Project Report: Book a Doctor Using MERN

Introduction

"Book a Doctor Using MERN" is a full-stack web application designed to simplify the process of booking doctor appointments online. The platform provides an efficient, user-friendly interface where users can find doctors, book appointments, and manage their medical consultations. It eliminates the traditional hassles of scheduling appointments, offering real-time availability and seamless integration with healthcare providers.

Objective

To develop a robust, scalable, and user-friendly system that:

1. Allows patients to browse doctors based on specialization, location, and availability.
2. Provides an intuitive booking process.
3. Enables doctors to manage their schedules and appointments.
4. Offers administrative tools for overall governance and platform maintenance.

System Requirements

Hardware Requirements

- Windows 8 or higher machine.
- Bandwidth: 30 Mbps.

Software Requirements

- Two web browsers installed (e.g., Chrome, Firefox).
- Node.js, MongoDB, React.js, Express.js.

Features

User Roles and Responsibilities

Customer/Patient

- **Registration and Login**: Create an account using an email and password.
- **Browse Doctors**: Filter doctors by specialization, location, or availability.
- **Book Appointments**: Choose a doctor, select a date, and upload documents if necessary.
- **Manage Appointments**: View, cancel, or reschedule appointments and receive notifications.

Doctor

- **Registration and Approval**: Register and get approved by an admin.
- **Manage Appointments**: Accept, reschedule, or cancel bookings and update appointment statuses.
- **Maintain Records**: Update patients' medical histories and provide follow-up instructions.

Admin

- **Governance**: Approve doctor registrations, monitor activities, and enforce policies.
- **User Management**: Address disputes and maintain platform compliance.

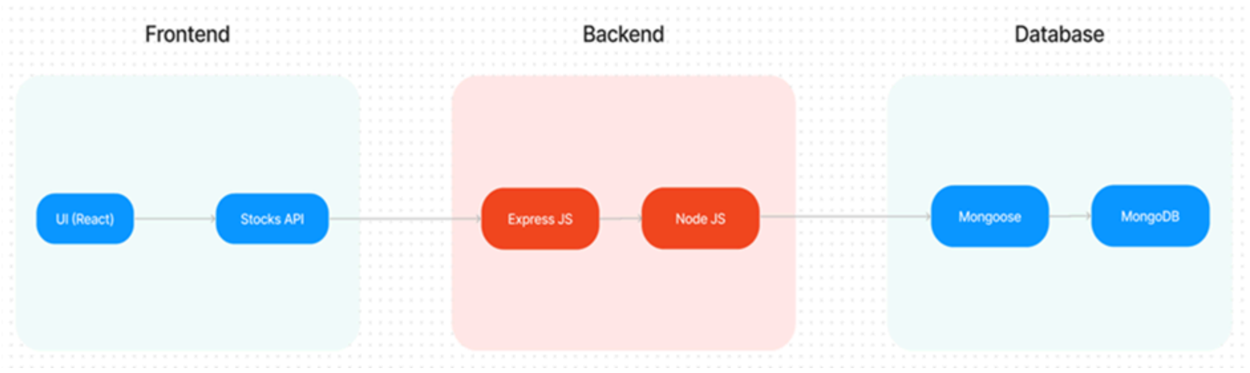
Scenario-Based Case Study

- **Example**: John books an appointment with Dr. Smith for a routine check-up. After signing up, he selects a date, uploads documents, and receives a confirmation. Dr. Smith manages the appointment on his dashboard and updates John's records post-consultation.

