EXPERIMENT NO:01

INTRODUCTION

ReactJS is a simple, feature-rich, component-based JavaScript UI library. It can be used to develop small applications as well as big, complex applications. ReactJS provides a minimal and solid feature set to kick-start a web application. React community compliments React library by providing a large set of ready-made components to develop web applications in a record time. React community also provides advanced concepts like state management, routing, etc., on top of the React library.

React versions

The initial version, 0.3.0 of React is released in May 2013 and the latest version, 17.0.1 is released in October 2020. The major version introduces breaking changes and the minor version introduces new features without breaking the existing functionality. Bug fixes are released as and when necessary. React follows the Semantic Versioning (semver) principle.

Features

The salient features of React library are as follows −

Solid base architecture

Extensible architecture

Component-based library

JSX-based design architecture

Declarative UI library

Benefits

A few benefits of using React library are as follows −

Easy to learn

Easy to adapt in modern as well as legacy application

A faster way to code a functionality

Availability of a large number of ready-made component

Large and active community

Applications

A few popular websites powered by React library are listed below −

Facebook, the popular social media application

Instagram, a popular photo-sharing application

Netflix, a popular media streaming application

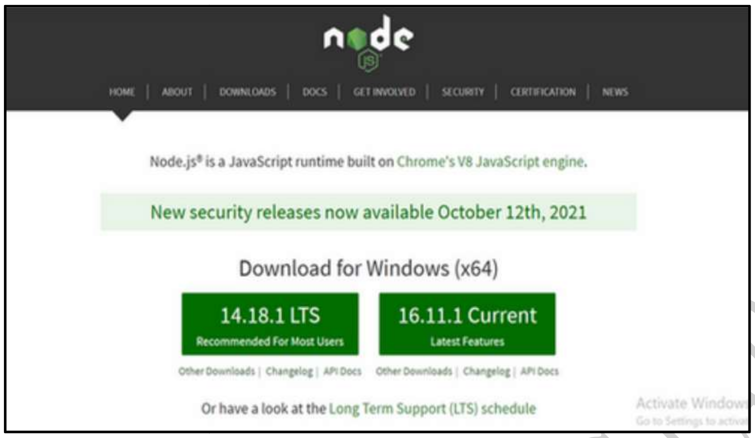
Code Academy, a popular online training application

Reddit, a popular content-sharing application

Date		EXP.NO.		Page No.	

EXPERIMENT NO:02
INSTALLATION OF REACT JS ON WINDOWS

Step 1: Install Node.js installer for windows. Click on this https://nodejs.org/en/. Here install the LTS version (the one present on the left). Once downloaded open NodeJS without disturbing other settings, and click on the Next button until it's completely installed.



Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine.

New security releases now available October 12th, 2021

Download for Windows (x64)

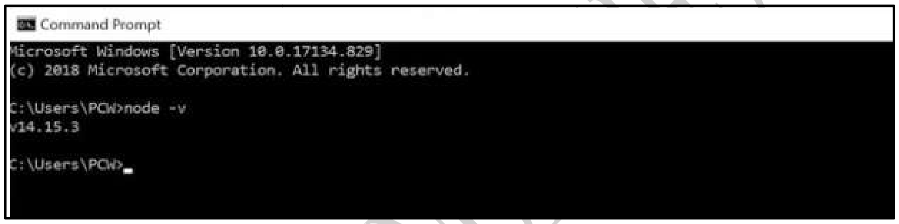
14.18.1 LTS
Recommended For Most Users

16.11.1 Current
Latest Features

Other Downloads | Changelog | API Docs   Other Downloads | Changelog | API Docs

Or have a look at the Long Term Support (LTS) schedule

Step 2: Open the command prompt to check whether it is completely installed or not type the command –> node -v

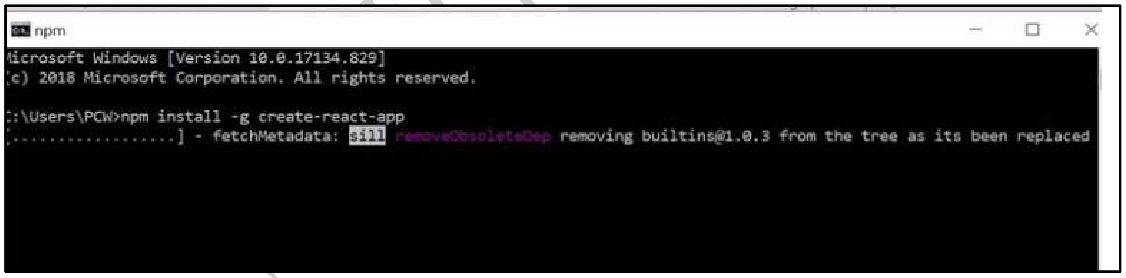


```
Command Prompt
Microsoft Windows [Version 10.0.17134.829]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\PCW>node -v
v14.15.3

C:\Users\PCW>_
```

