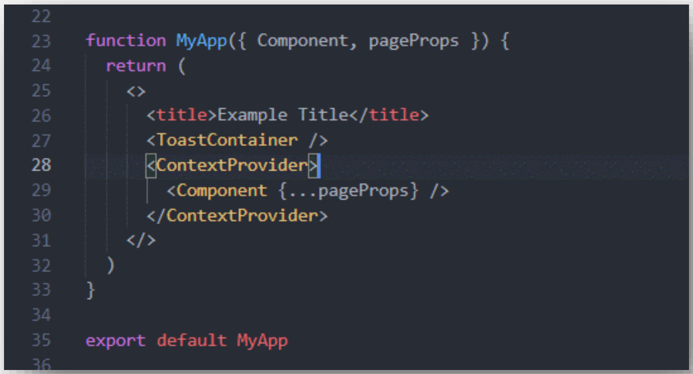
Next.js Sample source code

Welcome to the this next.js sample source code document I hope we could upgrade together for better world

➢ Guide pages:

❖ First, we want to talk about structure of the folders is this source:

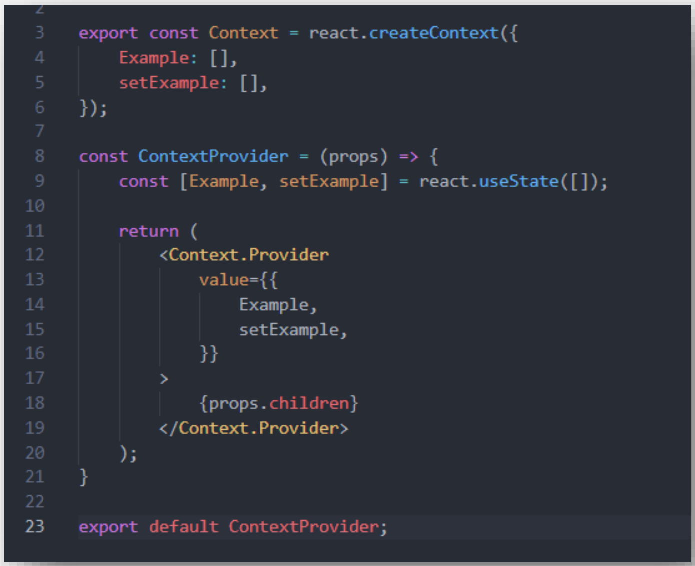
- In folder **pages > _app.js**: we have our context provider, title of website, and toastify



```
function MyApp({ Component, pageProps }) {
  return (
    <>
      <title>Example Title</title>
      <ToastContainer />
      <ContextProvider>
        <Component {...pageProps} />
      </ContextProvider>
    </>
  )
}

export default MyApp
```

modules that help us to show some errors or warning messages to the user

- The context provider imported from **> context > index.js**: in this file we have our global state that we created with **useState** and **passed** the data to the **value** of the



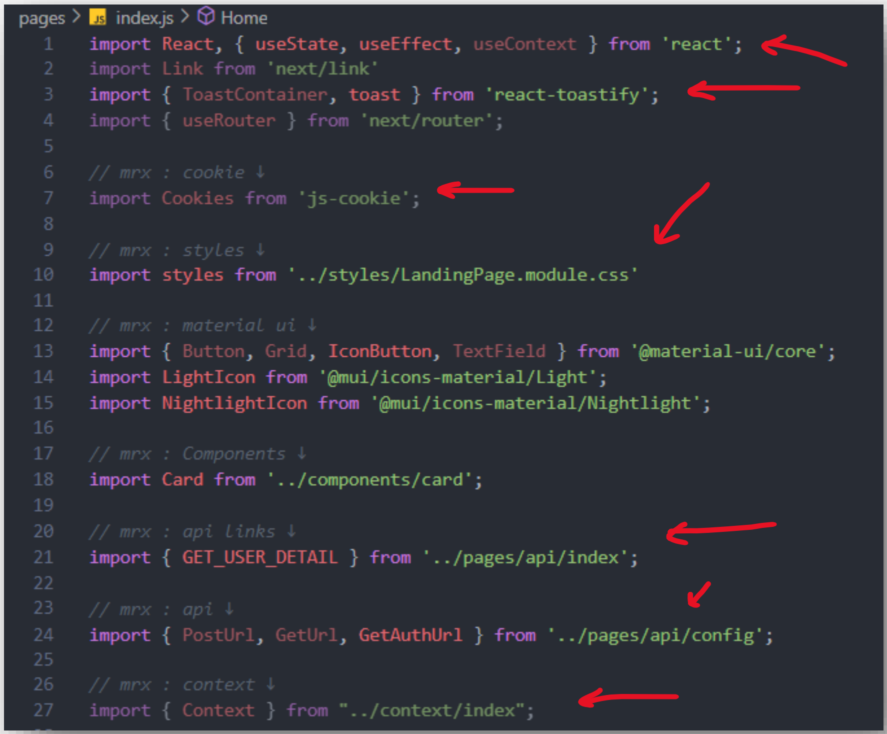
```
export const Context = react.createContext({
    Example: [],
    setExample: [],
});

const ContextProvider = (props) => {
    const [Example, setExample] = react.useState([]);

    return (
        <Context.Provider
            value={{
                Example,
                setExample,
            }}
        >
            {props.children}
        </Context.Provider>
    );
}

export default ContextProvider;
```

