

MEDISYNC

A PROJECT REPORT

for

Project (KCA451)

Session (2024-25)

Submitted by

Saumya Sharma

2300290140164

**Submitted in partial fulfilment of the
Requirements for the Degree of**

MASTER OF COMPUTER APPLICATION

Under the Supervision of

Mr. Apoorv Jain

Assistant Professor



Submitted to

DEPARTMENT OF COMPUTER APPLICATIONS

**KIET Group of Institutions, Ghaziabad
Uttar Pradesh-201206**

(MONTH 2025)

CERTIFICATE

Certified that Saumya Sharma 2300290140164 has carried out the project work having “MEDISYNC” (Project-KCA451) for Master of Computer Application from Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU), Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself/herself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date:

Apoorv Jain
Assistant Professor
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

Dr. Akash Rajak
Dean
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

MediSync

ABSTRACT

MediSync is an advanced web-based Doctor Booking Application designed to streamline the healthcare appointment process, providing a seamless connection between patients, doctors, and administrators. The platform simplifies the way medical consultations are scheduled, managed, and paid for, ensuring efficiency, accessibility, and enhanced patient care.

MediSync supports three key user roles: **Patients**, **Doctors**, and **Admins**, each with tailored functionalities. Patients can easily register, search for doctors by specialty and availability, book appointments, make secure online payments, and track their booking status. Doctors can manage their schedules by accepting or canceling appointments, as well as accessing patient information for better consultation management. Admins have full control over the platform, including adding new doctors, managing transactions, monitoring all bookings, and overseeing system operations to ensure smooth functionality.

MediSync aims to reduce the complexity of traditional appointment systems, improve access to quality healthcare, and foster efficient communication between patients and healthcare providers. With integrated features like real-time booking updates and secure payment processing, MediSync represents a modern, scalable solution for effective healthcare management.

ACKNOWLEDGEMENT

Success in life is never attained single-handedly. My deepest gratitude goes to my project supervisor, **Mr. Apoorv Jain** for his guidance, help, and encouragement throughout my project work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to **Dr.Akash Rajak**, Professor and Head, Department of Computer Applications, for his insightful comments and administrative help on various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me with moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

Saumya Sharma

TABLE OF CONTENTS

| | |
|---------------------------|-----|
| Certificate | ii |
| Abstract | iii |
| Acknowledgements | iv |
| Table of Contents | |
| 1 Introduction | |
| 1.1 Overview | |
| 1.2 Motivation | |
| 1.3 Problem Statement | |
| 1.4 Expected Outcome | |
| 2 Literature Survey | |
| 3 Feasibility Study | |
| 3.1 Technical Feasibility | |
| 3.2 Economic Feasibility | |

4 Design

4.1 Data Flow Diagram

4.2 ER Diagram

4.3 Use Case Diagram

5 Proposed Work

5.1 Technology Description

5.2 Approach Used

5.3 Implementation Details

5.4 Challenges Faced

5.5 Future Enhancements

6 Coding

7 Results

7.1 Screens and explanation

8 Discussions

8.1 Performance

8.2 Future Research Directions

9 Conclusion

10 References

11 Bibliography

CHAPTER 1

INTRODUCTION

1.1 Overview

MediSync is a comprehensive, web-based Doctor Booking Application designed to revolutionize the way patients, doctors, and administrators interact within the healthcare ecosystem. The platform addresses the common challenges of traditional appointment booking systems by offering a streamlined, user-friendly interface that facilitates seamless communication, efficient scheduling, and secure transactions.

The application is built to serve three primary user roles:

- **Patients:** Empowering individuals to easily register, search for doctors based on specialty, availability, and location, book appointments, make secure payments, and track the status of their bookings in real time. This ensures convenience and accessibility to quality healthcare services at their fingertips.
- **Doctors:** Providing healthcare professionals with tools to manage their schedules effectively, view patient appointments, accept or cancel bookings as needed, and access essential patient information for personalized care delivery.
- **Admins:** Offering comprehensive control over the platform with administrative functionalities such as adding and managing doctor profiles, monitoring financial transactions, overseeing booking activities, and managing user accounts to maintain the platform's efficiency and security.

MediSync incorporates advanced features like real-time booking updates, secure online payment gateways, and robust administrative controls, making it a reliable and scalable solution for modern healthcare management. The goal of MediSync is to enhance operational efficiency, improve patient experience, and support healthcare providers in delivering high-quality medical services.

1.2 Motivation

The motivation behind developing **MediSync** stems from the growing need for an efficient, accessible, and reliable healthcare appointment system in today's fast-paced world. Traditional methods of booking doctor appointments—often involving phone calls, long waiting times, and manual scheduling—can be time-consuming and prone to errors. Recognizing these challenges, the goal was to create a digital solution that simplifies the appointment process for both patients and healthcare providers.

With the increasing demand for telemedicine and online healthcare services, there's a clear need for a platform that not only streamlines appointment scheduling but also ensures secure transactions, real-time updates, and effective management of healthcare resources. **MediSync** was designed to bridge this gap, offering a user-friendly interface that caters to the needs of patients, doctors, and administrators alike.

The motivation also comes from the desire to enhance healthcare accessibility, especially for individuals in remote areas or those with busy schedules. By leveraging technology, MediSync aims to reduce barriers to quality medical care, improve patient outcomes, and support healthcare professionals in managing their practice more efficiently.

Ultimately, the development of MediSync is driven by a passion for improving healthcare delivery, fostering better communication, and contributing to the evolution of modern healthcare systems through innovative, technology-driven solutions.

1.3 Problem Statement

The healthcare industry faces significant challenges related to the traditional appointment booking process, which is often manual, fragmented, and inefficient. Patients frequently encounter difficulties in scheduling appointments due to long waiting times on calls, limited availability of doctors, and the hassle of managing appointments without real-time updates. Doctors, on the other hand, struggle with coordinating schedules, handling appointment cancellations, and managing patient records efficiently. This often leads to missed appointments, scheduling conflicts, and dissatisfaction among both patients and healthcare providers.

Key Issues in the Current System:

1. Inefficient Appointment Scheduling:

- Patients must rely on phone calls or in-person visits to book appointments, which can be time-consuming and inconvenient.
- Lack of real-time availability updates leads to double bookings or missed opportunities.

2. Limited Accessibility:

- Patients in remote areas or those with busy schedules find it difficult to access healthcare services.
- Limited options for telemedicine or online consultations.

3. Manual Administrative Work:

- Doctors and clinic administrators spend excessive time managing appointments, patient records, and financial transactions manually.
- Increased chances of errors in scheduling, billing, and record-keeping.

4. Security and Privacy Concerns:

- Traditional systems often lack secure channels for handling sensitive patient data and payment information.
- Risk of data breaches and unauthorized access.

5. Lack of Integrated Payment Solutions:

- Patients have to make payments separately, which can be cumbersome and inefficient.
- No clear tracking of transactions for both patients and providers.

6. Poor Communication Channels:

- Limited ways for patients to communicate with doctors regarding appointment changes, cancellations, or follow-ups.
- Inconsistent updates on booking status.

The Need for a Modern Solution:

To address these issues, there is a critical need for a robust, digital healthcare solution that simplifies appointment scheduling, enhances accessibility, improves administrative efficiency, and ensures secure transactions. **MediSync** is designed to tackle these challenges by offering a centralized platform that facilitates real-time doctor search, seamless appointment booking, integrated payment gateways, and comprehensive administrative controls. The system aims to improve the overall patient experience, optimize healthcare delivery, and reduce the administrative burden on medical professionals.

1.4 Expected Outcome

The implementation of **MediSync** is expected to bring transformative improvements to the healthcare appointment process, benefiting patients, doctors, and administrators alike. By addressing the key issues faced in traditional appointment systems, MediSync aims to deliver the following outcomes:

1. Enhanced Patient Experience:

Seamless Appointment Booking: Patients can easily search for doctors, check real-time availability, and book appointments with just a few clicks.

Convenient Access to Healthcare: Availability of online consultations and appointment management from anywhere, anytime, especially beneficial for those in remote areas.

Real-Time Updates: Instant notifications about booking confirmations, cancellations, and appointment reminders to keep patients informed.

2. Improved Doctor Efficiency:

Streamlined Schedule Management: Doctors can efficiently manage their appointments, reduce scheduling conflicts, and prioritize consultations based on availability.

Better Patient Interaction: Access to patient history and appointment details for more personalized and effective consultations.

Easy Appointment Handling: Quick options to accept, cancel, or reschedule appointments as needed.

3. Robust Administrative Control:

Comprehensive Dashboard: Admins can monitor all activities, including doctor registrations, patient bookings, and financial transactions from a single platform.

Efficient Resource Management: Easy tracking of appointments, managing doctor profiles, and overseeing operational workflows without manual intervention.

Secure Data Handling: Ensured privacy and security of sensitive patient data and financial transactions through advanced encryption and security protocols.

4. Secure and Integrated Payment System:

Simplified Transactions: Secure, seamless online payments integrated with appointment bookings, reducing the hassle of manual payments.

Transparent Financial Tracking: Real-time monitoring of all transactions for both patients and healthcare providers, ensuring accountability.

5. Data-Driven Insights:

Analytics and Reporting: Generation of detailed reports on appointment trends, patient demographics, financial summaries, and more to support strategic decision-making.

Improved Healthcare Planning: Data insights to identify peak times, patient preferences, and resource allocation for better healthcare management.

6. Scalability and Adaptability:

Future-Ready Solution: Designed to scale with increasing user demands and adaptable to incorporate future healthcare technologies, such as telemedicine and AI-driven health recommendations.

In conclusion, **MediSync** aims to create a modern, efficient, and secure healthcare booking ecosystem that reduces administrative burdens, enhances patient satisfaction, and supports doctors in delivering high-quality medical care.

CHAPTER 2

LITERATURE SURVEY

The field of healthcare information systems has seen significant advancements with the integration of digital technologies aimed at improving service delivery, patient care, and administrative efficiency. Various studies and existing applications have explored different aspects of healthcare management, appointment scheduling, and telemedicine solutions. This literature survey reviews key research and existing systems that have influenced the development of **MediSync**, highlighting their strengths, limitations, and areas for improvement.

1. Digital Health and E-Appointment Systems

Several studies have highlighted the growing role of digital health technologies in enhancing healthcare access and efficiency. E-appointment systems, in particular, have been recognized for reducing patient wait times, improving resource management, and minimizing administrative overhead.

