
# ASHESI UNIVERSITY COLLEGE

## CENTRALIZED PATIENT HEALTH MANAGEMENT SYSTEM FOR HOSPITALS AND PHARMACIES IN GHANA

**APPLIED PROJECT**

B.Sc. Computer Science

**Fredrick Oheneba Abayie**

**2016**

**ASHESI UNIVERSITY COLLEGE**

# Centralized Patient Health Management System for Hospitals and Pharmacies in Ghana

**APPLIED PROJECT**

Applied Project submitted to the Department of Computer Science, Ashesi University College in partial fulfilment of the requirements for the award of Bachelor of Science degree in Computer Science

**Fredrick Oheneba Abayie**

**April 2016**

## DECLARATION

I hereby declare that this applied project is the result of my own original work and that no part of it has been presented for another degree in this university or elsewhere.

Candidate's Signature:

……

Candidate's Name:

……

Date:

……

I hereby declare that preparation and presentation of this applied project were supervised in accordance with the guidelines on supervision of applied project laid down by Ashesi University College.

Supervisor's Signature:

……

Supervisor's Name:

……

Date:

……

## Acknowledgement

I would like to express my sincerest gratitude to my project supervisor Mr. Aelaf Dafla, for his guidance and support throughout this project.

# Abstract

This project aims to provide a centralized system where hospitals and pharmacies interact and share information to improve services and the health of patients. The application is developed in order to address the issue of hospitals and pharmacies not having updated records of patients. The system seeks to primarily help patients keep track of medications that were prescribed to them by the hospital and given to them by the pharmacy. The application will also allow the patient to check the times medications must be taken on the application.

Moreover, hospitals and pharmacies, through a centralized, system can access patient information and details. Both hospital and pharmacy will have an updated version of a patient's information whenever an update is made.

# Table of Contents

# Chapter 1: Introduction

## 1.1 Introduction and Background

This project is aimed at developing a software for hospitals and pharmacies in Ghana to help minimize self-medication and also to assist health personnel in keeping accurate records of patients. Most hospitals in Ghana do not have proper record keeping systems in place to record patient's details (including name, age, disease, prescriptions etc.) and manage them properly (Nyonmorworko, 2015). Again, patients do not get a formal report after visiting hospitals. This software will allow health personnel to enter details of patients as well as generate statistical reports and also to process health insurance claims. The software will allow pharmacies know more information about the medications prescribed to a patient by a doctor. Examples of such information will be the name of the patient, age, the hospital he/she visited and the patient's doctor.

### 1.1.1 Problem Statement

Hospitals record patient information paper which are then stored in paper folders. In a scenario where there is a fire outbreak or flooding in the hospitals, all the necessary documents of patients could be lost. This is a crucial problem because if the health of a patient is not tracked properly, it could lead to more complications or the patient losing his/her life (Poissant, Pereira, Tamblyn, & Kwawasumi, 2005). The time it takes for an OPD staff to search for the folder of a particular patient has an impact on quality of health services delivered. Furthermore, service delivery, waiting time and reimbursement are affected due to the format in which the national health insurance is submitted for processing (Sakyi, Atinga, & Adzei, 2012). The software will help in transitioning from paper based record

keeping to electronic record keeping, generate statistical reports and help health personnel to send health insurance claims.

### 1.1.2 Motivation

A pressing concern discovered in most hospitals is the time that patients wait for their files to be found before they get to see a doctor. It takes longer in cases where the patient has lost his hospital card or does not remember the last time he visited that hospital. This could be a difficult job for the stuff who are to search for a particular file amongst thousands of other files. In certain cases, the file of a patient could not be found or if found could be badly damaged. The time it takes a nurse to search for a particular patient's file could be reduced drastically when proper electronic record keeping systems such as Central~HealthMS are put in place and the outmoded paper methods are discarded (Poissant, Pereira, Tamblyn, & Kwawasumi, 2005).

Normally when a patient goes to a hospital, after waiting several minutes or even hours for his/her files to be found, the doctors write medication prescriptions on paper for patients to buy from a pharmacy. There are scenarios where the pharmacist mistakenly gives a patient the wrong medication or dose due to the poor visibility of the doctor's handwriting (Institute for Safe Medication Practices, 2009). The software will however enable doctors to send prescriptions of patients to pharmacists with ease. Also the details of patients will be stored using the software in a secured database. A database will serve as the folder for keeping records of patients who visit the hospital. It will also be highly effective in searching for a patient's health information because it is faster when searching from a database.

Moreover, the availability of the management software will help the hospital management keep proper records of important data. The management will be able to

generate statistical reports (including number of patients that visited the hospital in a month, the number of malaria cases treated within a week etc.) which will in the long run help to minimize certain diseases and improve the services of the hospitals.

## 1.2 Objectives

- Develop a software that will allow effective record keeping systems and structures for most hospitals in Ghana.
- Develop a software that will allow better communication between hospitals and pharmacies in Ghana to provide patients with the best of services.
- Develop a software to provide statistical reports for hospital management to effectively manage the daily activities in hospitals
- Develop a software to allow health personnel submit an electronic health insurance claim with ease.
- Develop a software to reduce the time it takes a nurse to search for a patient's file or update the details of a patient.

## 1.3 Research & Existing Solutions

A research conducted showed that most hospitals in Ghana encounter the problem of bad record keeping of patients' information (Adams, 2015). Through the research, three applications were discovered which were purposely developed to tackle these persisting challenges. The few hospitals that used software in managing patient information were mostly open-sourced software. Some of these software's were developed by foreign developers hence does not meet the demand of the hospitals in Ghana (Poikonen, 2009).

Some of the existing software found had difficult user interfaces that made users confused as to what to do next. The installation process of some of the software was hectic because it demanded other requirement before it could be up and running. Also, the computers in some of the hospitals were not powerful enough to support some of the software because they demanded more memory to be able to work efficiently.

After the research, it was concluded that most of the applications were developed without considering the platform or computer it will be operated on or the users of the software. The results and observations derived of the research prove the necessity of this project. Below are related works that are already in use;

### 1.3.1 Central~HealthMS

This application was developed by a graduate student in Ashesi University to help hospitals effectively keep records of patients that visit the hospital. The application allows system users to add patients and edit their details. It is also a platform that contains information of patients such as their medical records, visits to the hospital and the various hospitals visited.

The application accomplished its user requirement but the system requirement for the application was at risk. The application used old versions of programming languages and frameworks that had numerous security issues making the application unsecured and prone to attacks. Some of the forms in the application also do not check for unacceptable input which is another security issue.

### 1.3.2 GNUmed

This medical software aims to provide documentation, understanding and management of patient's health records. The software is accessible on both Linux and Windows Operating Systems.

After installation of the software for testing purposes as well as to learn how the software tackles the problems in the health sector, one challenge observed was that the instructions for installation was not straight forward. Hence for a user who has no programming background, the user will experience difficulties in installing and running the software.

# Chapter 2: Requirement Engineering

This chapter describes what the system sets out to do and the constraints on its implementation and operation; thus explaining the user and system functional requirements as well as the systems non-functional requirements.

Additionally, Unified Modeling Language (UML) will be used to represent the system at different perspectives. UML notation will be used to describe the interaction and structural models to illustrate how the components of the system communicate with one another.

## 2.1 Feasibility

The project aimed at solving existing problems such as record keeping and management of patients in hospitals and pharmacies in Ghana. These were existing challenges in the health sector that the project tackled. Also there was availability of useful and open source technologies (libraries, frameworks etc.) that made the implementation of the project possible.

The most vital of these resources was time which was limited for the implementation of the project. This was because the project comprised of complex systems hence more time was needed to think through the project critically on how to secure records and integrate it with other complex existing systems. Due to the time limit, the project did not need a huge budget to succeed. The time associated also affected the cost to be incurred in commencing with the project.

Moreover, the necessary skilled developer for the implementation of the project was also readily available. The skills required for the successful implementation of the project

was knowledge in PHP programming, android mobile development, Java, MySQLi, Apache Server and also the use of Adobe Illustrator for graphic designing.

## 2.2 Requirement Discovery

Before the system's design and architecture was modeled, the user and system requirements were gathered. Also information about required systems and existing systems were gathered to help come up with the requirement documentation. The process that was used in gathering requirements for the system stakeholders was interviews and observations. The users were presented with scenarios and paper prototypes to guide and help them understand some of the functions of the system. Discussed below are the resources that was utilized to gather the requirements to implement the system.

### 2.2.1 Requirement Gathering

The hospital personnel were a major factor in gathering the requirement for the system. Nurses, doctors, pharmacist and laboratory technicians were interviewed about the challenges they face when it comes to managing the records of patients. Also observations were made on how the hospital personnel conducted their various activities related to their services to patients. The processes they went through in registering a new patient, how records were managed and the challenges they faced were all documented.

To get an overview of how some of these problems were tackled, a related project implemented by a graduate was analyzed. The analysis of the project included how the systems architecture was designed, the listed user and system requirements and the challenges that were encountered during the implementation of the project.

Moreover, other requirements were gathered from the project supervisor through several interviews during the implementation of the project. With his experience in implementing and deploying health sector systems, he provided certain factors in the sector that needed more attention and analysis.

## 2.3 Requirement Elicitation and Analysis

For this system, requirement elicitation and analysis was needed to determine the domain of the application. Also it was important to highlight the services that the system will provide to its end-users. The following scenarios were written to capture and organize the information gathered for system analysis.

### 2.3.1 Scenarios

Mr. Kwame, a 26 years old business man, visits a hospital at Kwabenya due to a recent pain he has developed in his waist. Upon arrival to the hospital he goes to the Nurses to check his temperature, weight and height to update his health information on the system. In case Mr. Kwame was not already on the system, he will then be registered onto the system. After his health details have been updated, he is then assigned to Dr. Obaapa. The doctor then accesses Mr. Kwame's health information to view his track of illnesses and also adds the new symptoms the patient is witnessing to his hospital records. The doctor then refers him to the laboratory to undertake diagnostic tests.

At the laboratory, the technician can see the list of patients coming for test. Mr. Kwame's name appears on the list and is called by the technician. After the test, the results of the test are recorded against his health information.

Mr. Kwame then goes back to the doctor. By this time the doctor would have the updated lab results. Based on that, the doctor then prescribes medications to the patient.

When Mr. Kwame leaves the hospital, he then goes to the nearest pharmacy to get the medications that the doctor prescribed to him. He then presents his system identification to the pharmacist, the pharmacist then searches the system for the patient. When the patient is found, the pharmacist can then see the medications that were prescribed, the hospital the patient visited, the name of the doctor that gave the prescription and other relevant information.

The pharmacist, Dr. Pharm then gives the prescribed medications to the patient and indicates it on the system as delivered. When this is done Mr. Kwame can view his medications on the mobile application. It will also display the time and quantity of the medication he has to take. Mr. Kwame can pay for the medications the pharmacist is giving to him by using MTN mobile money.

### 2.3.2 Use Cases

After the scenarios were written, it made it possible to highlight the defined requirement of the system stakeholders. The following use case diagram was generated based on the analysis of the user scenarios. This diagram shows how the actors interacted with the system. The use case also gives a visual representation of the user scenarios defined. The type of interactions that toke place and the actors involved are illustrated in figure 2.1. The system has numerous actors and they all interacted with the system differently by performing different tasks.

