**NUXT PINIA and LARAVEL SANCTUM**

We will be using Yarn instead of Npm, as it is a little tricker to get Pinia with Nuxt in npm. We can try with Npm as well, but now we will use yarn here.

To install Yarn run the following command

npm install --global yarn

now create a Nuxt app.

npx nuxi@latest init <YOUR-APP-NAME>

now type following in cmd

yarn install

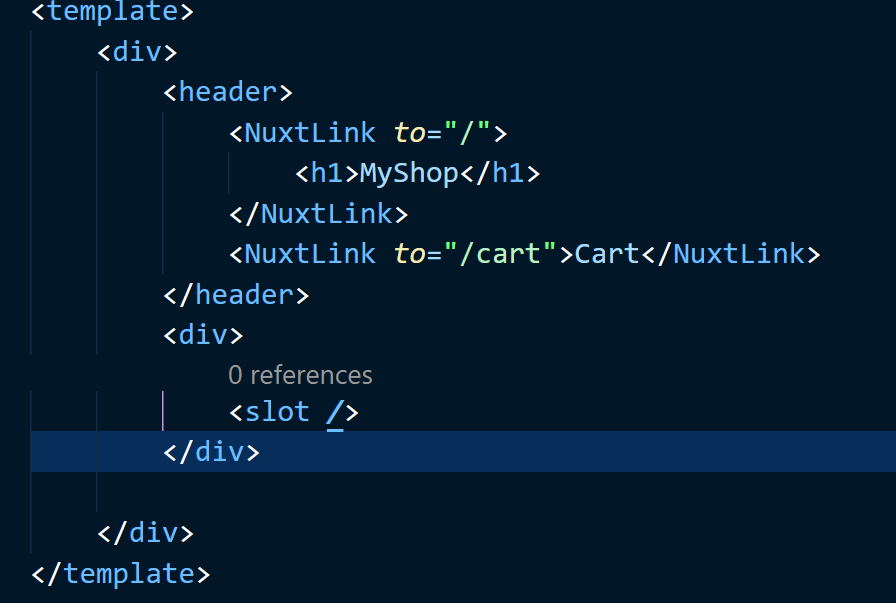
now in project directory create a folder named pages

now create a file in pages directory called index.vue, the pages/index.vue file will be our home page now. Also delete the app.vue in root directory.

Create a new file pages/cart. Vue, where you will add all the items in a basket selected by user.

Create a new folder in root directory called layouts. And create a file layout/default.vue. in this file we will create a default layout of the website.

Now type the following code inside default.vue file



NOW, type yarn dev -- -o in cmd. The -- -o will open browser window automatically

