Project Report

On

DocSpot: Seamless Appointment Booking for Health

SUBMITTED BY

Team ID: LTVIP2025TMID49202

Team Size:4

Team Leader: C. Kamalanath

Team member: V. Anjali

Team member: Gonegandla Inthiyaz Tasha

eam Bmember: B

SUBMITTED TO

SMARTBRIDGE
Let's Bridge the Gap

Smart Internz

1. INTRODUCTION

1.1 Project Overview

DocSpot is a full-stack web application developed using the MERN (MongoDB, Express.js, React.js, Node.js) stack. It is designed to modernize and simplify the process of scheduling medical appointments. The platform serves as a centralized hub connecting patients, doctors, and administrators, providing each with a tailored interface to manage the appointment lifecycle efficiently. By eliminating traditional booking hassles like phone calls and long wait times, DocSpot offers a convenient, user-friendly, and real-time solution for healthcare management. The system is built on a robust client-server architecture, features role-based access control for security, and provides a seamless user experience from registration to post-appointment follow-up.

1.2 Purpose

The primary purpose of the DocSpot project is to address the inefficiencies of traditional appointment booking systems. The goal is to create a seamless digital experience that benefits all stakeholders.

- **For Patients:** To provide 24/7 access to browse doctors, view real-time availability, and book appointments from the comfort of their homes, reducing wait times and uncertainty.
- **For Doctors:** To offer a powerful dashboard to manage their schedules, view patient information, and control their appointment confirmations, thereby reducing the administrative load on their staff.
- **For the Platform:** To create a scalable, secure, and reliable system that can be trusted by both patients and healthcare professionals.

---

2. IDEATION PHASE

2.1 Problem Statement

Booking a doctor's appointment is an outdated and often frustrating process for patients. It typically involves making phone calls during limited office hours, being placed on hold, playing phone tag with receptionists, and having no clear visibility into a doctor's availability. This leads to wasted time, delayed healthcare, and a poor patient experience. For doctors' offices, this manual process consumes significant administrative resources that could be better spent on patient care.

2.2 Empathy Map Canvas (for a typical patient)

- **THINKS & FEELS:** "I hope I can get an appointment this week." "This is so frustrating, I've been on hold for 10 minutes." "I wish I knew which doctors accepted my insurance." "I feel anxious about my health, and this process is just adding more stress."
- **SEES:** Busy receptionists, limited time slots, confusing clinic websites, long waiting lists.
- **SAYS & DOES:** "Do you have any availability for next Tuesday morning?" "I'll have to call back later." Spends time searching online for doctor reviews. Asks friends for recommendations.
- **HEARS:** "The doctor is fully booked for the next two weeks." "Please hold, I'll check the schedule." "You'll need to call back on Monday to book."
- **PAINS:** Wasted time, frustration, uncertainty, inability to book outside of 9-5 hours, delayed access to care.
- **GAINS:** Convenience, control over their schedule, quick confirmation, peace of mind, ability to book anytime, anywhere.

2.3 Brainstorming

Initial brainstorming sessions produced several potential solutions, including a dedicated mobile application, an automated phone-based IVR system, and a comprehensive web platform. The web platform was chosen as the initial product due to its universal accessibility across devices (desktops, tablets, phones) without requiring a download, and for allowing a faster development cycle. Key features brainstormed included:

- User profiles for patients and doctors.
- Real-time calendar availability.
- Search and filter functionality (by specialty, location).
- Automated email/SMS notifications.
- An admin panel for quality control and verification.
- Secure document handling for medical information.

---

3. REQUIREMENT ANALYSIS

3.1 Customer Journey Map (Patient)

1. **Registration/Login:** The user visits the site and creates a new patient account or logs into an existing one.

2. **Discovery:** The user lands on a dashboard where they can search for doctors. They apply filters such as specialty and location.
3. **Selection:** The user reviews the profiles of matching doctors and selects one based on their credentials and availability.
4. **Booking:** The user clicks "Book Now," selects a desired date and time from the doctor's real-time calendar, and submits the appointment request.
5. **Confirmation:** The user receives an initial notification that their request has been sent. Once the doctor approves it, the user receives a final confirmation notification.
6. **Management:** The user can view their upcoming appointments in their dashboard and has the option to cancel or reschedule.
7. **Appointment Day:** The user attends the appointment.
8. **Follow-up:** The user receives a visit summary or follow-up instructions via the app.

