WEEK 6

REACTJS-HOL

Q.1)OBJECTIVES

---

1. Define SPA and its benefits

SPA (Single-Page Application) is a web application that loads a single HTML page and dynamically updates content as the user interacts with the app, without refreshing the entire page.

Benefits:

- **Fast User Experience:** Only content changes, not the full page reload.
- **Reduced Server Load:** Since pages aren't reloaded, fewer requests go to the server.
- **Smooth Navigation:** Seamless transitions between views.
- **Better Performance:** Especially after initial load, SPAs can be faster.

---

2. Define React and identify its working

React is an open-source JavaScript library developed by Facebook for building **user interfaces**, especially for SPAs.

How React Works:

- React uses a **component-based** architecture.
- It maintains a **virtual DOM** (a lightweight copy of the real DOM).
- When a change occurs, React updates only the part of the DOM that changed, improving efficiency.

---

3. Identify the differences between SPA and MPA

Feature	SPA (Single-Page App)	MPA (Multi-Page App)
Page Load	Loads once	Reloads on each page
Speed	Faster after initial load	Slower due to reloads
UX	Seamless	Page flicker on reload
Development	Requires JS frameworks like React	Often built with traditional HTML/CSS/JS
SEO	More complex to handle	Easier due to static pages

---

4. Explain Pros & Cons of Single-Page Application

Pros:

- Faster interactions
- Better user experience
- Efficient use of API and data

Cons:

- SEO can be challenging
- Initial load may be slower
- JavaScript must work correctly (JS disabled = broken app)

---

5. Explain about React

React is:

- A **JavaScript library** for building user interfaces.
- Known for **reusability of components**.
- Uses **JSX** (JavaScript XML) to mix HTML with JS.
- Allows **efficient DOM updates** via virtual DOM.
- Backed by a large community and used in many real-world apps.

---

6. Define virtual DOM

Virtual DOM is a programming concept used in React where a **virtual representation** of the actual DOM is kept in memory.

How it helps:

- When the UI changes, React updates the virtual DOM first.
- It then compares the virtual DOM with the old one (**diffing**).
- Only the changed parts are updated in the real DOM (**reconciliation**), making updates fast.

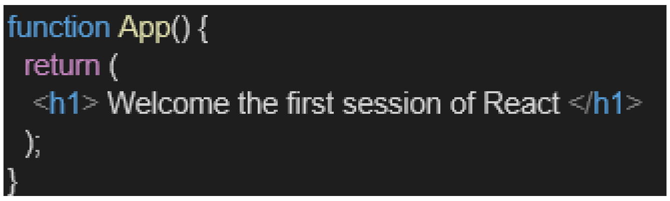
---

7. Explain Features of React

- **Component-Based Architecture:** UI is built with reusable components.
- **JSX Syntax:** Combines HTML and JS in one file.
- **Virtual DOM:** Enables fast and efficient updates.
- **Unidirectional Data Flow:** Data flows from parent to child components.

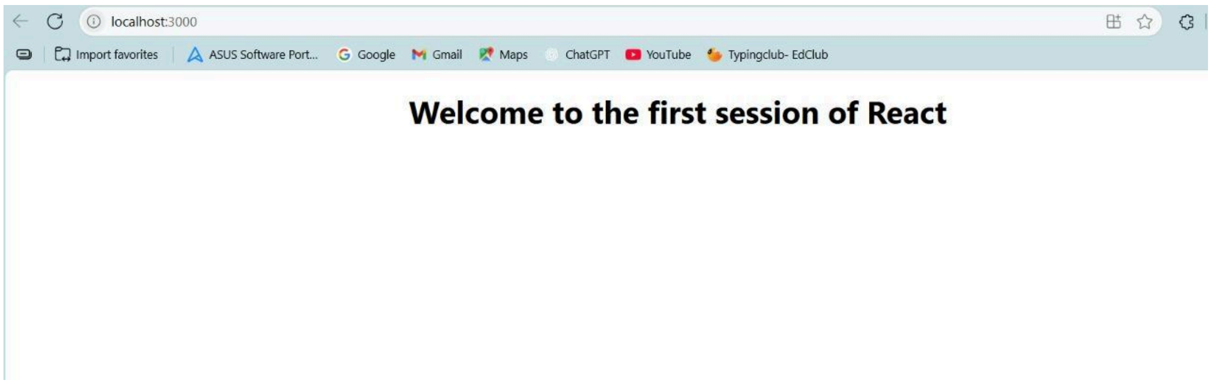
- **Hooks:** Use state and other features without writing classes.
- **Strong Community Support:** Vast number of libraries and tools.

