Chapter 4 - React Setup

-:Content:-

* Introduction
* React setup
* Install packagers
* App.js
* Connect backend using api.js
* Login.js
* Registraion.js
* Router.js
* Home.js
* Use login and registration.

1. Introduction

* React is a JavaScript library for building user interfaces.
* React is used to build single-page applications.
* React allows us to create reusable UI components.

2. React Setup

First thing before starting our setup , we have to organize our codes so that we can have parallel services running in the frontend . Create one folder and call it **backend** ,move all the codes that we do in previous tutorial and put it in this folder **(backend).** For run the two different server split the terminal for run two server.

**Steps to run the server from these two different directories .**

**Step 1** - If we want to start server from backend folder , the command we use to switch from main directory to backend folder is **cd backend** With the help of this command , we will switch to the folder backend .

**Step 2** – After switch to backend folder we will now to run the server , **npm start**

**Step 3 –**  If we want to come back to main folder from backend folder , **cd..**

So now ,after organize the codes we will do setup of react application.

**npx i create-react-app frontend**

Wait few minutes for creating the app . So after setup we will cleanup an unused code that we have . There is library that is called reactstrap , which is the version of bootstrap . On terminal , we are going to install this library .

**npm install --save bootstrap**

After finishing , let’s install reactstarp library .

**npm install - -save reactstrap**

To import bootstarp CSS in **src/index.js** file ,

**import‘bootstrap/dist/css/bootstrap.min.css’**

Start the server once again and remember that you need to be inside the frontend folder .For start the server use **npm start**.After that , we will move to **App.js** file , and start writing some code

* **Src > app.js**

import { Container } from 'reactstrap'

return (

    <Container>

    <h1>Sport's App</h1>

    </Container>

  );

First thing we will do , create new folder in src and name it as **“pages ”.** in this folder we have different pages like login page ,registration page ,dashboard page and home etc . Inside this folder we will create another initial folder **“Login”**. As we do not have anything in this folder , we will create new file **“login.js”**  **.**

Flowing this path :- **Pages > login > login.js**

* In login.js we write the code

import React from 'react'

const Login =()=>{

return (

<div>

<h1>login</h1>

</div>

)

}

export default Login;

React has a library they’re kind of very helpful us in navigating around the things , that is called **react-router-dom .**  Stop the server and for installing. Use this command for install react-router- dom

**npm i react-router-dom**

In source folder , we made new file called **“router.js” . following like this Scr >router.js.**

**Router.js**

import React from 'react'

import { BrowserRouter, Routes,Route } from 'react-router-dom';

import Login from './pages/Login/'

export class router extends Component {

  render() {

    return (

        <BrowserRouter>

            <Routes>

               <Route path="/login" element={<Login />} />

</Routes >

        </BrowserRouter>

    );

}

After that we are import router.js on “**app.js**”.

import Routes from './routes';

return (

    <Container>

    <h1>Sport's App</h1>

      <Routes/>

    </Container>

  );

Api Setup

Next thing is the configuration that allow us to use another library to fetch the data from API . Stop the server once again.

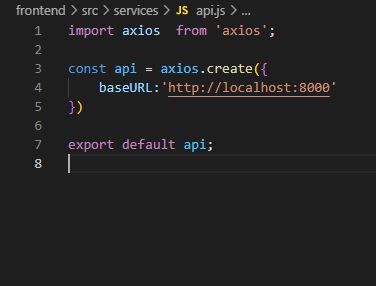
For backend connectivity we use Axios. It is a Javascript library used **to make HTTP requests from node.** **js or XMLHttpRequests from the browser** and it supports the Promise API that is native to JS ES6.

