HEALTHCARE STORAGE SYSTEM

ABDULLAH ABDELMONEM ABDELRAZEK ZAYAN

UNIVERSITI TEKNOLOGI MALAYSIA

PSZ 19:16 (Pind. 1/13)

UNIVERSITI TEKNOLOGI MALAYSIA



DECLARATION OF THESIS / UNDERGRADUATE PROJECT REPORT AND COPYRIGHT

Author’s full name : ABDULLAH ABDELMONEM ABDELRAZEK ZAYAN

Date of Birth : 2/9/2003

Title : HEALTHCARE STORAGE SYSTEM

Academic Session : 2024/2025

I declare that this thesis is classified as:

|  |  |  |
| --- | --- | --- |
|  | **CONFIDENTIAL** | (Contains confidential information under the Official Secret Act 1972)\* |
|  |  |  |
|  | **RESTRICTED** | (Contains restricted information as specified by the organization where research was done)\* |
|  |  |  |
| **✓** | **OPEN ACCESS** | I agree that my thesis to be published as online open access (full text) |

1. I acknowledged that Universiti Teknologi Malaysia reserves the right as follows:
   1. The thesis is the property of Universiti Teknologi Malaysia
   2. The Library of Universiti Teknologi Malaysia has the right to make copies for the purpose of research only.
   3. The Library has the right to make copies of the thesis for academic exchange.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  | Certified by: |  |
|  |  |  |  |  |
|  | Abdullah Zayan |  |  |  |
|  |  |  |  |  |
|  | **SIGNATURE OF STUDENT** |  | **SIGNATURE OF SUPERVISOR** |  |
|  |  |  |  |  |
|  | A21EC4039 |  | DR. RUHAIDAH SAMSUDIN |  |
|  | **MATRIX NUMBER** |  | **NAME OF SUPERVISOR** |  |
|  |  |  |  |  |
|  | Date: 24 NOV 2024 |  | Date: 24 NOV 2024 |  |

NOTES : If the thesis is CONFIDENTIAL or RESTRICTED, please attach with the letter from the organization with period and reasons for confidentiality or restriction

“I hereby declare that we have read this thesis and in my

opinion this thesis is suffcient in term of scope and quality for the

award of the degree of Bachelor of Computer Science (Software Engineer)”

|  |  |  |
| --- | --- | --- |
| Signature | : | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Name of Supervisor | : | DR. RUHAIDAH SAMSUDIN |
| Date | : | 24 November 2024 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

HEALTHCARE STORAGE SYSTEM

ABDULLAH ABDELMONEM ABDELRAZEK ZAYAN

A thesis submitted in fulfilment of the

requirements for the award of the degree of

Bachelor of Computer Science (Software Engineer)

Faculty of Computing

Universiti Teknologi Malaysia

NOV 2024

**DECLARATION**

I declare that this thesis entitled *“HealthCare Storage System”* is the result of my own research except as cited in the references. The thesis has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

|  |  |  |
| --- | --- | --- |
| Signature | : | Abdullah Zayan |
| Name | : | ABDULLAH ABDELMONEM ABDELRAZEK ZAYAN |
| Date | : | 24 November 2024 |
|  |  |  |
|  |  |  |

**DEDICATION**

This thesis is dedicated to my father, who taught me that the best kind of knowledge to have is that which is learned for its own sake. It is also dedicated to my mother, who taught me that even the largest task can be accomplished if it is done one step at a time.

**ACKNOWLEDGEMENT**

In preparing this thesis, I was in contact with many people, researchers, academicians, and practitioners. They have contributed towards my understanding and thoughts. In particular, I wish to express my sincere appreciation to my main thesis supervisor, Dr. RUHAIDAH SAMSUDIN, for encouragement, guidance, critics and friendship. Without their continued support and interest, this thesis would not have been the same as presented here.

My fellow student should also be recognised for their support. My sincere appreciation also extends to all my colleagues and others who have provided assistance at various occasions. Their views and tips are useful indeed. Unfortunately, it is not possible to list all of them in this limited space. I am grateful to all my family member.

**ABSTRACT**

The functions that surround the access and authorization of medical records are crucial to patient care to reflect what may be seen in real life scenarios regarding patients that may be on the move, and as such, may alter healthcare providers in the process, or give them an incomplete or no record of the patient at all. These bring other challenges like lost or stolen records, which poses serious threat to patient safety and care. The HealthCare Storage System is intended to solve these problems by offering patients an online storage space for all their medical records. The goal of this project is to enable patients to remain in charge of their health information and increase their level of participation in the management of their health. A goal of the project is to create a system that will meet the user needs and at the same time be secure. The system is built with the frontend using React.js, backend using Node.js with Express.js, and MongoDB for the secure storage of data. These are an reminds patients to schedule follow-ups, and helps patients to monitor their heart rate trends. The system includes several strengthened security layers, for example, JWT security for authorization and data encryption. The proposed HealthCare Storage System is expected to increase patients’ satisfaction and chronic disease early intervention due to easy access to their medical history and personalized alerts. Apart from improving patient safety by counterchecking the risks of lost records this system also boosts interactions with care givers making health choices more accurate and timelier. In summary, the HealthCare Storage System is designed for increasing the health information security and patient engagement as well as enhancing patients’ processes and outcomes.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
|  | **TITLE** | **PAGE** |

[**DECLARATION ii**](#_Toc2850685)

[**DEDICATION iii**](#_Toc2850686)

[**ACKNOWLEDGEMENT iv**](#_Toc2850687)

[**ABSTRACT v**](#_Toc2850688)

[**TABLE OF CONTENTS vi**](#_Toc2850690)

[**LIST OF TABLES ix**](#_Toc2850691)

[**LIST OF FIGURES x**](#_Toc2850692)

[**LIST OF ABBREVIATIONS xi**](#_Toc2850693)

[**LIST OF APPENDICES xii**](#_Toc2850695)

[**CHAPTER 1** **INTRODUCTION 1**](#_Toc2850696)

[1.1 Introduction 1](#_Toc2850697)

[1.2 Problem Background 2](#_Toc2850698)

[1.3 Project Aim 4](#_Toc2850699)

[1.4 Project Objectives 4](#_Toc2850700)

[1.5 Project Scope 5](#_Toc2850701)

[1.6 Project Importance 5](#_Toc2850702)

[1.7 Report Organization 6](#_Toc2850703)

[**CHAPTER 2** **LITERATURE REVIEW 7**](#_Toc2850704)

[2.1 Introduction 7](#_Toc2850705)

[2.2 Project Domain 7](#_Toc2850705)

[2.3 Current System Analysis 8](#_Toc2850709)

[2.4 Comparison between existing systems 8](#_Toc2850710)

[2.4.1 Uniqueness of the Proposed System 11](#_Toc2850710)

[2.5 Literature Review of Technology Used 11](#_Toc2850711)

[2.6 Chapter Summary 12](#_Toc2850712)

[**CHAPTER 3** **SYSTEM DEVELOPMENT METHODOLOGY 13**](#_Toc2850713)

[3.1 Introduction 13](#_Toc2850714)

[3.2 Methodology Choice and Justification 13](#_Toc2850715)

[3.3 Phases of the Chosen Methodology 14](#_Toc2850716)

[3.3.1 Requirement Analysis 14](#_Toc2850716)

[3.3.2 System Design 15](#_Toc2850716)

[3.3.3 Implemntation 15](#_Toc2850716)

[3.3.4 Integration and Testing 15](#_Toc2850716)

[3.3.5 Deployment 15](#_Toc2850716)

[3.3.6 Maintenance 16](#_Toc2850716)

[3.4 Gantt Chart PSM 16](#_Toc2850716)

[3.5 Technology Used Description 16](#_Toc2850717)

[3.5.1 Front-End Development 16](#_Toc2850717)

[3.5.2 Back-End Development 16](#_Toc2850717)

[3.5.3 Database 17](#_Toc2850717)

[3.5.4 Security and Performance 17](#_Toc2850717)

[3.6 System Requirement Analysis 17](#_Toc2850718)

[3.7 Chapter Summary 18](#_Toc2850719)

[**CHAPTER 4** **REQUIREMENT ANALYSIS AND DESIGN 19**](#_Toc2850720)

[4.1 Introduction 19](#_Toc2850721)

[4.2 Requirement Analysis 19](#_Toc2850722)

[4.3 Project Design 20](#_Toc2850723)

[4.3.1 Use Case Diagram 20](#_Toc2850723)

[4.3.2 Sequence Diagram 23](#_Toc2850723)

[4.3.3 Activity Diagram 24](#_Toc2850723)

[4.3.4 Class Diagram 25](#_Toc2850723)

[4.4 Project Design 26](#_Toc2850724)

[4.5 Database Design 27](#_Toc2850724)

[4.6 Interface Design 29](#_Toc2850725)

[4.7 Chapter Summary 30](#_Toc2850726)

[**CHAPTER 5** **RESULTS, ANALYSIS AND DISCUSSION 31**](#_Toc2850739)

[51 Introduction 31](#_Toc2850740)

[5.2 Coding of System Main Functions 31](#_Toc2850741)

5.2.1 Registration Function 31

5.2.2 Login Function 32

5.2.3 Upload and Download Medical Reports Function 34

5.2.4 View analysis heart rate dashboard Function 36

5.2.5 Feedback Function 37

5.2.6 Add comment Function 38

5.2.7 Check-up reminder Function 39

5.2.8 Edit Profile Function 40

[5.3 Interface of System Main Function 41](#_Toc2850742)

5.3.1 Register page 41

5.3.2 Patients Login page 42

5.3.3 Patients Home page 42

5.3.4 Heart Rate Dashboard page 43

5.3.5 Admin Login page 44

5.3.6 Admin Dashboard page 44

[5.4 Testing 45](#_Toc2850742)

5.4.1 White Box Testing 45

5.4.2 Black Box Testing 46

5.4.3 User Testing 47

5.4.4 Non-Functional Requirements Testing 51

[5.5 Chapter Summary 53](#_Toc2850742)

[**CHAPTER 6** **CONCLUSION 54**](#_Toc2850739)

[6.1 Introduction 54](#_Toc2850740)

[6.2 Achievement of Project Objectives 54](#_Toc2850741)

[6.3 Suggestions for Future Improvement 55](#_Toc2850742)

[6.4 Chapter Summary 56](#_Toc2850742)

[**REFERENCES 57**](#_Toc2850743)

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **TABLE NO.** | **TITLE** | **PAGE** |

[Table 3.1 Hardware Requirements 18](#_Toc2850744)

[Table 3.2 Software Requirements 18](#_Toc2850745)

[Table 4.1 Actors and their descriptions 22](#_Toc2850746)

[Table 4.2 Use case and their descriptions 22](#_Toc2850747)

Table 5.1 User Testing 48

Table 5.2 Users for User Testing 50

Table 5.3 Non-Functional Requirements Testing 51

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO.** | **TITLE** | **PAGE** |

[Figure 3.1 Waterfall model 14](#_Toc2850749)

[Figure 4.1 Use case diagram 21](#_Toc2850750)

[Figure 4.2 Sequence diagram for UC001 Register 24](#_Toc2850750)

[Figure 4.3 Activity Diagram for UC001 Register 25](#_Toc2850750)

[Figure 4.4 class diagram 26](#_Toc2850750)

[Figure 4.5 Tool Architecture 26](#_Toc2850750)

[Figure 4.6 Entity Relation Diagram 28](#_Toc2850750)

Figure 5.1 Register Patient Function 31

Figure 5.2 Login Patient Function 32

Figure 5.3 Login Admin Function 33

Figure 5.4 Upload medical reports Function 34

Figure 5.5 Download medical reports Function 35

Figure 5.6 Heart Rate Function 36

Figure 5.7 Feedback Function 37

Figure 5.8 Comment Function 38

Figure 5.9 Check-up reminder Function 39

Figure 5.10 Edit profile Function 40

Figure 5.11 Register Page 41

Figure 5.12 Patients Login Page 42

Figure 5.13 Patients Home Page 42

Figure 5.14 Heart Rate Dashboard Page 43

Figure 5.15 Admin Login Page 44

Figure 5.16 Admin Dashboard Page 44

|  |  |  |
| --- | --- | --- |
|  |  | **LIST OF ABBREVIATIONS** |
|  |  |  |
|  |  |  |
| SRS | - | Software requirement Specification |
| SDD  STD | -  - | Software Design Document  Software Test Description |
| OS | - | Operating System |
| HTML | - | Hyper Text Machine Language |
| JS | - | JavaScript |
| ERD  JWT | -  - | Entity Relationship Diagram  Json Web Token |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**LIST OF APPENDICES**

|  |  |  |
| --- | --- | --- |
| **APPENDIX** | **TITLE** | **PAGE** |

[Appendix A SRS 58](#_Toc2850752)

[Appendix B SDD 91](#_Toc2850753)

[Appendix C STD 106](#_Toc2850754)

[Appendix E User Interview 137](#_Toc2850754)

[Appendix D Gantt Chart 139](#_Toc2850754)

1. **INTRODUCTION**
   1. **Introduction**

In the current health care environment, handling, availability and protection of patients’ records is central in organizing quality patient care and optimal health. claims that medical records are crucial in offerance diagnosis and create effective treatment regime since they offer details on patient’s medical history, test results, diagnoses, and treatment plans. However, many patients face several problems when it comes to their personal health records and information that they have access to in order to manage, making all sorts of problems that affect their health ensue.

One major issue is when a patient goes for a checkup or moves around, most of the time they stop with limited records that a new attending doctor can use. This can result into duplication of tests, delay in treatment and where there is severe complication, wrong and delayed medical management may occur. Such disruptions are anti-patient eventualities and are extremely detrimental when delicate information concerning the patient’s health is missing during emergent situations or critical consultation.

Further, most patients carry with of indexed card, cumulative record, or scattered records on computer; they are easily lost, destroyed, or stolen. This makes patients have no information that the health care providers may require very much in making decisions. However, it is regrettable that the increase of digital systems does not solve the problem yet, because at the moment, no patient has a secure and easily accessed information storage and retrieval system for medical data on hand. This deficiency leaves a gap in patient-centered care since the patient lacks the means to control or access his or her health information meaning any patient’s participation in positive health management will be affected.

Another challenge is that patients fail to recognize the convenience and safety of the service, or they miss or forget scheduled check-ups or follow-ups, respectively. Reviewing and follow-up visits are important to diagnose and treat illnesses at their preliminary stages so that they do not become severe or long-lasting. Nevertheless, since there is no specific system that would update patients on their needs concerning the health, the majority of them abandon these measures.

This is especially the case given the current rate of technology advancement thus the need for a secure central place to store the patient medical record that can efficiently be accessed and managed by the patient. Another way in which a stronger and more comprehensive digital architecture might benefit patients is that it will make them masters of their medical information, to a certain extent, while keeping that information secure from hackers and thieves.

In this project, these challenges are to be met by introducing a system that will not only focus on the question of data safety and availability but also would take into account the patient’s experience and desire in managing his or her health. Through this project was to close the gap between the time that healthcare providers open patient’s records and patients access them, the vision of this project was to empower patients to take full control of their health by making the right decisions.

* 1. **Problem Background**

Being able to obtain accurate and complete information about a patient or upcoming operation at the earliest is imperative to the healthcare industry. However, there is always a challenge for many patients in the process of managing and storing their records, which may be dangerous in the future. A major challenge arises when the patient is mobile or changes their residence often, meaning that not all the information about the patient is passed on to new healthcare providers. Such lack of continuity in medical records leads to redundant operations such as repeated diagnostics and treatment, delays, and in some instances, incorrect treatments that can significantly harm patients. Moreover, many conventional records—whether paper-based or in disorganized digital formats—are vulnerable to being lost, damaged, or stolen, depriving patients and providers of crucial medical information during treatment.

Another key problem is the absence of an easily accessible, secure, and user-controlled platform for managing personal health records. Without such a system, patients rely on informal means of record-keeping that make it difficult to organize health information and maintain regular follow-up care. Many patients forget or miss scheduled medical check-ups, increasing the likelihood of developing undiagnosed or poorly managed health issues. A well-structured solution would empower patients with a centralized tool for managing their health, reducing the risk of medical errors and supporting more informed decisions about their care.

To address these issues, the proposed HealthCare Storage System offers a secure digital vault where users can manually store and manage their health records. The system also includes two key features aimed at enhancing patient engagement and preventive care. First, the **Medical Check-Up Reminder System** allows patients to manually enter the date of their most recent full medical check-up. Based on this date, the system automatically schedules and sends a follow-up reminder after one year. This feature helps patients keep track of their health and promotes regular preventive care.

Second, the system includes a **Heart Rate Analysis Dashboard**. Patients can now put their heart rate values along with the date and time of measurement. The system then visualizes this data on a dashboard, enabling patients and healthcare professionals to track changes in heart rate over time. This is particularly important for monitoring chronic conditions such as cardiovascular disease, where consistent tracking of heart rate trends can provide valuable insights into a patient's ongoing health status.

Unlike previous efforts such as Microsoft HealthVault and Google Health, which were discontinued due to concerns over privacy, security, and user engagement, the HealthCare Storage System prioritizes robust encryption, secure authentication processes, and patient ownership of data. These safeguards help ensure that users can confidently access and manage their medical data anytime, from anywhere, thereby improving patient satisfaction and encouraging long-term engagement with the platform.

* 1. **Project Aim**

The purpose of this project is to create a HealthCare Storage System, which is a web application where patients can store their health records and have tools such as notifications and health data to improve the decision making and patients’ involvement.

* 1. **Project Objectives**

The objectives of the project are:

1. To identify and analyze the specific user requirements for the Health care storage system, focusing on the need for secure medical report storage, patient reminders, and health data visualization.
2. To design a secure, user-friendly web-based platform that allows patients to store, access, and manage their medical reports, while also incorporating features like check-up reminders and heart rate analysis dashboards.
3. To develop the platform using React.js for the frontend, Node.js with Express.js for the backend, and MongoDB for secure data management, implementing authentication (JWT) and encryption for protecting sensitive medical data.
4. To test the system's functionalities, including medical report uploading/downloading, automated reminders, heart rate dashboard, ensuring that all use cases perform as expected based on user requirements.
   1. **Project Scope**

The scopes of the project are:

1. The system will focus on creating a comprehensive medical document storage platform, accessible as a website that can be used on laptops only.
2. The system will allow users to manage their medical history, including uploading and downloading test results, with personalized comments and document metadata.
3. The system will allow user to upload only pdf file for set alert and to analysis the Heart rate based on report result.
4. The system aims to provide a secure and accessible medical data vault, the project will not focus on long-term scalability and maintainability in this version.
   1. **Project Importance**

The HealthCare Storage System is an essential solution in the healthcare sector that responds to the major issues in health record storage and retrieval. This project has made it easy to have a safe central place of storing and retrieving patient records hence increasing patients’ safety, reducing the issues of loss of data and puts the patients in a position to be fully involved in receiving their care. Other tools of the system include; check-up reminder and heart rate analysis dashboard, these ensure that patients adopt suitable health check-up habits since they readily embrace their health situation and make appropriate decisions. Apart from the enhancement of patients’ satisfaction and engagement it also benefits the healthcare workers by making certain that all patient’s record are holistically and accurately captured, eliminating repetitive tests and acute ill treatment. In the long run, HealthCare Storage System enhances the management of health care in that information is protected, retrievable and personalized to meet the patient’s requirements in the long run.

* 1. **Report Organization**

This chapter is the first one of the thesis and it has presented the project and the background, the relevance, and the aims of the HealthCare Storage System. It has also raised awareness of the necessity of a safe and convenient environment for storing patients’ records and the characteristics that meet this requirement. The next chapter will be a literature review, which will discuss the systems and research related to this project. Chapter 3 will then present the approach used in the design and development of the system, and the reasons for the selection of the specific approaches and technologies. Chapter 4 will present the system requirements and design; the functional and non-functional requirements will be described, as well as the architecture and interface design. Subsequently, Chapter 5 will be devoted to the description of the system implementation and testing, which will include code snapshots, main features, and results of the user testing. Last, Chapter 6 will present the conclusion and recommendation for future improvement of the system.

1. **LITERATURE REVIEW**
   1. **Introduction**

In this chapter, the current solutions and tools that are used for medical records storage and management will be discussed and compared to the HealthCare Storage System that is proposed in this work. First, we will review the existing systems in the healthcare market and compare them to demonstrate the gaps that can prevent them from meeting patients’ requirements for secure and convenient medical record storage. Furthermore, we will discuss the selected technologies for constructing our system and how each of them helps in the accomplishment of the project objectives. This literature review will establish a background to the HealthCare Storage System and explain why it is required to enhance the healthcare system’s functionality, and why the chosen design and development approach is suitable.

* 1. **Project Domain**

The domain of this project includes healthcare information management with an emphasis on the protection of electronic records. It entails the understanding of how data privacy is done, how secure authentication is done and how web technologies are used to develop a patient centric system. It also contains aspects of health data visualization and user experience design to make medical records both retrievable and easy to handle for patients.

* 1. **Current System Analysis**

The current system, (Sehhaty), is a government-based platform in Saudi Arabia that aims to offer diverse health care services to the citizens and the residents in the country. Some of the features include; medical consultation, appointments, personal health records, and electronic health information. Sehhaty aspires to improve the quality of the healthcare services by making it easier for people to access basic health information and services. Yet, Sehhaty is mainly aimed at providing state health services throughout the Kingdom of Saudi Arabia only and it lacks unique personalized health management features like automatically timed medical check-up or the heart rate analysis software. Moreover, while it offers the electronic copy of the health records, it does not allow patients to have a personal health vault where patients can organize, maintain and store various health records on their own. The proposed HealthCare Storage System is actually an extension of Sehhaty that allows people to take charge of their health and manage their medical data in a secure central repository from anywhere.

* 1. **Comparison between existing systems**

To stress the benefits of the proposed HealthCare Storage System, one has to consider its counterparts – the current healthcare management platforms. The following table gives an overview of some of the features in (Sehhaty), Huawei Health and the proposed HealthCare Storage System.

|  |  |  |  |
| --- | --- | --- | --- |
| **Characteristic** | **Sehhaty** | **Huawei Health System** | **HealthCare Storage System** |
| Web-based | ✘ | ✘ | ✔ |
| Mobile Application | ✔ | ✔ | ✘ |
| Accessible Outside Saudi Arabia | ✘ | ✔ | ✔ |
| Personalized Health Management Tools | ✘ | ✔ | ✔ |
| Medical Check-Up Reminder | ✘ | ✘ | ✔ |
| Heart Rate Analysis Dashboard | ✘ | ✔ | ✔ |
| Secure Personal Health Vault | ✘ | ✘ | ✔ |
| Government-Operated | ✔ | ✘ | ✘ |
| User-Controlled Data Management | ✘ | ✘ | ✔ |
| Available for General Public | ✘ | ✔ | ✔ |

(Sehhaty) is a mobile application launched by the ministry of health in Saudi Arabia to enable the citizens and residents to access different types of health services. They can use it to book doctor appointments, review and access their health records, request prescriptions and receive health notifications. However, Sehhaty is only available on mobile and is only operational in Saudi Arabia. It mainly concerns government health services and does not provide features for controlling one’s condition and monitoring heart rate. Also, being a state-owned application, the users have relatively low data ownership and act as an interface to the governmental health services rather than as a personal health data storage.

Huawei Health is an application which is intended for checking and organizing users’ health and fitness activities. Some of the features include heart monitor, tracking of sleep, and exercise program tailored to an individual. It offers medical check-up reminders and a heart rate analysis dashboard which will improve the users’ health management. Huawei Health is available for use around the globe and has the option of selecting multiple languages to work with. However, it mainly serves the purpose of the fitness and health tracking tool and does not have proper privacy features where users can put all their personalized and complex health records. The data management is mainly centralized with the application, and the users have little control over their health data.

The HealthCare Storage System is an online solution that aims to help patients store, retrieve and organize their medical records using the Internet, irrespective of their location in the world. It is not limited to the Sehhaty application and can be used worldwide. The system focuses on the user’s control of the data, and the patient has complete control over their medical data. Some of the features include a Medical Check-Up Reminder System, which is used to remind patients when they are due for a checkup and the Heart Rate Analysis Dashboard where patients can input their heart rate data from a PDF report and view the trend. These one-of-a-kind personalized health management tools enable self-management of condition allowing patient to be charged with their health. The platform is online based, which makes it easier to use from any device that has internet connectivity, but at the moment I don’t have a mobile application. Thus, the HealthCare Storage System is free from the disadvantages of the existing systems such as Sehhaty and Huawei Health, as it aims at the secure storage of personal data and active health management.

### Uniqueness of the Proposed System

The HealthCare Storage System solves healthcare problems by managing medical reports securely and effectively. The HealthCare Storage System delivers medical data protection features that stand apart from basic cloud storage solutions including Google Drive. The system's distinct advantage comes from its specialized medical records vault design which includes personal health reminders and analysis tools that standard storage services lack. Google Health represented a health-focused system from Google but the company ended its operation between 2012 and 2013 because users lacked trust and privacy fears existed. The failed launch of Google Health shows why we need a dedicated platform for medical data management.

