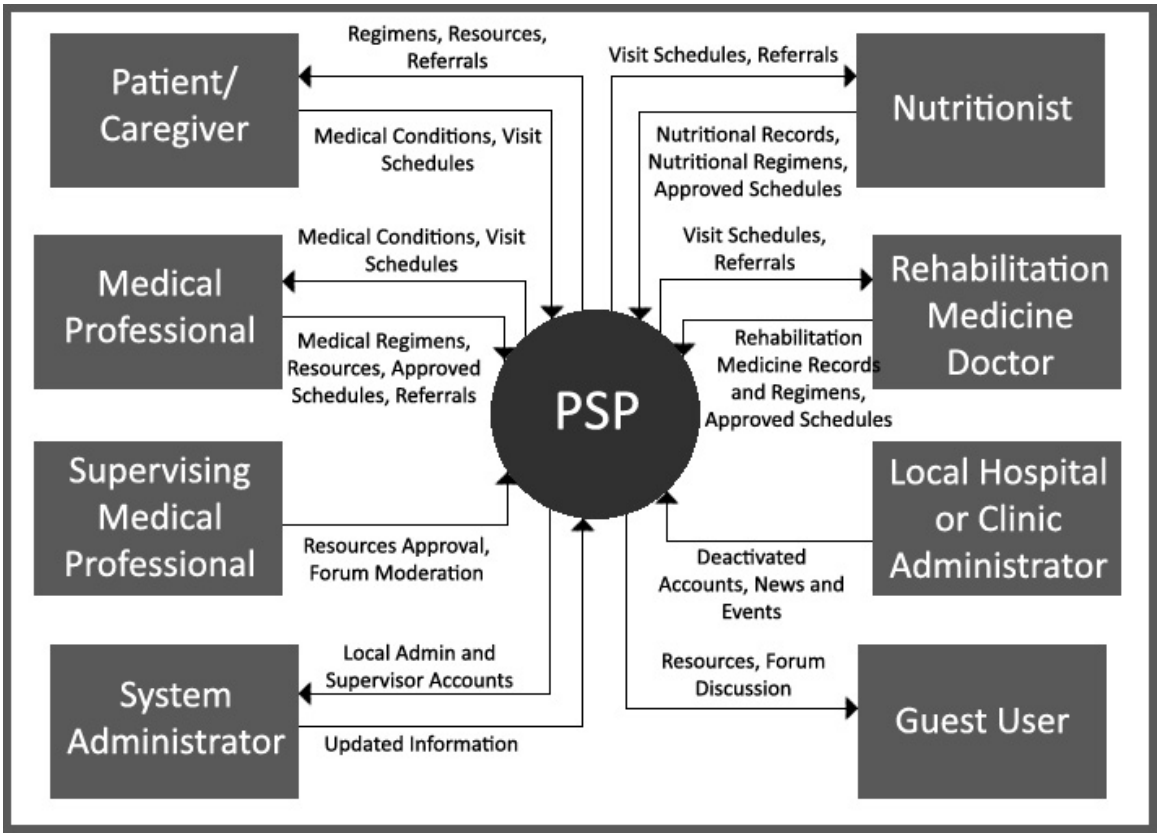

Figure 1: Context Diagram of PSP

B. Use Case Diagram

There are eight types of users in the web portal. A patient or caregiver can input patient records and conditions, set visit schedules, participate in forums, view regimens, resources, and referrals. A medical professional can register a staff account; add regimens, resources, and referrals; approve visit schedules; participate in forums; and view patient records. The supervising medical professional can register a supervisor account, approve medical resources, and moderate forums. Both the nutritionist and rehabilitation medicine can register a staff account, add regimens, approve visit schedules, participate in forums, and view patient records. The local administrator can register a supervisor account, manage accounts of the local staff, and post news and events. The system administrator can manage accounts of the local administrators and supervisors. The guest user can view the medical resources and forums. Figure 2 shows the use case diagram of the portal.

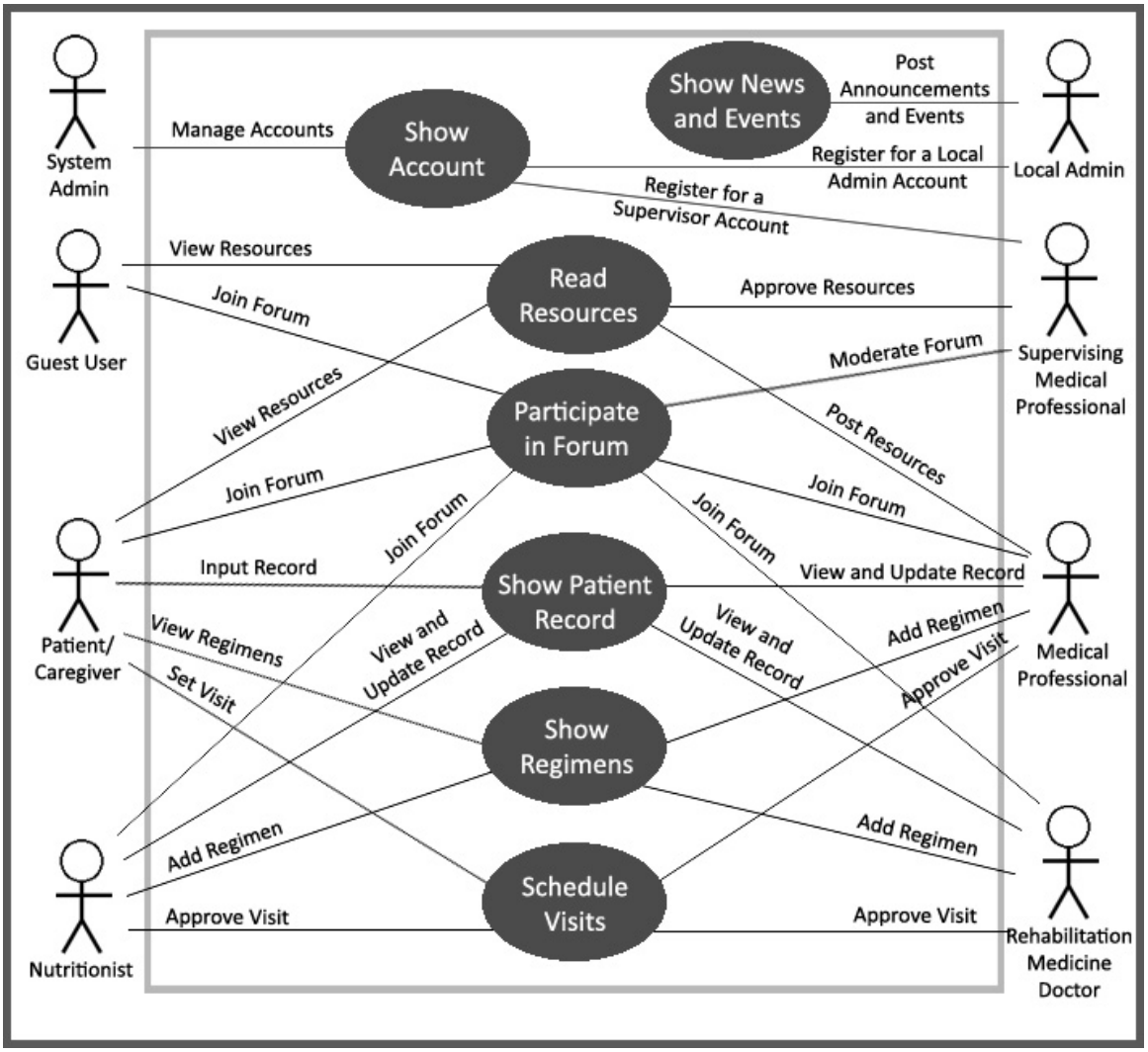

Figure 2: Use Case Diagram of PSP

C. Dataflow Diagram

The Data Flow Diagram, displayed in Figure 3, shows the functionalities of the system. The patient or caregiver can input medical record, view regimens, set visits, and message in forums. The health professional can view patient record, give regimens, approve visits, reply in forum threads, and post medical resources. The supervisor can moderate forum threads and approve resources. The local administrator and system administrator can manage staff accounts and supervisor accounts, respectively.

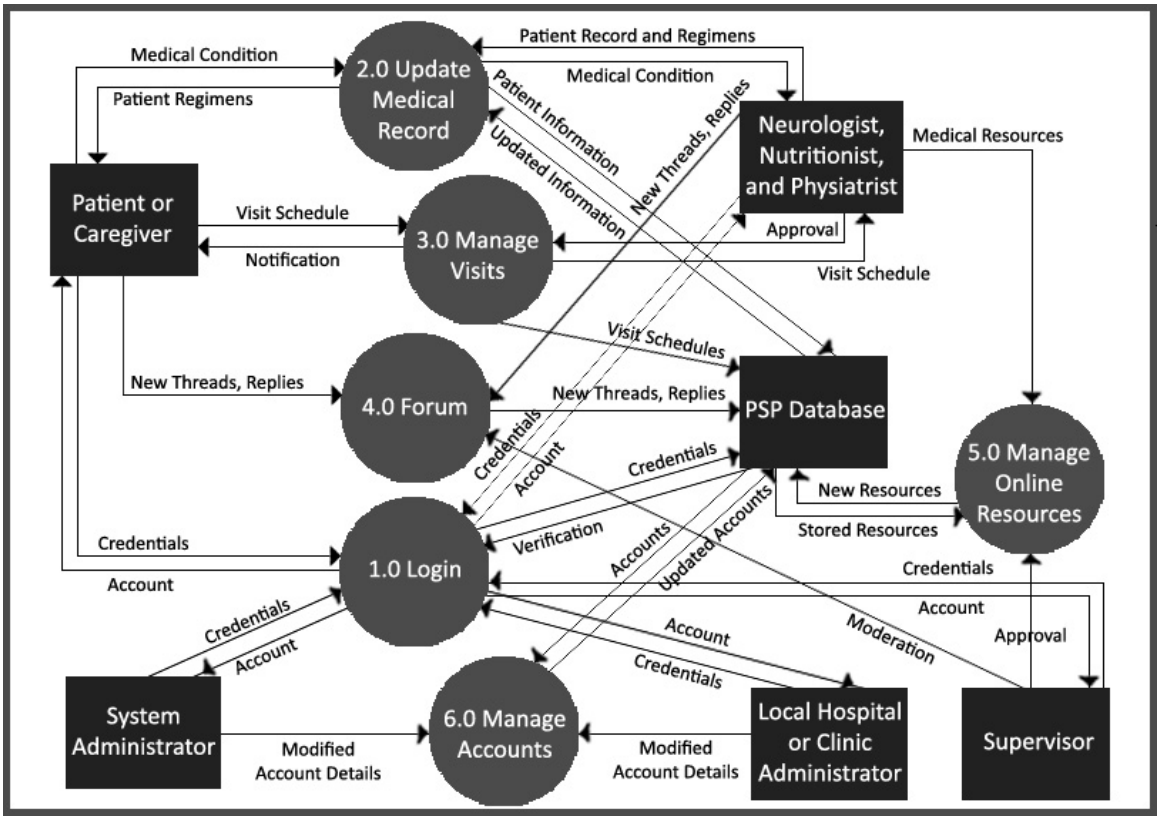

Figure 3: Top Level Data Flow Diagram of PSP

The functionalities of the users in the web portal are illustrated in the succeeding figures. The login function of users is display in Figure 4. The handling of daily medical conditions, patient medical records, and patient regimens is displayed in Figure 5. The managing of the scheduled follow-up visits in neurology, nutrition, or rehabilitation medicine clinic is illustrated in Figure 6. The implementation of forum threads is shown in Figure 7. In addition, the managing of online medical resources is displayed in Figure 8. Lastly, the managing of user accounts by the local hospital or clinic administrator and system administrator is illustrated in Figure 9.

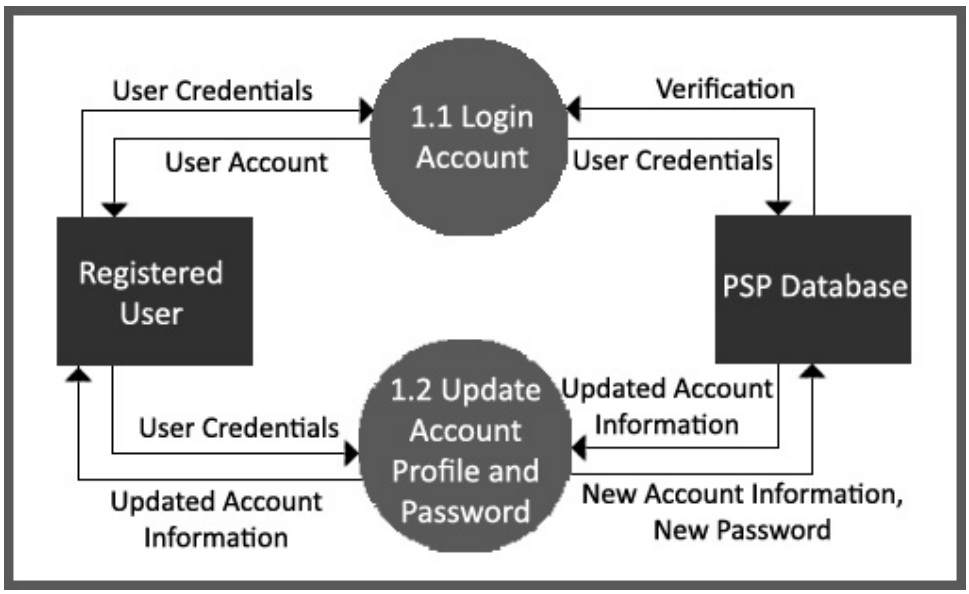

Figure 4: Sub-explosion for Login Function of PSP

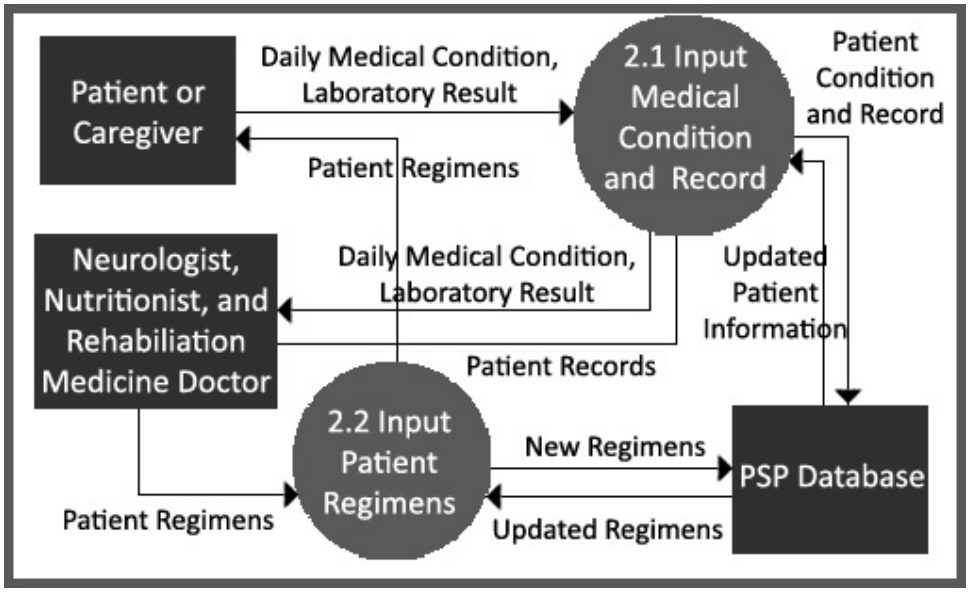

Figure 5: Sub-explosion for Input Medical Condition, Record, and Regimen Function of PSP

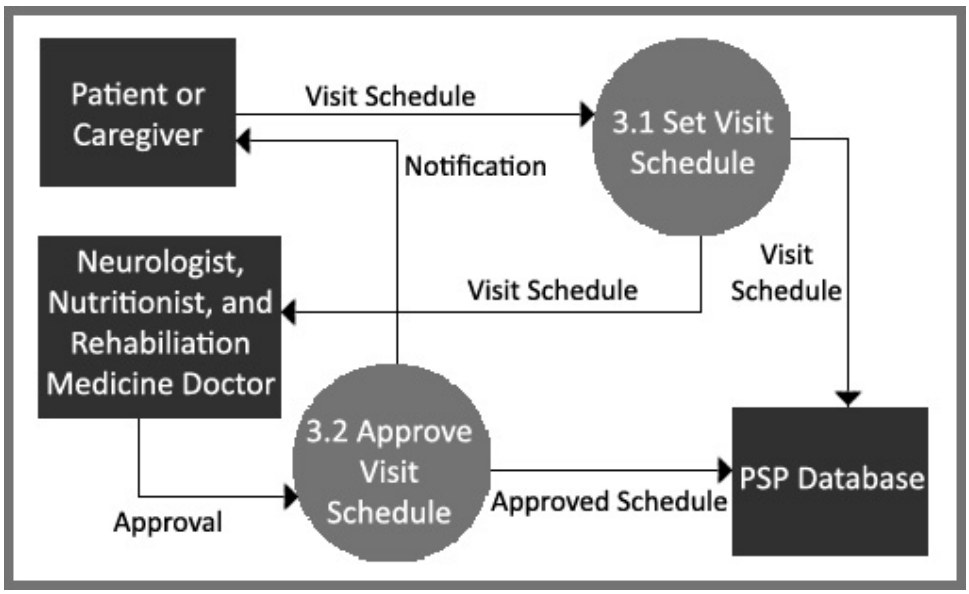

Figure 6: Sub-explosion for Manage Visits Function of PSP

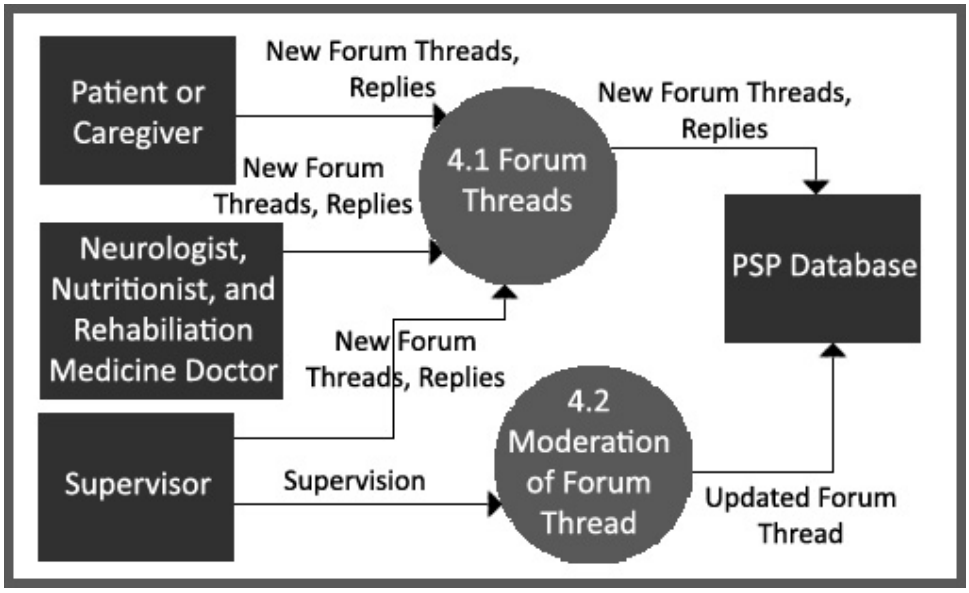

Figure 7: Sub-explosion for Forum Thread Function of PSP

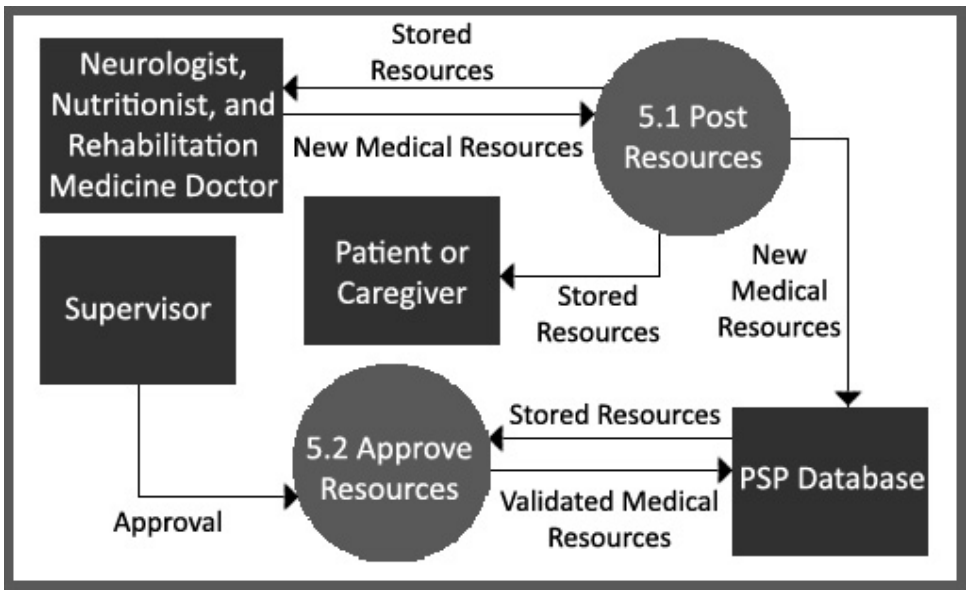

Figure 8: Sub-explosion for Manage Online Medical Resources Function of PSP

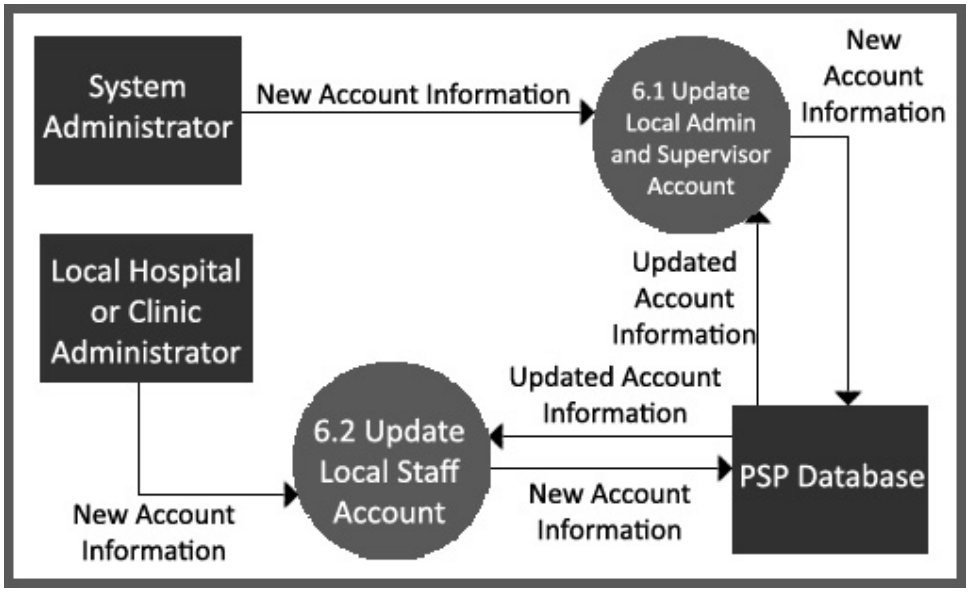

Figure 9: Sub-explosion for Manage Accounts Function of PSP

D. Entity Relationship Diagram

A database is necessary for the storage of all the user accounts, information, and online resources. This will also enable the saving of the forum threads, patient condition, records, and regimens. The Entity Relationship Diagram is displayed on Figure 10.

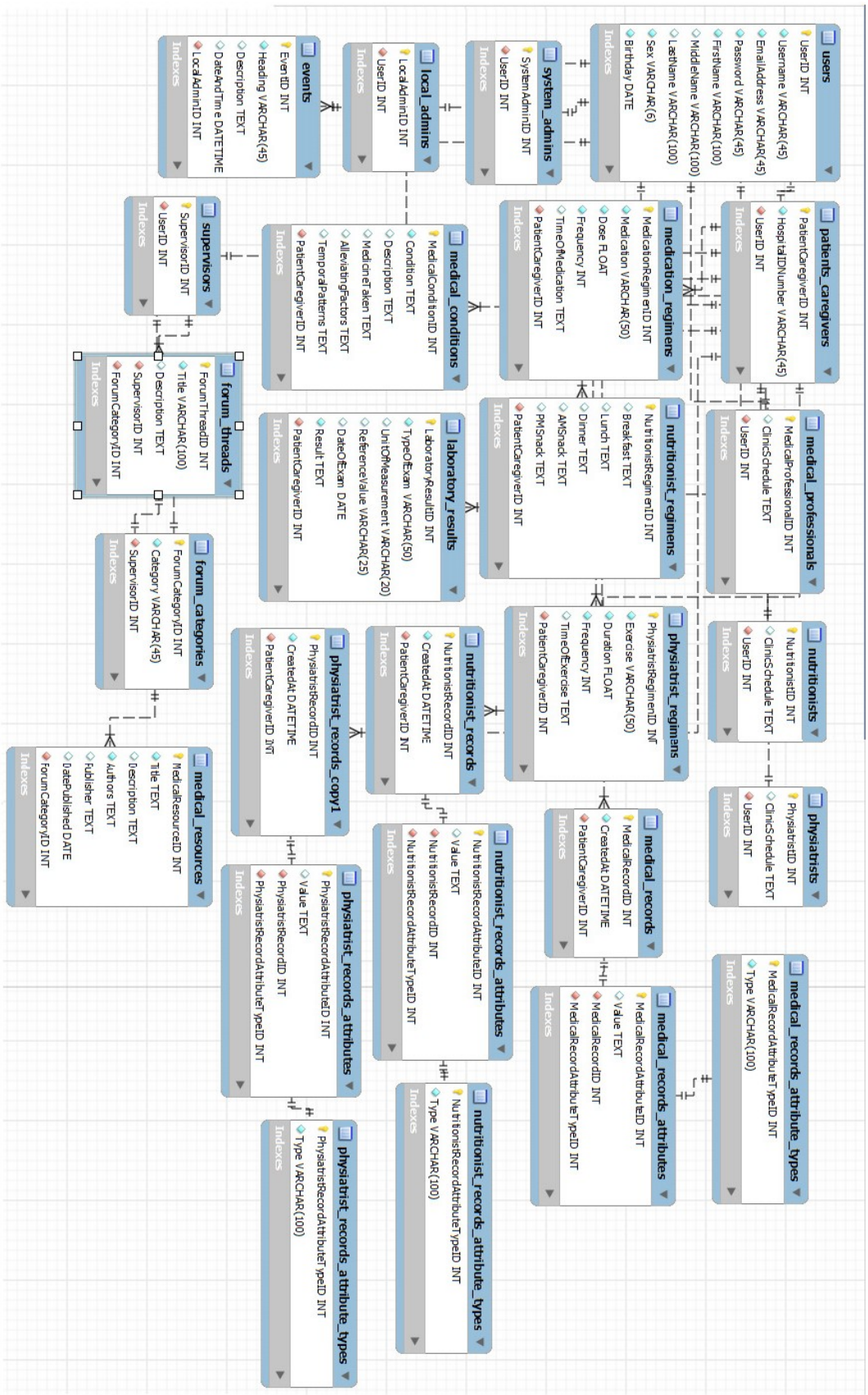

Figure 10: Entity Relationship Diagram of PSP

E. Database Design

The Philippine Stroke Portal will have one database. This will contain thirty eight (38) tables. Nine (9) of these tables will handle user accounts and its various classifications (e.g., patients, medical professionals). Four (4) will handle the records of hospitals and clinics, as well as the professions of the users in the hospitals. Thirteen (13) will handle the patient information and records, particularly medical, nutritional, and rehabilitation medicine records. Three (3) will handle the recommended regimens for the patient. Three (3) will handle the forum threads, categories, and replies. Two (2) will handle the set schedules for follow-up visits and decisions on medical conditions. The remaining four (4) tables will handle the announcements and events, medical resources, uploaded files, and notifications to users in the portal. The data dictionary is elaborated in the following tables.

Attribute	**Data Type**	**Description**
id	int	Unique ID of the user table
username	varchar(50)	Unique username of the user
email	varchar(100)	Unique e-mail address of the user
firstName	varchar(100)	First name of the user
middleName	varchar(100)	First name of the user
lastName	varchar(100)	First name of the user
sex	varchar(6)	Sex of the user
birthday	date	Birthday of the user
isUnlocked	boolean	To check if the user account is deactivated or not
isEmailConfirmed	boolean	To check if the e-mail address of the user is confirmed or not
confirmationCode	varchar(30)	Confirmation code of the e-mail address

Table 1: Users table

Attribute	Data Type	Description
email	varchar(100)	E-mail address where the password reset permission is sent
token	varchar(30)	Token of the password reset

Table 2: Password resets table

Attribute	Data Type	Description
id	int	Unique ID of the patient or caregiver
hospitalIDNumber	varchar(100)	Hospital ID number of the patient
caregiverFirstName	varchar(100)	First name of the caregiver
caregiverMiddleName	varchar(100)	Middle name of the caregiver
caregiverLastName	varchar(100)	Last name of the caregiver
userID	int	Foreign key to the users table
hospitalID	int	Foreign key to the hospitals table
medicalProfessionalID	int	Foreign key to the medical professionals table
nutritionistID	int	Foreign key to the nutritionists table
physiatristID	int	Foreign key to the physiatrists table

Table 3: Patients table

Attribute	Data Type	Description
id	int	Unique ID of the medical professional
specialties	varchar(100)	Specialty of the medical professional
clinicSchedule	text	Clinic schedule of the medical professional
userID	int	Foreign key to the users table

Table 4: Medical professionals table

Attribute	Data Type	Description
id	int	Unique ID of the nutritionist
clinicSchedule	text	Clinic schedule of the nutritionist
userID	int	Foreign key to the users table

Table 5: Nutritionists table

Attribute	Data Type	Description
id	int	Unique ID of the physiatrist
clinicSchedule	text	Clinic schedule of the physiatrist
userID	int	Foreign key to the users table

Table 6: Physiatrists table

Attribute	Data Type	Description
id	int	Unique ID of the supervisor
userID	int	Foreign key to the users table
forumCategoryID	int	Foreign key to the forum categories table

Table 7: Supervisors table

Attribute	Data Type	Description
id	int	Unique ID of the local admin
userID	int	Foreign key to the users table

Table 8: Local admins table

Attribute	Data Type	Description
id	int	Unique ID of the system admin
userID	int	Foreign key to the users table

Table 9: System admins table

Attribute	Data Type	Description
id	int	Unique ID of the hospital
name	varchar(100)	Name of the hospital
address	text	Address of the hospital
landlineNumber	varchar(20)	Landline number of the hospital
mobileNumber	varchar(20)	Mobile number of the hospital
faxNumber	varchar(20)	Fax number of the hospital
emailAddress	varchar(30)	E-mail address of the hospital
isUnlocked	boolean	To check if the hospital is locked or unlocked by the system admin

Table 10: Hospitals table

Attribute	Data Type	Description
id	int	Unique ID of the hospital medical profession
hospitalID	int	Foreign key to the hospitals table
medicalProfessionalID	int	Foreign key to the medical professionals table
isApproved	boolean	To check if the profession is approved or not by the local admin

Table 11: Hospital medical professions table

Attribute	Data Type	Description
id	int	Unique ID of the hospital nutritionist profession
hospitalID	int	Foreign key to the hospitals table
nutritionistID	int	Foreign key to the nutritionists table
isApproved	boolean	To check if the profession is approved or not by the local admin

Table 12: Hospital nutritionist professions table

Attribute	Data Type	Description
id	int	Unique ID of the hospital physiatrist profession
hospitalID	int	Foreign key to the hospitals table
physiatristID	int	Foreign key to the physiatrists table
isApproved	boolean	To check if the profession is approved or not by the local admin

Table 13: Hospital physiatrist professions table

Attribute	Data Type	Description
id	int	Unique ID of the medical condition
dailyMedicalProblem	text	The daily medical problem of the patient
describeTheSymptom	text	The description of the problem
medicineTaken	text	The over-the-counter medicines taken to alleviate the problem
effectsNoted	text	The effects noted from the medicines
anyManeuverTaken	text	The maneuvers taken to alleviate the problem
patientID	int	Foreign key to the patients table

Table 14: Medical conditions table

Attribute	Data Type	Description
id	int	Unique ID of the laboratory result
typeOfLaboratory Exam	varchar(30)	The type of laboratory exam
unitOfMeasurement	varchar(30)	The unit of measurement of laboratory exam
referenceValues	varchar(30)	The reference values of laboratory exam
dateOfTest	date	The date of laboratory exam
result	text	The result of laboratory exam
patientID	int	Foreign key to the patients table

Table 15: Laboratory results table

Attribute	Data Type	Description
id	int	Unique ID of the patient attribute
value	text	The value of patient attribute or field
patientID	int	Foreign key to the patients table
patientAttribute TypeID	int	Foreign key to the patient attribute types table

Table 16: Patient attributes table

Attribute	Data Type	Description
id	int	Unique ID of the patient attribute type
type	varchar(255)	The name or type of attribute or field

Table 17: Patient attribute types table

Attribute	Data Type	Description
id	int	Unique ID of the medication record
createdAt	datetime	Date and time when the medication record is created
patientID	int	Foreign key to the patients table

Table 18: Medication records table

Attribute	Data Type	Description
id	int	Unique ID of the medical record attribute
value	text	The value of medical attribute or field
medicationRecordID	int	Foreign key to the medication records table
medicalRecord AttributeTypeID	int	Foreign key to the medical record attribute types table

Table 19: Medical record attributes table

Attribute	Data Type	Description
id	int	Unique ID of the medical record attribute type
type	varchar(255)	The name or type of attribute or field

Table 20: Medical record attribute types table

Attribute	Data Type	Description
id	int	Unique ID of the nutritionist record
createdAt	datetime	Date and time when the nutritionist record is created
patientID	int	Foreign key to the patients table

Table 21: Nutritionist records table

Attribute	Data Type	Description
id	int	Unique ID of the nutritionist record attribute
value	text	The value of nutritionist attribute or field
nutritionistRecordID	int	Foreign key to the nutritionist records table
nutritionistRecord AttributeTypeID	int	Foreign key to the nutritionist record attribute types table

Table 22: Nutritionist record attributes table

Attribute	Data Type	Description
id	int	Unique ID of the nutritionist record attribute type
type	varchar(255)	The name or type of attribute or field

Table 23: Nutritionist record attribute types table

Attribute	Data Type	Description
id	int	Unique ID of the physiatrist record
createdAt	datetime	Date and time when the physiatrist record is created
patientID	int	Foreign key to the patients table

Table 24: Physiatrist records table

Attribute	Data Type	Description
id	int	Unique ID of the physiatrist record attribute
value	text	The value of physiatrist attribute or field
physiatristRecordID	int	Foreign key to the physiatrist records table
physiatristRecord AttributeTypeID	int	Foreign key to the physiatrist record attribute types table

Table 25: Physiatrist record attributes table

Attribute	Data Type	Description
id	int	Unique ID of the physiatrist record attribute type
type	varchar(255)	The name or type of attribute or field

Table 26: Physiatrist record attribute types table

Attribute	Data Type	Description
id	int	Unique ID of the medication regimen
medication	varchar(100)	The name of the medication regimen
dose	float	The dose of the medication regimen
frequency	int	The frequency of the medication regimen (number of times a day)
timeOfMedication	text	The time when medication regimen is to be taken
comments	text	The comments about the medication regimen
patientID	int	Foreign key to the patients table

Table 27: Medication regimens table

Attribute	Data Type	Description
id	int	Unique ID of the nutritionist regimen
breakfast	text	The option for breakfast
lunch	text	The option for lunch
dinner	text	The option for dinner
amSnack	text	The option for AM snack
pmSnack	text	The option for PM snack
patientID	int	Foreign key to the patients table

Table 28: Nutritionist regimens table

Attribute	Data Type	Description
id	int	Unique ID of the physiatrist regimen
typeOfExercise	varchar(100)	The type of exercise in the physiatrist regimen
duration	int	The duration of the exercise (in minutes)
frequency	int	The frequency of the physiatrist regimen (number of times a day)
timeOfExercise	text	The time when physiatrist regimen is to be performed
comments	text	The comments about the physiatrist regimen
patientID	int	Foreign key to the patients table

Table 29: Physiatrist regimens table

Attribute	Data Type	Description
id	int	Unique ID of the forum thread
title	varchar(100)	The title of the forum thread
description	text	The description of the forum thread
userID	int	Foreign key to the users table
supervisorID	int	Foreign key to the supervisors table
forumCategoryID	int	Foreign key to the forum categories table

Table 30: Forum threads table

Attribute	Data Type	Description
id	int	Unique ID of the forum category
name	varchar(50)	The name of the forum category

Table 31: Forum categories table

Attribute	Data Type	Description
id	int	Unique ID of the forum reply
comment	text	The reply to the forum thread
userID	int	Foreign key to the users table
forumID	int	Foreign key to the forum threads table

Table 32: Forum replies table

Attribute	Data Type	Description
id	int	Unique ID of the follow-up visit
dateOfVisit	date	The date of the visit
isApproved	boolean	To check if the visit is approved or not
userID	int	Foreign key to the users table (medical professional, nutritionist, and physiatrist)
patientID	int	Foreign key to the patients table
medicalConditionID	int	Foreign key to the medical conditions table

Table 33: Follow-up visits table

Attribute	Data Type	Description
id	int	Unique ID of the decision on condition
decision	text	The decision on the medical condition
additionalNotes	boolean	The additional notes regarding the decision
userID	int	Foreign key to the users table (medical professional, nutritionist, and physiatrist)
medical ProfessionalID	int	Foreign key to the medical professionals table
medicalConditionID	int	Foreign key to the medical conditions table

Table 34: Decision on medical conditions table

Attribute	Data Type	Description
id	int	Unique ID of the announcement and event
heading	varchar(100)	The heading of the announcement and event
description	boolean	The description of the announcement and event
dateOfEvent	date	The date of the announcement and event
timeOfEvent	time	The time of the announcement and event
hospitalID	int	Foreign key to the hospitals table

Table 35: Announcements and events table

Attribute	Data Type	Description
id	int	Unique ID of the medical resource
title	varchar(100)	The title of the medical resource
description	boolean	The description of the medical resource
authors	varchar(100)	The authors of the medical resource
publisher	varchar(100)	The publisher of the medical resource
datePublished	date	The date when the medical resource is published
isApproved	boolean	To check if the medical resource is approved or not
userID	int	Foreign key to the users table
forumCatergoryID	int	Foreign key to the forum categories table

Table 36: Medical resources table

Attribute	Data Type	Description
id	int	Unique ID of the file
filename	varchar(255)	The filename of the file
extension	varchar(5)	The file extension of the file
mimeType	varchar(255)	The mime type of the file
path	varchar(50)	The path to the file
userID	int	Foreign key to the users table
medicalResourceID	int	Foreign key to the medical resources table
laboratoryResultID	int	Foreign key to the laboratory results table

Table 37: Files table

Attribute	Data Type	Description
id	int	Unique ID of the notification
description	varchar(100)	The description of the notification
link	varchar(100)	The link to the notification
isSeen	boolean	To check if the user has seen the notification
userID	int	Foreign key to the users table

Table 38: Notifications table

F. System Architecture

The system will be implemented using PHP as the main programming language and Laravel 5 as the application framework. HTML, CSS, JavaScript, JQuery, and the Foundation 5 framework by Zurb will be used for interface design. SQLite version 3 will be used in the database layer. The system architecture is shown in Figure 11.

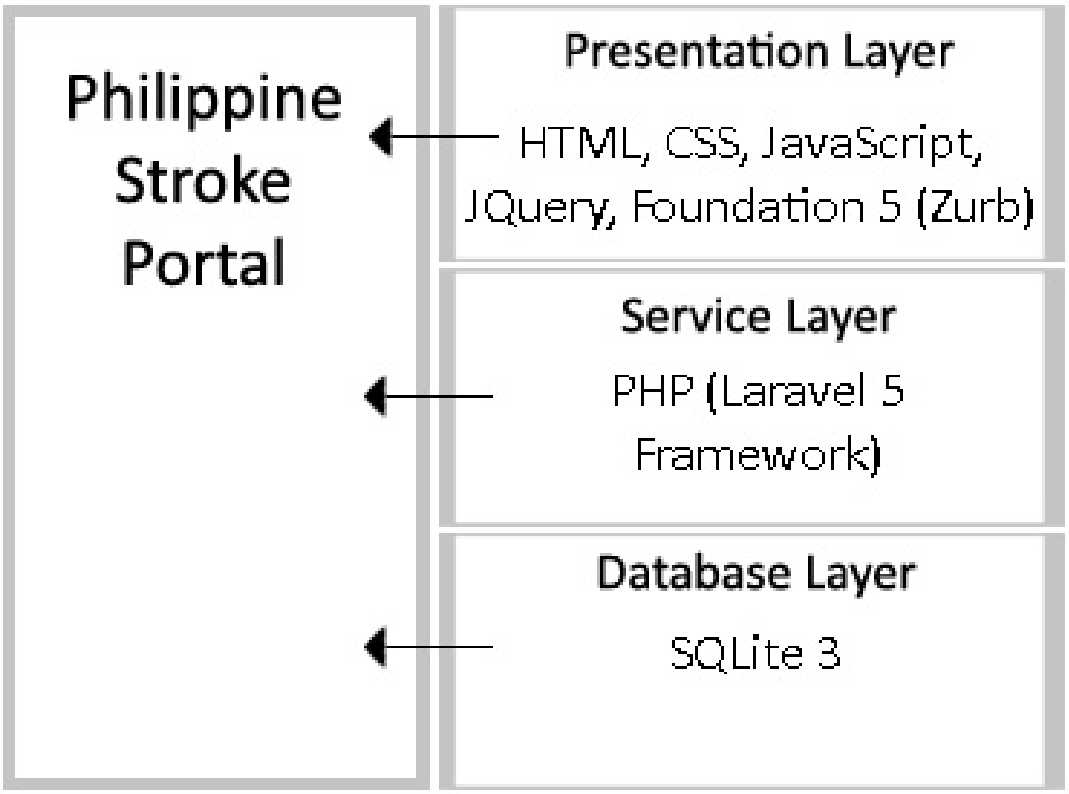

Figure 11: System Architecture of PSP

G. Technical Architecture

The minimum requirements for the server machine include:

- PHP version 5.6
- SQLite version 3
- 1 GB free disk space
- 2.4 MHz processor
- 500MB RAM

The minimum requirements for the client include:

- 1 GB RAM
- JavaScript-enabled browser

V. Results

1. General View

The home page of Philippine Stroke Portal is shown in Figure 12. The guest users will see the latest announcements and events of hospitals, newest medically validated resources on stroke, and latest forum threads. In addition, a user can view the full listing of announcements and events, medical resources, and forum threads by clicking the buttons in the menu on the left. Figures 13-15 show these listings of announcements and events, medical resources, and forum threads, respectively.

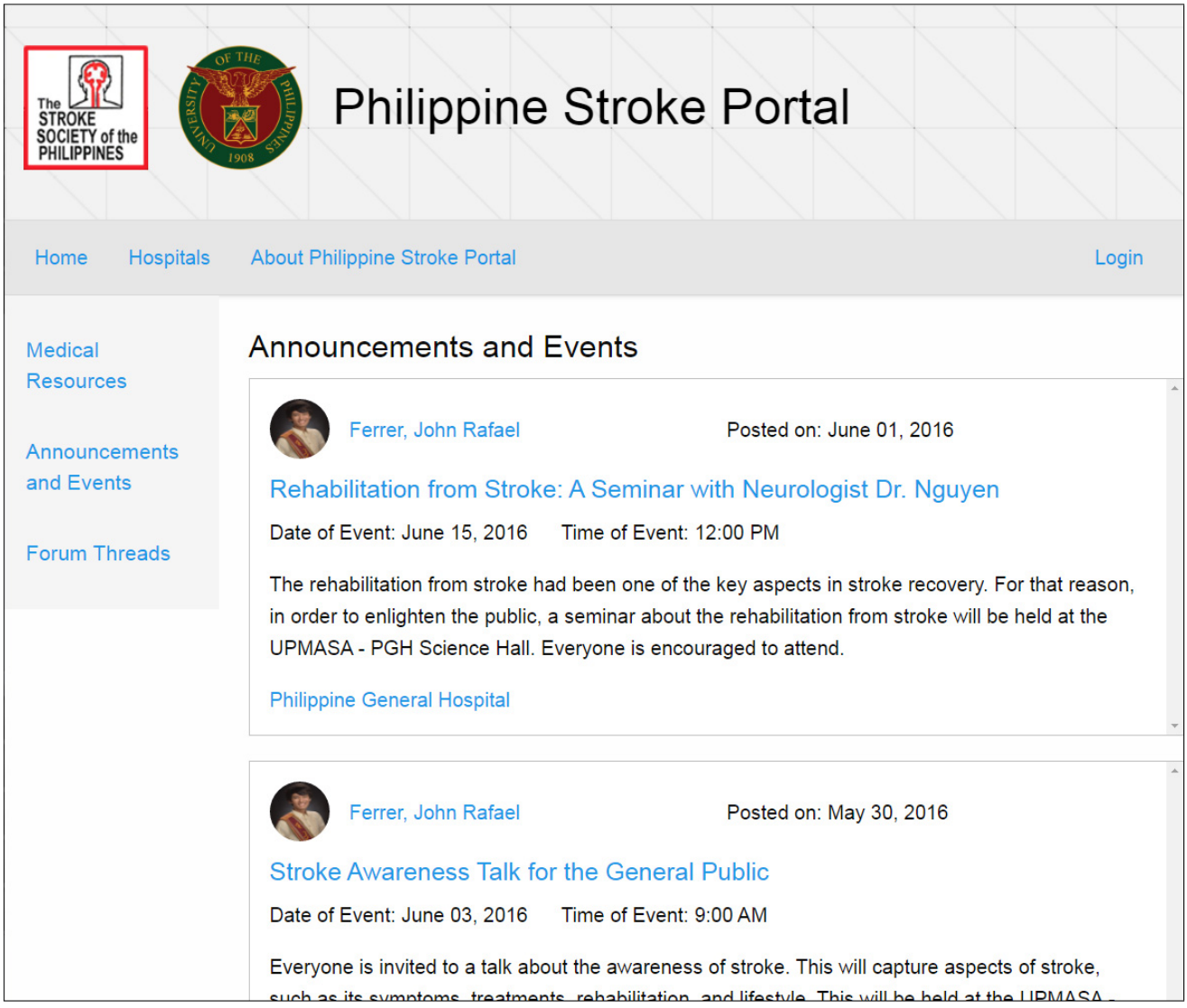

Figure 12: Home page of PSP

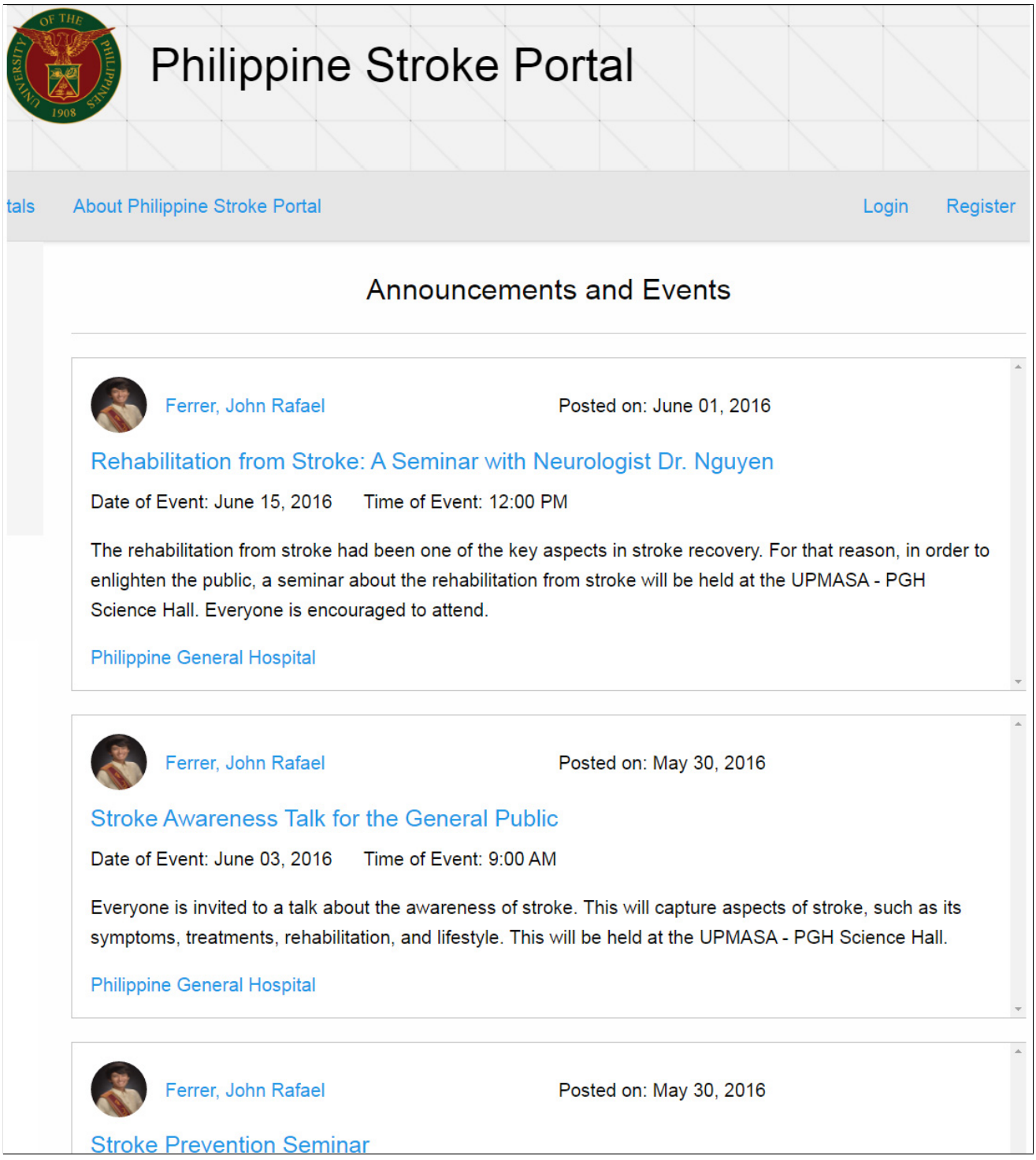

Figure 13: Index page of announcements and events of PSP

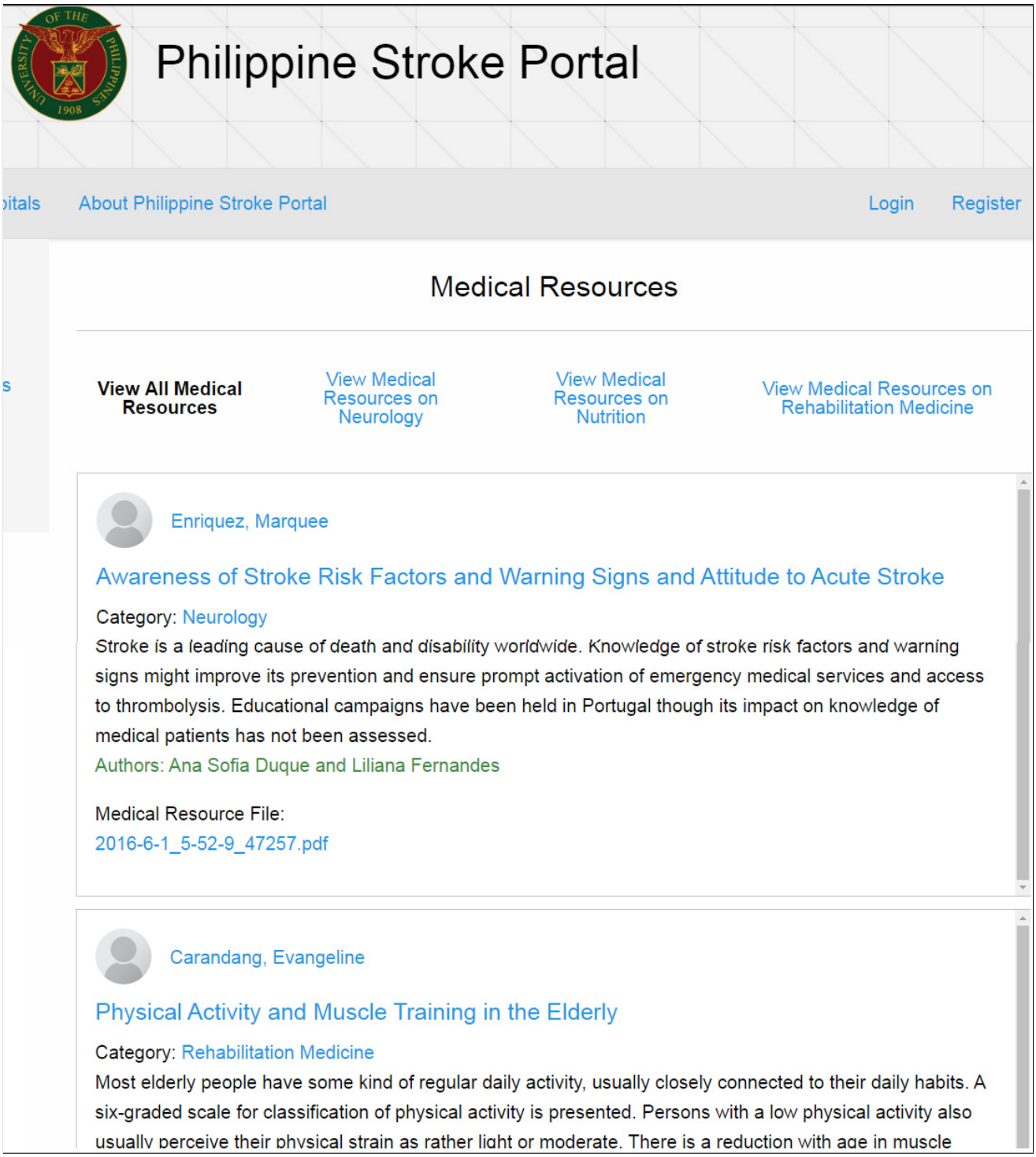

Figure 14: Index page of medical resources of PSP

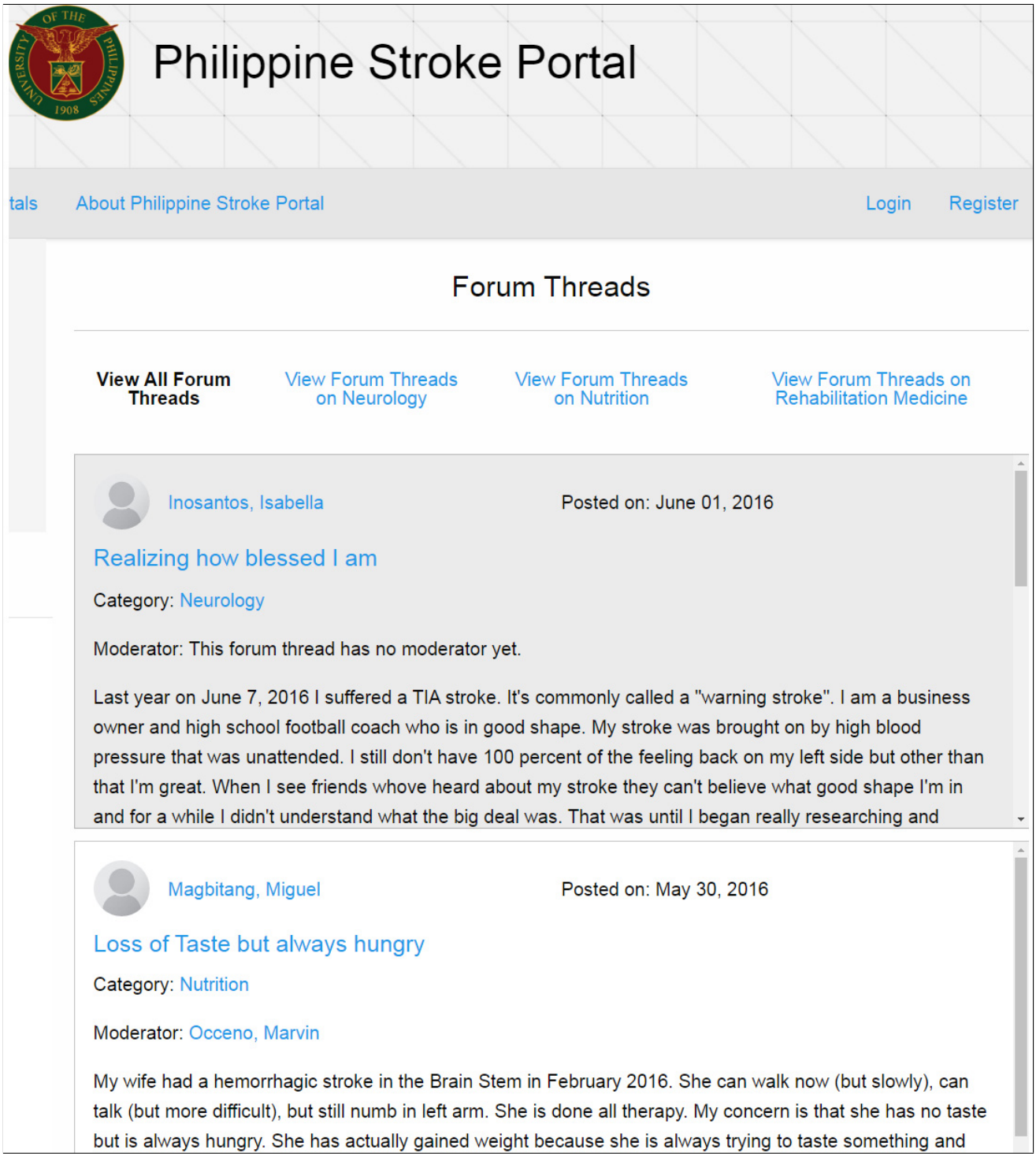

Figure 15: Index page of forum threads of PSP

The guest user can also view a medical resource and download it. Medical resource files include images, videos, Portable Document Format files, and other types of files. Figure 16 shows a user downloading a neurology resource file. Other types of users also have the same functionality of viewing and downloading medical resources.

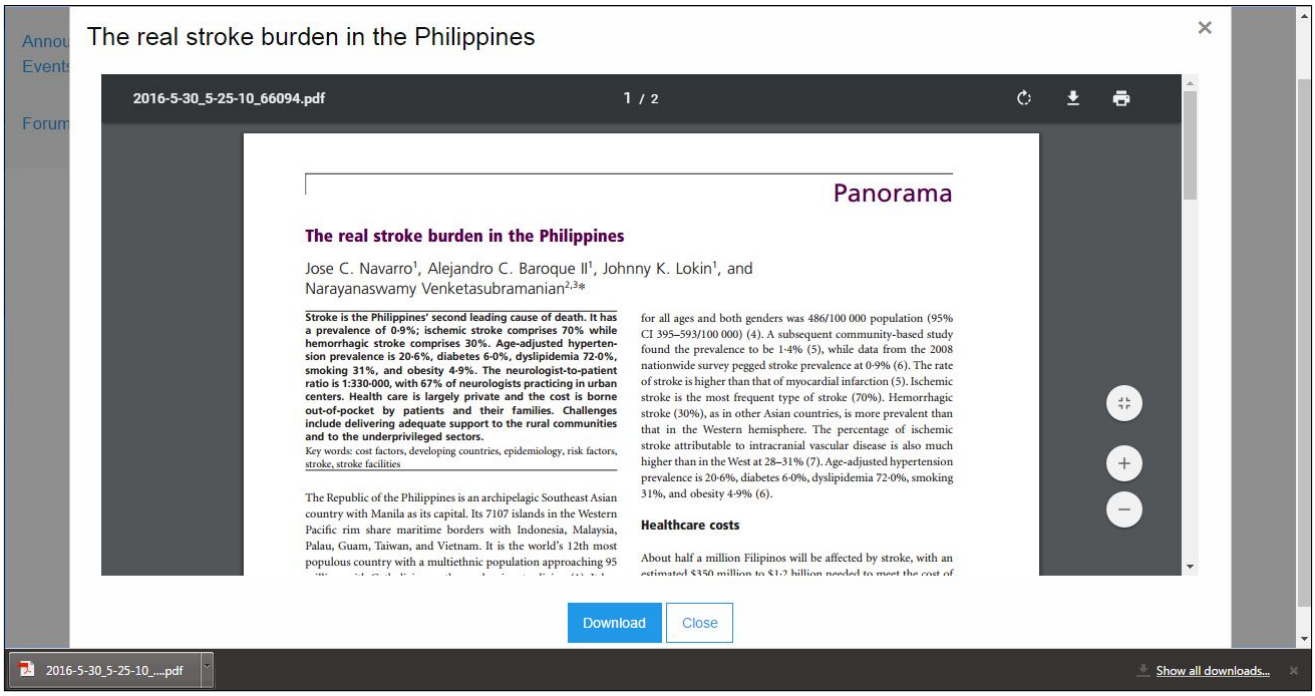

Figure 16: Download a medical resource file of PSP

The login page of PSP is seen in Figure 17. It also contains a link in the bottom of the form in case the user forgot his/her password. This is illustrated in Figure 18. The registration page of PSP in Figure 19 allows the medical professionals, nutritionists, and rehabilitation medicine doctors to register to PSP

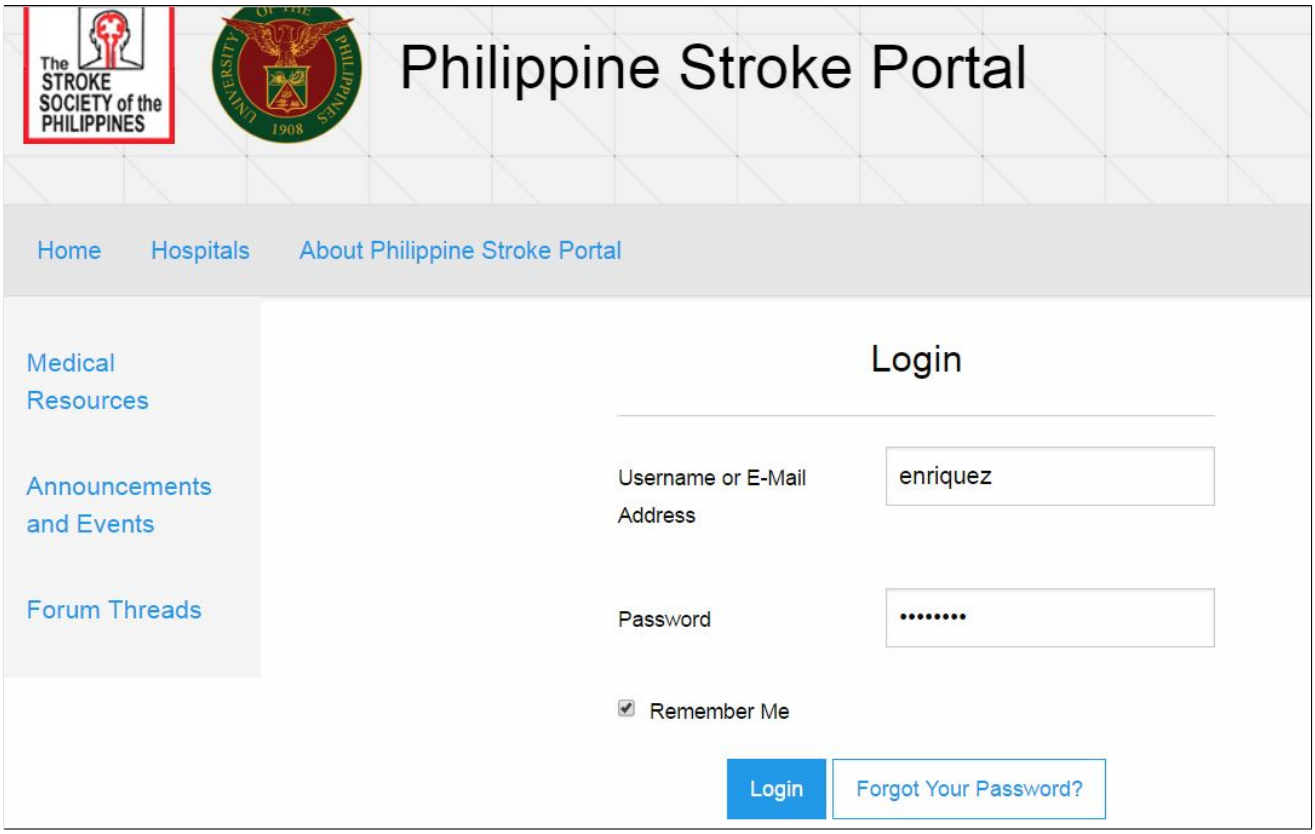

Figure 17: Login page of PSP

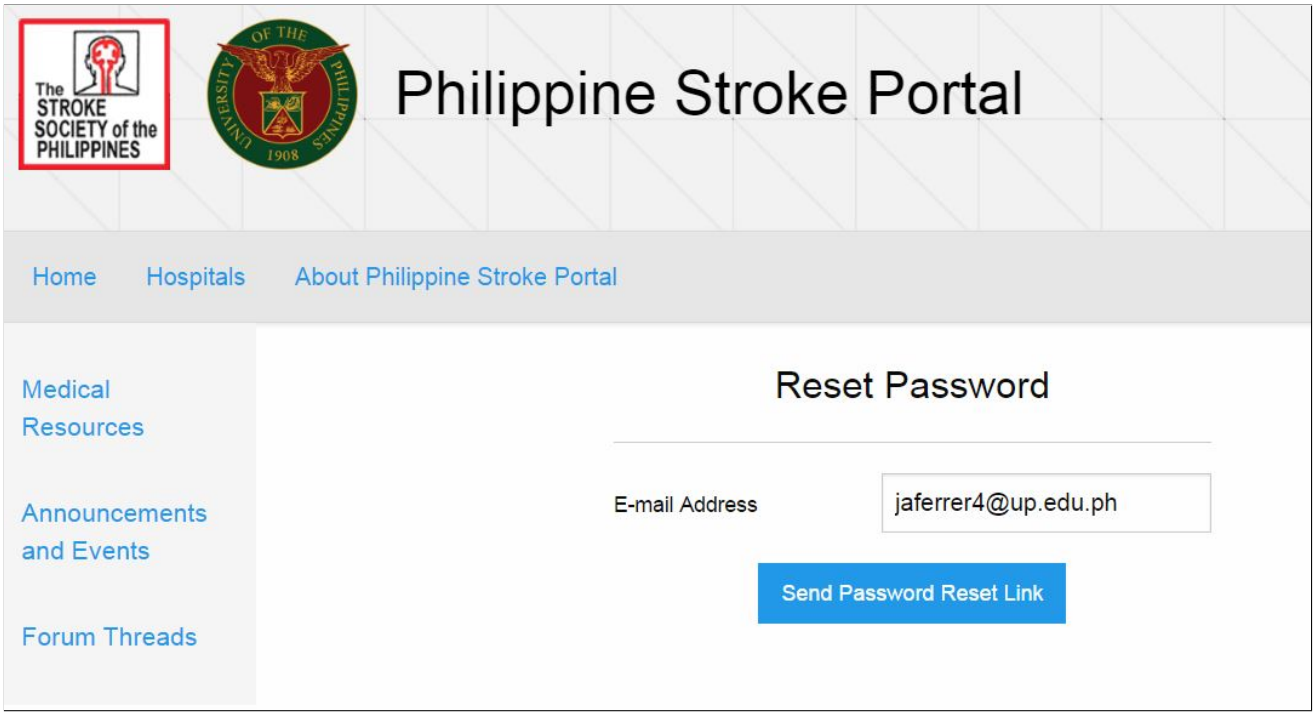

Figure 18: Forgot password page of PSP

Figure 19: Register page of PSP

2. **Patient View**

The patient's home page is similar to the one as seen in Figure 12. The only difference is the added menu buttons that emphasize the functionalities of the patient in the portal. The patient can also view his/her account profile by clicking the patient name at the top navigation bar. Figure 20 shows the account profile of a patient user. Any user of the portal can also update their account profile, profile picture, and password by the links above the account profile page. Figure 21 shows the edit page of an account profile. Figure 22 shows

the update page of profile picture, while Figure 23 shows the change page of password. Other users also have the same functionalities in terms of their account profile.

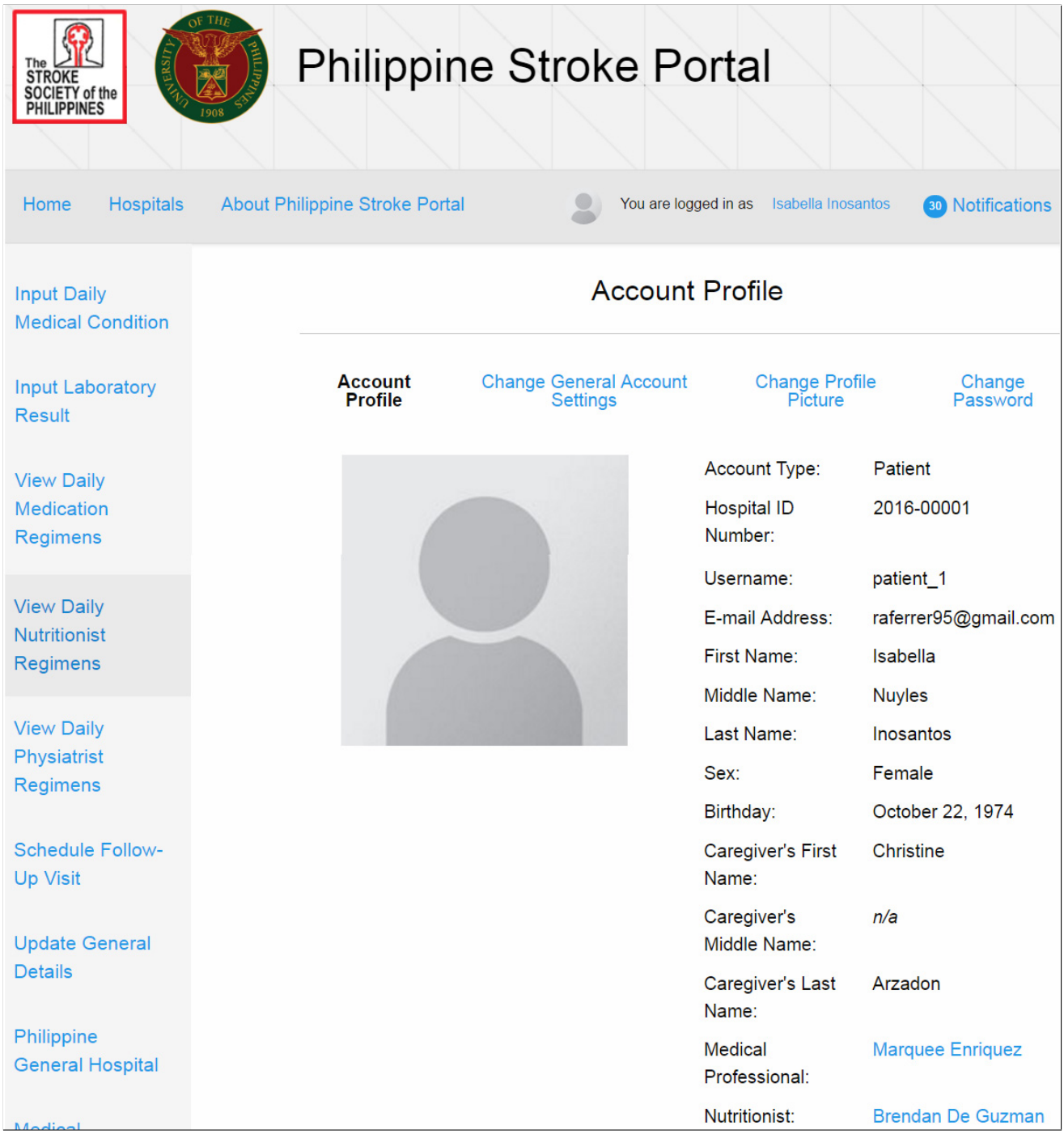

Figure 20: Account profile of a user of PSP.

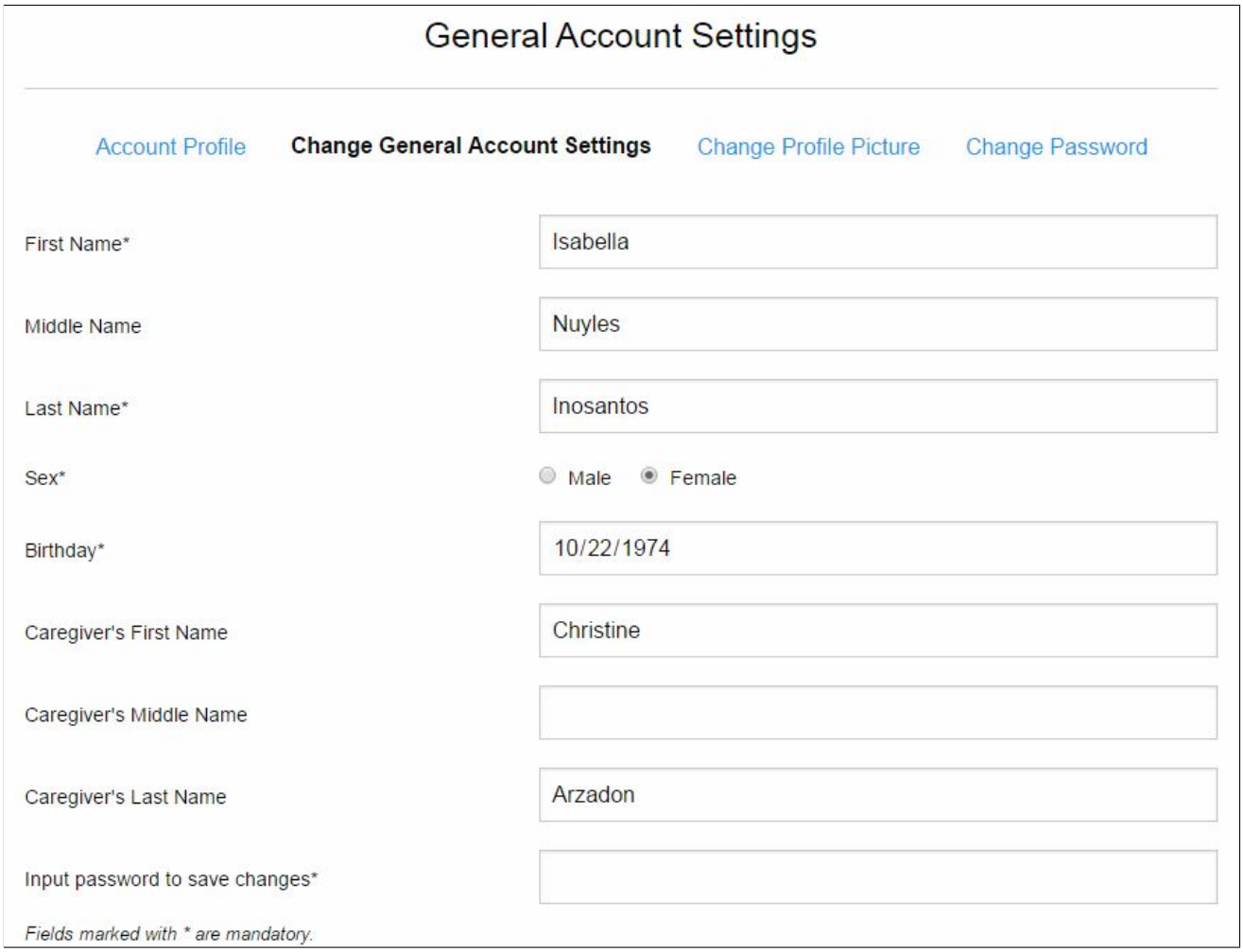

Figure 21: Edit page of account profile of a user of PSP.

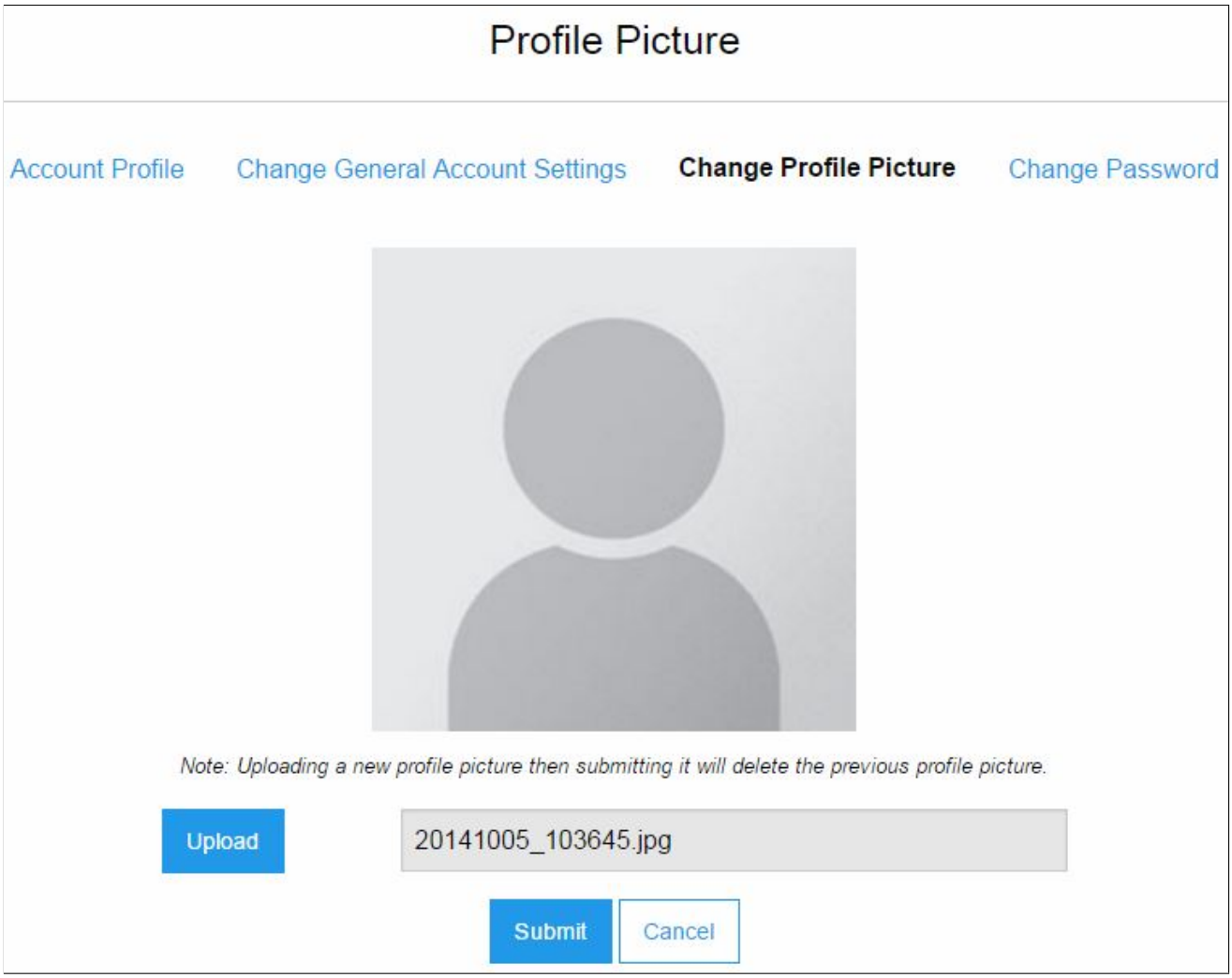

Figure 22: Upload page of profile picture of a user of PSP.

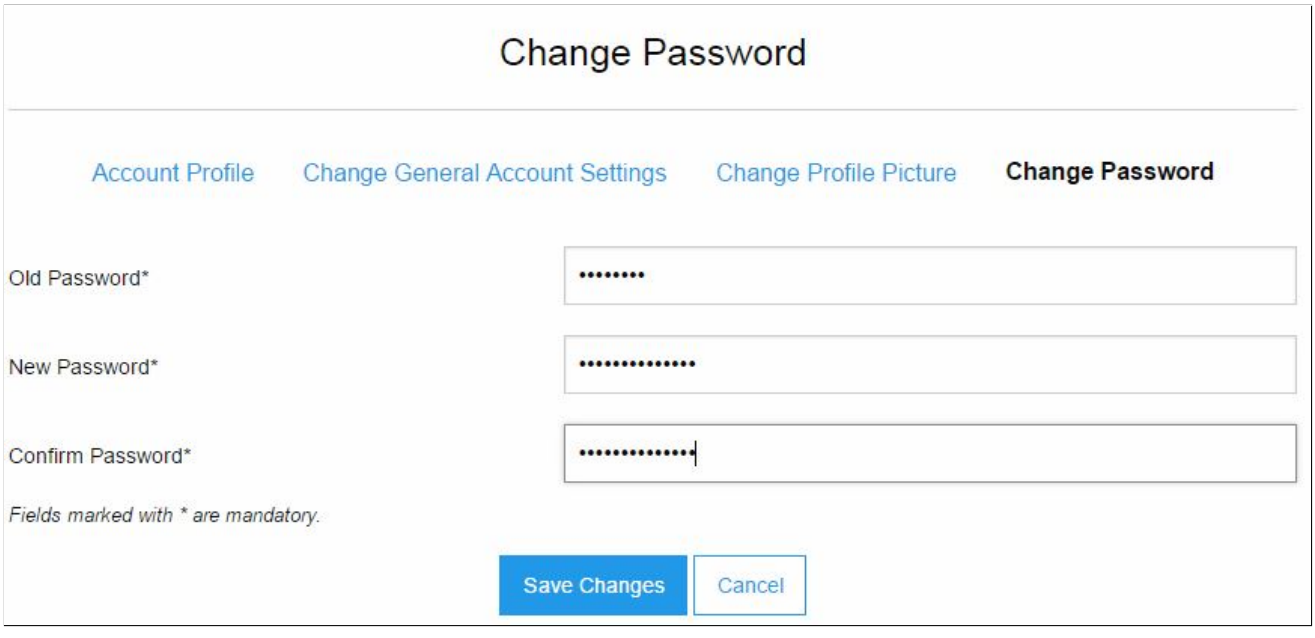

Figure 23: Change password page of PSP.

The patient can view the medication, nutrition, and rehabilitation medicine regimens suggested by the health professionals. This can be done by accessing the menu buttons on the left. Figures 24-26 show the viewing of medication, nutrition, and rehabilitation medicine regimens, respectively.

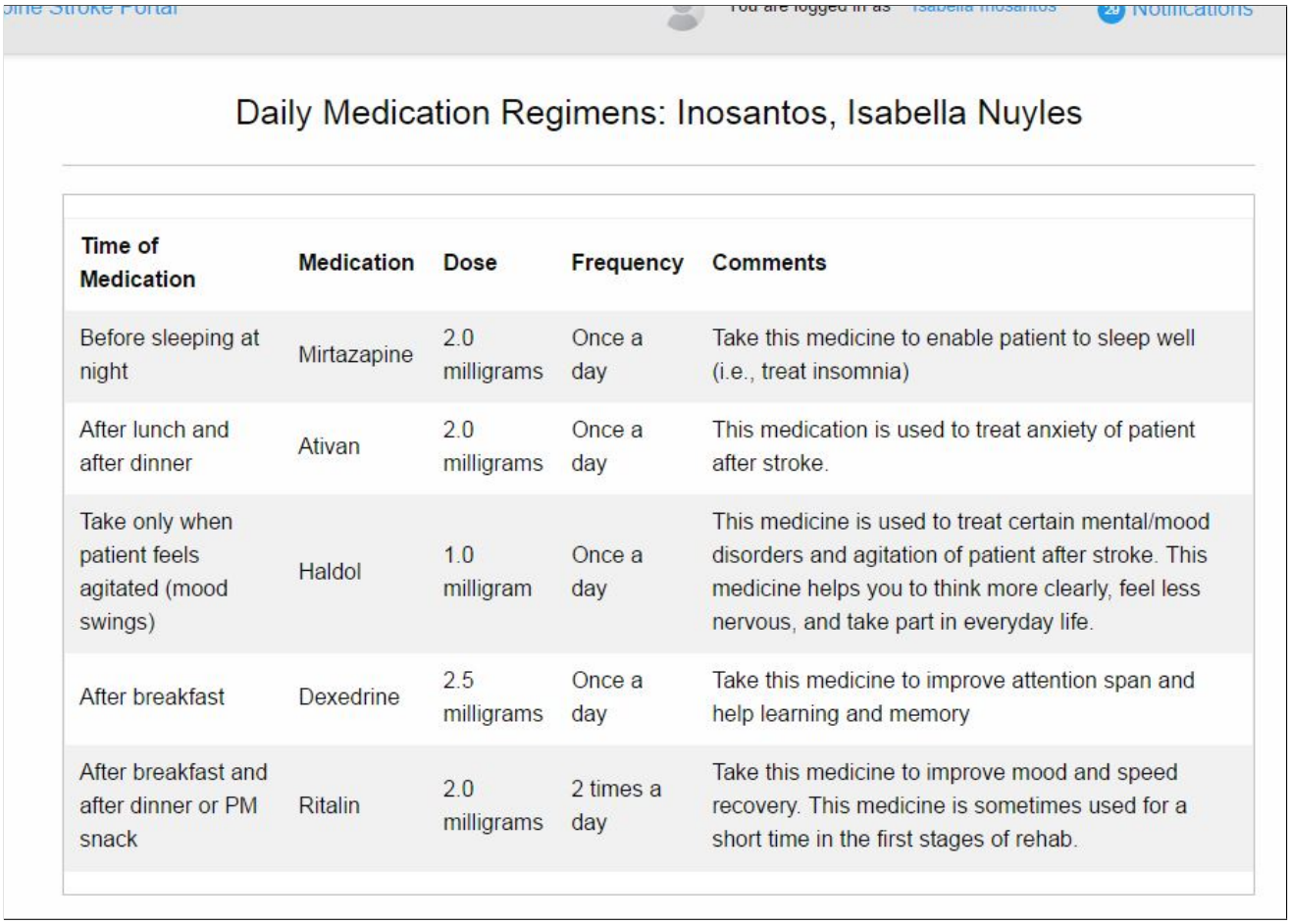

Figure 24: View medication regimen page of PSP.

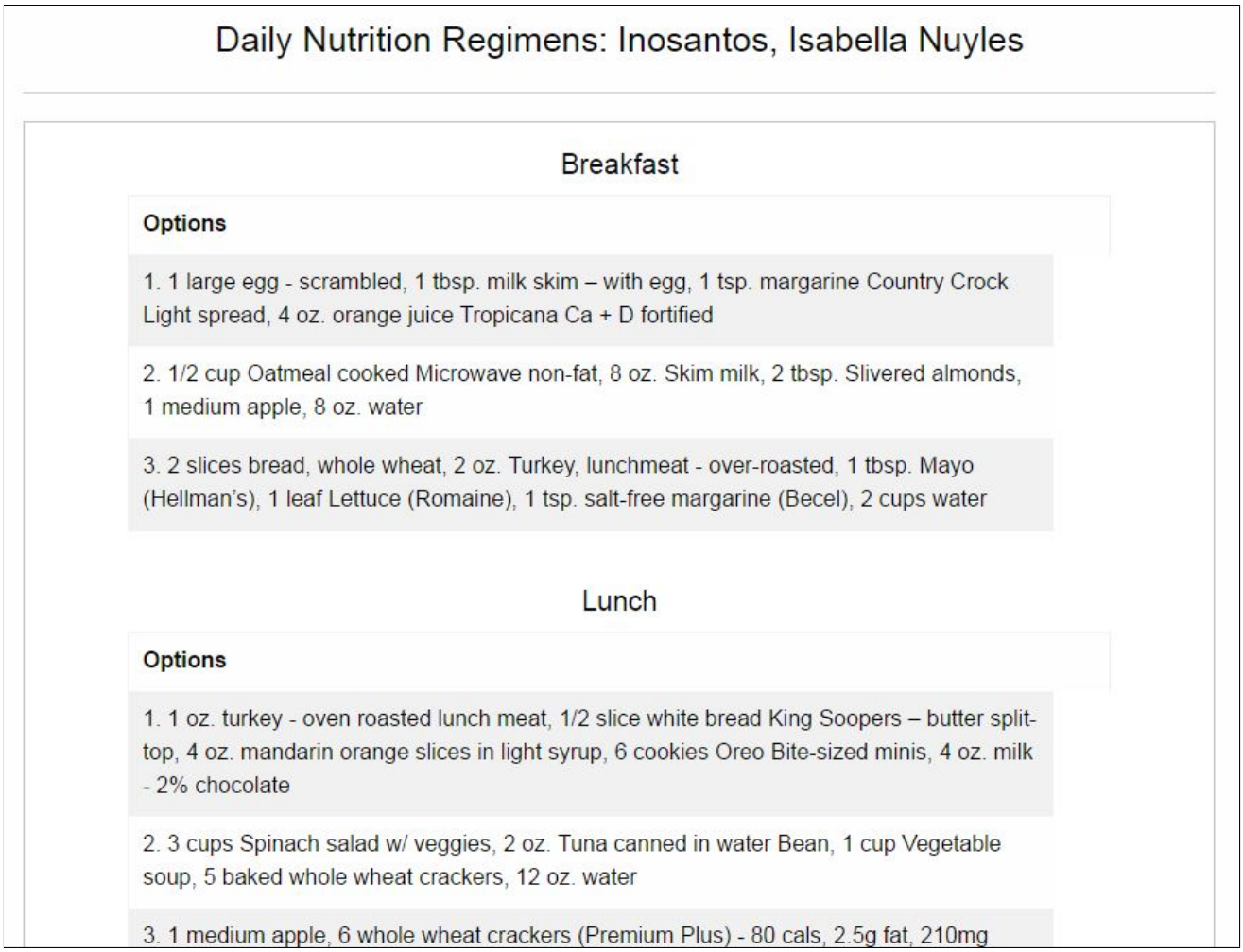

Figure 25: View nutrition regimen page of PSP.

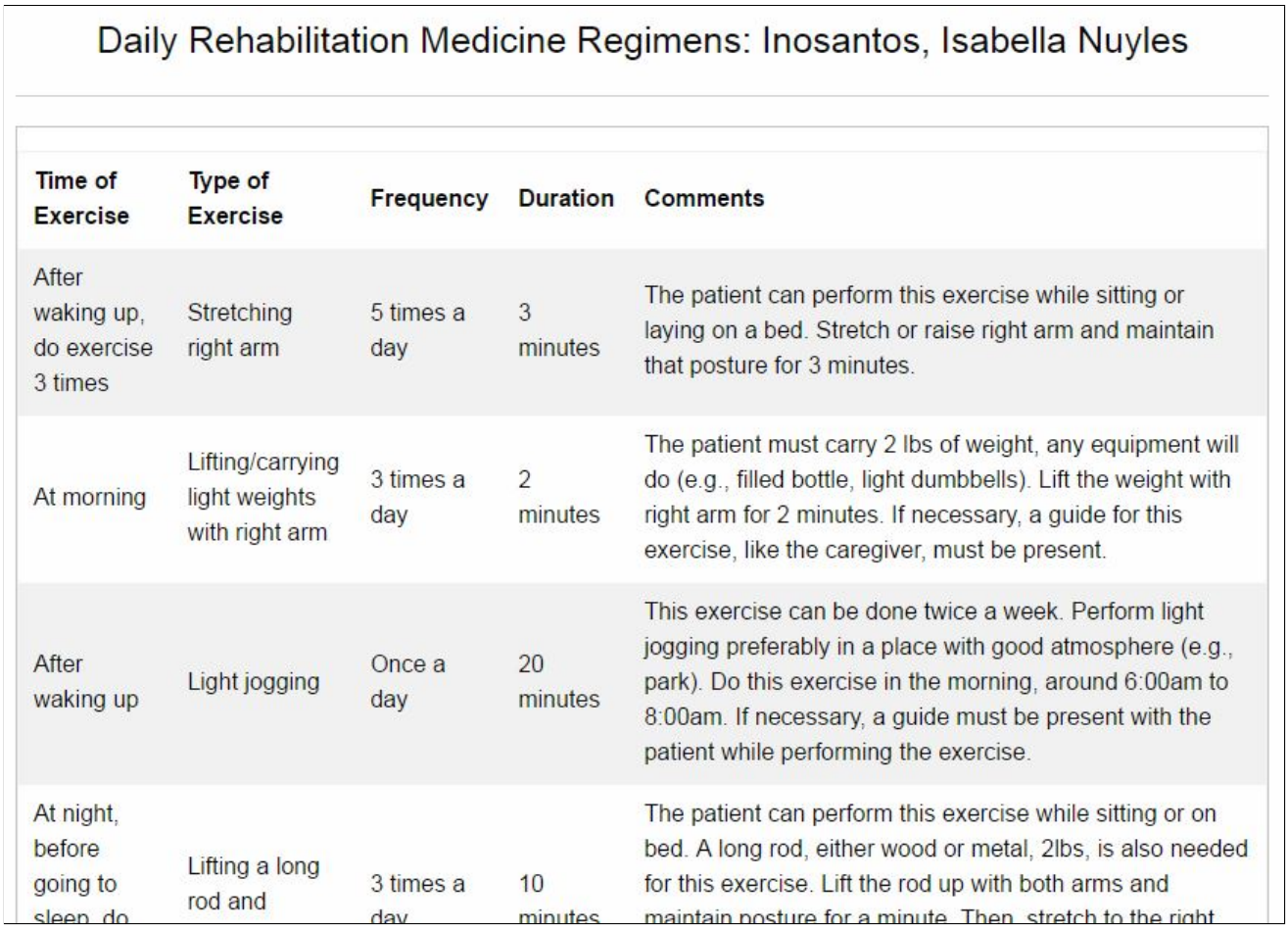

Figure 26: View rehabilitation medicine regimen page of PSP.

The patient can also input their daily medical condition and laboratory results done on outpatient basis. Figure 27 shows the page for inputting daily medical condition, while Figure 28 displays the page for inputting a laboratory result. The patient can also input the scan files of his/her laboratory result, seen in Figure 29.

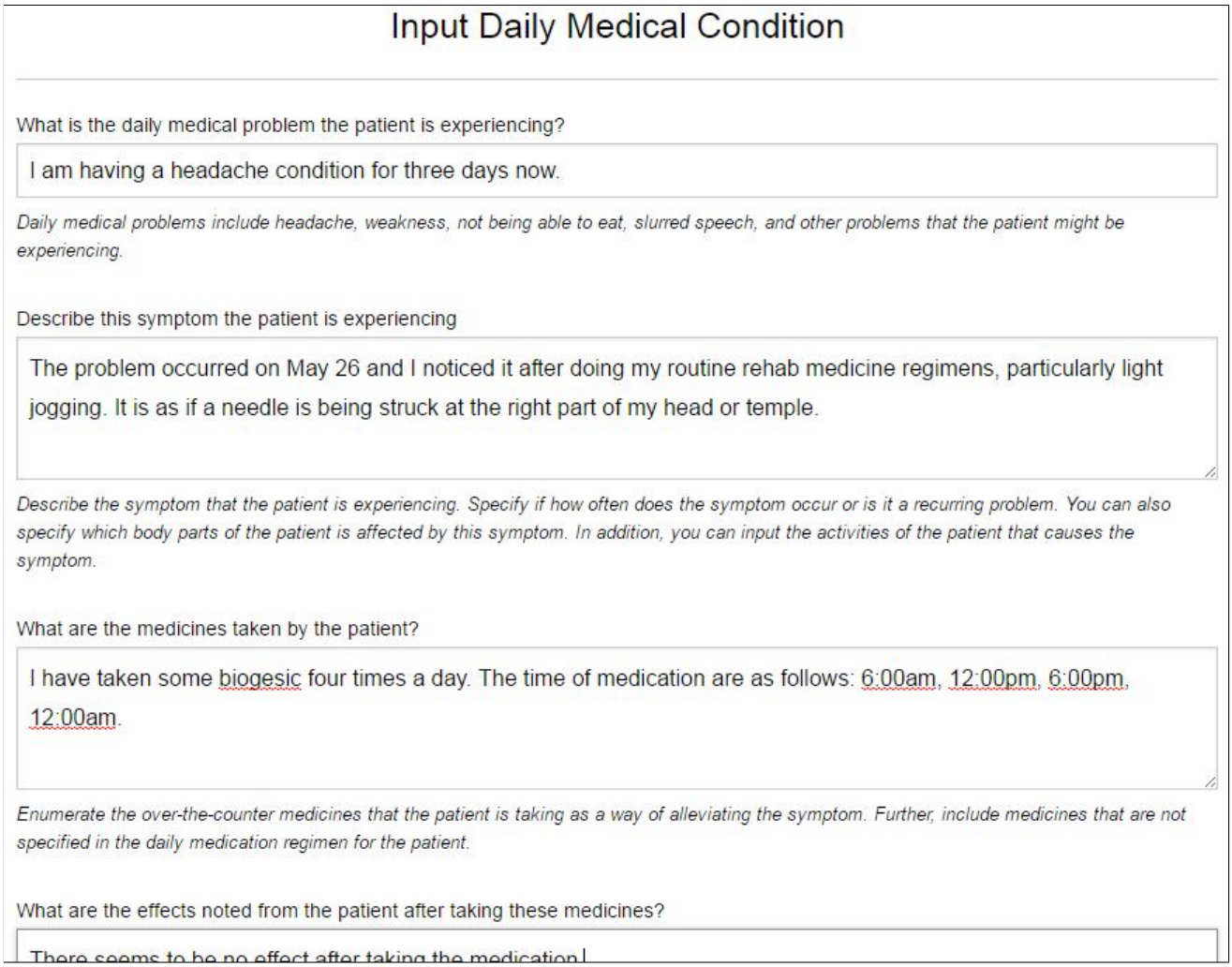

Figure 27: Input daily medical condition page of PSP.

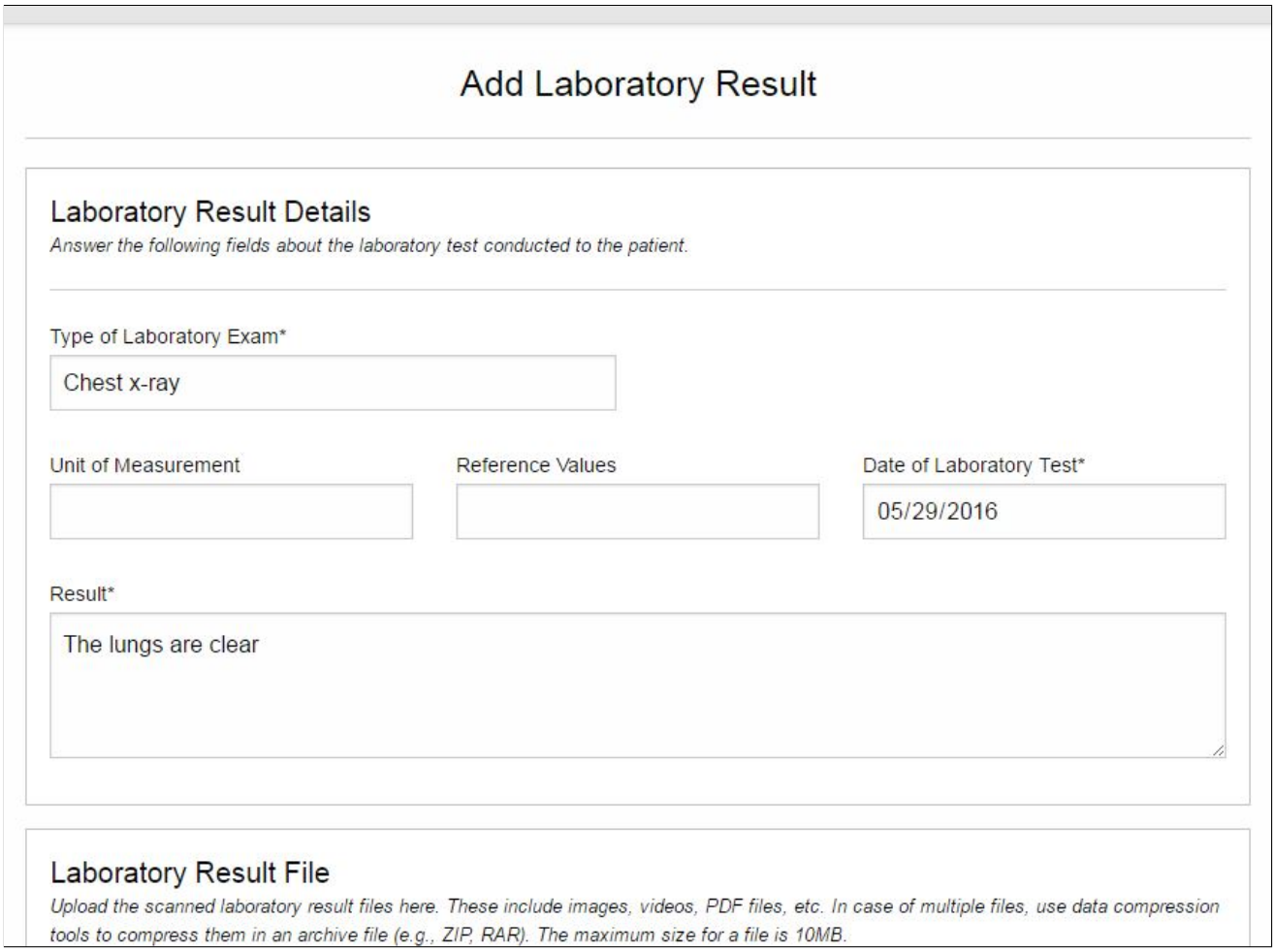

Figure 28: Input laboratory result page of PSP.

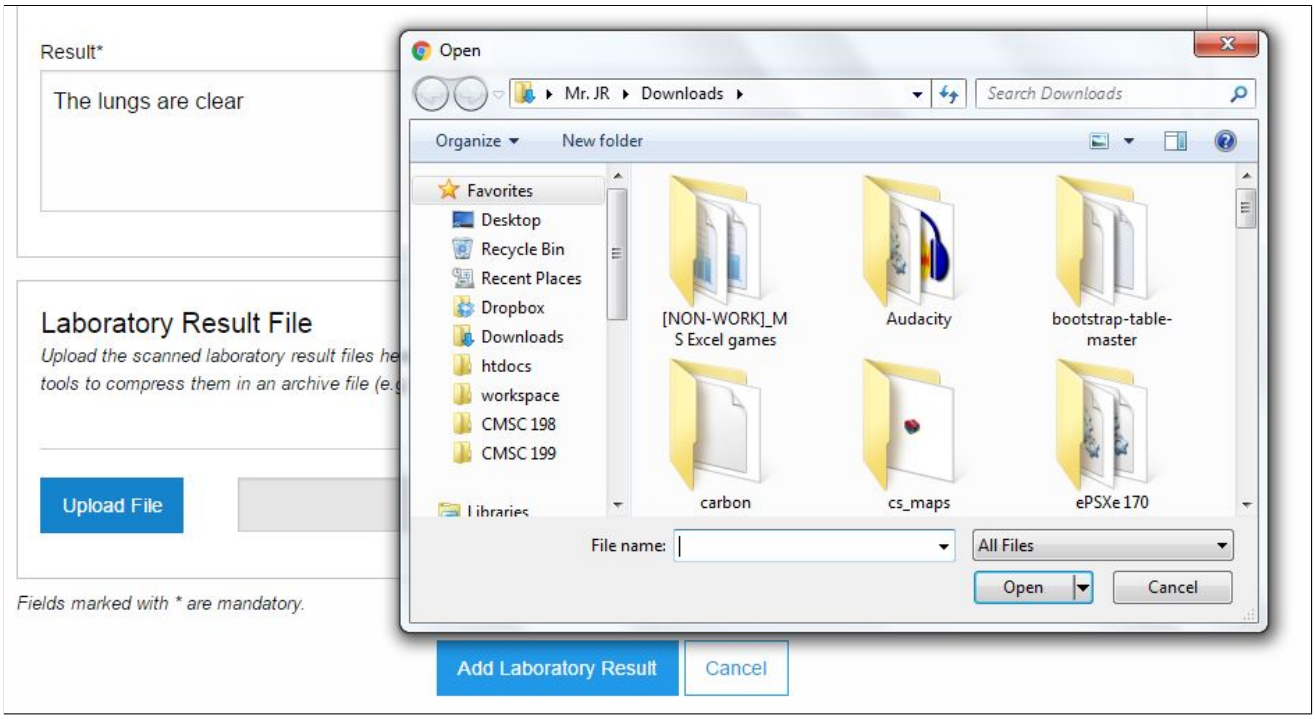

Figure 29: Uploading a scan file of laboratory result of PSP.

After the input of daily medical condition, then patient can set for a follow-up visit with his/her doctor. This is shown in Figure 30. On one hand, if the patient wants to set a follow-up visit directly without adding a medical condition, the patient may access the Schedule for Visits page through the left menu, which is shown in Figure 31. The patient will receive notification through e-mail when a schedule is approved by the health professional. An e-mail containing the approved schedule is displayed in Figure 32.

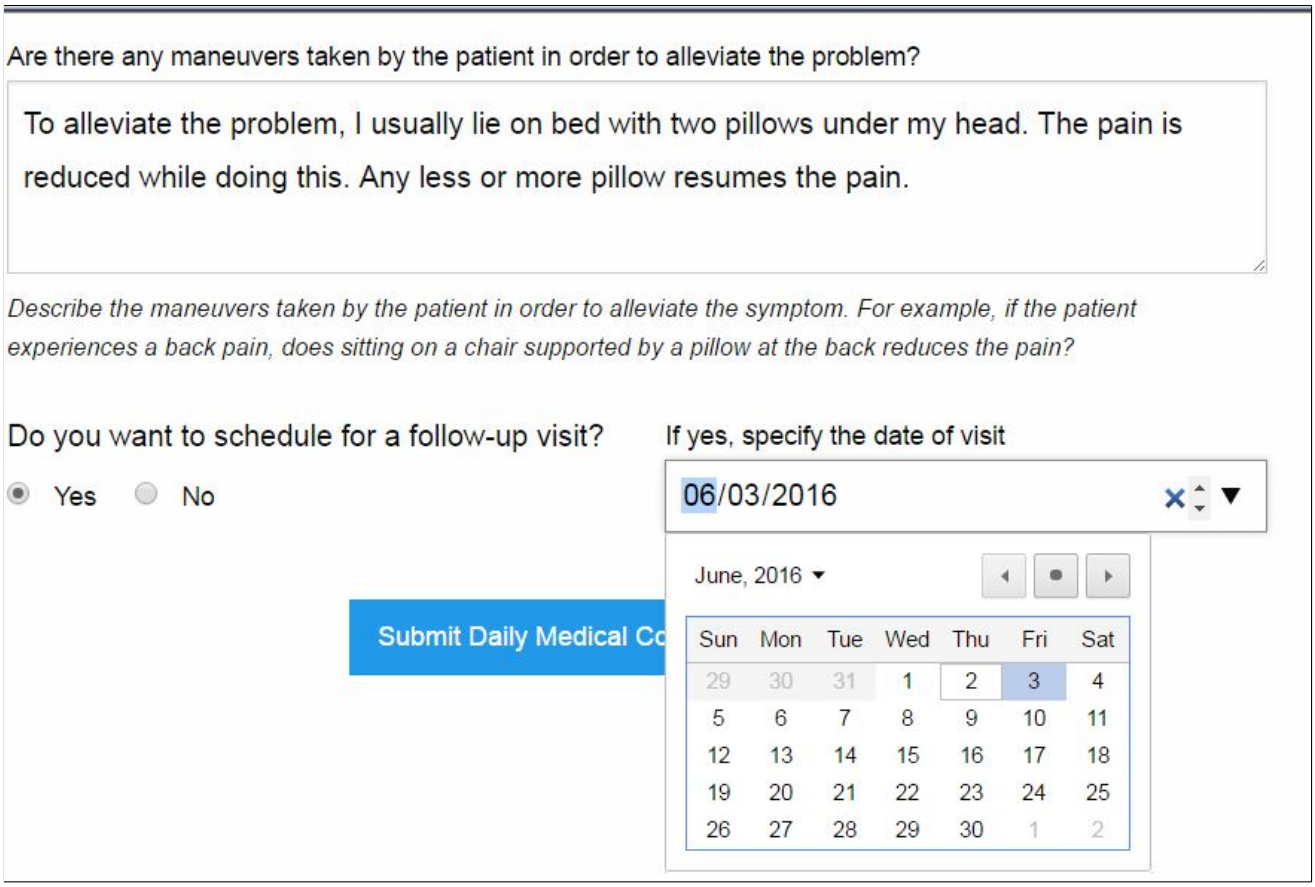

Figure 30: Setting a follow-up visit after inputting a medical condition of PSP.

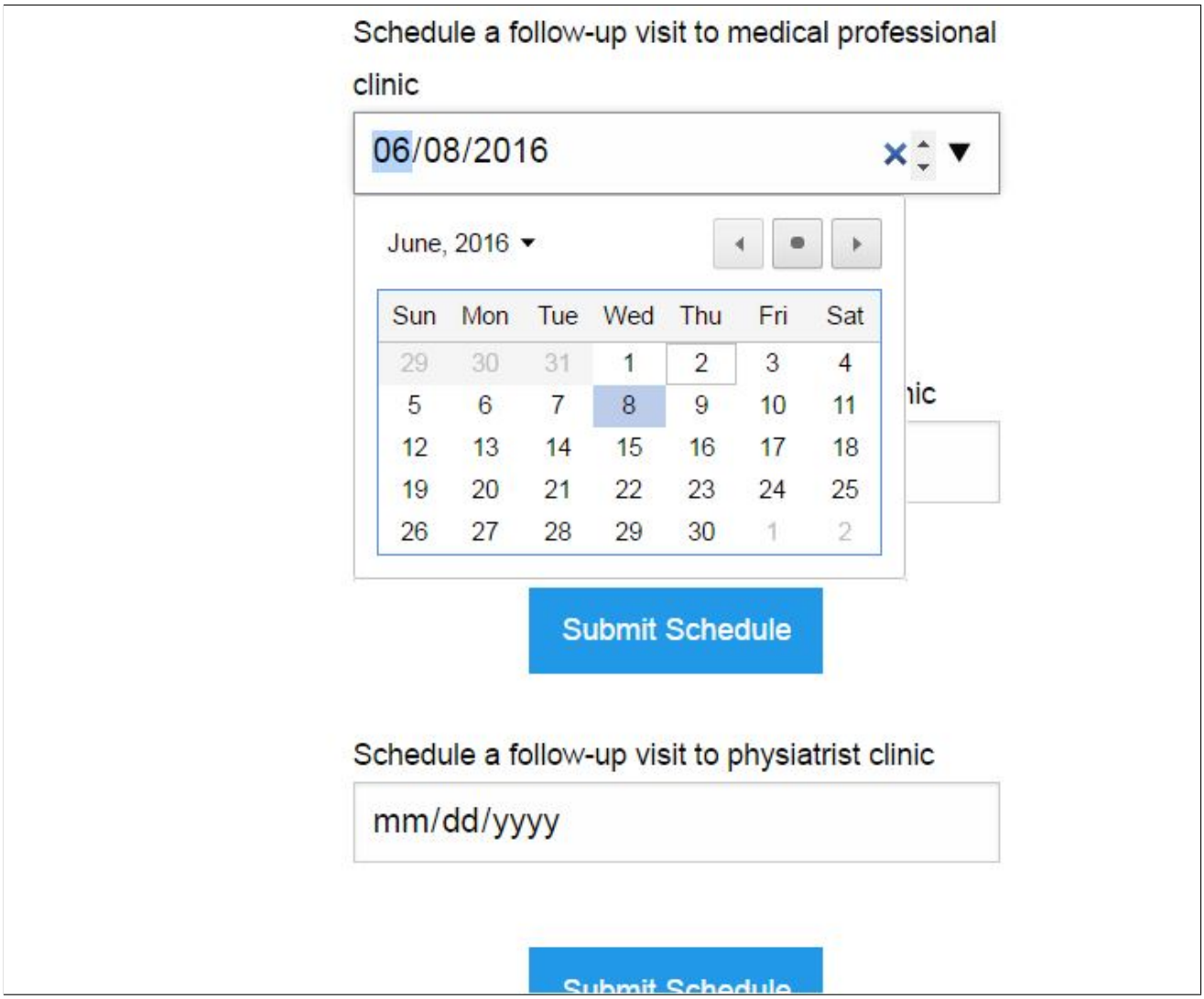

Figure 31: Setting a follow-up visit directly to the doctor of PSP.