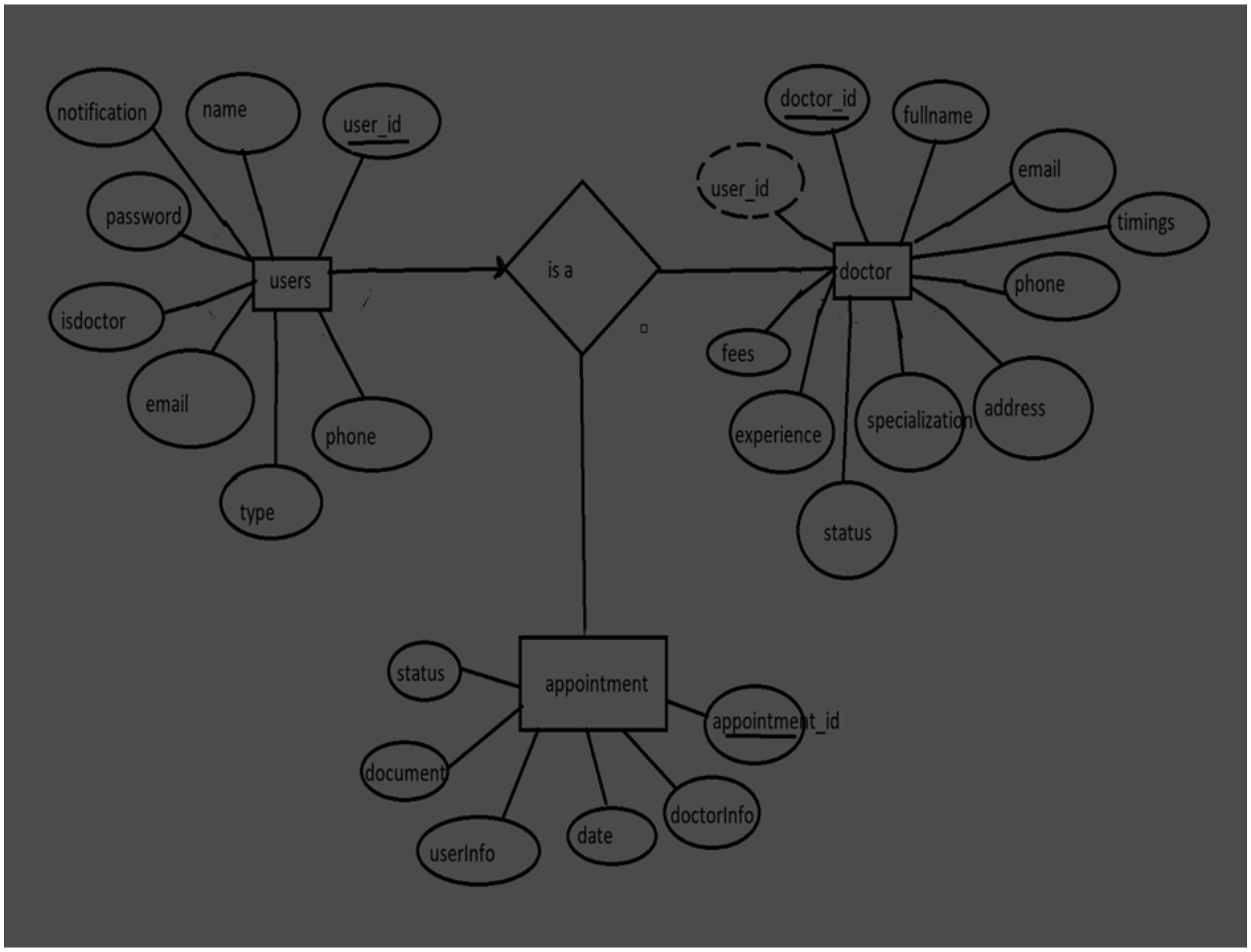
Technical Architecture

The system follows a **client-server model** with the following components:

1. **Frontend**: Built using React.js, Bootstrap, and Material-UI for dynamic and responsive user interfaces. Axios is used for API communication.
2. **Backend**: Utilizes Express.js to handle server-side logic, routing, and RESTful APIs.
3. **Database**: MongoDB stores data for users, doctors, and appointments, ensuring scalability and efficient querying.



Database Design



Collections

1. **Users**

    - Fields: `_id`, `name`, `email`, `password`, `isDoctor`, `type`, `phone`, `notifications`.
2. **Doctors**
    - Fields: `_id`, `userID`, `fullName`, `email`, `timings`, `phone`, `address`, `specialization`, `status`, `experience`, `fees`.
3. **Appointments**
    - Fields: `_id`, `doctorInfo`, `userInfo`, `date`, `document`, `status`.