Q1)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.



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

OUTPUT:



Welcome to the first session of React

Q.2)OBJECTIVES

---

1. Explain React components

React components are building blocks of a React application. A component is a self-contained piece of UI, which can include HTML, CSS, and JavaScript logic. Components can be reused and composed together to build complex interfaces.

Example:

```
function Welcome() {
  return <h1>Hello, React!</h1>;
}
```

---

2. Identify the differences between components and JavaScript functions

Feature	React Component	JavaScript Function
Purpose	Returns JSX (UI)	Returns data or performs logic

Feature	React Component	JavaScript Function
Usage	Renders UI	Logic or calculations
Lifecycle Methods	Yes (for class components)	No
State & Hooks	Yes (in components)	No (unless used inside component)

---

3. Identify the types of components

React has two main types of components:

- Class Components: ES6 classes that extend React.Component and use lifecycle methods.
- Function Components: Simpler components using functions; modern React uses Hooks to manage state and lifecycle inside them.

---

4. Explain class component

A class component is a React component defined using an ES6 class. It has access to state and lifecycle methods like componentDidMount.

Example:

```
class Welcome extends React.Component {
  render() {
    return <h1>Hello, {this.props.name}</h1>;
  }
}
```

---

5. Explain function component

A function component is a React component defined as a JavaScript function. In modern React, you can use Hooks like useState and useEffect to handle state and side effects.

Example:

```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}
```

With Hook:

```
function Counter() {
  const [count, setCount] = useState(0);
  return <button onClick={() => setCount(count + 1)}>Count: {count}</button>;
}
```

---

6. Define component constructor

In class components, the constructor is a special function used to:

- Initialize state
- Bind event handler methods

Syntax:

```
class MyComponent extends React.Component {
  constructor(props) {
    super(props); // Required to access 'this.props'
    this.state = { count: 0 };
  }
}
```

---

7. Define render() function

The **render()** function is a mandatory method in class components. It returns the JSX (UI) that should be displayed.

Example:

```
class MyComponent extends React.Component {
  render() {
    return <h1>Hello World</h1>;
  }
}
```

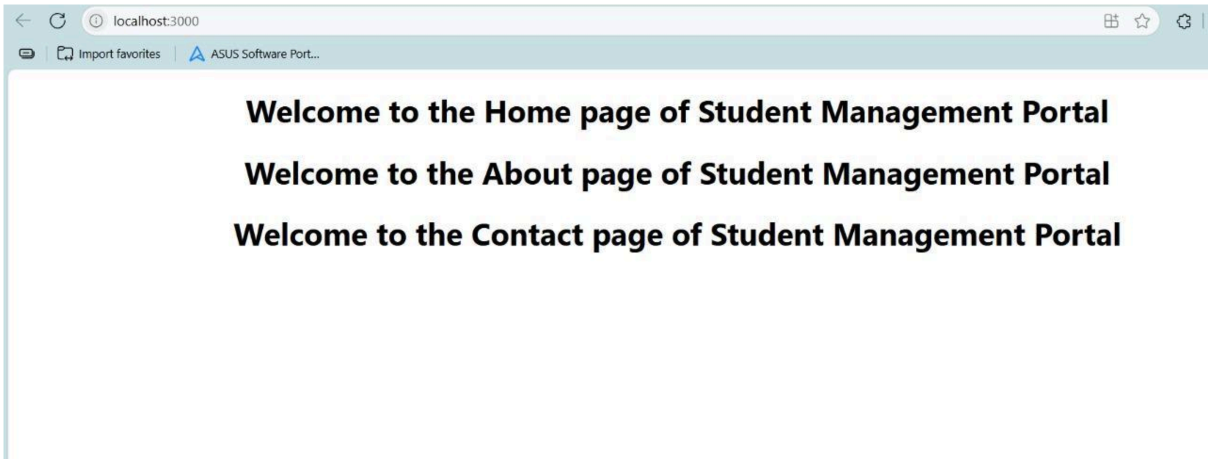
In function components, there’s no render() method — the component itself acts as the render function.

