F24-143-D-HealthBridge

Project Team

Muhammad Abdullah	21I-0643
Muneel Haider	21I-0640
Abdullah Zahoor	21I-2481

Session 2021-2025

Supervised by

Mr. Muhammad Aadil Ur Rehman

Co-Supervised by

Dr. Zeeshan Khan

Department of Computer Science

National University of Computer and Emerging Sciences
Islamabad, Pakistan

June, 2025

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Pakistan's healthcare system consists of public and private sectors, which provide various healthcare services such as hospitals, clinics, and diagnostic centers. The public sector is affiliated with the federal and provincial governments that offer various services which is either free or heavily subsidized, with the main focus on the general population, especially those living in rural areas. On the other hand, the private sector provides a lot more special and highly individual care services relatively, but at a higher cost, through hospitals, clinics, and diagnostic centers. In both sectors, the process of receiving healthcare begins with booking an appointment. For public hospitals, patients generally visit the outpatient department (OPD), where they ask for consultancy with doctors for a specific problem and register. Furthermore, the appointments are typically on first-come, first-served basis. Alternatively, private hospitals offer booking in advance via phone calls or an online management system which significantly reducing waiting times. After booking an appointment, the doctors may consult patients to get diagnostic tests with specialists before diagnosing a problem or a recommending a treatment. For instance, if a doctor suspects that further diagnosis is necessary, they refer the patients for additional diagnostic tests such as blood tests, X-ray, MRI, CT scans. The patients then visit the diagnostic centers for taking tests and then receive a report accordingly depending on the scans which sometimes take a few hours or several days. When the reports are generated, the patient returns to the doctor and again wait for their turn to get a follow-up consultation from the doctor who reviews the findings and decides on the appropriate course of action or recommends a treatment process along with the prescription. This structured approach is consistent for both public and private healthcare systems nationwide, in order to ensure complete diagnosis and management to assist patients.

1.1 Existing Solutions

For the Existing Solutions section, students should research and describe various solutions, methods, or approaches that have been previously developed to solve the problem

they are addressing. This section should include an overview of the different techniques, technologies, or models that are currently in use or have been applied in the past. Students should critically assess the strengths and weaknesses of each solution, highlighting any gaps or limitations that their project aims to address. Additionally, this section should show an understanding of the state-of-the-art in the field and provide context for why the proposed solution is necessary or an improvement over existing ones.
Several solutions have been developed in the recent years to cater such healthcare problems, especially for the patient-doctor communication, diagnostics, and appointment management to reduce further delays which affect many patients' health. These systems consist of advanced technologies such as telehealth, automated workflows, and artificial intelligence in order to address such challenges. Below is the detailed analysis of the most well-known applications that have tried to eliminate such problems.

Table 1.1: Comparison of Existing Solutions

System Name	**System Overview**	**System Limitations**
Oladoc	A digital healthcare service in Pakistan that offers appointment booking.	Mostly not applicable in rural areas. Threat to data privacy.
Marham	Enables patients to book appointments with specific doctors for better communication.	Reliant on digital literacy and only used for appointment booking.
Qure.ai	Focuses on developing AI tools for CT and X-ray scans assistance. Enhances diagnostic accuracy for radiologists.	Focuses on X-ray and CT scans only and not MRI scans. Potential errors in complex scans.
HeartFlow	Provides an artery disease diagnosis using advanced imaging algorithms along with 3D artery models.	Restricted to only artery diseases and high cost.
Aidoc	An AI-based platform for several specialties (cardiology and neurology) which provides accurate diagnosis.	Its only limited to specific specialties along with the high cost.

However, the above solutions are only effective in their own domains as they lack an all-in-one platform for addressing such challenges on a broader aspect, particularly in regions like Pakistan, especially for rural areas. Some of the key gaps include:

- **AI-driven liver diagnostics:** Provides tumor and fatty liver detection with accurate

results.

- **Online appointment scheduling and real-time consultations:** Enables chat systems and video calls for better accessibility.
- **Comprehensive support for patients:** Especially designed for rural areas to ensure enhanced healthcare delivery on time.

By integrating such features into a single application, **HealthBridge** not only addresses the limitations of existing applications but also enhances overall efficiency, accessibility, and provides better solutions for modern real-time problems.

1.2 Problem Statement

The healthcare system in Pakistan for both public and private sectors, face various problems which effect the overall quality and accessibility of the services. These challenges create difficulties for both patients and doctors in several ways. Most of the healthcare services in rural areas consist of public hospitals and clinics. These systems provide free medication and treatment for most of the patients due to which many patients prefer such medical services. However, these facilities are mostly inadequate as they provide very limited advanced diagnostics and specialized care generally. Hospitals mainly in urban areas are mostly overcrowded as many patients are forced to wait in long queues which hinders the severity of the problem for patients. Hence, further endangering the health of the patients. Most of the hospitals especially in rural areas do not offer a pre-booking appointment system especially in underdeveloped areas. Alternatively, private hospitals appointment systems are more efficient but cannot be accessed through digital and mobile services. Delays in diagnostic reports have caused several critical issues to the healthcare system whereby almost every hospital in the country would take a number of diagnostic reports and test results for each patient, mostly in the case of specialized procedures like MRI, CT scans and related diagnostics reports. Such reports are usually processed by the diagnostic centers within hours or sometimes even days, which leads to further delays to the whole process of diagnosing a problem and recommending some treatment for the patient. Patients mostly visit doctors' multiple times for consultation and follow-ups, mainly after receiving the test results that are previously recommended by the doctors. This increases the wait time and leads to more logistical challenges. Patients follow a navigated healthcare system, moving from doctors to specialists and diagnostic centers without getting a final result or a recommended treatment. Such system leads to inefficiencies as patient is unclear of the problem and the doctor remains confused. Such problems highlight the need for improving the overall healthcare accessibility and efficiency in Pakistan. Although there are apps available for appointment booking and report generation, they are hosted on different platforms. Patients need to navigate separate apps to utilize these services, making the process more complicated and time-consuming. While AI-assisted diagnostic apps exist, very few focus on liver diseases, an area that requires more attention, especially given the rising prevalence of liver-related health issues.

1.3 Scope

HealthBridge is an all-in-one platform which aims to provide a liver-related medical diagnostics by integrating an AI-Powered healthcare services. The platform allows the users to book an appointment with their relevant doctors, whether for physical or video consultations, ensuring accessibility across all regions, including those with limited facilities. It features diagnostics specifically for liver diseases such Tumor, Cancer, and Fatty Liver

(hepatic steatosis). These reports are generated in real-time with basic terminologies that are easy to comprehend. Such AI model also aims to assist healthcare professionals and patients. HealthBridge is meant to be accessible on web and mobile applications to ensure the ease of use for users. This platform aims to reduce the delays which are often associated with traditional diagnostic methods by providing an instant report and treatment recommendations when necessary. Focusing exclusively on liver diagnostics, HealthBridge aims to deliver a centralized and efficient solution, ensuring the quality of healthcare is accessible and reliable for patients in need.

1.4 Modules

The modules listed below focus on each specific aspect and features of the project in order to ensure a structured approach for the development. These modules comprise of web application, HealthBridge virtual assistant, diagnostic liver model, and android app interface for a comprehensive healthcare solution.

1.4.1 Module 1: Web Application

Design and implementation of Web Application.

1. Signup Page: Design and implementation of Signup Pages, unique for every type of user.
2. Login Page: Design and implementation of Login page, same for every type of user.
3. Authentication: Implement necessary authentication methods using tools like JWT or OAuth2.
4. Patient Interaction Platform: All pages related to searching, booking, communicating and using AI Model.
5. Doctor Dashboard and Services: All pages related to booking, communicating and using AI Model.
6. Design Database: Designing our projects database, with ERDs, and relational schemas.
7. Backend APIs: APIs will be made for our project and be used later to connect our webapp with the server.

1.4.2 Module 2: HealthBridge Virtual Assistant

Creation of HealthBridge Assistant Chatbot where users can interact with the Virtual Assistant.

1. Virtual Assistant Page Design: Design a single page for the Virtual Assistant, which will flow like a conversation until a result can be concluded

2. Connecting Virtual Assistant: Our Virtual Assistant will be fine-tuned and connected to our backend.

1.4.3 Module 3: Diagnostic Liver Model

Pre-processing our collected datasets and development/training of AI Model.

1. Labelling: We will start labelling our datasets and prepare them for segmentation.

2. Model Training: Here we will start to develop our notebooks/neural networks required to train our model.

3. Segmentation: Since we do not have algorithms required for automatic segmentation, we will also need to develop algorithms required for this task before we proceed with Model Training and Data Cleaning.

1.4.4 Module 4: Android App Interface

Start the development of our Android Application, and further improve communication and previous module features.

1. Portal: Algorithms developed for communication between users (patient and doctor) will be improved.

2. Android Application Development: All APIs have been made and tested on the web application, now the development of the android application will start and communicate with our backend.

3. Patient Interaction Platform: All pages related to searching, booking, and using AI Model.

4. Doctor Dashboard and Services: All pages related to booking, communicating and using AI Model.

5. Design Database: Designing our projects database, with ERDs, and relational schemas.

1.5 Work Division

Table 1.2: Work Division

Name	Registration	Responsibility / Module / Feature
Muneel Haider	21i-0640	(Module 1 - Feat 3, 6) (Module 1 - Feat 5) (Module 3 - Feat 2) (Module 4 - Feat 2) (Module 4 - Feat 1)
Muhammad Abdullah	21i-0643	(Module 1- Feat 1, 4, 5) (Module 1 – Feat 4) (Module 4 – Feat 3) (Module 4 – Feat 2, 3)
Abdullah Zahoor	21i-2481	(Module 1- Feat 2, 8) (Module 1 – Feat 5) (Module 3 – Feat 1) (Module 4 – Feat 1) (Module 4 – Feat 2, 3)

Chapter 2

Project Requirements

The project requirements define the main functionalities for the HealthBridge system. These requirements are functional and non-functional requirements, which describe the system's core components and operations, performance and usability standards.

