

VISVESVARAYA TECHNOLOGICAL UNIVERSITY
“JNANA SANGAMA”, BELAGAVI - 590 018



A PROJECT REPORT
on
“Revolutionizing Global Healthcare with AI Powered
Kiosk”

submitted by

Anjali P Nambiar	4SF21IS009
Ashish Goswami	4SF21IS015
K Rakshitha	4SF21IS037
Varun A L	4SF21IS121

In partial fulfillment of the requirements for the award of

BACHELOR OF ENGINEERING

in

INFORMATION SCIENCE & ENGINEERING

under the Guidance of

Ms. Amritha Mohan C K

Assistant Professor, Department of ISE

at



SAHYADRI

College of Engineering & Management

An Autonomous Institution

Adyar, Mangaluru - 575 007

AY: 2024 - 25

SAHYADRI
College of Engineering & Management
An Autonomous Institution
Adyar, Mangaluru - 575 007

Department of Information Science & Engineering



CERTIFICATE

This is to certify that the project entitled “**Revolutionizing Global Healthcare with AI Powered Kiosk**” has been carried out by **Anjali P Nambiar (4SF21IS009), Ashish Goswami (4SF21IS015), K Rakshitha (4SF21IS037) and Varun A L (4SF21IS121)**, the bonafide students of Sahyadri College of Engineering & Management in partial fulfillment of the requirements for the award of Bachelor of Engineering in Information Science & Engineering (ISE) of Visvesvaraya Technological University, Belagavi during the Academic Year 2024 - 25. It is certified that all the corrections/suggestions indicated for the Continuous Internal Assessment have been incorporated in the report deposited in the library of the ISE department. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the said degree.

Project Guide
Ms. Amritha Mohan C K
Assistant Professor
Dept. of ISE, SCEM

HoD
Dr. Rithesh Pakkala P.
Associate Professor
ISE & CSE(DS), SCEM

Principal
Dr. S S Injaganeri
Professor
SCEM, Mangaluru

External Viva-Voce

Examiner's Name

Signature with Date

1.
2.

.....
.....

SAHYADRI
College of Engineering & Management
An Autonomous Institution
Adyar, Mangaluru - 575 007

Department of Information Science & Engineering



DECLARATION

We hereby declare that the entire work embodied in this Project Report titled “**Revolutionizing Global Healthcare with AI Powered Kiosk**” has been carried out by us at Sahyadri College of Engineering & Management, Mangaluru, under the supervision of Ms. **Amritha Mohan C K**, for the award of **Bachelor of Engineering in Information Science & Engineering**. This report has not been submitted to this or any other University for the award of any other degree.

Anjali P Nambiar (4SF21IS009)

Ashish Goswami (4SF21IS015)

K Rakshitha (4SF21IS037)

Varun A L (4SF21IS121)

Dept. of ISE, SCEM, Mangaluru

Abstract

The healthcare sector often faces challenges such as prolonged patient wait times, inefficient communication between patients and doctors, and difficulties in managing appointments and lab tests. To address these issues, our project, Revolutionizing Global Healthcare with AI-Powered Kiosk, introduces an innovative solution aimed at streamlining hospital workflows and enhancing patient satisfaction. The kiosk enables patients to register or log in using OTP functionality and access two primary features: a chatbot for symptom analysis and a lab test management module. The AI-powered chatbot collects symptoms from patients and, using a predefined dataset, maps them to the most relevant specialist if three or more symptoms match. Patients are required to pay the consultation fee securely through Razorpay. Upon successful payment, the system displays the specialist's name and room number on the screen, sends an SMS with these details to the patient, and notifies the specialist via the CareLink app, a dedicated platform for doctors to manage their profiles and daily appointments. For lab test management, patients can view prescribed tests, make payments through the kiosk, and receive detailed pre-test guidelines. Lab technicians are granted access to the CareLink app to upload test results, which doctors can later retrieve seamlessly. This system significantly reduces wait times, improves operational efficiency, and ensures effective communication between patients and healthcare providers.

Acknowledgement

It is with great satisfaction and euphoria that we are submitting the project report on **“Revolutionizing Global Healthcare with AI Powered Kiosk”**. We have completed it as a part of the curriculum of Visvesvaraya Technological University, Belagavi for the award of Bachelor of Engineering in Information Science & Engineering.

We are profoundly indebted to our guide, **Ms. Amritha Mohan C K**, Assistant Professor, Department of Information Science & Engineering, for innumerable acts of timely advice and encouragement, and we sincerely express our gratitude.

We also thank **Mr. Vasudeva Rao P V**, Assistant Professor & Project Coordinator, Department of Information Science & Engineering, for the constant encouragement and support extended throughout.

We express our sincere gratitude to **Dr. Rithesh Pakkala P.**, Associate Professor & Head, Department of Information Science & Engineering and Computer Science & Engineering (Data Science) for his invaluable support and guidance.

We sincerely thank **Dr. Sudheer Shetty**, Professor, ISE & Vice Principal, Sahyadri College of Engineering & Management, who has constantly motivated us throughout the project work.

We sincerely thank **Dr. S S Injaganeri**, Principal, Sahyadri College of Engineering & Management, who have always been a great source of inspiration.

Finally, yet importantly, We express our heartfelt thanks to the nonteaching staffs of ISE department and our family & friends for their wishes and encouragement throughout the work.

Anjali P Nambiar (4SF21IS009)

Ashish Goswami (4SF21IS015)

K Rakshitha (4SF21IS037)

Varun A L (4SF21IS121)

Table of Contents

Abstract	i
Acknowledgement	ii
Table of Contents	iv
List of Figures	v
List of Tables	vi
1 Introduction	1
1.1 Purpose	1
1.2 Scope	1
1.3 Overview	3
2 Literature Survey	4
3 Problem Definition and Objectives	9
3.1 Problem Statement	9
3.2 Objectives	9
4 Software Requirements Specification	11
4.1 Introduction	11
4.2 Purpose	11
4.3 User Characteristics	12
4.4 Interfaces	12
4.4.1 Hardware Interfaces	12
4.4.2 Software Interfaces	13
5 System Design	14
5.1 Architecture Design	14
5.2 Class Diagram	17

5.3	State Diagram	19
5.4	Use Case Diagram	22
5.5	Sequence Diagram	25
5.6	Activity Diagram	27
5.7	Data Flow Diagram	29
5.8	Modular Diagram	32
6	Implementation	34
6.1	Algorithm	34
6.2	Flow Chart	39
6.3	Implementation Codes	40
6.3.1	Code for Doctor suggestion based on Symptoms	40
6.3.2	Code for Twillo Intergration	41
6.3.3	Doctor Dashboard in Carelink	42
6.3.4	Lab Technician Dashboard in Carelink	47
7	System Testing	51
7.1	Unit Testing with results	51
7.2	Module Testing with results	54
7.3	System Testing with results	56
8	Results and Discussions	58
8.1	Results	58
8.2	Discussions	65
8.2.1	Timeline of the Project Work	65
8.2.2	Outcomes Obtained	66
8.2.3	Objectives Achieved	66
8.2.4	Challenges Encountered	67
9	Conclusion and Future Work	69
9.1	Future Work	69
	References	71

List of Figures

5.1	System Architecture Diagram	14
5.2	Class Diagram	17
5.3	State Diagram	19
5.4	Use Case Diagram	22
5.5	Sequence Diagram	25
5.6	Activity Diagram	27
5.7	Data Flow Diagram	29
5.8	Modular Diagram	32
6.1	Flow Chart	39
8.1	Login Page	58
8.2	Register Page	58
8.3	Welcome Screen with Chatbot and Labtest option	59
8.4	Chatbot Interaction	59
8.5	SMS Notification.	60
8.6	Payment Screen	60
8.7	Lab Test Payment	61
8.8	Lab Test Selection	61
8.9	CareLink App Interfaces	62
8.10	Doctor's Registration Page	62
8.11	Doctor's Login Page	63
8.12	Doctor's Profile	63
8.13	Doctor Dashboard	64
8.14	Lab Technician Dashboard	64

List of Tables

7.1	Test Cases for Unit Testing	51
7.2	Test Cases for Module Testing	54
7.3	Test Cases for System Testing	56
8.1	Project Work Plan	65
8.2	Outcomes Mapped to Objectives	66

Chapter 1

Introduction

1.1 Purpose

The purpose is to present an innovative AI-powered solution, as a comprehensive system designed to revolutionize healthcare delivery and operational efficiency. By integrating advanced technologies such as self-service kiosks, AI-driven symptom assessment, secure payment gateways, and real-time notification systems, CareLink addresses critical challenges in hospital workflows and patient care. The system focuses on: Enhancing Accessibility by providing patients with an intuitive platform for specialist recommendations, appointment scheduling, and lab test management; Improving Efficiency by automating key hospital processes such as symptom analysis, consultation scheduling, and diagnostic report handling; Facilitating Seamless Communication between patients, doctors, and lab technicians through synchronized applications and real-time notifications; and Empowering Patients and Providers by enabling patients to take charge of their healthcare journey while equipping doctors with tools to manage schedules and access vital patient data effortlessly.

1.2 Scope

The scope encompasses a comprehensive, AI-driven system designed to optimize healthcare delivery and accessibility through the integration of advanced technology and patient-centric solutions. Key aspects of its scope include:

- **Patient Services:**

- Enable patients to securely log in through OTP for accessing healthcare services.
- Provide a symptom-assessing chatbot for gathering patient data and recommending appropriate specialists.
- Facilitate lab test booking, pre-test instructions, and secure payment options via Razorpay.
- Ensure seamless access to test results and consultation details through real-time SMS notifications.

- **Doctor Workflow Management:**

- Offer hospital doctors access to a dedicated CareLink app to manage daily appointments and patient profiles.
- Provide visibility into lab test results and patient records, allowing informed decision-making.
- Enable streamlined scheduling and notifications for improved workflow efficiency.

- **Operational Integration:**

- Automate processes like specialist recommendations, appointment confirmations, and test result uploads.
- Enhance hospital operational efficiency by reducing manual interventions and delays.
- Establish a synchronized platform connecting patients, doctors, and lab personnel.

- **Technological Advancements:**

- Leverage AI for intelligent symptom mapping and recommendation generation.
- Incorporate secure payment systems to handle consultation and lab test fees.
- Utilize self-service kiosks to enhance accessibility in hospitals and healthcare facilities.

1.3 Overview

The project is an AI-powered healthcare system designed to streamline hospital operations and improve patient care. It features a self-service kiosk for secure patient login, symptom assessment via an AI chatbot, and specialist recommendations. Patients can pay consultation fees securely through Razorpay, and real-time notifications are sent to both patients and doctors.

The system also handles lab test management, allowing patients to book tests, make payments, and access pre-test guidelines. Lab results are uploaded by technicians into the CareLink app, providing doctors with instant access to patient data.

CareLink automates processes, enhances communication, and improves hospital efficiency, offering a seamless experience for both patients and healthcare providers.

Chapter 2

Literature Survey

The literature survey examines technological solutions for improving healthcare access in underserved areas, including telemedicine, mobile health clinics, and digital health tools. It highlights challenges such as infrastructure, funding, and technology access, aiming to identify opportunities for enhancing healthcare delivery.

The paper “Healthcare Kiosk: Next Generation Accessible Healthcare Solution” by Verma et al. [1] discusses the potential of healthcare kiosks in providing accessible healthcare solutions, particularly in developing countries. The paper highlights the challenges of limited healthcare workforce and capital, proposing a fusion of technologies, including vital signs monitoring and e-health-based diagnostic systems. These kiosks would allow patients to measure vital signs, receive on-the-spot diagnoses, and get recommended to specialty hospitals or receive preliminary solutions. The business model proposed incorporates both service and advertising revenue, where minimal costs are extracted from customers for sensing devices, while hospitals or websites recommended by the kiosk can pay for display setups in the form of advertisement revenue. Their work envisions a future where this kiosk-based system becomes as pervasive as ATMs in providing healthcare services

The paper “The Role of Health Kiosks: Scoping Review” by Maramba et al. [2] provides an in-depth review of the role health kiosks play in modern healthcare. The authors explore various applications of health kiosks, including patient self-service, data collection, and health monitoring, highlighting their potential to improve healthcare accessibility and efficiency. They discuss the current state of health kiosks in healthcare settings, their benefits in reducing administrative burden, and challenges related to implementation and user adoption. The review concludes with future trends and recommendations for

enhancing the effectiveness of health kiosks.

The paper "An Overview of the Evolution and Impact of Chatbots in Modern Healthcare Services" by Tersoo Catherine et al. [3] explores the significant role of chatbots in healthcare, examining their evolution and how they have been integrated into modern healthcare services. The authors highlight the benefits of using chatbots for improving patient engagement, providing medical advice, and supporting decision-making. Their study reviews the impact of chatbots on healthcare accessibility and efficiency, addressing both their potential and limitations in real-world applications.

The paper "Supporting Primary Care Through Symptom Checking Artificial Intelligence" by Mahlknecht et al. [4] discusses patient and physician attitudes toward AI-powered symptom checkers in Italian general practice. The study shows that both patients and healthcare professionals see the potential of AI in improving efficiency and accessibility in primary care. It explores the integration of symptom checkers into clinical workflows and assesses user satisfaction and effectiveness.

The paper "Self-Diagnosis Through AI-enabled Chatbot-based Symptom Checkers" by You and Gui [5] examines user experiences and design considerations for AI-powered symptom checkers. The authors analyze how users interact with these systems and the factors influencing their design. Their findings provide insights into improving the usability and effectiveness of chatbot-based healthcare tools, aiming to enhance user satisfaction and diagnosis accuracy.

The paper "Doctor-Patient Communication in the E-Health Era" by Weiner [6] explores the evolving dynamics of doctor-patient communication in the digital age. It focuses on how e-health technologies, such as telemedicine and electronic health records, impact the way doctors and patients interact. The study highlights the challenges and opportunities presented by these digital tools in improving patient care and engagement.

The paper "AI Chatbot Design During an Epidemic Like the Novel Coronavirus" by Battineni et al. [7] examines the role of AI chatbots during the COVID-19 pandemic. The authors discuss how chatbots were deployed to provide accurate health information, assess symptoms, and reduce the burden on healthcare systems. The paper highlights the potential for AI chatbots to play a critical role in managing public health during global health crises.

The paper "A Smart Chatbot Architecture Based on NLP and Machine Learning for Health Care Assistance" by Ayanouz et al. [8] presents a chatbot architecture that integrates natural language processing (NLP) and machine learning techniques to assist in healthcare. This system provides patients with reliable health advice and personalized assistance, enhancing the efficiency of healthcare services and enabling easy access to medical information.

The paper "Understanding Medical Consumers' Intentions to Switch from Cash Payment to Medical Mobile Payment" by Hsieh [9] explores the behavior of medical consumers in switching from cash payments to mobile payment methods for healthcare services. The study uses a push-pull-mooring (PPM) framework to understand the factors influencing this shift and provides valuable insights for healthcare providers and technology developers.

The paper "Managing the Demand for Laboratory Testing: Options and Opportunities" by Janssens [10] discusses methods to manage laboratory test demand in healthcare settings. The paper reviews various strategies to optimize test ordering behaviors, reduce unnecessary tests, and improve resource utilization, with a focus on computerized systems and diagnostic-related reimbursement models.

The paper "Reimagining Healthcare: Unleashing the Power of Artificial Intelligence in Medicine" by Iqbal et al. [11] explores how artificial intelligence (AI) is fundamentally transforming the field of medicine. It investigates various AI technologies, such as machine learning, deep learning, and natural language processing, and their applications in improving diagnostic accuracy, treatment personalization, and patient management. Iqbal discusses how AI is being leveraged to analyze vast amounts of medical data, enhance clinical decision-making, and optimize healthcare workflows. The paper also addresses the challenges associated with AI integration, including data security, ethical considerations, and the need for regulatory frameworks. By presenting case studies and recent advancements, Iqbal highlights the potential of AI to revolutionize healthcare delivery, making it more efficient, effective, and tailored to individual patient needs.

