# Exercise 1

**Week 6 ReactJS**

**Rubina S SupersetID:6383615**

**Objectives**

* Define SPA and its benefits
* Define React and identify its working
* Identify the differences between SPA and MPA
* Explain Pros & Cons of Single-Page Application
* Explain about React
* Define virtual DOM
* Explain Features of React

# Answers

1. **Define SPA and its benefits**
   * A Single-Page Application (SPA) is a web application that interacts with the user by rewriting the current page instead of loading new pages from the server.
   * Benefits: Faster performance after the first load, smooth user experience, reduced network usage, and quicker transitions between views.

# Define React and identify its working

* + React is an open-source JavaScript library used for building user interfaces, primarily for dynamic and interactive web apps.
  + It works by creating reusable UI components and updating the UI efficiently through a virtual DOM comparison and reconciliation process.

# Identify the differences between SPA and MPA

* + SPA: Renders a single HTML page and updates content dynamically using JavaScript without reloading the page.
  + MPA (Multi-Page Application): Loads a new HTML page from the server on each user interaction, resulting in full page refreshes.

# Explain Pros & Cons of Single-Page Application

* + Pros: Faster client-side performance, seamless navigation, reduced server round-trips, and offline capabilities with caching.
  + Cons: Slower initial load time, more reliance on JavaScript, SEO limitations without SSR, and complex state management.

# Explain about React

* + React allows developers to build modern, interactive web interfaces using a component-based approach.
  + It uses declarative programming and automatically manages UI updates when the underlying data changes, making development efficient.

# Define virtual DOM

* + The virtual DOM is a lightweight, in-memory representation of the actual DOM used by React to optimize UI updates.
  + React compares the virtual DOM with the previous version (diffing) and updates only the changed parts in the real DOM (reconciliation).

# Explain Features of React

* + Declarative UI programming and reusable component-based architecture.
  + Features include JSX syntax, virtual DOM, unidirectional data flow, React Hooks, and support for third-party libraries and tools.

# 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.

Steps Followed:

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



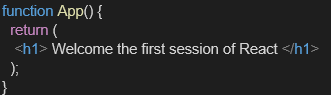
1. To create a React Application with the name of “myfirstreact”, type the following command:



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



1. Open the folder of myfirstreact in Visual Studio Code
2. Open the App.js file in Src Folder of myfirstreact
3. Remove the current content of “App.js”
4. Replace it with the following:



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



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

# Code

function App() { return ( <div>

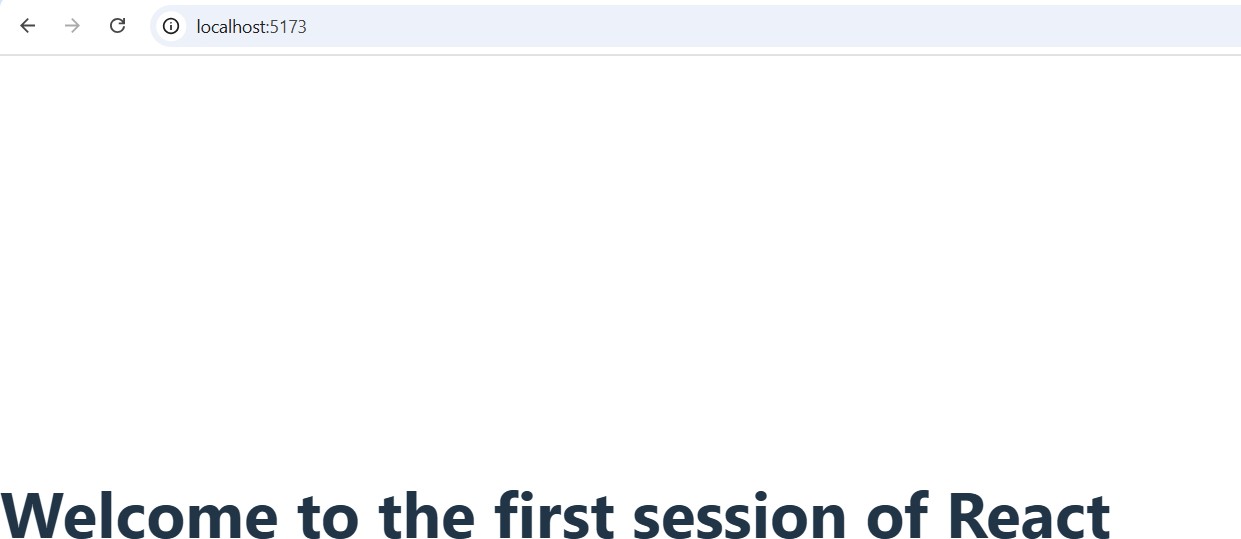
<h1>Welcome to the first session of React</h1>