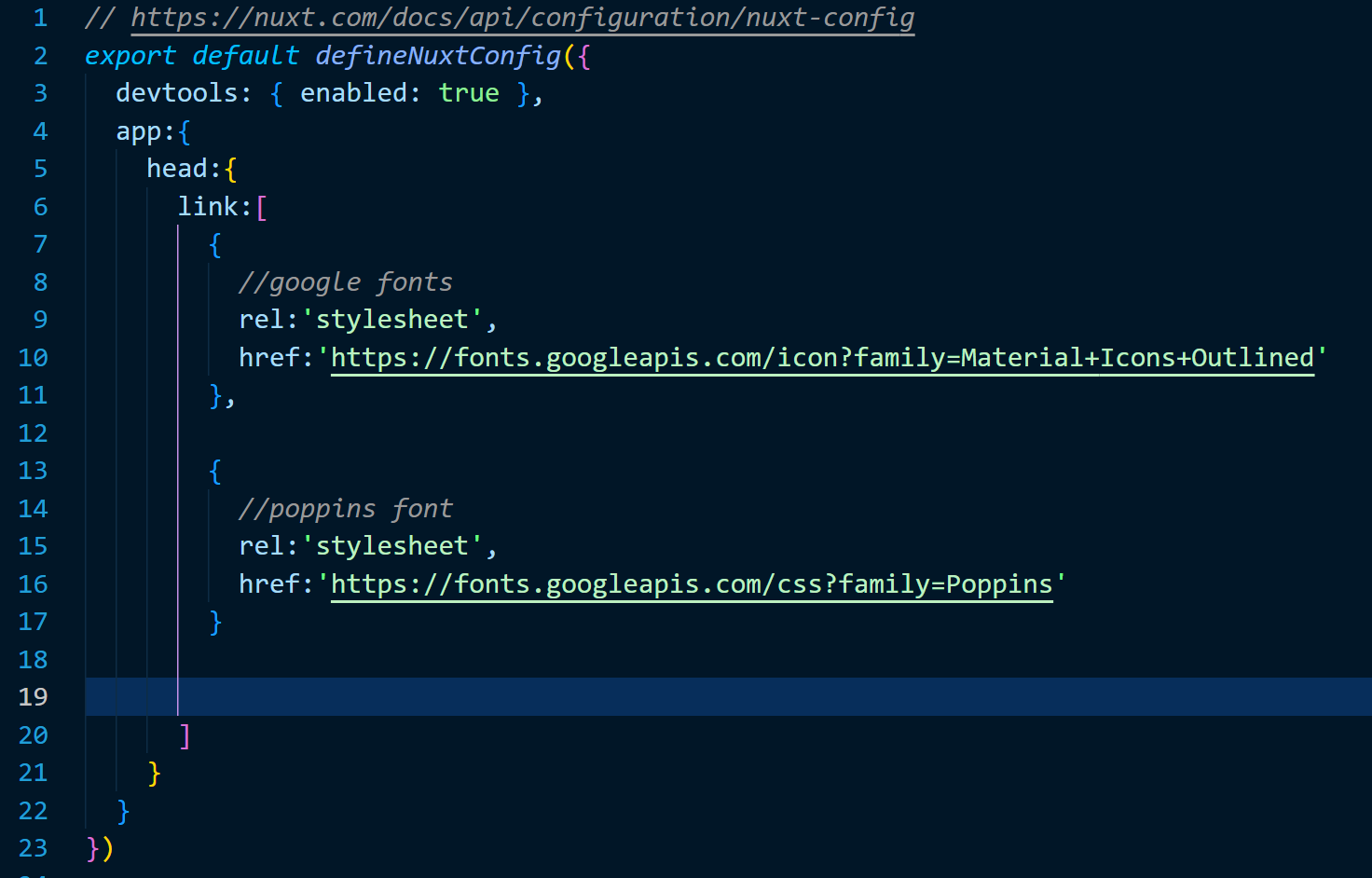
**Adding Google Material Icons**

Go to google material icons website

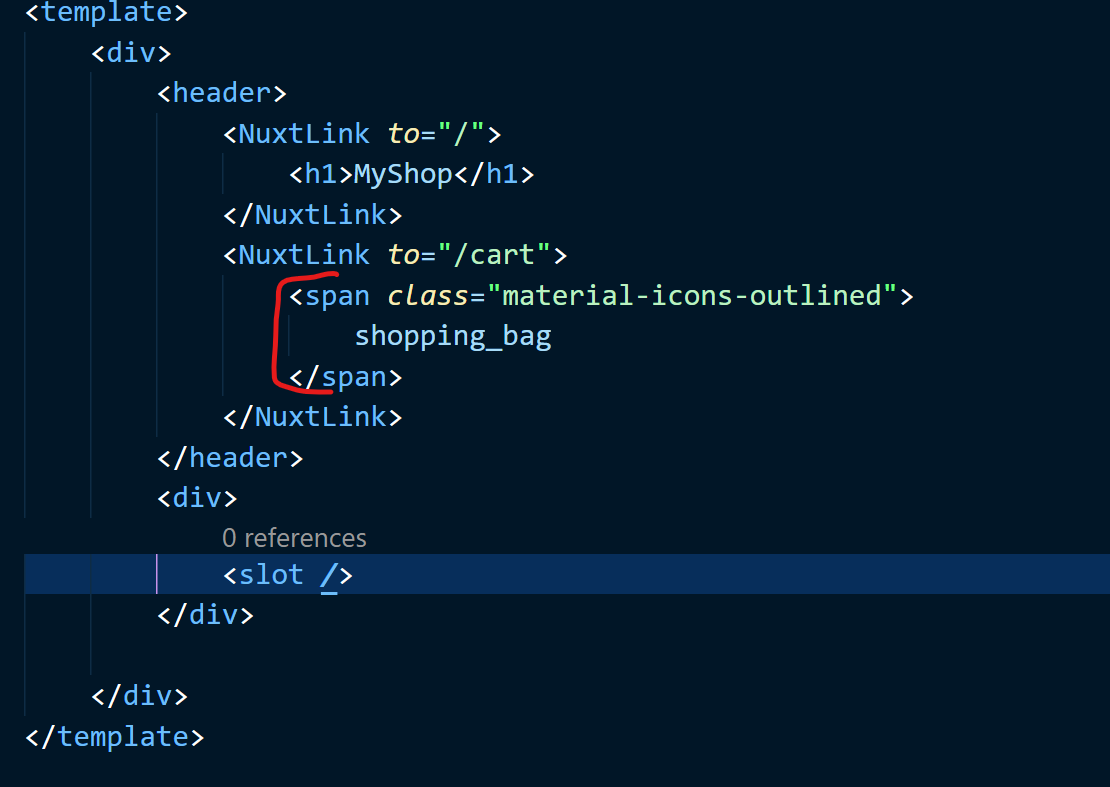
<https://fonts.google.com/icons?selected=Material+Symbols+Outlined:shopping_bag:FILL@0;wght@400;GRAD@0;opsz@48&icon.query=cart>

to add Material icons to Nuxt project.

Go to nuxt.config.ts file



Now go to layout/default.vue file and add google fonts in template



**Adding Tailwind CSS**

To extend tailwind CSS we create a new folder in root directory called assets, and create a folder inside it called CSS, and inside assets/CSS folder create a file called tailwind.css. Nuxt will automatically look for this file (assets/CSS/tailwind.css) and apply any additional classes or styles that we implement in assets/CSS/tailwind.css file

To get Tailwind CSS in Nuxt application run the following command inside cmd

yarn add -D @nuxtjs/tailwindcss

now in assets/CSS/tailwind.css file copy and paste the following.

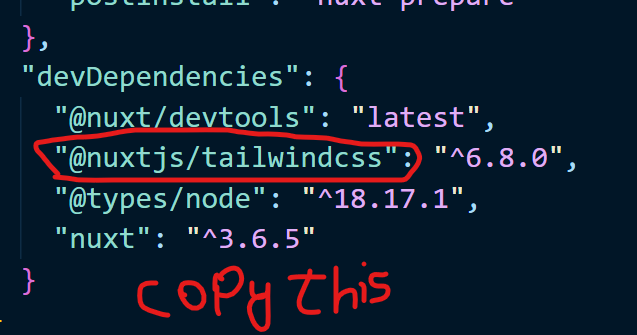
@tailwind base;

