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 Adil Ur Rehman

Co-Supervised by

Dr. Zeshan 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 [AS PER SCOPE DOCUMENT]

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	Pakistani digital healthcare service offering appointment booking and online consultations. Features a large network of verified doctors and user-friendly access to healthcare.	Geographically limited; limited features compared to global telehealth platforms; potential data privacy concerns in less regulated environments.
Marham	Digital healthcare platform in Pakistan enabling patients to search for doctors, book appointments, and access online consultations. It connects patients with verified specialists.	Limited to Pakistan; challenges in reaching rural areas; reliant on internet connectivity and user digital literacy.

Qure.ai	Develops AI tools for analyzing head CT scans and chest X-rays, focusing on detecting abnormalities like brain bleeds and tuberculosis. Enhances diagnostic speed and accuracy in radiology.	Focused mainly on CT scans and X-rays; adoption can be slow in resource-limited settings; potential errors in rare or complex cases; challenges in integrating with diverse workflows..
HeartFlow	Provides non-invasive coronary artery disease diagnosis using advanced imaging and computational fluid dynamics. Creates personalized 3D artery models for better diagnosis and treatment planning.	Restricted to coronary artery disease; limited accessibility due to high costs; requires specific imaging inputs like coronary CT angiograms.
Zebra Medical Vision	Offers AI-based analysis for medical imaging, focusing on breast cancer, brain bleeds, and fractures. It provides an all-in-one solution for radiology support and diagnosis.	Heavily reliant on high-quality imaging data; not all algorithms are FDA-approved; challenges in adoption across diverse healthcare systems.
Aidoc	AI-powered platform for medical imaging analysis, focusing on radiology, cardiology, and neurology. It focuses more on critical cases and integrates into workflows to improve diagnostic accuracy and speed.	Limited to imaging models and specific specialties; dependency on data quality; high cost of implementation; limited in handling atypical or rare cases.
Viz.ai	Specializes in stroke detection and rapid medical image analysis, facilitating communication between healthcare providers for timely treatment decisions.	Limited scope focusing primarily on stroke; high initial costs; dependent on integration with hospital systems and imaging technologies.

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, more particularly in regions like Pakistan especially for rural areas. Some of the key gaps include:

- AI-driven liver diagnostics for tumor and fatty liver detection with accurate results.
- Online Appointment scheduling and real-time consultations (chat system and video call) for better accessibility.
- Comprehensive support for patients especially in rural areas for enhanced healthcare on-time delivery.

By integrating such features in a single application, HealthBridge not only addresses such limitations that already exist in these applications, but also enhances the 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.

The following is a literature review table, which shows information about research papers that we used to identify more about the algorithms and methods used in research. We intent to practically implement these algorithms in HealthBridge.

Table 1.1: Summary of Multimodal AI Healthcare Research

Reference	Year	Objective	Model	Key Findings	Limitations
Multimodal for Healthcare	2024	AI diagnostics in medical imaging	RAG (Retrieval Augmented Generation)	Improved diagnostic accuracy	Limited dataset diversity and potential overfitting
Enhanced Diagnostics in Radiology	2022	Multimodal data integration in diagnostics	Multimodal Models	Enhanced diagnostic capabilities	High computational costs and complexity in model training
Multimodal RAG for Medical Radiology	2021	Large Language models in clinical support	GPT 3.5	Relevant medical text generation	Heavy reliance on fine-tuning and risk of inaccuracies
Radiologic Assistant model prediction	2023	Fact-aware Generation in radiology reports	Fact MM-RAG	Improved factual accuracy in reports	Dependence on retrieval systems, potential outdated information
Medical vision Language Models	2023	Few short learning in clinical applications	Med-Flamingo	Effective with minimal data	Limited by initial data quality and struggles with specialized knowledge

1.4 Modules

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 Pages: All pages related to searching, booking, communicating and using AI Model.
5. Doctor Pages: All pages related to booking, communicating and using AI Model.
6. Institute Pages: Pages only meant for various institutes, which will only use our AI Model.

1.4.2 Module 2: User Management System (Backend)

Design and implementation of User Management System with connectivity to Web Application and Chatbot.

1. Design Database: Designing our projects database, with ERDs, and relational schemas.
2. Implementation: The designed database will come into existence.
3. Backend APIs: APIs will be made for our project and be used later to connect our webapp with the server.

1.4.3 Module 3: Chatbot

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

1. Chatbot Page Design: Design a single page for the Chatbot, which will flow like a conversation until a result can be concluded.
2. Connecting Chatbot: Our Chatbot will be fine-tuned and connected to our backend.

2. Project Description

1.4.4 Module 4: Model Development

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.

1.4.5 Module 5: Android Development and System Integration

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. Integration: All our modules which include different modules and their integration into one will be strengthened.

1.5 Work Division

Incorporating Horizontal Work Division, we have divided our work in the following way:

Table 2.1: Work Division

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

Chapter 2

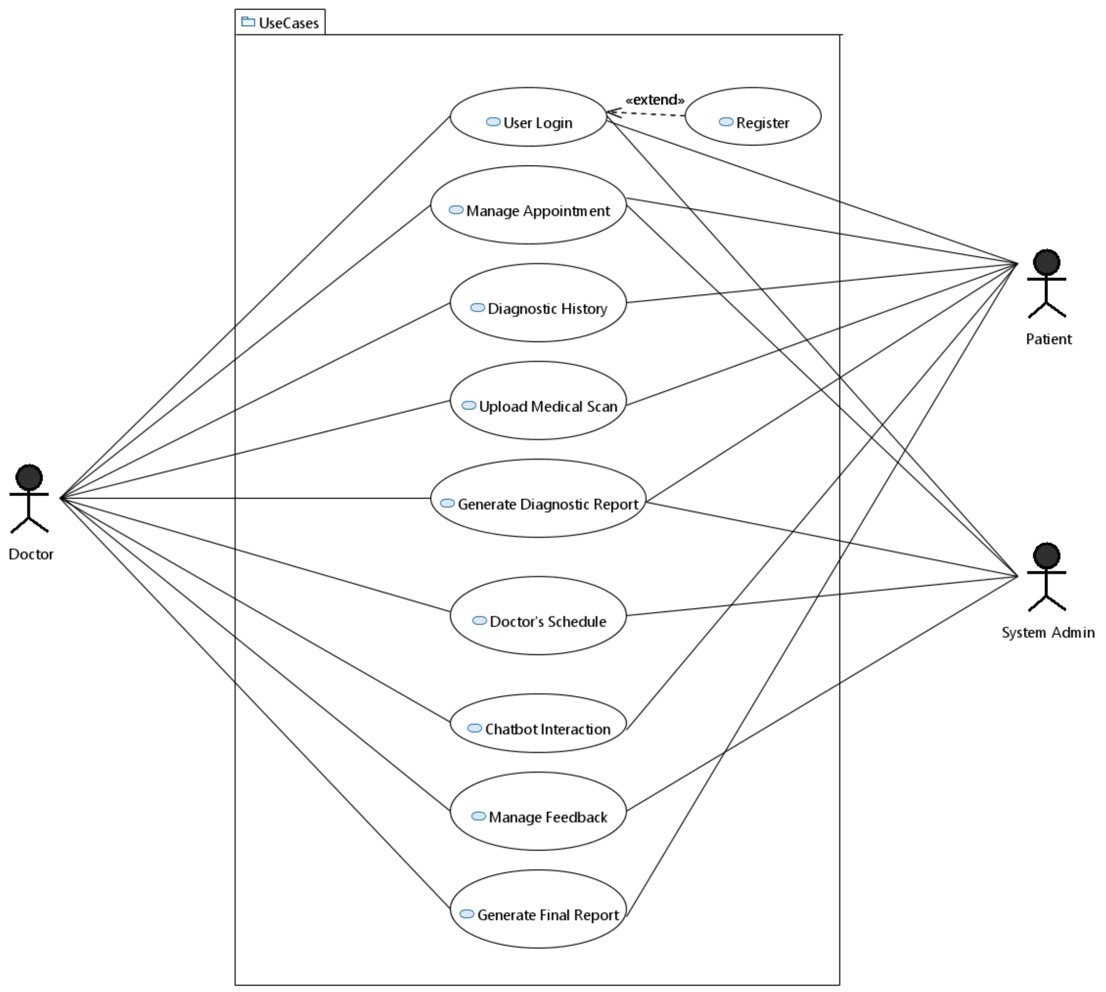
Project Requirements [AS PER FYP1 MID REPORT]

This section outlines the necessary requirements for the successful completion of the project. Project requirements can be divided into two main categories: functional requirements, which describe the system's core operations, and non-functional requirements, which specify performance, security, and usability standards.

2.1 Use-case/Event Response Table/Storyboarding

This section describes how users will interact with the system through various scenarios, often referred to as use cases or event responses.
A use case represents a specific interaction between the user and the system, detailing the steps involved and the expected system responses. Each use case typically includes key components such as a unique identifier, a description of the interaction, the user or system actor involved, preconditions, the main flow of events, and postconditions.
Additionally, storyboarding can be used as a visual aid to depict the sequence of actions, illustrating how the user progresses through the system step by step. Storyboards are helpful for capturing the user experience and ensuring that the design aligns with the intended functionality.

Depending on the project, use cases, storyboarding or Event-response table can be chosen as appropriate methodologies to illustrate user interactions. Examples of all the approaches are provided in Appendix A.



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 password reset.
- Special Requirements:
  1. The encrypted password is stored in the database for a secure management system.
- Technology and Data Variations:
  1. Input: CNIC and password
  2. Output: Success or failure message.

2.1.2 Use-case 2: Manage 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 navigate 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

2.1.3 Use-case 3: Diagnostic History

- Scope: HealthBridge System
- Level: User goal
- Primary Actor: Patient, Doctor, Institute Admin
- Stakeholders and Interests:
  1. Patient: Upload diagnostic reports for review.
  2. Doctors: They are required to review diagnostic histories of patient for accurate consultations.
  3. Institute Admin: For history of previous reports.
- Pre-condition:
  1. All users are successfully logged in.
  2. Doctors and Institute admin can view the uploaded medical history reports of patients.
- Post-condition:
  1. Both doctors and patients can review all the details given in the uploaded medical reports.
- Main success scenario:
  1. User selects the ‘Diagnostic History’ from their dashboard.
  2. The user uploads all the medical history reports.
  3. The system ensures the uploaded files can be viewed from both doctors and institute admin.
- Extensions:
  1. The system notifies the user in case of no medical history records.
  2. Error message in case of incorrect file path of uploaded report.
- Special Requirements:

1. Secure storage of medical history reports.

- Technology and Data Variations:
  1. Previous medical reports

2.1.4 Use-case 4: Upload Medical Scans

- Scope: HealthBridge System
- Level: User goal
- Primary Actor: Patient, Doctor
- Stakeholders and Interests:
  1. Patient: Upload scans for a diagnostic report.
  2. Doctors: Upload scans or analyze the diagnostic report for an accurate final report generation.
  3. System Admin: For patient history in case of future appointments.
- Pre-condition:
  1. User is successfully logged in.
  2. User has the medical scans in the correct format for uploading.
- Post-condition:
  1. The AI processes the uploaded scans and provides an accurate diagnostic report for the scans.
- Main success scenario:
  1. User selects the ‘Upload Medical Scan’ from the respective page.
  2. The user chooses the scan files uploads.
  3. The system successfully verifies the uploaded to ensure they are in the correct file format.
  4. The AI model processes the scans and generates the automated diagnostic report.
  5. Both patient and doctor can view the diagnostic report.
- Extensions:
  1. Error message if file format is not supported.

- Special Requirements:
  1. Secure file upload for diagnostic report.
- Technology and Data Variations:
  1. Medical scan files (dicom, nii)
  2. Various image formats for scan

2.1.5 Use-case 5: Generate Diagnostic Report

- Scope: HealthBridge System
- Level: User goal
- Primary Actor: Patient, Doctor
- Stakeholders and Interests:
  1. Patient: Quick and accurate diagnostics for uploaded scans.
  2. Doctors: Analyzes the diagnostic report of AI model for diagnosis and recommended treatment procedure.
  3. System Admin: Checks the overall accuracy and efficiency of the complete process.
- Pre-condition:
  1. The patient has uploaded the medical scan with the correct file format successfully.
- Post-condition:
  1. The AI model generates a diagnostic report based on the uploaded scans that the user can now download.
- Main success scenario:
  1. The AI model receives the medical scans.
  2. Using pre-trained models for liver diagnosis, the AI model processes the uploaded medical scans.
  3. The AI models then generates an accurate diagnostic report after processing.
  4. The report is sent to the patient and doctor.
- Extensions:

1. The system notifies the patient and doctor in case of any troubleshoot or error.

- Special Requirements:

1. Highly accurate AI model for liver diagnosis.

- Technology and Data Variations:

1. Scanned medical data
2. Diagnostic report

2.1.6 Use-case 6: Doctor's Schedule

- Scope: HealthBridge System
- Level: User goal
- Primary Actor: Patient, Doctor
- Stakeholders and Interests:

1. Doctor: Updates the weekly/monthly schedule for available slots with date and time.
2. Patient: Real-time availability of slots for booking an appointment.
3. System Admin: Makes sure that the overall system runs smoothly without any errors and timely update of slots.

- Pre-condition:

1. The doctor can edit their available slots and schedule of the day.
2. The patient can view the available slots with time for the relevant doctors.

- Post-condition:

1. The doctors schedule and availability is updated for the appointment.

- Main success scenario:

1. The doctor selects 'Manage Schedule' option from the respective page.
2. The doctor updates their schedule of the day along with time slots.
3. The doctor saves changes and the system updates the complete schedule.

- Extensions:

1. If the doctor tries to update an already scheduled appointment after approving the appointment request, the system notifies them and does not update the timeslot for the already scheduled appointment.
2. In case of any error for updating schedule, the system suggests alternative slots.

- Special Requirements:

1. Real-time update for appointment scheduling.

- Technology and Data Variations:

1. Available date and time for the slots.
2. Schedule update.

2.1.7 Use-case 7: Chatbot Interaction

- Scope: HealthBridge System
- Level: User goal
- Primary Actor: Patient, Doctor
- Stakeholders and Interests:

1. Patients: Use of Chatbot for getting a generic report for the mentioned problem in order to get recommended doctors or a diagnostic report for medical scans.
2. Doctors: Can use Chatbot for additional analysis of diagnostic report or report generation.
3. System Admin: Maintains Chatbot for smooth interaction.

- Pre-condition:

1. The user is logged in and navigates to Chatbot to start a conversation.

- Post-condition:

1. The Chatbot assists the user in identifying the problem and recommends relevant doctors on the app.

- Main success scenario:

1. The patient selects the option for ‘Chatbot’ interaction from the dashboard.
2. The patient is navigated to the Chatbot page and prompts their relevant query.

3. The Chatbot processes the request and returns the message that is relevant to the patient’s query. 4. The patient understands the given response and the task is completed.

- Extensions:

1. If the system delays or stops working, the Chatbot notifies the patient.

- Special Requirements:

1. The Chatbot must give a response which is relevant to the user query/question.
2. The Chatbot must respond in real-time.

- Technology and Data Variations:

1. User prompt
2. Chatbot response

2.1.8 Use-case 8: Manage Feedback

- Scope: HealthBridge System
- Level: User goal
- Primary Actor: Patient, Doctor, System Admin
- Stakeholders and Interests:

1. Patients: They can provide feedback for the app and doctors ratings.
2. Doctors: Each individual can give their feedback over each appointment on the app.
3. System Admin: Analyzes the feedback for future improvements.

- Pre-condition:

1. The users are successfully registered and logged in to the app.
2. The users can use the app for diagnostic reports and appointments.

- Post-condition:

1. The users have used the application for diagnostic report and appointments and can give the overall feedback of the system.

- Main success scenario:

1. The user logs into the app.

2. The patient selects the option of “Provide Feedback”.
3. The patient gives rating for their respective doctor after the appointment.
4. Similarly, the doctor also provides a detailed feedback for the overall system after the appointment.
5. The system saves the feedback after the submission.

