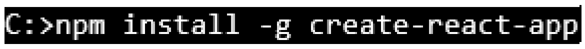
Exercise1

Create a new React Application with the name “myfirstreact”, Run the application to print “welcome to the first session of React” as heading of that page.

1. To create a new React app, Install Nodejs and Npm from the following link:

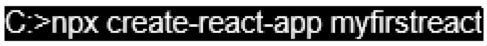
https://nodejs.org/en/download/

2. Install Create-react-app by running the following command in the command prompt:



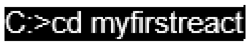
```
C:>npm install -g create-react-app
```

3. To create a React Application with the name of “myfirstreact”, type the following command:



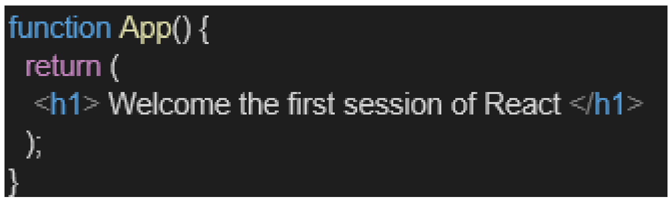
```
C:>npx create-react-app myfirstreact
```

4. Once the App is created, navigate into the folder of myfirstreact by typing the following command:



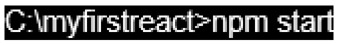
```
C:>cd myfirstreact
```

5. Open the folder of myfirstreact in Visual Studio Code
6. Open the App.js file in Src Folder of myfirstreact
7. Remove the current content of “App.js”
8. Replace it with the following:



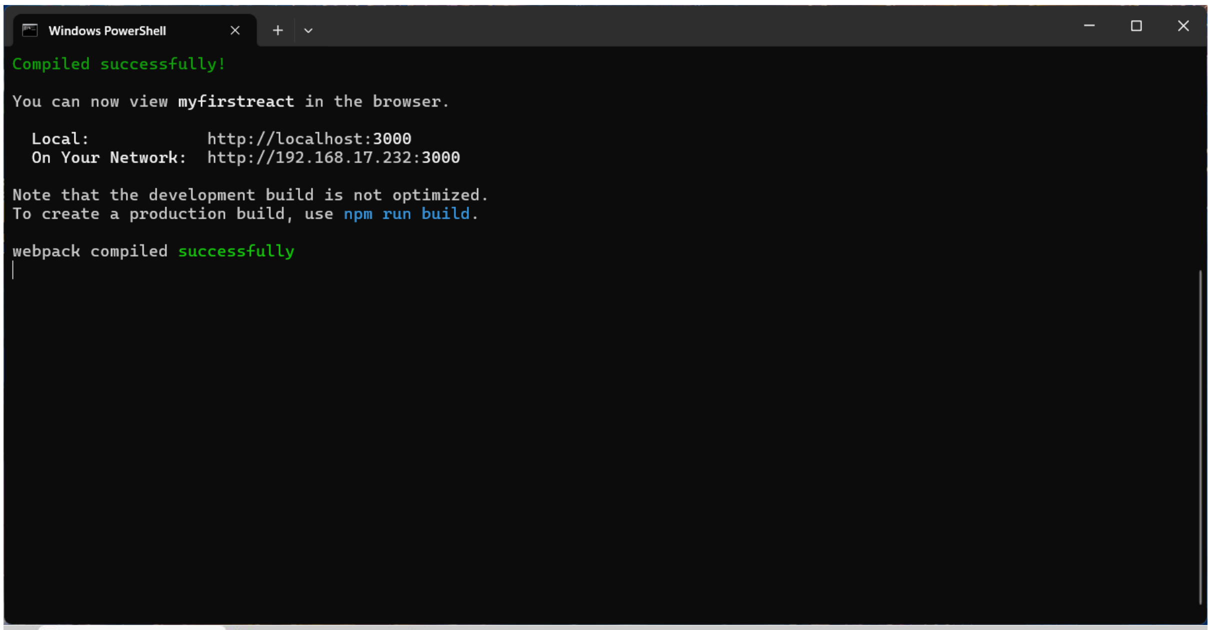
```
function App() {
  return (
   <h1> Welcome the first session of React </h1>
  );
}
```

9. Run the following command to execute the React application:



```
C:\myfirstreact>npm start
```

10. Open a new browser window and type “localhost:3000” in the address bar

11. 