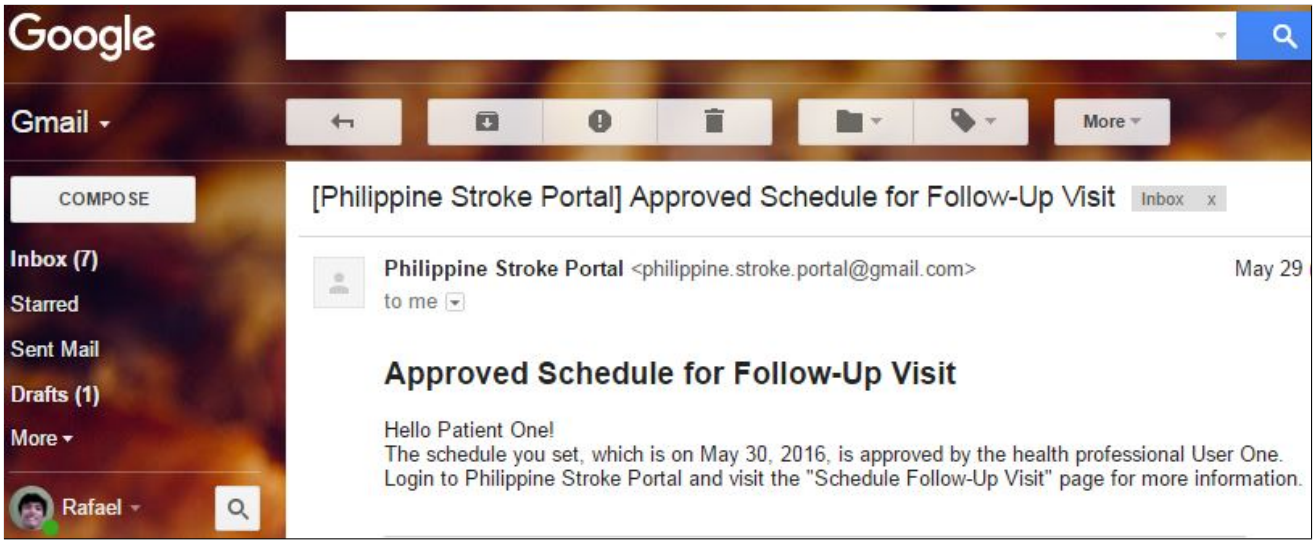

Figure 32: E-mail notification on the approved visit schedule of PSP.

The patient can also participate in forum threads on various topics on stroke and daily coping mechanism to it. Figure 33 shows a user viewing a forum thread and replying to it. Other types of registered users can also perform this functionality.

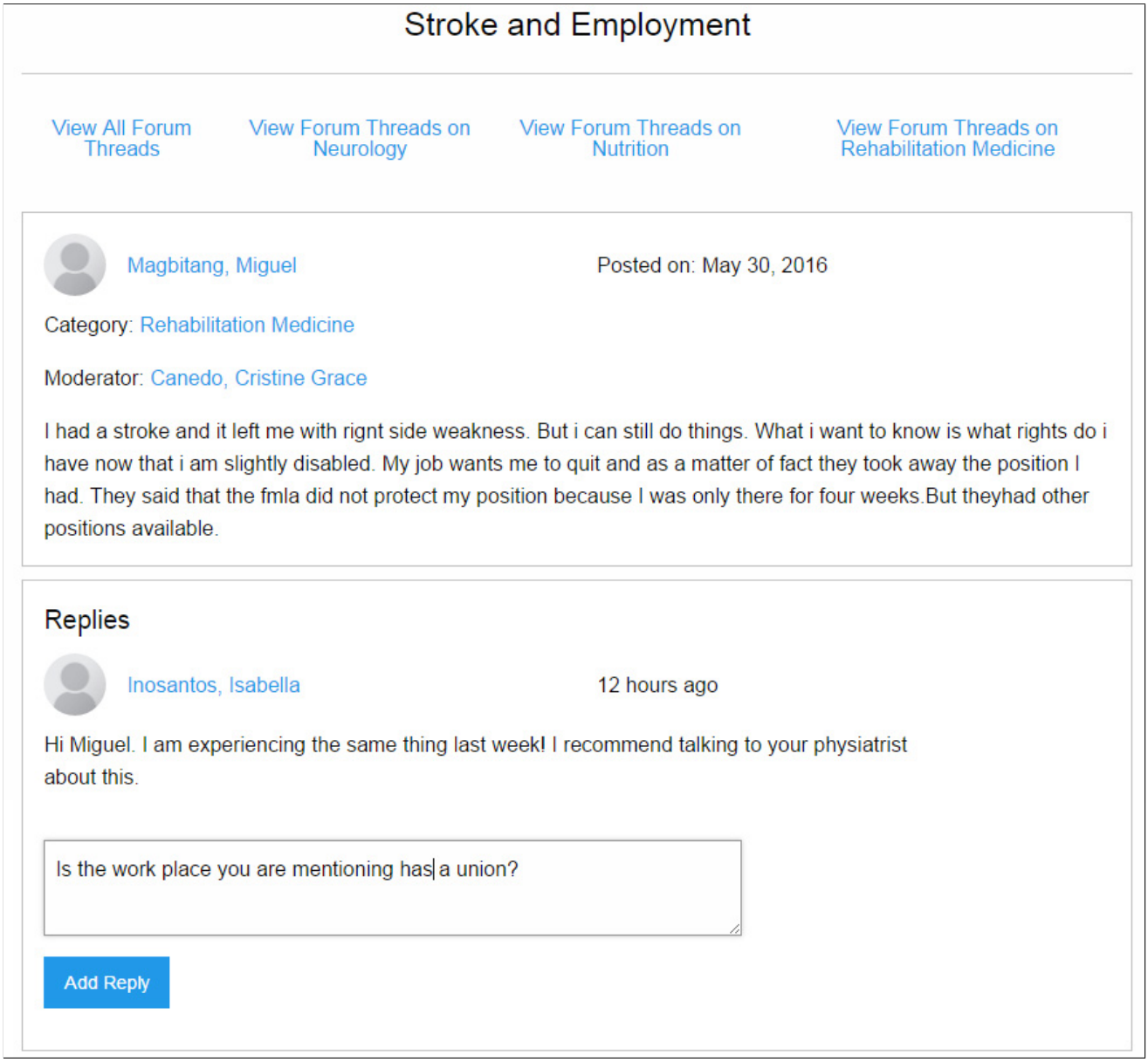

Figure 33: Participate in a forum thread of PSP.

3. **Medical Professional View**

The medical professional can view his/her patients through the Patients page, which is illustrated in Figure 34. Other health professionals, particularly nutritionists and rehabilitation medicine doctors, have the same page for viewing their patients. The medical professional

can view the added daily medical conditions and laboratory results of his/her patients. The view of daily medical condition is showsn in Figure 35, while the view of laboratory results is shown in Figure 36. After seeing them, the doctor can make a decision on the medical condition, which is displayed in Figure 37.

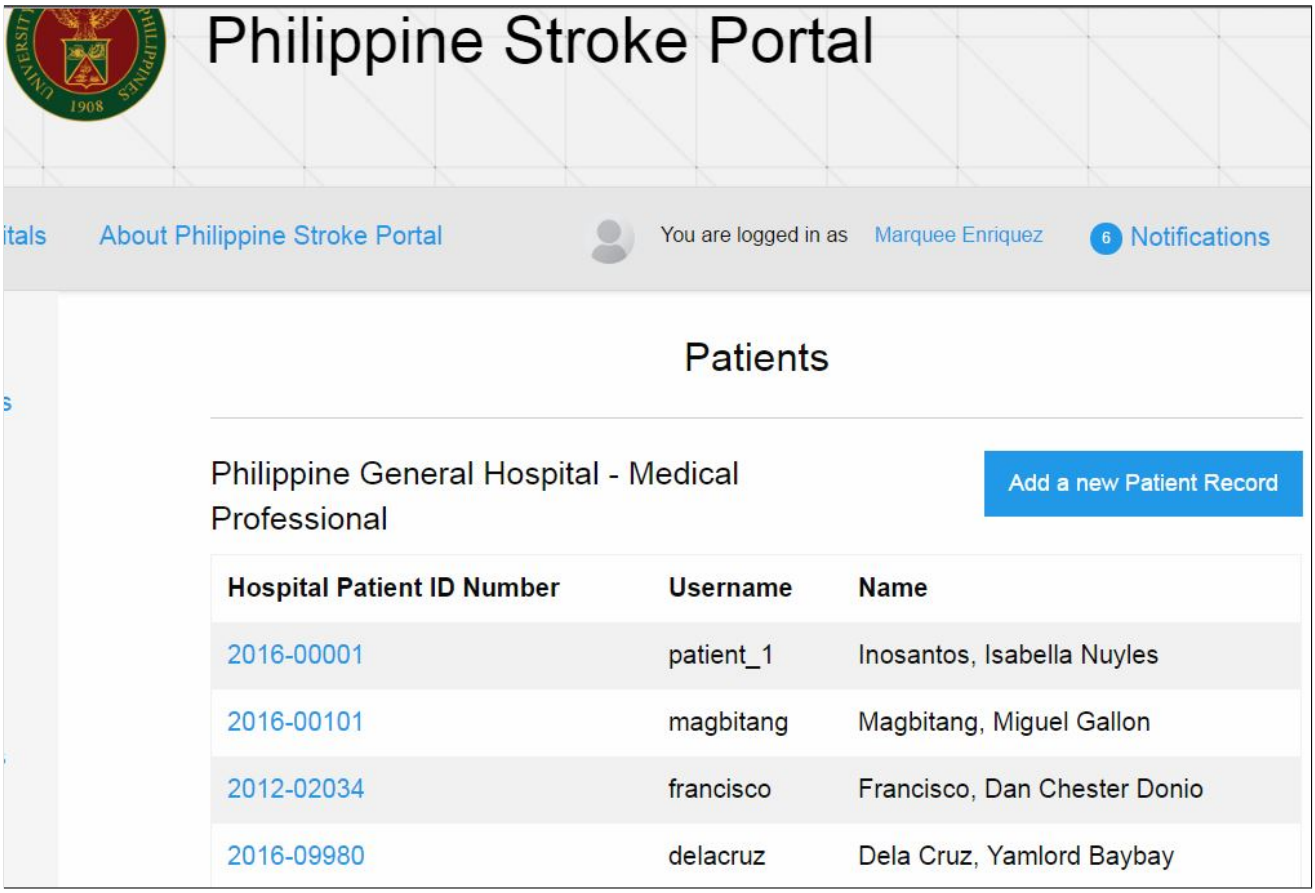

Figure 34: View page of the patients of the health professional of PSP.

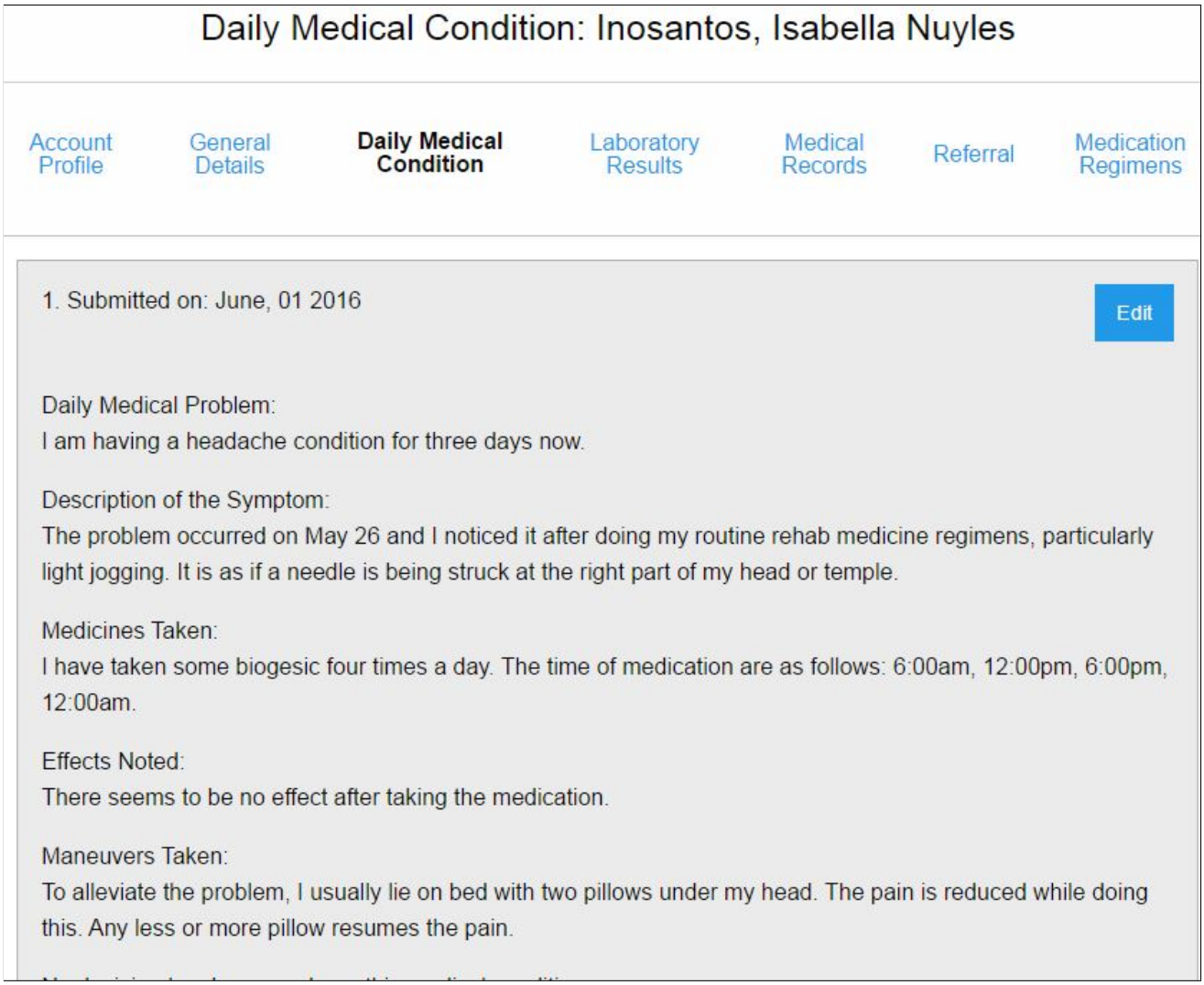

Figure 35: View page of patient's daily medical conditions of PSP.

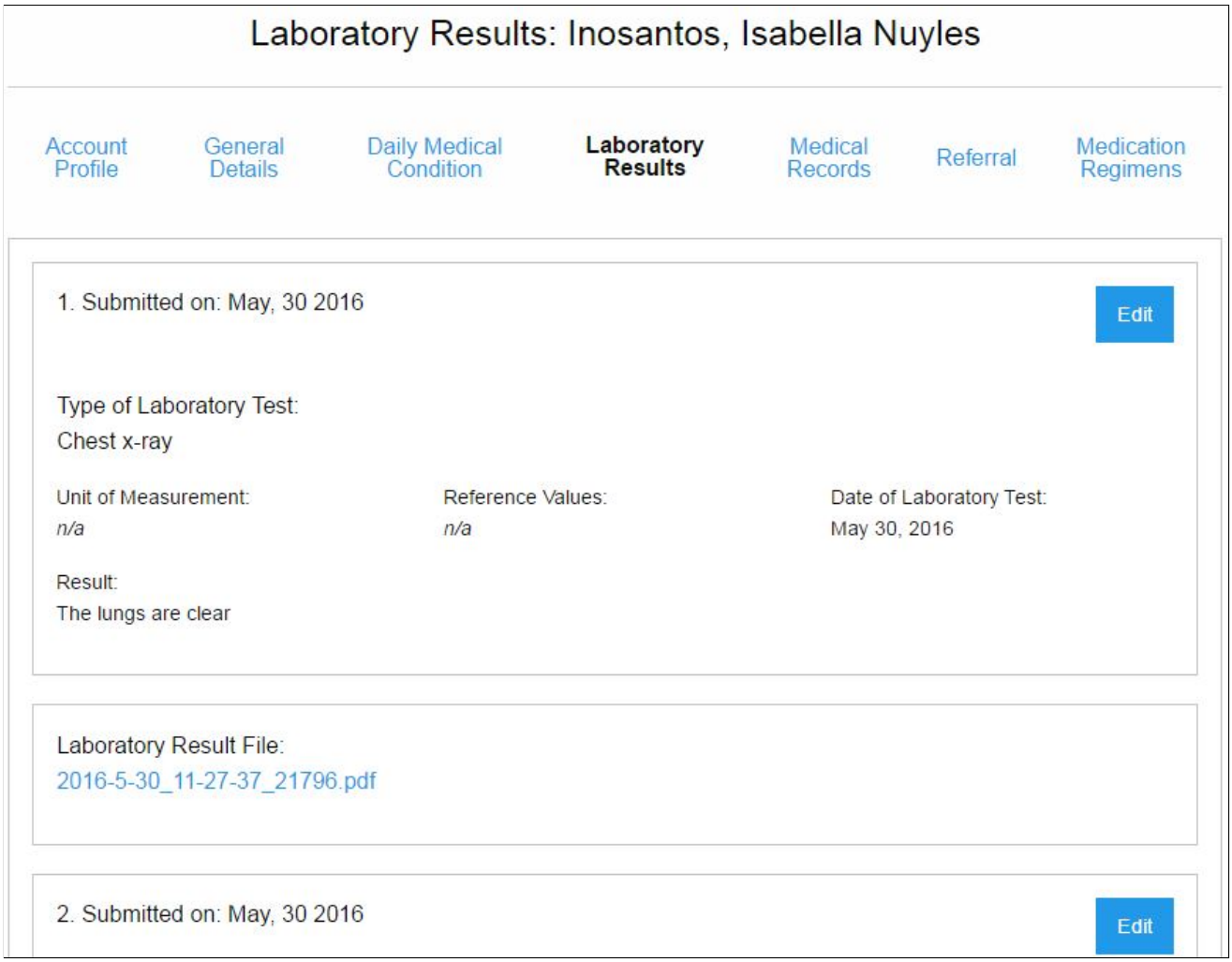

Figure 36: View page of patient's laboratory results of PSP.

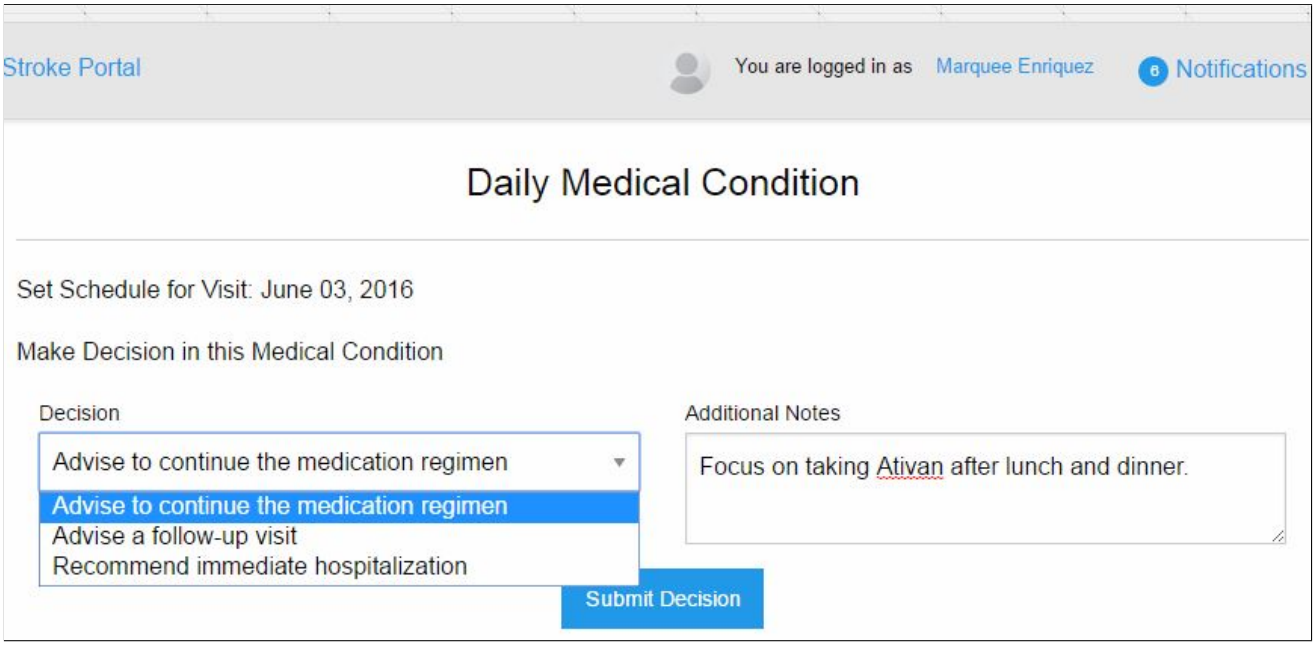

Figure 37: Make a decision to a medical condition of a patient of PSP.

The medical professional can view the listing of patient medical records and update them. The view page of this listing is shown in Figure 38. Further, the updating of these medical records is dsplayed in Figure 39.

Figure 38: View page of medical records of a patient of PSP.

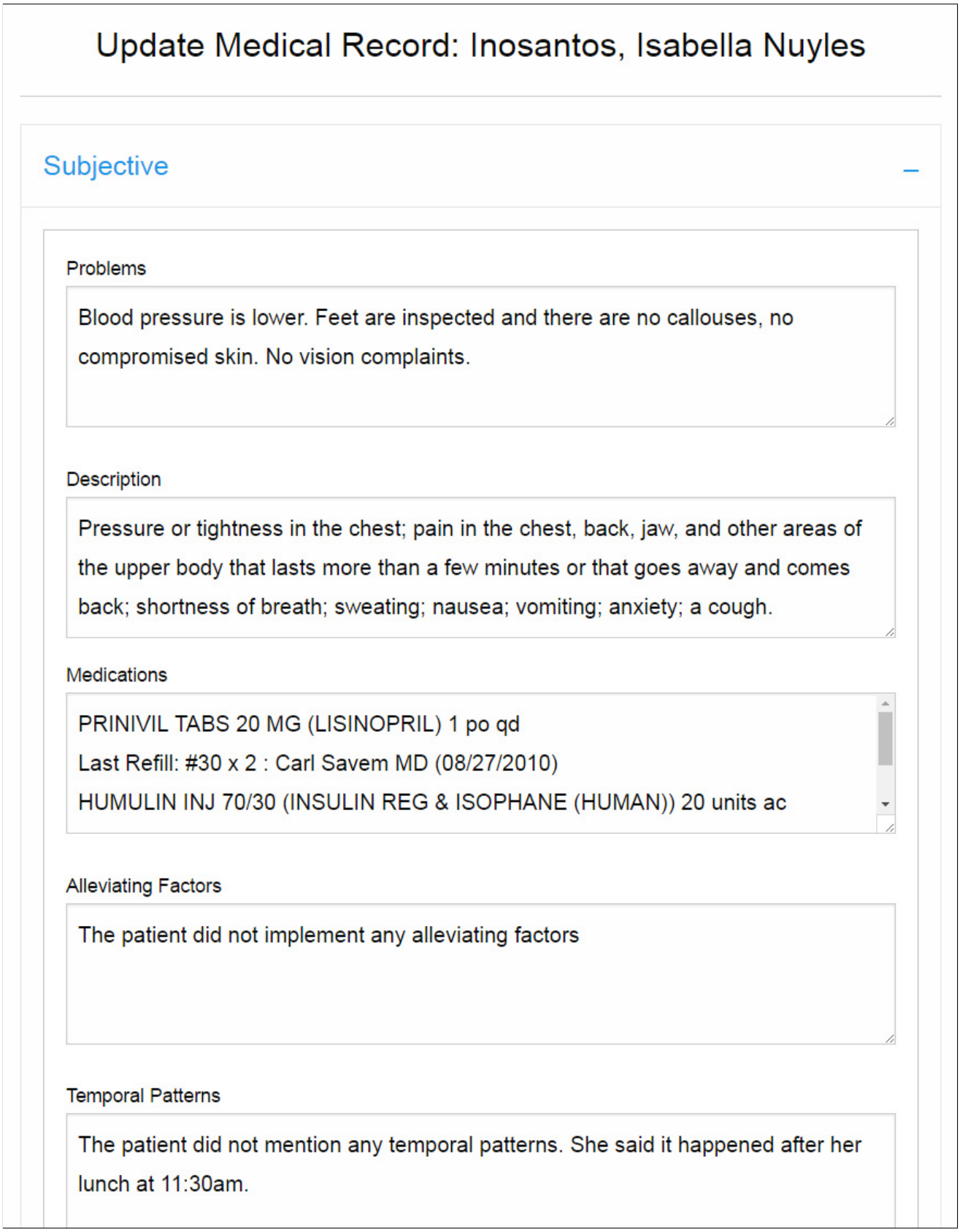

Figure 39: Update page of medical records of a patient of PSP.

The medical professional can also refer the patient to another doctor for other medical

concerns, to a nutritionist, and to a rehabilitation medicine doctors. This is shown in Figure 40.

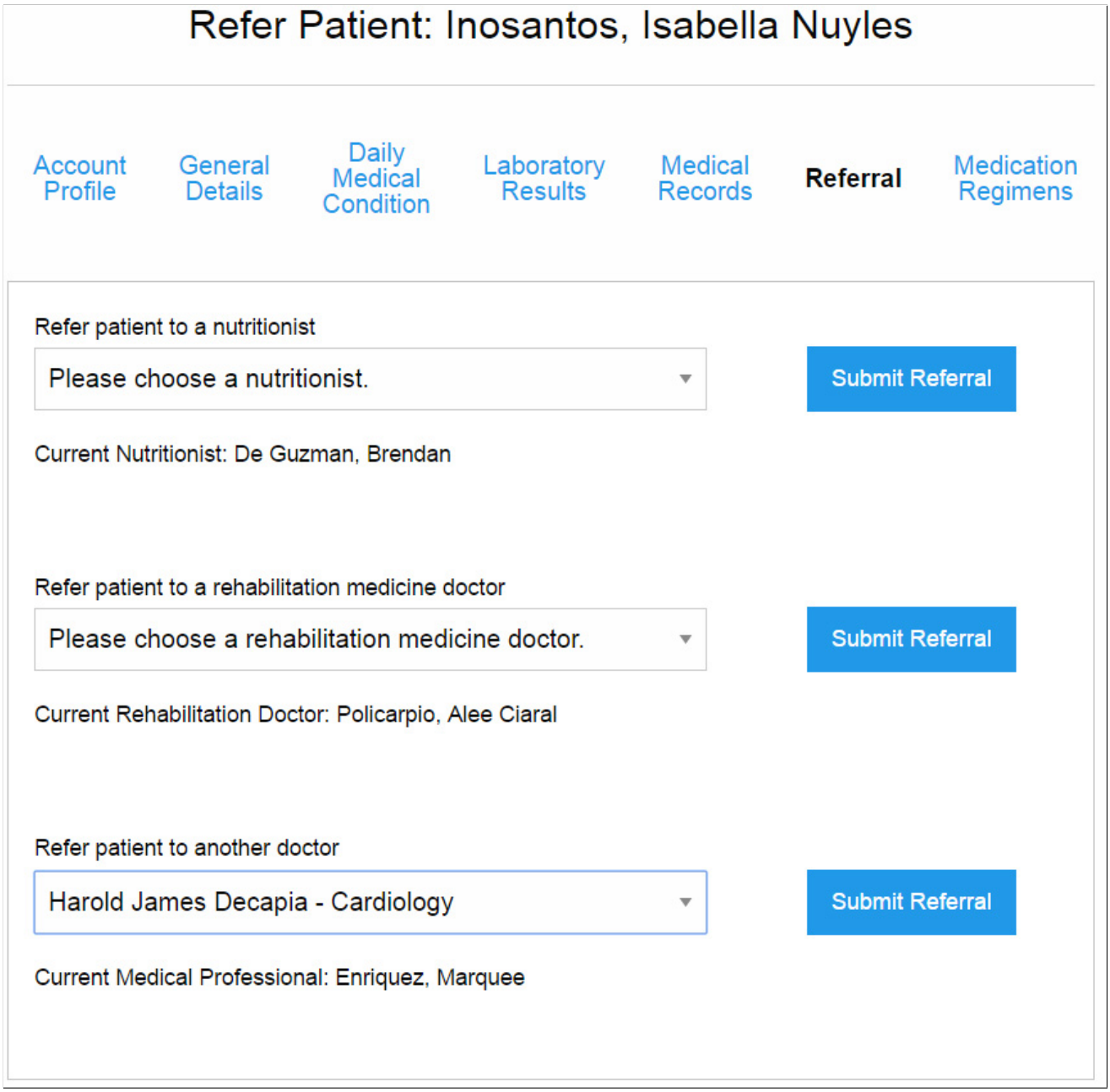

Figure 40: Refer patient to another health professional of PSP.

The medication regimens for the patient can also be viewed by the medical professional. He/she can also update these medication regimens. Other users, particularly nutritionists and rehabilitation medicine doctors, have the same functionalities and interfaces in terms of patient regimens. Figure 41 shows the view page of the medication regimens, while Figure 42 shows the edit page of the medication regimens. The doctor can also delete some of the

regimens.

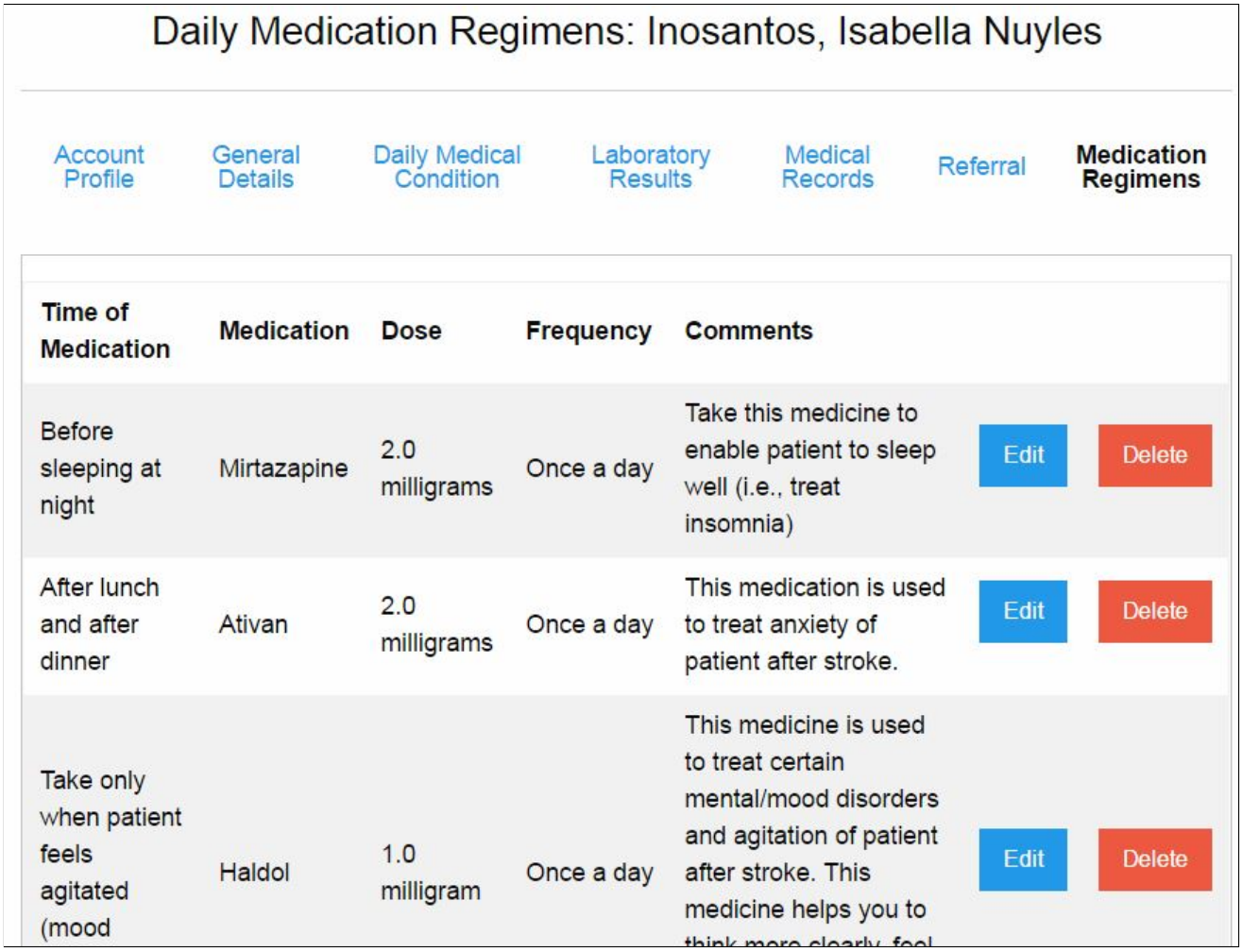

Figure 41: View page of the medication regimens of PSP.

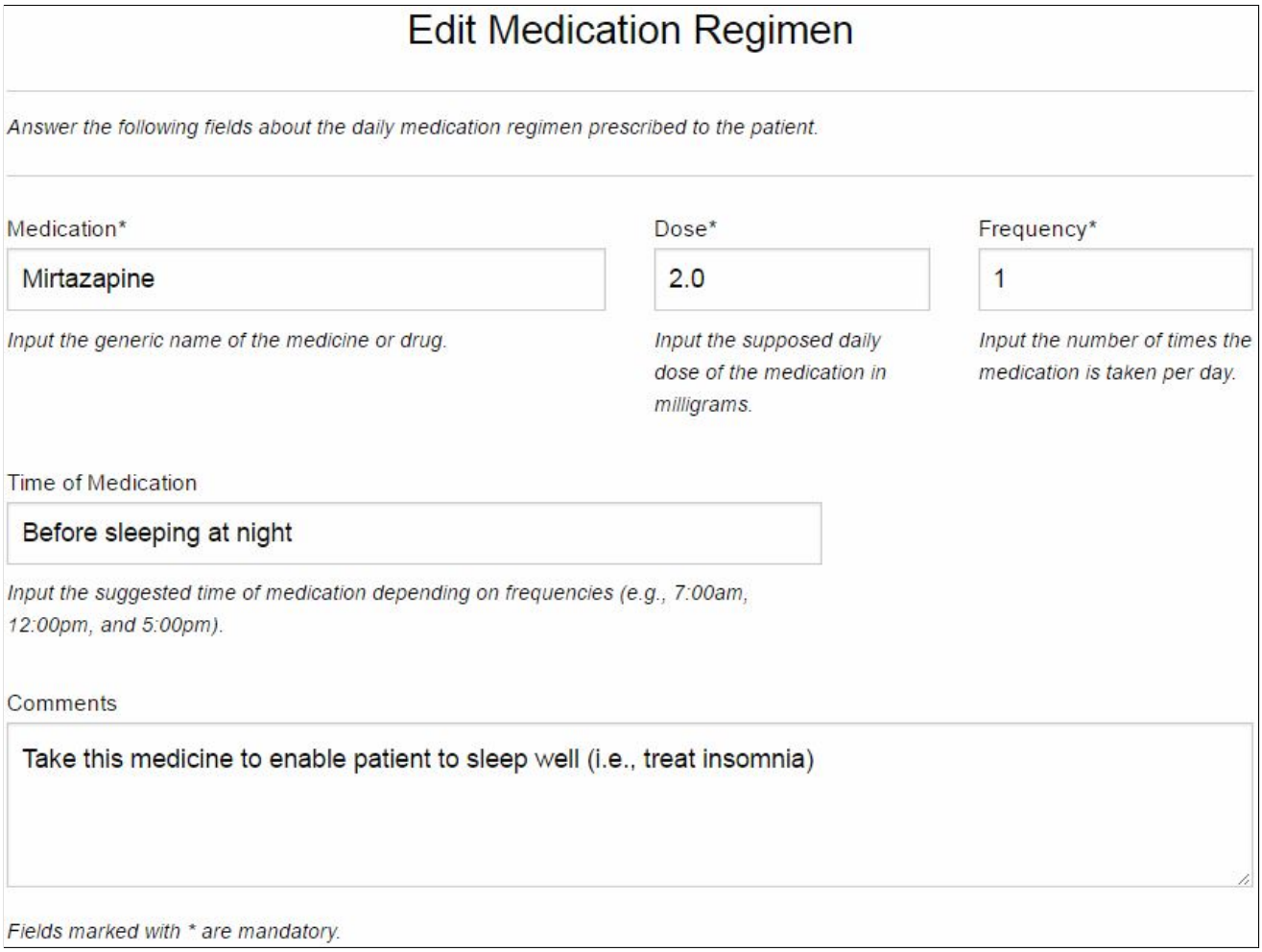

Figure 42: Update page of the medication regimens of PSP.

The medical professional can approve the schedule of follow-up visits from the patients. This is illustrated in Figure 43. In addition, nutritionists and rehabilitation medicine doctors can approve scheduled visits in the same manner as the medical professional.

Figure 43: Approve of follow-up visit schedules page of PSP.

Medical resources can also be uploaded by the medical professional for the viewing of the general public, as shown in Figure 44. However, it will not be posted immediately after uploading it; a supervisor must approve it first. Accordingly, the medical professional will receive a notification in the web portal when the uploaded medical resource is approved by a supervisor. This is displayed in Figure 45. Other health professionals, like nutritionists and rehabilitation medicine doctors, can also upload medical resources similarly.

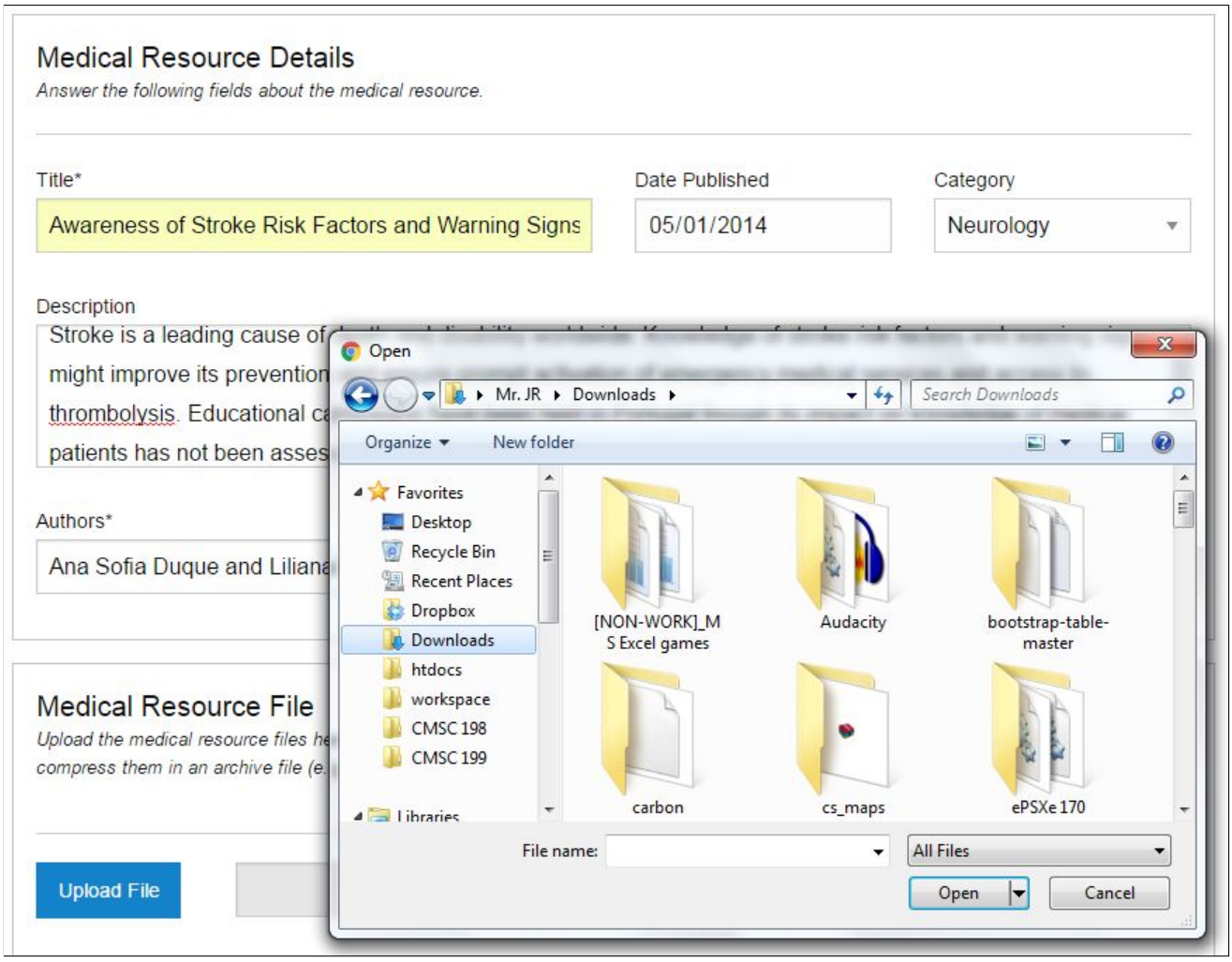

Figure 44: Upload medical resource page of PSP.

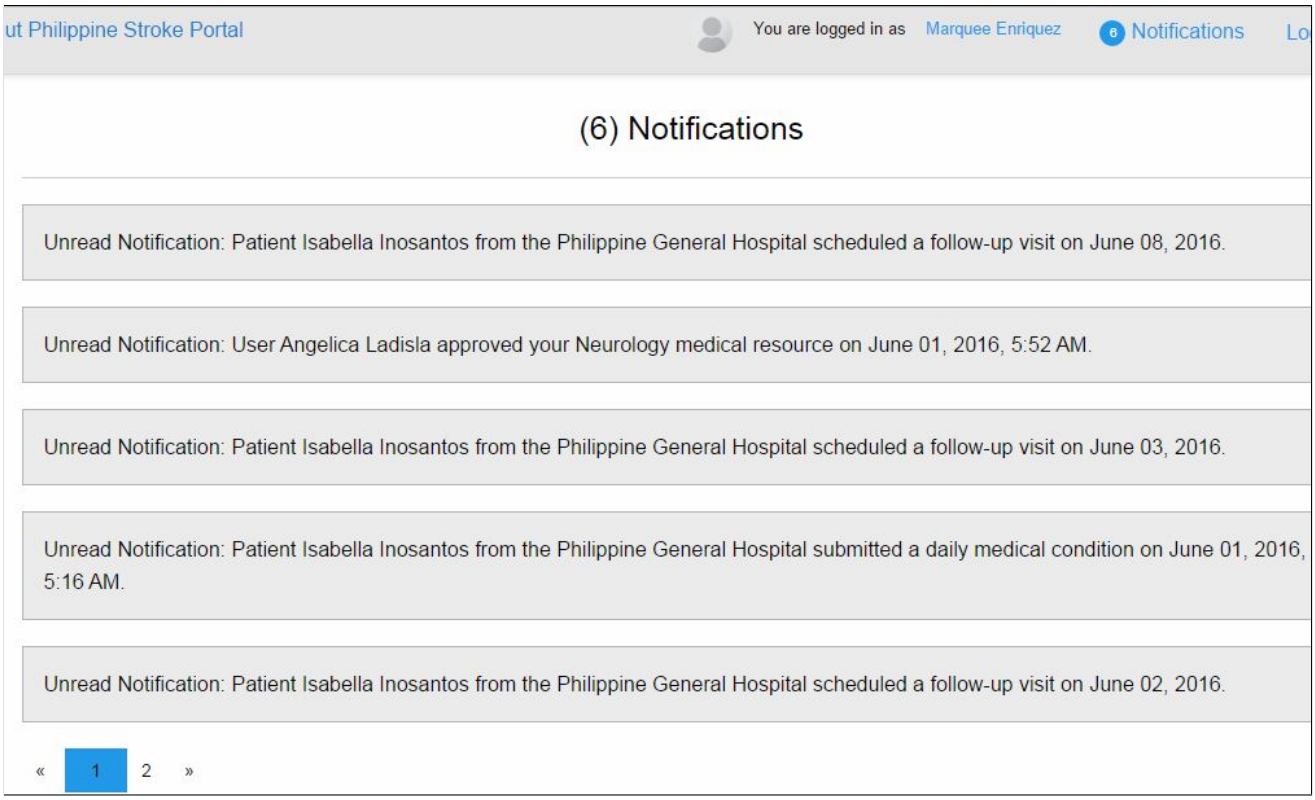

Figure 45: Notification of an approved medical resource page of PSP.

4. **Supervisor View**

   First, the supervisor will receive a notification when a medical resource under his/her category (e.g., neurology) is uploaded. The notification is shown in Figure 46. Likewise, the supervisor can approve these medical resource, as illustrated in Figure 47.

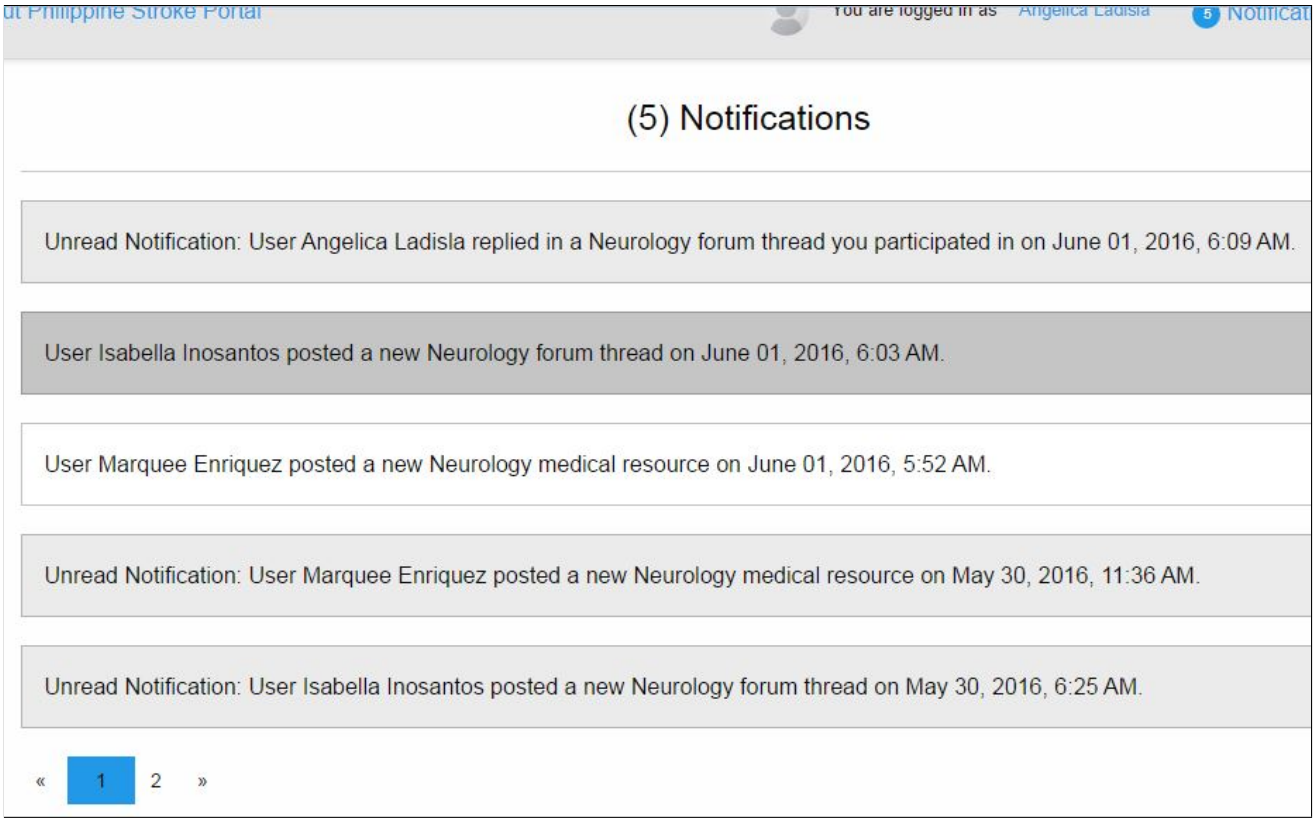

Figure 46: Notification of an uploaded medical resource page of PSP.

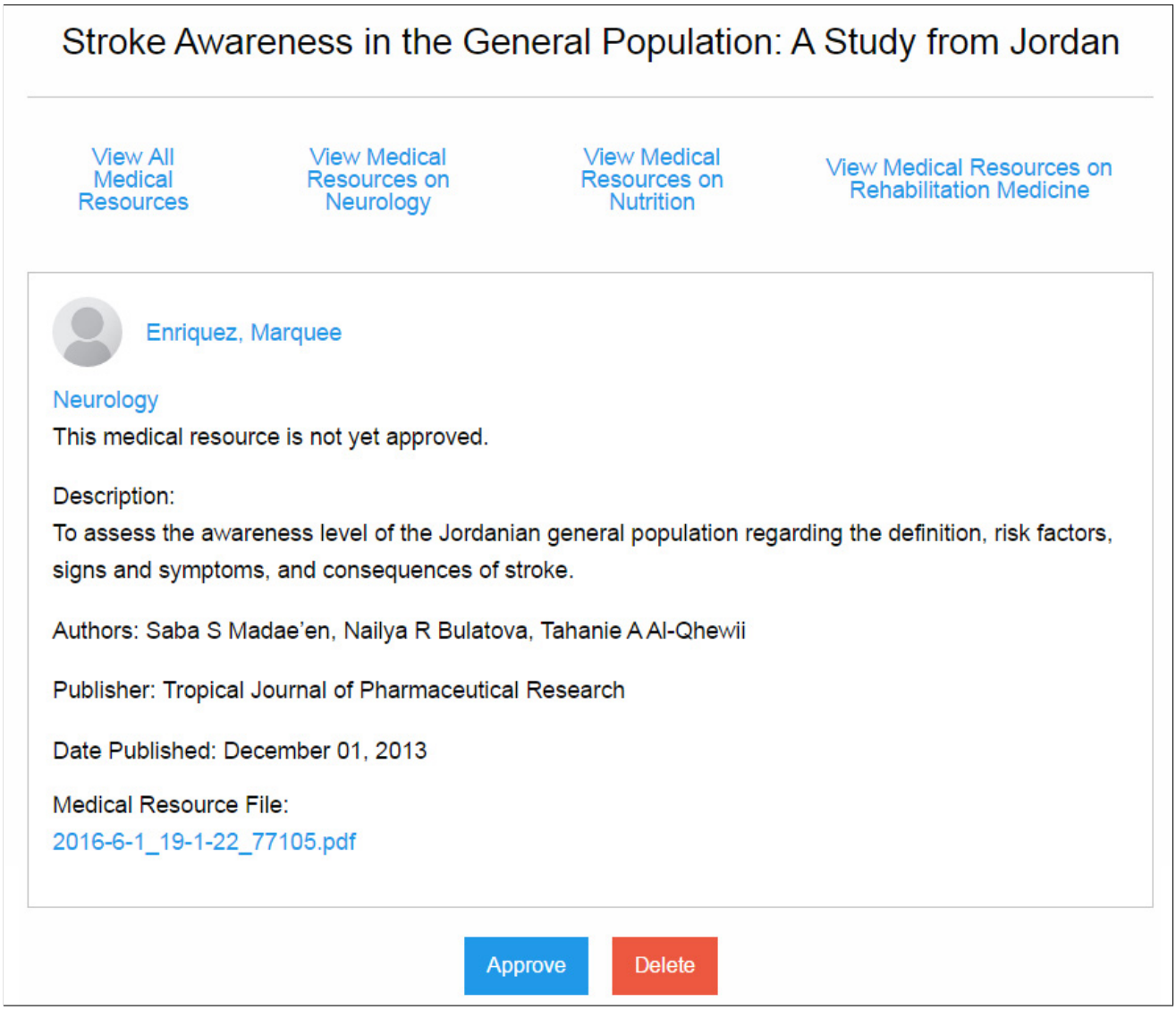

Figure 47: Approve medical resource page of PSP.

The supervisor can also act as a moderator in a forum thread. The moderation includes an authority to delete a forum reply or to delete the whole forum thread. These functionalities are shown in Figure 48. The supervisor can also choose to moderate recently created forum threads under his/her category. This is displayed in Figure 49.

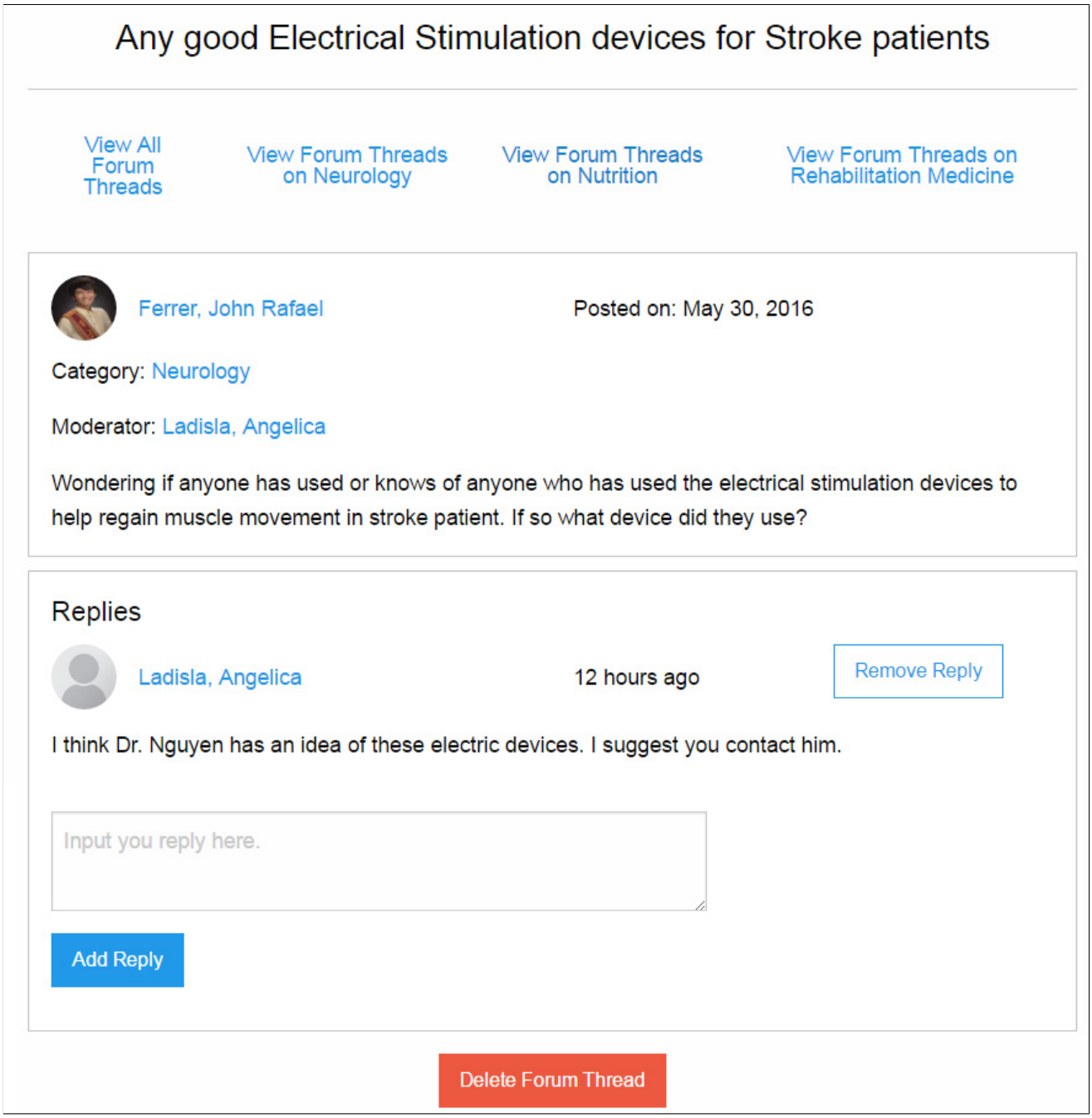

Figure 48: Moderate particular forum thread of PSP.

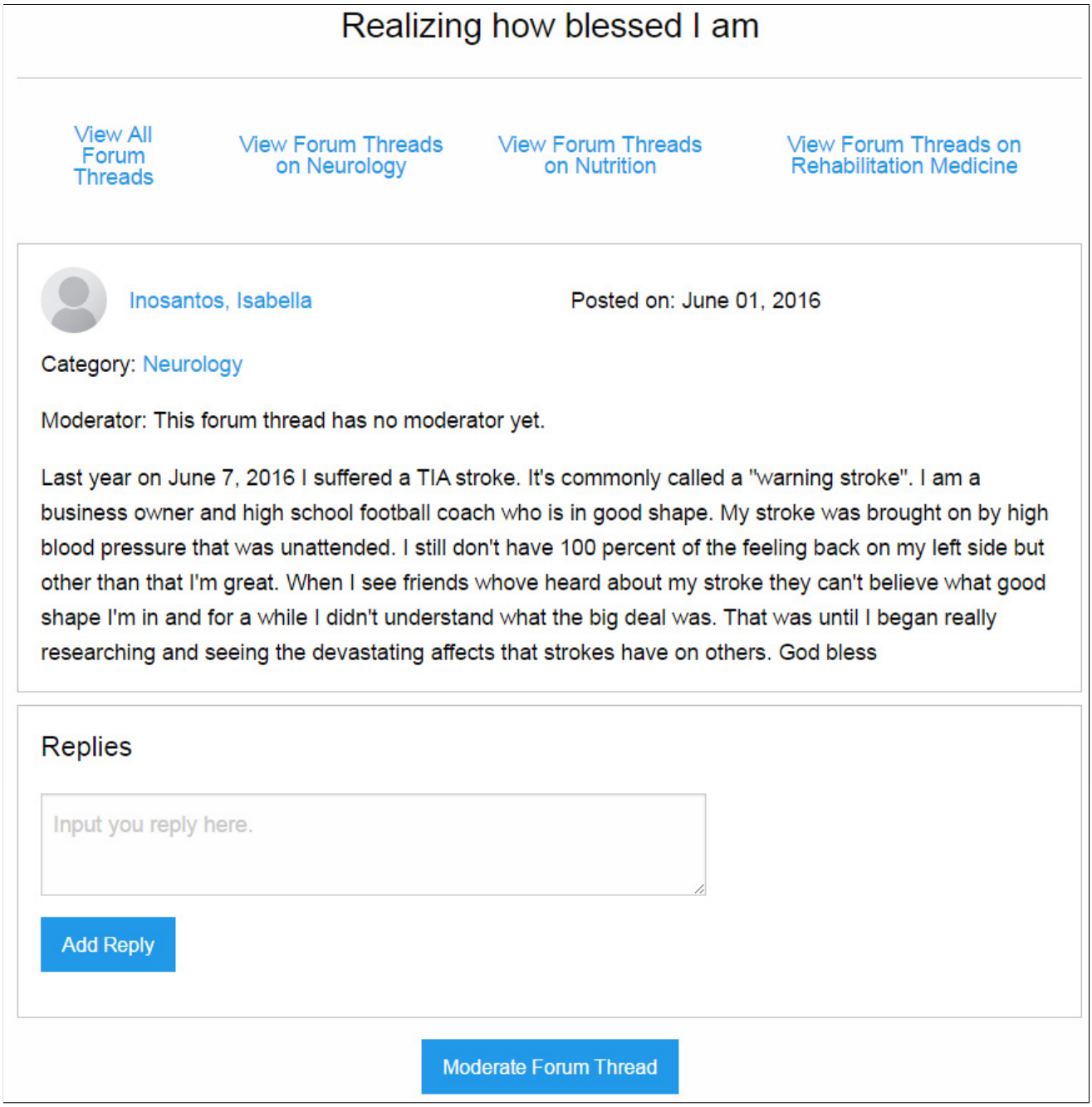

Figure 49: Approve a forum thread of PSP.

5. **Nutritionist View**

The nutritionist can view and update the patient nutrition records, as shown in Figure 50. Other functionalities of the nutritionist are similar to the functionalities of the medical professional.

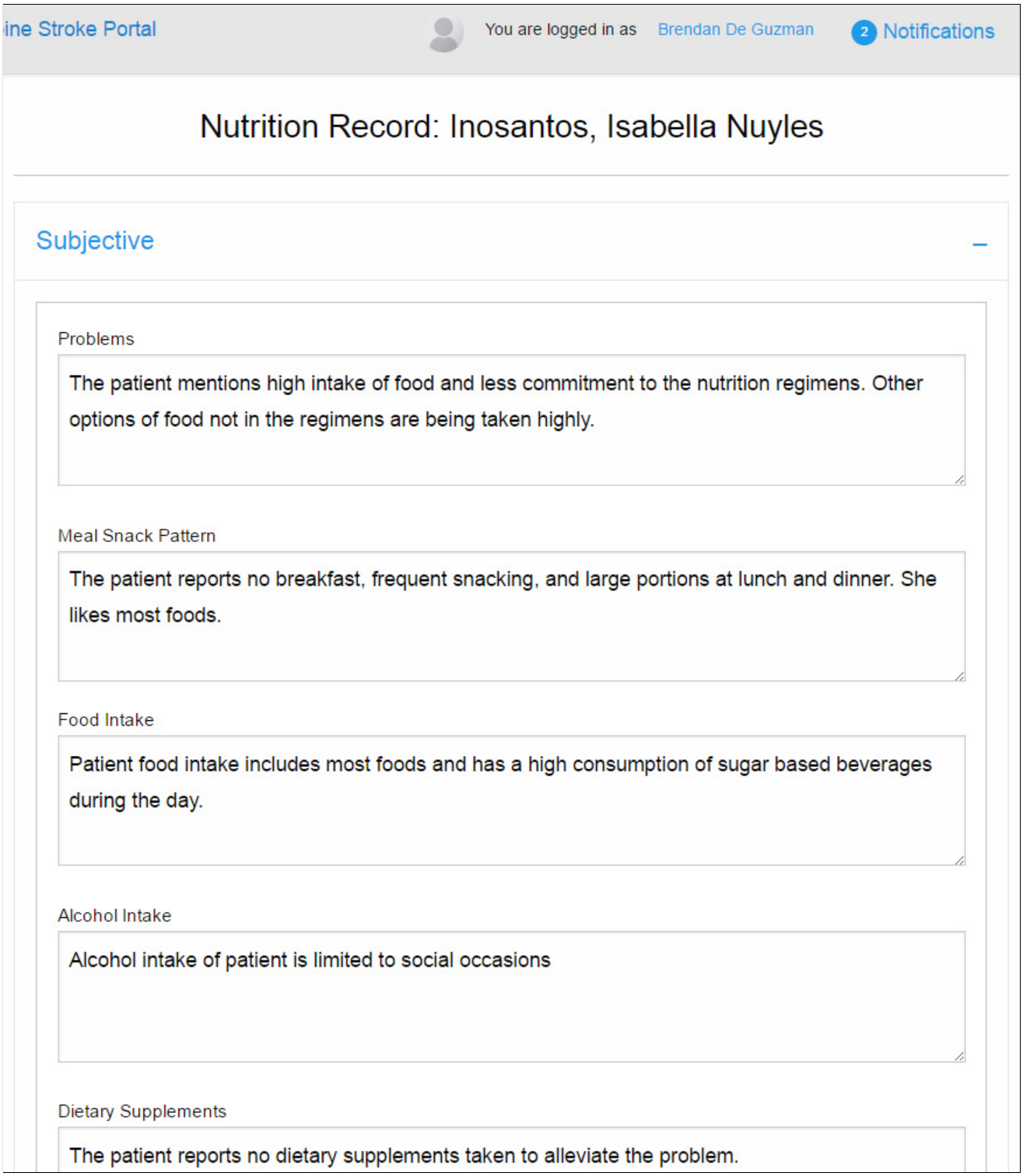

Figure 50: Update page of patient nutrition records of PSP.

6. **Rehabilitation Medicine Doctor View**

The rehabilitation medicine doctor can view and update the patient rehabilitation medicine records, as shown in Figure 51. Other functionalities of the rehabilitation medicine doctor

are similar to the functionalities of the medical professional.

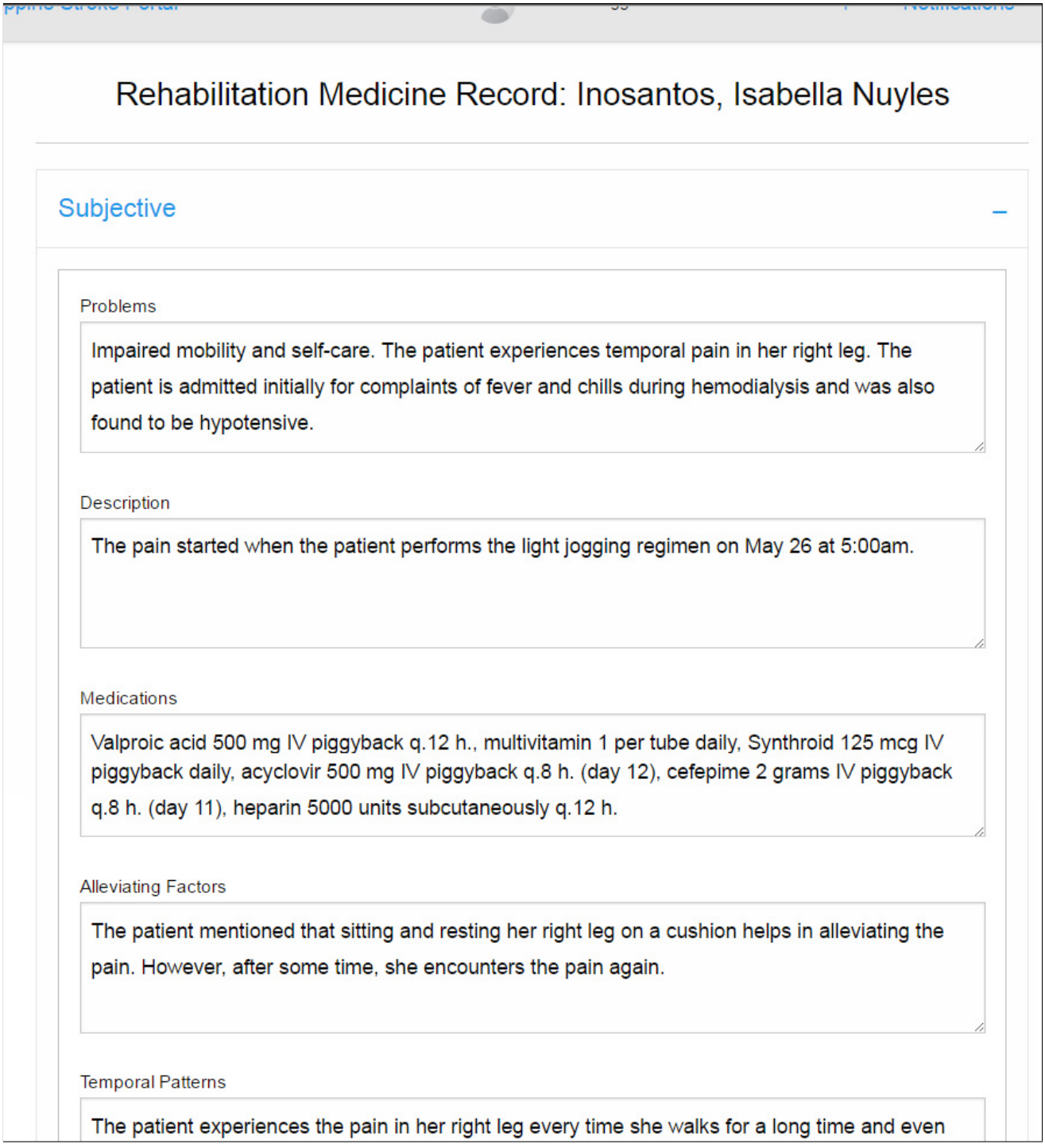

Figure 51: Update page of patient rehabilitation medicine records of PSP.

7. Local Hospital or Clinic Administrator View

The local hospital or clinic administrator can manage the accounts of the staff of the hospital. This means that he/she has an authority to activate or deactivate the accounts of the staff

members or professionals. This is shown in Figure 52. The local administrator can also post announcements and events from the hospital, which is illustrated in Figure 53.

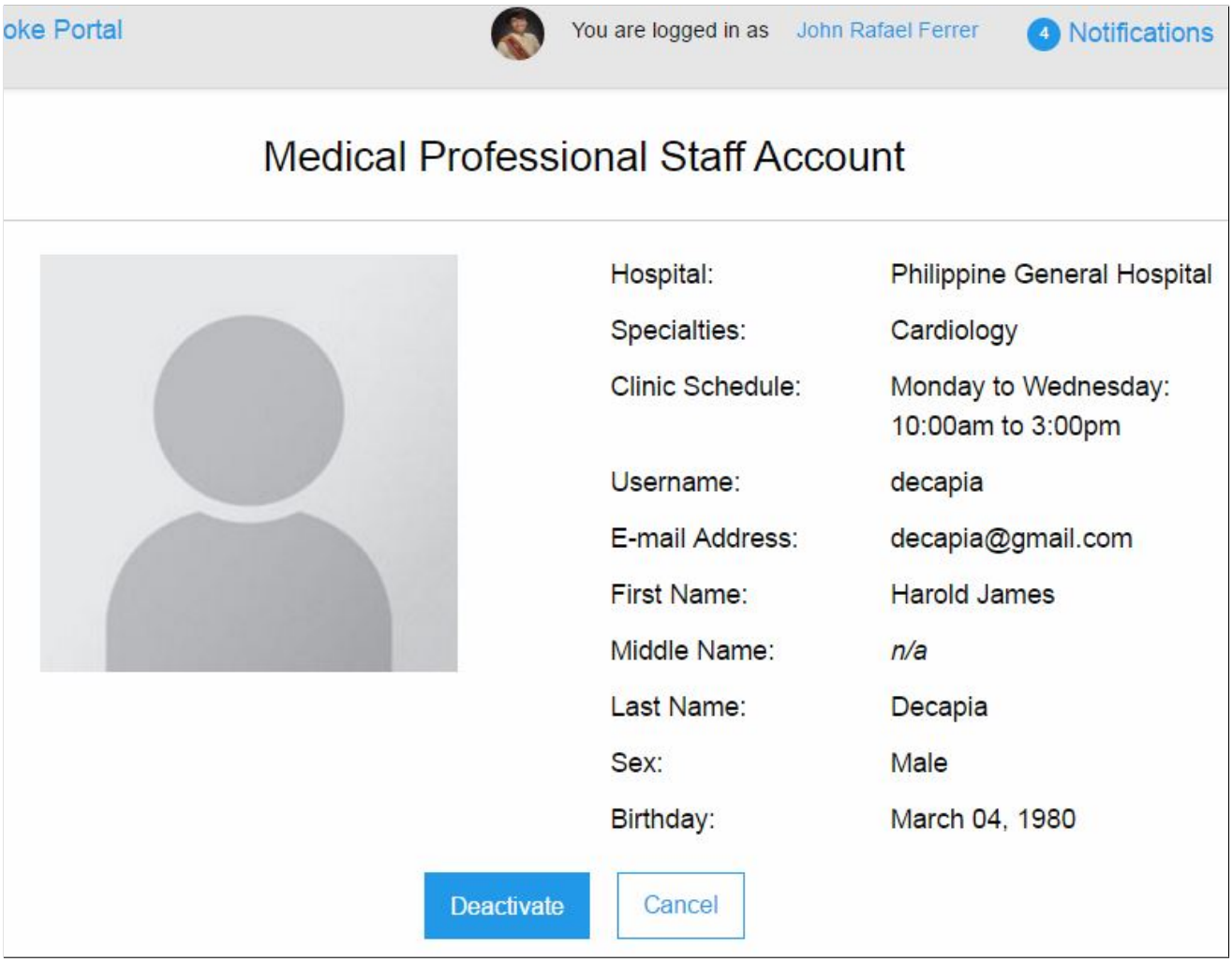

Figure 52: Activate/deactivate staff accounts of PSP.

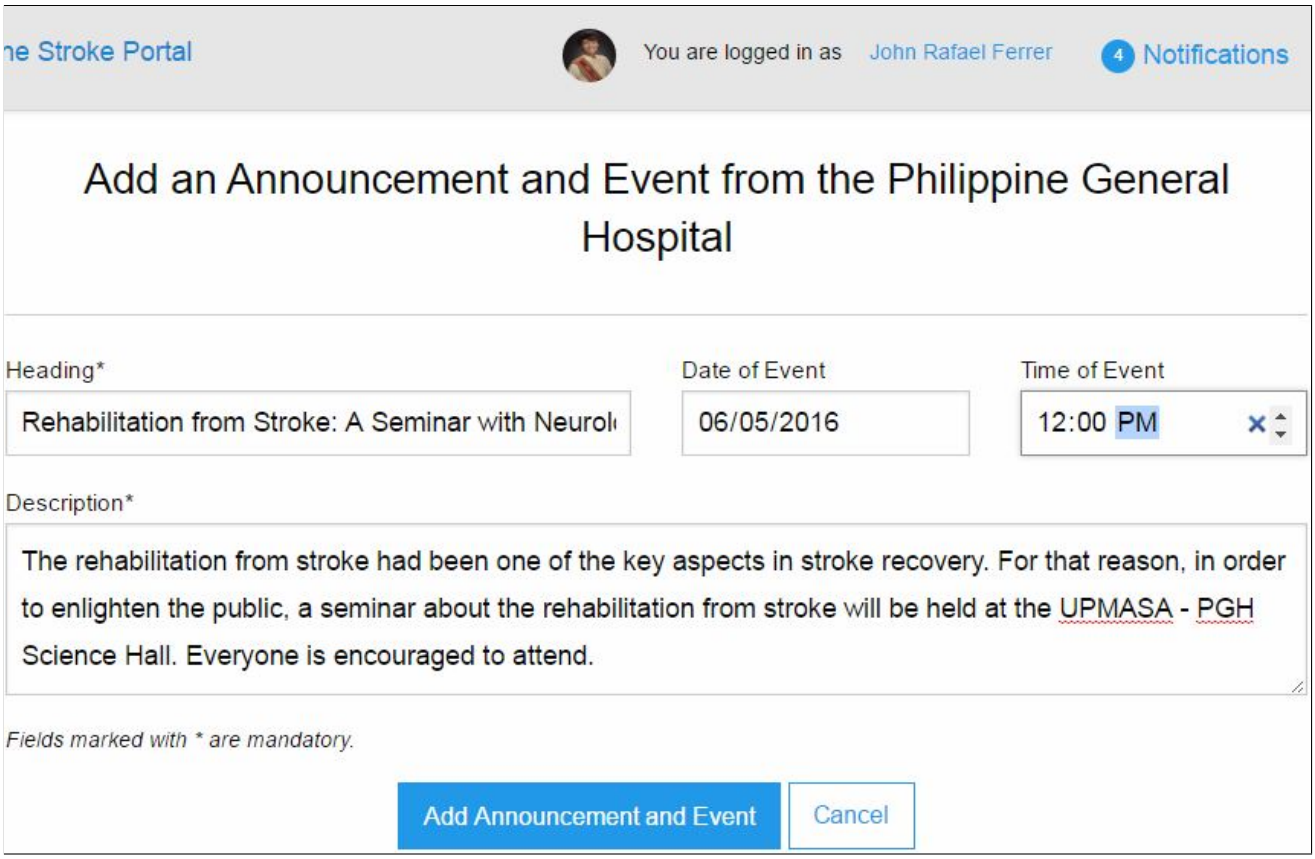

Figure 53: Posting announcements and events page of PSP.

8. **System Administrator View**

The system administrator can manage the accounts of supervisors and local hospital or clinic administrators. Figure 54 shows the management of accounts of supervisors, while Figure 55 shows the management of accounts of local hospital or clinic administrators.

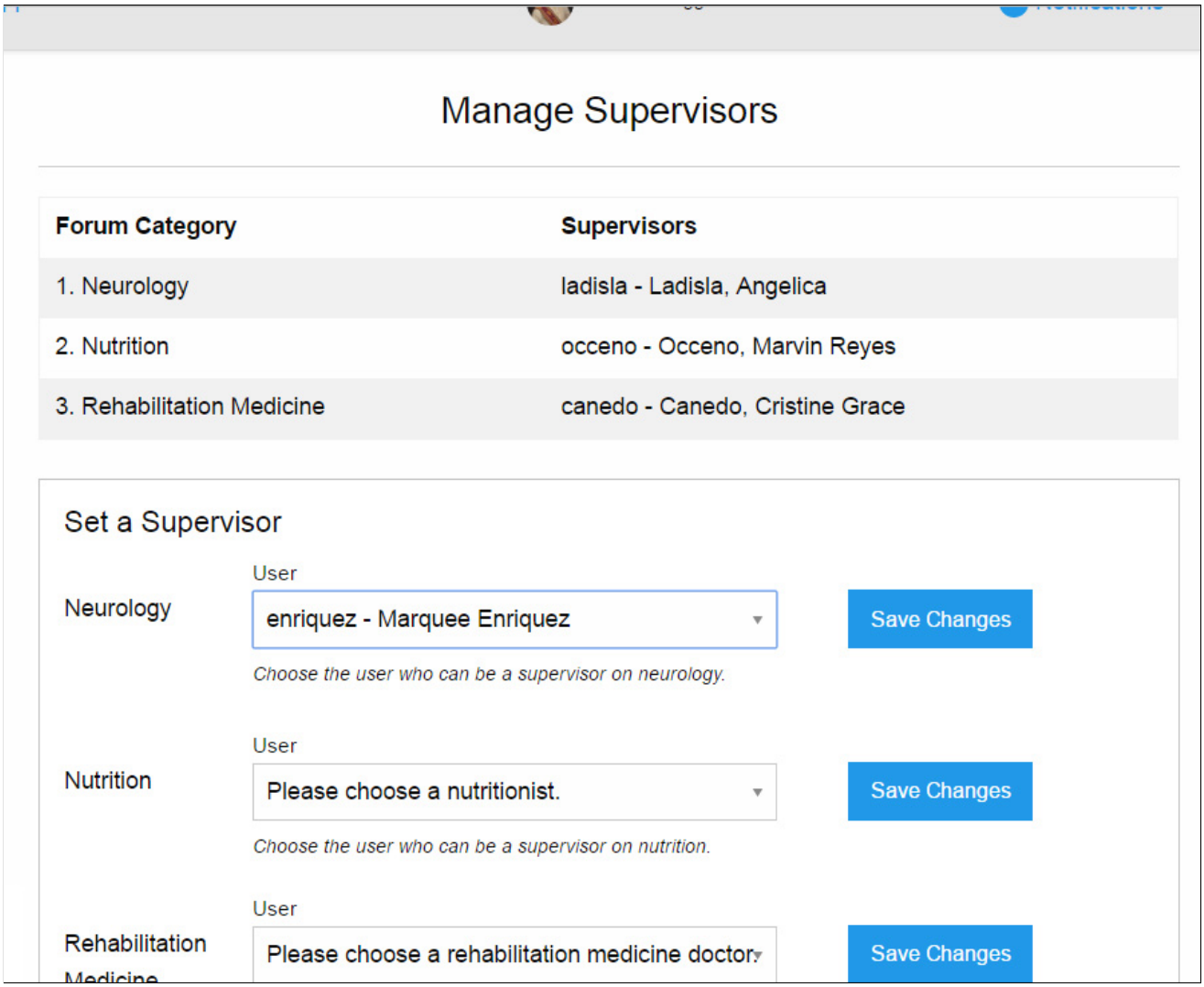

Figure 54: Manage supervisor accounts page of PSP.

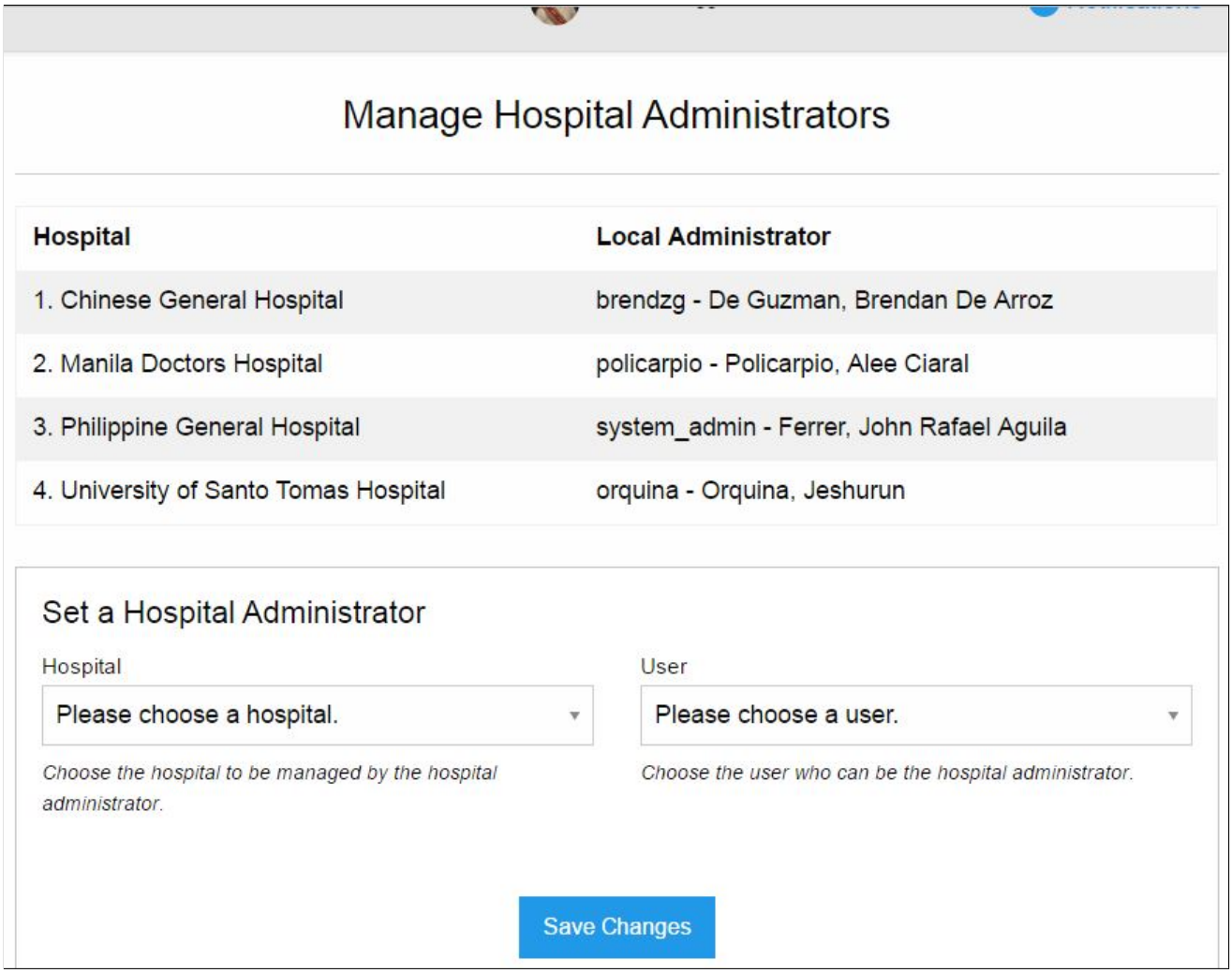

Figure 55: Manage local hospital or clinic administrator page of PSP.

VI. Discussions

The Philippine Stroke Portal (PSP) is a web portal that aggregates various medical resources about stroke for stroke patients and caregivers. It also allows them to carry out the submission or input of minor daily conditions to be analyzed by the corresponding medical professional. The medical professional then makes a decision on the medical condition of the patient: either the doctor advises to continue the medication regimen, advises to set a follow-up visit to his/her clinic, or recommends immediate hospitalization. Consequently, the web portal allows an e-mail notification of the patient about this decision, particularly when a schedule for a follow-up visit is involved.

This web portal on stroke also allows medical professionals, nutritionists, and rehabilitation medicine doctors to view and update their patient records and suggested regimens. They can also post medical resources, which will be approved by a corresponding supervisor, for the viewing of the general public. The web portal also implements the use of forum threads to highlight non-medical remedies on stroke and its other aspects, like lifestyle, rehabilitation, and diet.

One of the advantages of PSP is that it aids in performing day-to-day online submission and viewing of patient medical conditions. There will be less cost on the part of stroke patients and caregivers, especially when they do face-to-face visits in neurological clinics and hospitals in particular scenarios for minor daily medical conditions.

In addition, a second advantage of PSP is that patients of the portal have an alternative way of setting follow-up visit schedules with their doctors. Aside from the usual setting of schedule for visit after inputting a daily medical condition, patients can also directly schedule a follow-up visit to the neurology, nutrition, or rehabilitation medicine clinic through the appointments page. With this in mind, the process of scheduling appointments with doctors is hastened for the stroke patients. The portal allows the stroke patients to set an appointment with their doctors immediately or to submit a medical condition to their doctor, together

with the suggested schedule for follow-up visit.

Another advantage is that supervisors, who are verified by professional associations, such as the Stroke Society of the Philippines, will ensure that the posted resources on stroke, nutrition, and physical medicine are medically validated and up to date. This presents a high level of credibility in bringing these resources to the patients and caregivers.

Furthermore, another advantage is that when patients want to emphasize on non-medical remedies on stroke, the use of forum threads is a remarkable way of highlighting them. In addition, these validity of the information from the forum threads will be high since these are moderated by supervisors from professional associations, unlike other websites with forum threads that are not moderated.

Lastly, the communication among stroke patients, caregivers, neurologists, nutritionists, and rehabilitation medicine doctors through the forum threads is an advantage of the web portal. Participants in forum threads can highlight any aspect of stroke rehabilitation, such as daily living activities, coping mechanisms, non-medical techniques in stroke recovery, and inspirational stroke survivor stories. These aspects cannot be captured completely by medical resources on stroke. These are usually more effectively conveyed through the communication with other patients who currently have stroke or who experienced stroke in the past.

However, the use of web portal on stroke has disadvantages. Even though the web portal elaborates on several aspects of stroke, such as symptoms, initial treatments, and regimens, it is still difficult for the patients or caregivers and medical professionals to convey their thoughts to one another. The submission of medical condition online might not encapsulate the complete description of the condition and there is no way yet of telling the doctor about this except through implementing private messaging in the portal or carrying out the usual face-to-face visits.

Another disadvantage of the system is that the referral to another health professional with a different specialty (e.g., cardiology) for other medical concerns is

not really a part of the system. The web portal is limited only to professional accounts for neurologists, nutritionists, and rehabilitation medicine doctors; hence, patient medical records are categorized only to these three specialties. Even if the referral is an important part of providing healthcare to the stroke patients, the system does not include other health professionals in its set of users.

Also, the few number of centers or hospitals that are registered in the portal is a disadvantage. Only few medical resources, announcements and events on stroke, and topics in forum threads are available to the stroke patients. In light to this, it is encouraged that other centers or hospitals should register to the portal to attain wider audience and more information dissemination, thus providing wider access to the appropriate healthcare for the stroke patients.

Additionally, the file types of the medical resources and laboratory results in the portal are limited only to images, videos, and Portable Document Format files. It is possible for medical resources on stroke and laboratory result files to be classified in other file types, especially files with Microsoft Office file extensions, like Word and PowerPoint.

The notification implemented in the system is also limited. A user receives a notification only when he/she is online and logged in to the portal. The system does not employ other ways of sending notifications, such as SMS-based notification and e-mail notification.

The use of a web portal in raising awareness on stroke is effective in reducing the prevalence of stroke and risk of stroke and in improving the quality of life and coping mechanisms of patients. One example of a web system that was able to achieve this is the Power To End Stroke, which is created by the American Heart Association. The Power To End Stroke system educates American citizens about stroke and increases awareness in reducing the incidence of stroke. They have studied that African Americans are more impacted by stroke than any other group within the American population. The African Americans have almost twice the risk of first encounter of stroke compared to other Americans. In addition,

the prevalence of high blood pressure in African Americans in the United States is the highest in the world. [39]

With these in mind, the Power To End Stroke system collaborated with different professional associations to disseminate information on stroke, particularly its warning signs. Their website includes resources that helps in understanding stroke and being able to distinguish its symptoms. The system also presents different stroke survivor stories, information about stroke recovery, and activities for a healthy lifestyle. As a result, they system was able to raise stroke awareness within the African American population. They were able to improve individual health of the population and build healthier community environments by consolidating different resources on stroke and presenting it to the public. [39]

Altogether, taking the Power To End Stroke system into account, it is also possible for the Philippine Stroke Portal to raise stroke awareness, reduce incidence of stroke, and improve the quality of life of stroke patients in the country. This is essential because there is also a high prevalence of stroke in the Philippines.

VII. Conclusions

The Philippine Stroke Portal helps stroke patients and caregivers in assisting them in their coping mechanisms and improving their quality of life by providing medically validated online resources, forum threads that emphasizes non-medical remedies, and hospital announcements and events, especially seminars on stroke and rehabilitation. These are all presented in a single web portal on stroke. It is also managed by supervisors from professional associations and local hospital or clinic administrators.

Furthermore, the system allows the patients to input their daily medical conditions and laboratory results for the viewing of their medical professionals, nutritionists, and rehabilitation medicine doctors. On the other hand, the health professionals is presented with an organized and convenient way of inputting and updating patient medication, nutrition, and physical medicine records. They are also efficiently enabled by the web portal in providing medical regimens to their patients in hastening their stroke recovery and progressing their lifestyle.

VIII. Recommendations

The Philippine Stroke Portal (PSP) is created to support stroke patients, caregivers, and health professionals with functionalities that makes the processes, stages of stroke recovery, and handling of patient records and conditions more efficient. Altogether, the PSP attained all of these objectives. However, the system is limited to being a web application. In order to supplement the awareness on stroke using medical resources and improve the coping mechanisms of patients, the development of PSP for Android and iOS systems is recommended. It is to enable the users to access the portal on a regular basis.

Also, it is recommended to implement the use of private messaging and contacts between users, particularly between patients and medical professionals. With this in mind, it will be easier for the patients and medical professionals to elaborate the medical records, decisions, schedules for visit, regimens, and other aspects of the PSP. Additionally, it will allow them to convey the complete expression of daily medical conditions and replies in certain forum threads.

The addition of professional accounts for other health professionals is also recommended. This will allow the doctors in the portal to refer their patients to a particular health professional for other medical concerns. It will also contribute to the addition of medical resources on other specialties, such as cardiology and nephrology.

Furthermore, other centers and hospitals are encouraged to register to the portal. The reason for this is to provide stroke patients and caregivers with more information dissemination and wider access to appropriate healthcare. It will be easier for announcements and events of hospitals about stroke awareness and rehabilitation to be publicized.

The file types of medical resources and laboratory results in the portal are also limited only to images, videos, and Portable Document Format files. Hence, it is recommended to the web portal to include other file types (e.g., Word files, Excel files, PowerPoint files) in order to gather more online medical resources and add

other laboratory result files.

In addition, the system is also limited to its way of notifying its users. The notifications in the portal are only available when a user is online and logged in to the portal. For that reason, it is recommended that other ways of notifications, such as SMS-based notification and e-mail notification, must be implemented in the portal.

IX. Bibliography

[1] W. H. Organization, "Stroke, cerebrovascular accident". http://www.who.int/topics/cerebrovascular_accident/en/. Accessed: 2015-11-30.

[2] N. S. Association, "What is stroke?." http://www.stroke.org/understand-stroke/what-stroke. Accessed: 2015-11-30.

[3] M. N. Today, "Stroke: symptoms and diagnosis." http://www.medicalnewstoday.com/articles/7624.php?page=2. Accessed: 2015-11-30.

[4] N. I. N. D. Stroke, "Stroke." https://www.nlm.nih.gov/medlineplus/stroke.html. Accessed: 2015-11-30.

[5] W. L. Expectancy, "Philippines: Stroke." http://www.worldlifeexpectancy.com/philippines-stroke. Accessed: 2015-11-30.

[6] J. Navarro, A. Baroque II, J. Lokin, and N. Venketasubramanian, "The real stroke burden in the Philippines," *International Journal of Stroke*, vol. 9, 2014.

[7] P. N. Association, "History." http://www.pna.org.ph/history.html. Accessed: 2015-12-09.

[8] S. S. Philippines, "History." http://www.strokesocietyphil.org/?p=history. Accessed: 2015-12-09.

[9] P. C. H. R. Development, "Empowering people vs. stroke." http://www.pchrd.dost.gov.ph/index.php/news/library-health-news/284-empowering-people-vs-stroke. Accessed: 2015-12-09.

[10] V. Beal, "portal." http://www.webopedia.com/TERM/P/portal.html. Accessed: 2015-11-30.

[11] C. H. P. W. Portal, "Welcome to the cardiovascular health partners web portal!." http://www.heartandstrokepartners.org/. Accessed: 2015-12-09.

[12] I. Katzan, Y. Fan, M. Speck, J. Morton, L. Fromwiller, J. Urchek, K. Uchino, S. Griffith, and M. Modic, "Electronic Stroke CarePath: integrated approach to stroke care," *Circulation: Cardiovascular Quality and Outcomes*, vol. 8, 2015.

[13] J. Anderson, K. Godwin, J. Saleem, S. Russell, J. Robinson, and B. Kimmel, "Accessibility, usability, and usefulness of a Web-based clinical decision support tool to enhance provider-patient communication around Self-management TO Prevent (STOP) Stroke," *Health Informatics Journal*, vol. 20, 2014.

[14] K. Sanders, C. Figiel, J. Kiely, M. Gwynn, and L. Johnston, "Expanding access to intravenous tissue-type plasminogen activator treatment with a practice-based telestroke system," *Journal of Stroke and Cerebrovascular Diseases*, vol. 22, 2013.

[15] K. Sanders, R. Patel, J. Kiely, M. Gwynn, and L. Johnston, "Improving telestroke treatment times in an expanding network of hospitals," *Journal of Stroke and Cerebrovascular Diseases*, vol. 24, 2015.

[16] A. Mort, L. Eadie, L. Regan, A. Macaden, D. Heaney, M. Bouamrane, G. Rushworth, and P. Wilson, "Combining transcranial ultrasound with intelligent communication methods to enhance the remote assessment and management of stroke patients: Framework for a technology demonstrator," *Health Informatics Journal*, 2015.

[17] Apple Inc., "IScore - Ischemic Stroke Predictive Risk Score by USquare Soft Inc. https://itunes.apple.com/us/app/iscore-ischemic-stroke-predictive/id423473762?mt=8. Accessed: 2015-12-03.

[18] T. Park, S. Park, Y. Ko, S. Lee, K. Lee, J. Lee, K. Kang, J. Park, J. Choi, D. Kim, Y. Cho, K. Hong, J. Kim, D. Kim, J. Cha, M. Han, J. Lee, J. Lee, K. Yu, B. Lee, B. Yoon, H. Bae, and G. Saposnik, "The iScore predicts clinical response to tissue plasminogen activator in Korean stroke patients," *Journal of Stroke and Cerebrovascular Diseases*, vol. 23, 2014.

[19] B. Ovbiagele, "Phone-based Intervention under Nurse Guidance after Stroke: Concept for Lowering Blood Pressure after Stroke in Sub-Saharan Africa," *Journal of Stroke and Cerebrovascular Diseases*, vol. 24, 2015.

[20] J. van der Linden, V. Waights, Y. Rogers, and C. Taylor, "A blended design approach for pervasive healthcare: Bringing together users, experts and technology," *Health Informatics Journal*, vol. 18, 2012.

[21] D. Loudon, A. Macdonald, B. Carse, H. Thikey, L. Jones, P. Rowe, S. Uzor, M. Ayoade, and L. Baillie, "Developing visualisation software for rehabilitation: Investigating the requirements of patients, therapists and the rehabilitation process," *Health Informatics Journal*, vol. 18, 2012.

[22] J. Chen, W. Jin, X. Zhang, W. Xu, X. Liu, and C. Ren, "Telerehabilitation approaches for stroke patients: systematic review and meta-analysis of randomized controlled trials," *Journal of Stroke and Cerebrovascular Diseases*, 2015.

[23] G. Mazzaglia, C. Piccinni, A. Filippi, G. Sini, F. Lapi, E. Sessa, I. Cricelli, P. Cutroneo, G. Trifir, C. Cricelli, and A. Caputi, "Effects of a computerized decision support system in improving pharmacological management in high-risk cardiovascular patients: A cluster-randomized open-label controlled trial," *Health Informatics Journal*, 2014.

[24] J. H. Medicine, "History of stroke." http://www.hopkinsmedicine.org/healthlibrary/conditions/nervous_system_disorders/history_of_stroke_85P00223. Accessed: 2015-12-07.

[25] MedicineNet, "Definition of Apoplexy." http://www.medicinenet.com/script/main/art.asp?articlekey=9748. Accessed: 2015-12-07.

[26] S. Foundation, "Types of stroke." https://strokefoundation.com.au/about-stroke/types-of-stroke. Accessed: 2015-12-07.

[27] A. H. Association, "TIA (transient ischemic attack)." http://www.strokeassociation.org/STROKEORG/AboutStroke/TypesofStroke/TIA/TIA-Transient-Ischemic-Attack_UCM_310942_Article.jsp. Accessed: 2015-12-07.

[28] T. I. S. Center, "Warning signs." http://www.strokecenter.org/patients/about-stroke/warning-signs-of-stroke/. Accessed: 2015-12-07.

[29] S. Foundation, Stroke symptoms. https://strokefoundation.com.au/about-stroke/stroke-symptoms. Accessed: 2015-12-07.

[30] N. H. L. B. Institute, How is a stroke diagnosed?. http://www.nhlbi.nih.gov/health/health-topics/topics/stroke/diagnosis. Accessed: 2015-12-07.

[31] WebMD, "Stroke: life-threatening complications - topic overview." http://www.webmd.com/stroke/tc/stroke-life-threatening-complications-topic-overvi Accessed: 2015-12-07.

[32] N. H. L. B. Institute, "How is a stroke treated?." http://www.nhlbi.nih.gov/health/health-topics/topics/stroke/treatment. Accessed: 2015-12-07.

[33] E. C. Health, "Stroke." http://www.healthofchildren.com/S/Stroke.html. Accessed: 2015-12-07.

[34] Saebo, "The stages of stroke recovery." http://www.saebo.com/the-stages-of-stroke- Accessed: 2015-12-07.

[35] P. C. Physicians, "Philippine neurological association: annual report for year 2013." http://www.pcp.org.ph/documents/Annual%20Reports%20of%20PCP%

20Chapter%20and%20Society%20Presidents%|20FY%202013-2014/annual_report_aff_societies/PNA_2013AnnualReport.pdf. Accessed: 2015-12-07.

[36] S. S. Philippines, "Guidelines for primary and secondary prevention of stroke." http://www.strokesocietyphil.org/files/chapter-3.pdf. Accessed: 2015-12-09.

[37] A. W. Fitters, What is a web portal. http://www.atlanticwebfitters.ca/AboutCMS/WhatisaWebPortal/tabid/95/Default.aspx. Accessed: 2015-12-08.

[38] . M. H. Care, Medical records. http://www.muhealth.org/patient/medical-records/. Accessed: 2015-12-08.

[39] E. T. Serve, "Power to end stroke." http://www.empoweredtoserve.org/index.php/power-to-end-stroke/. Accessed: 2016-06-02.

