1. Auth page

2. Booking page

3. Service page

4. Admin page (to approve and reject booking)

5. Confirmation page to checkout and confirm appointment (user)

6. User page

1. **Project Setup:**
   * Initialize projects by installing modules and libraries needed in terminal.
   * Set up your Node.js and Express.js environment on Replit.
   * Connect to your PostgreSQL database using Neon Console.
2. **Design Database:**
   * Design the schema for your PostgreSQL database. You will need tables for Users, Services, and Bookings.
   * Create the tables in your database using Neon Console.
3. **Implement User Authentication (Firebase):**
   * Integrate Firebase into your front-end project and set up the authentication system.
   * Implement the signup and login pages using Firebase's authentication services.
4. **Setup Redux:**
   * Install Redux and Redux Toolkit in your project.
   * Setup your Redux store and create slices for user data, services data, and bookings data.
5. **Build Out API (Replit & Express.js):**
   * Implement the endpoints you need for your application as described in the previous section.
   * Secure your endpoints with Firebase authentication where necessary.
6. **Implement Services Page:**
   * Retrieve services from your backend using fetch or axios and display them on the services page.
   * Use Redux to manage state for the services data.
7. **Implement Booking Page:**
   * Create a form to collect information about the booking.
   * When the form is submitted, send a request to your backend to create a booking.
8. **Implement Confirmation Page:**
   * Once a booking is successful, redirect the user to the confirmation page.
   * Display a summary of the booking information on the confirmation page.
9. **Implement User Account:**
   * Display user profile information retrieved from Firebase.
   * Retrieve and display the user's bookings from your backend.
   * Implement functionality to allow the user to reschedule or cancel bookings.
10. **Implement Admin Page:**

* Create a special view or set of views for admin users.
* Implement functionality to allow admins to approve or deny bookings.
* Add features for admins to manage services (add, edit, delete).

1. **Test & Debug:**

* Test your application thoroughly, making sure all components work as expected and all edge cases are handled.
* Debug any issues that arise and make necessary adjustments.

1. **Deploy and Monitor:**

* Deploy your application and monitor its performance.
* Fix any issues that arise post-deployment.

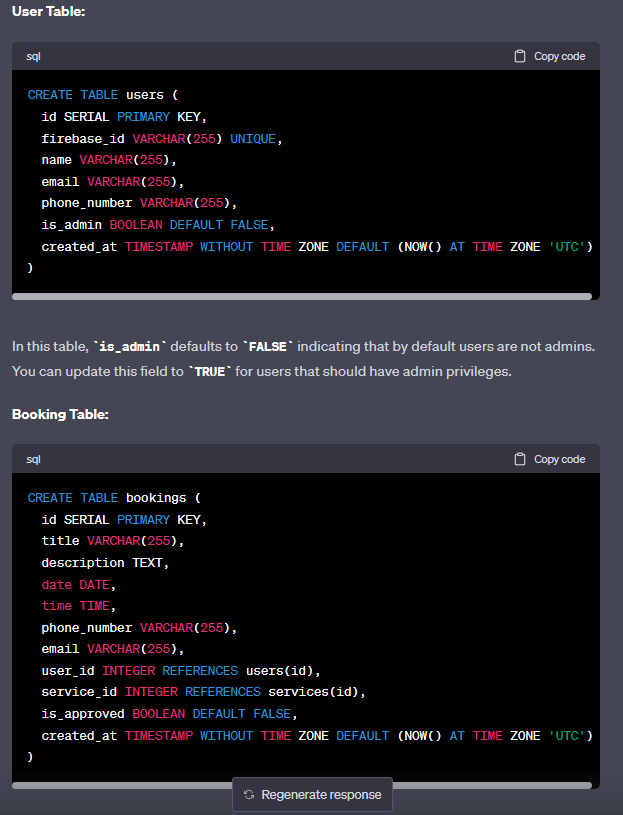
**Front-End (React):**

1. **Signup/Login Page**
   * Route: **/signup** for Sign Up and **/login** for Login
2. **Services Page**
   * Route: **/services**
3. **Booking Page**
   * Route: **/booking**
4. **Confirmation Page**
   * Route: **/confirmation**
5. **User Account**
   * Route: **/account** (Includes account details and booking management. You may want to use sub-routes such as **/account/details**, **/account/bookings**, **/account/bookings/:id** for individual booking management)

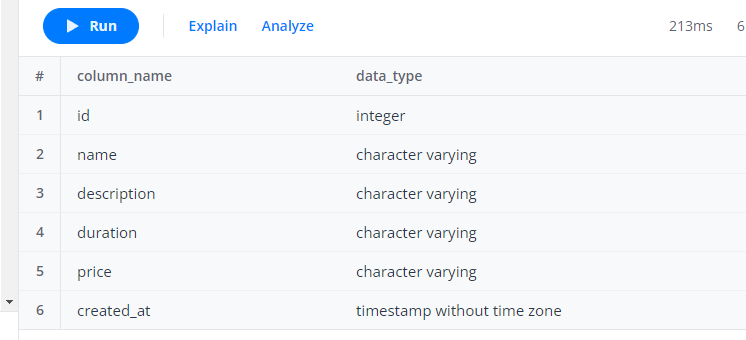
**Back-End (Node.js):**

1. **User Management**
   * Registration: POST **/api/users/signup**
   * Login: POST **/api/users/login**
   * Retrieve Account Details: GET **/api/users/me**
   * Update Account Details: PUT **/api/users/me**
2. **Services Management**
   * Retrieve Services: GET **/api/services**
3. **Booking Management**
   * Create Booking: POST **/api/bookings**
   * Retrieve Bookings: GET **/api/bookings** for all bookings, GET **/api/bookings/upcoming** for upcoming bookings, and GET **/api/bookings/:id** for a specific booking.
   * Update Booking: PUT **/api/bookings/:id** to reschedule or cancel a booking.
4. **Confirmation**
   * Confirmation: POST **/api/bookings/:id/confirm** to send a booking confirmation (This is a backend task triggered after a booking is created or updated, and may not have a corresponding front-end route)

Remember, the exact implementation of your back-end routes will depend on your exact requirements and the specific design of your system.



**SERVICES:**

****