

Add Google Authentication to your React project in 10 Mins

- The Complete Reference



Sivanesh Shanmugam

Posted on Jun 27, 2020 • Updated on Oct 26, 2020

Add Google Login to your React Apps in 10 mins

#react #security #webdev #googlecloud



Sign in with Google

As a client developer, Creating apps with authentication might be a tedious process. As we need to handle authentication in the server and maintain the code. It does have some multiple cases to be maintained based on authentication mechanisms. (Basic, OAuth, etc) This blog helps you create a standalone backend less authentication for your react app. (or additionally) you can add server configurations too to authenticate with your server.

Why OAuth?

There are many reasons to use OAuth.

- It is secure.
- It doesn't require any credentials from the user. So no need of remembering multiple passwords.
- Websites can also get essential information about users without spending much time in forms.

And specifically in our case, we don't need a server to authenticate or to get initial details of the user. 😊

I found a package which is called as `react-google-login`. Which provides simple mechanisms to add the Google login. You can either directly use their `GoogleLogin` component or you can use custom buttons. They also provide custom hooks like `useGoogleLogin` and `useGoogleLogout` so it will be easy for hooks lovers. *(Both methods are described below)*

The [documentation](#) they provided for this repository is awesome. But it's missing in some ways which are addressed in this blog. Like in detail, this blog helps in creating your application in the Google developer console and access `clientId`. The initial `access_token` obtained by this package will last for only one hour *(For security reasons)*. And we need to iterate the process to access the new `access_token` using `refresh_token`. So you can make a **production-ready application**.

Steps

- We need to create an application in the Google developer console. It provides `clientId` is used to identify your application for authentication details. Follow the below steps to get the client ID.
 1. Go to the [Credentials page](#). *(if you are new, then [create a project](#) and follow these steps)*
 2. Click **Create credentials > OAuth client ID**.
 3. Select the **Web application** type.
 4. Name your OAuth 2.0 client and click **Create**
- Make sure you provided your domain and redirect URL. So that Google identifies the origin domain to which it can provide authentication.

URIs

http://localhost:3000

https://sivanesh-s.github.io

+ ADD URI

Authorized redirect URIs ?

For use with requests from a web server

URIs

http://localhost:3000

https://sivanesh-s.github.io/react-google-authentication/

+ ADD URI

You can also add your local route for development. Now authentication setup in Google developer console is ready.

Lets code

The Components way



The Hooks way



Sign in with Google



Sign out

Let's start with code. [View my repo](#) for accessing all code snippets. View [Demo](#).

In your CRA install `react-google-login` package

```
npm i react-google-login
```

Create a `Login` component that acts as a login button.

```
import React from 'react';
import { GoogleLogin } from 'react-google-login';

const clientId = 'YOUR_CLIENT_ID.apps.googleusercontent.com';


function Login() {
  const onSuccess = (res) => {
    console.log('[Login Success] currentUser:', res.profileObj);
  };

  const onFailure = (res) => {
    console.log('[Login failed] res:', res);
  };

  return (
    <div>
      <GoogleLogin
        clientId={clientId}
        buttonText="Login"
        onSuccess={onSuccess}
        onFailure={onFailure}
        cookiePolicy={'single_host_origin'}
        style={{ marginTop: '100px' }}
        isSignedIn={true}
      />
    </div>
  );
}

export default Login;
```

Similarly, create a Logout component



```
import React from 'react';
import { GoogleLogout } from 'react-google-login';

const clientId = 'YOUR_CLIENT_ID.apps.googleusercontent.com';

function Logout() {
  const onSuccess = () => {
    alert('Logout made successfully 🙌');
  };

  return (
    <div>
      <GoogleLogout
        clientId={clientId}
        buttonText="Logout"
        onLogoutSuccess={onSuccess}
      ></GoogleLogout>
    </div>
  );
}

export default Logout;
```

And add both in the required location of your app. Mine is in `App.js`



```
import React from 'react';
import './App.css';
import Login from './components/Login';
import Logout from './components/Logout';

function App() {
  return (
    <div className="App">
      <Login />
      <Logout />
    </div>
  );
}

export default App;
```

Now running your application will show `profileObj` in the console after logging in.

Congrats 🎉 You successfully made it 😊.

