Health Companion
Health Care System

Project Report

Submitted by

Jince Abraham

Reg. No.: AJC23MCA-2036

In Partial fulfillment for the Award of the Degree of

MASTER OF COMPUTER APPLICATIONS

(MCA TWO YEAR)

(Accredited by NBA)

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

AMAL JYOTHI COLLEGE OF ENGINEERING AUTONOMOUS

KANJIRAPPALLY

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

2023-2025

DEPARTMENT OF COMPUTER APPLICATIONS

AMAL JYOTHI COLLEGE OF ENGINEERING AUTONOMOUS

KANJIRAPPALLY

CERTIFICATE

This is to certify that the Project report titled "**HEALTH COMPANION**" is the bona fide work of **JINCE ABRAHAM (Regno: AJC23MCA-2036)** carried out in partial fulfillment of the requirements for the award of the **Degree of Master of Computer Applications** at **Amal Jyothi College of Engineering Autonomous, Kanjirappally,** Affiliated to **APJ Abdul Kalam Technological University.** The project was undertaken during the period from **December 10, 2024,** to **March 27, 2025**.

Mr. Ajith G. S
Internal Guide

Ms. Meera Rose Mathew
Coordinator

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

External Examiner

DECLARATION

I hereby declare that the project report **"HEALTH COMPANION"** is a bona fide work done at **Amal Jyothi College of Engineering Autonomous, Kanjirappally**, Affiliated to **APJ Abdul Kalam Technological University**, towards the partial fulfilment of the requirements for the award of the **Master of Computer Applications (MCA)** during the period from **December 10, 2024** to **March 27, 2025**.

Date: 24/03/2025
KANJIRAPPALLY

Jince Abraham
Reg: AJC23MCA-2036

ACKNOWLEDGEMENT

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

I owe a great depth of gratitude towards our Head of the Department **Rev.Fr.Dr. Rubin Thottupurathu Jose** for helping us. I extend my whole hearted thanks to the project coordinator **Coordinator Name** for her valuable suggestions and for overwhelming concern and guidance from the beginning to the end of the project. I would also express sincere gratitude to my guide **Guide Name** for his inspiration and helping hand.

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

Jince Abraham

ABSTRACT

.

The Health Companion represents a complete online healthcare solution that combines virtual consultations with medicine shopping and laboratory testing capabilities under one technological system. Regular patients can search for doctors by symptoms or specialties via the platform to book appointments with the capability of receiving systematic alerts through their mobile devices. Partnership between doctors and pharmaceutical shops becomes efficient through the platform's capability to let doctors handle appointments and prescriptions along with shop delivery orders. The diagnostic services of laboratories function through their offering of home testing procedures followed by automatic submission of test results to patient profiles. Users have the ability to rate healthcare professionals and services on the system in order to build trust and increase transparency levels. Through the platform all stakeholders experience a convenient end-to-end journey where patients can receive home delivered medicines and laboratory tests conducted by personnel at their request within their homes.

CONTENT

List of Abbreviations

HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
NoSQL	Not Only SQL (referring to non-relational databases)

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

The Health Companion System represents a modern digital management solution which combines doctor appointments with medication supply and laboratory facilities into one unified healthcare system. Patients can utilize the platform to find doctors either by symptoms or specialties while also booking appointments through which they receive notifications by email or SMS or mobile alerts. Doctors use the system to schedule appointments and furnish medical prescriptions and keep contact with pharmacological supply associations for medication distribution. The platform supports diagnostics testing at laboratories which permits examination at home followed by automatic result transmission to patient profiles. Healthcare professionals as well as medical service providers receive increased transparency through the platform by enabling users to rate and review their performance. The system provides remote access to vital healthcare services that combines home medications delivery with diagnostic tests which guarantees accessibility and patient care improvement and convenience for users.

1.2 PROJECT SPECIFICATION

HealthCompanion delivers different features to various user groups starting with patients and continuing with doctors and pharmacies and laboratories.:

Users

- **Patient**
    - **Login and Registration**: Users can establish secure profiles during login procedures while performing registration steps to regulate their health information management.
    - **Search Doctor**: Patients can perform doctor searches by entering symptoms or selecting from doctor specializations or one of their numerous medical categories.
    - **Appointment Booking**: The system allows patients to reserve consultations with doctors who have their individual preference.
    - **Medication Purchase**: prescription Purchase is available through this system once the consultation phase concludes so patients can place orders for prescribed medications at affiliated pharmacy locations.
    - **Review and Rating**: Healthcare transparency increases through the patient review and rating feature which enables doctor assessments based on consultation experiences.
- **Doctor**
    - **Login and Registration**: The platform provides protected entry points for profile

generation and healthcare service management through its registration and login functionalities.

  - **Schedule Management**: Through the feature doctors gain the capability to both establish and adjust their slots when conducting online consultations. After prescribing medicines following consultations doctors enable patients to get their medications from the network of partner pharmacies.
  - **Prescription Management**: The system allows doctors to modify their appointment schedules when adjusted according to their requirements.
  - **Rescheduling**: Doctors have the ability to adjust and reschedule their consultation timings as needed.

- **Pharmaceutical Shops**
  - **Inventory Management**: Users can maintain inventory control and prescription execution records through the inventory management system.
- **Laboratory**
  - **Lab Report Management**: The system enables secure reporting processes that enables users to upload results into patient profiles.
  - **Test Requests**: The system enables employees to operate test requests along with track and manage incoming requests for laboratory tests.
- **Admin**
  - **Manage User**: The administrative staff uses their Manage User authority to ensure both security and efficiency within the system.
  - **Verify Doctors**: The role of admins consists in doctor verification to confirm their professional qualifications for delivering quality medical care.

Technology Stack

- **Front End**: The user interface uses React.js to construct its front end delivering a dynamic responsive experience to platform users.
- **Back End**: Users interact with the Node.js back-end system to achieve processed consultation and prescription management and user interaction processing.
- **Database**: The platform stores patient doctor and consultation information through the NoSQL MongoDB database which provides extended flexibility and database scalability.

CHAPTER 2

SYSTEM STUDY

2.1 INTRODUCTION

2.2

Health Companion System advances healthcare services by combining medical consultation, drug prescription and diagnostic testing within an accessible interface. This research analyzes medical treatment structures as well as their weaknesses before delivering a total solution that enhances patient participation and system entry.

2.3 EXISTING SYSTEM

2.2.1 NATURAL SYSTEM STUDIED

Healthcare facilities manage traditional off-the-grid services that consist of doctor-to-patient care alongside pharmaceutical medicine distribution combined with laboratory diagnostic diagnosis services. These include:

- **In-person Doctor Visits:** The healthcare practice includes patients coming to clinic or hospital buildings for their medical consultations.
- **Pharmacy Purchases:** People obtain drugs through neighborhood pharmacies to complete their medication needs.
- **Laboratory Visits:** Patients undertake laboratory tests as well as access test results by visiting these facilities.
- **Manual Records:** Medical professionals keep patient documentation in manual physical documents.
- **Community Support:** Patients ask health information to their family members and healthcare workers who function as part of the community support network.

2.2.2 DESIGNED SYSTEM STUDIED

Digital healthcare systems provide modern solutions for medical consultations and medicine ordering and laboratory testing operations. Notable examples include:

- **Telemedicine Platforms:** Services like Practo and Teladoc for online consultations.
- **E-Pharmacy Services:** The 1mg and Medlife online prescribing services offer e-pharmacy capabilities to users.
- **Diagnostic Platforms:** Websites like Dr. Lal Path Labs for lab test bookings.
- **Patient Portals:** The healthcare system allows patients to use hospital portals for booking appointments and accessing health records.

2.4 DRAWBACKS OF EXISTING SYSTEM

- Rural areas together with patients facing mobility issues face difficulties accessing health care.
- Inefficiencies due to time-consuming processes.
- Risk of errors in paper-based records.
- Limited communication options for urgent needs.
- Fragmented services with no unified patient data.

2.5 PROPOSED SYSTEM

HealthCompanion solves cost issues in online healthcare by developing an intuitive system that provides affordable solutions for patient and physician care. The proposed system will enhance consulting procedures to provide healthcare at better costs while maintaining quality standards. The following system components will merge cost reduction with service accessibility enhancement:

- **Competitive Pricing for Consultations**: The system allows patients to choose doctors for consultations based on their budget because it provides medical professionals at different price points. The system provides complete consultation prices at initial interaction to enable patients to decide on medical services beforehand.
- **Reduced Travel and Time Costs**: Remote healthcare access through the system reduces healthcare expenses by removing requirements for patients to meet doctors face-to-face thus eliminating travel together with work absence and accommodation costs..
- **Efficient Time Slot Management for Doctors**: Doctors can maximize their use of time during consultation slots through better organization thus growing their availability while avoiding extra administrative work. The service cuts down expenses for doctors as well as their patients combined.
- **Scalable Online Platform**: The deployment of React.js front end technologies together with Node.js back end and MongoDB data storage systems exhibits operational flexibility that makes the platform affordable for user-base growth management.
- **Reduced Operational Costs for Doctors**: Online functionality allows doctors to lower their operational spending on clinics to reduce consultation prices that patients must pay.
- **No Hidden Fees**: Users of HealthCompanion will receive clear information about all consulting fees and medication costs and examination expenses before making any payments during their transactions. Every patient fee can be seen upfront because

HealthCompanion will not add any concealed expenses thus helping users budget healthcare expenses easily.

HealthCompanion enables users to access quality healthcare through affordable pricing and efficient service and easily obtainable operations that resolve existing issues with high online healthcare fees.

2.6 ADVANTAGES OF PROPOSED SYSTEM

- **Comprehensive Healthcare Solution:** A single platform provides complete healthcare services which eliminates the requirement of different services.
- **Affordability:** The proposed system provides affordable consultation expenses medication pricing and testing fees at costs which exceed those found on the majority of current platforms.
- **Personalized Healthcare:** The platform recommends personalized healthcare solutions by processing patient health information which enhances customer satisfaction.
- **Data Integration:** A unified health profile for better management of healthcare journeys.
- **Enhanced Security and Privacy:** Robust data protection systems provided within the platform ensure complete patient confidentiality.
- **Time-Saving:** The system decreases patient and healthcare provider waiting times while creating time efficiency.
- **Accessibility:** This system enables access from any time as well as any location providing essential convenience particularly for patients in remote regions and individuals with mobility issues..
- **Community Support:** The system benefits from community support because patients who rate and review services help patients make smarter healthcare decisions while building trust in the entire system.

CHAPTER 3
REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDY

Every software development process needs an essential feasibility assessment to assess project feasibility and define operational needs and forecasted effects. Tests verify both operational and user need fulfillment and optimal resource utilization of the system. Patients can receive doctor consultations from an internet-based tool developed by the Health Companion project which also facilitates both remote medication prescription and laboratory testing service access. Project development commences following positive approval from the feasibility study which includes evaluations of technical functionality and operational requirements together with economic viability assessments.

3.1.1 ECONOMICAL FEASIBILITY

The project needs a cost-benefit analysis to determine its financial sustainability. The cost-effectiveness of the project depends on developing a balance between project development expenses and sustained advantages.

3.1.2 TECHNICAL FEASIBILITY

The section tests how well selected technologies fulfill the system requirements for performance together with scalability needs..

Key Considerations:

- **Frontend:**
  Users can experience effortless interactions through **React.js** as it helps build responsive components which support interface creation across patients and doctors and other stakeholders. The modular structure of React makes it possible to reuse code sections and complete projects faster.

- **Backend**
  Node.js along with Express creates a backend system which scales to manage real-time operations including doctor consultations along with prescription orders. The asynchronous operating model enables efficient processing of various requests at a time thus meeting the requirements of an expanding user population.

3.1.3 BEHAVIORAL FEASIBILITY

Users interact with the system effectively through behavioral feasibility which addresses system satisfaction and task efficiency as well as user satisfaction. We evaluate behavioral feasibility of the HealthCompanion system against user needs through assessments based on patients as well as doctors and pharmaceutical shops and laboratories and administrators.

1. Patients

- **User Requirements:**
  - Patients require a search engine and appointment booking system which incorporates an interface that enables effortless access to all features including physician discovery and medicine ordering and medical results readability..
  - Users must achieve straight-forward access to notifications and personal medical history and test results by using an intuitive interface design.
- **Feasibility:**
  - Users receive smooth and dynamic interactions with a basic interface through React.js which powers the front-end section. The system design enables patients to find doctors using symptoms or areas of expertise and book schedule visits and obtain medications in a hassle-free manner.
  - Through React components the user experience enhances since these elements deliver live feedback including appointment and order notifications.
  - Patients experience better health record management thanks to dashboards in simple user interfaces that prevent them from searching for test outcomes or medical records.

2. Doctors

- **User Requirements:**
  - Doctors require an efficient system for scheduling patient consultations and medication prescriptions together with test laboratory management. Doctors require scheduling programs to allow them flexibility when rescheduling their appointments.
- **Feasibility:**
  - By implementing React.js doctors can benefit from the development of a schedule management feature which enables smooth creation and edition of clinical time slots through a highly responsive calendar display.
  - Allows medical practitioners to create prescriptions and send pharmacy requests that enable them to complete their tasks rapidly.

- The doctor will benefit from React-based tools which guarantee secure input and accurate information entry for both prescriptions and lab test orders.
- Scheduled consultations will acquire dynamic modification capabilities to enhance the flexibility of consultative rescheduling operations.

3. Pharmaceutical Shops

- **User Requirements:**
  - A new system should exist for pharmaceutical shops to organize their stock and medication search capabilities and delivery procedures..
- **Feasibility:**
  - Through React.js pharmaceutical shops obtain an inventory dashboard which lets them execute fast medicine searches and fulfill stock control and real-time updates of availability.
  - Notifying and tracking ordering systems integrated into delivery management ensure pharmacies can perform efficient medicine delivery services.
  - The system requires integrated simple navigation tools with clear management features that decrease the complexity of inventory and order handling for pharmacies.

4. Laboratories

- **User Requirements:**
  - Test laboratories need to take charge of receiving test orders and updating patient profiles by adding results while modifying test costs.
- **Feasibility:**
  - An interactive user interface available for laboratories will help them manage their test requests with efficiency. React components help users access patient information efficiently while supporting the handling of both open and finished tests and aiding result uploading operations.
  - Laboratory workers can maintain test price information through clearly designed React.js forms which prevent mistakes during edits.
  - Dynamic forms will manage secure result uploads for patient profiles by ensuring proper data input.

5. Admin

- **User Requirements:**
  - System-wide statistics and user registration and verification of doctors and pharmacies as well as laboratory and user account administration lie within admin

responsibilities.

- **Feasibility:**
    - The React.js front end component enables swift data selection across system databases using administrator dashboards. The system uses simple workflows which allow administrators to authenticate doctors together with labs and pharmacies.
    - The management system becomes more efficient through React.js because it simplifies the process of handling user accounts and credential security protocols.

3.1.4 FEASIBILITY STUDY QUESTIONNAIRE

3.1.4.1 Project Overview

The HealthCompanion platform functions as an internet-based healthcare solution which unites medical service users with healthcare providers through its whole digital framework. Through this platform patients gain access to doctors for consultations while they can both buy medicines and get laboratory services plus instant notifications. Alongside its core functionality the project considers expanding through ML-driven solutions for doctor recommendations and a medical chatbot for instant healthcare support. The main functionality of this system includes delivering a healthcare system which prioritizes simplicity along with customization and unified services.

3.1.4.2 System Scope

The design of HealthCompanion includes a complete healthcare platform yet it provides space for customers to add ML-based recommendation capabilities. HealthCompanion will serve all groups who access healthcare services including patients in addition to doctors pharmacists and lab technicians and health system administrators. HealthCompanion functions through maximizing healthcare efficiency by presenting integrated patient-professional services such as scheduling meetings and medicine control while processing lab examination results.

3.1.4.3 Target Audience

- **Patients:** The patients who need healthcare treatments seek consultations together with lab tests testing and medicine purchases comprise the primary audience..

- **Doctors:** Doctors operate appointments while performing consultations and issuing medical prescriptions to patients..

- **Pharmaceutical Shops:** Providers managing medicine inventory and sales.

- **Laboratories:** Facilities handling diagnostic tests and delivering results.

- **Administrators:** System managers ensuring smooth operations and managing users.

3.1.4.4 Modules

1. **User Authentication:** Secure login and user management for patients, doctors, labs, pharmacies, and admins.
2. **Consultation Scheduling:** The consultation scheduling system includes a platform where patients find doctors based on their specialties for arranging their required appointments.
3. **Prescription Management:** Patients have access to prescribed medicines collected from the integrated pharmaceutical shops network after doctors provide medical consultations.
4. **Lab Test Services:** Lab Test Services let doctors order assessments which patients can appear online.
5. **Notifications:** Scheduled patients receive automated alerts which notify them of upcoming medical sessions as well as laboratory results and medicine delays via notifications.
6. **Rating System:** A rating system enables patients to evaluate healthcare services for assurance of quality control.
7. **Admin Features:** The administrative section allows managers to track all systems and control user administration together with transaction monitoring.

3.1.4.5 User Roles

- **Admin:**
  - Manages the entire system.
  - The system performs monitoring duties to track users together with services which operate throughout the platform.
  - Oversees transactions and system performance.
- **Patient:**
  - Books appointments with doctors.
  - Purchases medicines from pharmaceutical shops.
  - Patient views laboratory test outcomes after receiving medical consultations through the system.
  - Provides ratings for healthcare services.

- **Doctor:**
    - Manages appointments and schedules.
    - Provides consultations and prescribes medicines.
    - The doctor both requests medical laboratory tests and evaluates the test results which come back.
- **Pharmaceutical Shop:**
    - The staff member at this position handles medicine inventory management while processing all medicine orders placed by patients.
- **Laboratory:**
    - The platform ships request test results which becomes accessible to doctors and patients only.

3.1.4.6 System Ownership

The ownership of HealthCompanion belongs to its developer yet the team expects joint collaborations with medical experts and healthcare organizations to enhance patient care services.

3.1.4.7 Industry/Domain

The framework belongs to healthcare and wellness operations and includes digital healthcare administration together with doctor-patient interaction and medical services and laboratory testing.

3.1.4.8 Data Collection Contacts

- **Sr. Dr. Lizamma Joseph**: General Physician, Holy Cross Hospital, 9496628983
- **Sr. Laeitta**: Pharmacist, Holy Cross Hospital, 9497454590
- **Sr. Delmi Paul**: Administrator, Holy Cross Hospital, 9074705223

3.1.4.9 Questionnaire for Data Collection

1. **What are the most common health concerns patients consult you for?**

- *Answer 1:* "Most patients consult me for routine checkups, chronic conditions like diabetes, and seasonal illnesses."
- *Answer 2:* "I also see many patients for mental health concerns, especially stress-related issues."

2. **How do you currently manage patient appointments and medical records?**

- *Answer 1:* "I use a combination of manual logs and a simple online scheduling app."

- *Answer 2:* "I rely on an electronic health record (EHR) system, but it's not fully integrated with pharmacies and labs."

3. **What features do you find most useful in digital healthcare platforms?**

- *Answer 1:* "The ability to view patient history and previous prescriptions is very helpful."
- *Answer 2:* "Online appointment booking and prescription management features save time for both doctors and patients."

4. **How do you handle prescriptions and medicine delivery for patients?**

- *Answer 1:* "I write prescriptions that patients take to pharmacies themselves."
- *Answer 2:* "Some patients ask for online prescription services, but I currently don't use those features."

5. **What challenges do you face in managing patient interactions digitally?**

- *Answer 1:* "Many systems are fragmented, so I can't easily share patient data between doctors, pharmacies, and labs."
- *Answer 2:* "Ensuring the privacy of patient data while offering personalized services is a major concern."

6. **How important is the integration of pharmacy and lab services into a healthcare platform?**

- *Answer 1:* "Very important. It allows patients to access all necessary services from one place, improving their experience."
- *Answer 2:* "It reduces delays in receiving lab results and medicines, ensuring timely treatment."

7. **What kind of personalized recommendations would benefit your patients?**

- *Answer 1:* "Suggestions for preventive health checkups and lifestyle changes based on medical history."
- *Answer 2:* "Diet and exercise recommendations tailored to chronic conditions like hypertension or diabetes."

8. **How do you manage communication with patients outside consultations?**

- *Answer 1:* "I rely on phone calls or emails, but it’s not very efficient."
- *Answer 2:* "Some patients prefer messaging services, but I don't have a dedicated system for that."

9. **What are the most common health-related concerns when it comes to medicine availability?**

- *Answer 1:* "Patients often face issues with stock availability for prescription medicines."
- *Answer 2:* "Timely delivery of medicines, especially for chronic conditions, is a concern."

10. **How would you rate the importance of data privacy in healthcare systems?**

- *Answer 1:* "Data privacy is a top priority. Patients need to trust that their medical data is safe."
- *Answer 2:* "It’s very important, but systems also need to be user-friendly."

3.1.5 GEOTAGGED PHOTOGRAPH

3.1.5.1 MEETING WITH GENERAL PHYSICIAN

GPS Map Camera
Google
koovapally, Kanjirapally 686518, Kerala, India
Holy Cross Hospital
Lat 9.51018° Long 76.824797°
11/01/25 11:50 AM GMT +05:30

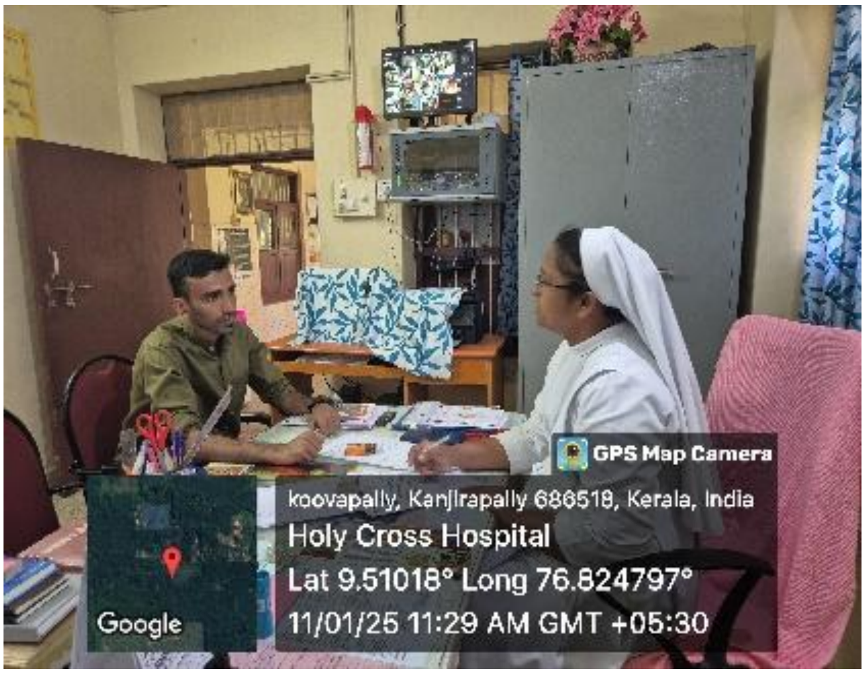
Sr. Dr. Lizamma Joseph: General Physician, Holy Cross Hospital, 9496628983

3.1.5.2 MEETING WITH PHARMACIST

GPS Map Camera
koovapally, Kanjirapally 686518, Kerala, India
Holy Cross Hospital
Lat 9.51018° Long 76.824797°
11/01/25 12:06 PM GMT +05:30
Google

Sr. Laeitta: Pharmacist, Holy Cross Hospital, 9497454590

3.1.5.3 MEETING WITH ADMINISTRATOR


GPS Map Camera
koovapally, Kanjirapally 686518, Kerala, India
Holy Cross Hospital
Lat 9.51018° Long 76.824797°
11/01/25 11:29 AM GMT +05:30
Google

Sr. Delmi Paul: Administrator, Holy Cross Hospital, 9074705223

3.2 SYSTEM SPECIFICATION

3.2.1 Hardware Specification

Processor	- i3
RAM	- 4 G B
Hard disk	- 4 G B

3.2.2 Software Specification

Front End	- React
Back End	– Node js
Database	- Mongo
Client on PC	- Windows 7 and above
Technologies used	- JS, HTML5, TAILWIND, NOSQL

3.3 SOFTWARE DESCRIPTION

3.3.1 Frontend: React.js

To build its user interfaces the HealthCompanion platform needs a perfect frontend solution that React.js delivers by acting as a JavaScript library for creating dynamic interfaces. The platform keeps its platform consistently efficient thanks to developer-reusable user interface components. The key benefits together with significant characteristics of React.js include:

- **Component-Based Architecture**: The platform implements React.js components as part of its component-based architecture to include features such as appointment scheduling with doctor search and user profile functions. By organizing this pattern developers achieve better maintenance capabilities and scalability since they can handle separate components independently.
- **Virtual DOM**: The virtual DOM in React.js delivers improved application speed by only refreshing important parts of the user interface instead of reloading everything on the page. The system delivers smooth functions throughout all data-intensive operations between user activities and medical record access.
- **Interactive User Experience**: The HealthCompanion platform delivers an interactive user

experience because it combines real-time features with appointment availability updates and instant notifications and dynamic form validation capabilities from React.js.