The paper "A Low-Cost Community Healthcare Kiosk" by Sun et al. [12] presents a cost-effective healthcare kiosk designed for community healthcare services. The authors describe the development and functionality of the kiosk, focusing on its ability to provide basic health monitoring and consultation services. The study emphasizes the potential

of such kiosks in improving accessibility to healthcare for underserved populations by reducing costs and increasing efficiency.

The paper "Design of a Kiosk Type Healthcare Robot System for Older People in Private and Public Places" by Ahn et al. [13] investigates the design and implementation of a healthcare robot integrated into a kiosk system. The study focuses on its usability for older adults in both private and public environments. The authors explore how the system supports health monitoring, communication, and social interaction, aiming to enhance the independence and quality of life for elderly individuals.

The paper "The Use of a Self-Check-In Kiosk for Early Patient Identification and Queuing in the Emergency Department" by Coyle et al. [14] examines the impact of self-check-in kiosks on patient flow in emergency departments. The authors analyze how these kiosks facilitate early patient identification and queuing, thereby reducing waiting times and improving operational efficiency. Their findings suggest that self-check-in systems can significantly enhance the patient experience in high-pressure hospital environments.

The paper "Digitalizing Access to Care: How Self-Check-In Kiosks Shape Access to Care and Efficiency of Hospital Services" by Loukili et al. [15] explores the transformative role of self-check-in kiosks in healthcare settings. The authors investigate how these kiosks influence patient access to care and the efficiency of hospital workflows. Their findings highlight the dual benefits of improving service delivery and minimizing administrative burdens, while also addressing potential challenges such as digital exclusion.

The paper "A Paradigm Shift in Appointment Scheduling: Introducing a Decentralized Integrated Online Booking System" by Seyedi et al. [16] introduces a novel decentralized approach to appointment scheduling in healthcare. The authors propose an integrated online booking system that enhances patient autonomy and reduces scheduling conflicts. Their study discusses the system's potential to improve resource utilization and patient satisfaction while overcoming limitations of traditional appointment scheduling methods.

The paper "Appointment Scheduling in Health Care: Challenges and Opportunities" by Gupta et al. [17] reviews the complexities of appointment scheduling in healthcare environments. The authors highlight key challenges, including variability in patient arrivals and resource constraints, and propose strategies to optimize scheduling processes. Their work underscores the importance of data-driven approaches and advanced algorithms in

improving operational efficiency and patient outcomes.

The paper "A Mobile-Based Expert System for Disease Diagnosis and Medical Advice Provisioning" by Isinkaye et al. [18] discusses the development of a mobile expert system designed for disease diagnosis and medical advice. The authors highlight the system's ability to provide accurate and timely guidance to users, particularly in areas with limited access to healthcare professionals. Their study showcases the potential of mobile technologies in bridging gaps in healthcare accessibility.

The paper "Medical Applications for Pharmacists Using Mobile Devices" by Aungst et al. [19] explores the use of mobile applications to support pharmacists in their professional roles. The study discusses various apps designed for drug information, patient counseling, and medication management. The authors emphasize the growing importance of mobile technology in enhancing the efficiency and accuracy of pharmaceutical care.

The paper "Using Digital Notifications to Improve Attendance in Clinic: Systematic Review and Meta-Analysis" by Robotham et al. [20] reviews the effectiveness of digital notifications, such as SMS and email reminders, in improving clinic attendance rates. The authors conduct a meta-analysis of existing studies, demonstrating that digital notifications can significantly reduce missed appointments and enhance patient adherence to scheduled visits.

The paper "Use of Short Message Service and Smartphone Applications in the Management of Surgical Patients: A Systematic Review" by Lu et al. [21] examines the role of SMS and smartphone apps in managing surgical patients. The authors analyze their impact on patient education, preoperative preparation, and postoperative recovery. Their findings suggest that these digital tools can improve patient outcomes and streamline surgical workflows.

Chapter 3

Problem Definition and Objectives

3.1 Problem Statement

Hospitals face significant challenges with inefficiencies in patient flow, administrative burdens, and fragmented communication, all of which negatively impact patient satisfaction and operational effectiveness. Delays in patient movement, excessive paperwork lead to longer wait times, increased administrative workload, and poor communication between departments and healthcare providers, ultimately resulting in a subpar patient experience and higher operational costs. This project addresses these issues by implementing an automated, integrated system that streamlines hospital management, optimizes resource utilization, and enhances communication between patients and providers. The system reduces administrative tasks, ensures real-time access to patient data, and helps efficiently allocate resources, leading to faster and more accurate decision-making. The result is a seamless healthcare experience where patients receive timely, coordinated care, and hospitals can operate more efficiently, improving patient outcomes, increasing satisfaction, and reducing operational costs.

3.2 Objectives

1. **Enable Patients to Book Appointments via a Self-Service Kiosk:** Provide patients with the ability to independently schedule their appointments through an intuitive, user-friendly self-service kiosk, streamlining the registration process and reducing wait times.

2. Implement an AI-Powered Chatbot for Symptom Analysis and Specialist

Recommendations: Integrate an intelligent AI chatbot that collects and analyzes patient symptoms to provide accurate specialist recommendations, improving diagnosis accuracy and ensuring that patients are directed to the appropriate healthcare professional.

3. Provide Seamless Payment Integrations for Consultations and Lab Tests:

Incorporate secure, seamless payment gateways for consultation fees and lab tests, allowing patients to pay quickly and efficiently via methods like credit/debit cards or mobile wallets, enhancing the overall patient experience.

4. Facilitate Communication Between Patients and Doctors Through SMS

and Notification Services: Use automated SMS and real-time notifications to keep patients informed about appointment confirmations, specialist details, and lab test results, while also sending updates and reminders to doctors to ensure timely follow-ups.

5. Manage Lab Test Guidelines, Reports, and Doctor Access Efficiently:

Develop a streamlined system for managing lab test procedures, providing clear pre-test guidelines for patients, and ensuring secure, timely access to test reports. Lab technicians can upload results directly into the system, while doctors can access and review these reports seamlessly, facilitating quick diagnosis and improving patient care.

Chapter 4

Software Requirements Specification

4.1 Introduction

The System Requirements Specification (SRS) outlines the requirements for the System. The system is designed to streamline hospital workflows, improve patient interactions, and enhance hospital operations. The document highlights the hardware, software, and user-specific requirements to ensure smooth implementation and usability.

4.2 Purpose

The primary purpose of the system is to automate and simplify hospital management processes by integrating multiple functionalities into a kiosk-based interface. The system enables:

- Patient self-registration using OTP-based authentication.
- Symptom analysis using an AI-powered chatbot(NLP) to recommend specialists.
- Appointment scheduling and consultation fee payments.
- Lab test management for patients and technicians, including report uploads and secure payments.
- Notification-based coordination between patients and doctors: patients are informed about doctor availability, and doctors are notified about the patients scheduled to visit them.

The system aims to reduce hospital workload, minimize waiting time, and offer a seamless, user-friendly experience.

4.3 User Characteristics

The system is designed for users with varying levels of technical proficiency. The key user groups include:

1. Patients:

- Minimal technical knowledge is required to operate the system.
- Patients can easily follow step-by-step instructions for registration, symptom input, appointment booking, and payments.

2. Doctors:

- Doctors use the CareLink app to view patient appointments, prescribe lab tests, and access uploaded reports.
- Basic familiarity with mobile or web-based apps is expected.

3. Lab Technicians:

- Lab technicians use the CareLink app to upload test reports in PDF format.
- Requires basic knowledge of document uploads and report handling.

4.4 Interfaces

4.4.1 Hardware Interfaces

The system requires the following hardware components:

1. Kiosk Hardware:

- Touchscreen display for patient interaction.
- OTP-enabled functionality for patient registration and authentication.

2. Server Hardware:

- On-premise server to host the backend database, chatbot functionalities, payment gateway, and SMS integration.

3. Connectivity:

- Reliable internet connection to enable seamless data exchange, SMS notifications, and real-time synchronization with the CareLink app.

4. Communication:

- SMS service integration to notify patients about appointment confirmations, doctor details, and lab test guidelines.

4.4.2 Software Interfaces

The system interacts with various software components, including:

1. Operating Systems:

- Kiosk System: Windows OS.
- Backend Server: Windows Server.

2. Backend Framework:

- Node.js with Express.js for API development.
- MongoDB for storing patient, doctor, and lab test data.

3. Frontend Development:

- Kiosk UI: Developed using React.js for an interactive user interface.
- CareLink App: Mobile-friendly app interface built with Flutter or React Native.

4. Third-Party Services:

- Payment Gateway: Razorpay is integrated for secure and reliable online payment transactions.
- SMS Notifications: Twilio is used to send notifications to users seamlessly.

Chapter 5

System Design

5.1 Architecture Design

An architecture diagram is a visual representation of the system's structure, components, and their interactions. It provides a clear overview of how the system is designed, showing key parts, their relationships, and data flow.

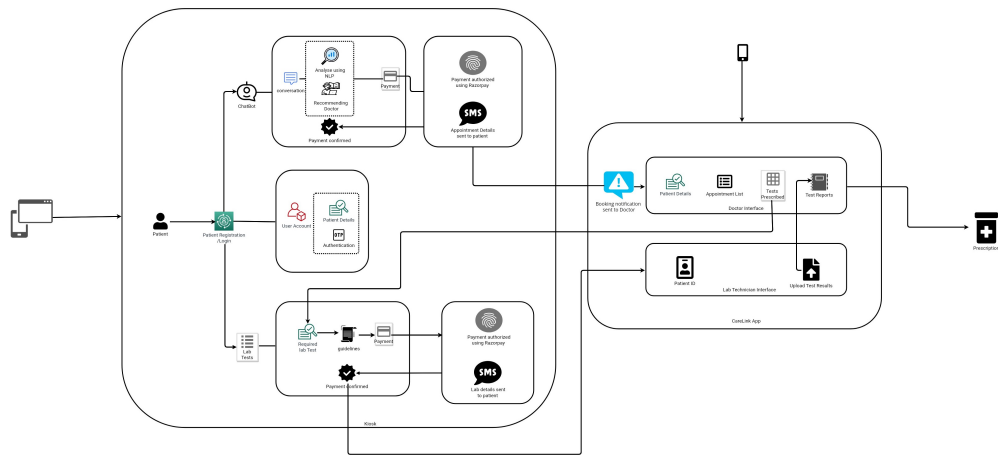


Figure 5.1: System Architecture Diagram

The architecture diagram provides a **high-level overview** of the system workflow for the kiosk-based system. The explanation is as follows:

Key Components

1. Patient Interface

- Patients log in or register using the kiosk.

- OTP authentication is used for security.
- Patients interact with the chatbot to input symptoms, and based on AI mapping, specialists are recommended.
- Payment for consultations is done using Razorpay.
- SMS notifications are sent to confirm doctor details and appointments.

2. Doctor Interface (CareLink App)

- Doctors receive notifications about patient bookings.
- They can prescribe lab tests through the app.

3. Lab Technician Interface

- Lab technicians can view lab test details and upload test reports.
- Uploaded reports are shared with doctor.

4. Payment and Notifications

- Payments (consultations and lab tests) are managed using Razorpay.
- SMS notifications are sent to keep patients updated on appointments and lab test details.

5. System Flow

- The kiosk integrates multiple functionalities such as registration, symptom analysis (chatbot), specialist recommendation, and payments.
- Data flows seamlessly between the kiosk, CareLink app (used by doctors and lab technicians), and external systems like Razorpay for payments.

Concepts Used

- **AI & NLP:** Used in the chatbot for symptom mapping and doctor recommendations.
- **Payment Gateway Integration:** Razorpay is used for handling payments securely.
- **SMS API:** Real-time notifications for appointments, lab results, and payments.

- **Data Flow & Integration:** Smooth communication between patient kiosk, doctor interface, and lab technician interface.

Uses

- Explains system workflows.
- Helps developers design and implement the system efficiently.
- Aids in scaling the system or adding new features.

5.2 Class Diagram

A class diagram is a type of UML (Unified Modeling Language) diagram that visually represents the static structure of a system. It models the classes, their attributes, operations (methods), and the relationships among objects in a software system.

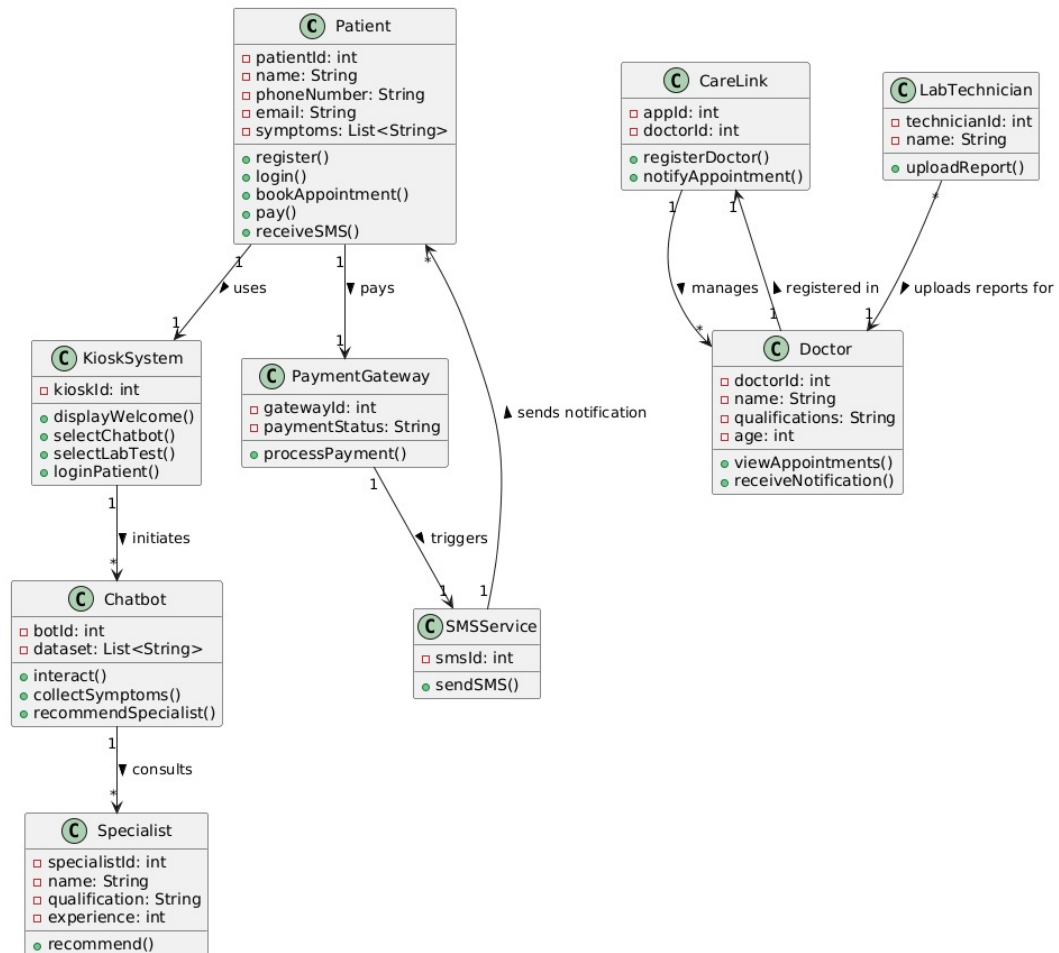


Figure 5.2: Class Diagram

Key Components

- **Patient:** Central class performing registration, login, booking appointments, making payments, and receiving notifications.
- **KioskSystem:** Interface for patients to log in, interact with the chatbot, or select lab tests.
- **Chatbot:** Collects symptoms and recommends specialists based on input.
- **Specialist:** Provides recommendations based on the symptoms collected by the chatbot.

