LAB MANUAL FOR

**WEB DESIGN AND DEVELOPMENT**

REGULATIONS- R20

by

Dr.PamidiSrinivasulu, M.Tech, Ph.D

Mr.Subhabrata Sinha, M.SC(M.Tech)

**Institution Vision:**

To produce global competent, ethical and dynamic professionals by creating Centre of Excellence in Technical Education for societal empowerment.

**Institution Mission:**

**M1:** To provide quality education with knowledge and skills for rural and urban students.

**M2:** To collaborate the industries with academia for empowering the students to meet global standards.

**M3:** To induce highly ethical entrepreneurship in young minds with good leadership quality for the society.

**M4:** To enhance the institution in Research and Development by human intellectual capability

**Department Vision:**

To empower Computer Science Engineers as highly proficient, innovative, self-driven, and socially responsible professionals by offering a multidimensional education.

**Department Mission:**

**M1**: **Empowerment of Computer Science Engineers**: To empower Computer

Science Engineers through comprehensive education, fostering high proficiency.

**M2: Multidimensional Education**: To provide a diverse education, equipping engineers with skills to navigate the global needs.

**M3:Development of Proficiency and Innovation**: To ensure graduates are wellprepared to address technical issues with creativity and innovation.

**M4: Social Responsibility and Professional Excellence**: To instill a strong sense of commitment, shaping graduates into socially responsible professionals and self- driven.

**Preface**

Welcome to the WEB DESIGN AMD DEVELOPMENT Lab Manual for students pursuing a Bachelor of Technology (B.Tech) degree in Computer Science and Engineering. This manual is designed to accompany the practical aspects of the WEB DESIGN AND DEVELOPMENT course, providing students with hands-on experience in applying WDD algorithms to real-world problems.

WDD has emerged as a language with its wide-ranging applications in areas such as healthcare, finance, marketing, and more, understanding WDD concepts and techniques is essential for aspiring computer science professionals.

This manual serves as a comprehensive guide for conducting experiments, coding exercises, and projects related to various WDD algorithms and techniques. It is structured to align with the curriculum of the WDD course, covering fundamental concepts as well as advanced topics.

**Key features of this manual include:**

**Experiment Instructions:** Step-by-step instructions for conducting experiments and implementing WDD algorithms from scratch.

**Code Samples:** WDD code samples to facilitate understanding and implementation of WDD concepts.

**Datasets:** Access to datasets for conducting experiments and building WDD models.

**Learning Objectives:** Clear learning objectives outlined for each experiment, helping students understand the purpose and expected outcomes.

**Assessment Guidelines:** Criteria for evaluating students' performance and understanding of WDD concepts.

**Additional Resources:** References to supplementary resources, including textbooks, online tutorials, and research papers, to further enhance learning.

We encourage students to actively engage with the material presented in this manual, to explore, experiment, and apply WDD algorithms in practical scenarios. By doing so, students will develop critical problem-solving skills, gain proficiency in using WDD tools and libraries, and prepare themselves for careers in WDD, and related fields.

We would like to express our gratitude to all the contributors, reviewers, and educators who have contributed to the development of this manual. We hope that it will serve as a valuable resource in your journey to mastering WDD concepts and techniques.

Best wishes for your learning journey!

Dr.PamidiSrinivasulu, M. Tech, Ph. D

Professor & HOD

Swarnandhra College of Engineering and Technology

**Title: Web Design and Development Lab Manual**

**Objective:**

The WDD Lab Manual for B. Tech Computer Science and Engineering students in their 3rd year, VIth semester aims to provide a structured framework for practical learning and skill development in the field of WDD. The primary objectives of this lab manual are as follows:

**Hands-on Experience:** Offer students practical exposure to various Python WDD algorithms, techniques, and tools through hands-on lab exercises. This practical experience is crucial for reinforcing theoretical concepts learned in lectures and textbooks.

**Skill Development:** Focus on developing essential skills required for applying WDD algorithms to real-world problems. This includes core concepts and advance concepts of WDD.

**Understanding Algorithms:** Help students gain a deeper understanding of WDD algorithms by implementing them from scratch. This approach enhances students' comprehension of algorithmic principles and fosters critical thinking and problem-solving skills.

**Application of Concepts:** Encourage students to apply WDD concepts and techniques to solve diverse problems across different domains. By working on practical projects and case studies, students learn to translate theoretical knowledge into actionable insights and solutions.