Q2) 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.

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

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

```
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;
```

OUTPUT:



localhost:3000

Import favorites | ASUS Software Port...

Welcome to the Home page of Student Management Portal

Welcome to the About page of Student Management Portal

Welcome to the Contact page of Student Management Portal

Q.3) OBJECTIVES

---

1. Explain React Components

React components are **independent, reusable pieces of UI** in a React application. Each component can return HTML (via JSX), and can be composed with others to build complex user interfaces. They help in maintaining a **modular and maintainable** code structure.

---

2. Identify the Differences Between Components and JavaScript Functions

Aspect	React Component	JavaScript Function
Purpose	Renders UI (using JSX)	Performs logic or returns values
Return Type	Returns JSX	Returns primitive/data values
State & Lifecycle	Supports state and lifecycle (with class/Hooks)	Does not handle UI state
Usage in React	Used to build UIs	Used for logic/utility operations

---

3. Identify the Types of Components

React components can be categorized as:

1. **Class Components**
    - Use ES6 class syntax
    - Have lifecycle methods
    - Use this.state and this.props
2. **Function Components**
    - Simple JavaScript functions
    - Use React Hooks for state/lifecycle
    - Preferred in modern React

---

4. Explain Class Component

A **class component** is a React component written as a JavaScript class that extends React.Component. It must include a render() method and can maintain its own state.

Example:

```
class Welcome extends React.Component {
  render() {
    return <h1>Hello, {this.props.name}</h1>;
  }
}
```

---

5. Explain Function Component

A **function component** is a plain JavaScript function that returns JSX. Modern function components can use **Hooks** (useState, useEffect, etc.) to manage state and side effects.

Example:

```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}
```

---

6. Define Component Constructor

In class components, the **constructor** method is used to:

- Initialize local state
- Bind methods to this

It is called before the component is mounted.

Syntax:

```
constructor(props) {
  super(props);
  this.state = { count: 0 };
}
```

---

7. Define render() Function

The render() method is **required** in class components. It returns the JSX that should be displayed in the UI.

Example:

```
render() {
  return <div>Hello World</div>;
}
```

In **function components**, the JSX is returned directly, so there's no need for a render() function.

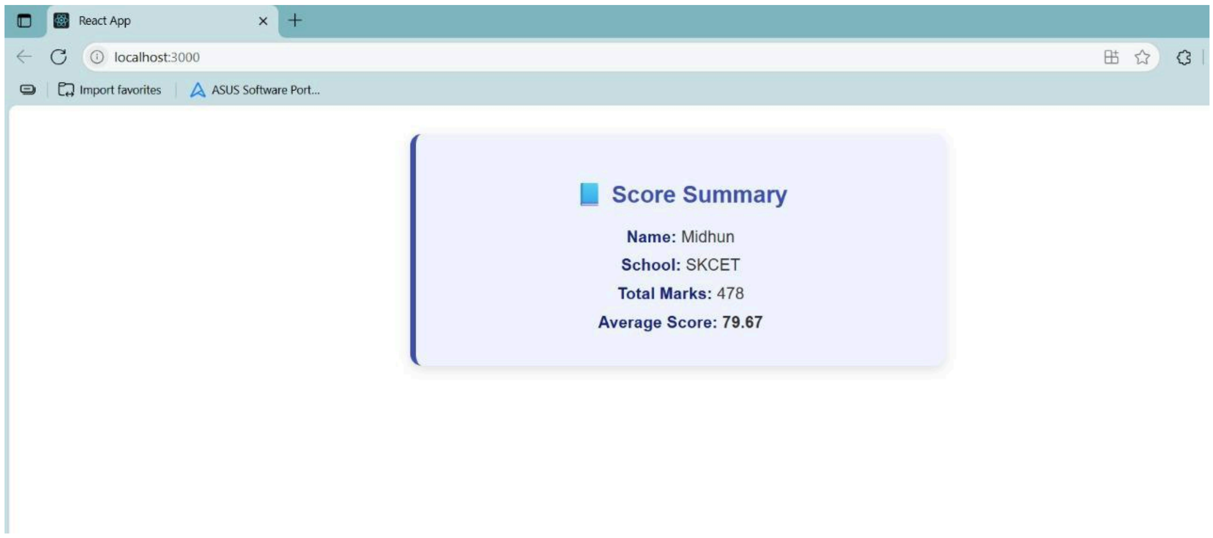
Q.3) Create a react app for Student Management Portal named scorecalculatorapp and create a function component named "CalculateScore" which will accept Name, School, Total and goal in order to calculate the average score of a student and display the same.

