**Authentication in Angular 18**

This documentation explains how to implement authentication in an Angular 18 standalone application using three providers:

* Google Login (via angular-oauth2-oidc )
* Microsoft Login (via MSAL Angular)
* Facebook Login (via Facebook’s JavaScript SDK)

**1. Prerequisites**

Angular 18 App:

* Ensure you have Angular CLI installed and create a new Angular application.

Node.js and npm:

* Required for package installation and development.

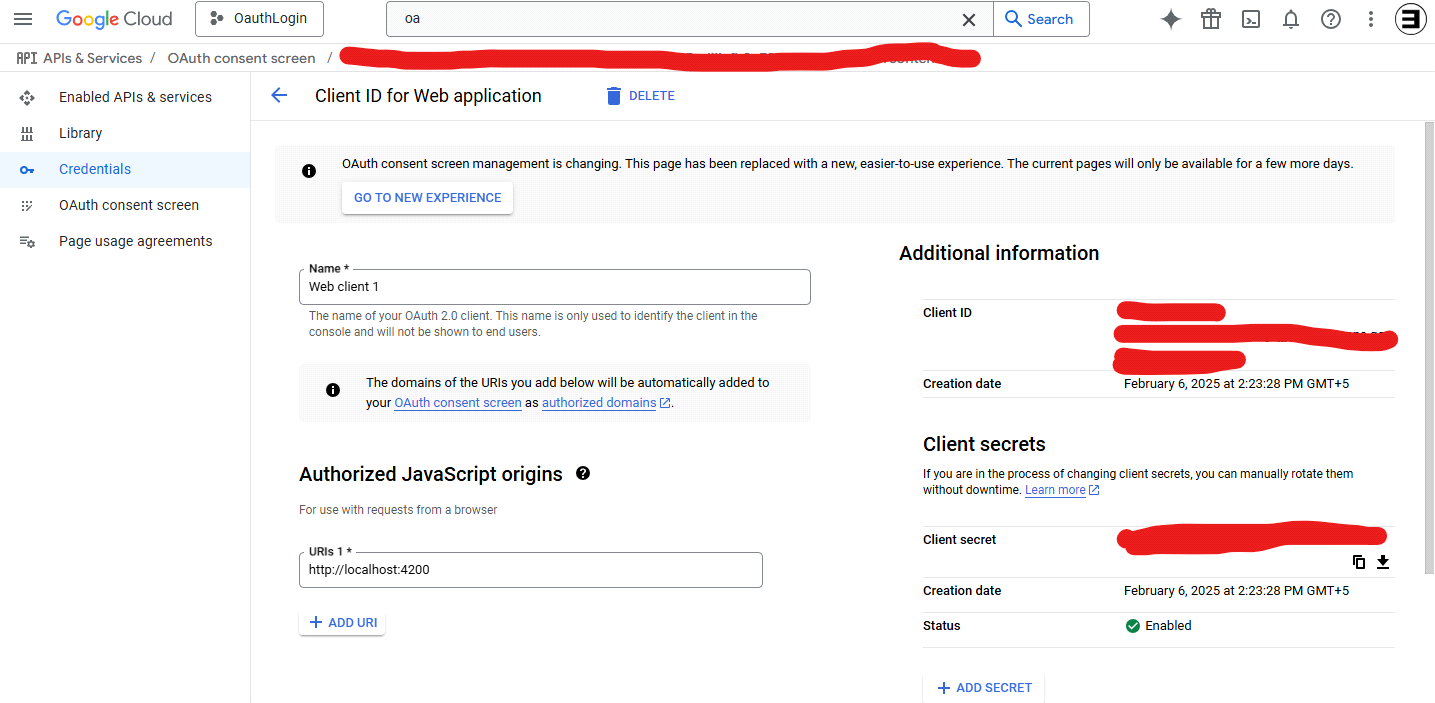
Installed Packages:

* angular-oauth2-oidc – for Google login.
* @azure/msal-angular and @azure/msal-browser – for Microsoft login.
* Facebook’s JavaScript SDK – loaded via a script tag in index.html.