Context.Provider

- **We don't use redux** for our global state we use context but if you needed Reducer we **recommend** to use.

❖ Now we gone talk about our index file in **pages > index.js:**

- In this file, which is our **landing page** file we import all modules that we need but we must import some module almost every time to make our source code clean and this module (v1) are: **Cookies, toast, useState, useEffect, useContext, Our context provider**
- **Cookies:** is a module for set our cookies it makes it really simple

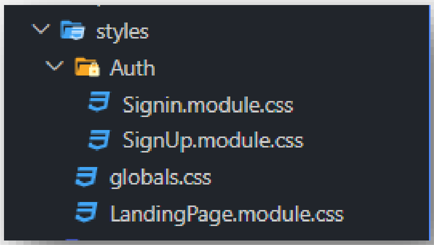


```
import React, { useState, useEffect, useContext } from 'react';
import Link from 'next/link'
import { ToastContainer, toast } from 'react-toastify';
import { useRouter } from 'next/router';

// mrx : cookie ↓
import Cookies from 'js-cookie';

// mrx : styles ↓
import styles from '../styles/LandingPage.module.css'

// mrx : material ui ↓
import { Button, Grid, IconButton, TextField } from '@material-ui/core';
import LightIcon from '@mui/icons-material/Light';
import NightlightIcon from '@mui/icons-material/Nightlight';

// mrx : Components ↓
import Card from '../components/card';

// mrx : api links ↓
import { GET_USER_DETAIL } from '../pages/api/index';

// mrx : api ↓
import { PostUrl, GetUrl, GetAuthUrl } from '../pages/api/config';

// mrx : context ↓
import { Context } from "../context/index";
```

- **We import this tree (useState, useEffect, useContext) in every page** *
- **Styles:** our style of this page that I will talk about it in page 2222
- **We use material-Ui for our ui module**
- We are having some **Api links** and **Api** that I talk about it in page: 2222
- A context in our provider to access to the global states that I talk about it in **(Page 2)**

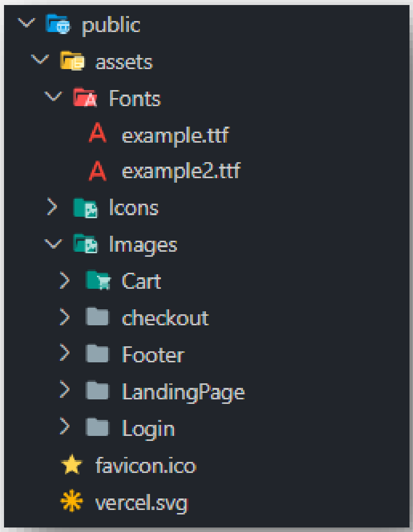
And one of the most important things is that we must have comments in this source if attention you see that we have comments every Vere (comments must be just like the examples that we have in source)

❖ In this section we are gone talk about **Styles** and **images** or **icons** (**assets**):


```
∨ styles
  ∨ Auth
      Signin.module.css
      SignUp.module.css
    globals.css
    LandingPage.module.css
```

