WEEK 6

ReactJS Hands-on Assessment

Task 1: Create React App 'myfirstreact'
Objective: Create a simple React app that prints a heading.

Steps:
1. Install Node.js and npm from: https://nodejs.org/en/download/
2. Open VS Code Terminal and run:

```
npx create-react-app myfirstreact
cd myfirstreact
code .
```

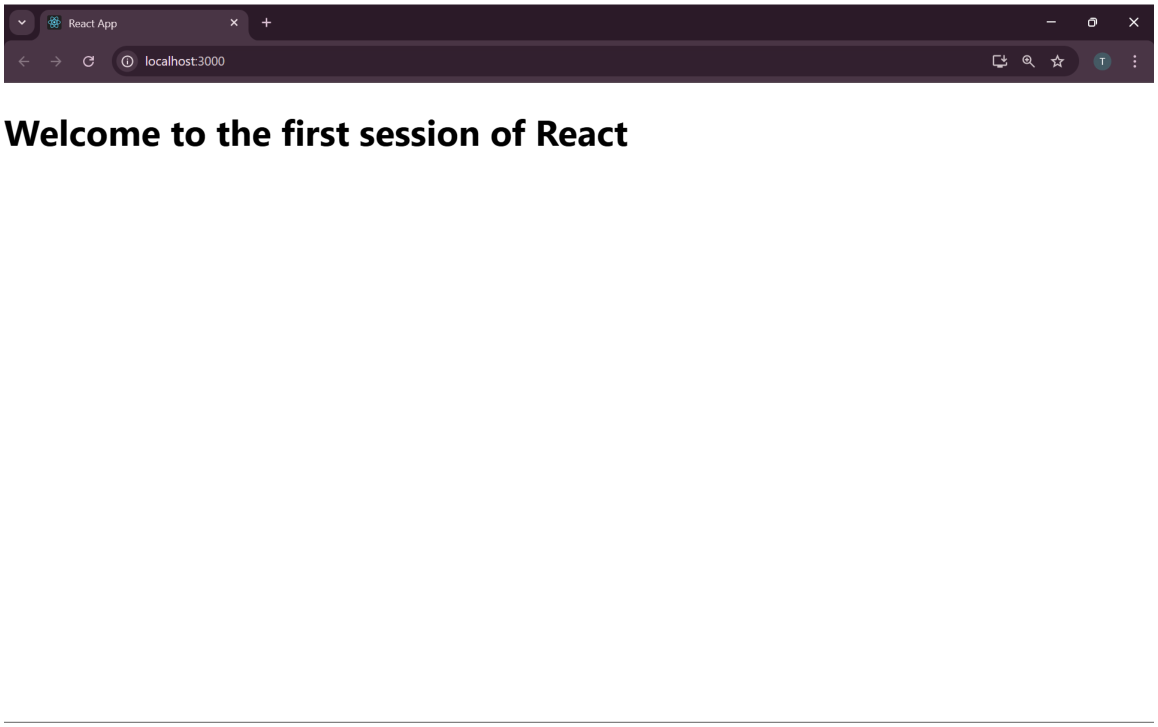
3. Open src/App.js and replace content with:

```
import React from 'react';

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

export default App;
```

4. Run the application:

```
npm start
```

OUTPUT:



Welcome to the first session of React

Task 2: StudentApp with Class Components
Objective: Display Home, About, and Contact pages using class components.

Steps:
1. Create project:

```
npx create-react-app studentapp
cd studentapp
code .
```

2. Under src, create folder Components and files Home.js, About.js, Contact.js

Example: Home.js

```
import React from 'react';

class Home extends React.Component {
render() {
return <h2>Welcome to the Home page of Student Management Portal</h2>;
}
}

export default Home;
```

