DocSpot - Project Documentation

INTRODUCTION :

The Book a Doctor App is an innovative healthcare booking platform designed to streamline the process of connecting patients with healthcare providers. This system enables users to easily find, schedule, and manage medical appointments, all within a user-friendly interface. By offering functionalities like doctor browsing, appointment scheduling, and secure document uploading, the app caters to the needs of patients, doctors, and administrators alike.

Patients can search for doctors based on specialty, location, and availability, ensuring they find the right healthcare professional for their needs. Once a suitable doctor is selected, users can book appointments, manage their schedules, and receive notifications and reminders. Doctors benefit from a dedicated interface to manage appointments, update patient records, and communicate effectively, while administrators oversee the app's smooth operation, ensuring compliance and resolving any disputes.

Built with a robust technical architecture, the Book a Doctor App leverages a client-server model, using front-end frameworks like Bootstrap and Material UI for an engaging user experience, and a back end powered by Express.js and MongoDB to handle secure data transactions. This system offers a seamless, efficient, and secure healthcare booking experience, meeting the growing demand for accessible and well-organised healthcare services.

DESCRIPTION :

The Book a Doctor App is a user-centric platform designed to make healthcare appointment booking easy and efficient. The app connects patients and healthcare providers through a streamlined digital interface, allowing users to search, filter, and book appointments based on specialty, location, and real-time availability.

For patients, the app offers secure registration, profile creation, and document upload, with automated notifications and reminders to ensure no missed appointments. Doctors benefit from a dedicated dashboard where they can manage availability, confirm bookings, view patient records, and provide post-visit summaries. An admin interface allows for doctor registration approvals, system monitoring, and compliance management, ensuring a smooth, reliable experience.

Built using Bootstrap and Material UI for a modern frontend, the app also uses Axios for seamless backend communication, with Express.js and MongoDB handling server logic and data storage. Moment.js supports precise scheduling, and security libraries like bcrypt ensure secure handling of user data.

With features that enhance accessibility, communication, and efficiency, the Book a Doctor App supports the growing demand for accessible healthcare options, providing patients with convenient, reliable access to healthcare while helping providers manage their schedules effectively.

SCENARIO-BASED CASE STUDY :

1. USER REGISTRATION :

John, a patient in need of a routine check-up, downloads and opens the Book a Doctor App. He starts by registering as a customer, providing his email address and creating a password to ensure a secure login. Once the registration process is complete, John is welcomed to the app with the option to log in.

2. BROWSING DOCTORS :

After logging in, John is directed to a dashboard showcasing a list of doctors available for appointments. The app offers various filters for him to search for healthcare providers based on criteria such as specialty, location, and availability. John filters the list to find a family physician in his area, available for a routine check-up.

3. BOOKING AN APPOINTMENT :

John selects Dr. Smith, a family physician, and clicks the "Book Now" button. A booking form appears, prompting John to select his preferred appointment date and time. He is also asked to upload relevant documents, such as his medical records and insurance details. Once the form is completed, John submits the appointment request. He receives an immediate confirmation message indicating that his request is under review.

4. APPOINTMENT CONFIRMATION :

Dr. Smith, upon reviewing the request and his schedule, confirms the appointment. The status of John's appointment changes to "Scheduled," and John receives a notification with the appointment details—date, time, and location—via both email and SMS.

5. APPOINTMENT MANAGEMENT :

As the appointment date nears, John can access his booking history through the app's dashboard. Here, he can manage upcoming appointments, cancel or reschedule them, and update their status. If needed, he can contact the doctor or the support team for assistance.

6. ADMIN APPROVAL (BACKGROUND PROCESS) :

In the background, the app's admin is reviewing new doctor registrations. Dr. Smith, as a legitimate healthcare professional, is approved and added to the platform. The admin ensures that only verified doctors are listed, and the platform remains compliant with healthcare regulations and policies.

7. PLATFORM GOVERNANCE :

The admin monitors the platform's overall operation, addressing any issues, disputes, or system improvements. Ensuring the app's compliance with privacy regulations and the terms of service is also a key responsibility, ensuring a smooth and secure experience for all users.

8. DOCTOR'S APPOINTMENT MANAGEMENT :

On the day of the appointment, Dr. Smith logs into his dashboard and reviews his scheduled appointments. He sees John's appointment and confirms the time. Throughout the day, Dr. Smith manages other appointments, updates their statuses, and ensures patients are attended to efficiently.

9. APPOINTMENT CONSULTATION :

At the scheduled time, John visits Dr. Smith's office. During the consultation, Dr. Smith provides medical care, performs the check-up, and gives advice on maintaining good health. John's health concerns are addressed, and he feels assured that his routine check-up is complete.

10. POST-APPOINTMENT FOLLOW-UP :

After the consultation, Dr. Smith updates John's medical records within the app, noting any important observations, medications prescribed, or further treatments recommended. John receives a summary of his visit, including a prescription and any follow-up instructions via the app.

TECHNICAL ARCHITECTURE :

The Book a Doctor App features a modern technical architecture based on a client-server model. The frontend utilises Bootstrap and Material UI for a responsive user interface, with Axios handling seamless API communication. The backend is powered by Express.js, offering robust server-side logic, while MongoDB provides scalable data storage for user profiles, appointments, and doctor information. Authentication is secured using JWT for session management and bcrypt for password hashing. Moment.js manages date and time functionalities, ensuring accurate appointment scheduling. The admin interfaces overseas doctor registration, platform governance, and ensures compliance, with Role-based Access Control (RBAC) managing access levels. Scalability is supported by MongoDB, and performance optimization is achieved with load balancing and caching techniques.

