

# **Unified Medical System (UMS) for India with Early Disease Outbreak Detection**

*Main Project Report*

*Submitted by*

**Devadethan R**

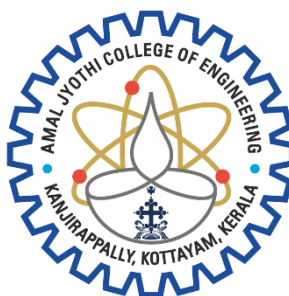
**Reg. No.: AJC20MCA-I029**

*In Partial fulfillment for the Award of the Degree of*

**INTEGRATED MASTER OF COMPUTER APPLICATIONS**

**(INMCA)**

**APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY**



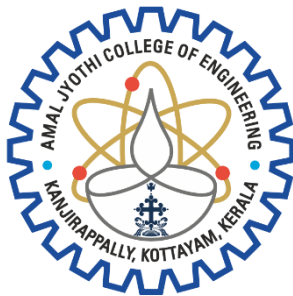
**AMAL JYOTHI COLLEGE OF ENGINEERING AUTONOMOUS**

**KANJIRAPPALLY**

[Affiliated to APJ Abdul Kalam Technological University, Kerala. Approved by AICTE,  
Accredited by NAAC. Koovappally, Kanjirappally, Kottayam, Kerala – 686518]

**2020-2025**

**DEPARTMENT OF COMPUTER APPLICATIONS**  
**AMAL JYOTHI COLLEGE OF ENGINEERING AUTONOMOUS**  
**KANJIRAPPALLY**



**CERTIFICATE**

This is to certify that the Project report titled “**UNIFIED MEDICAL SYSTEM (UMS)**” is the bona fide work of **DEVADETHAN R (Regno: AJC20MCA-I029)** carried out in partial fulfillment of the requirements for the award of the **Degree of Integrated Master of Computer Applications** at **Amal Jyothi College of Engineering Autonomous, Kanjirappally**, Affiliated to **APJ Abdul Kalam Technological University**. The project was undertaken during the period from **January 01, 2025** to **April 23, 2025**.

**Ms. Jetty Benjamin**  
Internal Guide

**Ms. Meera Rose Mathew**  
Coordinator

**Rev. Fr. Dr. Rubin Thottupurathu Jose**  
Head of the Department

**External Examiner**



**AMAL JYOTHI**  
**COLLEGE OF ENGINEERING**  
**AUTONOMOUS**  
KANJIRAPPALLY



**Department of Computer Applications**  
**CERTIFICATE ON PLAGIARISM CHECK**

1	Name of the Scholar	Devadethan R
2	Title of the Publication	Unified Medical System (UMS)
3	Name of the Guide	Ms. Jetty Benjamin
4	Similar Content (%) identified	25%
5	Acceptable Maximum Limit	25%
6	Software Used	TURNITIN
8	No. of pages verified	

\*Report on plagiarism check result with % of similarity shall be attached.

Devadethan R

Name & Signature of the Scholar:

Mr. Amal K Jose  Signed by: AMAL K. JOSE  
Reason: Plagiarism Check  
Location: Koozappally,  
Kanjirappally  
Date: 11-Apr-2025 (02:42 PM)

Name & Signature of the Guide:

Name & Signature of the Head of the Department:

Dept. Seal

## **DECLARATION**

I hereby declare that the project report “**UNIFIED MEDICAL SYSTEM (UMS)**” is a bona fide work done at **Amal Jyothi College of Engineering Autonomous, Kanjirappally**, Affiliated to **APJ Abdul Kalam Technological University**, towards the partial fulfilment of the requirements for the award of the **Integrated Master of Computer Applications (MCA)** during the period from **January 01, 2025 to April 23, 2025**.

**Date:**

**DEVADETHAN R**

**KANJIRAPPALLY**

**Reg: AJC20MCA-I029**

## ACKNOWLEDGEMENT

First and foremost, I thank God almighty for his eternal love and protection throughout the project. I take this opportunity to express my gratitude to all who helped me in completing this project successfully. It has been said that gratitude is the memory of the heart. I wish to express my sincere gratitude to our Director (Administration) **Rev. Fr. Dr. Roy Abraham Pazhayaparampil** and Principal **Dr. Lillykutty Jacob** for providing good faculty for guidance.

I owe a great depth of gratitude towards our Head of the Department **Rev.Fr.Dr. Rubin Thottupurathu Jose** for helping us. I extend my whole hearted thanks to the project coordinator **Ms. Meera Rose Mathew (Assistant Professor)** for her valuable suggestions and for overwhelming concern and guidance from the beginning to the end of the project. I would also express sincere gratitude to my guide **Ms. Jetty Benjamin (Assistant Professor)** for his inspiration and helping hand.

I thank our beloved teachers for their cooperation and suggestions that helped me throughout the project. I express my thanks to all my friends and classmates for their interest, dedication, and encouragement shown towards the project. I convey my hearty thanks to my family for the moral support, suggestions, and encouragement to make this venture a success.

DEVADETHAN R

# ABSTRACT

The Unified Medical System (UMS) for India is a comprehensive platform for digital health care developed for the unification of patient services, coordinating healthcare providers and monitoring public health within a single integrated ecosystem. It deals with the main challenges in the management of healthcare and the outbreak of diseases by using modern technologies to create an affordable patient -focused solution.

UMS allows patients to register, manage their profiles, books, access to medical records such as recipes and laboratory results, symptoms of protocol for disease supervision, downloading medical certificates and engaging in chatbot driven by artist for health instructions and support. Healthcare providers, including doctors, can manage meetings, safely approach patients with consent and exchange medical records effectively.

The hospital benefits from effective coordination of appointment and access to anonymized data for operational improvements. Public health officials, epidemiologists and authorized scientists have access to the outbreaks and relevant data for planning research and intervention. A key part of the UMS is the detection of the outbreak of the disease that uses machine learning to analyze the symptoms data in real time from different regions, allowing early warnings for potential outbreaks. This ability enables public health authorities to take proactive steps, increasing the resistance of the community's health.

The platform technology storage includes the Bootstrap and Jinja templates for the frontnd, the Backend and API, and the Mongoddb flask for secure and scalable data storage.

The UMS is designed to adhere to standards such as the Unified Health Interface (UHI) in the Ayushman Bharat digital mission and HL7 FHIR for Interoperable and secure health care data. By combining real-time analysts, supported by AI, and the design of privacy, UMS provides a transformation approach to health care-it supports early detection of diseases, trouble-free interaction of the provider, and informed the patient's involvement across the Indian population.

# CONTENT

SL. NO	TOPIC	PAGE NO
<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
<b>1.1</b>	<b>PROJECT OVERVIEW</b>	<b>2</b>
<b>1.2</b>	<b>PROJECT SPECIFICATION</b>	<b>2</b>
<b>2</b>	<b>SYSTEM STUDY</b>	<b>3</b>
<b>2.1</b>	<b>INTRODUCTION</b>	<b>4</b>
<b>2.2</b>	<b>EXISTING SYSTEM</b>	<b>4</b>
<b>2.2.1</b>	<b>DRAWBACKS OF EXISTING SYSTEM</b>	<b>5</b>
<b>2.3</b>	<b>PROPOSED SYSTEM</b>	<b>6</b>
<b>2.3.1</b>	<b>ADVANTAGES OF PROPOSED SYSTEM</b>	<b>6</b>
<b>3</b>	<b>REQUIREMENT ANALYSIS</b>	<b>7</b>
<b>3.1</b>	<b>FEASIBILITY STUDY</b>	<b>8</b>
<b>3.1.1</b>	<b>ECONOMICAL FEASIBILITY</b>	<b>8</b>
<b>3.1.2</b>	<b>TECHNICAL FEASIBILITY</b>	<b>8</b>
<b>3.1.3</b>	<b>BEHAVIORAL FEASIBILITY</b>	<b>8</b>
<b>3.1.4</b>	<b>FEASIBILITY STUDY QUESTIONNAIRE</b>	<b>9</b>
<b>3.1.5</b>	<b>GEOTAGGED PHOTOGRAPH</b>	<b>11</b>
<b>3.2</b>	<b>SYSTEM SPECIFICATION</b>	<b>11</b>
<b>3.2.1</b>	<b>HARDWARE SPECIFICATION</b>	<b>11</b>
<b>3.2.2</b>	<b>SOFTWARE SPECIFICATION</b>	<b>11</b>
<b>3.3</b>	<b>SOFTWARE DESCRIPTION</b>	<b>12</b>
<b>3.3.1</b>	<b>FLASK</b>	<b>12</b>
<b>3.3.2</b>	<b>MONGODB</b>	<b>13</b>
<b>4</b>	<b>SYSTEM DESIGN</b>	<b>14</b>
<b>4.1</b>	<b>INTRODUCTION</b>	<b>15</b>
<b>4.2</b>	<b>UML DIAGRAM</b>	<b>15</b>
<b>4.2.1</b>	<b>USE CASE DIAGRAM</b>	<b>16</b>
<b>4.2.2</b>	<b>SEQUENCE DIAGRAM</b>	<b>17</b>
<b>4.2.3</b>	<b>STATE CHART DIAGRAM</b>	<b>18</b>
<b>4.2.4</b>	<b>ACTIVITY DIAGRAM</b>	<b>19</b>
<b>4.2.5</b>	<b>CLASS DIAGRAM</b>	<b>21</b>
<b>4.2.6</b>	<b>OBJECT DIAGRAM</b>	<b>22</b>

4.2.7	COMPONENT DIAGRAM	23
4.2.8	DEPLOYMENT DIAGRAM	24
4.3	USER INTERFACE DESIGN USING FIGMA	25
4.4	DATABASE DESIGN	29
5	SYSTEM TESTING	37
5.1	INTRODUCTION	38
5.2	TEST PLAN	38
5.2.1	UNIT TESTING	39
5.2.2	INTEGRATION TESTING	39
5.2.3	VALIDATION TESTING	39
5.2.4	USER ACCEPTANCE TESTING	39
5.2.5	AUTOMATION TESTING	40
5.2.6	SELENIUM TESTING	40
6	IMPLEMENTATION	62
6.1	INTRODUCTION	63
6.2	IMPLEMENTATION PROCEDURE	63
6.2.1	USER TRAINING	64
6.2.2	TRAINING ON APPLICATION SOFTWARE	64
6.2.3	SYSTEM MAINTENANCE	64
6.2.4	HOSTING	64
7	CONCLUSION & FUTURE SCOPE	67
7.1	CONCLUSION	68
7.2	FUTURE SCOPE	68
8	BIBLIOGRAPHY	69
9	APPENDIX	71
9.1	SAMPLE CODE	72
9.2	SCREEN SHOTS	76
9.3	GIT LOG	79
9.4	CERTIFICATES	82
9.5	PLAGIARISM REPORT	85



## **List of Abbreviations**

- **UMS** - Unified Medical System
- **ABDM** - Ayushman Bharat Digital Mission
- **AI** - Artificial Intelligence
- **API** - Application Programming Interface
- **EMR** - Electronic Medical Records
- **EHR** - Electronic Health Records
- **HL7** - Health Level 7 (Interoperability Standard for Health Information)
- **FHIR** - Fast Healthcare Interoperability Resources
- **GDPR** - General Data Protection Regulation
- **PHI** - Protected Health Information
- **ROI** - Return on Investment

# **CHAPTER 1**

## **INTRODUCTION**

## 1.1 PROJECT OVERVIEW

The Unified Medical System (UMS) for India is a platform for digital health care that seeks to streamline access to medical services, seize patients and strengthen the supervision of public health. The UMS, which is designed to promote effective provision of health care in India, integrates basic functions such as meeting reservation, treatment of medical records and chatbot to support patients, and creates a cohesive system for patients, healthcare providers and public health organs. The platform uses machine learning to detect potential outbreaks disease by analyzing data symptoms in real time from patients across regions, allowing early intervention and control measures. The aim of this unified approach is to strengthen the accessibility of health care, education of patients and monitoring of data based on data.

## 1.2 PROJECT SPECIFICATION

The UMS platform is based on a robust technological magazine that includes bootstrap for web applications, flutter for mobile applications, a bag for backend services and MongoDB for secure data storage. The key modules include:

- Patient portal: Patients may register, book meetings, access to medical records, symptoms of protocol for disease supervision and receive instructions from Chatbot driven AI.
- Healthcare Provider Portal: Allows providers to safely access patient records (with consent), manage meetings and share data within the patient effective care system.
- United health data platform: standardizes and safely stores medical data and supports effective data sharing among users with strict personal data protection protocols.
- Module Detection of the outbreak of the disease: uses machine learning to analyze the patient's symptoms in real time, allowing the system to detect and warn the health authorities for potential outbreaks of the disease soon.
- Patient support Chatbot: helps patients by answering frequently asked medical issues, leading them through the system and providing information about self-care.

Unique features: UMS platform excels in the ability to detect disease outbreak, trouble-free exchange of health data between health care providers and patient design, which allows individuals to manage their health information. Chatbot driven AI further increases the patient's accessibility by providing instructions and support within the platform and contributes to comprehensive experience in health care in India.

## **CHAPTER 2**

### **SYSTEM STUDY**

## 2.1 INTRODUCTION

The health environment in India faces numerous challenges, including fragmented services, insufficient access to medical information and urgent needs of early detection of the outbreak of the disease. In response to these challenges, the Unified Medical System (UMS) appears as an innovative solution aimed at the revolution of the availability of health care and strengthening public health results throughout the country.

This digital health platform integrates basic services such as meeting planning, secured medical records and chatbot for patient support, creating a cohesive ecosystem for patients, healthcare providers and public health providers. UMS uses state-of-the-art technology, including machine learning algorithms, to analyze the symptoms reported by patient in real time, allowing early identification of potential outbreaks of the disease.

This proactive approach not only facilitates early interventions, but also strengthens the overall public health infrastructure. By centralizing health data and support of smooth communication between UMS users, patients allow patients to take control of their health information, while ensuring that health care providers can provide coordinated care. The UMS, based on a scalable technological magazine, is designed to be user-friendly and in accordance with the protection of personal data protection, which ensures that sensitive health information is safely managed. The aim of the unified medical system supports the environmental environment and increases the ability to supervise the disease is to transform the provision of health care in India and eventually contributed to improved health results and a more resistant health care system.

## 2.2 EXISTING SYSTEM

The current ecosystem of India's healthcare is largely fragmented, with patients' records, appointment systems and diagnostic data often maintained in isolated institution-specific systems. Most hospitals and clinics work on their own digital or manual infrastructure, lack standardized data formats and unified access mechanisms. Telemedicine services exist separately, but usually do not integrate well in local hospitals or government health monitoring systems. As a result, the continuity of care is prevented and patients often have to repeat diagnostic tests or carry physical documents when switching providers. Public health monitoring is delayed due to the absence of real-time monitoring mechanisms, making it more difficult to detect outbreaks and responses based on data.

## NATURAL SYSTEM STUDIED

The observed natural system is the conventional process of providing health care in rural and urban India, where patients interact directly with medical experts through personal consultations with minimal or without digital tools. Patients usually carry hand -written recipes and the results of physical tests. In rural areas, addiction to government hospitals and primary health centers is high, with limited digital infrastructure.

## DESIGNED SYSTEM STUDIED

Several digital health initiatives such as Ayushman Bharat (ABDM) and private hospital management systems have been studied to understand the advantages and reduction of existing digital health care models. The united health interface (UHI) within the ABDM aims to standardize digital health records and ensure interoperability across platforms. Similarly, information systems for health management (HMIS) used in urban hospitals provide reservation of meetings and functions of EMR, but are not accessible across various institutions or geographies. On global platforms, such as Babylon Health and ADA, offering automated medical advice based on the inputs of symptoms driven by AI powered by health chatbots and symptoms.

### 2.2.1 DRAWBACKS OF EXISTING SYSTEM

- **Lack of integration:** Patient and healthcare records are scattered on various platforms or are kept manually, making it difficult to replace data and continuity of care.
- **Limited patient control:** Patients have little or no control over their health records and often need to physically carry recipes, laboratory messages and previous diagnoses.
- **No real -time supervision:** Existing systems do not support the logging of symptoms in real time or early detection of outbreaks, delaying public health reactions.
- **Poor availability:** Digital health platforms often lack language support, mobile sensitivity and user -friendly interfaces, limiting their adoption in rural and insufficient areas.

### 2.3 PROPOSED SYSTEM

The proposed united medical system (UMS) is designed to engage in fragmentation and inefficiency in the current Indian health system by providing a centralized digital platform focused on the patient. UMS allows patients to manage their health profiles, book a meeting and safely access to their medical records, allowing them to make informed health decisions. Healthcare providers can safely access and share patient information (with consent), ensure the continuity of care and support

the treatment of cooperation. The key innovation is the integration of machine learning to analyze the patient's symptoms in real time across regions, allowing early detection of the focus of the disease and early warning of public health authorities. Chatbot driven AI offers patients with basic health management, directs them to relevant services and helps with meeting planning, increasing users' involvement and availability. UMS, which are built with robust personal data protection protocols, ensure trouble -free, coordinated experience in health care that improves the patient's results and strengthens public health resistance in India.

### **2.3.1 ADVANTAGES OF PROPOSED SYSTEM**

- Integrated and streamlined care.
- Enhanced patient empowerment.
- Early disease outbreak detection.
- Improved accessibility and support.
- Secure data sharing and privacy compliance.
- Efficient resource utilization.
- Scalable and flexible architecture.

## **CHAPTER 3**

### **REQUIREMENT ANALYSIS**



### **3.1 FEASIBILITY STUDY**

The feasibility study for the unified medical system (UMS) evaluates the technical, operational and economic viability of the development of centralized platform for health care for India. It examines contemporary Indian healthcare infrastructure, data security and challenges in existing systems together with the potential acceptance of the parties involved. Through interviews, data collection and research analysis, the study identifies the strengths and challenges associated with UMS implementation. Technical feasibility is supported by the cultivation of internet infrastructure and cloud technology; Operating feasibility relies on users' training and integration of the system; And economic feasibility suggests long -term benefits, including reduced health care costs and improved results. This assessment emphasizes UMS potential with careful planning and recommended implementation.

#### **3.1.1 Economical Feasibility**

The economic feasibility of the unified medical system (UMS) focuses on its potential to provide long -term financial benefits that outweigh the cost of development, implementation and maintenance. UMS promises to save costs for healthcare providers and improve the results for patients by streamlining health care processes. The sustainable financing model could be created through a combination of public and private investments with possible user charges for advanced services. Although initial costs may be significant, the expected return on investment (ROI) from increased health care efficiency and better resource allocation makes UMS economically promising.

#### **3.1.2 Technical Feasibility**

The technical feasibility of the unified medical system (UMS) is supported by the growing Indian Internet infrastructure, especially in urban areas, and the scalable possibilities of cloud computing for storage and access to data. Implementation of strong data security protocols such as encryption and alignment with personal data protection standards ensures patient data protection. The integration of UMS into existing health care systems using standardized formats (eg HL7 FHIR) promotes interoperability.

#### **3.1.3 Behavioral Feasibility**

The behavioral feasibility of the unified medical system (UMS) examines the willingness and ability of users - including patients, health care providers and administrative staff - to accept and

effectively use a new platform. The key considerations include:

- **User acceptance:** Success of UMS largely depends on its acceptance by the parties involved. Patients and healthcare providers must recognize the benefits of the system such as improved access to medical records, effective meeting planning and increased communication.
- **Training and support:** Comprehensive educational programs will be essential to ensure that users are satisfied with the new system. Continuous support and resources can help users efficiently through the platform and reduce resistance to change.
- **Applicability and design of the interface:** A user -friendly interface adapted to different groups of users (patients, doctors, hospital staff) will make it easy to navigate and wiring. Positive user experience is decisive for long -term acceptance.
- **Cultural factors:** Understanding cultural attitudes to technology and health care is essential. Acceptance of acceptance and engagement can adapt communication and training to deal with these factors.

### **3.1.4 Feasibility Study Questionnaire**

#### **Project Overview:**

The aim of this project is to develop a comprehensive unified medical system (UMS) for India. The UMS deals with the challenges of fragmented healthcare data, limited accessibility to medical records and control of reactive diseases.

#### **Main objectives**

- Increase the effectiveness of healthcare through data sharing, booking meetings and securing access to medical record
- Implement the supervision of data -based diseases by machine learning for early detection of focus.
- Using patients with support for Chatbot patients to access information and improve health education.

#### **System Scope:**

The initial phase will focus on the development of a minimum viable product (MVP) with a core function. This MVP will be piloted in a particular region in front of a nationwide implementation.

#### **Target audience**

- Patients (the general public)
- Doctors and other healthcare providers (doctors, nurses, etc.)
- Hospital (hospital administration)

- Laboratory technicians
- Public health officials
- Epidemiologists
- Authorized Scientists

**Industry/Domain:**

Healthcare

**Data Collection Contacts:**

Dr. Ashwathy J MBBS

[aswathyjayan@gmail.com](mailto:aswathyjayan@gmail.com)

Nurse. Anunand

+917356076832

**Questionnaire:**

1. Do you currently utilize a digital system for patient appointment booking and management? (Yes/No)

Yes, we use a basic online system, and integrate with all departments in a single hospital.

2. How often do you share patient medical records with other healthcare providers (e.g., specialists, hospitals)? (Always/Sometimes/Rarely/Never)

We share records occasionally with specialists, typically by physical records and emails, which are insecure.

3. In your experience, what are the biggest challenges related to managing patient medical records in the current system? (e.g., data fragmentation, accessibility, security)

Fragmentation is a big issue. Labs and specialists often have separate systems, making it time-consuming to get a complete picture of a patient's history.

4. How much time on average do you spend per day searching for or retrieving patient medical records? (Minutes/Hours)

On average, 5-10 minutes per patient, searching through different systems and paper charts.

Perceptions on a Unified Medical System:

5. How beneficial do you believe a national Unified Medical System (UMS) would be for improving patient care in India? (Very beneficial/Somewhat beneficial/Neutral/Not beneficial)

I believe a UMS could be very beneficial for patient care. Streamlined records and easier data sharing would improve efficiency and continuity of care.