**Teamwork and Collaboration:** Promote teamwork and collaboration by assigning group projects and lab activities. Collaboration fosters creativity, communication, and peer learning, which are essential skills in the professional world.

**Project-based Learning:** Emphasize project-based learning to provide students with an opportunity to work on real-world WDD projects. Projects enable students to explore their interests, apply their skills in a practical setting, and showcase their accomplishments.

**Ethical Awareness:** Raise awareness about the ethical considerations and implications of WDD technologies. Students learn to recognize and address ethical challenges related to data privacy, bias, fairness, and transparency in their projects and applications.

**Continuous Assessment and Feedback:** Implement continuous assessment mechanisms to evaluate students' progress and understanding throughout the lab sessions. Regular feedback sessions provide students with constructive feedback and opportunities for improvement.

**Preparation for Industry Roles:** Equip students with the practical skills and experience required to pursue careers in WDD, and related fields.

The lab manual prepares students to tackle real-world challenges and succeed in industry roles upon graduation.

**Lifelong Learning:**Instill a mindset of lifelong learning and curiosity about emerging trends and advancements in WDD. The lab manual encourages students to stay updated with the latest developments in the field and continue their learning journey beyond the classroom.

By achieving these objectives, the WDD Lab Manual aims to empower B.Tech Computer Science and Engineering students with practical skills, knowledge, and confidence to excel in the field of WDD and make meaningful contributions to society and industry.

**Equipment:**

**Computers:** Each student should have access to a computer or laptop with sufficient processing power and memory to run WDD algorithms and perform development.

**Internet Connectivity:** Reliable internet connectivity is essential for installing WDD software, accessing online resources for WDD experiments.

**Projection System:** A projector or display screen is necessary for instructors to present lecture materials, code demonstrations, and visualizations during lab sessions.

**Software:**

**node-v18.14.0-x64:**ReactJs is the primary programming language used in the field of web design and development due to its extensive libraries and frameworks. Ensure that this software is installed on all computers.

**VSCodeUserSetup-x64-1.84.2:**Visual Studio Code Editor is used as a IDE for ReactJs. Ensure that this software is installed on all computers.

**eclipse-jee-kepler-SR2-win32-x86\_64:**Eclipse Kepler is used as a IDE for developing web applications using servlet and jsp. Ensure that this software is installed on all computers.

**apache-tomcat-7.0.79:**Apache Tomcat is used as a Web Server for Deploying Web Applications developed using servlet and jsp. With Eclipse Kepler IDE we can add this server.

**Database Management Systems (DBMS):** Depending on the lab activities, students may need access to relational databases (e.g., Oracle 11g) for data storage and retrieval.

Ensure that all software tools and libraries are properly installed and configured on lab computers before the start of the course. Provide clear instructions and resources for students to install necessary software on their personal devices if required. Additionally, regularly update software and libraries to ensure compatibility with the latest versions and to access new features and improvements.

**Pre-lab Preparation for Web Design and Development Lab Course**

Before attending the WDD Lab sessions, students should complete the following pre-lab preparation to ensure they are ready to engage effectively with the course material:

**Familiarize Yourself with HTML, CSS, and Javascript:**

Ensure that you have a basic understanding of the HTML, CSS and Javascript programming languages, including different types of Tags, Attributes, Stylesheet, events, variables, data types, loops, functions, and control structures.

If you're new to HTML, CSS and Javascript, consider completing online tutorials or courses to get up to speed before the lab sessions begin.

**Install Necessary Software:**

Install NodeJs, Visual Studio Code Editor, Eclipse Kepler IDE, Apache Tomcat Web Server and Oracle 11g.

**Review Basic Programming Concepts:**

Review fundamental concepts of programming such as HTML Tags, Attributes, Stylesheet, events, variable, constant, data type, conditional construct, iterative construct, loop control statements, function, class, object, etc.

**Refresh Object Oriented Programming Knowledge:**

Brush up on basic Object Orientation. Features of object orientation. How to write class and objects.

**Complete Pre-lab Exercises:**

Review any pre-lab exercises or assignments provided by the instructor.

Practice coding exercises related to HTML, CSS, Javascript programming.

**Set Up Development Environment:**

Ensure that your development environment is set up and ready for the lab sessions. This includes having access to a reliable internet connection, a comfortable workspace, and necessary software tools.