For install Axios we use following command:- **npm i axios.**

After istall Axios we move to in source folder , create another folder **“services” .** Inside this folder we’re going to create an new file and will be called **“api.js” .**

**Flowing the path Src >Services > api.js**

**Api.js**

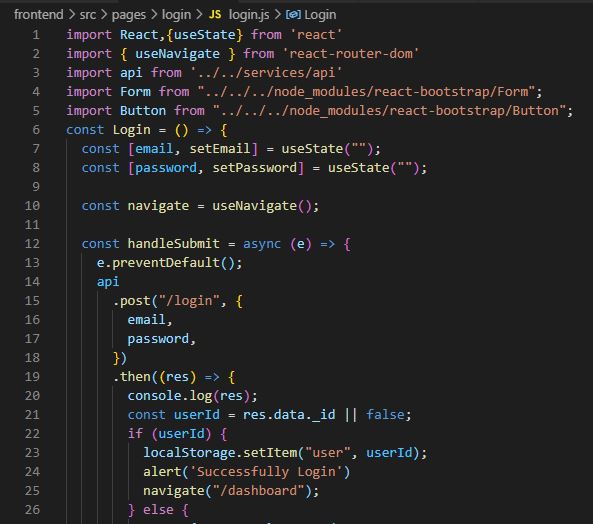


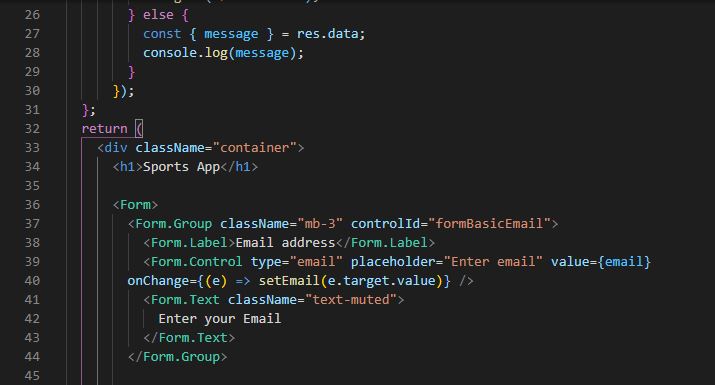
After use axios we are connect to backend and use the create method we are go the port:8000 and process the data from backend part.

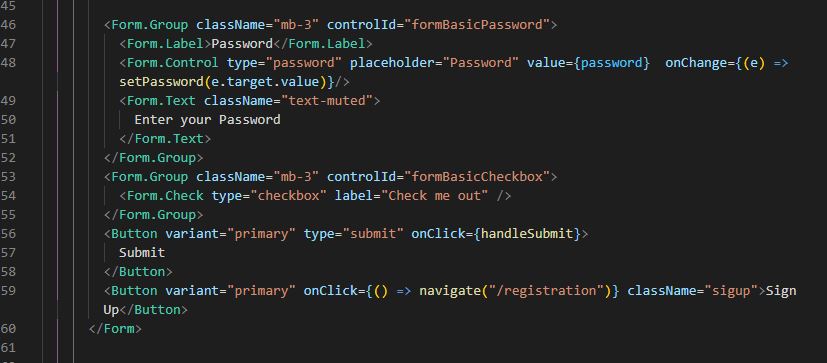
After that go to in Src folder and go to login folder and open the login.js file

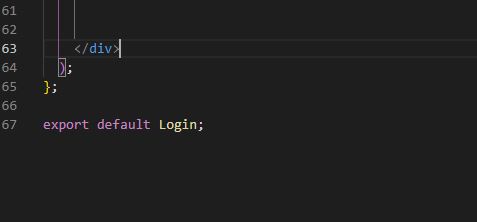
Write the login code that are use for create a login page.