</div> );

}

export default App;

# Output

****

**Exercise 2** **Objectives**

* Explain React components
* Identify the differences between components and JavaScript functions
* Identify the types of components
* Explain class component
* Explain function component
* Define component constructor
* Define render() function

# Answers

1. **Explain React components**
   * React components are reusable, self-contained building blocks used to create UI in React applications.
   * Each component returns JSX and can manage its own state and behavior.

# Identify the differences between components and JavaScript functions

* + Both can accept arguments (props), but **React components** return JSX and interact with React features like state, lifecycle, and hooks.
  + **JavaScript functions** are general-purpose code blocks and don’t inherently support UI rendering or React-specific features.

# Identify the types of components

* + There are **two main types** of React components:

**Class Components** – ES6 classes that extend React.Component.

**Function Components** – JavaScript functions that return JSX (can use Hooks for state/lifecycle).

# Explain class component

* + A class component is a React component defined using a JavaScript class.
  + It must extend React.Component and include a render() method to return JSX. class Welcome extends React.Component {

render() {

return <h1>Hello, {this.props.name}</h1>;

}

}

# Explain function component

* + A function component is a simpler way to write components using a plain JavaScript function.
  + It returns JSX and can use React Hooks to manage state and lifecycle.

function Welcome(props) {

return <h1>Hello, {props.name}</h1>;}

# Define component constructor

* + The constructor is a special method in class components used to initialize state and bind methods.
  + It is called once when the component is created and must call super(props). constructor(props) {

super(props);

this.state = { count: 0 };

}

# Define render() function

* + The render() function is mandatory in class components and returns the JSX to be displayed.
  + React calls render() to determine what to show on the screen. render() {

return <div>Hello World</div>;

}

# 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.

**Code Home.jsx**

function Home() { return (

<div>

<h2>Home Component</h2>

<p>Welcome to the Home page!</p>

</div>

);

}

export default Home;

**About.jsx** function About() { return (

<div>

<h2>About Component</h2>

<p>This is the About page.</p>

</div>

);

}

export default About;

**Contact.jsx** function Contact() { return (

<div>

<h2>Contact Component</h2>

<p>Get in touch with us here.</p>

</div>

);

}

export default Contact;

# App.jsx

import Home from './components/Home'; import About from './components/About'; import Contact from './components/Contact'; function App() {