- **State Management**: State management implementation in React becomes possible through the useState hook and useEffect which allows efficient control of data complexity including user status maintenance and appointment record modification.
- **Cross-Platform Compatibility**: Through its ability to reengineer platform operations React.js creates identical user experiences that run across all platforms and desktops and mobile phones and tablets.

3.3.2 Backend: Node.js and MongoDB

Node.js operates as a runtime environment that summons JavaScript for server-side operations which matches precisely with the requirements of HealthCompanion backend development. The combination of Node.js server environment with MongoDB NoSQL database forms an efficient solution which delivers strong scalability for data management and user request processing capabilities. Key features and benefits include:

- **Asynchronous Processing**: Node.js runs multiple demands in parallel through its event-based I/O processing model which ensures that system responsiveness remains optimal even when many users simultaneously make bookings or watch doctor profiles.
- **RESTful API Integration**: Node.js allows developers to build RESTful APIs for frontend-backend communication through its RESTful API Integration method. These application programming interfaces (APIs) support user authorization functions and the retrieval of doctor schedules as well as appointment scheduling operations.
- **Data Management with MongoDB**: MongoDB stores healthcare records using its flexible data format based on JSON which provides excellent handling capabilities for dynamic health database records. The platform grows efficiently because user information including prescribing schedules and appointment data can be stored as individual documents that can be easily retrieved from the system.
- **Scalability and Flexibility**: Through their synergy Node.js and MongoDB enable the platform to grow efficiently while sustaining additional users and services. Additional medical personnel additions and increased patient appointments can be implemented by the system without requiring fundamental changes to its base operations framework.
- **Security**: Security measures through Node.js and MongoDB include data encryption along with authentication tokens and access controls that deliver protection to user personal medical records.

HealthCompanion delivers a strong high-performing platform which fulfills contemporary medical service requirements using frontend React.js and backend combination of Node.js with MongoDB.

CHAPTER 4

SYSTEM DESIGN

4.1 INTRODUCTION

HealthCommunity's system design structure establishes the framework of architecture as well as the components and their functional relations within the platform. OceanExpress aims to transform functional requirements into a structured blueprint which demonstrates the method through which system modules will communicate their functions between each other.

The selection process for an appropriate technology stack along with defining the database structure stands at the forefront of system requirements to execute user requests efficiently. The implementation integrates the frontend components using React.js with the backend components using Node.js and links both to the database powered by MongoDB.

Users transmit information through RESTful APIs to retrieve data from a server that represents the database component of the system. User authentication and doctor schedule management and prescription processing functions exist in the backend section but the frontend section delivers sensitive responsive interfaces to users.

A discussion about Unified Modeling Language (UML) diagrams will follow this part as they help depict system architecture through visual displays of system entities and between-entity interactions.

4.2 UML DIAGRAM

The essential visual tools for observing system design and functional behavior are Unified Modeling Language diagrams. The UML diagrams for HealthCompanion platform consist of three main diagrams:

4.2.1 USE CASE DIAGRAM

The diagram displays user role-system relations through the Use Case Diagram. Each type of user receives an outline describing their actions on different components from login to appointment booking to doctor schedule management and user review and ratings. For example:

- **Patient**: Users can perform these operations through the patient interface including login followed by searching doctors then booking appointments followed by reviewing doctors.
- **Doctor**: Login, Manage Schedule, Prescribe Medication, Adjust Schedule.
- **Laboratory**: The system allows the laboratory team to direct Lab Technicians as well as perform acceptance and rejection of test orders while inserting lab outcomes to patient medical profiles.

- **Pharmacy:** Pharmacists manage the pharmaceutical stock while processing medical orders for prescription drugs and organize home delivery services.
- **Admin**: The Admin user group possesses functionality to verify physicians while managing users through the dashboard and to access statistical data.

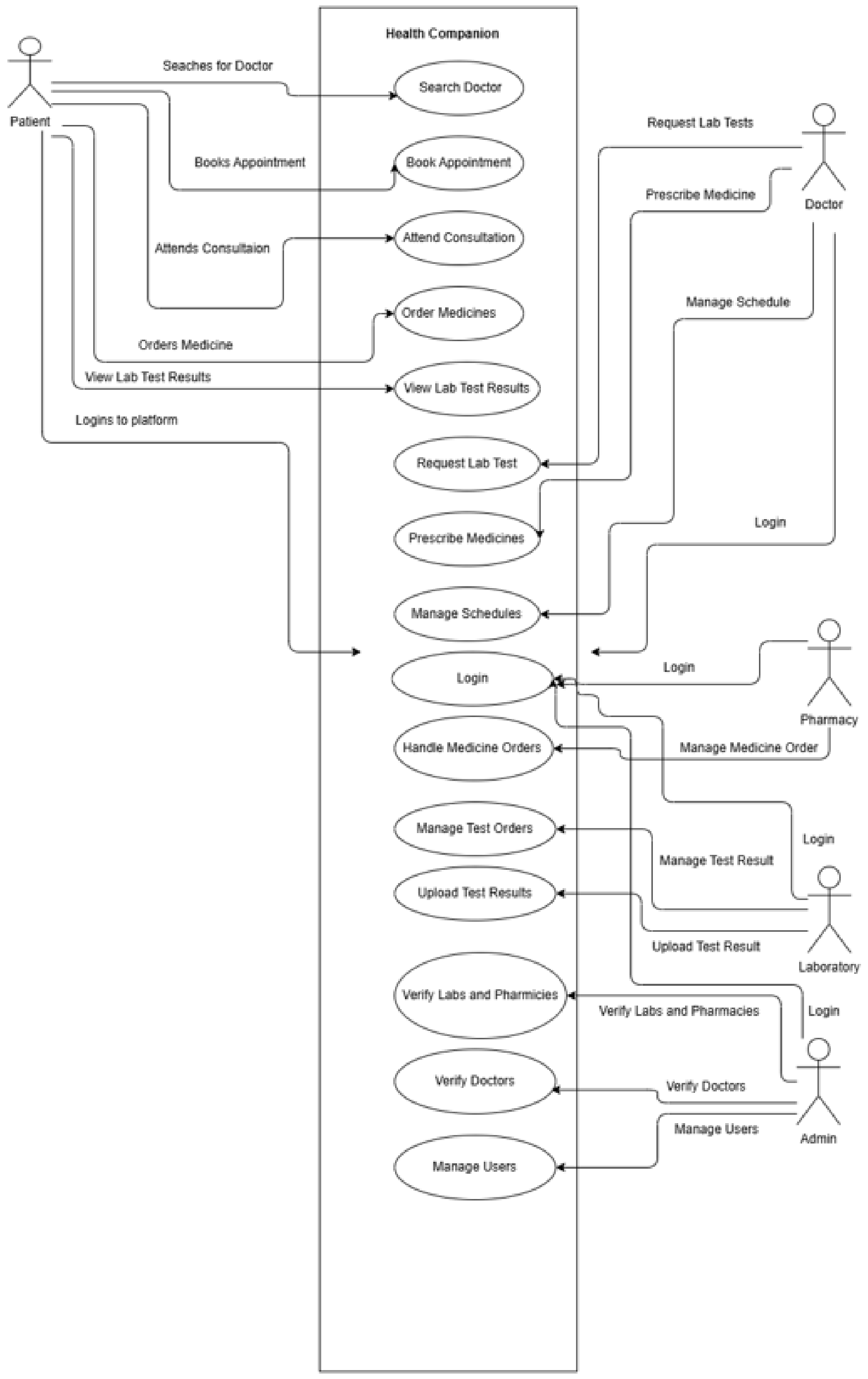


Figure 4.2.1.1

4.2.2 SEQUENCE DIAGRAM

The Sequence Diagram shows the operational process flow during important system interactions. For example, the sequence of actions during a doctor appointment booking:

Doctor Appointment Booking

1. Patient searches for a doctor.
2. The system displays all available doctors together with their available times.
3. The patient uses the system t o reserve a time slot and complete appointment confirmation.
4. System maintains the availability records while sending alerts to Patient along with Doctor.
5. Doctors check their appointments and modify their availability.

Medication Prescription & Delivery

1. Prescription from the doctor.
2. Prescription system goes to Pharmacy.
3. Dispensing pharmacy fills / ships medication Ordering of Laboratory Test

Laboratory Test Request

1. A diagnostic test is requested by patient.
2. Request is forwarded to Laboratory.
3. Laboratory schedules a test and collects samples.
4. Test results are uploaded to the profile of the patient.

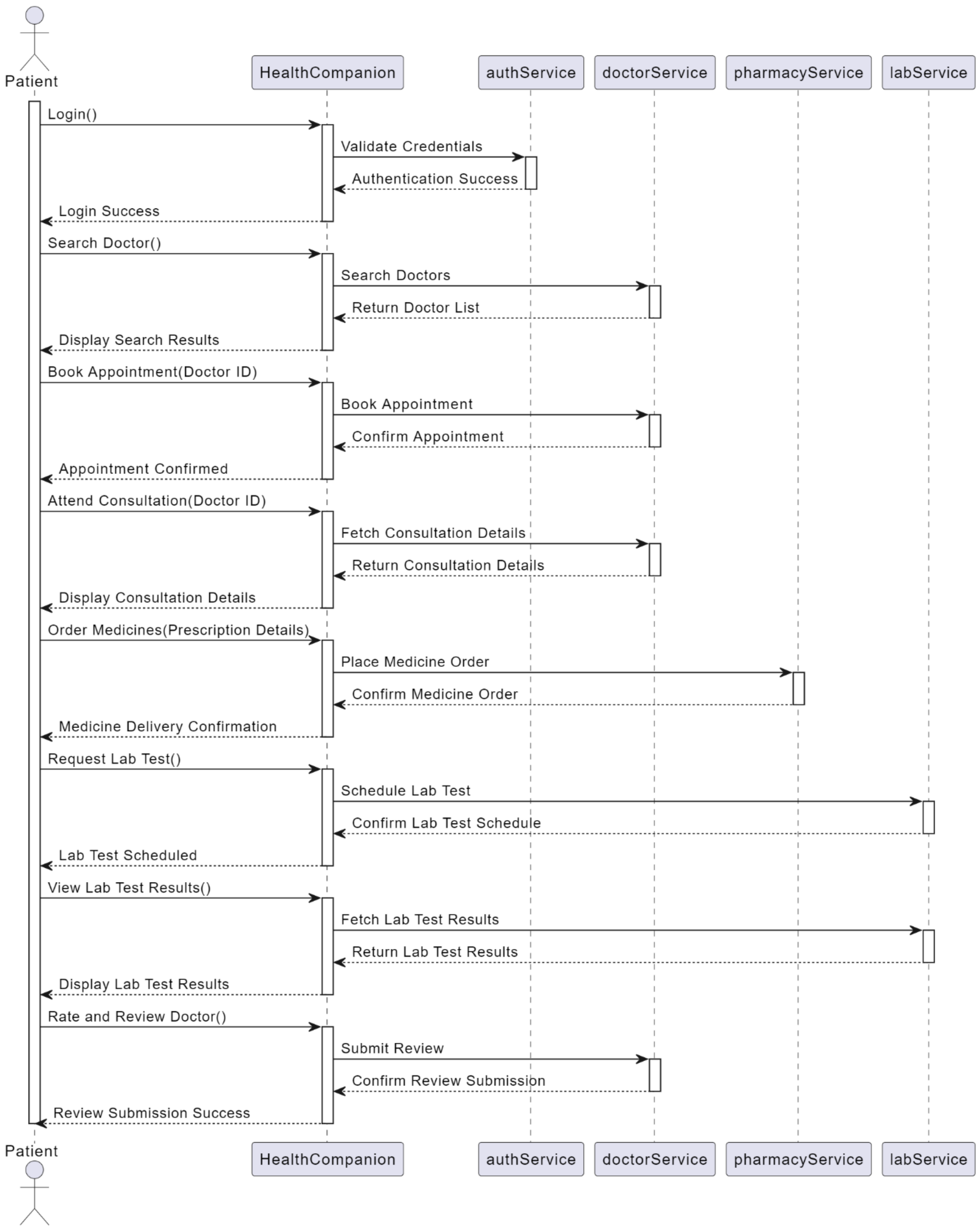

Patient
HealthCompanion
authService
doctorService
pharmacyService
labService
Login()
Validate Credentials
Authentication Success
Login Success
Search Doctor()
Search Doctors
Return Doctor List
Display Search Results
Book Appointment(Doctor ID)
Book Appointment
Confirm Appointment
Appointment Confirmed
Attend Consultation(Doctor ID)
Fetch Consultation Details
Return Consultation Details
Display Consultation Details
Order Medicines(Prescription Details)
Place Medicine Order
Confirm Medicine Order
Medicine Delivery Confirmation
Request Lab Test()
Schedule Lab Test
Confirm Lab Test Schedule
Lab Test Scheduled
View Lab Test Results()
Fetch Lab Test Results
Return Lab Test Results
Display Lab Test Results
Rate and Review Doctor()
Submit Review
Confirm Review Submission
Review Submission Success

Figure 4.2.2.1

4.2.3 STATE CHART DIAGRAM

State Chart Diagram is a behavioral model to design/ depict how an entity (user, system component/process etc.) can be in state and also what transition they will experience upon events within a system between these states. In project term State Chart Diagram picturizes how user or a system component moving from state to different states in its life circle. For example, a user account can move through Registered → Verified → Logged In -> SessionExpired Similarly the system might have statesIdle (Default), Processing Request, Data Being Displayed,,, Error Handling and so on triggered on particular events or user action. This State Chart Diagram helps to visualize the dynamic behavior of the system, so that all interactions or changes in state are fulfilled ensuring more robust and predictable functionality.

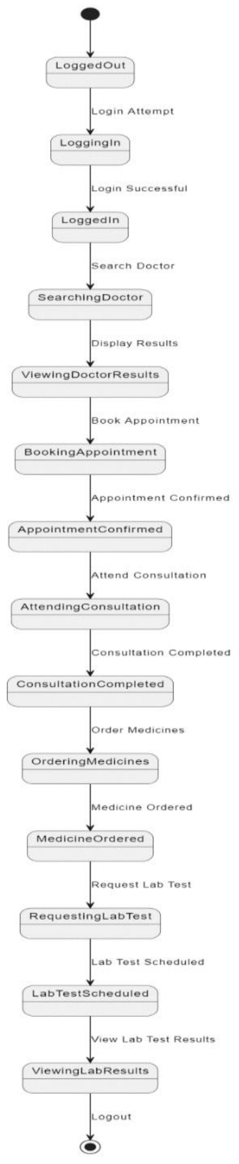


Figure 4.2.3.1

4.2.4 ACTIVITY DIAGRAM

HealthCompanion The Activity Diagram pictorially depicts the sequential operation of certain major processes in healthcare such as doctor consultations, medication prescriptions and requests for the laboratory test. This captures things like doctor search, reserving an appointment, validating patient demographic information, processing prescriptions and relay of test results Decision points, e.g. time when users have to choose slots or confirm delivery phases makes sure those user interacting with the system smooth. This well-fired workflow identifies bottlenecks and improves user experience in all kinds of services supporting health provided by HealthCompanion.

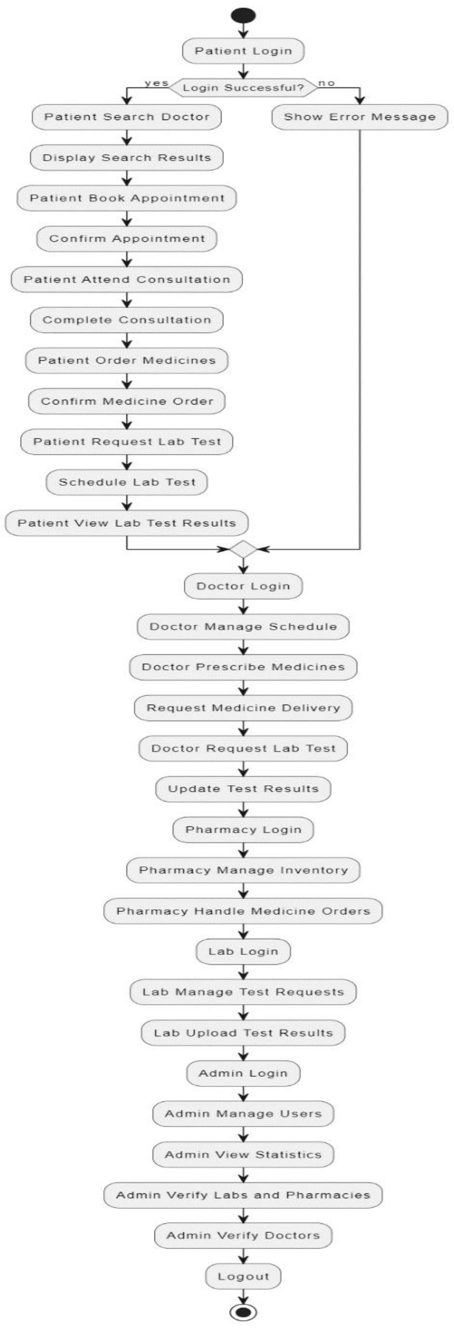


Figure 4.2.4.1

4.2.5 CLASS DIAGRAM

The Class Diagram describes the basic classes or objects that form the core of the system and their properties and behaviors. It has classes like:

- **User:** Attributes (userid, name, email, rol and methods login()| register())
- **Doctor:** doctorID, specialization and availableSlots Methods : createSlot(), rescheduleAppointment().
- **Patient:** Patient(patientID, medicalHistory, info) and calls (findDoctor()), bookAppointment()).
- **Appointment:** Attributes (appointmentID, date, time and status) methods (bookAppointment(), cancelAppointment()).
- **Prescription:** Attributes (prescriptionId,medicines, dosage).. methods GeneratePrescription( ), sendToPharmacy()).
- **Laboratory:** testTypes and reports(Object labID) methods(requestTest(), uploadResults()).

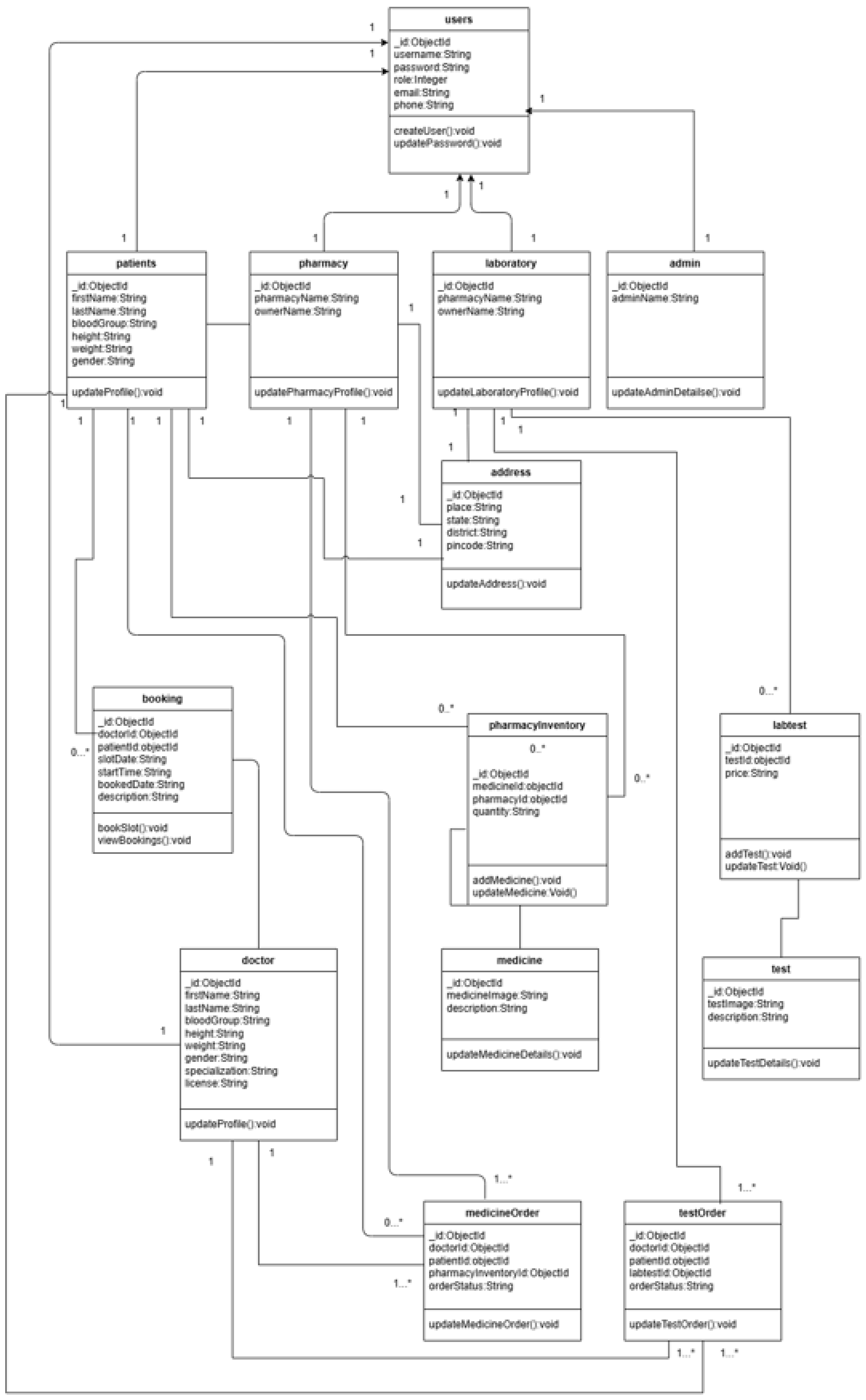

users
_id:ObjectId
username:String
password:String
role:Integer
email:String
phone:String
createUser():void
updatePassword():void
patients
_id:ObjectId
firstName:String
lastName:String
bloodGroup:String
height:String
weight:String
gender:String
updateProfile():void
pharmacy
_id:ObjectId
pharmacyName:String
ownerName:String
updatePharmacyProfile():void
laboratory
_id:ObjectId
pharmacyName:String
ownerName:String
updateLaboratoryProfile():void
admin
_id:ObjectId
adminName:String
updateAdminDetailse():void
address
_id:ObjectId
place:String
state:String
district:String
pincode:String
updateAddress():void
booking
_id:ObjectId
doctorId:ObjectId
patientId:objectId
slotDate:String
startTime:String
bookedDate:String
description:String
bookSlot():void
viewBookings():void
pharmacyInventory
_id:ObjectId
medicineId:objectId
pharmacyId:objectId
quantity:String
addMedicine():void
updateMedicine:Void()
labtest
_id:ObjectId
testId:objectId
price:String
addTest():void
updateTest:Void()
doctor
_id:ObjectId
firstName:String
lastName:String
bloodGroup:String
height:String
weight:String
gender:String
specialization:String
license:String
updateProfile():void
medicine
_id:ObjectId
medicineImage:String
description:String
updateMedicineDetails():void
test
_id:ObjectId
testImage:String
description:String
updateTestDetails():void
medicineOrder
_id:ObjectId
doctorId:ObjectId
patientId:objectId
pharmacyInventoryId:ObjectId
orderStatus:String
updateMedicineOrder():void
testOrder
_id:ObjectId
doctorId:ObjectId
patientId:objectId
labtestId:ObjectId
orderStatus:String
updateTestOrder():void
1
0...*
0..*
1...*

Figure 4.2.5.1

4.2.6 OBJECT DIAGRAM

The Object Diagram Provides a system at instance view of Object i.e., it portrays the instances of significant class(es) and their relationship to each other at the point in time. That can help demonstrate the interactions between objects in an appointment booking use case. For example:

- **Patient Object**: patientID: P001, name: John Doe, email: john@example.com
- **Doctor Object**: doctorID: D101, name: Dr. Smith, specialization: Cardiology
- **Appointment Object**: appointmentID: A2024, date: 2025-03-12, time: 10:30 AM
- **Prescription Object**: prescriptionID: PR567, medicine: Aspirin, dosage: 1 tablet/day

These objects real-time data in system, providing visualizations of patients-doctor/appointments interactions through which a certain punctual moment happens.

