**Authenticate Google Sign-In for webpages**

Why Use Google Firebase?

Implementing Google Sign-In using Firebase aims to secure authentication to create applications building easy while improving the sign in and onboard experience for end users. Also, it provides other benefits for an end-to-end identity solution, supporting email and password accounts, phone auth, and Google, Twitter, Facebook, and GitHub login, and more.

**Authenticate Using Google Sign-In with JavaScript**

By integrating Google Sign-In into your app, you can allow your users to login with Firebase using their Google Accounts. You can integrate Google Sign-In by utilizing the Firebase SDK to perform the sign-in flow or manually performing the Google Sign-In flow and giving the generated ID token to Firebase.

Click the link below and follow the steps in order to launch successfully. <https://firebase.google.com/docs/auth/web/google-signin#before_you_begin>

1. Add Firebase to your JavaScript project.
2. Enable Google Sign-In in the Firebase console:

a. In the Firebase console, open the Auth section. b. On the Sign in method tab, enable the Google sign-in method and click Save.

The one way to successfully handle the sign in flow is using the firebase SDK. Click to read more <https://firebase.google.com/docs/auth/web/google-signin#handle_the_sign-in_flow_with_the_firebase_sdk>

1. To handle the sign-in flow with the Firebase JavaScript SDK, follow these steps:

Create an instance of the Google provider object: import { GoogleAuthProvider } from "firebase/auth";

const provider = new GoogleAuthProvider();

1. (Optional)To access APIs from the right authentication provider we need to specify an additional OAuth 2.0.

To add a scope, call addScope. For example: provider.addScope('<https://www.googleapis.com/auth/contacts.readonly>');

1. Authenticate with Firebase using the Google provider object. You can prompt your users to sign in with their Google Accounts either by opening a pop-up window or by redirecting to the sign-in page.

. To sign in with a pop-up window, call signInWithPopup: import { getAuth, signInWithPopup, GoogleAuthProvider } from "firebase/auth";

const auth = getAuth(); signInWithPopup(auth, provider) .then((result) => { // This gives you a Google Access Token. You can use it to access the Google API. const credential = GoogleAuthProvider.credentialFromResult(result); const token = credential.accessToken; // The signed-in user info. const user = result.user; // ... }).catch((error) => { // Handle Errors here. const errorCode = error.code; const errorMessage = error.message; // The email of the user's account used. const email = error.email; // The AuthCredential type that was used. const credential = GoogleAuthProvider.credentialFromError(error); // ... });

.To sign in by redirecting to the sign-in page, call signInWithRedirect: import { getAuth, signInWithRedirect } from "firebase/auth";

const auth = getAuth(); signInWithRedirect(auth, provider);

.Then, you can also retrieve the Google provider's OAuth token by calling getRedirectResult when your page loads: import { getAuth, getRedirectResult, GoogleAuthProvider } from "firebase/auth";

const auth = getAuth(); getRedirectResult(auth) .then((result) => { // This gives you a Google Access Token. You can use it to access Google APIs. const credential = GoogleAuthProvider.credentialFromResult(result); const token = credential.accessToken;

// The signed-in user info.

const user = result.user;

}).catch((error) => { // Handle Errors here. const errorCode = error.code; const errorMessage = error.message; // The email of the user's account used. const email = error.email; // The AuthCredential type that was used. const credential = GoogleAuthProvider.credentialFromError(error); // ... });

Ci cd