6. What functionalities within a UMS would be most valuable to you in your daily practice?

(e.g., secure data sharing, appointment scheduling, patient portal access)

Secure data sharing, online appointment scheduling, and a patient portal for accessing medical history would be most valuable.

7. Would you be comfortable using a UMS for accessing patient information from other hospitals or clinics? (Yes/No)

Yes, I would be comfortable using a UMS to access patient information from other hospitals, as long as it's secure and reliable.

Security and Privacy:

8. What security measures are most important to you regarding patient data stored within a UMS? (e.g., encryption, access control, audit logs)

Encryption, access control based on user roles, and a strong audit log to track data access are crucial.

9. How can a UMS ensure patient privacy while still facilitating data exchange for improved healthcare delivery?

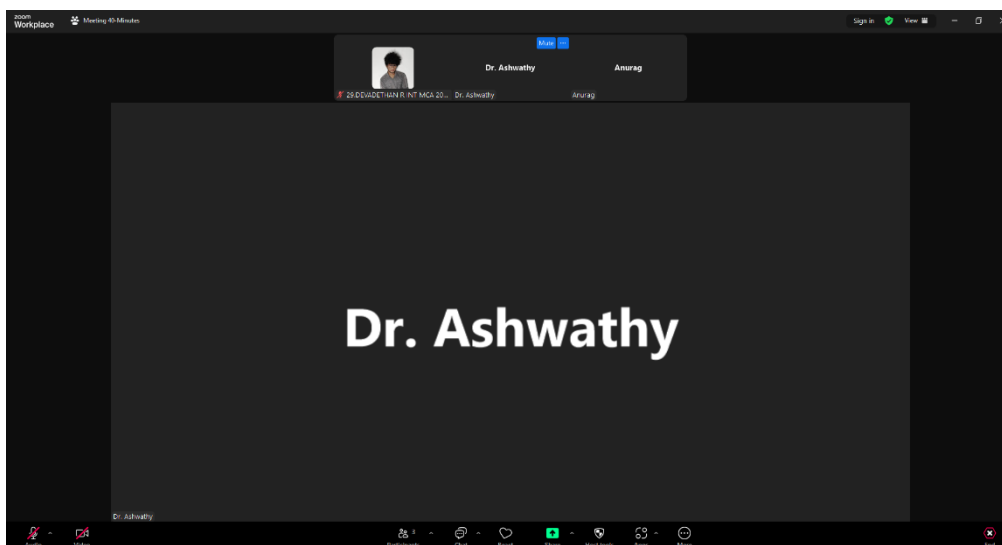
A UMS should ensure privacy through strong authentication, clear patient consent for data sharing, and anonymized data for research and outbreak detection.

Looking Ahead:

10. Do you have any suggestions or specific requirements for a UMS that would be helpful in your practice?

A user-friendly interface for doctors and patients would be essential. Additionally, the system should be designed with offline functionality in case of internet disruptions.

### 3.1.5 Geotagged Photograph



### 3.1 SYSTEM SPECIFICATION

#### 3.2.1 Hardware Specification

Processor - Intel Core i3

RAM - 8 G B

Hard disk - 2 5 6 G B S S D o r h i g h e r

#### 3.2.2 Software Specification

Front End - BOOTSTRAP v 5.2, Daisy UI

Back End - FLASK, Cloud Mongo, Lang Chain

Database - MongoDB

Client on PC - Windows 7 and above.

Technologies used - JS, HTML5, AJAX, Bootstrap, FLASK, Pytorch (for deploy model), Lang Chain (for agents), Google AI Studio (model inference)

### 3.3 SOFTWARE DESCRIPTION

#### 3.3.1 Eg.FLASK

The flask is a lightweight and flexible web frame written in Python, designed to quickly create web applications. It monitors a modular approach, offers only basic tools, and allows developers to add extensions as needed to connect to database, forms processing, verification and more. The flask works on the WSGA (Interface Web Server Gateway Interface) and uses for the Jinja2 Templing and provides robust tools for rendering a dynamic HTML with minimal effort. Being a "micro" is a non -optionate flask, which means that it does not force any project structure or addiction . The flask is a powerful but minimalist web frame written in Python, ideal for developers who prefer simplicity and checking in the development of web applications. As a “micro -line”, the flask includes only basic components for the requirements for handling, routing and templing, which is exceptionally light and flexible. It uses WSGA (Interface Web Server Gateway Interface) to process web requirements and yinja2, fast and expressive Templing engine, to create dynamic HTML responses.

One of the features of Blacker is its extensibility - Developers can choose specific libraries or

extensions for functions such as database integration, form and verification validation without unnecessary dependencies. This "plug-and-play" approach allows developers to create simple applications and scalable systems at the production level. The flask is also highly compatible with other Python libraries, which is popular for projects that include machine learning, data visualization or comprehensive business logic. Thanks to its direct syntax and clear documentation, they make the best choice for beginners, while its flexibility and powerful extensions attract experienced developers who need a customized, high -performance application magazine.

### **3.3.2Eg. MongoDB**

MongoDB is a NoSQL database known for its scalability, flexibility, and document-based structure. Unlike traditional relational databases that store data in tables with rows and columns, MongoDB stores data in BSON (Binary JSON) documents, allowing it to handle unstructured or semi-structured data efficiently. Each document is a self-contained data unit, making MongoDB well-suited for applications that require rapid data integration, real-time analytics, and frequent updates to data structure without disrupting the entire system.

MongoDB's schema-less nature allows developers to add or modify fields easily without predefined data schemas. This flexibility makes it ideal for projects that evolve quickly or handle large volumes of diverse data types, like social media applications, e-commerce platforms, and IoT solutions. It also supports a rich query language, indexing, aggregation, and powerful features like horizontal scaling (sharding) and replication for high availability, which enhances its performance and fault tolerance. MongoDB integrates well with modern tech stacks, making it a popular choice for full-stack developers and organizations adopting cloud-native, distributed application architectures.

## **CHAPTER 4**

### **SYSTEM DESIGN**

## 4.1 INTRODUCTION

Design is the first step into the development phase for any technical product or system. Design is a creative process. Good design is the key to an efficient system. The term "design" is defined as "the process of using different techniques and principles to define a process or system in sufficient detail to allow its physical realization". It can be defined as the process of using different techniques and principles to define a device, process, or system in a sufficient detailed system to make it possible to make it physical. Software design sits in the technical core of the software engineering process and is applied regardless of the developmental paradigm used. The design of the system develops architectural details needed to create a system or product. As with any systematic approach, this software has also undergone the best possible design phase of fine tuning all efficiency, performance and accuracy. The design phase is a transition from a user -oriented document to a document for programmers or database staff. The system design goes through two phases of development: logical and physical design.

## 4.2 UML DIAGRAM

UML is a standard language for specification, visualization, design and documenting artifacts of software systems. UML was created by a group for managing objects (OMG) and designing the UML 1.0 specification was designed to OMG in January 1997. UML means a uniform modeling language. UML differs from other common programming languages such as C ++, Java, Cobol, etc. UML is a picture language used to create software plans. UML can be described as a language of visual modeling for visualization, specification, design and software design. Although UML is generally used to model software systems, it is not limited at this limit. It is also used to model systems without software. For example, the process flow in the manufacturing unit, etc. UML is not a programming language, but the tools can be used to generate code in different languages using UML diagrams. UML has a direct relation to object -oriented analysis and design. After some standardization, UML has become the standard of OMG. All elements, relationships are used to create a complete UML diagram and the diagram is a system. The visual effect of the UML diagram is the most important part of the process. All other elements are used to complete. UML contains the following nine diagrams.

- Class diagram
- Object diagram
- Use case diagram



- Sequence diagram
- Activity diagram
- State chart diagram
- Deployment diagram
- Component diagram

### **4.2.1 USE CASE DIAGRAM**

A Use Case Diagram is a visual representation of a system's functionality, highlighting the interactions between users (actors) and the system itself. It serves as a high-level overview, showing how users achieve their goals by utilizing the system's various functions, represented as "use cases." These diagrams are commonly used in the early stages of software design to communicate system requirements, focusing on what the system does rather than how it does it.

In a Use Case Diagram:

- Actors represent the users or external systems that interact with the system, such as admins, customers, or third-party services.
- Use Cases represent specific functionalities or actions the system performs, such as "Login," "Register," "Purchase Product," or "View Orders."
- Associations are the lines connecting actors and use cases, indicating interactions or relationships.
- System Boundary is a rectangle that encapsulates all the use cases, defining the scope of the system.

Use Case Diagrams are useful for understanding user requirements, identifying main functionalities, and defining roles within a system. They also serve as a basis for further detailed analysis and design, helping ensure the system aligns with user needs.

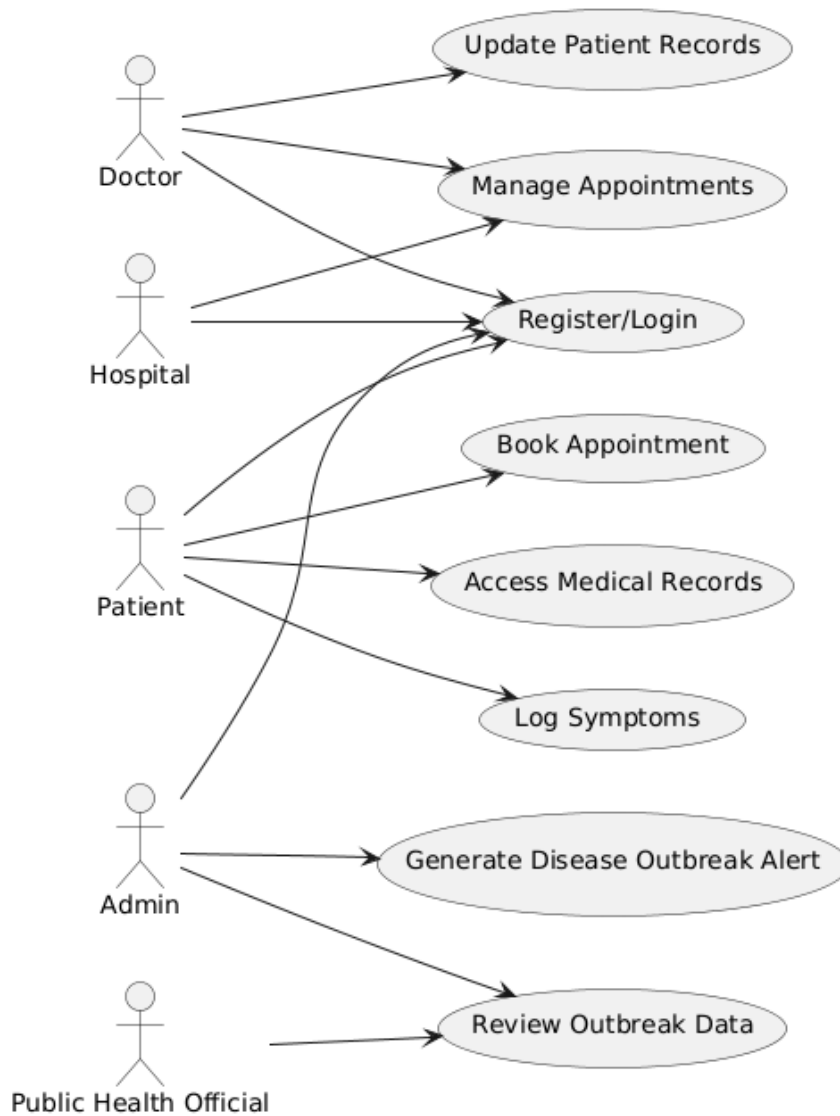


Figure 1. Use Case Diagram

#### 4.2.2 SEQUENCE DIAGRAM

A sequence diagram is a type of interaction diagram in Unified Modeling Language (UML) that shows how objects interact in a particular sequence within a system. It visually represents the flow of messages or events between objects over time, illustrating how and in what order operations are carried out.

Key components of a sequence diagram include:

**Actors:** Represent users or external systems interacting with the system.

**Objects or Classes:** Represent entities within the system that perform actions.

**Lifelines:** Vertical lines showing the life span of an object during the sequence.

**Activation Bars:** Rectangles on the lifelines indicating the period when an object is active.

**Messages:** Arrows between objects indicating communication; they can be synchronous or asynchronous.

**Return Messages:** Dashed lines that show the return or response after a message is processed.

### Use and Advantages:

Sequence diagrams are integral to modelling the dynamic aspects of systems and are especially useful during the design phase for:

- **Clear Communication:** They provide a visual, step-by-step breakdown of processes, improving communication among team members.
- **Requirement Validation:** Diagrams help validate requirements by demonstrating how processes should flow and helping stakeholders confirm intended functionality.
- **System Design and Debugging:** Developers use sequence diagrams to refine system architecture, identify bottlenecks, and troubleshoot issues by understanding exact message flow.

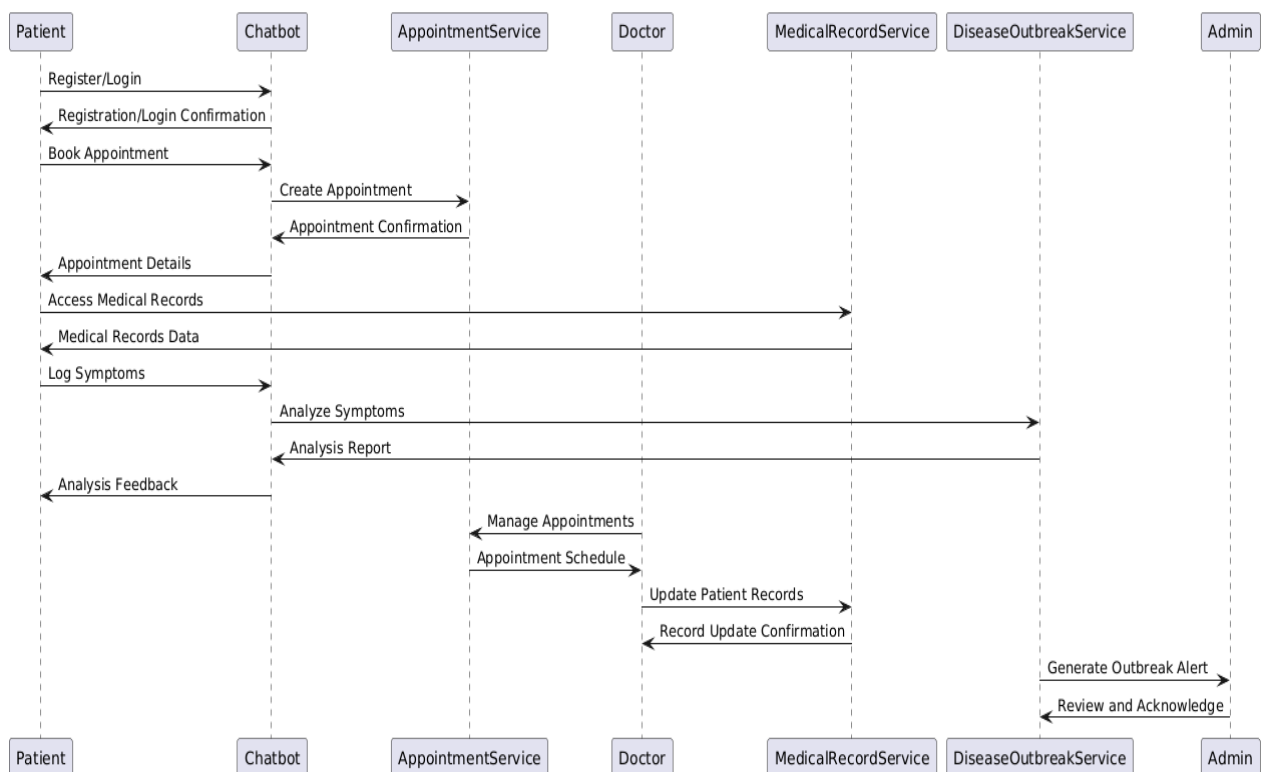


Figure 2. Sequence Diagram

### 4.2.3 State Chart Diagram

A state chart diagram, or state diagram, illustrates the various states an object or system component can undergo throughout its lifecycle, along with the transitions between those states triggered by

events. Each state is represented as a rounded rectangle, while transitions are shown as arrows connecting the states, labeled with the events that cause the transitions.

### Key Elements:

**States:** Conditions an object can be in (e.g., Idle, Processing).

**Transitions:** Arrows indicating changes between states, triggered by events.

**Events and Actions:** Triggers for state changes and activities that occur during transitions.

### Use and Benefits:

State chart diagrams model dynamic behavior, simplify complex systems, and validate that all expected states and transitions are covered. They are commonly used in applications with state-dependent behavior, such as user interfaces and workflow management systems, providing a clear view of how an object responds to various events.

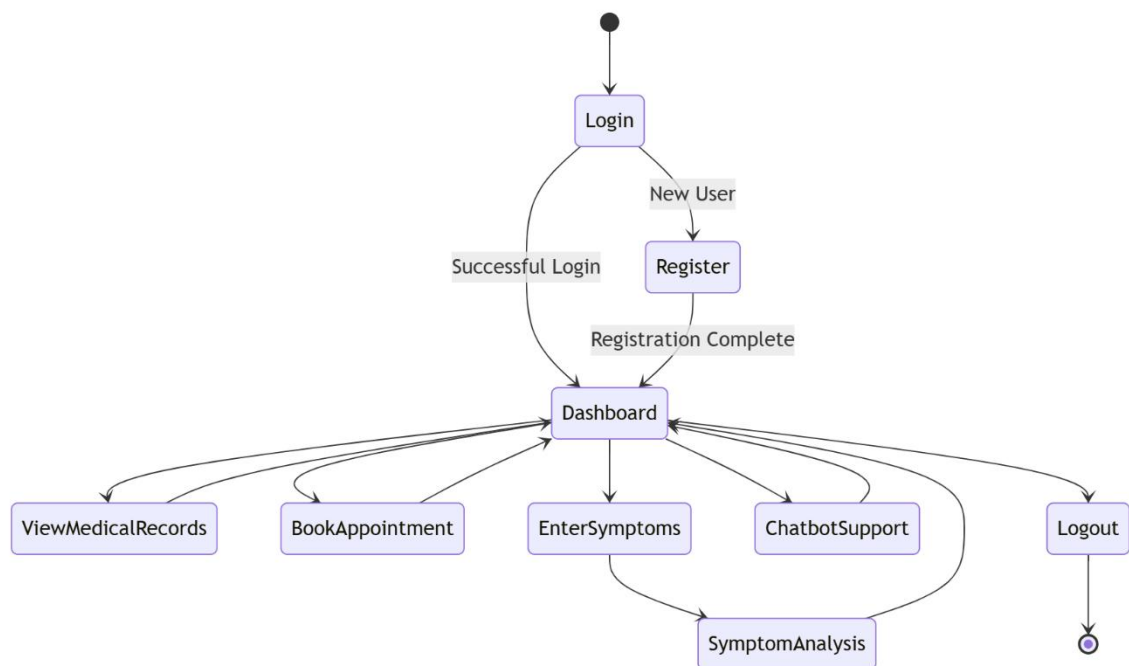


Figure 3. State chart Diagram

## 4.2.4 Activity Diagram

The activity scheme is a diagram of behavior in the language of unified modeling (UML), which illustrates the flow of activities or actions in the system and emphasizes the dynamic aspects of processes. It is equipped with key elements such as activities represented by rounded rectangles that indicate tasks performed in the system. The scheme begins with the initial node displayed as a filled circle, and closes the end node, a circle enclosed in another circle, indicating the completion of the process. Transitions represented by arrows connect activities to demonstrate the flow from one task to another. The decision nodes, illustrated as diamonds, indicate points where the flow may vary on the basis of specific conditions, while the fork and connections (displayed as rods)

represent parallel activities. Activities schemes are particularly valuable for modeling complex workflows, scenarios of use and business processes because they clarify sequence of operations, identify potential obstacles and facilitate communication between the parties involved in visualization of activities and their interdependence.

