Google Login

To implement Google Login in a React application, the easiest and most efficient way is to use Firebase Authentication. Firebase provides a simple SDK to handle various social logins, including Google. Here's a step-by-step guide to implementing Google Login using Firebase:

1. Create a Firebase Project

- Go to the Firebase Console (https://console.firebase.google.com) and create a new project.
- After creating the project, navigate to the Authentication section and enable the Google sign-in method.

2. Set Up Firebase in Your React Project

- Go to the Firebase project settings and register your web app. Copy the Firebase Config information.
- Install Firebase in your React project:

```
npm install firebase
```

- Create a firebase.js file in your project root or in the src directory to initialize Firebase:

```
// firebase.js
import { initializeApp } from "firebase/app";
import { getAuth, GoogleAuthProvider, signInWithPopup, signOut

const firebaseConfig = {
  apiKey: "YOUR_API_KEY",
  authDomain: "YOUR_AUTH_DOMAIN",
  projectId: "YOUR_PROJECT_ID",
  storageBucket: "YOUR_STORAGE_BUCKET",
```

```
  messagingSenderId: "YOUR_MESSAGING_SENDER_ID",
  appId: "YOUR_APP_ID"
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);
const auth = getAuth(app);
const provider = new GoogleAuthProvider();

export { auth, provider, signInWithPopup, signOut };
```

3. Add Google Login Logic in Your React Component

Now, create a React component to implement login and logout functionality.

- Add the following code to App.js or an appropriate component file:

```
// App.js
import React, { useState } from "react";
import { auth, provider, signInWithPopup, signOut } from "./fire

function App() {
  const [user, setUser] = useState(null);

  // Handle Google Sign-in
  const signInWithGoogle = () => {
    signInWithPopup(auth, provider)
      .then((result) => {
        const user = result.user;
        setUser(user);
      })
      .catch((error) => {
        console.error("Error signing in with Google:", error);
      });
  };
```

```
  // Handle Logout
  const handleLogout = () => {
    signOut(auth)
      .then(() => {
        setUser(null);
      })
      .catch((error) => {
        console.error("Error signing out:", error);
      });
  };

  return (
    <div className="App">
      <h1>Google Login with Firebase</h1>
      {user ? (
        <>
          <p>Welcome, {user.displayName}!</p>
          <img src={user.photoURL} alt="User Avatar" />
          <button onClick={handleLogout}>Logout</button>
        </>
      ) : (
        <button onClick={signInWithGoogle}>Sign in with Google</
      )}
    </div>
  );
}

export default App;
```

4. Styling the Google Login Button

You can style the button and layout as you prefer, or follow Google's style guidelines.

```css
/* App.css */
.App {
  text-align: center;
}

button {
  background-color: #4285f4;
  color: white;
  border: none;
  padding: 10px 20px;
  font-size: 16px;
  border-radius: 4px;
  cursor: pointer;
}

button:hover {
  background-color: #357ae8;
}
```

5. Complete Code Overview

firebase.js:

```javascript
// Import Firebase dependencies
import { initializeApp } from "firebase/app";
import { getAuth, GoogleAuthProvider, signInWithPopup, signOut

// Firebase configuration object from Firebase Console
const firebaseConfig = {
  apiKey: "YOUR_API_KEY",
  authDomain: "YOUR_AUTH_DOMAIN",
  projectId: "YOUR_PROJECT_ID",
  storageBucket: "YOUR_STORAGE_BUCKET",
  messagingSenderId: "YOUR_MESSAGING_SENDER_ID",
  appId: "YOUR_APP_ID"
```

```
};

// Initialize Firebase app
const app = initializeApp(firebaseConfig);
const auth = getAuth(app);
const provider = new GoogleAuthProvider();

export { auth, provider, signInWithPopup, signOut };
```

App.js:

```
import React, { useState } from "react";
import { auth, provider, signInWithPopup, signOut } from "./fire
import "./App.css";

function App() {
  const [user, setUser] = useState(null);

  const signInWithGoogle = () => {
    signInWithPopup(auth, provider)
      .then((result) => {
        setUser(result.user);
      })
      .catch((error) => {
        console.error("Error signing in with Google:", error);
      });
  };

  const handleLogout = () => {
    signOut(auth)
      .then(() => {
        setUser(null);
      })
      .catch((error) => {
        console.error("Error signing out:", error);
      });
```

```
  };

  return (
    <div className="App">
      <h1>Google Login with Firebase</h1>
      {user ? (
        <>
          <p>Welcome, {user.displayName}!</p>
          <img src={user.photoURL} alt="User Avatar" />
          <button onClick={handleLogout}>Logout</button>
        </>
      ) : (
        <button onClick={signInWithGoogle}>Sign in with Google</
      )}
    </div>
  );
}

export default App;
```

6. Test Google Login

Test the Google login by signing in with the email addresses you authorized in Firebase's OAuth consent screen settings. When you run your React application and click the Google login button, you should see a Google account selection screen. Once you log in, the user information will be displayed on the screen.

Summary

- Set up Firebase Project and Enable Google Login
- Connect Firebase to React Project and Configure
- Implement Google Login Logic in Reactement Google Login Logic in React