SQL

REST

WEB

UI

(React JS App)

Web

Server

Application

(Spring Boot Application)

DB

(DB2/Maria-DB/Oracle/MySQL/MS-SQL/H2)

Development Environment Check:

Check for node js:

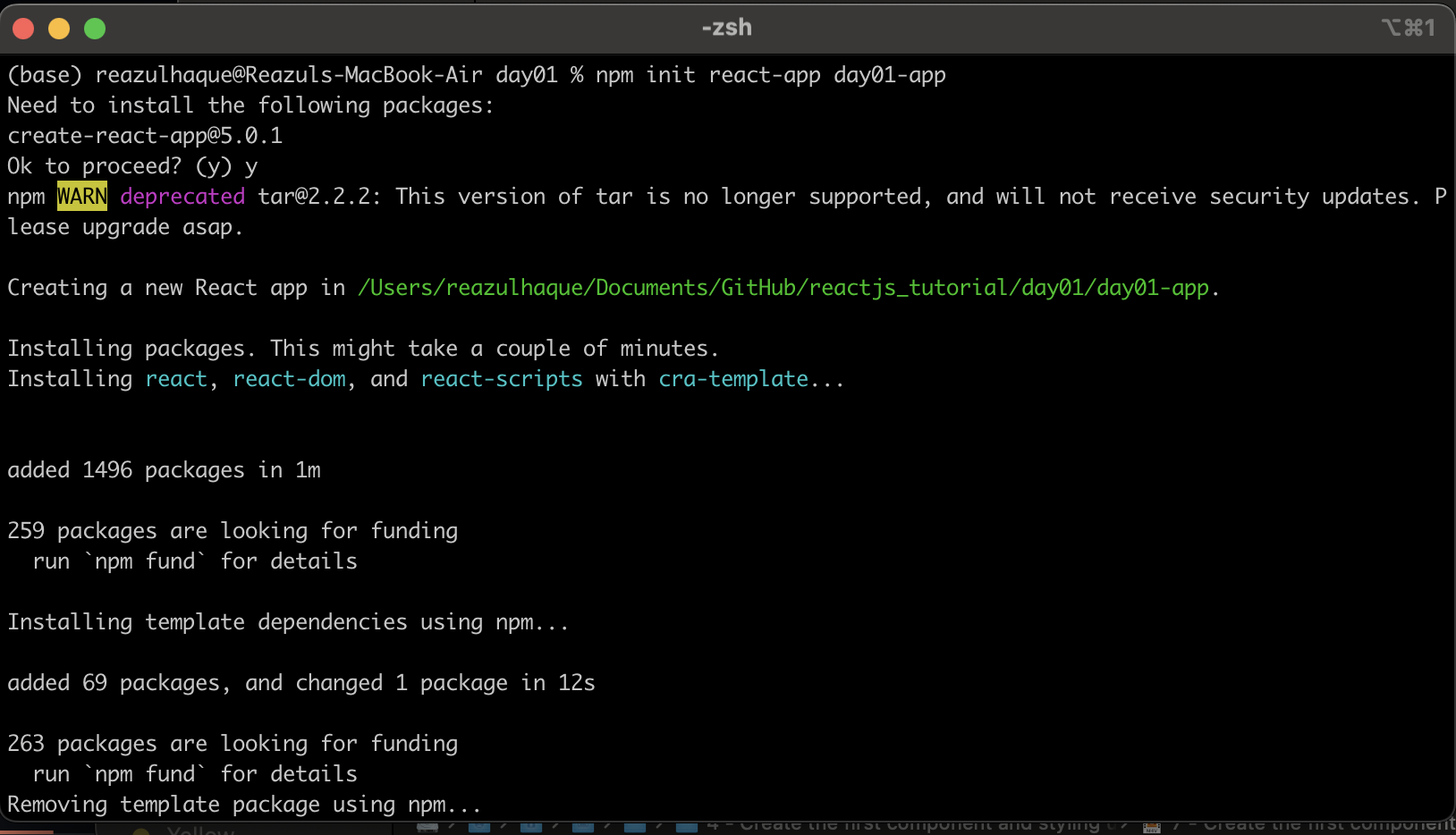
node --version

Check for npm version:

npm --version

Create a Ractjs application

npm init react-app day01-app



Go to the application folder

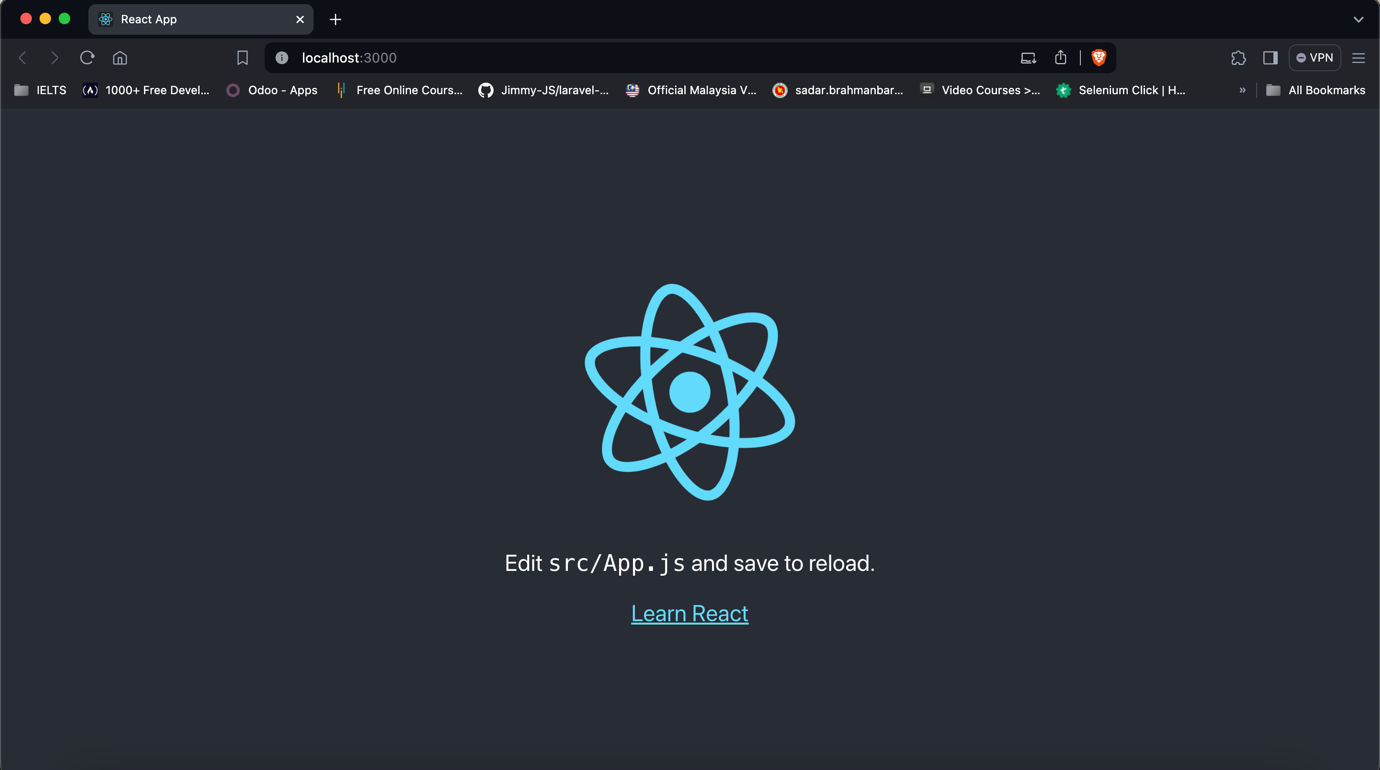
cd day01-app

Then run

npm start

This command will run the reactjs application on default port(3000). If the default port used by other application, reactjs will try to use next available port for 3001. It’s a incremental way of searching free port, means next port to serach is 3002, 3003 and so on….

Fig:Reactjs\_defaultPage



Let’s look to the folder structure:

We’ll use Visual Studio Code (VS Code) for reactjs development, you can also use your favorite IDE for reactjs. Let’s add the application folder to VS code workspace.