3.2 Solution Requirement

- **Functional Requirements:**
  - User registration and authentication for all three roles (Patient, Doctor, Admin).
  - Secure login with password hashing (bcrypt) and session management (JWT).
  - Role-based dashboards with different functionalities for each user type.
  - Admin functionality to view and approve/reject new doctor applications.
  - Doctor functionality to manage their profile, set availability, and confirm/cancel appointments.
  - Patient functionality to search for doctors, book appointments, and view appointment history.
  - A notification system for appointment status changes.
- **Non-Functional Requirements:**
  - **Security:** All sensitive data must be encrypted, and access to routes must be protected based on user roles.
  - **Performance:** The application must have fast load times and quick API responses.
  - **Usability:** The user interface must be intuitive, clean, and easy to navigate for non-technical users.

- **Scalability:** The architecture should be able to handle a growing number of users and appointments.

3.3 Data Flow Diagram (DFD)

A typical data flow for booking an appointment is as follows:

1. **User (Patient)** provides booking details (date, time) via the **React Frontend**.
2. The Frontend, using **Axios**, sends a POST request with the booking data and the user's JWT authentication token to the **Express Backend API** (e.g., /api/appointments/book-appointment).
3. The **Express Router** directs the request to the appropriate **Controller** function.
4. The Controller validates the input and uses the **Mongoose Model** to create a new appointment document.
5. **Mongoose** translates this into a command that saves the document in the **MongoDB Database**.
6. The database confirms the save, and the Controller sends a success response (e.g., 201 Created) back to the Frontend.
7. The React Frontend receives the success response and displays a confirmation message to the user.

3.4 Technology Stack

- **Frontend:** React.js, React Router, Axios, Ant Design (for UI components)
- **Backend:** Node.js, Express.js
- **Database:** MongoDB **ODM (Object Data Modeling):** Mongoose
- **Authentication:** JSON Web Tokens (JWT), bcrypt.js
- **Development Tools:** VS Code, Git, npm

---

4. PROJECT DESIGN

4.1 Problem Solution Fit

The DocSpot platform directly addresses the core problems identified.

- **Problem:** Limited booking hours and phone tag. **Solution:** A 24/7 web platform that allows instant booking requests.
- **Problem:** Lack of transparency in doctor availability. **Solution:** A real-time calendar system that shows doctors' actual open slots.

- **Problem:** High administrative burden on clinics.
    - **Solution:** A self-service platform for patients and a simple management dashboard for doctors, automating the booking process.

4.2 Proposed Solution

The proposed solution is a three-tiered MERN stack application. It consists of a client-side React application that communicates with a server-side Express API, which in turn interacts with a MongoDB database. The system is divided into three core user modules:

1. **Patient Module:** Allows users to register, log in, search for doctors, and manage their appointments.
2. **Doctor Module:** Allows approved doctors to manage their professional profile, set their schedule, and approve or decline incoming appointment requests.
3. **Admin Module:** A backend interface for administrators to verify and approve new doctor registrations, ensuring the quality and legitimacy of providers on the platform.

4.3 Solution Architecture

The application follows a classic three-tier architecture:

1. **Presentation Layer (Frontend):** A React-based Single Page Application (SPA) that renders the UI in the user's browser. It manages the view layer and user interactions.
2. **Logic Layer (Backend):** A Node.js/Express.js server that exposes a RESTful API. It handles all business logic, authentication, authorization, and acts as the intermediary between the frontend and the database.
3. **Data Layer (Database):** A MongoDB database managed by Mongoose schemas. It is responsible for the persistent storage of all application data, including user profiles, doctor details, and appointment records.

---

5. PROJECT PLANNING & SCHEDULING

5.1 Project Planning

The project was executed in four distinct phases:

- **Phase 1: Foundation & Backend (Week 1):**
    - Environment setup and project structure initialization.
    - Design of Mongoose schemas for Users and Appointments.
    - Development of the backend API for user registration and login, including JWT authentication.
- **Phase 2: Core Frontend & Patient Features (Weeks 2-3):**

- Setup of the React frontend with routing.
- Development of UI pages for Login, Registration, and the main dashboard using Ant Design.
- Integration of registration/login forms with the backend API.
- Implementation of doctor listing and search functionality.