**Login.js**

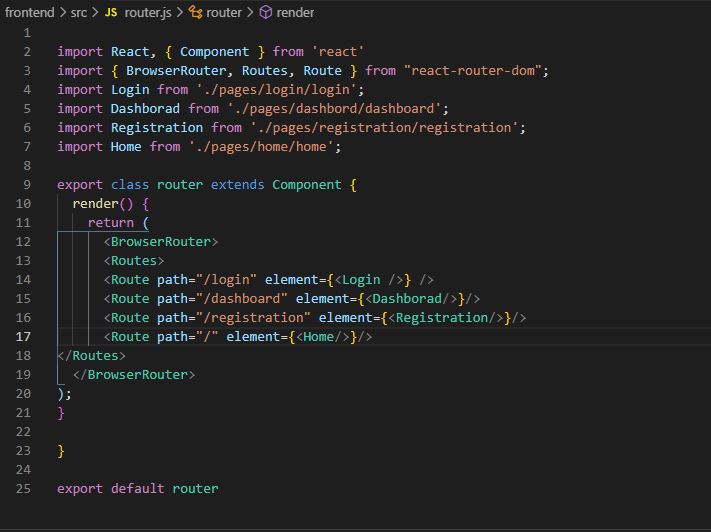








**Add login path into router.js for parfect routing.**



**Login functionality**

* In login.js file we are use Hooks. Hooks is a spacel function that lets you where you are hooks into. useState is a hooks that add react state to a functional component.After that we are use **handelSubmit** and **onChange** funcanality that **onChange** use for In input tag , we are basically pass another property . And use the **handelSubmit** property we are submit the data value .After that it Use API to send value to the server and check against the server .

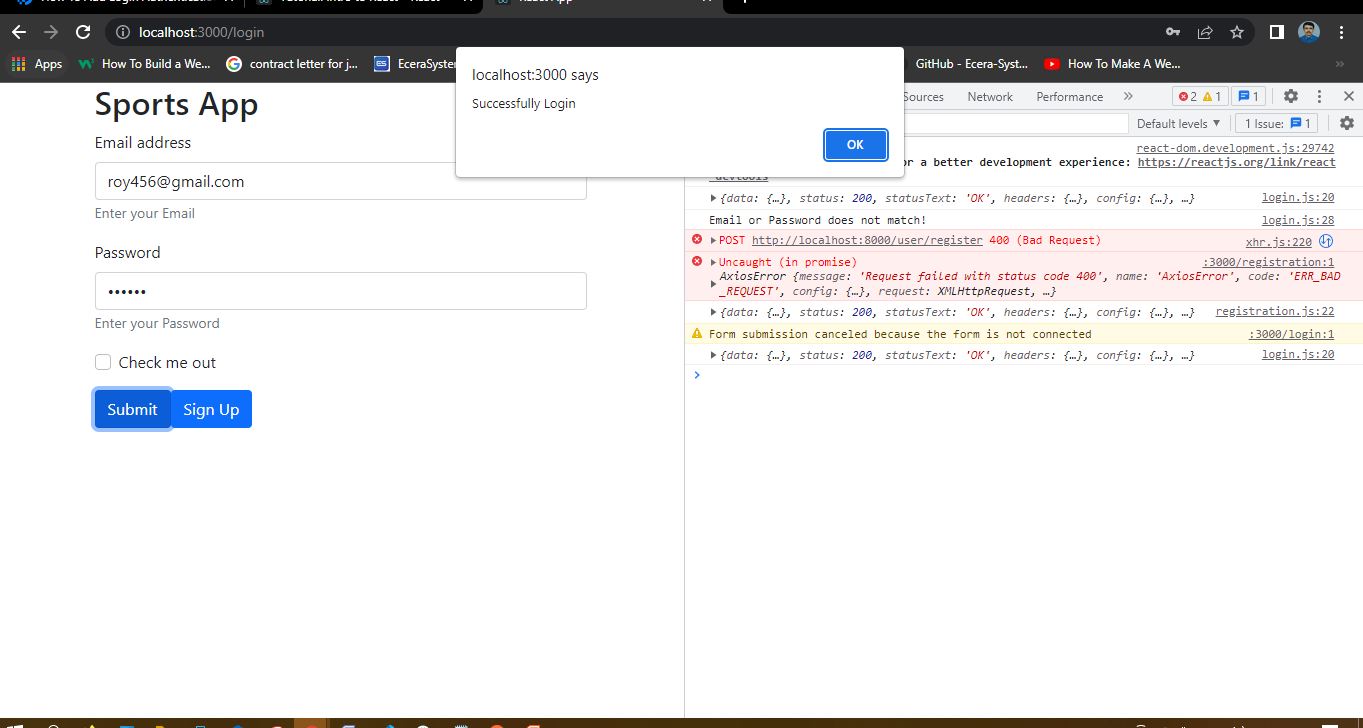
**Check the login**

Now ,it’s time to run code by server and test it . So , we will follow the following steps –