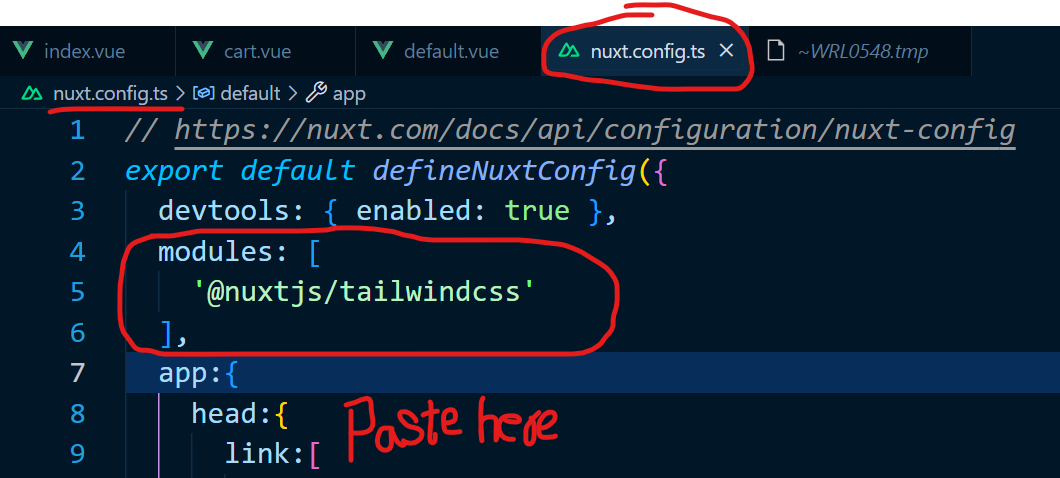
@tailwind components;

@tailwind utilities;

and after that go to nuxt.config.ts file and register tailwind CSS as a module.

To register the module, we go to package. Json file and find the nuxt/tailwind module, copy it and paste it in nuxt.config.ts ->modules



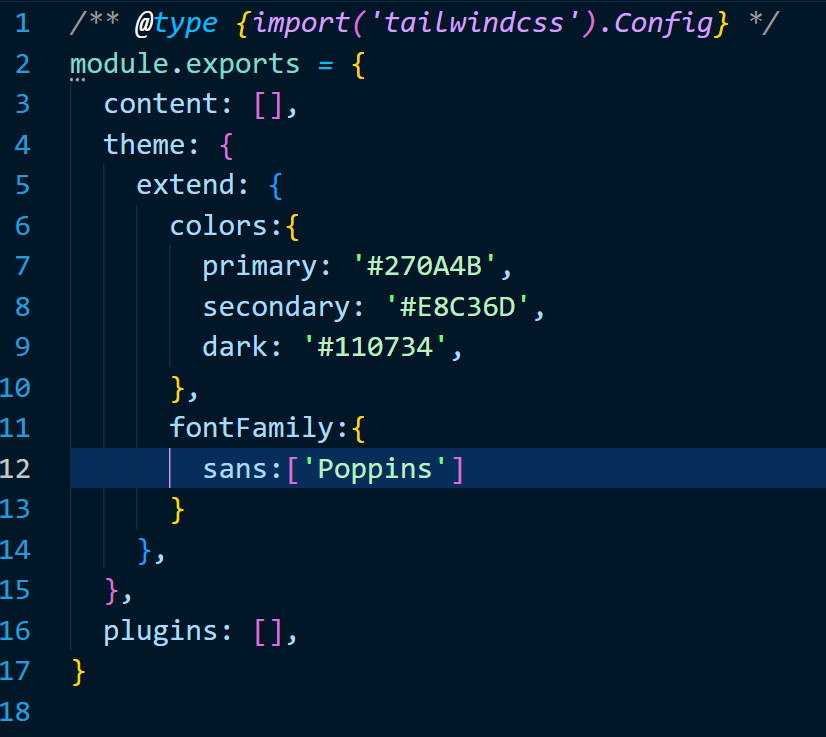


Now the module is registered. 👍

Now, we need tailwind.config.js file in our project, in tailwind.config.js file we can do things like extend the theme, add new colours and fonts etc. to get this file we type the following command in cmd

npx tailwindcss init

Now in tailwind.config.js file, we want to extend the theme, so we write the following code:



Now we can go to our layouts/default.vue file and make use of some tailwind classes made in tailwind.config.js file (primary, secondary and dark colour classes)

**Adding Pinia**

Type the following command to add pinia to Nuxt application

**yarn add Pinia @pinia/nuxt**

now, add Pinia to modules in your nuxt.config.js file

export default defineNuxtConfig ({

// ... other options

modules: [

// ...