**Review Lab Schedule and Materials:**

Review the lab schedule, objectives, and any materials provided by the instructor, such as lecture slides, lab manuals, or coding examples.

Familiarize yourself with the topics and activities planned for each lab session to maximize your learning experience.

By completing these pre-lab preparations, students can arrive at the WDD Lab sessions with a solid foundation in HTML, CSS, Javascript programming, OOPS concepts, and necessary software tools, enabling them to actively participate and engage in the learning process from the outset.

**Lab Record/Report Must Contain the following items:**

**Year of Study: 2024 Semester:VI Academic Year: 2024**

**Course Title:** Web Design and Development Lab

**Lab Session Date:** [Date] **Lab Session Topic:** [Topic]

**Lab Instructor:** [Instructor's Name]

Name of the Student: Roll No:

**Objective:**

**Tasks Completed:**

[Task 1 Description]

* + [Details of task completion]
  + [Any challenges faced and how they were resolved]

**Results and Observations:** [Provide any results, observations, or findings from the lab tasks.]

**Discussion:** [Discuss the significance of the lab tasks, any insights gained, and how they relate to the course objectives. Address any questions or areas of confusion that arose during the lab session.]

**Conclusions:** [Summarize the key findings and conclusions drawn from the lab session. Discuss any implications for future work or additional experimentation.]

**Lab Instructor's Remarks:** [Provide any feedback, suggestions, or comments from the lab instructor regarding students' performance, participation, or the overall lab session.]

**Lab Instructor's Signature with Date:**

**Lab Evaluation :**

**Internal marks : 30 and External marks: 70**

**CONTENTS OF THE EXPERIMENTS**

|  |  |  |
| --- | --- | --- |
| E.no | Title of the experiment | page-no |
| 1. | Create a React Application Called hello-world and display the message “Hello World” within the browser. Write down all the steps to perform the above operation. |  |
| 2. | Create a React Application hello-world. Implement a functional component Greet.js and import it in App.js as a <Greet/> tag which will display a Greetings in the Browser. |  |
| 3. | Create a React Application hello-world. Implement a class component Welcome.js and import it in App.js as a <Welcome> tag which will display a Welcome message in the Browser. |  |
| 4. | Create a React Application hello-world. Implement a functional component Hello.js. Within Hello.js return “Hello Vishwas” in <h1> text by using JSX and without using JSX. import it in App.js as a <Hello> tag which will display “Hello Vishwas” in the Browser. |  |
| 5. | Create a React Application hello-world. Implement a functional component Greet.js and import it in App.js as a <Greet name=”Bruce”/> tag which will display a “Welcome Bruce” in the Browser.(use {props.name} to return the value of name attribute in the Greet.js |  |
| 6. | Create a React Application hello-world. Implement a class component Message.js. Under Message.js write a Constructor within Message class within the constructor assign a message to the state.hint(this.state={message:”Welcome Visitor”}).Within the render()method of the class return html code as (return <h1>{this.state.message}</h1> after displaying “Welcome Visitor” display a button with the text “subscribe”.Whenever the user clicks on the button the message changes to “Thank you for Subscribing”.hint(use this.setState({message:”Thank You For Subscribing”}) within the message handler). |  |
| 7. | Create a React Application hello-world. Implement a functional component FunctionClick.js and import it in App.js as a <FunctionClick/> tag which will display a button .Clicking on the button log a message to the console as “ButtonClicked”.(use event handling concept) |  |
| 8. | Create a React Application hello-world. Implement a class component Form.js.Within Form.js create a controlled component textfield.(use event handelling ,state and setState method). |  |
| 9. | Create a React Application react-http. Stay inside the react-http directory and execute the command “npm install axios”. Implement a class component PostList.js. Within PostList class inside the constructor set the state to posts array.Now write componentDidMount() method inside the class. Within componentDidMount() method write the following code  “axios.get(“<https://jsonplacehoder.typicode.com/posts>”) “.Write the code to store all the data within the posts array. Now using the map method display all the posts title inside the render() method. |  |
| 10. | Create a React Application react-http. Stay inside the react-http directory and execute the command “npm install axios”.Implement a class component PostForm.js. Under the PostForm class under the render method create a form with the input(text) field userid, title, body. Set the state for userid, title, body. Now store the userid, title and body from the state to the value property of each text field.Now on the onChange event of each text field called the changeHandler()method to store the userid,title,body in the state from the value property of the text fields.Now add a submit button.On the onSubmit event of the form call a submitHandler() method generate post request using axios and send the state as a data.Then display userid, title, and body retrieved from the server to the console |  |

1.