**node\_modules**: this folder contains all dependent libraries required by reactjs

**public**: all public files are in public folder

**src**: this is our main folder, where all codes belong.

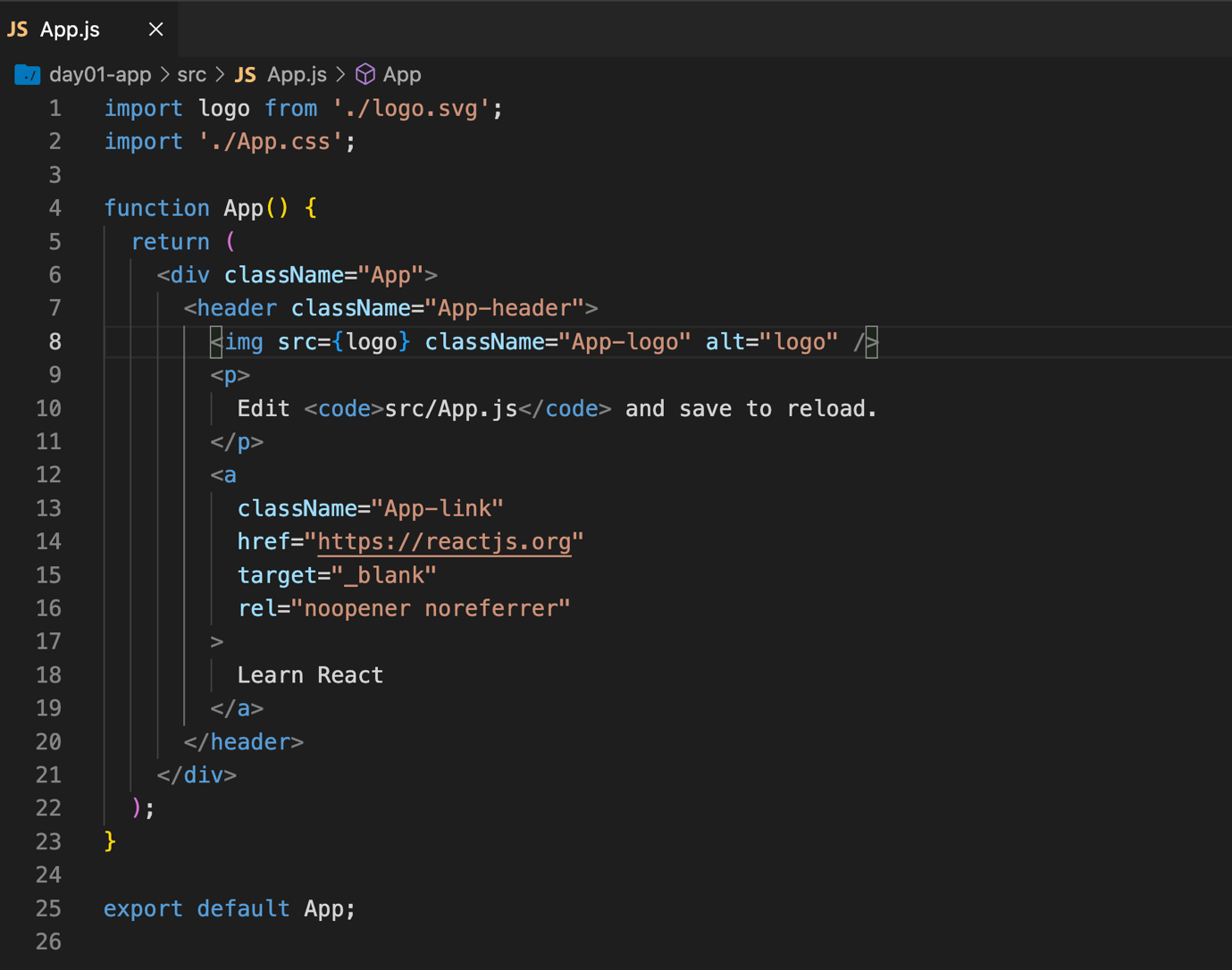
**.gitignore**: it’s git file for ignoring which file or folder we don’t add to git e.g node\_modules folder. We don’t need to add this folder to git.

