UNIVERSITY OF BUEA
FACULTY OF ENGINEERING AND TECHNOLOGY
COMPUTER ENGINEERING

CEF 344: CLIENT-SERVER AND WEB APPLICATION DEVELOPMENT

CLIENT PART (Practical React Js)

As an aspiring software engineer, it is important for every student to design their portfolio and host it online. This is proof that you are putting the knowledge you are learning into practice. This portfolio will be dynamic and will serve later in your multiple job searches as a witness to your competence.

That said, throughout this hands-on course, we'll be building a personal portfolio. The first part of the course will be done in the classroom and the second part will constitute the project which will serve as your evaluation.

The link (https://client-henna-xi.vercel.app/)is the representation of what we will design throughout this practical course. Feel free to change the values used to your liking while obtaining a responsive and readable platform.

What we will learn during this course:
- Build a react project from Scratch
- Use react hook (Usestae hook)
- Use of react icon
- Modern and responsive design with CSS3
- Multiple contact option (Email, Messenger, WhatsApp).
- Create carousels/slides with swiper
- Deploy your website.

During the previous lesson, we downloaded and installed nodejs and created our first react project. we will use the same command to create our react project named portfolio.

Let's open VsCode in the folder where we want to host our project locally and run the command to create our portfolio project: **npx create-react-app portfolio**.

Once the project is created, we go to the src folder and we are going to delete everything that is there. We will do the same for the public folder.

The folder project structure should now look like this:

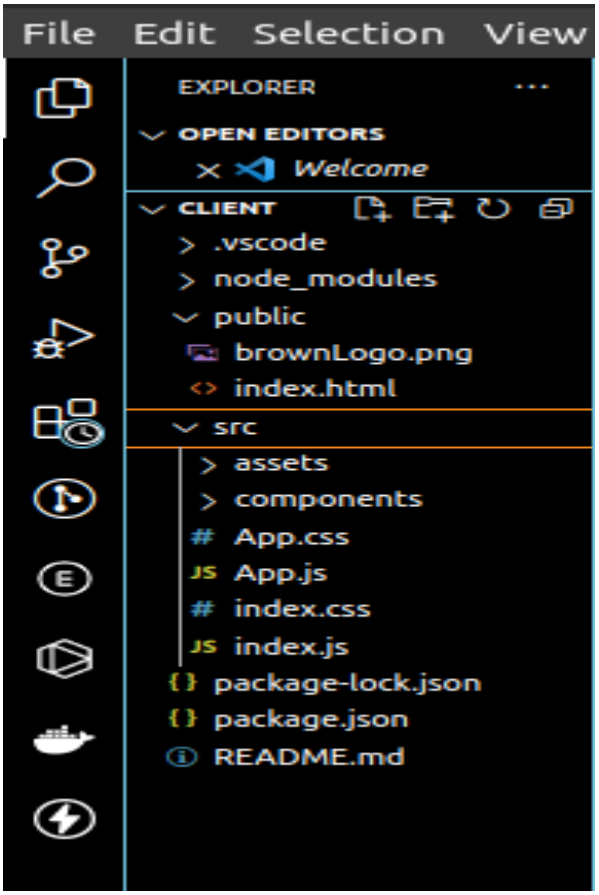

Figure 1: Project folder structure

In the public folder, create an index.html file; at the far left, on the block with 4 squares (extension), enter in the search bar es7 + react/redux/react-native snippets, copilot, github, react in short all the extensions that will allow you to carry out finish your react project.

in the recently created index.html file in public, enter the <!Doctype html> and using es7 + react/redux/react-native snippets, you will have the basic structure of an html project.

Change the content of the **<title>**MyName portfolio**</title>.**
also in the
<body>
//add the div tag like follow
** ** // this is where component of page of our app will be rerendered
</body>

```
<!DOCTYPE html>
<html lang="en">
 <head>
   <meta charset="utf-8" />
   <link rel="icon" href="./brownLogo.png" />
   <meta name="viewport" content="width=device-width, initial-scale=1" />
   <meta name="theme-color" content="#000000" />
   <meta
     name="description"
     content="Web site created using create-react-app">
   <title>Brown Portfolio</title>
 </head>
 <body>
```

```
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
  </body>
</html>
```

Script 1: content of the new created index.html in the public folder

in the src folder, create another folder called assets and inside, put all the images that will be used

in the same src folder, create and index.js file and index.css file. The structure of the src folder is shown in the above image.