Our system solves these problems by using strong encryption methods plus secure login systems that put users in control. Our system enables patients to take charge of their medical data while meeting all healthcare data protection requirements. The HealthCare Storage System offers a vital secure medical data platform that people need across the globe as health awareness grows and data protection becomes essential.

* 1. **Literature Review of Technology Used**

The HealthCare Storage System presented here is a proposed system that uses several web technologies to ensure the secure storage and management of patients’ records. The technologies used are as follows:

1. Frontend Development using React.js

The choice of the frontend is React.js – one of the most popular and widely used JavaScript libraries for creating reactive web interfaces. It is based on the components which allow enhancing the look and feel of the elements developed by the developers, and making a system more user-friendly and easier to navigate.

1. Node.js with Express.js (Web Development Backend)

Node.js is used for the multiple processes and Express.js is used for multiple requests on the backend part. As for Node.js, it is rather appropriate to use in the context of the development of large-scale systems; at the same time, with help of Express.js working with server-side logic becomes rather easy, as well as interacting with a database.

1. MongoDB (Database Management)

Patient records are stored and retrieved using MongoDB, a NoSQL database that is secure in handling patient’s data. It provides a distinct and versatile management of various medical papers while it also ensures that the data is stored in an encrypted way by giving importance to the protection of essential health information. JWT which stands for JSON Web Tokens, is mostly used for Authentication. The Just In Time (JWT) is used for the purpose of user authentication and security. It offers an effective means of transmitting messages between the client and server with security and protection of integrity of the information being transferred.

1. bcrypt.js (Password Hashing)

bcrypt.js is used to hash user passwords to make it safer. It also provides for the safe storage of user credentials so as eliminate possibilities of unauthorized access.

The following technologies have been chosen in order to meet the requirements for reliability, security and interactivity of the HealthCare Storage System, thus making it suitable for the management of healthcare data.

* 1. **Chapter Summary**

This chapter compared the features of the existing systems, (Sehhaty) and Huawei Health and then, the proposed HealthCare Storage System has been highlighted with the different advantages of the proposed platforms. The comparisons showed how current systems fail in aspects like web accessibility, user-owned data control, and personal health management applications. Furthermore, it was established that the proposed HealthCare Storage System offers features that are not offered by the existing systems, including a secure personal health vault, medical check-up reminders, and a heart rate analysis dashboard that can fill this gap sufficiently.

1. **SYSTEM DEVELOPMENT METHODOLOGY**
   1. **Introduction**

In this chapter, the approach that will be used to design the HealthCare Storage System will be presented and rationale for its selection will be provided. To ensure that the reader has a clear understanding of the development process, the phases of the chosen methodology will be described. Also, this chapter will also present the technologies and tools employed in the development of the system. A Gantt chart will also be used in showing the project timeline and the system requirements will be evaluated in order to check on the compliance with the project objectives.

* 1. **Methodology Choice and Justification**

The Waterfall model has been selected as the method for the development of the HealthCare Storage System. This model is linear and each phase of development is in a sequence with the previous and the next phase. The phases of the Waterfall model are well defined and therefore the management and control of the project over its life cycle is well done.

The Waterfall model suits this project best because it is easy to follow a clear structure that is laid down by this model. This is useful because any requirement for the system is captured and well-studied before getting to the designing stage thus eliminating cases where some requirements are missed out in the process. Also, the phase-based approach means that the tasks are straightforward to assign and manage to prevent gaps in a systematic approach to project completion.

Following Waterfall model for the HealthCare Storage System implies a clear path for the project from beginning to the end, which enhances communication, resources, and time controls to envisage the project.

* 1. **Phases of the Chosen Methodology**

The Waterfall model, as shown in the figure 3.1 below, is divided into six phases. Phases are done consecutively, and no phase shall start until the one before it is done. In PSM1, the focus is on the first phase which is the Requirement Analysis.



Figure 3.1: Waterfall model

* + 1. **Requirement Analysis**

The first phase involves identification and documentation of requirements of the HealthCare Storage System. This process consists of determining the needs of the users, i.e., the patients and the weaknesses of the related systems, including the Sehhaty and Huawei Health systems. From interviews with volunteer patients and a software engineer of DoctorOnCall, it was possible to identify the necessary features: data storage, medical check-up reminders, and heart rate analysis. A primary goal of this phase is to confirm all system requirements before one move to the design phase of the software.

* + 1. **System Design**

The design phase involves mapping of the requirements into a detailed architecture of the HealthCare Storage System. This involves drawing of diagrams for use case diagrams, class diagrams and activity diagrams for the purposes of defining system interactions and structure of the system. Also, decision making of suitable technologies during this phase involves choosing of the React.js for the frontend application development, Node.js and Express,js for the back end application development while data management is handled by MongoDB.

* + 1. **Implementation**

The implementation phase includes putting into practice the system that had been designed. In this stage, implementation of the design occurs to form the required software application. All the parts of the architecture are combined to make a working system: the frontend, the backend, and the database. In this phase, the effectiveness of the proposed features, for example safe space for data storage, individualized reminder, among others, are provided.

* + 1. **Integration and Testing**

When individual modules are created, they are combined to form the complete HealthCare Storage System. This is followed by a set of phases of testing such as unit testing, integration testing and user acceptance testing with a view of identifying major defects. This phase ensures that the system is running as required by and expected by the users.

* + 1. **Deployment**

After testing, the system is used by the intended audience as a tool in their day-to-day activities. The deployment phase also decrees the system available for patients and healthcare providers to use through features that enhances the secure management of records.

* + 1. **Maintenance**

The last stage is the reassessment of a system where problems that may occur post implementation are identified. This involves making some correction, implementing changes, bug fixing and an upgrading of the system as the case maybe. Maintenance is important as it keeps the HealthCare Storage System up and running and meeting users’ needs as they are in the future.

* 1. **Gantt Chart PSM**

A Gantt chart is used to give an idea of the timeframes needed to be done with each task and the progress made, and ensures the tasks are performed on time or before deadlines. A Gantt chart that represents the progress made in PSM1 is available in appendix D.

* 1. **Technology Used Description**

The main technologies that will be used in developing the HealthCare Storage System are React.js, Node.js with Express.js, MongoDB, JWT, and bcrypt.js.

* + 1. **Front-end Development**

The front end of the HealthCare Storage System will be in React.js a JavaScript library that is well known for its features of creating interactive UIs. It is possible to create reusable components in React.js, and it also provides the best user interface and responsiveness. This helps to make the system as friendly as possible to all the users of the system.

* + 1. **Back-end Development**

The back end of the application will be created using Node.js with Express.js which create enviable conditions for executing server-side logic, plus handling multiple requests in an organized manner. In terms of scalability Node.js is a great match, Express.js is convenient for creating of RESTful API for the fast communication between the front end and the database. These technologies also enable the fast integration of proper measures in handling data, so that the system can run effectively and securely.

* + 1. **Database**

Patient data will be stored and managed by MongoDB which is a NoSQL database. MongoDB is preferred for flexibility to accommodate all these data formats, and its capacity to expand as the data increases. Also, the client information is accommodated in MongoDB with the encryption to make sure of the revelation of the sensitive medical information.

* + 1. **Security and Performance**

For security reasons, user authentication will use JSON Web Tokens (JWT). JWT is a good means to authenticate users and protect the data exchange between the client and the server. As for passwords, there is a bcrypt.js module that will be used for the hashing of users’ passwords to minimize the risks of break-ins. In combination, these technologies guarantee data security for users across the system.

* 1. **System Requirement Analysis**

It is necessary to determine the minimum set of requirements for the development and functioning of the HealthCare Storage System to guarantee its technical viability and efficient functioning. The requirements of the system are divided into two types, the hardware requirement and the software requirement.

Table 3.1: Hardware Requirements

|  |  |
| --- | --- |
| **Hardware** | **Specification** |
| Processor | Intel Core i5 or higher |
| Memory | At least 8GB RAM |
| Hard Drive | At least 255 GB HDD/SDD |
| Internet Connection | Ethernet connection. |

Table 3.2: Software Requirements

|  |  |
| --- | --- |
| **Software** | **Specification** |
| Operating System | Windows 10 or above, macOS or Linux |
| Web Browser | Google Chrome, Mozilla Firefox |
| IDE | Visual Studio Code |
| Front-end Tools | React |
| Back-end Tools | Node.js with Express.js |
| Database | MongoDB |
| Security Tools | JSON Web Tokens (JWT), bcrypt.js |

* 1. **Chapter Summary**

This chapter explained the approach taken for the development of the HealthCare Storage System where the Waterfall model was adopted due to its linear approach. This model has a clear sequential development where each phase is only commenced once the previous phase is fully done. The phases of the methodology were described and the relevance of each step was explained in relation to the accomplishment of the project objectives. Besides, paired with the major technologies of the project, React.js, Node.js, Express.js, MongoDB, JWT, bcrypt.js were introduced to specify that this is suitable for the creation of a reliable, efficient, and friendly system. In the next chapter, we will discuss user requirements and describe the system in detail.

1. **REQUIREMENT ANALYSIS AND DESIGN**
   1. **Introduction**

This chapter is devoted to the analysis of the requirements to the HealthCare Storage System as well as the design of this system, including the database and the interface. The first objective is to establish what the system is expected to do, how it is going to be constructed and what major activities and entities are involved. The requirement analysis was done by talking to volunteer patients and a software engineer from DoctorOnCall to ascertain the features and functionalities required for the system. Furthermore, this chapter will also describe the database and the user interface components to achieve the goal of security, ease of use, and accessibility.

* 1. **Requirement Gathering**

To obtain the requirements for the HealthCare Storage System, the interview method was integrated with document review. These methods were selected to gain a broad understanding of the requirements for the system and to overcome the deficiencies of the existing processes. The system is designed to offer a safe environment for patients to maintain their records and for the admins to control the system’s operations efficiently.

Volunteer patients were also interviewed together with a software engineer for the DoctorOnCall, a healthcare technology firm. In all, patients were interviewed on the difficulties they encounter when storing and retrieving their medical records. Furthermore, the conversations with the software engineer helped to understand how data should be stored securely and what is important to know when designing a system. These interviews revealed several critical pain points:

1. In order to have a better understanding of what they actually need, the patients themselves may misplace their records or may forget when they have been to a health check-doctor.
2. To complement the proposed design structure, the software engineer emphasised the need for secure authentication and encryption of medical data.

A sample of similar systems which include the Sehhaty application used in Saudi Arabia and other health care management tools was examined to draw useful information on possible features and functionality that could be incorporated into the HealthCare Storage System. It also helped in pointing out the drawbacks of the existing systems, for example, restricted access and absence of user control over the data which is to be managed by the proposed system.

* 1. **Requirement Analysis**

Requirement analysis is defined as who needs the system and what they expect from it to be developed. It embraces requirement definition, documentation, and validation and management of the identified requirements to meet the intended purpose of the developed system. This process is crucial in order to clearly and in detail define what the system has to deliver to meet the needs of the users. In the context of the HealthCare Storage System, the requirement analysis means the identification of the functionalities that can be interesting for Patients and Admins, for example, to work with the medical reports, to receive individual recommendations, and to have the administrative access to the system. This section will also demonstrate use case diagram which will detail the functional and operational requirements of the system and the interactions of the sequence diagram, class diagram, and activity diagram will also be used in this section to demonstrate the functionality of the system.

* + 1. **Use Case Diagram**

The use case diagram for the proposed HealthCare Storage System illustrates two primary actors: the Patient and the Admin. The Patient uses only account management, report management, and health-related analytics, the admin deals with system operation, including account authorization and reports’ considering. To increase the readability of the diagram, the use cases are divided into categories such as Account Management and Report Management.

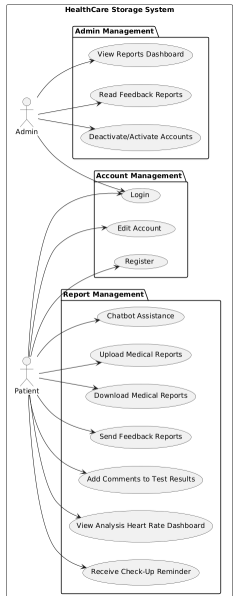


Figure 4.1: Use case diagram

The use case diagram includes two primary actors: the Patient and the Admin. The relationship of each component with the HealthCare Storage System is described in Table 4.1 below.

Table 4.1: Actors and their descriptions

|  |  |
| --- | --- |
| Actor | Description |
| Patients | A user who creates an account with the system, authenticates the account and uploads, downloads and manages medical reports. Patients also utilize the application to take a heart rate analysis, receive medical check-up reminders or provide feedback to administrators. |
| Admin | A user with rights to review feedback reports, to operate with accounts, and to view the dashboard reports. They make sure the platform runs with no hitches and is safe. |

Table 4.2 below will include a brief description of each use case in the use case diagram.

Table 4.2: Use cases and their descriptions

|  |  |  |
| --- | --- | --- |
| Use Case ID | Use Case | Description |
| UC001 | Register | Enables a new patient to sign up for the platform. |
| UC002 | Login | Allows users to login into their accounts through their credentials securely. |
| UC003 | Edit Account | Enables patients to change their details that may be in the account. |
| UC004 | Upload Medical Reports | Allows patients to upload their medical records in the system. |
| UC005 | Download Medical Reports | Enables patients to download their medical reports that they had uploaded earlier. |
| UC006 | Send Feedback Reports | Allows the patients to forward their comments or complaints about the system to the admin. |
| UC007 | Add Comments to Test Results | Enables patients to add notes to their medical test results to help them remember in the future. |
| UC008 | View Analysis Heart Rate Dashboard | Helps patients to have a visual understanding of the patterns of their heart rates by analysing the input that the patients upload. |
| UC009 | Receive Check-Up Reminder | Informs patients automatically for the need of medical check-ups based on their last date they put. |
| UC010 | View Reports Dashboard | Enables the admin to see specific or general reports about the use and performance of the system. |
| UC011 | Read Feedback Reports | Allows the admins to view the feedback that patients have given. |
| UC012 | Deactivate/Activate Accounts | Enables the administrators to have control over the users’ account and especially when one needs to be activated or deactivated. |
| UC012 | Chatbot Assistance | Offers patients a quick, automated help service for resolving queries related to system functionality, such as how to upload reports or navigate the platform. |

* + 1. **Sequence Diagram**

This diagram provides information on the events, their sequence and interactions between the actors and the system that occurs in a particular use case. It offers an account of how the user and the system engage to achieve the goals. The sequence diagram below indicates the register process. Other sequence diagrams of other use cases like download report or receive check-up reminders are provided in the appendix. Another important type of diagrams is Sequenced that helps to show the dynamic view of the interaction between the user of the software system and this system during specific operations.

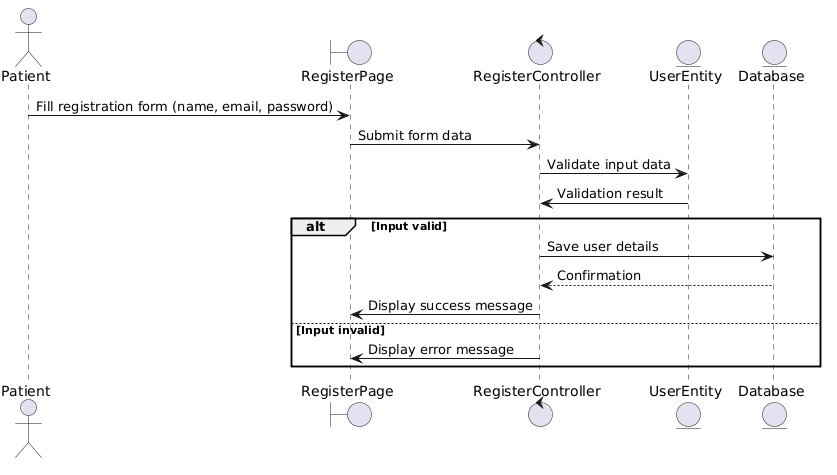


Figure 4.2: Sequence diagram for UC001 Register

* + 1. **Activity Diagram**

An activity diagram describes the dynamic aspect of a system through the modelling of the flow of activities and interactions. It shows how the chain of activities and choices is performed in the system throughout a specific process. The activity diagram below embodies the actions in the process of the “Register” use case, UC001, whereby a new patient signs up for an account in HealthCare Storage System. Other use cases activity diagrams can be seen in the Appendix of this document.

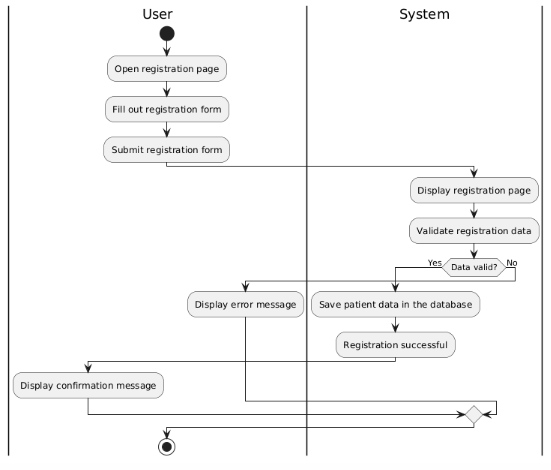


Figure 4.3 Activity Diagram for UC001 Register

* + 1. **Class Diagram**

The next figure represents the HealthCare Storage System from a static perspective that is, it shows the classes that exist in the system, the attributes of the classes, and the operations that can be performed on the classes. As well as, it shows the hierarchy and partnerships of different agents in the given system and helps comprehending the structure and interactions to provide the implementation of the features of the system.

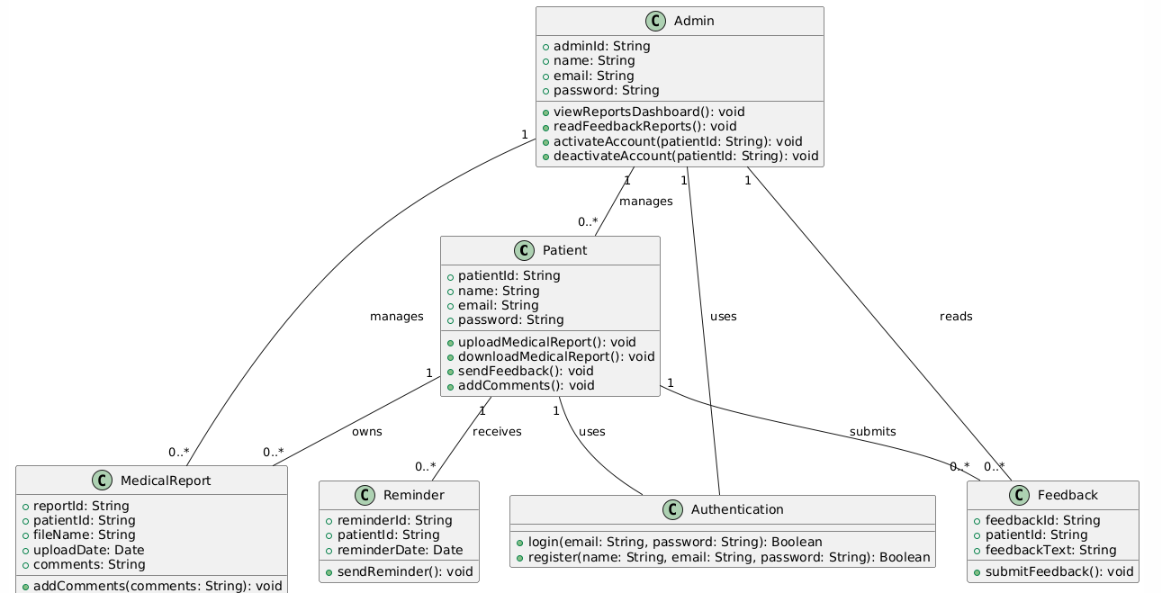


Figure 4.4: Class diagram

* 1. **Project Design**

In this section, we will describe the architecture of the HealthCare Storage System and describe how the components work together as shown in the following Figure 4.5

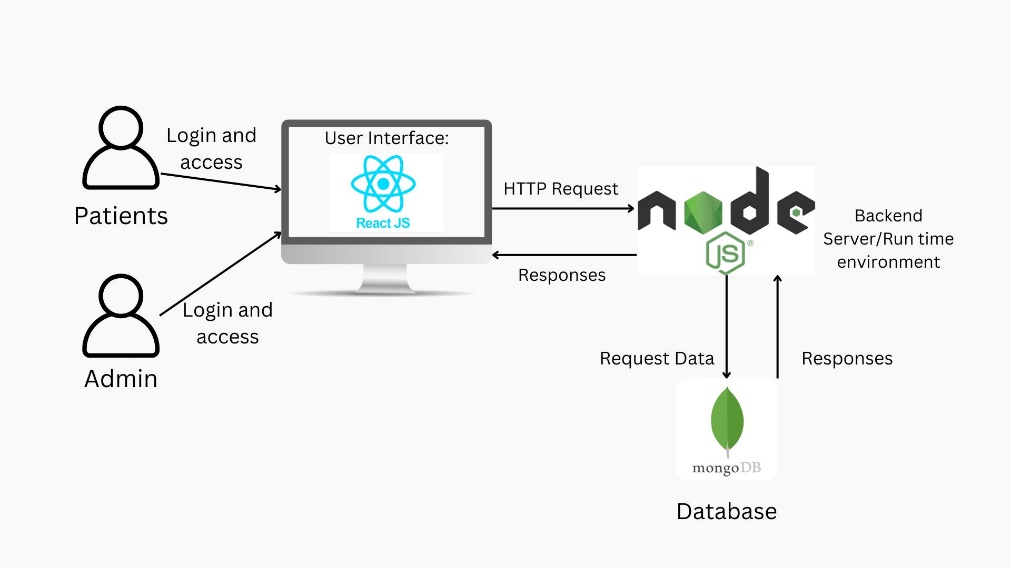


Figure 4.5: Tool Architecture

The system architecture shows how information and operations move from the user, through the interface, the backend server and the database. The users (patients and admins) directly engage the application through the browser with the help of the developed User Interface (UI) using React.js. Any operation like registration, upload of reports or even the viewing of dashboards is done through the UI. These actions produce related HTTP requests that are transmitted to the back-end server through secure connection.

The backend server, designed by Node.js with Express.js, works on handling the received requests and utilizes the MongoDB DBMS for arising/fetching/modifying/deleting medical records and other system associated data. This allows for dynamic functionality of the system, for instance storing of new medical reports or generating check-up reminders.

After the backend server has gone through the request, it forwards the responses to the React.js UI and updates the system on the side of the users. Authentications measures for example JSON Web Tokens (JWT) enable reliable communication between the user, the back-end, and the database.

In the administrative aspect, the system offers the admin a special panel from which can track the reports and feedback. Other backend and database operations such as account activation or deactivation of admin specific features are also performed.

* 1. **Database Design**

Database design is significant in the development of the HealthCare Storage System since all data such as the user data, medical reports, feedback, and reminders will be stored, managed and retrieved from the database. The database helps to make the system run effectively and securely with the assurance of data accuracy. Entity Relationship Diagrams (ERD) are normally used to illustrate the entities within the database, of the attributes and the relationships between them. In this subchapter the ER Diagram for the system will be shown, as well as the tables, attributes and relations between the entities.

