**Project Document 5**

**Task 1: User Authentication & Role-Based Access**

To implement secure authentication with roles like **Admin**, **Analyst**, and **Viewer** using **JWT (JSON Web Token)**, follow these steps:

**1. Set Up Dependencies**

You'll need to install libraries for JWT handling, user management, and password hashing. In a Node.js environment, you can use libraries like jsonwebtoken and bcryptjs.

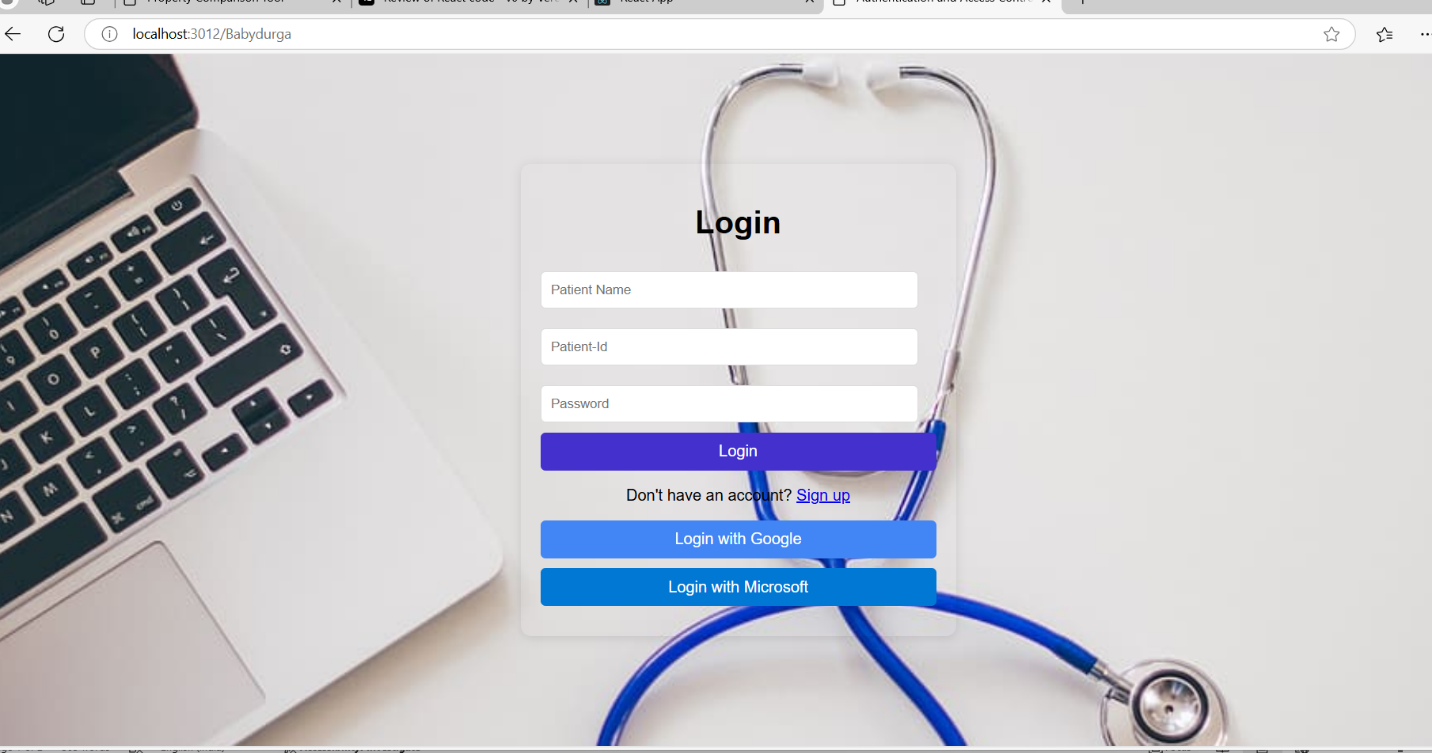
**2. Create User Model**

Define a user schema that includes the role field (Admin, Analyst, Viewer) and a password field. The password will be hashed for security.

**3. Create Authentication Route (Login and JWT Generation)**

Create an authentication route that validates the user's credentials and generates a JWT containing the user’s role.