'@pinia/nuxt',

],

})

**Adding form kit**

<https://nuxt.com/modules/formkit>

use the following command to get form kit

yarn add @formkit/nuxt

yarn add @formkit/themes

yarn add @formkit/tailwindcss (for styling forms)-> not using in this project

see this tutorial for reference : <https://www.youtube.com/watch?v=VegRSHUW04c>

git repository: <https://github.com/geekyshow1/nuxtauthui/tree/main>

**SETTING UP LARAVEL SANCTUM WITH NUXT 3**

<https://www.youtube.com/watch?v=2zKoS8GsKK8>**-> fixing common mistakes in sanctum setup**

<https://cdruc.com/laravel-spa-auth-extended> **-> fixing common mistakes in sanctum setup**

<https://www.youtube.com/watch?v=HLPoKz9j9KY>

<https://www.youtube.com/watch?v=NY9yoqoN72w>**-> using this**

**-------------------------------Laravel configuration of sanctum----------------------------**

1. Create a new Laravel project by running following commands

composer global require laravel/installer

laravel new YOUR-APP-NAME

1. After that you will be displayed with different options, select the following options here



Done -> open in vs code by typing Code.

1. Make sure to configure your .env file correctly as below

APP\_URL=http://localhost:8000 (//your back-end URL)

FRONTEND\_URL=http://localhost:3000 (//your front-end URL)

SESSION\_DOMAIN=localhost

SANCTUM\_STATEFUL\_DOMAINS=localhost:3000

DB\_CONNECTION=mysql

DB\_HOST=127.0.0.1

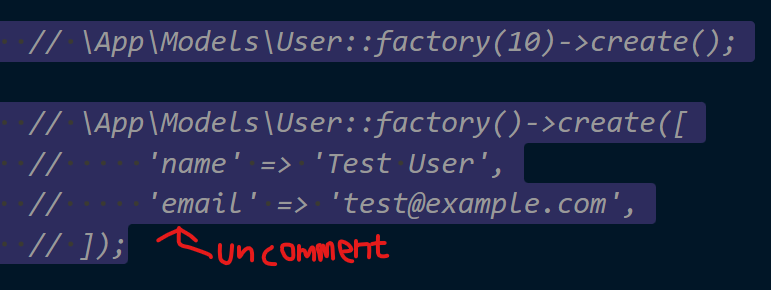
DB\_PORT=3306

DB\_DATABASE=laravelauthenticationdb

DB\_USERNAME=root

DB\_PASSWORD=

1. Now go to database/seeders/DatabaseSeeder.php file and uncomment the following code



1. Check if the cors.php file is configured correctly in config /cors.php. if you are using Laravel breeze, mostly it will be configured correctly already.
2. Now in terminal run the following command

php artisan migrate:fresh --seed

now we can use the test user, the password for the test user is password which we can get from database/factories/UserFactory.php file

1. Now run php artisan serve and come to Nuxt front-end now

**-------------------------------Nuxt configuration of sanctum----------------------------**

1. Create your Nuxt app by using following command

npx nuxi@latest init YOUR-APP-NAME

1. Now cd into the folder and type yarn install in cmd to install all dependencies
2. To run the server type npm yarn dev in cmd and open URL shown in cmd in browser
3. Work on your Authentication pages (login, register, forgot password, reset password, guest-only, auth-only) and style of theme

Follow the repository below to setup theme

git repository: <https://github.com/geekyshow1/nuxtauthui/tree/main>

video reference: <https://www.youtube.com/watch?v=VegRSHUW04c>

1. After completing the theme, create an Authstore in pinia.

import {defineStore} from "pinia";

export const SanctumAuth = defineStore ('Authentication', {

state:()=> ({}),

actions: {}

})

1. Now in login.vue file import the Auth store and call submitloginform function on click from form kit submit button.

<script setup>

import {SanctumAuth} from '@/stores/AuthStore'

const authStore = SanctumAuth ();

const submitLoginForm = async (formData) => {

authStore.loginsetup (formData);

};

</script>

1. The first thing that we need to do even before attempting to login is to get CSRF TOKEN, we do that by sending an initial request to <http://localhost:8000/sanctum/csrf-cookie>.

