ASHESI

ASHESI UNIVERSITY COLLEGE

AN ENTERPRISE RESOURCE PLANING SYSTEM WITH INFORMATION SHARING AND DATA RE-USE CAPABILITIES FOR HOSPITALS

Applied Project

B.Sc. Computer Science

Ezekiel Mardillu Sebastine

2018

ASHESI UNIVERSITY COLLEGE

An Enterprise Resource Planning System with Information Sharing and Data Re-Use Capabilities for Hospitals

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

Ezekiel Mardillu Sebastine

April 2018

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: ………………………………………………………………………………………

Abstract

Technology is bringing people and activities closer each day. And a quick review of the technological trends for 2018 reveals that the top 10 trends focus on connecting to a big ‘tree’ with heavy reliance on data (Newman, 2018). In other words, the technological trends are such that each individual technology is developing an interface to link and interact with other technologies. We have seen already, that this is eventually giving rise to artificial intelligence, machine learning and more concepts on big data and data warehousing. This project focuses on applying the trend of data sharing and re-use on a software application. Specifically, integrating individual software modules, each serving a specific organizational function, into a single software. The target is hospitals. The application will allow a hospital to be able to share information across all its domains, maintain patient records nicely and save time and resources.

The application was tested both on and offsite. From the results, it can be concluded that the solution is relevant as well as usable. By implementing this solution and applying additional features such as machine learning and artificial intelligence, hospitals can save time and resources while giving better medical care to patients.

Acknowledgement

I thank almighty God, the giver of life.

I also want to thank Stephane Nwolley, my supervisor for the guidance and business inspiration.

Finally, I want to thank all my lecturers thus far for a job well done, may God bless you.

Contents

Table of Figures

List of Abbreviations

Abbreviation	**Full meaning**
AJAX	Asynchronous JavaScript and XML
CRUD	Create, Read, Update, Delete
CRM	Customer Relationship Management
CSS	Cascading Style Sheet
ERP	Enterprise Resource Planning System
HTML	Hypertext Markup Language
MySQL	My Structured Query Language
MVC	Model-View-Controller
PHP	Hypertext Preprocessor
XAMPP	Cross-Platform (X), Apache, MariaDB, PHP and Perl
XML	Extensive Markup Language

Chapter 1: Introduction

We have all learned things from other people who were willing to share information with us in the same way we share our information with others. Thus, we form a network of shared information. In this sense, human society can be perceived as being composed partly of new and reused pieces of information. This is powerful because most of the things we know come from the shared network of information. Another reason is because most of the existing literary works and data stores are a product of these shared information – most of which are simply reused. In the same way we possess information about the world, so do organizations (such as universities, hospitals etc.) which they store as database information. So, imagine the possibilities that would arise if this vast data could be shared and used more purposefully in the same way as humans. However, concurrently with new data coming in, such that while we are reusing existing data, the process of the reuse generates more data.

An Enterprise Resource Planning system (ERP) is a system that incorporates various subsystems into one complete system to streamline processes, information and data exchange across an entire organization. Subsystems here refer to functions such as employee management, inventory management, customer relationship management, patient management etc. Perhaps, a better description of ERP subsystems is the one put forward by Sayeed. According to Sayeed, "the central feature of all ERP systems is a shared database that supports multiple functions used by different business units. In practice, this means that employees in different departments, for example, accounting and sales, can rely on the same information for their specific needs" (Sayeed, 2016)

One reason for the wide use of ERP systems in organizations is the degree of synchronized reporting and automation that the systems offer. Since ERPs use a shared database, employees do not have to maintain separate databases and spreadsheets that must be manually merged to generate reports. Employees can simply pull relevant data from the central database. Some ERPs also include a portal or dashboard containing key indicators and reports to enable users to quickly understand the business' performance on those key metrics and indicators (Sayeed, 2016) (Netsuit, 2017). All these make ERP systems attractive and widely used in organizations

ERP systems may be attractive, but current systems do not harness their full potentials. For example, even though the current ERP systems have a shared database, the data is only accessible to the organization that runs the ERP system. What if we had an ERP system that allows cross-organizational data access and reuse. A research into medical informatics proves that this idea will especially be useful in the health sector (Jonesa, et al., 2017) (Safran, 2014). Currently, patients' details are retaken in every new hospital that they visit. With a cross-organizational ERP, when a patient's details are taken in one hospital, they could be accessed in another hospital, hence, there will be no need for a retake of the same details. In the long run, hospitals will be able to save resources especially on very expensive tests and examinations.

ERP systems that try to deal with this problem do exist. One of their shortcomings is that they try to support an organization's growth and expansion only, i.e. they provide a centralized database for all branches of the same organization that are distributed in different regions of the world.

The World Health Organization, for example, has developed a platform for patients' health data and record access and reuse. Hospitals can tap into this platform to access these data. However, even with this innovation, there is still untapped potential especially when you consider its use in the health sector. We can ask questions like, 'can we determine some basic information such as temperature of a patient from the phone that he/she is holding', 'can we use the vast amount of data at our disposal to form a disease map or map out areas prone to disease and possibly help to direct health aid', 'can we use this same data to predict disease outbreak before they happen'?

This project's focus is to build an ERP for the health sector. The project takes a novel approach based on original ERP concepts and adding the idea of data reuse, Internet of Things and artificial intelligence. In addition to traditional subsystem functions such as customer relations management, patient management among others, we include disease epidemic prediction functionality, internet of things and data reuse and information sharing across several health organizations on a single platform.

Chapter 2: Requirements Specification

2.1 Requirements Overview

The services provided by a system and their corresponding constraint are referred to as the requirements of the system. They describe the needs of the system and its users. This section describes the requirements for the Enterprise Relation Planning (ERP) Software.

Users of this ERP software include Doctors, Nurses, Accounting officials, Human Resource Managers, Inventory Managers etc. These automatically define the modules and functionalities that will be implemented in this system.

2.2 User requirements.

- The ERP shall store patients' information in a central database
- The ERP shall have interfaces for all the business units of hospitals
- The ERP shall generate monthly reports of all activities in the various business units
- The ERP shall make the entire hospital processes seamless and less burdensome.
- The ERP shall have a mobile application that will allow patients to book appointments, among other things.
- The mobile application shall have a calorie tracking functionality.

2.3 System requirements

The user requirements, highlighted above, were used to generate the system requirements. The system requirements are categorized into Functional and Non-functional requirements and discussed below.

2.4 Functional requirements

- The ERP will have a central database that is accessible by all modules of the ERP. All organization data will be stored in this database for seamless access.
- The system will have another database which will be accessible by all hospitals that use this system. This is where all patients' health information and medical record will be stored securely.
- The system will have interfaces for the following modules that exist in the hospital:
    - Human Resource Management: Helps to manage employee information
        - HR managers will be able to designate employees,
        - give job descriptions,
        - view employee skill matrix,
        - track attendance of employees
        - generate salaries and payments report.
    - Inventory Management: Keeps track of items in stock
        - this module will be integrated with purchase module
        - the module will allow location of items in the organization
        - it will generate report of current stock of items
    - Accounting and Financial management: Keeps track of transactions such as expenditures, account ledgers etc.
    - Customer relationship management: This module will manage all patients who have registered in a hospital.
    - Purchase: This module is concerned with procurement of items.

        - keep track of suppliers and vendors and match them with the products that they supply
        - generating and sending quotations
        - it will be integrated with inventory module

- Each user of the system will be uniquely identified by their unique staff IDs
- Users will have to be authorized to access a module before they can access the module.
- The system shall be able to predict diseases outbreak by analyzing database data.
- The system will have machine learning capabilities for predicting diseases outbreaks and generate demographic information of diseases based on data in database.
- The mobile application will leverage on Watson Studio[1] visual recognition engine to predict food from images taken using the mobile device camera.
- The mobile application will use Nutritionx Database[2] API to access food nutrient data.

2.5 Non-functional requirement

- Product requirement
    - Performance requirement: This system requires an active internet connection to work. Users must be connected to the internet and must be running a compatible web browser to use the system
    - Security Requirement: Since this is a hospital system that will be storing patient health record and information, security and confidentially is the topmost priority. The system shall be secured with encryptions and use of passwords

[1] Watson Studio is IBM's machine learning platform that allows users to seamlessly create, evaluate, and manage custom machine learning models. (IBM, n.d.)
[2] Nutritionx is a service that provides a large database of food nutrients which can be accessed via API calls

where necessary. When inactivity is detected for a period, the system will automatically log the user out.

- Usability requirement: The system shall have responsive, user friendly interfaces. After a fleeting period of guided interaction, a novice user should be able to use the system comfortably with few errors.
- Availability: The system shall be available 99.9% of the time.

- Organization requirement
  - Only staff of the hospital can have access to the system and its databases.
  - When a patient registers in one hospital, his/her details can only be accessed in another hospital if he/she authorizes so by scanning his/her card. However, his/her details can be accessed by the hospital that he/she registered in with or without the card.
  - The system shall maintain concurrency; many users will be able to access the system at the same time.
- External requirement
  - Regulatory requirement: The system must be approved by the Ministry of Health of any country where it will be used.
  - Ethical requirement: Since this is a hospital ERP system that will be storing patient health record and information, security and confidentially is the topmost priority. The system shall be secured with encryptions and use of passwords where necessary. When inactivity is detected for a period, the system will automatically log the user out. No confidential information of patients will be shared or sold to third parties

2.6 Use cases

Table 2.1 Use case scenarios

Use case: Send health data or record	**Primary actor:** **Patients**	**ID: MD1**	**Priority: High**
Interested stakeholders	Patients, Doctors		
Description	The use case describes the ability of the patients to send existing health data and health track record to doctors of the hospital they are registered in.		
Goal	The goal is for a patient to send his/her health data and track record to a doctor so that the hospital can have his/her existing health records in their database		
Success measurement	Success on sending and receiving of patient health data and track record to the doctor		
Precondition	The patient should have an account with MEDY; should also have gone through authentication and authorization. Also, the customer should have enough health data and/or track record to send		
Trigger	The patient clicks on the share record button		
Relationship	Include: Cancel share, share to multiple doctors		
Event flow	1. Patient logins to the application 2. Patient selects health data option 3. Patient selects share health data from the menu 4. The Patient selects a doctor and a hospital 5. The Patient enters their unique number for re-authentication 6. The system sends a message to confirm the decision to share		

	7. Patient confirms by clicking ok 8. The system sends the health information the doctor specified and gives a feedback message of a successful delivery to the patient.		
Use Case: **Doctors delete patients' records**	**Primary Actor:** Doctor	**ID**:MD2	**Priority**: High
Interested Stakeholders:	Doctors, patients		
Description:	The use case describes the ability of the doctor to delete records of the patients registered in their hospital.		
Goal:	Doctors can delete unwanted records of patients in their hospital records successfully.		
Success Measurement:	Doctors ability to delete patients' records permanent successfully		
Precondition:	The patient whose record is to be deleted should already have an existing account with MEDY. Also, the patient should have existing health records with the particular hospital		
Trigger:	The doctor clicks the delete record button		
Relationships:	Include: cancel, delete permanently		

Event Flow:	1. Doctor logins into the application 2. Doctor selects patients from the menu 3. The doctor selects / searches the patient whose record is to be deleted. Using patient's id or their name 4. The doctor clicks on delete record button 5. The doctor confirms permanent record deletion Or cancels the deletion 6. If deletion, the record is deleted successfully 7. There is a pop-up message notifying the doctor of a successful record deletion

Use case: Register patients	**Primary actor**: Doctor	**ID**: MD3	**Priority**: High
Interested stakeholders	Patients, Doctors		
Description	The use case describes the ability of the doctors to register new patients unto the MEDY platform. Doctors can do this from their dashboard of MEDY.		
Goal	The goal is for a patient to be registered unto the MEDY platform by doctors after taking their health data. After registration, the patient receives a username and a password to enable them access MEDY.		
Success measurement	Success on registering a patient. Patients receives a username and a password		
Precondition	The patient should NOT have an account with MEDY. Also, the doctor must have taken all the relevant data from the patients, like height, weight, age, BP etc.		

