Google SSO sign in steps:  
  
  
1> Go to <https://console.cloud.google.com/> => click on project => New project.

2> Select a particular project and click on **APIs & Services =>** under the section of **Quick access**.

3> Click on **Credentials** => **Configure consent screen** =>select **External** and click on the **create** button.

4> After completion of **OAuth consent screen** setup, Click on **Credentials.**

5> Under **Credentials** click on **Create credentials** => **OAuth client id => Select application type, fill data.**6> How to create a SHA-1 **signing certificate fingerprint ?**

expo credentials:manager -p android

=> update upload keystore

=> generate new keystore

=> go back to expo overview

=> google certificate fingerprint copy and paste into cloud project.  
  
7> Click on the **create** button.  
  
8> Same for **iOS** and **web** as well.

9> Copy **client id** and paste it into your project.

10> Here is the expo native code for your reference.

import { useEffect, useState } from "react";

import { StatusBar } from 'expo-status-bar';

import { StyleSheet, Text, View, Button, Image } from 'react-native';

import \* as WebBrowser from "expo-web-browser";

import \* as Google from "expo-auth-session/providers/google";

import AsyncStorage from "@react-native-async-storage/async-storage";

WebBrowser.maybeCompleteAuthSession();

export default function App() {

const [token, setToken] = useState("");

const [userInfo, setUserInfo] = useState(null);

const [request, response, promptAsync] = Google.useAuthRequest({

androidClientId: "paste your android client id",

iosClientId: "paste your ios client id",

webClientId: "paste your web client id",

});

useEffect(() => {

handleSigninwithGoogle();

}, [response, token]);

const getLocalUser = async () => {

const data = await AsyncStorage.getItem("@user");

if (!data) return null;

return JSON.parse(data);

};

async function handleSigninwithGoogle() {

const user = await getLocalUser();

console.log("user", user);

if (!user) {

if (response?.type === "success") {

// setToken(response.authentication.accessToken);

getUserInfo(response.authentication.accessToken);

}

} else {

setUserInfo(user);

console.log("loaded locally");

}

}

const getUserInfo = async (token) => {

if (!token) return;

try {

const response = await fetch(

"https://www.googleapis.com/userinfo/v2/me",

{

headers: { Authorization: `Bearer ${token}` },

}

);

const user = await response.json();

await AsyncStorage.setItem("@user", JSON.stringify(user));

setUserInfo(user);

} catch (error) {

// Add your own error handler here

}

};

return (

<View style={styles.container}>

{!userInfo ? (

<Button

title="Sign in with Google"

disabled={!request}

onPress={() => {

promptAsync();

}}

/>

) : (

<View style={styles.card}>

{userInfo?.picture && (

<Image source={{ uri: userInfo?.picture }} style={styles.image} />

)}

<Text style={styles.text}>Email: {userInfo.email}</Text>

<Text style={styles.text}>

Verified: {userInfo.verified\_email ? "yes" : "no"}

</Text>

<Text style={styles.text}>Name: {userInfo.name}</Text>

{/\* <Text style={styles.text}>{JSON.stringify(userInfo, null, 2)}</Text> \*/}

</View>

)}

<Button

title="remove local store"

onPress={async () => await AsyncStorage.removeItem("@user")}

/>

</View>

);

}

const styles = StyleSheet.create({

container: {

flex: 1,

backgroundColor: '#fff',

alignItems: 'center',

justifyContent: 'center',

},

});