**MEDICARE**

DESIGN DOCUMENT

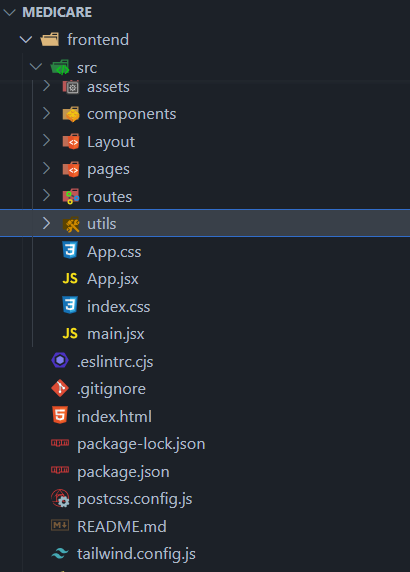
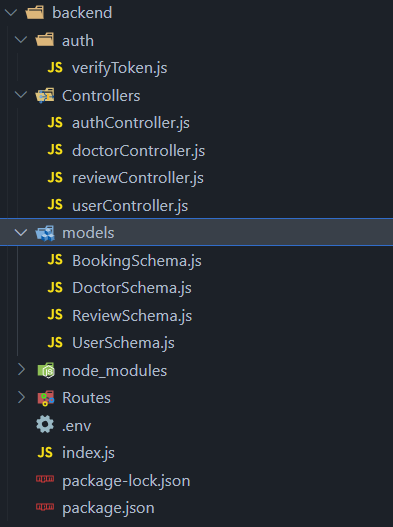
**1. Introduction**

Medicare website is a comprehensive platform built on the MERN stack, allowing users to seamlessly book appointments with specialist doctors. Users authenticate securely to access the system, where they browse through a curated list of doctors, select their preferred specialist, and book appointments at their convenience. Doctors receive appointment requests and have the capability to approve them, ensuring efficient scheduling. The platform facilitates secure payments for appointments, providing users with a hassle-free experience. After appointments, users can leave reviews for doctors, fostering transparency and trust within the community. This project embodies the fusion of modern technology with healthcare, delivering a user-centric solution for managing medical appointments with ease.

**2. Architecture Overview**

Medicare is built upon the MERN (MongoDB, Express.js, React.js, Node.js) stack:

* **Frontend:** Utilizing React.js, users can seamlessly navigate through the platform to book appointments with doctors. The interface is designed to be intuitive, allowing users to search for doctors, view their availability, and schedule appointments conveniently.
* **Backend:** Powered by Express.js and Node.js, the backend facilitates the doctor booking appointment functionality through RESTful APIs. These APIs handle tasks such as user authentication, appointment scheduling, and retrieval of doctor availability.
* **Database:** MongoDB serves as the backbone for storing crucial data related to doctor profiles, appointment schedules, and user information. It ensures efficient data management and retrieval, enabling seamless interactions between users and healthcare providers.

**3. User Roles**

**a. Patient**

- Register/Login: Patients can create accounts and log in to the system.

- Browse Doctors: View a list of registered doctors along with their specialities.

- Book Doctor: Send appointment requests to specific doctors.

- View Appointment Status: Check the status of appointment requests and confirmations.

**b. Doctor**

- Register/Login: Doctors can create accounts and log in to the system.

- Approve/Reject Appointments: Doctors can manage appointment requests from patients.

- View Appointment Schedule: Access a schedule of approved appointments.