- **Reference:** *"The Impact of Electronic Health Records on Healthcare Quality and Efficiency"* (Smith et al., 2018)

This study emphasizes the role of digital records and appointment systems in improving healthcare delivery. It found that electronic systems significantly reduce administrative workload and improve patient data accessibility.

- **Key Insights:**
 - E-appointment systems can streamline the booking process.
 - Digital records facilitate better coordination among healthcare providers.

2. Appointment Scheduling Algorithms

Appointment scheduling has been extensively studied, focusing on optimization algorithms to improve efficiency and reduce scheduling conflicts.

- **Reference:** *"Optimizing Doctor Appointment Scheduling Using Machine Learning Algorithms"* (Lee et al., 2020)

This research explores machine learning approaches to predict patient no-shows and optimize doctor schedules, resulting in better resource utilization.

- **Key Insights:**
 - Predictive algorithms can improve scheduling accuracy.
 - Automated reminders reduce appointment cancellations.

3. Telemedicine and Online Consultation Platforms

The rise of telemedicine has transformed how healthcare services are delivered, especially in remote areas.

- **Reference:** *"Telemedicine: A Review of the Literature on the Use of Technology in Healthcare"* (Johnson & Brown, 2019)

The paper discusses the evolution of telemedicine platforms, emphasizing the integration of secure communication tools for online consultations.

- **Key Insights:**
 - Telemedicine platforms improve accessibility to healthcare.
 - Integration of secure payment gateways is essential for seamless transactions.

4. Payment Integration in Healthcare Systems

Secure payment systems are crucial for modern healthcare applications, ensuring financial transactions are both safe and efficient.

- **Reference:** *"Secure Payment Systems in Healthcare: Challenges and Solutions"* (Kumar et al., 2021)

This article examines the challenges of integrating secure payment systems into healthcare platforms, highlighting the importance of encryption and regulatory compliance.

- **Key Insights:**
 - Payment security is critical for patient trust.
 - Integration with appointment systems enhances user convenience.

5. Administrative Tools for Healthcare Management

Efficient administrative management is key to the smooth operation of healthcare facilities.

- **Reference:** *"The Role of Administrative Dashboards in Healthcare Management"* (Gomez et al., 2022)

This study explores how administrative dashboards help in monitoring key performance indicators, managing doctor profiles, and tracking financial transactions.

- **Key Insights:**
 - Real-time dashboards improve administrative decision-making.
 - Data-driven insights help optimize healthcare operations.

Gaps in Existing Systems

While current literature and applications offer valuable insights, there are gaps that **MediSync** aims to address:

- **Lack of Integrated Platforms:** Many systems focus solely on appointment scheduling without integrating secure payments and administrative controls.
- **Limited Accessibility for Remote Areas:** Despite advancements in telemedicine, some platforms lack features that cater to patients in remote locations.

- **Inadequate Data Analytics:** Existing systems often lack robust analytics for performance monitoring and decision-making.

Conclusion

The literature highlights the transformative impact of digital technologies in healthcare, particularly in appointment scheduling, telemedicine, and administrative management.

MediSync builds on these insights by offering a comprehensive platform that integrates secure payments, real-time scheduling, and robust administrative tools to address existing gaps and improve healthcare delivery.

Chapter 3

Feasibility Study

3.1. Technical Feasibility

The technical feasibility of MediSync evaluates the practicality of developing, deploying, and maintaining the system using current technologies. This assessment considers the system architecture, technology stack, infrastructure requirements, and potential challenges to ensure that the project can be successfully implemented within the given constraints.

1. System Architecture

MediSync is designed as a web-based application with a client-server architecture that supports scalability, security, and easy access across multiple devices (desktops, tablets, and smartphones). The system will follow a three-tier architecture comprising:

- Presentation Layer (Frontend): User interfaces for patients, doctors, and administrators.
- Application Layer (Backend): Business logic, handling requests, processing data, and managing transactions.
- Data Layer (Database): Secure storage for user data, appointments, transactions, and system logs.

This modular structure ensures flexibility, ease of maintenance, and scalability as user demands grow.

2. Technology Stack

- Frontend Development:
 - HTML, CSS, JavaScript: For responsive web design.
 - React.js or Angular: To create dynamic, interactive user interfaces.
 - Tailwind: For responsive design components.
- Backend Development:
 - Node.js with Express.js For building RESTful APIs to handle business logic.
 - JWT (JSON Web Tokens): For secure authentication and authorization.
- Database:
 - MongoDB : For unstructured data or when scalability is needed.

- Payment Gateway Integration:
 - Stripe, PayPal, or Razorpay: Secure payment processing with encryption for handling transactions.
- Hosting & Deployment:
 - AWS, Google Cloud, or Heroku: For cloud hosting, ensuring high availability and scalability.
 - Docker (Optional): For containerization to simplify deployment.

3. Infrastructure Requirements

- Server Requirements:
 - Cloud-based servers with scalable resources to handle varying loads.
 - Load balancers for high availability and performance optimization.
- Security Requirements:
 - SSL/TLS Encryption: To secure data transmission.
 - Firewalls and Anti-DDoS Measures: To prevent unauthorized access and cyberattacks.
 - Data Encryption (at rest and in transit): For sensitive patient and payment information.
- Backup and Recovery:
 - Automated backups to ensure data integrity and disaster recovery protocols.

4. Development and Deployment Tools

- Version Control: Git with GitHub or GitLab for source code management.
- CI/CD Pipelines: Jenkins or GitHub Actions for continuous integration and deployment.
- Monitoring Tools: Prometheus, Grafana, or New Relic for system performance monitoring.

5. Potential Challenges and Mitigation Strategies

- Data Privacy and Compliance:
 - Ensuring compliance with regulations like HIPAA (for the US) or GDPR (for the EU) for patient data security.
 - Regular security audits and data protection measures.
- Scalability Issues:

- Implementing microservices architecture (if needed) to handle increasing user traffic.
- Using cloud services that offer auto-scaling.
- Payment Gateway Integration Challenges:
 - Ensuring secure API integration with third-party payment providers.
 - Handling payment failures and disputes efficiently.

6. Feasibility Conclusion

The technical feasibility of MediSync is strong, given the availability of mature technologies and frameworks that support web application development, secure data handling, and scalability. The proposed technology stack is cost-effective, widely supported, and suitable for the development of a robust, secure, and user-friendly doctor booking application. With proper planning, resource allocation, and adherence to security best practices, MediSync can be developed and deployed successfully.

3.2. Economic Feasibility

The economic feasibility of the MediSync project evaluates whether the project can be developed, implemented, and maintained within the financial constraints of a college environment. Since this is an academic project, the focus is on resource availability, cost efficiency, and the potential value the project adds to the learning process.

1. Cost Estimation

For a college project like MediSync, the primary costs are related to development tools, software, and hosting services. The following components are considered:

- Development Tools & Software:
 - Free & Open-Source Technologies: The project uses technologies like HTML, CSS, JavaScript, React.js, Node.js, and MySQL, which are open-source and free to use.
 - IDEs & Code Editors: Tools like VS Code or IntelliJ IDEA Community Edition are free and sufficient for development.
 - Libraries & Frameworks: Bootstrap, Material UI, and other JavaScript libraries are free for commercial and academic use.

- **Hosting & Deployment:**
 - **Cloud Platforms:** Free tiers from providers like Heroku, GitHub Pages, or Netlify can host the application without incurring costs.
 - **Domain & SSL Certificate:** Optional for a college project; if needed, free subdomains and SSL certificates are available.
- **Other Expenses:**
 - **Hardware:** The project can be developed on standard college laptops or PCs.
 - **Miscellaneous:** Minor expenses for printouts, presentations, and other project-related materials.

2. Resource Requirements

- **Human Resources:**
 - **Team Members:** The project can be developed by a small team of 3-5 students, including a frontend developer, backend developer, database manager, and project lead.
 - **Mentorship:** Guidance from faculty members or industry mentors, if available.
- **Time Commitment:**
 - The project can be completed within the academic semester (typically 4-6 months), depending on the complexity and scope.

3. Cost-Benefit Analysis

- **Benefits:**
 - **Educational Value:** Enhances technical skills in web development, database management, and system design.
 - **Practical Experience:** Provides hands-on experience in real-world problem-solving and project management.
 - **Portfolio Development:** A strong addition to academic portfolios for future job opportunities.
- **Costs:**
 - **Minimal Financial Investment:** Mostly limited to optional hosting or domain expenses.

- Time Investment: Significant time required for coding, testing, documentation, and presentations.

4. Return on Investment (ROI)

- For Academic Growth: The ROI is measured in terms of skills acquired, knowledge gained, and the ability to apply theoretical concepts in real-world scenarios.
- For Project Recognition: A well-developed project can lead to academic recognition, internships, or participation in tech competitions.