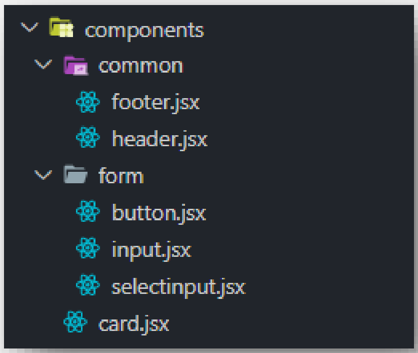
- In directory: > **styles** we have our CSS or styles the structure of this folder, we have one global.css that is our main style and we imported in **> pages > _app.js** we set styles of our components in this file and we must use comments as I say it in **page (3)** it's really important
- We having a landingPage.css that it's a style of the **single files** but for multi files like **authentication** we **make a new folder** for theme thane set the styles

- For images and icons, Fonts, we import theme in public folder


```
∨ public
  ∨ assets
    ∨ Fonts
        example.ttf
        example2.ttf
    > Icons
    ∨ Images
      > Cart
      > checkout
      > Footer
      > LandingPage
      > Login
    favicon.ico
    vercel.svg
```

❖ In this section we are gone talk about **Components**:

- In folder: **> components >** we are having our component structure **common** folder for some **components like Header, Footer, LeftSideMenu …**
- **Form folder is for your inputs will having your form components** the rest of the file are the **components of folders or files**

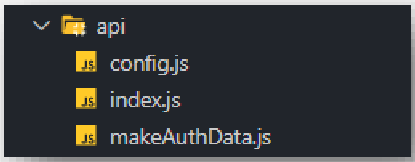


```
components
  common
    footer.jsx
    header.jsx
  form
    button.jsx
    input.jsx
    selectinput.jsx
  card.jsx
```

- **The components must me JSX files not JS**

❖ In this section we are gone talk about (API), one of the most important parts of this source:

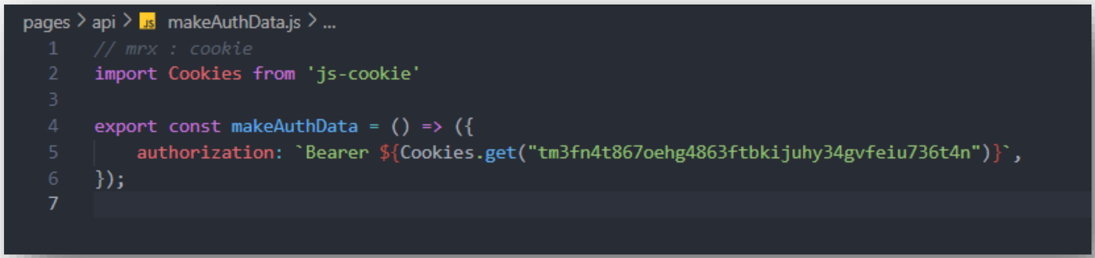
- having our api directory in **> pages > api >** in this folder we have 3 files
- **First one is:** config.js, in this file **we** have our crude methods Get, Post, Put, Delete



```
api
  config.js
  index.js
  makeAuthData.js
```

- we create dynamic functions for these crude methods also have a **GetAuthUrl** these items are the methods that need token in header and we send the token in

  > pages > api > makeAuthData.js



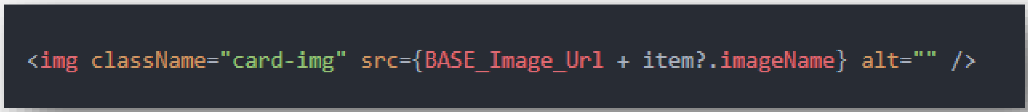
pages > api > makeAuthData.js > ...

```js
// mrx : cookie
import Cookies from 'js-cookie'

export const makeAuthData = () => ({
    authorization: `Bearer ${Cookies.get("tm3fn4t867oehg4863ftbkijuhy34gvfeiu736t4n")}`,
});

```

- we used **Cookies** module for get token that set-in cookie when user did login or register

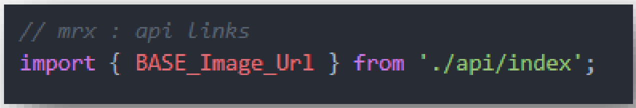
  and set it in json and also give it a key (authorization) that backend could get the token in header with this key.

