**Week 6: React JS**

**ReactJS-HOL 1: Creating First React Component**

**Scenario:**

This exercise involves creating a new React application and modifying the main App component to display a simple welcome message. This will ensure your development environment is set up correctly and you understand the basic structure of a React application.

**Procedure:**

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

* + Open your terminal or command prompt.
  + Navigate to the directory where you want to create your project.
  + Run the command npx create-react-app my-first-app. This will create a new folder named my-first-app with a complete React project structure.

**Step 2: Navigate to the Project Directory**

* + Once the creation process is complete, move into the project directory by running cd my-first-app.

**Step 3: Modify the App.js file**

* + Open the src/App.js file in your code editor.
  + Replace the existing code with the code provided in the implementation section below. This will change the component to render a simple <h1> tag.

**Step 4: Start the Development Server**

* + In your terminal (while inside the my-first-app directory), run the command npm start.
  + This will launch a development server, and your new React application will automatically open in your default web browser.

**Step 5: Verify the Output**

* + The browser should display the message: "Welcome the first session of React, Vivek!".

**Implementation:**

**App.js**

function App() {

  return (

    <h1> Welcome the first session of React, Vivek! </h1>

  );

}

export default App;

**Index.js**

import React from 'react';

import ReactDOM from 'react-dom/client';

import './index.css';

import App from './App';

import reportWebVitals from './reportWebVitals';

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

root.render(

  <React.StrictMode>

    <App />

  </React.StrictMode>

);

reportWebVitals();

**Index.css**

body {

  margin: 0;

  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', 'Oxygen',

    'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',

    sans-serif;

  -webkit-font-smoothing: antialiased;

  -moz-osx-font-smoothing: grayscale;

}

code {

  font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',

    monospace;

}

**Output:**

A screenshot of a computer

AI-generated content may be incorrect.

**ReactJS-HOL 2**

**Scenario:**

In this exercise, you will build a basic structure for a "Student Management Portal". You'll learn how to create multiple class components for different 'pages' (Home, About, Contact) and then compose them together in the main App component to build a simple, multi-section user interface.

**Procedure:**

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

* + Open your terminal and run the following command to create a new React project named StudentApp:
  + npx create-react-app StudentApp

**Step 2: Create a Components Folder**

* + Navigate into your new project directory: cd StudentApp.
  + Inside the src folder, create a new folder named Components.

**Step 3: Create the Home Component**

* + Inside the src/Components folder, create a new file named Home.js.
  + Add the code from the implementation section below to this file.

**Step 4: Create the About Component**

* + In the same src/Components folder, create a new file named About.js.
  + Add the corresponding code to this file.

**Step 5: Create the Contact Component**

* + Create a final component file named Contact.js in the src/Components folder.
  + Add its code to the file.

**Step 6: Update the Main App Component**

* + Open the src/App.js file.
  + Modify it to import and display the Home, About, and Contact components.

**Step 7: Start the Application:**

* + In your terminal, run the command npm start.
  + Your browser will open to http://localhost:3000, displaying the content from all three components.

**Implementation:**

**components/Home.js**

import React, { Component } from 'react';

class Home extends Component {

  render() {

    return (

      <div>

        <h3>Welcome to the Home Page of Student Management Portal</h3>

      </div>

    );

  }

}

export default Home;

**components/About.js**

import React, { Component } from 'react';

class About extends Component {

  render() {

    return (

      <div>

        <h3>Welcome to the About Page of the Student Management Portal</h3>

      </div>

    );

  }

}

export default About;

**components/Contact.js**

import React, { Component } from 'react';

class Contact extends Component {

  render() {

    return (

      <div>

        <h3>Welcome to the Contact Page of the Student Management Portal</h3>

        <h3>Done by Vivek S</h3>

      </div>

    );

  }

}

export default Contact;

**App.js**

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;

A screenshot of a computer

AI-generated content may be incorrect.**Output:**

**ReactJS-HOL 3**

**Scenario:**

For the Student Management Portal, you need to create a reusable React component that displays individual student details. This component will accept student information (like name, school, and marks) as props, calculate the percentage score, and display all the information with custom styling applied from an external CSS file.