- **Phase 3: Doctor & Admin Modules (Week 4):**
  - Development of the "Apply as Doctor" feature.
  - Creation of the Admin dashboard for viewing and approving doctors.
  - Creation of the Doctor dashboard for managing appointments.
- **Phase 4: Testing & Refinement (Week 5):**
  - End-to-end functional testing of all user journeys. o UI/UX refinement and bug fixing.
  - Preparation of the final report and project demonstration.

6. FUNCTIONAL AND PERFORMANCE TESTING

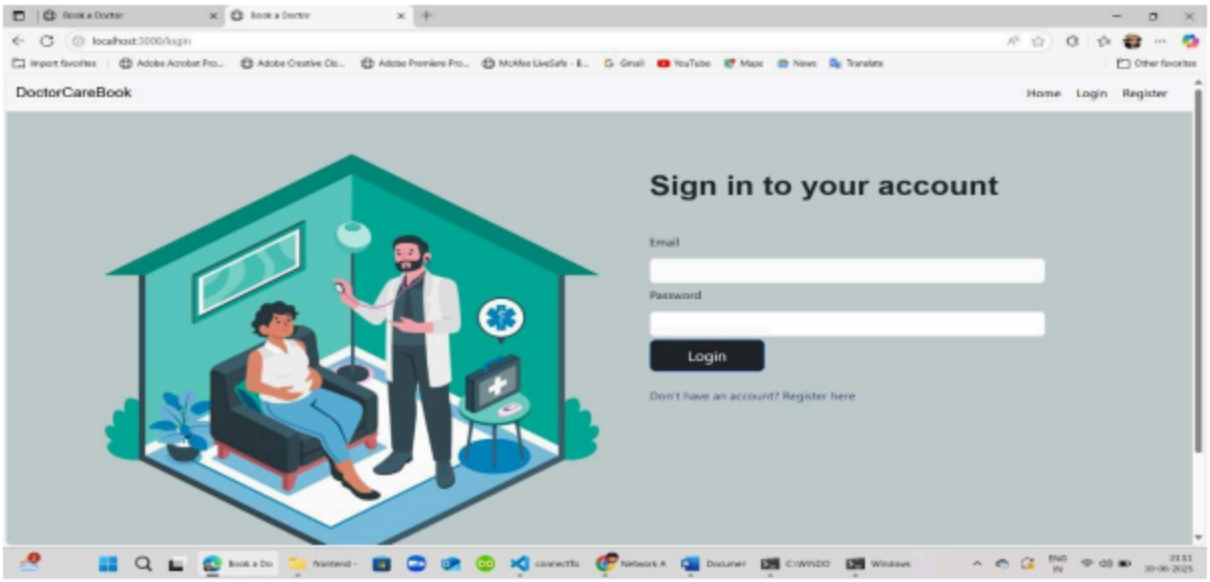
6.1 Performance Testing

While extensive automated testing was outside the project's immediate scope, the following testing strategies were considered and manually performed:

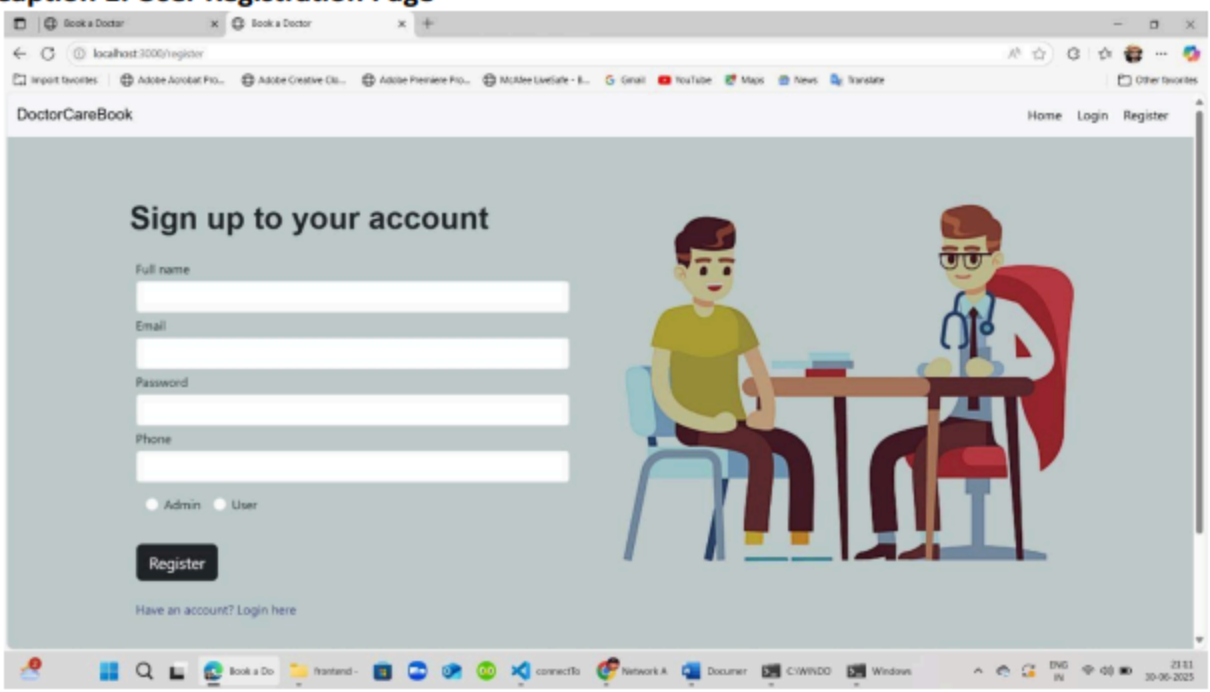
- **Functional Testing:** Manually tested all core user flows, including patient registration, doctor application, admin approval, and the complete appointment booking cycle to ensure they function as expected.
- **API Endpoint Testing:** Used the frontend application to test API responses for correctness, including success and error states for all endpoints.
- **Usability Testing:** Ensured the UI is intuitive and responsive across different screen sizes (desktops and mobile browsers).
- **Security Testing:** Verified that protected routes are inaccessible without a valid JWT token and that one user cannot access another user's data.