NOTE: - As Nuxt 3 does not have support for Axios wo, we will be using useFetch composable function.

1. Now to get CSRF TOKEN , go to Auth pinia store and create a function in action: {} object to fetch CSRF token from laravel backend as shown below:

actions: {

async loginsetup () {

await useFetch ('http://localhost:8000/sanctum/csrf-cookie',);

},

}

1. Now call loginsetup () function from login. Vue file using authStore Object

<script setup>

import {SanctumAuth} from '@/stores/AuthStore'

const authStore = SanctumAuth ();

const submitLoginForm = async (formData) => {

authStore.loginsetup();

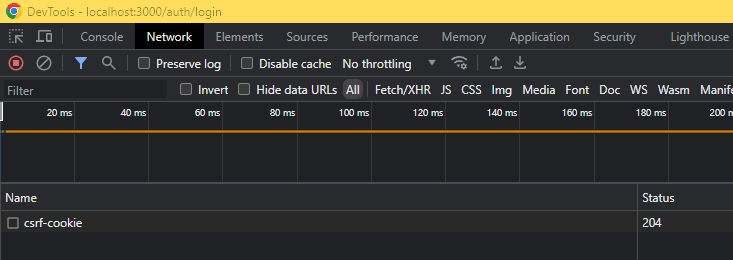
};

</script>

1. Now go to login page in browser, open up the dev tools by pressing F12 or right click->inspect element, then go to the network tab, type in the credentials

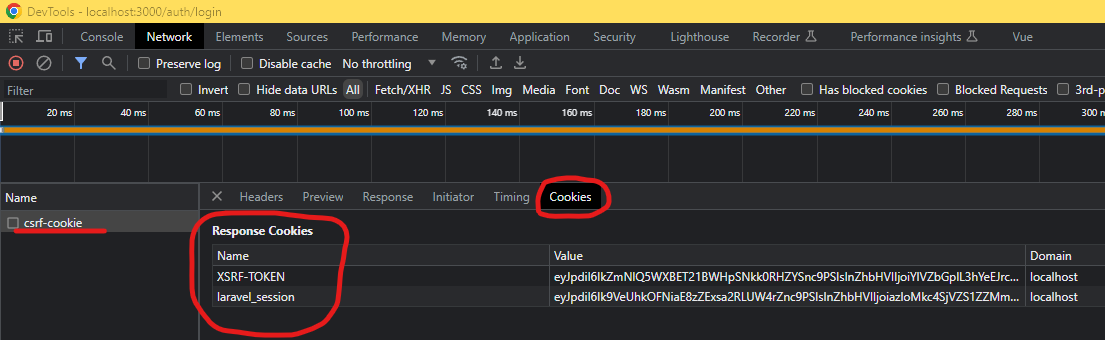
Email: [test@example.com](mailto:test@example.com) Password: password

And click on submit, you will see in network tab that we get a request named csrf-cookie.

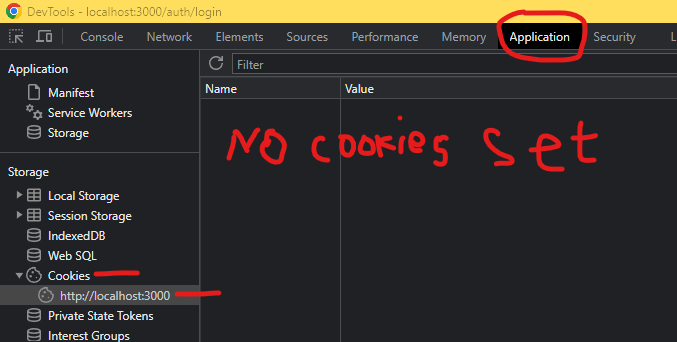


NOTE: The '/sanctum/csrf-cookie' route return a 204 response when successful.