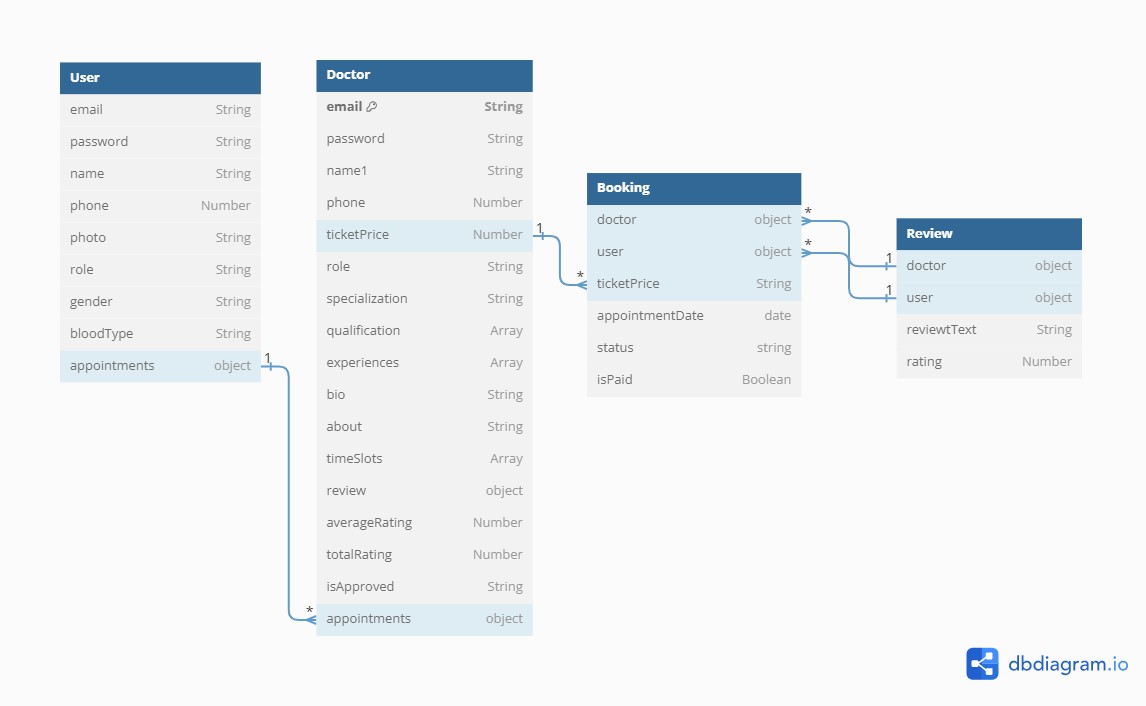
- Update Profile: Maintain personal and professional information.

**4. Database Design**

HealthCare MongoDB database consists of the following collections:

* **Users:** Stores user information, including name, email, password hash, phone number, gender etc.
* **Doctor:** Contains personal-> name, email, phone, role and professional-> specialization, qualification, experience, timeslots details.
* **Booking:** Contains details of user booking the appointment, the doctor details who is booked, ticketprice, date, status, ispaid.
* **Review:** Contains information about user giving the review for a particular doctor can reviewtext and give rating.

**5. Schema Models**



**6. Technologies Used**

* **Front-end:** JavaScript,Reactjs,Tailwind
* **Back-end:** Node.js,Express.js,Mongoose
* **Database:** MongoDB
* **Cloud Services:** Cloudinary.
* **Authentication:** JWT,Bcrypt.js,dotenv
* **Payment Processing:** Stripe

**7. API Design**

Medicare backend offers RESTful APIs for various functionalities:

* **Authentication APIs (/api/auth):**
  + /register (POST): Register a new user.
  + /login (POST): Authenticate user credentials and generate a JWT token.
* **Doctor APIs (/api/doctors):**
  + /:id (GET): Get a Single Doctor.
  + / (GET): Get all Doctors.
  + /:id (PUT): Update Doctor.
  + /:id (DELETE): Delete Doctor.
  + /profile (GET): Get Doctor Profile.
* **User APIs (/api/users):**
  + /:id (GET): Get a Single User.
  + / (GET): Get all Users.
  + /:id (PUT): Update User.
  + /:id (DELETE): Delete User.
  + /profile (GET): Get User Profile.
  + /appointments/my-appointments (GET): Get User appointment.
* **Bookings API (/api/bookings):**
  + /checkout-session/:doctorId (POST): Checkout process for a booking session with a specific doctor.
* **Reviews API (/api/reviews):**
  + /getAllReviews (GET): Get reviews from users.
* **External API:**
  + https://jsapi.apiary.io/apis/practoplatform/reference/doctor-details/list-all-doctors.html

**8. Frontend Design**

Medicare frontend, developed using React.js and styled with CSS, encompasses various components:

1. User Interface(UI):

* Create a clean and intuitive user interface that allow users to navigate easily.
* Use appropriate color schemes, typography, and visual elements to enhance readability and aesthetics.
* Design responsive layouts to ensure compatibility across various devices and screen sizes.