X. Appendix

A. Source Codes

composer.json

```
{
"name": "laravel/laravel",
"description": "The Laravel Framework.",
"keywords": ["framework", "laravel"],
"license": "MIT",
"type": "project",
"require": {
"laravel/framework": "5.0.*",
"doctrine/dbal": "^2.5",
"illuminate/html": "^5.0",
"laracasts/flash": "^1.3",
"components/foundation": "^6.0",
"eduard44/foundation-pagination": "2.0.*"
},
"require-dev": {
"phpunit/phpunit": "~4.0",
"phpspec/phpspec": "~2.1"
},
"autoload": {
"classmap": [
"database"
],
"psr-4": {
"App\\": "app/"
}
},
"autoload-dev": {
"classmap": [
"tests/TestCase.php"
]
},
"scripts": {
"post-install-cmd": [
"php artisan clear-compiled",
"php artisan optimize"
],
"post-update-cmd": [
"php artisan clear-compiled",
"php artisan optimize"
],
"post-create-project-cmd": [
"php -r \"copy('.env.example', '.env');\"",
"php artisan key:generate"
]
},
"config": {
"preferred-install": "dist"
}
}
```

Auth Controller

```
<?php namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use Illuminate\Contracts\Auth\Guard;
use Illuminate\Contracts\Auth\Registrar;
use Illuminate\Foundation\Auth\
    AuthenticatesAndRegistersUsers;

class AuthController extends Controller {

/*
|--------------------------------------------------------------------------

| Registration & Login Controller
|--------------------------------------------------------------------------

|
| This controller handles the registration of new users
    , as well as the
| authentication of existing users. By default, this
    controller uses
| a simple trait to add these behaviors. Why don't you
    explore it?
|
*/

use AuthenticatesAndRegistersUsers;

/**
* The url to redirect to after registration or login
*
* @var string
*/
protected $redirectTo = '/';

/**
* Create a new authentication controller instance.
*
* @param  \Illuminate\Contracts\Auth\Guard  $auth
* @param  \Illuminate\Contracts\Auth\Registrar
    $registrar
*/
public function __construct(Guard $auth, Registrar
    $registrar) {
$this->auth = $auth;
$this->registrar = $registrar;
$this->middleware('guest', ['except' => 'getLogout']);
}

}
```

Authenticates And Registers Users Trait

```
<?php namespace Illuminate\Foundation\Auth;

use App\Models\Hospital;
use App\User;
use Illuminate\Http\Request;
```

```
use Illuminate\Contracts\Auth\Guard;
use Illuminate\Contracts\Auth\Registrar;

trait AuthenticatesAndRegistersUsers {

/**
* The Guard implementation.
*
* @var Guard
*/
protected $auth;

/**
* The registrar implementation.
*
* @var Registrar
*/
protected $registrar;

/**
* The db_lister implementation.
*
* @var DBLister
*/
protected $db_lister;

/**
* Show the application registration form.
*
* @return \Illuminate\Http\Response
*/
public function getRegister() {
$view = view('auth.register');
$view->title = 'Register | Philippine Stroke Portal';

$view->profession_list = [
'mp' => 'Medical Professional',
'n' => 'Nutritionist',
'p' => 'Rehabilitation Medicine Doctor',
];
$view->hospital_list = Hospital::where('is_unlocked', '
    =', true)->lists('name', 'id');

return $view;
}

/**
* Handle a registration request for the application.
*
* @param  \Illuminate\Http\Request  $request
* @return \Illuminate\Http\Response
*/
public function postRegister(Request $request) {
$validator = $this->registrar->validator($request->all
    ());

if ($validator->fails()) {
$this->throwValidationException(
$request, $validator
);
}

$this->registrar->create($request->all());

return redirect($this->redirectPath());
}

/**
* Handle confirmation of e-mail address.
*
* @param string $confirmation_code
* @return \Illuminate\Http\Response
*/
public function confirm($confirmation_code) {
if(!$confirmation_code) {
return redirect($this->redirectPath());
}

$user = User::where('confirmation_code',
    $confirmation_code)->first();
if (!$user) {
return redirect($this->redirectPath());
}

$user->is_email_confirmed = true;
$user->confirmation_code = null;
$user->save();
return redirect($this->loginPath());
}



/**
* Show the application login form.
*
* @return \Illuminate\Http\Response
*/
public function getLogin() {
$view = view('auth.login');
$view->title = 'Login | Philippine Stroke Portal';

return $view;
}

/**
* Handle a login request to the application.
*
* @param  \Illuminate\Http\Request  $request
* @return \Illuminate\Http\Response
*/
public function postLogin(Request $request) {
$credentials1 = [
'username' => $request->input('username_or_email'),
'password' => $request->input('password'),
];
$credentials2 = [
'email' => $request->input('username_or_email'),
'password' => $request->input('password'),
];

$is_correct_credentials = false;
```

```
$is_email_confirmed = false;
$is_unlocked = false;

if ($this->auth->attempt($credentials1, $request->has('
    remember')) or
$this->auth->attempt($credentials2, $request->has('
    remember'))) { // Try the input as username then
    email
$is_correct_credentials = true;
$is_email_confirmed = $this->auth->user()->
    is_email_confirmed;
$is_unlocked = $this->auth->user()->is_unlocked;

if ($is_email_confirmed == false or $is_unlocked ==
    false)
$this->auth->logout(); // log user out if account is
    locked or e-mail address is not confirmed
}

if (! $is_correct_credentials) // Login failed
return redirect($this->loginPath())->withErrors(['
    credentials' => 'We were unable to sign you in.'])
    ;
else if (! $is_email_confirmed) // E-mail address is
    not yet confirmed
return redirect($this->loginPath())
->withErrors([
'credentials' => 'Your e-mail address is not yet
    confirmed.'
]);
else if (! $is_unlocked) // Account is locked
return redirect($this->loginPath())
->withErrors([
'credentials' => 'Your account is locked. You will
    receive an e-mail if it is unlocked.'
]);
else // Login successful
return redirect()->intended($this->redirectPath());
}

/**
* Get the failed login message.
*
* @return string
*/
protected function getFailedLoginMessage() {
return 'These credentials do not match our records.';
}

/**
* Log the user out of the application.
*
* @return \Illuminate\Http\Response
*/
public function getLogout() {
$this->auth->logout();
return redirect(property_exists($this, '
    redirectAfterLogout') ? $this->redirectAfterLogout
     : '/');
}

/**
* Get the post register / login redirect path.
*
* @return string
*/
public function redirectPath() {
if (property_exists($this, 'redirectPath')) {
return $this->redirectPath;
}

return property_exists($this, 'redirectTo') ? $this->
    redirectTo : '/';
}

/**
* Get the path to the login route.
*
* @return string
*/
public function loginPath() {
return property_exists($this, 'loginPath') ? $this->
    loginPath : '/auth/login';
}

}
```

Password Controller

```
<?php namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use Illuminate\Contracts\Auth\Guard;
use Illuminate\Contracts\Auth\PasswordBroker;
use Illuminate\Foundation\Auth\ResetsPasswords;

class PasswordController extends Controller {

/*
|--------------------------------------------------------------------------

| Password Reset Controller
|--------------------------------------------------------------------------

|
| This controller is responsible for handling password
    reset requests
| and uses a simple trait to include this behavior. You
    're free to
| explore this trait and override any methods you wish
    to tweak.
|
*/

use ResetsPasswords;

/**
* Create a new password controller instance.
*
* @param  \Illuminate\Contracts\Auth\Guard  $auth
```

```
* @param  \Illuminate\Contracts\Auth\PasswordBroker
    $passwords
*/
public function __construct(Guard $auth, PasswordBroker
     $passwords) {
$this->auth = $auth;
$this->passwords = $passwords;

$this->middleware('guest');
}

}
```

Resets Passwords Trait

```
<?php namespace Illuminate\Foundation\Auth;

use Illuminate\Http\Request;
use Illuminate\Contracts\Auth\Guard;
use Illuminate\Contracts\Auth\PasswordBroker;
use Symfony\Component\HttpKernel\Exception\
    NotFoundHttpException;

trait ResetsPasswords {

/**
* The Guard implementation.
*
* @var Guard
*/
protected $auth;

/**
* The password broker implementation.
*
* @var PasswordBroker
*/
protected $passwords;

/**
* Display the form to request a password reset link.
*
* @return Response
*/
public function getEmail() {
$view = view('auth.password');
$view->title = 'Reset Password | Philippine Stroke
    Portal';

return $view;
}

/**
* Send a reset link to the given user.
*
* @param  Request  $request
* @return Response
*/
public function postEmail(Request $request) {
$this->validate($request, ['email' => 'required|email'
    ]);

$response = $this->passwords->sendResetLink($request->
    only('email'), function($m) {
$m->subject($this->getEmailSubject());
});

switch ($response) {
case PasswordBroker::RESET_LINK_SENT:
return redirect()->back()->with('status', trans(
    $response));

case PasswordBroker::INVALID_USER:
return redirect()->back()->withErrors(['email' => trans
    ($response)]);
}
}

/**
* Get the e-mail subject line to be used for the reset
    link email.
*
* @return string
*/
protected function getEmailSubject() {
return isset($this->subject) ? $this->subject : 'Your
    Password Reset Link';
}

/**
* Display the password reset view for the given token.
*
* @param  string  $token
* @return Response
*/
public function getReset($token = null) {
if (is_null($token)) {
throw new NotFoundHttpException;
}

$view = view('auth.reset');
$view->title = 'Reset Password | Philippine Stroke
    Portal';

return $view->with('token', $token);
}

/**
* Reset the given user's password.
*
* @param  Request  $request
* @return Response
*/
public function postReset(Request $request) {
$this->validate($request, [
'token' => 'required',
'email' => 'required|email',
'password' => 'required|confirmed',
]);
```

```
$credentials = $request->only(
'email', 'password', 'password_confirmation', 'token'
);

$response = $this->passwords->reset($credentials,
    function($user, $password) {
$user->password = bcrypt($password);
$user->save();
$this->auth->login($user);
});

switch ($response) {
case PasswordBroker::PASSWORD_RESET:
return redirect($this->redirectPath());

default:
return redirect()->back()
->withInput($request->only('email'))
->withErrors(['email' => trans($response)]);
}
}

/**
* Get the post register / login redirect path.
*
* @return string
*/
public function redirectPath() {
if (property_exists($this, 'redirectPath')) {
return $this->redirectPath;
}

return property_exists($this, 'redirectTo') ? $this->
    redirectTo : '/';
}

}
```

Events Controller

```
<?php namespace App\Http\Controllers\HomeControllers;

use App\Http\Middleware\LocalAdminMiddleware;
use App\Http\Requests;
use App\Http\Controllers\Controller;

use App\Models\Event;
use App\Models\Hospital;
use Carbon\Carbon;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Validator;

use Chromabits\Pagination\FoundationPresenter;

class EventsController extends Controller {

/**
* ForumsController constructor.
*
* @param \App\Http\Middleware\LocalAdminMiddleware
    $localAdminMiddleware
*/
public function __construct(LocalAdminMiddleware
    $localAdminMiddleware) {
//$this->middleware('localAdmin', ['only' => ['create',
     'store', 'edit', 'update', 'destroy']]);
$this->localAdminMiddleware = $localAdminMiddleware;
}

/**
* Display a listing of events.
*
* @return \Illuminate\Http\Response
*/
public function viewAllEvents() {
$view = view('events.view_all_events');
$view->title = 'Announcements and Events | Philippine
    Stroke Portal';
$view->events = Event::latest()->paginate(5);
$view->events->setPath('announcements-and-events');
$view->page_links = $view->events->render(new
    FoundationPresenter($view->events));

return $view;
}

/**
* Display a listing of the events.
*
* @param string $hospital_name
* @return \Illuminate\Http\Response
*/
public function viewEventsInHospital($hospital_name) {
$hospital = Hospital::where('name', $hospital_name)->
    first();
if ($hospital) {
$view = view('events.view_events_in_hospital');
$view->title = 'Announcements and Events of the ' .
    $hospital->name . ' | Philippine Stroke Portal';
$view->hospital = $hospital;

$view->events = $view->hospital->events()->latest()->
    paginate(5);
$view->events->setPath($hospital_name);
$view->page_links = $view->events->render(new
    FoundationPresenter($view->events));

$user = Auth::user();
$view->is_local_admin_and_hospital = $this->
    localAdminMiddleware->checkLocalAdminAndHospital(
    $user,
$hospital->id
);

return $view;
} else {
return redirect('/');
}
}
```

```
/**
* Show the form for creating a new event.
*
* @param int $hospital_id
* @return \Illuminate\Http\Response
*/
public function create($hospital_id) {
$view = view('events.create');
$view->hospital = Hospital::findOrFail($hospital_id);
$view->title = 'Add Announcement and Event from the ' .
      $view->hospital->name . ' | Philippine Stroke
     Portal';
$view->current_date = Carbon::now();

return $view;
}

/**
* Store a newly created event in storage.
*
* @param int $hospital_id
* @param \Illuminate\Http\Request $request
* @return \Illuminate\Http\Response
*/
public function store($hospital_id, Request $request) {
$hospital = Hospital::findOrFail($hospital_id);

$validator = Validator::make($request->all(), [
'heading' => 'required',
'description' => 'required',
]);

if ($validator->fails()) {
return redirect('announcements-and-events/hospitals/' .
      $hospital_id . '/create')
->with('alert', 'There is a problem with your input.')
->withErrors($validator)
->withInput();
}

Event::create([
'hospital_id' => $hospital_id,
'heading' => $request->input('heading'),
'description' => $request->input('description'),
'date_of_event' => $request->input('date_of_event'),
'time_of_event' => $request->input('time_of_event'),
]);

return redirect('announcements-and-events/' . $hospital
     ->name)
->with('success', 'Announcement and event created.');
}

/**
* Show the form for editing the specified event.
*
* @param  int  $hospital_id
* @param  int  $event_id
* @return Response
*/
public function show($hospital_id, $event_id) {
$hospital = Hospital::findOrFail($hospital_id);
$event = Event::findOrFail($event_id);
$user = Auth::user();

if ($event->hospital) {
if ($event->hospital->id == $hospital_id) {
$view = view('events.show');
$view->title = $event->heading . ' | Philippine Stroke
     Portal';
$view->event = $event;
$view->hospital = $hospital;
$view->is_local_admin_and_hospital = $this->
     localAdminMiddleware->checkLocalAdminAndHospital(
     $user,
$hospital_id
);

return $view;
} else {
return redirect('/');
}
} else {
return redirect('/');
}
}

/**
* Show the form for editing the specified event.
*
* @param  int  $hospital_id
* @param  int  $event_id
* @return Response
*/
public function edit($hospital_id, $event_id) {
$view = view('events.edit');
$view->title = 'Edit Announcement and Event | 
     Philippine Stroke Portal';
$view->hospital = Hospital::findOrFail($hospital_id);
$view->event = Event::findOrFail($event_id);
$user = Auth::user();
$view->is_local_admin_and_hospital = $this->
     localAdminMiddleware->checkLocalAdminAndHospital(
     $user,
$hospital_id
);
$view->current_date = Carbon::now();

return $view;
}

/**
* Update the specified event in storage.
*
* @param  int  $hospital_id
* @param  int  $event_id
* @param \Illuminate\Http\Request $request
* @return Response
*/
```

```
public function update($hospital_id, $event_id, Request
      $request) {
Hospital::findOrFail($hospital_id);
$event = Event::findOrFail($event_id);

$validator = Validator::make($request->all(), [
'heading' => 'required',
'description' => 'required',
]);

if ($validator->fails()) {
return redirect('announcements-and-events/hospitals/' .
      $hospital_id . '/' . $event_id . '/edit')
->with('alert', 'There is a problem with your input.')
->withErrors($validator)
->withInput();
}

$event->update([
'hospital_id' => $hospital_id,
'heading' => $request->input('heading'),
'description' => $request->input('description'),
'date_of_event' => $request->input('date_of_event'),
'time_of_event' => $request->input('time_of_event'),
]);

return redirect('announcements-and-events/hospitals/' .
      $hospital_id . '/' . $event_id)
->with('success', 'Announcement and event updated.');;
}

/**
* Remove the specified event from storage.
*
* @param  int  $hospital_id
* @param  int  $event_id
* @return Response
*/
public function destroy($hospital_id, $event_id) {
$hospital = Hospital::findOrFail($hospital_id);
$event = Event::findOrFail($event_id);

$event->delete();

return redirect('announcements-and-events/' . $hospital
     ->name)
->with('success', 'Announcement and event deleted.');;
}

}
```

Forums Controller

```
<?php namespace App\Http\Controllers\HomeControllers;

use App\Http\Middleware\SupervisorMiddleware;
use App\Http\Requests;
use App\Http\Controllers\Controller;

use App\Models\Forum;
use App\Models\ForumCategory;
use App\Models\ForumComment;
use App\Models\Notification;
use App\Models\Supervisor;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Validator;

use Chromabits\Pagination\FoundationPresenter;

class ForumsController extends Controller {

/**
* ForumsController constructor.
*
* @param \App\Http\Middleware\SupervisorMiddleware
     $supervisorMiddleware
*/
public function __construct(SupervisorMiddleware
     $supervisorMiddleware) {
$this->middleware('auth', ['only' => ['create', 'store'
     , 'addReply', 'moderate']]);
$this->middleware('supervisor', ['only' => ['destroy',
     'destroyReply']]);
$this->supervisorMiddleware = $supervisorMiddleware;
}

/**
* Display a listing of the forums.
*
* @return Response
*/
public function viewAllForums() {
$view = view('forums.view_all_forums');
$view->title = 'Forum Threads | Philippine Stroke
     Portal';

$view->forums = Forum::latest()->paginate(5);
$view->forums->setPath('all');
$view->page_links = $view->forums->render(new
     FoundationPresenter($view->forums));

return $view;
}

/**
* Display a listing of the forums on the specified
     category.
*
* @param string $category
* @return Response
*/
public function viewForumsInCategory($category) {
$forum_category = ForumCategory::where('name',
     $category)->first();

if ($forum_category) {
$view = view('forums.view_forums_in_category');
$view->title = $forum_category->name . ' | Philippine
     Stroke Portal';
```

```
$view->forum_category = $forum_category;
$view->category = $category;

$view->forums = $forum_category->forums()->latest()->
    paginate(5);
$view->forums->setPath($category);
$view->page_links = $view->forums->render(new
    FoundationPresenter($view->forums));

return $view;

} else {
return redirect('/');
}
}

/**
* Show the form for creating a new forum.
*
* @return Response
*/
public function create() {
$view = view('forums.create');
$view->title = 'Add Forum Thread | Philippine Stroke
    Portal';
$view->forum_categories = ForumCategory::lists('name',
    'id');

return $view;
}

/**
* Store a newly created forum in storage.
*
* @param \Illuminate\Http\Request $request
* @return Response
*/
public function store(Request $request) {
$validator = Validator::make($request->all(), [
'title' => 'required',
'forum_category' => 'required',
'description' => 'required',
]);

if ($validator->fails()) {
return redirect('forums/create')
->with('alert', 'There is a problem with your input.')
->withErrors($validator)
->withInput();
}

$user = Auth::user();
if ($user) {
$forum = Forum::create([
'user_id' => $user->id,
'forum_category_id' => $request->input('forum_category'
    ),
'title' => $request->input('title'),
'description' => $request->input('description'),
]);

$supervisors = Supervisor::where('forum_category_id',
    $request->input('forum_category'))->get();
foreach ($supervisors as $supervisor) {
if ($supervisor->user and $forum->forumCategory) {
$notifier = $user;
$description = 'User ' . $notifier->first_name . ' ' .
    $notifier->last_name .
' posted a new ' . $forum->forumCategory->name . '
    forum thread on ' .
$forum->created_at->format('F d, Y, g:i A') . '.';
Notification::create([
'user_id' => $supervisor->user->id,
'description' => $description,
'link' => 'forum-threads/' . $forum->forumCategory->
    name,
]);
}
}
}

return redirect('forum-threads/all')->with('success', '
    Forum thread created.');
}

/**
* Show forum.
*
* @param string $category
* @param int $forum_id
* @return \Illuminate\Http\Response
*/
public function show($category, $forum_id) {
$forum_category = ForumCategory::where('name',
    $category)->first();
if ($forum_category) {
$forum = Forum::findOrFail($forum_id);
$view = view('forums.show');
$view->title = $forum->title . ' | Philippine Stroke
    Portal';
$view->forum = $forum;
$view->forum_category = $forum_category;
$view->forum_comments = $forum->forumComments;
$user = Auth::user();

if ($user) {
$view->is_supervisor = $this->supervisorMiddleware->
    checkSupervisorAndForumCategory($user,
$forum->forumCategory->id);
$view->can_moderate = $this->supervisorMiddleware->
    checkSupervisorAndForum($user,
$forum->id);
} else {
$view->is_supervisor = false;
$view->can_moderate = false;
}

return $view;
} else {
return redirect('forum-threads/all');
```

```
}
}

/**
* Add reply in forum.
*
* @param string $category
* @param int $forum_id
* @param \Illuminate\Http\Request $request
* @return \Illuminate\Http\Response
*/
public function addReply($category, $forum_id, Request
    $request) {
$forum_category = ForumCategory::where('name',
    $category)->first();
if ($forum_category) {
$forum = Forum::findOrFail($forum_id);
$user = Auth::user();

if ($request->input('comment')) {
$forum_comment = new ForumComment();
$forum_comment['user_id'] = $user->id;
$forum_comment['forum_id'] = $forum->id;
$forum_comment['comment'] = $request->input('comment');
$forum_comment->save();

$forum_comments = ForumComment::where('forum_id',
    $forum->id)->get();
foreach ($forum_comments as $old_forum_comment) {
if ($old_forum_comment->user) {
$notifier = $user;
$description = 'User ' . $notifier->first_name . ' ' .
    $notifier->last_name .
    ' replied in a ' . $forum_category->name . ' forum
        thread you participated in on ' .
    $forum_comment->created_at->format('F d, Y, g:i A')
        . '.';
Notification::create([
    'user_id' => $user->id,
    'description' => $description,
    'link' => 'forum-threads/' . $forum_category->name
        . '/' . $forum->id,
]);
}
}
}

return redirect('forum-threads/' . $forum_category->
    name . '/' . $forum_id);

} else {
return redirect('forum-threads/all');
}
}

/**
* Moderate forum thread.
*
* @param string $category
* @param int $forum_id
* @return \Illuminate\Http\Response
*/
public function moderate($category, $forum_id) {
$forum_category = ForumCategory::where('name',
    $category)->first();
$user = Auth::user();
if ($forum_category and $user->supervisor) {
$forum = Forum::findOrFail($forum_id);
if (! $forum->supervisor and $forum->forumCategory and
    $user->supervisor->forumCategory) {
if (strcmp($forum->forumCategory->name, $user->
    supervisor->forumCategory->name) == 0) {
$forum['supervisor_id'] = $user->supervisor->id;
$forum->save();
return redirect('forum-threads/' . $category . '/' .
    $forum_id)
->with('success', 'You can now moderate this forum
    thread.');
}
}
}

return redirect('forum-threads/' . $category . '/' .
    $forum_id);
}

/**
* Remove the specified forum from storage.
*
* @param string $category
* @param int $forum_id
* @return Response
*/
public function destroy($category, $forum_id) {
$forum_category = ForumCategory::where('name',
    $category)->first();
if ($forum_category) {
$forum = Forum::findOrFail($forum_id);
if ($forum->forumComments) {
foreach ($forum->forumComments as $forumComment) {
$forumComment->delete();
}
}
$forum->delete();
return redirect('forum-threads/all')->with('success', '
    Forum thread deleted.');
}

return redirect('forum-threads/all');
}

/**
* Remove the specified forum reply from storage.
*
* @param string $category
* @param int $forum_id
* @param int $forum_comment_id
* @return Response
*/
public function destroyReply($category, $forum_id,
```

```
    $forum_comment_id) {
$forum_category = ForumCategory::where('name',
    $category)->first();
if ($forum_category) {
$forum = Forum::findOrFail($forum_id);
if ($forum->forumComments) {
$forum_comment = $forum->forumComments()->find(
    $forum_comment_id);
$forum_comment->delete();
return redirect('forum-threads/' . $category . '/' .
    $forum_id)
->with('success', 'Forum thread reply deleted.');
}
}

return redirect('forum-threads/' . $category . '/' .
    $forum_id);
}

}
```

Home Controller

```
<?php namespace App\Http\Controllers\HomeControllers;

use App\Http\Controllers\Controller;
use App\Models\Event;
use App\Models\Forum;
use App\Models\MedicalResource;
use App\Models\PortalSectionItem;
use App\Services\Retrievers\SelectFieldListsRetriever;
use App\User;

class HomeController extends Controller {

/*
|--------------------------------------------------------------------------
| Home Controller
|--------------------------------------------------------------------------
|
| This controller renders your application's "dashboard
    " for users that
| are authenticated. Of course, you are free to change
    or remove the
| controller as you wish. It is just here to get your
    app started!
|
*/

/**
* HomeController constructor.
*
* @param \App\Services\Retrievers\
    SelectFieldListsRetriever
    $selectFieldListsRetriever
*/
public function __construct(SelectFieldListsRetriever
    $selectFieldListsRetriever) {
$this->middleware('auth', ['only' => ['showUser',]]);
$this->selectFieldListsRetriever =
    $selectFieldListsRetriever;
}

/**
* Show the application dashboard to the user.
*
* @return Response
*/
public function index() {
$view = view('home.home');
$view->title = "Philippine Stroke Portal";

$view->medical_resources = MedicalResource::where('
    is_approved', true)->latest()->take(5)->get();
$view->events = Event::latest()->take(5)->get();
$view->forums = Forum::latest()->take(5)->get();
$view->image_extensions = $this->
    selectFieldListsRetriever->getImageExtensions();
$view->video_extensions = $this->
    selectFieldListsRetriever->getVideoExtensions();
return $view;
}

/**
* Show the "About Us" page.
*
* @return \Illuminate\Http\Response
*/
public function aboutUs() {
$view = view('home.about_us');
$view->title = 'About Us | Philippine Stroke Portal';

return $view;
}

/**
* Show specified user.
*
* @param string $username
* @return \Illuminate\Http\Response
*/
public function showUser($username) {
$user = User::where('username', $username)->first();
if ($user) {
$view = view('home.show_user');
$view->title = $user->username . ' | Philippine Stroke
    Portal';
$view->user = $user;

return $view;
} else {
return redirect('/');
}
}

}
```

Medical Resources Controller

```php
<?php namespace App\Http\Controllers\HomeControllers;

use App\Http\Middleware\MedicalResourceMiddleware;
use App\Http\Middleware\StaffMiddleware;
use App\Http\Middleware\SupervisorMiddleware;
use App\Http\Requests;
use App\Http\Controllers\Controller;

use App\Models\ForumCategory;
use App\Models\MedicalResource;
use App\Models\Notification;
use App\Models\Supervisor;
use App\Services\Retrievers\SelectFieldListsRetriever;
use App\Services\Validators\MedicalResourcesValidator;
use Carbon\Carbon;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\File;

use Chromabits\Pagination\FoundationPresenter;

class MedicalResourcesController extends Controller {

/*
|--------------------------------------------------------------------------

| Medical Resources Controller
|--------------------------------------------------------------------------

|
| This controller handles the presentation of all the medical
| resources, as well as the forms for creating,
| editing, downloading, and deleting them.
|
*/

/**
* MedicalResourcesController constructor.
*
* @param \App\Services\Validators\
    MedicalResourcesValidator
    $medicalResourcesValidator
* @param \App\Http\Middleware\MedicalResourceMiddleware
    $medicalResourceMiddleware
* @param \App\Http\Middleware\SupervisorMiddleware
    $supervisorMiddleware
* @param \App\Http\Middleware\StaffMiddleware
    $staffMiddleware
* @param \App\Services\Retrievers\
    SelectFieldListsRetriever
    $selectFieldListsRetriever
*/
public function __construct(
MedicalResourcesValidator $medicalResourcesValidator,
    MedicalResourceMiddleware
    $medicalResourceMiddleware,
SupervisorMiddleware $supervisorMiddleware,
    StaffMiddleware $staffMiddleware,
SelectFieldListsRetriever $selectFieldListsRetriever
) {
$this->middleware('auth', ['only' => ['create', 'store'
    , 'edit', 'update', 'destroy', 'approve',]]);
$this->middleware('staff', ['only' => ['create', 'store
    ',]]);
$this->middleware('medicalResource', ['only' => ['edit'
    , 'update', 'destroy',]]);
$this->middleware('supervisor', ['only' => ['approve',
    'destroy',]]);

$this->medicalResourcesValidator =
    $medicalResourcesValidator;
$this->medicalResourceMiddleware =
    $medicalResourceMiddleware;
$this->supervisorMiddleware = $supervisorMiddleware;
$this->staffMiddleware = $staffMiddleware;
$this->selectFieldListsRetriever =
    $selectFieldListsRetriever;
}

/**
* Display a listing of the medical resources.
*
* @return \Illuminate\Http\Response
*/
public function viewAllMedicalResources() {
$view = view('medical_resources.index');
$view->title = 'Medical Resources | Philippine Stroke
    Portal';
$view->index = true;
$user = Auth::user();

$view->medical_resources = MedicalResource::where('
    is_approved', true)->latest()->paginate(5);
$view->medical_resources->setPath('medical-resources');
$view->page_links = $view->medical_resources->render(
    new FoundationPresenter($view->medical_resources))
    ;

$view->is_staff = $this->staffMiddleware->checkIfStaff(
    $user);
$view->image_extensions = $this->
    selectFieldListsRetriever->getImageExtensions();
$view->video_extensions = $this->
    selectFieldListsRetriever->getVideoExtensions();

return $view;
}

/**
* Display a listing of the medical resources based on
    forum category.
*
* @param string $category
* @return \Illuminate\Http\Response
*/
public function viewMedicalResourcesInCategory(
    $category) {
$forum_category = ForumCategory::where('name',
    $category)->first();
```

```
if ($forum_category) {
$view = view('medical_resources.index');
$view->title = 'Medical Resources | Philippine Stroke
    Portal';
$view->category = $category;

$view->medical_resources = MedicalResource::where('
    forum_category_id', $forum_category->id)
->where('is_approved', true)->latest()->paginate(5);

$user = Auth::user();
if ($user) {
if ($user->supervisor) {
if ($user->supervisor->forumCategory) {
if (strcmp($user->supervisor->forumCategory->name,
    $category) == 0) {
    $view->medical_resources = MedicalResource::where('
        forum_category_id', $forum_category->id)
        ->orderBy('is_approved', 'asc')->latest()->
            paginate(5);
}
}
}
}

$view->medical_resources->setPath($category);
$view->page_links = $view->medical_resources->render(
    new FoundationPresenter($view->medical_resources))
    ;

$view->is_staff = $this->staffMiddleware->checkIfStaff(
    $user);
$view->image_extensions = $this->
    selectFieldListsRetriever->getImageExtensions();
$view->video_extensions = $this->
    selectFieldListsRetriever->getVideoExtensions();

return $view;

} else {
return redirect('/');
}
}

/**
* Download medical resource file.
*
* @param int $medical_resource_id
* @return \Illuminate\Http\\Illuminate\Http\Response
*/
public function download($medical_resource_id) {
$medical_resource = MedicalResource::findOrFail(
    $medical_resource_id);
$file = $medical_resource->file;
if ($file) {
$filename = $file->filename;
$path_to_file = $file->path . '/' . $filename;
$headers = [
'Content-Type' => $file->mimeType,
];
return response()->download($path_to_file, $filename,
    $headers);
} else {
return response();
}
}

/**
* Show the form for creating a new medical resource.
*
* @return \Illuminate\Http\Response
*/
public function create() {
$view = view('medical_resources.create');
$view->title = 'Add Medical Resource | Philippine
    Stroke Portal';
$view->forum_categories = ForumCategory::lists('name',
    'id');
$view->current_date = Carbon::now();

return $view;
}

/**
* Store a newly created medical resource in storage.
*
* @param \Illuminate\Http\Request $request
* @return \Illuminate\Http\Response
*/
public function store(Request $request) {
$validator = $this->medicalResourcesValidator->validate
    ($request->all(), $request->hasFile('file'),
$request->file('file'));

if ($validator->fails()) {
return redirect('medical-resources/create')
->with('alert', 'There is a problem with your input.')
->withErrors($validator)
->withInput();
}

$user = Auth::user();
$medical_resource = $this->medicalResourcesValidator->
    store($user->id, $request->all(), $request->
    hasFile('file'),
$request->file('file'));

$supervisors = Supervisor::where('forum_category_id',
    $request->input('category'))->get();
foreach ($supervisors as $supervisor) {
if ($supervisor->user and $medical_resource->
    forumCategory) {
$notifier = $user;
$description = 'User ' . $notifier->first_name . ' ' .
    $notifier->last_name .
' posted a new ' . $medical_resource->forumCategory->
    name . ' medical resource on ' .
$medical_resource->created_at->format('F d, Y, g:i A')
    . '.';
```

```
Notification::create([
'user_id' => $supervisor->user->id,
'description' => $description,
'link' => 'medical-resources/category/' .
    $medical_resource->forumCategory->name,
]);
}
}

if ($this->supervisorMiddleware->checkIfSupervisor(
    $user)) {
$success_message = "Medical resource created.";
} else {
$success_message = "Medical resource created. It will
    be posted after approval from a supervisor.";
}
return redirect('medical-resources')->with('success',
    $success_message);
}

/**
* Show the medical resource.
*
* @param int $medical_resource_id
* @return \Illuminate\Http\Response
*/
public function show($medical_resource_id) {
$view = view('medical_resources.show');
$view->title = 'Medical Resource | Philippine Stroke
    Portal';
$view->medical_resource = MedicalResource::findOrFail(
    $medical_resource_id);

$user = Auth::user();
$view->is_supervisor = $this->supervisorMiddleware->
    checkIfSupervisor($user);
$view->can_manage_medical_resource = $this->
    medicalResourceMiddleware->
    checkIfCanManageMedicalResource($user,
$medical_resource_id);
$view->image_extensions = $this->
    selectFieldListsRetriever->getImageExtensions();
$view->video_extensions = $this->
    selectFieldListsRetriever->getVideoExtensions();

return $view;
}

/**
* Show the form for editing the specified medical
    resource.
*
* @param int $medical_resource_id
* @return \Illuminate\Http\Response
*/
public function edit($medical_resource_id) {
$view = view('medical_resources.edit');
$view->title = 'Edit Medical Resource | Philippine
    Stroke Portal';
$view->medical_resource = MedicalResource::findOrFail(
    $medical_resource_id);
$view->forum_categories = ForumCategory::lists('name',
    'id');

$user = Auth::user();
$view->is_supervisor = $this->supervisorMiddleware->
    checkIfSupervisor($user);
$view->can_manage_medical_resource = $this->
    medicalResourceMiddleware->
    checkIfCanManageMedicalResource($user,
$medical_resource_id);
$view->image_extensions = $this->
    selectFieldListsRetriever->getImageExtensions();
$view->video_extensions = $this->
    selectFieldListsRetriever->getVideoExtensions();
$view->current_date = Carbon::now();

return $view;
}

/**
* Update the specified medical resource in storage.
*
* @param int $medical_resource_id
* @param \Illuminate\Http\Request $request
* @return \Illuminate\Http\Response
*/
public function update($medical_resource_id, Request
    $request) {
MedicalResource::findOrFail($medical_resource_id);
$validator = $this->medicalResourcesValidator->validate
    ($request->all(), $request->hasFile('file'),
$request->file('file'));

if ($validator->fails()) {
return redirect('medical-resources/' .
    $medical_resource_id . '/edit')
->with('alert', 'There is a problem with your input.')
->withErrors($validator)
->withInput();
}

$user = Auth::user();
$this->medicalResourcesValidator->update(
    $medical_resource_id, $user->id, $request->all(),
$request->hasFile('file'), $request->file('file'));

if ($this->supervisorMiddleware->checkIfSupervisor(
    $user)) {
$success_message = "Medical resource updated.";
} else {
$success_message = "Medical resource updated. It will
    be posted after approval from a supervisor.";
}

return redirect('medical-resources/' .
    $medical_resource_id)->with('success',
    $success_message);
}
```

```
/**
* Remove the specified medical resource from storage.
*
* @param int $medical_resource_id
* @return \Illuminate\Http\Response
*/
public function destroy($medical_resource_id) {
$medical_resource = MedicalResource::findOrFail(
    $medical_resource_id);
$file = $medical_resource->file;

if ($file) {
File::delete($file->path . '/' . $file->filename);
$file->delete();
}
$medical_resource->delete();

return redirect('medical-resources')->with('success', '
    Medical resource deleted.');
}

/**
* Approve medical resource.
*
* @param int $medical_resource_id
* @return \Illuminate\Http\Response
*/
public function approve($medical_resource_id) {
$medical_resource = MedicalResource::findOrFail(
    $medical_resource_id);
$medical_resource['is_approved'] = true;
$medical_resource->save();

$user = Auth::user();
if ($user->supervisor and $medical_resource->user and
    $medical_resource->forumCategory) {
if ($user->supervisor->forumCategory) {
if (strcmp($user->supervisor->forumCategory->name,
    $medical_resource->forumCategory->name) == 0) {
$notifier = $user;
$description = 'User ' . $notifier->first_name . ' ' .
    $notifier->last_name .
' approved your ' . $medical_resource->forumCategory->
    name . ' medical resource on ' .
$medical_resource->created_at->format('F d, Y, g:i A')
    . '.';
Notification::create([
'user_id' => $medical_resource->user->id,
'description' => $description,
'link' => 'medical-resources/category/' .
    $medical_resource->forumCategory->name,
]);
}
}
}

return redirect('medical-resources/' .
    $medical_resource_id)->with('success', 'Medical
    resource approved.');
}
```

```
}
```

Notifications Controller

```
<?php namespace App\Http\Controllers\HomeControllers;

use App\Http\Requests;
use App\Http\Controllers\Controller;

use App\Models\Notification;
use Chromabits\Pagination\FoundationPresenter;
use Illuminate\Support\Facades\Auth;

class NotificationsController extends Controller {

/**
* NotificationsController constructor.
*
*/
public function __construct(){
$this->middleware('auth');
}

/**
* Display a listing of the notifications.
*
* @return \Illuminate\Http\Response
*/
public function index(){
$user = Auth::user();
$unread_notifications = Notification::where('user_id',
    $user->id)->where('is_seen', false)->latest()->get
    ();

$view = view('home.notifications');
$view->unread_notifications = count(
    $unread_notifications);
if ($view->unread_notifications > 0)
$view->title = '(' . $view->unread_notifications . ')
    Notifications | Philippine Stroke Portal';
else
$view->title = 'Notifications | Philippine Stroke
    Portal';

$view->user = $user;
$view->notifications = $user->notifications()->latest()
    ->paginate(5);
$view->notifications->setPath('notifications');
$view->page_links = $view->notifications->render(new
    FoundationPresenter($view->notifications));

return $view;
}

/**
* Redirect user to notification link.
*
* @param int $notification_id
* @return \Illuminate\Http\Response
```

```
*/
public function viewNotification($notification_id) {
$notification = Notification::findOrFail(
    $notification_id);
$user = Auth::user();

if ($notification->user) {
if ($user->id == $notification->user->id) {
$notification['is_seen'] = true;
$notification->save();

if ($notification->link)
return redirect($notification->link);
}
}

return redirect('/');
}

}
```

Hospitals Controller

```
<?php namespace App\Http\Controllers\Hospitals;

use App\Http\Middleware\
    MaintainPatientAccountMiddleware;
use App\Http\Middleware\ProfessionalMiddleware;
use App\Http\Requests;
use App\Http\Controllers\Controller;

use App\Services\Registrar;
use App\Services\Validators\HospitalsValidator;

class HospitalsController extends Controller {

/*
|--------------------------------------------------------------------------
| Hospitals Controller
|--------------------------------------------------------------------------
|
| This controller handles the presentation of all the
    hospitals
| and clinics, as well as the forms for creating,
| editing, locking, and unlocking them.
|
| Registering to and leaving a hospital are also
    included here. In
| addition, the management of its staff and its
    patients
| are done through this controller.
|
*/

use ManagesHospitals;

use ShowsPatientsInHospital;

use ShowsStaffInHospital;

use ManagesMemberships;

/**
* Create a new hospitals controller instance
*
* @param \App\Services\Validators\HospitalsValidator
    $hospitalsValidator
* @param \App\Http\Middleware\ProfessionalMiddleware
    $professionalMiddleware
* @param \App\Http\Middleware\
    MaintainPatientAccountMiddleware
    $maintainPatientAccountMiddleware
* @param \App\Services\Registrar $registrar
*/
public function __construct(HospitalsValidator
    $hospitalsValidator, ProfessionalMiddleware
    $professionalMiddleware,
MaintainPatientAccountMiddleware
    $maintainPatientAccountMiddleware, Registrar
    $registrar) {
$this->middleware('auth', ['only' => ['show', 'edit', '
    update', 'lockHospital', 'unlockHospital',
'registerInHospital', 'leaveHospital', 'showStaffMember
    ', 'lockStaffMember', 'unlockStaffMember',
'emailStaffMember', 'showPatient', 'lockPatient', '
    unlockPatient', 'emailPatient', 'maintainPatient'
    ]]);

$this->middleware('staff', ['only' => ['
    registerInHospital', 'maintainPatient']]);

$this->middleware('maintainPatient', ['only' => ['
    maintainPatient']]);

$this->middleware('unlockedHospital', ['only' => ['edit
    ', 'update', 'registerInHospital',
'leaveHospital', 'lockStaffMember', 'unlockStaffMember'
    , 'emailStaffMember', 'lockPatient',
'unlockPatient', 'emailPatient', 'maintainPatient']]);

$this->middleware('localAdmin', ['only' => ['edit', '
    update', 'lockStaffMember', 'unlockStaffMember',
'emailStaffMember', 'lockPatient', 'unlockPatient', '
    emailPatient']]);

$this->middleware('systemAdmin', ['only' => ['
    lockHospital', 'unlockHospital']]);

$this->hospitalsValidator = $hospitalsValidator;
$this->professionalMiddleware = $professionalMiddleware
    ;
$this->maintainPatientAccountMiddleware =
    $maintainPatientAccountMiddleware;
$this->registrar = $registrar;
}

}
```

Manages Hospitals Trait

```
<?php namespace App\Http\Controllers\Hospitals;

use App\Models\Hospital;
use App\Http\Requests;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

trait ManagesHospitals
{

/**
* Display a listing of the hospital records.
*
* @return \Illuminate\Http\Response
*/
public function index() {
$view = view('hospitals.index');
$view->title = 'Hospitals and Clinics | Philippine
    Stroke Portal';
if (! Auth::check() or ! Auth::user()->systemAdmin)
$view->hospitals = Hospital::where('is_unlocked', true)
    ->orderBy('name', 'asc')->get();
else
$view->hospitals = Hospital::orderBy('is_unlocked', '
    asc')->orderBy('name', 'asc')->get();

return $view;
}

/**
* Show the form for creating a new hospital record.
*
* @return \Illuminate\Http\Response
*/
public function create() {
$view = view('hospitals.create');
$view->title = 'Add Hospital or Clinic Record | 
    Philippine Stroke Portal';

return $view;
}

/**
* Store a newly created hospital record in storage.
*
* @param \Illuminate\Http\Request $request
* @return \Illuminate\Http\Response
*/
public function store(Request $request) {
$validator = $this->hospitalsValidator->validate(
    $request->all());

if ($validator->fails()) {
return redirect('hospitals/create')
->with('alert', 'There is a problem with your input.')
->withErrors($validator)
->withInput();
}

$this->hospitalsValidator->store($request->all());

if (Auth::check())
return redirect('hospitals')->with('success', 'Hospital
    record created. Please wait for approval.');
else
return redirect('auth/register')->with('success', '
    Hospital record created. Please wait for approval.
    ');
}

/**
* Display the specified hospital record.
*
* @param int $hospital_id
* @return \Illuminate\Http\Response
*/
public function show($hospital_id) {
$hospital = Hospital::findOrFail($hospital_id);
$view = view('hospitals.show');
$view->title = $hospital->name . ' | Philippine Stroke
    Portal';

$view->hospital = $hospital;
$view->professionals = $this->retrieveProfessionals(
    $hospital_id);

$user = Auth::user();
$view->is_local_admin = $this->professionalMiddleware->
    checkLocalAdmin($user, $hospital_id);
$view->is_system_admin = ($user->systemAdmin) ? true :
    false;
$view->is_staff_in_hospital = $this->
    professionalMiddleware->checkStaffInHospital($user
    , $hospital_id);
$view->is_staff = $this->professionalMiddleware->
    checkProfession($user);

return $view;
}

/**
* Lock hospital or clinic record.
*
* @param int $hospital_id
* @return \Illuminate\Http\Response
*/
public function lockHospital($hospital_id) {
$hospital = Hospital::findOrFail($hospital_id);
$hospital['is_unlocked'] = false;
$hospital->save();
return redirect('hospitals/' . $hospital_id)->with('
    success', 'Hospital locked.');
}

/**
* Unlock hospital or clinic record.
*
* @param int $hospital_id
* @return \Illuminate\Http\Response
```

```
*/
public function unlockHospital($hospital_id) {
$hospital = Hospital::findOrFail($hospital_id);
$hospital['is_unlocked'] = true;
$hospital->save();
return redirect('hospitals/' . $hospital_id)->with('
    success', 'Hospital unlocked.');;
}

/**
* Show the form for editing the specified hospital
    record.
*
* @param int $hospital_id
* @return \Illuminate\Http\Response
*/
public function edit($hospital_id) {
$hospital = Hospital::findOrFail($hospital_id);
$view = view('hospitals.edit');
$view->title = 'Edit Hospital or Clinic Record | '
    Philippine Stroke Portal';
$view->hospital = $hospital;
$view->is_system_admin = (Auth::user()->systemAdmin) ?
    true : false;

return $view;
}

/**
* Update the specified hospital in storage record.
*
* @param int $hospital_id
* @param \Illuminate\Http\Request $request
* @return \Illuminate\Http\Response
*/
public function update($hospital_id, Request $request)
    {
Hospital::findOrFail($hospital_id);
$validator = $this->hospitalsValidator->validate(
    $request->all(), $hospital_id);

if ($validator->fails()) {
return redirect('hospitals/' . $hospital_id . '/edit')
->with('alert', 'There is a problem with your input.')
->withErrors($validator)
->withInput();
}

$this->hospitalsValidator->update($hospital_id,
    $request->all());

return redirect('hospitals/' . $hospital_id);
}

/**
* Retrieve professionals in the hospital record.
*
* @param int $hospital_id
* @return \Illuminate\Http\Response
*/
private function retrieveProfessionals($hospital_id) {
$hospital = Hospital::findOrFail($hospital_id);
$professionals = [];
$medical_professionals = [];
$nutritionists = [];
$physiatrists = [];

// get certain hospital professions in
    HospitalMedicalProfession,
    HospitalNutritionistProfession, and
// HospitalPhysiatristProfession
foreach ($hospital->hospitalMedicalProfessions as
    $medical_profession) {
$medical_professionals[] = $medical_profession->
    medicalProfessional;
}
foreach ($hospital->hospitalNutritionistProfessions as
    $nutritionist_profession) {
$nutritionists[] = $nutritionist_profession->
    nutritionist;
}
foreach ($hospital->hospitalPhysiatristProfessions as
    $physiatrist_profession) {
$physiatrists[] = $physiatrist_profession->physiatrist;
}

// sort medical, nutritionist, and physiatrist
    professions based on last name, first name, and
    middle name
$professionals['medical_professionals'] = collect(
    $medical_professionals)
->sortBy(function($medical_professional) {
return sprintf("%s-%s-%s", $medical_professional->user
    ->last_name, $medical_professional->user->
    first_name,
$medical_professional->user->middle_name);
});

$professionals['nutritionists'] = collect(
    $nutritionists)->sortBy(function($nutritionist) {
return sprintf("%s-%s-%s", $nutritionist->user->
    last_name, $nutritionist->user->first_name,
$nutritionist->user->middle_name);
});

$professionals['physiatrists'] = collect($physiatrists)
    ->sortBy(function($physiatrist) {
return sprintf("%s-%s-%s", $physiatrist->user->
    last_name, $physiatrist->user->first_name,
$physiatrist->user->middle_name);
});

return $professionals;
}

}
```

Manages Memberships Trait

```
<?php namespace App\Http\Controllers\Hospitals;
```

```
use App\Models\Hospital;
use App\Http\Requests;
use App\Models\HospitalMedicalProfession;
use App\Models\HospitalNutritionistProfession;
use App\Models\HospitalPhysiatristProfession;
use App\Models\Patient;
use Illuminate\Support\Facades\Auth;

trait ManagesMemberships
{
/**
* Register current user in specified hospital.
*
* @param int $hospital_id
* @return \Illuminate\Http\Response
*/
public function registerInHospital($hospital_id) {
$user = Auth::user();
Hospital::findOrFail($hospital_id);

if ($user->medicalProfessional) {
$professions = $user->medicalProfessional->
    hospitalMedicalProfessions();
if ($professions) {
if (count($professions->where('hospital_id',
    $hospital_id)->get()) == 0) {
$hospital_profession = new HospitalMedicalProfession();
$hospital_profession['hospital_id'] = $hospital_id;
$hospital_profession['medical_professional_id'] = $user
    ->medicalProfessional->id;
$hospital_profession->save();
}
}

} else if ($user->nutritionist) {
$professions = $user->nutritionist->
    hospitalNutritionistProfessions();
if ($professions) {
if (count($professions->where('hospital_id',
    $hospital_id)->get()) == 0) {
$hospital_profession = new
    HospitalNutritionistProfession();
$hospital_profession['hospital_id'] = $hospital_id;
$hospital_profession['nutritionist_id'] = $user->
    nutritionist->id;
$hospital_profession->save();
}
}

} else if ($user->physiatrist) {
$professions = $user->physiatrist->
    hospitalPhysiatristProfessions();
if ($professions) {
if (count($professions->where('hospital_id',
    $hospital_id)->get()) == 0) {
$hospital_profession = new
    HospitalPhysiatristProfession();
$hospital_profession['hospital_id'] = $hospital_id;
$hospital_profession['physiatrist_id'] = $user->
    physiatrist->id;
$hospital_profession->save();
}
}
}

return redirect('hospitals/' . $hospital_id)
->with('success', 'Registration␣successful.␣Please␣wait
    ␣for␣approval.');
}

/**
* Leave hospital (i.e., remove profession in hospital).
*
* @param int $hospital_id
* @return \Illuminate\Http\Response
*/
public function leaveHospital($hospital_id) {
$user = Auth::user();
Hospital::findOrFail($hospital_id);

if ($user->medicalProfessional) {
$professions = $user->medicalProfessional->
    hospitalMedicalProfessions();
if ($professions) {
$revoked_profession = $professions->where('hospital_id'
    , $hospital_id)->first();
if ($revoked_profession) {
$patients = Patient::where('hospital_id', $hospital_id)
    ->where('medical_professional_id',
$user->medicalProfessional->id)->get();
foreach ($patients as $patient) {
$patient['medical_professional_id'] = null;
$patient->save();
}
}
$revoked_profession->delete();
}

} else if ($user->nutritionist) {
$professions = $user->nutritionist->
    hospitalNutritionistProfessions();
if ($professions) {
$revoked_profession = $professions->where('hospital_id'
    , $hospital_id)->first();
if ($revoked_profession) {
$patients = Patient::where('hospital_id', $hospital_id)
    ->where('nutritionist_id',
$user->nutritionist->id)->get();
foreach ($patients as $patient) {
$patient['nutritionist_id'] = null;
$patient->save();
}
}
$revoked_profession->delete();
}

} else if ($user->physiatrist) {
$professions = $user->physiatrist->
    hospitalPhysiatristProfessions();
```

```
if ($professions) {
$revoked_profession = $professions->where('hospital_id'
    , $hospital_id)->first();
if ($revoked_profession) {
$patients = Patient::where('hospital_id', $hospital_id)
    ->where('physiatrist_id',
$user->physiatrist->id)->get();
foreach ($patients as $patient) {
$patient['physiatrist_id'] = null;
$patient->save();
}
}
$revoked_profession->delete();
}
}

return redirect('hospitals/' . $hospital_id)->with('
    success', 'You are no longer in the hospital.');
}


}
```

Shows Patients In Hospital Trait

```
<?php namespace App\Http\Controllers\Hospitals;

use App\Models\Hospital;
use App\Http\Requests;
use App\Models\Patient;
use Illuminate\Support\Facades\Auth;

trait ShowsPatientsInHospital
{
/**
* Show account of patient.
*
* @param int $hospital_id
* @param int $patient_id
* @return \Illuminate\Http\Response
*/
public function showPatient($hospital_id, $patient_id)
    {
$hospital = Hospital::findOrFail($hospital_id);
$patient = Patient::findOrFail($patient_id);

if ($patient->hospital) {
if ($patient->hospital->id = $hospital_id) {
$view = view('hospitals.show_patient');
$view->title = $patient->user->fullName() . ' | '
    Philippine Stroke Portal';
$view->patient = $patient;
$view->hospital = $hospital;

$user = Auth::user();
$view->is_local_admin = $this->professionalMiddleware->
    checkLocalAdmin($user, $hospital_id);
$view->is_patient_in_hospital = $this->
    professionalMiddleware->checkPatientInHospital(
    $hospital_id,
$patient_id);
$view->is_user_in_hospital = $this->
    professionalMiddleware->checkStaffInHospital($user
    , $hospital_id);
$view->can_maintain = $this->
    maintainPatientAccountMiddleware->
    checkMaintainingPatient($user,
$patient_id);

return $view;
}
}

return redirect('/');
}

/**
* Lock account of patient.
*
* @param int $hospital_id
* @param int $patient_id
* @return \Illuminate\Http\Response
*/
public function lockPatient($hospital_id, $patient_id)
    {
$patient = Patient::findOrFail($patient_id);

if ($patient->hospital) {
if ($patient->hospital->id = $hospital_id and $patient
    ->user) {
$patient->user->is_unlocked = false;
$patient->user->save();
}
}

return redirect('hospitals/' . $hospital_id . '/
    patients/' . $patient_id)->with('success', '
    Patient locked.');
}

/**
* Unlock account of patient.
*
* @param int $hospital_id
* @param int $patient_id
* @return \Illuminate\Http\Response
*/
public function unlockPatient($hospital_id, $patient_id
    ) {
$patient = Patient::findOrFail($patient_id);

if ($patient->hospital) {
if ($patient->hospital->id = $hospital_id and $patient
    ->user) {
$patient->user->is_unlocked = true;
$patient->user->save();
}
}

return redirect('hospitals/' . $hospital_id . '/
    patients/' . $patient_id)
```

```
->with('success', 'Patient_unlocked.');
}

/**
* Send confirmation e-mail to patient.
*
* @param int $hospital_id
* @param int $patient_id
* @return \Illuminate\Http\Response
*/
public function emailPatient($hospital_id, $patient_id)
    {
$patient = Patient::findOrFail($patient_id);

if ($patient->hospital) {
if ($patient->hospital->id = $hospital_id and $patient
    ->user) {
if (!$patient->user->is_email_confirmed) {
$confirmation_code = str_random(30);
$patient->user->confirmation_code = $confirmation_code;
$patient->user->save();

$this->registrar->sendEmailConfirmation($patient->user
    ->username, $patient->user->email,
$confirmation_code);
}
}
}

return redirect('hospitals/' . $hospital_id . '/
    patients/' . $patient_id)
->with('success', 'Confirmation_e-mail_sent.');
}

/**
* Maintain the account of the patient.
*
* @param int $hospital_id
* @param int $patient_id
* @return \Illuminate\Http\Response
*/
public function maintainPatient($hospital_id,
    $patient_id) {
Hospital::findOrFail($hospital_id);
$patient = Patient::findOrFail($patient_id);
$user = Auth::user();

if ($patient) {
if ($user->medicalProfessional and $patient->
    medicalProfessional == null) {
$medical_professional_id = $user->medicalProfessional->
    id;
$patient['medical_professional_id'] =
    $medical_professional_id;
$patient->save();

} else if ($user->nutritionist and $patient->
    nutritionist == null) {
$nutritionist_id = $user->nutritionist->id;
$patient['nutritionist_id'] = $nutritionist_id;
$patient->save();

} else if ($user->physiatrist and $patient->physiatrist
    == null) {
$physiatrist_id = $user->physiatrist->id;
$patient['physiatrist_id'] = $physiatrist_id;
$patient->save();
}
}

return redirect('hospitals/' . $hospital_id . '/
    patients/' . $patient_id)
->with('success', 'Patient_account_maintained.');
}

}
```

Shows Staffs In Hospital Trait

```
<?php namespace App\Http\Controllers\Hospitals;

use App\Models\Hospital;
use App\Http\Requests;

use App\Models\HospitalMedicalProfession;
use App\Models\HospitalNutritionistProfession;
use App\Models\HospitalPhysiatristProfession;
use App\Models\MedicalProfessional;
use App\Models\Nutritionist;
use App\Models\Patient;
use App\Models\Physiatrist;
use Illuminate\Support\Facades\Auth;

trait ShowsStaffInHospital
{
/**
* Show account of hospital or clinic staff member.
*
* @param int $hospital_id
* @param string $profession
* @param int $staff_id
* @return \Illuminate\Http\Response
*/
public function showStaffMember($hospital_id,
    $profession, $staff_id) {
$hospital = Hospital::findOrFail($hospital_id);

if ($profession == 'mp') {
$profession_name = 'Medical_Professional';
$staff = MedicalProfessional::findOrFail($staff_id);
$is_staff_approved = HospitalMedicalProfession::where('
    hospital_id', $hospital_id)
->where('medical_professional_id', $staff_id)->first()
    ->is_approved;

} else if ($profession == 'n') {
$profession_name = 'Nutritionist';
$staff = Nutritionist::findOrFail($staff_id);
$is_staff_approved = HospitalNutritionistProfession::
    where('hospital_id', $hospital_id)
```

```
->where('nutritionist_id', $staff_id)->first()->
    is_approved;

} else if ($profession == 'p') {
$profession_name = 'Physiatrist';
$staff = Physiatrist::findOrFail($staff_id);
$is_staff_approved = HospitalPhysiatristProfession::
    where('hospital_id', $hospital_id)
->where('physiatrist_id', $staff_id)->first()->
    is_approved;

} else {
return redirect('/');
}

$view = view('hospitals.show_staff_member');
$view->title = $staff->user->fullName() . '_|_
    Philippine_Stroke_Portal';
$view->staff = $staff;
$view->is_staff_approved = $is_staff_approved;
$view->hospital = $hospital;
$view->profession = $profession;
$view->profession_name = $profession_name;

$user = Auth::user();
$view->is_local_admin = $this->professionalMiddleware->
    checkLocalAdmin($user, $hospital_id);

return $view;
}

/**
* Lock account of hospital or clinic staff member.
*
* @param int $hospital_id
* @param string $profession
* @param int $staff_id
* @return \Illuminate\Http\Response
*/
public function lockStaffMember($hospital_id,
    $profession, $staff_id) {
if ($profession == 'mp') {
$staff_profession = HospitalMedicalProfession::where('
    hospital_id', $hospital_id)
->where('medical_professional_id', $staff_id)->first();
$patients = Patient::where('hospital_id', $hospital_id)
    ->where('medical_professional_id', $staff_id)
->get();
foreach ($patients as $patient) {
$patient['medical_professional_id'] = null;
$patient->save();
}

} else if ($profession == 'n') {
$staff_profession = HospitalNutritionistProfession::
    where('hospital_id', $hospital_id)
->where('nutritionist_id', $staff_id)->first();
$patients = Patient::where('hospital_id', $hospital_id)
    ->where('nutritionist_id', $staff_id)
->get();
foreach ($patients as $patient) {
$patient['nutritionist_id'] = null;
$patient->save();
}

} else if ($profession == 'p') {
$staff_profession = HospitalPhysiatristProfession::
    where('hospital_id', $hospital_id)
->where('physiatrist_id', $staff_id)->first();
$patients = Patient::where('hospital_id', $hospital_id)
    ->where('physiatrist_id', $staff_id)
->get();
foreach ($patients as $patient) {
$patient['physiatrist_id'] = null;
$patient->save();
}

} else {
return redirect('/');
}

$staff_profession['is_approved'] = false;
$staff_profession->save();

return redirect('hospitals/' . $hospital_id . '/' .
    $profession . '/' . $staff_id)
->with('success', 'Staff_member_deactivated');
}

/**
* Unlock account of hospital or clinic staff member.
*
* @param int $hospital_id
* @param string $profession
* @param int $staff_id
* @return \Illuminate\Http\Response
*/
public function unlockStaffMember($hospital_id,
    $profession, $staff_id) {
if ($profession == 'mp') {
$staff_profession = HospitalMedicalProfession::where('
    hospital_id', $hospital_id)
->where('medical_professional_id', $staff_id)->first();

} else if ($profession == 'n') {
$staff_profession = HospitalNutritionistProfession::
    where('hospital_id', $hospital_id)
->where('nutritionist_id', $staff_id)->first();

} else if ($profession == 'p') {
$staff_profession = HospitalPhysiatristProfession::
    where('hospital_id', $hospital_id)
->where('physiatrist_id', $staff_id)->first();

} else {
return redirect('/');
}

$staff_profession['is_approved'] = true;
$staff_profession->save();
```

```
return redirect('hospitals/' . $hospital_id . '/' .
    $profession . '/' . $staff_id)
->with('success', 'Staff member activated.');
}

/**
* Send confirmation e-mail to staff member.
*
* @param int $hospital_id
* @param string $profession
* @param int $staff_id
* @return \Illuminate\Http\Response
*/
public function emailStaffMember($hospital_id,
    $profession, $staff_id) {
if ($profession == 'mp') {
$staff = HospitalMedicalProfession::where('hospital_id'
    , $hospital_id)
->where('medical_professional_id', $staff_id)->first()
    ->medicalProfessional;

} else if ($profession == 'n') {
$staff = HospitalNutritionistProfession::where('
    hospital_id', $hospital_id)
->where('nutritionist_id', $staff_id)->first()->
    nutritionist;

} else if ($profession == 'p') {
$staff = HospitalPhysiatristProfession::where('
    hospital_id', $hospital_id)
->where('physiatrist_id', $staff_id)->first()->
    physiatrist;

} else {
return redirect('/');
}

// send email confirmation if the e-mail address is not
     confirmed yet
if ($staff->user) {
if (! $staff->user->is_email_confirmed) {
$confirmation_code = str_random(30);
$staff->user->confirmation_code = $confirmation_code;
$staff->user->save();

$this->registrar->sendEmailConfirmation($staff->user->
    username, $staff->user->email,
$confirmation_code);
}
}

return redirect('hospitals/' . $hospital_id . '/' .
    $profession . '/' . $staff_id)
->with('success', 'Confirmation e-mail sent.');
}

}
```

Medical Records Controller

```
<?php namespace App\Http\Controllers\MedicalRecords;

use App\Http\Middleware\MedicalProfessionalMiddleware;
use App\Http\Middleware\NutritionistMiddleware;
use App\Http\Middleware\PhysiatristMiddleware;
use App\Http\Middleware\StaffMiddleware;
use App\Http\Requests;
use App\Http\Controllers\Controller;

use App\Models\DailyRecord;
use App\Services\Retrievers\MedicalRecordRetriever;
use App\Services\Retrievers\SelectFieldListsRetriever;
use App\Services\Validators\MedicalRecordsValidator;

class MedicalRecordsController extends Controller {

/*
|--------------------------------------------------------------------------
| Medical Records Controller
|--------------------------------------------------------------------------
|
| This controller handles the presentation of all the
    attributes of
| the medical record of the patient, as well as
    creating and
| updating these attributes.
|
*/

use ManagesMedicationRecords;

use ManagesNutritionistRecords;

use ManagesPhysiatristRecords;

/**
* Create a new medical records controller instance.
*
* @param \App\Services\Validators\
    MedicalRecordsValidator $medicalRecordsValidator
* @param \App\Services\Retrievers\
    MedicalRecordRetriever $medicalRecordRetriever
* @param \App\Services\Retrievers\
    SelectFieldListsRetriever
    $selectFieldListsRetriever
* @param \App\Http\Middleware\
    MedicalProfessionalMiddleware
    $medicalProfessionalMiddleware
* @param \App\Http\Middleware\NutritionistMiddleware
    $nutritionistMiddleware
* @param \App\Http\Middleware\PhysiatristMiddleware
    $physiatristMiddleware
* @param \App\Http\Middleware\StaffMiddleware
    $staffMiddleware
*/
public function __construct(
MedicalRecordsValidator $medicalRecordsValidator,
    MedicalRecordRetriever $medicalRecordRetriever,
```

```
SelectFieldListsRetriever $selectFieldListsRetriever,
MedicalProfessionalMiddleware
    $medicalProfessionalMiddleware,
    NutritionistMiddleware $nutritionistMiddleware,
PhysiatristMiddleware $physiatristMiddleware,
    StaffMiddleware $staffMiddleware
) {
$this->middleware('auth');
$this->middleware('medicalProfessional', ['only' => ['
    indexMedicationRecord', 'createMedicationRecord',
'editMedicationRecord', 'storeMedicationRecord',]]);
$this->middleware('nutritionist', ['only' => ['
    indexNutritionistRecord', '
    createNutritionistRecord',
'editNutritionistRecord', 'storeNutritionistRecord',]])
    ;
$this->middleware('physiatrist', ['only' => ['
    indexPhysiatristRecord', 'createPhysiatristRecord'
    ,
'editPhysiatristRecord', 'storePhysiatristRecord',]]);
$this->medicalRecordsValidator =
    $medicalRecordsValidator;
$this->medicalRecordsRetriever =
    $medicalRecordRetriever;
$this->selectFieldListsRetriever =
    $selectFieldListsRetriever;
$this->medicalProfessionalMiddleware =
    $medicalProfessionalMiddleware;
$this->nutritionistMiddleware = $nutritionistMiddleware
    ;
$this->physiatristMiddleware = $physiatristMiddleware;
$this->staffMiddleware = $staffMiddleware;
}

}
```

Laboratory Results Controller

```
<?php namespace App\Http\Controllers\MedicalRecords;

use App\Http\Middleware\MedicalProfessionalMiddleware;
use App\Http\Middleware\NutritionistMiddleware;
use App\Http\Middleware\PhysiatristMiddleware;
use App\Http\Middleware\StaffMiddleware;
use App\Http\Requests;
use App\Http\Controllers\Controller;

use App\Models\Patient;
use App\Models\LaboratoryResult;
use App\Services\Retrievers\SelectFieldListsRetriever;
use App\Services\Validators\LaboratoryResultsValidator;
use Carbon\Carbon;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

use Chromabits\Pagination\FoundationPresenter;

class LaboratoryResultsController extends Controller {

/*
|--------------------------------------------------------------------------
| Laboratory Results Controller
|--------------------------------------------------------------------------
|
| This controller handles the presentation of all the
    laboratory
| results of the patient, as well as the forms for
    creating,
| editing, downloading, and deleting them.
|
*/

/**
* Creates a new laboratory results controller instance.
*
* @param \App\Services\Validators\
    LaboratoryResultsValidator
    $laboratoryResultsValidator
* @param \App\Services\Retrievers\
    SelectFieldListsRetriever
    $selectFieldListsRetriever
* @param \App\Http\Middleware\
    MedicalProfessionalMiddleware
    $medicalProfessionalMiddleware
* @param \App\Http\Middleware\NutritionistMiddleware
    $nutritionistMiddleware
* @param \App\Http\Middleware\PhysiatristMiddleware
    $physiatristMiddleware
* @param \App\Http\Middleware\StaffMiddleware
    $staffMiddleware
*/
public function __construct(
LaboratoryResultsValidator $laboratoryResultsValidator,
     SelectFieldListsRetriever
    $selectFieldListsRetriever,
MedicalProfessionalMiddleware
    $medicalProfessionalMiddleware,
    NutritionistMiddleware $nutritionistMiddleware,
PhysiatristMiddleware $physiatristMiddleware,
    StaffMiddleware $staffMiddleware
) {
$this->middleware('auth');
$this->middleware('patient', ['only' => ['index', '
    download', 'create', 'store', 'edit', 'update',]])
    ;
$this->laboratoryResultsValidator =
    $laboratoryResultsValidator;
$this->selectFieldListsRetriever =
    $selectFieldListsRetriever;
$this->medicalProfessionalMiddleware =
    $medicalProfessionalMiddleware;
$this->nutritionistMiddleware = $nutritionistMiddleware
    ;
$this->physiatristMiddleware = $physiatristMiddleware;
$this->staffMiddleware = $staffMiddleware;
}

/**
```

```
* Display a listing of the laboratory results.
*
* @param int $patient_id
* @return \Illuminate\Http\Response
*/
public function index($patient_id) {
$view = view('laboratory_results.index');
$view->title = 'Laboratory Results | Philippine Stroke
    Portal';
$view->patient = Patient::findOrFail($patient_id);
$view->patient_full_name = $view->patient->user->
    fullName2();

$view->laboratory_results = LaboratoryResult::where('
    patient_id', $patient_id)->latest()->paginate(5);
$view->laboratory_results->setPath($patient_id);
$view->page_links = $view->laboratory_results->render(
    new FoundationPresenter($view->laboratory_results)
    );

$view->user = Auth::user();
$view->is_staff = $this->staffMiddleware->checkIfStaff(
    $view->user);
$view->image_extensions = $this->
    selectFieldListsRetriever->getImageExtensions();
$view->video_extensions = $this->
    selectFieldListsRetriever->getVideoExtensions();

return $view;
}

/**
* Download laboratory result file.
*
* @param int $laboratory_result_id
* @return \Illuminate\Http\\Illuminate\Http\Response
*/
public function download($laboratory_result_id) {
$laboratory_result = LaboratoryResult::findOrFail(
    $laboratory_result_id);
$file = $laboratory_result->file;
if ($file) {
$filename = $file->filename;
$path_to_file = $file->path . '/' . $filename;
$headers = [
'Content-Type' => $file->mimeType,
];
return response()->download($path_to_file, $filename,
    $headers);
} else {
return response();
}
}

/**
* Show the form for creating a new laboratory result.
*
* @param int $patient_id
* @return \Illuminate\Http\Response
*/
public function create($patient_id) {
$view = view('laboratory_results.create');
$view->title = 'Add Laboratory Result | Philippine
    Stroke Portal';
$view->patient = Patient::findOrFail($patient_id);
$view->current_date = Carbon::now();

return $view;
}

/**
* Store a newly created laboratory result in storage.
*
* @param int $patient_id
* @param \Illuminate\Http\Request $request
* @return \Illuminate\Http\Response
*/
public function store($patient_id, Request $request) {
Patient::findOrFail($patient_id);
$validator = $this->laboratoryResultsValidator->
    validate($request->all(), $request->hasFile('file'
    ),
$request->file('file'));

if ($validator->fails()) {
return redirect('laboratory-results/'. $patient_id . '/
    create')
->with('alert', 'There is a problem with your input.')
->withErrors($validator)
->withInput();
}

$this->laboratoryResultsValidator->store($patient_id,
    $request->all(), $request->hasFile('file'),
$request->file('file'));

$success_message = "Laboratory result submitted.";

return redirect('laboratory-results/' . $patient_id)->
    with('success', $success_message);
}

/**
* Show the form for editing the specified laboratory
    result.
*
* @param int $laboratory_result_id
* @return \Illuminate\Http\Response
*/
public function edit($laboratory_result_id) {
$view = view('laboratory_results.edit');
$view->title = 'Edit Laboratory Result | Philippine
    Stroke Portal';
$view->laboratory_result = LaboratoryResult::findOrFail
    ($laboratory_result_id);
$view->patient = $view->laboratory_result->patient;
$view->image_extensions = $this->
    selectFieldListsRetriever->getImageExtensions();
$view->video_extensions = $this->
    selectFieldListsRetriever->getVideoExtensions();
```

```
$view->current_date = Carbon::now();

return $view;
}

/**
* Update the specified laboratory result in storage.
*
* @param int $laboratory_result_id
* @param \Illuminate\Http\Request $request
* @return \Illuminate\Http\Response
*/
public function update($laboratory_result_id, Request
    $request) {
$laboratory_result = LaboratoryResult::findOrFail(
    $laboratory_result_id);
$validator = $this->laboratoryResultsValidator->
    validate($request->all(), $request->hasFile('file'
    ),
$request->file('file'));

if ($validator->fails()) {
return redirect('laboratory-results/edit/'.
    $laboratory_result_id)
->with('alert', 'There is a problem with your input.')
->withErrors($validator)
->withInput();
}

$this->laboratoryResultsValidator->update(
    $laboratory_result_id, $request->all(), $request->
    hasFile('file'),
$request->file('file'));

$success_message = "Laboratory result updated.";

return redirect('laboratory-results/' .
    $laboratory_result->patient->id)->with('success',
    $success_message);
}

}
```

Manages Medication Records Trait

```
<?php namespace App\Http\Controllers\MedicalRecords;

use App\Http\Requests;

use App\Models\MedicationRecord;
use App\Models\DailyRecord;
use App\Models\Patient;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

trait ManagesMedicationRecords
{
/**
* Display the listing of medication records.
*
* @param int $patient_id
* @return \Illuminate\Http\Response
*/
public function indexMedicationRecord($patient_id) {
$view = view('medical_records.medication_records.index'
    );
$view->title = 'Update Medical Records | Philippine
    Stroke Portal';
$view->patient = Patient::findOrFail($patient_id);
$view->patient_full_name = $view->patient->user->
    fullName2();
$view->medication_records = $view->patient->
    medicationRecords()->latest()->get();
$view->user = Auth::user();
$view->is_staff = $this->staffMiddleware->checkIfStaff(
    $view->user);

return $view;
}

/**
* Store the new medication record.
*
* @param int $patient_id
* @return \Illuminate\Http\RedirectResponse
*/
public function createMedicationRecord($patient_id) {
$patient = Patient::findOrFail($patient_id);
$medication_record = new MedicationRecord();
$patient->medicationRecords()->save($medication_record)
    ;

return redirect('medical-records/' . $patient_id)->with
    ('success', "Medical record created.");
}

/**
* Show the form for creating and editing stroke and
    medication record attributes.
*
* @param int $patient_id
* @param int $medication_record_id
* @return \Illuminate\Http\Response
*/
public function editMedicationRecord($patient_id,
    $medication_record_id) {
$view = view('medical_records.medication_records.
    edit_medication_record');
$view->title = 'Medical Record | Philippine Stroke
    Portal';
$view->patient = Patient::findOrFail($patient_id);
$view->medication_record = MedicationRecord::findOrFail
    ($medication_record_id);
$view->patient_full_name = $view->patient->user->
    fullName2();

$view->nihss_scale = $this->selectFieldListsRetriever->
    getNIHSS();

$view->attr = $this->medicalRecordsRetriever->
```

```
        getMedicationRecord($medication_record_id);

    return $view;
    }

    /**
    * Store the stroke and medication record attributes of
        the patient.
    *
    * @param int $patient_id
    * @param int $medication_record_id
    * @param \Illuminate\Http\Request $request
    * @return \Illuminate\Http\Response
    */
    public function storeMedicationRecord($patient_id,
        $medication_record_id, Request $request) {
    $validator = $this->medicalRecordsValidator->
        validateMedicationRecord($request->all());

    if ($validator->fails()) {
    return redirect('medical-records/'. $patient_id . '/' .
         $medication_record_id)
    ->with('alert', 'There is a problem with your input.')
    ->withErrors($validator)
    ->withInput();
    }

    $this->medicalRecordsValidator->storeMedicationRecord(
        $medication_record_id, $request->all());

    return redirect('medical-records/'. $patient_id)->with(
        'success', "Medical record updated.");
    }

    }
```

Manages Nutritionist Records Trait

```
<?php namespace App\Http\Controllers\MedicalRecords;

use App\Http\Requests;

use App\Models\DailyRecord;
use App\Models\NutritionistRecord;
use App\Models\Patient;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

trait ManagesNutritionistRecords
{
/**
* Display the listing of nutrition records.
*
* @param int $patient_id
* @return \Illuminate\Http\Response
*/
public function indexNutritionistRecord($patient_id) {
$view = view('medical_records.nutritionist_records.
    index');
$view->title = 'Update Nutrition Records | Philippine
    Stroke Portal';
$view->patient = Patient::findOrFail($patient_id);
$view->patient_full_name = $view->patient->user->
    fullName2();
$view->nutritionist_records = $view->patient->
    nutritionistRecords()->latest()->get();
$view->user = Auth::user();
$view->is_staff = $this->staffMiddleware->checkIfStaff(
    $view->user);

return $view;
}

/**
* Store the new nutrition record.
*
* @param $patient_id
* @return \Illuminate\Http\RedirectResponse
*/
public function createNutritionistRecord($patient_id) {
$patient = Patient::findOrFail($patient_id);
$nutritionist_record = new NutritionistRecord();
$patient->nutritionistRecords()->save(
    $nutritionist_record);

return redirect('nutritionist-records/' . $patient_id)
    ->with('success', "Nutrition record created.");
}

/**
* Show the form for creating and editing nutrition
    record attributes.
*
* @param int $patient_id
* @param int $nutritionist_record_id
* @return \Illuminate\Http\Response
*/
public function editNutritionistRecord($patient_id,
    $nutritionist_record_id) {
$view = view('medical_records.nutritionist_records.
    edit_nutritionist_record');
$view->title = 'Nutrition Record | Philippine Stroke
    Portal';
$view->patient = Patient::findOrFail($patient_id);
$view->nutritionist_record = NutritionistRecord::
    findOrFail($nutritionist_record_id);
$view->patient_full_name = $view->patient->user->
    fullName2();

$view->attr = $this->medicalRecordsRetriever->
    getNutritionistRecord($nutritionist_record_id);

return $view;
}

/**
* Store the nutrition record attributes of the patient.
*
* @param int $patient_id
* @param int $nutritionist_record_id
```

```
* @param \Illuminate\Http\Request $request
* @return \Illuminate\Http\Response
*/
public function storeNutritionistRecord($patient_id,
    $nutritionist_record_id, Request $request) {
$validator = $this->medicalRecordsValidator->
    validateNutritionistRecord($request->all());

if ($validator->fails()) {
return redirect('nutritionist-records/'. $patient_id .
    '/' . $nutritionist_record_id)
->with('alert', 'There is a problem with your input.')
->withErrors($validator)
->withInput();
}

$this->medicalRecordsValidator->storeNutritionistRecord
    ($nutritionist_record_id, $request->all());

return redirect('nutritionist-records/'. $patient_id)->
    with('success', "Nutrition record updated.");
}

}
```

Manages Physiatrist Records Trait

```
<?php namespace App\Http\Controllers\MedicalRecords;

use App\Http\Requests;

use App\Models\DailyRecord;
use App\Models\Patient;
use App\Models\PhysiatristRecord;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

trait ManagesPhysiatristRecords
{
/**
* Display the listing of rehabilitation medicine
    records.
*
* @param int $patient_id
* @return \Illuminate\Http\Response
*/
public function indexPhysiatristRecord($patient_id) {
$view = view('medical_records.physiatrist_records.index
    ');
$view->title = 'Update Rehabilitation Medicine Records
    | Philippine Stroke Portal';
$view->patient = Patient::findOrFail($patient_id);
$view->patient_full_name = $view->patient->user->
    fullName2();
$view->physiatrist_records = $view->patient->
    physiatristRecords()->latest()->get();
$view->user = Auth::user();
$view->is_staff = $this->staffMiddleware->checkIfStaff(
    $view->user);
return $view;
}

/**
* Store the new rehabilitation medicine record.
*
* @param int $patient_id
* @return \Illuminate\Http\RedirectResponse
*/
public function createPhysiatristRecord($patient_id) {
$patient = Patient::findOrFail($patient_id);
$physiatrist_record = new PhysiatristRecord();
$patient->physiatristRecords()->save(
    $physiatrist_record);

return redirect('physiatrist-records/' . $patient_id)
->with('success', "Rehabilitation medicine record
    created.");
}

/**
* Show the form for creating and editing rehabilitation
     medicine record attributes.
*
* @param int $patient_id
* @param int $physiatrist_record_id
* @return \Illuminate\Http\Response
*/
public function editPhysiatristRecord($patient_id,
    $physiatrist_record_id) {
$view = view('medical_records.physiatrist_records.
    edit_physiatrist_record');
$view->title = 'Rehabilitation Medicine Record | 
    Philippine Stroke Portal';
$view->patient = Patient::findOrFail($patient_id);
$view->physiatrist_record = PhysiatristRecord::
    findOrFail($physiatrist_record_id);
$view->patient_full_name = $view->patient->user->
    fullName2();

$view->attr = $this->medicalRecordsRetriever->
    getPhysiatristRecord($physiatrist_record_id);

return $view;
}

/**
* Store the rehabilitation medicine record attributes
    of the patient.
*
* @param int $patient_id
* @param int $physiatrist_record_id
* @param \Illuminate\Http\Request $request
* @return \Illuminate\Http\Response
*/
public function storePhysiatristRecord($patient_id,
    $physiatrist_record_id, Request $request) {
$validator = $this->medicalRecordsValidator->
    validatePhysiatristRecord($request->all());
```

```
if ($validator->fails()) {
return redirect('physiatrist-records/'. $patient_id . '
    /' . $physiatrist_record_id)
->with('alert', 'There is a problem with your input.')
->withErrors($validator)
->withInput();
}

$this->medicalRecordsValidator->storePhysiatristRecord(
    $physiatrist_record_id, $request->all());

return redirect('physiatrist-records/'. $patient_id)
->with('success', "Rehabilitation medicine record
    updated.");;
}

}
```

Patients Controller

```
<?php namespace App\Http\Controllers\Patients;

use App\Http\Middleware\MedicalProfessionalMiddleware;
use App\Http\Middleware\NutritionistMiddleware;
use App\Http\Middleware\PhysiatristMiddleware;
use App\Http\Middleware\StaffMiddleware;
use App\Models\HospitalMedicalProfession;
use App\Models\Patient;
use App\Http\Requests;
use App\Http\Controllers\Controller;

use App\Services\Retrievers\PatientRetriever;
use App\Services\Validators\PatientsValidator;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

class PatientsController extends Controller {

/*
|--------------------------------------------------------------------------
| Patients Controller
|--------------------------------------------------------------------------
|
| This controller handles the presentation of the
    patients of a
| certain professional in all of the hospitals he/she
    works
| in, as well as the registration of new patients,
| updating, and un/locking their accounts.
|
*/

/**
* Create a new patients controller instance.
*
* @param \App\Services\Validators\PatientsValidator
    $patientsValidator
* @param \App\Services\Retrievers\PatientRetriever
    $patientRetriever
* @param \App\Http\Middleware\
    MedicalProfessionalMiddleware
    $medicalProfessionalMiddleware
* @param \App\Http\Middleware\NutritionistMiddleware
    $nutritionistMiddleware
* @param \App\Http\Middleware\PhysiatristMiddleware
    $physiatristMiddleware
* @param \App\Http\Middleware\StaffMiddleware
    $staffMiddleware
*/
public function __construct(
PatientsValidator $patientsValidator, PatientRetriever
    $patientRetriever,
MedicalProfessionalMiddleware
    $medicalProfessionalMiddleware,
    NutritionistMiddleware $nutritionistMiddleware,
PhysiatristMiddleware $physiatristMiddleware,
    StaffMiddleware $staffMiddleware
) {
$this->middleware('auth');
$this->middleware('staff', ['only' => ['index']]);
$this->middleware('medicalProfessional', ['only' => ['
    create', 'store', 'edit', 'update']]);
$this->middleware('patient', ['only' => ['show']]);
$this->patientValidator = $patientsValidator;
$this->patientRetriever = $patientRetriever;
$this->medicalProfessionalMiddleware =
    $medicalProfessionalMiddleware;
$this->nutritionistMiddleware = $nutritionistMiddleware
    ;
$this->physiatristMiddleware = $physiatristMiddleware;
$this->staffMiddleware = $staffMiddleware;
}

/**
* Display a listing of the patients of a certain
    professional.
*
* @return \Illuminate\Http\Response
*/
public function index() {
$view = view('patients.index');
$view->title = 'Patients | Philippine Stroke Portal';
$user = Auth::user();
$view->full_name = $user->fullName();

$medical_professional = $user->medicalProfessional;
$nutritionist = $user->nutritionist;
$physiatrist = $user->physiatrist;

$view->mp_patients = $this->patientRetriever->
    getMedicalProfessionalPatients(
    $medical_professional);
$view->n_patients = $this->patientRetriever->
    getNutritionistPatients($nutritionist);
$view->p_patients = $this->patientRetriever->
    getPhysiatristPatients($physiatrist);

return $view;
```

```
}

/**
* Show the form for creating a new patient record in a
     given hospital and profession.
*
* @param int $profession_id
* @return \Illuminate\Http\Response
*/
public function create($profession_id) {
$view = view('patients.create');
$view->title = 'Register a Patient | Philippine Stroke
     Portal';
$view->full_name = Auth::user()->fullName();

$view->hospital = HospitalMedicalProfession::findOrFail
     ($profession_id)->hospital;
$view->profession_id = $profession_id; // id of the
     profession of the user in a given hospital
$view->professional = Auth::user()->fullName(); // full
      name of the professional
return $view;
}

/**
* Store a newly created patient record in storage.
*
* @param \Illuminate\Http\Request $request
* @param int $profession_id
* @return \Illuminate\Http\Response
*/
public function store(Request $request, $profession_id)
      {
$validator = $this->patientValidator->validate($request
     ->all());

if ($validator->fails()) {
return redirect('patients/create/' . $profession_id)
->with('alert', 'There is a problem with your input.')
->withErrors($validator)
->withInput($request->except(['password', '
     password_confirmation']));
}

$medical_profession = HospitalMedicalProfession::
     findOrFail($profession_id);
$this->patientValidator->create($request->all(),
     $medical_profession);

return redirect('patients')->with('success', 'Patient
     registered.');
}

/**
* Display the specified patient record.
*
* @param  int  $patient_id
* @return \Illuminate\Http\Response
*/
public function show($patient_id) {
$view = view('patients.show');
$view->patient = Patient::findOrFail($patient_id);
$view->patient_full_name = $view->patient->user->
     fullName2();
$view->title = $view->patient_full_name . ' | '
     Philippine Stroke Portal';
$view->user = Auth::user();
$view->full_name = $view->user->fullName();

$view->is_mp_or_patient = $this->
     medicalProfessionalMiddleware->
     checkIfMedicalProfessionalOrPatient(
$view->user,
$patient_id
);
$view->is_n_or_patient = $this->nutritionistMiddleware
     ->checkIfNutritionistOrPatient($view->user,
     $patient_id);
$view->is_p_or_patient = $this->physiatristMiddleware->
     checkIfPhysiatristOrPatient($view->user,
     $patient_id);
$view->is_staff = $this->staffMiddleware->checkIfStaff(
     $view->user);

$view->patient_id = $patient_id;
if ($view->patient->medical_professional)
$view->medical_professional = $view->patient->
     medical_professional->user->fullname();
if ($view->patient->nutritionist)
$view->nutritionist = $view->patient->nutritionist->
     user->fullname();
if ($view->patient->physiatrist)
$view->physiatrist = $view->patient->physiatrist->user
     ->fullname();

return $view;
}

/**
* Show the form for editing the specified patient
     record.
*
* @param  int  $patient_id
* @return \Illuminate\Http\Response
*/
public function edit($patient_id) {
$view = view('patients.edit');
$view->patient = Patient::findOrFail($patient_id);
$view->title = 'Update Patient Account Profile | '
     Philippine Stroke Portal';
$view->full_name = Auth::user()->fullName();

return $view;
}

/**
* Update the specified patient record in storage.
*
* @param  int  $patient_id
* @param \Illuminate\Http\Request $request
```

```
* @return \Illuminate\Http\Response
*/
public function update($patient_id, Request $request) {
$patient = Patient::findOrFail($patient_id);

$validator = $this->patientValidator->validateUpdate(
    $request->all(), $patient_id);

if ($validator->fails()) {
return redirect('patients/' . $patient_id . '/edit')
->with('alert', 'There is a problem with your input.')
->withErrors($validator)
->withInput();
}

$this->patientValidator->update($request->all(),
    $patient);

return redirect('patients/' . $patient_id)->with('
    success', 'Patient account profile updated.');
}
}
```

Patient Attributes Controller

```
<?php namespace App\Http\Controllers\Patients;

use App\Http\Middleware\MedicalProfessionalMiddleware;
use App\Http\Middleware\NutritionistMiddleware;
use App\Http\Middleware\PhysiatristMiddleware;
use App\Http\Middleware\StaffMiddleware;
use App\Http\Requests;
use App\Http\Controllers\Controller;

use App\Models\Patient;
use App\Models\PatientAttribute;
use App\Models\PatientAttributeType;
use App\Services\Retrievers\SelectFieldListsRetriever;
use App\Services\Validators\PatientAttributesValidator;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

class PatientAttributesController extends Controller {

/*
|--------------------------------------------------------------------------
| Patient Attributes Controller
|--------------------------------------------------------------------------
|
| This controller handles the presentation of all the
    attributes of
| the patient, as well as creating and updating these
| attributes.
|
*/

/**
* Creates a new patients controller instance
*
* @param \App\Services\Validators\
    PatientAttributesValidator
    $patientAttributesValidator
* @param \App\Services\Retrievers\
    SelectFieldListsRetriever
    $selectFieldListsRetriever
* @param \App\Http\Middleware\
    MedicalProfessionalMiddleware
    $medicalProfessionalMiddleware
* @param \App\Http\Middleware\NutritionistMiddleware
    $nutritionistMiddleware
* @param \App\Http\Middleware\PhysiatristMiddleware
    $physiatristMiddleware
* @param \App\Http\Middleware\StaffMiddleware
    $staffMiddleware
*/
public function __construct(
PatientAttributesValidator $patientAttributesValidator,
     SelectFieldListsRetriever
    $selectFieldListsRetriever,
MedicalProfessionalMiddleware
    $medicalProfessionalMiddleware,
    NutritionistMiddleware $nutritionistMiddleware,
PhysiatristMiddleware $physiatristMiddleware,
    StaffMiddleware $staffMiddleware
) {
$this->middleware('auth');
$this->middleware('patient', ['only' => ['index', '
    store', ]]);
$this->patientAttributesValidator =
    $patientAttributesValidator;
$this->selectFieldListsRetriever =
    $selectFieldListsRetriever;
$this->medicalProfessionalMiddleware =
    $medicalProfessionalMiddleware;
$this->nutritionistMiddleware = $nutritionistMiddleware
    ;
$this->physiatristMiddleware = $physiatristMiddleware;
$this->staffMiddleware = $staffMiddleware;
}

/**
* Display a listing of the patient attributes, as well
    as the form in creating and editing them.
*
* @param int $patient_id
* @return \Illuminate\Http\Response
*/
public function index($patient_id)      {
$view = view('patient_attributes.index');
$view->title = 'General Patient Details | Philippine
    Stroke Portal';

$view->patient = Patient::findOrFail($patient_id);
$view->patient_full_name = $view->patient->user->
    fullName2();
$view->civil_statuses = $this->
    selectFieldListsRetriever->getCivilStatuses();
$view->highest_educational_attainments = $this->
```

```
        selectFieldListsRetriever->
        getHighestEducationalAttainments();
$view->types_of_stroke = $this->
        selectFieldListsRetriever->getTypesOfStroke();
$view->attr = $this->getPatientAttributes($patient_id);

$view->patient_id = $patient_id;
$view->user = Auth::user();
$view->is_mp_or_patient = $this->
        medicalProfessionalMiddleware->
        checkIfMedicalProfessionalOrPatient(
$view->user, $patient_id
);
$view->is_n_or_patient = $this->nutritionistMiddleware
        ->checkIfNutritionistOrPatient($view->user,
        $patient_id);
$view->is_p_or_patient = $this->physiatristMiddleware->
        checkIfPhysiatristOrPatient($view->user,
        $patient_id);
$view->is_staff = $this->staffMiddleware->checkIfStaff(
        $view->user);

return $view;
}

/**
* Store the created or updated patient attributes in
        storage.
*
* @param int $patient_id
* @param \Illuminate\Http\Request $request
* @return \Illuminate\Http\Response
*/
public function store($patient_id, Request $request) {
$validator = $this->patientAttributesValidator->
        validate($request->all());

if ($validator->fails()) {
return redirect('patient-attributes/'. $patient_id)
->with('alert', 'There is a problem with your input.')
->withErrors($validator)
->withInput($request->except([
'relationship_to_family_member',
'type_of_stroke_of_family_member',
'type_of_surgery',
'year_of_surgery',
]));
}

$this->patientAttributesValidator->store($patient_id,
        $request->all());

return redirect('patient-attributes/' . $patient_id)->
        with('success', 'General details updated.');
}

/**
* Get the attributes of the patient stored in the
        database.
*
* @param int $patient_id
* @return array $attr
*/
private function getPatientAttributes($patient_id) {
$patient_attribute_types = PatientAttributeType::all()
        ->lists('type', 'id');
$patient_attributes = PatientAttribute::where('
        patient_id', $patient_id)->get();
$array_attributes = ['relationship_to_family_member', '
        type_of_stroke_of_family_member', 'type_of_surgery
        ',
'year_of_surgery', 'disease_or_attack'];
$attr = [];

foreach ($patient_attributes as $patient_attribute) {
        // loop through the existing attributes of the
        patient
$type = $patient_attribute->patientAttributeType->type;
if (! isset($attr[$type])) { // if attribute is not set
         yet
if (in_array($type, $array_attributes)) // if attribute
         is supposed to be an array
$attr[$type] = [$patient_attribute->value];
else
$attr[$type] = $patient_attribute->value;
} else { // else, it is either a string or an array
if (is_string($attr[$type])) { // if attribute is a
        string
$temp = $attr[$type];
$attr[$type] = array($temp);
}
$attr[$type][] = $patient_attribute->value;
}
}

// add other attributes (set to null) the patient does
        not have
foreach ($patient_attribute_types as
        $patient_attribute_type) {
if (! isset($attr[$patient_attribute_type]))
$attr[$patient_attribute_type] = null;
}

$attr = $this->getPatientDiseasesAndAttacks($attr);

return $attr;
}

/**
* Get patient's history of diseases and attacks
*
* @param array $attr
* @return array
*/
private function getPatientDiseasesAndAttacks(array
        $attr) {
$diseases_and_attacks = $this->
        selectFieldListsRetriever->getDiseasesAndAttacks()
        ;
$attributes_disease_and_attack = [];
```

```
// Set up the presenting symptom attributes
foreach ($diseases_and_attacks as $disease_and_attack
    => $disease_and_attack_display_text) {
$attributes_disease_and_attack[$disease_and_attack] = [
    $disease_and_attack_display_text];
if ($attr['disease_or_attack'] != null and in_array(
    $disease_and_attack, $attr['disease_or_attack']))
    {
// if the field exists on the attribute list, the
    checkbox is selected
$attributes_disease_and_attack[$disease_and_attack][] =
     '1';
} else {
// else, the checkbox is not selected
$attributes_disease_and_attack[$disease_and_attack][] =
     '0';
}
}
$attr['disease_or_attack'] =
    $attributes_disease_and_attack;
return $attr;
}
}
```

Medical Conditions Controller

```
<?php namespace App\Http\Controllers\Patients;

use App\Http\Middleware\StaffMiddleware;
use App\Http\Requests;
use App\Http\Controllers\Controller;

use App\Models\DecisionOnCondition;
use App\Models\FollowUpVisit;
use App\Models\MedicalCondition;
use App\Models\Notification;
use App\Models\Patient;
use App\Services\Retrievers\SelectFieldListsRetriever;
use Carbon\Carbon;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Validator;

use Chromabits\Pagination\FoundationPresenter;

class MedicalConditionController extends Controller {

/**
* MedicalConditionController constructor.
* @param SelectFieldListsRetriever
    $selectFieldListsRetriever
* @param StaffMiddleware $staffMiddleware
*/
public function __construct(
SelectFieldListsRetriever $selectFieldListsRetriever,
StaffMiddleware $staffMiddleware
) {
$this->middleware('auth');
$this->selectFieldListsRetriever =
    $selectFieldListsRetriever;
$this->staffMiddleware = $staffMiddleware;
}

/**
* Display a listing of the medical conditions.
*
* @param int $patient_id
* @return \Illuminate\Http\Response
*/
public function index($patient_id) {
$patient = Patient::findOrFail($patient_id);
$view = view('medical_conditions.index');
$view->title = 'Daily Medical Conditions | Philippine
    Stroke Portal';
$view->patient = $patient;

$view->medical_conditions = $patient->medicalConditions
    ()->latest()->paginate(5);
$view->medical_conditions->setPath($patient_id);
$view->page_links = $view->medical_conditions->render(
    new FoundationPresenter($view->medical_conditions)
    );
$view->decisions = $this->selectFieldListsRetriever->
    getDecisions();

$view->user = Auth::user();
$view->is_staff = $this->staffMiddleware->checkIfStaff(
    $view->user);

return $view;
}

/**
* Show the form for creating a new medical condition.
*
* @param int $patient_id
* @return \Illuminate\Http\Response
*/
public function input($patient_id) {
$view = view('medical_conditions.input');
$view->title = 'Input Daily Medical Condition | '
    Philippine Stroke Portal';
$view->patient = Patient::findOrFail($patient_id);
$view->current_date = Carbon::now();

return $view;
}

/**
* Store a newly created medical condition in storage.
*
* @param int $patient_id
* @param \Illuminate\Http\Request $request
* @return \Illuminate\Http\Response
*/
public function store($patient_id, Request $request) {
$patient = Patient::findOrFail($patient_id);
$validator = Validator::make($request->all(), [
'daily_medical_problem' => 'required',
'describe_the_symptom' => 'required',
```

```
'set_schedule_for_visit' => 'required',
]);

if ($validator->fails()) {
return redirect('medical-conditions/input')
->with('alert', 'There is a problem with your input.')
->withErrors($validator)
->withInput();
}

$medical_condition = MedicalCondition::create([
'patient_id' => $patient->id,
'daily_medical_problem' => $request->input('
    daily_medical_problem'),
'describe_the_symptom' => $request->input('
    describe_the_symptom'),
'medicine_taken' => $request->input('medicine_taken'),
'effects_noted' => $request->input('effects_noted'),
'any_maneuver_taken' => $request->input('
    any_maneuver_taken'),
]);

if ($patient->medicalProfessional and $patient->
    hospital) {
if ($patient->medicalProfessional->user) {
$description = 'Patient ' . $patient->user->first_name
    . ' ' . $patient->user->last_name . ' from the ' .
$patient->hospital->name . ' submitted a daily medical
    condition on ' .
$medical_condition->created_at->format('F d, Y, g:i A')
    . '.';
Notification::create([
'user_id' => $patient->medicalProfessional->user->id,
'description' => $description,
'link' => 'medical-conditions/' . $patient->id,
]);
}
}

$medical_professional_user_id = ($patient->
    medicalProfessional) ? $patient->
    medicalProfessional->user->id : 0;

if ($request->input('set_schedule_for_visit') == 'yes')
    {
if ($request->input('date_of_visit')) {
if ($medical_professional_user_id > 0) {
$follow_up_visit = FollowUpVisit::create([
'user_id' => $medical_professional_user_id,
'patient_id' => $patient->id,
'medical_condition_id' => $medical_condition->id,
'date_of_visit' => $request->input('date_of_visit')
]);

if ($patient->medicalProfessional and $patient->
    hospital) {
if ($patient->medicalProfessional->user) {
$description = 'Patient ' . $patient->user->first_name
    . ' ' . $patient->user->last_name . ' from the ' .
    $patient->hospital->name . ' scheduled a follow-up
        visit on ' .
    $follow_up_visit->date_of_visit[1] . '.';
Notification::create([
    'user_id' => $patient->medicalProfessional->user->
        id,
    'description' => $description,
    'link' => 'approve-visits',
]);
}
}

} else {
FollowUpVisit::create([
'patient_id' => $patient->id,
'medical_condition_id' => $medical_condition->id,
'date_of_visit' => $request->input('date_of_visit')
]);
}
}
}

$success_message = "Medical condition submitted. Please
    wait for the decision of your doctor.";

return redirect('medical-conditions/' . $patient_id)->
    with('success', $success_message);
}

/**
* Show the form for editing the specified medical
    condition.
*
* @param int $patient_id
* @param int $medical_condition_id
* @return \Illuminate\Http\Response
*/
public function edit($patient_id, $medical_condition_id
    ) {
$view = view('medical_conditions.edit');
$view->title = 'Edit Daily Medical Condition | '.
    Philippine Stroke Portal';
$view->patient = Patient::findOrFail($patient_id);
$view->medical_condition = MedicalCondition::findOrFail
    ($medical_condition_id);
$view->decisions = $this->selectFieldListsRetriever->
    getDecisions();
$view->current_date = Carbon::now();

return $view;
}

/**
* Update the specified medical condition in storage.
*
* @param int $patient_id
* @param int $medical_condition_id
* @param \Illuminate\Http\Request $request
* @return \Illuminate\Http\Response
*/
public function update($patient_id,
```

```
    $medical_condition_id, Request $request) {
$patient = Patient::findOrFail($patient_id);
$medical_condition = MedicalCondition::findOrFail(
    $medical_condition_id);
$validator = Validator::make($request->all(), [
'daily_medical_problem' => 'required',
'describe_the_symptom' => 'required',
]);

if ($validator->fails()) {
return redirect('medical-conditions/' . $patient->id .
    '/edit')

->withErrors($validator)
->withInput();
}

$medical_condition->update([
'patient_id' => $patient->id,
'daily_medical_problem' => $request->input('
    daily_medical_problem'),
'describe_the_symptom' => $request->input('
    describe_the_symptom'),
'medicine_taken' => $request->input('medicine_taken'),
'effects_noted' => $request->input('effects_noted'),
'any_maneuver_taken' => $request->input('
    any_maneuver_taken'),
]);

$success_message = "Medical condition updated.";

return redirect('medical-conditions/' . $patient->id .
    '/' . $medical_condition->id . '/edit')
->with('success', $success_message);
}

/**
* Make decision on set schedule for visit.
*
* @param int $patient_id
* @param int $medical_condition_id
* @param \Illuminate\Http\Request $request
* @return \Illuminate\Http\Response
*/
public function makeDecision($patient_id,
    $medical_condition_id, Request $request) {
$user = Auth::user();
if ($user->medicalProfessional) {
$decision = DecisionOnCondition::create([
'medical_professional_id' => $user->medicalProfessional
    ->id,
'medical_condition_id' => $medical_condition_id,
'decision' => $request->input('decision'),
'additional_notes' => $request->input('additional_notes
    '),
]);

$patient = Patient::findOrFail($patient_id);
if ($decision->medicalProfessional and $patient->user
    and $patient->hospital) {
if ($decision->medicalProfessional->user) {
$notifier = $decision->medicalProfessional->user;
$description = 'Medical Professional ' . $notifier->
    first_name . ' ' . $notifier->last_name . ' from
    the ' .
$patient->hospital->name . ' made a decision on your
    medical condition on ' .
$decision->created_at->format('F d, Y, g:i A') . '.';
Notification::create([
'user_id' => $patient->user->id,
'description' => $description,
'link' => 'medical-conditions/' . $patient_id,
]);
}
}

if (strcmp($request->input('decision'), '
    follow_up_visit') == 0) {
$follow_up_visit = $decision->followUpVisit;
if ($follow_up_visit) {
if ($follow_up_visit->patient) {
if ($follow_up_visit->patient->user) {
$patient_user = $follow_up_visit->patient->user;
Mail::send('emails.approved_visit',
    array(
        'patient_user' => $patient_user,
        'health_professional_user' => $user,
        'date_of_visit' => $follow_up_visit->
            date_of_visit[1],
    ),
    function ($message) use ($patient_user) {
        $message->to($patient_user->email,
            $patient_user->username)
            ->subject('[Philippine Stroke Portal]
                Approved Schedule for Follow-Up Visit'
                );
    }
);
}
}
}
}
}

return redirect('medical-conditions/' . $patient_id)
->with('success', 'The decision on the medical
    condition was sent to the patient.');
}

/**
* Refer patient to another health professional.
*
* @param int $patient_id
* @return \Illuminate\Http\Response
*/
public function refer($patient_id) {
$view = view('refer.index');
$view->title = 'Refer Patient | Philippine Stroke
    Portal';
$view->patient = Patient::findOrFail($patient_id);
```

```
$hospital = $view->patient->hospital;

$view->user = Auth::user();
$view->is_staff = $this->staffMiddleware->checkIfStaff(
    $view->user);

$hospital_medical_professions = $hospital->
    hospitalMedicalProfessions;
$view->medical_professionals = [0 => 'Please choose a
    medical professional.'];
foreach ($hospital_medical_professions as
    $hospital_medical_profession) {
$medical_professional = $hospital_medical_profession->
    medicalProfessional;
if ($view->patient->medicalProfessional) {
if ($medical_professional->id == $view->patient->
    medicalProfessional->id) {
continue;
}
}
if ($medical_professional->specialties) {
$view->medical_professionals[$medical_professional->id]
     = $medical_professional->user->fullName() .
' - ' . $medical_professional->specialties;
} else {
$view->medical_professionals[$medical_professional->id]
     = $medical_professional->user->fullName();
}
}

$hospital_nutritionist_professions = $hospital->
    hospitalNutritionistProfessions;
$view->nutritionists = [0 => 'Please choose a
    nutritionist.'];
foreach ($hospital_nutritionist_professions as
    $hospital_nutritionist_profession) {
$nutritionist = $hospital_nutritionist_profession->
    nutritionist;
if ($view->patient->nutritionist) {
if ($nutritionist->id == $view->patient->nutritionist->
    id) {
continue;
}
}
$view->nutritionists[$nutritionist->id] = $nutritionist
    ->user->fullName();
}

$hospital_physiatrist_professions = $hospital->
    hospitalPhysiatristProfessions;
$view->physiatrists = [0 => 'Please choose a
    rehabilitation medicine doctor.'];
foreach ($hospital_physiatrist_professions as
    $hospital_physiatrist_profession) {
$physiatrist = $hospital_physiatrist_profession->
    physiatrist;
if ($view->patient->physiatrist) {
if ($physiatrist->id == $view->patient->physiatrist->id
    ) {
continue;
}
}
$view->physiatrists[$physiatrist->id] = $physiatrist->
    user->fullName();
}

return $view;
}

/**
* Refer patient to a medical professional.
*
* @param int $patient_id
* @param Request $request
* @return \Illuminate\Http\RedirectResponse
*/
public function referToAMedicalProfessional($patient_id
    , Request $request) {
$patient = Patient::findOrFail($patient_id);
$medical_professional_id = $request->input('
    medical_professional');
if ($medical_professional_id) {
if (strcmp($medical_professional_id, '0') != 0) {
$patient['medical_professional_id'] =
    $medical_professional_id;
$patient->save();
}
}

return redirect('patients')->with('success', 'Patient
    was referred to a medical professional.');
}

/**
* Refer patient to a nutritionist.
*
* @param int $patient_id
* @param Request $request
* @return \Illuminate\Http\RedirectResponse
*/
public function referToANutritionist($patient_id,
    Request $request) {
$patient = Patient::findOrFail($patient_id);
$nutritionist_id = $request->input('nutritionist');
if ($nutritionist_id) {
if (strcmp($nutritionist_id, '0') != 0) {
$patient['nutritionist_id'] = $nutritionist_id;
$patient->save();
}
}

return redirect('referral/' . $patient_id)->with('
    success', 'Patient was referred to a nutritionist.
    ');
}

/**
* Refer patient to a rehabilitation medicine doctor.
*
* @param $patient_id
```

```
* @param Request $request
* @return \Illuminate\Http\RedirectResponse
*/
public function referToAPhysiatrist($patient_id,
    Request $request) {
$patient = Patient::findOrFail($patient_id);
$physiatrist_id = $request->input('physiatrist');
if ($physiatrist_id) {
if (strcmp($physiatrist_id, '0') != 0) {
$patient['physiatrist_id'] = $physiatrist_id;
$patient->save();
}
}

return redirect('referral/' . $patient_id)->with('
    success',
'Patient was referred to a rehabilitation medicine
    doctor.');
}

}
```

Request Visits Controller

```
<?php namespace App\Http\Controllers\Patients;

use App\Http\Requests;
use App\Http\Controllers\Controller;

use App\Models\FollowUpVisit;
use App\Models\Notification;
use App\Services\Retrievers\SelectFieldListsRetriever;
use Carbon\Carbon;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Mail;
use Illuminate\Support\Facades\Validator;

class RequestVisitsController extends Controller {

public function __construct(
SelectFieldListsRetriever $selectFieldListsRetriever
) {
$this->middleware('auth');
$this->selectFieldListsRetriever =
    $selectFieldListsRetriever;
}

/**
* Display a listing of the resource.
*
* @return Response
*/
public function indexPatient() {
$view = view('visit.index_patient');
$view->title = 'Schedule for Follow-Up Visit | '.
    Philippine Stroke Portal';
$user = Auth::user();
if ($user->patient) {
$view->patient = $user->patient;
$view->follow_up_visits = FollowUpVisit::where('
    patient_id', $user->patient->id)
->orderBy('is_approved', 'asc')->latest()->get();
$view->decisions = $this->selectFieldListsRetriever->
    getDecisions();
$view->current_date = Carbon::now();
return $view;
} else {
return redirect('/');
}
}

/**
* Approve visit.
*
* @return \Illuminate\Http\Response
*/
public function setMedicalProfessionalVisit(Request
    $request) {
$patient = Auth::user()->patient;
if ($patient) {
$validator = Validator::make($request->all(), [
'schedule_medical_professional' => 'required',
]);

if ($validator->fails()) {
return redirect('schedule-visits')
->with('alert', 'There is a problem with your input.')
->withErrors($validator)
->withInput();
}
}

$follow_up_visit = FollowUpVisit::create([
'patient_id' => $patient->id,
'user_id' => $patient->medicalProfessional->user->id,
'date_of_visit' => $request->input('
    schedule_medical_professional'),
]);

if ($patient->medicalProfessional and $patient->
    hospital) {
if ($patient->medicalProfessional->user) {
$description = 'Patient ' . $patient->user->first_name
    . ' ' . $patient->user->last_name . ' from the ' .
$patient->hospital->name . ' scheduled a follow-up
    visit on ' .
$follow_up_visit->date_of_visit[1] . '.';
Notification::create([
'user_id' => $patient->medicalProfessional->user->id,
'description' => $description,
'link' => 'approve-visits',
]);
}
}

return redirect('schedule-visits')->with('success', '
    Schedule for follow-up visit is sent.');
}
```

```
/**
* Approve visit.
*
* @return \Illuminate\Http\Response
*/
public function setNutritionistVisit(Request $request)
    {
$patient = Auth::user()->patient;
if ($patient) {
$validator = Validator::make($request->all(), [
'schedule_nutritionist' => 'required',
]);

if ($validator->fails()) {
return redirect('schedule-visits')
->with('alert', 'There is a problem with your input.')
->withErrors($validator)
->withInput();
}
}

$follow_up_visit = FollowUpVisit::create([
'patient_id' => $patient->id,
'user_id' => $patient->nutritionist->user->id,
'date_of_visit' => $request->input('
    schedule_nutritionist'),
]);

if ($patient->nutritionist and $patient->hospital) {
if ($patient->nutritionist->user) {
$description = 'Patient ' . $patient->user->first_name
    . ' ' . $patient->user->last_name . ' from the ' .
$patient->hospital->name . ' scheduled a follow-up
    visit on ' .
$follow_up_visit->date_of_visit[1] . '.';
Notification::create([
'user_id' => $patient->nutritionist->user->id,
'description' => $description,
'link' => 'approve-visits',
]);
}
}

return redirect('schedule-visits')->with('success', '
    Schedule for follow-up visit is sent.');
}

/**
* Approve visit.
*
* @return \Illuminate\Http\Response
*/
public function setPhysiatristVisit(Request $request) {
$patient = Auth::user()->patient;
if ($patient) {
$validator = Validator::make($request->all(), [
'schedule_physiatrist' => 'required',
]);

if ($validator->fails()) {
return redirect('schedule-visits')
->with('alert', 'There is a problem with your input.')
->withErrors($validator)
->withInput();
}
}

$follow_up_visit = FollowUpVisit::create([
'patient_id' => $patient->id,
'user_id' => $patient->physiatrist->user->id,
'date_of_visit' => $request->input('
    schedule_physiatrist'),
]);

if ($patient->physiatrist and $patient->hospital) {
if ($patient->physiatrist->user) {
$description = 'Patient ' . $patient->user->first_name
    . ' ' . $patient->user->last_name . ' from the ' .
$patient->hospital->name . ' scheduled a follow-up
    visit on ' .
$follow_up_visit->date_of_visit[1] . '.';
Notification::create([
'user_id' => $patient->physiatrist->user->id,
'description' => $description,
'link' => 'approve-visits',
]);
}
}

return redirect('schedule-visits')->with('success', '
    Schedule for follow-up visit is sent.');
}

/**
* Display a listing of the resource.
*
* @return Response
*/
public function indexStaff() {
$view = view('visit.index_staff');
$view->title = 'Approve Patient Request Visits | 
    Philippine Stroke Portal';
$user = Auth::user();
$view->follow_up_visits = FollowUpVisit::where('user_id
    ', $user->id)->orderBy('is_approved', 'asc')
->latest()->get();
return $view;
}

/**
* Display a listing of the resource.
*
* @return Response
*/
public function approveVisit(Request $request) {
$follow_up_visit_id = $request->input('
    follow_up_visit_id');
$follow_up_visit = FollowUpVisit::findOrFail(
    $follow_up_visit_id);
$follow_up_visit['is_approved'] = true;
```

```
$follow_up_visit->save();

if ($follow_up_visit->user and $follow_up_visit->
    patient) {
if ($follow_up_visit->patient->user) {
$description = 'Health Professional ' .
    $follow_up_visit->user->first_name . ' ' .
    $follow_up_visit->user->last_name . ' from the ' .
$follow_up_visit->patient->hospital->name . ' approved
    the set schedule for follow-up visit on ' .
$follow_up_visit->date_of_visit[1] . '.';
Notification::create([
'user_id' => $follow_up_visit->patient->user->id,
'description' => $description,
'link' => 'schedule-visits',
]);
}
}

$user = Auth::user();
if ($follow_up_visit->patient) {
if ($follow_up_visit->patient->user) {
$patient_user = $follow_up_visit->patient->user;
Mail::send('emails.approved_visit',
array(
'patient_user' => $patient_user,
'health_professional_user' => $user,
'date_of_visit' => $follow_up_visit->date_of_visit[1],
),
function ($message) use ($patient_user) {
$message->to($patient_user->email, $patient_user->
    username)
->subject('[Philippine Stroke Portal] Approved Schedule
    for Follow-Up Visit');
}
);
}
}

return redirect('approve-visits');
}
}
```

Medication Regimens Controller

```
<?php namespace App\Http\Controllers\Regimens;

use App\Http\Middleware\MedicalProfessionalMiddleware;
use App\Http\Middleware\NutritionistMiddleware;
use App\Http\Middleware\PhysiatristMiddleware;
use App\Http\Middleware\StaffMiddleware;
use App\Http\Requests;
use App\Http\Controllers\Controller;

use App\Models\MedicationRegimen;
use App\Models\Notification;
use App\Models\Patient;
use App\Services\Validators\MedicationRegimensValidator
    ;
use Carbon\Carbon;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

class MedicationRegimensController extends Controller {

/*
|--------------------------------------------------------------------------
| Medication Regimens Controller
|--------------------------------------------------------------------------
|
| This controller handles the presentation of all the
    medication regimens
| for the patient, as well as the forms for creating,
| editing, and deleting them.
|
*/

/**
* Creates a new medication regimens controller instance
    .
*
* @param \App\Services\Validators\
    MedicationRegimensValidator
    $medicationRegimenValidator
* @param \App\Http\Middleware\
    MedicalProfessionalMiddleware
    $medicalProfessionalMiddleware
* @param \App\Http\Middleware\NutritionistMiddleware
    $nutritionistMiddleware
* @param \App\Http\Middleware\PhysiatristMiddleware
    $physiatristMiddleware
* @param \App\Http\Middleware\StaffMiddleware
    $staffMiddleware
*/
public function __construct(
MedicationRegimensValidator $medicationRegimenValidator
    ,
MedicalProfessionalMiddleware
    $medicalProfessionalMiddleware,
    NutritionistMiddleware $nutritionistMiddleware,
PhysiatristMiddleware $physiatristMiddleware,
    StaffMiddleware $staffMiddleware
) {
$this->middleware('auth');
$this->middleware('medicalProfessional', ['only' => ['
    index', 'create', 'store', 'edit', 'update',
'destroy',]]);
$this->middleware('staff', ['only' => ['create', 'store
    ', 'edit', 'update', 'destroy',]]);
$this->medicationRegimenValidator =
    $medicationRegimenValidator;
$this->medicalProfessionalMiddleware =
    $medicalProfessionalMiddleware;
$this->nutritionistMiddleware = $nutritionistMiddleware
    ;
$this->physiatristMiddleware = $physiatristMiddleware;
$this->staffMiddleware = $staffMiddleware;
}
```

```
/**
* Display a listing of the medication regimens.
*
* @param int $patient_id
* @return \Illuminate\Http\Response
*/
public function index($patient_id) {
$view = view('regimens.medication_regimens.index');
$view->title = 'Daily Medication Regimens | Philippine
    Stroke Portal';
$view->patient = Patient::findOrFail($patient_id);
$view->patient_full_name = $view->patient->user->
    fullName2();
$view->regimens = $view->patient->medicationRegimens;

$view->patient_id = $patient_id;
$view->user = Auth::user();
$view->is_staff = $this->staffMiddleware->checkIfStaff(
    $view->user);

return $view;
}

/**
* Show the form for creating a new medication regimen.
*
* @param int $patient_id
* @return \Illuminate\Http\Response
*/
public function create($patient_id) {
$view = view('regimens.medication_regimens.create');
$view->title = 'Add Medication Regimen | Philippine
    Stroke Portal';
$view->patient = Patient::findOrFail($patient_id);

$view->patient_id = $patient_id;

return $view;
}

/**
* Store a newly created medication regimen in storage.
*
* @param int $patient_id
* @param \Illuminate\Http\Request $request
* @return \Illuminate\Http\Response
*/
public function store($patient_id, Request $request) {
$validator = $this->medicationRegimenValidator->
    validate($request->all());

if ($validator->fails()) {
return redirect('medication-regimens/'. $patient_id . '
    /create')
->with('alert', 'There is a problem with your input.')
->withErrors($validator)
->withInput();
}

$this->medicationRegimenValidator->store($patient_id,
    $request->all());

$patient = Patient::findOrFail($patient_id);
if ($patient->medicalProfessional and $patient->
    hospital) {
if ($patient->medicalProfessional->user) {
$notifier = $patient->medicalProfessional->user;
$description = 'Medical Professional ' . $notifier->
    first_name . ' ' . $notifier->last_name . ' from
    the ' .
$patient->hospital->name . ' recommended a new
    medication regimen on ' .
Carbon::now()->format('F d, Y, g:i A') . '.';
Notification::create([
'user_id' => $patient->user->id,
'description' => $description,
'link' => 'medication-regimens/' . $patient->id,
]);
}
}

return redirect('medication-regimens/' . $patient_id)->
    with('success', 'Medication regimen added.');
}

/**
* Show the form for editing the specified medication
    regimen.
*
* @param int $medication_regimen_id
* @return \Illuminate\Http\Response
*/
public function edit($medication_regimen_id) {
$view = view('regimens.medication_regimens.edit');
$view->title = 'Edit Medication Regimen | Philippine
    Stroke Portal';
$view->medication_regimen = MedicationRegimen::
    findOrFail($medication_regimen_id);
$view->patient = $view->medication_regimen->patient;

return $view;
}

/**
* Update the specified medication regimen in storage.
*
* @param int $medication_regimen_id
* @param \Illuminate\Http\Request $request
* @return \Illuminate\Http\Response
*/
public function update($medication_regimen_id, Request
    $request) {
$medication_regimen = MedicationRegimen::findOrFail(
    $medication_regimen_id);
$validator = $this->medicationRegimenValidator->
    validate($request->all());

if ($validator->fails()) {
return redirect('medication-regimens/edit/'.
```

```
        $medication_regimen_id)
    ->with('alert', 'There is a problem with your input.')
    ->withErrors($validator)
    ->withInput();
    }

    $this->medicationRegimenValidator->update(
        $medication_regimen_id, $request->all());

    if ($medication_regimen->patient) {
    if ($medication_regimen->patient->medicalProfessional
        and $medication_regimen->patient->hospital) {
    if ($medication_regimen->patient->medicalProfessional->
        user) {
    $notifier = $medication_regimen->patient->
        medicalProfessional->user;
    $description = 'Medical Professional ' . $notifier->
        first_name . ' ' . $notifier->last_name . ' from
        the ' .
    $medication_regimen->patient->hospital->name . '
        updated a medication regimen on ' .
    Carbon::now()->format('F d, Y, g:i A') . '.';
    Notification::create([
    'user_id' => $medication_regimen->patient->user->id,
    'description' => $description,
    'link' => 'medication-regimens/' . $medication_regimen
        ->patient->id,
    ]);
    }
    }
    }

    return redirect('medication-regimens/' .
        $medication_regimen->patient->id)
    ->with('success', 'Medication regimen updated.');
    }

    /**
    * Remove the specified medication regimen from storage.
    *
    * @param  int  $medication_regimen_id
    * @return \Illuminate\Http\Response
    */
    public function destroy($medication_regimen_id) {
    $medication_regimen = MedicationRegimen::findOrFail(
        $medication_regimen_id);
    $patient_id = $medication_regimen->patient->id;

    if ($medication_regimen->patient) {
    if ($medication_regimen->patient->medicalProfessional
        and $medication_regimen->patient->hospital) {
    if ($medication_regimen->patient->medicalProfessional->
        user) {
    $notifier = $medication_regimen->patient->
        medicalProfessional->user;
    $description = 'Medical Professional ' . $notifier->
        first_name . ' ' . $notifier->last_name . ' from
        the ' .
    $medication_regimen->patient->hospital->name . '
        deleted a medication regimen on ' .
    Carbon::now()->format('F d, Y, g:i A') . '.';
    Notification::create([
    'user_id' => $medication_regimen->patient->user->id,
    'description' => $description,
    'link' => 'medication-regimens/' . $medication_regimen
        ->patient->id,
    ]);
    }
    }
    }

    $medication_regimen->delete();

    return redirect('medication-regimens/' . $patient_id)->
        with('success', 'Medication regimen deleted.');;
    }

    }
```

