

Patient Management System

J P L Gomes

2020



Patient Management System

**A dissertation submitted for the Degree of Master of
Information Technology**

J P L Gomes

University of Colombo School of Computing

2020



Abstract

In this study an attempt is made to explore many limitations of manual and computer-based systems to manage patient information in small scale private medical clinics. Currently, many small-scale medical clinics are using a local computer-based system or manual system to manage healthcare records. In such systems it is inconvenient to access the patients' medical histories. Also, if a patient moves to a new doctor, that doctor cannot access their previous records instantly. This is a critical issue in emergency situations. A cloud-based patient management system will address these problems. The objective of such an application is to store data in a centralized cloud-based system. Doctors, patients, support staff and the system administrator will be granted different levels of access. Data reliability, data security, efficiency and overall customer experience is also addressed. Implementation of this system was done to host as a cloud-based system adhering to client-server architecture. Spring boot was used for the server component and Angular was used for the client component. The application was made more portable and easier to deploy by containerization using docker. As patient information security plays a critical role in patient management system, a special authentication and authorization system was introduced to the system using Spring security, JWT tokens and REDIS server (token blacklist). Implemented Patient Management System is streamlines processes within a medical clinic to support doctor, support staff and patients to ease their work. Therefore it has a positive impact on healthcare providing small scale medical clinic. It allows smooth interactions with the patient care by automating daily operations. This PMS produce a good chance of an effective and efficient business model for healthcare practitioners.

Declaration

The thesis is my original work and has not been submitted previously for a degree at this or any other university/institute.

To the best of my knowledge it does not contain any material published or written by another person, except as acknowledged in the text.

Student Name: J P L Gomes

Registration Number: 2017/MIT/019

Index Number: 17550196

Signature:



Date: 20.11.2020

This is to certify that this thesis is based on the work of

~~Mr.~~/Ms. J.P.L Gomes

under my supervision. The thesis has been prepared according to the format stipulated and is of acceptable standard.

Certified by:

Supervisor Name: Malik Silva

Signature:

Date: 20.11.2020

Acknowledgements

Foremost, I would like to express my sincere gratitude to my supervisor Mr. Malik Silva, Senior Lecturer of University of Colombo School of Computing for his valuable support and guidance.

A special thanks goes to Dr. Megha Pemasinghe and his staff for providing me with this chance to implement this Patient Management System – GPMate, according to their requirements, providing necessary information amidst their busy work schedule.

Also, I would like to give my sincere gratitude for the academic staff members of University of Colombo School Computing for all the knowledge and support given throughout the MIT degree program which immensely helped to complete the dissertation and the final individual project.

Last but not least, I would like to express my thanks to my loving family: especially my husband for introducing me to this topic as well as for the huge support given throughout, my parents for their support and encouragement throughout the whole time and especially my little son for tolerating me when I could not attend to him, when he needed me mostly. I will be grateful forever for your love.

Table of contents

Abstract.....	i
Declaration.....	ii
Acknowledgements.....	iii
Table of contents.....	iv
List of Figures	vi
List of Tables	vii
List of Acronyms	viii
Chapter 1 : Introduction.....	1
1.1 Motivation.....	1
1.2 Scope and objectives.....	3
1.3 Structure of thesis.....	3
Chapter 2 : Review of similar systems and technologies	5
2.1 Introduction	5
2.2 Related literature on automated patient management	5
2.3 Short code standards for recording diagnosis	7
2.4 Cloud computing and healthcare data security	8
Chapter 3 : Methodology	9
3.1 Introduction.....	9
3.2 Software development process.....	9
3.3 Requirements specification	11
3.3.1 Functional requirements.....	11
3.3.2 Non-functional requirements	12
3.4 Design.....	13
3.4.1 Designing phase	13
3.4.2 Workflow of the design	14
3.4.3 Use case diagram	15
3.4.4 Mind map	17
3.4.5 Class diagram.....	17
3.4.6 User interfaces	19
Chapter 4 : Implementation	24
4.1 Introduction.....	24
4.2 Version controlling system	24
4.3 Backend implementation	24

4.4 Front end implementation	28
4.5 Hosting on cloud	29
4.6 System security	30
Chapter 5 : Testing and evaluation	32
5.1 Introduction.....	32
5.2 Testing Strategy	32
5.3 Test Plan.....	32
5.4 Test Cases	33
Chapter 6 : Conclusion and future work	37
6.1 Introduction.....	37
6.2 Problems encountered.....	37
6.3 Lessons learnt.....	38
6.4 Future enhancements	38
References.....	40
Appendices.....	42
Source code	42
User manual	42
API Documentation	43
SMS notifications examples	44
Sample report generated	45

List of Figures

Figure 3.1 - Agile software life cycle	9
Figure 3.2 - Agile prototyping process model	10
Figure 3.3 – Overview of web-based application	11
Figure 3.4 – Usecase diagram for GPMate	16
Figure 3.5 – Mind map of GPMate	17
Figure 3.6 – Class diagram	18
Figure 3.7 – Login user interface	19
Figure 3.8 – UI when logged in as the doctor	19
Figure 3.9 – UI when logged in as the system admin	20
Figure 3.10 – UI when logged in as the staff assistant	20
Figure 3.11 – UI for managing units	21
Figure 3.12 – UI for placing a patient in the queue	21
Figure 3.13 – Patient information view	22
Figure 3.14 – Patient dashboard	22
Figure 3.15 - Report generation UI	22
Figure 3.16 – Session management UI	23
Figure 3.17 – Session filter UI	23
Figure 4.1 – Overview of REST API	25
Figure 4.2 – Create patient API	27
Figure 4.3 – Create medical alert API	27
Figure 4.4 – Create user API	27
Figure 4.5 – MVC architecture	28
Figure 4.6 – Cloud hosting infrastructure	29
Figure 4.7 – Example for JSON web token	31
Figure 5.1- User login with valid username and password combination	34
Figure 5.2 - User login with invalid username and password combination	34
Figure 5.3 – Patient detail search with wrong identifier value	35
Figure 5.4 - Patient detail search with correct identifier type and value	36

List of Tables

Table 4-1 – The APPI endpoints.....	26
Table 5-1 – Test case structure	33
Table 5-2 – Test cases for login function.....	33
Table 5-3 – Test cases for search patient details.....	35

List of Acronyms

API: Application Programming Interface

CKM: Clinical Knowledge Management

CPABE: Cipher text-Policy Attribute-Based Encryption

ECDSA: Elliptic Curve Digital Signature Algorithm

EHR: Electronic Health Record

EMR: Electronic Medical Report

FHIR: Fast Healthcare Interoperability Resources

GP: General practitioner

GUI: Graphical User Interface

HCI: Human Computer Interaction

HMAC: Hash-based Message Authentication Code

HTML: Hyper Text Markup Language

HTTP: Hyper Text Transfer Protocol

JSON: Java Script Object Notation

JWT: JSON Web Token

LTS: Long Term Support

MVC: Model-View-Controller

NIC: National Identity Card

ONC: The Office of the National Coordinator for Health Information Technology

ORM: Object/Relational Mapping

PMS: Patient Management System

REST: Representational State Transfer

RSA: Rivest–Shamir–Adleman - one of the first public-key cryptosystems

WHO: World health Organization

XML: Extensible Markup Language

Chapter 1 : Introduction

1.1 Motivation

Patient management refers to a software tool that streamlines processes within a medical practice or hospital or refers to an entire system of care involving both patient and practice[1].

Healthcare has become a precious commodity around the world and Srilankan is no exception. If we consider the Sri Lankan healthcare system, people are more likely to keep a general practitioner (GP) for their health-related needs. This GP or as we call them “The family doctor” plays a vital role in our day-to-day lives.

According to the central bank annual report 2017, patient-doctor ratio is 1800 patients per one doctor in Sri Lanka. It can be a larger number of patients in urban areas. Therefore, memorizing every patient, each visit, diagnosis history, prescriptions, allergies, other health conditions, etc are almost impossible for doctors in their private clinics. On the other hand, the patient will not be able to tell every detail due to their illnesses or other various factors during their visits.

For example, take a scenario in which a patient comes requesting medical assistance regarding a bacterial infection. In this situation, most of the doctors would prescribe an antibiotic. But the patient might be also suffering from gastritis and he might forget to tell the doctor that he has such an illness on that visit. The doctor also does not have any records of the patient's previous diagnoses and he has not prescribed any medicine for gastritis. Thus, a way to grab the doctor's attention on patient's such conditions would prevent any side effects which can occur from his medication.

In some occasions, the regular GP is unable to visit, and they send some other GP to cover him up on that day. On such situations, the patients who are making their secondary visits will face difficulties as they must describe their health conditions all over again. If there is a system that could retain the previous diagnosis, it will help the substitute doctor and the patients immensely. PMS also could provide features to read, update and store important patient information in the system which can be access at future encounters. For the medical practitioners’ providing patients a quality and timely treatment is their main objective. A PMS guarantees that in an unobstructed way.

In a medical clinic, patients are the customers, and satisfying their needs and providing a better service is important. Doctors are also considering many aspects to improve and expand patient care and manage smooth relation between patients and their visits. Most often support staff of a medical clinic needs to automate their tasks such as store information, administrative tasks. Patient management system will be able to perform all those tasks, saving money and time too.

Doctor's assistant is responsible for the registration of new patients and accepting visits and placing them in the queue. The delays, paperwork, effort, which has to be put into managing patient records can be avoided, by carrying out those functions in whole or in part in an electronic environment. The doctor is provided with short codes for recording diagnosis and prescriptions without taking much time. Those codes could be mapped to internationally accepted codes associated with open EHR or FHIR standards [2] (healthcare informatics standards) as a future enhancement. These records will be stored in an application built on the cloud. Therefore, the same patient's information can be shared with other GPs' if they use the same proposed system. Through a role-based access control login, the system offers an easy way to access patient history, managing solid confidentiality of the records. For documenting relevant medical history of a patient, documents from visits to other clinics also can be allowed for uploading. To summarize the features of Patient management system, it can integrate, patient information, diagnoses, prescriptions, billing records, appointment history and more. It also automates tasks like scheduling appointments.

Although the current manual process is fulfilling the daily requirements of the medical clinic, it is a difficult task to store, manage and maintain daily records of the medical clinic such as patient information, prescribed medicines and schedules resulting wastage of man power, time and money. Following tasks should be considered when implementing the automated patient management system.

- Cost of software development, implement and maintenance deployment is high.
- Because of both staff and patients are familiar with the manual process, it is difficult to migrate from manual process to automated system.
- Poor computer literacy among the staff is also a challenging problem that has to be addressed through a rich human computer interaction solution.
- Large number of patient count visiting the medical clinic makes the migrating process difficult. First time registration and data entry will take some time.

1.2 Scope and objectives

Design and implement a cloud based patient management system that streamlines the processes within a medical clinic to support doctor, support staff and patients to ease their work with efficient and effective output by providing best solutions complying with latest international medical standards with effective combination in to Human-computer interaction(HCI) is the main objective of this project.

Scope of the project can be outlined as follows:

- Patient registration
- Patient information management
- Patient scheduling
- Track patient encounter (Electronic Medical Record - EMR)
- Notification alerts to doctor about patient's special health conditions
- Generate reports, prescriptions
- SMS notifications for cancelled sessions.

1.3 Structure of thesis

CHAPTER 02: LITERATURE REVIEW

An overview of most relevant academic and industrial literature in the areas of patient management, cloud computing, short code standards for recording diagnosis and healthcare data security.

CHAPTER 03: PROBLEM ANALYSIS

This chapter will demonstrate the software engineer processes used to analyses the problem and why that method was adopted.

CHAPTER 04: DESIGN

An overview of the designing the system is described here. Workflow of the design, higher level diagrams, object-oriented diagrams which adopted to model the design also included.

CHAPTER 05: IMPLEMENTATION

Steps carried out in the implementation phase of the GPMate is described in this chapter. It includes details about overall implementation of the proposed GPMate, and the selection of technology tools for the implementation and explanations for selecting those tools.

CHAPTER 06: EVALUATION AND TESTING

This chapter discusses how the testing and evaluation was carried out to measure the level of success related to the project.

CHAPTER 07: CONCLUSION AND FUTURE WORK

Whether the project objectives were satisfied and if not, the reasons for them are stated in this chapter. Lessons learnt during the project should also be extended upon. Difficulties raised which are beyond control, that have affected the progress are also stated here. Conclusion of the work indicating a summary of the results of the project and future enhancements which can be carried out.

Chapter 2 : Review of similar systems and technologies

2.1 Introduction

The purpose of this chapter is to review past efforts that relate to proposed PMS, with their limitations or weaknesses, which make them insufficient for the required solution. There are several researches and implementations done in the areas of Patient Management, Hospital Management, Electronic medical recording. Patient Management is a major area in healthcare. It also a critical indicator in the quality of healthcare service. The studies cited in this chapter highlight the key features in a Patient Management System. Also, an overview of short code standards for recording diagnosis, cloud computing and healthcare data security is provided.

2.2 Related literature on automated patient management

HHIMS (Hospital Health Information Management System) [3]

Initially in 2005 HHIMS was created on software built by the WHO country office for Sri Lanka. It was further developed in 2006-2009 in a project implemented by the Austrian/ Swiss Red Cross. After that in 2010 this HHIMS was carried out for the regional director of health services, Kegalle, by NetCom Technologies.

HHIMS is a medical record software developed for use in Sri Lankan hospitals. It stores the following information.