But after 1 hour your `tokenId` gets expired and hence it won't be used to access data or authenticate users. And hence we need to generate new `tokenId`. To make things work we need to add some additional cases in the `Login` component.



```

import React from 'react';

import { GoogleLogin } from 'react-google-login';
// refresh token
import { refreshTokenSetup } from '../utils/refreshToken';

const clientId = 'YOUR_CLIENT_ID.apps.googleusercontent.com';

function Login() {
  const onSuccess = (res) => {
    console.log('[Login Success] currentUser:', res.profileObj);

    // initializing the setup
    refreshTokenSetup(res);
  };

  const onFailure = (res) => {
    console.log('[Login failed] res:', res);
  };

  return (
    <div>
      <GoogleLogin
        clientId={clientId}
        buttonText="Login"
        onSuccess={onSuccess}
        onFailure={onFailure}
        cookiePolicy={'single_host_origin'}
        style={{ marginTop: '100px' }}
        isSignedIn={true}
      />
    </div>
  );
}

export default Login;

```

refreshTokenSetup function will take care of handling new tokenIds


```

export const refreshTokenSetup = (res) => {
  // Timing to renew access token
  let refreshTiming = (res.tokenObj.expires_in || 3600 - 5 * 60) * 1000;

  const refreshToken = async () => {
    const newAuthRes = await res.reloadAuthResponse();
    refreshTiming = (newAuthRes.expires_in || 3600 - 5 * 60) * 1000;
    console.log('newAuthRes:', newAuthRes);
    // saveUserToken(newAuthRes.access_token); <-- save new token
    console.log('new auth Token', newAuthRes.id_token);

    // Setup the other timer after the first one
    setTimeout(refreshToken, refreshTiming);
  };

  // Setup first refresh timer
  setTimeout(refreshToken, refreshTiming);
};

```

This function checks for `expires_in` timestamp or our custom time (*before the token expires*) and calls `reloadAuthResponse` which is a util function provided by the library and it handles `refresh_token` and obtains new `tokenId`.

And Yeah 😊 Google login is added to your application successfully 🎉. So now you can access the user's **name, photo URL, email, google Id**, etc.

Additional

The above way uses the Google Login default button. You can also use your custom button using `render` prop.

```

<GoogleLogin
  clientId={clientId}
  render={renderProps => (
    <button onClick={renderProps.onClick} disabled={renderProps.disabled}>
      This is my custom Google button
    </button>
  )}
  buttonText="Login"
  onSuccess={responseGoogle}
  onFailure={responseGoogle}
  cookiePolicy={'single_host_origin'}
/>

```

We can also implement the same functionality using **React Hooks**. ❤️

LoginHooks.js



```

import React from 'react';
import { useGoogleLogin } from 'react-google-login';

// refresh token
import { refreshTokenSetup } from '../utils/refreshToken';

const clientId = 'YOUR_CLIENT_ID.apps.googleusercontent.com';

function LoginHooks() {
  const onSuccess = (res) => {
    console.log('Login Success: currentUser:', res.profileObj);
    refreshTokenSetup(res);
  };

  const onFailure = (res) => {
    console.log('Login failed: res:', res);
  };

  const { signIn } = useGoogleLogin({
    onSuccess,
    onFailure,
    clientId,
    isSignedIn: true,
    accessType: 'offline',
  });

  // My custom button
  return (
    <button onClick={signIn} className="button">
      </img>
      <span className="buttonText">Sign in with Google</span>
    </button>
  );
}

export default LoginHooks;

```

LogoutHooks.js

```

import React from 'react';
import { useGoogleLogout } from 'react-google-login';

const clientId = 'YOUR_CLIENT_ID.apps.googleusercontent.com';

function LogoutHooks() {
  const onLogoutSuccess = (res) => {
    alert('Logged out Successfully 🙌');
  };

  const onFailure = () => {
    console.log('Handle failure cases');
  };

  const { signOut } = useGoogleLogout({
    clientId,
    onLogoutSuccess,
    onFailure,
  });

  return (
    <button onClick={signOut} className="button">
      </img>

      <span className="buttonText">Sign out</span>
    </button>
  );
}

export default LogoutHooks;

```

Server-side verification

If you wish to add Server side verification, Send the `tokenId` from client to server once `onSuccess` in `Login` component called.

So while handling authenticated routes, All requests from the client need to send the `tokenId` of the user in the header as **Bearer token**. At Server, once token received it must be verified either this `tokenId`


- belongs to the current application.