CODE:

```
import React from 'react';

import '../Stylesheets/mystyle.css';

function CalculateScore(props) {

  const { name, school, total, goal } = props;

  const average = total / goal;

  return (

    <div className="score-card">

      <h2>📘 Score Summary</h2>

      <p><span>Name:</span> {name}</p>

      <p><span>School:</span> {school}</p>

      <p><span>Total Marks:</span> {total}</p>

      <p><span>Number of Subjects:</span> {goal}</p>

      <p className="highlight"><span>Average Score:</span> {average.toFixed(2)}</p>

    </div>

  );

}

export default CalculateScore;

.score-card {

  background-color: #f0f4ff;

  border-left: 6px solid #3f51b5;

  padding: 25px;

  margin: 30px auto;

  max-width: 500px;

  border-radius: 12px;

  font-family: 'Helvetica Neue', sans-serif;

  box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);

}
```

```
.score-card h2 {
  color: #3f51b5;
  text-align: center;
  margin-bottom: 20px;
}
.score-card p {
  font-size: 17px;
  margin: 10px 0;
  color: #333;
}
.score-card span {
  font-weight: bold;
  color: #1a237e;
}
.highlight {
  color: #1b5e20;
  font-weight: bold;
  font-size: 18px;
}
import React from 'react';
import './App.css';
import CalculateScore from './Components/CalculateScore';
function App() {
  return (
    <div className="App">
      <CalculateScore name="Midhun" school="ABC Public School" total={478} goal={6}
/>
    </div>
  );
}

export default App;
```

OUTPUT



📘 Score Summary

Name: Midhun

School: SKCET

Total Marks: 478

Average Score: 79.67

Q.4) **Objectives – React Component Lifecycle**

1. Explain the Need and Benefits of Component Lifecycle

Need:

React components go through a series of **phases** from creation to removal. Understanding the **component lifecycle** helps developers:

- Control the behavior of components at specific points in their existence.
- Execute logic like data fetching, DOM manipulation, or clean-up at the right time.

Benefits:

- Efficient resource management (e.g., cleaning up intervals or network requests).
- Improved performance through controlled rendering.
- Easier debugging and optimization.
- Better structure and separation of concerns (initialization, updates, unmounting).

---

2. Identify Various Lifecycle Hook Methods

React class components have built-in **lifecycle methods** grouped into three phases:

Mounting ***(component is being created and inserted into the DOM)***

- constructor() – Initializes state and props.
- static getDerivedStateFromProps() – Syncs state from props.
- render() – Returns JSX to render.

- componentDidMount() – Invoked after the component is added to the DOM.

Updating *(when props or state change)*

- static getDerivedStateFromProps()
- shouldComponentUpdate() – Decides if re-render is needed.
- render()
- getSnapshotBeforeUpdate() – Captures values before the update (e.g., scroll position).
- componentDidUpdate() – Called after the update is applied.

🗑️ **Unmounting *(component is being removed from DOM)***

- componentWillUnmount() – Clean up before removal (e.g., cancel timers or remove listeners).

In **function components**, we use **Hooks** like useEffect() to mimic lifecycle behavior.