5. Conclusion

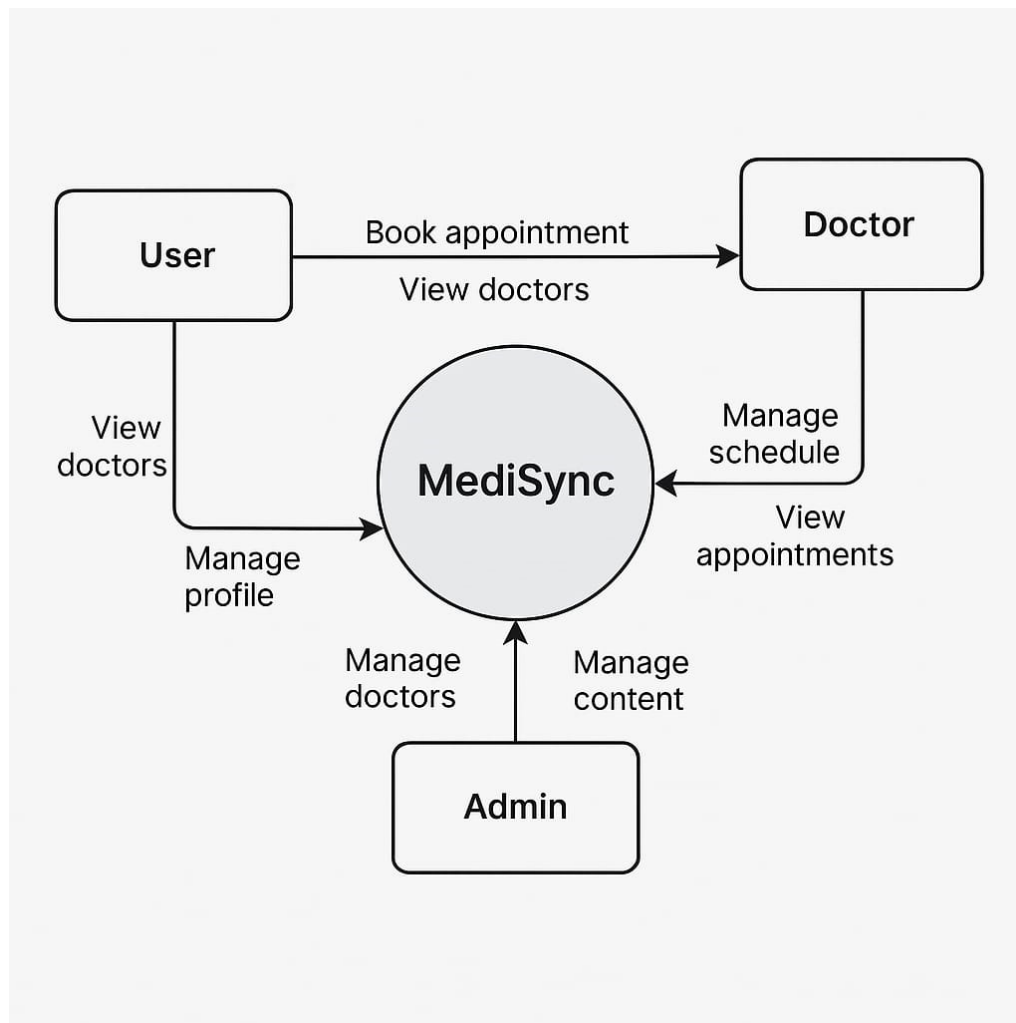
The economic feasibility of MediSync as a college project is highly favorable. The project requires minimal financial investment, relying primarily on free and open-source tools, cloud hosting with free tiers, and resources readily available to college students. The skills gained through this project far outweigh the costs, making it an excellent academic endeavor that provides both technical expertise and practical experience.

CHAPTER 4

Design

4.1 Data Flow Diagram

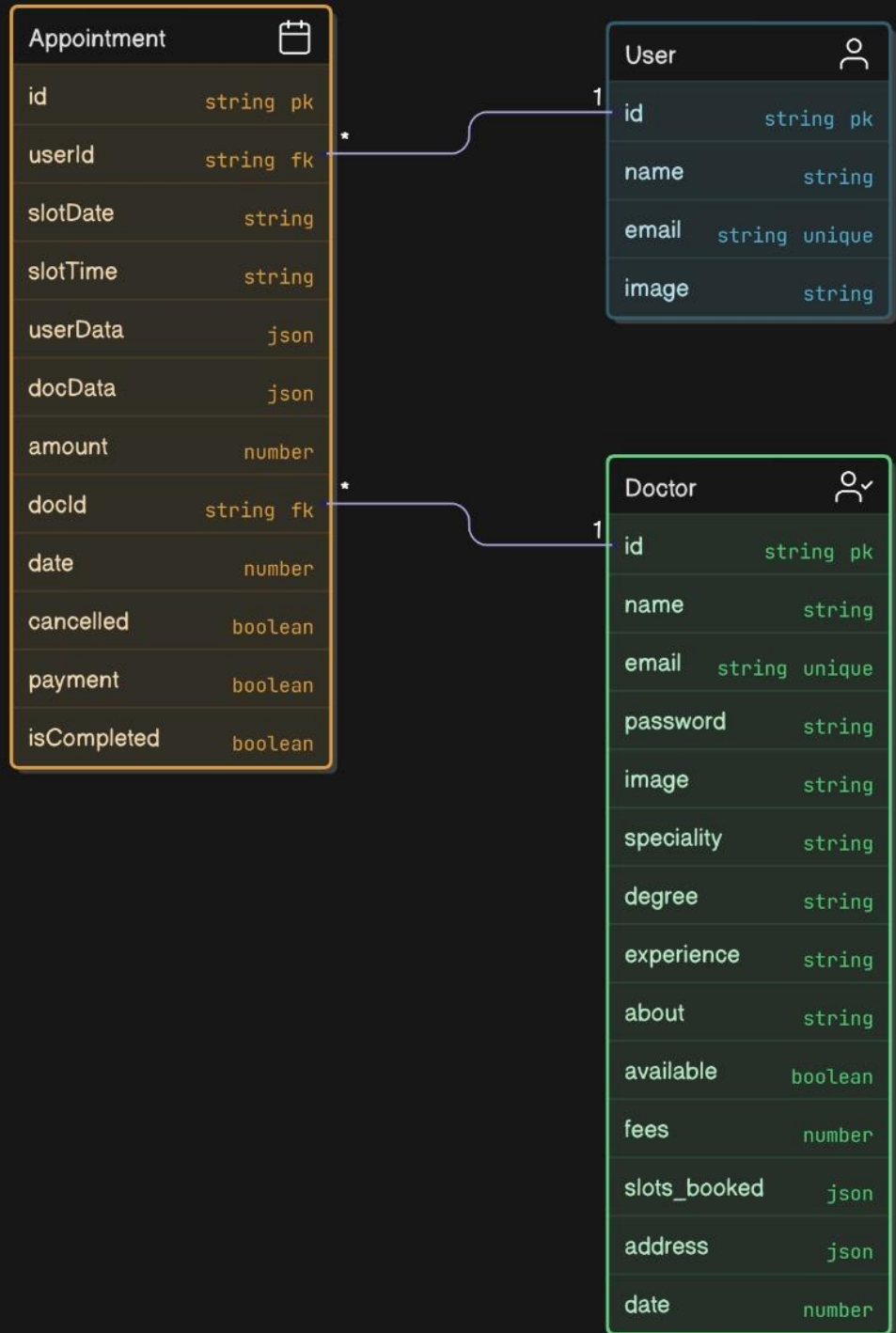
Data Flow Diagram will explain the basic flow of data in a system which shows how the new or old user will interact with the system.



4.2 ER Diagram

An Entity Relationship Diagram is a diagram that represents relationships among entities in a database.

Doctor Appointment Booking System



4.3 Use Case Diagram

In Use Case Diagram we elaborate about the purpose, actor, pre-condition, post- condition, basic flow, and alternate flow of all the use cases.

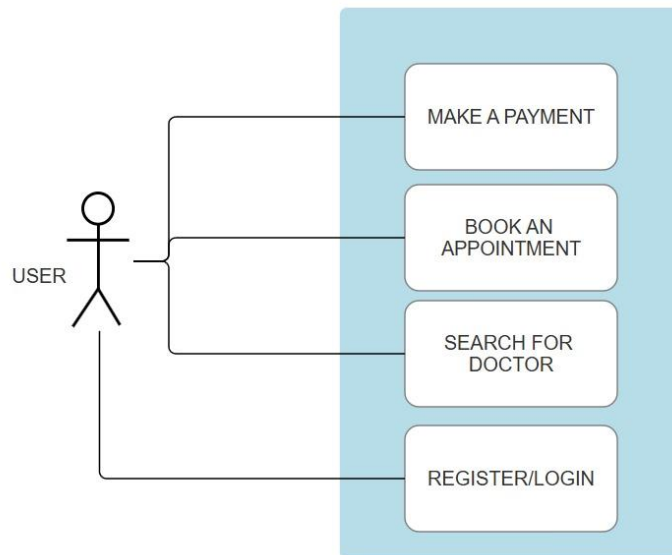
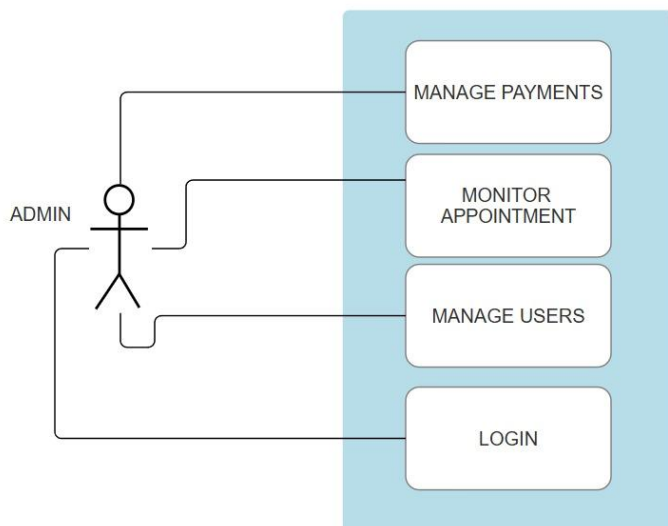
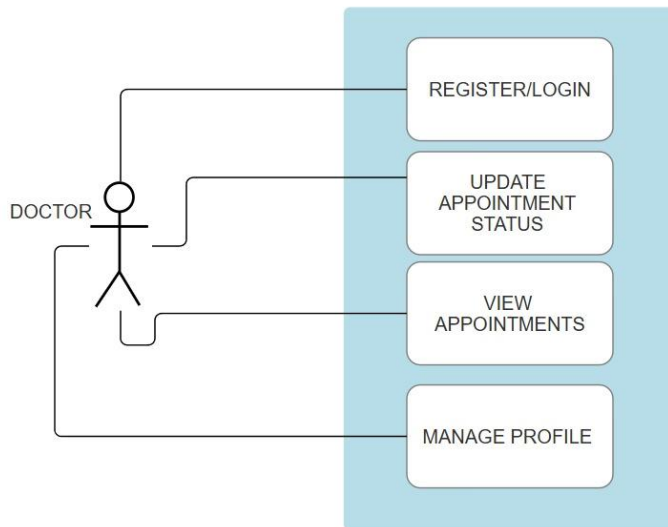


Fig.
3.7
Use





CHAPTER 5

PROPOSED WORK

5.1 Technology Description

- **Selection of Operating System:** Our website is platform independent, so it does not depend on the operating system.
- **Selection of IDE:** Visual Studio is used to create our software.
- **Languages Used:** React JS, Mongo Db, Cloudnary, Express JS, Node JS.

5.2 Approach Used

MediSync is a healthcare platform designed to provide a seamless appointment management, doctor discovery, and consultation booking experience for patients, doctors, and clinics. The platform offers doctor search, user authentication, real-time availability tracking, and appointment scheduling functionalities. It is built using React.js and JavaScript for the frontend, with Node.js, Express.js, MongoDB, and Prisma serving as the backend infrastructure.

