Documentation for the Project: Book a Doctor

1. Introduction

Project Title: Book a Doctor

Team Members:

- Elden Nicholas.N 211221205004(Frontend Developer)
- Arjun .P 211221205001(Backend Developer)
- Bahavathy sen.211221205002[(UI/UX Designer)
- Deepika .J 211221205004(Tester)

2. Project Overview

Purpose:

Book a Doctor is a web application designed to simplify the process of scheduling medical appointments. It bridges the gap between patients and doctors, providing a seamless, efficient, and secure platform to book consultations.

Features:

- User-friendly interface for patients to browse doctors by specialty.
- Doctor profiles with ratings, experience, and availability.
- Instant appointment booking with live schedule updates.
- Secure login and account management for doctors and patients.
- SMS/Email reminders for upcoming appointments.

3. Architecture

Frontend:

Built using **React.js**, the frontend features a responsive design to cater to all devices. It uses state management with React Context API and dynamic routing with React Router.

Backend:

The backend is developed using **Node.js** and **Express.js**. RESTful APIs handle user requests, doctor data, and appointment management.

Database:

The application uses **MongoDB** to store user data, doctor profiles, and appointment schedules. The schema includes collections for users, doctors, and bookings, ensuring efficient queries and relations.

4. Setup Instructions

Prerequisites:

- Node.js
- MongoDB
- Git

Installation:

1. Clone the repository:
   git clone https://github.com/yourusername/book-a-doctor.git
2. Navigate to the project directory.
3. Install dependencies:
   - For frontend: cd client && npm install
   - For backend: cd server && npm install
4. Set up environment variables in .env for database connection and API keys.
5. Folder Structure

```
client/
├── src/
│   ├── components/
│   ├── pages/
│   ├── context/
│   ├── utils/
│   └── App.js
└── public/
server/
├── controllers/
```

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

Authentication

Method: JWT tokens are used for authentication and authorization.

Tokens are issued at login and stored securely in HTTP-only cookies.

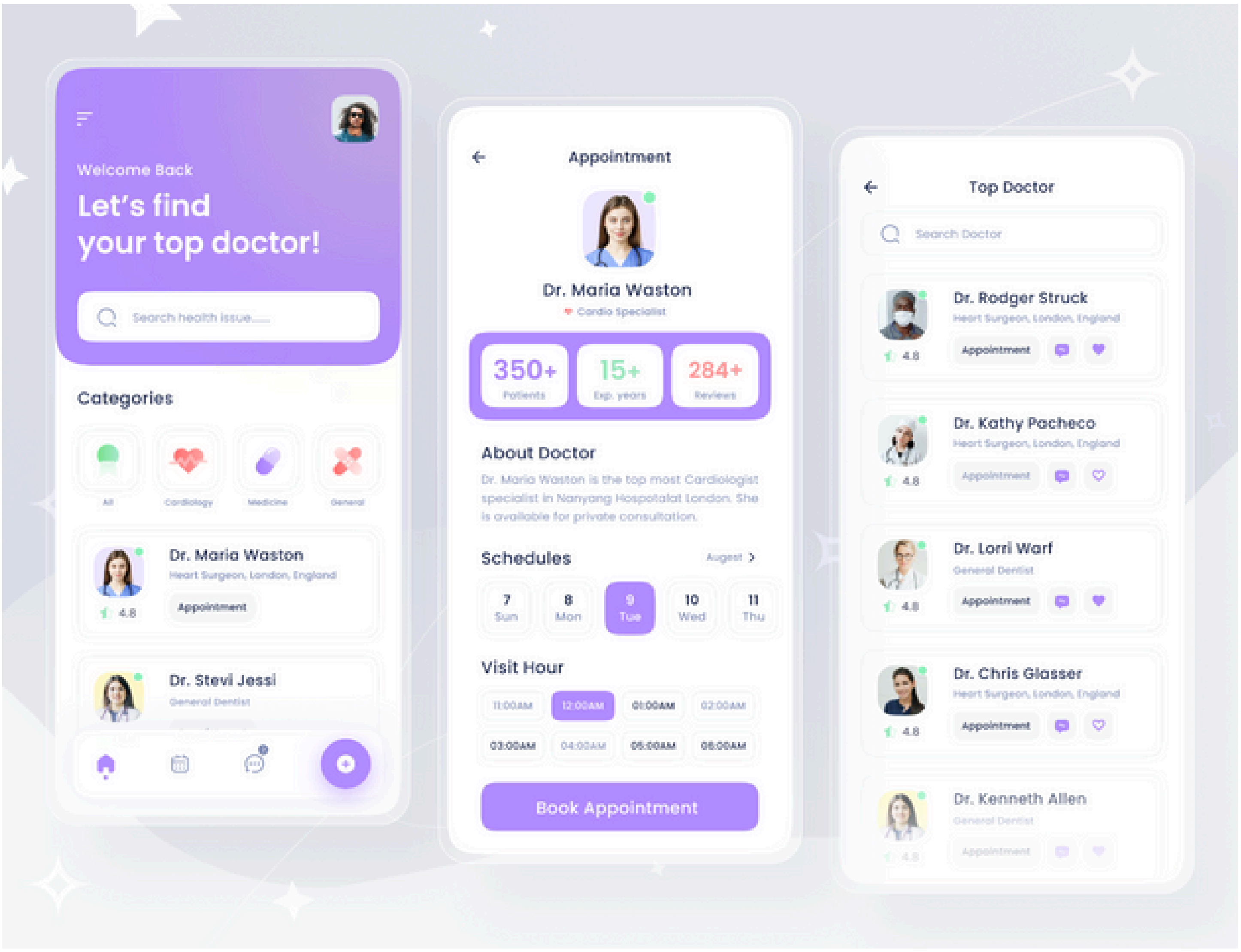
Middleware checks validate tokens for protected routes.

User Interface

Dashboard with doctor listings and filters.

Booking page with a calendar view.

Account settings page for patients and doctors.



Testing Strategy:

Unit testing for components using Jest.

Integration testing for API endpoints using Postman.

End-to-end testing using Cypress

API Documentation

Endpoint	Method	Description	Parameters	Example Response
/api/doctors	GET	Fetch all doctors	None	JSON List of Doctors
/api/bookings	POST	Book an appointment	User ID, Doctor ID, Date	JSON Confirmation
/api/auth/login	POST	Login user	Email, Password	Auth Token

Known Issues

Minor delays in appointment confirmation due to server-side processing.

Lack of offline support.

Future Enhancements

- Add a feature for telemedicine consultations via video calls.
- Implement AI-based doctor recommendations based on patient history.
- Include support for multi-language interfaces.
- Develop a mobile app version of the platform.