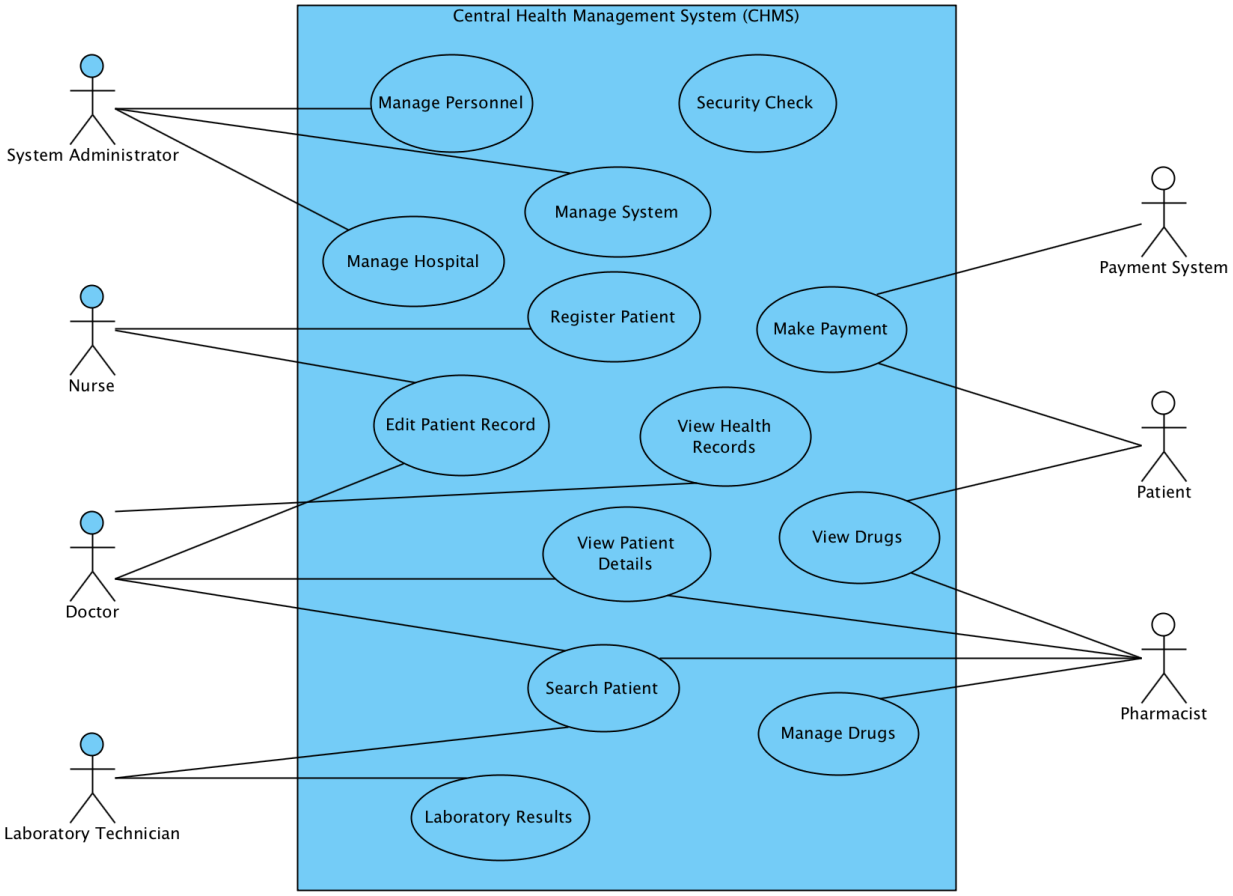


Figure 2.1: System use case diagram

System Users: All the system users must pass through a security check before they can get access to the system to perform their tasks. The security check involves a login page where users will be required to provide system username and password to be allowed access to the system.

System Administrator: The role of the system administrator is to manage other users of the system. These users are the health officials that use the system to perform their tasks. The management of personnel involves adding new users, editing their profile, granting and denying users access to the system. The administrator is also responsible for managing the

system. This will be done by adding users to different roles and each role has its own rights and privileges on the system. The management of hospitals and pharmacies is also another task for the system administrator. The admin can add, edit and disable new hospital and pharmacies on the system.

Patient: A patient will be able to view medications prescribed to them on the mobile application. Before a patient can access information on the system, an authentication must be provided to make the results of the request targeted to that particular patient only. Patients will not be able to alter the medications on the system since they can view them on the mobile application. Patients can also make payment to pharmacies after medications are handed over to them. The payment of medications will be done on the mobile application.

### 2.3.4 Process

To explain the processes in the scenario presented, an activity diagram was used to visualize the events involved in data processing. The diagram show cases the events that happened when a user interacted with Central~HealthMS. Figure 2.2 shows how three of the system stakeholders interacted with the system to complete a series of user requirement. Also it illustrates the process that the stakeholders of the system went through to deliver health care to a patient.

The processes were analyzed under hospital, pharmacist, and patient. With the scenario given the hospital searched for the patient in the database. No result was found hence a new record was created for the patient. The patient was assigned a doctor who referred the patient to the laboratory for tests. After the laboratory tests, the patient record

was updated on the system. The doctor then prescribed medications to the patient based on the laboratory results.

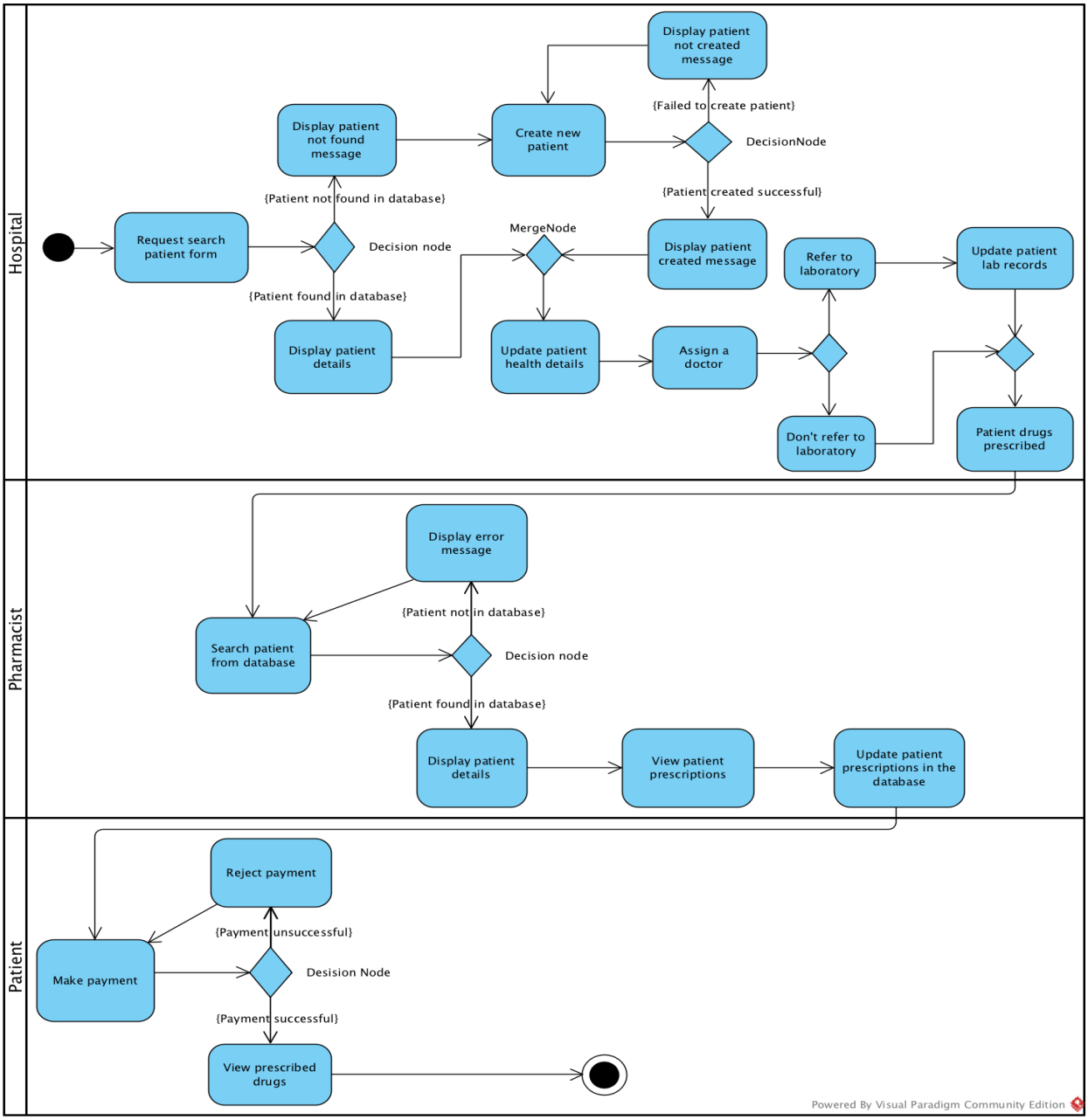


Figure 2.2: System activity diagram

Additionally, the pharmacist also searched for the patient to view his details including, the medications prescribed by the doctor. After the pharmacist specified the times

and quantity of drugs to be taken the patient was able to view the drugs prescribed to him by the doctor.

## 2.4 Functional Requirement

The functional requirement of the Health Management System describes what the users of the system can do. Thus the tasks each user of the system can perform and the constraints to those tasks.

### 2.4.1 System Administrator

The system administrator of the Health Management System has rights that the other users of the system do not have. But the system administrator cannot perform all the functions of the system. The administrator also has certain constraints. Listed below are the tasks the system administrator can and cannot perform in the system.

i. **Management of Doctors**: The administrator of the Health Management System will have access to the registration of a new doctor in the hospital. The administrator will also be able to view details, edit and search for registered doctors on the HMS. The administrator can also grant or deny a doctor access to the system.

ii. **Management of Nurses**: The administrator can also add new nurses, update their details and search for a particular nurse in the system. The administrator can grant or invoke a doctor's access to the system. The administrator can also grant or deny a nurse's access to the system.

iii. **Management of Laboratory Technicians**: Adding of laboratory technicians to the system can also be accomplished by the system administrator. The details of

technicians can also be updated and searched. The administrator can also grant or deny a laboratory technician access to the system.

iv. **Management of Patient**: The system administrator cannot add new patients to the system, prescribe medications to a patient and edit the details of a patient in the system. The system administrator does not have access to delete a patient from the system.

### 2.4.2 Doctors

The system doctors also have their own tasks that they can perform on the system. Their functionalities are not related to the other users of the system. There are also constraints on some functionalities for doctors are they therefore cannot perform all the functions of the system.

i. **Management of Patients Medical Records**: The doctor has access to view a patient's medical records. The doctor can search for a patient's record, can update or edit a patient's record and can also add a new medical record for a patient.

ii. **Management of Patient**: The doctor can search for a patient and view the patient's information but cannot add a new patient to the system, edit or update the patient's information.

### 2.4.3 Nurses

The nurses also have their unique functionalities on the system. They are also bound by constraints and can only perform tasks that they are allowed access to by the system administrator.

i. **Management of Patients Information**: The nurses can add a new patient, edit or update a patient's information but cannot prescribe medication to a patient.

ii. **Management of Profile**: A nurse does not have the access to edit individual profiles of themselves. That task is allowed by only the system administrator.

