Implementation Report: React Native Expo App with Google SSO and NativeWind

Shopnil Shrestha

June 30, 2024

1 Project Setup

1. **Node.js and Expo CLI Installation:**
   - Node.js was installed following the official installation guide.
   - Expo CLI was installed globally using npm (`npm install -g expo-cli`).
2. **Creating a New React Native Project:**
   - Initialized a new React Native project with Expo using `expo init my-app`.
   - Selected a blank TypeScript template for project setup.
3. **Project Structure Setup:**
   - Established a modular structure to facilitate scalability and maintenance.
   - Created folders like `components` and `hooks` for reusable components and hooks.
   - Inside the `app` folder, created the `_layout.tsx` for main app layout.
   - Inside the `app` folder, created the `tabs` folder for organizing screens.
4. **Nativewind Setup:**
   - Installed Nativewind following the official installation guide.
   - Installed Tailwindcss and configured necessary config files.
   - Configured a nativewind-end.d.ts file for TypeScript types configuration.

2 Implementing Google SSO with Clerk

1. **Integrating Google SSO:**
   - Used Clerk's authentication APIs to initiate Google Sign-In within the app.
   - Configured Google OAuth credentials and used .env file to comply with security practices.
   - Implemented a login screen with Clerk's SDK and initiated the Google Sign-In flow.
2. **Handling Authentication States:**
   - Managed loading, success, and error states during authentication.
   - Handled token management and persistence securely within the app.
3. **Cross-Platform Compatibility:**
   - Ensured compatibility and consistent behavior across both Android and iOS platforms.
   - Tested thoroughly on simulators and physical devices to validate functionality.

3 Displaying User Profile

1. **Retrieving Profile Information:**
   - On successful authentication, fetched user profile data from Google's API using access tokens.
   - Managed API calls and responses asynchronously using `fetch` and `useEffect` hooks.
2. **User Profile Screen:**
   - Designed a user-friendly profile screen displaying the user's name and email.
   - Styled components using NativeWind for consistent UI across different screen sizes.

4 Extra Functionality

1. **Recording Income and Expenses:**
   - Implemented forms or input fields for users to log their income and expenses.
   - Validated and stored transaction data securely using AsyncStorage.
2. **Viewing Transaction History:**
   - Designed a history screen to display recorded transactions in a list format.
   - Enabled users to filter transactions by date, category or type (income/expense).
3. **UI/UX Considerations:**
   - Ensured the transaction recording UI is intuitive and easy to use.
   - Implemented navigation between transaction entry and history screens for seamless user experience.