1. Booking Flow:

* Design a seamless booking flow with clear steps for users to search for doctors, select appointment times, and complete the booking process.
* Provide filters and search functionalities to help users find specific doctors or specialties efficiently.
* Incorporate interactive elements for selecting appointment dates and times.

1. Authentication and Authorization:

* Implement a secure authentication system to verify user identities and manage access to sensitive information.
* Design user-friendly login and registration forms with validation to ensure data integrity and security.
* Provide appropriate error messages and feedback to guide users through the authentication process.

1. Appointment Management:

* Display users' booked appointments in a clear and organized manner, allowing them to view upcoming appointments and manage their schedules.
* Include features for users to modify or cancel appointments if necessary, with confirmation prompts to prevent accidental actions.
* Design interfaces for doctors to review and manage appointment requests, with options to approve or reschedule appointments.

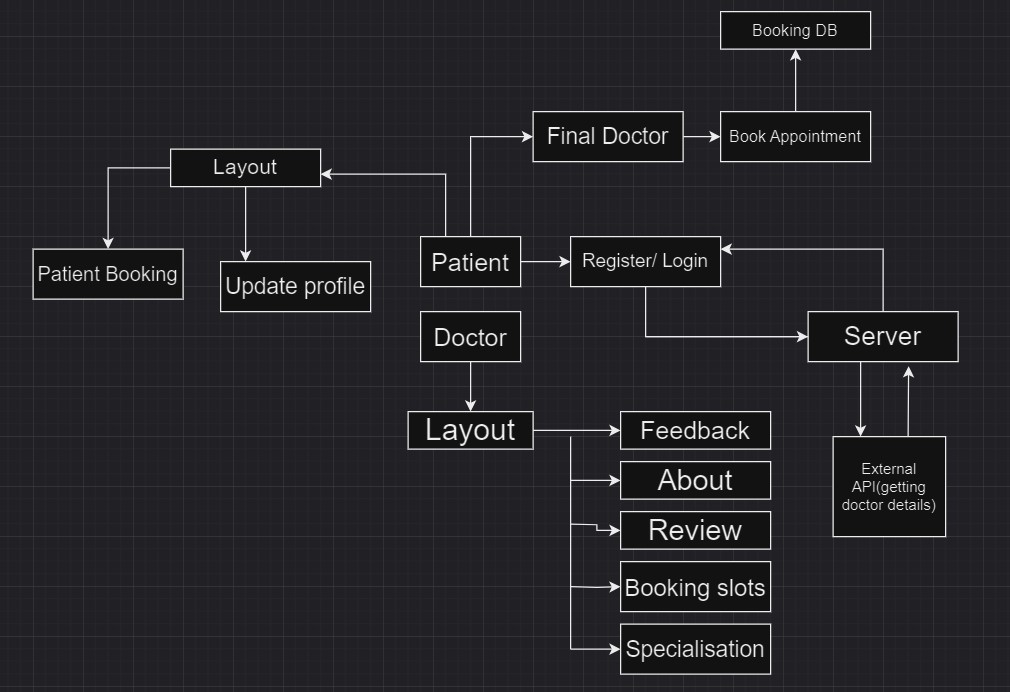
1. Payment Integration:

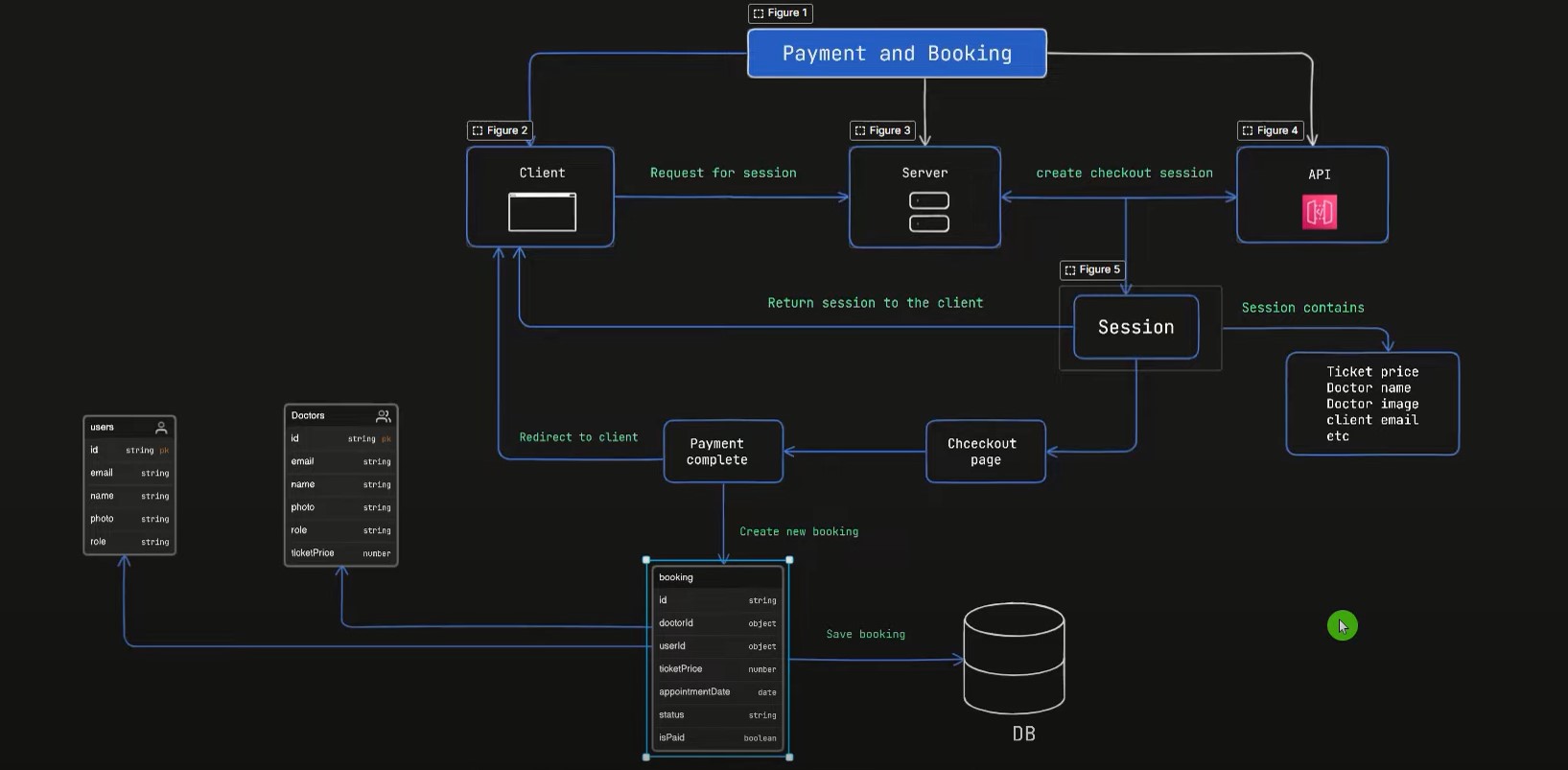
* Integrate a secure payment gateway to facilitate online payments for appointment bookings.
* Design checkout flows that are intuitive and straightforward, guiding users through the payment process with clear instructions and feedback.
* Provide options for users to securely save payment information for future use, if desired.

1. Review and Feedback:

* Allow users to leave reviews and ratings for doctors after their appointments, promoting transparency and accountability.
* Design interfaces for users to view and filter doctor reviews, helping them make informed decisions when booking appointments.
* Implement moderation features to manage and address user feedback effectively.

**9. Data Flow Diagram**





**10. Deployment**

Medicare can be deployed on cloud platforms like Heroku, AWS, or DigitalOcean. Frontend hosting services such as Netlify or Vercel, coupled with MongoDB Atlas for database hosting, offer a robust deployment solution.

**11. Testing**

Unit tests for backend APIs using Postman ensure robustness.

**12. Security**

Medicare prioritizes security by employing robust measures such as encrypting sensitive data and utilizing secure authentication mechanisms like JSON Web Tokens(JWT). This ensures that user information remains protected and only authorized individuals can access the platform, fostering trust and confidence among doctors and patients.

**13. Conclusion**

Medicare offers a seamless user experience, allowing easy booking and management for both doctors and patients. With authentication and secure communication, it enhances accessibility and fosters trust, showcasing the transformative power of technology in healthcare.