If the installation went well it will give you the version you have installed

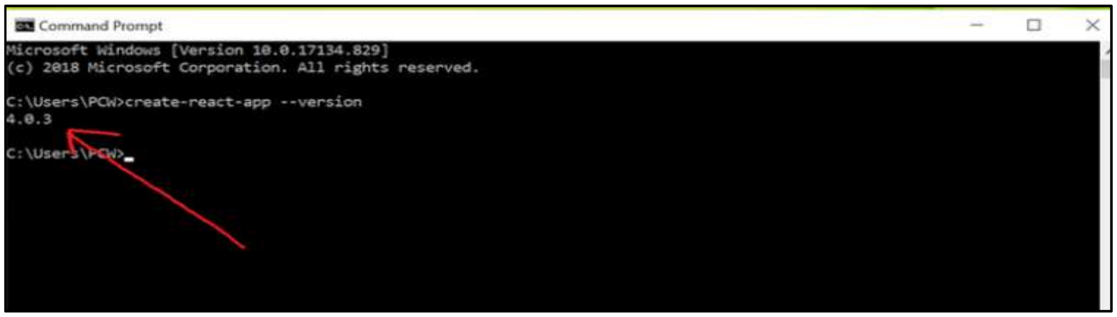
Step 3: Now in the terminal run the below command:

npm install -g create-react-app



```
npm
Microsoft Windows [Version 10.0.17134.829]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\PCW>npm install -g create-react-app
[..................] - fetchMetadata: sill removeObsoleteDep removing builtins@1.0.3 from the tree as its been replaced
```

It will globally install react app for you. To check everything went well run the command create-react-app –version



```
Command Prompt
Microsoft Windows [Version 10.0.17134.829]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\PCW>create-react-app --version
4.0.3

C:\Users\PCW>_
```

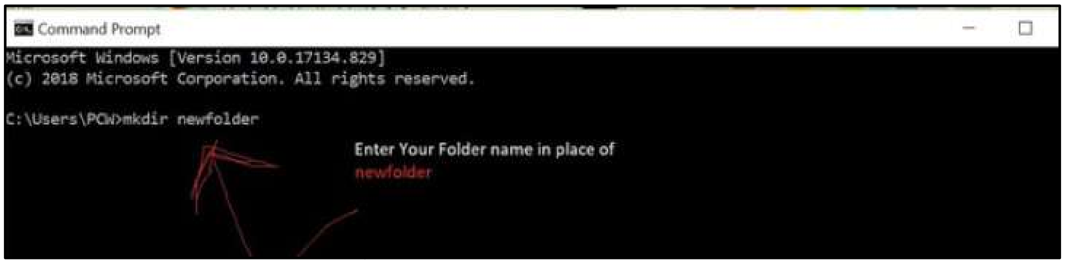
If everything went well it will give you the installed version of react app

Step 4: Now Create a new folder where you want to make your react app using the below command:

mkdirnewfolder

Date		EXP.NO.		Page No.	

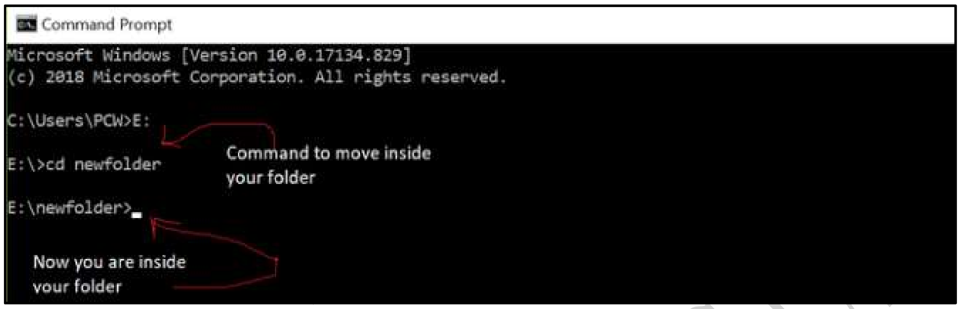
Note: The newfolder in the above command is the name of the folder and can be anything.



```
Command Prompt
Microsoft Windows [Version 10.0.17134.829]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\PCW>mkdir newfolder
```

Move inside the same folder using the below command:

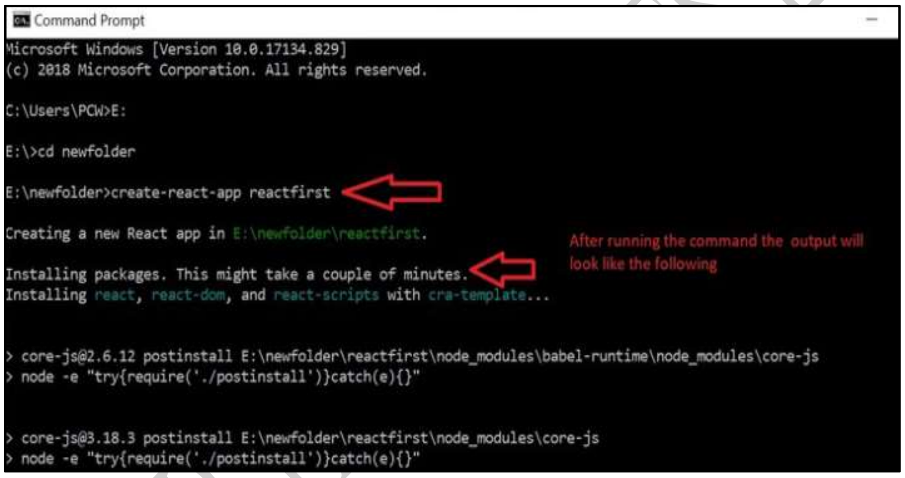
cd newfolder (your folder name)



```
Command Prompt
Microsoft Windows [Version 10.0.17134.829]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\PCW>E:

E:\>cd newfolder

E:\newfolder>_
```

Command to move inside your folder
Now you are inside your folder

Step 5: Now inside this folder run the command –>create-react-app reactfirst YOUR_APP_NAME

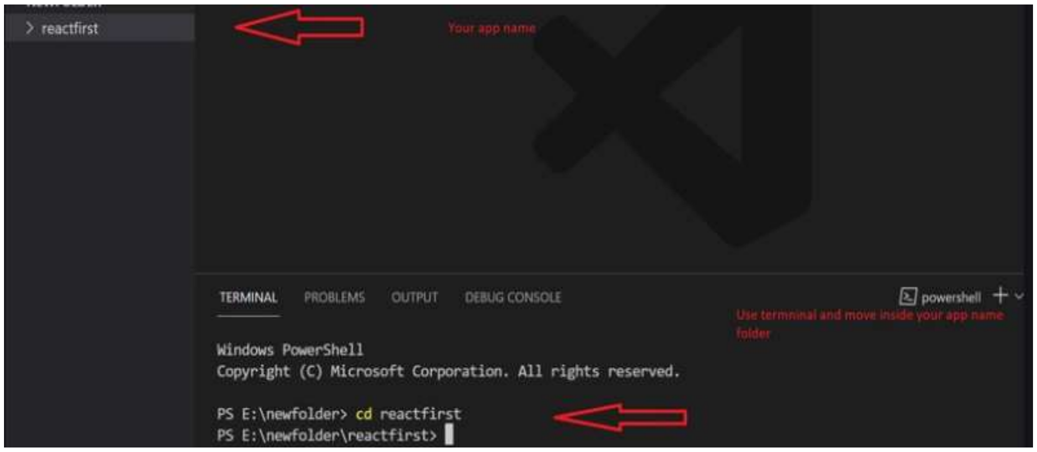


```
Command Prompt
Microsoft Windows [Version 10.0.17134.829]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\PCW>E:

E:\>cd newfolder

E:\newfolder>create-react-app reactfirst

Creating a new React app in E:\newfolder\reactfirst.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...


> core-js@2.6.12 postinstall E:\newfolder\reactfirst\node_modules\babel-runtime\node_modules\core-js
> node -e "try{require('./postinstall')}catch(e){}"


> core-js@3.18.3 postinstall E:\newfolder\reactfirst\node_modules\core-js
> node -e "try{require('./postinstall')}catch(e){}"
```

After running the command the output will look like the following

It will take some time to install the required dependencies

Step 6: Now open the IDE of your choice for eg. Visual studio code and open the folder where you have installed the react app newolder (in the above example) inside the folder you will see your app's name reactapp (In our example). Use the terminal and move inside your app name folder. Use command cdreactapp (your app name)