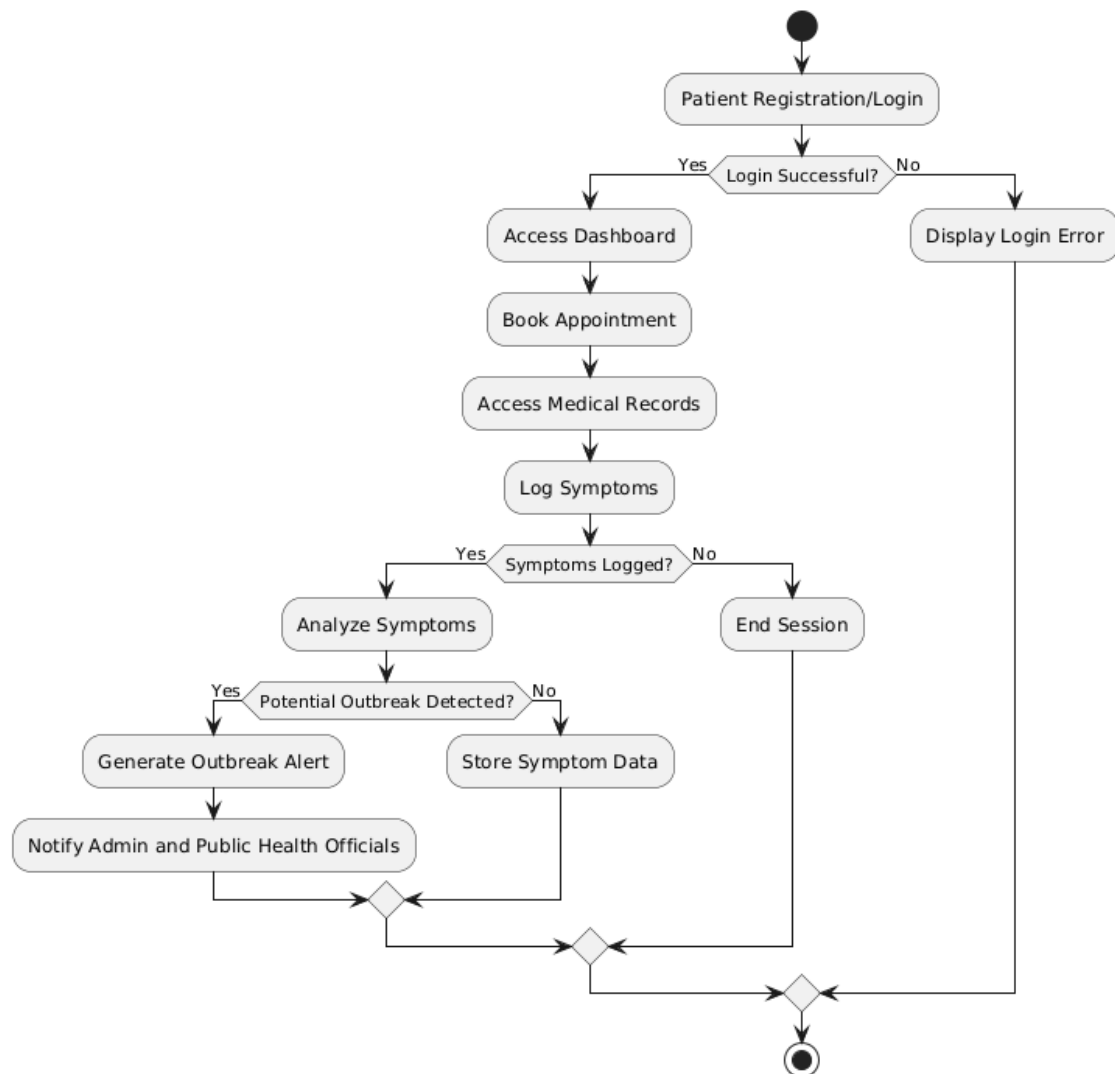


Figure 4. Activity Diagram

#### 4.2.5 Class Diagram

A class diagram is a static structure diagram in Unified Modeling Language (UML) that represents the system's classes, their attributes, methods, and the relationships among them. It provides a visual blueprint of the system's architecture, highlighting how different classes interact and how data is organized. Each class is depicted as a rectangle divided into three sections: the top section contains the class name, the middle section lists its attributes, and the bottom section shows its methods (or operations). Relationships between classes are illustrated through lines, with various notations to

indicate the type of relationship, such as inheritance (generalization), associations, aggregations, and compositions. Class diagrams are essential for object-oriented design, as they help in understanding the system's structure, guiding the implementation of classes, and ensuring that all components interact correctly.

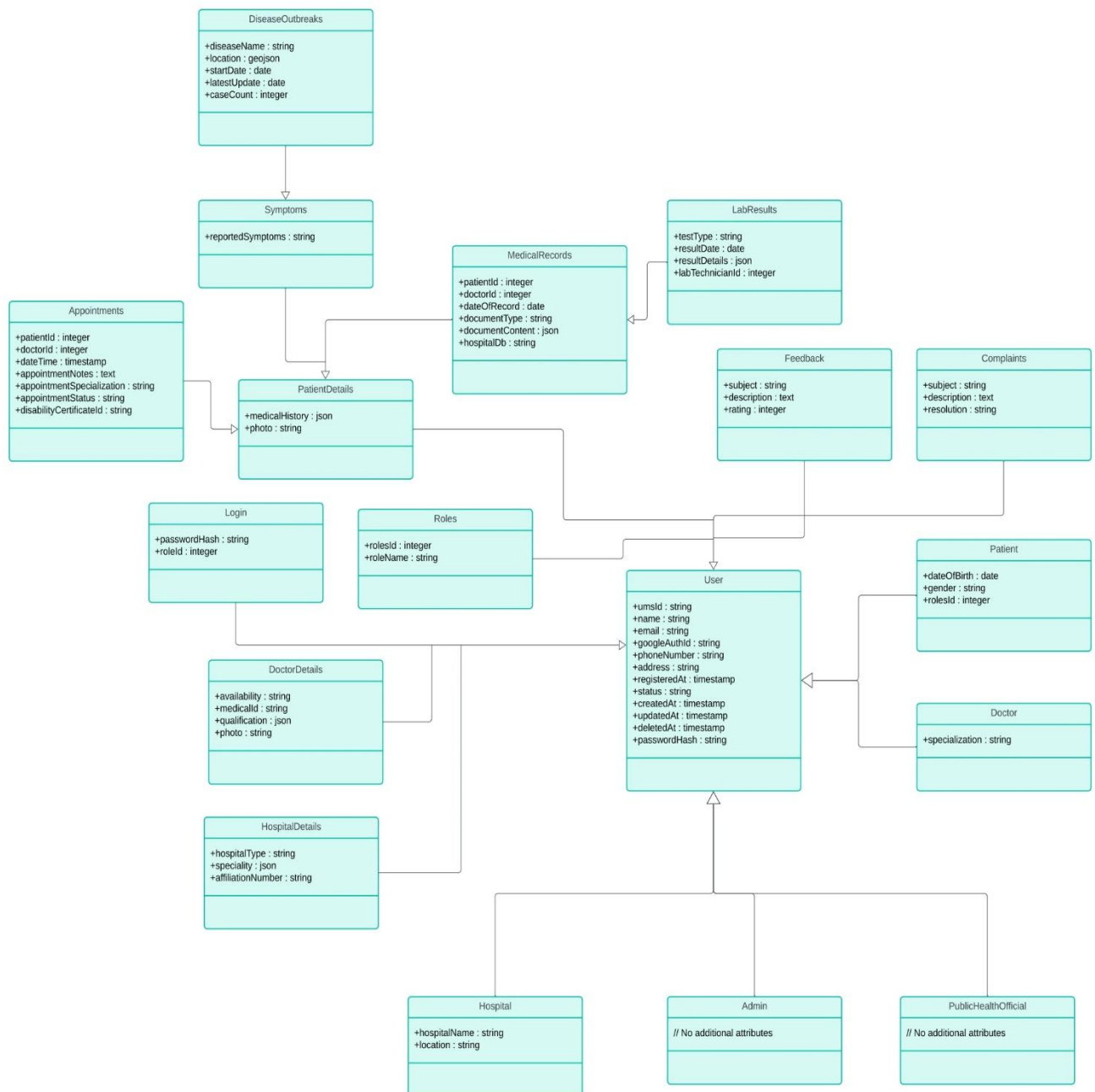


Figure 5. Class Diagram

#### 4.2.6 Object Diagram

An object diagram is a type of static structure diagram in Unified Modeling Language (UML) that

provides a snapshot of the instances of classes (objects) and their relationships at a specific moment in time. Unlike class diagrams, which focus on the blueprint of classes and their relationships, object diagrams emphasize the concrete examples of those classes in a particular state.

Each object is represented by a rectangle, similar to a class in a class diagram, but it includes the object's name and its current state or attribute values. Lines connecting the objects illustrate their relationships, such as associations, aggregations, or compositions. Object diagrams are particularly useful for visualizing complex relationships and interactions among objects in a system, demonstrating how they collaborate to perform tasks or represent data.

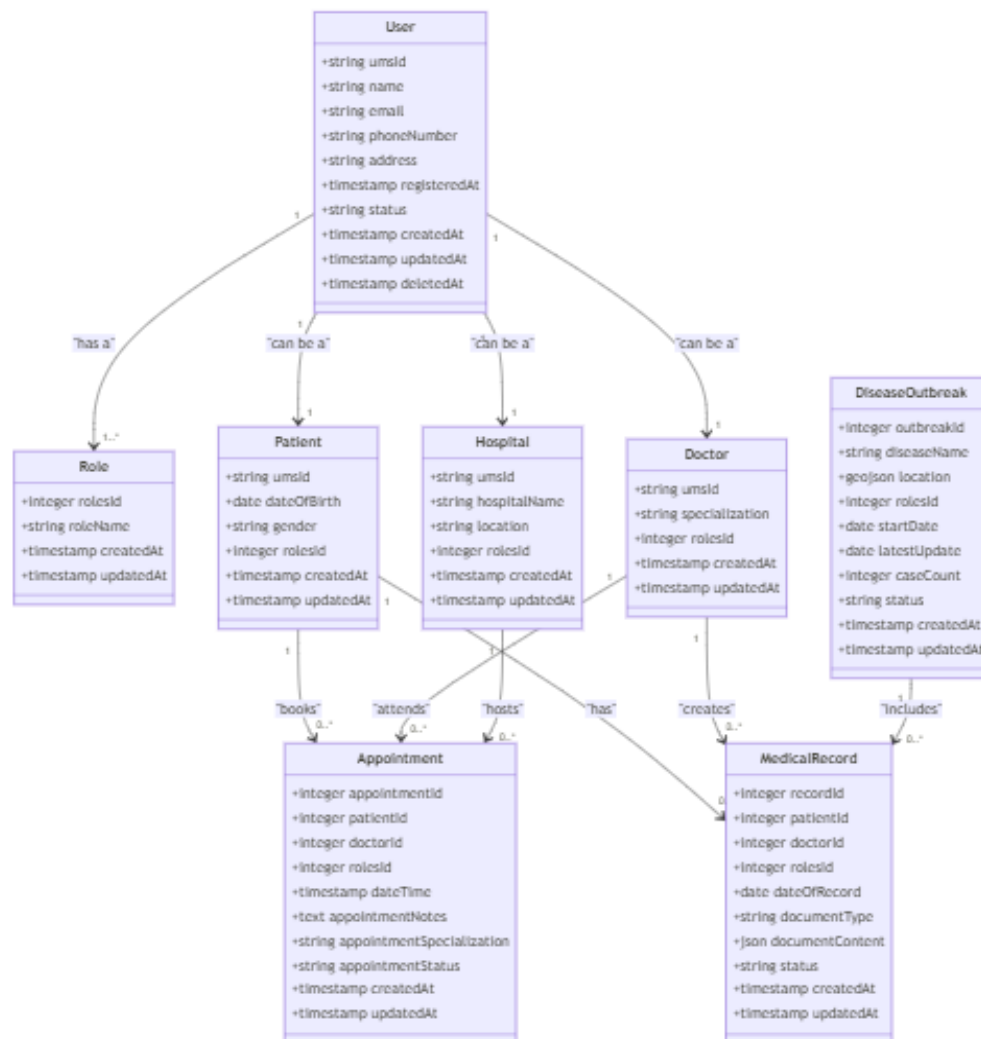


Figure 6. Object Diagram

#### 4.2.7 Component Diagram

A component diagram is a type of structural diagram in Unified Modeling Language (UML) that represents the physical components in a system, such as software applications, libraries, and

packages. It illustrates how these components interact with one another and their dependencies, providing a high-level view of the system's architecture.

Components are depicted as rectangles with two smaller rectangles on the left side, indicating that they can be independently developed and deployed. They may also include interfaces that define the operations available for other components to use. Component diagrams are particularly useful for modeling complex systems, as they help to visualize the relationships between various software components, their functionalities, and how they fit into the overall system architecture. This aids in understanding the system's modular structure, facilitates better planning for system integration, and enhances the communication between developers.

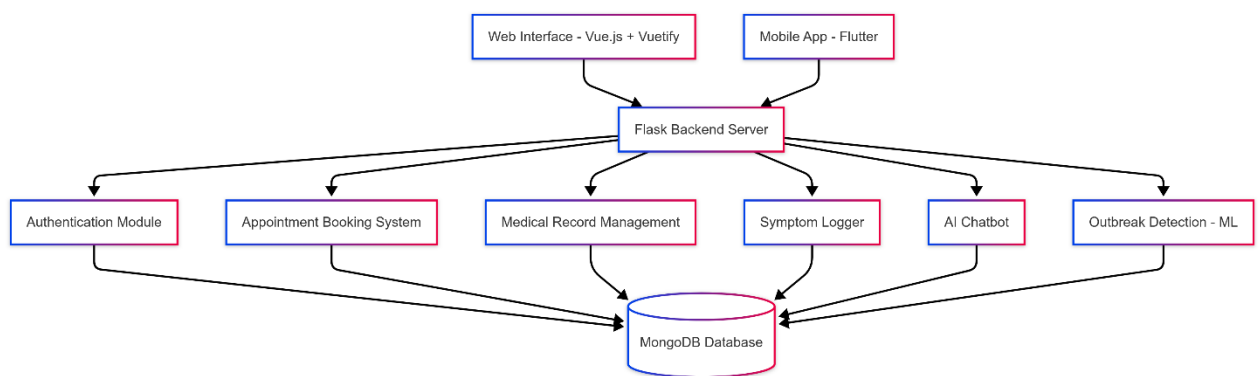


Figure 7. Component Diagram

#### 4.2.8 Deployment Diagram

A deployment diagram is another type of UML diagram that shows the physical deployment of artifacts on nodes. It illustrates how software is distributed across hardware and how different components communicate within the system infrastructure. Deployment diagrams are particularly useful in visualizing the physical arrangement of software components in a distributed system, such as client-server architectures or cloud-based applications.

In a deployment diagram, nodes represent physical devices (like servers, computers, or mobile devices), and artifacts (such as executables, libraries, or database schemas) represent the software deployed on those nodes. Connections between nodes depict communication paths, showing how information flows within the system.



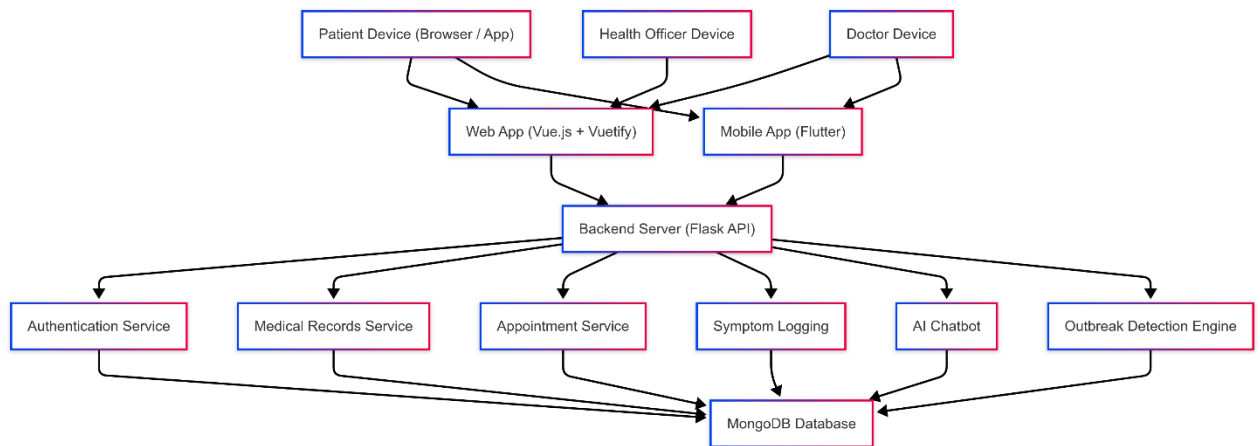


Figure 8. Deployment Diagram

## 4.3 USER INTERFACE DESIGN USING FIGMA

Form Name: Login

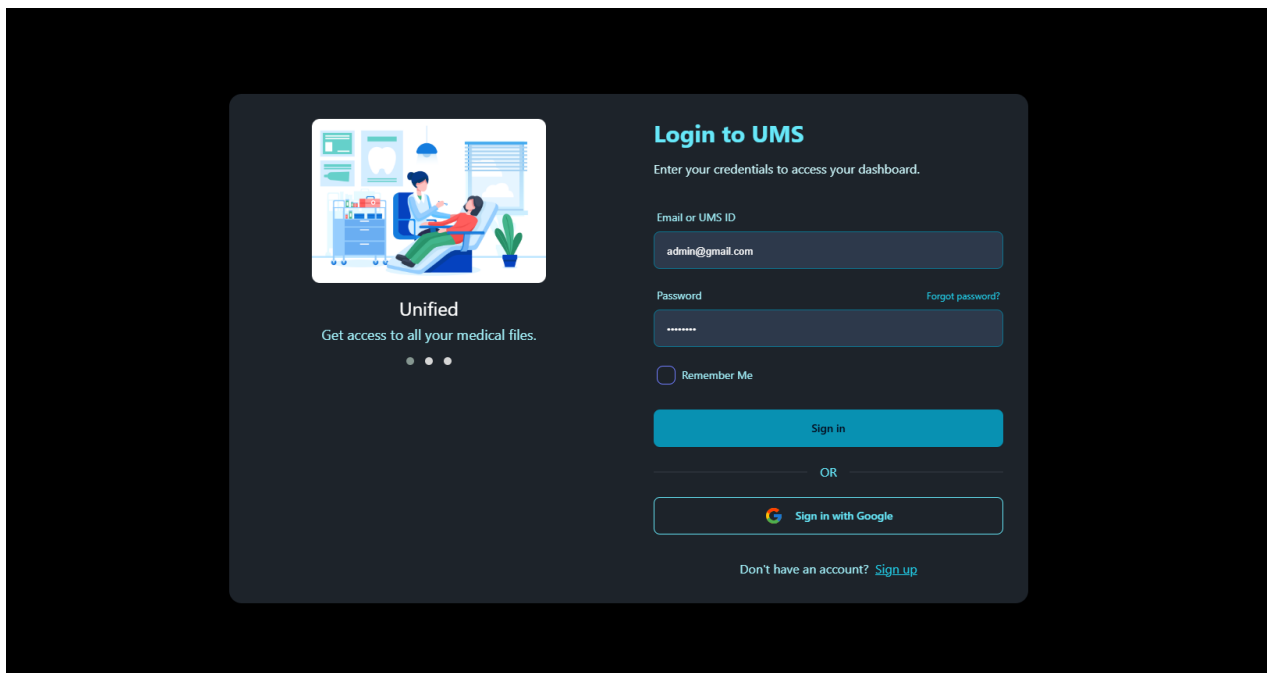
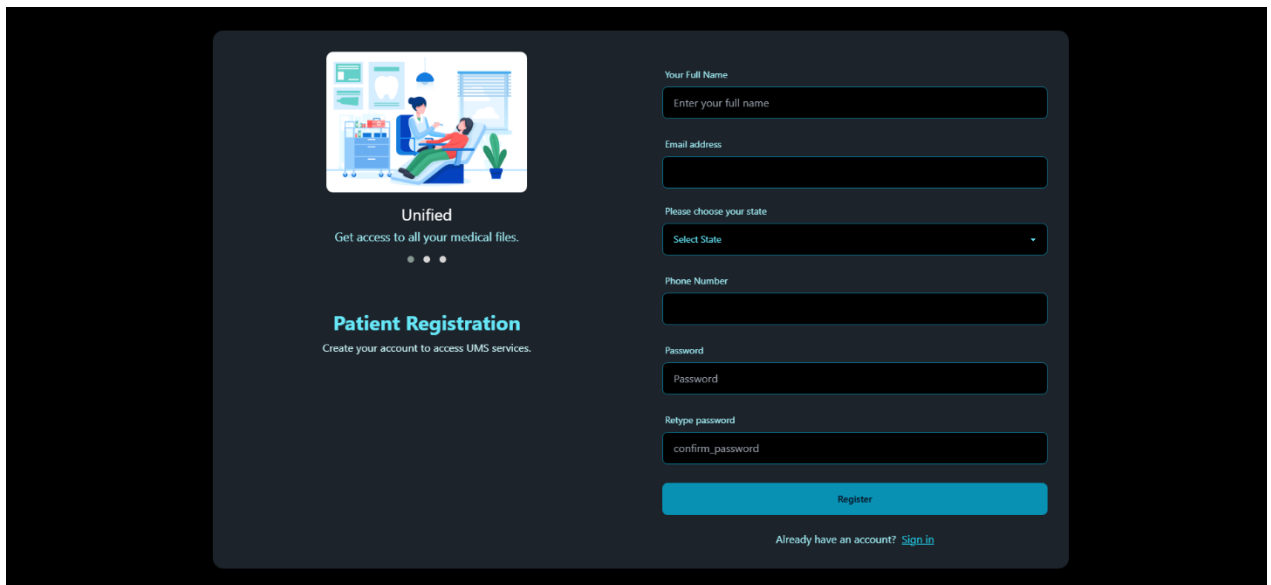


Figure 1. Login Page

Form Name: Register



The image shows a patient registration form titled "Unified Patient Registration". It includes an illustration of a doctor and a patient. The form fields are: "Your Full Name" (text input), "Email address" (text input), "Please choose your state" (dropdown menu), "Phone Number" (text input), "Password" (text input), and "Retype password" (text input). A "Register" button is at the bottom, and a link "Already have an account? Sign in" is below it.