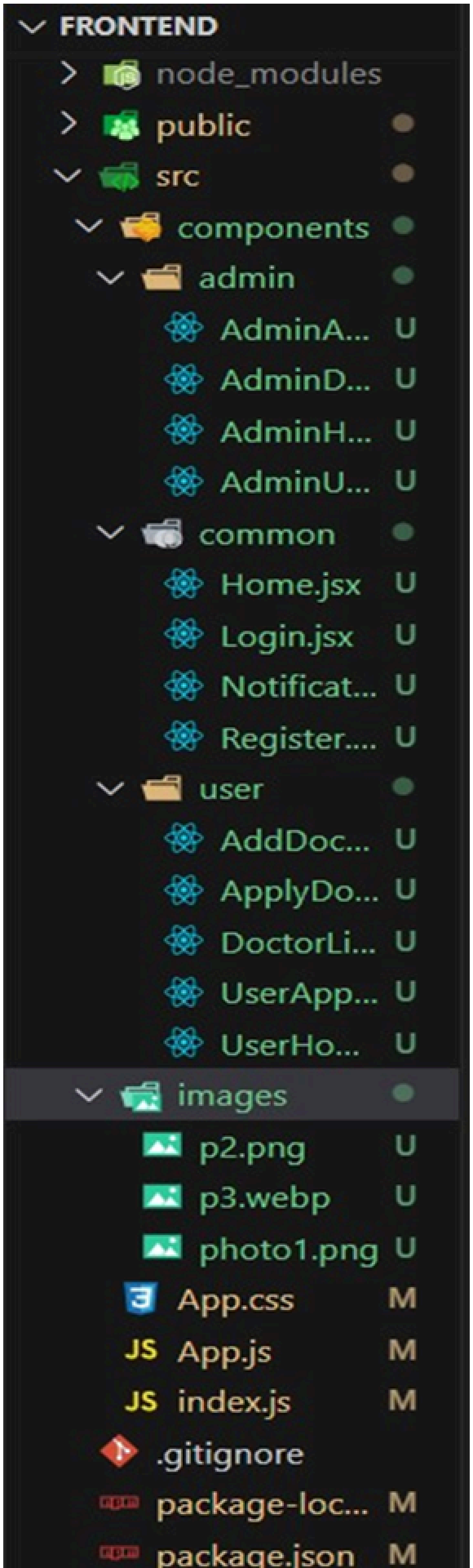
Pre-Requisites

1. **Node.js**: Install and configure to run server-side JavaScript.
2. **Express.js**: Install for handling backend routing and middleware.
3. **MongoDB**: Set up for database storage.
4. **React.js**: Develop dynamic frontend interfaces.
5. **Auxiliary Tools**: Install Moment.js, Material-UI, Ant Design, and Bootstrap for enhanced functionality and styling.