7. RESULTS

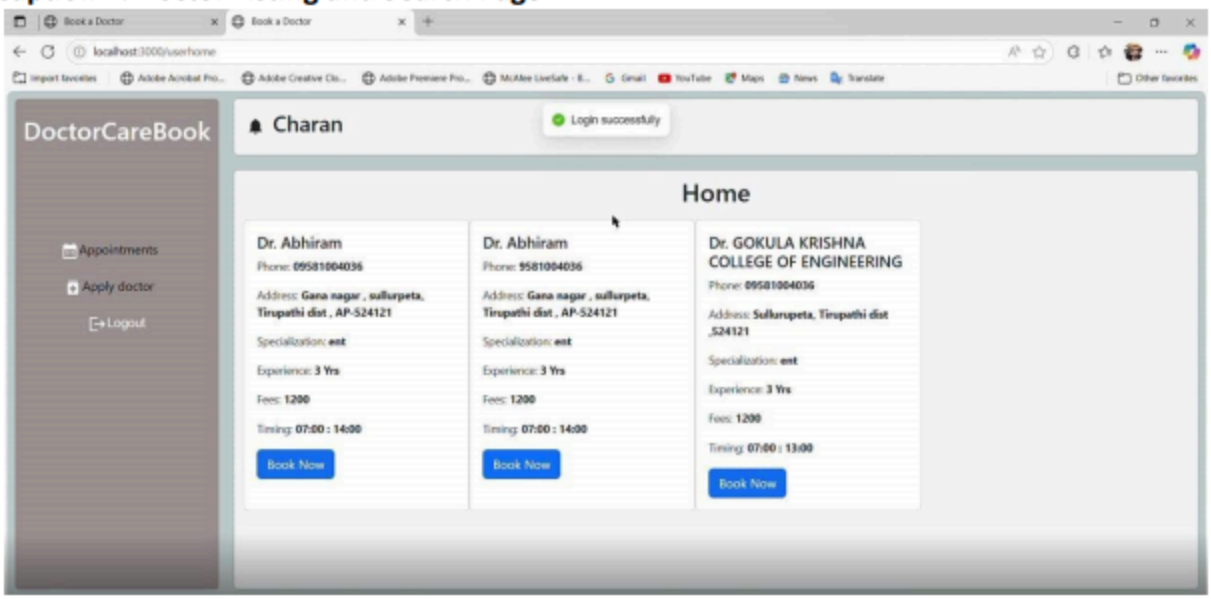
7.1 Output Screenshots



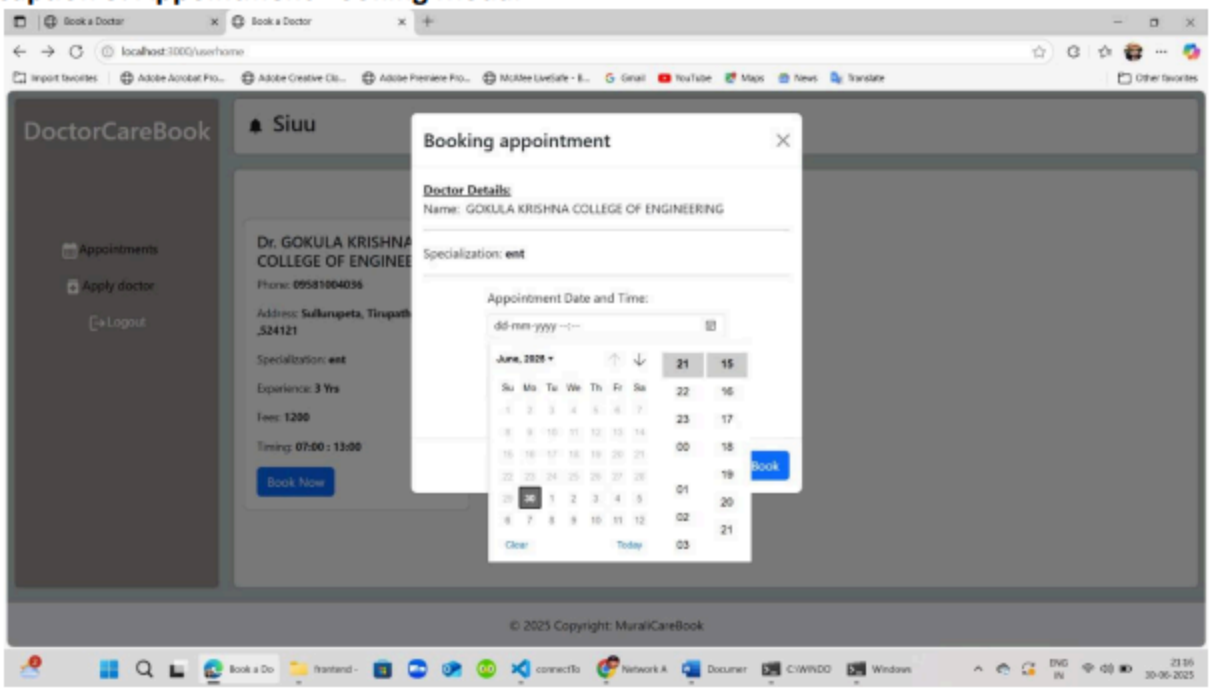