Nutritionist Regimens Controller

```
<?php namespace App\Http\Controllers\Regimens;

use App\Http\Middleware\MedicalProfessionalMiddleware;
use App\Http\Middleware\NutritionistMiddleware;
use App\Http\Middleware\PhysiatristMiddleware;
use App\Http\Middleware\StaffMiddleware;
use App\Http\Requests;
use App\Http\Controllers\Controller;

use App\Models\Notification;
use App\Models\NutritionistRegimen;
use App\Models\Patient;
use App\Services\Validators\
    NutritionistRegimensValidator;
use Carbon\Carbon;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

class NutritionistRegimensController extends Controller
    {

/*
|--------------------------------------------------------------------------

| Nutritionist Regimens Controller
|--------------------------------------------------------------------------

|
| This controller handles the presentation of all the
    nutritionist regimens
| for the patient, as well as the forms for creating,
| editing, and deleting them.
|
*/

/**
* Creates a new medication regimens controller instance
    .
*
```

```
* @param \App\Services\Validators\
      NutritionistRegimensValidator
      $nutritionistRegimensValidator
* @param \App\Http\Middleware\
      MedicalProfessionalMiddleware
      $medicalProfessionalMiddleware
* @param \App\Http\Middleware\NutritionistMiddleware
      $nutritionistMiddleware
* @param \App\Http\Middleware\PhysiatristMiddleware
      $physiatristMiddleware
* @param \App\Http\Middleware\StaffMiddleware
      $staffMiddleware
*/
public function __construct(
NutritionistRegimensValidator
      $nutritionistRegimensValidator,
MedicalProfessionalMiddleware
      $medicalProfessionalMiddleware,
      NutritionistMiddleware $nutritionistMiddleware,
PhysiatristMiddleware $physiatristMiddleware,
      StaffMiddleware $staffMiddleware
) {
$this->middleware('auth');
$this->middleware('nutritionist', ['only' => ['index',
      'create', 'store', 'edit', 'update', 'destroy',]])
      ;
$this->middleware('staff', ['only' => ['create', 'store
      ', 'edit', 'update', 'destroy',]]);
$this->nutritionistRegimensValidator =
      $nutritionistRegimensValidator;
$this->medicalProfessionalMiddleware =
      $medicalProfessionalMiddleware;
$this->nutritionistMiddleware = $nutritionistMiddleware
      ;
$this->physiatristMiddleware = $physiatristMiddleware;
$this->staffMiddleware = $staffMiddleware;
}

/**
* Display a listing of the nutritionist regimens.
*
* @param int $patient_id
* @return \Illuminate\Http\Response
*/
public function index($patient_id) {
$view = view('regimens.nutritionist_regimens.index');
$view->title = 'Daily Nutrition Regimens | Philippine
      Stroke Portal';
$view->patient = Patient::findOrFail($patient_id);
$view->patient_full_name = $view->patient->user->
      fullName2();
$view->regimens = $view->patient->nutritionistRegimens;

$view->patient_id = $patient_id;
$view->user = Auth::user();
$view->is_staff = $this->staffMiddleware->checkIfStaff(
      $view->user);

return $view;
}

/**
* Show the form for creating a new nutritionist regimen
      .
*
* @param int $patient_id
* @return \Illuminate\Http\Response
*/
public function create($patient_id) {
$view = view('regimens.nutritionist_regimens.create');
$view->title = 'Add Nutrition Regimen | Philippine
      Stroke Portal';
$view->patient = Patient::findOrFail($patient_id);

$view->patient_id = $patient_id;

return $view;
}

/**
* Store a newly created nutritionist regimen in storage
      .
*
* @param int $patient_id
* @param \Illuminate\Http\Request $request
* @return \Illuminate\Http\Response
*/
public function store($patient_id, Request $request) {
$validator = $this->nutritionistRegimensValidator->
      validate($request->all());

if ($validator->fails()) {
return redirect('nutritionist-regimens/'. $patient_id .
      '/create')
->with('alert', 'There is a problem with your input.')
->withErrors($validator)
->withInput();
}

$this->nutritionistRegimensValidator->store($patient_id
      , $request->all());

$patient = Patient::findOrFail($patient_id);
if ($patient->nutritionist and $patient->hospital) {
if ($patient->nutritionist->user) {
$notifier = $patient->nutritionist->user;
$description = 'Nutritionist ' . $notifier->first_name
      . ' ' . $notifier->last_name . ' from the ' .
$patient->hospital->name . ' recommended a new
      nutrition regimen on ' .
Carbon::now()->format('F d, Y, g:i A') . '.';
Notification::create([
'user_id' => $patient->user->id,
'description' => $description,
'link' => 'nutritionist-regimens/' . $patient->id,
]);
}
}

return redirect('nutritionist-regimens/' . $patient_id)
```

```
->with('success', 'Nutrition regimen created.');
}

/**
* Show the form for editing the specified nutritionist
    regimen.
*
* @param int $nutritionist_regimen_id
* @return \Illuminate\Http\Response
*/
public function edit($nutritionist_regimen_id) {
$view = view('regimens.nutritionist_regimens.edit');
$view->title = 'Edit Nutrition Regimen | Philippine
    Stroke Portal';
$view->nutritionist_regimen = NutritionistRegimen::
    findOrFail($nutritionist_regimen_id);
$view->patient = $view->nutritionist_regimen->patient;

return $view;
}

/**
* Update the specified nutritionist regimen in storage.
*
* @param int $nutritionist_regimen_id
* @param \Illuminate\Http\Request $request
* @return \Illuminate\Http\Response
*/
public function update($nutritionist_regimen_id,
    Request $request) {
$nutritionist_regimen = NutritionistRegimen::findOrFail
    ($nutritionist_regimen_id);
$validator = $this->nutritionistRegimensValidator->
    validate($request->all());

if ($validator->fails()) {
return redirect('nutritionist-regimens/edit/'.
    $nutritionist_regimen_id)
->with('alert', 'There is a problem with your input.')
->withErrors($validator)
->withInput();
}

$this->nutritionistRegimensValidator->update(
    $nutritionist_regimen_id, $request->all());

if ($nutritionist_regimen->patient) {
if ($nutritionist_regimen->patient->nutritionist and
    $nutritionist_regimen->patient->hospital) {
if ($nutritionist_regimen->patient->nutritionist->user)
     {
$notifier = $nutritionist_regimen->patient->
    nutritionist->user;
$description = 'Nutritionist ' . $notifier->first_name
    . ' ' . $notifier->last_name . ' from the ' .
$nutritionist_regimen->patient->hospital->name . '
    updated a nutrition regimen on ' .
Carbon::now()->format('F d, Y, g:i A') . '.';
Notification::create([
'user_id' => $nutritionist_regimen->patient->user->id,
'description' => $description,
'link' => 'nutritionist-regimens/' .
    $nutritionist_regimen->patient->id,
]);
}
}
}

return redirect('nutritionist-regimens/' .
    $nutritionist_regimen->patient->id)
->with('success', 'Nutrition regimen updated.');
}

/**
* Remove the specified nutritionist regimen from
    storage.
*
* @param int $nutritionist_regimen_id
* @return \Illuminate\Http\Response
*/
public function destroy($nutritionist_regimen_id) {
$nutritionist_regimen = NutritionistRegimen::findOrFail
    ($nutritionist_regimen_id);
$patient_id = $nutritionist_regimen->patient->id;

if ($nutritionist_regimen->patient) {
if ($nutritionist_regimen->patient->nutritionist and
    $nutritionist_regimen->patient->hospital) {
if ($nutritionist_regimen->patient->nutritionist->user)
     {
$notifier = $nutritionist_regimen->patient->
    nutritionist->user;
$description = 'Nutritionist ' . $notifier->first_name
    . ' ' . $notifier->last_name . ' from the ' .
$nutritionist_regimen->patient->hospital->name . '
    deleted a nutrition regimen on ' .
Carbon::now()->format('F d, Y, g:i A') . '.';
Notification::create([
'user_id' => $nutritionist_regimen->patient->user->id,
'description' => $description,
'link' => 'nutritionist-regimens/' .
    $nutritionist_regimen->patient->id,
]);
}
}
}

$nutritionist_regimen->delete();

return redirect('nutritionist-regimens/' . $patient_id)
    ->with('success', 'Nutrition regimen deleted.');
}


}
```

Physiatrist Regimens Controller

```
<?php namespace App\Http\Controllers\Regimens;

use App\Http\Middleware\MedicalProfessionalMiddleware;
```

```
use App\Http\Middleware\NutritionistMiddleware;
use App\Http\Middleware\PhysiatristMiddleware;
use App\Http\Middleware\StaffMiddleware;
use App\Http\Requests;
use App\Http\Controllers\Controller;

use App\Models\Notification;
use App\Models\Patient;
use App\Models\PhysiatristRegimen;
use App\Services\Validators\
    PhysiatristRegimensValidator;
use Carbon\Carbon;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

class PhysiatristRegimensController extends Controller
    {

/*
|--------------------------------------------------------------------------

| Physiatrist Regimens Controller
|--------------------------------------------------------------------------

|
| This controller handles the presentation of all the
    physiatrist regimens
| for the patient, as well as the forms for creating,
| editing, and deleting them.
|
*/

/**
* Creates a new medication regimens controller instance
    .
*
* @param \App\Services\Validators\
    PhysiatristRegimensValidator
    $physiatristRegimensValidator
* @param \App\Http\Middleware\
    MedicalProfessionalMiddleware
    $medicalProfessionalMiddleware
* @param \App\Http\Middleware\NutritionistMiddleware
    $nutritionistMiddleware
* @param \App\Http\Middleware\PhysiatristMiddleware
    $physiatristMiddleware
* @param \App\Http\Middleware\StaffMiddleware
    $staffMiddleware
*/
public function __construct(
PhysiatristRegimensValidator
    $physiatristRegimensValidator,
MedicalProfessionalMiddleware
    $medicalProfessionalMiddleware,
    NutritionistMiddleware $nutritionistMiddleware,
PhysiatristMiddleware $physiatristMiddleware,
    StaffMiddleware $staffMiddleware
) {
$this->middleware('auth');
$this->middleware('physiatrist', ['only' => ['index', '
    create', 'store', 'edit', 'update', 'destroy',]]);
$this->middleware('staff', ['only' => ['create', 'store
    ', 'edit', 'update', 'destroy',]]);
$this->physiatristRegimensValidator =
    $physiatristRegimensValidator;
$this->medicalProfessionalMiddleware =
    $medicalProfessionalMiddleware;
$this->nutritionistMiddleware = $nutritionistMiddleware
    ;
$this->physiatristMiddleware = $physiatristMiddleware;
$this->staffMiddleware = $staffMiddleware;
}

/**
* Display a listing of the physiatrist regimens.
*
* @param int $patient_id
* @return \Illuminate\Http\Response
*/
public function index($patient_id) {
$view = view('regimens.physiatrist_regimens.index');
$view->title = 'Daily Rehabilitation Medicine Regimens
    | Philippine Stroke Portal';
$view->patient = Patient::findOrFail($patient_id);
$view->patient_full_name = $view->patient->user->
    fullName2();
$view->regimens = $view->patient->physiatristRegimens;

$view->patient_id = $patient_id;
$view->user = Auth::user();
$view->is_staff = $this->staffMiddleware->checkIfStaff(
    $view->user);

return $view;
}

/**
* Show the form for creating a new physiatrist regimen.
*
* @param int $patient_id
* @return \Illuminate\Http\Response
*/
public function create($patient_id) {
$view = view('regimens.physiatrist_regimens.create');
$view->title = 'Add Rehabilitation Medicine Regimen | -
    Philippine Stroke Portal';
$view->patient = Patient::findOrFail($patient_id);

$view->patient_id = $patient_id;

return $view;
}

/**
* Store a newly created physiatrist regimen in storage.
*
* @param int $patient_id
* @param \Illuminate\Http\Request $request
* @return \Illuminate\Http\Response
*/
```

```
public function store($patient_id, Request $request) {
$validator = $this->physiatristRegimensValidator->
    validate($request->all());

if ($validator->fails()) {
return redirect('physiatrist-regimens/'. $patient_id .
    '/create')
->with('alert', 'There is a problem with your input.')
->withErrors($validator)
->withInput();
}

$this->physiatristRegimensValidator->store($patient_id,
     $request->all());

$patient = Patient::findOrFail($patient_id);
if ($patient->physiatrist and $patient->hospital) {
if ($patient->physiatrist->user) {
$notifier = $patient->physiatrist->user;
$description = 'Rehabilitation Medicine Doctor ' .
    $notifier->first_name . ' ' . $notifier->last_name
     . ' from the ' .
$patient->hospital->name . ' recommended a new
    rehabilitation medicine regimen on ' .
Carbon::now()->format('F d, Y, g:i A') . '.';
Notification::create([
'user_id' => $patient->user->id,
'description' => $description,
'link' => 'physiatrist-regimens/' . $patient->id,
]);
}
}

return redirect('physiatrist-regimens/' . $patient_id)
->with('sucess', 'Rehabilitation medicine regimen
    created.');
}

/**
* Show the form for editing the specified physiatrist
    regimen.
*
* @param int $physiatrist_regimen_id
* @return \Illuminate\Http\Response
*/
public function edit($physiatrist_regimen_id) {
$view = view('regimens.physiatrist_regimens.edit');
$view->title = 'Edit Rehabilitation Medicine Regimen |
    Philippine Stroke Portal';
$view->physiatrist_regimen = PhysiatristRegimen::
    findOrFail($physiatrist_regimen_id);
$view->patient = $view->physiatrist_regimen->patient;

return $view;
}

/**
* Update the specified physiatrist regimen in storage.
*
* @param int $physiatrist_regimen_id
* @param \Illuminate\Http\Request $request
* @return \Illuminate\Http\Response
*/
public function update($physiatrist_regimen_id, Request
     $request) {
$physiatrist_regimen = PhysiatristRegimen::findOrFail(
    $physiatrist_regimen_id);
$validator = $this->physiatristRegimensValidator->
    validate($request->all());

if ($validator->fails()) {
return redirect('physiatrist-regimens/edit/'.
    $physiatrist_regimen_id)
->with('alert', 'There is a problem with your input.')
->withErrors($validator)
->withInput();
}

$this->physiatristRegimensValidator->update(
    $physiatrist_regimen_id, $request->all());

if ($physiatrist_regimen->patient) {
if ($physiatrist_regimen->patient->physiatrist and
    $physiatrist_regimen->patient->hospital) {
if ($physiatrist_regimen->patient->physiatrist->user) {
$notifier = $physiatrist_regimen->patient->physiatrist
    ->user;
$description = 'Rehabilitation Medicine Doctor ' .
    $notifier->first_name . ' ' . $notifier->last_name
     . ' from the ' .
$physiatrist_regimen->patient->hospital->name . '
    updated a rehabilitation medicine regimen on ' .
Carbon::now()->format('F d, Y, g:i A') . '.';
Notification::create([
'user_id' => $physiatrist_regimen->patient->user->id,
'description' => $description,
'link' => 'physiatrist-regimens/' .
    $physiatrist_regimen->patient->id,
]);
}
}
}

return redirect('physiatrist-regimens/' .
    $physiatrist_regimen->patient->id)
->with('sucess', 'Rehabilitation medicine regimen
    updated.');;
}

/**
* Remove the specified physiatrist regimen from storage
    .
*
* @param int $physiatrist_regimen_id
* @return \Illuminate\Http\Response
*/
public function destroy($physiatrist_regimen_id) {
$physiatrist_regimen = PhysiatristRegimen::findOrFail(
    $physiatrist_regimen_id);
$patient_id = $physiatrist_regimen->patient->id;
```

```
if ($physiatrist_regimen->patient) {
if ($physiatrist_regimen->patient->physiatrist and
    $physiatrist_regimen->patient->hospital) {
if ($physiatrist_regimen->patient->physiatrist->user) {
$notifier = $physiatrist_regimen->patient->physiatrist
    ->user;
$description = 'Rehabilitation Medicine Doctor ' .
    $notifier->first_name . ' ' . $notifier->last_name
     . ' from the ' .
$physiatrist_regimen->patient->hospital->name . '
    deleted a rehabilitation medicine regimen on ' .
Carbon::now()->format('F d, Y, g:i A') . '.';
Notification::create([
'user_id' => $physiatrist_regimen->patient->user->id,
'description' => $description,
'link' => 'physiatrist-regimens/' .
    $physiatrist_regimen->patient->id,
]);
}
}
}

$physiatrist_regimen->delete();

return redirect('physiatrist-regimens/' . $patient_id)
->with('sucess', 'Rehabilitation medicine regimen
    deleted.');;
}

}
```

Account Settings Controller

```
<?php namespace App\Http\Controllers\Users;

use App\Http\Middleware\StaffMiddleware;
use App\Http\Requests;
use App\Http\Controllers\Controller;

use App\Services\Validators\AccountSettingsValidator;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\File;

class AccountSettingsController extends Controller {

/*
|--------------------------------------------------------------------------
| Account Settings Controller
|--------------------------------------------------------------------------
|
| This controller handles the account settings of users
    . The general
| account settings are managed here. Changing password
    is also
| handled here.
|
*/

/**
* Create a new accounts controller instance.
*
* @param \App\Services\Validators\
    AccountSettingsValidator $accountValidator
* @param \App\Http\Middleware\StaffMiddleware
    $staffMiddleware
*/
public function __construct(
AccountSettingsValidator $accountValidator,
    StaffMiddleware $staffMiddleware
) {
$this->middleware('auth');
$this->accountValidator = $accountValidator;
$this->staffMiddleware = $staffMiddleware;
}

/**
* Display the account settings.
*
* @return \Illuminate\Http\Response
*/
public function displayAccountSettings(){
$view = view('auth.accounts.general.display');
$user = Auth::user();
$view->full_name = $user->fullName();
$view->title = $view->full_name . " | Philippine Stroke
    Portal";
$view->user = $user;

$patient = $user->patient;
if ($patient){
$medical_professional = $patient->medicalProfessional;
if ($medical_professional) {
$view->medical_professional = $medical_professional->
    user->fullName();
$view->medical_professional_username =
    $medical_professional->user->username;
}

$nutritionist = $patient->nutritionist;
if ($nutritionist) {
$view->nutritionist = $nutritionist->user->fullName();
$view->nutritionist_username = $nutritionist->user->
    username;
}
$physiatrist = $patient->physiatrist;
if ($physiatrist) {
$view->physiatrist = $physiatrist->user->fullName();
$view->physiatrist_username = $physiatrist->user->
    username;
}
}

$view->is_staff = $this->staffMiddleware->checkIfStaff(
    $user);
if ($view->is_staff) {
```

```
if ($user->medicalProfessional) {
$view->clinic_schedule = $user->medicalProfessional->
    clinic_schedule;
} else if ($user->nutritionist) {
$view->clinic_schedule = $user->nutritionist->
    clinic_schedule;
} else {
$view->clinic_schedule = $user->physiatrist->
    clinic_schedule;
}
}

return $view;
}

/**
* Present the form for updating general account
    settings.
*
* @return \Illuminate\Http\Response
*/
public function editGeneralSettings(){
$user = Auth::user();
$view = view('auth.accounts.general.edit');
$view->title = 'General Account Settings | Philippine
    Stroke Portal';
$view->full_name = $user->fullName();
$view->user = $user;

$view->is_staff = $this->staffMiddleware->checkIfStaff(
    $user);
if ($view->is_staff) {
if ($user->medicalProfessional) {
$view->clinic_schedule = $user->medicalProfessional->
    clinic_schedule;
} else if ($user->nutritionist) {
$view->clinic_schedule = $user->nutritionist->
    clinic_schedule;
} else {
$view->clinic_schedule = $user->physiatrist->
    clinic_schedule;
}
}

return $view;
}

/**
* Update the general account settings of user.
*
* @param \Illuminate\Http\Request $request
* @return \Illuminate\Http\Response
*/
public function updateGeneralSettings(Request $request)
    {
$user = Auth::user();

$validator = $this->accountValidator->
    validateGeneralSettings($request->all(), $user);

if ($validator->fails()) {
return redirect('settings/general')
->with('alert', 'There is a problem with your input.')
->withErrors($validator)
->withInput($request->except(['password']));
}

$this->accountValidator->updateGeneralSettings($request
    ->all(), $user);

return redirect('profile')->with('success', 'General
    account settings updated.');
}

/**
* Present the form for uploading a profile picture.
*
* @return \Illuminate\Http\Response
*/
public function uploadProfilePicture() {
$view = view('auth.accounts.profile_picture.upload');
$view->title = 'Profile Picture | Philippine Stroke
    Portal';
$view->user = Auth::user();

return $view;
}

/**
* Update the profile picture of the user.
*
* @param \Illuminate\Http\Request $request
* @return \Illuminate\Http\Response
*/
public function updateProfilePicture(Request $request)
    {
$user = Auth::user();

$validator = $this->accountValidator->
    validateProfilePicture($request->all());

if ($validator->fails()) {
return redirect('settings/profile-picture')
->with('alert', 'There is a problem with your input.')
->withErrors($validator);
}

$this->accountValidator->updateProfilePicture($user,
    $request->hasFile('profile_picture'),
$request->file('profile_picture'));

return redirect('profile')->with('success', 'Profile
    picture updated.');
}

/**
* Delete the profile picture of the user.
*
* @return \Illuminate\Http\Response
*/
```

```
public function deleteProfilePicture() {
$user = Auth::user();
$file = $user->file;

if ($file) {
File::delete($file->path . '/' . $file->filename);
$file->delete();
}

return redirect('profile')->with('success', 'Profile
    picture deleted.');
}

/**
* Present the form for changing password.
*
* @return \Illuminate\Http\Response
*/
public function changePassword() {
$user = Auth::user();
$view = view('auth.accounts.passwords.change');
$view->title = 'Change Password | Philippine Stroke
    Portal';
$view->full_name = $user->fullName();

return $view;
}

/**
* Update the password of the user.
*
* @param \Illuminate\Http\Request $request
* @return \Illuminate\Http\Response
*/
public function updatePassword(Request $request) {
$user = Auth::user();

$validator = $this->accountValidator->validatePassword(
    $request->all(), $user);

if ($validator->fails()) {
return redirect('settings/password')
->with('alert', 'There is a problem with your input.')
->withErrors($validator);
}

$user->password = bcrypt($request->input('new_password'
    ));
$user->save();

return redirect('profile')->with('success', 'Password
    changed.');
}

}
```

System Admin Controller

```
<?php namespace App\Http\Controllers\Users;

use App\Http\Requests;
use App\Http\Controllers\Controller;

use App\Models\ForumCategory;
use App\Models\Hospital;
use App\Models\LocalAdmin;
use App\Models\Supervisor;
use App\User;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Validator;

class SystemAdminController extends Controller {

/**
* SystemAdminController constructor.
*
*/
public function __construct() {
$this->middleware('auth');
$this->middleware('systemAdmin');
}

/**
* Manage local administrators.
*
* @return \Illuminate\Http\Response
*/
public function manageLocalAdmins() {
$view = view('system_admin.manage_local_admins');
$view->title = 'Manage Hospital Administrators |
    Philippine Stroke Portal';
$view->hospitals = Hospital::orderBy('name', 'asc')->
    get();

$hospitals = Hospital::orderBy('name', 'asc')->get();
$view->hospital_list = [0 => 'Please choose a hospital.
    '];
foreach ($hospitals as $hospital) {
$view->hospital_list[$hospital->id] = $hospital->name;
}

$users = User::orderBy('username', 'asc')->get();
$view->user_list = [0 => 'Please choose a user.'];
foreach ($users as $user) {
if (! $user->patient)
$view->user_list[$user->id] = $user->username . ' - ' .
     $user->fullName();
}

return $view;
}

/**
* Save local administrator.
*
* @param \Illuminate\Http\Request $request
* @return \Illuminate\Http\Response
*/
public function saveLocalAdmin(Request $request) {
```

```
$validator = Validator::make($request->all(), [
'hospital_id' => 'required',
'user_id' => 'required',
]);

if ($validator->fails()) {
return redirect('hospital-admins')
->with('alert', 'There is a problem with your input.')
->withErrors($validator)
->withInput();
}

if (strcmp($request->input('hospital_id'), '0') != 0
    and strcmp($request->input('user_id'), '0') != 0)
    {
$hospital = Hospital::findOrFail($request->input('
    hospital_id'));
$user = User::findOrFail($request->input('user_id'));

if ($hospital->localAdmin) {
$old_local_admin = $hospital->localAdmin;
$old_local_admin->delete();
}

if (! $user->localAdmin) {
$new_local_admin = new LocalAdmin();
$new_local_admin['hospital_id'] = $hospital->id;
$new_local_admin['user_id'] = $user->id;
$new_local_admin->save();
} else {
$user->localAdmin['hospital_id'] = $hospital->id;
$user->localAdmin->save();
}

return redirect('hospital-admins')->with('success', '
    Hospital administrator saved.');
} else {
return redirect('hospital-admins');
}
}

/**
* Manage supervisors.
*
* @return \Illuminate\Http\Response
*/
public function manageSupervisors() {
$view = view('system_admin.manage_supervisors');
$view->title = 'Manage Supervisors | Philippine Stroke
    Portal';
$view->user = Auth::user();
$view->forum_categories = ForumCategory::all();

$users = User::orderBy('username', 'asc')->get();
$view->user_list = [0 => 'Please choose a neurologist.'
    ];
foreach ($users as $user) {
if (! $user->patient and $user->medicalProfessional) {
if ($user->medicalProfessional->specialties) {
if (strcmp($user->medicalProfessional->specialties, '
    Neurology') == 0)
$view->user_list[$user->id] = $user->username . ' - ' .
     $user->fullName();
}
}
}

$view->user_list2 = [0 => 'Please choose a nutritionist
    .'];
foreach ($users as $user) {
if (! $user->patient and $user->nutritionist) {
$view->user_list2[$user->id] = $user->username . ' - '
    . $user->fullName();
}
}

$view->user_list3 = [0 => 'Please choose a
    rehabilitation medicine doctor.'];
foreach ($users as $user) {
if (! $user->patient and $user->physiatrist) {
$view->user_list3[$user->id] = $user->username . ' - '
    . $user->fullName();
}
}

return $view;
}

/**
* Save neurology supervisor.
*
* @param \Illuminate\Http\Request $request
* @return \Illuminate\Http\Response
*/
public function saveNeurologySupervisor(Request
    $request) {
$validator = Validator::make($request->all(), [
'user_id' => 'required',
]);

if ($validator->fails()) {
return redirect('supervisors')
->with('alert', 'There is a problem with your input.')
->withErrors($validator)
->withInput();
}

$forum_category = ForumCategory::where('name', '
    Neurology')->first();
if ($forum_category and strcmp($request->input('user_id
    '), '0') != 0) {
$user = User::findOrFail($request->input('user_id'));

if (! $user->supervisor) {
$new_supervisor = new Supervisor();
$new_supervisor['user_id'] = $user->id;
$new_supervisor['forum_category_id'] = $forum_category
    ->id;
$new_supervisor->save();
} else {
```

```
$user->supervisor['forum_category_id'] =
    $forum_category->id;
$user->supervisor->save();
}

return redirect('supervisors')->with('success', '
    Supervisor_on_neurology_saved.');
} else {
return redirect('supervisors');
}
}

/**
* Save nutrition supervisor.
*
* @param \Illuminate\Http\Request $request
* @return \Illuminate\Http\Response
*/
public function saveNutritionSupervisor(Request
    $request) {
$validator = Validator::make($request->all(), [
'user_id2' => 'required',
]);

if ($validator->fails()) {
return redirect('supervisors')
->with('alert', 'There_is_a_problem_with_your_input.')
->withErrors($validator)
->withInput();
}

$forum_category = ForumCategory::where('name', '
    Nutrition')->first();
if ($forum_category and strcmp($request->input('
    user_id2'), '0') != 0) {
$user = User::findOrFail($request->input('user_id2'));

if (! $user->supervisor) {
$new_supervisor = new Supervisor();
$new_supervisor['user_id'] = $user->id;
$new_supervisor['forum_category_id'] = $forum_category
    ->id;
$new_supervisor->save();
} else {
$user->supervisor['forum_category_id'] =
    $forum_category->id;
$user->supervisor->save();
}

return redirect('supervisors')->with('success', '
    Supervisor_on_nutrition_saved.');
} else {
return redirect('supervisors');
}
}

/**
* Save rehabilitation medicine supervisor.
*
* @param \Illuminate\Http\Request $request
* @return \Illuminate\Http\Response
*/
public function saveRehabilitationMedicineSupervisor(
    Request $request) {
$validator = Validator::make($request->all(), [
'user_id3' => 'required',
]);

if ($validator->fails()) {
return redirect('supervisors')
->with('alert', 'There_is_a_problem_with_your_input.')
->withErrors($validator)
->withInput();
}

$forum_category = ForumCategory::where('name', '
    Rehabilitation_Medicine')->first();
if ($forum_category and strcmp($request->input('
    user_id3'), '0') != 0) {
$user = User::findOrFail($request->input('user_id3'));

if (! $user->supervisor) {
$new_supervisor = new Supervisor();
$new_supervisor['user_id'] = $user->id;
$new_supervisor['forum_category_id'] = $forum_category
    ->id;
$new_supervisor->save();
} else {
$user->supervisor['forum_category_id'] =
    $forum_category->id;
$user->supervisor->save();
}

return redirect('supervisors')->with('success', '
    Supervisor_on_rehabilitation_medicine_saved.');
} else {
return redirect('supervisors');
}
}

/**
* Remove supervisor on neurology.
*
* @return \Illuminate\Http\Response
*/
public function removeNeurologySupervisor(Request
    $request) {
$validator = Validator::make($request->all(), [
'user_id4' => 'required',
]);

if ($validator->fails()) {
return redirect('supervisors')
->with('alert', 'There_is_a_problem_with_your_input.')
->withErrors($validator)
->withInput();
}

$forum_category = ForumCategory::where('name', '
    Neurology')->first();
```

```
if ($forum_category and strcmp($request->input('
    user_id4'), '0') != 0) {
$user = User::findOrFail($request->input('user_id4'));

if ($user->supervisor) {
$user->supervisor['forum_category_id'] = null;
$user->supervisor->save();
}

return redirect('supervisors')->with('success', '
    Supervisor_on_neurology_removed.');
} else {
return redirect('supervisors');
}
}

/**
* Remove supervisor on nutrition.
*
* @return \Illuminate\Http\Response
*/
public function removeNutritionSupervisor(Request
    $request) {
$validator = Validator::make($request->all(), [
'user_id5' => 'required',
]);

if ($validator->fails()) {
return redirect('supervisors')
->with('alert', 'There_is_a_problem_with_your_input.')
->withErrors($validator)
->withInput();
}

$forum_category = ForumCategory::where('name', '
    Nutrition')->first();
if ($forum_category and strcmp($request->input('
    user_id5'), '0') != 0) {
$user = User::findOrFail($request->input('user_id5'));

if ($user->supervisor) {
$user->supervisor['forum_category_id'] = null;
$user->supervisor->save();
}

return redirect('supervisors')->with('success', '
    Supervisor_on_nutrition_removed.');
} else {
return redirect('supervisors');
}
}

/**
* Remove supervisor on rehabilitation medicine.
*
* @return \Illuminate\Http\Response
*/
public function removeRehabilitationMedicineSupervisor(
    Request $request) {
$validator = Validator::make($request->all(), [
'user_id6' => 'required',
]);

if ($validator->fails()) {
return redirect('supervisors')
->with('alert', 'There_is_a_problem_with_your_input.')
->withErrors($validator)
->withInput();
}

$forum_category = ForumCategory::where('name', '
    Rehabilitation_Medicine')->first();
if ($forum_category and strcmp($request->input('
    user_id6'), '0') != 0) {
$user = User::findOrFail($request->input('user_id6'));

if ($user->supervisor) {
$user->supervisor['forum_category_id'] = null;
$user->supervisor->save();
}

return redirect('supervisors')->with('success', '
    Supervisor_on_rehabilitation_medicine_removed.');
} else {
return redirect('supervisors');
}
}

}
```

Extra Controller

```
<?php namespace App\Http\Controllers;

use App\Models\DailyRecord;
use App\Models\File;
use App\Models\ForumCategory;
use App\Models\Hospital;
use App\Models\HospitalMedicalProfession;
use App\Models\HospitalNutritionistProfession;
use App\Models\HospitalPhysiatristProfession;
use App\Http\Requests;

use App\Models\LaboratoryResult;
use App\Models\LocalAdmin;
use App\Models\MedicalProfessional;
use App\Models\MedicalRecordAttribute;
use App\Models\MedicalRecordAttributeType;
use App\Models\MedicationRegimen;
use App\Models\Nutritionist;
use App\Models\NutritionistRecordAttribute;
use App\Models\NutritionistRecordAttributeType;
use App\Models\NutritionistRegimen;
use App\Models\Patient;
use App\Models\PatientAttribute;
use App\Models\PatientAttributeType;
use App\Models\Physiatrist;
use App\Models\PhysiatristRecordAttribute;
use App\Models\PhysiatristRecordAttributeType;
use App\Models\PhysiatristRegimen;
```

```
use App\Models\PortalSectionItem;
use App\Models\Supervisor;
use App\Models\SystemAdmin;
use App\User;
use Illuminate\Http\RedirectResponse;

class ExtraController extends Controller {

public function displayAll() {
$all_entries = [];
$all_entries[0] = ['Users', User::all()->toArray()];
$all_entries[1] = ['Hospitals', Hospital::all()->
    toArray()];
$all_entries[2] = ['System Admins', SystemAdmin::all()
    ->toArray()];
$all_entries[3] = ['Local Admins', LocalAdmin::all()->
    toArray()];
$all_entries[4] = ['Medical Professionals',
    MedicalProfessional::all()->toArray()];
$all_entries[5] = ['Nutritionists', Nutritionist::all()
    ->toArray()];
$all_entries[6] = ['Physiatrists', Physiatrist::all()->
    toArray()];
$all_entries[7] = ['Patients', Patient::all()->toArray
    ()];
$all_entries[8] = ['Hospital Medical Professions',
    HospitalMedicalProfession::all()->toArray()];
$all_entries[9] = ['Hospital Nutritionist Professions',
     HospitalNutritionistProfession::all()->toArray()
    ];
$all_entries[10] = ['Hospital Physiatrist Professions',
     HospitalPhysiatristProfession::all()->toArray()];
return $all_entries;
}

/**
* display 2
*
* @return array
*/
public function displayAll2() {
$all_entries = [];
$all_entries[0] = ['Patients', Patient::all()->toArray
    ()];
$all_entries[1] = ['Patient Attribute Types',
    PatientAttributeType::all()->toArray()];
$all_entries[2] = ['Patient Attributes',
    PatientAttribute::all()->toArray()];
$all_entries[3] = ['Medical Record Attribute Types',
    MedicalRecordAttributeType::all()->toArray()];
$all_entries[4] = ['Medical Record Attributes',
    MedicalRecordAttribute::all()->toArray()];
$all_entries[5] = ['Nutritionist Record Attribute Types
    ', NutritionistRecordAttributeType::all()->toArray
    ()];
$all_entries[6] = ['Nutritionist Record Attributes',
    NutritionistRecordAttribute::all()->toArray()];
$all_entries[7] = ['Physiatrist Record Attribute Types'
    , PhysiatristRecordAttributeType::all()->toArray()
    ];
$all_entries[8] = ['Physiatrist Record Attributes',
    PhysiatristRecordAttribute::all()->toArray()];
$all_entries[9] = ['Laboratory Results',
    LaboratoryResult::all()->toArray()];
$all_entries[10] = ['Files', File::all()->toArray()];
$all_entries[11] = ['Medication Regimens',
    MedicationRegimen::all()->toArray()];
$all_entries[12] = ['Nutritionist Regimens',
    NutritionistRegimen::all()->toArray()];
$all_entries[13] = ['Physiatrist Regimens',
    PhysiatristRegimen::all()->toArray()];
return $all_entries;
}

public function fill($table){
switch($table){
case 0: // Hospital
Hospital::create(['name' => 'Philippine General
    Hospital', 'address' => '123 Padre Faura St., 
    Ermita, Manila']);
Hospital::create(['name' => 'Manila Doctors Hospital',
    'address' => '123 Padre Faura St., Ermita, Manila'
    ]);
Hospital::create(['name' => 'Chinese General Hospital',
     'address' => '123 Padre Faura St., Ermita, Manila
    ']);
Hospital::create(['name' => 'University of Santo Tomas
    Hospital', 'address' => '123 Padre Faura St., 
    Ermita, Manila']);
break;
case 1: // Patient Attribute Type
PatientAttributeType::create(['type' => 'civil_status'
    ]);
PatientAttributeType::create(['type' => '
    mother_s_maiden_first_name']);
PatientAttributeType::create(['type' => '
    mother_s_maiden_middle_name']);
PatientAttributeType::create(['type' => '
    mother_s_maiden_last_name']);
PatientAttributeType::create(['type' => '
    father_s_first_name']);
PatientAttributeType::create(['type' => '
    father_s_middle_name']);
PatientAttributeType::create(['type' => '
    father_s_last_name']);
PatientAttributeType::create(['type' => '
    spouse_s_first_name']);
PatientAttributeType::create(['type' => '
    spouse_s_middle_name']);
PatientAttributeType::create(['type' => '
    spouse_s_last_name']);
PatientAttributeType::create(['type' => '
    permanent_address']);
PatientAttributeType::create(['type' => '
    current_address']);
PatientAttributeType::create(['type' => '
    landline_number']);
PatientAttributeType::create(['type' => 'mobile_number'
    ]);
PatientAttributeType::create(['type' => 'place_of_birth
```

```
        ']);
    PatientAttributeType::create(['type' => 'religion']);
    PatientAttributeType::create(['type' => 'nationality'])
        ;
    PatientAttributeType::create(['type' => '
        highest_educational_attainment']);
    PatientAttributeType::create(['type' => 'occupation']);
    PatientAttributeType::create(['type' => 'company']);
    PatientAttributeType::create(['type' => '
        philhealth_number']);
    PatientAttributeType::create(['type' => '
        common_reference_number']);
    PatientAttributeType::create(['type' => '
        contact_person_s_first_name']);
    PatientAttributeType::create(['type' => '
        contact_person_s_middle_name']);
    PatientAttributeType::create(['type' => '
        contact_person_s_last_name']);
    PatientAttributeType::create(['type' => '
        contact_person_s_landline_number']);
    PatientAttributeType::create(['type' => '
        contact_person_s_mobile_number']);
    PatientAttributeType::create(['type' => '
        contact_person_s_e-mail_address']);
    PatientAttributeType::create(['type' => '
        contact_person_s_address']);
    PatientAttributeType::create(['type' => '
        relationship_to_family_member']);
    PatientAttributeType::create(['type' => '
        type_of_stroke_of_family_member']);
    PatientAttributeType::create(['type' => '
        disease_or_attack']);
    PatientAttributeType::create(['type' => '
        other_diseases_or_attacks']);
    PatientAttributeType::create(['type' => '
        type_of_surgery']);
    PatientAttributeType::create(['type' => '
        year_of_surgery']);
    break;
    case 2: // Medical Record Attribute Type
    MedicalRecordAttributeType::create(['type' => 'problems
        ']);
    MedicalRecordAttributeType::create(['type' => '
        description']);
    MedicalRecordAttributeType::create(['type' => '
        medications']);
    MedicalRecordAttributeType::create(['type' => '
        radiation']);
    MedicalRecordAttributeType::create(['type' => '
        alleviating_factors']);
    MedicalRecordAttributeType::create(['type' => '
        temporal_patterns']);
    MedicalRecordAttributeType::create(['type' => 'weight'
        ]);
    MedicalRecordAttributeType::create(['type' => 'height'
        ]);
    MedicalRecordAttributeType::create(['type' => '
        systolic_blood_pressure']);
    MedicalRecordAttributeType::create(['type' => '
        diastolic_blood_pressure']);
    MedicalRecordAttributeType::create(['type' => '
        temperature']);
    MedicalRecordAttributeType::create(['type' => '
        pulse_rate']);
    MedicalRecordAttributeType::create(['type' => '
        respiration']);
    MedicalRecordAttributeType::create(['type' => 'general'
        ]);
    MedicalRecordAttributeType::create(['type' => 'eyes']);
    MedicalRecordAttributeType::create(['type' => '
        ears_nose_throat']);
    MedicalRecordAttributeType::create(['type' => '
        cardiovascular']);
    MedicalRecordAttributeType::create(['type' => '
        gastrointestinal']);
    MedicalRecordAttributeType::create(['type' => '
        genitourinary']);
    MedicalRecordAttributeType::create(['type' => '
        musculoskeletal']);
    MedicalRecordAttributeType::create(['type' => 'skin']);
    MedicalRecordAttributeType::create(['type' => '
        neurologic']);
    MedicalRecordAttributeType::create(['type' => '
        psychiatric']);
    MedicalRecordAttributeType::create(['type' => '
        endocrine']);
    MedicalRecordAttributeType::create(['type' => '
        lymphatic']);
    MedicalRecordAttributeType::create(['type' => '
        immunologic']);
    MedicalRecordAttributeType::create(['type' => '
        level_of_consciousness']);
    MedicalRecordAttributeType::create(['type' => '
        loc_questions']);
    MedicalRecordAttributeType::create(['type' => '
        loc_commands']);
    MedicalRecordAttributeType::create(['type' => '
        best_gaze']);
    MedicalRecordAttributeType::create(['type' => 'visual'
        ]);
    MedicalRecordAttributeType::create(['type' => '
        facial_palsy']);
    MedicalRecordAttributeType::create(['type' => '
        motor_arm_left']);
    MedicalRecordAttributeType::create(['type' => '
        motor_arm_right']);
    MedicalRecordAttributeType::create(['type' => '
        motor_leg_left']);
    MedicalRecordAttributeType::create(['type' => '
        motor_leg_right']);
    MedicalRecordAttributeType::create(['type' => '
        limb_ataxia']);
    MedicalRecordAttributeType::create(['type' => 'sensory'
        ]);
    MedicalRecordAttributeType::create(['type' => '
        best_language']);
    MedicalRecordAttributeType::create(['type' => '
        dysarthia']);
    MedicalRecordAttributeType::create(['type' => '
        extinction_and_inattention']);
```

```
MedicalRecordAttributeType::create(['type' => '
    latest_nihss_score']);
MedicalRecordAttributeType::create(['type' => '
    diagnosis']);
MedicalRecordAttributeType::create(['type' => 'plans'])
    ;
break;
case 3: // Nutritionist Record Attribute Type
NutritionistRecordAttributeType::create(['type' => '
    problems']);
NutritionistRecordAttributeType::create(['type' => '
    meal_snack_pattern']);
NutritionistRecordAttributeType::create(['type' => '
    food_intake']);
NutritionistRecordAttributeType::create(['type' => '
    alcohol_intake']);
NutritionistRecordAttributeType::create(['type' => '
    dietary_supplements']);
NutritionistRecordAttributeType::create(['type' => '
    readiness_to_change']);
NutritionistRecordAttributeType::create(['type' => '
    weight_change']);
NutritionistRecordAttributeType::create(['type' => '
    physical_activity_history']);
NutritionistRecordAttributeType::create(['type' => '
    weight']);
NutritionistRecordAttributeType::create(['type' => '
    height']);
NutritionistRecordAttributeType::create(['type' => 'bmi
    ']);
NutritionistRecordAttributeType::create(['type' => '
    total_energy_intake']);
NutritionistRecordAttributeType::create(['type' => '
    total_fat_intake_and_saturated_fat_intake']);
NutritionistRecordAttributeType::create(['type' => '
    body_compartment_estimates']);
NutritionistRecordAttributeType::create(['type' => '
    recommended_body_weight']);
NutritionistRecordAttributeType::create(['type' => '
    estimated_energy_needs']);
NutritionistRecordAttributeType::create(['type' => '
    nutrition_diagnosis']);
NutritionistRecordAttributeType::create(['type' => '
    nutrition_plan']);
break;
case 4: // Physiatrist Record Attribute Type
PhysiatristRecordAttributeType::create(['type' => '
    problems']);
PhysiatristRecordAttributeType::create(['type' => '
    description']);
PhysiatristRecordAttributeType::create(['type' => '
    medications']);
PhysiatristRecordAttributeType::create(['type' => '
    radiation']);
PhysiatristRecordAttributeType::create(['type' => '
    alleviating_factors']);
PhysiatristRecordAttributeType::create(['type' => '
    temporal_patterns']);
PhysiatristRecordAttributeType::create(['type' => '
    weight']);
PhysiatristRecordAttributeType::create(['type' => '
    height']);
PhysiatristRecordAttributeType::create(['type' => '
    systolic_blood_pressure']);
PhysiatristRecordAttributeType::create(['type' => '
    diastolic_blood_pressure']);
PhysiatristRecordAttributeType::create(['type' => '
    temperature']);
PhysiatristRecordAttributeType::create(['type' => '
    pulse_rate']);
PhysiatristRecordAttributeType::create(['type' => '
    respiration']);
PhysiatristRecordAttributeType::create(['type' => '
    general']);
PhysiatristRecordAttributeType::create(['type' => '
    head_eyes_ears_neck_throat']);
PhysiatristRecordAttributeType::create(['type' => '
    lungs']);
PhysiatristRecordAttributeType::create(['type' => '
    cardiovascular']);
PhysiatristRecordAttributeType::create(['type' => '
    abdomen']);
PhysiatristRecordAttributeType::create(['type' => '
    extremities']);
PhysiatristRecordAttributeType::create(['type' => '
    diagnosis']);
PhysiatristRecordAttributeType::create(['type' => '
    rehabilitation_medicine_plan']);
break;
case 5:
ForumCategory::create(['name' => 'Neurology']);
ForumCategory::create(['name' => 'Nutrition']);
ForumCategory::create(['name' => 'Rehabilitation_
    Medicine']);
break;
default:;
}
return redirect('extra/all2');
}

/**
* Fill all.
*
* @return \Illuminate\Http\Response
*/
public function fillAll() {
$this->fill(0);
$this->fill(1);
$this->fill(2);
$this->fill(3);
$this->fill(4);
$this->fill(5);
return redirect('extra/all2');
}

/**
* Unlock users
*
* @return RedirectResponse
*/
```

```
public function unlockUsers() {
$users = User::all();
foreach($users as $user){
$user->is_unlocked = true;
$user->save();
}
return redirect('extra/all');
}

/**
* Unlock email
*
* @return RedirectResponse
*/
public function unlockEmails() {
$users = User::all();
foreach($users as $user){
$user->is_email_confirmed = true;
$user->save();
}
return redirect('extra/all');
}

/**
* Unlock hospitals
*
* @return RedirectResponse
*/
public function unlockHospitals() {
$hospitals = Hospital::all();
foreach($hospitals as $hospital){
$hospital->is_unlocked = true;
$hospital->save();
}
return redirect('extra/all');
}

/**
* Delete database records
*
* @param int $table
* @return RedirectResponse
*/
public function delete($table) {
switch($table) {
case 0: // Hospital
$hospitals = Hospital::all();
foreach ($hospitals as $hospital) {
$hospital->delete();
}
break;
case 1: // Patient Attribute Type
$patient_attribute_types = PatientAttributeType::all();
foreach ($patient_attribute_types as
    $patient_attribute_type) {
$patient_attribute_type->delete();
}
break;
case 2: // Medical Record Attribute Type
$medical_record_attribute_types =
    MedicalRecordAttributeType::all();
foreach ($medical_record_attribute_types as
    $medical_record_attribute_type) {
$medical_record_attribute_type->delete();
}
break;
case 3: // Nutritionist Record Attribute Type
$nutritionist_record_attribute_types =
    NutritionistRecordAttributeType::all();
foreach ($nutritionist_record_attribute_types as
    $nutritionist_record_attribute_type) {
$nutritionist_record_attribute_type->delete();
}
break;
case 4: // Physiatrist Record Attribute Type
$physiatrist_record_attribute_types =
    PhysiatristRecordAttributeType::all();
foreach ($physiatrist_record_attribute_types as
    $physiatrist_record_attribute_type) {
$physiatrist_record_attribute_type->delete();
}
break;
case 5: // Patient Attribute
$patient_attributes = PatientAttribute::all();
foreach ($patient_attributes as $patient_attribute) {
$patient_attribute->delete();
}
break;
case 6: // Medical Record Attribute
$medical_record_attributes = MedicalRecordAttribute::
    all();
foreach ($medical_record_attributes as
    $medical_record_attribute) {
$medical_record_attribute->delete();
}
break;
case 7: // Nutritionist Record Attribute
$nutritionist_record_attributes =
    NutritionistRecordAttribute::all();
foreach ($nutritionist_record_attributes as
    $nutritionist_record_attribute) {
$nutritionist_record_attribute->delete();
}
break;
case 8: // Physiatrist Record Attribute
$physiatrist_record_attributes =
    PhysiatristRecordAttribute::all();
foreach ($physiatrist_record_attributes as
    $physiatrist_record_attribute) {
$physiatrist_record_attribute->delete();
}
break;
case 9: // Laboratory Result
$laboratory_results = LaboratoryResult::all();
foreach ($laboratory_results as $laboratory_result) {
$file = $laboratory_result->file;
if ($file)
\Illuminate\Support\Facades\File::delete($file->path .
    '/' . $file->filename);
$laboratory_result->delete();
```

```
}
break;
case 10: // Medical Record Attribute
$files = File::all();
foreach ($files as $file) {
\Illuminate\Support\Facades\File::delete($file->path .
    '/' . $file->filename);
$file->delete();
}
break;
case 11: // User
$users = User::all();
foreach ($users as $user) {
$user->delete();
}
break;
case 12: // Patient
$patients = Patient::all();
foreach ($patients as $patient) {
$patient->delete();
}
break;
case 13: // Medical Professional
$medical_professionals = MedicalProfessional::all();
foreach ($medical_professionals as
    $medical_professional) {
$medical_professional->delete();
}
break;
case 14: // Nutritionist
$nutritionists = Nutritionist::all();
foreach ($nutritionists as $nutritionist) {
$nutritionist->delete();
}
break;
case 15: // Physiatrist
$physiatrists = Physiatrist::all();
foreach ($physiatrists as $physiatrist) {
$physiatrist->delete();
}
break;
case 16: // Hospital Medical Profession
$hospital_medical_professions =
    HospitalMedicalProfession::all();
foreach ($hospital_medical_professions as
    $hospital_medical_profession) {
$hospital_medical_profession->delete();
}
break;
case 17: // Hospital Nutritionist Profession
$hospital_nutritionist_professions =
    HospitalNutritionistProfession::all();
foreach ($hospital_nutritionist_professions as
    $hospital_nutritionist_profession) {
$hospital_nutritionist_profession->delete();
}
break;
case 18: // Hospital Physiatrist Profession
$hospital_physiatrist_professions =
    HospitalPhysiatristProfession::all();
foreach ($hospital_physiatrist_professions as
    $hospital_physiatrist_profession) {
$hospital_physiatrist_profession->delete();
}
break;
default:;
}
return redirect('extra/all2');
}

/**
* Create user_1 and patient_1
*
* @return \Illuminate\Http\Response
*/
public function createUsers() {
$user = new User();
$user['username'] = "system_admin";
$user['email'] = "jaferrer4@up.edu.ph";
$user['password'] = bcrypt("12345678");
$user['first_name'] = "John_Rafael";
$user['middle_name'] = "Aguila";
$user['last_name'] = "Ferrer";
$user['sex'] = "male";
$user['birthday'] = "1985-11-22";
$user['is_unlocked'] = true;
$user['is_email_confirmed'] = true;
$user->save();

$hospital = Hospital::all()->first();
$medical_professional = new MedicalProfessional();
$user->medicalProfessional()->save(
    $medical_professional);

$system_admin = new SystemAdmin();
$user->systemAdmin()->save($system_admin);

$hospital_profession = new HospitalMedicalProfession();
$hospital_profession->hospital_id = $hospital->id;
$hospital_profession['is_approved'] = true;
$medical_professional->hospitalMedicalProfessions()->
    save($hospital_profession);

$patient_user = new User();
$patient_user['username'] = "patient_1";
$patient_user['email'] = "raferrer95@gmail.com";
$patient_user['password'] = bcrypt("12345678");
$patient_user['first_name'] = "Isabella";
$patient_user['middle_name'] = "Nuyles";
$patient_user['last_name'] = "Inosantos";
$patient_user['sex'] = "female";
$patient_user['birthday'] = "1974-10-22";
$patient_user['is_unlocked'] = true;
$patient_user['is_email_confirmed'] = true;
$patient_user->save();

$patient = new Patient([
'hospital_id_number' => "2016-00001",
]);
$patient['hospital_id'] = $hospital->id;
```

```
$patient['medical_professional_id'] =
    $medical_professional->id;
$patient_user->patient()->save($patient);

return redirect('extra/all');
}

}
```

Authenticate Middleware

```
<?php namespace App\Http\Middleware;

use Closure;
use Illuminate\Contracts\Auth\Guard;

class Authenticate {

/**
* The Guard implementation.
*
* @var Guard
*/
protected $auth;

/**
* Create a new filter instance.
*
* @param  Guard  $auth
*/
public function __construct(Guard $auth)
{
$this->auth = $auth;
}

/**
* Handle an incoming request.
*
* @param  \Illuminate\Http\Request  $request
* @param  \Closure  $next
* @return mixed
*/
public function handle($request, Closure $next)
{
if ($this->auth->guest()) {
if ($request->ajax()) {
return response('Unauthorized.', 401);
} else {
return redirect()->guest('auth/login');
}

} else {
$user = $request->user();
if ($user) {
if ((! $user->is_unlocked) or (! $user->
    is_email_confirmed)) {
$this->auth->logout();
return redirect()->guest('auth/login');
}
}

return $next($request);
}

}
```

Local Admin Middleware

```
<?php namespace App\Http\Middleware;

use App\Models\Event;
use App\User;
use Closure;

class LocalAdminMiddleware {

/**
* Check if the local admin owns the event.
*
* @param \App\User $user
* @param int $event_id
* @return \Illuminate\Http\Response
*/
public function checkLocalAdminAndEvent(User $user =
    null, $event_id = 0) {
$is_authorized = false;
$event = Event::findOrFail($event_id);

if ($user) {
if ($user->localAdmin) {
if ($user->localAdmin->hospital and $event->hospital) {
if ($user->localAdmin->hospital->id == $event->hospital
    ->id) {
$is_authorized = true;
}
}
}
}

return $is_authorized;
}

/**
* Check if the local admin owns the hospital.
*
* @param \App\User $user
* @param int $hospital_id
* @return \Illuminate\Http\Response
*/
public function checkLocalAdminAndHospital(User $user =
     null, $hospital_id = 0) {
$is_authorized = false;

if ($user and $hospital_id > 0) {
if ($user->localAdmin) {
if ($user->localAdmin->hospital) {
if ($user->localAdmin->hospital->id == $hospital_id) {
$is_authorized = true;
}
```

```
}
}
}

return $is_authorized;
}

/**
* Handle an incoming request.
*
* @param  \Illuminate\Http\Request  $request
* @param  \Closure  $next
* @return mixed
*/
public function handle($request, Closure $next) {
$is_authorized = false;
$hospital_id = $request->hospital_id;
$event_id = $request->event_id;
$user = $request->user();

if ($user) {
if ($user->localAdmin) {
if (isset($hospital_id) and isset($event_id)) {
$is_local_admin_of_hospital = $this->
    checkLocalAdminAndHospital($user, $hospital_id);
$is_event_of_local_admin = $this->
    checkLocalAdminAndEvent($user, $event_id);
if ($is_local_admin_of_hospital and
    $is_event_of_local_admin) {
$is_authorized = true;
}

} else if (isset($hospital_id)) {
$is_authorized = $this->checkLocalAdminAndHospital(
    $user, $hospital_id);
}
}
}

if (! $is_authorized) {
return redirect('/');
} else {
return $next($request);
}
}

}
```

Maintain Patient Account Middleware

```
<?php namespace App\Http\Middleware;

use App\Models\Patient;
use App\User;
use Closure;

class MaintainPatientAccountMiddleware {

/**
* MaintainPatientAccountMiddleware constructor.
*
* @param \App\Http\Middleware\ProfessionalMiddleware
    $professionalMiddleware
*/
public function __construct(ProfessionalMiddleware
    $professionalMiddleware) {
$this->professionalMiddleware = $professionalMiddleware
    ;
}

/**
* Check if user can maintain patient.
*
* @param \App\User $user
* @param int $patient_id
* @return \Illuminate\Http\Response
*/
public function checkMaintainingPatient(User $user =
    null, $patient_id = 0) {
$is_authorized = false;

$patient = Patient::findOrFail($patient_id);
if ($user) {
if ($user->medicalProfessional) {
if ($patient->medicalProfessional == null) {
$is_authorized = true;
}
} else if ($user->nutritionist) {
if ($patient->nutritionist == null) {
$is_authorized = true;
}
} else if ($user->physiatrist) {
if ($patient->physiatrist == null) {
$is_authorized = true;
}
}
}

return $is_authorized;
}

/**
* Handle an incoming request.
*
* @param  \Illuminate\Http\Request  $request
* @param  \Closure  $next
* @return mixed
*/
public function handle($request, Closure $next) {
$is_authorized = false;
$user = $request->user();
$hospital_id = $request->hospital_id;
$patient_id = $request->patient_id;

if ($user and $hospital_id and $patient_id) {
$is_patient_in_hospital = $this->professionalMiddleware
    ->checkPatientInHospital($hospital_id, $patient_id
    );
$is_user_in_hospital = $this->professionalMiddleware->
    checkStaffInHospital($user, $hospital_id);
```

```
if ($is_patient_in_hospital and $is_user_in_hospital) {
$is_authorized = $this->checkMaintainingPatient($user,
    $patient_id);
}
}

if (! $is_authorized) {
return redirect('/');
} else {
return $next($request);
}
}

}
```

MedicalProfessional Middleware

```
<?php namespace App\Http\Middleware;

use App\Models\DailyRecord;
use App\Models\File;
use App\Models\LaboratoryResult;
use App\Models\MedicalProfessional;
use App\Models\MedicationRegimen;
use App\Models\Patient;
use App\User;
use Closure;

class MedicalProfessionalMiddleware {

/**
* MedicalProfessionalMiddleware constructor.
*
* @param \App\Http\Middleware\PatientMiddleware
    $patientMiddleware
*/
public function __construct(PatientMiddleware
    $patientMiddleware) {
$this->patientMiddleware = $patientMiddleware;
}

/**
* Check if the patient is a patient of the medical
    professional.
*
* @param \App\Models\MedicalProfessional
    $medical_professional
* @param int $patient_id
* @return boolean
*/
public function checkPatientFromMedicalProfessional(
MedicalProfessional $medical_professional = null,
$patient_id = 0
) {
$is_authorized = false;
$patient = Patient::findOrFail($patient_id);

if ($patient->medicalProfessional) {
if ($medical_professional) {
if ($patient->medicalProfessional->id ==
    $medical_professional->id) {
$is_authorized = true;
}
}
}

return $is_authorized;
}

/**
* Check if the specified patient account is maintained
    by a medical professional or by a patient.
*
* @param \App\User $user
* @param int $patient_id
* @return \Illuminate\Http\Response
*/
public function checkIfMedicalProfessionalOrPatient(
    User $user = null, $patient_id = 0) {
$is_authorized = false;

if ($user and $patient_id > 0) {
if ($user->medicalProfessional) { // user is a medical
    professional
// check if medical professional maintains the patient
$is_authorized = $this->
    checkPatientFromMedicalProfessional($user->
    medicalProfessional, $patient_id);
} else if ($user->patient) { // user is a patient
// check if the user is the specified patient
$is_authorized = $this->patientMiddleware->
    checkIfPatient($user, $patient_id);
}
}

return $is_authorized;
}

/**
* Handle an incoming request.
*
* @param  \Illuminate\Http\Request  $request
* @param  \Closure  $next
* @return mixed
*/
public function handle($request, Closure $next)
{
$is_authorized = false;
$user = $request->user();
$patient_id = $request->patient_id;
$daily_record_id = $request->daily_record_id;
$laboratory_result_id = $request->laboratory_result_id;
$medication_regimen_id = $request->
    medication_regimen_id;

if ($user) {
if (isset($patient_id)) {
$is_authorized = $this->
    checkIfMedicalProfessionalOrPatient($user,
```

```
        $patient_id);

} else if (isset($daily_record_id)) {
$daily_record = DailyRecord::findOrFail(
        $daily_record_id);
if ($daily_record->patient) {
$is_authorized = $this->
        checkIfMedicalProfessionalOrPatient($user,
        $daily_record->patient->id);
}

} else if (isset($laboratory_result_id)) {
$laboratory_result = LaboratoryResult::findOrFail(
        $laboratory_result_id);
if ($laboratory_result->patient) {
$is_authorized = $this->
        checkIfMedicalProfessionalOrPatient($user,
$laboratory_result->patient->id);
}

} else if (isset($medication_regimen_id)) {
$medication_regimen = MedicationRegimen::findOrFail(
        $medication_regimen_id);
if ($medication_regimen->patient) {
$is_authorized = $this->
        checkIfMedicalProfessionalOrPatient($user,
$medication_regimen->patient->id);
}

} else if ($user->medicalProfessional) {
$is_authorized = true;
}
}

if (! $is_authorized)
return redirect('/');
else
return $next($request);
}

}
```

Medical Resource Middleware

```
<?php namespace App\Http\Middleware;

use App\Models\MedicalResource;
use App\User;
use Closure;

class MedicalResourceMiddleware {

/**
* Check if the user can manage medical resources.
*
* @param \App\User $user
* @param int $medical_resource_id
* @return \Illuminate\Http\Response
*/
public function checkIfCanManageMedicalResource(User
        $user = null, $medical_resource_id = 0) {
$is_authorized = false;
$medical_resource = MedicalResource::findOrFail(
        $medical_resource_id);

if ($medical_resource->user) {
if ($user) {
if ($medical_resource->user->id == $user->id) {
$is_authorized = true;
}
}
}

return $is_authorized;
}

/**
* Handle an incoming request.
*
* @param  \Illuminate\Http\Request  $request
* @param  \Closure  $next
* @return mixed
*/
public function handle($request, Closure $next) {
$is_authorized = false;
$user = $request->user();
$medical_resource_id = $request->medical_resource_id;

if ($user) {
if ($user->supervisor) {
$is_authorized = true;

} else if (isset($medical_resource_id)) {
$is_authorized = $this->checkIfCanManageMedicalResource
        ($user, $medical_resource_id);
}
}

if (! $is_authorized)
return redirect('/');
else
return $next($request);
}

}
```

Nutritionist Middleware

```
<?php namespace App\Http\Middleware;

use App\Models\DailyRecord;
use App\Models\File;
use App\Models\LaboratoryResult;
use App\Models\Nutritionist;
use App\Models\NutritionistRegimen;
use App\Models\Patient;
use App\User;
use Closure;

class NutritionistMiddleware {
```

```
/**
* NutritionistMiddleware constructor.
*
* @param \App\Http\Middleware\PatientMiddleware
    $patientMiddleware
*/
public function __construct(PatientMiddleware
    $patientMiddleware) {
$this->patientMiddleware = $patientMiddleware;
}

/**
* Check if the patient is a patient of the nutritionist
    .
*
* @param \App\Models\Nutritionist $nutritionist
* @param int $patient_id
* @return boolean
*/
public function checkPatientFromNutritionist(
    Nutritionist $nutritionist = null, $patient_id =
    0) {
$is_authorized = false;
$patient = Patient::findOrFail($patient_id);

if ($patient->nutritionist) {
if ($nutritionist) {
if ($patient->nutritionist->id == $nutritionist->id) {
$is_authorized = true;
}
}
}

return $is_authorized;
}

/**
* Check if the specified patient account is maintained
    by a nutritionist or by a patient.
*
* @param \App\User $user
* @param int $patient_id
* @return \Illuminate\Http\Response
*/
public function checkIfNutritionistOrPatient(User $user
     = null, $patient_id = 0) {
$is_authorized = false;

if ($user and $patient_id > 0) {
if ($user->nutritionist) { // user is a nutritionist
// check if nutritionist maintains the patient
$is_authorized = $this->checkPatientFromNutritionist(
    $user->nutritionist, $patient_id);
} else if ($user->patient) { // user is a patient
// check if the user is the specified patient
$is_authorized = $this->patientMiddleware->
    checkIfPatient($user, $patient_id);
}
}

return $is_authorized;
}

/**
* Handle an incoming request.
*
* @param  \Illuminate\Http\Request  $request
* @param  \Closure  $next
* @return mixed
*/
public function handle($request, Closure $next)
{
$is_authorized = false;
$user = $request->user();
$patient_id = $request->patient_id;
$daily_record_id = $request->daily_record_id;
$laboratory_result_id = $request->laboratory_result_id;
$nutritionist_regimen_id = $request->
    nutritionist_regimen_id;

if ($user) {
if (isset($patient_id)) {
$is_authorized = $this->checkIfNutritionistOrPatient(
    $user, $patient_id);

} else if (isset($daily_record_id)) {
$daily_record = DailyRecord::findOrFail(
    $daily_record_id);
if ($daily_record->patient) {
$is_authorized = $this->checkIfNutritionistOrPatient(
    $user, $daily_record->patient->id);
}

} else if (isset($laboratory_result_id)) {
$laboratory_result = LaboratoryResult::findOrFail(
    $laboratory_result_id);
if ($laboratory_result->patient) {
$is_authorized = $this->checkIfNutritionistOrPatient(
    $user, $laboratory_result->patient->id);
}

} else if (isset($nutritionist_regimen_id)) {
$nutritionist_regimen = NutritionistRegimen::findOrFail
    ($nutritionist_regimen_id);
if ($nutritionist_regimen->patient) {
$is_authorized = $this->checkIfNutritionistOrPatient(
    $user, $nutritionist_regimen->patient->id);
}

} else if ($user->nutritionist) {
$is_authorized = true;
}
}

if (! $is_authorized)
return redirect('/');
else
return $next($request);
}
```

```
}
```

Patient Middleware

```
<?php namespace App\Http\Middleware;

use App\Models\LaboratoryResult;
use App\Models\MedicalCondition;
use App\Models\MedicationRecord;
use App\Models\MedicationRegimen;
use App\Models\NutritionistRecord;
use App\Models\NutritionistRegimen;
use App\Models\PhysiatristRecord;
use App\Models\PhysiatristRegimen;
use App\User;
use Closure;

class PatientMiddleware {

/**
* PatientMiddleware constructor.
*
* @param \App\Http\Middleware\ProfessionalMiddleware
    $professionalMiddleware
*/
public function __construct(ProfessionalMiddleware
    $professionalMiddleware) {
$this->professionalMiddleware = $professionalMiddleware
    ;
}

/**
* Check if the user is the specified patient.
*
* @param \App\User $user
* @param int $patient_id
* @return boolean
*/
public function checkIfPatient(User $user = null,
    $patient_id = 0) {
$is_authorized = false;

if ($user and $patient_id > 0) {
if ($user->patient) {
if ($user->patient->id == $patient_id) {
$is_authorized = true;
}
}
}

return $is_authorized;
}

/**
* Check if the specified patient account is maintained
    by a professional or by a patient.
*
* @param \App\User $user
* @param int $patient_id
* @return \Illuminate\Http\Response
*/
public function checkIfProfessionalOrPatient(User $user
     = null, $patient_id = 0) {
$is_authorized = false;

if ($user and $patient_id > 0) {
if ($this->professionalMiddleware->checkProfession(
    $user)) { // user is a professional
// check if professional user maintains the patient
$is_authorized = $this->professionalMiddleware->
    checkPatientFromUser($user, $patient_id);
} else if ($user->patient) { // user is a patient
// check if the user is the specified patient
$is_authorized = $this->checkIfPatient($user,
    $patient_id);
}
}

return $is_authorized;
}

/**
* Handle an incoming request.
*
* @param  \Illuminate\Http\Request  $request
* @param  \Closure  $next
* @return mixed
*/
public function handle($request, Closure $next)
{
$is_authorized = false;
$user = $request->user();
$patient_id = $request->patient_id;
$medical_condition_id = $request->medical_condition_id;
$laboratory_result_id = $request->laboratory_result_id;
$medication_record_id = $request->medication_record_id;
$nutritionist_record_id = $request->
    nutritionist_record_id;
$physiatrist_record_id = $request->
    physiatrist_record_id;
$medication_regimen_id = $request->
    medication_regimen_id;
$nutritionist_regimen_id = $request->
    nutritionist_regimen_id;
$physiatrist_regimen_id = $request->
    physiatrist_regimen_id;

if ($user) {
if (isset($patient_id)) {
$is_authorized = $this->checkIfProfessionalOrPatient(
    $user, $patient_id);

} else if (isset($daily_record_id)) {
$daily_record = DailyRecord::findOrFail(
    $daily_record_id);
if ($daily_record->patient) {
$is_authorized = $this->checkIfProfessionalOrPatient(
    $user, $daily_record->patient->id);
}
```

```
} else if (isset($medical_condition_id)) {
$medical_condition = MedicalCondition::findOrFail(
    $medical_condition_id);
if ($medical_condition->patient) {
$is_authorized = $this->checkIfProfessionalOrPatient(
    $user, $medical_condition->patient->id);
}

} else if (isset($laboratory_result_id)) {
$laboratory_result = LaboratoryResult::findOrFail(
    $laboratory_result_id);
if ($laboratory_result->patient) {
$is_authorized = $this->checkIfProfessionalOrPatient(
    $user, $laboratory_result->patient->id);
}

} else if (isset($medication_record_id)) {
$medication_record = MedicationRecord::findOrFail(
    $medication_record_id);
if ($medication_record->patient) {
$is_authorized = $this->checkIfProfessionalOrPatient(
    $user, $medication_record->patient->id);
}

} else if (isset($nutritionist_record_id)) {
$nutritionist_record = NutritionistRecord::findOrFail(
    $nutritionist_record_id);
if ($nutritionist_record->patient) {
$is_authorized = $this->checkIfProfessionalOrPatient(
    $user, $nutritionist_record->patient->id);
}

} else if (isset($physiatrist_record_id)) {
$physiatrist_record = PhysiatristRecord::findOrFail(
    $physiatrist_record_id);
if ($physiatrist_record->patient) {
$is_authorized = $this->checkIfProfessionalOrPatient(
    $user, $physiatrist_record->patient->id);
}

} else if (isset($medication_regimen_id)) {
$medication_regimen = MedicationRegimen::findOrFail(
    $medication_regimen_id);
if ($medication_regimen->patient) {
$is_authorized = $this->checkIfProfessionalOrPatient(
    $user, $medication_regimen->patient->id);
}

} else if (isset($nutritionist_regimen_id)) {
$nutritionist_regimen = NutritionistRegimen::findOrFail
    ($nutritionist_regimen_id);
if ($nutritionist_regimen->patient) {
$is_authorized = $this->checkIfProfessionalOrPatient(
    $user, $nutritionist_regimen->patient->id);
}

} else if (isset($physiatrist_regimen_id)) {
$physiatrist_regimen = PhysiatristRegimen::findOrFail(
    $physiatrist_regimen_id);
if ($physiatrist_regimen->patient) {
$is_authorized = $this->checkIfProfessionalOrPatient(
    $user, $physiatrist_regimen->patient->id);
}

} else {
if ($user->patient) { // user is a patient
$is_authorized = true;
}
}
}

if (! $is_authorized)
return redirect('/');
else
return $next($request);
}

}
```

Physiatrist Middleware

```
<?php namespace App\Http\Middleware;

use App\Models\DailyRecord;
use App\Models\File;
use App\Models\LaboratoryResult;
use App\Models\Patient;
use App\Models\Physiatrist;
use App\Models\PhysiatristRegimen;
use App\User;
use Closure;

class PhysiatristMiddleware {

/**
* PhysiatristMiddleware constructor.
*
* @param \App\Http\Middleware\PatientMiddleware
    $patientMiddleware
*/
public function __construct(PatientMiddleware
    $patientMiddleware) {
$this->patientMiddleware = $patientMiddleware;
}

/**
* Check if the patient is a patient of the physiatrist.
*
* @param \App\Models\Physiatrist $physiatrist
* @param int $patient_id
* @return boolean
*/
public function checkPatientFromPhysiatrist(Physiatrist
     $physiatrist = null, $patient_id = 0) {
$is_authorized = false;
$patient = Patient::findOrFail($patient_id);

if ($patient->physiatrist) {
if ($physiatrist) {
```

```
if ($patient->physiatrist->id == $physiatrist->id) {
$is_authorized = true;
}
}
}

return $is_authorized;
}

/**
* Check if the specified patient account is maintained
    by a physiatrist or by a patient.
*
* @param \App\User $user
* @param int $patient_id
* @return \Illuminate\Http\Response
*/
public function checkIfPhysiatristOrPatient(User $user
    = null, $patient_id = 0) {
$is_authorized = false;

if ($user and $patient_id > 0) {
if ($user->physiatrist) { // user is a physiatrist
// check if physiatrist maintains the patient
$is_authorized = $this->checkPatientFromPhysiatrist(
    $user->physiatrist, $patient_id);
} else if ($user->patient) { // user is a patient
// check if the user is the specified patient
$is_authorized = $this->patientMiddleware->
    checkIfPatient($user, $patient_id);
}
}

return $is_authorized;
}

/**
* Handle an incoming request.
*
* @param  \Illuminate\Http\Request  $request
* @param  \Closure  $next
* @return mixed
*/
public function handle($request, Closure $next)
{
$is_authorized = false;
$user = $request->user();
$patient_id = $request->patient_id;
$daily_record_id = $request->daily_record_id;
$laboratory_result_id = $request->laboratory_result_id;
$physiatrist_regimen_id = $request->
    physiatrist_regimen_id;

if ($user) {
if (isset($patient_id)) {
$is_authorized = $this->checkIfPhysiatristOrPatient(
    $user, $patient_id);

} else if (isset($daily_record_id)) {
$daily_record = DailyRecord::findOrFail(
    $daily_record_id);
if ($daily_record->patient) {
$is_authorized = $this->checkIfPhysiatristOrPatient(
    $user, $daily_record->patient->id);
}

} else if (isset($laboratory_result_id)) {
$laboratory_result = LaboratoryResult::findOrFail(
    $laboratory_result_id);
if ($laboratory_result->patient) {
$is_authorized = $this->checkIfPhysiatristOrPatient(
    $user, $laboratory_result->patient->id);
}

} else if (isset($physiatrist_regimen_id)) {
$physiatrist_regimen = PhysiatristRegimen::findOrFail(
    $physiatrist_regimen_id);
if ($physiatrist_regimen->patient) {
$is_authorized = $this->checkIfPhysiatristOrPatient(
    $user, $physiatrist_regimen->patient->id);
}

} else if ($user->nutritionist) {
$is_authorized = true;
}
}

if (! $is_authorized)
return redirect('/');
else
return $next($request);
}

}
```

Professional Middleware

```
<?php namespace App\Http\Middleware;

use App\Models\Hospital;
use App\Models\Patient;
use App\User;
use Closure;

class ProfessionalMiddleware {

/**
* Check if user is the local administrator of the
    hospital.
*
* @param \App\User $user
* @param int $hospital_id
* @return boolean
*/
public function checkLocalAdmin(User $user = null,
    $hospital_id = 0) {
$is_authorized = false;

if ($user and $hospital_id > 0) {
if ($user->localAdmin) {
```

```
if ($user->localAdmin->hospital) {
if ($user->localAdmin->hospital->id == $hospital_id) {
$is_authorized = true;
}
}
}
}

return $is_authorized;
}

/**
* Check if the user is a professional.
*
* @param \App\User $user
* @return boolean
*/
public function checkProfession(User $user = null) {
if ($user) {
if ($user->medicalProfessional or $user->nutritionist
    or $user->physiatrist) {
return true;
} else {
return false;
}
}
}

/**
* Check if the patient is from the specified hospital.
*
* @param int $hospital_id
* @param int $patient_id
* @return boolean
*/
public function checkPatientInHospital($hospital_id =
    0, $patient_id = 0) {
$is_authorized = false;

if ($hospital_id > 0 and $patient_id > 0) {
$hospital = Hospital::findOrFail($hospital_id);
if ($hospital->patients) {
foreach ($hospital->patients as $patient) {
if ($patient->id == $patient_id) {
$is_authorized = true;
break;
}
}
}
}

return $is_authorized;
}

/**
* Determine if user is a staff of the hospital.
*
* @param \App\User $user
* @param int $hospital_id
* @return boolean
*/
public function checkStaffInHospital(User $user = null,
     $hospital_id = 0) {
$is_authorized = false;

if ($user and $hospital_id > 0) {
if ($this->checkProfession($user)) {
if ($user->medicalProfessional) {
$professions = $user->medicalProfessional->
    hospitalMedicalProfessions();
} else if ($user->nutritionist) {
$professions = $user->nutritionist->
    hospitalNutritionistProfessions();
} else {
$professions = $user->physiatrist->
    hospitalPhysiatristProfessions();
}

if ($professions) {
if (count($professions->where('hospital_id',
    $hospital_id)->where('is_approved', true)
->get()) > 0) {
$is_authorized = true;
}
}
}
}

return $is_authorized;
}

/**
* Check if the patient is a patient of the user.
*
* @param \App\User $user
* @param int $patient_id
* @return boolean
*/
public function checkPatientFromUser(User $user = null,
     $patient_id = 0) {
$is_authorized = false;

if ($user and $patient_id > 0) {
if ($this->checkProfession($user)) {
if ($user->medicalProfessional) {
$patients = $user->medicalProfessional->patients();
} else if ($user->nutritionist) {
$patients = $user->nutritionist->patients();
} else {
$patients = $user->physiatrist->patients();
}

if ($patients) {
if (count($patients->where('id', $patient_id)->get()) >
    0) {
$is_authorized = true;
}
}
}
}
```

```
return $is_authorized;
}

/**
* Handle an incoming request.
*
* @param  \Illuminate\Http\Request  $request
* @param  \Closure  $next
* @return mixed
*/
public function handle($request, Closure $next) {
$is_authorized = false;
$user = $request->user();
$hospital_id = $request->hospital_id;
$patient_id = $request->patient_id;

if ($user) {
if (isset($hospital_id)) {
if ($user->localAdmin) {
$is_authorized = $this->checkLocalAdmin($user,
    $hospital_id);

} else if (isset($patient_id)) {
$is_patient_in_hospital = $this->checkPatientInHospital
    ($hospital_id, $patient_id);
$is_user_in_hospital = $this->checkStaffInHospital(
    $user, $hospital_id);

if ($is_patient_in_hospital and $is_user_in_hospital) {
$is_authorized = true;
}

} else {
$is_authorized = $this->checkStaffInHospital($user,
    $hospital_id);
}

} else if (isset($patient_id)) {
$is_authorized = $this->checkPatientFromUser($user,
    $patient_id);

} else if ($this->checkProfession($user)) {
$is_authorized = true;
}
}

if (! $is_authorized)
return redirect('/');
else
return $next($request);
}

}
```

Redirect If Authenticated Middleware

```
<?php namespace App\Http\Middleware;

use Closure;
use Illuminate\Contracts\Auth\Guard;
use Illuminate\Http\RedirectResponse;

class RedirectIfAuthenticated {

/**
* The Guard implementation.
*
* @var Guard
*/
protected $auth;

/**
* Create a new filter instance.
*
* @param  Guard  $auth
* @return void
*/
public function __construct(Guard $auth)
{
$this->auth = $auth;
}

/**
* Handle an incoming request.
*
* @param  \Illuminate\Http\Request  $request
* @param  \Closure  $next
* @return mixed
*/
public function handle($request, Closure $next)
{
if ($this->auth->check())
{
return new RedirectResponse(url('/'));
}

return $next($request);
}

}
```

Staff Middleware

```
<?php namespace App\Http\Middleware;

use App\User;
use Closure;

class StaffMiddleware {

/**
* Check if the user is a professional.
*
* @param \App\User $user
* @return \Illuminate\Http\Response
*/
public function checkIfStaff(User $user = null) {
$is_authorized = false;
```

```
if ($user) {
if ($user->medicalProfessional or $user->nutritionist
    or $user->physiatrist) {
$is_authorized = true;
}
}

return $is_authorized;
}

/**
* Handle an incoming request.
*
* @param  \Illuminate\Http\Request  $request
* @param  \Closure  $next
* @return mixed
*/
public function handle($request, Closure $next) {
$is_authorized = false;
$user = $request->user();

if ($user) {
$is_authorized = $this->checkIfStaff($user);
}

if (! $is_authorized) {
return redirect('/');
} else {
return $next($request);
}
}

}
```

Supervisor Middleware

```
<?php namespace App\Http\Middleware;

use App\Models\Forum;
use App\Models\MedicalResource;
use App\User;
use Closure;

class SupervisorMiddleware {

/**
* Check if the user is a supervisor.
*
* @param \App\User $user
* @return \Illuminate\Http\Response
*/
public function checkIfSupervisor(User $user = null) {
$is_authorized = false;

if ($user) {
if ($user->supervisor) {
$is_authorized = true;
}
}

return $is_authorized;
}

/**
* Check if the supervisor user owns the medical
    resource.
*
* @param \App\User $user
* @param int $medical_resource_id
* @return \Illuminate\Http\Response
*/
public function checkSupervisorAndMedicalResource(User
    $user = null, $medical_resource_id = 0) {
$is_authorized = false;

if ($user and $medical_resource_id > 0) {
$medical_resource = MedicalResource::findOrFail(
    $medical_resource_id);
if ($user->supervisor) {
if ($user->supervisor->forumCategory and
    $medical_resource->forumCategory) {
if ($user->supervisor->forumCategory->id ==
    $medical_resource->forumCategory->id) {
$is_authorized = true;
}
}
}
}

return $is_authorized;
}

/**
* Check if the supervisor user owns the forum category.
*
* @param \App\User $user
* @param int $forum_category_id
* @return \Illuminate\Http\Response
*/
public function checkSupervisorAndForumCategory(User
    $user = null, $forum_category_id = 0) {
$is_authorized = false;

if ($user and $forum_category_id > 0) {
if ($user->supervisor) {
if ($user->supervisor->forumCategory) {
if ($user->supervisor->forumCategory->id ==
    $forum_category_id) {
$is_authorized = true;
}
}
}
}

return $is_authorized;
}

/**
* Check if the supervisor user owns the forum thread.
*
```

```
* @param \App\User $user
* @param int $forum_id
* @return \Illuminate\Http\Response
*/
public function checkSupervisorAndForum(User $user =
    null, $forum_id = 0) {
$is_authorized = false;

if ($user and $forum_id > 0) {
$forum = Forum::findOrFail($forum_id);
if ($user->supervisor and $forum->supervisor) {
if ($user->supervisor->id == $forum->supervisor->id) {
$is_authorized = true;
}
}
}

return $is_authorized;
}

/**
* Handle an incoming request.
*
* @param  \Illuminate\Http\Request  $request
* @param  \Closure  $next
* @return mixed
*/
public function handle($request, Closure $next) {
$is_authorized = false;
$user = $request->user();
$medical_resource_id = $request->medical_resource_id;
$forum_category_id = $request->forum_category_id;
$forum_id = $request->forum_id;

if ($user) {
if ($this->checkIfSupervisor($user)) {
if (isset($medical_resource_id)) {
$is_authorized = $this->
    checkSupervisorAndMedicalResource($user,
    $medical_resource_id);
} else if (isset($forum_category_id)) {
$is_authorized = $this->checkSupervisorAndForumCategory
    ($user, $forum_category_id);
} else if (isset($forum_id)) {
$is_authorized = $this->checkSupervisorAndForum($user,
    $forum_id);
} else {
$is_authorized = true;
}
}
}

if (! $is_authorized) {
return redirect('/');
} else {
return $next($request);
}
}
}
```

System Admin Middleware

```
<?php namespace App\Http\Middleware;

use Closure;

class SystemAdminMiddleware {

/**
* Handle an incoming request.
*
* @param  \Illuminate\Http\Request  $request
* @param  \Closure  $next
* @return mixed
*/
public function handle($request, Closure $next) {
$user = $request->user();

if ($user) {
if ($user->systemAdmin) {
return $next($request);
}
}

return redirect('/');
}

}
```

Unlocked Hospital Middleware

```
<?php namespace App\Http\Middleware;

use App\Models\Hospital;
use Closure;

class UnlockedHospitalMiddleware {

/**
* Handle an incoming request.
*
* @param  \Illuminate\Http\Request  $request
* @param  \Closure  $next
* @return mixed
*/
public function handle($request, Closure $next) {
$is_authorized = false;
$hospital_id = $request->hospital_id;

if (isset($hospital_id)) {
$hospital = Hospital::findOrFail($request->hospital_id)
    ;
if ($hospital->is_unlocked) {
$is_authorized = true;
}
}

if (! $is_authorized) {
return redirect('/');
} else {
return $next($request);
```

```
}
}

}
```

Kernel.php

```
<?php namespace App\Http;

use Illuminate\Foundation\Http\Kernel as HttpKernel;

class Kernel extends HttpKernel {

/**
* The application's global HTTP middleware stack.
*
* @var array
*/
protected $middleware = [
'Illuminate\Foundation\Http\Middleware\
    CheckForMaintenanceMode',
'Illuminate\Cookie\Middleware\EncryptCookies',
'Illuminate\Cookie\Middleware\
    AddQueuedCookiesToResponse',
'Illuminate\Session\Middleware\StartSession',
'Illuminate\View\Middleware\ShareErrorsFromSession',
'App\Http\Middleware\VerifyCsrfToken',
];

/**
* The application's route middleware.
*
* @var array
*/
protected $routeMiddleware = [
'auth' => 'App\Http\Middleware\Authenticate',
'auth.basic' => 'Illuminate\Auth\Middleware\
    AuthenticateWithBasicAuth',
'guest' => 'App\Http\Middleware\RedirectIfAuthenticated
    ',
'medicalResource' => 'App\Http\Middleware\
    MedicalResourceMiddleware',
'staff' => 'App\Http\Middleware\StaffMiddleware',
'professional' => 'App\Http\Middleware\
    ProfessionalMiddleware',
'maintainPatient' => 'App\Http\Middleware\
    MaintainPatientAccountMiddleware',
'patient' => 'App\Http\Middleware\PatientMiddleware',
'medicalProfessional' => 'App\Http\Middleware\
    MedicalProfessionalMiddleware',
'nutritionist' => 'App\Http\Middleware\
    NutritionistMiddleware',
'physiatrist' => 'App\Http\Middleware\
    PhysiatristMiddleware',
'supervisor' => 'App\Http\Middleware\
    SupervisorMiddleware',
'localAdmin' => 'App\Http\Middleware\
    LocalAdminMiddleware',
'systemAdmin' => 'App\Http\Middleware\
    SystemAdminMiddleware',
'unlockedHospital' => 'App\Http\Middleware\
    UnlockedHospitalMiddleware',
];

}
```

Routes.php

```
<?php

/*
|--------------------------------------------------------------------------

| Application Routes
|--------------------------------------------------------------------------

|
| Here is where you can register all of the routes for
    an application.
| It's a breeze. Simply tell Laravel the URIs it should
     respond to
| and give it the controller to call when that URI is
    requested.
|
*/

Route::get('auth/register/confirm/{confirmation_code}',
     [
'as' => 'confirmation_path',
'uses' => 'Auth\AuthController@confirm'
]);

Route::controllers([
'auth' => 'Auth\AuthController',
'password' => 'Auth\PasswordController',
]);

Route::get('/', 'HomeControllers\HomeController@index')
    ;
Route::get('about-us', 'HomeControllers\
    HomeController@aboutUs');
Route::get('users/{username}', 'HomeControllers\
    HomeController@showUser');

Route::get('notifications', 'HomeControllers\
    NotificationsController@index');
Route::get('notifications/{notification_id}', '
    HomeControllers\
    NotificationsController@viewNotification');

Route::get('medical-resources', 'HomeControllers\
    MedicalResourcesController@viewAllMedicalResources
    ');
Route::get('medical-resources/create', 'HomeControllers
    \MedicalResourcesController@create');
Route::post('medical-resources', 'HomeControllers\
    MedicalResourcesController@store');
Route::get('medical-resources/category/{category}', '
    HomeControllers\
    MedicalResourcesController@viewMedicalResourcesInCategory
```

```
');
Route::get('medical-resources/{medical_resource_id}', '
    HomeControllers\MedicalResourcesController@show');
Route::get('medical-resources/{medical_resource_id}/
    download', 'HomeControllers\
    MedicalResourcesController@download');
Route::get('medical-resources/{medical_resource_id}/
    edit', 'HomeControllers\
    MedicalResourcesController@edit');
Route::patch('medical-resources/{medical_resource_id}',
     'HomeControllers\
    MedicalResourcesController@update');
Route::delete('medical-resources/{medical_resource_id}/
    delete', 'HomeControllers\
    MedicalResourcesController@destroy');
Route::post('medical-resources/{medical_resource_id}/
    approve', 'HomeControllers\
    MedicalResourcesController@approve');

Route::get('forum-threads/all', 'HomeControllers\
    ForumsController@viewAllForums');
Route::get('forum-threads/create', 'HomeControllers\
    ForumsController@create');
Route::post('forum-threads', 'HomeControllers\
    ForumsController@store');
Route::get('forum-threads/{category}', 'HomeControllers
    \ForumsController@viewForumsInCategory');
Route::get('forum-threads/{category}/{forum_id}', '
    HomeControllers\ForumsController@show');
Route::post('forum-threads/{category}/{forum_id}', '
    HomeControllers\ForumsController@addReply');
Route::post('forum-threads/{category}/{forum_id}/
    moderate', 'HomeControllers\
    ForumsController@moderate');
Route::delete('forum-threads/{category}/{forum_id}/
    delete', 'HomeControllers\ForumsController@destroy
    ');
Route::delete('forum-threads/{category}/{forum_id}/{
    forum_comment_id}', 'HomeControllers\
    ForumsController@destroyReply');

Route::get('announcements-and-events', 'HomeControllers
    \EventsController@viewAllEvents');
Route::get('announcements-and-events/{hospital_name}',
    'HomeControllers\
    EventsController@viewEventsInHospital');
Route::get('announcements-and-events/hospitals/{
    hospital_id}/create', 'HomeControllers\
    EventsController@create');
Route::post('announcements-and-events/hospitals/{
    hospital_id}/create', 'HomeControllers\
    EventsController@store');
Route::get('announcements-and-events/hospitals/{
    hospital_id}/{event_id}', 'HomeControllers\
    EventsController@show');
Route::get('announcements-and-events/hospitals/{
    hospital_id}/{event_id}/edit', 'HomeControllers\
    EventsController@edit');
Route::patch('announcements-and-events/hospitals/{
    hospital_id}/{event_id}/edit', 'HomeControllers\
    EventsController@update');
Route::delete('announcements-and-events/hospitals/{
    hospital_id}/{event_id}/delete', 'HomeControllers\
    EventsController@destroy');

Route::get('profile', 'Users\
    AccountSettingsController@displayAccountSettings')
    ;
Route::get('settings/general', 'Users\
    AccountSettingsController@editGeneralSettings');
Route::patch('settings', 'Users\
    AccountSettingsController@updateGeneralSettings');
Route::get('settings/profile-picture', 'Users\
    AccountSettingsController@uploadProfilePicture');
Route::patch('settings/profile-picture', 'Users\
    AccountSettingsController@updateProfilePicture');
Route::delete('settings/profile-picture', 'Users\
    AccountSettingsController@deleteProfilePicture');
Route::get('settings/password', 'Users\
    AccountSettingsController@changePassword');
Route::post('settings/password', 'Users\
    AccountSettingsController@updatePassword');

Route::get('hospitals', 'Hospitals\
    HospitalsController@index');
Route::get('hospitals/create', 'Hospitals\
    HospitalsController@create');
Route::post('hospitals', 'Hospitals\
    HospitalsController@store');
Route::get('hospitals/{hospital_id}', 'Hospitals\
    HospitalsController@show');
Route::get('hospitals/{hospital_id}/edit', 'Hospitals\
    HospitalsController@edit');
Route::post('hospitals/{hospital_id}/lock', 'Hospitals\
    HospitalsController@lockHospital');
Route::post('hospitals/{hospital_id}/unlock', '
    Hospitals\HospitalsController@unlockHospital');
Route::post('hospitals/{hospital_id}/register', '
    Hospitals\HospitalsController@registerInHospital')
    ;
Route::post('hospitals/{hospital_id}/leave', 'Hospitals
    \HospitalsController@leaveHospital');
Route::patch('hospitals/{hospital_id}', 'Hospitals\
    HospitalsController@update');

Route::get('hospitals/{hospital_id}/patients/{
    patient_id}', 'Hospitals\
    HospitalsController@showPatient');
Route::post('hospitals/{hospital_id}/patients/{
    patient_id}/lock', 'Hospitals\
    HospitalsController@lockPatient');
Route::post('hospitals/{hospital_id}/patients/{
    patient_id}/unlock', 'Hospitals\
    HospitalsController@unlockPatient');
Route::post('hospitals/{hospital_id}/patients/{
    patient_id}/email', 'Hospitals\
    HospitalsController@emailPatient');
Route::post('hospitals/{hospital_id}/patients/{
    patient_id}/maintain-patient', 'Hospitals\
    HospitalsController@maintainPatient');
```

```
Route::get('hospitals/{hospital_id}/{profession}/{
    staff_id}', 'Hospitals\
    HospitalsController@showStaffMember');
Route::post('hospitals/{hospital_id}/{profession}/{
    staff_id}/lock', 'Hospitals\
    HospitalsController@lockStaffMember');
Route::post('hospitals/{hospital_id}/{profession}/{
    staff_id}/unlock', 'Hospitals\
    HospitalsController@unlockStaffMember');
Route::post('hospitals/{hospital_id}/{profession}/{
    staff_id}/email', 'Hospitals\
    HospitalsController@emailStaffMember');

Route::get('patients', 'Patients\
    PatientsController@index');
Route::get('patients/create/{profession_id}', 'Patients
    \PatientsController@create');
Route::post('patients/create/{profession_id}', '
    Patients\PatientsController@store');
Route::get('patients/{patient_id}', 'Patients\
    PatientsController@show');
Route::get('patients/{patient_id}/edit', 'Patients\
    PatientsController@edit');
Route::patch('patients/{patient_id}', 'Patients\
    PatientsController@update');

Route::get('patient-attributes/{patient_id}', 'Patients
    \PatientAttributesController@index');
Route::post('patient-attributes/{patient_id}', '
    Patients\PatientAttributesController@store');

Route::get('medical-records/{patient_id}', '
    MedicalRecords\
    MedicalRecordsController@indexMedicationRecord');
Route::post('medical-records/{patient_id}', '
    MedicalRecords\
    MedicalRecordsController@createMedicationRecord');
Route::get('medical-records/{patient_id}/{
    medication_record_id}', 'MedicalRecords\
    MedicalRecordsController@editMedicationRecord');
Route::post('medical-records/{patient_id}/{
    medication_record_id}', 'MedicalRecords\
    MedicalRecordsController@storeMedicationRecord');

Route::get('nutritionist-records/{patient_id}', '
    MedicalRecords\
    MedicalRecordsController@indexNutritionistRecord')
    ;
Route::post('nutritionist-records/{patient_id}', '
    MedicalRecords\
    MedicalRecordsController@createNutritionistRecord'
    );
Route::get('nutritionist-records/{patient_id}/{
    nutritionist_record}', 'MedicalRecords\
    MedicalRecordsController@editNutritionistRecord');
Route::post('nutritionist-records/{patient_id}/{
    nutritionist_record}', 'MedicalRecords\
    MedicalRecordsController@storeNutritionistRecord')
    ;

Route::get('physiatrist-records/{patient_id}', '
    MedicalRecords\
    MedicalRecordsController@indexPhysiatristRecord');
Route::post('physiatrist-records/{patient_id}', '
    MedicalRecords\
    MedicalRecordsController@createPhysiatristRecord')
    ;
Route::get('physiatrist-records/{patient_id}/{
    physiatrist_record}', 'MedicalRecords\
    MedicalRecordsController@editPhysiatristRecord');
Route::post('physiatrist-records/{patient_id}/{
    physiatrist_record}', 'MedicalRecords\
    MedicalRecordsController@storePhysiatristRecord');

Route::get('medical-conditions/{patient_id}', 'Patients
    \MedicalConditionController@index');
Route::get('medical-conditions/{patient_id}/input', '
    Patients\MedicalConditionController@input');
Route::post('medical-conditions/{patient_id}', '
    Patients\MedicalConditionController@store');
Route::get('medical-conditions/{patient_id}/{
    medical_condition_id}/edit', 'Patients\
    MedicalConditionController@edit');
Route::patch('medical-conditions/{patient_id}/{
    medical_condition_id}', 'Patients\
    MedicalConditionController@update');
Route::post('medical-conditions/{patient_id}/{
    medical_condition_id}/decision', 'Patients\
    MedicalConditionController@makeDecision');

Route::get('referral/{patient_id}', 'Patients\
    MedicalConditionController@refer');
Route::post('referral/{patient_id}/medical-professional
    ', 'Patients\
    MedicalConditionController@referToAMedicalProfessional
    ');
Route::post('referral/{patient_id}/nutritionist', '
    Patients\
    MedicalConditionController@referToANutritionist');
Route::post('referral/{patient_id}/physiatrist', '
    Patients\
    MedicalConditionController@referToAPhysiatrist');

Route::get('schedule-visits', 'Patients\
    RequestVisitsController@indexPatient');
Route::post('schedule-visits/medical-professional', '
    Patients\
    RequestVisitsController@setMedicalProfessionalVisit
    ');
Route::post('schedule-visits/nutritionist', 'Patients\
    RequestVisitsController@setNutritionistVisit');
Route::post('schedule-visits/physiatrist', 'Patients\
    RequestVisitsController@setPhysiatristVisit');
Route::get('approve-visits', 'Patients\
    RequestVisitsController@indexStaff');
Route::post('approve-visits', 'Patients\
    RequestVisitsController@approveVisit');

Route::get('laboratory-results/{patient_id}', '
```

```
        MedicalRecords\LaboratoryResultsController@index')
        ;
Route::get('laboratory-results/{patient_id}/input', '
        MedicalRecords\LaboratoryResultsController@create'
        );
Route::post('laboratory-results/{patient_id}', '
        MedicalRecords\LaboratoryResultsController@store')
        ;
Route::get('laboratory-results/download/{
        laboratory_result_id}', 'MedicalRecords\
        LaboratoryResultsController@download');
Route::get('laboratory-results/edit/{
        laboratory_result_id}', 'MedicalRecords\
        LaboratoryResultsController@edit');
Route::patch('laboratory-results/edit/{
        laboratory_result_id}', 'MedicalRecords\
        LaboratoryResultsController@update');

Route::get('medication-regimens/edit/{
        medication_regimen_id}', 'Regimens\
        MedicationRegimensController@edit');
Route::patch('medication-regimens/edit/{
        medication_regimen_id}', 'Regimens\
        MedicationRegimensController@update');
Route::delete('medication-regimens/delete/{
        medication_regimen_id}', 'Regimens\
        MedicationRegimensController@destroy');
Route::get('medication-regimens/{patient_id}/create', '
        Regimens\MedicationRegimensController@create');
Route::post('medication-regimens/{patient_id}/create',
        'Regimens\MedicationRegimensController@store');
Route::get('medication-regimens/{patient_id}', '
        Regimens\MedicationRegimensController@index');

Route::get('nutritionist-regimens/edit/{
        nutritionist_regimen_id}', 'Regimens\
        NutritionistRegimensController@edit');
Route::patch('nutritionist-regimens/edit/{
        nutritionist_regimen_id}', 'Regimens\
        NutritionistRegimensController@update');
Route::delete('nutritionist-regimens/delete/{
        nutritionist_regimen_id}', 'Regimens\
        NutritionistRegimensController@destroy');
Route::get('nutritionist-regimens/{patient_id}/create',
         'Regimens\NutritionistRegimensController@create')
        ;
Route::post('nutritionist-regimens/{patient_id}/create'
        , 'Regimens\NutritionistRegimensController@store')
        ;
Route::get('nutritionist-regimens/{patient_id}', '
        Regimens\NutritionistRegimensController@index');

Route::get('physiatrist-regimens/edit/{
        physiatrist_regimen_id}', 'Regimens\
        PhysiatristRegimensController@edit');
Route::patch('physiatrist-regimens/edit/{
        physiatrist_regimen_id}', 'Regimens\
        PhysiatristRegimensController@update');
Route::delete('physiatrist-regimens/delete/{
        physiatrist_regimen_id}', 'Regimens\
        PhysiatristRegimensController@destroy');
Route::get('physiatrist-regimens/{patient_id}/create',
        'Regimens\PhysiatristRegimensController@create');
Route::post('physiatrist-regimens/{patient_id}/create',
         'Regimens\PhysiatristRegimensController@store');
Route::get('physiatrist-regimens/{patient_id}', '
        Regimens\PhysiatristRegimensController@index');

Route::get('hospital-admins', 'Users\
        SystemAdminController@manageLocalAdmins');
Route::post('hospital-admins', 'Users\
        SystemAdminController@saveLocalAdmin');
Route::get('supervisors', 'Users\
        SystemAdminController@manageSupervisors');
Route::post('supervisors/neurology', 'Users\
        SystemAdminController@saveNeurologySupervisor');
Route::post('supervisors/nutrition', 'Users\
        SystemAdminController@saveNutritionSupervisor');
Route::post('supervisors/rehabilitation-medicine', '
        Users\
        SystemAdminController@saveRehabilitationMedicineSupervisor
        ');
Route::delete('supervisors/remove-neurology', 'Users\
        SystemAdminController@removeNeurologySupervisor');
Route::delete('supervisors/remove-nutrition', 'Users\
        SystemAdminController@removeNutritionSupervisor');
Route::delete('supervisors/remove-rehabilitation-
        medicine', 'Users\
        SystemAdminController@removeRehabilitationMedicineSupervisor
        ');

Route::get('extra/all', 'ExtraController@displayAll');
Route::get('extra/all2', 'ExtraController@displayAll2')
        ;
Route::get('extra/fill/{table}', 'ExtraController@fill'
        );
Route::get('extra/fillAll', 'ExtraController@fillAll');
Route::get('extra/delete/{table}', '
        ExtraController@delete');
Route::get('extra/unlock-users', '
        ExtraController@unlockUsers');
Route::get('extra/unlock-emails', '
        ExtraController@unlockEmails');
Route::get('extra/unlock-hospitals', '
        ExtraController@unlockHospitals');
Route::get('extra/create-users', '
        ExtraController@createUsers');
```

Form Macro

```
<?php
/**
* HTML5 introduced 13 new input types:
*    url, tel, email, search, number, color, range,
      datetime, datetime-local, date, time, week, month
*
* The benefit of using these types are:
*    Most browsers provide basic auto validation on
      fields such as email addresses
*    Mobile browsers/OS modify the input device (digital
```

```
     keyboard) to match the input
*
* Laravel 4 supports rendering all HTML5 elements out
     of the box, but only 2 (email & url) via alias.
*   eg: Form::text('input_name', 'input_value',
     $options_array);
*
* All other inputs are available by using the input
     class
*   eg: Form::input('number', 'input_name', '
     input_value', $options_array);
*
* This file allows you to use the other 11 HTML
     elements the same way as text, email, and URL.
*
* 1. Create the folder /app/misc, or use your preferred
      directory
* 2. Copy this form_macros.php into the directory
* 3. You should now have access to the shorthand for
     all HTML5 elements
*
* This file manually creates each macro, allowing you
     to customize any given input to add defaults or
     restrictions.
*
* *Note: The only alias which differs from the element
     input type is datetime-local, as the dash breaks
     the alias.
*   Instead, create the element using Input::
     datetimelocal()
*/


/**
* Color input - http://www.w3.org/TR/html-markup/input.
     color.html
*
* Value - A string exactly seven characters long,
     consisting of the following parts, in exactly the
     following order:
*  A "#" character.
*  Six characters in the range 0  9, a  f, and A  F.
*
* Note - Color keywords (for example, strings such as
       red   or   green  ) are not allowed.
*/
Form::macro('color', function($name, $default = NULL,
     $attrs = array())
{
$item = '<input type="color" name="'. $name .'" ';

if ($default) {
$item .= 'value="'. $default .'" ';
}

if (is_array($attrs)) {
foreach ($attrs as $a => $v)
$item .= $a .'="'. $v .'" ';
}
$item .= ">";

return $item;
});


/**
* Date input - http://www.w3.org/TR/html-markup/input.
     date.html
*
* Value - A valid full-date as defined in [RFC 3339],
     with the additional qualification
*   that the year component is four or more digits
     representing a number greater than 0.
*/
Form::macro('date', function($name, $default = NULL,
     $attrs = array())
{
$item = '<input type="date" name="'. $name .'" ';

if ($default) {
$item .= 'value="'. $default .'" ';
}

if (is_array($attrs)) {
foreach ($attrs as $a => $v)
$item .= $a .'="'. $v .'" ';
}
$item .= ">";

return $item;
});


/**
* Datetime input - http://www.w3.org/TR/html-markup/
     input.datetime.html
*
* The input element with a type attribute whose value
     is "datetime" represents a control
* for setting the element s value to a string
     representing a global date and time (with
* timezone information).
*
* Value - A valid date-time as defined in [RFC 3339],
     with these additional qualifications:
*  1. The literal letters T and Z in the date/time
     syntax must always be uppercase
*  2. The date-fullyear production is instead defined
     as four or more digits representing a
*     number greater than 0
*/
Form::macro('datetime', function($name, $default = NULL
     , $attrs = array())
{
$item = '<input type="datetime" name="'. $name .'" ';

if ($default) {
$item .= 'value="'. $default .'" ';
}
```

```
if (is_array($attrs)) {
foreach ($attrs as $a => $v)
$item .= $a .'="'. $v .'" ';
}
$item .= ">";

return $item;
});


/**
* Datetime-local input - http://www.w3.org/TR/html-
    markup/input.datetime-local.html
*
* The input element with a type attribute whose value
    is "datetime-local" represents a control
* for setting the element s value to a string
    representing a local date and time (with no
* timezone information).
*
* Value - A string representing a local date and time.
    The following parts, in exactly the
*  following order:
*  1. A date.
*  2. The literal string "T".
*  3. A time.
*/
Form::macro('datetimelocal', function($name, $default =
    NULL, $attrs = array())
{
$item = '<input type="date" name="'. $name .'" ';

if ($default) {
$item .= 'value="'. $default .'" ';
}

if (is_array($attrs)) {
foreach ($attrs as $a => $v)
$item .= $a .'="'. $v .'" ';
}
$item .= ">";

return $item;
});


/**
* Month input - http://www.w3.org/TR/html-markup/input.
    month.html
*
* The input element with a type attribute whose value
    is "month" represents a control for
* setting the element s value to a string
    representing a month.
*
* Value - A string representing a month. The following
    parts, in exactly the following order:
*  1. Four or more digits representing a number greater
     than 0.
*  2. The literal string "-".
*  3. Two digits, representing the month month, in the
    range 1    month,    12.
*/
Form::macro('month', function($name, $default = NULL,
    $attrs = array())
{
$item = '<input type="month" name="'. $name .'" ';

if ($default) {
$item .= 'value="'. $default .'" ';
}

if (is_array($attrs)) {
foreach ($attrs as $a => $v)
$item .= $a .'="'. $v .'" ';
}
$item .= ">";

return $item;
});


/**
* Number input - http://www.w3.org/TR/html-markup/input
    .number.html
*
* The input element with a type attribute whose value
    is "number" represents a precise control for
* setting the element s value to a string
    representing a number.
*
* Value - A string representing a number.
*/
Form::macro('number', function($name, $default = NULL,
    $attrs = array())
{
$item = '<input type="number" name="'. $name .'" ';

if ($default) {
$item .= 'value="'. $default .'" ';
}

if (is_array($attrs)) {
foreach ($attrs as $a => $v)
$item .= $a .'="'. $v .'" ';
}
$item .= ">";

return $item;
});


/**
* Range input - http://www.w3.org/TR/html-markup/input.
    range.html
*
* The input element with a type attribute whose value
    is "range" represents an imprecise control for
* setting the element s value to a string
    representing a number.
```

```
*
* Value - A string representing a number.
*/
Form::macro('range', function($name, $default = NULL,
    $attrs = array())
{
$item = '<input type="range" name="'. $name .'" ';

if ($default) {
$item .= 'value="'. $default .'" ';
}

if (is_array($attrs)) {
foreach ($attrs as $a => $v)
$item .= $a .'="'. $v .'" ';
}
$item .= ">";

return $item;
});


/**
* Search input - http://www.w3.org/TR/html-markup/input
    .search.html
*
* The input element with a type attribute whose value
    is "search" represents a one-line plain-text
* edit control for entering one or more search terms.
*
* Value - Any string that contains no line feed (U+000A
    , LF ) or carriage return (U+000D, CR )
*   characters.
*/
Form::macro('search', function($name, $default = NULL,
    $attrs = array())
{
$item = '<input type="search" name="'. $name .'" ';

if ($default) {
$item .= 'value="'. $default .'" ';
}

if (is_array($attrs)) {
foreach ($attrs as $a => $v)
$item .= $a .'="'. $v .'" ';
}
$item .= ">";

return $item;
});


/**
* Tel input - http://www.w3.org/TR/html-markup/input.
    tel.html
*
* The input element with a type attribute whose value
    is "tel" represents a one-line plain-text edit
* control for entering a telephone number.
*
* Value - Any string that contains no line feed (U+000A
    , LF ) or carriage return (U+000D, CR )
*   characters.
*/
Form::macro('tel', function($name, $default = NULL,
    $attrs = array())
{
$item = '<input type="tel" name="'. $name .'" ';

if ($default) {
$item .= 'value="'. $default .'" ';
}

if (is_array($attrs)) {
foreach ($attrs as $a => $v)
$item .= $a .'="'. $v .'" ';
}
$item .= ">";

return $item;
});


/**
* Time input - http://www.w3.org/TR/html-markup/input.
    time.html
*
* The input element with a type attribute whose value
    is "time" represents a control for setting the
* element s value to a string representing a time (
    with no timezone information).
*
* Value - A string representing a time (with no
    timezone information). A valid partial-time as
    defined
*  in [RFC 3339].
*/
Form::macro('time', function($name, $default = NULL,
    $attrs = array())
{
$item = '<input type="time" name="'. $name .'" ';

if ($default) {
$item .= 'value="'. $default .'" ';
}

if (is_array($attrs)) {
foreach ($attrs as $a => $v)
$item .= $a .'="'. $v .'" ';
}
$item .= ">";

return $item;
});


/**
* Week input - http://www.w3.org/TR/html-markup/input.
    week.html
```

```
*
* The input element with a type attribute whose value
     is "week" represents a control for setting the
* element s value to a string representing a week.
*
* Value - A string representing a week. The following
     parts, in exactly the following order:
*   1. Four or more digits representing year year, where
      year > 0.
*   2. The literal string "-W".
*   3. Two digits, representing the week week, in the
     range 1     week     maxweek, where maxweek is
     either
*     52 or 53, depending on the particular year.
*/
Form::macro('week', function($name, $default = NULL,
     $attrs = array())
{
$item = '<input type="week" name="'. $name .'" ';

if ($default) {
$item .= 'value="'. $default .'" ';
}

if (is_array($attrs)) {
foreach ($attrs as $a => $v)
$item .= $a .'="'. $v .'" ';
}
$item .= ">";

return $item;
});
```