**2. Obtaining Client IDs**

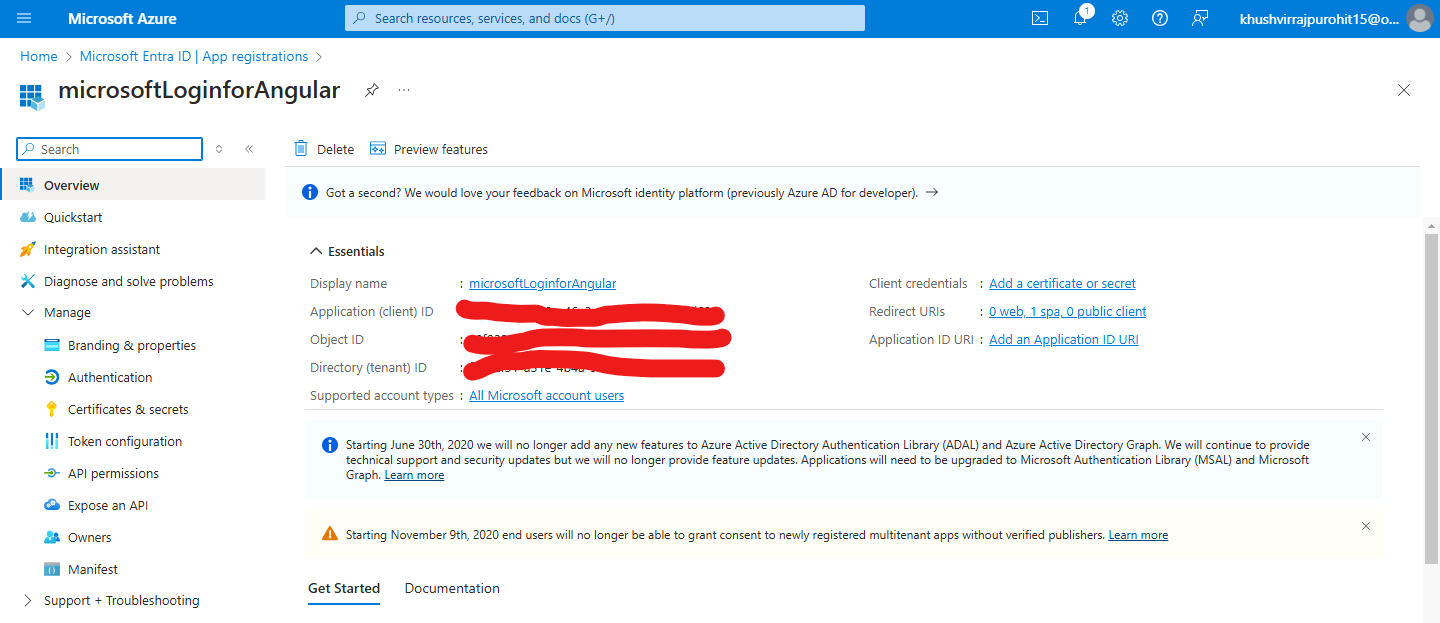
**Google Client ID**

* Visit Google Cloud Console.
* Create a new project or select an existing one.
* Navigate to APIs & Services > Credentials.
* Click Create Credentials and choose OAuth client ID.
* Select Web application as the application type.
* Under Authorized redirect URIs, add your redirect URI (e.g., <http://localhost:4200/dashboard>).
* Copy the generated Client ID and use it in your configuration.