AIM :Create a React Application Called hello-world and display the message “Hello World” within the browser. Write down all the steps to perform the above operation.

DESCRIPTION :

React is a popular JavaScript library used for building web applications. It was developed by Facebook and has become a widely adopted technology for front-end development due to its simplicity, efficiency, and flexibility.

One of the key features of React is its virtual DOM, which is an in-memory representation of the actual DOM that allows React to efficiently update the UI when the state of the application changes. This means that instead of re-rendering the entire page when a change occurs, React only updates the specific parts of the UI that need to be changed.

PROGRAM

Step 1: Install Node.js

* If you don't have Node.js installed on your computer, go to <https://nodejs.org/en/> and download and install the latest version for your operating system.

Step 2: Install Create React App

* Open a terminal or command prompt and run the following command to install Create React App: (lua)

npm install -g create-react-app

Step 3: Create a new React application

* Run the following command to create a new React application called hello-world: (lua)

npx create-react-app hello-world

Step 4: Open the project directory

* Navigate to the hello-world directory by running the following command: (bash)

cd hello-world

Step 5: Update the App.js file

* Open the App.js file located in the src directory using your favorite text editor
* Replace the content with the following code:(javascript)

import React from 'react';

import './App.css';

function App() {

return (

<div className="App">

<h1>Hello World</h1>

</div>

);

}

export default App;

Step 6: Start the development server

* Run the following command to start the dev00elopment server:(sql)

npm start

Step 7: View the app in the browser

* Open your web browser and go to [http://localhost:3000](http://localhost:3000/)
* You should see the message "Hello World" displayed within the browser.

OUTPUT :

Localhost:3000 says

Hello world

2.

AIM :

Create a React Application hello-world. Implement a functional component Greet.js and import it in App.js as a <Greet/> tag which will display a Greetings in the Browser.

DESCRIPTION :

In React, a functional component is a JavaScript function that returns a React element. It is one of the two primary ways of defining components in React, the other being class components.

Functional components are simpler to write and understand than class components because they are just plain functions that take in props as input and return JSX as output. They are also more lightweight and faster to render than class components because they don't have any additional overhead of maintaining state or lifecycle methods.

PROGRAM :

React is a popular JavaScript library used for building web applications. It was developed by Facebook and has become a widely adopted technology for front-end development due to its simplicity, efficiency, and flexibility.

Step 1: Install Node.js

* If you don't have Node.js installed on your computer, go to <https://nodejs.org/en/> and download and install the latest version for your operating system.

Step 2: Install Create React App

* Open a terminal or command prompt and run the following command to install Create React App: (lua)

npm install -g create-react-app

Step 3: Create a new React application

* Run the following command to create a new React application called hello-world: (lua)

npx create-react-app hello-world

Step 4: Open the project directory

* Navigate to the hello-world directory by running the following command: (bash)

cd hello-world

Step 5: Create Greet.js component

* Create a new file named Greet.js in the src directory using your favorite text editor
* Add the following code to the Greet.js file: (jsx)

import React from 'react';

function Greet() {

return (

<div>

<h1>Greetings!</h1>

<p>Welcome to my React App</p>

</div>

);

}

export default Greet;

Step 6: Update the App.js file

* Open the App.js file located in the src directory using your favorite text editor
* Replace the content with the following code: (jsx)

import React from 'react';

import Greet from './Greet';

function App() {

return (

<div>

<Greet />

</div>

);

}

export default App;

Step 7: Start the development server

* Run the following command to start the development server: (sql)

npm start

Step 8: View the app in the browser

* Open your web browser and go to [http://localhost:3000](http://localhost:3000/)
* You should see the message "Greetings! Welcome to my React App" displayed within the browser.

OUTPUT :

Greetings! Welcome to my React App

**3.**

AIM :

Create a React Application hello-world. Implement a class component Welcome.js and import it in App.js as a <Welcome> tag which will display a Welcome message in the Browser.

DESCRIPTION:

In React, a functional component is a JavaScript function that returns a React element. It is one of the two primary ways of defining components in React, the other being class components.

A class component is a type of component used in object-oriented programming (OOP) languages like JavaScript and Python. It is a blueprint or template for creating objects with specific properties and behaviors.

