# **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.

## **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 Collection**s**

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

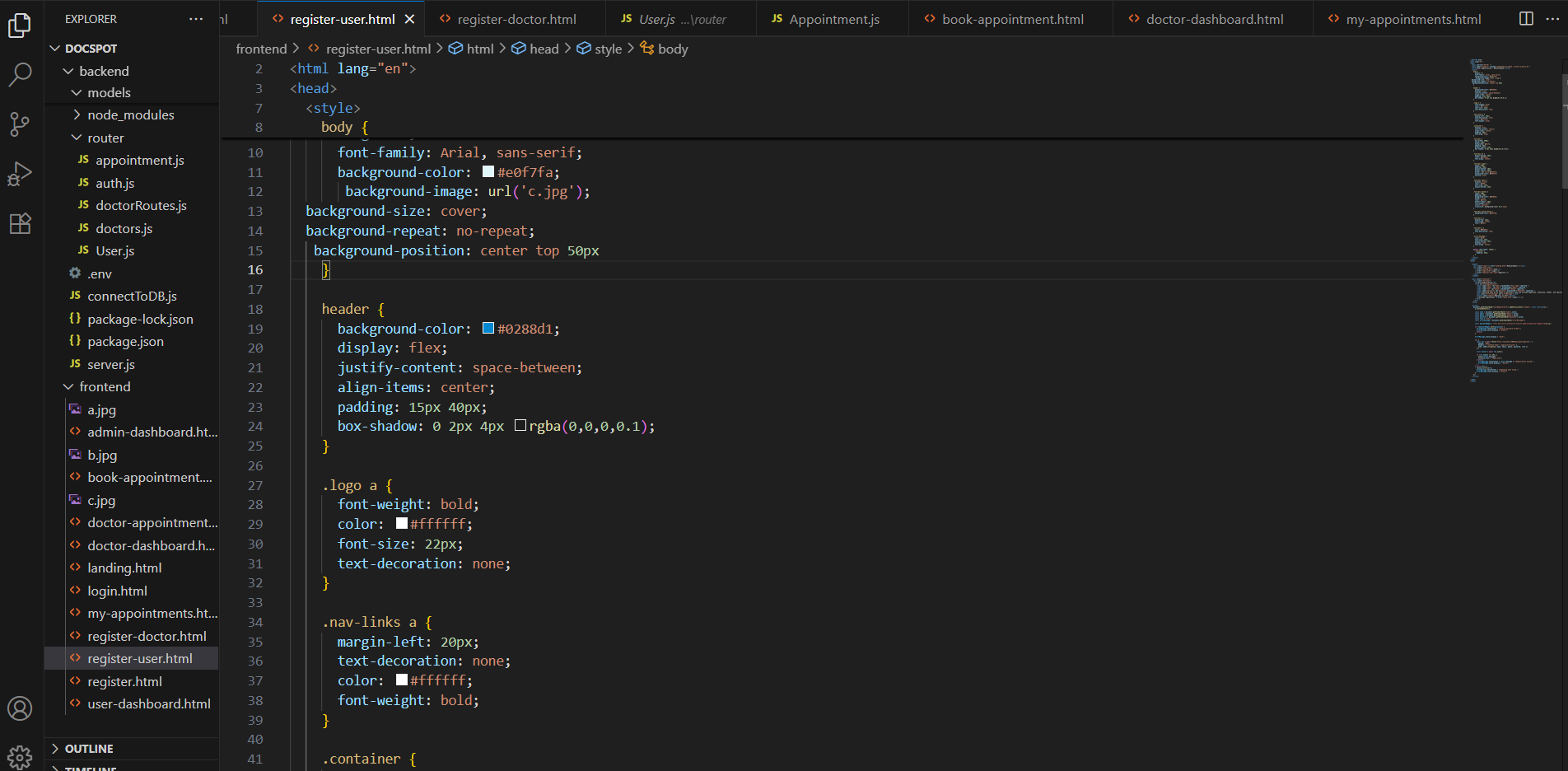
Each appointment document:

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

## Testing & Debugging

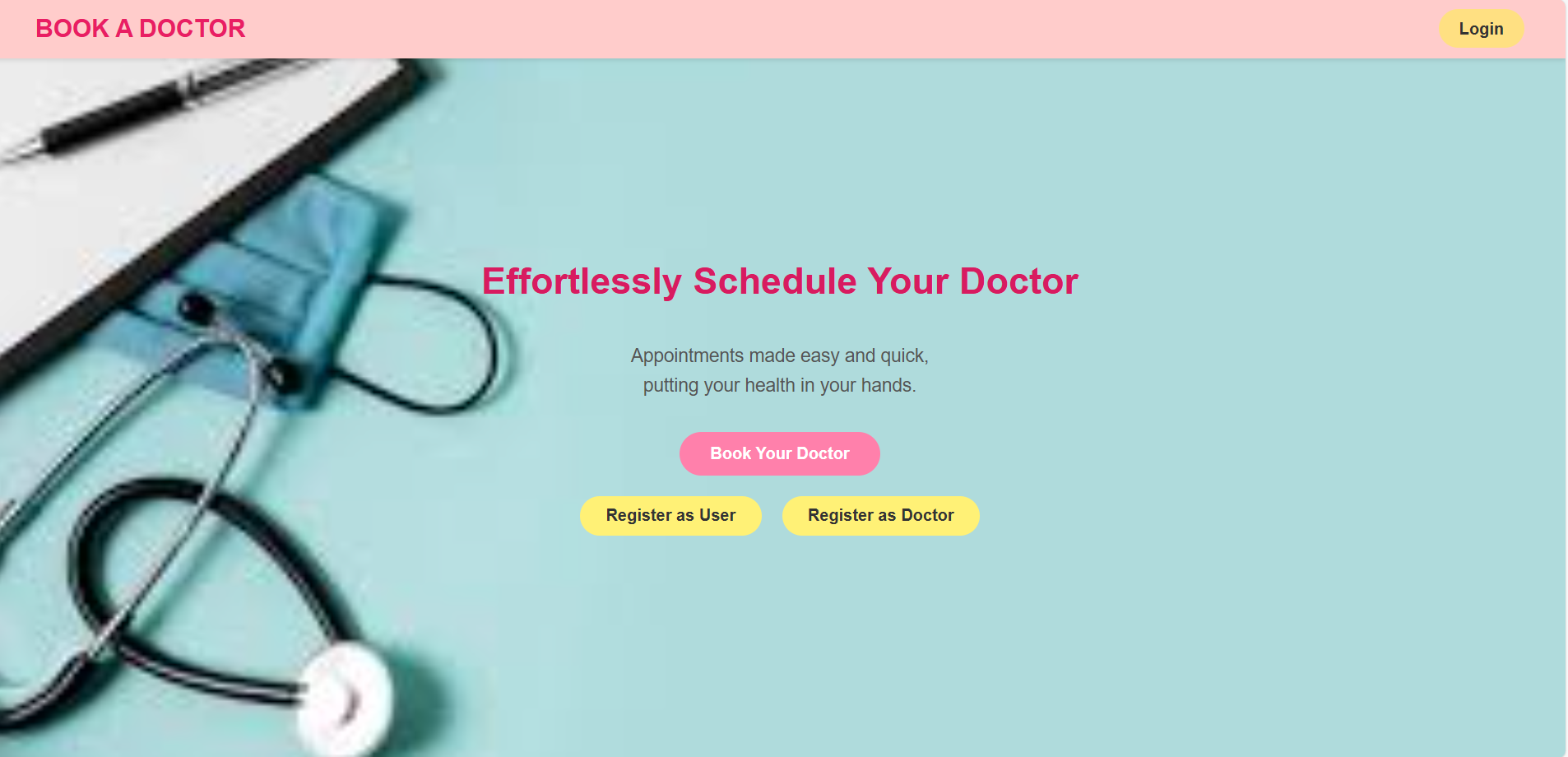
* Run backend: node server.js
* Ensure MongoDB is running on localhost:27017
* Open HTML pages using Live Server (VS Code)
* Use DevTools → Console/Network for errors
* Use Postman to test API endpoints directly