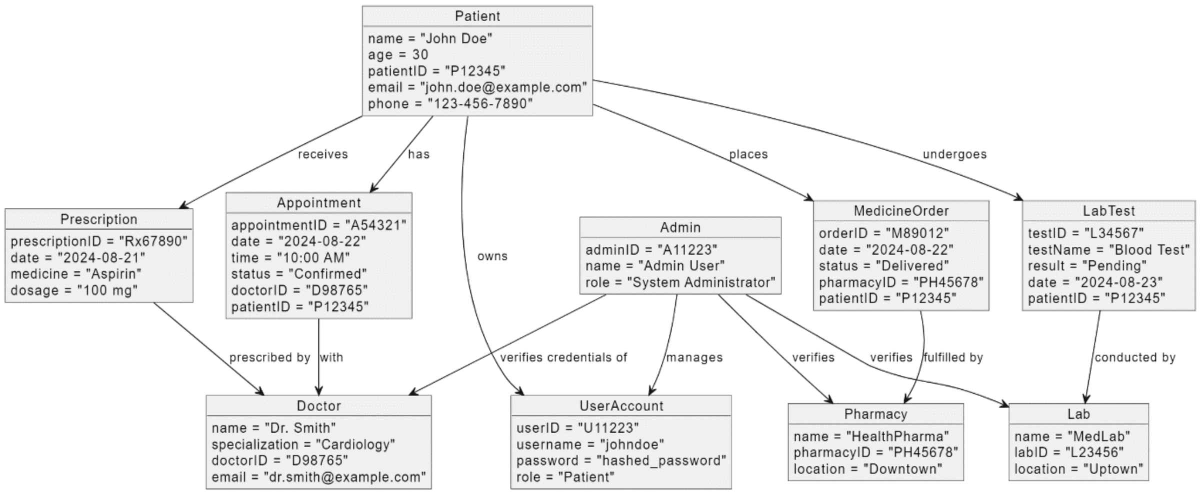


Figure 4.2.6.1

4.2.7 COMPONENT DIAGRAM

Overview of the software components in HealthCompanion platform, interacting with each other is provided at the Component Diagram level. It describes the system as collections of component modules, and identifies the dependency between them.

For HealthCompanion, the different parts are basically:

- Frontend (React.js) → This handles the UI and user interactions; searching doctors, making appoinments, profile and feedback administrative necessities.
- Backend (node .js, express) — handling business logic, API endpoints, database operations etc. This component handles the authentication, appointment scheduling and prescription management.
- Database (MongoDB): All the data is stored in the database: patient / doctor information, appointment info and feedback.
- Authentication Service: a unit to deal with user Login/Registration and perform Authentication using JWT token based.
- Notification Service: The component which is responsible of pushing notifications to an email and SMS about appointment confirmations, reminders and feedback.

The Component Diagram depicts these interactions with the components For example the backend API is contacted by frontend when a user books an appointment and commits them to database as well notification service with the required notifications..

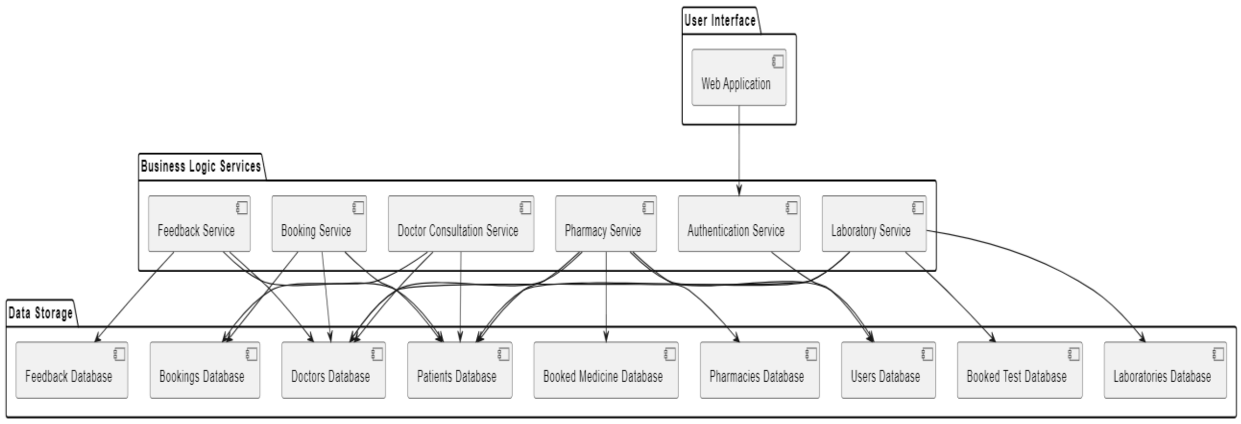


Figure 4.2.7.1

4.2.8 DEPLOYMENT DIAGRAM

Deployment Diagram describes the HealthCompanion system deployment in hardware nodes like servers and client systems both. It provides a visual depiction of where the software components are in the system architecture and their logical and physical connections.

- Client Node: The various devices that users (patient, doctors and admins) interface with the system like PC, tablets or smartphones. React frontend gets deployed on this node and talks to the backend as an HTTP serve.
- Application Server: This is Node backend Node.js deployment The the server processes client requests, executes business logic, communicates with the DB and handles all sorts of API end points such as auth, appointments or user-management
- Database Server: Node that runs this MongoDB database. It keeps and fetch the data like users profiles, appointments, prescriptions.
- Notification Service Server: Don't miss an appointment or another crucial event: This way the appointment server that can send you real-life updates directly in your RSS feed and notify you via email/sms.

Deployment Diagram depicts traversing of the data, how it flows between these servers, and the way they use clients to provide the functionalities in the system.

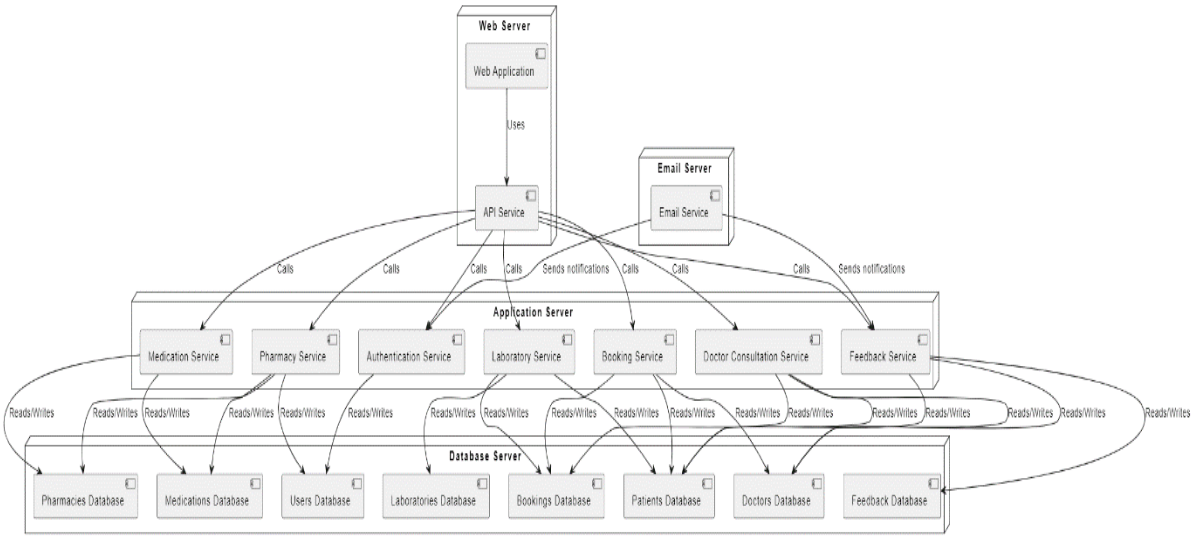


Figure 4.2.8.1

4.2.9 COLLABORATION DIAGRAM

The Collaboration Diagram for showing the interaction between objects or components in the system — shows objects collaborating and working towards a goal. The collaboration of important entities and components in HealthCompanion have been critical for realizing all the user actions (e.g., schedule an appointment, manage doctor schedules) to be done.

Appointment Booking process has:

1. The Search Doctor component, which is part of the Patient (user interface) will take an end user input to search for some doctors.
2. Search Doctor will be having an interaction with the Doctor and Schedule components to retrieve the time slots.
3. First, based on the requirement, the patient clicks on slot and book slot and triggers the collaboration between Appointment, Doctor's Schedule and Notify Service to book slot and notify people.
4. Appointment details are stored in the Database, and Notification Service sends Appointment Confirmation. Alternatively other interactions reflect medication dispensing, lab request for lab and drug delivery in a smooth manner within the system collaboration for healthcare experience

Similarly, other interactions include prescription handling, lab test requests, and medicine delivery, ensuring a seamless healthcare experience through efficient system collaboration.

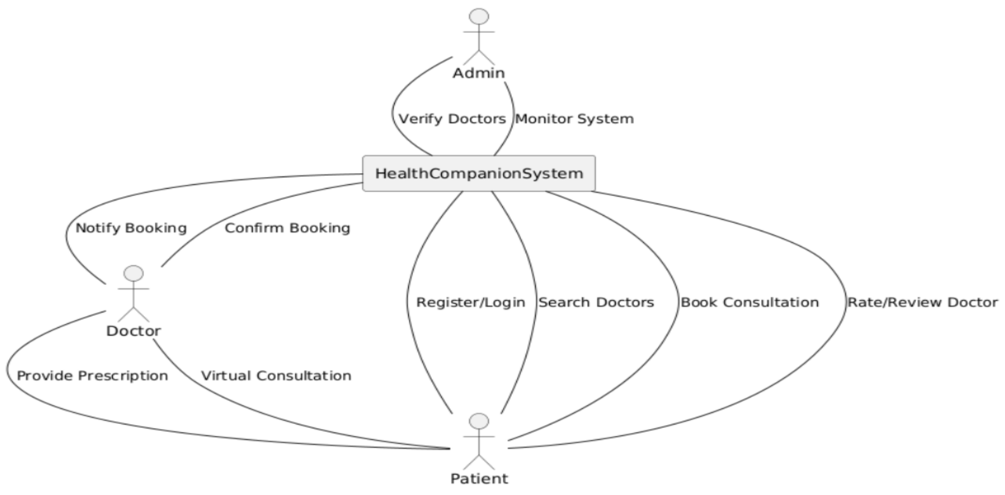
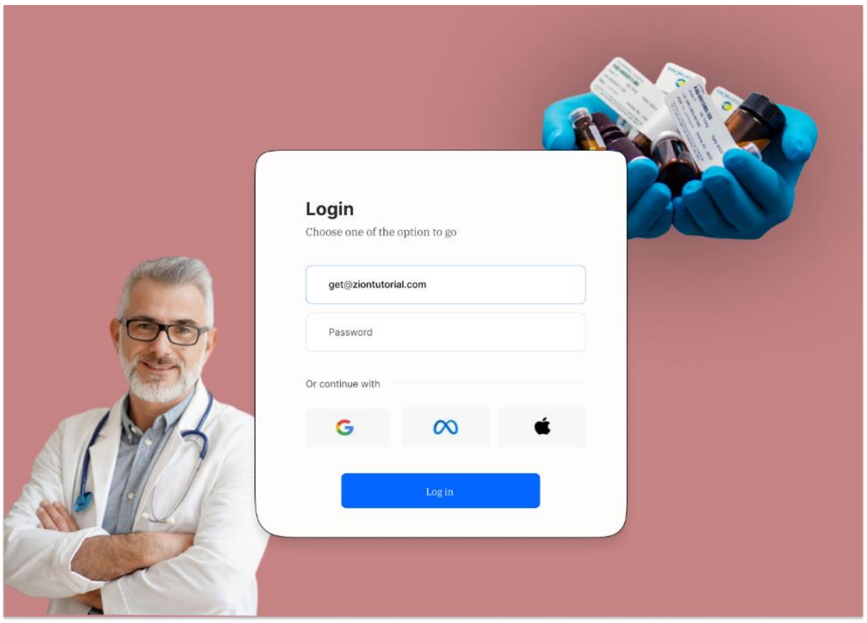


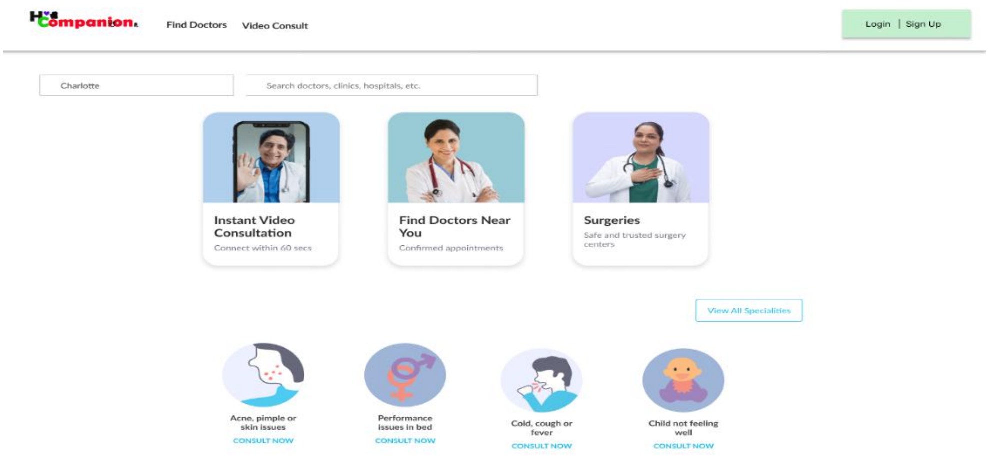
Figure 4.2.9.1

4.3 USER INTERFACE DESIGN USING FIGMA

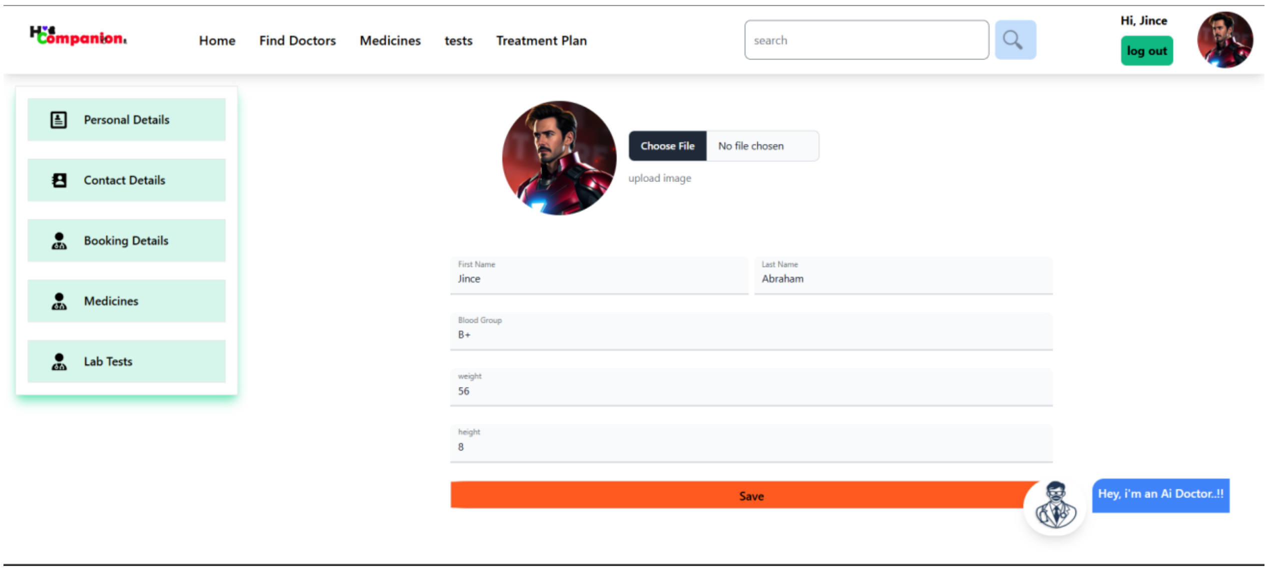
4.3.1 FORM DESIGN : LOGIN PAGE



4.3.2 FORM DESIGN : PATIENT DASHBOARD

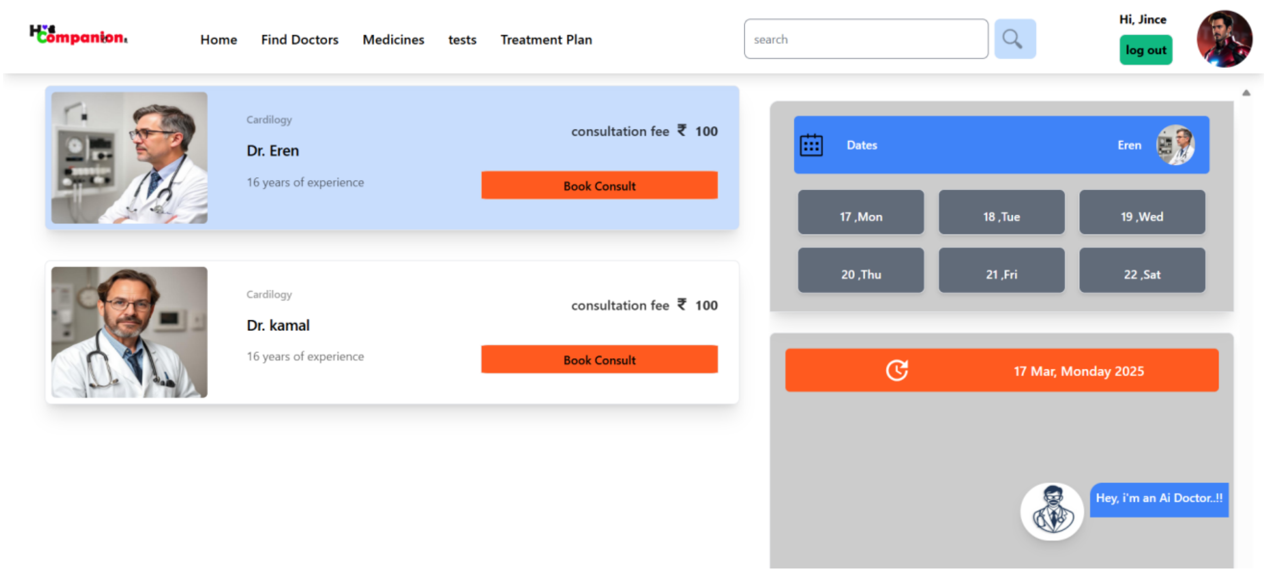


4.3.3 FORM DESIGN : PATIENT PROFILE PAGE



First Name: Jince
Last Name: Abraham
Blood Group: B+
weight: 56
height: 8

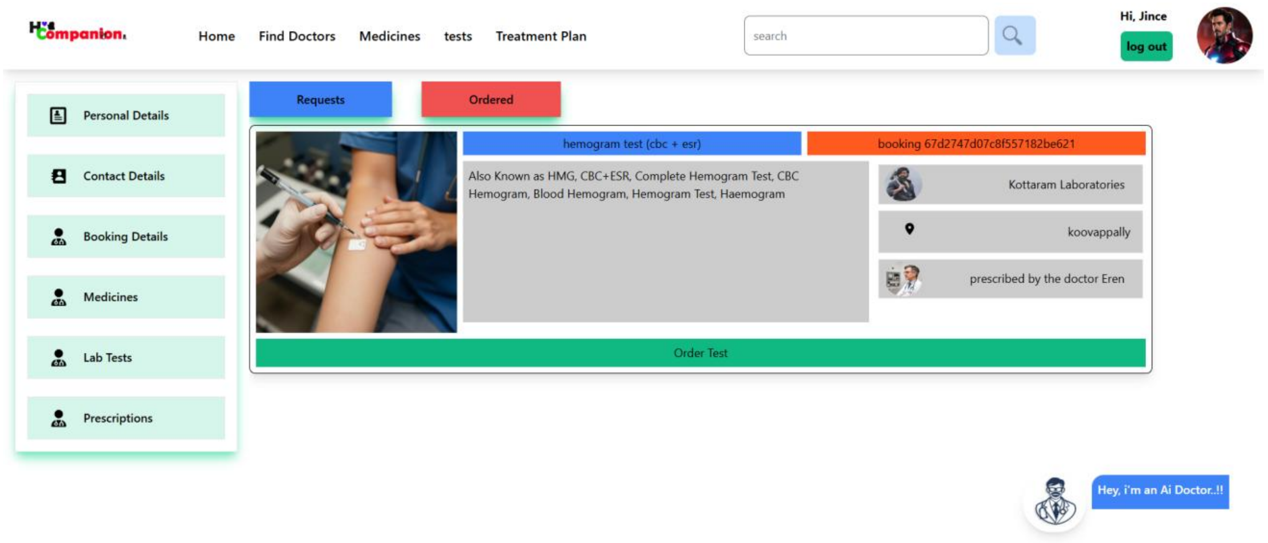
4.3.4 FORM DESIGN : PATIENT BOOKING PAGE



Cardiology
Dr. Eren
16 years of experience
consultation fee ₹ 100

Cardiology
Dr. kamal
16 years of experience
consultation fee ₹ 100

4.3.5 FORM DESIGN : PATIENT TESTS REQUESTED FROM DOCTOR PAGE



hemogram test (cbc + esr)

booking 67d2747d07c8f557182be621

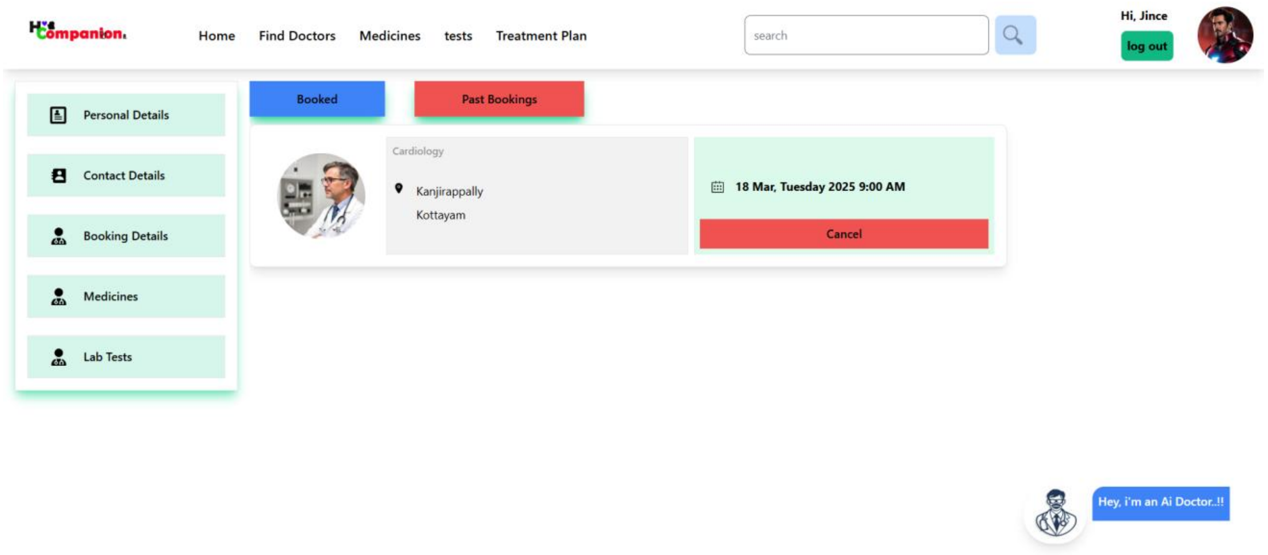
Also Known as HMG, CBC+ESR, Complete Hemogram Test, CBC Hemogram, Blood Hemogram, Hemogram Test, Haemogram