- Extensions:

1. The user is notified if the system fails to save the feedback due to any troubleshooting issues.

- Special Requirements:

1. The system should be able to save the information in the database for each user.

- Technology and Data Variations:

1. Ratings and comments.
2. Confirmation message for submission.

2.1.9 Use-case 9: Generate Final Report

- Scope: HealthBridge System
- Level: User goal
- Primary Actor: Patient, Doctor, AI Model
- Stakeholders and Interests:

1. Patient: Receives a final report based on doctor’s complete analysis assisted by AI generated diagnostic report.
2. Doctor: Uses the AI based automated report for final diagnosis for recommended treatment or prescription.
3. AI Model: Provides a diagnostic report to assist doctor for their finalized report.

- Pre-condition:

1. The patient uploads the medical scans.
2. The model processes the uploaded scans and provides an initial diagnostic report.

- Post-condition:

1. The final report is generated by the doctor with the assistance of the initial diagnostic report of scans. The patient can view the report and download it.

- Main success scenario:

1. The model provides and initial diagnostic report based on the uploaded scans.
2. The doctor analyzes the diagnostic report based on the scans.
3. After analyzing the diagnostic report in detail along with the recommended tests, the doctor will then generate a final report based on the patient details along with the description of the diagnosed problem and the relevant steps for the treatment.
4. The patient will be notified when the final report will be generated. They can proceed to view it or download the finalized report if necessary.

- Extensions:

1. The diagnostic report generated by the model is incomplete.
2. The diagnostic report has ambiguities.
3. If the model based report is not generated completely, the system will restart it.

- Special Requirements:

1. The final report must consist of all the details of the respective patient, the diagnosed problem according to the scans and tests in detail, and the recommended treatment signed by doctor.

- Technology and Data Variations:

1. Medical scans.
2. Doctors detailed analysis of the diagnosed problem.
3. Final report.

Table 2.1: System Events and Responses

Event	System Response
New user registers for the first time	1. User enters all the required details for registration. 2. The system verifies whether the information is correct with all constraints. 3. The system sends an email verification for authentication.

	4. The user is successfully registered with the given information.
User logs in with valid credentials	1. The system validates the entered credentials. 2. The system authorizes the user to log in and the user is navigated to their respective dashboard.
User logs in with invalid credentials	1. The system shows an error message for invalid credentials. 2. The user can select the option of password reset if necessary.
User requests for password reset	1. The system sends a password reset link/code on the registered email. 2. The user is required to check their email for updating the password securely and successfully.
Patient books an appointment	1. The patient searches for relevant doctors. 2. The system navigates the patients to the searched doctors page based on the selected option (specialty, rating, region). 3. The patient selects the preferred doctor. 4. The system displays the available timeslots of each day for the specific doctor. 5. The patient selects the timeslot for the appointment with the relevant doctor. 6. The doctor chooses the option for accepting or rejecting the pending appointment. 7. The system saves the information and navigates the patient to the payment page. 8. The patient proceeds to the payment page and enters payment details. 9. The system notifies both users for the confirmation of the appointment.
Doctor accepts/rejects the appointment	1. The system updates the timeslots based on doctor's action in real-time. 2. Both users are notified with the updated status of the appointment.
Patient/Doctor cancels or reschedules appointment	1. The system allows both users to cancel or reschedule the appointment within 24 hours after the confirmation of the appointment. 2. The system updates the schedule and slots of doctor and notifies both end users.

Patient uploads medical history	1. The patients can upload previous medical history reports and tests that the doctor can access after the confirmation of the appointment for future reference.

Table 2.2: System Events and Responses (2)

Patient uploads medical scans	1. The patient selects the option for interacting with the diagnostic model. 2. The patient uploads the relevant liver scans for diagnosis with the correct file format. 3. The system validates the file format of the uploaded scans. 4. The model processes the uploaded scans and generates a diagnostic report. 5. Both users can view the report after the completion of the process.
Diagnostic model generates diagnostic report	1. The diagnostic model processes the information and generates a diagnostic report for the uploaded scans. 2. The user is notified after the completion of the report.
Doctor analyzes diagnostic report	1. The doctor analyzes the report generated by the diagnostic model and either confirms the diagnosis or recommends further tests/scans. 2. The system allows the doctor to add their comments for the diagnosis in the generated report.
Patient views/downloads diagnostic report	1. The patient can view the diagnostic report and the comments added by the doctor after the report analysis. 2. The system ensures that the report is accessible from the patient's dashboard.
User interacts with Chatbot	1. The Chatbot responds to the user inquiries, such as the problem explained, recommending the available doctors, or a diagnostic report. 2. The system processes the request in real-time and provides the appropriate response.
Doctor generates finalized patient report	1. After a detailed analysis of the report generated by the diagnostic model, the doctor finalizes the report which comprises of the complete details of the patient, description of the diagnosed problem, and the recommended treatment plan. 2. The system notifies the patient once the finalized report is completed and uploaded by the doctor. 3. The patient can view and download the final report.
User provides feedback	1. The system allows users to provide feedback after using the application. 2. The patient can provide feedback after the appointment with the doctor. 3. The feedback and rating is stored in the database for analysis and for future improvement. 4. The doctors can provide feedback for the overall provided system.

Doctor updates the schedule	1. The doctor can update their available timeslots for the complete schedule of a week. The doctor clicks on “Save Changes”. 2. The system updates the complete schedule of the respective doctor with the timeslots for each day. 3. The system reflects these changes onto the patient’s side for the future appointments.19
User stays inactive for 15 minutes	1. The system automatically logs out the user if the user stays inactive for more than 15 minutes. 2. A message is displayed to log in again for every user.

2.2 Functional Requirements

2.2.1 Module 1: Web Application

1. The user is successfully able to register and login to the platform.
2. The user is properly authenticated before being able to access the platform.
3. The user's respective pages have all use cases implemented.

2.2.2 Module 2: User Management System

1. The database communicates with the web application.
2. APIs cannot be accessed outside of our web application.

2.2.3 Module 3: Chatbot

1. The chatbot responds according to the user's prompt.
2. The chatbot successfully communicates with the user, as well as our backend server.
3. The chatbot accurately suggests recommended doctors.
4. The chatbot can process various formats of the JSON body, since we will later integrate this chatbot with our AI model.

2.2.4 Module 4: Model Development

1. Datasets are correctly labeled.
2. The model returns the result in 2 different formats.
3. It should return a highlighted image and a JSON body that concludes the diagnosis made which can later be presented however we want.

2.2.5 Module 5: Android Development and System Integration

1. The android application can communicate with our backend.
2. The web application and android application are updated at the same time.

2.3 Non-Functional Requirements

2.3.1 Reliability

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

2.4.2 Usability

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

2.4.3 Performance

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

2.4.4 Security

8. All sensitive data that is saved in the database will be encrypted.

9. The system will incorporate authorization methods, ensuring users are only granted access to features they are authorized to access.

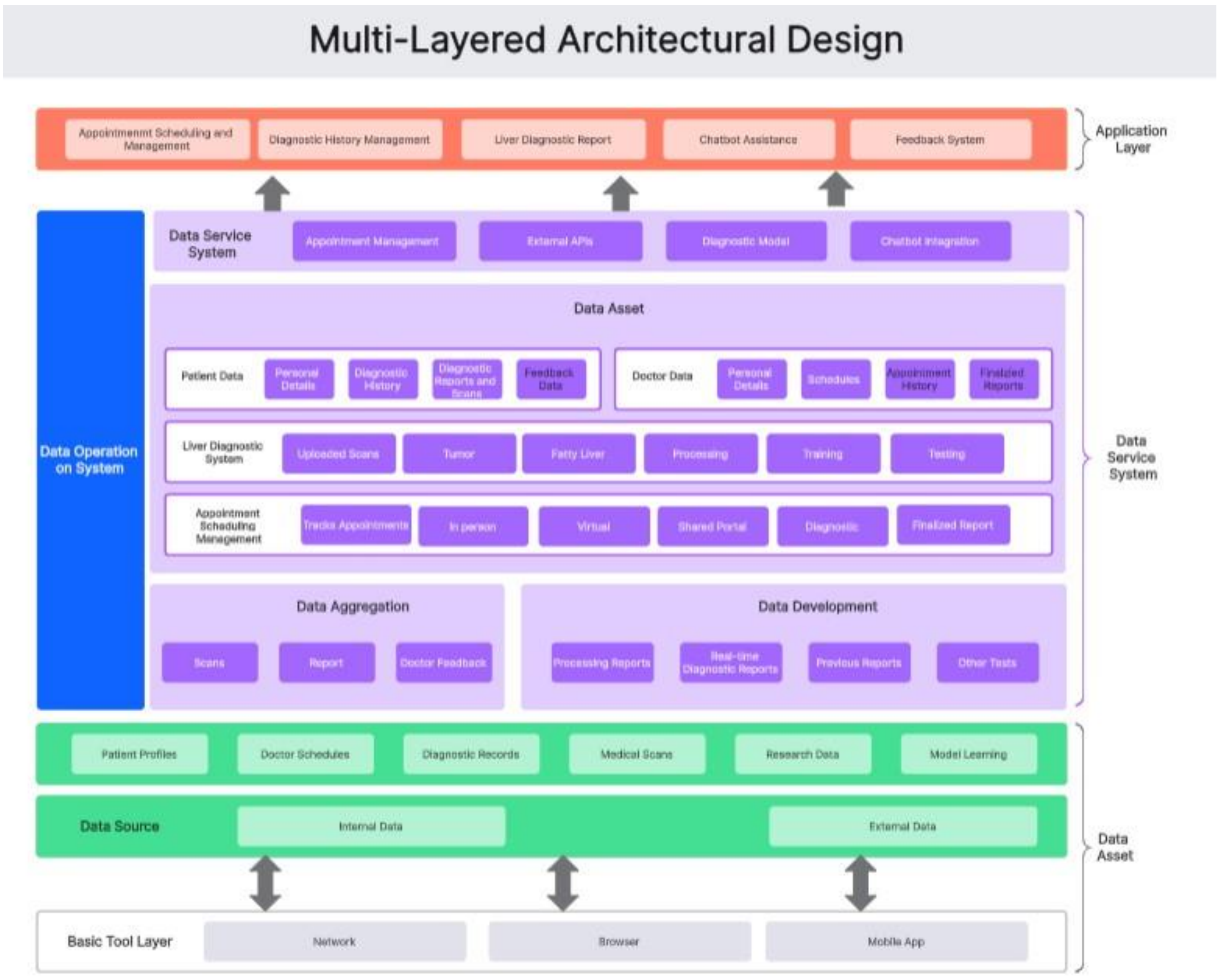
10. The system will use an authentication method, which ensures users that are logged in are not only authorized, but also have valid token sessions through proper authentication.

Chapter 3

System Overview [AS PER FYP1 MID REPORT]

Give a general description of the functionality, context, and design of your project. Provide any background information if necessary.

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 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:

- patient: 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

Health Bridge 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.

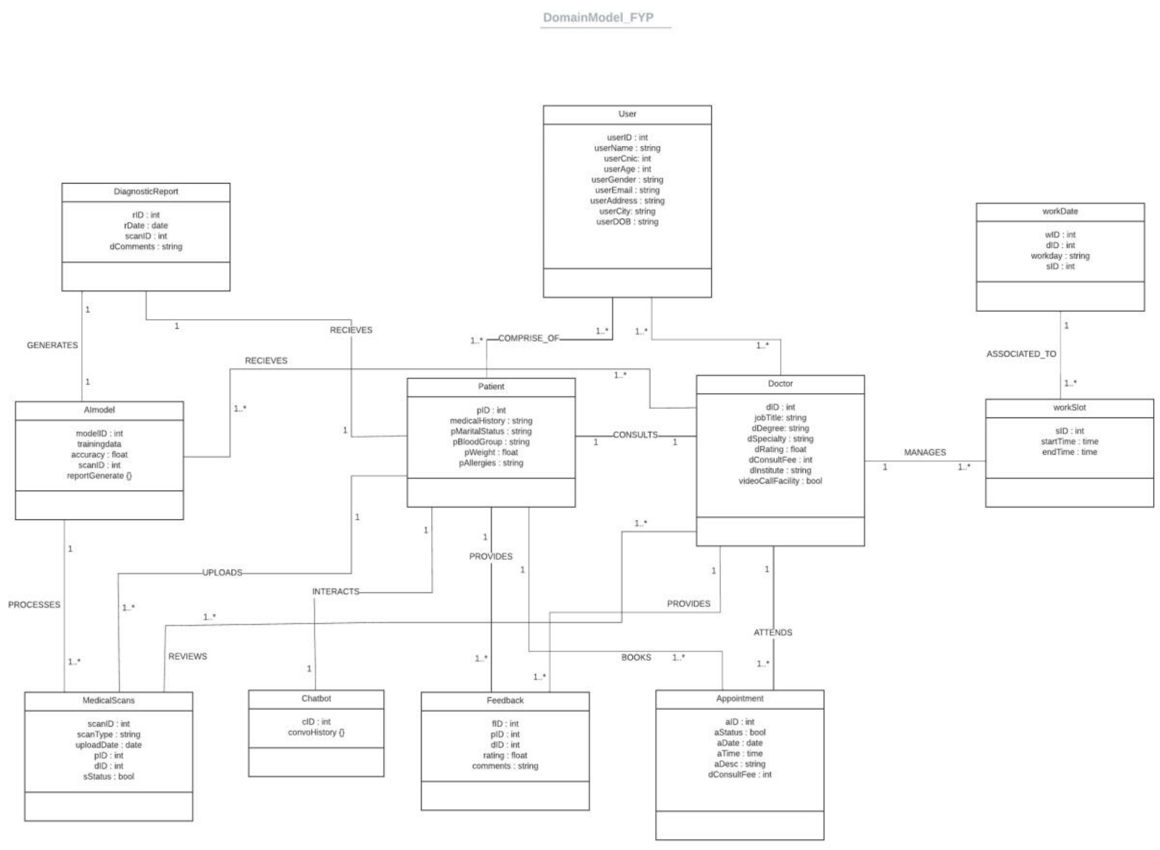
3.2.7 Diagnostic Model

A core feature of Health Bridge 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

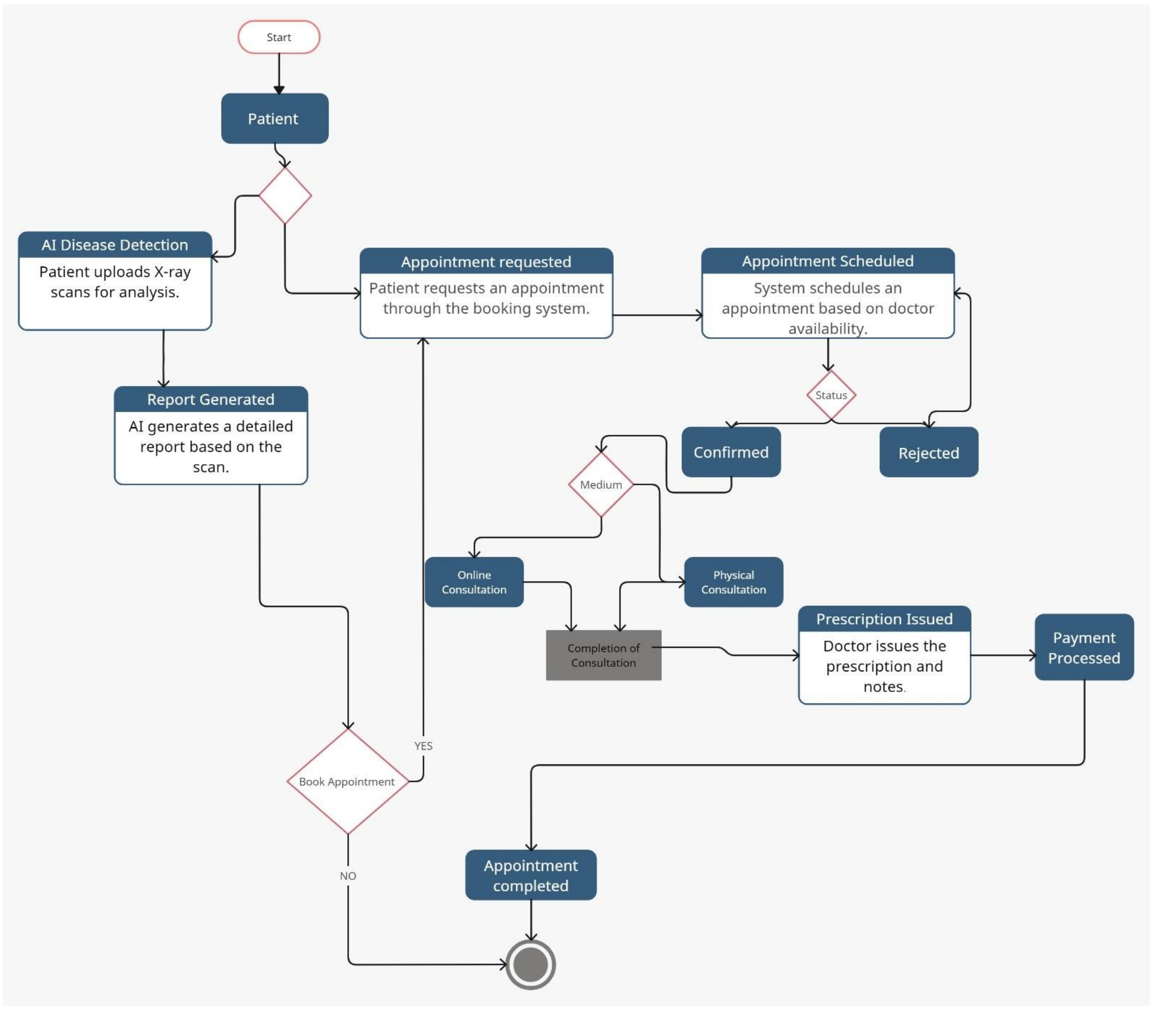
In conclusion, the Health Bridge system leverages MongoDB's document based structure to efficiently manage healthcare data. The data is organized into collections that reflect real-world entities such as appointments, doctors, patients, and payments. These data structures are further enhanced with AI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