2.1 Use-case

This section describes how users will interact with the HealthBridge system through various use cases that define certain scenarios and the relevant system response. Each use case provides a detailed interaction for each user (patient, doctor, system admin) and highlights the pre-conditions, main success scenario, and post conditions for the overall flow of events.

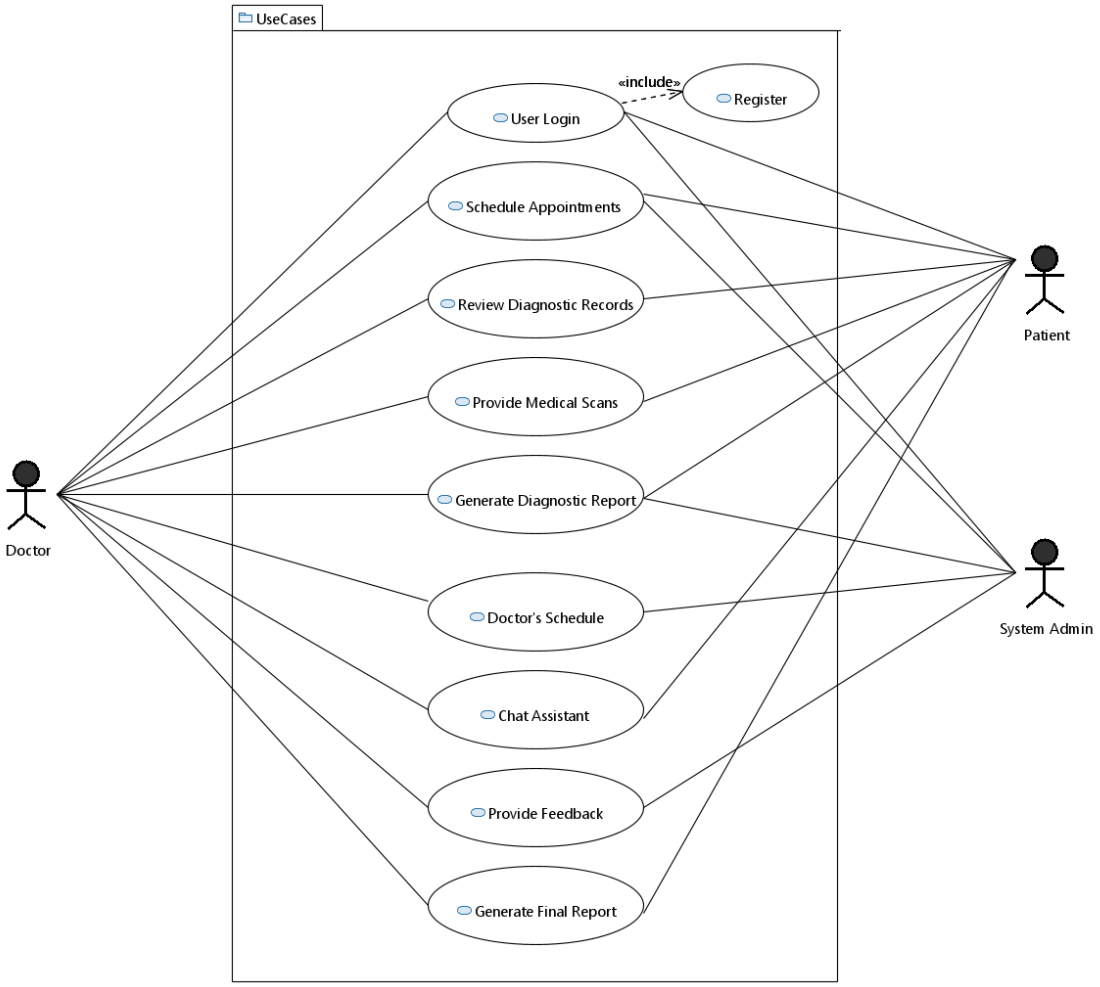


Figure 2.1: Use Case Diagram

2.1.1 Use-case 1: User Registration and Login

- **Scope:** HealthBridge System
- **Level:** User goal
- **Primary Actor:** Patient, Doctor, Institute Admin
- **Stakeholders and Interests:**
  1. Patient: Secure access to book an appointment and upload scans.
  2. Doctors: Access to schedule appointments and download assistive reports for patients safely.
  3. System Admin: To ensure that the platform is available for authorized users.
- **Pre-condition:**

1. Users do not have an account on the platform.
2. Invalid credentials.

- **Post-condition:**
  1. Users are successfully registered and logged in and are redirected to their dashboards.
- **Main success scenario:**
  1. User navigates to the login/registration page.
  2. User enters credentials (email, CNIC, password).
  3. The system validates the entered credentials.
  4. If the user logs in, they are successfully redirected to their dashboard.
- **Extensions:**
  1. The system shows an error message if the CNIC/email is already registered.
  2. The system shows an error message if the credentials are invalid.
  3. If the user forgets the password, they can request a password reset.
- **Special Requirements:**
  1. The encrypted password is stored in the database for secure management.
- **Technology and Data Variations:**
  1. Input: CNIC and password
  2. Output: Success or failure message

2.1.2 Use-case 2: Schedule Appointments

- **Scope:** HealthBridge System (Appointment Scheduling)
- **Level:** User goal
- **Primary Actor:** Patient, Doctor, System Admin
- **Stakeholders and Interests:**
  1. Patient: Authorized users can book, cancel, or reschedule appointments efficiently.
  2. Doctors: They can manage their schedule according to the given slots for patient consultations.

3. System Admin: Ensures appointment scheduling successfully without any logical errors.

- **Pre-condition:**
  1. Both patients and doctors successfully log into the system.
  2. Patients can see the available slots for booking an appointment with their relative doctor.
- **Post-condition:**
  1. The patient successfully books the appointment with their respective doctor and both users are notified.
- **Main success scenario:**
  1. Patient navigates to "Book Appointment" from their dashboard.
  2. Patient is redirected to the relevant page for booking.
  3. The system displays the list of available doctors with their respective schedule.
  4. The patient is then redirected to the payment method page.
  5. Patient enters the required credentials for secure payment.
  6. System validates the given credentials and proceeds with the overall payment process.
  7. The system notifies the patient for successful payment.
  8. System updates the schedule of the relevant doctor for the appointment after the approval.
  9. Both patient and doctor are notified in case of rescheduling or canceling the appointment.
- **Extensions:**
  1. The system shows only empty time slots for each doctor to remove ambiguity.
  2. If a patient cancels the appointment, the system updates both patient and doctor and the time slot is updated for the doctor.
- **Special Requirements:**
  1. Real-time updates
  2. Confirmation notification
- **Technology and Data Variations:**
  1. Doctor selection with date and time
  2. Appointment confirmation

2.2 Functional Requirements

2.2.1 Reliability

- The accuracy of our AI model will be above 80%.
- The system will accurately recommend respective doctors according to the problem statement of the patient.

2.2.2 Usability

- The system servers will be running at all times, ensuring the availability of the platform.
- The system will log out any users who are inactive for 15 minutes in a single session automatically, unless they choose to permanently log in.

2.2.3 Performance

- The system will be built on fast and reliable APIs, ensuring quick responses to users.
- The system will handle approximately 25 users in 5 seconds.
- The Chatbot should return a response within 4 seconds for a user query in 90% of cases.

2.2.4 Security

- All sensitive data that is saved in the database will be encrypted.
- The system will incorporate authorization methods, ensuring users are only granted access to features they are authorized to access.
- The system will use an authentication method, which ensures users that are logged in are not only authorized but also have valid token sessions through proper authentication.

Chapter 3

System Overview

The System Overview provides a more comprehensive description of the HealthBridge System. It showcases how several components of the system work using architecture design, data design, domain model, activity diagram, data-flow diagram, system-level sequence diagrams, and state transition diagram. These mentioned diagrams highlight the integration of modules like web application, liver diagnostic model, and virtual assistant in order to follow a systematic approach.

3.1 Architectural Design

The Architectural Design for HealthBridge uses a multi-layered MVC architecture design. This design provides a user-friendly interface and efficient data processing as each layer has its own specific role within the system.