return (

<div>

<h1>Welcome to Student App</h1>

<Home />

<About />

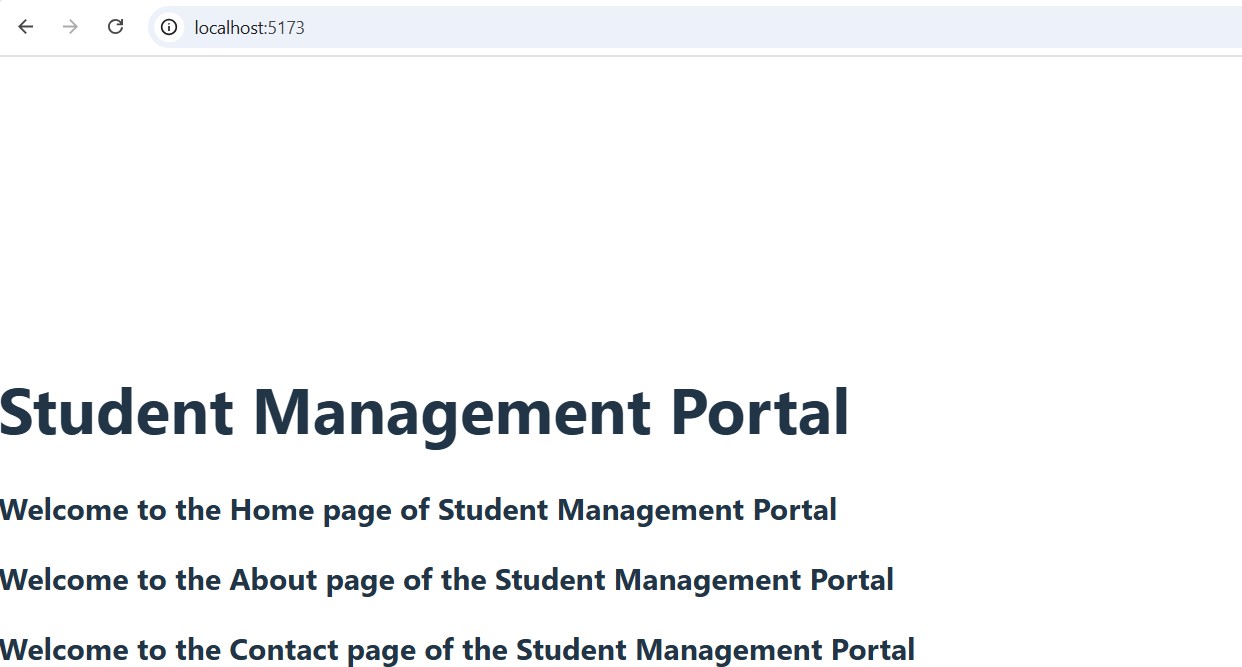
<Contact />

</div>

);

}export default App;

# Output

****

**Exercise 3 Objectives**

* Explain React components
* Identify the differences between components and JavaScript functions
* Identify the types of components
* Explain class component
* Explain function component
* Define component constructor
* Define render() function

# Answers

1. **Explain React Components**

React components are independent, reusable pieces of UI in a React app. They take input (props) and return JSX that describes what should appear on the screen.

# Identify the Differences Between Components and JavaScript Functions

JavaScript functions perform logic and return values like numbers or strings. React components return JSX and are used to render UI; they follow specific rules and integrate with the React framework.

# Identify the Types of Components

There are two types of components in React:

* 1. **Class Components** – Use ES6 classes and lifecycle methods.
  2. **Function Components** – Use JavaScript functions and can use React hooks for state and side effects.

# Explain Class Component

A class component is defined using a JavaScript class that extends React.Component. It must have a render() method and can manage state and lifecycle methods.

# Explain Function Component

A function component is a simple JavaScript function that returns JSX. It uses hooks like useState and useEffect to manage state and behavior.

# Define Component Constructor

The constructor is a method in class components used to initialize state and bind methods. It is called once when the component is created and often includes super(props).

# Define render() Function

The render() function is used in class components to return JSX. React calls it to determine what should be displayed on the screen.

# 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** **CalculateScore.js**

import React from 'react';

import '../stylesheets/mystyle.css'; export function CalculateScore(props) {

const { name, school, total, goal } = props; const average = (total / goal).toFixed(2); return (

<div className="container">

<h1>Student Details</h1>

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

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

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

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

<p><strong>Average Score:</strong> {average}</p>

</div>

);

}

# mystyle.css

.container {

border: 2px solid #4CAF50; padding: 20px;

margin: 40px auto; width: 80%;

font-family: Arial, sans-serif; background-color: #f0f9ff; border-radius: 10px;

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

}

.container h1 { color: #333;

text-align: center;

margin-bottom: 20px;}

.container p { font-size: 18px; margin: 10px 0;