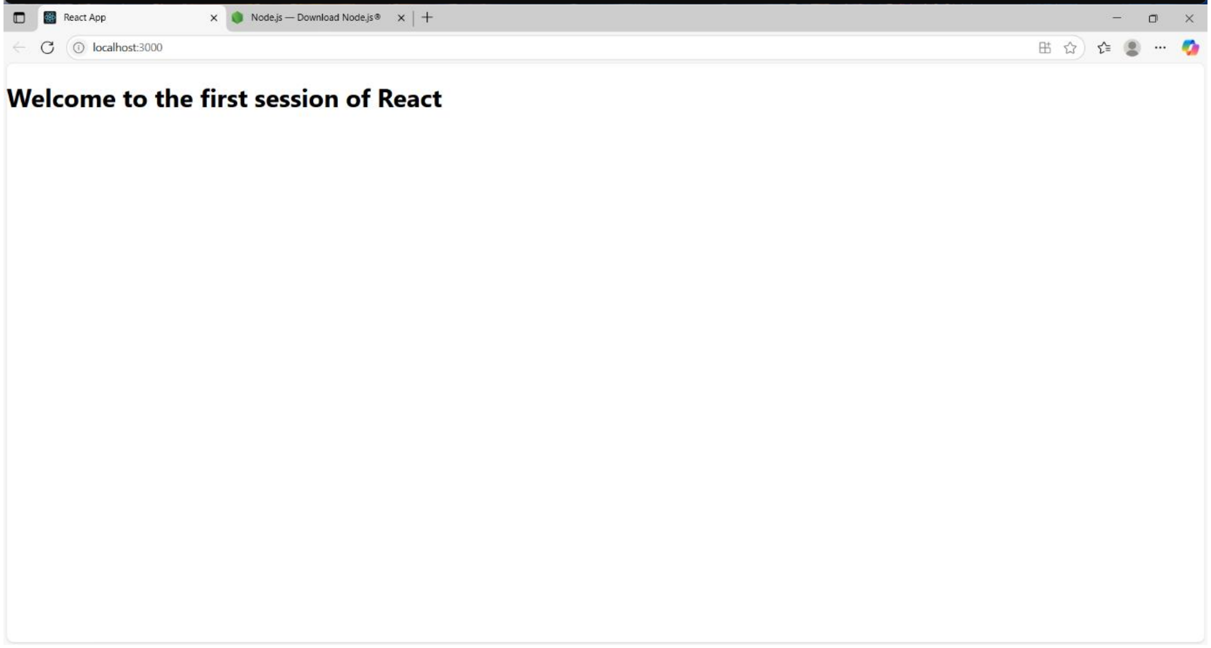
```
Compiled successfully!

You can now view myfirstreact in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.17.232:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

12. 

Welcome to the first session of React

Exercise 2:

Create a react app for Student Management Portal named StudentApp and create a component named Home which will display the Message “Welcome to the Home page of Student Management Portal”. Create another component named About and display the Message “Welcome to the About page of the Student Management Portal”. Create a third component named Contact and display the Message “Welcome to the Contact page of the Student Management Portal”. Call all the three components.

1. Create a React project named “StudentApp” type the following command in terminal of Visual studio:

2. Create a new folder under Src folder with the name “Components”. Add a new file named “Home.js”

3. Type the following code in Home.js

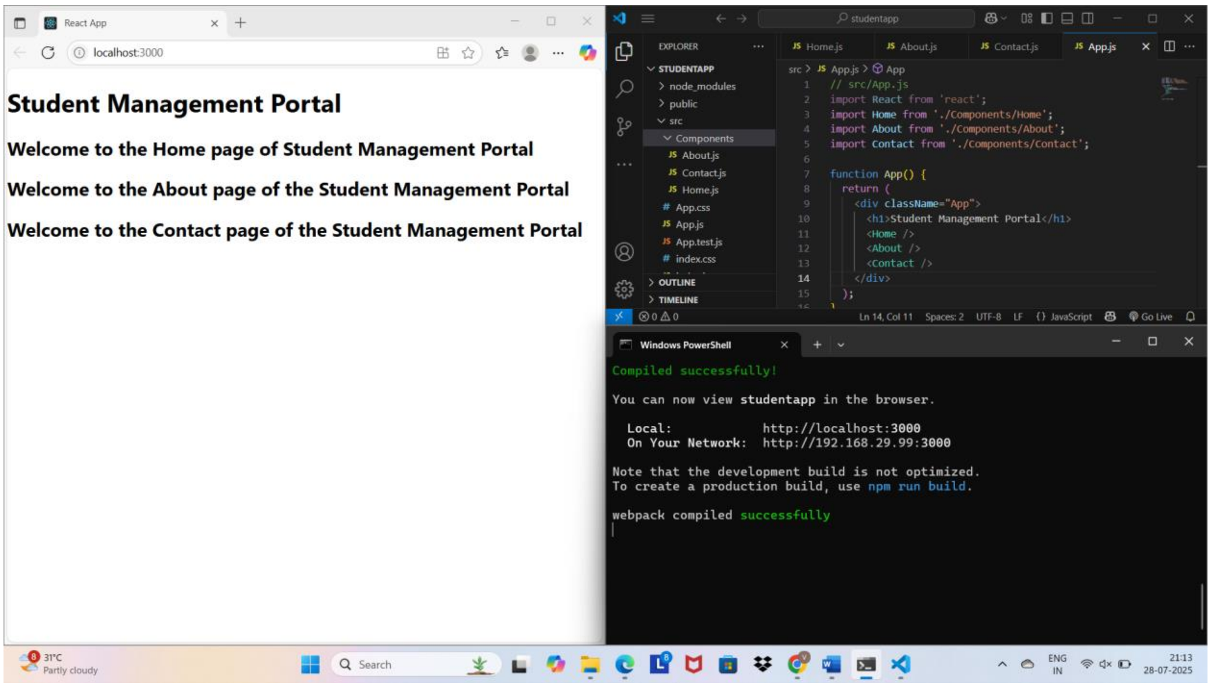
4. Under Src folder add another file named “About.js”

5. Repeat the same steps for Creating “About” and “Contact” component by adding a new file as ”About.js”, “Contact.js” under “Src” folder and edit the code as mentioned for “Home” Component.

6. Edit the App.js to invoke the Home, About and Contact component as follows:

7. In command Prompt, navigate into StudentApp and execute the code by typing the following command:

8. Open browser and type “localhost:3000” in the address bar:



Student Management Portal

Welcome to the Home page of Student Management Portal

Welcome to the About page of the Student Management Portal

Welcome to the Contact page of the Student Management Portal

```javascript
// src/App.js
import React from 'react';
import Home from './Components/Home';
import About from './Components/About';
import Contact from './Components/Contact';

function App() {
  return (
    <div className="App">
      <h1>Student Management Portal</h1>
      <Home />
      <About />
      <Contact />
    </div>
  );
```

```
Compiled successfully!

You can now view studentapp in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.29.99:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

Exercise 3:

Create a react app for Student Management Portal named scorecalculatorapp and create a function component named “CalculateScore” which will accept Name, School, Total and goal in order to calculate the average score of a student and display the same.

1.Create a React project named “scorecalculatorapp” type the following command in terminal of Visual studio:

2.Create a new folder under Src folder with the name “Components”. Add a new file named “CalculateScore.js”

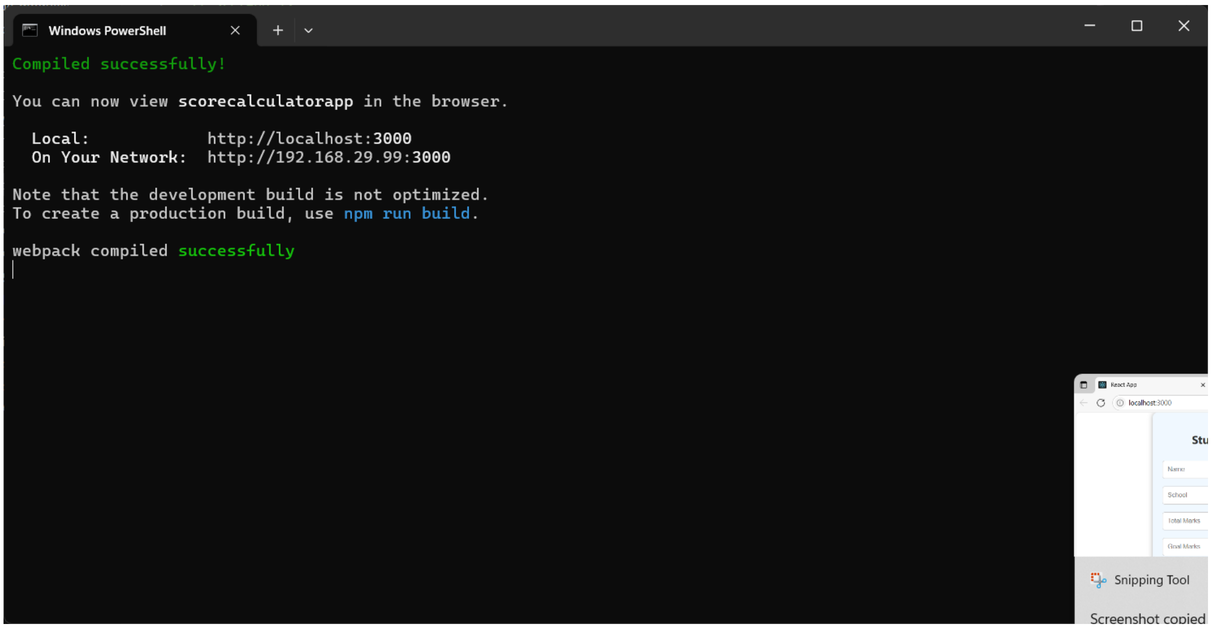
3.Type the following code in CalculateScore.js