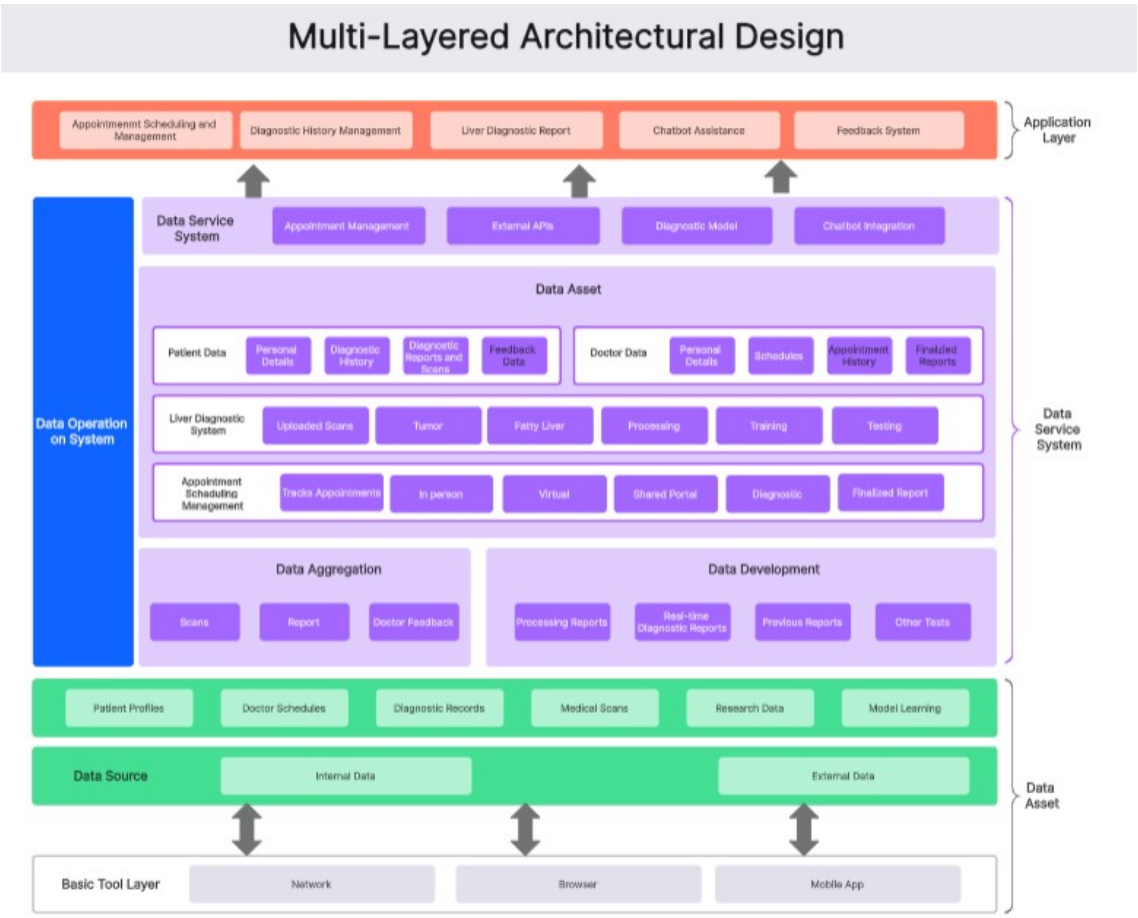


Figure 3.1: Architectural Design

3.2 Data Design

The Health Bridge system utilizes a MongoDB database to manage the various entities involved in the online appointment booking process, video consultations, AI-based disease detection, and automated report generation. MongoDB's flexibility and scalability are well-suited for managing the complex and dynamic nature of healthcare data, such as patient records, doctor schedules, and AI-generated reports. The major entities in the system include Appointments, Doctors, Patients, Payments, and Diagnosis. These entities are stored as collections in MongoDB, each representing a key part of the healthcare

interaction workflow. Below is a breakdown of the core entities, their relationships, and how they support the system's overall functionality.

3.2.1 Appointment Data

The **appointment** collection captures all patient-doctor appointment details. Key fields include:

- **appointmentid:** Unique identifier for the appointment.
- **doctorid:** Refers to the doctor assigned to the appointment.
- **patientid:** Refers to the patient booking the appointment.
- **astatus, aaccepted:** Manage appointment statuses (e.g., confirmed, canceled).
- **adate, atime:** Date and time for the appointment.
- **appointmenttype:** Type of consultation (e.g., "Regular Checkup" or "Video Consultation").
- **areminder:** Automated reminders for upcoming appointments.
- **paymentid:** Links to the payment details.

This entity interacts closely with the **Doctors** and **Patients** collections to ensure appointment scheduling and status updates. Video consultations are facilitated through a third-party integration that links to the appointment entity.

3.2.2 Doctor Data

The **doctor** collection stores essential details about the healthcare professionals in the system. Key fields include:

- **doctorid:** Unique identifier for each doctor.
- **name, email, phone, specialization:** Personal and professional details of the doctor.
- **dConsultFee, videoCallFacility:** Information regarding consultation fees and video consultation availability.
- **dOperatingDay, workingslots:** Track the doctor's availability across different days and times.

Doctors' profiles can include qualifications from the **qualification** collection and certifications from the **certification** collection, providing complete professional details. This data supports both online appointments and video consultations, ensuring accurate scheduling and professional validation.

3.2.3 Patient Data

The **patient** collection holds vital information about the patients:

- **patientid:** Unique identifier for each patient.
- **name, email, phone, address, city:** Personal details.
- **medicalHistory, allergies, medicationList:** Track the patient's medical conditions and history.
- **appointmenthistory:** List of past appointments with details on doctors consulted and services availed.

This data helps doctors provide personalized treatment and allows patients to manage their health records easily.

3.2.4 Prescription and Reports

The **prescription** collection links directly to appointments and doctors. It includes:

- **prescriptionid:** Unique identifier for prescriptions.
- **doctorid, patientid, appointmentid:** Links to the corresponding doctor, patient, and appointment.
- **prescriptionStatus, Prescription:** Details of the medications prescribed.
- **prescriptionCategory:** Specifies the medical category (e.g., Cardiology).

Similarly, the **report** collection stores AI-generated medical reports:

- **reportid:** Identifier for each medical report.
- **reportDocument, reportSummary, reportReviewer:** Stores the generated report document and review comments.
- **diseaseDetectionResult:** Stores results of AI-based scan analysis (e.g., liver disease detection based on uploaded X-rays).

These entities play a key role in automating healthcare delivery, supporting both in-person and virtual consultations, and providing diagnostic assistance through AI.

3.2.5 Payment Data

The **payment** collection handles all financial transactions in the system:

- **paymentid, paymentamount, paymentdate, paymentmethod:** Capture details of payments made by patients for appointments, lab tests, and other services.
- **appointmentid:** Links payments to specific appointments to ensure financial records are in sync with healthcare services provided.

3.2.6 Third-Party Integrations

HealthBridge integrates with third-party APIs to enable online video consultations between patients and doctors. These APIs are linked to the **appointment** collection, where the appointment type is marked as "Video Consultation" if it involves a video call. This integration ensures that online consultations are smooth and seamless, improving the overall user experience.

3.2.7 Diagnostic Model

A core feature of **HealthBridge** is its AI-enabled disease detection model, specifically designed for liver disease detection using MRI/CT scans. The patient uploads the scan through the system, which is then processed by the AI model. The results are stored in the **report** collection under the field *AI disease detection result*. This data is further used for generating a detailed report for the patient.

3.2.8 Conclusion

In conclusion, the **HealthBridge** system leverages MongoDB's document-based structure to efficiently manage healthcare data. The data is organized into collections that reflect real-world entities such as appointments, doctors, patients, and payments. These data structures are further enhanced with AI and third-party integrations to provide a comprehensive and secure healthcare platform that supports both in-person and virtual consultations.