In the index.js file, enter the following code:

```
import React from 'react';
import ReactDOM from 'react-dom/client';
reactDom.render(<APP/> , document.getElementById('root')); //THIS LINE INDICATE THAT
// ALL THE CONTENT OF THIS FILE SHOULD BE RENDERED IN THE ROOT ID OF THE HTML FILE
```

Script 2: content of the index.js

The `<APP/>` component is not yet created. The editor may indicate and error and if we start the server, an error will be displayed indicating that the <APP/> is not found.

Let's create the `<APP/>` component in the **App.jsx** file under the src folder like indicated in **figure 1;** inside the folder, let's create a basic functional component using the emmit shortcut extension by typing **rafce.** This is how the content will look like:

```
import React from 'react'

export const App = () => {
  return (
    <div>
      App
    </div>
  )
}
export default App; // always export the component as default
```

Script 3: content of the app.jsx

Then go back in the index.js and import the component we just create

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import App from './App';

reactDom.render(<APP/> , document.getElementById('root')); //THIS LINE INDICATE THAT
// ALL THE CONTENT OF THIS FILE SHOULD BE RENDERED IN THE ROOT ID OF THE HTML FILE
```

Script 4: updated version of the index.js

run **npm start** to strat the server.

Now we will create an index.css file and inside this file, we will put our global css file.

```
@import
url('https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;500;600&display=swap'); // used of google fonts

*{
 margin: 0;
 padding: 0;
 border: 0;
 outline: 0;
 box-sizing: border-box;
 list-style: none;
 text-decoration: none;
}

:root{ // as root variable we can used it everywhere
 --color-bg:#1f1f38; // used two(-) to declare css variable
 --color-bg-variant:#2c2c6c;
 --color-primary:#4db5ff;
 --color-primary-variant:rgba(77, 181, 255, 0.4);
 --color-white:#fff;
 --color-light:rgba(255,255,255, 0.6);

 --container-width-lg:75%; /*size of the container we are going to use on different*/
 --container-width-md:86%;/*screen size*/
 --container-width-sm:90%;

 --transition: all 400ms ease;

}

html{
 scroll-behavior: smooth;
}

::-webkit-scrollbar{
 display:none
}

body {
 font-family: 'Poppins', sans-serif;
 background: var(--color-bg);
 color: var(--color-white);
 line-height: 1.7;
}

.container{
 width: var(--container-width-lg);
```

```
 margin:0 auto;
}

h1, h2, h3, h4, h5{
 font-weight: 500;
}

h1{
 font-size: 2.5rem;
}

section{
 margin-top: 8rem;
}

section > h2,
section > h5{
 text-align: center;
 color: var(--color-light);
}

section > h2{
 color: var(--color-primary);
 margin-bottom: 3rem;
}

.text-light{
 color: var(--color-light);
}

a{
 color: var(--color-primary);
 transition: var(--transition);
}

a:hover{
 color: var(--color-white);
}

.btn{
 width: max-content;
 display: inline-block;
 color: var(--color-primary);
 padding: .75rem 1.2rem;
 border-radius: .4rem;
 cursor: pointer;
 border: 1px solid var(--color-primary);
 transition: var(--transition);
}
```

```
.btn:hover{
 background:var(--color-white);
 color: var(--color-bg);
 border-color: transparent;

}

.btn-primary{
   background: var(--color-primary);
   color: var(--color-bg);
}

img{
 display: block;
 width: 100%;
 object-fit: cover;
}


@media screen and (max-width:1024px) {
 .container{
   width: var(--container-width-md);
 }

 section{
   margin-top: 6rem;
 }
}

@media screen and (max-width:600px) {
 .container{
   width: var(--container-width-sm);
 }

 section > h2{
   margin-bottom: 2rem;
 }
}
```

Script 5: general css file

In the src folder, create a folder called components; this folder will hold all components of this project. Bellow, the structure of the component folder.

Figure 2: components folder structure

As you can see, the css of our project is not all written in a single css file, on the contrary, each component is associated with its css. Thereafter, all the components created will be imported and rendered in the App.jsx file as shown in the screenshot below:

```
import './App.css';
import Header from './components/header/Header'
import Navbar from './components/navbar/Navbar'
import About from './components/about/About'
import Experience from './components/experience/Experience'
import Services from './components/services/Services'
import Portfolio from './components/portfolio/Portfolio'
import Testimonial from './components/testimonial/Testimonial'
import Contact from './components/contact/Contact'
import Footer from './components/footer/Footer'

function App() {
 return (
   <>
     <Header/>
     <Navbar/>
     <About/>
     <Experience/>
     <Services/>
     <Portfolio/>
```

```
      <Testimonial/>
      <Contact/>
      <Footer/>
    </>
  );
}

export default App;
```

Script 6: General App.jsx file updated

With all of this done, let’s start working on different sections of our portfolio starting on the header section.

So open/create the header folder under the src folder,