Caption 1: User Registration Page



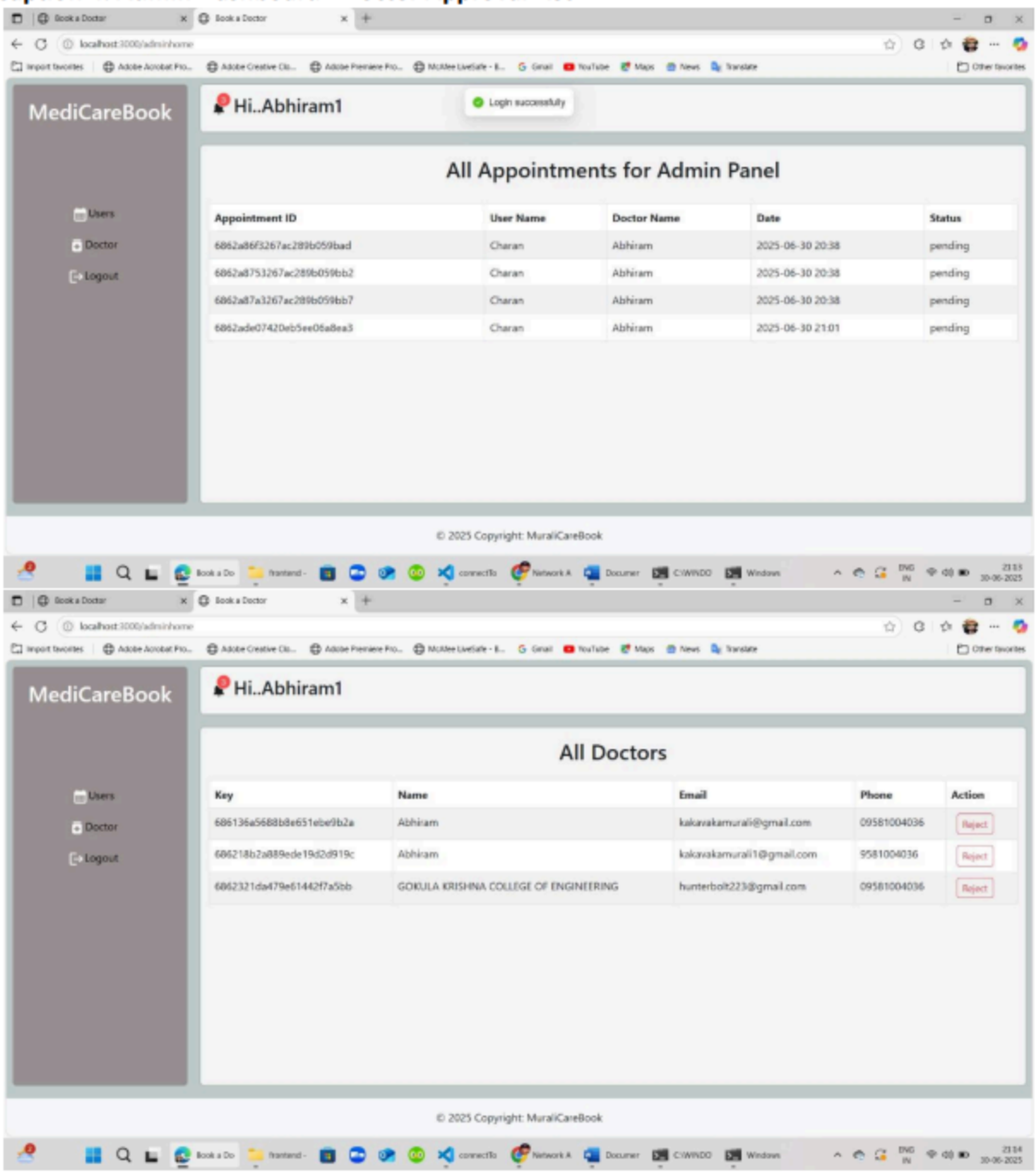
Caption 2: Doctor Listing and Search Page