Kottaram Laboratories

koovappally

prescribed by the doctor Eren

4.3.6 FORM DESIGN : PATIENT CONSULTATION FOR TODAY PAGE



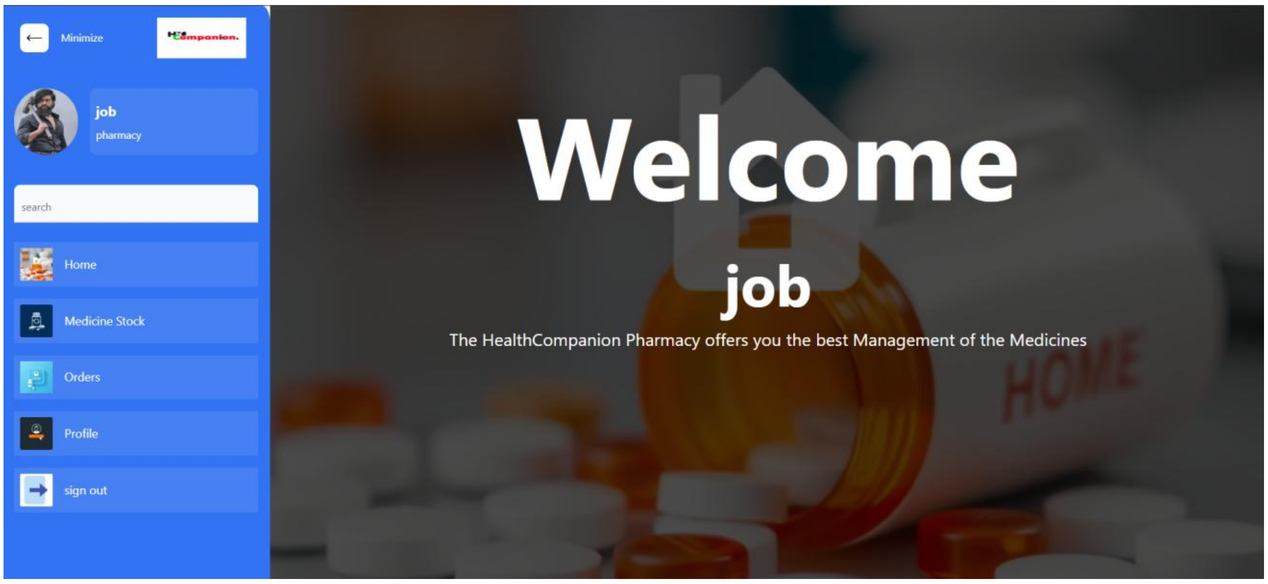
Cardiology

Kanjirappally

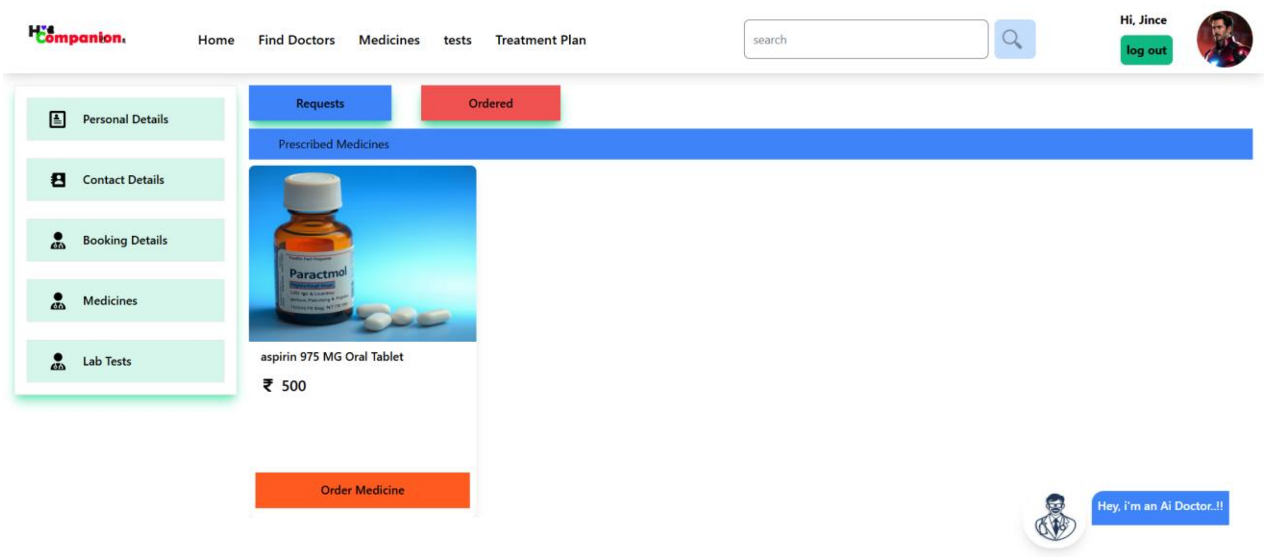
Kottayam

18 Mar, Tuesday 2025 9:00 AM

4.3.7 FORM DESIGN : PHARMACY DASHBOARD PAGE


Minimize
job
pharmacy
search
Home
Medicine Stock
Orders
Profile
sign out
Welcome
job
The HealthCompanion Pharmacy offers you the best Management of the Medicines

4.3.8 FORM DESIGN : PATIENT MEDICINES REQUESTED FROM DOCTOR PAGE



4.4 DATABASE DESIGN

4.4.1 Mongo DB

The HealthCompanion system utilizes Mongo DB a NoSQL document oriented database for storing, retrieving, and managing data. MongoDB, on the other hand does not work in terms of tables and rows as you would in normal SQL based relational databases but instead with collections and documents which makes it way more flexible and scalable for complex hierarchically structured data. The flexibility is good for applications such as HealthCompanion in which data may be stored for different user types (patients, doctors and admins) in an efficient manner.

HealthCompanion MongoDB Collections:

1. **Users Collection**: This collection stores data for both patients and doctors. Each document contains information like userID, name, email, passwordHash, role (patient or doctor), and profileDetails.
2. **Appointments Collection**: Stores appointment details such as appointmentID, patientID, doctorID, date, time, and status (confirmed, pending, or completed).
3. **Doctors Collection**: Contains detailed information about each doctor, including doctorID, specialization, availableSlots, profileDetails, and reviews.
4. **Review Collection**: Stores patient reviews and ratings for doctors. Each document includes reviewID, patientID, doctorID, rating, and comment.

The storage of complex data structures, e.g., saving a single doctor's Time Slots(nested) within his/her document hence MongoDB document model might reduce the number of joins for each operation. This structure is designed to provide fast queries for real time applications like doctor search and appointment booking.

MongoDB horizontal scaling capabilities enable the system too grow and scale with growing loads as userbase grows which in turn ensures HealthCompanion platform can scale for any growth.

4.5 TABLE DESIGN

4.5.1 Tbl_users

Eg.Object id: id

No:	Fields	Type	Description
1	__id	objectID	Unique identifier
2	username	String	username
3	password	String	password
4	role	String	Admin/patient/doctor/pharmacy/laboratory
5	email	String	Email of the user
6	phone	String	Phone

4.5.2 Tbl_Patients

Eg.Object id: _id

No:	Fields	Type	Description
1	_id	objectID	Unique identifier
2	firstname	String	firstname
3	lastname	String	lastname
4	bloodgroup	String	bloodgroup
5	height	String	Height
6	weight	String	Weight
7	gender	String	Male/Female
8	altphone	String	Alternate Phone no
9	userId	String	Reference to the User table

4.5.3 Tbl_Doctors

Eg.Object id: _id

No:	Fields	Type	Description
1	_id	objectID	Unique identifier
2	firstname	String	firstname
3	lastname	String	lastname
4	bloodgroup	String	bloodgroup
5	height	String	Height
6	weight	String	Weight
7	gender	String	Male/Female
8	altphone	String	Alternate Phone no
9	specialization	String	Specialization
10	address	objectID	Reference to Address Table
11	license	Pdf	License of the doctor
12	educationDetails	document	Data related to Educational details of the doctor
13	status	String	(Approved or pending) status regarding the profile to be seen under the doctor profile in the system

4.5.4 Tbl_Booking

Object id: _id

No:	Fields	Type	Description
1	startTime	String	Start Time of the slot eg..5 pm for 5 pm – 6 pm
2	patientId	ObjectId	References patient table
3	doctorId	ObjectId	References doctor table
4	slotDate	String	Date
5	patientDescription	String	Description about the meeting
6	bookedDate	String	Date on which the booking done
7	bookedStatus	String	Status of meeting
8	paymentId	ObjectId	References payment table

4.5.5 Tbl_Pharmacy

Object id: _id

No:	Fields	Type	Description
1	userId	Object ID	References user table
2	pharmacyName	String	Name of the pharmacy
3	ownerName	String	Name of the owner
4	profileImage	String	Path to the profile image
5	addressId	String	Reference to the address table
6	status	String	Active status

4.5.6 Tbl_Laboratory

Object id: _id

No:	Fields	Type	Description
1	userId	Object ID	References user table
2	laboratoryName	String	Name of the pharmacy
3	ownerName	String	Name of the owner
4	profileImage	String	Path to the profile image
5	addressId	String	Reference to the address table
6	status	String	Active status

4.5.7 Tbl_Payment

Object id: _id

No:	Fields	Type	Description
1	payerId	Object Id	References the user table
2	receiverId	Object Id	References the user table
3	paymentCategory	String	Category for which under does the payment happens
4	paymentDetails	JSON	Razor pay fulfilled details

4.5.8 Tbl_Otp

Object id: _id

No:	Fields	Type	Description
1	email	String	email
2	otp	String	Otp sent
3	expiredAt	Date	Otp expiry time

4.5.9 Tbl_Medicine

Object id: _id

No:	Fields	Type	Description
1	medicineName	String	Name of the medicine
2	medicineType	String	Type of the medicine
3	description	String	Descriptio
4	medicineImage	String	Path to the medicine image
5	price	String	Price of the medicine
6	status	String	Active status of the medicine

4.5.10 Tbl_PharmacyInventory

Object id: _id

No:	Fields	Type	Description
1	medicineId	Object Id	Reference to the medicine table
2	pharmacyId	Object Id	Reference to the pharmacy table
3	costPrice	String	Cost of the medicine
4	medicineImage	String	Path to the medicine image
5	sellingPrice	String	Selling price of the medicine
6	Stock	String	Stock of the medicine
7	status	String	Active status of the medicine

4.5.11 Tbl_MedicineOrder

Object id: _id

No:	Fields	Type	Description
1	pharmacyInventoryId	Object Id	Reference to the pharmacy Inventory table
2	paymentId	Object Id	Reference to the payment table
3	patinetId	Object Id	References the patient table
4	doctorId	Object Id	References the doctor table
5	bookingId	Object Id	References the booking table
6	orderStatus	String	Status of the order

4.5.12 Tbl_Test

Object id: _id

No:	Fields	Type	Description
1	testName	String	Name of the test
2	testDescription	String	Test description
3	testImage	String	Path to the test image

4.5.13 Tbl_LabTest

Object id: _id

No:	Fields	Type	Description
1	labId	ObjectId	Reference to the laboratory table
2	testId	ObjectId	Reference to the test table
3	atHome	Boolean	At home service status
4	atLab	Boolean	At Lab service status
5	priceHome	String	Price of the service at home
6	priceLab	String	Price of the service at lab

4.5.14 Tbl_testOrder

Object id: _id

No:	Fields	Type	Description
1	labTestId	ObjectId	Reference to the labTest table
2	paymentId	ObjectId	Reference to the payment table
3	patinetId	Object Id	References the patient table
4	doctorId	Object Id	References the doctor table
5	bookingId	Object Id	References the booking table
6	orderStatus	String	Status of the order

4.5.15 Tbl_prescription

Object id: _id

No:	Fields	Type	Description
1	disease	ObjectId	Reference to the labTest table
2	prescription	ObjectId	Reference to the prescription table
3	patinetId	Object Id	References the patient table
4	doctorId	Object Id	References the doctor table
5	bookingId	Object Id	References the booking table

4.5.16 Tbl_address

Object id: _id

No:	Fields	Type	Description
1	place	String	Place name
2	state	String	State name
3	district	String	Strict name
4	country	String	Country name
5	pincode	String	Pincode
6	housename	String	house name

4.5.17 Tbl_Feedback

Object id: _id

No:	**Fields**	**Type**	**Description**
1	_id	objectID	Unique identifier
2	doctorId	ObjectId	Reference to the doctor id
3	patientId	ObjectId	Reference to the patient id
4	rating	String	Ratings out of 5
5	comment	String	Comment

CHAPTER 5
SYSTEM TESTING

5.1 INTRODUCTION

System Testing is one of the very important phase of software development, In which the all the system is tested to prove there functionality and what it should output with given input based on needed requirements. It is the phase where we test the units and hardware/ integrated components as well as the HealthCompanion platform in general to find any defects and make sure that the system works all conditions correctly. System testing is done to validate the system functionality, performance and security of the system prior to its deployment to users.

Testing steps are in various stages it starts from the smallest components i.e (unit testing), to testing interactions of components (integration testing) all the way to testing the whole system (system testing). So that ultimately the HealthCompanion platform provides a secure, robust and user friendly platform to its users.

5.2 TEST PLAN

Test Plan Test Phase Approach for proper functioning of HealthCompanion platform at testing level to be outlined in the Test Plan. It outlines how the testing approaches, tools, schedules and resources are to be applied. For instance, the test plan will detail the types of test (unit testing, integration testing, validation testing, user acceptance testing, automation testing, selenium testing etc.)
The key objectives of the test plan include:

- Verifying that all the modules(functions) are working as expected(example user in, appoint booking, timing managment of a doctor etc..
- To do a good integration of frontend (React.js) → backend(Node. js) as well database (mongodb)
- Bugs, performance issues and the likes.. sorting these out.
- Makes sure security & privacy norms are followed.
- Validating that the system satisfies user requirement using acceptance testing.

5.2.1 UNIT TESTING

Unit Testing — which is the lowest level of testing, where individual components or unit of system is tested as standalone. A unit corresponds to small testable part of application (function, method, class in HealthCompanion for example).

For example:

- **Doctor Availability Service**: A working test for getting a ply of availabe slots from doctor.
- **Appointment Booking Function**: To make sure that the appointments are booked by the patients and all the inputs are validated.
- **Login Authentication**: Validating that all of your login step (Login , check credential, generate tokens) are working fine.

Unit testing makes sure that every piece is working as intended before we integrate into the whole system. It is commonly automated and usually done through testing frameworks such as Jest (for React) & Mocha/Chai (for Node.js).

5.2.2 INTEGRATION TESTING

Integration Testing focuses on verifying the interactions between different components or modules in the system. In the **HealthCompanion** platform, integration testing ensures that the front end, backend, and database work together seamlessly.

For example:

- **Appointment Booking Flow**: Integration Testing of React.Js frontend from where the user picks a doctor and Node.Js backend for appointment booking (saving the appointment details in MongoDB to the database).
- **Doctor Schedule Management**: Whether The Doctor Page Time Slots get updated and the patient scheduled appointments are getting updated properly.
- **Notification Service**: Ensure that notification and email & SMS notifications are triggered for confirmed appointments -Shared services notifaction service: to trigger email and SMS notifications for appointments confirmed.

In order to test the glue of different modules, integration testing finds defects related to the interaction where components are being combined (even if unit testing already passed for individual component).

5.2.3 VALIDATION TESTING OR SYSTEM TESTING

System/Validation Testing or System Testing in which the complete integrated system is tested to ensure all required functionality and works as per the real world conditions are met. Objective namely to ensure that the HealthCompanion platform meets the functional, performance and security requirements.

Key areas tested include:

- **Functional Testing**: Verifying that all features (e.g., appointment booking, doctor search, patient profile management) work as intended.
- **Performance Testing**: Ensuring the platform can handle a large number of users and appointments without performance degradation.
- **Security Testing**: Testing the system's ability to protect sensitive user data (e.g., patient profiles, medical records) and ensure secure authentication.
- **Usability Testing**: Ensuring the system is easy to use and navigate for patients and doctors.

System testing is performed in a real-world like environment, which deploys production without actual deployment.

5.2.4 OUTPUT TESTING OR USER ACCEPTANCE TESTING

Output Testing / User Acceptance Testing (UAT) is the last phase of testing- here the system is tested from end users side (patients, Doctors and admins) for validating if their expectations and requirements met**.**.

In the testing phase of the HealthCompanion system, UAT relates to:

- Patients who book appointments online, search for doctors, and get notifications for specific actions.
- Doctors who manage their schedules and see appointments along with medication prescriptions.
- Admin in managing users, doctor qualifications, and the operations of the system.

UAT will typically involve users interacting with the system to do real-life tasks-from calling an appointment to search for the doctor, up to receiving reminders-and everything else the system should offer. Any differences or failures in carrying out such work are reported, and feedback is taken to improve or enhance the system.

5.2.5 AUTOMATION TESTING

Automation Testing as its name implies, it executes tests automatically using specific software tools instead of using a person. The real feature here is the fact that it can save much manual work as well speed up testing and increase its coverage. Automation testing within the HealthCompanion system will be especially useful on repeating tasks like:

- **Login Functionality**: Automatically testing login scenarios such as valid credentials, invalid credentials, and a password reset.
- **Appointment Booking**: Automatically test various booking scenarios, with some edge cases like double booking and time slot conflicts

For UI tests, one tool uses Selenium, while Cypress can be used. Other frameworks, such as Jest or Mocha, can be employed to automate unit tests.

A Finally, the automated approach to testing will also improve the operational efficiency of the entire process of testing such that test and retest will never compromise or allow any new future changes to shrink the system.

5.2.6 SELENIUM TESTING

Selenium Testing is a widely used web-based testing automation framework, which permits testing of user interface and interactions of the HealthCompanion platform. Selenium acts out user actions by clicking buttons, filling out forms, or traversing a website.

Selenium can thus be used for testing in HealthCompanion for:

- **Patient Login**: Testing the login on different browsers.
- **Doctor Search**: Automating the search for a doctor by the symptoms or specialization and verifying the results.
- **Appointment Booking**: Simulating the entire booking operation from the selection of a doctor, the choice of time slots, to confirmation of an appointment.

By testing these platforms with Selenium, the front-end functionality is guaranteed to be retained and consistent across various browsers and devices, thereby becoming user-friendly.

Example:

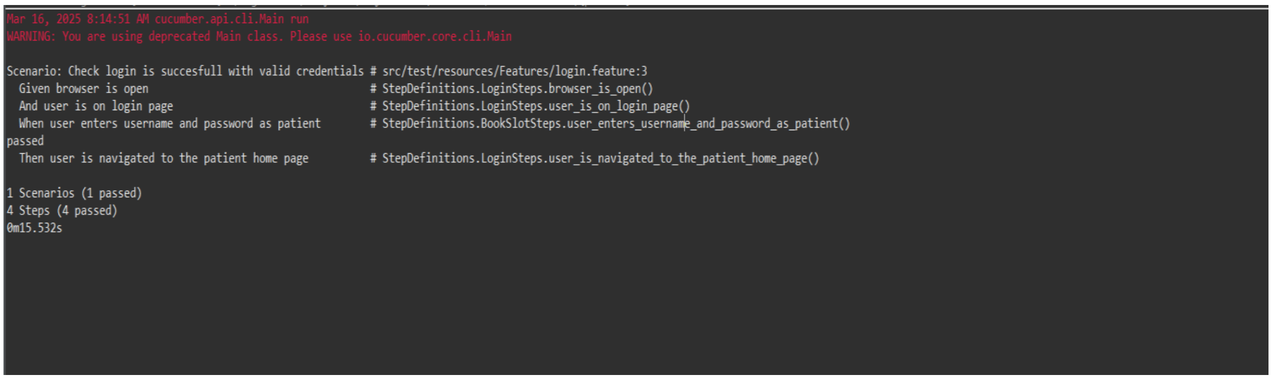
Test Case 1

Code

```
public class LoginSteps {
        public static WebDriver driver;
        public static void main(String[] args) {
        }
        @Given("browser is open")
        public void browser_is_open() throws InterruptedException {
                System.setProperty("webdriver.gecko.marionette","E:\\geckodriver-
v0.32.0-win64\\geckodriver.exe");
                driver=new FirefoxDriver();
                driver.manage().window().maximize();
                driver.navigate().to("http://localhost:5173/");
                Thread.sleep(2000);
          }
        @And("user is on login page")
        public void user_is_on_login_page() throws Exception {
                driver.findElement(By.id("id_loginNav")).click();
                Thread.sleep(2000);
        }
        @When("user enters username and password as admin")
        public void user_enters_username_and_password_as_admin() throws
        InterruptedException {
        driver.findElement(By.id("id_username")).sendKeys("jinceabraham2025@mc
        a.ajce.in");
          driver.findElement(By.id("id_password")).sendKeys("Jjjjj@11");
          Thread.sleep(2000);
        }
        @And("User clicks on login")
        public void user_clicks_on_login() throws InterruptedException {
          driver.findElement(By.id("id_loginButton")).click();
          WebDriverWait wait = new WebDriverWait(driver,
Duration.ofMinutes(1));
```

```
        WebElement sweetAlertOkButton =
wait.until(ExpectedConditions.visibilityOfElementLocated(By.cssSelector(".swal2-
confirm.swal2-styled")));
        sweetAlertOkButton.click();
            Thread.sleep(2000);
          }
          @Then("user is navigated to the home page")
          public void user_is_navigated_to_the_home_page() throws
InterruptedException {
                if(driver.findElement(By.id("id_adminDashboard")).isDisplayed()) {
                      System.out.println("passed");
                }
                else {
                      System.out.println("failed");
                }
                Thread.sleep(2000);
                driver.quit();
          }
          @Then("user is navigated to the patient home page")
          public void user_is_navigated_to_the_patient_home_page() throws
InterruptedException{
                if(driver.findElement(By.id("id_patientDashboard")).isDisplayed()) {
                      System.out.println("passed");
                }
                else {
                      System.out.println("failed");
                }
                Thread.sleep(2000);
                driver.quit();
          }
}
```

Screenshot



```
Mar 16, 2025 8:14:51 AM cucumber.api.cli.Main run
WARNING: You are using deprecated Main class. Please use io.cucumber.core.cli.Main

Scenario: Check login is succesfull with valid credentials # src/test/resources/Features/login.feature:3
  Given browser is open                                    # StepDefinitions.LoginSteps.browser_is_open()
  And user is on login page                                # StepDefinitions.LoginSteps.user_is_on_login_page()
  When user enters username and password as patient        # StepDefinitions.BookSlotSteps.user_enters_username_and_password_as_patient()
passed
  Then user is navigated to the patient home page          # StepDefinitions.LoginSteps.user_is_navigated_to_the_patient_home_page()

1 Scenarios (1 passed)
4 Steps (4 passed)
0m15.532s
```

Eg.Test Report

Test Case 1

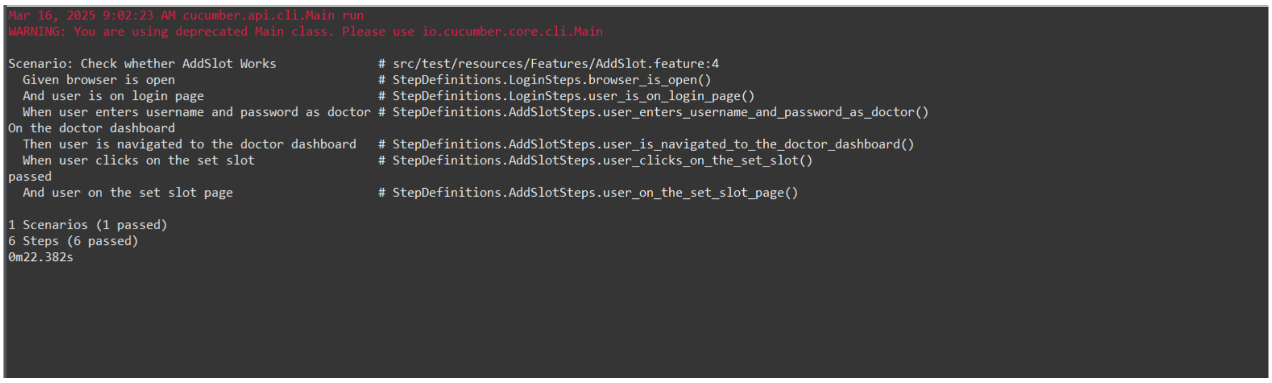
Project Name: Health Companion					
Login Test Case					
Test Case ID: Test_1			**Test Designed By: Jince Abraham**		
Test Priority(Low/Medium/High): Low			**Test Designed Date: 03/03/2025**		
Module Name: Login			**Test Executed By : Mr Ajith G S**		
Test Title : Login Testing			**Test Execution Date: 05/03/2025**		
Description: Check login functionality works on credentials			Test Executed Successfully		
Pre-Condition : User has valid username and password					
Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status(Pass/ Fail)**
1	Naviagate to the login page	NA	Login page loads successfully	Login page loads successfully	pass
2	Fill in username and password	“job” for username and “job” for password	Username and password fields are successfully filled in	Username and password fields are successfully filled in	pass
3	Click on the login Button	NA	Navigates to the dashboard based of the user role successfully	Navigates to the dashboard based of the user role successfully	pass
Post-Condition: user is redirected to dashboard					

