### Sign In with Google for web

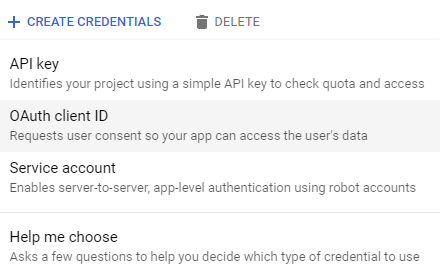
#### Objectives and Outcomes

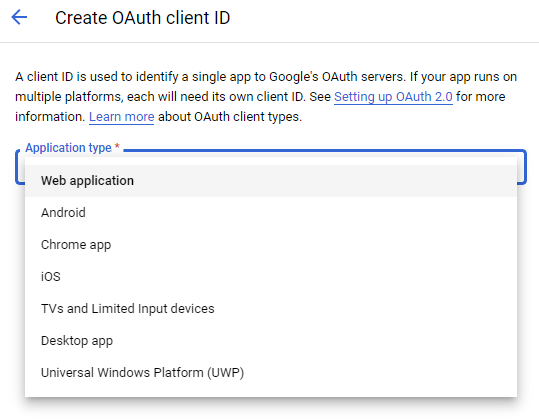
In this exercise, you will configure and build a login function by Google account using Google Identity. At the end of this exercise, you will be able to:

* Setup your OAuth configuration
* Create a login function with Google using Google Identity.

#### Login and set up your Google API client ID

* Access to the link: <https://console.developers.google.com/apis>
* Open the Credentials page of the Google APIs console.
* Create or select a Google APIs project, chose OAuth client ID.



**

*Chose Web application*

Graphical user interface, text, application, email

Description automatically generated

*Enter your URLs for authorizing*



Copy your Client ID

**Load the client library**

* Create a React application, open the index.html file in public folder, update the code:

<script src="https://accounts.google.com/gsi/client" async defer></script>

**Use the One Tap JavaScript API**

* Open the App.js file, update code follow:

import { useEffect } from 'react';

…….

const handleCredentialResponse = (response)=> {

console.log("Encoded JWT ID token: " + response.credential);

document.getElementById('buttonDiv').hidden =true;

}

useEffect(() => {

/\* global google\*/

window.onload = function () {

google.accounts.id.initialize({

client\_id: "Your Client ID……",

callback: handleCredentialResponse

});

google.accounts.id.renderButton(

document.getElementById("buttonDiv"),

{ theme: "outline", size: "large" } // customization attributes

);

google.accounts.id.prompt(); // also display the One Tap dialog

}

}, []);

return (

<div id='buttonDiv'></div>

);

* Open the App.js file, update code follow:
* Install jwt-decode library for decoding JWTs token to get the information from JWT token:

npm install jwt-decode

* Import and use the jwt-decode in handleCredentialResponse function:

……

import jwt\_decode from "jwt-decode";

…..

const handleCredentialResponse = (response)=> {

console.log("Encoded JWT ID token: " + response.credential);

var decoded = jwt\_decode(response.credential);

}

* Create a state for store your login statement and show some information from JWT decode:

import { useState } from 'react';

…..

const [user, setUser] = useState({})

const handleCredentialResponse = (response)=> {

console.log("Encoded JWT ID token: " + response.credential);

var decoded = jwt\_decode(response.credential);

setUser(decoded);

}

…….

return (

<>

<div id='buttonDiv'></div>

{user &&

<div>

<h5>{user.name}</h5>

</div>

}

</>

);

}

**Update some code for complete your application**

* Full code:

import { useEffect } from 'react';

import { useState } from 'react';

import jwt\_decode from "jwt-decode";

function App() {

const [user, setUser] = useState({})

const handleCredentialResponse = (response)=> {

console.log("Encoded JWT ID token: " + response.credential);

var decoded = jwt\_decode(response.credential);

setUser(decoded);

document.getElementById('buttonDiv').hidden =true;

}

const handleLogOut =(e)=>{

setUser({});

document.getElementById('buttonDiv').hidden =false;

}

useEffect(() => {

/\* global google\*/

window.onload = function () {

google.accounts.id.initialize({

client\_id: "Your Client ID…..",

callback: handleCredentialResponse

});

google.accounts.id.renderButton(

document.getElementById("buttonDiv"),

{ theme: "outline", size: "large" } // customization attributes

);

google.accounts.id.prompt(); // also display the One Tap dialog

}

}, []);

return (

<>

<div id='buttonDiv'></div>

{Object.keys(user).length!=0 &&

<button onClick={handleLogOut}>logout</button>

}

{user &&

<div>

<h5>{user.name}</h5>

</div>

}

</>

);

}

export default App;

#### Conclusions

In this exercise, you learnt how to configure and build a login function by Google account using Google Identity platform.