Trigger	The doctor clicks on the add patient button on the dashboard
Relationship	Include: Cancel button, Update patient data
Event flow	1. Doctor logins to the application 2. Doctor selects add patient from the menu 3. The Doctor completes the registration form displayed by taking all measurements and checks from the patients 4. The doctor clicks submit 5. The system adds the patient to the database with the information provided 6. The system gives a feedback message of addition success 7. The system generates a username and password for the patient which they will use to access MEDY 8. The system generates a unique number for the patients which will be used to access the data anywhere else

Use Case: Doctors view patients records	**Primary Actor**: Doctor	**ID**:MD4	**Priority**: High
Interested Stakeholders:	Doctors		
Description:	The use case describes the ability of the doctor to view records of the patients registered in their hospital.		
Goal:	Doctors can view all records of the patients or one patient record in their hospital database		

Success Measurement:	Doctor's ability to view all the patients' records successfully. Also, ability of the doctor to view one record or a group of records as per their requests. They can also search the record using patient's ID or name or any other attribute		
Precondition:	The patient whose record is to be viewed should already have an existing account with MEDY The hospital has to have an account with MEDY		
Trigger:	The doctor clicks the view all record button or searches a record on the search bar		
Relationships:	Include: cancel, print records, sort records		
Event Flow:	1. Doctor logins into the application 2. Doctor selects patient from the menu 3. The doctor clicks on view all record button or searches the id / ids of the patients to view their records 4. All records of the registered patients are displayed, or the records of corresponding search are displayed		
Use Case: Patients delete their records	**Primary Actor:** Patient	**ID**:MD5	**Priority**: High
Interested Stakeholders:	Patients		

Description:	The use case describes the ability of patients to delete their records from their corresponding hospitals
Goal:	Patients can delete their records permanently from MEDY
Success Measurement:	Patients' ability to delete their records successfully. This will also delete their records from the doctors' or hospital's side.
Precondition:	The patient whose record is to be deleted should already have an existing account with MEDY
Trigger:	The patient clicks the delete record button
Relationships:	Include: cancel, delete permanent,
Event Flow:	1. Patient logins into their account 2. Patient selects my profile menu 3. The patient selects the delete account option 4. The patient confirms permanent account deletion Or cancels the deletion 5. If deletion, the account is deleted successfully 6. There is a pop-up message notifying the patient of a successful account deletion 7. Patient seizes to access MEDY

Use case: update patient data	**Primary actor**: Doctor	**ID**: MD6	**Priority**: High
Interested stakeholders	Patients, Doctors		

Description	The use case describes the ability of doctors to update the record of patients. Doctors can do this from their dashboard of MEDY.
Goal	The goal is for a Doctors to be able to update the data of patients which might have changed with time such as age, height, weight etc.
Success measurement	Success on updating a patient record
Precondition	The patient should have an account with MEDY; The doctor must have taken all the relevant data from the patients, like height, weight, age, BP etc. The doctor must have been authenticated
Trigger	The doctor clicks on the update patient button
Relationship	Include: Cancel
Event flow	1. Doctor logins to the application 2. Doctor selects update patient from the menu 3. The doctor enters the unique ID of the patient 4. The Doctor completes the update form displayed by taking all measurements and checks from the patients 5. The doctor clicks submit 6. The system updates the patient to the database with the information provided 7. The system gives a feedback message of addition success

2.7 Use case diagrams

The following are the use case diagrams that represent the different use scenarios of the system

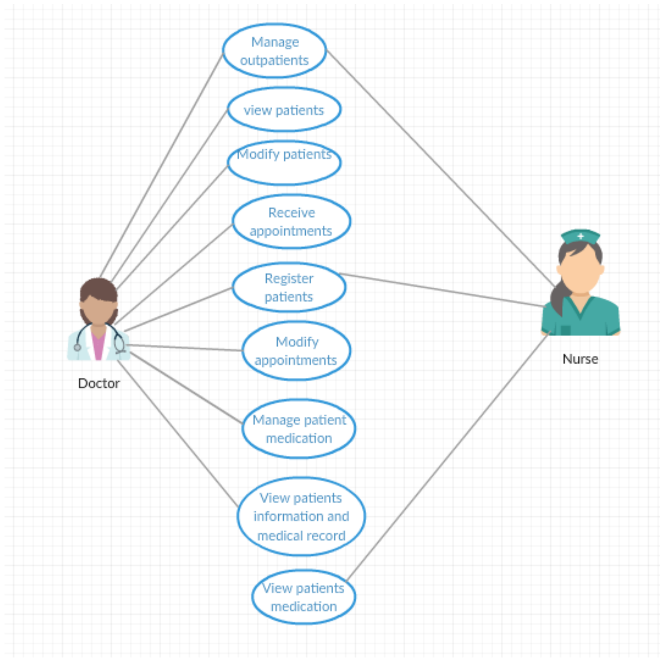


Figure 2 Use case scenario for Doctor and Nurses

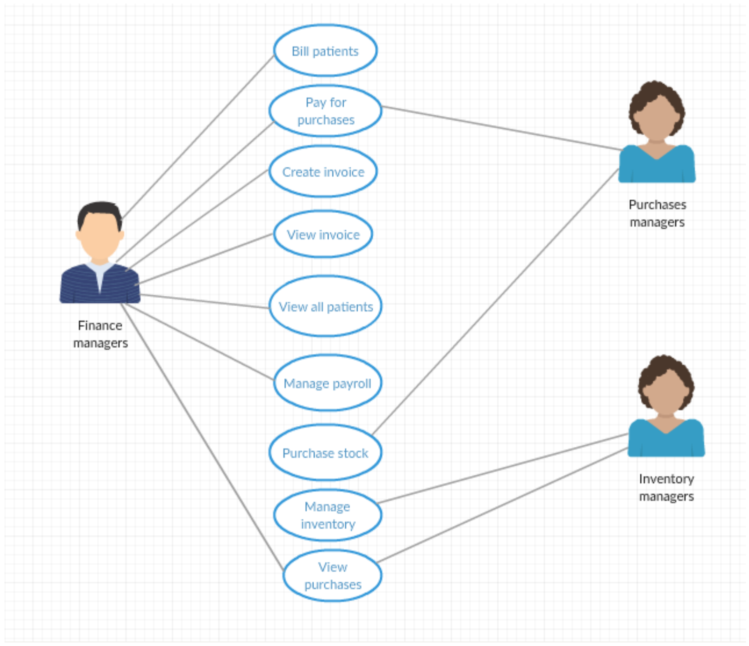


Figure 1 Use case scenario for Finance and Purchase managers

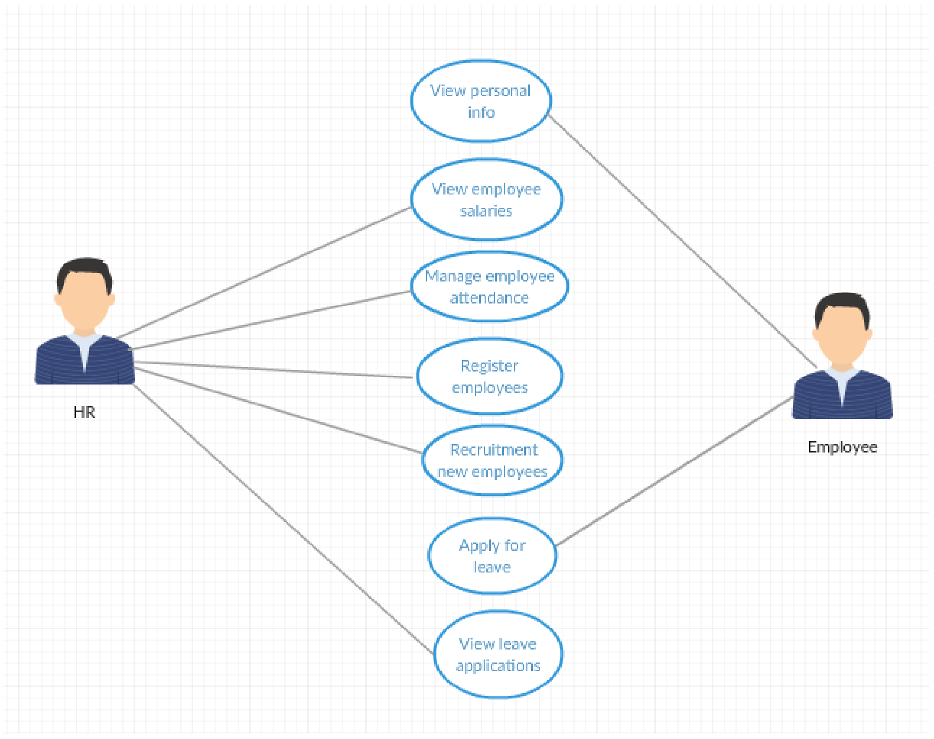


Figure 3 Use case scenario for Human resource managers

Chapter 3: Design and Architecture

This chapter describes the system's high-level architecture.

3.1 System Architecture

The system's security requirement and the robust features informed the architecture that was used. The Model, View and Controller (MVC) architecture has been an industry standard for software projects. This architecture does not only make modifying software components neat, but also offers flexibility for doing it. It separates the software's logic, interfaces and databases such that a developer can focus on one component without depending on the other, hence the decision to use the MVC architecture. Because this software is robust, the MVC architecture is necessary.

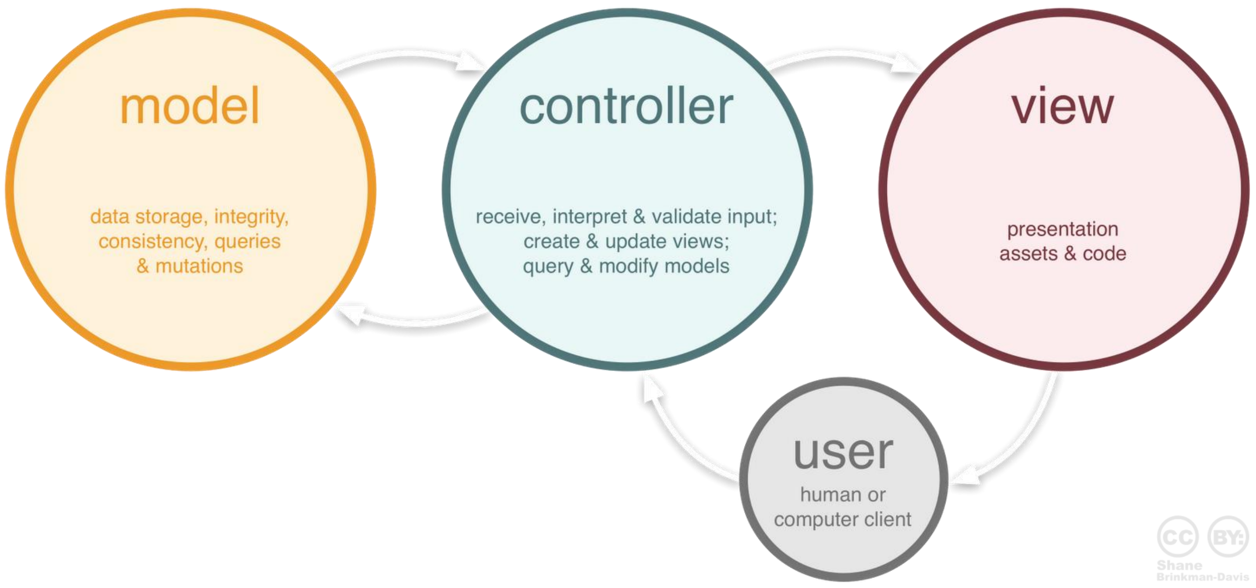

model
data storage, integrity, consistency, queries & mutations
controller
receive, interpret & validate input; create & update views; query & modify models
view
presentation assets & code
user
human or computer client
Shane Brinkman-Davis

Figure 3.1 MVC Architecture (Zeeshan, 2015)

In the figure of and MVC system above, interface status is displayed to the user by the view module. User actions such as clicks are relayed to the Controller which receives and interprets them. The controller is the intermediary between the Model and the View. When the Controller interprets the user actions, data may be written or read from the database and display to the user via the View.

While the web application used the MVC architecture, the mobile application was implemented using the MVVM architecture. Model-View-ViewModel works like the MVC model but adapted for mobile applications (and other applications that use data binding technologies). It separates the visual interfaces of a mobile application from the business logic.