### 2.4.4 Pharmacist

The pharmacist in the hospital is also restricted on the system hence can perform only limited actions on the system.

i. **Updating of Patient Medication**: The pharmacist will be able to update a patient's medication. That is, indicating that the medications have been given to the patient.

ii. **Management of Medications**: Another task of the pharmacist is the management of drugs in the inventory. The pharmacist will be able to add new drugs and update drug details in the inventory.

### 2.4.5 Laboratory Technicians

The laboratory technicians are the users of the system with the least priority. They are also constraints by the action and functionality of the system.

i. **Management of Patient Labs**: The major task of the pharmacist on the system will be to manage and add the labs done by patients.

## 2.5 Non-Functional Requirement

The non-functional requirements of the system are the services that the Health Management System does not provide to the users of the system but are relevant for the operation of the system. These services include the security of the system, its performance, availability, how reliable the system will be, privacy, usability and the environment in which it will be used.

### 2.5.1 Product Requirement

The Health Management System shall be available any time an authorized user desires to access the system. Thus it will be available during all working hours and can be accessed by all users at any given time.

### 2.5.2 Organizational Requirement

Users of the Health Management System will have to login to the system by providing details given to them by the system administrator. Cross site scripting will be checked each time a user performs a task to prevent unauthorized data manipulation in the system.

### 2.5.3 External Requirement

The Health Management System shall implement security measures to make sure that, the confidentiality of patients are not breached in case the system is attacked by a hacker. The problem of cross-site scripting and request forgeries will be prevented in the implementation stages to make the application secure.

# Chapter 3: Architecture and Design

This chapter gives an overview of the design and architecture needed for the implementation of the Central~HealthMS. High-level designs are illustrated to represent the system's architecture and how it's components interact with one another. Diagrams will be used to visualize the user's interaction with the system and how data requested moves from one channel to the other.

The architectures that are discussed in the chapter includes the system's general architecture, how the mobile application interacts with the server, the unique identification architecture for users, the architectural pattern needed for implementation, the models and the database architecture.

## 3.1 System Architecture

Central~HealthMS is a web application that should be accessed by all the users at any given time. The system however follows the generic client-server architecture to be able to allow all the users to access the application.

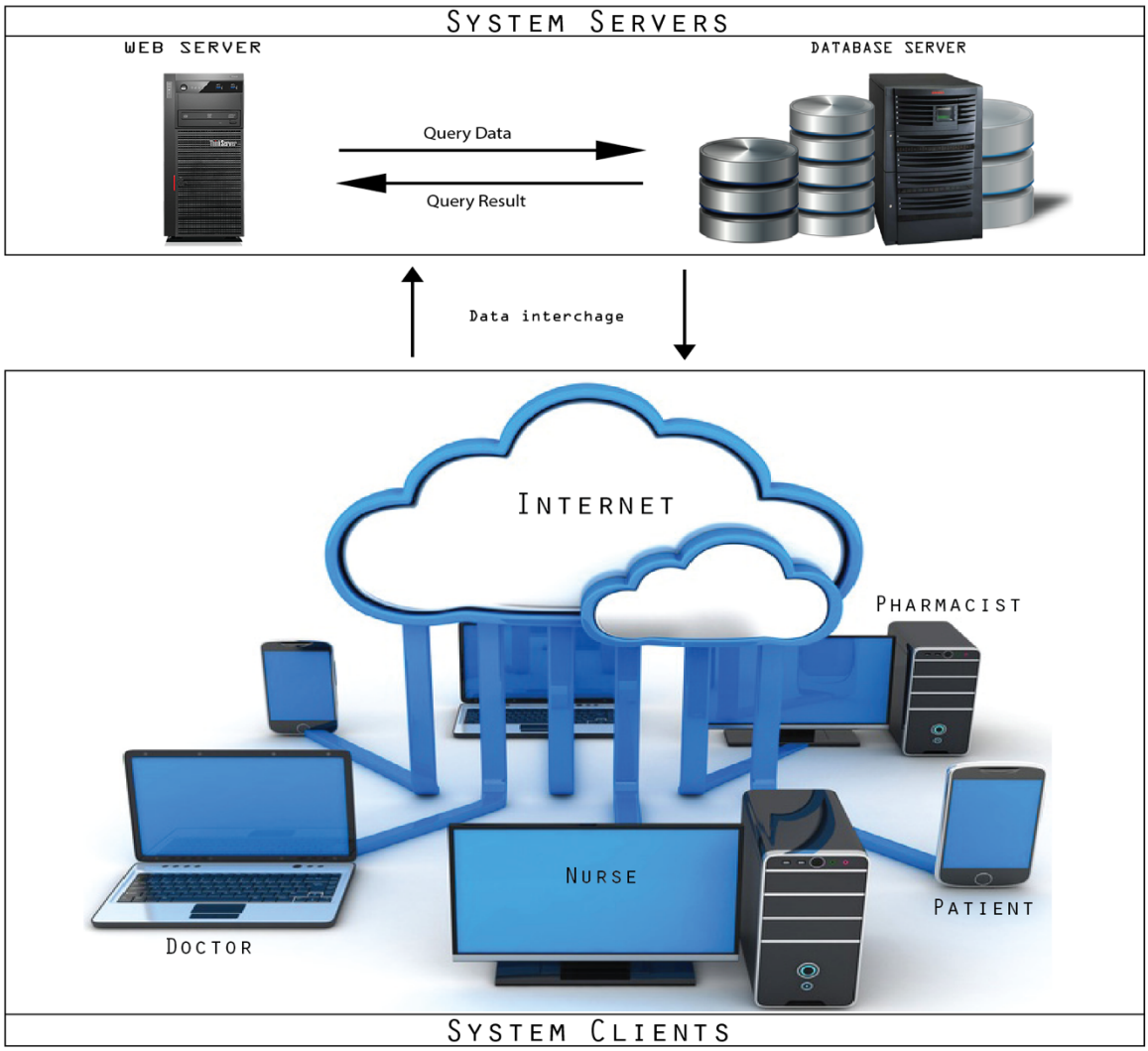


Figure 3.1: Client-server system architecture

For a client to access the application, there must be a stable Internet connection to be able to make an HTTPS request to the server hosting Central~HealthMS application. When the client requests for a page from the server, the client waits until it receives a reply in the form of content requested for from the server. This architecture is preferred because users can then access the application no matter their location. All they need is access to the internet.

### 3.1.1 Mobile Server Interaction

The mobile application for the patients follows the standards of a three tier architecture which includes the presentation, logic and database layer. The presentation layer of the mobile application consists of the interfaces that the patient interacts with to be able to perform the functions necessary. The logic layer is also responsible for processing and handling user requests fetching the data from the database and updating the view.

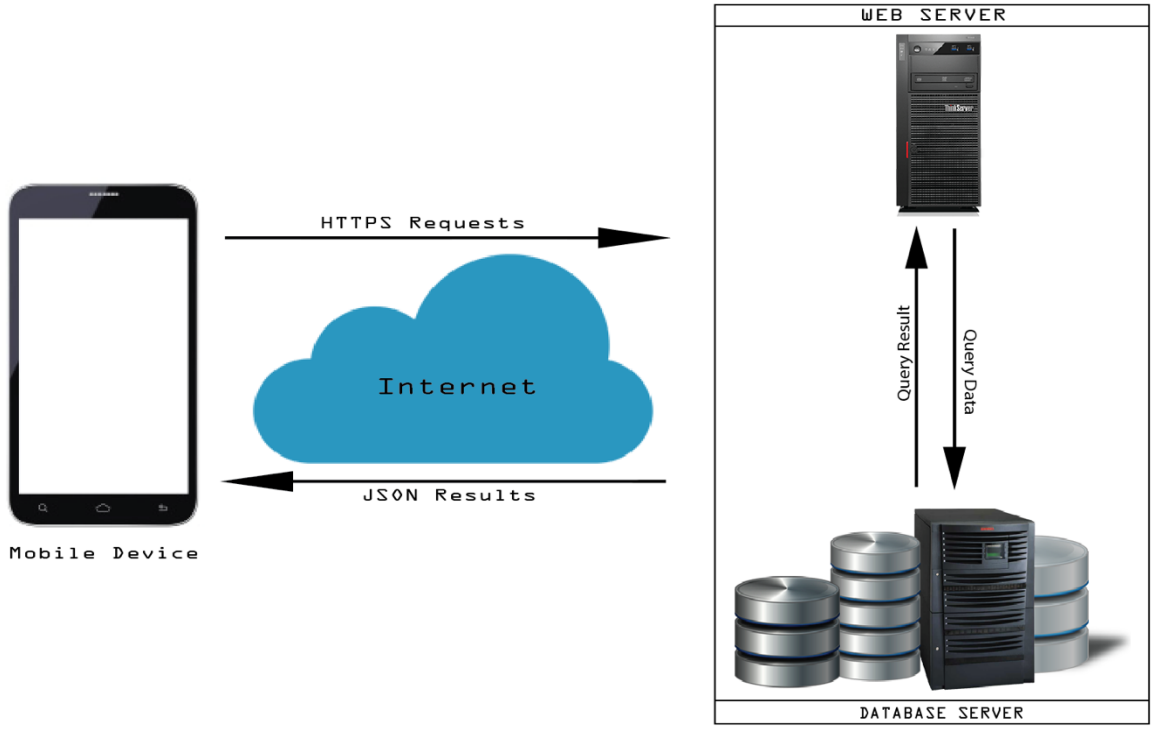


Figure 3.2: Mobile server interaction

Furthermore, the database layer is also vital to the mobile application since it is dependent on it for data handling and updates. The mobile application communicates with a PHP script on the server that queries the database according to the requested information. Figure 3.2 is a diagram that illustrates how the mobile application interact with the server. The mobile application interacts with the server by sending an HTTPS URL request to the server to process the request and response with the desired results. HTTPS connection was

used to handle security since it encrypts the URL sent to the server so that hackers will not be able to interfere with requests.

### 3.1.2 Unique Identification

In identifying each user or patient in the system uniquely, the system employs unique identification architecture. Each user has a unique identification in the system's database that differentiates one user from the other. This is highly important because the users of the web application and the mobile application are two different set of users, hence there is the need to be able to assign each user a unique identification that will be used in updating and tracking the activities of the user.

The patients on the system are also identified uniquely across the platform. When patient's medication details are updated by a pharmacist on the web application, only the patient will be able to get the updated medication since it was effected with the patient's unique identification on the system. It is highly impossible for two users of the application to have the same identification number across the platform.

The unique identification of a user is generated randomly by using the PHP function rand() with a range of numbers from one thousand to twenty thousand. This takes place on the server and inserted into the database in addition to the user's details such as full name, age, address and etc. The initials of the user are added to the randomly generated identification to increase its uniqueness.

### 3.2 Architectural Pattern

The Central~HealthMS will be developed based on the Model-View-Controller pattern. The structure of the system is based on three components that interact with each other as illustrated in figure 3.2. The model structure is a representation of the database. The model consists of the database architecture that the user interacts with indirectly. The model encompasses the data entities involved with the system. The views represent the interface which serves as an interpreter for the data in the model for the users of the system. The controller component serves as a channel connecting the model and the views. The controller updates the users view when there are changes in the model and updates the model when the user sends a command.