Macro Service Provider User Model

```
<?php namespace App;

use Carbon\Carbon;
use Illuminate\Auth\Authenticatable;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Auth\Passwords\CanResetPassword;
use Illuminate\Contracts\Auth\Authenticatable as
     AuthenticatableContract;
use Illuminate\Contracts\Auth\CanResetPassword as
     CanResetPasswordContract;

class User extends Model implements
     AuthenticatableContract, CanResetPasswordContract
     {

use Authenticatable, CanResetPassword;

/**
* The database table used by the model.
*
* @var string
*/
protected $table = 'users';

/**
* The attributes that are mass assignable.
*
* @var array
*/
protected $fillable = [
'first_name',
'middle_name',
'last_name',
'sex',
'birthday',
];

/**
* The attributes that are treated as Carbon.
*
* @var array
*/
protected $dates = [
'birthday'
];

/**
* The attributes excluded from the model's JSON form.
*
* @var array
*/
protected $hidden = [
'password',
'remember_token',
'is_unlocked',
'is_email_confirmed',
'confirmation_code',
];

/**
* Do something in birthday attribute after getting it.
*
* @param Carbon $date
* @return array
*/
public function getBirthdayAttribute($date){
return [
Carbon::parse($date),
Carbon::parse($date)->format('F d, Y')
];
}

/**
* Do something in birthday attribute before setting it.
*
* @param Carbon $date
*/
public function setBirthdayAttribute($date){
$this->attributes['birthday'] = Carbon::parse($date);
}

/**
* A user has a file.
*
* @return \Illuminate\Database\Eloquent\Relations\
```

```
     HasOne
*/
public function file(){
return $this->hasOne('App\Models\File');
}

/**
* A user can have many notifications.
*
* @return \Illuminate\Database\Eloquent\Relations\
     HasMany
*/
public function notifications(){
return $this->hasMany('App\Models\Notification');
}

/**
* A user can have many medical resources.
*
* @return \Illuminate\Database\Eloquent\Relations\
     HasMany
*/
public function medicalResources(){
return $this->hasMany('App\Models\MedicalResource');
}

/**
* A user can be a patient or caregiver.
*
* @return \Illuminate\Database\Eloquent\Relations\
     HasOne
*/
public function patient(){
return $this->hasOne('App\Models\Patient');
}

/**
* A user can be a medical professional.
*
* @return \Illuminate\Database\Eloquent\Relations\
     HasOne
*/
public function medicalProfessional(){
return $this->hasOne('App\Models\MedicalProfessional');
}

/**
* A user can be a nutritionist.
*
* @return \Illuminate\Database\Eloquent\Relations\
     HasOne
*/
public function nutritionist(){
return $this->hasOne('App\Models\Nutritionist');
}

/**
* A user can be a physiatrist.
*
* @return \Illuminate\Database\Eloquent\Relations\
     HasOne
*/
public function physiatrist(){
return $this->hasOne('App\Models\Physiatrist');
}

/**
* A user can be a patient or caregiver.
*
* @return \Illuminate\Database\Eloquent\Relations\
     HasOne
*/
public function followUpVisit(){
return $this->hasOne('App\Models\FollowUpVisit');
}

/**
* A user can be a supervisor.
*
* @return \Illuminate\Database\Eloquent\Relations\
     HasOne
*/
public function supervisor(){
return $this->hasOne('App\Models\Supervisor');
}

/**
* A user can be a local admin.
*
* @return \Illuminate\Database\Eloquent\Relations\
     HasOne
*/
public function localAdmin(){
return $this->hasOne('App\Models\LocalAdmin');
}

/**
* A user can be a system admin.
*
* @return \Illuminate\Database\Eloquent\Relations\
     HasOne
*/
public function systemAdmin(){
return $this->hasOne('App\Models\SystemAdmin');
}

/**
* A user can have many portal section items.
*
* @return \Illuminate\Database\Eloquent\Relations\
     HasMany
*/
public function portalSectionItems(){
return $this->hasMany('App\Models\PortalSectionItem');
}

/**
* A user can have many forum posts.
*
* @return \Illuminate\Database\Eloquent\Relations\
```

```
     HasMany
*/
public function forums(){
return $this->hasMany('App\Models\Forum');
}

/**
* A user can have many forum comments.
*
* @return \Illuminate\Database\Eloquent\Relations\
     HasMany
*/
public function forumComments(){
return $this->hasMany('App\Models\ForumComment');
}

/**
* A user can have many contacts.
*
* @return \Illuminate\Database\Eloquent\Relations\
     HasMany
*/
public function contacts(){
return $this->hasMany('App\Models\Contact');
}

/**
* A user can have many posts.
*
* @return \Illuminate\Database\Eloquent\Relations\
     HasMany
*/
public function posts(){
return $this->hasMany('App\Models\Post');
}

/**
* Return full name of user in a "first-name last-name"
     format.
*
* @return var
*/
public function fullName() {
return $this->attributes['first_name'] . ' ' . $this->
     attributes['last_name'];
}

/**
* Return full name of user in a "last-name, first-name
     middle-name" format.
*
* @return var
*/
public function fullName2() {
return $this->attributes['last_name'] . ', ' . $this->
     attributes['first_name'] . ' ' .
$this->attributes['middle_name'];
}

}
```

```
<?php namespace App\Providers;

use Illuminate\Support\ServiceProvider;

class MacroServiceProvider extends ServiceProvider {

/**
* Bootstrap the application services.
*
* @return void
*/
public function boot()
{
require base_path() . '/app/misc/form_macro.php';
}

/**
* Register the application services.
*
* @return void
*/
public function register()
{
//
}

}
```

Decision on Condition Model

```
<?php namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class DecisionOnCondition extends Model {

/**
* The database table used by the model.
*
* @var string
*/
protected $table = 'decision_on_conditions';

/**
* The attributes that are mass assignable.
*
* @var array
*/
protected $fillable = [
'medical_professional_id',
'medical_condition_id',
'decision',
'additional_notes',
];

/**
* The attributes that are treated as Carbon.
*
* @var array
*/
```

```
protected $dates = [];

/**
* The attributes excluded from the model's JSON form.
*
* @var array
*/
protected $hidden = [
'medical_professional_id',
'medical_condition_id',
];

/**
* A decision belongs to exactly one medical
    professional.
*
* @return \Illuminate\Database\Eloquent\Relations\
    BelongsTo
*/
public function medicalProfessional(){
return $this->belongsTo('App\Models\MedicalProfessional
    ');
}

/**
* A decision belongs to exactly one medical condition.
*
* @return \Illuminate\Database\Eloquent\Relations\
    BelongsTo
*/
public function medicalCondition(){
return $this->belongsTo('App\Models\MedicalCondition');
}

}
```

Event Model

```
<?php namespace App\Models;

use Carbon\Carbon;
use Illuminate\Database\Eloquent\Model;

class Event extends Model {

/**
* The database table used by the model.
*
* @var string
*/
protected $table = 'events';

/**
* The attributes that are mass assignable.
*
* @var array
*/
protected $fillable = [
'hospital_id',
'heading',
'description',
'date_of_event',
'time_of_event',
];

/**
* The attributes that are treated as Carbon.
*
* @var array
*/
protected $dates = [
'date_of_event',
];

/**
* The attributes excluded from the model's JSON form.
*
* @var array
*/
protected $hidden = [
'hospital_id',
];

/**
* Do something in date_of_event attribute after getting
    it.
*
* @param Carbon $date
* @return array
*/
public function getDateOfEventAttribute($date){
$array = [Carbon::parse($date)];
if ($date) {
$array[] = Carbon::parse($date)->format('F_d,_Y');
} else {
$array[] = null;
}
return $array;
}

/**
* Do something in date_of_event attribute before
    setting it.
*
* @param Carbon $date
*/
public function setDateOfEventAttribute($date){
if ($date)
$this->attributes['date_of_event'] = Carbon::parse(
    $date);
else
$this->attributes['date_of_event'] = null;
}

/**
* Do something in time_of_event attribute after getting
    it.
*
* @param time $time
* @return array
```

```
*/
public function getTimeOfEventAttribute($time){
$array = [$time];
if ($time) {
$array[] = date("g:i_A", strtotime($time));
} else {
$array[] = null;
}
return $array;
}

/**
* A news of an event belongs to exactly one hospital.
*
* @return \Illuminate\Database\Eloquent\Relations\
    BelongsTo
*/
public function hospital(){
return $this->belongsTo('App\Models\Hospital');
}

}
```

File Model

```
<?php namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class File extends Model {

/**
* The database table used by the model.
*
* @var string
*/
protected $table = 'files';

/**
* The attributes that are mass assignable.
*
* @var array
*/
protected $fillable = [
'filename',
'extension',
'mimeType',
'path',
'user_id',
'medical_resource_id',
'laboratory_result_id',
];

/**
* The attributes that are treated as Carbon.
*
* @var array
*/
protected $dates = [];

/**
* The attributes excluded from the model's JSON form.
*
* @var array
*/
protected $hidden = [
'user_id',
'laboratory_result_id',
];

/**
* A file belongs to exactly one user.
*
* @return \Illuminate\Database\Eloquent\Relations\
    BelongsTo
*/
public function user(){
return $this->belongsTo('App\User');
}

/**
* A file belongs to exactly one medical resource.
*
* @return \Illuminate\Database\Eloquent\Relations\
    BelongsTo
*/
public function medicalResource(){
return $this->belongsTo('App\Models\MedicalResource');
}

/**
* A file belongs to exactly one laboratory result.
*
* @return \Illuminate\Database\Eloquent\Relations\
    BelongsTo
*/
public function laboratoryResult(){
return $this->belongsTo('App\Models\LaboratoryResult');
}

}
```

Follow Up Visit Model

```
<?php namespace App\Models;

use Carbon\Carbon;
use Illuminate\Database\Eloquent\Model;

class FollowUpVisit extends Model {

/**
* The database table used by the model.
*
* @var string
*/
protected $table = 'follow_up_visits';

/**
* The attributes that are mass assignable.
```

```
*
* @var array
*/
protected $fillable = [
'user_id',
'patient_id',
'medical_condition_id',
'date_of_visit',
];

/**
* The attributes that are treated as Carbon.
*
* @var array
*/
protected $dates = [
'date_of_visit',
];

/**
* The attributes excluded from the model's JSON form.
*
* @var array
*/
protected $hidden = [
'user_id',
'patient_id',
'medical_condition_id',
];

/**
* Do something in date_of_visit attribute after getting
      it.
*
* @param Carbon $date
* @return array
*/
public function getDateOfVisitAttribute($date){
$array = [Carbon::parse($date)];
if ($date) {
$array[] = Carbon::parse($date)->format('F d, Y');
} else {
$array[] = null;
}
return $array;
}

/**
* Do something in date_of_visit attribute before
     setting it.
*
* @param Carbon $date
*/
public function setDateOfVisitAttribute($date){
if ($date)
$this->attributes['date_of_visit'] = Carbon::parse(
     $date);
else
$this->attributes['date_of_visit'] = Carbon::now();
}

/**
* A follow-up visit belongs to exactly one user.
*
* @return \Illuminate\Database\Eloquent\Relations\
     BelongsTo
*/
public function user(){
return $this->belongsTo('App\User');
}

/**
* A follow-up visit belongs to exactly one patient.
*
* @return \Illuminate\Database\Eloquent\Relations\
     BelongsTo
*/
public function patient(){
return $this->belongsTo('App\Models\Patient');
}

/**
* A follow-up visit belongs to exactly one medical
     condition.
*
* @return \Illuminate\Database\Eloquent\Relations\
     BelongsTo
*/
public function medicalCondition(){
return $this->belongsTo('App\Models\MedicalCondition');
}

}
```

Forum Model

```
<?php namespace App\Models;

use Carbon\Carbon;
use Illuminate\Database\Eloquent\Model;

class Forum extends Model {

/**
* The database table used by the model.
*
* @var string
*/
protected $table = 'forums';

/**
* The attributes that are mass assignable.
*
* @var array
*/
protected $fillable = [
'user_id',
'supervisor_id',
'forum_category_id',
'title',
```

```
'description',
];

/**
* The attributes that are treated as Carbon.
*
* @var array
*/
protected $dates = [];

/**
* The attributes excluded from the model's JSON form.
*
* @var array
*/
protected $hidden = [
'user_id',
'supervisor_id',
'forum_category_id',
];

/**
* A forum post belongs to exactly one user.
*
* @return \Illuminate\Database\Eloquent\Relations\
    BelongsTo
*/
public function user(){
return $this->belongsTo('App\User');
}

/**
* A forum post belongs to exactly one supervisor.
*
* @return \Illuminate\Database\Eloquent\Relations\
    BelongsTo
*/
public function supervisor(){
return $this->belongsTo('App\Models\Supervisor');
}

/**
* A forum post belongs to exactly one forum category.
*
* @return \Illuminate\Database\Eloquent\Relations\
    BelongsTo
*/
public function forumCategory(){
return $this->belongsTo('App\Models\ForumCategory');
}

/**
* A forum post can have many forum comments.
*
* @return \Illuminate\Database\Eloquent\Relations\
    HasMany
*/
public function forumComments(){
return $this->hasMany('App\Models\ForumComment');
}
}
```

Forum Category Model

```
<?php namespace App\Models;

use Carbon\Carbon;
use Illuminate\Database\Eloquent\Model;

class ForumCategory extends Model {

/**
* The database table used by the model.
*
* @var string
*/
protected $table = 'forum_categories';

/**
* The attributes that are mass assignable.
*
* @var array
*/
protected $fillable = [
'name',
];

/**
* The attributes that are treated as Carbon.
*
* @var array
*/
protected $dates = [];

/**
* The attributes excluded from the model's JSON form.
*
* @var array
*/
protected $hidden = [];

/**
* A forum category can have many supervisors.
*
* @return \Illuminate\Database\Eloquent\Relations\
    HasMany
*/
public function supervisors(){
return $this->hasMany('App\Models\Supervisor');
}

/**
* A forum can have many forums.
*
* @return \Illuminate\Database\Eloquent\Relations\
    HasMany
*/
public function forums(){
return $this->hasMany('App\Models\Forum');
```

```
}

}
```

Forum Comment Model

```
<?php namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class ForumComment extends Model {

/**
* The database table used by the model.
*
* @var string
*/
protected $table = 'forum_comments';

/**
* The attributes that are mass assignable.
*
* @var array
*/
protected $fillable = [
'comment',
];

/**
* The attributes that are treated as Carbon.
*
* @var array
*/
protected $dates = [];

/**
* The attributes excluded from the model's JSON form.
*
* @var array
*/
protected $hidden = [
'user_id',
'forum_id',
];

/**
* A forum comment belongs to exactly one user.
*
* @return \Illuminate\Database\Eloquent\Relations\
    BelongsTo
*/
public function user(){
return $this->belongsTo('App\User');
}

/**
* A forum comment belongs to exactly one forum.
*
* @return \Illuminate\Database\Eloquent\Relations\
    BelongsTo
*/
public function forum(){
return $this->belongsTo('App\Models\Forum');
}

}
```

Hospital Model

```
<?php namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Hospital extends Model {

/**
* The database table used by the model.
*
* @var string
*/
protected $table = 'hospitals';

/**
* The attributes that are mass assignable.
*
* @var array
*/
protected $fillable = [
'name',
'address',
'landline_number',
'mobile_number',
'fax_number',
'email_address'
];

/**
* The attributes that are treated as Carbon.
*
* @var array
*/
protected $dates = [];

/**
* The attributes excluded from the model's JSON form.
*
* @var array
*/
protected $hidden = [];

/**
* A hospital has exactly one local admin.
*
* @return \Illuminate\Database\Eloquent\Relations\
    HasOne
*/
public function localAdmin(){
return $this->hasOne('App\Models\LocalAdmin');
}
```

```
/**
* A hospital can have many events.
*
* @return \Illuminate\Database\Eloquent\Relations\
    HasMany
*/
public function events(){
return $this->hasMany('App\Models\Event');
}

/**
* A hospital can have many patients.
*
* @return \Illuminate\Database\Eloquent\Relations\
    HasMany
*/
public function patients(){
return $this->hasMany('App\Models\Patient');
}

/**
* A hospital can have many medical professions.
*
* @return \Illuminate\Database\Eloquent\Relations\
    HasMany
*/
public function hospitalMedicalProfessions(){
return $this->hasMany('App\Models\
    HospitalMedicalProfession');
}

/**
* A hospital can have many nutritionist professions.
*
* @return \Illuminate\Database\Eloquent\Relations\
    HasMany
*/
public function hospitalNutritionistProfessions(){
return $this->hasMany('App\Models\
    HospitalNutritionistProfession');
}

/**
* A hospital can have many physiatrist professions.
*
* @return \Illuminate\Database\Eloquent\Relations\
    HasMany
*/
public function hospitalPhysiatristProfessions(){
return $this->hasMany('App\Models\
    HospitalPhysiatristProfession');
}

}
```

Hospital Medical Profession Model

```
<?php namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class HospitalMedicalProfession extends Model {

/**
* The database table used by the model.
*
* @var string
*/
protected $table = 'hospital_medical_professions';

/**
* The attributes that are mass assignable.
*
* @var array
*/
protected $fillable = [];

/**
* The attributes that are treated as Carbon.
*
* @var array
*/
protected $dates = [];

/**
* The attributes excluded from the model's JSON form.
*
* @var array
*/
protected $hidden = [
'hospital_id',
'medical_professional_id',
'is_approved',
];

/**
* A medical profession belongs to a hospital.
*
* @return \Illuminate\Database\Eloquent\Relations\
    BelongsTo
*/
public function hospital(){
return $this->belongsTo('App\Models\Hospital');
}

/**
* A medical profession belongs to a medical
    professional.
*
* @return \Illuminate\Database\Eloquent\Relations\
    BelongsTo
*/
public function medicalProfessional(){
return $this->belongsTo('App\Models\MedicalProfessional
    ');
}

}
```

Hospital Nutritionist Profession Model

```php
<?php namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class HospitalNutritionistProfession extends Model {

/**
* The database table used by the model.
*
* @var string
*/
protected $table = 'hospital_nutritionist_professions';

/**
* The attributes that are mass assignable.
*
* @var array
*/
protected $fillable = [];

/**
* The attributes that are treated as Carbon.
*
* @var array
*/
protected $dates = [];

/**
* The attributes excluded from the model's JSON form.
*
* @var array
*/
protected $hidden = [
'hospital_id',
'nutritionist_id',
'is_approved',
];

/**
* A nutritionist profession belongs to a hospital.
*
* @return \Illuminate\Database\Eloquent\Relations\
    BelongsTo
*/
public function hospital(){
return $this->belongsTo('App\Models\Hospital');
}

/**
* A nutritionist profession belongs to a nutritionist.
*
* @return \Illuminate\Database\Eloquent\Relations\
    BelongsTo
*/
public function nutritionist(){
return $this->belongsTo('App\Models\Nutritionist');
}

}
```

Hospital Physiatrist Profession Model

```php
<?php namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class HospitalPhysiatristProfession extends Model {

/**
* The database table used by the model.
*
* @var string
*/
protected $table = 'hospital_physiatrist_professions';

/**
* The attributes that are mass assignable.
*
* @var array
*/
protected $fillable = [];

/**
* The attributes that are treated as Carbon.
*
* @var array
*/
protected $dates = [];

/**
* The attributes excluded from the model's JSON form.
*
* @var array
*/
protected $hidden = [
'hospital_id',
'physiatrist_id',
'is_approved',
];

/**
* A physiatrist profession belongs to a hospital.
*
* @return \Illuminate\Database\Eloquent\Relations\
    BelongsTo
*/
public function hospital(){
return $this->belongsTo('App\Models\Hospital');
}

/**
* A physiatrist profession belongs to a physiatrist.
*
* @return \Illuminate\Database\Eloquent\Relations\
    BelongsTo
*/
public function physiatrist(){
return $this->belongsTo('App\Models\Physiatrist');
}

}
```

Laboratory Result Model

```
<?php namespace App\Models;

use Carbon\Carbon;
use Illuminate\Database\Eloquent\Model;

class LaboratoryResult extends Model {

/**
* The database table used by the model.
*
* @var string
*/
protected $table = 'laboratory_results';

/**
* The attributes that are mass assignable.
*
* @var array
*/
protected $fillable = [
'patient_id',
'type_of_laboratory_exam',
'unit_of_measurement',
'reference_values',
'date_of_test',
'result',
];

/**
* The attributes that are treated as Carbon.
*
* @var array
*/
protected $dates = [
'date_of_test',
];

/**
* The attributes excluded from the model's JSON form.
*
* @var array
*/
protected $hidden = [
'patient_id',
];

/**
* Do something in date_of_test attribute after getting
    it
*
* @param Carbon $date
* @return array
*/
public function getDateOfTestAttribute($date){
$array = [Carbon::parse($date)];
if ($date) {
$array[] = Carbon::parse($date)->format('F_d,_Y');
} else {
$array[] = null;
}
return $array;
}

/**
* Do something in date_of_start attribute before
    setting it
*
* @param Carbon $date
*/
public function setDateOfTestAttribute($date){
if ($date)
$this->attributes['date_of_test'] = Carbon::parse($date
    );
else
$this->attributes['date_of_test'] = null;
}

/**
* A laboratory result belongs to exactly one daily
    record.
*
* @return \Illuminate\Database\Eloquent\Relations\
    BelongsTo
*/
public function patient(){
return $this->belongsTo('App\Models\Patient');
}

/**
* A laboratory result can have exactly one file.
*
* @return \Illuminate\Database\Eloquent\Relations\
    HasOne
*/
public function file(){
return $this->hasOne('App\Models\File');
}

}
```

Local Admin Model

```
<?php namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class LocalAdmin extends Model {

/**
* The database table used by the model.
*
* @var string
*/
protected $table = 'local_admins';

/**
* The attributes that are mass assignable.
*
* @var array
```

```
*/
protected $fillable = [];

/**
* The attributes that are treated as Carbon.
*
* @var array
*/
protected $dates = [];

/**
* The attributes excluded from the model's JSON form.
*
* @var array
*/
protected $hidden = [
'user_id',
'hospital_id',
];

/**
* A local admin account belongs to exactly one user.
*
* @return \Illuminate\Database\Eloquent\Relations\
    BelongsTo
*/
public function user(){
return $this->belongsTo('App\User');
}

/**
* A local admin belongs to exactly one hospital.
*
* @return \Illuminate\Database\Eloquent\Relations\
    BelongsTo
*/
public function hospital(){
return $this->belongsTo('App\Models\Hospital');
}

}
```

Medical Condition Model

```
<?php namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class MedicalCondition extends Model {

/**
* The database table used by the model.
*
* @var string
*/
protected $table = 'medical_conditions';

/**
* The attributes that are mass assignable.
*
* @var array
*/
protected $fillable = [
'patient_id',
'daily_medical_problem',
'describe_the_symptom',
'medicine_taken',
'effects_noted',
'any_maneuver_taken',
];

/**
* The attributes that are treated as Carbon.
*
* @var array
*/
protected $dates = [];

/**
* The attributes excluded from the model's JSON form.
*
* @var array
*/
protected $hidden = [];

/**
* A medical condition belongs to exactly one patient.
*
* @return \Illuminate\Database\Eloquent\Relations\
    BelongsTo
*/
public function patient(){
return $this->belongsTo('App\Models\Patient');
}

/**
* A medical condition can have a follow-up visit.
*
* @return \Illuminate\Database\Eloquent\Relations\
    HasOne
*/
public function followUpVisit(){
return $this->hasOne('App\Models\FollowUpVisit');
}

/**
* A medical condition can have a decision.
*
* @return \Illuminate\Database\Eloquent\Relations\
    HasOne
*/
public function decisionOnCondition(){
return $this->hasOne('App\Models\DecisionOnCondition');
}

}
```

Medical Professional Model

```
<?php namespace App\Models;
```

```
use Illuminate\Database\Eloquent\Model;

class MedicalProfessional extends Model {

/**
* The database table used by the model.
*
* @var string
*/
protected $table = 'medical_professionals';

/**
* The attributes that are mass assignable.
*
* @var array
*/
protected $fillable = [
'specialties',
'clinic_schedule',
];

/**
* The attributes that are treated as Carbon.
*
* @var array
*/
protected $dates = [];

/**
* The attributes excluded from the model's JSON form.
*
* @var array
*/
protected $hidden = [
'user_id',
];

/**
* A medical professional account belongs to exactly one
      user.
*
* @return \Illuminate\Database\Eloquent\Relations\
     BelongsTo
*/
public function user(){
return $this->belongsTo('App\User');
}

/**
* A medical professional can have many patients.
*
* @return \Illuminate\Database\Eloquent\Relations\
     HasMany
*/
public function patients(){
return $this->hasMany('App\Models\Patient');
}

/**
* A medical professional can have many medical
     professions.
*
* @return \Illuminate\Database\Eloquent\Relations\
     HasMany
*/
public function hospitalMedicalProfessions(){
return $this->hasMany('App\Models\
     HospitalMedicalProfession');
}

/**
* A medical professional can have many decision.
*
* @return \Illuminate\Database\Eloquent\Relations\
     HasMany
*/
public function decisionOnCondition(){
return $this->hasMany('App\Models\DecisionOnCondition')
     ;
}

}
```

Medical Record Attribute Model

```
<?php namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class MedicalRecordAttribute extends Model {

/**
* The database table used by the model.
*
* @var string
*/
protected $table = 'medical_record_attributes';

/**
* The attributes that are mass assignable.
*
* @var array
*/
protected $fillable = [
'value',
'medication_record_id',
'medical_record_attribute_type_id'
];

/**
* The attributes that are treated as Carbon.
*
* @var array
*/
protected $dates = [];

/**
* The attributes excluded from the model's JSON form.
*
```

```
* @var array
*/
protected $hidden = [
'medical_record_attribute_type_id'
];

/**
* A medical record attribute belongs to exactly one
    medication record.
*
* @return \Illuminate\Database\Eloquent\Relations\
    BelongsTo
*/
public function medicationRecord(){
return $this->belongsTo('App\Models\MedicationRecord');
}

/**
* A medical record attribute belongs to exactly one
    medical record attribute type.
*
* @return \Illuminate\Database\Eloquent\Relations\
    BelongsTo
*/
public function medicalRecordAttributeType(){
return $this->belongsTo('App\Models\
    MedicalRecordAttributeType');
}

}
```

Medical Record Attribute Type Model

```
<?php namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class MedicalRecordAttributeType extends Model {

/**
* The database table used by the model.
*
* @var string
*/
protected $table = 'medical_record_attribute_types';

/**
* The attributes that are mass assignable.
*
* @var array
*/
protected $fillable = [
'type',
];

/**
* The attributes that are treated as Carbon.
*
* @var array
*/
protected $dates = [];

/**
* The attributes excluded from the model's JSON form.
*
* @var array
*/
protected $hidden = [];

/**
* A medical record attribute type has one or more
    medical record attribute/s.
*
* @return \Illuminate\Database\Eloquent\Relations\
    HasMany
*/
public function medicalRecordAttributes(){
return $this->hasMany('App\Models\
    MedicalRecordAttribute');
}

}
```

Medical Resource Model

```
<?php namespace App\Models;

use Carbon\Carbon;
use Illuminate\Database\Eloquent\Model;

class MedicalResource extends Model {

/**
* The database table used by the model.
*
* @var string
*/
protected $table = 'medical_resources';

/**
* The attributes that are mass assignable.
*
* @var array
*/
protected $fillable = [
'title',
'description',
'authors',
'publisher',
'date_published',
];

/**
* The attributes that are treated as Carbon.
*
* @var array
*/
protected $dates = [
'date_published',
];
```

```
/**
* The attributes excluded from the model's JSON form.
*
* @var array
*/
protected $hidden = [
'user_id',
'forum_category_id',
];

/**
* Do something in date_published attribute after
    getting it.
*
* @param Carbon $date
* @return array
*/
public function getDatePublishedAttribute($date){
$array = [Carbon::parse($date)];
if ($date) {
$array[] = Carbon::parse($date)->format('F_d,_Y');
} else {
$array[] = null;
}
return $array;
}

/**
* Do something in date_published attribute before
    setting it.
*
* @param Carbon $date
*/
public function setDatePublishedAttribute($date){
if ($date)
$this->attributes['date_published'] = Carbon::parse(
    $date);
else
$this->attributes['date_published'] = null;
}

/**
* A medical resource belongs to exactly one user.
*
* @return \Illuminate\Database\Eloquent\Relations\
    BelongsTo
*/
public function user(){
return $this->belongsTo('App\User');
}

/**
* A medical resource belongs to exactly one forum
    category.
*
* @return \Illuminate\Database\Eloquent\Relations\
    BelongsTo
*/
public function forumCategory(){
return $this->belongsTo('App\Models\ForumCategory');
}

/**
* A medical resource can have exactly one file.
*
* @return \Illuminate\Database\Eloquent\Relations\
    HasOne
*/
public function file(){
return $this->hasOne('App\Models\File');
}

}
```

Medication Record Model

```
<?php namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class MedicationRecord extends Model {

/**
* The database table used by the model.
*
* @var string
*/
protected $table = 'medication_records';

/**
* The attributes that are mass assignable.
*
* @var array
*/
protected $fillable = [
'patient_id',
];

/**
* The attributes that are treated as Carbon.
*
* @var array
*/
protected $dates = [];

/**
* The attributes excluded from the model's JSON form.
*
* @var array
*/
protected $hidden = [
'patient_id',
];

/**
* A medication record belongs to exactly one patient.
*
* @return \Illuminate\Database\Eloquent\Relations\
    BelongsTo
```

```
*/
public function patient(){
return $this->belongsTo('App\Models\Patient');
}

/**
* A medication record has many medical record
    attributes.
*
* @return \Illuminate\Database\Eloquent\Relations\
    HasMany
*/
public function medicalRecordAttributes(){
return $this->hasMany('App\Models\
    MedicalRecordAttribute');
}

}
```

Medication Regimen Model

```
<?php namespace App\Models;

use Carbon\Carbon;
use Illuminate\Database\Eloquent\Model;

class MedicationRegimen extends Model {

/**
* The database table used by the model.
*
* @var string
*/
protected $table = 'medication_regimens';

/**
* The attributes that are mass assignable.
*
* @var array
*/
protected $fillable = [
'patient_id',
'medication',
'dose',
'frequency',
'time_of_medication',
'comments',
];

/**
* The attributes that are treated as Carbon.
*
* @var array
*/
protected $dates = [];

/**
* The attributes excluded from the model's JSON form.
*
* @var array
*/
protected $hidden = [
'patient_id',
];

/**
* A medication regimen belongs to exactly one patient.
*
* @return \Illuminate\Database\Eloquent\Relations\
    BelongsTo
*/
public function patient(){
return $this->belongsTo('App\Models\Patient');
}

}
```

Notification Model

```
<?php namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Notification extends Model {

/**
* The database table used by the model.
*
* @var string
*/
protected $table = 'notifications';

/**
* The attributes that are mass assignable.
*
* @var array
*/
protected $fillable = [
'user_id',
'description',
'link',
'is_seen',
];

/**
* The attributes that are treated as Carbon.
*
* @var array
*/
protected $dates = [];

/**
* The attributes excluded from the model's JSON form.
*
* @var array
*/
protected $hidden = [
'user_id',
];
```

```
/**
* A notification belongs to exactly one user.
*
* @return \Illuminate\Database\Eloquent\Relations\
    BelongsTo
*/
public function user(){
return $this->belongsTo('App\User');
}

}
```

Nutritionist Model

```
<?php namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Nutritionist extends Model {

/**
* The database table used by the model.
*
* @var string
*/
protected $table = 'nutritionists';

/**
* The attributes that are mass assignable.
*
* @var array
*/
protected $fillable = [
'clinic_schedule',
];

/**
* The attributes that are treated as Carbon.
*
* @var array
*/
protected $dates = [];

/**
* The attributes excluded from the model's JSON form.
*
* @var array
*/
protected $hidden = [
'user_id',
];

/**
* A nutritionist account belongs to exactly one user.
*
* @return \Illuminate\Database\Eloquent\Relations\
    BelongsTo
*/
public function user(){
return $this->belongsTo('App\User');
}

/**
* A nutritionist can have many patients.
*
* @return \Illuminate\Database\Eloquent\Relations\
    HasMany
*/
public function patients(){
return $this->hasMany('App\Models\Patient');
}

/**
* A nutritionist can have many nutritionist professions
    .
*
* @return \Illuminate\Database\Eloquent\Relations\
    HasMany
*/
public function hospitalNutritionistProfessions(){
return $this->hasMany('App\Models\
    HospitalNutritionistProfession');
}

}
```

Nutritionist Record Model

```
<?php namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class NutritionistRecord extends Model {

/**
* The database table used by the model.
*
* @var string
*/
protected $table = 'nutritionist_records';

/**
* The attributes that are mass assignable.
*
* @var array
*/
protected $fillable = [
'patient_id',
];

/**
* The attributes that are treated as Carbon.
*
* @var array
*/
protected $dates = [];

/**
* The attributes excluded from the model's JSON form.
*
```

```
* @var array
*/
protected $hidden = [
'patient_id',
];

/**
* A nutritionist record belongs to exactly one patient.
*
* @return \Illuminate\Database\Eloquent\Relations\
    BelongsTo
*/
public function patient(){
return $this->belongsTo('App\Models\Patient');
}

/**
* A nutritionist record has many nutritionist record
    attributes.
*
* @return \Illuminate\Database\Eloquent\Relations\
    HasMany
*/
public function nutritionistRecordAttributes(){
return $this->hasMany('App\Models\
    NutritionistRecordAttribute');
}

}
```

Nutritionist Record Attribute Model

```
<?php namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class NutritionistRecordAttribute extends Model {

/**
* The database table used by the model.
*
* @var string
*/
protected $table = 'nutritionist_record_attributes';

/**
* The attributes that are mass assignable.
*
* @var array
*/
protected $fillable = [
'value',
'nutritionist_record_id',
'nutritionist_record_attribute_type_id'
];

/**
* The attributes that are treated as Carbon.
*
* @var array
*/
protected $dates = [];

/**
* The attributes excluded from the model's JSON form.
*
* @var array
*/
protected $hidden = [
'nutritionist_record_attribute_type_id'
];

/**
* A nutritionist record attribute belongs to exactly
    one nutritionist record.
*
* @return \Illuminate\Database\Eloquent\Relations\
    BelongsTo
*/
public function nutritionistRecord(){
return $this->belongsTo('App\Models\NutritionistRecord'
    );
}

/**
* A nutritionist record attribute belongs to exactly
    one nutritionist record attribute type.
*
* @return \Illuminate\Database\Eloquent\Relations\
    BelongsTo
*/
public function nutritionistRecordAttributeType(){
return $this->belongsTo('App\Models\
    NutritionistRecordAttributeType');
}

}
```

Nutritionist Record Attribute Type Model

```
<?php namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class NutritionistRecordAttributeType extends Model {

/**
* The database table used by the model.
*
* @var string
*/
protected $table = 'nutritionist_record_attribute_types
    ';

/**
* The attributes that are mass assignable.
*
* @var array
*/
protected $fillable = [
```

```
'type',
];

/**
* The attributes that are treated as Carbon.
*
* @var array
*/
protected $dates = [];

/**
* The attributes excluded from the model's JSON form.
*
* @var array
*/
protected $hidden = [];

/**
* A nutritionist record attribute type has one or more
    nutritionist record attribute/s.
*
* @return \Illuminate\Database\Eloquent\Relations\
    HasMany
*/
public function nutritionistRecordAttributes(){
return $this->hasMany('App\Models\
    NutritionistRecordAttribute');
}

}
```

Nutritionist Regimen Model

```
<?php namespace App\Models;

use Carbon\Carbon;
use Illuminate\Database\Eloquent\Model;

class NutritionistRegimen extends Model {

/**
* The database table used by the model.
*
* @var string
*/
protected $table = 'nutritionist_regimens';

/**
* The attributes that are mass assignable.
*
* @var array
*/
protected $fillable = [
'patient_id',
'breakfast',
'lunch',
'dinner',
'am_snack',
'pm_snack',
];

/**
* The attributes that are treated as Carbon.
*
* @var array
*/
protected $dates = [];

/**
* The attributes excluded from the model's JSON form.
*
* @var array
*/
protected $hidden = [
'patient_id',
];

/**
* A nutritionist regimen belongs to exactly one patient
    .
*
* @return \Illuminate\Database\Eloquent\Relations\
    BelongsTo
*/
public function patient(){
return $this->belongsTo('App\Models\Patient');
}

}
```

Patient Model

```
<?php namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Patient extends Model {

/**
* The database table used by the model.
*
* @var string
*/
protected $table = 'patients';

/**
* The attributes that are mass assignable.
*
* @var array
*/
protected $fillable = [
'hospital_id_number',
'caregiver_first_name',
'caregiver_middle_name',
'caregiver_last_name'
];

/**
* The attributes that are treated as Carbon.
*
```

```
* @var array
*/
protected $dates = [];

/**
* The attributes excluded from the model's JSON form.
*
* @var array
*/
protected $hidden = [
'user_id',
'hospital_id',
'medical_professional_id',
'nutritionist_id',
'physiatrist_id',
'hospital_id_number'
];

/**
* A patient or caregiver account belongs to exactly one
      user.
*
* @return \Illuminate\Database\Eloquent\Relations\
     BelongsTo
*/
public function user(){
return $this->belongsTo('App\User');
}

/**
* A patient or caregiver belongs to exactly one
     hospital.
*
* @return \Illuminate\Database\Eloquent\Relations\
     BelongsTo
*/
public function hospital(){
return $this->belongsTo('App\Models\Hospital');
}

/**
* A patient or caregiver belongs to exactly one medical
      professional.
*
* @return \Illuminate\Database\Eloquent\Relations\
     BelongsTo
*/
public function medicalProfessional(){
return $this->belongsTo('App\Models\MedicalProfessional
     ');
}

/**
* A patient or caregiver belongs to exactly one
     nutritionist.
*
* @return \Illuminate\Database\Eloquent\Relations\
     BelongsTo
*/
public function nutritionist(){
return $this->belongsTo('App\Models\Nutritionist');
}

/**
* A patient or caregiver belongs to exactly one
     physiatrist.
*
* @return \Illuminate\Database\Eloquent\Relations\
     BelongsTo
*/
public function physiatrist(){
return $this->belongsTo('App\Models\Physiatrist');
}

/**
* A patient can have many medical conditions.
*
* @return \Illuminate\Database\Eloquent\Relations\
     HasMany
*/
public function medicalConditions(){
return $this->hasMany('App\Models\MedicalCondition');
}

/**
* A patient can have many patient attributes.
*
* @return \Illuminate\Database\Eloquent\Relations\
     HasMany
*/
public function patientAttributes(){
return $this->hasMany('App\Models\PatientAttribute');
}

/**
* A patient can have many medical record attributes.
*
* @return \Illuminate\Database\Eloquent\Relations\
     HasMany
*/
public function medicationRecords(){
return $this->hasMany('App\Models\MedicationRecord');
}

/**
* A patient can have many medical record attributs.
*
* @return \Illuminate\Database\Eloquent\Relations\
     HasMany
*/
public function nutritionistRecords(){
return $this->hasMany('App\Models\NutritionistRecord');
}

/**
* A patient can have many medical record attributs.
*
* @return \Illuminate\Database\Eloquent\Relations\
     HasMany
*/
```

```
public function physiatristRecords(){
return $this->hasMany('App\Models\PhysiatristRecord');
}

/**
* A patient can have many laboratory results.
*
* @return \Illuminate\Database\Eloquent\Relations\
    HasMany
*/
public function laboratoryResults(){
return $this->hasMany('App\Models\LaboratoryResult');
}

/**
* A patient can have many medication regimens.
*
* @return \Illuminate\Database\Eloquent\Relations\
    HasMany
*/
public function medicationRegimens(){
return $this->hasMany('App\Models\MedicationRegimen');
}

/**
* A patient can have many nutritionist regimens.
*
* @return \Illuminate\Database\Eloquent\Relations\
    HasMany
*/
public function nutritionistRegimens(){
return $this->hasMany('App\Models\NutritionistRegimen')
    ;
}

/**
* A patient can have many physiatrist regimens.
*
* @return \Illuminate\Database\Eloquent\Relations\
    HasMany
*/
public function physiatristRegimens(){
return $this->hasMany('App\Models\PhysiatristRegimen');
}

/**
* A patient can have many follow-up visits.
*
* @return \Illuminate\Database\Eloquent\Relations\
    HasMany
*/
public function followUpVisits(){
return $this->hasMany('App\Models\FollowUpVisits');
}

}
```

Patient Attribute Model

```
<?php namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class PatientAttribute extends Model {

/**
* The database table used by the model.
*
* @var string
*/
protected $table = 'patient_attributes';

/**
* The attributes that are mass assignable.
*
* @var array
*/
protected $fillable = [
'value',
'patient_id',
'patient_attribute_type_id'
];

/**
* The attributes that are treated as Carbon.
*
* @var array
*/
protected $dates = [];

/**
* The attributes excluded from the model's JSON form.
*
* @var array
*/
protected $hidden = [
'patient_id',
'patient_attribute_type_id',
];

/**
* A patient attribute belongs to exactly one patient.
*
* @return \Illuminate\Database\Eloquent\Relations\
    BelongsTo
*/
public function patient(){
return $this->belongsTo('App\Models\Patient');
}

/**
* A patient attribute belongs to exactly one patient
    attribute type.
*
* @return \Illuminate\Database\Eloquent\Relations\
    BelongsTo
*/
public function patientAttributeType(){
return $this->belongsTo('App\Models\
    PatientAttributeType');
```

```
}

}
```

Patient Attribute Type Model

```
<?php namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class PatientAttributeType extends Model {

/**
* The database table used by the model.
*
* @var string
*/
protected $table = 'patient_attribute_types';

/**
* The attributes that are mass assignable.
*
* @var array
*/
protected $fillable = [
'type',
];

/**
* The attributes that are treated as Carbon.
*
* @var array
*/
protected $dates = [];

/**
* The attributes excluded from the model's JSON form.
*
* @var array
*/
protected $hidden = [];

/**
* A patient attribute type has one or many patient
    attribute/s.
*
* @return \Illuminate\Database\Eloquent\Relations\
    HasMany
*/
public function patientAttributes(){
return $this->hasMany('App\Models\PatientAttribute');
}

}
```

Physiatrist Model

```
<?php namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Physiatrist extends Model {

/**
* The database table used by the model.
*
* @var string
*/
protected $table = 'physiatrists';

/**
* The attributes that are mass assignable.
*
* @var array
*/
protected $fillable = [
'clinic_schedule',
];

/**
* The attributes that are treated as Carbon.
*
* @var array
*/
protected $dates = [];

/**
* The attributes excluded from the model's JSON form.
*
* @var array
*/
protected $hidden = [
'user_id',
];

/**
* A physiatrist account belongs to exactly one user.
*
* @return \Illuminate\Database\Eloquent\Relations\
    BelongsTo
*/
public function user(){
return $this->belongsTo('App\User');
}

/**
* A physiatrist can have many patients.
*
* @return \Illuminate\Database\Eloquent\Relations\
    HasMany
*/
public function patients(){
return $this->hasMany('App\Models\Patient');
}

/**
* A physiatrist can have many physiatrist professions.
*
* @return \Illuminate\Database\Eloquent\Relations\
    HasMany
```

```
*/
public function hospitalPhysiatristProfessions(){
return $this->hasMany('App\Models\
    HospitalPhysiatristProfession');
}

}
```

Physiatrist Record Model

```
<?php namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class PhysiatristRecord extends Model {

/**
* The database table used by the model.
*
* @var string
*/
protected $table = 'physiatrist_records';

/**
* The attributes that are mass assignable.
*
* @var array
*/
protected $fillable = [
'patient_id',
];

/**
* The attributes that are treated as Carbon.
*
* @var array
*/
protected $dates = [
];

/**
* The attributes excluded from the model's JSON form.
*
* @var array
*/
protected $hidden = [
'patient_id',
];

/**
* A physiatrist record belongs to exactly one patient.
*
* @return \Illuminate\Database\Eloquent\Relations\
    BelongsTo
*/
public function patient(){
return $this->belongsTo('App\Models\Patient');
}

/**
* A physiatrist record has many physiatrist record
    attributes.
*
* @return \Illuminate\Database\Eloquent\Relations\
    HasMany
*/
public function physiatristRecordAttributes(){
return $this->hasMany('App\Models\
    PhysiatristRecordAttribute');
}

}
```

Physiatrist Record Attribute Model

```
<?php namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class PhysiatristRecordAttribute extends Model {

/**
* The database table used by the model.
*
* @var string
*/
protected $table = 'physiatrist_record_attributes';

/**
* The attributes that are mass assignable.
*
* @var array
*/
protected $fillable = [
'value',
'physiatrist_record_id',
'physiatrist_record_attribute_type_id'
];

/**
* The attributes that are treated as Carbon.
*
* @var array
*/
protected $dates = [];

/**
* The attributes excluded from the model's JSON form.
*
* @var array
*/
protected $hidden = [
'physiatrist_record_attribute_type_id'
];

/**
* A physiatrist record attribute belongs to exactly one
    physiatrist record.
*
* @return \Illuminate\Database\Eloquent\Relations\
```

```
     BelongsTo
*/
public function physiatristRecord(){
return $this->belongsTo('App\Models\PhysiatristRecord')
     ;
}

/**
* A physiatrist record attribute belongs to exactly one
      physiatrist record attribute type.
*
* @return \Illuminate\Database\Eloquent\Relations\
     BelongsTo
*/
public function physiatristRecordAttributeType(){
return $this->belongsTo('App\Models\
     PhysiatristRecordAttributeType');
}

}
```

Physiatrist Record Attribute Type Model

```
<?php namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class PhysiatristRecordAttributeType extends Model {

/**
* The database table used by the model.
*
* @var string
*/
protected $table = 'physiatrist_record_attribute_types'
     ;

/**
* The attributes that are mass assignable.
*
* @var array
*/
protected $fillable = [
'type',
];

/**
* The attributes that are treated as Carbon.
*
* @var array
*/
protected $dates = [];

/**
* The attributes excluded from the model's JSON form.
*
* @var array
*/
protected $hidden = [];

/**
* A physiatrist record attribute type has one or more
     physiatrist record attribute/s.
*
* @return \Illuminate\Database\Eloquent\Relations\
     HasMany
*/
public function physiatristRecordAttributes(){
return $this->hasMany('App\Models\
     PhysiatristRecordAttribute');
}

}
```

Physiatrist Regimen Model

```
<?php namespace App\Models;

use Carbon\Carbon;
use Illuminate\Database\Eloquent\Model;

class PhysiatristRegimen extends Model {

/**
* The database table used by the model.
*
* @var string
*/
protected $table = 'physiatrist_regimens';

/**
* The attributes that are mass assignable.
*
* @var array
*/
protected $fillable = [
'patient_id',
'type_of_exercise',
'frequency',
'duration',
'time_of_exercise',
'comments',
];

/**
* The attributes that are treated as Carbon.
*
* @var array
*/
protected $dates = [];

/**
* The attributes excluded from the model's JSON form.
*
* @var array
*/
protected $hidden = [
'patient_id',
];
```

```
/**
* A physiatrist regimen belongs to exactly one patient.
*
* @return \Illuminate\Database\Eloquent\Relations\
    BelongsTo
*/
public function patient(){
return $this->belongsTo('App\Models\Patient');
}

}
```

Supervisor Model

```
<?php namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Supervisor extends Model {

/**
* The database table used by the model.
*
* @var string
*/
protected $table = 'supervisors';

/**
* The attributes that are mass assignable.
*
* @var array
*/
protected $fillable = [];

/**
* The attributes that are treated as Carbon.
*
* @var array
*/
protected $dates = [];

/**
* The attributes excluded from the model's JSON form.
*
* @var array
*/
protected $hidden = [
'user_id',
'forum_category_id',
];

/**
* A supervisor account belongs to exactly one user.
*
* @return \Illuminate\Database\Eloquent\Relations\
    BelongsTo
*/
public function user(){
return $this->belongsTo('App\User');
}

/**
* A supervisor belongs to exactly one forum category.
*
* @return \Illuminate\Database\Eloquent\Relations\
    BelongsTo
*/
public function forumCategory(){
return $this->belongsTo('App\Models\ForumCategory');
}

}
```

System Admin Model

```
<?php namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class SystemAdmin extends Model {

/**
* The database table used by the model.
*
* @var string
*/
protected $table = 'system_admins';

/**
* The attributes that are mass assignable.
*
* @var array
*/
protected $fillable = [];

/**
* The attributes that are treated as Carbon.
*
* @var array
*/
protected $dates = [];

/**
* The attributes excluded from the model's JSON form.
*
* @var array
*/
protected $hidden = [
'user_id',
];

/**
* A system admin account belongs to exactly one user.
*
* @return \Illuminate\Database\Eloquent\Relations\
    BelongsTo
*/
public function user(){
return $this->belongsTo('App\User');
}
```

```
}
```

Medical Record Retriever

```
<?php

namespace App\Services\Retrievers;

use App\Models\DailyRecord;
use App\Models\MedicalRecordAttribute;
use App\Models\MedicalRecordAttributeType;
use App\Models\NutritionistRecordAttribute;
use App\Models\NutritionistRecordAttributeType;
use App\Models\PhysiatristRecordAttribute;
use App\Models\PhysiatristRecordAttributeType;

class MedicalRecordRetriever
{
/*
|--------------------------------------------------------------------------

| Medical Record Retriever Service
|--------------------------------------------------------------------------

|
| This service class handles the retrieval of the
    stroke and medication record,
| nutrition record, rehabilitation medicine record, and
     laboratory results
| of the specific daily medical record of a given
    patient.
|
*/

/**
* Constructor of the class.
*
* @param \App\Services\Retrievers\
    SelectFieldListsRetriever
    $selectFieldListsRetriever
*/
public function __construct(SelectFieldListsRetriever
    $selectFieldListsRetriever) {
$this->selectFieldListsRetriever =
    $selectFieldListsRetriever;
}

/**
* Get the stroke and medication record attributes.
*
* @param int $medication_record_id
* @return array
*/
public function getMedicationRecord(
    $medication_record_id) {
$medical_record_attribute_types =
    MedicalRecordAttributeType::lists('type', 'id');
$medical_record_attributes = MedicalRecordAttribute::
    where('medication_record_id',
    $medication_record_id)->get();
$array_attributes = [];
$attr = [];

// loop through the existing attributes of the medical
    record
foreach ($medical_record_attributes as
    $medical_record_attribute) {
$type = $medical_record_attribute->
    medicalRecordAttributeType->type;
if (!isset($attr[$type])) { // if attribute is not set
    yet
if (in_array($type, $array_attributes)) // if attribute
     is supposed to be an array
$attr[$type] = [$medical_record_attribute->value];
else
$attr[$type] = $medical_record_attribute->value;
} else { // else, it is either a string or an array
if (is_string($attr[$type])) { // if attribute is a
    string
$temp = $attr[$type];
$attr[$type] = array($temp);
}
$attr[$type][] = $medical_record_attribute->value;
}
}

// add other attributes (set to null) the medical
    record does not have
foreach ($medical_record_attribute_types as
    $medical_record_attribute_type) {
if (! isset($attr[$medical_record_attribute_type]))
$attr[$medical_record_attribute_type] = null;
}

if ($attr['latest_nihss_score'] == null)
$attr['latest_nihss_score'] = '0';

$attr = $this->setArrayMedicationRecordAttributes($attr
    );

return $attr;
}

/**
* Set up attributes to include all checkbox and radio
    fields in the stroke medication record form.
*
* @param array $attr
* @return array
*/
private function setArrayMedicationRecordAttributes(
    array $attr) {
$nihss = $this->selectFieldListsRetriever->getNIHSS();
$attributes_nihss = [];

// Set up the NIHSS attributes
foreach ($nihss as $nihss_item => $nihss_item_choices)
    {
$attributes_nihss[$nihss_item] = [$nihss_item_choices
```

```
    [0]]; // get display text
$attributes_nihss[$nihss_item][] = [];
foreach ($nihss_item_choices[1] as $choice) {
if ($attr[$nihss_item] == $choice) {
// if the specific choice is the answer in the
    attributes, the radio button is selected
$attributes_nihss[$nihss_item][1][$choice] = '1';
} else {
// else, the radio button is not selected
$attributes_nihss[$nihss_item][1][$choice] = '0';
}
}
$attr[$nihss_item] = $attributes_nihss[$nihss_item];
}

return $attr;
}

/**
* Get the nutritionist record attributes.
*
* @param int $nutritionist_record_id
* @return array
*/
public function getNutritionistRecord(
    $nutritionist_record_id) {
$nutritionist_record_attribute_types =
    NutritionistRecordAttributeType::lists('type', 'id
    ');
$nutritionist_record_attributes =
    NutritionistRecordAttribute::where('
    nutritionist_record_id', $nutritionist_record_id)
->get();
$array_attributes = [];
$attr = [];

// loop through the existing attributes of the
    nutrition record
foreach ($nutritionist_record_attributes as
    $nutritionist_record_attribute) {
$type = $nutritionist_record_attribute->
    nutritionistRecordAttributeType->type;
if (!isset($attr[$type])) { // if attribute is not set
    yet
if (in_array($type, $array_attributes)) // if attribute
     is supposed to be an array
$attr[$type] = [$nutritionist_record_attribute->value];
else
$attr[$type] = $nutritionist_record_attribute->value;
} else { // else, it is either a string or an array
if (is_string($attr[$type])) { // if attribute is a
    string
$temp = $attr[$type];
$attr[$type] = array($temp);
}
$attr[$type][] = $nutritionist_record_attribute->value;
}
}

// add other attributes (set to null) the medical
    record does not have
foreach ($nutritionist_record_attribute_types as
    $nutritionist_record_attribute_type) {
if (! isset($attr[$nutritionist_record_attribute_type])
    )
$attr[$nutritionist_record_attribute_type] = null;
}

return $attr;
}

/**
* Get the physiatrist record attributes.
*
* @param int $physiatrist_record_id
* @return array
*/
public function getPhysiatristRecord(
    $physiatrist_record_id) {
$physiatrist_record_attribute_types =
    PhysiatristRecordAttributeType::lists('type', 'id'
    );
$physiatrist_record_attributes =
    PhysiatristRecordAttribute::where('
    physiatrist_record_id', $physiatrist_record_id)->
    get();
$array_attributes = [];
$attr = [];

// loop through the existing attributes of the
    rehabilitation medicine record
foreach ($physiatrist_record_attributes as
    $physiatrist_record_attribute) {
$type = $physiatrist_record_attribute->
    physiatristRecordAttributeType->type;
if (!isset($attr[$type])) { // if attribute is not set
    yet
if (in_array($type, $array_attributes)) // if attribute
     is supposed to be an array
$attr[$type] = [$physiatrist_record_attribute->value];
else
$attr[$type] = $physiatrist_record_attribute->value;
} else { // else, it is either a string or an array
if (is_string($attr[$type])) { // if attribute is a
    string
$temp = $attr[$type];
$attr[$type] = array($temp);
}
$attr[$type][] = $physiatrist_record_attribute->value;
}
}

// add other attributes (set to null) the medical
    record does not have
foreach ($physiatrist_record_attribute_types as
    $physiatrist_record_attribute_type) {
if (! isset($attr[$physiatrist_record_attribute_type]))
$attr[$physiatrist_record_attribute_type] = null;
}
```

```
//$attr = $this->setArrayPhysiatristRecordAttributes(
    $attr);

return $attr;
}

}
```

Patient Retriever

```
<?php namespace App\Services\Retrievers;

use App\Models\HospitalMedicalProfession;
use App\Models\HospitalNutritionistProfession;
use App\Models\HospitalPhysiatristProfession;
use App\Models\MedicalProfessional;
use App\Models\Nutritionist;
use App\Models\Patient;
use App\Models\Physiatrist;

class PatientRetriever
{
/*
|--------------------------------------------------------------------------
| Patient Retriever Service
|--------------------------------------------------------------------------
|
| This service class handles the retrieval of the
    patients of a
| certain professional in all the hospitals where the
| professional is included.
|
*/

/**
* Get all of the patients of the medical professional
    under all hospitals he/she works at
*
* @param \App\Models\MedicalProfessional
    $medical_professional
* @return array
*/
public function getMedicalProfessionalPatients(
    MedicalProfessional $medical_professional = null)
    {
$overall_array = [];

if ($medical_professional) {
$hospital_medical_professions =
    HospitalMedicalProfession::where('
    medical_professional_id',
$medical_professional->id)->where('is_approved', true)
    ->get();
$medical_professional_patients = Patient::where('
    medical_professional_id', $medical_professional->
    id);

foreach ($hospital_medical_professions as
    $hospital_medical_profession) {
$hospital_patients = $medical_professional_patients->
    where('hospital_id',
$hospital_medical_profession->hospital->id)->get();
$hospital_patient_array = [];
foreach ($hospital_patients as $hospital_patient) {
$hospital_patient_array[] = [
'id' => $hospital_patient->id,
'hospital_id_number' => $hospital_patient->
    hospital_id_number,
'username' => $hospital_patient->user->username,
'full_name' => $hospital_patient->user->fullName2()
];
}
$overall_array[] = [
'id' => $hospital_medical_profession->id,
'hospital_name' => $hospital_medical_profession->
    hospital->name,
'patients' => $hospital_patient_array
];
}
}

return $overall_array;
}

/**
* Get all of the patients of the nutritionist under all
     hospitals he/she works at
*
* @param \App\Models\Nutritionist $nutritionist
* @return array
*/
public function getNutritionistPatients(Nutritionist
    $nutritionist = null) {
$overall_array = [];

if ($nutritionist) {
$hospital_nutritionist_professions =
    HospitalNutritionistProfession::where('
    nutritionist_id',
$nutritionist->id)->where('is_approved', true)->get();
$nutritionist_patients = Patient::where('
    nutritionist_id', $nutritionist->id);

foreach ($hospital_nutritionist_professions as
    $hospital_nutritionist_profession) {
$hospital_patients = $nutritionist_patients->where('
    hospital_id',
$hospital_nutritionist_profession->hospital->id)->get()
    ;
$hospital_patient_array = [];
foreach ($hospital_patients as $hospital_patient) {
$hospital_patient_array[] = [
'id' => $hospital_patient->id,
'hospital_id_number' => $hospital_patient->
    hospital_id_number,
'username' => $hospital_patient->user->username,
'full_name' => $hospital_patient->user->fullName2()
```

```
];
}
$overall_array[] = [
'id' => $hospital_nutritionist_profession->id,
'hospital_name' => $hospital_nutritionist_profession->
    hospital->name,
'patients' => $hospital_patient_array
];
}
}

return $overall_array;
}

/**
* Get all of the patients of the physiatrist under all
    hospitals he/she works at
*
* @param \App\Models\Physiatrist $physiatrist
* @return array
*/
public function getPhysiatristPatients(Physiatrist
    $physiatrist = null) {
$overall_array = [];

if ($physiatrist) {
$hospital_physiatrist_professions =
    HospitalPhysiatristProfession::where('
    physiatrist_id',
$physiatrist->id)->where('is_approved', true)->get();
$physiatrist_patients = Patient::where('physiatrist_id'
    , $physiatrist->id);

foreach ($hospital_physiatrist_professions as
    $hospital_physiatrist_profession) {
$hospital_patients = $physiatrist_patients->where('
    hospital_id',
$hospital_physiatrist_profession->hospital->id)->get();
$hospital_patient_array = [];
foreach ($hospital_patients as $hospital_patient) {
$hospital_patient_array[] = [
'id' => $hospital_patient->id,
'hospital_id_number' => $hospital_patient->
    hospital_id_number,
'username' => $hospital_patient->user->username,
'full_name' => $hospital_patient->user->fullName2()
];
}
$overall_array[] = [
'id' => $hospital_physiatrist_profession->id,
'hospital_name' => $hospital_physiatrist_profession->
    hospital->name,
'patients' => $hospital_patient_array
];
}
}

return $overall_array;
}
}
```

Select Field Lists Retriever

```
<?php namespace App\Services\Retrievers;

class SelectFieldListsRetriever
{

/**
* Get decisions.
*
* @return \Illuminate\Http\Response
*/
public function getDecisions() {
$decisions = [
'continue_regimen' => 'Advise to continue the
    medication regimen',
'follow_up_visit' => 'Advise a follow-up visit',
'immediate_hospitalization' => 'Recommend immediate
    hospitalization',
];
return $decisions;
}

/**
* Get the list of civil statuses.
*
* @return array
*/
public function getCivilStatuses() {
$civil_statuses = [
'single' => 'Single',
'widowed' => 'Widowed',
'co-habitation' => 'Co-Habitation',
'married' => 'Married',
'annulled' => 'Annulled',
'divorced' => 'Divorced',
];
return $civil_statuses;
}

/**
* Get the list of highest educational attainments.
*
* @return array
*/
public function getHighestEducationalAttainments() {
$highest_educational_attainments = [
'no_grade_completed' => 'No grade completed',
'preschool' => 'Preschool',
'elementary_undergraduate' => 'Elementary Undergraduate
    ',
'elementary_graduate' => 'Elementary Graduate',
'high_school_undergraduate' => 'High School
    Undergraduate',
'high_school_graduate' => 'High School Graduate',
'post_secondary_undergraduate' => 'Post Secondary
    Undergraduate',
'post_secondary_graduate' => 'Post Secondary Graduate',
'college_undergraduate' => 'College Undergraduate',
'college_graduate' => 'College Graduate',
'post_baccalaureate' => 'Post Baccalaureate',
```

```
];
return $highest_educational_attainments;
}

/**
* Get the list of types of stroke.
*
* @return array
*/
public function getTypesOfStroke() {
$types_of_stroke = [
'ischemic_stroke' => 'Ischemic Stroke',
'hemorrhagic_stroke' => 'Hemorrhagic Stroke',
];
return $types_of_stroke;
}

/**
* Get the National Institutes of Health Stroke Scale.
*
* @return array
*/
public function getNIHSS() {
$nihss = [
'level_of_consciousness' => [
'1.a. Level of Consciousness', ['0', '1', '2', '3'],
],
'loc_questions' => [
'1.b. LOC Questions', ['0', '1', '2'],
],
'loc_commands' => [
'1.c. LOC Commands', ['0', '1', '2'],
],
'best_gaze' => [
'2. Best Gaze', ['0', '1', '2'],
],
'visual' => [
'3. Visual', ['0', '1', '2', '3'],
],
'facial_palsy' => [
'4. Facial Palsy', ['0', '1', '2', '3'],
],
'motor_arm_left' => [
'5.a. Motor Arm: Left Arm', ['0', '1', '2', '3', '4', '
    UN'],
],
'motor_arm_right' => [
'5.b. Motor Arm: Right Arm', ['0', '1', '2', '3', '4',
    'UN'],
],
'motor_leg_left' => [
'6.a. Motor Leg: Left Leg', ['0', '1', '2', '3', '4', '
    UN'],
],
'motor_leg_right' => [
'6.b. Motor Leg: Right Leg', ['0', '1', '2', '3', '4',
    'UN'],
],
'limb_ataxia' => [
'7. Limb Ataxia', ['0', '1', '2', 'UN'],
],
'sensory' => [
'8. Sensory', ['0', '1', '2'],
],
'best_language' => [
'9. Best Language', ['0', '1', '2', '3'],
],
'dysarthia' => [
'10. Dysarthia', ['0', '1', '2', 'UN'],
],
'extinction_and_inattention' => [
'11. Extinction and Inattention (formerly Neglect)', ['
    0', '1', '2'],
],
];
return $nihss;
}

/**
* Get the list of diseases or attacks relevant to
    stroke.
*
* @return array
*/
public function getDiseasesAndAttacks() {
$diseasesAndAttacks = [
'hypertension' => 'Hypertension',
'diabetes_mellitus' => 'Diabetes Mellitus',
'dyslipidemia' => 'Dyslipidemia',
'intracranial_artery_stenosis' => 'Intracranial Artery
    Stenosis',
'extracranial_artery_stenosis' => 'Extracranial Artery
    Stenosis',
'ischemic_heart_disease' => 'Ischemic Heart Disease',
'valvular_heart_disease' => 'Valvular Heart Disease',
'atrial_fibrillation' => 'Atrial Fibrillation',
'arrhythmia' => 'Arrhythmia',
];

return $diseasesAndAttacks;
}

/**
* Get list of image extensions.
*
* @return array
*/
public function getImageExtensions() {
$image_extensions = ['jpg', 'jpeg', 'png', 'gif', 'bmp'
    ,];
return $image_extensions;
}

/**
* Get list of video extensions.
*
* @return array
*/
public function getVideoExtensions() {
$video_extensions = ['mp4', 'webm', 'ogg',];
```

```
return $video_extensions;
}
}
```

Account Settings Validator

```
<?php namespace App\Services\Validators;

use App\Http\Middleware\StaffMiddleware;
use App\User;
use Carbon\Carbon;
use Illuminate\Support\Facades\File;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Facades\Validator;
use Symfony\Component\HttpFoundation\File\UploadedFile;

class AccountSettingsValidator
{
/*
|--------------------------------------------------------------------------

| Account Settings Validator Service
|--------------------------------------------------------------------------

|
| This service class handles the validation of updating
      user account settings.
|
*/

/**
* AccountSettingsValidator constructor.
*
* @param \App\Http\Middleware\StaffMiddleware
     $staffMiddleware
*/
public function __construct(StaffMiddleware
     $staffMiddleware) {
$this->staffMiddleware = $staffMiddleware;
}

/**
* Validate the update request on the general account
     settings.
*
* @param array $data
* @param \App\User $user
* @return \Illuminate\Contracts\Validation\Validator
*/
public function validateGeneralSettings(array $data,
     User $user) {
$validator_array = [
'first_name' => 'required|max:100',
'middle_name' => 'max:100',
'last_name' => 'required|max:100',
'sex' => 'required',
'birthday' => 'required|date_format:"Y-m-d"'
];

if ($user->patient) {
$validator_array['caregiver_first_name'] = 'max:100';
$validator_array['caregiver_middle_name'] = 'max:100';
$validator_array['caregiver_last_name'] = 'max:100';
} else if ($this->staffMiddleware->checkIfStaff($user))
     {
if ($user->medicalProfessional) {
$validator_array['specialties'] = 'max:100';
}
}

$validator = Validator::make($data, $validator_array);

$validator->after(function($validator) use ($data,
     $user) {
if (! Hash::check($data['password'], $user->password)){
$validator->errors()->add('password', 'The password is
     incorrect.');
}
});

return $validator;
}

/**
* Update the general account settings.
*
* @param array $data
* @param \App\User $user
* @return \Illuminate\Http\Response
*/
public function updateGeneralSettings($data, User $user
     ) {
$user->update([
'first_name' => $data['first_name'],
'middle_name' => $data['middle_name'],
'last_name' => $data['last_name'],
'sex' => $data['sex'],
'birthday' => $data['birthday'],
]);

if ($user->patient) {
$user->patient->update([
'caregiver_first_name' => $data['caregiver_first_name'
     ],
'caregiver_middle_name' => $data['caregiver_middle_name
     '],
'caregiver_last_name' => $data['caregiver_last_name'],
]);
} else if ($this->staffMiddleware->checkIfStaff($user))
     {
if ($user->medicalProfessional) {
$user->medicalProfessional->update([
'specialties' => $data['specialties'],
'clinic_schedule' => $data['clinic_schedule'],
]);
} else if ($user->nutritionist) {
$user->nutritionist->update([
'clinic_schedule' => $data['clinic_schedule'],
]);
} else if ($user->physiatrist) {
```

```
$user->physiatrist->update([
'clinic_schedule' => $data['clinic_schedule'],
]);
}
}
}

/**
* Validate the update request on the profile picture of
      the user.
*
* @param array $data
* @return \Illuminate\Http\Response
*/
public function validateProfilePicture(array $data) {
$rules = ['profile_picture' => 'required|mimes:jpeg,jpg
     ,png,gif,bmp|max:10000',];
$validator = Validator::make($data, $rules);
return $validator;
}

/**
* Update the profile picture.
*
* @param \App\User $user
* @param boolean $has_file
* @param \Symfony\Component\HttpFoundation\File\
     UploadedFile $uploaded_file
*/
public function updateProfilePicture(User $user,
     $has_file = false, UploadedFile $uploaded_file =
     null) {
if ($has_file and $uploaded_file and $uploaded_file->
     isValid()) {
$date = Carbon::now();
$extension = $uploaded_file->getClientOriginalExtension
     (); // getting file extension
$filename = $user->username . '-' .
$date->year . '-' . $date->month . '-' . $date->day . '
     -' .
$date->hour . '-' . $date->minute . '-' . $date->second
      . '-' .
rand(11111, 99999) . '.' . $extension; // renaming file
$mimeType = $uploaded_file->getMimeType();
$path = 'files/profile_pictures'; // upload path
$uploaded_file->move($path, $filename);

$stored_file = $user->file;
if ($stored_file != null) { // delete old file and
     update file record
File::delete($stored_file->path . '/' . $stored_file->
     filename);
$stored_file->update([
'filename' => $filename,
'extension' => $extension,
'mimeType' => $mimeType,
'path' => $path,
]);
} else { // create new file record
\App\Models\File::create([
'filename' => $filename,
'extension' => $extension,
'mimeType' => $mimeType,
'path' => $path,
'user_id' => $user->id,
]);
}
}
}

/**
* Validate the update request on the password of the
     user.
*
* @param array $data
* @param \App\User $user
* @return \Illuminate\Contracts\Validation\Validator
*/
public function validatePassword(array $data, User
     $user) {
$validator = Validator::make($data, [
'new_password' => 'required|confirmed'
]);

$validator->after(function($validator) use ($data,
     $user) {
if (! Hash::check($data['old_password'], $user->
     password)){
$validator->errors()->add('old_password', 'The password
     is incorrect.');
} elseif (strcmp($data['new_password'], $data['
     old_password']) == 0) {
$validator->errors()->add('new_password', 'The new
     password cannot be the same as the old password.')
     ;
}
});

return $validator;
}

}
```

Hospitals Validator

```
<?php namespace App\Services\Validators;

use App\Models\Hospital;
use Illuminate\Support\Facades\Validator;

class HospitalsValidator
{
/*
|--------------------------------------------------------------------------

| Hospitals Validator Service
|--------------------------------------------------------------------------

|
| This service class handles the validation of creating
```

```
      and updating
| hospital records.
|
*/

/**
* Validate the hospital record.
*
* @param array $data
* @param int $hospital_id
* @return \Illuminate\Contracts\Validation\Validator
*/
public function validate(array $data, $hospital_id = 0)
      {
$messages = [
'email_address.email' => 'The e-mail address field must
     be a valid e-mail address.'
];

$validator = Validator::make($data, [
'name' => 'required|unique:hospitals,' . $hospital_id,
'address' => 'required',
'email_address' => 'email',
], $messages);
return $validator;
}

/**
* Store the hospital record.
*
* @param array $data
*/
public function store(array $data) {
Hospital::create([
'name' => $data['name'],
'address' => $data['address'],
'landline_number' => $data['landline_number'],
'mobile_number' => $data['mobile_number'],
'fax_number' => $data['fax_number'],
'email_address' => $data['email_address'],
]);
}

/**
* Update the laboratory result.
*
* @param int $hospital_id
* @param array $data
*/
public function update($hospital_id, array $data) {
$hospital = Hospital::findOrFail($hospital_id);
$hospital->update([
'name' => $data['name'],
'address' => $data['address'],
'landline_number' => $data['landline_number'],
'mobile_number' => $data['mobile_number'],
'fax_number' => $data['fax_number'],
'email_address' => $data['email_address'],
]);
}
```

```
}
```

Laboratory Results Validator

```
<?php namespace App\Services\Validators;

use App\Models\DailyRecord;
use App\Models\File;
use App\Models\LaboratoryResult;
use App\Models\Patient;
use Carbon\Carbon;
use Illuminate\Support\Facades\Validator;
use Symfony\Component\HttpFoundation\File\UploadedFile;

class LaboratoryResultsValidator
{
/*
|--------------------------------------------------------------------------

| Laboratory Results Validator Service
|--------------------------------------------------------------------------

|
| This service class handles the validation of creating
      and updating
| laboratory results of a given medical record.
|
*/

/**
* Validate the laboratory result.
*
* @param array $data
* @param \Symfony\Component\HttpFoundation\File\
     UploadedFile $file
* @param boolean $has_file
* @return \Illuminate\Contracts\Validation\Validator
*/
public function validate(array $data, $has_file = false
     , UploadedFile $file = null) {
$validator = Validator::make($data, [
'type_of_laboratory_exam' => 'required',
'result' => 'required',
'date_of_test' => 'required',
]);
$validator->after(function($validator) use ($has_file,
     $file) {
if ($has_file and $file and ! $file->isValid()){
$validator->errors()->add('file', 'Uploaded file is not
     valid.');
}
});
return $validator;
}

/**
* Store the laboratory result.
*
* @param int $patient_id
```

```
* @param boolean $has_file
* @param \Symfony\Component\HttpFoundation\File\
    UploadedFile $file
* @param array $data
*/
public function store($patient_id, array $data,
    $has_file = false, UploadedFile $file = null) {
Patient::findOrFail($patient_id);
$laboratory_result = LaboratoryResult::create([
'type_of_laboratory_exam' => $data['
    type_of_laboratory_exam'],
'result' => $data['result'],
'unit_of_measurement' => $data['unit_of_measurement'],
'reference_values' => $data['reference_values'],
'date_of_test' => $data['date_of_test'],
'patient_id' => $patient_id,
]);

if ($has_file and $file) {
if ($file->isValid()) {
$date = Carbon::now();
$extension = $file->getClientOriginalExtension(); //
    getting file extension
$filename =
$date->year . '-' . $date->month . '-' . $date->day . '
    _' .
$date->hour . '-' . $date->minute . '-' . $date->second
     . '_' .
rand(11111, 99999) . '.' . $extension; // renaming file
$mimeType = $file->getMimeType();
$path = 'files/medical_resources'; // upload path
$file->move($path, $filename);

File::create([
'filename' => $filename,
'extension' => $extension,
'mimeType' => $mimeType,
'path' => $path,
'laboratory_result_id' => $laboratory_result->id,
]);
}
}
}

/**
* Update the laboratory result.
*
* @param int $laboratory_result_id
* @param array $data
* @param boolean $has_file
* @param \Symfony\Component\HttpFoundation\File\
    UploadedFile $uploaded_file
*/
public function update($laboratory_result_id, array
    $data, $has_file = false, UploadedFile
    $uploaded_file = null) {
$laboratory_result = LaboratoryResult::findOrFail(
    $laboratory_result_id);
$laboratory_result->update([
'type_of_laboratory_exam' => $data['
    type_of_laboratory_exam'],
'result' => $data['result'],
'unit_of_measurement' => $data['unit_of_measurement'],
'reference_values' => $data['reference_values'],
'date_of_test' => $data['date_of_test'],
]);

if ($has_file and $uploaded_file) {
if ($uploaded_file->isValid()) {
$date = Carbon::now();
$extension = $uploaded_file->getClientOriginalExtension
    (); // getting file extension
$filename =
$date->year . '-' . $date->month . '-' . $date->day . '
    _' .
$date->hour . '-' . $date->minute . '-' . $date->second
     . '_' .
rand(11111, 99999) . '.' . $extension; // renaming file
$mimeType = $uploaded_file->getMimeType();
$path = 'files/laboratory_results'; // upload path
$uploaded_file->move($path, $filename);

$stored_file = $laboratory_result->file;
if ($stored_file != null) { // delete old file and
    update file record
\Illuminate\Support\Facades\File::delete($stored_file->
    path . '/' . $stored_file->filename);
$stored_file->update([
'filename' => $filename,
'extension' => $extension,
'mimeType' => $mimeType,
'path' => $path,
]);
} else { // create new file record
File::create([
'filename' => $filename,
'extension' => $extension,
'mimeType' => $mimeType,
'path' => $path,
'laboratory_result_id' => $laboratory_result_id,
]);
}
}
}
}

}
```

Medical Records Validator

```
<?php namespace App\Services\Validators;

use App\Models\MedicalRecordAttribute;
use App\Models\MedicalRecordAttributeType;
use App\Models\NutritionistRecordAttribute;
use App\Models\NutritionistRecordAttributeType;
use App\Models\PhysiatristRecordAttribute;
use App\Models\PhysiatristRecordAttributeType;
use Illuminate\Support\Facades\Validator;
```

```
class MedicalRecordsValidator
{
/*
|--------------------------------------------------------------------------
| Medical Record Attributes Validator Service
|--------------------------------------------------------------------------
|
| This service class handles the validation of creating and updating medical
| record attributes in the stroke and medication form, nutrition form,
| rehabilitation medicine form, and laboratory results.
|
*/

/**
* Validate the stroke and medication record.
*
* @param array $data
* @return \Illuminate\Contracts\Validation\Validator
*/
public function validateMedicationRecord(array $data) {
$messages = [];
$validator = Validator::make($data, [], $messages);
return $validator;
}

/**
* Create or update the stroke and medication record attributes of the patient.
*
* @param int $medication_record_id
* @param array $data
*/
public function storeMedicationRecord($medication_record_id, array $data) {
$this->storeMedicationRecordAttributes($medication_record_id, $data);
}

/**
* Store the stroke and medication record attributes.
*
* @param int $medication_record_id
* @param array $data
*/
private function storeMedicationRecordAttributes($medication_record_id, array $data) {
$medical_record_attribute_types = MedicalRecordAttributeType::lists('type', 'id');

foreach ($medical_record_attribute_types as $id => $type) {
if (isset($data[$type])) { // if the data with the attribute type exists,
if (is_string($data[$type]) and strlen($data[$type]) > 0) {
// if the data with the attribute type is a string and has a value
$current_attribute = MedicalRecordAttribute::where('medication_record_id', $medication_record_id)->where('medical_record_attribute_type_id', $id)->first();
if ($current_attribute == null) {
// if attribute does not exist, create one
MedicalRecordAttribute::create([
'value' => $data[$type],
'medication_record_id' => $medication_record_id,
'medical_record_attribute_type_id' => $id,
]);
} else { // else, attribute exists in the attribute table
$current_attribute->update(['value' => $data[$type]]);
}

} else if (is_array($data[$type])) {
// else, the data with the attribute type is an array
// update these current attributes; add new ones or delete old ones if necessary
$attributes = MedicalRecordAttribute::where('medication_record_id', $medication_record_id)->where('medical_record_attribute_type_id', $id)->get();

$index = 0;
while ($index < count($attributes) and $index < count($data[$type])) {
$attributes[$index]->update(['value' => $data[$type][$index]]);
$index++;
}

while ($index < count($data[$type])) { // either add new ones, or
MedicalRecordAttribute::create([
'value' => $data[$type][$index],
'medication_record_id' => $medication_record_id,
'medical_record_attribute_type_id' => $id,
]);
$index++;
}

while ($index < count($attributes)) { // delete old ones
$attributes[$index]->delete();
$index++;
}
}
}
}
}

/**
* Validate the nutrition record.
*
* @param array $data
* @return \Illuminate\Contracts\Validation\Validator
*/
```

```
public function validateNutritionistRecord(array $data)
    {
$messages = [];
$validator = Validator::make($data, [], $messages);
return $validator;
}

/**
* Create or update the nutrition record attributes of
    the patient.
*
* @param int $nutritionist_record_id
* @param array $data
*/
public function storeNutritionistRecord(
    $nutritionist_record_id, array $data) {
$this->storeNutritionistRecordAttributes(
    $nutritionist_record_id, $data);
}

/**
* Store the nutrition record attributes.
*
* @param int $nutritionist_record_id
* @param array $data
*/
private function storeNutritionistRecordAttributes(
    $nutritionist_record_id, array $data) {
$nutritionist_record_attribute_types =
    NutritionistRecordAttributeType::lists('type', 'id
    ');

foreach ($nutritionist_record_attribute_types as $id =>
     $type) {
if (isset($data[$type])) { // if the data with the
    attribute type exists,
if (is_string($data[$type]) and strlen($data[$type]) >
    0) {
// if the data with the attribute type is a string and
    has a value
$current_attribute = NutritionistRecordAttribute::where
    ('nutritionist_record_id', $nutritionist_record_id
    )
->where('nutritionist_record_attribute_type_id', $id)->
    first();
if ($current_attribute == null) {
// if attribute does not exist, create one
NutritionistRecordAttribute::create([
'value' => $data[$type],
'nutritionist_record_id' => $nutritionist_record_id,
'nutritionist_record_attribute_type_id' => $id,
]);
} else { // else, attribute exists in the attribute
    table
$current_attribute->update(['value' => $data[$type]]);
}

} else if (is_array($data[$type])) {
// else, the data with the attribute type is an array
// update these current attributes; add new ones or
    delete old ones if necessary
$attributes = NutritionistRecordAttribute::where('
    nutritionist_record_id', $nutritionist_record_id)
->where('nutritionist_record_attribute_type_id', $id)->
    get();

$index = 0;
while ($index < count($attributes) and $index < count(
    $data[$type])) {
$attributes[$index]->update(['value' => $data[$type][
    $index]]);
$index++;
}

while ($index < count($data[$type])) { // either add
    new ones, or
NutritionistRecordAttribute::create([
'value' => $data[$type][$index],
'nutritionist_record_id' => $nutritionist_record_id,
'nutritionist_record_attribute_type_id' => $id,
]);
$index++;
}

while ($index < count($attributes)) { // delete old
    ones
$attributes[$index]->delete();
$index++;
}
}
}
}
}

/**
* Validate the rehabilitation medicine record.
*
* @param array $data
* @return \Illuminate\Contracts\Validation\Validator
*/
public function validatePhysiatristRecord(array $data)
    {
$messages = [];
$validator = Validator::make($data, [], $messages);
return $validator;
}

/**
* Create or update the rehabilitation medicine record
    attributes of the patient.
*
* @param int $physiatrist_record_id
* @param array $data
*/
public function storePhysiatristRecord(
    $physiatrist_record_id, array $data) {
$this->storePhysiatristRecordAttributes(
    $physiatrist_record_id, $data);
}
```

```
/**
* Store the rehabilitation medicine record attributes.
*
* @param int $physiatrist_record_id
* @param array $data
*/
private function storePhysiatristRecordAttributes(
    $physiatrist_record_id, array $data) {
$physiatrist_record_attribute_types =
    PhysiatristRecordAttributeType::lists('type', 'id'
    );

foreach ($physiatrist_record_attribute_types as $id =>
    $type) {
if (isset($data[$type])) { // if the data with the
    attribute type exists,
if (is_string($data[$type]) and strlen($data[$type]) >
    0) {
// if the data with the attribute type is a string and
    has a value
$current_attribute = PhysiatristRecordAttribute::where(
    'physiatrist_record_id', $physiatrist_record_id)
->where('physiatrist_record_attribute_type_id', $id)->
    first();
if ($current_attribute == null) {
// if attribute does not exist, create one
PhysiatristRecordAttribute::create([
'value' => $data[$type],
'physiatrist_record_id' => $physiatrist_record_id,
'physiatrist_record_attribute_type_id' => $id,
]);
} else { // else, attribute exists in the attribute
    table
$current_attribute->update(['value' => $data[$type]]);
}

} else if (is_array($data[$type])) {
// else, the data with the attribute type is an array
// update these current attributes; add new ones or
    delete old ones if necessary
$attributes = PhysiatristRecordAttribute::where('
    physiatrist_record_id', $physiatrist_record_id)
->where('physiatrist_record_attribute_type_id', $id)->
    get();

$index = 0;
while ($index < count($attributes) and $index < count(
    $data[$type])) {
$attributes[$index]->update(['value' => $data[$type][
    $index]]);
$index++;
}

while ($index < count($data[$type])) { // either add
    new ones, or
PhysiatristRecordAttribute::create([
'value' => $data[$type][$index],
'physiatrist_record_id' => $physiatrist_record_id,
'physiatrist_record_attribute_type_id' => $id,
]);
$index++;
}

while ($index < count($attributes)) { // delete old
    ones
$attributes[$index]->delete();
$index++;
}
}
}
}
}

}
```

Medical Resources Validator

```
<?php namespace App\Services\Validators;

use App\Models\DailyRecord;
use App\Models\File;
use App\Models\ForumCategory;
use App\Models\MedicalResource;
use App\User;
use Carbon\Carbon;
use Illuminate\Support\Facades\Validator;
use Symfony\Component\HttpFoundation\File\UploadedFile;

class MedicalResourcesValidator
{
/*
|--------------------------------------------------------------------------

| Medical Resources Validator Service
|--------------------------------------------------------------------------

|
| This service class handles the validation of creating
    and updating
| medical resources.
|
*/

/**
* Validate the medical resource.
*
* @param array $data
* @param \Symfony\Component\HttpFoundation\File\
    UploadedFile $file
* @param boolean $has_file
* @return \Illuminate\Contracts\Validation\Validator
*/
public function validate(array $data, $has_file = false
    , UploadedFile $file = null) {
$validator = Validator::make($data, [
'title' => 'required|max:100',
'authors' => 'required|max:100',
'publisher' => 'max:100',
'date_published' => 'date_format:"Y-m-d"',
]);
```

```
$validator->after(function($validator) use ($has_file,
    $file) {
if ($has_file and $file and ! $file->isValid()){
$validator->errors()->add('file', 'Uploaded file is not
    valid.');
}
});
return $validator;
}

/**
* Store the medical resource.
*
* @param int $user_id
* @param array $data
* @param boolean $has_file
* @param \Symfony\Component\HttpFoundation\File\
    UploadedFile $file
* @return \App\Models\MedicalResource
*/
public function store($user_id, array $data, $has_file
    = false, UploadedFile $file = null) {
$medical_resource = new MedicalResource();
$medical_resource['title'] = $data['title'];
$medical_resource['description'] = $data['description'
    ];
$medical_resource['authors'] = $data['authors'];
$medical_resource['publisher'] = $data['publisher'];
$medical_resource['date_published'] = $data['
    date_published'];

$user = User::findOrFail($user_id);
$medical_resource['user_id'] = $user->id;

$forum_category = ForumCategory::findOrFail($data['
    category']);
$medical_resource['forum_category_id'] =
    $forum_category->id;

$medical_resource['is_approved'] = false;
if ($user->supervisor) {
if ($user->supervisor->forumCategory) {
if ($user->supervisor->forumCategory->id ==
    $forum_category->id)
$medical_resource['is_approved'] = true;
}
}

$medical_resource->save();

if ($has_file and $file) {
if ($file->isValid()) {
$date = Carbon::now();
$extension = $file->getClientOriginalExtension(); //
    getting file extension
$filename =
$date->year . '-' . $date->month . '-' . $date->day . '
    _' .
$date->hour . '-' . $date->minute . '-' . $date->second
     . '-' .
rand(11111, 99999) . '.' . $extension; // renaming file
$mimeType = $file->getMimeType();
$path = 'files/medical_resources'; // upload path
$file->move($path, $filename);

File::create([
'filename' => $filename,
'extension' => $extension,
'mimeType' => $mimeType,
'path' => $path,
'medical_resource_id' => $medical_resource->id,
]);
}
}

return $medical_resource;
}

/**
* Update the laboratory result.
*
* @param int $medical_resource_id
* @param int $user_id
* @param array $data
* @param boolean $has_file
* @param \Symfony\Component\HttpFoundation\File\
    UploadedFile $uploaded_file
*/
public function update($medical_resource_id, $user_id,
    array $data, $has_file = false,
   UploadedFile $uploaded_file = null) {
$medical_resource = MedicalResource::findOrFail(
    $medical_resource_id);
$medical_resource->update([
'title' => $data['title'],
'description' => $data['description'],
'authors' => $data['authors'],
'publisher' => $data['publisher'],
'date_published' => $data['date_published'],
]);

$forum_category = ForumCategory::findOrFail($data['
    category']);
$medical_resource['forum_category_id'] =
    $forum_category->id;

$user = User::findOrFail($user_id);
$medical_resource['is_approved'] = false;
if ($user->supervisor) {
if ($user->supervisor->forumCategory) {
if ($user->supervisor->forumCategory->id ==
    $forum_category->id)
$medical_resource['is_approved'] = true;
}
}

$medical_resource->save();

if ($has_file and $uploaded_file) {
if ($uploaded_file->isValid()) {
```

```
$date = Carbon::now();
$extension = $uploaded_file->getClientOriginalExtension
    (); // getting file extension
$filename =
$date->year . '-' . $date->month . '-' . $date->day . '
    _' .
$date->hour . '-' . $date->minute . '-' . $date->second
     . '_' .
rand(11111, 99999) . '.' . $extension; // renaming file
$mimeType = $uploaded_file->getMimeType();
$path = 'files/laboratory_results'; // upload path
$uploaded_file->move($path, $filename);

$stored_file = $medical_resource->file;
if ($stored_file != null) { // delete old file and
    update file record
\Illuminate\Support\Facades\File::delete($stored_file->
    path . '/' . $stored_file->filename);
$stored_file->update([
'filename' => $filename,
'extension' => $extension,
'mimeType' => $mimeType,
'path' => $path,
]);
} else { // create new file record
File::create([
'filename' => $filename,
'extension' => $extension,
'mimeType' => $mimeType,
'path' => $path,
'medical_resource_id' => $medical_resource_id,
]);
}
}
}
}
```

Medication Regimens Validator

```
<?php namespace App\Services\Validators;

use App\Models\DailyRecord;
use App\Models\MedicationRegimen;
use Illuminate\Support\Facades\Validator;

class MedicationRegimensValidator
{
/*
|--------------------------------------------------------------------------

| Medication Regimens Validator Service
|--------------------------------------------------------------------------

|
| This service class handles the validation of creating
    and updating
| medication regimen for a particular patient.
|
*/

/**
* Validate the medication regimen.
*
* @param array $data
* @return \Illuminate\Contracts\Validation\Validator
*/
public function validate(array $data) {
$validator = Validator::make($data, [
'medication' => 'required|max:100',
'dose' => 'required|numeric|between:0,9999.99',
'frequency' => 'required|integer|between:0,99',
]);
return $validator;
}

/**
* Store the medication regimen.
*
* @param int $patient_id
* @param array $data
*/
public function store($patient_id, array $data) {
MedicationRegimen::create([
'medication' => $data['medication'],
'dose' => $data['dose'],
'frequency' => $data['frequency'],
'comments' => $data['comments'],
'time_of_medication' => $data['time_of_medication'],
'patient_id' => $patient_id,
]);
}

/**
* Update the medication regimen.
*
* @param int $medication_regimen_id
* @param array $data
*/
public function update($medication_regimen_id, array
    $data) {
$medication_regimen = MedicationRegimen::findOrFail(
    $medication_regimen_id);
$medication_regimen->update([
'medication' => $data['medication'],
'dose' => $data['dose'],
'frequency' => $data['frequency'],
'time_of_medication' => $data['time_of_medication'],
'comments' => $data['comments'],
]);
}
}
```

Nutritionist Regimens Validator

```
<?php namespace App\Services\Validators;

use App\Models\DailyRecord;
use App\Models\NutritionistRegimen;
use Illuminate\Support\Facades\Validator;
```

```
class NutritionistRegimensValidator
{
/*
|--------------------------------------------------------------------------
| Nutritionist Regimens Validator Service
|--------------------------------------------------------------------------
|
| This service class handles the validation of creating and updating
| nutritionist regimen for a particular patient.
|
*/

/**
* Validate the nutritionist regimen.
*
* @param array $data
* @return \Illuminate\Contracts\Validation\Validator
*/
public function validate(array $data) {
$validator = Validator::make($data, [

]);
return $validator;
}

/**
* Store the nutritionist regimen.
*
* @param int $patient_id
* @param array $data
*/
public function store($patient_id, array $data) {
NutritionistRegimen::create([
'breakfast' => $data['breakfast'],
'lunch' => $data['lunch'],
'dinner' => $data['dinner'],
'am_snack' => $data['am_snack'],
'pm_snack' => $data['pm_snack'],
'patient_id' => $patient_id,
]);
}

/**
* Update the nutritionist regimen.
*
* @param int $nutritionist_regimen_id
* @param array $data
*/
public function update($nutritionist_regimen_id, array $data) {
$nutritionist_regimen = NutritionistRegimen::findOrFail($nutritionist_regimen_id);
$nutritionist_regimen->update([
'breakfast' => $data['breakfast'],
'lunch' => $data['lunch'],
'dinner' => $data['dinner'],
'am_snack' => $data['am_snack'],
'pm_snack' => $data['pm_snack'],
]);
}
}
```

Patient Attributes Validator

```
<?php namespace App\Services\Validators;

use App\Models\PatientAttribute;
use App\Models\PatientAttributeType;
use Illuminate\Support\Facades\Validator;

class PatientAttributesValidator
{
/*
|--------------------------------------------------------------------------
| Patient Attributes Validator Service
|--------------------------------------------------------------------------
|
| This service class handles the validation of creating and
| updating patient attributes.
|
*/

/**
* Validate the update request on the general account settings.
*
* @param array $data
* @return \Illuminate\Contracts\Validation\Validator
*/
public function validate(array $data) {
$messages = [
'mother_s_maiden_first_name.required' => 'The mother\'s maiden first name field is required.',
'mother_s_maiden_middle_name.required' => 'The mother\'s maiden middle name field is required.',
'mother_s_maiden_last_name.required' => 'The mother\'s maiden last name field is required.',
'father_s_first_name.required' => 'The father\'s first name field is required.',
'father_s_middle_name.required' => 'The father\'s middle name field is required.',
'father_s_last_name.required' => 'The father\'s last name field is required.',
'contact_person_s_e-mail_address.email' => 'The contact person\'s e-mail address field must be a valid
e-mail address.',
];

return Validator::make($data, [
'civil_status' => 'required',
'mother_s_maiden_first_name' => 'required',
'mother_s_maiden_middle_name' => 'required',
'mother_s_maiden_last_name' => 'required',
```

```
'father_s_first_name' => 'required',
'father_s_middle_name' => 'required',
'father_s_last_name' => 'required',
'place_of_birth' => 'required',
'permanent_address' => 'required',
'religion' => 'required',
'nationality' => 'required',
'highest_educational_attainment' => 'required',
'occupation' => 'required',
'contact_person_s_e-mail_address' => 'email',
], $messages);
}

/**
* Create or update the patient attributes of the
     patient.
*
* @param int $id
* @param array $data
*/
public function store($id, array $data) {
$this->storeAttributes($id, $data);
$this->deleteDependentAttributes($id);
}

/**
* Store patient attributes.
*
* @param int $patient_id
* @param array $data
*/
private function storeAttributes($patient_id, array
     $data) {
$patient_attribute_types = PatientAttributeType::lists(
     'type', 'id');

foreach ($patient_attribute_types as $id => $type) {
if (isset($data[$type])) { // if the data with the
     attribute type exists,
if (is_string($data[$type]) and strlen($data[$type]) >
     0) {
// if the data with the attribute type is a string and
     has a value
$current_attribute = PatientAttribute::where('
     patient_id', $patient_id)
->where('patient_attribute_type_id', $id)->first();
if ($current_attribute == null) {
// if attribute does not exist, create one
PatientAttribute::create([
'value' => $data[$type],
'patient_id' => $patient_id,
'patient_attribute_type_id' => $id
]);
} else { // else, attribute exists in the attribute
     table
$current_attribute->update(['value' => $data[$type]]);
}

} else if (is_array($data[$type])) {
// else, the data with the attribute type is an array
// update these current attributes; add new ones or
     delete old ones if necessary
$attributes = PatientAttribute::where('patient_id',
     $patient_id)
->where('patient_attribute_type_id', $id)->get();

$index = 0;
while ($index < count($attributes) and $index < count(
     $data[$type])) {
$attributes[$index]->update(['value' => $data[$type][
     $index]]);
$index++;
}

while ($index < count($data[$type])) { // either add
     new ones, or
PatientAttribute::create([
'value' => $data[$type][$index],
'patient_id' => $patient_id,
'patient_attribute_type_id' => $id,
]);
$index++;
}

while ($index < count($attributes)) { // delete old
     ones
$attributes[$index]->delete();
$index++;
}
}
}
}
}

/**
* Delete patient attributes that are dependent to
     attributes that have no values.
*
* @param int $patient_id
*/
private function deleteDependentAttributes($patient_id)
     {
$dependent_attribute_types = [
['relationship_to_family_member', '
     type_of_stroke_of_family_member',],
['type_of_surgery', 'year_of_surgery',],
];
$attributes = [];

foreach ($dependent_attribute_types as $index =>
     $dependent_attribute_type) {
$attributes[$index] = [];
// Get the attributes given the dependent attribute
     types
foreach ($dependent_attribute_type as $attribute) {
$type_id = PatientAttributeType::where('type',
     $attribute)->first()->id;
$attributes[$index][] = PatientAttribute::where('
     patient_id', $patient_id)
->where('patient_attribute_type_id', $type_id)->get();
```

```
}

if (count($attributes[$index]) > 0) {
// If the first attribute has no value, delete other
    attributes with the same index
foreach ($attributes[$index][0] as $index2 =>
    $attribute) {
if (! $attribute->value) {
foreach ($attributes[$index] as $attribute_list) {
$attribute_list[$index2]->delete();
}
}
}
}
}
}

}
```

Patients Validator

```
<?php namespace App\Services\Validators;

use App\Models\HospitalMedicalProfession;
use App\Models\Patient;
use App\Services\Registrar;
use App\User;
use Illuminate\Support\Facades\Validator;

class PatientsValidator
{
/*
|--------------------------------------------------------------------------
| Patients Validator Service
|--------------------------------------------------------------------------
|
| This service class handles the validation of creating
    and
| updating patient account.
|
*/

/**
* Create a new patients validator instance.
*
* @param \App\Services\Registrar $registrar
*/
public function __construct(Registrar $registrar){
$this->registrar = $registrar;
}

/**
* Validate the request of a patient account.
*
* @param array $data
* @param int $patient_id
* @return \Illuminate\Contracts\Validation\Validator
*/
public function validate(array $data, $patient_id = 0)
    {
$messages = [
'hospital_id_number.required' => 'The hospital ID
    number field is required.'
];

return Validator::make($data, [
'hospital_id_number' => 'required|max:100|unique:
    patients,' . $patient_id,
'username' => 'required|max:50|min:6|unique:users',
'email' => 'required|email|max:255|unique:users',
'password' => 'required|confirmed|min:8',
'first_name' => 'required|max:100',
'middle_name' => 'max:100',
'last_name' => 'required|max:100',
'sex' => 'required',
'birthday' => 'required',
'caregiver_first_name' => 'max:100',
'caregiver_middle_name' => 'max:100',
'caregiver_last_name' => 'max:100'
], $messages);
}

/**
* Create the new patient account, as well as the
    patient record.
*
* @param array $data
* @param \App\Models\HospitalMedicalProfession
    $medicalProfession
*/
public function create(array $data,
    HospitalMedicalProfession $medicalProfession) {
$patient = $this->createPatient($data,
    $medicalProfession);

if ($patient) {
$user = new User();
$user['username'] = $data['username'];
$user['email'] = $data['email'];
$user['password'] = bcrypt($data['password']);
$user['is_unlocked'] = true;
$user['first_name'] = $data['first_name'];
$user['middle_name'] = $data['middle_name'];
$user['last_name'] = $data['last_name'];
$user['sex'] = $data['sex'];
$user['birthday'] = $data['birthday'];
$confirmation_code = str_random(30);
$user['confirmation_code'] = $confirmation_code;

$user['is_unlocked'] = true;
$user['is_email_confirmed'] = true;
$user->save();

$user->patient()->save($patient);

//$this->registrar->sendEmailConfirmation($data['
    username'], $data['email'], $confirmation_code);
```

```
}

return;
}

/**
* Create the patient record and link the professional
    record and the hospital record to it.
*
* @param array $data
* @param \App\Models\HospitalMedicalProfession
    $medicalProfession
* @return \App\Models\Patient
*/
private function createPatient(array $data,
    HospitalMedicalProfession $medicalProfession) {
$patient = new Patient([
'hospital_id_number' => $data['hospital_id_number'],
'caregiver_first_name' => $data['caregiver_first_name'
    ],
'caregiver_middle_name' => $data['caregiver_middle_name
    '],
'caregiver_last_name' => $data['caregiver_last_name']
]);

if ($medicalProfession) {
if ($medicalProfession->medicalProfessional and
    $medicalProfession->hospital) {
$patient['medical_professional_id'] =
    $medicalProfession->medicalProfessional->id;
$patient['hospital_id'] = $medicalProfession->hospital
    ->id;
return $patient;
}
}

return null;
}

/**
* Validate the update request of a patient account.
*
* @param array $data
* @param int $patient_id
* @return \Illuminate\Contracts\Validation\Validator
*/
public function validateUpdate(array $data, $patient_id
    = 0) {
$messages = [
'hospital_id_number.required' => 'The hospital ID
    number field is required.'
];

return Validator::make($data, [
'hospital_id_number' => 'required|max:100|unique:
    patients,' . $patient_id,
'first_name' => 'required|max:100',
'middle_name' => 'max:100',
'last_name' => 'required|max:100',
'sex' => 'required',
'birthday' => 'required',
'caregiver_first_name' => 'max:100',
'caregiver_middle_name' => 'max:100',
'caregiver_last_name' => 'max:100'
], $messages);
}

/**
* Update the patient account.
*
* @param array $data
* @param \App\Models\Patient $patient
*/
public function update(array $data, Patient $patient) {
$patient->user->update([
'first_name' => $data['first_name'],
'middle_name' => $data['middle_name'],
'last_name' => $data['last_name'],
'sex' => $data['sex'],
'birthday' => $data['birthday']
]);

$patient->update([
'hospital_id_number' => $data['hospital_id_number'],
'caregiver_first_name' => $data['caregiver_first_name'
    ],
'caregiver_middle_name' => $data['caregiver_middle_name
    '],
'caregiver_last_name' => $data['caregiver_last_name']
]);
}

}
```

Physiatrist Regimens Validator

```
<?php namespace App\Services\Validators;

use App\Models\DailyRecord;
use App\Models\PhysiatristRegimen;
use Illuminate\Support\Facades\Validator;

class PhysiatristRegimensValidator
{
/*
|--------------------------------------------------------------------------

| Physiatrist Regimens Validator Service
|--------------------------------------------------------------------------

|
| This service class handles the validation of creating
    and updating
| physiatrist regimen for a particular patient.
|
*/

/**
* Validate the physiatrist regimen.
*
```

```
* @param array $data
* @return \Illuminate\Contracts\Validation\Validator
*/
public function validate(array $data) {
$validator = Validator::make($data, [
'type_of_exercise' => 'required|max:100',
'frequency' => 'required|integer|between:0,9999',
'duration' => 'required|integer|between:0,9999',
]);
return $validator;
}

/**
* Store the physiatrist regimen.
*
* @param int $patient_id
* @param array $data
*/
public function store($patient_id, array $data) {
PhysiatristRegimen::create([
'type_of_exercise' => $data['type_of_exercise'],
'frequency' => $data['frequency'],
'duration' => $data['duration'],
'time_of_exercise' => $data['time_of_exercise'],
'comments' => $data['comments'],
'patient_id' => $patient_id,
]);
}

/**
* Update the physiatrist regimen.
*
* @param int $physiatrist_regimen_id
* @param array $data
*/
public function update($physiatrist_regimen_id, array
    $data) {
$physiatrist_regimen = PhysiatristRegimen::findOrFail(
    $physiatrist_regimen_id);
$physiatrist_regimen->update([
'type_of_exercise' => $data['type_of_exercise'],
'frequency' => $data['frequency'],
'duration' => $data['duration'],
'time_of_exercise' => $data['time_of_exercise'],
'comments' => $data['comments'],
]);
}
}
```

Registrar Validator

```
<?php namespace App\Services;

use App\Models\Hospital;
use App\Models\HospitalMedicalProfession;
use App\Models\HospitalNutritionistProfession;
use App\Models\HospitalPhysiatristProfession;
use App\Models\MedicalProfessional;
use App\Models\Nutritionist;
use App\Models\Physiatrist;
use App\User;
use Illuminate\Support\Facades\Mail;
use Validator;
use Illuminate\Contracts\Auth\Registrar as
    RegistrarContract;

class Registrar implements RegistrarContract {

/**
* Get a validator for an incoming registration request.
*
* @param array $data
* @return \Illuminate\Contracts\Validation\Validator
*/
public function validator(array $data)
{
return Validator::make($data, [
'username' => 'required|max:50|min:6|unique:users',
'email' => 'required|email|max:255|unique:users',
'password' => 'required|confirmed|min:8',
'first_name' => 'required|max:100',
'middle_name' => 'max:100',
'last_name' => 'required|max:100',
'sex' => 'required',
'birthday' => 'required',
'profession' => 'required|in:mp,n,p',
'hospital' => 'required|exists:hospitals,id',
]);
}

/**
* Create a new user instance after a valid registration
    .
*
* @param array $data
* @return \App\User
*/
public function create(array $data)
{
$hospital = Hospital::findOrFail($data['hospital']);

$user = new User();
$user['username'] = $data['username'];
$user['email'] = $data['email'];
$user['password'] = bcrypt($data['password']);
$user['first_name'] = $data['first_name'];
$user['middle_name'] = $data['middle_name'];
$user['last_name'] = $data['last_name'];
$user['sex'] = $data['sex'];
$user['birthday'] = $data['birthday'];
$confirmation_code = str_random(30);
$user['confirmation_code'] = $confirmation_code;
$user['is_unlocked'] = true;
$user['is_email_confirmed'] = true;
$user->save();

$this->addProfession($user, $data, $hospital, $data['
    profession']);
//$this->sendEmailConfirmation($data['username'], $data
    ['email'], $confirmation_code);
```

```
return $user;
}

/**
* Add a new profession of the new user in the chosen
    hospital.
*
* @param \App\User $user
* @param array $data
* @param \App\Models\Hospital $hospital
* @param string $profession
*/
private function addProfession(User $user, $data,
    Hospital $hospital, $profession) {
$hospital_id = $hospital->id;
switch($profession){
case 'mp': // Medical Professional
// save medical professional and its foreign keys to
    user and hospital
$medical_professional = new MedicalProfessional();
$medical_professional['clinic_schedule'] = $data['
    clinic_schedule'];
$user->medicalProfessional()->save(
    $medical_professional);

$hospital_profession = new HospitalMedicalProfession();
$hospital_profession->hospital_id = $hospital_id;
$hospital_profession->is_approved = true;
$medical_professional->hospitalMedicalProfessions()->
    save($hospital_profession);
break;
case 'n': // Nutritionist
// save nutritionist and its foreign keys to user and
    hospital
$nutritionist = new Nutritionist();
$nutritionist['clinic_schedule'] = $data['
    clinic_schedule'];
$user->nutritionist()->save($nutritionist);

$hospital_profession = new
    HospitalNutritionistProfession();
$hospital_profession->hospital_id = $hospital_id;
$hospital_profession->is_approved = true;
$nutritionist->hospitalNutritionistProfessions()->save(
    $hospital_profession);
break;
case 'p': // Physiatrist
// save physiatrist and its foreign keys to user and
    hospital
$physiatrist = new Physiatrist();
$physiatrist['clinic_schedule'] = $data['
    clinic_schedule'];
$user->physiatrist()->save($physiatrist);

$hospital_profession = new
    HospitalPhysiatristProfession();
$hospital_profession->hospital_id = $hospital_id;
$hospital_profession->is_approved = true;
$physiatrist->hospitalPhysiatristProfessions()->save(
    $hospital_profession);
break;
default:;
}
}

/**
* Send an e-mail containing the confirmation of the e-
    mail address.
*
* @param string $username
* @param string $email
* @param string $confirmation_code
*/
public function sendEmailConfirmation($username, $email
    , $confirmation_code) {
Mail::send('emails.confirm_email',
array(
'confirmation_code' => $confirmation_code,
'username' => $username,
),
function($message) use ($username, $email) {
$message->to($email, $username)
->subject('[Philippine_Stroke_Portal]_Confirm_your_E-
    mail_Address');
}
);
}

}
```

app.php

```
<?php

return [

/*
|--------------------------------------------------------------------------

| Application Debug Mode
|--------------------------------------------------------------------------

|
| When your application is in debug mode, detailed
    error messages with
| stack traces will be shown on every error that occurs
    within your
| application. If disabled, a simple generic error page
    is shown.
|
*/

'debug' => env('APP_DEBUG'),

/*
|--------------------------------------------------------------------------

| Application URL
|--------------------------------------------------------------------------
```

```
|
| This URL is used by the console to properly generate
    URLs when using
| the Artisan command line tool. You should set this to
     the root of
| your application so that it is used when running
    Artisan tasks.
|
*/

'url' => 'http://localhost',

/*
|--------------------------------------------------------------------------
| Application Timezone
|--------------------------------------------------------------------------
|
| Here you may specify the default timezone for your
    application, which
| will be used by the PHP date and date-time functions.
     We have gone
| ahead and set this to a sensible default for you out
    of the box.
|
*/

'timezone' => 'UTC',

/*
|--------------------------------------------------------------------------
| Application Locale Configuration
|--------------------------------------------------------------------------
|
| The application locale determines the default locale
    that will be used
| by the translation service provider. You are free to
    set this value
| to any of the locales which will be supported by the
    application.
|
*/

'locale' => 'en',

/*
|--------------------------------------------------------------------------
| Application Fallback Locale
|--------------------------------------------------------------------------
|
| The fallback locale determines the locale to use when
     the current one
| is not available. You may change the value to
    correspond to any of
| the language folders that are provided through your
    application.
|
*/

'fallback_locale' => 'en',

/*
|--------------------------------------------------------------------------
| Encryption Key
|--------------------------------------------------------------------------
|
| This key is used by the Illuminate encrypter service
    and should be set
| to a random, 32 character string, otherwise these
    encrypted strings
| will not be safe. Please do this before deploying an
    application!
|
*/

'key' => env('APP_KEY', 'SomeRandomString'),

'cipher' => MCRYPT_RIJNDAEL_128,

/*
|--------------------------------------------------------------------------
| Logging Configuration
|--------------------------------------------------------------------------
|
| Here you may configure the log settings for your
    application. Out of
| the box, Laravel uses the Monolog PHP logging library
    . This gives
| you a variety of powerful log handlers / formatters
    to utilize.
|
| Available Settings: "single", "daily", "syslog", "
    errorlog"
|
*/

'log' => 'daily',

/*
|--------------------------------------------------------------------------
| Autoloaded Service Providers
|--------------------------------------------------------------------------
|
| The service providers listed here will be
    automatically loaded on the
| request to your application. Feel free to add your
    own services to
| this array to grant expanded functionality to your
```

```
    applications.
|
*/

'providers' => [

/*
* Laravel Framework Service Providers...
*/
'Illuminate\Foundation\Providers\ArtisanServiceProvider
    ',
'Illuminate\Auth\AuthServiceProvider',
'Illuminate\Bus\BusServiceProvider',
'Illuminate\Cache\CacheServiceProvider',
'Illuminate\Foundation\Providers\
    ConsoleSupportServiceProvider',
'Illuminate\Routing\ControllerServiceProvider',
'Illuminate\Cookie\CookieServiceProvider',
'Illuminate\Database\DatabaseServiceProvider',
'Illuminate\Encryption\EncryptionServiceProvider',
'Illuminate\Filesystem\FilesystemServiceProvider',
'Illuminate\Foundation\Providers\
    FoundationServiceProvider',
'Illuminate\Hashing\HashServiceProvider',
'Illuminate\Mail\MailServiceProvider',
'Illuminate\Pagination\PaginationServiceProvider',
'Illuminate\Pipeline\PipelineServiceProvider',
'Illuminate\Queue\QueueServiceProvider',
'Illuminate\Redis\RedisServiceProvider',
'Illuminate\Auth\Passwords\PasswordResetServiceProvider
    ',
'Illuminate\Session\SessionServiceProvider',
'Illuminate\Translation\TranslationServiceProvider',
'Illuminate\Validation\ValidationServiceProvider',
'Illuminate\View\ViewServiceProvider',
'Illuminate\Html\HtmlServiceProvider',

/*
* Application Service Providers...
*/
'App\Providers\AppServiceProvider',
'App\Providers\BusServiceProvider',
'App\Providers\ConfigServiceProvider',
'App\Providers\EventServiceProvider',
'App\Providers\MacroServiceProvider',
'App\Providers\RouteServiceProvider',

'Laracasts\Flash\FlashServiceProvider',
],

/*
|---------------------------------------------------------
|
| Class Aliases
|---------------------------------------------------------
|
| This array of class aliases will be registered when
    this application
| is started. However, feel free to register as many as
    you wish as
| the aliases are "lazy" loaded so they don't hinder
    performance.
|
*/

'aliases' => [

'App'       => 'Illuminate\Support\Facades\App',
'Artisan'   => 'Illuminate\Support\Facades\Artisan',
'Auth'      => 'Illuminate\Support\Facades\Auth',
'Blade'     => 'Illuminate\Support\Facades\Blade',
'Bus'       => 'Illuminate\Support\Facades\Bus',
'Cache'     => 'Illuminate\Support\Facades\Cache',
'Config'    => 'Illuminate\Support\Facades\Config',
'Cookie'    => 'Illuminate\Support\Facades\Cookie',
'Crypt'     => 'Illuminate\Support\Facades\Crypt',
'DB'        => 'Illuminate\Support\Facades\DB',
'Eloquent'  => 'Illuminate\Database\Eloquent\Model',
'Event'     => 'Illuminate\Support\Facades\Event',
'File'      => 'Illuminate\Support\Facades\File',
'Hash'      => 'Illuminate\Support\Facades\Hash',
'Input'     => 'Illuminate\Support\Facades\Input',
'Inspiring' => 'Illuminate\Foundation\Inspiring',
'Lang'      => 'Illuminate\Support\Facades\Lang',
'Log'       => 'Illuminate\Support\Facades\Log',
'Mail'      => 'Illuminate\Support\Facades\Mail',
'Password'  => 'Illuminate\Support\Facades\Password',
'Queue'     => 'Illuminate\Support\Facades\Queue',
'Redirect'  => 'Illuminate\Support\Facades\Redirect',
'Redis'     => 'Illuminate\Support\Facades\Redis',
'Request'   => 'Illuminate\Support\Facades\Request',
'Response'  => 'Illuminate\Support\Facades\Response',
'Route'     => 'Illuminate\Support\Facades\Route',
'Schema'    => 'Illuminate\Support\Facades\Schema',
'Session'   => 'Illuminate\Support\Facades\Session',
'Storage'   => 'Illuminate\Support\Facades\Storage',
'URL'       => 'Illuminate\Support\Facades\URL',
'Validator' => 'Illuminate\Support\Facades\Validator',
'View'      => 'Illuminate\Support\Facades\View',
'Form'      => 'Illuminate\Html\FormFacade',
'Html'      => 'Illuminate\Html\HtmlFacade',
'Flash'     => 'Laracasts\Flash\Flash',
],

];
```

database.php

```
<?php

return [

/*
|---------------------------------------------------------
|
| PDO Fetch Style
|---------------------------------------------------------
|
```

```
| By default, database results will be returned as
    instances of the PHP
| stdClass object; however, you may desire to retrieve
    records in an
| array format for simplicity. Here you can tweak the
    fetch style.
|
*/

'fetch' => PDO::FETCH_CLASS,

/*
|--------------------------------------------------------------------------
| Default Database Connection Name
|--------------------------------------------------------------------------
|
| Here you may specify which of the database
    connections below you wish
| to use as your default connection for all database
    work. Of course
| you may use many connections at once using the
    Database library.
|
*/

'default' => 'sqlite',

/*
|--------------------------------------------------------------------------
| Database Connections
|--------------------------------------------------------------------------
|
| Here are each of the database connections setup for
    your application.
| Of course, examples of configuring each database
    platform that is
| supported by Laravel is shown below to make
    development simple.
|
|
| All database work in Laravel is done through the PHP
    PDO facilities
| so make sure you have the driver for your particular
    database of
| choice installed on your machine before you begin
    development.
|
*/

'connections' => [

'sqlite' => [
'driver'   => 'sqlite',
'database' => storage_path().'/database.sqlite',
'prefix'   => '',
],

'mysql' => [
'driver'    => 'mysql',
'host'      => env('DB_HOST', 'localhost'),
'database'  => env('DB_DATABASE', 'forge'),
'username'  => env('DB_USERNAME', 'forge'),
'password'  => env('DB_PASSWORD', ''),
'charset'   => 'utf8',
'collation' => 'utf8_unicode_ci',
'prefix'    => '',
'strict'    => false,
],

'pgsql' => [
'driver'   => 'pgsql',
'host'     => env('DB_HOST', 'localhost'),
'database' => env('DB_DATABASE', 'forge'),
'username' => env('DB_USERNAME', 'forge'),
'password' => env('DB_PASSWORD', ''),
'charset'  => 'utf8',
'prefix'   => '',
'schema'   => 'public',
],

'sqlsrv' => [
'driver'   => 'sqlsrv',
'host'     => env('DB_HOST', 'localhost'),
'database' => env('DB_DATABASE', 'forge'),
'username' => env('DB_USERNAME', 'forge'),
'password' => env('DB_PASSWORD', ''),
'prefix'   => '',
],

],

/*
|--------------------------------------------------------------------------
| Migration Repository Table
|--------------------------------------------------------------------------
|
| This table keeps track of all the migrations that
    have already run for
| your application. Using this information, we can
    determine which of
| the migrations on disk haven't actually been run in
    the database.
|
*/

'migrations' => 'migrations',

/*
|--------------------------------------------------------------------------
| Redis Databases
|--------------------------------------------------------------------------
|
```

```
| Redis is an open source, fast, and advanced key-value
      store that also
| provides a richer set of commands than a typical key-
      value systems
| such as APC or Memcached. Laravel makes it easy to
      dig right in.
|
*/

'redis' => [

'cluster' => false,

'default' => [
'host'     => '127.0.0.1',
'port'     => 6379,
'database' => 0,
],

],

];
```

mail.php

```
<?php

return [

/*
|--------------------------------------------------------------------------
| Mail Driver
|--------------------------------------------------------------------------
|
| Laravel supports both SMTP and PHP's "mail" function
      as drivers for the
| sending of e-mail. You may specify which one you're
      using throughout
| your application here. By default, Laravel is setup
      for SMTP mail.
|
| Supported: "smtp", "mail", "sendmail", "mailgun", "
      mandrill", "log"
|
*/

'driver' => env('MAIL_DRIVER', 'smtp'),

/*
|--------------------------------------------------------------------------
| SMTP Host Address
|--------------------------------------------------------------------------
|
| Here you may provide the host address of the SMTP
      server used by your
| applications. A default option is provided that is
      compatible with
| the Mailgun mail service which will provide reliable
      deliveries.
|
*/

'host' => env('MAIL_HOST'), // smtp.mailgun.org, .env
      -> mailtrap.io

/*
|--------------------------------------------------------------------------
| SMTP Host Port
|--------------------------------------------------------------------------
|
| This is the SMTP port used by your application to
      deliver e-mails to
| users of the application. Like the host we have set
      this value to
| stay compatible with the Mailgun e-mail application
      by default.
|
*/

'port' => env('MAIL_PORT', 587),

/*
|--------------------------------------------------------------------------
| Global "From" Address
|--------------------------------------------------------------------------
|
| You may wish for all e-mails sent by your application
      to be sent from
| the same address. Here, you may specify a name and
      address that is
| used globally for all e-mails that are sent by your
      application.
|
*/

'from' => ['address' => 'philippine.stroke.portal@gmail
      .com', 'name' => 'Philippine␣Stroke␣Portal'],

/*
|--------------------------------------------------------------------------
| E-Mail Encryption Protocol
|--------------------------------------------------------------------------
|
| Here you may specify the encryption protocol that
      should be used when
| the application send e-mail messages. A sensible
      default using the
| transport layer security protocol should provide
      great security.
|
```

```
*/

'encryption' => 'tls',

/*
|--------------------------------------------------------------------------
| SMTP Server Username
|--------------------------------------------------------------------------
|
| If your SMTP server requires a username for authentication, you should
| set it here. This will get used to authenticate with your server on
| connection. You may also set the "password" value below this one.
|
*/

'username' => env('MAIL_USERNAME'),

/*
|--------------------------------------------------------------------------
| SMTP Server Password
|--------------------------------------------------------------------------
|
| Here you may set the password required by your SMTP server to send out
| messages from your application. This will be given to the server on
| connection so that the application will be able to send messages.
|
*/

'password' => env('MAIL_PASSWORD'),

/*
|--------------------------------------------------------------------------
| Sendmail System Path
|--------------------------------------------------------------------------
|
| When using the "sendmail" driver to send e-mails, we will need to know
| the path to where Sendmail lives on this server. A default path has
| been provided here, which will work well on most of your systems.
|
*/

'sendmail' => '/usr/sbin/sendmail -bs',

/*
|--------------------------------------------------------------------------
| Mail "Pretend"
|--------------------------------------------------------------------------
|
| When this option is enabled, e-mail will not actually be sent over the
| web and will instead be written to your application's logs files so
| you may inspect the message. This is great for local development.
|
*/

'pretend' => false,

];
```

view.php

```
<?php

return [

/*
|--------------------------------------------------------------------------
| View Storage Paths
|--------------------------------------------------------------------------
|
| Most templating systems load templates from disk. Here you may specify
| an array of paths that should be checked for your views. Of course
| the usual Laravel view path has already been registered for you.
|
*/

'paths' => [
realpath(base_path('resources/views'))
],

/*
|--------------------------------------------------------------------------
| Compiled View Path
|--------------------------------------------------------------------------
|
| This option determines where all the compiled Blade templates will be
| stored for your application. Typically, this is within the storage
| directory. However, as usual, you are free to change this value.
|
*/
```

```
'compiled' => realpath(storage_path().'/framework/views
    '),

];
```