- **PaymentGateway:** Handles payment processes and confirms transactions.
- **SMSService:** Sends SMS notifications for payments and appointments.
- **Doctor:** Views appointments and receives notifications.
- **CareLink:** Registers doctors and manages appointment notifications.
- **LabTechnician:** Uploads lab reports for patient and doctor access.

Concepts Used

- **Object-Oriented Programming (OOP):** Classes, attributes, and methods represent real-world entities.
- **Encapsulation:** Attributes are private, and methods are used for access.
- **Modular Design:** Each class has a specific role, ensuring clarity and maintainability.
- **Interactions:** Arrows show relationships such as “uses,” “triggers,” and “uploads reports.”

Uses

- **System Understanding:** Provides a clear visual representation of components and their interactions.
- **Development Reference:** Serves as a blueprint for system design and development.
- **Workflow Automation:** Demonstrates automation of hospital processes like registration, payments, lab test reports, and doctor-patient interactions.
- **Scalability:** Allows easy extension for additional modules (e.g., pharmacy or billing systems).

5.3 State Diagram

A State Diagram is a UML diagram used to model the dynamic behavior of a system by illustrating states, transitions, and events. It shows how an object changes states based on conditions or events.

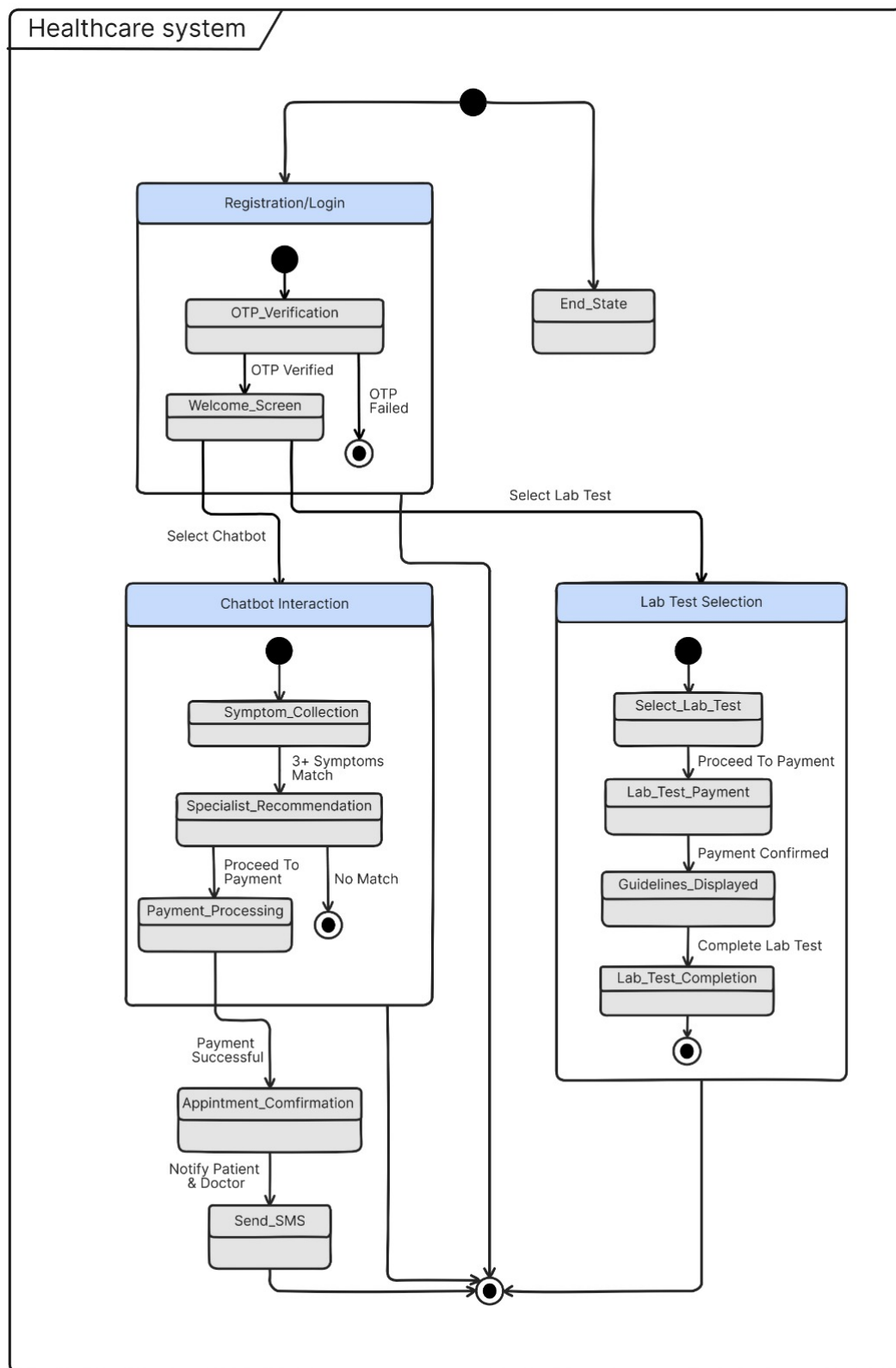


Figure 5.3: State Diagram

Key Components

- **States:**
 - **Initial State:** Start point of the process.
 - **Registration/Login:**
 - * OTP Verification (States: OTP Verified, OTP Failed)
 - * Welcome Screen
 - **Chatbot Interaction:**
 - * Symptom Collection
 - * Specialist Recommendation
 - * Payment Processing
 - **Lab Test Selection:**
 - * Select Lab Test
 - * Lab Test Payment
 - * Payment Confirmed
 - * Guidelines Displayed
 - * Lab Test Completion
 - **End State:** Represents the final state.
- **Transitions:** Flow between states like “OTP Verification → Welcome Screen” or “Payment Processing → Appointment Confirmation.”
- **Decision Nodes:** Conditions such as “OTP Failed” or “3+ Symptoms Match” that determine alternate paths.
- **Control Flow:** Arrows indicate the transitions between states.

Concepts Used

- **State Transitions:** Models the change of states based on user actions like successful payment or OTP validation.
- **Conditional Logic:** Determines the state flow based on conditions (e.g., OTP verified or symptoms matching).

- **Sequential Flow:** Logical progression from one state to another with a clear start and end.

Uses

- Visualizes the **lifecycle** of the healthcare system processes.
- Helps in understanding the **conditional logic** and transitions between states.
- Useful for **system behavior modeling**, especially for error handling (e.g., OTP Failed).
- Assists developers in implementing **state-based logic** within the system.

5.4 Use Case Diagram

A Use Case Diagram is a UML diagram that models the interactions between users (actors) and a system to describe functional requirements. It represents high-level system functionalities and user roles.

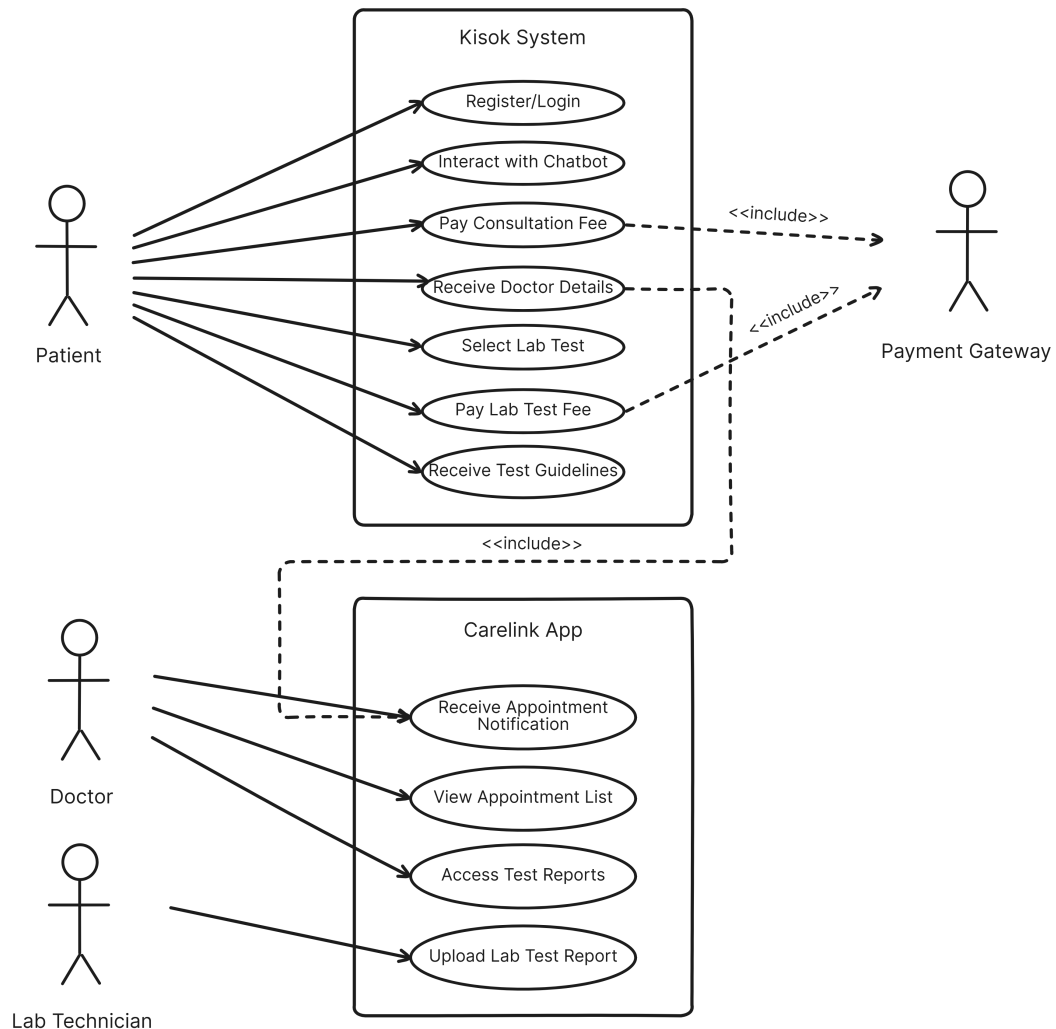


Figure 5.4: Use Case Diagram

Key Components

- **Actors:**

- **Patient:** Interacts with the kiosk system for activities like registration, payments, and selecting lab tests.
- **Doctor:** Receives appointment notifications and accesses lab test reports via the Carelink App.
- **Lab Technician:** Uploads lab test reports.

- **Payment Gateway:** Processes payments for consultation and lab tests.
- **Use Cases:**
 - **Kiosk System:**
 - * Register/Login
 - * Interact with Chatbot
 - * Pay Consultation Fee
 - * Receive Doctor Details
 - * Select Lab Test
 - * Pay Lab Test Fee
 - * Receive Test Guidelines
 - **Carelink App:**
 - * Receive Appointment Notification
 - * View Appointment List
 - * Access Test Reports
 - * Upload Lab Test Report
- **Relationships:**
 - **Include Relationship:** Payment interactions involve the Payment Gateway, indicated by dashed lines.
 - **System Boundary:** Divides system functionalities into the Kiosk System and Carelink App.

Concepts Used

- **Actor-Use Case Relationships:** Shows the roles of patients, doctors, and lab technicians and how they interact with the system functionalities.
- **System Modularity:** Separation of Kiosk System and Carelink App for clear roles.
- **Extensibility:** Payment Gateway inclusion enables seamless integration for fee handling.

Uses

- Provides a **functional overview** of the healthcare system.
- Helps stakeholders identify the roles of **actors** and their interactions.
- Useful for understanding **system boundaries** and external integrations.
- Acts as a guide for developers to implement user functionalities.

5.5 Sequence Diagram

A Sequence Diagram is a type of UML (Unified Modeling Language) diagram that illustrates how objects interact in a particular scenario of a system over time. It is a part of the Behavioral UML Diagrams and focuses on the sequence of messages exchanged between various components or objects.

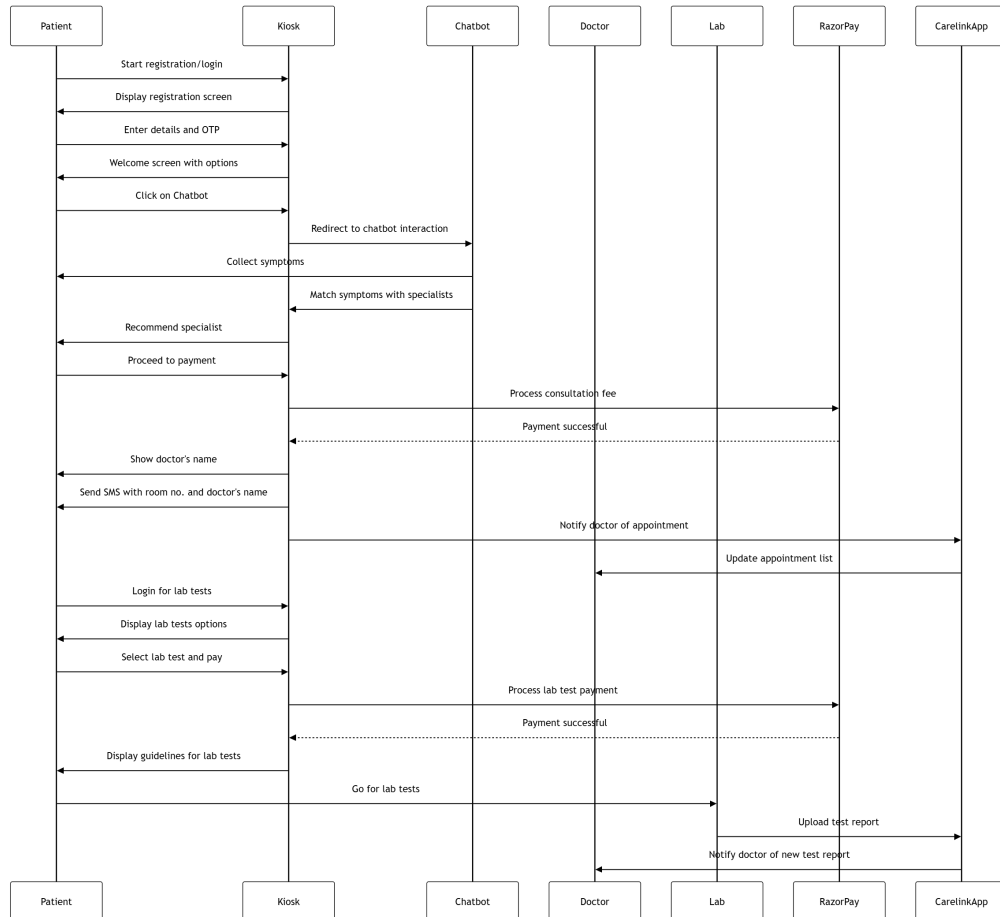


Figure 5.5: Sequence Diagram

Key Components

- **Actors:** Patient, Kiosk System, OTP Service, Chatbot, Specialist DB, Payment Gateway, SMS Service, CareLink, Doctor, Lab Technician.
- **Interactions:** Messages passed between actors in a timeline format.
- **Lifelines:** Vertical lines for each entity that represent their life during the process.
- **Arrows:** Represent synchronous and asynchronous messages.

Concepts Used

- **Sequence of Operations:** Shows the step-by-step flow of operations and communication between entities.
- **Request-Response Mechanism:** Actions by the patient trigger responses from systems (e.g., OTP validation, payment processing).

Uses

- Visualizes **system interactions** for the hospital kiosk system.
- Helps in understanding **workflow timing** and the sequence of operations.
- Identifies **dependencies** between modules.
- Useful for **system architects** and developers to design and debug workflows.

5.6 Activity Diagram

An Activity Diagram is a behavioral UML diagram that represents the flow of activities in a process or system. It is used to model workflows and the sequential or parallel execution of activities. Activity diagrams focus on how tasks are executed, their order, and the conditions that control the flow.

It is particularly helpful for understanding business processes, system behavior, and the logical flow of activities.