3.3 Domain Model



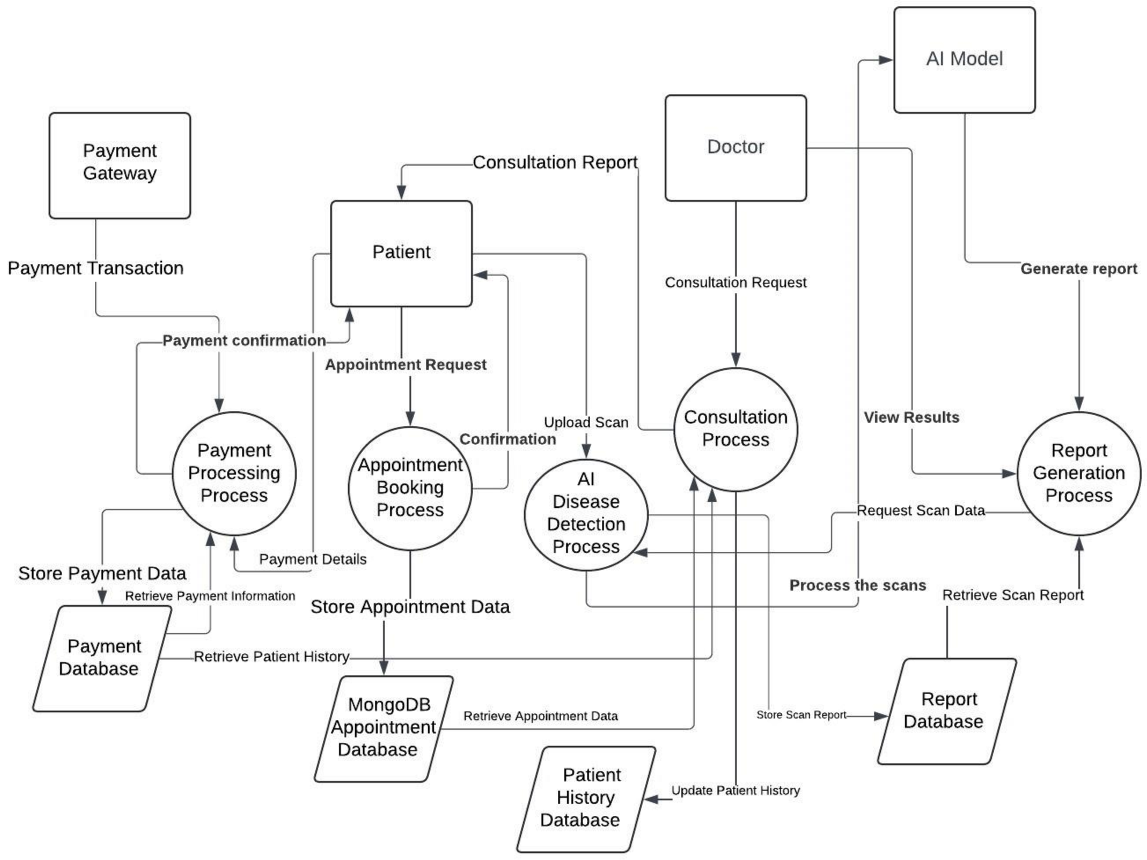
3.4 Design Models [UPTO THE CURRENT ITERATION ONLY]

3.4.1 Activity Diagram



Start
Patient
AI Disease Detection
Patient uploads X-ray scans for analysis.
Appointment requested
Patient requests an appointment through the booking system.
Appointment Scheduled
System schedules an appointment based on doctor availability.
Report Generated
AI generates a detailed report based on the scan.
Status
Confirmed
Rejected
Medium
Online Consultation
Physical Consultation
Completion of Consultation
Prescription Issued
Doctor issues the prescription and notes.
Payment Processed
YES
Book Appointment
NO
Appointment completed