Project Overview

DocSpot is a web-based doctor appointment booking system that allows users to:

- Register as a user or doctor
- Book appointments with doctors
- Doctors can confirm or cancel appointments
- Admin can manage users and doctors

Technologies Used

- **Frontend**: HTML, CSS, JavaScript
- **Backend**: Node.js, Express.js
- **Database**: MongoDB (with Mongoose)

Project Structure

```
docspot/
├── backend/
│   ├── models/
│   │   ├── User.js
│   │   ├── Doctor.js
│   │   └── Appointment.js
│   ├── router/
│   │   ├── auth.js
│   │   ├── doctors.js
│   │   └── appointments.js
│   ├── authMiddleware.js
│   └── server.js
├── frontend/
│   ├── landing.html
│   ├── login.html
│   ├── register.html
│   ├── register-user.html
│   ├── register-doctor.html
│   ├── user-dashboard.html
│   ├── doctor-dashboard.html
│   ├── doctor-appointments.html
│   ├── my-appointments.html
│   └── book-appointment.html
```

User Roles

1. **User**

- Register and login
- View available doctors
- Book appointments
- View their booked appointments

2. **Doctor**

- Register and login
- View their appointment requests
- Confirm or cancel pending appointments

3. **Admin** *(optional)*

- View all users and doctors
- Approve/reject doctors
- Delete users

Authentication

- Token-based authentication using JWT

- Tokens stored in localStorage or sessionStorage
- Sessions expire after 30 minutes

Appointment Booking Flow

1. User logs in → browses doctors
2. Clicks “Book Now” → fills date/time form
3. Backend saves appointment with status "Pending"
4. Doctor logs in → sees appointment
5. Doctor confirms or cancels
6. Status updated to "Confirmed" or "Cancelled"
7. User sees status in their dashboard

API Endpoints

Auth (auth.js)

- POST /api/auth/register
- POST /api/auth/login

Doctors (doctors.js)

- GET /api/doctors (list all approved doctors)

Appointments (appointments.js)

- POST /api/appointments (book an appointment)
- GET /api/appointments/user/:id
- GET /api/appointments/doctor/:id
- PATCH /api/appointments/:id/status

MongoDB Collections

- users: stores user data
- doctors: stores doctor data
- appointments: stores booked appointments

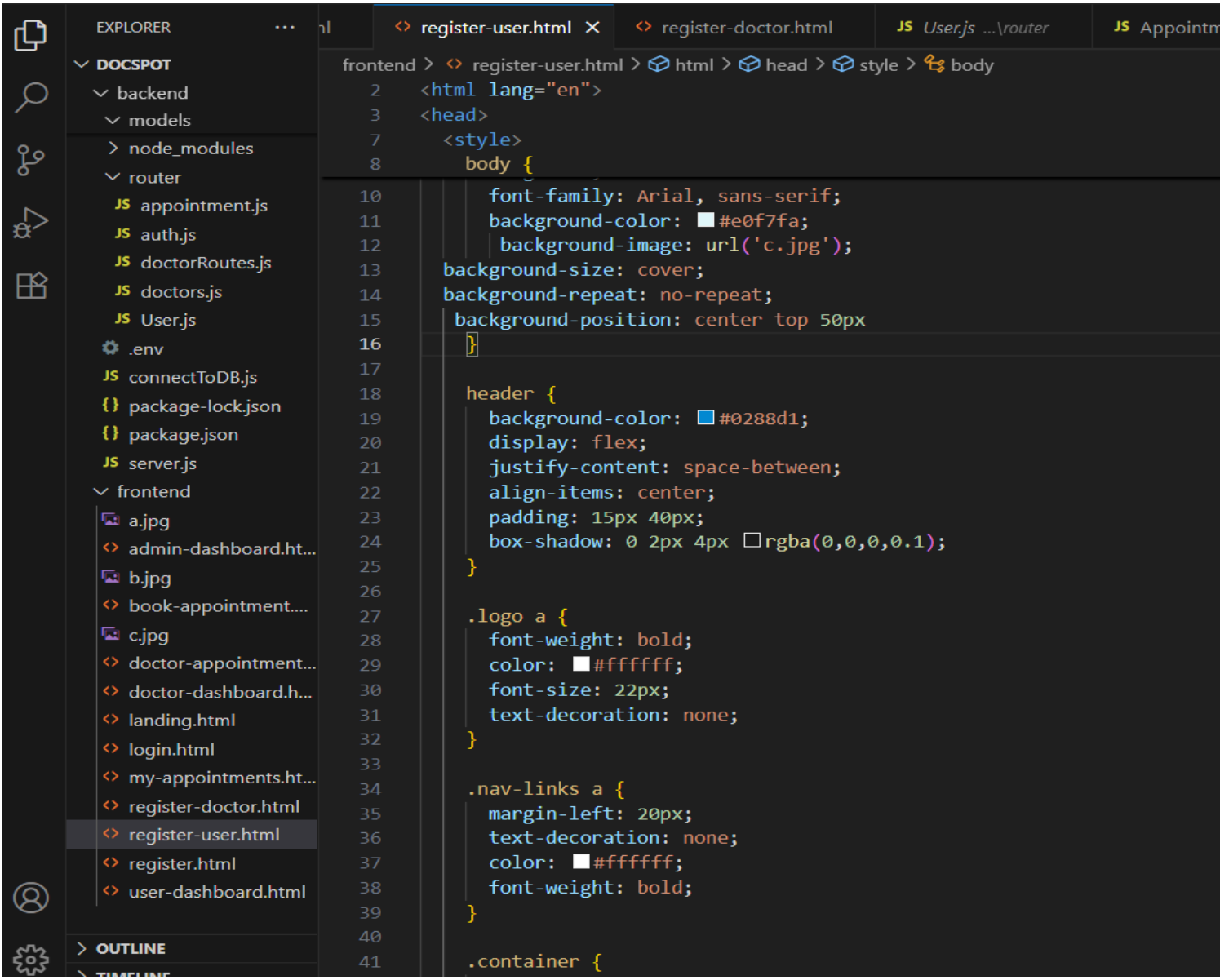
Each appointment document:

```
{
  "user": ObjectId,
  "doctor": ObjectId,
  "date": "YYYY-MM-DD",
  "time": "HH:MM",
  "status": "Pending" | "Confirmed" | "Cancelled"
}
```