Q.4)

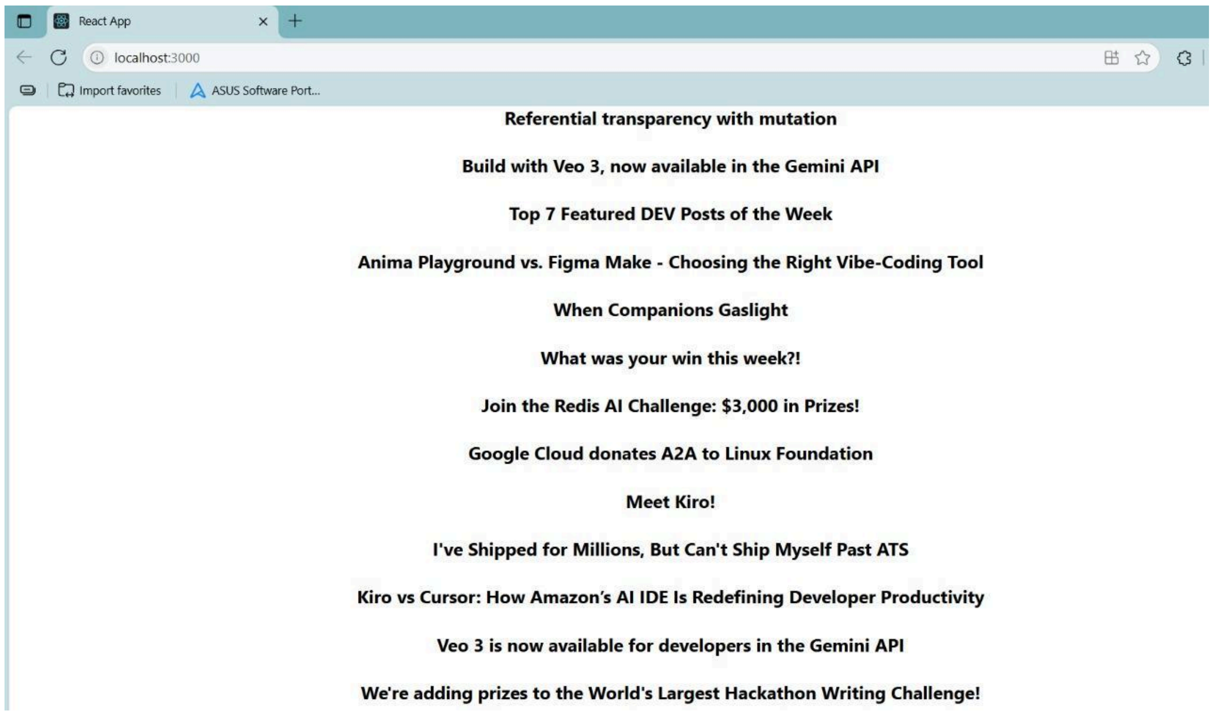
1. Create a new react application using *create-react-app* tool with the name as “blogapp”
2. Open the application using VS Code
3. Create a new file named as **Post.js** in **src folder** with following properties

CODE:

```
import React, { Component } from 'react';
import Post from './Post';
class Posts extends Component {
  constructor(props) {
    super(props);
    this.state = {
      posts: [],
      hasError: false,
    };
  }
  loadPosts = async () => {
    try {
const response = await fetch('https://dev.to/api/articles?per_page=10');      const data =
await response.json();
      const posts = data.map(item => new Post(item.id, item.title, item.body));
      this.setState({ posts });
    } catch (error) {
      console.error("Error fetching posts:", error);
      this.setState({ hasError: true });
    }
  };
  componentDidMount() {
    this.loadPosts();
  }
  componentDidCatch(error, info) {
    alert('Something went wrong while rendering the component.');
    console.error("Error caught:", error, info);
    this.setState({ hasError: true });
  }
  render() {
    if (this.state.hasError) {
      return <h2>Error occurred while displaying posts.</h2>;
    }
    return (
      <div>
        <h1>Blog Posts</h1>
        {this.state.posts.map(post => (
          <div key={post.id} style={{ marginBottom: '25px' }}>
            <h3>{post.title}</h3>
            <p>{post.body}</p>
          </div>
        ))}
      </div>
    );
  }
}
export default Posts;
```

```
class Post {
  constructor(id, title, body) {
    this.id = id;
    this.title = title;
    this.body = body;
  }
}
export default Post;
import React from 'react';
import './App.css';
import Posts from './Posts';
function App() {
  return (
    <div className="App">
      <Posts />
    </div>
  );
}
export default App;
```