3.4.2 Data Flow Diagram



3.4.3 System-Level Sequence Diagrams

3.2.3.1 Login

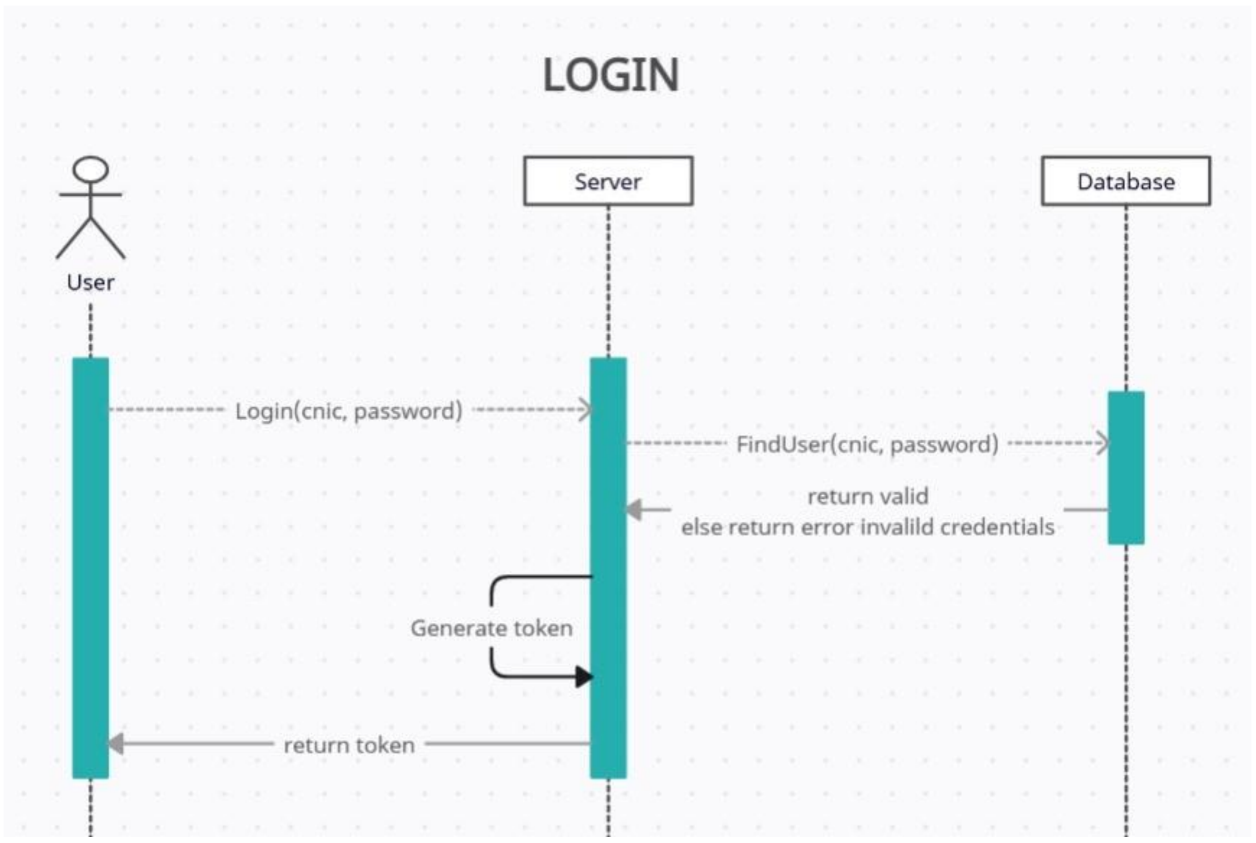


Figure 3.4: Login

3.2.3.2 Registration

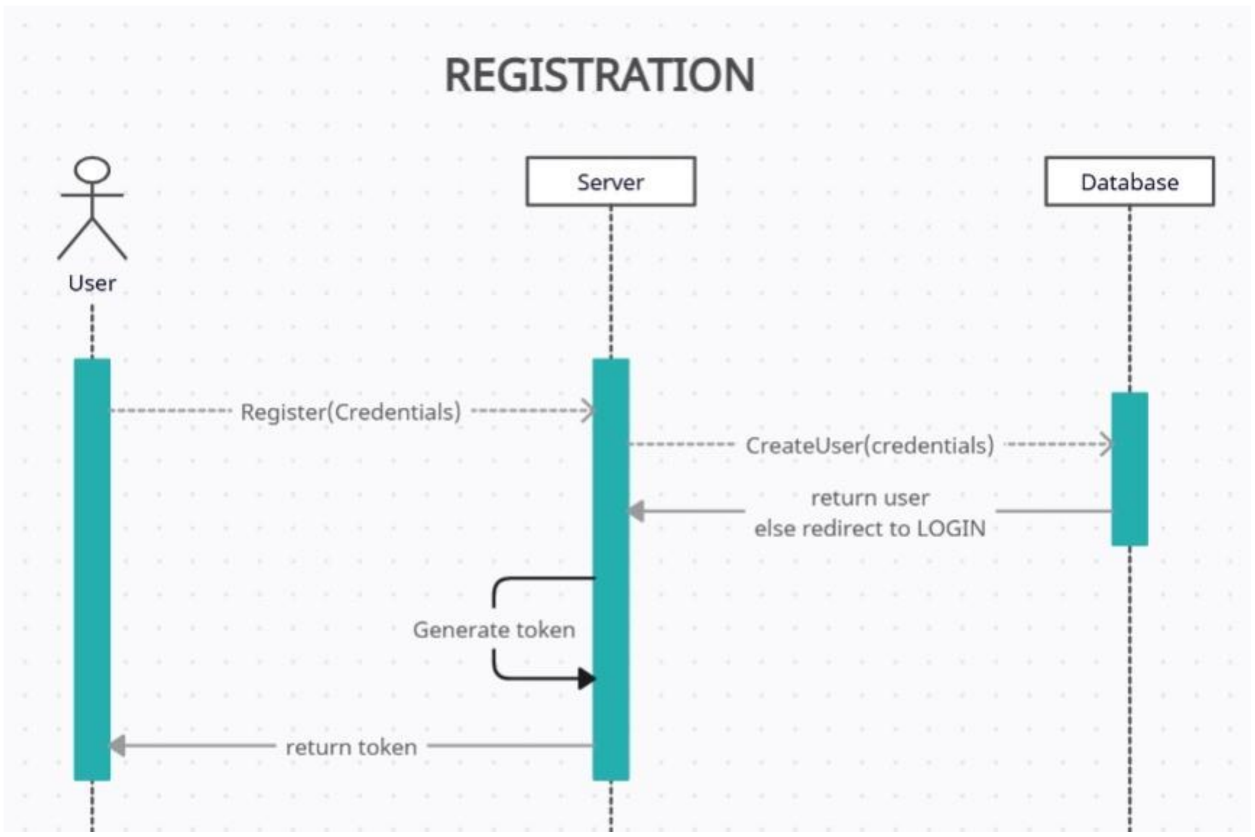


Figure 3.5: Registration

3.2.3.3 Give Feedback

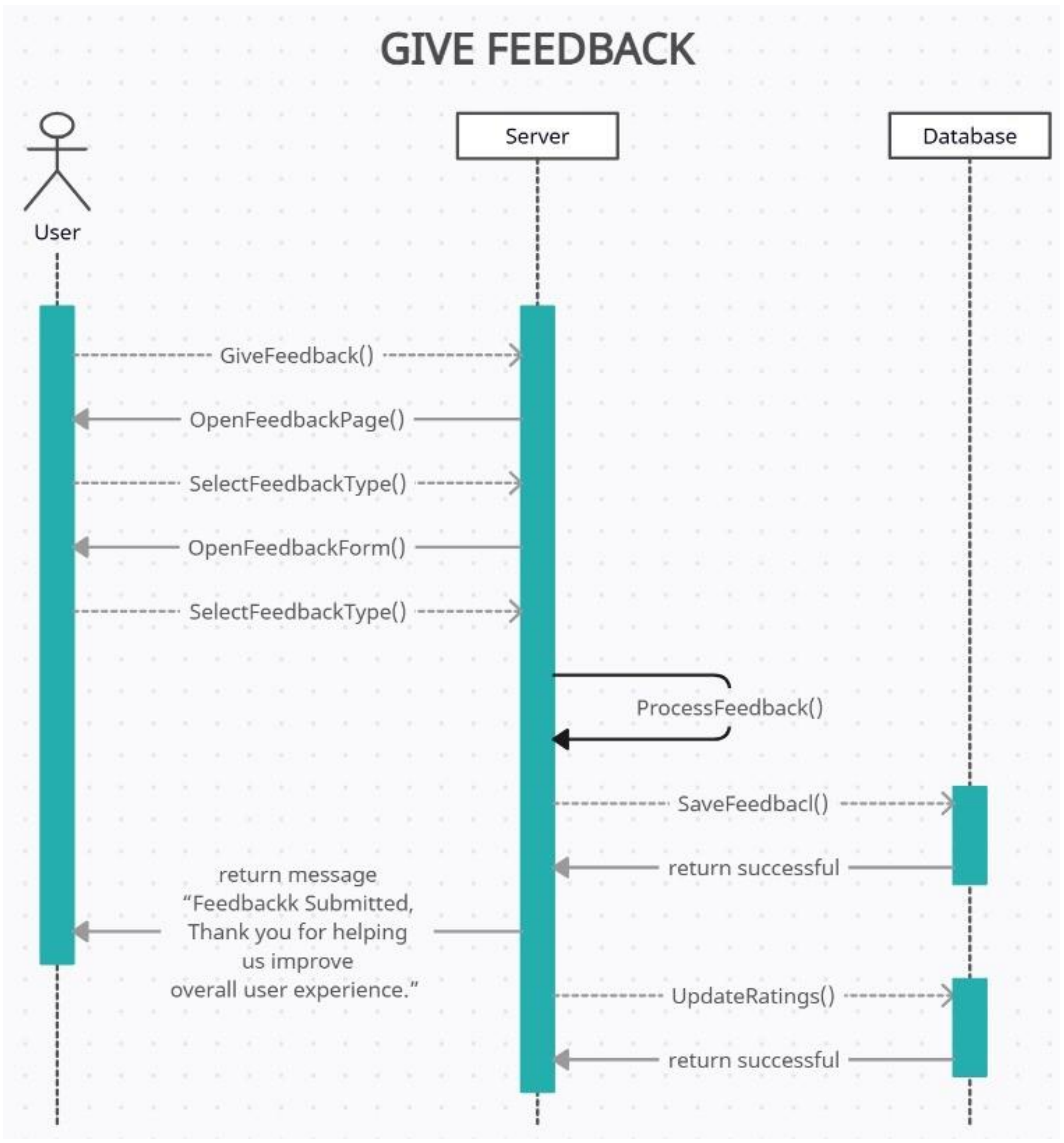
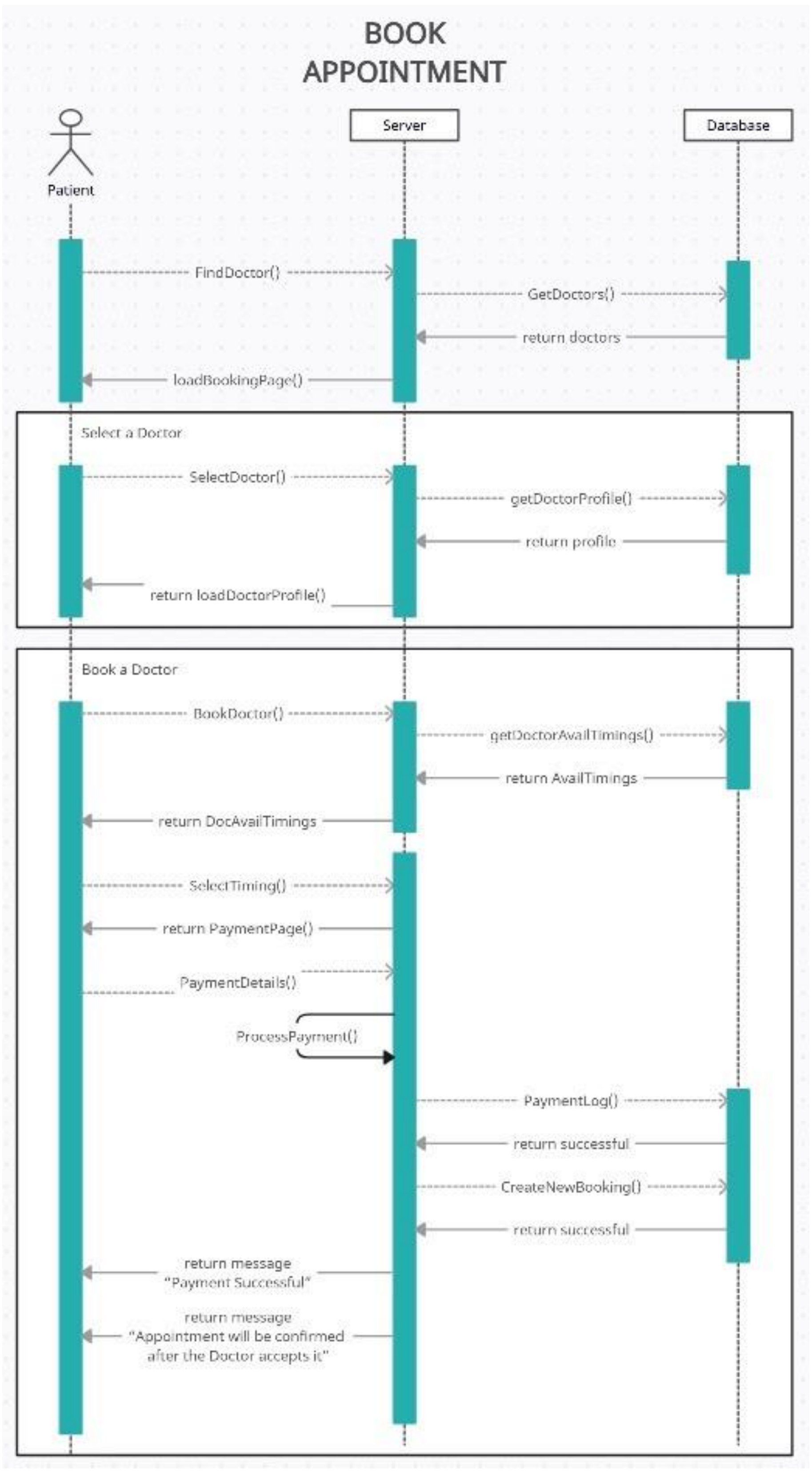