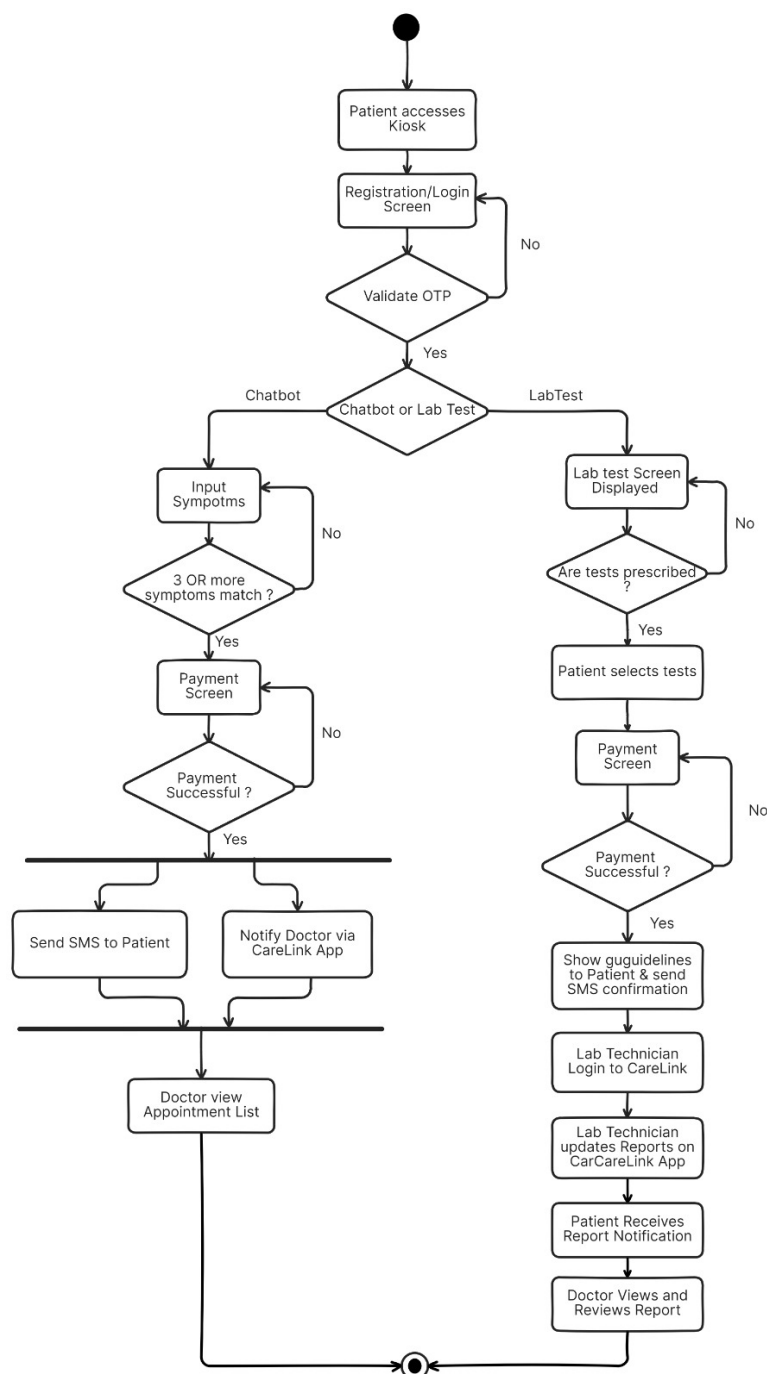


Figure 5.6: Activity Diagram

Key Components

- **Actions/Decisions:** Registration/Login, OTP Validation, Symptoms Input or Lab Test Selection, Payment Processing, Notifications, Lab Report Management.
- **Decision Nodes:** Conditions like “Are tests prescribed?” or “Payment Successful?”.
- **Control Flow:** Arrows represent the flow of control between activities.

Concepts Used

- **Conditional Logic:** Decisions such as OTP validation success or test prescriptions.
- **Parallel Processes:** Actions like sending SMS or notifying doctors occur concurrently.
- **End-to-End Process:** Depicts the full workflow from kiosk access to report notifications.

Uses

- Visualizes the **dynamic behavior** of the hospital kiosk system.
- Useful for identifying **bottlenecks** and understanding **decision-making processes**.
- Assists in **business process modeling** for optimization.
- Helps developers and analysts understand **system logic** and data flow.

5.7 Data Flow Diagram

A Data Flow Model is a graphical representation of how data moves through a system, illustrating the processes, data stores, and external entities involved. It focuses on the transformation of data as it flows between components within a system. The model provides a clear, structured view of how inputs are transformed into outputs, helping stakeholders and developers understand the flow of information and identify inefficiencies or areas for improvement

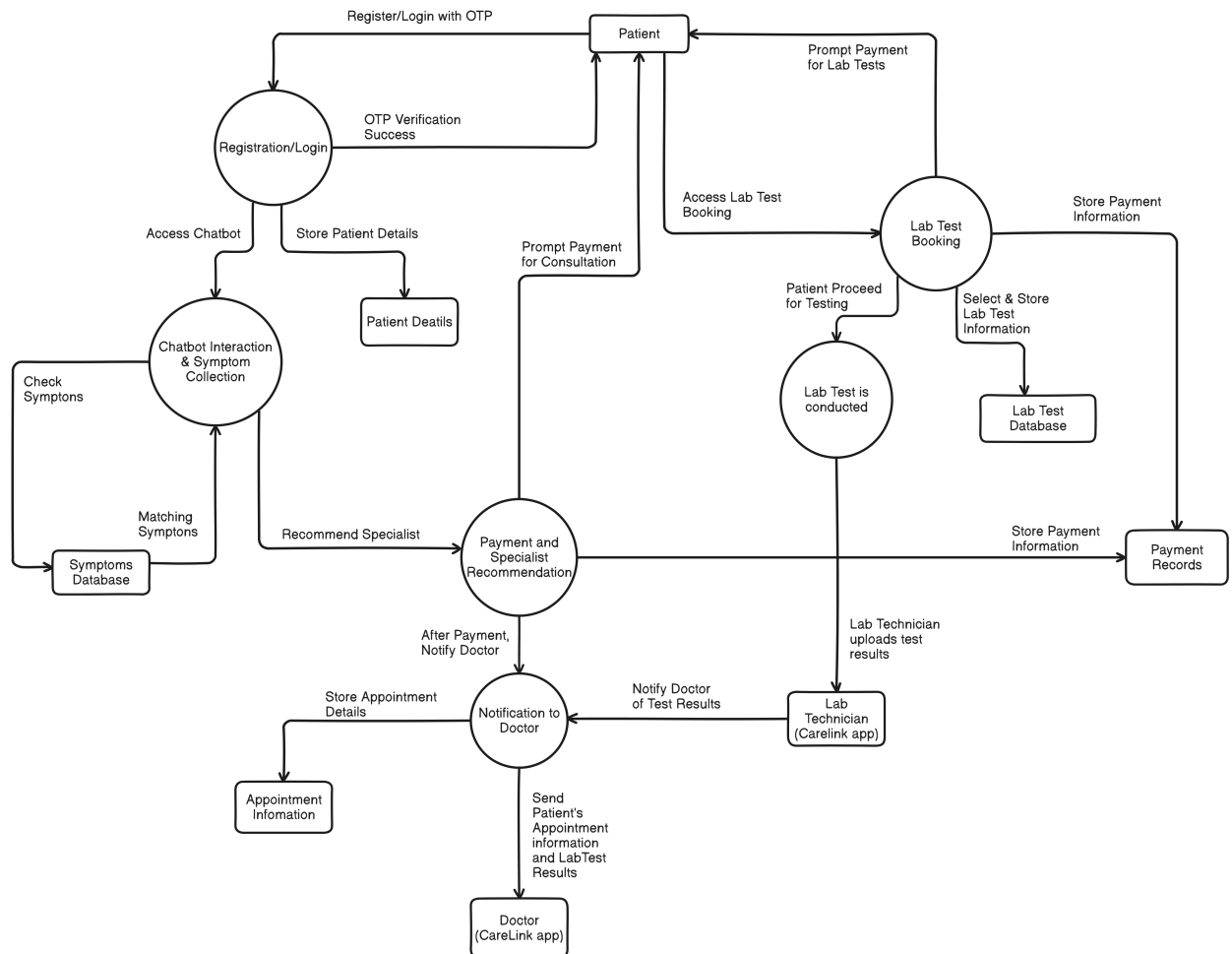


Figure 5.7: Data Flow Diagram

Key Components

- **External Entities:**

- **Patient:** Initiates registration, payment, and lab test booking.
- **Doctor (CareLink App):** Receives appointment notifications and lab test results.

- **Lab Technician (CareLink App):** Uploads lab test results after conducting tests.
- **Processes:**
 - **Registration/Login:** Verifies OTP and stores patient details.
 - **Chatbot Interaction & Symptom Collection:** Collects symptoms, checks them against the database, and recommends specialists.
 - **Payment and Specialist Recommendation:** Prompts for payments and notifies doctors.
 - **Lab Test Booking:** Manages lab test selection and payment.
 - **Notification to Doctor:** Sends appointment and lab test details to doctors.
 - **Lab Test is Conducted:** Conducts lab tests and uploads results.
- **Data Stores:**
 - **Symptoms Database:** Stores symptom information for matching purposes.
 - **Lab Test Database:** Stores selected lab test details.
 - **Payment Records:** Stores payment details for consultation and lab tests.
 - **Appointment Information:** Holds appointment-related data.
- **Data Flow:**
 - Arrows indicate the flow of data between processes, data stores, and external entities.
 - Examples: OTP Verification Success, Store Patient Details, Notification to Doctor.

Concepts Used

- **Process Flow:** Represents how data moves through different processes like registration, payment, lab test booking, and notification.
- **Data Storage:** Highlights databases such as the Symptoms Database, Lab Test Database, and Payment Records that store critical information.

- **Interaction Between Entities:** Models interactions between external entities (Patient, Doctor, Lab Technician) and internal processes.
- **Data Transformation:** Focuses on how inputs like symptoms or test booking details are processed into outputs like specialist recommendations or lab test results.

Uses

- Provides a **visual representation** of data flow within the healthcare system.
- Helps identify **data dependencies** and critical processes in the workflow.
- Useful for **system analysis** to understand how data is processed and stored.
- Facilitates communication between developers, stakeholders, and analysts by depicting system behavior.
- Supports **optimization** of data handling and process efficiency within the system.

5.8 Modular Diagram

A Modular Diagram is a representation of a system that shows how a complex system is broken down into smaller, manageable, and independent modules. Each module represents a distinct functionality or component of the system, making it easier to understand, design, and manage.

Modular diagrams are commonly used in software engineering, system design, and project management to demonstrate the structure of a system and the interaction between its various components.

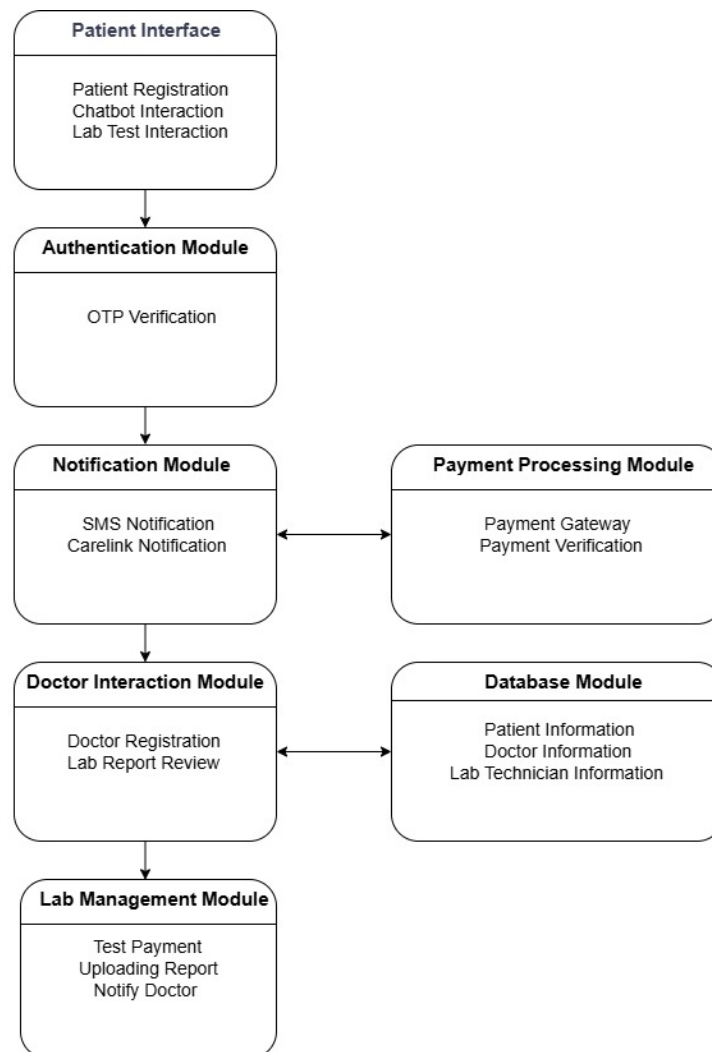


Figure 5.8: Modular Diagram

Key Components

- **Patient Interface:** Registration, chatbot interaction, and lab test processes.
- **Authentication Module:** OTP verification process.

- **Notification Module:** Sends notifications via SMS and CareLink.
- **Payment Processing Module:** Manages payment gateways and verifies transactions.
- **Doctor Interaction Module:** Doctor registration and lab report reviews.
- **Database Module:** Stores patient, doctor, and lab technician information.
- **Lab Management Module:** Handles lab test payment, report uploads, and doctor notifications.

Concepts Used

- **Modularity:** The system is broken down into independent, manageable modules.
- **Separation of Concerns:** Each module performs a specific task.
- **Interdependencies:** Modules interact to complete workflows.

Uses

- Provides a **high-level view** of the system's architecture.
- Helps in **module testing** and independent development.
- Promotes **scalability** and maintainability.
- Assists developers and stakeholders in **understanding functionalities**.

Chapter 6

Implementation

6.1 Algorithm

The Doctor Suggestion System in the project uses a rule-based matching algorithm to suggest a doctor based on the symptoms that the user provides. This algorithm is simple yet effective for the task at hand. Here's how it works:

1. Input Processing:

- User symptoms are collected as a string (e.g., “fever, headache”).
- The symptoms are preprocessed to ensure consistency:
 - Converted to lowercase.
 - Extraneous spaces, punctuation, and stop words (such as “I have”) are stripped.
 - Symptoms are broken up into a list for further processing.

2. Dataset Preprocessing:

- A loaded dataset includes data about doctors, their specializations, room numbers, and corresponding symptoms.
- The symptoms of each doctor are preprocessed just like the user's input for uniformity.

3. Symptom Matching:

- For each doctor in the dataset, the system matches the user's symptoms with the doctor's defined symptoms.

- A count of matching symptoms is calculated with an iteration:
 - Each user symptom is checked against the doctor's symptom list.
 - In case of a match, the counter is incremented.

4. Choosing the Best Doctor:

- The best doctor will be the one who has the highest number of matching symptoms. However, there must be a threshold for making the recommendation meaningful:
 - In case the number of matching symptoms is less than 2, no specific doctor will be recommended.
 - The recommendation also includes the doctor's name, specialization, consultation room number, and price.

5. Payment Verification:

- Before showing the doctor's details, the system verifies if the payment is done.
- If the payment is pending, the user is asked to complete the payment to proceed further.

6. SMS Notification:

- After recommending a doctor, the system sends an SMS to the user and relevant stakeholders using the Twilio API.

Module 1: Patient Registration and Login

Purpose: Allow patients to register or log in to the kiosk using OTP.