**PROJECT STRUCTURE :**

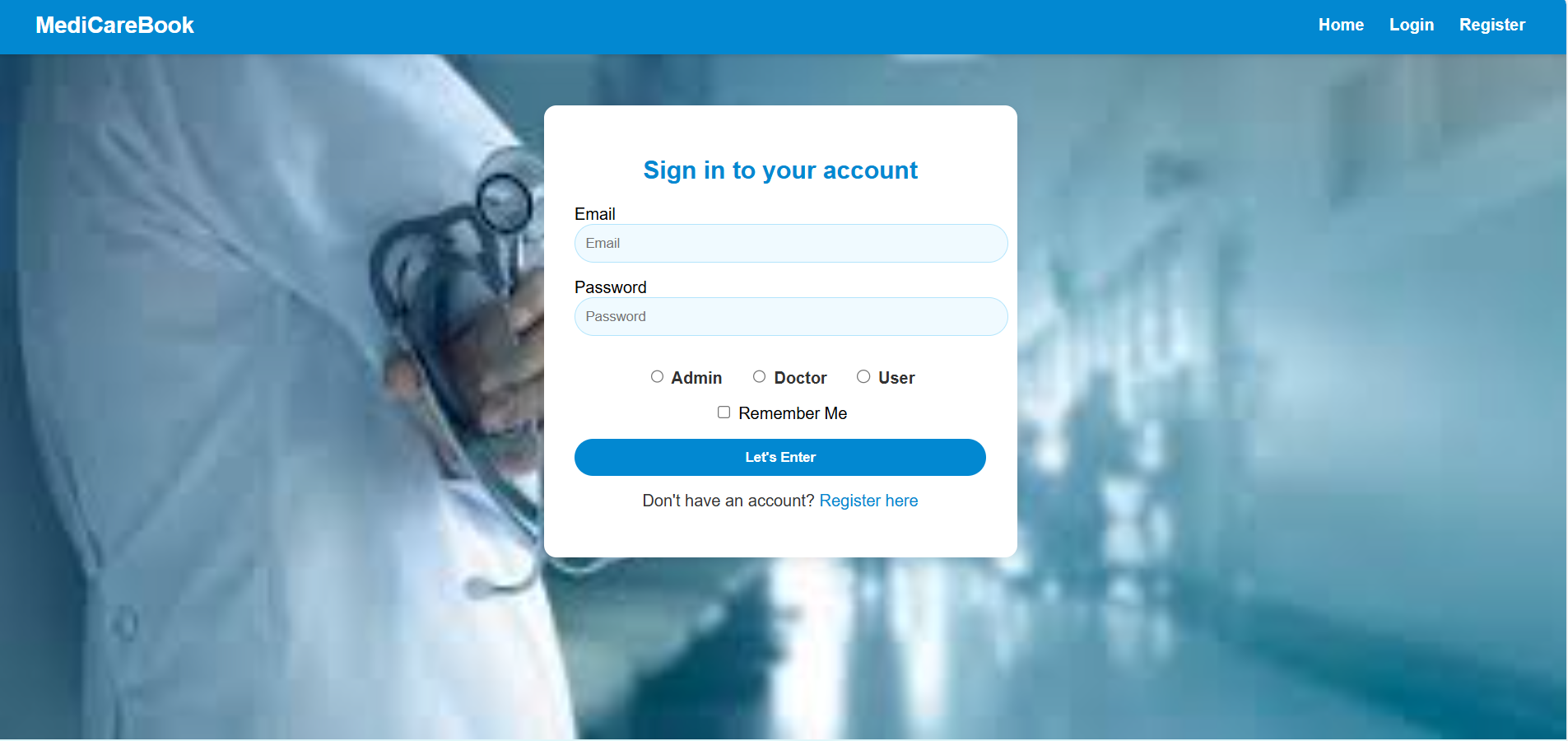


**Frontend page**