Test Case 2:

Code

```
public AddSlotSteps() {
                driver=LoginSteps.driver;
        }
        @When("user enters username and password as doctor")
        public void user_enters_username_and_password_as_doctor() throws
InterruptedException {
        driver.findElement(By.id("id_username")).sendKeys("jinceabraham2025@mc
a.ajce.in");
          driver.findElement(By.id("id_password")).sendKeys("Jjjjj@11");
          driver.findElement(By.id("id_loginButton")).click();
          WebDriverWait wait = new WebDriverWait(driver,
Duration.ofMinutes(1));
     WebElement sweetAlertOkButton =
wait.until(ExpectedConditions.visibilityOfElementLocated(By.cssSelector(".swal2-
confirm.swal2-styled")));
     sweetAlertOkButton.click();
          Thread.sleep(2000);
        }

        @Then("user is navigated to the doctor dashboard")
        public void user_is_navigated_to_the_doctor_dashboard() {

                System.out.println("On the doctor dashboard");

        }
        @When("user clicks on the set slot")
        public void user_clicks_on_the_set_slot() throws InterruptedException {

                driver.findElement(By.id("id_navSlot")).click();
                Thread.sleep(2000);
        }
        @And("user on the set slot page")
        public void user_on_the_set_slot_page() throws InterruptedException {

                driver.findElement(By.id("id_slotDate_1")).click();
                Thread.sleep(2000);
                driver.findElement(By.id("id_btnaddslot_2")).click();
                Thread.sleep(2000);
                WebDriverWait wait = new WebDriverWait(driver,
Duration.ofMinutes(1));
                WebElement sweetAlertOkButton =
wait.until(ExpectedConditions.visibilityOfElementLocated(By.cssSelector(".swal2-
confirm.swal2-styled")));
                if(driver.findElement(By.cssSelector(".swal2-confirm.swal2-
styled")).isDisplayed()) {
                        System.out.println("passed");
                }
                sweetAlertOkButton.click();

                Thread.sleep(2000);

        }
```

Screenshot



```
Mar 16, 2025 9:02:23 AM cucumber.api.cli.Main run
WARNING: You are using deprecated Main class. Please use io.cucumber.core.cli.Main

Scenario: Check whether AddSlot Works             # src/test/resources/Features/AddSlot.feature:4
  Given browser is open                           # StepDefinitions.LoginSteps.browser_is_open()
  And user is on login page                       # StepDefinitions.LoginSteps.user_is_on_login_page()
  When user enters username and password as doctor # StepDefinitions.AddSlotSteps.user_enters_username_and_password_as_doctor()
On the doctor dashboard
  Then user is navigated to the doctor dashboard   # StepDefinitions.AddSlotSteps.user_is_navigated_to_the_doctor_dashboard()
  When user clicks on the set slot                # StepDefinitions.AddSlotSteps.user_clicks_on_the_set_slot()
passed
  And user on the set slot page                   # StepDefinitions.AddSlotSteps.user_on_the_set_slot_page()

1 Scenarios (1 passed)
6 Steps (6 passed)
0m22.382s
```

Test report

Test Case 2

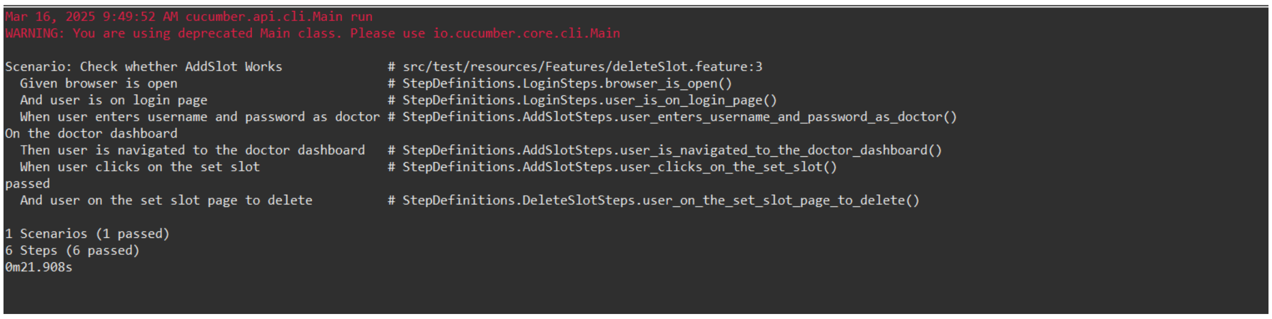
Project Name: Health Companion					
Add Slot Test Case					
Test Case ID: Test_2			**Test Designed By: Jince Abraham**		
Test Priority(Low/Medium/High): Low			**Test Designed Date: 03/03/2025**		
Module Name: Add Slot for Doctors			**Test Executed By : Mr. Ajith G S**		
Test Title : Add Slot Testing			**Test Execution Date: 05/03/2025**		
Description: Check add slot functionality			Test Executed Successfully		
Pre-Condition : User has valid username and password					
Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status(Pass/ Fail)**
1	Naviagate to the login page	NA	Login page loads successfully	Login page loads successfully	pass
2	Fill in username and password	“jince” for username and “jjjj@10” for password	Username and password fields are successfully filled in	Username and password fields are successfully filled in	pass
3	Click on the login Button	NA	Navigates to the doctor dashboard	Navigates to the doctor dashboard	pass
4	Clicks on the Slot Date	NA	Set slot page loads successfully	Stock Page loads successfully	pass
5	Choose a slot and clicks Add slot	NA	Slot is added successfully	Slot is added successfully	pass
Post-Condition: Slot is added					

Test Case 3:

Code

```
public DeleteSlotSteps() {
                driver=LoginSteps.driver;
        }
        @And("user on the set slot page to delete")
        public void user_on_the_set_slot_page_to_delete() throws
InterruptedException {
                driver.findElement(By.id("id_slotDate_1")).click();
                Thread.sleep(2000);
                driver.findElement(By.id("id_btnRemoveSlot_2")).click();
                Thread.sleep(2000);
                WebDriverWait wait = new WebDriverWait(driver,
Duration.ofMinutes(1));
                WebElement sweetAlertOkButton =
wait.until(ExpectedConditions.visibilityOfElementLocated(By.cssSelector(".swal2-
confirm.swal2-styled")));
                if(driver.findElement(By.cssSelector(".swal2-confirm.swal2-
styled")).isDisplayed()) {
                        System.out.println("passed");
                }
                sweetAlertOkButton.click();
                Thread.sleep(2000);
        }
```

Screenshot



```
Mar 16, 2025 9:49:52 AM cucumber.api.cli.Main run
WARNING: You are using deprecated Main class. Please use io.cucumber.core.cli.Main

Scenario: Check whether AddSlot Works             # src/test/resources/Features/deleteSlot.feature:3
  Given browser is open                           # StepDefinitions.LoginSteps.browser_is_open()
  And user is on login page                       # StepDefinitions.LoginSteps.user_is_on_login_page()
  When user enters username and password as doctor # StepDefinitions.AddSlotSteps.user_enters_username_and_password_as_doctor()
On the doctor dashboard
  Then user is navigated to the doctor dashboard  # StepDefinitions.AddSlotSteps.user_is_navigated_to_the_doctor_dashboard()
  When user clicks on the set slot                # StepDefinitions.AddSlotSteps.user_clicks_on_the_set_slot()
passed
  And user on the set slot page to delete         # StepDefinitions.DeleteSlotSteps.user_on_the_set_slot_page_to_delete()

1 Scenarios (1 passed)
6 Steps (6 passed)
0m21.908s
```

Test report

Test Case 3

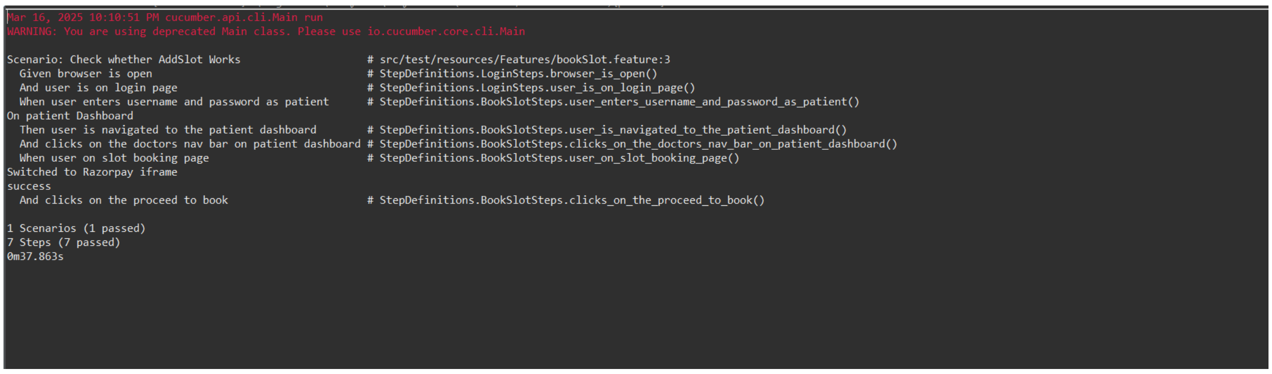
Project Name: Health Companion					
Delete Slot Test Case					
Test Case ID: Test_3			**Test Designed By: Jince Abraham**		
Test Priority(Low/Medium/High): Low			**Test Designed Date: 03/03/2025**		
Module Name: Delete Added Slot for Doctors			**Test Executed By : Mr. Ajith G S**		
Test Title : Delete Slot Testing			**Test Execution Date: 03/03/2025**		
Description: Check delete slot functionality			Test Executed Successfully		
Pre-Condition : User has valid username and password					
Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status(Pass/ Fail)**
1	Naviagate to the login page	NA	Login page loads successfully	Login page loads successfully	pass
2	Fill in username and password	“jince” for username and “jjjj@10” for password	Username and password fields are successfully filled in	Username and password fields are successfully filled in	pass
3	Click on the login Button	NA	Navigates to the doctor dashboard	Navigates to the doctor dashboard	pass
4	Clicks on a slot date	NA	Set slot page loads successfully	Set slot page loads successfully	pass
5	Choose a set slot and clicks on delete slot	NA	Slot is removed successfully	Slot is removed successfully	pass
Post-Condition: Added Slot is removed					

Test Case 4:

Code

```
@When("user enters username and password as patient")
        public void user_enters_username_and_password_as_patient() throws
InterruptedException {
                driver.findElement(By.id("id_username")).sendKeys("jince");
            driver.findElement(By.id("id_password")).sendKeys("Jjjjj@11");
            driver.findElement(By.id("id_loginButton")).click();
            WebDriverWait wait = new WebDriverWait(driver,
Duration.ofMinutes(1));
      WebElement sweetAlertOkButton =
wait.until(ExpectedConditions.visibilityOfElementLocated(By.cssSelector(".swal2-
confirm.swal2-styled")));
      sweetAlertOkButton.click();
            Thread.sleep(2000);
        }
        @Then("user is navigated to the patient dashboard")
        public void user_is_navigated_to_the_patient_dashboard() throws
InterruptedException {
            System.out.println("On patient Dashboard");
            Thread.sleep(2000);
        }
        @And("clicks on the doctors nav bar on patient dashboard")
        public void clicks_on_the_doctors_nav_bar_on_patient_dashboard() throws
InterruptedException {
                driver.findElement(By.id("id_navDoctors")).click();
                Thread.sleep(2000);
        }

        @When("user on slot booking page")
        public void user_on_slot_booking_page() throws InterruptedException {
                driver.findElement(By.id("id_doctor_0")).click();
                Thread.sleep(2000);
                driver.findElement(By.id("id_slotDate_1")).click();
                Thread.sleep(2000);
                driver.findElement(By.id("id_slotTime_0")).click();
                Thread.sleep(2000);
        }
        @And("clicks on the proceed to book")
        public void clicks_on_the_proceed_to_book() throws InterruptedException {
                driver.findElement(By.id("id_btnBookSlot")).click();
                Thread.sleep(4000);
                WebDriverWait wait = new WebDriverWait(driver,
  Duration.ofMinutes(1));
                WebElement razorpayIframe =
  wait.until(ExpectedConditions.presenceOfElementLocated(By.tagName("iframe")));
                driver.switchTo().frame(razorpayIframe);
                System.out.println("Switched to Razorpay iframe");
                WebElement continueButton =
  wait.until(ExpectedConditions.elementToBeClickable(By.cssSelector("button[data-
  test-id='add-card-cta']")));
                continueButton.click();
                Thread.sleep(4000);
                driver.findElement(By.name("pay_without_saving_card")).click();
                Thread.sleep(4000);
                WebElement paybtn=
  wait.until(ExpectedConditions.elementToBeClickable(By.cssSelector("button[validat
  eform='true']")));
                WebElement otpInput=
  wait.until(ExpectedConditions.elementToBeClickable(By.cssSelector("input[name='o
  tp']")));
                otpInput.sendKeys("12345");
                System.out.println("success");
  pay_continueButton);
        }
```

Screenshot



```
Mar 16, 2025 10:10:51 PM cucumber.api.cli.Main run
WARNING: You are using deprecated Main class. Please use io.cucumber.core.cli.Main

Scenario: Check whether AddSlot Works                   # src/test/resources/Features/bookSlot.feature:3
  Given browser is open                                 # StepDefinitions.LoginSteps.browser_is_open()
  And user is on login page                             # StepDefinitions.LoginSteps.user_is_on_login_page()
  When user enters username and password as patient     # StepDefinitions.BookSlotSteps.user_enters_username_and_password_as_patient()
On patient Dashboard
  Then user is navigated to the patient dashboard       # StepDefinitions.BookSlotSteps.user_is_navigated_to_the_patient_dashboard()
  And clicks on the doctors nav bar on patient dashboard # StepDefinitions.BookSlotSteps.clicks_on_the_doctors_nav_bar_on_patient_dashboard()
  When user on slot booking page                        # StepDefinitions.BookSlotSteps.user_on_slot_booking_page()
Switched to Razorpay iframe
success
  And clicks on the proceed to book                     # StepDefinitions.BookSlotSteps.clicks_on_the_proceed_to_book()

1 Scenarios (1 passed)
7 Steps (7 passed)
0m37.863s
```

Test report

Test Case 4

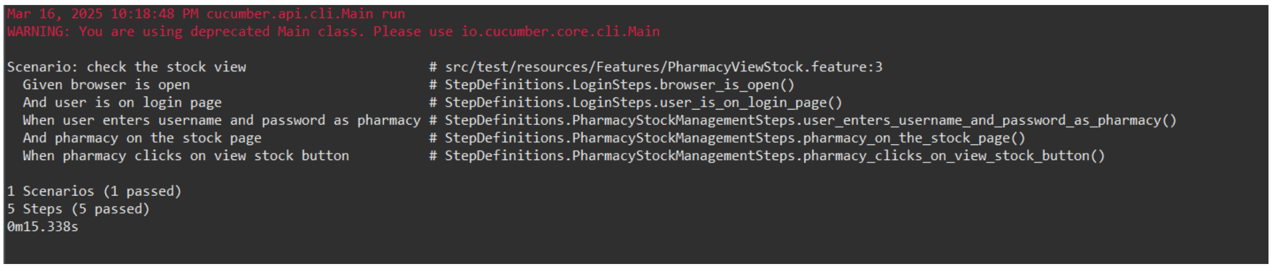
Project Name: Health Companion					
Book Slot Test Case					
Test Case ID: Test_4			**Test Designed By: Jince Abraham**		
Test Priority(Low/Medium/High): Low			**Test Designed Date: 03/03/2025**		
Module Name: Booking Slot for patients			**Test Executed By : Mr. Ajith GS**		
Test Title : Book Slot Testing			**Test Execution Date: 05/03/2025**		
Description: Check whether slot booking works for patient			Test Executed Successfully		
Pre-Condition : User has valid username and password					
Step	**Test Step**	**Test Data**	**Expected Result**	**ActualResult**	**Status(Pass/ Fail)**
1	Naviagate to the login page	NA	Login page loads successfully	Login page loads successfully	pass
2	Fill in username and password	“edward” for username and “jjjj@10” for password	Username and password fields are successfully filled in	Username and password fields are successfully filled in	pass
3	Click on the login Button	NA	Navigates to the patient dashboard	Navigates to the patient dashboard	pass
4	Clicks on the Doctors nav option	NA	Doctor page loads up successfully	Doctor page loads up successfully	pass
5	Choose a doctor and click on procced to book	NA	Booking page for the clicked doctor load up successfully	Booking page for the clicked doctor load up successfully	pass
6	Select a slot and clicks on book button	NA	Slot gets successfully booked after filling in a description and clicking a confirm button	Slot gets successfully booked after filling in a description and clicking a confirm button	pass
Post-Condition: Available Slot has been booked					

Test Case 5:

Code

```
@When("user enters username and password as pharmacy")
	public void user_enters_username_and_password_as_pharmacy() throws
InterruptedException {
	driver.findElement(By.id("id_username")).sendKeys("jince.experiment10@g
mail.com");
	    driver.findElement(By.id("id_password")).sendKeys("Jjjjj@11");
	    driver.findElement(By.id("id_loginButton")).click();
	    WebDriverWait wait = new WebDriverWait(driver,
Duration.ofMinutes(1));
        WebElement sweetAlertOkButton =
wait.until(ExpectedConditions.visibilityOfElementLocated(By.cssSelector(".swal2-
confirm.swal2-styled")));
        sweetAlertOkButton.click();
	    Thread.sleep(2000);
	}
@And("pharmacy on the stock page")
	public void pharmacy_on_the_stock_page() throws InterruptedException {

		driver.findElement(By.id("id_navMedicineStock")).click();
		Thread.sleep(2000);
	}
	@When("pharmacy clicks on view stock button")
	public void pharmacy_clicks_on_view_stock_button() {
		WebDriverWait wait = new WebDriverWait(driver,
Duration.ofMinutes(1));
		WebElement topItem=
wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("id_navEdit_0")));
		topItem.click();
	}
```

Screenshot



```
Mar 16, 2025 10:18:48 PM cucumber.api.cli.Main run
WARNING: You are using deprecated Main class. Please use io.cucumber.core.cli.Main

Scenario: check the stock view                       # src/test/resources/Features/PharmacyViewStock.feature:3
  Given browser is open                              # StepDefinitions.LoginSteps.browser_is_open()
  And user is on login page                          # StepDefinitions.LoginSteps.user_is_on_login_page()
  When user enters username and password as pharmacy # StepDefinitions.PharmacyStockManagementSteps.user_enters_username_and_password_as_pharmacy()
  And pharmacy on the stock page                     # StepDefinitions.PharmacyStockManagementSteps.pharmacy_on_the_stock_page()
  When pharmacy clicks on view stock button          # StepDefinitions.PharmacyStockManagementSteps.pharmacy_clicks_on_view_stock_button()

1 Scenarios (1 passed)
5 Steps (5 passed)
0m15.338s
```

Test report

Test Case 5

Project Name: Health Companion	
Pharmacy View Stock	
Test Case ID: Test_5	**Test Designed By: Jince Abraham**
Test Priority(Low/Medium/High): Low	**Test Designed Date: 03/03/2025**
Module Name: Pharmacy View Stock of Medicines	**Test Executed By : Mr. Ajith G S**
Test Title : Pharmacy View Stock	**Test Execution Date: 05/03/2025**
Description: Check stock view functionality of pharmacy	Test Executed Successfully

Pre-Condition : User has valid username and password

Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/ Fail)
1	Naviagate to the login page	NA	Login page loads successfully	Login page loads successfully	pass
2	Fill in username and password	“kottaram” for username and “Jjjjj@11” for password	Username and password fields are successfully filled in	Username and password fields are successfully filled in	pass
3	Click on the login Button	NA	Navigates to the doctor dashboard	Navigates to the pharmacy dashboard	pass
4	Clicks on the Stock Button	NA	Navigates to the stock page	Stock page loads successfully	pass
5	Click on the details of the medicine	NA	Navigates to the page that views the details of the medicine	Medicine Details viewed successfully	pass

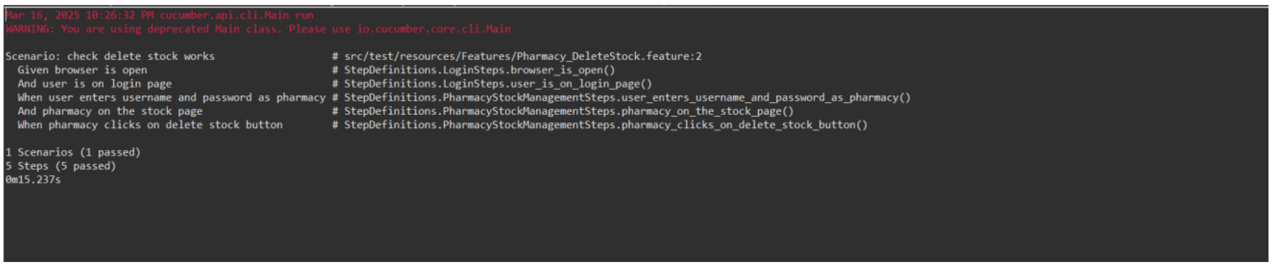
Post-Condition: Successfully viewed the stock

Test Case 6:

Code

```
        @When("pharmacy clicks on Edit stock button")
        public void pharmacy_clicks_on_Edit_stock_button() {
                WebDriverWait wait = new WebDriverWait(driver,
Duration.ofMinutes(1));
                WebElement topItem=
wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("id_navEdit_0")));
                topItem.click();
        }
```

Screenshot



```
Mar 16, 2025 10:26:32 PM cucumber.api.cli.Main run
WARNING: You are using deprecated Main class. Please use io.cucumber.core.cli.Main

Scenario: check delete stock works                 # src/test/resources/Features/Pharmacy_DeleteStock.feature:2
  Given browser is open                            # StepDefinitions.LoginSteps.browser_is_open()
  And user is on login page                        # StepDefinitions.LoginSteps.user_is_on_login_page()
  When user enters username and password as pharmacy # StepDefinitions.PharmacyStockManagementSteps.user_enters_username_and_password_as_pharmacy()
  And pharmacy on the stock page                   # StepDefinitions.PharmacyStockManagementSteps.pharmacy_on_the_stock_page()
  When pharmacy clicks on delete stock button      # StepDefinitions.PharmacyStockManagementSteps.pharmacy_clicks_on_delete_stock_button()

1 Scenarios (1 passed)
5 Steps (5 passed)
0m15.237s
```

Test report

Test Case 6

Project Name: Health Companion					
Pharmacy View Stock					
Test Case ID: Test_6			**Test Designed By: Jince Abraham**		
Test Priority(Low/Medium/High): Low			**Test Designed Date: 03/03/2025**		
Module Name: Pharmacy Delete Stock			**Test Executed By : Mr.Ajith G S**		
Test Title : Pharmacy Delete Stock			**Test Execution Date: 05/03/2025**		
Description: Check Delete stock functionality of pharmacy			Test Executed Successfully		
Pre-Condition : User has valid username and password					
Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status(Pass/ Fail)**
1	Naviagate to the login page	NA	Login page loads successfully	Login page loads successfully	pass
2	Fill in username and password	“kottaram” for username and “Jjjjj@11” for password	Username and password fields are successfully filled in	Username and password fields are successfully filled in	pass
3	Click on the login Button	NA	Navigates to the doctor dashboard	Navigates to the pharmacy dashboard	pass
4	Clicks on the Stock Button	NA	Navigates to the stock page	Stock page loads successfully	pass
5	Click on the delete button of the medicine	NA	Deletes the medicine from the stock	Medicine successfully deleted from the stock	pass
Post-Condition: Successfully Deleted the stock					

CHAPTER 6
IMPLEMENTATION

6.1 INTRODUCTION

The HealthCompanion Implementation entails production of an operative software application from a previously designed system. Following an approach that projects deployment of core functions of the system like user registration and appointment booking and consultation management of doctors, this phase of Implementation is the vital one in making everything come alive by means of code, and testing against all requirements.