3.2 Frameworks

For this project, The Laravel PHP framework was used. Laravel was used mainly because of the security and the simplicity that it offers. Also, Laravel uses the MVC architecture for its modules.

Another framework that was used is Bootstrap's front-end framework. This framework was merely inherited from Laravel, since Laravel uses the Bootstraps framework to render elements beautifully.

3.3 Software Modules

This software is divided into the following modules:

- Human Resource Management

  This module handles everything that has to do with employees in the hospital. It will store information about employees, allow employees to request leave. The Human Resource Department will be able to manage employees from this module

- Inventory Management

This module will keep inventory of stock. It will allow users to generate report of available stock. Users will also be able to mark items as used when they use a resource. This module will provide, to a certain degree of accuracy, information about where an item is in the company.

- Accounting and Finance Management

  The accounting and Finance Module will allow users to be able to manage the finance of the company. It will keep track of assets and liabilities, records of purchases, records of salaries etc. It will provide interface for billing patients. It will display patients whose payments are pending. It will register and display expenditures, purchase ledgers etc.

- Hospital Management

  This module will keep record of all the patients in the hospital. From here, information about a patient can be pulled from the database and displayed. Patients can be marked as discharged or admitted from this module. Users will also be able to manage other aspects of the hospital from here.

3.4 Database Architecture

The entity-relationship database model was used to generate a database schema that was normalized. Data normalization is important especially on update operations to reduce possibilities of data anomaly and cases of data redundancy. Some entities needed to be decomposed to maintain consistency of data across all the entities. Other entities needed to be generalized/specialized, and the generalization-specialization was done accordingly.

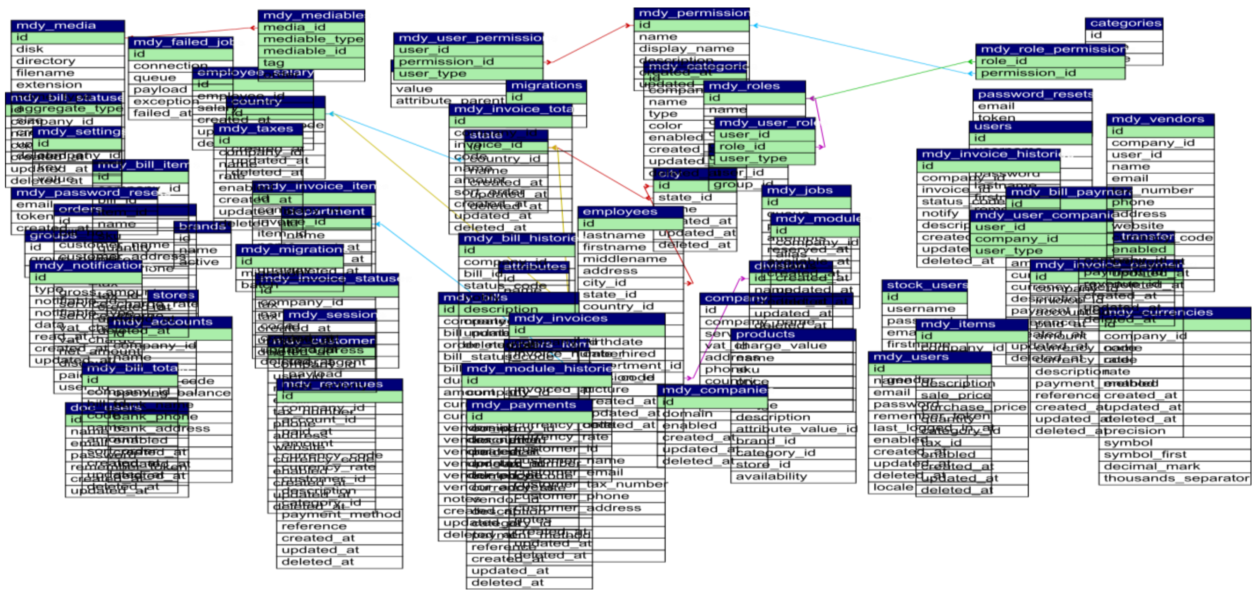


Figure 5 Database schema showing relationships between tables

3.5 Data Design

Appendix B shows the data configuration that was used in this software. [See appendix B]

3.6 Activity Diagrams:

The following diagrams show the activities for various users of the system.