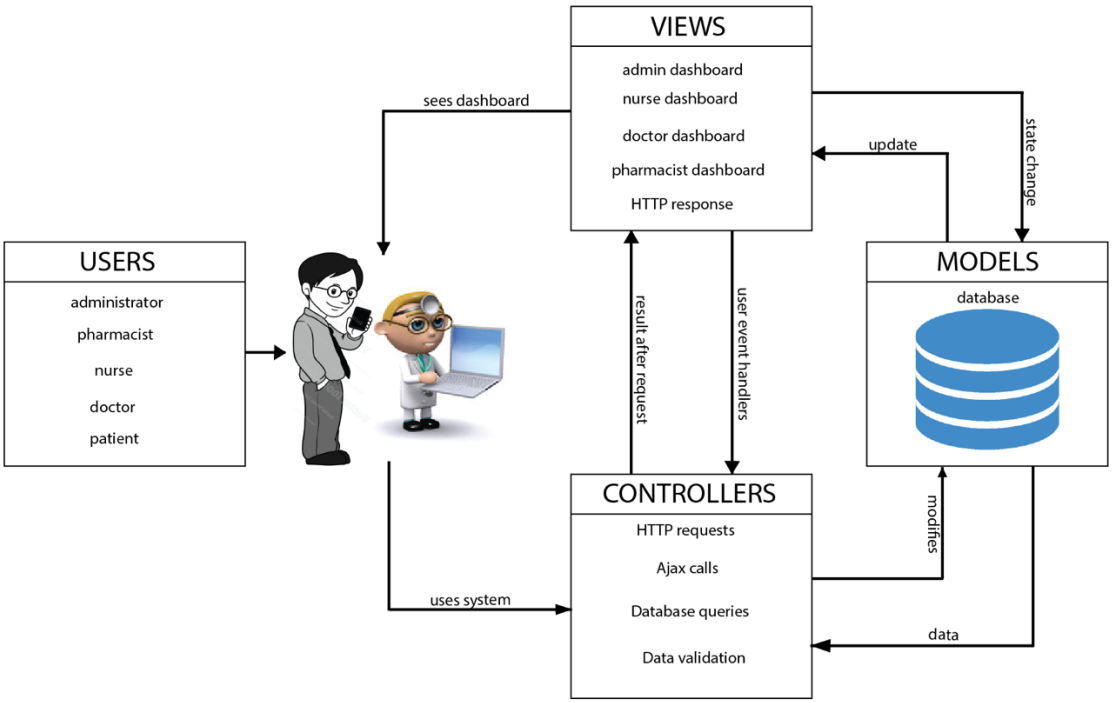


Figure 3.3: System design pattern

The reason MVC was chosen as the architectural design pattern for the Central~HealthMS system was due to its numerous advantages over other patterns. The

pattern will help to represent the data in the model in multiple ways. Central~HealthMS system is a complicated system hence MVC will make updating the system easy. This is because when changes are made to the data entities in the model, these changes do not have any effect on how the views are displayed to the system users. This makes the components in the Central~HealthMS system independent from each other. The use of this pattern for the design of the software is also due to scalability of the system - Since it is advisable for future updates and additions of new functionalities at minimum cost. Since Central~HealthMS system consists of various components, the design pattern will help in faster development and also dividing the components into smaller modules to work on. This will make it easy for each component in the system to be easily debugged.

### 3.3 Model Architecture

There are numerous stakeholders that make use of the Central~HealthMS and have various functions that they perform on the system by interacting with it. Since each user has unique function that they perform on the Central~HealthMS, each user is represented by a model in the system.

All the other users are represented as models in the system with each having a custom class defining the roles and privileges of that particular user. All the model classes extend to the adb class which is responsible for handling the communication between the client-side and the database server. They also implement interfaces that define the function that are to be implemented.

Figure 3.3 is an illustration of the patient model in the system. The patient model has functions and privileges in the system that other users do not have. The patient is also restricted and cannot perform roles of other users in the system. The patient model is

represented using a class and implements the patient interface class that has the defined functions of the patient model. The patient class also extends the adb class that is used to establish a connection to the database server so the patient can interact with the server and query data.

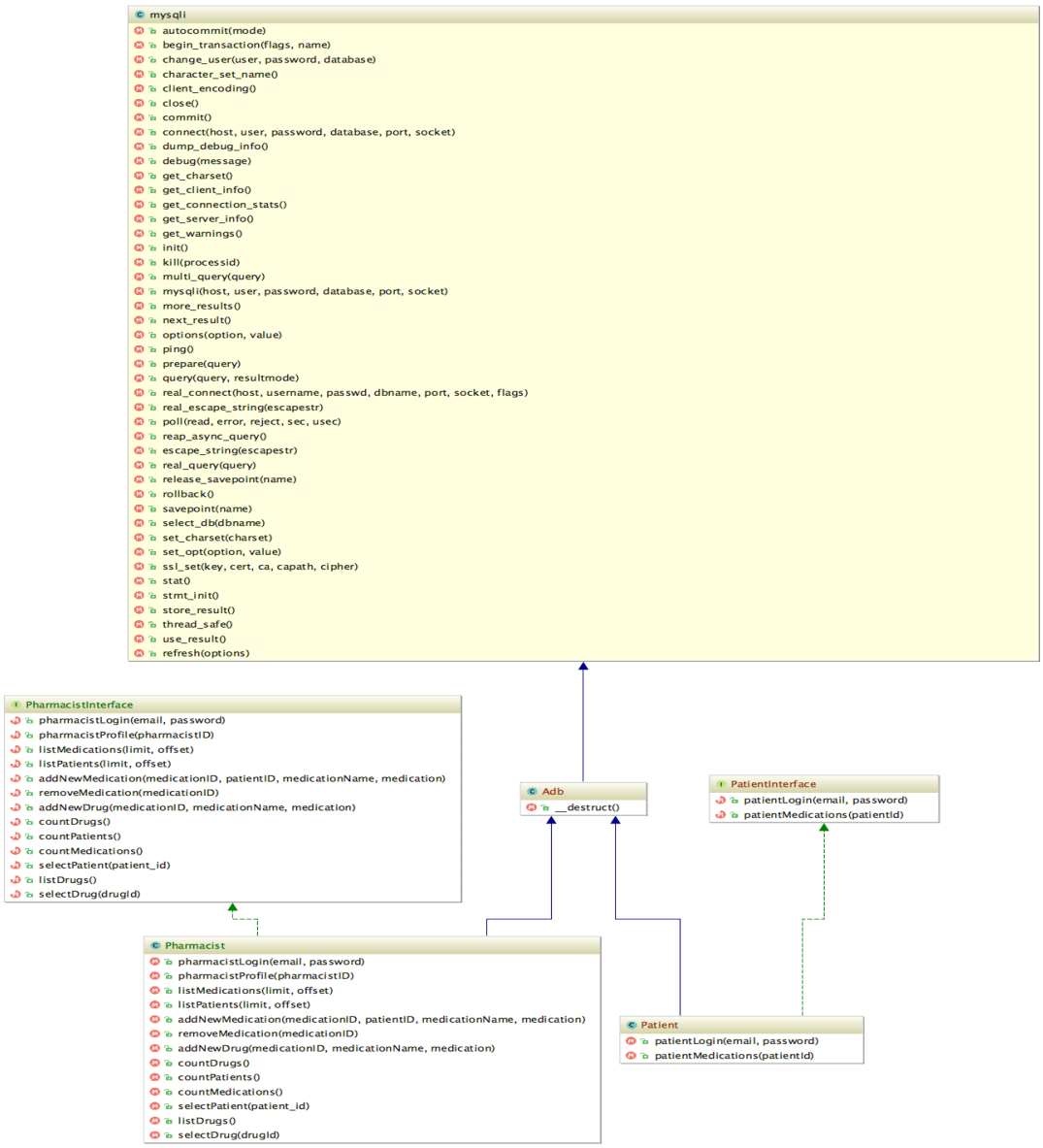


Figure 3.4: Class diagrams

### 3.4 Database Architecture

The Central~HealthMS is a centralized management system hence the model and structure of the database is vital to the implementation of the system. The database is the data store for the system and all the information about the system users are stored there. Choosing a particular database for this system was challenging. There were options to go for graph database such as Neo4j and other competitive relational database like PostgreSQL. MySQL database server was chosen due to the learning curve, how fast it processed requests and the security involved.

The components and models of the Central~HealthMS mostly interact with one another hence it was necessary to be able to represent that interaction in the database. Each table in the database has a primary key that is used to uniquely identify each record in the table. Figure 3.4 shows the modeled database with tables and their relations and attributes. A table called hms local patient registration that has a column called local patient identification is the primary key in that particular table because each patient is different from another patient in the table. Each tuple in that table must be uniquely identified because they are all independent of one another.

Furthermore, there was an issue of data redundancy when modeling the database for the system. This resulted in having more tables than needed in the database which could have also affected the performance when several tables were queried for data. The issue was solved by the use of foreign key in tables that have an attribute in other tables. This reduced the number of tables used to model the database and the redundancy problem.

