**Spring Security - OAuth2 Example Using GitHub | SpringBoot**

In this tutorial we will see How to use OAuth2 in Spring Boot Application. Lets first see what is this OAuth2 or **Open Authorization. OAuth** ("**O**pen **Auth**orization"[[1]](https://en.wikipedia.org/wiki/OAuth#cite_note-NIST-1)[[2]](https://en.wikipedia.org/wiki/OAuth#cite_note-RFC6749-2)) is an open standard for access [delegation](https://en.wikipedia.org/wiki/Delegation_(computer_security)).

commonly used as a way for internet users to grant websites or applications access to their information on other websites but without giving them the passwords.[[3]](https://en.wikipedia.org/wiki/OAuth#cite_note-3)[[4]](https://en.wikipedia.org/wiki/OAuth#cite_note-4) This mechanism is used by companies such as [Amazon](https://en.wikipedia.org/wiki/Amazon_(company)),[[5]](https://en.wikipedia.org/wiki/OAuth#cite_note-5) [Google](https://en.wikipedia.org/wiki/Google), [Facebook](https://en.wikipedia.org/wiki/Facebook), [Microsoft](https://en.wikipedia.org/wiki/Microsoft), and [Twitter](https://en.wikipedia.org/wiki/Twitter) to permit the users to share information about their accounts with third-party applications or websites.

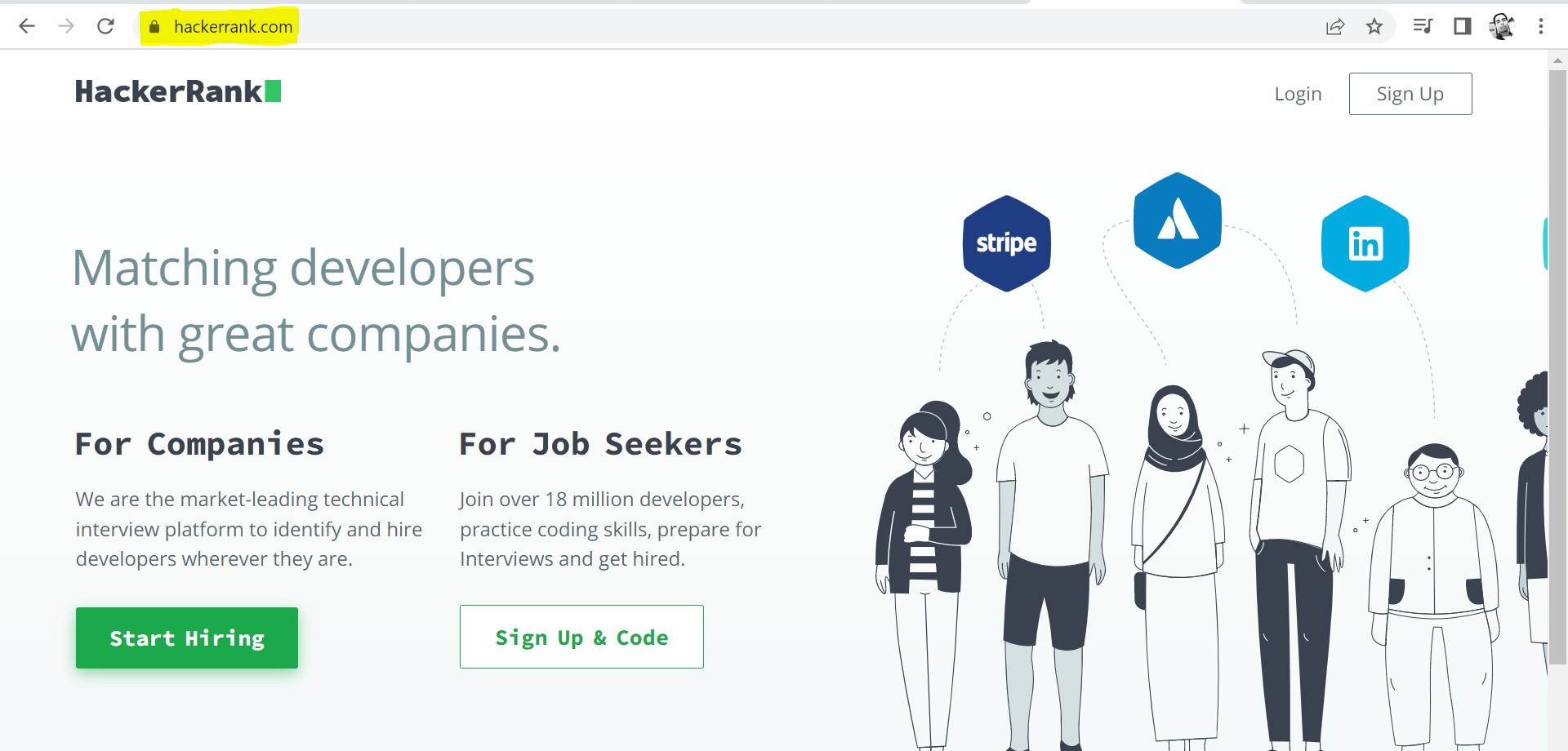
Generally, OAuth provides clients a "secure delegated access" to server resources on behalf of a resource owner. It specifies a process for resource owners to authorize third-party access to their server resources without providing credentials. Designed specifically to work with [Hypertext Transfer Protocol](https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol) (HTTP), OAuth essentially allows [access tokens](https://en.wikipedia.org/wiki/Access_token) to be issued to third-party clients by an authorization server, with the approval of the resource owner. The third party then uses the access token to access the protected resources hosted by the resource server.[[2]](https://en.wikipedia.org/wiki/OAuth#cite_note-RFC6749-2) In particular, OAuth 2.0 provides specific authorization flows for web applications, desktop applications, mobile phones, and [smart devices](https://en.wikipedia.org/wiki/Smart_device).

In this tutorial we will see How to use OAuth2 in Spring Boot Application. Let’s first see what is this **OAuth2** or **Open Authorization**. Before jump to the concepts lets first see what is OAuth2 or Open Authorization.

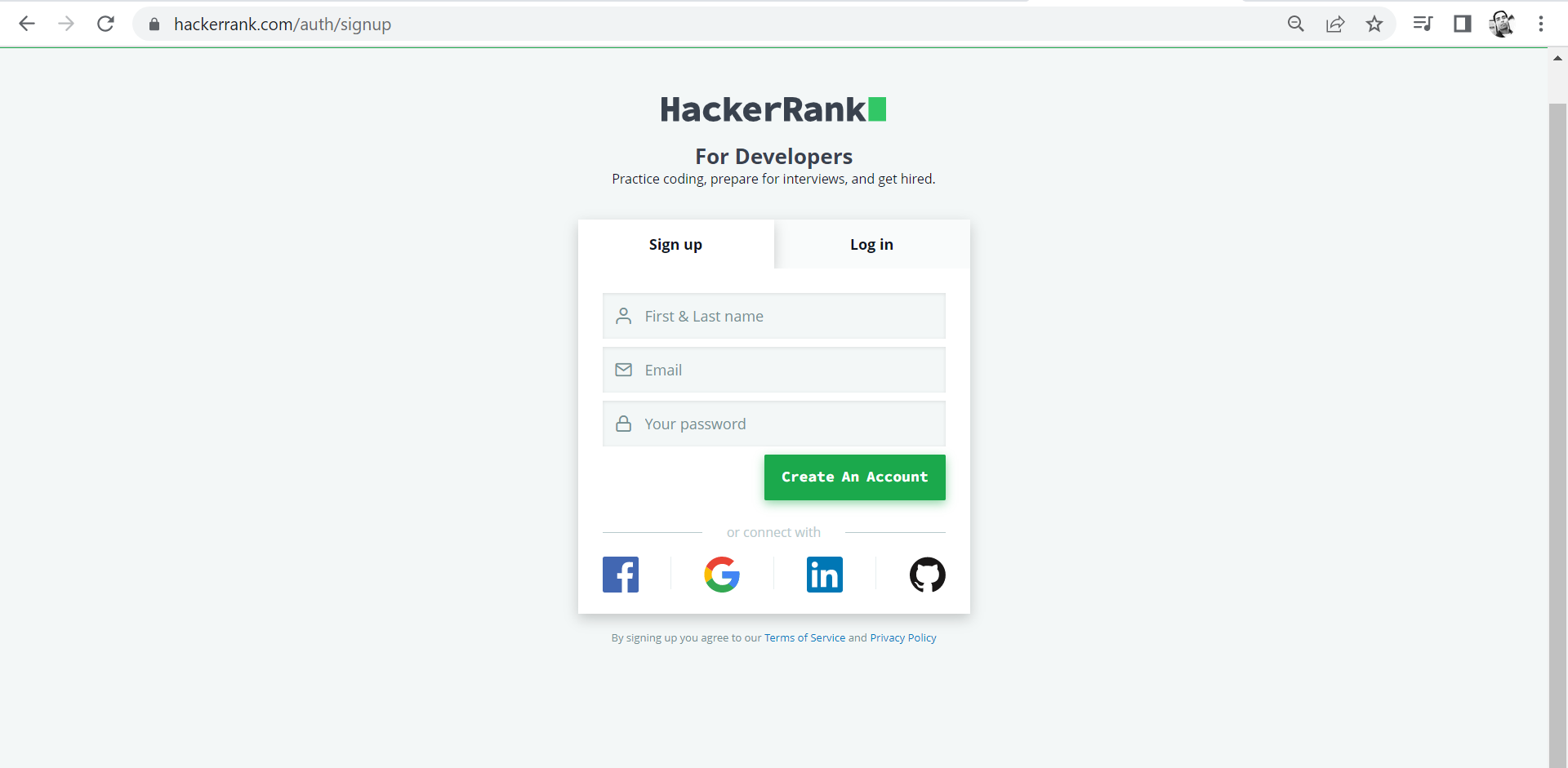
Let me show u one live example so that we can sink with the concepts. So, let me go to the browser I opened **HackerRank.com**

**How OAuth2 (Authorize Request Delegation) Works Internally??**

1. User choose Sign up using GitHub.
2. User need to login to GitHub for Authentication.
3. Page will Redirect for GitHub Authorization.
4. Once User Authorized it, then GitHub will generate one access token and return back to the client.
5. Client will send request with same access token provided by GitHub.
6. GitHub Resource Server will decode that token and validate it.
7. If it is valid token, then client can receive user information from GitHub Resource Server.

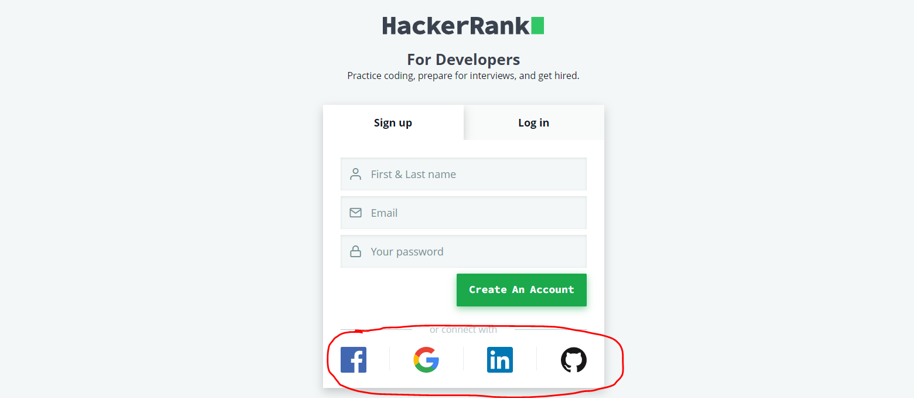


I choose this website because OAuth2 concepts already Implemented here. If I click on Signup & Code. It redirects me to the Signup page.