Testing & Debugging

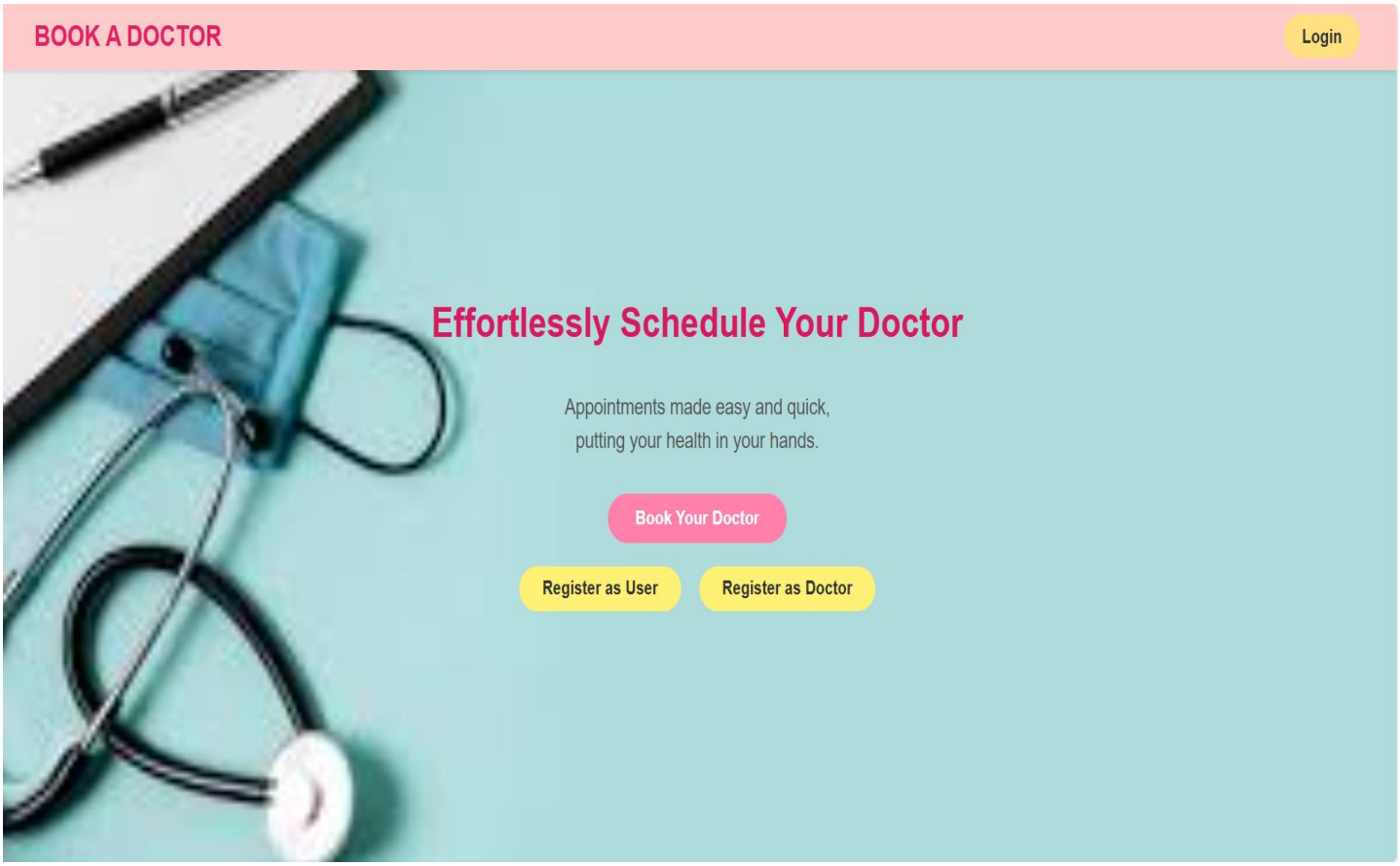
- Run backend: node server.js
- Ensure MongoDB is running on localhost:27017
- Open HTML pages using Live Server (VS Code)
- Use DevTools → Console/Network for errors
- Use Postman to test API endpoints directly

PROJECT STRUCTURE :