- **tm3fn4t867oehg4863ftbkijuhy34gvfeiu736t4n: this is a name of our cookie for make it secret** 😊
- in **> pages > api > index.js** we imported our API router of backend links and make a
- **Base URL:** for the backend link to make it dynamic
- **BASE_Image_Url:** for the dynamic image URL in our files for example



```jsx
<img className="card-img" src={BASE_Image_Url + item?.imageName} alt="" />
```

we import **BASE_Image_Url** from this directory and were using it

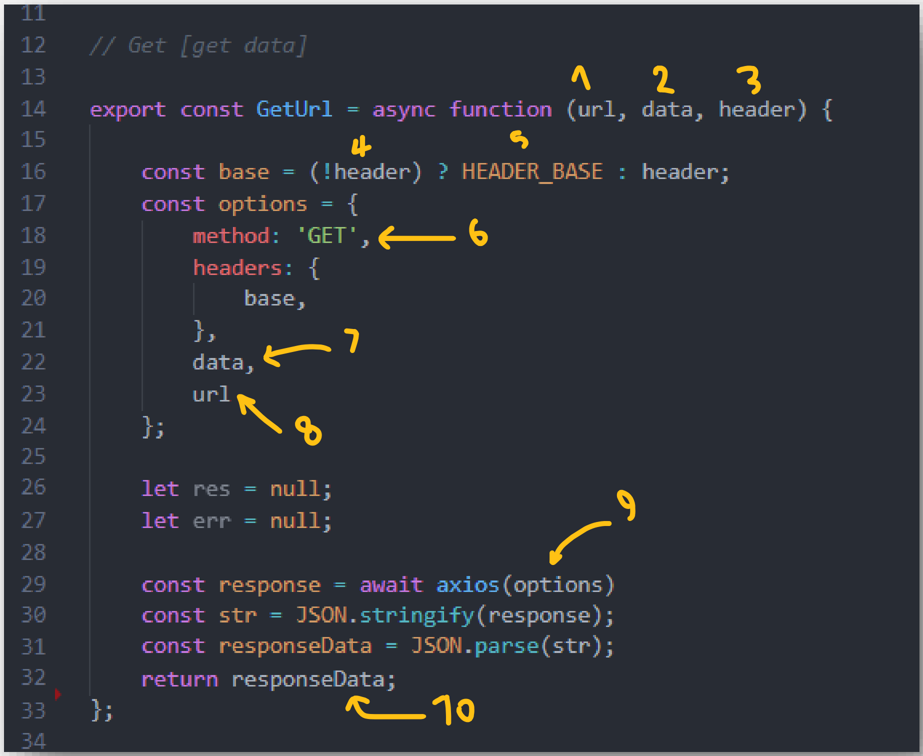


```
// mrx : api links
import { BASE_Image_Url } from './api/index';
```

- Now we talk about this tree file in **Api folder** in next item were gone talk about to how use the api and **fetching** the **data**:

  We use Axios For fetching data from backend, we have 4 methods as I say:

Get + GetAuthUrl:



```

// Get [get data]

export const GetUrl = async function (url, data, header) {

    const base = (!header) ? HEADER_BASE : header;
    const options = {
        method: 'GET',
        headers: {
            base,
        },
        data,
        url
    };

    let res = null;
    let err = null;

    const response = await axios(options)
    const str = JSON.stringify(response);
    const responseData = JSON.parse(str);
    return responseData;
};