- Patient's clinical details during out-patient visits
- Clinic consultations
- Ward admissions

And it is designed to replace the paper records. Medical information of a patient can be entered to the system database directly as the patient is examined. Laboratory tests, prescriptions can be managed through the computer network and processed without the need of paper records. When a patient visits the hospital again or admitted to the hospital, an overview of all the patient's clinical details can be displayed on one single screen.

OpenEMR [4]

A most popular free and open source Hospital Management Software solution available today is OpenEMR. It satisfies the hospital management functionalities and electronic record management of health clinics of all scales.

Some major features of OpenEMR is listed below.

- Freely available, Free to download, use, modify, and upgrade, fFree documentation
- ONC Certified as a cComplete EHR
- Patient Demographics
- Fully Customizable
- Patient Scheduling
- Electronic Medical Records
- Patient Reports
- Electronic digital document management
- Voice recognition ready (MS Windows Operating Systems)
- Electronic Syndrome Surveillance reporting
- Clinic Messaging
- Send and Receive Medical Records via Direct Messaging
- Dated Reminders
- Online drug search
- Track patient prescriptions and medications
- Medical Billing
- Medical claim management interface
- Physician and patient Reminders
- Modern User Interface
- Scheduling and Appointments
- Secure Messaging and Chat
- Online Payments
- New Patient Registration
- Prescriptions and Drug Dispensing
- Supports use of multiple languages within the same clinic
- Support for Role Based Menus and Custom Menus

- Ability to Encrypt Patient Documents

Existing systems within the country are only shared within separated hospitals or districts. It will be resulted in high construction costs and waste of resources. Especially for small scale hospitals and private medical clinics which are unable to set up an independent information platform.

Systems like OpenEMR need lots of records to be entered. But in the area that we concerned, a GP with a small medical clinic has not much time to input bulk records. Some GPs do not have assistants to enter information. Therefore, the system must be very simple and easy to use. It should be very convenient to be used by the Doctor himself. Also, in today's world as we are focusing on customer satisfaction in every business, the system should provide a better service to the patient also.

Based on the concepts of cloud computing, a solution is proposed to implement in this project. Therefore, the high-end processing and information sharing is available in the 'cloud'.

2.3 Short code standards for recording diagnosis

OpenEHR

openEHR is an open standard specification in health informatics that describes the management and storage, retrieval and exchange of health-related data in electronic health records (EHRs). In openEHR, all health-related data for a person is stored in a "one lifetime", vendor-independent, person-centred EHR.

This openEHR specifications are managed by the openEHR foundation. These are based on European and Australian research and development for 15 years. This openEHR specifications include information and service models for the EHR, demographics, clinical workflow and archetypes. [5]

One of the outcomes of openEHR modelling approach is the open development of archetypes, templates and terminology subsets to represent health data. Due to the open nature of openEHR, these structures are publicly available to be used and implemented in health information systems. Community users are able to share, discuss and approve these structures in a collaborative repository known as the Clinical Knowledge Manager (CKM). Some currently used openEHR CKMs:

- openEHR Clinical Knowledge Manager

- NEHTA Clinical Knowledge Manager
- UK Clinical Knowledge Manager

2.4 Cloud computing and healthcare data security

According to Elmogazy, Huda and Bamasak, Omaima (2013), healthcare data has strict security requirements for confidentiality, availability to authorized users, and traceability of access. In that paper they propose a solution for healthcare industry which is provided by a cloud, that will help in protecting patients' data they host, focusing on specific cloud computing healthcare security concerns and how homomorphic encryption with splitting key and key delegation can help in meeting healthcare requirements. The suggested technique is based on FHE algorithm with key delegation to ensure data confidentiality, authentication, integrity, and availability in a multilevel hierarchical order. This will allow the healthcare practitioner to ignore any access rule in any order, especially in a medical research environment [6].

As Suhair Alshehri ; Stanislaw P. Radziszowski ; Rajendra K. Raj (2012) point out, many healthcare organizations adopt electronic health records (EHRs) and the case for cloud data storage becomes compelling for deploying EHR systems: not only is it inexpensive but it also provides flexible, wide-area mobile access increasingly needed in the modern world. In this paper they propose the use of Ciphertext-Policy Attribute-Based Encryption (CPABE) to encrypt EHRs based on healthcare providers' attributes or credentials, to decrypt EHRs. They must possess the set of attributes needed for proper access [7].

Chapter 3 : Methodology

3.1 Introduction

The goal of this project is to design and implement a simple cloud-based web application, that can be accessed from any device, for ease of access. An interactive GUI will be provided to facilitate accessing the patients' records online. The patient records can be accessed from any medical facility which adopts the developed system. Therefore, the patient can visit any of the doctors to get a consultation if needed. The framework has been built to accommodate further modifications, adding additional features without disturbing the existing system.

3.2 Software development process

Agile development process was used as the software development process. Following figure 3.1 illustrates the Agile software development life cycle.

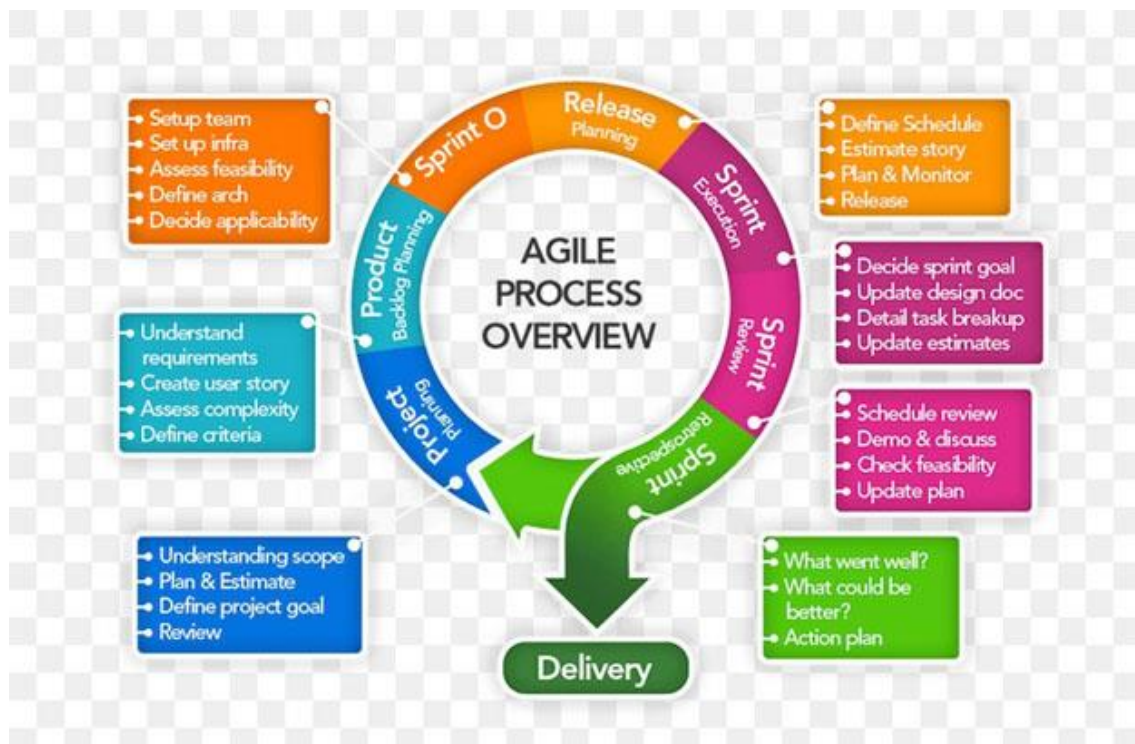


Figure 3.1 Agile software life cycle

Basically, Agile is a model comprehending process clusters run in sequence within a definite period in iteration. A feedback is looped to the customer for validating the solution. It can be found in especially in software development industry. The context has been adopted successfully. Likewise, this method can be used to patient management system development projects in healthcare [8].

Agile methodology was selected for this project because of the following listed reasons;

- Most of the solution, but not all is identified. The goal needs to be specific and measurable.
- There may be several deviations from unidentified ranges of the solution.
- Allow stakeholders to involve in developing the solution.
- Evaluating and validating the influence of alternatives.

For managing the impact of changing the system incremental prototype is a good approach. Incremental prototype paired with Agile aids to cover changing requirements, benefits and outputs more rapidly. The following figure 3.2 illustrates a higher-level approach of accomplishing this project.

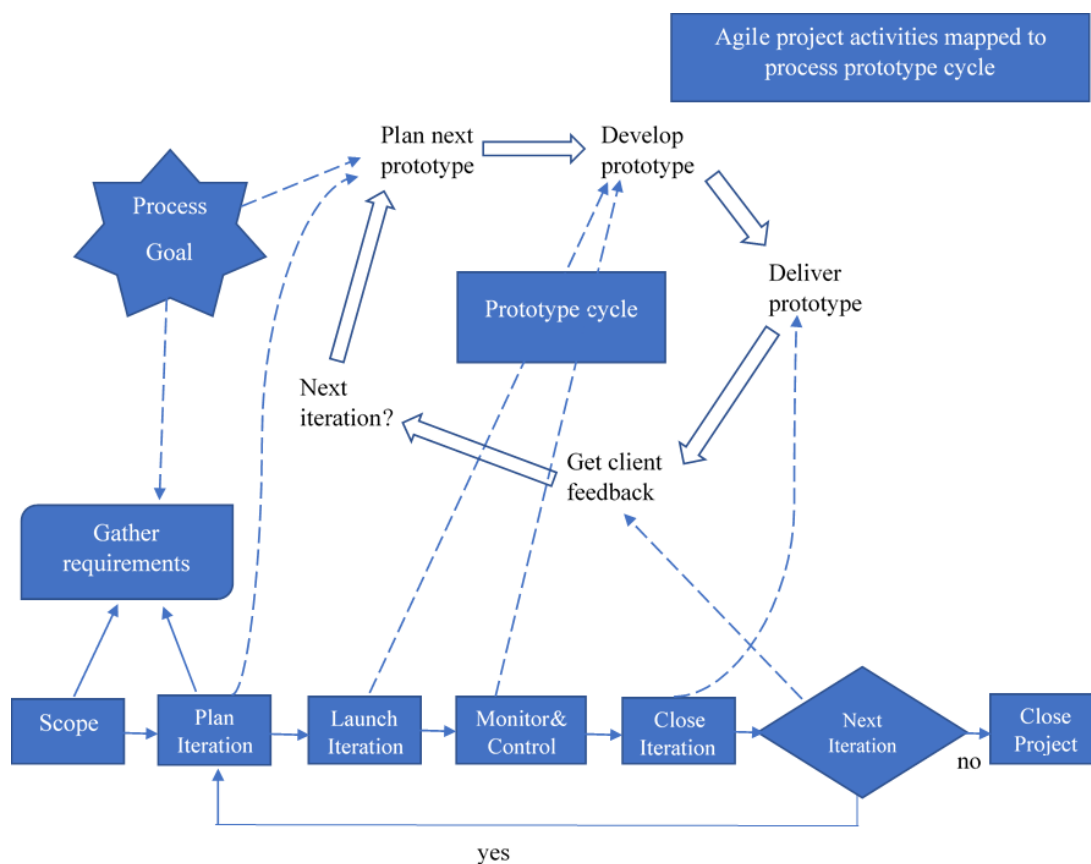


Figure 3.2 - Agile prototyping process model

The PMS is to be deployed as a web-based application (micron/progressive web application). It is a type of software and there is a huge increase in popularity than the desktop applications. Portability is the most significant advantage over traditional desktop applications. Not only that web-based applications have many number of advantages over desktop applications. It let a user to interact with a application server through a web browser interface, users can use the software without installing additional softwares and the application software is not depend on the operating system of the device. Therefore, it is not necessary to code different versions for different operating systems. Following figure 3.3 presents an overview of web based application.

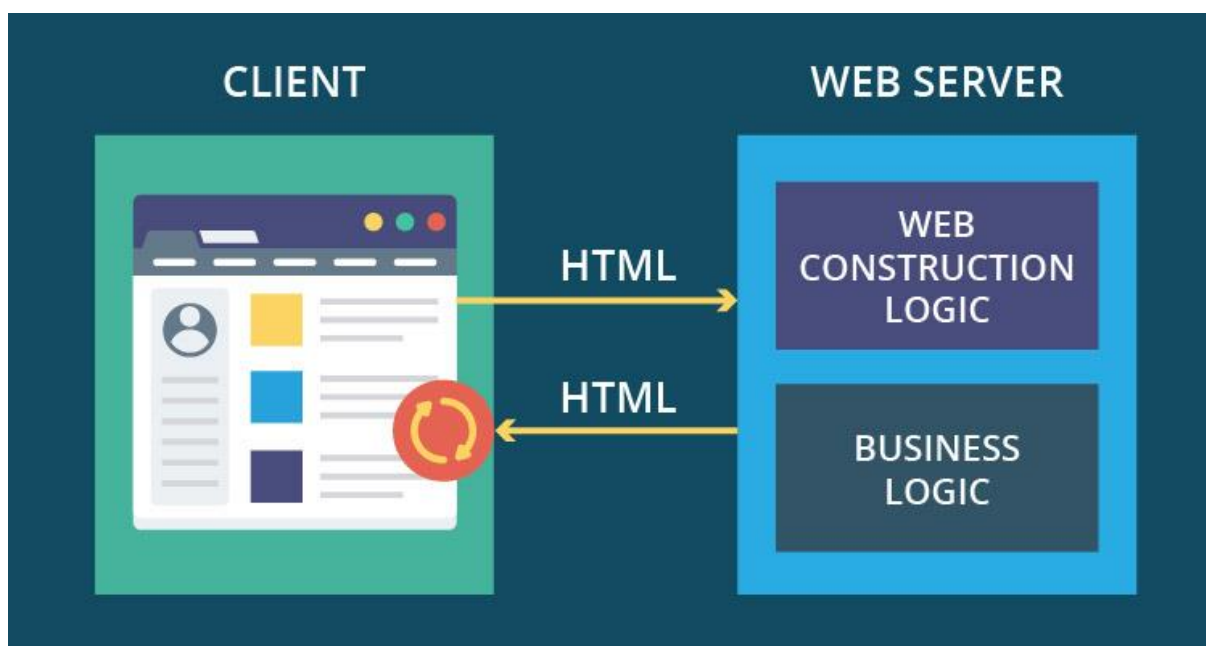


Figure 3.3 – Overview of web-based application

3.3 Requirements specification

3.3.1 Functional requirements

- There should be users who are authorized and authenticated based on their roles for the system.
- The system should allow users to login with their usernames and passwords.
- Authorized staff member should be able to perform CRUD operations on staff member details.
- Authorized staff member should be able to register a patient to the system.