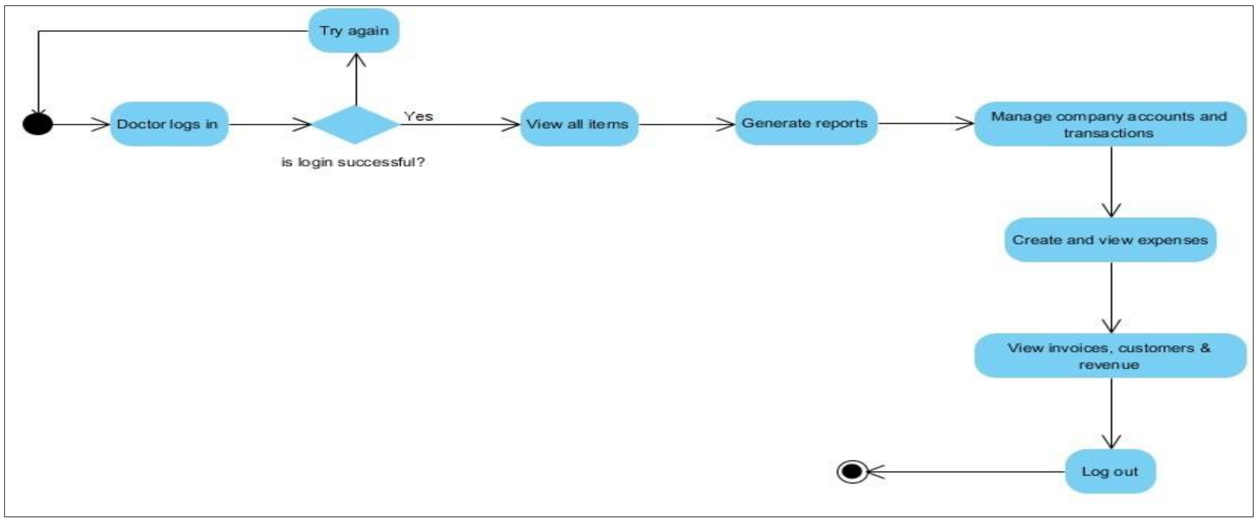


Figure 6 Activity diagram for Accountant

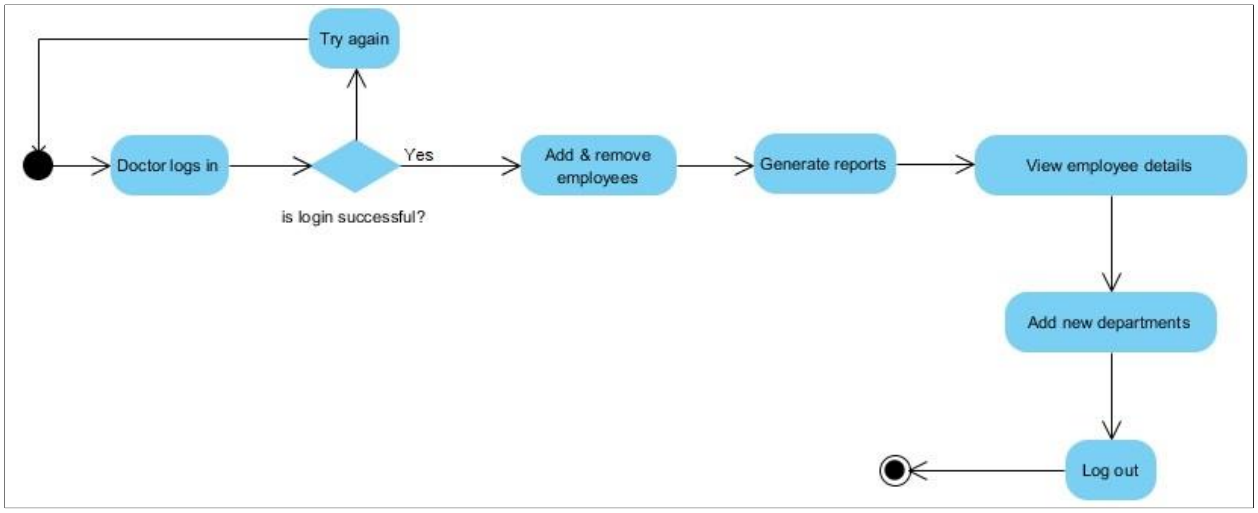


Figure 7 Activity diagram for HR manager

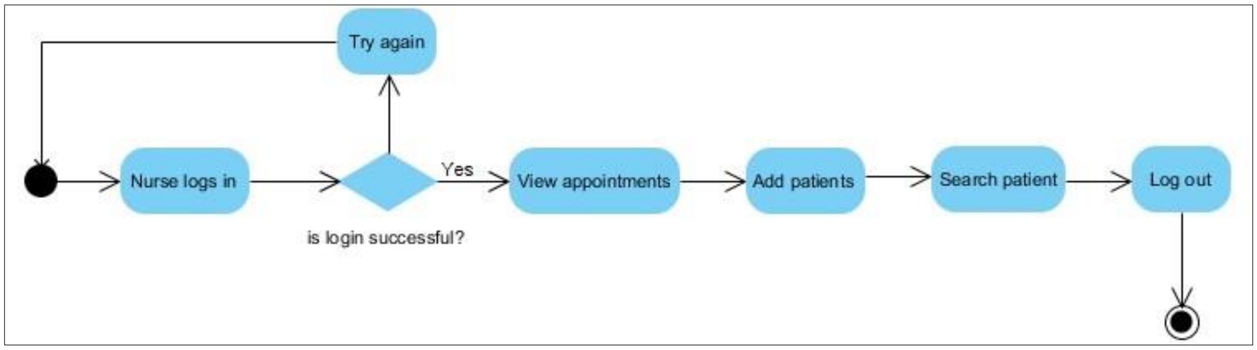


Figure 8 Activity diagram for Nurse

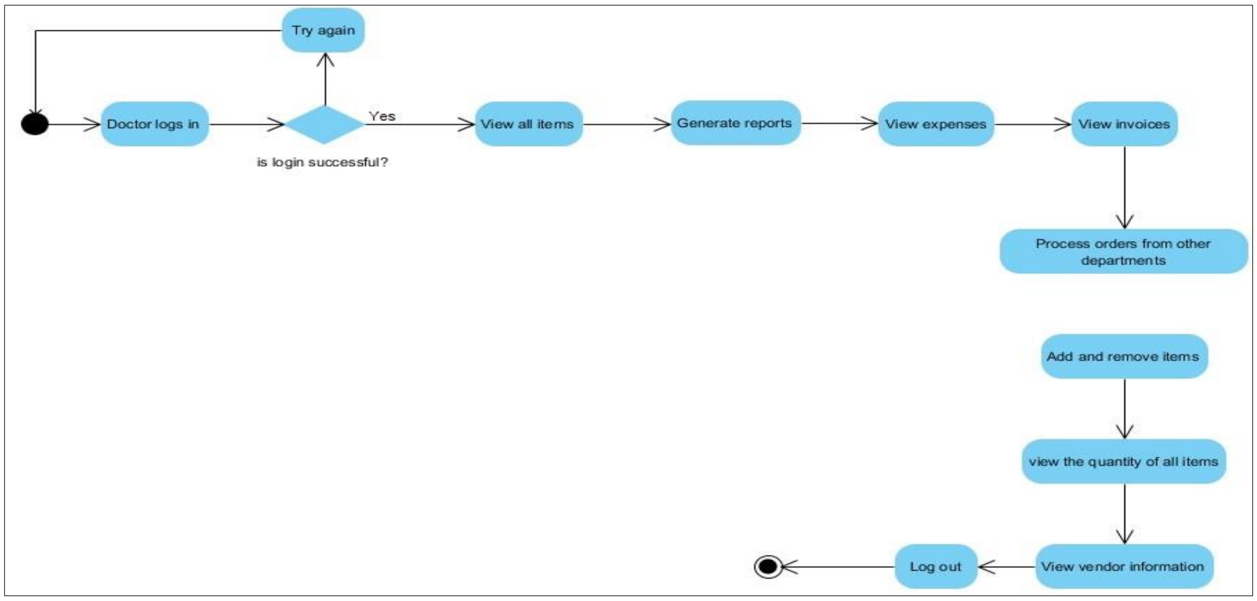


Figure 9 Activity diagram for Inventory manager

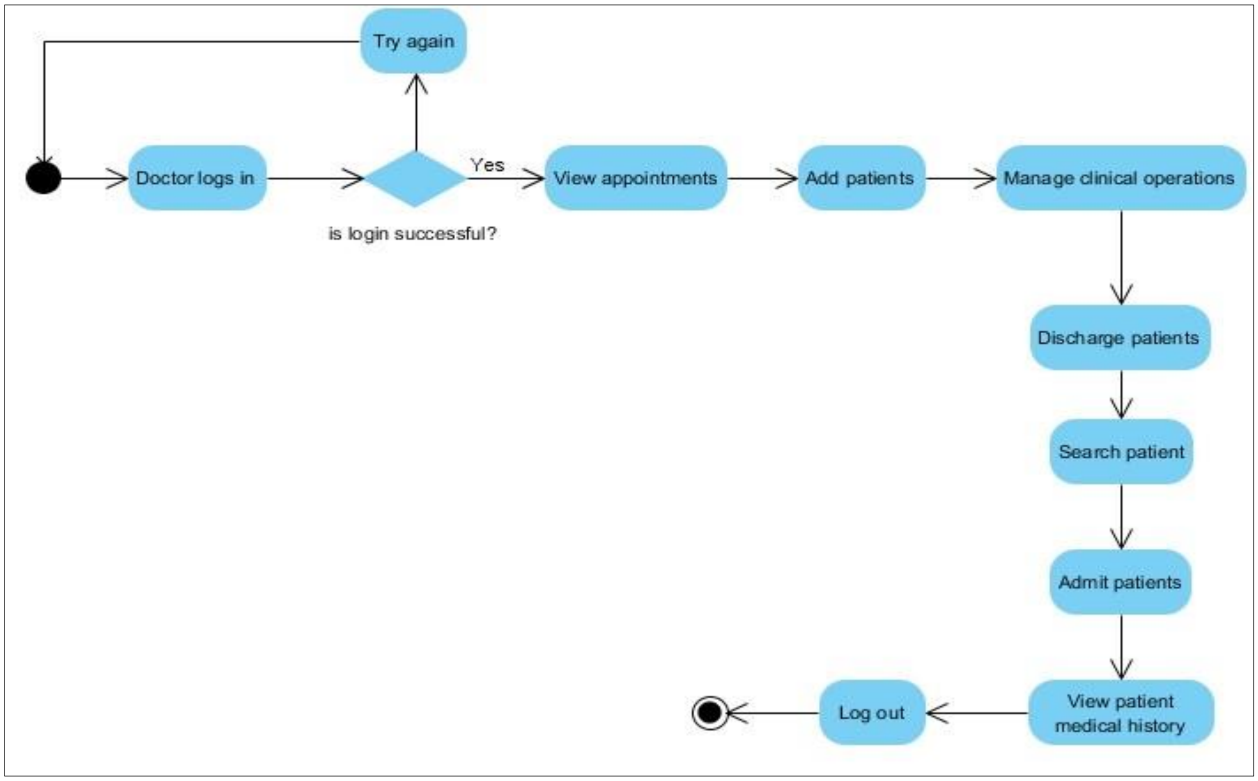


Figure 10 Activity diagram for Doctor

Chapter 4: Implementation

4.1 Overview

The implementation phase of this software specifies how the requirement specifications were converted into a usable system. This phase draws from the software design and involves actual programming and refinement of some of the requirement specifications. The implementation started by first using the requirement specifications to build a low-level prototype of the system. Through testing and evaluation, the prototype was then iteratively improved upon.

Considering the different users of the system and the various use case scenarios that will occur, a combination of data and event driven behavioral models were adopted as models of the dynamic behavior of the system as it executes.

4.2 Data-driven Modeling

"Data-driven models show the sequence of actions involved in processing input data and generating an associated output" (Sommerville, 2011)

4.3 Event-driven modeling

Event-driven modeling assumes that a system has a finite number of states and events cause a transition from one of the states to another (Sommerville, 2011). The event-driven model is important for showing how a system responds to events that occur.

4.4 The implementation

As already stated, implementation started with a low-level prototype of the system. Through continuous testing and evaluation, the system was iteratively improved upon. Three principal factors were taken into account in building the system: Security, usability and consistency.

Given that the system is a health ERP, which will be storing and managing medical details of patients, as well as other important operations in an organization such as finance, employee records, inventory etc., the system needs to be as secured as possible. In achieving this, the choice of Laravel as a backend framework was made (more details on Laravel below). To further ensure security and privacy of records, every important attribute is encrypted.

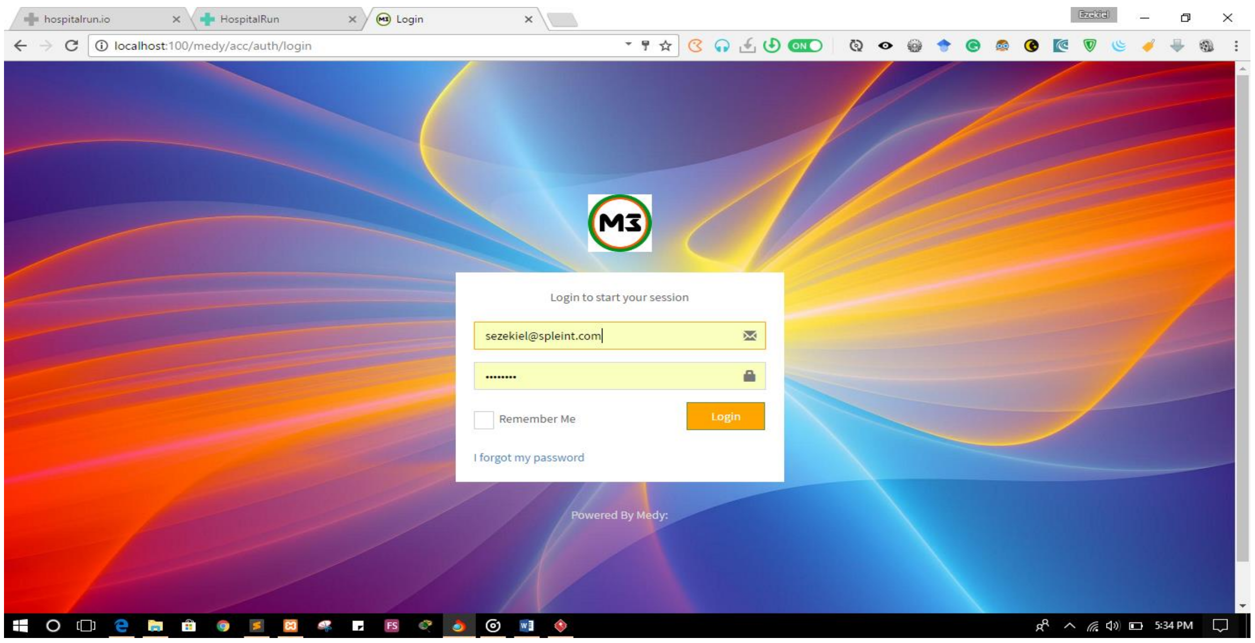
The system was also built with focus on usability and learnability. Even though it is not expected that users of the system will be novice users of a computer application, in general, no user likes to spend too much time in figuring out how to use a new software. The system was therefore built to be responsive and have intuitive, user friendly interfaces with icons and descriptions. After a brief period of guided interaction, a novice user should be able to use the system comfortably with few errors.

Every software needs to be consistent. Consistent with standard rules and guidelines and with the operations in similar software. This is important because it reduces the learning time, reduces errors in using the software and memory load on users. This system has adhered strictly to standard rules and guidelines and conventional practices, from the code structure to the physical interfaces.

This application is divided into five (5) modules; each module handles specific functions of the organization or gives privilege to users to perform specific operations. The modules include Accounting module, Hospital management module, Human Resource module and

Inventory module. The Hospital module is further subdivided into the Doctor and Nurse modules. The Nurse module is simply the Doctor module with limited privileges.

As it is with many other administrative applications, when administrators log into the system, they are redirected to the dashboard – ideally having a navigation bar with notifications and profile links on the top right corner of the screen, and a navigation pane on the left side of the screen. This is to ensure consistency and for users to quickly acquaint themselves with the new system.



Login to start your session

sezekiel@spleint.com

Powered By Medy:

Figure 11 Login screen to the accounting module

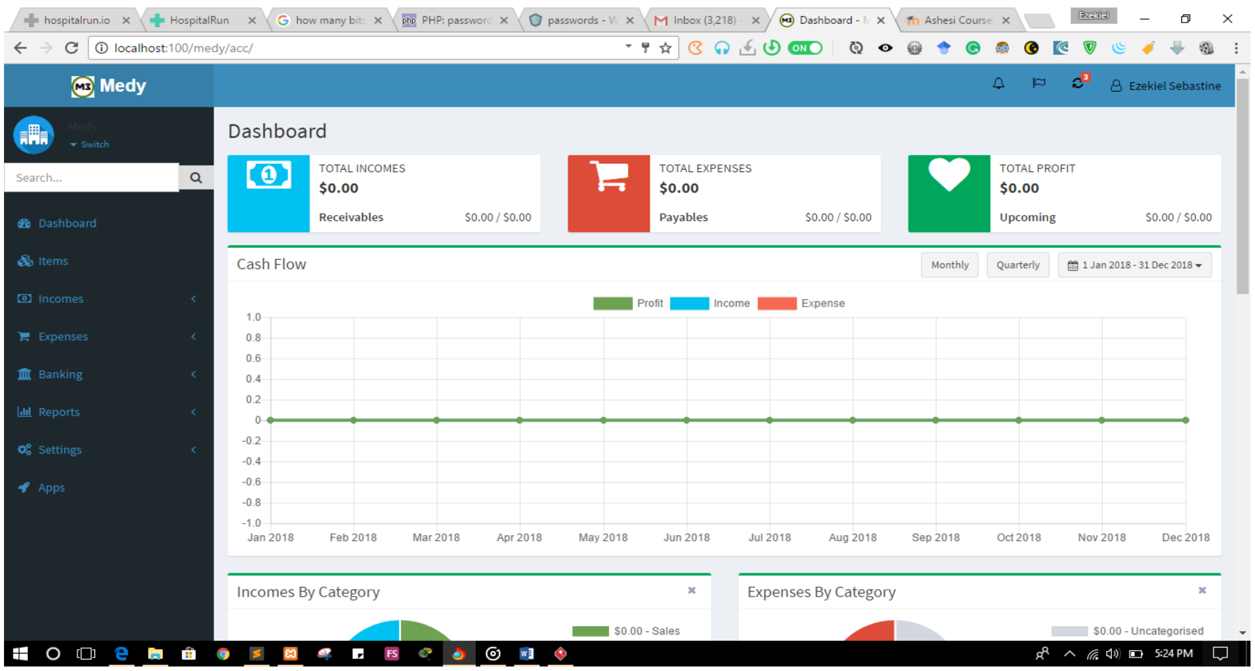


Figure 12 The account management dashboard showing the navigation bar and navigation pane

By clicking on any of the links on the navigation pane, the administrator is sent to a page which initially contains a table of all records for the link clicked. For example, clicking on 'items' will display a table with all available items, likewise, clicking on 'invoices' displays a table with all invoices and so on.

The records in the table can be sorted either in ascending or descending order and according whichever column that the administrator wants. From here also, new records can be added, existing ones can be updated or deleted.