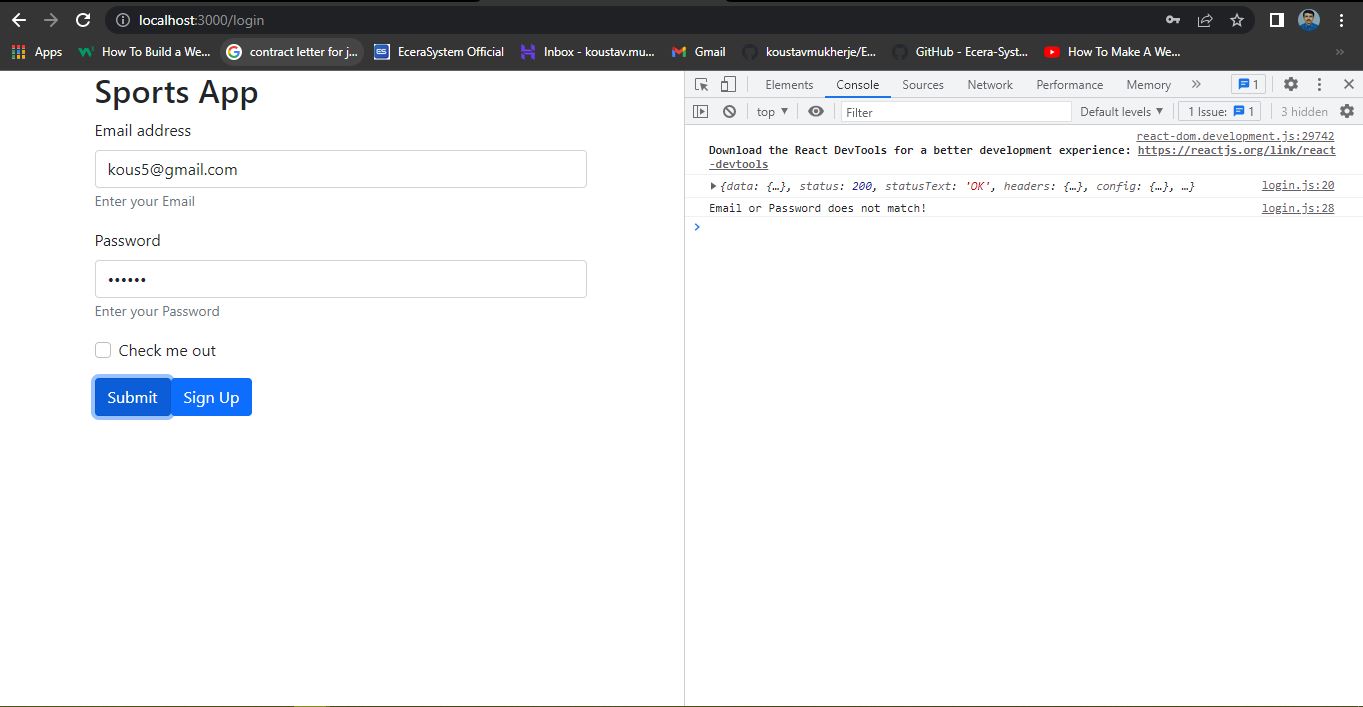
**Step 1** - Open Insomnia or any other API that you are using previously . Here I am using Insomnia . From getUserById request, I copy EmailID and I hope you remember the password that you write while creating new user . So go on frontend UI and paste email ID, type password and click on Submit button .

