“MediConnect- A Telemedicine Platform”

A Major Project Report Submitted to

Rajiv Gandhi Proudyogiki Vishwavidyalaya

Towards Partial Fulfillment for the Award of

Bachelor of Engineering in Computer Science Engineering

Submitted By	***Guided By***
Ishant Mandloi (0827CS211105)	**Prof. Ritika Bhatt**
Jigyansh Sisodiya (0827CS211111)	**Assistant professor, CSE**
Krishna Bhawsar (0827CS211128)	**AITR, Indore**
Krishna Gupta (0827CS211129)	
Kunal Yadav (0827CS211132)	

Acropolis Institute of Technology & Research, Indore

Jul - Dec 2024

EXAMINER APPROVAL

The Major Project entitled ***"MediConnect- A Telemedicine Platform"*** submitted by **Ishant Mandloi (0827CS211105), Jigyansh Sisodiya (0827CS211111), Krishna Bhawsar (0827CS211128), Krishna Gupta (0827CS211129), Kunal Yadav (0827CS211132)** has been examined and is hereby approved towards partial fulfillment for the award of ***Bachelor of Technology degree in Computer Science Engineering*** discipline, for which it has been submitted. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed, or conclusion drawn therein, but approve the project only for the purpose for which it has been submitted.

(Internal Examiner)	**(External Examiner)**
Date:	**Date:**

RECOMMENDATION

This is to certify that the work embodied in this Major project entitled ***"MediConnect- A Telemedicine Platform"*** submitted by **Ishant Mandloi (0827CS211105), Jigyansh Sisodiya (0827CS211111), Krishna Bhawsar (0827CS211128), Krishna Gupta (0827CS211129), Kunal Yadav (0827CS211132)** is a satisfactory account of the bonafide work done under the supervision of ***Prof. Ritika Bhatt***, is recommended towards partial fulfillment for the award of the Bachelor of Technology (Computer Science Engineering) degree by Rajiv Gandhi Proudyogiki Vishwavidhyalaya, Bhopal.

(Project Guide)

(Project Coordinator)

(Dean Academics)

STUDENT UNDERTAKING

This is to certify that the Major project entitled ***"MediConnect- A Telemedicine Platform"*** has been developed by us under the supervision of ***Prof. Ritika Bhatt***. The whole responsibility of the work done in this project is ours. The sole intention of this work is only for practical learning and research.

We further declare that to the best of our knowledge; this report does not contain any part of any work which has been submitted for the award of any degree either in this University or in any other University / Deemed University without proper citation and if the same work is found then we are liable for explanation to this.

Ishant Mandloi

Jigyansh Sisodiya

Krishna Bhawsar

Krishna Gupta

Kunal Yadav

ACKNOWLEDGEMENT

There are a number of people without whom this project would not have been feasible. Their high academic standards and personal integrity provided me with continuous guidance and support.

We owe a debt of sincere gratitude, deep sense of reverence and respect to our guide and mentor **Prof. Ritika Bhatt,** Professor, AITR, Indore for his motivation, sagacious guidance, constant encouragement, vigilant supervision, and valuable critical appreciation throughout this project work, which helped us to successfully complete the project on time.

We express profound gratitude and heartfelt thanks to **Dr. Kamal Kumar Sethi**, Professor & Head CSE, AITR Indore for his support, suggestion, and inspiration for carrying out this project. I am very much thankful to other faculty and staff members of the department for providing me all support, help and advice during the project. We would be failing in our duty if we do not acknowledge the support and guidance received from **Dr. S.C. Sharma**, Director, AITR, Indore whenever needed. We take the opportunity to convey my regards to the management of Acropolis Institute, Indore for extending academic and administrative support and providing me all necessary facilities for the project to achieve our objectives.

We are grateful to **our parents** and **family members** who have always loved and supported us unconditionally. To all of them, we want to say “Thank you”, for being the best family that one could ever have and without whom none of this would have been possible.

Ishant Mandloi

Jigyansh Sisodiya

Krishna Bhawsar

Krishna Gupta

Kunal Yadav

EXECUTIVE SUMMARY

This project is presented in fulfillment of the requirements for an innovative attendance management system, Vidyarthi, designed with a forward-thinking approach, leveraging advanced technologies. It was developed under the dedicated guidance of **Prof. Ritika Bhatt** and submitted to Rajiv Gandhi Proudyogiki Vishwavidhyalaya, Bhopal (MP), India.

The **"MediConnect – A Telemedicine Platform"** project that seeks to develop a user-friendly and efficient healthcare platform. The platform's primary goal is to facilitate easy access to medical care through online appointment scheduling and secure video consultations. In response to the growing demand for convenient healthcare services, this project aims to streamline the appointment booking process, reduce wait times, and bring medical expertise to patients homes through video consultations. The platform will prioritize data security and privacy, implementing robust encryption measures and adhering to stringent compliance standards. Additionally, it will incorporate online payment options, ensuring a seamless financial aspect of healthcare services. A user-friendly interface, scheduling system, automated reminders, and a feedback and rating system will collectively enhance the overall patient and healthcare provider experience. This project strives to contribute to the accessibility and efficiency of healthcare services in the digital age, ultimately benefiting the well-being of the community.

“Where the vision is one year,

cultivate flowers;

Where the vision is ten years,

cultivate trees;

Where the vision is eternity,

cultivate people.” -

Oriental Saying

LIST OF FIGURES

Figure 1: Use Case Diagram…………………………………………………………….. 17

Figure 2: Sequence Diagram…………………………………………………………...18

Figure 3: Activity Diagram…………………………………………………………….. 19

Figure 4: Entity Relationship Diagram………………………………………….....20

Figure 5: [Test Case: 1] Duplicate Email………………………………………..…30

Figure 6: [Test Case: 2] Verify OTP Functionality..……………………………....31

Figure 7: [Test Case: 3] Search Functionality……………………………….......32

Figure 8: [Test Case: 4] Verify Password Mismatch Error…………………....33

Figure 9: [Test Case: 5] Verify "No doctors found"……………..……………...34

Figure 10: [Test Case: 6] User does not exist……………………..…………......35

LIST OF TABLES

Table 1: [Test Case: 1] Duplicate email test.………………………..…….…..30

Table 2: [Test Case: 2] Verify OTP functionality……………………………...31

Table 3: [Test Case: 3] Search functionality……………………………….....32

Table 4: [Test Case: 4] Verify password mismatch……………………..…....33

Table 5: [Test Case: 5] Verify "No doctors found"……………..……………...34

Table 6: [Test Case: 6] User does not exists test……………………………...35

TABLE OF CONTENTS

EXAMINER APPROVAL .... **I**

RECOMMENDATION .... **II**

STUDENT UNDERTAKING .... **III**

ACKNOWLEDGEMENT .... **IV**

EXECUTIVE SUMMARY .... **V**

ORIENTAL SAYING .... **VI**

CHAPTER 1: INTRODUCTION 1

1.1 Overview 2

1.2 Background and Motivation 3

1.3 Problem Statement and Objectives 3

1.4 Scope of the Project 4

1.5 Report Structure 5

CHAPTER 2: REVIEW OF LITERATURE 6

2.1 Preliminary Investigation 7

2.2 Requirement Identification and Analysis for Project 8

2.2.1 Functional Requirement…. 8

2.2.2 Non Functional Requirement 10

2.3 Chapter Summary 10

CHAPTER 3: PROPOSED SYSTEM - MediConnect **12**

3.1 The Proposal 13

3.2 Benefits of the Proposed System 14

3.3 Design Representation 17

3.3.1 Use Case Diagram 17

3.3.2 Sequence Diagram 18

3.3.3 Activity Diagram 19

3.3.4 ER Diagram 20

3.4 Database Structure 20
3.4.1 Patient 20
3.4.2 Doctor 21
3.4.3 Appointment 21
3.5 Deployment Requirements 21
3.5.1 Hardware 21
3.5.2 Software 22
3.5.4 Summary 24

CHAPTER 4: DEVELOPMENT 25
4.1 Techniques Used 26
4.1.1 Agile Development Methodology 26
4.2 Tools Used 26
4.2.1 Integrated Development Environment (IDE)........26
4.2.2 Version Control and Collaboration 27
4.2.3 Front-End Development 27
4.2.4 Back-End Development 27
4.3 Languages Used 27
4.4 Testing 29
4.4.1 Unit Testing 29
4.4.2 System Testing 30

CHAPTER 5: CONCLUSION 36
CHAPTER 6: OPPORTUNITIES AND RECOMMENDATIONS FOR FUTURE RESEARCH 38

BIBLIOGRAPHY........ 42
A. Reference........43
B. Research Paper........ 44
C. Technical Poster 50
PROJECT PLAN........51

GUIDE INTERACTION SHEET……………………………………………………………...52

SOURCE CODE…………………………………………………………………………….53

A. Models.. 54

B. Controllers.. 58

CHAPTER 1

INTRODUCTION

CHAPTER 1: INTRODUCTION

1.1 Overview

In an era characterized by rapid technological advancement and changing societal needs, the healthcare landscape is evolving as well. The project, "Online Video Consultation and Appointment Booking," emerges as a response to the growing demand for healthcare services that are accessible, convenient, and in alignment with the digital age. This initiative aims to address the multifaceted challenges faced by both patients and healthcare providers by introducing a comprehensive platform that revolutionizes the way medical care is sought and delivered.

The traditional model of healthcare, reliant on in-person appointments and physical clinic visits, has often proven cumbersome and inefficient, particularly in scenarios where immediate medical attention is required, or when geographical barriers impede accessibility.

This project endeavors to bridge the gap between patients and healthcare professionals through the implementation of two key components: online appointment booking and video consultations. By introducing a user-friendly interface that allows patients to effortlessly schedule appointments with their chosen healthcare providers, we aim to streamline the process, reduce waiting times, and enhance the overall patient experience.

1.2 Background and Motivation

In recent years, the healthcare industry has witnessed rapid technological advancements, yet patients often face challenges accessing reliable information, managing appointments, and maintaining communication with healthcare providers. Particularly in areas where healthcare resources are limited, patients encounter long waiting times, lack of personalized care, and difficulty coordinating with multiple healthcare providers. Digital solutions that enable seamless patient-provider interaction, consolidate medical records, and provide users with accessible information are increasingly essential.

MediConnect aims to bridge the gap between patients and healthcare providers by providing an integrated platform that enhances communication, simplifies appointment management, and centralizes patient information.

1.3 Problem Statement and Objectives

MediConnect seeks to address these challenges by developing a digital platform that centralizes patient information, simplifies appointment scheduling, and enhances communication between patients and healthcare providers. The platform aims to provide a seamless, user-friendly experience that empowers patients to manage their healthcare needs efficiently while supporting providers in delivering coordinated, high-quality care, the following objectives:

1. **Data Accuracy**: Ensure high levels of accuracy in attendance records, minimizing errors associated with traditional methods.

2. **User Authentication**: Implement secure user authentication protocols to protect sensitive student data.

3. **Scalability**: Design the system to accommodate a growing user base without compromising performance.

1.4 Scope of the Project

The scope of the *MediConnect* project encompasses the development of a comprehensive TeleMedicine platform that incorporates secure data handling, and advanced reporting capabilities. The following key components outline the primary focus areas within the project's scope:

Front-End Development: The project will leverage React.js to create an interactive, responsive, and visually appealing user interface. Key features, such as real-time notifications and a dashboard view of attendance status, will enhance user engagement and allow educators and students to access attendance information conveniently. The front-end will include functionalities like user registration, log-in, attendance viewing, and real-time updates, making it both functional and user-friendly.

Back-End Development: The project's back end will be built on Spring Boot and Hibernate, which provide a stable and scalable foundation for handling user authentication, attendance data processing, and secure communication with the database. This backend architecture will ensure efficient data handling, seamless integration with MySQL, and the flexibility needed to support advanced attendance features.

User Authentication: *MediConnect* incorporates secure user authentication protocols, including password encryption and session management. The system may also explore multi-factor authentication to provide an additional layer of security, ensuring that sensitive user information is adequately protected.

1.5 Report Structure

This report is organized into seven chapters as follows:

Chapter 1: Introduction: Provides an overview of the project, including its background, motivation, problem statement, objectives, and scope, laying the groundwork for the subsequent chapters.

Chapter 2: Review of Literature: Explores existing literature on Telemedicine systems and identifies the limitations of current solutions, helping to contextualize *MediConnect* within the larger landscape of Medical technology.

Chapter 3: Proposed System: Describes the proposed system in detail, outlining its architecture, benefits, block diagram, feasibility, and deployment requirements, and explaining how it addresses the gaps identified in the literature review.

Chapter 4: Development: Discusses the tools, techniques, and programming languages used to develop *MediConnect*. This chapter also elaborates on testing procedures, providing insights into how the system was refined and validated to ensure performance and accuracy.

Chapter 5: Conclusion: Summarizes the key findings and contributions of the project, discusses its limitations, and provides recommendations for future enhancements.

Chapter 6: Opportunities and Recommendations for Future Research: Highlights any constraints encountered during development and suggests potential improvements and additional features that could further enhance MediConnect.

CHAPTER 2

REVIEW OF LITERATURE

CHAPTER 2: REVIEW OF LITERATURE

2.1 Preliminary Investigation

1. **Telemedicine and Digital Health Platforms**
   Studies show that telemedicine and digital health platforms have seen significant growth in recent years, especially with the global increase in demand for remote healthcare solutions. Research by Kvedar et al. (2020) highlights the effectiveness of telemedicine in improving healthcare accessibility, reducing patient wait times, and enabling remote monitoring, particularly beneficial in underserved areas. Digital health platforms are found to enhance patient engagement by providing tools for direct communication with healthcare providers, access to medical records, and the ability to manage appointments online, which MediConnect aims to provide.

