**React - HandsOn**

**1. ReactJs-HOL**

## **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

In this hands-on lab, you will learn how to:

* Set up a react environment
* Use create-react-app

## **Prerequisites**

The following is required to complete this hands-on lab:

* Node.js
* NPM
* Visual Studio Code

## **Notes**

Estimated time to complete this lab: **30 minutes.**

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/>

1. 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



**Answer:**

## **Part 1: Theoretical Concepts**

### SPA (Single Page Application) and Its Benefits

**SPA** is a web application that interacts with the user by dynamically rewriting the current page, rather than loading entire new pages from the server.

**Benefits:**

* Faster navigation
* Better user experience
* Less server load
* Efficient client-side routing

### React and How It Works

**React** is a JavaScript library developed by Facebook for building user interfaces, especially for single-page applications.

**Working:**

* Uses **component-based architecture**
* Uses **Virtual DOM** to optimize rendering
* Enables **unidirectional data flow**

### SPA vs. MPA (Multi Page Application)

| **Feature** | **SPA** | **MPA** |
| --- | --- | --- |
| Page Load | Single HTML file | Multiple HTML pages |
| Speed | Fast after initial load | Reloads page on every click |
| Routing | Client-side routing (React-Router) | Server-side routing |
| Example | Gmail | Amazon, Flipkart |

### Pros & Cons of SPA

**Pros:**

* Fast navigation
* Better user experience
* Efficient caching and development

**Cons:**

* SEO is difficult
* Initial load time may be higher
* Browser history can be tricky

### Virtual DOM

Virtual DOM is a lightweight JavaScript object that is a copy of the real DOM. React compares virtual DOM with the actual DOM using **diffing algorithms** and updates only what is necessary, improving performance.

### Features of React

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

**Code:**

import React from 'react';

function App() {

  return (

    <div style={{ textAlign: 'center', marginTop: '50px' }}>

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

    </div>

  );

}

export default App;

**Output:**

****

**2. ReactJs-HOL**

## **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

In this hands-on lab, you will learn how to:

* Create a class component
* Create multiple components
* Render a component

## **Prerequisites**

The following is required to complete this hands-on lab:

* Node.js
* NPM
* Visual Studio Code

## **Notes**

Estimated time to complete this lab: **30 minutes.**

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:



1. Create a new folder under Src folder with the name “Components”. Add a new file named “Home.js”
2. Type the following code in Home.js



1. Under Src folder add another file named “About.js”
2. 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.
3. Edit the App.js to invoke the Home, About and Contact component as follows:



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



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



**Answer:**

### React Components

React components are the building blocks of any React application. They let you split the UI into reusable pieces.

### Components vs JavaScript Functions

| **Feature** | **JavaScript Function** | **React Component** |
| --- | --- | --- |
| Purpose | Perform logic or calculation | Render UI |
| Return | Any value | JSX |
| Reusability | Usually logic-based | UI + logic together |
| Example | function add(a,b) {} | function Home() { return <h1>Home</h1> } |

### Types of React Components

1. **Functional Components** – Simpler, use hooks (useState, useEffect)
2. **Class Components** – Use constructor, render(), and this.state

### Class Component

A class component is an ES6 class that extends from React.Component and must define a render() method.

class MyComponent extends React.Component {

render() {

return <h1>Hello</h1>;

}

}

### Constructor

The constructor() is used to initialize state or bind methods in class components.

### render() Method

This method is required in a class component and returns the JSX UI.

**Code:**

**Home.js**

import React, { Component } from 'react';

class Home extends Component {

  render() {

    return (

      <div>

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

      </div>

    );

  }

}

export default Home;

**About.js**

import React, { Component } from 'react';

class About extends Component {

  render() {

    return (

      <div>

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

      </div>

    );

  }

}

export default About;

**Contact.js**

// src/Components/Contact.js

import React, { Component } from 'react';

class Contact extends Component {

  render() {

    return (

      <div>

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

      </div>

    );

  }

}

export default Contact;

**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 style={{ textAlign: 'center', marginTop: '50px' }}>

      <h1>Student Management Portal</h1>

      <Home />

      <About />

      <Contact />

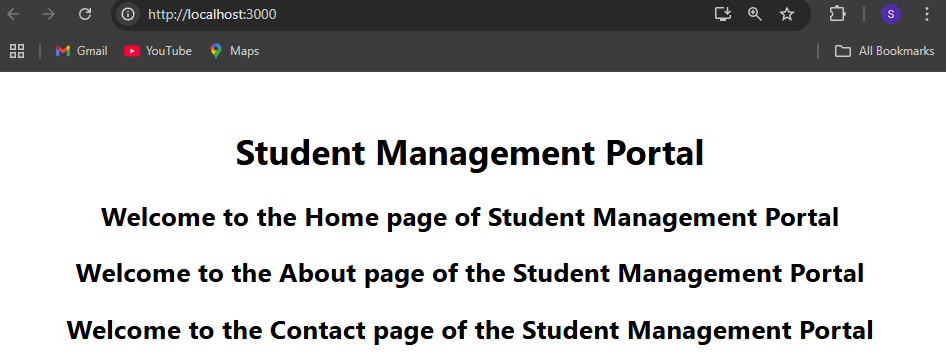
    </div>

  );

}

export default App;

**Output:**

****

**3. ReactJs-HOL**

## **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

In this hands-on lab, you will learn how to:

* Create a function component
* Apply style to components
* Render a component

## **Prerequisites**

The following is required to complete this hands-on lab:

* Node.js
* NPM
* Visual Studio Code

## **Notes**

Estimated time to complete this lab: **30 minutes.**

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:



1. Create a new folder under Src folder with the name “Components”. Add a new file named “CalculateScore.js”
2. Type the following code in CalculateScore.js





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



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



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



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



**Answer:**

### React Components

React components are the building blocks of any React application. They let you split the UI into reusable pieces.

### Components vs JavaScript Functions

| **Feature** | **JavaScript Function** | **React Component** |
| --- | --- | --- |
| Purpose | Perform logic or calculation | Render UI |
| Return | Any value | JSX |
| Reusability | Usually logic-based | UI + logic together |
| Example | function add(a,b) {} | function Home() { return <h1>Home</h1> } |

### Types of React Components

1. **Functional Components** – Simpler, use hooks (useState, useEffect)
2. **Class Components** – Use constructor, render(), and this.state

### Class Component

A class component is an ES6 class that extends from React.Component and must define a render() method.

class MyComponent extends React.Component {

render() {

return <h1>Hello</h1>;

}

}

### Constructor

The constructor() is used to initialize state or bind methods in class components.

### render() Method

This method is required in a class component and returns the JSX UI.

**Code:**

**CalculateScore.js**

import React from 'react';

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

function CalculateScore(props) {

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

  const average = (total / goal).toFixed(2);

  return (

    <div className="score-container">

      <h2 className="score-heading">Student Details:</h2>

      <p className="detail-line"><span className="label name">Name:</span> {name}</p>

      <p className="detail-line"><span className="label school">School:</span> {school}</p>

      <p className="detail-line"><span className="label total">Total:</span> {total}Marks</p>

      <p className="detail-line"><span className="label score">Score:</span>{average}%</p>

    </div>

  );

}

export default CalculateScore;

**mystyle.css**

/\* mystyle.css \*/

.score-container {

  text-align: center;

  font-family: Arial, sans-serif;

  margin-top: 40px;

}

.score-heading {

  font-size: 28px;

  font-weight: bold;

  color: darkred;

  margin-bottom: 20px;

}

.detail-line {

  font-size: 18px;

  margin: 5px 0;

}

.detail-line .label {

  font-weight: bold;

}

.name {

  color: blue;

}

.school {

  color: crimson;

}

.total {

  color: purple;

}

.score {

  color: green;

}

**App.js**

import React from 'react';

import CalculateScore from './Components/CalculateScore';

function App() {

  return (

    <div>

      <CalculateScore name="Steeve" school="DNV Public School" total={284} goal={3} />

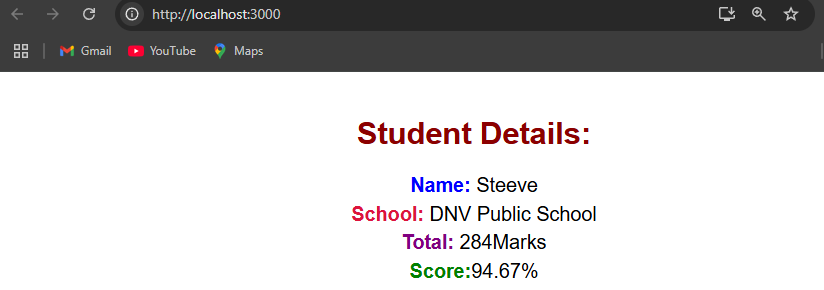
    </div>

  );

}

export default App;

**Output:**

****

**4. ReactJs-HOL**

**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

In this hands-on lab, you will learn how to:

* Implement componentDidMount() hook
* Implementing componentDidCatch() life cycle hook.

## **Prerequisites**

The following is required to complete this hands-on lab:

* Node.js
* NPM
* Visual Studio Code

## **Notes**

Estimated time to complete this lab: **60 minutes.**

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



Figure 2: Post class

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



Figure 3: Posts Component

1. Initialize the component with a list of Post in state of the component using the constructor
2. 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>)



Figure 4: loadPosts() method

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



Figure 5: componentDidMount() hook

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



Figure 6: render() method

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



Figure 7: componentDidCatch() hook

1. Add the Posts component to App component.
2. Build and Run the application using *npm start* command.

**Answer:**

### Why Component Lifecycle Is Important?

* Manages what happens **before, during, and after** a component is rendered
* Helps in **data fetching**, **DOM manipulation**, **cleanup**, and **error handling**

### Key Lifecycle Hook Methods

1. constructor() – Initializes state and binds methods
2. componentDidMount() – Called after component is mounted; good for data fetching
3. render() – Renders the UI
4. componentDidCatch() – Catches errors in child components and logs or displays them

**Code:**

**Post.js**

import React, { Component } from 'react';

class Post extends Component {

  render() {

    const { title, body } = this.props;

    return (

      <div style={{ border: '1px solid #ccc', padding: '10px', marginBottom: '15px' }}>

        <h3>{title}</h3>

        <p>{body}</p>

      </div>

    );

  }

}

export default Post;

**Posts.js**

// src/Posts.js

import React, { Component } from 'react';

import Post from './Post';

class Posts extends Component {

  constructor(props) {

    super(props);

    this.state = {

      posts: [],

      error: null

    };

  }

  componentDidMount() {

    this.loadPosts();

  }

  loadPosts() {

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

      .then(response => {

        if (!response.ok) {

          throw new Error('Network response was not OK');

        }

        return response.json();

      })

      .then(data => this.setState({ posts: data.slice(0, 10) })) // load first 10 posts

      .catch(error => this.setState({ error }));

  }

  componentDidCatch(error, info) {

    alert("An error occurred while loading the posts.");

    console.error("Error caught by componentDidCatch:", error, info);

  }

  render() {

    const { posts, error } = this.state;

    if (error) {

      return <h2>Error loading posts!</h2>;

    }

    return (

      <div style={{ width: '80%', margin: '0 auto' }}>

        <h2>Blog Posts</h2>

        {posts.map(post => (

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

        ))}

      </div>

    );

  }

}

export default Posts;

**App.js**

import React from 'react';

import Posts from './Posts';

function App() {

  return (

    <div className="App">

      <h1 style={{ textAlign: 'center' }}>Welcome to BlogApp</h1>

      <Posts />

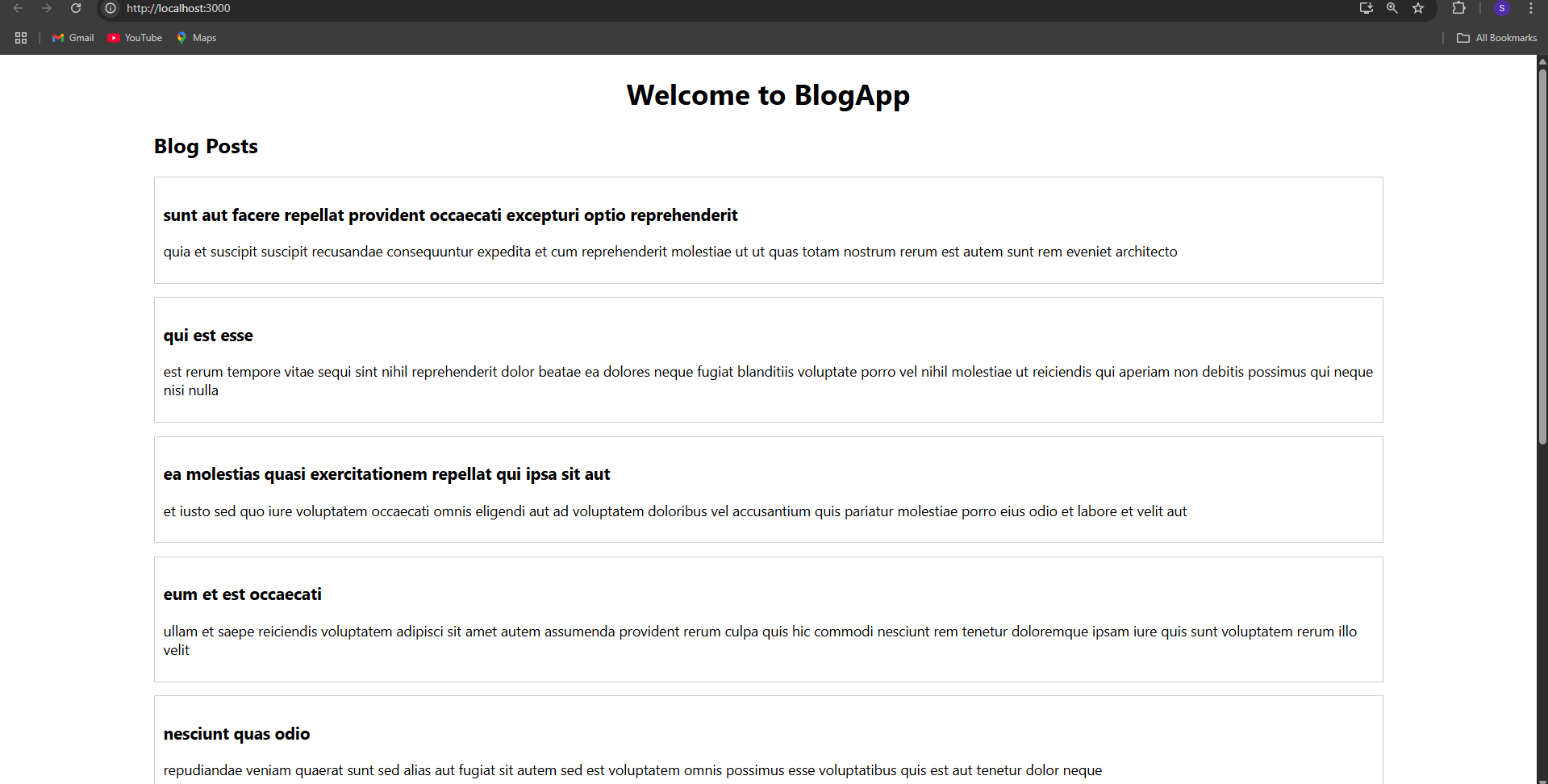
    </div>

  );

}

export default App;

**Output:**

****

**5. ReactJs-HOL**

**Objectives**

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

In this hands-on lab, you will learn how to:

* Style a react component
* Define styles using the CSS Module
* Apply styles to components using className and style properties

## **Prerequisites**

The following is required to complete this hands-on lab:

* Node.js
* NPM
* Visual Studio Code

## **Notes**

Estimated time to complete this lab: **30 minutes.**

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



Figure 1: Restore packages

1. Open the application using VS Code
2. Create a new CSS Module in a file called “CohortDetails.module.css”
3. 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*

1. Define a css style for html <dt> element using tag selector. Set the font weight to 500.
2. Open the cohort details component and import the CSS Module
3. Apply the box class to the container div
4. Define the style for <h3> element to use “green” color font when cohort status is “ongoing” and “blue” color in all other scenarios.
5. Final result should look similar to the below image



Figure 2: Final Result

**Code:**

**CohortDetails.module.css**

.container {

  display: flex;

  flex-wrap: wrap;

  gap: 20px;

  margin-top: 20px;

}

.box {

  width: 300px;

  border: 1px solid black;

  border-radius: 10px;

  padding: 10px 20px;

  box-sizing: border-box;

  background-color: white;

}

dt {

  font-weight: 500;

}

**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.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>

    </div>

  );

}

export default CohortDetails;

**App.js**

import React from 'react';

import CohortDetails from './Components/CohortDetails';

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

function App() {

  const cohorts = [

    {

      name: 'INTADMDF10 - .NET FSD',

      startDate: '22-Feb-2022',

      status: 'Scheduled',

      coach: 'Aathma',

      trainer: 'Jojo Jose',

    },

    {

      name: 'ADM21JF014 - Java FSD',

      startDate: '10-Sep-2021',

      status: 'Ongoing',

      coach: 'Apoorv',

      trainer: 'Elisa Smith',

    },

    {

      name: 'CDBJF21025 - Java FSD',

      startDate: '24-Dec-2021',

      status: 'Ongoing',

      coach: 'Aathma',

      trainer: 'John Doe',

    },

  ];

  return (

    <div style={{ padding: '20px' }}>

      <h2>Cohorts Details</h2>

      <div className={styles.container}>

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

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

        ))}

      </div>

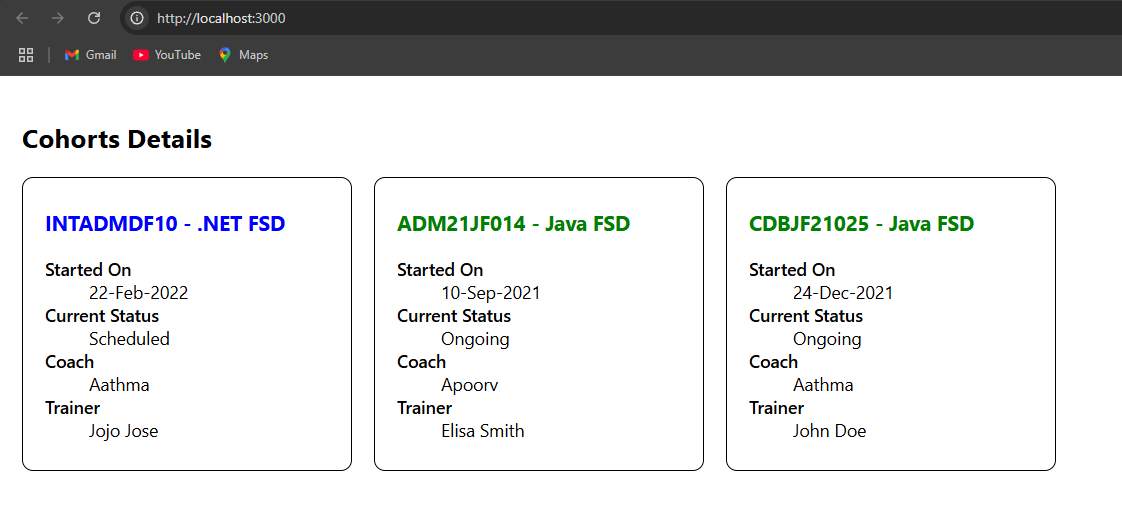
    </div>

  );

}

export default App;

**Output:**

****

**9. ReactJs-HOL**

## **Objectives**

* List the features of ES6
* Explain JavaScript let
* Identify the differences between var and let
* Explain JavaScript const
* Explain ES6 class fundamentals
* Explain ES6 class inheritance
* Define ES6 arrow functions
* Identify set(), map()

In this hands-on lab, you will learn how to:

* Use map() method of ES6
* Apply arrow functions of ES6
* Implement Destructuring features of ES6

## **Prerequisites**

The following is required to complete this hands-on lab:

* Node.js
* NPM
* Visual Studio Code

## **Notes**

Estimated time to complete this lab: **60 minutes.**

Create a React Application named “cricketapp” with the following components:

1. ListofPlayers

* Declare an array with 11 players and store details of their names and scores using the map feature of ES6



* Filter the players with scores below 70 using arrow functions of ES6.



1. IndianPlayers
   1. Display the Odd Team Player and Even Team players using the Destructuring features of ES6



* 1. Declare two arrays T20players and RanjiTrophy players and merge the two arrays and display them using the Merge feature of ES6



Display these two components in the same home page using a simple if else in the flag variable.

