FULL STACK DEVELOPMENT WITH MERN

1.Introduction

- **Project Title**: Book Your Doctor
- **Team Members**:SUBASH R , VASANTH M ,NAVEEN K , VELMURUGAN M , SETHU AKASH

2.Project Overview

- **Purpose:**

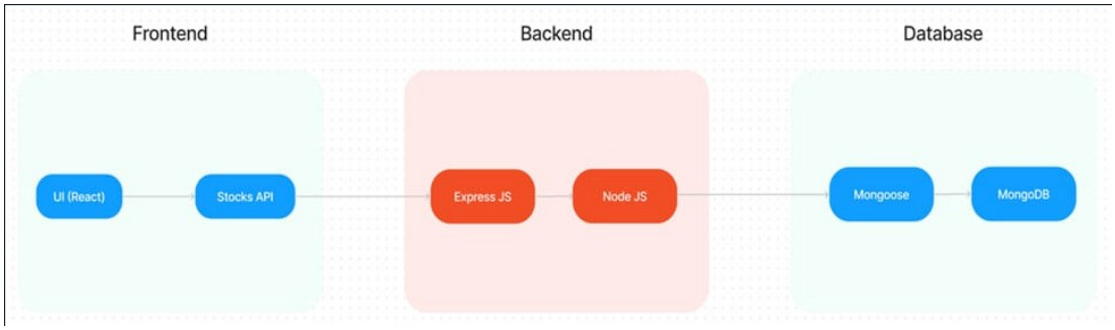
The **purpose** of the "Book Your Doctor" project is to provide a **convenient, efficient, and accessible way for patients to schedule medical appointments** with healthcare professionals.

This platform aims to modernize the healthcare appointment process by leveraging technology to address several key challenges in the healthcare industry**.**

- **Features:**

Build a "Book Your Doctor" platform using MERN (MongoDB, Express, React, Node.js) stack, where users can browse doctors, view profiles, and book appointments. Include authentication, doctor availability, and appointment management features for a seamless booking experience.

3.Architecture



- **Frontend:**

Build a user interface where patients can search for doctors, view profiles, select time slots, and confirm appointments.

- **Backend:**

Doctor Listing and Scheduling: MongoDB stores doctor profiles and available time slots, and an endpoint (/doctors) retrieves this data for users.

- Booking Appointments: Another endpoint (/appointments) allows users to book slots, saving appointment details to MongoDB with JWT-based user authentication

- **Database:**

Create REST APIs for managing doctor profiles, appointment slots, and bookings, and store data in MongoDB. Implement authentication using JWT for secure access.

4.Setup Instruction

- **Prerequisites:**

User Registration and Login (Patients and Doctors).

- Profile Management: For both patients and doctors.
- Doctor Search: Based on specialty, location, or availability.
- Booking System: Appointment scheduling with time slots.
- Notifications: Email or SMS reminders for appointments.
- Admin Panel: For managing users and data.

- ***Installation:***

Open the Google Play Store.

- Search for the **"Book Your Doctor"** admin app (or the app's specific name for providers).
- Tap on the app, then click **Install**.
- Once installed, log in using your doctor's account or create an account if you don't have one.
- Follow the prompts to set up your profile, including uploading your **qualification details**, **specializations**, **available time slots**, and **payment settings**.
- You'll be able to view patient bookings, manage your schedule, and confirm or reschedule appointments.

5.Folder structure

Client:

The **client** in the "Book Your Doctor" platform refers to two primary groups: **patients** seeking convenient and accessible healthcare and **healthcare providers** managing their schedules and offering services.

Both groups benefit from the platform's features, with patients enjoying ease of access and healthcare providers benefiting from streamlined administrative tools, patient management systems, and

telemedicine options. The overall aim of the platform is to bridge the gap between healthcare professionals and patients, creating a seamless and efficient experience for both client and medical field.

Server

The server in the "Book Your Doctor" platform is the backbone of the system, responsible for handling all requests from patients and healthcare providers, processing data, managing appointments, handling payments, and supporting telemedicine sessions.

A well-structured, scalable, and secure server infrastructure ensures that the platform can provide a seamless, reliable, and efficient user experience for both patients and healthcare providers.

6. Running The Application

Frontend

Navigate to the client directory.
Start the React app:
bash npm start

Backend

Navigate to the server directory.
Start the server:
bash npm start

7. Api Documentation

User Authentication (JWT):

POST /api/auth/register: Register new users.
POST /api/auth/login: User login, returns JWT token.
Doctor Management:

GET /api/doctors: Fetch all available doctors.
POST /api/doctors/add: Admin adds a new doctor (auth required).
Appointment Booking:

POST /api/appointments/book: Book an appointment with a doctor (auth required).
GET /api/appointments/user: Get user-specific appointment history.
Reviews and Feedback:

POST /api/reviews: Add a review for a doctor (auth required).
GET /api/reviews/
: Fetch reviews for a specific doctor.