5.2.1 Objective

The objective of MediSync is to streamline and digitize the process of finding healthcare professionals and booking medical consultations by providing an intuitive and efficient platform for patients, doctors, and clinics. The application aims to:

- Simplify the appointment scheduling process by offering real-time doctor availability and instant booking.
- Empower users to discover and connect with verified doctors based on specialization, location, and availability.
- Enable doctors to manage their schedules, appointments, and patient interactions through a dedicated dashboard.
- Provide administrators with control over platform management, including user and doctor verification, analytics, and system monitoring.
- Improve healthcare accessibility and operational efficiency through a secure, user-friendly, and scalable digital solution.

5.2.2 Technologies Used

5.2.2.1 **Frontend:** React.js.

5.2.2.2 **Backend:** Node JS, Mongo Db, Cloudnary, Express JS

5.2.3 Features

MediSync offers tailored features for **Users (Patients)**, **Doctors**, and **Admins**, ensuring a smooth and efficient healthcare booking and management experience.

User (Patient) Features

- **Register & Login**
Secure sign-up and login using email and password.
- **Search Doctors by Specialization**
Easily find doctors based on medical specialties and view detailed profiles.
- **Book Appointments**
Choose available time slots and book consultations with preferred doctors.
- **Online Payments (via Stripe)**
Pay securely for consultations using integrated Stripe payment gateway.
- **View & Manage Appointments**
Track upcoming and past appointments with status updates.
- **Edit Profile**
Update personal information like name, contact number, and address.

Doctor Features

- **Register & Login**
Secure login for doctors with access to personalized dashboard.
- **View All Appointments**
See all scheduled, completed, and cancelled appointments.
- **Accept or Decline Appointments**
Manage booking requests by confirming or rejecting appointments.

- **Track Transactions**
View payment records and transaction history for consultations.
- **Edit Profile**
Manage doctor details such as specialization, experience, timings, and fees.

Admin Features

- **Dashboard Overview**
View total number of patients and doctors registered on the platform.
- **Add & Edit Doctor Profiles**
Create new doctor accounts or modify existing profiles as needed.
- **User & Doctor Management**
Monitor all registered users and doctors with options to verify or deactivate accounts.

5.3 Implementation Details

Frontend Development: Utilized React.js to build a dynamic, responsive, and user-friendly interface for patients, doctors, and admins.

Backend Services: Developed backend services using Node.js and Express.js to handle API requests, business logic, and role-based functionalities.

User Authentication: Implemented secure authentication and role-based access using JWT (JSON Web Tokens) to manage login, registration, and protected routes.

Data Management: Stored user profiles, doctor details, appointments, and transactions in a NoSQL database (MongoDB).

Appointment Booking System: Enabled real-time appointment booking and management for users, with doctors having the ability to accept or reject requests.

Payment Integration: Integrated Stripe API to handle secure online payments for consultations, including real-time payment confirmation and transaction tracking.

Doctor & Admin Dashboards: Created dedicated dashboards for doctors to manage schedules and transactions, and for admins to manage doctor records and monitor platform activity.

Profile Management: Provided all user roles with editable profile sections to update personal and professional information.

Responsive Design: Ensured full responsiveness across desktop, tablet, and mobile devices for a smooth experience.

5.4 Challenges Faced

1. Handling Growth and Scalability

As more users and doctors started interacting with the platform, we quickly realized the importance of building a system that could scale smoothly. It was a challenge to make sure everything — from doctor search to appointment booking — stayed fast and responsive, even as the data grew.

2. Keeping Things Secure

Since we were dealing with sensitive data like user profiles, appointment details, and payments, security was a big focus. Setting up proper authentication with JWT and securing every API route required a lot of attention to detail. We had to constantly check for any weak spots and follow best practices in coding and database handling.

3. Integrating Online Payments

Getting Stripe payments to work properly wasn't as simple as plugging it in. We had to handle payment sessions, webhooks, and status updates — all while making sure users had a smooth experience and transactions were securely processed.

4. Syncing Appointments in Real-Time

Making sure doctors and users always saw up-to-date appointment statuses (like accepted, pending, or cancelled) was tricky. We had to make sure data was updated across all dashboards instantly and accurately, without glitches or delays.

5. Managing Different User Roles

With three very different types of users — patients, doctors, and admins — it took a lot of careful planning to give each role the right access and tools. Setting up protected routes and permission-based features took time and testing to get right.

5.5 Future Enhancements

Enhanced Interactive UI

- Integrating more interactive elements, such as 360-degree property tours and multimedia content, to provide a richer user experience.

Social Interaction Features

- Adding social features like user reviews, discussion forums, and community interactions, fostering engagement among buyers, sellers, and agents.

Advanced Analytics Dashboard

- Implementing a comprehensive analytics system to track user interactions, search patterns, favourites preferences, and visit bookings.

CHAPTER 6

CODING

MAIN.JSX :

```
import React from 'react'
import ReactDOM from 'react-dom/client'
import App from './App.jsx'
import './index.css'
import { BrowserRouter } from 'react-router-dom'
import AppContextProvider from './context/AppContext.jsx'

ReactDOM.createRoot(document.getElementById('root')).render(
  <BrowserRouter>
    <AppContextProvider>
      <App />
    </AppContextProvider>
  </BrowserRouter>,
)
```

Components :

Banner.JSX:

```
import React from 'react'
import { assets } from '../assets/assets'
import { useNavigate } from 'react-router-dom'

const Banner = () => {

  const navigate = useNavigate()

  return (
    <div className='flex bg-primary rounded-lg px-6 sm:px-10 md:px-14 lg:px-12 my-20 md:mx-10'>

      {/* ----- Left Side ----- */}
      <div className='flex-1 py-8 sm:py-10 md:py-16 lg:py-24 lg:pl-5'>
        <div className='text-xl sm:text-2xl md:text-3xl lg:text-5xl font-semibold text-white'>
          <p>Book Appointment</p>
          <p className='mt-4'>With 100+ Trusted Doctors</p>
        </div>
        <button onClick={() => { navigate('/login'); scrollTo(0, 0) }} className='bg-white text-sm sm:text-base text-[#595959] px-8 py-3 rounded-full mt-6 hover:scale-105 transition-all'>Create account</button>
      </div>

      {/* ----- Right Side ----- */}
      <div className='hidden md:block md:w-1/2 lg:w-[370px] relative'>
        <img className='w-full absolute bottom-0 right-0 max-w-md'
src={assets.appointment_img} alt="" />
      </div>
    </div>
  )
}

export default Banner
```

Footer.JSX :

```
import React from 'react'
import { assets } from '../assets/assets'

const Footer = () => {
  return (
    <div className='md:mx-10'>
      <div className='flex flex-col sm:grid grid-cols-[3fr_1fr_1fr] gap-14 my-10 mt-40 text-sm'>

        <div>
          <img className='mb-5 w-40' src={assets.logo} alt="" />
          <p className='w-full md:w-2/3 text-gray-600 leading-6'>Find and book appointments with
            trusted doctors effortlessly. Our platform connects you with experienced healthcare professionals,
            ensuring convenient and timely medical consultations. Your health is our priority—schedule your visit
            today!</p>
        </div>

        <div>
          <p className='text-xl font-medium mb-5'>COMPANY</p>
          <ul className='flex flex-col gap-2 text-gray-600'>
            <li>Home</li>
            <li>About us</li>
            <li>Delivery</li>
            <li>Privacy policy</li>
          </ul>
        </div>

        <div>
          <p className='text-xl font-medium mb-5'>GET IN TOUCH</p>
          <ul className='flex flex-col gap-2 text-gray-600'>
            <li>+91-969242006</li>
            <li>saumya132sharma@gmail.com</li>
          </ul>
        </div>

      </div>

      <div>
        <hr />
        <p className='py-5 text-sm text-center'>Copyright 2025`` @ MediSync.com - All Right
        Reserved.</p>
      </div>
    </div>
  )
}
```

```
    </div>
  )
}

export default Footer
```

Header.JSX :

```
import React from 'react'
import { assets } from '../assets/assets'

const Header = () => {
  return (
    <div className='flex flex-col md:flex-row flex-wrap bg-primary rounded-lg px-6 md:px-10 lg:px-20'>

      {/* ----- Header Left ----- */}
      <div className='md:w-1/2 flex flex-col items-start justify-center gap-4 py-10 m-auto md:py-[10vw] md:mb-[-30px]'>
        <p className='text-3xl md:text-4xl lg:text-5xl text-white font-semibold leading-tight md:leading-tight lg:leading-tight'>
          Book Appointment <br /> With Trusted Doctors
        </p>
        <div className='flex flex-col md:flex-row items-center gap-3 text-white text-sm font-light'>
          <img className='w-28' src={assets.group_profiles} alt="" />
          <p>Simply browse through our extensive list of trusted doctors, <br className='hidden sm:block' /> schedule your appointment hassle-free.</p>
        </div>
        <a href='#speciality' className='flex items-center gap-2 bg-white px-8 py-3 rounded-full text-[#595959] text-sm m-auto md:m-0 hover:scale-105 transition-all duration-300'>
          Book appointment <img className='w-3' src={assets.arrow_icon} alt="" />
        </a>
      </div>

      {/* ----- Header Right ----- */}
      <div className='md:w-1/2 relative'>
        <img className='w-full md:absolute bottom-0 h-auto rounded-lg' src={assets.header_img} alt="" />
      </div>
    </div>
  )
}

export default Header
```