**Step 2** - After Clicking on Submit button , you will redirect on Dashboard page . If you type wrong mailId and password it will give you error message and you can see Network (Ctrl + Shift + I ).

Successful login



Unsuccessful login wrong password

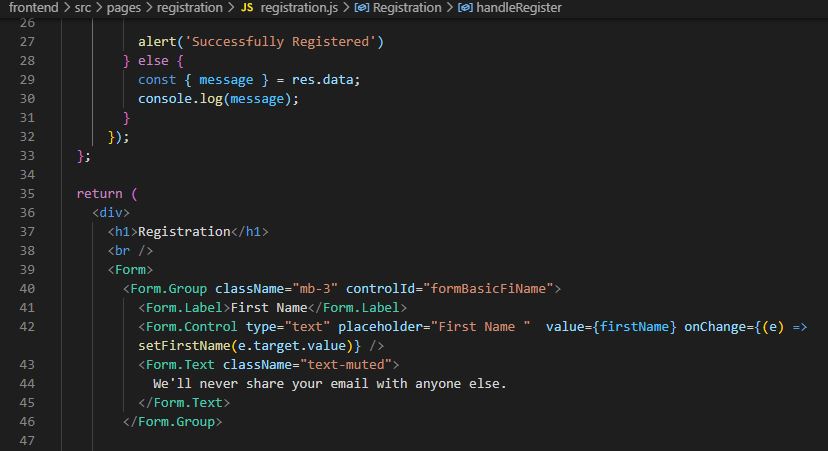


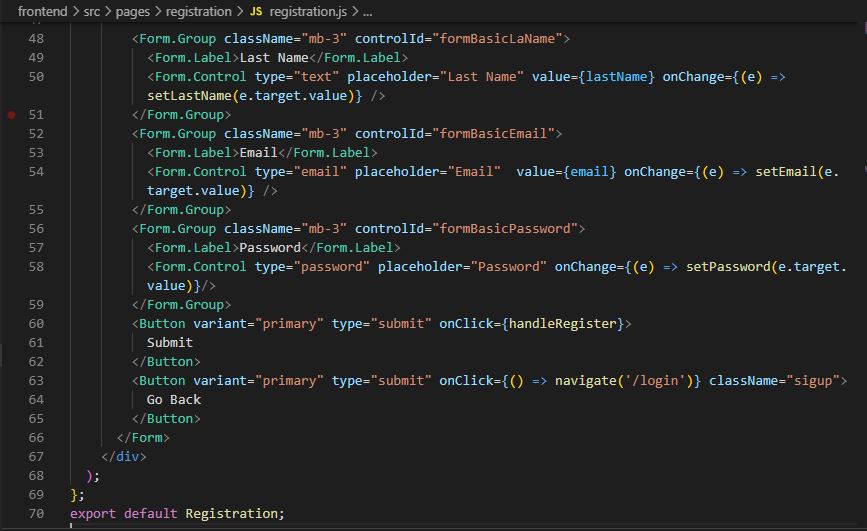
**Registration**

**After the completing the login page we are create the registration page for registration a user . for registration go to Src folder and then go pages folder into the pages folder we are create an folder registration after the create registration folder create registration.js file into the registration folder.**

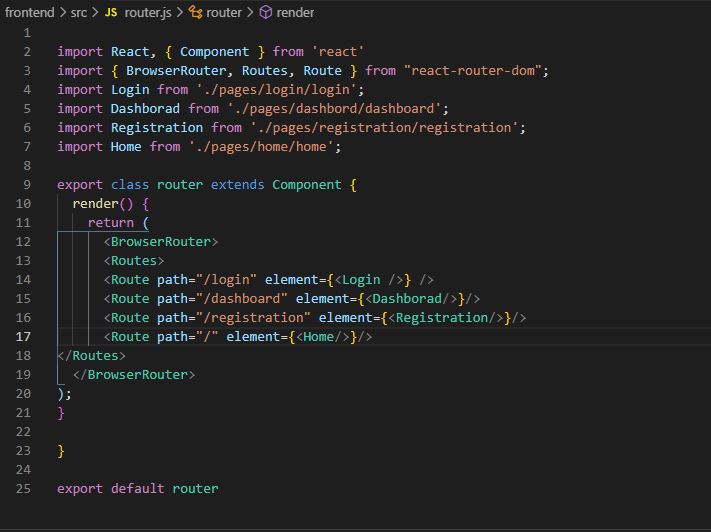
**Src > pages > registration > registration.js**

****





After that add registration routes paths on router.js file.