}

# App.js

import './App.css';

import { CalculateScore } from './components/CalculateScore'; function App() {

return (

<div className="App">

<CalculateScore name="Rubina" school="ABC Public School" total={284}

goal={3}

/>

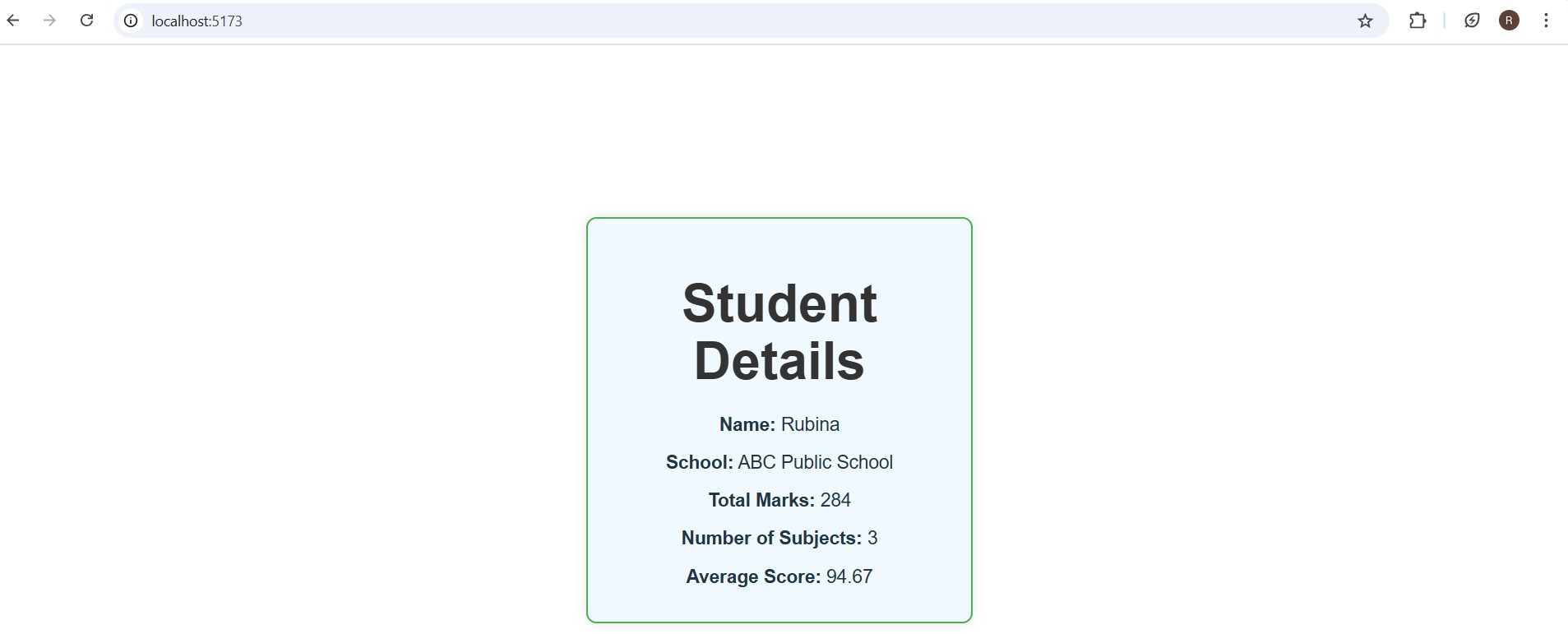
</div>

);

}

export default App;

# Output

****

**Exercise 4 Objectives**

* Explain the need and Benefits of component life cycle
* Identify various life cycle hook methods
* List the sequence of steps in rendering a component

# Answers

1. **Need and Benefits of Component Lifecycle**
   * Helps perform tasks at specific stages like mounting, updating, and unmounting.
   * Allows data fetching, cleanup, or DOM updates at the right time.
   * Improves performance and resource management.
   * Aids in debugging and controlling component behavior effectively.

# Lifecycle Hook Methods

* + **Class Components:** constructor(), render(), componentDidMount(), componentWillUnmount() etc.
  + **Function Components:** Use hooks like useEffect() and useLayoutEffect().
  + Hooks replace class lifecycle methods and simplify logic.
  + Run during mounting, updating, and unmounting phases.

# Sequence of Steps in Rendering a Component

* + **Mounting:** constructor() → render() → componentDidMount()
  + **Updating:** render() → componentDidUpdate() (or useEffect())
  + **Unmounting:** componentWillUnmount() (or cleanup in useEffect())
  + React handles these steps in order for every component lifecycle.