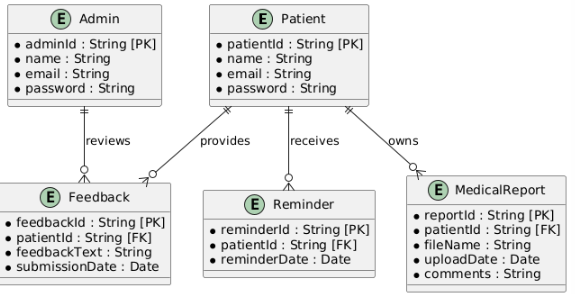
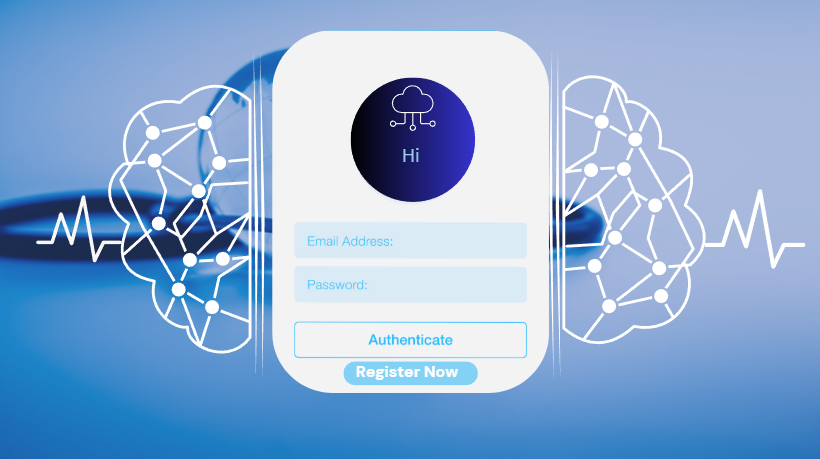
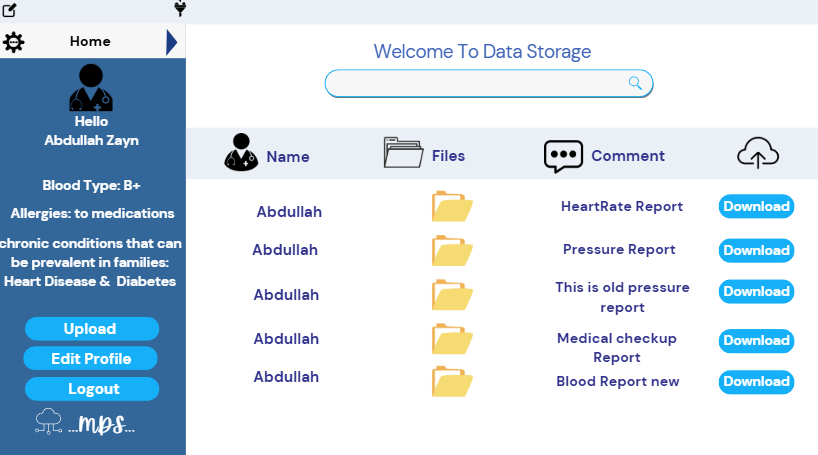


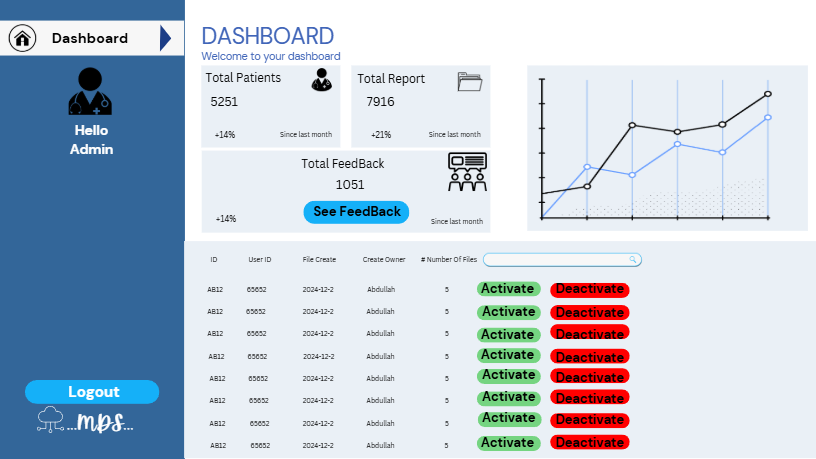
Figure 4.6: Entity Relation Diagram

* 1. **Interface Design**

The design of the user interface is an essential aspect of the HealthCare Storage System to allow ease of use. To this end, the design emphasizes usability, including navigation, layout, and accessibility to all the system components. Front-end language used for interface development where complex and interative parts of components can be developed. The interface also contain patient’s page and the administrator’s page each of which is specialized to their activities and responsibilities. The figure below shows the Patient Login Page, the Patient Dashboard and how the system ensures that users sign, as well as assist the client in user-friendly features like the upload of medical reports, viewing of dashboards, and notification among others.







* 1. **Chapter Summary**

This chapter has described the requirement analysis and design of the HealthCare Storage System. The use case diagram, sequence diagram, activity diagram, and class diagram were used to model the system and its interactions. The database design was also discussed including the layout of the system’s data and how different entities are linked. Moreover, the system’s architecture and the user interface were introduced, which set the basis for the development stage. These designs make the system usable, functional and secure in order to address user and stakeholder requirements.

1. **RESULTS, ANALYSIS AND DISCUSSION**

## Introduction

This chapter discusses the process of testing the HealthCare Storage System and the results obtained from evaluation. It includes the main system functions, for example, registering and logging in patients, storing medical reports safely, entering heart rate data manually, and giving patients reminders for their medical check-ups based on the dates they provide. The chapter talks about the methods used during development to test the system for reliability and accuracy. It also keeps a record of the test cases, the environment where the system was tried, and the results obtained. This chapter serves to assess the system’s ability to meet its functions and discover any parts that could be improved in the future.