NavBar.JSX :

```
import React, { useContext, useState } from 'react'
import { assets } from '../assets/assets'
import { NavLink, useNavigate } from 'react-router-dom'
import { AppContext } from '../context/AppContext'

const Navbar = () => {

  const navigate = useNavigate()

  const [showMenu, setShowMenu] = useState(false)
  const { token, setToken, userData } = useContext(AppContext)

  const logout = () => {
    localStorage.removeItem('token')
    setToken(false)
    navigate('/login')
  }

  return (
    <div className='flex items-center justify-between text-sm py-4 mb-5 border-b border-b-
[#ADADAD]'>
      <img onClick={() => navigate('/') } className='w-44 cursor-pointer' src={assets.logo} alt="" />
      <ul className='md:flex items-start gap-10 font-medium hidden'>
        <NavLink to='/' >
          <li className='py-1'>HOME</li>
          <hr className='border-none outline-none h-0.5 bg-primary w-3/5 m-auto hidden' />
        </NavLink>
        <NavLink to='/doctors' >
          <li className='py-1'>ALL DOCTORS</li>
          <hr className='border-none outline-none h-0.5 bg-primary w-3/5 m-auto hidden' />
        </NavLink>
        <NavLink to='/about' >
          <li className='py-1'>ABOUT</li>
          <hr className='border-none outline-none h-0.5 bg-primary w-3/5 m-auto hidden' />
        </NavLink>
        <NavLink to='/contact' >
          <li className='py-1'>CONTACT</li>
          <hr className='border-none outline-none h-0.5 bg-primary w-3/5 m-auto hidden' />
        </NavLink>
      </ul>
    </div>
  )
}
```

```

<div className='flex items-center gap-4 '>
  {
    token && userData
    ? <div className='flex items-center gap-2 cursor-pointer group relative'>
      <img className='w-8 rounded-full' src={userData.image} alt="" />
      <img className='w-2.5' src={assets.dropdown_icon} alt="" />
      <div className='absolute top-0 right-0 pt-14 text-base font-medium text-gray-600 z-20
hidden group-hover:block'>
        <div className='min-w-48 bg-gray-50 rounded flex flex-col gap-4 p-4'>
          <p onClick={() => navigate('/my-profile')} className='hover:text-black cursor-
pointer'>My Profile</p>
          <p onClick={() => navigate('/my-appointments')} className='hover:text-black cursor-
pointer'>My Appointments</p>
          <p onClick={logout} className='hover:text-black cursor-pointer'>Logout</p>
        </div>
      </div>
    : <button onClick={() => navigate('/login')} className='bg-primary text-white px-8 py-3
rounded-full font-light hidden md:block'>Create account</button>
  }
  <img onClick={() => setShowMenu(true)} className='w-6 md:hidden' src={assets.menu_icon}
alt="" />

  { /* ---- Mobile Menu ---- */
    <div className={`md:hidden ${showMenu ? 'fixed w-full' : 'h-0 w-0'} right-0 top-0 bottom-0 z-
20 overflow-hidden bg-white transition-all`} >
      <div className='flex items-center justify-between px-5 py-6'>
        <img src={assets.logo} className='w-36' alt="" />
        <img onClick={() => setShowMenu(false)} src={assets.cross_icon} className='w-7' alt=""
/>
      </div>
      <ul className='flex flex-col items-center gap-2 mt-5 px-5 text-lg font-medium'>
        <NavLink onClick={() => setShowMenu(false)} to="/"><p className='px-4 py-2 rounded full
inline-block'>HOME</p></NavLink>
        <NavLink onClick={() => setShowMenu(false)} to="/doctors" ><p className='px-4 py-2
rounded full inline-block'>ALL DOCTORS</p></NavLink>
        <NavLink onClick={() => setShowMenu(false)} to="/about" ><p className='px-4 py-2
rounded full inline-block'>ABOUT</p></NavLink>
        <NavLink onClick={() => setShowMenu(false)} to="/contact" ><p className='px-4 py-2
rounded full inline-block'>CONTACT</p></NavLink>
      </ul>
    </div>
  </div>

```



```
)  
}
```

```
export default Navbar
```

Related Doctor :

```
import React, { useContext, useEffect, useState } from 'react'
import { useNavigate } from 'react-router-dom'
import { AppContext } from '../context/AppContext'
const RelatedDoctors = ({ speciality, docId }) => {

  const navigate = useNavigate()
  const { doctors } = useContext(AppContext)

  const [relDoc, setRelDoc] = useState([])

  useEffect(() => {
    if (doctors.length > 0 && speciality) {
      const doctorsData = doctors.filter((doc) => doc.speciality === speciality && doc._id !== docId)
      setRelDoc(doctorsData)
    }
  }, [doctors, speciality, docId])

  return (
    <div className='flex flex-col items-center gap-4 my-16 text-[#262626]'>
      <h1 className='text-3xl font-medium'>Related Doctors</h1>
      <p className='sm:w-1/3 text-center text-sm'>Simply browse through our extensive list of trusted doctors.</p>
      <div className='w-full grid grid-cols-auto gap-4 pt-5 gap-y-6 px-3 sm:px-0'>
        {relDoc.map((item, index) => (
          <div onClick={() => { navigate(`/appointment/${item._id}`); scrollTo(0, 0) }}
            className='border border-[#C9D8FF] rounded-xl overflow-hidden cursor-pointer hover:translate-y-[-10px] transition-all duration-500' key={index}>
              <img className='bg-[#EAEFFF]' src={item.image} alt="" />
              <div className='p-4'>
                <div className={`flex items-center gap-2 text-sm text-center ${item.available ? 'text-green-500' : 'text-gray-500'}`}>
                  <p className={`${w-2 h-2 rounded-full ${item.available ? 'bg-green-500' : 'bg-gray-500'}`}`></p><p>{item.available ? 'Available' : 'Not Available'}</p>
                </div>
                <p className='text-[#262626] text-lg font-medium'>{item.name}</p>
                <p className='text-[#5C5C5C] text-sm'>{item.speciality}</p>
              </div>
            </div>
          )))
      </div>
    </div>
  )
}
```

```
      { /* <button className='bg-[#EAEFFF] text-gray-600 px-12 py-3 rounded-full mt-
10'>more</button> */}
      </div>
    )
  }

export default RelatedDoctors
```

Speciality Menu :

```
import React from 'react'
import { specialityData } from '../assets/assets'
import { Link } from 'react-router-dom'

const SpecialityMenu = () => {
  return (
    <div id='speciality' className='flex flex-col items-center gap-4 py-16 text-[#262626]'>
      <h1 className='text-3xl font-medium'>Find by Speciality</h1>
      <p className='sm:w-1/3 text-center text-sm'>Simply browse through our extensive list of
trusted doctors, schedule your appointment hassle-free.</p>
      <div className='flex sm:justify-center gap-4 pt-5 w-full overflow-scroll '>
        {specialityData.map((item, index) => (
          <Link to={`'/doctors/${item.speciality}`} onClick={() => scrollTo(0, 0)}
className='flex flex-col items-center text-xs cursor-pointer flex-shrink-0 hover:translate-y-[-10px]
transition-all duration-500' key={index}>
          <img className='w-16 sm:w-24 mb-2 ' src={item.image} alt="" />
          <p>{item.speciality}</p>
        </Link>
        ))}
      </div>
    </div>
  )
}

export default SpecialityMenu
```

TopDoctor :

```
import React, { useContext } from 'react'
import { useNavigate } from 'react-router-dom'
import { AppContext } from '../context/AppContext'
const TopDoctors = () => {

  const navigate = useNavigate()

  const { doctors } = useContext(AppContext)

  return (
    <div className='flex flex-col items-center gap-4 my-16 text-[#262626] md:mx-10'>
      <h1 className='text-3xl font-medium'>Top Doctors to Book</h1>
      <p className='sm:w-1/3 text-center text-sm'>Simply browse through our extensive list of
trusted doctors.</p>
      <div className='w-full grid grid-cols-auto gap-4 pt-5 gap-y-6 px-3 sm:px-0'>
        {doctors.slice(0, 10).map((item, index) => (
          <div onClick={() => { navigate(`/appointment/${item._id}`); scrollTo(0, 0) }}
className='border border-[#C9D8FF] rounded-xl overflow-hidden cursor-pointer hover:translate-y-
[-10px] transition-all duration-500' key={index}>
            <img className='bg-[#EAEFFF]' src={item.image} alt="" />
            <div className='p-4'>
              <div className={`flex items-center gap-2 text-sm text-center ${item.available ?
'text-green-500' : "text-gray-500"}`}>
                <p className={`${w-2 h-2 rounded-full ${item.available ? 'bg-green-500' : "bg-
gray-500"}`} `></p><p>{item.available ? 'Available' : "Not Available"}</p>
              </div>
              <p className='text-[#262626] text-lg font-medium'>{item.name}</p>
              <p className='text-[#5C5C5C] text-sm'>{item.speciality}</p>
            </div>
          </div>
        ))}
      </div>
      <button onClick={() => { navigate('/doctors'); scrollTo(0, 0) }} className='bg-[#EAEFFF]
text-gray-600 px-12 py-3 rounded-full mt-10'>more</button>
    </div>

  )
}

export default TopDoctors
```

AppContext.JSX :

```
import { createContext, useEffect, useState } from "react";
import { toast } from "react-toastify";
import axios from 'axios'

export const AppContext = createContext()

const AppContextProvider = (props) => {

  const currencySymbol = '₹'
  const backendUrl = import.meta.env.VITE_BACKEND_URL

  const [doctors, setDoctors] = useState([])
  const [token, setToken] = useState(localStorage.getItem('token') ? localStorage.getItem('token') : "")
  const [userData, setUserData] = useState(false)

  // Getting Doctors using API
  const getDoctosData = async () => {

    try {

      const { data } = await axios.get(backendUrl + '/api/doctor/list')
      if (data.success) {
        setDoctors(data.doctors)
      } else {
        toast.error(data.message)
      }
    } catch (error) {
      console.log(error)
      toast.error(error.message)
    }
  }

  // Getting User Profile using API
  const loadUserProfileData = async () => {

    try {

      const { data } = await axios.get(backendUrl + '/api/user/get-profile', { headers: { token } })

      if (data.success) {
```

```

        setUserData(data.userData)
      } else {
        toast.error(data.message)
      }
    } catch (error) {
      console.log(error)
      toast.error(error.message)
    }
  }

  useEffect(() => {
    getDoctosData()
  }, [])

  useEffect(() => {
    if (token) {
      loadUserProfileData()
    }
  }, [token])

  const value = {
    doctors, getDoctosData,
    currencySymbol,
    backendUrl,
    token, setToken,
    userData, setUserData, loadUserProfileData
  }

  return (
    <AppContext.Provider value={value}>
      {props.children}
    </AppContext.Provider>
  )
}

export default AppContextProvider

```

PAGES :