# Code [Posts.js](http://posts.js/)

import React, { Component } from 'react'; class Posts extends Component { constructor(props) {

super(props);

this.state = { posts: [], hasError: false

};

}

loadPosts() { fetch('https://jsonplaceholder.typicode.com/posts')

.then(res => res.json())

.then(data => {

this.setState({ posts: data.slice(0, 10) }); // Load top 10 posts

})

.catch(error => {

console.error("Failed to fetch posts:", error); this.setState({ hasError: true });

});

}

componentDidMount() { this.loadPosts();

componentDidCatch(error, info) {

alert("Something went wrong while loading the posts.");

console.log("Error:", error); console.log("Info:", info);

}

render() {

if (this.state.hasError) {

return <h2>Unable to display posts due to an error.</h2>;

}

return (

<div style={{ padding: '20px' }}>

<h2>Blog Posts</h2>

{this.state.posts.map(post => (

<div key={post.id} style={{ marginBottom: '20px' }}>

<h3>{post.title}</h3>

<p>{post.body}</p>

<hr />

</div>

))}

</div>

);

}

}

export default Posts;

# App.jsx

import React from 'react'; import './App.css';

import Posts from './Posts'; function App() {

return (

<div className="App">

<h1>Welcome to the Blog Application</h1>

<Posts />

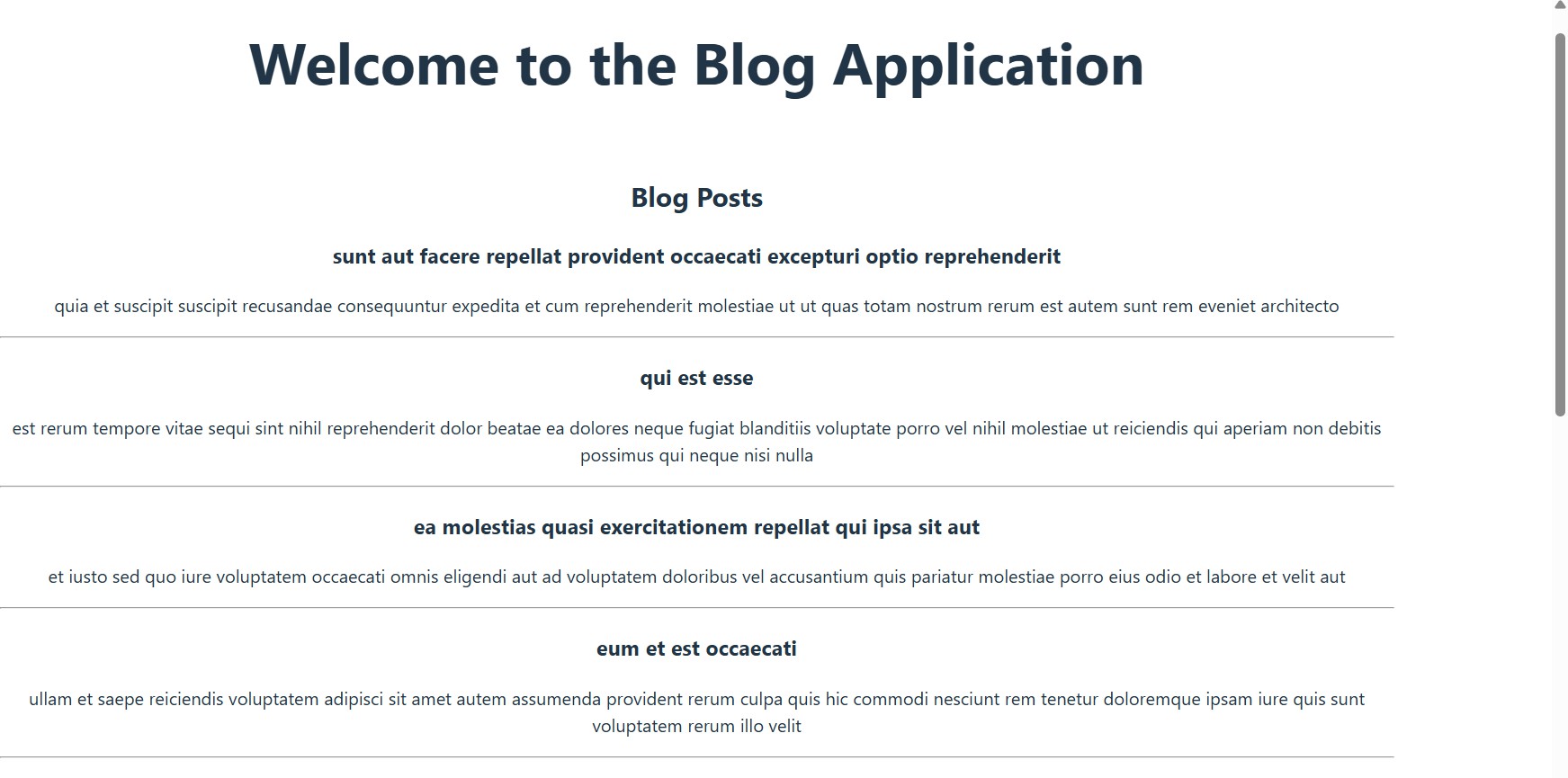
</div>

);

}

export default App;

# Output

****

**Exercise 5 Objectives**

* Understanding the need for styling react component
* Working with CSS Module and inline styles

# Answers

1. **Understanding the Need for Styling React Components**
   * Enhances the visual appeal and usability of the application.
   * Helps maintain consistent design across components.
   * Separates concerns between logic and presentation.
   * Improves user experience and interaction clarity.

# Working with CSS Modules and Inline Styles

* + CSS Modules provide scoped styling to avoid class name conflicts.
  + Inline styles use JavaScript objects to style elements directly.
  + CSS Modules are ideal for reusable and organized styles.
  + Inline styles are useful for dynamic or conditional styling.