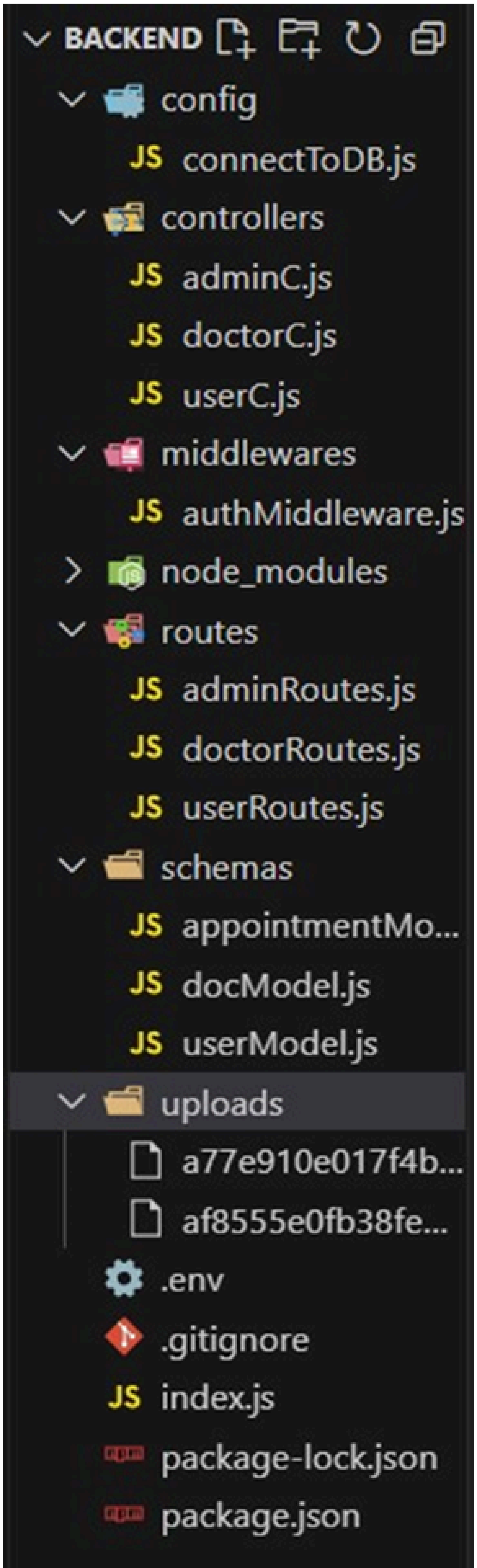
Implementation Steps

Setup

1. **Project Structure**
    - Create separate folders for `frontend` and `backend`.
    - Install required dependencies in each directory using `npm install`.



```
∨ FRONTEND
  > node_modules
  > public                 ●
  ∨ src                    ●
    ∨ components           ●
      ∨ admin              ●
          AdminA...        U
          AdminD...        U
          AdminH...        U
          AdminU...        U
      ∨ common             ●
          Home.jsx         U
          Login.jsx        U
          Notificat...     U
          Register....     U
      ∨ user               ●
          AddDoc...        U
          ApplyDo...       U
          DoctorLi...      U
          UserApp...       U
          UserHo...        U
    ∨ images               ●
        p2.png             U
        p3.webp            U
        photo1.png         U
      App.css              M
      App.js               M
      index.js             M
    .gitignore
    package-loc...         M
    package.json           M
```

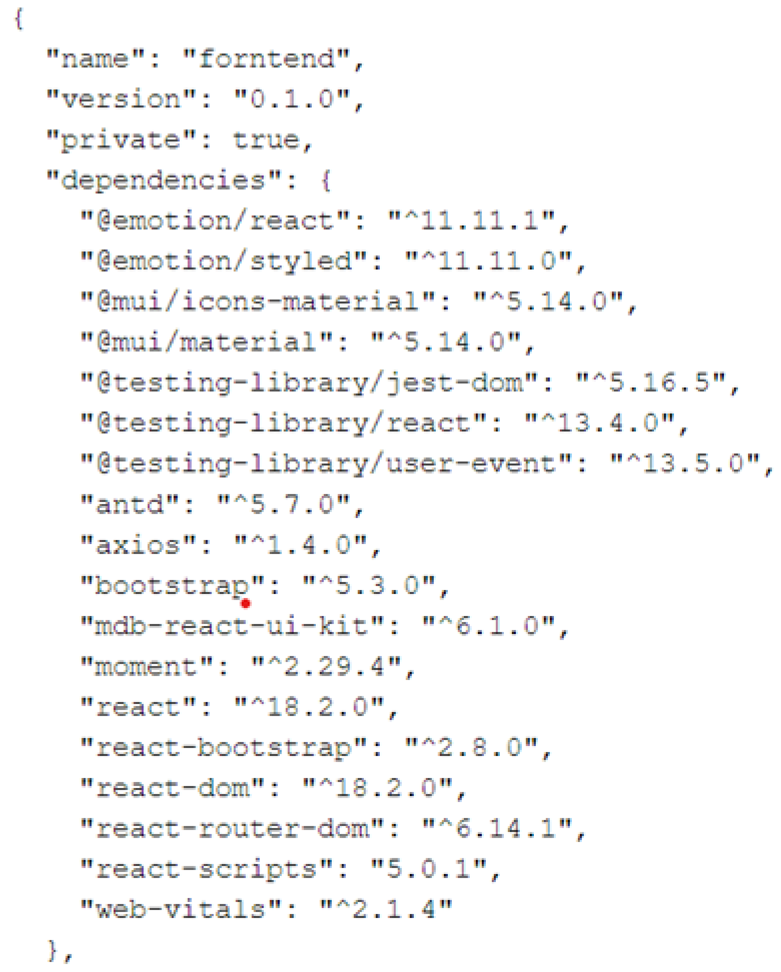


```
∨ BACKEND
  ∨ config
      connectToDB.js
  ∨ controllers
      adminC.js
      doctorC.js
      userC.js
  ∨ middlewares
      authMiddleware.js
  > node_modules
  ∨ routes
      adminRoutes.js
      doctorRoutes.js
      userRoutes.js
  ∨ schemas
      appointmentMo...
      docModel.js
      userModel.js
  ∨ uploads
      a77e910e017f4b...
      af8555e0fb38fe...
    .env
    .gitignore
    index.js
    package-lock.json
    package.json
```