**package.json**: it’s a dependency manager file like pom.xml. We can add our required library in this file.

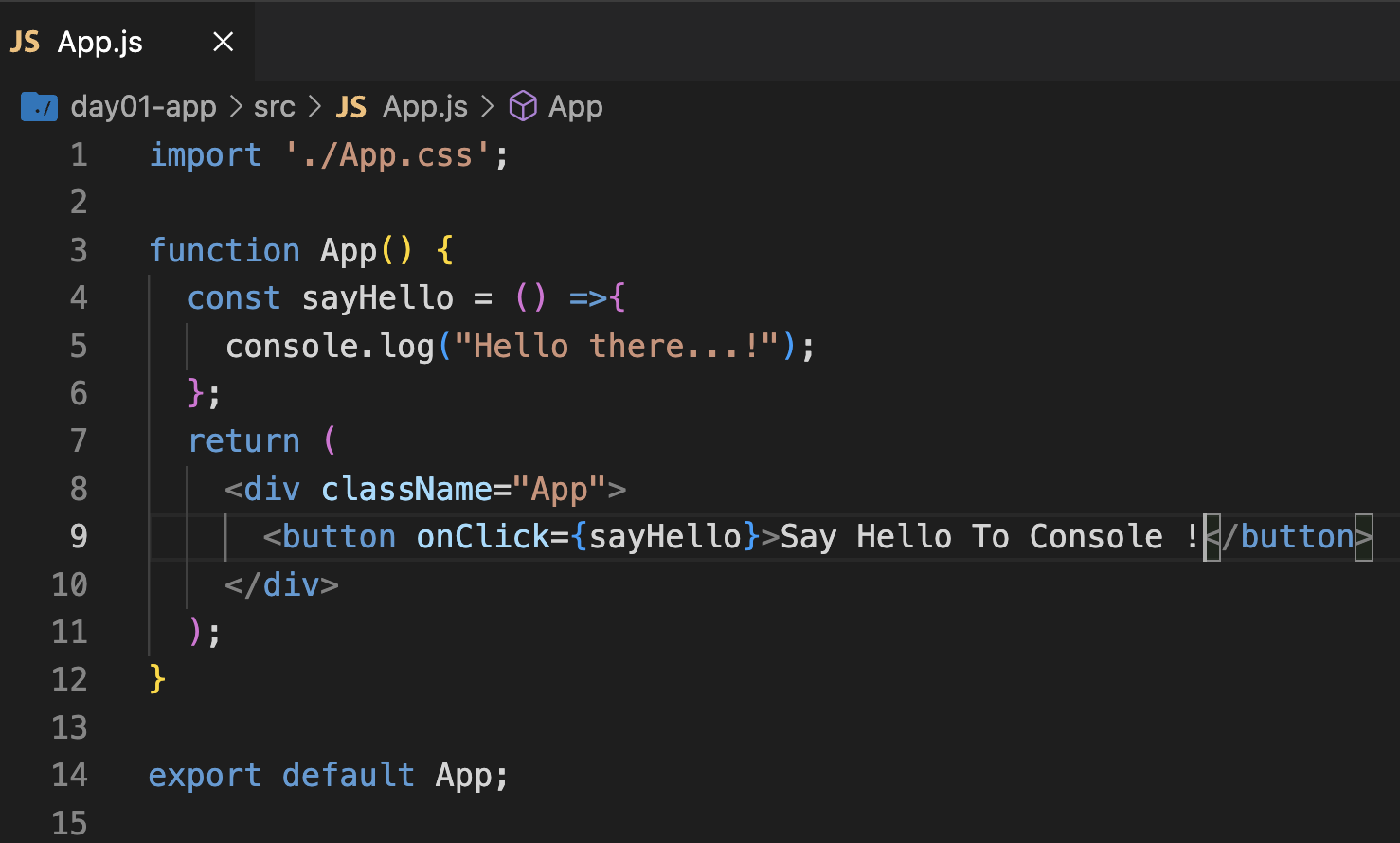
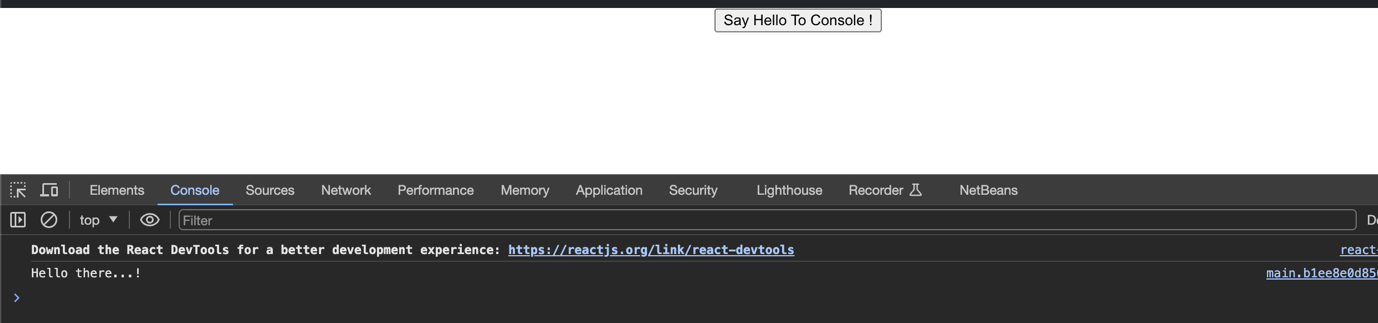
**README.md**: default file for git, to describe about the application or whatever comments we want to add for others.