```

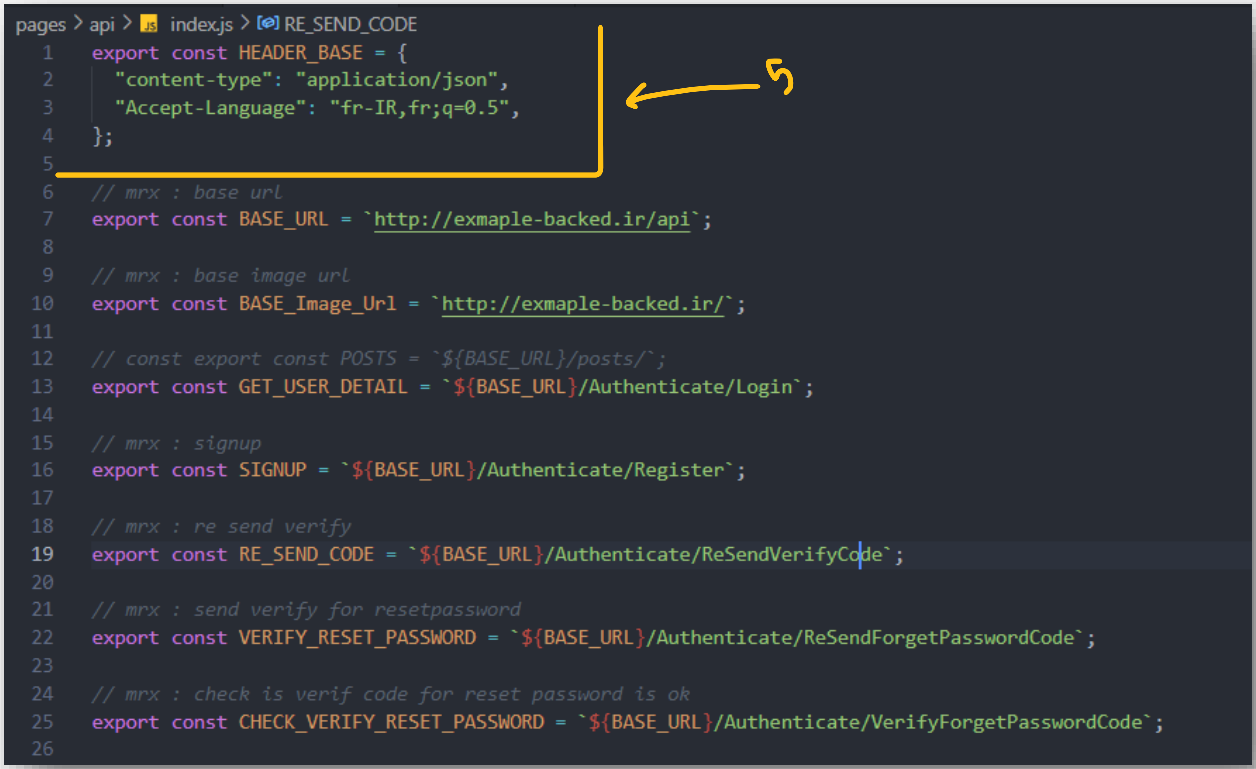
1: URL: this is a link, if the fetch that we send, here with props

2: data: this is a (json) data that we send it from fetch with props

3: header: we don't need to send this header in props (in this version v1)

4: header: we don't need to send this header in props (in this version v1)

5: Header base: we import this from our index.js file in Api folder



```js
export const HEADER_BASE = {
  "content-type": "application/json",
  "Accept-Language": "fr-IR,fr;q=0.5",
};

// mrx : base url
export const BASE_URL = `http://exmaple-backed.ir/api`;

// mrx : base image url
export const BASE_Image_Url = `http://exmaple-backed.ir/`;

// const export const POSTS = `${BASE_URL}/posts/`;
export const GET_USER_DETAIL = `${BASE_URL}/Authenticate/Login`;

// mrx : signup
export const SIGNUP = `${BASE_URL}/Authenticate/Register`;

// mrx : re send verify
export const RE_SEND_CODE = `${BASE_URL}/Authenticate/ReSendVerifyCode`;

// mrx : send verify for resetpassword
export const VERIFY_RESET_PASSWORD = `${BASE_URL}/Authenticate/ReSendForgetPasswordCode`;

// mrx : check is verif code for reset password is ok
export const CHECK_VERIFY_RESET_PASSWORD = `${BASE_URL}/Authenticate/VerifyForgetPasswordCode`;
```

6: We create an options function to set our options to Axios to make the code clean, in this option function we have headers, method, data, URL