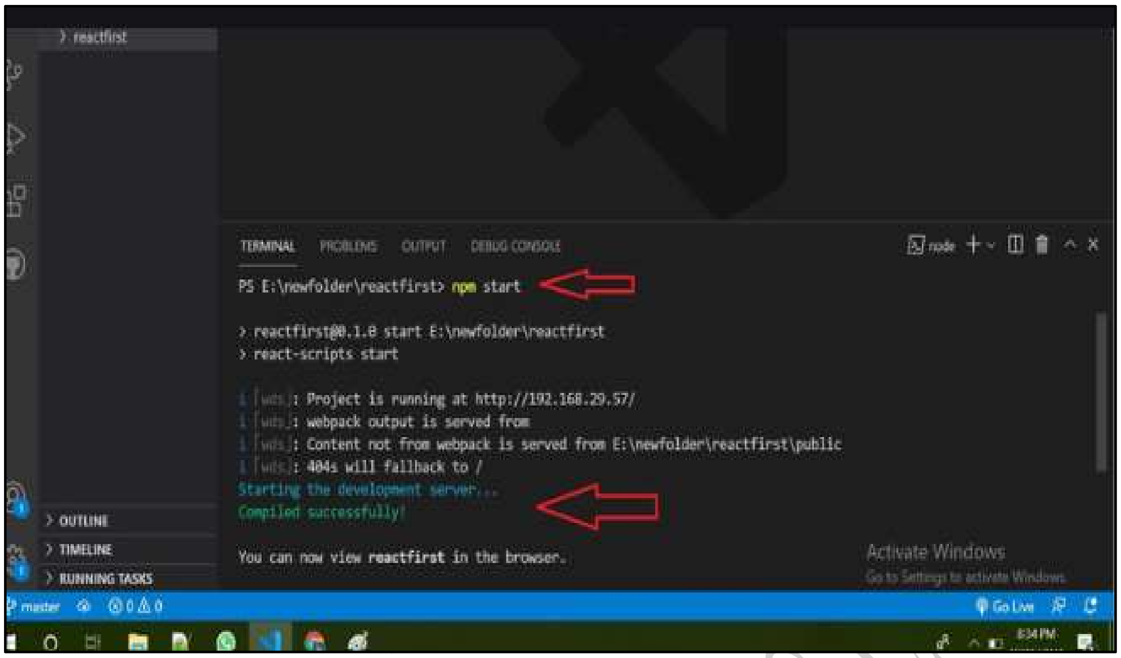


```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS E:\newfolder> cd reactfirst
PS E:\newfolder\reactfirst>
```

Step 7: To start your app run the below command :

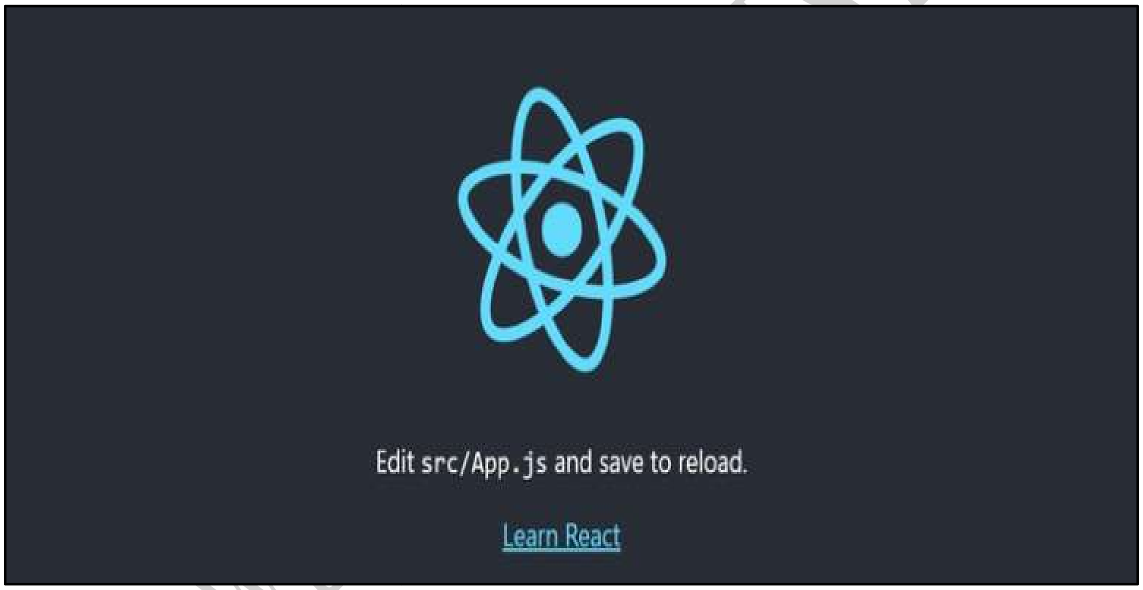
Date		EXP.NO.		Page No.	

npm start



```
PS E:\newfolder\reactfirst> npm start

> reactfirst@0.1.0 start E:\newfolder\reactfirst
> react-scripts start

i ｢wds｣: Project is running at http://192.168.29.57/
i ｢wds｣: webpack output is served from
i ｢wds｣: Content not from webpack is served from E:\newfolder\reactfirst\public
i ｢wds｣: 404s will fallback to /
Starting the development server...
Compiled successfully!

You can now view reactfirst in the browser.
```

Once you run the above command a new tab will open in your browser showing React logo as shown.