The purpose of implementation is to get the system running in a live setting with all the users (patients, doctors, and admins) interacting efficiently with the system. Moreover, implementation includes user training, system maintenance, and hosting the system on a web platform for wider access.

6.2 IMPLEMENTATION PROCEDURES

These Implementation Procedures provide details on how the HealthCompanion system was developed and deployed. This includes writing code for the various components, including the front-end, back-end, working with the database, testing, and deployment on a hosting platform. The following activities were undertaken :

- Front-end development using React.js, providing the user interface for patients and doctors.
- Back-end development using Node.js and Express for user requests, authentication, and database interactions with MongoDB.
- Connecting to the database for storing and retrieving user profiles, appointment details, and doctor schedules.

After development, the system is tested for functionality, performance, and security before it is made accessible to users.

6.2.1 User Training

User Training establishes user proficiency by familiarizing patients doctors and administrative personnel with the system features along with the necessary understanding of its operational functions. The HealthCompanion system training includes separate lessons for each role type :

- HealthCompanion users learn to establish accounts and perform doctor searches as well as make bookings and assess previous consultation materials.
- Doctors are trained on how to manage their schedules, view patient appointments, and update consultation notes.
- The system allows admins to perform user management and doctor verification duties and access system statistical data.

Through user training programs we aim to deliver detailed instructions and practice opportunities which guide people toward understanding the system capabilities and operational routes.

6.2.2 Training on the Application Software

It is necessary that training on application software is intensively conducted for the users. The types of learning include the following:

- **Training for Patients**: The patient learns to navigate the patient interface and find a doctor, schedule appointments, and learn to read updates. Video tutorials and user guidebook training are included.
- **Training for Doctors**: Doctors are trained on how to manage schedules, work with patient data, and prescribe medication, which includes the accessing and usage of the doctor's dashboard.
- **Admin Training** Admins undergo training on system oversight, user management, and managing system configurations .

  Training is done through video tutorials, user manuals, and live training events.

6.2.3 System Maintenance

System Maintenance: It guarantees that HealthCompanion function well after its deployment. It consists of:

- **Bug Fixes**: In fixing the bugs related to the users' reports or checks up on routine activities by the system.
- **System Updates**: Introduce new features, security patches, and performance enhancements as necessary.
- **Data Backup**: All user information (like patient profiles, appointments) is safe and recoverable when the system is broken.
- **User Support**: Offers continuous technical support to resolve any user issues with entering or using the system.

System maintenance should be a continues process that ensures that the same system meets users' needs.

6.2.4 Hosting

Hosting is the process through which the HealthCompanion system is uploaded to a server so that it becomes available to users on the internet. Hosting ensures system access via web browsers from any device for patients and doctors.

- Ease of Deployment: Render offers a streamlined deployment process that enables direct connections to GitHub repositories, thereby simplifying the setup and application update process.
- Scalability: With Render, there are scalable hosting options that allow the HealthCompanion platform to grow with the additional increase in traffic and resource requirements
- Full-Stack Support: Render supports frontend and backend deployments, thus allowing for effective implementation of the HealthCompanion MERN stack architecture.

Procedure for hosting a website on Render:

Step 1: Connect the Repository

- Users should access the Render website while already logged in to their account.
- You should create a new web service on Render by connecting it to the GitHub repository which contains your HealthCompanion project.
- The main deployment branch must be selected followed by entry of essential build commands that correspond to your application's requirements.

Step 2: Configure Environment Variables

- The service settings require you to enter necessary environment variables such as VITE_BACKEND_URL for frontend settings together with database connection strings for backend operations.
- The runtime configuration of security-reliable application settings occurs through automatic variable insertion by Render.

Step 3: Deploy the Application

- The deployment system of Render automatically identifies Node.js as the build environment to start the deployment process.
- The deployment logs should be monitored for errors while checking that all installed packages with their dependencies function as intended.

Step 5: Testing and Monitoring

- Check functionality by accessing the application URL once it is deployed to make sure all pages run with their features and API endpoints work.
- The monitoring tools at Render will allow tracking application performance with the option

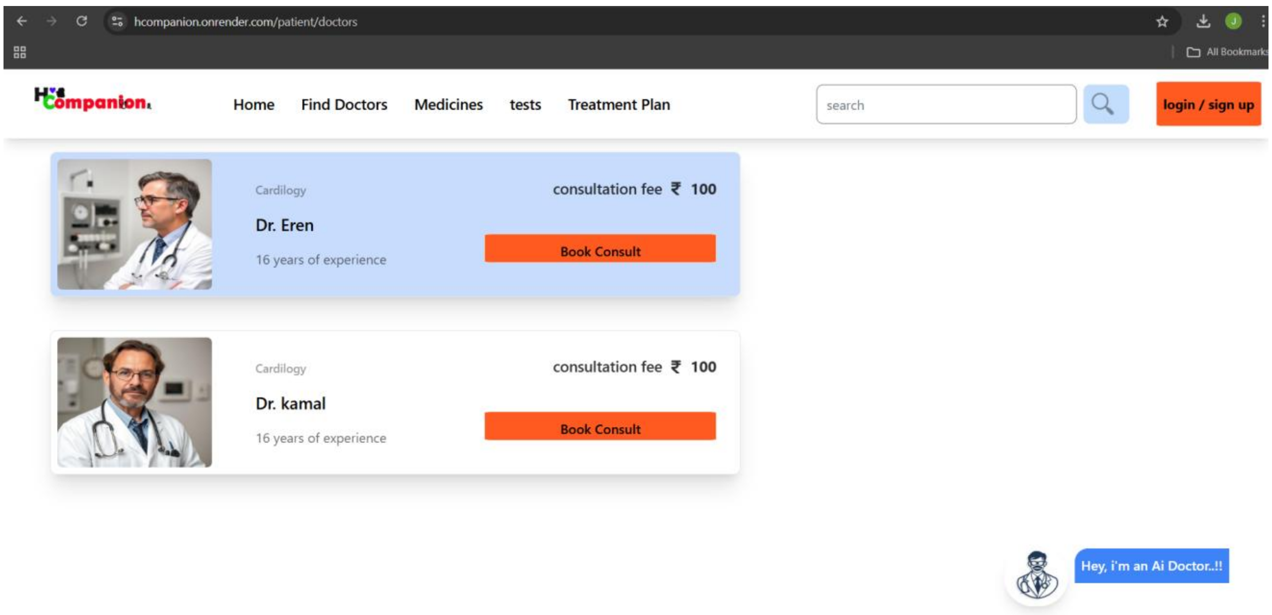
to automatically redeploy changes from the GitHub repository.

Hosted Link: https://hcompanion.onrender.com/

Hosted QR:

HEALTH COMPANION

Screenshot



CHAPTER 7

CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

The HealthCompanion project efficiently addresses the need to build integrated digital health solutions that establish enhanced healthcare accessibility as well as operational success. Through the system patients access a seamless procedure because they can confirm appointments while checking medical histories without encountering any interruptions in their provider communication. The platform delivers security along with convenience to users because of its appointment booking features and verification tools paired with authenticated procedures. The project exemplifies that healthcare digital transformation leads to better patient-provider collaboration that enhances medical care quality while cutting administrative procedures. HealthCompanion has shown robust dependability and adjustment potential during extensive testing stages that will support subsequent advancement of the platform.

7.2 FUTURE SCOPE

The potential growth of the HealthCompanion project will result from different system enhancements alongside expanded applications. The HealthCompanion project presents two opportunities for future growth: AI diagnosis addition and patient-specific health recommendation functions which will enhance patient health management capabilities. Patient data analysis by means of machine learning algorithms enables doctors to gain important health prediction insights and valuable patient trend data. HealthCompanion should expand its functionalities by adding remote medical capabilities along with secure document exchange methods to help more patients. The integration of language support capabilities alongside wearable integration capabilities would modernize HealthCompanion to meet present-day healthcare needs better. The organization needs to examine these emerging domains to turn HealthCompanion into a total innovative healthcare system.

CHAPTER 8

BIBLIOGRAPHY

REFERENCES:

- Bass, L., Clements, P., & Kazman, R. (2012). Software Architecture in Practice. Addison-Wesley Professional.

• Fowler, M. (2002). Patterns of Enterprise Application Architecture. Addison-Wesley.

• Pressman, R. S. (2014). Software Engineering: A Practitioner's Approach (8th ed.). McGraw-Hill.

• Rescorla, E. (2001). SSL and TLS: Designing and Building Secure Systems. Addison-Wesley.

• MongoDB Documentation (2023).Used for managing data storage in the IntelliStay Management System with MongoDB. Retrieved from: https://docs.mongodb.com/

• ReactJS Documentation (2023).Reference for the frontend development of IntelliStay using ReactJS. Retrieved from: https://reactjs.org/docs/getting-started.html

• Node.js Documentation (2023).Guide for server-side development using Node.js. Retrieved from: https://nodejs.org/en/docs/

WEBSITES:

• World Health Organization (2023). Digital Health Overview. Retrieved from https://www.who.int/health-topics/digital-health

• HealthIT.gov (2022). What is Telehealth? Retrieved from https://www.healthit.gov

• CDC (Centers for Disease Control and Prevention) (2023). Public Health Data Modernization Initiative. Retrieved from https://www.cdc.gov

• Mayo Clinic (2023). Telemedicine and Virtual Care Options. Retrieved from https://www.mayoclinic.org

• National Institutes of Health (NIH) (2023). Technology and Health Innovation. Retrieved from https://www.nih.gov

CHAPTER 9

APPENDIX

9.1 Sample Code

login.jsx

```
import React, { useEffect, useState } from "react";
import { useFormik } from "formik";
import { loginValidation } from "../validations/userValidation";
// import { jwtDecode } from "jwt-decode";
import { accountCreationGoogle, checkForgotPasswordOtp, forgotPasswordOtp, loginUser,
loginUsingGoogle, resetPassword } from "../services/loginService";
import { useSelector, useDispatch } from 'react-redux'
import { setUser } from "../redux/slices/userSlice";
import Swal from 'sweetalert2'
import { useNavigate } from "react-router-dom";
import Modal from "react-modal"
import { resetPasswordSchema } from "../validations/forgotPasswordValidations";
import styles from "./login.module.css";
import { maskedEmail } from "../utils/email";
import { FloatingLabel } from "flowbite-react";
import NavBar from "../patient/components/navBar/NavBar";
Modal.setAppElement('#root')

function Login() {

  const dispatch=useDispatch()
  const {username,role}=useSelector((state)=>state.user)
  const [isForgotPasswordChecked,setisForgotPasswordChecked]=useState(false)
  const [forgotEmail,setForgotEmail]=useState("jjjjjjjj")
  const navigate=useNavigate()
  const formik = useFormik({
    initialValues: {
      username: "",
      password: "",
    },
    validationSchema: loginValidation,
    onSubmit: async (values, actions) => {
      const userData=await loginUser(values)
      if(userData){
        await Swal.fire({
          title:"Successfull",
          icon:"success",
          text:"you have been successfully logged in"
        })
        await
dispatch(setUser({username:userData.username,role:userData.role,email:userData.email,isLogged
In:true}))
        if(userData.role=="2")
          return navigate('/pharmacy/home')
        else if(userData.role=="1")
          return navigate('/doctor')
        else if(userData.role=="3")
          return navigate('/laboratory')
        else
           navigate('/')
      }

    },
  });


  useEffect(()=>{

    const clientId = import.meta.env.VITE_GOOGLE_CLIENT_ID;
    google.accounts.id.initialize({
      client_id: clientId,
      callback: async (response)=>{
        const userData=await loginUsingGoogle({googleToken:response.credential})
        if(!userData){
          await selectRoleGoogleSign(response)
        }
        else{
          await Swal.fire({
            title:"Successfull",
            icon:"success",
            text:"you have been successfully logged in"
          })
          dispatch(setUser({username:userData.username,role:userData.role,email:userData.email}))
          if(userData.role=="2")
            return navigate('/pharmacy/home')
          else if(userData.role=="3")
            return navigate('/laboratory/home')
          navigate('/')

        }

      },
    });



    google.accounts.id.renderButton(document.getElementById("id_google"), {
      theme: "filled_blue",
      size: "large",
      width: 350,
      type: "standard",
    });

  },[])


  const handleForgotPassword=async ()=>{
    try {

     const {value:email} =await Swal.fire({
       title:"forgot password",
       input:"email",
       showCancelButton:true,
       inputLabel:"Enter email",
      })
      console.log(email)
      await setForgotEmail(email)
      const sendOtp=await forgotPasswordOtp({email})
      if(email && sendOtp){
        const {value:otp} =await Swal.fire({
          title:"Enter Otp",
          input:"number",
          showCancelButton:true,
          inputLabel:"Enter Otp",
        })
        if(otp){
          const otpCheck=await checkForgotPasswordOtp({email,otp})
          if(otpCheck){
            await setisForgotPasswordChecked(true)
          }
        }

      }

    } catch (error) {
      console.log(`error at handle forgot password ${error}`)

    }
  }

  return (

    <div className='homePage w-full h-full'>
        <div className='header w-full h-[12vh] shadow-lg bg-white'>
            <NavBar/>
        </div>
        <div className="flex flex-row w-screen h-screen">
      {(isForgotPasswordChecked) && <ForgotPasswordModal email={forgotEmail}
isOtpChecked={isForgotPasswordChecked}
setisForgotPasswordChecked={setisForgotPasswordChecked}/>}
      <div className="w-[40%] ">
        <img src="logo.png" alt="" />
      </div>
      <div className="w-[60%] mainContainer h-[90vh] mt-1  flex flex-col justify-start items-
center text-emerald-300 gap-10 bg-sky-300 rounded-l-[5%]">
        <form
          onSubmit={formik.handleSubmit}
          className="border w-[50%] rounded-[5%] p-20 flex flex-col gap-4 bg-white  border-
```

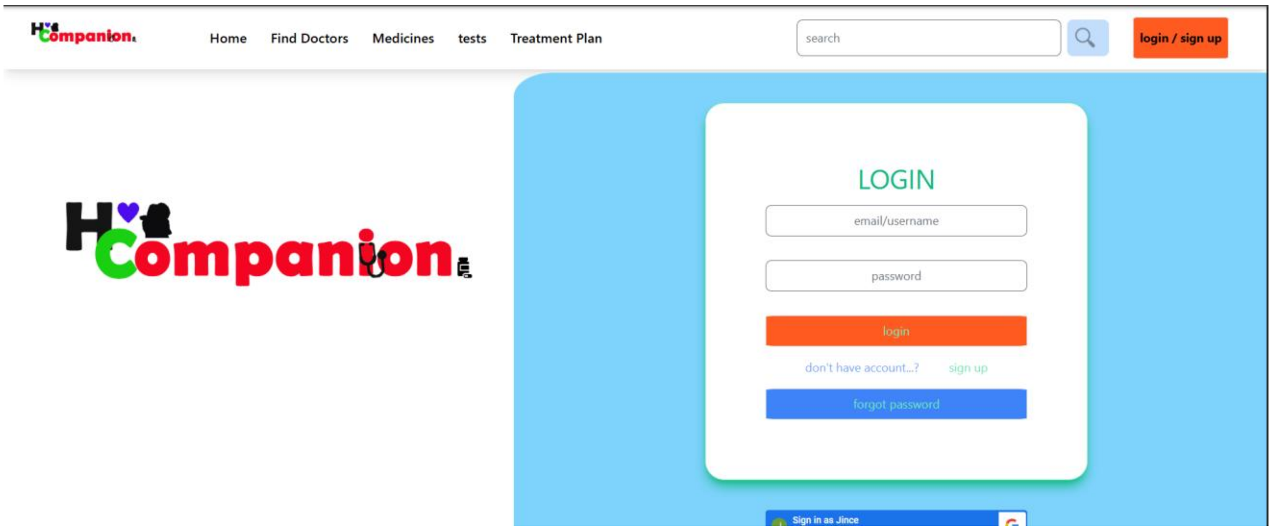
```
emerald-300 shadow-lg shadow-emerald-500 mt-10"
        >
          <div className="flex justify-center">
            <h1 className="text-4xl text-bold text-emerald-500 ">LOGIN</h1>
          </div>
          <div className="flex flex-col gap-4 text-black">
            <input
              type="text"
              name="username"
              id="id_username"
              onChange={formik.handleChange}
              onBlur={formik.handleBlur}
              className="border p-2 rounded-lg text-center"
              placeholder="email/username"
            />
            <p className="text-red-500">
              {formik.touched.username &&
                formik.errors.username &&
                formik.errors.username}
            </p>
            <input
              type="password"
              name="password"
              id="id_password"
              onChange={formik.handleChange}
              onBlur={formik.handleBlur}
              className="border p-2 rounded-lg text-center"
              placeholder="password"
            />

            <p className="text-red-500">
              {formik.touched.password &&
                formik.errors.password &&
                formik.errors.password}
            </p>
          </div>

          <div className="flex flex-col gap-4">
            <button
              id="id_loginButton"
              className="w-[full] h-[50%] border rounded-[5%] bg-orange-500 p-2"
              type="submit"
            >
              login
            </button>
            <div className="flex flex-row gap-10 h-[50%] justify-center">
              <h2 className="text-blue-400">don't have account...?</h2>
              <button type="button" onClick={()=>{
                navigate('/register')
              }
              }>sign up</button>
          </div>
          <button type="button" onClick={handleForgotPassword} className="w-[full] h-[50%]
border rounded-[5%] bg-blue-500 p-2">
           forgot password
          </button>
         </div>
        </form>
        <div className="">
         <button id="id_google"></button>
        </div>
       </div>

      </div>
      </div>

   );
  }

  const selectRoleGoogleSign=async (response)=>{
   const {value:role}=await Swal.fire({
    input:"select",
    showCancelButton:true,
    inputPlaceholder:"select role to proceed",
    title:"Role",
    inputOptions:{
     0:"patient",
     1:"doctor",
     2:"pharmacy",
     3:"laboratory"
    }
   })
   if(role){
    const userData=await accountCreationGoogle({googleToken:response.credential,role})
    if(userData){
     Swal.fire({
      title:"Account Creation Successfull",
      icon:"success",
      text:"Account has been created"
     })
    }
   }


  }

  function ForgotPasswordModal(props){

   const[isOpen,setIsOpen]=useState(false)

   const closeModal=async ()=>setIsOpen(false)
```

```
    const formik=useFormik({
      initialValues:{
          password:"",
          confirmPassword:""
      },
      validationSchema:resetPasswordSchema,
      onSubmit:async (values,actions)=>{
        const resetPasswordStatus=await
resetPassword({password:values.password,email:props.email})
        if(resetPasswordStatus){
          Swal.fire({
            icon:"success",
            html:`<b>Password has been reset</b>`
          }).then(async ()=>{
            await closeModal()
            await props.setisForgotPasswordChecked(false)
          })
        }

      }

    })

    useState(()=>{

      if(props.isOtpChecked)
        setIsOpen(true)

      console.log("hiiiii")

    },[])

   return(
    <div className="fixed w-screen h-screen">
      <Modal
            isOpen={isOpen}
            onRequestClose={closeModal}
            overlayClassName="fixed bg-black bg-opacity-50 inset-0 flex justify-center items-center"
            className="bg-white opacity-100 mx-w-[60vw] flex flex-col gap-10 p-10 rounded-lg
justify-center items-center">

              <h1 className="text-center font-bold">Verification</h1>
              <h3>Otp has been generated and sent to the email {maskedEmail(props.email)}</h3>

              <form onSubmit={formik.handleSubmit} className="w-full flex flex-col gap-10
justify-center items-center">
                <div className="flex flex-col gap-10 w-full">

                    <FloatingLabel name="password" type="password" variant="filled" label="Enter the
password" onChange={formik.handleChange} onBlur={formik.handleBlur} className="w-
[40vw]"
```

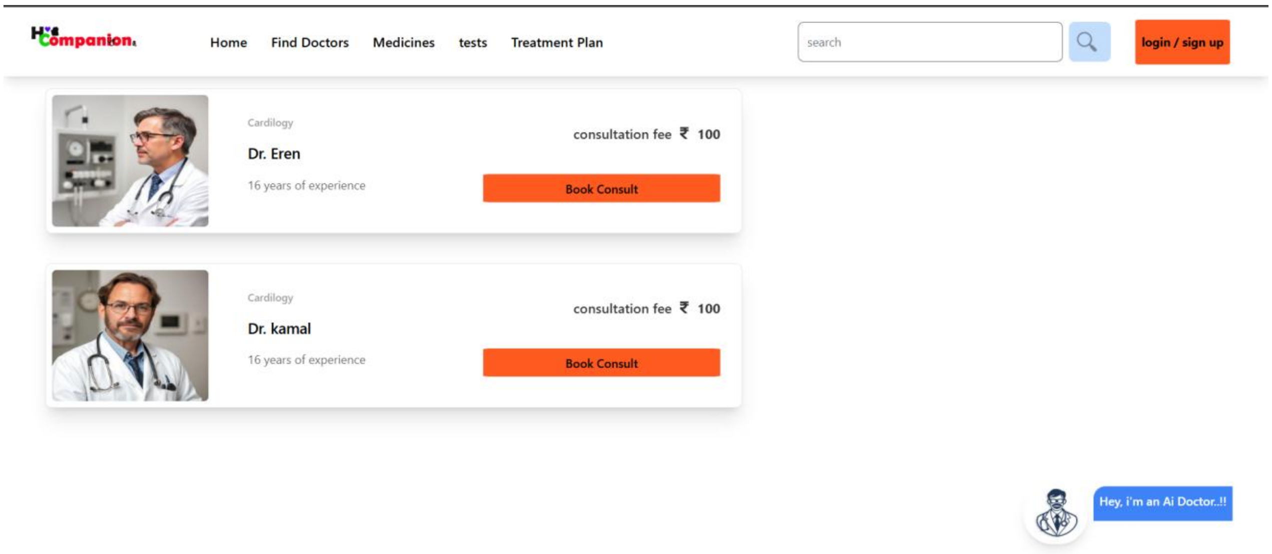
```
              {...(formik.touched.password &&
((formik.errors.password)?{color:"error",helperText:`${formik.errors.password}`}:{color:"succes
s"}))}
              />
              <FloatingLabel name="confirmPassword" type="password"  variant="filled"
label="Enter the confirm password" onChange={formik.handleChange}
onBlur={formik.handleBlur} className="w-[40vw]"
              {...(formik.touched.confirmPassword &&
((formik.errors.confirmPassword)?{color:"error",helperText:`${formik.errors.confirmPassword}`
}:{color:"success"}))}
              />
            </div>
            <button type="submit" className="w-full border rounded-[5%] bg-orange-500 p-
2">Submit</button>

          </form>

      </Modal>

    </div>
  )
}


export default Login;
```