- Has it expired

You can do them manually but google suggests using their authentication library (*Minimal effort required*).

In the Server side (*Here Node.js is used*).

Install Google's official supported library `google-auth-library` package which is used to authenticate and verify OAuth apps.



```
// token confirmation
const { OAuth2Client } = require('google-auth-library');
const client = new OAuth2Client(process.env.GOOGLE_CLIENT_ID);

const googleAuth = async (token) => {
  const ticket = await client.verifyIdToken({
    idToken: token,
    audience: process.env.GOOGLE_CLIENT_ID,
  });
  const payload = ticket.getPayload();
  // console.log('payload:', payload);

  console.log(`User ${payload.name} verified`);

  const { sub, email, name, picture } = payload;
  const userId = sub;
  return { userId, email, fullName: name, photoUrl: picture };
};

module.exports = googleAuth;
```

Here, With our `GOOGLE_CLIENT_ID` sent it verifies whether this token belongs to our application or not. And it parses them and provides profile details in the `getPayload` function. And you can use them to access user data.

Any queries feel free to comment or chat with me personally in my social media accounts below.

I love to be connected with developers. 😊

[Twitter](#) | [LinkedIn](#) | [GitHub](#)

Discussion (35)



Nick Trierweiler • Sep 26 '20 • Edited on Sep 26



This article is great! Thanks for sharing your example and this writeup! I personally would prefer code snippets over images but I am happy that you also shared the source code and I was able to get this working without too much trouble.

There is a confused sentence above which could use refactoring. "To make things work we need to make things work we have to add some additional cases in the Login component."

For others following this tutorial who get a failure message because of cookies try changing `cookiePolicy={'single_host_origin'}` to `cookiePolicy='http://localhost:8000/'` or whatever your localhost is. Adding the address in google wasn't enough. If I am mistaken please comment and I will try that too!



Sivanesh Shanmugam 🌟 • Oct 26 '20



Thanks for your Feedback ❤️ Sorry for the late response. And I changed that confused sentence.



olsard • Oct 26 '20 • Edited on Oct 26



Great! Thanks for sharing.

There is a little issue with google icon in /public folder.
It could be moved into /src instead

```
import GoogleIcon from '../assets/icons/google.svg';  
<img src={GoogleIcon} alt="google login" className="icon" />
```



Pratik Patil • Jan 12 '21



The article is Awesome but I would like a little help. we already have a logout button and

how can I add this to it without making any change to the existing logout code which gets executed onClick of that button



Dilip-Valiya99 • Aug 6 '21



✕ refreshTokenSetup function is not working b'coz if someone will refresh page after login then setTimeout() will be cleared also refreshTokenSetup function is only run after on success can we call it after expiry time of oath token?



sathishkumar18594 • Jan 13 '21



✕ Thanks for this article!!

I'm getting "Failed to retrieve certificates: request to googleapis.com/oauth2/v1/certs failed, reason: self signed certificate in certificate chain" error message when I send google's token id to my mock API for further validations.

Also noticed the token id does not have valid signature.



Akash Shyam • Jul 15 '21



✕ Great article! An absolute life saver. Could you expand a bit about using google login with our backend API. So in most applications, you have a custom contact form plus the social logins. I'd like to save the data like email, password etc to my backend(Mongoose + NodeJS).

I could directly just send an API call to the backend in the onSuccess method and register the user in my DB. Is this the correct approach?



Ivan Chavez Escamilla • Sep 21 '20



✕ Great tutorial!
One thing though, I believe the last image is the same as the one for LogoutHooks. There's no image for the Server Side verification code



Sivanesh Shanmugam 🌟 • Oct 26 '20



✕ Thanks. Changed them



Manish Kumar • Apr 24 '21 • Edited on Apr 27



Hi Sivanesh, This Article is very helpful for me, I was using this same code for google auth setup but I am getting one error - Token is getting expired after 1 hour. I have used refresh token setup also , but didn't get success . Error - "Uncaught exception 'Google_Auth_Exception' with message 'Token used too late, 1410345101 > 1410341783: " Can you please help me on this... Thanks [@sivaneshs](#)



Manish Kumar • Apr 26 '21



[@sivaneshs](#) please help



Comment deleted



Sivanesh Shanmugam 🌟 • Feb 23 '21 • Edited on Feb 23



Yeah, I found them after some hours of debugging and found this helpful code. Thanks for your advice I'll follow them in forthcoming blogs. But actually, It's not your code. I copied the snippet from [github.com/anthonyjgrove/react-goo...](#) which he posted two months before your comment on the thread.