Users Migration

```
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateUsersTable extends Migration {

/**
* Run the migrations.
*
* @return void
*/
public function up()
{
Schema::create('users', function(Blueprint $table)
{
$table->increments('id');
$table->string('username', 50)->unique();
$table->string('email')->unique();
$table->string('password');
$table->string('first_name', 100);
$table->string('middle_name', 100)->nullable();
$table->string('last_name', 100);
$table->string('sex', 6);
$table->timestamp('birthday');
$table->boolean('is_unlocked')->default(true);
$table->boolean('is_email_confirmed')->default(true);
$table->string('confirmation_code')->nullable();
$table->rememberToken();
$table->timestamps();
});
}

/**
* Reverse the migrations.
*
* @return void
*/
public function down()
{
Schema::drop('users');
}

}
```

Password Resets Migration

```
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreatePasswordResetsTable extends Migration {

/**
* Run the migrations.
*
* @return void
*/
public function up()
{
Schema::create('password_resets', function(Blueprint
    $table)
{
$table->string('email')->index();
$table->string('token')->index();
$table->timestamp('created_at');
});
}

/**
* Reverse the migrations.
*
* @return void
*/
public function down()
{
Schema::drop('password_resets');
}

}
```

Medical Professionals Migration

```
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateMedicalProfessionalsTable extends Migration
    {

/**
* Run the migrations.
*
* @return void
*/
public function up()
{
Schema::create('medical_professionals', function(
    Blueprint $table)
{
$table->increments('id');
$table->integer('user_id')->unsigned();
$table->string('specialties')->nullable();
$table->text('clinic_schedule')->nullable();
$table->timestamps();

$table->foreign('user_id')->references('id')->on('users
    ')->onDelete('cascade');
});
}
```

```
/**
* Reverse the migrations.
*
* @return void
*/
public function down()
{
Schema::drop('medical_professionals');
}

}
```

Patients Migration

```
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreatePatientsTable extends Migration {

/**
* Run the migrations.
*
* @return void
*/
public function up()
{
Schema::create('patients', function(Blueprint $table)
{
$table->increments('id');
$table->integer('user_id')->unsigned();
$table->integer('hospital_id')->unsigned();
$table->integer('medical_professional_id')->unsigned()
    ->nullable();
$table->integer('nutritionist_id')->unsigned()->
    nullable();
$table->integer('physiatrist_id')->unsigned()->nullable
    ();
$table->string('hospital_id_number', 100);
$table->string('caregiver_first_name', 100)->nullable()
    ;
$table->string('caregiver_middle_name', 100)->nullable
    ();
$table->string('caregiver_last_name', 100)->nullable();
$table->timestamps();

$table->foreign('user_id')->references('id')->on('users
    ')->onDelete('cascade');
$table->foreign('hospital_id')->references('id')->on('
    hospitals')->onDelete('cascade');
$table->foreign('medical_professional_id')->references(
    'id')->on('medical_professionals');
$table->foreign('nutritionist_id')->references('id')->
    on('nutritionists');
$table->foreign('physiatrist_id')->references('id')->on
    ('physiatrists');
});
}
```

```
/**
* Reverse the migrations.
*
* @return void
*/
public function down()
{
Schema::drop('patients');
}

}
```

Hospitals Migration

```
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateHospitalsTable extends Migration {

/**
* Run the migrations.
*
* @return void
*/
public function up()
{
Schema::create('hospitals', function(Blueprint $table)
{
$table->increments('id');
$table->string('name', 255);
$table->text('address');
$table->string('landline_number', 20)->nullable();
$table->string('mobile_number', 20)->nullable();
$table->string('fax_number', 20)->nullable();
$table->string('email_address', 30)->nullable();
$table->boolean('is_unlocked')->default(false);
$table->timestamps();
});
}

/**
* Reverse the migrations.
*
* @return void
*/
public function down()
{
Schema::drop('hospitals');
}

}
```

Nutritionists Migration

```
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;
```

```
class CreateNutritionistsTable extends Migration {

/**
* Run the migrations.
*
* @return void
*/
public function up()
{
Schema::create('nutritionists', function(Blueprint
    $table)
{
$table->increments('id');
$table->integer('user_id')->unsigned();
$table->text('clinic_schedule')->nullable();
$table->timestamps();

$table->foreign('user_id')->references('id')->on('users
    ')->onDelete('cascade');
});
}

/**
* Reverse the migrations.
*
* @return void
*/
public function down()
{
Schema::drop('nutritionists');
}

}
```

Physiatrists Migration

```
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreatePhysiatristsTable extends Migration {

/**
* Run the migrations.
*
* @return void
*/
public function up()
{
Schema::create('physiatrists', function(Blueprint
    $table)
{
$table->increments('id');
$table->integer('user_id')->unsigned();
$table->text('clinic_schedule')->nullable();
$table->timestamps();

$table->foreign('user_id')->references('id')->on('users
    ')->onDelete('cascade');
});
}

/**
* Reverse the migrations.
*
* @return void
*/
public function down()
{
Schema::drop('physiatrists');
}

}
```

System Admins Migration

```
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateSystemAdminsTable extends Migration {

/**
* Run the migrations.
*
* @return void
*/
public function up()
{
Schema::create('system_admins', function(Blueprint
    $table)
{
$table->increments('id');
$table->integer('user_id')->unsigned();
$table->timestamps();

$table->foreign('user_id')->references('id')->on('users
    ')->onDelete('cascade');
});
}

/**
* Reverse the migrations.
*
* @return void
*/
public function down()
{
Schema::drop('system_admins');
}

}
```

Hospital Medical Professions Migration

```
<?php
```

```
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateHospitalMedicalProfessionsTable extends
    Migration {

/**
* Run the migrations.
*
* @return void
*/
public function up()
{
Schema::create('hospital_medical_professions', function
    (Blueprint $table)
{
$table->increments('id');
$table->integer('hospital_id')->unsigned();
$table->integer('medical_professional_id')->unsigned();
$table->boolean('is_approved')->default(true);
$table->timestamps();

$table->foreign('hospital_id')->references('id')->on('
    hospitals')->onDelete('cascade');
$table->foreign('medical_professional_id')->references(
    'id')->on('medical_professionals')->onDelete('
    cascade');
});
}

/**
* Reverse the migrations.
*
* @return void
*/
public function down()
{
Schema::drop('hospital_medical_professions');
}

}
```

Hospital Nutritionist Professions Migration

```
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateHospitalNutritionistProfessionsTable
    extends Migration {

/**
* Run the migrations.
*
* @return void
*/
public function up()
{
Schema::create('hospital_nutritionist_professions',
    function(Blueprint $table)
{
$table->increments('id');
$table->integer('hospital_id')->unsigned();
$table->integer('nutritionist_id')->unsigned();
$table->boolean('is_approved')->default(true);
$table->timestamps();

$table->foreign('hospital_id')->references('id')->on('
    hospitals')->onDelete('cascade');
$table->foreign('nutritionist_id')->references('id')->
    on('nutritionists')->onDelete('cascade');
});
}

/**
* Reverse the migrations.
*
* @return void
*/
public function down()
{
Schema::drop('hospital_nutritionist_professions');
}

}
```

Hospital Physiatrist Professions Migration

```
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateHospitalPhysiatristProfessionsTable extends
     Migration {

/**
* Run the migrations.
*
* @return void
*/
public function up()
{
Schema::create('hospital_physiatrist_professions',
    function(Blueprint $table)
{
$table->increments('id');
$table->integer('hospital_id')->unsigned();
$table->integer('physiatrist_id')->unsigned();
$table->boolean('is_approved')->default(true);
$table->timestamps();

$table->foreign('hospital_id')->references('id')->on('
    hospitals')->onDelete('cascade');
$table->foreign('physiatrist_id')->references('id')->on
    ('physiatrists')->onDelete('cascade');
});
}
```

```
/**
* Reverse the migrations.
*
* @return void
*/
public function down()
{
Schema::drop('hospital_physiatrist_professions');
}

}
```

Local Admins Migration

```
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateLocalAdminsTable extends Migration {

/**
* Run the migrations.
*
* @return void
*/
public function up()
{
Schema::create('local_admins', function(Blueprint
    $table)
{
$table->increments('id');
$table->integer('user_id')->unsigned();
$table->integer('hospital_id')->unsigned();
$table->timestamps();

$table->foreign('user_id')->references('id')->on('users
    ')->onDelete('cascade');
$table->foreign('hospital_id')->references('id')->on('
    hospitals')->onDelete('cascade');
});
}

/**
* Reverse the migrations.
*
* @return void
*/
public function down()
{
Schema::drop('local_admins');
}

}
```

Patient Attributes Migration

```
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreatePatientAttributesTable extends Migration {

/**
* Run the migrations.
*
* @return void
*/
public function up()
{
Schema::create('patient_attributes', function(Blueprint
    $table)
{
$table->increments('id');
$table->integer('patient_id')->unsigned();
$table->integer('patient_attribute_type_id')->unsigned
    ();
$table->string('value', 255)->nullable();
$table->timestamps();

$table->foreign('patient_id')->references('id')->on('
    patients')->onDelete('cascade');
$table->foreign('patient_attribute_type_id')->
    references('id')->on('patient_attribute_types')
->onDelete('cascade');
});
}

/**
* Reverse the migrations.
*
* @return void
*/
public function down()
{
Schema::drop('patient_attributes');
}

}
```

Patient Attribute Types Migration

```
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreatePatientAttributeTypesTable extends
    Migration {

/**
* Run the migrations.
*
* @return void
*/
public function up()
{
Schema::create('patient_attribute_types', function(
    Blueprint $table)
```

```
{
$table->increments('id');
$table->string('type', 255)->unique();
$table->timestamps();
});
}

/**
* Reverse the migrations.
*
* @return void
*/
public function down()
{
Schema::drop('patient_attribute_types');
}

}
```

Medical Record Attributes Migration

```
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateMedicalRecordAttributesTable extends
    Migration {

/**
* Run the migrations.
*
* @return void
*/
public function up()
{
Schema::create('medical_record_attributes', function(
    Blueprint $table)
{
$table->increments('id');
$table->integer('medication_record_id')->unsigned();
$table->integer('medical_record_attribute_type_id')->
    unsigned();
$table->string('value', 255)->nullable();
$table->timestamps();

$table->foreign('medication_record_id')->references('id
    ')->on('medication_records')->onDelete('cascade');
$table->foreign('medical_record_attribute_type_id')->
    references('id')->on('
    medical_record_attribute_types')
->onDelete('cascade');
});
}

/**
* Reverse the migrations.
*
* @return void
*/
public function down()
{
Schema::drop('medical_record_attributes');
}

}
```

Medical Record Attribute Types Migration

```
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateMedicalRecordAttributeTypesTable extends
    Migration {

/**
* Run the migrations.
*
* @return void
*/
public function up()
{
Schema::create('medical_record_attribute_types',
    function(Blueprint $table)
{
$table->increments('id');
$table->string('type', 255)->unique();
$table->timestamps();
});
}

/**
* Reverse the migrations.
*
* @return void
*/
public function down()
{
Schema::drop('medical_record_attribute_types');
}

}
```

Physiatrist Regimens Migration

```
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreatePhysiatristRegimensTable extends Migration
    {

/**
* Run the migrations.
*
* @return void
*/
```

```
public function up()
{
Schema::create('physiatrist_regimens', function(
    Blueprint $table)
{
$table->increments('id');
$table->integer('patient_id')->unsigned();
$table->string('type_of_exercise', 100);
$table->integer('frequency');
$table->integer('duration');
$table->text('time_of_exercise');
$table->text('comments')->nullable();
$table->timestamps();

$table->foreign('patient_id')->references('id')->on('
    patients')->onDelete('cascade');
});
}

/**
* Reverse the migrations.
*
* @return void
*/
public function down()
{
Schema::drop('physiatrist_regimens');
}

}
```

Medication Regimens Migration

```
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateMedicationRegimensTable extends Migration {

/**
* Run the migrations.
*
* @return void
*/
public function up()
{
Schema::create('medication_regimens', function(
    Blueprint $table)
{
$table->increments('id');
$table->integer('patient_id')->unsigned();
$table->string('medication', 100);
$table->float('dose');
$table->integer('frequency');
$table->text('time_of_medication');
$table->text('comments')->nullable();
$table->timestamps();

$table->foreign('patient_id')->references('id')->on('
    patients')->onDelete('cascade');
});
}

/**
* Reverse the migrations.
*
* @return void
*/
public function down()
{
Schema::drop('medication_regimens');
}

}
```

Nutritionist Regimens Migration

```
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateNutritionistRegimensTable extends Migration
    {

/**
* Run the migrations.
*
* @return void
*/
public function up()
{
Schema::create('nutritionist_regimens', function(
    Blueprint $table)
{
$table->increments('id');
$table->integer('patient_id')->unsigned();
$table->text('breakfast')->nullable();
$table->text('lunch')->nullable();
$table->text('dinner')->nullable();
$table->text('am_snack')->nullable();
$table->text('pm_snack')->nullable();
$table->timestamps();

$table->foreign('patient_id')->references('id')->on('
    patients')->onDelete('cascade');
});
}

/**
* Reverse the migrations.
*
* @return void
*/
public function down()
{
Schema::drop('nutritionist_regimens');
}
```

```
}
```

Laboratory Results Migration

```
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateLaboratoryResultsTable extends Migration {

/**
* Run the migrations.
*
* @return void
*/
public function up()
{
Schema::create('laboratory_results', function(Blueprint
      $table)
{
$table->increments('id');
$table->integer('patient_id')->unsigned();
$table->string('type_of_laboratory_exam', 30);
$table->string('unit_of_measurement', 30)->nullable();
$table->string('reference_values', 30)->nullable();
$table->date('date_of_test')->nullable();
$table->text('result');
$table->timestamps();

$table->foreign('patient_id')->references('id')->on('
     patients')->onDelete('cascade');
});
}

/**
* Reverse the migrations.
*
* @return void
*/
public function down()
{
Schema::drop('laboratory_results');
}

}
```

Nutritionist Record Attributes Migration

```
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateNutritionistRecordAttributesTable extends
     Migration {

/**
* Run the migrations.
*
* @return void
*/
public function up()
{
Schema::create('nutritionist_record_attributes',
     function(Blueprint $table)
{
$table->increments('id');
$table->integer('nutritionist_record_id')->unsigned();
$table->integer('nutritionist_record_attribute_type_id'
     )->unsigned();
$table->string('value', 255)->nullable();
$table->timestamps();

$table->foreign('nutritionist_record_id')->references('
     id')->on('nutritionist_records')->onDelete('
     cascade');
$table->foreign('nutritionist_record_attribute_type_id'
     )->references('id')
->on('nutritionist_record_attribute_types')->onDelete('
     cascade');
});
}

/**
* Reverse the migrations.
*
* @return void
*/
public function down()
{
Schema::drop('nutritionist_record_attributes');
}

}
```

Nutritionist Record Attribute Types Migration

```
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateNutritionistRecordAttributeTypesTable
     extends Migration {

/**
* Run the migrations.
*
* @return void
*/
public function up()
{
Schema::create('nutritionist_record_attribute_types',
     function(Blueprint $table)
{
$table->increments('id');
$table->string('type', 255)->unique();
$table->timestamps();
});
```

```
}

/**
* Reverse the migrations.
*
* @return void
*/
public function down()
{
Schema::drop('nutritionist_record_attribute_types');
}

}
```

Physiatrist Record Attributes Migration

```
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreatePhysiatristRecordAttributesTable extends
    Migration {

/**
* Run the migrations.
*
* @return void
*/
public function up()
{
Schema::create('physiatrist_record_attributes',
    function(Blueprint $table)
{
$table->increments('id');
$table->integer('physiatrist_record_id')->unsigned();
$table->integer('physiatrist_record_attribute_type_id')
    ->unsigned();
$table->string('value', 255)->nullable();
$table->timestamps();

$table->foreign('physiatrist_record_id')->references('
    id')->on('physiatrist_records')->onDelete('cascade
    ');
$table->foreign('physiatrist_record_attribute_type_id')
    ->references('id')
->on('physiatrist_record_attribute_types')->onDelete('
    cascade');
});
}

/**
* Reverse the migrations.
*
* @return void
*/
public function down()
{
Schema::drop('physiatrist_record_attributes');
}
}
```

Physiatrist Record Attribute Types Migration

```
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreatePhysiatristRecordAttributeTypesTable
    extends Migration {

/**
* Run the migrations.
*
* @return void
*/
public function up()
{
Schema::create('physiatrist_record_attribute_types',
    function(Blueprint $table)
{
$table->increments('id');
$table->string('type', 255)->unique();
$table->timestamps();
});
}

/**
* Reverse the migrations.
*
* @return void
*/
public function down()
{
Schema::drop('physiatrist_record_attribute_types');
}

}
```

Files Migration

```
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateFilesTable extends Migration {

/**
* Run the migrations.
*
* @return void
*/
public function up()
{
Schema::create('files', function(Blueprint $table)
{
$table->increments('id');
```

```
$table->integer('user_id')->unsigned()->nullable();
$table->integer('medical_resource_id')->unsigned()->
    nullable();
$table->integer('laboratory_result_id')->unsigned()->
    nullable();
$table->string('filename', 255);
$table->string('extension', 5);
$table->string('mimeType', 255);
$table->string('path', 50);
$table->timestamps();

$table->foreign('user_id')->references('id')->on('users
    ')->onDelete('cascade');
$table->foreign('medical_resource_id')->references('id'
    )->on('medical_resources')->onDelete('cascade');
$table->foreign('laboratory_result_id')->references('id
    ')->on('laboratory_results')->onDelete('cascade');
});
}

/**
* Reverse the migrations.
*
* @return void
*/
public function down()
{
Schema::drop('files');
}

}
```

Medical Resources Migration

```
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateMedicalResourcesTable extends Migration {

/**
* Run the migrations.
*
* @return void
*/
public function up()
{
Schema::create('medical_resources', function(Blueprint
    $table)
{
$table->increments('id');
$table->integer('user_id')->unsigned()->nullable();
$table->integer('forum_category_id')->unsigned();
$table->boolean('is_approved')->default(false);
$table->string('title', 100);
$table->text('description')->nullable();
$table->string('authors', 100);
$table->string('publisher', 100)->nullable();
$table->date('date_published')->nullable();
$table->timestamps();

$table->foreign('user_id')->references('id')->on('users
    ')->onDelete('cascade');
$table->foreign('forum_category_id')->references('id')
    ->on('forum_categories')->onDelete('cascade');
});
}

/**
* Reverse the migrations.
*
* @return void
*/
public function down()
{
Schema::drop('medical_resources');
}

}
```

Events Migration

```
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateEventsTable extends Migration {

/**
* Run the migrations.
*
* @return void
*/
public function up()
{
Schema::create('events', function(Blueprint $table)
{
$table->increments('id');
$table->integer('hospital_id')->unsigned();
$table->string('heading', 100);
$table->text('description')->nullable();
$table->date('date_of_event')->nullable();
$table->date('time_of_event')->nullable();
$table->timestamps();

$table->foreign('hospital_id')->references('id')->on('
    hospitals')->onDelete('cascade');
});
}

/**
* Reverse the migrations.
*
* @return void
*/
public function down()
{
Schema::drop('events');
```

```
}

}
```

Forums Migration

```
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateForumsTable extends Migration {

/**
* Run the migrations.
*
* @return void
*/
public function up()
{
Schema::create('forums', function(Blueprint $table)
{
$table->increments('id');
$table->integer('user_id')->unsigned();
$table->integer('supervisor_id')->unsigned()->nullable
    ();
$table->integer('forum_category_id')->unsigned();
$table->string('title', 100);
$table->text('description');
$table->timestamps();

$table->foreign('user_id')->references('id')->on('users
    ')->onDelete('cascade');
$table->foreign('supervisor_id')->references('id')->on(
    'supervisors')->onDelete('cascade');
$table->foreign('forum_category_id')->references('id')
    ->on('forum_categories')->onDelete('cascade');
});
}

/**
* Reverse the migrations.
*
* @return void
*/
public function down()
{
Schema::drop('forums');
}

}
```

Forum Categories Migration

```
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateForumCategoriesTable extends Migration {

/**
* Run the migrations.
*
* @return void
*/
public function up()
{
Schema::create('forum_categories', function(Blueprint
    $table)
{
$table->increments('id');
$table->string('name', 50)->unique();
$table->timestamps();
});
}

/**
* Reverse the migrations.
*
* @return void
*/
public function down()
{
Schema::drop('forum_categories');
}

}
```

Supervisors Migration

```
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateSupervisorsTable extends Migration {

/**
* Run the migrations.
*
* @return void
*/
public function up()
{
Schema::create('supervisors', function(Blueprint $table
    )
{
$table->increments('id');
$table->integer('user_id')->unsigned();
$table->integer('forum_category_id')->unsigned()->
    nullable();
$table->timestamps();

$table->foreign('user_id')->references('id')->on('users
    ')->onDelete('cascade');
$table->foreign('forum_category_id')->references('id')
    ->on('forum_categories')->onDelete('cascade');
});
}
```

```
/**
* Reverse the migrations.
*
* @return void
*/
public function down()
{
Schema::drop('supervisors');
}

}
```

Forum Comments Migration

```
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateForumCommentsTable extends Migration {

/**
* Run the migrations.
*
* @return void
*/
public function up()
{
Schema::create('forum_comments', function(Blueprint
    $table)
{
$table->increments('id');
$table->integer('user_id')->unsigned();
$table->integer('forum_id')->unsigned();
$table->text('comment');
$table->timestamps();

$table->foreign('user_id')->references('id')->on('user'
    )->onDelete('cascade');
$table->foreign('forum_id')->references('id')->on('
    forum')->onDelete('cascade');
});
}

/**
* Reverse the migrations.
*
* @return void
*/
public function down()
{
Schema::drop('forum_comments');
}

}
```

Follow-up Visits Migration

```
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateFollowUpVisitsTable extends Migration {

/**
* Run the migrations.
*
* @return void
*/
public function up()
{
Schema::create('follow_up_visits', function(Blueprint
    $table)
{
$table->increments('id');
$table->integer('user_id')->unsigned()->nullable();
$table->integer('patient_id')->unsigned();
$table->integer('medical_condition_id')->unsigned()->
    nullable();
$table->boolean('is_approved')->default(false);
$table->date('date_of_visit');
$table->timestamps();

$table->foreign('patient_id')->references('id')->on('
    patients');
$table->foreign('medical_condition_id')->references('id
    ')->on('medical_conditions');
});
}

/**
* Reverse the migrations.
*
* @return void
*/
public function down()
{
Schema::drop('follow_up_visits');
}

}
```

Medical Conditions Migration

```
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateMedicalConditionsTable extends Migration {

/**
* Run the migrations.
*
* @return void
*/
public function up()
{
```

```
Schema::create('medical_conditions', function(Blueprint
    $table)
{
$table->increments('id');
$table->integer('patient_id')->unsigned();
$table->text('daily_medical_problem')->nullable();
$table->text('describe_the_symptom')->nullable();
$table->text('medicine_taken')->nullable();
$table->text('effects_noted')->nullable();
$table->text('any_maneuver_taken')->nullable();
$table->timestamps();

$table->foreign('patient_id')->references('id')->on('
    patients');
});
}

/**
* Reverse the migrations.
*
* @return void
*/
public function down()
{
Schema::drop('medical_conditions');
}

}
```

Decision on Conditions Migration

```
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateDecisionOnConditionsTable extends Migration
    {

/**
* Run the migrations.
*
* @return void
*/
public function up()
{
Schema::create('decision_on_conditions', function(
    Blueprint $table)
{
$table->increments('id');
$table->integer('medical_professional_id')->unsigned();
$table->integer('medical_condition_id')->unsigned();
$table->text('decision');
$table->text('additional_notes')->nullable();
$table->timestamps();

$table->foreign('medical_professional_id')->references(
    'id')->on('medical_professionals');
$table->foreign('medical_condition_id')->references('id
    ')->on('medical_conditions');
});
}

/**
* Reverse the migrations.
*
* @return void
*/
public function down()
{
Schema::drop('decision_on_conditions');
}

}
```

Medication Records Migration

```
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateMedicationRecordsTable extends Migration {

/**
* Run the migrations.
*
* @return void
*/
public function up()
{
Schema::create('medication_records', function(Blueprint
    $table)
{
$table->increments('id');
$table->integer('patient_id')->unsigned();
$table->timestamps();

$table->foreign('patient_id')->references('id')->on('
    patients');
});
}

/**
* Reverse the migrations.
*
* @return void
*/
public function down()
{
Schema::drop('medication_records');
}

}
```

Nutritionist Records Migration

```
<?php

use Illuminate\Database\Schema\Blueprint;
```

```
use Illuminate\Database\Migrations\Migration;

class CreateNutritionistRecordsTable extends Migration
    {

/**
* Run the migrations.
*
* @return void
*/
public function up()
{
Schema::create('nutritionist_records', function(
    Blueprint $table)
{
$table->increments('id');
$table->integer('patient_id')->unsigned();
$table->timestamps();

$table->foreign('patient_id')->references('id')->on('
    patients');
});
}

/**
* Reverse the migrations.
*
* @return void
*/
public function down()
{
Schema::drop('nutritionist_records');
}

}
```

Physiatrist Records Migration

```
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreatePhysiatristRecordsTable extends Migration {

/**
* Run the migrations.
*
* @return void
*/
public function up()
{
Schema::create('physiatrist_records', function(
    Blueprint $table)
{
$table->increments('id');
$table->integer('patient_id')->unsigned();
$table->timestamps();

$table->foreign('patient_id')->references('id')->on('
    patients');
});
}

/**
* Reverse the migrations.
*
* @return void
*/
public function down()
{
Schema::drop('physiatrist_records');
}

}
```

Notifications Migration

```
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateNotificationsTable extends Migration {

/**
* Run the migrations.
*
* @return void
*/
public function up()
{
Schema::create('notifications', function(Blueprint
    $table)
{
$table->increments('id');
$table->integer('user_id');
$table->string('description', 100);
$table->string('link', 100);
$table->boolean('is_seen')->default(false);
$table->timestamps();
});
}

/**
* Reverse the migrations.
*
* @return void
*/
public function down()
{
Schema::drop('notifications');
}

}
```

master.css

```
/* TITLE BAR */
.custom-title-bar {
```

```
    width: 100%;
    height: auto;
    padding-top: 2rem;
    padding-bottom: 2rem;
    background: #f1eff2 url("../files/backgrounds/axiom
        -pattern-1920x1080.png") center center fixed;
    box-shadow: 0 0 5px #ddd;
    border-bottom: 1px solid #d8d8d8;
}
img.logo {
    margin-right: 1.5rem;
}
.custom-title-bar h2 {
    height: 100px;
    line-height: 100px;
}
.custom-title-bar img {
    width: 100px;
    height: 100px;
}

/* NAVIGATION BAR */
.navigation-area {
    box-shadow: 0 1px 5px #ddd;
    border-bottom: 1px solid #d8d8d8;
}

/* MAIN PART */
.custom-main-part {
    padding-bottom: 7rem;
}

/* MENU */
.custom-menu {
    background-color: #f5f5f5;
}
.no-margin {
    max-width: 100%;
    margin: 0;
}
.no-padding {
    padding: 0;
}
.docs-nav.menu {
    margin-left: 0;
    padding-top: 1rem;
    padding-bottom: 1rem;
}
.docs-nav.menu li small {
    color: #999;
    text-transform: uppercase;
    font-weight: bold;
    display: block;
    margin-top: 5px;
}
.docs-nav.menu a {
    line-height: 150%;
    font-size: 17px;
    padding-top: 1rem;
    padding-bottom: 1rem;
}
.docs-nav.menu a:hover {
    background: rgba(0, 0, 0, 0.05);
}
.docs-nav.menu .current a {
    background: rgba(0, 0, 0, 0.05);
}
.docs-nav.menu .label {
    background-color: #ddd;
    color: #777;
    margin-left: 5px;
    padding: 0.22222rem 0.44444rem 0.22222rem;
    font-size: 0.61111rem;
}

/* BODY */
.custom-body {
    padding-top: 1.5rem;
    padding-left: 1.5rem;
}
.height-scroll {
    max-height: 20rem;
    overflow-y: scroll;
}
p.indent {
    text-indent: 2rem;
}

/* FOOTER */
.custom-footer {
    background-color: #09516b;
    width: 100%;
    bottom: 0%;
    padding-top: 1.5rem;
    padding-bottom: 1.5rem;
}

/* BADGE */
.badge {
    padding: 0.5em;
    min-height: 2.1em;
    vertical-align: middle;
}

/* NOTIFICATIONS */
.callout.notification:hover, .callout.notification:
    hover {
    background-color: #c5c5c5;
}

/* CENTERING ELEMENTS */
.center {
    text-align: center;
}
.tmid {
    margin: 0px auto;
}

/* MARGIN */
.margin-top-20 {
```

```
    margin-top: 20px;
}
.margin-bottom-10 {
    margin-bottom: 10px;
}
.margin-bottom-20 {
    margin-bottom: 20px;
}

/* PADDING */
.padding-top, .padding-top-profile-picture {
    padding-top: 13px;
}
.padding-top-30 {
    padding-top: 30px;
}

/* LIST */
ul.no-bullet {
    list-style-type: none;
    padding: 0;
    margin: 0;
}

/* FORM FIELDS */
input.default {
    color: #0a0a0a;
}
select.default {
    color: #0a0a0a;
    background-color: #fefefe;
}

/* FONT */
.font-white {
    color: #ffffff;
}
.font-black {
    color: #000000;
}
.font-green {
    color: #3e8f3e;
}
.font-0875rem{
    font-size: 0.875rem;
}
.font-12-px {
    font-size: 0.9rem;
}
.font-1-rem {
    font-size: 1rem;
}
.font-125-rem {
    font-size: 1.25rem;
}
a.font-white:link, a.font-white:visited, a.font-white:
    hover, a.font-white:active {
    color: #cacaca;
}
a.font-black:link, a.font-black:visited, a.font-black:
    hover, a.font-black:active {
    color: #000000;
}

/* ERROR VALIDATION */
span.custom-form-error, small.custom-form-error {
    display: block;
    font-size: 0.66667rem;
    font-style: italic;
    font-weight: normal;
    margin-top: -1rem;
    padding: 0.33333rem 0.5rem 0.5rem;
    background: #f04124;
    color: #ffffff;
}
span.no-margin-top, small.no-margin-top {
    margin-top: -1px;
}
.fade-in.mui-enter, .fade-out.mui-leave {
    transition-duration: 200ms;
}

/* PROFILE PICTURE */
img.profile-picture {
    width: 80rem;
    height: 80rem;
    margin-left: 5rem;
}
div.profile-picture-container {
    display: inline-block;
    width: 50rem;
    height: 50rem;
    border-radius: 50%;
    background-repeat: no-repeat;
    background-position: center center;
    background-size: cover;
}

/* MENU BUTTONS */
.menu li a {
    padding: 0.85em 1em;
}
.menu.ul-margin li a, .menu.ul-margin li input.button {
    margin-left: 0.5rem;
    margin-right: 0.5rem;
}
button.menu-button-width {
    width: 18.75rem;
}
```

foundation-profile-card.css

```
.profile-card {

}
.profile-card img {
    display: block;
    text-align: center;
    margin-right: 1rem;
    float: left;
```

```
    border-radius: 50%;
}
#rem-2 img {
    width: 2rem;
    height: 2rem;
}
#rem-3 img {
    width: 3rem;
    height: 3rem;
}
```

Register Form View

```
<div class="row">
<div class="small-4 columns">
<label for="username" class="middle">Username*</label>
</div>
<div class="small-8 columns end">
<input type="text" class="default" id="username" name="
    username" value="{{ old('username') }}">
@foreach ($errors->get('username') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('username')) - 1)<br>
    @endif
@endforeach
</div>
</div>
<div class="row">
<div class="small-4 columns">
<label for="email" class="middle">E-mail Address*</
    label>
</div>
<div class="small-8 columns end">
<input type="email" class="default" id="email" name="
    email" value="{{ old('email') }}">
@foreach ($errors->get('email') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('email')) - 1)<br>
    @endif
@endforeach
</div>
</div>
<div class="row">
<div class="small-4 columns">
<label for="password" class="middle">Password*</label>
</div>
<div class="small-8 columns end">
<input type="password" class="default" id="password"
    name="password">
@foreach ($errors->get('password') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('password')) - 1)<br>
    @endif
@endforeach
</div>
</div>
<div class="row">
<div class="small-4 columns">
<label for="password2" class="middle">Confirm Password
    *</label>
</div>
<div class="small-8 columns end">
<input type="password" class="default" id="password2"
    name="password_confirmation">
@foreach ($errors->get('password_confirmation') as $key
    => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('password_confirmation'
    )) - 1)<br>@endif
@endforeach
</div>
</div>
<!-- First Name Form Input -->
<div class="row">
<div class="small-4 columns">
{!! Form::label('first_name', 'First Name*', ['class'
    => 'middle', 'for' => 'first_name']) !!}
</div>
<div class="small-8 columns end">
{!! Form::text('first_name', null, ['class' => 'default
    ', 'id' => 'first_name']) !!}
@foreach ($errors->get('first_name') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('first_name')) - 1)<br>
    @endif
@endforeach
</div>
</div>
<!-- Middle Name Form Input -->
<div class="row">
<div class="small-4 columns">
{!! Form::label('middle_name', 'Middle Name',
['class' => 'middle', 'for' => 'middle_name']) !!}
</div>
<div class="small-8 columns end">
{!! Form::text('middle_name', null, ['class' => '
    default', 'id' => 'middle_name']) !!}
@foreach ($errors->get('middle_name') as $key => $error
    )
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('middle_name')) - 1)<br
    >@endif
@endforeach
</div>
</div>
<!-- Last Name Form Input -->
<div class="row">
<div class="small-4 columns">
{!! Form::label('last_name', 'Last Name*', ['class' =>
    'middle', 'for' => 'last_name']) !!}
</div>
<div class="small-8 columns end">
{!! Form::text('last_name', null, ['class' => 'default'
    , 'id' => 'last_name']) !!}
@foreach ($errors->get('last_name') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('last_name')) - 1)<br>
    @endif
@endforeach
</div>
```

```
</div>
<!-- Sex Form Input -->
<div class="row margin-bottom-10">
<div class="small-4 columns">
{!! Form::label('sex', 'Sex*') !!}
</div>
<fieldset class="small-8 columns end">
{!! Form::radio('sex', 'male', false, ['id' => 'male'])
    !!}<label for="male">Male</label>
{!! Form::radio('sex', 'female', false, ['id' => '
    female']) !!}
<label for="female">Female</label>
@foreach ($errors->get('sex') as $key => $error)
<small class="custom-form-error no-margin-top">{{
    $error }}</small>
@if ($key == count($errors->get('sex')) - 1)<br>@endif
@endforeach
</fieldset>
</div>
<!-- Birthday Form Input -->
<div class="row">
<div class="small-4 columns">
{!! Form::label('birthday', 'Birthday*', ['class' => '
    middle', 'for' => 'birthday']) !!}
</div>
<div class="small-8 columns end">
{!! Form::input('date', 'birthday', null, ['class' => '
    default', 'id' => 'birthday']) !!}
@foreach ($errors->get('birthday') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('birthday')) - 1)<br>
    @endif
@endforeach
</div>
</div>
```

Login View

```
@extends('app')
@section('css')@endsection
@section('content')
<div class="row">
<div class="small-10 small-centered medium-8 medium-
    centered large-6 large-centered column">
<div class="center"><h4>Login</h4></div>
<hr>
@if (count($errors) > 0)
<div class="callout alert">
<p>{{ $errors->first() }}</p>
</div>
@endif
<form role="form" method="POST" action="{{ url('/auth/
    login') }}">
<input type="hidden" name="_token" value="{{ csrf_token
    () }}">
<div class="row margin-bottom-10">
<div class="small-5 columns">
<label for="username_or_email" class="middle">Username
    or E-Mail Address</label>
</div>
<div class="small-7 columns end">
<input type="text" id="username_or_email" name="
    username_or_email"
value="{{ old('username_or_email') }}">
</div>
</div>
<div class="row margin-bottom-10">
<div class="small-5 columns">
<label for="password" class="middle">Password</label>
</div>
<div class="small-7 columns end">
<input type="password" id="password" name="password">
</div>
</div>
<div class="row margin-bottom-10">
<fieldset class="small-12 column end">
<input type="checkbox" name="remember" id="remember"><
    label for="remember">Remember Me</label>
</fieldset>
</div>
<div class="row">
<div class="small-12 column center end">
<button type="submit" class="button">Login</button>
<a class="hollow button" href="{{ url('/password/email
    ') }}">Forgot Your Password?</a>
</div>
</div>
</form>
</div>
</div>
@endsection
@section('js')@endsection
```

Password View

```
@extends('app')
@section('css')@endsection
@section('content')
<div class="row">
<div class="small-10 small-centered medium-8 medium-
    centered large-6 large-centered column">
<div class="center"><h4>Reset Password</h4></div>
<hr>
@if (session('status'))
<div class="callout success">
{{ session('status') }}
</div>
@endif
@if (count($errors) > 0)
<div class="callout alert">
<p>{{ $errors->first() }}</p>
</div>
@endif
<form role="form" method="POST" action="{{ url('
    password/email') }}">
<input type="hidden" name="_token" value="{{ csrf_token
    () }}">
<!-- E-mail Address Form Input -->
<div class="row">
<div class="small-5 columns">
```

```
{!! Form::label('email', 'E-mail Address', ['class' =>
    'middle']) !!}
</div>
<div class="small-7 columns end">
{!! Form::text('email', old('email'), ['class' => '
    default', 'id' => 'email']) !!}
</div>
</div>
<div class="row">
<div class="small-12 column center end">
<button type="submit" class="button">Send Password
    Reset Link</button>
</div>
</div>
</form>
</div>
</div>
@endsection
@section('js')@endsection
```

Register View

```
@extends('app')
@section('css')@endsection
@section('content')
<div class="row">
<div class="small-10 small-centered medium-9 medium-
    centered large-8 large-centered column">
<div class="center"><h4>Register</h4></div>
<hr>
@if (count($errors) > 0)
<div class="callout alert">
<p>There is a problem with your input.</p>
</div>
@endif
<p class="font-12-px">
Note: This registration page is for medical
    professionals, nutritionists, and rehabilitation
    medicine
doctors only.
</p>
<form role="form" method="POST" action="{{ url('/auth/
    register') }}">
<input type="hidden" name="_token" value="{{ csrf_token
    () }}">
@include ('auth._register_form')
<!-- Profession Form Input -->
<div class="row">
<div class="small-4 columns">
{!! Form::label('profession', 'Profession*', ['class'
    => 'middle', 'for' => 'profession']) !!}
</div>
<div class="small-8 columns end">
{!! Form::select('profession', $profession_list, null,
['class' => 'default', 'id' => 'profession_list']) !!}
@foreach ($errors->get('profession') as $key => $error)
<small class="form-error">{{ $error }}</small>
@if ($key == count($errors->get('profession')) - 1)<br>
    @endif
@endforeach
</div>
</div>
<!-- Hospital or Clinic Form Input -->
<div class="row">
<div class="small-4 columns">
{!! Form::label('hospital', 'Hospital or Clinic*',
['class' => 'middle', 'for' => 'hospital']) !!}
</div>
<div class="small-8 columns end">
{!! Form::select('hospital', $hospital_list, null,
['class' => 'default', 'id' => 'profession_list']) !!}
@foreach ($errors->get('hospital') as $key => $error)
<small class="form-error">{{ $error }}</small>
@if ($key == count($errors->get('hospital')) - 1)<br>
    @endif
@endforeach
</div>
</div>
<div class="row column">
<p class="font-12-px">
Is the hospital you are searching for not in the list?
<a href="{{ url('/hospitals/create') }}">Add a new
    hospital record here.</a>
</p>
</div>
<!-- Clinic Schedule Form Input -->
<div class="row">
<div class="small-4 columns">
{!! Form::label('clinic_schedule', 'Clinic Schedule', [
    'class' => 'middle']) !!}
</div>
<div class="small-8 columns end">
{!! Form::textarea('clinic_schedule', null,
['class' => 'default', 'id' => 'clinic_schedule', 'rows
    ' => '3', 'placeholder' =>
'Input your schedule in your clinic (e.g., Monday:
    10:00am to 3:00pm, Tuesday and Thursday: '
. '11:00am to 2:00pm, ...).']) !!}
</div>
</div>
<p class="float-left help-text">Fields marked with *
    are mandatory.</p>
<div class="row">
<div class="small-12 center column end">
<button type="submit" class="button">Register</button>
</div>
</div>
</form>
</div>
</div>
@endsection
@section('js')@endsection
```

Reset View

```
@extends('app')
@section('css')@endsection
@section('content')
<div class="row">
<div class="small-10 small-centered medium-8 medium-
```

```
    centered large-6 large-centered column">
<div class="center"><h4>Reset Password</h4></div>
<hr>
@if (count($errors) > 0)
<div class="callout alert">
<p>{{ $errors->first() }}</p>
</div>
@endif
<form role="form" method="POST" action="{{ url('
    password/reset') }}">
<input type="hidden" name="_token" value="{{ csrf_token
    () }}">
<input type="hidden" name="token" value="{{ $token }}">
<!-- E-mail Address Form Input -->
<div class="row">
<div class="small-5 columns">
{!! Form::label('email', 'E-mail Address', ['class' =>
    'middle']) !!}
</div>
<div class="small-7 columns end">
{!! Form::text('email', old('email'), ['class' => '
    default', 'id' => 'email']) !!}
</div>
</div>
<!-- Password Form Input -->
<div class="row">
<div class="small-5 columns">
{!! Form::label('password', 'Password', ['class' => '
    middle']) !!}
</div>
<div class="small-7 columns end">
{!! Form::password('password', ['class' => 'default', '
    id' => 'password']) !!}
</div>
</div>
<!-- Confirm Password Form Input -->
<div class="row">
<div class="small-5 columns">
{!! Form::label('password_confirmation', 'Confirm
    Password', ['class' => 'middle']) !!}
</div>
<div class="small-7 columns end">
{!! Form::password('password_confirmation', ['class' =>
     'default',
'id' => 'password_confirmation']) !!}
</div>
</div>
<div class="row">
<div class="small-12 column center end">
<button type="submit" class="button">Reset Password</
    button>
</div>
</div>
</form>
</div>
</div>
@endsection
@section('js')@endsection
```

General Account Form View

```
<!-- First Name Form Input -->
<div class="row">
<div class="small-5 columns">
{!! Form::label('first_name', 'First Name*', ['class'
    => 'middle', 'for' => 'first_name']) !!}
</div>
<div class="small-7 columns end">
{!! Form::text('first_name', null, ['class' => 'default
    ', 'id' => 'first_name']) !!}
@foreach ($errors->get('first_name') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('first_name')) - 1)<br>
    @endif
@endforeach
</div>
</div>
<!-- Middle Name Form Input -->
<div class="row">
<div class="small-5 columns">
{!! Form::label('middle_name', 'Middle Name',
['class' => 'middle', 'for' => 'middle_name']) !!}
</div>
<div class="small-7 columns end">
{!! Form::text('middle_name', null, ['class' => '
    default', 'id' => 'middle_name']) !!}
@foreach ($errors->get('middle_name') as $key => $error
    )
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('middle_name')) - 1)<br
    >@endif
@endforeach
</div>
</div>
<!-- Last Name Form Input -->
<div class="row">
<div class="small-5 columns">
{!! Form::label('last_name', 'Last Name*', ['class' =>
    'middle', 'for' => 'last_name']) !!}
</div>
<div class="small-7 columns end">
{!! Form::text('last_name', null, ['class' => 'default'
    , 'id' => 'last_name']) !!}
@foreach ($errors->get('last_name') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('last_name')) - 1)<br>
    @endif
@endforeach
</div>
</div>
<!-- Sex Form Input -->
<div class="row margin-bottom-10">
<div class="small-5 columns">
{!! Form::label('sex', 'Sex*') !!}
</div>
<fieldset class="small-7 columns end">
@if ($user->sex == "male")
{!! Form::radio('sex', 'male', true, ['id' => 'sex'])
    !!}<label for="sex">Male</label>
@else
{!! Form::radio('sex', 'male', false, ['id' => 'sex'])
```

```
    !!}<label for="sex">Male</label>
@endif
@if ($user->sex == "female")
{!! Form::radio('sex', 'female', true, ['id' => 'sex'])
    !!}<label for="sex">Female</label>
@else
{!! Form::radio('sex', 'female', false, ['id' => 'sex'
    ]) !!}<label for="sex">Female</label>
@endif
@foreach ($errors->get('sex') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('sex')) - 1)<br>@endif
@endforeach
</fieldset>
</div>
<!-- Birthday Form Input -->
<div class="row">
<div class="small-5 columns">
{!! Form::label('birthday', 'Birthday*', ['class' => '
    middle', 'for' => 'birthday']) !!}
</div>
<div class="small-7 columns end">
{!! Form::input('date', 'birthday', $user->birthday
    [0]->format('Y-m-d'),
['class' => 'default', 'id' => 'birthday']) !!}
@foreach ($errors->get('birthday') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('birthday')) - 1)<br>
    @endif
@endforeach
</div>
</div>
@if ($user->patient)
<!-- Caregiver\'s First Name Form Input -->
<div class="row">
<div class="small-5 columns">
{!! Form::label('caregiver_first_name', 'Caregiver\'s
    First Name', ['class' => 'middle']) !!}
</div>
<div class="small-7 columns end">
{!! Form::text('caregiver_first_name', $user->patient->
    caregiver_first_name,
['class' => 'default', 'id' => 'caregiver_first_name'])
    !!}
@foreach ($errors->get('caregiver_first_name') as $key
    => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('caregiver_first_name')
    ) - 1)<br>@endif
@endforeach
</div>
</div>
<!-- Caregiver\'s Middle Name Form Input -->
<div class="row">
<div class="small-5 columns">
{!! Form::label('caregiver_middle_name', 'Caregiver\'s
    Middle Name', ['class' => 'middle']) !!}
</div>
<div class="small-7 columns end">
{!! Form::text('caregiver_middle_name', $user->patient
    ->caregiver_middle_name,
['class' => 'default', 'id' => 'caregiver_middle_name'
    ]) !!}
@foreach ($errors->get('caregiver_middle_name') as $key
    => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('caregiver_middle_name'
    )) - 1)<br>@endif
@endforeach
</div>
</div>
<!-- Caregiver\'s Last Name Form Input -->
<div class="row">
<div class="small-5 columns">
{!! Form::label('caregiver_last_name', 'Caregiver\'s
    Last Name', ['class' => 'middle']) !!}
</div>
<div class="small-7 columns end">
{!! Form::text('caregiver_last_name', $user->patient->
    caregiver_last_name,
['class' => 'default', 'id' => 'caregiver_last_name'])
    !!}
@foreach ($errors->get('caregiver_last_name') as $key
    => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('caregiver_last_name'))
    - 1)<br>@endif
@endforeach
</div>
</div>
@elseif ($is_staff)
@if ($user->medicalProfessional)
<!-- Specialties Form Input -->
<div class="row">
<div class="small-5 columns">
{!! Form::label('specialties', 'Specialties', ['class'
    => 'middle']) !!}
</div>
<div class="small-7 columns end">
{!! Form::text('specialties', $user->
    medicalProfessional->specialties,
['class' => 'default', 'id' => 'specialties']) !!}
@foreach ($errors->get('specialties') as $key => $error
    )
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('specialties')) - 1)<br
    >@endif
@endforeach
</div>
</div>
@endif
<!-- Clinic Schedule Form Input -->
<div class="row">
<div class="small-5 columns">
{!! Form::label('clinic_schedule', 'Clinic Schedule', [
    'class' => 'middle']) !!}
</div>
<div class="small-7 columns end">
{!! Form::textarea('clinic_schedule', $clinic_schedule,
['class' => 'default', 'id' => 'clinic_schedule', 'rows
```

```
    ' => '3',
'placeholder' => 'Input your schedule in your clinic (e
    .g., Monday: 10:00am to '
. '3:00pm, Tuesday and Thursday: 11:00am to 2:00pm,
    ...).']) !!}
</div>
</div>
@endif
<!-- Password Form Input -->
<div class="row">
<div class="small-5 columns">
{!! Form::label('password', 'Input password to save
    changes*',
['class' => 'middle', 'for' => 'password']) !!}
</div>
<div class="small-7 columns end">
{!! Form::password('password', ['class' => 'default', '
    id' => 'password']) !!}
@foreach ($errors->get('password') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('password')) - 1)<br>
    @endif
@endforeach
</div>
</div>
<p class="float-left help-text">Fields marked with *
    are mandatory.</p>
<!-- Submit Button -->
<div class="row">
<div class="small-12 center column end">
{!! Form::submit($submitButtonText, ['class' => 'button
    ']) !!}
<a class="hollow button" href="{{ url('profile') }}">
    Cancel</a>
</div>
</div>
```

General Account Display View

```
@extends('app')
@section('css')@endsection
@section('content')
<div class="row margin-bottom-10">
<div class="small-10 small-centered medium-10 medium-
    centered large-10 large-centered columns">
<h4 class="text-center">Account Profile</h4>
<hr>
<div class="row column margin-bottom-20 menu-centered">
<ul class="vertical medium-horizontal menu">
<li class="menu-text">Account Profile</li>
<li>
<a href="{{ url('settings/general') }}">Change General
    Account Settings</a>
</li>
<li>
<a href="{{ url('settings/profile-picture') }}">Change
    Profile Picture</a>
</li>
<li><a href="{{ url('settings/password') }}">Change
    Password</a></li>
</ul>
</div>
<div class="row">
<div class="medium-6 columns center margin-bottom-10">
@if ($user->file) <img src="{{ url($user->file->path . '
    /' . $user->file->filename) }}" width="250rem">
@else <img src="{{ url('files/profile_pictures/anon-
    profile.jpg') }}" width="250rem"> @endif
</div>
<div class="medium-6 columns margin-bottom-10">
<div class="row margin-bottom-10">
<div class="medium-5 columns">Account Type:</div>
<div class="medium-7 columns end">
@if ($user->patient) Patient
@else
@if ($user->medicalProfessional) Medical Professional
@elseif ($user->nutritionist) Nutritionist
@elseif ($user->physiatrist) Rehabilitation Medicine
    Doctor
@else <em>n/a</em>
@endif
@endif
</div>
</div>
@if ($user->systemAdmin)
<div class="row margin-bottom-10">
<div class="medium-5 columns end">System Administrator
    </div>
</div>
@endif
@if ($user->localAdmin)
<div class="row margin-bottom-10">
<div class="medium-5 columns">Hospital Administrator:</
    div>
<div class="medium-7 columns end">
@if ($user->localAdmin->hospital)
<a href="{{ url('hospitals/' . $user->localAdmin->
    hospital->id) }}">
{{ $user->localAdmin->hospital->name }}
</a>
@else <em>n/a</em>
@endif
</div>
</div>
@endif
@if ($user->supervisor)
<div class="row margin-bottom-10">
<div class="medium-5 columns">Supervisor:</div>
<div class="medium-7 columns end">
@if ($user->supervisor->forumCategory) {{ $user->
    supervisor->forumCategory->name }}
@else <em>n/a</em>
@endif
</div>
</div>
@endif
@if ($user->patient)
<div class="row margin-bottom-10">
<div class="medium-5 columns">Hospital ID Number:</div>
<div class="medium-7 columns end">
```

```
@if ($user->patient->hospital_id_number) {{ $user->
    patient->hospital_id_number }}
@else <em>n/a</em> @endif
</div>
</div>
@elseif ($is_staff)
@if ($user->medicalProfessional)
<div class="row margin-bottom-10">
<div class="medium-5 columns">Specialties:</div>
<div class="medium-7 columns end">
@if ($user->medicalProfessional->specialties) {{ $user
    ->medicalProfessional->specialties }}
@else <em>n/a</em> @endif
</div>
</div>
@endif
<div class="row margin-bottom-10">
<div class="medium-5 columns">Clinic Schedule:</div>
<div class="medium-7 columns end">
@if ($clinic_schedule) {{ $clinic_schedule }} @else <em
    >n/a</em> @endif
</div>
</div>
@endif
<div class="row margin-bottom-10">
<div class="medium-5 columns">Username:</div>
<div class="medium-7 columns end">
@if ($user->username) {{ $user->username }} @else <em>n
    /a</em> @endif
</div>
</div>
<div class="row margin-bottom-10">
<div class="medium-5 columns">E-mail Address:</div>
<div class="medium-7 columns end">
@if ($user->email) {{ $user->email }} @else <em>n/a</em
    > @endif
</div>
</div>
<div class="row margin-bottom-10">
<div class="medium-5 columns">First Name:</div>
<div class="medium-7 columns end">
@if ($user->first_name) {{ $user->first_name }} @else <
    em>n/a</em> @endif
</div>
</div>
<div class="row margin-bottom-10">
<div class="medium-5 columns">Middle Name:</div>
<div class="medium-7 columns end">
@if ($user->middle_name) {{ $user->middle_name }} @else
     <em>n/a</em> @endif
</div>
</div>
<div class="row margin-bottom-10">
<div class="medium-5 columns">Last Name:</div>
<div class="medium-7 columns end">
@if ($user->last_name) {{ $user->last_name }} @else <em
    >n/a</em> @endif
</div>
</div>
<div class="row margin-bottom-10">
<div class="medium-5 columns">Sex:</div>
<div class="medium-7 columns end">
@if ($user->sex) {{ ucfirst($user->sex) }} @else <em>n/
    a</em> @endif
</div>
</div>
<div class="row margin-bottom-10">
<div class="medium-5 columns">Birthday:</div>
<div class="medium-7 columns end">
@if ($user->birthday[1]) {{ $user->birthday[1] }} @else
     <em>n/a</em> @endif
</div>
</div>
@if ($user->patient)
<div class="row margin-bottom-10">
<div class="medium-5 columns">Caregiver's First Name:</
    div>
<div class="medium-7 columns end">
@if ($user->patient->caregiver_first_name) {{ $user->
    patient->caregiver_first_name }}
@else <em>n/a</em> @endif
</div>
</div>
<div class="row margin-bottom-10">
<div class="medium-5 columns">Caregiver's Middle Name
    :</div>
<div class="medium-7 columns end">
@if ($user->patient->caregiver_middle_name) {{ $user->
    patient->caregiver_middle_name }}
@else <em>n/a</em> @endif
</div>
</div>
<div class="row margin-bottom-10">
<div class="medium-5 columns">Caregiver's Last Name:</
    div>
<div class="medium-7 columns end">
@if ($user->patient->caregiver_last_name) {{ $user->
    patient->caregiver_last_name }}
@else <em>n/a</em> @endif
</div>
</div>
@if (isset($medical_professional))
<div class="row margin-bottom-10">
<div class="medium-5 columns">Medical Professional:</
    div>
<div class="medium-7 columns end">
<a href="{{ url('users/' .
    $medical_professional_username) }}">{{
    $medical_professional }}</a>
</div>
</div>
@endif
@if (isset($nutritionist))
<div class="row margin-bottom-10">
<div class="medium-5 columns">Nutritionist:</div>
<div class="medium-7 columns end">
<a href="{{ url('users/' . $nutritionist_username)
    }}">{{ $nutritionist }}</a>
</div>
</div>
```

```
@endif
@if (isset($physiatrist))
<div class="row margin-bottom-10">
<div class="medium-5 columns">Rehabilitation Medicine
    Doctor:</div>
<div class="medium-7 columns end">
<a href="{{ url('users/' . $physiatrist username)
    }}">{{ $physiatrist }}</a>
</div>
</div>
@endif
<div class="row margin-bottom-10">
<div class="medium-5 columns">Hospital/Clinic:</div>
<div class="medium-7 columns end">
@if ($user->patient->hospital)
<a href="{{ url('hospitals/' . $user->patient->hospital
    ->id) }}">
{{ $user->patient->hospital->name }}
</a>
@else <em>n/a</em> @endif
</div>
</div>
@endif
</div>
</div>
</div>
</div>
@endsection
@section('js')@endsection
```

General Account Edit View

```
@extends('app')
@section('css')@endsection
@section('content')
<div class="row margin-bottom">
<div class="small-10 small-centered medium-10 medium-
    centered large-10 large-centered columns">
<h4 class="text-center">General Account Settings</h4>
<hr>
<div class="row column margin-bottom-20 menu-centered">
<ul class="vertical medium-horizontal menu">
<li><a href="{{ url('profile') }}">Account Profile</a
    ></li>
<li class="menu-text">Change General Account Settings</
    li>
<li>
<a href="{{ url('settings/profile-picture') }}">Change
    Profile Picture</a>
</li>
<li><a href="{{ url('settings/password') }}">Change
    Password</a></li>
</ul>
</div>
{!! Form::model($user, ['method' => 'PATCH', 'url' => [
    'settings']]) !!}
@include ('auth.accounts.general._form', ['
    submitButtonText' => 'Save Changes'])
{!! Form::close() !!}
</div>
</div>
@endsection
@section('js')@endsection
```

Account Passwords Form View

```
<!-- Old Password Form Input -->
<div class="row">
<div class="small-5 columns">
{!! Form::label('old_password', 'Old_Password*',
['class' => 'middle', 'for' => 'old_password']) !!}
</div>
<div class="small-7 columns end">
{!! Form::password('old_password',['id' => '
    old_password', 'class' => 'default']) !!}
@foreach ($errors->get('old_password') as $key =>
    $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('old_password')) - 1)<
    br>@endif
@endforeach
</div>
</div>
<!-- New Password Form Input -->
<div class="row">
<div class="small-5 columns">
{!! Form::label('new_password', 'New_Password*',
['class' => 'middle', 'for' => 'new_password']) !!}
</div>
<div class="small-7 columns end">
{!! Form::password('new_password',['id' => '
    new_password', 'class' => 'default']) !!}
@foreach ($errors->get('new_password') as $key =>
    $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('new_password')) - 1)<
    br>@endif
@endforeach
</div>
</div>
<!-- Confirm Password Form Input -->
<div class="row">
<div class="small-5 columns">
{!! Form::label('new_password_confirmation', 'Confirm
    Password*',
['class' => 'middle', 'for' => '
    new_password_confirmation']) !!}
</div>
<div class="small-7 columns end">
{!! Form::password('new_password_confirmation',['id' =>
     'new_password_confirmation',
'class' => 'default']) !!}
@foreach ($errors->get('new_password_confirmation') as
    $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('
    new_password_confirmation')) - 1)<br>@endif
@endforeach
</div>
</div>
```

```
<p class="float-left help-text">Fields marked with *
    are mandatory.</p>
<!-- Submit Button -->
<div class="row">
<div class="small-12 center column end">
{!! Form::submit($submitButtonText, ['class' => 'button
    ']) !!}
<a class="hollow button" href="{{ url('profile') }}">
    Cancel</a>
</div>
</div>
```

Account Passwords Change View

```
@extends('app')
@section('css')@endsection
@section('content')
<div class="row margin-bottom">
<div class="small-10 small-centered medium-10 medium-
    centered large-10 large-centered columns">
<h4 class="text-center">Change Password</h4>
<hr>
<div class="row column margin-bottom-20 menu-centered">
<ul class="vertical medium-horizontal menu">
<li><a href="{{ url('profile') }}">Account Profile</a
    ></li>
<li>
<a href="{{ url('settings/general') }}">Change General
    Account Settings</a>
</li>
<li>
<a href="{{ url('settings/profile-picture') }}">Change
    Profile Picture</a>
</li>
<li class="menu-text">Change Password</li>
</ul>
</div>
{!! Form::open(['method' => 'POST', 'url' => ['settings
    /password']]) !!}
@include ('auth.accounts.passwords._form', ['
    submitButtonText' => 'Save Changes'])
{!! Form::close() !!}
</div>
</div>
@endsection
@section('js')@endsection
```

Account Profile Picture Script View

```
<script type="application/javascript">
$(document).ready(function() {
$('input[type=file]').change(function(){
$(this).parent().next().find('input[type="text"]').val(
    $(this).val().split('\\').pop());
});
});
</script>
```

Account Profile Picture Upload View

```
@extends('app')
@section('css')@endsection
@section('content')
<div class="row margin-bottom">
<div class="small-10 small-centered medium-10 medium-
    centered large-10 large-centered columns">
<h4 class="text-center">Profile Picture</h4>
<hr>
<div class="row column margin-bottom-20 menu-centered">
<ul class="vertical medium-horizontal menu">
<li><a href="{{ url('profile') }}">Account Profile</a
    ></li>
<li>
<a href="{{ url('settings/general') }}">Change General
    Account Settings</a>
</li>
<li class="menu-text">Change Profile Picture</li>
<li><a href="{{ url('settings/password') }}">Change
    Password</a></li>
</ul>
</div>
<div class="row column margin-bottom-10">
<div class="row column center margin-bottom-20">
@if ($user->file)
<img src="{{ url($user->file->path . '/' . $user->file
    ->filename) }}" width="300rem">
@else <img src="{{ url('files/profile_pictures/anon-
    profile.jpg') }}" width="300rem">
@endif
</div>
@if ($user->file)
<div class="row column center margin-bottom-10">
{!! Form::open(['method' => 'DELETE', 'url' => ['
    settings/profile-picture']]) !!}
{!! Form::submit('Delete', ['class' => 'button hollow'
    ]) !!}
{!! Form::close() !!}
</div>
@endif
{!! Form::open(['method' => 'PATCH', 'url' => ['
    settings/profile-picture'], 'files' => true]) !!}
<div class="row column">
<p class="help-text text-center">
Note: Uploading a new profile picture then submitting
    it will delete the previous profile
picture.
</p>
</div>
<div class="row">
<!-- File Upload Form Input -->
<div class="medium-2 medium-offset-2 columns">
<label for="profile_picture" class="button">Upload</
    label>
{!! Form::file('profile_picture', ['id' => '
    profile_picture',
'class' => 'show-for-sr', ]) !!}
</div>
<div class="medium-6 columns end">
{!! Form::text('filename', null, ['id' => 'filename', '
    class' => 'default',
```

```
'disabled' => 'disabled']) !!}
@foreach ($errors->get('profile_picture') as $key =>
    $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('profile_picture')) -
    1)<br>@endif
@endforeach
</div>
</div>
<!-- Submit Button -->
<div class="row column center">
{!! Form::submit('Submit', ['class' => 'button']) !!}
<a href="{{ url('profile') }}" class="button hollow">
    Cancel</a>
</div>
{!! Form::close() !!}
</div>
</div>
</div>
@endsection
@section('js')
@include('auth.accounts.profile_picture._script')
@endsection
```

E-mail Approved Visit View

```
<!DOCTYPE html>
<html lang="en-US">
<head>
<meta charset="utf-8">
</head>
<body>
<h2>Approved Schedule for Follow-Up Visit</h2>
<div>
<p>
Hello {{ $patient_user->first_name }} {{ $patient_user
    ->last_name }}!<br>
The schedule you set, which is on {{ $date_of_visit }},
     is approved by the health professional
{{ $health_professional_user->first_name }} {{
    $health_professional_user->last_name }}.<br>
Login to Philippine Stroke Portal and visit the "
    Schedule Follow-Up Visit" page for more
    information.
</p>
</div>

</body>
</html>
```

E-mail Confirm View

```
<!DOCTYPE html>
<html lang="en-US">
<head>
<meta charset="utf-8">
</head>
<body>
<h2>Confirm your E-mail Address</h2>
<div>
<p>
You're receiving this e-mail because user "{{ $username
    }}" at Philippine Stroke Portal has given your e-
    mail
address to connect their account.<br>
To confirm this is correct, please follow the link
    below.<br>
{{ url('auth/register/confirm/'. $confirmation_code)
    }}
</p>
</div>

</body>
</html>
```

E-mail Password View

```
<!DOCTYPE html>
<html lang="en-US">
<head>
<meta charset="utf-8">
</head>
<body>
<h2>Reset Password</h2>
<div>
<p>
Click here to reset your password in Philippine Stroke
    Portal: {{ url('password/reset/'.$token) }}
</p>
</div>

</body>
</html>
```

Events Create View

```
@extends('app')
@section('css')@endsection
@section('content')
<div class="row">
<div class="small-10 small-centered medium-10 medium-
    centered large-10 large-centered column">
<div class="row column center"><h4>Add an Announcement
    and Event from the {{ $hospital->name }}</h4></div
    >
<hr>
<div class="row column">
{!! Form::open(['method' => 'POST', 'url' => '
    announcements-and-events/hospitals/' . $hospital->
    id .
'/create']) !!}
<div class="row">
<!-- Heading Form Input -->
<div class="medium-6 columns">
<label for="heading">Heading*
{!! Form::text('heading', null, ['class' => 'default',
    'id' => 'heading']) !!}
@foreach ($errors->get('heading') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
```

```
@if ($key == count($errors->get('heading')) - 1)<br>
    @endif
@endforeach
</label>
</div>
<!-- Date of Event Form Input -->
<div class="medium-3 columns">
<label for="date_of_event">Date of Event
{!! Form::date('date_of_event', null, ['class' => '
    default', 'id' => 'date_of_event',
'min' => $current_date->format('Y-m-d')]) !!}
@foreach ($errors->get('date_of_event') as $key =>
    $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('date_of_event')) - 1)<
    br>@endif
@endforeach
</label>
</div>
<!-- Time of Event Form Input -->
<div class="medium-3 columns">
<label for="time_of_event">Time of Event
{!! Form::time('time_of_event', null, ['class' => '
    default', 'id' => 'time_of_event']) !!}
@foreach ($errors->get('time_of_event') as $key =>
    $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('time_of_event')) - 1)<
    br>@endif
@endforeach
</label>
</div>
</div>
<div class="row">
<!-- Description Form Input -->
<div class="medium-12 columns">
<label for="description">Description*
{!! Form::textarea('description', null,
['class' => 'default', 'id' => 'description', 'rows' =>
     '3',
'placeholder' => 'Input the description of the event.'
    ]) !!}
@foreach ($errors->get('description') as $key => $error
    )
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('description')) - 1)<br
    >@endif
@endforeach
</label>
</div>
</div>
<p class="help-text">Fields marked with * are mandatory
    .</p>
<!-- Submit Button -->
<div class="row column">
<div class="center">
{!! Form::submit('Add Announcement and Event', ['class'
     => 'button']) !!}
<a href="{{ url('announcements-and-events/' . $hospital
    ->name) }}" class="button hollow">
Cancel
</a>
</div>
</div>
{!! Form::close() !!}
</div>
</div>
</div>
@endsection
@section('js')@endsection
```

Events Edit View

```
@extends('app')
@section('css')@endsection
@section('content')
<div class="row">
<div class="small-10 small-centered medium-10 medium-
    centered large-10 large-centered column">
<div class="row column center"><h4>Edit an Announcement
     and Event from the {{ $hospital->name }}</h4></
    div>
<hr>
<div class="row column">
{!! Form::open(['method' => 'PATCH', 'url' => '
    announcements-and-events/hospitals/' . $hospital->
    id .
'/' . $event->id . '/edit']) !!}
<div class="row">
<!-- Heading Form Input -->
<div class="medium-6 columns">
<label for="heading">Heading*
{!! Form::text('heading', $event->heading, ['class' =>
    'default', 'id' => 'heading']) !!}
@foreach ($errors->get('heading') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('heading')) - 1)<br>
    @endif
@endforeach
</label>
</div>
<!-- Date of Event Form Input -->
<div class="medium-3 columns">
<label for="date_of_event">Date of Event
{!! Form::date('date_of_event', $event->date_of_event
    [0]->format('Y-m-d'),
['class' => 'default', 'id' => 'date_of_event', 'min'
    => $current_date->format('Y-m-d')]) !!}
@foreach ($errors->get('date_of_event') as $key =>
    $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('date_of_event')) - 1)<
    br>@endif
@endforeach
</label>
</div>
<!-- Time of Event Form Input -->
<div class="medium-3 columns">
<label for="time_of_event">Time of Event
{!! Form::time('time_of_event', $event->time_of_event
```

```
    [0], ['class' => 'default',
'id' => 'time_of_event']) !!}
@foreach ($errors->get('time_of_event') as $key =>
    $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('time_of_event')) - 1)<
    br>@endif
@endforeach
</label>
</div>
</div>
<div class="row">
<!-- Description Form Input -->
<div class="medium-12 columns">
<label for="description">Description*
{!! Form::textarea('description', $event->description,
['class' => 'default', 'id' => 'description', 'rows' =>
     '3',
'placeholder' => 'Input the description of the event.'
    ]) !!}
@foreach ($errors->get('description') as $key => $error
    )
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('description')) - 1)<br
    >@endif
@endforeach
</label>
</div>
</div>
<p class="help-text">Fields marked with * are mandatory
    .</p>
<!-- Submit Button -->
<div class="row column">
<div class="center">
{!! Form::submit('Edit Announcement and Event', ['class
    ' => 'button']) !!}
<a href="{{ url('announcements-and-events/hospitals' .
    $hospital->id . '/' . $event->id) }}"
class="button hollow">
Cancel
</a>
</div>
</div>
{!! Form::close() !!}
</div>
</div>
</div>
@endsection
@section('js')@endsection
```

Events Show View

```
@extends('app')
@section('css')@endsection
@section('content')
<div class="row">
<div class="small-10 small-centered medium-10 medium-
    centered large-10 large-centered column">
<div class="center"><h4>{{ $event->heading }}</h4></div
    >
<hr>
<div class="row">
<div class="medium-12 column margin-bottom-10 end">
<div class="callout margin-bottom-20">
@if ($event->hospital->localAdmin)
@if ($event->hospital->localAdmin->user)
<div class="row margin-bottom-10">
<div class="medium-6 columns">
<div class="profile-card" id="rem-3">
@if ($event->hospital->localAdmin->user->file)
<img src="{{ url($event->hospital->localAdmin->user->
    file->path . '/' .
$event->hospital->localAdmin->user->file->filename) }}"
    >
@else
<img src="{{ url('files/profile_pictures/anon-profile.
    jpg') }}">
@endif
<div class="padding-top-profile-picture">
<a href="{{ url('users/' . $event->hospital->localAdmin
    ->user->username) }}">
{{ $event->hospital->localAdmin->user->last_name }},
{{ $event->hospital->localAdmin->user->first_name }}
</a>
</div>
</div>
</div>
<div class="medium-6 columns padding-top">
Posted on: {{ $event->created_at->format('F d, Y') }}
</div>
</div>
@endif
@endif
<p>
Date of Event: {{ $event->date_of_event[1] }} &
    nbsp;   
Time of Event: {{ $event->time_of_event[1] }}
</p>
<p>{{ $event->description }}</p>
<a href="{{ url('announcements-and-events/' . $event->
    hospital->name) }}">
{{ $event->hospital->name }}
</a>
</div>
</div>
</div>
<div class="row column margin-bottom-20 menu-centered">
<ul class="vertical medium-horizontal menu ul-margin">
@if ($is_local_admin_and_hospital)
<li>
<a href="{{ url('announcements-and-events/hospitals/' .
    $hospital->id . '/' . $event->id .
'/edit') }}" class="button">
Edit
</a>
</li>
<li>
{!! Form::open(['method' => 'DELETE', 'url' => '
    announcements-and-events/hospitals/' .
$hospital->id . '/' . $event->id . '/delete']) !!}
```

```
{!! Form::submit('Delete', ['class' => 'button alert']) !!}
{!! Form::close() !!}
</li>
@endif
<li>
<a href="{{ url('announcements-and-events/' . $hospital->name) }}" class="button hollow">
Return to Announcements and Events in Hospital
</a>
</li>
</ul>
</div>
</div>
</div>
@endsection
@section('js')@endsection
```

Events View All View

```
@extends('app')
@section('css')@endsection
@section('content')
<div class="row">
<div class="small-12 small-centered medium-12 medium-centered large-12 large-centered column">
<div class="center"><h4>Announcements and Events</h4></div>
<hr>
<div class="row">
<div class="medium-12 column margin-bottom-10 end">
@if (count($events) > 0)
@foreach ($events as $event)
@if ($event->hospital)
<div class="callout margin-bottom-20 height-scroll">
@if ($event->hospital->localAdmin)
@if ($event->hospital->localAdmin->user)
<div class="row margin-bottom-10">
<div class="medium-6 columns">
<div class="profile-card" id="rem-3">
@if ($event->hospital->localAdmin->user->file)
<img src="{{ url($event->hospital->localAdmin->user->file->path . '/' . $event->hospital->localAdmin->user->file->filename) }}">
@else
<img src="{{ url('files/profile_pictures/anon-profile.jpg') }}">
@endif
<div class="padding-top-profile-picture">
<a href="{{ url('users/' . $event->hospital->localAdmin->user->username) }}">
{{ $event->hospital->localAdmin->user->last_name }},
{{ $event->hospital->localAdmin->user->first_name }}
</a>
</div>
</div>
</div>
<div class="medium-6 columns padding-top">
Posted on: {{ $event->created_at->format('F d, Y') }}
</div>
</div>
@endif
@endif
<h5>
<a href="{{ url('announcements-and-events/hospitals/' . $event->hospital->id . '/' . $event->id) }}">
{{ $event->heading }}
</a>
</h5>
<p>
Date of Event: {{ $event->date_of_event[1] }}     
Time of Event: {{ $event->time_of_event[1] }}
</p>
<p>{{ $event->description }}</p>
<a href="{{ url('announcements-and-events/' . $event->hospital->name) }}">
{{ $event->hospital->name }}
</a>
</div>
@endif
@endforeach
{!! $page_links !!}
@else
There are currently no announcements and events.
@endif
</div>
</div>
</div>
</div>
@endsection
@section('js')@endsection
```

Events View in Hospital View

```
@extends('app')
@section('css')@endsection
@section('content')
<div class="row">
<div class="small-12 small-centered medium-12 medium-centered large-12 large-centered column">
<div class="center"><h4>Announcements and Events of the {{ $hospital->name }}</h4></div>
<hr>
<div class="row column margin-bottom-20 menu-centered">
<ul class="vertical medium-horizontal menu ul-margin">
<li><a href="{{ url('hospitals/' . $hospital->id) }}">View Hospital</a></li>
<li class="menu-text">Hospital Announcements and Events</li>
</ul>
</div>
<div class="row">
<div class="medium-12 column margin-bottom-10 end">
@if (count($events) > 0)
@foreach ($events as $event)
@if ($event->hospital)
<div class="callout margin-bottom-20 height-scroll">
```

```
@if ($event->hospital->localAdmin)
@if ($event->hospital->localAdmin->user)
<div class="row margin-bottom-10">
<div class="medium-6 columns">
<div class="profile-card" id="rem-3">
@if ($event->hospital->localAdmin->user->file)
<img src="{{ url($event->hospital->localAdmin->user->
    file->path . '/' .
$event->hospital->localAdmin->user->file->filename) }}"
    >
@else
<img src="{{ url('files/profile_pictures/anon-profile.
    jpg') }}">
@endif
<div class="padding-top-profile-picture">
<a href="{{ url('users/' . $event->hospital->localAdmin
    ->user->username) }}">
{{ $event->hospital->localAdmin->user->last_name }},
{{ $event->hospital->localAdmin->user->first_name }}
</a>
</div>
</div>
</div>
<div class="medium-6 columns padding-top">
Posted on: {{ $event->created_at->format('F d, Y') }}
</div>
</div>
@endif
@endif
<h5>
<a href="{{ url('announcements-and-events/hospitals/' .
    $event->hospital->id . '/' .
$event->id) }}">
{{ $event->heading }}
</a>
</h5>
<p>
Date of Event: {{ $event->date_of_event[1] }} &
    nbsp;   
Time of Event: {{ $event->time_of_event[1] }}
</p>
<p>{{ $event->description }}</p>
</div>
@endif
@endforeach
{!! $page_links !!}
@else
There are currently no announcements and events in this
     hospital.
@endif
<div class="row column margin-bottom-20 menu-centered">
<ul class="vertical medium-horizontal menu ul-margin">
@if ($is_local_admin_and_hospital)
<li>
<a href="{{ url('announcements-and-events/hospitals/' .
    $hospital->id . '/create') }}"
class="button">
Add an Announcement and Event
</a>
</li>
@endif
<li>
<a href="{{ url('announcements-and-events') }}" class="
    button hollow">Cancel</a>
</li>
</ul>
</div>
</div>
</div>
</div>
</div>
@endsection
@section('js')@endsection
```

Forum Threads Create View

```
@extends('app')
@section('css')@endsection
@section('content')
<div class="row">
<div class="small-10 small-centered medium-10 medium-
    centered large-10 large-centered column">
<div class="center"><h4>Add Forum Thread</h4></div>
<hr>
<div class="row column margin-bottom-20 menu-centered">
<ul class="vertical medium-horizontal menu">
<li><a href="{{ url('forum-threads/all') }}">View All
    Forum Threads</a></li>
<li>
<a href="{{ url('forum-threads/Neurology') }}">
View Forum Threads on Neurology
</a>
</li>
<li>
<a href="{{ url('forum-threads/Nutrition') }}">
View Forum Threads on Nutrition
</a>
</li>
<li>
<a href="{{ url('forum-threads/Rehabilitation%20
    Medicine') }}">
View Forum Threads on Rehabilitation Medicine
</a>
</li>
</ul>
</div>
<div class="row column margin-bottom-10">
{!! Form::model($forum = new \App\Models\Forum, ['
    method' => 'POST', 'url' => 'forum-threads']) !!}
<div class="row">
<!-- Title Form Input -->
<div class="medium-8 columns">
<label for="title">Title*
{!! Form::text('title', null, ['class' => 'default', '
    id' => 'title']) !!}
@foreach ($errors->get('title') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('title')) - 1)<br>
    @endif
@endforeach
```

```
</label>
</div>
<!-- Forum Category Form Input -->
<div class="medium-4 columns">
<label for="forum_category">Forum Category*
{!! Form::select('forum_category', $forum_categories,
    null,
['class' => 'default', 'id' => 'forum_category']) !!}
@foreach ($errors->get('forum_category') as $key =>
    $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('forum_category')) - 1)
    <br>@endif
@endforeach
</label>
</div>
</div>
<div class="row">
<!-- Description Form Input -->
<div class="medium-12 columns">
<label for="description">Description*
{!! Form::textarea('description', null,
['class' => 'default', 'id' => 'description', 'rows' =>
     '3',
'placeholder' => 'Input the details about this forum
    post.']) !!}
@foreach ($errors->get('description') as $key => $error
    )
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('description')) - 1)<br
    >@endif
@endforeach
</label>
</div>
</div>
<p class="help-text">Fields marked with * are mandatory
    .</p>
<div class="row">
<!-- Submit Button -->
<div class="row column center">
{!! Form::submit('Add Forum Thread', ['class' => '
    button']) !!}
<a href="{{ url('forum-threads/all') }}" class="button
    hollow">Cancel</a>
</div>
</div>
{!! Form::close() !!}
</div>
</div>
</div>
@endsection
@section('js')@endsection
```

Forum Threads Show View

```
@extends('app')
@section('css')@endsection
@section('content')
<div class="row">
<div class="small-10 small-centered medium-10 medium-
    centered large-10 large-centered column">
<div class="center"><h4>{{ ucfirst($forum->title) }}</
    h4></div>
<hr>
<div class="row column margin-bottom-20 menu-centered">
<ul class="vertical medium-horizontal menu">
<li><a href="{{ url('forum-threads/all') }}">View All
    Forum Threads</a></li>
<li>
<a href="{{ url('forum-threads/Neurology') }}">
View Forum Threads on Neurology
</a>
</li>
<li>
<a href="{{ url('forum-threads/Nutrition') }}">
View Forum Threads on Nutrition
</a>
</li>
<li>
<a href="{{ url('forum-threads/Rehabilitation%20
    Medicine') }}">
View Forum Threads on Rehabilitation Medicine
</a>
</li>
</ul>
</div>
<div class="row column margin-bottom-10">
<div class="callout margin-bottom-10">
@if ($forum->user)
<div class="row margin-bottom-10">
<div class="medium-6 columns">
<div class="profile-card" id="rem-3">
@if ($forum->user->file)
<img src="{{ url($forum->user->file->path . '/' .
$forum->user->file->filename) }}">
@else
<img src="{{ url('files/profile_pictures/anon-profile.
    jpg') }}">
@endif
<div class="padding-top-profile-picture">
<a href="{{ url('users/' . $forum->user->username) }}">
{{ $forum->user->last_name }},
{{ $forum->user->first_name }}
</a>
</div>
</div>
</div>
<div class="medium-6 columns padding-top">
Posted on: {{ $forum->created_at->format('F d, Y') }}
</div>
</div>
@endif
@if ($forum->forumCategory)
<p>
Category:
<a href="{{ url('forum-threads/' . $forum->
    forumCategory->name) }}">
{{ $forum->forumCategory->name }}
</a>
</p>
```

```
@endif
<p>
Moderator:
@if ($forum->supervisor)
@if ($forum->supervisor->user)
<a href="{{ url('users/'. $forum->user->username) }}">
{{ $forum->supervisor->user->last_name }},
{{ $forum->supervisor->user->first_name }}
</a>
@else This forum thread has no moderator yet.
@endif
@else This forum thread has no moderator yet.
@endif
</p>
<p>
{{ $forum->description }}
</p>
</div>
<div class="callout">
<div class="margin-bottom-10"><h5>Replies</h5></div>
@foreach ($forum_comments as $forum_comment)
@if ($forum_comment->user)
<div class="row margin-bottom-10">
<div class="medium-6 columns">
<div class="profile-card" id="rem-3">
@if ($forum_comment->user->file)
<img src="{{ url($forum_comment->user->file->path . '/'
    .
$forum_comment->user->file->filename) }}">
@else
<img src="{{ url('files/profile_pictures/anon-profile.
    jpg') }}">
@endif
<div class="padding-top-profile-picture">
<a href="{{ url('users/'. $forum_comment->user->
    username) }}">
{{ $forum_comment->user->last_name }},
{{ $forum_comment->user->first_name }}
</a>
</div>
</div>
</div>
<div class="medium-3 columns @if (! $is_supervisor or !
    $can_moderate) end @endif padding-top">
{{ $forum_comment->created_at->diffForHumans() }}
</div>
@if ($is_supervisor and $can_moderate)
<div class="medium-3 columns">
{!! Form::open(['method' => 'DELETE', 'url' => 'forum-
    threads/' .
$forum->forumCategory->name . '/' . $forum->id . '/' .
    $forum_comment->id]) !!}
{!! Form::submit('Remove Reply', ['class' => 'button
    hollow']) !!}
{!! Form::close() !!}
</div>
@endif
</div>
@endif
<div class="row">
<div class="medium-10 columns">
<p>{{ $forum_comment->comment }}</p>
</div>
</div>
@endforeach
@if (Auth::check())
<div class="row column margin-top-20">
{!! Form::open(['method' => 'POST', 'url' => 'forum-
    threads/' . $forum->forumCategory->name .
'/' . $forum->id]) !!}
<div class="row">
<!-- Reply Form Input -->
<div class="medium-8 columns">
<label for="comment">
{!! Form::textarea('comment', null,
['class' => 'default', 'id' => 'comment', 'rows' => '2'
    ,
'placeholder' => 'Input you reply here.']) !!}
</label>
</div>
</div>
<!-- Submit Button -->
<div class="row column">
{!! Form::submit('Add Reply', ['class' => 'button'])
    !!}
</div>
{!! Form::close() !!}
</div>
@endif
</div>
<div class="row column margin-bottom-20 menu-centered">
<ul class="vertical medium-horizontal menu ul-margin">
@if ($is_supervisor and ! $forum->supervisor)
<li>
{!! Form::open(['method' => 'POST', 'url' => 'forum-
    threads/' . $forum->forumCategory->name .
'/' . $forum->id . '/moderate']) !!}
{!! Form::submit('Moderate Forum Thread', ['class' => '
    button']) !!}
{!! Form::close() !!}
</li>
@endif
@if ($is_supervisor and $can_moderate)
<li>
{!! Form::open(['method' => 'DELETE', 'url' => 'forum-
    threads/' . $forum->forumCategory->name .
'/' . $forum->id . '/delete']) !!}
{!! Form::submit('Delete Forum Thread', ['class' => '
    button alert']) !!}
{!! Form::close() !!}
</li>
@endif
</ul>
</div>
</div>
</div>
</div>
@endsection
@section('js')@endsection
```

Forum Threads View All View

```
@extends('app')
@section('css')@endsection
@section('content')
<div class="row">
<div class="small-12 small-centered medium-12 medium-
    centered large-12 large-centered column">
<div class="center"><h4>Forum Threads</h4></div>
<hr>
<div class="row column margin-bottom-20 menu-centered">
<ul class="vertical medium-horizontal menu">
<li class="menu-text">View All Forum Threads</li>
<li>
<a href="{{ url('forum-threads/Neurology') }}">
View Forum Threads on Neurology
</a>
</li>
<li>
<a href="{{ url('forum-threads/Nutrition') }}">
View Forum Threads on Nutrition
</a>
</li>
<li>
<a href="{{ url('forum-threads/Rehabilitation%20
    Medicine') }}">
View Forum Threads on Rehabilitation Medicine
</a>
</li>
</ul>
</div>
<div class="row column margin-bottom-10">
@if (count($forums) > 0)
@foreach ($forums as $forum)
<div class="callout margin-bottom-10 height-scroll @if
    (! $forum->supervisor) secondary @endif">
@if ($forum->user)
<div class="row margin-bottom-10">
<div class="medium-6 columns">
<div class="profile-card" id="rem-3">
@if ($forum->user->file)
<img src="{{ url($forum->user->file->path . '/' .
$forum->user->file->filename) }}">
@else
<img src="{{ url('files/profile_pictures/anon-profile.
    jpg') }}">
@endif
<div class="padding-top-profile-picture">
<a href="{{ url('users/' . $forum->user->username) }}">
{{ $forum->user->last_name }},
{{ $forum->user->first_name }}
</a>
</div>
</div>
</div>
<div class="medium-6 columns padding-top">
Posted on: {{ $forum->created_at->format('F d, Y') }}
</div>
</div>
@endif
<h5>
<a href="{{ url('forum-threads/' . $forum->
    forumCategory->name . '/' . $forum->id) }}">
{{ $forum->title }}
</a>
</h5>
@if ($forum->forumCategory)
<p>
Category:
<a href="{{ url('forum-threads/' . $forum->
    forumCategory->name) }}">
{{ $forum->forumCategory->name }}
</a>
</p>
@endif
<p>
Moderator:
@if ($forum->supervisor)
@if ($forum->supervisor->user)
<a href="{{ url('users/' . $forum->user->username) }}">
{{ $forum->supervisor->user->last_name }},
{{ $forum->supervisor->user->first_name }}
</a>
@else This forum thread has no moderator yet.
@endif
@else This forum thread has no moderator yet.
@endif
</p>
<p>
{{ $forum->description }}
</p>
</div>
@endforeach
{!! $page_links !!}
@else
There are currently no forum threads.
@endif
<div class="row column center">
@if (Auth::check())
<a href="{{ url('forum-threads/create') }}" class="
    button">Add a Forum Thread</a>
@endif
</div>
</div>
</div>
</div>
@endsection
@section('js')@endsection
```

Forum Threads View in Category View

```
@extends('app')
@section('css')@endsection
@section('content')
<div class="row">
<div class="small-12 small-centered medium-12 medium-
    centered large-12 large-centered column">
<div class="center"><h4>Forum Threads under "{{
    $forum_category->name }}"</h4></div>
<hr>
<div class="row column margin-bottom-20 menu-centered">
```

```
<ul class="vertical medium-horizontal menu">
<li><a href="{{ url('forum-threads/all') }}">View All
    Forum Threads</a></li>

@if (strcmp($category, 'Neurology') == 0)
<li class="menu-text">View Forum Threads on Neurology</
    li>
@else
<li>
<a href="{{ url('forum-threads/Neurology') }}">
View Forum Threads on Neurology
</a>
</li>
@endif

@if (strcmp($category, 'Nutrition') == 0)
<li class="menu-text">View Forum Threads on Nutrition</
    li>
@else
<li>
<a href="{{ url('forum-threads/Nutrition') }}">
View Forum Threads on Nutrition
</a>
</li>
@endif

@if (strcmp($category, 'Rehabilitation Medicine') == 0)
<li class="menu-text">View Forum Threads on
    Rehabilitation Medicine</li>
@else
<li>
<a href="{{ url('forum-threads/Rehabilitation%20
    Medicine') }}">
View Forum Threads on Rehabilitation Medicine
</a>
</li>
@endif
</ul>
</div>
<div class="row column margin-bottom-10">
@if (count($forums) > 0)
@foreach ($forums as $forum)
<div class="callout margin-bottom-10 height-scroll @if
    (! $forum->supervisor) secondary @endif ">
@if ($forum->user)
<div class="row margin-bottom-10">
<div class="medium-6 columns">
<div class="profile-card" id="rem-3">
@if ($forum->user->file)
<img src="{{ url($forum->user->file->path . '/' .
$forum->user->file->filename) }}">
@else
<img src="{{ url('files/profile_pictures/anon-profile.
    jpg') }}">
@endif
<div class="padding-top-profile-picture">
<a href="{{ url('users/' . $forum->user->username) }}">
{{ $forum->user->last_name }},
{{ $forum->user->first_name }}
</a>
</div>
</div>
</div>
<div class="medium-6 columns padding-top">
Posted on: {{ $forum->created_at->format('F d, Y') }}
</div>
</div>
@endif
<h5>
<a href="{{ url('forum-threads/' . $forum->
    forumCategory->name . '/' . $forum->id) }}">
{{ $forum->title }}
</a>
</h5>
@if ($forum->forumCategory)
<p>
Category:
<a href="{{ url('forum-threads/' . $forum->
    forumCategory->name) }}">
{{ $forum->forumCategory->name }}
</a>
</p>
@endif
<p>
Moderator:
@if ($forum->supervisor)
@if ($forum->supervisor->user)
<a href="{{ url('users/' . $forum->user->username) }}">
{{ $forum->supervisor->user->last_name }},
{{ $forum->supervisor->user->first_name }}
</a>
@else This forum thread has no moderator yet.
@endif
@else This forum thread has no moderator yet.
@endif
</p>
<p>
{{ $forum->description }}
</p>
</div>
@endforeach
{!! $page_links !!}
@else
There are currently no forum threads.
@endif
<div class="row column center">
@if (Auth::check())
<a href="{{ url('forum-threads/create') }}" class="
    button">Add a Forum Thread</a>
@endif
</div>
</div>
</div>
</div>
@endsection
@section('js')@endsection
```

Home Events View

```
<div class="row">
```

```
<div class="small-12 column">
<h4>Announcements and Events</h4>
<div class="row small-11 column margin-bottom-10 end">
@if (count($events) > 0)
@foreach ($events as $event)
@if ($event->hospital)
<div class="callout margin-bottom-20 height-scroll">
@if ($event->hospital->localAdmin)
@if ($event->hospital->localAdmin->user)
<div class="row margin-bottom-10">
<div class="medium-6 columns">
<div class="profile-card" id="rem-3">
@if ($event->hospital->localAdmin->user->file)
<img src="{{ url($event->hospital->localAdmin->user->
    file->path . '/' .
$event->hospital->localAdmin->user->file->filename) }}"
    >
@else
<img src="{{ url('files/profile_pictures/anon-profile.
    jpg') }}">
@endif
<div class="padding-top-profile-picture">
<a href="{{ url('users/' . $event->hospital->localAdmin
    ->user->username) }}">
{{ $event->hospital->localAdmin->user->last_name }},
{{ $event->hospital->localAdmin->user->first_name }}
</a>
</div>
</div>
</div>
<div class="medium-6 columns padding-top">
Posted on: {{ $event->created_at->format('F d, Y') }}
</div>
</div>
@endif
@endif
<h5>
<a href="{{ url('announcements-and-events/hospitals/' .
    $event->hospital->id . '/' .
$event->id) }}">
{{ $event->heading }}
</a>
</h5>
<p>
Date of Event: {{ $event->date_of_event[1] }} &
    nbsp;   
Time of Event: {{ $event->time_of_event[1] }}
</p>
<p>{{ $event->description }}</p>
<a href="{{ url('announcements-and-events/' . $event->
    hospital->name) }}">
{{ $event->hospital->name }}
</a>
</div>
@endif
@endforeach
@else
There are currently no announcements and events.
@endif
</div>
```

```
</div>
</div>
```

Home Forums View

```
<div class="row">
<div class="small-12 column">
<h4>Latest Forum Threads</h4>
<div class="row small-11 column">
@if (count($forums) > 0)
@foreach ($forums as $forum)
<div class="callout margin-bottom-10 height-scroll">
@if ($forum->user)
<div class="row margin-bottom-10">
<div class="medium-6 columns">
<div class="profile-card" id="rem-3">
@if ($forum->user->file)
<img src="{{ url($forum->user->file->path . '/' .
$forum->user->file->filename) }}">
@else
<img src="{{ url('files/profile_pictures/anon-profile.
    jpg') }}">
@endif
<div class="padding-top-profile-picture">
<a href="{{ url('users/' . $forum->user->username) }}">
{{ $forum->user->last_name }},
{{ $forum->user->first_name }}
</a>
</div>
</div>
</div>
<div class="medium-6 columns padding-top">
Posted on: {{ $forum->created_at->format('F d, Y') }}
</div>
</div>
@endif
<h5>
<a href="{{ url('forum-threads/' . $forum->
    forumCategory->name . '/' . $forum->id) }}">
{{ $forum->title }}
</a>
</h5>
@if ($forum->forumCategory)
<p>
Category:
<a href="{{ url('forum-threads/' . $forum->
    forumCategory->name) }}">
{{ $forum->forumCategory->name }}
</a>
</p>
@endif
<p>
{{ $forum->description }}
</p>
</div>
@endforeach
@else
There are currently no forum threads.
@endif
</div>
```

```
</div>
</div>
```

Home Medical Resources View

```
<div class="row">
<div class="small-12 column">
<h4>Newest Medical Resources</h4>
<div class="row small-11 column">
@if (count($medical_resources) > 0)
@foreach ($medical_resources as $medical_resource)
<div class="callout margin-bottom-10 height-scroll">
@if ($medical_resource->user)
<div class="row column margin-bottom-10">
<div class="profile-card" id="rem-3">
@if ($medical_resource->user->file)
<img src="{{ url('files/profile_pictures/' .
$medical_resource->user->file->filename) }}">
@else
<img src="{{ url('files/profile_pictures/anon-profile.
jpg') }}">
@endif
<div class="padding-top-profile-picture">
<a href="{{ url('users/' . $medical_resource->user->
username) }}">
{{ $medical_resource->user->last_name }},
{{ $medical_resource->user->first_name }}
</a>
</div>
</div>
</div>
@endif
<div class="row column">
<h5>
<a href="{{ url('medical-resources/' .
$medical_resource->id) }}">
{{ $medical_resource->title }}
</a>
</h5>
<p>
{{ $medical_resource->description }}<br>
<span class="font-green">
Authors: {{ $medical_resource->authors }}
</span>
</p>
</div>
<div class="row medium-6 column">
<p>
Medical Resource File:<br>
@if ($medical_resource->file != null)
@if (in_array($medical_resource->file->extension,
$image_extensions)
or in_array($medical_resource->file->extension,
$video_extensions)
or $medical_resource->file->extension == 'pdf')
<a data-open="file-{{ $medical_resource->id }}">
{{ $medical_resource->file->filename }}
</a>
@else
<a href="{{ url('medical-resources/' .
$medical_resource->id . '/download') }}">
{{ $medical_resource->file->filename }}
</a>
@endif
@else <em>n/a</em>
@endif
</p>
@if ($medical_resource->file != null)
@if (in_array($medical_resource->file->extension,
$image_extensions)
or in_array($medical_resource->file->extension,
$video_extensions)
or ($medical_resource->file->extension == 'pdf'))
<div class="reveal large" id="file-{{ $medical_resource
->id }}" data-reveal
data-animation-in="fade-in" data-animation-out="fade-
out">
<h4>{{ $medical_resource->title }}</h4>
<div class="row column center margin-top-20 margin-
bottom-20">
@if (in_array($medical_resource->file->extension,
$image_extensions))
<img src="{{ url($medical_resource->file->path . '/' .
$medical_resource->file->filename) }}">
@elseif (in_array($medical_resource->file->extension,
$video_extensions))
<video height="500rem" controls>
@if ($medical_resource->file->extension == 'mp4')
<source src="{{ url($medical_resource->file->path . '/'
.
$medical_resource->file->filename) }}" type="video/mp4"
>
@elseif ($medical_resource->file->extension == 'webm')
<source src="{{ url($medical_resource->file->path . '/'
.
$medical_resource->file->filename) }}" type="video/webm
">
@elseif ($medical_resource->file->extension == 'ogg')
<source src="{{ url($medical_resource->file->path . '/'
.
$medical_resource->file->filename) }}" type="video/ogg"
>
@endif
Your browser does not support the video tag.
</video>
@else
<object data="{{ url($medical_resource->file->path . '
/' .
$medical_resource->file->filename) }}" type="
application/pdf" width="100%"
height="500rem">
</object>
@endif
</div>
<div class="row column center">
<a href="{{ url('medical-resources/' .
$medical_resource->id . '/download') }}"
class="button">
Download
```

```
</a>
<a class="button hollow" data-close>Close</a>
</div>
<button class="close-button" data-close aria-label="
    Close modal" type="button">
<span aria-hidden="true">&times;</span>
</button>
</div>
@endif
@endif
</div>
</div>
@endforeach
@else
There are currently no medical resources.
@endif
</div>
</div>
</div>
```

Home About Us View

```
@extends('app')
@section('css')@endsection
@section('content')
<div class="row column">
<p>
The Philippine Stroke Portal is a web portal that
    amasses medically validated information regarding
    stroke,
its symptoms, diagnosis, treatments, rehabilitation,
    and other aspects of stroke.
</p>
<p class="indent">
It also functions as an information system that allows
    stroke patients and caregivers to input their
    daily
medical conditions and laboratory results for the
    viewing of their doctors, nutritionists, and
    rehabilitation
medicine doctors. Accordingly, their corresponding
    health professionals can input their daily medical
    ,
nutrition, and rehabilitation medicine records;
    recommended medication, nutrition, and
    rehabilitation medicine
regimens for their patients; and medical resources on
    stroke. This will allow the patients and
    caregivers to
carry out ay-to-day medical visits online.
</p>
<p class="indent">
Likewise, it implements forum threads, which enables
    stroke patients and caregivers to communicate with
     medical
professionals and other patients and caregivers to
    discuss about their condition, other non-medical
    treatments,
and events in various hospitals
</p>
</div>
@endsection
@section('js')@endsection
```

Home View

```
@extends('app')
@section('css')@endsection
@section('content')
<div class="row column">
@include('home._events')
</div>
<br>
<hr>
<br>
<div class="row column">
@include('home._medical_resources')
</div>
<br>
<hr>
<br>
<div class="row column">
@include('home._forums')
</div>
@endsection
@section('js')
<script type="text/javascript">
$('.reveal').foundation();
</script>
@endsection
```

Home Notifications View

```
@extends('app')
@section('css')@endsection
@section('content')
<div class="row">
<div class="small-12 small-centered medium-12 medium-
    centered large-12 large-centered column">
<div class="center">
<h4> @if ($unread_notifications > 0) ({{
    $unread_notifications }}) @endif Notifications</h4
    >
</div>
<hr>
<div class="row">
<div class="medium-12 column margin-bottom-10 end">
@if (count($notifications) > 0)
@foreach ($notifications as $notification)
@if ($notification->link)
<a href="{{ url('notifications/'. $notification->id) }
    }">
<div class="callout margin-bottom-20 height-scroll
@if (!$notification->is_seen) secondary @endif
    notification">
<p>
@if (! $notification->is_seen) Unread Notification:
    @endif {{ $notification->description }}
</p>
</div>
```

```
</a>
@endif
@endforeach
{!! $page_links !!}
@else You currently have no notifications.
@endif
</div>
</div>
</div>
</div>
@endsection
@section('js')@endsection
```

Home Show User View

```
@extends('app')
@section('css')@endsection
@section('content')
<div class="row margin-bottom-10">
<div class="small-10 small-centered medium-10 medium-
    centered large-10 large-centered columns">
<h4 class="text-center">{{ $user->username }} Account
    Profile</h4>
<hr>
<div class="row">
<div class="medium-6 columns center margin-bottom-10">
@if ($user->file)
<img src="{{ url($user->file->path . '/' . $user->file
    ->filename) }}" width="250rem">
@else
<img src="{{ url('files/profile_pictures/anon-profile.
    jpg') }}" width="250rem">
@endif
</div>
<div class="medium-6 columns margin-bottom-10">
<div class="row margin-bottom-10">
<div class="medium-5 columns">Account Type:</div>
<div class="medium-7 columns end">
@if ($user->patient) Patient
@else
@if ($user->medicalProfessional) Medical Professional
@elseif ($user->nutritionist) Nutritionist
@elseif ($user->physiatrist) Rehabilitation Medicine
    Doctor
@else <em>n/a</em>
@endif
@endif
</div>
</div>
@if ($user->medicalProfessional)
<div class="row margin-bottom-10">
<div class="medium-5 columns">Specialties:</div>
<div class="medium-7 columns end">
@if ($user->medicalProfessional->specialties) {{ $user
    ->medicalProfessional->specialties }}
@else <em>n/a</em>
@endif
</div>
</div>
@endif
@if ($user->medicalProfessional)
<div class="row margin-bottom-10">
<div class="medium-5 columns">Clinic Schedule:</div>
<div class="medium-7 columns end">
@if ($user->medicalProfessional->clinic_schedule)
{{ $user->medicalProfessional->clinic_schedule }} @else
    <em>n/a</em> @endif
</div>
</div>
@elseif ($user->nutritionist)
<div class="row margin-bottom-10">
<div class="medium-5 columns">Clinic Schedule:</div>
<div class="medium-7 columns end">
@if ($user->nutritionist->clinic_schedule) {{ $user->
    nutritionist->clinic_schedule }}
@else <em>n/a</em> @endif
</div>
</div>
@elseif ($user->physiatrist)
<div class="row margin-bottom-10">
<div class="medium-5 columns">Clinic Schedule:</div>
<div class="medium-7 columns end">
@if ($user->physiatrist->clinic_schedule) {{ $user->
    physiatrist->clinic_schedule }}
@else <em>n/a</em> @endif
</div>
</div>
@endif
@if ($user->systemAdmin)
<div class="row margin-bottom-10">
<div class="medium-5 columns end">System Administrator
    </div>
</div>
@endif
@if ($user->localAdmin)
<div class="row margin-bottom-10">
<div class="medium-5 columns">Local Administrator:</div
    >
<div class="medium-7 columns end">
@if ($user->localAdmin->hospital)
<a href="{{ url('hospitals/' . $user->localAdmin->
    hospital->id) }}">
{{ $user->localAdmin->hospital->name }}
</a>
@else <em>n/a</em>
@endif
</div>
</div>
@endif
@if ($user->supervisor)
<div class="row margin-bottom-10">
<div class="medium-5 columns">Supervisor:</div>
<div class="medium-7 columns end">
@if ($user->supervisor->forumCategory) {{ $user->
    supervisor->forumCategory->name }}
@else <em>n/a</em>
@endif
</div>
</div>
@endif
```

```
<div class="row margin-bottom-10">
<div class="medium-5 columns">Username:</div>
<div class="medium-7 columns end">
@if ($user->username) {{ $user->username }} @else <em>n
    /a</em> @endif
</div>
</div>
<div class="row margin-bottom-10">
<div class="medium-5 columns">E-mail Address:</div>
<div class="medium-7 columns end">
@if ($user->email) {{ $user->email }} @else <em>n/a</em
    > @endif
</div>
</div>
<div class="row margin-bottom-10">
<div class="medium-5 columns">First Name:</div>
<div class="medium-7 columns end">
@if ($user->first_name) {{ $user->first_name }} @else <
    em>n/a</em> @endif
</div>
</div>
<div class="row margin-bottom-10">
<div class="medium-5 columns">Middle Name:</div>
<div class="medium-7 columns end">
@if ($user->middle_name) {{ $user->middle_name }} @else
     <em>n/a</em> @endif
</div>
</div>
<div class="row margin-bottom-10">
<div class="medium-5 columns">Last Name:</div>
<div class="medium-7 columns end">
@if ($user->last_name) {{ $user->last_name }} @else <em
    >n/a</em> @endif
</div>
</div>
<div class="row margin-bottom-10">
<div class="medium-5 columns">Sex:</div>
<div class="medium-7 columns end">
@if ($user->sex) {{ ucfirst($user->sex) }} @else <em>n/
    a</em> @endif
</div>
</div>
<div class="row margin-bottom-10">
<div class="medium-5 columns">Birthday:</div>
<div class="medium-7 columns end">
@if ($user->birthday[1]) {{ $user->birthday[1] }} @else
     <em>n/a</em> @endif
</div>
</div>
</div>
</div>
</div>
</div>
@endsection
@section('js')@endsection
```