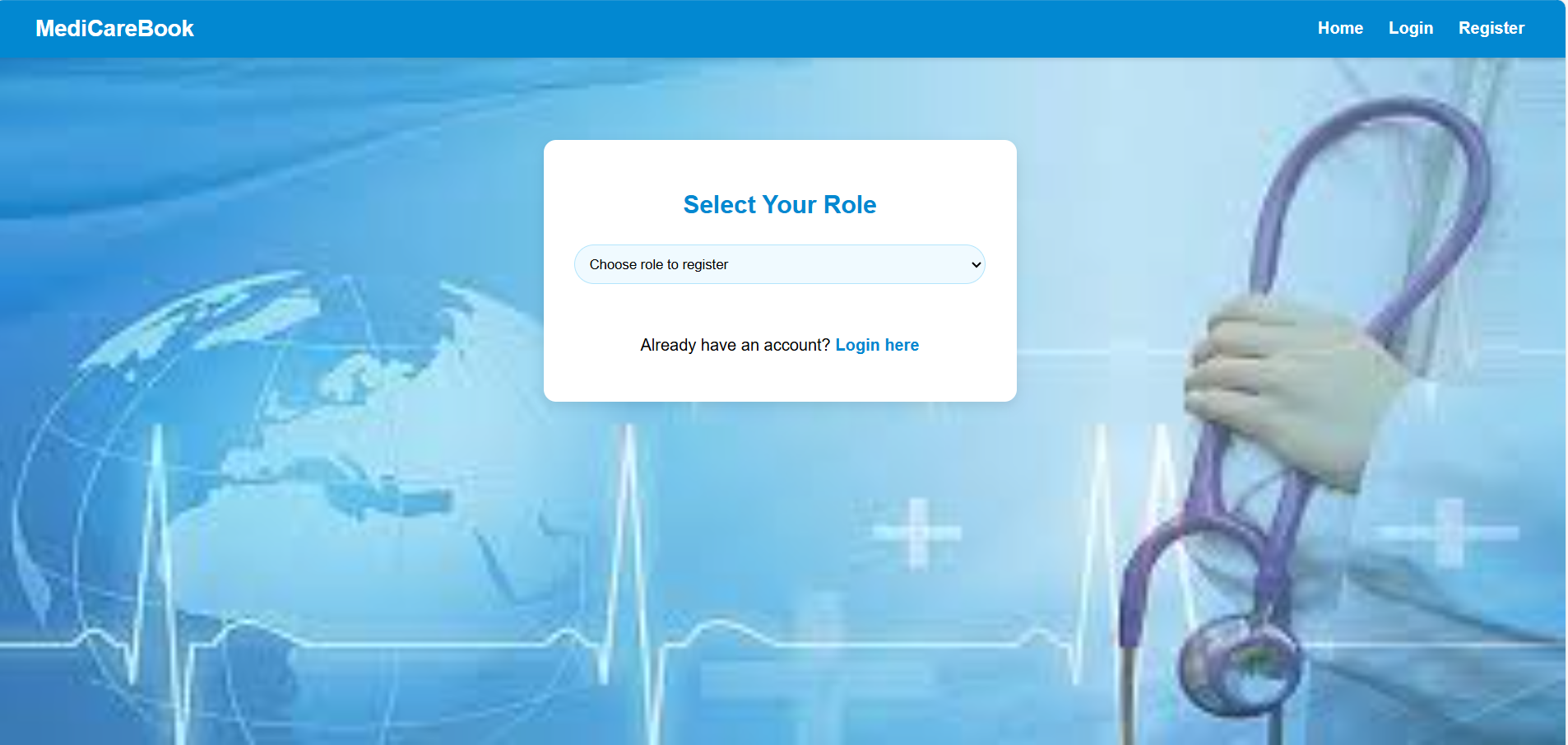
**Landing.html**



**Login page**

****

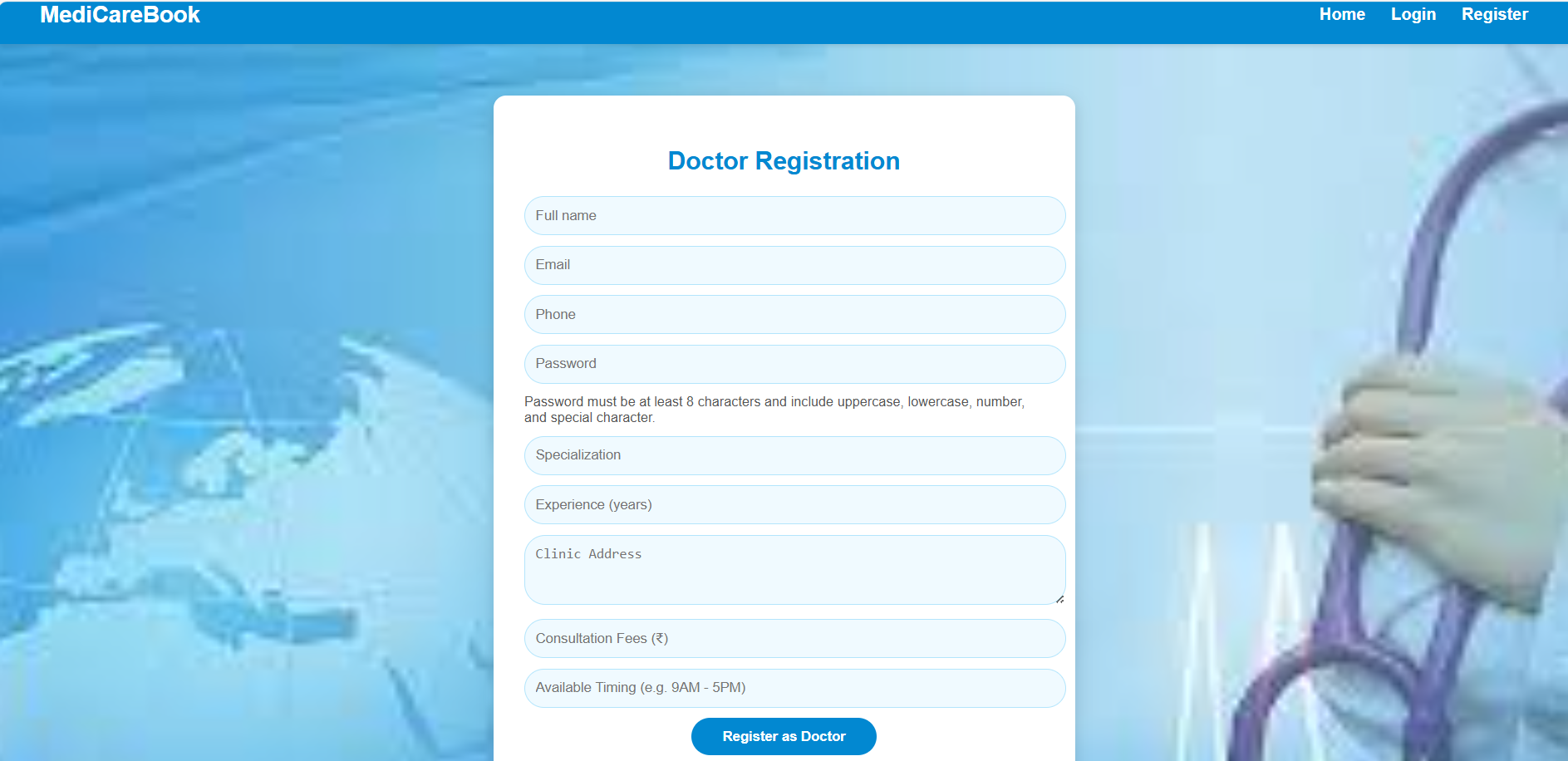