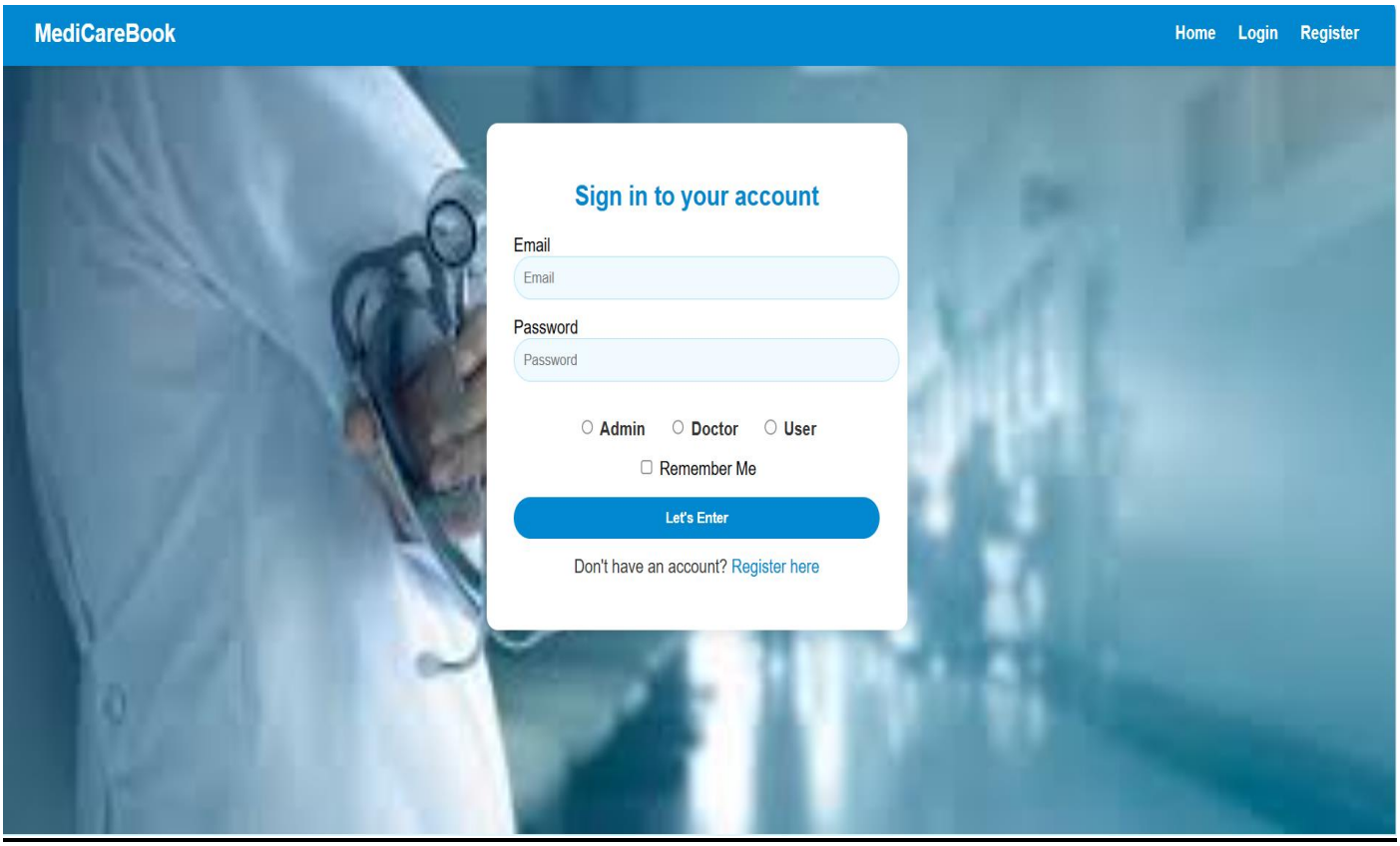


Frontend page

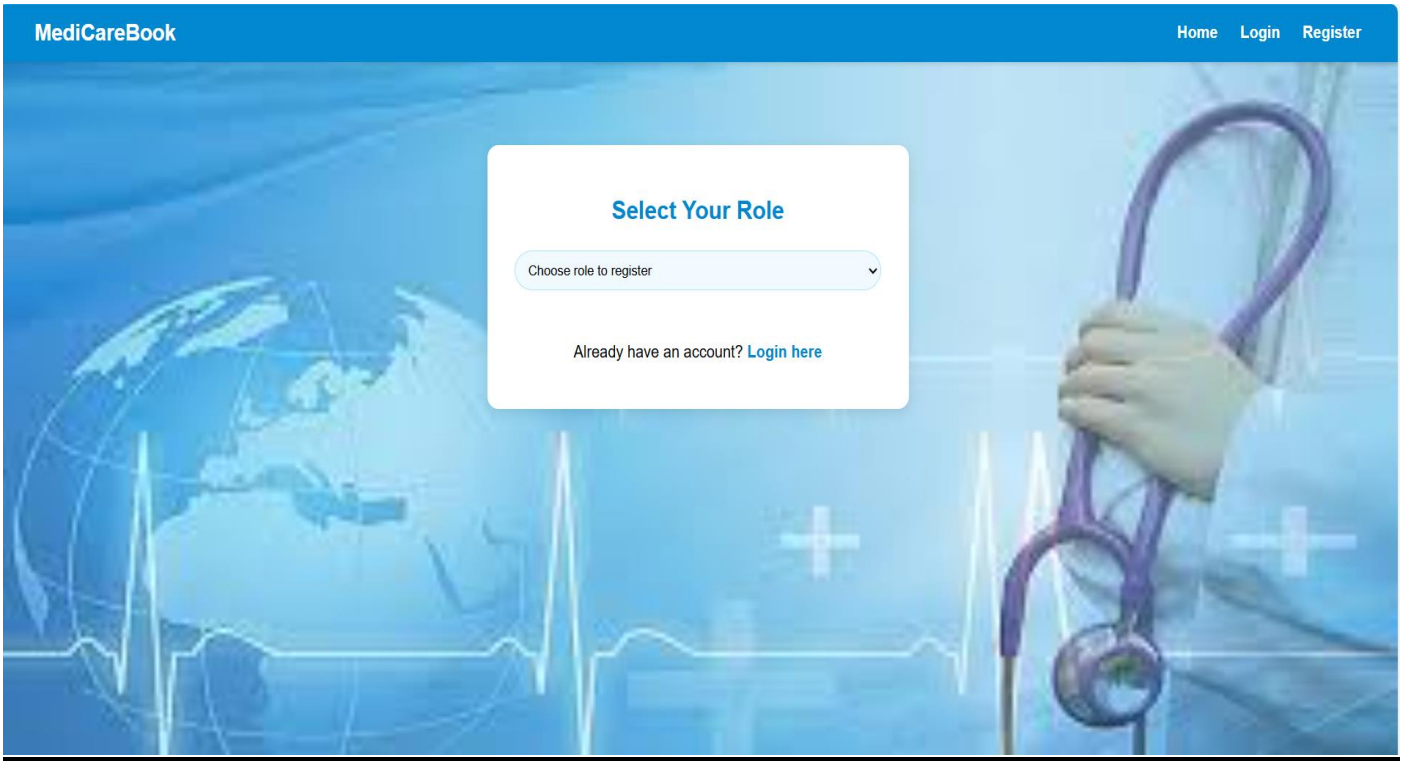
Landing.html



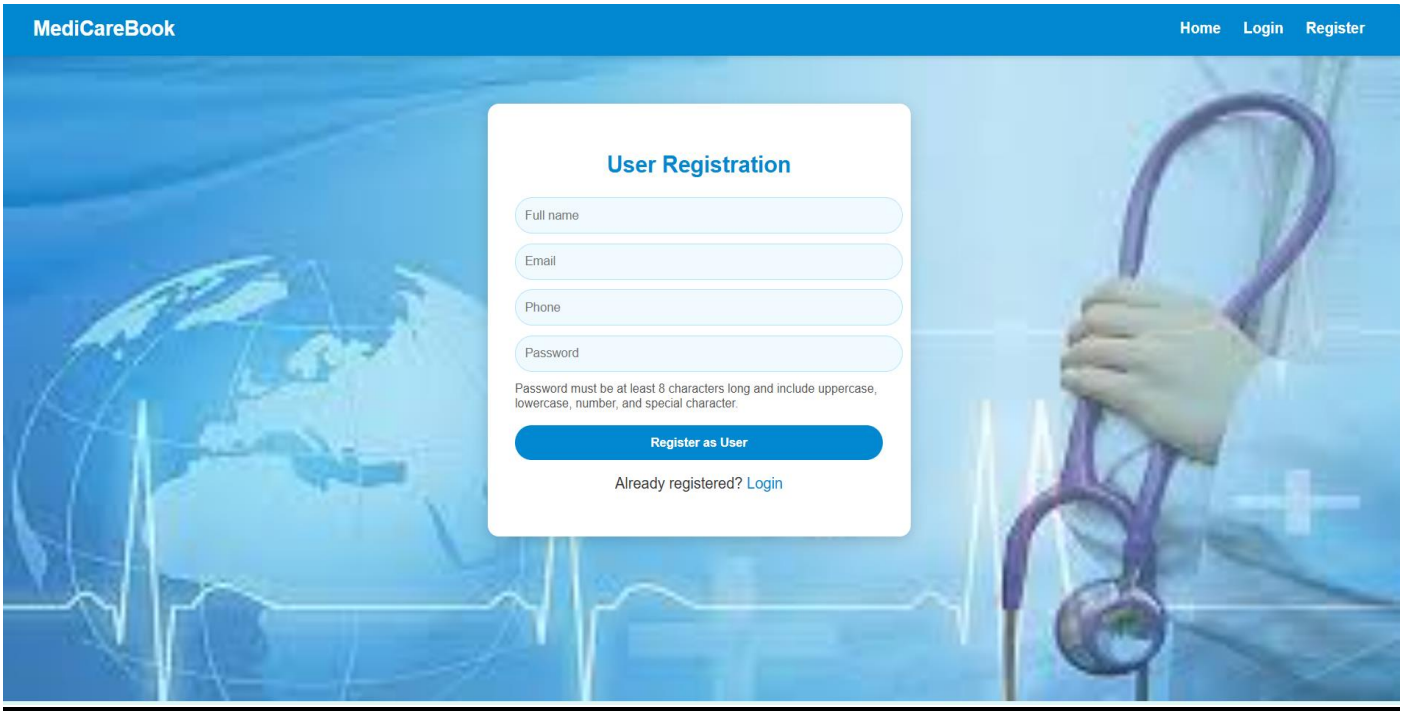
Login page