3.3 Domain Model

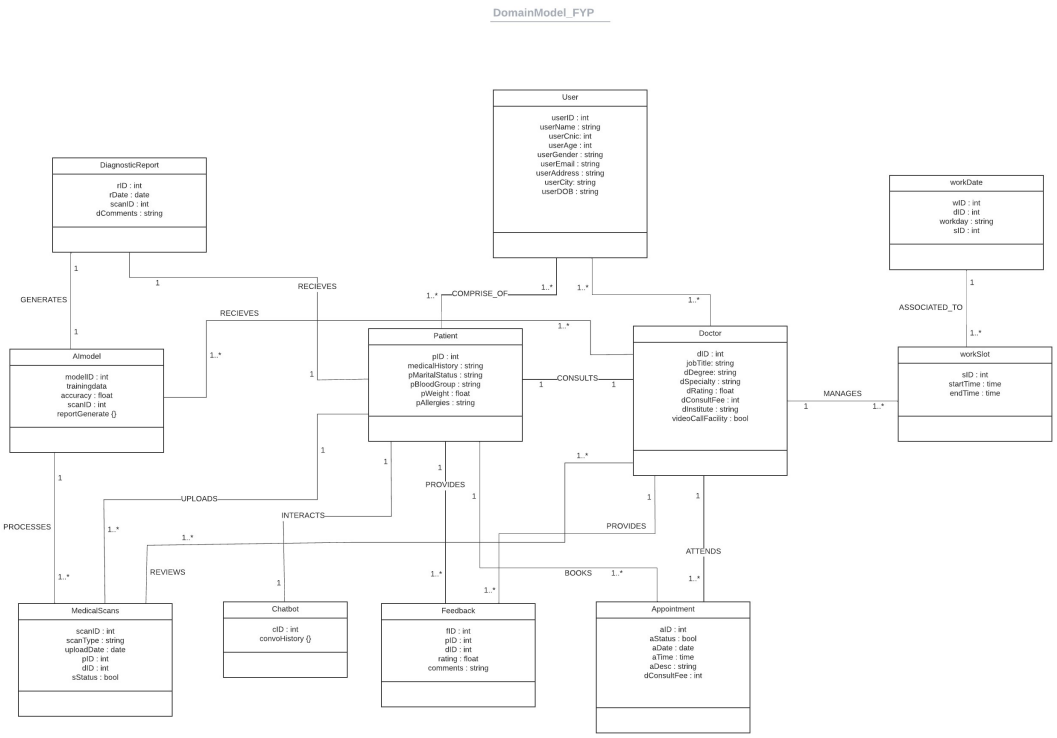


Figure 3.2: Domain Model

3.4 Design Models

3.4.1 Activity Diagram

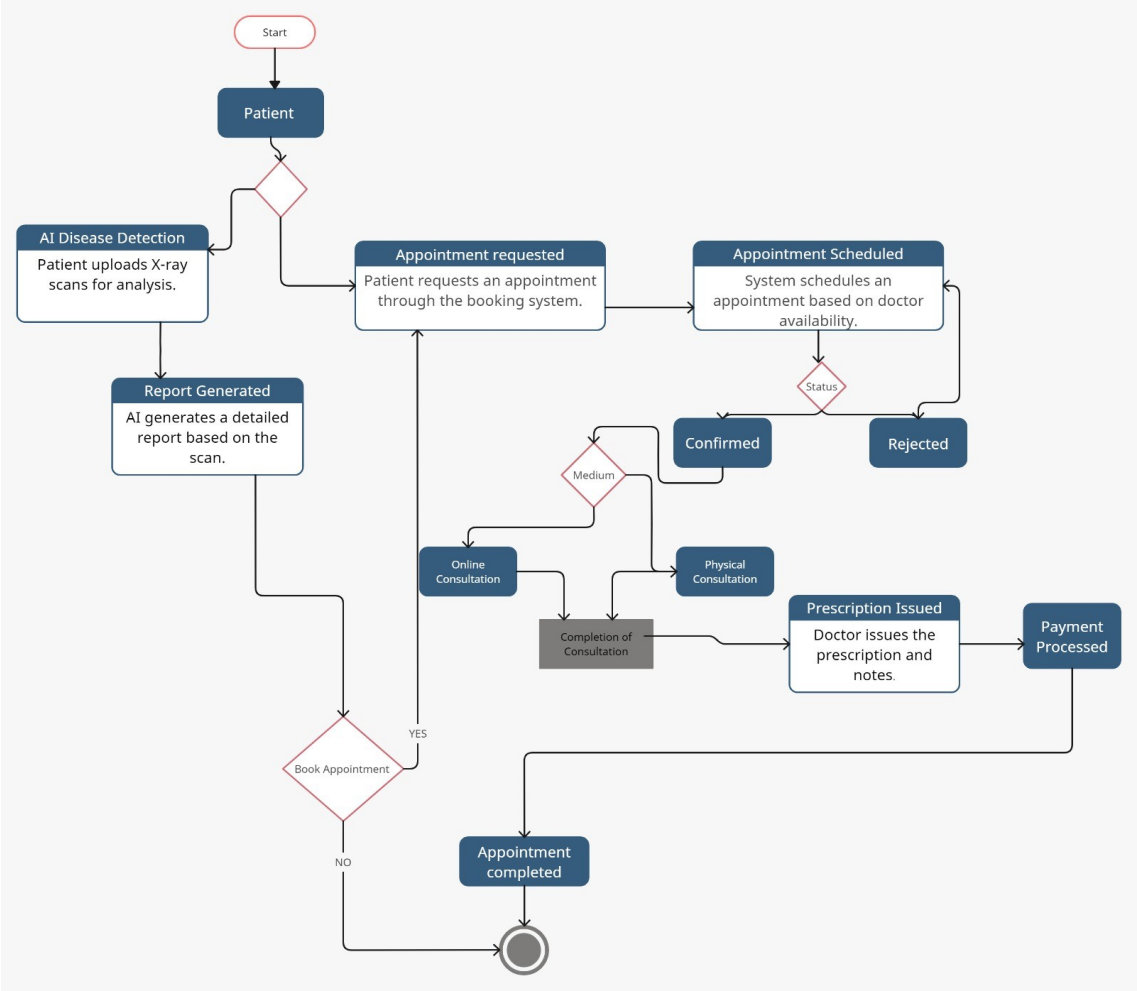


Figure 3.3: Activity Diagram

3.4.2 Data Flow Diagram

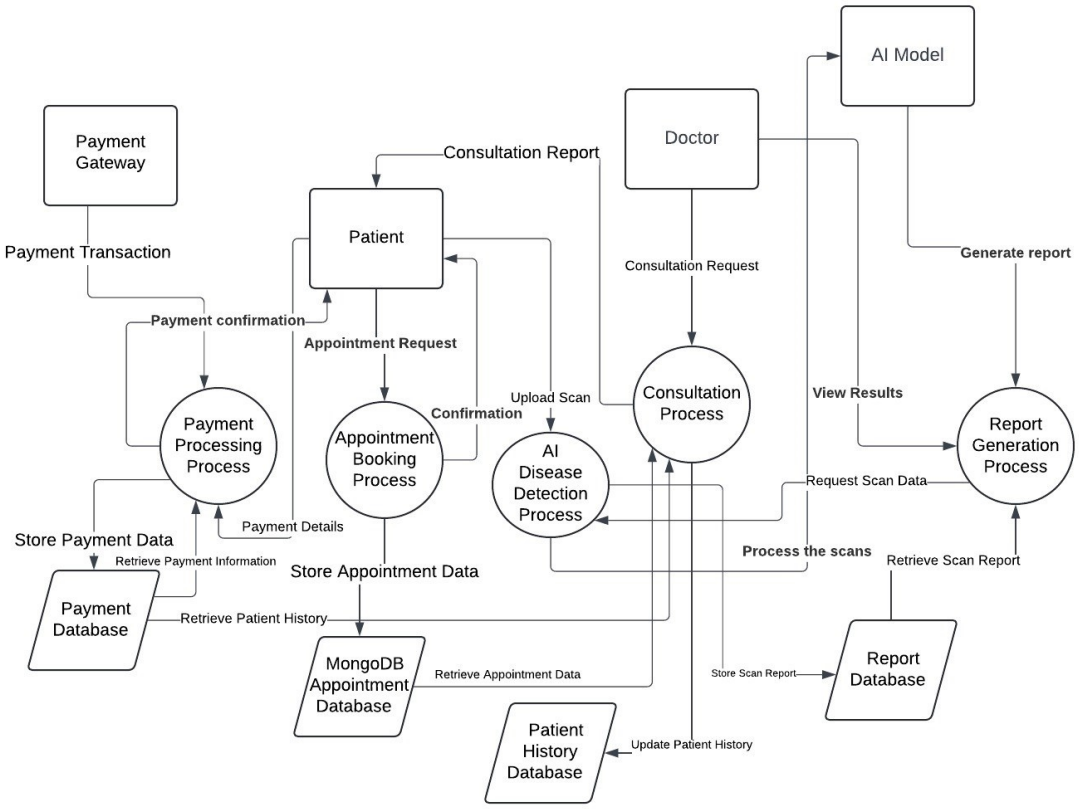


Figure 3.4: Data Flow Diagram