3. Repeat above for About.js and Contact.js with respective messages.

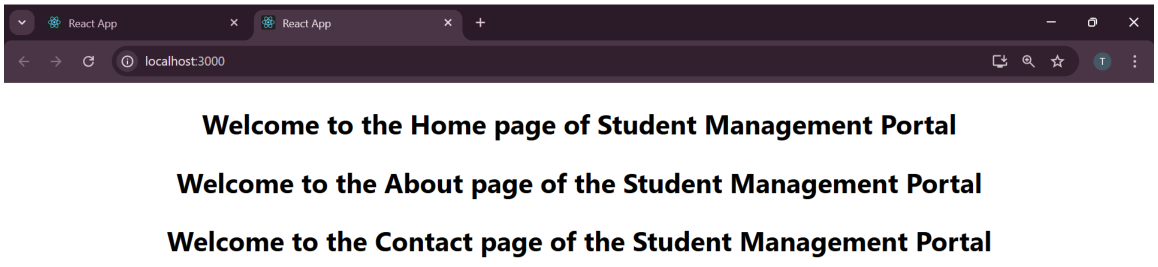
4. Update App.js:

```
import React from 'react';
import Home from './Components/Home';
import About from './Components/About';
import Contact from './Components/Contact';

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

export default App;
```

5. Run the application:

```
npm start
```

OUTPUT



```
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
```

Task 3: Score Calculator App

Objective: Create a function component to calculate and display average score.

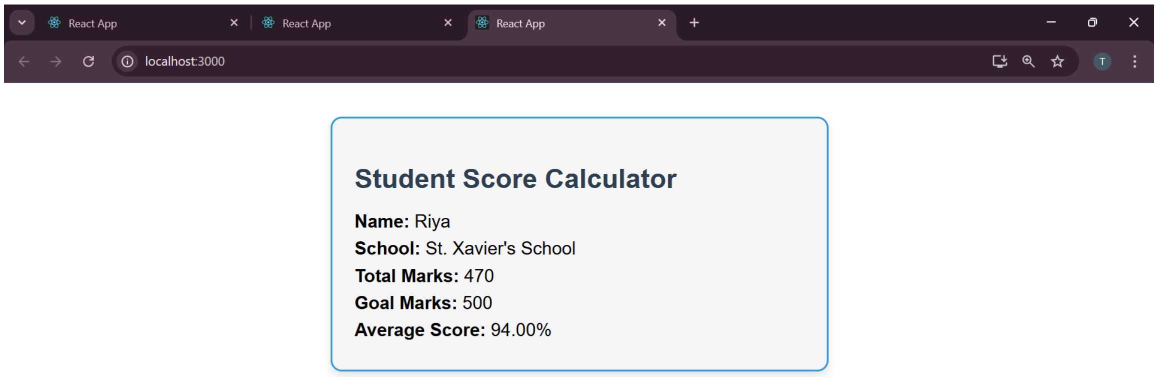
Steps:
1. Create project:
```
npx create-react-app scorecalculatorapp
cd scorecalculatorapp
code .
```
2. Create src/Components/CalculateScore.js

```
import React from 'react';
import '../Stylesheets/mystyle.css';

function CalculateScore() {
const name = "Riya";
const school = "St. Xavier's";
const total = 470;
const goal = 500;
const average = (total / goal) * 100;

return (
<div className="score-card">
<h2>Student Score Calculator</h2>
<p><strong>Name:</strong> {name}</p>
<p><strong>School:</strong> {school}</p>
<p><strong>Total:</strong> {total}</p>
<p><strong>Goal:</strong> {goal}</p>
<p><strong>Average:</strong> {average.toFixed(2)}%</p>
</div>
);
}

export default CalculateScore;
```

3. Create mystyle.css in Stylesheets folder:

```
.score-card {
width: 400px;
margin: 30px auto;
padding: 20px;
border: 2px solid #3498db;
border-radius: 10px;
box-shadow: 0px 4px 8px rgba(0,0,0,0.1);
font-family: Arial;
}
```

4. Import and use component in App.js
5. Run the application:
npm start

OUTPUT:


Student Score Calculator

Name: Riya
School: St. Xavier's School
Total Marks: 470
Goal Marks: 500
Average Score: 94.00%

Task 4: BlogApp with Lifecycle Methods
Objective: Use componentDidMount() to fetch posts and componentDidCatch() to handle errors.

Steps:
1. Create project:
npx create-react-app blogapp
cd blogapp
code .