Figure 2. Register Page

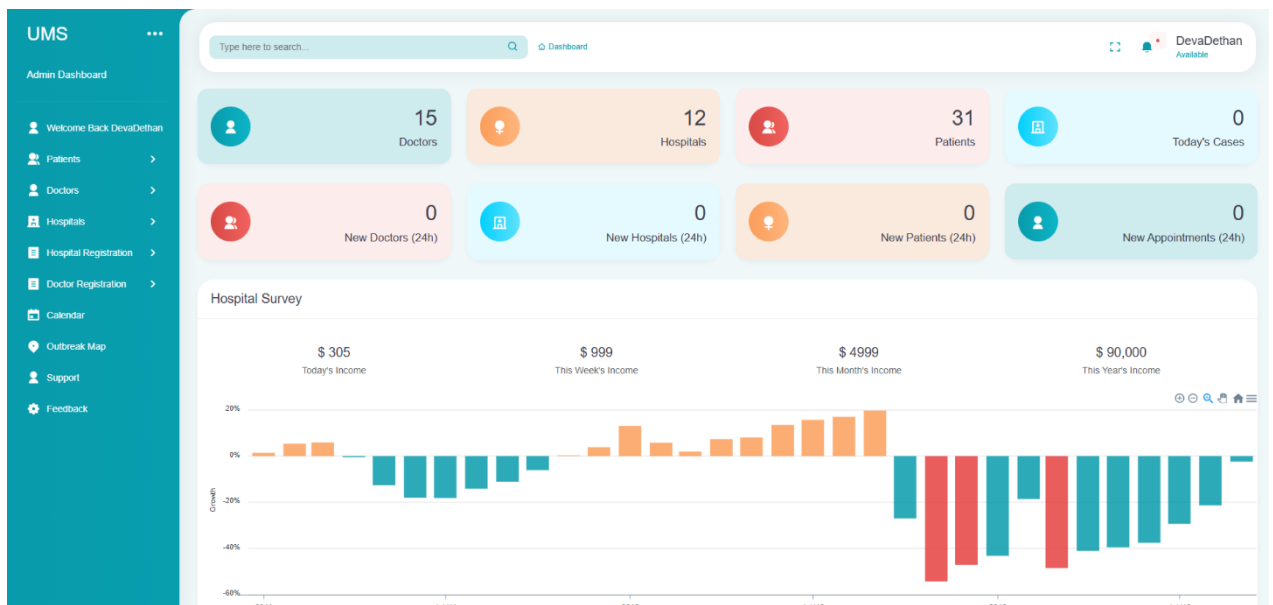
**Form Name: Admin Home Page**

Figure 3. Admin Home Page

**Form Name: Dcotor Dashboard**

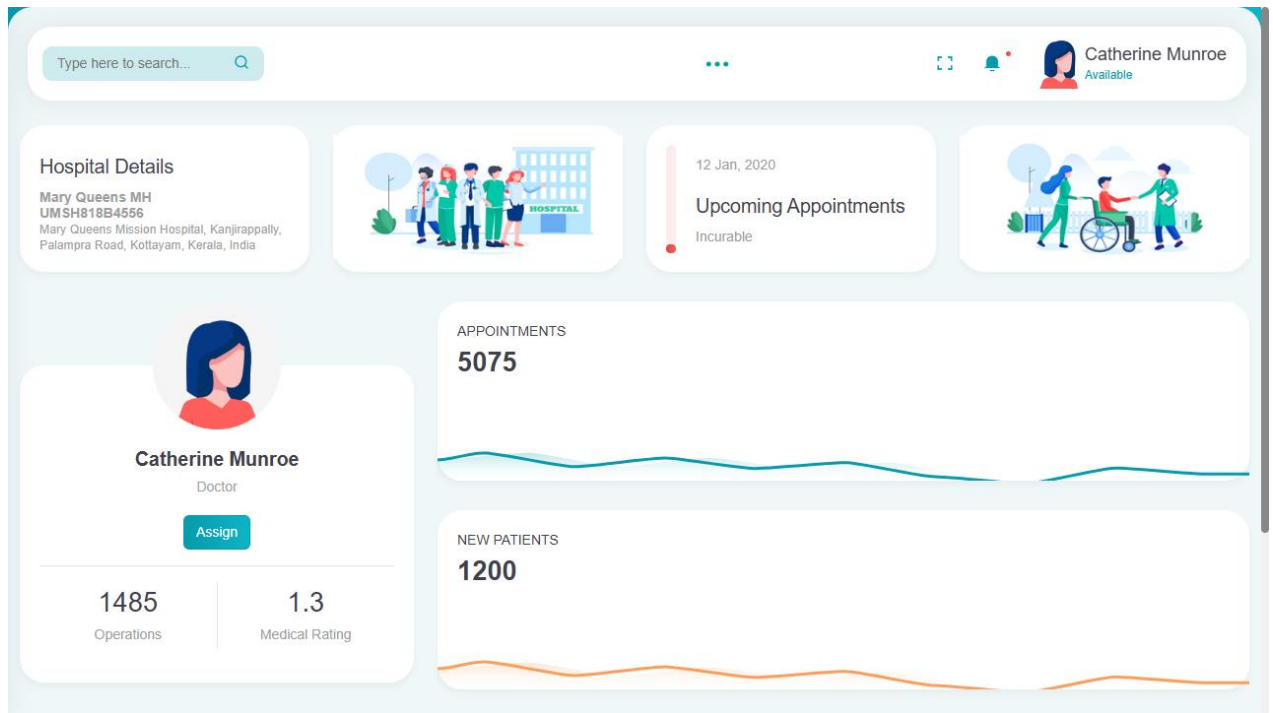


Figure 4. Doctor Dashboard

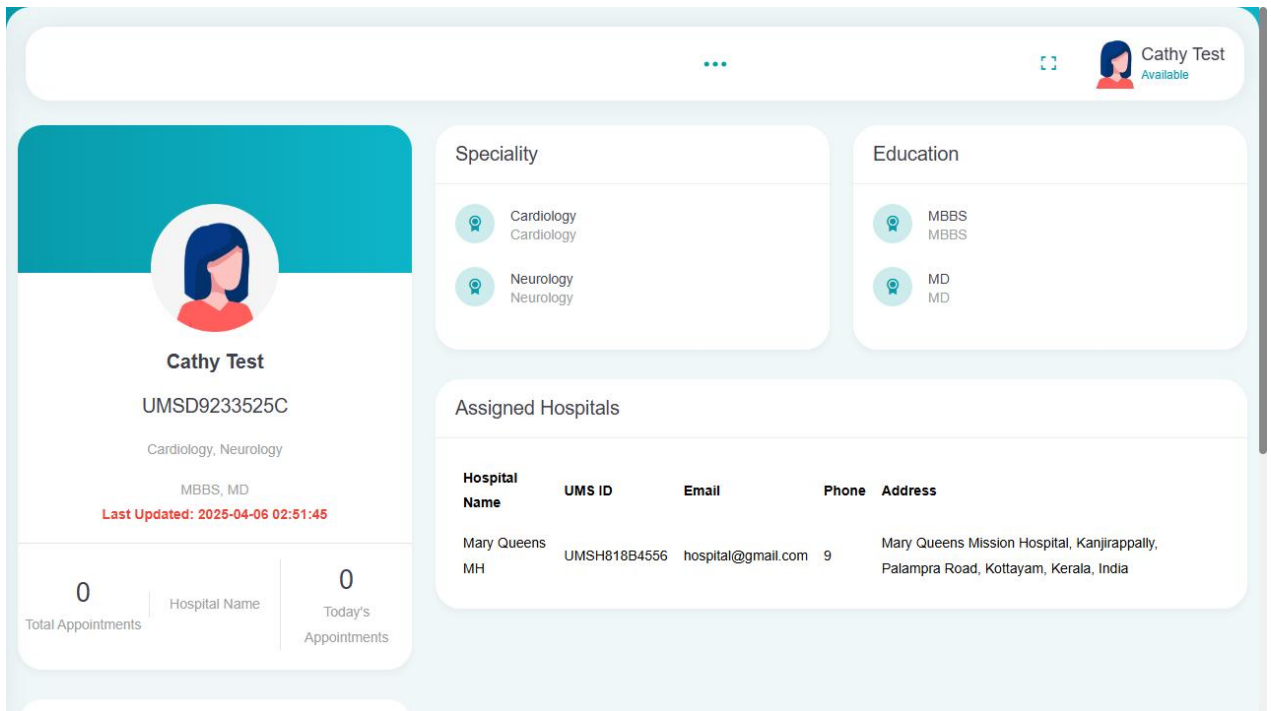
**Form Name: Edit profile page**

The Edit Profile form for Catherine Munroe includes a search bar, a tabbed interface with 'Personal Information' selected, and a warning message. The form contains the following fields:

Field	Value
Status	Active
Reason if any	Optional
Name (At least 3 characters)	Catherine Munroe
Date Of Birth	13-11-2002

**Update Carefully:**  
Please update the following fields carefully:

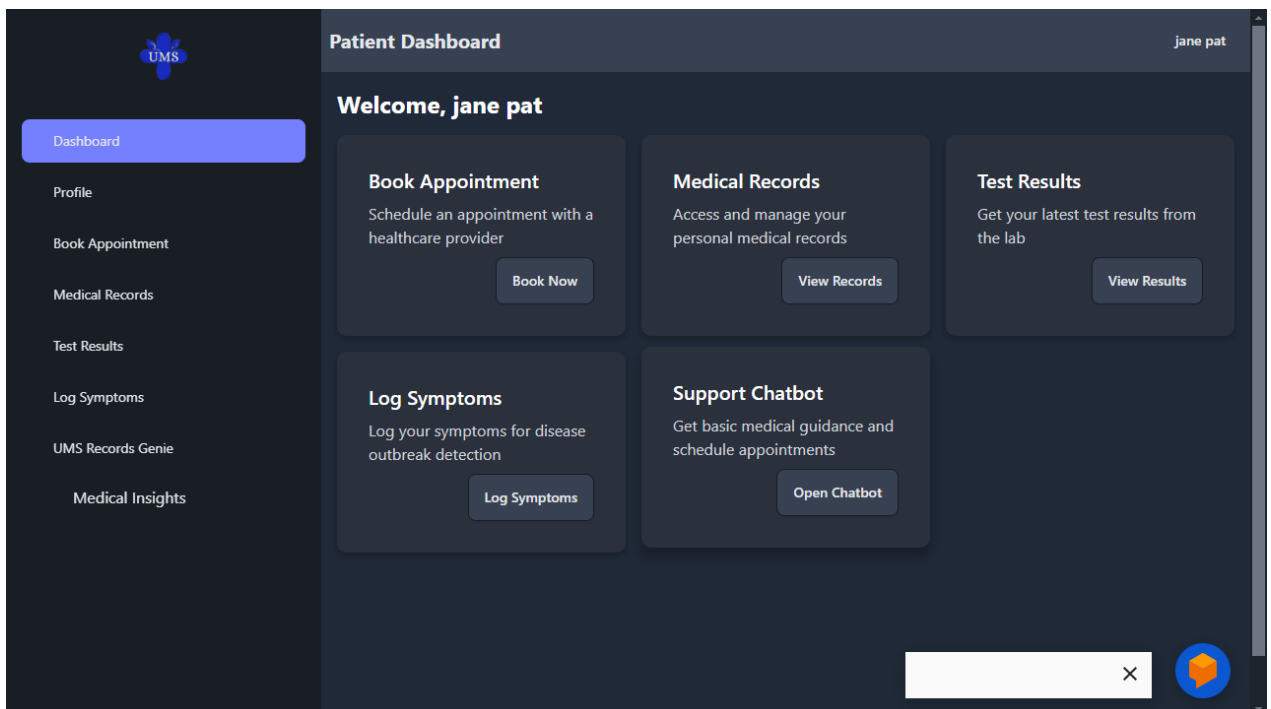
Figure 5. Edit Profile

**Form Name: Doctor profile page**

The Doctor Profile page for Cathy Test displays her personal and professional information. On the left, a profile card shows her name, UMS ID (UMSD9233525C), specialities (Cardiology, Neurology), and education (MBBS, MD). Below this, statistics for total and today's appointments are shown. The right side features sections for Speciality, Education, and Assigned Hospitals. The Assigned Hospitals section includes a table with hospital details.

Hospital Name	UMS ID	Email	Phone	Address
Mary Queens MH	UMSH818B4556	hospital@gmail.com	9	Mary Queens Mission Hospital, Kanjirappally, Palampra Road, Kottayam, Kerala, India

Figure 6. Doctor Profile

**Form Name: Patient Dashboard**

The Patient Dashboard for jane pat provides a central hub for medical services. The left sidebar lists navigation options: Dashboard, Profile, Book Appointment, Medical Records, Test Results, Log Symptoms, UMS Records Genie, and Medical Insights. The main content area is titled 'Welcome, jane pat' and contains five interactive cards: Book Appointment, Medical Records, Test Results, Log Symptoms, and Support Chatbot. Each card includes a brief description and a primary action button.

Figure 7. Patient Dashboard

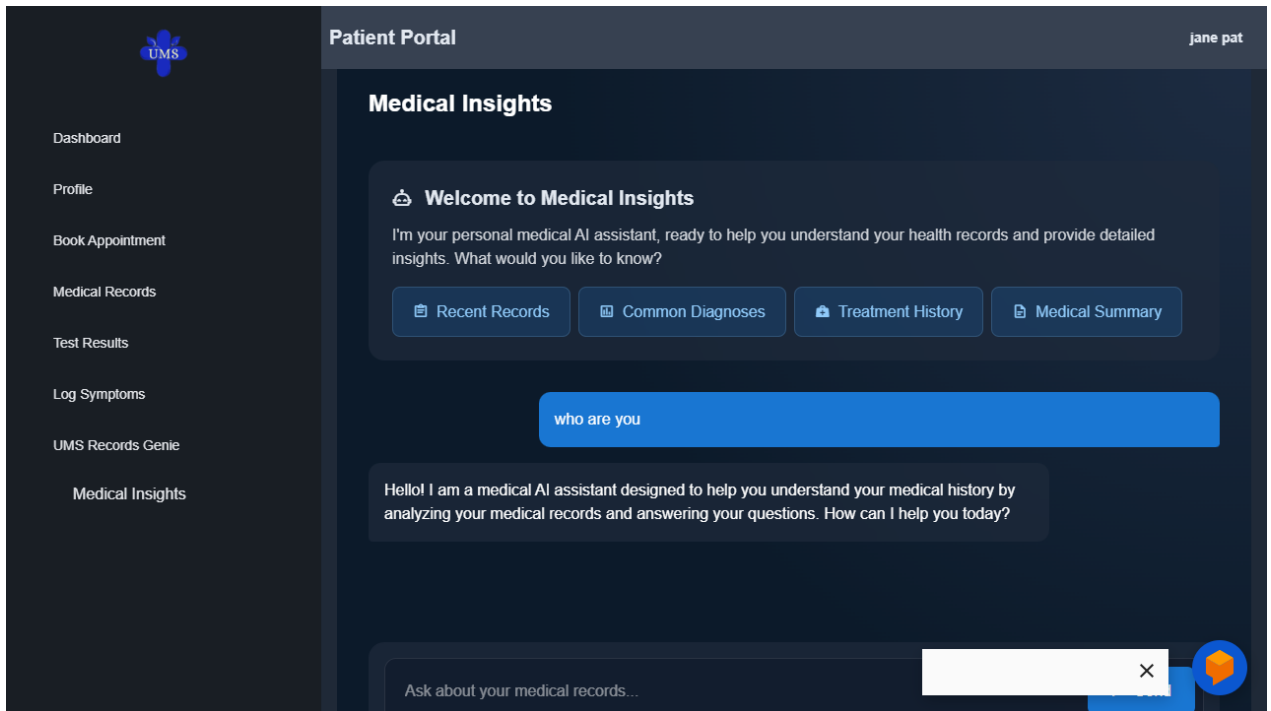
**Form Name: Chatbot**

Figure 8. AI Assistant

## 4.4 DATABASE DESIGN

Database design is the structuring of data in an orderly fashion to ensure effective storage, retrieval, and management. UMS is an RDBMS-supported system and data is stored in tables with primary and foreign keys to ensure relationships between tables. Reducing redundancy, maintaining data integrity, and performance optimization for managing user accounts, products, rentals, and transactions are the focal points of design.

### 4.4.1 Non - Relational Database Management System (RDBMS)

In a non-relational database (NoSQL), data is organized in collections and documents rather than tables and rows, allowing for flexible schemas where each document in a collection can have a unique structure. Documents, similar to objects, contain fields that can store diverse data types, including arrays or nested documents, enabling rich and hierarchical data structures. Relationships are managed through embedding related data directly within a document or referencing other documents by ID, rather than enforcing strict referential integrity. Fields and data types are not strictly bound, allowing for flexibility in handling various data formats. Atomicity applies at the document level, ensuring that operations on a document are consistent as a whole. This flexibility and schema-less nature make NoSQL databases ideal for applications requiring scalability and rapid data access across large volumes of unstructured or semi-structured data.

### 4.4.2 Normalization

Data are grouped together in the simplest way so that later changes can be made with minimum impact on data structures. Normalization is formal process of data structures in manners that eliminates redundancy and promotes integrity. Normalization is a technique of separating redundant fields and breaking up a large table into a smaller one. It is also used to avoid insertion, deletion, and updating anomalies. Normal form in data modelling use two concepts, keys, and relationships. A key uniquely identifies a row in a table. There are two types of keys, primary key and foreign key. A primary key is an element or a combination of elements in a table whose purpose is to identify records from the same table. A foreign key is a column in a table that uniquely identifies record from a different table.

#### First Normal Form

The First Normal Form states that the domain of an attribute must include only atomic values and that the value of any attribute in a tuple must be a single value from the domain of that attribute. In other words, 1NF disallows “relations within relations” or “relations as attribute values within tuples”. The only attribute values permitted by 1NF are single atomic or indivisible values. The first step is to put the data into First Normal Form.

**Second Normal Form**

According to Second Normal Form, for relations where primary key contains multiple attributes, no non-key attribute should be functionally dependent on a part of the primary key. In this we decompose and setup a new relation for each partial key with its dependent attributes. Make sure to keep a relation with the original primary key and any attributes that are fully functionally dependent on it. This step helps in taking out data that is only dependent on a part of the key.

**Third Normal Form**

According to Third Normal Form, Relation should not have a non-key attribute functionally determined by another non-key attribute or by a set of non-key attributes. That is, there should be no transitive dependency on the primary key. In this we decompose and set up relation that includes the non-key attributes that functionally determines other non-key attributes. This step is taken to get rid of anything that does not depend entirely on the Primary Key. A relation is said to be in third normal form if only if it is in second normal form and more over the non key attributes of the relation should not be depend on another non-key attribute.

**Fourth Normal Form**

The fourth normal form (4NF) is a database normalization rule that further refines data modeling by addressing multi-valued dependencies. When a table contains multiple independent sets of repeating data, we can break it down into smaller tables, with each table containing only one set of related data.

This reduces data redundancy and improves data consistency by ensuring that each table represents a single, well-defined concept or entity. To achieve 4NF, we need to ensure that all multi-valued dependencies are removed from the table, and that each table contains only attributes that are functionally dependent on the primary key.

**Fifth Normal Form**

5NF is indeed the highest level of normalization in relational database design, and it deals with complex data models that involve multiple overlapping multi-valued dependencies. In 5NF, tables are decomposed into smaller tables in order to eliminate any possible redundancy caused by overlapping dependencies, while ensuring that there is no loss of data.

The goal of 5NF is to ensure that each table represents a single entity or relationship, and that the data is organized in a way that minimizes redundancy, eliminates anomalies, and improves data integrity.

### 4.4.3 Sanitization

An automated procedure called "sanitization" is used to get a value ready for use in a SQL query. This process typically involves checking the value for characters that have a special significance for the target database. To prevent a SQL injection attack, you must sanitize(filter) the input string while processing a SQL query based on user input. For instance, the user and password input is a typical scenario. In that scenario, the server response would provide access to the 'target user' account without requiring a password check.

### 4.4.4 Indexing

By reducing the number of disk accesses needed when a query is completed, indexing helps a database perform better. It is a data structure method used to locate and access data in a database rapidly. Several database columns are used to generate indexes. The primary key or candidate key of the table is duplicated in the first column, which is the Search key. To make it easier to find the related data, these values are kept in sorted order. Recall that the information may or may not be kept in sorted order.

## 4.5 TABLE DESIGN

### 1.Tbl\_user

Eg.Primary key: umsId

Eg.Foreign key: umsId references table **Tbl\_login**

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1.	umsId	string	primary key	Alpha-numeric UUIDv4, unique identifier for the user
2.	name	string		Name of the user
3.	email	string	unique	Email address of the user
4.	googleAuthId	string		Google authentication ID
5.	phoneNumber	string		Phone number of the user
6.	address	string		Physical address of the user
7.	status	string		Status of the user account
8.	passwordHash	string		Hashed and salted passwor

### 2.Tbl\_roles

Eg.Primary key: **loginid**



Eg.Foreign key: **loginid** references table **Tbl\_users\_login**

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1.	rolesId	integer	primary key	Unique identifier for the role
2.	roleName	string		Name of the role

### 3.Tbl\_login

Eg.Primary key: umsId

Eg.Foreign key: umsId references table **Tbl\_patients**

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1.	umsId	string	primary key, foreign key (users.umsId)	Alpha-numeric identifier linking to users table
2.	passwordHash	string		Hashed and salted password
3.	email	string	unique	User's email address
4.	roleId	integer	foreign key (roles.rolesId)	Role identifier
5.	status	string		Login status

### 4.Tbl\_Patients

Eg.Primary key: umsId

Eg.Foreign key: umsId references table **Tbl\_patientsdetails**

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1.	umsId	string	primary key, foreign key (users.umsId)	UMSG-UUIDv4 format patient identifier
2.	dateOfBirth	date		Patient's date of birth
3.	gender	string		Patient's gender
4.	rolesId	integer	foreign key (roles.rolesId)	Role identifier

### 5.Tbl\_Patient detail

Eg.Primary key: **patientId**

Eg.Foreign key: **patientId** references table **Tbl\_appointments**

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1.	patientId	integer	primary key, foreign key (patients.umsId)	Patient identifier
2.	medicalHistory	json		Patient's medical history in JSON format
3.	photo	string		Path to patient's photo
4.	status	string		Current status
5.	patientId	integer	primary key, foreign key (patients.umsId)	Patient identifier

## 6.Tbl\_Doctors

Eg.Primary key: **umsid**

Eg.Foreign key: **umsid** references table **Tbl\_patients**

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1.	umsId	string	primary key, foreign key (users.umsId)	UMSD-UUIDv4 format doctor identifier
2.	specialization	string		Doctor's medical specialization
3.	rolesId	integer	foreign key (roles.rolesId)	Role identifier

## 7.Tbl\_doctor details

Eg.Primary key: **doctorid**

Eg.Foreign key: **doctorid** references table **Tbl\_appointments**

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1.	doctorId	integer	primary key, foreign key (doctors.umsId)	Doctor identifier
2.	availability	string		Doctor's availability schedule
3.	medicalId	string		Alpha-numeric medical license ID

4.	qualification	json		Doctor's qualifications in JSON format
5.	photo	string		Path to doctor's photo
6.	status	string		Current status
7.	doctorId	integer	primary key, foreign key (doctors.umsId)	Doctor identifier

### 8.Tbl\_Hospitals

Eg.Primary key: **umsid**

Eg.Foreign key: **umsid** references table **Tbl\_hospital\_Details**

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1.	umsId	string	primary key, foreign key (users.umsId)	UMSH-UUIDv4 format hospital identifier
2.	hospitalName	string		Name of the hospital
3.	location	string		Hospital location
4.	rolesId	integer	foreign key (roles.rolesId)	Role identifier