**Output:**

When Flag=true



When Flag=false



**Hint:**



**Code:**

**ListofPlayers.js**

import React from 'react';

const ListofPlayers = ({ players }) => {

  return (

    <ul>

      {players.map((player, index) => (

        <li key={index}>

          {player.name} - Score: {player.score}

        </li>

      ))}

    </ul>

  );

};

export default ListofPlayers;

**Scorebelow70.js**

import React from 'react';

const Scorebelow70 = ({ players }) => {

  const filteredPlayers = players.filter(player => player.score < 70);

  return (

    <ul>

      {filteredPlayers.map((player, index) => (

        <li key={index}>

          {player.name} - Score: {player.score}

        </li>

      ))}

    </ul>

  );

};

export default Scorebelow70;

**OddPlayers.js**

import React from 'react';

const OddPlayers = (IndianTeam) => {

  const { team } = IndianTeam;

  const oddPlayers = team.filter((\_, index) => index % 2 !== 0);

  return (

    <ul>

      {oddPlayers.map((player, i) => (

        <li key={i}>{player}</li>

      ))}

    </ul>

  );

};

export default OddPlayers;

**EvenPlayers.js**

import React from 'react';

const EvenPlayers = (IndianTeam) => {

  const { team } = IndianTeam;

  const evenPlayers = team.filter((\_, index) => index % 2 === 0);

  return (

    <ul>

      {evenPlayers.map((player, i) => (

        <li key={i}>{player}</li>

      ))}

    </ul>

  );

};

export default EvenPlayers;

**ListofIndianPlayers.js**

import React from 'react';

const ListofIndianPlayers = ({ IndianPlayers }) => {

  return (

    <ul>

      {IndianPlayers.map((player, index) => (

        <li key={index}>{player}</li>

      ))}

    </ul>

  );

};

export default ListofIndianPlayers;

**IndianPlayers.js**

import React from 'react';

import OddPlayers from './OddPlayers';

import EvenPlayers from './EvenPlayers';

import ListofIndianPlayers from './ListofIndianPlayers';

const IndianPlayers = () => {

  const team = ['Virat', 'Rohit', 'Dhoni', 'Rahul', 'Shami', 'Ashwin'];

  const T20players = ['Surya', 'Bumrah'];

  const RanjiPlayers = ['Pujara', 'Jadeja'];

  const IndianPlayers = [...T20players, ...RanjiPlayers];

  return (

    <div>

      <div>

        <h1>Indian Team</h1>

        <h1>Odd Players</h1>

        {OddPlayers({ team })}

        <hr />

        <h1>Even Players</h1>

        {EvenPlayers({ team })}

      </div>

      <hr />

      <div>

        <h1>List of Indian Players Merged:</h1>

        <ListofIndianPlayers IndianPlayers={IndianPlayers} />

      </div>

    </div>

  );

};

export default IndianPlayers;

**App.js**

import React from 'react';

import ListofPlayers from './Components/ListofPlayers';

import Scorebelow70 from './Components/Scorebelow70';

import IndianPlayers from './Components/IndianPlayers';

const App = () => {

  const flag = true;

  const players = [

    { name: 'Virat', score: 85 },

    { name: 'Rohit', score: 45 },

    { name: 'Dhoni', score: 95 },

    { name: 'Rahul', score: 60 },

    { name: 'Shami', score: 68 },

    { name: 'Ashwin', score: 78 },

    { name: 'Surya', score: 88 },

    { name: 'Bumrah', score: 30 },

    { name: 'Pujara', score: 50 },

    { name: 'Jadeja', score: 90 },

    { name: 'Kohli', score: 35 },

  ];

  if (flag === true) {

    return (

      <div>

        <h1>List of Players</h1>

        <ListofPlayers players={players} />

        <hr />

        <h1>List of Players having Scores Less than 70</h1>

        <Scorebelow70 players={players} />

      </div>

    );

  } else {

    return (

      <div>

        <IndianPlayers />

      </div>

    );

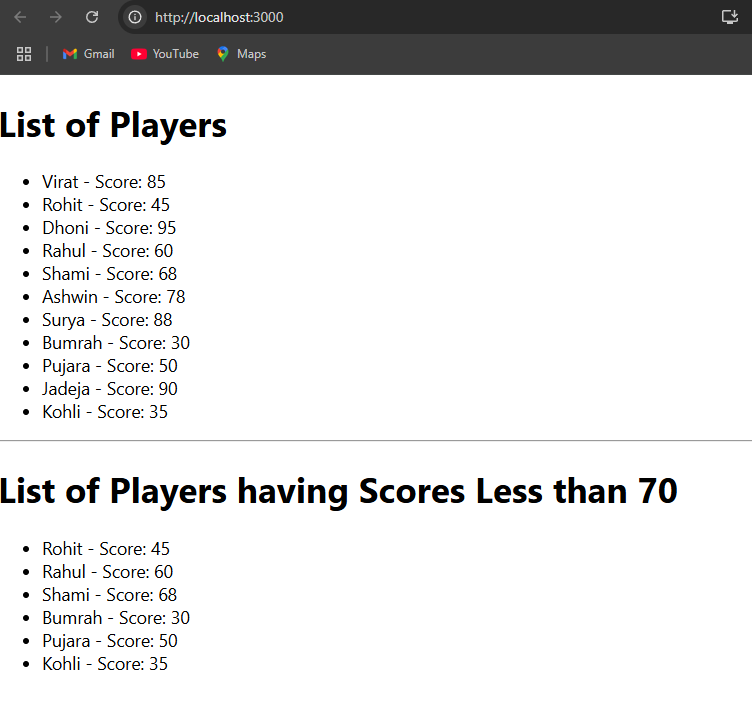
  }

};

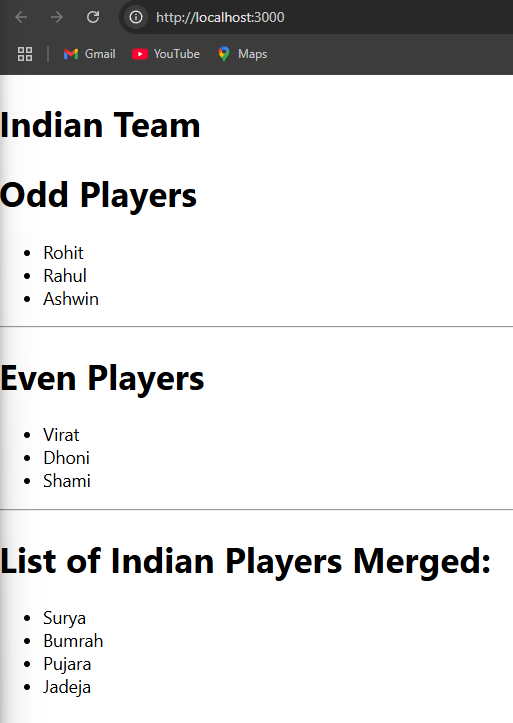
export default App;

**Output:**

**When flag == true**

****

**When flag == false**

****

**10. ReactJs-HOL**

## **Objectives**

* Define JSX
* Explain about ECMA Script
* Explain React.createElement()
* Explain how to create React nodes with JSX
* Define how to render JSX to DOM
* Explain how to use JavaScript expressions in JSX
* Explain how to use inline CSS in JSX

In this hands-on lab, you will learn how to:

* Use JSX syntax in React applications
* Use inline CSS in JSX

## **Prerequisites**

The following is required to complete this hands-on lab:

* Node.js
* NPM
* Visual Studio Code

## **Notes**

Estimated time to complete this lab: **60 minutes.**

Create a React Application named “officespacerentalapp” which uses React JSX to create elements, attributes and renders DOM to display the page.

Create an element to display the heading of the page.

Attribute to display the image of the office space

Create an object of office to display the details like Name, Rent and Address.

Create a list of Object and loop through the office space item to display more data.

To apply Css, Display the color of the Rent in Red if it’s below 60000 and in Green if it’s above 60000.

Output:



**Hint:**





**Answer:**

### 1. ****Define JSX****

**JSX (JavaScript XML)** is a syntax extension for JavaScript that allows you to write HTML-like code inside JavaScript. JSX is used with React to describe what the UI should look like.

**Example:**

const element = <h1>Hello, JSX!</h1>;

It looks like HTML, but it gets transpiled to React.createElement() calls under the hood.

### 2. ****Explain about ECMAScript****

**ECMAScript (ES)** is the standardized version of JavaScript, governed by ECMA International. React commonly uses modern ECMAScript versions like **ES6+**, which include features such as:

* let, const
* Arrow functions: () => {}
* Classes
* Template literals: `Hello ${name}`
* Destructuring, Spread/rest operators

**React code heavily relies on ES6 features.**

### 3. ****Explain**** React.createElement()

This is a method from the React library used to create a React element without using JSX.

**Syntax:**

React.createElement(type, props, ...children)

**Example:**

const element = React.createElement('h1', { className: 'title' }, 'Hello World');

This creates:

<h1 class="title">Hello World</h1>

JSX is just syntactic sugar for React.createElement().

### 4. ****Explain how to create React nodes with JSX****

React nodes are the elements you want to render on the screen. With JSX, you can directly write them like HTML.

**Example:**

const heading = <h1>Welcome to React</h1>;

const list = (

<ul>

<li>Item 1</li>

<li>Item 2</li>

</ul>

);

These JSX elements (nodes) will be compiled into React elements using React.createElement().

### 5. ****Define how to render JSX to the DOM****

You use **ReactDOM.createRoot()** and **root.render()** to attach your React component or JSX to the HTML DOM.

**Example:**

import React from 'react';

import ReactDOM from 'react-dom/client';

const element = <h1>Hello React!</h1>;

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(element);

This renders the element inside the HTML element with id="root".

### 6. ****Explain how to use JavaScript expressions in JSX****

You can embed any **valid JavaScript expression** inside JSX using **curly braces {}**.

**Example:**

const name = "John";

const element = <h1>Hello, {name}!</h1>;

You can also include function calls, ternary operators, and calculations:

<p>{10 + 20}</p>

<p>{isLoggedIn ? "Welcome!" : "Please log in"}</p>

### 7. ****Explain how to use inline CSS in JSX****

In JSX, inline styles are passed as a **JavaScript object**, with camelCase properties instead of hyphenated names.

**Example:**

const style = {

color: 'blue',

fontSize: '20px'

};

const element = <h1 style={style}>Styled Heading</h1>;

You can also apply styles directly:

<h2 style={{ backgroundColor: 'yellow', padding: '10px' }}>Inline CSS</h2>

**Code:**

**App.js**

import React from 'react';

function App() {

  const offices = [

    {

      name: 'Skyline Tower',

      image: '/images/skyline.png',

      location: 'Marathahalli',

      rent: 25000

    },

    {

      name: 'EcoSpace',

      image: '/images/ecospace.png',

      location: 'Bellandur',

      rent: 35000

    },

    {

      name: 'Brigade Tech Park',

      image: '/images/brigade.png',

      location: 'Whitefield',

      rent: 45000

    }

  ];

  return (

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

      <h1 style={{ color: 'green' }}>Office Space Rental App</h1>

      <div style={{

        display: 'flex',

        justifyContent: 'center',

        flexWrap: 'wrap',

        gap: '20px',

        padding: '20px'

      }}>

        {offices.map((office, index) => (

          <div key={index} style={{

            border: '2px solid green',

            borderRadius: '10px',

            padding: '15px',

            width: '250px',

            boxShadow: '0 4px 8px rgba(0, 0, 0, 0.1)'

          }}>

            <img

              src={office.image}

              alt={office.name}

              style={{

                width: '100%',

                height: '160px',

                objectFit: 'cover',

                borderRadius: '5px'

              }}

            />

            <h2>{office.name}</h2>

            <p><strong>Location:</strong> {office.location}</p>

            <p><strong>Rent:</strong> ₹{office.rent}</p>

          </div>

        ))}

      </div>

    </div>

  );

}

export default App;

**index.js**

import React from 'react';

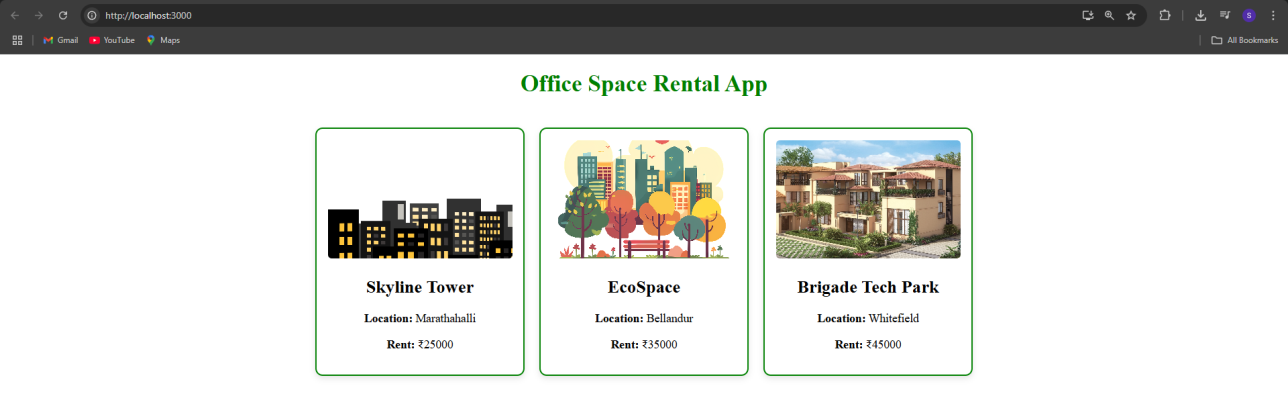
import ReactDOM from 'react-dom/client';

import App from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(<App />);

**Output:**

****

**11. ReactJs-HOL**

## **Objectives**

* Explain React events
* Explain about event handlers
* Define Synthetic event
* Identify React event naming convention

In this hands-on lab, you will learn how to:

* Implement Event handling concept in React applications
* Use this keyword
* Use synthetic event

## **Prerequisites**

The following is required to complete this hands-on lab:

* Node.js
* NPM
* Visual Studio Code

## **Notes**

Estimated time to complete this lab: **90 minutes.**

Create a React Application “eventexamplesapp” to handle various events of the form elements in HTML.

1. Create “Increment” button to increase the value of the counter and “Decrement” button to decrease the value of the counter. The “Increase” button should invoke multiple methods.
   1. To increment the value
   2. Say Hello followed by a static message.



1. Create a button “Say Welcome” which invokes the function which takes “welcome” as an argument.



1. Create a button which invokes synthetic event “OnPress” which display “I was clicked”



Create a “CurrencyConvertor” component which will convert the Indian Rupees to Euro when the Convert button is clicked.

Handle the Click event of the button to invoke the handleSubmit event and handle the conversion of the euro to rupees.

**Answer:**

### 1. ****Explain React Events****

React events are **JavaScript events** that work similarly to DOM events but are handled using React’s **own event system**.  
React wraps native events in a cross-browser wrapper called a **Synthetic Event** for better compatibility and performance.

Common React events include:

* onClick
* onChange
* onSubmit
* onMouseEnter
* onKeyDown

**Example:**

<button onClick={handleClick}>Click Me</button>

### 2. ****Explain About Event Handlers****

Event handlers are **functions that get triggered when an event occurs** on a component (like clicking a button or typing in a textbox).

React uses **camelCase** naming for event props and passes a **function reference**.

**Example:**

function handleClick() {

alert("Button clicked!");

}

<button onClick={handleClick}>Click</button>

The handleClick function is the event handler that runs when the button is clicked.

### 3. ****Define Synthetic Event****

A **SyntheticEvent** is a **wrapper** around the browser’s native event object in React.  
It ensures the event works **consistently across all browsers**.

All events in React (e.g., clicks, form inputs) are passed as SyntheticEvent objects.

**Example:**

function handleInput(event) {

console.log(event.target.value); // SyntheticEvent used here

}

<input type="text" onChange={handleInput} />

Note: Synthetic events are pooled for performance. If you want to access the event asynchronously, use event.persist().

### 4. ****React Event Naming Convention****

React follows **camelCase** naming for all event props, unlike HTML which uses lowercase.

| **HTML Attribute** | **React Equivalent** |
| --- | --- |
| onclick | onClick |
| onchange | onChange |
| onsubmit | onSubmit |

**Example:**

<button onClick={handleClick}>Click</button>

<button onclick="handleClick()">Click</button>

**Code:**

**Counter.js**

import React, { Component } from 'react';

class Counter extends Component {

  constructor(props) {

    super(props);

    this.state = {

      count: 0

    };

    this.sayWelcome = this.sayWelcome.bind(this);

  }

  increment = () => {

    this.setState({ count: this.state.count + 1 });

    this.sayHello();

    this.sayStaticMessage();

  };

  decrement = () => {

    this.setState({ count: this.state.count - 1 });

  };

  sayHello = () => {

    console.log("Hello from increment!");

  };

  sayStaticMessage = () => {

    console.log("This is a static message.");

  };

  sayWelcome(message) {

    alert(message);

  }

  handleSyntheticEvent = (e) => {

    e.preventDefault();

    alert("I was clicked (Synthetic Event)");

  };

  render() {

    return (

      <div>

        <h2>Counter: {this.state.count}</h2>

        <button onClick={this.increment}>Increment</button>

        <button onClick={this.decrement}>Decrement</button>

        <br /><br />

        <button onClick={() => this.sayWelcome("Welcome!")}>Say Welcome</button>

        <br /><br />

        <button onClick={this.handleSyntheticEvent}>OnPress (Synthetic Event)</button>

      </div>

    );

  }

}

export default Counter;

**CurrencyConverter.js**

import React, { useState } from 'react';

function CurrencyConverter() {

  const [amount, setAmount] = useState('');

  const [fromCurrency, setFromCurrency] = useState('INR');

  const [toCurrency, setToCurrency] = useState('EUR');

  const [convertedAmount, setConvertedAmount] = useState(null);

  const exchangeRates = {

    INR: { USD: 0.012, EUR: 0.011 },

    USD: { INR: 83.2, EUR: 0.93 },

    EUR: { INR: 90.5, USD: 1.08 }

  };

  const handleSubmit = (e) => {

    e.preventDefault();

    const amt = parseFloat(amount);

    if (!isNaN(amt)) {

      if (fromCurrency === toCurrency) {

        setConvertedAmount(amt);

      } else {

        const rate = exchangeRates[fromCurrency]?.[toCurrency];

        if (rate) {

          const result = (amt \* rate).toFixed(2);

          setConvertedAmount(result);

        } else {

          setConvertedAmount("Conversion rate not available.");

        }

      }

    } else {

      alert("Please enter a valid number.");

    }

  };

  return (

    <div>

      <h2>Currency Converter</h2>

      <form onSubmit={handleSubmit}>

        <div>

          <label>Amount: </label>

          <input

            type="number"

            placeholder="Enter amount"

            value={amount}

            onChange={(e) => setAmount(e.target.value)}

            required

          />

        </div>

        <div>

          <label>From: </label>

          <select value={fromCurrency} onChange={(e) => setFromCurrency(e.target.value)}>

            <option value="INR">INR</option>

            <option value="USD">USD</option>

            <option value="EUR">EUR</option>

          </select>

        </div>

        <div>

          <label>To: </label>

          <select value={toCurrency} onChange={(e) => setToCurrency(e.target.value)}>

            <option value="INR">INR</option>

            <option value="USD">USD</option>

            <option value="EUR">EUR</option>

          </select>

        </div>

        <button type="submit">Convert</button>

      </form>

      {convertedAmount !== null && (

        <p>

          {amount} {fromCurrency} = {convertedAmount} {toCurrency}

        </p>

      )}

    </div>

  );

}

export default CurrencyConverter;

**App.js**

import React from 'react';

import Counter from './Components/Counter';

import CurrencyConverter from './Components/CurrencyConverter';

function App() {

  return (

    <div style={{ padding: '20px' }}>

      <h1>React Event Examples App</h1>

      <Counter />

      <hr />

      <CurrencyConverter />

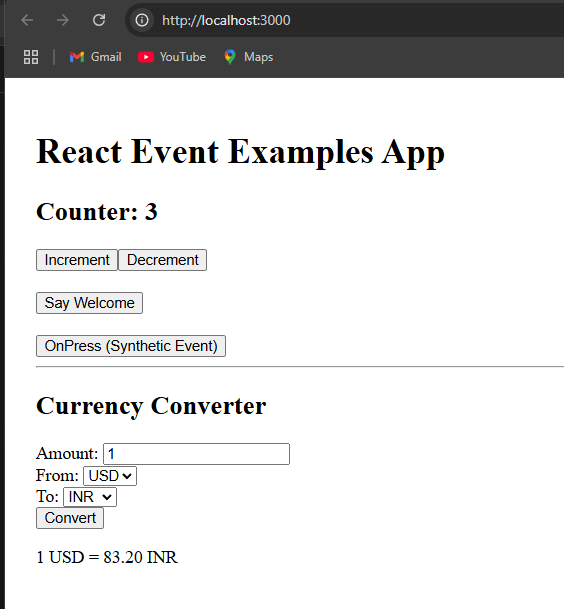
    </div>

  );

}

export default App;

**Output:**

****

**12. ReactJs-HOL**

## **Objectives**

* Explain about conditional rendering in React
* Define element variables
* Explain how to prevent components from rendering

In this hands-on lab, you will learn how to:

* Implement conditional rendering in React applications

## **Prerequisites**

The following is required to complete this hands-on lab:

* Node.js
* NPM
* Visual Studio Code

## **Notes**

Estimated time to complete this lab: **60 minutes.**

Create a React Application named “ticketbookingapp” where the guest user can browse the page where the flight details are displayed whereas the logged in user only can book tickets.

The Login and Logout buttons should accordingly display different pages. Once the user is logged in the User page should be displayed. When the user clicks on Logout, the Guest page should be displayed.





**Hint:**







**Answer:**

### ****1. Explain about Conditional Rendering in React****

**Conditional rendering** means showing or hiding elements/components based on certain conditions (like user logged in or not).  
React allows conditional rendering using:

* if statements
* ternary operators
* && (AND) logical operators
* element variables

**Example:**

{isLoggedIn ? <UserPage /> : <GuestPage />}

### ****2. Define Element Variables****

Element variables store components or elements that can be conditionally rendered.

**Example:**

let content;

if (isLoggedIn) {

content = <UserPage />;

} else {

content = <GuestPage />;

}

return <div>{content}</div>;

### ****3. How to Prevent Components from Rendering****

You can **prevent rendering** by returning null in a component.

**Example:**

function WarningBanner(props) {

if (!props.showWarning) return null;

return <div className="warning">Warning!</div>;

}

**Code:**

**GuestPage.js**

import React from 'react';

function GuestPage() {

  return (

    <div>

      <h2>Welcome, Guest!</h2>

      <p>Please login to book your flight.</p>

      <h3>Available Flights:</h3>

      <ul>

        <li>Flight A123 - Delhi to Mumbai</li>

        <li>Flight B456 - Bangalore to Chennai</li>

        <li>Flight C789 - Kolkata to Pune</li>

      </ul>

    </div>

  );

}

export default GuestPage;

**UserPage.js**

import React from 'react';

function UserPage() {

  return (

    <div>

      <h2>Welcome, User!</h2>

      <p>You can now book your flights.</p>

      <button>Book Flight A123</button>

      <button>Book Flight B456</button>

      <button>Book Flight C789</button>

    </div>

  );

}

export default UserPage;

**index.js**

import React from 'react';

import ReactDOM from 'react-dom/client';

import App from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(<App />);

**App.js**

import React, { useState } from 'react';

function Greeting({ isLoggedIn }) {

  return <h2>{isLoggedIn ? 'Welcome back!' : 'Please sign in.'}</h2>;

}

function LoginButton({ onClick }) {

  return <button onClick={onClick}>Login</button>;

}

function LogoutButton({ onClick }) {

  return <button onClick={onClick}>Logout</button>;

}

function FlightDetails() {

  return (

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

      <h3>Available Flights</h3>

      <ul>

        <li>✈️ Bangalore to Delhi - 8:00 AM</li>

        <li>✈️ Mumbai to Chennai - 2:30 PM</li>

        <li>✈️ Kolkata to Hyderabad - 6:45 PM</li>

      </ul>

    </div>

  );

}

function BookingForm() {

  return (

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

      <h3>Book Your Flight</h3>

      <form>

        <label>

          Passenger Name:

          <input type="text" name="name" required />

        </label>

        <br /><br />

        <label>

          Destination:

          <select required>

            <option value="">Select</option>

            <option value="Delhi">Delhi</option>

            <option value="Chennai">Chennai</option>

            <option value="Hyderabad">Hyderabad</option>

          </select>

        </label>

        <br /><br />

        <label>

          Travel Date:

          <input type="date" name="date" required />

        </label>

        <br /><br />

        <button type="submit">Book Flight</button>

      </form>

    </div>

  );

}

function App() {

  const [isLoggedIn, setIsLoggedIn] = useState(false);

  const handleLoginClick = () => setIsLoggedIn(true);

  const handleLogoutClick = () => setIsLoggedIn(false);

  return (

    <div style={{ textAlign: 'center', marginTop: '50px' }}>

      <Greeting isLoggedIn={isLoggedIn} />

      {isLoggedIn ? (

        <>

          <LogoutButton onClick={handleLogoutClick} />

          <FlightDetails />

          <BookingForm />

        </>

      ) : (

        <LoginButton onClick={handleLoginClick} />

      )}

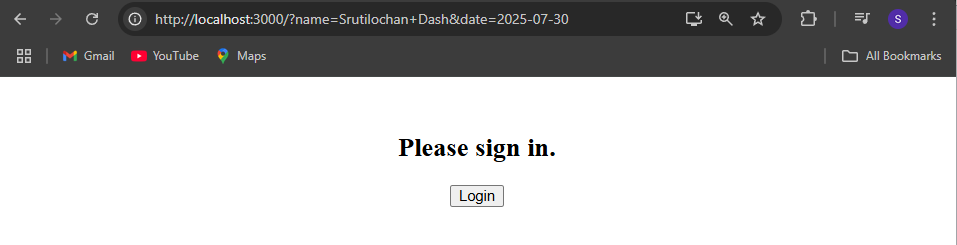
    </div>

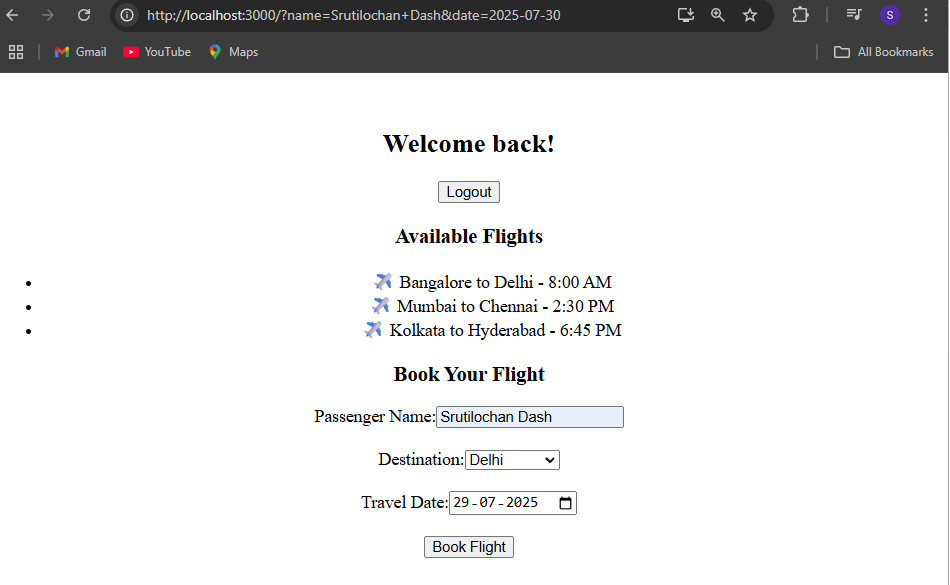
  );

}

export default App;

**Output:**

****

****

**13. ReactJs-HOL**

## **Objectives**

* Explain various ways of conditional rendering
* Explain how to render multiple components
* Define list component
* Explain about keys in React applications
* Explain how to extract components with keys
* Explain React Map, map() function

In this hands-on lab, you will learn how to:

* Implement conditional rendering in React applications

## **Prerequisites**

The following is required to complete this hands-on lab:

* Node.js
* NPM
* Visual Studio Code

## **Notes**

Estimated time to complete this lab: **60 minutes.**

Create a React App named “bloggerapp” in with 3 components.

1. Book Details
2. Blog Details
3. Course Details

Implement this with as many ways possible of Conditional Rendering.



**Hint:**







**Answer:**

### 1. ****Conditional Rendering in React****

React renders UI based on conditions using:

* if/else
* Ternary operator (condition ? true : false)
* Logical && operator
* Switch cases

### 2. ****Render Multiple Components****

You can render multiple components conditionally or together inside one parent:

<>

<BookDetails />

<BlogDetails />

</>

### 3. ****List Component****

A component that displays an array of items using .map():

{items.map((item) => (

<li key={item.id}>{item.title}</li>

))}

### 4. ****Keys in React****

Keys are unique identifiers for list elements to help React optimize rendering.

### 5. ****Extract Components with Keys****

Each child in a list should have a unique key prop when rendering:

<ListItem key={item.id} value={item.value} />

### 6. ****React Map / map() Function****

map() lets you loop over arrays and render elements:

const books = ['React', 'Node'];

const bookList = books.map((book, index) => <li key={index}>{book}</li>);

**Code:**

**BookDetails.js**

import React from 'react';

const BookDetails = () => {

  const books = [

    { id: 1, name: 'React Mastery' },

    { id: 2, name: 'Node Basics' },

    { id: 3, name: 'Advanced JS' },

  ];

  return (

    <div>

      <h3>📚 Book Details</h3>

      <ul>

        {books.map(book => (

          <li key={book.id}>{book.name}</li>

        ))}

      </ul>

    </div>

  );

};

export default BookDetails;

**BlogDetails.js**

import React from 'react';

const BlogDetails = () => {

  const blogs = [

    { id: 'a', title: 'Learning React' },

    { id: 'b', title: 'Understanding Hooks' },

    { id: 'c', title: 'State vs Props' },

  ];

  return (

    <div>

      <h3>📝 Blog Details</h3>

      <ul>

        {blogs.map(blog => (

          <li key={blog.id}>{blog.title}</li>

        ))}

      </ul>

    </div>

  );

};

export default BlogDetails;

**CourseDetails.js**

import React from 'react';

const CourseDetails = () => {

  const courses = ['React', 'Node', 'Express'];

  return (

    <div>

      <h3>🎓 Course Details</h3>

      <ul>

        {courses.map((course, index) => (

          <li key={index}>{course}</li>

        ))}

      </ul>

    </div>

  );

};

export default CourseDetails;

**index.js**

import React from 'react';

import ReactDOM from 'react-dom/client';

import App from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(<App />);

**App.js**

import React, { useState } from 'react';

import BookDetails from './Components/BookDetails';

import BlogDetails from './Components/BlogDetails';

import CourseDetails from './Components/CourseDetails';

function App() {

  const [section, setSection] = useState('book');

  return (

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

      <h1>📘 Blogger App</h1>

      <div style={{ marginBottom: '10px' }}>

        <button onClick={() => setSection('book')}>Show Book</button>

        <button onClick={() => setSection('blog')}>Show Blog</button>

        <button onClick={() => setSection('course')}>Show Course</button>

      </div>

      {section === 'book' && <BookDetails />}

      {section === 'blog' && <BlogDetails />}

      {section === 'course' && <CourseDetails />}

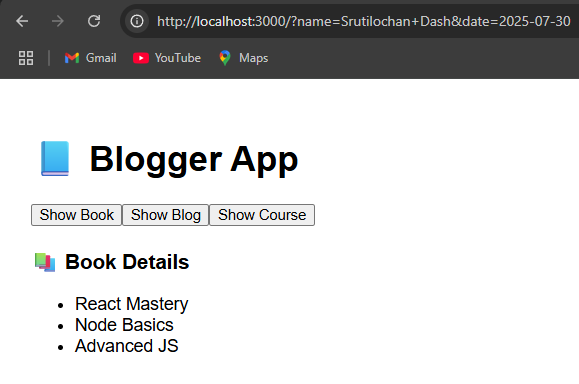
    </div>

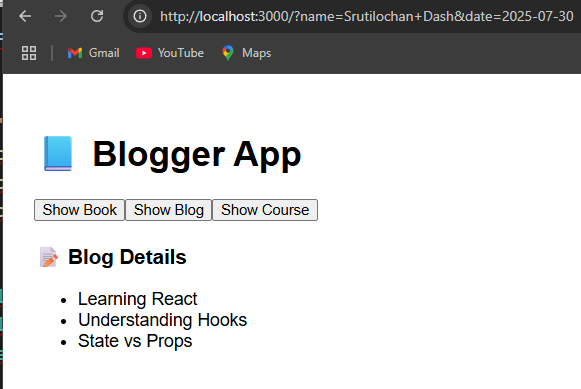
  );

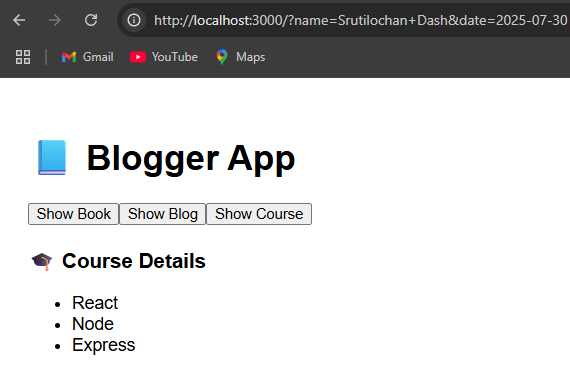
}

export default App;

**Output:**

****

****

****