**Registeration.page**

****

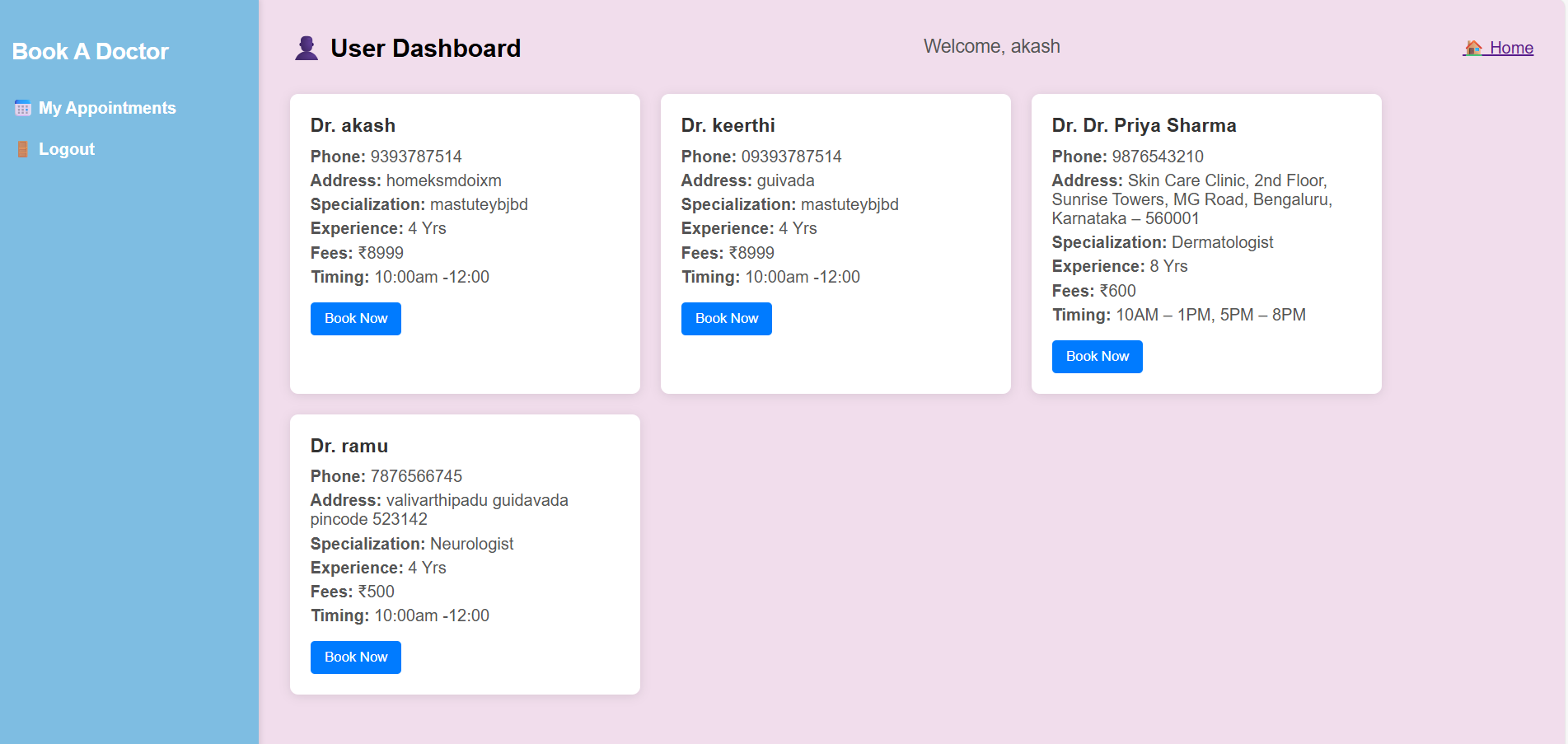
**User reigsteration**