9.2 SCREEN SHOTS

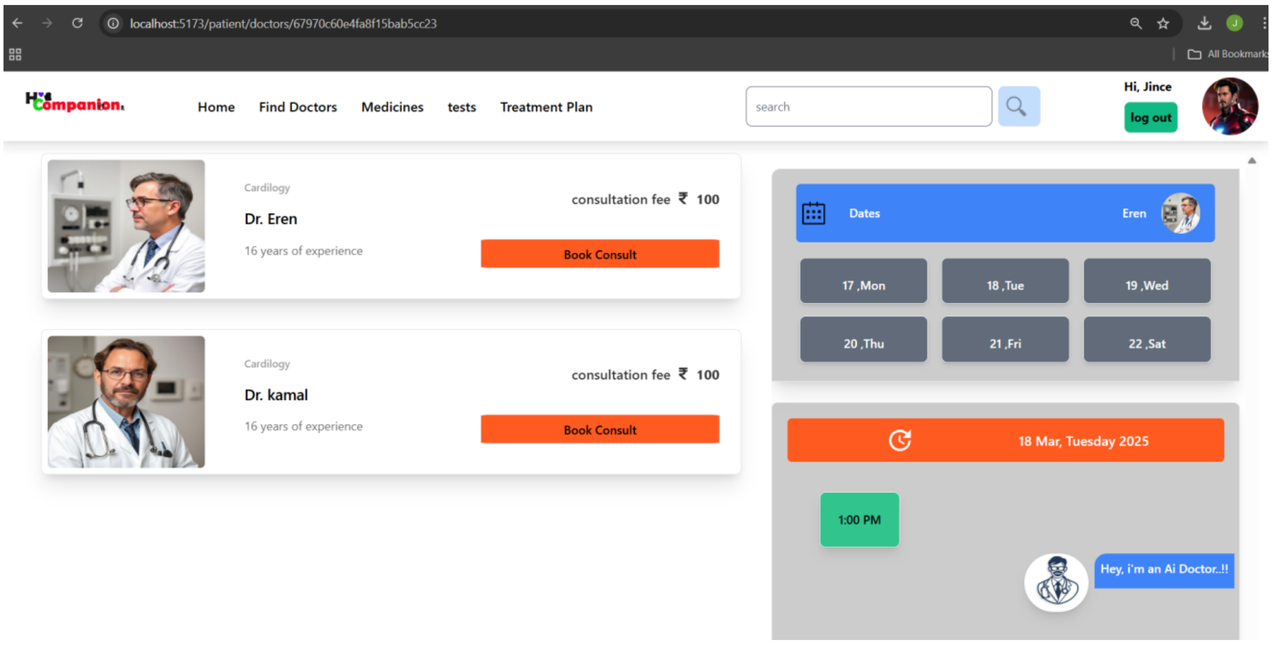
9.2.1 LOGIN PAGE



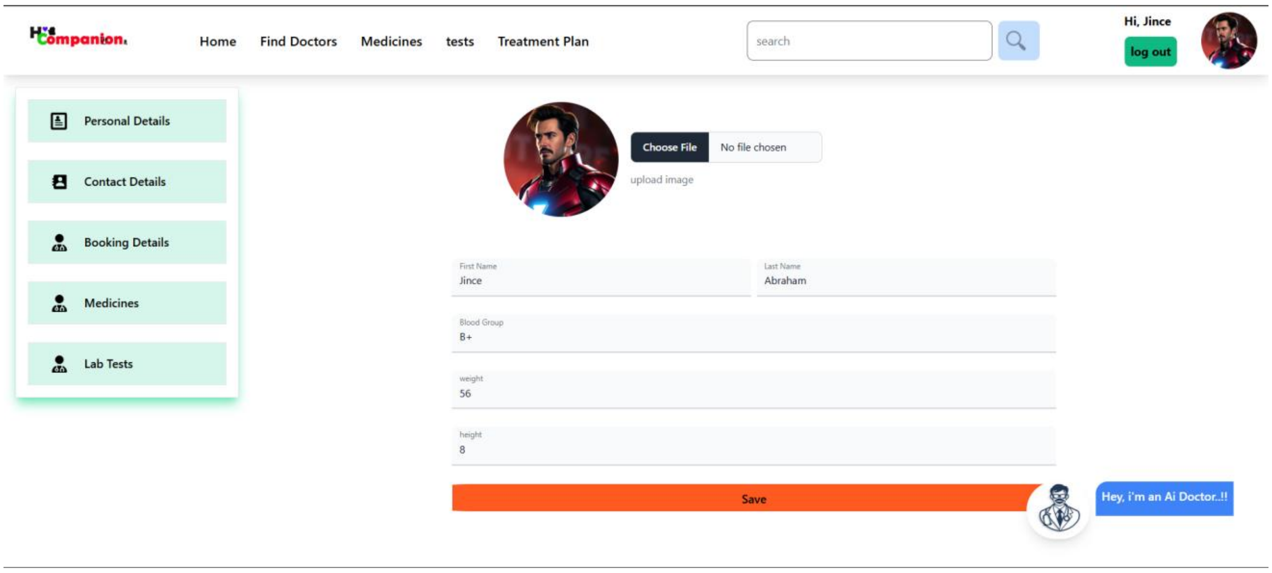
9.2.3 DASHBOARD



9.2.4 BOOKING PAGE

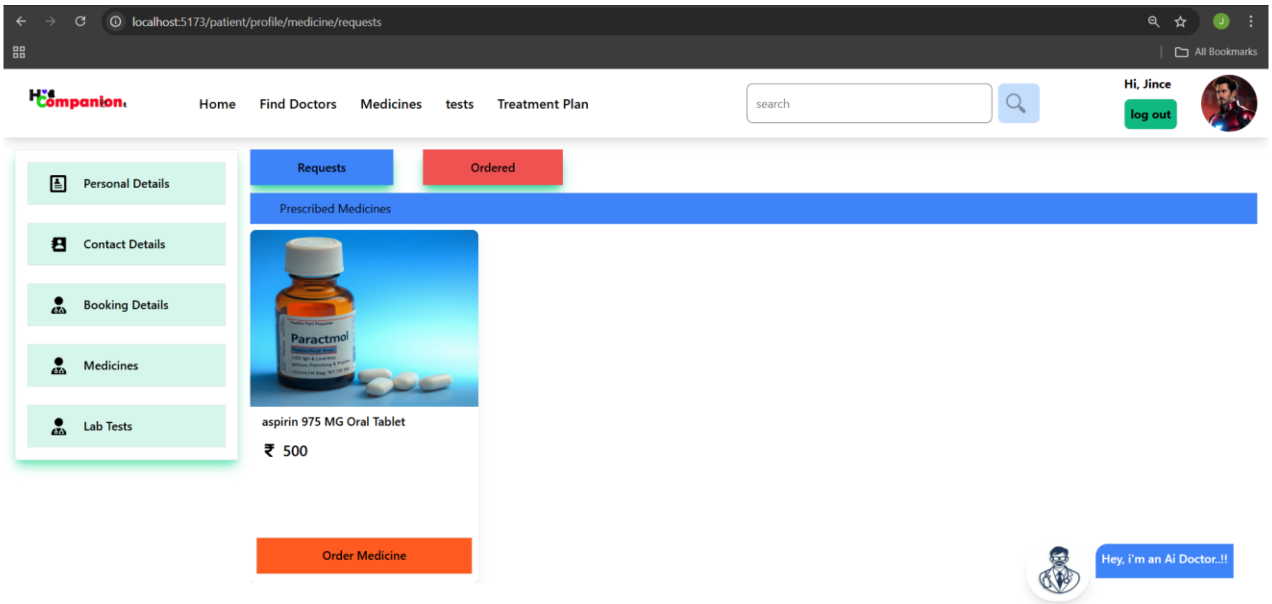


9.2.5 PROFILE PAGE

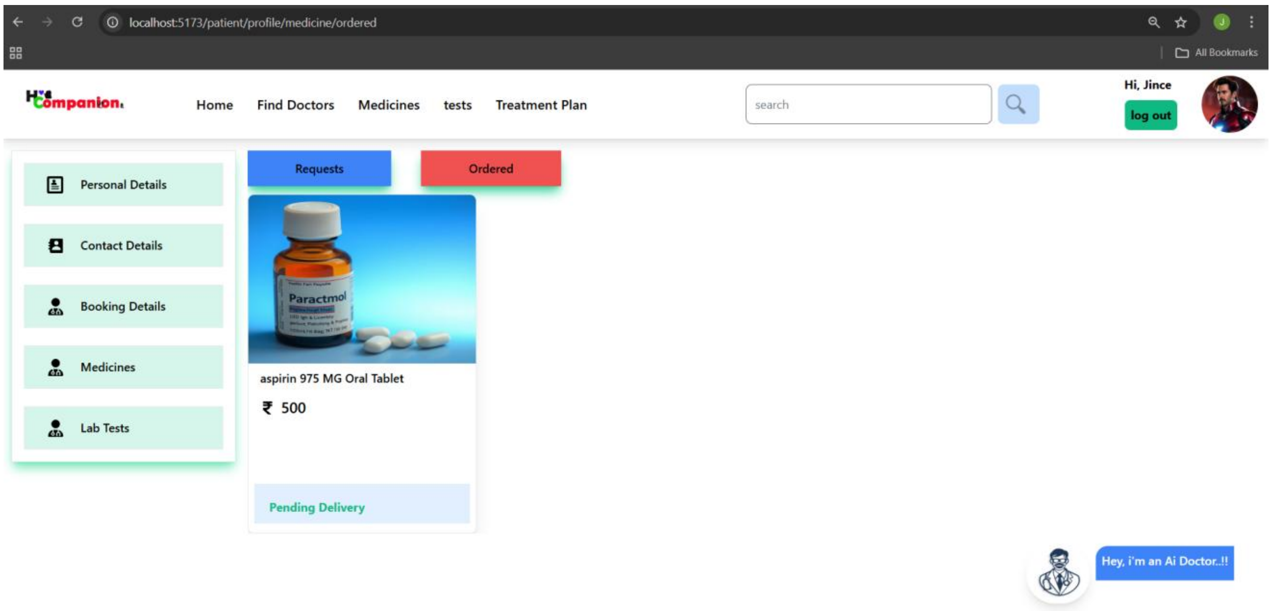


First Name: Jince
Last Name: Abraham
Blood Group: B+
weight: 56
height: 8

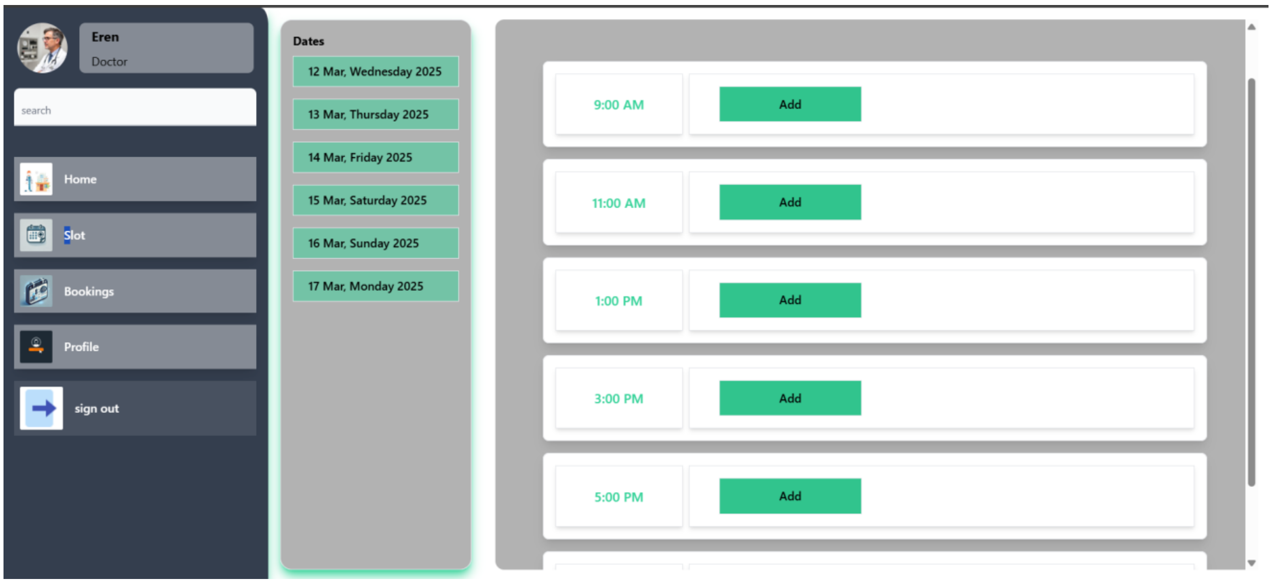
9.2.6 PATIENT REQUESTED MEDICINES FROM DOCTOR



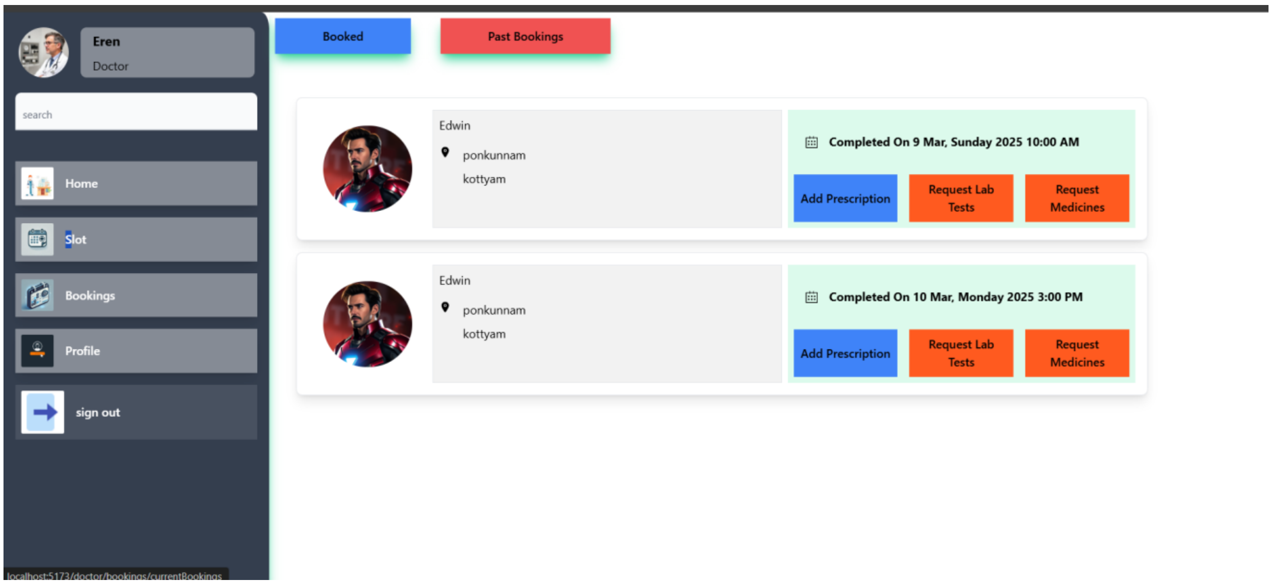
9.2.7 PATIENT ORDERED REQUESTED MEDICINES FROM DOCTOR



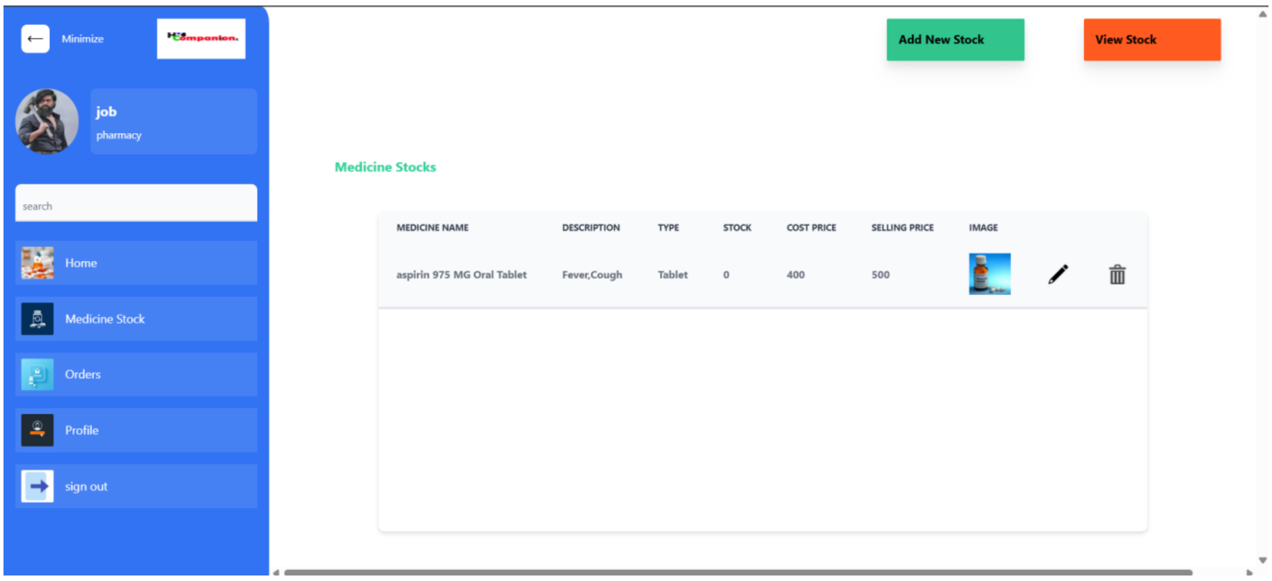
9.2.8 DOCTOR DASHBOARD



9.2.9 DOCTOR BOOKED PATIENTS PAGE



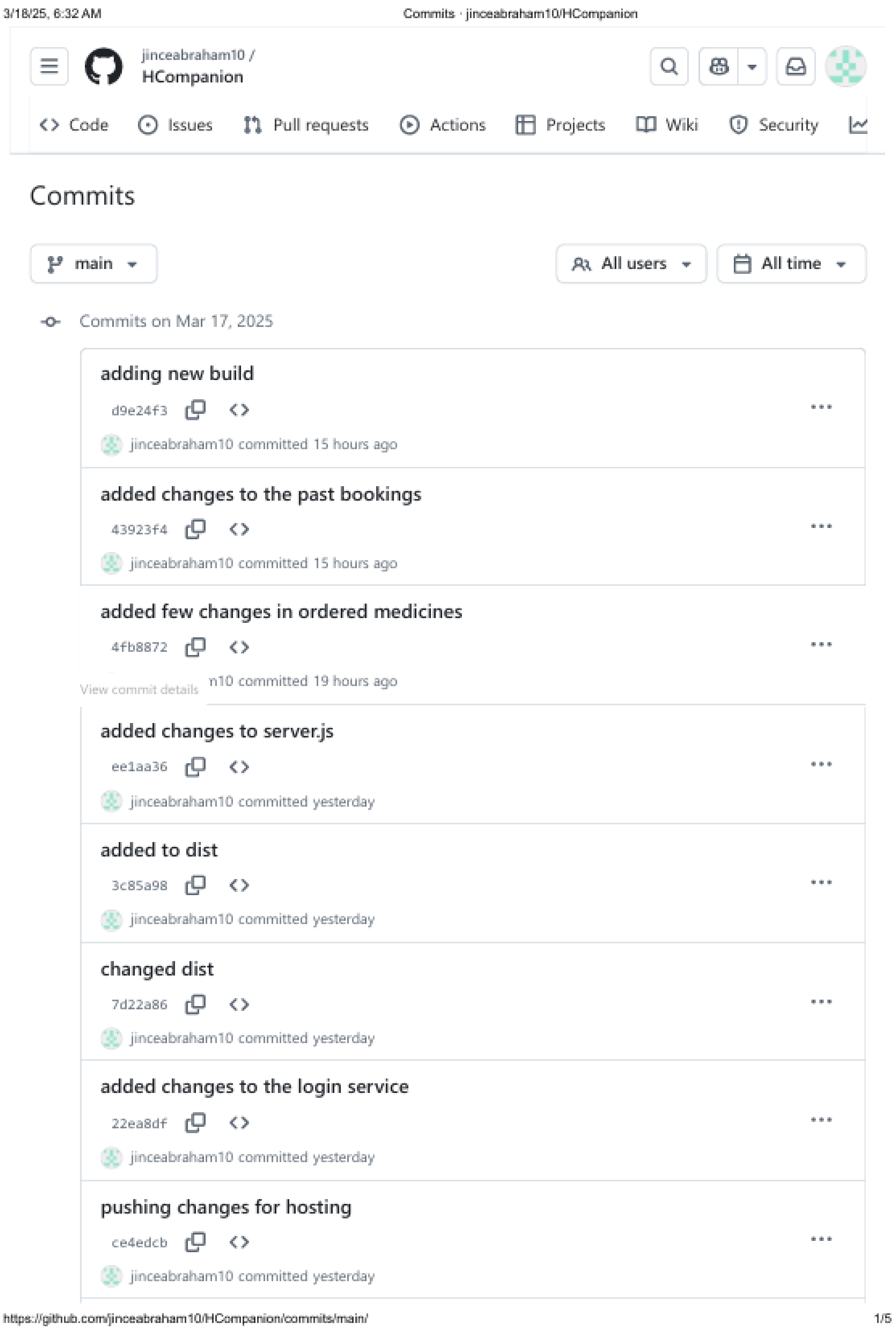
9.2.10 PHARMACY MEDICINE LIST PAGE



Medicine Stocks

MEDICINE NAME	DESCRIPTION	TYPE	STOCK	COST PRICE	SELLING PRICE	IMAGE		
aspirin 975 MG Oral Tablet	Fever,Cough	Tablet	0	400	500			

9.3 GIT LOG



3/18/25, 6:32 AM Commits · jinceabraham10/HCompanion

Commits

Commits on Mar 17, 2025

adding new build
d9e24f3
jinceabraham10 committed 15 hours ago

added changes to the past bookings
43923f4
jinceabraham10 committed 15 hours ago

added few changes in ordered medicines
4fb8872
View commit details n10 committed 19 hours ago

added changes to server.js
ee1aa36
jinceabraham10 committed yesterday

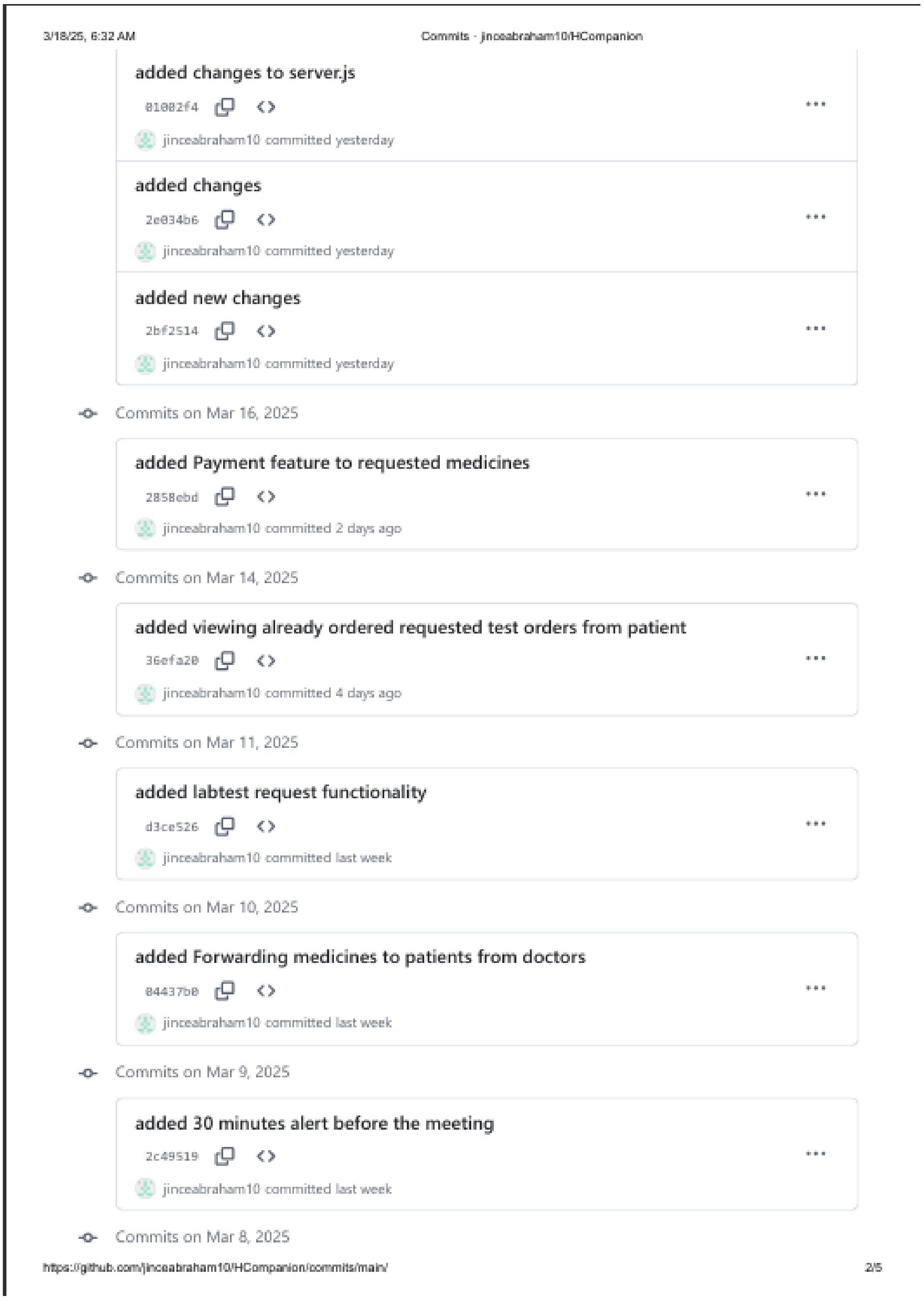
added to dist
3c85a98
jinceabraham10 committed yesterday

changed dist
7d22a86
jinceabraham10 committed yesterday

added changes to the login service
22ea8df
jinceabraham10 committed yesterday

pushing changes for hosting
ce4edcb
jinceabraham10 committed yesterday

https://github.com/jinceabraham10/HCompanion/commits/main/ 1/5


3/18/25, 6:32 AM Commits · jinceabraham10/HCompanion

added changes to server.js
81082f4
jinceabraham10 committed yesterday

added changes
2e034b6
jinceabraham10 committed yesterday

added new changes
2bf2514
jinceabraham10 committed yesterday

Commits on Mar 16, 2025

added Payment feature to requested medicines
2858ebd
jinceabraham10 committed 2 days ago