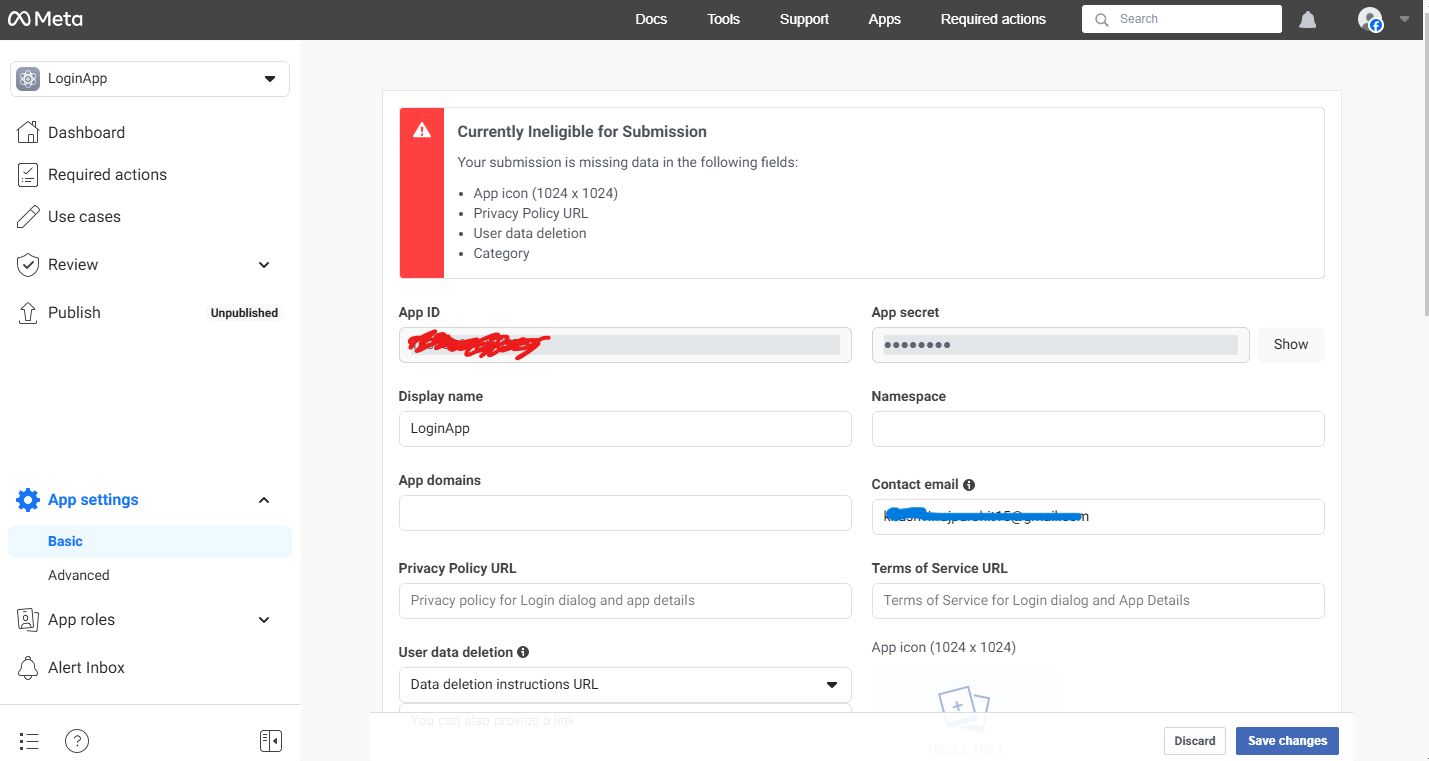
**Microsoft Client ID**

* Log in to the Azure Portal.
* Navigate to Azure Active Directory > App registrations.
* Click New registration.
* Enter a name, and select Accounts in any organizational directory (multitenant) if you want multi-tenant support.
* Under Redirect URI, select Single-page application (SPA) and enter your URI (e.g., <http://localhost:4200/dashboard>).
* Complete registration and copy the Application (client) ID.



**Facebook App ID**

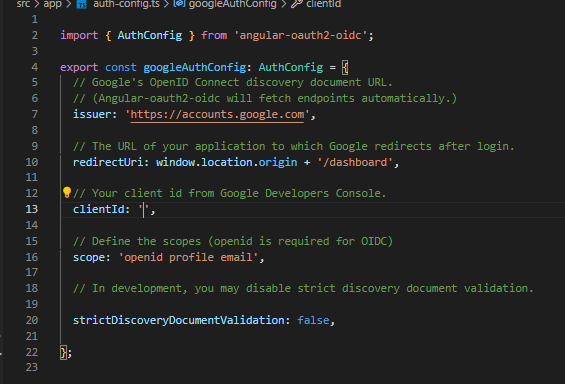
* Visit Facebook for Developers.
* Click My Apps and then Create App.
* Choose an app type (e.g., Consumer) and complete the setup.
* In your app’s Settings > Basic, locate the App ID.
* Add your site URL (e.g., <http://localhost:4200/dashboard>) under Add Platform > Website.
* Use the App ID in your configuration.