(When a patient arrives for the first time, that patient is given an application form to fill, to gather the basic information. Then the authorized staff member registers the patient for the system.)

- Authorized staff member should be able to do CRUD operations on patient details.
- According to the privileges granted for staff members, they should be able to create a session for patients.
- Patients should be given a queue number according to the order of arrival and placed in a queue.
- Doctor should be able to start the session that has been created.
- Doctor should be able to view the list of patients.
- Doctor should be able to select any patient from the list according to their need. i.e. in an emergency.
- Doctor should be able to edit, update patients' health record information.
- Doctor should be able to access the patients' medical history.
- Medical history should only be accessed after user authentication.
- Doctor should get an alert if a patient has any special health condition such as an allergy or chronic condition.
- Doctors should be able to upload medical test results of a patient to his/her account.
- Doctor must be able to record diagnosis and record prescriptions for each patient.
- Support staff should be able to view records entered by a doctor relevant to payments of a patient and generate bills.
- With proper authorization, doctors from all units should be able to access the patients' medical records, reports and past prescriptions.

3.3.2 Non-functional requirements

System properties such as reliability, response time, user friendliness, etc. can be state as non-functional requirements. They may define constraints on the system also.

systems.

Usability

- The system should be accessible on almost all OS platforms.

- The system should be accessible from all types of devices including mobile devices.
- The system should provide help menu for the ease of accessibility to users.
- The system should be user-friendly for people of all ages and different ability levels.

Security

- The system should have different levels of access to different people to prevent unauthorized usage of sensitive information.
- The system should not grant access to anyone unless the correct username-password combination has been entered.
- The system should only permit 3 failed attempts to login. After that, the account should be suspended, and the user will have to contact the PMS administrator and pass a security challenge in order to unlock the account.

Availability

- The system should always be available 24*7 .
- The system should not fail more than 3 times a year.
- When the system fails, it should not take more than a day to start back-up to avoid additional problems.

Performance

- Response time for any activity performed by the user must not exceed 20 seconds.

3.4 Design

3.4.1 Designing phase

Designing phase of a software development life cycle generally involves problem solving and planning the solution. What system will do and how the system will work is the output of this phase.

Requirement gathering from the private medical clinic is the first part of the design phase. Then the requirements are maps into design. The design defines the components, their behaviors

and user interfaces. The design documentation puts forward a plan to implement the identified requirements. A good design has several key factors.

- Accuracy - the extent to which the system's outputs are appropriate to satisfy their intended use.
- Reliability – the probability in free of failure processing of the software for a given period in a specified environment.
- Once the system is functioning, reliability is the probability of failure-free software operation for a specified period in a specified environment.
- Usability – the degree to which the software can be used.
- Interoperability - the ability of the software to exchange and make use of information.
- Security – to continue the implemented functions of the software properly, there should be a guard to the software against possible threats.
- Maintainability - ease of updating the software to satisfy new requirements after it has been developed.
- Efficiency - the ratio of the useful work performed by the system to the resources needed.

3.4.2 Workflow of the design

The system is designed as described herewith. When a user enters the URL of the system on a web browser, the user will be redirect to the login UI of the system. Upon entering both username and password credentials successfully the user will be able to access the system. According to the assigned role by the system to the user, a view will be loaded respectively which containing the functions that user could be handled. For an example if the logged in user's role is the doctor then the UI with side menu which contains functions that the doctor can carried out through the system will be loaded.

Likewise, system admin and other staff members will get a view according to their privileges they have. Patient portal was not considered when implementing this project because it will lengthen the scope of the project. But the user type 'patient' was created in design phase for consider the patient registration, edit and update patient profiles, upload test results by patients themselves as future enhancement.

Authorized staff member can register a patient via “Patient Registration” menu item. Patient identifier will be based on their NIC, driving license, passport or a temporary ID generated by the system in case the patient is not able to provide any of the mentioned.

A channeling session could be also created by the authorized staff member. It will store many items related to the channeling session. Then the patients will be placed in a queue according to the order of their arrival within that sessions. Channel session can be pre planned. When the doctor arrives and log in to the system, he could see a list of patients who are in the queue. Then he could start the session. The doctor can select any patient from the queue if an emergency patient visits the clinic.

The doctor can view the medical history of the relevant patient. After the examination doctor can record the diagnosis, any vital signs and enter prescriptions. If the relevant patient has a special health condition recorded previously that will notify to the doctor by the system. Doctor can enter medical test results of the patient. When the examination and the recording is over the doctor must end the visit of that patient. Then the doctor will direct to the next patient in the queue. At the end of the day doctor or staff assistant should end the sessions.

Distinct list of patients visited during a given period of time also can be generate as a report from this system. Considering the current pandemic situation of the country this kind of report is very important. From that report all the necessary details about patients visited during the given period can be obtained.

SMS Gateway is integrated for the system for notify patients when the doctor cancelled the planned channeling sessions due to unavoidable circumstances.

3.4.3 Use case diagram

A use case diagram is used to represent the users’ interaction with the system. It demonstrates the relationships of the user with different use cases. It is often accompanied by many other diagrams. From this diagram different types of users and different use cases of the system can be recognized. Following figure 3.4 represent the use case diagram of GPMate system.

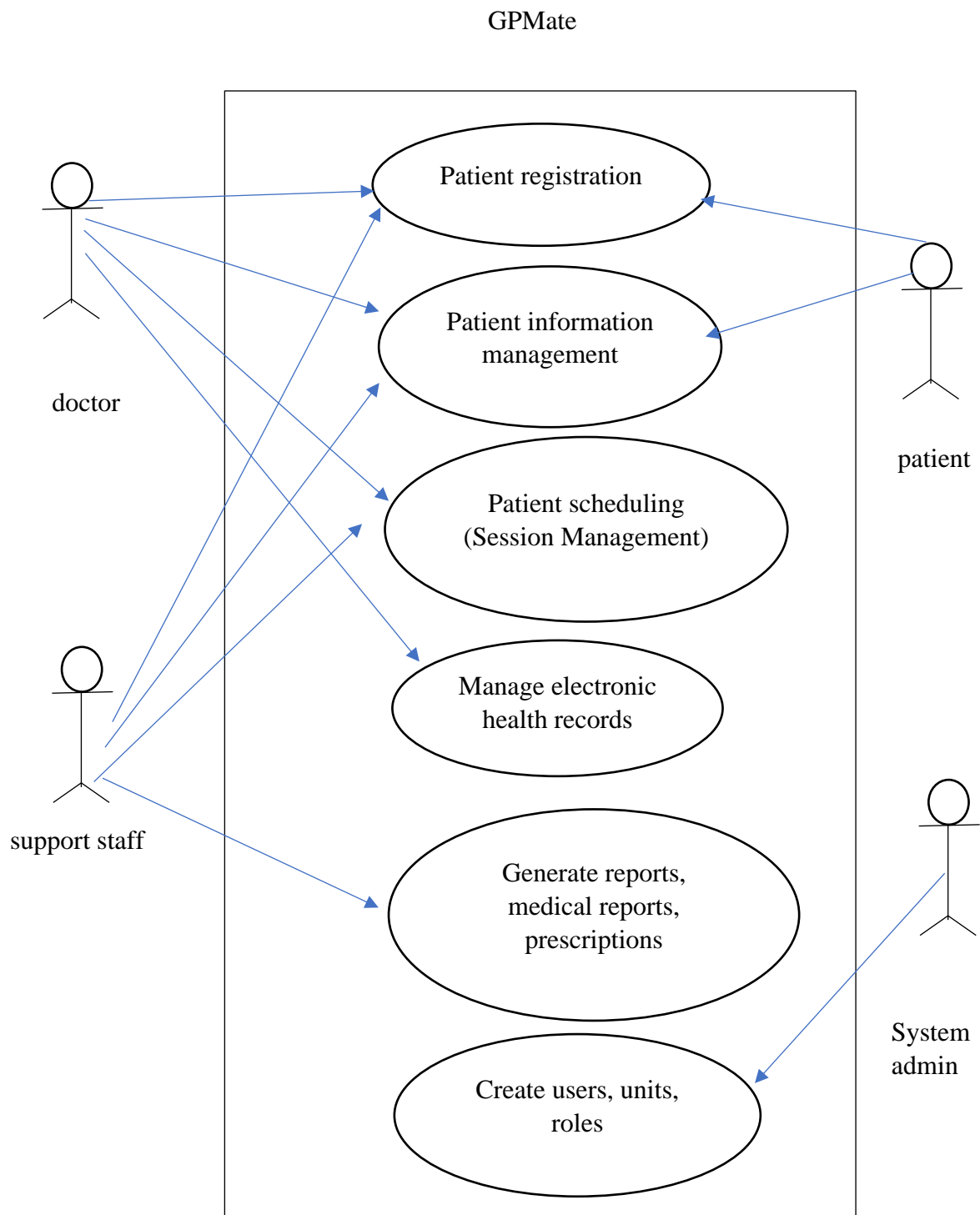


Figure 3.4 – Usecase diagram for GPMate

3.4.4 Mind map

Following mind map, figure 3.5 represent the visually organized information of the proposed Patient Management System, GP Mate. This mind map shows tasks, words, items linked to and arranged around GP Mate. This allows to build an in-built framework around GP Mate and it is hierarchical.

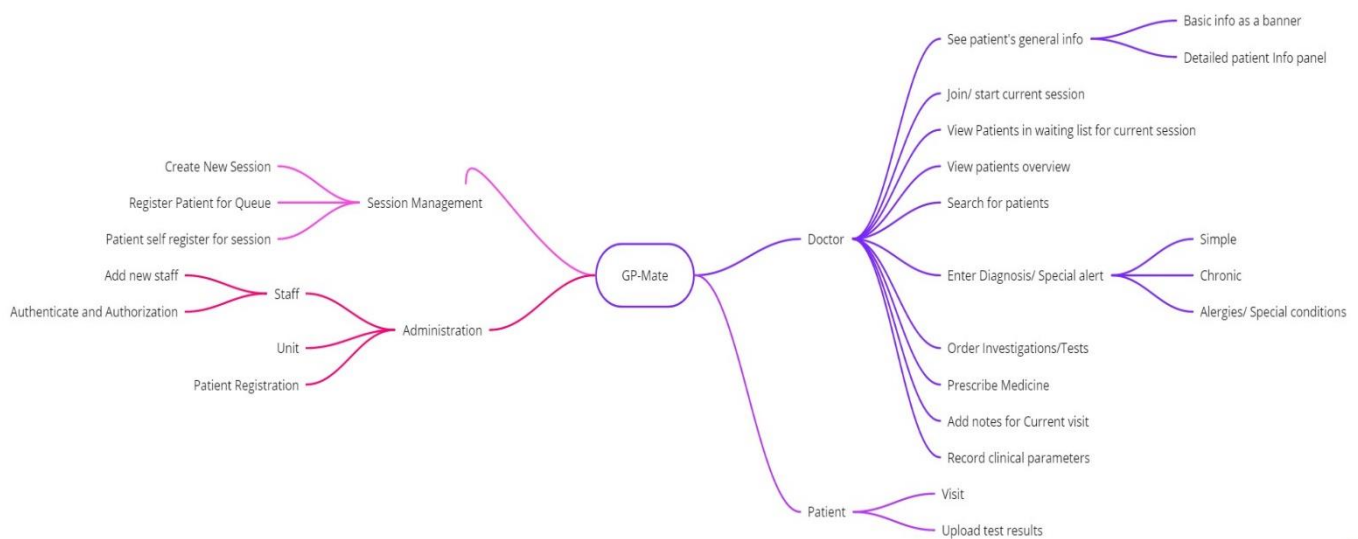


Figure 3.5 – Mind map of GPMate

3.4.5 Class diagram

The main principle of object-oriented modeling is the class diagram. It can be used to build the conceptual modeling structure of the application and for data modeling. The detailed modeling class diagrams can also be used for converting the models into programming code. Following figure 3.6 is the complete class diagram modeled for this project.

