**Course Code: CSE3150**

**Course Title: Front End Full Stack Development**

**Lab sheet 14 - Module 3**

**Lab Program 1**: React Class Initialization

* Create a new React component using class syntax, called MyComponent.
* Initialize the state of MyComponent with a single property called count, set to 0.
* Define a render method for MyComponent that returns a div element containing a p element displaying the current value of count.
* Add a button element to MyComponent that, when clicked, increments the value of count by 1.
* Test MyComponent by rendering it in a parent component, App.

**Answer:**

import React, { Component } from 'react';

class MyComponent extends Component {

constructor(props) {

super(props);

this.state = { count: 0 };

}

handleButtonClick = () => {

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

};

render() {

return (

<div>

<p>Count: {this.state.count}</p>

<button onClick={this.handleButtonClick}>Increment Count</button>

</div>

);

}

}

class App extends Component {

render() {

return <MyComponent />;

}

}

export default App;

**Lab Program 2**: Component Life Cycles

* Create a new React component called LifecycleComponent.
* Define a constructor for LifecycleComponent that sets an initial state property called count to 0.
* Define a componentDidMount method for LifecycleComponent that logs a message to the console when the component mounts.
* Define a componentDidUpdate method for LifecycleComponent that logs a message to the console when the component updates.
* Define a componentWillUnmount method for LifecycleComponent that logs a message to the console when the component unmounts.
* Render the LifecycleComponent in a parent component, App.

**Answer:**

import React, { Component } from 'react';

class LifecycleComponent extends Component {

constructor(props) {

super(props);

this.state = { count: 0 };

}

componentDidMount() {

console.log('LifecycleComponent mounted');

}

componentDidUpdate() {

console.log('LifecycleComponent updated');

}

componentWillUnmount() {

console.log('LifecycleComponent will unmount');

}

handleButtonClick = () => {

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

};

render() {

return (

<div>

<p>Count: {this.state.count}</p>

<button onClick={this.handleButtonClick}>Increment Count</button>

</div>

);

}

}

class App extends Component {

render() {

return <LifecycleComponent />;

}

}

export default App;

Lab Program 3: Component Mounting

* Create a new React component called MountingComponent.
* Define a constructor for MountingComponent that sets an initial state property called count to 0.
* Define a componentWillMount method for MountingComponent that logs a message to the console when the component will mount.
* Define a componentDidMount method for MountingComponent that logs a message to the console when the component mounts.
* Render the MountingComponent in a parent component, App.

**Answer:**

import React, { Component } from 'react';

class MountingComponent extends Component {

constructor(props) {

super(props);

console.log('MountingComponent constructor');

this.state = { count: 0 };

}

componentWillMount() {

console.log('MountingComponent will mount');

}

componentDidMount() {

console.log('MountingComponent mounted');

}

handleButtonClick = () => {

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

};

render() {

return (

<div>

<p>Count: {this.state.count}</p>

<button onClick={this.handleButtonClick}>IncrementCount</button>

</div>

);

}

}

class App extends Component {

render() {

return <MountingComponent />;

}

}

export default App;

Lab Program 4: Node.js and NPM

* Install Node.js and NPM on your computer.
* Create a new project folder for a React app.
* Use NPM to install the React and ReactDOM packages.
* Create a new index.js file in the project folder.
* In index.js, import React and ReactDOM.
* Create a new React component using function syntax, called MyComponent.
* Define a render method for MyComponent that returns a div element containing a p element displaying a message.
* Use ReactDOM to render MyComponent in the DOM.

**Answer:**

To install Node.js and NPM, go to https://nodejs.org and follow the installation instructions for your operating system.

Create a new project folder called my-react-app.

Open a command prompt in the my-react-app folder and run the command:

npm install react react-dom

Create a new file called index.js in the my-react-app folder.

In index.js, add the following code:

import React from 'react';

import ReactDOM from 'react-dom';

Add the following code to define the MyComponent function:

function MyComponent() {

return (

<div>

<p>Hello, world!</p>

</div>

);

}

Add the following code to render MyComponent in the DOM:

ReactDOM.render(<MyComponent />, document.getElementById('root'));

Create an HTML file called index.html in the my-react-app folder, with the following contents:

<!DOCTYPE html>

<html>

<head>

<title>My React App</title>

</head>

<body>

<div id="root"></div>

<script src="index.js"></script>

</body>

</html>

Open index.html in a web browser to see the rendered MyComponent.

Lab Program 5: JSX Walkthrough

* Create a new project folder for a React app.
* Use NPM to install the React and ReactDOM packages.
* Create a new index.js file in the project folder.
* In index.js, import React and ReactDOM.
* Create a new React component using class syntax, called MyComponent.
* Define a render method for MyComponent that returns a div element containing a h1 element with a title and a p element with some text.
* Use JSX syntax to define the div and its contents.
* Use ReactDOM to render MyComponent in the DOM.

**Answer:**

Create a new project folder called my-react-app.

Open a command prompt in the my-react-app folder and run the command:

npm install react react-dom

Create a new file called index.js in the my-react-app folder.

In index.js, add the following code:

import React from 'react';

import ReactDOM from 'react-dom';

Add the following code to define the MyComponent class:

class MyComponent extends React.Component {

render() {

return (

<div>

<h1>Hello, world!</h1>

<p>This is a React app.</p>

</div>

);

}

}

Add the following code to render MyComponent in the DOM:

ReactDOM.render(<MyComponent />, document.getElementById('root'));

JSX syntax allows you to use HTML-like syntax to define React elements. In this case, the MyComponent's render method returns a div element with an h1 element and a p element. Here is the code:

return (

<div>

<h1>Hello, world!</h1>

<p>This is a React app.</p>

</div>

);

Create an HTML file called index.html in the my-react-app folder, with the following contents:

<!DOCTYPE html>

<html>

<head>

<title>My React App</title>

</head>

<body>

<div id="root"></div>

<script src="index.js"></script>

</body>

</html>

Open index.html in a web browser to see the rendered MyComponent.