Figure 3.6: Give Feedback

3.2.3.4 Book Appointment



3.2.3.5 Chatbot Interaction

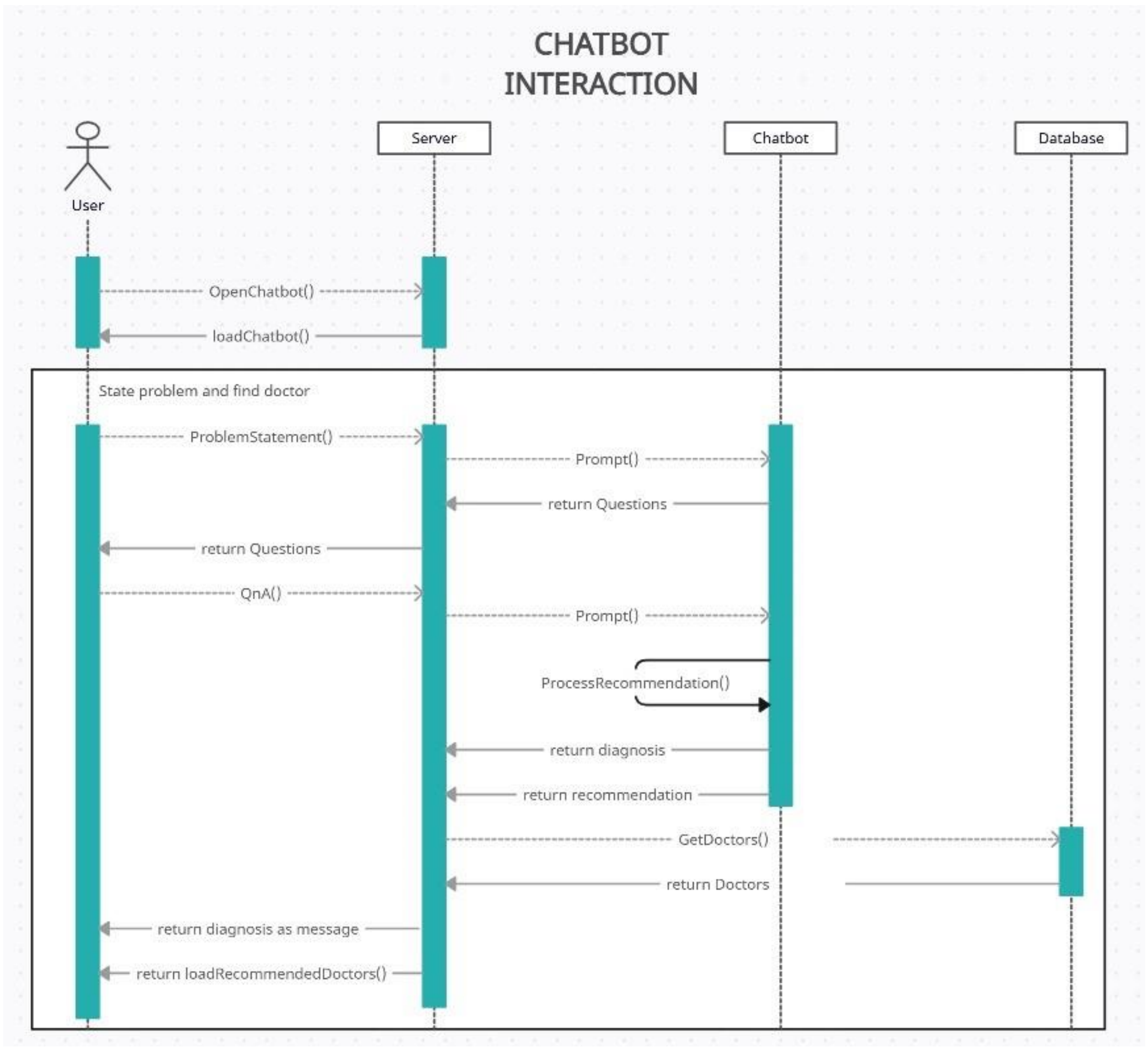


Figure 3.7: Chatbot Interaction

3.2.3.6 Diagnostic History

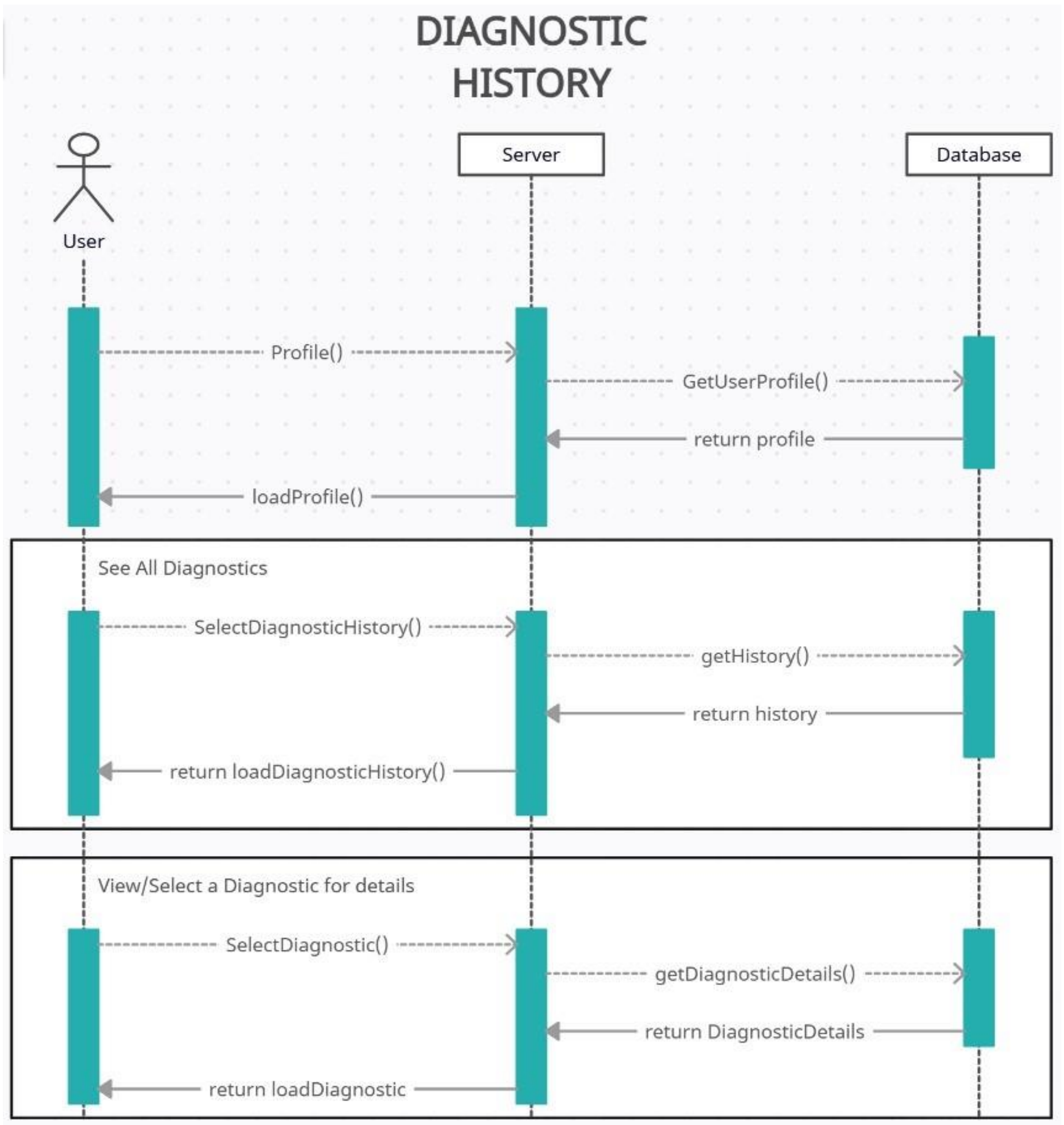


Figure 3.8: Diagnostic History

3.2.3.7 Manage Appointments

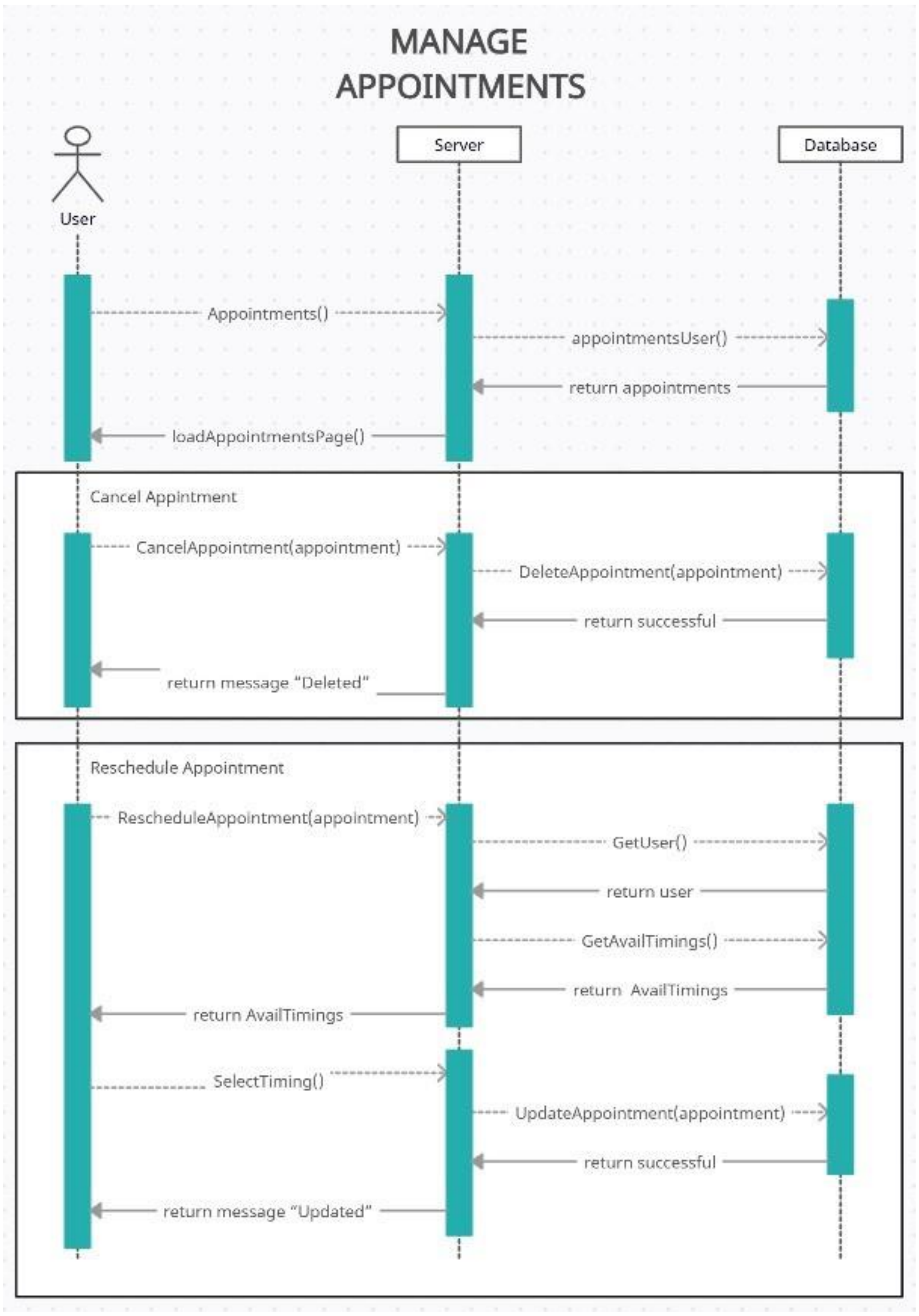


Figure 3.9: Manage Appointments

3.2.3.8 Upload Scans and generate Report

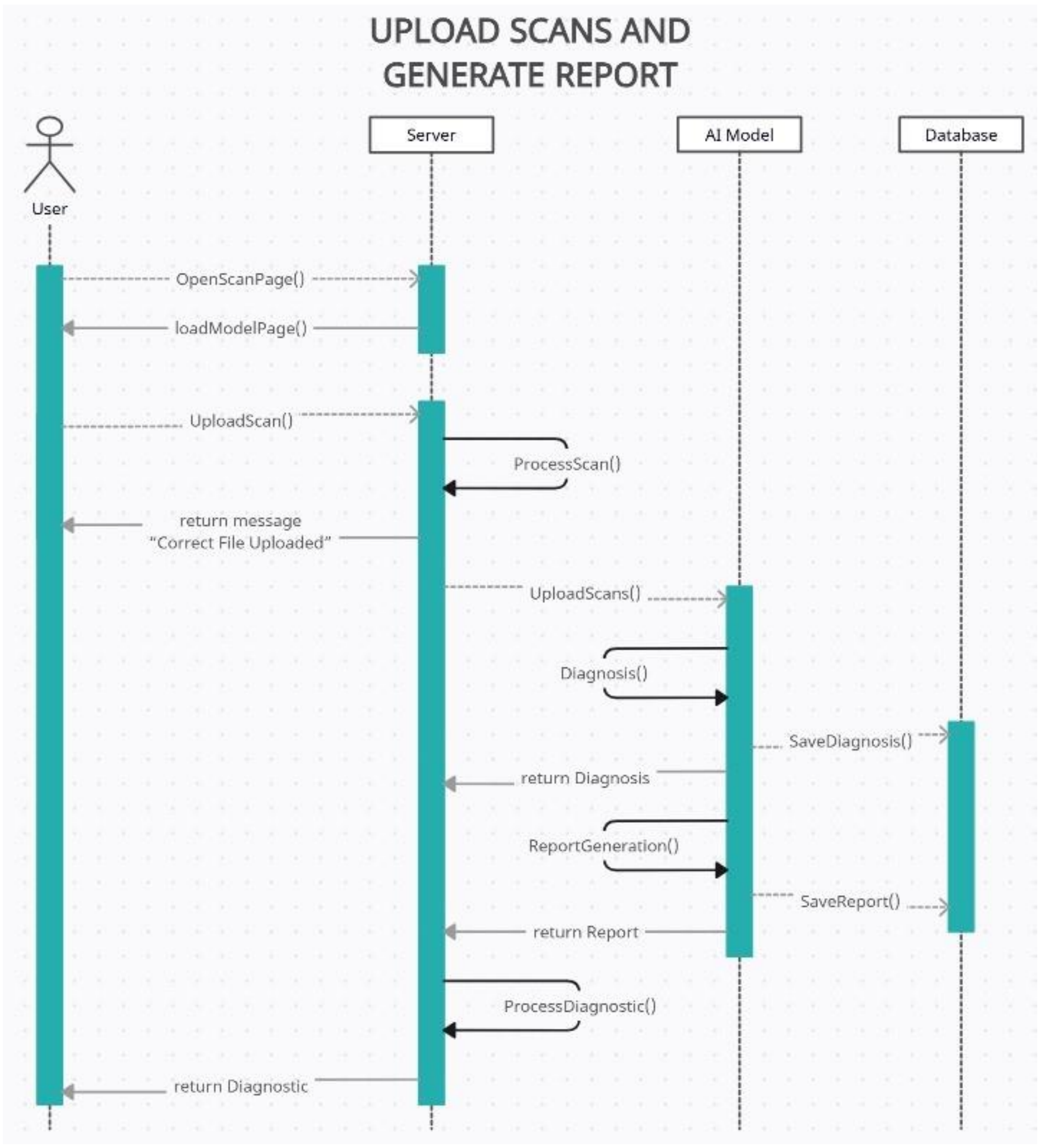
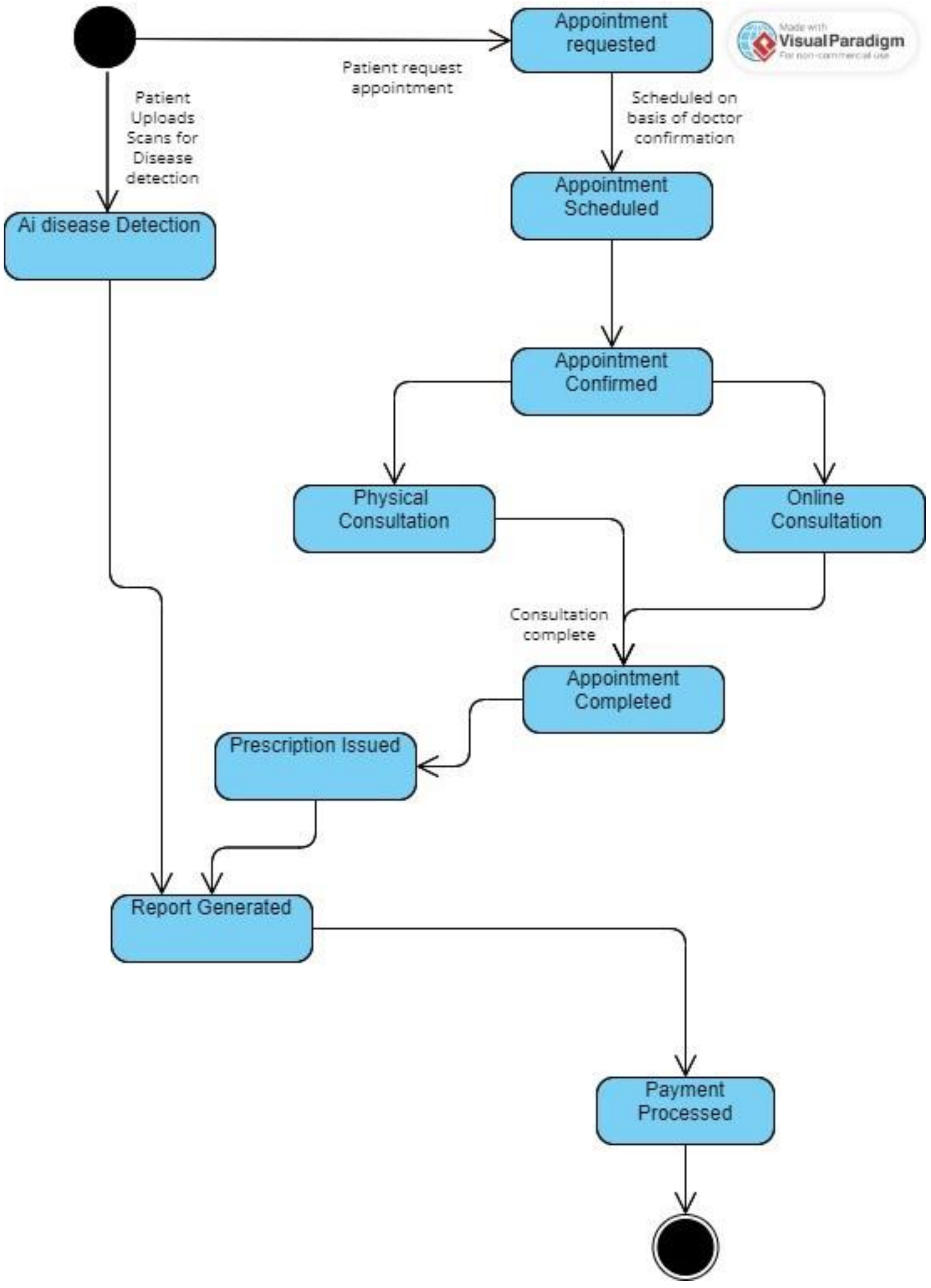


Figure 3.10: Upload Scans and generate Report

3.4.4 State Transition Diagram



Chapter 4

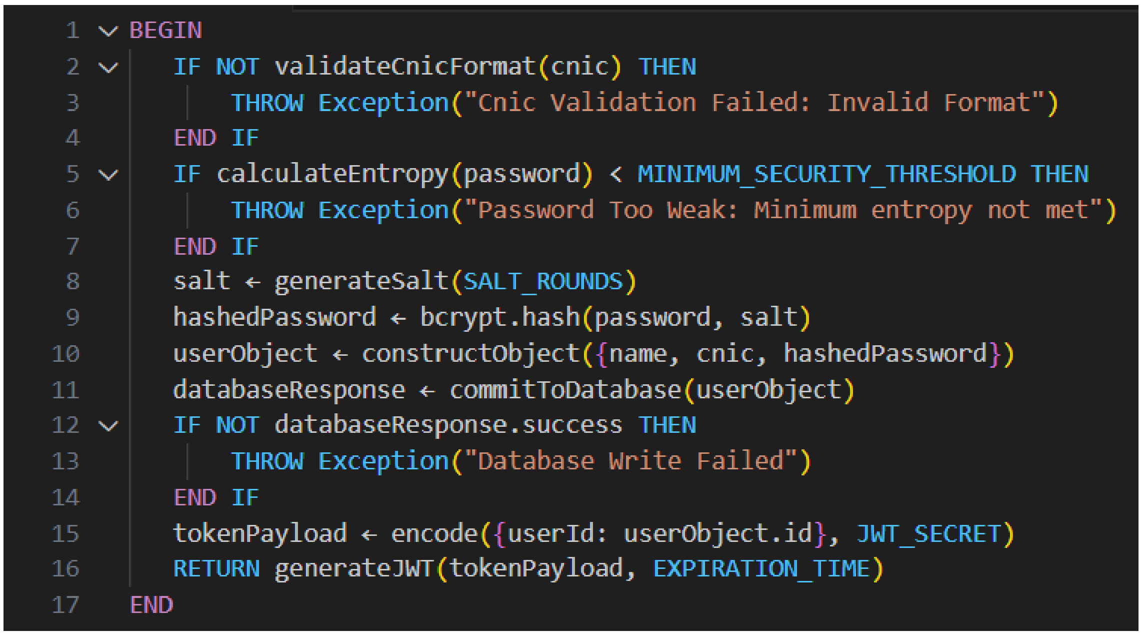
Implementation and Testing

4.1 Algorithm Design

4.1.1 Secure User Registration with Cryptographic Hashing:

Ensures secure storage of user credentials by validating inputs, hashing passwords, and issuing JWTs.

Input: name, cnic, password
Output: Securely hashed user credentials, JWT token

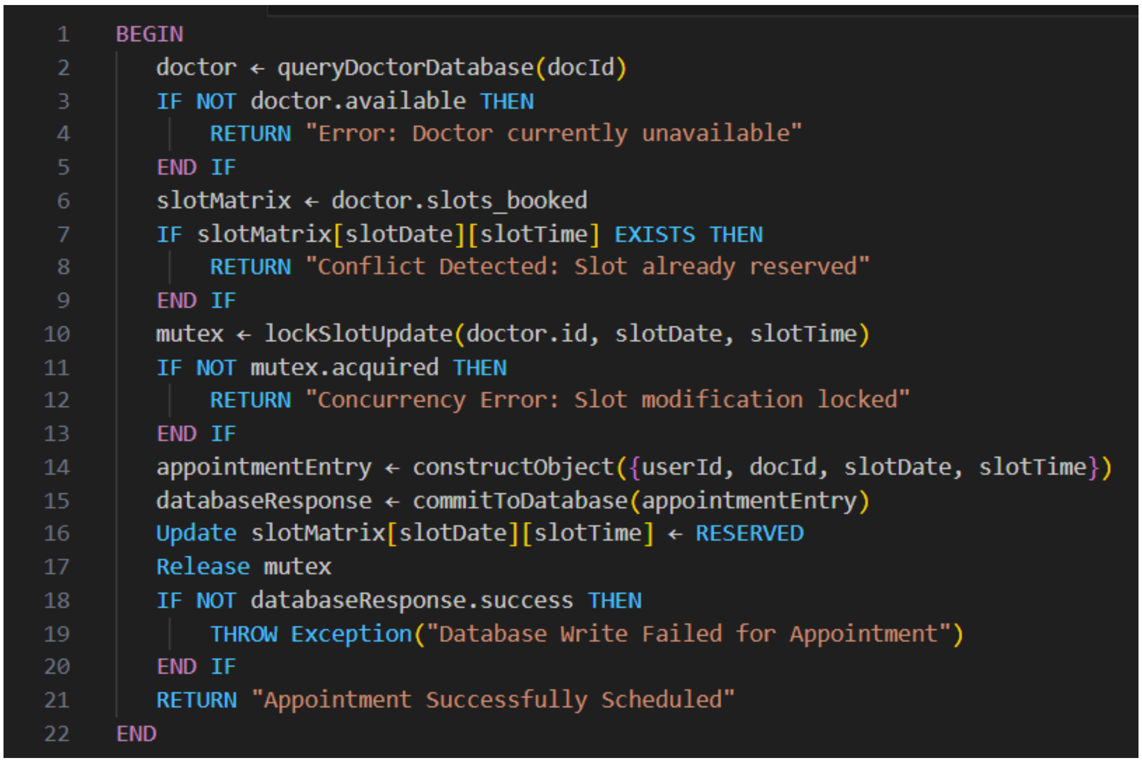


```
BEGIN
    IF NOT validateCnicFormat(cnic) THEN
        THROW Exception("Cnic Validation Failed: Invalid Format")
    END IF
    IF calculateEntropy(password) < MINIMUM_SECURITY_THRESHOLD THEN
        THROW Exception("Password Too Weak: Minimum entropy not met")
    END IF
    salt ← generateSalt(SALT_ROUNDS)
    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
```

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

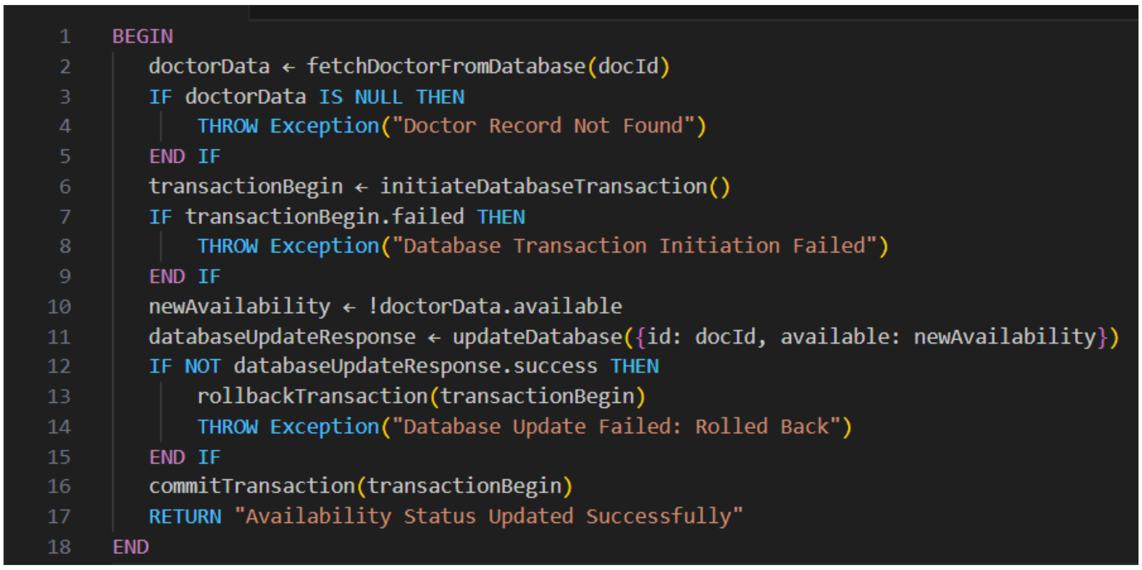


```
BEGIN
  doctor ← queryDoctorDatabase(docId)
  IF NOT doctor.available THEN
    RETURN "Error: Doctor currently unavailable"
  END IF
  slotMatrix ← doctor.slots_booked
  IF slotMatrix[slotDate][slotTime] EXISTS THEN
    RETURN "Conflict Detected: Slot already reserved"
  END IF
  mutex ← lockSlotUpdate(doctor.id, 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
```