OUTPUT:



Referential transparency with mutation

Build with Veo 3, now available in the Gemini API

Top 7 Featured DEV Posts of the Week

Anima Playground vs. Figma Make - Choosing the Right Vibe-Coding Tool

When Companions Gaslight

What was your win this week?!

Join the Redis AI Challenge: $3,000 in Prizes!

Google Cloud donates A2A to Linux Foundation

Meet Kiro!

I've Shipped for Millions, But Can't Ship Myself Past ATS

Kiro vs Cursor: How Amazon's AI IDE Is Redefining Developer Productivity

Veo 3 is now available for developers in the Gemini API

We're adding prizes to the World's Largest Hackathon Writing Challenge!

Q.5)OBJECTIVES

1. Understanding the Need for Styling React Components

Why Styling is Needed:

Styling in React is essential to:

- Make components **visually appealing** and **user-friendly**.
- Maintain **consistency** in UI across the app.
- Support **responsive design** for different devices.
- Highlight interactivity (e.g., hover, focus, active states).

Benefits:

- **Better UX/UI** – Clear layout and visual hierarchy.
- **Component Reusability** – Styles can be modular and scoped.
- **Maintainability** – Easy to manage styles per component.

---

2. Working with CSS Module and Inline Styles

CSS Modules

CSS Modules allow for **locally scoped** CSS files, avoiding class name conflicts.

Usage Example:

1. Create a CSS file with the .module.css extension:

```
/* Button.module.css */
.btn {
  background-color: blue;
  color: white;
}
```

2. Import and use it in your component:

```
import styles from './Button.module.css';
function Button() {
  return <button className={styles.btn}>Click Me</button>;
}
```

Benefits:

- Scoped styles to components
- No class name conflicts
- Easy to maintain in large projects

Inline Styles

React allows you to apply styles directly using the style attribute as a JavaScript object.

Example:

```
function Heading() {
  const headingStyle = {
    color: 'green',
    fontSize: '24px',
  };
  return <h1 style={headingStyle}>Welcome!</h1>;
}
```

Benefits:

- Useful for dynamic styles
- No external CSS file needed
- Good for quick styling or conditional changes

Drawbacks:

- No pseudo-classes like :hover
- Not ideal for complex styles

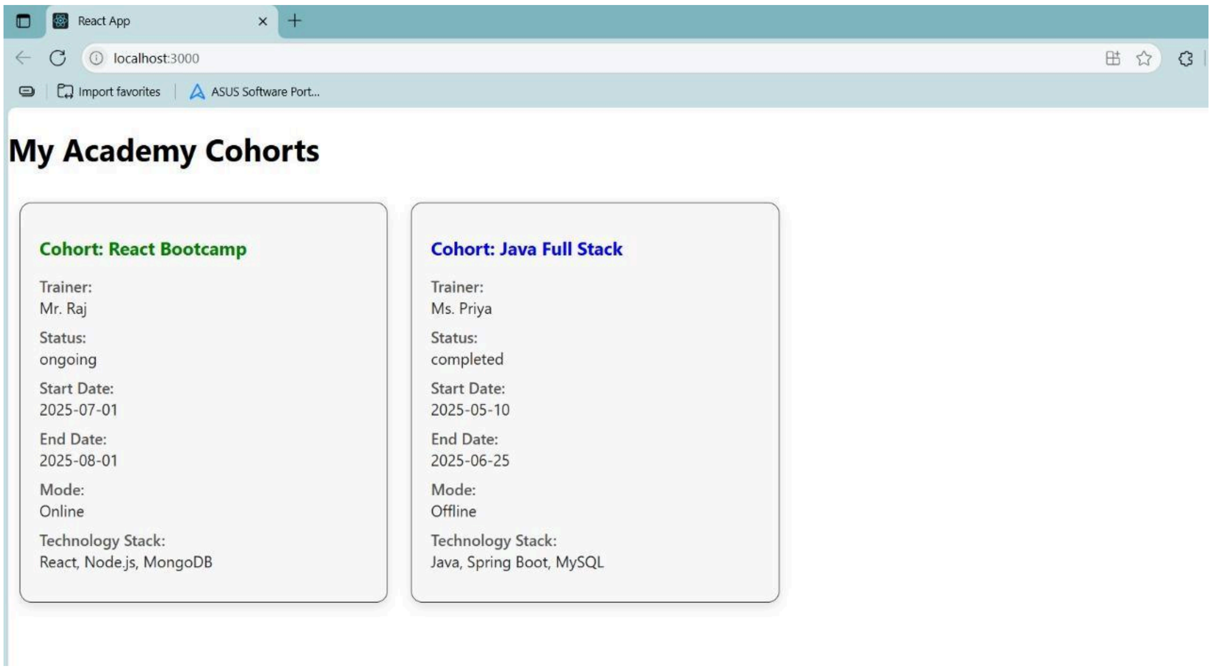
Q.5)My Academy team at Cognizant want to create a dashboard containing the details of ongoing and completed cohorts. A react application is created which displays the detail of the cohorts using react component. You are assigned the task of styling these react components.

CODE:

```
import React from 'react';

import './App.css';

import CohortDetails from './Components/CohortDetails'; // Update path if needed

function App() {

  return (

    <div className="App">

      <h1>My Academy Cohorts</h1>

      <CohortDetails

        name="React Bootcamp"

        trainer="Mr. Raj"

        status="ongoing"

        startDate="2025-07-01"

        endDate="2025-08-01"

        mode="Online"

        techStack="React, Node.js, MongoDB"

      />

      <CohortDetails

        name="Java Full Stack"

        trainer="Ms. Priya"

        status="completed"

        startDate="2025-05-10"

        endDate="2025-06-25"
        mode="Offline"
        techStack="Java, Spring Boot, MySQL"
      />
    </div>
  );
}
export default App;
.box {
  width: 340px;
  display: inline-block;
  margin: 12px;
  padding: 14px 20px;
  border: 1px solid #333;
  border-radius: 12px;
  background-color: #f9f9f9;
  box-shadow: 0px 4px 10px rgba(0,0,0,0.1);
}
dt {
  font-weight: 600;
  margin-top: 8px;
  color: #555;
}
dd {
  margin: 0 0 6px 0;
  color: #222;
}
```

```
import React from 'react';

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

function CohortDetails({ name, trainer, status, startDate, endDate, mode, techStack }) {

  const headingStyle = {

    color: status === 'ongoing' ? 'green' : 'blue',

    marginBottom: '15px',

  };

  return (

    <div className={styles.box}>

      <h3 style={headingStyle}>Cohort: {name}</h3>

      <dl>

        <dt>Trainer:</dt>

        <dd>{trainer}</dd>

        <dt>Status:</dt>

        <dd>{status}</dd>

        <dt>Start Date:</dt>

        <dd>{startDate}</dd>

        <dt>End Date:</dt>

        <dd>{endDate}</dd>

        <dt>Mode:</dt>

        <dd>{mode}</dd>

        <dt>Technology Stack:</dt>

        <dd>{techStack}</dd>

      </dl>

    </div>

  );
}

export default CohortDetails;
```

OUTPUT:



My Academy Cohorts

Cohort: React Bootcamp

Trainer:
Mr. Raj

Status:
ongoing

Start Date:
2025-07-01

End Date:
2025-08-01

Mode:
Online

Technology Stack:
React, Node.js, MongoDB

Cohort: Java Full Stack

Trainer:
Ms. Priya

Status:
completed

Start Date:
2025-05-10

End Date:
2025-06-25

Mode:
Offline

Technology Stack:
Java, Spring Boot, MySQL