4.Create a Folder named Stylesheets and add a file named “mystyle.css” in order to add some styles to the components:

5.Edit the App.js to invoke the CalculateScore functional component as follows:

6.In command Prompt, navigate into scorecalculatorapp and execute the code by typing the following command:

7.Open browser and type “localhost:3000” in the address bar:



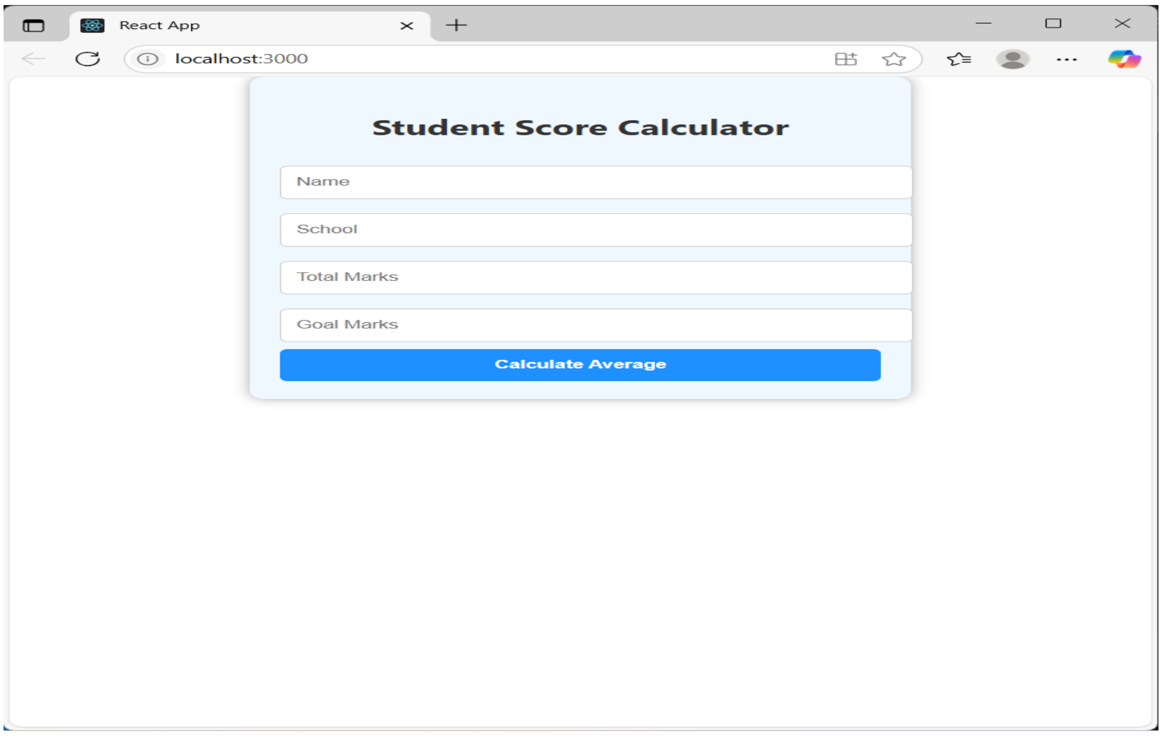
```
Compiled successfully!

You can now view scorecalculatorapp in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.29.99:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```



Student Score Calculator

Name

School

Total Marks

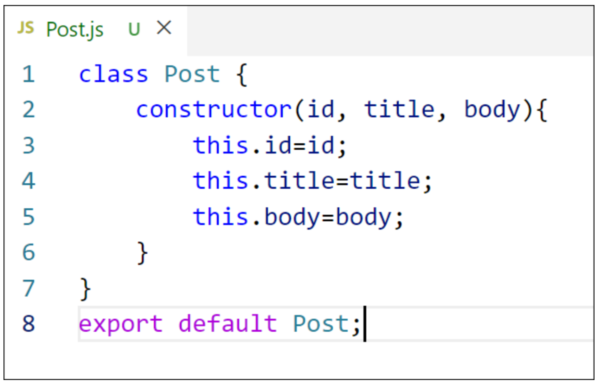
Goal Marks

Calculate Average

Exercise 4:

npx create-react-app blogapp

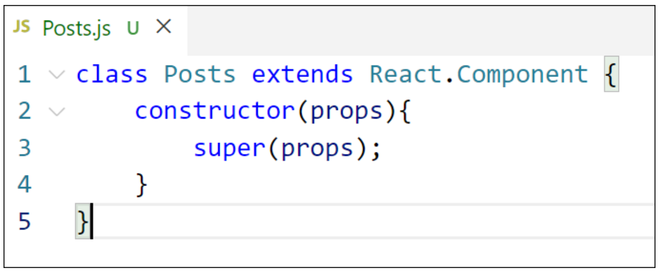
1. Create a new react application using *create-react-app* tool with the name as “blogapp”
2. Open the application using VS Code
3. Create a new file named as Post.js in src folder with following properties



```javascript
class Post {
    constructor(id, title, body){
        this.id=id;
        this.title=title;
        this.body=body;
    }
}
export default Post;
```

Figure 1: Post class

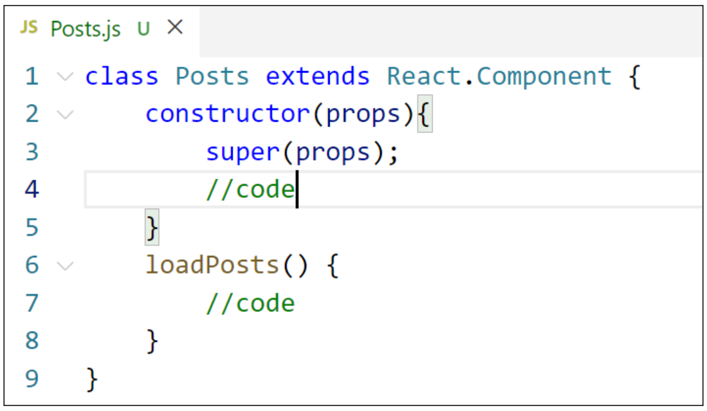
4. Create a new class based component named as Posts inside Posts.js file