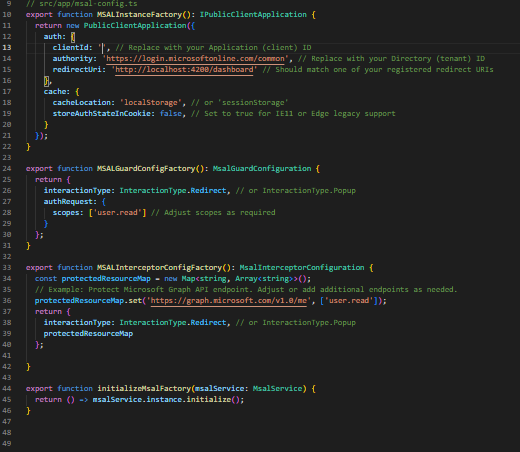
**3. Configuration Files**

**Note: Replace the client Id with your own client Id**

* Google OAuth Configuration



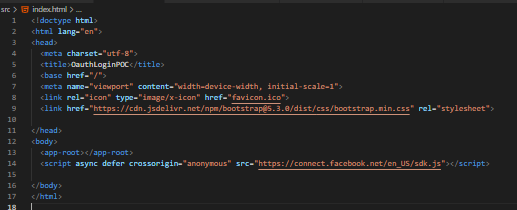
* MSAL Configuration for Microsoft Login



* Facebook sdkConfiguration

Add the below script in your index.html file

<script async defer crossorigin="anonymous" src="<https://connect.facebook.net/en_US/sdk.js>"></script>



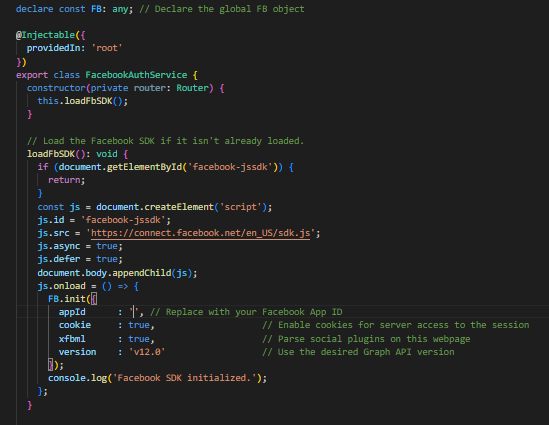
**4. Service Implementations**

* **FacebookAuthService**

This service uses Facebook’s JavaScript SDK to log in via a popup.