Class components have access to lifecycle methods such as componentDidMount(), componentDidUpdate(), and componentWillUnmount(). These methods allow developers to perform actions at specific points in the component's lifecycle, such as when it is mounted or updated.

PROGRAM :

1. First, you need to make sure that you have Node.js and npm installed on your system. You can check this by running the following commands in your terminal:

node -v

npm -v

If both of these commands return a version number, then you're good to go. Otherwise, you need to download and install Node.js and npm from the official website.

1. Once you have Node.js and npm installed, you can create a new React application using the create-react-app command. Open your terminal and run the following command: (lua)

npx create-react-app hello-world

This will create a new React application called "hello-world" in a new directory with the same name.

1. Next, you need to create a new class component called "Welcome". In the "src" directory of your React application, create a new file called "Welcome.js" and add the following code: (javascript)

import React, { Component } from 'react';

class Welcome extends Component {

render() {

return<h1>Welcome to React!</h1>;

}

}

export default Welcome;

This code imports the React library and the Component class from it. It then defines a new class component called "Welcome" that extends the Component class. The render() method of the Welcome component returns an <h1> element with a welcome message.

1. Finally, you need to import the Welcome component into the App.js file and use it in your React application. Open the "App.js" file in the "src" directory and replace its contents with the following code: (javascript)

import React, { Component } from 'react';

import Welcome from './Welcome';

class App extends Component {

render() {

return (

<div className="App">

<Welcome />

</div>

);

}

}

export default App;

This code imports the React library and the Welcome component from the "Welcome.js" file. It then defines a new class component called "App" that extends the Component class. The render() method of the App component returns a <div> element that contains the <Welcome> tag. When the React application is run, the <Welcome> tag will be replaced with the welcome message defined in the Welcome component.

1. Finally, you can start the React application by running the following command in your terminal: (sql)

npm start

This will start a development server and open your React application in your default browser. You should see a welcome message that says "Welcome to React!" displayed in the browser.

OUTPUT :

Welcome to React!

4.

AIM :

Create a React Application hello-world. Implement a functional component Hello.js. Within Hello.js return “Hello Vishwas” in <h1> text by using JSX and without using JSX. import it in App.js as a <Hello> tag which will display “Hello Vishwas” in the Browser.

DESCRIPTION :

React is a popular JavaScript library used for building web applications. It was developed by Facebook and has become a widely adopted technology for front-end development due to its simplicity, efficiency, and flexibility.

Functional components are simpler to write and understand than class components because they are just plain functions that take in props as input and return JSX as output. They are also more lightweight and faster to render than class components because they don't have any additional overhead of maintaining state or lifecycle methods.

JSX (JavaScript XML) is a syntax extension for JavaScript that allows developers to write HTML-like code in their JavaScript code. It is often used in the context of React, a popular JavaScript library for building user interfaces.

PROGRAM :

Step 1: Create a new React application named **hello-world** using the **create-react-app** command.(bash)

npx create-react-app hello-world

Step 2: Navigate to the **src** directory and create a new file called **Hello.js**.(bash)

cd hello-world/src

touch Hello.js

Step 3: Open **Hello.js** in your text editor and add the following code: (jsx)

import React from 'react';

function Hello() {

// Using JSX

return<h1>Hello Vishwas</h1>;

// Without using JSX

// return React.createElement('h1', null, 'Hello Vishwas');

}

export default Hello;

Step 4: Open **App.js** in your text editor and update the code to import **Hello** component and render it in your component as **<Hello />**.(jsx)

import React from 'react';

import Hello from './Hello';

function App() {

return (

<div>

<Hello />

</div>

);

}

export default App;

Step 5: Start the development server by running the following command in the terminal: (bash)

npm start

Step 6: Open your browser and go to **http://localhost:3000/** to see "Hello Vishwas" displayed on the screen.

OUTPUT :

Hello Vishwas

5.

AIM :

Create a React Application hello-world. Implement a functional component Greet.js and import it in App.js as a <Greet name=”Bruce”/> tag which will display a “Welcome Bruce” in the Browser.(use {props.name} to return the value of name attribute in the Greet.js

DESCRIPTION :

React is a popular JavaScript library used for building web applications. It was developed by Facebook and has become a widely adopted technology for front-end development due to its simplicity, efficiency, and flexibility.