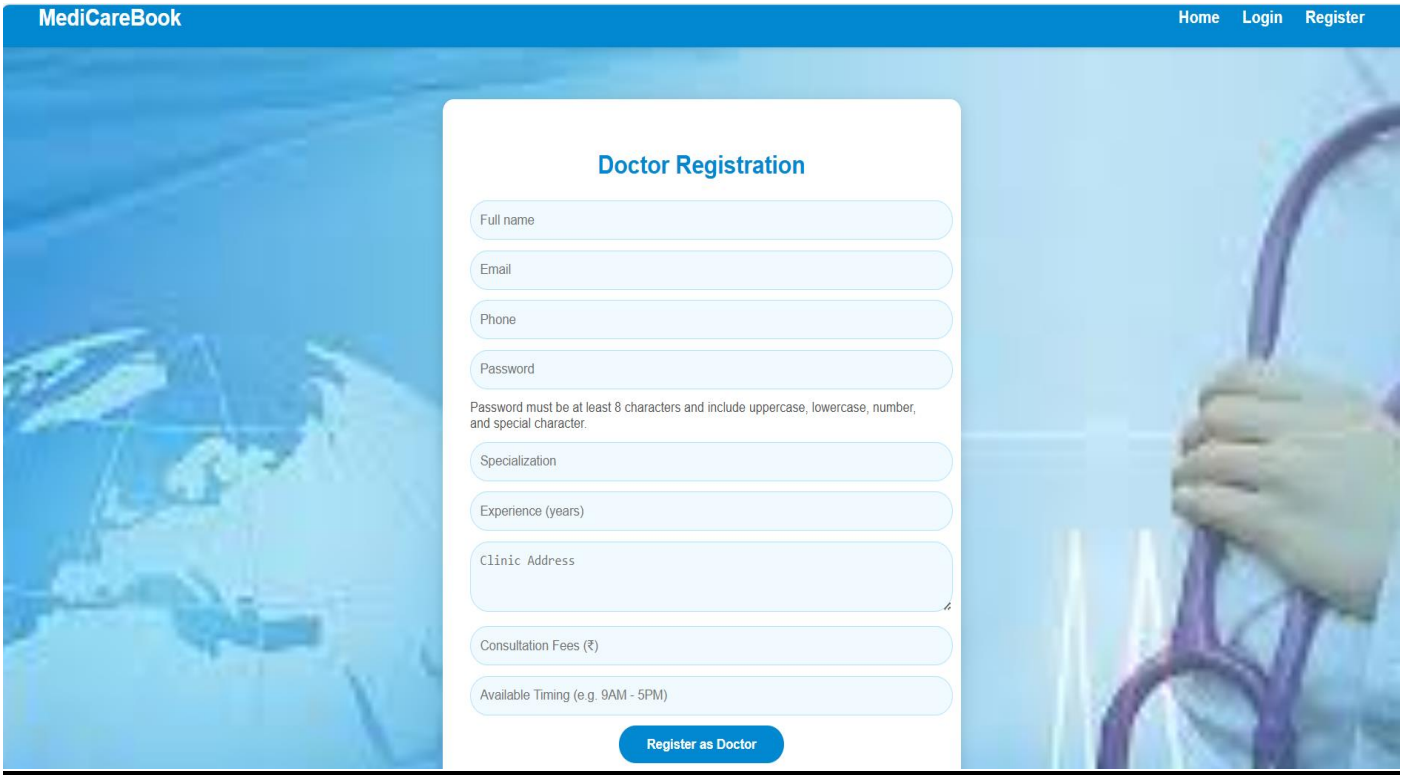
Registeration.page



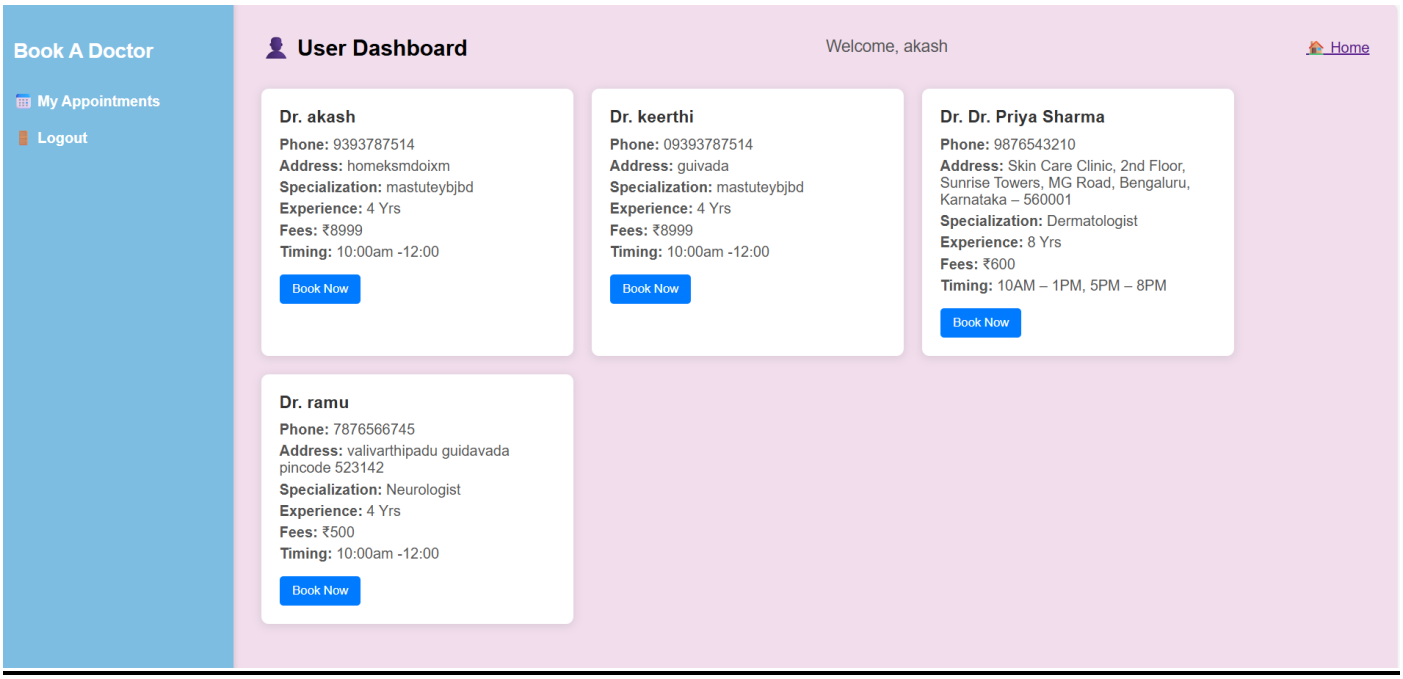
User reigsteration

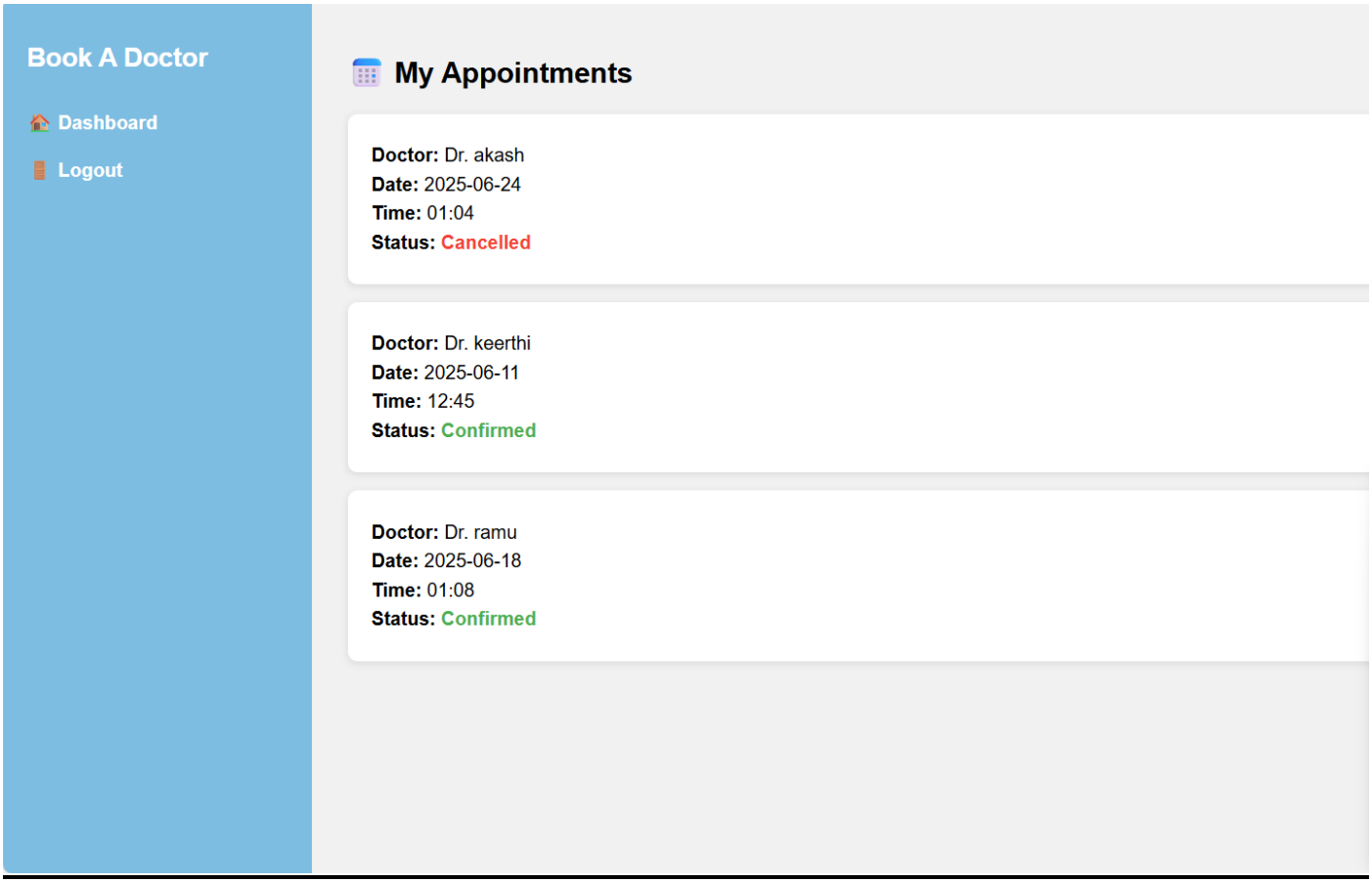


Doctor registered

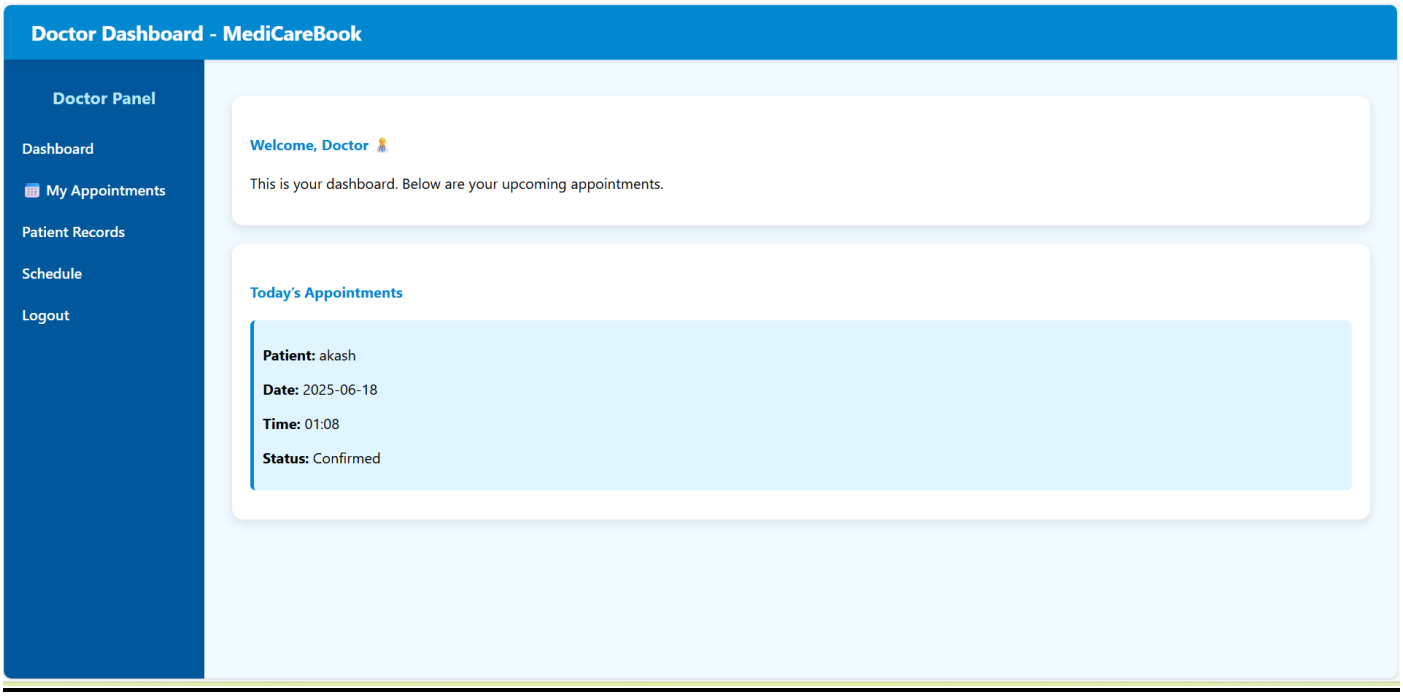