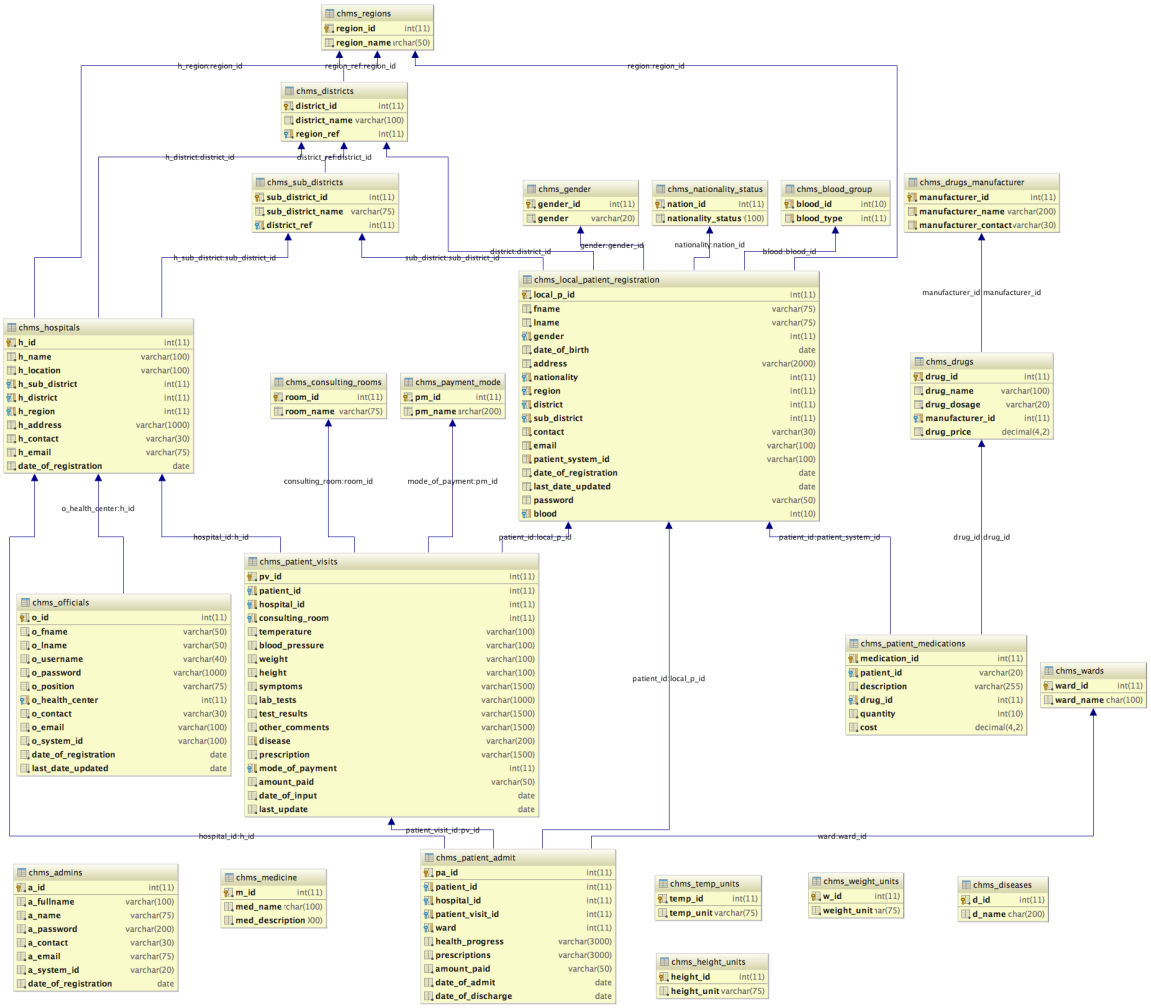


Figure 3.5: Database architecture model

The modeling of the database followed the normalization standards of data modeling. After the first normal form model of the systems database, which was to divide the data into logical units of tables, the second normal form was adopted to separate data that was partly dependent on the primary key in a table and created a new table for that data. Finally, the third normal form was used to remove data that was not dependent on the primary key in a particular table.

# Chapter 4: Implementation

This chapter focuses on the tools, components and libraries used for the implementation of Central~HealthMS. The chapter also provides screenshots of the various user interfaces of the system in different states.

## 4.1 Process Model

Before implementation of the software, it was necessary to know the user requirement and the design of the system. The waterfall software process model made it possible to be able to design the system before implementation. By following this model, the requirement of the system users was gathered and validated to know the implementable ones from the others. This was a major step towards the implementation of Central~HealthMS. After both functional and non-functional requirement of the target users were developed, the system was then designed to suit the requirement.

## 4.2 Tools and Technology

This section elaborates the tools, languages and libraries that were used during the implementation of the Central~HealthMS.

Table 4.1: Implementation tools

| Tools | Description |
|---|---|
| **PHP 7.0.4** | Hypertext Pre-processor (PHP) is a server-side scripting language. Since Central~HealthMS will be hosted on the server PHP makes it |

| | |
|---|---|
| | possible for clients to communicate with the application over the internet. |
| **MySQL** | MySQL is a relational database server that was used to model the database of the system. |
| **HTML5** | For the client to be able to use the application, Hyper Text MarkUp Language (HTML5) was used to design the Graphical User Interface for the system so that users of the systems will be able to use the system with ease. |
| **CSS3** | Cascading Style Sheets (CSS3) was used to design the user interfaces for the application. It was used to manipulate the HTML5 to produce a better looking Graphical User Interface. |
| **JavaScript** | JavaScript was used for animations and transitions in the application. |
| **Twig 1.24** | This is a PHP library template engine that was used with HTML5 to provide flexible templates for the application. |
| **Android** | This is the Operating System and language that the mobile version of the system was built with. It makes it possible to run the application on all android devices. |
| **PhpStorm** | This is the IDE that was used to develop the application. This IDE was used because it makes development easy and fast. |
| **GitHub** | This is a versioning control repository where the project was stored and used to track development progress during implementation. |
| **PHPUnit** | This is a testing framework for PHP. The framework was used to carry out test cases for methods and objects for the software. |

### 4.3 Framework and Template

During the implementation of the project, certain difficulties were encountered that led to a twist in the development of the project. The initial PHP framework used to start the development of the project was Laravel 5.2. Laravel comes with numerous libraries that can be downloaded and added to the project through composer. The downloading and setup of Laravel was challenging and took 2 weeks to get a good working project. Since the project was being developed using PhpStorm IDE, integrating Laravel needed more libraries to be downloaded and installed. When the Laravel project was finally setup, figuring out its folder structure was also another obstacle. The various Laravel versions have different folder structure hence getting tutorials to start developing with the framework was another challenge. The framework indeed did handle the database architecture of the project however, but that was not enough. The challenges with the PHP framework led to delay in the development of the project because much time was used to try and figure out ways and means to get it working.

A decision was made to discard the use of Laravel since it made the progress of the project slow due to its learning curve. Twig, a PHP template, was then used for the implementation of the project. Composer was also included to download certain dependencies the project will need for development. The use of twig template made development of the project fast and it was easy to learn and use.

### 4.4 Platform

This section of the chapter highlights the platform that Central~HealthMS will be able to operate in without challenges. The constraints of the application on various platforms are also discussed for the users to have an idea of the applications operating platforms.

### 4.4.1 Web Version

The web version of the application can be accessed by users through Google Chrome, Safari and Mozilla Firefox browsers. The browsers that the application runs on are not operating system specific. Hence, the application will run on all browsers on any operating system - whether Linux, Windows or Mac OS. There are also few constraints for the web application. It is not advisable to run it on Windows Internet Explorer because it has not been tested for that particular browser. Hence when run using internet explorer, it will not give the user the best of performance needed.

### 4.4.2 Mobile Version

The mobile version of the application will run on native Android Operating System. Hence any smart phone running the Android OS will be able to install the application and use with ease. The only thing needed after installation of the application is to provide login details and after that all required access will be granted. There is also a constraint on the mobile version of the application as there is for the web. The application however does not support android versions below Android 4.1 Jelly Bean. Hence users using any version of android below Jelly Bean will not be able to install the application.

```
/**
 * Function doInBackground
 *
 * This function is responsible for
 * making http requests to the server
 * and fetching response of the request
 * sent.
 *
 * @param urls The url to be sent
 * @return Returning the response
 */
@Override
protected String doInBackground(String... urls) {
```

```
    String response = "";
    HttpsURLConnection urlConnection = null;
    for (String url1 : urls) {
        try {
            URL url = new URL(url1);
            urlConnection = (HttpsURLConnection)
url.openConnection();
            InputStream in = new
BufferedInputStream(urlConnection.getInputStream());
            BufferedReader buffer = new BufferedReader(
                    new InputStreamReader(in));
            String s = "";
            while ((s = buffer.readLine()) != null) {
                System.out.println(s);
                response += s;
            }
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            assert urlConnection != null;
            urlConnection.disconnect();
        }
    }
    return response;
}
```

Figure 4.1: Source code to handle mobile request

Security is a relevant issue in the system hence necessary measures were taken to prevent any future attacks on the system. Figure 4.1 is the source code that the mobile application uses to communicate with the web server. It sends encrypted requests to server by the use of HTTPS. This helps prevent unauthorized disclosure of patient information to a passive hacker trying to eavesdropping on requests made by the application.

### 4.5 Web Interface

This section of the chapter illustrates the numerous user interfaces for the application. A detailed description of each interface is provided. Source code snippets of relevant implementations are also included in this section.

Since PHP 7 was used as the interpreter during the project implementation, prepare statements were used to query the database. This was used because the normal way of querying databases was deprecated in the latest version of PHP. The use of prepared statement in querying data from the database adds security to the entire project. Prepared statement was primarily created for security. Figure 4.2 is a function that queries a patient's login session and it does this by using prepared statement. The use of prepared statement prevents SQL injections by hackers. This is because the parameters of the query are sent to the database as different command.

```
/**
     * Function to allow patients to login
     * to the mobile application by providing
     * their email address and password
     * provided by the system administrator
     *
     * @param $email
     * @param $password
     * @return mixed
     */
    public function patientLogin($email, $password)
    {
        $patientLoginQuery = /** @lang MySQL */
            <<<PATIENTLOGINQUERY
                SELECT
                  `chms_local_patient_registration`.`email`,
                  `chms_local_patient_registration`.`local_p_id`,
                  `chms_local_patient_registration`.`fname`,
                  `chms_local_patient_registration`.`lname`
                FROM `chms_local_patient_registration`
        WHERE `chms_local_patient_registration`.`email` = ?
       AND `chms_local_patient_registration`.`password` = ?
                LIMIT 1
PATIENTLOGINQUERY;
        if ($statement = $this->prepare($patientLoginQuery)) {
            $statement->bind_param("ss", $email, $password);
            $statement->execute();
            return $statement->get_result();
        }
        $statement->close();
        return false;
    }
```

Figure 4.2: Prepared statement usage

One of the relevant features of the system is the login session of the web interface. This must be highly secured to prevent unauthorized access to the application and user data. For such reason the login credentials of the authorized users must be handled carefully. This security threat is tackled by hashing the user password before entering them into the database. All the user passwords in the database are encrypted to prevent database crawlers from accessing user accounts.

```
class PassHash
{
    // blowfish
    private static $algo = '$2a';
    // cost parameter
    private static $cost = '$10';
    // mainly for internal use
    public static function unique_salt()
    {
        return substr(sha1(mt_rand()), 100, 2000);
    }
    // this will be used to generate a hash
    public static function hash($password)
    {
        return crypt($password, self::$algo .
            self::$cost .
            '$' . self::unique_salt());
    }
    // this will be used to compare a password against a hash
    public static function check_password($hash, $password)
    {
        $full_salt = substr($hash, 0, 29);
        $new_hash = crypt($password, $full_salt);
        return ($hash == $new_hash);
    }
}
```

Figure 4.3: Login encryption source code

The passwords of the system users are not only hashed, but salt is also added to make it more secure. Figure 4.3 shows a class that handles the encryption and verification of user

passwords before being saved to the database. Before the actual user password is stored, random numbers from one hundred to two thousand are generated. Likelihood of having the same generated numbers after 50 tries is 1percent. Before the generated numbers are added to the actual user password as salt, the number is also encrypted using the SHA1 algorithm.

Finally, the generated encrypted number is added to the actual user password and encrypted again by using the PHP function 'crypt()' which take two parameters thus the password and the salt to be added. The user password is now encrypted and safe to be stored in the database.

- **Login Interface:** This is the first page that is used for security clearance for the application. When a user accesses the application, the user is provided with the login page in order to provide the necessary authentication details to be allowed access. Illustrated below is figure 4.4 the login interface.