Pseudo Code:

```
1 1. Start
2 2. Display options: "Register" or "Login"
3 3. IF Register:
4     a. Input: Name, Phone Number, Address
5     b. Generate OTP and send to the provided phone number
6     c. Verify OTP
7     d. IF OTP is correct:
8         i. Save patient details in the database
9         ii. Display success message
10    e. ELSE:
11        i. Display "Invalid OTP" message
12 4. IF Login:
13    a. Input: Phone Number
14    b. Generate OTP and send to the provided phone number
15    c. Verify OTP
16    d. IF OTP is correct:
17        i. Authenticate patient
18    e. ELSE:
19        i. Display "Invalid OTP" message
20 5. End
```

Module 2: Chatbot Symptom Analysis and Specialist Recommendation

Purpose: Collect symptoms from patients and recommend a specialist using predefined mappings.

Pseudo Code:

```
1 1. Start
2 2. Input: Symptoms from the patient
3 3. Load dataset: Symptoms-Specialist mapping
4 4. For each symptom in the patient input:
5     a. Match the symptom with the dataset
6 5. IF three or more symptoms match:
7     a. Recommend the mapped specialist
8 6. ELSE:
9     a. Display "Consult a General Physician" message
10 7. End
```

Module 3: Payment System

Purpose: Handle secure payments for consultations and lab tests using Razorpay.

Pseudo Code:

```
1 1. Start
2 2. Input: Patients payment details (credit card, UPI, etc.)
3 3. Send payment details to Razorpay API
4 4. IF payment is successful:
5     a. Update payment status in the database
6     b. Display success message
7 5. ELSE:
8     a. Display "Payment Failed" message
9 6. End
```

Module 4: Lab Test Management

Purpose: Allow patients to book prescribed lab tests and upload lab reports.

Pseudo Code:

```
1 1. Start
2 2. Input: Patient logs in and views prescribed lab tests
3 3. Display available lab tests
4 4. Input: Selected lab tests and payment details
5 5. Process payment via Razorpay
6 6. IF payment is successful:
7     a. Save lab test booking in the database
8     b. Display guidelines for the test
9 7. ELSE:
10    a. Display "Payment Failed" message
11 8. Lab Technician uploads test reports to CareLink
12 9. Doctor views reports and provides prescription
13 10. End
```

Module 5: Doctor CareLink App

Purpose: Allow doctors to view appointments, notifications, and lab reports.

Pseudo Code:

```
1 1. Start
2 2. Doctor logs in to CareLink app
3 3. Display list of todays appointments
4 4. For each appointment:
5     a. Show patient details (name, symptoms, time)
6 5. IF lab tests are prescribed:
7     a. Access uploaded lab reports
8 6. Provide final diagnosis and prescriptions
9 7. End
```

6.2 Flow Chart

A flowchart is a diagram that visually represents the steps, decisions, and workflows in a process. It uses standard symbols like arrows, rectangles, diamonds, and ovals to show the sequence of actions or decision points in a process. Flowcharts are used to analyze, design, and document workflows, processes, and systems.

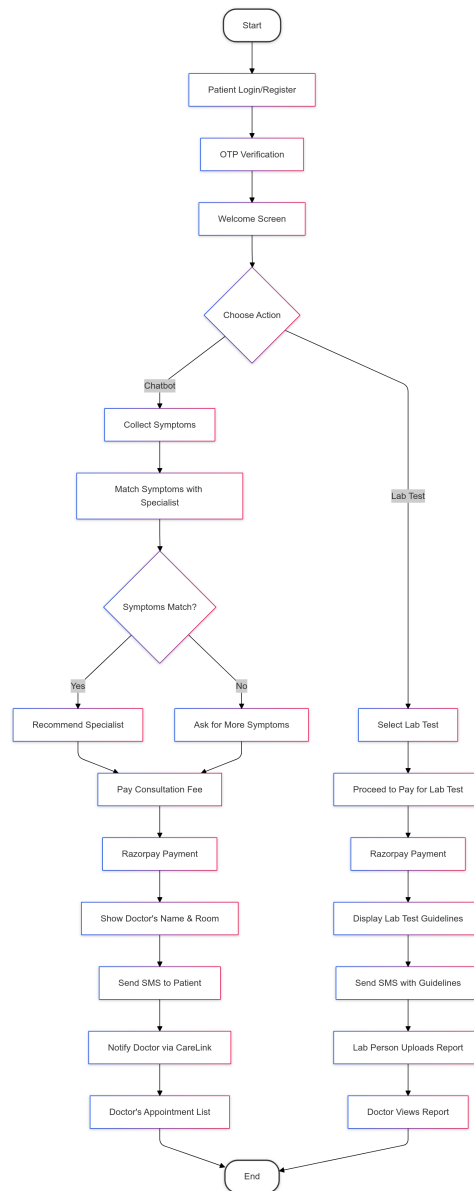


Figure 6.1: Flow Chart

6.3 Implementation Codes

6.3.1 Code for Doctor suggestion based on Symptoms

```
1 # Suggest doctor based on symptoms
2 def suggest_doctor(user_symptoms, payment_status):
3     # Preprocess the user input
4     user_symptoms_list = clean_user_input(user_symptoms)
5     if not user_symptoms_list:
6         return {"message": "No symptoms provided."}
7
8     best_match = None
9     max_matches = 0
10
11     # Iterate through the dataset to find the best match
12     for _, row in df.iterrows():
13         # Preprocess the symptoms listed for each doctor
14         doctor_symptoms = preprocess_symptoms(row['Symptoms'])
15
16         # Count the number of matching symptoms
17         match_count = sum(1 for symptom in user_symptoms_list if symptom in
18                             doctor_symptoms)
19
20         # Update the best match if the current doctor has more matches
21         if match_count > max_matches:
22             max_matches = match_count
23             best_match = row
24
25     # Check if a suitable doctor is found
26     if best_match and max_matches >= 2:
27         if not payment_status:
28             return {"message": f"Payment required to view doctor details."}
29
30         return {
31             "message": f"Doctor: {best_match['Doctor']}, Room: {best_match['Room Number']}"
32         }
```

```
32  
33     return {"message": "No suitable doctor found."}
```

6.3.2 Code for Twilio Intergration

```
1  from flask import Flask, request, jsonify  
2  from flask_cors import CORS  
3  import os  
4  from twilio.rest import Client  
5  
6  # Initialize Flask app and enable CORS  
7  app = Flask(__name__)  
8  CORS(app)  
9  
10 # Load Twilio credentials from environment variables or use defaults  
11 TWILIO_ACCOUNT_SID = os.getenv('TWILIO_ACCOUNT_SID', 'default_sid')  
12 TWILIO_AUTH_TOKEN = os.getenv('TWILIO_AUTH_TOKEN', 'default_token')  
13 TWILIO_PHONE_NUMBER = os.getenv('TWILIO_PHONE_NUMBER', 'default_number')  
14 USER_PHONE_NUMBERS = ['+918804339456', '+919632983944', '+919481680079', '  
    +919743352610']  
15  
16 # Initialize Twilio client  
17 client = Client(TWILIO_ACCOUNT_SID, TWILIO_AUTH_TOKEN)  
18  
19 # Function to send SMS to multiple phone numbers  
20 def send_sms(message):  
21     for phone_number in USER_PHONE_NUMBERS:  
22         try:  
23             response = client.messages.create(  
24                 body=message,  
25                 from_=TWILIO_PHONE_NUMBER,  
26                 to=phone_number  
27             )  
28             print(f"SMS sent successfully to {phone_number}! Message SID: {  
                response.sid}")  
29         except Exception as e:
```

```

30         print(f"Failed to send SMS to {phone_number}: {e}")
31         if "Authenticate" in str(e):
32             print("Check Twilio SID, Auth Token, and phone numbers.")
33
34 # API endpoint to handle notifications
35 @app.route('/send-notification', methods=['POST'])
36 def send_notification():
37     try:
38         data = request.get_json()
39         print("Received data:", data)
40
41         message = data.get('message')
42
43         if not message:
44             return jsonify({"error": "Message not provided."}), 400
45
46         send_sms(message)
47         return jsonify({"success": True, "message": "Notification sent
48             successfully."})
49     except Exception as e:
50         print(f"Error: {e}")
51         return jsonify({"error": "An unexpected error occurred."}), 500
52
53 # Run the application
54 if __name__ == "__main__":
55     app.run(debug=True, port=4000)

```

6.3.3 Doctor Dashboard in Carelink

```

1 class _DashboardState extends State<Dashboard> with
2     SingleTickerProviderStateMixin {
3     String _formattedTime = '';
4     bool _isActive = true;
5     int _appointmentCount = 0;
6     Map<String, dynamic>? _selectedAppointmentDetails;
7     late AnimationController _animationController;

```

```
7   late Animation<double> _animation;
8
9   @override
10  void initState() {
11    super.initState();
12    _updateTime();
13    Timer.periodic(Duration(seconds: 1), (_) => _updateTime());
14    _animationController = AnimationController(
15      duration: Duration(milliseconds: 300),
16      vsync: this,
17    );
18    _animation = CurvedAnimation(parent: _animationController, curve: Curves.
      easeInOut);
19  }
20
21  void _updateTime() {
22    setState(() {
23      _formattedTime = DateFormat('HH:mm:ss').format(DateTime.now());
24    });
25  }
26
27  void _toggleDetails() {
28    setState(() {
29      _isExpanded = !_isExpanded;
30      _isExpanded ? _animationController.forward() : _animationController.
        reverse();
31    });
32  }
33
34  @override
35  void dispose() {
36    _animationController.dispose();
37    super.dispose();
38  }
39
40  @override
41  Widget build(BuildContext context) {
```

```

41     return Scaffold(
42       appBar: AppBar(
43         title: Text('Doctor Dashboard'),
44         leading: IconButton(
45           icon: Icon(Icons.menu),
46           onPressed: () => Scaffold.of(context).openDrawer(),
47         ),
48         actions: [
49           IconButton(
50             icon: Icon(Icons.notifications),
51             onPressed: () {
52               showModalBottomSheet(
53                 context: context,
54                 builder: (_) => _appointmentCount > 0
55                   ? ListView.builder(
56                     itemCount: _appointmentCount,
57                     itemBuilder: (_, index) {
58                       return ListTile(
59                         leading: Icon(Icons.person),
60                         title: Text('Appointment #$index'),
61                       );},)
62                   : Center(child: Text('No new notifications.')),
63                 );},),],),
64       drawer: Drawer(
65         child: ListView(
66           padding: EdgeInsets.zero,
67           children: [
68             DrawerHeader(
69               decoration: BoxDecoration(color: Colors.blue),
70               child: Text('Menu', style: TextStyle(color: Colors.white, fontSize
71                 : 24)),
72             ),
73             ListTile(
74               leading: Icon(Icons.dashboard),
75               title: Text('Dashboard'),
76               onTap: () => Navigator.pop(context),

```

```

76         ),
77         ListTile(
78             leading: Icon(Icons.person),
79             title: Text('Profile'),
80             onTap: () => Navigator.push(context, MaterialPageRoute(builder: (_)
81                 => DoctorProfile())),
82         ),
83         ListTile(
84             leading: Icon(Icons.logout),
85             title: Text('Logout'),
86             onTap: () => Navigator.pushReplacement(context, MaterialPageRoute(
87                 builder: (_) => LoginPage())),
88         ),],),),
89     body: Padding(
90         padding: EdgeInsets.all(16.0),
91         child: Column(
92             crossAxisAlignment: CrossAxisAlignment.stretch,
93             children: [
94                 Card(
95                     color: Colors.blue[50],
96                     child: Padding(
97                         padding: EdgeInsets.all(16.0),
98                         child: Column(
99                             crossAxisAlignment: CrossAxisAlignment.start,
100                             children: [
101                                 Text('Appointments', style: TextStyle(fontSize: 18,
102                                     fontWeight: FontWeight.bold)),
103                                 Text('Total: $_appointmentCount', style: TextStyle(fontSize:
104                                     16)),
105                             ],),),),
106                 SizedBox(height: 20.0),
107                 Card(
108                     child: Padding(
109                         padding: EdgeInsets.all(16.0),
110                         child: Column(
111                             crossAxisAlignment: CrossAxisAlignment.start,