## Coding of System Main Functions

### Registration Function

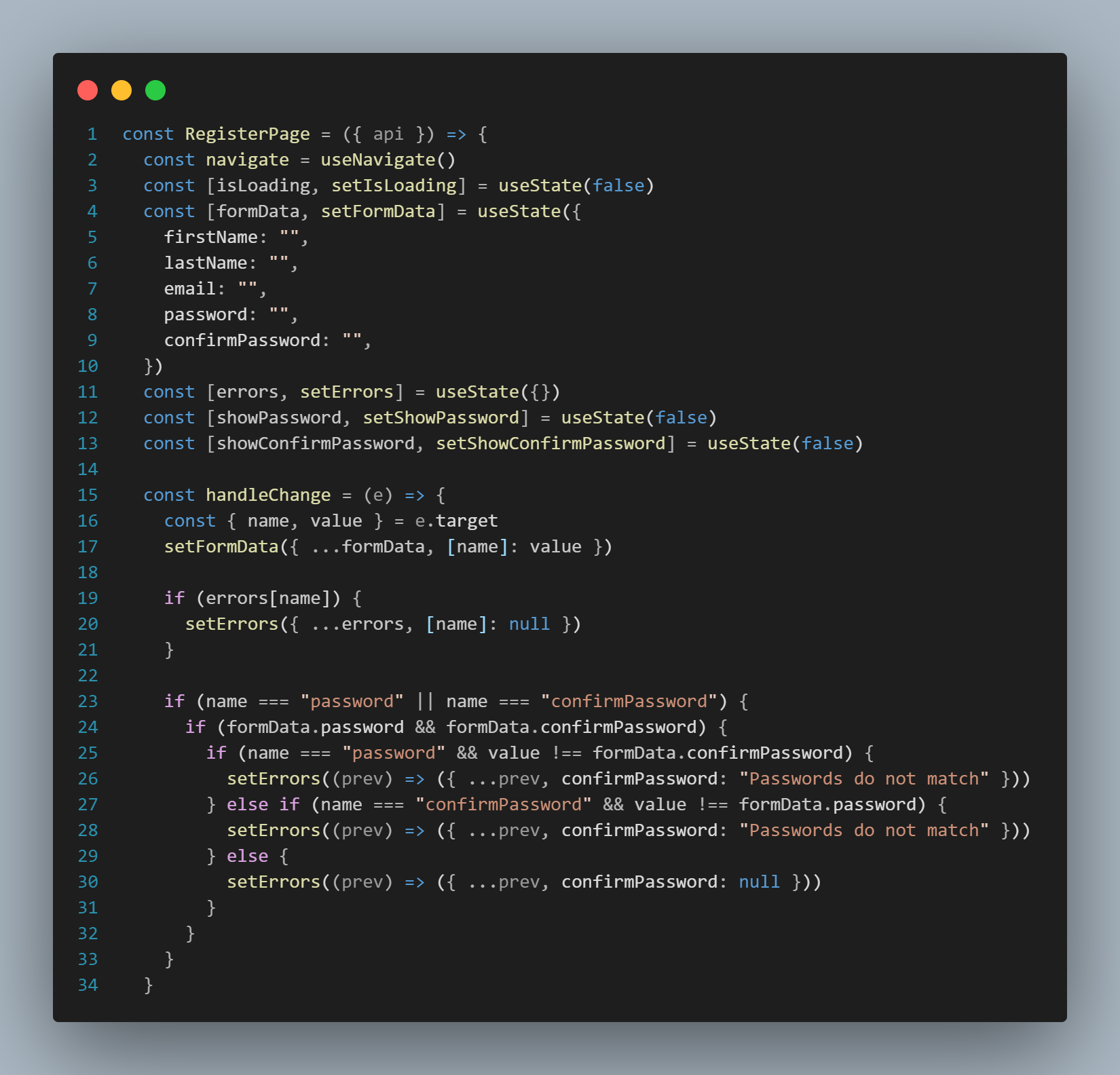


Figure 5.1: Register Patient Function

### Login Function



Figure 5.2: Login Patient Function



Figure 5.3: Login Admin Function

### Upload and Download Medical Reports Function

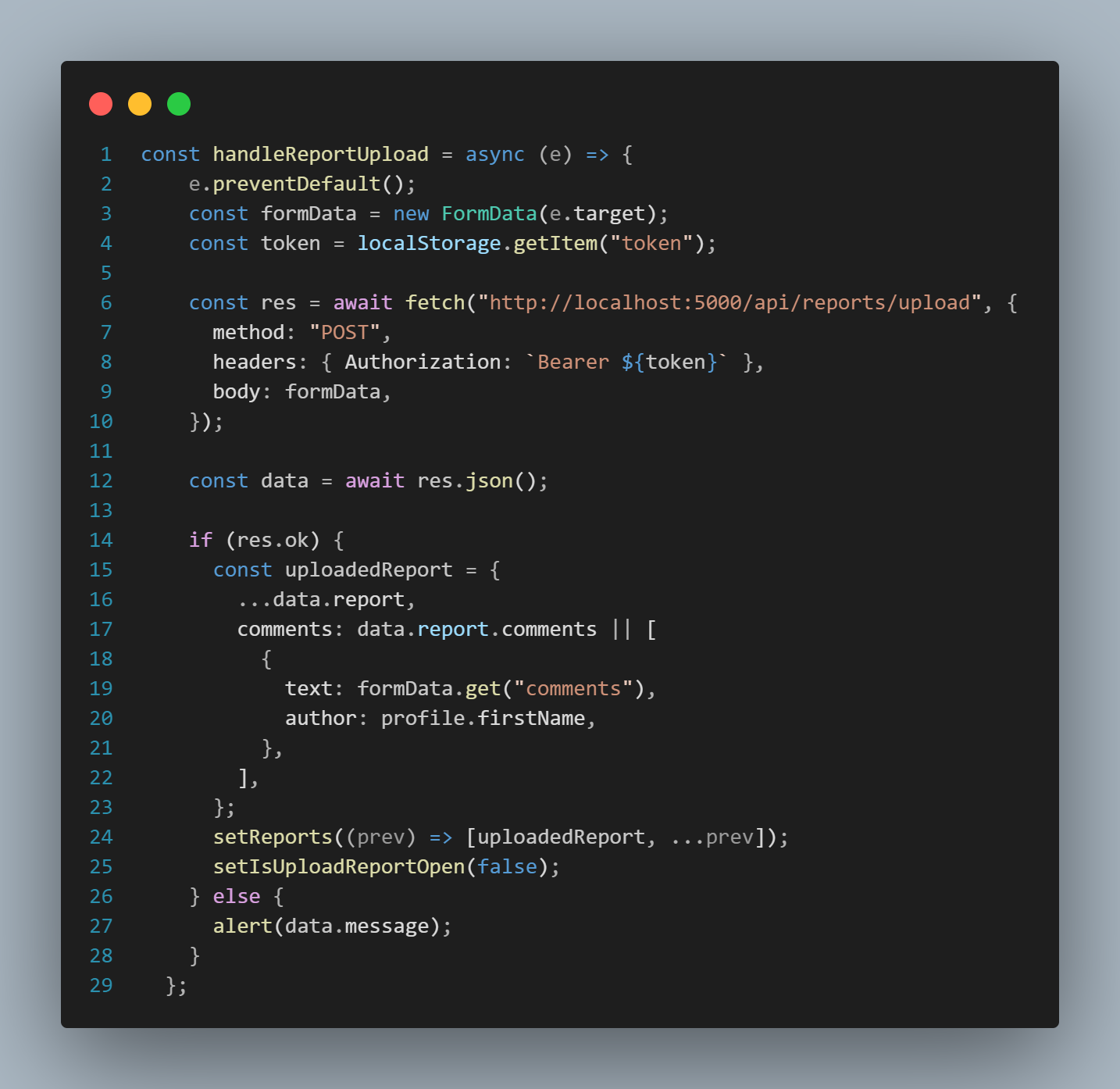


Figure 5.4: Upload medical reports Function

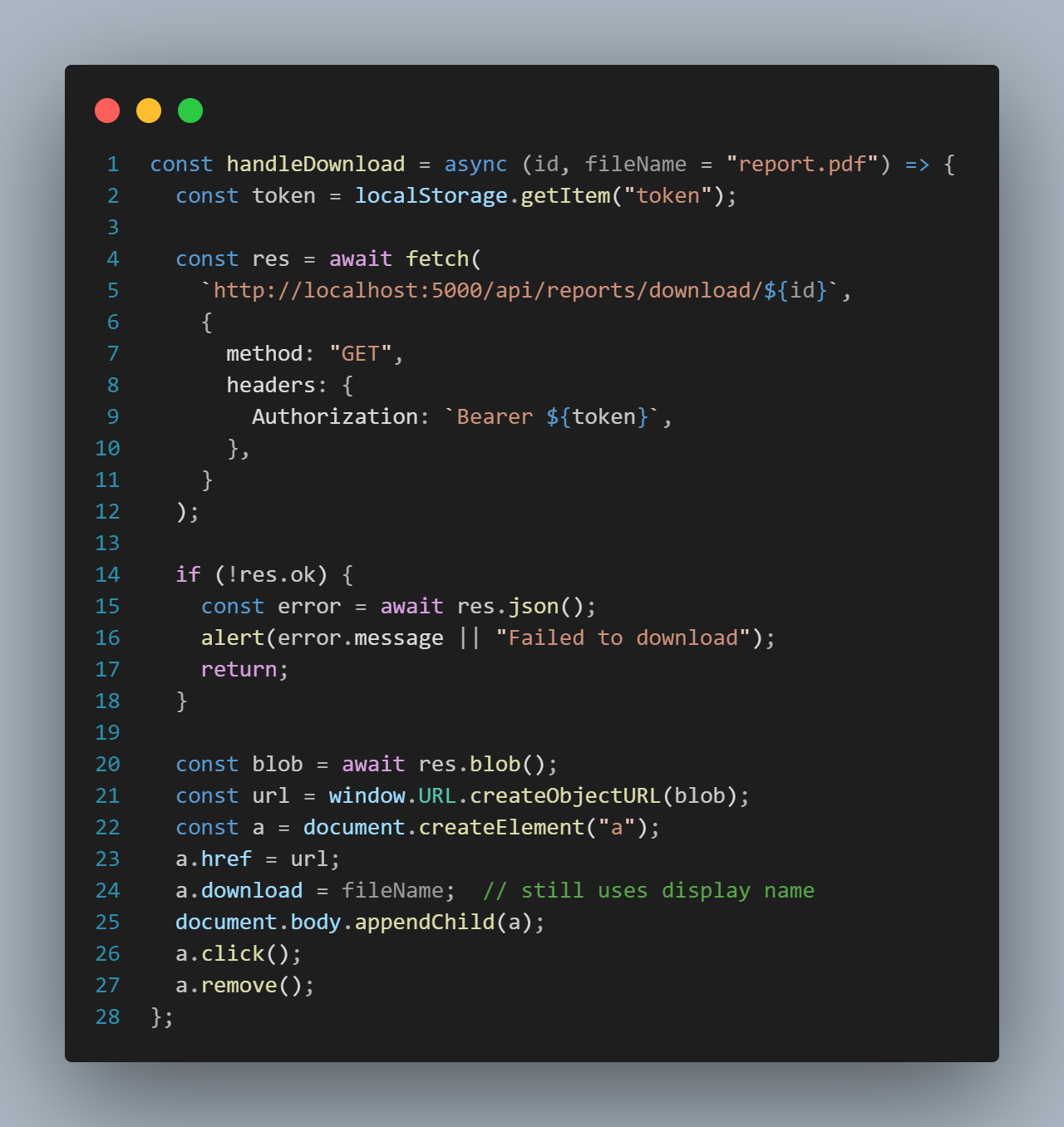


Figure 5.5: Download medical reports Function

### View analysis heart rate dashboard Function



Figure 5.6: Heart Rate Function

### Feedback Function



Figure 5.7: Feedback Function

### Add comment Function



Figure 5.8: Comment Function

### Check-up reminder Function

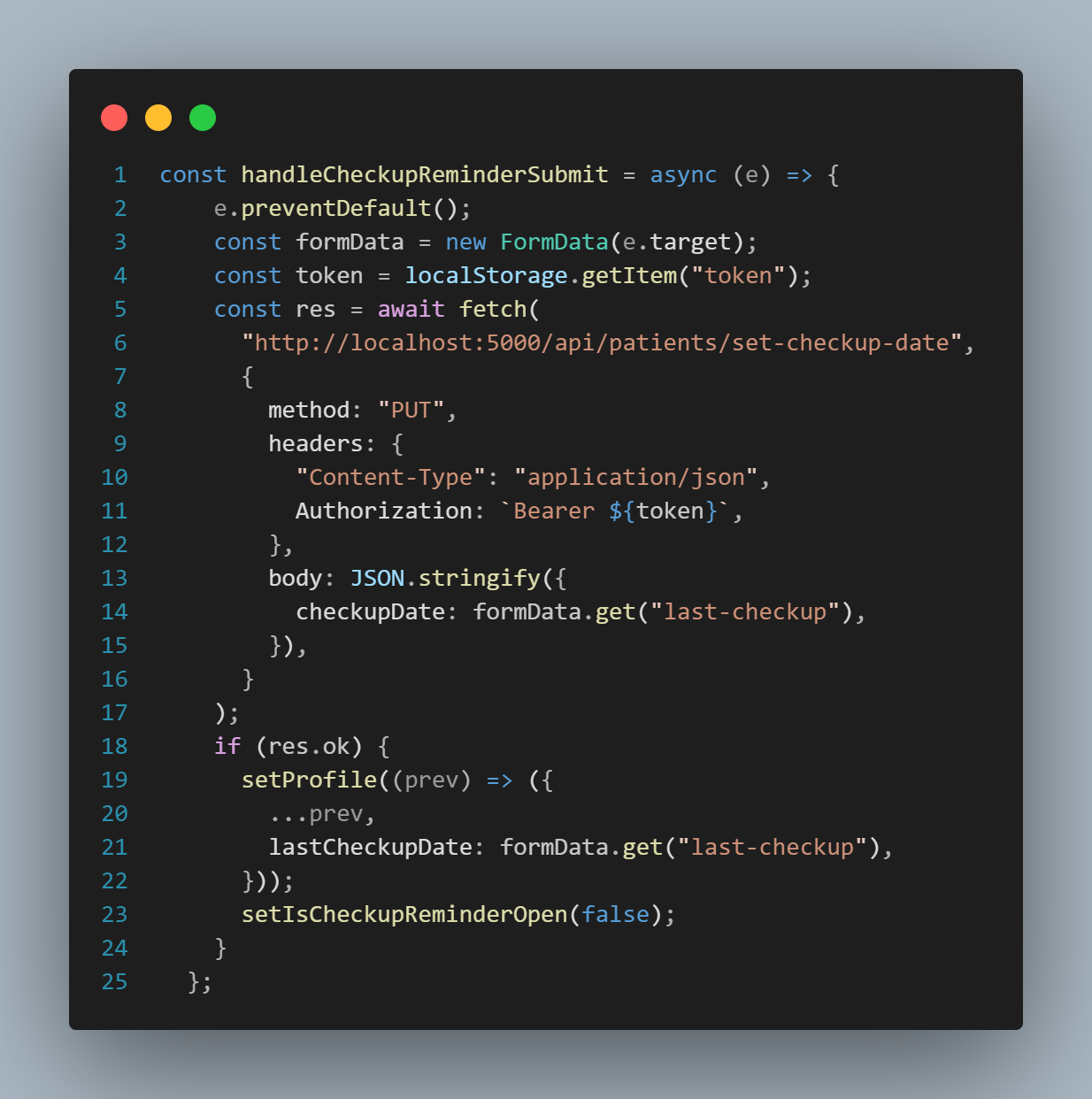


Figure 5.9: Check-up reminder Function

### Edit Profile Function

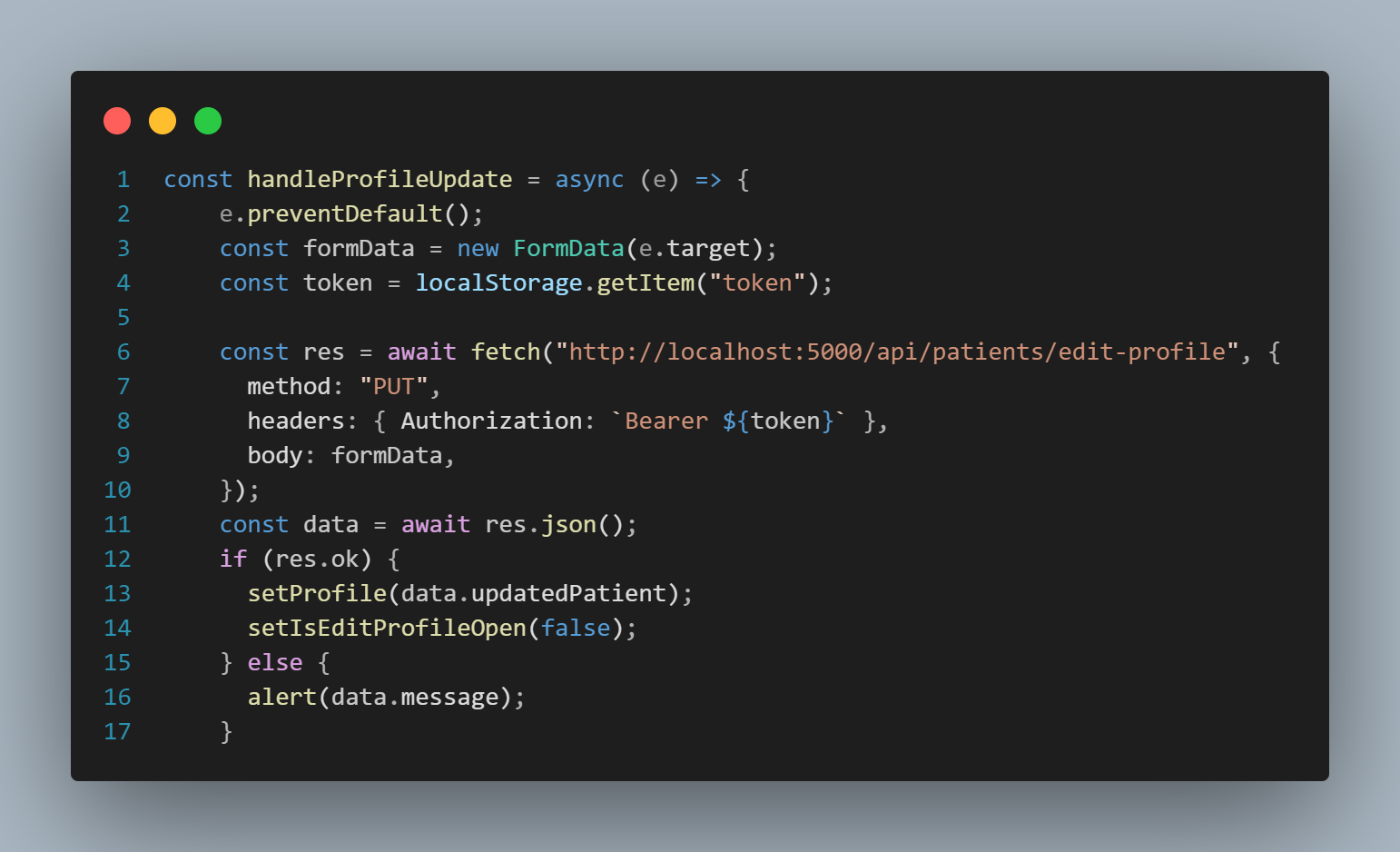


Figure 5.10: Edit profile Function

## Interface of System Main Function

### Register page

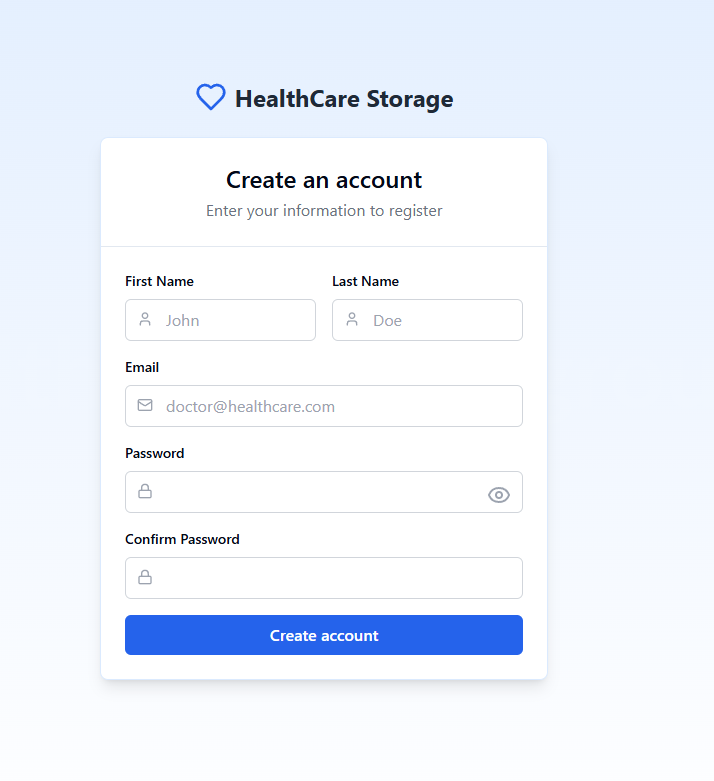


Figure 5.11: Register Page

### Patients Login page

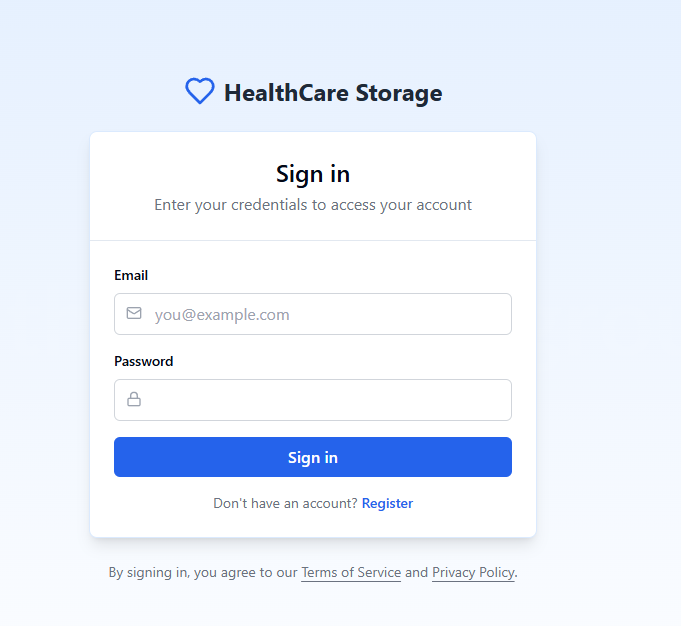


Figure 5.12: Patients Login Page

### Patients Home page

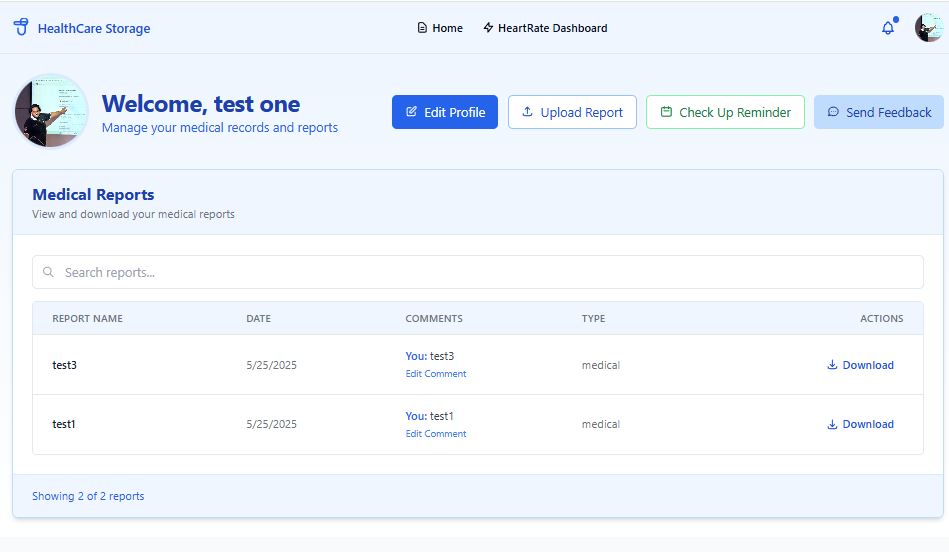


Figure 5.13: Patients Home Page

### Heart Rate Dashboard page

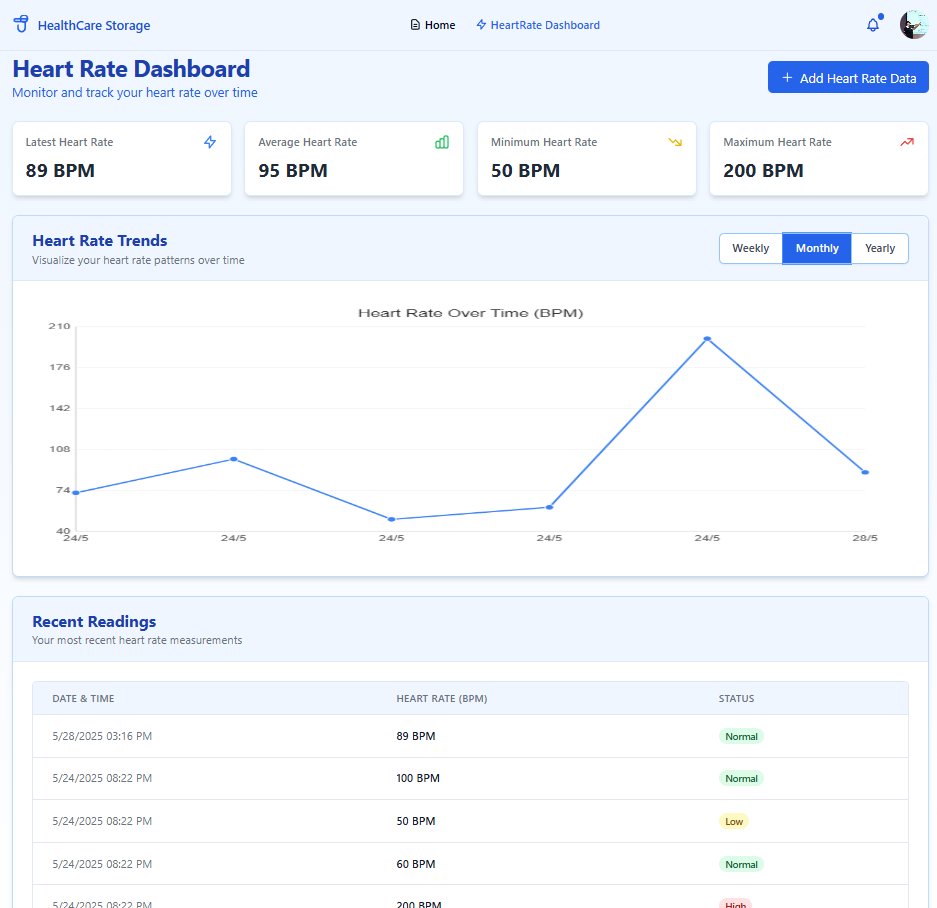


Figure 5.14: Heart Rate Dashboard Page

### Admin Login page



Figure 5.15: Admin Login Page

### Admin Dashboard page

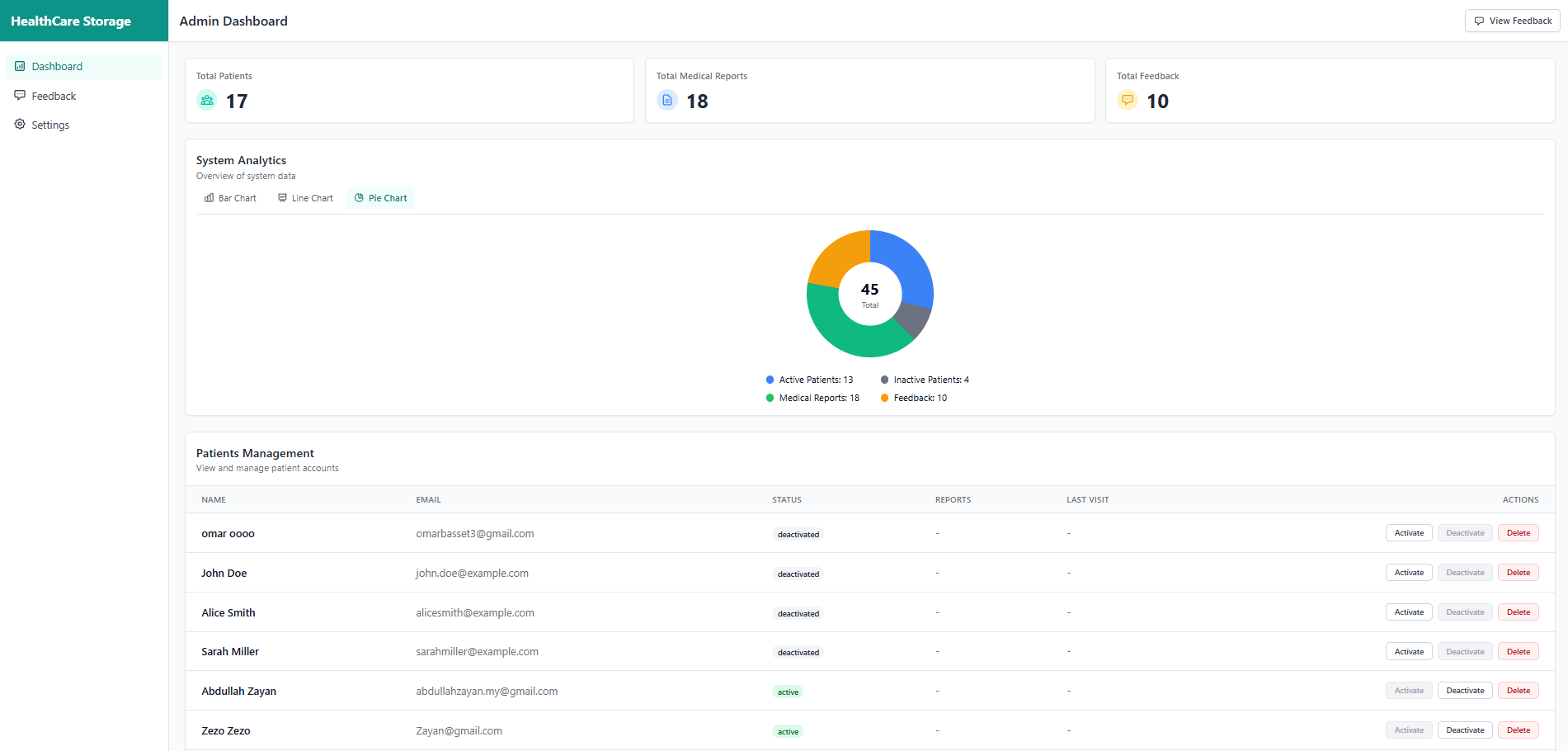


Figure 5.16: Admin Dashboard Page

## Testing

Testing the system during the HealthCare Storage System project is essential to confirm that it works as expected, provides a good experience, and meets every requirement set in the beginning. At this point, we need to make sure that secure login, manual heart rate input, and visualization, as well as reminders for medical check-ups are working correctly and precisely. The main methods used in testing the system are white-box testing to check the code’s logic, black-box testing to check how the system works from the outside, and user testing to see how users interact with it.

### White Box Testing

While testing the HealthCare Storage System, white box testing concentrated on the internal workings, design, and control sequence of the code. With this testing method, the development team could check and confirm that each module worked properly and was interacting well with the others.

Back-end features such as patient registration and login, uploading medical reports, entering heart rate manually, and setting notifications by date were subjected to white box testing. As part of testing, the logic inside these functions was reviewed to confirm that they stored the user’s data, correctly checked the vital details, and returned records to the user as needed.

In the medical check-up reminder system, the testing made sure that the reminder would only be sent after one year since the user’s check-up date. Tests were run to check that invalid dates, missing data, and expired tokens were handled by the application with error messages or by denying access. Unit testing and integration testing were applied to check the internal processes. Unit tests checked each function in isolation, and integration tests checked the way various modules interacted, for example, when user authentication was used with report upload and dashboard generation. Because of this thorough testing, any logical errors were found and fixed early, performance got better, and every path in the system was tested, making it more secure.

### Black Box Testing

To check the external behavior of the HealthCare Storage System, black box testing was used, where the system’s responses to users were examined without considering the code inside. The testing involved checking how the system behaved with different inputs, compared the results expected, and checked the real outcomes for important features.

In black box testing, test scenarios were made to mimic a person registering, logging in, uploading records, entering heart rate and time, and setting the last check-up date to receive the one-year reminder. The system was tested to see how input is checked, what kind of error messages appear, and how fast it responds.

Much care was taken to check that the heart rate dashboard was easy to use and that all user-entered values were correctly shown in the chart. In addition, the feature that reminds users of check-ups was tested by entering different dates to make sure notifications were sent on time.

With this kind of testing, it was confirmed that all interactive parts of the user interface, such as buttons, forms, modals, and alerts, behaved the same across all browsers and devices. It was confirmed that users get suitable messages right away, and the system feedback is easy to understand.

Both system and acceptance testing were included under black box testing in this project, with the application being checked from start to finish to meet the needed functional requirements. The tests showed that the HealthCare Storage System meets the user’s expectations, regardless of the way it is implemented.

### User Testing

The main goal of user testing was to check how real users use the HealthCare Storage System, to hear their feedback, and assess the system’s usability and how easy it is to use. This testing method made it possible to see how real users use the system, notice any difficulties, and judge how happy they are with the website’s design.

People were required to register, log in, add their medical records, enter their heart rate and the time it was taken, and set a date for their last medical check-up to receive reminders. By doing these activities, we were able to understand the system’s main functions and check the ease of using it.

The participants’ feedback showed that things that could be better were the instructions on the forms, how easy it was to notice notifications, and how well the dashboard adapted to different screen sizes. People found the system simple to operate, but a few of them said it would be better if the heart rate fields were labeled more clearly and if they got a visual confirmation when scheduling a reminder.

To perform usability studies, users were observed while they carried out the tasks, and their actions, struggles, and interactions were noted. Besides, guests were given forms to complete, where they could give numbers for their satisfaction and also provide comments for change.

Testing the system with users was essential to make it simple, suitable for its users, and meet their needs and expectations. Table 5.1 shows the criteria used to test usability, and Table 5.2 gives the details of the users who were involved. The testing was carried out with some students and faculty members from Universiti Teknologi Malaysia so that the feedback could be diverse.

Table 5.1 User Testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case ID** | **Test Case** | **Expected Results** | **Actual Results** | **Pass/ Fail** |
| **TC001** | Register | System creates new patient account and displays success message. | New patient account successfully created and confirmation message displayed. | Pass |
| **TC002** | Login | System authenticates user and redirects to dashboard. | User successfully logged in and redirected to respective dashboard. | Pass |
| **TC003** | Edit Account | System updates user profile information and shows confirmation. | User details updated in database and confirmation message shown. | Pass |
| **TC004** | Upload Medical Reports | System accepts valid PDF, stores it, and shows success message. | Medical report successfully uploaded and saved in database. | Pass |
| **TC005** | Download Medical Reports | System retrieves report and triggers browser download. | Report successfully downloaded by the user. | Pass |
| **TC006** | Send Feedback Reports | Feedback is saved in database and confirmation message is displayed. | Feedback successfully submitted and saved in the system. | Pass |
| **TC007** | Add Comments to Test Results | System saves comment under selected report and displays confirmation. | Comment successfully added and linked to appropriate report. | Pass |
| **TC008** | View Analysis Heart Rate Dashboard | Graphical representation of heart rate data is displayed based on user input. | Heart rate trends correctly shown on dashboard in graph format. | Pass |
| **TC009** | Receive Check-Up Reminder | Reminder is triggered and shown if one year has passed since last check-up date. | Notification displayed based on user-entered check-up date. | Pass |
| **TC010** | View Reports Dashboard | Admin views all uploaded reports and analytics in chart or table format. | Admin dashboard displays report summaries and charts correctly. | Pass |
| **TC011** | Read Feedback Reports | Admin retrieves and reads all patient-submitted feedback from the database. | Feedback list shown in readable format with patient details. | Pass |
| **TC012** | Deactivate/Activate Accounts | Admin successfully changes account status and confirmation is shown. | User accounts were activated or deactivated as selected by admin. | Pass |
| **TC013** | Chatbot Assistance | Chatbot responds with appropriate instructions based on user query. | Chatbot successfully replied to various queries with correct guidance. | Pass |

Table 5.2 Users for User Testing

|  |  |  |  |
| --- | --- | --- | --- |
| User | Phone Number | Matric | Signature |
| Majd | +966 53 570 7162 | A21EC4049 |  |
| Mohannad jebril | +974 7775 5441 | A21EC4027 |  |

### Non-Functional Requirements Testing

Table 5.3 Non-Functional Requirements Testing

|  |  |
| --- | --- |
| **NFR** | **Testing Details** |
| **Performance** | Load and response tests were carried out to make sure the HealthCare Storage System can work for many users at once without affecting its performance or speed. The tests included activities such as logging in, uploading and downloading reports, giving feedback, and using the dashboard at the same time. The system’s speed, capacity, and resource use were checked to see if any performance problems were present. |
| **Security** | Protecting sensitive patient data is made possible through security. The system was checked by using user authentication (JWT) and password encryption (bcrypt.js). The checks made sure that only users with permission could use the protected endpoints. The data saved in MongoDB was examined to check if it was encrypted properly. Tests were done on the API routes to see if unauthorized users were denied and input validation was always followed. |
| **Usability** | People from UTM were selected to take part in usability testing. People in the trial checked basic features, including how to handle their accounts, submit reports, and look at heart rate data. People were asked to comment on how easy it was to understand the interface, read the input labels, and use all the site’s pages. The information was applied to improve the way the application looks and how easy it is for new users to learn. |
| **Scalability** | The system was checked to make sure it could handle more data entries and users without any problems. Tests were done by letting users record many heart rate values and submit several reports. Performance of the backend and database was checked, and it was found that the system can handle more demand by scaling both horizontally and vertically. |

Because of these non-functional tests, it was confirmed that the HealthCare Storage System met its essential functions and could be trusted, secured, and used comfortably by anyone, now and in the future. Table 5.3 gives an overview of the important non-functional aspects tested and their outcomes.

* 1. **Chapter Summary**

This chapter explained in detail how the HealthCare Storage System was implemented and tested, which involved building its main functions, designing the user interface, and carefully checking its reliability and performance. With the system, patients can safely handle their medical information, enter their heart rate, get reminders for check-ups, and communicate with the administrators.

The developer created and merged important parts such as user login, managing medical reports, a heart rate dashboard, reminder alerts, and processing feedback. React.js was used in the front-end to offer a responsive and easy-to-use interface, while Node.js with Express.js managed server-side functions on the backend and MongoDB kept the data.

The software was tested for its functions by means of white box testing, black box testing, and user testing. The testing approaches confirmed that the system performed as expected in different cases and that the user interface was uncomplicated and dependable. Non-functional tests were done to check performance, security, usability, and scalability.

As a result of their work, the health record management system is both useful for patients and safe and secure.



**CONCLUSION**

* 1. **Introduction**

This chapter gives an overview of the objectives and outcomes of the proposed HealthCare Storage System in line with the objectives of the project. The chapter closes all the discussions made in the earlier chapters on the system objectives, literature review, methodologies, requirement analysis, and design.

The focus of the project is to identify, architect, implement, and validate a web application through which patients can maintain their medical records and interact with the functionality such as health alerts and health trend visualization. This chapter will also summarise the findings of the completed stages in PSM1 and detail the further steps to be taken in PSM2 to guarantee the successful implementation and deployment of the system.

* 1. **Achievement of Project Objectives**

Another accomplishment of this project is the specification and description of user needs in the HealthCare Storage System. In this case, the critical features of the system include; secure medical report management, check-up reminders, and health trend analysis that was established through interviews and document analysis to meet the patients’ needs. These were the requirements that were used in the formulation of the design and operation of the system.

In Chapter 2, a literature review was performed to assess the current state of healthcare systems including Sehhaty and Huawei Health. The gaps identified in the current systems include low availability and absence of tools designed with the user in mind, which the proposed system seeks to fill. This evaluation was helpful in determining the features and goals of the system. Some of the goals of designing and documenting the system were met in this phase. Use case, sequence, class, and activity diagrams were created to represent the structure of the system and its interactions. The first phase of database construction was carried out, which would serve as the basis for storing and protecting medical records.

Nevertheless, the construction and the evaluation of the system could not be carried out in this phase. These activities will be focused and performed in the PSM2 phase to achieve the complete implementation and verification of the HealthCare Storage System.

* 1. **Suggestions for Future Improvement**

The HealthCare Storage System will be completed in PSM2 with all the requirements and designs that were set in PSM1. The system will pass through the design, development and testing phase to ensure all of these functions including medical report management, check-up reminders and health dashboards are smoothly integrated. The waterfall model that was discussed in chapter 3 will be applied in PSM2 to ensure that each stage of the development process is done effectively to the next stage. In addition, there will be thorough testing in order to find weaknesses and failures in them in order to enhance system efficiency and functionality. Feedbacks from users and stakeholders will also be considered to further improve the system as well. The end product should be a secure, easy to use, and efficient system that fulfills all the stated aims and objectives and most importantly, the patients.

* 1. **Chapter Summary**

Chapter 5 is the last chapter of the PSM1 report for the HealthCare Storage System and it presents the assessment of the work done in this phase, namely the requirements and the design of the system. The goals that will be retained, including the development and testing of the system fully, will be achieved in PSM2. Possible enhancements in the subsequent phases and measures to guarantee the successful completion of the project were also considered. This chapter concludes the work done in PSM1 and establishes the context for the complete implementation of the system in PSM2.

**REFERENCES**

Royce, W. W. (1970). Managing the Development of Large Software Systems. *Proceedings of IEEE WESCON*, 1(26), 1–9.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley.

Boehm, B. W. (1981). *Software Engineering Economics.* Prentice Hall.

Bass, L., Clements, P., & Kazman, R. (2003). *Software Architecture in Practice.* Addison-Wesley.

Duvall, P. M., Matyas, S., & Glover, A. (2007). *Continuous Integration: Improving Software Quality and Reducing Risk.* Addison-Wesley.

Beck, K. (2003). *Test-Driven Development: By Example.* Addison-Wesley.

Monitoring patients via a secure and mobile healthcare system. (2010, February 1). *IEEE Journals & Magazine*. Retrieved from <https://ieeexplore.ieee.org>.

A medical healthcare system for privacy protection based on IoT. (2015, December 1). *IEEE Conference Publication*. Retrieved from <https://ieeexplore.ieee.org>.

A Secure Healthcare System Design Framework using Blockchain Technology. (2019, February 1). *IEEE Conference Publication*. Retrieved from <https://ieeexplore.ieee.org>.

Appendix A

**Software Requirements Specification**

**(SRS)**

1. **INTRODUCTION**

This SRS document provides further details of the HealthCare Storage System. The document also provides the clear description of the functional and non-functional requirements of the system. It depicts the relation between the users and the functions of the system through use case diagram, sequence diagram, activity diagram and class diagram.

* 1. **Purpose**

The purpose of this Software Requirements Specification (SRS) is to:

a. Identify and analyze the functional and non-functional requirements of the HealthCare Storage System.

b. Provide an overview of how the system operates, including user interactions and system processes.

c. Serve as a reference document for the development of the Software Design Document (SSD).

d. Ensure that the final system meets the defined requirements through proper validation.

* 1. **Scope**

The software product under development is the HealthCare Storage System which is a web application that will assist the patient in online record keeping of their health records and also improve the experience of the patient about his/her health. The system will provide the following functionalities:

1. It permits patients to upload, store, and download their medical reports in easily accessible and safely protected space.
2. Allow patients to be notified of their health checks up based on their records again.
3. Forecast Provide a way of presenting health early parameters such as heart rate trends over time to patients and the health care givers.
4. Enable patients to append comment to tests and give feedback to the system administrators to improve on it.
5. The system will not be used as a diagnostic or therapeutic instrument but as a record keeping and analysis tool only.

This application will help in minimizing the chances of misplaced medical records, enhance patient’s involvement in their treatment process and offer a dependable source of medical information for any time and any place. It complements the objectives of promoting security, utility, and patient accessibility of their health information.

* 1. **Definitions, Acronyms and Abbreviation**

|  |  |
| --- | --- |
| **Term** | **Definition** |
| SRS | Software Requirements Specification |
| SSD | Software Design Document |
| JWT | JSON Web Token |
| UI | User Interface |
| MongoDB | A NoSQL database used for storing and retrieving application data |
| Node.js | A JavaScript runtime environment used for backend development |
| React.js | A JavaScript library for building dynamic user interface |
| HealthCare Storage System | The web-based platform developed to manage and analyze medical recoreds |
| Patient | The user who will use the system |
| Admin | The user who will be responsible for managing the system |

* 1. **Overview**

This document consists of 2 parts:

1. Introduction

This section gives a brief description of the HealthCare Storage System, and the areas that the research will cover, and the users that will benefit from it. It also contains explanations of the terms and abbreviations used in the document.