Now click that request and go to cookies tab, and you will be able to see the response cookies there.



We received the cookies as response but if we go to application tab in dev tools then click on cookies on left side bar, then click on the URL displayed under it, we will notice that there are no cookies set



**Note: - these credentials come from laravel backend from database/seeders/DatabaseSeeder.php file and we get password from database/factories/UserFactory.php**

1. In order to exchange cookies with laravel api to store the cookies in the browser we need to modify the useFetch request for CSRF token. We need to add a new parameter to the request called {credentials: “include"}. We will add this parameter shown as below:

actions: {

async loginsetup () {

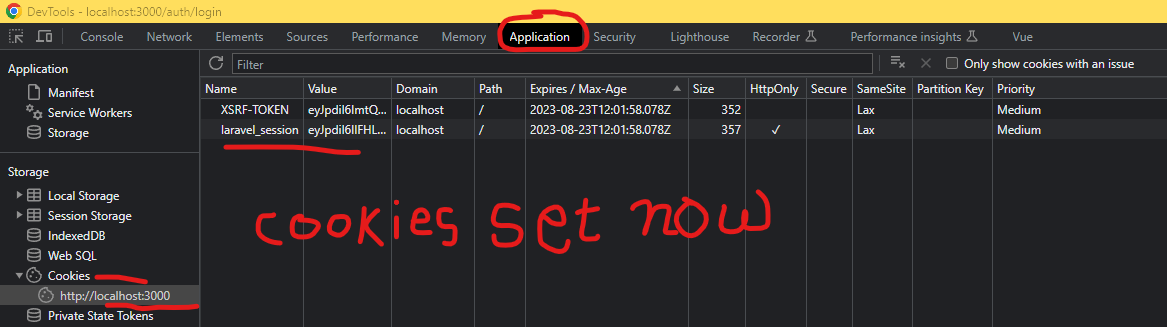
await useFetch ('http://localhost:8000/sanctum/csrf-cookie', {

credentials: “include"

});

},

}



1. Now as we successfully got our csrf token now let’s try to make a login request. To login the user, we will create a new function in Auth pinia store called loginUser () as shown below:

actions: {

async loginsetup () {

await useFetch ('http://localhost:8000/sanctum/csrf-cookie', {

credentials: “include"

});

this. loginUser(formData)

},

async loginUser (formData) {} //->login function

}

1. Now in login. Vue file we call this loginsetup () function using authStore object and pass the form data that we get from form kit on click submitLoginForm

<script setup>

import {SanctumAuth} from '@/stores/AuthStore'

const authStore = SanctumAuth ();

const submitLoginForm = async (formData) => {

authStore. loginsetup (formData);

};

</script>

1. Now inside async loginUser (formData) {} function in Auth Pinia store, we write a http request to login end point as shown below:

async loginUser (formData) {

await useFetch (' <http://localhost:8000/login> ’, {

credentials: “include",

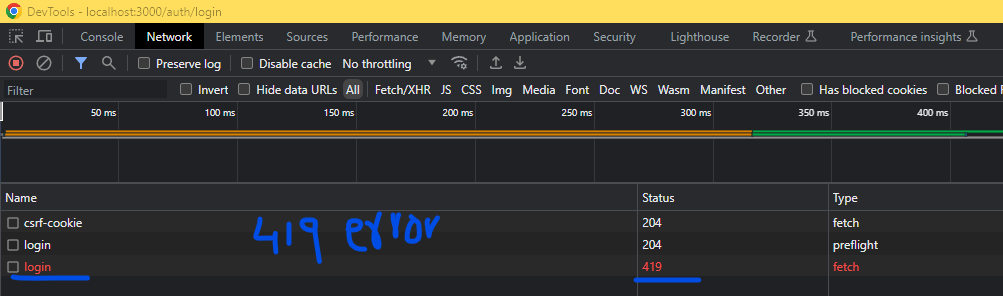
method: “POST",

body: formData,

});

}

1. Now if we go to network tab in browser dev tools, and click submit button we will get a 419 error. This means something is wrong with our csrf token.