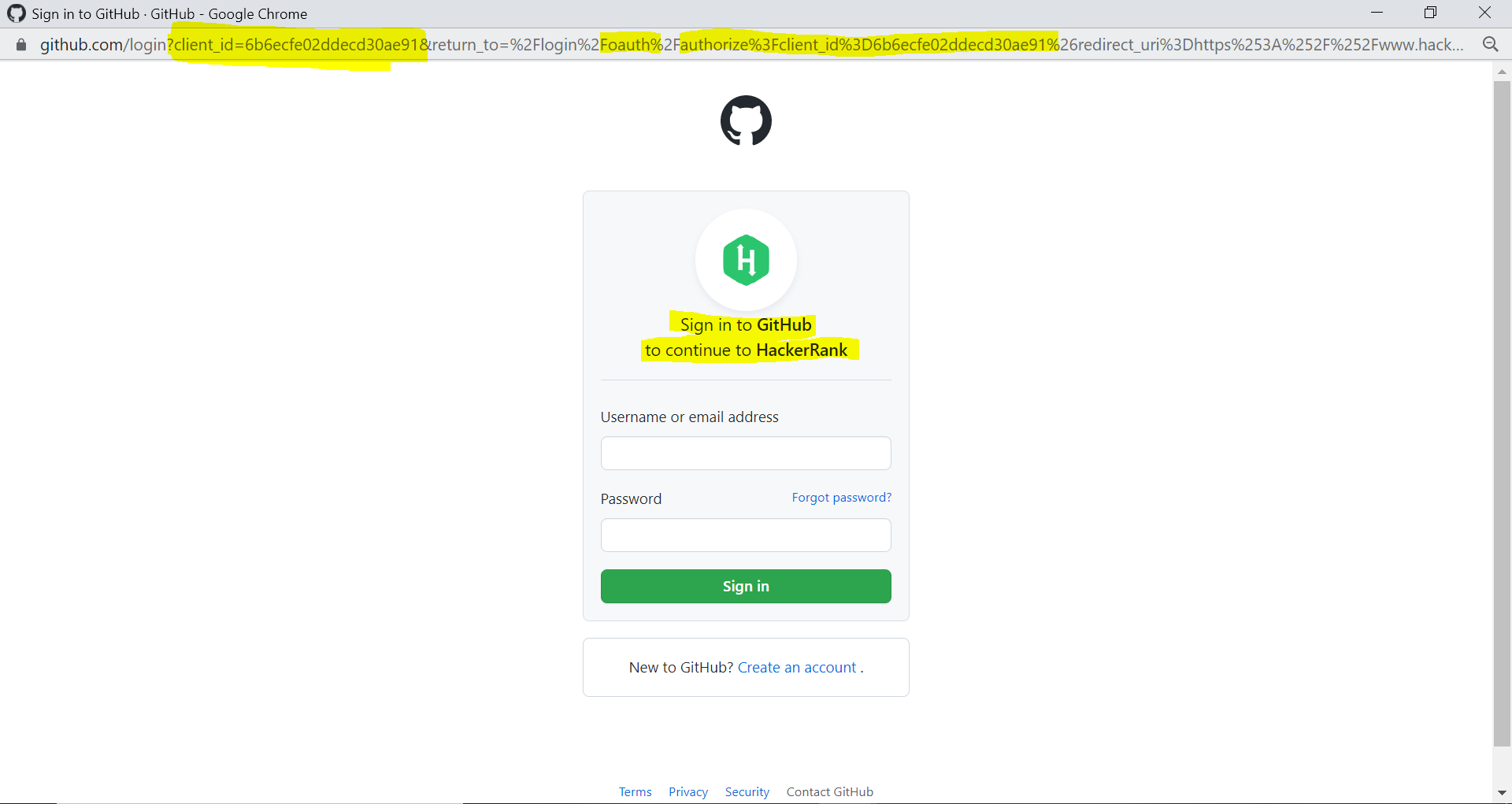
So here as a User we can give First Name, Last Name, Email and Password to continue the signup.

But if I don’t want to give the Input, I can delegate access Request to any of our third-party application. I can choose **Facebook** or **Github** or **LinkedIn**.

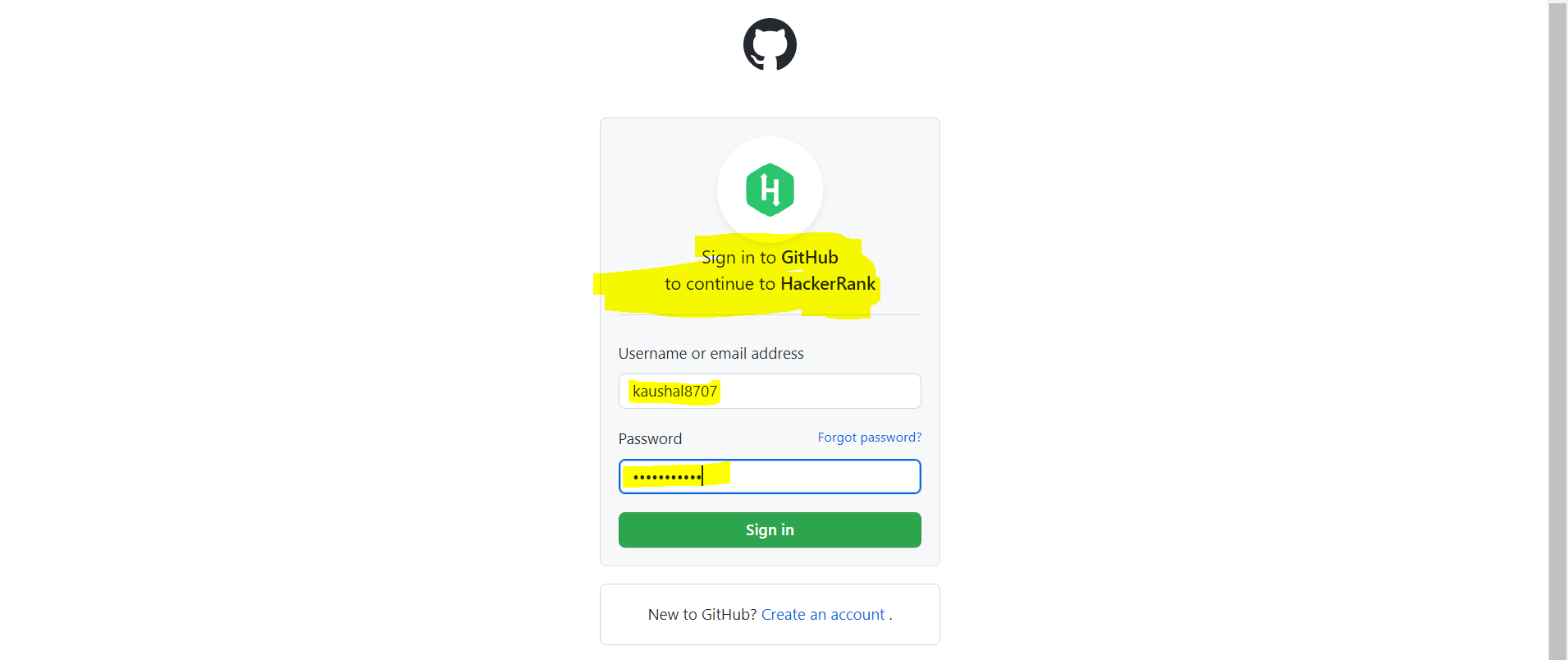


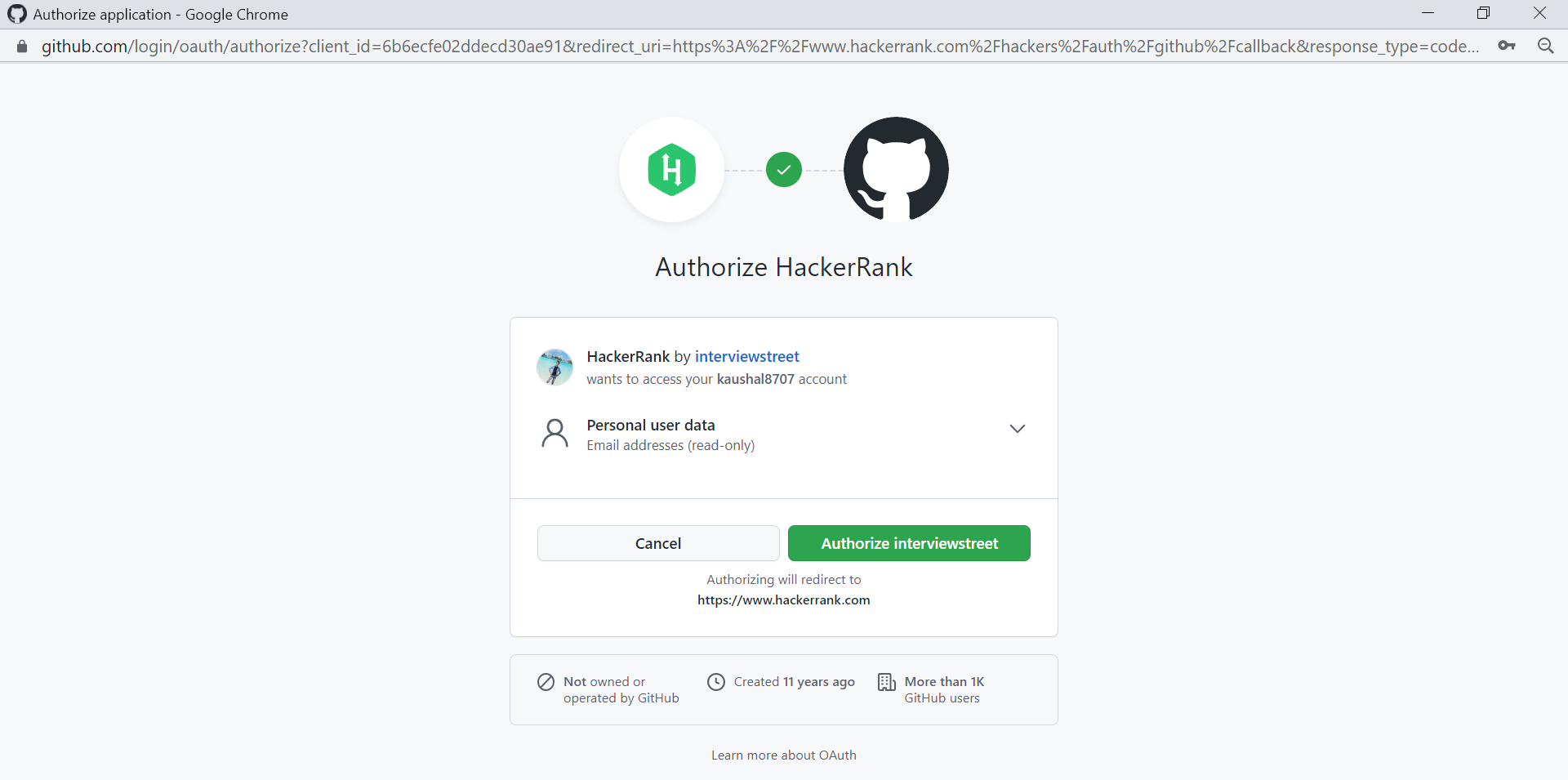
So instead of user gives the input he can access request to third party to get this First Name, Email or any other fields.

So, let me show you this if u click on this Github.