1. Specific Requirement

This section defines the functional requirements of the system, the description of the interfaces of the system and diagrams to illustrate the users and the system.

This structure helps to achieve a systematic understanding of the system and provides a reference for its construction and verification.

* 1. **Reference**

Sommerville, I. (2010). Software Engineering (9th ed.). Addison Wesley.

1. **Specific Requirements**

In this section, the functional and non-functional requirements are presented. A use case specification and its corresponding sequence and activity diagram are also included.

* 1. **User characteristics**

The HealthCare Storage System will be used by two main types of users: Patients and Admins.

* + 1. Patients
* Patients are expected to have basic computer skills, including familiarity with web-based applications and the ability to navigate a user-friendly interface.
* It might mean the patient simply has different degrees of literacy in information technologies but they should be able to fulfill activities such as uploading and downloading medical reports or commenting these reports.
* The patients will be able to store their medical records and track their health, set reminders for check-ups and view visualized trends such as heart rate analysis.
* They will give their identity details like name and email address when creating an account. The system will have privacy and data security so that important medical data will not be exposed to other individuals.
  + 1. Admins
* Everyone should have intermediate level of technical knowledge to maneuver within the admin panel and perform tasks of system administration.
* They will apply the system for patient accounts management, feedback, and reports checking through the protected panel.
* Users interact with the system and it is the duty of the admins to keep the system running effectively.

Two types of users will be using the system and both will access the system through a web-based to minimize the risk of data breaches while making ensure for data privacy and security.

* 1. **System Features**

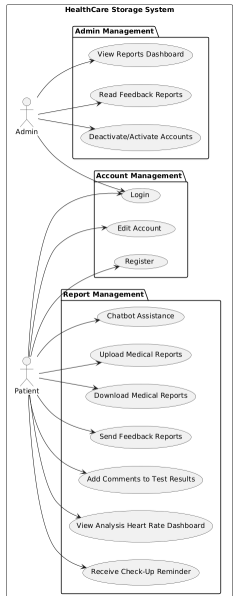


Figure 2.2.1 Use Case Diagram for HealthCare Storage System

Table 2.2.: Description of Module and Functions for <HealthCare Storage System>

|  |  |  |
| --- | --- | --- |
| **Module** | **Function** | **Description** |
| Patients Module | UC001 - Register | Enables a new patient to sign up for the platform. |
| Users Module | UC002- Login | Allows users to login into their accounts through their credentials securely. |
| Patients Module | UC003- Edit Account | Enables patients to change their details that may be in the account. |
| Patients Module | UC004- Upload Medical Reports | Allows patients to upload their medical records in the system. |
| Patients Module | UC005- Download Medical Reports | Enables patients to download their medical reports that they had uploaded earlier. |
| Patients Module | UC006- Send Feedback Reports | Allows the patients to forward their comments or complaints about the system to the admin. |
| Patients Module | UC007- Add Comments to Test Results | Enables patients to add notes to their medical test results to help them remember in the future. |
| *‘*Patients Module | UC008- View Analysis Heart Rate Dashboard | Helps patients to have a visual understanding of the patterns of their heart rates by analysing the input that the patients put. |
| Patients Module | UC009- Receive Check-Up Reminder | Informs patients automatically for the need of medical check-ups based on the last date they put. |
| Admin Module | UC010- View Reports Dashboard | Enables the admin to see specific or general reports about the use and performance of the system. |
| Admin Module | UC011- Read Feedback Reports | Allows the admins to view the feedback that patients have given. |
| Admin Module | UC012- Deactivate/Activate Accounts | Enables the administrators to have control over the users’ account and especially when one needs to be activated or deactivated. |
| Patients Module | UC013- Chatbot Assistance | Offers patients a quick, automated help service for resolving queries related to system functionality, such as how to upload reports or navigate the platform. |
|  |  |  |

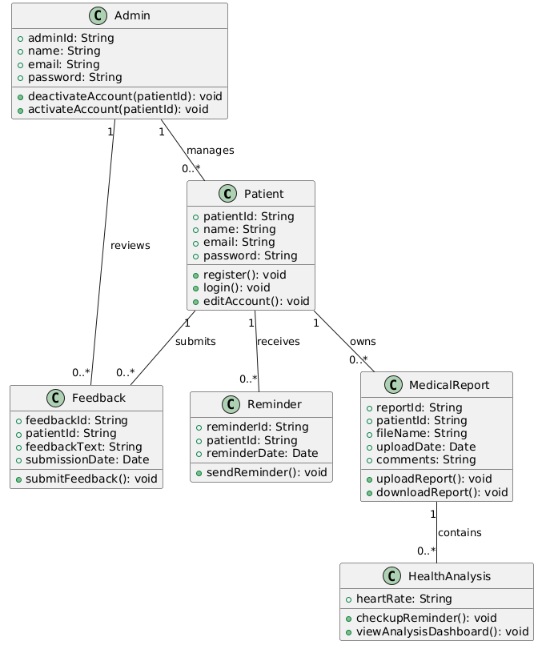


Figure 2.2.2: Domain Model for <HealthCare Storage System>

* 1. **Use Case Details**
     1. **UC001: Use Case <Register>**

**Table 2.3.1: Use Case Description for <Register>**

|  |  |
| --- | --- |
| **Use Case ID** | UC001 |
| **Use Case Name** | Register |
| **Description** | This case will show how a new user or patients register to the HealthCare Storage System by provide the information. |
| **Actor(s)** | Patients. |
| **Pre-condition(s)** | The patients must have the access to internet and go the system in registration page. |
| **Normal Flow(s)- NF** | 1. The patient goes to the registration page.  2. The patient inputs their full name, email address and password.  3. The patient fills the registration form.  4. The system checks on the correctness of the entered information.  5. The system then opens a new patient’s account and saves the details in the database.  6. The system then produces a success message to the patient. |
| **Alternative Flow(s) - AF** | AF1.The patient enters invalid or incomplete information:  - The system will tell the patient to correct the information. |
| **Exception Flow(s) - EF** | EF1. System didn’t success while saving data:  - The system displays an error message and asks the patient to try again later. |
| **Post-condition(s)** | A new patient account is created and stored in the system's database. |

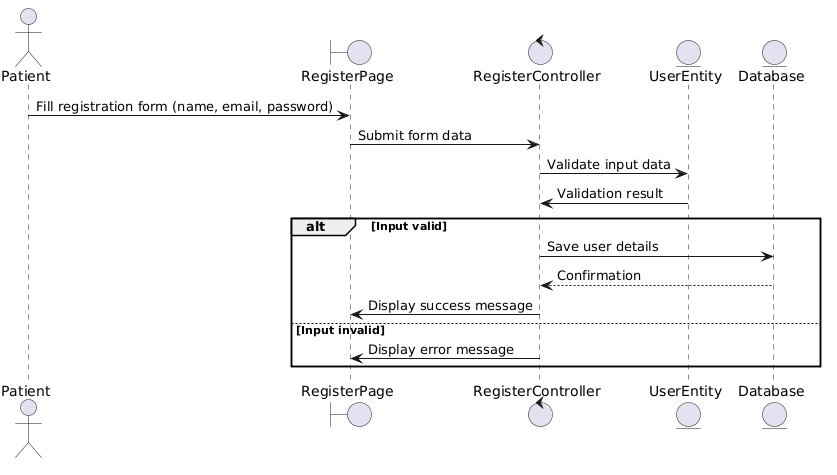


Figure 2.3.1: Sequence Diagram for <Register>

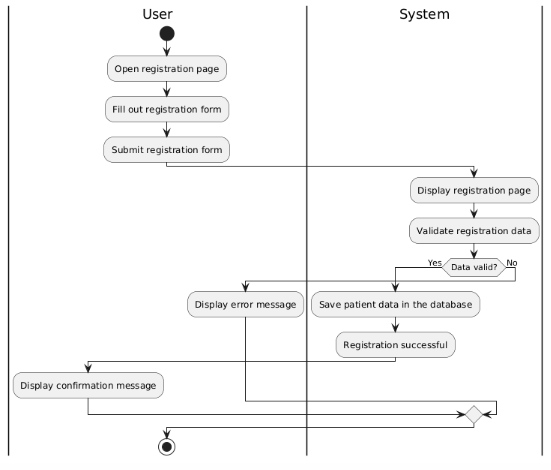


Figure 2.3.2: Activity Diagram for <Register>

* + 1. **UC002: Use Case <Login>**

**Table 2.3.2: Use Case Description for <Login>**

|  |  |
| --- | --- |
| **Use Case ID** | UC002 |
| **Use Case Name** | Login |
| **Description** | This case will show how registred patient and admin logs into HealthCare Storage System and access to their account |
| **Actor(s)** | Patients, Admin |
| **Pre-condition(s)** | The patients and admin must have account to enter the system |
| **Normal Flow(s)- NF** | 1. The user goes to the login page.  2. The user enters his or her email and password.  3. The user enters the login form.  4. The system checks the credentials that were entered in the system.  5. If valid, the system allows the user into the system and the user is taken to their home page while admin go to another specific dashboard |
| **Alternative Flow(s) - AF** | |  | | --- | | AF1. Invalid credentials: |  |  |  | | --- | --- | |  | - The system displays an error message and make the user to re-enter the credentials. | |
| **Exception Flow(s) - EF** | |  | | --- | | EF1. System failure during validation: |  |  |  | | --- | --- | |  | - The system displays an error message and asks the user to try again later. | |
| **Post-condition(s)** | The user will successfully logged in and go to specific page. |

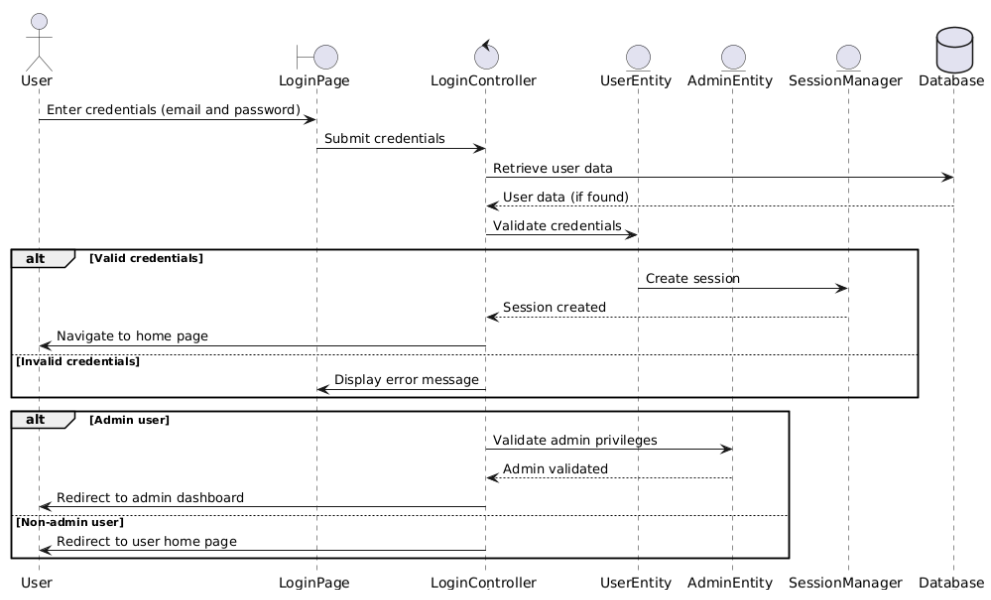
****

Figure 2.3.3: Sequence Diagram for <Login>

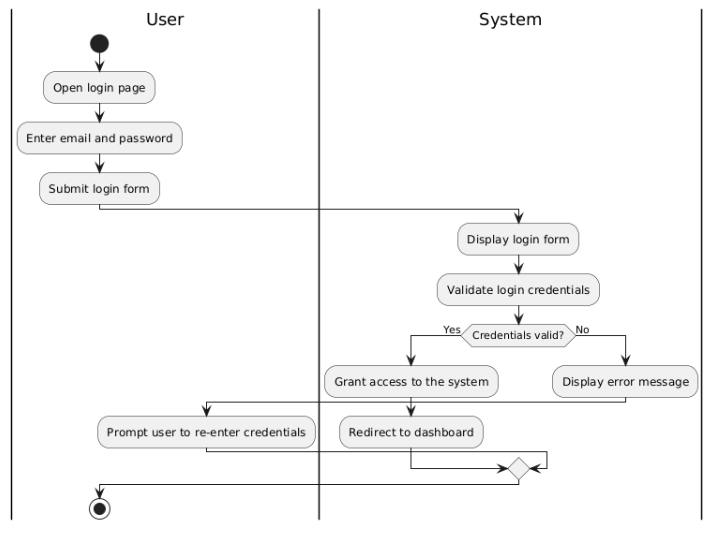


Figure 2.3.4: Activity Diagram for <Login>

* + 1. **UC003: Use Case <Edit Account>**

**Table 2.3.3: Use Case Description for <Edit Account>**

|  |  |
| --- | --- |
| **Use Case ID** | UC003 |
| **Use Case Name** | Edit Account |
| **Description** | This case will show how the user can update his account information for example name, email |
| **Actor(s)** | Patients |
| **Pre-condition(s)** | The patients must log in to the system |
| **Normal Flow(s)- NF** | 1. The user go to the Edit account.  2. The system shows the user’s account details.  3. The user changes the fields that wants to modify.  4. The user sends the new information to the system.  5. The system will check the new information that entered.  6. The system stores the changes made in the database.  7. In the end, the system shows a message to the user to confirm the input given by him. |
| **Alternative Flow(s) - AF** | |  | | --- | | AF1. The user enters invalid or incomplete information: |  |  |  | | --- | --- | |  | - The system will tell the user to correct the information. | |
| **Exception Flow(s) - EF** | |  | | --- | | EF1. System failure during validation: |  |  |  | | --- | --- | |  | - The system displays an error message and asks the user to try again later. | |
| **Post-condition(s)** | The user’s account information is updated and stored in the system’s database. |

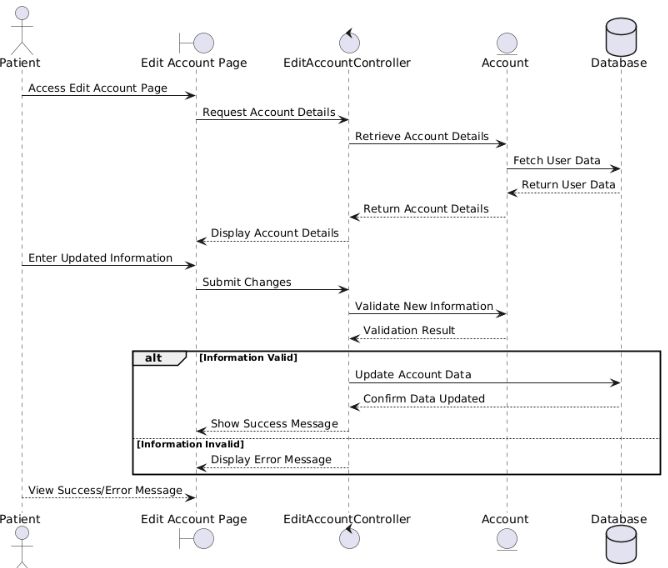
****

Figure 2.3.5: Sequence Diagram for <Edit Account>

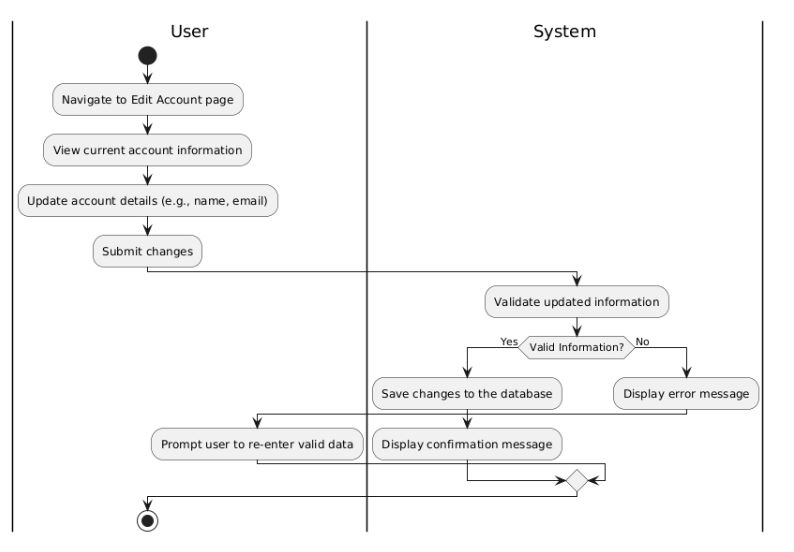
****

Figure 2.3.6: Activity Diagram for <Edit Account>

* + 1. **UC004: Use Case <Upload Medical Reports>**

**Table 2.3.4: Use Case Description for <Upload Medical Reports>**

|  |  |
| --- | --- |
| **Use Case ID** | UC004 |
| **Use Case Name** | Upload Medical Reports |
| **Description** | This case will show how patients upload their medical reports to HealthCare Storage System |
| **Actor(s)** | Patients |
| **Pre-condition(s)** | The patients must log in to the system |
| **Normal Flow(s)- NF** | 1. The patient is click to the Upload Medical Reports button.  2. The system shows the upload form.  3. The patient uploads medical report files.  4. The patient clicks the Confirm or Upload button.  5. The system checks the file format and its size.  6. The system will store the file uploaded and saved in the database.  7. The system shows a success message. |
| **Alternative Flow(s) - AF** | |  | | --- | | AF1. The patient selects an invalid file format or exceeds size limit: |  |  |  | | --- | --- | |  | - The system displays an error message and ask the patient to re-upload valid files. | |
| **Exception Flow(s) - EF** | |  | | --- | | EF1. System failure during validation: |  |  |  | | --- | --- | |  | - The system displays an error message and asks the user to try again later. | |
| **Post-condition(s)** | The medical report is successfully uploaded and stored in the system’s database. |

****

Figure 2.3.7: Sequence Diagram for <Upload Medical Reports>

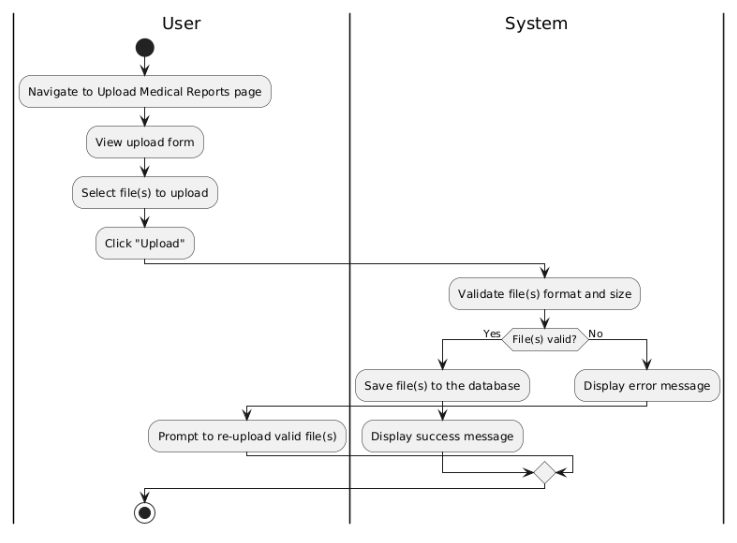


Figure 2.3.8: Activity Diagram for <Upload Medical Reports>

* + 1. **UC005: Use Case <Download Medical Report>**

**Table 2.3.5: Use Case Description for <Download Medical Reports>**

|  |  |
| --- | --- |
| **Use Case ID** | UC005 |
| **Use Case Name** | Download Medical Reports |
| **Description** | This case will show how patients download their medical reports to HealthCare Storage System |
| **Actor(s)** | Patients |
| **Pre-condition(s)** | The patients must log in to the system |
| **Normal Flow(s)- NF** | 1. The system presents a list of medical reports that are available.  2. The patient chooses the report of his/her choice in the format that will be downloaded.  3. The patient clicks on the “Download” button.  4. The system then pulls out the selected report from the database.  5. The system makes the report file downloadable. |
| **Alternative Flow(s) - AF** | |  | | --- | | AF1. No reports available for download: |  |  |  |  |  |  | | --- | --- | --- | --- | --- | |  | - The system displays a message stating no reports are available.   |  | | --- | | AF2. Download interrupted (network failure): |  |  |  | | --- | --- | |  | - The system displays an error message and allows the patient to retry. | | |
| **Exception Flow(s) - EF** | |  | | --- | | EF1. System fails to retrieve the report: |  |  |  | | --- | --- | |  | - The system displays an error message and asks the patient to try again later. | |
| **Post-condition(s)** | The selected medical report is successfully downloaded to the patient’s device. |

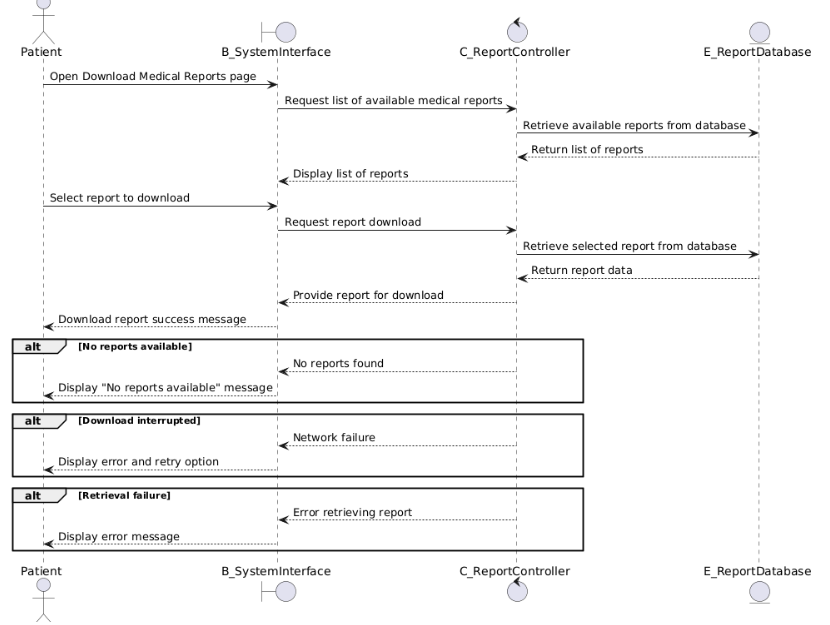
****

Figure 2.3.9: Sequence Diagram for <Download Medical Reports>

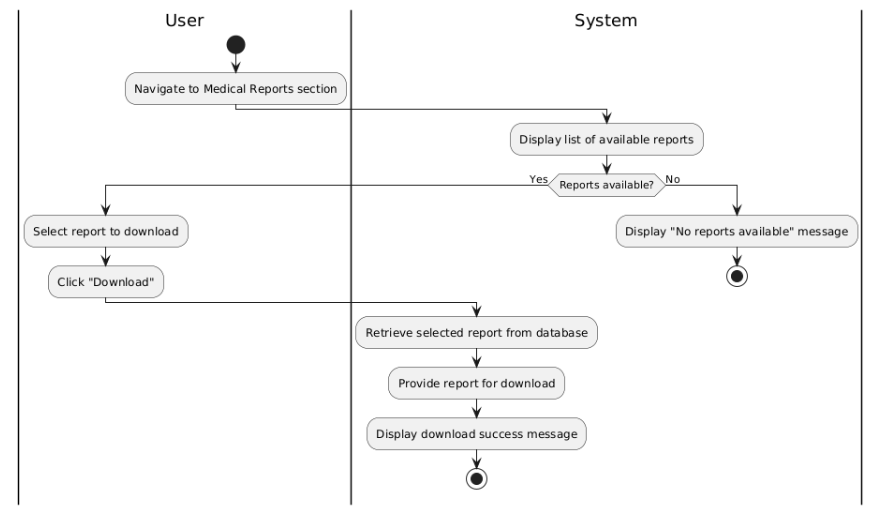


Figure 2.3.10: Activity Diagram for <Download Medical Reports>

* + 1. **UC006: Use Case <Send Feedback Report>**

**Table 2.3.6: Use Case Description for <Send Feedback Reports>**

|  |  |
| --- | --- |
| **Use Case ID** | UC006 |
| **Use Case Name** | Send Feedback Reports |
| **Description** | This case will show how patients can send the feedback reports to the HealthCare Storage System |
| **Actor(s)** | Patients |
| **Pre-condition(s)** | The patients must log in to the system |
| **Normal Flow(s)- NF** | 1. The patient navigates to the "Feedback" section.  2. The patient fills out the feedback form with relevant details.  3. The patient clicks the "Submit" button.  4. The system validates the feedback content.  5. The system saves the feedback to the database and displays a success message. |
| **Alternative Flow(s) - AF** | AF1. Feedback validation fails:  - The system displays an error message prompting the patient to correct invalid input. |
| **Exception Flow(s) - EF** | EF1. System fails to save feedback:  - The system displays an error message and asks the patient to retry later. |
| **Post-condition(s)** | The feedback is successfully submitted and saved in the database. |