Functional components are simpler to write and understand than class components because they are just plain functions that take in props as input and return JSX as output. They are also more lightweight and faster to render than class components because they don't have any additional overhead of maintaining state or lifecycle methods.

PROGRAM :

1. Install Node.js and npm on your system if you haven't already.
2. Open a terminal or command prompt and create a new React project using the create-react-app command: (lua)

npx create-react-app hello-world

1. Change directory into the newly created project: (bash)

cd hello-world

1. Open the project in your code editor of choice.
2. Create a new file called Greet.js in the src folder.
3. In the Greet.js file, define the functional component like so: (javascript)

import React from 'react';

function Greet(props) {

return<h1>Welcome {props.name}</h1>;

}

export default Greet;

1. In the App.js file, import the Greet component and use it like so: (javascrcipt)

import React from 'react';

import Greet from './Greet';

function App() {

return (

<div className="App">

<Greet name="Bruce" />

</div>

);

}

export default App;

1. Save the changes to both files and start the development server using the command: (sql)

npm start

You should now see the "Welcome Bruce" message displayed in your browser.

OUTPUT :

Welcome Bruce

6.

AIM :

Create a React Application hello-world. Implement a class component Message.js. Under Message.js write a Constructor within Message class within the constructor assign a message to the state.hint(this.state={message:”Welcome Visitor”}).

Within the render()method of the class return html code as (return <h1>{this.state.message}</h1> after displaying “Welcome Visitor” display a button with the text “subscribe”.Whenever the user clicks on the button the message changes to “Thank you for Subscribing”.hint(use this.setState({message:”Thank You For Subscribing”}) within the message handler).

DESCRIPTION :

React is a popular JavaScript library used for building web applications. It was developed by Facebook and has become a widely adopted technology for front-end development due to its simplicity, efficiency, and flexibility.

In the context of React, a class component is a way of defining a component using a class instead of a function. Class components are defined using the ES6 class syntax and extend the React.Component class. They have a render() method that returns the HTML markup that the component will render.

PROGRAM :

First, create a new React application by running **npx create-react-app hello-world** in your terminal/command prompt.

Then, create a new file called **Message.js** inside the **src** folder with the following code: (javascript)

import React, { Component } from 'react';

class Message extends Component {

constructor(props) {

super(props);

this.state = {

message: "Welcome Visitor"

};

this.handleSubscribeClick = this.handleSubscribeClick.bind(this);

}

handleSubscribeClick() {

this.setState({ message: "Thank You For Subscribing" });

}

render() {

return (

<div>

<h1>{this.state.message}</h1>

<button onClick={this.handleSubscribeClick}>Subscribe</button>

</div>

);

}

}

export default Message;

In the above code, we define a class component **Message** with a constructor that sets the initial state of **message** to "Welcome Visitor". We also define a method **handleSubscribeClick** that updates the state with the message "Thank You For Subscribing" whenever the user clicks on the **Subscribe** button.

In the **render** method, we return a **<div>** element containing an **<h1>** element displaying the current message state and a **<button>** element that triggers the **handleSubscribeClick** method when clicked.

Finally, to use the **Message** component in our app, we can modify the **App.js** file to look like this:(javascript)

import React from 'react';

import Message from './Message';

function App() {

return (

<div className="App">

<Message />

</div>

);

}

export default App;

In the above code, we import the **Message** component from the **Message.js** file and render it inside the **App** component. When we run the application using **npm start**, we should see a message that says "Welcome Visitor" and a button that says "Subscribe". When we click on the button, the message should change to "Thank You For Subscribing".

OUTPUT :

Welcome Visitor

Subscribe

Thank You For Subscribing

7.

AIM :

Create a React Application hello-world. Implement a functional component FunctionClick.js and import it in App.js as a <FunctionClick/> tag which will display a button .Clicking on the button log a message to the console as “ButtonClicked”.(use event handling concept)

DESCRIPTION :

React is a popular JavaScript library used for building web applications. It was developed by Facebook and has become a widely adopted technology for front-end development due to its simplicity, efficiency, and flexibility.

Functional components are simpler to write and understand than class components because they are just plain functions that take in props as input and return JSX as output. They are also more lightweight and faster to render than class components because they don't have any additional overhead of maintaining state or lifecycle methods.