7: this is the (Json) data that we get from props and send it here

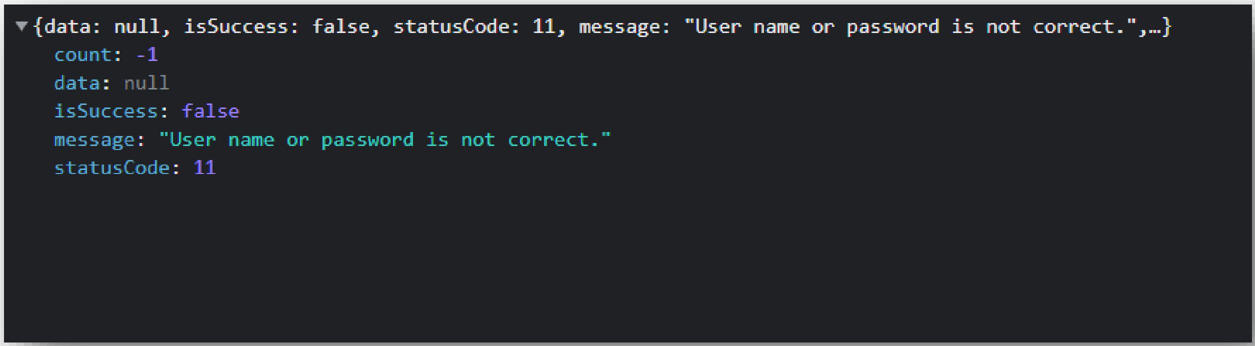
8: this is a URL that we get from props

9: now we set the options functions to the Axios

10: after that we got the response of the request that we send

… The response could be 200 or … but the way we using it is that we get an is success field that it is true or false

For example, we have a login auth that we send user name and password if the user name was wrong, we get **is Success false** and we have a message that tell us what is going on



```
▼{data: null, isSuccess: false, statusCode: 11, message: "User name or password is not correct.",…}
   count: -1
   data: null
   isSuccess: false
   message: "User name or password is not correct."
   statusCode: 11
```

But if the is **Success was true,** we get token an succeed message and the token!!

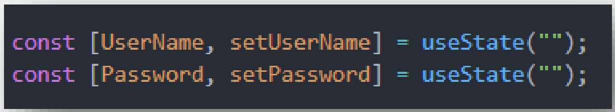
Ok now were gone talk about to how send request using the information up there:

For example, the login auth that we talk about it up there, now we want to send a request for login what we should do I'm going to talk about it step by step

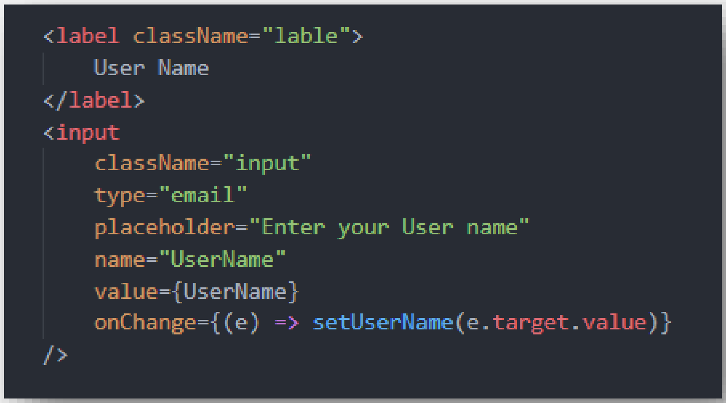
1. First, how set user information to the state
2. Then, how to verify the data before we send it to the backend
3. And in the end how we send data and get response

Number one:

We having email and password so we need two state on for email and one for password



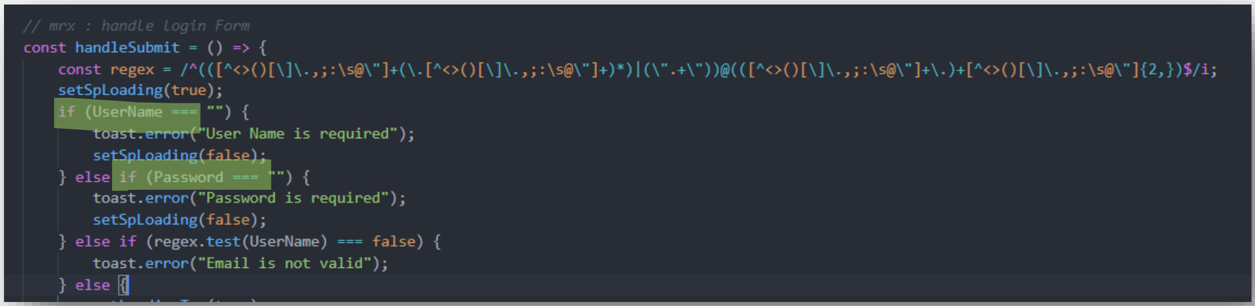
```
const [UserName, setUserName] = useState("");
const [Password, setPassword] = useState("");
```


```
<label className="lable">
    User Name
</label>
<input
    className="input"
    type="email"
    placeholder="Enter your User name"
    name="UserName"
    value={UserName}
    onChange={(e) => setUserName(e.target.value)}
/>
```