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

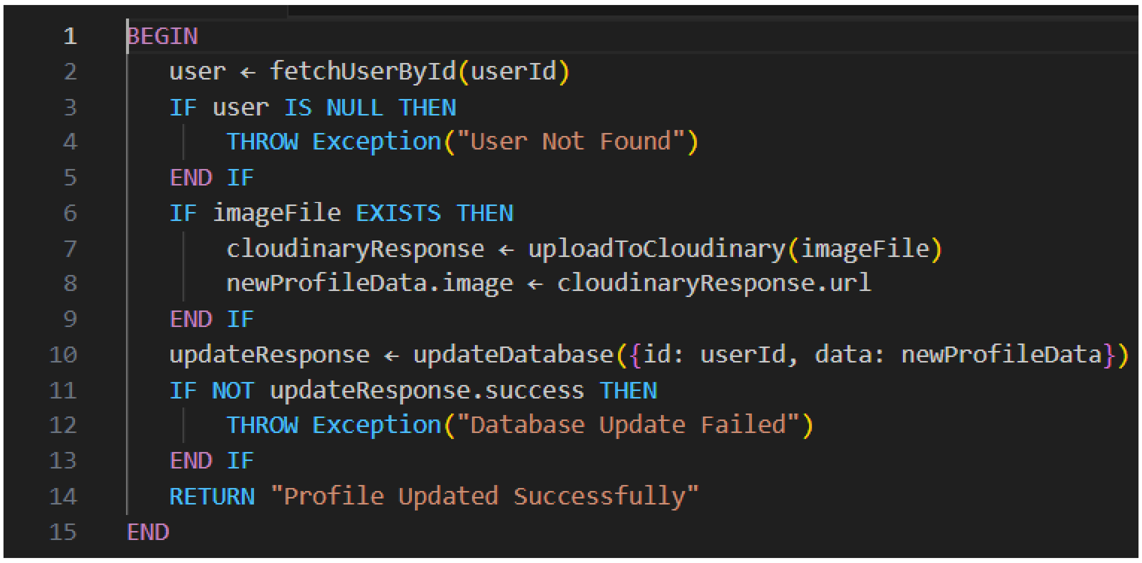


```
BEGIN
  doctorData ← fetchDoctorFromDatabase(docId)
  IF doctorData IS NULL THEN
    THROW Exception("Doctor Record Not Found")
  END IF
  transactionBegin ← initiateDatabaseTransaction()
  IF transactionBegin.failed THEN
    THROW Exception("Database Transaction Initiation Failed")
  END IF
  newAvailability ← !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
```

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

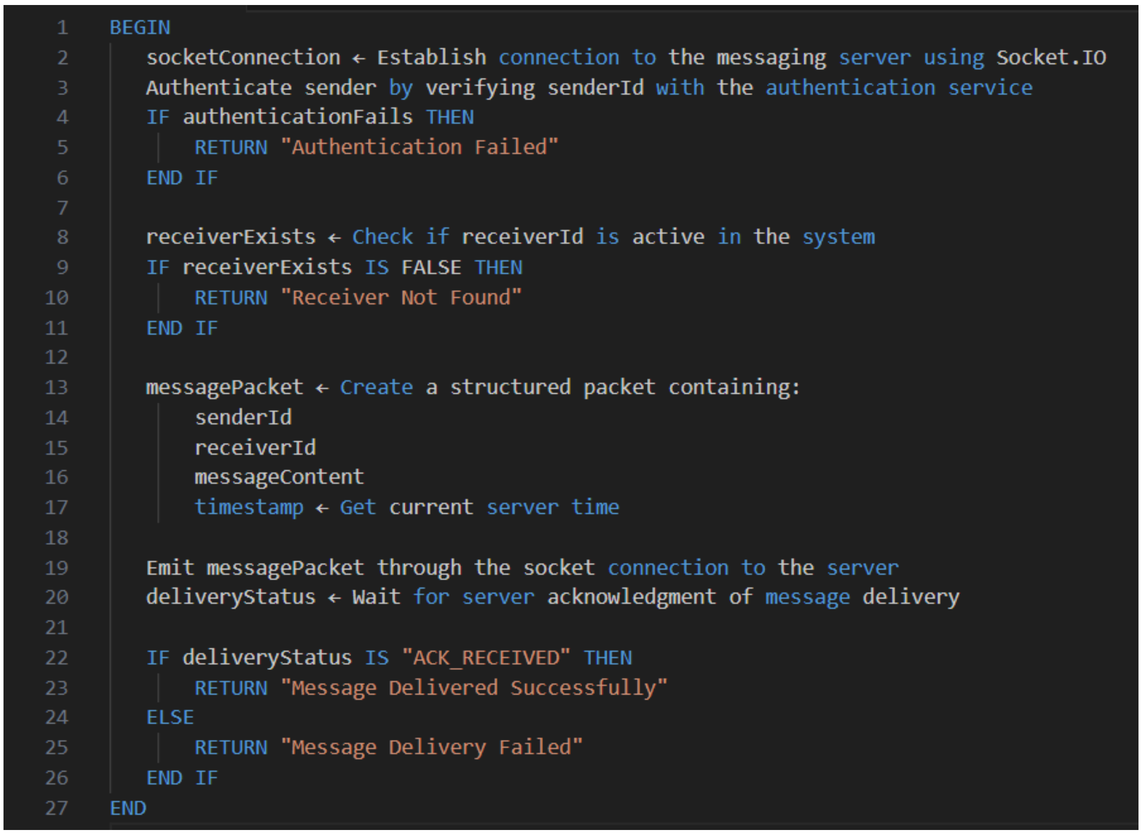


```
BEGIN
    user ← fetchUserById(userId)
    IF user IS NULL THEN
        THROW Exception("User Not Found")
    END IF
    IF imageFile EXISTS THEN
        cloudinaryResponse ← uploadToCloudinary(imageFile)
        newProfileData.image ← cloudinaryResponse.url
    END IF
    updateResponse ← updateDatabase({id: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

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)