3.4.3 System-Level Sequence Diagrams

3.4.3.1 Login

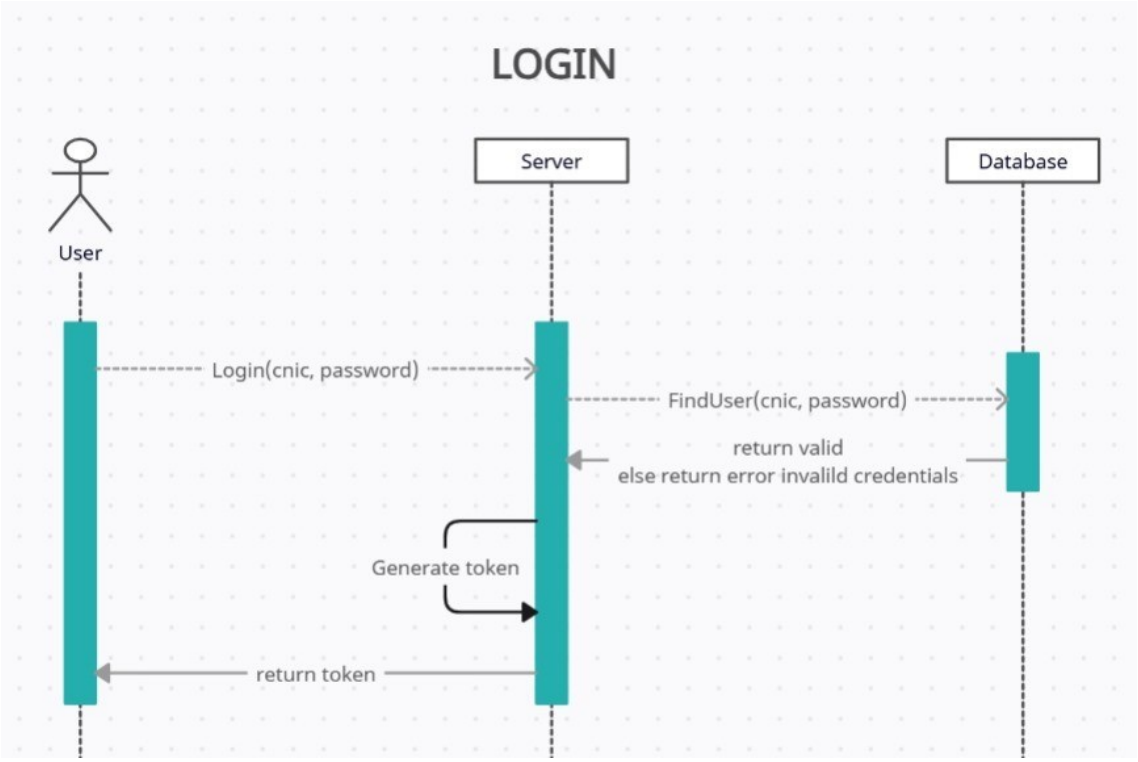


Figure 3.5: SSD-Login Diagram

3.4.3.2 Registration

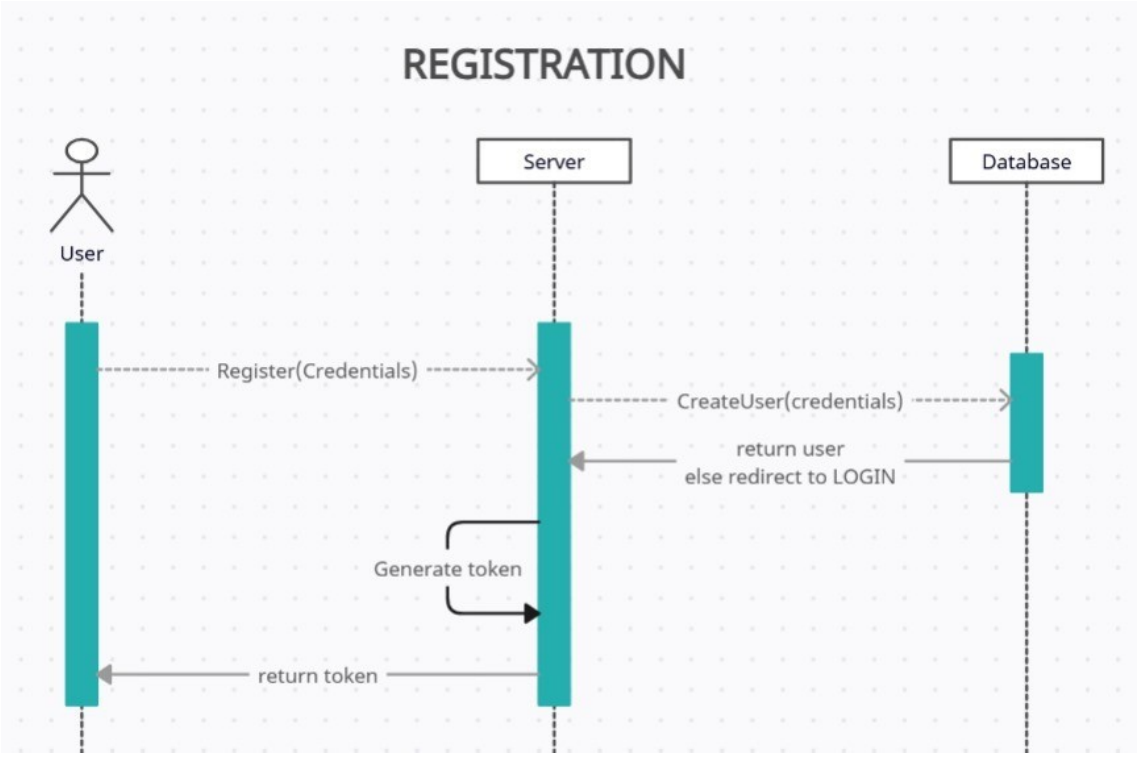


Figure 3.6: SSD-Registration Diagram

3.4.3.3 Give Feedback

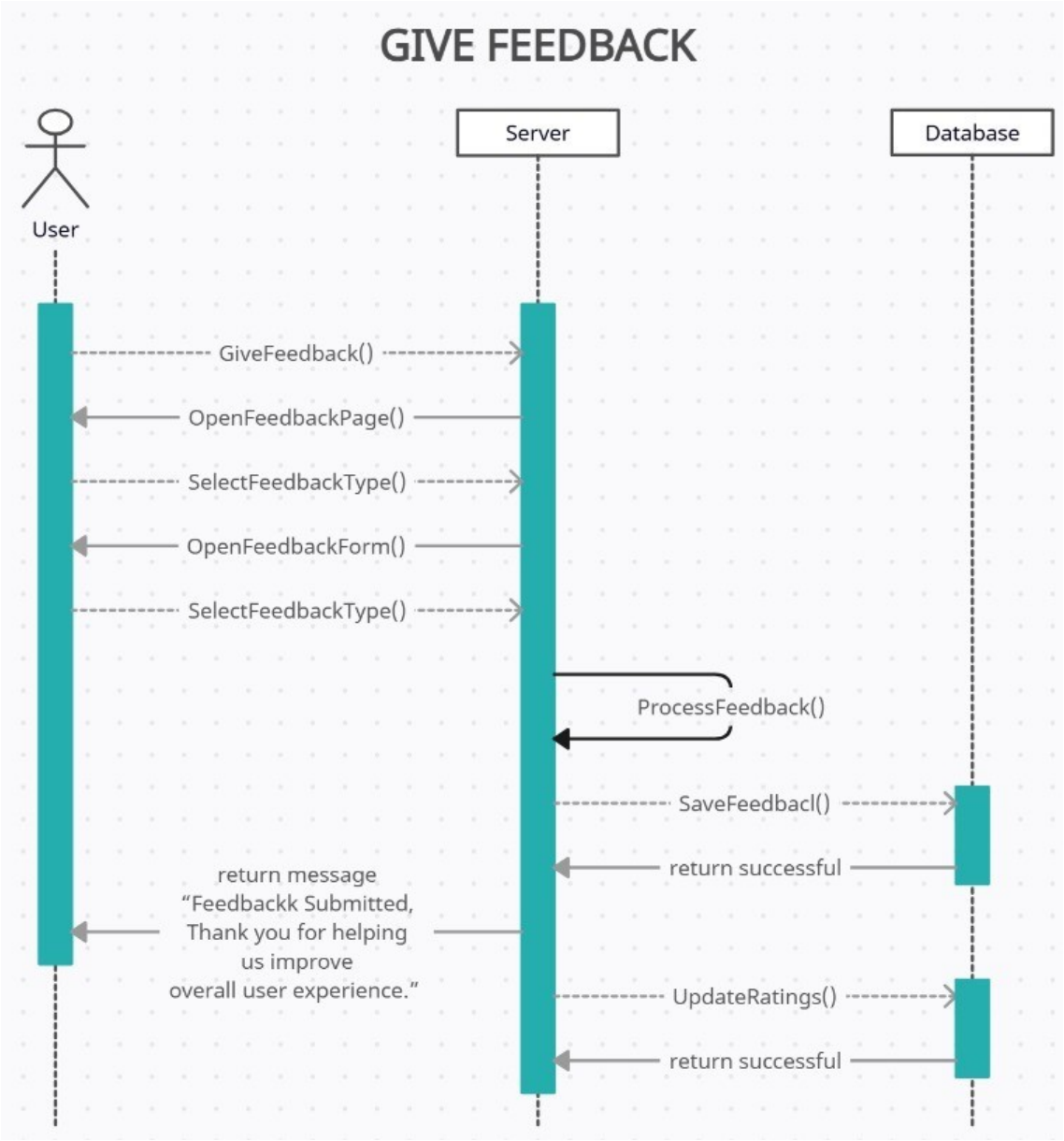
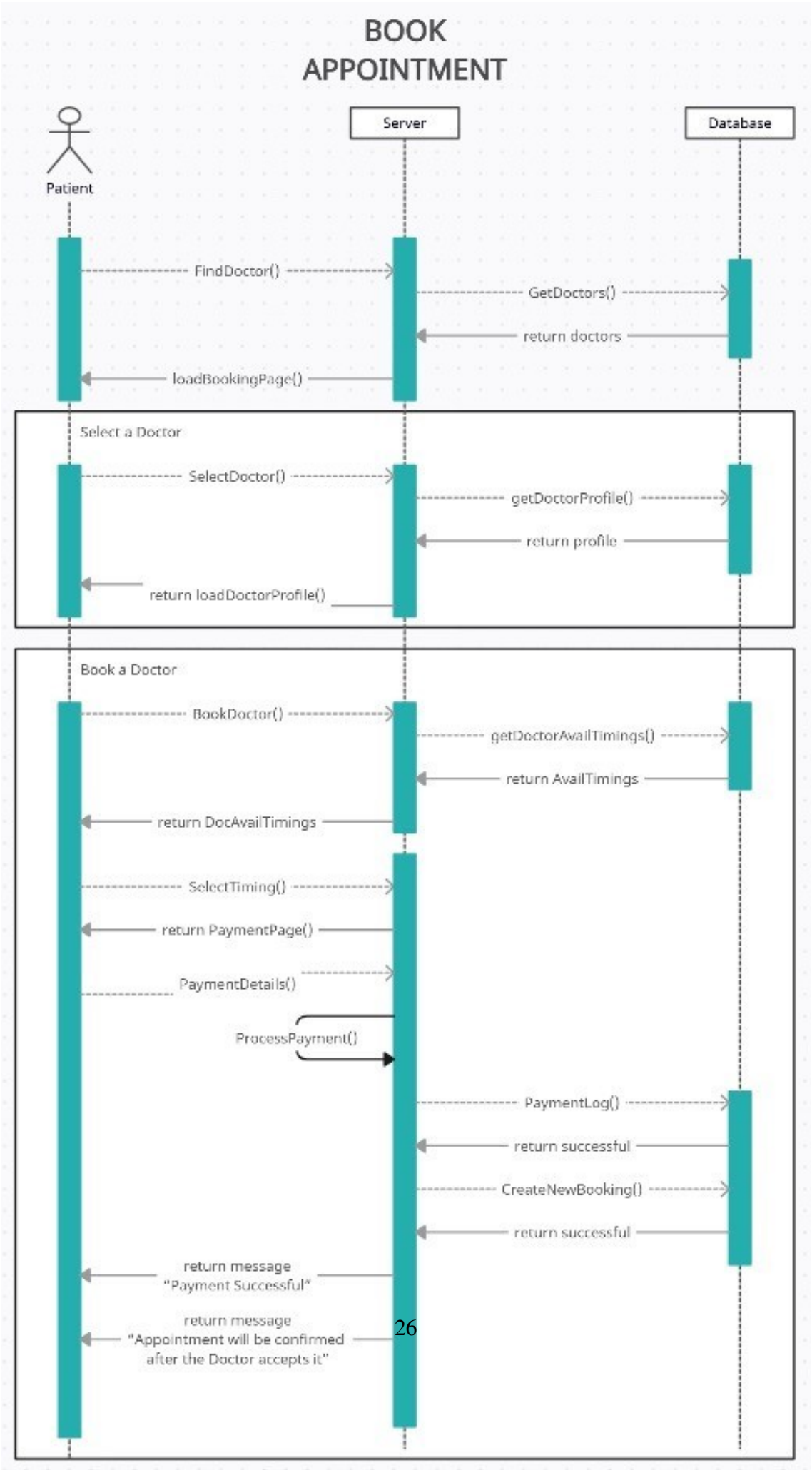