Edit Code:

src/App.js is the default file from reactjs



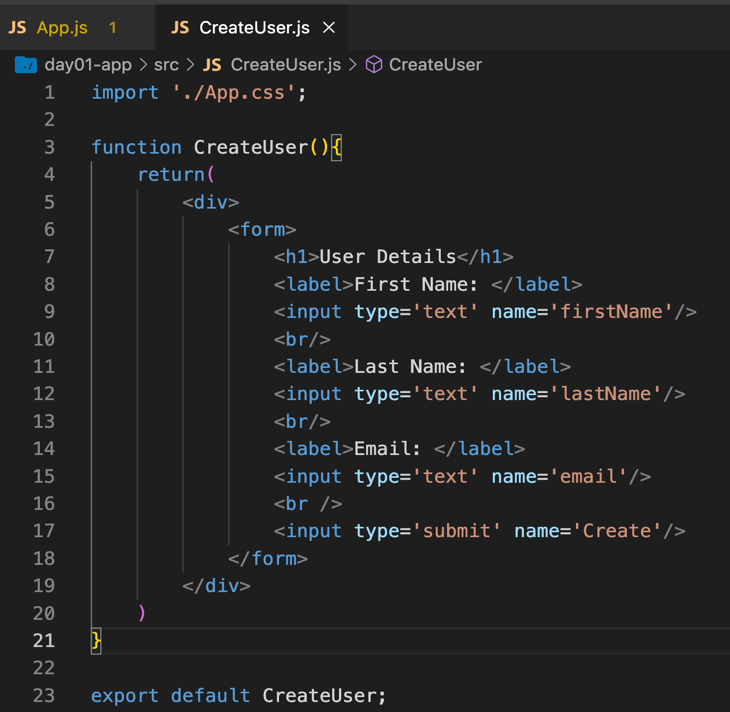
This is the default code by reactjs while we create our application. This code rander in browser as in Fig:Reactjs\_defaultPage. Now we’ll add our own code to src/App.js file:



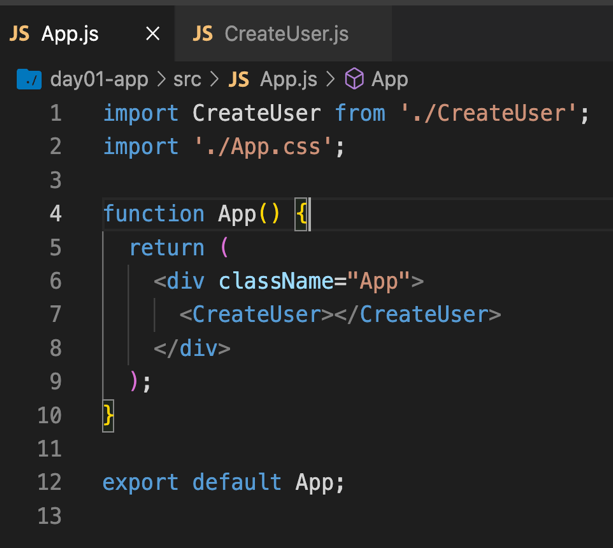
Let’s create our first component:

There are two different ways to implement component one is functional component (which is most popular) other one is class component. We’ll use functional component for learning and development.

Create a js file in src folder name CreateUser.js and the below code:



Now, import our newly created component to App.js as in below



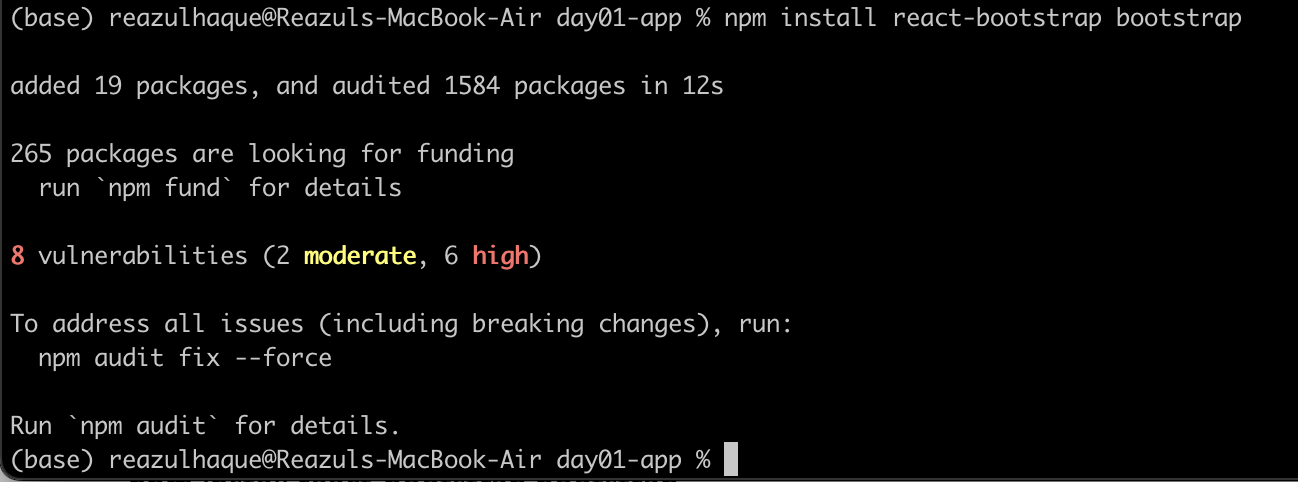
After update the App.js, we’ll get the view as below