8.Authentication

User Authentication (JWT):

POST /api/auth/register: Register new users.

POST /api/auth/login: User login, returns JWT token.

Doctor Management:

GET /api/doctors: Fetch all available doctors.

POST /api/doctors/add: Admin adds a new doctor (auth required).

Appointment Booking:

POST /api/appointments/book: Book an appointment with a doctor (auth required).

GET /api/appointments/user: Get user-specific appointment history.

Reviews and Feedback:

POST /api/reviews: Add a review for a doctor (auth required).

GET /api/reviews/

: Fetch reviews for a specific doctor.

9.User Interface

he user interface for a "Book Your Doctor" app using the MERN stack can feature a clean, responsive design with an easy-to-navigate dashboard for booking appointments, doctor search, and viewing medical history. It should include interactive forms, real-time availability updates, and secure login for users and healthcare providers.

10. Testing

1. Testing Strategy

a. Unit Testing

Objective: Test individual components/modules in isolation (e.g., React components, backend API endpoints).

Scope:

React Components: Validate proper rendering, state management, and event handling.

Backend APIs: Test each API endpoint for expected inputs and outputs.

Utility Functions: Test helper functions, validations, and other logic.

b. Integration Testing

Objective: Test the interaction between different modules (e.g., frontend React consuming backend APIs).
Scope:
API Integration: Validate frontend calls to backend endpoints.
Database Integration: Test database queries and responses with the backend.
c. End-to-End (E2E) Testing

Objective: Simulate real user workflows across the entire stack.
Scope: Booking a doctor appointment: Search, select, and confirm booking.
User Authentication: Sign up, login, and profile management.
Admin and Doctor Panel: Test dashboard functionalities.

d. Performance Testing

Objective: Ensure the system performs well under load.
API Response Times: Ensure APIs respond within acceptable limits.
Stress Testing: Simulate high user traffic on appointment booking.

e. Security Testing

Objective: Identify vulnerabilities in the application.
Scope: Test for SQL injection, XSS, CSRF, etc.
Validate role-based access control (e.g., only doctors can access the doctor panel).

2. Tools Used
Frontend Testing Tools
Jest:
Used for unit testing React components and utility functions.
Offers snapshots to ensure UI consistency.
React Testing Library:
Provides utilities to test DOM elements and component behavior.
Backend Testing Tools
Mocha & Chai:
Mocha is a test runner, and Chai is an assertion library for testing Node.js APIs and logic.
Supertest:
Simplifies testing HTTP API endpoints in Node.js applications.
Database Testing Tools
MongoDB Memory Server:
Provides an in-memory MongoDB instance for testing database operations without affecting production data.
Integration & End-to-End Testing Tools
Postman/Newman:
For manual and automated API testing workflows.
Cypress:
End-to-end testing framework for automating user flows like booking appointments or managing doctor profiles.
Selenium:

Can be used for browser-based testing for real-world scenarios.
Performance Testing Tools
Apache JMeter:
Load testing tool to simulate multiple concurrent users.
k6:
For stress testing APIs and evaluating performance under heavy traffic.
Security Testing Tools
OWASP ZAP (Zed Attack Proxy):
Identify vulnerabilities like XSS and CSRF in the application.
npm Audit:
Checks for vulnerabilities in the project dependencies.
Workflow Example
Unit Testing (Frontend & Backend):
Write Jest and Mocha tests for each React component and Node.js function.
Mock external API calls using libraries like Axios Mock Adapter.
Integration Testing:
Use Supertest with Mocha/Chai to validate API communication between frontend and backend.
E2E Testing:
Automate scenarios using Cypress (e.g., user booking a doctor).
Create comprehensive test cases for each user flow.
Continuous Testing:
Integrate tests with CI/CD pipelines using tools like GitHub Actions or Jenkins.
Run automated tests on every code push to ensure no regressions.

11. Screenshot or Demo Link

https://drive.google.com/file/d/1xGPpfQguc52BM-bf60jnOC8VG9hibOdc/view?usp=sharing

12.Known Issues

Known issues in the "Book Your Doctor" project include occasional latency with appointment booking during high traffic, likely due to database query times. Additionally, some users have reported inconsistent loading of doctor profiles, possibly caused by API call timeouts in slower network conditions.

13. Future Enhancement

Future enhancements for a "Book Your Doctor" app using MERN stack could include integrating AI-driven health assessments for personalized doctor recommendations and implementing real-time video consultations with doctors. Additionally, incorporating advanced data analytics for patient insights and feedback can further improve user experience and care quality.