Figure 3.7: SSD-Give Feedback Diagram


BOOK
APPOINTMENT
Patient
Server
Database
FindDoctor()
GetDoctors()
return doctors
loadBookingPage()
Select a Doctor
SelectDoctor()
getDoctorProfile()
return profile
return loadDoctorProfile()
Book a Doctor
BookDoctor()
getDoctorAvailTimings()
return AvailTimings
return DocAvailTimings
SelectTiming()
return PaymentPage()
PaymentDetails()
ProcessPayment()
PaymentLog()
return successful
CreateNewBooking()
return successful
return message
"Payment Successful"
return message
"Appointment will be confirmed
after the Doctor accepts it"

3.4.3.4 Chat Assistant

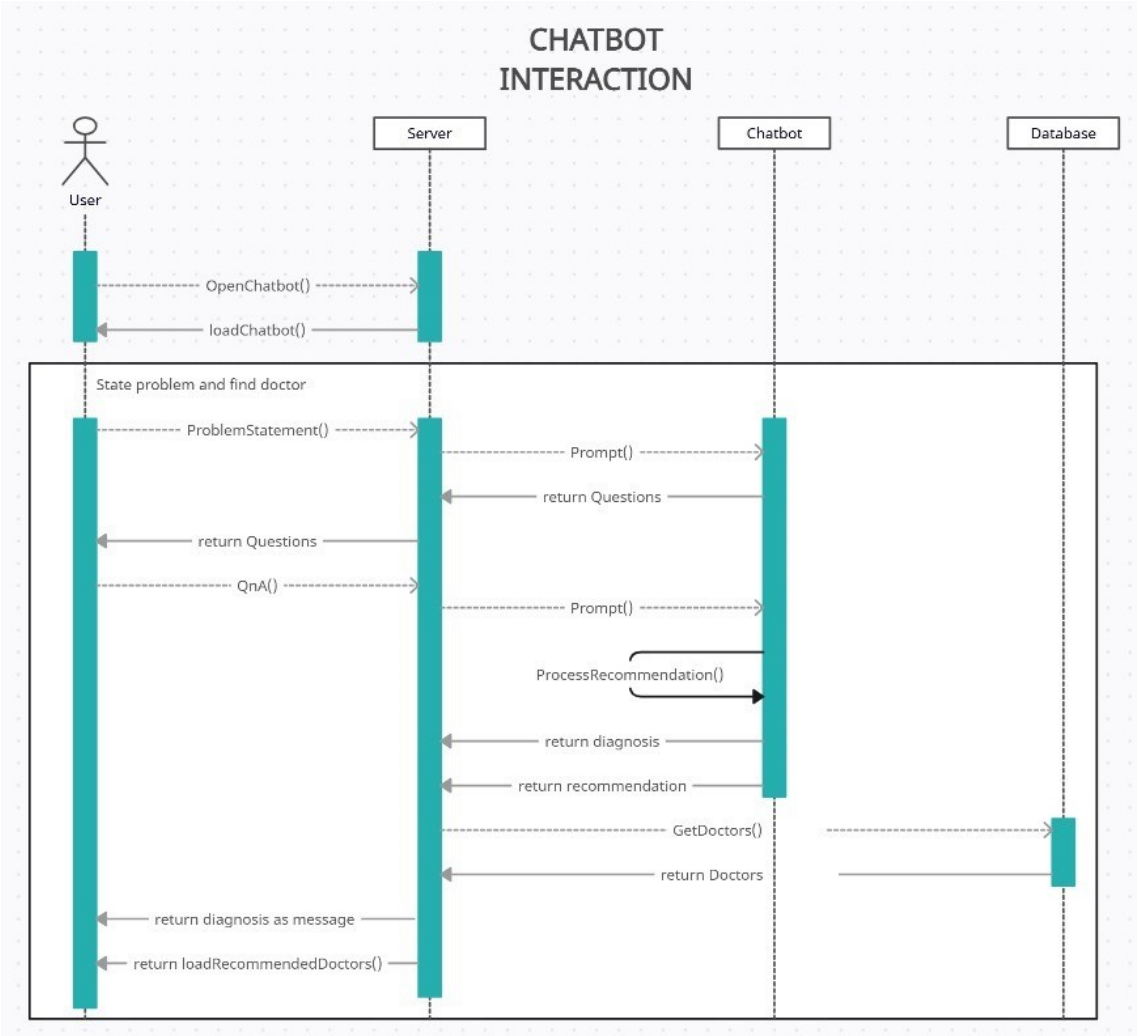


Figure 3.9: SSD-Chat Assistant

3.4.3.5 Diagnostic History

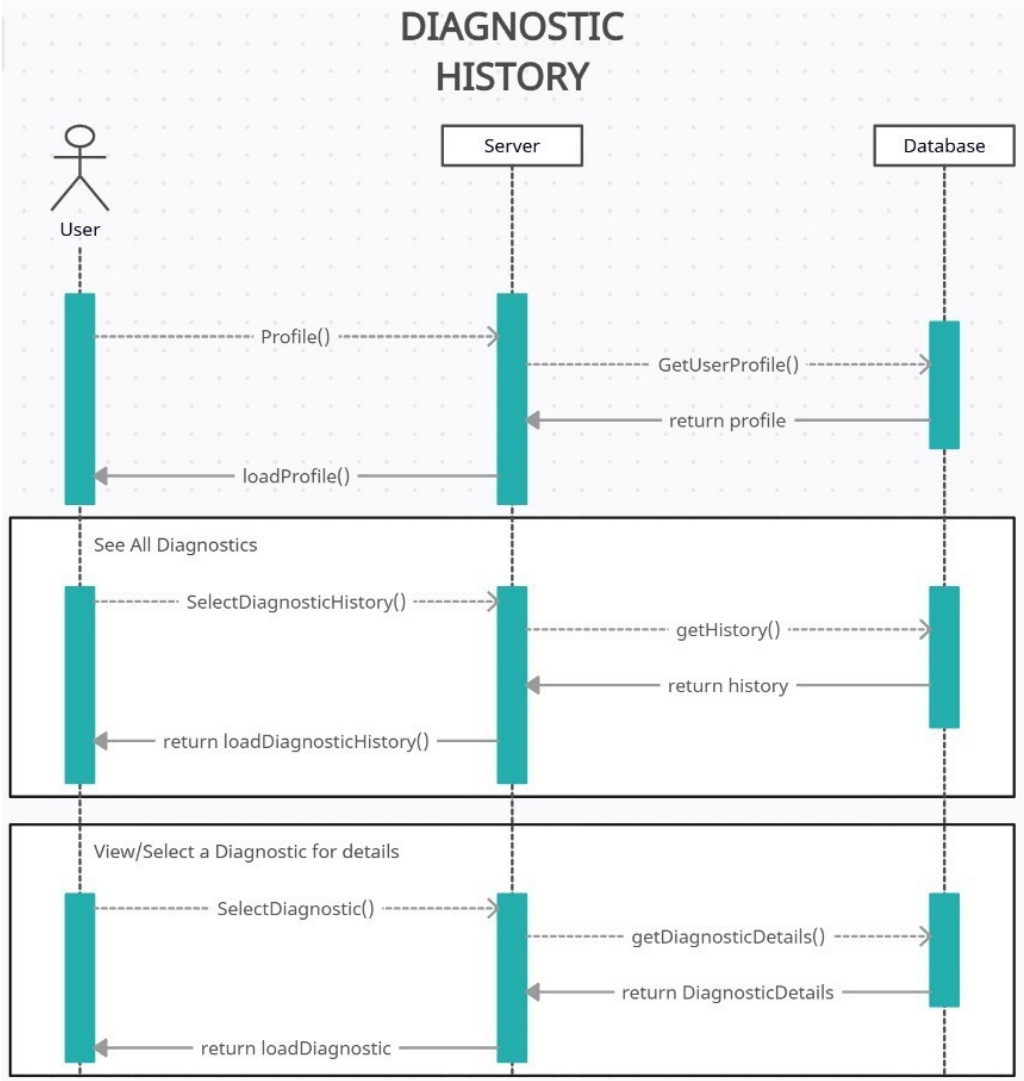


Figure 3.10: SSD-Diagnostic History

3.4.3.6 Manage Appointments

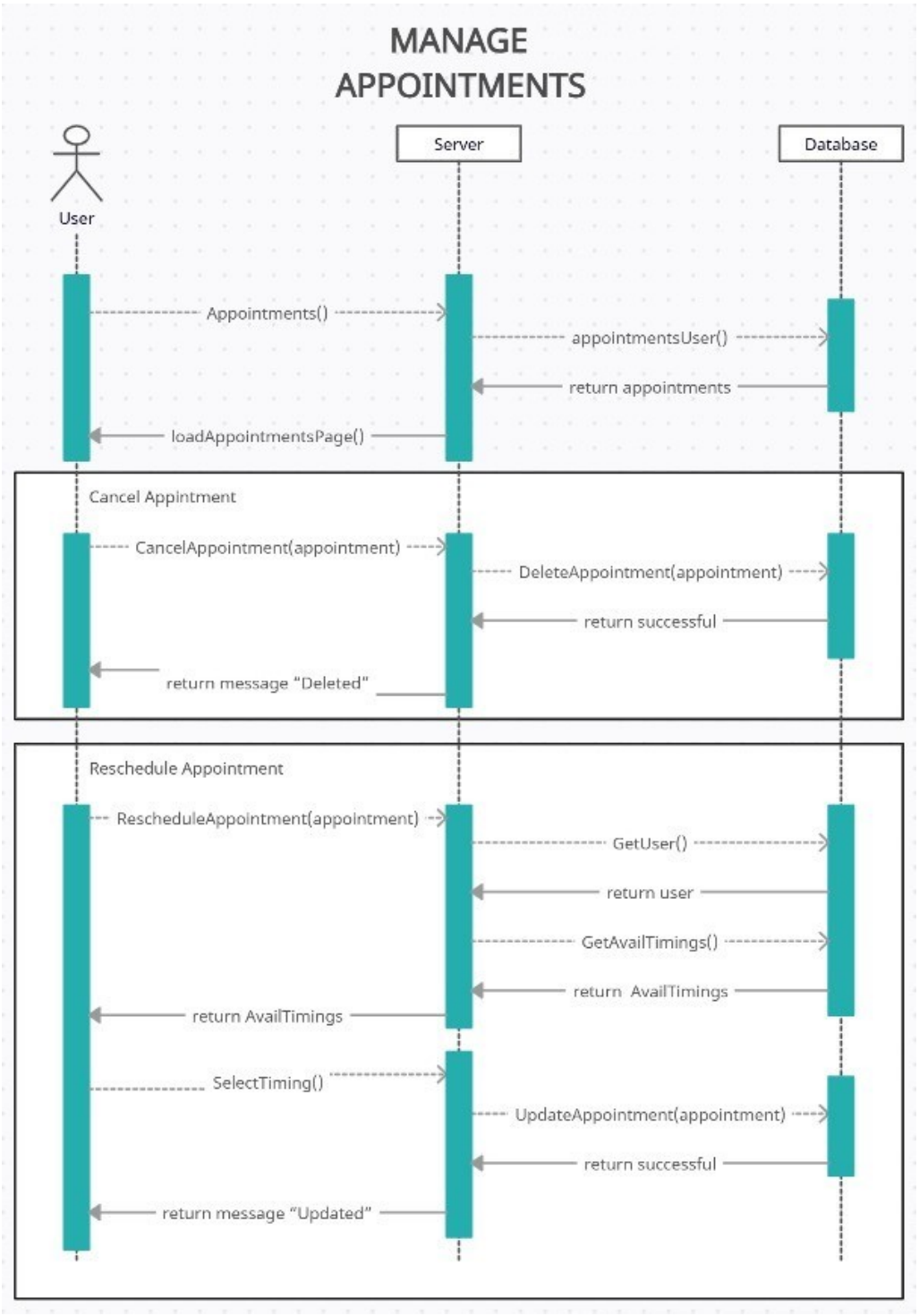


Figure 3.11: SSD-Manage Appointments

3.4.3.7 Upload Scans and Generate Report

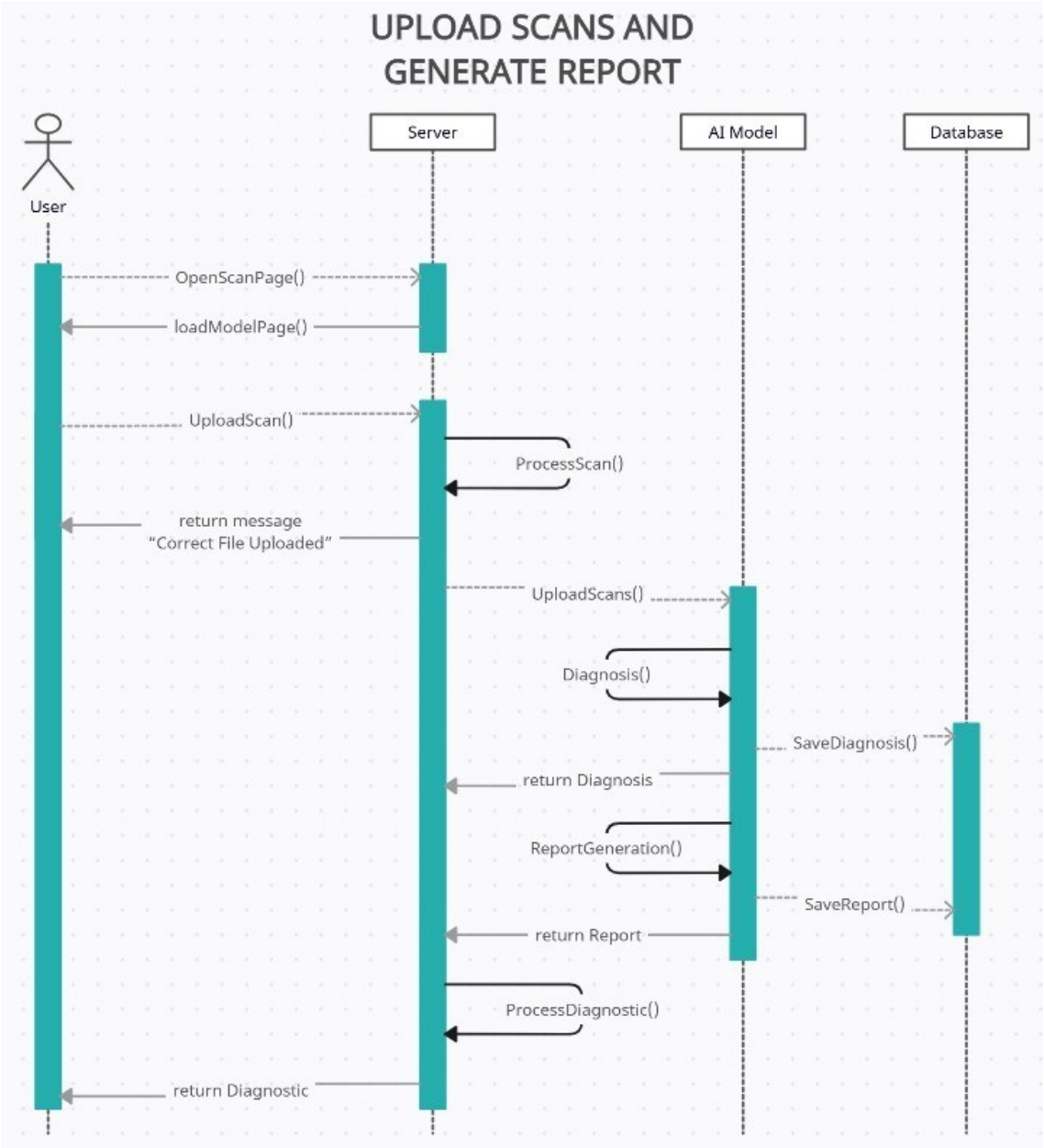


Figure 3.12: SSD-Upload Scans and Generate Report

3.4.4 State Transition Diagram

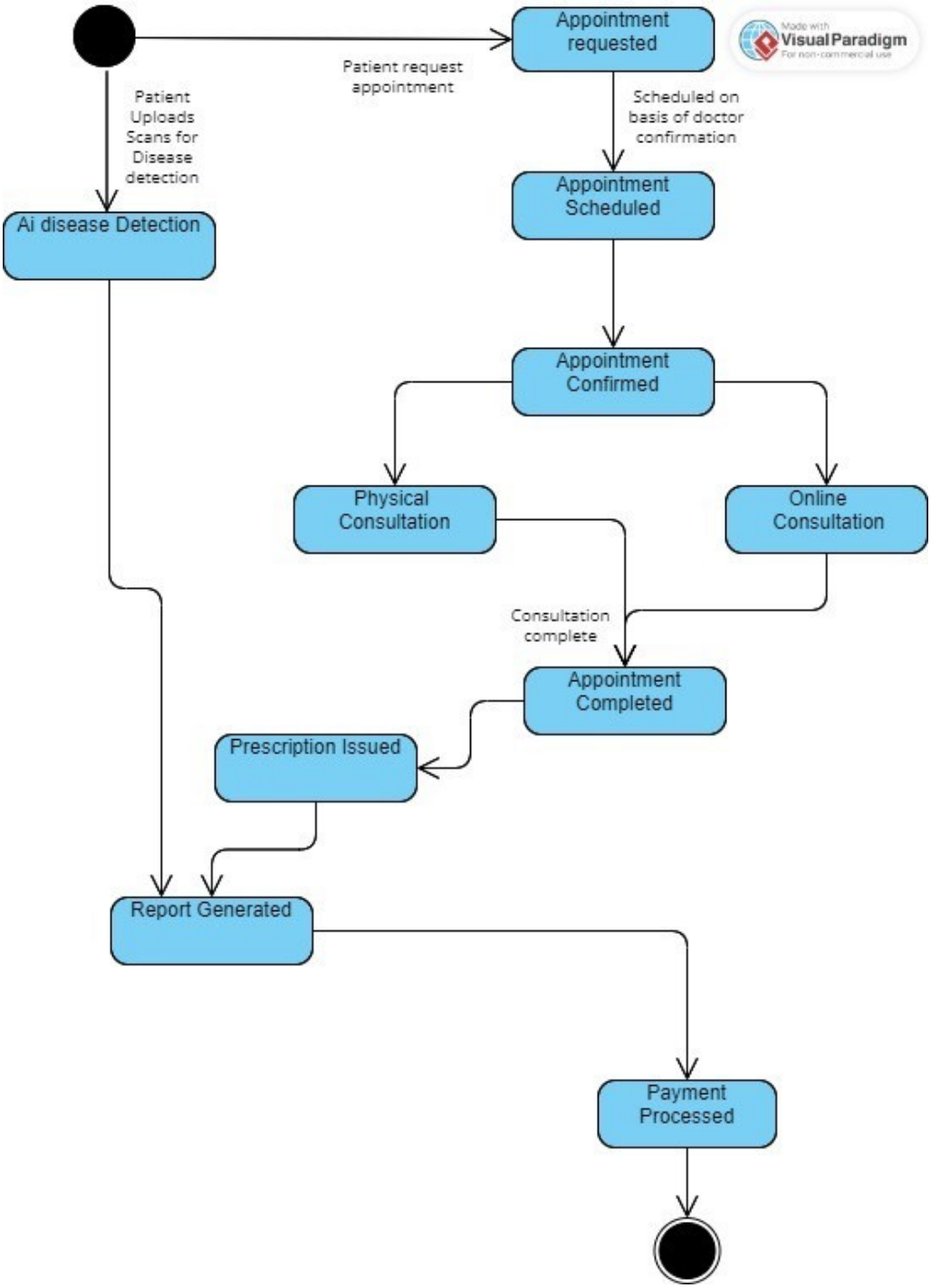


Figure 3.13: SSD-State Transition Diagram

Chapter 4

Implementation and Testing

4.1 Algorithm Design

4.1.1 Secure User Registration with Cryptographic Hashing

To ensure the secure storage of user credentials, this algorithm validates inputs, hashes passwords, and issues JWTs. The key details are summarized as follows:
Input: name, CNIC, password
Output: Securely hashed user credentials, JWT token
Example:

```
BEGIN
    IF NOT validateCnicFormat(cnic) THEN
        THROW Exception("Cnic Validation Failed: Invalid Format")
    END IF
    IF calculateEntropy(password) < MINIMUM_SECURITY_THRESHOLD THEN
        THROW Exception("Password Too Weak: Minimum entropy not met")
    END IF
    salt ← generateSalt(SALT_ROUNDS)
    hashedPassword ← bcrypt.hash(password, salt)
    userObject ← constructObject({name, cnic, hashedPassword})
    databaseResponse ← commitToDatabase(userObject)
    IF NOT databaseResponse.success THEN
        THROW Exception("Database Write Failed")
    END IF
    tokenPayload ← encode({userId: userObject.id}, JWT_SECRET)
    RETURN generateJWT(tokenPayload, EXPIRATION_TIME)
END
```

Figure 4.1: Secure User Registration with Cryptographic Hashing

4.1.2 Appointment Scheduling with Slot Validation

Dynamically handles appointment booking with real-time slot validation and conflict resolution.
Input: userId, docId, slotDate, slotTime
Output: Successful booking confirmation or conflict resolution failure
Example:

```
BEGIN
    doctor ← queryDoctorDatabase(docId)
    IF NOT doctor.available THEN
        RETURN "Error: Doctor currently unavailable"
    END IF
    slotMatrix ← doctor.slots_booked
    IF slotMatrix[slotDate][slotTime] EXISTS THEN
        RETURN "Conflict Detected: Slot already reserved"
    END IF
    mutex ← lockSlotUpdate(doctor.id, slotDate, slotTime)
    IF NOT mutex.acquired THEN
        RETURN "Concurrency Error: Slot modification locked"