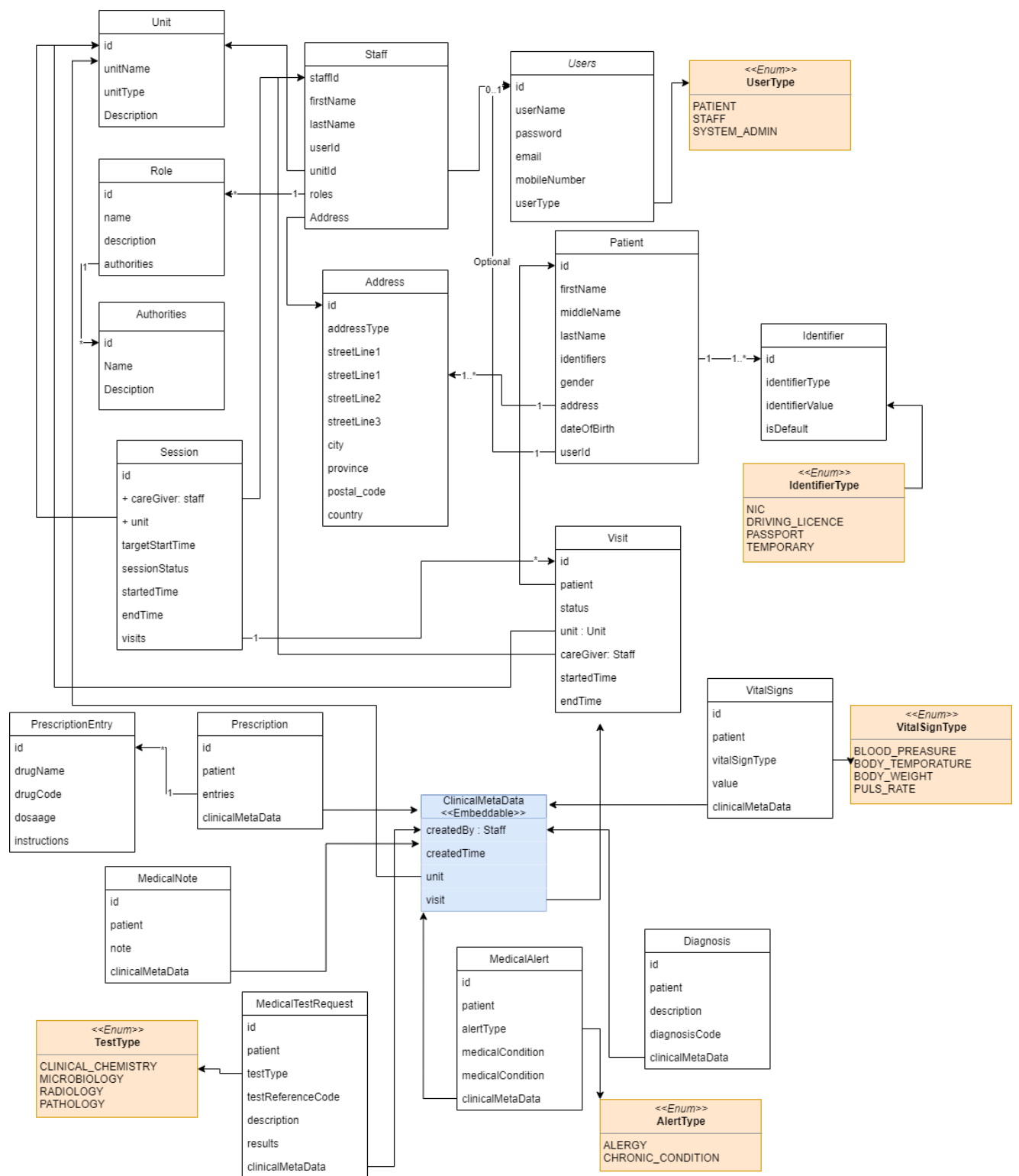
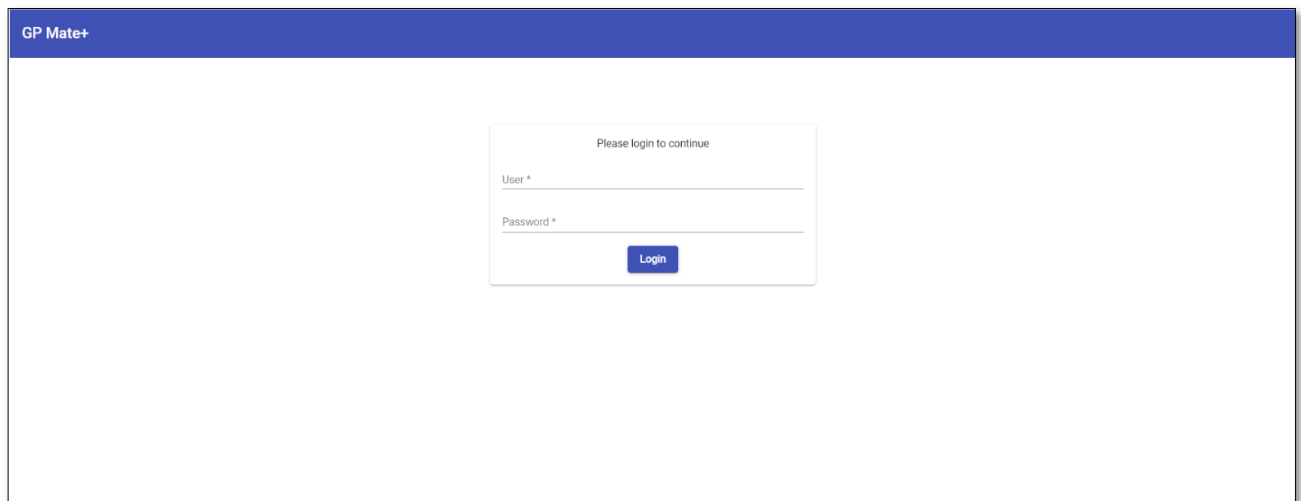


Figure 3.6 – Class diagram

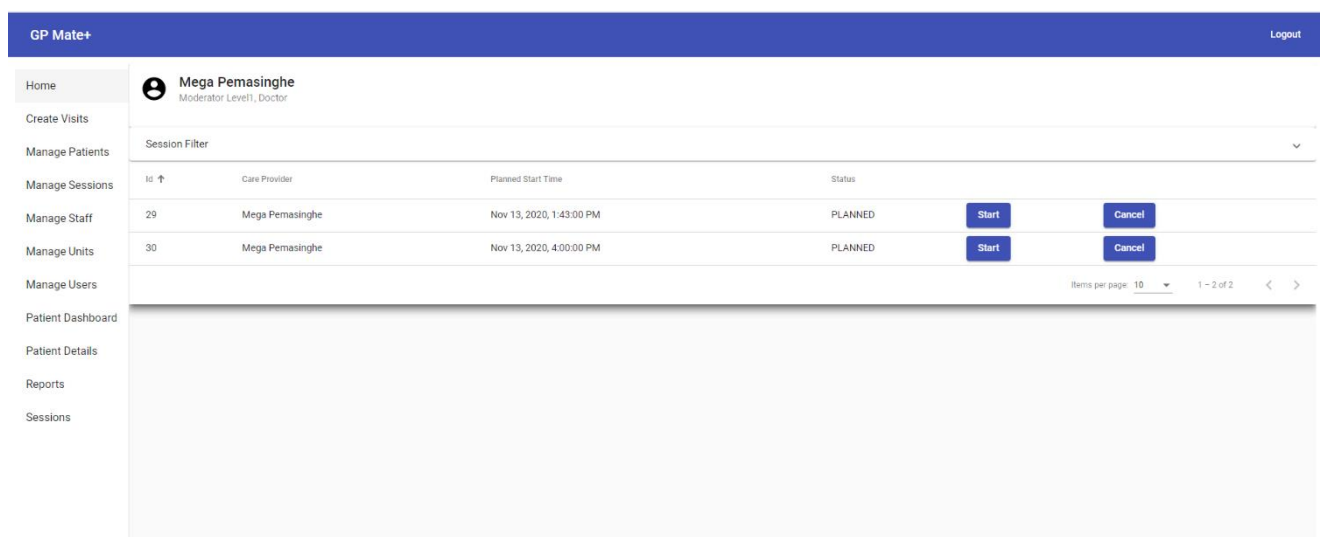
3.4.6 User interfaces

Following figures 3.7, 3.8, 3.9, 3.10 present few user interfaces designed to PMS. Different user interfaces for different functionalities were designed. When a user logged in to the GPMate the right panel of the UI will display a menu which contain the functionalities that user is authorized to carry out.



The image shows the login interface for GP Mate+. It features a blue header bar with the text "GP Mate+" on the left. The main content area is white and contains a centered login form. The form has a title "Please login to continue" and two input fields: "User *" and "Password *". Below the password field is a blue "Login" button.

Figure 3.7 – Login user interface



The image shows the GP Mate+ dashboard when logged in as a doctor. The header bar is blue with "GP Mate+" on the left and "Logout" on the right. Below the header, there is a sidebar on the left with a list of menu items: Home, Create Visits, Manage Patients, Manage Sessions, Manage Staff, Manage Units, Manage Users, Patient Dashboard, Patient Details, Reports, and Sessions. The main content area displays the user's profile: "Mega Pemasinghe" with the role "Moderator Level1, Doctor". Below the profile, there is a "Session Filter" dropdown and a table of sessions. The table has columns for "Id", "Care Provider", "Planned Start Time", and "Status". There are two rows of sessions, each with "Start" and "Cancel" buttons. At the bottom right, there is a pagination control showing "Items per page: 10" and "1 - 2 of 2".

Id	Care Provider	Planned Start Time	Status		
29	Mega Pemasinghe	Nov 13, 2020, 1:43:00 PM	PLANNED	Start	Cancel
30	Mega Pemasinghe	Nov 13, 2020, 4:00:00 PM	PLANNED	Start	Cancel