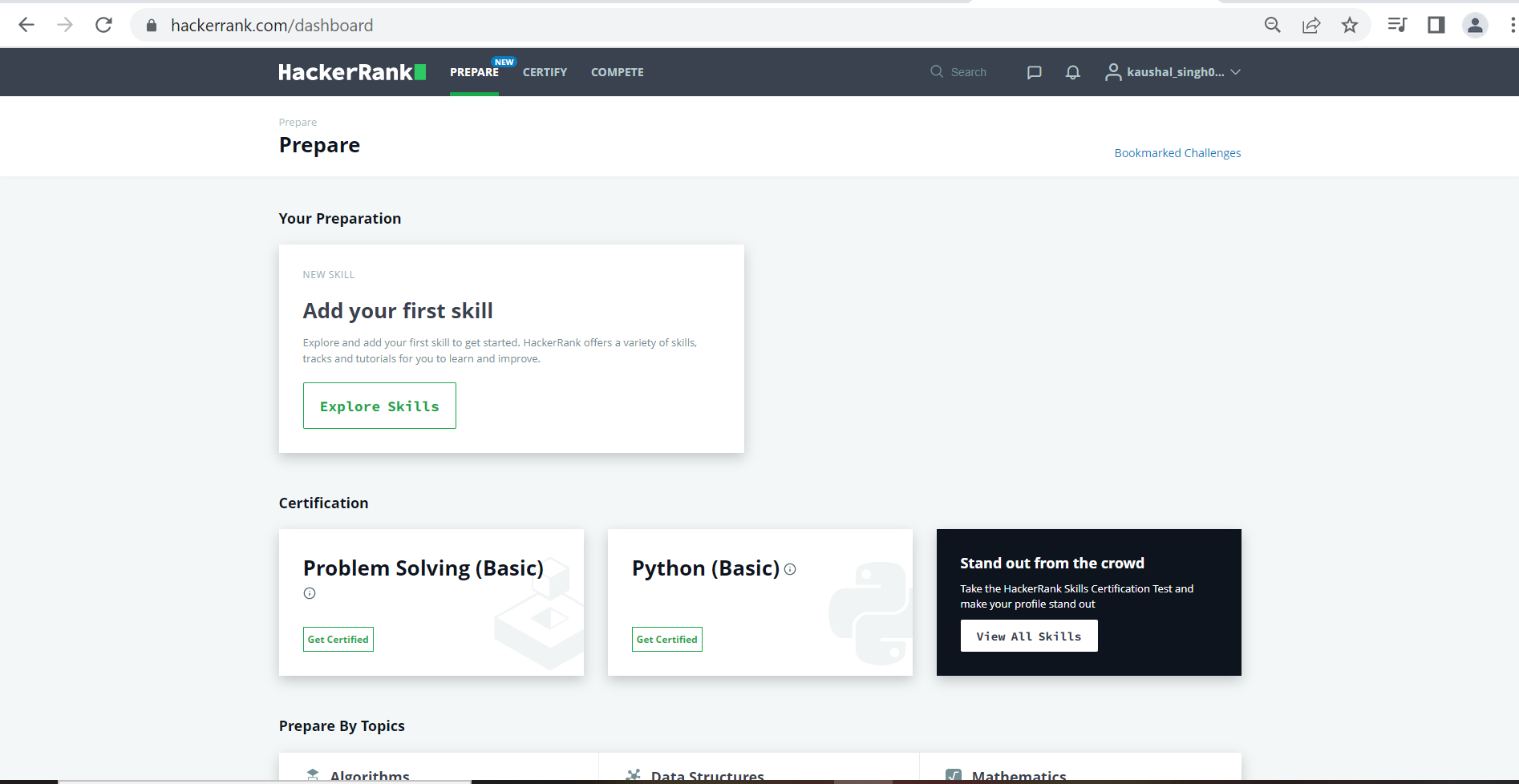
I need to provide the Github Credentials here. If you Observed Sign in to Github to continue to HackerRank.





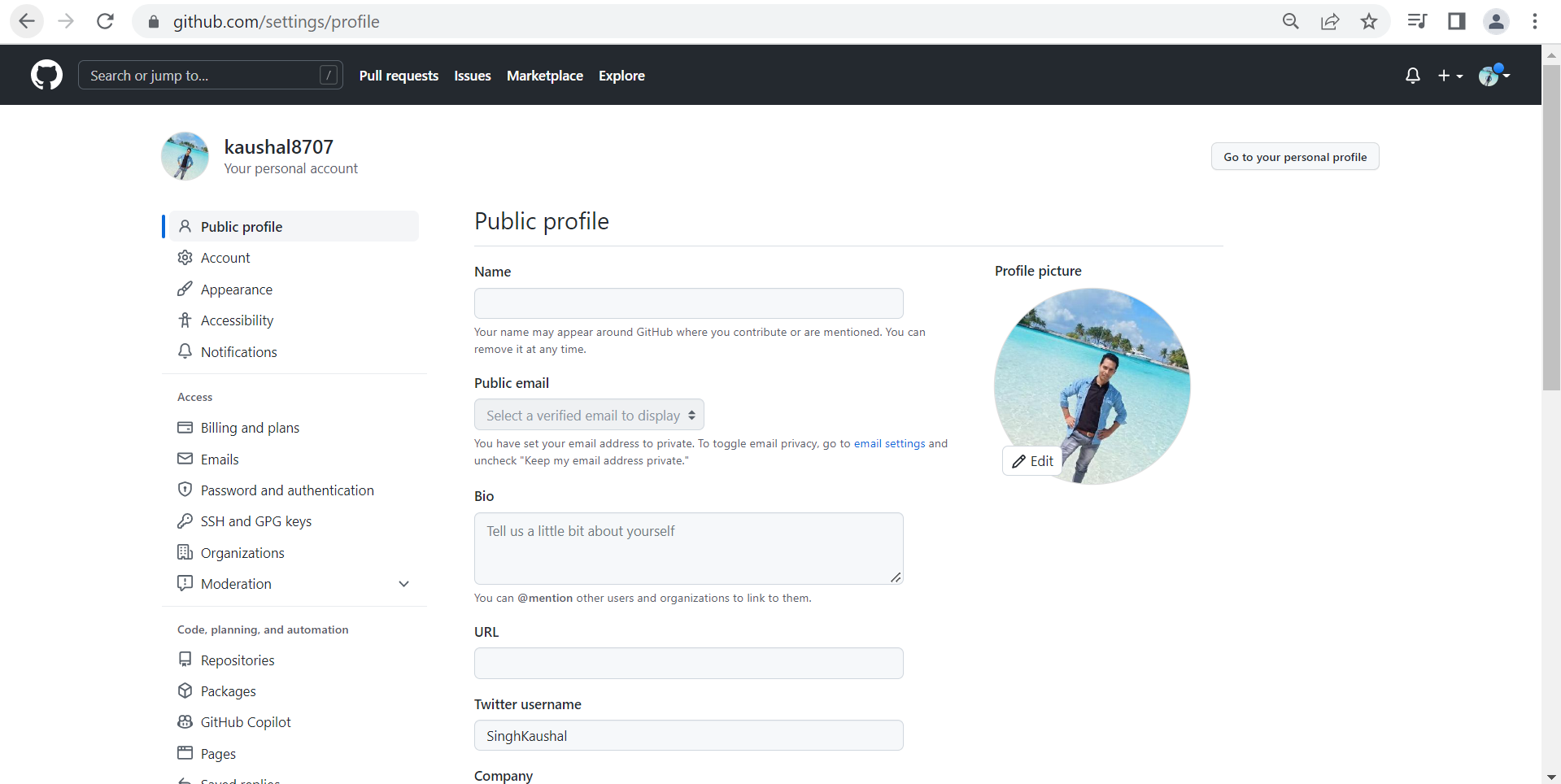
Now it is asking me to give Authorize to HackerRank to get your fields. This is where I am giving permission to Hacker Rank go to my Github and get my Username. this is what all about OAuth2.

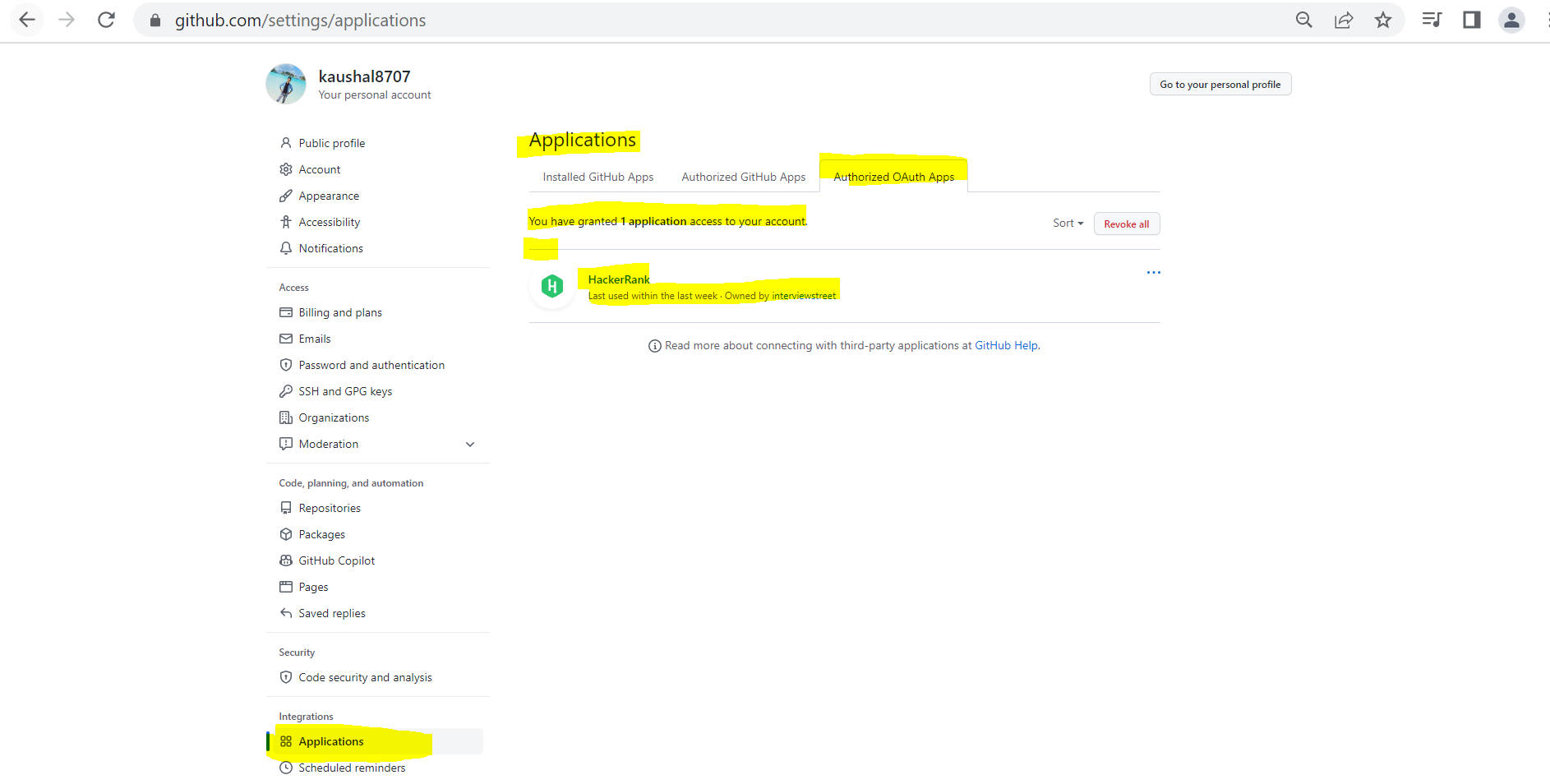
From Hacker Rank I delegate my access request, or I provide grant permission to Hacker Rank to get the fields from git hub. So, if I will click on **Authorize,** then directly it will re-direct me to the login page.



So, here I didn’t provide any input then also I got logged in. so here I give permission to Hacker Rank to access my Github user details.