```javascript
class Posts extends React.Component {
    constructor(props){
        super(props);
    }
}
```

Figure 2: Posts Component

5. Initialize the component with a list of Post in state of the component using the constructor
6. Create a new method in component with the name as loadPosts() which will be responsible for using Fetch API and assign it to the component state created earlier. To get the posts use the url (https://jsonplaceholder.typicode.com/posts)

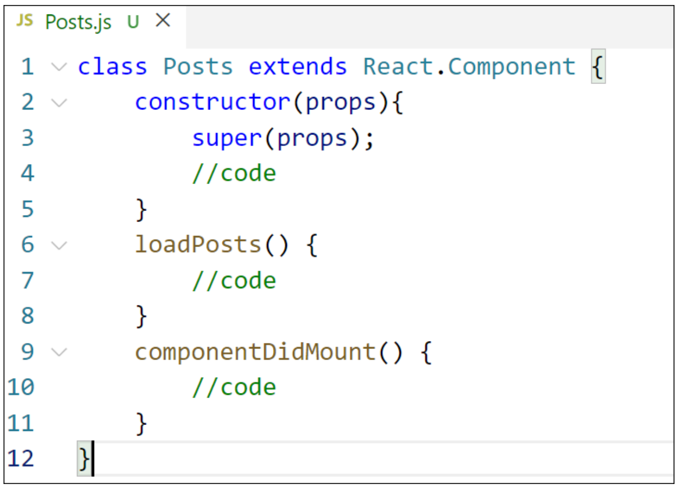


Posts.js U ×

```
class Posts extends React.Component {
    constructor(props){
        super(props);
        //code
    }
    loadPosts() {
        //code
    }
}
```

Figure 3: loadPosts() method

7. Implement the componentDidMount() hook to make calls to loadPosts() which will fetch the posts

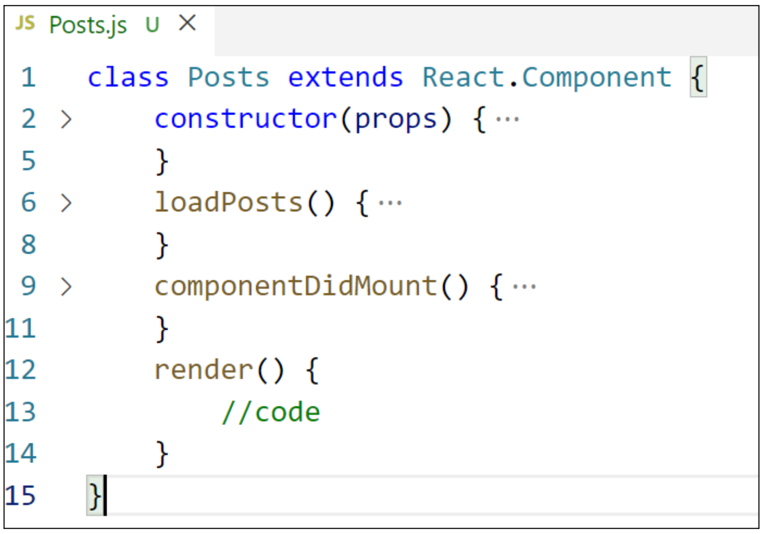


Posts.js U ×

```
class Posts extends React.Component {
    constructor(props){
        super(props);
        //code
    }
    loadPosts() {
        //code
    }
    componentDidMount() {
        //code
    }
}
```

Figure 4: componentDidMount() hook

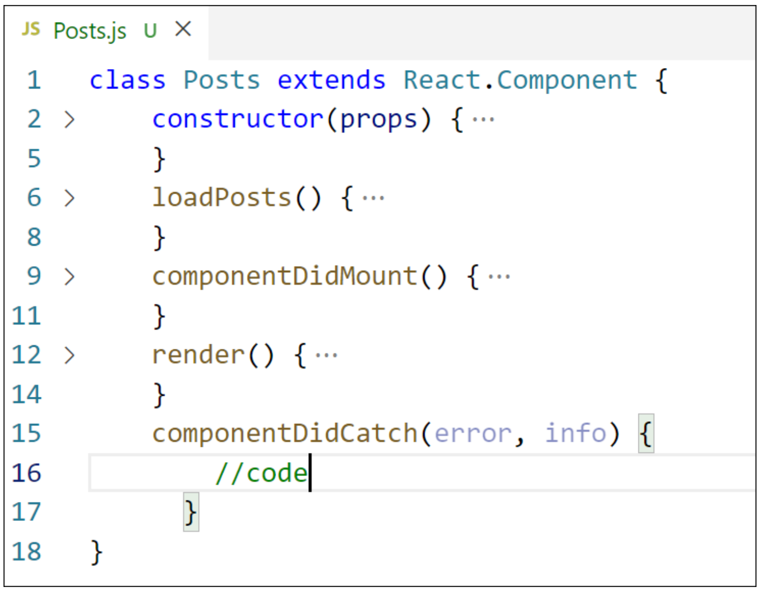
8. Implement the render() which will display the title and post of posts in html page using heading and paragraphs respectively.



```
Posts.js  U  ×
class Posts extends React.Component {
    constructor(props) {···
    }
    loadPosts() {···
    }
    componentDidMount() {···
    }
    render() {
        //code
    }
}
```

Figure 5: render() method

9. Define a componentDidCatch() method which will be responsible for displaying any error happing in the component as alert messages.

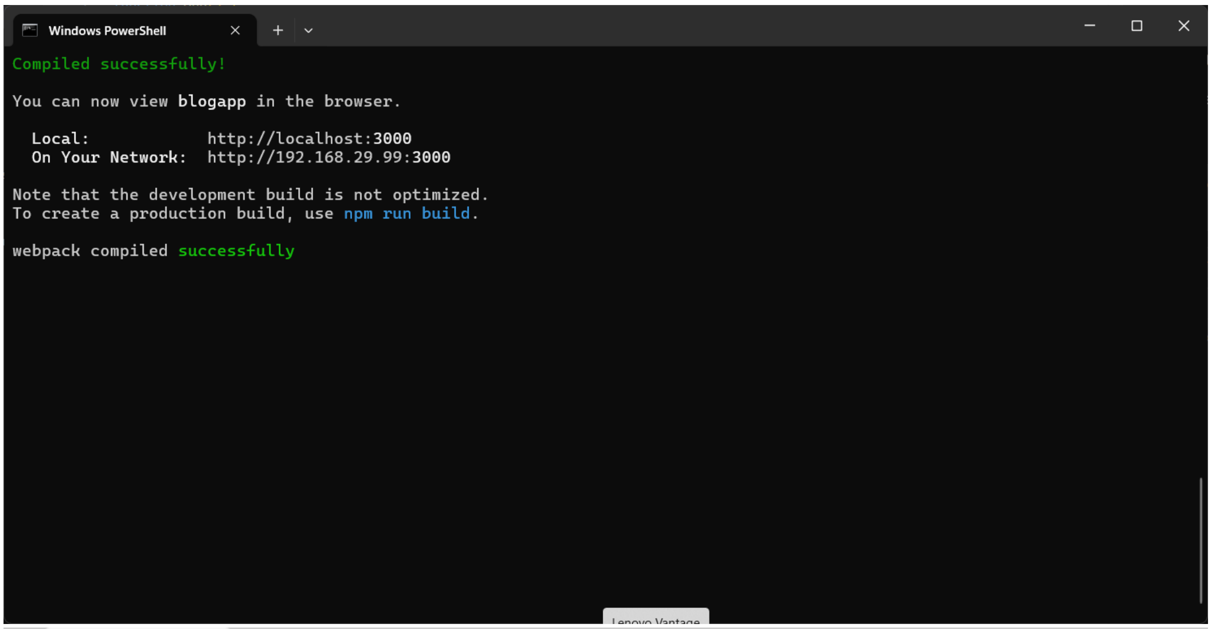


```
Posts.js  U  ×
class Posts extends React.Component {
    constructor(props) {···
    }
    loadPosts() {···
    }
    componentDidMount() {···
    }
    render() {···
    }
    componentDidCatch(error, info) {
        //code
      }
}
```

Figure 6: componentDidCatch() hook

10. Add the Posts component to App component.

Build and Run the application



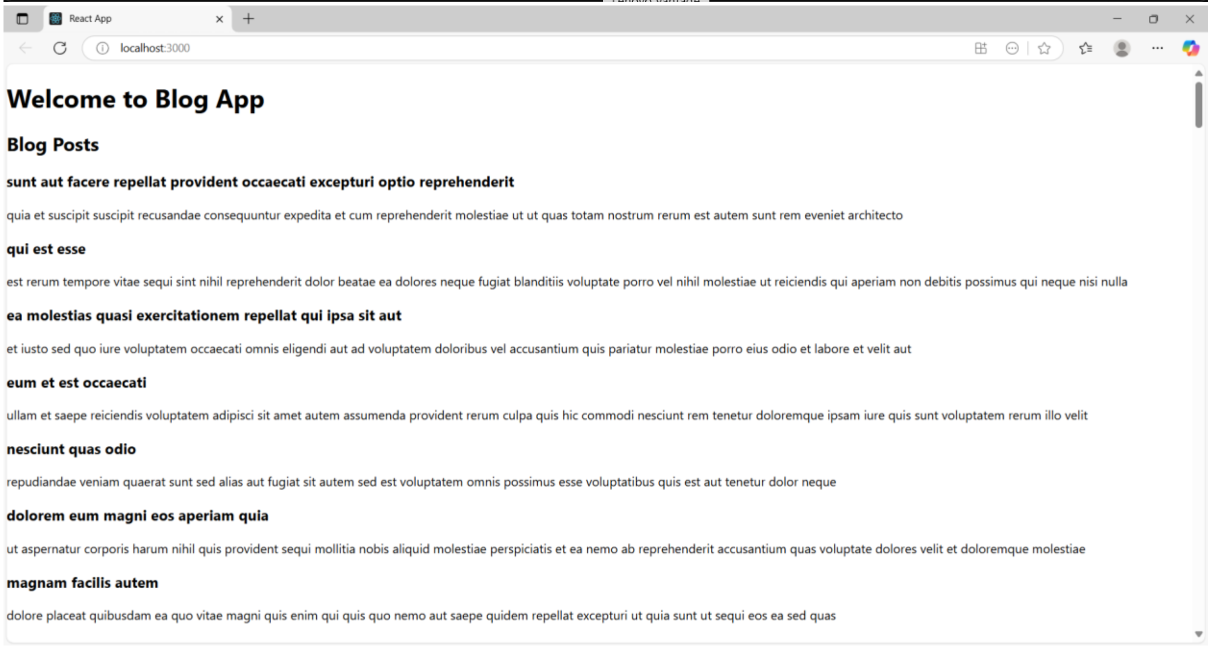
```
Compiled successfully!

You can now view blogapp in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.29.99:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```



Welcome to Blog App

Blog Posts

sunt aut facere repellat provident occaecati excepturi optio reprehenderit

quia et suscipit suscipit recusandae consequuntur expedita et cum reprehenderit molestiae ut ut quas totam nostrum rerum est autem sunt rem eveniet architecto

qui est esse

est rerum tempore vitae sequi sint nihil reprehenderit dolor beatae ea dolores neque fugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis qui aperiam non debitis possimus qui neque nisi nulla

ea molestias quasi exercitationem repellat qui ipsa sit aut

et iusto sed quo iure voluptatem occaecati omnis eligendi aut ad voluptatem doloribus vel accusantium quis pariatur molestiae porro eius odio et labore et velit aut

eum et est occaecati

ullam et saepe reiciendis voluptatem adipisci sit amet autem assumenda provident rerum culpa quis hic commodi nesciunt rem tenetur doloremque ipsam iure quis sunt voluptatem rerum illo velit

nesciunt quas odio

repudiandae veniam quaerat sunt sed alias aut fugiat sit autem sed est voluptatem omnis possimus esse voluptatibus quis est aut tenetur dolor neque

dolorem eum magni eos aperiam quia

ut aspernatur corporis harum nihil quis provident sequi mollitia nobis aliquid molestiae perspiciatis et ea nemo ab reprehenderit accusantium quas voluptate dolores velit et doloremque molestiae

magnam facilis autem

dolore placeat quibusdam ea quo vitae magni quis enim qui quis quo nemo aut saepe quidem repellat excepturi ut quia sunt ut sequi eos ea sed quas

Example 5:

My Academy team at Cognizant want to create a dashboard containing the details of ongoing and completed cohorts. A react application is created which displays the detail of the cohorts using react component. You are assigned the task of styling these react components.

Download and build the attached react application.

1. Unzip the react application in a folder
2. Open command prompt and switch to the react application folder
3. Restore the node packages using the following commands
4. Open the application using VS Code
5. Create a new CSS Module in a file called “CohortDetails.module.css”

6. Define a css class with the name as “box” with following properties

Width = 300px;

Display = inline block;

Overall 10px margin

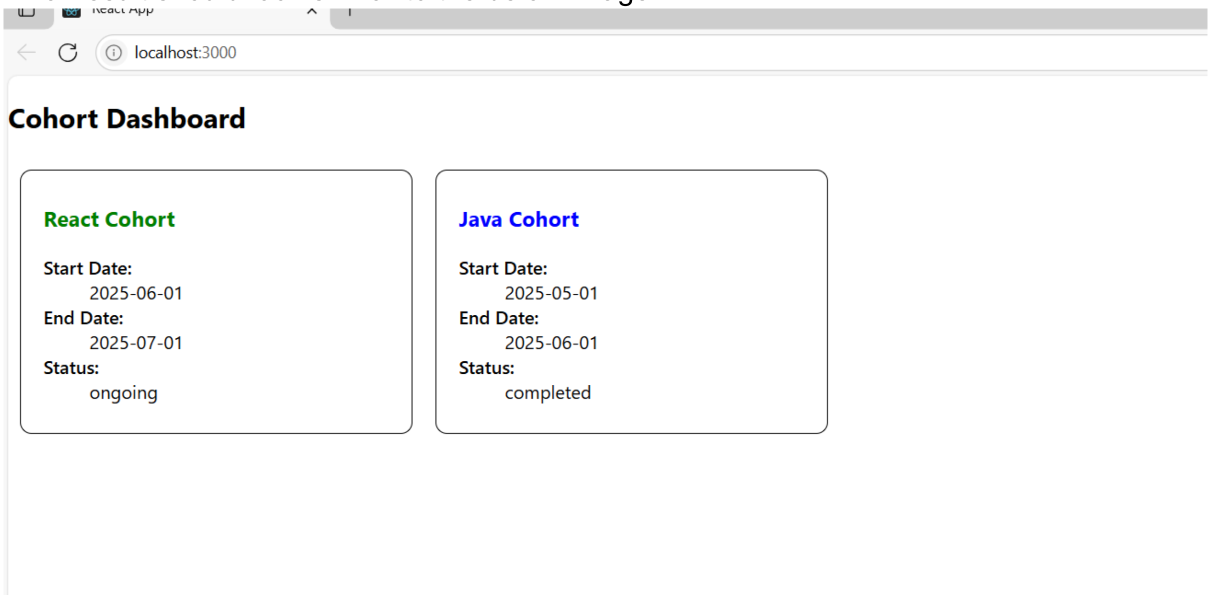
Top and bottom padding as 10px

Left and right padding as 20px

1 px border in black color

A border radius of 10px

7. Define a css style for html <dt> element using tag selector. Set the font weight to 500.
8. Open the cohort details component and import the CSS Module
9. Apply the box class to the container div
10. Define the style for <h3> element to use “green” color font when cohort status is “ongoing” and “blue” color in all other scenarios.
11. Final result should look similar to the below image



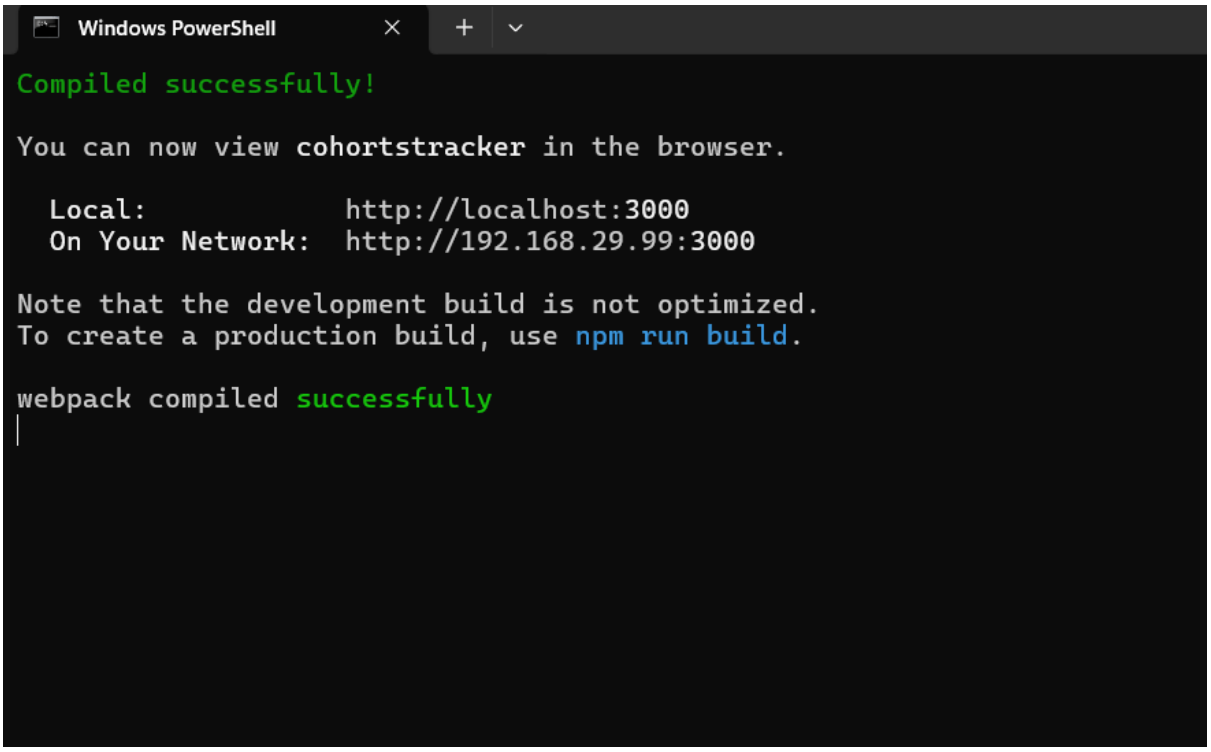
Cohort Dashboard

React Cohort

Start Date:
2025-06-01
End Date:
2025-07-01
Status:
ongoing

Java Cohort

Start Date:
2025-05-01
End Date:
2025-06-01
Status:
completed



```
Compiled successfully!

You can now view cohortstracker in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.29.99:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

Example 6:

Cognizant Academy teams want to maintain a list of trainers along with their expertise in a SPA using React as the technology. You are assigned the task of creating this React app.

The following trainers' data application will deal.

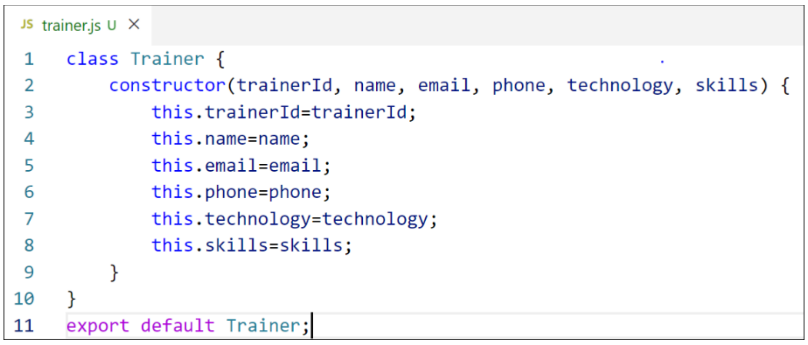
1. T-ID
2. Name
3. Phone
4. Email
5. Stream
6. Skills

1. Create a new React app using *create-react-app* tool with the as “TrainersApp”
2. Open the application using the VS Code
3. Add a new file called *trainer.js* inside the **src folder** and define a class named as “Trainer” with the following properties
   a. TrainerId
   b. Name
   c. Email

d. Phone

e. Technology

f. Skills

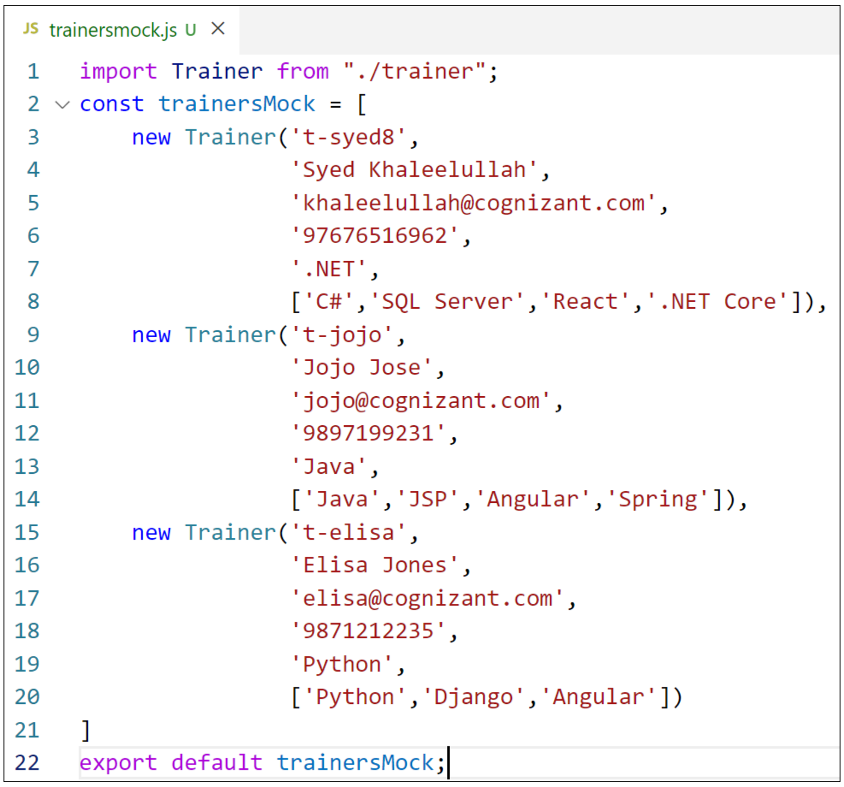


trainer.js U

```javascript
class Trainer {
    constructor(trainerId, name, email, phone, technology, skills) {
        this.trainerId=trainerId;
        this.name=name;
        this.email=email;
        this.phone=phone;
        this.technology=technology;
        this.skills=skills;
    }
}
export default Trainer;
```

Figure 7: Trainer.js

4. Create a new TrainersMock.js file which will contain the mock trainer data. Refer the following screenshot for mock data

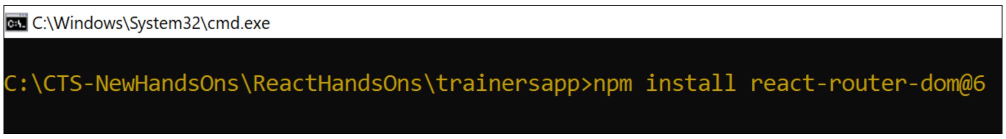


trainersmock.js U

```javascript
import Trainer from "./trainer";
const trainersMock = [
    new Trainer('t-syed8',
                'Syed Khaleelullah',
                'khaleelullah@cognizant.com',
                '97676516962',
                '.NET',
                ['C#','SQL Server','React','.NET Core']),
    new Trainer('t-jojo',
                'Jojo Jose',
                'jojo@cognizant.com',
                '9897199231',
                'Java',
                ['Java','JSP','Angular','Spring']),
    new Trainer('t-elisa',
                'Elisa Jones',
                'elisa@cognizant.com',
                '9871212235',
                'Python',
                ['Python','Django','Angular'])
]
export default trainersMock;
```

Figure 8: TrainersMock.js

5. Install the support for React router for the dom. Execute the following command.



C:\Windows\System32\cmd.exe

```
C:\CTS-NewHandsOns\ReactHandsOns\trainersapp>npm install react-router-dom@6
```

Figure 9: Install React Router

6. Create new component named as **TrainersList** inside *Trainerlist.js* file. The component should accept the trainer's data as parameter and render it as a list. The list should display names of each trainers which must be clickable like a hyper link. Refer the following screenshot for the component layout.
7. Create a new component named as Home inside Home.js which will be responsible for displaying the following
8. Modify the App component to add support for routing and defining the navigation links to Home component and TrainersList component. Use BrowserRouter, Routes, Route and Link components from the react-router-dom library.

Define the following URL

1. / - must redirect to home component
2. /trainers – must redirect to trainers list component.

The layout of the page must be similar to the following

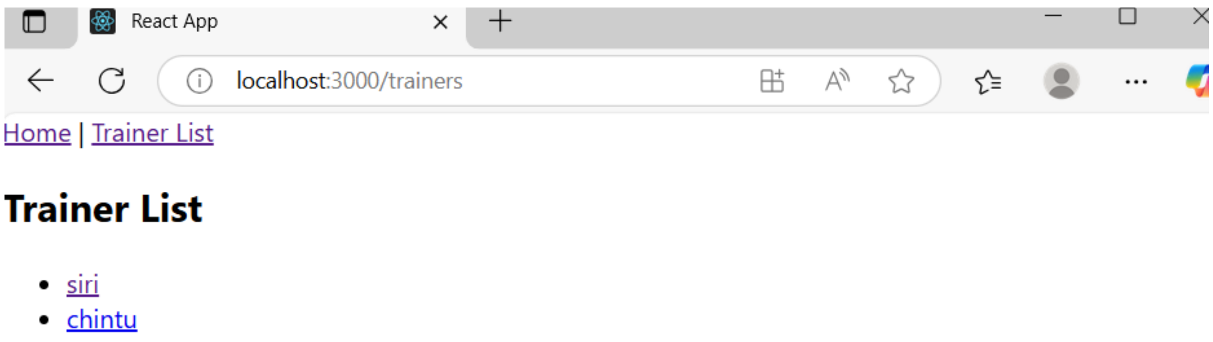
9. Create a new component named **TrainerDetail** in *TrainerDetails.js* file.

The component should retrieve a parameter named id from the URL with the help of "useParams" hook from the React router DOM library.

It should query the mock trainer data using the id and display the trainer details as show in screenshot.

Modify the TrainersList component to add Links to TrainerDetail component while passing the ID. Define a route in App component for the same.

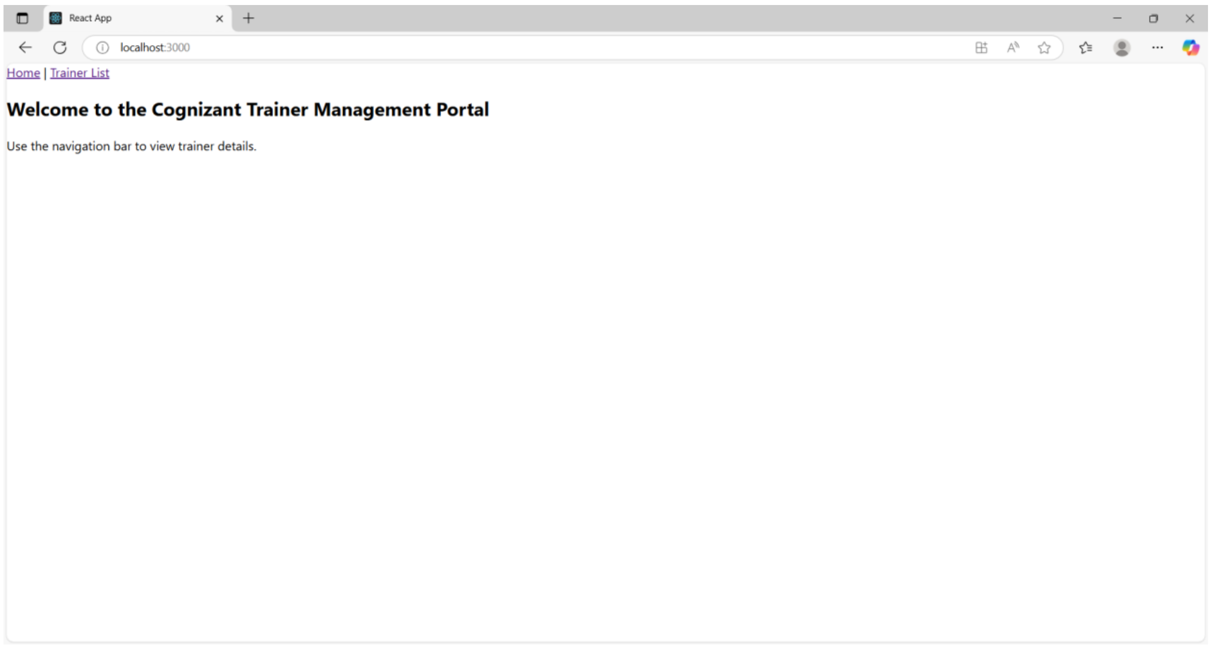
10. Build and run the application. The complete layout of the application will look as follows.



Home | Trainer List

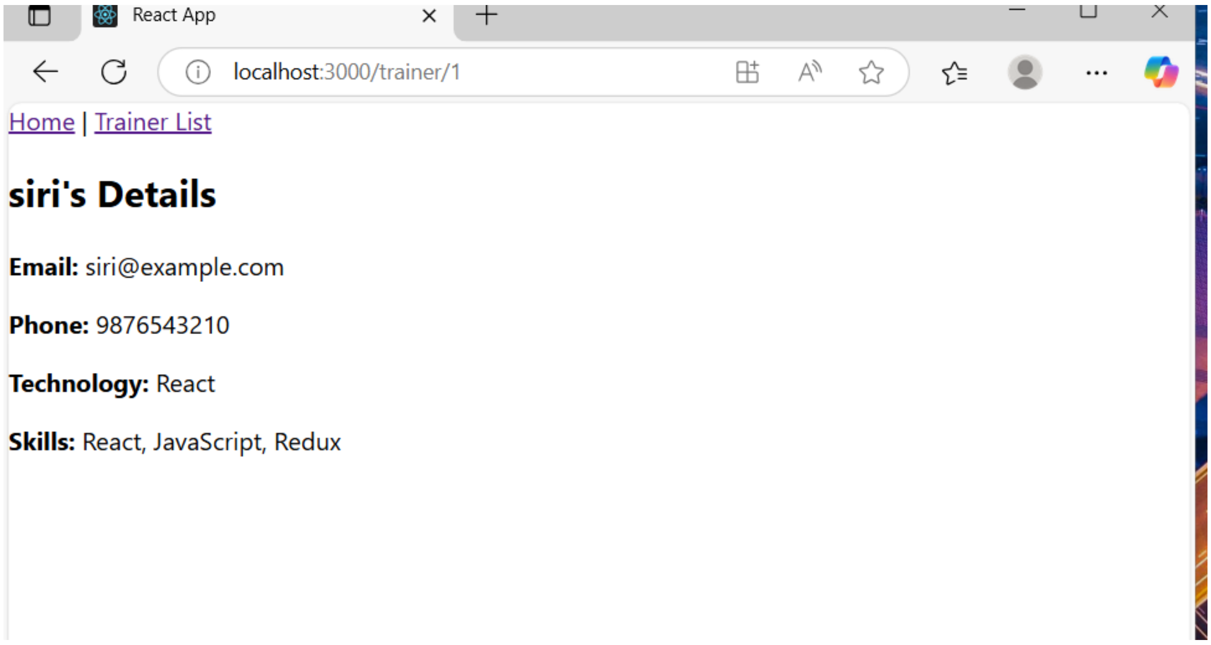
Trainer List

- siri
- chintu



Welcome to the Cognizant Trainer Management Portal

Use the navigation bar to view trainer details.



siri's Details

Email: siri@example.com

Phone: 9876543210

Technology: React

Skills: React, JavaScript, Redux