PROGRAM :

1. First, make sure you have Node.js installed on your computer.
2. Open your terminal/command prompt and navigate to the folder where you want to create your React application.
3. Run the following command to create a new React application: (lua)

npx create-react-app hello-world

1. Once the application is created, navigate into the project folder: (bash)

cd hello-world

1. Now, open the project in your favorite code editor. I'll assume you're using VSCode.
2. Open the App.js file from the src folder and replace its contents with the following code:(javascript)

import React from 'react';

importFunctionClick from './FunctionClick';

function App() {

return (

<div className="App">

<FunctionClick />

</div>

);

}

export default App;

1. Now, create a new file called FunctionClick.js inside the src folder and add the following code to it: (javascript)

import React from 'react';

functionFunctionClick() {

consthandleClick = () => {

console.log('ButtonClicked');

}

return (

<div>

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

</div>

)

}

export default FunctionClick;

1. Save the files and start the development server by running the following command in the terminal: (sql)

npm start

1. Now, open your browser and navigate to [http://localhost:3000](http://localhost:3000/) to see your React application in action.
2. Click the button on the page and check the console in your browser's developer tools. You should see the message "ButtonClicked" logged to the console.

That's it! You have successfully created a React application called "hello-world" and implemented a functional component FunctionClick.js that logs a message to the console when a button is clicked.

OUTPUT :

ButtonClicked

8.

AIM :

Create a React Application hello-world. Implement a class component Form.js.Within Form.js create a controlled component textfield.(use event handelling ,state and setState method).

DESCRIPTION :

React is a popular JavaScript library used for building web applications. It was developed by Facebook and has become a widely adopted technology for front-end development due to its simplicity, efficiency, and flexibility.

In the context of React, a class component is a way of defining a component using a class instead of a function. Class components are defined using the ES6 class syntax and extend the React.Component class. They have a render() method that returns the HTML markup that the component will render.

PROGRAM :

1.Create a new React application by running the following command in your terminal: (lua)

npx create-react-app hello-world

2.Once the installation is complete, navigate to the project directory and open it in your code editor. (bash)

cd hello-world

code .

3.Create a new file named Form.js in the src folder and add the following code: (jsx)

import React, { Component } from "react";

class Form extends Component {

state = {

name: "",

};

handleChange = (event) => {

this.setState({ name: event.target.value });

};

handleSubmit = (event) => {

event.preventDefault();

alert("Hello " + this.state.name);

};

render() {

return (

<form onSubmit={this.handleSubmit}>

<label>

Name:

<input

type="text"

value={this.state.name}

onChange={this.handleChange}

/>

</label>

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

</form>

);

}

}

export default Form;

4.In the App.js file, import the Form component and render it. (jsx)

import React from "react";

import Form from "./Form";

function App() {

return (

<div className="App">

<Form />

</div>

);

}

export default App;

5.Start the development server by running the following command: (sql)

npm start

6.Open your web browser and navigate to [http://localhost:3000](http://localhost:3000/) to see the hello-world React application with a controlled textfield implemented in Form.js.

OUTPUT :

Hello World

9.

AIM :

Create a React Application react-http. Stay inside the react-http directory and execute the command “npm install axios”. Implement a class component PostList.js. Within PostList class inside the constructor set the state to posts array.Now write componentDidMount() method inside the class. Within componentDidMount() method write the following code

“axios.get(“<https://jsonplacehoder.typicode.com/posts>”) “.Write the code to store all the data within the posts array. Now using the map method display all the posts title inside the render() method.

DESCRIPTION :

React is a popular JavaScript library used for building web applications. It was developed by Facebook and has become a widely adopted technology for front-end development due to its simplicity, efficiency, and flexibility.

In the context of React, a class component is a way of defining a component using a class instead of a function. Class components are defined using the ES6 class syntax and extend the React.Component class. They have a render() method that returns the HTML markup that the component will render.

Class components have access to lifecycle methods such as componentDidMount(), componentDidUpdate(), and componentWillUnmount(). These methods allow developers to perform actions at specific points in the component's lifecycle, such as when it is mounted or updated.