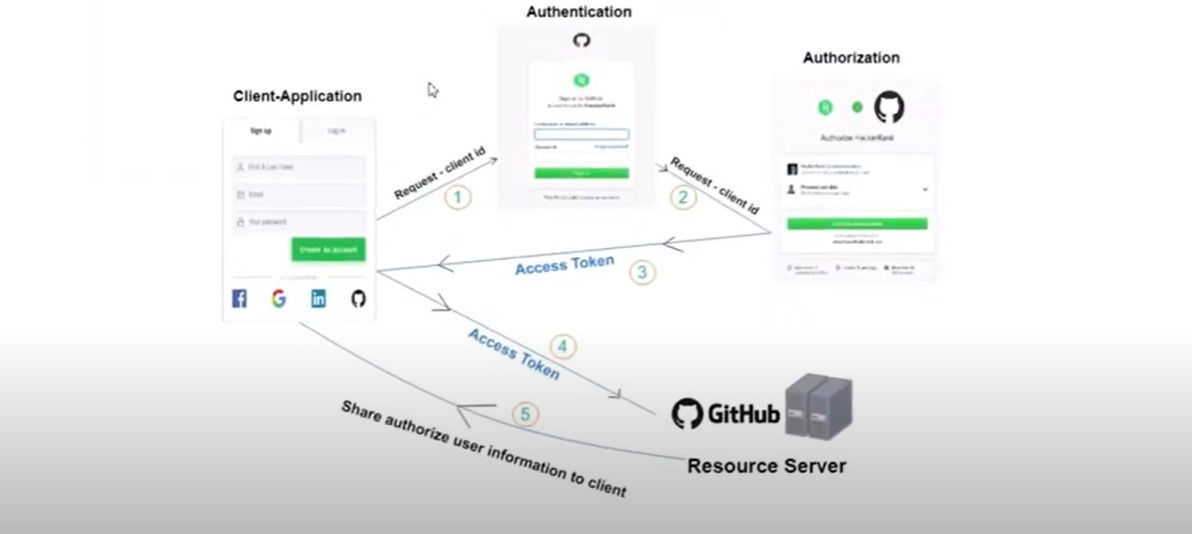
So, if I will go to my Github Setting Page and then Application, we ll get one tab Authorize OAuth2 Application.



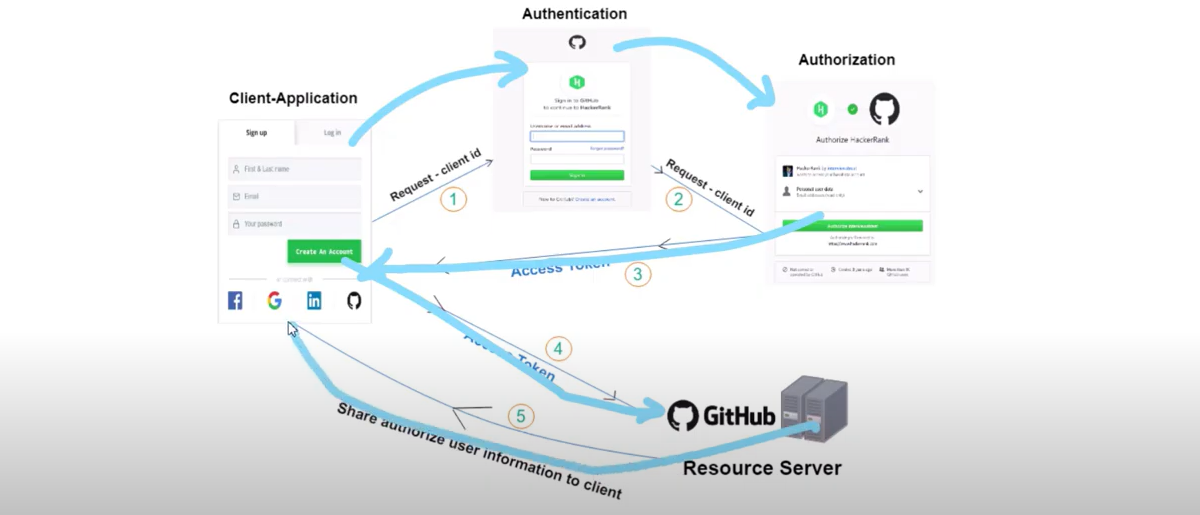


Its mentioned, you have granted 1 application access to your account. So, this is what I provided grant permission to Hacker Rank.

So now its time to understand how this OAuth2 flow is Internally working like how from hacker rank my request is come to Github and how they fetch our user details. So, for that I prepared one flow diagram let me show that….



So, once User choose as a Github for Signup purpose, one **Request** will go to a **Github Authentication Server** with the **Request – Client id,** then from **Authentication Server** request will delegate to the **Github Authorization server** with the same **Request-Client Id**. Once User **Authorized** It then Github will send one **access token** to the **Client**. And with same **access-token** Client will send a Request to a **Github Resource Server**, and this **Github Resource Server** will decrypt the access-token and then it will give **User Details** to the **Client** **Application**.



So the main role once the User Authorized it will send access token to the client, and same access token client need to send as part of Request Header so that Github Resource Server can decrypt that access token or validate that access token if that is the valid token then Github Resource Server will go to the Resource Server and get the User Details and returned it back to the client.

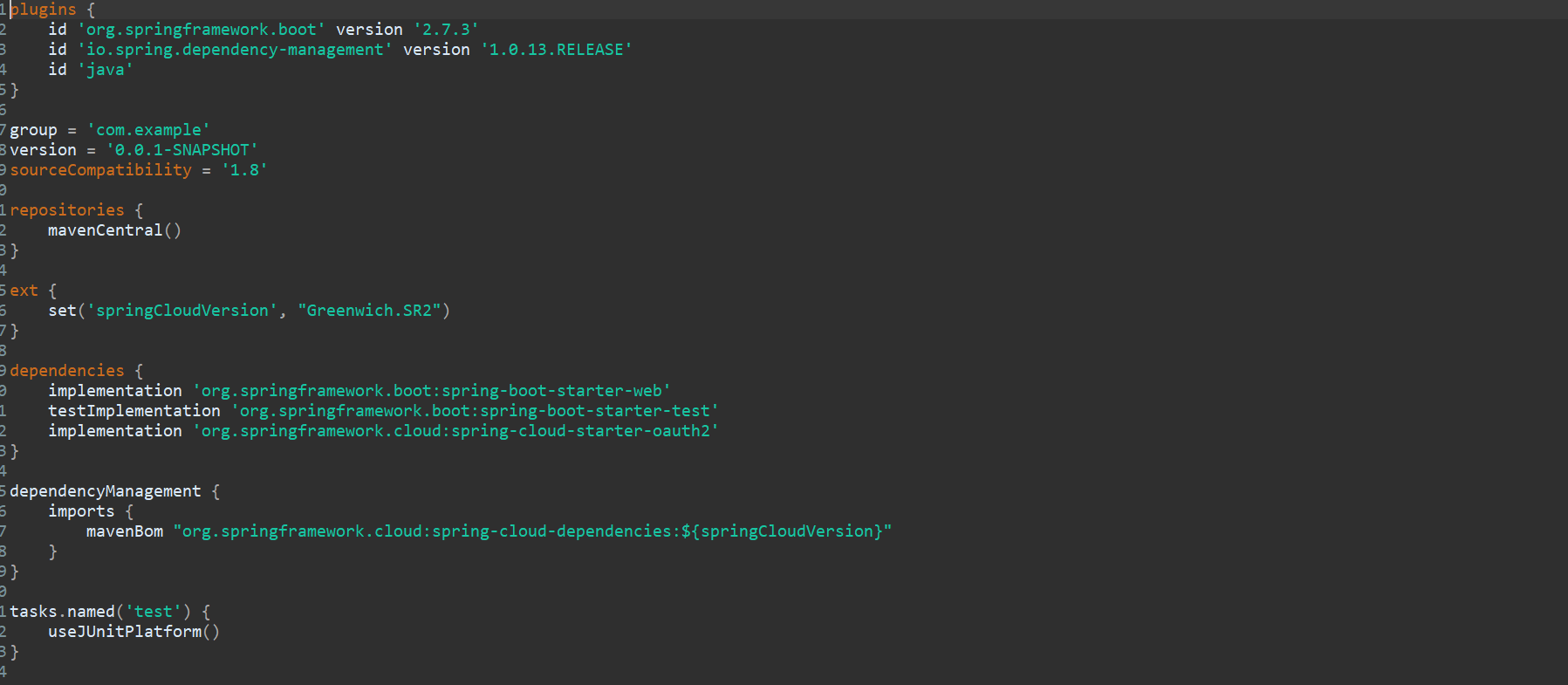
So, this is what Internal flow of Oauth2 from Client to Github Authorization Server.

Let’s build same kind of application we are going to create one client application, and from application we will delegate to our Authentication and Authorization request to this Github Server.

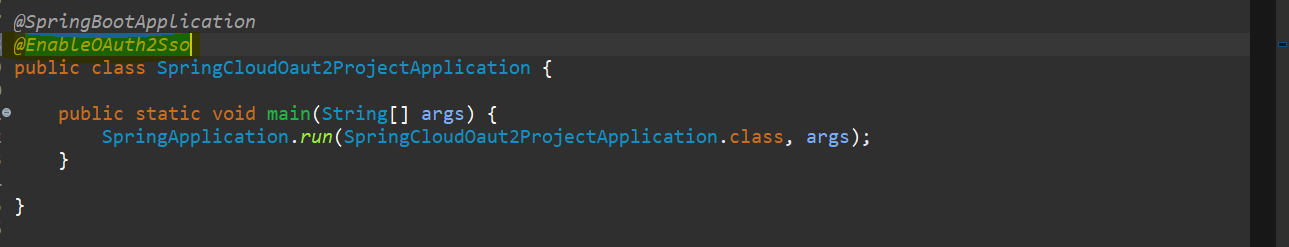


**Application**: **spring-cloud-oaut2-example**

**Dependencies**: **Spring Web, Spring Security, Spring Cloud Oauth2**



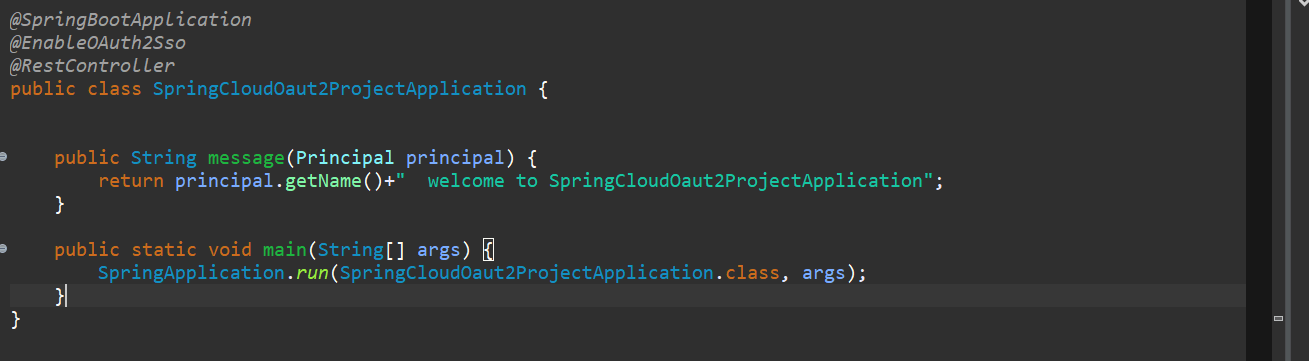
Now go to main class as you want to enable cloud oauth2. So, we need to add one annotation oauth2 with Single Sign On.



So, to enable oauth2 in our application we need to add annotation oauth2 Single Sign On.