### 9.Tbl\_hospital details

Eg.Primary key: **hospitalid**

Eg.Foreign key: **hospitalid** references table **Tbl\_user**

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1.	hospitalId	integer	primary key, foreign key (hospitals.umsId)	Hospital identifier
2.	hospitalType	string		Type of hospital
3.	speciality	json		Hospital specialities in JSON format
4.	affiliationNumber	string		Alpha-numeric affiliation number
5.	status	string		Current status

**10.Tbl\_appointments**

Eg.Primary key: **appointmentid**

Eg.Foreign key: **loginid** references table **Tbl\_users\_login**

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1.	appointmentId	integer	primary key	Unique identifier for appointment
2.	patientId	integer	foreign key (patients.umsId)	Patient identifier
3.	doctorId	integer	foreign key (doctors.umsId)	Doctor identifier
4.	rolesId	integer	foreign key (roles.rolesId)	Role identifier
5.	dateTime	timestamp		Appointment date and time
6.	appointmentNotes	text		Notes about the appointment
7.	appointmentSpecialization	string		Specialization required for appointment
8.	appointmentStatus	string		Status of the appointment
9.	disabilityCertificateId	string		Alpha-numeric disability certificate ID
10.	status	string		Current status

**11.Tbl\_medical records**

Eg.Primary key: **recordid**

Eg.Foreign key: **recordid** references table **Tbl\_appointment**

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1.	recordId	integer	primary key	Unique identifier for medical record
2.	patientId	integer	foreign key (patients.umsId)	Patient identifier
3.	doctorId	integer	foreign key (doctors.umsId)	Doctor identifier
4.	rolesId	integer	foreign key (roles.rolesId)	Role identifier
5.	dateOfRecord	date		Date of the medical record
6.	documentType	string		Type of medical document
7.	documentContent	json		Medical record content in JSON format

8.	hospitalDb	string		Hospital database reference
9.	status	string		Current status

## 12.Tbl\_symptoms

Eg.Primary key: **symptomid**

Eg.Foreign key: **symptomid** references table **Tbl\_appointment**

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1.	symptomId	integer	primary key	Unique identifier for symptom record
2.	patientId	integer	foreign key (patients.umsId)	Patient identifier
3.	rolesId	integer	foreign key (roles.rolesId)	Role identifier
4.	date	date		Date of symptom report
5.	reportedSymptoms	string	increment	List of reported symptoms
6.	status	string		Current status

## 13.Tbl\_disease outbreaks

Eg.Primary key: **outbreakId**

Eg.Foreign key: **outbreakId** references table **Tbl\_medical records**

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1.	outbreakId	integer	primary key	Unique identifier for disease outbreak
2.	diseaseName	string		Name of the disease
3.	location	geojson		Geographic location data
4.	rolesId	integer	foreign key (roles.rolesId)	Role identifier
5.	startDate	date		Outbreak start date
6.	latestUpdate	date		Date of latest update
7.	caseCount	integer		Number of cases
8.	status	string		Current status

## **CHAPTER 5**

### **SYSTEM TESTING**

## 5.1 INTRODUCTION

Software Testing is the software implementation process in a controlled way to answer the question - does the software behave as mentioned? Software testing is often used in conjunction with verification and validation of the term. Validation is inspection or testing of items, includes software for compliance and consistency with associated specification. Software testing is just one type of verification that also uses techniques such as reviews, analysis, inspection and passages. Validation is a process of control that what was specified is what the user wanted. Other activities that are often associated with software testing are static analysis and dynamic analysis. Static analysis examines the source code of the software, looks for problems and collects metrics without code. Dynamic analysis focuses on software behavior, providing information such as traces of implementation, timing profiles and test coverage information. Testing is a set of activity that can be planned in advanced and systematically performed. Testing starts at the module level and works on the integration of the entire computer -based system. Without testing, there is nothing complete, because it is a vital success of the system testing goals, there are several rules that can serve as testing goals. They are:

- Testing is the process of implementing the program with the intention of finding an error.
- A good test case is a case that has a high opportunity to find an undiscovered error.
- A successful test is a test that reveals an undiscovered error If testing is successfully performed according to the above objectives, it would reveal the software errors.
- Testing also shows that the software function seems to work according to the specification, the performance requirement seems to have been met.

## 5.2 TEST PLAN

The test plan means a number of required procedures that need to be followed when performing various test methods. The test plan works as a blue print for the event to be followed. Software engineers create a computer program, its documentation and related data structures. Software developers are always responsible for testing individual units of programs and ensure that everyone performs the function for which it was designed. There is an independent test group (ITG) to eliminate natural problems associated with the builder to try what was created. Specific testing objectives should be given in a measurable way. In order for the average failure time, the cost of finding and repairing defects, the remaining density of the defect or frequency of occurrence, and working hours testing for the regression test should be listed within the test plan.

The levels of testing include:

- ❖ Unit testing
- ❖ Integration Testing
- ❖ Data validation Testing
- ❖ Output Testing

### **5.2.1 Unit Testing**

Testing units focuses on verification efforts on the smallest software design unit - software component or module. Using the component level design as a wizard, important control paths are tested to detect errors in the module limit. Before starting any other test, data flow tests across the module interface are required. If the data do not enter and leave correctly, all other tests are deepened. Selective trial testing is a necessary task during the unit test. Good design dictates that error conditions and set paths of handling errors set to redirect or clean processing are expected when an error occurs. Border testing is the last task of the step testing step. Software often fails on its borders.

### **5.2.2 Integration Testing**

Integration testing is a systematic technique for the construction of the program structure and at the same time perform tests to detect errors associated with the connection. The aim is to take the components tested to the unit and create a program structure that was dictated by the proposal. The whole program is tested as a whole. Correction is difficult because the insulation of causes is complicated by the huge area of the entire program. Once these errors are fixed, a new one appears and the process continues to see the seemingly endless loop. After testing the units in the system, all modules were integrated for testing any inconsistency in interfaces. In addition, the differences in program structures were removed and the unique structure of the program has developed.

### **5.2.3 Validation Testing or System Testing**

This is the last step in testing. In this, the whole system was tested with all forms, code, modules and class modules. This form of testing is known as a black box test or system tests. The black box test method focuses on functional software requirements. This means that the Black Box testing allows you to deduce a set of input conditions that fully perform all functional requirements for the program.

### **5.2.3 Output Testing or User Acceptance Testing**

The system under consideration is tested to receive the user; Here it should satisfy the need for the company. Software should be in contact with a perspective system; The user at the time of development and making changes whenever necessary. This happened with respect to the following points:



- Input screens Designs
- Output screen designs

The above testing is performed with the reception of different types of test data. The preparation of test data plays an essential role in system testing. After testing the test data, the monitored system is tested using these test data.

#### **5.2.4 Automation Testing**

The Suite test case is performed using specialized automated test software tools as part of the software testing technique known as automation testing. The test phases are carefully performed by human performing manual testing while sitting in front of the computer. In addition, automation testing software can generate thorough test messages, compare the expected and actual findings, and enter test data into the test system. The automation of the software test requires significant financial and material inputs.

#### **5.2.5 Selenium Testing**

Selenium is a free and open source tool for testing web applications in multiple browsers and operating systems. Selenium test scripts can be written in various programming languages, including Java, C#, JavaScript, Python, etc. Automation performed by selenium frame is referred to as selenium automation testing.

#### **Example:**

##### **Test Case 1**

##### **Code**

```
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.common.exceptions import TimeoutException, ElementClickInterceptedException,
NoSuchElementException
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
import requests
from selenium.common.exceptions import WebDriverException
import time
def setup_driver():
    service = Service(ChromeDriverManager().install())
    driver = webdriver.Chrome(service=service)
```

```
    return driver

def wait_for_overlay_to_disappear(driver, timeout=10):
    try:
        WebDriverWait(driver, timeout).until_not(
            EC.presence_of_element_located((By.CSS_SELECTOR, ".overlay, .modal, .loading"))
        )
    except TimeoutException:
        print("No overlay found or it didn't disappear")

def click_element_safely(driver, element):
    try:
        element.click()
    except ElementClickInterceptedException:
        wait_for_overlay_to_disappear(driver)
        try:
            element.click()
        except ElementClickInterceptedException:
            driver.execute_script("arguments[0].click();", element)

def test_successful_login(driver):
    driver.get("http://127.0.0.1:5000/auth/login")
    # Wait for the login form to be loaded
    try:
        WebDriverWait(driver, 10).until(
            EC.presence_of_element_located((By.ID, "loginForm"))
        )
        print("Login form found")
    except TimeoutException:
        print("Login form not found within timeout")
        raise
    # Fill in credentials
    try:
        email_field = WebDriverWait(driver, 10).until(
            EC.presence_of_element_located((By.NAME, "identifier"))
        )
        email_field.send_keys("janepat@gmail.com")
```

```
print("Email entered")
password_field = WebDriverWait(driver, 10).until(
    EC.presence_of_element_located((By.NAME, "password"))
)
password_field.send_keys("PPpp!@12")
print("Password entered")
login_button = WebDriverWait(driver, 10).until(
    EC.element_to_be_clickable((By.XPATH, "//button[@type='submit']"))
)
click_element_safely(driver, login_button)
print("Login button clicked")
# Check for error messages
try:
    error_message = driver.find_element(By.CLASS_NAME, "alert")
    print(f"Login error: {error_message.text}")
    return
except NoSuchElementException:
    pass
# Wait for URL change with longer timeout
WebDriverWait(driver, 20).until(
    lambda driver: driver.current_url == "http://127.0.0.1:5000/patient/dashboard"
)
print(f"URL changed to: {driver.current_url}")
# Verify exact URL match
assert driver.current_url == "http://127.0.0.1:5000/patient/dashboard", \
    f"Unexpected URL after login: {driver.current_url}"
print("Successfully redirected to patient dashboard")
except Exception as e:
    print(f"Test failed: {str(e)}")
    print(f"Current URL: {driver.current_url}")
    # Take screenshot on failure
    driver.save_screenshot("login_error.png")
    raise
```

```

def check_server_running():
    try:
        response = requests.get("http://127.0.0.1:8000/")
        return response.status_code == 200
    except requests.ConnectionError:
        return False

def main():
    driver = setup_driver()
    try:
        test_successful_login(driver)
    finally:
        driver.quit()
if __name__ == "__main__":
    main()

```

### Eg.Screenshot

```

(venv) PS D:\dev\ed\masters in computer application\project\unified-medical-system-main\unified-medical-system\tests\test> python .\logintest.py
DevTools listening on ws://127.0.0.1:59157/devtools/browser/f2c152d3-936a-4121-bf5d-1218b79dfc72
=====
RESTART: Login Test
*****Login Test*****
*****
Login form found
Email entered
Password entered
Login button clicked
URL changed to: http://127.0.0.1:5908/patient/dashboard
Successfully redirected to patient dashboard
Login Test successful
(venv) PS D:\dev\ed\masters in computer application\project\unified-medical-system-main\unified-medical-system\tests\test>

```

### Eg.Test Report

Test Case 1	
Project Name: Unified Medical System	
Login Test Case	
Test Case ID: Test_1	Test Designed By: Devadethan R
Test Priority(Low/Medium/High):	Test Designed Date: 05/04/2025
Module Name: Login Page	Test Executed By : Jetty Benjamin
Test Title : Login page of user account	Test Execution Date: 06/04/2025
Description: Login page to user	
Pre-Condition :User has valid username and password	

Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/Fail)
1	Navigate to the login page	http://127.0.0.1:5000/auth/login	Login page should be displayed	Login page was loaded	Pass
2	Enter email	"janepat@gmail.com"	The email field should be filled with "janepat@gmail.com"	email entered	Pass
3	Enter Password	"PPpp!@12"	The password field should be filled with "PPpp!@12"	Password entered	Pass
4	Click on the submit button	N/A	If login is successful, user should be redirected to user login page	User was redirected to user login page	Pass
<b>Post-Condition: User should be logged in and have access to patient dashboard</b>					

**Test Case 2:****Code**

```

import os
import sys
from playwright.sync_api import sync_playwright
import time
from datetime import datetime
import colorama
from colorama import Fore, Back, Style
# Initialize colorama for colored terminal output
colorama.init(autoreset=True)

# Step 1: Navigate to login page
print_step(1, "Navigating to login page")
page.goto(f"{BASE_URL}/auth/login")
page.wait_for_load_state("domcontentloaded")
page.screenshot(path=os.path.join(SCREENSHOTS_DIR,

```

```
"1_login_page.png"))
    print_info(f"Screenshot saved: 1_login_page.png")
    # Step 2: Login as doctor
    print_step(2, "Logging in as doctor")
    print_info(f"Using email: {DOCTOR_EMAIL}")
    page.fill("input[name='identifier']", DOCTOR_EMAIL)
    page.fill("input[name='password']", DOCTOR_PASSWORD)
    page.screenshot(path=os.path.join(SCREENSHOTS_DIR,
"2_login_form_filled.png"))
    print_info(f"Screenshot saved: 2_login_form_filled.png")
    # Submit login form
    print_info("Submitting login form...")
    page.click("button[type='submit']")
    page.wait_for_load_state("networkidle")
    # Check if login was successful
    if "/doctor/" not in page.url:
        print_error("Login failed. Please check credentials.")
        page.screenshot(path=os.path.join(SCREENSHOTS_DIR,
"login_failed.png"))
        print_info(f"Screenshot saved: login_failed.png")
        return False
    print_success("Login successful")
    page.screenshot(path=os.path.join(SCREENSHOTS_DIR,
"3_login_success.png"))
    print_info(f"Screenshot saved: 3_login_success.png")
    # Step 3: Navigate to edit profile page
    print_step(3, "Navigating to edit profile page")
    page.goto(f"{BASE_URL}/doctor/edit_profile")
    page.wait_for_load_state("domcontentloaded")
    # Step 4: Test updating doctor's name
    print_step(4, "Updating doctor's name")
    # Make sure we're on the Personal Information tab
    print_info("Selecting Personal Information tab...")
    page.click("a[href='#personal-information']")
```

```
page.wait_for_selector("#personal-information.active.show")
# Get initial name value for comparison
name_input = page.locator("input#name")
initial_name = name_input.input_value() if name_input.is_visible() else "Not
visible"

print_info(f"Initial name: {initial_name}")
# Update name
if name_input.is_visible():
    name_input.fill("")
    name_input.fill(TEST_NAME)
    print_success(f"Updated name to: {TEST_NAME}")
    page.screenshot(path=os.path.join(SCREENSHOTS_DIR,
"5_name_updated.png"))
    print_info(f"Screenshot saved: 5_name_updated.png")
else:
    print_error("Name field not found")
    page.screenshot(path=os.path.join(SCREENSHOTS_DIR,
"name_field_not_found.png"))
    print_info(f"Screenshot saved: name_field_not_found.png")
    return False
# Check if name was updated
page_content = page.content()
if TEST_NAME in page_content:
    print_success(f"Name verification: {TEST_NAME} found on profile page")
else:
    print_error(f"Name verification failed: {TEST_NAME} not found")
    page.screenshot(path=os.path.join(SCREENSHOTS_DIR,
"name_verification_failed.png"))
    print_info(f"Screenshot saved: name_verification_failed.png")
    return False

def generate_html_report(test_result):
    """Generate a simple HTML test report"""
    timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    report_path = os.path.join(RESULTS_DIR, "doctor_profile_edit_report.html")
```

```

print_info("Generating HTML test report...")

# Get list of screenshots
screenshots = sorted([f for f in os.listdir(SCREENSHOTS_DIR) if
f.endswith('.png')])

```

## Screenshot

```

C:\Users\devadethan> python .\tests\test1.py

= DOCTOR PROFILE EDIT TEST AUTOMATION
INFO: Test execution started at: 2025-04-06 02:51:38
INFO: Launching browser...

= DOCTOR PROFILE EDIT TEST
INFO: Test started at: 2025-04-06 02:51:40
INFO: Testing URL: http://127.0.0.1:5000/doctor/edit_profile

[STEP 1] Navigating to login page
INFO: Screenshot saved: 1_login_page.png

[STEP 2] Logging in as doctor
INFO: Using email: doctor@gmail.com
INFO: Screenshot saved: 2_login_form_filled.png
INFO: Submitting login form...
SUCCESS: Login successful
INFO: Screenshot saved: 3_login_success.png

[STEP 3] Navigating to edit profile page
SUCCESS: Successfully navigated to edit profile page
INFO: Screenshot saved: 4_edit_profile_page.png

[STEP 4] Updating doctor's name
INFO: Selecting Personal Information tab...
INFO: Initial name: Catherine Rurue
SUCCESS: Updated name to: Cathy Test
INFO: Screenshot saved: 5_name_updated.png

[STEP 5] Submitting form
SUCCESS: Form submitted

[STEP 6] Verifying submission result
WARNING: No success message found

[STEP 7] Verifying changes on profile page
INFO: Screenshot saved: 7_profile_page.png
SUCCESS: Name verification: Cathy Test Found on profile page

[STEP 8] Logging out
SUCCESS: Successfully logged out
INFO: Screenshot saved: 8_logout.png

= TEST COMPLETED SUCCESSFULLY!
INFO: Test finished at: 2025-04-06 02:51:40
INFO: Closing browser...
INFO: Generating HTML test report...
SUCCESS: HTML report generated at: C:\Users\devadethan\project\unified-medical-system-main\unified-medical-system\tests\test2_results\doctor_profile_edit_report.html
INFO: To view the report, open the file in a web browser.
SUCCESS: Test execution completed successfully
INFO: Test execution finished at: 2025-04-06 02:51:40

```

## Test report

Test Case 2					
Project Name: Unified Medical System					
Doctor Profile Edit Test Case					
Test Case ID: Test_2			Test Designed By: Devadethan R		
Test Priority(Low/Medium/High):			Test Designed Date: 05/04/2025		
Module Name: Doctor Profile Edit			Test Executed By : Jetty Benjamin		
Test Title : Login to doctor Profile Edit			Test Execution Date: 06/04/2025		
Description: Doctor Profile Edit					
Pre-Condition :Doctor has valid profile					
Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/Fail)
1	Navigate to the login page	http://127.0.0.1:5000	Login page should be displayed	Login page was loaded	Pass



2	Enter email	“doctor1 @gmail.com”	The email field should be filled with “doctor1 @gmail.com”	email entered	Pass
3	Enter Password	“DDdd!@12”	The password field should be filled with “DDdd!@12”	Password entered	Pass
4	Click on the submit button	N/A	If login is successful, user should be redirected to user login page	User was redirected to user login page	Pass
5.	Navigate to edit profile page	N/A	Login successful and navigate to user edit page	User was redirected to user edit page	Pass
6.	Update name	“Cathy Munroe”	User name edit form	User name edited successfully	Pass
7.	Submit the form	N/A	Form submitted	Name edit successfully and form submitted	Pass
<b>Post-Condition: User should be logged in and have edited doctor name</b>					

**Test Case 3:****Code**

```

from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.support.ui import Select
from selenium.common.exceptions import TimeoutException,
ElementClickInterceptedException, NoSuchElementException
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
import requests
import time
import random
import string

def setup_driver():
    service = Service(ChromeDriverManager().install())
    driver = webdriver.Chrome(service=service)
    return driver

```

```
def wait_for_overlay_to_disappear(driver, timeout=10):
    try:
        WebDriverWait(driver, timeout).until_not(
            EC.presence_of_element_located((By.CSS_SELECTOR, ".overlay, .modal,
.loading")))
    )
    except TimeoutException:
        print("No overlay found or it didn't disappear")

def click_element_safely(driver, element):
    try:
        element.click()
    except ElementClickInterceptedException:
        wait_for_overlay_to_disappear(driver)
        try:
            element.click()
        except ElementClickInterceptedException:
            driver.execute_script("arguments[0].click();", element)
def print_test_success():
    print("Registration Test successful!")

def test_successful_registration(driver):
    print_test_header()
    driver.get("http://127.0.0.1:5000/patient/register")

    # Wait for the registration form to be loaded
    try:
        WebDriverWait(driver, 10).until(
            EC.presence_of_element_located((By.ID, "registerForm")))
        )
        print("Registration form found")
    except TimeoutException:
        print("Registration form not found within timeout")
        raise

    try:
        # Fill in registration details
        name_field = WebDriverWait(driver, 10).until(
            EC.presence_of_element_located((By.NAME, "name")))
        )
        name_field.send_keys("Test User")
        print("Name entered")

        email_field = WebDriverWait(driver, 10).until(
            EC.presence_of_element_located((By.NAME, "email")))
        )
        email_field.send_keys(generate_random_email())
        print("Email entered")

        # Select state
```

```
state_select = Select(WebDriverWait(driver, 10).until(
    EC.presence_of_element_located((By.NAME, "state")))
))
state_select.select_by_value("KA") # Selecting Karnataka as an example
print("State selected")

phone_field = WebDriverWait(driver, 10).until(
    EC.presence_of_element_located((By.NAME, "phonenummer")))
)
phone_field.send_keys("9876543210")
print("Phone number entered")

password_field = WebDriverWait(driver, 10).until(
    EC.presence_of_element_located((By.NAME, "password")))
)
password_field.send_keys("Test@123")
print("Password entered")

confirm_password_field = WebDriverWait(driver, 10).until(
    EC.presence_of_element_located((By.NAME, "confirm_password")))
)
confirm_password_field.send_keys("Test@123")
print("Confirm password entered")

# Submit the form
register_button = WebDriverWait(driver, 10).until(
    EC.element_to_be_clickable((By.XPATH, "//button[@type='submit']")))
)
click_element_safely(driver, register_button)
print("Register button clicked")
# Wait for redirect to login page
WebDriverWait(driver, 20).until(
    lambda driver: driver.current_url == "http://127.0.0.1:5000/auth/login"
)
print(f"URL changed to: {driver.current_url}")

# Verify exact URL match
assert driver.current_url == "http://127.0.0.1:5000/auth/login", \
    f"Unexpected URL after registration: {driver.current_url}"
print("Successfully redirected to login page")
def check_server_running():
    try:
        response = requests.get("http://127.0.0.1:8000/")
        return response.status_code == 200
    except requests.ConnectionError:
        return False

def main():
    driver = setup_driver()
    try:
        test_successful_registration(driver)
```

```

finally:
    driver.quit()

if __name__ == "__main__":
    main()