I don't know the reason why you did copy the same code and posted it in the same thread, and claiming for a reference. 😂



Kiana-Alize Diaz • Dec 6 '20



Hi under what file would we place server side verification?



Sivanesh Shanmugam 🌟 • Dec 7 '20



Placing it in a middleware would help.



Pit McDonald • Dec 9 '20





Can you please specify how to do this?



Pragati Varshney • Jan 4



Hey,

Its a great article for beginners like me.

But, I am facing a challenge with log out, cookies and local storage are not getting cleared on log out and when I try to login the second time it auto-logins.

can you please guide me how to fix or what could have gone wrong?



codeuiandy • Nov 15 '21



Thanks [@sivaneshs](#) this was very helpfull and save me the time of debugging and all



leoigel • Nov 26 '20



i got this error :

Uncaught qw {message: "Missing required parameter 'client_id'", xM: true, stack: "gapi.auth2.ExternallyVisibleError: Missing require...tXYzGQD7-oxLsteJNmllmx3Ysqg/cb=gapi.loaded_0:1:15"}



anoopvm • Apr 5



I ended up with this because of wrong case. I used clientid instead of "clientId"



Michelle Ann Terrazas • Aug 3 '21



Thank you so much for this, it helped me immensely with my task for my project last week! :)



Kalin Panmanee • May 24 '21





I'm trying to finish coding on the server side verification part but still struggle with a this problem. "TypeError: jwt.split is not a function " . Does anyone can help with this issue? By the way, I already got the token ID.




joedotnot • Dec 17 '21 • Edited on Dec 17




- 

Can you do other articles for login with Facebook, Github, Twitter, etc :)
- 


Rahul More • Nov 21 '21

How can i loader the google login button Until google send any response?
- 


Sudharsan Srinivasan • Nov 7 '21

This is just perfect article for someone like me who is really new to UI world.
Sivanesh shamnugam is the my saviour today.
- 


chayagolo1 • Nov 4 '21

Hi
This is amazing and works as wxpected in http.
But when i changed the http protocol to https i got this error:
"redirect_uri_mismatch".
how can i resolve it?
thank you so much!!
- 


Himanshu Khaitan • Oct 20 '21

Hey! I am stuck with a mern stack auth problem. Can you help..?
I am really stuck with auth stuff
- 

Anton Istomin • Feb 24 '21

Nice article. But it got me thinking: what about non-pure setTimeout and whether we can
(or have to) extract it using redux-saga?
- 

sd-roop • Nov 24 '21

Thank you Shivanesh. On Logging in, I get a failure "popup_closed_by_user". Please advise.
- 

Congying • Sep 8 '21



Thanks, it works fine for me. But I'm wondering how to get user info in another react components. Thanks in advance.



Sivanesh Shanmugam • Sep 9 '21



You can add it in a React context and use it anywhere inside your components.



Pratik Wadekar • Oct 14 '21



Can someone help me know how do I persist the login with react-google-login?



namansamra • Mar 9 '21



hi I have a query like if i have a database and then user tries to login with google then how can I find the user from the database .



Sivanesh Shanmugam • Mar 9 '21



With the token obtained, you can extract the user's information in which you can get email, Google user ID etc. With that I think you can find the user.



Jules Grenier • Mar 31 '21



This is **exactly** what I was looking for!! Thank you so much! ❤️🐾

[View full discussion \(35 comments\)](#)

[Code of Conduct](#) • [Report abuse](#)



Sivanesh Shanmugam

A full-stack developer who loves to work with React.js, Express, Node.js, Redux, Javascript, PWA (Progressive Web Apps), Service workers, HTML, CSS, custom plugins in Babel, ESLint.

LOCATION

Madurai

EDUCATION

B.Tech IT

WORK

Member Technical Staff (React Developer) at Zoho Technologies

JOINED

Mar 12, 2020

More from [Sivanesh Shanmugam](#)

A Minimalistic Web Portfolio for all devs 🧐

#productivity #portfolio #design #webdev