2. Create Post.js:

import React from 'react';

class Post extends React.Component {
render() {
return (
<div>

```
<h3>{this.props.title}</h3>
<p>{this.props.body}</p>
</div>
);
}
}

export default Post;
```

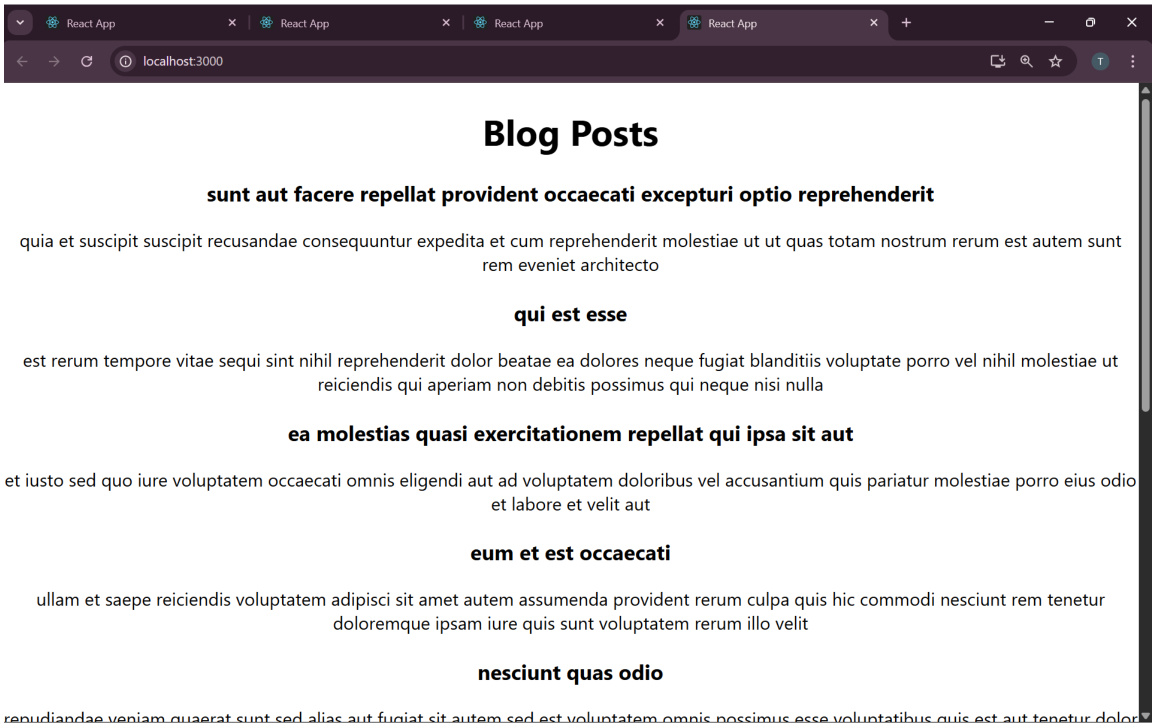
3. Create Posts.js:

```
import React from 'react';
import Post from './Post';

class Posts extends React.Component {
constructor() {
super();
this.state = { posts: [], hasError: false };
}

componentDidMount() {
this.loadPosts();
}

loadPosts() {
fetch('https://jsonplaceholder.typicode.com/posts')
.then(response => response.json())
.then(data => this.setState({ posts: data }))
.catch(() => this.setState({ hasError: true }));
}

componentDidCatch() {
alert("An error occurred");
}

render() {
if (this.state.hasError) return <h2>Error loading posts</h2>;
return (
<div>
<h1>Blog Posts</h1>
{this.state.posts.slice(0, 10).map(post => (
<Post key={post.id} title={post.title} body={post.body} />
))}
</div>
);
}
}

export default Posts;
```

4. Update App.js to import and use <Posts />
5. Run the application:

```
npm start
```

OUTPUT:



```
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
```

Task 5: Style Cohort Cards using CSS Modules
Objective: Style cards using CSS Modules and inline styles.