2. **Backend Setup**
    - Configure Express.js and MongoDB.
    - Define API routes for user registration, login, booking, and appointment management.
    - Implement JWT-based authentication.

```json
{
  "name": "backend",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "start": "nodemon index"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "bcryptjs": "^2.4.3",
    "cors": "^2.8.5",
    "dotenv": "^16.3.1",
    "express": "^4.18.2",
    "jsonwebtoken": "^9.0.1",
    "mongoose": "^7.3.2",
    "multer": "^1.4.5-lts.1",
    "nodemon": "^3.0.1"
  }
}
```

3. **Frontend Setup**
    - Build components for user dashboards, doctor management, and admin tools.
    - Use Material-UI and Bootstrap for responsive design.
4. **Deployment**
    - Use `npm start` to run the server.
    - Host the application locally at `http://localhost:3000` for testing.



```
{
  "name": "forntend",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@emotion/react": "^11.11.1",
    "@emotion/styled": "^11.11.0",
    "@mui/icons-material": "^5.14.0",
    "@mui/material": "^5.14.0",
    "@testing-library/jest-dom": "^5.16.5",
    "@testing-library/react": "^13.4.0",
    "@testing-library/user-event": "^13.5.0",
    "antd": "^5.7.0",
    "axios": "^1.4.0",
    "bootstrap": "^5.3.0",
    "mdb-react-ui-kit": "^6.1.0",
    "moment": "^2.29.4",
    "react": "^18.2.0",
    "react-bootstrap": "^2.8.0",
    "react-dom": "^18.2.0",
    "react-router-dom": "^6.14.1",
    "react-scripts": "5.0.1",
    "web-vitals": "^2.1.4"
  },
```

ER Diagram

The ER diagram illustrates the relationships between three main entities:

1. **Users**: Linked to appointments and indirectly connected to doctors.
2. **Doctors**: Manage appointments and are registered by admin approval.
3. **Appointments**: Central entity linking users and doctors.

Key Features

1. **Real-Time Booking**: View available slots and book instantly.
2. **User Notifications**: Receive updates for appointment confirmations, cancellations, and reminders.
3. **Admin Governance**: Monitor and manage all platform activities.
4. **Scalable Design**: Supports high user traffic with efficient data storage.

Challenges and Solutions

1. **Challenge**: Real-time data synchronization.
    - **Solution**: Implemented efficient APIs and WebSocket communication.
2. **Challenge**: Securing user data.
    - **Solution**: Used JWT authentication and encrypted sensitive information.
3. **Challenge**: Ensuring cross-browser compatibility.
    - **Solution**: Extensively tested on multiple browsers and platforms.

Future Enhancements

1. **Teleconsultation**: Add video conferencing features for virtual appointments.
2. **AI Integration**: Include AI-driven doctor recommendations based on user preferences.
3. **Mobile App Development**: Extend functionality to Android and iOS platforms.

Conclusion

The "Book a Doctor Using MERN" project demonstrates the potential of full-stack development in addressing real-world challenges. By leveraging the MERN stack, the platform ensures a seamless, efficient, and secure user experience, revolutionizing traditional appointment booking systems.