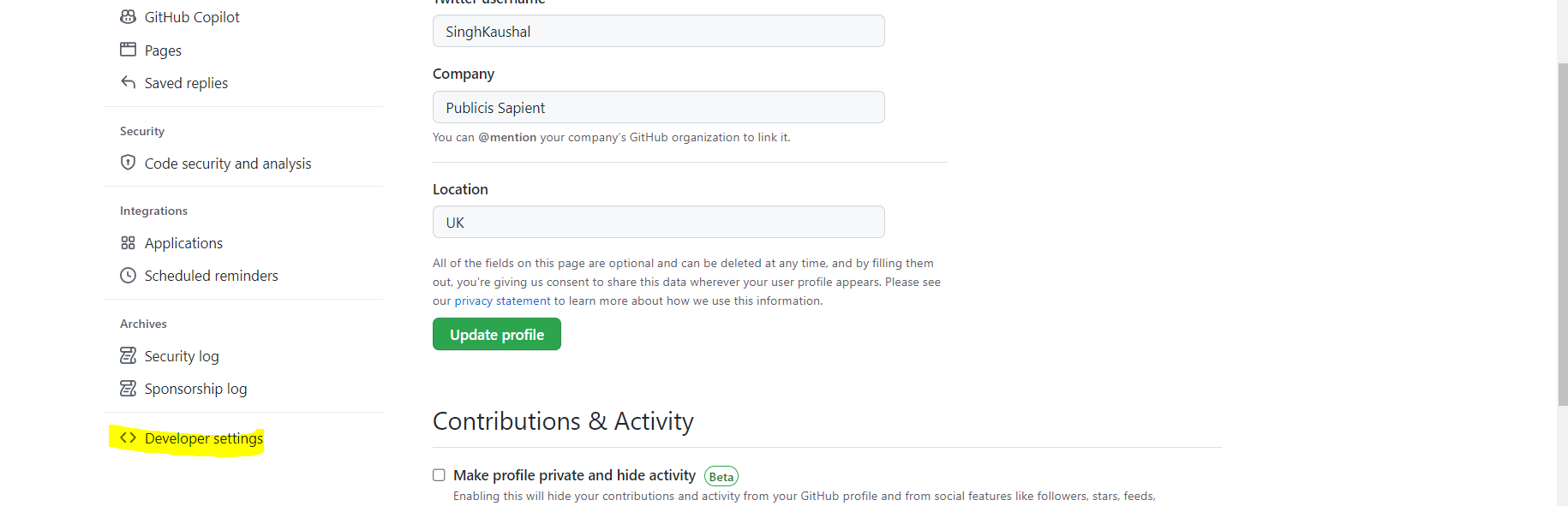
Here I want to expose one rest endpoint so I can confirm when user send the request to my application whether it is redirecting to the GitHub server or not.

So here I am going to add the Principal Object so that I can check the Username. And use GetMapping with default URL.

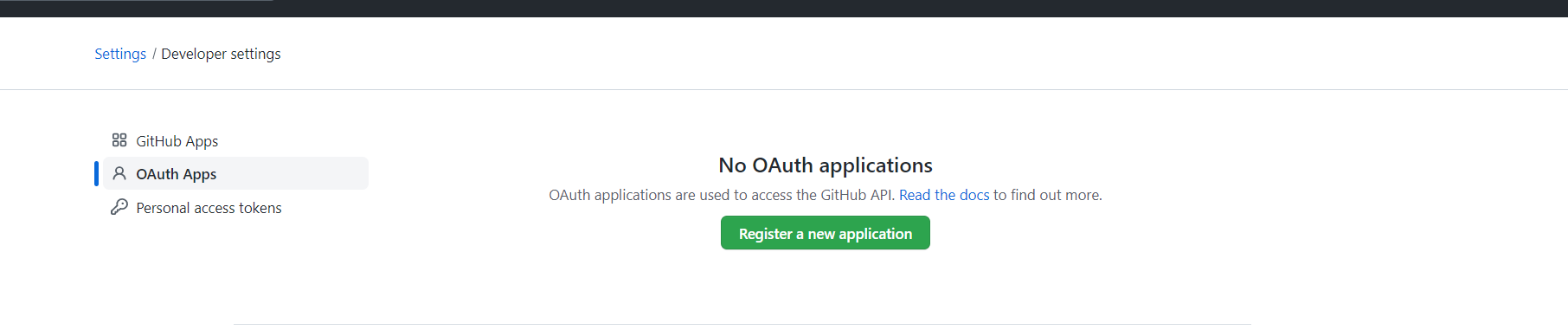


So, we created Client Side Application and we enable OAuth2.

Now we need to sink with Github. So, to sink with Github what we need to do. go to GitHub and try to find Developer Setting.



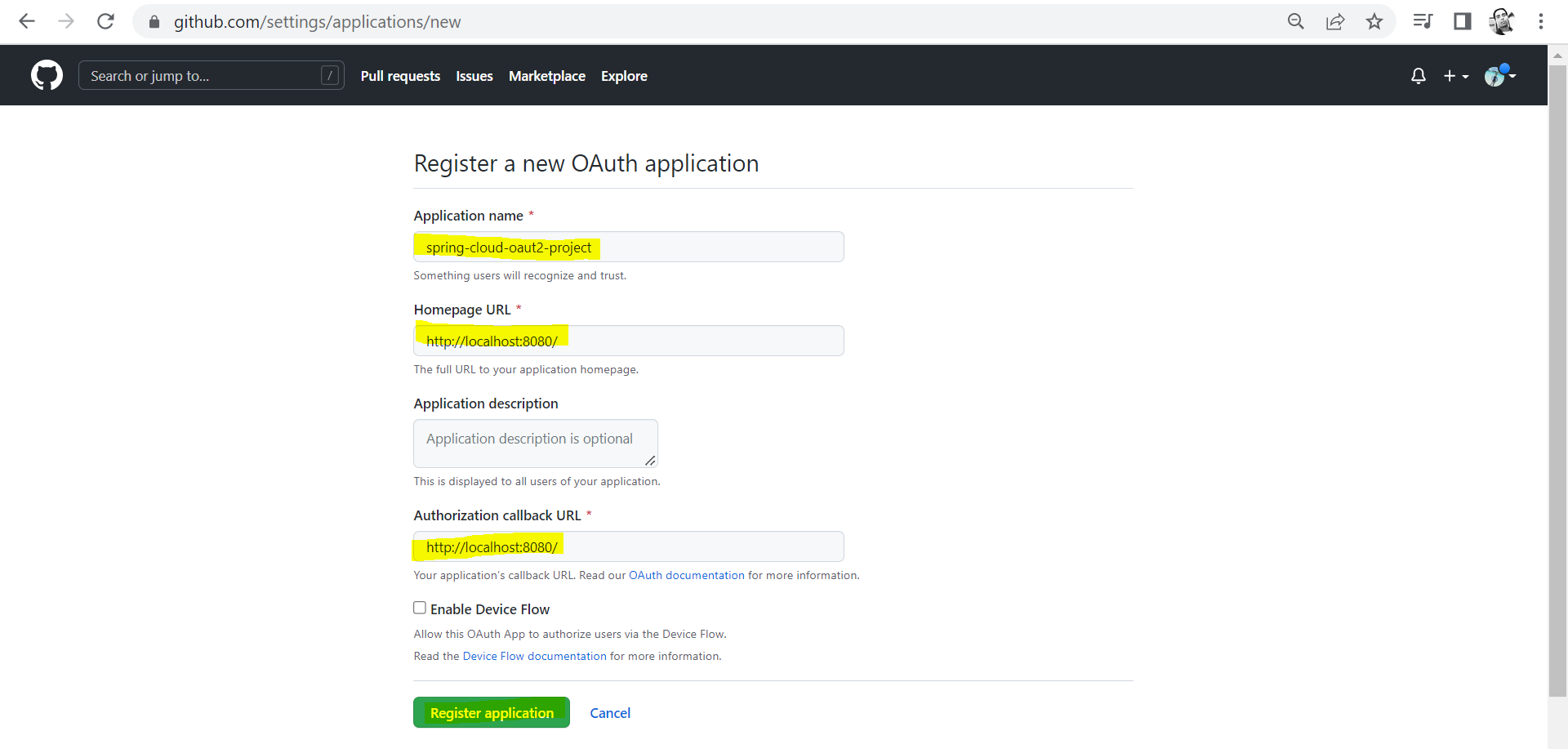
Then go to the oaut2 apps…



**Here you need to create an application in Github so that you can get the client-id and secret-id. So that from our application based on that client -id we can communicate to the Github server.**

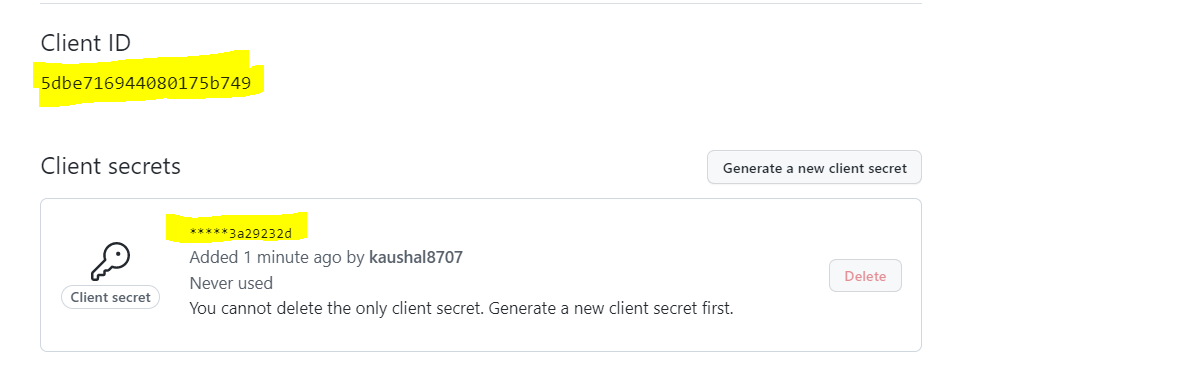
now let me give the application name… <http://localhost:8080/>

this is what the default url I am giving here. And then need to set the callback url I am giving the same url here. <http://localhost:8080/> then Register Application.



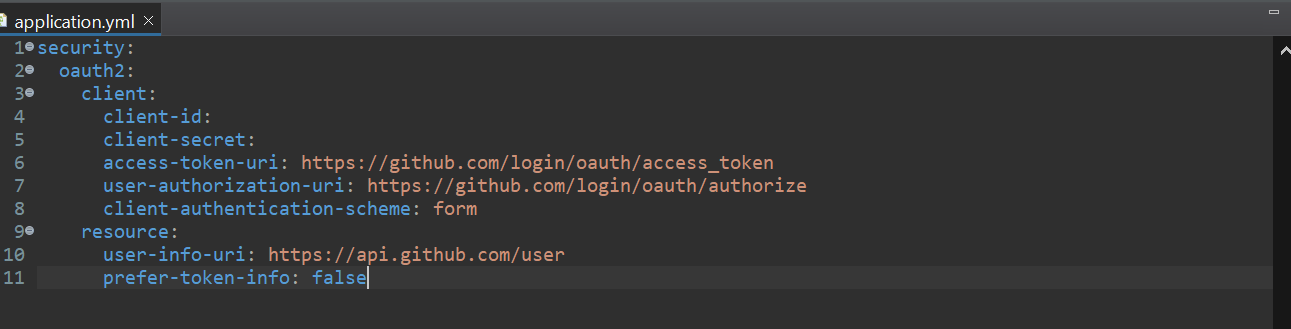
If you observed, we get here the client-id and client-secret ….