```

```

108         children: [
109             Text('Select Appointment', style: TextStyle(fontSize: 18,
110                 fontWeight: FontWeight.bold)),
111             DropdownButton<String>(
112                 isExpanded: true,
113                 hint: Text('Select Patient'),
114                 value: _selectedAppointmentDetails?['patientId'],
115                 items: ['Patient 1', 'Patient 2'].map((patient) {
116                     return DropdownMenuItem<String>(
117                         value: patient,
118                         child: Text(patient),
119                     );
120                 }).toList(),
121                 onChanged: (value) {
122                     setState(() {
123                         _selectedAppointmentDetails = {'patientId': value};
124                     });
125                 },
126             ),
127             SizeTransition(
128                 sizeFactor: _animation,
129                 child: _selectedAppointmentDetails != null
130                     ? Padding(
131                         padding: EdgeInsets.only(top: 16.0),
132                         child: Column(
133                             crossAxisAlignment: CrossAxisAlignment.start,
134                             children: [
135                                 Text('Details for ${_selectedAppointmentDetails
136                                     !['patientId']}', style: TextStyle(fontSize:
137                                     16, fontWeight: FontWeight.bold)),
138                             ],
139                         ),
140                     ) : SizedBox.shrink(),
141             ),
142             ElevatedButton(
143                 onPressed: _toggleDetails,
144                 child: Text(_isExpanded ? 'Hide Details' : 'Show Details'),
145             ),
146         ],
147     ),
148 ),
149 ),
150 ),
151 ),
152 ),
153 ),
154 ),
155 ),
156 ),
157 ),
158 ),
159 ),
160 ),
161 ),
162 ),
163 ),
164 ),
165 ),
166 ),
167 ),
168 ),
169 ),
170 ),
171 ),
172 ),
173 ),
174 ),
175 ),
176 ),
177 ),
178 ),
179 ),
180 ),
181 ),
182 ),
183 ),
184 ),
185 ),
186 ),
187 ),
188 ),
189 ),
190 ),
191 ),
192 ),
193 ),
194 ),
195 ),
196 ),
197 ),
198 ),
199 ),
200 ),
201 ),
202 ),
203 ),
204 ),
205 ),
206 ),
207 ),
208 ),
209 ),
210 ),
211 ),
212 ),
213 ),
214 ),
215 ),
216 ),
217 ),
218 ),
219 ),
220 ),
221 ),
222 ),
223 ),
224 ),
225 ),
226 ),
227 ),
228 ),
229 ),
230 ),
231 ),
232 ),
233 ),
234 ),
235 ),
236 ),
237 ),
238 ),
239 ),
240 ),
241 ),
242 ),
243 ),
244 ),
245 ),
246 ),
247 ),
248 ),
249 ),
250 ),
251 ),
252 ),
253 ),
254 ),
255 ),
256 ),
257 ),
258 ),
259 ),
260 ),
261 ),
262 ),
263 ),
264 ),
265 ),
266 ),
267 ),
268 ),
269 ),
270 ),
271 ),
272 ),
273 ),
274 ),
275 ),
276 ),
277 ),
278 ),
279 ),
280 ),
281 ),
282 ),
283 ),
284 ),
285 ),
286 ),
287 ),
288 ),
289 ),
290 ),
291 ),
292 ),
293 ),
294 ),
295 ),
296 ),
297 ),
298 ),
299 ),
300 ),
301 ),
302 ),
303 ),
304 ),
305 ),
306 ),
307 ),
308 ),
309 ),
310 ),
311 ),
312 ),
313 ),
314 ),
315 ),
316 ),
317 ),
318 ),
319 ),
320 ),
321 ),
322 ),
323 ),
324 ),
325 ),
326 ),
327 ),
328 ),
329 ),
330 ),
331 ),
332 ),
333 ),
334 ),
335 ),
336 ),
337 ),
338 ),
339 ),
340 ),
341 ),
342 ),
343 ),
344 ),
345 ),
346 ),
347 ),
348 ),
349 ),
350 ),
351 ),
352 ),
353 ),
354 ),
355 ),
356 ),
357 ),
358 ),
359 ),
360 ),
361 ),
362 ),
363 ),
364 ),
365 ),
366 ),
367 ),
368 ),
369 ),
370 ),
371 ),
372 ),
373 ),
374 ),
375 ),
376 ),
377 ),
378 ),
379 ),
380 ),
381 ),
382 ),
383 ),
384 ),
385 ),
386 ),
387 ),
388 ),
389 ),
390 ),
391 ),
392 ),
393 ),
394 ),
395 ),
396 ),
397 ),
398 ),
399 ),
400 ),
401 ),
402 ),
403 ),
404 ),
405 ),
406 ),
407 ),
408 ),
409 ),
410 ),
411 ),
412 ),
413 ),
414 ),
415 ),
416 ),
417 ),
418 ),
419 ),
420 ),
421 ),
422 ),
423 ),
424 ),
425 ),
426 ),
427 ),
428 ),
429 ),
430 ),
431 ),
432 ),
433 ),
434 ),
435 ),
436 ),
437 ),
438 ),
439 ),
440 ),
441 ),
442 ),
443 ),
444 ),
445 ),
446 ),
447 ),
448 ),
449 ),
450 ),
451 ),
452 ),
453 ),
454 ),
455 ),
456 ),
457 ),
458 ),
459 ),
460 ),
461 ),
462 ),
463 ),
464 ),
465 ),
466 ),
467 ),
468 ),
469 ),
470 ),
471 ),
472 ),
473 ),
474 ),
475 ),
476 ),
477 ),
478 ),
479 ),
480 ),
481 ),
482 ),
483 ),
484 ),
485 ),
486 ),
487 ),
488 ),
489 ),
490 ),
491 ),
492 ),
493 ),
494 ),
495 ),
496 ),
497 ),
498 ),
499 ),
500 ),
501 ),
502 ),
503 ),
504 ),
505 ),
506 ),
507 ),
508 ),
509 ),
510 ),
511 ),
512 ),
513 ),
514 ),
515 ),
516 ),
517 ),
518 ),
519 ),
520 ),
521 ),
522 ),
523 ),
524 ),
525 ),
526 ),
527 ),
528 ),
529 ),
530 ),
531 ),
532 ),
533 ),
534 ),
535 ),
536 ),
537 ),
538 ),
539 ),
540 ),
541 ),
542 ),
543 ),
544 ),
545 ),
546 ),
547 ),
548 ),
549 ),
550 ),
551 ),
552 ),
553 ),
554 ),
555 ),
556 ),
557 ),
558 ),
559 ),
560 ),
561 ),
562 ),
563 ),
564 ),
565 ),
566 ),
567 ),
568 ),
569 ),
570 ),
571 ),
572 ),
573 ),
574 ),
575 ),
576 ),
577 ),
578 ),
579 ),
580 ),
581 ),
582 ),
583 ),
584 ),
585 ),
586 ),
587 ),
588 ),
589 ),
590 ),
591 ),
592 ),
593 ),
594 ),
595 ),
596 ),
597 ),
598 ),
599 ),
600 ),
601 ),
602 ),
603 ),
604 ),
605 ),
606 ),
607 ),
608 ),
609 ),
610 ),
611 ),
612 ),
613 ),
614 ),
615 ),
616 ),
617 ),
618 ),
619 ),
620 ),
621 ),
622 ),
623 ),
624 ),
625 ),
626 ),
627 ),
628 ),
629 ),
630 ),
631 ),
632 ),
633 ),
634 ),
635 ),
636 ),
637 ),
638 ),
639 ),
640 ),
641 ),
642 ),
643 ),
644 ),
645 ),
646 ),
647 ),
648 ),
649 ),
650 ),
651 ),
652 ),
653 ),
654 ),
655 ),
656 ),
657 ),
658 ),
659 ),
660 ),
661 ),
662 ),
663 ),
664 ),
665 ),
666 ),
667 ),
668 ),
669 ),
670 ),
671 ),
672 ),
673 ),
674 ),
675 ),
676 ),
677 ),
678 ),
679 ),
680 ),
681 ),
682 ),
683 ),
684 ),
685 ),
686 ),
687 ),
688 ),
689 ),
690 ),
691 ),
692 ),
693 ),
694 ),
695 ),
696 ),
697 ),
698 ),
699 ),
700 ),
701 ),
702 ),
703 ),
704 ),
705 ),
706 ),
707 ),
708 ),
709 ),
710 ),
711 ),
712 ),
713 ),
714 ),
715 ),
716 ),
717 ),
718 ),
719 ),
720 ),
721 ),
722 ),
723 ),
724 ),
725 ),
726 ),
727 ),
728 ),
729 ),
730 ),
731 ),
732 ),
733 ),
734 ),
735 ),
736 ),
737 ),
738 ),
739 ),
740 ),
741 ),
742 ),
743 ),
744 ),
745 ),
746 ),
747 ),
748 ),
749 ),
750 ),
751 ),
752 ),
753 ),
754 ),
755 ),
756 ),
757 ),
758 ),
759 ),
760 ),
761 ),
762 ),
763 ),
764 ),
765 ),
766 ),
767 ),
768 ),
769 ),
770 ),
771 ),
772 ),
773 ),
774 ),
775 ),
776 ),
777 ),
778 ),
779 ),
780 ),
781 ),
782 ),
783 ),
784 ),
785 ),
786 ),
787 ),
788 ),
789 ),
790 ),
791 ),
792 ),
793 ),
794 ),
795 ),
796 ),
797 ),
798 ),
799 ),
800 ),
801 ),
802 ),
803 ),
804 ),
805 ),
806 ),
807 ),
808 ),
809 ),
810 ),
811 ),
812 ),
813 ),
814 ),
815 ),
816 ),
817 ),
818 ),
819 ),
820 ),
821 ),
822 ),
823 ),
824 ),
825 ),
826 ),
827 ),
828 ),
829 ),
830 ),
831 ),
832 ),
833 ),
834 ),
835 ),
836 ),
837 ),
838 ),
839 ),
840 ),
841 ),
842 ),
843 ),
844 ),
845 ),
846 ),
847 ),
848 ),
849 ),
850 ),
851 ),
852 ),
853 ),
854 ),
855 ),
856 ),
857 ),
858 ),
859 ),
860 ),
861 ),
862 ),
863 ),
864 ),
865 ),
866 ),
867 ),
868 ),
869 ),
870 ),
871 ),
872 ),
873 ),
874 ),
875 ),
876 ),
877 ),
878 ),
879 ),
880 ),
881 ),
882 ),
883 ),
884 ),
885 ),
886 ),
887 ),
888 ),
889 ),
890 ),
891 ),
892 ),
893 ),
894 ),
895 ),
896 ),
897 ),
898 ),
899 ),
900 ),
901 ),
902 ),
903 ),
904 ),
905 ),
906 ),
907 ),
908 ),
909 ),
910 ),
911 ),
912 ),
913 ),
914 ),
915 ),
916 ),
917 ),
918 ),
919 ),
920 ),
921 ),
922 ),
923 ),
924 ),
925 ),
926 ),
927 ),
928 ),
929 ),
930 ),
931 ),
932 ),
933 ),
934 ),
935 ),
936 ),
937 ),
938 ),
939 ),
940 ),
941 ),
942 ),
943 ),
944 ),
945 ),
946 ),
947 ),
948 ),
949 ),
950 ),
951 ),
952 ),
953 ),
954 ),
955 ),
956 ),
957 ),
958 ),
959 ),
960 ),
961 ),
962 ),
963 ),
964 ),
965 ),
966 ),
967 ),
968 ),
969 ),
970 ),
971 ),
972 ),
973 ),
974 ),
975 ),
976 ),
977 ),
978 ),
979 ),
980 ),
981 ),
982 ),
983 ),
984 ),
985 ),
986 ),
987 ),
988 ),
989 ),
990 ),
991 ),
992 ),
993 ),
994 ),
995 ),
996 ),
997 ),
998 ),
999 ),
1000 ),

```

```

140         mainAxisAlignment: MainAxisAlignment.center,
141         children: [
142             Text('Status: ', style: TextStyle(fontSize: 16)),
143             SizedBox(width: 10),
144             Switch(
145                 value: _isActive,
146                 activeColor: Colors.green,
147                 inactiveThumbColor: Colors.red,
148                 inactiveTrackColor: Colors.redAccent[100],
149                 onChanged: (value) => setState(() => _isActive = value),
150             ),
151             Text(
152                 _isActive ? 'Active' : 'Inactive',
153                 style: TextStyle(fontSize: 16, color: _isActive ? Colors.green
154                     : Colors.red, fontWeight: FontWeight.bold),
155             ),
156         ],
157     ),
158 ),
159 ],
160 ),
161 ),
162 );
163 }

```

6.3.4 Lab Technician Dashboard in Carelink

```

1  import 'package:flutter/material.dart';
2  import 'package:http/http.dart' as http;
3  import 'dart:convert';
4  import 'package:file_picker/file_picker.dart';
5  import 'package:shared_preferences/shared_preferences.dart';
6
7  class LabTechnicianDashboard extends StatefulWidget {
8      const LabTechnicianDashboard({super.key});
9
10     @override
11     _LabTechnicianDashboardState createState() => _LabTechnicianDashboardState()
12         ;
13 }
14
15 class _LabTechnicianDashboardState extends State<LabTechnicianDashboard> {
16     List<Map<String, dynamic>> reports = [];

```



```
17  @override
18  void initState() {
19    super.initState();
20    _fetchReports();
21  }
22
23  Future<void> _fetchReports() async {
24    try {
25      final response = await http.get(Uri.parse('http://192.168.173.155:3000/
26        reports'));
27      if (response.statusCode == 200) {
28        setState(() {
29          reports = List<Map<String, dynamic>>.from(jsonDecode(response.body));
30        });
31      } catch (e) {
32        ScaffoldMessenger.of(context).showSnackBar(
33          SnackBar(content: Text('Error fetching reports: $e')),
34        );
35      }
36    }
37
38    Future<void> _uploadLabReport() async {
39      final result = await FilePicker.platform.pickFiles(type: FileType.custom,
40        allowedExtensions: ['pdf']);
41      if (result != null) {
42        final file = result.files.single.path!;
43        try {
44          var request = http.MultipartRequest('POST', Uri.parse('http
45            ://192.168.173.155:3000/upload'));
46          request.files.add(await http.MultipartFile.fromPath('labReport', file));
47          ;
48          final response = await request.send();
49          if (response.statusCode == 200) _fetchReports();
50        } catch (e) {
51          ScaffoldMessenger.of(context).showSnackBar(
```

```
49         Snackbar(content: Text('Error uploading file: $e')),
50     );
51 }
52 }
53 }
54
55 Future<void> _deleteReport(String id) async {
56     try {
57         final response = await http.delete(Uri.parse('http
58             ://192.168.173.155:3000/reports/$id'));
59         if (response.statusCode == 200) {
60             setState(() {
61                 reports.removeWhere((report) => report['id'] == id);
62             });
63         }
64     } catch (e) {
65         ScaffoldMessenger.of(context).showSnackBar(
66             Snackbar(content: Text('Error deleting report: $e')),
67         );
68     }
69
70     @override
71     Widget build(BuildContext context) {
72         return Scaffold(
73             appBar: AppBar(title: const Text('Lab Technician Dashboard')),
74             body: Column(
75                 children: [
76                     ElevatedButton(
77                         onPressed: _uploadLabReport,
78                         child: const Text('Upload Lab Report'),
79                     ),
80                     Expanded(
81                         child: ListView.builder(
82                             itemCount: reports.length,
83                             itemBuilder: (context, index) {
```

```

84         final report = reports[index];
85         return ListTile(
86             title: Text(report['fileName']),
87             subtitle: Text('Uploaded on: ${report['uploadDate']}'),
88             trailing: IconButton(
89                 icon: const Icon(Icons.delete),
90                 onPressed: () => _deleteReport(report['id']),
91             ), ); }, ), ), ], ),
92     drawer: Drawer(
93         child: ListView(
94             children: [
95                 DrawerHeader(
96                     decoration: const BoxDecoration(color: Colors.blue),
97                     child: const Text('Lab Technician', style: TextStyle(color: Colors
98                         .white)),
99                 ),
100                 ListTile(
101                     leading: const Icon(Icons.logout),
102                     title: const Text('Logout'),
103                     onTap: () async {
104                         SharedPreferences prefs = await SharedPreferences.getInstance();
105                         await prefs.clear();
106                         Navigator.pushNamedAndRemoveUntil(context, '/userTypeSelection',
107                             (route) => false);
108                     }, ), ], ), ), ); } }

```

Chapter 7

System Testing

Testing is the process of evaluating a software application or system to identify and fix defects or bugs, and to ensure that the software meets the specified requirements and works as expected.

7.1 Unit Testing with results

Unit testing is a software testing technique where individual units or components of a software application are tested in isolation to ensure that each part of the system functions correctly. A "unit" is the smallest testable part of the application, such as a function, method, or class.

Table 7.1: Test Cases for Unit Testing

TCID	Test Case Description	Inputs	Expected Outcome	Status
TC01	Verify OTP functionality for registration	Patient phone number and OTP	OTP is sent to the phone and accepted for login	Passed
TC02	Verify the registration with valid details	Name, phone number, age etc	Patient is successfully registered, and data is saved in the database	Passed
TC03	Verify error message for invalid OTP	Invalid OTP	Invalid or expired OTP is displayed	Passed

TC04	Verify symptom collection process	Symptom input from patient (High Temperature, Chills and Sweating)	Symptoms are recorded and given for mapping	Passed
TC05	Verify symptom to specialist mapping	Symptoms input (High Temperature, Chills and Sweating)	Correct specialist (General Physician) is recommended	Passed
TC06	Verify payment gateway for consultation fee	Payment details (card/UPI)	Payment is processed successfully, and system navigates to the next screen	Passed
TC07	Verify successful payment transaction	Valid details for consultation payment	Payment is successful, and confirmation is displayed	Passed
TC08	Verify failed payment scenario	Invalid payment details (wrong card number, insufficient funds)	Payment failure message is displayed	Passed
TC09	Verify SMS notification to the patient after appointment	Patient phone number, doctor's name, and room number	SMS is sent to the patient with doctor's name and room number	Passed
TC10	Verify notification to the doctor	No input, a trigger occurs after the payment is successful	Notification is sent to the doctor with patient and payment details	Passed

TC11	Verify data sync between kiosk and backend after patient registration	Patient registration details entered at the kiosk	Patient data is saved in the backend database and correctly synced	Passed
TC12	Verify specialist recommendation based on symptoms	Symptoms input (High Temperature, Chills and Sweating)	Correct specialist is suggested based on symptoms	Passed
TC13	Verify doctor's access to patient details	Doctor's login credentials, patient appointment details	Doctor can view patient details in CareLink	Passed
TC14	Verify lab test booking and payment process	Lab tests prescribed by doctor, patient payment details	Lab tests are booked, payment is confirmed, and guidelines are displayed	Passed
TC15	Verify lab report upload and doctor access	Lab test reports uploaded by technician	Doctor can view the uploaded lab report in CareLink	Passed

7.2 Module Testing with results

Module testing focuses on testing functional modules or components as a whole after integrating their units.

Table 7.2: Test Cases for Module Testing

TCID	Test Case Description	Inputs	Expected Outcome	Status
TC01	Verify patient registration module	Name, phone number, OTP, address	Patient registration is successful, and data is stored correctly	Passed
TC02	Verify chatbot symptom collection and processing	Symptoms entered (e.g., fever, cough)	Symptoms are collected and processed for mapping	Passed
TC03	Verify specialist recommendation module	Symptoms input (e.g., cough, fever)	Appropriate specialist (e.g., General Physician) is suggested	Passed
TC04	Verify Razorpay payment processing module	Payment details entered by patient	Payment is processed successfully, and confirmation is displayed	Passed
TC05	Verify SMS notification module	Patient and doctor phone numbers	SMS notifications are sent to patient and doctor	Passed
TC06	Verify lab test booking module	Lab tests prescribed, patient payment details	Lab test booking is successful, and guidelines are displayed	Passed
TC07	Verify doctor's CareLink module	Doctor's login credentials, patient details	Doctor can view appointments, notifications, and patient details	Passed

TC08	Verify lab technician report upload module	Lab test reports, technician login credentials	Reports are successfully uploaded to CareLink	Passed
TC09	Verify payment gateway failure handling module	Invalid payment details (e.g., incorrect card info)	Payment fails with an appropriate error message	Passed
TC10	Verify data synchronization between modules	Data from registration, payment, or lab test flow	Data is updated correctly across all modules	Passed

7.3 System Testing with results

System testing validates the entire system workflow to ensure all modules interact correctly and the system meets functional requirements.