1. We get 419 errors because:

During this request, Laravel will set an XSRF-TOKEN cookie containing the current CSRF token. This token should then be passed in an X-XSRF-TOKEN header on every future request, which some HTTP client libraries like Axios and the Angular HTTP Client will do automatically for you. If your JavaScript HTTP library does not set the value for you, you will need to manually set the X-XSRF-TOKEN header to match the value of the XSRF-TOKEN cookie that is set by this route.

So, in order to set X-XSRF-TOKEN in header, we write the following code in

async loginUser (formData) {} function

async loginUser(formData) {

//getting 'XSRF-TOKEN' cookie by using useCookie composable Nuxt function

this. token = useCookie('XSRF-TOKEN');

await useFetch (' <http://localhost:8000/login> ’, {

credentials: “include",

method: “POST",

body: formData,

//assigning the cookie token value, we got from useCookie composable to headers

headers: {

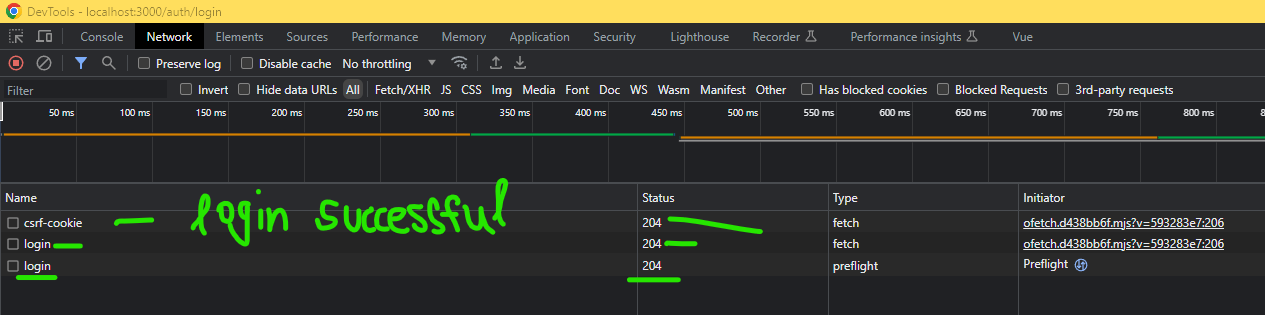
'X-XSRF-TOKEN’: this. token

}

});

}

Now if we go to developer tools-->network tab in browser, we will see our request for login was successful



async loginUser(formData) {

//getting 'XSRF-TOKEN' cookie by using useCookie composable Nuxt function

this. token = useCookie('XSRF-TOKEN');

await useFetch (' <http://localhost:8000/login> ’, {

credentials: “include",

method: “POST",

body: formData,

watch: false

//assigning the cookie token value, we got from useCookie composable to headers

headers: {

'X-XSRF-TOKEN’: this. token

}

});

this. getLoginUser ();

}

1. Now as the login was successful, we will need to grab the user data from the server, in order to do that we will create a new function called getLoginUser () and invoke this function at the end of loginUser function and add token as state as shown below

state:()=> ({

token:''

}),

async getLoginUser () {

const {data} = await useFetch (' <http://localhost:8000/login> ’, {

credentials: “include",

watch: false,

headers: {

'X-XSRF-TOKEN’: this. token

}

});

Console.log(data); // user data will be displayed on the console

}