```
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
```

4.2 External APIs/SDKs

API/SDK	Description	Purpose of Usage	Endpoint/Functionality
Socket.IO	Real-time communication library	Enables real-time messaging between users, including text, audio, and video	`socket.on`, `socket.emit`
Cloudinary	Cloud-based media management	Used for uploading, managing, and retrieving profile images and medical report files	`cloudinary.uploader.upload`
MongoDB (Mongoose)	NoSQL database for data storage	Provides a database to store user, doctor,	`mongoose.connect`

		appointment, and messaging data	
JWT (JsonWebToken)	Authentication library	Issues and verifies tokens for secure user, doctor, and admin authentication	`jwt.sign`, `jwt.verify`
Twilio	Communication API	Facilitates SMS notifications and reminders for appointments and user communication	`client.messages.create`

4.3 Testing Details

4.3.1 Unit Testing

Test Case ID	**Test Objective**	**Precondition**	**Steps**	**Test Data**	**Expected Result**	**Post-condition**	**Actual Result**	**Pass/ Fail**
TC 001	Verify user registration	User provides valid details	1. Enter `name`, `email`, and `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
TC 002	Verify user login	User is registered	1. Enter valid `email` and `password`. 2. Click	Email: john@example.com Password: Pass@123	Login successful, JWT token issued	User session starts	As expected	Pass

			'Login'.					
TC 003	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
TC 004	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] A Kolyshkin and S Nazarovs. Stability of slowly diverging flows in shallow water. *Mathematical Modeling and Analysis*, 2007.

Appendix A

Appendices

A.1 Appendix A

A.1.1 Use Case Diagram example (Online Shopping System)

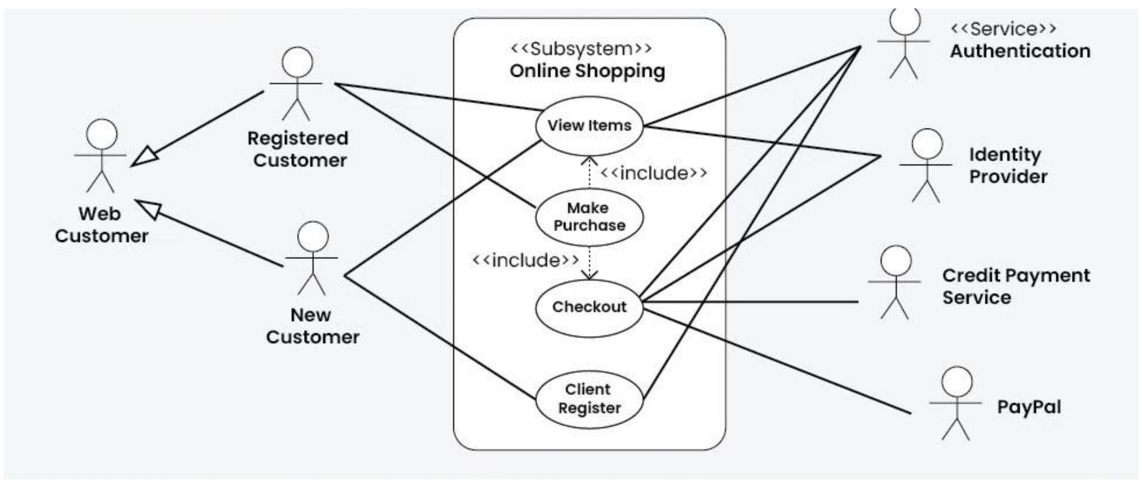
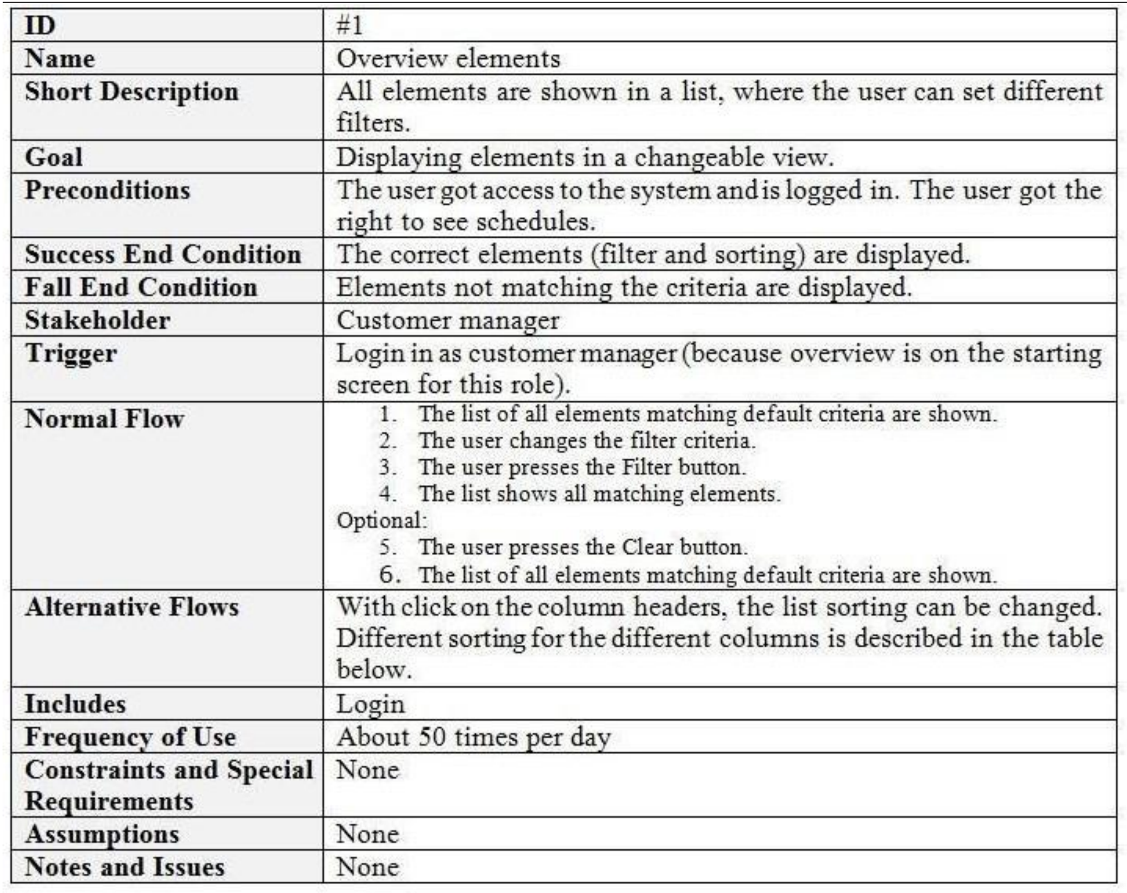


Figure A.1: Use Case Diagram for the Online Shopping System

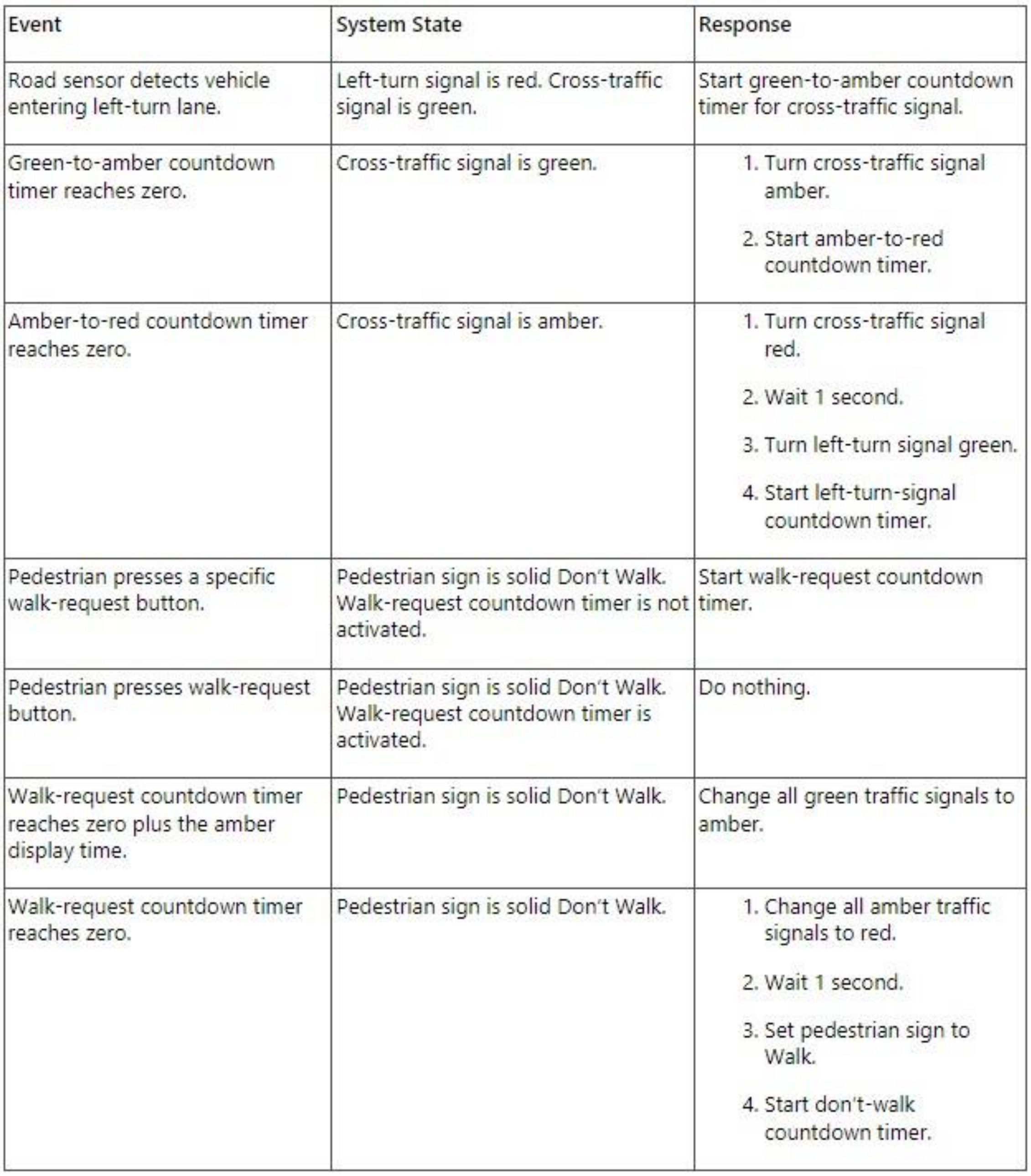
A.1.2 Detail Use Case Example



ID	#1
Name	Overview elements
Short Description	All elements are shown in a list, where the user can set different filters.
Goal	Displaying elements in a changeable view.
Preconditions	The user got access to the system and is logged in. The user got the right to see schedules.
Success End Condition	The correct elements (filter and sorting) are displayed.
Fall End Condition	Elements not matching the criteria are displayed.
Stakeholder	Customer manager
Trigger	Login in as customer manager (because overview is on the starting screen for this role).
Normal Flow	1. The list of all elements matching default criteria are shown. 2. The user changes the filter criteria. 3. The user presses the Filter button. 4. The list shows all matching elements. Optional: 5. The user presses the Clear button. 6. The list of all elements matching default criteria are shown.
Alternative Flows	With click on the column headers, the list sorting can be changed. Different sorting for the different columns is described in the table below.
Includes	Login
Frequency of Use	About 50 times per day
Constraints and Special Requirements	None
Assumptions	None
Notes and Issues	None

Figure A.2: Detail Use Case Example

A.1.3 Event-Response Table for a Highway Intersection



Event	System State	Response
Road sensor detects vehicle entering left-turn lane.	Left-turn signal is red. Cross-traffic signal is green.	Start green-to-amber countdown timer for cross-traffic signal.
Green-to-amber countdown timer reaches zero.	Cross-traffic signal is green.	1. Turn cross-traffic signal amber. 2. Start amber-to-red countdown timer.
Amber-to-red countdown timer reaches zero.	Cross-traffic signal is amber.	1. Turn cross-traffic signal red. 2. Wait 1 second. 3. Turn left-turn signal green. 4. Start left-turn-signal countdown timer.
Pedestrian presses a specific walk-request button.	Pedestrian sign is solid Don't Walk. Walk-request countdown timer is not activated.	Start walk-request countdown timer.
Pedestrian presses walk-request button.	Pedestrian sign is solid Don't Walk. Walk-request countdown timer is activated.	Do nothing.
Walk-request countdown timer reaches zero plus the amber display time.	Pedestrian sign is solid Don't Walk.	Change all green traffic signals to amber.