File: src/app/facebook-auth.service.ts

**Note: replace appId with your own appId**

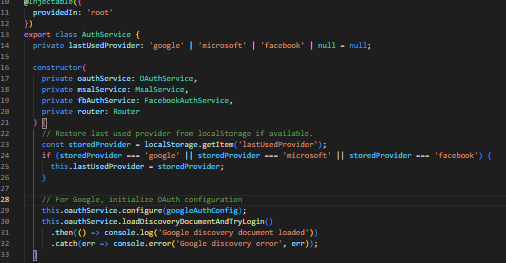


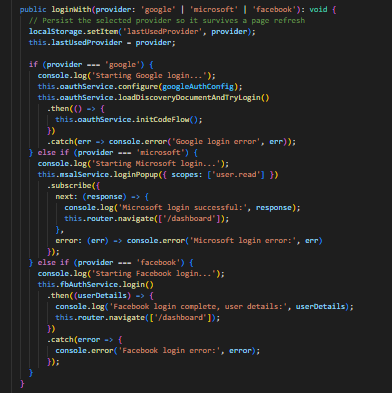


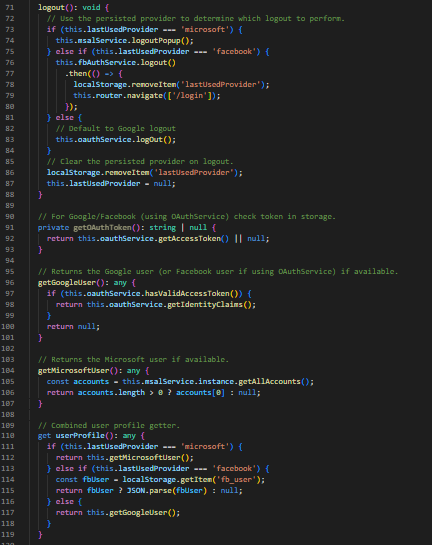
* **Unified AuthService**

This service integrates Google, Microsoft, and Facebook login flows and persists the last used provider.

File: src/app/auth.service.ts









**--Storing Last Used Provider**

* --The last used provider (google, microsoft, or facebook) is stored in localStorage to persist across page refreshes.

**-- Google Authentication**

* --Uses angular-oauth2-oidc to configure OAuth2/OIDC authentication.
* --Calls loadDiscoveryDocumentAndTryLogin() to check if a user is already logged in.
* --Triggers initCodeFlow() to start authentication when logging in.
* --Logs out using oauthService.logOut().

**--Microsoft Authentication**

* --Uses @azure/msal-angular for Microsoft authentication.
* --Calls loginPopup({ scopes: ['user.read'] }) to log in.
* --Retrieves the authenticated user from MSAL (getAllAccounts()).
* --Logs out using logoutPopup().

**--Facebook Authentication**

* --Uses a custom FacebookAuthService for Facebook login.
* --Calls login() to initiate authentication and stores the user in localStorage.
* --Logs out by clearing Facebook session data.
* **NOTE:**
* **Facebook login will only work if the app is running on HTTPS**
* **Only administrator and test users in your project can login via HTTP**

**--Token & User Profile Management**

* --getOAuthToken(): Retrieves Google/Facebook OAuth token.
* --getGoogleUser(): Retrieves Google user details.
* --getMicrosoftUser(): Retrieves Microsoft user details.
* --userProfile: Returns the authenticated user based on the last used provider.

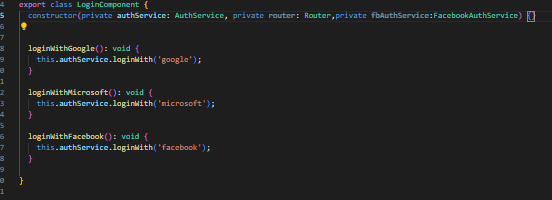
**--Login Status Check**

* --isLoggedIn: Determines login status based on the last used provider:
* --Checks token for Google & Facebook.
* --Uses msalService.instance.getAllAccounts() for Microsoft.

**5. Components**

**Login Component**

File: src/app/login.component.ts

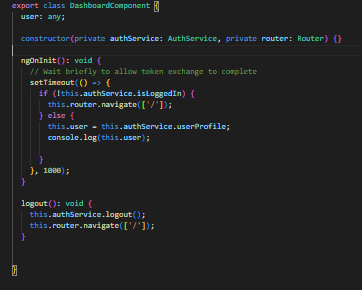


File: src/app/login.component.html

**---**Create 3 buttons calling each of the login functions in your html file

**Dashboard Component**

File: src/app/dashboard.component.ts

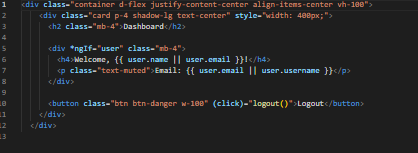


--If not logged in, redirects the user to the login page (/login).

--If logged in, retrieves and stores the authenticated user's profile in this.user.

--Logout() Redirects the user to /login after logout.

