Book a Doctor

Book a Doctor

1. Introduction

- **Project Title**: Book a Doctor
- **Team Members**:
  - ssTeam Member 1 Abinaya .S]- Frontend Developer
  - Team Member 2 Jayapratha.S]- Backend Developer
  - Team Member 3 Komathi.G - Database Administrator
  - Team Member 4 Sneha . A - Tester
- **Team Code:** 10323

2. Project Overview

- **Purpose**:

The "Book a Doctor" project aims to simplify the process of finding and scheduling appointments with doctors. It provides an intuitive interface for patients to browse doctors by specialty, location, and availability.

- **Features**:
  - User registration and login with authentication.
  - Search and filter doctors by specialization and location.
  - Appointment scheduling with calendar integration.
  - Real-time updates on appointment status.

3. Architecture

- **Frontend**:

Developed using React.js for a dynamic and user-friendly interface. It includes components for registration, doctor profiles, and appointment booking.

- **Backend**:

Built with Node.js and Express.js to handle API endpoints, authentication, and business logic.

- **Database**:

MongoDB stores user data, doctor profiles, and appointment records. It utilizes schemas for structured data handling.

4. Setup Instructions

- **Prerequisites**:
  - Node.js installed on your system.
  - MongoDB running locally or on a cloud provider.
- **Installation**:
  - Clone the repository: `git clone [repository_url]`
    ```
    cd book-a-doctor
    ```
  - Install dependencies for both frontend and backend: `cd client`
    ```
    npm install
    cd ../server
    npm install
    ```
  - Set environment variables:
    - Create a `.env` file in the server directory with the following:
      ```
      PORT=5000
      MONGO_URI=your_mongodb_connection_string
      JWT_SECRET=your_secret_key
      ```
  - Start MongoDB server and ensure it's running.

5. Folder Structure

- **Client**:

The React frontend includes the following structure:

```
src/
├── components/
├── pages/
├── services/
├── App.js
└── index.js
```

- **Server**:

The Node.js backend includes:

```
src/
├── controllers/
├── models/
├── routes/
├── middlewares/
└── server.js
```

6. Running the Application

- **Frontend**:

Run the following command in the client directory:

```
npm start
```

- **Backend**:

Run the following command in the server directory:

```
npm start
```

7. API Documentation

- **Endpoints**:
  - **User Registration**:
    - `POST /api/users/register`
    - Parameters: `{ name, email, password }`
    - Response: `{ success, userId }`
  - **Login**:
    - `POST /api/users/login`
    - Parameters: `{ email, password }`
    - Response: `{ success, token }`
  - **Fetch Doctors**:
    - `GET /api/doctors`
    - Response: `{ doctors: [ { id, name, specialty, location } ] }`

8. Authentication

- Authentication is handled using JSON Web Tokens (JWT).
- Users receive a token upon login, which is stored in local storage and sent with requests to protected endpoints.

9. User Interface

- The interface includes:
  - A homepage to search and filter doctors.
  - A profile page for doctors displaying availability and reviews.
  - A booking page with a calendar view.

Screenshots will be added here to showcase the interface.

10. Testing

- **Testing Strategy**:

Manual and automated testing for critical components.

- **Tools Used**:
  - Postman for API testing.
  - Jest for unit tests in the backend.

11. Screenshots or Demo

- Link to a live demo: https: //drive.google.com/file/d/1-RSa2yp2xc60g5JZwF6f-A63LTF0OhFP/view?usp=drive_link

- Add images or videos of the working application.

12. Known Issues

- The search functionality may take longer for large datasets.
- Email notifications for appointment confirmation are under development.

13. Future Enhancements

- Implement video consultation functionality.
- Add payment integration for online booking fees.
- Optimize the search feature with caching mechanisms.