**1.React Hands-on Lab – Getting Started**

**Objectives**

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

**Theory**

**1. What is SPA (Single Page Application)?**

A **SPA** loads a single HTML page and dynamically updates it without full page reloads.

**Benefits:**

* Faster user experience
* Smooth navigation
* Less load on the server

**2.What is React?**

**React** is a JavaScript library developed by Meta for building user interfaces.

**How it works:**

* Uses components
* Manages a virtual DOM
* React updates only the necessary parts of the UI efficiently

**3.SPA vs MPA**

| **Feature** | **SPA** | **MPA** |
| --- | --- | --- |
| Page Reloads | No | Yes |
| Speed | Faster after first load | Slower, reloads each page |
| SEO | More difficult | Easier |
| Complexity | More frontend logic | More backend rendering |

**4.Virtual DOM**

The **Virtual DOM** is an in-memory representation of the real DOM. React updates only what’s changed using an efficient diffing algorithm.

**5.Features of React**

* Component-based architecture
* Virtual DOM
* Reusable UI components
* JSX (JavaScript XML)
* Fast rendering
* Unidirectional data flow

**Hands-on Steps**

**Step 1: Install Node.js and npm**

* Download and install from: <https://nodejs.org/en/download>
* Open Command Prompt and check:

node -v

npm -v

**Step 2: Install Create-React-App globally:** npm install -g create-react-app

**Step 3: Create the React app:** npx create-react-app myfirstreact

**Step 4: Navigate into your app:** cd myfirstreact

**Step 5: Open in Visual Studio Code:** code .

**Step 6: Edit App.js**

* Go to src/App.js
* Delete everything
* Replace with:

import React from 'react';

function App() {

return (

<div>

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

</div>

);

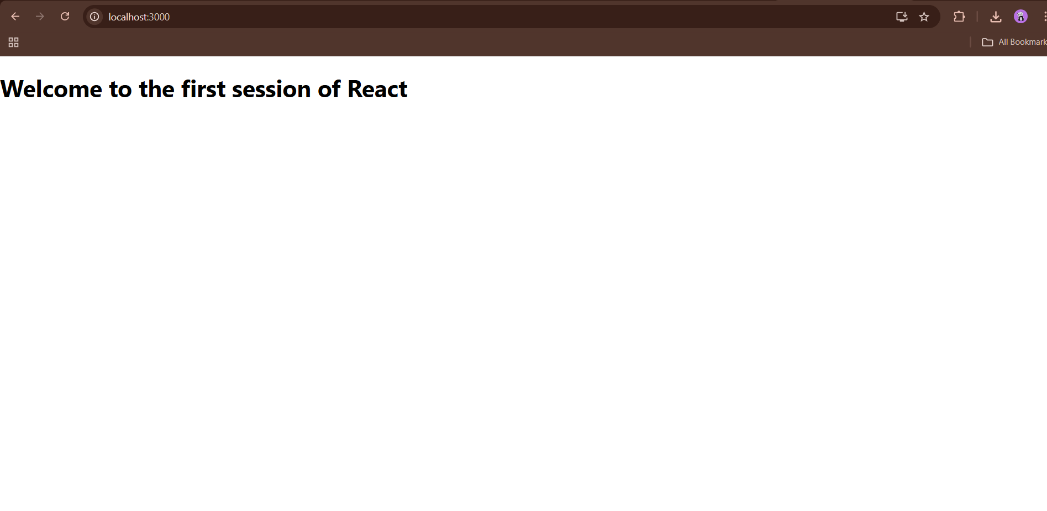
}

export default App;

**Step 7: Start the app:** npm start

**Step 8: View in Browser:** <http://localhost:3000>

**OUTPUT**



**2.React Hands-on Lab**

**Project: Student Management Portal**

**App Name: student-app**

**Objectives**

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

**Theory**

**1.What are React Components?**

* Components are the **building blocks of a React UI**. They let you split the UI into **independent, reusable pieces**.

**2.Difference: React Component vs JavaScript Function**

| **React Component** | **JavaScript Function** | |
| --- | --- | --- |
| Returns JSX | Returns values | |
| Has lifecycle methods | No lifecycle support | |
| Can be reused with props | Limited reusability | |
| Tied to UI | General-purpose logic | |
|  |  | |
|  | |

**Types of Components**

1. **Function Component** – Uses function keyword or arrow function
2. **Class Component** – Uses ES6 class syntax and extends React.Component

**Class Component**

A class component:

* Must extend React.Component
* Has a render() method
* Can have a constructor

import React, { Component } from 'react'

class Example extends Component {

render() {

return <h1>Hello from Class Component</h1>;

}

}

**Function Component**

A function component:

* Is a plain JavaScript function
* Returns JSX directly

function Example() {

return <h1>Hello from Function Component</h1>;

}

**Constructor**

Used to:

* Initialize state
* Bind event handlers

**render() Method**

* In class components, the render() method returns the **JSX UI** that should be displayed.

**Hands-on Lab – student-app**

**Step 1: Create React Project**

Open Visual Studio Code Terminal and run: npx create-react-app StudentApp

**Step 2: Create “Components” Folder**

Inside the src folder:

* Create a new folder called Components

**Step 3: Create Home.js in Components**

* Inside src/Components/Home.js, paste this:

import React, { Component } from 'react';

class Home extends Component {

render() {

return <h2>Welcome to the Home page of Student Management Portal</h2>;

}

}

export default Home;

**Step 4: Create About.js in Components**

Inside src/Components/About.js, paste this:

import React, { Component } from 'react';

class About extends Component {

render() {

return <h2>Welcome to the About page of the Student Management Portal</h2>;

}

}

export default About;

**Step 5: Create Contact.js in Components**

Inside src/Components/Contact.js, paste this:

import React, { Component } from 'react';

class Contact extends Component {

render() {

return <h2>Welcome to the Contact page of the Student Management Portal</h2>;

}

}

export default Contact;

**Step 6: Edit App.js to Call All Components**

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

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">

<Home />

<About />

<Contact />

</div>

);

}

export default App;

**Step 7: Run the Application**

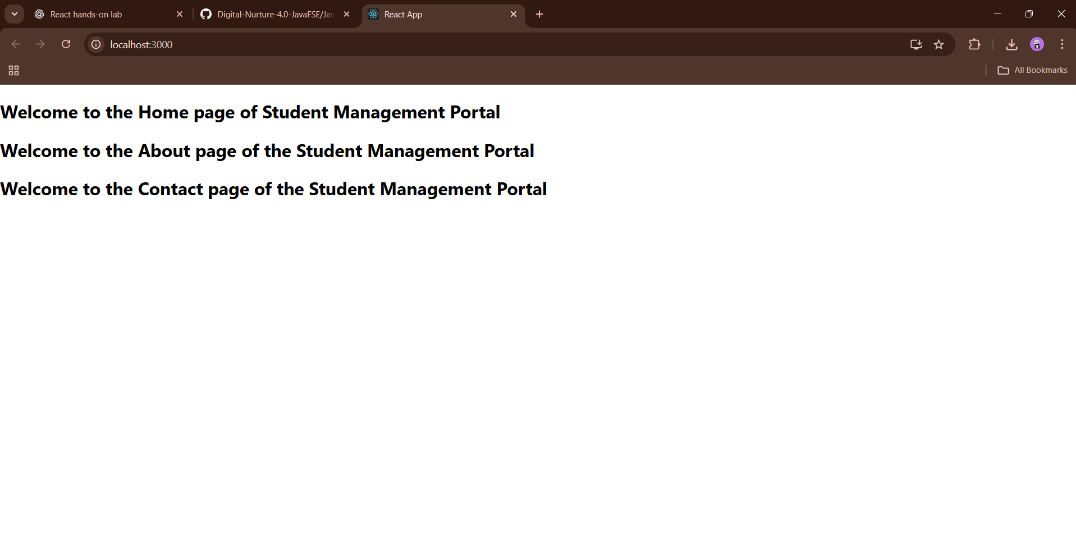
In terminal, navigate into your project: cd student-app

Then start the app:

npm start

**Step 8: Open in Browser :** [**http://localhost:3000**](http://localhost:3000)

**OUTPUT:**

****

**3.Score Calculator React App - Hands-On Lab**

**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

**Task:**

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

**Step 1: Create a React App**

* Open Visual Studio Code terminal and type the following command:

npx create-react-app scorecalculatorapp

**Step 2: Create Components Folder and File**

cd scorecalculatorapp

**Step 3: Add the Functional Component Code**

Inside CalculateScore.js, add the following code:

import React from 'react';

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

function CalculateScore() {

const student = {

name: "John Doe",

school: "XYZ High School",

total: 450,

goal: 500

};

const average = (student.total / student.goal) \* 100;

return (

<div className="container">

<h1>Score Calculator</h1>

<p>Name: {student.name}</p>

<p>School: {student.school}</p>

<p>Total Marks: {student.total}</p>

<p>Goal Marks: {student.goal}</p>

<p>Average Score: {average.toFixed(2)}%</p>

</div>

);

}

export default CalculateScore;

**Step 4: Create Stylesheet**

In the src folder, create a folder called: Stylesheets

Inside it, create a file called: mystyle.css

Add the following styles:

.container {

text-align: center;

font-family: Arial, sans-serif;

padding: 20px;

background-color: #f0f8ff;

border-radius: 8px;

width: 60%;

margin: 20px auto;

box-shadow: 0 4px 8px rgba(0,0,0,0.1);

}

h1 {

color: #333;

}

**Step 5: Modify App.js**

Replace the contents of src/App.js with:

import React from 'react';

import './App.css';

import CalculateScore from './Components/CalculateScore';

function App() {

return (

<div className="App">

<CalculateScore />

</div>

);

}

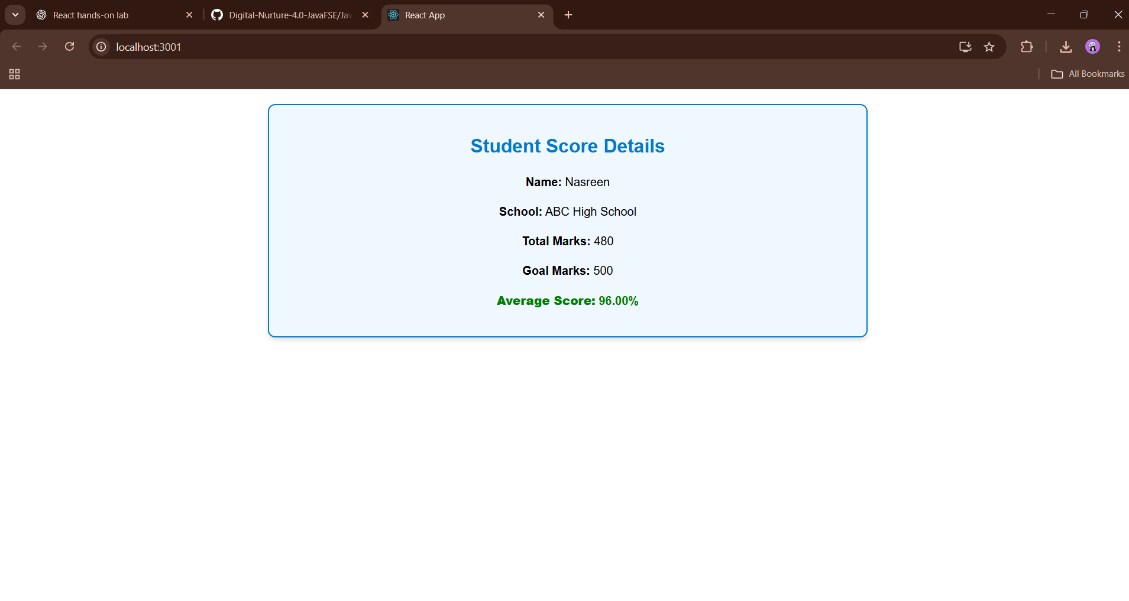
export default App;

**Step 6: Start the React App : npm start**

**Step 7: View in Browser**

**Open your browser and go to:**  **http://localhost:3001**

**OUTPUT**

****

**4.React Lifecycle Lab – Blog Application**

**Objectives**

* Explain the **need and benefits** of the component lifecycle.
* Identify various **lifecycle hook methods**.
* List the **sequence of steps** in rendering a component.

**Prerequisites**

* Node.js
* NPM
* Visual Studio Code

**Step 1: Create a New React App**

* Open a terminal and run:

npx create-react-app blogapp

cd blogapp

code .

**Step 2: Create Post.js in the src folder**

File: src/Post.js

import React from "react";

class Post extends React.Component {

render() {

const { title, body } = this.props;

return (

<div style={{ marginBottom: "20px" }}>

<h3>{title}</h3>

<p>{body}</p>

</div>

);

}

}

export default Post;

**Step 3: Create Posts.js in the src folder**

**File: src/Posts.js**

import React from "react";

import Post from "./Post";

class Posts extends React.Component {

constructor(props) {

super(props);

this.state = {

posts: [],

};

}

loadPosts = () => {

fetch("https://jsonplaceholder.typicode.com/posts")

.then((response) => response.json())

.then((data) => this.setState({ posts: data }))

.catch((error) => {

throw error;

});

};

componentDidMount() {

this.loadPosts();

}

componentDidCatch(error, info) {

alert("An error occurred: " + error.message);

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

}

render() {

const { posts } = this.state;

return (

<div>

<h2>All Posts</h2>

{posts.map((post) => (

<Post key={post.id} title={post.title} body={post.body} />

))}

</div>

);

}

}

export default Posts;

**Step 4: Update App.js to Use the Posts Component**

**File: src/App.js**

import React from "react";

import Posts from "./Posts";

function App() {

return (

<div className="App">

<h1>Blog Application</h1>

<Posts />

</div>

);

}

export default App;

**Step 5: Run the Application**

In your terminal: npm start

**Lifecycle Methods Implemented**

| **Method** | **Purpose** |
| --- | --- |
| constructor() | Initializes the state of the component |
| componentDidMount() | Fetches data once the component has mounted |
| render() | Displays the UI based on the current state |
| componentDidCatch() | Catches and handles any rendering errors |

**Output**

* Titles and bodies of blog posts from <https://jsonplaceholder.typicode.com/posts> are displayed.



**5.Styling React Components Using CSS Module and Inline Styles**

Objectives

* Understand the need for styling in React components
* Work with CSS Modules and inline styles

**Step 1: Create a New React App**

1. Open **Command Prompt** or **Terminal**
2. Run the following command: npx create-react-app cohort-dashboard
3. Navigate to the app folder: cd cohort-dashboard
4. Open the project in **VS Code**: code .

**Step 2: Create Component Folder and File**

1. Inside src/, create a folder:  
   src/components/
2. Inside components/, create a file:  
   CohortDetails.js
3. Add the following code to CohortDetails.js:

import React from 'react';

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

function CohortDetails(props) {

return (

<div className={styles.box}>

<h3 style={{ color: props.status === 'ongoing' ? 'green' : 'blue' }}>

{props.title}

</h3>

<dl>

<dt>Cohort ID</dt>

<dd>{props.id}</dd>

<dt>Status</dt>

<dd>{props.status}</dd>

</dl>

</div>

);

}

export default CohortDetails;

**Step 3: Create and Add CSS Module**

1. In the same components/ folder, create the file:  
   CohortDetails.module.css
2. Add the following CSS code:

.box {

width: 300px;

display: inline-block;

margin: 10px;

padding: 10px 20px;

border: 1px solid black;

border-radius: 10px;

}

dt {

font-weight: 500;

}

**Step 4: Use the Component in App.js**

1. Open the src/App.js file
2. Replace the existing content with:

import React from 'react';

import CohortDetails from './components/CohortDetails';

function App() {

return (

<div>

<CohortDetails title="React Training" id="C101" status="ongoing" />

<CohortDetails title="Node.js Bootcamp" id="C102" status="completed" />

</div>

);

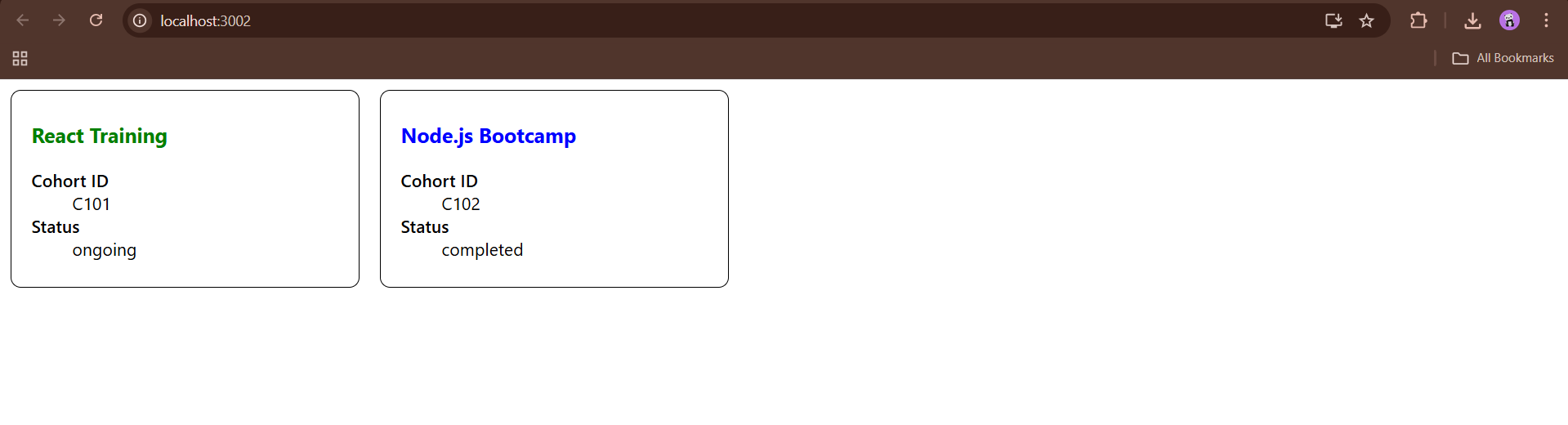
}

export default App;

**Step 5: Start the React App**

Run this in your terminal: npm start

**OUTPUT:**

****