Client-secret: e06d1ffb49a2c967a1314b19548e6fee3a29232d

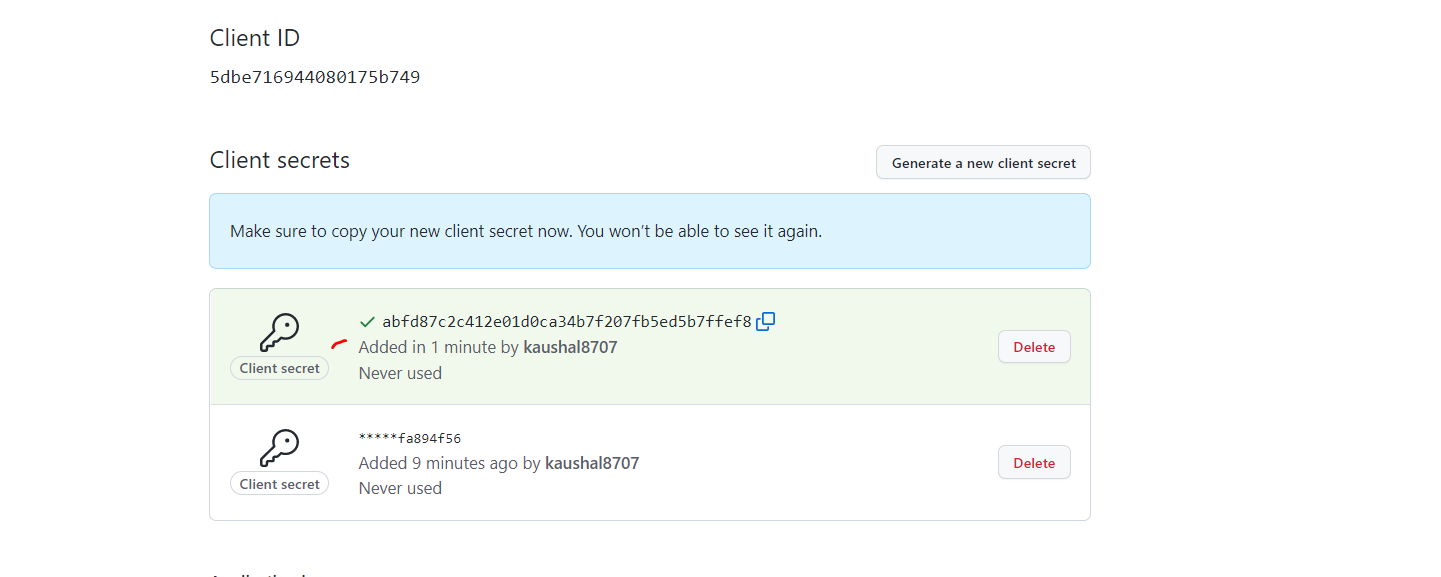


So, with this 2 fields from our application we can communicate to the application.

Client-id and client-secret we need to add in our code so let me add yml file.



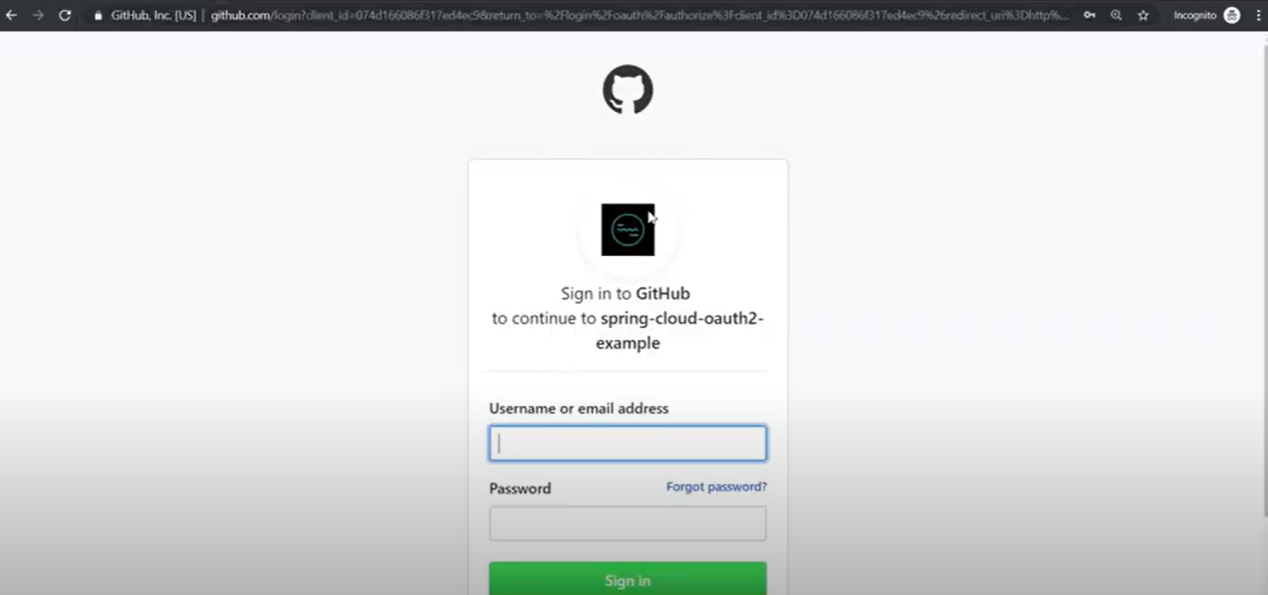
So, we added Client Id and Client Secret then access token uri, then user authorization uri then user info uri. So here we need to paste our client id and client secret.



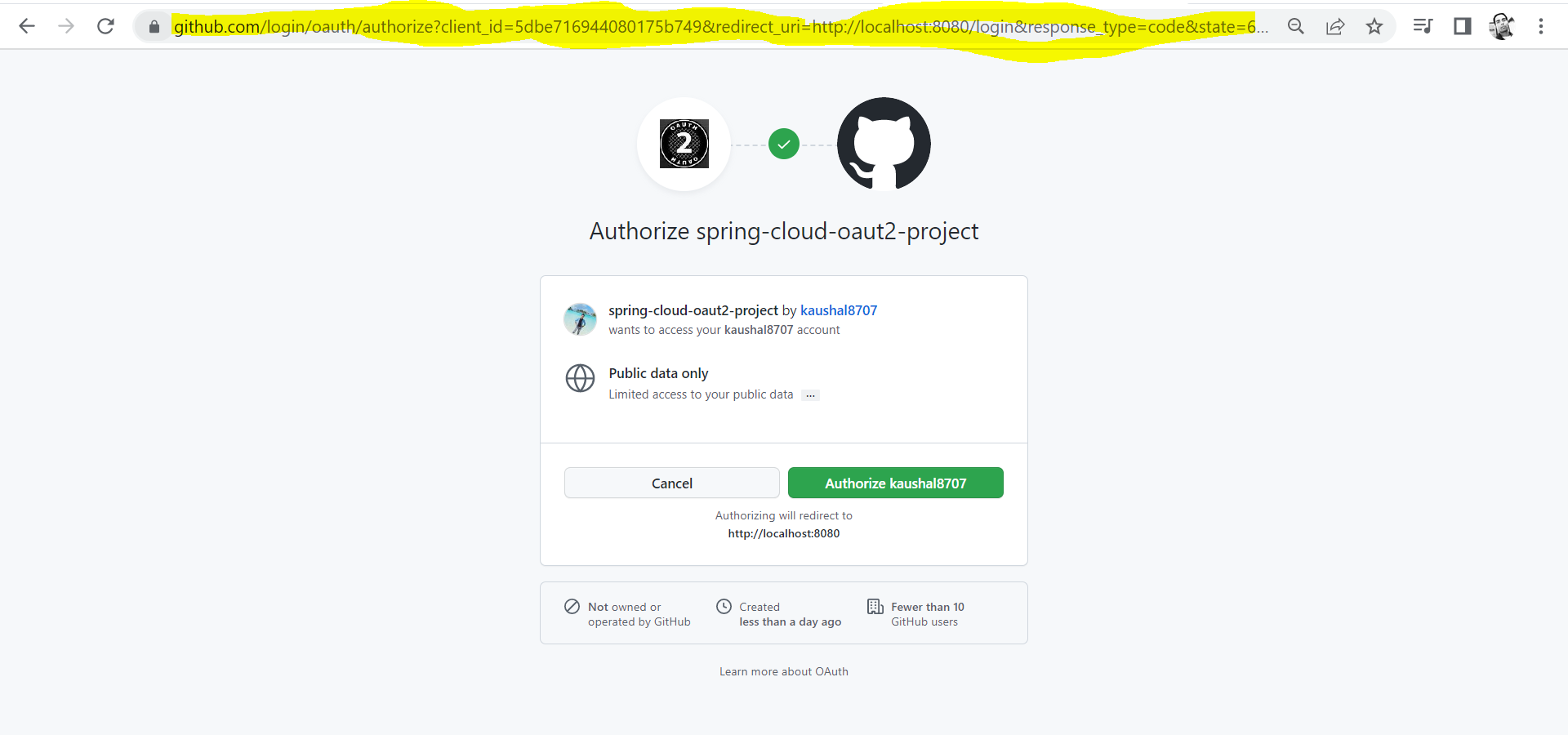
So now with this configuration we can sink up with our Github.

Let’s run our application…

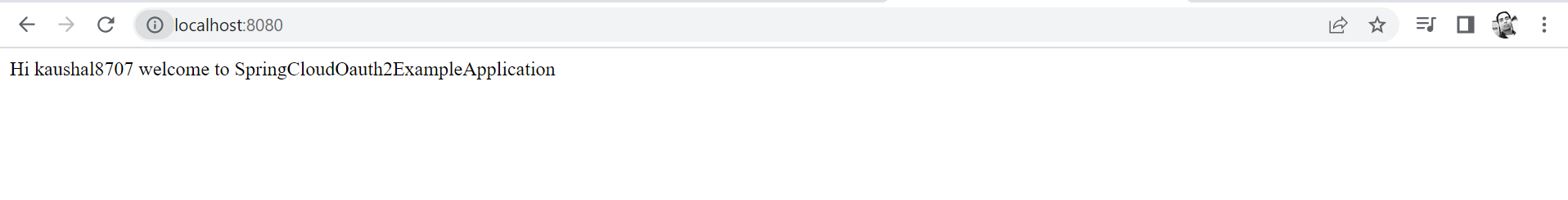
Let’s go to browser type **localhost:8080**



So, if u observed here from our application oauth2-spring-example it is delegating request to Github for authentication…so let me give my credentials here….so once we click on Sign In…so again I am getting Authorize my application to Github…Authorize spring-cloud-oauth2-project so that they can access your Github details…



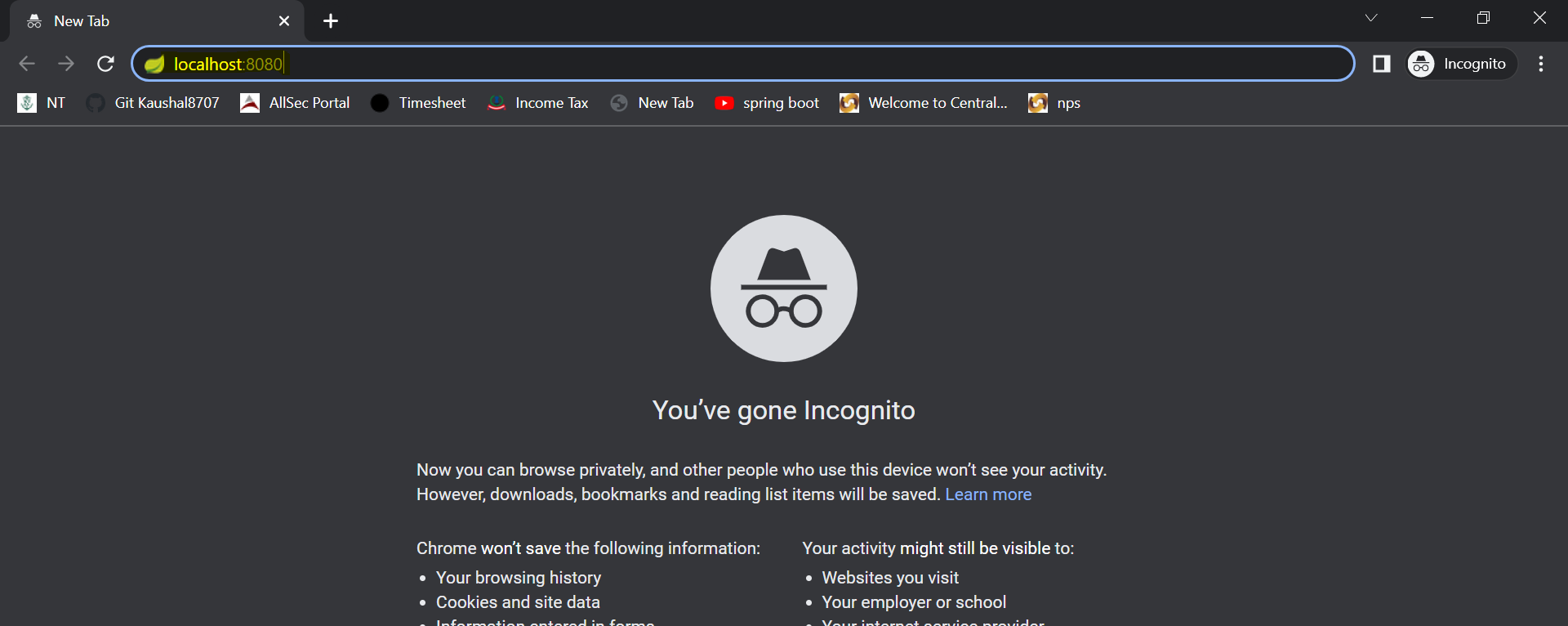
So, once I clicked on Authorized, I can see the output here.