```

## Screenshot

```

(venv) PS D:\dev\ed\masters in computer application\project\unified-medical-system-main\unified-medical-system\tests\test> python .\registertest.py
DevTools listening on ws://127.0.0.1:59385/devtools/browser/1c68f0d3-4aae-4e59-9c62-bafa9a849ac1
=====
RESTART: Registration Test
*****Place TEST*****
*****
Registration form found
Name entered
Email entered
State selected
Phone number entered
Password entered
Confirm password entered
Register button clicked
● URL changed to: http://127.0.0.1:5000/auth/login
Successfully redirected to login page
Registration Test successful!
[816:21560:1104/160135.719:ERROR:ssl_client_socket_impl.cc(878)] handshake failed; returned -1, SSL error code 1, net_error -101
[816:21560:1104/160135.772:ERROR:ssl_client_socket_impl.cc(878)] handshake failed; returned -1, SSL error code 1, net_error -101
Created TensorFlow Lite XNNPACK delegate for CPU.
Success message not found
(venv) PS D:\dev\ed\masters in computer application\project\unified-medical-system-main\unified-medical-system\tests\test>

```

## Test Report:

Test Case 3					
Project Name: Unified Medical System					
Admin Analytics Test Case					
Test Case ID: Test_3			Test Designed By: Devadethan R		
Test Priority(Low/Medium/High):			Test Designed Date: 03/04/2025		
Module Name: Admin Analytics			Test Executed By : Jetty Benjamin		
Test Title : Admin Analytics assistant			Test Execution Date: 04/04/2025		
Description: Admin Analytics					
Pre-Condition: Admin analytics assistant present					
Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/Fail)
1	Navigate to registration page	http://127.0.0.1:5000/patient/register	Registration page should be displayed	Registration page loaded	Pass
2	Enter full name	"Test User"	Name field should accept input	Name entered successfully	Pass
3	Enter email	[Random generated email]	Email field should accept input	Email entered successfully	Pass

4	Select state	"KA" (Karnataka)	State should be selectable	State selected successfully	Pass
5	Enter phone number	"9876543210"	Phone field should accept input	Phone number entered	Pass
6	Enter password	"Test@123"	Password field should accept input	Password entered	Pass
7	Confirm password	"Test@123"	Confirm password should match	Password confirmed	Pass
8	Submit registration	N/A	Should redirect to login page	Redirected to login page	Pass
<b>Post-Condition: Admin Analytics should be created and ready for assistant</b>					

**Test Case 4:****Code**

```

from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.common.exceptions import TimeoutException, ElementClickInterceptedException,
NoSuchElementException
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
import requests
import time
import os

def setup_driver():
    service = Service(ChromeDriverManager().install())
    driver = webdriver.Chrome(service=service)
    return driver

def wait_for_overlay_to_disappear(driver, timeout=10):
    try:
        WebDriverWait(driver, timeout).until_not(
            EC.presence_of_element_located((By.CSS_SELECTOR, ".overlay, .modal, .loading"))
        )
    except TimeoutException:
        print("No overlay found or it didn't disappear")

def print_test_header():
    print("=" * 50)

```

```
print("RESTART: Medical Record Download Test")
print("*" * 20 + "Place TEST" + "*" * 20)
print("*" * 50)

def print_test_success():
    print("*" * 50)
    print("Medical Record Downloaded Successfully!")
    print("*" * 50)

def login(driver):
    driver.get("http://127.0.0.1:5000/auth/login")

    try:
        # Wait for login form
        WebDriverWait(driver, 10).until(
            EC.presence_of_element_located((By.ID, "loginForm"))
        )
        print("Login form found")

        # Fill in credentials
        email_field = WebDriverWait(driver, 10).until(
            EC.presence_of_element_located((By.NAME, "identifier"))
        )
        email_field.send_keys("janepat@gmail.com")
        print("Email entered")

        password_field = WebDriverWait(driver, 10).until(
            EC.presence_of_element_located((By.NAME, "password"))
        )
        password_field.send_keys("password")
        print("Password entered")

        # Click login button
        login_button = WebDriverWait(driver, 10).until(
            EC.element_to_be_clickable((By.XPATH, "//button[@type='submit']"))
        )
        click_element_safely(driver, login_button)
        print("Login button clicked")

def test_medical_record_download(driver):
    print_test_header()

    # First login
    if not login(driver):
        raise Exception("Login failed")

    try:
        # Navigate to medical records page
        driver.get("http://127.0.0.1:5000/patient/medical_records")
        print("Navigated to medical records page")
```

```

# Wait for medical records to load
time.sleep(5) # Allow time for records to load

# Find and click the first download button
download_button = WebDriverWait(driver, 10).until(
    EC.presence_of_element_located((By.CSS_SELECTOR, ".btn-download"))
)
click_element_safely(driver, download_button)
print("Clicked download button")

# Wait for download to complete
time.sleep(5)
def check_server_running():
    try:
        response = requests.get("http://127.0.0.1:5000/")
        return response.status_code == 200
    except requests.ConnectionError:
        return False

def main():
    driver = setup_driver()
    try:
        test_medical_record_download(driver)
    finally:
        driver.quit()

if __name__ == "__main__":
    main()

```

## Screenshot

```

(venv) PS D:\dev\ed\masters in computer application\project\unified-medical-system-main\unified-medical-system\tests\test> python .\download.py
DevTools listening on ws://127.0.0.1:59889/devtools/browser/8cb23153-c39a-4752-8d77-6d284cc2f8b5
=====
RESTART: Medical Record Download Test
*****Place TEST*****
*****
Login form found
Email entered
Password entered
Login button clicked
Successfully logged in and redirected to dashboard
Navigated to medical records page
[22200:14980:1104/160917.962:ERROR:ssl_client_socket_impl.cc(878)] handshake failed; returned -1, SSL error code 1, net_error -101
[22200:14980:1104/160917.974:ERROR:ssl_client_socket_impl.cc(878)] handshake failed; returned -1, SSL error code 1, net_error -101
Clicked download button
Created TensorFlow Lite XNNPACK delegate for CPU.
PDF file downloaded successfully
*****
Medical Record Downloaded Successfully!
*****
(venv) PS D:\dev\ed\masters in computer application\project\unified-medical-system-main\unified-medical-system\tests\test>

```

## Test report

### Test Case 4

**Project Name: Unified Medical System**

### Medical Record Download Test Case

**Test Case ID: Test\_4**

**Test Designed By: Devadethan R**

<b>Test Priority(Low/Medium/High):</b>			<b>Test Designed Date: 04/04/2025</b>		
<b>Module Name:</b> Medical Record Download			<b>Test Executed By : Jetty Benjamin</b>		
<b>Test Title: Medical Record Download</b>			<b>Test Execution Date:06/04/2025</b>		
<b>Description: Medical Record Download</b>					
<b>Pre-Condition:</b> User is registered and has medical records					
Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/Fail)
1	Login to system	“janepat@gmail.com/PPpp!@12”	Login should be successful	Login successful	Pass
2	Navigate to medical records	http://127.0.0.1:5000/patient/medical_records	Medical records page should load	Page loaded successfully	Pass
3	Locate download button	N/A	Download button should be visible	Button found	Pass
4	Click download button	N/A	Download should initiate	Download started	Pass
5	Verify download	N/A	PDF file should be in downloads folder	File downloaded successfully	Pass
<b>Post-Condition: Medical record PDF should be available in user's downloads folder</b>					

**Test Case 5:****Code**

```

import os
import sys
from playwright.sync_api import sync_playwright
import time
from datetime import datetime
import colorama
from colorama import Fore, Back, Style
def test_medical_insights_chatbot():
    """Test the medical insights chatbot functionality"""
    with sync_playwright() as p:
        # Launch browser
        print_info("Launching browser...")

```



```
browser = p.chromium.launch(headless=False)
context = browser.new_context(viewport={"width": 1280, "height": 720})
page = context.new_page()

try:
    print_header("MEDICAL INSIGHTS CHATBOT TEST")
    print_info(f"Test started at: {datetime.now().strftime('%Y-%m-%d %H:%M:%S')}")
    print_info(f"Testing URL: {BASE_URL}/patient/medical_insights")

    # Step 1: Navigate to login page
    print_step(1, "Navigating to login page")
    page.goto(f"{BASE_URL}/auth/login")
    page.wait_for_load_state("domcontentloaded")
    page.screenshot(path=os.path.join(SCREENSHOTS_DIR, "1_login_page.png"))
    print_info(f"Screenshot saved: 1_login_page.png")

    # Step 2: Login as patient
    print_step(2, "Logging in as patient")
    print_info(f"Using email: {PATIENT_EMAIL}")
    page.fill("input[name='identifier']", PATIENT_EMAIL)
    page.fill("input[name='password']", PATIENT_PASSWORD)
    page.screenshot(path=os.path.join(SCREENSHOTS_DIR, "2_login_form_filled.png"))
    print_info(f"Screenshot saved: 2_login_form_filled.png")

    # Submit login form
    print_info("Submitting login form...")
    page.click("button[type='submit']")
    page.wait_for_load_state("networkidle")

    # Step 3: Navigate to medical insights page
    print_step(3, "Navigating to medical insights page")
    page.goto(f"{BASE_URL}/patient/medical_insights")
    page.wait_for_load_state("domcontentloaded")

    # Step 4: Ask first question to the chatbot
    print_step(4, f"Asking first question: '{FIRST_QUESTION}'")

    # Type and send first question
    print_info("Typing first question...")
    page.fill("#queryInput", FIRST_QUESTION)
    page.screenshot(path=os.path.join(SCREENSHOTS_DIR, "5_first_question_typed.png"))
    print_info(f"Screenshot saved: 5_first_question_typed.png")

    # Send the question
    print_info("Sending first question...")
    page.click(".send-button")

    # Wait for response (loading animation to appear and then disappear)
    print_info("Waiting for response...")
    page.wait_for_selector(".loading-animation", state="attached")
    page.wait_for_selector(".loading-animation", state="detached", timeout=30000)
```

```

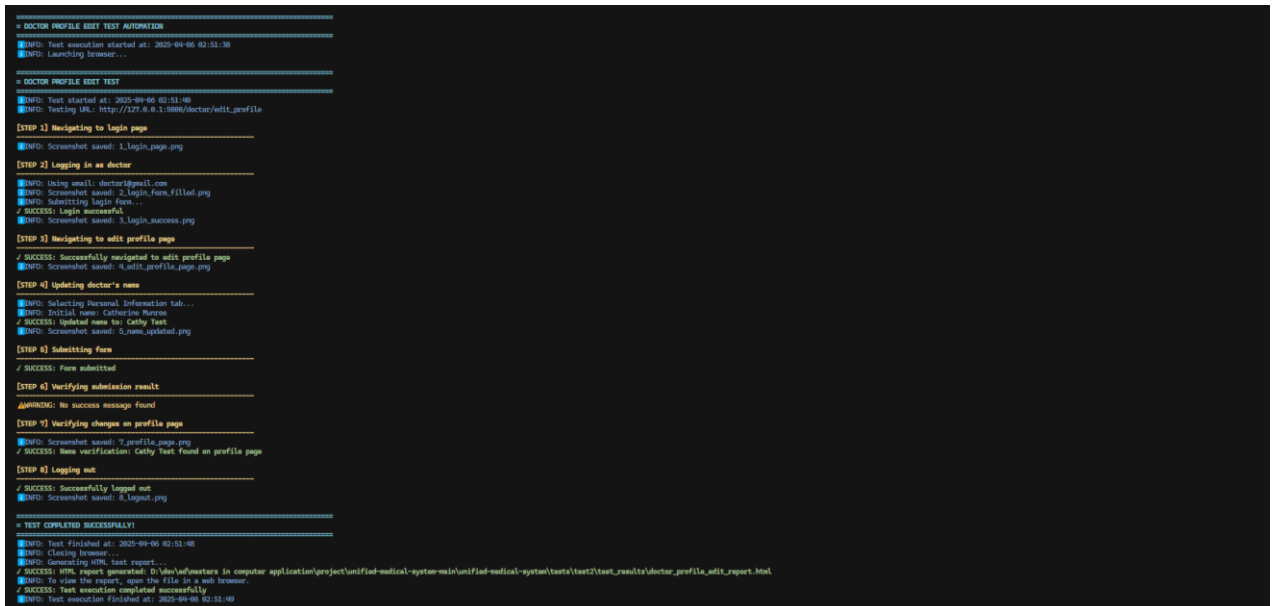
# Check if we got a response
bot_message = page.locator(".bot-message").last
if bot_message.is_visible():
    response_text = bot_message.text_content()
    print_success(f"Received response to first question: '{response_text[:50]}...'")
    page.screenshot(path=os.path.join(SCREENSHOTS_DIR, "6_first_response.png"))
    print_info(f"Screenshot saved: 6_first_response.png")
else:
    print_error("No response received for first question")
    page.screenshot(path=os.path.join(SCREENSHOTS_DIR,
"first_question_no_response.png"))
    print_info(f"Screenshot saved: first_question_no_response.png")
    return False

def generate_html_report(test_result):
    """Generate a simple HTML test report"""
    timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    report_path = os.path.join(RESULTS_DIR, "medical_insights_chatbot_report.html")

    print_info("Generating HTML test report...")

```

## Screenshot



## Test report

Test Case 5	
Project Name: Unified Medical System	
Patient Medical Chatbot Test Case	
Test Case ID: Test_5	Test Designed By: Devadethan R
Test Priority(Low/Medium/High):	Test Designed Date: 04/04/2025
Module Name: Patient Medical Chatbot	Test Executed By : Jetty Benjamin

Test Title: Patient Medical Chatbot			Test Execution Date:06/04/2025		
Description: Medical chatbot					
Pre-Condition: Patient can access medical chatbot					
Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/Fail)
1	Login to system	“janepat@gmail.com/PPpp!@12”	Login should be successful	Login successful	Pass
2	Navigate to medical chatbot	http://127.0.0.1:5000/medical chatbot	Chatbot loaded	Bot loaded successfully	Pass
3	Ask questions	N/A	Loading answers	Answers loaded	Pass
4	Capture final state of the conversation	N/A	Chatbot gave answers	Answers and solutions loaded	Pass
5	Logout	N/A	User logout	Logout successfully	Pass
Post-Condition: Medical chatbot successfully open to patients					

**Test Case 6:****Code**

```

from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.common.exceptions import TimeoutException
import unittest
import time

```

```

def print_test_header():
    print("=" * 50)
    print("Outbreak Test Started")
    print("=" * 50)

```

```

def print_test_success():
    print("=" * 50)
    print("Outbreak Map Test Completed Successfully!")
    print("Map was loaded and zoomed correctly")
    print("=" * 50)

```

```

class TestOutbreakMap(unittest.TestCase):
    def setUp(self):
        print_test_header()

```

```

self.driver = webdriver.Chrome()
self.driver.maximize_window()
# Navigate to login page
self.driver.get("http://127.0.0.1:5000/auth/login")
print("Navigated to login page")
try:
    # Wait for login form
    WebDriverWait(self.driver, 10).until(
        EC.presence_of_element_located((By.ID, "loginForm"))
    )
    print("Login form found")
    email_field = self.driver.find_element(By.NAME, "identifier")
    password_field = self.driver.find_element(By.NAME, "password")

    email_field.send_keys("d.dethanr@gmail.com")
    password_field.send_keys("AAaa!@12")
    print("Credentials entered")

    login_button = self.driver.find_element(By.XPATH,
        "//button[@type='submit']")
    login_button.click()
    print("Login button clicked")
    # Execute JavaScript to zoom in multiple times
    self.driver.execute_script("""
        var map
        =
document.querySelector('#map_ae515ffac4699a595b9141b0f72c6e09');
        var leafletMap = map._leaflet_map;
        if (leafletMap) {
            // Store initial zoom level
            var initialZoom = leafletMap.getZoom();
            console.log('Initial zoom level:', initialZoom);

            // Zoom in 3 times with delay
            setTimeout(function() {
                leafletMap.setZoom(initialZoom + 2);
                console.log('First zoom completed');

                setTimeout(function() {
                    leafletMap.setZoom(initialZoom + 4);
                    console.log('Second zoom completed');

                    setTimeout(function() {
                        leafletMap.setZoom(initialZoom + 6);
                        console.log('Final zoom completed');
                    }, 1000);
                }, 1000);
            }, 1000);
        }
    """)
    print("Executing zoom operations...")

```

```

        # Wait for zooming to complete
        time.sleep(4)
    def tearDown(self):
        if self.driver:
            self.driver.quit()
            print("Browser closed")

if __name__ == "__main__":
    unittest.main()

```

## Screenshot

```

(venv) PS D:\dev\ed\masters in computer application\project\unified-medical-system-main\unified-medical-system\tests\test> python .\openoutbreakmap.py
Outbreak Test Started
=====
DevTools listening on ws://127.0.0.1:61565/devtools/browser/333c6df6-87cf-47a7-9a69-c234cdcef346
Navigated to login page
Login form found
Credentials entered
Login button clicked
Successfully logged in and redirected to dashboard
Clicked on outbreak map link
Map loaded successfully
Executing zoom operations...
Map is still visible after zooming
URL verification successful
=====
Outbreak Map Test Completed Successfully!
Map was loaded and zoomed correctly
=====
Browser closed
.
=====
Ran 1 test in 15.359s
OK
(venv) PS D:\dev\ed\masters in computer application\project\unified-medical-system-main\unified-medical-system\tests\test>

```

## Test Report:

Test Case 6					
Project Name: Unified Medical System					
Outbreak Map Test Case					
Test Case ID: Test_6			Test Designed By: Devadethan R		
Test Priority(Low/Medium/High):			Test Designed Date: 05/04/2025		
Module Name: Admin Dashboard			Test Executed By : Jetty Benjamin		
Test Title : Outbreak Map Functionality			Test Execution Date: 06/04/2025		
Description: Verify outbreak map loading and interaction					
Pre-Condition: Admin account exists and has necessary permissions					
Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/Fail)
1	Login as admin	“d.dethanr@gmail.com/A Aaa!@12”	Admin login should succeed	Login successful	Pass

2	Navigate to outbreak map	Click outbreak map link	Map page should load	Page loaded successfully	Pass
3	Verify map loading	N/A	Map should be visible	Map displayed correctly	Pass
4	Test zoom functionality	Multiple zoom levels	Map should zoom properly	Zoom operations successful	Pass
5	Verify map visibility	N/A	Map should remain visible after zoom	Map remained visible	Pass
<b>Post-Condition: Admin should be able to view and interact with outbreak map</b>					

## **CHAPTER 6**

## **IMPLEMENTATION**

## 6.1 INTRODUCTION

Implementation is a project phase where theoretical design turns into a working system. It can be the most important phase in achieving a successful new system that gains users' confidence that the new system will work and will be efficient and accurate. It mainly deals with the training and documentation of users. Conversion usually takes place at about the same time when the user is trained or later. Implementation simply means convening a new system design into operation, a process of converting a new revised system design to an operating system. At this stage, the main working load, the greatest shocks and the main impact on the existing system move to the user department. If the implementation is not carefully planned or controlled, it can create chaos and confusion.

The system can only be implemented after testing and if it is found to work according to specifications. System employees check system feasibility. The implementation includes all those activities that are taken from the existing system to a new system. The new system can be a brand new, replacing an existing manual or automated system or can be edited by an existing system. The more complicated the implemented system is, the more the system analysis and design efforts needed to implement the three main aspects: education and training, testing and transition in the system. The state of implementation includes the following tasks:

- Careful planning
- System investigation and restriction
- Design of methods to achieve transition.

## 6.2 IMPLEMENTATION PROCEDURES

The implementation of the software refers to the final installation of the package in its real environment, to the satisfaction of the intended use and operation of the system. In many organizations someone who will not operate it will introduce a software development project. In the initial phase, people doubt the software but we have to ensure that the resistance does not create because it is necessary to make sure that:

- The active user must be aware of the benefits of using the new system. Their confidence in the software is created.
- The correct instructions are provided to the user to be convenient to use the application.

Before we continue and display the system, the user must know that the server program should be launched on the server to view the result.



### **6.2.1 User Training**

User training is designed to prepare a user for testing and converting a system. To achieve the goal and the benefits of the computer system, it is essential that people who are involved will convince their role in the new system. As the system becomes more complex, training is more important. User, the user learns how to enter data, respond to error messages, interrogate a database and call a routine that will create messages and perform other necessary functions.

### **6.2.2 Training on the Application Software**

After providing the necessary basic training of computer awareness, the user will have to be trained on new application software. This provides a basic philosophy of using a new system, such as the screen flow, the type of help on the screen, the type of data entering errors, corresponding to the verification check with each item and the method of repairing the specified date.

### **6.2.3 System Maintenance**

The maintenance of the system is an effective part, so the maintenance of the system applies to the ongoing activities needed to ensure that the system or application works efficiently and efficiently after implementation. It includes regular updates, bug fixes and performance optimization to keep the system smooth and safely in operation. The maintenance of the system is necessary to ensure that the system remains functional and effective after implementation. By introducing maintenance procedures and their consequence, project teams can ensure that the system works smoothly, remains safe and continues to suit the needs of end users.

### **6.2.4 Hosting**

Hosting allows the site to be accessible online by saving its files on the server. Different types of hosting ensure different needs: shared hosting is a budget friendly; VPS offers more control, reserved hosting provides full server access and cloud hosting increases scalability. Managed hosting processes technical tasks for you, while self -management allows complete inspection for technically proficient users. Good hosting ensures reliability, speed and security of websites, impact on user experience and SEO.

#### **Eg. Render Hosting**

Hosting the rendering often includes features such as server rendering (SSR), generating a static website (SSG) and content supply network (CDN) to optimize the load time and improve scalability. The server sends a fully designed HTML server by pre -rendering or using SSR, increasing SEO speed and performance.

**Procedure for hosting a website on Render Hosting:****Step 1: Set up a Render Account**

- Visit the render website and register for a new account or log in if you already have it.