File: src/app/dashboard.component.html



**6. Routing and Bootstrapping**

Routing Configuration

---File: src/app/app.routes.ts

**export const routes: Routes = [**

**{ path: '', component: LoginComponent },**

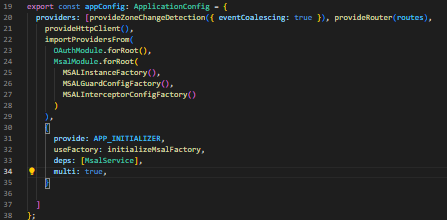
**{ path: 'dashboard', component: DashboardComponent },**

**{ path: '\*\*', redirectTo: '' }**

**];**

----File: src/app.config.ts

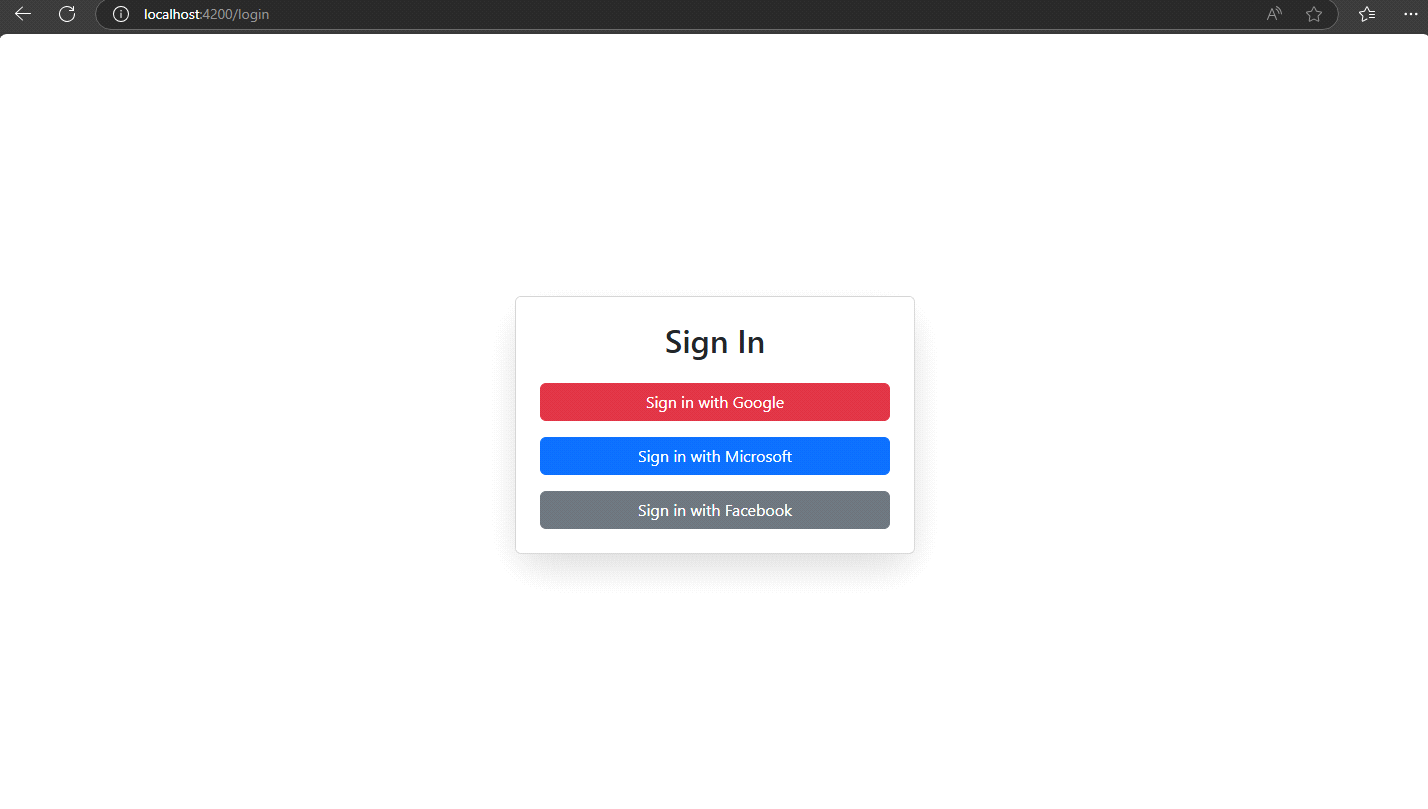
Update your app.config file

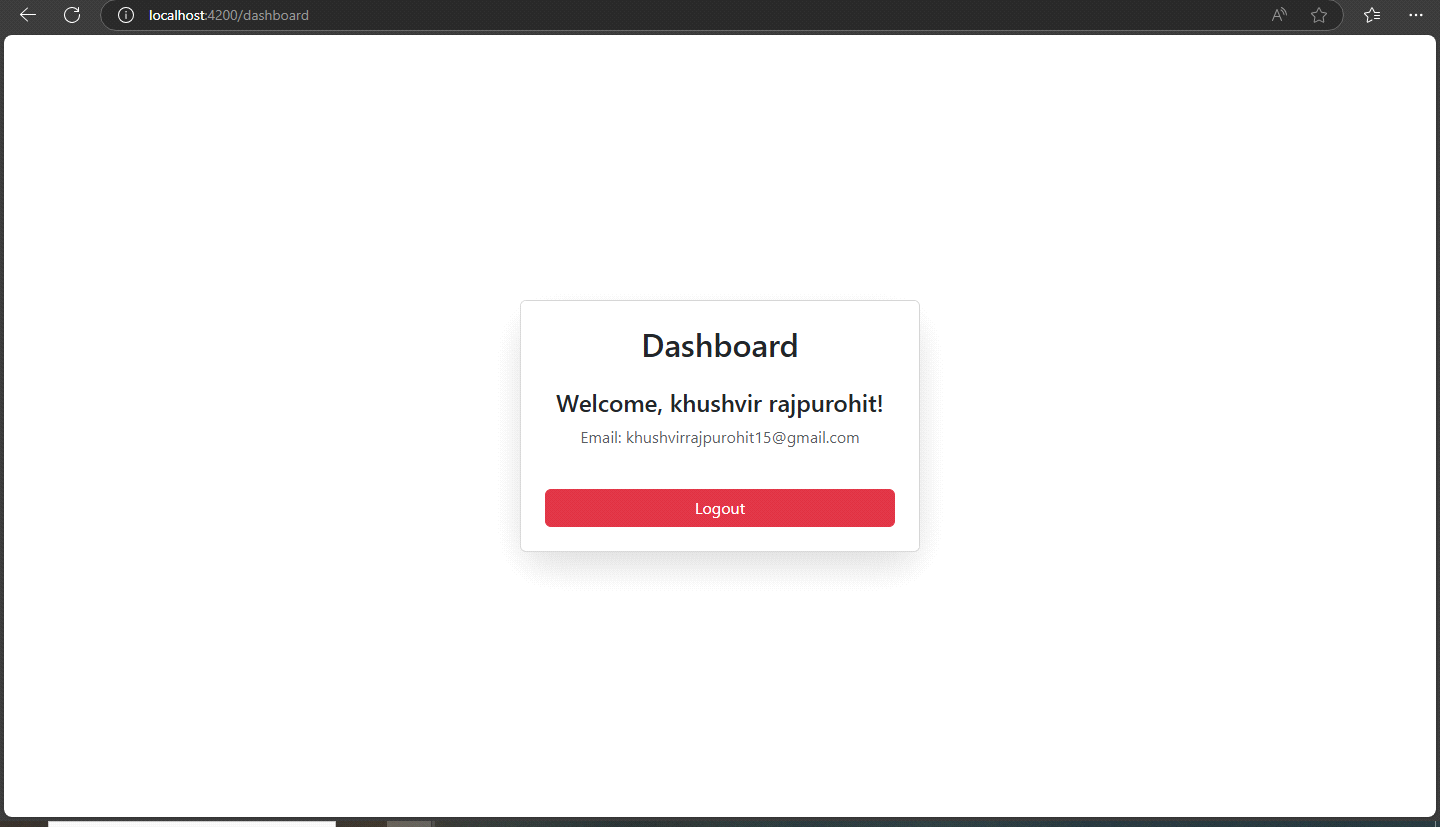
****

**Run Your App:**

Use ng serve (or your preferred command) to run your app locally.

**UI:**

****

****