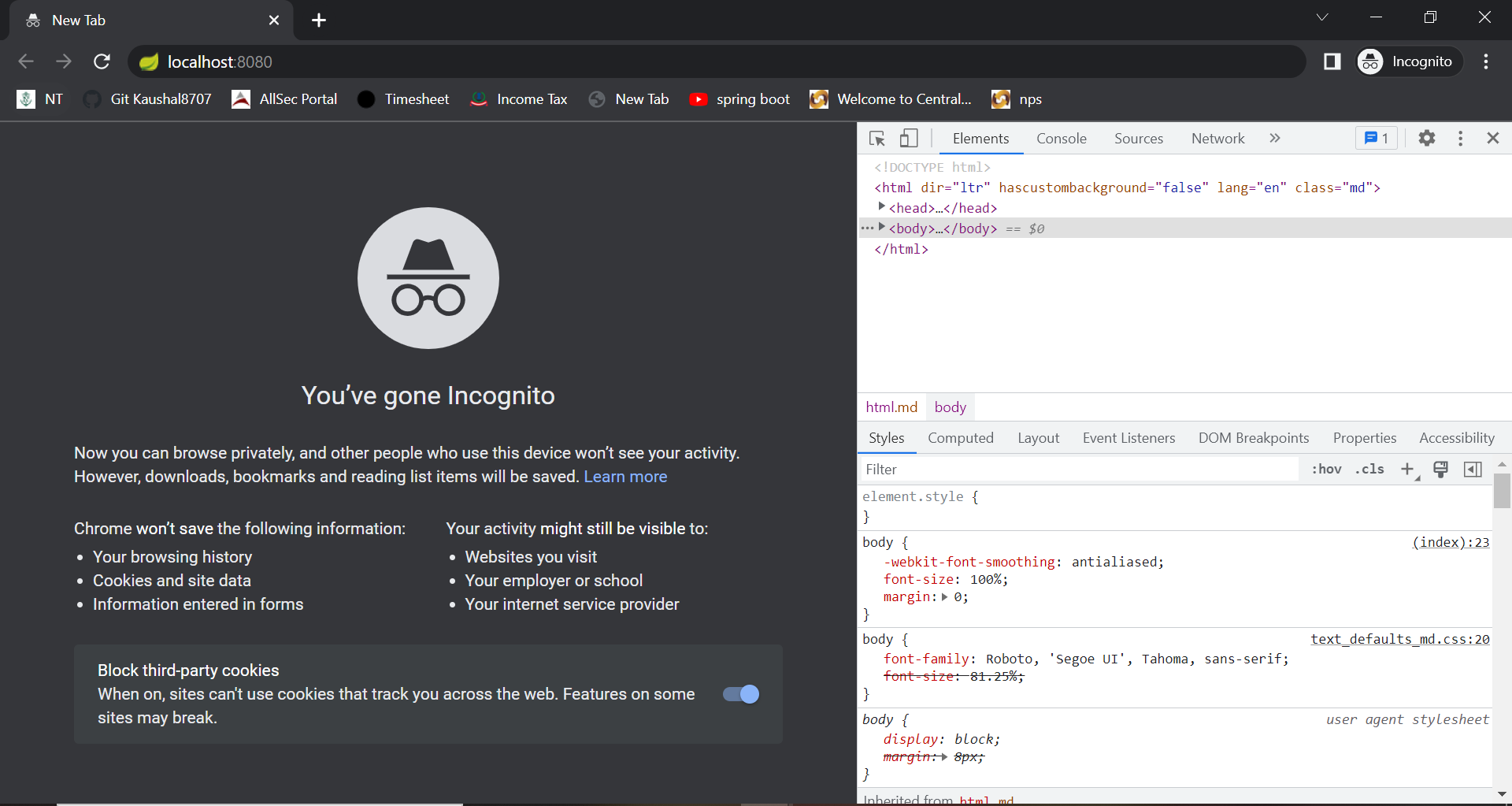
So kaushal8707 is the username I have set in Github account.

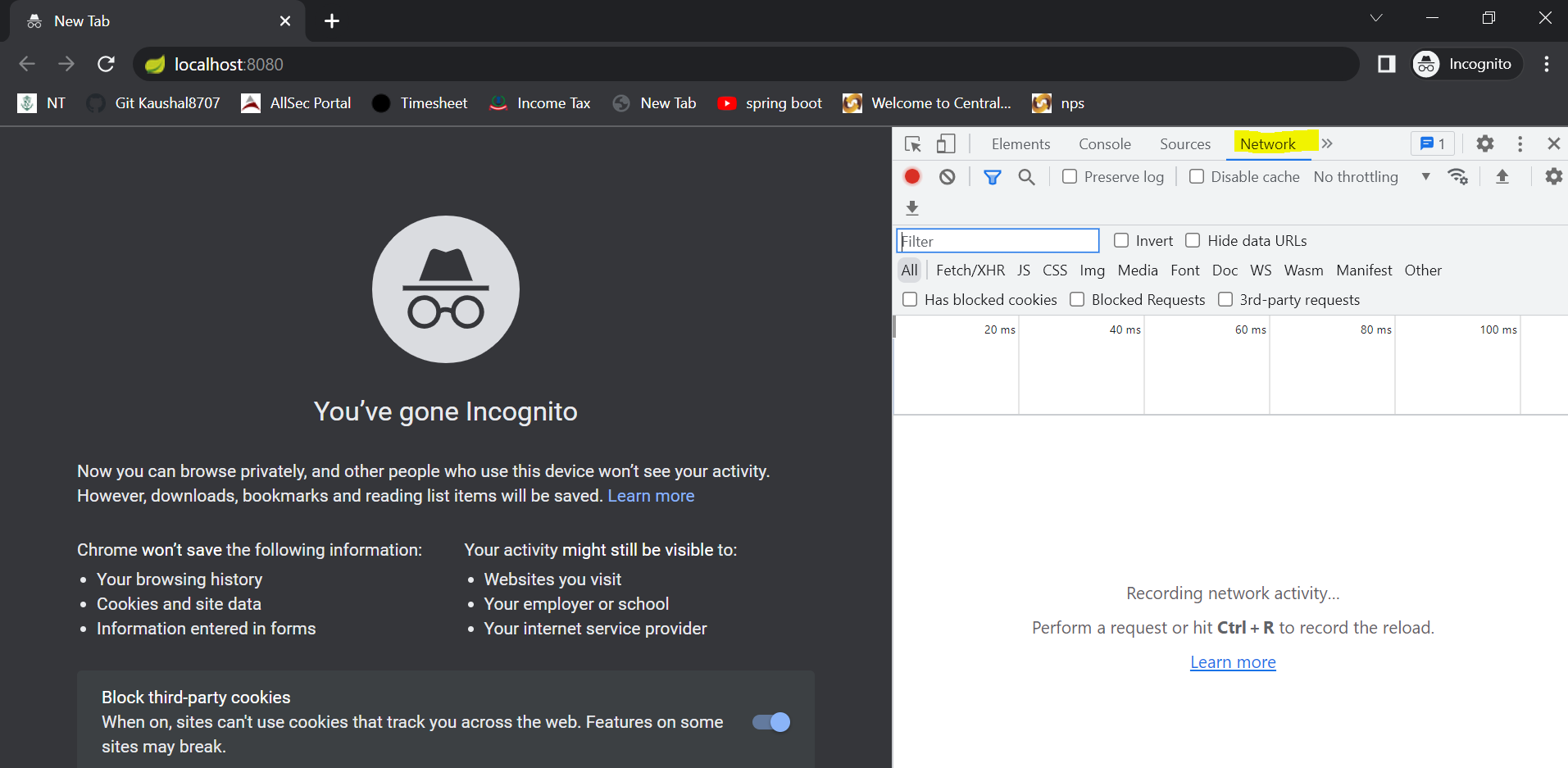
So, let me show you the same example by inspect this page so that i can show you how the access token sends by Github…