About.JSX :

```
import React from 'react'
import { assets } from '../assets/assets'

const About = () => {
  return (
    <div>

      <div className='text-center text-2xl pt-10 text-[#707070]'>
        <p>ABOUT <span className='text-gray-700 font-semibold'>US</span></p>
      </div>

      <div className='my-10 flex flex-col md:flex-row gap-12'>
        <img className='w-full md:max-w-[360px]' src={assets.about_image} alt="" />
        <div className='flex flex-col justify-center gap-6 md:w-2/4 text-sm text-gray-600'>
          <p>Welcome to MediSync, your trusted partner in managing your healthcare needs conveniently and efficiently. At MediSync, we understand the challenges individuals face when it comes to scheduling doctor appointments and managing their health records.</p>
          <p>MediSync is committed to excellence in healthcare technology. We continuously strive to enhance our platform, integrating the latest advancements to improve user experience and deliver superior service. Whether you're booking your first appointment or managing ongoing care, MediSync is here to support you every step of the way.</p>
          <b className='text-gray-800'>Our Vision</b>
          <p>Our vision at MediSync is to create a seamless healthcare experience for every user. We aim to bridge the gap between patients and healthcare providers, making it easier for you to access the care you need, when you need it.</p>
        </div>
      </div>

      <div className='text-xl my-4'>
        <p>WHY <span className='text-gray-700 font-semibold'>CHOOSE US</span></p>
      </div>

      <div className='flex flex-col md:flex-row mb-20'>
        <div className='border px-10 md:px-16 py-8 sm:py-16 flex flex-col gap-5 text-[15px] hover:bg-primary hover:text-white transition-all duration-300 text-gray-600 cursor-pointer'>
          <b>EFFICIENCY:</b>
          <p>Streamlined appointment scheduling that fits into your busy lifestyle.</p>
        </div>
      </div>
    </div>
  )
}
```



```

    </div>
    <div className='border px-10 md:px-16 py-8 sm:py-16 flex flex-col gap-5 text-[15px]
hover:bg-primary hover:text-white transition-all duration-300 text-gray-600 cursor-pointer'>
      <b>CONVENIENCE: </b>
      <p>Access to a network of trusted healthcare professionals in your area.</p>
    </div>
    <div className='border px-10 md:px-16 py-8 sm:py-16 flex flex-col gap-5 text-[15px]
hover:bg-primary hover:text-white transition-all duration-300 text-gray-600 cursor-pointer'>
      <b>PERSONALIZATION:</b>
      <p>Tailored recommendations and reminders to help you stay on top of your health.</p>
    </div>
  </div>
)
}

```

export default About

Appointment :

```
import React, { useContext, useEffect, useState } from 'react'
import { useNavigate, useParams } from 'react-router-dom'
import { AppContext } from '../context/AppContext'
import { assets } from '../assets/assets'
import RelatedDoctors from '../components/RelatedDoctors'
import axios from 'axios'
import { toast } from 'react-toastify'

const Appointment = () => {

  const { docId } = useParams()
  const { doctors, currencySymbol, backendUrl, token, getDoctorsData } = useContext(AppContext)
  const daysOfWeek = ['SUN', 'MON', 'TUE', 'WED', 'THU', 'FRI', 'SAT']

  const [docInfo, setDocInfo] = useState(false)
  const [docSlots, setDocSlots] = useState([])
  const [slotIndex, setSlotIndex] = useState(0)
  const [slotTime, setSlotTime] = useState("")

  const navigate = useNavigate()

  const fetchDocInfo = async () => {
    const docInfo = doctors.find((doc) => doc._id === docId)
    setDocInfo(docInfo)
  }

  const getAvailableSolts = async () => {

    setDocSlots([])

    // getting current date
    let today = new Date()

    for (let i = 0; i < 7; i++) {

      // getting date with index
      let currentDate = new Date(today)
      currentDate.setDate(today.getDate() + i)

      // setting end time of the date with index
      let endTime = new Date()
```

```

endTime.setDate(today.getDate() + i)
endTime.setHours(21, 0, 0, 0)

// setting hours
if (today.getDate() === currentDate.getDate()) {
  currentDate.setHours(currentDate.getHours() > 10 ? currentDate.getHours() + 1 : 10)
  currentDate.setMinutes(currentDate.getMinutes() > 30 ? 30 : 0)
} else {
  currentDate.setHours(10)
  currentDate.setMinutes(0)
}

let timeSlots = [];

while (currentDate < endTime) {
  let formattedTime = currentDate.toLocaleTimeString([], { hour: '2-digit', minute: '2-digit'
});

  let day = currentDate.getDate()
  let month = currentDate.getMonth() + 1
  let year = currentDate.getFullYear()

  const slotDate = day + "_" + month + "_" + year
  const slotTime = formattedTime

  const isSlotAvailable = docInfo.slots_booked[slotDate] &&
docInfo.slots_booked[slotDate].includes(slotTime) ? false : true

  if (isSlotAvailable) {

    // Add slot to array
    timeSlots.push({
      datetime: new Date(currentDate),
      time: formattedTime
    })
  }

  // Increment current time by 30 minutes
  currentDate.setMinutes(currentDate.getMinutes() + 30);
}

setDocSlots(prev => ([...prev, timeSlots]))
}

```

```

    }

    const bookAppointment = async () => {

      if (!token) {
        toast.warning('Login to book appointment')
        return navigate('/login')
      }

      const date = docSlots[slotIndex][0].datetime

      let day = date.getDate()
      let month = date.getMonth() + 1
      let year = date.getFullYear()

      const slotDate = day + "_" + month + "_" + year

      try {

        const { data } = await axios.post(backendUrl + '/api/user/book-appointment', { docId,
slotDate, slotTime }, { headers: { token } })
        if (data.success) {
          toast.success(data.message)
          getDoctosData()
          navigate('/my-appointments')
        } else {
          toast.error(data.message)
        }
      } catch (error) {
        console.log(error)
        toast.error(error.message)
      }
    }

    useEffect(() => {
      if (doctors.length > 0) {
        fetchDocInfo()
      }
    }, [doctors, docId])

    useEffect(() => {

```

```

        if (docInfo) {
            getAvailableSolts()
        }
    }, [docInfo])

    return docInfo ? (
        <div>

            {/* ----- Doctor Details ----- */}
            <div className='flex flex-col sm:flex-row gap-4'>
                <div>
                    <img className='bg-primary w-full sm:max-w-72 rounded-lg' src={docInfo.image}
alt="" />
                </div>

                <div className='flex-1 border border-[#ADADAD] rounded-lg p-8 py-7 bg-white mx-2
sm:mx-0 mt-[-80px] sm:mt-0'>

                    {/* ----- Doc Info : name, degree, experience ----- */}

                    <p className='flex items-center gap-2 text-3xl font-medium text-gray-
700'>{docInfo.name} <img className='w-5' src={assets.verified_icon} alt="" /></p>
                    <div className='flex items-center gap-2 mt-1 text-gray-600'>
                        <p>{docInfo.degree} - {docInfo.speciality}</p>
                        <button className='py-0.5 px-2 border text-xs rounded-
full'>{docInfo.experience}</button>
                    </div>

                    {/* ----- Doc About ----- */}
                    <div>
                        <p className='flex items-center gap-1 text-sm font-medium text-[#262626] mt-
3'>About <img className='w-3' src={assets.info_icon} alt="" /></p>
                        <p className='text-sm text-gray-600 max-w-[700px] mt-1'>{docInfo.about}</p>
                    </div>

                    <p className='text-gray-600 font-medium mt-4'>Appointment fee: <span
className='text-gray-800'>{currencySymbol} {docInfo.fees}</span> </p>
                    </div>
                </div>

                {/* Booking slots */}
                <div className='sm:ml-72 sm:pl-4 mt-8 font-medium text-[#565656]'>
                    <p>Booking slots</p>
                    <div className='flex gap-3 items-center w-full overflow-x-scroll mt-4'>

```

```

    {docSlots.length && docSlots.map((item, index) => (
      <div onClick={() => setSlotIndex(index)} key={index} className={`text-center py-6
min-w-16 rounded-full cursor-pointer ${slotIndex === index ? 'bg-primary text-white' : 'border
border-[#DDDDDD]`} `}>
        <p>{item[0] && daysOfWeek[item[0].datetime.getDay()]}</p>
        <p>{item[0] && item[0].datetime.getDate()}</p>
      </div>
    ))}
  </div>

  <div className='flex items-center gap-3 w-full overflow-x-scroll mt-4'>
    {docSlots.length && docSlots[slotIndex].map((item, index) => (
      <p onClick={() => setSlotTime(item.time)} key={index} className={`text-sm font-
light flex-shrink-0 px-5 py-2 rounded-full cursor-pointer ${item.time === slotTime ? 'bg-primary
text-white' : 'text-[#949494] border border-[#B4B4B4]`} `}>{item.time.toLowerCase()}</p>
    ))}
  </div>

  <button onClick={bookAppointment} className='bg-primary text-white text-sm font-light
px-20 py-3 rounded-full my-6'>Book an appointment</button>
</div>

  {/* Listing Related Doctors */}
  <RelatedDoctors speciality={docInfo.speciality} docId={docId} />
</div>
) : null
}

export default Appointment

```

Contact.JSX :

```
import React from 'react'
import { assets } from '../assets/assets'

const Contact = () => {
  return (
    <div>

      <div className='text-center text-2xl pt-10 text-[#707070]'>
        <p>CONTACT <span className='text-gray-700 font-semibold'>US</span></p>
      </div>

      <div className='my-10 flex flex-col justify-center md:flex-row gap-10 mb-28 text-sm'>
        <img className='w-full md:max-w-[360px]' src={assets.contact_image} alt="" />
        <div className='flex flex-col justify-center items-start gap-6'>
          <p className='font-semibold text-lg text-gray-600'>OUR OFFICE</p>
          <p className='text-gray-500'>54709 Willms Station <br /> Suite 350, Washington, USA</p>
          <p className='text-gray-500'>Tel: (415) 555-0132 <br /> Email:
            greatstackdev@gmail.com</p>
          <p className='font-semibold text-lg text-gray-600'>CAREERS AT PRESCRIPTO</p>
          <p className='text-gray-500'>Learn more about our teams and job openings.</p>
          <button className='border border-black px-8 py-4 text-sm hover:bg-black hover:text-white
            transition-all duration-500'>Explore Jobs</button>
        </div>
      </div>

    </div>
  )
}

export default Contact
```