    END IF
    appointmentEntry ← constructObject({userId, docId, slotDate, slotTime})
    databaseResponse ← commitToDatabase(appointmentEntry)
    Update slotMatrix[slotDate][slotTime] ← RESERVED
    Release mutex
    IF NOT databaseResponse.success THEN
        THROW Exception("Database Write Failed for Appointment")
    END IF
    RETURN "Appointment Successfully Scheduled"
END
```

Figure 4.2: Appointment Scheduling with Slot Validation

4.1.3 Dynamic Doctor Availability State Toggle

Flips a doctor's availability status with transactional safeguards for atomic updates.
Input: docId
Output: Doctor availability status flipped in the database
Example:

```
BEGIN
    doctorData ← fetchDoctorFromDatabase(docId)
    IF doctorData IS NULL THEN
        THROW Exception("Doctor Record Not Found")
    END IF
    transactionBegin ← initiateDatabaseTransaction()
    IF transactionBegin.failed THEN
        THROW Exception("Database Transaction Initiation Failed")
    END IF
    newAvailability ← !doctorData.available
    databaseUpdateResponse ← updateDatabase({id: docId, available: newAvailability})
    IF NOT databaseUpdateResponse.success THEN
        rollbackTransaction(transactionBegin)
        THROW Exception("Database Update Failed: Rolled Back")
    END IF
    commitTransaction(transactionBegin)
    RETURN "Availability Status Updated Successfully"
END
```

Figure 4.3: Dynamic Doctor Availability State Toggle

4.1.4 User Profile Update with Image Handling

Facilitates secure user profile updates with image uploads to Cloudinary.
Input: userId, newProfileData, imageFile
Output: Success or failure response
Example:

```
BEGIN
    user ← fetchUserById(userId)
    IF user IS NULL THEN
        THROW Exception("User Not Found")
    END IF
    IF imageFile EXISTS THEN
        cloudinaryResponse ← uploadToCloudinary(imageFile)
        newProfileData.image ← cloudinaryResponse.url
    END IF
    updateResponse ← updateDatabase({id: userId, data: newProfileData})
    IF NOT updateResponse.success THEN
        THROW Exception("Database Update Failed")
    END IF
    RETURN "Profile Updated Successfully"
END
```

Figure 4.4: User Profile Update with Image Handling

4.1.5 Text Messaging between Patient and Doctor

Facilitates real-time messaging between users by establishing a secure connection and ensuring message delivery through server acknowledgment.
Input: senderId, receiverId, messageContent

Output: Message delivery status (success or failure)
Example:

```
BEGIN
    socketConnection ← Establish connection to the messaging server using Socket.IO
    Authenticate sender by verifying senderId with the authentication service
    IF authenticationFails THEN
        RETURN "Authentication Failed"
    END IF

    receiverExists ← Check if receiverId is active in the system
    IF receiverExists IS FALSE THEN
        RETURN "Receiver Not Found"
    END IF

    messagePacket ← Create a structured packet containing:
        senderId
        receiverId
        messageContent