Figure 3.8 – UI when logged in as the doctor

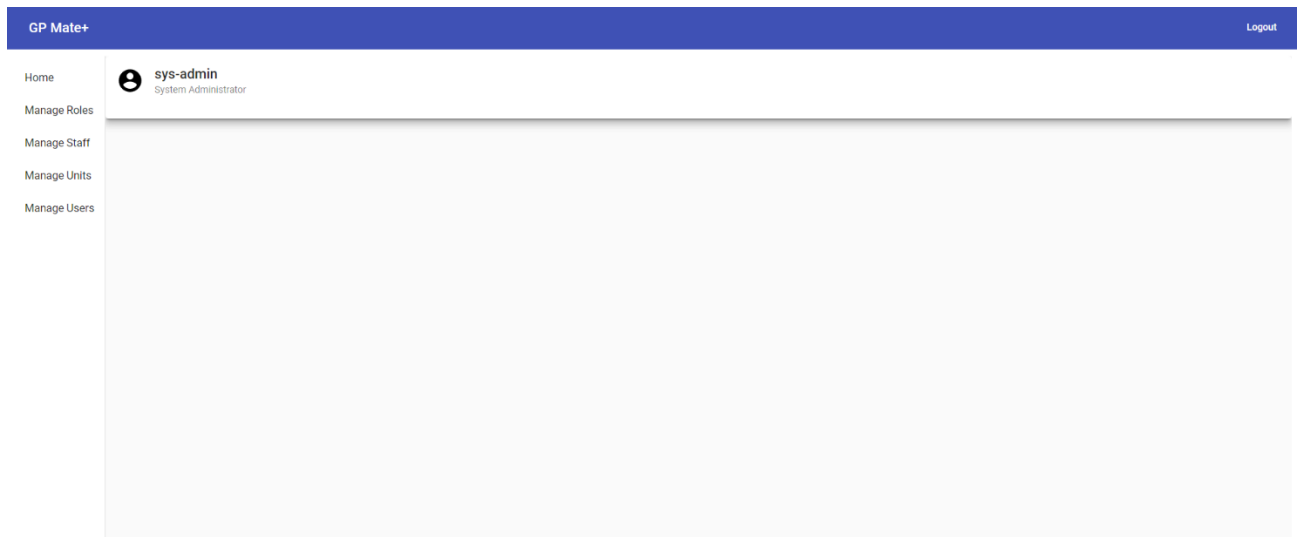


Figure 3.9 – UI when logged in as the system admin

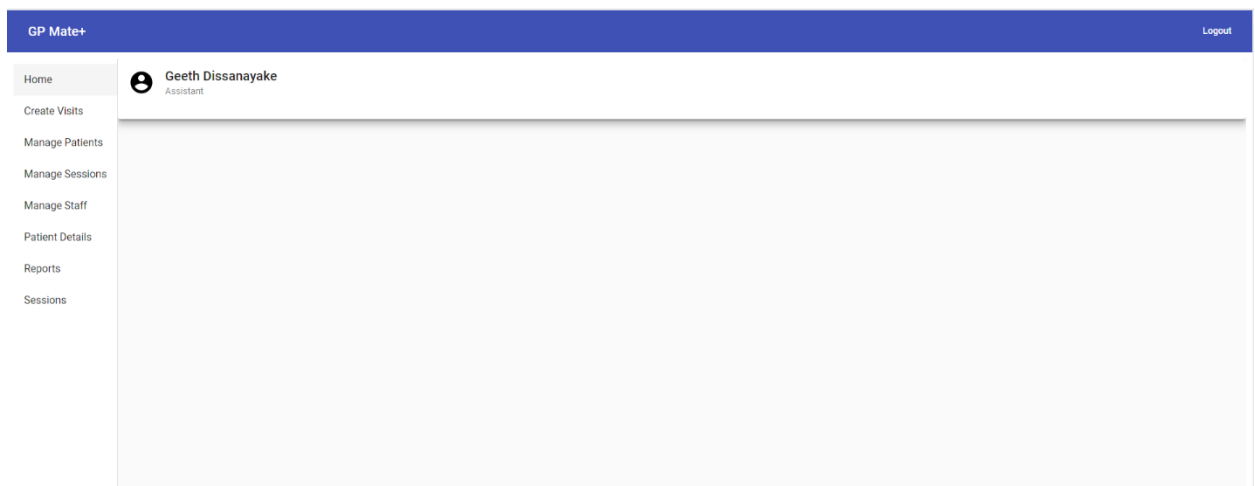


Figure 3.10 – UI when logged in as the staff assistant

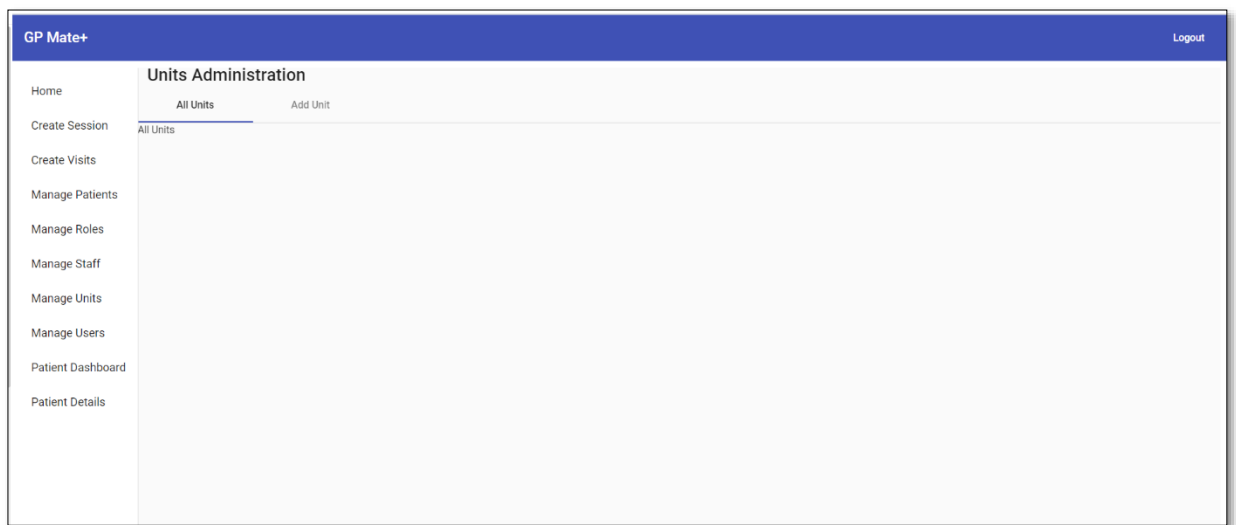


Figure 3.11 – UI for managing units

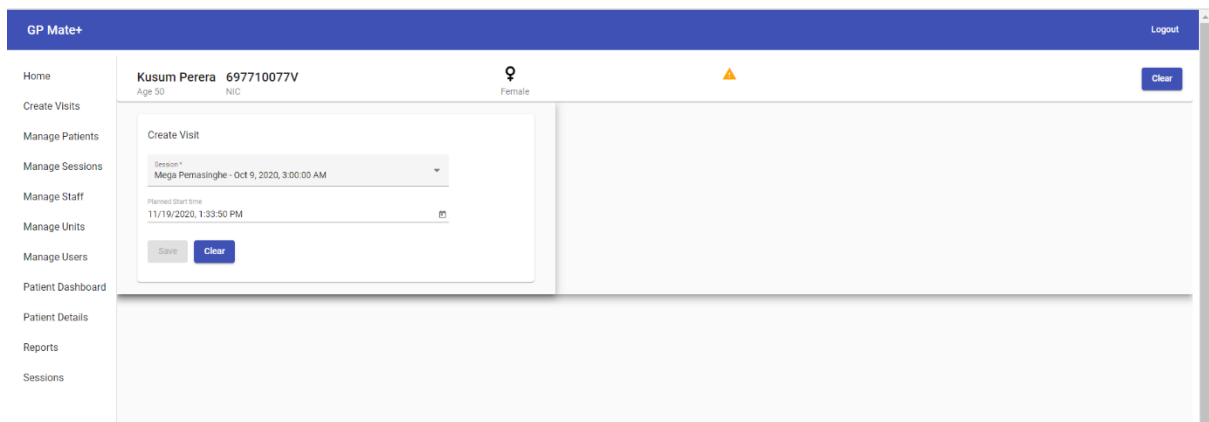


Figure 3.12 – UI for placing a patient in the queue

GP Mate+ Logout

Home
Create Visits
Manage Patients
Manage Sessions
Manage Staff
Manage Units
Manage Users
Patient Dashboard
Patient Details
Reports
Sessions

Kusum Perera 697710077V
Age 50 NIC Female Clear

First Name : Kusum
Middle Name :
Last Name : Perera
Identifier : NIC : 697710077V
Gender : FEMALE
Date of Birth : Dec 16, 1969
Contact Number : 0772345677
Address : Mirigama Edit

Figure 3.13 – Patient information view

GP Mate+ Logout

Home
Create Visits
Manage Patients
Manage Sessions
Manage Staff
Manage Units
Manage Users
Patient Dashboard
Patient Details
Reports
Sessions

Kusum Perera 697710077V
Age 50 NIC Female Clear

Patient Dashboard

Visits

Date	Care Provider	Status
Oct 9, 2020, 4:14 PM	Mega Pemasinghe	COMPLETED
Oct 8, 2020, 8:23 PM	Mega Pemasinghe	COMPLETED
Sep 4, 2020, 6:43 PM	Mega Pemasinghe	COMPLETED
Aug 9, 2020, 7:15 PM	Mega Pemasinghe	COMPLETED
Aug 8, 2020, 2:01 PM	Mega Pemasinghe	CANCELLED

Diagnoses

Recorded Date	Description
Oct 9, 2020	fever
Oct 8, 2020	cough
Sep 4, 2020	fever
Aug 9, 2020	Asthma

Medical Alerts

Type	Description
CHRONIC_CONDITION	food allergy

Vital Signs

Recorded Date	Type	Value	Unit of Measurement
May 24, 2020	BODY_WEIGHT	35	Kg

Figure 3.14 – Patient dashboard

GP Mate+ Logout

Home
Create Visits
Manage Patients
Manage Sessions
Manage Staff
Patient Details
Reports
Sessions

Distinct list of patients visited within period

From 9/3/2020

To 11/13/2020

Open Download Print

Figure 3.15 - Report generation UI

22

GP Mate+

Logout

Home

Create Visits

Manage Patients

Manage Sessions

Manage Staff

Manage Units

Manage Users

Patient Dashboard

Patient Details

Reports

Sessions

Session Administration

All Sessions

Add Session

Id ↑	Care Provider	Planned Start Time	Status	
22	Mega Pemasinghe	Oct 9, 2020, 3:00:00 AM	INPROGRESS	<div>Edit</div>
23	Mega Pemasinghe	Oct 10, 2020, 8:00:00 AM	CANCELLED	<div>Edit</div>
24	Mega Pemasinghe	Oct 9, 2020, 1:00:00 PM	CANCELLED	<div>Edit</div>
25	Mega Pemasinghe	Oct 9, 2020, 12:10:00 PM	CANCELLED	<div>Edit</div>
26	Mega Pemasinghe	Oct 9, 2020, 2:00:00 PM	CANCELLED	<div>Edit</div>
27	Mega Pemasinghe	Oct 9, 2020, 6:00:00 PM	COMPLETED	<div>Edit</div>
28	Mega Pemasinghe	Oct 9, 2020, 8:00:00 PM	PLANNED	<div>Edit</div>
29	Mega Pemasinghe	Nov 13, 2020, 1:43:00 PM	PLANNED	<div>Edit</div>
30	Mega Pemasinghe	Nov 13, 2020, 4:00:00 PM	PLANNED	<div>Edit</div>

Items per page: 10

21 - 29 of 29

<

>

Figure 3.16 – Session management UI

GP Mate+

Logout

Home

Create Visits

Manage Patients

Manage Sessions

Manage Staff

Manage Units

Manage Users

Patient Dashboard

Patient Details

Reports

Sessions

Statuses *
PLANNED, INPROGRESS

From *
11/19/2020

To *
11/19/2020

Id ↑

Care Provider

Planned Start Time

Status

Items per page: 10

0 of 0

<

>

Figure 3.17 – Session filter UI

Chapter 4 : Implementation

4.1 Introduction

This chapter describes the steps carried out in the implementation phase of GP Mate. Details about overall implementation of the proposed automated PMS, the choice of tools and technologies for the implementation and justifications for selecting those tools are included in this chapter.

4.2 Version controlling system

For this GPMate patient management system, a Git repository was created in GitHub. Git is an open source version control system. The constant changes made to the code when developing the system can have kept here. This allows easiness in collaboration, as the new versions of the software is downloadable, make updates and upload the latest version. Reason for choosing Git over other systems available was it can store changes to the files more efficiently and guarantees the integrity of the file well. With a unique URL, where the repository is located with all the files related to the project can be accessed. URL for accessing the GPMate project is stated below.

<https://github.com/piumigomes/gp-mate>

4.3 Backend implementation

To implement the back end of the web-based Patient Management System, Spring Boot was used. A stand-alone and ready to use spring application can be created using Spring Boot. Spring Boot is a very stable technology which was introduced by the Pivotal Team which is an open source Java based framework used to create a micro service [9]. This has a large good community forum for ask questions and get help with problems. For this project Spring Boot was especially selected to reduce the complexity of configuration. From using simple customizable properties automatic configuration can be done.

Using Spring Boot a REST API can be built. It is an architectural design for creating applications. Using REST, maintainable, lightweight web services can be created. It treats server objects as resources that can be created and destroyed. Virtually it can be used by any

programming language. Almost all programming languages can work with REST APIs. Using REST API requests can be submitted for CREATE, READ, UPDATE, DELETE operations using the HTTP protocol, and receive responses in the state. APIs can use HTTP requests to render information from a web application. The state can be in XML or JSON format [10]. Figure 5.1 demonstrate an overview of REST API.

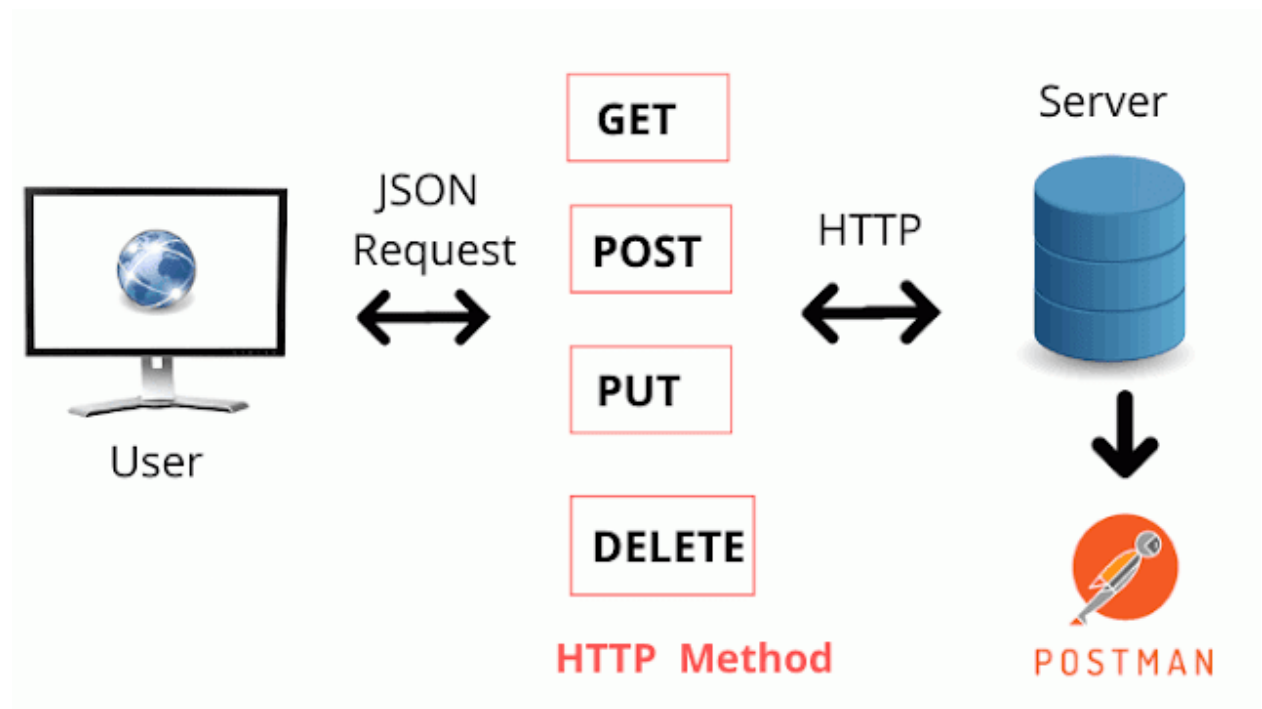


Figure 4.1 – Overview of REST API

The other advantage of using Spring Boot is when the data model is created the relational database tables are automatically mapped and generated. So, there is no need to write SQL queries to manage database and the tables. Spring Boot uses Hibernate ORM for that purpose. It enables programmers to code the applications more easily. As an Object/Relational Mapping (ORM) framework, hibernate is concerned with data persistence as it applies to relational databases [11].