Commits on Mar 14, 2025

added viewing already ordered requested test orders from patient
36efa20
jinceabraham10 committed 4 days ago

Commits on Mar 11, 2025

added labtest request functionality
d3ce526
jinceabraham10 committed last week

Commits on Mar 10, 2025

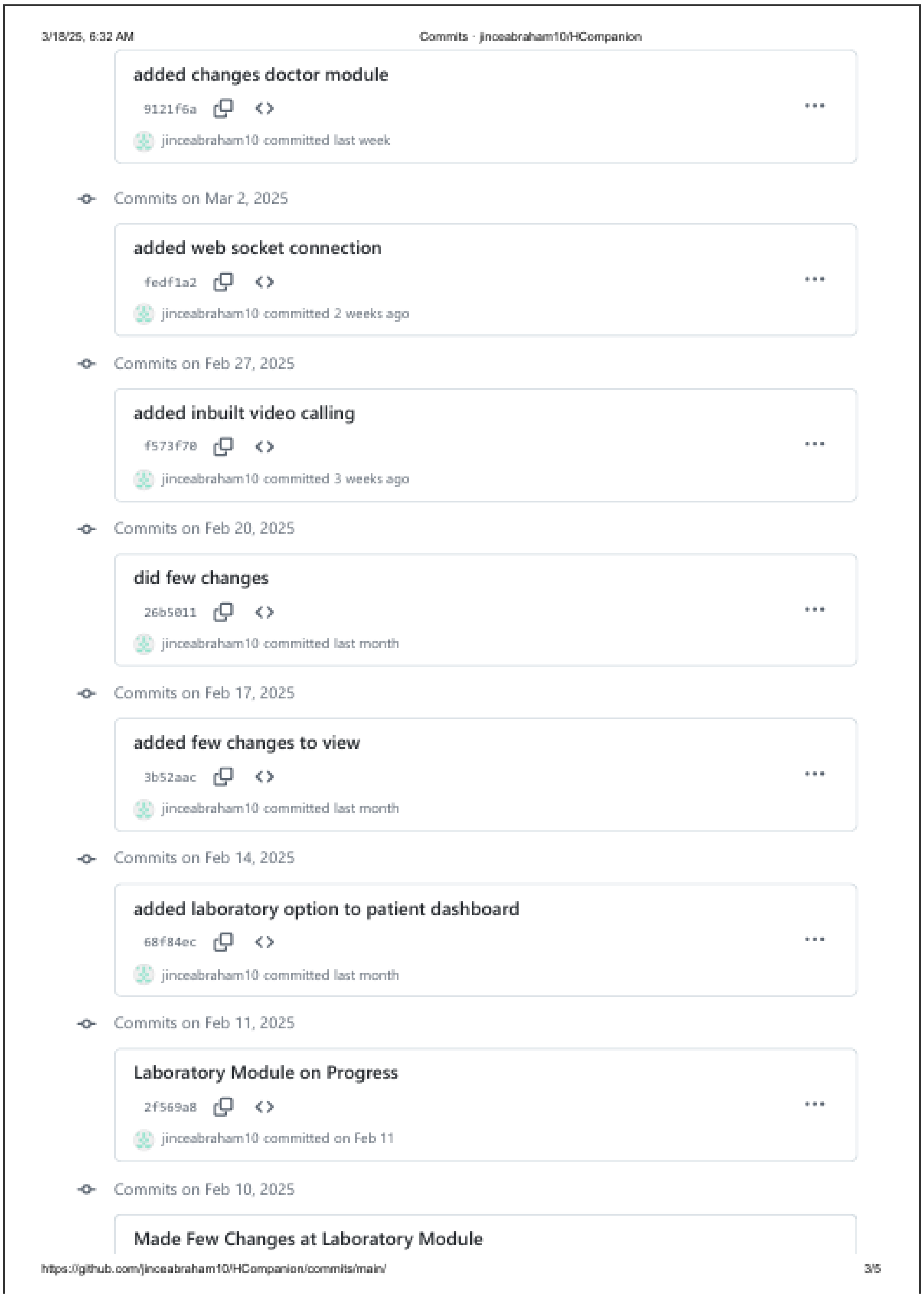
added Forwarding medicines to patients from doctors
84437b0
jinceabraham10 committed last week

Commits on Mar 9, 2025

added 30 minutes alert before the meeting
2c49519
jinceabraham10 committed last week

Commits on Mar 8, 2025

https://github.com/jinceabraham10/HCompanion/commits/main/ 2/5



3/18/25, 6:32 AM Commits · jinceabraham10/HCompanion

added changes doctor module
9121f6a
jinceabraham10 committed last week

Commits on Mar 2, 2025

added web socket connection
fedf1a2
jinceabraham10 committed 2 weeks ago

Commits on Feb 27, 2025

added inbuilt video calling
f573f70
jinceabraham10 committed 3 weeks ago

Commits on Feb 20, 2025

did few changes
26b5011
jinceabraham10 committed last month

Commits on Feb 17, 2025

added few changes to view
3b52aac
jinceabraham10 committed last month

Commits on Feb 14, 2025

added laboratory option to patient dashboard
68f84ec
jinceabraham10 committed last month

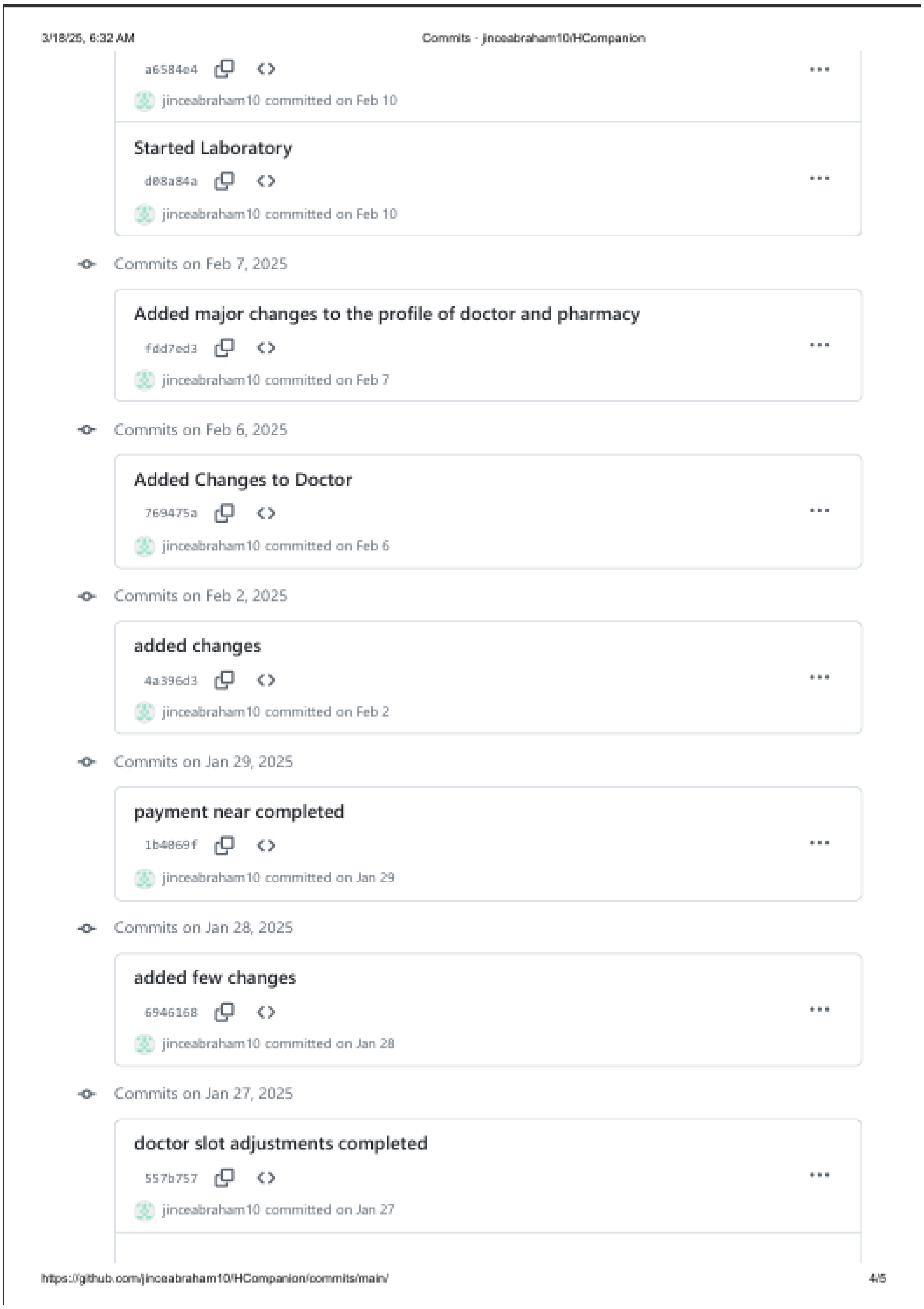
Commits on Feb 11, 2025

Laboratory Module on Progress
2f569a8
jinceabraham10 committed on Feb 11

Commits on Feb 10, 2025

Made Few Changes at Laboratory Module

https://github.com/jinceabraham10/HCompanion/commits/main/ 3/5


3/18/25, 6:32 AM Commits · jinceabraham10/HCompanion

a6584e4
jinceabraham10 committed on Feb 10

Started Laboratory
d08a84a
jinceabraham10 committed on Feb 10

Commits on Feb 7, 2025

Added major changes to the profile of doctor and pharmacy
fdd7ed3
jinceabraham10 committed on Feb 7

Commits on Feb 6, 2025

Added Changes to Doctor
769475a
jinceabraham10 committed on Feb 6

Commits on Feb 2, 2025

added changes
4a396d3
jinceabraham10 committed on Feb 2

Commits on Jan 29, 2025

payment near completed
1b4869f
jinceabraham10 committed on Jan 29

Commits on Jan 28, 2025

added few changes
6946168
jinceabraham10 committed on Jan 28

Commits on Jan 27, 2025

doctor slot adjustments completed
557b757
jinceabraham10 committed on Jan 27

https://github.com/jinceabraham10/HCompanion/commits/main/ 4/5

3/18/25, 6:32 AM Commits · jinceabraham10/HCompanion

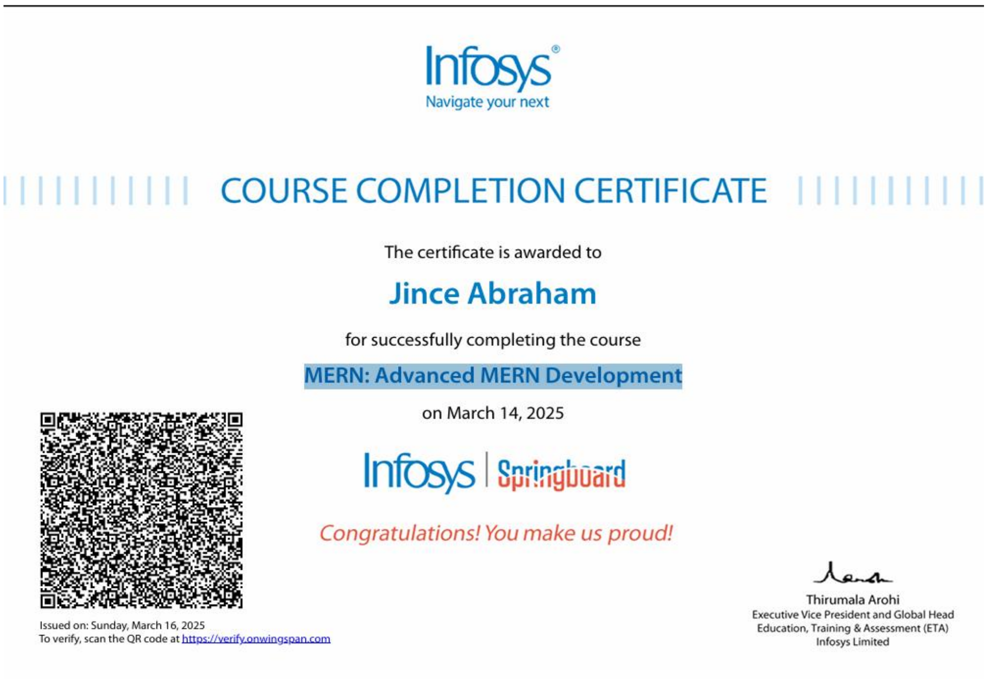
first commit

3fc6179

jinceabraham10 committed on Jan 27

9.4 CERTIFICATES

9.4.1 MERN: ADVANCED MERN DEVELOPMENT



Infosys®
Navigate your next

COURSE COMPLETION CERTIFICATE

The certificate is awarded to

Jince Abraham

for successfully completing the course

MERN: Advanced MERN Development

on March 14, 2025

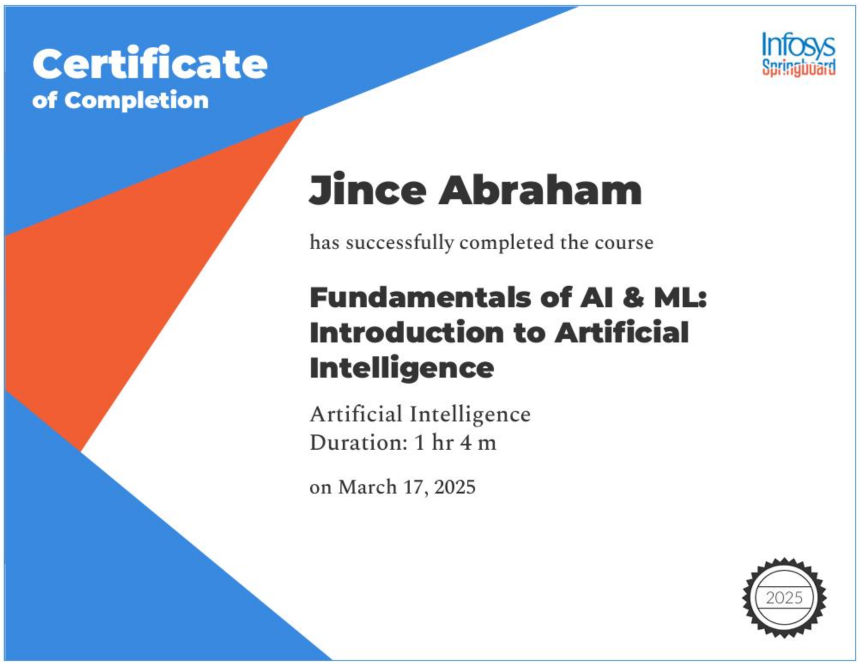
Infosys | Springboard

Congratulations! You make us proud!

Issued on: Sunday, March 16, 2025
To verify, scan the QR code at https://verify.onwingspan.com

Thirumala Arohi
Executive Vice President and Global Head
Education, Training & Assessment (ETA)
Infosys Limited

9.4.2 FUNDAMENTALS OF AI & ML: INTRODUCTION TO ARTIFICIAL INTELLIGENCE



Certificate
of Completion

Infosys
Springboard

Jince Abraham

has successfully completed the course

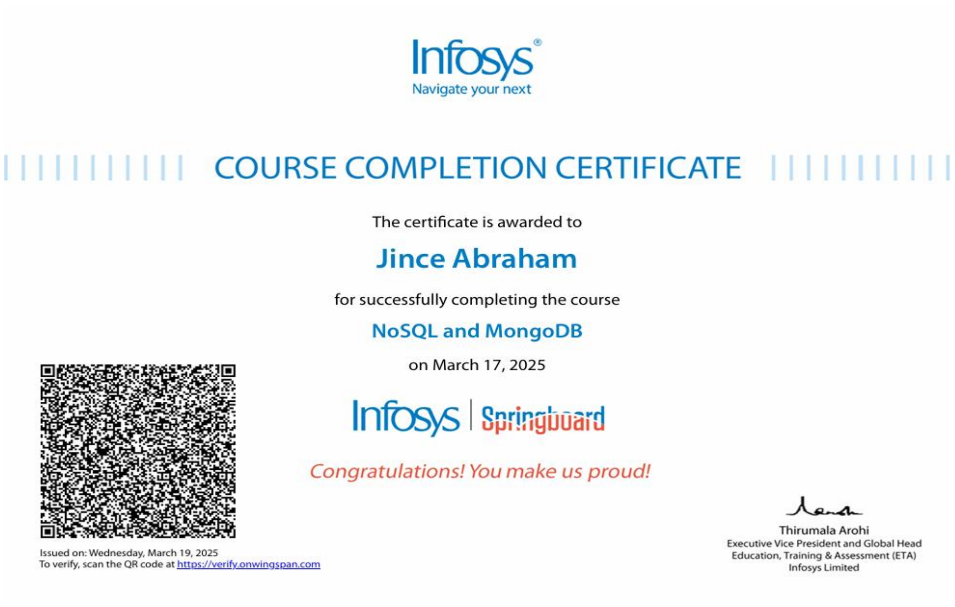
Fundamentals of AI & ML: Introduction to Artificial Intelligence

Artificial Intelligence
Duration: 1 hr 4 m

on March 17, 2025

2025

9.4.3 NOSQL AND MONGODB



Infosys
Navigate your next

COURSE COMPLETION CERTIFICATE

The certificate is awarded to

Jince Abraham

for successfully completing the course

NoSQL and MongoDB

on March 17, 2025

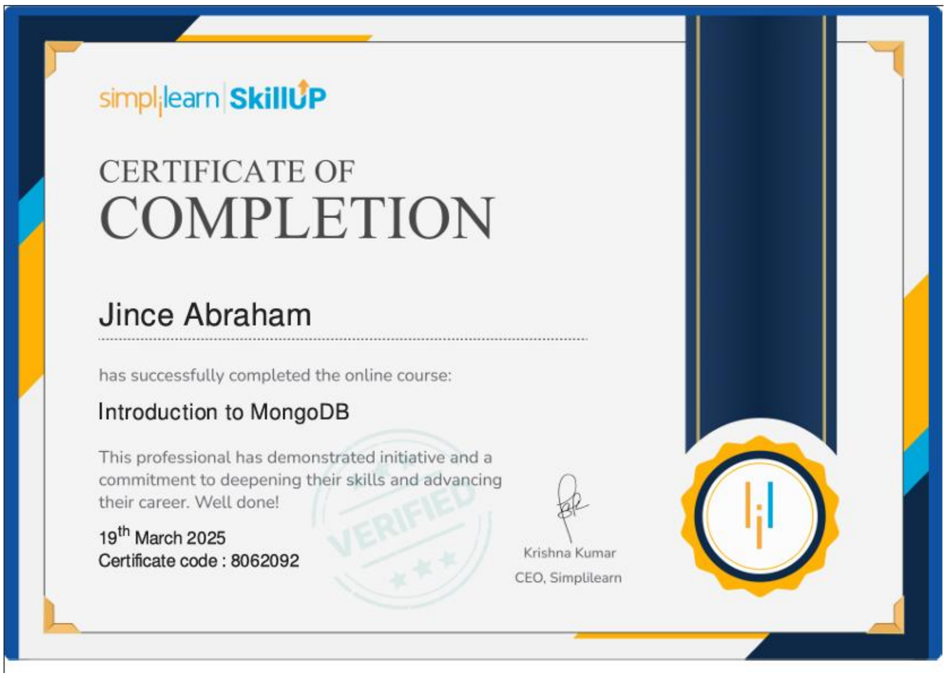
Infosys | Springboard

Congratulations! You make us proud!

Issued on: Wednesday, March 19, 2025
To verify, scan the QR code at https://verify.onwingspan.com

Thirumala Arohi
Executive Vice President and Global Head
Education, Training & Assessment (ETA)
Infosys Limited

9.4.4 INTRODUCTION TO MONGODB



simplilearn SkillUP

CERTIFICATE OF
COMPLETION

Jince Abraham

has successfully completed the online course:

Introduction to MongoDB

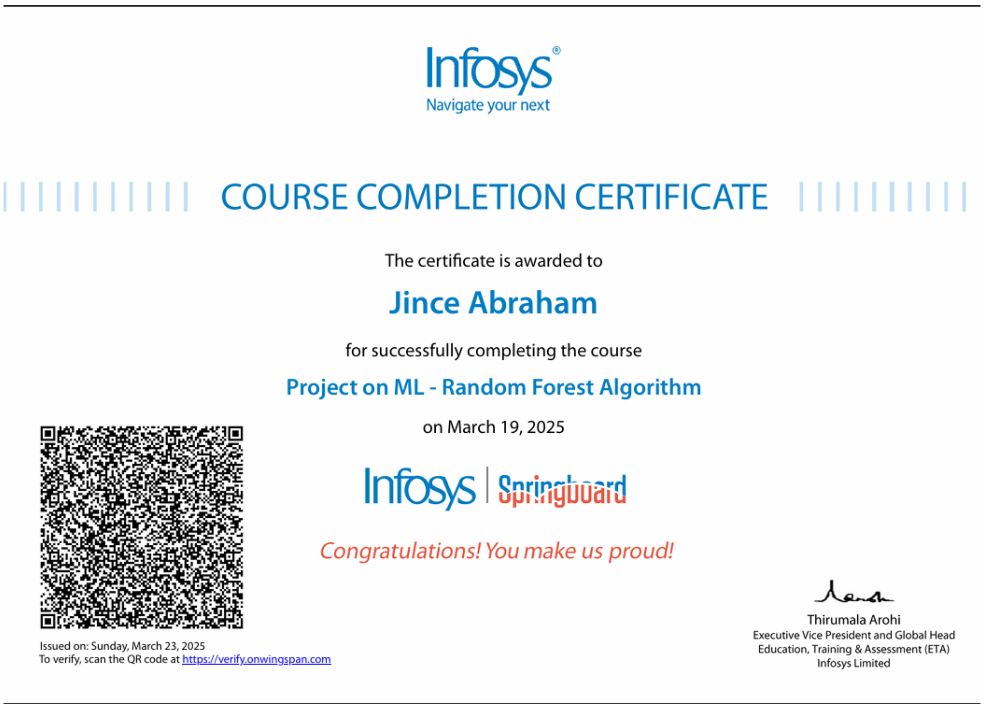
This professional has demonstrated initiative and a commitment to deepening their skills and advancing their career. Well done!

19th March 2025
Certificate code : 8062092

VERIFIED

Krishna Kumar
CEO, Simplilearn

9.4.5 RANDOM FOREST ALGORITHM



Infosys®
Navigate your next

COURSE COMPLETION CERTIFICATE

The certificate is awarded to

Jince Abraham

for successfully completing the course

Project on ML - Random Forest Algorithm

on March 19, 2025

Infosys | Springboard

Congratulations! You make us proud!

Issued on: Sunday, March 23, 2025
To verify, scan the QR code at https://verify.onwingspan.com

[illegible]
Thirumala Arohi
Executive Vice President and Global Head
Education, Training & Assessment (ETA)
Infosys Limited