**Procedure:**

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

* + Open your terminal and run the command to create a new project named scorecalculatorapp:
  + npx create-react-app scorecalculatorapp

**Step 2: Create the Component File**

* + Navigate into the project directory: cd scorecalculatorapp.
  + Inside the src folder, create a new folder named Components.
  + Inside src/Components, create a new file named CalculateScore.js.

**Step 3: Implement the CalculateScore Component**

* + Add the provided JavaScript code to the CalculateScore.js file. This component will receive props and calculate the score.

**Step 4: Create the Stylesheet**

* + Inside the src folder, create a new folder named Stylesheets.
  + Inside src/Stylesheets, create a new file named mystyle.css.
  + Add the provided CSS rules to this file to style the component.

**Step 5: Update the Main App Component**

* + Open the src/App.js file.
  + Modify it to import the CalculateScore component and its stylesheet.
  + Use the CalculateScore component and pass the student data to it as props.

**Step 6: Start the Application**

* + In your terminal, run the command npm start.
  + Your browser will open and display the styled student details.

**Implementation:**

**Stylesheets/mystyle.css**

Name {

  font-weight: 300;

  color: blue;

}

School {

  color: crimson;

}

Total {

  color: darkmagenta;

}

formatstyle {

  text-align: center;

  font-size: large;

}

Score {

  color: forestgreen;

}

**Components/CalculateScore.js**

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

const percentToDecimal = (decimal) => {

  return (decimal.toFixed(2) + "%");

}

const calcScore = (total, goal) => {

  return percentToDecimal(total / goal);

}

export const CalculateScore = ({ Name, School, total, goal }) => {

  return (

    <div className="formatstyle">

      <h1><font color="brown">Student Details:</font></h1>

      <div className="Name">

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

      </div>

      <div className="School">

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

      </div>

      <div className="Total">

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

      </div>

      <div className="Score">

        <p><span> Score: </span> <b>{calcScore(total, goal)}</b></p>

      </div>

    </div>

  );

}

**App.js**

import './App.css';

import { CalculateScore } from './Components/CalculateScore';

function App() {

  return (

    <div>

      <CalculateScore

        Name={"Vivek S"}

        School={"Vel Tech Rangarajan Dr.Sagunthala R&D Institute of Science and Technology"}

        total={360}

        goal={3}

      />

    </div>

  );

}

A screenshot of a computer

AI-generated content may be incorrect.export default App;

**Output:**

**ReactJS-HOL 4**

**Scenario:**

In this hands-on lab, you will create a React application that fetches a list of blog posts from a public API and displays them on the page. You will learn how to use class components, manage state, fetch data using the fetch API in the componentDidMount lifecycle method, and handle potential errors using componentDidCatch.

**Procedure:**

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

* + Open your terminal and create a new React project with the name blogapp:
  + npx create-react-app blogapp

**Step 2: Create the Post Class**

* + Navigate into your project directory: cd blogapp.
  + Inside the src folder, create a new file named Post.js.
  + Define a simple Post class in this file to structure the data for each post.

**Step 3: Create the Posts Component**

* + In the src folder, create another new file named Posts.js.
  + Implement a class component named Posts.
  + Initialize the component's state in the constructor to hold a list of posts.
  + Create a loadPosts() method to fetch data from https://jsonplaceholder.typicode.com/posts.
  + Call loadPosts() within the componentDidMount() lifecycle method.
  + Implement the render() method to display the fetched posts.
  + Add a componentDidCatch() method to handle any rendering errors.

**Step 4: Update the App Component**

* + Open the src/App.js file.
  + Import and render the Posts component.

