**Report on Interactive Dashboard Development Using Next.js, Recharts, and Supabase  
  
Submitted By : Budida Pradeep Reddy  
Submitted on : 13/10/2024**

**1. Introduction**

The goal of this project was to build a fully functional interactive dashboard using **Next.js** for the frontend, **Supabase** as the backend, and **Recharts** for visualizing data. The application allows users to sign up, log in, and view various metrics in real-time through charts.

**2. Approach**

**2.1 Project Setup and Authentication**

* The project was initialized using **Next.js** to take advantage of its SSR (server-side rendering) and built-in routing features.
* **Supabase Auth** was integrated to handle user authentication, with both sign-up and login pages implemented.
  + **API Endpoints**:
    - **Sign-up**: A POST request to <https://zchekhshgfxjjyehfnha.supabase.co/auth/v1/signup> allows users to sign up. User data is stored in Supabase's authentication system.
    - **Login**: A POST request to <https://zchekhshgfxjjyehfnha.supabase.co/auth/v1/token> handles login requests and returns a JWT (JSON Web Token).
* The application protects sensitive pages, such as the dashboard, by checking if a user is authenticated before allowing access. Unauthenticated users are redirected to the login page.

**2.2 Dashboard Implementation**

* The dashboard includes dynamic charts built using **Recharts**. These charts display real-time data stored in the Supabase database, visualizing it in a user-friendly way.
  + **Bar Chart**: Displays data trends over time.
  + **Line Chart**: Shows progress or change in specific metrics.
  + **Pie Chart**: Used for categorical data breakdown.
* **Supabase's real-time feature** was used to keep the data up-to-date. Any changes in the database were instantly reflected on the dashboard without needing a page refresh.
* A form was created to allow users to submit new data directly from the dashboard. This data is instantly processed and displayed.

**2.3 API and Data Handling**

* **Supabase Database** was used to store the application's data (e.g., metrics, chart data, user inputs). API routes in Next.js handle the interaction between the frontend and backend.
* **API Routes**:
  + **POST /api/data**: Adds new data to Supabase.
  + **GET /api/data**: Fetches the latest data from the database to populate the charts.

**2.4 Responsive Design**

* The webiste was made responsive using css so that the dashboard looks good on both desktop and mobile devices. Special attention was given to making charts responsive and ensuring the sidebar collapses on smaller screens for better usability.

**3. Challenges Faced**

**3.1 Authentication Integration**

* Handling Supabase’s session persistence across the dashboard was tricky at first. Ensuring that the JWT was properly stored and used for authenticated API requests took some debugging.

**3.2 API Error Handling**

* There were some issues with **API request errors**, especially with Supabase’s authentication restrictions (e.g., certain email domains not being authorized). Proper error handling had to be added to give users better feedback during signup and data input.

**4. Potential Improvements**

**4.1 Role-Based Access**

* Right now, any authenticated user can access the dashboard. Implementing **role-based access control (RBAC)** would allow for different user levels, such as admin and regular user, each with their own permissions.

4.2 **Server-Side Rendering**: Improving performance and SEO by implementing server-side rendering for initial data loading.

**5. Conclusion**

The dashboard application built with **Next.js**, **Recharts**, and **Supabase** was successfully deployed and met the initial project requirements. This documentation provides brief insights to the approach , potential improvements, challenges faced while implementing the project.