In this application, the server end points are '/users', to '/staff', '/patients', '/sessions. etc. An end point can be a URL of a server or service for APIs. Endpoint is the location where the APIs can access the resources which they require to carry out their functions. In other words, it is the place that APIs send requests where the resources located. APIs work using 'requests' and 'responses'.

These calls are made by the front end, using the angular resource objects using http methods; GET, POST, PUT DELETE. In this project 60+ APIs have been implemented spread across fourteen main endpoints. The list of main API endpoints is described in Table 4.1.

	URL	Description
1	/gpmate/api/auth/	Handle authentication
2	/gpmate/api/service/users	User creation
3	/gpmate/api/service/staff	Create profiles for staff
4	/gpmate/api/service/patients	Handle operations related to patient registration
5	/gpmate/api/service/roles	Create roles to assign users
6	/gpmate/api/service/visits	Manage each patient's encounter within a session
7	/gpmate/api/service/diagnoses	Record diagnoses of a patient
8	/gpmate/api/service/vital-signs	Record vital signs of a patient
9	/gpmate/api/service/medical-notes	Special medical note for patient by the doctor
10	/gpmate/api/service/prescriptions	Create prescription for patients
11	/gpmate/api/service/medical-alerts	Notifications to doctor about patient's special health conditions
12	/gpmate/api/service/channelling-sessions	Create a channeling session for a day to manage patients
13	/gpmate/api/service/investigation-requests	Record laboratory tests to be done
14	/gpmate/api/service/units	Create units which the doctor is available

Table 4-1 – The APPI endpoints

Complete API documentation is available at :

<https://documenter.getpostman.com/view/2724477/SzzoYubQ?version=latest>

Following figures 4.2, 4.3 and 4.4 presents few screenshots from the API documentation.

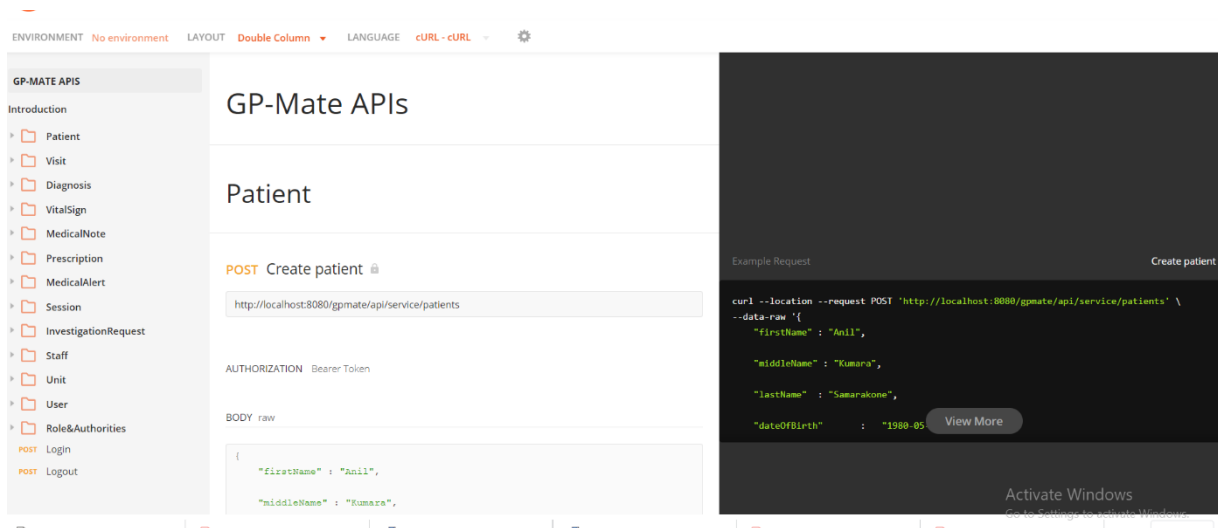


Figure 4.2 – Create patient API

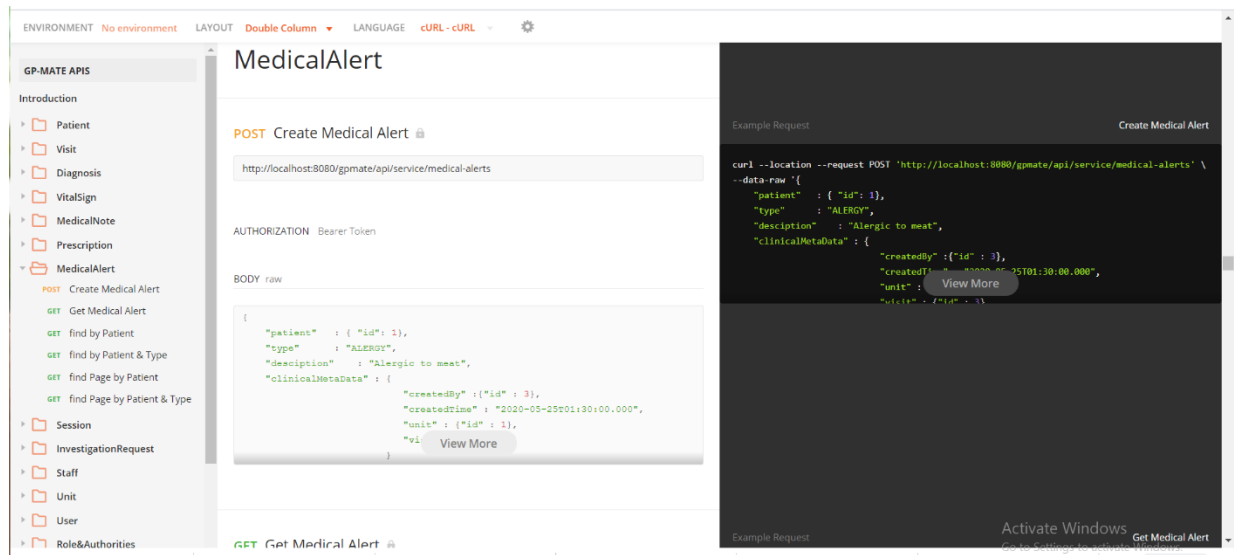


Figure 4.3 – Create medical alert API

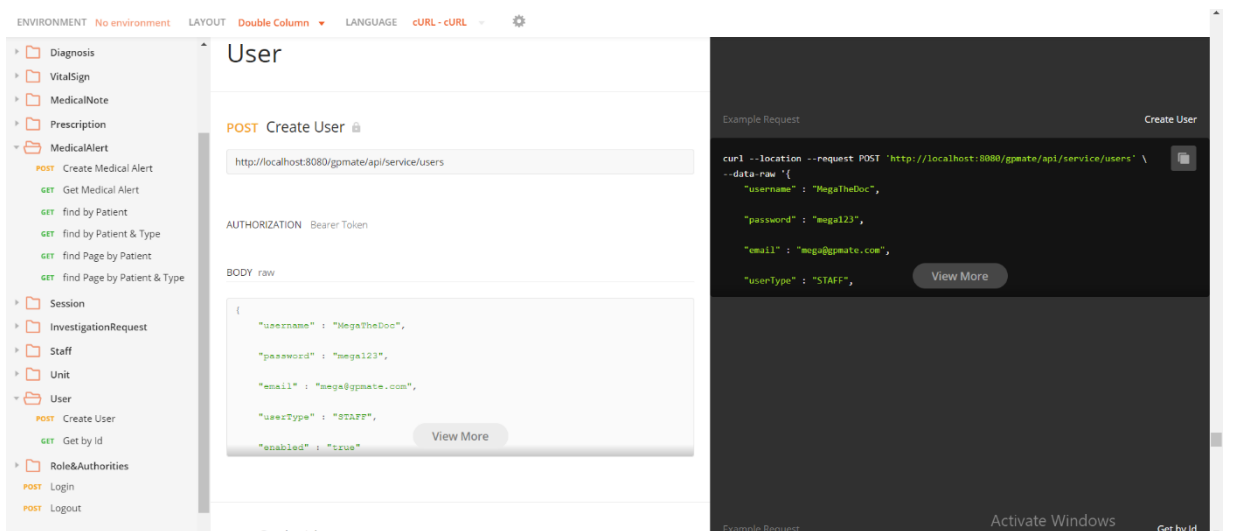


Figure 4.4 – Create user API

4.4 Front end implementation

Angular was used to implement the front end. Angular is an open-source front-end framework developed by Google for creating dynamic, modern web applications [12]. This has obtained a big popularity during recent years for ensuring lightweight and faster applications by eliminating unnecessary code. The most important advantage of Angular is that it is supported by Google's Long-Term Support (LTS).

Another advantage of using Angular is that it is built using TypeScript language, which guarantees high security as it supports types such as primitives, interfaces, etc. TypeScript code can be debug directly in the browser or in the editor if the map files are properly created during the build time.

HTML is used by Angular to define the UI of the application. HTML is a less complicated language as compared to other languages. Therefore, it is no need to spending time in program flows and deciding what loads first. Angular will take care of it when what you require is defined. Testing can be done easily in Angular. Angular.js modules have the parts of the application, which are easy to deploy. Necessary services can be loaded with module separation, while effectively performing automatic testing. Angular framework is adhered with MVC (Model-View-Controller) software architectural format. However, it is more simplified MVC pattern. Following figure 4.5 illustrate the MVC architecture.

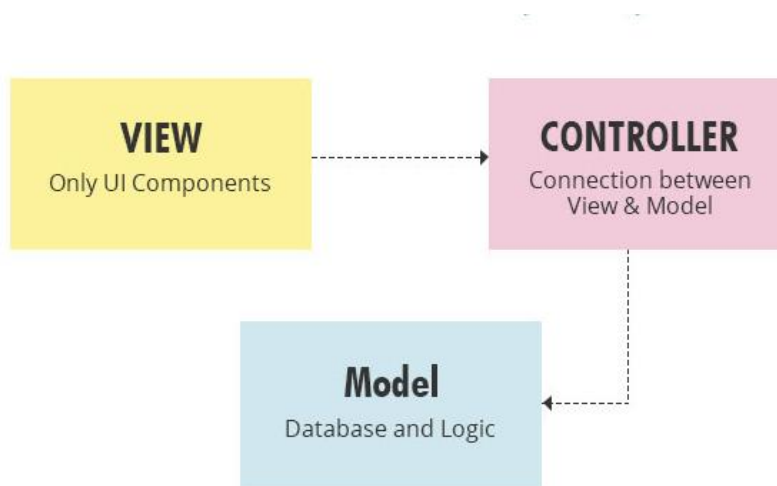


Figure 4.5 – MVC architecture

4.5 Hosting on cloud

Use of virtual hardware, network, storage and merge solutions from a cloud vendor refers principally as cloud hosting.

It is enabled through virtualization, whereby the entire computing capacity of an infrastructure or data center is distributed and delivered to multiple users simultaneously. To host the applications, services and data a user can use underlying infrastructure [13]. Figure 4.6 illustrates the cloud hosting infrastructure.

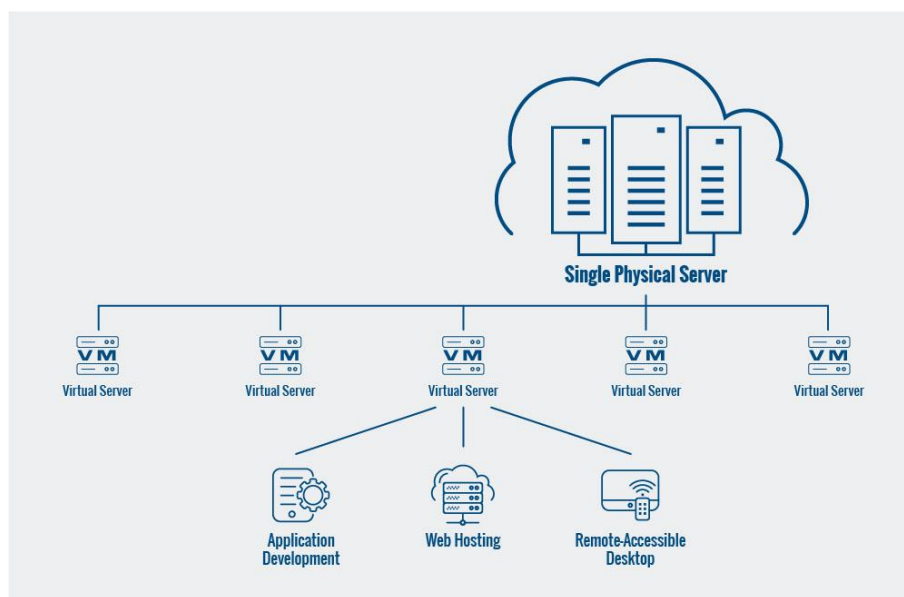


Figure 4.6 – Cloud hosting infrastructure

Cloud hosting is a very scalable, reliable, and flexible type of hosting because of the site is stored on several servers which let to pull resources from several places rather than from one server.

Major drawback with shared hosting is if the server gets down, the system also gets down. But when a cloud server gets down, other servers in that network will care about the system. When considering the today's world, cloud-based development become more and more popular because of the above facts. Where a single application can be range across multiple of servers in different networks with difficult port configurations, the capability to automate the deployment of "units" of code is a great help. Managing OS version, disk size, available

memory, port configuration are some advantages of the ability to control the execution environment. Containerization supports to avoid unexpected encounters when OS libraries create unexpected conflicts.