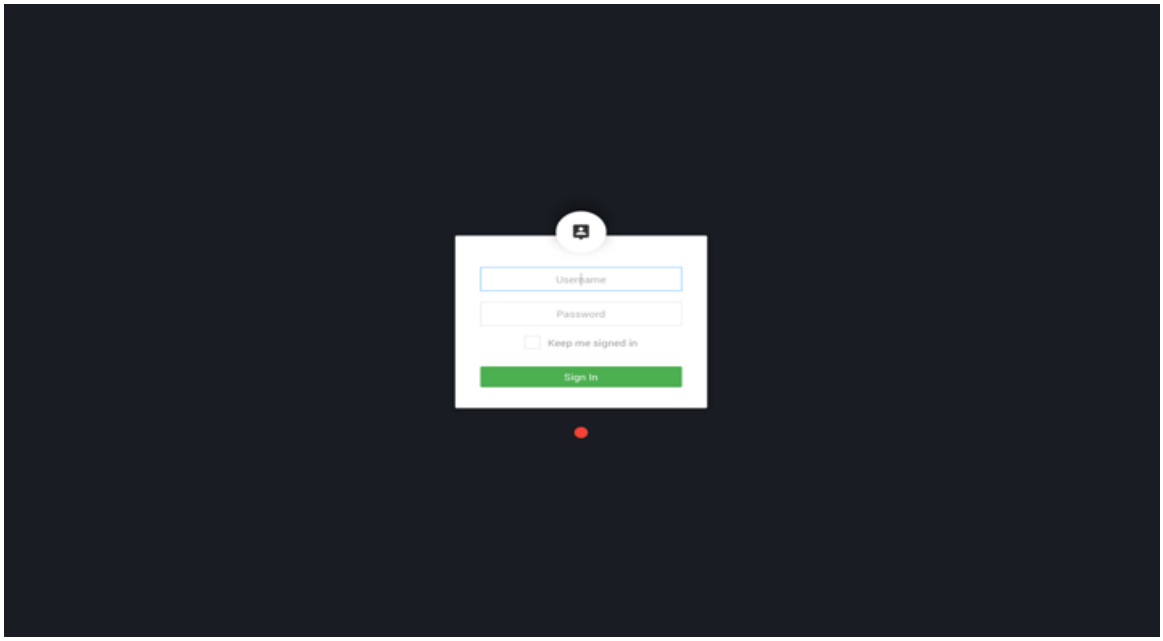


Figure 4.4: Login interface

- **Dashboard Interface:** This is the landing page for the user after login is successful. Statistics of the drugs and patients in the system database are displayed for the user to easily make meaning of them. It also has the time of the day displayed on the header of the page. Statistics such as the number of patients that have been served and the total amount of money earned.

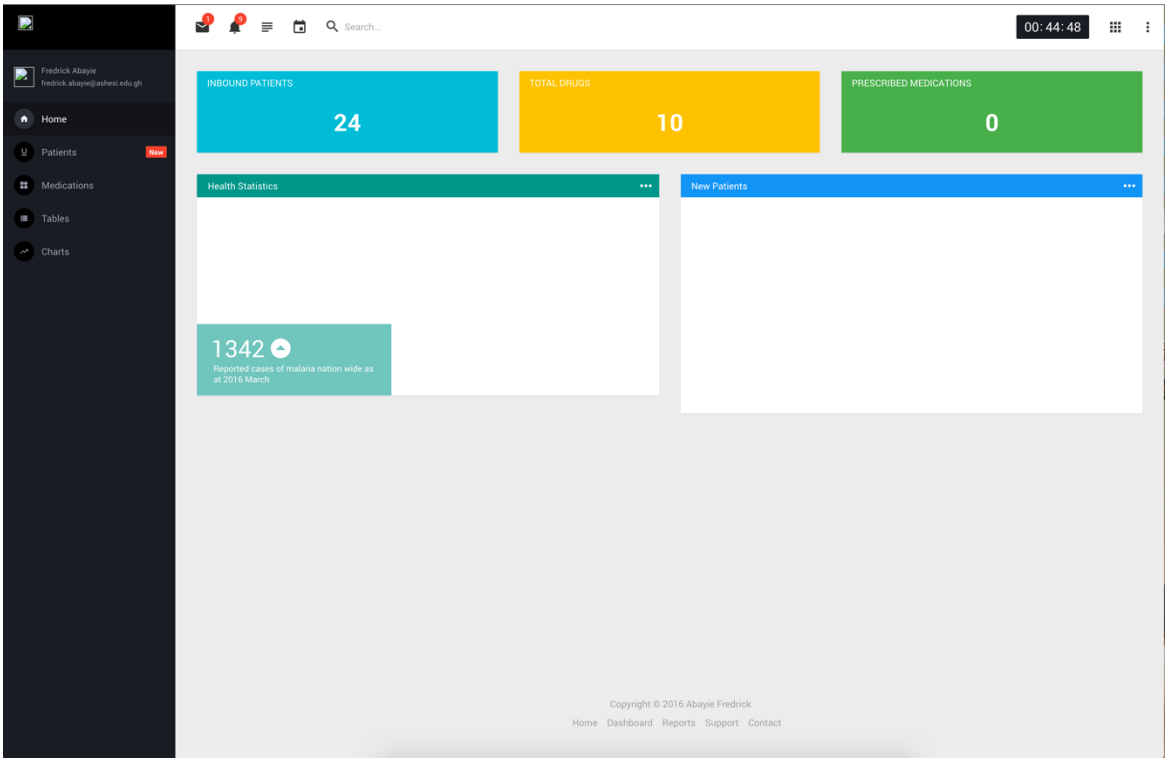


Figure 4.5: Dashboard Interface

- **View Patients Interface:** The registered patients in the systems database can be viewed and managed at the patient interface. This interface enables the searching of patients on the system, sorting of patient according to name, email or identification number. The user can also view more information about a patient by click on the

view button next to a patient email in the table row of the action column. The list of patients can also be paginated to view more patients stored in the database.

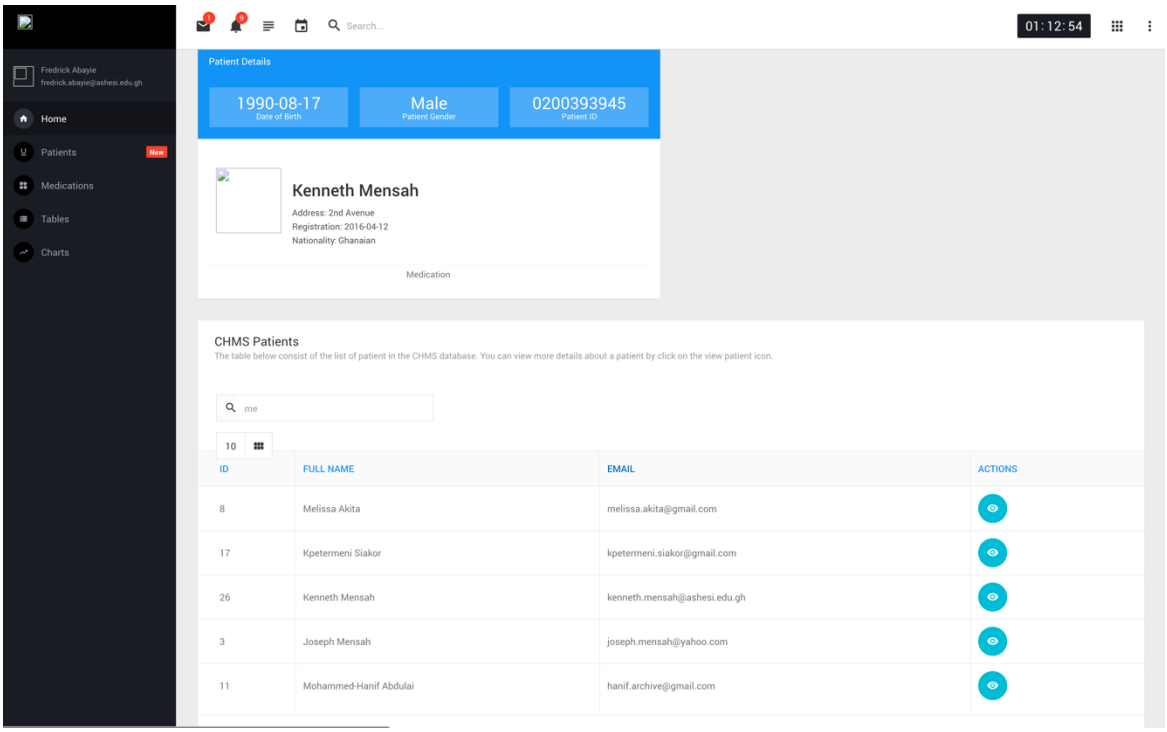


Figure 4.6: View patients interface

- **Prescribe Medications Interface:** The medication interface is the page to make prescriptions for patients. The interface has a table of drugs in the database displayed and the user can search for a drug by name and also sort the table of drugs. The user will also be able to paginate the table to view all the drugs in the database. Twenty drugs are displayed in the table at a time hence each time the user paginates they will be twenty records. The user can prescribe medications to the patient by click on the add button in the action column in the table. Below is figure 4.7 that illustrates the interface for patient medication prescription.

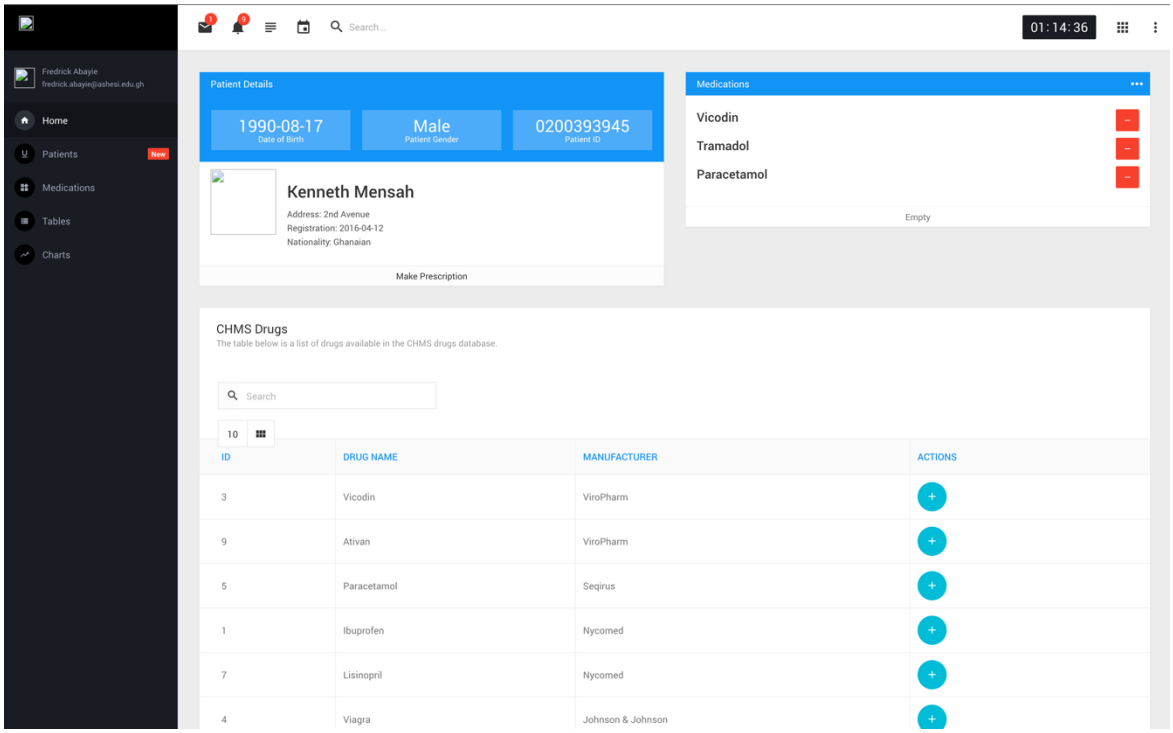


Figure 4.7: Prescribe medications interface

## 4.6 Mobile Interface

Below is the user interface available for the mobile version of the application. The mobile version is mostly for the patients only.