Hospitals Form View

```
<!-- Name Form Input -->
<div class="row">
<div class="small-5 columns">
{!! Form::label('name', 'Name*', ['class' => 'middle',
    'for' => 'name']) !!}
</div>
<div class="small-7 columns end">
{!! Form::text('name', $hospital->name, ['class' => '
    default', 'id' => 'name']) !!}
@foreach ($errors->get('name') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('name')) - 1)<br>@endif
@endforeach
</div>
</div>
<!-- Address Form Input -->
<div class="row">
<div class="small-5 columns">
{!! Form::label('address', 'Address*', ['class' => '
    middle', 'for' => 'address']) !!}
</div>
<div class="small-7 columns end">
{!! Form::text('address', $hospital->address, ['class'
    => 'default', 'id' => 'address']) !!}
@foreach ($errors->get('address') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('address')) - 1)<br>
    @endif
@endforeach
</div>
</div>
<!-- Landline Number Form Input -->
<div class="row">
<div class="small-5 columns">
{!! Form::label('landline_number', 'Landline Number', [
    'class' => 'middle']) !!}
</div>
<div class="small-7 columns end">
{!! Form::text('landline_number', $hospital->
    landline_number,
['class' => 'default', 'id' => 'landline_number']) !!}
@foreach ($errors->get('landline_number') as $key =>
    $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('landline_number')) -
    1)<br>@endif
@endforeach
</div>
</div>
<!-- Mobile Number Form Input -->
<div class="row">
<div class="small-5 columns">
{!! Form::label('mobile_number', 'Mobile Number', ['
    class' => 'middle']) !!}
</div>
<div class="small-7 columns end">
{!! Form::text('mobile_number', $hospital->
    mobile_number,
['class' => 'default', 'id' => 'mobile_number']) !!}
@foreach ($errors->get('mobile_number') as $key =>
    $error)
<small class="custom-form-error">{{ $error }}</small>
```

```
@if ($key == count($errors->get('mobile_number')) - 1)<
    br>@endif
@endforeach
</div>
</div>
<!-- Fax Number Form Input -->
<div class="row">
<div class="small-5 columns">
{!! Form::label('fax_number', 'Fax Number', ['class' =>
     'middle']) !!}
</div>
<div class="small-7 columns end">
{!! Form::text('fax_number', $hospital->fax_number,
['class' => 'default', 'id' => 'fax_number']) !!}
@foreach ($errors->get('fax_number') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('fax_number')) - 1)<br>
    @endif
@endforeach
</div>
</div>
<!-- E-mail Address Form Input -->
<div class="row">
<div class="small-5 columns">
{!! Form::label('email_address', 'E-mail Address', ['
    class' => 'middle']) !!}
</div>
<div class="small-7 columns end">
{!! Form::text('email_address', $hospital->
    email_address,
['class' => 'default', 'id' => 'email_address']) !!}
@foreach ($errors->get('email_address') as $key =>
    $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('email_address')) - 1)<
    br>@endif
@endforeach
</div>
</div>
<div class="row column">
<p class="help-text">Fields marked with * are mandatory
    .</p>
</div>
<!-- Submit Button -->
<div class="row">
<div class="small-12 center column end">
{!! Form::submit($submitButtonText, ['class' => 'button
    ']) !!}
@if ($hospital)
<a href="{{ url('hospitals/'. $hospital->id) }}" class
    ="button hollow">Cancel</a>
@else
<a href="{{ url('hospitals') }}" class="button hollow">
    Cancel</a>
@endif
</div>
</div>
```

Hospitals Create View

```
@extends('app')
@section('css')@endsection
@section('content')
<div class="row">
<div class="small-10 small-centered medium-10 medium-
    centered large-10 large-centered column">
<div class="row column center"><h4>Add a new Hospital
    or Clinic Record</h4></div>
<hr>
<div class="row">
<div class="medium-8 medium-offset-2 columns end">
{!! Form::model($hospital = new \App\Models\Hospital, [
    'url' => 'hospitals']) !!}
@include ('hospitals._form', ['submitButtonText' => '
    Add Hospital or Clinic Record'])
{!! Form::close() !!}
</div>
</div>
</div>
</div>
@endsection
@section('js')@endsection
```

Hospitals Edit View

```
@extends('app')
@section('css')@endsection
@section('content')
<div class="row">
<div class="small-10 small-centered medium-10 medium-
    centered large-10 large-centered column">
<div class="row column center"><h4>Add a new Hospital
    or Clinic Record</h4></div>
<hr>
<div class="row">
<div class="medium-8 medium-offset-2 columns end">
{!! Form::model($hospital, ['method' => 'PATCH', 'url'
    => 'hospitals/' . $hospital->id]) !!}
@include ('hospitals._form', ['submitButtonText' => '
    Edit Hospital or Clinic Record'])
{!! Form::close() !!}
</div>
</div>
</div>
</div>
@endsection
@section('js')@endsection
```

Hospitals Index View

```
@extends('app')
@section('css')@endsection
@section('content')
<div class="row">
<div class="medium-8 columns">
<h4>Hospitals and Clinics</h4>
</div>
<div class="medium-4 columns clearfix">
<a href="{{ url('hospitals/create') }}" class="button
    float-right show-for-medium">Add a Hospital Record
```

```
    </a>
<a href="{{ url('hospitals/create') }}" class="button
    show-for-small-only">Add a Hospital Record</a>
</div>
</div>
<div class="row">
<div class="medium-12 column margin-bottom-10 end">
<hr>
@if (count($hospitals) > 0)
@foreach ($hospitals as $hospital)
<a href="{{ url('hospitals/'. $hospital->id) }}">
<h5>{{ $hospital->name }}</h5>
</a>
<p>
Address: {{ $hospital->address }}<br>
@if ($hospital->landline_number) Landline Number: {{
    $hospital->landline_number }}<br> @endif
@if ($hospital->mobile_number) Mobile Number: {{
    $hospital->mobile_number }}<br> @endif
@if ($hospital->fax_number) Fax Number: {{ $hospital->
    fax_number }}<br> @endif
@if ($hospital->email_address) E-mail Address: {{
    $hospital->email_address }}<br> @endif
</p><br>
@endforeach
@else
<p>There are currently no hospital records.</p>
@endif
</div>
</div>
@endsection
@section('js')@endsection
```

Hospitals Show View

```
@extends('app')
@section('css'){!! Html::style('css/responsive-tables.
    css') !!}@endsection
@section('content')
<div class="row">
<div class="small-10 small-centered medium-10 medium-
    centered large-10 large-centered column">
<div class="row column center"><h4>{{ $hospital->name
    }}</h4></div>
<hr>
<div class="row column margin-bottom-20 menu-centered">
<ul class="vertical medium-horizontal menu">
<li class="menu-text">View Hospital</li>
<li>
<a href="{{ url('announcements-and-events/'. $hospital
    ->name) }}">
Hospital Announcements and Events
</a>
</li>
</ul>
</div>
<div class="row column margin-bottom-10">
<ul class="tabs" data-tabs id="hospital-tabs">
<li class="tabs-title is-active"><a href="#details"
    aria-selected="true">General Details</a></li>
<li class="tabs-title"><a href="#medical_professionals"
    >Medical Professionals</a></li>
<li class="tabs-title"><a href="#nutritionists">
    Nutritionists</a></li>
<li class="tabs-title"><a href="#physiatrists">
    Physiatrists</a></li>
@if ($is_local_admin or $is_staff_in_hospital)
<li class="tabs-title"><a href="#patients">Patients</a
    ></li>
@endif
</ul>
<div class="tabs-content" data-tabs-content="hospital-
    tabs">
<div class="tabs-panel is-active" id="details">
<h6>General Details:</h6>
<p>
Address: {{ $hospital->address }}<br>
@if ($hospital->landline_number) Landline Number: {{
    $hospital->landline_number }}<br> @endif
@if ($hospital->mobile_number) Mobile Number: {{
    $hospital->mobile_number }}<br> @endif
@if ($hospital->fax_number) Fax Number: {{ $hospital->
    fax_number }}<br> @endif
@if ($hospital->email_address) E-mail Address: {{
    $hospital->email_address }}<br> @endif
</p>
</div>
<div class="tabs-panel" id="medical_professionals">
<h6>Medical Professionals:</h6>
@if (count($professionals['medical_professionals']) >
    0)

<table class="responsive" width="100%">
<tr>
<th></th>
<th class="center">Medical Professional</th>
<th class="center">Specialties</th>
<th class="center">Clinic Schedule</th>
</tr>
@foreach ($professionals['medical_professionals'] as
    $medical_professional)
@if ($medical_professional->user)
<tr>
<td>
<div class="profile-card" id="rem-3">
@if ($medical_professional->user->file)
<img src="{{ url($medical_professional->user->file->
    path. '/'.
$medical_professional->user->file->filename) }}">
@else
<img src="{{ url('files/profile_pictures/anon-profile.
    jpg') }}">
@endif
</div>
</td>
<td>
<div class="profile-card" id="rem-3">
<div class="padding-top-profile-picture">
<a href="{{ url('hospitals/'. $hospital->id. '/mp/'.
$medical_professional->id) }}">
```

```
{{ $medical_professional->user->last_name }},
{{ $medical_professional->user->first_name }}
{{ $medical_professional->user->middle_name }}
</a>
</div>
</div>
</td>
<td>
@if ($medical_professional->specialties) {{
    $medical_professional->specialties }}
@else <em>n/a</em> @endif
</td>
<td>
@if ($medical_professional->clinic_schedule)
{{ $medical_professional->clinic_schedule }}
@else <em>n/a</em> @endif
</td>
</tr>
@endif
@endforeach
</table>

@else
<p>There are currently no medical professionals in this
    hospital or clinic record.</p>
@endif
</div>
<div class="tabs-panel" id="nutritionists">
<h6>Nutritionists:</h6>

@if (count($professionals['nutritionists']) > 0)
<table class="responsive" width="100%">
<tr>
<th></th>
<th class="center">Nutritionist</th>
<th class="center">Clinic Schedule</th>
</tr>
@foreach ($professionals['nutritionists'] as
    $nutritionist)
@if ($nutritionist->user)
<tr>
<td>
<div class="profile-card" id="rem-3">
@if ($nutritionist->user->file)
<img src="{{ url($nutritionist->user->file->path . '/' .
    $nutritionist->user->file->filename) }}">
@else
<img src="{{ url('files/profile_pictures/anon-profile.
    jpg') }}">
@endif
</div>
</td>
<td>
<div class="profile-card" id="rem-3">
<div class="padding-top-profile-picture">
<a href="{{ url('hospitals/' . $hospital->id . '/n/' .
$nutritionist->id) }}">
{{ $nutritionist->user->last_name }},
{{ $nutritionist->user->first_name }}
{{ $nutritionist->user->middle_name }}
</a>
</div>
</div>
</td>
<td>
@if ($nutritionist->clinic_schedule)
{{ $nutritionist->clinic_schedule }}
@else <em>n/a</em> @endif
</td>
</tr>
@endif
@endforeach
</table>

@else
<p>There are currently no nutritionists in this
    hospital or clinic record.</p>
@endif
</div>
<div class="tabs-panel" id="physiatrists">
<h6>Physiatrists:</h6>

@if (count($professionals['physiatrists']) > 0)
<table class="responsive" width="100%">
<tr>
<th></th>
<th class="center">Physiatrist</th>
<th class="center">Clinic Schedule</th>
</tr>
@foreach ($professionals['physiatrists'] as
    $physiatrist)
@if ($physiatrist->user)
<tr>
<td>
<div class="profile-card" id="rem-3">
@if ($physiatrist->user->file)
<img src="{{ url($physiatrist->user->file->path . '/' .
$physiatrist->user->file->filename) }}">
@else
<img src="{{ url('files/profile_pictures/anon-profile.
    jpg') }}">
@endif
</div>
</td>
<td>
<div class="profile-card" id="rem-3">
<div class="padding-top-profile-picture">
<a href="{{ url('hospitals/' . $hospital->id . '/p/' .
$physiatrist->id) }}">
{{ $physiatrist->user->last_name }},
{{ $physiatrist->user->first_name }}
{{ $physiatrist->user->middle_name }}
</a>
</div>
</div>
</td>
<td>
@if ($physiatrist->clinic_schedule)
{{ $physiatrist->clinic_schedule }}
```

```
@else <em>n/a</em> @endif
</td>
</tr>
@endif
@endforeach
</table>

@else
<p>There are currently no physiatrists in this hospital
      or clinic record.</p>
@endif
</div>
@if ($is_local_admin or $is_staff_in_hospital)
<div class="tabs-panel" id="patients">
<h6>Patients:</h6>
@if ($hospital->patients and count($hospital->patients)
      > 0)
@foreach ($hospital->patients as $patient)
@if ($patient->user)
<div class="row column margin-bottom-20">
<div class="profile-card" id="rem-3">
@if ($patient->user->file)
<img src="{{ url($patient->user->file->path . '/' .
$patient->user->file->filename) }}">
@else
<img src="{{ url('files/profile_pictures/anon-profile.
     jpg') }}">
@endif
<div class="padding-top-profile-picture">
<a href="{{ url('hospitals/' . $hospital->id . '/
     patients/' .
$patient->id) }}">
{{ $patient->user->last_name }}, {{ $patient->user->
     first_name }}
{{ $patient->user->middle_name }}
</a>
</div>
</div>
</div>
@endif
@endforeach
@else
<p>There are currently no patients in this hospital or
     clinic record.</p>
@endif
</div>
@endif
</div>
</div>
<div class="row column margin-bottom-20 menu-centered">
<ul class="vertical medium-horizontal menu ul-margin">
@if ($is_staff)
<li>
@if ($is_staff_in_hospital)
{!! Form::open(['method' => 'POST', 'url' => 'hospitals
     /' . $hospital->id . '/leave']) !!}
{!! Form::submit('Leave Hospital', ['class' => 'button'
     ]) !!}
{!! Form::close() !!}
@else
{!! Form::open(['method' => 'POST', 'url' => 'hospitals
     /' . $hospital->id . '/register']) !!}
{!! Form::submit('Register in Hospital', ['class' => '
     button']) !!}
{!! Form::close() !!}
@endif
</li>
@endif
@if ($is_local_admin and $hospital->is_unlocked)
<li>
<a href="{{ url('hospitals/' . $hospital->id . '/edit')
     }}" class="button">
Edit Hospital Record
</a>
</li>
@endif
@if ($is_system_admin)
<li>
@if ($hospital->is_unlocked)
{!! Form::open(['method' => 'POST', 'url' => 'hospitals
     /' . $hospital->id . '/lock']) !!}
{!! Form::submit('Lock Hospital', ['class' => 'button'
     ]) !!}
{!! Form::close() !!}
@else
{!! Form::open(['method' => 'POST', 'url' => 'hospitals
     /' . $hospital->id . '/unlock']) !!}
{!! Form::submit('Unlock Hospital', ['class' => 'button
     ']) !!}
{!! Form::close() !!}
@endif
</li>
@endif
<li><a href="{{ url('hospitals') }}" class="button
     hollow">Cancel</a></li>
</ul>
</div>
</div>
</div>
@endsection
@section('js')
{!! Html::script('js/responsive-tables.js') !!}
<script type="text/javascript">
$('.tabs').foundation();
</script>
@endsection
```

Hospitals Show Patient View

```
@extends('app')
@section('css')@endsection
@section('content')
<div class="row margin-bottom-10">
<div class="small-10 small-centered medium-10 medium-
     centered large-10 large-centered column">
<h4 class="text-center">Patient Account</h4>
<hr>
<div class="row">
<div class="medium-6 columns center margin-bottom-10">
@if ($patient->user->file)
```

```
<img src="{{ url($patient->user->file->path . '/' .
    $patient->user->file->filename) }}"
width="250rem">
@else
<img src="{{ url('files/profile_pictures/anon-profile.
    jpg') }}" width="250rem">
@endif
</div>
<div class="medium-6 columns margin-bottom-10">
<div class="row margin-bottom-10">
<div class="medium-5 columns">Hospital:</div>
<div class="medium-7 columns end">
@if ($hospital->name) {{ $hospital->name }} @else <em>n
    /a</em> @endif
</div>
</div>
<div class="row margin-bottom-10">
<div class="medium-5 columns">Hospital ID Number:</div>
<div class="medium-7 columns end">
@if ($patient->hospital_id_number) {{ $patient->
    hospital_id_number }}
@else <em>n/a</em> @endif
</div>
</div>
<div class="row margin-bottom-10">
<div class="medium-5 columns">Username:</div>
<div class="medium-7 columns end">
@if ($patient->user->username) {{ $patient->user->
    username }} @else <em>n/a</em> @endif
</div>
</div>
<div class="row margin-bottom-10">
<div class="medium-5 columns">E-mail Address:</div>
<div class="medium-7 columns end">
@if ($patient->user->email) {{ $patient->user->email }}
     @else <em>n/a</em> @endif
</div>
</div>
<div class="row margin-bottom-10">
<div class="medium-5 columns">First Name:</div>
<div class="medium-7 columns end">
@if ($patient->user->first_name) {{ $patient->user->
    first_name }} @else <em>n/a</em> @endif
</div>
</div>
<div class="row margin-bottom-10">
<div class="medium-5 columns">Middle Name:</div>
<div class="medium-7 columns end">
@if ($patient->user->middle_name) {{ $patient->user->
    middle_name }}
@else <em>n/a</em> @endif
</div>
</div>
<div class="row margin-bottom-10">
<div class="medium-5 columns">Last Name:</div>
<div class="medium-7 columns end">
@if ($patient->user->last_name) {{ $patient->user->
    last_name }} @else <em>n/a</em> @endif
</div>
</div>
<div class="row margin-bottom-10">
<div class="medium-5 columns">Sex:</div>
<div class="medium-7 columns end">
@if ($patient->user->sex) {{ ucfirst($patient->user->
    sex) }} @else <em>n/a</em> @endif
</div>
</div>
<div class="row margin-bottom-10">
<div class="medium-5 columns">Birthday:</div>
<div class="medium-7 columns end">
@if ($patient->user->birthday[1]) {{ $patient->user->
    birthday[1] }}
@else <em>n/a</em> @endif
</div>
</div>
<div class="row margin-bottom-10">
<div class="medium-5 columns">Caregiver First Name:</
    div>
<div class="medium-7 columns end">
@if ($patient->caregiver_first_name) {{ $patient->
    caregiver_first_name }}
@else <em>n/a</em> @endif
</div>
</div>
<div class="row margin-bottom-10">
<div class="medium-5 columns">Caregiver Middle Name:</
    div>
<div class="medium-7 columns end">
@if ($patient->caregiver_middle_name) {{ $patient->
    caregiver_middle_name }}
@else <em>n/a</em> @endif
</div>
</div>
<div class="row margin-bottom-10">
<div class="medium-5 columns">Caregiver Last Name:</div
    >
<div class="medium-7 columns end">
@if ($patient->caregiver_last_name) {{ $patient->
    caregiver_last_name }}
@else <em>n/a</em> @endif
</div>
</div>
<div class="row margin-bottom-10">
<div class="medium-5 columns">Medical Professional:</
    div>
<div class="medium-7 columns end">
@if ($patient->medicalProfessional)
{{ $patient->medicalProfessional->user->last_name }},
{{ $patient->medicalProfessional->user->first_name }}
{{ $patient->medicalProfessional->user->middle_name }}
@else <em>n/a</em> @endif
</div>
</div>
<div class="row margin-bottom-10">
<div class="medium-5 columns">Nutritionist:</div>
<div class="medium-7 columns end">
@if ($patient->nutritionist)
{{ $patient->nutritionist->user->last_name }},
{{ $patient->nutritionist->user->first_name }}
{{ $patient->nutritionist->user->middle_name }}
```

```
@else <em>n/a</em> @endif
</div>
</div>
<div class="row margin-bottom-10">
<div class="medium-5 columns">Physiatrist:</div>
<div class="medium-7 columns end">
@if ($patient->physiatrist)
{{ $patient->physiatrist->user->last_name }},
{{ $patient->physiatrist->user->first_name }}
{{ $patient->physiatrist->user->middle_name }}
@else <em>n/a</em> @endif
</div>
</div>
</div>
</div>
<div class="row column margin-bottom-20 menu-centered">
<ul class="vertical medium-horizontal menu ul-margin">
@if ($is_local_admin and $hospital->is_unlocked)
<li>
@if ($patient->user->is_unlocked)
{!! Form::open(['method' => 'POST',
'url' => 'hospitals/' . $hospital->id . '/patients/' .
    $patient->id . '/lock']) !!}
{!! Form::submit('Deactivate', ['class' => 'button'])
    !!}
{!! Form::close() !!}
@else
{!! Form::open(['method' => 'POST',
'url' => 'hospitals/' . $hospital->id . '/patients/' .
    $patient->id . '/unlock']) !!}
{!! Form::submit('Activate', ['class' => 'button']) !!}
{!! Form::close() !!}
@endif
</li>
@if (! $patient->user->is_email_confirmed)
<li>
{!! Form::open(['method' => 'POST', 'url' => 'hospitals
    /' . $hospital->id . '/patients/' .
$patient->id . '/email']) !!}
{!! Form::submit('Send E-mail Confirmation', ['class'
    => 'button']) !!}
{!! Form::close() !!}
</li>
@endif
@endif
@if ($is_patient_in_hospital and $is_user_in_hospital
    and $can_maintain)
<li>
{!! Form::open(['method' => 'POST', 'url' => 'hospitals
    /' . $hospital->id . '/patients/' .
$patient->id . '/maintain-patient']) !!}
{!! Form::submit('Maintain Patient', ['class' => '
    button']) !!}
{!! Form::close() !!}
</li>
@endif
<li><a href="{{ url('hospitals/' . $hospital->id) }}"
    class="button hollow">Cancel</a></li>
</ul>
</div>
</div>
</div>
@endsection
@section('js')@endsection
```

Hospitals Show Staff Member View

```
@extends('app')
@section('css')@endsection
@section('content')
<div class="row margin-bottom-10">
<div class="small-10 small-centered medium-10 medium-
    centered large-10 large-centered columns">
<h4 class="text-center">{{ $profession_name }} Staff
    Account</h4>
<hr>
<div class="row">
<div class="medium-6 columns center margin-bottom-10">
@if ($staff->user->file)
<img src="{{ url($staff->user->file->path . '/' .
    $staff->user->file->filename) }}" width="250rem">
@else
<img src="{{ url('files/profile_pictures/anon-profile.
    jpg') }}" width="250rem">
@endif
</div>
<div class="medium-6 columns margin-bottom-10">
<div class="row margin-bottom-10">
<div class="medium-5 columns">Hospital:</div>
<div class="medium-7 columns end">
@if ($hospital->name) {{ $hospital->name }} @else <em>n
    /a</em> @endif
</div>
</div>
@if ($profession == 'mp')
<div class="row margin-bottom-10">
<div class="medium-5 columns">Specialties:</div>
<div class="medium-7 columns end">
@if ($staff->specialties) {{ $staff->specialties }}
    @else <em>n/a</em> @endif
</div>
</div>
@endif
<div class="row margin-bottom-10">
<div class="medium-5 columns">Clinic Schedule:</div>
<div class="medium-7 columns end">
@if ($staff->clinic_schedule) {{ $staff->
    clinic_schedule }} @else <em>n/a</em> @endif
</div>
</div>
<div class="row margin-bottom-10">
<div class="medium-5 columns">Username:</div>
<div class="medium-7 columns end">
@if ($staff->user->username) {{ $staff->user->username
    }} @else <em>n/a</em> @endif
</div>
</div>
<div class="row margin-bottom-10">
<div class="medium-5 columns">E-mail Address:</div>
<div class="medium-7 columns end">
```

```
@if ($staff->user->email) {{ $staff->user->email }}
    @else <em>n/a</em> @endif
</div>
</div>
<div class="row margin-bottom-10">
<div class="medium-5 columns">First Name:</div>
<div class="medium-7 columns end">
@if ($staff->user->first_name) {{ $staff->user->
    first_name }} @else <em>n/a</em> @endif
</div>
</div>
<div class="row margin-bottom-10">
<div class="medium-5 columns">Middle Name:</div>
<div class="medium-7 columns end">
@if ($staff->user->middle_name) {{ $staff->user->
    middle_name }} @else <em>n/a</em> @endif
</div>
</div>
<div class="row margin-bottom-10">
<div class="medium-5 columns">Last Name:</div>
<div class="medium-7 columns end">
@if ($staff->user->last_name) {{ $staff->user->
    last_name }} @else <em>n/a</em> @endif
</div>
</div>
<div class="row margin-bottom-10">
<div class="medium-5 columns">Sex:</div>
<div class="medium-7 columns end">
@if ($staff->user->sex) {{ ucfirst($staff->user->sex)
    }} @else <em>n/a</em> @endif
</div>
</div>
<div class="row margin-bottom-10">
<div class="medium-5 columns">Birthday:</div>
<div class="medium-7 columns end">
@if ($staff->user->birthday[1]) {{ $staff->user->
    birthday[1] }} @else <em>n/a</em> @endif
</div>
</div>
</div>
</div>
<div class="row column margin-bottom-20 menu-centered">
<ul class="vertical medium-horizontal menu ul-margin">
@if ($is_local_admin and $hospital->is_unlocked)
<li>
@if ($is_staff_approved)
{!! Form::open(['method' => 'POST',
'url' => 'hospitals/' . $hospital->id . '/' .
    $profession . '/' . $staff->id . '/lock']) !!}
{!! Form::submit('Deactivate', ['class' => 'button'])
    !!}
{!! Form::close() !!}
@else
{!! Form::open(['method' => 'POST',
'url' => 'hospitals/' . $hospital->id . '/' .
    $profession . '/' . $staff->id . '/unlock']) !!}
{!! Form::submit('Activate', ['class' => 'button']) !!}
{!! Form::close() !!}
@endif
</li>
@if (! $staff->user->is_email_confirmed)
<li>
{!! Form::open(['method' => 'POST', 'url' => 'hospitals
    /' . $hospital->id . '/' . $profession .
'/' . $staff->id . '/email']) !!}
{!! Form::submit('Send E-mail Confirmation', ['class'
    => 'button']) !!}
{!! Form::close() !!}
</li>
@endif
@endif
<li><a href="{{ url('hospitals/' . $hospital->id) }}"
    class="button hollow">Cancel</a></li>
</ul>
</div>
</div>
</div>
@endsection
@section('js')@endsection
```

Laboratory Results Form View

```
<div class="callout">
<div class="row column">
<h5>Laboratory Result Details</h5>
<p class="help-text">
Answer the following fields about the laboratory test
    conducted to the patient.
</p>
<hr>
</div>
<div class="row margin-bottom-10">
<!-- Type of Laboratory Exam Form Input -->
<div class="medium-6 columns">
<label for="type_of_laboratory_exam">Type of Laboratory
     Exam*
{!! Form::text('type_of_laboratory_exam',
    $laboratory_result->type_of_laboratory_exam,
['class' => 'default', 'id' => 'type_of_laboratory_exam
    ']) !!}
@foreach ($errors->get('type_of_laboratory_exam') as
    $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('
    type_of_laboratory_exam')) - 1)<br>@endif
@endforeach
</label>
</div>
</div>
<div class="row margin-bottom-10">
<!-- Unit of Measurement Form Input -->
<div class="medium-4 columns">
<label for="unit_of_measurement">Unit of Measurement
{!! Form::text('unit_of_measurement',
    $laboratory_result->unit_of_measurement,
['class' => 'default', 'id' => 'unit_of_measurement'])
    !!}
@foreach ($errors->get('unit_of_measurement') as $key
    => $error)
<small class="custom-form-error">{{ $error }}</small>
```

```
@if ($key == count($errors->get('unit_of_measurement'))
    - 1)<br>@endif
@endforeach
</label>
</div>
<!-- Reference Values Form Input -->
<div class="medium-4 columns">
<label for="reference_values">Reference Values
{!! Form::text('reference_values', $laboratory_result->
    reference_values,
['class' => 'default', 'id' => 'reference_values']) !!}
@foreach ($errors->get('reference_values') as $key =>
    $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('reference_values')) -
    1)<br>@endif
@endforeach
</label>
</div>
<!-- Date of Laboratory Test Form Input -->
<div class="medium-4 columns">
<label for="date_of_test">Date of Laboratory Test*
{!! Form::date('date_of_test',
$laboratory_result->date_of_test[0]->format('Y-m-d'),
['class' => 'default', 'id' => 'date_of_test', 'max' =>
    $current_date->format('Y-m-d')]) !!}
@foreach ($errors->get('date_of_test') as $key =>
    $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('date_of_test')) - 1)<
    br>@endif
@endforeach
</label>
</div>
</div>
<div class="row margin-bottom-10">
<!-- Result Form Input -->
<div class="medium-12 columns">
<label for="result">Result*
{!! Form::textarea('result', $laboratory_result->result
    ,
['class' => 'default', 'id' => 'result', 'rows' => '3',
'placeholder' => 'Input the result of the laboratory
    test.']) !!}
@foreach ($errors->get('result') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('result')) - 1)<br>
    @endif
@endforeach
</label>
</div>
</div>
</div>
<div class="callout">
<div class="row column">
<h5>Laboratory Result File</h5>
<p class="help-text">
Upload the scanned laboratory result files here. These
    include images, videos, PDF
files, etc. In case of multiple files, use data
    compression tools to compress them in
an archive file (e.g., ZIP, RAR). The maximum size for
    a file is 10MB.
</p>
</div>
<hr>
@if ($laboratory_result->file)
<div class="row column margin-bottom-10">
<p class="font-0875rem">
Currently Stored File:
@if ($laboratory_result->file != null)
@if (in_array($laboratory_result->file->extension,
    $image_extensions)
or in_array($laboratory_result->file->extension,
    $video_extensions)
or $laboratory_result->file->extension == 'pdf')
<a data-open="file_0">
{{ $laboratory_result->file->filename }}
</a>
@else
<a href="{{ url('laboratory-results/download/' .
    $laboratory_result->id) }}">
{{ $laboratory_result->file->filename }}
</a>
@endif
@else <em>n/a</em>
@endif
</p>
<p class="help-text">
Note: The stored file will be deleted if a new file is
    uploaded below then submitted.
</p>
@if ($laboratory_result->file and
(in_array($laboratory_result->file->extension,
    $image_extensions))
or (in_array($laboratory_result->file->extension,
    $video_extensions))
or ($laboratory_result->file->extension == 'pdf'))
<div class="reveal large" id="file_0" data-reveal
data-animation-in="fade-in" data-animation-out="fade-
    out">
<h4>Laboratory Result File</h4>
<div class="row column center margin-top-20 margin-
    bottom-20">
@if (in_array($laboratory_result->file->extension,
    $image_extensions))
<img src="{{ url($laboratory_result->file->path . '/' .
$laboratory_result->file->filename) }}">
@elseif (in_array($laboratory_result->file->extension,
    $video_extensions))
<video height="500rem" controls>
@if ($laboratory_result->file->extension == 'mp4')
<source src="{{ url($laboratory_result->file->path .
    '/' .
$laboratory_result->file->filename) }}" type="video/mp4
    ">
@elseif ($laboratory_result->file->extension == 'webm')
<source src="{{ url($laboratory_result->file->path .
    '/' .
$laboratory_result->file->filename) }}" type="video/
```

```
    webm">
@elseif ($laboratory_result->file->extension == 'ogg')
<source src="{{ url($laboratory_result->file->path . '/' .
$laboratory_result->file->filename) }}" type="video/ogg
    ">
@endif
Your browser does not support the video tag.
</video>
@else
<object data="{{ url($laboratory_result->file->path . '/' .
$laboratory_result->file->filename) }}" type="
    application/pdf" width="100%"
height="500rem">
</object>
@endif
</div>
<div class="row column center">
<a href="{{ url('laboratory-results/download/' .
    $laboratory_result->id)}}"
class="button">
Download
</a>
<a class="button hollow" data-close>Close</a>
</div>
<button class="close-button" data-close aria-label="
    Close modal" type="button">
<span aria-hidden="true">&times;</span>
</button>
</div>
@endif
</div>
@endif
<div class="row margin-bottom-10">
<!-- File Upload Form Input -->
<div class="large-2 columns">
<label for="file" class="button">Upload File</label>
{!! Form::file('file', ['id' => 'file', 'class' => '
    show-for-sr',]) !!}
</div>
<div class="large-6 medium-6 columns end">
{!! Form::text('filename', null, ['id' => 'filename', '
    class' => 'default',
'disabled' => 'disabled']) !!}
@foreach ($errors->get('file') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('file')) - 1)<br>@endif
@endforeach
</div>
</div>
</div>
<div class="row column">
<p class="help-text">Fields marked with * are mandatory
    .</p>
</div>
<!-- Submit Button -->
<div class="row column center">
{!! Form::submit($submit_button_text, ['class' => '
    button',]) !!}
<a href="{{ url('laboratory-results/' . $patient->id)
    }}" class="button hollow">Cancel</a>
</div>
```

Laboratory Results Script View

```
<script type="application/javascript">
$(document).ready(function() {
$('input[type=file]').change(function(){
$(this).parent().next().find('input[type="text"]').val(
    $(this).val().split('\\').pop());
});
});
</script>
```

Laboratory Results Create View

```
@extends('app')
@section('css')@endsection
@section('content')
<div class="row">
<div class="small-10 small-centered medium-10 medium-
    centered large-10 large-centered column">
<div class="center"><h4>Add Laboratory Result</h4></div
    >
<hr>
<div class="row column margin-bottom-10">
{!! Form::model($laboratory_result = new \App\Models\
    LaboratoryResult, ['method' => 'POST',
'url' => 'laboratory-results/' . $patient->id, 'files'
    => true]) !!}
@include('laboratory_results._form',
['submit_button_text' => 'Add Laboratory Result'])
{!! Form::close() !!}
</div>
</div>
</div>
@endsection
@section('js')
@include('laboratory_results._script')
@endsection
```

Laboratory Results Edit View

```
@extends('app')
@section('css')@endsection
@section('content')
<div class="row">
<div class="small-10 small-centered medium-10 medium-
    centered large-10 large-centered column">
<div class="center"><h4>Edit Laboratory Result</h4></
    div>
<hr>
<div class="row column margin-bottom-10">
{!! Form::model($laboratory_result, ['method' => 'PATCH
    ',
'url' => 'laboratory-results/edit/' .
    $laboratory_result->id, 'files' => true]) !!}
@include('laboratory_results._form',
['submit_button_text' => 'Edit Laboratory Result'])
```

```
{!! Form::close() !!}
</div>
</div>
</div>
@endsection
@section('js')
<script type="text/javascript">
$('.reveal').foundation();
</script>
@include('laboratory_results._script')
@endsection
```

Laboratory Results Index View

```
@extends('app')
@section('css')@endsection
@section('content')
<div class="row">
<div class="small-10 small-centered medium-10 medium-
    centered large-10 large-centered column">
<div class="center"><h4>Laboratory Results: {{
    $patient_full_name }}</h4></div>
<hr>
@if ($is_staff)
<div class="row column margin-bottom-20 menu-centered">
<ul class="vertical medium-horizontal menu">
<li><a href="{{ url('patients/' . $patient->id) }}">
    Account Profile</a></li>
@if ($user->medicalProfessional)
<li><a href="{{ url('patient-attributes/' . $patient->
    id) }}">General Details</a></li>
<li><a href="{{ url('medical-conditions/' . $patient->
    id) }}">Daily Medical Condition</a></li>
<li class="menu-text">Laboratory Results</li>
<li><a href="{{ url('medical-records/' . $patient->id) }
    }">Medical Records</a></li>
<li><a href="{{ url('referral/' . $patient->id) }}">
    Referral</a></li>
<li><a href="{{ url('medication-regimens/' . $patient->
    id) }}">Medication Regimens</a></li>
@endif
</ul>
</div>
@endif
<div class="row column margin-bottom-10">
<div class="callout">
@if (isset($laboratory_results) and count(
    $laboratory_results) > 0)
@foreach ($laboratory_results as $index =>
    $laboratory_result)
<div class="callout margin-bottom-20">
<div class="row">
<div class="medium-6 columns">
{{ $index + 1 }}. Submitted on: {{ $laboratory_result->
    created_at->format('F, d, Y') }}
</div>
@if ($user->medicalProfessional)
<div class="medium-6 columns">
<p class="float-right">
<a href="{{ url('laboratory-results/edit/' .
    $laboratory_result->id) }}"
class="button">
Edit
</a>
</p>
</div>
@endif
</div>
<div class="row">
<div class="medium-12 columns">
<p>
Type of Laboratory Test:<br>
@if ($laboratory_result->type_of_laboratory_exam)
{{ $laboratory_result->type_of_laboratory_exam }}
@else <em>n/a</em>
@endif
</p>
</div>
</div>
<div class="row">
<div class="medium-4 columns">
<p class="font-0875rem">
Unit of Measurement:<br>
@if ($laboratory_result->unit_of_measurement)
{{ $laboratory_result->unit_of_measurement }}
@else <em>n/a</em>
@endif
</p>
</div>
<div class="medium-4 columns">
<p class="font-0875rem">
Reference Values:<br>
@if ($laboratory_result->reference_values)
{{ $laboratory_result->reference_values }}
@else <em>n/a</em>
@endif
</p>
</div>
<div class="medium-4 columns">
<p class="font-0875rem">
Date of Laboratory Test:<br>
@if ($laboratory_result->date_of_test[1])
{{ $laboratory_result->date_of_test[1] }}
@else <em>n/a</em>
@endif
</p>
</div>
</div>
<div class="row">
<div class="medium-12 columns">
<p class="font-0875rem">
Result:<br>
@if ($laboratory_result->result)
{{ $laboratory_result->result }}
@else <em>n/a</em>
@endif
</p>
</div>
</div>
</div>
```

```
<div class="callout margin-bottom-20">
<div class="row medium-6 column">
<p>
Laboratory Result File:<br>
@if ($laboratory_result->file != null)
@if (in_array($laboratory_result->file->extension,
    $image_extensions)
or in_array($laboratory_result->file->extension,
    $video_extensions)
or $laboratory_result->file->extension == 'pdf')
<a data-open="file_{{ $laboratory_result->id }}">
{{ $laboratory_result->file->filename }}
</a>
@else
<a href="{{ url('laboratory-results/download/' .
    $laboratory_result->id) }}">
{{ $laboratory_result->file->filename }}
</a>
@endif
@else <em>n/a</em>
@endif
</p>
@if ($laboratory_result->file != null)
@if (in_array($laboratory_result->file->extension,
    $image_extensions)
or in_array($laboratory_result->file->extension,
    $video_extensions)
or ($laboratory_result->file->extension == 'pdf'))
<div class="reveal large" id="file_{{
    $laboratory_result->id }}" data-reveal
data-animation-in="fade-in" data-animation-out="fade-
    out">
<h4>Laboratory Result File</h4>
<div class="row column center margin-top-20 margin-
    bottom-20">
@if (in_array($laboratory_result->file->extension,
    $image_extensions))
<img src="{{ url($laboratory_result->file->path . '/' .
$laboratory_result->file->filename) }}">
@elseif (in_array($laboratory_result->file->extension,
    $video_extensions))
<video height="500rem" controls>
@if ($laboratory_result->file->extension == 'mp4')
<source src="{{ url($laboratory_result->file->path .
    '/' .
$laboratory_result->file->filename) }}" type="video/mp4
    ">
@elseif ($laboratory_result->file->extension == 'webm')
<source src="{{ url($laboratory_result->file->path .
    '/' .
$laboratory_result->file->filename) }}" type="video/
    webm">
@elseif ($laboratory_result->file->extension == 'ogg')
<source src="{{ url($laboratory_result->file->path .
    '/' .
$laboratory_result->file->filename) }}" type="video/ogg
    ">
@endif
Your browser does not support the video tag.
</video>
@else
<object data="{{ url($laboratory_result->file->path . .
    '/' .
$laboratory_result->file->filename) }}" type="
    application/pdf" width="100%"
height="500rem">
</object>
@endif
</div>
<div class="row column center">
<a href="{{ url('laboratory-results/download/' .
    $laboratory_result->id) }}"
class="button">
Download
</a>
<a class="button hollow" data-close>Close</a>
</div>
<button class="close-button" data-close aria-label="
    Close modal" type="button">
<span aria-hidden="true">&times;</span>
</button>
</div>
@endif
@endif
</div>
</div>
@endforeach
{!! $page_links !!}
@else
There are currently no laboratory results.
@endif
</div>
</div>
@if ($user->patient)
<div class="row column center">
<a href="{{ url('laboratory-results/' . $user->patient
    ->id . '/input') }}" class="button">
Input Laboratory Result
</a>
</div>
@endif
</div>
</div>
@endsection
@section('js')
<script type="text/javascript">
$('.reveal').foundation();
</script>
@endsection
```

Medical Conditions Edit View

```
@extends('app')
@section('css')@endsection
@section('content')
<div class="row">
<div class="small-10 small-centered medium-10 medium-
    centered large-10 large-centered column">
<div class="center"><h4>Daily Medical Condition</h4></
    div>
```

```
<hr>
@if ($medical_condition->followUpVisit)
<div class="row column">
<p>Set Schedule for Visit: {{ $medical_condition->
    followUpVisit->date_of_visit[1] }}</p>
</div>
@endif
@if (! $medical_condition->decisionOnCondition)
<div class="row column margin-bottom-10">
<p>Make Decision in this Medical Condition</p>
{!! Form::open(['method' => 'POST', 'url' => 'medical-
    conditions/' . $patient->id . '/' .
    $medical_condition->id . '/decision']) !!}
<!-- Decision Form Input -->
<div class="medium-6 columns">
<label for="decision">Decision
{!! Form::select('decision', $decisions, null, ['class'
     => 'default', 'id' => 'decision']) !!}
</label>
</div>
<!-- Additional Notes Form Input -->
<div class="medium-6 columns">
<label for="additional_notes">Additional Notes
{!! Form::textarea('additional_notes', null,
['class' => 'default', 'id' => 'additional_notes', '
    rows' => '2', 'placeholder' => '']) !!}
</label>
</div>
<!-- Submit Button -->
<div class="row column">
<div class="center">
{!! Form::submit('Submit Decision', ['class' => 'button
    ']) !!}
</div>
</div>
{!! Form::close() !!}
</div>
@endif
<div class="row column margin-bottom-10">
{!! Form::open(['method' => 'PATCH', 'url' => 'medical-
    conditions/' . $patient->id . '/' .
    $medical_condition->id]) !!}
<div class="row margin-bottom-10">
<!-- What is the daily medical problem the patient is
    experiencing? Form Input -->
<div class="medium-12 columns">
<label for="daily_medical_problem">What is the daily
    medical problem the patient is experiencing?
{!! Form::text('daily_medical_problem',
    $medical_condition->daily_medical_problem, ['class
    ' => 'default',
'id' => 'daily_medical_problem', 'aria-describedby' =>
    'daily_medical_problem_helper']) !!}
</label>
<p class="help-text" id="daily_medical_problem_helper">
Daily medical problems include headache, weakness, not
    being able to eat, slurred speech,
and other problems that the patient might be
    experiencing.
</p>
</div>
</div>
<div class="row margin-bottom-10">
<!-- Describe the symptom Form Input -->
<div class="medium- columns">
<label for="describe_the_symptom">Describe this symptom
     the patient is experiencing
{!! Form::textarea('describe_the_symptom',
    $medical_condition->describe_the_symptom,
['class' => 'default', 'id' => 'describe_the_symptom',
    'rows' => '3',
'placeholder' => '', 'aria-describedby' => '
    describe_helper']) !!}
</label>
<p class="help-text" id="describe_helper">
Describe the symptom that the patient is experiencing.
    Specify if how often does the symptom
occur or is it a recurring problem. You can also
    specify which body parts of the patient is
affected by this symptom. In addition, you can input
    the activities of the patient that causes
the symptom.
</p>
</div>
</div>
<div class="row margin-bottom-10">
<!-- What are the medicines taken by the patient? Form
    Input -->
<div class="medium-12 columns">
<label for="medicine_taken">What are the medicines
    taken by the patient?
{!! Form::textarea('medicine_taken', $medical_condition
    ->medicine_taken,
['class' => 'default', 'id' => 'medicine_taken', 'rows'
     => '3', 'placeholder' => '',
'aria-describedby' => 'medicine_taken_helper']) !!}
</label>
<p class="help-text" id="medicine_taken_helper">
Enumerate the over-the-counter medicines that the
    patient is taking as a way of alleviating
the symptom. Further, include medicines that are not
    specified in the daily medication
regimen for the patient.
</p>
</div>
</div>
<div class="row margin-bottom-10">
<!-- What are the effects noted from the patient after
    taking these medicines? Form Input -->
<div class="medium-12 columns">
<label for="effects_noted">What are the effects noted
    from the patient after taking these medicines?
{!! Form::textarea('effects_noted', $medical_condition
    ->effects_noted,
['class' => 'default', 'id' => 'effects_noted', 'rows'
    => '3', 'placeholder' => '',
'aria-describedby' => 'effects_noted_helper']) !!}
</label>
<p class="help-text" id="effects_noted_helper">
Describe the effects noted from the patient after
```

```
    taking these medicines. Was there any
improvement? Or did the symptom experienced by the
    patient become worse?
</p>
</div>
</div>
<div class="row margin-bottom-10">
<!-- Are there any maneuvers taken by the patient in
    order to alleviate the problem? Form Input -->
<div class="medium-12 columns">
<label for="any_maneuver_taken">Are there any maneuvers
     taken by the patient in order to alleviate the
    problem?
{!! Form::textarea('any_maneuver_taken',
    $medical_condition->any_maneuver_taken,
['class' => 'default', 'id' => '"any_maneuver_taken"',
    'rows' => '3', 'placeholder' => '',
'aria-describedby' => 'any_maneuver_taken_helper']) !!}
</label>
<p class="help-text" id="any_maneuver_taken_helper">
Describe the maneuvers taken by the patient in order to
     alleviate the symptom. For example,
if the patient experiences a back pain, does sitting on
     a chair supported by a pillow at the
back reduces the pain?
</p>
</div>
</div>
<!-- Submit Button -->
<div class="row column center margin-top-20">
{!! Form::submit('Update Daily Medical Condition', ['
    class' => 'button', 'id' => 'submit_button']) !!}
<a href="{{ url('medical-conditions/' . $patient->id)
    }}" class="button hollow">Cancel</a>
</div>
{!! Form::close() !!}
</div>
</div>
</div>
@endsection

@section('js')@endsection
```

Medical Conditions Index View

```
@extends('app')
@section('css'){!! Html::style('css/responsive-tables.
    css') !!}@endsection
@section('content')
<div class="row">
<div class="small-10 small-centered medium-10 medium-
    centered large-10 large-centered column">
<div class="center"><h4>Daily Medical Condition: {{
    $patient->user->last_name }},
{{ $patient->user->first_name }} {{ $patient->user->
    middle_name }}</h4></div>
<hr>
@if ($is_staff)
<div class="row column margin-bottom-20 menu-centered">
<ul class="vertical medium-horizontal menu">
<li><a href="{{ url('patients/' . $patient->id) }}">
    Account Profile</a></li>
@if ($user->medicalProfessional)
<li><a href="{{ url('patient-attributes/' . $patient->
    id) }}">General Details</a></li>
<li class="menu-text">Daily Medical Condition</li>
<li><a href="{{ url('laboratory-results/' . $patient->
    id) }}">Laboratory Results</a></li>
<li><a href="{{ url('medical-records/' . $patient->id)
    }}">Medical Records</a></li>
<li><a href="{{ url('referral/' . $patient->id) }}">
    Referral</a></li>
<li><a href="{{ url('medication-regimens/' . $patient->
    id) }}">Medication Regimens</a></li>
@endif
</ul>
</div>
@endif
<div class="row column margin-bottom-10">
<div class="callout">
@if (count($medical_conditions) > 0)
@foreach ($medical_conditions as $index =>
    $medical_condition)
<div class="callout margin-bottom-20 @if (!
    $medical_condition->decisionOnCondition) secondary
@elseif (strcmp($medical_condition->decisionOnCondition
    ->decision, 'continue_regimen') == 0) primary
@elseif (strcmp($medical_condition->decisionOnCondition
    ->decision, 'follow_up_visit') == 0) primary
@elseif (strcmp($medical_condition->decisionOnCondition
    ->decision, 'immediate_hospitalization') == 0)
    alert
@endif ">
<div class="row">
<div class="medium-6 columns">
{{ $index + 1 }}. Submitted on: {{ $medical_condition->
    created_at->format('F, d, Y') }}
</div>
@if ($user->medicalProfessional)
<div class="medium-6 columns">
<p class="float-right">
<a href="{{ url('medical-conditions/' . $patient->id .
    '/' .
$medical_condition->id . '/edit') }}" class="button">
Edit
</a>
</p>
</div>
@endif
</div>
<div class="row column">
<p>
Daily Medical Problem:<br>
{{ $medical_condition->daily_medical_problem }}
</p>
<p>
Description of the Symptom:<br>
{{ $medical_condition->describe_the_symptom }}
</p>
<p>
```

```
Medicines Taken:<br>
{{ $medical_condition->medicine_taken }}
</p>
<p>
Effects Noted:<br>
{{ $medical_condition->effects_noted }}
</p>
<p>
Maneuvers Taken:<br>
{{ $medical_condition->any_maneuver_taken }}
</p>
@if (! $medical_condition->decisionOnCondition)
<p>No decision has been made on this medical condition
    .</p>
@else
<p>
Decision: {{ $decisions[$medical_condition->
    decisionOnCondition->decision] }}<br>
Additional Notes:
@if ($medical_condition->decisionOnCondition->
    additional_notes)
{{ $medical_condition->decisionOnCondition->
    additional_notes }}
@else <em>n/a</em>
@endif
</p>
@endif
</div>
</div>
@endforeach
{!! $page_links !!}
@else
<p>The patient currently has no daily medical
    conditions.</p>
@endif
</div>
</div>
@if ($user->patient)
<div class="row column center">
<a href="{{ url('medical-conditions/' . $user->patient
    ->id . '/input') }}" class="button">
Input Daily Medication Regimen
</a>
</div>
@endif
</div>
</div>
@endsection
@section('js'){!! Html::script('js/responsive-tables.js
    ') !!}@endsection
```

Medical Conditions Input View

```
@extends('app')
@section('css')@endsection
@section('content')
<div class="row">
<div class="small-10 small-centered medium-10 medium-
    centered large-10 large-centered column">
<div class="center"><h4>Input Daily Medical Condition</
    h4></div>
<hr>
<div class="row column margin-bottom-10">
{!! Form::open(['method' => 'POST', 'url' => 'medical-
    conditions/' . $patient->id]) !!}
<div class="row margin-bottom-10">
<!-- What is the daily medical problem the patient is
    experiencing? Form Input -->
<div class="medium-12 columns">
<label for="daily_medical_problem">What is the daily
    medical problem the patient is experiencing?
{!! Form::text('daily_medical_problem', null, ['class'
    => 'default',
'id' => 'daily_medical_problem', 'aria-describedby' =>
    'daily_medical_problem_helper']) !!}
</label>
<p class="help-text" id="daily_medical_problem_helper">
Daily medical problems include headache, weakness, not
    being able to eat, slurred speech,
and other problems that the patient might be
    experiencing.
</p>
</div>
</div>
<div class="row margin-bottom-10">
<!-- Describe the symptom Form Input -->
<div class="medium- columns">
<label for="describe_the_symptom">Describe this symptom
     the patient is experiencing
{!! Form::textarea('describe_the_symptom', null,
['class' => 'default', 'id' => 'describe_the_symptom',
    'rows' => '3',
'placeholder' => '', 'aria-describedby' => '
    describe_helper']) !!}
</label>
<p class="help-text" id="describe_helper">
Describe the symptom that the patient is experiencing.
    Specify if how often does the symptom
occur or is it a recurring problem. You can also
    specify which body parts of the patient is
affected by this symptom. In addition, you can input
    the activities of the patient that causes
the symptom.
</p>
</div>
</div>
<div class="row margin-bottom-10">
<!-- What are the medicines taken by the patient? Form
    Input -->
<div class="medium-12 columns">
<label for="medicine_taken">What are the medicines
    taken by the patient?
{!! Form::textarea('medicine_taken', null,
['class' => 'default', 'id' => 'medicine_taken', 'rows'
     => '3', 'placeholder' => '',
'aria-describedby' => 'medicine_taken_helper']) !!}
</label>
<p class="help-text" id="medicine_taken_helper">
Enumerate the over-the-counter medicines that the
    patient is taking as a way of alleviating
```

```
the symptom. Further, include medicines that are not
     specified in the daily medication
regimen for the patient.
</p>
</div>
</div>
<div class="row margin-bottom-10">
<!-- What are the effects noted from the patient after
     taking these medicines? Form Input -->
<div class="medium-12 columns">
<label for="effects_noted">What are the effects noted
     from the patient after taking these medicines?
{!! Form::textarea('effects_noted', null,
['class' => 'default', 'id' => 'effects_noted', 'rows'
     => '3', 'placeholder' => '',
'aria-describedby' => 'effects_noted_helper']) !!}
</label>
<p class="help-text" id="effects_noted_helper">
Describe the effects noted from the patient after
     taking these medicines. Was there any
improvement? Or did the symptom experienced by the
     patient become worse?
</p>
</div>
</div>
<div class="row margin-bottom-10">
<!-- Are there any maneuvers taken by the patient in
     order to alleviate the problem? Form Input -->
<div class="medium-12 columns">
<label for="any_maneuver_taken">Are there any maneuvers
      taken by the patient in order to alleviate the
     problem?
{!! Form::textarea('any_maneuver_taken', null,
['class' => 'default', 'id' => '"any_maneuver_taken"',
     'rows' => '3', 'placeholder' => '',
'aria-describedby' => 'any_maneuver_taken_helper']) !!}
</label>
<p class="help-text" id="any_maneuver_taken_helper">
Describe the maneuvers taken by the patient in order to
      alleviate the symptom. For example,
if the patient experiences a back pain, does sitting on
      a chair supported by a pillow at the
back reduces the pain?
</p>
</div>
</div>
<div class="row margin-bottom-10">
<!-- Do you want to schedule for a follow-up visit?
     Form Input -->
<div class="medium-6 columns">
<fieldset>
<legend>Do you want to schedule for a follow-up visit
     ?</legend>
{!! Form::radio('set_schedule_for_visit', 'yes', false,
['id' => 'setScheduleForVisitYes']) !!}
<label for="setScheduleForVisitYes">Yes</label>
{!! Form::radio('set_schedule_for_visit', 'no', false,
['id' => 'setScheduleForVisitNo']) !!}
<label for="setScheduleForVisitNo">No</label>
</fieldset>
</div>
<!-- Date of visit Form Input -->
<div class="medium-6 columns">
<label for="date_of_visit">If yes, specify the date of
     visit
{!! Form::date('date_of_visit', null, ['class' => '
     default', 'id' => 'date_of_visit',
'min' => $current_date->format('Y-m-d')]) !!}
</label>
</div>
</div>
<!-- Submit Button -->
<div class="row column center margin-top-20">
{!! Form::submit('Submit Daily Medical Condition', ['
     class' => 'button', 'id' => 'submit_button']) !!}
<a href="{{ url('medical-conditions/' . $patient->id)
     }}" class="button hollow">
Cancel
</a>
</div>
{!! Form::close() !!}
</div>
</div>
</div>
@endsection
@section('js')@endsection
```

Medication Records Assessment Form View

```
<div class="callout">
<div class="row margin-bottom-10">
<!-- Diagnosis Form Input -->
<div class="medium-12 columns">
<label for="diagnosis">Diagnosis
{!! Form::textarea('diagnosis', $attr['diagnosis'],
['class' => 'default', 'id' => 'diagnosis', 'rows' => '
     5',
'placeholder' => '']) !!}
</label>
</div>
</div>
</div>
```

Medication Records Script View

```
<script type="application/javascript">
$(document).ready(function() {
var sum = parseInt($("input[name='latest_nihss_score']"
     ).val());

$('#nihss_input').on('change', function(){
sum = calculateNIHSSScore;
});

var calculate_nihss_score = $('#calculate_nihss_score')
     ;
$(calculate_nihss_score).click(function(e){
e.preventDefault();
$("input[name='latest_nihss_score']").val(sum);
});
```

```
var submit_button = $('#submit_button');
$(submit_button).click(function(){
$("input[name='latest_nihss_score']").val(
    calculateNIHSSScore);
return true;
});
});

function calculateNIHSSScore() {
var nihss_item = new Array(15);
var form = $('#stroke_and_medication_form');
nihss_item[0] = parseInt($("input[name='
    level_of_consciousness']:checked", form).val());
nihss_item[1] = parseInt($("input[name='loc_questions
    ']:checked", form).val());
nihss_item[2] = parseInt($("input[name='loc_commands']:
    checked", form).val());
nihss_item[3] = parseInt($("input[name='best_gaze']:
    checked", form).val());
nihss_item[4] = parseInt($("input[name='visual']:
    checked", form).val());
nihss_item[5] = parseInt($("input[name='facial_palsy']:
    checked", form).val());
nihss_item[6] = parseInt($("input[name='motor_arm_left
    ']:checked", form).val());
nihss_item[7] = parseInt($("input[name='motor_arm_right
    ']:checked", form).val());
nihss_item[8] = parseInt($("input[name='motor_leg_left
    ']:checked", form).val());
nihss_item[9] = parseInt($("input[name='motor_leg_right
    ']:checked", form).val());
nihss_item[10] = parseInt($("input[name='limb_ataxia']:
    checked", form).val());
nihss_item[11] = parseInt($("input[name='sensory']:
    checked", form).val());
nihss_item[12] = parseInt($("input[name='best_language
    ']:checked", form).val());
nihss_item[13] = parseInt($("input[name='dysarthia']:
    checked", form).val());
nihss_item[14] = parseInt($("input[name='
    extinction_and_inattention']:checked", form).val()
    );

var i;
var sum = 0;
for (i = 0; i < nihss_item.length; i++) {
if (! isNaN(nihss_item[i])) {
sum += nihss_item[i];
}
}

return sum;
}
</script>
```

Medication Records Objective Form View

```
<div class="callout">
<h5>Vital Signs</h5><hr>
<div class="row margin-bottom-10">
<!-- Weight (in kilograms) Form Input -->
<div class="large-3 columns">
<label for="weight">Weight (in kilograms)
{!! Form::text('weight', $attr['weight'],
['class' => 'default', 'id' => 'weight']) !!}
@foreach ($errors->get('weight') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('weight')) - 1)<br>
    @endif
@endforeach
</label>
</div>
<!-- Height (in centimeters) Form Input -->
<div class="large-3 columns">
<label for="height">Height (in centimeters)
{!! Form::text('height', $attr['height'],
['class' => 'default', 'id' => 'height']) !!}
@foreach ($errors->get('height') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('height')) - 1)<br>
    @endif
@endforeach
</label>
</div>
<!-- Systolic Blood Pressure Form Input -->
<div class="large-3 columns">
<label for="systolic_blood_pressure">Systolic Blood
    Pressure
{!! Form::text('systolic_blood_pressure', $attr['
    systolic_blood_pressure'],
['class' => 'default', 'id' => 'systolic_blood_pressure
    ']) !!}
@foreach ($errors->get('systolic_blood_pressure') as
    $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('
    systolic_blood_pressure')) - 1)<br>@endif
@endforeach
</label>
</div>
<!-- Diastolic Blood Pressure Form Input -->
<div class="large-3 columns">
<label for="diastolic_blood_pressure">Diastolic Blood
    Pressure
{!! Form::text('diastolic_blood_pressure', $attr['
    diastolic_blood_pressure'],
['class' => 'default', 'id' => '
    diastolic_blood_pressure']) !!}
@foreach ($errors->get('diastolic_blood_pressure') as
    $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('
    diastolic_blood_pressure')) - 1)<br>@endif
@endforeach
</label>
</div>
</div>
<div class="row margin-bottom-10">
<!-- Temperature Form Input -->
```

```
<div class="large-3 columns">
<label for="temperature">Temperature
{!! Form::text('temperature', $attr['temperature'], ['
    class' => 'default', 'id' => 'temperature']) !!}
</label>
</div>
<!-- Pulse Rate Form Input -->
<div class="large-3 columns">
<label for="pulse_rate">Pulse Rate
{!! Form::text('pulse_rate', $attr['pulse_rate'], ['
    class' => 'default', 'id' => 'pulse_rate']) !!}
</label>
</div>
<!-- Respiration Form Input -->
<div class="large-3 columns end">
<label for="respiration">Respiration
{!! Form::text('respiration', $attr['respiration'], ['
    class' => 'default', 'id' => 'respiration']) !!}
</label>
</div>
</div>
</div>
<div class="callout">
<h5>Reviewing the Systems</h5><hr>
<div class="row margin-bottom-10">
<!-- General Form Input -->
<div class="medium-12 columns">
<label for="general">General
{!! Form::textarea('general', $attr['general'],
['class' => 'default', 'id' => 'general', 'rows' => '3'
    ,
'placeholder' => '']) !!}
</label>
</div>
</div>
<div class="row margin-bottom-10">
<!-- Eyes Form Input -->
<div class="medium-12 columns">
<label for="eyes">Eyes
{!! Form::textarea('eyes', $attr['eyes'],
['class' => 'default', 'id' => 'eyes', 'rows' => '3',
'placeholder' => '']) !!}
</label>
</div>
</div>
<div class="row margin-bottom-10">
<!-- Ears, Nose, and Throat Form Input -->
<div class="medium-12 columns">
<label for="ears_nose_throat">Ears, Nose, and Throat
{!! Form::textarea('ears_nose_throat', $attr['
    ears_nose_throat'],
['class' => 'default', 'id' => 'ears_nose_throat', '
    rows' => '3',
'placeholder' => '']) !!}
</label>
</div>
</div>
<div class="row margin-bottom-10">
<!-- Cardiovascular Form Input -->
<div class="medium-12 columns">
<label for="cardiovascular">Cardiovascular
{!! Form::textarea('cardiovascular', $attr['
    cardiovascular'],
['class' => 'default', 'id' => 'cardiovascular', 'rows'
    => '3',
'placeholder' => '']) !!}
</label>
</div>
</div>
<div class="row margin-bottom-10">
<!-- Gastrointestinal Form Input -->
<div class="medium-12 columns">
<label for="gastrointestinal">Gastrointestinal
{!! Form::textarea('gastrointestinal', $attr['
    gastrointestinal'],
['class' => 'default', 'id' => 'gastrointestinal', '
    rows' => '3',
'placeholder' => '']) !!}
</label>
</div>
</div>
<div class="row margin-bottom-10">
<!-- Genitourinary Form Input -->
<div class="medium-12 columns">
<label for="genitourinary">Genitourinary
{!! Form::textarea('genitourinary', $attr['
    genitourinary'],
['class' => 'default', 'id' => 'genitourinary', 'rows'
    => '3',
'placeholder' => '']) !!}
</label>
</div>
</div>
<div class="row margin-bottom-10">
<!-- Musculoskeletal Form Input -->
<div class="medium-12 columns">
<label for="musculoskeletal">Musculoskeletal
{!! Form::textarea('musculoskeletal', $attr['
    musculoskeletal'],
['class' => 'default', 'id' => 'musculoskeletal', 'rows
    ' => '3',
'placeholder' => '']) !!}
</label>
</div>
</div>
<div class="row margin-bottom-10">
<!-- Skin Form Input -->
<div class="medium-12 columns">
<label for="skin">Skin
{!! Form::textarea('skin', $attr['skin'],
['class' => 'default', 'id' => 'skin', 'rows' => '3',
'placeholder' => '']) !!}
</label>
</div>
</div>
<div class="row margin-bottom-10">
<!-- Neurologic Form Input -->
<div class="medium-12 columns">
<label for="neurologic">Neurologic
{!! Form::textarea('neurologic', $attr['neurologic'],
```

```
['class' => 'default', 'id' => 'neurologic', 'rows' =>
    '3',
'placeholder' => '']) !!}
</label>
</div>
</div>
<div class="row margin-bottom-10">
<!-- Psychiatric Form Input -->
<div class="medium-12 columns">
<label for="psychiatric">Psychiatric
{!! Form::textarea('psychiatric', $attr['psychiatric'],
['class' => 'default', 'id' => 'psychiatric', 'rows' =>
    '3',
'placeholder' => '']) !!}
</label>
</div>
</div>
<div class="row margin-bottom-10">
<!-- Endocrine Form Input -->
<div class="medium-12 columns">
<label for="endocrine">Endocrine
{!! Form::textarea('endocrine', $attr['endocrine'],
['class' => 'default', 'id' => 'endocrine', 'rows' => '
    3',
'placeholder' => '']) !!}
</label>
</div>
</div>
<div class="row margin-bottom-10">
<!-- Lymphatic Form Input -->
<div class="medium-12 columns">
<label for="lymphatic">Lymphatic
{!! Form::textarea('lymphatic', $attr['lymphatic'],
['class' => 'default', 'id' => 'lymphatic', 'rows' => '
    3',
'placeholder' => '']) !!}
</label>
</div>
</div>
<div class="row margin-bottom-10">
<!-- Immunologic Form Input -->
<div class="medium-12 columns">
<label for="immunologic">Immunologic
{!! Form::textarea('immunologic', $attr['immunologic'],
['class' => 'default', 'id' => 'immunologic', 'rows' =>
    '3',
'placeholder' => '']) !!}
</label>
</div>
</div>
</div>
<div class="callout" id="nihss">
<h5>The National Institutes of Health Stroke Scale
    Score</h5>
<p class="help-text">
If applicable, input the latest NIHSS score of the
    patient.
</p><hr>
@foreach ($nihss_scale as $key => $nihss)
@if ($key == 'level_of_consciousness' or $key == '
    best_gaze' or $key == 'motor_arm_left' or
$key == 'motor_leg_left' or $key == 'limb_ataxia' or
    $key == 'dysarthia')
<div class="row margin-bottom-20">
@endif
<div class="@if ($key == 'motor_arm_left' or $key == "
    motor_arm_right" or
$key == "motor_leg_left" or $key == "motor_leg_right") 
    medium-6
@elseif ($key == 'extinction_and_inattention') medium-8
    @else medium-4 @endif columns">
<fieldset>
<legend>{{ $attr[$key][0] }}</legend>
@foreach ($attr[$key][1] as $index => $value)
{!! Form::radio($key, $index, $value,
['id' => $key . '-' . $index]) !!}
<label for="{{ $key . '-' . $index }}">{{ $index }}</
    label>
@endforeach
</fieldset>
</div>
@if ($key == 'loc_commands' or $key == 'facial_palsy'
    or $key == 'motor_arm_right' or
$key == 'motor_leg_right' or $key == 'best_language' or
$key == 'extinction_and_inattention')
</div>
@endif
@endforeach
<div class="row">
<div class="medium-5 columns">
<span class="hide-for-small-only"><br></span>
<button type="button" class="button" id="
    calculate_nihss_score">
Calculate NIHSS Score
</button>
</div>
<!-- Latest NIHSS score Form Input -->
<div class="medium-5 columns end">
{!! Form::hidden('latest_nihss_score', $attr['
    latest_nihss_score']) !!}
<label for="latest_nihss_score">NIHSS Score
{!! Form::text('latest_nihss_score', $attr['
    latest_nihss_score'],
['class' => 'default', 'id' => 'latest_nihss_score', '
    disabled' => 'disabled']) !!}
</label>
</div>
</div>
</div>
```

Medication Records Plan Form View

```
<div class="callout">
<div class="row margin-bottom-10">
<!-- Plans Form Input -->
<div class="medium-12 columns">
<label for="plans">Plans
{!! Form::textarea('plans', $attr['plans'],
['class' => 'default', 'id' => 'plans', 'rows' => '5',
'placeholder' => '']) !!}
```

```
</label>
</div>
</div>
</div>
```

Medication Records Subjective Form View

```
<div class="callout">
<div class="row margin-bottom-10">
<!-- Problems Form Input -->
<div class="medium-12 columns">
<label for="problems">Problems
{!! Form::textarea('problems', $attr['problems'],
['class' => 'default', 'id' => 'problems', 'rows' => '3
    ',
'placeholder' => '']) !!}
</label>
</div>
</div>
<div class="row margin-bottom-10">
<!-- Description Form Input -->
<div class="medium-12 columns">
<label for="description">Description
{!! Form::textarea('description', $attr['description'],
['class' => 'default', 'id' => 'description', 'rows' =>
    '3',
'placeholder' => '']) !!}
</label>
</div>
</div>
<div class="row margin-bottom-10">
<!-- Medications Form Input -->
<div class="medium-12 columns">
<label for="medications">Medications
{!! Form::textarea('medications', $attr['medications'],
['class' => 'default', 'id' => 'medications', 'rows' =>
    '3',
'placeholder' => '']) !!}
</label>
</div>
</div>
<div class="row margin-bottom-10">
<!-- Alleviating Factors Form Input -->
<div class="medium-12 columns">
<label for="alleviating_factors">Alleviating Factors
{!! Form::textarea('alleviating_factors', $attr['
    alleviating_factors'],
['class' => 'default', 'id' => 'alleviating_factors', '
    rows' => '3',
'placeholder' => '']) !!}
</label>
</div>
</div>
<div class="row margin-bottom-10">
<!-- Temporal Patterns Form Input -->
<div class="medium-12 columns">
<label for="temporal_patterns">Temporal Patterns
{!! Form::textarea('temporal_patterns', $attr['
    temporal_patterns'],
['class' => 'default', 'id' => 'temporal_patterns', '
    rows' => '3',
'placeholder' => '']) !!}
</label>
</div>
</div>
</div>
```

Medication Records Edit View

```
@extends('app')
@section('css')@endsection
@section('content')
<div class="row">
<div class="small-10 small-centered medium-10 medium-
    centered large-10 large-centered column">
<div class="center"><h4>Update Medical Record: {{
    $patient_full_name }}</h4></div>
<hr>
<div class="row column margin-bottom-10">
{!! Form::open(['method' => 'POST', 'url' => 'medical-
    records/' . $patient->id . '/' .
    $medication_record->id,
'id' => 'stroke_and_medication_form']) !!}
<ul class="accordion" data-accordion data-accordion
    data-multi-expand="true"
data-accordion data-allow-all-closed="true">
<li class="accordion-item is-active" data-accordion-
    item>
<a href="#" class="accordion-title font-125-rem">
    Subjective</a>
<div class="accordion-content" data-tab-content>
@include('medical_records.medication_records.
    _subjective_form')
</div>
</li>
<li class="accordion-item is-active" data-accordion-
    item>
<a href="#" class="accordion-title font-125-rem">
    Objective</a>
<div class="accordion-content" data-tab-content>
@include('medical_records.medication_records.
    _objective_form')
</div>
</li>
<li class="accordion-item is-active" data-accordion-
    item>
<a href="#" class="accordion-title font-125-rem">
    Assessment</a>
<div class="accordion-content" data-tab-content>
@include('medical_records.medication_records.
    _assessment_form')
</div>
</li>
<li class="accordion-item is-active" data-accordion-
    item>
<a href="#" class="accordion-title font-125-rem">Plan</
    a>
<div class="accordion-content" data-tab-content>
@include('medical_records.medication_records._plan_form
    ')
```

```
</div>
</li>
</ul>
<!-- Submit Button -->
<div class="row column center margin-top-20">
{!! Form::submit('Update Medical Record', ['class' => '
    button',
'id' => 'submit_button']) !!}
<a href="{{ url('medical-records/'. $patient->id) }}"
    class="button hollow">Cancel</a>
</div>
{!! Form::close() !!}
</div>
</div>
</div>
@endsection

@section('js')
<script type="text/javascript">
$('.accordion').foundation();
</script>
@include('medical_records.medication_records.
    _medication_record_script')
@endsection
```

Medication Records Index View

```
@extends('app')
@section('css'){!! Html::style('css/responsive-tables.
    css') !!}@endsection
@section('content')
<div class="row">
<div class="small-10 small-centered medium-10 medium-
    centered large-10 large-centered column">
<div class="center"><h4>Update Medical Records: {{
    $patient_full_name }}</h4></div>
<hr>
@if ($is_staff)
<div class="row column margin-bottom-20 menu-centered">
<ul class="vertical medium-horizontal menu">
<li><a href="{{ url('patients/'. $patient->id) }}">
    Account Profile</a></li>
@if ($user->medicalProfessional)
<li><a href="{{ url('patient-attributes/'. $patient->
    id) }}">General Details</a></li>
<li><a href="{{ url('medical-conditions/'. $patient->
    id) }}">Daily Medical Condition</a></li>
<li><a href="{{ url('laboratory-results/'. $patient->
    id) }}">Laboratory Results</a></li>
<li class="menu-text">Medical Records</li>
<li><a href="{{ url('referral/'. $patient->id) }}">
    Referral</a></li>
<li><a href="{{ url('medication-regimens/'. $patient->
    id) }}">Medication Regimens</a></li>
@endif
</ul>
</div>
@endif
<div class="row column margin-bottom-10">
<div class="callout">
@if (count($medication_records) > 0)
<ul>
@foreach ($medication_records as $medication_record)
<li>
<a href="{{ url('medical-records/'. $patient->id .
    '/'. $medication_record->id) }}">
Medical Record: {{ $medication_record['created_at']->
    format('F, d Y, g:i:A') }}
</a>
</li>
@endforeach
</ul>
@else
<p>The patient has no medical records.</p>
@endif
</div>
<div class="row column center">
{!! Form::open(['method' => 'POST', 'url' => 'medical-
    records/' . $patient->id]) !!}
{!! Form::submit('Add New Medical Record', ['class' =>
    'button']) !!}
{!! Form::close() !!}
</div>
</div>
</div>
</div>
@endsection
@section('js'){!! Html::script('js/responsive-tables.js
    ') !!}@endsection
```

Nutritionist Records Assessment Form View

```
<div class="callout">
<div class="row margin-bottom-10">
<!-- Total Energy Intake Form Input -->
<div class="medium-12 columns">
<label for="total_energy_intake">Total Energy Intake
{!! Form::textarea('total_energy_intake', $attr['
    total_energy_intake'],
['class' => 'default', 'id' => 'total_energy_intake', '
    rows' => '3',
'placeholder' => '']) !!}
</label>
</div>
</div>
<div class="row margin-bottom-10">
<!-- Total Fat Intake and Saturated Fat Intake Form
    Input -->
<div class="medium-12 columns">
<label for="total_fat_intake_and_saturated_fat_intake">
    Total Fat Intake and Saturated Fat Intake
{!! Form::textarea('
    total_fat_intake_and_saturated_fat_intake', $attr[
    'total_fat_intake_and_saturated_fat_intake'],
['class' => 'default', 'id' => '
    total_fat_intake_and_saturated_fat_intake', 'rows'
     => '3',
'placeholder' => '']) !!}
</label>
</div>
```

```
</div>
<div class="row margin-bottom-10">
<!-- Body Compartment Estimates Form Input -->
<div class="medium-12 columns">
<label for="body_compartment_estimates">Body
    Compartment Estimates
{!! Form::textarea('body_compartment_estimates', $attr[
    'body_compartment_estimates'],
['class' => 'default', 'id' => '
    body_compartment_estimates', 'rows' => '3',
'placeholder' => '']) !!}
</label>
</div>
</div>
<div class="row margin-bottom-10">
<!-- Recommended Body Weight Form Input -->
<div class="medium-12 columns">
<label for="recommended_body_weight">Recommended Body
    Weight
{!! Form::textarea('recommended_body_weight', $attr['
    recommended_body_weight'],
['class' => 'default', 'id' => 'recommended_body_weight
    ', 'rows' => '3',
'placeholder' => '']) !!}
</label>
</div>
</div>
<div class="row margin-bottom-10">
<!-- Estimated Energy Needs Form Input -->
<div class="medium-12 columns">
<label for="estimated_energy_needs">Estimated Energy
    Needs
{!! Form::textarea('estimated_energy_needs', $attr['
    estimated_energy_needs'],
['class' => 'default', 'id' => 'estimated_energy_needs'
    , 'rows' => '3',
'placeholder' => '']) !!}
</label>
</div>
</div>
<div class="row margin-bottom-10">
<!-- Nutrition Diagnosis Form Input -->
<div class="medium-12 columns">
<label for="nutrition_diagnosis">Nutrition Diagnosis
{!! Form::textarea('nutrition_diagnosis', $attr['
    nutrition_diagnosis'],
['class' => 'default', 'id' => 'nutrition_diagnosis', '
    rows' => '5',
'placeholder' => '']) !!}
</label>
</div>
</div>
</div>
```

Nutritionist Records Objective Form View

```
<div class="callout">
<h5>Vital Signs</h5><hr>
<div class="row margin-bottom-10">
<!-- Weight (in kilograms) Form Input -->
<div class="large-3 columns">
<label for="weight">Weight (in kilograms)
{!! Form::text('weight', $attr['weight'],
['class' => 'default', 'id' => 'weight']) !!}
@foreach ($errors->get('weight') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('weight')) - 1)<br>
    @endif
@endforeach
</label>
</div>
<!-- Height (in centimeters) Form Input -->
<div class="large-3 columns">
<label for="height">Height (in centimeters)
{!! Form::text('height', $attr['height'],
['class' => 'default', 'id' => 'height']) !!}
@foreach ($errors->get('height') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('height')) - 1)<br>
    @endif
@endforeach
</label>
</div>
<!-- BMI Form Input -->
<div class="large-3 columns end">
<label for="bmi">BMI
{!! Form::text('bmi', $attr['bmi'], ['class' => '
    default', 'id' => 'bmi']) !!}
</label>
</div>
</div>
</div>
```

Nutritionist Records Plan Form View

```
<div class="callout">
<div class="row margin-bottom-10">
<!-- Nutrition Plan Form Input -->
<div class="medium-12 columns">
<label for="nutrition_plan">Nutrition Plan
{!! Form::textarea('nutrition_plan', $attr['
    nutrition_plan'],
['class' => 'default', 'id' => 'nutrition_plan', 'rows'
    => '5',
'placeholder' => '']) !!}
</label>
</div>
</div>
</div>
```

Nutritionist Records Subjective Form View

```
<div class="callout">
<div class="row margin-bottom-10">
<!-- Problems Form Input -->
<div class="medium-12 columns">
<label for="problems">Problems
{!! Form::textarea('problems', $attr['problems'],
['class' => 'default', 'id' => 'problems', 'rows' => '3
    ',
```

```
'placeholder' => '']) !!}
</label>
</div>
</div>
<div class="row margin-bottom-10">
<!-- Meal Snack Pattern Form Input -->
<div class="medium-12 columns">
<label for="meal_snack_pattern">Meal Snack Pattern
{!! Form::textarea('meal_snack_pattern', $attr['
    meal_snack_pattern'],
['class' => 'default', 'id' => 'meal_snack_pattern', '
    rows' => '3',
'placeholder' => '']) !!}
</label>
</div>
</div>
<div class="row margin-bottom-10">
<!-- Food Intake Form Input -->
<div class="medium-12 columns">
<label for="food_intake">Food Intake
{!! Form::textarea('food_intake', $attr['food_intake'],
['class' => 'default', 'id' => 'food_intake', 'rows' =>
    '3',
'placeholder' => '']) !!}
</label>
</div>
</div>
<div class="row margin-bottom-10">
<!-- Alcohol Intake Form Input -->
<div class="medium-12 columns">
<label for="alcohol_intake">Alcohol Intake
{!! Form::textarea('alcohol_intake', $attr['
    alcohol_intake'],
['class' => 'default', 'id' => 'alcohol_intake', 'rows'
    => '3',
'placeholder' => '']) !!}
</label>
</div>
</div>
<div class="row margin-bottom-10">
<!-- Dietary Supplements Form Input -->
<div class="medium-12 columns">
<label for="dietary_supplements">Dietary Supplements
{!! Form::textarea('dietary_supplements', $attr['
    dietary_supplements'],
['class' => 'default', 'id' => 'dietary_supplements', '
    rows' => '3',
'placeholder' => '']) !!}
</label>
</div>
</div>
<div class="row margin-bottom-10">
<!-- Readiness to Change Nutrition-Related Behaviors
    Form Input -->
<div class="medium-12 columns">
<label for="readiness_to_change">Readiness to Change
    Nutrition-Related Behaviors
{!! Form::textarea('readiness_to_change', $attr['
    readiness_to_change'],
['class' => 'default', 'id' => 'readiness_to_change', '
    rows' => '3',
'placeholder' => '']) !!}
</label>
</div>
</div>
<div class="row margin-bottom-10">
<!-- Weight Change Form Input -->
<div class="medium-12 columns">
<label for="weight_change">Weight Change
{!! Form::textarea('weight_change', $attr['
    weight_change'],
['class' => 'default', 'id' => 'weight_change', 'rows'
    => '3',
'placeholder' => '']) !!}
</label>
</div>
</div>
<div class="row margin-bottom-10">
<!-- Physical Activity History Form Input -->
<div class="medium-12 columns">
<label for="physical_activity_history">Physical
    Activity History
{!! Form::textarea('physical_activity_history', $attr['
    physical_activity_history'],
['class' => 'default', 'id' => '
    physical_activity_history', 'rows' => '3',
'placeholder' => '']) !!}
</label>
</div>
</div>
</div>
```

Nutritionist Records Edit View

```
@extends('app')
@section('css')@endsection
@section('content')
<div class="row">
<div class="small-10 small-centered medium-10 medium-
    centered large-10 large-centered column">
<div class="center"><h4>Nutrition Record: {{
    $patient_full_name }}</h4></div>
<hr>
<div class="row column margin-bottom-10">
{!! Form::open(['method' => 'POST',
'url' => 'nutritionist-records/' . $patient->id . '/' .
    $nutritionist_record->id]) !!}
<ul class="accordion" data-accordion data-accordion
    data-multi-expand="true"
data-accordion data-allow-all-closed="true">
<li class="accordion-item is-active" data-accordion-
    item>
<a href="#" class="accordion-title font-125-rem">
    Subjective</a>
<div class="accordion-content" data-tab-content>
@include('medical_records.nutritionist_records.
    _subjective_form')
</div>
</li>
<li class="accordion-item is-active" data-accordion-
```

```
    item>
<a href="#" class="accordion-title␣font-125-rem">
    Objective</a>
<div class="accordion-content" data-tab-content>
@include('medical_records.nutritionist_records.
    _objective_form')
</div>
</li>
<li class="accordion-item␣is-active" data-accordion-
    item>
<a href="#" class="accordion-title␣font-125-rem">
    Assessment</a>
<div class="accordion-content" data-tab-content>
@include('medical_records.nutritionist_records.
    _assessment_form')
</div>
</li>
<li class="accordion-item␣is-active" data-accordion-
    item>
<a href="#" class="accordion-title␣font-125-rem">Plan</
    a>
<div class="accordion-content" data-tab-content>
@include('medical_records.nutritionist_records.
    _plan_form')
</div>
</li>
</ul>
<!-- Submit Button -->
<div class="row␣column␣center␣margin-top-20">
{!! Form::submit('Update␣Nutrition␣Record', ['class' =>
     'button']) !!}
<a href="{{␣url('nutritionist-records/'␣.␣$patient->id)
    ␣}}" class="button␣hollow">Cancel</a>
</div>
{!! Form::close() !!}
</div>
</div>
</div>
@endsection
@section('js')
<script type="text/javascript">
$('.accordion').foundation();
</script>
@endsection
```

Nutritionist Records Index View

```
@extends('app')
@section('css'){!! Html::style('css/responsive-tables.
    css') !!}@endsection
@section('content')
<div class="row">
<div class="small-10␣small-centered␣medium-10␣medium-
    centered␣large-10␣large-centered␣column">
<div class="center"><h4>Update Nutrition Records: {{
    $patient_full_name }}</h4></div>
<hr>
@if ($is_staff)
<div class="row␣column␣margin-bottom-20␣menu-centered">
<ul class="vertical␣medium-horizontal␣menu">
<li><a href="{{␣url('patients/'␣.␣$patient->id)␣}}">
    Account Profile</a></li>
@if ($user->nutritionist)
<li class="menu-text">Nutrition Records</li>
<li><a href="{{␣url('nutritionist-regimens/'␣.␣$patient
    ->id)␣}}">Nutrition Regimens</a></li>
@endif
</ul>
</div>
@endif
<div class="row␣column␣margin-bottom-10">
<div class="callout">
@if (count($nutritionist_records) > 0)
<ul>
@foreach ($nutritionist_records as $nutritionist_record
    )
<li>
<a href="{{␣url('nutritionist-records/'␣.␣$patient->id␣
    .␣'/'.␣$nutritionist_record->id)␣}}">
Nutrition Record: {{ $nutritionist_record['created_at'
    ]->format('F,␣d␣Y,␣g:i:A') }}
</a>
</li>
@endforeach
</ul>
@else
<p>The patient has no nutrition records.</p>
@endif
</div>
<div class="row␣column␣center">
{!! Form::open(['method' => 'POST', 'url' => '
    nutritionist-records/' . $patient->id]) !!}
{!! Form::submit('Add␣New␣Nutrition␣Record', ['class'
    => 'button']) !!}
{!! Form::close() !!}
</div>
</div>
</div>
</div>
@endsection
@section('js'){!! Html::script('js/responsive-tables.js
    ') !!}@endsection
```

Physiatrist Records Assessment Form View

```
<div class="callout">
<div class="row␣margin-bottom-10">
<!-- Diagnosis Form Input -->
<div class="medium-12␣columns">
<label for="diagnosis">Diagnosis
{!! Form::textarea('diagnosis', $attr['diagnosis'],
['class' => 'default', 'id' => 'diagnosis', 'rows' => '
    5',
'placeholder' => '']) !!}
</label>
</div>
</div>
</div>
```

Physiatrist Records Objective Form View

```
<div class="callout">
<h5>Vital Signs</h5><hr>
<div class="row margin-bottom-10">
<!-- Weight (in kilograms) Form Input -->
<div class="large-3 columns">
<label for="weight">Weight (in kilograms)
{!! Form::text('weight', $attr['weight'],
['class' => 'default', 'id' => 'weight']) !!}
@foreach ($errors->get('weight') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('weight')) - 1)<br>
    @endif
@endforeach
</label>
</div>
<!-- Height (in centimeters) Form Input -->
<div class="large-3 columns">
<label for="height">Height (in centimeters)
{!! Form::text('height', $attr['height'],
['class' => 'default', 'id' => 'height']) !!}
@foreach ($errors->get('height') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('height')) - 1)<br>
    @endif
@endforeach
</label>
</div>
<!-- Systolic Blood Pressure Form Input -->
<div class="large-3 columns">
<label for="systolic_blood_pressure">Systolic Blood
    Pressure
{!! Form::text('systolic_blood_pressure', $attr['
    systolic_blood_pressure'],
['class' => 'default', 'id' => 'systolic_blood_pressure
    ']) !!}
@foreach ($errors->get('systolic_blood_pressure') as
    $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('
    systolic_blood_pressure')) - 1)<br>@endif
@endforeach
</label>
</div>
<!-- Diastolic Blood Pressure Form Input -->
<div class="large-3 columns">
<label for="diastolic_blood_pressure">Diastolic Blood
    Pressure
{!! Form::text('diastolic_blood_pressure', $attr['
    diastolic_blood_pressure'],
['class' => 'default', 'id' => '
    diastolic_blood_pressure']) !!}
@foreach ($errors->get('diastolic_blood_pressure') as
    $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('
    diastolic_blood_pressure')) - 1)<br>@endif
@endforeach
</label>
</div>
</div>
<div class="row margin-bottom-10">
<!-- Temperature Form Input -->
<div class="large-3 columns">
<label for="temperature">Temperature
{!! Form::text('temperature', $attr['temperature'], ['
    class' => 'default', 'id' => 'temperature']) !!}
</label>
</div>
<!-- Pulse Rate Form Input -->
<div class="large-3 columns">
<label for="pulse_rate">Pulse Rate
{!! Form::text('pulse_rate', $attr['pulse_rate'], ['
    class' => 'default', 'id' => 'pulse_rate']) !!}
</label>
</div>
<!-- Respiration Form Input -->
<div class="large-3 columns end">
<label for="respiration">Respiration
{!! Form::text('respiration', $attr['respiration'], ['
    class' => 'default', 'id' => 'respiration']) !!}
</label>
</div>
</div>
</div>
<div class="callout">
<h5>Reviewing the Systems</h5><hr>
<div class="row margin-bottom-10">
<!-- General Form Input -->
<div class="medium-12 columns">
<label for="general">General
{!! Form::textarea('general', $attr['general'],
['class' => 'default', 'id' => 'general', 'rows' => '3'
    ,
'placeholder' => '']) !!}
</label>
</div>
</div>
<div class="row margin-bottom-10">
<!-- Head, Eyes, Ears, Neck, and Throat Form Input -->
<div class="medium-12 columns">
<label for="head_eyes_ears_neck_throat">Head, Eyes,
    Ears, Neck, and Throat
{!! Form::textarea('head_eyes_ears_neck_throat', $attr[
    'head_eyes_ears_neck_throat'],
['class' => 'default', 'id' => '
    head_eyes_ears_neck_throat', 'rows' => '3',
'placeholder' => '']) !!}
</label>
</div>
</div>
<div class="row margin-bottom-10">
<!-- Lungs Form Input -->
<div class="medium-12 columns">
<label for="lungs">Lungs
{!! Form::textarea('lungs', $attr['lungs'],
['class' => 'default', 'id' => 'lungs', 'rows' => '3',
'placeholder' => '']) !!}
</label>
</div>
</div>
```

```
<div class="row margin-bottom-10">
<!-- Cardiovascular Form Input -->
<div class="medium-12 columns">
<label for="cardiovascular">Cardiovascular
{!! Form::textarea('cardiovascular', $attr['
    cardiovascular'],
['class' => 'default', 'id' => 'cardiovascular', 'rows'
    => '3',
'placeholder' => '']) !!}
</label>
</div>
</div>
<div class="row margin-bottom-10">
<!-- Abdomen Form Input -->
<div class="medium-12 columns">
<label for="abdomen">Abdomen
{!! Form::textarea('abdomen', $attr['abdomen'],
['class' => 'default', 'id' => 'abdomen', 'rows' => '3'
    ,
'placeholder' => '']) !!}
</label>
</div>
</div>
<div class="row margin-bottom-10">
<!-- Extremities Form Input -->
<div class="medium-12 columns">
<label for="extremities">Extremities
{!! Form::textarea('extremities', $attr['extremities'],
['class' => 'default', 'id' => 'extremities', 'rows' =>
    '3',
'placeholder' => '']) !!}
</label>
</div>
</div>
</div>
```

Physiatrist Records Plan Form View

```
<div class="callout">
<div class="row margin-bottom-10">
<!-- Rehabilitation Medicine Plan Form Input -->
<div class="medium-12 columns">
<label for="rehabilitation_medicine_plan">
    Rehabilitation Medicine Plan
{!! Form::textarea('rehabilitation_medicine_plan',
    $attr['rehabilitation_medicine_plan'],
['class' => 'default', 'id' => '
    rehabilitation_medicine_plan', 'rows' => '5',
'placeholder' => '']) !!}
</label>
</div>
</div>
</div>
```

Physiatrist Records Subjective Form View

```
<div class="callout">
<div class="row margin-bottom-10">
<!-- Problems Form Input -->
<div class="medium-12 columns">
<label for="problems">Problems
{!! Form::textarea('problems', $attr['problems'],
['class' => 'default', 'id' => 'problems', 'rows' => '3
    ',
'placeholder' => '']) !!}
</label>
</div>
</div>
<div class="row margin-bottom-10">
<!-- Description Form Input -->
<div class="medium-12 columns">
<label for="description">Description
{!! Form::textarea('description', $attr['description'],
['class' => 'default', 'id' => 'description', 'rows' =>
    '3',
'placeholder' => '']) !!}
</label>
</div>
</div>
<div class="row margin-bottom-10">
<!-- Medications Form Input -->
<div class="medium-12 columns">
<label for="medications">Medications
{!! Form::textarea('medications', $attr['medications'],
['class' => 'default', 'id' => 'medications', 'rows' =>
    '3',
'placeholder' => '']) !!}
</label>
</div>
</div>
<div class="row margin-bottom-10">
<!-- Alleviating Factors Form Input -->
<div class="medium-12 columns">
<label for="alleviating_factors">Alleviating Factors
{!! Form::textarea('alleviating_factors', $attr['
    alleviating_factors'],
['class' => 'default', 'id' => 'alleviating_factors', '
    rows' => '3',
'placeholder' => '']) !!}
</label>
</div>
</div>
<div class="row margin-bottom-10">
<!-- Temporal Patterns Form Input -->
<div class="medium-12 columns">
<label for="temporal_patterns">Temporal Patterns
{!! Form::textarea('temporal_patterns', $attr['
    temporal_patterns'],
['class' => 'default', 'id' => 'temporal_patterns', '
    rows' => '3',
'placeholder' => '']) !!}
</label>
</div>
</div>
</div>
```

Physiatrist Records Edit View

```
@extends('app')
@section('css')@endsection
```

```
@section('content')
<div class="row">
<div class="small-10 small-centered medium-10 medium-
    centered large-10 large-centered column">
<div class="center"><h4>Rehabilitation Medicine Record:
     {{ $patient_full_name }}</h4></div>
<hr>
<div class="row column margin-bottom-10">
{!! Form::open(['method' => 'POST',
'url' => 'physiatrist-records/' . $patient->id . '/' .
    $physiatrist_record->id]) !!}
<ul class="accordion" data-accordion data-accordion
    data-multi-expand="true"
data-accordion data-allow-all-closed="true">
<li class="accordion-item is-active" data-accordion-
    item>
<a href="#" class="accordion-title font-125-rem">
    Subjective</a>
<div class="accordion-content" data-tab-content>
@include('medical_records.physiatrist_records.
    _subjective_form')
</div>
</li>
<li class="accordion-item is-active" data-accordion-
    item>
<a href="#" class="accordion-title font-125-rem">
    Objective</a>
<div class="accordion-content" data-tab-content>
@include('medical_records.physiatrist_records.
    _objective_form')
</div>
</li>
<li class="accordion-item is-active" data-accordion-
    item>
<a href="#" class="accordion-title font-125-rem">
    Assessment</a>
<div class="accordion-content" data-tab-content>
@include('medical_records.physiatrist_records.
    _assessment_form')
</div>
</li>
<li class="accordion-item is-active" data-accordion-
    item>
<a href="#" class="accordion-title font-125-rem">Plan</
    a>
<div class="accordion-content" data-tab-content>
@include('medical_records.physiatrist_records.
    _plan_form')
</div>
</li>
</ul>
<!-- Submit Button -->
<div class="row column center margin-top-20">
{!! Form::submit('Update Rehabilitation Medicine Record
    ', ['class' => 'button']) !!}
<a href="{{ url('physiatrist-records/' . $patient->id)
    }}" class="button hollow">Cancel</a>
</div>
{!! Form::close() !!}
</div>
</div>
</div>
@endsection
@section('js')
<script type="text/javascript">
$('.accordion').foundation();
</script>
@endsection
```

Physiatrist Records Index View

```
@extends('app')
@section('css'){!! Html::style('css/responsive-tables.
    css') !!}@endsection
@section('content')
<div class="row">
<div class="small-10 small-centered medium-10 medium-
    centered large-10 large-centered column">
<div class="center"><h4>Update Rehabilitation Medicine
    Records: {{ $patient_full_name }}</h4></div>
<hr>
@if ($is_staff)
<div class="row column margin-bottom-20 menu-centered">
<ul class="vertical medium-horizontal menu">
<li><a href="{{ url('patients/' . $patient->id) }}">
    Account Profile</a></li>
@if ($user->physiatrist)
<li class="menu-text">Rehabilitation Medicine Records</
    li>
<li><a href="{{ url('physiatrist-regimens/' . $patient
    ->id) }}">Rehabilitation Medicine Regimens</a></li
    >
@endif
</ul>
</div>
@endif
<div class="row column margin-bottom-10">
<div class="callout">
@if (count($physiatrist_records) > 0)
<ul>
@foreach ($physiatrist_records as $physiatrist_record)
<li>
<a href="{{ url('physiatrist-records/' . $patient->id .
    '/' . $physiatrist_record->id) }}">
Rehabilitation Medicine Record: {{ $physiatrist_record[
    'created_at']->format('F, d Y, g:i:A') }}
</a>
</li>
@endforeach
</ul>
@else
<p>The patient has no rehabilitation medicine records
    .</p>
@endif
</div>
<div class="row column center">
{!! Form::open(['method' => 'POST', 'url' => '
    physiatrist-records/' . $patient->id]) !!}
{!! Form::submit('Add New Rehabilitation Medicine
    Record', ['class' => 'button']) !!}
```

```
{!! Form::close() !!}
</div>
</div>
</div>
</div>
@endsection
@section('js'){!! Html::script('js/responsive-tables.js
    ') !!}@endsection
```

Medical Resources Form View

```
<div class="callout">
<div class="row column">
<h5>Medical Resource Details</h5>
<p class="help-text">
Answer the following fields about the medical resource.
</p>
<hr>
</div>
<div class="row margin-bottom-10">
<!-- Title Form Input -->
<div class="medium-6 columns">
<label for="title">Title*
{!! Form::text('title', $medical_resource->title, ['
    class' => 'default',
'id' => 'title']) !!}
@foreach ($errors->get('title') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('title')) - 1)<br>
    @endif
@endforeach
</label>
</div>
<!-- Date Published Form Input -->
<div class="medium-3 columns">
<label for="date_published">Date Published
{!! Form::date('date_published',
$medical_resource->date_published[0]->format('Y-m-d'),
['class' => 'default', 'id' => 'date_published', 'max'
    => $current_date->format('Y-m-d')]) !!}
@foreach ($errors->get('date_published') as $key =>
    $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('date_published')) - 1)
    <br>@endif
@endforeach
</label>
</div>
<!-- Category Form Input -->
<div class="medium-3 columns">
<label for="category">Category
{!! Form::select('category', $forum_categories, null,
['class' => 'default', 'id' => 'category']) !!}
@foreach ($errors->get('category') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('category')) - 1)<br>
    @endif
@endforeach
</label>
</div>
</div>
<div class="row margin-bottom-10">
<!-- Description Form Input -->
<div class="medium-12 columns end">
<label for="description">Description
{!! Form::textarea('description', $medical_resource->
    description,
['class' => 'default', 'id' => 'description', 'rows' =>
    '3',
'placeholder' => 'Input the description of the medical
    resource.']) !!}
@foreach ($errors->get('description') as $key => $error
    )
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('description')) - 1)<br
    >@endif
@endforeach
</label>
</div>
</div>
<div class="row margin-bottom-10">
<!-- Authors Form Input -->
<div class="medium-6 columns">
<label for="authors">Authors*
{!! Form::text('authors', $medical_resource->authors, [
    'class' => 'default',
'id' => 'authors']) !!}
@foreach ($errors->get('authors') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('authors')) - 1)<br>
    @endif
@endforeach
</label>
</div>
<!-- Publisher Form Input -->
<div class="medium-6 columns">
<label for="publisher">Publisher
{!! Form::text('publisher', $medical_resource->
    publisher, ['class' => 'default',
'id' => 'publisher']) !!}
@foreach ($errors->get('publisher') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('publisher')) - 1)<br>
    @endif
@endforeach
</label>
</div>
</div>
</div>
<div class="callout">
<div class="row column">
<h5>Medical Resource File</h5>
<p class="help-text">
Upload the medical resource files here. These include
    images, videos, PDF files, etc.
In case of multiple files, use data compression tools
    to compress them in an archive
file (e.g., ZIP, RAR). The maximum size for a file is
    10MB.
</p>
```

```
</div>
<hr>
@if ($medical_resource->file)
<div class="row column margin-bottom-10">
<p class="font-0875rem">
Currently Stored File:
@if ($medical_resource->file != null)
@if (in_array($medical_resource->file->extension,
    $image_extensions)
or in_array($medical_resource->file->extension,
    $video_extensions)
or $medical_resource->file->extension == 'pdf')
<a data-open="file_0">
{{ $medical_resource->file->filename }}
</a>
@else
<a href="{{ url('medical-resources/' .
    $medical_resource->id . '/download') }}">
{{ $laboratory_result->file->filename }}
</a>
@endif
@else <em>n/a</em>
@endif
</p>
<p class="help-text">
Note: The stored file will be deleted if a new file is
    uploaded below then submitted.
</p>
@if ($medical_resource->file and
(in_array($medical_resource->file->extension,
    $image_extensions))
or (in_array($medical_resource->file->extension,
    $video_extensions))
or ($medical_resource->file->extension == 'pdf'))
<div class="reveal large" id="file_0" data-reveal
data-animation-in="fade-in" data-animation-out="fade-
    out">
<h4>{{ $medical_resource->title }}</h4>
<div class="row column center margin-top-20 margin-
    bottom-20">
@if (in_array($medical_resource->file->extension,
    $image_extensions))
<img src="{{ url($medical_resource->file->path . '/' .
$medical_resource->file->filename) }}">
@elseif (in_array($medical_resource->file->extension,
    $video_extensions))
<video height="500rem" controls>
@if ($medical_resource->file->extension == 'mp4')
<source src="{{ url($medical_resource->file->path . '/'
    .
$medical_resource->file->filename) }}" type="video/mp4"
    >
@elseif ($medical_resource->file->extension == 'webm')
<source src="{{ url($medical_resource->file->path . '/'
    .
$medical_resource->file->filename) }}" type="video/webm
    ">
@elseif ($medical_resource->file->extension == 'ogg')
<source src="{{ url($medical_resource->file->path . '/'
    .
$medical_resource->file->filename) }}" type="video/ogg"
    >
@endif
Your browser does not support the video tag.
</video>
@else
<object data="{{ url($medical_resource->file->path . '
    /' .
$medical_resource->file->filename) }}" type="
    application/pdf" width="100%"
height="500rem">
</object>
@endif
</div>
<div class="row column center">
<a href="{{ url('medical-resources/' .
    $medical_resource->id . '/download') }}"
class="button">
Download
</a>
<a class="button hollow" data-close>Close</a>
</div>
<button class="close-button" data-close aria-label="
    Close modal" type="button">
<span aria-hidden="true">&times;</span>
</button>
</div>
@endif
</div>
@endif
<div class="row margin-bottom-10">
<!-- File Upload Form Input -->
<div class="large-2 columns">
<label for="file" class="button">Upload File</label>
{!! Form::file('file', ['id' => 'file', 'class' => '
    show-for-sr',]) !!}
</div>
<div class="large-6 medium-6 columns end">
{!! Form::text('filename', null, ['id' => 'filename', '
    class' => 'default',
'disabled' => 'disabled']) !!}
@foreach ($errors->get('file') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('file')) - 1)<br>@endif
@endforeach
</div>
</div>
</div>
<div class="row column">
<p class="help-text">Fields marked with * are mandatory
    .</p>
</div>
<!-- Submit Button -->
<div class="row column center">
{!! Form::submit($submit_button_text, ['class' => '
    button',]) !!}
@if ($medical_resource->title)
<a class="button hollow" href="{{ url('medical-
    resources/' . $medical_resource->id) }}">
Cancel
```

```
</a>
@else
<a class="button hollow" href="{{ url('medical-
    resources') }}">
Cancel
</a>
@endif
</div>
```

Medical Resources Script View

```
<script type="application/javascript">
$(document).ready(function() {
$('input[type=file]').change(function(){
$(this).parent().next().find('input[type="text"]').val(
    $(this).val().split('\\').pop());
});
});
</script>
```

Medical Resources Create View

```
@extends('app')
@section('css')@endsection
@section('content')
<div class="row">
<div class="small-10 small-centered medium-10 medium-
    centered large-10 large-centered column">
<div class="center"><h4>Add Medical Resource</h4></div>
<hr>
<div class="row column margin-bottom-20 menu-centered">
<ul class="vertical medium-horizontal menu">
<li><a href="{{ url('medical-resources') }}">View All
    Medical Resources</a></li>
<li>
<a href="{{ url('medical-resources/category/Neurology')
    }}">
View Medical Resources on Neurology
</a>
</li>
<li>
<a href="{{ url('medical-resources/category/Nutrition')
    }}">
View Medical Resources on Nutrition
</a>
</li>
<li>
<a href="{{ url('medical-resources/category/
    Rehabilitation%20Medicine') }}">
View Medical Resources on Rehabilitation Medicine
</a>
</li>
</ul>
</div>
<div class="row column margin-bottom-10">
{!! Form::model($medical_resource = new \App\Models\
    MedicalResource, ['method' => 'POST',
'url' => 'medical-resources', 'files' => true]) !!}
@include('medical_resources._form',
['submit_button_text' => 'Add Medical Resource'])
{!! Form::close() !!}
</div>
</div>
</div>
@endsection
@section('js')
@include('medical_resources._script')
@endsection
```

Medical Resources Edit View

```
@extends('app')
@section('css')@endsection
@section('content')
<div class="row">
<div class="small-10 small-centered medium-10 medium-
    centered large-10 large-centered column">
<div class="center"><h4>Edit Medical Resource</h4></div
    >
<hr>
<div class="row column margin-bottom-20 menu-centered">
<ul class="vertical medium-horizontal menu">
<li><a href="{{ url('medical-resources') }}">View All
    Medical Resources</a></li>
<li>
<a href="{{ url('medical-resources/category/Neurology')
    }}">
View Medical Resources on Neurology
</a>
</li>
<li>
<a href="{{ url('medical-resources/category/Nutrition')
    }}">
View Medical Resources on Nutrition
</a>
</li>
<li>
<a href="{{ url('medical-resources/category/
    Rehabilitation%20Medicine') }}">
View Medical Resources on Rehabilitation Medicine
</a>
</li>
</ul>
</div>
<div class="row column margin-bottom-10">
{!! Form::model($medical_resource, ['method' => 'PATCH'
    ,
'url' => 'medical-resources/' . $medical_resource->id,
    'files' => true]) !!}
@include('medical_resources._form',
['submit_button_text' => 'Edit Medical Resource'])
{!! Form::close() !!}
</div>
</div>
</div>
@endsection
@section('js')
<script type="text/javascript">
$('.reveal').foundation();
</script>
```

```
@include('medical_resources._script')
@endsection
```

Medical Resources Index View

```
@extends('app')
@section('css')@endsection
@section('content')
<div class="row">
<div class="small-12 small-centered medium-12 medium-
    centered large-12 large-centered column">
<div class="center"><h4>Medical Resources</h4></div>
<hr>
<div class="row column margin-bottom-20 menu-centered">
<ul class="vertical medium-horizontal menu">
@if (isset($index))
<li class="menu-text">View All Medical Resources</li>
@else
<li><a href="{{ url('medical-resources') }}">View All
    Medical Resources</a></li>
@endif

@if (isset($category))
@if (strcmp($category, 'Neurology') == 0)
<li class="menu-text">View Medical Resources on
    Neurology</li>
@else
<li>
<a href="{{ url('medical-resources/category/Neurology')
    }}">
View Medical Resources on Neurology
</a>
</li>
@endif

@if (strcmp($category, 'Nutrition') == 0)
<li class="menu-text">View Medical Resources on
    Nutrition</li>
@else
<li>
<a href="{{ url('medical-resources/category/Nutrition')
    }}">
View Medical Resources on Nutrition
</a>
</li>
@endif

@if (strcmp($category, 'Rehabilitation Medicine') == 0)
<li class="menu-text">View Medical Resources on
    Rehabilitation Medicine</li>
@else
<li>
<a href="{{ url('medical-resources/category/
    Rehabilitation%20Medicine') }}">
View Medical Resources on Rehabilitation Medicine
</a>
</li>
@endif

@else
<li>
<a href="{{ url('medical-resources/category/Neurology')
    }}">
View Medical Resources on Neurology
</a>
</li>
<li>
<a href="{{ url('medical-resources/category/Nutrition')
    }}">
View Medical Resources on Nutrition
</a>
</li>
<li>
<a href="{{ url('medical-resources/category/
    Rehabilitation%20Medicine') }}">
View Medical Resources on Rehabilitation Medicine
</a>
</li>
@endif
</ul>
</div>
<div class="row column margin-bottom-10">
@if (count($medical_resources) > 0)
@foreach ($medical_resources as $medical_resource)
<div class="callout margin-bottom-10 height-scroll @if
    (!$medical_resource->is_approved) secondary
    @endif">
@if ($medical_resource->user)
<div class="row column margin-bottom-10">
<div class="profile-card" id="rem-3">
@if ($medical_resource->user->file)
<img src="{{ url($medical_resource->user->file->path .
    '/' .
$medical_resource->user->file->filename) }}">
@else
<img src="{{ url('files/profile_pictures/anon-profile.
    jpg') }}">
@endif
<div class="padding-top-profile-picture">
<a href="{{ url('users/' .$medical_resource->user->
    username) }}">
{{ $medical_resource->user->last_name }}, {{
    $medical_resource->user->first_name }}
</a>
</div>
</div>
</div>
@endif
<div class="row column">
<h5>
<a href="{{ url('medical-resources/' .
    $medical_resource->id) }}">
{{ $medical_resource->title }}
</a>
</h5>
<p>
@if ($medical_resource->forumCategory)
@if ($medical_resource->forumCategory->name)
Category:
<a href="{{ url('medical-resources/category/' .
```

```
    $medical_resource->forumCategory->name) }}">
{{ $medical_resource->forumCategory->name }}
</a><br>
@endif
@endif
@if (! $medical_resource->is_approved)
This medical resource is not yet approved.<br>
@endif
{{ $medical_resource->description }}<br>
<span class="font-green">
Authors: {{ $medical_resource->authors }}
</span>
</p>
</div>
<div class="row medium-6 column">
<p>
Medical Resource File:<br>
@if ($medical_resource->file != null)
@if (in_array($medical_resource->file->extension,
    $image_extensions)
or in_array($medical_resource->file->extension,
    $video_extensions)
or $medical_resource->file->extension == 'pdf')
<a data-open="file-{{ $medical_resource->id }}">{{
    $medical_resource->file->filename }}</a>
@else
<a href="{{ url('medical-resources/' .
    $medical_resource->id . '/download') }}">
{{ $medical_resource->file->filename }}
</a>
@endif
@else <em>n/a</em>
@endif
</p>
@if ($medical_resource->file != null)
@if (in_array($medical_resource->file->extension,
    $image_extensions)
or in_array($medical_resource->file->extension,
    $video_extensions)
or ($medical_resource->file->extension == 'pdf'))
<div class="reveal large" id="file-{{ $medical_resource
    ->id }}" data-reveal
data-animation-in="fade-in" data-animation-out="fade-
    out">
<h4>{{ $medical_resource->title }}</h4>
<div class="row column center margin-top-20 margin-
    bottom-20">
@if (in_array($medical_resource->file->extension,
    $image_extensions))
<img src="{{ url($medical_resource->file->path . '/' .
$medical_resource->file->filename) }}">
@elseif (in_array($medical_resource->file->extension,
    $video_extensions))
<video height="500rem" controls>
@if ($medical_resource->file->extension == 'mp4')
<source src="{{ url($medical_resource->file->path . '/'
    .
$medical_resource->file->filename) }}" type="video/mp4"
    >
@elseif ($medical_resource->file->extension == 'webm')
<source src="{{ url($medical_resource->file->path . '/'
    .
$medical_resource->file->filename) }}" type="video/webm
    ">
@elseif ($medical_resource->file->extension == 'ogg')
<source src="{{ url($medical_resource->file->path . '/'
    .
$medical_resource->file->filename) }}" type="video/ogg"
    >
@endif
Your browser does not support the video tag.
</video>
@else
<object data="{{ url($medical_resource->file->path . '
    /' .
$medical_resource->file->filename) }}" type="
    application/pdf" width="100%"
height="500rem">
</object>
@endif
</div>
<div class="row column center">
<a href="{{ url('medical-resources/' .
    $medical_resource->id . '/download') }}"
class="button">
Download
</a>
<a class="button hollow" data-close>Close</a>
</div>
<button class="close-button" data-close aria-label="
    Close modal" type="button">
<span aria-hidden="true">&times;</span>
</button>
</div>
@endif
@endif
</div>
</div>
@endforeach
{!! $page_links !!}
@else
There are currently no medical resources.
@endif
@if ($is_staff)
<div class="row column center">
<a href="{{ url('medical-resources/create') }}" class="
    button">Add Medical Resource</a>
</div>
@endif
</div>
</div>
</div>
@endsection
@section('js')
<script type="text/javascript">
$('.reveal').foundation();
</script>
@endsection
```