# 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** **Cohort.js** class Cohort {

constructor(cohortCode, startDate, technology, trainerName, coachName,

currentStatus) {

this.cohortCode = cohortCode; this.coachName = coachName; this.trainerName = trainerName; this.technology = technology; this.startDate = startDate; this.currentStatus = currentStatus;

}

}

const CohortsData = [

new Cohort('INTADMDF10', '22-Feb-2022', '.NET FSD', 'Jojo Jose', 'Aathma', 'Scheduled'),

new Cohort('ADM21JF014', '10-Sep-2021', 'Java FSD', 'Elisa Smith', 'Apoorv', 'Ongoing'),

new Cohort('CDBJF21025', '24-Dec-2021', 'Java FSD', 'John Doe', 'Aathma', 'Ongoing'),

new Cohort('INTADMJF12', '22-Feb-2022', 'Java FSD', 'To Be Assigned', 'Ibrahim', 'Scheduled'),

new Cohort('CDE22JF011', '24-Dec-2021', 'Java FSD', 'Emma Swan', 'Apoorv', 'Ongoing'),

new Cohort('INTADMDF09', '22-Feb-2022', 'Dataware Housing', 'Babjee Rao', 'Aathma', 'Scheduled'),

new Cohort('ADM22DF001', '10-Sep-2021', '.NET FSD', 'Marie Curie', 'Ibrahim', 'Ongoing'),

];

export { Cohort, CohortsData };

# 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;

}

# CohortDetails.js

import React from 'react';

import styles from './CohortDetails.module.css'; function CohortDetails({ cohort }) {

return (

<div className={styles.box}>

<h3 style={{ color: cohort.currentStatus === 'Ongoing' ? 'green' : 'blue' }}>

{cohort.cohortCode}

</h3>

<dl>

<dt>Start Date</dt>

<dd>{cohort.startDate}</dd>

<dt>Technology</dt>

<dd>{cohort.technology}</dd>

<dt>Trainer</dt>

<dd>{cohort.trainerName}</dd>

<dt>Coach</dt>

<dd>{cohort.coachName}</dd>

<dt>Status</dt>

<dd>{cohort.currentStatus}</dd>

</dl>

</div>

);

}

export default CohortDetails;

# App.js

import React from 'react';

import CohortDetails from './CohortDetails'; import { CohortsData } from './Cohort'; function App() {

return (

<div style={{ textAlign: 'center' }}>

<h1>Cohort Tracker</h1>

{CohortsData.map((cohort, index) => (

<CohortDetails key={index} cohort={cohort} />

))}

</div>

);

}

export default App;

**Output**