Walk-request countdown timer reaches zero.	Pedestrian sign is solid Don't Walk.	1. Change all amber traffic signals to red. 2. Wait 1 second. 3. Set pedestrian sign to Walk. 4. Start don't-walk countdown timer.

Figure A.3: Example of Event Response Table

A.1.4 Story Board Example For Android App

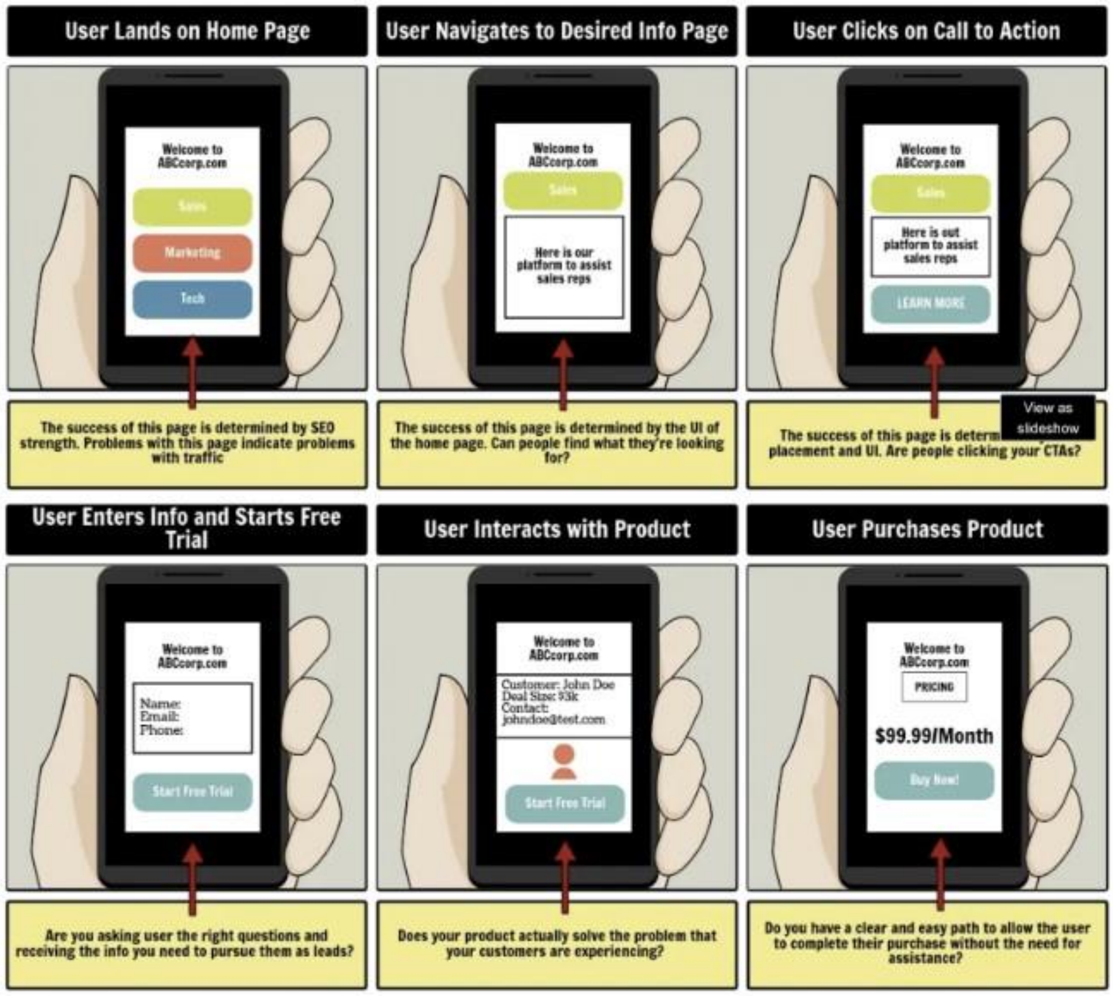

User Lands on Home Page
Welcome to ABCcorp.com
Sales
Marketing
Tech
The success of this page is determined by SEO strength. Problems with this page indicate problems with traffic
User Navigates to Desired Info Page
Welcome to ABCcorp.com
Sales
Here is our platform to assist sales reps
The success of this page is determined by the UI of the home page. Can people find what they're looking for?
User Clicks on Call to Action
Welcome to ABCcorp.com
Sales
Here is out platform to assist sales reps
LEARN MORE
View as slideshow
The success of this page is determ placement and UI. Are people clicking your CTAs?
User Enters Info and Starts Free Trial
Welcome to ABCcorp.com
Name:
Email:
Phone:
Start Free Trial
Are you asking user the right questions and receiving the info you need to pursue them as leads?
User Interacts with Product
Welcome to ABCcorp.com
Customer: John Doe
Deal Size: $3k
Contact: johndoe@test.com
Start Free Trial
Does your product actually solve the problem that your customers are experiencing?
User Purchases Product
Welcome to ABCcorp.com
PRICING
$99.99/Month
Buy Now!
Do you have a clear and easy path to allow the user to complete their purchase without the need for assistance?

Figure A.4: Example of Story Board

A.2 Appendix B

A.2.1 Domain Model Example For Online Shopping

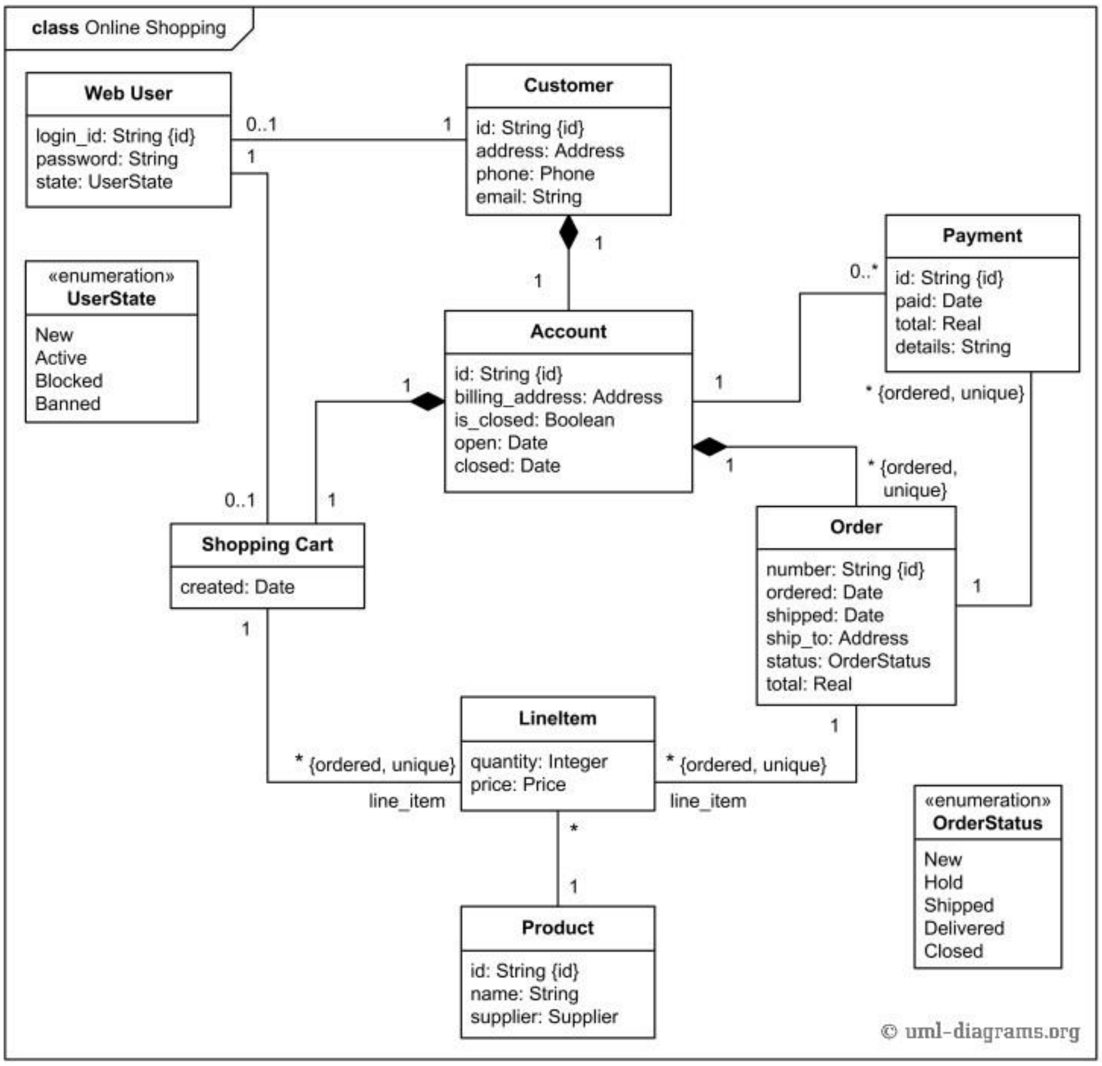


Figure A.5: Domain Model Example For Online Shopping Application

A.3 Appendix C

A.3.1 Box And Line Example For Online Shopping

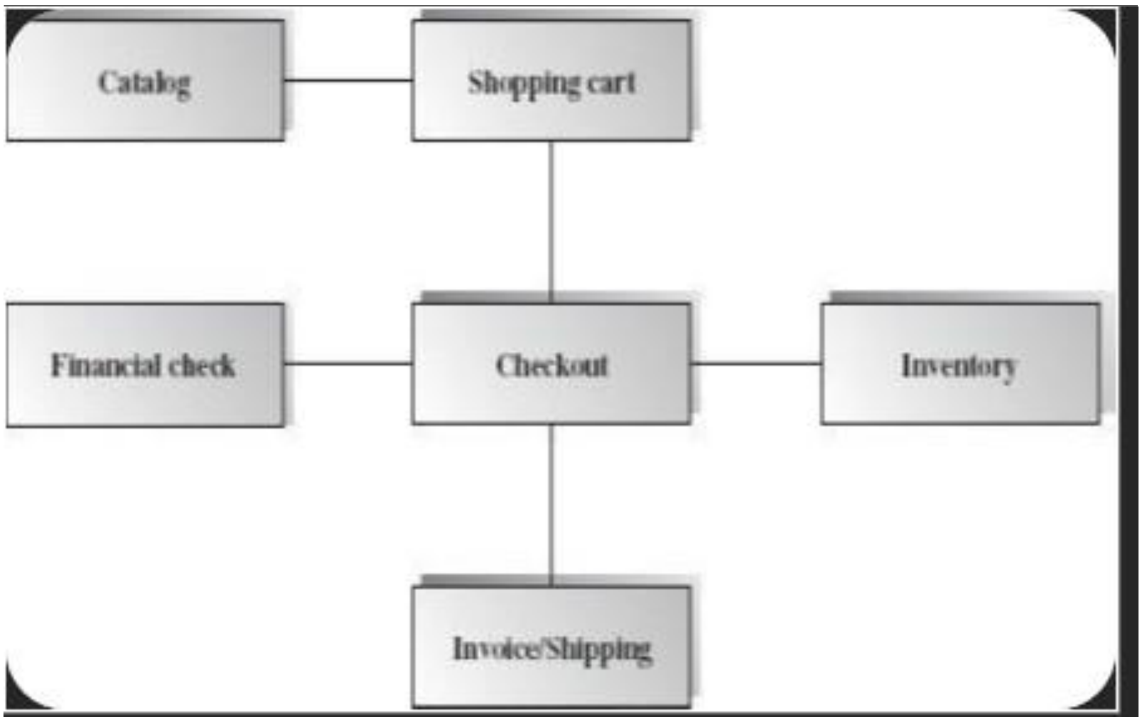


Figure A.6: Box and Line Diagram For Online Shopping Application

A.3.2 Architecture Pattern Example For Online Shopping

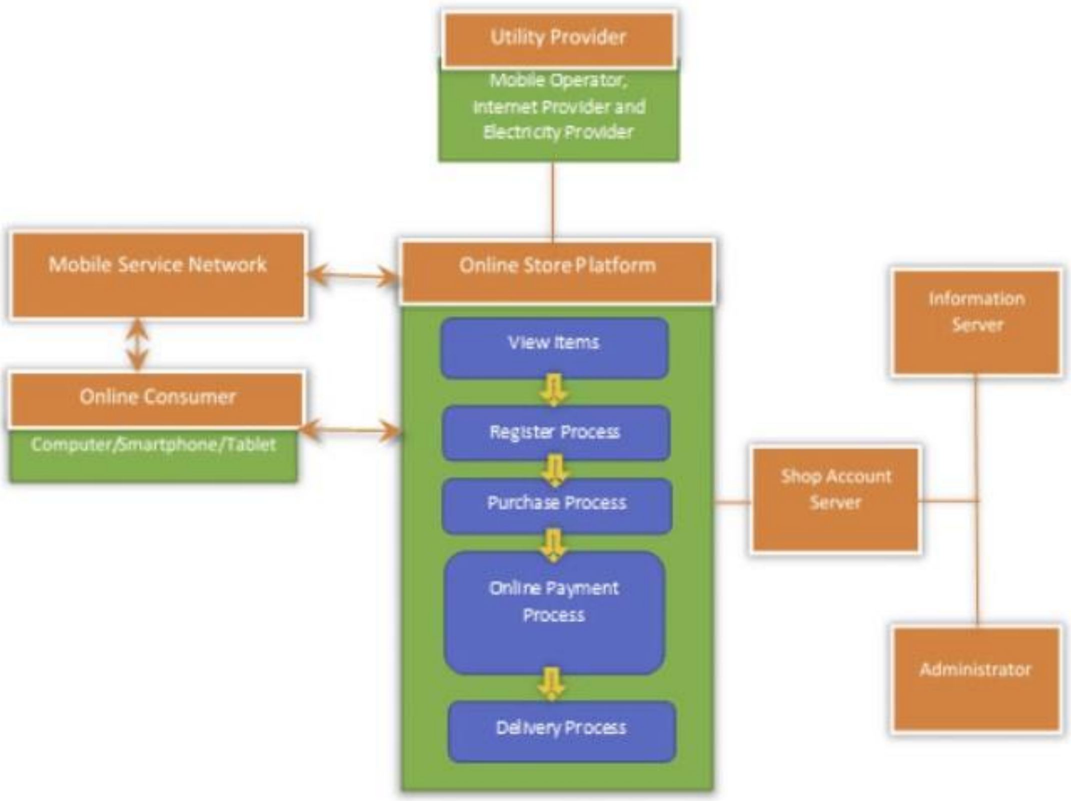


Figure A.7: Architecture Pattern For Online Shopping Application

A.4 Appendix D

A.4.1 Activity Diagram

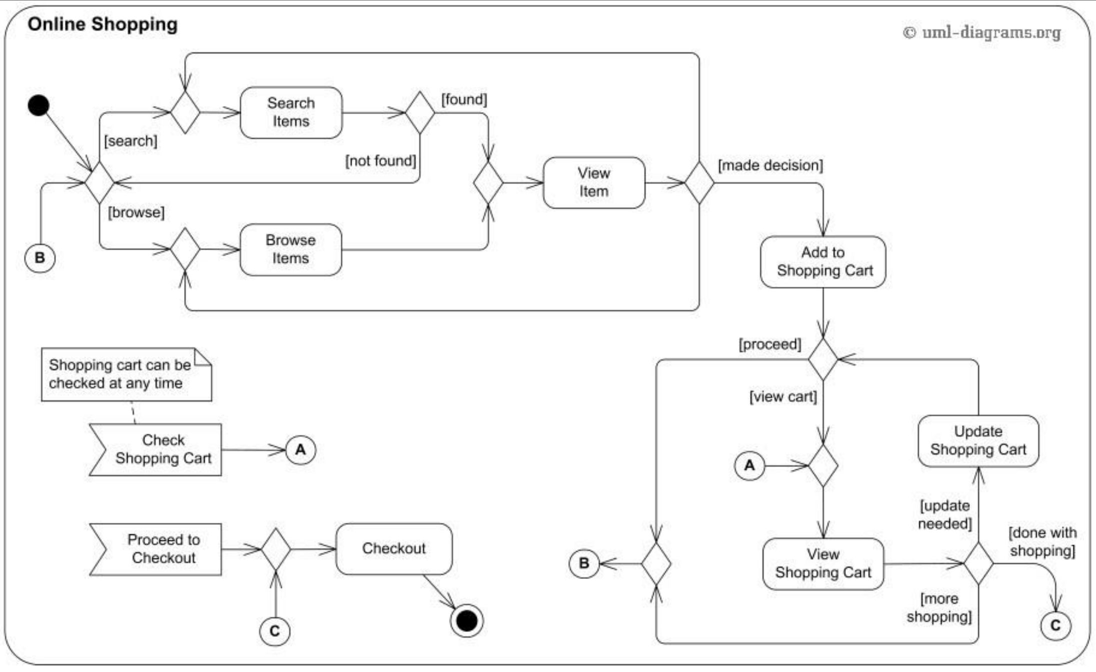


Figure A.8: Activity Diagram For Online Shopping Application

A.4.2 Class Diagram

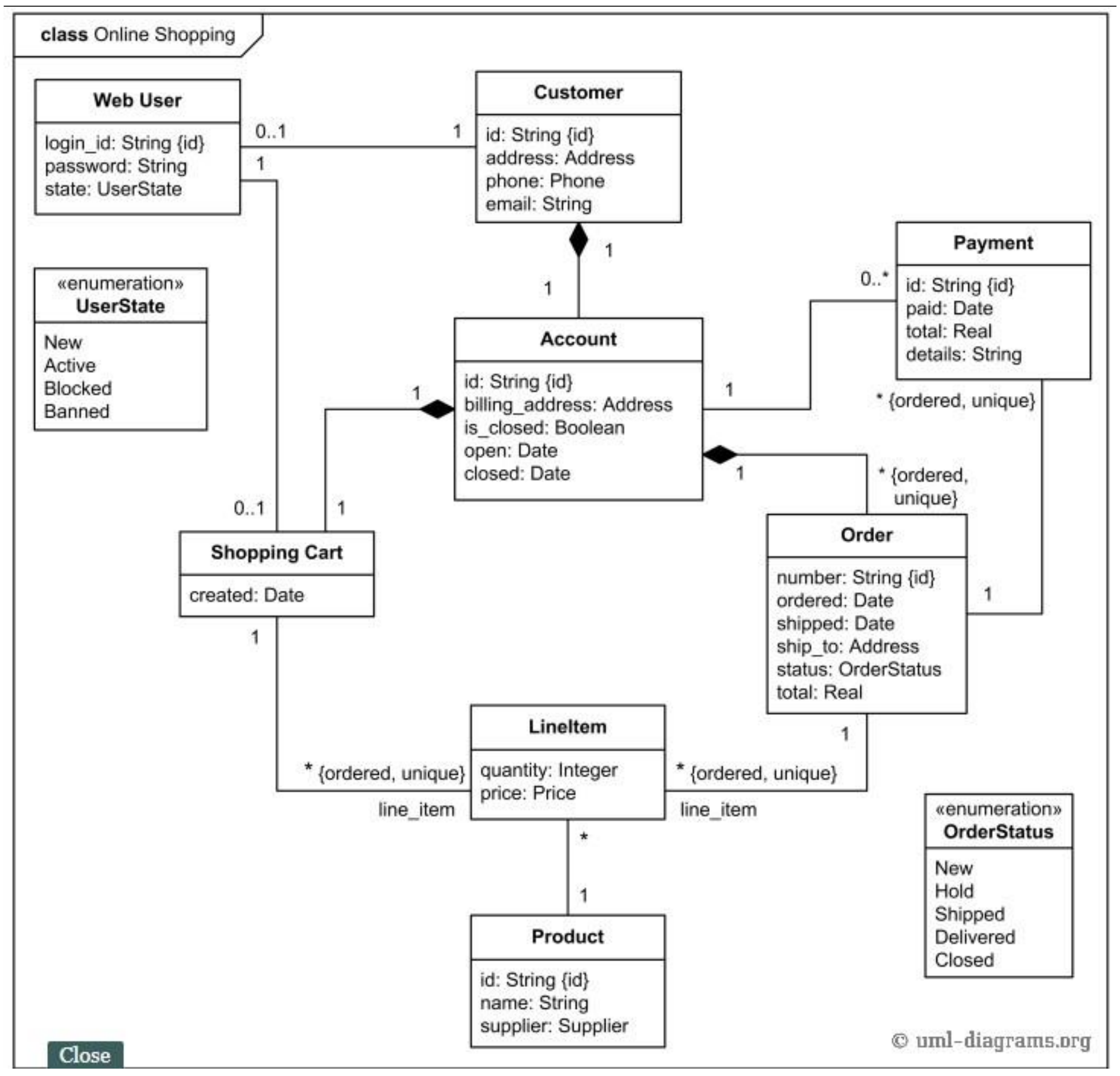


Figure A.9: Class Diagram For Online Shopping Application

A.4.3 Sequence Diagram

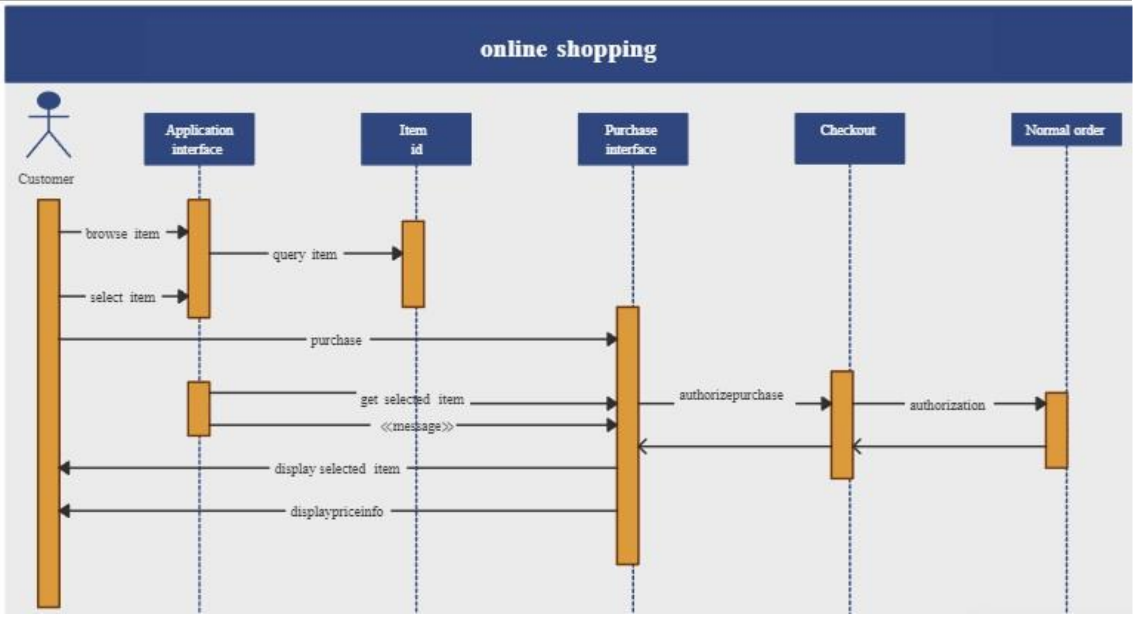


Figure A.10: Sequence Diagram For Online Shopping Application

A.4.4 State Transition Diagram

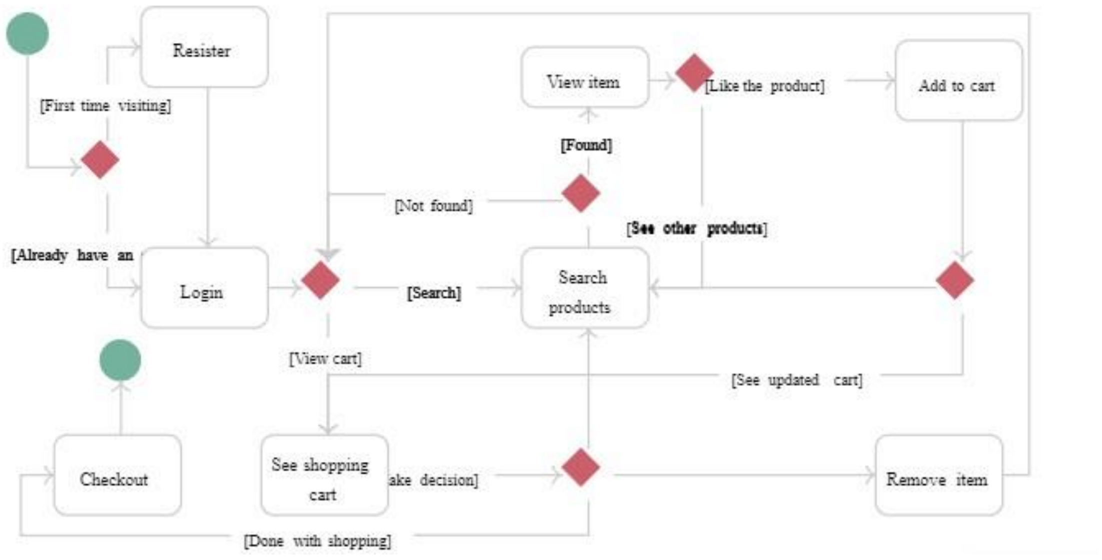


Figure A.11: State Transition Diagram For Online Shopping Application

A.4 Appendix D

A.4.5 Data Flow Diagram

Data Flow Diagram

Data flow diagram symbols, symbol names, and examples

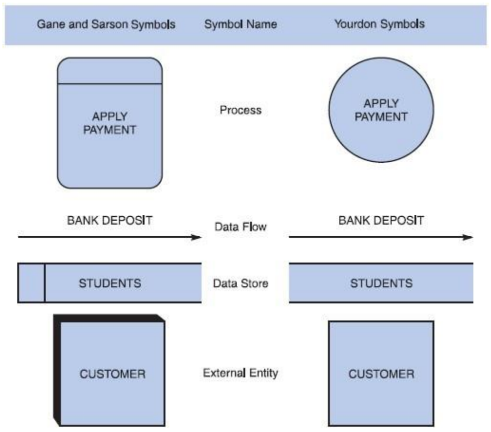


Figure B-5 Data flow diagram symbols, symbol names, and examples of the Gane and Sarson and Yourdon symbol sets.

Guidelines for Drawing DFDs

Step 1: Draw a Context Diagram: The first step in constructing a set of DFDs is to draw a context diagram. A context diagram is a top-level view of an information system that shows the system's boundaries and scope. Data stores are not shown in the context diagram because they are contained within the system and remain hidden until more detailed diagrams are created.

Example

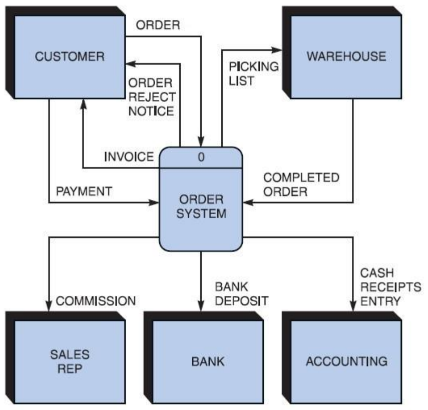


Figure B-6 Context diagram DFD for an order system.

Step 2: Draw a Diagram 0 DFD: To show the detail inside the black box, you create DFD diagram 0. Diagram 0 zooms in on the system and shows major internal processes, data flows, and data stores. Diagram 0 also repeats the entities and data flows that appear in the context diagram. When you expand the context diagram into DFD diagram 0, you must retain all the connections that flow into and out of process 0. Example

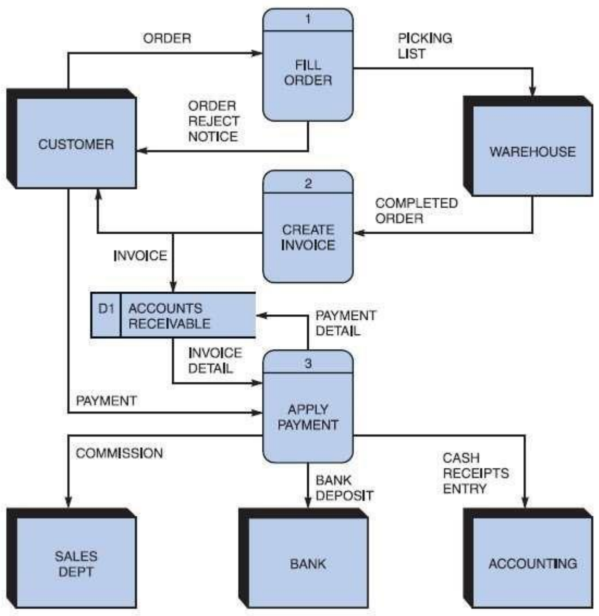


Figure B-7 Diagram 0 DFD for the order system.

Step 3: Draw the Lower-Level Diagrams:

To create lower-level diagrams, you must use leveling and balancing techniques. Leveling is the process of drawing a series of increasingly detailed diagrams, until all functional primitives are identified.

Leveling Example

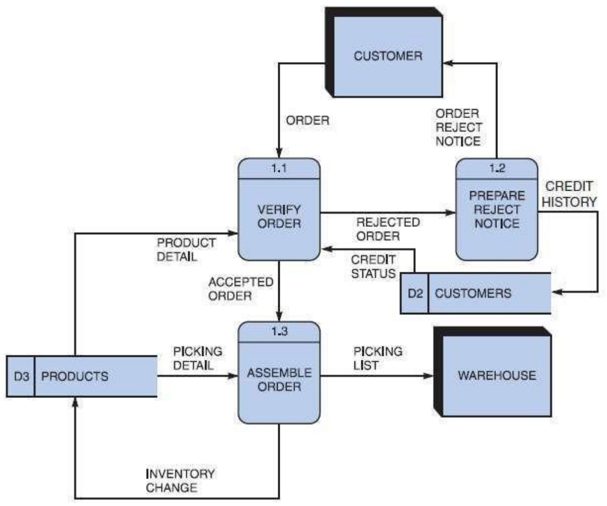


Figure B-8 Diagram 1 DFD shows details of the FILL ORDER process in the order system.