Docker is a very popular system for containerizing applications. Containerization packages the executable code along with the runtime environment in deployable virtual images using a repeatable, automatable process [14]. For this project Docker is used as the container management system.

4.6 System security

System security is a very crucial part when considering health care. The system must adhere a strong security level as it contains sensitive data. In order to acquire that, a token-based authentication was introduced to this GPMate. Using JSON Web Token, the token-based authentication system was created.

JSON Web Token (JWT) is an open standard (RFC 7519) that defines a compact and self-contained way for securely transmitting information between parties as a JSON object. This information can be verified and trusted because it is digitally signed. JWTs can be signed using a secret (with the HMAC algorithm) or a public/private key pair using RSA or ECDSA [15].

JWTs can be encrypted to provide secrecy among users. But the signed tokens are considered as they can verify the integrity of the claims contained within it, while encrypted tokens hide those claims from other parties. The signature also certifies that only the party holding the private key is the one that signed it, when the tokens are signed using public/ private key pairs.

JSON web tokens are very useful in authorization and information exchange scenarios. Most common scenario for using JWT is authorization. Once a user logged in with a valid token, each request will include the JWT, allowing the user to access permitted resources with that token

JSON Web Tokens contain three parts separated by dots (.), which are:

- I. Header
- II. Payload
- III. Signature

The result is three Base64-URL strings separated by dots. It can be easily passed in HTML and HTTP environments. Following figure 5.4 illustrate an example for JSON web token which was to decoded, verified, and generated using jwt.io Debugger.

Encoded <small>PASTE A TOKEN HERE</small>	Decoded <small>EDIT THE PAYLOAD AND SECRET</small>
<pre>eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJTdW5pbFRoZU1vZGVyYXRvciIsIm1hdCI6MTU4ODg3Mjc4MCwiZXhwIjoxNTg4OTU5MTgwfwQ.11Ju-50tstwB_Cw_msJIwELgVYpVl0tgQD9KTHSoppfiXv7YeH0a3Qe8VP4U-V85SVbz8LA8MmYQYbaA3eRwg</pre>	<div>HEADER: ALGORITHM & TOKEN TYPE</div> <pre>{ "alg": "HS512"}</pre> <div>PAYLOAD: DATA</div> <pre>{ "sub": "SunilTheModerator", "iat": 1588872780, "exp": 1588959180}</pre> <div>VERIFY SIGNATURE</div> <pre>HMACSHA512(base64UrlEncode(header) + "." + base64UrlEncode(payload), your-256-bit-secret) <input type="checkbox"/> secret base64 encoded</pre>

Figure 4.7 – Example for JSON web token

The application or client requests authorization to the server. An access token to the application returns from the authorization server when the authorization is granted. Then the application uses the access token to access a protected resource. As all the information within the token is visible to user, sensitive information should not put in that token. But even though they are visible to users they unable to edit it.

Chapter 5 : Testing and evaluation

5.1 Introduction

Most health care providing organizations are using automated software systems to maintain the functions of their hospitals, clinics, etc. Interrelating all these functions to a single web application is a huge task and making it work properly is even a challenging task. Therefore, rigorous testing of the PMS is compulsory, and it must go through various testing phases. Testing is the process of validating and verifying a system. The system or its components are compared against requirements and specifications through testing. Evaluation of test outputs combines with assessing progress of design, performance, supportability, etc. This chapter discusses how the testing and evaluation was carried out to measure the degree of success associated with the project.

5.2 Testing Strategy

Manual testing was used to test the PMS. To test the APIs, Postman tool was used. All the 60+ APIs were tested with the use of Postman tool, which is a collaboration platform for API development. Some basic functions such as login functionality was automated using Selenium tool for learning purposes. But could not extend to whole system due to the time plan.

Testing was planned to perform for user acceptance testing. User acceptance testing could not have been carried out because of the prevailing situation of the country due to the Covid – 19 outbreak.

5.3 Test Plan

A test plan is a document which states the testing scope and process. It contains the functionalities to be tested, testing procedures, test data and the expected result. Following table 5.1 describes the structure which was followed in testing the system.

Test case no.	Test data	Purpose	Expected result	Actual result

Table 5-1 – Test case structure

5.4 Test Cases

Table 5.2 displays the test cases used for a user sign in. Several test cases try different entries of data fields.

Test case no.	Test data	Purpose	Expected result	Actual result
1.	Enter a wrong username	Test username	Display error message. User not logged in.	Display error message. User not logged in.
2.	Enter a wrong password	Test password	Display error message. User not logged in.	Display error message. User not logged in.
3.	Click login button with empty username	Test login without an account	Display error message. User not logged in.	Display error message. User not logged in.
4.	Click login button with empty password	Test login without a password	Display error message. User not logged in.	Display error message. User not logged in.
5.	Enter correct username and password	Test correct login to system	User logged in.	User logged in.

Table 5-2 – Test cases for login function

Following are some test cases and test outputs resulted in postman tool.

Scenario 01

User login with valid username and password.

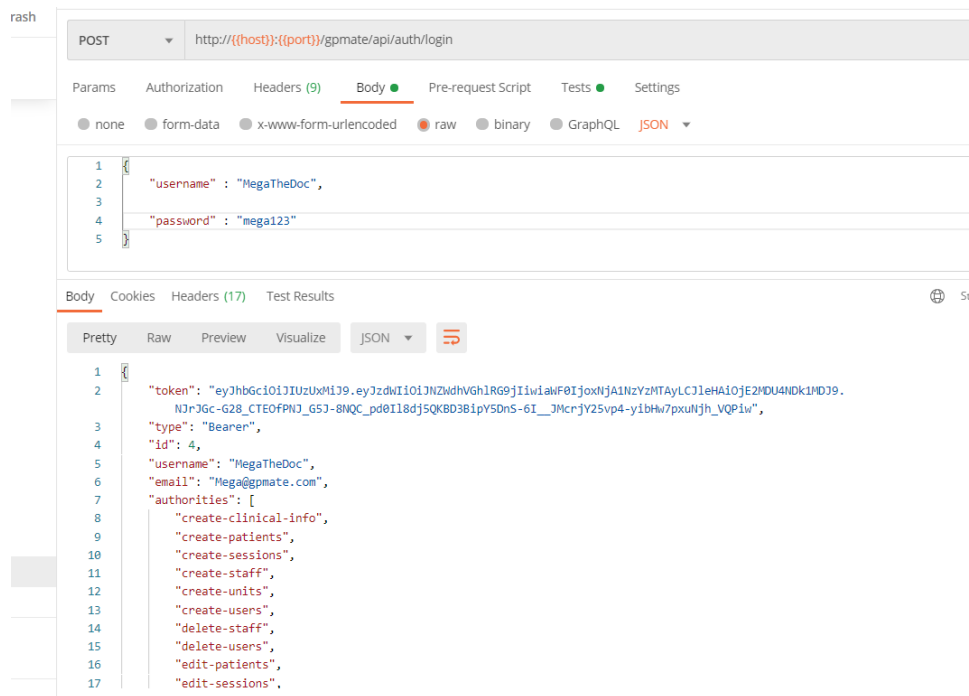


Figure 5.1- User login with valid username and password combination

Scenario 2

User login with invalid username and password combination.

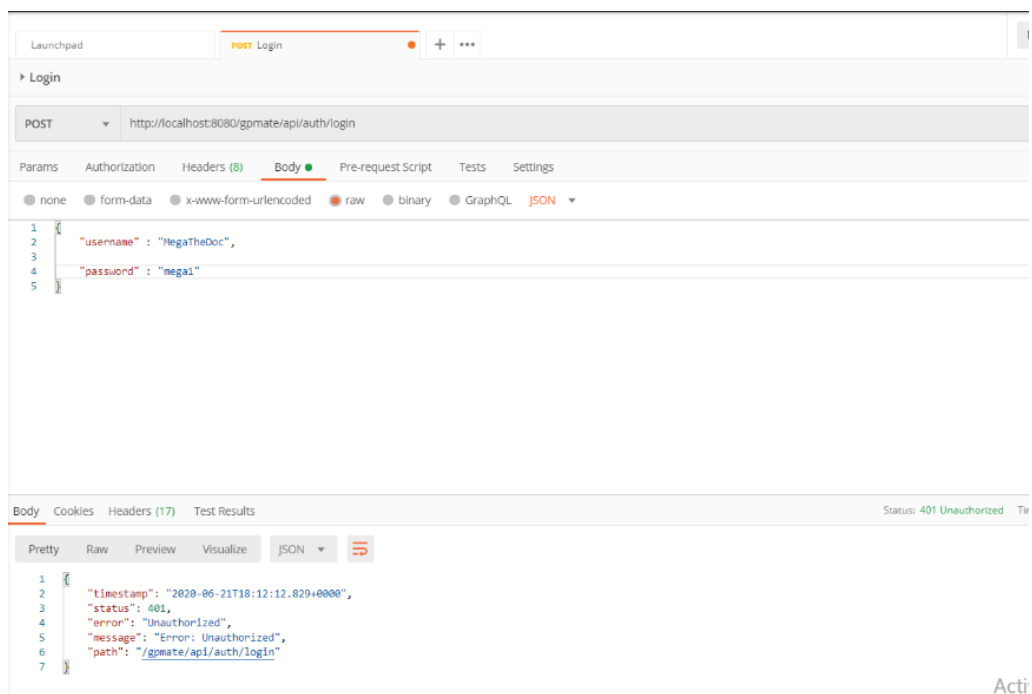


Figure 5.2 - User login with invalid username and password combination

Table 5.3 displays the test cases used for search patient details.

Test case no.	Test data	Purpose	Expected result	Actual result
1.	Enter a wrong identifier type	Test identifier type	Display error message.	Display error message.
2.	Enter a wrong identifier value	Test identifier value	Display error message	Display error message.
3.	Click search button without entering identifier value	Test search without an identifier value	Disable search button Display error message	Disable search button Display error message
4.	Enter correct identifier type and value combination	Test search patient details	Enable search button Patient details	Enable search button Patient details

Table 5-3 – Test cases for search patient details

Scenario 1

Search patient details with wrong identifier value.

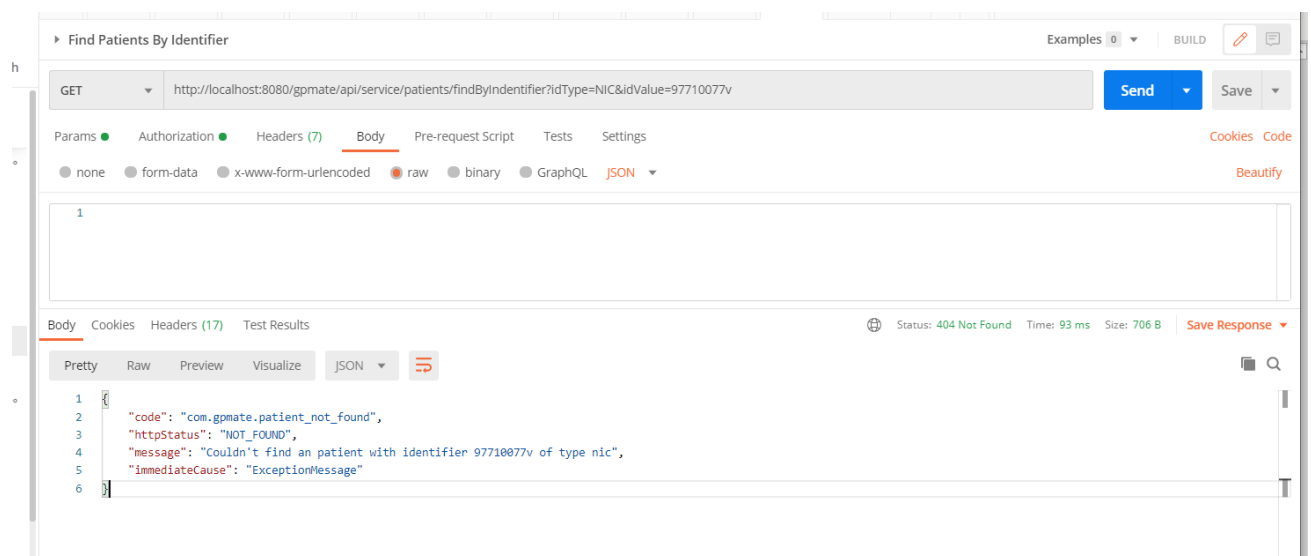


Figure 5.3 – Patient detail search with wrong identifier value

Scenario 2

Search patient details with correct identifier type and value.