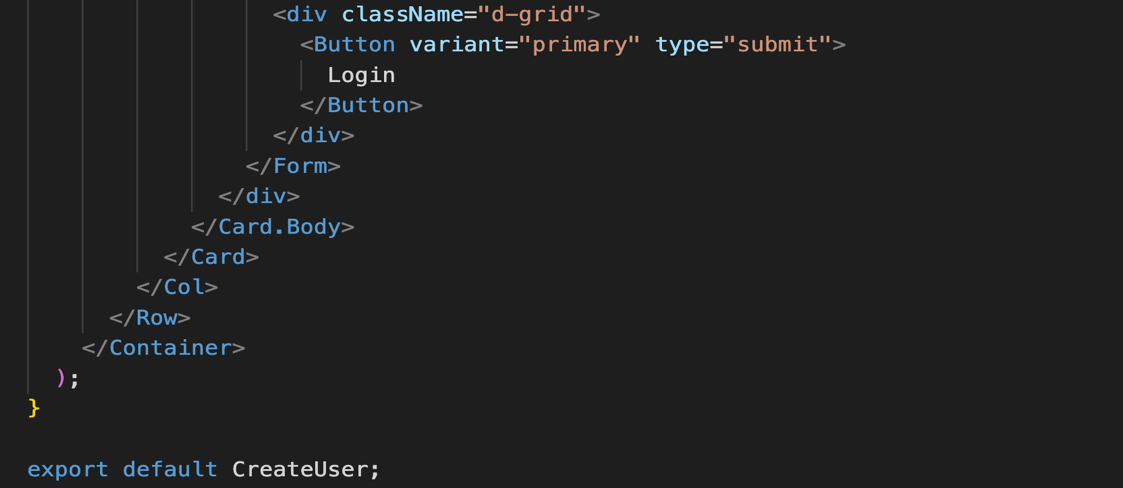
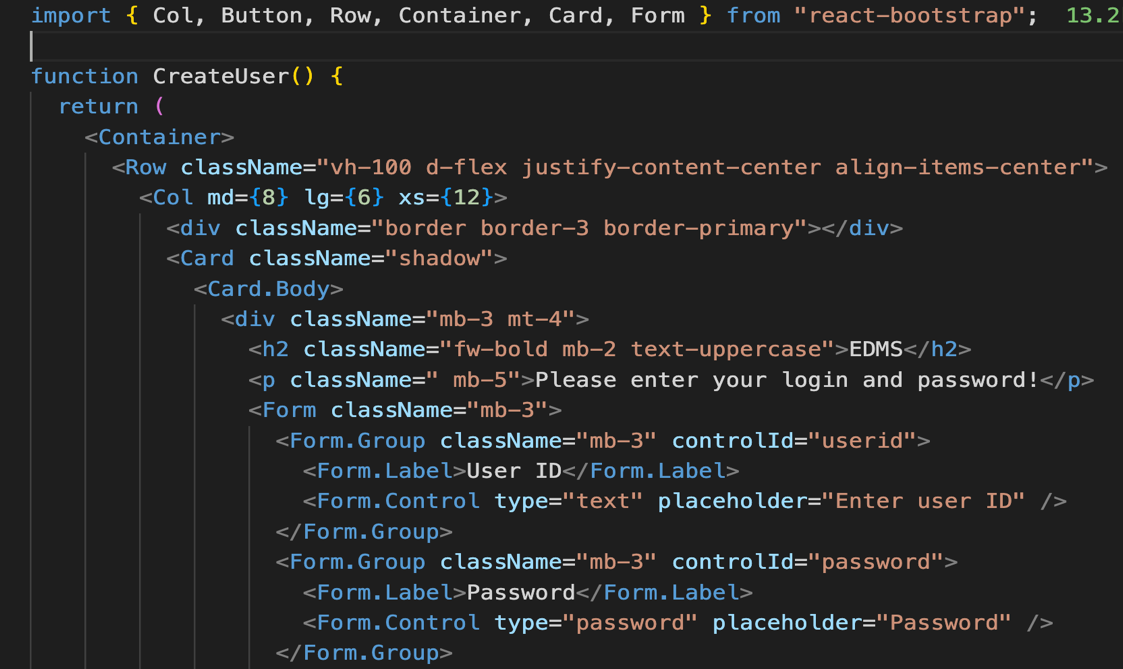
Oops… it’s not a professional look at all, so it’s time to add mighty bootstrap to our project using below command

npm install react-bootstrap bootstrap

We’ll get a view as in below after bootstrap installation



Now, we’ll add bootstrap to our CreateUser component. After adding bootstrap, our CreateUser component will be look like in below