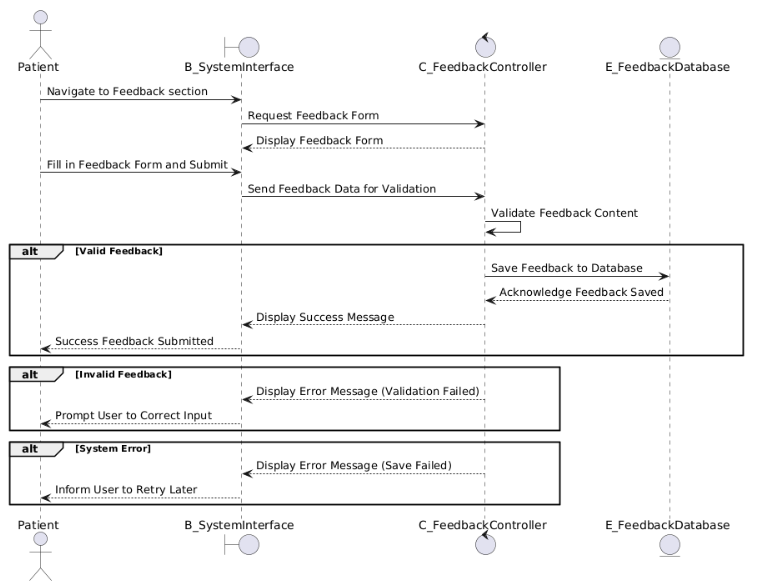
****

Figure 2.3.11: Sequence Diagram for <Send Feedback Reports>

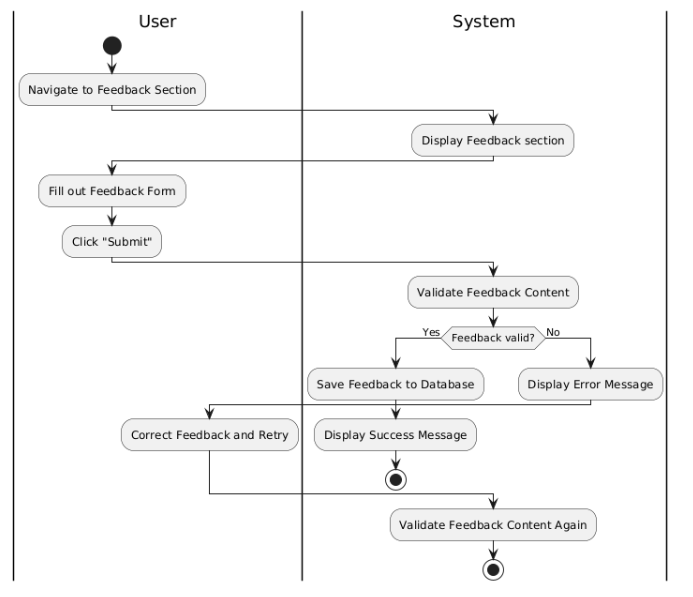


Figure 2.3.12: Activity Diagram for <Send Feedback Reports>

* + 1. **UC007: Use Case <Add Comments to test result>**

**Table 2.3.7: Use Case Description for <Add Comments to test result>**

|  |  |
| --- | --- |
| **Use Case ID** | UC007 |
| **Use Case Name** | Add Comments to test result |
| **Description** | This case will show how patients can add comments to theirs test result or reports in the HealthCare Storage System. |
| **Actor(s)** | Patients |
| **Pre-condition(s)** | 1. The patients must log in to the system.  2. The patients must have at least available test result or report. |
| **Normal Flow(s)- NF** | 1. The patients go to the test or report section. 2. The system shows the list of reports or test. 3. The patients choose specific report. 4. The patient types in a comment in the box that is provided for the comment. 5. The patient clicks on the Submit Comment button. 6. The system checks the comment and stores the comment in the database as valid. 7. The system shows a success message to the patient. |
| **Alternative Flow(s) - AF** | AF1. No test results available:  - The system displays a message stating that there are no test results to comment on. |
| **Exception Flow(s) - EF** | EF1. Comment submission fails due to server error:  - The system displays an error message and allows the patient to retry. |
| **Post-condition(s)** | The comment is successfully added to the selected test result and stored in the database. |

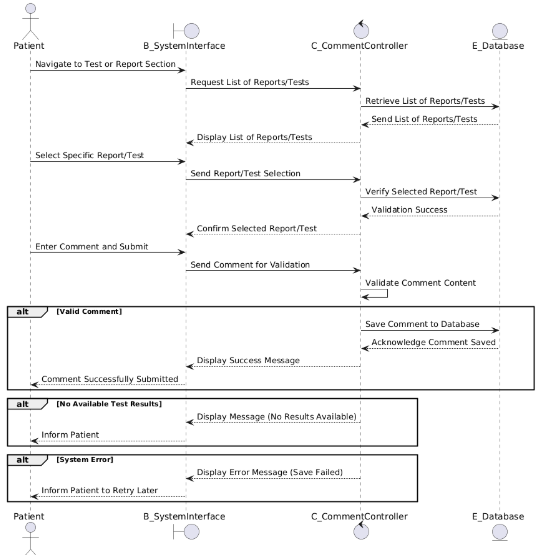


Figure 2.3.13: Sequence Diagram for <Add Comments to test result>

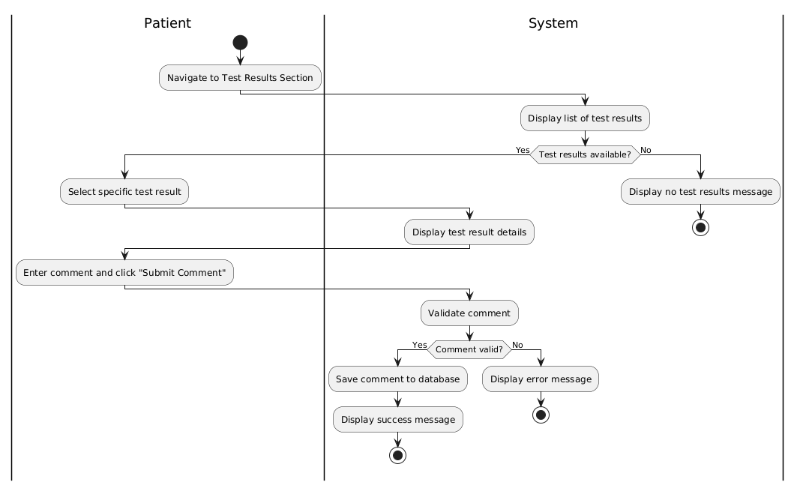


Figure 2.3.14: Activity Diagram for <Add Comments to test result>

* + 1. **UC008: Use Case <** **View Analysis Heart Rate Dashboard>**

**Table 2.3.8: Use Case Description for <** **View Analysis Heart Rate Dashboard>**

|  |  |
| --- | --- |
| **Use Case ID** | UC008 |
| **Use Case Name** | View Analysis Heart Rate Dashboard |
| **Description** | This case will show how patients view the heart rate analysis dashboard, where uploaded heart rate data is processed and presented in a graphical format for easy analysis. |
| **Actor(s)** | Patients |
| **Pre-condition(s)** | 1. The patients must log in to the system.  2. The patient must have uploaded heart rate report. |
| **Normal Flow(s)- NF** | 1. The patient navigates to the "Heart Rate Dashboard" section. The system shows the list of reports or test. 2. The system then calls the heart rate data from the database. 3. The system takes the heart rate data and produces graphs as illustrations. 4. The graphical analysis is shown on the dashboard of the system. 5. The patient observes the graphical trends of the heart rate. |
| **Alternative Flow(s) - AF** | AF1. No heart rate data available:  - The system displays a message stating no data is available. - The patient is prompted to upload heart rate data. |
| **Exception Flow(s) - EF** | EF1. System failure to retrieve heart rate data:  - The system displays an error message and asks the patient to try again later. |
| **Post-condition(s)** | The patient successfully views the heart rate analysis dashboard. |

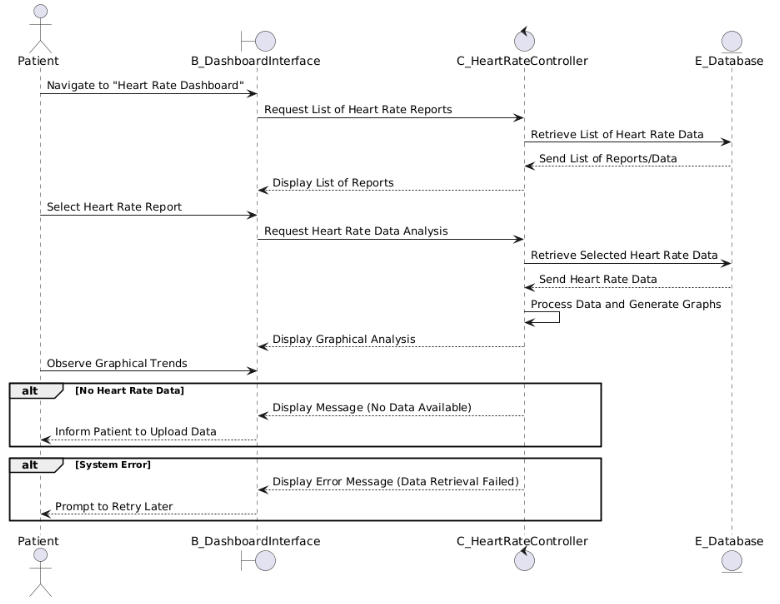
****

Figure 2.3.15: Sequence Diagram for < View Analysis Heart Rate Dashboard >

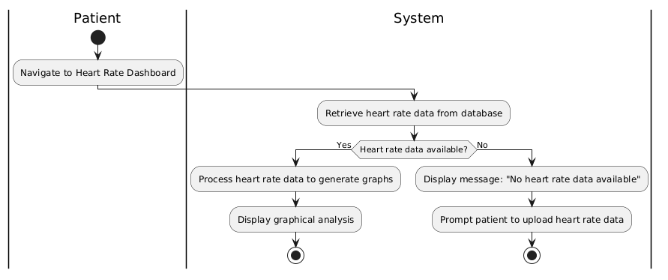


Figure 2.3.16: Activity Diagram for < View Analysis Heart Rate Dashboard >

* + 1. **UC009: Use Case <Receive Check-Up Reminder>**

**Table 2.3.9: Use Case Description for <** **Receive Check-Up Reminder>**

|  |  |
| --- | --- |
| **Use Case ID** | UC009 |
| **Use Case Name** | Receive Check-Up Reminder |
| **Description** | This case will show how patients will receive a notification or reminder when they are due for a medical check-up based on their last uploaded medical report. |
| **Actor(s)** | Patients |
| **Pre-condition(s)** | 1. The patients must log in to the system.  2. The patient must have uploaded a medical check-up report.  3. The system has scheduled reminders based on the uploaded medical report date. |
| **Normal Flow(s)- NF** | 1. The system checks the date of the patient's last medical report. 2. The system determines if a medical check-up is due. 3. The system sends a reminder to the patient via notification or email. 4. The patient receives and views the reminder. |
| **Alternative Flow(s) - AF** | AF1. No medical check-up report uploaded:  - The system cannot schedule a reminder and prompts the patient to upload a report. |
| **Exception Flow(s) - EF** | EF1. System error in sending the reminder: - The system logs the error and retries sending the reminder. |
| **Post-condition(s)** | The patient is reminded to schedule a medical check-up or is prompted to upload a medical report. |

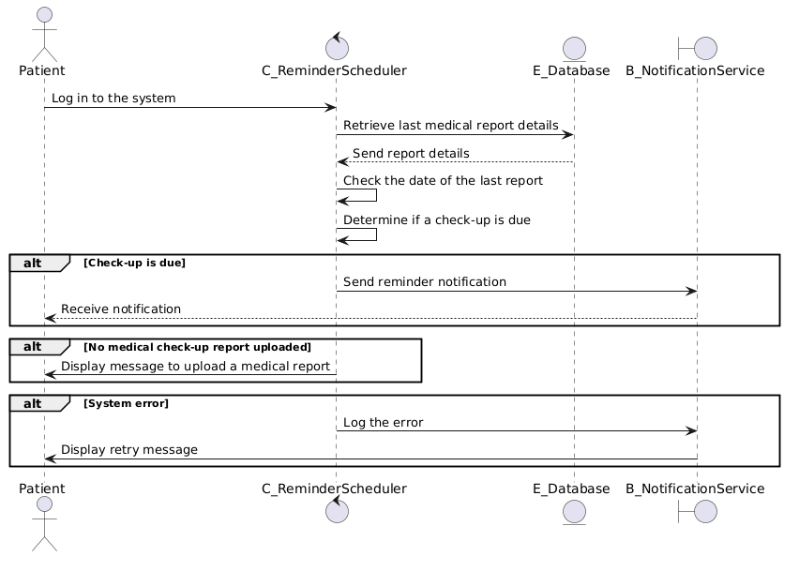
****

Figure 2.3.17: Sequence Diagram for < Receive Check-Up Reminder>

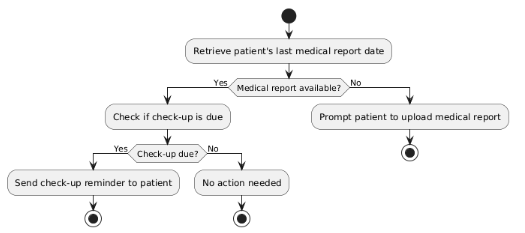


Figure 2.3.18: Activity Diagram for < Receive Check-Up Reminder >

* + 1. **UC010: Use Case <View Reports Dashboard>**

**Table 2.3.10: Use Case Description for <View Reports Dashboard>**

|  |  |
| --- | --- |
| **Use Case ID** | UC010 |
| **Use Case Name** | View Reports Dashboard |
| **Description** | This use case describes how the admin views the dashboard displaying patient reports and analytics in the HealthCare Storage System. |
| **Actor(s)** | Admin |
| **Pre-condition(s)** | 1. The admin must log in to the system. |
| **Normal Flow(s)- NF** | 1. The admin navigates to the Reports Dashboard section. 2. The system fetches the relevant patient reports and analytics data. 3. The system displays the dashboard with graphs, tables, or charts summarizing patient data. |
| **Alternative Flow(s) - AF** | AF1. No data available to display:  - The system displays a message stating No data available at this time. |
| **Exception Flow(s) - EF** | EF1. System error while fetching data: - The system displays an error message and logs the issue. |
| **Post-condition(s)** | The admin views the Reports Dashboard with the relevant data displayed successfully. |

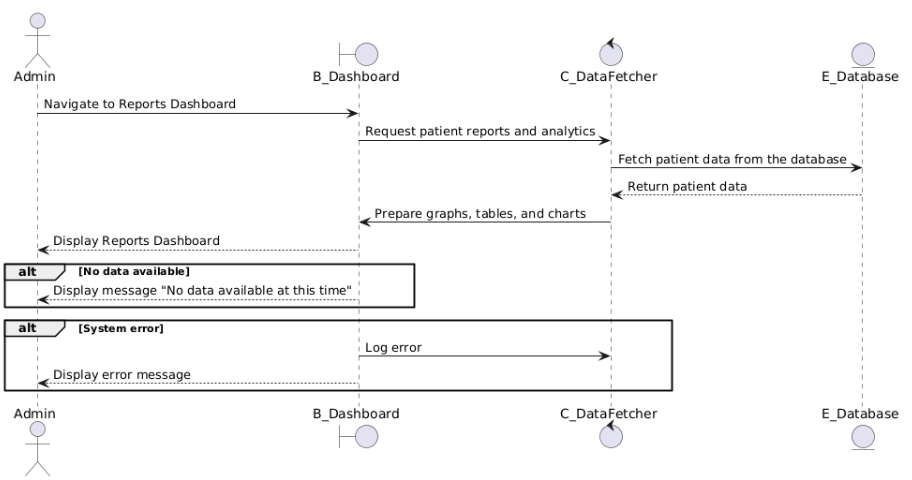
****

Figure 2.3.19: Sequence Diagram for <View Reports Dashboard>

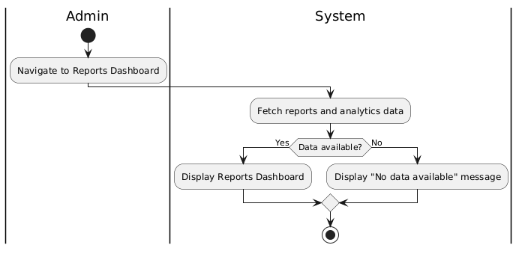
****

Figure 2.3.20: Activity Diagram for <View Reports Dashboard>

* + 1. **UC011: Use Case <Read Feedback Reports>**

**Table 2.3.11: Use Case Description for <Read Feedback Reports>**

|  |  |
| --- | --- |
| **Use Case ID** | UC011 |
| **Use Case Name** | Read Feedback Reports |
| **Description** | This use case describes how the admin can view feedback reports submitted by patients in the HealthCare Storage System. |
| **Actor(s)** | Admin |
| **Pre-condition(s)** | 1. The admin must log in to the system.  2. Feedback reports must be available in the database. |
| **Normal Flow(s)- NF** | 1. The admin navigates to the Feedback Reports section. 2. The system retrieves all feedback reports from the database. 3. The system displays the feedback reports in a readable format for the admin. |
| **Alternative Flow(s) - AF** | AF1. No feedback reports available: - The system displays a message stating No feedback reports available. |
| **Exception Flow(s) - EF** | EF1. System error while retrieving feedback: - The system displays an error message and logs the issue for review. |
| **Post-condition(s)** | The admin successfully views the available feedback reports. |

****

Figure 2.3.21: Sequence Diagram for <Read Feedback Reports>

****

Figure 2.3.22: Activity Diagram for <Read Feedback Reports>

* + 1. **UC012: Use Case <Deactivate/Activate Accounts>**

**Table 2.3.12: Use Case Description for <Deactivate/Activate Accounts>**

|  |  |
| --- | --- |
| **Use Case ID** | UC012 |
| **Use Case Name** | Deactivate/Activate Accounts |
| **Description** | This use case describes how the admin can deactivate or activate patient accounts in the HealthCare Storage System. |
| **Actor(s)** | Admin |
| **Pre-condition(s)** | 1. The admin must log in to the system.  2. The patient accounts must exist in the system. |
| **Normal Flow(s)- NF** | 1. The admin navigates to the Manage Accounts section. 2. The system retrieves a list of all patient accounts. 3. The admin selects an account to deactivate or activate. 4. The admin clicks the "Deactivate" or "Activate" button. 5. The system updates the status of the selected account in the database. 6. The system displays a confirmation message. |
| **Alternative Flow(s) - AF** | - |
| **Exception Flow(s) - EF** | EF1. System error while updating account status: - The system displays an error message and logs the issue for review. |
| **Post-condition(s)** | The selected account is successfully deactivated or activated, and the updated status is reflected in the system. |

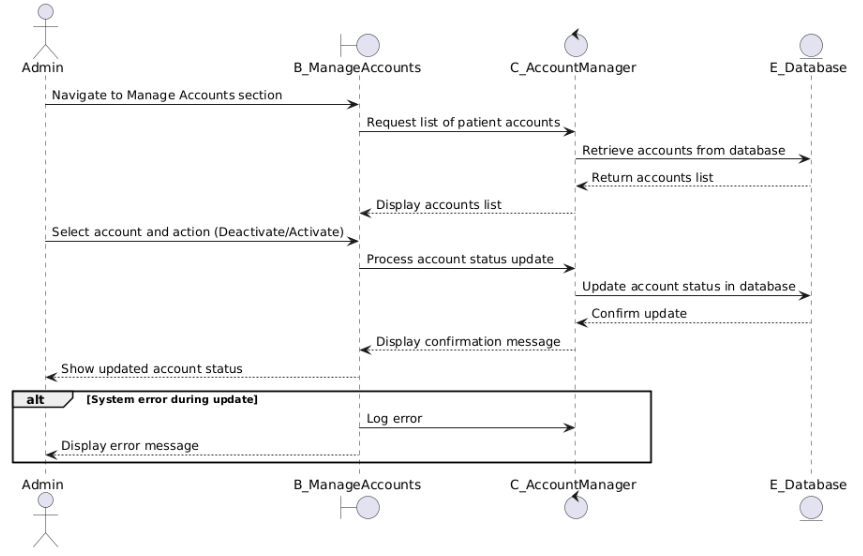
****

Figure 2.3.23: Sequence Diagram for < Deactivate/Activate Accounts>

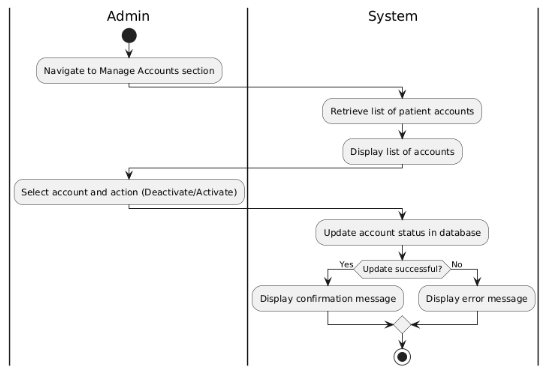
****

Figure 2.3.24: Activity Diagram for < Deactivate/Activate Accounts>

* + 1. **UC013: Use Case <Deactivate/Activate Accounts>**

**Table 2.3.13: Use Case Description for <Deactivate/Activate Accounts>**

|  |  |
| --- | --- |
| **Use Case ID** | UC012 |
| **Use Case Name** | Chatbot Assistance |
| **Description** | This use case allows patients to interact with an automated chatbot for assistance in understanding and navigating the system's functionalities. |
| **Actor(s)** | |  | | --- | |  |  |  | | --- | | Patients | |
| **Pre-condition(s)** | 1. The patient must be logged into the system.  2. The system must have the chatbot feature enabled. |
| **Normal Flow(s)- NF** | 1. The patient opens the chatbot from the dashboard. 2. The patient types a query (e.g., "How to upload a report?"). 3. The chatbot processes the query and fetches relevant information or steps. 4. The chatbot provides an appropriate response to the patient. |
| **Alternative Flow(s) - AF** | AF1: The chatbot cannot understand the query.  - The chatbot asks the patient to rephrase the question or provides contact information for further assistance. |
| **Exception Flow(s) - EF** | EF1: System error in chatbot service.  - The system displays an error message and logs the issue for system administrators to review. |
| **Post-condition(s)** | The patient receives the required assistance, or the chatbot provides further instructions for unresolved queries. |

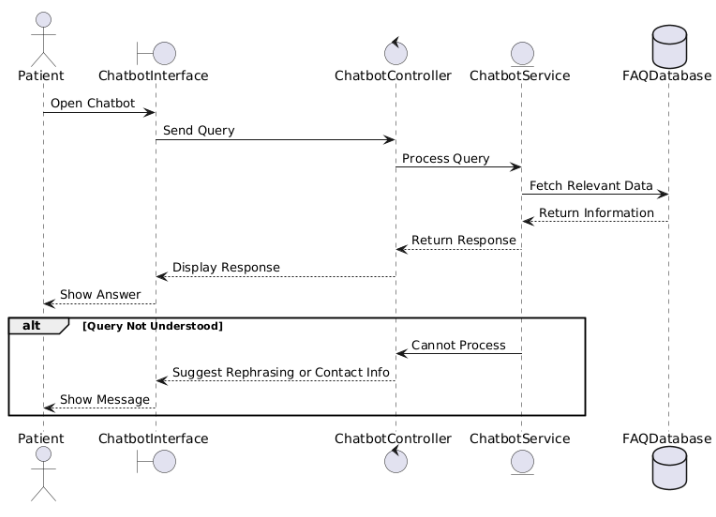


Figure 2.3.25: Sequence Diagram for < Chatbot Assistance >

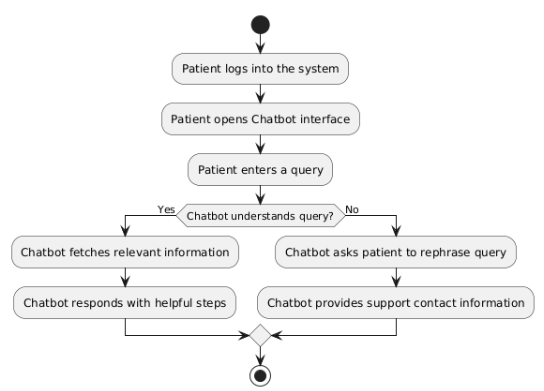
****

Figure 2.3.26: Activity Diagram for < Chatbot Assistance>

* 1. **Performance and Other Requirements**

This section describes the aspects of the HealthCare Storage System, which are not related to the content it will store or process, but rather how it will do it. Such requirement guarantee that the system is again reliable secure and meet the stakeholder expectations and also easy to use.

* + 1. **Software System Attributes**

- Usability

The system should be easily understandable and easily manageable by patients and admins without the need for training. The use of the website should not require a high level of IT literacy.

- Reliability

The system must be up 99.9% of the time to be available for patients and admins as needed. Data consistency must be maintained throughout the entire system and data synchronization must be precise.

* 1. **Design Constraints**

-Portability

The system has to be implemented on various web hosting services and work smoothly on any current web browsers.

Appendix B

**Software Design Document**

**(SDD)**

**1INTRODUCTION**

* 1. **Purpose**

The SDD is intended to give a detailed account of the design and structure of the HealthCare Storage System. The present document offers the precise instruction of how the software system will be designed, built and run to address the stated needs comprehensively. It helps the decisions made during the design procedure be coherent to functional and non-functional requirements as stated in SRS.

The SDD is advantageous to developers, software architects, system analysts and project managers since it provides the common ground of the system architecture including design patterns, data structures, algorithms and interfaces. This understanding creates harmony during development and also makes implementation, testing and maintenance of the system easier. Secondly, it also serves as a guide whenever there is future system development, changes, or modifications, in a bid to keep the design ahsame across the system.

Overall, the SDD is a critical document that supports software system deployment and comprehension, guaranteeing clarity, consistency, and collaboration among the development team and key stakeholders.

* 1. **Scope**

The HealthCare Storage System software product scopes include:

* User Registration and Authentication: Allowing patients to create accounts, create accounts for themselves and to verify themselves to be allowed into the system.
* Medical Report Management: Enabling patient to upload, download and manage their medical reports easily in their own personal safe box.
* Check-Up Reminder System: A feature that provides notification to the patient on when they should go for a medical check up next depending on the last report that was uploaded.
* Heart Rate Analysis Dashboard: Developing an application that would allow patients to upload their medical records and then see their heart rate trends in a dashboard.
* Feedback Submission: Enabling patients to give feedback concerning the system or to forward their medical reports for enhancement and monitoring.
* Commenting on Test Results: Allowing patients to include additional comments to the test results they upload for easier organization and monitoring.
* Admin Account Management: Enabling the administrators to open and close the patient accounts as when need be and also check on the system’s health.
* Reports and Feedback Dashboard for Admins: Enabling the system administrators to have a single interface to monitor the system reports, patient feedbacks and other operational issues.
  1. **Definitions, Acronyms and Abbreviation**

|  |  |  |
| --- | --- | --- |
| **Abbreviation** | **Acronym** | **Definition** |
| SDD | Software Design Document | a paper that gives a full description of the software system's design and architecture. |
| SRS | Software Requirements Specification | A document that describes the functional and non-functional requirements of a system. |
| UI | User Interface | the system's graphical or textual representation that allows users to engage with the program |
| DBMS | Database Management System | A component of software that aids in the management, storage, retrieval, and modification of data within a database. |
| JWT | JSON Web Token | A compact, URL-safe method for representing claims to be transferred between two parties securely. |
| CRUD | Create, Read, Update, Delete | The four fundamental activities required in maintaining persistent data in a system or database. |
| HTTP | Hypertext Transfer Protocol | Protocol used by the World Wide Web |
| AF | Alternative Flow | The use case scenario differs from the basic/normal flow. |
| EF | Exception Flow | Unintended course through the system, frequently due to a lack of information or system availability |
| API | Application Programming Interface | a system of rules and protocols that let various software programs to communicate and interact with one another. |
| AES | Advanced Encryption Standard | A secure encryption standard used for protecting sensitive data. |

* 1. **Overview**

The current edition of the SDD paper discusses the system architecture design, component detailed description, data design, and user interface design of the system.

* 1. **Reference**

Kaminski, P. (2007, October). Reforming software design documentation. *14th Working Conference on Reverse Engineering (WCRE 2007)*, 277–280. IEEE.

Osman, H., van Zadelhoff, A., Stikkolorum, D. R., & Chaudron, M. R. (2012, October). UML class diagram simplification: What is in the developer's mind? *Proceedings of the Second Edition of the International Workshop on Experiences and Empirical Studies in Software Modelling*, 1–6.

Li, L. (2000, September). Translating use cases to sequence diagrams. *ASE 2000: Proceedings of the Fifteenth IEEE International Conference on Automated Software Engineering*, 293–296. IEEE.

1. **SYSTEM ARCHITECTURAL DESIGN**

This section explains and demonstrates the chosen architecture to be implemented in this system. The chosen architecture is MVC which stands for Model-view-controller.

* 1. **Architecture Style and Rationale**

The Model-View-Controller (MVC) architectural design pattern is used for the Healthcare Storage System to avail modularity and separation of concerns. The Model contains the data and the business logic of the system and its tasks include access and processing of patient medical records, feedback reports and reminders. The View is in charge of the user interface since it presents data and records the users’ inputs, including uploading medical reports or viewing dashboards. The Controller works in between the Model and the View where all the data needed and for processing the user’s input, the appropriate task to be performed in the Model and relaying the results back to the View. By separating the system into such three parts, MVC improves these characteristics as maintainability, scalability, and testability to work and improve the part of UI, business rule, and user interaction without affecting another part. This approach is suitable for healthcare systems because it offers future expandability and keeps neatly structured architecture. The MVC architecture is shown in Figure 2.1

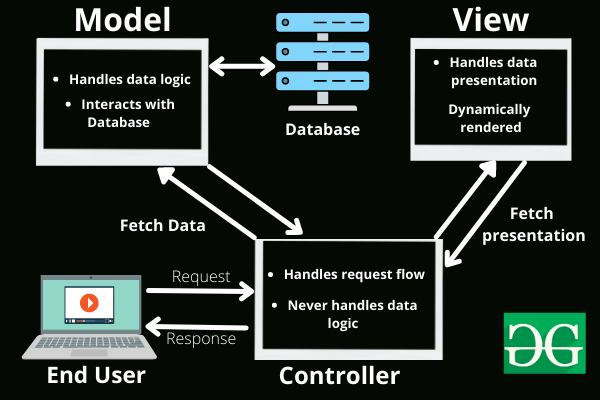


Figure 2.1 MVC Architecture

* 1. **Architecture Model**

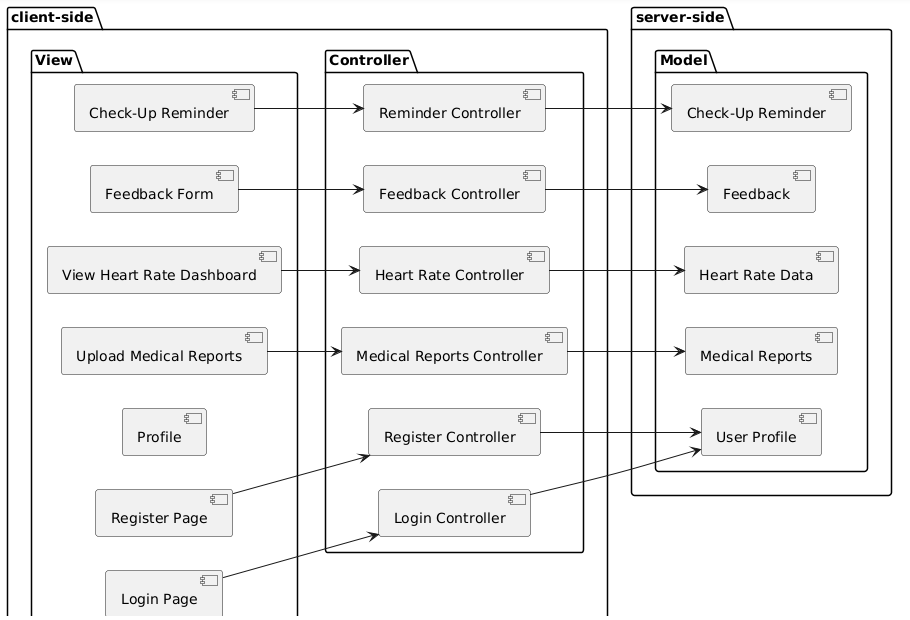
****

Figure 2.2: Component Model of HealthCare Storage System

* 1. **Use Case Diagram**

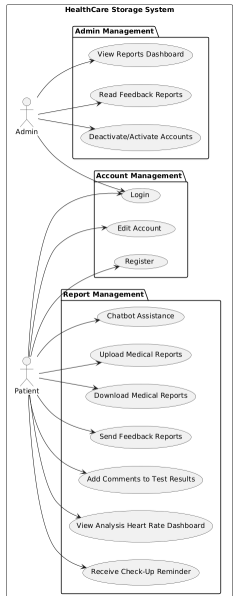


Figure 2.3: Use Case Diagram of HealthCare Storage System

1. **Data Design**
   1. **Data Description**

Database Design is a very important factor in the creation, deployment, and management of the HealthCare Storage System. It arranges all the required information that is necessary for storage, manipulation and retrieval of user and system information. This database design allows the implementation of other functions like users’ management, medical report processing, health evaluation, and feedback processing. The following figure 3.1 represents the database structure of the HealthCare Storage System where the tables like User Profile, Medical Reports, Heart Rate Data, Feedback, Check-Up Reminders have been shown. These tables are well integrated to offer a well structured and normalized database for the support of the system.

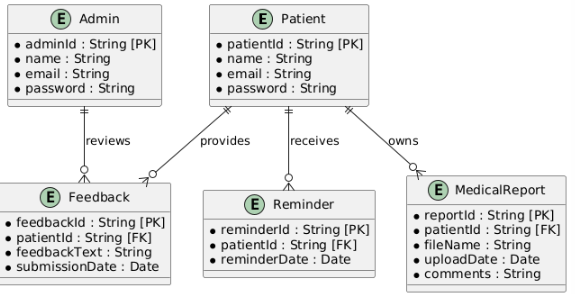


Figure 3.1 ERD HealthCare Storage System

* 1. **Data Dictionary**

This section will demonstrate all the attributes involved the database of the HealthCare Storage System.

* + 1. **Admin**

Table 3.1 Admin Table

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Data Type** | **Description** |
| Admin\_id | String (PK) | A unique identifier for each admin. |
| Name | String | Full name of the admin. |
| Email | String | Email address of the admin. |
| Password | String | Password to authenticate and secure the admin account |

* + 1. **Patient**

Table 3.2 Patient Table

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Data Type** | **Description** |
| Patient\_id | String (PK) | A unique identifier for each patient. |
| Name | String | Full name of the patient. |
| Email | String | Email address of the patient. |
| Password | String | Password to authenticate and secure the patient account. |

* + 1. **Feedback**

Table 3.3 Feedback Table

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Data Type** | **Description** |
| Feedback\_id | String (PK) | A unique identifier for each feedback. |
| Paitent\_id | String (FK) | A foreign key referencing the patient who provided feedback. |
| FeedbackText | String | Content of the feedback submitted by the patient. |
| SubmissionDate | Date | The date when feedback was submitted. |

* + 1. **Reminder**

Table 3.4 Reminder Table

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Data Type** | **Description** |
| Reminder\_id | String (PK) | A unique identifier for each reminder. |
| Patient\_id | String (FK) | A foreign key referencing the patient receiving the reminder. |
| ReminderDate | String | The date when the reminder is due. |

* + 1. **MedicalReport**

Table 3.5 MedicalReport Table

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Data Type** | **Description** |
| report\_id | String (PK) | A unique identifier for each medical report. |
| Patient\_id | String (FK) | A foreign key referencing the patient who owns the report. |
| FileName | String | Name of the medical report file. |
| UploadDate | Date | The date when the report was uploaded. |
| Comments | String | Any additional comments regarding the medical report. |

1. **User interface design**
   1. **Overview of user interface**

From the user’s point of view, the HealthCare Storage System offers a rich set of features to support patient’s experience in handling their health information. Patients can create an account, login, upload medical reports, view heart rate graphs, receive check-up reminders, download medical reports, read feedback reports and give feedback to the health care professionals.

The work conducted on the system’s user interface guarantees that it will be easy to use and easy to navigate by the patients. Formerly, privacy policies allowed users to divulge basic account information including name, email address, password. Once the patient logs in successfully, he/she can go to patient profile dashboard where he/she can update his/her information or engage with the different features.

Patients can upload their medical reports and the system checks the format and size of the report before accepting it. After that, the reports can be saved in the cloud for future use and are safe from unauthorized access. The heart rate analysis dashboard gives the patient graphical representation of their heart rate over time so that they do not feel out of touch with their health.

The system also includes a check up reminder that alerts the patient when a medical check up is due. They also provide proper health care at the right time to the patients hence increasing positive patient results. Also, patients can download their medical reports for offline use which can be easily shared with other doctors.

Patients have the freedom to comment on their experiences through a special section where they can write any comment they wish to give. The feedback reports are presented in a tabular format and patients can easily track responses or actions taken from their feedbacks.

In a broad sense HealthCare Storage System is friendly to the Users and interactive in such a way that it will enable the patients to access, store, analyze and correspond with the health care services in a very convenient manner.

* 1. **Screen Images**
     1. **Login Page**

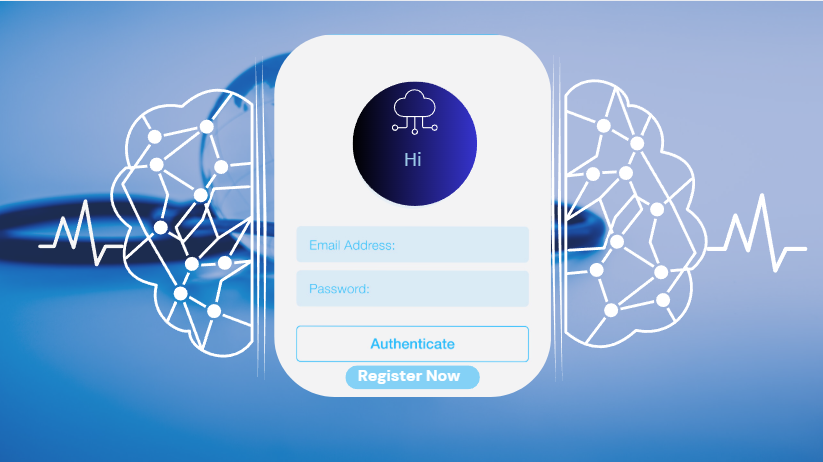
****

Figure 4.1 login Page

* + 1. **Register Page**

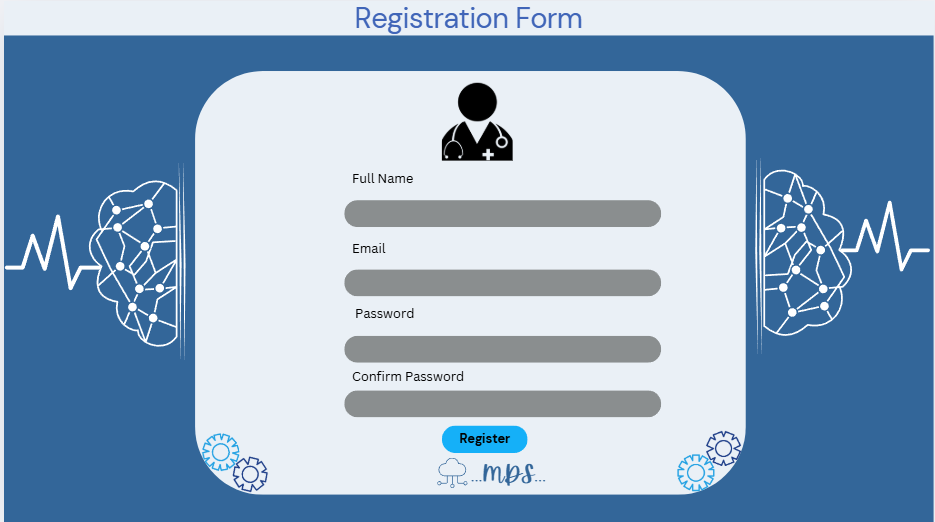
****

Figure 4.2 Register Page

* + 1. **Home page**

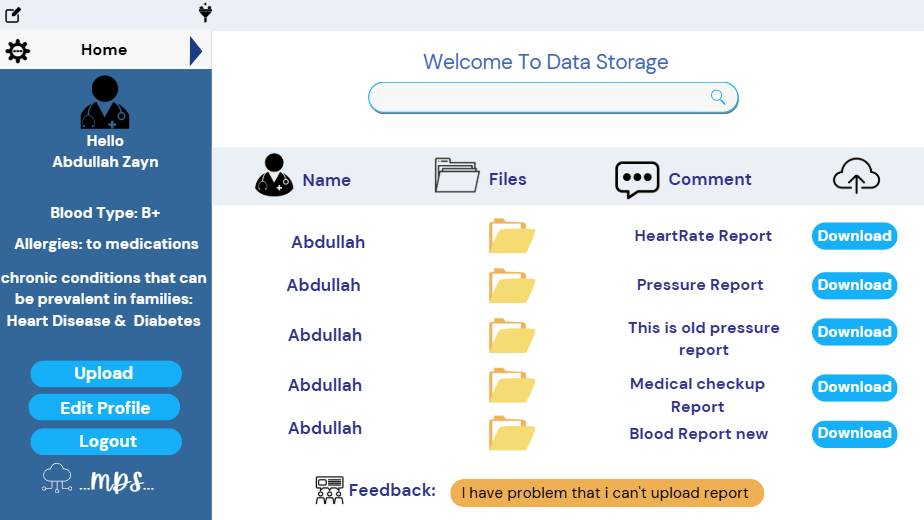
****

Figure 4.3 Home Page

* + 1. **Edit Profile**

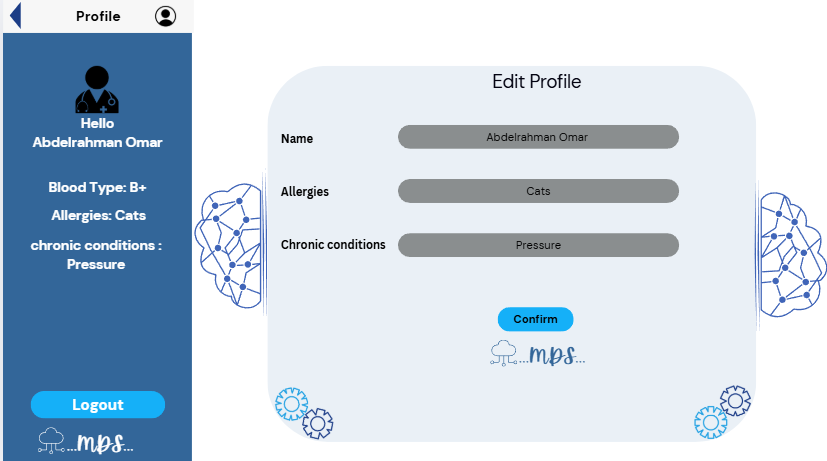
****

Figure 4.4 Edit Profile page

* + 1. **Admin Dashboard**

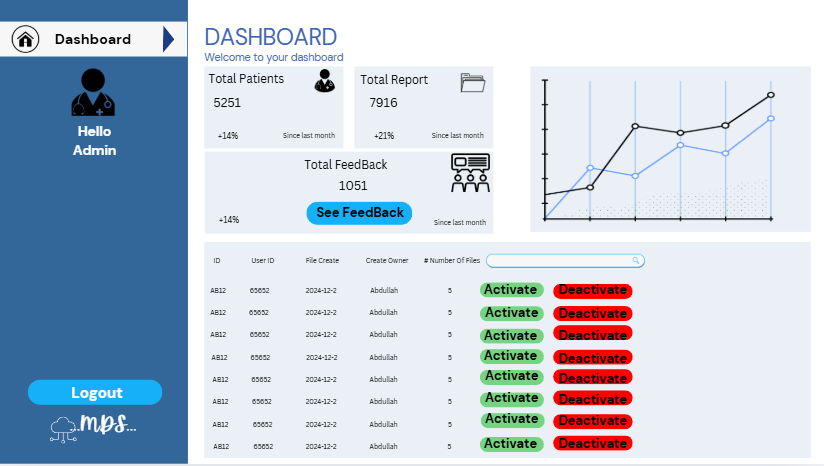
****

Figure 4.5 Admin Dashboard

* + 1. **Feedback Page**

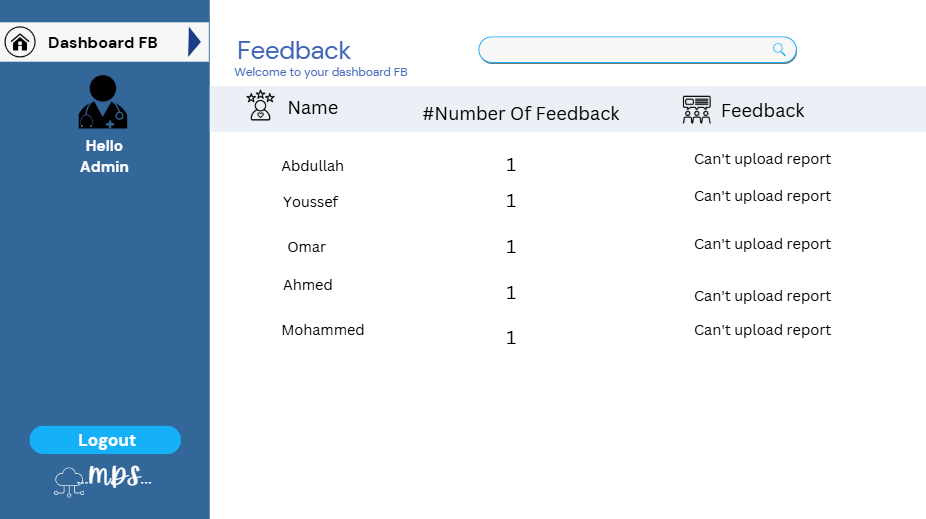
****

Figure 4.6 Feedback page

Appendix C

**Software Testing Document**

**(STD)**

1. **INTRODUCTION**
   1. **Purpose**

Currently, the Software Test Description (STD) can be considered as a complete roadmap for testing the functionality, performance, and reliability of the HealthCare Storage System. This document explains the general approach to testing, as well as the types of testing that are to be performed in order to achieve the stated and recognized purpose of the system. It shows system, environment for each test case with clear description of situations, events, actions, and outcome.

The primary objectives of this STD include:

* Identifying the test cases for measuring the capability and efficiency of the system.
* Offering a systematic way of how software testers may write and document their test findings.
* Serving as a source of information during the testing process to maintain coherence and totality of the testing process.
* Consulting on what to do if a problem or an inconsistency is to be found during the testing process.
  1. **Scope**

The HealthCare Storage System encompasses the following software product scopes:

* User Registration and Authentication: Allowing patients and administrators to create accounts, create and sign in to an account for the purpose of using different health care services.
* Medical Report Management: Enabling patients to view and download their medical records with strict adherence to patients’ data privacy and convenience.
* Feedback and Review System: Patients can make feedback and also comments on medical reports so it can boost patient participation.
* Check-Up Reminders: Helping patients through automatic notifications for the upcoming checkup to improve their healthcare operations.
* Health Data Analysis: Patients’ heart rate dashboard and analysis in a more presentable and easier to understand format to help them understand their health state.
* User Profile Management: Enabling patients to make changes to their personal details, account status and profiles at their convenience.
* Administrative Functions: Providing account administrators with means to work with patient accounts, track feedback reports, and maintain proper functioning of the system.
  1. **Overview**

The current version of the Software Test Description (STD) document gives a clear structure of the testing strategy of the HealthCare Storage System. It comes with the testing strategies, test cases and test scenarios that were used to ascertain the capability of the various features in the system. The document also includes the test coverage in order to cover all the important functionalities including the creation of users, medical reports, hearts rate dashboard and feedback submission. This STD helps the testing team to verify that the system meets all the functional and non-functional requirements improving reliability, usability and performance.

* 1. **Definitions, Acronyms and Abbreviation**

|  |  |  |
| --- | --- | --- |
| Abbreviation | Acronym | Definition |
| STD | Software Testing Description | a paper that outlines the testing strategy, tactics, and techniques. |
| QA | Quality Assurance | It is a process to ensure a product meets standards. |

* 1. **Reference**

Craig, R. D., & Jaskiel, S. P. (2002). Systematic Software Testing. Artech House.

1. **Test Cases, Data and Expected Result** 
   1. **Test TC001 for Registration**
      1. **UC001: Success Register**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case  ID | TC001\_01\_01 | Test Case Description | This test case will test functionality of register account when user put valid input are provided. |  |
| Created by: | Abdullah | Version: | 1 |  |
| No | Prerequisites |  | No. | Test Data |
| 1 | The user must go to register page. | 1 | [abdullahzayan@gmail.com](mailto:abdullahzayan@gmail.com) |
|  |  | 2 | Abdullah Zayan |
|  |  |  | 3 | A123# |
| Test Conditions |  | | | |
|  | | | | |
|  |
| Step # | Step Details | Expected Result |  | |  |
| 1 | Enter a valid email in the email field. | User can go to the next field. |  |
| 2 | Enter a valid full name in the name field. | User can go to the next field. |  |
| 3 | Enter a valid password in the password field. | User can go to the next field. |  |
| 4 | Enter the confirmation password in the confirmation password field. | User can go to the next field. |  |
| 5 | Click to Register button. | The system will display the success register. |  |

* + 1. **UC001: Failed Register**

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case  ID | TC001\_01\_02 | Test Case Description | This test case will test functionality of register account when user put invalid input or submitted with empty fields. |
| Created by: | Abdullah | Version: | 1 |  |
| No | Prerequisites |  | No. | Test Data |
| 1 | The user must go to register page. | 1 | asd@gmail.com |
|  |  | 2 | reqos |
|  |  |  | 3 | 1111 |
| Test Conditions |  | | | |
|  | | | | |
|  |
| Step # | Step Details | Expected Result |  | |  |
| 1 | Enter invalid email or leave the email field empty. | The system should display a error message for invalid input or empty field. |  |
| 2 | Enter invalid name or leave full name field empty. | The system should display a error message for invalid input or empty field. |  |
| 3 | Enter invalid password or leave the password field empty. | The system should display a error message for invalid input or empty field. |  |
| 4 | Enter invalid confirm password or leave the confirm password. field empty. | The system should display a error message for invalid input or empty field. |  |
| 5 | Click to Register button. | The account will not register in the system. |  |

* 1. **Test TC002 for login**
     1. **UC002 Success Login**

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case  ID | TC002\_01\_01 | Test Case Description | This test case will test functionality of login feature when valid inputs are provided. |
| Created by: | Abdullah | Version: | 1 |  |
| No | Prerequisites |  | No. | Test Data |
| 1 | The user already registers an account for the system. | 1 | [Abdullahzayan.my@gmail.com](mailto:Abdullahzayan.my@gmail.com)  A123# |
| 2 | The user goes to login page |  |  |
| Test Conditions |  | | | |
|  | | | | |
|  |
| Step # | Step Details | Expected Result |  | |  |
| 1 | Enter a valid email in email field. | The system will check for the email and it will be valid email. |  |
| 2 | Enter a valid password in password field. | The system will check for the password and it will be valid password. |  |
| 3 | Click to Login button. |  |  |

* + 1. **UC002 Failed Login**

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case  ID | TC002\_01\_02 | Test Case Description | This test case will test the functionality of the login feature when the email or password or both field is left empty. |
| Created by: | Abdullah | Version: | 1 |  |
| No | Prerequisites |  | No. | Test Data |
| 1 | The user goes to login page | 1 | …….. |
|  |  |  |  |
| Test Conditions |  | | | |
|  | | | | |
|  |
| Step # | Step Details | Expected Result |  | |  |
| 1 | Leave the email field empty. | The system displays an error message indicating that the email or password or both field is required. |  |
| 2 | leave the password empty. | The system displays an error message indicating that the email or password or both field is required. |  |
| 3 | Click on the login button. | The system displays an error message indicating that the email or password or both field is required. |  |

* + 1. **UC002 Failed Login**

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case  ID | TC002\_01\_03 | Test Case Description | This test case will test the functionality of the login feature when invalid input provided in email or password or both. |
| Created by: | Abdullah | Version: | 1 |  |
| No | Prerequisites |  | No. | Test Data |
| 1 | The user goes to login page | 1 | [As@gmail.com](mailto:As@gmail.com)  123 |
|  |  |  |  |
|  | | | | |
| Test Conditions |  | | | |
|  | | | | |
|  |
| Step # | Step Details | Expected Result |  | |  |
| 1 | Enter invalid email in the email field. | The system displays an error message indicating invalid login credentials. |  |
| 2 | Enter invalid password in the password field. | The system displays an error message indicating invalid login credentials. |  |
| 3 | Click on the login button. | The system displays an error message indicating invalid login credentials. |  |

* 1. **Test TC003 for Edit account**
     1. **UC003 Success Edit account**

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case  ID | TC003\_01\_01 | Test Case Description | This test case will test the functionality of the Edit account feature when valid input provided. |
| Created by: | Abdullah | Version: | 1 |  |
| No | Prerequisites |  | No. | Test Data |
| 1 | The user logs in to the system. | 1 | Omar mohammed  Dogs  diabetes |
| 2 | The user goes to edit account page. |  |  |
|  | | | | |
| Test Conditions |  | | | |
|  | | | | |
|  |
| Step # | Step Details | Expected Result |  | |  |
| 1 | Enter a valid name in the full name field. | The system will update the account information. |  |
| 2 | Enter a valid allergic in the allergic field. | The system will update the account information. |  |
| 3 | Enter a valid chronic condition in the chronic condition field. | The system will update the account information. |  |

* + 1. **UC003 Failed Edit account**

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case  ID | TC003\_01\_02 | Test Case Description | This test case will test the functionality of edit account when user leave fields empty. |
| Created by: | Abdullah | Version: |  |  |
| No | Prerequisites |  | No. | Test Data |
| 1 | The user logs in the system. | 1 | - |
| 2 | The user goes to edit account page. | 2 | - |
| Test Conditions |  | | | |
|  | | | | |
|  |
| Step # | Step Details | Expected Result |  | |  |
| 1 | Leave the name field empty. | The system display error in the field empty. |  |
| 2 | Leave the allergic field empty. | The system display error in the field empty. |  |
| 3 | Leave the chronic condition field empty. | The system display error in the field empty. |  |

* 1. **Test TC004 for Upload Medical Reports**
     1. **UC004 Success Upload Medical Reports**

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case  ID | TC004\_01\_01 | Test Case Description | This test case verifies the functionality of the Upload medical reports feature when valid file inputs are provided. |
| Created by: | Abdullah | Version: | 1 |  |
| No | Prerequisites |  | No. | Test Data |
| 1 | The user logs in to system | 1 | HeartRate.pdf(1MB) |
| 2 | The user goes to upload page (button). |  |  |
| Test Conditions |  | | | |
|  | | | | |
|  |
| Step # | Step Details | Expected Result |  | |  |
| 1 | Click on the Upload button. | The system validates the file format and size and report is successfully uploaded. |  |
| 2 | Select valid report file. | The system validates the file format and size and report is successfully uploaded. |  |
| 3 | Confirm the file upload. | The system validates the file format and size and report is successfully uploaded. |  |

* + 1. **UC004 Faild Upload Medical Reports**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case  ID | TC004\_01\_02 | Test Case Description | This test case verifies the functionality of the Upload medical reports feature when no file is uploaded or selected. |  |
| Created by: | Abdullah | Version: | 1 |  |
| No | Prerequisites |  | No. | Test Data |
| 1 | The user logs in to system. | 1 | - |
| 2 | The user goes to upload page (button). |  |  |
|  | | | | |
| Test Conditions |  | | | |
|  | | | | |
|  |
| Step # | Step Details | Expected Result |  | |  |
| 1 | Click on the Upload button. | The system display error for no file was selected. |  |
| 2 | Confirm upload without select file | The system display error for no file was selected. |  |

* + 1. **UC004 Faild Upload Medical Reports**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case  ID | TC004\_01\_03 | Test Case Description | This test case verifies the functionality of the upload medical reports feature when an invalid file format is provided. |  |
| Created by: | Abdullah | Version: | 1 |  |
| No | Prerequisites |  | No. | Test Data |
| 1 | The user logs in to system. | 1 | HeartRate.exe |
| 2 | The user goes to upload page (button). |  |  |
|  | | | | |
| Test Conditions |  | | | |
|  | | | | |
|  |
| Step # | Step Details | Expected Result |  | |  |
| 1 | Click on the Upload button. | The system display error for not support file format. |  |
| 2 | Select invalid file format | The system display error for not support file format. |  |
| 3 | Confirm the file upload | The system display error for not support file format. |  |

* + 1. **UC004 Faild Upload Medical Reports**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case  ID | TC004\_01\_04 | Test Case Description | This test case verifies the functionality of the upload medical reports feature when the file size exceeds the allowed limit. |  |
| Created by: | Abdullah | Version: | 1 |  |
| No | Prerequisites |  | No. | Test Data |
| 1 | The user logs in to system. | 1 | HeartRate.pdf(10GB) |
| 2 | The user goes to upload page (button). |  |  |
| Test Conditions |  | | | |
|  | | | | |
|  |
| Step # | Step Details | Expected Result |  | |  |
| 1 | Click on the Upload button. | The system display error for file size exceeds the limit. |  |
| 2 | Select file exceeding the size limit. | The system display error for file size exceeds the limit. |  |
| 3 | Confirm the file upload. | The system display error for file size exceeds the limit. |  |

* 1. **Test TC005 for Download Medical Reports**
     1. **UC005 Success Download Medical Reports**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case  ID | TC005\_01\_01 | Test Case Description | This test case verifies the functionality of the download medical reports feature when valid reports are available for download. |  |
| Created by: | Abdullah | Version: | 1 |  |
| No | Prerequisites |  | No. | Test Data |
| 1 | The user logs in to the system. | 1 | HeartRate.pdf |
| 2 | The user already uploaded reports. |  |  |
| 3 | The user goes to download page/button |  |  |  |
|  | | | | |
| Test Conditions |  | | | |
|  | | | | |
|  |
| Step # | Step Details | Expected Result |  | |  |
| 1 | Navigate to the medical report page. | The system retrieves the report from database and  the report will download successfully to the user device |  |
| 2 | Select the specific medical report from the list. | The system retrieves the report from database and  the report will download successfully to the user device |  |
| 3 | Click to download button. | The system retrieves the report from database and  the report will download successfully to the user device |  |

* + 1. **UC005 Faild Download Medical Reports**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case  ID | TC005\_01\_02 | Test Case Description | This test case verifies the functionality of the download medical reports feature when the report can’t download due network failure. |  |
| Created by: | Abdullah | Version: | 1 |  |
| No | Prerequisites |  | No. | Test Data |
| 1 | The user logs in to the system. | 1 | HeartRate.pdf |
| 2 | The user already uploaded reports. |  |  |
| 3 | The user goes to download page/button. |  |  |  |
|  | | | | |
| Test Conditions |  | | | |
|  | | | | |
|  |
| Step # | Step Details | Expected Result |  | |  |
| 1 | Navigate to the medical report page. | The system will display an error that download was interrupted. |  |
| 2 | Select the specific medical report from the list. | The system will display an error that download was interrupted. |  |
| 3 | Click to download button. | The system will display an error that download was interrupted. |  |

* 1. **Test TC006 for Send Feedback Reports**
     1. **UC006 Success Send Feedback Reports**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case  ID | TC006\_01\_01 | Test Case Description | This test case verifies the functionality of the send feedback reports feature with valid inputs. |  |
| Created by: | Abdullah | Version: | 1 |  |
| No | Prerequisites |  | No. | Test Data |
| 1 | The user logs in to the system. | 1 | I can’t upload my document. |
| 2 | The user goes to feedback page. |  |  |
|  | | | | |
| Test Conditions |  | | | |
|  | | | | |
|  |
| Step # | Step Details | Expected Result |  | |  |
| 1 | Enter a valid feedback text in the feedback field. | The system will validate the feedback content and  the feedback will save in the database. |  |
| 2 | Click to submit button. | The system will validate the feedback content and  the feedback will save in the database. |  |

* + 1. **UC006 Faild Send Feedback Reports**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case  ID | | TC006\_01\_02 | Test Case Description | This test case verifies the functionality of the send feedback reports feature when the feedback field is empty. |  |
| Created by: | | Abdullah | Version: | 1 |  |
| No | | Prerequisites |  | No. | Test Data |
| 1 | | The user logs in to the system. |  |  |
| 2 | | The user goes to feedback page. |  |  |
|  | | | | | |
| Test Conditions | |  | | | |
|  | | | | | |
|  |
| Step # | | Step Details | Expected Result |  | |  |
| 1 | | Leave the feedback field empty. | The system will display error that the field is empty. |  |
| 2 | | Click to submit button. | The system will display error that the field is empty. |  |
|  |

* 1. **Test TC007 for Add Comments to Test Result**
     1. **UC007 Success Add Comments to Test Result**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case  ID | TC007\_01\_01 | Test Case Description | This test case verifies the functionality of the Add comments to test result feature with valid inputs. |  |
| Created by: | Abdullah | Version: | 1 |  |
| No | Prerequisites |  | No. | Test Data |
| 1 | The user logs in the system. | 1 | HeartRate.pdf |
| 2 | The user goes to the specific test file. | 2 | This file is my test result on 2020. |
| 3 | The selected test is available for comment. |  |  |  |
|  | | | | |
| Test Conditions |  | | | |
|  | | | | |
|  |
| Step # | Step Details | Expected Result |  | |  |
| 1 | User select the test result file. | The system will validate the comment and  the comment is saved in database and will link to specific test result file. |  |
| 2 | Enter a comment in a comment field. | The system will validate the comment and  the comment is saved in database and will link to specific test result file. |  |
| 3 | Click confirm button. | The system will validate the comment and  the comment is saved in database and will link to specific test result file. |  |

* + 1. **UC007 Faild Add Comments to Test Result**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case  ID | TC007\_01\_02 | Test Case Description | This test case verifies the functionality of the add comments to test result feature when the comment field is empty. |  |
| Created by: | Abdullah | Version: | 1 |  |
| No | Prerequisites |  | No. | Test Data |
| 1 | User select the test result file | 1 | HeartRate.pdf |
| 2 | Leave the comment field empty |  |  |
| 3 | The selected test is available for comment |  |  |  |
|  | | | | |
| Test Conditions |  | | | |
|  | | | | |
|  |
| Step # | Step Details | Expected Result |  | |  |
| 1 | User select the test result file | The system displays an error for comment field cannot be empty if you want to click to confirm button |  |
| 2 | Leave the comment field empty | The system displays an error for comment field cannot be empty if you want to click to confirm button |  |
| 3 | Click confirm button | The system displays an error for comment field cannot be empty if you want to click to confirm button |  |

* 1. **Test TC008 for View Analysis Heart Rate Dashboard**
     1. **UC008 Success View Analysis Heart Rate Dashboard**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case  ID | TC008\_01\_01 | Test Case Description | This test case verifies the functionality of the view analysis heart rate dashboard feature with valid data available. | |  |
| Created by: | Abdullah | Version: | 1 | |  |
| No | Prerequisites |  | No. | Test Data | |
| 1 | The user logs in to the system | 1 | User1 | |
| 2 | The user already uploads the heart rate report file |  |  | |
|  | | | | | |
| Test Conditions |  | | | | |
|  | | | | | |
|  |
| Step # | Step Details | Expected Result |  | | |  |
| 1 | User go to Heart rate dashboard page | The system displays the heart rate dashboard successfully |  |
| 2 | The page will verify the loads successfully | The system displays the heart rate dashboard successfully |  |
| 3 | View the graphical representation of heart rate based on the user heart rate medical reports | The system displays the heart rate dashboard successfully |  |

* + 1. **UC008 Faild View Analysis Heart Rate Dashboard**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case  ID | TC008\_01\_02 | Test Case Description | This test case verifies the functionality of the view analysis heart rate dashboard feature when no heart rate data is available. | |  |
| Created by: | Abdullah | Version: | 1 | |  |
| No | Prerequisites |  | No. | Test Data | |
| 1 | The user logs in to the system | 1 | User 1 | |
|  |  |  |  | |
|  | | | | | |
| Test Conditions |  | | | | |
|  | | | | | |
|  |
| Step # | Step Details | Expected Result |  | | |  |
| 1 | User go to Heart rate dashboard page | The system display message that no heart rate data is available |  |
| 2 | Dashboard loads successfully | The system display message that no heart rate data is available |  |

* 1. **Test TC009 for Receive Check-up Reminder**
     1. **UC009 Success Receive Check-up Reminder**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case  ID | TC009\_01\_01 | Test Case Description | This test case verifies the functionality of the receive check-up reminder feature when the reminder is available. | |  |
| Created by: | Abdullah | Version: | 1 | |  |
| No | Prerequisites |  | No. | Test Data | |
| 1 | The user already registers in to the system | 1 | Email: abdullahzayan.my  @gmail.com | |
| 2 | The user already uploads a check-up medical report |  |  | |
| 3 | A check-up reminder is scheduled for user in the system |  |  |  | |
|  | | | | | |
| Test Conditions |  | | | | |
|  | | | | | |
|  |
| Step # | Step Details | Expected Result |  | | |  |
| 1 | User go to notification icon or his/her email that register in the system | The system will display or sent the check-up reminder to user |  |
| 2 | View the notification for check-up reminders | The system will display or sent the check-up reminder to user |  |

* + 1. **UC009 Faild Receive Check-up Reminder**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case  ID | TC009\_01\_02 | Test Case Description | This test case verifies the functionality of the receive check-up reminder feature when no reminders are available. | |  |
| Created by: | Abdullah | Version: | 1 | |  |
| No | Prerequisites |  | No. | Test Data | |
| 1 | The user already registers in to the system | 1 | Email: abdullahzayan.my  @gmail.com | |
|  |  |  |  | |
|  | | | | | |
| Test Conditions |  | | | | |
|  | | | | | |
|  |
| Step # | Step Details | Expected Result |  | | |  |
| 1 | User go to notification icon or his/her email that register in the system | The system will not display any check-up reminder are scheduled |  |
| 2 | View the notification for check-up reminders | The system will not display any check-up reminder are scheduled |  |

* 1. **Test TC010 for View Report Dashboard**
     1. **UC010 Success View Report Dashboard**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case  ID | TC0010\_01\_01 | Test Case Description | This test case verifies the functionality of the view reports dashboard feature when reports are available. |  |
| Created by: | Abdullah | Version: | 1 |  |
|  | | | | |
| No | Prerequisites |  | No. | Test Data |
| 1 | The admin is logging the system |  |  |
| 2 | Reports data is already available in the system |  |  |
|  | | | | |
| Test Conditions |  | | | |
|  | | | | |
|  |
| Step # | Step Details | Expected Result |  | |  |
| 1 | Admin go to admin dashboard page | The system displays a admin dashboard with all available reports |  |
| 2 | Verify that dashboard loads successfully | The system displays a admin dashboard with all available reports |  |
| 3 | View graphical analysis of report summary | The system displays a admin dashboard with all available reports |  |

* + 1. **UC010 Faild View Report Dashboard**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case  ID | TC0010\_01\_02 | Test Case Description | This test case verifies the functionality of the view reports dashboard feature when no reports available. | |  |
| Created by: | Abdullah | Version: | 1 | |  |
| No | Prerequisites |  | No. | Test Data | |
| 1 | The admin is logging the system |  |  | |
| 2 | No report data is available in the system |  |  | |
|  | | | | | |
| Test Conditions |  | | | | |
|  | | | | | |
|  |
| Step # | Step Details | Expected Result |  | | |  |
| 1 | Admin go to admin dashboard page | The system displays a admin dashboard with no reports available |  |
| 2 | Verify that dashboard loads successfully | The system displays a admin dashboard with no reports available |  |
| 3 | View graphical analysis of report summary | The system displays a admin dashboard with no reports available |  |

* 1. **Test TC011 for Read Feedback Reports**
     1. **UC011 Success Read Feedback Reports**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case  ID | TC0011\_01\_01 | Test Case Description | This test case verifies the functionality of the read feedback reports feature when feedback reports are available. | |  |
| Created by: | Abdullah | Version: | 1 | |  |
| No | Prerequisites |  | No. | Test Data | |
| 1 | The admin is logging the system | 1 | Feedback: the system is very slow when I upload the file | |
| 2 | Users already send feedback in the system |  |  | |
| 3 | Feedback already exists in the system |  |  |  | |
|  | | | | | |
| Test Conditions |  | | | | |
|  | | | | | |
|  |
| Step # | Step Details | Expected Result |  | | |  |
| 1 | Admin go to feedback page | The system will display the list of available feedback |  |
| 2 | Admin can select or view feedback list | The system will display the list of available feedback |  |

* + 1. **UC011 Faild Read Feedback Reports**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case  ID | TC0011\_01\_02 | Test Case Description | This test case verifies the functionality of the read feedback reports feature when no feedback is available. |  |
| Created by: | Abdullah | Version: | 1 |  |
| No | Prerequisites |  | No. | Test Data |
| 1 | The admin is logging the system |  |  |
| 2 | User don’t put any feedback in feedback field |  |  |
| 3 | No feedback exists in the system |  |  |  |
|  | | | | |
| Test Conditions |  | | | |
|  | | | | |
|  |
| Step # | Step Details | Expected Result |  | |  |
| 1 | Admin go to feedback page | The system will display the feedback list with no feedback available |  |
| 2 | Admin can view the feedback list but no feedback available | The system will display the feedback list with no feedback available |  |

* 1. **Test TC012 for Deactivate/Activate Accounts**
     1. **UC012 Success Deactivate/Activate Accounts**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case  ID | TC0012\_01\_01 | Test Case Description | This test case verifies the functionality of the deactivate or activate account feature when a valid account is selected. | |  |
| Created by: | Abdullah | Version: | 1 | |  |
| No | Prerequisites |  | No. | Test Data | |
| 1 | The admin logs in to the system | 1 | Admin1 | |
| 2 | The user already registers in the system | 2 | User1 | |
|  | | | | | |
| Test Conditions |  | | | | |
|  | | | | | |
|  |
| Step # | Step Details | Expected Result |  | | |  |
| 1 | Admin go to user account table | The system will update the user account status to deactivated or activated |  |
| 2 | Select a specific user account | The system will update the user account status to deactivated or activated |  |
| 3 | Admin can click to Activate button or deactivate button | The system will update the user account status to deactivated or activated |  |

* + 1. **UC012 Faild Deactivate/Activate Accounts**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case  ID | TC0012\_01\_02 | Test Case Description | This test case verifies the functionality of the deactivate or activate account feature when no account display for admin |  |
| Created by: | Abdullah | Version: | 1 |  |
| No | Prerequisites |  | No. | Test Data |
| 1 | The admin logs in to the system | 1 | Admin1 |
|  |  |  |  |
|  | | | | |
| Test Conditions |  | | | |
|  | | | | |
|  |
| Step # | Step Details | Expected Result |  | |  |
| 1 | Admin go to user account table | The system will show that there is no account register to system so admin can activate or deactivate the account |  |
| 2 | Admin view the user account table | The system will show that there is no account register to system so admin can activate or deactivate the account |  |

Appendix D

**User Interview**

List of questions that have been asked during the interview with the users:

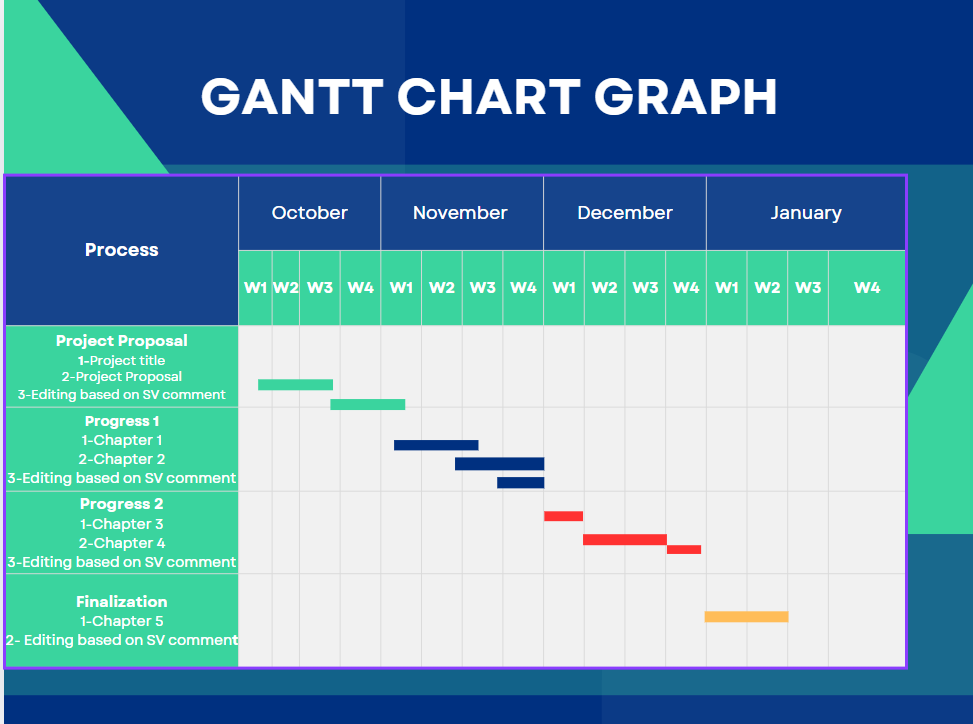
1. In what manner do you deal with your medical records?
2. Have you ever misplaced or sometimes you find it hard to access your medical records?
3. If you would be having reminders every time you need to go for a routine health check-up how would you feel?
4. What other features would you like to be included in a medical record storage system?
5. In your opinion, would it help you to visualize health trends, for example, by analysing the rhythm of your heartbeat?
6. How crucial is it for you to have protected access to your health information?
7. What issues do you have with the current health care systems if any?



Appendix E

**Gantt Chart Diagram**

**PSM1 Gantt Chart:**



**PSM2 Gantt Chart:**