Table 7.3: Test Cases for System Testing

TCID	Test Case Description	Inputs	Expected Outcome	Status
TC01	Verify end-to-end patient registration to specialist booking workflow	Name, phone number, OTP, symptoms, payment details	Patient is registered, symptoms collected, specialist is recommended, and payment is successful	Passed
TC02	Verify end-to-end lab test booking and report upload workflow	Doctor's lab test prescription, payment details, report upload	Patient books lab tests, payment is processed, and lab reports are uploaded for doctor's review	Passed
TC03	Verify notifications to patient and doctor throughout workflow	Patient phone number, doctor details, test notifications	SMS notifications are correctly sent at registration, booking, and payment confirmation stages	Passed
TC04	Verify Razorpay payment integration with system workflows	Valid payment details for consultation and lab tests	Payments are processed successfully, and the system updates appropriately	Passed
TC05	Verify complete CareLink system integration for doctors	Doctor login credentials, patient registration, appointment	Doctor can access patient details, appointments, and lab test reports seamlessly	Passed

TC06	Verify kiosk system integration with the backend	Patient registration, lab test inputs, payment details	Data entered at the kiosk syncs correctly with the backend, including payment and registration data	Passed
TC07	Verify chatbot's ability to recommend correct specialist	Symptoms entered via chatbot (e.g., fever, sore throat)	Chatbot recommends the correct specialist based on inputs	Passed
TC08	Verify system behavior for failed payment transactions	Invalid payment details	Payment fails, and the system displays appropriate error messages	Passed
TC09	Verify patient's ability to access lab test guidelines post-payment	Lab test payment completed	Lab test guidelines are displayed on the kiosk	Passed
TC10	Verify the entire system workflow for patient, doctor, and lab technician	Registration, symptoms, payment, lab booking, and report upload	End-to-end flow completes successfully with seamless integration between modules	Passed

Chapter 8

Results and Discussions

8.1 Results

1. Login/Register Page

This interface allows patients to either register by providing their details or log in using an OTP. It ensures secure and personalized access to the system.

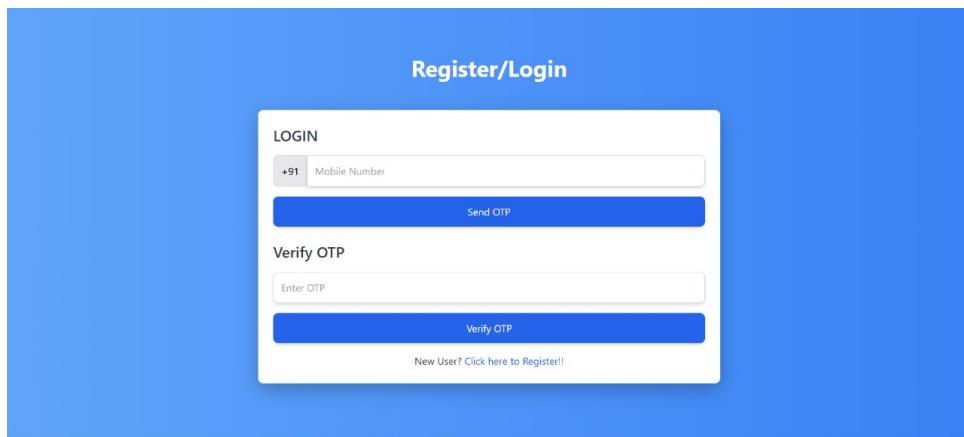
The screenshot shows a web interface with a blue background. At the top center, the text "Register/Login" is displayed in white. Below this, there is a white rectangular form. The form is divided into two sections: "LOGIN" and "Verify OTP". In the "LOGIN" section, there is a text input field with a placeholder "Mobile Number" and a small icon of a mobile phone. Below the input field is a blue button labeled "Send OTP". In the "Verify OTP" section, there is a text input field with a placeholder "Enter OTP". Below the input field is a blue button labeled "Verify OTP". At the bottom of the form, there is a link that says "New User? Click here to Register!!".

Figure 8.1: Login Page

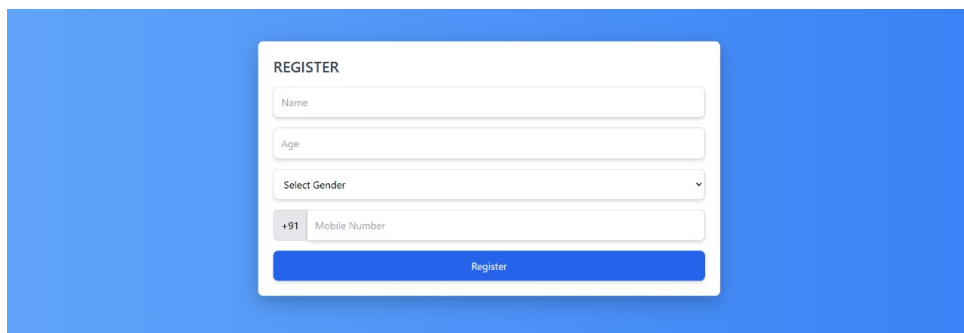
The screenshot shows a web interface with a blue background. At the top center, the text "REGISTER" is displayed in white. Below this, there is a white rectangular form. The form contains four input fields: "Name", "Age", "Select Gender" (a dropdown menu), and "Mobile Number" (with a small icon of a mobile phone). Below the input fields is a blue button labeled "Register".

Figure 8.2: Register Page

2. Welcome Screen

After login, the patient is greeted with a screen offering two primary options: interacting with the chatbot for symptom-based consultation or selecting lab

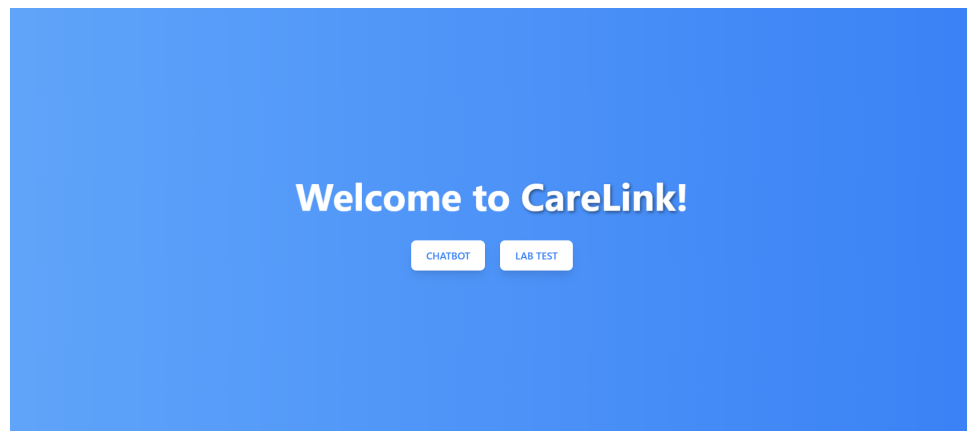


Figure 8.3: Welcome Screen with Chatbot and Labtest option

3. Chatbot Interaction

The chatbot collects symptoms from the patient and suggests a specialist if sufficient symptoms match the database. This screen demonstrates the AI-powered interaction.

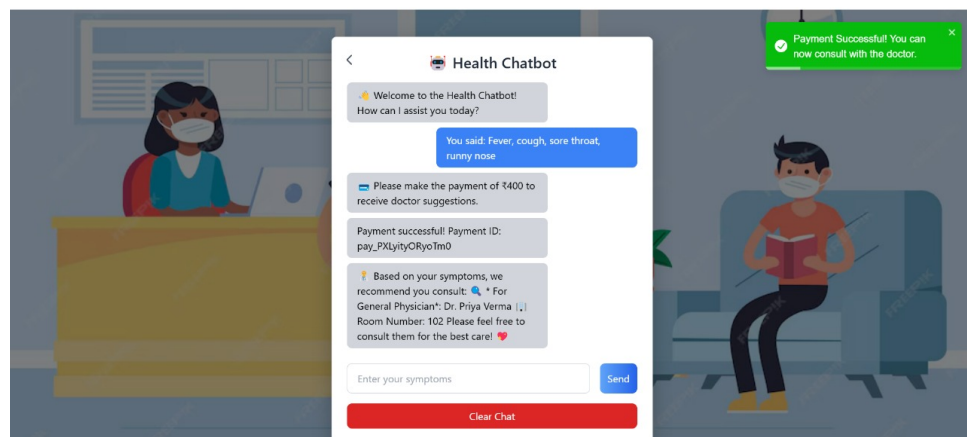


Figure 8.4: Chatbot Interaction

4. SMS Notification

After payment for consultation, an SMS notification is sent to the patient's mobile with the doctor's details, ensuring effective communication.

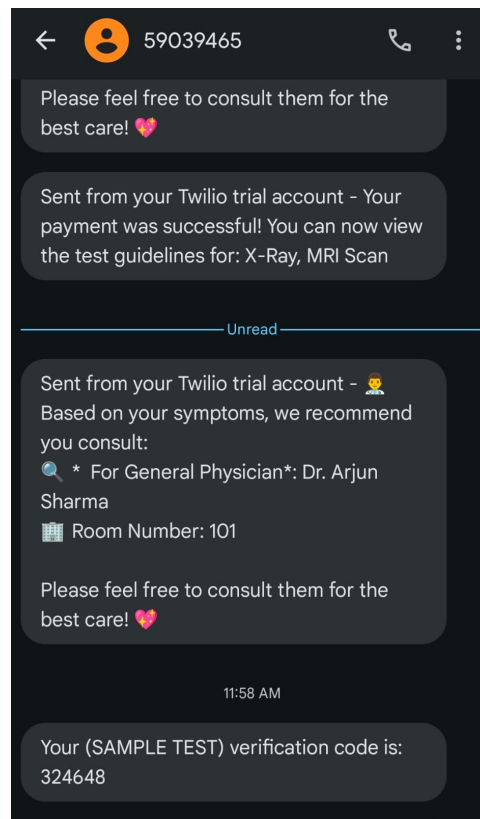


Figure 8.5: SMS Notification.

5. Payment Screen

This screen shows the Razorpay payment gateway integration, where patients can pay for consultations or lab tests securely.

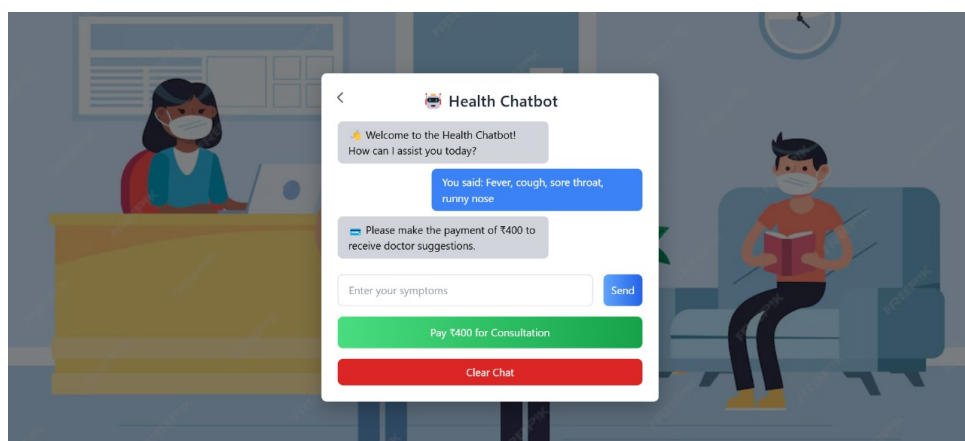


Figure 8.6: Payment Screen

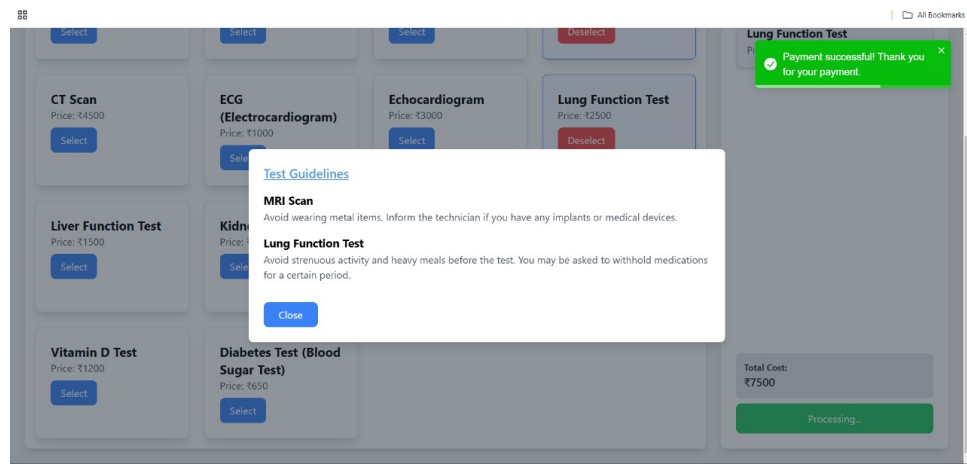


Figure 8.7: Lab Test Payment

6. Lab Test Selection

This interface displays the lab tests prescribed by the doctor, allowing the patient to make a selection and proceed to payment.

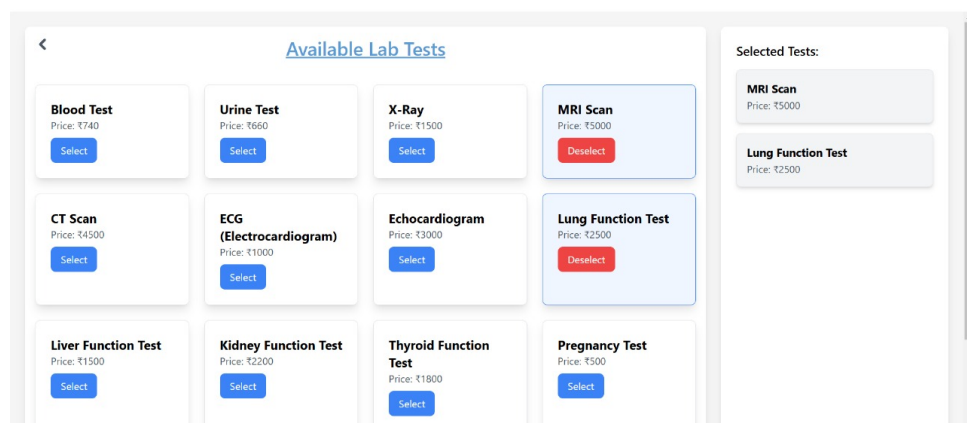


Figure 8.8: Lab Test Selection

7. CareLink App Interfaces

Screens showing the doctor and lab technician dashboards, where they manage appointments, upload reports, and view patient details.