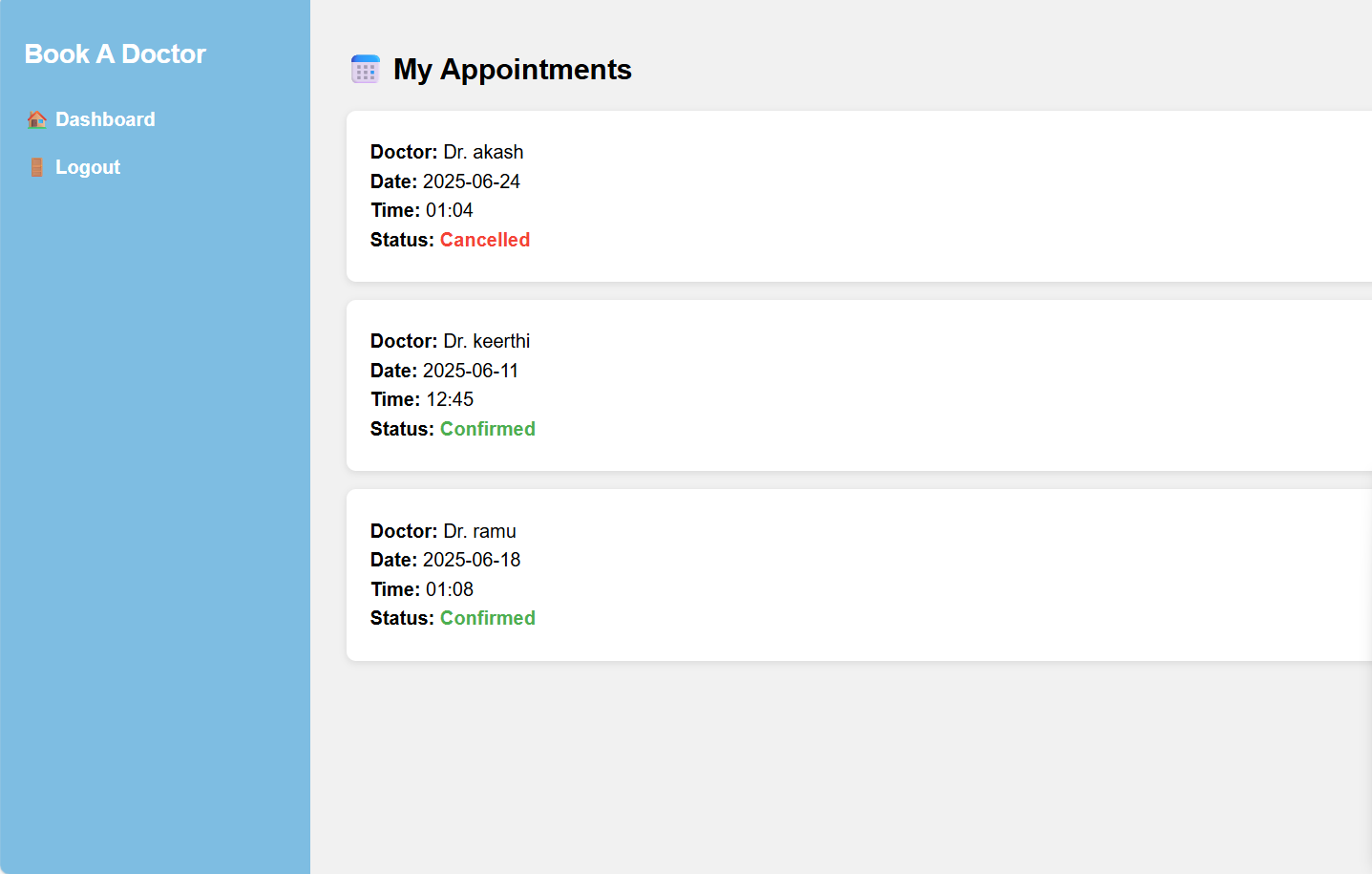
****

**Doctor registered**

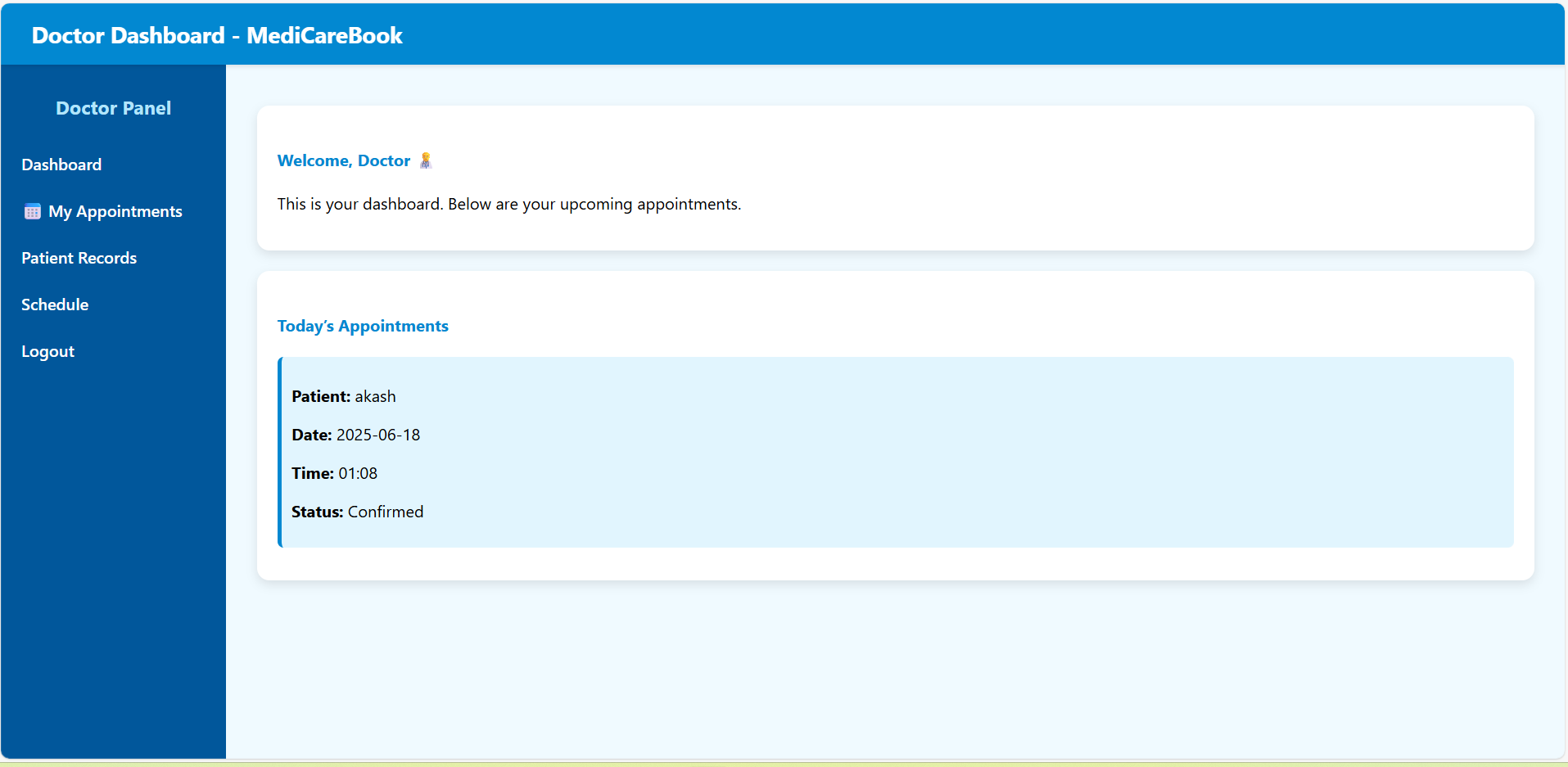