PROGRAM :

Step 1: Create a new React application named "react-http" by executing the following command in your terminal: (Lua)

npx create-react-app react-http

Step 2: Navigate into the "react-http" directory and execute the following command to install the Axios library: (bash)

cd react-http

npm install axios

Step 3: Create a new file named "PostList.js" in the "src" directory and paste the following code: (javascript)

import React, { Component } from 'react';

importaxios from 'axios';

classPostList extends Component {

constructor(props) {

super(props);

this.state = {

posts: []

};

}

componentDidMount() {

axios.get('https://jsonplaceholder.typicode.com/posts')

.then(response => {

this.setState({ posts: response.data });

})

.catch(error => {

console.log(error);

});

}

render() {

const { posts } = this.state;

constpostList = posts.length ? (

posts.map(post => {

return (

<div key={post.id}>

<h2>{post.title}</h2>

</div>

)

})

) : (

<div>No posts yet</div>

);

return (

<div>

<h1>Posts</h1>

{postList}

</div>

)

}

}

export default PostList;

Step 4: In the "App.js" file, import the "PostList" component and add it to the render method: (javascript)

import React from 'react';

importPostList from './PostList';

function App() {

return (

<div className="App">

<PostList />

</div>

);

}

export default App;

Step 5: Start the application by executing the following command: (sql)

npm start

This should start the application on [http://localhost:3000](http://localhost:3000/) and you should see a list of post titles fetched from the JSONPlaceholder API.

10.

AIM :

Create a React Application react-http. Stay inside the react-http directory and execute the command “npm install axios”.Implement a class component PostForm.js. Under the PostForm class under the render method create a form with the input(text) field userid, title, body. Set the state for userid, title, body. Now store the userid, title and body from the state to the value property of each text field.Now on the onChange event of each text field called the changeHandler()method to store the userid,title,body in the state from the value property of the text fields.Now add a submit button.On the onSubmit event of the form call a submitHandler() method generate post request using axios and send the state as a data.Then display userid, title, and body retrieved from the server to the console

DESCRIPTION :

React is a popular JavaScript library used for building web applications. It was developed by Facebook and has become a widely adopted technology for front-end development due to its simplicity, efficiency, and flexibility.

In the context of React, a class component is a way of defining a component using a class instead of a function. Class components are defined using the ES6 class syntax and extend the React.Component class. They have a render() method that returns the HTML markup that the component will render.

Class components have access to lifecycle methods such as componentDidMount(), componentDidUpdate(), and componentWillUnmount(). These methods allow developers to perform actions at specific points in the component's lifecycle, such as when it is mounted or updated.

PROGRAM :

1. Create a new React application by running the following command in your terminal: (lua)

npx create-react-app react-http

1. Navigate to the **react-http** directory by running: (bash)

cd react-http

1. Install the **axios** package by running:

npm install axios

1. Create a new file called **PostForm.js** inside the **src** folder of your React application.
2. Add the following code to **PostForm.js**: (javascript)

import React, { Component } from 'react';

importaxios from 'axios';

classPostForm extends Component {

constructor(props) {

super(props);

this.state = {

userid: '',

title: '',

body: ''

};

}

changeHandler = (event) => {

this.setState({ [event.target.name]: event.target.value });

}

submitHandler = (event) => {

event.preventDefault();

axios.post('https://jsonplaceholder.typicode.com/posts', this.state)

.then(response => {

console.log(response.data);

this.setState({

userid: '',

title: '',

body: ''

});

})

.catch(error => {

console.log(error);

});

}

render() {

const { userid, title, body } = this.state;

return (

<form onSubmit={this.submitHandler}>

<div>

<label htmlFor="userid">User ID</label>

<input

type="text"

id="userid"

name="userid"

value={userid}

onChange={this.changeHandler}

/>

</div>

<div>

<label htmlFor="title">Title</label>

<input

type="text"

id="title"

name="title"

value={title}

onChange={this.changeHandler}

/>

</div>

<div>

<label htmlFor="body">Body</label>

<textarea

id="body"

name="body"

value={body}

onChange={this.changeHandler}

/>

</div>

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

</form>

);

}

}

export default PostForm;

This code creates a class component called **PostForm** that contains a form with three input fields (userid, title, body) and a submit button. The state of the component stores the values entered by the user in these fields. The **changeHandler** method is called whenever the user types something in the fields, and it updates the state accordingly. The **submitHandler** method is called when the user clicks the submit button, and it sends a POST request to the server using the **axios** library. After receiving a response from the server, it logs the retrieved data to the console and clears the state.

1. In the **App.js** file of your React application, add the following code to render the **PostForm** component: (javascript)

import React from 'react';

importPostForm from './PostForm';

function App() {

return (

<div className="App">

<PostForm />

</div>

);

}

export default App;