Steps:
1. Create project:

```
npx create-react-app cohortstracker
cd cohortstracker
code .
```

2. Create src/components/CohortDetails.js

```
import React from 'react';
import styles from './CohortDetails.module.css';

function CohortDetails({ cohort }) {
```

```
const titleStyle = { color: cohort.status.toLowerCase() === 'ongoing' ? 'green' :
'blue' };

return (
<div className={styles.box}>
<h3 style={titleStyle}>{cohort.name}</h3>
<dl>
<dt>Started On</dt><dd>{cohort.started}</dd>
<dt>Current Status</dt><dd>{cohort.status}</dd>
<dt>Coach</dt><dd>{cohort.coach}</dd>
<dt>Trainer</dt><dd>{cohort.trainer}</dd>
</dl>
</div>
);
}

export default CohortDetails;
```

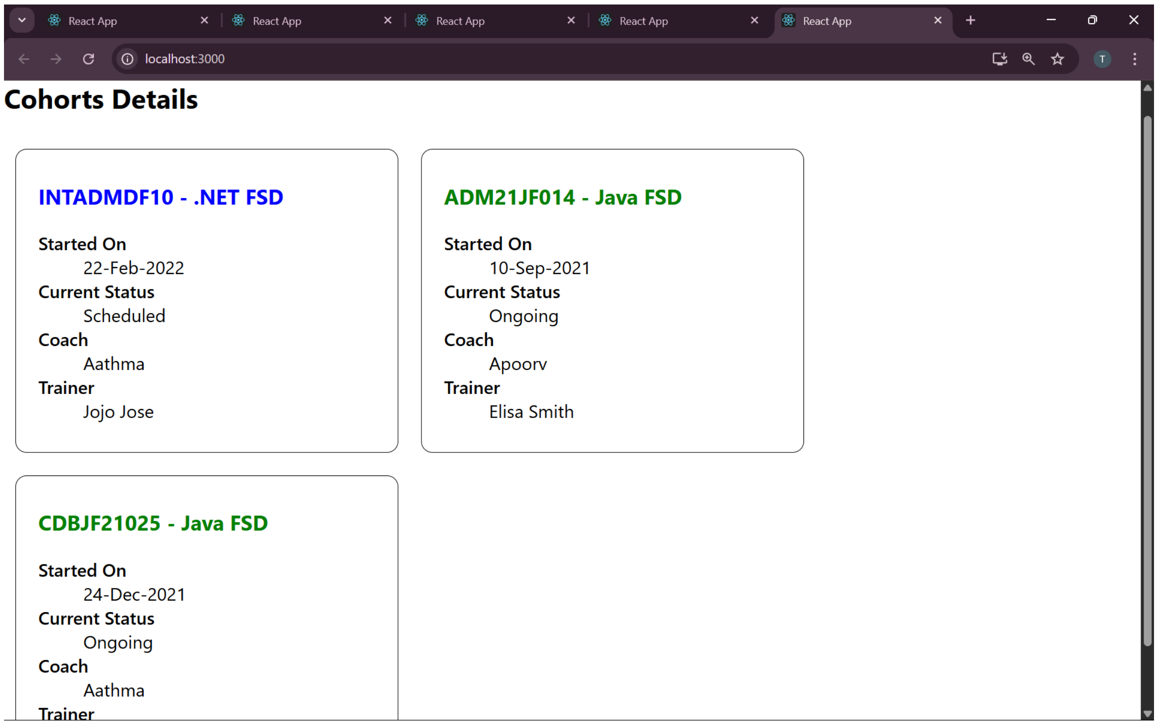
3. Create CohortDetails.module.css:

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

dt {
font-weight: 500;
}
```

4. Add multiple cohort data in App.js and render using map and <CohortDetails />
5. Run the application:

```
npm start
```

OUTPUT:



Cohorts Details

INTADMDF10 - .NET FSD

Started On
22-Feb-2022
Current Status
Scheduled
Coach
Aathma
Trainer
Jojo Jose

ADM21JF014 - Java FSD

Started On
10-Sep-2021
Current Status
Ongoing
Coach
Apoorv
Trainer
Elisa Smith

CDBJF21025 - Java FSD

Started On
24-Dec-2021
Current Status
Ongoing
Coach
Aathma
Trainer