Doctor.jsx :

```
import React, { useContext, useEffect, useState } from 'react'
import { AppContext } from '../context/AppContext'
import { useNavigate, useParams } from 'react-router-dom'

const Doctors = () => {

  const { speciality } = useParams()

  const [filterDoc, setFilterDoc] = useState([])
  const [showFilter, setShowFilter] = useState(false)
  const navigate = useNavigate();

  const { doctors } = useContext(AppContext)

  const applyFilter = () => {
    if (speciality) {
      setFilterDoc(doctors.filter(doc => doc.speciality === speciality))
    } else {
      setFilterDoc(doctors)
    }
  }

  useEffect(() => {
    applyFilter()
  }, [doctors, speciality])

  return (
    <div>
      <p className='text-gray-600'>Browse through the doctors specialist.</p>
      <div className='flex flex-col sm:flex-row items-start gap-5 mt-5'>
        <button onClick={() => setShowFilter(!showFilter)} className={`py-1 px-3 border rounded text-sm transition-all sm:hidden ${showFilter ? 'bg-primary text-white' : ''}`>Filters</button>
        <div className={`flex-col gap-4 text-sm text-gray-600 ${showFilter ? 'flex' : 'hidden sm:flex'}`>
          <p onClick={() => speciality === 'General physician' ? navigate('/doctors') : navigate('/doctors/General physician')} className={`w-[94vw] sm:w-auto pl-3 py-1.5 pr-16 border border-gray-300 rounded transition-all cursor-pointer ${speciality === 'General physician' ? 'bg-[#E2E5FF] text-black' : ''}`>General physician</p>
          <p onClick={() => speciality === 'Gynecologist' ? navigate('/doctors') : navigate('/doctors/Gynecologist')} className={`w-[94vw] sm:w-auto pl-3 py-1.5 pr-16 border border-gray-300 rounded transition-all cursor-pointer ${speciality === 'Gynecologist' ? 'bg-[#E2E5FF] text-black' : ''}`>Gynecologist</p>
        </div>
      </div>
    </div>
  )
}
```



```

    <p onClick={() => speciality === 'Dermatologist' ? navigate('/doctors') :
navigate('/doctors/Dermatologist')} className={`w-[94vw] sm:w-auto pl-3 py-1.5 pr-16 border
border-gray-300 rounded transition-all cursor-pointer ${speciality === 'Dermatologist' ? 'bg-
[#E2E5FF] text-black' : ''}`>Dermatologist</p>
    <p onClick={() => speciality === 'Pediatricians' ? navigate('/doctors') :
navigate('/doctors/Pediatricians')} className={`w-[94vw] sm:w-auto pl-3 py-1.5 pr-16 border
border-gray-300 rounded transition-all cursor-pointer ${speciality === 'Pediatricians' ? 'bg-
[#E2E5FF] text-black' : ''}`>Pediatricians</p>
    <p onClick={() => speciality === 'Neurologist' ? navigate('/doctors') :
navigate('/doctors/Neurologist')} className={`w-[94vw] sm:w-auto pl-3 py-1.5 pr-16 border border-
gray-300 rounded transition-all cursor-pointer ${speciality === 'Neurologist' ? 'bg-[#E2E5FF] text-
black' : ''}`>Neurologist</p>
    <p onClick={() => speciality === 'Gastroenterologist' ? navigate('/doctors') :
navigate('/doctors/Gastroenterologist')} className={`w-[94vw] sm:w-auto pl-3 py-1.5 pr-16 border
border-gray-300 rounded transition-all cursor-pointer ${speciality === 'Gastroenterologist' ? 'bg-
[#E2E5FF] text-black' : ''}`>Gastroenterologist</p>
  </div>
  <div className='w-full grid grid-cols-auto gap-4 gap-y-6'>
    {filterDoc.map((item, index) => (
      <div onClick={() => { navigate(`/appointment/${item._id}`); scrollTo(0, 0) }}
className='border border-[#C9D8FF] rounded-xl overflow-hidden cursor-pointer hover:translate-y-
[-10px] transition-all duration-500' key={index}>
        <img className='bg-[#EAEFFF]' src={item.image} alt="" />
        <div className='p-4'>
          <div className={`flex items-center gap-2 text-sm text-center ${item.available ? 'text-
green-500' : 'text-gray-500'}`>
            <p className={`w-2 h-2 rounded-full ${item.available ? 'bg-green-500' : 'bg-gray-
500'}`></p><p>{item.available ? 'Available' : 'Not Available'}</p>
          </div>
          <p className='text-[#262626] text-lg font-medium'>{item.name}</p>
          <p className='text-[#5C5C5C] text-sm'>{item.speciality}</p>
        </div>
      </div>
    ))}
  </div>
</div>
)
}

```

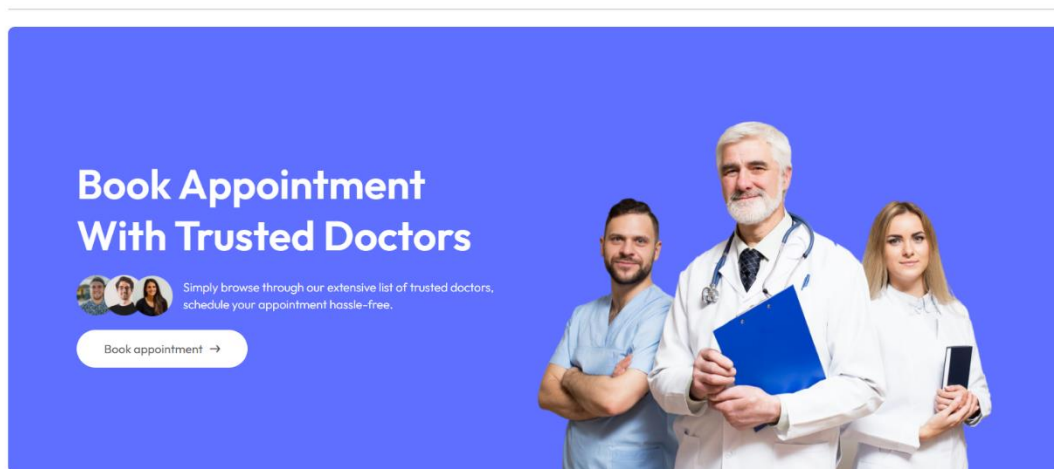
export default Doctors

CHAPTER 7

RESULTS

7.1 Screens and Explanations

This chapter will include all the screens available in the project




Find by Speciality

Simply browse through our extensive list of trusted doctors,
schedule your appointment hassle-free.


Hero Section

Top Doctors to Book


Simply browse through our extensive list of trusted doctors.



● Available
Saumya
Neurologist




● Available
Vansh Verma
General physician




● Available
Shikhar
General physician


more

[View All Doctors](#)



Saumya 

MD - Neurologist 3 Year

About 

Hey I'm Saumya Sharma!

Appointment fee: ₹1000

Booking slots

FRI
18

SAT
19

SUN
20

MON
21

TUE
22

WED
23

Book an appointment

Appointment Booking Page

CONTACT US



OUR OFFICE

54709 Willms Station
Suite 350, Washington, USA

Tel: (415) 555-0132
Email: greatstackdev@gmail.com

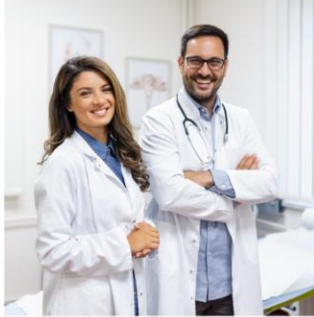
CAREERS AT PRESCRIPTO

Learn more about our teams and job openings.

[Explore Jobs](#)

Contact Us Page

ABOUT US



Welcome to MediSync, your trusted partner in managing your healthcare needs conveniently and efficiently. At MediSync, we understand the challenges individuals face when it comes to scheduling doctor appointments and managing their health records.

MediSync is committed to excellence in healthcare technology. We continuously strive to enhance our platform, integrating the latest advancements to improve user experience and deliver superior service. Whether you're booking your first appointment or managing ongoing care, MediSync is here to support you every step of the way.

Our Vision

Our vision at MediSync is to create a seamless healthcare experience for every user. We aim to bridge the gap between patients and healthcare providers, making it easier for you to access the care you need, when you need it.

WHY CHOOSE US

EFFICIENCY

CONVENIENCE

PERSONALIZATION

About Us Page

Create Account

Please sign up to book appointment

Full Name

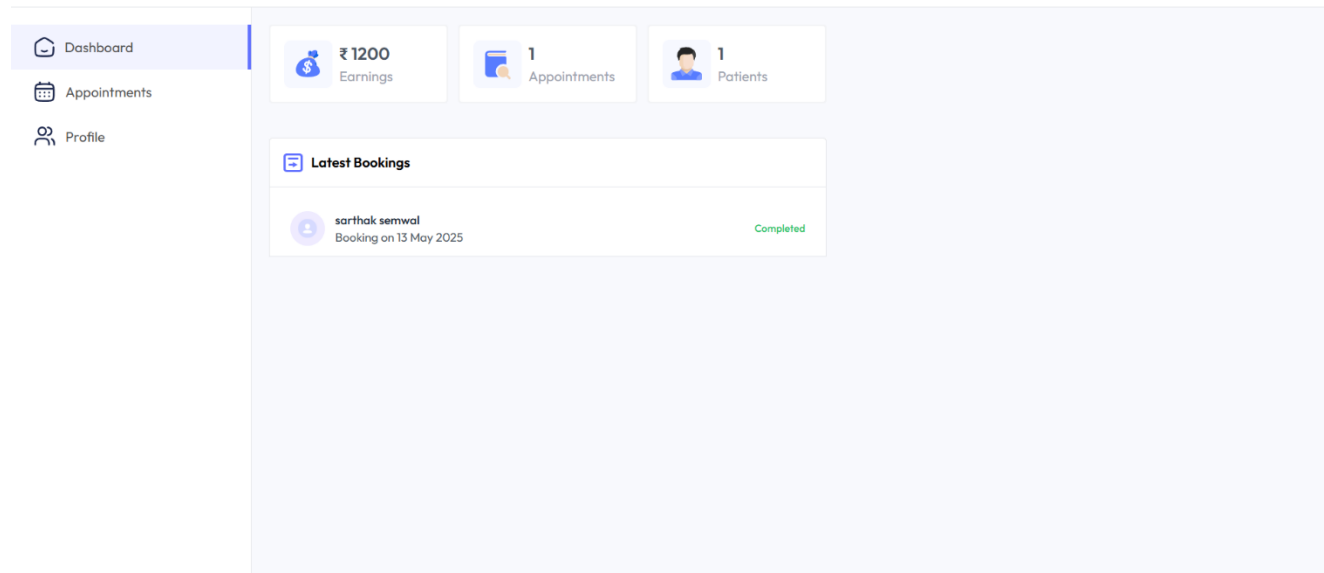
Email

Password


Create account

Already have an account? [Login here](#)

Create Account Page






Doctor Panel

| | | | | | | | |
|--------------|------------------|--|---------|-----|-----------------------|-------|-----------|
| Dashboard | All Appointments | | | | | | |
| Appointments | # | Patient | Payment | Age | Date & Time | Fees | Action |
| Profile | 0 |  sarthak semwal | Online | NaN | 13 May 2025, 05:30 PM | ₹1200 | Completed |

localhost:5174/doctor-appointments

Doctor Appointment Panel

-  Dashboard
-  Appointments
-  Profile



Shikhar

MBBS/MD(General Phy) - General physician 4 Year

About:

Hello there!