Medical Resources Show View

```
@extends('app')
@section('css')@endsection
@section('content')
<div class="row">
<div class="small-10 small-centered medium-10 medium-
    centered large-10 large-centered column">
<div class="center">
<h4> @if ($medical_resource->title) {{
    $medical_resource->title }} @else Medical Resource
     @endif </h4>
</div>
<hr>
<div class="row column margin-bottom-20 menu-centered">
<ul class="vertical medium-horizontal menu">
<li><a href="{{ url('medical-resources') }}">View All
    Medical Resources</a></li>
<li>
<a href="{{ url('medical-resources/category/Neurology')
    }}">
View Medical Resources on Neurology
</a>
</li>
<li>
<a href="{{ url('medical-resources/category/Nutrition')
    }}">
View Medical Resources on Nutrition
</a>
</li>
<li>
<a href="{{ url('medical-resources/category/
    Rehabilitation%20Medicine') }}">
View Medical Resources on Rehabilitation Medicine
</a>
</li>
</ul>
</div>
<div class="row column margin-bottom-10">
<div class="callout margin-bottom-10">
@if ($medical_resource->user)
<div class="row column margin-bottom-10">
<div class="profile-card" id="rem-3">
@if ($medical_resource->user->file)
<img src="{{ url($medical_resource->user->file->path .
    '/' .
$medical_resource->user->file->filename) }}">
@else
<img src="{{ url('files/profile_pictures/anon-profile.
    jpg') }}">
@endif
<div class="padding-top-profile-picture">
<a href="{{ url('users/' . $medical_resource->user->
    username) }}">
{{ $medical_resource->user->last_name }},
{{ $medical_resource->user->first_name }}
</a>
</div>
</div>
</div>
@endif
<div class="row column">
<p>
@if ($medical_resource->forumCategory)
@if ($medical_resource->forumCategory->name)
<a href="{{ url('medical-resources/category/' .
    $medical_resource->forumCategory->name) }}">
{{ $medical_resource->forumCategory->name }}
</a><br>
@endif
@endif
@if (! $medical_resource->is_approved)
This medical resource is not yet approved.
@endif
</p>
<p>
Description:<br>
@if ($medical_resource->description) {{
    $medical_resource->description }}
@else <em>n/a</em>
@endif
</p>
<p>
Authors:
@if ($medical_resource->authors) {{ $medical_resource->
    authors }}
@else <em>n/a</em> @endif
</p>
<p>
Publisher:
@if ($medical_resource->publisher) {{ $medical_resource
    ->publisher }}
@else <em>n/a</em> @endif
</p>
<p>
Date Published:
@if ($medical_resource->date_published[1]) {{
    $medical_resource->date_published[1] }}
@else <em>n/a</em> @endif
</p>
</div>
<div class="row medium-6 column">
<p>
Medical Resource File:<br>
@if ($medical_resource->file != null)
@if (in_array($medical_resource->file->extension,
    $image_extensions)
or in_array($medical_resource->file->extension,
    $video_extensions)
or $medical_resource->file->extension == 'pdf')
<a data-open="file-{{ $medical_resource->id }}">
{{ $medical_resource->file->filename }}
</a>
@else
<a href="{{ url('medical-resources/' .
    $medical_resource->id . '/download') }}">
{{ $medical_resource->file->filename }}
</a>
@endif
@else <em>n/a</em>
@endif
</p>
```

```
@if ($medical_resource->file != null)
@if (in_array($medical_resource->file->extension,
    $image_extensions)
or in_array($medical_resource->file->extension,
    $video_extensions)
or ($medical_resource->file->extension == 'pdf'))
<div class="reveal large" id="file {{ $medical_resource
    ->id }}" data-reveal
data-animation-in="fade-in" data-animation-out="fade-
    out">
<h4>{{ $medical_resource->title }}</h4>
<div class="row column center margin-top-20 margin-
    bottom-20">
@if (in_array($medical_resource->file->extension,
    $image_extensions))
<img src="{{ url($medical_resource->file->path . '/' .
$medical_resource->file->filename) }}">
@elseif (in_array($medical_resource->file->extension,
    $video_extensions))
<video height="500rem" controls>
@if ($medical_resource->file->extension == 'mp4')
<source src="{{ url($medical_resource->file->path . '/'
    .
$medical_resource->file->filename) }}" type="video/mp4"
    >
@elseif ($medical_resource->file->extension == 'webm')
<source src="{{ url($medical_resource->file->path . '/'
    .
$medical_resource->file->filename) }}" type="video/webm
    ">
@elseif ($medical_resource->file->extension == 'ogg')
<source src="{{ url($medical_resource->file->path . '/'
    .
$medical_resource->file->filename) }}" type="video/ogg"
    >
@endif
Your browser does not support the video tag.
</video>
@else
<object data="{{ url($medical_resource->file->path . '
    /' .
$medical_resource->file->filename) }}" type="
    application/pdf" width="100%"
height="500rem">
</object>
@endif
</div>
<div class="row column center">
<a href="{{ url('medical-resources/' .
    $medical_resource->id . '/download') }}"
class="button">
Download
</a>
<a class="button hollow" data-close>Close</a>
</div>
<button class="close-button" data-close aria-label="
    Close modal" type="button">
<span aria-hidden="true">&times;</span>
</button>
</div>
@endif
@endif
</div>
</div>
</div>
<div class="row column margin-bottom-20 menu-centered">
<ul class="vertical medium-horizontal menu ul-margin">
@if ($is_supervisor and (! $medical_resource->
    is_approved))
<li>
{!! Form::open(['method' => 'POST',
'url' => 'medical-resources/' . $medical_resource->id .
    '/approve']) !!}
{!! Form::submit('Approve', ['class' => 'button ']) !!}
{!! Form::close() !!}
</li>
@endif
@if ($can_manage_medical_resource)
<li>
<a href="{{ url('medical-resources/' .
    $medical_resource->id . '/edit') }}"
class="button">
Edit
</a>
</li>
@endif
@if ($is_supervisor or $can_manage_medical_resource)
<li>
{!! Form::open(['method' => 'DELETE',
'url' => 'medical-resources/' . $medical_resource->id .
    '/delete']) !!}
{!! Form::submit('Delete', ['class' => 'button alert'])
    !!}
{!! Form::close() !!}
</li>
@endif
</ul>
</div>
</div>
</div>
@endsection
@section('js')
<script type="text/javascript">
$('.reveal').foundation();
</script>
@endsection
```

Partials Footer View

```
<div class="row">
<div class="medium-4 columns">
<h6 class="font-white">Developed by <strong>John Rafael
    Ferrer</strong> &#169; 2016</h6>
<h6 class="font-white">
<small>
University of the Philippines Manila<br>
College of Arts and Sciences<br>
Department of Physical Sciences and Mathematics<br>
Mathematics and Computing Sciences Unit<br>
</small>
```

```
</h6>
</div>
<div class="medium-4 columns">
<h6 class="font-white">
Contact E-mail Addresses:<br>
<small>
<a href="mailto:philippinestrokeportal@gmail.com?
    Subject=Phillippine%20Stroke%20Portal"
target="_top" class="font-white">
philippinestrokeportal@gmail.com
</a><br>
<a href="mailto:jaferrer4@up.edu.ph?Subject=Phillippine
    %20Stroke%20Portal"
target="_top" class="font-white">
jaferrer4@up.edu.ph
</a><br>
</small>
</h6>
</div>
<div class="medium-4 columns">
<h6 class="font-white">Related Links</h6>
<h6>
<ul class="no-bullet">
<li><a href="{{ url('http://www.strokesocietyphil.org')
    }}">Stroke Society of the Philippines</a></li>
<li><a href="{{ url('http://www.stroke.org/') }}">
    Stroke.org</a></li>
<li><a href="{{ url('http://www.up.edu.ph') }}">
    University of the Philippines Manila</a></li>
</ul>
</h6>
</div>
</div>
```

Partials Menu View

```
<ul class="vertical menu docs-nav">
@if (Auth::check())
@if (Auth::user()->patient)
<li>
<a href="{{ url('medical-conditions/'. Auth::user()->
    patient->id) }}">
Input Daily Medical Condition
</a>
</li>
<li>
<a href="{{ url('laboratory-results/'. Auth::user()->
    patient->id) }}">
Input Laboratory Result
</a>
</li>
<li>
<a href="{{ url('medication-regimens/'. Auth::user()->
    patient->id) }}">
View Daily Medication Regimens
</a>
</li>
<li>
<a href="{{ url('nutritionist-regimens/'. Auth::user()
    ->patient->id) }}">
View Daily Nutritionist Regimens
</a>
</li>
<li><a href="{{ url('physiatrist-regimens/'. Auth::
    user()->patient->id) }}">
View Daily Physiatrist Regimens
</a>
</li>
<li><a href="{{ url('schedule-visits') }}">Schedule
    Follow-Up Visit</a></li>
<li>
<a href="{{ url('patient-attributes/'. Auth::user()->
    patient->id) }}">
Update General Details
</a>
</li>
@if (Auth::user()->patient->hospital)
<li>
<a href="{{ url('hospitals/'. Auth::user()->patient->
    hospital->id) }}">
{{ Auth::user()->patient->hospital->name }}
</a>
</li>
@endif

@elseif (Auth::user()->medicalProfessional or Auth::
    user()->nutritionist or Auth::user()->physiatrist)
@if (Auth::user()->systemAdmin)
<li><a href="{{ url('hospital-admins') }}">Manage
    Hospital Administrators</a></li>
<li><a href="{{ url('supervisors') }}">Manage
    Supervisors</a></li>
@endif

@if (Auth::user()->localAdmin)
@if (Auth::user()->localAdmin->hospital)
<li><a href="{{ url('hospitals/'. Auth::user()->
    localAdmin->hospital->id) }}">Manage Hospital</a
    ></li>
<li>
<a href="{{ url('announcements-and-events/'. Auth::
    user()->localAdmin->hospital->name) }}">
Post Announcements and Events
</a>
</li>
@endif
@endif

@if (Auth::user()->supervisor)
@if (Auth::user()->supervisor->forumCategory)
<li>
<a href="{{ url('forum-threads/'. Auth::user()->
    supervisor->forumCategory->name) }}">
Moderate Forum Threads on {{ Auth::user()->supervisor->
    forumCategory->name }}
</a>
</li>
<li>
<a href="{{ url('medical-resources/category/'. Auth::
    user()->supervisor->forumCategory->name) }}">
```

```
Manage Medical Resources on {{ Auth::user()->supervisor
    ->forumCategory->name }}
</a>
</li>
@endif
@endif

@if (Auth::user()->medicalProfessional)
<li><a href="{{ url('patients') }}">Update Patient
    Medical Records</a></li>
<li><a href="{{ url('approve-visits') }}">Approve
    Patient Request Visits</a></li>

@elseif (Auth::user()->nutritionist)
<li><a href="{{ url('patients') }}">Update Nutrition
    Records</a></li>
<li><a href="{{ url('approve-visits') }}">Approve
    Patient Request Visits</a></li>

@else
<li><a href="{{ url('patients') }}">Update
    Rehabilitation Medicine Records</a></li>
<li><a href="{{ url('approve-visits') }}">Approve
    Patient Request Visits</a></li>

@endif
@endif
@endif
<li><a href="{{ url('medical-resources') }}">Medical
    Resources</a></li>
<li><a href="{{ url('announcements-and-events') }}">
    Announcements and Events</a></li>
<li><a href="{{ url('forum-threads/all') }}">Forum
    Threads</a></li>
</ul>
```

Partials Nav View

```
<div class="full-width navigation-area">
<div class="top-bar" id="top-menu">
<div class="top-bar-left">
<ul class="menu">
<li><a href="{{ url('/') }}">Home</a></li>
<li><a href="{{ url('hospitals') }}">Hospitals</a></li>
<li><a href="{{ url('about-us') }}">About Philippine
    Stroke Portal</a></li>
</ul>
</div>
<div class="top-bar-right">
<ul class="menu">
@if (Auth::guest())
<li><a href="{{ url('auth/login') }}">Login</a></li>
<li><a href="{{ url('auth/register') }}">Register</a>
    </li>
@else
<li>
<div class="profile-card" id="rem-2">
@if (Auth::user()->file)
<img src="{{ url(Auth::user()->file->path . '/' . Auth
    ::user()->file->filename) }}">
@else
<img src="{{ url('files/profile_pictures/anon-profile.
    jpg') }}">
@endif
<small>
You are logged in as
<a href="{{ url('profile') }}">
{{ Auth::user()->first_name . ' ' . Auth::user()->
    last_name }}
</a>
</small>
</div>
</li>
<li>
<a href="{{ url('notifications') }}">
@if (count(Auth::user()->notifications()->where('
    is_seen', false)->get()) > 0)
<span class="badge">{{ count(Auth::user()->
    notifications()->where('is_seen', false)->get())
    }}</span>
@endif
Notifications
</a>
</li>
<li><a href="{{ url('auth/logout') }}">Logout</a></li>
@endif
</ul>
</div>
</div>
</div>
```

Partials Status View

```
@if (session('alert'))
<div class="alert callout" data-closable>
<p>{{ session('alert') }}</p>
<button class="close-button" aria-label="Dismiss alert"
     type="button" data-close>
<span aria-hidden="true">&times;</span>
</button>
</div>
@endif
@if (session('success'))
<div class="success callout" data-closable>
<p>{{ session('success') }}</p>
<button class="close-button" aria-label="Dismiss alert"
     type="button" data-close>
<span aria-hidden="true">&times;</span>
</button>
</div>
@endif
```

Partials Title View

```
<div class="custom-title-bar">
<div class="row">
<div class="medium-12 columns end">
<a href="{{ url('http://www.strokesocietyphil.org/') }}
    " target="_blank">
```

```
<img src="{{ url('files/logos/logo-ssp.png') }}" class=
    "logo float-left">
</a>
<a href="{{ url('http://www.up.edu.ph/') }}" target="
    _blank">
<img src="{{ url('files/logos/logo-up.png') }}" class="
    logo float-left">
</a>
<h2 class="show-for-large">
<a href="{{ url('/') }}" class="font-black">Philippine
    Stroke Portal</a>
</h2>
<h3 class="show-for-medium-only">
<br>
<a href="{{ url('/') }}" class="font-black">Philippine
    Stroke Portal</a>
</h3>
</div>
</div>
<div class="row show-for-small-only">
<div class="small-12 columns end">
<h2><a href="{{ url('/') }}" class="font-black">
    Philippine Stroke Portal</a></h2>
</div>
</div>
</div>
```

Patient Attributes General Details Form View

```
<div class="callout">
<h5>General Data</h5>
<hr>
<div class="row">
<!-- Civil Status Form Input -->
<div class="large-4 columns">
<label for="civil_status">Civil Status*
{!! Form::select('civil_status', $civil_statuses, $attr
    ['civil_status'],
['class' => 'default', 'id' => 'civil_status']) !!}
@foreach ($errors->get('civil_status') as $key =>
    $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('civil_status')) - 1)<
    br>@endif
@endforeach
</label>
</div>
<!-- Landline Number Form Input -->
<div class="large-4 columns">
<label for="landline_number">Landline Number
{!! Form::text('landline_number', $attr['
    landline_number'],
['class' => 'default', 'id' => 'landline_number',
'aria-describedby' => 'landline_number_helper']) !!}
@foreach ($errors->get('landline_number') as $key =>
    $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('landline_number')) -
    1)<br>@endif
@endforeach
</label>
<p class="help-text" id="landline_number_helper">
(e.g., 1234567, 02-1234567)
</p>
</div>
<!-- Mobile Number Form Input -->
<div class="large-4 columns">
<label for="mobile_number">Mobile Number
{!! Form::text('mobile_number', $attr['mobile_number'],
['class' => 'default', 'id' => 'mobile_number',
'aria-describedby' => 'mobile_number_helper']) !!}
@foreach ($errors->get('mobile_number') as $key =>
    $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('mobile_number')) - 1)<
    br>@endif
@endforeach
</label>
<p class="help-text" id="mobile_number_helper">
(e.g., +639012345678)
</p>
</div>
</div>
<div class="row column">
<fieldset class="fieldset">
<legend>Mother's Maiden Name</legend>
<div class="row column">
<p class="help-text">
Input the name of the mother of the patient before
    marriage. The full middle name must
be entered. If there is no middle name, write  n/a
    .
</p>
</div>
<div class="row">
<div class="medium-4 columns">
<label for="mother_s_maiden_first_name">First Name*
{!! Form::text('mother_s_maiden_first_name',
$attr['mother_s_maiden_first_name'],
['class' => 'default', 'id' =>
    mother_s_maiden_first_name']) !!}
@foreach ($errors->get('mother_s_maiden_first_name') as
    $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('
    mother_s_maiden_first_name')) - 1)<br>@endif
@endforeach
</label>
</div>
<div class="medium-4 columns">
<label for="mother_s_maiden_middle_name">Middle Name*
{!! Form::text('mother_s_maiden_middle_name',
$attr['mother_s_maiden_middle_name'],
['class' => 'default', 'id' =>
    mother_s_maiden_middle_name']) !!}
@foreach ($errors->get('mother_s_maiden_middle_name')
    as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('
    mother_s_maiden_middle_name')) - 1)<br>@endif
```

```
@endforeach
</label>
</div>
<div class="medium-4 columns">
<label for="mother_s_maiden_last_name">Last Name*
{!! Form::text('mother_s_maiden_last_name',
$attr['mother_s_maiden_last_name'],
['class' => 'default', 'id' => '
    mother_s_maiden_last_name']) !!}
@foreach ($errors->get('mother_s_maiden_last_name') as
    $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('
    mother_s_maiden_last_name')) - 1)<br>@endif
@endforeach
</label>
</div>
</div>
</fieldset>
</div>
<div class="row column">
<fieldset class="fieldset">
<legend>Father's Name</legend>
<div class="row column">
<p class="help-text">
Input the name of the father of the patient.
</p>
</div>
<div class="row">
<div class="medium-4 columns">
<label for="father_s_first_name">First Name*
{!! Form::text('father_s_first_name', $attr['
    father_s_first_name'],
['class' => 'default', 'id' => 'father_s_first_name'])
    !!}
@foreach ($errors->get('father_s_first_name') as $key
    => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('father_s_first_name'))
     - 1)<br>@endif
@endforeach
</label>
</div>
<div class="medium-4 columns">
<label for="father_s_middle_name">Middle Name*
{!! Form::text('father_s_middle_name', $attr['
    father_s_middle_name'],
['class' => 'default', 'id' => 'father_s_middle_name'])
     !!}
@foreach ($errors->get('father_s_middle_name') as $key
    => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('father_s_middle_name')
    ) - 1)<br>@endif
@endforeach
</label>
</div>
<div class="medium-4 columns">
<label for="father_s_last_name">Last Name*
{!! Form::text('father_s_last_name', $attr['
    father_s_last_name'],
['class' => 'default', 'id' => 'father_s_last_name'])
    !!}
@foreach ($errors->get('father_s_last_name') as $key =>
     $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('father_s_last_name'))
    - 1)<br>@endif
@endforeach
</label>
</div>
</div>
</fieldset>
</div>
<div class="row column">
<fieldset class="fieldset">
<legend>Spouse's Name</legend>
<div class="row column">
<p class="help-text">
Input the name of the spouse of the patient, if
    applicable.
</p>
</div>
<div class="row">
<div class="medium-4 columns">
<label for="spouse_s_first_name">First Name
{!! Form::text('spouse_s_first_name', $attr['
    spouse_s_first_name'],
['class' => 'default', 'id' => 'spouse_s_first_name'])
    !!}
@foreach ($errors->get('spouse_s_first_name') as $key
    => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('spouse_s_first_name'))
    - 1)<br>@endif
@endforeach
</label>
</div>
<div class="medium-4 columns">
<label for="spouse_s_middle_name">Middle Name
{!! Form::text('spouse_s_middle_name', $attr['
    spouse_s_middle_name'],
['class' => 'default', 'id' => 'spouse_s_middle_name'])
    !!}
@foreach ($errors->get('spouse_s_middle_name') as $key
    => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('spouse_s_middle_name')
    ) - 1)<br>@endif
@endforeach
</label>
</div>
<div class="medium-4 columns">
<label for="spouse_s_last_name">Last Name
{!! Form::text('spouse_s_last_name', $attr['
    spouse_s_last_name'],
['class' => 'default', 'id' => 'spouse_s_last_name'])
    !!}
@foreach ($errors->get('spouse_s_last_name') as $key =>
    $error)
```

```
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('spouse_s_last_name')) - 1)<br>@endif
@endforeach
</label>
</div>
</div>
</fieldset>
</div>
<div class="row margin-bottom-10">
<!-- Place of Birth Form Input -->
<div class="large-6 column end">
<label for="place_of_birth">Place of Birth*
{!! Form::text('place_of_birth', $attr['place_of_birth'], ['class' => 'default',
'id' => 'place_of_birth', 'aria-describedby' => 'place_of_birth_helper']) !!}
@foreach ($errors->get('place_of_birth') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('place_of_birth')) - 1)<br>@endif
@endforeach
</label>
<p class="help-text" id="place_of_birth_helper">
Format: Municipality/City, Province
</p>
</div>
</div>
<div class="row margin-bottom-10">
<!-- Permanent Address Form Input -->
<div class="large-6 columns">
<label for="permanent_address">Permanent Address*
{!! Form::text('permanent_address', $attr['permanent_address'],
['class' => 'default', 'id' => 'permanent_address',
'aria-describedby' => 'permanent_address_helper']) !!}
@foreach ($errors->get('permanent_address') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('permanent_address')) - 1)<br>@endif
@endforeach
</label>
<p class="help-text" id="permanent_address_helper">
Format: House No. and Street, Barangay, Municipality/City and Province
</p>
</div>
<!-- Current Address Form Input -->
<div class="large-6 columns">
<label for="current_address">Current Address
{!! Form::text('current_address', $attr['current_address'], ['class' => 'default',
'id' => 'current_address', 'aria-describedby' => 'current_address_helper']) !!}
@foreach ($errors->get('current_address') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('current_address')) - 1)<br>@endif
@endforeach
</label>
<p class="help-text" id="current_address_helper">
Format: House No. and Street, Barangay, Municipality/City and Province
</p>
</div>
</div>
<div class="row margin-bottom-10">
<!-- Religion Form Input -->
<div class="large-4 columns">
<label for="religion">Religion*
{!! Form::text('religion', $attr['religion'],
['class' => 'default', 'id' => 'religion']) !!}
@foreach ($errors->get('religion') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('religion')) - 1)<br>@endif
@endforeach
</label>
</div>
<!-- Nationality Form Input -->
<div class="large-4 columns">
<label for="nationality">Nationality*
{!! Form::text('nationality', $attr['nationality'],
['class' => 'default', 'id' => 'nationality']) !!}
@foreach ($errors->get('nationality') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('nationality')) - 1)<br>@endif
@endforeach
</label>
</div>
<!-- Highest Educational Attainment Form Input -->
<div class="large-4 columns">
<label for="highest_educational_attainment">Highest Educational Attainment*
{!! Form::select('highest_educational_attainment', $highest_educational_attainments,
$attr['highest_educational_attainment'],
['class' => 'default', 'id' => 'highest_educational_attainment']) !!}
@foreach ($errors->get('highest_educational_attainment') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('highest_educational_attainment')) - 1)<br>@endif
@endforeach
</label>
</div>
</div>
<div class="row margin-bottom-10">
<!-- Occupation Form Input -->
<div class="large-4 columns">
<label for="occupation">Occupation*
{!! Form::text('occupation', $attr['occupation'],
['class' => 'default', 'id' => 'occupation']) !!}
@foreach ($errors->get('occupation') as $key => $error)
```

```
<small class="custom-form-error">{{ $error }}</small>
@if ($key === count($errors->get('occupation')) - 1)<br>
    @endif
@endforeach
</label>
</div>
<!-- Company Form Input -->
<div class="large-4 columns">
<label for="company">Company or Institution
{!! Form::text('company', $attr['company'], ['class' =>
    'default',
'id' => 'company', 'aria-describedby' => '
    company_helper']) !!}
@foreach ($errors->get('company') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key === count($errors->get('company')) - 1)<br>
    @endif
@endforeach
</label>
<p class="help-text" id="company_helper">
If applicable, input the name of the company or the
    institution where the patient is
working.
</p>
</div>
<!-- PhilHealth Number Form Input -->
<div class="large-4 columns">
<label for="philhealth_number">PhilHealth Number
{!! Form::text('philhealth_number', $attr['
    philhealth_number'],
['class' => 'default', 'id' => 'philhealth_number',
'aria-describedby' => 'philhealth_number_helper']) !!}
@foreach ($errors->get('philhealth_number') as $key =>
    $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key === count($errors->get('philhealth_number')) -
    1)<br>@endif
@endforeach
</label>
<p class="help-text" id="philhealth_number_helper">
If applicable, input the PhilHealth Number of the
    patient if he/she is a member or
dependent.
</p>
</div>
</div>
<div class="row margin-bottom-10">
<!-- Common Reference Number Form Input -->
<div class="large-4 column end">
<label for="common_reference_number">Common Reference
    Number
{!! Form::text('common_reference_number', $attr['
    common_reference_number'],
['class' => 'default', 'id' => 'common_reference_number
    ',
'aria-describedby' => 'common_reference_number_helper'
    ]) !!}
@foreach ($errors->get('common_reference_number') as
    $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key === count($errors->get('
    common_reference_number')) - 1)<br>@endif
@endforeach
</label>
<p class="help-text" id="common_reference_number_helper
    ">
Input the Unified Multi-Purpose ID Common Reference
    Number if the patient has any.
(UMID_CRN can be found in the upgraded, present
    government IDs, such as the SSS, GSIS
and Philippine Health Insurance Corp. UMID-CRN is the
    primary identifier of an
individual transacting business or availing services
    from any government agency.)
</p>
</div>
</div>
</div>
```

Patient Attributes Medical History Form View

```
<div class="callout">
<div class="row">
<div class="medium-8 columns">
<h5>Family History on Stroke</h5>
<p class="help-text">
If applicable, identify who among the family members of
     the patient has or had stroke
(e.g., mother, father, brother, uncle, grandparent).
    Accordingly, select the type of
stroke the family member has or had been diagnosed of.
</p>
</div>
<div class="medium-4 columns clearfix">
<span class="hide-for-small-only"><br><br></span>
<button type="button" class="button float-right" id="
    add_another_family_member">
Add Another Family Member
</button>
</div>
</div>
<hr>
<div class="row column" id="family_members_wrapper">
@if (count($attr['relationship_to_family_member']) > 0)
@foreach ($attr['relationship_to_family_member'] as
    $index => $relationship)
<div class="row margin-bottom-10">
<!-- Relationship to Family Member Form Input -->
<div class="medium-4 columns">
<label>Relationship to Family Member
{!! Form::text('relationship_to_family_member[]',
    $relationship,
['class' => 'default']) !!}
</label>
</div>
<!-- Type of Stroke of Family Member Form Input -->
<div class="medium-4 columns @if ($index == 0) end
    @endif">
<label>Type of Stroke of Family Member
{!! Form::select('type_of_stroke_of_family_member[]',
```

```
    $types_of_stroke,
$attr['type_of_stroke_of_family_member'][$index],
['class' => 'default']) !!}
</label>
</div>
@if ($index > 0)
<div class="medium-2 columns end">
<span class="hide-for-small-only"><br></span>
<button type="button" class="button hollow" id="
    remove_family_member">Remove</button>
</div>
@endif
</div>
@endforeach
@else
<div class="row margin-bottom-10">
<!-- Relationship to Family Member Form Input -->
<div class="medium-4 columns">
<label>Relationship to Family Member
{!! Form::text('relationship_to_family_member[]', null,
    ['class' => 'default']) !!}
</label>
</div>
<!-- Type of Stroke Form Input -->
<div class="medium-4 columns end">
<label>Type of Stroke
{!! Form::select('type_of_stroke_of_family_member[]',
    $types_of_stroke, null,
['class' => 'default']) !!}
</label>
</div>
</div>
@endif
</div>
</div>
<div class="callout">
<h5>Past Medical History</h5>
<p class="help-text">
If applicable, specify the type/s of disease/s and/or
    attack/s the family of the patient has/had
been diagnosed of or has a history of.
</p><hr>
<div class="row column margin-bottom-10">
<!-- Diseases or Attacks Form Input -->
<fieldset>
<legend>Disease or Attacks</legend>
@foreach ($attr['disease_or_attack'] as $key =>
    $attribute)
@if ($key == 'hypertension' or $key == '
    intracranial_artery_stenosis' or
$key == 'valvular_heart_disease')
<div class="row">
@endif
<div class="medium-4 columns">
<label for="disease_or_attack_{{ $key }}">
{{ $attribute[0] }}
</label>
{!! Form::checkbox('disease_or_attack[]', $key,
$attribute[1], ['id' => 'disease_or_attack_' . $key])
    !!}
</div>
@if ($key == 'dyslipidemia' or $key == '
    ischemic_heart_disease' or
$key == 'arrhythmia')
</div>
@endif
@endforeach
<div class="row medium-6 column">
<!-- Others Form Input -->
<label for="other_diseases_or_attacks">Others, specify
{!! Form::text('other_diseases_or_attacks', $attr['
    other_diseases_or_attacks'],
['class' => 'default', 'id' => '
    other_diseases_or_attacks']) !!}
</label>
@foreach ($errors->get('disease_or_attack') as $key =>
    $error)
<small class="custom-form-error no-margin-top">{{
    $error }}</small>
@if ($key == count($errors->get('disease_or_attack')) -
    1)<br>@endif
@endforeach
</div>
</fieldset>
</div>
</div>
<div class="callout">
<div class="row">
<div class="medium-8 columns">
<h5>Previous Surgeries</h5>
<p class="help-text">
If applicable, specify the type/s of surgery/ies the
    patient has/had undergone, as well
as the year/s of the surgery/ies.
</p>
</div>
<div class="medium-4 columns clearfix">
<span class="hide-for-small-only"><br><br></span>
<button type="button" class="button float-right" id="
    add_another_surgery">
Add Another Type of Surgery
</button>
</div>
</div>
<hr>
<div class="row column" id="surgeries_wrapper">
@if (count($attr['type_of_surgery']) > 0)
@foreach ($attr['type_of_surgery'] as $index => $type)
<div class="row margin-bottom-10">
<!-- Type of Surgery Form Input -->
<div class="medium-4 columns">
<label>Type of Surgery
{!! Form::text('type_of_surgery[]', $type, ['class' =>
    'default']) !!}
</label>
</div>
<!-- Year of Surgery Form Input -->
<div class="medium-4 columns @if ($index == 0) end
    @endif">
<label>Year of Surgery
```

```
{!! Form::number('year_of_surgery[]', $attr['
    year_of_surgery'][$index],
['class' => 'default']) !!}
</label>
</div>
@if ($index > 0)
<div class="medium-2 columns end">
<span class="hide-for-small-only"><br></span>
<button type="button" class="button hollow" id="
    remove_surgery">Remove</button>
</div>
@endif
</div>
@endforeach
@else
<div class="row margin-bottom-10">
<!-- Type of Surgery Form Input -->
<div class="medium-4 columns">
<label>Type of Surgery
{!! Form::text('type_of_surgery[]', null, ['class' => '
    default']) !!}
</label>
</div>
<!-- Year of Surgery Form Input -->
<div class="medium-4 columns @if (count($attr['
    type_of_surgery']) === 0) end @endif">
<label>Year of Surgery
{!! Form::number('year_of_surgery[]', null, ['class' =>
     'default']) !!}
</label>
</div>
</div>
@endif
</div>
</div>
```

Patient Attributes Script View

```
<script type="application/javascript">
$(document).ready(function() {
var family_member_wrapper = $("#family_members_wrapper"
    ); // Fields wrapper
var add_family_member_button = $("#
    add_another_family_member"); // Add button ID
var input_family_member =
'                        <div class="row margin-bottom
    -10">\n' +
'                            <!-- Relationship to Family
    Member Form Input -->\n' +
'                            <div class="medium-4
    columns">\n' +
'                                <label>Relationship to
    Family Member\n' +
'                                    <input class="
    default" name="relationship_to_family_member[]"
    type="text">\n' +
'                                </label>\n' +
'                            </div>\n' +
'                            <!-- Type of Stroke Form
    Input -->\n' +
'                            <div class="medium-4
    columns">\n' +
'                                <label>Type of Stroke\n
    ' +
'                                    <select class="
    default" name="type_of_stroke_of_family_member[]">
    ' +
'                                        <option value="
    ischemic_stroke">Ischemic Stroke</option>' +
'                                        <option value="
    hemorrhagic_stroke">Hemorrhagic Stroke</option>' +
'                                    </select>\n' +
'                                </label>\n' +
'                            </div>\n' +
'                            <div class="medium-2
    columns end">\n' +
'                                <span class="hide-for-
    small-only"><br></span>\n' +
'                                <button type="button"
    class="button hollow" id="remove_family_member">'
    +
'                                    Remove' +
'                                </button>\n' +
'                            </div>\n' +
'                        </div>\n';

var surgery_wrapper = $("#surgeries_wrapper"); //
    Fields wrapper
var add_surgery_button = $("#add_another_surgery"); //
    Add button ID
var input_surgery =
'                        <div class="row margin-bottom
    -10">\n' +
'                            <!-- Type Of Surgery Form
    Input -->\n' +
'                            <div class="medium-4
    columns">\n' +
'                                <label>Type Of Surgery\
    n' +
'                                    <input class="
    default" name="type_of_surgery[]" type="text">\n'
    +
'                                </label>\n' +
'                            </div>\n' +
'                            <!-- Year of Surgery Form
    Input -->\n' +
'                            <div class="medium-4
    columns">\n' +
'                                <label>Year of Surgery\
    n' +
'                                    <input class="
    default" name="year_of_surgery[]" type="number">\n
    ' +
'                                </label>\n' +
'                            </div>\n' +
'                            <div class="medium-2
    columns end">\n' +
'                                <span class="hide-for-
    small-only"><br></span>\n' +
'                                <button type="button"
```

```
    class="button␣hollow"␣id="remove_surgery">' +
'␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣Remove' +
'␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣</button>\n' +
'␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣</div>\n' +
'␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣</div>\n';

$(add_family_member_button).click(function(e){ // on
    add input button click
e.preventDefault();
$(family_member_wrapper).append(input_family_member);
    // add input fields
});

$(family_member_wrapper).on('click', "#
    remove_family_member", function(e){ // user click
    on remove text
e.preventDefault();
$(this).parent('div').parent('div').remove();
});

$(add_surgery_button).click(function(e){ // on add
    input button click
e.preventDefault();
$(surgery_wrapper).append(input_surgery); // add input
    fields
});

$(surgery_wrapper).on('click', "#remove_surgery",
    function(e){ // user click on remove text
e.preventDefault();
$(this).parent('div').parent('div').remove();
});
});
</script>
```

Patient Attributes Index View

```
@extends('app')
@section('css')@endsection
@section('content')
<div class="row">
<div class="small-10␣small-centered␣medium-10␣medium-
    centered␣large-10␣large-centered␣column">
<div class="center"><h4>General Patient Details: {{
    $patient_full_name }}</h4></div>
<hr>
@if ($is_staff)
<div class="row␣column␣margin-bottom-20␣menu-centered">
<ul class="vertical␣medium-horizontal␣menu">
<li><a href="{{␣url('patients/'␣.␣$patient->id)␣}}">
    Account Profile</a></li>
@if ($user->medicalProfessional)
<li class="menu-text">General Details</li>
<li><a href="{{␣url('medical-conditions/'␣.␣$patient->
    id)␣}}">Daily Medical Condition</a></li>
<li><a href="{{␣url('laboratory-results/'␣.␣$patient->
    id)␣}}">Laboratory Results</a></li>
<li><a href="{{␣url('medical-records/'␣.␣$patient->id)␣
    }}">Medical Records</a></li>
<li><a href="{{␣url('referral/'␣.␣$patient->id)␣}}">
    Referral</a></li>
<li><a href="{{␣url('medication-regimens/'␣.␣$patient->
    id)␣}}">Medication Regimens</a></li>
@endif
</ul>
</div>
@endif
<div class="row␣column␣margin-bottom-10">
<div class="callout">
<h5>Account Details</h5>
<hr>
<div class="row">
<div class="large-4␣columns␣margin-bottom-10">
<div class="font-0875rem">Hospital Patient ID Number:</
    div>{{ $patient->hospital_id_number }}
</div>
<div class="large-4␣columns␣margin-bottom-10">
<div class="font-0875rem">Username:</div>{{ $patient->
    user->username }}
</div>
<div class="large-4␣columns␣margin-bottom-10">
<div class="font-0875rem">E-mail Address:</div>{{
    $patient->user->email }}
</div>
</div>
<div class="row␣margin-bottom-10">
<div class="large-4␣columns␣margin-bottom-10">
<div class="font-0875rem">Name:</div>{{
    $patient_full_name }}
</div>
<div class="large-4␣columns␣margin-bottom-10">
<div class="font-0875rem">Sex:</div>{{ ucfirst($patient
    ->user->sex) }}
</div>
<div class="large-4␣columns␣margin-bottom-10">
<div class="font-0875rem">Birthday:</div>{{ $patient->
    user->birthday[1] }}
</div>
</div>
</div>
{!! Form::open(['method' => 'POST', 'url' => 'patient-
    attributes/' . $patient->id]) !!}
@include('patient_attributes._general_details_form')
@include('patient_attributes._medical_history_form')
<p class="help-text">Fields marked with * are mandatory
    .</p>
<!-- Submit Button -->
<div class="row␣column␣center">
{!! Form::submit('Update␣General␣Patient␣Details', ['
    class' => 'button']) !!}
<a class="button␣hollow" href="{{␣url('patients',␣
    $patient->id)␣}}">Cancel</a>
</div>
{!! Form::close() !!}
</div>
</div>
</div>
@endsection
@section('js')
@include('patient_attributes._script')
```

```
@endsection
```

Patients Form View

```
<!-- Hospital ID Number Form Input -->
<div class="row">
<div class="small-4 columns">
{!! Form::label('hospital_id_number', 'Hospital ID
    Number*',
['class' => 'middle', 'for' => 'hospital_id_number'])
    !!}
</div>
<div class="small-8 columns end">
{!! Form::text('hospital_id_number', $patient->
    hospital_id_number, ['class' => 'default',
'id' => 'identifier', 'aria-describedby' => '
    hospital_id_number_helper']) !!}
@foreach ($errors->get('hospital_id_number') as $key =>
     $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('hospital_id_number'))
    - 1)<br>@endif
@endforeach
<p class="help-text" id="hospital_id_number_helper">
Input the hospital-based, issued ID/number to identify
    the patient.
</p>
</div>
</div>
<!-- First Name Form Input -->
<div class="row">
<div class="small-4 columns">
{!! Form::label('first_name', 'First Name*', ['class'
    => 'middle', 'for' => 'first_name']) !!}
</div>
<div class="small-8 columns end">
{!! Form::text('first_name', $patient->user->first_name
    ,
['class' => 'default', 'id' => 'first_name']) !!}
@foreach ($errors->get('first_name') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('first_name')) - 1)<br>
    @endif
@endforeach
</div>
</div>
<!-- Middle Name Form Input -->
<div class="row">
<div class="small-4 columns">
{!! Form::label('middle_name', 'Middle Name',
['class' => 'middle', 'for' => 'middle_name']) !!}
</div>
<div class="small-8 columns end">
{!! Form::text('middle_name', $patient->user->
    middle_name,
['class' => 'default', 'id' => 'middle_name']) !!}
@foreach ($errors->get('middle_name') as $key => $error
    )
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('middle_name')) - 1)<br
    >@endif
@endforeach
</div>
</div>
<!-- Last Name Form Input -->
<div class="row">
<div class="small-4 columns">
{!! Form::label('last_name', 'Last Name*', ['class' =>
    'middle', 'for' => 'last_name']) !!}
</div>
<div class="small-8 columns end">
{!! Form::text('last_name', $patient->user->last_name,
['class' => 'default', 'id' => 'last_name']) !!}
@foreach ($errors->get('last_name') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('last_name')) - 1)<br>
    @endif
@endforeach
</div>
</div>
<!-- Sex Form Input -->
<div class="row margin-bottom-10">
<div class="small-4 columns">
{!! Form::label('sex', 'Sex*') !!}
</div>
<fieldset class="small-8 columns end">
@if ($patient->user->sex == "male")
{!! Form::radio('sex', 'male', true, ['id' => 'sex'])
    !!}<label for="sex">Male</label>
@else
{!! Form::radio('sex', 'male', false, ['id' => 'sex'])
    !!}<label for="sex">Male</label>
@endif
@if ($patient->user->sex == "female")
{!! Form::radio('sex', 'female', true, ['id' => 'sex'])
     !!}<label for="sex">Female</label>
@else
{!! Form::radio('sex', 'female', false, ['id' => 'sex'
    ]) !!}<label for="sex">Female</label>
@endif
@foreach ($errors->get('sex') as $key => $error)
<small class="custom-form-error no-margin-top">{{
    $error }}</small>
@if ($key == count($errors->get('sex')) - 1)<br>@endif
@endforeach
</fieldset>
</div>
<!-- Birthday Form Input -->
<div class="row">
<div class="small-4 columns">
{!! Form::label('birthday', 'Birthday*', ['class' => '
    middle', 'for' => 'birthday']) !!}
</div>
<div class="small-8 columns end">
{!! Form::input('date', 'birthday', $patient->user->
    birthday[0]->format('Y-m-d'),
['class' => 'default', 'id' => 'birthday']) !!}
@foreach ($errors->get('birthday') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('birthday')) - 1)<br>
```

```
    @endif
@endforeach
</div>
</div>
<!-- Caregiver's First Name Form Input -->
<div class="row">
<div class="small-4 columns">
{!! Form::label('caregiver_first_name', 'Caregiver\'s
    First Name',
['class' => 'middle', 'for' => 'caregiver_first_name'])
     !!}
</div>
<div class="small-8 columns end">
{!! Form::text('caregiver_first_name', $patient->
    caregiver_first_name, ['class' => 'default',
'id' => 'caregiver_first_name', 'aria-describedby' => '
    caregiver_first_name_helper']) !!}
@foreach ($errors->get('caregiver_first_name') as $key
    => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('caregiver_first_name')
    ) - 1)<br>@endif
@endforeach
</div>
</div>
<!-- Caregiver's Middle Name Form Input -->
<div class="row">
<div class="small-4 columns">
{!! Form::label('caregiver_middle_name', 'Caregiver\'s
    Middle Name',
['class' => 'middle', 'for' => 'caregiver_middle_name'
    ]) !!}
</div>
<div class="small-8 columns end">
{!! Form::text('caregiver_middle_name', $patient->
    caregiver_middle_name, ['class' => 'default',
'id' => 'caregiver_middle_name', 'aria-describedby' =>
    'caregiver_middle_name_helper']) !!}
@foreach ($errors->get('caregiver_middle_name') as $key
     => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('caregiver_middle_name'
    )) - 1)<br>@endif
@endforeach
</div>
</div>
<!-- Caregiver's Last Name Form Input -->
<div class="row">
<div class="small-4 columns">
{!! Form::label('caregiver_last_name', 'Caregiver\'s
    Last Name',
['class' => 'middle', 'for' => 'caregiver_last_name'])
    !!}
</div>
<div class="small-8 columns end">
{!! Form::text('caregiver_last_name', $patient->
    caregiver_last_name, ['class' => 'default',
'id' => 'caregiver_last_name','aria-describedby' => '
    caregiver_last_name_helper']) !!}
@foreach ($errors->get('caregiver_last_name') as $key
    => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('caregiver_last_name'))
     - 1)<br>@endif
@endforeach
</div>
</div>
<p class="float-left help-text">Fields marked with *
    are mandatory.</p>
<!-- Submit Button -->
<div class="row">
<div class="small-12 center column end">
{!! Form::submit($submitButtonText, ['class' => 'button
    ']) !!}
<a class="hollow button" href="{{ url('patients/'.
    $patient->id) }}">Cancel</a>
</div>
</div>
```

Patients Create View

```
@extends('app')
@section('css')@endsection
@section('content')
<div class="row">
<div class="small-10 small-centered medium-8 medium-
    centered large-6 large-centered column">
<div class="center"><h4>Register a New Patient</h4></
    div>
<hr>
<div class="row margin-bottom-10">
<div class="small-4 columns"><p class="font-12-px">
    Hospital/Clinic</p></div>
<div class="small-8 columns">{{ $hospital->name }}</div
    >
</div>
<div class="row margin-bottom-10">
<div class="small-4 columns"><p class="font-12-px">
    Medical Professional</p></div>
<div class="small-8 columns">{{ $professional }}</div>
</div>
{!! Form::model($patient = new \App\Models\Patient, ['
    url' => 'patients/create/' . $profession_id]) !!}
<!-- Hospital ID Number Form Input -->
<div class="row">
<div class="small-4 columns">
{!! Form::label('hospital_id_number', 'Hospital ID
    Number*',
['class' => 'middle', 'for' => 'hospital_id_number'])
    !!}
</div>
<div class="small-8 columns end">
{!! Form::text('hospital_id_number', null, ['class' =>
    'default', 'id' => 'identifier',
'aria-describedby' => 'hospital_id_number_helper']) !!}
@foreach ($errors->get('hospital_id_number') as $key =>
     $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('hospital_id_number'))
    - 1)<br>@endif
```

```
@endforeach
<p class="help-text" id="hospital_id_number_helper">
Input the hospital-based, issued ID/number to identify
    the patient.
</p>
</div>
</div>
@include ('auth._register_form')
<!-- Caregiver's First Name Form Input -->
<div class="row">
<div class="small-4 columns">
{!! Form::label('caregiver_first_name', 'Caregiver\'s
    First Name',
['class' => 'middle', 'for' => 'caregiver_first_name'])
     !!}
</div>
<div class="small-8 columns end">
{!! Form::text('caregiver_first_name', null, ['class'
    => 'default',
'id' => 'caregiver_first_name', 'aria-describedby' => '
    caregiver_first_name_helper']) !!}
@foreach ($errors->get('caregiver_first_name') as $key
    => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('caregiver_first_name')
    ) - 1)<br>@endif
@endforeach
</div>
</div>
<!-- Caregiver's Middle Name Form Input -->
<div class="row">
<div class="small-4 columns">
{!! Form::label('caregiver_middle_name', 'Caregiver\'s
    Middle Name',
['class' => 'middle', 'for' => 'caregiver_middle_name'
    ]) !!}
</div>
<div class="small-8 columns end">
{!! Form::text('caregiver_middle_name', null, ['class'
    => 'default',
'id' => 'caregiver_middle_name', 'aria-describedby' =>
    'caregiver_middle_name_helper']) !!}
@foreach ($errors->get('caregiver_middle_name') as $key
     => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('caregiver_middle_name'
    )) - 1)<br>@endif
@endforeach
</div>
</div>
<!-- Caregiver's Last Name Form Input -->
<div class="row">
<div class="small-4 columns">
{!! Form::label('caregiver_last_name', 'Caregiver\'s
    Last Name',
['class' => 'middle', 'for' => 'caregiver_last_name'])
    !!}
</div>
<div class="small-8 columns end">
{!! Form::text('caregiver_last_name', null, ['class' =>
     'default',
'id' => 'caregiver_last_name','aria-describedby' => '
    caregiver_last_name_helper']) !!}
@foreach ($errors->get('caregiver_last_name') as $key
    => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('caregiver_last_name'))
     - 1)<br>@endif
@endforeach
</div>
</div>
<p class="float-left help-text">Fields marked with *
    are mandatory.</p>
<!-- Submit Button -->
<div class="row">
<div class="small-12 center column end">
{!! Form::submit('Register Patient', ['class' => '
    button']) !!}
<a class="hollow button" href="{{ url('patients') }}">
    Cancel</a>
</div>
</div>
{!! Form::close() !!}
</div>
</div>
@endsection
@section('js')@endsection
```

Patients Edit View

```
@extends('app')
@section('css')@endsection
@section('content')
<div class="row">
<div class="small-10 small-centered medium-8 medium-
    centered large-6 large-centered columns custom-
    panel">
<h4 class="text-center">Update Patient Account Profile
    </h4>
<hr>
{!! Form::open(['method' => 'PATCH', 'url' => ['
    patients', $patient->id]]) !!}
@include ('patients._form', ['submitButtonText' => '
    Update Profile'])
{!! Form::close() !!}
</div>
</div>
@endsection
@section('js')@endsection
```

Patients Index View

```
@extends('app')
@section('css'){!! Html::style('css/responsive-tables.
    css') !!}@endsection
@section('content')
<div class="row">
<div class="small-10 small-centered medium-10 medium-
    centered large-10 large-centered column">
<div class="center"><h4>Patients</h4></div>
```

```
<hr>
@if (count($mp_patients) > 0 or count($n_patients) > 0
    or count($p_patients) > 0)
@foreach ($mp_patients as $mp_patient)
<div class="row column margin-bottom-10">
<div class="row">
<div class="medium-8 columns">
<h5>{{ $mp_patient['hospital_name'] }} - Medical
    Professional</h5>
</div>
<div class="medium-4 columns clearfix">
<a class="button float-right show-for-medium"
href="{{ url('patients/create/' . $mp_patient['id']) }}
    ">
Add a new Patient Record
</a>
<a class="button show-for-small-only"
href="{{ url('patients/create/' . $mp_patient['id']) }}
    ">
Add a new Patient Record
</a>
</div>
</div>
@if (count($mp_patient['patients']) > 0)
<table class="responsive" width="100%">
<tr>
<th>Hospital Patient ID Number</th>
<th>Username</th>
<th>Name</th>
</tr>
@foreach ($mp_patient['patients'] as $patient)
<tr>
<td>
<a href="{{ url('patients/' . $patient['id']) }}">
{{ $patient['hospital_id_number'] }}
</a>
</td>
<td>{{ $patient['username'] }}</td>
<td>{{ $patient['full_name'] }}</td>
</tr>
@endforeach
</table>
@else
<p>You have no patients in this hospital.</p>
@endif
</div>
@endforeach
@foreach ($n_patients as $n_patient)
<div class="row column margin-bottom-10">
<div class="row">
<div class="medium-8 columns end">
<h5>{{ $n_patient['hospital_name'] }} - Nutritionist</
    h5>
</div>
</div>
@if (count($n_patient['patients']) > 0)
<table class="responsive" width="100%">
<tr>
<th>Hospital Patient ID Number</th>
<th>Username</th>
<th>Name</th>
</tr>
@foreach ($n_patient['patients'] as $patient)
<tr>
<td>
<a href="{{ url('patients/' . $patient['id']) }}">
{{ $patient['hospital_id_number'] }}
</a>
</td>
<td>{{ $patient['username'] }}</td>
<td>{{ $patient['full_name'] }}</td>
</tr>
@endforeach
</table>
@else
<p>You have no patients in this hospital.</p>
@endif
</div>
@endforeach
@foreach ($p_patients as $p_patient)
<div class="row column margin-bottom-10">
<div class="row">
<div class="medium-8 columns end">
<h5>{{ $p_patient['hospital_name'] }} - Rehabilitation
    Medicine Doctor</h5>
</div>
</div>
@if (count($p_patient['patients']) > 0)
<table class="responsive" width="100%">
<tr>
<th>Hospital Patient ID Number</th>
<th>Username</th>
<th>Name</th>
</tr>
@foreach ($p_patient['patients'] as $patient)
<tr>
<td>
<a href="{{ url('patients/' . $patient['id']) }}">
{{ $patient['hospital_id_number'] }}
</a>
</td>
<td>{{ $patient['username'] }}</td>
<td>{{ $patient['full_name'] }}</td>
</tr>
@endforeach
</table>
@else
<p>You have no patients in this hospital.</p>
@endif
</div>
@endforeach
@else
<p>
You are either not registered to any hospitals or
    locked in certain hospital records.
Find your hospital or clinic <a href="{{ url('hospitals
    ') }}">here</a> and register your account.
</p>
@endif
</div>
```

```
</div>
@endsection
@section('js'){!! Html::script('js/responsive-tables.js
    ') !!}@endsection
```

Patients Show View

```
@extends('app')
@section('css')@endsection
@section('content')
<div class="row">
<div class="small-10 small-centered meduim-10 medium-
    centered large-10 large-centered column">
<div class="center"><h4>Patient Account: {{
    $patient_full_name }}</h4></div>
<hr>
@if ($is_staff)
<div class="row column margin-bottom-20 menu-centered">
<ul class="vertical medium-horizontal menu">
<li class="menu-text">Account Profile</li>
@if ($user->medicalProfessional)
<li><a href="{{ url('patient-attributes/' . $patient->
    id) }}">General Details</a></li>
<li><a href="{{ url('medical-conditions/' . $patient->
    id) }}">Daily Medical Condition</a></li>
<li><a href="{{ url('laboratory-results/' . $patient->
    id) }}">Laboratory Results</a></li>
<li><a href="{{ url('medical-records/' . $patient->id) }
    }">Medical Records</a></li>
<li><a href="{{ url('referral/' . $patient->id) }}">
    Referral</a></li>
<li><a href="{{ url('medication-regimens/' . $patient->
    id) }}">Medication Regimens</a></li>
@elseif ($user->nutritionist)
<li><a href="{{ url('nutritionist-records/' . $patient
    ->id) }}">Nutrition Records</a></li>
<li><a href="{{ url('nutritionist-regimens/' . $patient
    ->id) }}">Nutrition Regimens</a></li>
@elseif ($user->physiatrist)
<li><a href="{{ url('physiatrist-records/' . $patient->
    id) }}">Rehabilitation Medicine Records</a></li>
<li><a href="{{ url('physiatrist-regimens/' . $patient
    ->id) }}">Rehabilitation Medicine Regimens</a></li
    >
@endif
</ul>
</div>
@endif
<div class="row">
<div class="medium-6 columns center margin-bottom-10">
@if ($patient->user->file)
<img src="{{ url($patient->user->file->path . '/' .
    $patient->user->file->filename) }}" width="300rem"
    >
@else
<img src="{{ url('files/profile_pictures/anon-profile.
    jpg') }}" width="300rem">
@endif
</div>
<div class="medium-6 columns margin-bottom-10">
<div class="row margin-bottom-10">
<div class="medium-5 columns">Hospital ID Number:</div>
<div class="medium-7 columns end">
@if ($patient->hospital_id_number) {{ $patient->
    hospital_id_number }}
@else <em>n/a</em> @endif
</div>
</div>
<div class="row margin-bottom-10">
<div class="medium-5 columns">Username:</div>
<div class="medium-7 columns end">
@if ($patient->user->username) {{ $patient->user->
    username }}
@else <em>n/a</em> @endif
</div>
</div>
<div class="row margin-bottom-10">
<div class="medium-5 columns">E-mail Address:</div>
<div class="medium-7 columns end">
@if ($patient->user->email) {{ $patient->user->email }}
@else <em>n/a</em> @endif
</div>
</div>
<div class="row margin-bottom-10">
<div class="medium-5 columns">First Name:</div>
<div class="medium-7 columns end">
@if ($patient->user->first_name) {{ $patient->user->
    first_name }}
@else <em>n/a</em> @endif
</div>
</div>
<div class="row margin-bottom-10">
<div class="medium-5 columns">Middle Name:</div>
<div class="medium-7 columns end">
@if ($patient->user->middle_name) {{ $patient->user->
    middle_name }}
@else <em>n/a</em> @endif
</div>
</div>
<div class="row margin-bottom-10">
<div class="medium-5 columns">Last Name:</div>
<div class="medium-7 columns end">
@if ($patient->user->last_name) {{ $patient->user->
    last_name }}
@else <em>n/a</em> @endif
</div>
</div>
<div class="row margin-bottom-10">
<div class="medium-5 columns">Sex:</div>
<div class="medium-7 columns end">
@if ($patient->user->sex) {{ ucfirst($patient->user->
    sex) }}
@else <em>n/a</em> @endif
</div>
</div>
<div class="row margin-bottom-10">
<div class="medium-5 columns">Birthday:</div>
<div class="medium-7 columns end">
@if ($patient->user->birthday[1]) {{ $patient->user->
    birthday[1] }}
```

```
@else <em>n/a</em> @endif
</div>
</div>
<div class="row margin-bottom-10">
<div class="medium-5 columns">Caregiver First Name:</
    div>
<div class="medium-7 columns end">
@if ($patient->caregiver_first_name) {{ $patient->
    caregiver_first_name }}
@else <em>n/a</em> @endif
</div>
</div>
<div class="row margin-bottom-10">
<div class="medium-5 columns">Caregiver Middle Name:</
    div>
<div class="medium-7 columns end">
@if ($patient->caregiver_middle_name) {{ $patient->
    caregiver_middle_name }}
@else <em>n/a</em> @endif
</div>
</div>
<div class="row margin-bottom-10">
<div class="medium-5 columns">Caregiver Last Name:</div
    >
<div class="medium-7 columns end">
@if ($patient->caregiver_last_name) {{ $patient->
    caregiver_last_name }}
@else <em>n/a</em> @endif
</div>
</div>
<div class="row margin-bottom-10">
<div class="medium-5 columns">Medical Professional:</
    div>
<div class="medium-7 columns end">
@if ($patient->medicalProfessional)
{{ $patient->medicalProfessional->user->last_name }},
{{ $patient->medicalProfessional->user->first_name }}
{{ $patient->medicalProfessional->user->middle_name }}
@else <em>n/a</em> @endif
</div>
</div>
<div class="row margin-bottom-10">
<div class="medium-5 columns">Nutritionist:</div>
<div class="medium-7 columns end">
@if ($patient->nutritionist)
{{ $patient->nutritionist->user->last_name }},
{{ $patient->nutritionist->user->first_name }}
{{ $patient->nutritionist->user->middle_name }}
@else <em>n/a</em> @endif
</div>
</div>
<div class="row margin-bottom-10">
<div class="medium-5 columns">Physiatrist:</div>
<div class="medium-7 columns end">
@if ($patient->physiatrist)
{{ $patient->physiatrist->user->last_name }},
{{ $patient->physiatrist->user->first_name }}
{{ $patient->physiatrist->user->middle_name }}
@else <em>n/a</em> @endif
</div>
</div>
</div>
</div>
@if ($user->medicalProfessional)
<div class="row column center">
<a href="{{ url('patients/' . $patient->id . '/edit') }}" class="button">Edit Patient Account Profile</a
    >
<a href="{{ url('patients') }}" class="button hollow">
    Cancel</a>
</div>
@endif
</div>
</div>
@endsection
@section('js')@endsection
```

Refer Index View

```
@extends('app')
@section('css'){!! Html::style('css/responsive-tables.
    css') !!}@endsection
@section('content')
<div class="row">
<div class="small-10 small-centered medium-10 medium-
    centered large-10 large-centered column">
<div class="center"><h4>Refer Patient: {{ $patient->
    user->last_name }},
{{ $patient->user->first_name }} {{ $patient->user->
    middle_name }}</h4></div>
<hr>
@if ($is_staff)
<div class="row column margin-bottom-20 menu-centered">
<ul class="vertical medium-horizontal menu">
<li><a href="{{ url('patients/' . $patient->id) }}">
    Account Profile</a></li>
@if ($user->medicalProfessional)
<li><a href="{{ url('patient-attributes/' . $patient->
    id) }}">General Details</a></li>
<li><a href="{{ url('medical-conditions/' . $patient->
    id) }}">Daily Medical Condition</a></li>
<li><a href="{{ url('laboratory-results/' . $patient->
    id) }}">Laboratory Results</a></li>
<li><a href="{{ url('medical-records/' . $patient->id)
    }}">Medical Records</a></li>
<li class="menu-text">Referral</li>
<li><a href="{{ url('medication-regimens/' . $patient->
    id) }}">Medication Regimens</a></li>
@endif
</ul>
</div>
@endif
<div class="row column margin-bottom-10">
<div class="callout">
{!! Form::open(['method' => 'POST', 'url' => 'referral/
    ' . $patient->id . '/nutritionist']) !!}
<div class="row margin-bottom-20">
<!-- Refer patient to a nutritionist Form Input -->
<div class="medium-8 columns">
<label for="nutritionist">Refer patient to a
```

```
    nutritionist
{!! Form::select('nutritionist', $nutritionists, null,
    ['class' => 'default',
'id' => 'nutritionist']) !!}
</label>
<p class="font-0875rem">
Current Nutritionist:
@if ($patient->nutritionist)
@if ($patient->nutritionist->user)
{{ $patient->nutritionist->user->last_name }},
{{ $patient->nutritionist->user->first_name }}
@else <em>n/a</em>@endif
@else <em>n/a</em>@endif
</p>
</div>
<!-- Submit Button -->
<div class="medium-4 columns">
<div class="center">
<br>{!! Form::submit('Submit Referral', ['class' => '
    button']) !!}
</div>
</div>
</div><br>
{!! Form::close() !!}
{!! Form::open(['method' => 'POST', 'url' => 'referral/
    ' . $patient->id . '/physiatrist']) !!}
<div class="row margin-bottom-20">
<!-- Refer patient to a rehabilitation medicine doctorr
     Form Input -->
<div class="medium-8 columns">
<label for="physiatrist">Refer patient to a
    rehabilitation medicine doctor
{!! Form::select('physiatrist', $physiatrists, null, ['
    class' => 'default',
'id' => 'physiatrist']) !!}
</label>
<p class="font-0875rem">
Current Rehabilitation Doctor:
@if ($patient->physiatrist)
@if ($patient->physiatrist->user)
{{ $patient->physiatrist->user->last_name }},
{{ $patient->physiatrist->user->first_name }}
@else <em>n/a</em>@endif
@else <em>n/a</em>@endif
</p>
</div>
<!-- Submit Button -->
<div class="medium-4 columns">
<div class="center">
<br>{!! Form::submit('Submit Referral', ['class' => '
    button']) !!}
</div>
</div>
</div><br>
{!! Form::close() !!}
{!! Form::open(['method' => 'POST', 'url' => 'referral/
    ' . $patient->id .
'/medical-professional']) !!}
<div class="row margin-bottom-20">
<!-- Refer patient to another doctor Form Input -->
<div class="medium-8 columns">
<label for="medical_professional">Refer patient to
    another doctor
{!! Form::select('medical_professional',
    $medical_professionals, null,
['class' => 'default', 'id' => 'medical_professional'])
     !!}
</label>
<p class="font-0875rem">
Current Medical Professional:
@if ($patient->medicalProfessional)
@if ($patient->medicalProfessional->user)
{{ $patient->medicalProfessional->user->last_name }},
{{ $patient->medicalProfessional->user->first_name }}
@else <em>n/a</em>@endif
@else <em>n/a</em>@endif
</p>
</div>
<!-- Submit Button -->
<div class="medium-4 columns">
<div class="center">
<br>{!! Form::submit('Submit Referral', ['class' => '
    button']) !!}
</div>
</div>
</div>
{!! Form::close() !!}
</div>
</div>
</div>
</div>
@endsection
@section('js'){!! Html::script('js/responsive-tables.js
    ') !!}@endsection
```

Medication Regimens Form View

```
<div class="row column">
<p class="help-text">
Answer the following fields about the daily medication
    regimen prescribed to the patient.
</p>
<hr>
</div>
<div class="row margin-bottom-10">
<!-- Medication Form Input -->
<div class="medium-6 columns">
<label for="medication">Medication*
{!! Form::text('medication', $medication_regimen->
    medication, ['class' => 'default',
'id' => 'medication', 'aria-describedby' => '
    medication_helper']) !!}
@foreach ($errors->get('medication') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('medication')) - 1)<br>
    @endif
@endforeach
</label>
<p class="help-text" id="medication_helper">
Input the generic name of the medicine or drug.
```

```
</p>
</div>
<!-- Dose Form Input -->
<div class="medium-3 columns">
<label for="dose">Dose*
{!! Form::number('dose', $medication_regimen->dose, ['
    class' => 'default',
'id' => 'dose', 'aria-describedby' => 'dose_helper', '
    step' => 'any']) !!}
@foreach ($errors->get('dose') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('dose')) - 1)<br>@endif
@endforeach
</label>
<p class="help-text" id="dose_helper">
Input the supposed daily dose of the medication in
    milligrams.
</p>
</div>
<!-- Frequency Form Input -->
<div class="medium-3 columns">
<label for="frequency">Frequency*
{!! Form::number('frequency', $medication_regimen->
    frequency, ['class' => 'default',
'id' => 'frequency', 'aria-describedby' => '
    frequency_helper']) !!}
@foreach ($errors->get('frequency') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('frequency')) - 1)<br>
    @endif
@endforeach
</label>
<p class="help-text" id="frequency_helper">
Input the number of times the medication is taken per
    day.
</p>
</div>
</div>
<div class="row margin-bottom-10">
<!-- Time of Medication Form Input -->
<div class="medium-8 columns">
<label for="time_of_medication">Time of Medication
{!! Form::text('time_of_medication',
    $medication_regimen->time_of_medication,
['class' => 'default', 'id' => 'time_of_medication',
'aria-describedby' => 'time_of_medication_helper']) !!}
@foreach ($errors->get('time_of_medication') as $key =>
     $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('time_of_medication'))
    - 1)<br>@endif
@endforeach
</label>
<p class="help-text" id="time_of_medication_helper">
Input the suggested time of medication depending on
    frequencies (e.g., 7:00am, 12:00pm,
and 5:00pm).
</p>
</div>
</div>
<div class="row margin-bottom-10">
<!-- Comments Form Input -->
<div class="medium-12 columns">
<label for="comments">Comments
{!! Form::textarea('comments', $medication_regimen->
    comments,
['class' => 'default', 'id' => 'comments', 'rows' => '3
    ',
'placeholder' => 'Input if there are additional notes
    about the regimen.']) !!}
@foreach ($errors->get('comments') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('comments')) - 1)<br>
    @endif
@endforeach
</label>
</div>
</div>
<p class="help-text">Fields marked with * are mandatory
    .</p>
<!-- Submit Button -->
<div class="row column center">
{!! Form::submit($submit_button_text, ['class' => '
    button',]) !!}
@if ($medication_regimen->patient)
<a class="button hollow"
href="{{ url('medication-regimens/'. .
    $medication_regimen->patient->id) }}">
Cancel
</a>
@else
<a class="button hollow"
href="{{ url('medication-regimens/'. .$patient->id) }}"
    >
Cancel
</a>
@endif
</div>
```

Medication Regimens Create View

```
@extends('app')
@section('css')@endsection
@section('content')
<div class="row">
<div class="small-10 small-centered medium-10 medium-
    centered large-10 large-centered column">
<div class="center"><h4>Add Medication Regimen</h4></
    div>
<hr>
<div class="row column margin-bottom-10">
{!! Form::model($medication_regimen = new \App\Models\
    MedicationRegimen, ['method' => 'POST',
'url' => 'medication-regimens/' . $patient->id . '/
    create']) !!}
@include('regimens.medication_regimens._form',
['submit_button_text' => 'Add Medication Regimen'])
{!! Form::close() !!}
</div>
</div>
```

```
</div>
@endsection
@section('js')@endsection
```

Medication Regimens Edit View

```
@extends('app')
@section('css')@endsection
@section('content')
<div class="row">
<div class="small-10 small-centered medium-10 medium-
    centered large-10 large-centered column">
<div class="center"><h4>Edit Medication Regimen</h4></
    div>
<hr>
<div class="row column margin-bottom-10">
{!! Form::model($medication_regimen, ['method' => '
    PATCH',
'url' => 'medication-regimens/edit/' .
    $medication_regimen->id]) !!}
@include('regimens.medication_regimens._form',
['submit_button_text' => 'Edit Medication Regimen'])
{!! Form::close() !!}
</div>
</div>
</div>
@endsection
@section('js')@endsection
```

Medication Regimens Index View

```
@extends('app')
@section('css'){!! Html::style('css/responsive-tables.
    css') !!}@endsection
@section('content')
<div class="row">
<div class="small-10 small-centered medium-10 medium-
    centered large-10 large-centered column">
<div class="center"><h4>Daily Medication Regimens: {{
    $patient_full_name }}</h4></div>
<hr>
@if ($is_staff)
<div class="row column margin-bottom-20 menu-centered">
<ul class="vertical medium-horizontal menu">
<li><a href="{{ url('patients/' . $patient->id) }}">
    Account Profile</a></li>
@if ($user->medicalProfessional)
<li><a href="{{ url('patient-attributes/' . $patient->
    id) }}">General Details</a></li>
<li><a href="{{ url('medical-conditions/' . $patient->
    id) }}">Daily Medical Condition</a></li>
<li><a href="{{ url('laboratory-results/' . $patient->
    id) }}">Laboratory Results</a></li>
<li><a href="{{ url('medical-records/' . $patient->id)
    }}">Medical Records</a></li>
<li><a href="{{ url('referral/' . $patient->id) }}">
    Referral</a></li>
<li class="menu-text">Medication Regimens</li>
@endif
</ul>
</div>
@endif
<div class="row column margin-bottom-10 callout">
@if (isset($regimens) and count($regimens) > 0)
<table class="responsive small" width="100%">
<tr>
<th>Time of Medication</th>
<th>Medication</th>
<th>Dose</th>
<th>Frequency</th>
<th>Comments</th>
@if ($is_staff)
<th></th>
<th></th>
@endif
</tr>
@foreach ($regimens as $index => $regimen)
<tr>
<td>{{ $regimen->time_of_medication }}</td>
<td>{{ $regimen->medication }}</td>
<td>
@if ($regimen->dose > 1) {{ $regimen->dose }}
    milligrams
@else {{ $regimen->dose }} milligram @endif
</td>
<td>
@if ($regimen->frequency > 1) {{ $regimen->frequency }}
     times a day
@else Once a day @endif
</td>
<td>{{ $regimen->comments }}</td>
@if ($is_staff)
<td>
<a href="{{ url('medication-regimens/edit/' . $regimen
    ->id) }}"
class="button">
Edit
</a>
</td>
<td>
{!! Form::open(['method' => 'DELETE',
'url' => 'medication-regimens/delete/' . $regimen->id])
     !!}
{!! Form::submit('Delete', ['class' => 'button alert'])
     !!}
{!! Form::close() !!}
</td>
@endif
</tr>
@endforeach
</table>
@else
<p class="font-0875rem">There are currently no
    medication regimens available.</p>
@endif
</div>
@if ($is_staff)
<div class="row column center">
<a href="{{ url('medication-regimens/' . $patient->id .
    '/create') }}"
```

```
class="button">
Add Medication Regimen
</a>
<a href="{{ url('patients/'. $patient->id) }}" class="
    button hollow">
Cancel
</a>
</div>
@endif
</div>
</div>
@endsection
@section('js'){!! Html::script('js/responsive-tables.js
    ') !!} @endsection
```

Nutritionist Regimens Form View

```
<div class="row column">
<p class="help-text">
Answer the following fields about the daily nutrition
    regimen prescribed to the patient.
</p>
<hr>
</div>
<div class="row margin-bottom-10">
<!-- An Option for Breakfast Form Input -->
<div class="medium-12 columns">
<label for="breakfast">An Option for Breakfast
{!! Form::textarea('breakfast', $nutritionist_regimen->
    breakfast,
['class' => 'default', 'id' => 'breakfast', 'rows' => '
    3', 'placeholder' => '']) !!}
@foreach ($errors->get('breakfast') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('breakfast')) - 1)<br>
    @endif
@endforeach
</label>
</div>
</div>
<div class="row margin-bottom-10">
<!-- An Option for Lunch Form Input -->
<div class="medium-12 columns">
<label for="lunch">An Option for Lunch
{!! Form::textarea('lunch', $nutritionist_regimen->
    lunch,
['class' => 'default', 'id' => 'lunch', 'rows' => '3',
    'placeholder' => '']) !!}
@foreach ($errors->get('lunch') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('lunch')) - 1)<br>
    @endif
@endforeach
</label>
</div>
</div>
<div class="row margin-bottom-10">
<!-- An Option for Dinner Form Input -->
<div class="medium-12 columns">
<label for="dinner">An Option for Dinner
{!! Form::textarea('dinner', $nutritionist_regimen->
    dinner,
['class' => 'default', 'id' => 'dinner', 'rows' => '3',
    'placeholder' => '']) !!}
@foreach ($errors->get('dinner') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('dinner')) - 1)<br>
    @endif
@endforeach
</label>
</div>
</div>
<div class="row margin-bottom-10">
<!-- An Option for AM Snack Form Input -->
<div class="medium-12 columns">
<label for="am_snack">An Option for AM Snack
{!! Form::textarea('am_snack', $nutritionist_regimen->
    am_snack,
['class' => 'default', 'id' => 'am_snack', 'rows' => '3
    ', 'placeholder' => '']) !!}
@foreach ($errors->get('am_snack') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('am_snack')) - 1)<br>
    @endif
@endforeach
</label>
</div>
</div>
<div class="row margin-bottom-10">
<!-- An Option for PM Snack Form Input -->
<div class="medium-12 columns">
<label for="pm_snack">An Option for PM Snack
{!! Form::textarea('pm_snack', $nutritionist_regimen->
    pm_snack,
['class' => 'default', 'id' => 'pm_snack', 'rows' => '3
    ', 'placeholder' => '']) !!}
@foreach ($errors->get('pm_snack') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('pm_snack')) - 1)<br>
    @endif
@endforeach
</label>
</div>
</div>
<!-- Submit Button -->
<div class="row column center">
{!! Form::submit($submit_button_text, ['class' => '
    button',]) !!}
@if ($nutritionist_regimen->patient)
<a class="button hollow"
href="{{ url('nutritionist-regimens/'.
    $nutritionist_regimen->patient->id) }}">
Cancel
</a>
@else
<a class="button hollow"
href="{{ url('nutritionist-regimens/'. $patient->id)
    }}">
Cancel
</a>
```

```
@endif
</div>
```

Nutritionist Regimens Create View

```
@extends('app')
@section('css')@endsection
@section('content')
<div class="row">
<div class="small-10 small-centered medium-10 medium-
    centered large-10 large-centered column">
<div class="center"><h4>Add Nutrition Regimen</h4></div
    >
<hr>
<div class="row column margin-bottom-10">
{!! Form::model($nutritionist_regimen = new \App\Models
    \NutritionistRegimen, ['method' => 'POST',
'url' => 'nutritionist-regimens/' . $patient->id . '/
    create']) !!}
@include('regimens.nutritionist_regimens._form',
['submit_button_text' => 'Add Nutrition Regimen'])
{!! Form::close() !!}
</div>
</div>
</div>
@endsection
@section('js')@endsection
```

Nutritionist Regimens Edit View

```
@extends('app')
@section('css')@endsection
@section('content')
<div class="row">
<div class="small-10 small-centered medium-10 medium-
    centered large-10 large-centered column">
<div class="center"><h4>Edit Nutrition Regimen</h4></
    div>
<hr>
<div class="row column margin-bottom-10">
{!! Form::model($nutritionist_regimen, ['method' => '
    PATCH',
'url' => 'nutritionist-regimens/edit/' .
    $nutritionist_regimen->id]) !!}
@include('regimens.nutritionist_regimens._form',
['submit_button_text' => 'Edit Nutrition Regimen'])
{!! Form::close() !!}
</div>
</div>
</div>
@endsection
@section('js')@endsection
```

Nutritionist Regimens Index View

```
@extends('app')
@section('css'){!! Html::style('css/responsive-tables.
    css') !!}@endsection
@section('content')
<div class="row">
<div class="small-10 small-centered medium-10 medium-
    centered large-10 large-centered column">
<div class="center"><h4>Daily Nutrition Regimens: {{
    $patient_full_name }}</h4></div>
<hr>
@if ($is_staff)
<div class="row column margin-bottom-20 menu-centered">
<ul class="vertical medium-horizontal menu">
<li><a href="{{ url('patients/' . $patient->id) }}">
    Account Profile</a></li>
@if ($user->nutritionist)
<li><a href="{{ url('nutritionist-records/' . $patient
    ->id) }}">Nutrition Records</a></li>
<li class="menu-text">Nutrition Regimens</li>
@endif
</ul>
</div>
@endif
<div class="row column margin-bottom-10 callout">
@if (isset($regimens) and count($regimens) > 0)
<div class="row margin-bottom-20">
<div class="medium-10 medium-offset-1 columns end">
<div class="center"><h5>Breakfast</h5></div>
<table class="responsive small" width="100%">
<tr>
<th>Options</th>
<th></th>
<th></th>
</tr>
@foreach ($regimens as $index => $regimen)
@if ($regimen->breakfast)
<tr>
<td>{{ $index + 1 }}. {{ $regimen->breakfast }}</td>
@if ($is_staff)
<td>
<a href="{{ url('nutritionist-regimens/edit/' .
    $regimen->id) }}"
class="button">
Edit
</a>
</td>
<td>
{!! Form::open(['method' => 'DELETE',
'url' => 'nutritionist-regimens/delete/' . $regimen->id
    ]) !!}
{!! Form::submit('Delete', ['class' => 'button alert'])
    !!}
{!! Form::close() !!}
</td>
@endif
</tr>
@endif
@endforeach
</table>
</div>
</div>
<div class="row margin-bottom-20">
<div class="medium-10 medium-offset-1 columns end">
<div class="center"><h5>Lunch</h5></div>
<table class="responsive small" width="100%">
```

```
<tr>
<th>Options</th>
<th></th>
<th></th>
</tr>
@foreach ($regimens as $index => $regimen)
@if ($regimen->lunch)
<tr>
<td>{{ $index + 1 }}. {{ $regimen->lunch }}</td>
@if ($is_staff)
<td>
<a href="{{ url('nutritionist-regimens/edit/'. .
    $regimen->id) }}"
class="button">
Edit
</a>
</td>
<td>
{!! Form::open(['method' => 'DELETE',
'url' => 'nutritionist-regimens/delete/' . $regimen->id
    ]) !!}
{!! Form::submit('Delete', ['class' => 'button alert'])
     !!}
{!! Form::close() !!}
</td>
@endif
</tr>
@endif
@endforeach
</table>
</div>
</div>
<div class="row margin-bottom-20">
<div class="medium-10 medium-offset-1 columns end">
<div class="center"><h5>Dinner</h5></div>
<table class="responsive small" width="100%">
<tr>
<th>Options</th>
<th></th>
<th></th>
</tr>
@foreach ($regimens as $index => $regimen)
@if ($regimen->dinner)
<tr>
<td>{{ $index + 1 }}. {{ $regimen->dinner }}</td>
@if ($is_staff)
<td>
<a href="{{ url('nutritionist-regimens/edit/'. .
    $regimen->id) }}"
class="button">
Edit
</a>
</td>
<td>
{!! Form::open(['method' => 'DELETE',
'url' => 'nutritionist-regimens/delete/' . $regimen->id
    ]) !!}
{!! Form::submit('Delete', ['class' => 'button alert'])
     !!}
{!! Form::close() !!}
</td>
@endif
</tr>
@endif
@endforeach
</table>
</div>
</div>
<div class="row margin-bottom-20">
<div class="medium-10 medium-offset-1 columns end">
<div class="center"><h5>AM Snack</h5></div>
<table class="responsive small" width="100%">
<tr>
<th>Options</th>
<th></th>
<th></th>
</tr>
@foreach ($regimens as $index => $regimen)
@if ($regimen->am_snack)
<tr>
<td>{{ $index + 1 }}. {{ $regimen->am_snack }}</td>
@if ($is_staff)
<td>
<a href="{{ url('nutritionist-regimens/edit/'. .
    $regimen->id) }}"
class="button">
Edit
</a>
</td>
<td>
{!! Form::open(['method' => 'DELETE',
'url' => 'nutritionist-regimens/delete/' . $regimen->id
    ]) !!}
{!! Form::submit('Delete', ['class' => 'button alert'])
     !!}
{!! Form::close() !!}
</td>
@endif
</tr>
@endif
@endforeach
</table>
</div>
</div>
<div class="row margin-bottom-20">
<div class="medium-10 medium-offset-1 columns end">
<div class="center"><h5>PM Snack</h5></div>
<table class="responsive small" width="100%">
<tr>
<th>Options</th>
<th></th>
<th></th>
</tr>
@foreach ($regimens as $index => $regimen)
@if ($regimen->pm_snack)
<tr>
<td>{{ $index + 1 }}. {{ $regimen->pm_snack }}</td>
@if ($is_staff)
<td>
<a href="{{ url('nutritionist-regimens/edit/'. .
```

```
    $regimen->id) }}"
class="button">
Edit
</a>
</td>
<td>
{!! Form::open(['method' => 'DELETE',
'url' => 'nutritionist-regimens/delete/' . $regimen->id
    ]) !!}
{!! Form::submit('Delete', ['class' => 'button alert'])
     !!}
{!! Form::close() !!}
</td>
@endif
</tr>
@endif
@endforeach
</table>
</div>
</div>
@else
<p class="font-0875rem">There are currently no
    nutritionist regimens available.</p>
@endif
</div>
</div>
@if ($is_staff)
<div class="row column center">
<a href="{{ url('nutritionist-regimens/' . $patient->id
    . '/create') }}"
class="button">
Add Nutrition Regimen
</a>
<a href="{{ url('patients/' . $patient->id) }}" class="
    button hollow">
Cancel
</a>
</div>
@endif
</div>
</div>
@endsection
@section('js') {!! Html::script('js/responsive-tables.js
    ') !!} @endsection
```

Physiatrist Regimens Form View

```
<div class="row column">
<p class="help-text">
Answer the following fields about the daily
    rehabilitation medicine regimen prescribed to
the patient.
</p>
<hr>
</div>
<div class="row margin-bottom-10">
<!-- Type of Exercise Form Input -->
<div class="medium-6 columns">
<label for="type_of_exercise">Type of Exercise*
{!! Form::text('type_of_exercise', $physiatrist_regimen
    ->type_of_exercise,
['class' => 'default', 'id' => 'type_of_exercise',
'aria-describedby' => 'type_of_exercise_helper']) !!}
@foreach ($errors->get('type_of_exercise') as $key =>
    $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('type_of_exercise')) -
    1)<br>@endif
@endforeach
</label>
<p class="help-text" id="type_of_exercise_helper">
Input the type of exercise recommended for the patient.
</p>
</div>
<!-- Frequency Form Input -->
<div class="medium-3 columns">
<label for="frequency">Frequency*
{!! Form::number('frequency', $physiatrist_regimen->
    frequency,
['class' => 'default', 'id' => 'frequency',
'aria-describedby' => 'frequency_helper']) !!}
@foreach ($errors->get('frequency') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('frequency')) - 1)<br>
    @endif
@endforeach
</label>
<p class="help-text" id="frequency_helper">
Input the number of times the exercise is performed per
     day.
</p>
</div>
<!-- Duration (in minutes) Form Input -->
<div class="medium-3 columns">
<label for="duration">Duration (in minutes)*
{!! Form::number('duration', $physiatrist_regimen->
    duration, ['class' => 'default',
'id' => 'duration', 'aria-describedby' => '
    duration_helper']) !!}
@foreach ($errors->get('duration') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('duration')) - 1)<br>
    @endif
@endforeach
</label>
<p class="help-text" id="duration_helper">
Input the duration of the whole activity in minutes.
</p>
</div>
</div>
<div class="row margin-bottom-10">
<!-- Time of Exercise Form Input -->
<div class="medium-8 columns">
<label for="time_of_exercise">Time of Exercise
{!! Form::text('time_of_exercise', $physiatrist_regimen
    ->time_of_exercise,
['class' => 'default', 'id' => 'time_of_exercise',
'aria-describedby' => 'time_of_exercise_helper']) !!}
@foreach ($errors->get('time_of_exercise') as $key =>
    $error)
```

```
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('time_of_exercise')) -
    1)<br>@endif
@endforeach
</label>
<p class="help-text" id="time_of_exercise_helper">
Input the suggested time of exercise depending on
    frequencies (e.g., 7:00am, 12:00pm,
and 5:00pm).
</p>
</div>
</div>
<div class="row margin-bottom-10">
<!-- Comments Form Input -->
<div class="medium-12 columns">
<label for="comments">Comments
{!! Form::textarea('comments', $physiatrist_regimen->
    comments,
['class' => 'default', 'id' => 'comments', 'rows' => '3
    ',
'placeholder' => 'Input if there are additional notes
    about the regimen.']) !!}
@foreach ($errors->get('comments') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('comments')) - 1)<br>
    @endif
@endforeach
</label>
</div>
</div>
<p class="help-text">Fields marked with * are mandatory
    .</p>
<!-- Submit Button -->
<div class="row column center">
{!! Form::submit($submit_button_text, ['class' => '
    button',]) !!}
@if ($physiatrist_regimen->patient)
<a class="button hollow"
href="{{ url('physiatrist-regimens/' .
    $physiatrist_regimen->patient->id) }}">
Cancel
</a>
@else
<a class="button hollow"
href="{{ url('physiatrist-regimens/' . $patient->id) }}
    ">
Cancel
</a>
@endif
</div>
```

Physiatrist Regimens Create View

```
@extends('app')
@section('css')@endsection
@section('content')
<div class="row">
<div class="small-10 small-centered medium-10 medium-
    centered large-10 large-centered column">
<div class="center"><h4>Add Rehabilitation Medicine
    Regimen</h4></div>
<hr>
<div class="row column margin-bottom-10">
{!! Form::model($physiatrist_regimen = new \App\Models\
    PhysiatristRegimen, ['method' => 'POST',
'url' => 'physiatrist-regimens/' . $patient->id . '/
    create']) !!}
@include('regimens.physiatrist_regimens._form',
['submit_button_text' => 'Add Rehabilitation Medicine
    Regimen'])
{!! Form::close() !!}
</div>
</div>
</div>
@endsection
@section('js')@endsection
```

Physiatrist Regimens Edit View

```
@extends('app')
@section('css')@endsection
@section('content')
<div class="row">
<div class="small-10 small-centered medium-10 medium-
    centered large-10 large-centered column">
<div class="center"><h4>Edit Rehabilitation Medicine
    Regimen</h4></div>
<hr>
<div class="row column margin-bottom-10">
{!! Form::model($physiatrist_regimen, ['method' => '
    PATCH',
'url' => 'physiatrist-regimens/edit/' .
    $physiatrist_regimen->id]) !!}
@include('regimens.physiatrist_regimens._form',
['submit_button_text' => 'Edit Rehabilitation Medicine
    Regimen'])
{!! Form::close() !!}
</div>
</div>
</div>
@endsection
@section('js')@endsection
```

Physiatrist Regimens Index View

```
@extends('app')
@section('css'){!! Html::style('css/responsive-tables.
    css') !!}@endsection
@section('content')
<div class="row">
<div class="small-10 small-centered medium-10 medium-
    centered large-10 large-centered column">
<div class="center"><h4>Daily Rehabilitation Medicine
    Regimens: {{ $patient_full_name }}</h4></div>
<hr>
@if ($is_staff)
<div class="row column margin-bottom-20 menu-centered">
<ul class="vertical medium-horizontal menu">
<li><a href="{{ url('patients/' . $patient->id) }}">
    Account Profile</a></li>
```

```
@if ($user->physiatrist)
<li><a href="{{ url('physiatrist-records/' . $patient->id) }}">Rehabilitation Medicine Records</a></li>
<li class="menu-text">Rehabilitation Medicine Regimens</li>
@endif
</ul>
</div>
@endif
<div class="row column margin-bottom-10 callout">
@if (isset($regimens) and count($regimens) > 0)
<table class="responsive small" width="100%">
<tr>
<th>Time of Exercise</th>
<th>Type of Exercise</th>
<th>Frequency</th>
<th>Duration</th>
<th>Comments</th>
@if ($is_staff)
<th></th>
<th></th>
@endif
</tr>
@foreach ($regimens as $index => $regimen)
<tr>
<td>{{ $regimen->time_of_exercise }}</td>
<td>{{ $regimen->type_of_exercise }}</td>
<td>
@if ($regimen->frequency > 1) {{ $regimen->frequency }} times a day
@else Once a day @endif
</td>
<td>
@if ($regimen->duration > 1) {{ $regimen->duration }} minutes
@else {{ $regimen->duration }} minute @endif
</td>
<td>{{ $regimen->comments }}</td>
@if ($is_staff)
<td>
<a href="{{ url('physiatrist-regimens/edit/' . $regimen->id) }}"
class="button">
Edit
</a>
</td>
<td>
{!! Form::open(['method' => 'DELETE',
'url' => 'physiatrist-regimens/delete/' . $regimen->id]) !!}
{!! Form::submit('Delete', ['class' => 'button alert']) !!}
{!! Form::close() !!}
</td>
@endif
</tr>
@endforeach
</table>
@else
<p class="font-0875rem">There are currently no rehabilitation medicine regimens available.</p>
@endif
</div>
@if ($is_staff)
<div class="row column center">
<a href="{{ url('physiatrist-regimens/' . $patient->id . '/create') }}"
class="button">
Add Rehabilitation Medicine Regimen
</a>
<a href="{{ url('patients/' . $patient->id) }}" class="button hollow">
Cancel
</a>
</div>
@endif
</div>
</div>
@endsection
@section('js'){!! Html::script('js/responsive-tables.js') !!}@endsection
```

System Admin Manage Local Admins View

```
@extends('app')
@section('css'){!! Html::style('css/responsive-tables.css') !!}@endsection
@section('content')
<div class="row">
<div class="small-10 small-centered medium-10 medium-centered large-10 large-centered column">
<div class="center"><h4>Manage Hospital Administrators</h4></div>
<hr>
<div class="row column center margin-bottom-10">
@if (count($hospitals) > 0)
<table class="responsive small" width="100%">
<tr>
<th>Hospital</th>
<th>Local Administrator</th>
</tr>
@foreach ($hospitals as $index => $hospital)
<tr>
<td>{{ $index + 1 }}. {{ $hospital->name }}</td>
<td>
@if ($hospital->localAdmin)
@if ($hospital->localAdmin->user)
{{ $hospital->localAdmin->user->username }} -
{{ $hospital->localAdmin->user->last_name }}, {{ $hospital->localAdmin->user->first_name }}
{{ $hospital->localAdmin->user->middle_name }}
@else <em>n/a</em>
@endif
@else <em>n/a</em>
@endif
</td>
</tr>
@endforeach
</table>
@else
```

```
There are currently no hospital records.
@endif
</div>
<div class="row column margin-bottom-10">
<div class="callout">
<h5>Set a Hospital Administrator</h5>
{!! Form::open(['method' => 'POST', 'url' => 'hospital-
    admins']) !!}
<div class="row margin-bottom-10">
<!-- Hospital Form Input -->
<div class="medium-6 columns">
<label for="hospital_id">Hospital
{!! Form::select('hospital_id', $hospital_list, null, [
    'class' => 'default',
'id' => 'hospital_id', 'aria-describedby' => '
    hospital_id_helper']) !!}
@foreach ($errors->get('hospital_id') as $key => $error
    )
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('hospital_id')) - 1)<br
    >@endif
@endforeach
</label>
<p class="help-text" id="hospital_id_helper">
Choose the hospital to be managed by the hospital
    administrator.
</p>
</div>
<!-- User Form Input -->
<div class="medium-6 columns">
<label for="user_id">User
{!! Form::select('user_id', $user_list, null, ['class'
    => 'default',
'id' => 'user_id', 'aria-describedby' => '
    user_id_helper']) !!}
@foreach ($errors->get('user_id') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('user_id')) - 1)<br>
    @endif
@endforeach
</label>
<p class="help-text" id="user_id_helper">
Choose the user who can be the hospital administrator.
</p>
</div>
</div>
<div class="row column margin-bottom-10">
<!-- Submit Button -->
<div class="center">
<br>{!! Form::submit('Save Changes', ['class' => '
    button']) !!}
</div>
</div>
{!! Form::close() !!}
</div>
</div>
</div>
</div>
@endsection
@section('js'){!! Html::script('js/responsive-tables.js
    ') !!}@endsection
```

System Admin Manage Supervisors View

```
@extends('app')
@section('css'){!! Html::style('css/responsive-tables.
    css') !!}@endsection
@section('content')
<div class="row">
<div class="small-10 small-centered medium-10 medium-
    centered large-10 large-centered column">
<div class="center"><h4>Manage Supervisors</h4></div>
<hr>
<div class="row column center margin-bottom-10">
@if (count($forum_categories) > 0)
<table class="responsive small" width="100%">
<tr>
<th>Forum Category</th>
<th>Supervisors</th>
</tr>
@foreach ($forum_categories as $index =>
    $forum_category)
<tr>
<td>{{ $index + 1 }}. {{ $forum_category->name }}</td>
<td>
@if (count($forum_category->supervisors) > 0)
@foreach ($forum_category->supervisors as $index =>
    $supervisor)
@if ($supervisor->user)
{{ $supervisor->user->username }} -
{{ $supervisor->user->last_name }},
{{ $supervisor->user->first_name }}
{{ $supervisor->user->middle_name }}
@if ($index < count($forum_category->supervisors) - 1)
|
@endif
@endif
@endforeach
@else <em>n/a</em>
@endif
</td>
</tr>
@endforeach
</table>
@else
There are currently no forum categories.
@endif
</div>
<div class="row column margin-bottom-10">
<div class="callout">
<h5>Set a Supervisor</h5>
<div class="row column">
{!! Form::open(['method' => 'POST', 'url' => '
    supervisors/neurology']) !!}
<div class="row margin-bottom-10">
<div class="medium-2 columns">
<br><p>Neurology</p>
</div>
<!-- User Form Input -->
<div class="medium-6 columns">
```

```
<label for="user_id">User
{!! Form::select('user_id', $user_list, null, ['class'
    => 'default',
'id' => 'user_id', 'aria-describedby' => '
    user_id_helper']) !!}
@foreach ($errors->get('user_id') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('user_id')) - 1)<br>
    @endif
@endforeach
</label>
<p class="help-text" id="user_id_helper">
Choose the user who can be a supervisor on neurology.
</p>
</div>
<div class="medium-3 columns end">
<!-- Submit Button -->
<div class="center">
<br>{!! Form::submit('Save Changes', ['class' => '
    button']) !!}
</div>
</div>
</div>
{!! Form::close() !!}
</div>
<div class="row column">
{!! Form::open(['method' => 'POST', 'url' => '
    supervisors/nutrition']) !!}
<div class="row margin-bottom-10">
<div class="medium-2 columns">
<br><p>Nutrition</p>
</div>
<!-- User Form Input -->
<div class="medium-6 columns">
<label for="user_id2">User
{!! Form::select('user_id2', $user_list2, null, ['class
    ' => 'default',
'id' => 'user_id2', 'aria-describedby' => '
    user_id2_helper']) !!}
@foreach ($errors->get('user_id2') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('user_id2')) - 1)<br>
    @endif
@endforeach
</label>
<p class="help-text" id="user_id2_helper">
Choose the user who can be a supervisor on nutrition.
</p>
</div>
<div class="medium-3 columns end">
<!-- Submit Button -->
<div class="center">
<br>{!! Form::submit('Save Changes', ['class' => '
    button']) !!}
</div>
</div>
</div>
{!! Form::close() !!}
</div>
<div class="row column">
{!! Form::open(['method' => 'POST', 'url' => '
    supervisors/rehabilitation-medicine']) !!}
<div class="row margin-bottom-10">
<div class="medium-2 columns">
<br><p>Rehabilitation Medicine</p>
</div>
<!-- User Form Input -->
<div class="medium-6 columns">
<label for="user_id3">User
{!! Form::select('user_id3', $user_list3, null, ['class
    ' => 'default',
'id' => 'user_id3', 'aria-describedby' => '
    user_id3_helper']) !!}
@foreach ($errors->get('user_id3') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('user_id3')) - 1)<br>
    @endif
@endforeach
</label>
<p class="help-text" id="user_id3_helper">
Choose the user who can be a supervisor on
    rehabilitation medicine.
</p>
</div>
<div class="medium-3 columns end">
<!-- Submit Button -->
<div class="center">
<br>{!! Form::submit('Save Changes', ['class' => '
    button']) !!}
</div>
</div>
</div>
{!! Form::close() !!}
</div>
</div>
</div>
<div class="row column margin-bottom-10">
<div class="callout">
<h5>Remove a Supervisor</h5>
<div class="row column">
{!! Form::open(['method' => 'DELETE', 'url' => '
    supervisors/remove-neurology']) !!}
<div class="row margin-bottom-10">
<div class="medium-2 columns">
<br><p>Neurology</p>
</div>
<!-- User Form Input -->
<div class="medium-6 columns">
<label for="user_id4">User
{!! Form::select('user_id4', $user_list, null, ['class'
     => 'default',
'id' => 'user_id4', 'aria-describedby' => '
    user_id4_helper']) !!}
@foreach ($errors->get('user_id4') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('user_id4')) - 1)<br>
    @endif
@endforeach
</label>
<p class="help-text" id="user_id4_helper">
```

```
Choose the user who will be removed from the
    supervisors on neurology.
</p>
</div>
<div class="medium-3 columns end">
<!-- Submit Button -->
<div class="center">
<br>{!! Form::submit('Save Changes', ['class' => '
    button']) !!}
</div>
</div>
</div>
{!! Form::close() !!}
</div>
<div class="row column">
{!! Form::open(['method' => 'DELETE', 'url' => '
    supervisors/remove-nutrition']) !!}
<div class="row margin-bottom-10">
<div class="medium-2 columns">
<br><p>Nutrition</p>
</div>
<!-- User Form Input -->
<div class="medium-6 columns">
<label for="user_id5">User
{!! Form::select('user_id5', $user_list2, null, ['class
    ' => 'default',
'id' => 'user_id5', 'aria-describedby' => '
    user_id5_helper']) !!}
@foreach ($errors->get('user_id5') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('user_id5')) - 1)<br>
    @endif
@endforeach
</label>
<p class="help-text" id="user_id5_helper">
Choose the user who will be removed from the
    supervisors on nutrition.
</p>
</div>
<div class="medium-3 columns end">
<!-- Submit Button -->
<div class="center">
<br>{!! Form::submit('Save Changes', ['class' => '
    button']) !!}
</div>
</div>
</div>
{!! Form::close() !!}
</div>
<div class="row column">
{!! Form::open(['method' => 'DELETE', 'url' => '
    supervisors/remove-rehabilitation-medicine']) !!}
<div class="row margin-bottom-10">
<div class="medium-2 columns">
<br><p>Rehabilitation Medicine</p>
</div>
<!-- User Form Input -->
<div class="medium-6 columns">
<label for="user_id6">User
{!! Form::select('user_id6', $user_list3, null, ['class
    ' => 'default',
'id' => 'user_id6', 'aria-describedby' => '
    user_id6_helper']) !!}
@foreach ($errors->get('user_id6') as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('user_id6')) - 1)<br>
    @endif
@endforeach
</label>
<p class="help-text" id="user_id6_helper">
Choose the user who will be removed from the
    supervisors on rehabilitation medicine.
</p>
</div>
<div class="medium-3 columns end">
<!-- Submit Button -->
<div class="center">
<br>{!! Form::submit('Save Changes', ['class' => '
    button']) !!}
</div>
</div>
</div>
{!! Form::close() !!}
</div>
</div>
</div>
</div>
</div>
@endsection
@section('js'){!! Html::script('js/responsive-tables.js
    ') !!}@endsection
```

Visit Index Patient View

```
@extends('app')
@section('css')@endsection
@section('content')
<div class="row">
<div class="small-10 small-centered medium-10 medium-
    centered large-10 large-centered column">
<div class="center"><h4>Schedule for Follow-Up Visit</
    h4></div>
<hr>
<div class="row column margin-bottom-10">
<div class="callout">
@if (count($follow_up_visits) > 0)
@foreach ($follow_up_visits as $index =>
    $follow_up_visit)
<div class="callout @if ($follow_up_visit->is_approved)
    success
@elseif ($follow_up_visit->medicalCondition)
@if ($follow_up_visit->medicalCondition->
    decisionOnCondition) primary @endif @endif">
<p>
{{ $index + 1 }}. Schedule for follow-up visit: {{
    $follow_up_visit->date_of_visit[1] }}<br>
@if ($follow_up_visit->user)
@if ($follow_up_visit->user->medicalProfessional)
    Medical Professional:
@elseif ($follow_up_visit->user->nutritionist)
```

```
    Nutritionist:
@elseif ($follow_up_visit->user->physiatrist)
    Physiatrist:
@endif
{{ $follow_up_visit->user->last_name }}, {{
    $follow_up_visit->user->first_name }} {{
    $follow_up_visit->user->middle_name }}<br>
@endif
Hospital: {{ $patient->hospital->name }}<br>
@if (! $follow_up_visit->medicalCondition)
@if (! $follow_up_visit->is_approved) This set of
    schedule is not yet approved.
@else This set of schedule is already approved by the
    health professional.
@endif<br>
@elseif ($follow_up_visit->medicalCondition->
    decisionOnCondition)
Medical Professional: {{ $follow_up_visit->
    medicalCondition->decisionOnCondition->
    medicalProfessional->user->first_name }}
{{ $follow_up_visit->medicalCondition->
    decisionOnCondition->medicalProfessional->user->
    last_name }}<br><br>

Decision: {{ $decisions[$follow_up_visit->
    medicalCondition->decisionOnCondition->decision]
    }}<br>
Additional Notes:
@if ($follow_up_visit->medicalCondition->
    decisionOnCondition->additional_notes)
{{ $follow_up_visit->medicalCondition->
    decisionOnCondition->additional_notes }}
@else <em>n/a</em>
@endif
@endif
</p>
</div>
@endforeach
@endif
</div>
<div class="callout">
@if ($patient->medicalProfessional)
{!! Form::open(['method' => 'POST', 'url' => 'schedule-
    visits/medical-professional']) !!}
<div class="row margin-bottom-10">
<!-- Schedule a follow-up visit to medical professional
     clinic Form Input -->
<div class="medium-6 medium-offset-3 columns end">
<label for="schedule_medical_professional">Schedule a
    follow-up visit to medical professional clinic
{!! Form::date('schedule_medical_professional', null, [
    'class' => 'default',
'id' => 'schedule_medical_professional', 'min' =>
    $current_date->format('Y-m-d')]) !!}
@foreach ($errors->get('schedule_medical_professional')
     as $key => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('
    schedule_medical_professional')) - 1)<br>@endif
@endforeach
</label>
<div class="center">
{!! Form::submit('Submit Schedule', ['class' => 'button
    ']) !!}
</div>
</div>
</div>
{!! Form::close() !!}
@endif
@if ($patient->nutritionist)
{!! Form::open(['method' => 'POST', 'url' => 'schedule-
    visits/nutritionist']) !!}
<div class="row margin-bottom-10">
<!-- Schedule a follow-up visit to nutritionist clinic
    Form Input -->
<div class="medium-6 medium-offset-3 columns end">
<label for="schedule_nutritionist">Schedule a follow-up
     visit to nutritionist clinic
{!! Form::date('schedule_nutritionist', null, ['class'
    => 'default',
'id' => 'schedule_nutritionist', 'min' => $current_date
    ->format('Y-m-d')]) !!}
@foreach ($errors->get('schedule_nutritionist') as $key
     => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('schedule_nutritionist'
    )) - 1)<br>@endif
@endforeach
</label>
<div class="center">
<br>{!! Form::submit('Submit Schedule', ['class' => '
    button']) !!}
</div>
</div>
</div>
{!! Form::close() !!}
@endif
@if ($patient->physiatrist)
{!! Form::open(['method' => 'POST', 'url' => 'schedule-
    visits/physiatrist']) !!}
<div class="row margin-bottom-10">
<!-- Schedule a follow-up visit to physiatrist clinic
    Form Input -->
<div class="medium-6 medium-offset-3 columns end">
<label for="schedule_physiatrist">Schedule a follow-up
    visit to physiatrist clinic
{!! Form::date('schedule_physiatrist', null, ['class'
    => 'default',
'id' => 'schedule_physiatrist', 'min' => $current_date
    ->format('Y-m-d')]) !!}
@foreach ($errors->get('schedule_physiatrist') as $key
    => $error)
<small class="custom-form-error">{{ $error }}</small>
@if ($key == count($errors->get('schedule_physiatrist')
    ) - 1)<br>@endif
@endforeach
</label>
<div class="center">
<br>{!! Form::submit('Submit Schedule', ['class' => '
    button']) !!}
```

```
</div>
</div>
</div>
{!! Form::close() !!}
@endif
</div>
</div>
</div>
</div>
@endsection
@section('js')@endsection
```

Visit Index Staff View

```
@extends('app')
@section('css')@endsection
@section('content')
<div class="row">
<div class="small-10 small-centered medium-10 medium-
    centered large-10 large-centered column">
<div class="center"><h4>Approve Patient Request Visits
    </h4></div>
<hr>
<div class="row column margin-bottom-10">
<div class="callout">
@if (count($follow_up_visits) > 0)
@foreach ($follow_up_visits as $index =>
    $follow_up_visit)
<div class="callout @if ($follow_up_visit->is_approved)
    success @endif">
<div class="row">
<div class="medium-6 @if (! $follow_up_visit->
    is_approved) medium-offset-2
@else medium-offset-3 @endif columns">
<p>
{{ $index + 1 }}. Patient: {{ $follow_up_visit->patient
    ->user->last_name }},
{{ $follow_up_visit->patient->user->first_name }}
{{ $follow_up_visit->patient->user->middle_name }}<br>
Schedule for follow-up visit: {{ $follow_up_visit->
    date_of_visit[1] }}<br>
Hospital: {{ $follow_up_visit->patient->hospital->name
    }}<br>
@if ($follow_up_visit->is_approved)
This schedule for follow-up visit is approved.
@endif
</p>
</div>
@if (! $follow_up_visit->is_approved)
<div class="medium-4 columns">
{!! Form::open(['method' => 'POST', 'url' => 'approve-
    visits']) !!}
{!! Form::hidden('follow_up_visit_id', $follow_up_visit
    ->id) !!}
{!! Form::submit('Approve', ['class' => 'button']) !!}
{!! Form::close() !!}
</div>
@endif
</div>
</div>
@endforeach
@endif
</div>
</div>
</div>
</div>
@endsection
@section('js')@endsection
```

App View

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width,
    initial-scale=1">
<title>{{ $title }}</title>

{!! Html::style('css/normalize.css') !!}
{!! Html::style('css/foundation.css') !!}
{!! Html::style('css/foundation-profile-card.css') !!}
@yield('css')
{!! Html::style('css/master.css') !!}
<link rel="shortcut icon" href="{{ url('files/logos/
    icon-psp.ico') }}">

<!-- HTML5 shim and Respond.js for IE8 support of HTML5
     elements and media queries -->
<!-- WARNING: Respond.js doesn't work if you view the
    page via file:// -->
<!--[if lt IE 9]>
<script src="https://oss.maxcdn.com/html5shiv/3.7.2/
    html5shiv.min.js"></script>
<script src="https://oss.maxcdn.com/respond/1.4.2/
    respond.min.js"></script>
<![endif]-->
</head>
<body>
@include('partials._title')

@include('partials._nav')

<div class="row custom-main-part no-margin">
<div class="large-2 medium-3 columns custom-menu no-
    padding show-for-medium">
@include('partials._menu')
</div>

<div class="small-12 columns custom-menu no-padding
    show-for-small-only">
@include('partials._menu')
</div>

<div class="large-10 medium-9 columns custom-body">
@include('partials._status')
@yield('content')
</div>
</div>
```

```
<div class="row column custom-footer no-margin">
@include('partials._footer')
</div>

{!! Html::script('js/jquery.js') !!}
{!! Html::script('js/modernizr.js') !!}
{!! Html::script('js/what-input.js') !!}
{!! Html::script('js/foundation.js') !!}
@yield('js')
</body>
</html>
```

XI. Acknowledgement

Hello all!!!

Una sa lahat, gusto kong pasalamatan si Doc Magboo sa pagiging matiyaga sa akin at sa SP. Grabe, kahit ilang beses ko nang naisipang sukuan 'to, ikaw Doc, hindi ka nawalan ng tiwala sakin all throughout ng taon na ito. Sinuportahan mo ko sa abot ng makakaya niyo. Maraming maraming salamat Doc! Iba ka! Salamat po sa lahat!

Maraming salamat rin, Mommy at Daddy, sa suporta ninyo sakin, lalo sa mga araw at gabi na pinanghihinaan ako ng loob sa dami ng ginagawa. Salamat sa pagtimpla lagi ng gatas saka sa pagdala ng donut tuwing umaga para makakain ako habang nagco-code hahaha. Salamat Mommy sa pag-akap mo sakin lagi lalo kapag nahihirapan na ko sa SP. Salamat Daddy sa suporta, hanggang sa pagsama sakin sa pagprint nitong document nang gabi. Grabe, para sa inyo ito! Mahal na mahal ko kayo! HAHA cheesy.

Maraming salamat rin sa blockmates ko! Noong mga buwan na naghahanda tayo para sa proposal and defense, sama-sama tayong lumusong sa sunud-sunod na Hell Weekssssss. Biruin niyo, may proposal at defense na tayo, meron ding kilotons of exams. Pero nakayanan natin! Hindi ko makakayanan lahat nang 'yon kung di ko kayo kasama. Kaya salamat sa inyong lahat! Labyu guys.

Special mention ko pala 'ung mga sumuporta sakin na blockmates. Salamat Krizia! Salamat sa suporta mo sakin nung time na 'di ko na talaga kinaya tong project na to (HAHA). Kahit puro patawa and pang-aasar ung mga sinabi mo sakin that time, nabuhayan pa rin ako ng loob, as in hahaha. Salamat rin sa mga kwentuhan events pag sabay tayong umuuwi sa bus. Okay na sana eh kaso mahilig kang mang-asar HAHAHAJK.

Salamat Earl! Sa mga kwentuhan natin tungkol sa mga pangarap, pagkatao, pag-ibig (HAHA), and other sensical things, lalo nung proposal season. Salamat sa pakikiramay sa mga nagaganap at that time. Salamat rin sa pagpapahiram mo ng The Alchemist! Balang araw, ipapahiram ko sayo lahat ng libro ni Coelho

hahaha.

Sa mga kapwa kong advisees na sina Bella at Adriel, maraming salamat rin! Salamat sa pakikiramay lalo nung defense season. Salamat sa group chat na DocBBs HAHAHAHA. And Bella, salamat rin sa mga self-proclaimed consultations natin every week nung second sem. Nainspire ako gumawa lalo nung sign-up page pa lang meron sa system ko HAHAHA.

Uyy ung isa dyan, alam ko binabasa mo to ngayon. Main event to HAHA. Reinier Tiglao Maristela! Pre! Maraming salamat! Salamat nang lubos sa unli mong suporta sakin. Salamat rin sa every week na kwentuhan natin tungkol sa iba't ibang bagay. Lagi kong hinihintay ung Friday para run HAHA. Marami akong natutunan sayo hahaha. 'Wag ka mag-alala pre, suportado rin kita sa lahat ng tatahakin mo sa buhay. Basta ah, walang iwanan; sabay nating puntahan mga pangarap natin sa buhay, pati na rin mga chix sa DLSU HAHAHAHAJK. Salamat pre!

At sa mga nagbabasa nito ngayon, wag ka rin mag-alala, kaya mo yan. Kayang-kaya mo yan. Mararating mo rin ang gusto mo. Hindi ko magagawa to kung hindi dahil sayo. Para sayo tong SP na ito. Wag kang susuko! Suportadong-suportado kita!