**Page1: Output**



**Registration Form**:

* **Fields**:
  + **Patient Name**: A text input field for the patient's full name.
  + **Patient ID**: A unique identifier for the patient within the healthcare system.
  + **Password**: A secure password field for creating a login credential.

PORT: Server is running at port http://localhost:3012

MONGO\_URI: MongoDB connection string.

SECRET\_KEY: Secret key used for signing JWT tokens.

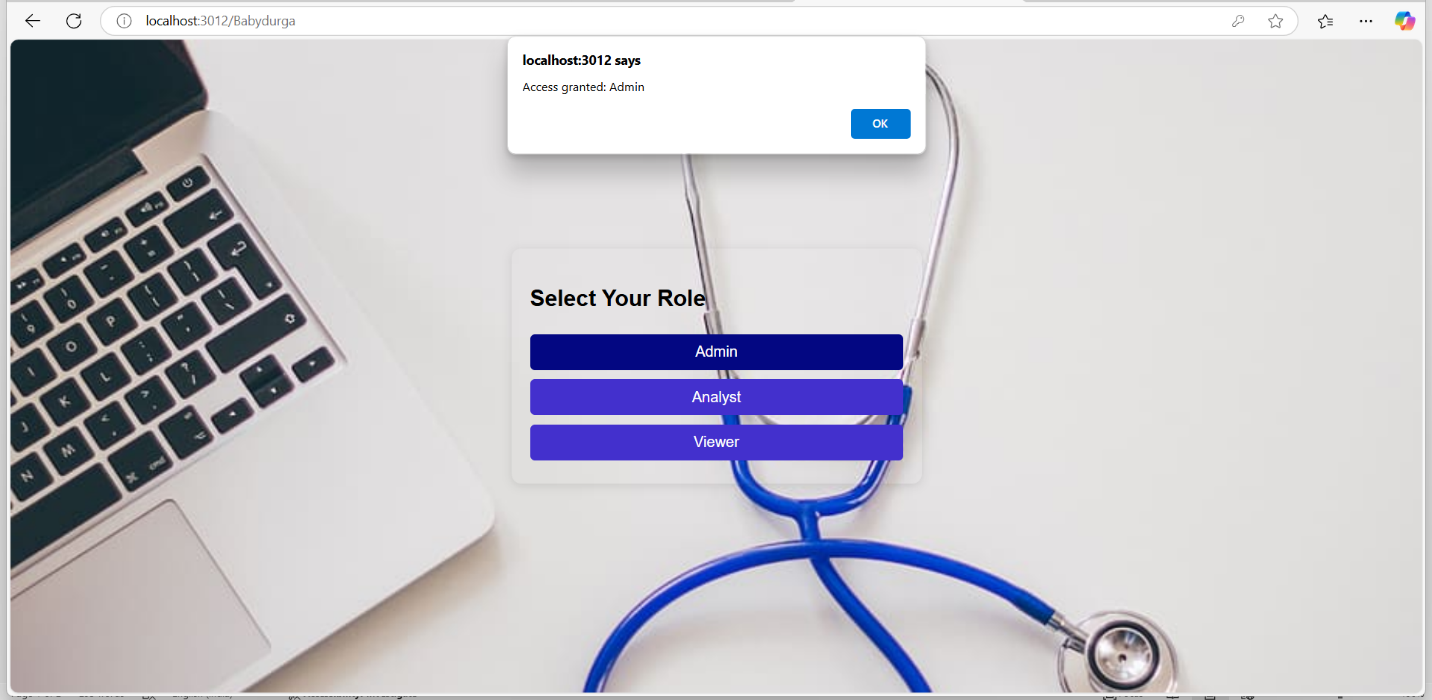
**Added Extra New Feature:**

1. We are added delete access for the Admin. Its Applicable for Admin only not a user.

**Test the Application**

1. **Start the MongoDB server** (if using local MongoDB). Run MongoDB if it's not already running.
2. **Run the app**
3. **Once Admin login this page Access granted**.
4. **User login this page view**.

**Page2:**



Final Notes:

* Security: Remember to keep your SECRET\_KEY in a secure location (e.g., .env file) and avoid exposing it in your code.
* JWT Expiration: You can adjust the expiration of the JWT token by modifying the expires In field in the jwt.sign method.
* Database: This setup assumes you're using MongoDB. If you're using another database, you’ll need to adjust the connection and schema definitions according