The designing of the user interface for the mobile application took seven iterations to get the best interface that was easy to navigation by the users and also implementable. Before the final interface for the mobile application was decided on, various sketches were made and shown to users as paper prototypes. The prototypes were made with a requirement hence it greatly influenced the design of the mobile interface.

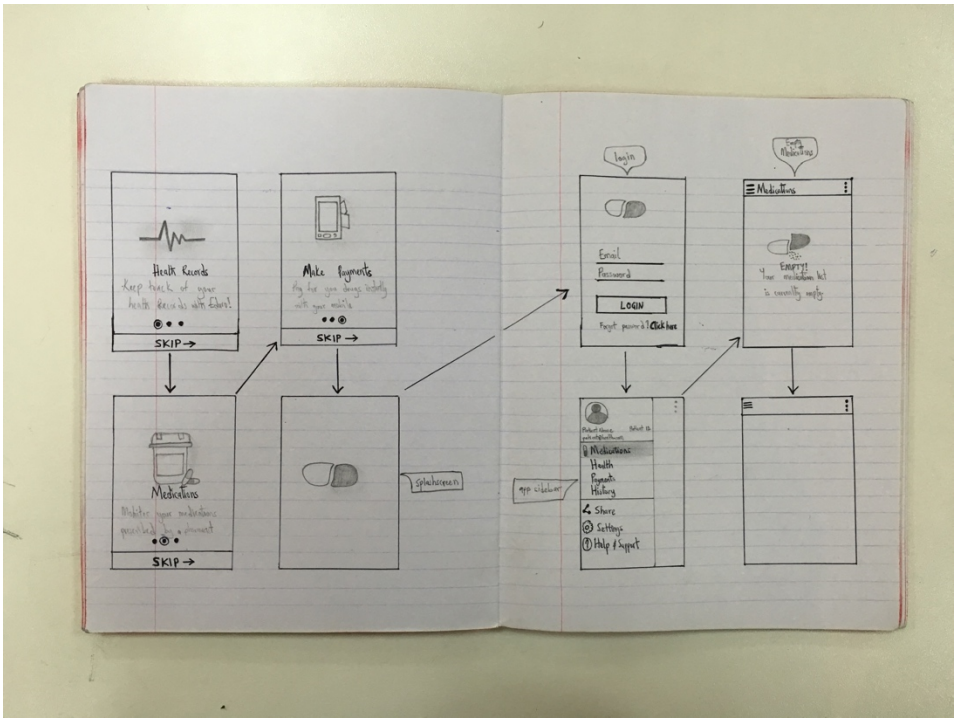


Figure 4.8: User interface mockup

Users gave constructive feedback after and before trying out the paper prototypes with a scenario that met user requirement. Illustrated above is figure 4.8 the final user interface mockup that was used in designing the user interface of the mobile application.

Management of patient sessions is vital aspect in the mobile application. This is suitable for usability of the application and also easily getting access to the user details. When the patient logs in to the application, the patient's details are fetched from the database and the session manager class as indicated in figure 4.5 stores the details to that it will be available the next time the user opens the application. The check login function runs each time the application is opened to make sure that the user is logged in if not the user is redirected to the login screen.

```
/**
 * Check Login method will check user Login status
 * If false it will redirect user to Login page
 * Else won't do anything
 */
```

```
public boolean checkLogin() {
    // Check Login status
    if (!this.isLoggedIn()) {
        // user is not logged in redirect him to Login Activity
        Intent i = new Intent(_context, Login.class);
        // Closing all the Activities
        i.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
        // Add new Flag to start new Activity
        i.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        // Staring Login Activity
        _context.startActivity(i);
        return true;
    }
    return false;
}
/**
 * Get stored session data
 */
public HashMap<String, String> getPatientDetail() {
    HashMap<String, String> patient = new HashMap<>();
    // patient email
    patient.put(KEY_EMAIL, pref.getString(KEY_EMAIL, null));
    // patient email id
    patient.put(KEY_PATIENTID, pref.getString(KEY_PATIENTID,
null));
    // patient name
    patient.put(KEY_PATIENTNAME, pref.getString(KEY_PATIENTNAME,
null));
    // return user
    return patient;
}
```

Figure 4.9: Session manager

The patient does not need to provide his/her login details each time the application is accessed. It can be argued that this also pose security threats to the data of a particular user in case another party gets hold of the device. But when it comes to security and usability, there are always challenges.

- **Screen On Boarding:** This interface provides first users of the application an overview of the mobile applications features and the functions that the users can perform on them. Figure 4.10 represents the screen on boarding interface.

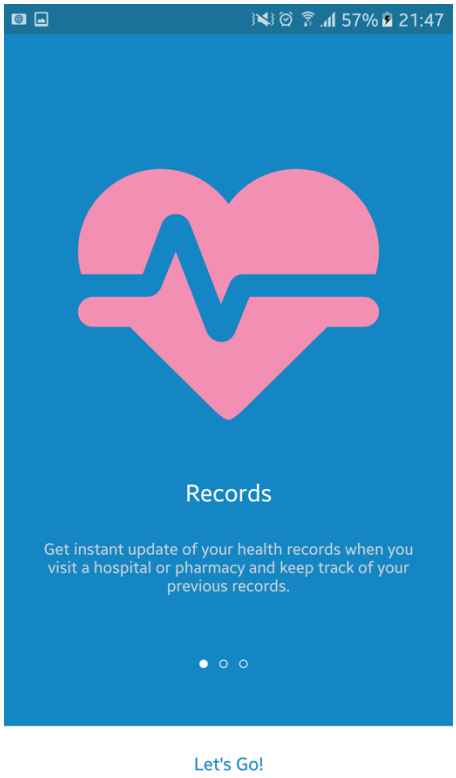


Figure 4.10: Screen on boarding

- **Login Screen:** For the mobile application, the patient will also have to provide login details to be allowed access through the application. The patient login will be a one-time authentication in order to improve efficiency of the application. Figure 4.6 is the interface for the login screen.

- **Settings Screen:** The patient can also access the settings of the mobile application to tweak how the application should behave. The user can turn off notifications and change the display name. Below is Figure 4.7 the interface for the settings page.

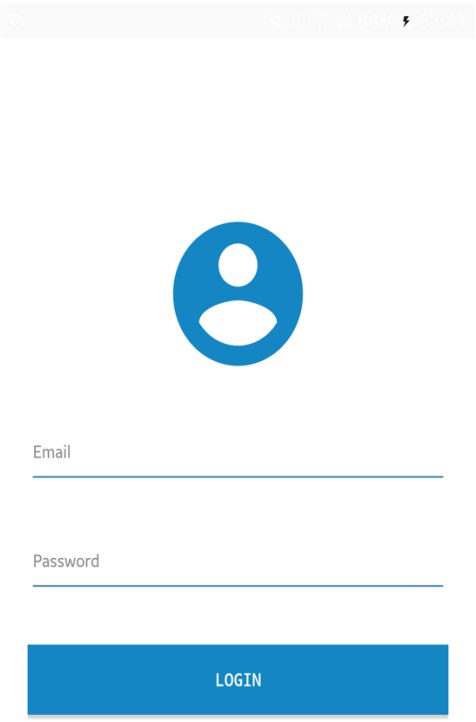


Figure 4.11: Mobile login interface

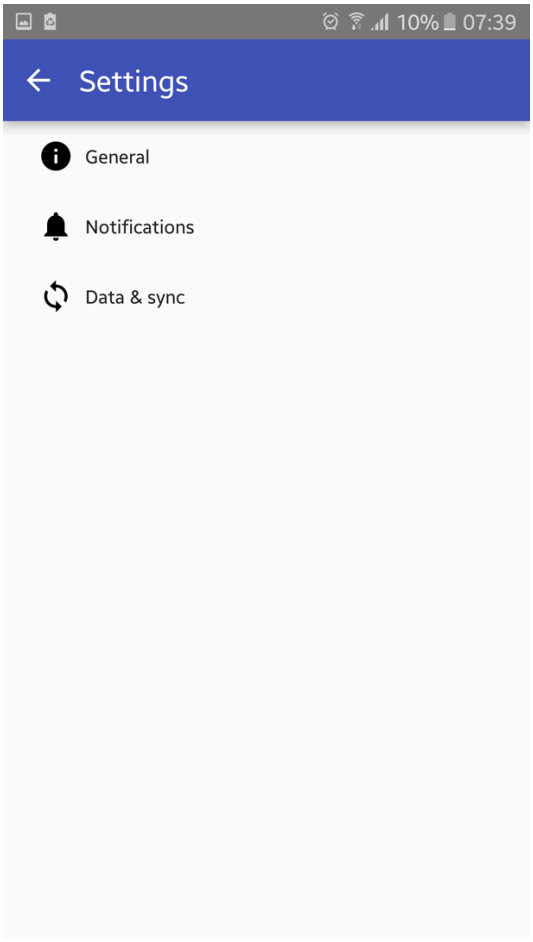


Figure 4.12: Mobile settings interface

- **Navigation Screen:** The navigation screen helps the patient to navigate to various parts of the application. This is a representation of menus in the application where a patient can switch tasks, or screen from this view. Figure 4.8 is the illustration for the navigation view for the application.

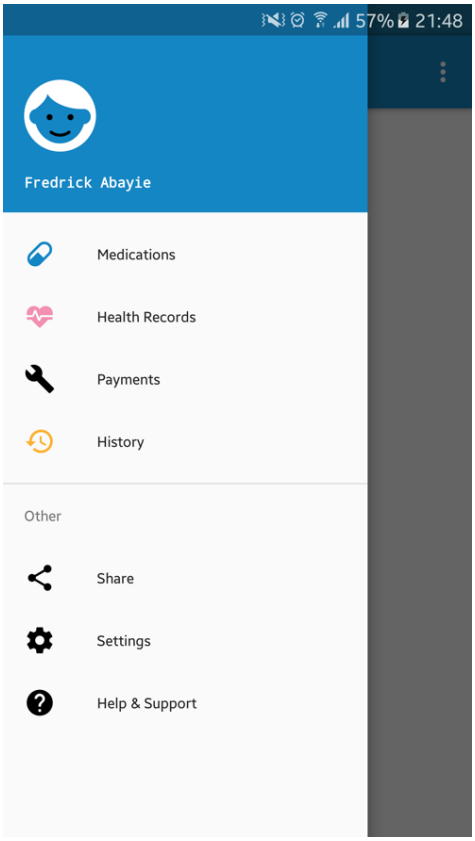


Figure 4.13: Navigation screen

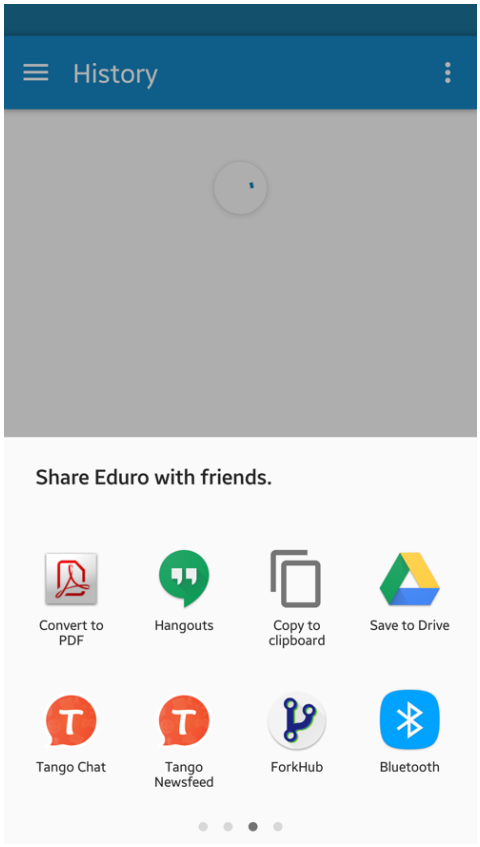


Figure 4.14: Share screen

- **Share Screen:** The patient gets to share the application with friends and family on any social media or by email. This will help promote the application to draw more user onto the application. Figure 4.9 is the illustration of the share screen interface.