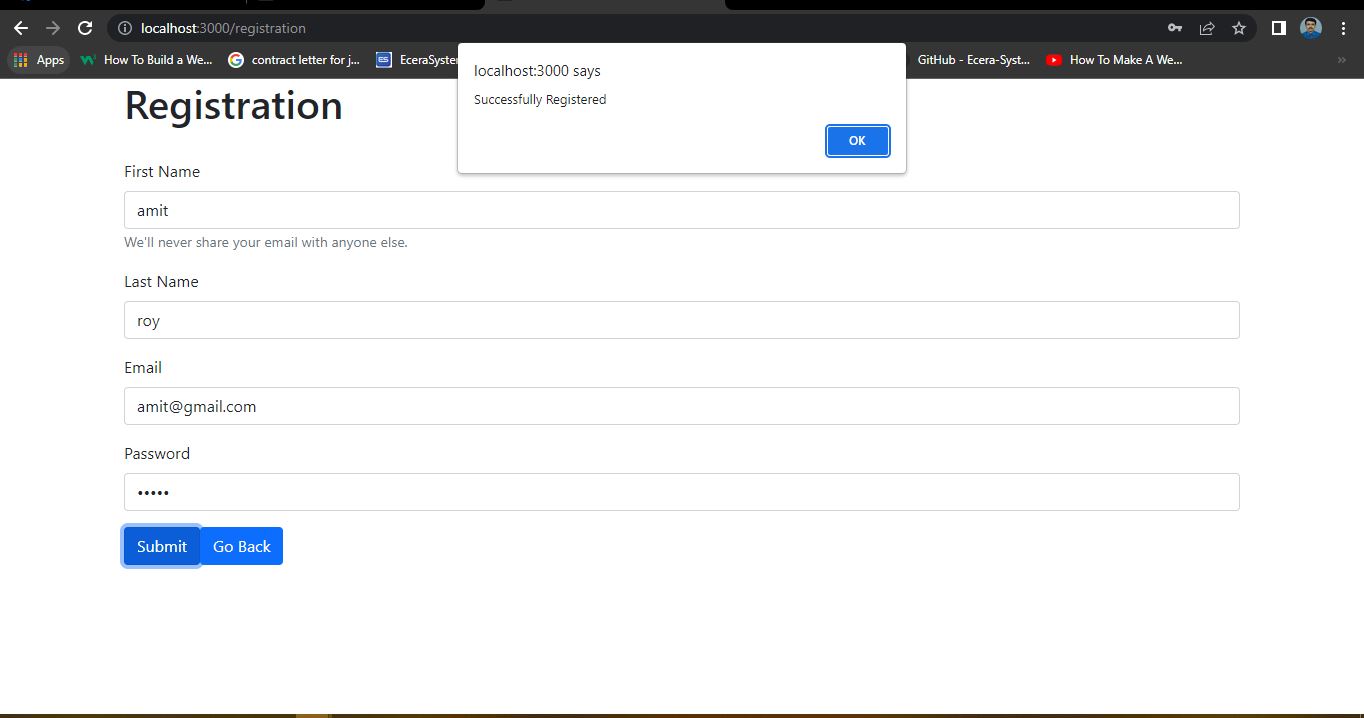


**Check the Registration page**

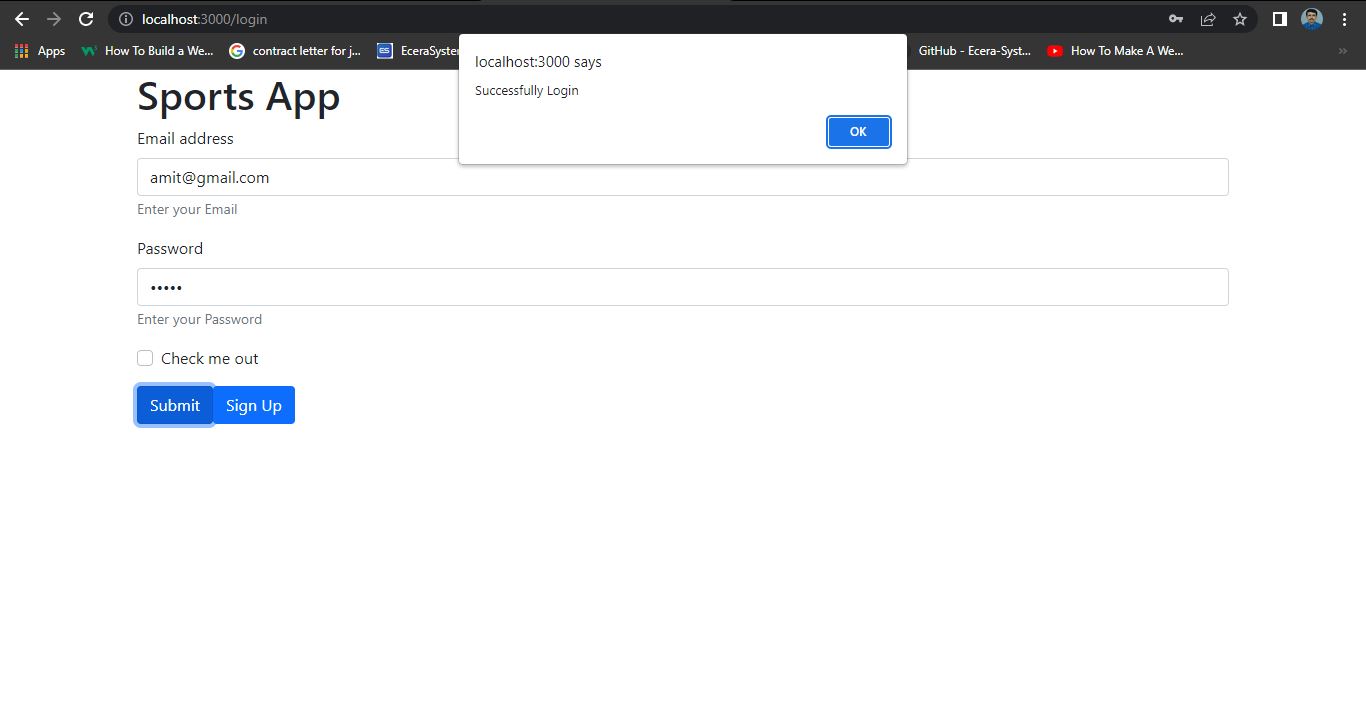
After all this , we will test the register page is working or not . So on URL path we type “localhost:3000/registration” .

After the successful registration we go to login page and after login its shows login successful and show the dashboard.