**Step 5: Run the Application**

* + In your terminal, start the development server:
  + npm start
  + Open your browser to see the list of blog posts fetched from the API.

**Implementation:**

**Post.js**

class Post {

  constructor(id, title, body) {

    this.id = id;

    this.title = title;

    this.body = body;

  }

}

export default Post;

**Posts.js**

import React from 'react';

import Post from './Post';

class Posts extends React.Component {

  constructor(props) {

    super(props);

    this.state = {

      posts: [],

      error: null

    };

  }

  loadPosts() {

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

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

      .then((data) => {

        const postList = data.map(

          (item) => new Post(item.id, item.title, item.body)

        );

        this.setState({ posts: postList });

      })

      .catch((error) => {

        this.setState({ error: "Failed to fetch posts" });

      });

  }

  componentDidMount() {

    this.loadPosts();

  }

  componentDidCatch(error, info) {

    this.setState({ error: "An unexpected error occurred." });

  }

  render() {

    const { posts, error } = this.state;

    if (error) {

      return <div style={{ color: 'red' }}>{error}</div>;

    }

    return (

      <div>

        <h2>Blog Posts</h2>

        {posts.map((post) => (

          <div key={post.id}>

            <h3>{post.title}</h3>

            <p>{post.body}</p>

            <hr />

          </div>

        ))}

      </div>

    );

  }

}

export default Posts;

**App.js**

import './App.css';

import Posts from './Posts';