# Chapter 5: Testing and Results

This section of the document highlights the techniques used for testing Central~HealthMS and the results that were obtained from each independent test. Since the software has stakeholders, and is intended to be used by them, it is necessary to go through various test cases both during and after development to make sure the software meets its requirements. Below are the various tests and results that were used on the system:

## 5.1 Unit Testing

During the development phase of the software, several objects and components were tested to make sure that they function as they were supposed to. This testing was primarily to detect bugs in the software during development and fix them before deployment. To carry out the development testing successfully PHPUnit testing framework was used.

In testing the numerous objects and methods in the software, unit testing was conducted sequentially. This was used to verify that all the operations associated with the particular objects accepted and produced the required output.

### 5.1.1 Database Connection Testing

The test was carried out to verify if the system makes a connection to the database server successfully. Since the system always makes requests to and receives response from the database server it is vital to test the function that establishes that link. Figure 4.1 shows the source code for the AdbTest class that was used for assessing the database connection to the server. The testConnection function needed an instance of the Adb class that extends to the mysqli class and establishes a database connection as indicated in figure 4.1. The

setUp function is the constructor of the AdbTest class and is called each time the class is run. Also the tearDown function is the destructor and evaluates the variable adb to null to terminate the connection instance. The testConnection function asserts the connection instance to true if a connection is established to the database server.

```
/**
 * Created by PhpStorm.
 * User: fredrickabayie
 * Date: 03/04/2016
 * Time: 19:51
 /
require '../vendor/autoload.php';
require '../models/Adb.php';
/*
 * Class AdbTest
 *
 * The class that tests the Adb class to make
 * sure all the function works. This class tests
 * the connection the class makes to the mysql
 * server.
 /
class AdbTest extends PHPUnit_Framework_TestCase
{
    // Declaration of local variable
    private $adb;
    /*
     * Function 43eard
     *
     * Constructor for AdbTest
     /
    public function setUp()
    {
        $this->adb = new Adb();
    }
    /*
     * Function 43eardown
     *
     * Destructor for AdbTest
     /
    public function tearDown()
    {
        $this->adb = null;
    }
    /*
     * Function testConnection
     *
     * The function to test the connection to the
     * database server.
     */
    public function testConnection()
    {
```

```
        $connection = $this->adb;
        $this->assertTrue($connection == true);
    }
}
```

Figure 5.1: AdbTest class for the database connection

After testing the database connection with the AdbTest class as indicated in figure 4.1, the test result was successful. The connection returned by the adb instance variable was true and this represents a successful connection to the database server. The test result from the AdbTest class is shown in figure 4.2. The results show the time it took to establish a connection to the database server and the amount of memory consumed in establishing the connection. It took 63 milliseconds to establish a connection to the database server which is fast enough for the system.

```
/usr/local/bin/phpunit --no-configuration AdbTest
/Applications/XAMPP/htdocs/Capstone/Project/Mobile/Server/tests/Ad
bTest.php –teamcity

Testing started at 01:07 ...

PHPUnit 5.0.10 by Sebastian Bergmann and contributors.

Time: 63 ms, Memory: 8.00Mb

OK (1 test, 1 assertion)

Process finished with exit code 0
```

Figure 5.2: AdbTest results for database connection

### 5.1.2 Patient Login Testing

Another vital unit test that was carried out was the login function of a patient using the mobile application. A PatientTest class was created to test all the functions available in the patient class. The function to be tested was the patientLogin. Figure 4.3 is a representative of the test class for the patient model. The setUp function, which is the constructor, is used to create a new instance of the patient class to work with. The testPatientLogin function is the function responsible for checking that a successful login detail has been provided. To get a successful test case, the patient's email address and password were provided to the function.

```
/**
 * Created by PhpStorm.
 * User: fredrickabayie
 * Date: 03/04/2016
 * Time: 21:42
 *
 */
require '../vendor/autoload.php';
require '../models/Patient.php';
/**
 * Class PatientTest
 *
 * The class that tests the patient class to make
 * sure all the function works. This class tests
 * the connection the class makes to the mysql
 * server.
 */
class PatientTest extends PHPUnit_Framework_TestCase
{
    // Declaration of local variable
    private $patient = null;
    /**
     * Function setUp
     *
     * Constructor for the PatientTest class
     */
    public function setUp()
    {
        $this->patient = new Patient();
    }
    /**
```

```
     * Function tearDown
     *
     * Destructor for the PatientTest class
     */
    public function tearDown()
    {
        $this->patient = null;
    }
    /**
     * Function testPatientLogin
     *
     * The function to test the patient
     * login function in the patient class.
     */
    public function testPatientLogin()
    {
        $patientLogin = $this->patient-
>patientLogin('fredrick.abayie@ashesi.edu.gh', '');
        $this->assertEquals(1, count($patientLogin));
    }
}
```

Figure 5.3: PatientTest class for login

The patient login function test was successful. This is because it produced the necessary and expected output from the test which was 1. This is because when the patient logs in, the patients detail is returned from the database in one row. Hence the result of the patient login function was compared to 1 to detect if there was a successful login or not. It also took 66milliseconds for the login test to be carried out and 8.00Mb of memory.

```
php /usr/local/bin/phpunit --no-configuration PatientTest
/Applications/XAMPP/htdocs/Capstone/Project/Mobile/Server/tests/Pa
tientTest.php --teamcity
Testing started at 03:24 ...
PHPUnit 5.0.10 by Sebastian Bergmann and contributors.
Time: 66 ms, Memory: 8.00Mb
OK (1 test, 1 assertion)
Process finished with exit code 0
```

Figure 5.4: PatientTest result for login

## 5.2 Release Testing

Since Central~HealthMS was developed in releases it was essential to conduct a release test for each version that was intended for users. Explained in this section is the technique used for the release testing of the software.

### 5.2.1 Requirement Based Testing

For the release testing of the software, it was necessary to make a requirement based testing to verify that the defined requirements of the system have been met. This was done by having the various requirements and deriving set of tests for it. Viewing of patients and medications, prescribing patients their medications, patients viewing their medications through the mobile application are examples of requirement test that was carried out.

After all the requirement based test was conducted on the software, it resulted that the software passed all the test cases. However, this indicates that the software has been properly implemented to satisfy the defined requirements.

## 5.3 User Testing

The user testing was the most important part of the software testing. Since the software has stakeholders, it was necessary to make the software available to the target customers for general feedback on the use of the software.

### 5.3.1 Beta Testing

Central~HealthMS users were provided a release of the software for evaluation. The purpose of this test was to learn the challenges that may arise when the software is exposed to the working environment and the interaction problems.

Several feedbacks came through from the software users which helped to improve the features and functionality of the software. However, the testing proved to be a success, by helping improve the usability of the software.

## 5.4 Other Testing

Since Central~HealthMS is a web application, it tested on four popular browsers. They are Google Chrome, Mozilla Firefox, Safari and Internet Explorer. Even though the application worked well on the latest versions of the browsers, it was also tested on previous versions. After running several tests, it was realized that the application has issues rendering on previous versions of Internet Explorer and some versions of Safari.

This discovery led to optimizing the source code to allow the application to function well on these previous versions of browsers. Due to the tests, improvements have been made to allow the application render well on some previous versions of browsers.

# Chapter 6: Conclusion and Recommendation

To conclude, this application will solve the problems in the Ghanaian hospitals in relation to record keeping successfully. Hospital officials can now record patient details and prescribe medications to them with ease and the patient information is always updated. Since it is a centralized system, the same data for a particular patient is seen by different health officials. Also nurses can now quickly search for patient records and keep them updated when a patient visits the hospital. This will also help doctors keep track of a patient's health record and to be able to detect any abnormal changes in the health of a particular patient. This will also help doctors and pharmacist provide better diagnosis to patient.

Recommendations for future include allowing a patient view his/her medications locally thus by storing the medication details in phone's database. This local database should only be updated when the database on the server has been updated else the data should be read from the local database which will be SQLite.

# References

About GNUmed. (2013, January 20). Retrieved from http://wiki.gnumed.de/bin/view/Gnumed/AboutGNUmed

About OpenMRS. (2013). Retrieved from http://openmrs.org/about/

Adams, C. (2015). Proceedings of the 15th European Conference on eGovernment 2015*: ECEG 2015* (illustrated ed.). Portsmouth: Academic Conferences Limited.

Healthcare omron company info. (2013). Retrieved from http://www.omron-healthcare.com/eu/en/company-info

Improving Medication Safety in Community Pharmacy. (2009). Retrieved from https://www.ismp.org/communityRx/aroc/files/ISMP_AROC.pdf

Nyonmorworko, N. D. (2015, April 15). Reflections in the Mirror; Records keeping in Ghana's Health Sector. Retrieved from http://graphic.com.gh/features/opinion/41676-reflections-in-the-mirror-records-keeping-in-ghana-s-health-sector.html

Ohene-Bonsu, K. G. (2015). *Centralized Hospital Management System For Health Centers In Ghana.* (Unpublished undergraduate thesis). Ashesi University College, Computer Science, Berekuso.

Poikonen, J. (2009, March 16). 50 Successful Open Source Projects That Are Changing Medicine. Retrieved from http://pharmacyinformatics.blogspot.com/2009/03/50-successful-open-source-projects-that.html

Poissant, L., Pereira, J., Tamblyn, R., & Kwawasumi, Y. (2005). The impact of electronic health records on time efficiency of physicians and nurses. A systematic review. *Journal of the American Medical Informatics Association, 12*(5), 505-516.

Sakyi, K. E., Atinga, A. R., & Adzei, A. F. (2012). Managerial problems of hospitals under Ghana's National Health Insurance Scheme. *Clinical Governance: An International Journal, 17*(3), 178-190.

Sommerville, I. (2011). *Software Engineering* (9th Edition ed.). United States of America: Addison-Wesley.

Software Informer. (2015). Retrieved from http://omron-health-management-software.software.informer.com/

Wallen, J. (2009, October 20). Let your medical practice go open source with Gnumed. (gHacks Technology News). Retrieved from http://www.ghacks.net/2009/10/20/let-your-medical-practice-go-open-source-with-gnumed/