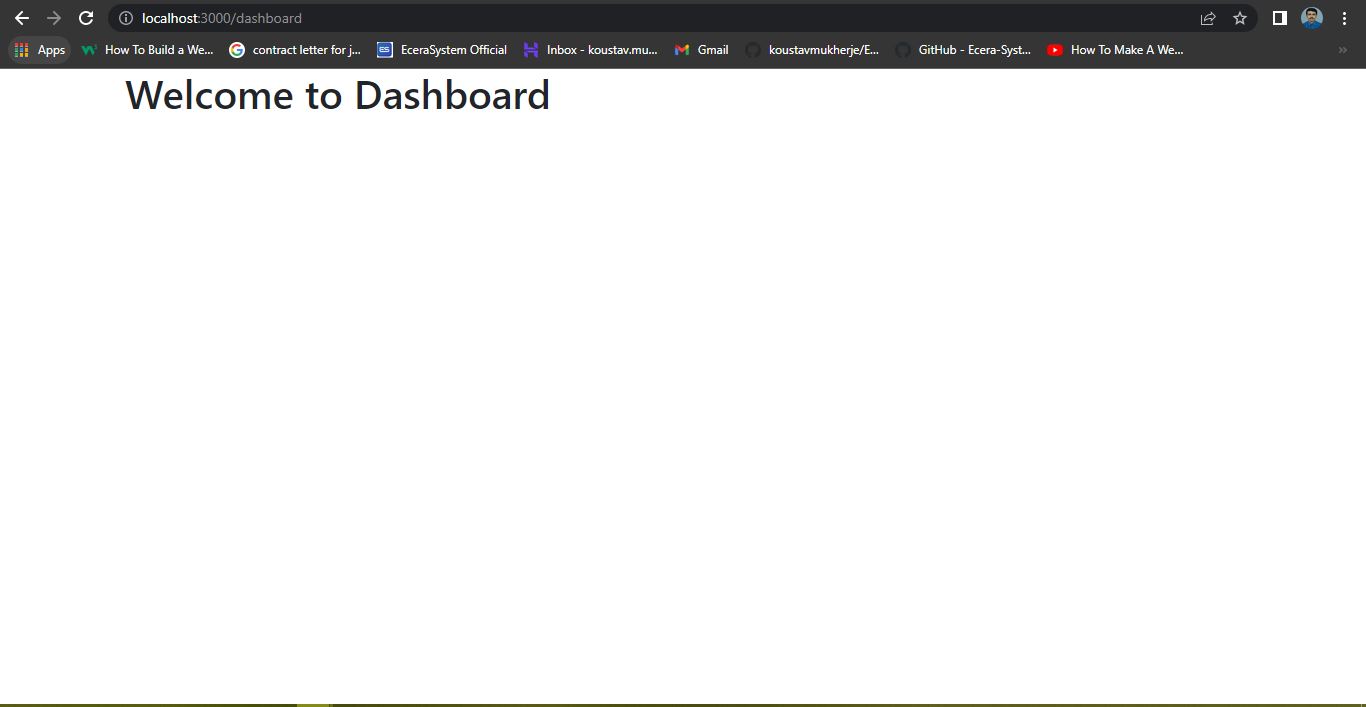
**Registration a user successful**



**After registration login**



**After successful login dashboard**



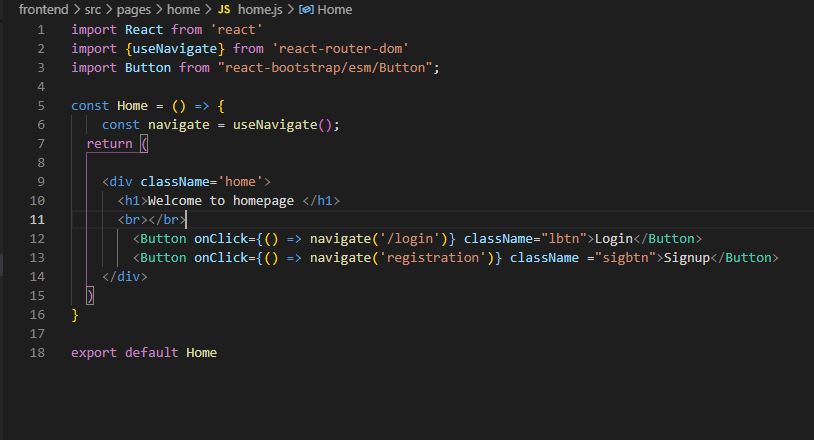
**Home.js**

After we create home.js for connecting all component like a home page. Use of **navigate** function we are navigate and each page and back to each page.

For create home go to **Src** folder then **pages** folder and create another folder “**home”** and into the home create a file **home.js**.

Following the path **Src > pages > home > home.js**

**Home.js**

 browser home page