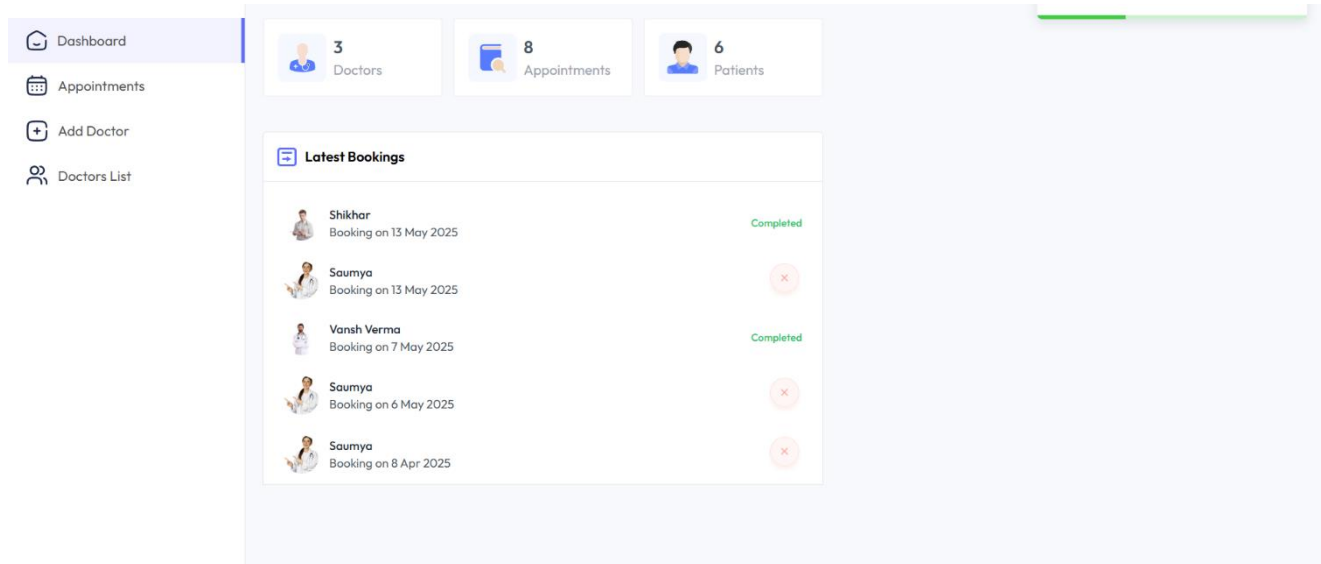
Appointment fee: ₹ 1200

Address: Street 7
Jalandhar





☒ Available




















Edit

Doctor Profile



Admin Panel

-  Dashboard
-  Appointments
-  Add Doctor
-  Doctors List

| All Appointments | | | | | | |
|------------------|--|-----|-----------------------|---|--------|---|
| # | Patient | Age | Date & Time | Doctor | Fees | Action |
| 1 |  sarthak semwal | NaN | 13 May 2025, 05:30 PM |  Shikhar | ₹1200 | Completed |
| 2 |  sarthak semwal | NaN | 13 May 2025, 05:30 PM |  Saumya | ₹1000 | Cancelled |
| 3 |  Rahul Tyagi | NaN | 7 May 2025, 10:00 am |  Vansh Verma | ₹10000 | Completed |
| 4 |  Rahul Tyagi | NaN | 6 May 2025, 11:00 am |  Saumya | ₹1000 |  |
| 5 |  Amit Gupta | NaN | 8 Apr 2025, 11:00 am |  Saumya | ₹1000 |  |
| 6 |  Disha Gupta | NaN | 8 Apr 2025, 10:00 am |  Saumya | ₹1000 |  |
| 7 |  david joe | NaN | 15 Mar 2025, 03:00 pm |  Saumya | ₹1000 | Cancelled |
| 8 |  John Doe | NaN | 15 Mar 2025, 02:30 PM |  Saumya | ₹1000 | Cancelled |

Admin Panel Appointment

Dashboard

Appointments

Add Doctor

Doctors List

Add Doctor

Upload doctor picture

Your name

Name

Speciality

General physician

Doctor Email

Email

Degree

Degree

Set Password

Password

Address

Address 1

Address 2

Experience

1 Year

Fees

Doctor fees

About Doctor

Add Doctors Admin Panel

CHAPTER 8

DISCUSSION

The MediSync platform represents a significant advancement in the healthcare sector, offering a seamless experience for patients, doctors, and admins. By leveraging modern web technologies such as the MERN stack, MediSync ensures scalability, efficiency, and a user-friendly interface. This discussion covers the performance of MediSync, its impact on healthcare management, and potential areas for future enhancement.

8.1 Performance

MediSync has demonstrated itself as a reliable and efficient platform due to the following key factors:

- **Scalability**
The use of the MERN stack ensures that MediSync can scale effortlessly to accommodate an increasing number of users, doctors, and appointments without compromising system performance. MongoDB's flexible and efficient data storage allows quick retrieval of user profiles, appointment histories, and medical details.
- **User-Friendly Interface**
React.js provides an intuitive and responsive frontend, enabling users to book appointments, search for doctors by specialty, and view their consultation schedules seamlessly across all devices. The simple yet engaging UI improves accessibility and promotes active user interaction.
- **Real-Time Features**
The platform integrates real-time appointment tracking, allowing patients to book available slots based on doctors' updated schedules. Doctors can instantly accept, reject, or reschedule appointments, ensuring smooth communication and better time management.
- **Security**
Security is a critical concern in healthcare applications. MediSync uses Auth0 for secure authentication and role-based access control, ensuring that user information, medical data, and transaction details are handled safely. All data exchanges are encrypted to maintain privacy and protect sensitive information.
- **Quick Search and Filtering**
Patients can search for doctors by specialty, location, and availability, thanks to advanced filtering features. This ensures users can quickly find the most relevant healthcare professionals tailored to their needs, improving efficiency in the booking process.
- **Robust Backend**
MediSync uses Node.js and Express.js for backend operations, which ensures fast data handling and minimal lag when processing appointment requests, payment transactions, and user queries.

8.2 Future Research Directions

While MediSync is already a powerful healthcare platform, there are several opportunities for future improvements and additional features:

1. **Video Consultation Integration**
Implementing secure video call features will allow doctors and patients to conduct remote consultations, especially for non-urgent or follow-up appointments, expanding the reach of healthcare services.
2. **User Feedback System**
A feedback system where patients can rate their consultation experience and provide suggestions will help improve doctor-patient interactions and allow MediSync to evolve based on user preferences.
3. **Multilingual Support**
Adding multilingual support will allow MediSync to cater to a more diverse user base, particularly in regions with multiple languages, making the platform accessible to a broader audience.
4. **Mobile App Development**
Developing dedicated mobile apps for iOS and Android will improve accessibility and user experience, allowing patients and doctors to manage appointments on the go, even when they are offline or on mobile networks.

CHAPTER 9

CONCLUSION

The development of **MediSync** — a comprehensive doctor booking platform — has been an exciting and rewarding journey, designed to simplify healthcare access for both patients and doctors. By leveraging modern web technologies such as React.js, Node.js, and MongoDB, the platform offers a user-friendly interface, secure authentication, real-time appointment management, and seamless payment processing.

Throughout the project, we focused on addressing key challenges such as scalability, security, and performance optimization, ensuring that the platform can grow alongside its user base while maintaining speed and efficiency. Integrating third-party APIs, such as Stripe for payments and Google Maps for location services, further enriched the user experience by providing essential, real-time functionalities.

The system's architecture supports the unique needs of three distinct user roles — patients, doctors, and admins — each with tailored features that ensure smooth interactions and data management. Additionally, careful attention was given to ensuring data privacy and secure handling of sensitive information, which is paramount in healthcare applications.

Looking ahead, there are several opportunities for enhancement, including the integration of video consultations, AI-based doctor recommendations, and multilingual support, which could further improve accessibility and user engagement. The implementation of these features will continue to evolve **MediSync** into an even more robust and inclusive healthcare solution.

In conclusion, **MediSync** successfully addresses the needs of modern healthcare management, enabling patients to easily book consultations, doctors to manage their schedules, and admins to oversee the platform's overall performance. As the healthcare landscape continues to evolve, **MediSync** has the potential to play a significant role in enhancing the delivery of healthcare services, making them more efficient, accessible, and patient-centered.

CHAPTER 10

REFERENCES

- 1 **MongoDB Documentation:** <https://www.mongodb.com/docs/>
- 2 **React.js Documentation:** <https://reactjs.org/docs/>
- 3 **Express.js Guide:** <https://expressjs.com/>
- 4 **Node.js Documentation:** <https://nodejs.org/en/docs/>
- 5 **W3Schools Tutorials on Full-Stack Development:** <https://www.w3schools.com/>

CHAPTER 11

BIBLIOGRAPHY

1. Books

- Garrison, D. R., & Akyol, Z. (2011). *Digital Tools for Enhanced User Engagement in Web Applications*. International Journal of Web Technologies.
- Hawkins, L., & Meyer, J. (2017). *The Impact of Modern Technologies in Healthcare Applications*. Journal of Technology in Health.
- Smith, J., & Lee, K. (2019). *User-Centric Design for Healthcare Platforms: A Comparative Study*. Routledge.

2. Websites

- **MongoDB Documentation:** <https://www.mongodb.com/docs/>
- **React.js Documentation:** <https://reactjs.org/docs/>
- **Node.js Resources:** <https://nodejs.org/en/docs/>
- **Auth0 Authentication Guide:** <https://auth0.com/docs/>
- **Stripe API Documentation:** <https://stripe.com/docs>
- **Google Maps API Documentation:** <https://developers.google.com/maps/documentation>

3. Reports and Industry Insights

- World Health Organization. (2023). *Telemedicine and the Future of Healthcare Delivery*.
- Deloitte Insights. (2023). *The Rise of Telemedicine: Trends and Opportunities in Healthcare*.

4. Online Tutorials

- W3Schools. (2024). **Full-Stack Development with MERN Stack:** <https://www.w3schools.com/>
- Codecademy. (2024). **Building Secure Web Applications with JWT Authentication:** <https://www.codecademy.com/>