2. **Patient-Centric Health Management Systems**
   Patient-centric platforms focus on empowering patients to manage their health by giving them direct access to their health data. According to a study by Miller (2018), such systems reduce medical errors, improve treatment outcomes, and facilitate informed decision-making by providing patients with access to their own health records. MediConnect aligns with this patient-centric approach by enabling users to manage appointments, track prescriptions, and communicate with healthcare providers, creating a more engaged and informed patient experience.

3. **Challenges in Health Information Exchange (HIE)**
   Despite technological advancements, healthcare remains fragmented, with patients often facing challenges in sharing medical records across providers. According to Rumbold et al. (2017), barriers in Health Information Exchange (HIE) result in duplicated efforts, inconsistencies in patient care, and increased costs. Platforms like MediConnect address these issues by serving as a centralized hub for patient data, supporting continuity of care and enhancing information flow across providers.

4. **Appointment Scheduling and Patient Wait Time Reduction**
   A significant challenge in healthcare is the inefficiency in appointment scheduling, often leading to long wait times and patient dissatisfaction. Studies by Huang et al. (2019) indicate that online appointment scheduling systems reduce wait times, optimize healthcare provider availability, and increase patient satisfaction. By incorporating efficient scheduling features, MediConnect helps minimize delays and ensures a smoother healthcare experience.

5. **Data Security and Privacy in Digital Health**
   Protecting patient data is crucial, as healthcare information is highly sensitive. Research by Park et al. (2021) discusses the importance of data privacy and security in healthcare applications, especially with the rise of cyber threats. MediConnect places a high priority on secure data handling and compliance with health privacy regulations, ensuring that users' medical information is safeguarded throughout all interactions.

2.2 Requirement Identification and Analysis for Project

The requirement identification and analysis phase is crucial for defining the goals and functionality of the MediConnect system. This phase builds upon the insights gained from the limitations of traditional and contemporary Telemedicine platforms. By focusing on the needs of doctors and patients, we have identified key requirements that will drive the development of an efficient, user-friendly, and scalable solution.

2.2.1. Functional Requirements

- **User Registration and Authentication**
  - **Requirement**: Users (patients and healthcare providers) must be able to register with the platform and authenticate securely.
  - **Analysis**: A simple, user-friendly registration process should be in place, incorporating secure authentication methods (e.g., two-factor authentication) to protect patient and provider data.

- **Appointment Scheduling and Management**
  - **Requirement**: Patients should be able to view provider availability, schedule, reschedule, and cancel appointments, and receive appointment reminders.
  - **Analysis**: Implement a calendar-based scheduling system integrated with notification services (e.g., SMS/email reminders). It must also handle overlapping appointments, avoid scheduling conflicts, and provide real-time availability.

- **Health Record Management**
  - **Requirement**: Patients and providers should access, update, and share medical records through a centralized system.
  - **Analysis**: Build a secure, cloud-based repository that allows easy access to medical records while ensuring HIPAA compliance. Role-based access control (RBAC) should restrict data access based on user roles to protect sensitive information.

- **Prescription Tracking**
  - **Requirement**: Patients need a system to view and track their prescriptions, including dosage, frequency, and refill dates.
  - **Analysis**: Include prescription details in patient profiles, with notifications for upcoming refills and warnings if medications are discontinued. This feature also allows providers to monitor patient adherence.

- **Patient-Provider Communication**
  - **Requirement**: The platform should facilitate direct communication between patients and providers for follow-ups, queries, and updates.
  - **Analysis**: Enable secure messaging or chat functionality to allow timely communication. Message encryption is essential for privacy, and quick response tracking can enhance engagement.

2.2.2. Non-Functional Requirements

- **Security and Data Privacy**
    - **Requirement**: All data must be protected by encryption, role-based access, and compliance with healthcare regulations.
    - **Analysis**: Implement end-to-end encryption, data anonymization where possible, and secure storage practices to protect patient and provider information. Auditing and monitoring should be in place to detect and respond to security breaches.

- **Scalability**
    - **Requirement**: The system should handle an increasing number of users, especially during peak usage times.
    - **Analysis**: Use scalable cloud infrastructure and microservices architecture to allow the platform to expand as user demand grows. Load balancing and caching mechanisms will help maintain performance as traffic increases.

- **Usability**
    - **Requirement**: The interface should be intuitive and accessible for users of all ages and tech familiarity levels.
    - **Analysis**: Design a clean and simple UI, adhering to UX best practices, to ensure ease of navigation. Accessibility features (like screen reader compatibility) should make the platform usable for individuals with disabilities.

2.3 Chapter Summary

This report introduces MediConnect, a healthcare platform designed to improve patient-provider interactions by offering a centralized solution for scheduling, health records, and communication. Similar to E-Sanjivani and Tata 1MG, MediConnect aims to address challenges such as fragmented health information and limited access to coordinated care. E-Sanjivani struggles with inconsistent regional availability, especially in rural areas, and has a complex interface that is difficult for some users. Tata 1MG focuses on e-commerce, limiting its ability to provide comprehensive healthcare, particularly in remote areas. MediConnect addresses these flaws by offering better regional availability, a more intuitive interface, and a broader range of healthcare services, making it accessible to a wider population.

The literature review highlights the benefits of patient-centric platforms, which improve healthcare accessibility, reduce medical errors, and support better decision-making. However, issues with Health Information Exchange (HIE), like those seen in E-Sanjivani, create inefficiencies and fragmented care. While Tata 1MG excels in e-commerce, it lacks integration with local healthcare systems for holistic care. MediConnect solves these problems by ensuring seamless data exchange, better integration with healthcare providers, and a more comprehensive service offering.

In the requirements analysis, MediConnect's functional needs include user registration, secure health records, appointment scheduling, and messaging features. Non-functional requirements focus on security, scalability, and usability, ensuring the platform complies with regulations like HIPAA. MediConnect addresses the gaps in E-Sanjivani's regional coverage and Tata 1MG's logistical challenges, creating a platform that is accessible, user-friendly, and capable of providing integrated healthcare services across diverse regions.

CHAPTER 3

PROPOSED SYSTEM

-MediConnect

CHAPTER 3: PROPOSED SYSTEM - MediConnect

3.1 The Proposal

The MediConnect platform is designed to enhance patient-provider interactions by providing an integrated digital solution for healthcare management, focused on features like appointment scheduling, health record management, prescription tracking, and secure communication. The system aims to tackle the challenges of healthcare accessibility, data fragmentation, and care continuity, making it simpler for patients to manage their health and for providers to deliver coordinated services efficiently.

Core functionalities of MediConnect include **User Registration and Authentication**, allowing patients and providers to create secure accounts with two-factor authentication to protect sensitive data. The **Appointment Scheduling** feature enables patients to view provider availability, book appointments, and receive automated reminders, reducing no-shows and minimizing wait times. **Health Record Management** provides patients and providers with secure access to medical history, supporting accurate diagnosis and continuity of care. Additionally, **Prescription Tracking** helps patients manage medications with refill alerts, improving adherence to prescribed treatments.

- **Purpose and Scope**

MediConnect is a healthcare platform designed to improve patient-provider interactions and simplify healthcare management.
The platform addresses issues of healthcare accessibility, data fragmentation, and continuity of care.

- **Core Functionalities**

  - **User Registration and Authentication:** Enables secure account creation with two-factor authentication for both patients and providers.
  - **Appointment Scheduling:** Allows patients to view provider availability, book appointments, and receive automated reminders, reducing no-shows.
  - **Health Record Management:** Provides secure access to medical histories, supporting continuity of care and accurate diagnoses.
  - **Prescription Tracking:** Helps patients track medications, with alerts for refills and dosage reminders, supporting adherence.

3.2 Benefits of the Proposed System

Here are the benefits of the proposed MediConnect system for both patients and doctors:

Benefits for Patients:

1. **Convenient Access to Healthcare**: Patients can easily schedule appointments with their preferred healthcare providers, reducing wait times and eliminating the need for phone calls or in-person visits to book appointments. This increases convenience and ensures timely access to care.

2. **Centralized Health Records**: MediConnect enables patients to access their medical history, lab results, prescriptions, and treatment plans from a single

platform. This centralized access empowers patients to track their health progress and share important information with healthcare providers without the hassle of paper records or fragmented systems.

3. **Improved Medication Management**: The prescription tracking feature helps patients manage their medications by providing reminders for refills, dosage schedules, and alerts for medication changes. This improves adherence to prescribed treatments and enhances patient outcomes.

4. **Better Communication with Providers**: Through secure messaging, patients can communicate with their doctors for follow-ups, clarifications, or advice outside of traditional office visits. This improves patient engagement and satisfaction while allowing for quicker resolution of concerns.

5. **Enhanced Convenience and Time-Saving**: With the ability to schedule, manage, and reschedule appointments online, patients save time previously spent on phone calls or waiting in clinics. Automated reminders ensure patients do not forget their appointments or medication schedules.

6. **Increased Control and Empowerment**: MediConnect gives patients greater control over their healthcare management, allowing them to track appointments, prescriptions, and health data in real-time. This empowers patients to take charge of their health, improving decision-making and overall care satisfaction.

Benefits for Doctors:

1. **Streamlined Appointment Scheduling**: Doctors can manage their schedules more efficiently by allowing patients to book appointments directly through the platform. This reduces administrative overhead and eliminates scheduling conflicts. Automated reminders also reduce the number of missed appointments.

2. **Access to Comprehensive Patient Data**: With access to centralized patient records, doctors can view up-to-date medical histories, lab results, and previous treatments, enabling more accurate diagnoses and treatment plans. This enhances decision-making and reduces errors due to incomplete or outdated information.

3. **Reduced Administrative Burden**: Automated features like appointment scheduling, reminders, and prescription tracking reduce the administrative workload for doctors and their staff, allowing them to focus more on patient care rather than administrative tasks.

4. **Improved Patient Follow-up and Engagement**: Doctors can use the secure messaging feature to communicate with patients between visits, providing timely advice, answering questions, and offering guidance. This enhances patient engagement, improves adherence to treatment plans, and leads to better patient outcomes.

5. **Enhanced Collaboration with Healthcare Providers**: Doctors can easily share patient records with other specialists, ensuring continuity of care and reducing the risk of duplicating tests or treatments. This is especially useful for multi-provider healthcare teams or when patients see multiple specialists.

6. **Increased Patient Satisfaction and Loyalty**: The improved efficiency, better communication, and easy access to healthcare resources contribute to a higher level of patient satisfaction. Satisfied patients are more likely to return for future care and recommend the doctor to others.

3.3 Design Representation

3.3.1 Use Case Diagram

The design representation of the Telemedicine system involves visual models and diagrams that illustrate the system's structure, and database organization.

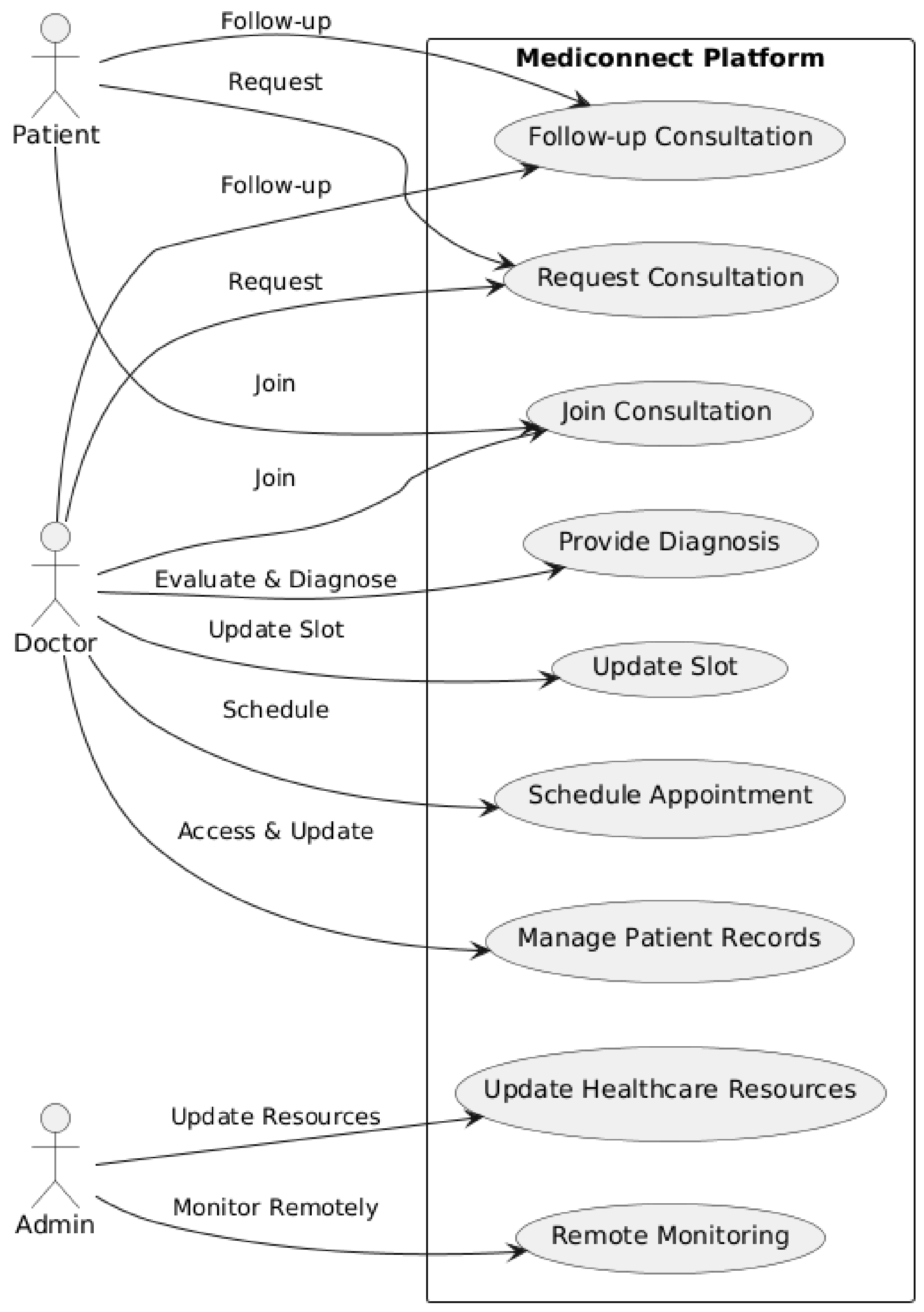


Figure 1. Use Case Diagram

3.3.2 Sequence Diagram

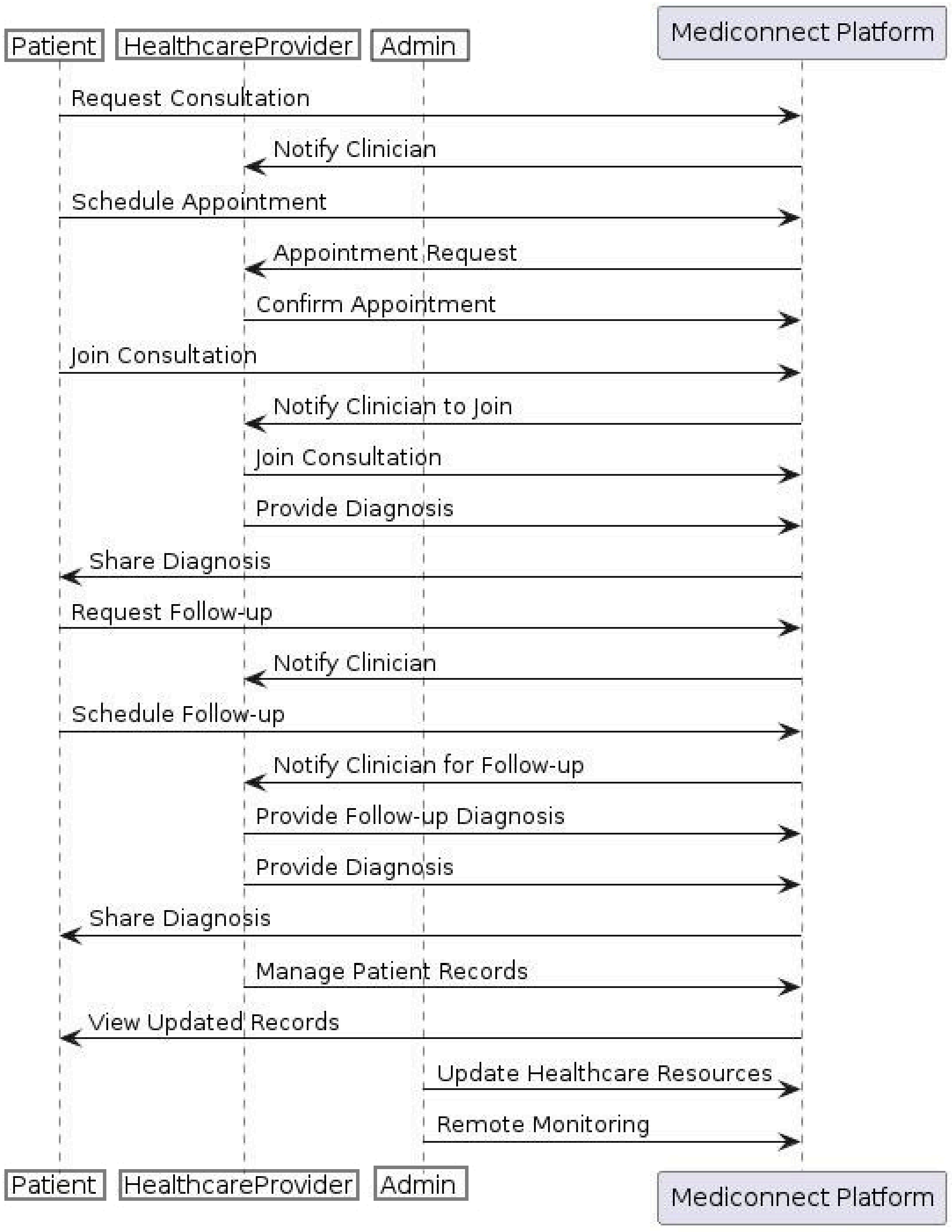


Figure 2. Sequence Diagram

3.3.3 Activity Diagram

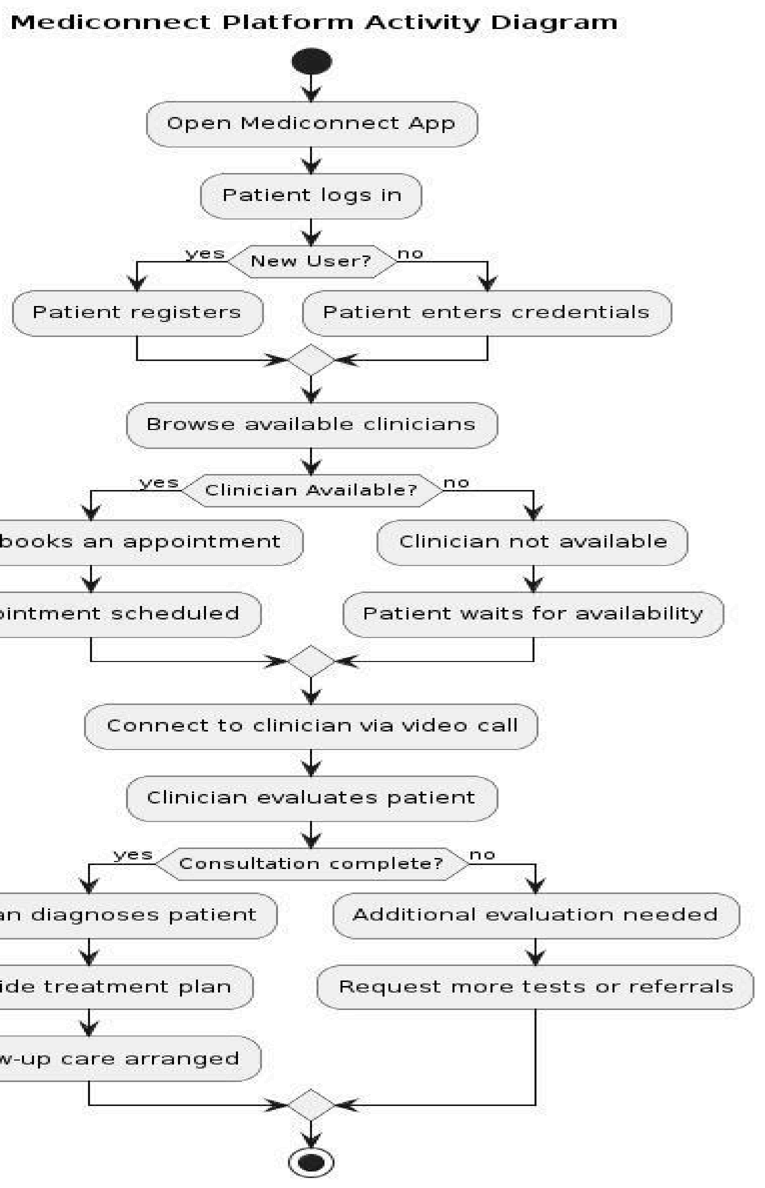

Mediconnect Platform Activity Diagram
Open Mediconnect App
Patient logs in
New User?
yes
no
Patient registers
Patient enters credentials
Browse available clinicians
Clinician Available?
yes
no
Patient books an appointment
Appointment scheduled
Clinician not available
Patient waits for availability
Connect to clinician via video call
Clinician evaluates patient
Consultation complete?
yes
no
Clinician diagnoses patient
Provide treatment plan
Follow-up care arranged
Additional evaluation needed
Request more tests or referrals

Figure 3. Activity Diagram

3.3.4 Entity Relationship Diagram

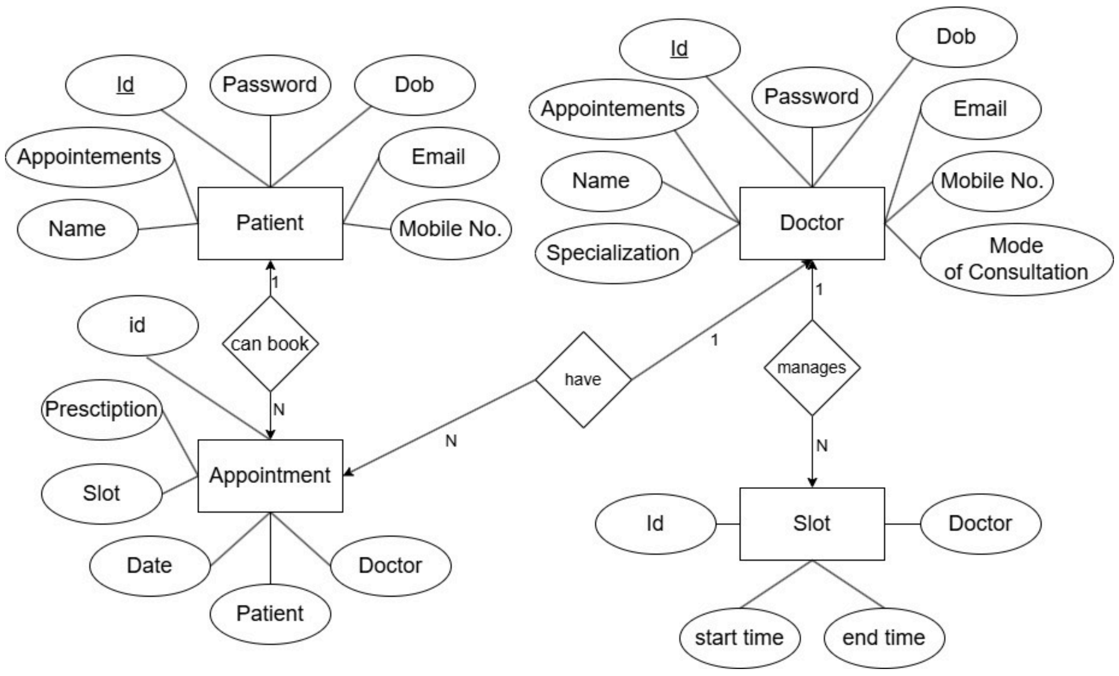


Figure 4: Entity Relationship diagram

3.4 Database Structure

The database structure for the *MediConnect* is designed to efficiently store and manage data related to users (doctor and patient), health records, and appointment data. It consists of several relationships to ensure seamless functionality and data integrity:

3..4.1 Patient:

- **User ID**: Unique identifier for each patient
- **Username**: The username associated with the user
- **Email**: Contact email for the user
- **Password**: Encrypted password for secure user authentication
- **Profile Information**: Additional personal information like name, contact details (optional)

3.4.2 Doctor:

- **User ID**: Unique identifier for each doctor
- **Username**: The username associated with the user
- **Email**: Contact email for the user
- **Clinic Address**: address of their clinic
- **Mode**: Online or Offline
- **Password**: Encrypted password for secure user authentication
- **Profile Information**: Additional personal information like name, contact details (optional)

3.4.3 Appointment:

- **Appointment ID**: Unique identifier for each appointment record
- **Date**: Date of the class for which attendance is being marked
- **Status**: Upcoming or completed
- **Patient**: For which patient
- **Doctor**: Which doctor appointed
- **Prescription**: Given by doctor(Optional)

3.5 Deployment Requirements

The successful deployment of the *MediConnect System* is crucial for ensuring the smooth and secure operation of the platform in a live environment. This phase encompasses the necessary hardware and software specifications to support the functionalities and performance goals as outlined in the project description. The following deployment requirements will ensure that the system operates at optimal performance and security levels.

3.5.1 Hardware

The deployment of the *MediConnect* platform requires a robust hardware setup to accommodate both the user-facing web application and the backend components, including the database processes. The following hardware specifications are recommended to ensure seamless functionality and scalability:

1. **Web Server:**
    - ○ **CPU and RAM**: The web server should be equipped with sufficient CPU and RAM resources to handle user requests, particularly for video call requests, and appointment scheduling.
    - ○ **Storage**: Adequate disk space is required to store user data, appointment records.
2. **Database Server:**
    - ○ **CPU and RAM**: The database server should be robust enough to support the operations of the MySQL database, with adequate CPU and RAM to store user profiles, appointment logs, verification data, and system logs efficiently.
    - ○ **Storage**: Sufficient storage capacity is necessary to handle the large volume of data, as well as metadata related to system.
3. **Network Infrastructure:**
    - ○ **High-Speed Internet**: A reliable, high-speed internet connection is crucial for ensuring fast data transfer between the web server and the database server, particularly for handling real-time data updates and data processing.

3.5.2 Software

The deployment of the *MediConnect System* will rely on a specific set of software components that are aligned with the project's technology stack and the features outlined in the system design.

1. Development Environment Setup

- **Languages and Frameworks**:
    - ○ **Backend**: Java (Spring Boot) for the API, along with MySQL/PostgreSQL for relational data storage.

- **Frontend**: React.js for a dynamic and responsive user interface.
- **Mobile**: React Native for cross-platform mobile app development (iOS and Android).

- **Version Control**: Git for source code management with GitHub/GitLab.
- **Development Tools**: IntelliJ IDEA for backend, Visual Studio Code for frontend, and Android Studio for mobile app development.

2. Testing Environment

- **Automated Testing**:
    - **Unit Testing**: JUnit and Mockito for backend testing.
    - **UI Testing**: Jest and React Testing Library for frontend testing.
    - **Integration Testing**: TestNG for testing APIs and overall system integrations.
- **Continuous Integration/Continuous Deployment (CI/CD)**:
    - Tools like Jenkins, GitLab CI, or CircleCI will be used to automate the deployment pipeline, ensuring that each code change is automatically tested and deployed.
- **Staging Environment**:
    - A staging server that replicates the production environment for user acceptance testing (UAT). This ensures that everything works as expected before the final deployment.

3. Production Environment Setup

- **Cloud Infrastructure**:
    - The system will be hosted on cloud platforms such as **AWS** or **Google Cloud Platform (GCP)** to ensure scalability and reliability.
    - Use of **Elastic Beanstalk (AWS)** or **Google Kubernetes Engine (GKE)** for managing microservices and containers in the production environment.

- **Database**:
  - Use a managed relational database such as **Amazon RDS** for MySQL/PostgreSQL or **Google Cloud SQL** to host patient and provider data.
- **Load Balancing and Auto-Scaling**:
  - Configure load balancers to distribute traffic evenly across multiple server instances and set up auto-scaling to adjust resources based on demand.
- **Content Delivery Network (CDN)**:
  - Use **AWS CloudFront** or **Cloudflare** for faster delivery of static assets (images, CSS, JS) and to improve performance.

3.6 Summary

The successful deployment of the *MediConnect System* requires careful consideration of both hardware and software requirements. By provisioning the appropriate infrastructure, including web servers, database servers, network resources, and security components, the platform will be able to support its users' needs effectively. Additionally, leveraging the appropriate technology stack for ensuring secure data handling will guarantee that the system remains robust, scalable, and secure for the long term. These deployment requirements form the foundation for a seamless and efficient deployment process, ensuring the platform operates as intended once it goes live.

CHAPTER 4

DEVELOPMENT

CHAPTER 4: DEVELOPMENT

4.1: Techniques Used

The *MediConnect System* incorporates a variety of development techniques and methodologies to ensure it functions effectively and can easily adapt to the dynamic needs of Medical Clinics. The chosen techniques contribute significantly to the platform's flexibility, scalability, and alignment with modern web development practices. Below are the key techniques used in the project:

4.1.1 Agile Development Methodology

Agile development played a vital role in the development of *MediConnect*, enabling continuous improvement, iterative updates, and alignment with user feedback. Agile methodology facilitated the flexible adaptation of the system, ensuring it could meet the evolving needs of students, teachers, and administrators. By working in short development cycles (sprints), the team could frequently assess the progress, address issues quickly, and enhance features based on user input. This approach ensured that the project remained customer-focused, and delivery times remained manageable.

4.2 Tools Used

The development of the *MediConnect System* utilized a wide array of tools and technologies that facilitated the creation of a modern, scalable, and secure application. These tools were integral to ensuring a smooth development process, robust functionality, and optimal user experience. Below are the key tools used in the project:

4.2.1 Integrated Development Environment (IDE)

- **IntelliJ Idea:** IntelliJ IDEA is a powerful and feature-rich Integrated Development Environment (IDE) that enhances productivity and is widely used for Java-based backend development.

4.2.2 Version Control and Collaboration

- **Git:** Git served as the version control system, enabling efficient tracking of code changes, collaborative development, and versioning. It ensured that all changes made during development could be safely integrated into the main codebase.
- **GitHub:** GitHub was used for hosting the project repositories, managing code reviews, tracking issues, and facilitating team collaboration.

4.2.3 Front-End Development

- **HTML5:** HTML5 was employed to structure the web pages and ensure semantic markup across different browsers and devices.
- **CSS3:** CSS3 was utilized for styling the application, providing a responsive, user-friendly design, and ensuring that the platform is visually appealing on both desktop and mobile devices.
- **JavaScript:** JavaScript was used to create dynamic, interactive elements on the front end, including real-time features such as the live attendance tracking and notifications.
- **React.js:** React.js was used to build dynamic, component-based user interfaces, enabling efficient rendering and real-time updates, which is crucial for interactive applications like attendance management.

4.2.4 Back-End Development

- **Java and Spring Boot Development:** IntelliJ IDEA is the primary IDE for writing backend code in Java. With built-in support for Spring Boot, it enables rapid development of RESTful APIs, simplifying tasks such as dependency injection, security configuration, and database interaction. The IDE provides code suggestions, refactorings, and error detection for Spring Boot projects.

4.3 Languages Used

The development of *MediConnect* involved several languages and technologies, each serving a specific purpose to deliver a seamless and efficient application. Below are the key languages and technologies employed:

4.3.1 Front-End Technologies

- **JavaScript:** JavaScript played a central role in the platform, enabling dynamic interactions between users and the application, both in terms of the web interface and server-side communications.
- **HTML5:** HTML5 provided the structure and foundation for the web pages, ensuring compatibility with different browsers and devices.
- **CSS3:** CSS3 was used to ensure the platform's visual aesthetics and responsiveness, allowing users to access the system from various devices without issues.
- **React.js:** React.js was used to create a fast, interactive, and modular user interface, simplifying the process of building dynamic and reusable components.

4.3.2 Back-End Technologies

- **Java:** Java is chosen for backend development due to its robustness, security features, and scalability. It has a vast ecosystem of libraries and frameworks, making it an ideal choice for building enterprise-grade applications like MediConnect.
- **Spring Boot:** It is a popular framework for Java-based backend development. It simplifies the process of creating production-ready applications by providing ready-to-use features like:
- **REST API** development: Spring Boot makes it easy to build RESTful services that can handle various client requests such as creating, reading, updating, and deleting (CRUD) data.

4.3.3 Database Management

- **MySQL:** MySQL was chosen as the database management solution for its flexible, SQL architecture. It efficiently stores and manages user profiles, attendance records, and images, all of which are crucial for the functioning of the platform.

4.3.4 Version Control

- **Git:** Git enabled version control and collaboration between developers, ensuring that all changes were tracked and managed throughout the development cycle.

4.4: Testing

A comprehensive testing strategy was crucial in ensuring the reliability, security, and overall performance of the *MediConnect*. The testing process was designed to cover all aspects of the system to guarantee a seamless user experience and high-quality output. Below are the key testing approaches used in the project:

4.4.1 Unit Testing

- **Objective:** Unit testing aimed to validate the individual components of the platform, ensuring that each function, module, and API endpoint performed as expected.
- **Components to Test:**
  - Front-end components (React.js components, form validation).
  - Back-end modules (API routes, business logic, image processing functions).

4.4.3 System Testing

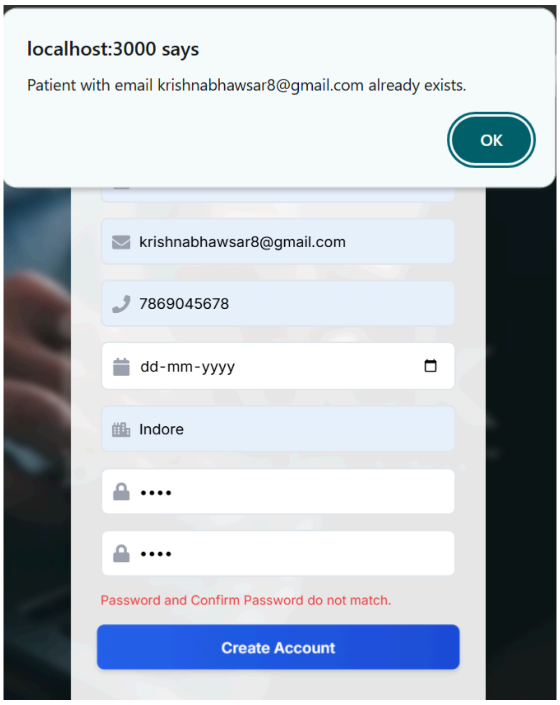
Test Case and Analysis

TEST CASE - 1

Test Case ID	TC001
Test Case Summary	Error Handling for Duplicate Email
Steps to Reproduce	1. Navigate to the "Create Account" page. 2. Enter "krishnabhawsar8@gmail.com" as the email address. 3. Fill in other required fields. 4. Click "Create Account".
Expected Result	The system should display an error message indicating that the email address is already in use.
Actual Result	The system displays the message "Patient with email krishnabhawsar8@gmail.com already exists."
Status	Pass

Table 1: Duplicate Email Test Case

TEST CASE 1 OUTPUT



localhost:3000 says

Patient with email krishnabhawsar8@gmail.com already exists.

Password and Confirm Password do not match.

Figure 5: Duplicate Email Test Case Output

TEST CASE - 2

Test Case ID	TC002
Test Case Summary	Verify OTP Functionality
Steps to Reproduce	1. Navigate to the email verification page.
Expected Result	The system should successfully verify the OTP and redirect the user to the next page.
Actual Result	The application sent a 4 digit otp to krishnabhawsar8@gmail.com
Status	Pass

Table 2: Verify OTP Functionality Test Case

TEST CASE 2 OUTPUT

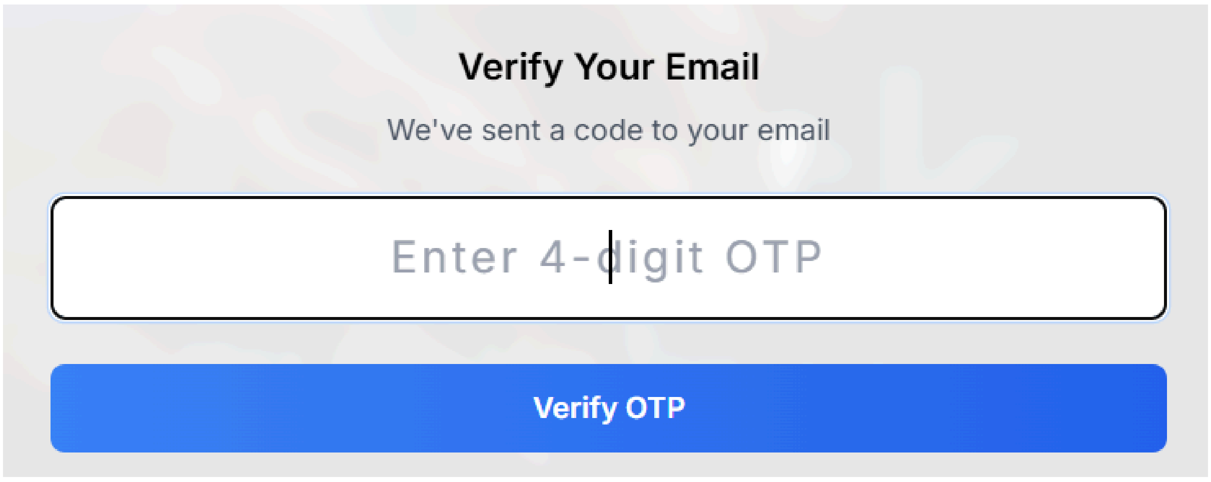


Figure 6: Verify OTP Functionality Test Case Output

TEST CASE - 3

Test Case ID	TC003
Test Case Summary	Search Functionality
Steps to Reproduce	1. Navigate to the MediConnect homepage. 2. Select a search criteria (Name, City, or Specialisation). 3. Enter a valid search query. 4. Click the "Search" button.
Expected Result	The system should display a list of doctors matching the search criteria.
Actual Result	Search Functionality works proper for finding doctor
Status	Pass

Table 3: Search Functionality Test Case

TEST CASE 3 OUTPUT

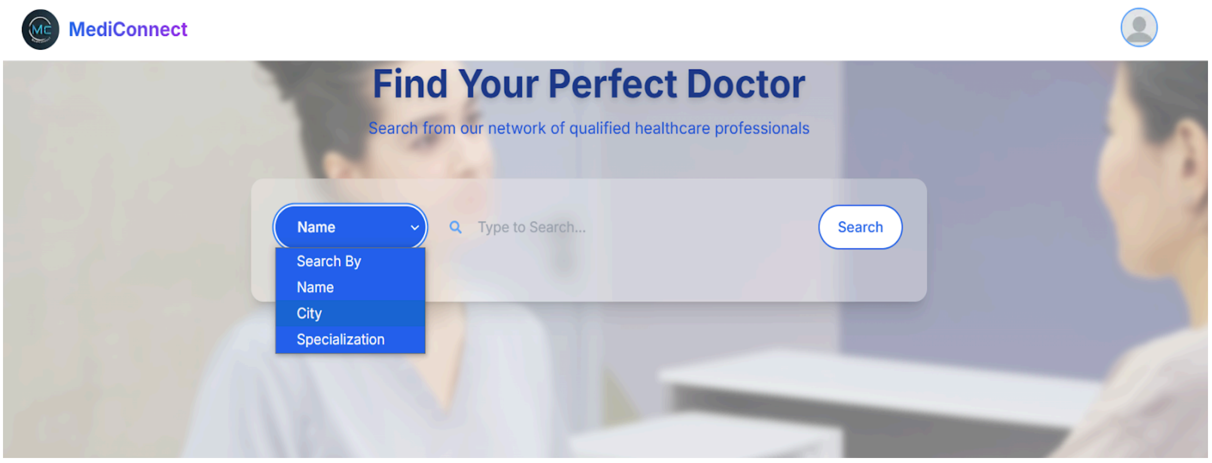


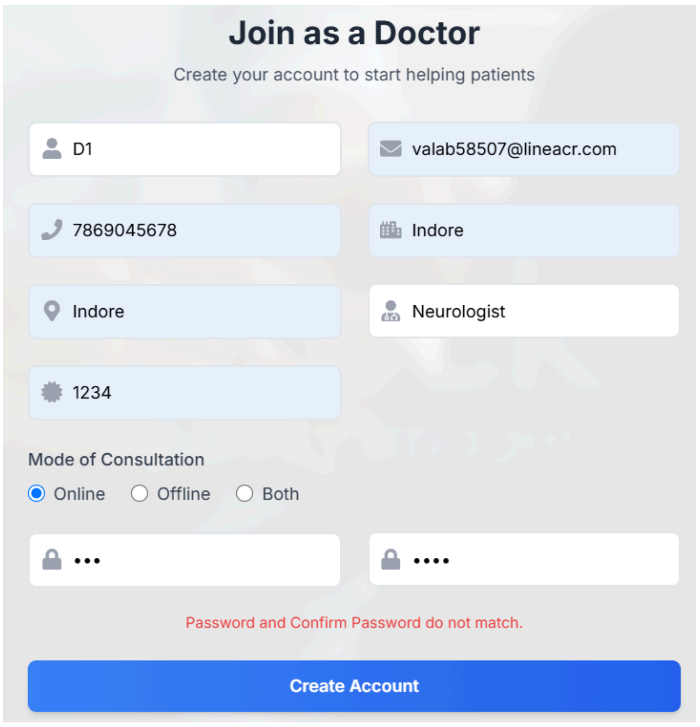
Figure 7 : Search Functionality Test Case Output

TEST CASE - 4

Test Case ID	TC004
Test Case Summary	Verify Password Mismatch Error
Steps to Reproduce	1. Navigate to the "Join as a Doctor" page. 2. Fill in all required fields. 3. Enter different passwords in the "Password" and "Confirm Password" fields. 4. Click the "Create Account" button.
Expected Result	The system should display an error message indicating that the passwords do not match.
Actual Result	Enter different passwords in the "Password" and "Confirm Password" fields and it shows password and confirm password do not match
Status	Pass

Table 3: Verify Password Mismatch Error Test Case

TEST CASE 4 OUTPUT



Join as a Doctor

Create your account to start helping patients

D1 | valab58507@lineacr.com

7869045678 | Indore

Indore | Neurologist

1234

Mode of Consultation

(● Online) ○ Offline ○ Both

••• | ••••

Password and Confirm Password do not match.

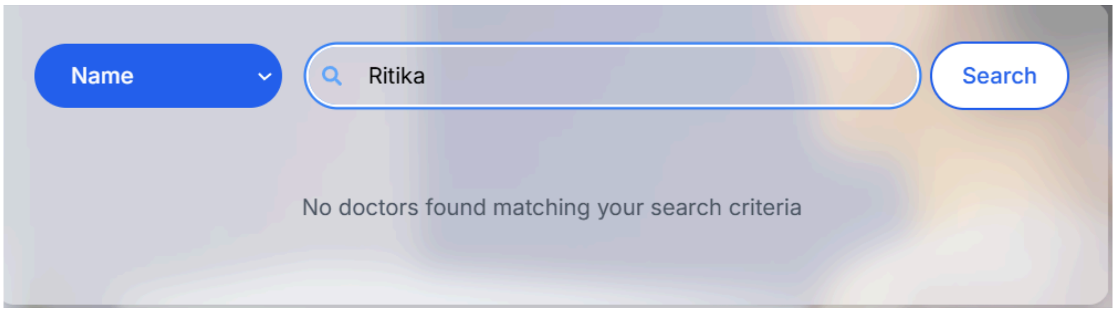
Figure 8 : Verify Password Mismatch Error Test Case Output

TEST CASE - 5

Test Case ID	**TC005**
Test Case Summary	Verify the system displays "No doctors found" message when searching for a non-existent doctor.
Steps to Reproduce	1. Navigate to the "Find a Doctor" search page. 2. In the search bar, set the search filter to "Name". 3. Enter "Ritika" in the search input field. 4. Click the "Search" button.
Expected Result	The system should display the message: "No doctors found matching your search criteria".
Actual Result	The system displays the message: "No doctors found matching your search criteria" as expected.
Status	Pass

Table 5: Verify the system displays "No doctors found" Test Case

TEST CASE 5 OUTPUT



No doctors found matching your search criteria

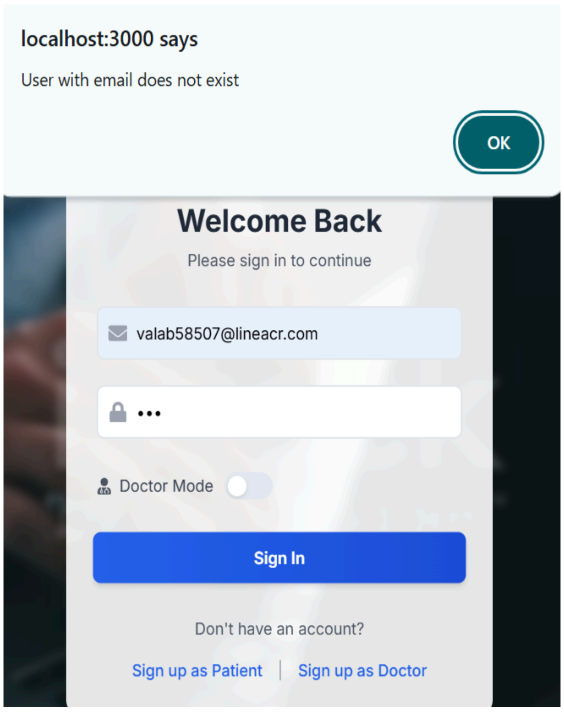
Figure 9 : Verify the system displays "No doctors found" Test Case Output

TEST CASE - 6

Test Case ID	TC006
Test Case Summary	Verify the system displays "User with email does not exist" message when trying to log in with a non-existent email.
Steps to Reproduce	1. Navigate to the login page. 2. Enter email (valab58507@lineacr.com) in the email input field. 3. Enter a password in the password input field. 4. Click the "Sign In" button.
Expected Result	The system should display a message: "User with email does not exist".
Actual Result	The system displays the message: "User with email does not exist" as expected.
Status	Pass

Table6: Verify the system displays "User with email does not exist" Test Case

TEST CASE 6 OUTPUT


localhost:3000 says

User with email does not exist

Figure 10 : Verify the system displays "User with email does not exist" Test Case Output

CHAPTER 5

CONCLUSION

Chapter 5: CONCLUSION

The MediConnect platform aims to revolutionize healthcare delivery by providing a secure, efficient, and user-friendly system for managing patient information, doctor appointments, and medical records. By leveraging modern backend technologies such as Java, Spring Boot, and MySQL/PostgreSQL, the platform ensures reliable data management, secure authentication, and seamless interaction between users and healthcare services. The adoption of technologies like RESTful APIs guarantees a secure and scalable solution that can handle the growing needs of healthcare providers and patients.

In conclusion, the MediConnect platform stands as a comprehensive solution to modern healthcare challenges, enabling seamless communication, improving patient care, and streamlining healthcare management processes. With its robust technological stack and a focus on security, scalability, and user experience, MediConnect has the potential to drive significant improvements in healthcare systems, making it easier for patients and healthcare providers to interact efficiently and effectively.

CHAPTER 6

Opportunities and Recommendations for Future Research

Chapter 6: Opportunities and Recommendations for Future Research

Despite the advancements made in the MediConnect project, there are several limitations that need to be addressed to further enhance the platform's capabilities.

Limitations

1. **Limited Integration with External Healthcare Systems**: Currently, the system may be limited in integrating with existing hospital management systems, electronic health records (EHR), and other third-party healthcare tools. Integration with legacy systems can be challenging due to varying standards and protocols, hindering the seamless exchange of medical data across platforms.

2. **Scalability Constraints in Peak Usage**: While cloud deployment and microservices architecture ensure scalability, there could still be performance bottlenecks during peak usage, such as when multiple users access the platform simultaneously, especially if the infrastructure is not dynamically scaled in real-time.

3. **User Experience for Elderly Patients**: The platform may not fully address the needs of elderly patients who may face challenges in navigating digital platforms. The user interface, though optimized for the majority, may need additional adjustments to cater to less tech-savvy users.

4. **Regulatory and Compliance Issues**: Compliance with healthcare regulations, such as HIPAA (Health Insurance Portability and Accountability Act) in the U.S., or similar regulations in other countries, may require constant updates to the platform to ensure legal conformity. Adapting to varying regulatory standards across different regions can pose challenges.

Suggestions for Future Work

1. **Enhanced Data Security with Blockchain**: A potential future enhancement could be the integration of blockchain technology to ensure tamper-proof medical records. Blockchain can provide an additional layer of security and transparency for patient data, ensuring it remains immutable and trustworthy while allowing patients to control their medical records more securely.

2. **Interoperability with Other Healthcare Systems**: Future work should focus on improving the platform's ability to integrate with existing Electronic Health Records (EHR), hospital management systems **(HMS)**, and other external healthcare solutions. Implementing standardized data exchange protocols like FHIR (Fast Healthcare Interoperability Resources) will enable smoother data transfer and improve overall healthcare service delivery.

3. **AI Integration for Personalized Healthcare**: Introducing artificial intelligence (AI) and machine learning (ML) algorithms could improve the platform's capability in areas like predictive analytics, personalized health recommendations, and automated diagnosis assistance. AI could analyze historical health data to predict future health issues or suggest preventive measures for patients.

4. **Improved User Interface for Diverse Demographics**: To ensure that the platform is more accessible, future versions could focus on designing a more intuitive user interface, especially for elderly patients. This could involve larger text, voice commands, easy-to-navigate layouts, and integration with accessibility features such as screen readers.

5. **Real-Time Analytics and Health Monitoring**: Future versions of the platform could include real-time health monitoring through integration with wearable health devices like smartwatches and fitness trackers. This would allow continuous tracking of patient health metrics (e.g., heart

rate, blood pressure) and provide real-time data to healthcare providers for better decision-making.

6. **Mobile App Enhancement for Remote Areas**: While the platform will be accessible through both web and mobile apps, future work can focus on enhancing the mobile app's offline capabilities. This will allow patients and healthcare providers in remote areas with limited internet access to continue using key features of the platform without disruption.

7. **Compliance and Regulation Updates**: Continuous updates to meet the evolving healthcare regulations and data privacy laws are necessary. Future work should include a framework for easily updating the platform to remain compliant with regional and global health data protection standards.

BIBLIOGRAPHY

REFERENCE

[1] React. (2024). *Official React Documentation*. React. https://react.dev/

[2] Tata 1mg. (2024). *The Journey of Tata 1mg: A Tale of Innovation and Healthcare Transformation*. The Times Reach. Retrieved fromThe Times Reach ps://1mg. Tata 1mg. (2024). *Editorial Policy and Processes*. Tata 1mg. Retrieved from [https:/1mgmg.com/editorial-policy-processes].

[3] Practo. (2024). *Find Doctors, Clinics, and Hospitals*. Practo. Retrieved from https://www.practo.in.

[4] Spring. (2024). *Spring Framework Documentation*. Spring. https://docs.spring.io/

[5] GeeksforGeeks. (2021). Data Flow Diagrams (DFD) for System Analysis Design.GeeksforGeeks.https://www.geeksforgeeks.org/data-flow-diagrams-dfd-fo r-system-analysis-and-design/

[6] A Literature Review: Website Design andUserEngagement. guides.himmelfarb.gwu.edu

Research Paper

MediConnect – A Telemedicine Platform

KRISHNA BHAWSAR
CSE DEPARTMENT
ACROPOLIS INSTITUTE OF TECHNOLOGY AND RESEARCH INDORE, INDIA
krishnabhawsar210222@acropolis.in

JIGYANSH SISODIYA
CSE DEPARTMENT
ACROPOLIS INSTITUTE OF TECHNOLOGY AND RESEARCH INDORE, INDIA
jigyanshsisodiya210446@acropolis.in

KRISHNA GUPTA
CSE DEPARTMENT
ACROPOLIS INSTITUTE OF TECHNOLOGY AND RESEARCH INDORE, INDIA
krishnagupta210208@acropolis.in

ISHANT MANDLOI
CSE DEPARTMENT
ACROPOLIS INSTITUTE OF TECHNOLOGY AND RESEARCH INDORE, INDIA
ishantmandloi210298@acropolis.in

KUNAL YADAV
CSE DEPARTMENT
ACROPOLIS INSTITUTE OF TECHNOLOGY AND RESEARCH INDORE, INDIA
kunalyadav210416@acropolis.in

Abstract: MediConnect is a digital medical platform designed to connect patients and doctors. It is an online healthcare website that facilitates remote consultation, diagnosis, and treatment recommendations in the field of telemedicine. By leveraging technology, MediConnect aims to democratize healthcare by providing simple, affordable solutions to deliver better, more optimal care. This platform eliminates the need for travel and other time-consuming expenses, allowing patients to connect directly with the right doctors from the comfort of their homes.

Keywords: Telemedicine, Remote consultation, diagnosis, healthcare.

1. INTRODUCTION

The rapid development of information and communication technology has enabled telemedicine to fill the gap in healthcare through access to treatment. Telemedicine involves the use of communication technology to deliver healthcare services with the aim of improving access to healthcare in many underserved healthcare settings. This can be done through the use of healthcare systems such as MediConnect, which provide remote communication or the ability to manage healthcare with a physician. Collaboration, especially for rural areas and the underserved. Research shows that telemedicine not only reduces healthcare burden but also improves health outcomes due to timely management (Smith et al.,

2020) [1, p. 230]. This paper aims to provide a detailed overview of MediConnect, its development and potential for health impact, and to address issues and ethics in leadership.

2. LITERATURE REVIEW

Telemedicine in healthcare has been well-documented and is still a major topic in the literature, with the promise of eliminating the problem area and improving access to medical services. The use of telemedicine in rural and underserved areas alleviates the problem of limited medical facilities and fewer medical staff (Jones & Lee, 2021) [2, p. 155]. A good review by Smith et al. (2020) showed that the benefits of telemedicine interventions improve medical outcomes and reduce medical costs, especially in remote areas where medical standards are poor [1, p. 232]. Remote patient care, teleradiology, etc. have been used worldwide and success in care has been reported (Kumar et al., 2019).

3. METHODOLOGY

The development and design of MediConnect are summarized in the methodology. The Application (MediConnect) is made with modern web technologies and the purpose of the platform is to provide telemedicine services, help patients and doctors consult and manage medical records.

3.1 Front-end (React.js):

MediConnect's front end is built using a widely used JavaScript library called React.js that supports dynamic and reactive user interfaces. The module has modular features that make it possible to create scalable applications. The features of this platform include:

- Patient and provider dashboards to log in and access appointments, medical information, and consultations.
- A secure messaging system to ensure data security.
- A video conferencing interface for real-time consultations.

3.2 Back-end (Spring Boot):

Mediconnect's back-end is built using a powerful Java framework called Spring Boot, making it easily scalable, secure, and accessible. Spring Boot was chosen for its ability to manage business logic and its scalability associated with multiple systems. The backend has the following functions:

- A RESTful API that facilitates communication between the frontend and the data.
- Authentication and authorization functions to ensure that users have the correct access rights.
- A video conferencing service for instant meetings.

3.3 Database (MySQL):

MySQL database to store reference information, medical information, appointment information, and consultation history. MySQL was chosen because it is powerful, scalable, and has good support for handling relational data models. It also provides data integrity and easy management of large medical records.

All communication between the frontend written in React.js and the backend used in Spring Boot takes place using the REST API. The backend contains logic for scheduling, medical records, and video conferencing. MySQL handles data storage and retrieval. The platform integrates third-party services for video chat.

3.4 Design Diagram

The MediConnect architecture follow client-server model, with the frontend (React.js) communicating with the backend (Spring Boot) via REST APIs. The backend handles logic related to appointments, medical records, and video consultations. MySQL manages data storage and retrieval. The platform also integrates third-party services for video conferencing.

3.4.1 Entity Relationship Diagram

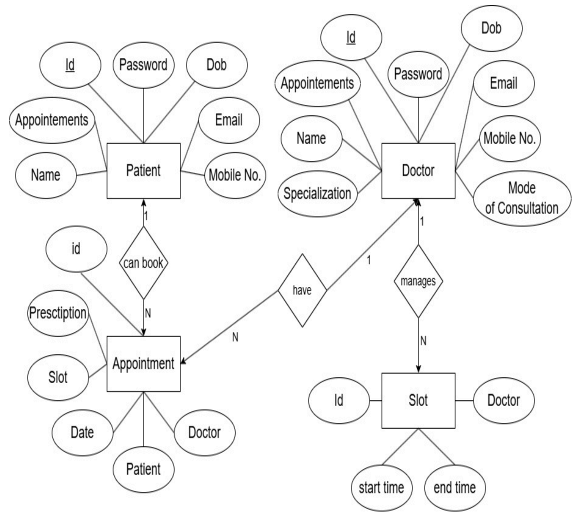


Figure 1. MediConnect- ER Diagram

3.4.2 Use Case Diagram

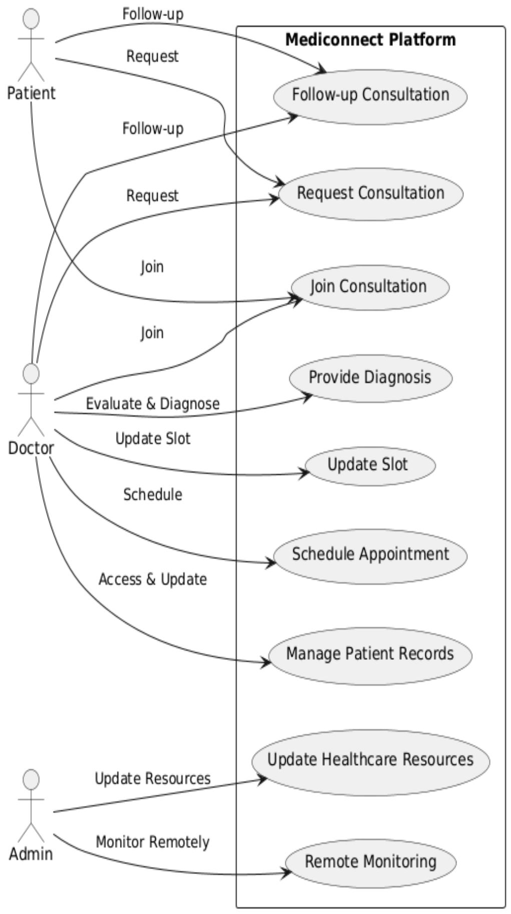


Figure 2. MediConnect- Use Case Diagram

4. TELEMEDICINE APPLICATIONS IN HEALTHCARE

Telemedicine applications have recently been used in many areas for the provision of healthcare services and provide many benefits to patients and doctors. Mediconnect is designed to support the following telemedicine applications:

Remote Consultation:
Virtual consultations are the basis of MediConnect. Patients can share their health problems with doctors via video on the platform without having to travel long distances.

This application is beneficial for people living in rural areas because they have difficulty finding a doctor.

Chronic Disease Management:
Mediconnect can be adapted for remote monitoring of chronic diseases such as diabetes and hypertension.The system allows the device to monitor vital signs, allowing doctors to remotely monitor the patient's condition and intervene when necessary. Mediconnect provides a secure platform for patients to consult with psychiatrists and therapists, especially those living in areas where access to psychiatrists is limited. It allows images to be sent for medical evaluation via remote communication. This application is especially useful in areas where there is a shortage of specialists.

5. BENEFITS AND IMPACT OF TELEMEDICINE IN RURAL AND UNDERSERVED COMMUNITIES

Telemedicine platforms like Mediconnect solve telemedicine problems by connecting specialists in remote areas. Patients living in remote areas can now consult specialists who are otherwise unavailable. Mediconnect is so effective that it not only reduces the patient's transportation costs but also the doctor's operating costs. The basic principle of Mediconnect is to enable doctors to intervene when needed, without the delays that can occur in face-to-face visits. Continuous monitoring of Mediconnect enables the management of chronic diseases.

5.1 Challenges and Decisions:
The processing of telemedicine use depends on the ability to maintain a reliable internet connection. Low connectivity will limit the effectiveness of platforms like Mediconnect in most rural areas. The involvement of both patients and doctors requires them to have the technological knowledge to accept and adopt the platform. Mediconnect provides strict security and data encryption to protect patient privacy. However, risks cannot be ignored, and companies need to develop and improve their

security in connection with cyberattacks. Therefore, these regulations must be followed to ensure that the platform complies with legal precedents in terms of conduct and data protection.

6. CHALLENGES AND CONSIDERATIONS

Being one of the advantages of telemedicine, the following challenges arise with the use of the platforms:

Technological Barriers:
Telemedicine success realization depends on having access to reliable internet connectivity; otherwise, low connectivity may limit the effectiveness for most rural areas using platforms like Mediconnect (Jones & Lee, 2021) [2, p. 160]. Patients and providers are involved, which makes them lack technological literacy to accept and adopt.

Privacy and Security Issues:
Critical issues related to the use of telemedicine include the confidentiality of sensitive health information. Mediconnect ensures strict security measures and encryption of data in order to protect the confidentiality of the patient. However, the risk cannot be overlooked and companies need to develop and improve upon their security in continuous cycles of cyberattacks [3, p. 362].

Regulatory and Legal Issues:
Regulations of telemedicine differ from country to country and regionally, hence access or delivery of health care from one state to another can be complicated. Therefore, to ensure that the platform has conformity with the legal premises for medical practice and data protection, observance of such regulations is a must (Kumar et al., 2019) [3, p. 358].

7. RESULTS AND ANALYSIS DISCUSSION

The results of pilot user testing for Mediconnect were promising as patients and healthcare professionals were overall satisfied with the usability of the platform. Several key points in summary of some of the findings are as follows:
User Experience:
- Users noted that the interface of the platform is very friendly and easy to use.
- However, from time to time, problems related to connectivity, particularly internet connectivity, had impacted the quality of video consultations, especially in rural areas (Kumar et al., 2019) [3, p. 364].

Performance:
- The platform can handle heavy loads with several concurrent users without noticeable delays.
- However, system improvements will be necessary

when Mediconnect expands to accommodate more users.

8. REFERENCES

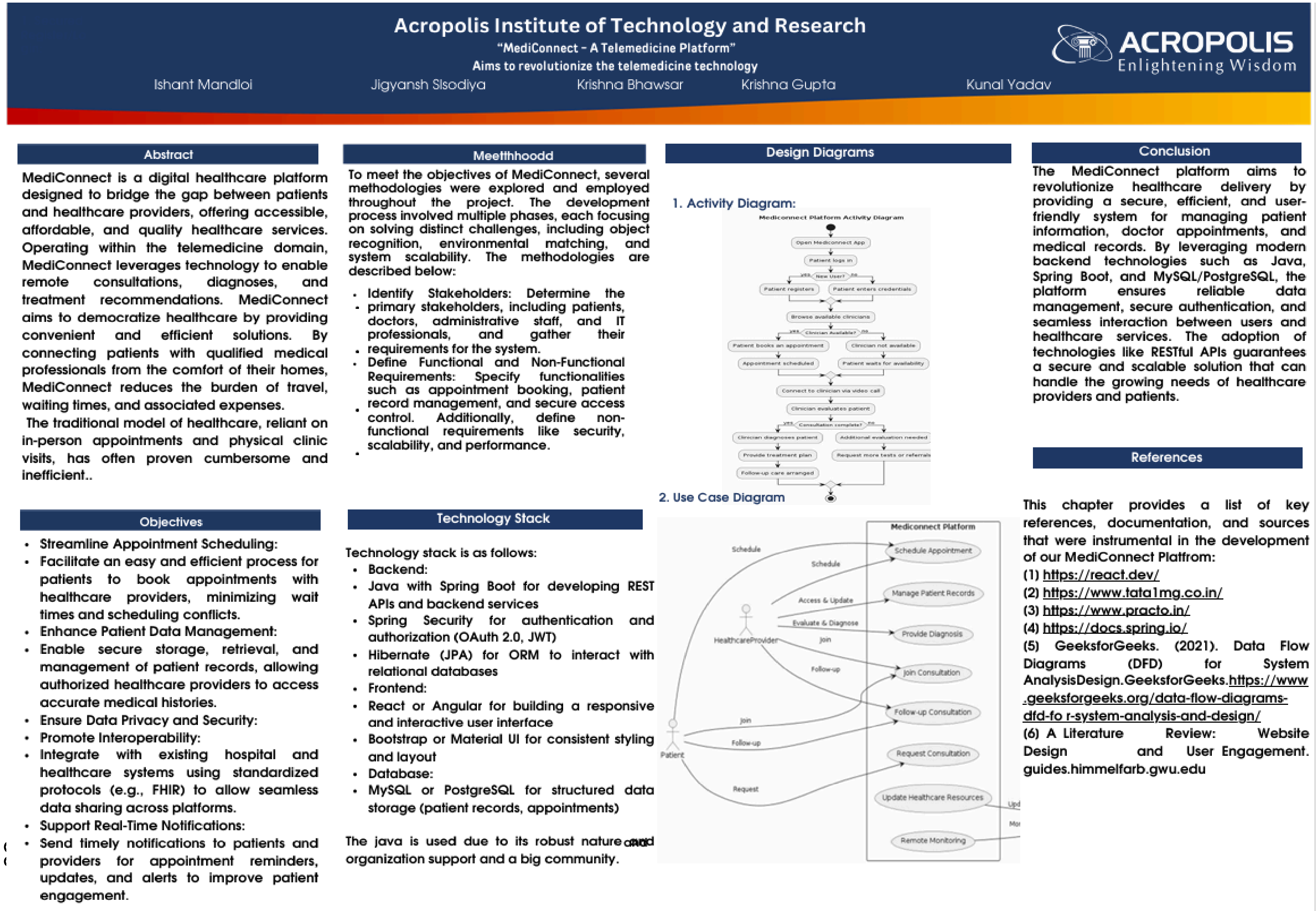
[1] Smith, A., Johnson, B., & Richards, C. (2020). The impact of telemedicine on healthcare costs and outcomes: A systematic review. Journal of Health Informatics, 15(4), 229-241. (https://pmc.ncbi.nlm.nih.gov/articles/PMC10993086/)

[2] Jones, M., & Lee, J. (2021). Telemedicine: Transforming access to healthcare in rural communities. Rural Health Journal, 10(3), 154-162. (https://www.ncbi.nlm.nih.gov/books/NBK588278/

[3] Kumar, S., Patel, M., & Shankar, R. (2019). Telemedicine solutions for rural health care delivery. International Journal of Telehealth and Telecare, 24(5), 356-367. (https://pubmed.ncbi.nlm.nih.gov/16388171/

[4] Spring Framework Documentation: Web MVC (https://docs.spring.io/spring-framework/reference/web/webmvc.html)

Technical Poster



Acropolis Institute of Technology and Research

"MediConnect - A Telemedicine Platform"

Aims to revolutionize the telemedicine technology

Ishant Mandloi Jigyansh Sisodiya Krishna Bhawsar Krishna Gupta Kunal Yadav

ACROPOLIS Enlightening Wisdom

Abstract

MediConnect is a digital healthcare platform designed to bridge the gap between patients and healthcare providers, offering accessible, affordable, and quality healthcare services. Operating within the telemedicine domain, MediConnect leverages technology to enable remote consultations, diagnoses, and treatment recommendations. MediConnect aims to democratize healthcare by providing convenient and efficient solutions. By connecting patients with qualified medical professionals from the comfort of their homes, MediConnect reduces the burden of travel, waiting times, and associated expenses.

The traditional model of healthcare, reliant on in-person appointments and physical clinic visits, has often proven cumbersome and inefficient..

Objectives

- Streamline Appointment Scheduling:
- Facilitate an easy and efficient process for patients to book appointments with healthcare providers, minimizing wait times and scheduling conflicts.
- Enhance Patient Data Management:
- Enable secure storage, retrieval, and management of patient records, allowing authorized healthcare providers to access accurate medical histories.
- Ensure Data Privacy and Security:
- Promote Interoperability:
- Integrate with existing hospital and healthcare systems using standardized protocols (e.g., FHIR) to allow seamless data sharing across platforms.
- Support Real-Time Notifications:
- Send timely notifications to patients and providers for appointment reminders, updates, and alerts to improve patient engagement.

Meethhoodd

To meet the objectives of MediConnect, several methodologies were explored and employed throughout the project. The development process involved multiple phases, each focusing on solving distinct challenges, including object recognition, environmental matching, and system scalability. The methodologies are described below:

- Identify Stakeholders: Determine the
- primary stakeholders, including patients, doctors, administrative staff, and IT professionals, and gather their
- requirements for the system.
- Define Functional and Non-Functional Requirements: Specify functionalities such as appointment booking, patient
- record management, and secure access control. Additionally, define non-functional requirements like security, scalability, and performance.
-

Technology Stack

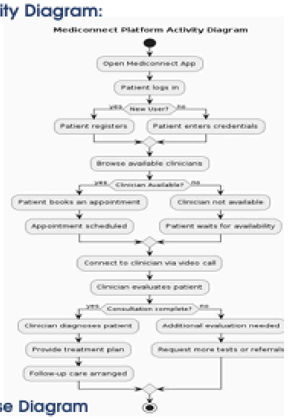
Technology stack is as follows:

- Backend:
- Java with Spring Boot for developing REST APIs and backend services
- Spring Security for authentication and authorization (OAuth 2.0, JWT)
- Hibernate (JPA) for ORM to interact with relational databases
- Frontend:
- React or Angular for building a responsive and interactive user interface
- Bootstrap or Material UI for consistent styling and layout
- Database:
- MySQL or PostgreSQL for structured data storage (patient records, appointments)

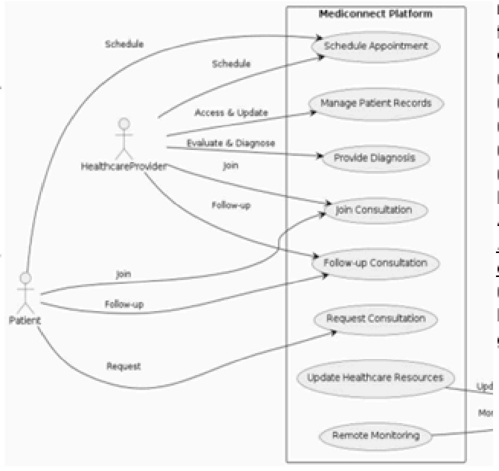
The java is used due to its robust nature and organization support and a big community.

Design Diagrams

1. Activity Diagram:


Mediconnect Platform Activity Diagram
Open Mediconnect App
Patient logs in
yes
New User?
no
Patient registers
Patient enters credentials
Browse available clinicians
yes
Clinician Available?
no
Patient books an appointment
Clinician not available
Appointment scheduled
Patient waits for availability
Connect to clinician via video call
Clinician evaluates patient
yes
Consultation complete?
no
Clinician diagnoses patient
Additional evaluation needed
Provide treatment plan
Request more tests or referrals
Follow-up care arranged

2. Use Case Diagram



Conclusion

The MediConnect platform aims to revolutionize healthcare delivery by providing a secure, efficient, and user-friendly system for managing patient information, doctor appointments, and medical records. By leveraging modern backend technologies such as Java, Spring Boot, and MySQL/PostgreSQL, the platform ensures reliable data management, secure authentication, and seamless interaction between users and healthcare services. The adoption of technologies like RESTful APIs guarantees a secure and scalable solution that can handle the growing needs of healthcare providers and patients.

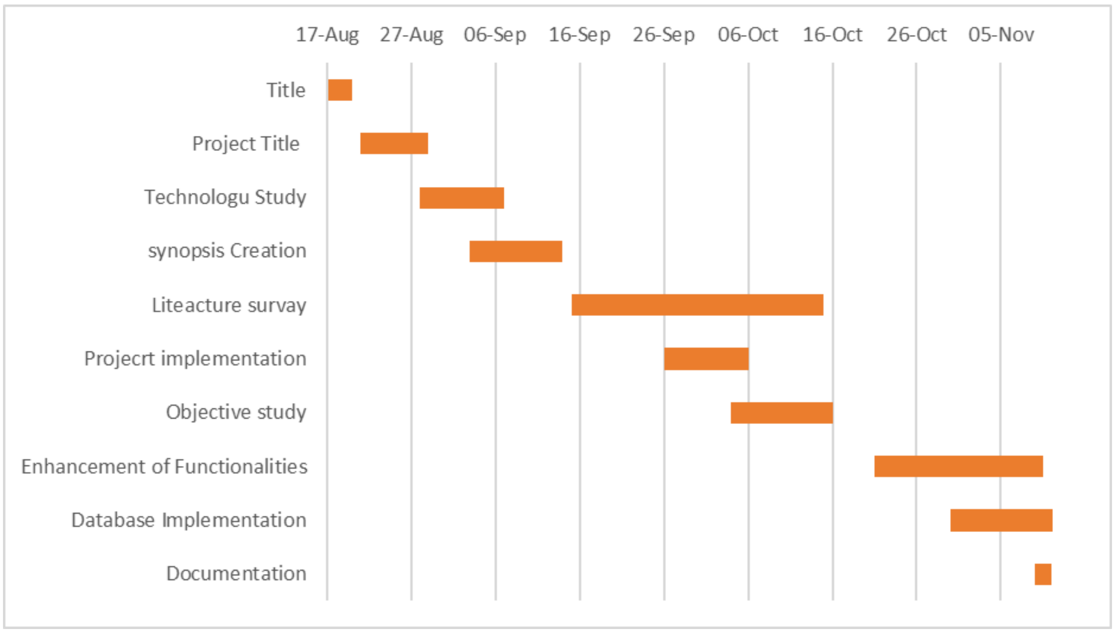
References

This chapter provides a list of key references, documentation, and sources that were instrumental in the development of our MediConnect Platfrom:

(1) https://react.dev/
(2) https://www.tata1mg.co.in/
(3) https://www.practo.in/
(4) https://docs.spring.io/
(5) GeeksforGeeks. (2021). Data Flow Diagrams (DFD) for System AnalysisDesign.GeeksforGeeks.https://www.geeksforgeeks.org/data-flow-diagrams-dfd-fo r-system-analysis-and-design/
(6) A Literature Review: Website Design and User Engagement. guides.himmelfarb.gwu.edu

Project Plan

Gantt Chart



Guide Interaction Sheet

Date	Discussion	Action Plan
17/08/2024	Discussed about the title of the project	Finalized the title MediConnect
21/08/2024	Discussion on the technology to be used for the project	Finalized technologies: Spring, MySQL,Hibernate and REST APIs
28/08/2024	Discussion on the creation of the project synopsis	Gathered information for synopsis creation and outlined project objectives
03/09/2024	Suggestions on literature survey and preliminary research	Read and analyzed research papers; summarized key findings for report inclusion
19/09/2024	Discussion on project implementation approach	Decided to implement core features like user authentication and appointment booking
26/09/2024	Discussion on project objectives	Finalized objectives: seamless patient-doctor communication.
04/10/2024	Suggestions for adding a feature for appointment scheduling	Planned steps for integrating a appointment scheduling functionality
21/10/2024	Discussion on database design for log storage	Decided to store user and appointment logs in the database for analytics
28/10/2024	Discussion on project documentation	Started writing the project documentation and structured it according to format

SOURCE CODE

Models

1. Patient

```
import jakarta.persistence.*;
import lombok.*;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;

@Entity
@AllArgsConstructor
@RequiredArgsConstructor
@Getter
@Setter
@ToString
public class Patient {
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY )
  public int id;
  public String name;
  @Column(unique = true)
  public String email;
  public String phoneNo;
  public String city;
  public String password;
  private Date dob;
  @OneToMany
  private List<Appointment> appointments=new ArrayList<>();
  public void setAppointments(Appointment appointment) {
     appointments.add(appointment);
  }
}
```

2. Doctor

```
import com.telemed24.model.secondarymodel.TimeSlot;
import jakarta.persistence.*;
import lombok.*;
import java.util.ArrayList;
import java.util.List;

@Entity
@AllArgsConstructor
@RequiredArgsConstructor
@Getter
@Setter
@ToString
public class Doctor {
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  public int id;
  public String name;
  @Column(unique = true)
  public String email;
  public String phoneNo;
  public String city;
  public String password;
  private String certificateNo;
  private int rating;
  private String specialization;
  private String address;
  private String modeOfConsultation;
  @OneToMany
  private List<Appointment> appointments=new ArrayList<>();
  @OneToMany
  private List<TimeSlot> appointmentTimeSlots=new ArrayList<>();

  public void setAppointments(Appointment appointment) {
     appointments.add(appointment);
  }
}
```

3. Appointment

```
import jakarta.persistence.*;
import lombok.*;
import java.util.List;

@Entity
@AllArgsConstructor
@RequiredArgsConstructor
@Getter
@Setter
@ToString(exclude = {"patient","doctor"})
public class Appointment {
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private int id;
  @ManyToOne
  private Patient patient;
  @ManyToOne
  private Doctor doctor;
  private String date;
  private String time;
  private String mode;
}
```

4. TimeSlot

```
import com.telemed24.model.Doctor;
import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import java.time.LocalTime;

@Entity
@AllArgsConstructor
@NoArgsConstructor
@Getter
@Setter
public class TimeSlot {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    @ManyToOne
    private Doctor doctor;
    private LocalTime startTime;
    private LocalTime endTime;
    private int currentNoOfPatient;
    private int maxPatientPerSlot;
}
```

Controllers

1. LoginController

```
import com.telemed24.model.Doctor;
import com.telemed24.model.Patient;
import com.telemed24.service.DoctorService;
import com.telemed24.service.PatientService;
import jakarta.servlet.http.Cookie;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import jakarta.servlet.http.HttpSession;
import lombok.RequiredArgsConstructor;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

import java.util.Map;
import java.util.NoSuchElementException;

@RestController
@RequiredArgsConstructor
public class LoginController {
    private final PatientService patientService;
    private final DoctorService doctorService;

    @PostMapping("/login")
    public ResponseEntity<String> login(@RequestBody
Map<String,String> requestBody,
                                        HttpServletRequest request,
                                        HttpServletResponse response) {

        HttpSession session=request.getSession();
        if(session.isNew()) {
            System.out.println("\nUser New Session");
        } else {
            System.out.println("\nUser session already existed");
        }

        System.out.println(requestBody.get("user")+" to login");
        System.out.println("login detail: email =
"+requestBody.get("email")+
                "\n              password = "+requestBody.get("password")+"\n");

        String responseString="login successful";
        try {
            if (requestBody.get("user").equals("PATIENT")) {
                Patient patient =
patientService.findByEmail(requestBody.get("email")).get();
                if
(!requestBody.get("password").equals(patient.getPassword()))
                    responseString = "Incorrect password";
            } else {
                Doctor doctor =
doctorService.findByEmail(requestBody.get("email")).get();
                if
(!requestBody.get("password").equals(doctor.getPassword()))
                    responseString = "Incorrect password";
            }
            session.setAttribute("USER_EMAIL", requestBody.get("email"));
            session.setAttribute("USER_MODE", requestBody.get("user"));
            Cookie cookie = new Cookie("USER_MODE",
requestBody.get("user"));
            response.addCookie(cookie);
        } catch (NoSuchElementException exception) {
            responseString="User Not Found";
        } finally {
            System.out.println(responseString);
            return new ResponseEntity<>(responseString, HttpStatus.OK);
        }
    }
}
```

2. PatientController

```
import com.telemed24.emailservice.EmailServiceImpl;
import com.telemed24.exception.UserWithEmailAlreadyExistException;
import com.telemed24.model.Appointment;
import com.telemed24.model.Patient;
import com.telemed24.service.AppointmentService;
import com.telemed24.service.DoctorService;
import com.telemed24.service.PatientService;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpSession;
import lombok.*;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import java.time.LocalDate;
import java.time.LocalTime;
import java.util.*;

@RestController
@CrossOrigin
@RequiredArgsConstructor
@RequestMapping("/patient")
public class PatientController {
    private final PatientService patientService;
    private final DoctorService doctorService;
    private final EmailServiceImpl mailService;
    private final AppointmentService appointmentService;

    // View Profile
    @GetMapping("/view-profile")
    public ResponseEntity<Patient> viewProfile(HttpServletRequest
request) {
        HttpSession session=request.getSession(false);

        if(session==null) {
            System.out.println("\nPatient is not logged-in Session=
"+session);
```

```
            return new ResponseEntity<>(null, HttpStatus.OK);
        } else   System.out.println("Patient session already existed");

        String email=(String) session.getAttribute("USER_EMAIL");
        Patient patient=patientService.findByEmail(email).get();
        System.out.println("Viewing patient profile: "+email);
        System.out.println("View Profile: " + patient);
        return new ResponseEntity<>(patient,HttpStatus.OK);
    }

    // Logout API
    @GetMapping("/logout")
    public ResponseEntity<String> logout(HttpServletRequest request) {
        System.out.println("\nPatient logout");
        HttpSession session = request.getSession(false); // Get the current
session without creating a new one
        if (session != null) {
            session.invalidate(); // Invalidate the user's session
        }
        return new ResponseEntity<String>("user logout",HttpStatus.OK);
    }

    @PostMapping("/reqOTP")
    public ResponseEntity<String> requestOtp(@RequestBody
Map<String,String> requestBody)
    {
        String to=requestBody.get("email");
        System.out.println("Sending otp to "+to);
        String otp=mailService.sendOtp(to);
        System.out.println("otp = "+otp);  // working

        return new ResponseEntity<String>(otp,HttpStatus.OK);
    }

    @PostMapping("/emailexist")
    public ResponseEntity<String> checkEmailExists(@RequestBody
Map<String,String> request) {
        System.out.println("Checking patient exist in DB :
"+request.get("email"));
        Optional<Patient>
optionalPatient=patientService.findByEmail(request.get("email"));
        System.out.println("dfghdfgdfg");
        if(optionalPatient.isPresent()) {
            throw new UserWithEmailAlreadyExistException();
        }
        else return ResponseEntity.ok("user not exist");
    }

    @PostMapping ("/register")
    public ResponseEntity<String> register(@RequestBody Patient
patient) {
        System.out.println("Storing patient into db");
        System.out.println(patient);

        //One exception may be patient already present with given email
        patientService.register(patient);

        return new ResponseEntity<>("sign-up successful",HttpStatus.OK);
    }


    @AllArgsConstructor
    @NoArgsConstructor
    @ToString
    @Setter
    @Getter
    static class ViewAppointment {
        private String doctorName;
        private LocalDate date;
        private LocalTime time;
        private  String location;
        private String specialization;
    }

//     To extract all appointments of patient
```

```
}

   @GetMapping("/getappointments")
   public ResponseEntity<List<ViewAppointment>>
getAppointments(HttpServletRequest request) {
       List<Appointment> appointments=null;
       List<ViewAppointment> viewAppointments=new ArrayList<>();

       HttpSession session=request.getSession(false);

       if(session==null) {
          System.out.println("\nPatient is not logged-in Session= "+ null);
          return new ResponseEntity<>(null,HttpStatus.OK);
       } else   System.out.println("Patient session already existed");
       String patientEmail=(String) session.getAttribute("USER_EMAIL");
       System.out.println(patientEmail);
       Patient patient=patientService.findByEmail(patientEmail).get();
       System.out.println("Patinet = "+patient);

       appointments=appointmentService.findAllByPatient(patient);
       System.out.println("appointments = "+appointments);
       for(Appointment appointment:appointments) {
          ViewAppointment viewAppointment=new ViewAppointment();

viewAppointment.setDoctorName(appointment.getDoctor().getName());
          viewAppointment.setDate(LocalDate.now());

viewAppointment.setTime(LocalTime.parse(appointment.getTime()));

viewAppointment.setLocation(appointment.getDoctor().getAddress());

viewAppointment.setSpecialization(appointment.getDoctor().getSpeciali
zation());
          viewAppointments.add(viewAppointment);
       }

       return ResponseEntity.ok(viewAppointments);
   }
}
```

3. DoctorController

```
import java.text.ParseException;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import java.util.Optional;

import com.telemed24.emailservice.EmailServiceImpl;
import com.telemed24.exception.UserWithEmailAlreadyExistException;
import com.telemed24.model.Appointment;
import com.telemed24.model.Doctor;
import com.telemed24.model.Patient;
import com.telemed24.model.secondarymodel.TimeSlot;
import com.telemed24.service.AppointmentService;
import com.telemed24.service.DoctorService;
import com.telemed24.service.PatientService;
import com.telemed24.service.TimeSlotService;
import lombok.AllArgsConstructor;
import lombok.RequiredArgsConstructor;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpSession;
```

```
@RestController
@CrossOrigin
@RequiredArgsConstructor
@RequestMapping("/doctor")
public class DoctorController {

   private final DoctorService doctorService;
   private final PatientService patientService;
   private final TimeSlotService timeSlotService;
   private final AppointmentService appointmentService;
   private final EmailServiceImpl mailService;

   @GetMapping("/view-profile")
   public ResponseEntity<Doctor> viewProfile(HttpServletRequest
request) {
      HttpSession session=request.getSession(false);

      if(session==null) {
         System.out.println("\nPatient is not logged-in Session=
"+session);
         return new ResponseEntity<>(null,HttpStatus.OK);
      } else   System.out.println("Patient session already existed");

      String email=(String) session.getAttribute("USER_EMAIL");
      Doctor doctor=doctorService.findByEmail(email).get();
      System.out.println("Viewing doctor profile: "+email);
      System.out.println("View Profile: " + doctor);
      return new ResponseEntity<>(doctor,HttpStatus.OK);
   }

   @GetMapping("/getalldoctors")
   public ResponseEntity<List<Doctor>>
getAllDoctors(HttpServletRequest request) {
      List<Doctor> doctors=doctorService.findAll();
      return new
ResponseEntity<List<Doctor>>(doctors,HttpStatus.OK);
   }
   @GetMapping("/getdoctorby")
   public ResponseEntity<List<ResponseDoctor>>
getDoctorBy(@RequestParam("searchBy") String
searchBy,@RequestParam("value") String value) {
      List<Doctor> doctors=null;
      System.out.println("\nSearching doctor by "+searchBy+"\n value =
"+value);

      if(searchBy.equals("name")) {
         doctors=doctorService.findByName(value);
      } else if(searchBy.equals("city")) {
         doctors=doctorService.findByCity(value);
      } else {
         doctors=doctorService.findBySpecialization(value);
      }

      System.out.println(doctors);
      List<ResponseDoctor> responseDoctors=new ArrayList<>();
      for(Doctor doctor:doctors) {
         ResponseDoctor responseDoctor = new
ResponseDoctor(doctor.getId(), doctor.getSpecialization(),
doctor.getCity(), doctor.getName());
         responseDoctors.add(responseDoctor);
      }
      return new
ResponseEntity<List<ResponseDoctor>>(responseDoctors,HttpStatus.
OK);
   }

   // Get all available Time slots of a particular doctor
   @GetMapping("/getavailableslots")
   public ResponseEntity<List<TimeSlot>>
getAvailableSlots(@RequestParam("doctorId") int doctorId) {
      List<TimeSlot> slots=null;
      System.out.println("\nSearching available slots of doctor
:"+doctorId);

      Doctor doctor=doctorService.findById(doctorId);
```

```
        slots=timeSlotService.extractAvailable(doctor);
        return new ResponseEntity<List<TimeSlot>>(slots,HttpStatus.OK);
    }

    // Booking slot
    @PutMapping("/bookslot")
    public ResponseEntity<String>
bookSlot(@RequestParam("doctorId") String doctorIdString,
                                @RequestParam("slotId") String slotIdString,
                                HttpServletRequest request) {
        int doctorId=Integer.parseInt(doctorIdString);
        int slotId=Integer.parseInt(slotIdString);

        System.out.println("Booking slot of doctor:"+doctorId);

        HttpSession session=request.getSession(false);

        String patientEmail=(String) session.getAttribute("USER_EMAIL");

        System.out.println("Inside slot book");

        TimeSlot slot=timeSlotService.findById(slotId);
        Doctor doctor=doctorService.findById(doctorId);
        Patient patient=patientService.findByEmail(patientEmail).get();

        String doctorName=doctor.getName();
        String patientName=patient.getName();
        String clinicAddress=doctor.getAddress();
        String appointmentTime=slot.getStartTime().toString();

        LocalDate currentDate = LocalDate.now();
        DateTimeFormatter formatter =
DateTimeFormatter.ofPattern("yyyy-MM-dd");
        String appointmentDate = currentDate.format(formatter);

        Appointment appointment=new Appointment();
        appointment.setPatient(patient);
        appointment.setDoctor(doctor);
        appointment.setDate(appointmentDate);
        appointment.setMode("OFFLINE");

        patient.setAppointments(appointment);
        doctor.setAppointments(appointment);

//        timeSlotDao.updateSlot(slotId);

mailService.sendAppointmentConfirmationMail(patientEmail,patientNa
me,doctorName,
                clinicAddress,appointmentDate,appointmentTime);
        return new ResponseEntity<String>("slot booked",HttpStatus.OK);
    }

    @PutMapping("/update")
    public ResponseEntity<String> update(@RequestBody
Map<String,String> request,HttpServletRequest requestSession) throws
ParseException  {
        System.out.println("\n in patient update");
        String name=request.get("name");
        String phoneNo=request.get("phoneNo");
        String city=request.get("city");
        String address=request.get("address");
        String email=(String)
requestSession.getSession().getAttribute("USER_EMAIL");

        Doctor updatedDoctor=new Doctor();
        updatedDoctor.setName(name);
        updatedDoctor.setPhoneNo(phoneNo);
        updatedDoctor.setCity(city);
        updatedDoctor.setAddress(address);
        doctorService.update(email,updatedDoctor);
        return new ResponseEntity<String>("patient
update",HttpStatus.OK);
    }

    @GetMapping("/getappointments")
    public ResponseEntity<List<Appointment>>
getAppointments(HttpServletRequest request) {
        List<Appointment> appointments=null;
        HttpSession session=request.getSession(false);
        String doctorEmail=(String) session.getAttribute("USER_EMAIL");

        Doctor doctor=doctorService.findByEmail(doctorEmail).get();
        appointments=appointmentService.findAllByDoctor(doctor);

        return new
ResponseEntity<List<Appointment>>(appointments,HttpStatus.OK);
    }

    @GetMapping("/logout")
    public ResponseEntity<String> logout(HttpServletRequest request) {
        System.out.println("\nDoctor logout");
        HttpSession session=request.getSession(false);
        if (session != null) {
            session.invalidate(); // Invalidate the user's session
        }
        return new ResponseEntity<>("user logout",HttpStatus.OK);
    }

    @PostMapping("/reqOTP")
    public ResponseEntity<String> requestOtp(@RequestBody
Map<String,String> requestBody)
    {
        String to=requestBody.get("email");
        System.out.println("Sending otp to "+to);
        String otp=mailService.sendOtp(to);
        System.out.println("otp = "+otp);  // working

        return new ResponseEntity<String>(otp,HttpStatus.OK);
    }
    @PostMapping("/register")
    public ResponseEntity<Doctor> register(@RequestBody Doctor
doctor) {
        System.out.println("Storing patient into db");
        System.out.println(doctor);

        //One exception may be doctor already present with given email
        doctorService.register(doctor);
        return new ResponseEntity<>(doctor,HttpStatus.OK);
    }

    @PostMapping("/emailexist")
    public ResponseEntity<String> checkEmailExists(@RequestBody
Map<String,String> doctor) {
        System.out.println("Checking doctor exist in DB :
"+doctor.get("email"));
        Optional<Doctor>
optionalDoctor=doctorService.findByEmail(doctor.get("email"));
        System.out.println("dfgegwerg");
        if(optionalDoctor.isPresent())
            throw new UserWithEmailAlreadyExistException();
        return ResponseEntity.ok("user not exist");
    }
    @PostMapping("/addtimeslot")
    public ResponseEntity<String> addTimeSlot(@RequestBody
List<TimeSlot> slots,@RequestParam("email") String email) {
        System.out.println("inside add time slot");
        timeSlotService.addSlot(slots,email);
        return ResponseEntity.ok("slots added");
    }
}
```