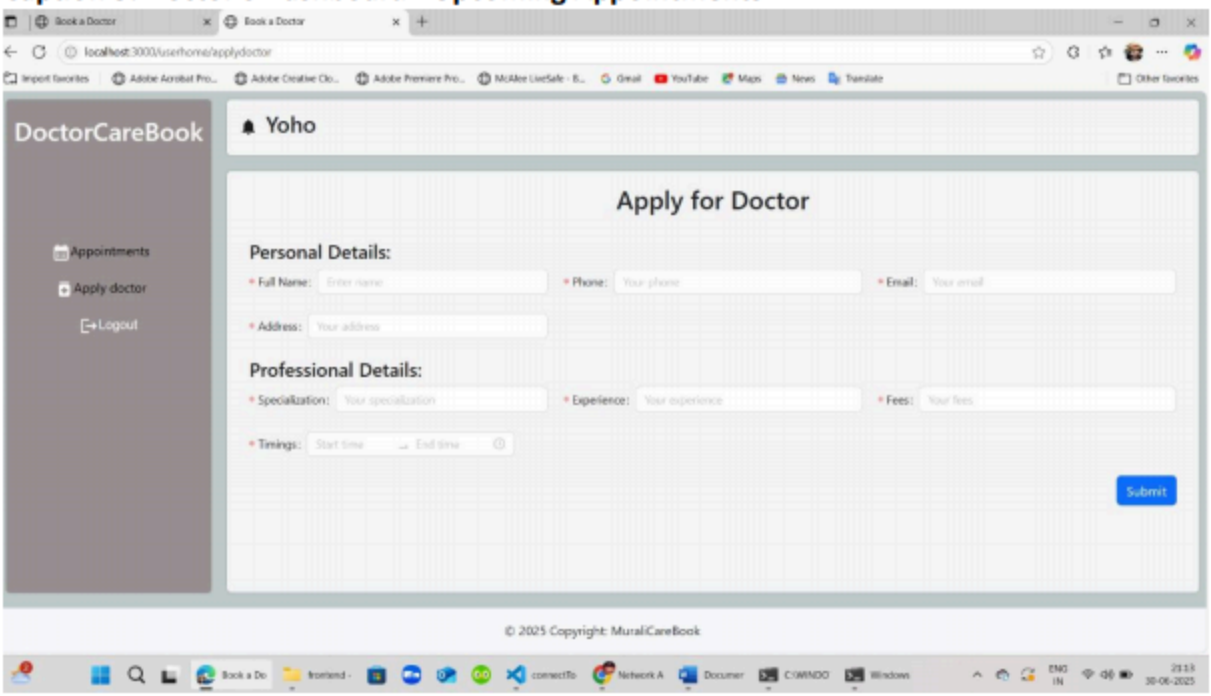
Caption 3: Appointment Booking Modal



Caption 4: Admin Dashboard - Doctor Approval List



Caption 5: Doctor's Dashboard - Upcoming Appointments



---

8. ADVANTAGES & DISADVANTAGES

Advantages

- **Convenience:** Users can book appointments 24/7 without being restricted to office hours.
- **Efficiency:** Automates the booking process, reducing manual work for clinic staff.
- **Transparency:** Patients have clear visibility into doctors' schedules and credentials.
- **Scalability:** The MERN stack architecture allows for easy scaling as the user base grows.
- **Centralized Management:** Provides a single platform for patients and doctors to manage all appointment-related activities.

Disadvantages

- **Internet Dependency:** The platform is entirely dependent on users having a stable internet connection.
- **Adoption Hurdle:** Requires both patients and doctors to adopt and actively use the new system.
- **Data Security Responsibility:** Handling sensitive health information requires strict adherence to privacy regulations (like HIPAA), which adds complexity.
- **Lack of Telehealth:** The current version does not include integrated video consultations.

9. CONCLUSION

The DocSpot project successfully achieved its objective of creating a modern, functional, and user-friendly platform for booking medical appointments. By leveraging the MERN stack, the application provides a robust and scalable solution to a common real-world problem. The project demonstrates a strong understanding of full-stack development principles, including client-server architecture, RESTful API design, database management, and user authentication. The final product is a testament to the power of modern web technologies in improving and streamlining traditional processes.

---

10. FUTURE SCOPE

The current platform serves as a strong foundation that can be expanded with several highvalue features:

- **Telehealth Integration:** Integrate a secure video conferencing API to allow for virtual consultations.
- **Online Payments:** Integrate a payment gateway like Stripe to handle consultation fees and co-pays.
- **Prescription Management:** Allow doctors to generate and send e-prescriptions to pharmacies.
- **Family Accounts:** Enable a user to manage appointments for dependents like children or elderly parents.
- **Dedicated Mobile Application:** Develop native iOS and Android applications for an enhanced mobile experience.
- **Ratings and Reviews:** Implement a system for verified patients to leave reviews for doctors.

---

11. APPENDIX Source

Code

The complete source code for the project is available at the GitHub repository linked below.

Dataset Link

This application does not use a static, predefined dataset. All data (user profiles, appointments) is dynamically generated and stored in the MongoDB database through user interaction with the live application.

GitHub & Project Demo Link

- **GitHub Repository:** https://github.com/YonoStorm/DocSpot.git
- **Live Demo Link:** https://drive.google.com/file/d/1oiK4WxJP56RnAz8qwo81Quf3xH98JJab/view?usp=sharing