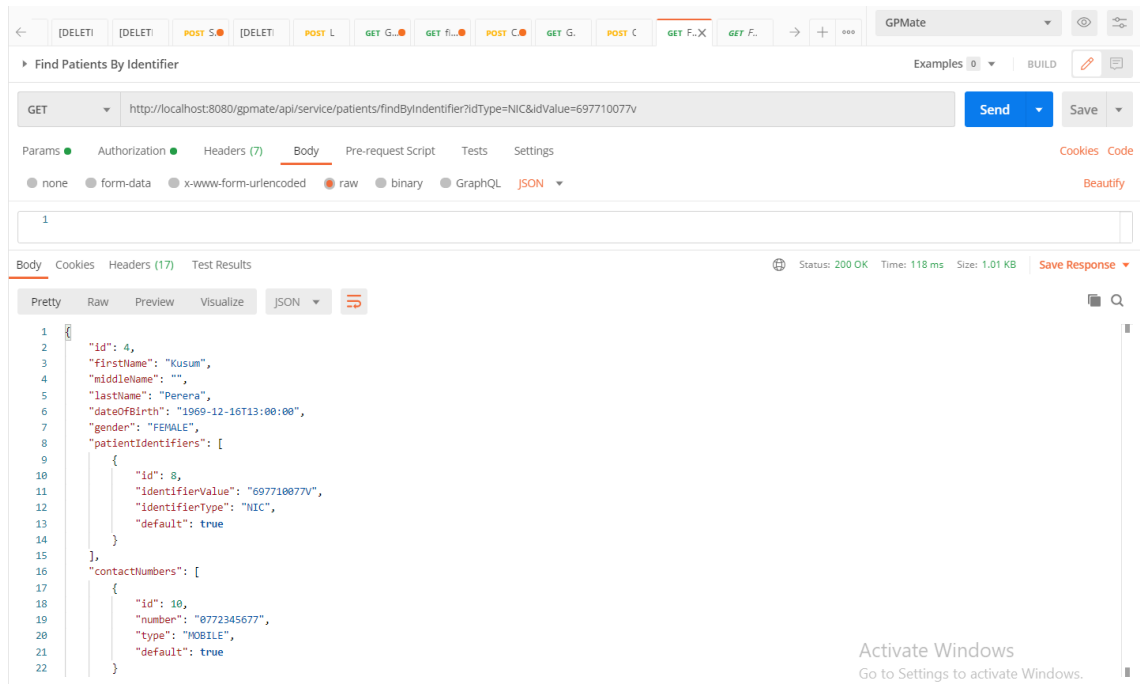


Figure 5.4 - Patient detail search with correct identifier type and value

Chapter 6 : Conclusion and future work

6.1 Introduction

The study reveals that a PMS has a positive impact on healthcare providing medical clinic. It allows smooth interactions with the patient care by automating daily operations. To produce an effective and efficient business model for healthcare practitioners a PMS is a good chance. It allows to store all the records related to patients, better user communication, organize and simplify day to day operations, manage human resources and finally a better service to the customer. Implemented Patient Management System is streamlines processes within the medical clinic to support doctor, support staff and patients to ease their work.

6.2 Problems encountered

- Busy schedule of the doctor, low level computer literacy among the staff, collecting data from the manual records and privacy of the patient's data were the problems encountered when gathering the requirements.
- As the system proposed a short code typing technique for recording diagnosis, a good knowledge had to be gained about the medical terms to find an internationally recognized standards for such recording system.
- Spring Boot and Angular tools was used as development tools. As they have not being used before, going through many tutorials to learn about those technologies to get a thorough idea to use in this project was required.
- Could not manage to finalize the method of mapping diagnosis short codes to the system as the busy work schedule of the doctor and due to the current situation of the country. Therefore, inputting diagnosis through a text box was implemented as suggested by the doctor initially.
- Due to the prevailing situation of the country, user testing and user evaluation could not be carried out.
- Testing could not have been carried out using an automation tool due to the lack of fluency using those tools. Time was not enough to get a through idea about those tools as I'm novice to all the technologies used in this project.

- In this project the user authentication was done using JWT token. When the user is logged, JWT token is set to a predefined expiration time. Even if the user wants logout from the system before the token expiration time, that token will be remain valid. Because the server cannot determine whether this token is invalid as it is still within the expiration period. If the token is maliciously used by some other party before the expiration period, the server sees it as a valid token and accept as a valid request. This was a problem that came up with user authentication. In order to overcome this the token should be blacklisted after a user logout. Therefore, logged out blacklist tokens are maintained. If this blacklist is maintained in the database, it will cause decrease on performance level as it has to check whether the user is logged in every server call. As a solution REDIS in memory blacklist store was implemented. It is much faster and time taking for API authentication is also low.

6.3 Lessons learnt

- Gained a good knowledge on how to work on a collaborative environment. Learnt many technical and personal skills.
- Having done a good basic technical survey on the tools and technologies that must use, the work to carryout can be organized more efficiently.
- And last not least, learning and applying latest technologies to implement the project was a very challenging task. Fulfilling that task amidst under a hectic office, study and family schedules made it more unimaginable. Manage time for work to achieve a goal in a planned time duration is a great lesson learnt during this project.

6.4 Future enhancements

- Create a patient registration portal which enables patients to be registered by themselves.
- Provide facilities to scan and upload laboratory reports, documents from visits to other clinics also can be allowed for documenting medical history of the patients.
- Introducing short code mapping system which is internationally accepted that is associated with open EHR and FHIR standards to enter diagnosis details easily for the doctor.

- Support local languages.
- Generate and print medical reports and prescriptions.

References

- [1] Smartsheet. (2019). *How Patient Management Software Improves the Health Care Experience*. [online] Available at: <https://www.smartsheet.com/patient-management-software-systems> [Accessed 28 Nov. 2019].
- [2] En.wikipedia.org. (2019). *Fast Healthcare Interoperability Resources*. [online] Available at: https://en.wikipedia.org/wiki/Fast_Healthcare_Interoperability_Resources [Accessed 29 Nov. 2019].
- [3] Hhims.org. (2019). *hhimsv2*. [online] Available at: <http://www.hhims.org/> [Accessed 12 Jul. 2019].
- [4] Open-emr.org. (2019). *OpenEMR*. [online] Available at: <https://www.open-emr.org/> [Accessed 1 Oct. 2019].
- [5] Huda, E. and Omaina, B. (2013). *Towards healthcare data security in cloud computing - IEEE Conference Publication*. [online] Ieeexplore.ieee.org. Available at: <https://ieeexplore.ieee.org/abstract/document/6750223> [Accessed 20 Jan. 2020].
- [6] <https://ieeexplore.ieee.org/abstract/document/6750223> [Accessed 20 Jan. 2020].
En.wikipedia.org. (2020). *OpenEHR*. [online] Available at: <https://en.wikipedia.org/wiki/OpenEHR> [Accessed 20 Jan. 2020].
- [7] Suhair Alshehri ; Stanislaw P. Radziszowski ; Rajendra K. Raj (2012). *Secure Access for Healthcare Data in the Cloud Using Ciphertext-Policy Attribute-Based Encryption - IEEE Conference Publication*. [online] Ieeexplore.ieee.org. Available at: <https://ieeexplore.ieee.org/abstract/document/6313671> [Accessed 20 Jan. 2020].
- [8] Stagnaro, C. (2019). *Agile management can benefit healthcare process improvement projects: Health care organization leaders are working hard to create, update and continuously improve on their processes to align with the Triple Aim initiative (see Figure 1. Triple Aim Model)*. [online] Beckershospitalreview.com. Available at: <https://www.beckershospitalreview.com/human-resources/agile-management-can-benefit-healthcare-process-improvement-projects.html> [Accessed 1 Oct. 2019].
- [9] “Spring Boot - Introduction,” *Tutorialspoint*. [Online]. Available: https://www.tutorialspoint.com/spring_boot/spring_boot_introduction.htm. [Accessed: 07-Jan-2020].

- [10] H. Shaikh, “PHP REST API Tutorial Step by Step [Beginners],” *onlyxcodes*, 01-May-2020. [Online]. Available: <https://www.onlyxcodes.com/2019/12/php-rest-api-tutorial.html>. [Accessed: 10-Feb-2020].
- [11] “Hibernate ORM,” *Hibernate*. [Online]. Available: <https://hibernate.org/orm/>. [Accessed: 03-May-2020].
- [12] *Angular*. [Online]. Available: <https://angular.io/features>. [Accessed: 20-Feb-2020].
- [13] “What is Cloud Hosting? - Definition from Techopedia,” *Techopedia.com*. [Online]. Available: <https://www.techopedia.com/definition/29018/cloud-hosting>. [Accessed: 01-May-2020].
- [14] “What is a Container?” Docker. [Online]. Available: <https://www.docker.com/resources/what-container>. [Accessed: 01-May-2020].
- [15] auth0.com, “JSON Web Tokens Introduction,” *JSON Web Token Introduction*. [Online]. Available: <https://jwt.io/introduction/>. [Accessed: 08-April-2020].

Appendices

Source code

Full source code implemented is available at:

<https://github.com/piumigomes/gp-mate>

User manual

User manual to set up the system in a windows environment is as follows:

Prerequisites:

Install;

-docker

-java 1.8

-maven

Step 01 – Go to the following git hub URL:

<https://github.com/piumigomes/gp-mate>

Step 02 - Clone the gp-mate project from git hub link using ‘git clone’ command.

```
>git clone git@github.com:piumigomes/gp-mate.git
```

Step 03 – After successfully cloning the project run the ‘up_db.bat’ file located in

gp-mate >> db_docker folder.

This will compose db_docker_compose file, which would set up and start the application database, Redis - in memory datastore which is used for authentication token blacklisting and adminer db console for db management.

Step 04 – To set up the application server, go to ‘gpmate-server’ folder.

Step 05 – Build the project using the following command:

```
> mvn clean install
```

When the built is successful, the 'target' folder will be created.

Step 06 – inside the 'target' folder run the 'gp-mate-server-0.0.1-SNAPSHOT.jar' file.

Application server will be up and running after this point.

Step 07 – To set up the client server, go to 'gpmate-client' folder.

Step 08 – Run the up_client.bat inside that folder.

When the built is successful production artifact of the application will be created.

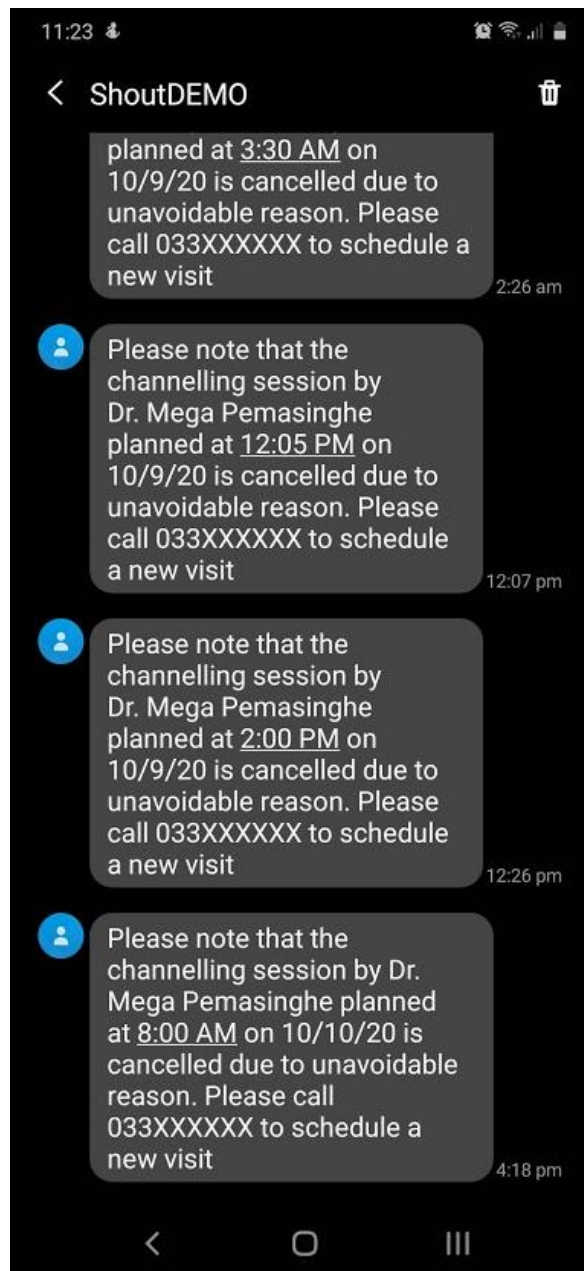
Step 09 – Execute the client docker file to deploy the application using NGINX server.

API Documentation

Complete API documentation is available at :

<https://documenter.getpostman.com/view/2724477/SzzoYubQ?version=latest>

SMS notifications examples



Sample report generated

GPMate						
<u>List of all patients visited during a period</u>						
<u>Duration</u>				<u>Report details</u>		
From : 6/3/2020				Generated at : 11/19/2020, 11:38:21 PM		
To : 11/20/2020				Generated by : Mega Pemasinghe		
<u>Patients</u>						
ID	Name	Gender	Age	City / Town	Contact	Last Visit
NIC : 697710077V	Kusum Perera	F	50	Mirigama	0772345677	10/9/2020 04:14 PM
NIC : 521780775V	Vimal De Costa	M	68	Yakkala	0765544332	10/9/2020 02:41 PM
NIC : 581780666V	Amal Samarakoon	M	62	Gampaha	0755645341	10/8/2020 12:07 PM
NIC : 465724511V	Piyasiri Gomes	M	74	Yakkala	0774887069	10/9/2020 04:15 PM
TEMPORARY : 201506241001	Oneli Perera	F	5	Ja ela	0765432141	10/8/2020 12:07 PM
NIC : 908873761V	Sanduni Fonseka	F	30	Gampaha	+94772531511	10/8/2020 12:08 PM
NIC : 868482117V	Piumi Gomes	F	33	Gampaha	0772531511	10/9/2020 04:15 PM
NIC : 776852546V	Tharangani Ranasinghe	F	43	Raddolugama	0772607181	10/8/2020 12:09 PM
Total Count						8
<u>Summary</u>						
<u>By gender</u>				<u>By age group</u>		
Male	3			Age below 15	1	
Female	5			Age between 16 & 35	2	
				Age between 36 & 59	2	
				Age above 60	3	