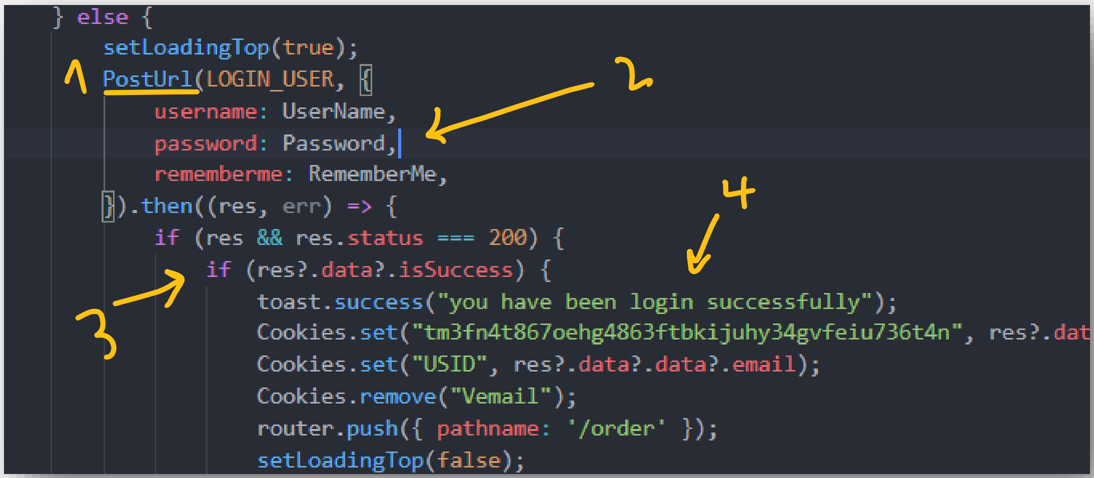
Number two:

We must verify the data for send we created a function and when click in login button we call that

We created a regex for email


```
// mrx : handle login Form
const handleSubmit = () => {
    const regex = /^(([^<>()[\]\.,;:\s@\"]+(\.[^<>()[\]\.,;:\s@\"]+)*)|(\".+\"))@(([^<>()[\]\.,;:\s@\"]+\.)+[^<>()[\]\.,;:\s@\"]{2,})$/i;
    setSpLoading(true);
    if (UserName === "") {
        toast.error("User Name is required");
        setSpLoading(false);
    } else if (Password === "") {
        toast.error("Password is required");
        setSpLoading(false);
    } else if (regex.test(UserName) === false) {
        toast.error("Email is not valid");
    } else {
```

As you can see, we have comment that say what this function does and we check the data is ok after everything was ok, we send the fetch API



```
} else {
    setLoadingTop(true);
    PostUrl(LOGIN_USER, {
        username: UserName,
        password: Password,
        rememberme: RememberMe,
    }).then((res, err) => {
        if (res && res.status === 200) {
            if (res?.data?.isSuccess) {
                toast.success("you have been login successfully");
                Cookies.set("tm3fn4t867oehg4863ftbkijuhy34gvfeiu736t4n", res?.dat
                Cookies.set("USID", res?.data?.data?.email);
                Cookies.remove("Vemail");
                router.push({ pathname: '/order' });
                setLoadingTop(false);
```

1: we imported the method from > pages > API > config.js and we send the URL with the props and we imported the URL from > pages > API > index.js

2: then we send the data like user name, password, …

3: we get response

4: if the response true we show user a message and set token with

jscookie and …

Created by Alireza askari