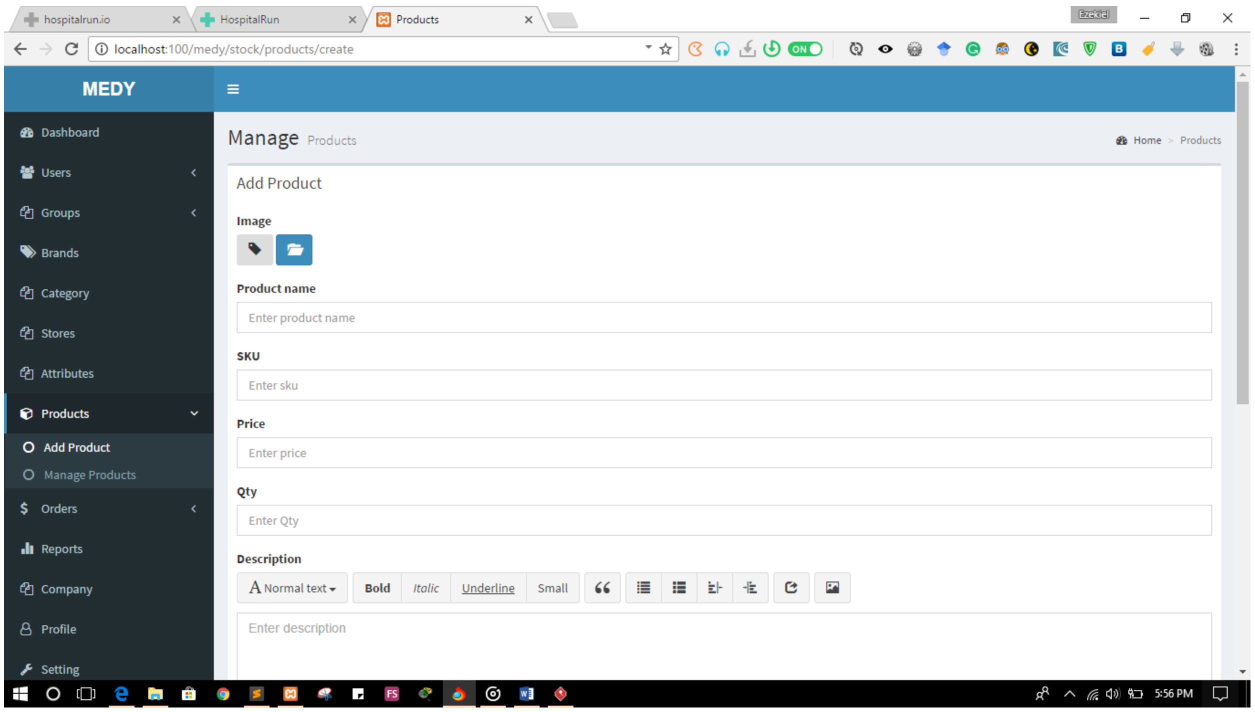
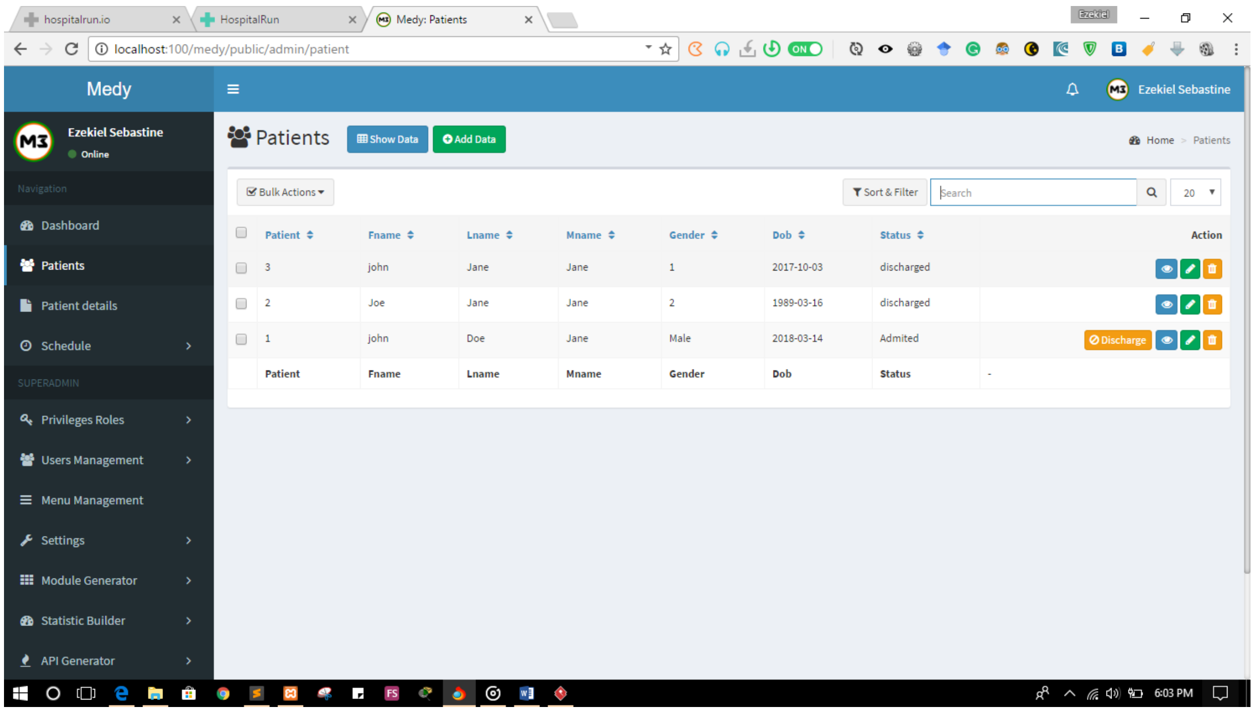


Figure 13 Inventory module showing form to add an additional item



Patient	Fname	Lname	Mname	Gender	Dob	Status	Action
3	john	Jane	Jane	1	2017-10-03	discharged	
2	Joe	Jane	Jane	2	1989-03-16	discharged	
1	john	Doe	Jane	Male	2018-03-14	Admited	Discharge
Patient	Fname	Lname	Mname	Gender	Dob	Status	-

Figure 14 Doctor module showing list of all patients

4.5 Tools and Technologies

The underlying development platform that was used to build this system is Windows Operating system. The operating system housed all the other tools and technologies.

XAMPP was used as the development environment. The choice of XAMPP as the development environment was because of the features that it has. First, it is free and open-source, which makes it possible for anyone to use it. Secondly, it is simple and easy to learn and has a robust community of users that are ready to help. XAMPP also houses other important tools relevant for this project like the Apache PHP web server and MySQL database server. PhpMyAdmin, a free and open source administration tool for MySQL database is also bundled into XAMPP and was used to administer the MySQL database. The table below shows the list of technologies in a more organized form.

Table 4.1 List of Technologies

Frontend	CSS, JavaScript/JQuery, HTML, AJAX, XML
Backend	PHP, Laravel, CRUDBooster, Java
Database	MySQL, SQLite

4.6 Frontend

- HTML and CSS: HTML and CSS are the languages of the web. HTML is the building block of any web application and CSS is its beauty. Their importance in this system cannot be overemphasized since the system cannot exist without them.

- JavaScript/JQuery: JavaScript is used to make web pages dynamic and to have some form of interactivity in them. JavaScript can modify CSS in a HTML document. JQuery is a JavaScript technology which combines CSS and JavaScript to make writing JavaScript simpler. For instance:

```
document.getElementById('id').innerHTML = "Hello world";
```

can be written simply in JQuery as:

```
$('#id').html ("Hello world");
```

In this application, JQuery was used to manipulate dates, display graphs, filter and sort table records without necessarily querying the database.

- AJAX: was used to make asynchronous request to the server.
- XML: XML was used to create and render the mobile application components.

4.7 Backend

- PHP: PHP is the most popular server-side scripting language. It is simple to learn, and it comes in two different forms; as a procedural language and as an object-oriented language. This application uses the object-oriented form. PHP is also flexible and allows HTML, JavaScript and even CSS to be embedded in it.
- Laravel: Laravel is another free and open-source PHP web framework, that is intended for the rapid development of web applications. The choice of Laravel framework over others was influenced first by its security and second by its MVC pattern, then by its simplicity and robust community.

- CRUDBooster: This is a framework that implements Laravel and makes generating pages for CRUD operations faster. The framework was used so that more time and attention will be directed on the overall functionality of the system.
- Java: Java is the standard language for developing native android applications.

4.8 Database

- The choice of MySQL as the database for this application is its simplicity of use and ease of integration with PHP and Laravel framework. It was chosen because it is also a relational database.
- SQLite: SQLite database a non-relational database that works well with Android applications.

4.9 Other Tools

Table 4.2 List of tools

Tool	**Use**
Sublime text	Primary source code editor
Browsers (Chrome, Firefox, Edge)	Used to view the web pages
AdminLTE HTML template	Used to create the dashboards
Android Studio	Integrated Development Environment for native Android applications

Chapter 5: Testing

5.1 Testing Overview

After development, the software needs to be tested. Testing ensures that a software does exactly what it was built to do. It is part of the validation and verification process. The following tests were conducted: Development testing, Release testing and User testing.

5.2 Development Testing

Development testing spanned the entire implementation phase. Development testing was done by the developer to identify and fix bugs in the system as the implementation process continues. The development testing phase is divided into three components:

5.2.1 Unit Testing

During unit testing, the smallest single units of the system such as functions, classes and database queries were tested. PHP provides a good library called PHPUnit for doing the unit testing. Most queries were first executed directly in phpMyAdmin to validate the results before embedding them in the code.

5.2.2 Component Testing

The single units mentioned above usually cannot function alone. Single units of implementation are integrated to form components and modules. These modules were also tested to ensure that there are no compatibility issues between the single units that make up the components and that the process of integration did not break things.

5.2.3 System Testing

After component testing was successful, individual components were further integrated to form a complete system. The system was tested with various inputs, both correct and invalid input were tested, and the outcome was validated.

5.3 Release Testing

In the release testing phase, testing was done to ensure that the software did what it was built to do, according to the requirement specifications. The software was uploaded to a live server and a real world working scenario was staged to depict how the software will be used in production. Key issues that arose due to this change of environment were that some of the images which displayed properly on the local server appeared broken on the live server. This and many other test results were used to tune the system.

5.4 User Testing

Various users were asked to use the system while the developer observed the flow of events. The results of this testing were used to iteratively retune the system until the system achieved stability.

Table 5.1 User test cases

Test Suite	TS1					
Title	Testing login functionality					
Description	To test the different scenarios that might arise while a user is trying to login					
S/N	Summary	Dependency	Precondition	Post condition	Execution Steps	Expected output

TC1	Verify that user who already registered with MEDY can login with correct user ID and password		User must have registered with MEDY	User is logged in	• Type username • Type password Click on login	"Home" page for the user is displayed
TC2	Verify that user who already registered with MEDY cannot login with incorrect user ID and or password		User enters an incorrect username or password	User is denied login	• Type username • Type password • Click on login	The login page stays with error message displayed
TC3	Verify that user who is not registered with MEDY cannot login		User enters username and password	User is not logged in	• Type username • Type password Click on login	The login page stays with error message displayed

TEST SUITE	TS2
Title	Verify “Patient or Doctor logout” functionality
Description	To test the different scenarios that might arise when the patient or the doctor is trying to logout of the system

S/N	Summary	Dependency	Pre-condition	Post-condition	Execution Steps	Expected output
TC1	Verify that patients or doctors can logout of the system successfully		The patient of the doctor should already be logged into the system	Patient or the user is successfully logged out of the system	• Click logout button	Both patients and doctors are taken back to login page

TEST SUITE	TS3
Title	Verify “Patient delete account” functionality
Description	To test the different scenarios that might arise when the patient is deleting the account

S/N	Summary	Dependency	Pre-condition	Post-condition	Execution Steps	Expected output

TC1	Verify that the patient can delete their account from the system successfully		The patient should already have an existing account with MEDY	The patient will seize to exist in MEDY's system	The patient logins into the system successfully They click on the delete account option They confirm they want to delete their account	A message of successful account deletion is displayed

Test Suite	TS4					
Title	Testing Adding new patient's functionality					
Description	To test the different scenarios that might arise while a patient is being added					
S/N	Summary	Dependency	Precondition	Post condition	Execution Steps	Expected output
TC1	Verify that an unregistered patient can be registered unto to MEDY		• Patient has an account with Medy	Patient is registered by doctor	• Complete the patient form • Click on the add	Success message is displayed, and a unique ID is