So, let me re-open if I will try the same url localhost:8080

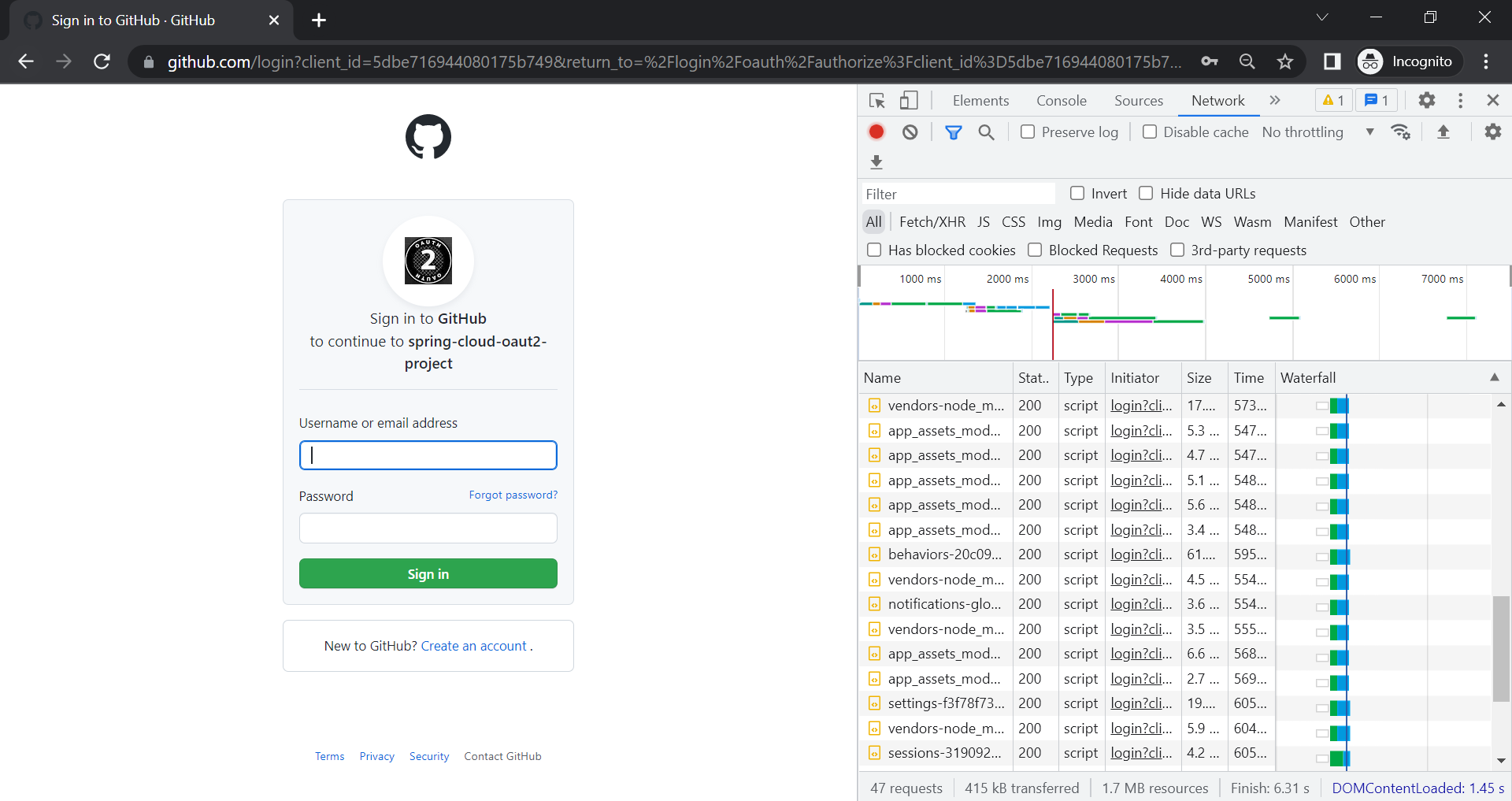


So before hit the rest endpoint let me inspect this



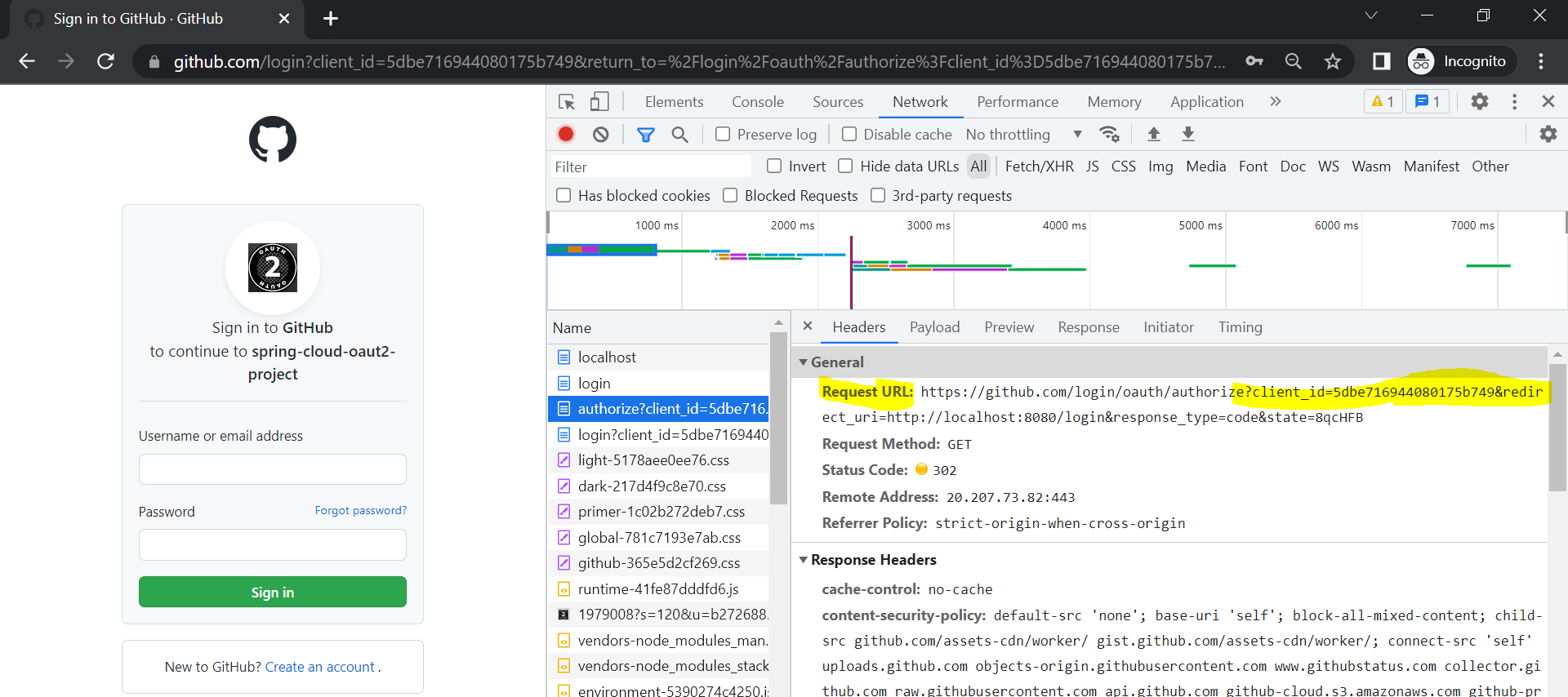


Now if we hit this….

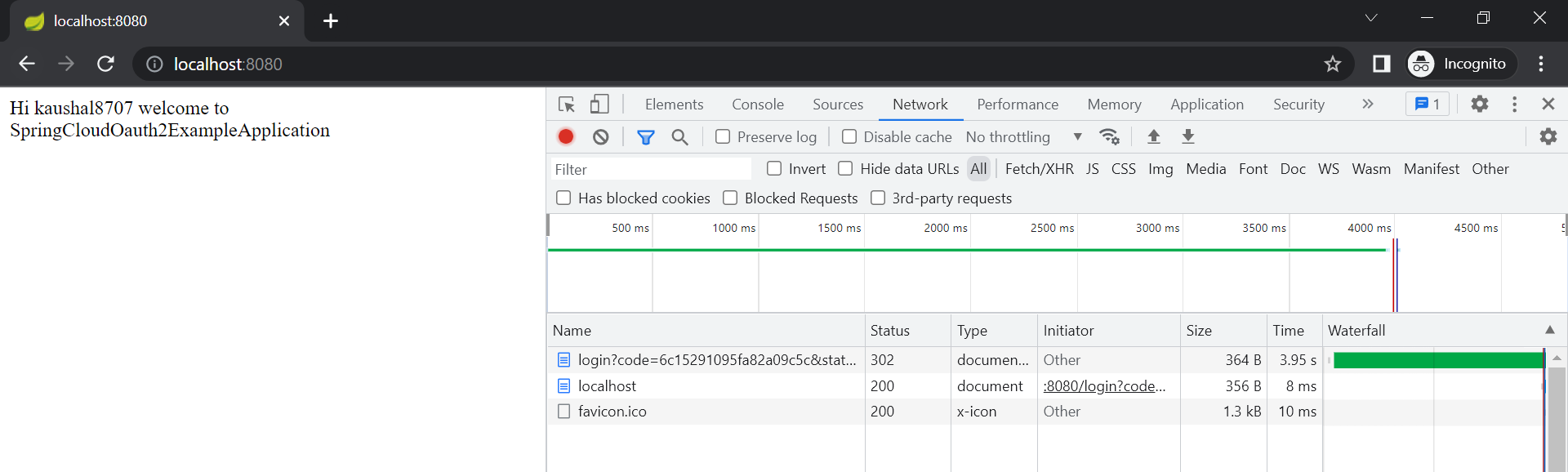


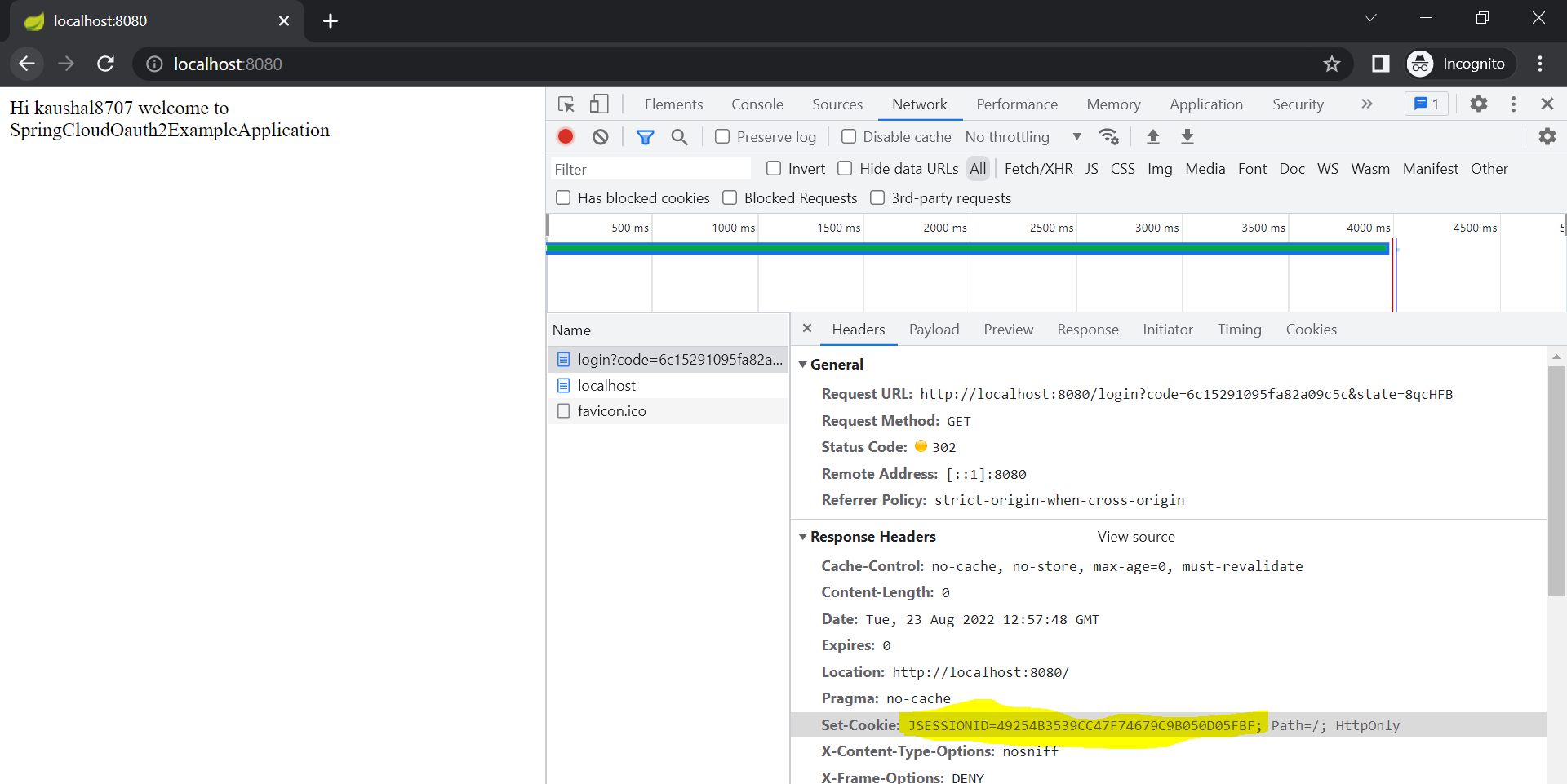
Again, we are getting this login credentials which is Authentication…

So, let me show you here… here Github is sending with the ClientID which we configured in our application.



So, let me login again...





So, we see in Response header **JSessionId**. So, this is what all about **OAuth2** Internally works.

**Summary**: -

**How OAuth2 (Authorize Request Delegation) Works Internally??**

1. User choose Sign up using GitHub.
2. User need to login to GitHub for Authentication.
3. Page will Redirect for GitHub Authorization.
4. Once User Authorized it, then GitHub will generate one access token and return back to the client.
5. Client will send request with same access token provided by GitHub.
6. GitHub Resource Server will decode that token and validate it.
7. If it is valid token, then client can receive user information from GitHub Resource Server.