**Step 2: Link GitHub Repository**

- After logging in to plot, go to the dashboard and click on the new button to create a new service.
- Select a web service as the type of service.
- If there is a prompt, connect your Github account to render if you haven't done it yet so.
- After connecting the GitHub account, select UMS storage from the list Available repositories.

**Step 3: Configure Deployment Settings**

- Select the appropriate branch for deployment (usually the main or main).
- Select the correct run for your project (for example, Python 3.x for flask applications).
- Set the assembly command to install addiction: PIP install -R requests.txt.
- Set the start command to start the app for the flask: Gunicorn UMS.wsgi.

**Step 4: Set up the Database (MongoDB)**

- Since the project uses as a Mongoddb database, no comprehensive database configuration is required to render. However, make sure the database file is part of the repository And it is correctly configured in your project.
- If you plan to migrate your database, make sure the correct Mongoddb commands are running During deployment to use migration (eg Python Manage.Py migrate).

**Step 5: Deploy the Project**

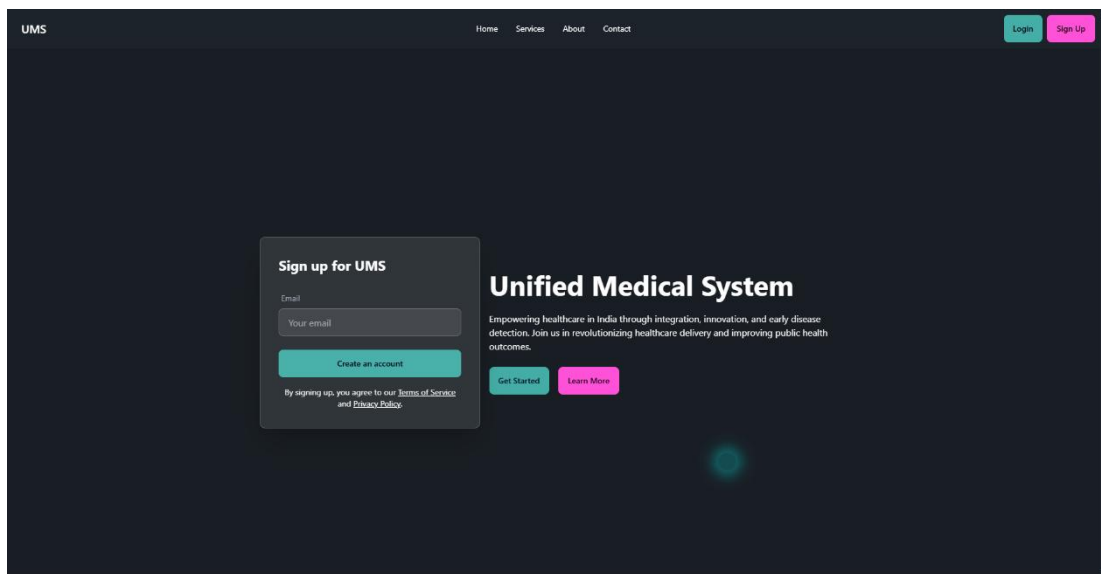
- After setting everything, click Create a web service to start the deployment process.
- Drawing automatically creates a project, installs addiction and starts the web service.
- After the deployment is completed, your project will be alive on the URL provided.

**Hosted Link:** <https://unified-medical-system.onrender.com/>

**Hosted Link QR Code**



## Screenshot



## **CHAPTER 7**

### **CONCLUSION AND FUTURE SCOPE**

## 7.1 CONCLUSION

In conclusion, the Unified Medical System (UMS) represents transformation solutions for providing health care in India, centralizing services such as reservation, access to medical record, data sharing and detection of timeless diseases within a secure, user -friendly platform. By using modern UMS technologies, it not only increases the availability and efficacy of health care, but also employs patients to actively manage their health information and promote proactive public health measures. Although challenges exist in infrastructure, privacy and user acceptance, gradual implementation in combination with robust data security protocols and comprehensive users' training ensures technical, economic and behavioral system feasibility. UMS has a significant potential to improve health care results, optimize resource allocation and promote the vision of India for a more integrated, more sensitive and resistant health care system.

## 7.2 FUTURE SCOPE

The future scope of the unified medical system (UMS) has a huge potential for transforming health care in India. As the infrastructure develops, UM can be expanded to rural areas and ensures fair access to nationwide health care sources. Telemedicine Integration offers patients of comfort of remote consultations, making health care accessible from any place, especially for those in distant areas. With advanced data analysis, the system can use the machine learning to predict the needs of health care, allowing better allocation of resources and proactive care. UMS could also integrate data from wearable devices to allow real -time monitoring, early intervention for chronic diseases and preventive health measures. In addition, aggregated health data from UM can help public health officials in monitoring trends, predicting the outbreak of diseases and creating evidence -based policies and improving public health results. Diagnosis AI controlled and personalized care solutions based on patient history will also seize health care providers to offer targeted and effective care. With these advances, UMS has the potential to become a model for health systems around the world, to facilitate international cooperation and contribute to a more resistant, interconnected medical framework.

.

.

## **CHAPTER 8**

### **BIBLIOGRAPHY**

**REFERENCES:**

- Chatterjee, A., Kumar, V., & Mahapatra, S. (2020). Machine learning models for disease surveillance: A review. *Journal of Public Health and Epidemiology*, 12(4), 98–104.
- Kumar, R., & Singh, P. (2022). Digital health infrastructure in India: A review of Ayushman Bharat Digital Mission. *Journal of Health Policy and Systems Research*, 10(2), 67–73.
- Nair, S., Gupta, A., & Bhatia, M. (2021). The impact of AI-driven chatbots on healthcare: Improving patient engagement and self-management. *Healthcare Informatics Research*, 27(1)

**WEBSITES:**

- <https://doi.org/10.5897/JPHE2020>
- <https://doi.org/10.1016/j.jhpsr.2022.07.004>
- <https://doi.org/10.1146/hsj2021.601>

## **CHAPTER 9**

### **APPENDIX**



## 9.1 Sample Code

### Login

```
import from flask import Blueprint, render_template, redirect, url_for, request, flash, session
from werkzeug.security import generate_password_hash, check_password_hash
from flask_login import login_user, logout_user, login_required
from app import mongo, login_manager, oauth, mail
from app.models import User
from bson.objectid import ObjectId
from flask_mail import Message
from app.utils import generate_patient_id
from datetime import datetime
import random
```

```
auth_bp = Blueprint('auth', __name__)
```

```
@login_manager.user_loader
```

```
def load_user(user_id):
    return User.get(user_id)
```

```
@auth_bp.route('/register', methods=['GET'])
```

```
def register():
    return render_template('common/register.html')
```

```
@auth_bp.route('/login', methods=['GET', 'POST'])
```

```
def login():
    if request.method == 'POST':
        identifier = request.form['identifier']
        password = request.form['password']
        user = User.find_by_identifier(identifier)
        if user:
            if user.passwordHash is None:
                flash('User password is missing')
                return redirect(url_for('auth.login'))

        #role selection
        if check_password_hash(user.passwordHash, password):
            session['umsId'] = user.umsId
            if user.rolesId == 4:
                login_user(user)
                return redirect(url_for('patient.index'), code=302)
            elif user.rolesId == 3:
                login_user(user)
                if user.status == 'awaiting_approval':
                    flash('Your account is waiting for admin approval')
                    return render_template('common/awaiting_response.html')
                elif user.status == 'active':
                    return redirect(url_for('doctor.index'), code=302)
            elif user.rolesId == 2:
                login_user(user)
```

```

        return redirect(url_for('hospital.index'), code=302)
    elif user.rolesId == 1:
        login_user(user)
        return redirect(url_for('admin.index'), code=302)
    else:
        flash('Invalid identifier or password')
    else:
        flash('User not found, please create an account')
    return render_template('auth/login.html')

```

## Register

```

def generate_patient_id():
    return 'UMSP' + re.sub('-', '', str(uuid.uuid4()))[:8].upper()

@patient_bp.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        email = request.form['email']
        password = request.form['password']
        if password != request.form['confirm_password']:
            flash('Passwords do not match. Please try again.', 'error')
            return redirect(url_for('patient.register'))

    existing_user = mongo.db.users.find_one({'email': email})
    if existing_user:
        flash('Email already exists! Please login.', 'error')
        return redirect(url_for('patient.register'))
    name = request.form['name']
    phone_number = request.form['phonenummer']
    state = request.form['state']
    umsId = generate_patient_id()
    created_at = updated_at = datetime.now()
    new_user = {
        'umsId': umsId,
        'name': name,
        'email': email,
        'phoneNumber': [phone_number],
        'state': state,
        'status': 'active',
        'createdAt': created_at,
        'updatedAt': updated_at,
        'passwordHash': [generate_password_hash(password)],
        'rolesId': 4
    }
    mongo.db.users.insert_one(new_user)
    mongo.db.patients.insert_one({
        'umsId': umsId,
        'dateOfBirth': None,
        'gender': None,
        'createdAt': created_at,
        'updatedAt': updated_at
    })
    mongo.db.login.insert_one({

```

```

        'umsId': umsId,
        'email': email,
        'passwordHash': new_user['passwordHash'],
        'rolesId': 4,
        'status': 'active',
    })
    flash('User registered successfully!', 'success')
    return redirect(url_for('auth.login'))
    return render_template('patient/register.html')

```

## Book Appointment

```

@patient_bp.route('/book_appointment', methods=['POST'])
@login_required
def book_appointment():
    data = request.json
    patient_id = current_user.umsId
    hospital_id = data.get('hospitalId')
    doctor_id = data.get('doctorId')
    category = data.get('category')
    appointment_date_str = data.get('appointmentDate')
    is_disabled = data.get('isDisabled', False)
    disability_id = data.get('disabilityId')
    reason = data.get('reason')
    if not all([hospital_id, doctor_id, category, appointment_date_str]):
        return jsonify({'success': False, 'message': 'Missing required fields'}), 400
    try:
        appointment_date = parser.isoparse(appointment_date_str)
        appointment_date_utc = appointment_date.astimezone(timezone.utc)
        new_appointment = {
            'patientId': patient_id,
            'hospitalId': hospital_id,
            'doctorId': doctor_id,
            'category': category,
            'appointmentDate': appointment_date_utc,
            'status': 'pending',
            'isDisabled': is_disabled,
            'disabilityId': disability_id if is_disabled else None,
            'reason': reason,
            'createdAt': datetime.now(timezone.utc),
            'updatedAt': datetime.now(timezone.utc)
        }
        result = mongo.db.appointments.insert_one(new_appointment)
        if result.inserted_id:
            return jsonify({'success': True, 'message': 'Appointment booked successfully'})
        else:
            return jsonify({'success': False, 'message': 'Failed to book appointment'}), 500
    except Exception as e:
        return jsonify({'success': False, 'message': 'An error occurred while booking the appointment'}), 500

```

## Download medical certificate

```

@patient_bp.route('/api/medical_records/<block_id>/pdf', methods=['GET'])
@login_required
def download_medical_record_pdf(block_id):

```

```

# Find the specific block
block = mongo.db.medicalRecords.find_one({'_id': ObjectId(block_id)})
if not block:
    return jsonify({'error': 'Record not found'}), 404
# Find the transaction for the current user
transaction = next((t for t in block['transactions'] if t['patientId'] == current_user.umsId), None)
if not transaction:
    return jsonify({'error': 'Record not found for this patient'}), 404
# Create a PDF
buffer = BytesIO()
doc = SimpleDocTemplate(buffer, pagesize=letter,
                        rightMargin=72, leftMargin=72,
                        topMargin=72, bottomMargin=18)
# Container for the 'Flowable' objects
elements = []
# Styles
styles = getSampleStyleSheet()
styles.add(ParagraphStyle(name='Justify', alignment=1))
styles.add(ParagraphStyle(name='Center', alignment=1))
# Add UMS logo
logo_path = os.path.join(current_app.root_path, 'static', 'images', 'ums_logo.png')
if os.path.exists(logo_path):
    logo = Image(logo_path, width=1.5*inch, height=1.5*inch)
    elements.append(logo)
# Add title
title = Paragraph("Unified Medical System", styles['Heading1'])
elements.append(title)
elements.append(Spacer(1, 12))
# Add subtitle
subtitle = Paragraph("Medical Certificate", styles['Heading2'])
elements.append(subtitle)
elements.append(Spacer(1, 24))
# Add content
content = f"""
This is to certify that the patient with UMS ID: {current_user.umsId} has been examined and treated at
our facility.
The following medical record details the diagnosis, treatment, and recommendations for the patient.
"""
elements.append(Paragraph(content, styles['Justify']))
elements.append(Spacer(1, 12))
# Create a table for the medical record details
data = [
    ['Patient ID:', current_user.umsId],
    ['Date of Record:', transaction['createdAt'].strftime('%Y-%m-%d %H:%M:%S')],
    ['Doctor ID:', transaction['doctorId']],
    ['Hospital ID:', transaction['hospitalId']],
    ['Symptoms:', transaction['Symptoms']],
    ['Diagnosis:', transaction['Diagnosis']],
    ['Treatment Plan:', transaction['TreatmentPlan']],
    ['Prescription:', transaction['Prescription']],
    ['Additional Notes:', transaction['AdditionalNotes']],
    ['Follow-up Date:', transaction['FollowUpDate']],
]
table = Table(data, colWidths=[2*inch, 4*inch])
table.setStyle(TableStyle([

```

```

('BACKGROUND', (0, 0), (0, -1), colors.lightblue),
('TEXTCOLOR', (0, 0), (-1, 0), colors.darkblue),
('ALIGN', (0, 0), (-1, -1), 'LEFT'),
('FONTNAME', (0, 0), (-1, 0), 'Helvetica-Bold'),
('FONTSIZE', (0, 0), (-1, -1), 10),
('BOTTOMPADDING', (0, 0), (-1, -1), 12),
('BACKGROUND', (1, 1), (-1, -1), colors.lightgreen),
('BOX', (0, 0), (-1, -1), 1, colors.black),
('GRID', (0, 0), (-1, -1), 0.5, colors.black),
]))
elements.append(table)
elements.append(Spacer(1, 24))
# Add footer
footer_text = f"This medical certificate is electronically generated and is valid without a
signature.\nBlock Hash: {block['hash']}"
footer = Paragraph(footer_text, styles['Center'])
elements.append(footer)
# Build the PDF
doc.build(elements)
buffer.seek(0)
return send_file(buffer, as_attachment=True, download_name=f'medical_certificate_{block_id}.pdf',
mimetype='application/pdf')

```

## 9.2 Screen Shots

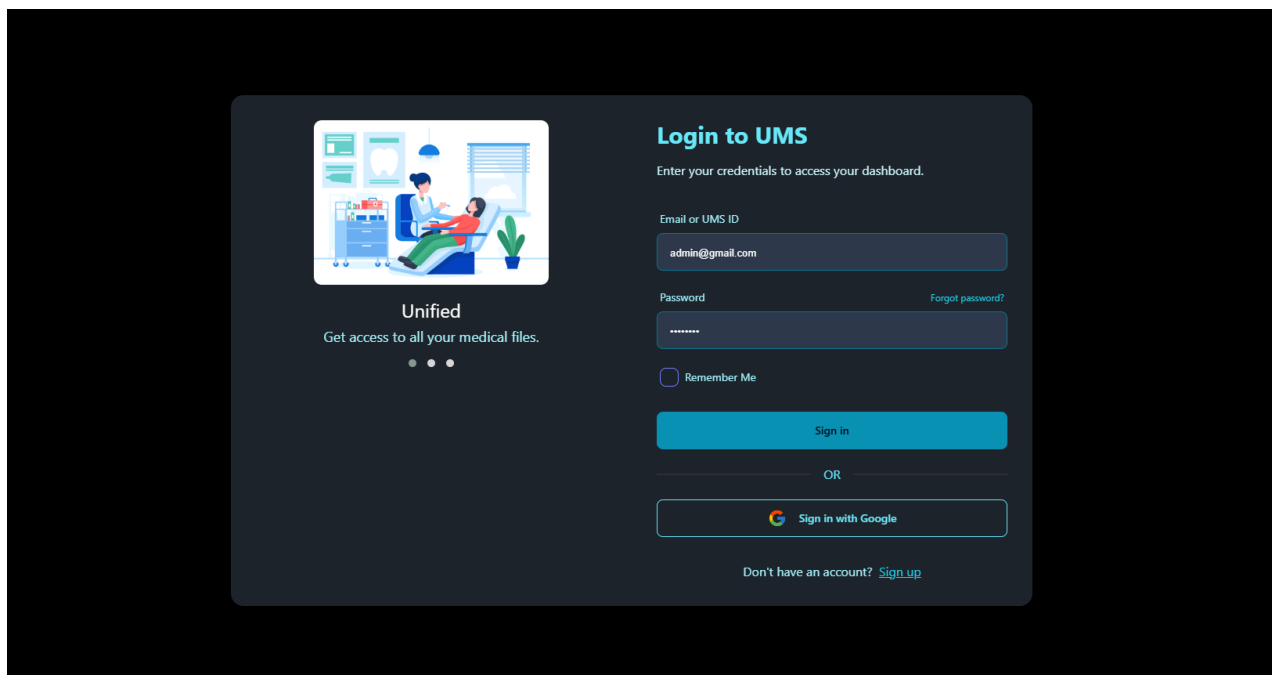
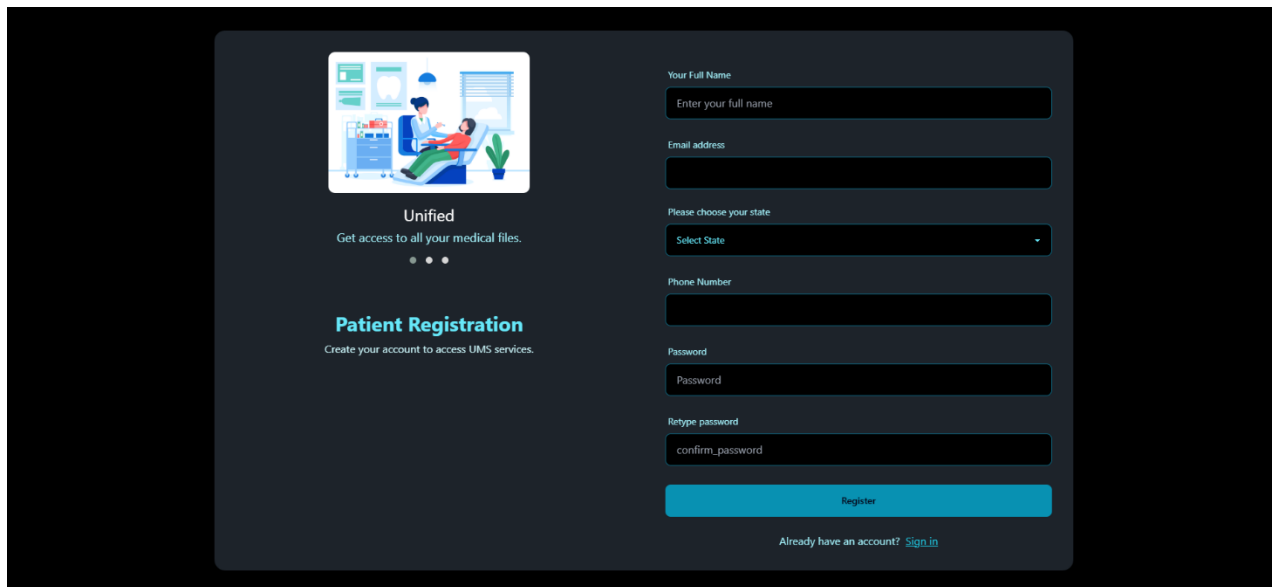


Figure 1. Login page



The registration page features a dark-themed layout. On the left, an illustration shows a doctor attending to a patient in a hospital bed. Below this, the text 'Unified' is followed by 'Get access to all your medical files.' and three dots. The main heading is 'Patient Registration' with the subtext 'Create your account to access UMS services.' On the right, a registration form includes fields for 'Your Full Name', 'Email address', 'Please choose your state' (a dropdown menu), 'Phone Number', 'Password', and 'Retype password'. A blue 'Register' button is at the bottom of the form, and a link 'Already have an account? Sign in' is below it.

