|  |
| --- |
| **Book a doctor using MERN**      **Team Members:**  **1.Paturu Suneel Chowdary**  **2.Vaddiboina PattabhiRami Reddy**  **3.Botta Sridhar**  **4.Jaladanki HarshaVardhan Reddy**  Project overview  This project is a full-stack web application built using the MERN stack (MongoDB, Express.js, React.js, and Node.js) to provide a user-friendly and efficient system for managing doctor appointments. Patients can search for doctors by specialty, location, or availability, book appointments, view their appointment history, and manage their profile information.    Features   * User Registration and Login * Doctor Profile Management * Appointment Booking and Management * Appointment History and Notification * Responsive and user-friendly   Project Architecture  Frontend : Botta Sridhar,HarshaVardhan Reddy   * The frontend for the **our** application AppointDoc is developed using **React.js** as its core framework, with **CSS** and **HTML** for styling and structure. * The frontend includes essential functionality for users, particularly patients, to navigate and book appointments with doctors. * **React Router** for navigating between pages (e.g., booking form, doctor list, and appointment history). * **CSS Frameworks** like Bootstrap or Material-UI to ensure responsive design across devices. * **State Management** (e.g., Context API or Redux) to manage data across components, such as user information and selected appointments.       Backend : Vaddiboina PattabhiRami Reddy   * **Node.js** with **Express.js** for creating RESTful APIs to handle requests for booking, canceling, and rescheduling. * Endpoints for managing user roles and appointments (e.g., POST /api/appointments for booking). * RESTful API endpoints for user authentication, doctor management, and appointment booking. * Middleware for authentication and error handling.   DatabSase : Paturu Suneel Chowdary   * MongoDB was used as the database due to its scalability and flexibility in managing unstructured data. * MongoDB is ideal for storing diverse information such as user profiles and appointment details. * MongoDB provides a straightforward way to manage the relationships between **patients**, **doctors**, **available time slots**, and **appointments**. * Each collection helps manage a specific aspect, making it easy to track bookings, availability, and user details.     Setup Instructions    To set up BloodLife locally, follow these steps:   * Clone the repository:   <https://github.com/SUNEELCHOWDARY/nm_book_a_doc_mern.git>   * Install the required dependencies for backend:   ***npm install express***   * Navigate to the client directory:   ***cd client***   * Install the dependencies for the client:   ***npm i react-router-dom react-redux axios antd @reduxjs/toolkit react-bootstrap moment***   * Start the server:   ***npm start***   * In a new terminal window, navigate to the client directory:   ***cd client***   * *Start the client:*   ***npm start***  File Structure        **Running the Application**    Frontend   * In a new terminal window, navigate to the client directory:   ***cd client***   * *Start the client:*   ***npm start***  Backend   * Start the server:   ***npm start***  API Documentation    **User API**   * POST /api/auth/register – Register a new user * POST /api/auth/login – Log in an existing user       **Doctor API**   * GET /api/doctors – Get all doctor profiles * POST /api/doctors – Create a new doctor profile * GET /api/doctors/:id – Get a specific doctor profile by ID     **Appointment API**   * POST /api/appointments – Book an appointment * GET /api/appointments/:userId – Get all appointments for a user         **User Interface**  The **User Interface (UI)** for the **Doctor Booking Application** built with **React** as part of the **MERN Stack**. The UI should be designed in a way that allows users to easily:   * Register and log in to their accounts * Search for and view doctor profiles * Book appointments with available doctors * View their appointment history  1. Header Component   **This is the top navigation bar, which might include links to different pages like Home, Doctors, Appointments, and a Logout button.**        2.Footer Component  The footer typically contains some information about the app, copyright, or contact details.      3.Doctor Card Component (components/DoctorCard.js)  This component displays the information for each doctor in a card format. It will show details like the doctor’s name, specialty, and available time slots.      4. **HomePage (pages/HomePage.js)**  This is the landing page that could display some introductory information about the service and provide links to book appointments or log in.      **TESTING**  Testing is crucial in any software development workflow, and the **MERN stack** has a wide range of tools to support different types of testing. From unit tests for React components to API tests with Supertest, and end-to-end testing with Cypress    1. **Cypress (End-to-End Testing)**  **Cypress** is a popular tool for **end-to-end testing**. It allows you to simulate real user interactions with your app in a real browser environment. It’s especially useful for testing full user flows (e.g., login, booking an appointment).   * **Features**:   + Simulates real user interactions   + Works in a real browser environment   + Provides easy debugging with video recordings and screenshots of tests   + Can test authentication, form submissions, page routing, etc.   Installiation      2.Jest    **Jest** is the most widely used testing framework for JavaScript applications, especially with React. It’s fast, reliable, and comes with built-in test runners, assertions, and mocking capabilities.   * **Features**:   + Snapshot testing   + Mocking of modules and functions   + Coverage reporting   + Asynchronous testing support   Installiation      3.Supertest    **Supertest** is used for testing RESTful APIs. It allows you to make HTTP requests to your Express app and test the responses (status codes, body content, etc.).  Installiation    **SCREENSHOTS OR DEMO**    1.For User Profile   * Homepage      * Appointment Lists      * Booking Appointments      * Apply As Doctor      * New Notifications     **2.For Doctor Profile**   * Homepage      * Appointment Lists      * Manage Profile     **3.For Admin Profile**   * Homepage      * Doctor List      * User List         **FUTURE ENHANCEMENTS**    Future work may include:   * **Telehealth Integration**: Adding telehealth services for remote consultations, allowing patients to connect with doctors through video calls. * **Multi-language Support**: Expanding the application to support multiple languages for wider accessibility. * **AI-based Recommendations**: Implementing AI algorithms to recommend doctors based on patient history and preferences. * **Enhanced Data Security**: Using blockchain technology for secure data management, ensuring patient data is safe and immutable. * **Patient Feedback Analytics**: Incorporating advanced analytics to analyze patient feedback and improve the system based on user preferences and experiences.   The development of this application highlights the importance of user-centered design in healthcare, emphasizing the need for continuous iteration based on feedback to achieve the best outcomes.  **Thank you** |