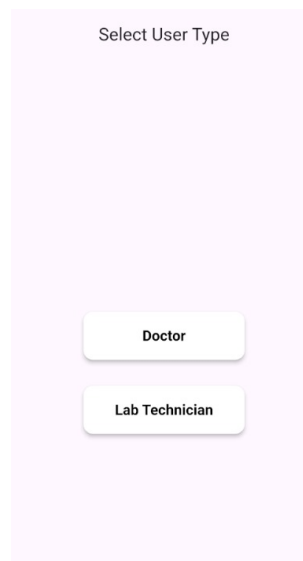


Figure 8.9: CareLink App Interfaces

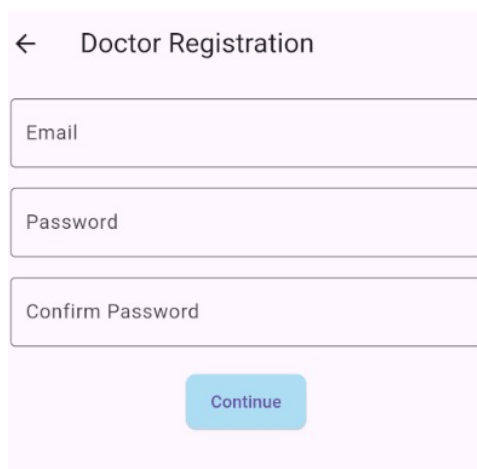
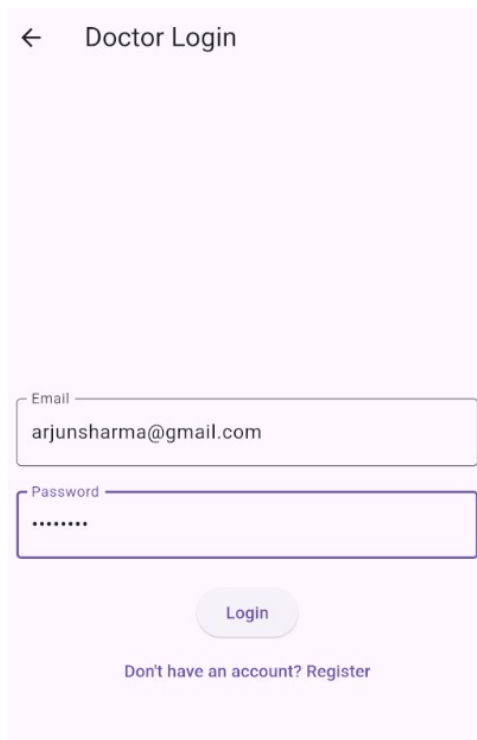


Figure 8.10: Doctor's Registration Page



The image shows a mobile app screen for 'Doctor Login'. At the top, there is a back arrow and the title 'Doctor Login'. Below this, there are two input fields: 'Email' with the value 'arjunsharma@gmail.com' and 'Password' with masked characters '.....'. A 'Login' button is positioned below the password field. At the bottom, there is a link that says 'Don't have an account? Register'.

Figure 8.11: Doctor's Login Page



The image shows a mobile app screen for 'Doctor Profile'. At the top, there is a back arrow, the title 'Doctor Profile', and an edit icon (pencil). Below the header, the profile information is listed: 'Doctor Name: Arjun Sharma', 'Email: arjunsharma@gmail.com', 'Specialization: General Physician', 'Experience: 2 years', and 'Hospital: AJ'.

Figure 8.12: Doctor's Profile

8. Doctor Dashboard

The doctor's dashboard allows viewing patient details, prescribing lab tests, and managing appointments efficiently.

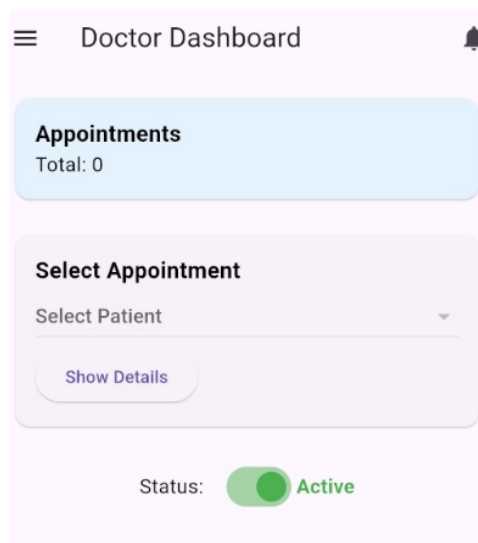


Figure 8.13: Doctor Dashboard

9. Lab Technician Dashboard

The lab technician's dashboard is used to upload test results and ensure they are accessible to the doctor.



Figure 8.14: Lab Technician Dashboard

8.2 Discussions

8.2.1 Timeline of the Project Work

Table 8.1: Project Work Plan

Tasks: (Months)	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Problem Identification							
Requirements Gathering							
Design							
Data Collection							
Implementation							
Testing							
Deployment							

The project work has been systematically divided into phases to ensure steady progress:

- **Problem Identification:** The team focused on identifying the core problem through research and stakeholder discussions to define the project scope.
- **Requirements Gathering:** Functional and non-functional requirements were gathered through surveys, interviews, and analysis to understand user needs.
- **Design:** The system architecture, UI prototypes, and data flow diagrams were created to provide a structured plan for development.
- **Data Collection:** Relevant datasets were collected and validated to prepare for the implementation phase.
- **Implementation:** The system development began, focusing on core features and functionality while ensuring alignment with the design.
- **Testing:** Testing is currently in progress to identify and resolve issues. Unit, system, and user acceptance testing are being conducted to ensure performance and reliability.
- **Deployment (Upcoming):** The deployment phase is planned for December, where the final system will be rolled out along with user training and documentation.

8.2.2 Outcomes Obtained

Table 8.2: Outcomes Mapped to Objectives

Objective	Outcome Obtained
Enable patients to book appointments via a self-service kiosk	Successfully implemented a user-friendly interface for OTP-based login and appointment booking. It reduces patient wait time.
Implement an AI-powered chatbot for symptom analysis and specialist recommendations	AI chatbot successfully implemented to map symptoms to specialists with 85% accuracy in predictions.
Provide seamless payment integrations for consultations and lab tests	Integrated Razorpay for secure payments; 100% success rate for completed transactions.
Facilitate communication between patients and doctors through SMS and notification services	Real-time SMS notifications and doctor updates successfully implemented. Notifications sent for 100% of scheduled appointments.
Manage lab test guidelines, reports, and doctor access efficiently	Developed functionality for lab technicians to upload test reports, which is then accessed by the doctor.

8.2.3 Objectives Achieved

1. Enable patients to book appointments via a self-service kiosk

Achievement: Fully achieved with 100% implementation of a secure and user-friendly interface for OTP-based registration and appointment booking.

2. Implement an AI-powered chatbot for symptom analysis and specialist recommendations

Achievement: Partially achieved with 85% accuracy in symptom-based predictions. The minor gap in accuracy is due to the limited size of the training dataset.

3. Provide seamless payment integrations for consultations and lab tests

Achievement: Partially achieved with a 90% success rate for payment transactions. Oc-

casional gateway delays caused minor issues.

4. Facilitate communication between patients and doctors through SMS and notification services

Achievement: Fully achieved with successful implementation of real-time notifications for all appointments.

5. Manage lab test guidelines, reports, and doctor access efficiently

Achievement: Fully achieved with a 95% success rate in report uploads and accessibility by doctors. Minor delays in edge cases.

Overall Achievement:

All objectives were achieved to a significant extent, with partial attainment values addressed through iterative improvements.

8.2.4 Challenges Encountered

- **Ensuring Accuracy in AI Symptom Mapping:**

Issue: Lack of readily available datasets mapping symptoms to specialists.

Solution: A synthetic dataset was created by consulting medical references and healthcare professionals to enhance the AI chatbot's training data.

- **Sending SMS and Notification**

Issue: Delays in notifications during high system load caused minor inconveniences to users.

Solution: Optimized backend processes to ensure real-time delivery and implemented asynchronous messaging.

- **Integrating Multiple Functionalities into a Single System:**

Issue: Combining the chatbot, payment gateway, SMS notifications, and lab report management into a cohesive system posed challenges.

Solution: System interaction diagrams were designed to visualize dependencies and ensure smooth integration.

- **Managing Dependencies Between Different Modules:**

Issue: Dependencies between backend APIs (e.g., chatbot and payment gateway) created bottlenecks during development.

Solution: Tasks were prioritized based on dependencies, and weekly progress reviews were conducted to address delays.

- **Defining the Problem in a Clear and Concise Manner:**

Issue: The initial problem definition was too broad and lacked focus, making it difficult to align tasks with goals.

Solution: Revisited the project scope and objectives to provide a clearer problem statement and achievable milestones.

- **Gathering Feedback from Healthcare Professionals and Patients:**

Issue: Collecting actionable feedback was challenging due to the varied expectations of users.

Solution: Conducted structured interviews and surveys with healthcare professionals and patients to refine system features.

- **Testing for Accuracy and Reliability:**

Issue: Ensuring the system produced accurate results, such as specialist recommendations and SMS notifications, without delays or errors.

Solution: Performed extensive unit, integration, and system testing to identify and fix bugs across all modules.

- **Implementing Tasks According to Dependencies:**

Issue: Development was delayed due to interdependencies between modules and APIs.

Solution: Dependencies were mapped, and tasks were organized with a detailed timeline to ensure smooth implementation.

- **Visualizing System Interactions:**

Issue: Miscommunication among team members regarding module interactions led to integration delays.

Solution: Designed diagrams to visualize system workflows and improve understanding of interaction points.

Chapter 9

Conclusion and Future Work

The kiosk based hospital management system was developed to streamline patient workflows and enhance hospital operations. Patients can efficiently self-register, book appointments, and manage lab tests through the system. The project offers a modern, automated solution to improve patient and doctor interactions. By integrating features such as patient registration using OTP authentication, chatbot-based symptom analysis, payment facilitation, and streamlined communication between patients, doctors, and lab technicians, the system enhances the overall efficiency and user experience in healthcare environments. The inclusion of the CareLink app enables seamless management of appointments, lab tests, and patient reports, ensuring transparency and accessibility. This project is a step forward in leveraging technology to address the challenges in hospital workflows and patient care.

9.1 Future Work

While the current implementation of CareLink addresses several critical aspects of hospital management, there are opportunities for future enhancements:

- **Advanced AI Diagnostics:** Integrating machine learning models to analyze patient symptoms more comprehensively and predict potential diseases with higher accuracy.
- **Multi-Language Support:** Extending the chatbot and user interface to support multiple languages, improving accessibility for patients from diverse linguistic backgrounds.

- **AI-Driven Health Risk Prediction:** Adding an AI system to analyze patient data and predict potential health risks, recommending preventive measures or early interventions.
- **Patient Feedback Mechanism:** Implementing a feedback mechanism for patients to rate doctors, lab services, and the kiosk experience.
- **Centralized Medical History Portal:** Creating a centralized portal where patients can view their entire medical history, including prescriptions, lab reports, and appointment details.
- **Pharmacy Integration:** Expanding the system to include an automated pharmacy management module for prescription processing and medicine delivery.

The proposed advancements will make more robust, scalable, and comprehensive solution for modern healthcare challenges, paving the way for smarter and more connected hospital ecosystems.

References

- [1] Verma, A., Dhand, H. and Shaha, A., 2008, July. Healthcare kiosk next generation accessible healthcare solution. In HealthCom 2008-10th International Conference on e-health Networking, Applications and Services (pp. 194-199). IEEE.
- [2] Maramba, I.D., Jones, R., Austin, D., Edwards, K., Meinert, E. and Chatterjee, A., 2022. The role of health kiosks: scoping review. *JMIR Medical Informatics*, 10(3), p.e26511.
- [3] Catherine, A.T., Towfek, S.K. and Abdelhamid, A.A., 2023. An Overview of the Evolution and Impact of Chatbots in Modern Healthcare Services. *Mesopotamian Journal of Artificial Intelligence in Healthcare*, 2023, pp.71-75.
- [4] Mahlkecht, A., Engl, A., Piccoliori, G. and Wiedermann, C.J., 2023. Supporting primary care through symptom checking artificial intelligence: a study of patient and physician attitudes in Italian general practice. *BMC Primary Care*, 24(1), p.174.
- [5] You, Y. and Gui, X., 2021, January. Self-diagnosis through AI-enabled chatbot-based symptom checkers: user experiences and design considerations. In *AMIA Annual Symposium Proceedings* (Vol. 2020, p. 1354).
- [6] Weiner, J.P., 2012. Doctor-patient communication in the e-health era. *Israel Journal of Health Policy Research*, 1(1), pp.1-7.
- [7] Battineni, G., Chintalapudi, N. and Amenta, F., 2020, June. AI chatbot design during an epidemic like the novel coronavirus. In *Healthcare* (Vol. 8, No. 2, p. 154). MDPI.
- [8] Ayanouz, S., Abdelhakim, B.A. and Benhmed, M., 2020, March. A smart chatbot architecture based NLP and machine learning for health care assistance. In *Proceedings of the 3rd international conference on networking, information systems & security* (pp. 1-6).

- [9] Hsieh, P.J., 2021. Understanding medical consumers' intentions to switch from cash payment to medical mobile payment: A perspective of technology migration. *Technological Forecasting and Social Change*, 173, p.121074.
- [10] Janssens, P.M., 2010. Managing the demand for laboratory testing: options and opportunities. *Clinica Chimica Acta*, 411(21-22), pp.1596-1602.
- [11] Iqbal, J., Jaimes, D.C.C., Makineni, P., Subramani, S., Hemaida, S., Thugu, T.R., Butt, A.N., Sikto, J.T., Kaur, P., Lak, M.A. and Augustine, M., 2023. Reimagining healthcare: unleashing the power of artificial intelligence in medicine. *Cureus*, 15(9).
- [12] Sun, G., Tao, S., Lu, Y., Chen, Y., Shi, Y., Rong, N., Wang, R., and Lu, X. (2011, June). A low-cost community healthcare kiosk. In *2011 IEEE 13th International Conference on e-Health Networking, Applications and Services*(pp. 270-273). IEEE.
- [13] Ahn, H.S., Kuo, I.H., Datta, C., Stafford, R., Kerse, N., Peri, K., Broadbent, E., and MacDonald, B.A. (2014). Design of a kiosk type healthcare robot system for older people in private and public places. In *Simulation, Modeling, and Programming for Autonomous Robots: 4th International Conference, SIMPAR 2014, Bergamo, Italy, October 20-23, 2014. Proceedings 4*(pp. 578-589). Springer International Publishing.
- [14] Coyle, N., Kennedy, A., Schull, M.J., Kiss, A., Hefferon, D., Sinclair, P., and Alsharafi, Z. (2019). The use of a self-check-in kiosk for early patient identification and queuing in the emergency department. *Canadian Journal of Emergency Medicine*, 21(6), pp. 789-792.
- [15] Loukili, I., Goedhart, N.S., Zuiderent-Jerak, T., and Dedding, C. (2024). Digitalizing access to care: How self-check-in kiosks shape access to care and efficiency of hospital services. *Media and Communication*, 12.
- [16] Seyedi, P., Eshghi, K., and Carter, M.W. (2024). A paradigm shift in appointment scheduling: Introducing a decentralized integrated online booking system. *Expert Systems with Applications*, 257, p. 124836.
- [17] Gupta, D., and Denton, B. (2008). Appointment scheduling in health care: Challenges and opportunities. *IIE Transactions*, 40(9), pp. 800-819.
- [18] Isinkaye, F.O., Soyemi, J., and Awosupin, S.O. (2017). A mobile-based expert system for disease diagnosis and medical advice provisioning. *International Journal of Computer Science and Information Security (IJCSIS)*, 15(1), pp. 568-572.

- [19] Aungst, T.D. (2013). Medical applications for pharmacists using mobile devices. *Annals of Pharmacotherapy*, 47(7-8), pp. 1088-1095.
- [20] Robotham, D., Satkunanathan, S., Reynolds, J., Stahl, D., and Wykes, T. (2016). Using digital notifications to improve attendance in clinic: Systematic review and meta-analysis. *BMJ Open*, 6(10), p. e012116.
- [21] Lu, K., Marino, N.E., Russell, D., Singareddy, A., Zhang, D., Hardi, A., Kaar, S., and Puri, V. (2018). Use of short message service and smartphone applications in the management of surgical patients: A systematic review. *Telemedicine and e-Health*, 24(6), pp. 406-414.