Figure 2. Registration page

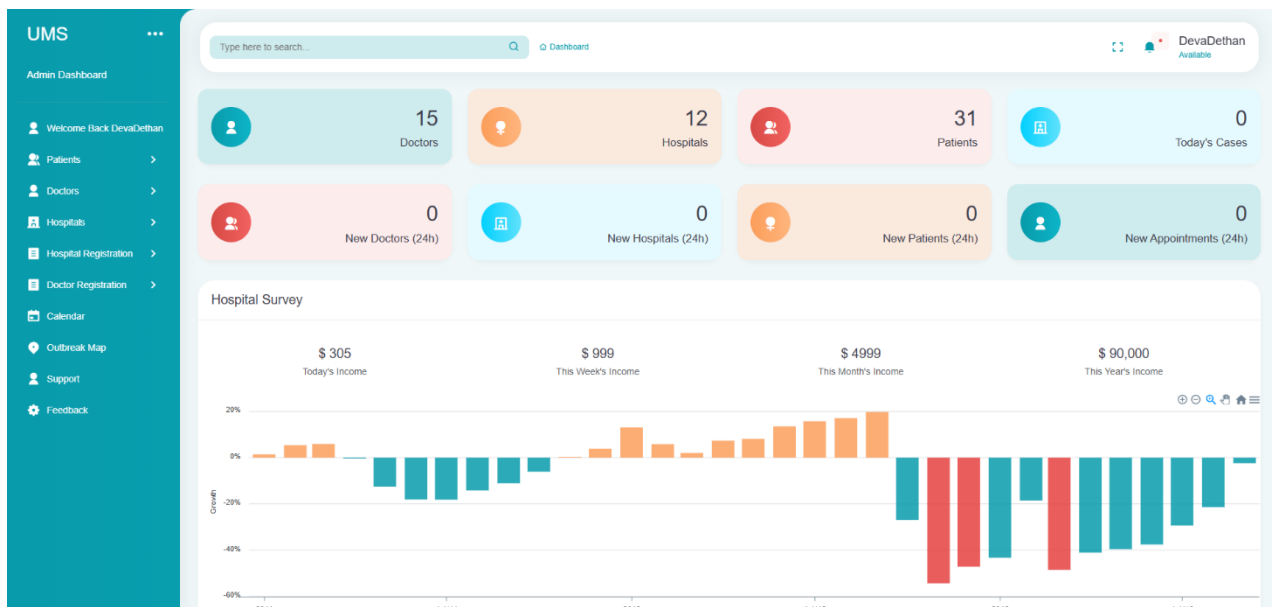


Figure 3. Admin page

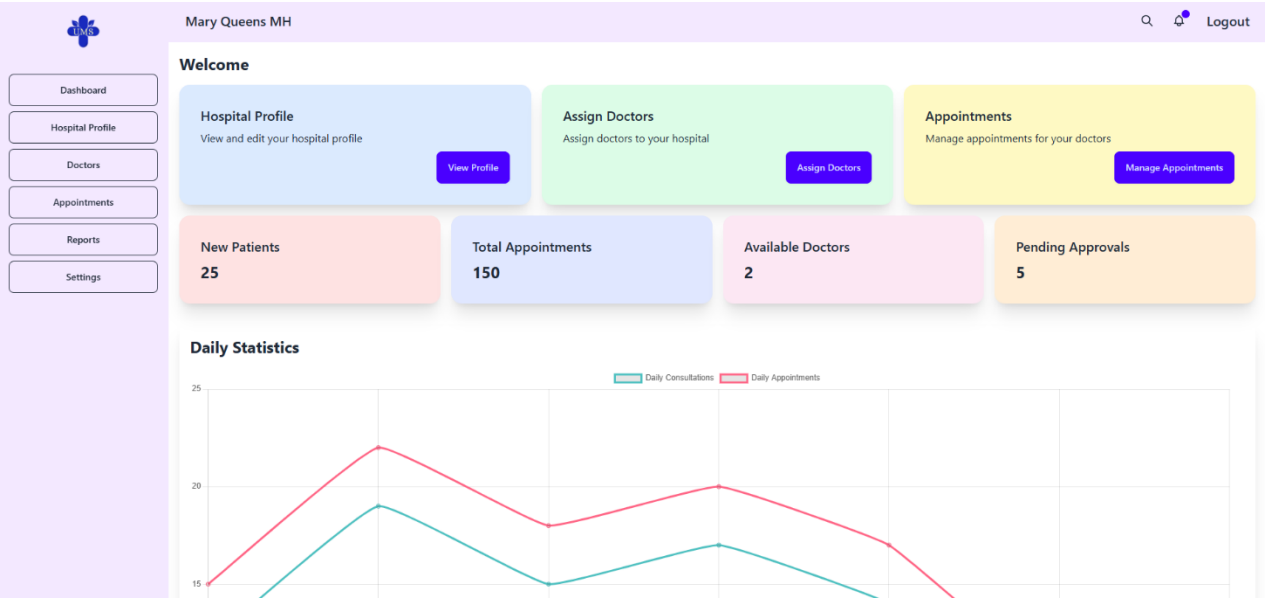


Figure 4. Hospital Dashboard

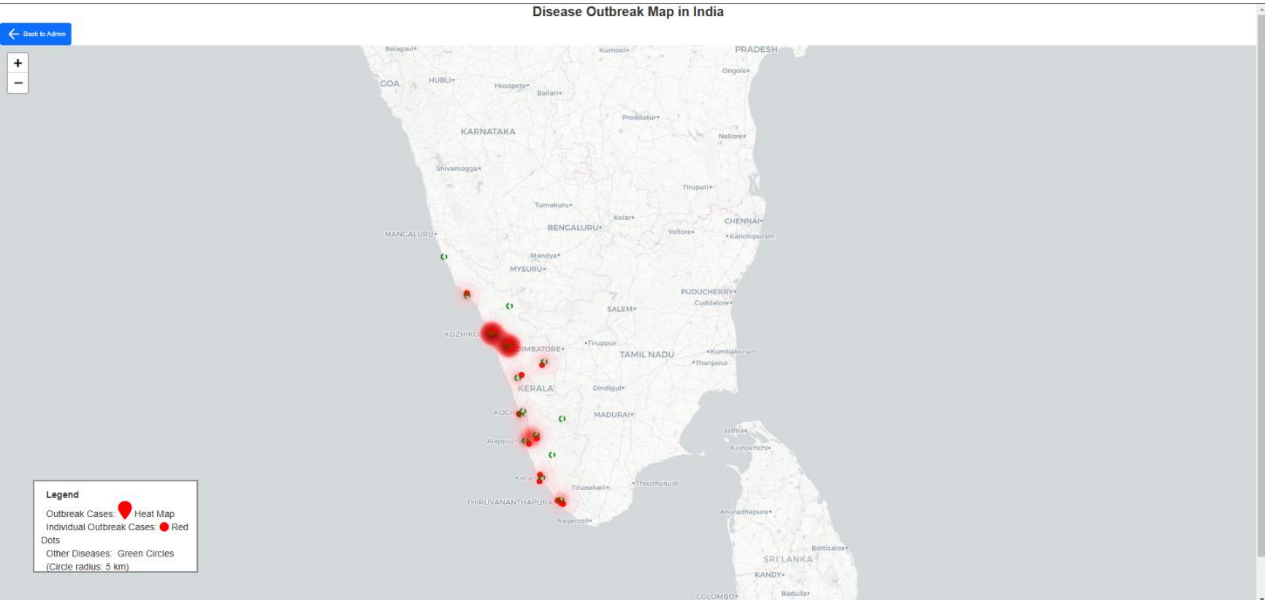


Figure 5. Disease Outbreak Map

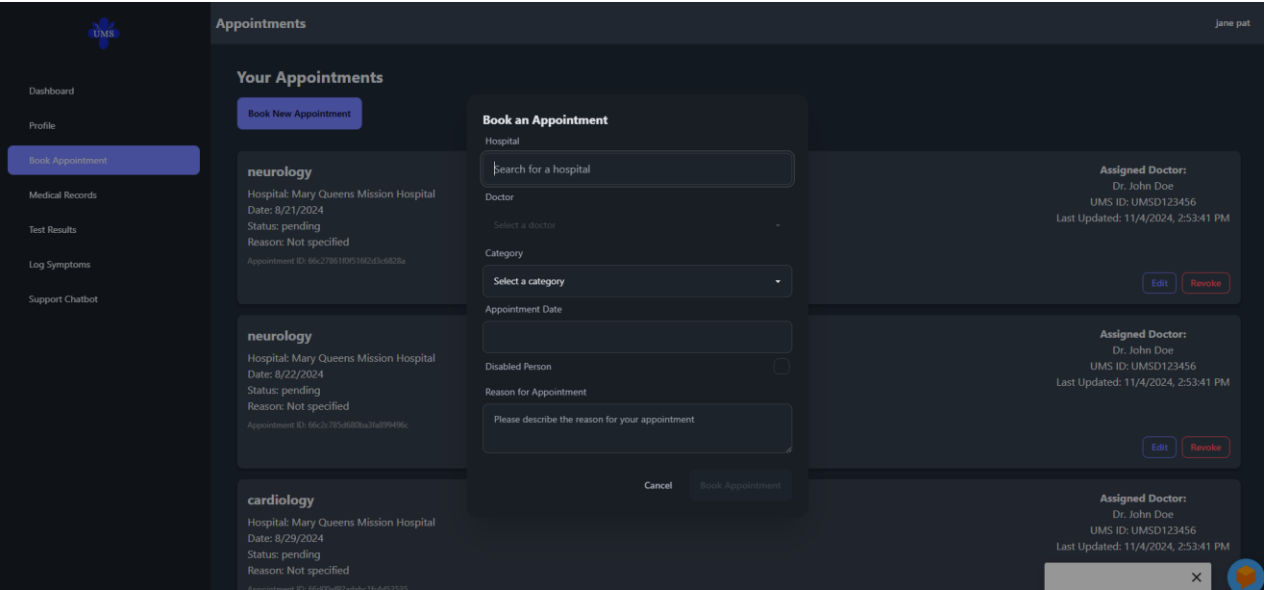


Figure 6. Appointment Page

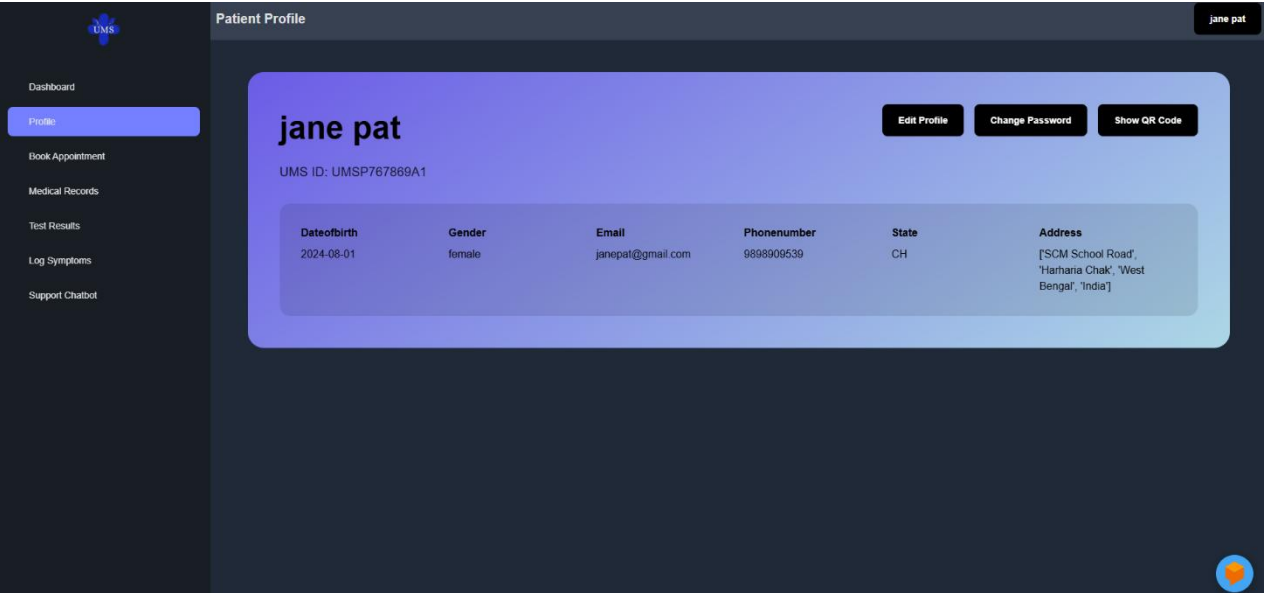


Figure 7. Patient Dashboard

9.3Git Log



4/8/25, 9:56 PM

Commits · devadethanr/unified-medical-system-main · GitHub

devadethanr / unified-medical-system-main

Public

<> Code

Issues

Pull requests

Actions

Projects

Security

Insights

Commits

main

All users

All time

Commits on Feb 11, 2025

initial ui for dbchat

11d8a56

<>

...

devadethanr committed on Feb 11 · 2 / 2

Commits on Dec 8, 2024

changes to hospital route

7112474

<>

...

devadethanr committed on Dec 8, 2024 · 1 / 1

Commits on Nov 24, 2024

changes to temp

89688a8

<>

...

devadethanr committed on Nov 24, 2024 · 1 / 1

changes to base.html

8c79aac

<>

...

devadethanr committed on Nov 24, 2024 · 1 / 1

Commits on Nov 17, 2024

updated docs

9f95c83

<>

...

devadethanr committed on Nov 17, 2024 · 1 / 1

Commits on Nov 5, 2024

qr

8a8377d

<>

...

devadethanr committed on Nov 5, 2024 · 1 / 1

https://github.com/devadethanr/unified-medical-system-main/commits/main/

1/5

4/8/25, 9:56 PM

Commits · devadethanr/unified-medical-system-main · GitHub

req

e69b3ec

<>

...

devadethanr committed on Nov 5, 2024 · 1 / 1

render

2ba57e6

<>

...

devadethanr committed on Nov 5, 2024 · 1 / 1

Commits on Nov 4, 2024

testing and docs

b7d718a

<>

...

devadethanr committed on Nov 4, 2024 · 1 / 1

Commits on Oct 28, 2024

Fix deployment configuration and add gunicorn to requirements

f681201

<>

...

devadethanr committed on Oct 28, 2024 · 1 / 1

Fix deployment configuration and add gunicorn to requirements

a874c1d

<>

...

devadethanr committed on Oct 28, 2024 · 1 / 1

render

67dc336

<>

...

devadethanr committed on Oct 28, 2024 · 1 / 1

render

a5f3cd7

<>

...

devadethanr committed on Oct 28, 2024 · 1 / 1

changes to render

39a4809

<>

...

devadethanr committed on Oct 28, 2024 · 1 / 1

req

b1aefb84

<>

...

devadethanr committed on Oct 28, 2024 · 1 / 1

changes on req

273d9d7

<>

...

devadethanr committed on Oct 28, 2024 · 1 / 1

https://github.com/devadethanr/unified-medical-system-main/commits/main/

2/5

Amal Jyothi College of Engineering Autonomous, Kanjirappally

Department of Computer Applications

4/8/25, 9:56 PM

Commits · devadethan/unified-medical-system-main · GitHub

render config

78567cf

<>

...

devadethan committed on Oct 28, 2024 · ✓ 1 / 1

Commits on Oct 18, 2024

close error fixed

b4f897d

<>

...

devadethan committed on Oct 18, 2024 · ✓ 1 / 1

verify on blockchain done

a35453b

<>

...

devadethan committed on Oct 18, 2024 · ✓ 1 / 1

added the pdf and hashcode

9598238

<>

...

devadethan committed on Oct 18, 2024 · ✓ 1 / 1

Commits on Sep 23, 2024

trying vercel deployment\

594e593

<>

...

devadethan committed on Sep 24, 2024 · ✓ 1 / 1

trying vercel deployment\

feb7a2b

<>

...

devadethan committed on Sep 24, 2024 · ✓ 1 / 1

trying vercel deployment\

982f8de

<>

...

devadethan committed on Sep 24, 2024 · ✓ 1 / 1

trying vercel deployment\

d4c8ed5

<>

...

devadethan committed on Sep 24, 2024 · ✓ 1 / 1

files added for Transformer model

a374491

<>

...

devadethan committed on Sep 23, 2024

outbreak detection

<>

...

devadethan committed on Sep 23, 2024

https://github.com/devadethan/unified-medical-system-main/commits/main/

35

4/8/25, 9:56 PM

Commits · devadethan/unified-medical-system-main · GitHub

a23a2da

<>

...

devadethan committed on Sep 23, 2024

outbreak detection

8f2541c

<>

...

devadethan committed on Sep 23, 2024

files added for Transformer model

44148b9

<>

...

devadethan committed on Sep 23, 2024

outbreak map

22ab50e

<>

...

devadethan committed on Sep 23, 2024

changes done locally

ba55f9c

<>

...

devadethan committed on Sep 23, 2024

changes done locally

877f34d

<>

...

devadethan committed on Sep 23, 2024

Commits on Aug 18, 2024

new gitignore

67d983c

<>

...

devadethan committed on Aug 19, 2024

Remove \_\_pycache\_\_ directories from repository

9587853

<>

...

devadethan committed on Aug 19, 2024

client secrets

4447a78

<>

...

devadethan committed on Aug 19, 2024

pycache

a23aeF4

<>

...

devadethan committed on Aug 19, 2024

Previous

Next >

https://github.com/devadethan/unified-medical-system-main/commits/main/

45

Amal Jyothi College of Engineering Autonomous, Kanjirappally

Department of Computer Applications

## 9.4 Certificates



Figure 1. Certificate 1



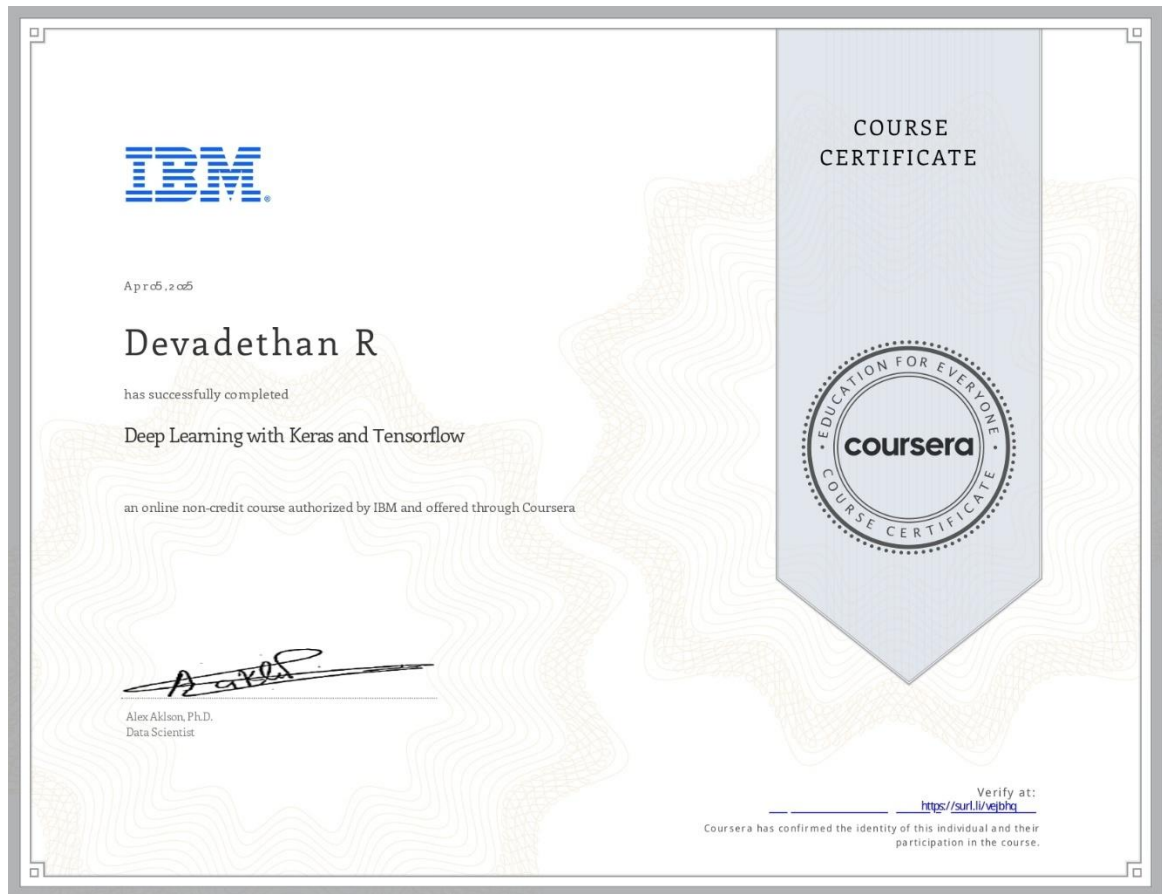
Figure 2. Certificate 2



Figure 3. Certificate 3



Figure 4. Certificate 4



**Figure 5. Certificate 5**

## Page 1 of 22 - Cover Page



Page 1 of 12 - Current Page

bioRxiv preprint doi: <https://doi.org/10.1101/286461>; this version posted April 11, 2018. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC-ND 4.0 International license.

Project2  
AJCE-MCA  
Amal Jyothi College of Engineering

Submission ID  
b7c0ddc1-f32f-2646f732

Submission Date  
Apr 11, 2025, 12:12 PM GMT+5:30

Download Date  
Apr 11, 2025, 1:57 PM GMT+5:30

File Name  
Report\_DesadethanRi\_-\_Copy.pdf

File Size  
2.0 MB

70 Pages  
11,189 Words  
78,484 Characters



Page 1 of 22 - Current Page

bioRxiv preprint doi: <https://doi.org/10.1101/284613>; this version posted November 15, 2018. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC-ND 4.0 International license.



Page 2 of 22 - AI Writing Overview

Submission ID: 123456789

## \*% detected as AI

AI detection includes the possibility of false positives. Although some text in this submission is likely AI-generated, scores below the 20% threshold are not surfaced because they have a higher likelihood of false positives.

### Caution: Review required.

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.

### Disclaimer

Our AI writing assessment is designed to help educators identify text that might be prepared by a generative AI tool. Our AI writing assessment may not always be accurate (it may occasionally writing that is likely AI-generated or AI-paraphrased or likely AI-generated and AI-paraphrased writing as only AI-generated) so it should not be used as the sole basis for adverse actions against a student. It takes further scrutiny and human judgment in conjunction with an organization's application of its specific academic policies to determine whether any academic misconduct has occurred.

## Frequently Asked Questions

### How should I interpret Turnitin's AI writing percentage and false positives?

The percentage shown in the AI writing report is the amount of qualifying text within the submission that Turnitin's AI writing detection model determines was either likely AI-generated text from a large-language model or likely AI-generated text that was likely revised using an AI-paraphrase tool or word spinner.

False positives (incorrectly flagging human-written text as AI-generated) are a possibility in AI models.

AI detection scores under 20%, which we do not surface in new reports, have a higher likelihood of false positives. To reduce the likelihood of misinterpretation, no score or highlights are attributed and are indicated with an asterisk in the report (\*%).

The AI writing percentage should not be the sole basis to determine whether misconduct has occurred. The reviewer/instructor should use the percentage as a means to start a formative conversation with their student and/or use it to examine the submitted assignment in accordance with their school's policies.

### What does 'qualifying text' mean?

Our model only processes qualifying text in the form of long-form writing. Long-form writing means individual sentences contained in paragraphs that make up a longer piece of written work, such as an essay, a dissertation, or an article, etc. Qualifying text that has been determined to be likely AI-generated will be highlighted in cyan in the submission, and likely AI-generated and then likely AI-paraphrased will be highlighted purple.

Non-qualifying text, such as bullet points, annotated bibliographies, etc., will not be processed and can create disparity between the submission highlights and the percentage shown.



Page 2 of 22 - AI Writing Overview

Submission ID: 123456789