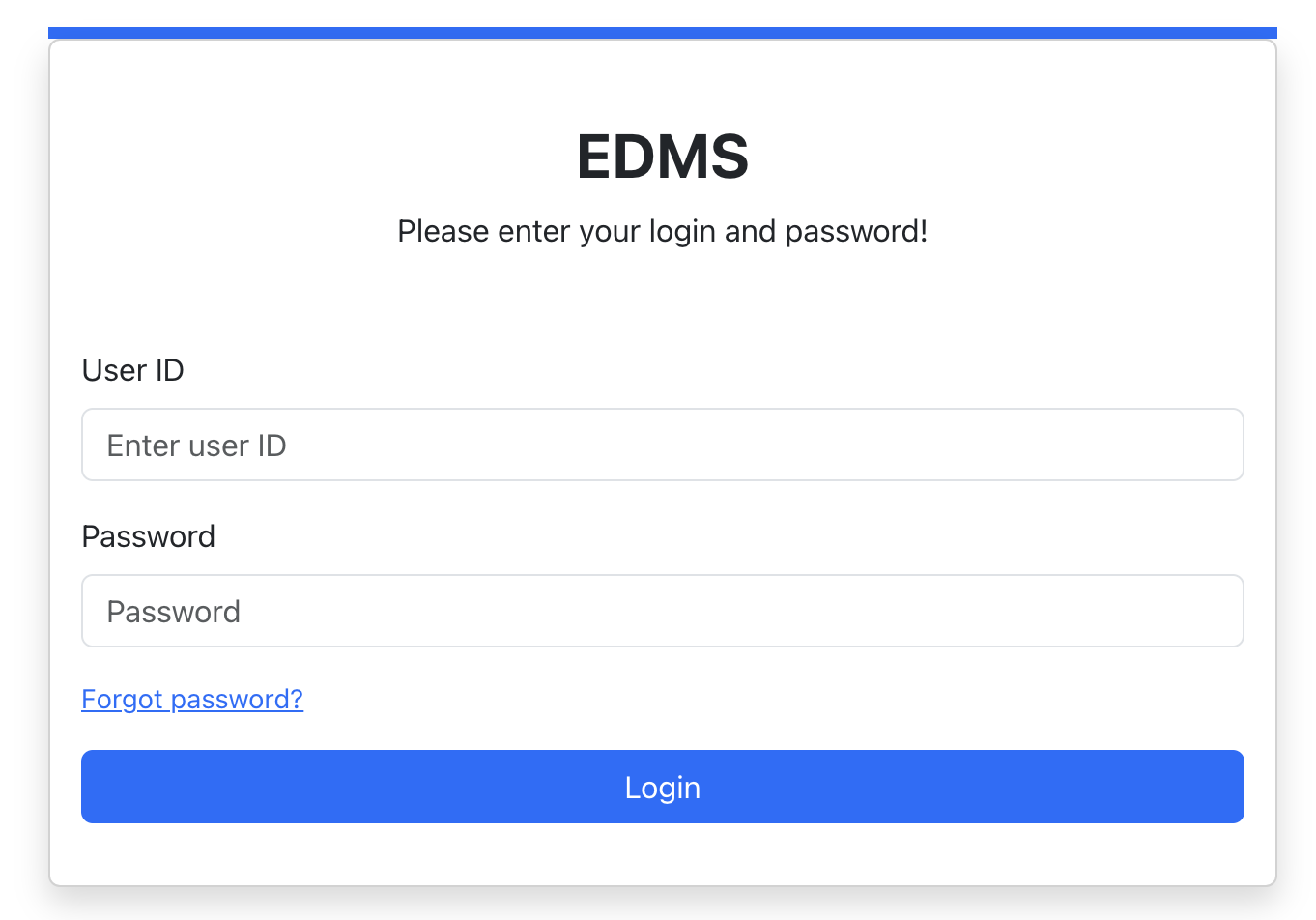


Also need to add bootstrap css to index.js file



Adding bootstrap css

By adding bootstrap our application looks like below



Cool…. You can checkout react-bootstrap official documentation for some awesome design

<https://react-bootstrap.netlify.app/>

Now, we’ll try to add react hook useState. useState is react hook that allows you to add state to a functional component. It returns an array with two values, one is the current state and other is a function to update it. The hook takes an initial state as an argument and returns an updated state value whenever the setter function is called.

Example:

const [userId, setUserId] = useState("");

const [welcomeMsg, setWelcomeMsg] = useState("");

In react, useState can store any type of value, whereas the state in a class component is limited to being an object. This includes primitive data types like string, number and Boolean as well as complex data types such as array, object and function. It can even cover custom data types like class instances. Basically, anything that can be stored in a JavaScript variable can be stored in a state managed by useState.

Updated CreateUser.js File:

