1. ReactJS-HOL

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”

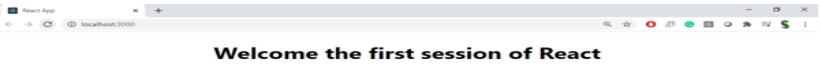
8. Replace it with the following:

```
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 bar1.



SOLUTION

1. Check if NodeJs and npm are installed

```
OUTPUT   DEBUG CONSOLE   POSTMAN CONSOLE   PROBLEMS   TERMINAL   PORTS        powershell - solutions

PS D:\DN-4.0-Java\C3-Products and Frameworks\M10-React\solutions> node -v
v22.11.0
PS D:\DN-4.0-Java\C3-Products and Frameworks\M10-React\solutions> npm -v
10.9.2
PS D:\DN-4.0-Java\C3-Products and Frameworks\M10-React\solutions>
```

2. Installing **create-react-app**

```
PS D:\DN-4.0-Java\C3-Products and Frameworks\M10-React\solutions> npm install -g create-react-app
npm warn deprecated inflight@1.0.6: This module is not supported, and leaks memory. Do not use it. Check out lru-cache if you want a good and tested way to coalesce async re
quests by a key value, which is much more comprehensive and powerful.
npm warn deprecated fstream-ignore@1.0.5: This package is no longer supported.
npm warn deprecated uid-number@0.0.6: This package is no longer supported.
npm warn deprecated rimraf@2.7.1: Rimraf versions prior to v4 are no longer supported
npm warn deprecated glob@7.2.3: Glob versions prior to v9 are no longer supported
npm warn deprecated fstream@1.0.12: This package is no longer supported.
npm warn deprecated tar@2.2.2: This version of tar is no longer supported, and will not receive security updates. Please upgrade asap.

added 64 packages in 29s

4 packages are looking for funding
  run `npm fund` for details
PS D:\DN-4.0-Java\C3-Products and Frameworks\M10-React\solutions>
```

3. Creating a react application with the name **myfirstreact**

```
PS D:\DN-4.0-Java\C3-Products and Frameworks\M10-React\solutions> npx create-react-app myfirstreact
  create-react-app is deprecated.

  You can find a list of up-to-date React frameworks on react.dev
  For more info see:https://react.dev/link/cra

  This error message will only be shown once per install.

  Creating a new React app in D:\DN-4.0-Java\C3-Products and Frameworks\M10-React\solutions\myfirstreact.

  Installing packages. This might take a couple of minutes.
  Installing react, react-dom, and react-scripts with cra-template...


  added 1323 packages in 2m

  269 packages are looking for funding
    run `npm fund` for details

  Installing template dependencies using npm...

  added 18 packages, and changed 1 package in 13s

  269 packages are looking for funding
    run `npm fund` for details
  Removing template package using npm...


  removed 1 package, and audited 1341 packages in 9s

  269 packages are looking for funding
    run `npm fund` for details
```

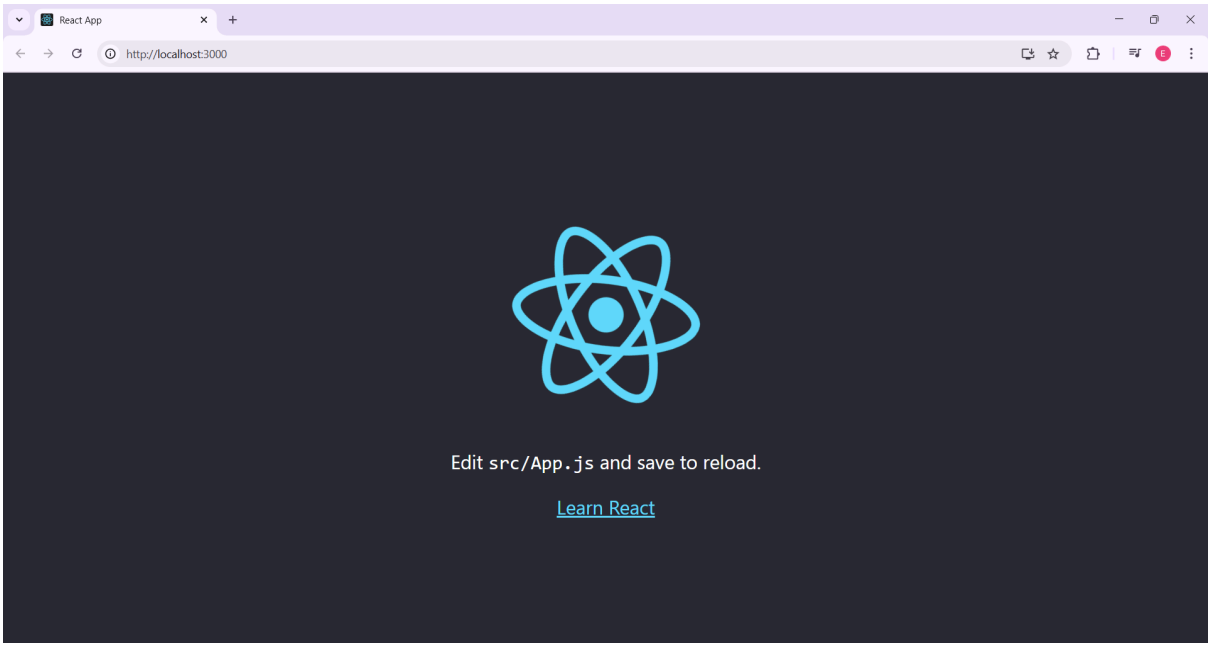
```
  269 packages are looking for funding
    run `npm fund` for details

  9 vulnerabilities (3 moderate, 6 high)

  To address all issues (including breaking changes), run:
    npm audit fix --force

  Run `npm audit` for details.

  Success! Created myfirstreact at D:\DN-4.0-Java\C3-Products and Frameworks\M10-React\solutions\myfirstreact
  Inside that directory, you can run several commands:

    npm start
      Starts the development server.

    npm run build
      Bundles the app into static files for production.

    npm test
      Starts the test runner.

    npm run eject
      Removes this tool and copies build dependencies, configuration files
      and scripts into the app directory. If you do this, you can’t go back!

  We suggest that you begin by typing:

    cd myfirstreact
    npm start

  Happy hacking!
PS D:\DN-4.0-Java\C3-Products and Frameworks\M10-React\solutions>
```

4. Folder structure of our newly created **myfirstreact**

5. Navigate into the folder and start the server

```
PS D:\DN-4.0-Java\C3-Products and Frameworks\M10-React\solutions> cd myfirstreact
PS D:\DN-4.0-Java\C3-Products and Frameworks\M10-React\solutions\myfirstreact> npm start

> myfirstreact@0.1.0 start
> react-scripts start

(node:3468) [DEP_WEBPACK_DEV_SERVER_ON_AFTER_SETUP_MIDDLEWARE] DeprecationWarning: 'onAfterSetupMiddleware' option is deprecated. Please use the 'setupMiddlewares' option
.
(Use `node --trace-deprecation ...` to show where the warning was created)
(node:3468) [DEP_WEBPACK_DEV_SERVER_ON_BEFORE_SETUP_MIDDLEWARE] DeprecationWarning: 'onBeforeSetupMiddleware' option is deprecated. Please use the 'setupMiddlewares' opti
on.
Starting the development server...
Compiled successfully!

You can now view myfirstreact in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.0.5:3000

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

webpack compiled successfully
```

6. A look at the existing default webpage



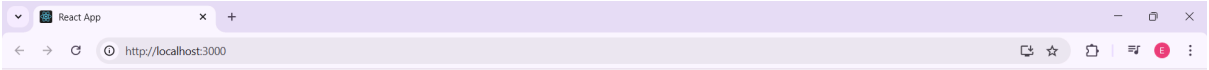
7. Replace the existing contents of **App.js** with the following code

```
import './App.css';

function App() {
  return (
    <h1>It's the first session of React!! Welcome onboard :)</h1>
  );
}

export default App;
```

8. Save the file and restart the development server

```
PS D:\DN-4.0-Java\C3-Products and Frameworks\M10-React\solutions\myfirstreact> npm start

  > myfirstreact@0.1.0 start
  > react-scripts start

  (node:22068) [DEP_WEBPACK_DEV_SERVER_ON_AFTER_SETUP_MIDDLEWARE] DeprecationWarning: 'onAfterSetupMiddleware' option is deprecated. Please use the 'setupMiddlewares' optio
  n.
  (Use `node --trace-deprecation ...` to show where the warning was created)
  (node:22068) [DEP_WEBPACK_DEV_SERVER_ON_BEFORE_SETUP_MIDDLEWARE] DeprecationWarning: 'onBeforeSetupMiddleware' option is deprecated. Please use the 'setupMiddlewares' opt
  ion.
  Starting the development server...
  Compiled successfully!

  You can now view myfirstreact in the browser.

    Local:            http://localhost:3000
    On Your Network:  http://192.168.0.5:3000

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

  webpack compiled successfully
```

9. Open http://localhost:3000/ on browser to view results



It's the first session of React!! Welcome onboard :)

2. ReactJS-HOL

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:

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

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

```
import React, {Component} from 'react';

import class Home extends Component{
    render(){
        <div>
        <h3> Welcome to the Home Page of Student Management Portal </h3>
        </div>
    }
}
```

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.

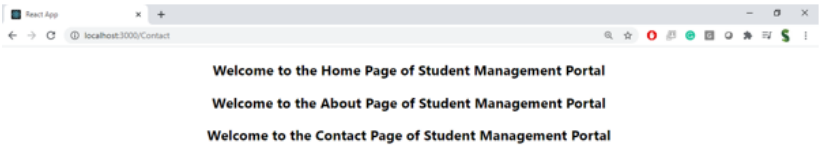
6. Edit the App.js to invoke the Home, About and Contact component as follows:

```
import logo from './logo.svg';
import './App.css';
import {Home} from './Components/Home';
import {About} from './Components/About';
import {Contact} from './Components/Contact';

function App() {
  return (
    <div className="container">
     <Home/>
     <About/>
     <Contact/>
    </div>
  );
}

export default App;
```

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

```
C:\studentapp>npm start
```

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



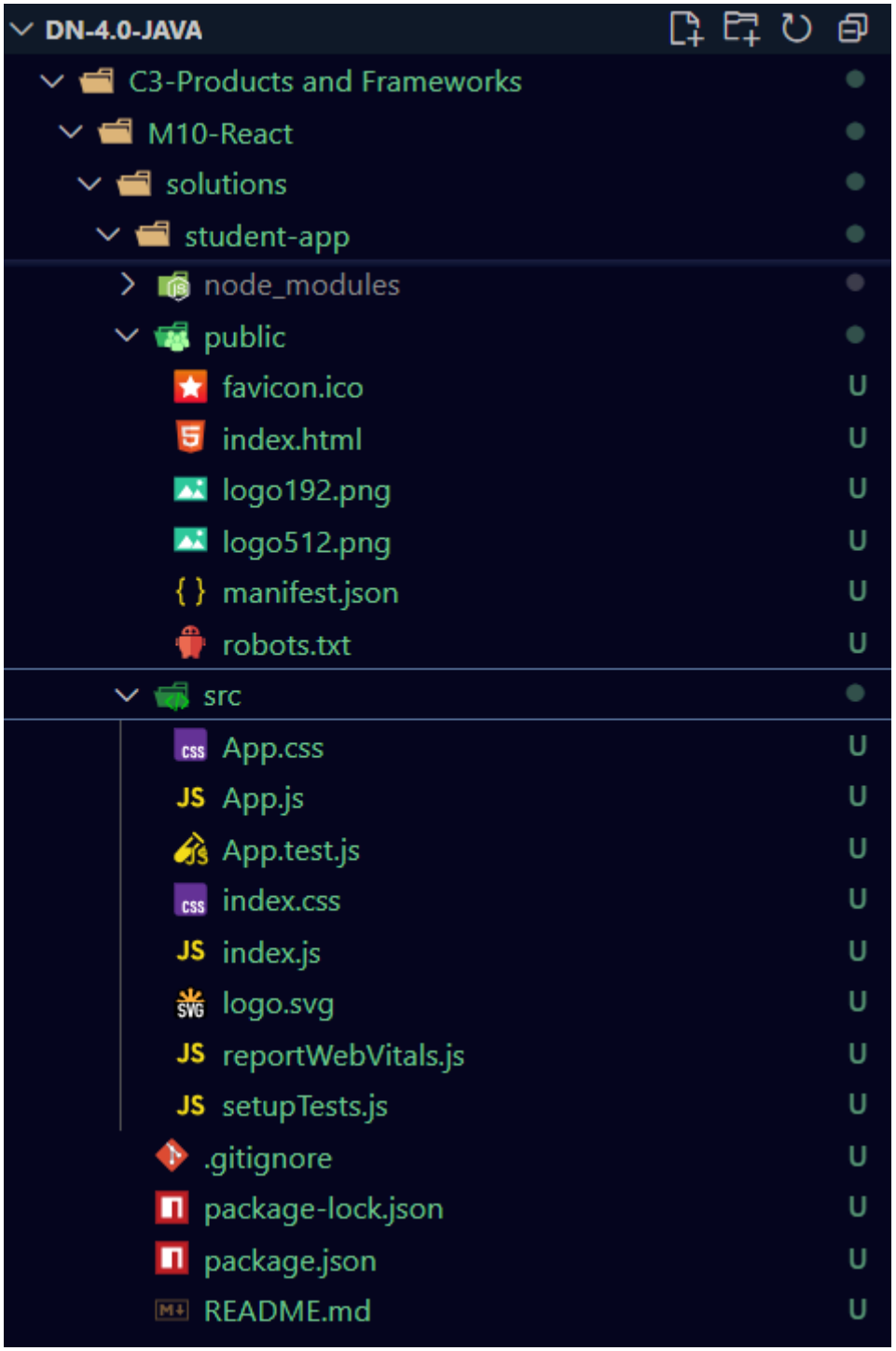
SOLUTION

1. Create a new react app with the name **student-app**

```
PS D:\DN-4.0-Java\C3-Products and Frameworks\M10-React\solutions> npx create-react-app studentApp
Cannot create a project named "studentApp" because of npm naming restrictions:

  * name can no longer contain capital letters

Please choose a different project name.
PS D:\DN-4.0-Java\C3-Products and Frameworks\M10-React\solutions> npx create-react-app student-app

Creating a new React app in D:\DN-4.0-Java\C3-Products and Frameworks\M10-React\solutions\student-app.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...


added 1323 packages in 1m

269 packages are looking for funding
  run `npm fund` for details

Installing template dependencies using npm...

added 18 packages, and changed 1 package in 7s

269 packages are looking for funding
  run `npm fund` for details
Removing template package using npm...


removed 1 package, and audited 1341 packages in 6s

269 packages are looking for funding
  run `npm fund` for details

9 vulnerabilities (3 moderate, 6 high)

To address all issues (including breaking changes), run:
  npm audit fix --force
```

```
9 vulnerabilities (3 moderate, 6 high)

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.

Success! Created student-app at D:\DN-4.0-Java\C3-Products and Frameworks\M10-React\solutions\student-app
Inside that directory, you can run several commands:

  npm start
    Starts the development server.

  npm run build
    Bundles the app into static files for production.

  npm test
    Starts the test runner.

  npm run eject
    Removes this tool and copies build dependencies, configuration files
    and scripts into the app directory. If you do this, you can’t go back!

We suggest that you begin by typing:

  cd student-app
  npm start

Happy hacking!
PS D:\DN-4.0-Java\C3-Products and Frameworks\M10-React\solutions>
```

2. Folder structure of **student-app**



3. Create a components folder and create new files: **Home.js**, **About.js**, **Contact.js**

4. Complete code of **Home.js**

```
import React, { Component } from "react";

class Home extends Component {
  render() {
    return (
      <div>
        <h3>Welcome to the Home Page of Student Management Portal</h3>
      </div>
```

```
    );
  }
}

export { Home };
```

5. Complete code of **About.js**

```
import React, { Component } from "react";

class About extends Component {
  render() {
    return (
      <div>
        <h3>Welcome to the About Page of Student Management Portal</h3>
      </div>
    );
  }
}

export { About };
```

6. Complete code of **Contact.js**

```
import React, { Component } from "react";

class Contact extends Component {
  render() {
    return (
      <div>
        <h3>Welcome to the Contact Page of Student Managemnet Portal</h3>
      </div>
    );
  }
}

export { Contact };
```

7. Complete code of **App.js**

```
import {Home} from './components/Home'
import {About} from './components/About'
import {Contact} from './components/Contact'

function App() {
  return (
    <div className="App">
      <Home/>
      <About />
      <Contact />
    </div>
  );
}

export default App;
```

8. Start the development server by **npm start**

```
PS D:\DN-4.0-Java\C3-Products and Frameworks\M10-React\solutions> cd student-app
PS D:\DN-4.0-Java\C3-Products and Frameworks\M10-React\solutions\student-app> npm start

> student-app@0.1.0 start
> react-scripts start

(node:2840) [DEP_WEBPACK_DEV_SERVER_ON_AFTER_SETUP_MIDDLEWARE] DeprecationWarning: 'onAfterSetupMiddleware' option is deprecated. Please use the 'se
tupMiddlewares' option.
(Use `node --trace-deprecation ...` to show where the warning was created)
(node:2840) [DEP_WEBPACK_DEV_SERVER_ON_BEFORE_SETUP_MIDDLEWARE] DeprecationWarning: 'onBeforeSetupMiddleware' option is deprecated. Please use the '
setupMiddlewares' option.
Starting the development server...
Compiled successfully!

You can now view student-app in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.0.5:3000

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

webpack compiled successfully
```

9. View the output at http://localhost:3000 on Chrome Browser

3. ReactJS-HOL

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:

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

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

```
import '../Stylesheets/mystyle.css'

const percentToDecimal= (decimal) => {
   return (decimal.toFixed(2) + '%')
}

const calcScore = (total, goal) => {
   return percentToDecimal(total/goal)
}
```

```
export const CalculateScore = ({Name,School, total, goal}) => (
      <div className="formatstyle">
         <h1><font color="Brown">Student Details:</font></h1>
         <div className="Name">
           <b> <span> Name:  </span> </b>
           <span>{Name}</span>
         </div>
         <div className="School">
           <b> <span> School: </span> </b>
           <span>{School}</span>
         </div>
         <div className="Total">
           <b><span>Total:</span> </b>
           <span>{total}</span>
           <span>Marks</span>
         </div>
         <div className="Score">
           <b>Score:</b>
           <span>
             {calcScore(
               total,
               goal
             )}
           </span>
         </div>
      </div>
```

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

```
.Name
{
  font-weight:300;
 color:blue;
}
.School
{
 color:crimson;
}
.Total
{
 color:darkmagenta;
}
.formatstyle
{
   text-align:center;
   font-size:large;
}
.Score
{
   color:forestgreen;
}
```

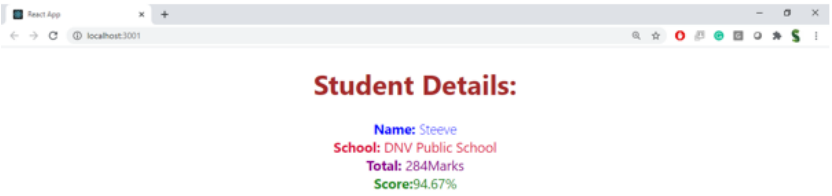
5. Edit the App.js to invoke the CalculateScore functional component as follows:

```
import {CalculateScore} from '../src/components/CalculateScore';

function App()
{

return(
<div>
   <CalculateScore Name={"Steeve"}
   School={"DNV Public School"}
   total={284}
   goal={3}
   />
</div>
)
}

export default App;
```

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

```
C:\scorecalculatorapp>npm start
```

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



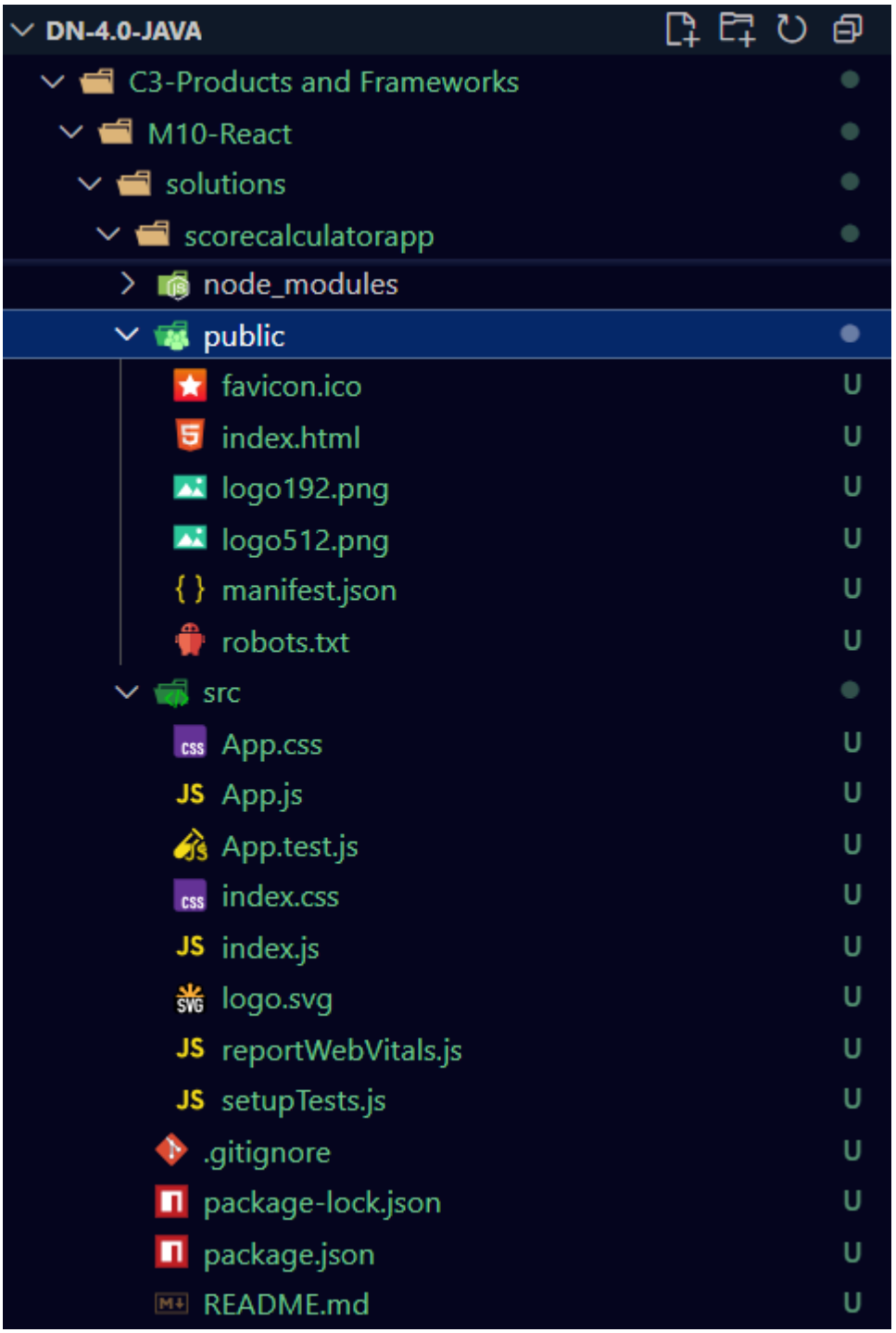
SOLUTION

1. Create the **scorecalculatorapp** react application

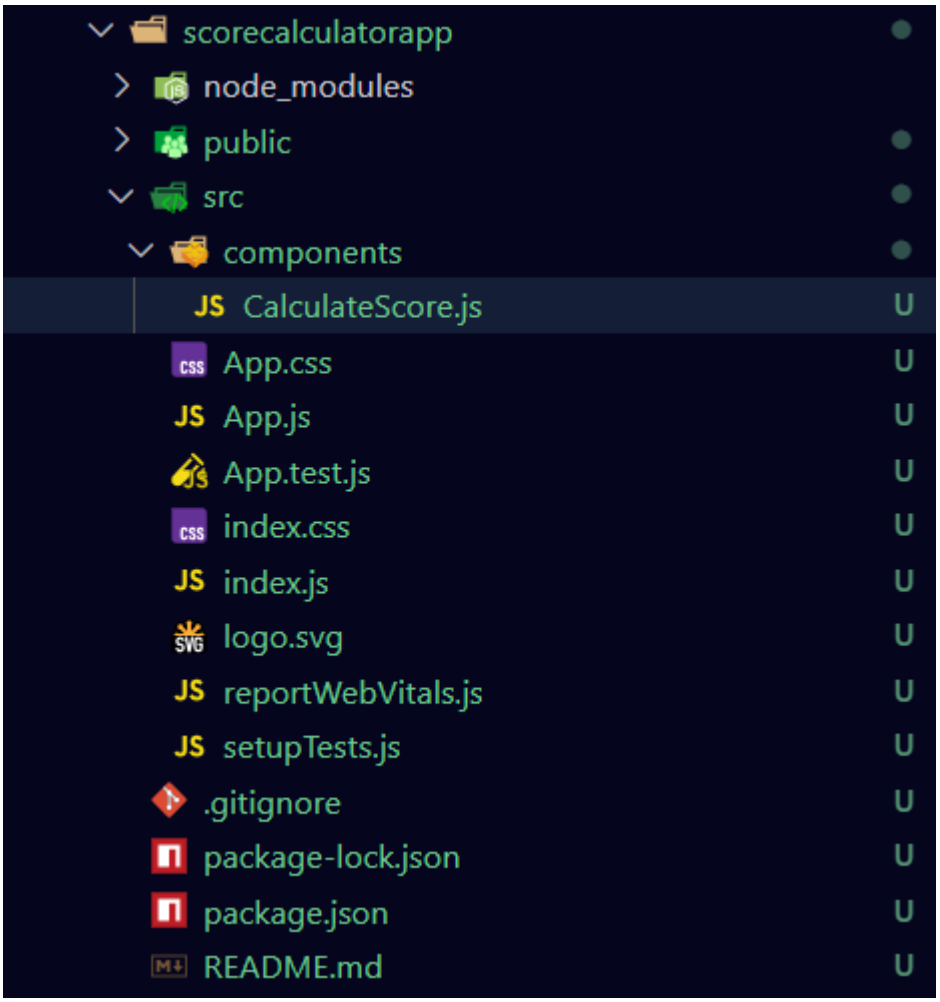
```
PS D:\DN-4.0-Java\C3-Products and Frameworks\M10-React\solutions> npx create-react-app scorecalculatorapp

Creating a new React app in D:\DN-4.0-Java\C3-Products and Frameworks\M10-React\solutions\scorecalculatorapp.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...


added 1323 packages in 57s

269 packages are looking for funding
  run `npm fund` for details

Installing template dependencies using npm...

added 18 packages, and changed 1 package in 7s

269 packages are looking for funding
  run `npm fund` for details
Removing template package using npm...


removed 1 package, and audited 1341 packages in 5s

269 packages are looking for funding
  run `npm fund` for details

9 vulnerabilities (3 moderate, 6 high)

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.

Success! Created scorecalculatorapp at D:\DN-4.0-Java\C3-Products and Frameworks\M10-React\solutions\scorecalculatorapp
Inside that directory, you can run several commands:
```

```
Success! Created scorecalculatorapp at D:\DN-4.0-Java\C3-Products and Frameworks\M10-React\solutions\scorecalculatorapp
Inside that directory, you can run several commands:

  npm start
    Starts the development server.

  npm run build
    Bundles the app into static files for production.

  npm test
    Starts the test runner.

  npm run eject
    Removes this tool and copies build dependencies, configuration files
    and scripts into the app directory. If you do this, you can't go back!

We suggest that you begin by typing:

  cd scorecalculatorapp
  npm start

Happy hacking!
PS D:\DN-4.0-Java\C3-Products and Frameworks\M10-React\solutions> cd scorecalculatorapp
PS D:\DN-4.0-Java\C3-Products and Frameworks\M10-React\solutions\scorecalculatorapp>
```

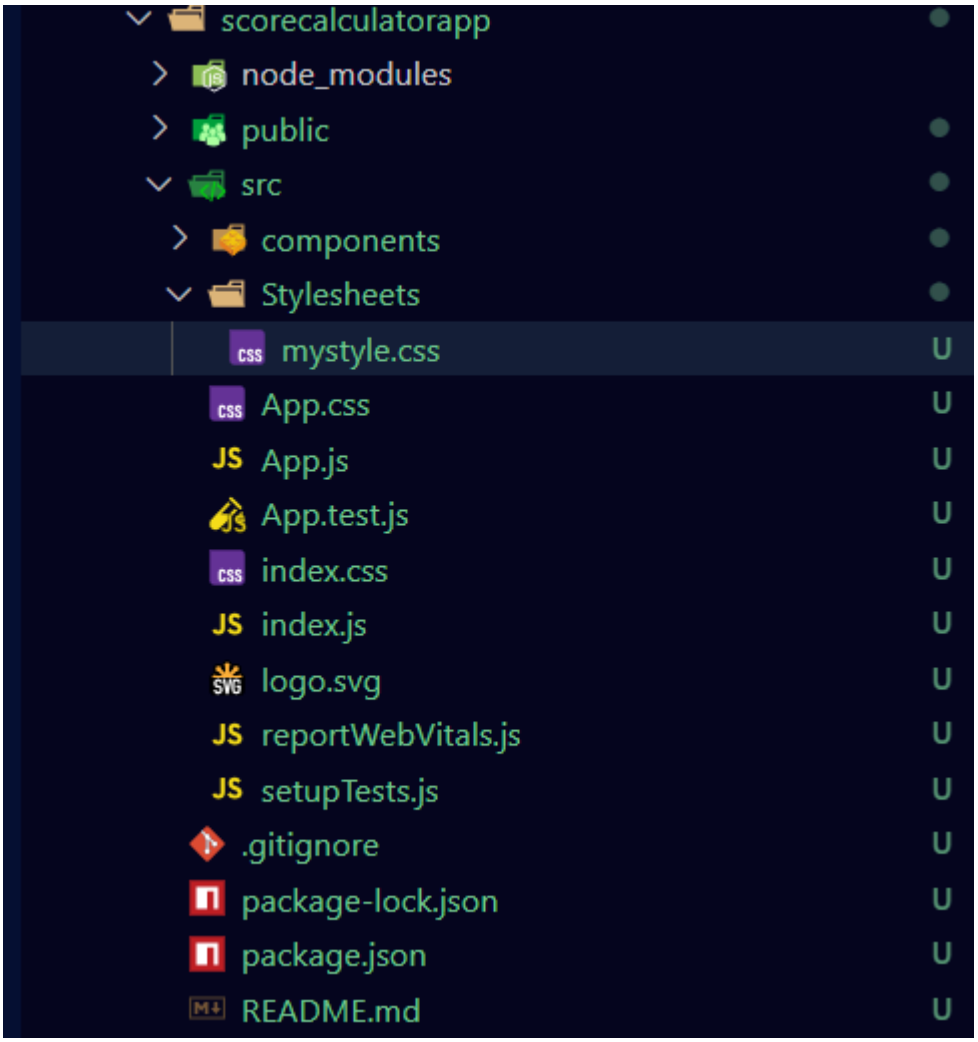
2. Folder structure of the app



3. Create the components folder and create a new file named **CalculateScore.js**



4. Create Stylesheets folder and create a new file called **mystyle.css**



5. Code of **CalculateScore.js**

```
import '../Stylesheets/mystyle.css'

const percentToDecimal = (decimal) => {
    return (decimal.toFixed(2) + '%')
}

const calculateScore = (total, goal) => {
    return percentToDecimal(total/goal)
}

export const CalculateScore = ({Name, School, total, goal}) => (
    <div className='formatstyle'>
        <h1><font color = "#FBDB93">Student Details: </font></h1>
```

```
        <div className='Name'>
            <b><span>Name: </span></b>
            <span>{Name}</span>
        </div>

        <div className='School'>
            <b><span>School: </span></b>
            <span> {School}</span>
        </div>

        <div className='Total'>
            <b><span>Total: </span></b>
            <span> {total}</span>
            <span> Marks </span>
        </div>

        <div className='Score'>
            <b> Score: </b>
            <span>
                {calculateScore(total, goal)}
            </span>
        </div>
    </div>
)
```

6. Code of **mystyle.css**

```
.Name {
    font-weight: 600;
    font-size: 1.2em;
    color : #F0F1C5;
}

.School {
    color: #E1EEBC;
}

.Total {
    color : #EBE8DB;
}

.formatstyle {
    text-align: center;
    font-size: large;
```

```
    background-color: hsla(337, 78%, 23%, 0.599);
    box-shadow: 0 1px 10px rgba(100, 10, 0, 0.7);
    width: 500px;
    border-radius: 15px;
    padding: 14px;
    margin: auto;
    margin-top : 250px;
}

.Score {
    color : #FFC6C6;
}

body {
    background-color: #EBE8DB;
}
```

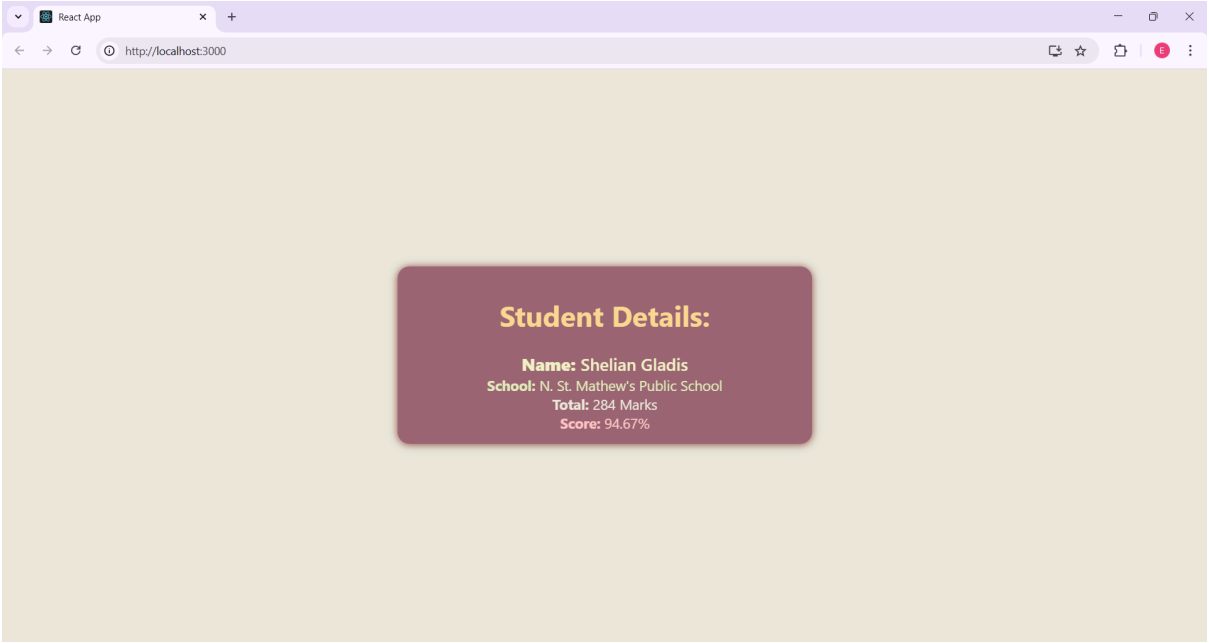
7. Code of **App.js**

```
import { CalculateScore } from "./components/CalculateScore";

function App() {
  return (
    <div>
      <CalculateScore Name={"Shelian Gladis"}
                      School = {"N. St. Mathew's Public School"}
                      total = {284}
                      goal = {3}
      />
    </div>
  );
}

export default App;
```

8. Start the development server

```
webpack compiled successfully
PS D:\DN-4.0-Java\C3-Products and Frameworks\M10-React\solutions\scorecalculatorapp> npm start

  > scorecalculatorapp@0.1.0 start
  > react-scripts start

  (node:9812) [DEP_WEBPACK_DEV_SERVER_ON_AFTER_SETUP_MIDDLEWARE] DeprecationWarning: 'onAfterSetupMiddleware' option is deprecated. Please use the 'setupMiddlewares' option.
  (Use `node --trace-deprecation ...` to show where the warning was created)
  (node:9812) [DEP_WEBPACK_DEV_SERVER_ON_BEFORE_SETUP_MIDDLEWARE] DeprecationWarning: 'onBeforeSetupMiddleware' option is deprecated. Please use the 'setupMiddlewares' option.
  Starting the development server...
  Compiled successfully!

  You can now view scorecalculatorapp in the browser.

    Local:            http://localhost:3000
    On Your Network:  http://192.168.0.5:3000

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

  webpack compiled successfully
```

9. Open the browser to view the output



4. ReactJS-HOL

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

```
JS Post.js U ×
class Post {
    constructor(id, title, body){
        this.id=id;
        this.title=title;
        this.body=body;
    }
}
export default Post;
```

Figure 2: Post class

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

```
JS Posts.js U ×
class Posts extends React.Component {
    constructor(props){
        super(props);
    }
}
```

Figure 3: 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)

```
JS Posts.js U ×
1  class Posts extends React.Component {
2      constructor(props){
3          super(props);
4          //code
5      }
6      loadPosts() {
7          //code
8      }
9  }
```

Figure 4: loadPosts() method

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

```
JS Posts.js U ×
1  class Posts extends React.Component {
2      constructor(props){
3          super(props);
4          //code
5      }
6      loadPosts() {
7          //code
8      }
9      componentDidMount() {
10         //code
11     }
12 }
```

Figure 5: componentDidMount() hook

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

```
JS Posts.js U ×
1  class Posts extends React.Component {
2  >   constructor(props) {···
5      }
6  >   loadPosts() {···
8      }
9  >   componentDidMount() {···
11     }
12     render() {
13         //code
14     }
15 }
```

Figure 6: render() method

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

```
JS Posts.js U ×
1   class Posts extends React.Component {
2 >     constructor(props) {…
5       }
6 >     loadPosts() {…
8       }
9 >     componentDidMount() {…
11      }
12 >    render() {…
14      }
15      componentDidCatch(error, info) {
16          //code
17        }
18  }
```

Figure 7: componentDidCatch() hook

10. Add the Posts component to App component.

Build and Run the application using npm start command.

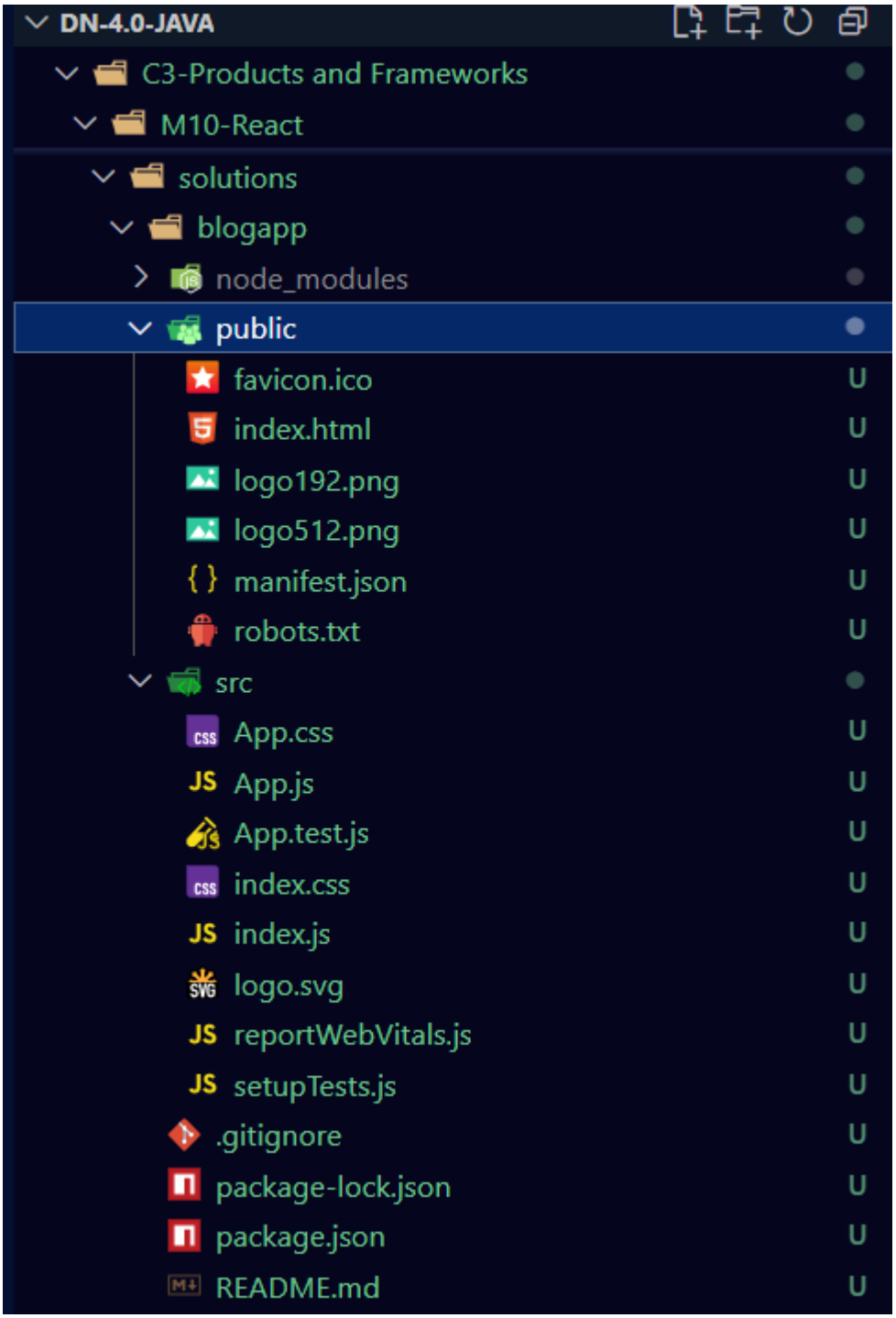
SOLUTION

1. Create **blogapp** react application

```
PS D:\DN-4.0-Java\C3-Products and Frameworks\M10-React\solutions> npx create-react-app blogapp

Creating a new React app in D:\DN-4.0-Java\C3-Products and Frameworks\M10-React\solutions\blogapp.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...


added 1323 packages in 58s

269 packages are looking for funding
  run `npm fund` for details

Installing template dependencies using npm...

added 18 packages, and changed 1 package in 7s

269 packages are looking for funding
  run `npm fund` for details
Removing template package using npm...


removed 1 package, and audited 1341 packages in 5s

269 packages are looking for funding
  run `npm fund` for details

9 vulnerabilities (3 moderate, 6 high)

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.

Success! Created blogapp at D:\DN-4.0-Java\C3-Products and Frameworks\M10-React\solutions\blogapp
Inside that directory, you can run several commands:
```

```
Success! Created blogapp at D:\DN-4.0-Java\C3-Products and Frameworks\M10-React\solutions\blogapp
Inside that directory, you can run several commands:

  npm start
    Starts the development server.

  npm run build
    Bundles the app into static files for production.

  npm test
    Starts the test runner.

  npm run eject
    Removes this tool and copies build dependencies, configuration files
    and scripts into the app directory. If you do this, you can’t go back!

We suggest that you begin by typing:

  cd blogapp
  npm start

Happy hacking!
PS D:\DN-4.0-Java\C3-Products and Frameworks\M10-React\solutions>
```

2. Folder structure of **blogapp**



3. Creation of **Post.js** in src folder

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

export default Post;
```

4. Create a **Posts** Component inside **Posts.js**

JS Post.js U JS Posts.js U × 4. ReactJS-HOL.docx U

C3-Products and Frameworks > M10-React > solutions > blogapp > src > JS Posts.js > Posts

```
1  import React from 'react';
2
3  class Posts extends React.Component {
4      constructor(props) {
5          super(props)
6      }
7
8
```

5. Initialize the component with a list of Post in state of the component using the constructor

JS Post.js U JS Posts.js U × 4. ReactJS-HOL.docx U

C3-Products and Frameworks > M10-React > solutions > blogapp > src > JS Posts.js > Posts

```
1  import React from 'react';
2
3  class Posts extends React.Component {
4      constructor(props) {
5          super(props)
6          this.state = {
7              posts : []
8          }
9      }
10
```

6. Create a method **loadPosts()** to fetch data from API https://jsonplaceholder.typicode.com/posts

```
JS Post.js U    JS Posts.js U ×    4. ReactJS-HOL.docx U
C3-Products and Frameworks > M10-React > solutions > blogapp > src > JS Posts.js > Posts
1   import React from 'react';
2
3   class Posts extends React.Component {
4       constructor(props) {
5           super(props)
6           this.state = {
7               posts : []
8           }
9       }
10
11
12      // load posts by calling an API
13      loadPosts() {
14          fetch('https://jsonplaceholder.typicode.com/posts')
15          .then(response => response.json())
16          .then(data => this.setState({posts: data}))
17          .catch((err) => console.log("oops, couldnt load posts" + err))
18      }
19
```

7. Make calls to **loadPosts()** which fetches the posts when the component mounts by implementing the **componentDidMount()** hook

```
JS Post.js U    JS Posts.js U ×    4. ReactJS-HOL.docx U
C3-Products and Frameworks > M10-React > solutions > blogapp > src > JS Posts.js > Posts
3   class Posts extends React.Component {
4       constructor(props) {
7               posts : []
8           }
9       }
10
11
12      // load posts by calling an API
13      loadPosts() {
14          fetch('https://jsonplaceholder.typicode.com/posts')
15          .then(response => response.json())
16          .then(data => this.setState({posts: data}))
17          .catch((err) => console.log("oops, couldnt load posts" + err))
18      }
19
20
21      // fetch the posts when component monts
22      componentDidMount() {
23          this.loadPosts();
24      }
25
```

8. The **render()** displays the posts fetched

JS Post.js U | JS Posts.js U × | Posts.css U | 4. ReactJS-HOL.docx U

C3-Products and Frameworks > M10-React > solutions > blogapp > src > JS Posts.js > default

```
4   class Posts extends React.Component {
22      // fetch the posts when component monts
23      componentDidMount() {
24          this.loadPosts();
25      }
26
27      // display the posts prettily
28      render() {
29          return (
30              <div className='posts'>
31                  {this.state.posts.map(post => {
32                      return(
33                      <div key = {post.id} className='post'>
34                          <circle className='post-id'>{post.id}</circle>
35                          <h3 className='post-title'>{post.title}</h3>
36                          <p className='post-body'>{post.body}</p>
37                      </div>
38                  )})}
39              </div>
40          )
41      }
42  }
43
44  export default Posts;
```

9. The **componentDidCatch()** method displays errors in the component as alerts.

JS Post.js U | JS Posts.js U × | Posts.css U | 4. ReactJS-HOL.docx U

C3-Products and Frameworks > M10-React > solutions > blogapp > src > JS Posts.js > default

```
4   class Posts extends React.Component {
28      render() {
31                  {this.state.posts.map(post => {
38                  )})}
39              </div>
40          )
41      }
42
43      // display error messages happening in this compoenent
44      componentDidCatch(error, info) {
45          alert("oops, something went wrong :(" + error)
46          console.log(error)
47          console.log(info)
48          this.setState({hasError : true})
49      }
50  }
51
52  export default Posts;
```

10. Start the development server by using **npm start**

```
PS D:\DN-4.0-Java\C3-Products and Frameworks\M10-React\solutions\blogapp> npm start

  > blogapp@0.1.0 start
  > react-scripts start

  (node:6060) [DEP_WEBPACK_DEV_SERVER_ON_AFTER_SETUP_MIDDLEWARE] DeprecationWarning: 'onAfterSetupMiddleware' option is deprecated. Please use the 'se
  tupMiddlewares' option.
  (Use `node --trace-deprecation ...` to show where the warning was created)
  (node:6060) [DEP_WEBPACK_DEV_SERVER_ON_BEFORE_SETUP_MIDDLEWARE] DeprecationWarning: 'onBeforeSetupMiddleware' option is deprecated. Please use the '
  setupMiddlewares' option.
  Starting the development server...
  Compiled successfully!

  You can now view blogapp in the browser.

    Local:            http://localhost:3000
    On Your Network:  http://192.168.0.5:3000

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

  webpack compiled successfully
```

11. Open the browser to view the results

12. Complete code of **Post.js**

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

export default Post;
```

13. Complete code of **Posts.js**

```
import React from 'react';
import './Posts.css'

class Posts extends React.Component {
    constructor(props) {
        super(props)
        this.state = {
            posts : []
        }
    }


    // load posts by calling an API
    loadPosts() {
        fetch('https://jsonplaceholder.typicode.com/posts')
        .then(response => response.json())
        .then(data => this.setState({posts: data}))
        .catch((err) => console.log("oops, couldnt load posts" + err))
    }


    // fetch the posts when component monts
    componentDidMount() {
        this.loadPosts();
    }

    // display the posts prettily
    render() {
        return (
            <div className='posts'>
```

```
                {this.state.posts.map(post => {
                    return(
                    <div key = {post.id} className='post'>
                        <div className='post-id'>{post.id}</div>
                        <h3 className='post-title'>{post.title}</h3>
                        <p className='post-body'>{post.body}</p>
                    </div>
                )})}
            </div>
        )
    }

    // display error messages happening in this compoenent
    componentDidCatch(error, info) {
        alert("oops, something went wrong :(" + error)
        console.log(error)
        console.log(info)
        this.setState({hasError : true})
    }
}

export default Posts;
```

14. Complete code of **Posts.css**

```
.posts {
    display: grid;
    grid-template-columns: repeat(3, 1fr);
    grid-gap: 2rem;
    padding: 1 rem;
}

.post {
    box-shadow: 0 3px 10px 0 rgba(0,0,0,0.2);
    border-radius: 20px;
    padding: 1rem;
    background-color: #DDDAD0;
}

.post-id {
    background-color: #0A400C;
    border: 2px dashed #fff;
    color: white;
    border-radius: 50%;
    font-size: 1.5rem;
```

```
    width: 2rem;
    height: 2rem;
    display: flex;
    justify-content: center;
    align-items: center;
}
.post-title {
    font-family : Arial, Helvetica, sans-serif;
    color: #0A400C;
    font-weight: 700;
    font-size : 2em;
}

.post-body{
    color: #708A58;
    font-family : Arial, Helvetica, sans-serif;
    font-weight: 400;
    font-size : 1.3em;
}
```

15. Complete code of **App.js**

```
import Posts from './Posts.js';

function App() {
  return (
    <div className="App">
      <Posts/>
    </div>
  );
}

export default App;
```

5. ReactJS-HOL

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

```
C:\Windows\System32\cmd.exe

C:\CTS-NewHandsOns\ReactHandsOns\cohortstracker>npm install
```

Figure 1: Restore packages

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

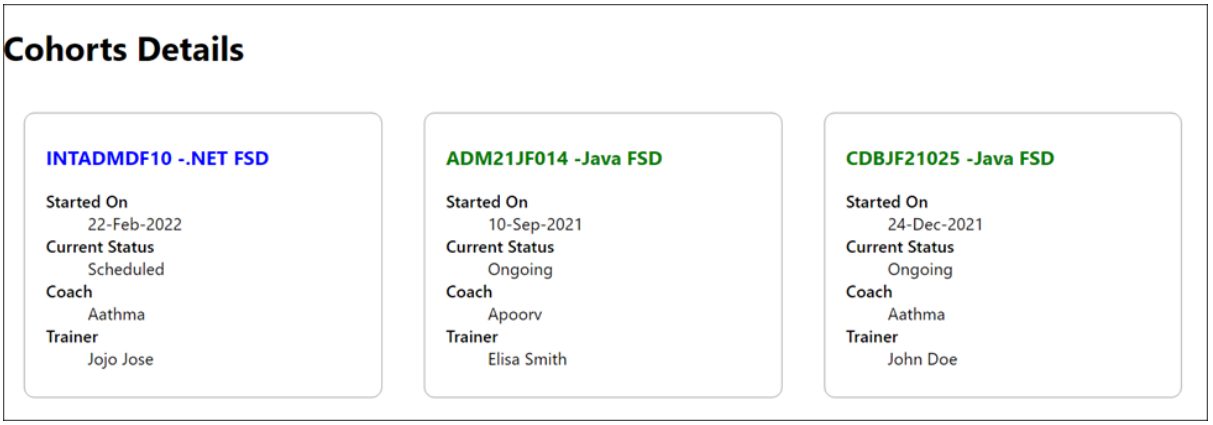
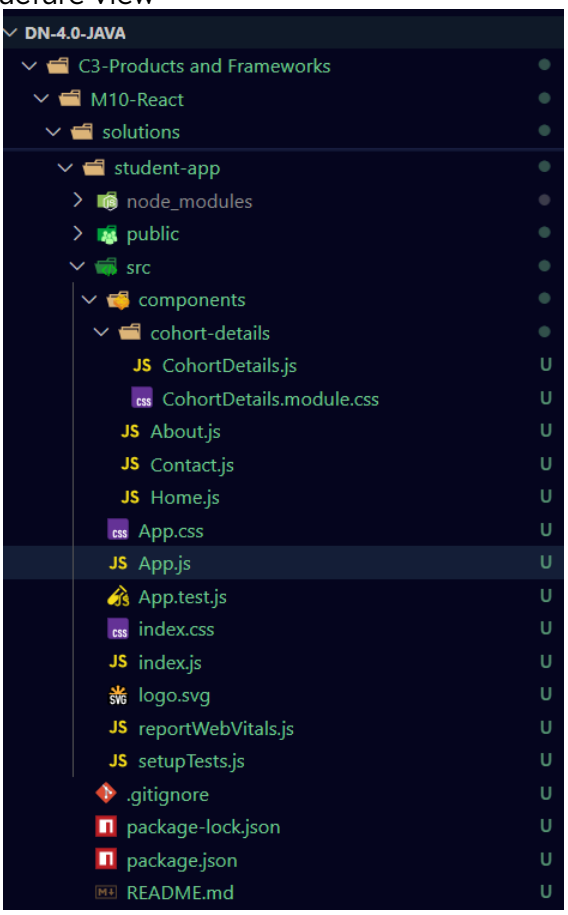


Figure 2: Final Result

SOLUTION

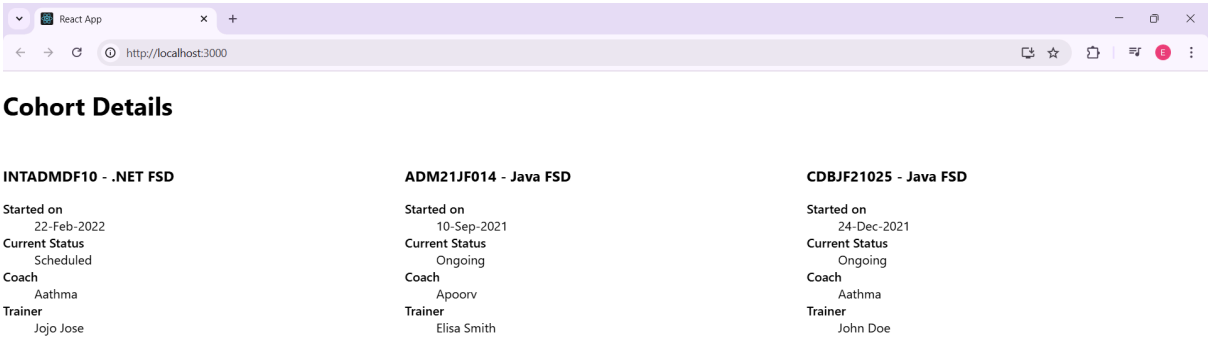
1. Download and unzip the project. In my case, I am going to create a **CohortDetails.js** functional component to mimic the above image.
2. Folder structure view



3. Turn the development server on by running **npm start**