****

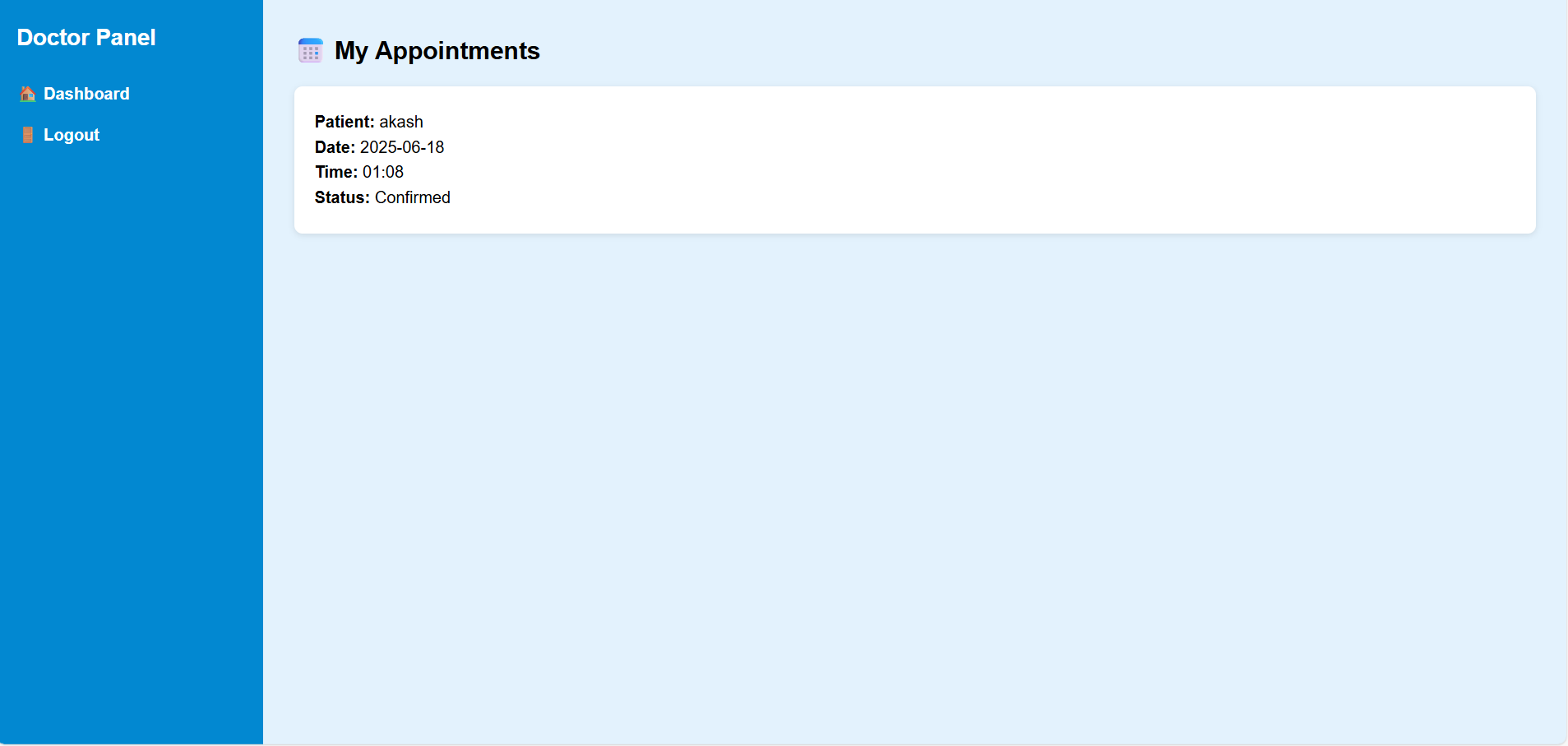
**User dash borad**

****