function App() {

  return (

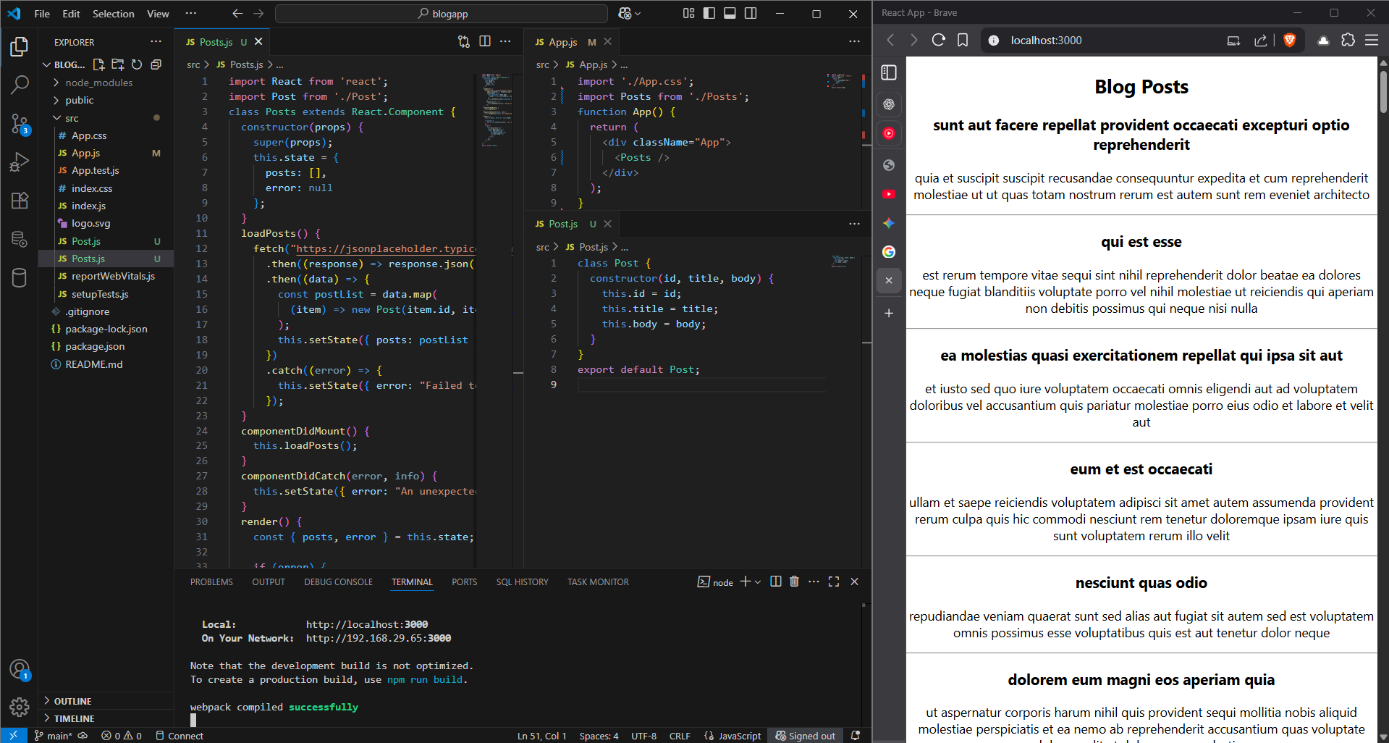
    <div className="App">

      <Posts />

    </div>

  );

}

export default App;

**Output:**

**ReactJS-HOL 5**

**Scenario:**

The "My Academy" team at Cognizant wants to create a dashboard that displays the details of ongoing and completed training cohorts. Your task is to take an existing React application and apply specific styling to the components using both CSS Modules for component-scoped styles and inline styles for dynamic, data-driven styling.

**Procedure:**

**Step 1: Setup the Project**

* + Unzip the provided cohorttracker.zip application in a folder.
  + Open a command prompt and navigate into the cohortstracker folder.
  + Restore the necessary packages by running the command:
  + npm install

**Step 2: Create a CSS Module**

* + Open the project in Visual Studio Code.
  + In the src folder, create a new file named CohortDetails.module.css.
  + Define a class named box with the following properties:
    - width: 300px
    - display: inline-block
    - margin: 10px
    - padding: 10px 20px
    - border: 1px solid black
    - border-radius: 10px
  + Style the <dt> element to have a font-weight of 500.

**Step 3: Apply Styles to the Component**

* + Open the CohortDetails.js file.
  + Import the CohortDetails.module.css file as a styles object.
  + Apply the box class from the CSS module to the main div element.
  + Implement an inline style for the <h3> element that sets the font color to "green" if the cohort status is "Ongoing" and "blue" otherwise.

**Step 4: Run the Application**

* + In your terminal, start the development server:
  + npm start
  + Open your browser to see the styled cohort details cards.

**Implementation:**

**CohortDetails.module.css**

.box {

  width: 300px;

  display: inline-block;

  margin: 10px;

  padding: 10px 20px;

  border: 1px solid black;

  border-radius: 10px;

  background-color: #f9f9f9;

}

dt {

  font-weight: 500;

}

.ongoing {

  color: green;

}

.completed {

  color: blue;

}

**CohortDetails.js**

import React from 'react';

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

function CohortDetails({ cohort }) {

  const titleStyle = {

    color: cohort.status === 'Ongoing' ? 'green' : 'blue'

  };

  return (

    <div className={styles.box}>

      <h3 style={titleStyle}>{cohort.name}</h3>

      <dl>

        <dt>Status:</dt>

        <dd>{cohort.status}</dd>

        <dt>Start Date:</dt>

        <dd>{cohort.startDate}</dd>

        <dt>End Date:</dt>

        <dd>{cohort.endDate}</dd>

      </dl>

    </div>

  );

}

export default CohortDetails;

**App.js**

import React from 'react';

import CohortDetails from './CohortDetails';

function App() {

  const cohorts = [

    { name: 'React Bootcamp', status: 'Ongoing', startDate: '2025-08-01', endDate: '2025-09-22' },

    { name: 'Angular Basics', status: 'Completed', startDate: '2025-06-03', endDate: '2025-07-13' }

  ];

  return (

    <div>

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

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

      ))}

    </div>

  );

}

export default App;

**Output:**

A screenshot of a computer

AI-generated content may be incorrect.