        timestamp ← Get current server time

    Emit messagePacket through the socket connection to the server
    deliveryStatus ← Wait for server acknowledgment of message delivery

    IF deliveryStatus IS "ACK_RECEIVED" THEN
        RETURN "Message Delivered Successfully"
    ELSE
        RETURN "Message Delivery Failed"
    END IF
END
```

Figure 4.5: Text Messaging between Patient and Doctor

4.2 External APIs/SDKs

Table 4.1: External APIs/SDKs Overview

API and Version	**Description**	**Purpose of Usage**	**API End-point/Function/Class Used**
Socket.IO	Real-time communication library	Enables real-time text, calls	socket.on, socket.emit
Cloudinary	Cloud-based media management	Managing profile images and medical reports	cloudinary.uploader.upload
MongoDB (Mongoose)	Data storage	Database to store platform data	mongoose.connect
JWT	Authentication library	Issues and verifies tokens for authentication	jwt.sign, jwt.verify

4.3 Testing Details

4.3.1 Unit Testing

Each unit test is designed to test a specific function or method independently from other components, helping to identify issues directly related to the functionality being tested.

Table 4.2: Unit Testing Table

Test Case ID	Test Objective	Pre-condition	Steps	Test Data	Expected Result	Post-condition	Actual Result	Pass/Fail
TC001	Verify user registration	User provides valid details	1. Enter *name, email, password*. 2. Click 'Register'.	Name: John Email: john@example.com Password: Pass@123	User account created and JWT token generated	User details are stored in the database	As expected	Pass
TC002	Verify user login	User is registered	1. Enter valid *email and password*. 2. Click 'Login'.	Email: john@example.com Password: Pass@123	Login successful, JWT token issued	User session starts	As expected	Pass
TC003	Verify appointment booking	Doctor is available	1. Select doctor and date/time slot. 2. Confirm booking.	Doctor ID: 123 Date: 2024-12-10 Time: 10:00 AM	Appointment booked successfully	Slot is reserved for the user	As expected	Pass
TC004	Verify image upload to Cloudinary	User uploads a valid image	1. Select image file. 2. Click 'Upload'.	File: profile.jpg	Image uploaded and URL returned	Image URL is stored in the database	As expected	Pass

Bibliography

[1] Biomed central, 2024.

[2] D-rax: Domain-specific radiologic assistant leveraging multi-modal data and expert model predictions. *arXiv*, 2024.

[3] The express tribune, 2024.

[4] Fact-aware multimodal retrieval augmentation for accurate medical radiology report generation. *arXiv*, 2024.

[5] Ihme: Institute for health metrics and evaluation, 2024.

[6] Multimodal healthcare ai: Identifying and designing clinically relevant vision-language applications for radiology. *ACM*, 2024.

[7] Pbs government website, 2024.

[8] Radio rag: Factual large language models for enhanced diagnostics in radiology using dynamic retrieval augmented generation. *arXiv*, 2024.

[9] Rule: Reliable multimodal rag for factuality in medical vision language models. *arXiv*, 2024.