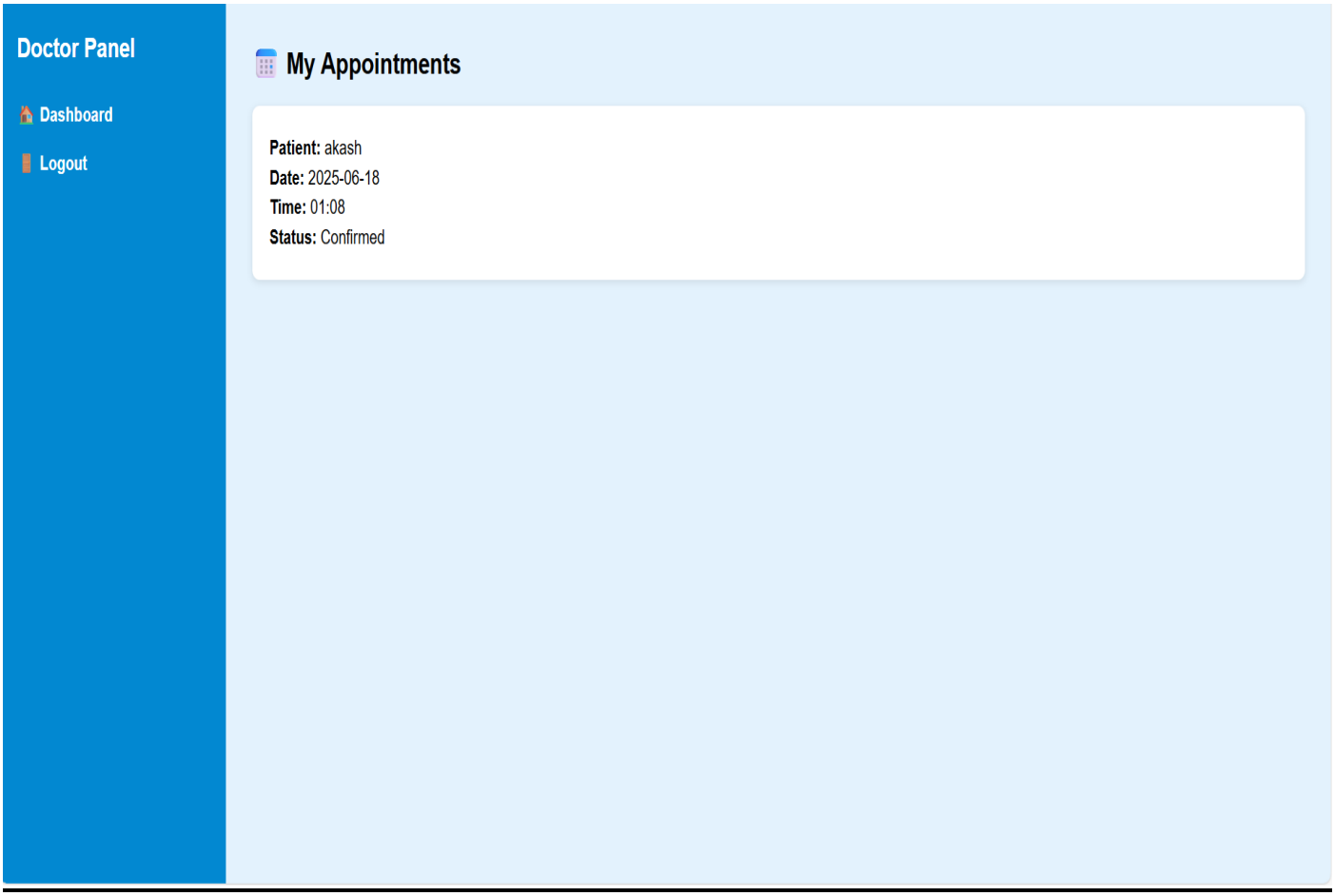
User dash borad



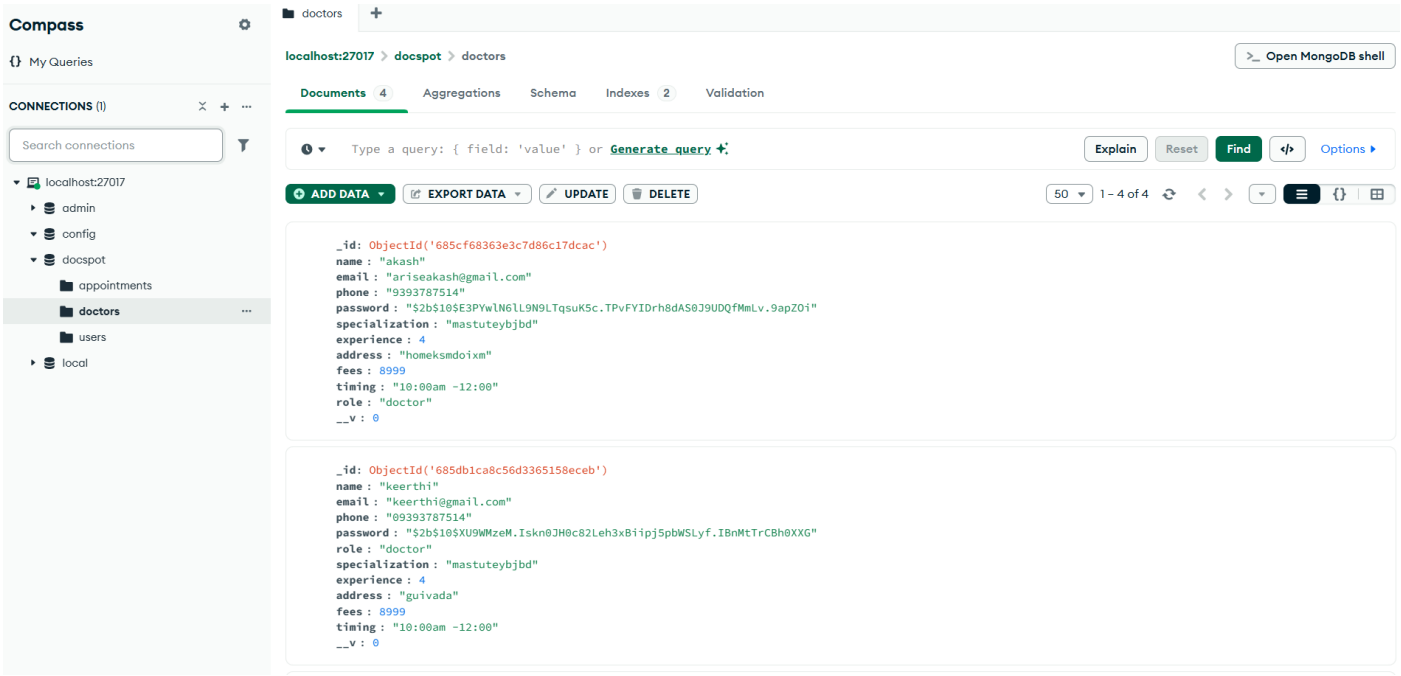


Doctor user borad page





Database page



CONCLUSION:

This project outlines the development of a comprehensive appointment booking system with user roles for customers, doctors, and admin. Each milestone—setup, backend, database, frontend, and final implementation—forms a structured foundation for a scalable application. The integrated functionalities allow for seamless appointment management, user authentication, and role-specific operations