```
webpack compiled with 1 warning
PS D:\DN-4.0-Java\C3-Products and Frameworks\M10-React\solutions\student-app> npm start

> student-app@0.1.0 start
> react-scripts start

(node:20056) [DEP_WEBPACK_DEV_SERVER_ON_AFTER_SETUP_MIDDLEWARE] DeprecationWarning: 'onAfterSetupMiddleware' option is deprecated. Please use the 's
etupMiddlewares' option.
(Use `node --trace-deprecation ...` to show where the warning was created)
(node:20056) [DEP_WEBPACK_DEV_SERVER_ON_BEFORE_SETUP_MIDDLEWARE] DeprecationWarning: 'onBeforeSetupMiddleware' option is deprecated. Please use the
'setupMiddlewares' option.
Starting the development server...
Compiled successfully!

You can now view student-app in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.0.5:3000

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

webpack compiled successfully
```

4. We can see the initial output by opening our browser at https://localhost:3000



5. Completed the code for adding box styles to the details. Now the output is:

6. Now, added the functionality of green header for ongoing status and blue header for other scenarios

7. Code for **CohortDetails.js**

```
import styles from './CohortDetails.module.css'

function CohortDetails({cohort}) {

    const headerStyle = cohort.status === "Ongoing" ? styles.ongoingHeader :
styles.nonOngoingHeader
    return (
        <dl className={styles.box}>
```

```
            <h3 className={headerStyle}>{cohort.ID}</h3>
            <dt>Started on </dt>
            <dd>{cohort.startDate}</dd>

            <dt>Current Status</dt>
            <dd>{cohort.status}</dd>

            <dt>Coach</dt>
            <dd>{cohort.coach}</dd>

            <dt>Trainer</dt>
            <dd>{cohort.trainer}</dd>
        </dl>
    )
}

export default CohortDetails;
```

8. Complete code of **CohortDetails.module.css**

```
.box {
    width: 300px;
    display: inline-block;
    margin: 10px;
    padding : 10px 20px;
    border: 1px solid black;
    border-radius: 10px;
}

dt {
    font-weight: 500;
}

.ongoingHeader {
    color: green;
}

.nonOngoingHeader {
    color: blue;
}
```

9. Complete code of **App.js**

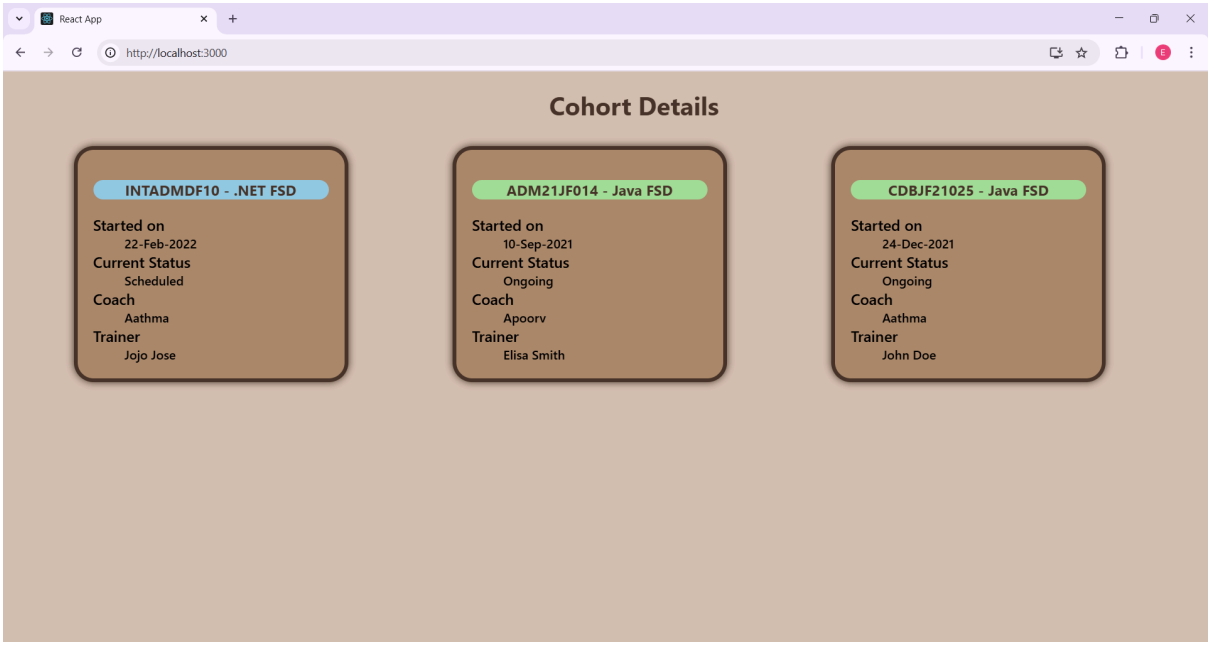
```
import './App.css'
import  CohortDetails  from "./components/cohort-details/CohortDetails";
```

```
function App() {
  const cohorts = [
    {
      ID: "INTADMDF10 - .NET FSD",
      startDate: "22-Feb-2022",
      status: "Scheduled",
      coach: "Aathma",
      trainer: "Jojo Jose",
    },
    {
      ID: "ADM21JF014 - Java FSD",
      startDate: "10-Sep-2021",
      status: "Ongoing",
      coach: "Apoorv",
      trainer: "Elisa Smith",
    },
    {
      ID: "CDBJF21025 - Java FSD",
      startDate: "24-Dec-2021",
      status: "Ongoing",
      coach: "Aathma",
      trainer: "John Doe",
    },
  ];

  return (
    <div className="App">
      <h1>Cohort Details</h1>
      <div className="cohort-grid">
        {cohorts.map((cohort, index) => (
          <CohortDetails key={index} cohort={cohort} />
        ))}
      </div>
    </div>
  );
}

export default App;
```

10. Complete code of **App.css**

```
.cohort-grid {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
}
```

11. Additional Styling for the output



12. Styled codes:
    a. **App.css**

```
.cohort-grid {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
}

.App h1 {
  color : #493628;
  text-align: center;
  font-size: 2rem;
}
```

    b. **CohortDetails.module.css**

```
.box {
    width: 300px;
    display: inline-block;
    margin: 10px;
    padding : 20px 20px;
    border: 5px solid #493628;
    border-radius: 25px;
    background-color: #AB886D;
    box-shadow: 0 0 10px #493628;
}
```

```
dt {
    font-weight: 500;
    font-size: 1.2rem;
}

dd {
    font-weight: 500;
    font-size: 1rem;
}

.ongoingHeader {
    /* color: green;    */
    color : #493628;
    background: #A3DC9A;
    text-align: center;
    border-radius: 40px;
}

.nonOngoingHeader {
    /* color: blue; */
     color : #493628;
    background: #91C8E4;
    text-align: center;
    border-radius: 40px;
}
```

c. **index.css**

```
body {
  margin-left: 80px;
  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto',
'Oxygen',
    'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',
    sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  background-color: #D6C0B3;
}

code {
  font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',
    monospace;
}
```