			• Patients details have been taken by doctor		patient button	generated and given to patient
TC2	Verify that a registered user is not able to register on MEDY		User has an account with MEDY. Patients details have been taken by doctor	Patient does not get registered	• Complete the patient form Click on the add patient button	The registration page stays, and an error message is displayed
TC3	Verify that a patient with invalid details is not able to register unto MEDY		Patient does not have account MEDY	Patient is not registered	• Complete the patient form Click on the add patient button	The registration page stays, and an error message is displayed

Test Suite	TS6
Title	Testing Remove patient functionality
Description	To test the different scenarios that might arise while a patient is being removed

S/N	Summary	Dependency	Precondition	Post condition	Execution Steps	Expected output
TC1	Verify that a patient exists to be removed		The patient exists on database		Search a patient through ID or name or any convenient attribute	Patients record is found and displayed
TC2	Verify that an existing patient can be removed from database	TC1	The patient exists on database	Patient is removed	Click on the remove button	Removal success message is displayed

Test suite	TS8
Title	Verify “view appointments” functionality
Description	To test the different scenarios that might arise when the doctor is trying to view the appointments

S/N	Summary	Dependency	Pre-condition	Post-condition	Execution Steps	Expected output
TC1	Verify that a doctor logged into the system can view appointments successfully		The doctor should be successfully logged into the system	The doctor views all appointments successfully	The doctor logins into the system successfully They click on view appointments link	"Appointment" page containing all appointments is displayed
TC2	Verify that a doctor who is not logged into the system is unable to view appointments		The session has timed out	The doctor cannot view appointments	The doctor clicks on view appointments link	The "login" page is displayed and the doctor is prompted to login into the system
TC3	Verify that 'view appointments' works	TC1	The doctor should be successfully logged	There is a message notifying the doctor	The doctor logins into the system successfully	The message "Ops, There are

	successfully even if there are no appointments		into the system	that there are no appointments available	They click on view appointments link	no appointments available today" appears

Test suite	TS9
Title	Verify "book appointments" functionality
Description	To test the different scenarios that might arise when the patient is trying to book an appointment

S/N	Summary	Dependency	Pre-condition	Post-condition	Execution Steps	Expected output
TC1	Verify that a patient logged into the system can book an appointment successfully		The patient should be successfully logged into the system	The patient books an appointment successfully	The patient logins into the system successfully They click on book appointments link	"Appointment" booking page is displayed where the patient will enter appointment details

TC2	Verify that a patient who is not logged into the system is unable to book an appointment		The session has timed out	The patient cannot book an appointmen t	The patient clicks on book an appointment link	The "login" page is displayed, and the patient is prompted to login into the system
TC3	Verify that a patient cannot book the same appointment twice	TC1	The patient should be successfull y logged into the system	The patient cannot book an appointmen t	The patient logins into the system Clicks on book appointments link Books an appointment and tries to submit it OR The patient tries to re-submit the appointment form	"You cannot book the same appointme nt twice" message is displayed

Test suite		TS10				
Title		Verify “update patients’ data” functionality				
Description		To test the different scenarios that might arise when the doctor is trying to update patients’ data in the database				
S/N	Summary	Dependency	Pre-condition	Post-condition	Execution Steps	Expected output
TC1	Verify that a doctor logged into the system can update a patient’s record successfully		The doctor should be successfully logged into the system The patient whose record is to be updated should have an active account with the specific hospital	The doctor updates the patients’ records successfully	The doctor logins into the system successfully They click on view patients’ menu and selects the patient whose record is to be updated	Patient’s records in the database are updated and the changes are shown on their page
TC2	Verify that a doctor who is not logged into the		The session has timed out	The doctor cannot update a	The doctor clicks on edit button	The “login” page is displayed,

	system is unable to update a patient's record			patient's record		and the doctor is prompted to login into the system

Test Suite	TS14
Title	Testing Retrieving patient details functionality
Description	To test the different scenarios that might arise while a patient details is being retrieved

S/N	Summary	Dependency	Precondition	Post condition	Execution Steps	Expected output
TC1	Verify that a registered patient's details can be retrieved		Patient must have registered with MEDY Doctor must have logged in	Patient's details are shown	Scan patient id	Patient's details are displayed

			Patients id has been scanned			
TC2	Verify that a patient that has not registered details cannot be shown		Doctor must have logged in Patients does not have a ID	Patient's details are not shown		Patient's details are not displayed
TC3	Verify that a patient with wrong ID is not shown any details		Doctor must have logged in Patients supplies the wrong ID	Patient's details are not shown	Scan patient ID	Patient's details are not displayed

Chapter 6: Conclusion

To wrap up this report, this section discusses possible improvements to the system, key challenges faced, and recommendation.

6.1 Challenges

- Time constraint: An ERP will normally take about 8 months for a well-coordinated development team to complete, even with minimal modules. This was a challenge for me because I am only a single developer with few months at my disposal to complete the entire ERP. As a result, I had to relinquish some functionalities that I planned to implement.
- Data security: As a hospital ERP, I needed to make sure that the entire system, especially the data, is as secured as possible. Accomplishing this was where the difficulty stood.

6.2 Future Work

Because of time constraint explained above, some functionalities could not be implemented in the current version. However, the next version will have the following:

- Machine learning: It goes without saying that machine learning is the way of the future. A greater chunk of technology will be powered by machine learning and artificial intelligence. To this end, it is imperative to use data mining and machine learning strategies in this application that is data driven, not just to be part of the machine learning drive, but to harness its endless possibilities as well. This functionality will make it possible to predict epidemics even before they break out, help health agencies in distributing aid etc.

- Mobile application features: Inspired by Dr. Ayorko Korsah and Francis Delali Vorgbe’s work on using native phone features to determine road surface quality (Korsah, 2014) (Vorgbe, 2014), in the future, native sensors of mobile devices such as heat sensors, gyroscopes, accelerometers etc., will be used to quickly retrieve patient health data with high degree of accuracy without using a clinical device. This will save hospital resources and save costs for both the patients and hospitals.
- More modules: This ERP is minimal. In the future, more modules that are capable of handling other specific organizational activities can be added. Such modules may include project management, Customer Relationship Management (CRM)

6.3 Conclusion

With the rapid pace of technological advancement, this application will revolutionize how hospitals operate and how healthcare is delivered to patients.

References

IBM. (n.d.). *Visual Recognition Tool Beta*. Retrieved from IBM Bluemix: https://watson-visual-recognition.ng.bluemix.net/

Jonesa, K. H., Laurieb, G., Stevensb, L., Dobbsa, C., Forda, D. V., & Leac, N. (2017). The otherr side of the coin: Harm Due to Non-use of Health Related Data. *International Journal of Medical Informatics*, 43-51.

Korsah, G. A. (2014). *Faculty and Staff*. Retrieved from Ashesi univeristy: http://www.ashesi.edu.gh/academics/computer-science/faculty-pages-cs/1220-getrude-ayorkor-korsah.html

Netsuit. (2017). *What is an ERP*. Retrieved from Netsuits: http://www.netsuite.com/portal/resource/articles/erp/what-is-erp.shtml

Newman, D. (2018, January 16). *Top 18 Tech Trends At CES 2018*. Retrieved from Forbes: https://www.forbes.com/sites/danielnewman/2018/01/16/top-18-tech-trends-at-ces-2018/#3b4675bf452f

Safran, C. (2014). Reuse of Clinical Data. *IMIA Yearbook of Medical Informatics*, 52-54.

Sayeed, Z. (2016, July 12). *ERP Solutions*. Retrieved from LinkeddIn: https://www.linkedin.com/pulse/what-erp-zara-sayeed/

Sommerville, I. (2011). *Software Engineering.* Boston: Addison-Wesley.

Vorgbe, F. D. (2014). *Classification Of Road Surface Quality Using.* Accra.

Zeeshan, A. A. (2015, 02 24). *Code Project*. Retrieved from Articles: https://www.codeproject.com/Articles/879896/Programming-in-Java-using-the-MVC-architecture

Appendix

Appendix A:

Application Screenshots

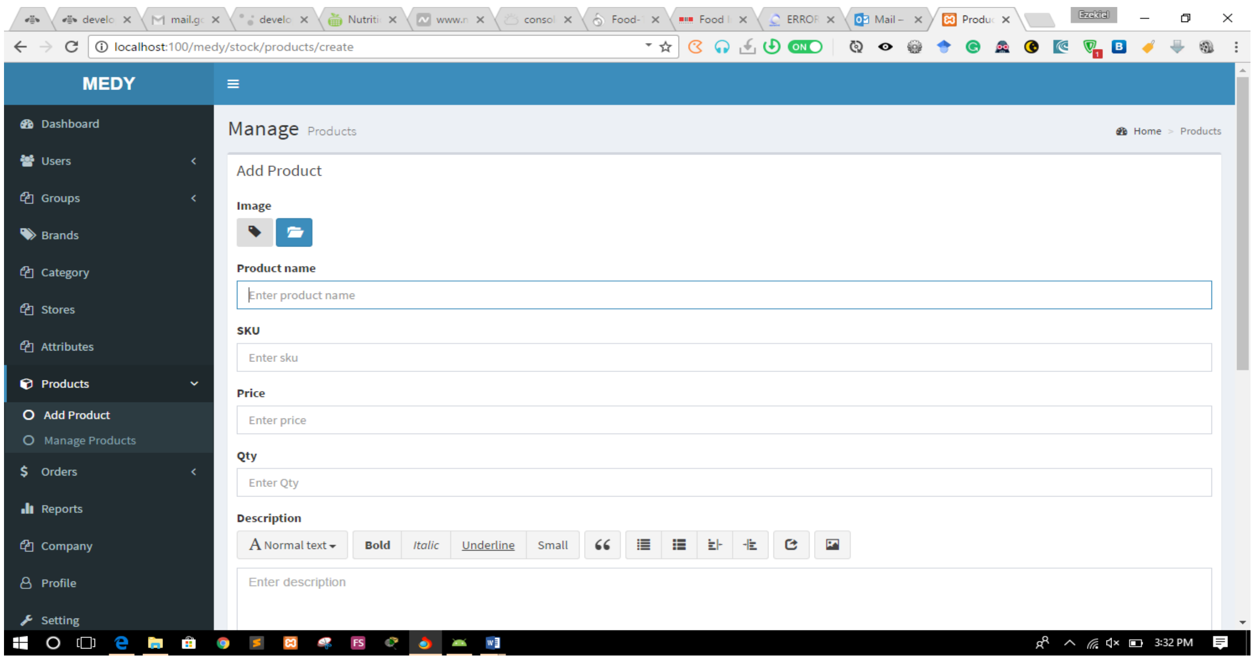


Figure 15 Inventory dashboard showing add new product

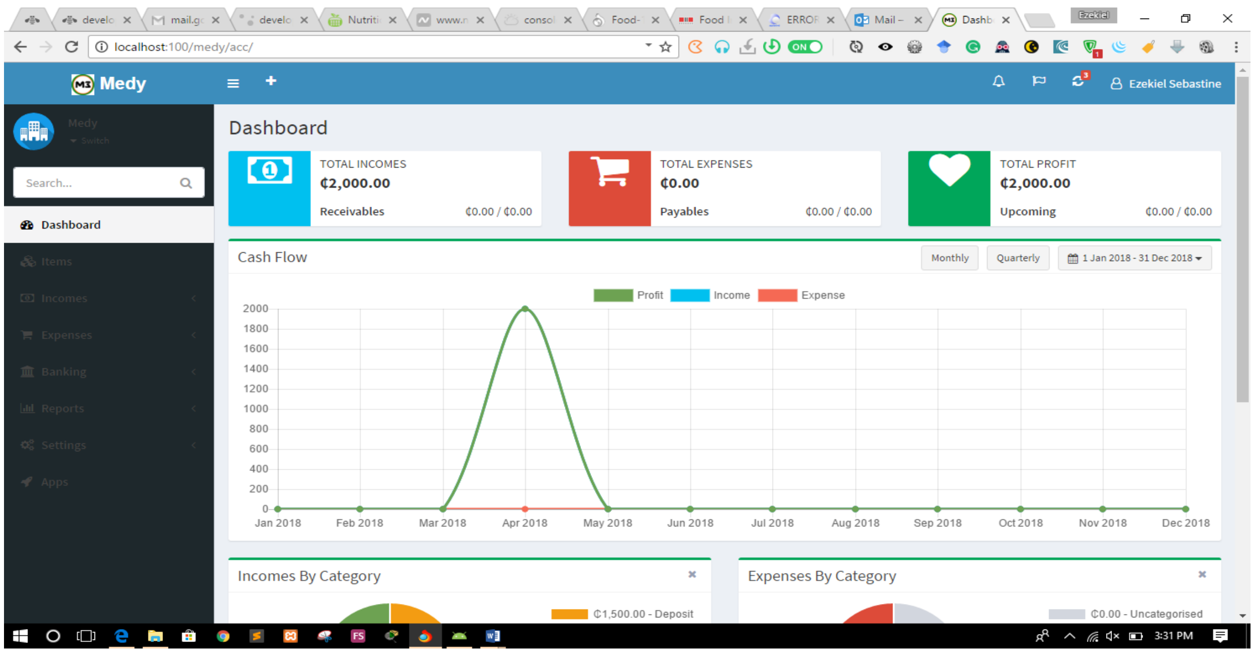
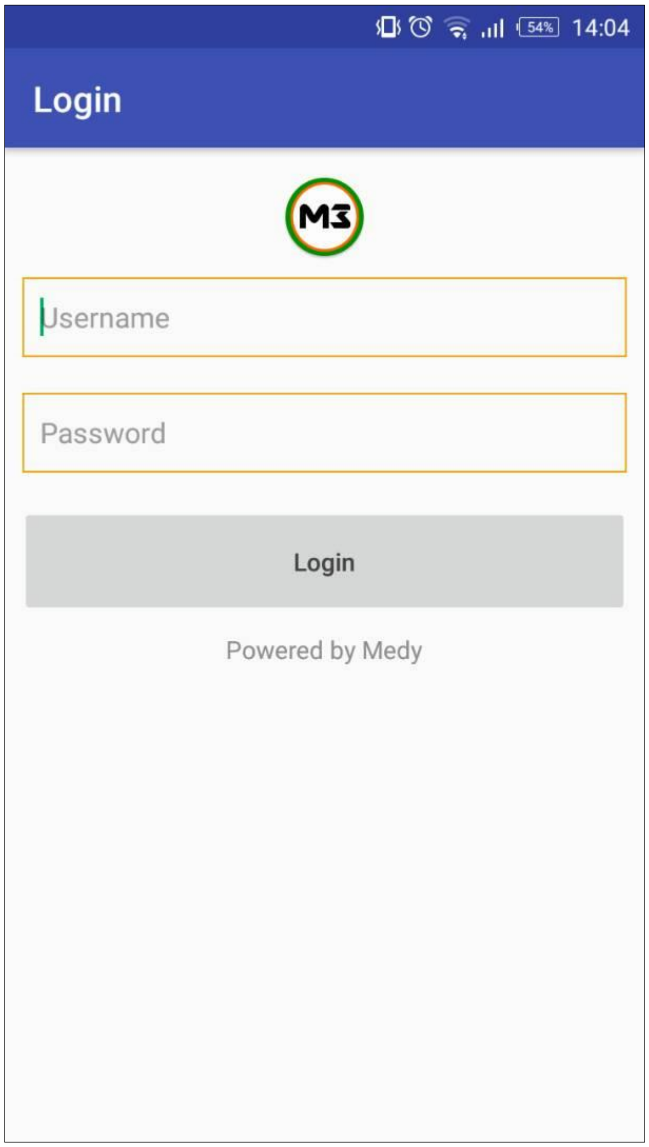


Figure 16 Accounts dashboard showing sales summary



14:04

Login

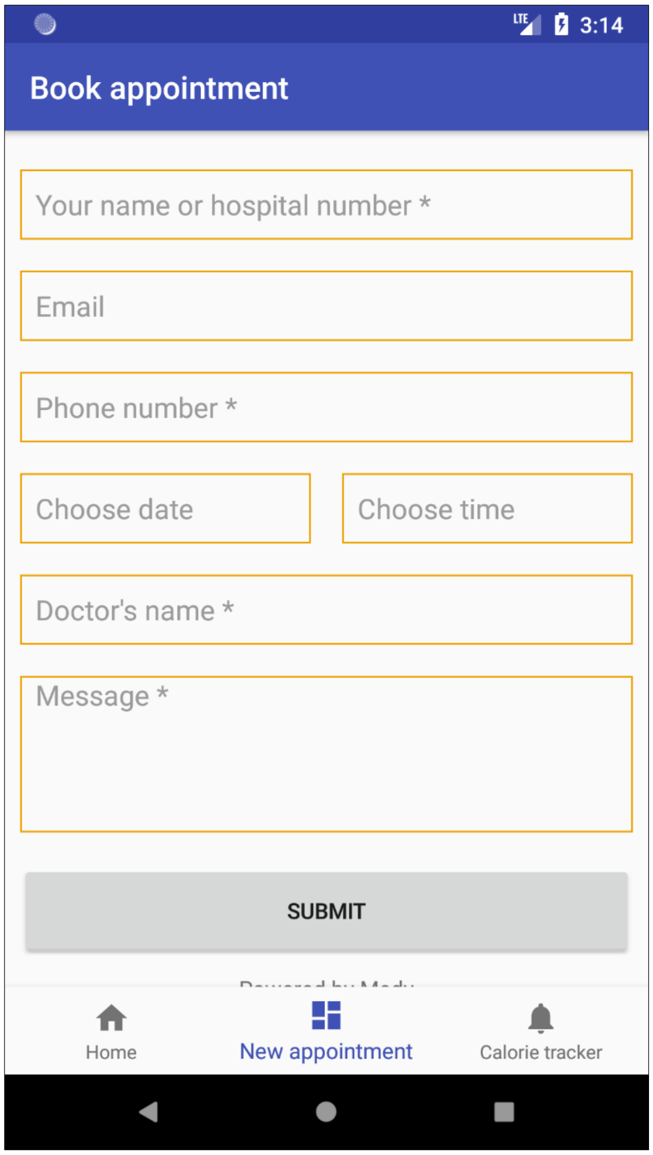
Username

Password

Login

Powered by Medy

Figure 18 Mobile App showing login screen



3:14

Book appointment

Your name or hospital number *

Email

Phone number *

Choose date

Choose time

Doctor's name *

Message *

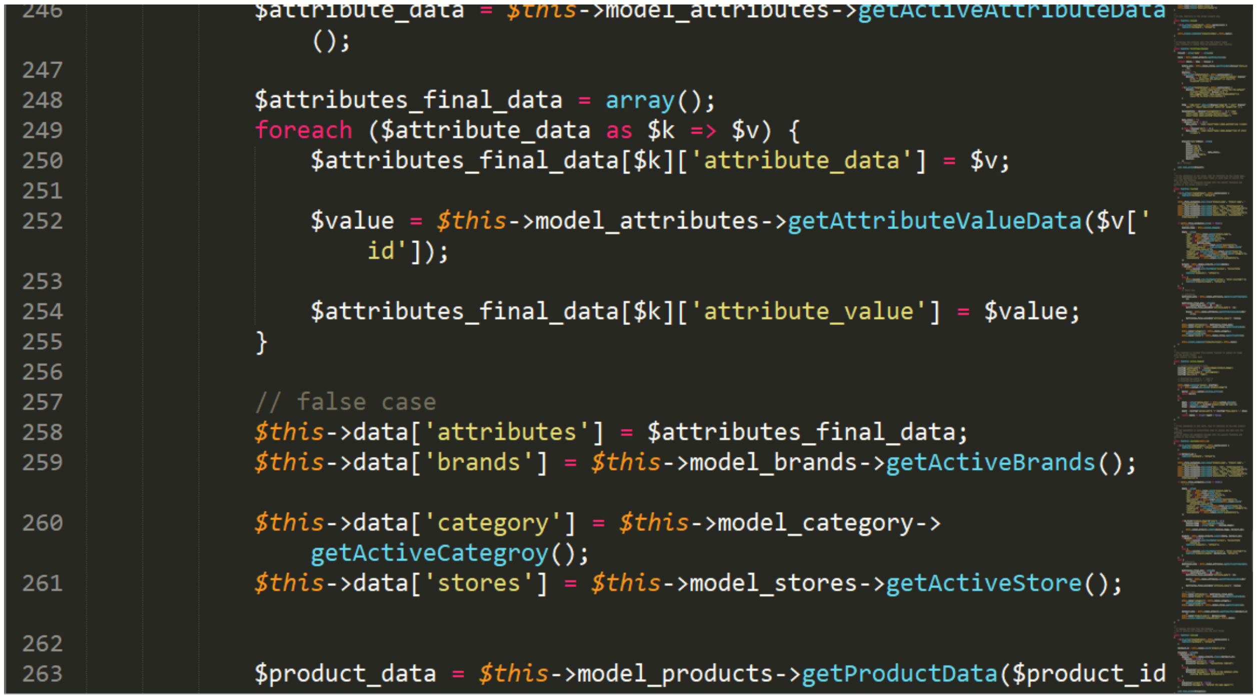
SUBMIT

Home

New appointment

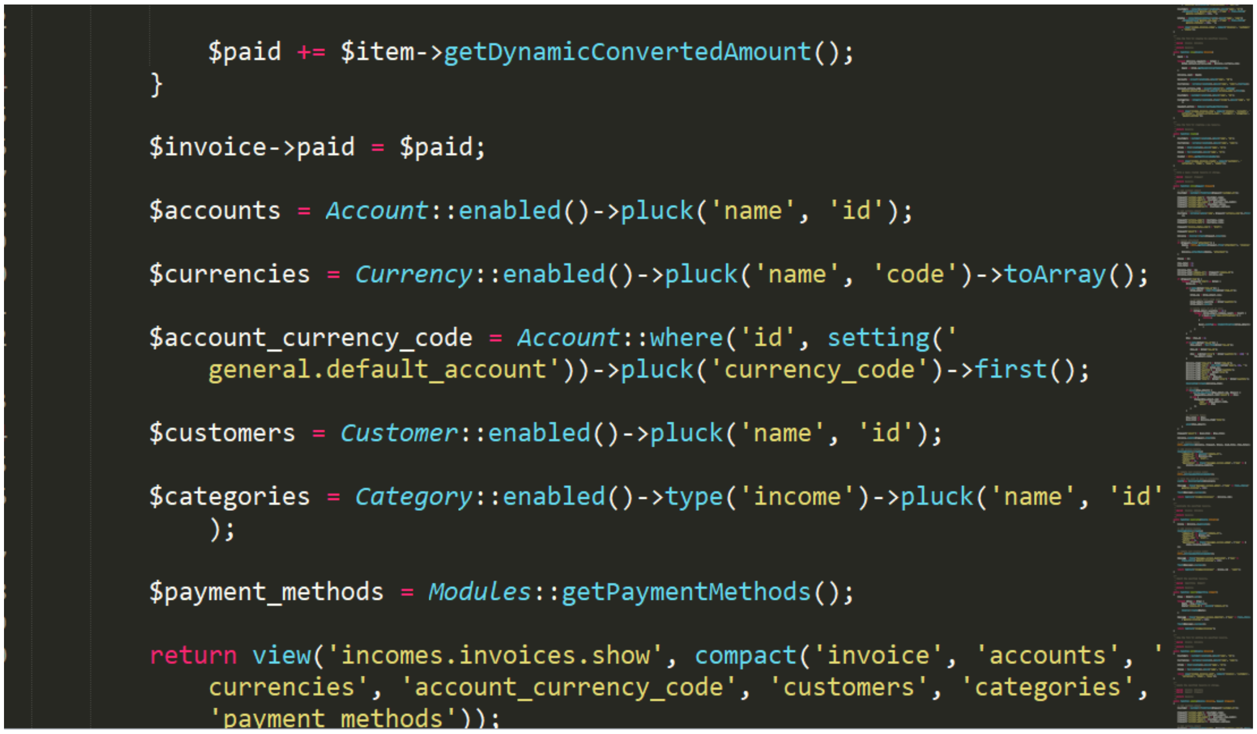
Calorie tracker

Figure 17 Mobile App showing book appointment



```
            $attribute_data = $this->model_attributes->getActiveAttributeData
                ();

            $attributes_final_data = array();
            foreach ($attribute_data as $k => $v) {
                $attributes_final_data[$k]['attribute_data'] = $v;

                $value = $this->model_attributes->getAttributeValueData($v['
                    id']);

                $attributes_final_data[$k]['attribute_value'] = $value;
            }

            // false case
            $this->data['attributes'] = $attributes_final_data;
            $this->data['brands'] = $this->model_brands->getActiveBrands();

            $this->data['category'] = $this->model_category->
                getActiveCategroy();
            $this->data['stores'] = $this->model_stores->getActiveStore();


            $product_data = $this->model_products->getProductData($product_id
```

Figure 19 Code snippet of a function that updates a stock item in the inventory module

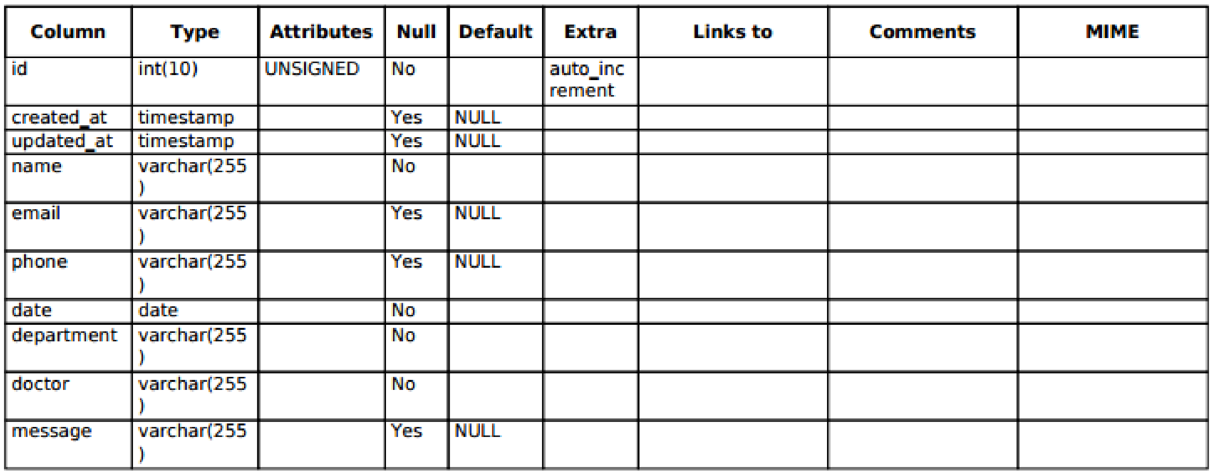


```
            $paid += $item->getDynamicConvertedAmount();
        }

        $invoice->paid = $paid;

        $accounts = Account::enabled()->pluck('name', 'id');

        $currencies = Currency::enabled()->pluck('name', 'code')->toArray();

        $account_currency_code = Account::where('id', setting('
            general.default_account'))->pluck('currency_code')->first();

        $customers = Customer::enabled()->pluck('name', 'id');

        $categories = Category::enabled()->type('income')->pluck('name', 'id'
            );

        $payment_methods = Modules::getPaymentMethods();

        return view('incomes.invoices.show', compact('invoice', 'accounts', '
            currencies', 'account_currency_code', 'customers', 'categories',
            'payment_methods'));
```

Figure 20 Code snippet of a function that displays an invoice in the accounts module

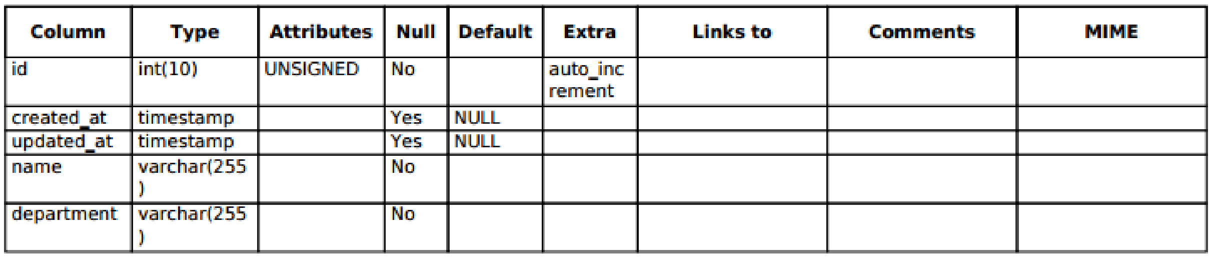
Appendix B

Important Database Tables



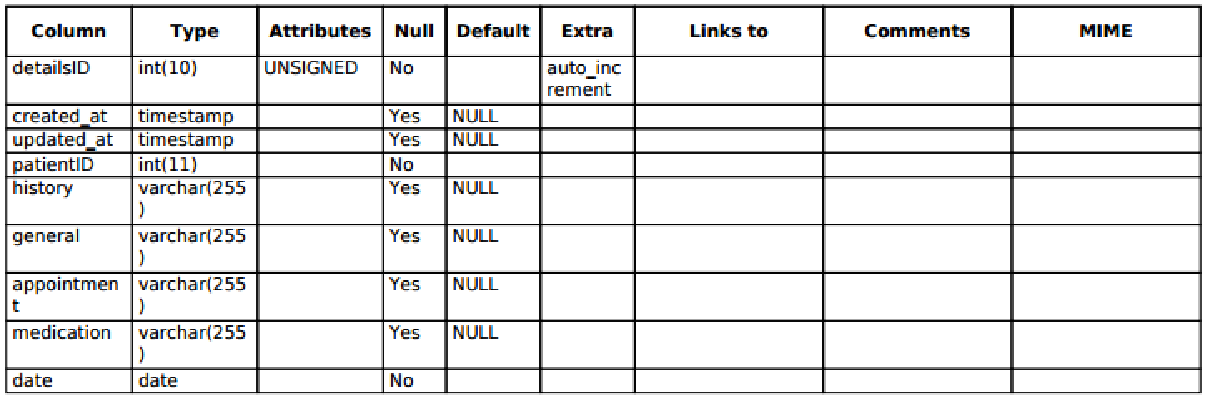
Column	Type	Attributes	Null	Default	Extra	Links to	Comments	MIME
id	int(10)	UNSIGNED	No		auto_increment			
created_at	timestamp		Yes	NULL				
updated_at	timestamp		Yes	NULL				
name	varchar(255)		No					
email	varchar(255)		Yes	NULL				
phone	varchar(255)		Yes	NULL				
date	date		No					
department	varchar(255)		No					
doctor	varchar(255)		No					
message	varchar(255)		Yes	NULL				

Figure 21 Appointment table



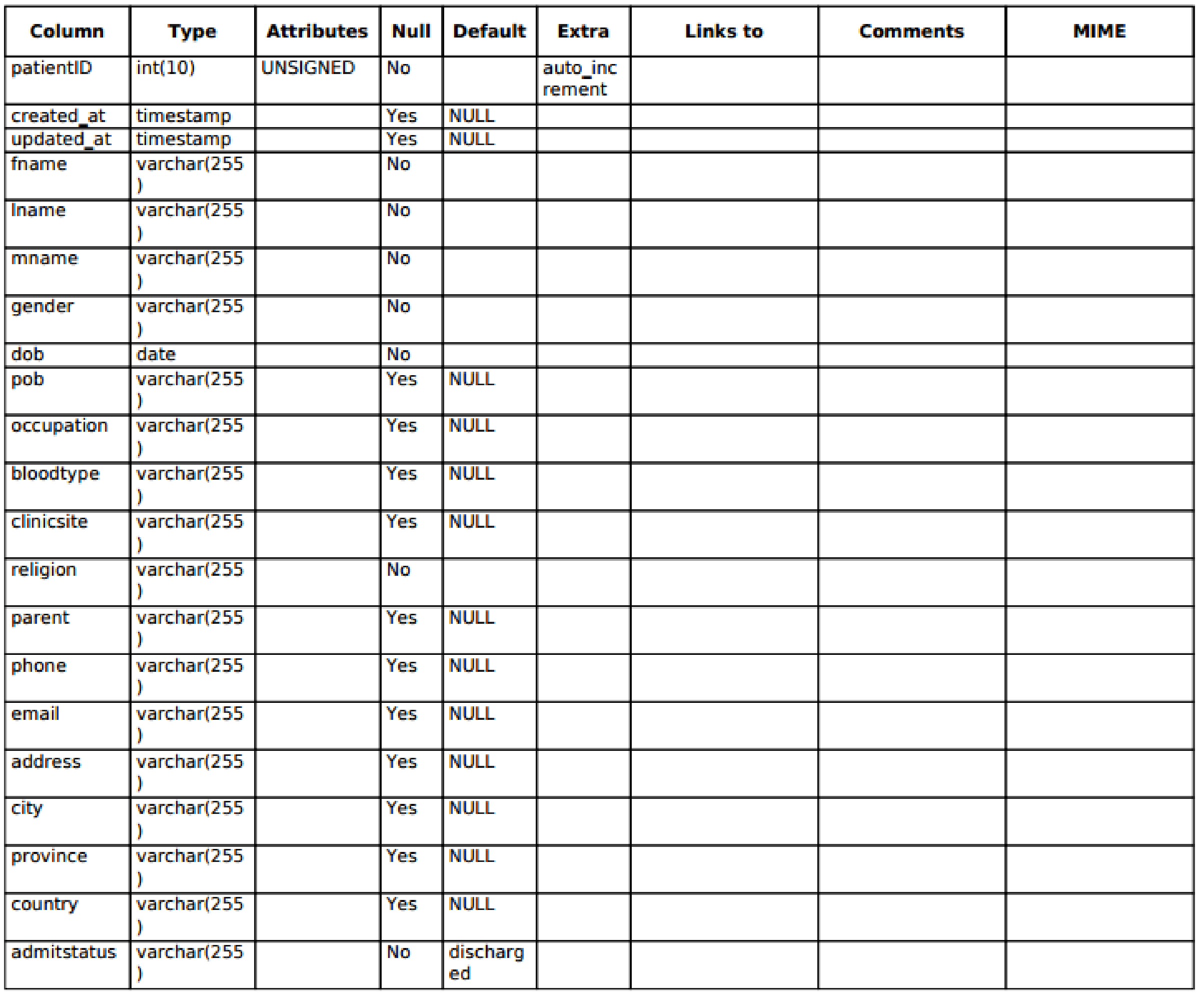
Column	Type	Attributes	Null	Default	Extra	Links to	Comments	MIME
id	int(10)	UNSIGNED	No		auto_increment			
created_at	timestamp		Yes	NULL				
updated_at	timestamp		Yes	NULL				
name	varchar(255)		No					
department	varchar(255)		No					

Figure 22 Doctors table



Column	Type	Attributes	Null	Default	Extra	Links to	Comments	MIME
detailsID	int(10)	UNSIGNED	No		auto_increment			
created_at	timestamp		Yes	NULL				
updated_at	timestamp		Yes	NULL				
patientID	int(11)		No					
history	varchar(255)		Yes	NULL				
general	varchar(255)		Yes	NULL				
appointment	varchar(255)		Yes	NULL				
medication	varchar(255)		Yes	NULL				
date	date		No					

Figure 23 Patients details table



Column	Type	Attributes	Null	Default	Extra	Links to	Comments	MIME
patientID	int(10)	UNSIGNED	No		auto_increment			
created_at	timestamp		Yes	NULL				
updated_at	timestamp		Yes	NULL				
fname	varchar(255)		No					
lname	varchar(255)		No					
mname	varchar(255)		No					
gender	varchar(255)		No					
dob	date		No					
pob	varchar(255)		Yes	NULL				
occupation	varchar(255)		Yes	NULL				
bloodtype	varchar(255)		Yes	NULL				
clinicsite	varchar(255)		Yes	NULL				
religion	varchar(255)		No					
parent	varchar(255)		Yes	NULL				
phone	varchar(255)		Yes	NULL				
email	varchar(255)		Yes	NULL				
address	varchar(255)		Yes	NULL				
city	varchar(255)		Yes	NULL				
province	varchar(255)		Yes	NULL				
country	varchar(255)		Yes	NULL				
admitstatus	varchar(255)		No	discharged				

Figure 24 Patients table