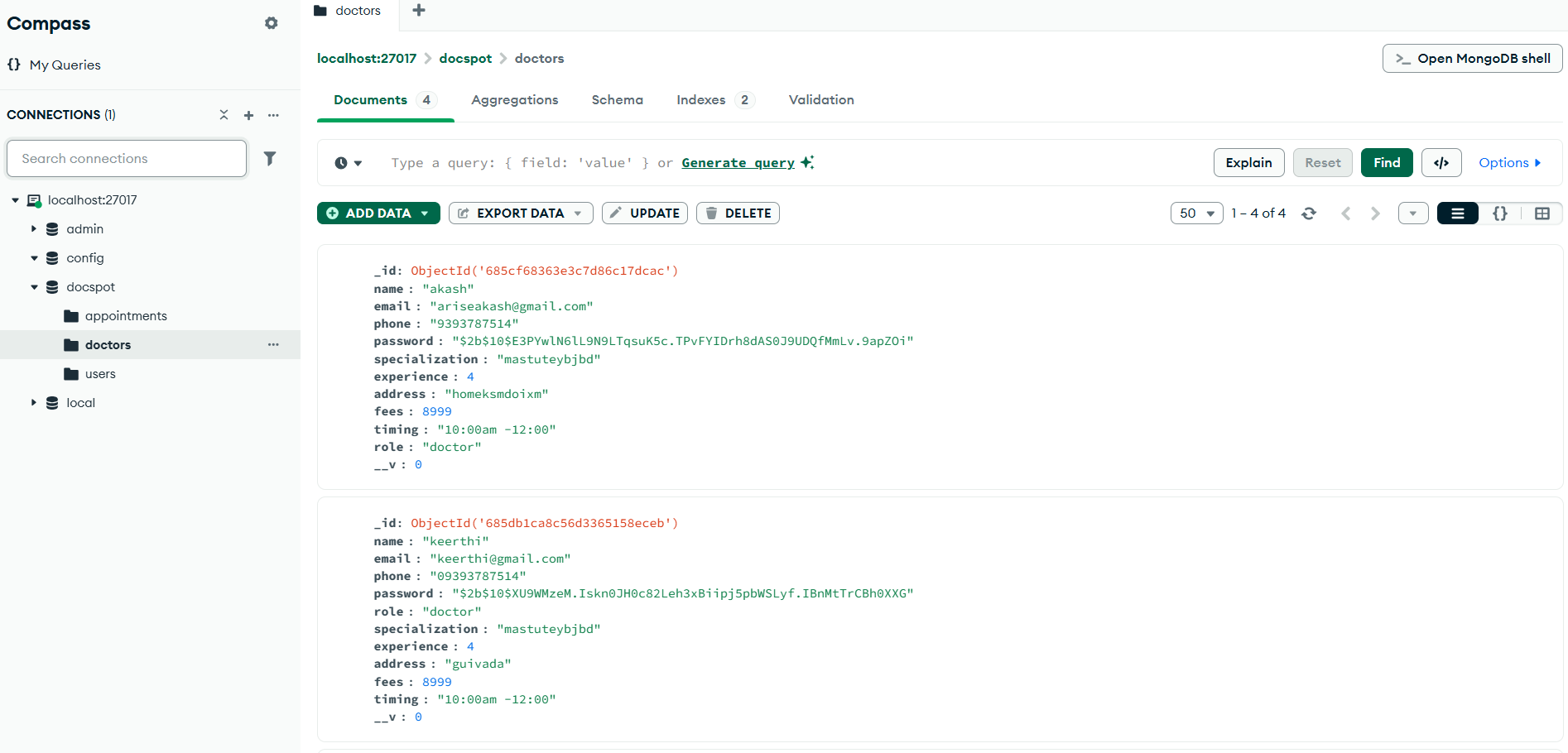
****

**Doctor user borad page**

****

****

**Database page**

****