Edit src/App.js and save to reload.

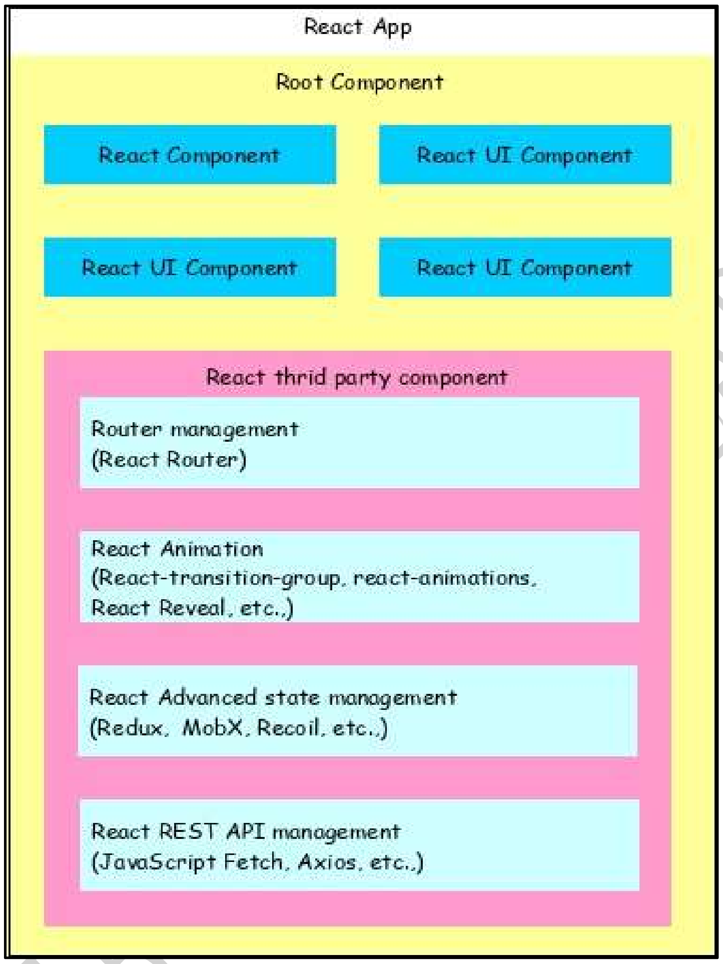
Learn React

Congratulation you have successfully installed the react-app and are ready to build awesome websites and apps

Date		EXP.NO.		Page No.	

ARCHITECTURE

React library is just a UI library and it does not enforce any particular pattern to write a complex application. Developers are free to choose the design pattern of their choice. React community advocates certain design patterns. One of the patterns is the Flux pattern. React library also provides a lot of concepts like Higher Order components, Context, Render props, Refs, etc., to write better code. React Hooks is evolving concept to do state management in big projects. Let us try to understand the high-level architecture of a React application.


React App
Root Component
React Component
React UI Component
React UI Component
React UI Component
React thrid party component
Router management
(React Router)
React Animation
(React-transition-group, react-animations,
React Reveal, etc.,)
React Advanced state management
(Redux, MobX, Recoil, etc.,)
React REST API management
(JavaScript Fetch, Axios, etc.,)

React app starts with a single root component.
The root component is built using one or more components.
Each component can be nested with another component to any level.
Composition is one of the core concepts of React library. So, each component is built by composing smaller components instead of inheriting one component from another component.
Most of the components are user interface components.
React apps can include a third-party component for a specific purpose such as routing, animation, state management, etc.

Date		EXP.NØ.		Page No.	

EXPERIMENT 3
CREATING A REACT APPLICATION

AIM: To create a react Application.

PROCEDURE:

Step1: Setting up a React Environment

If you have npx and Node.js installed, you can create a React application by using create-react-app

Run this command to create a React application named my-react-app:

```
npx create-react-app my-react-app
```

The create-react-app will set up everything you need to run a React application.

Step2: Run the React Application

Now you are ready to run your first real React application!

Run this command to move to the my-react-app directory:

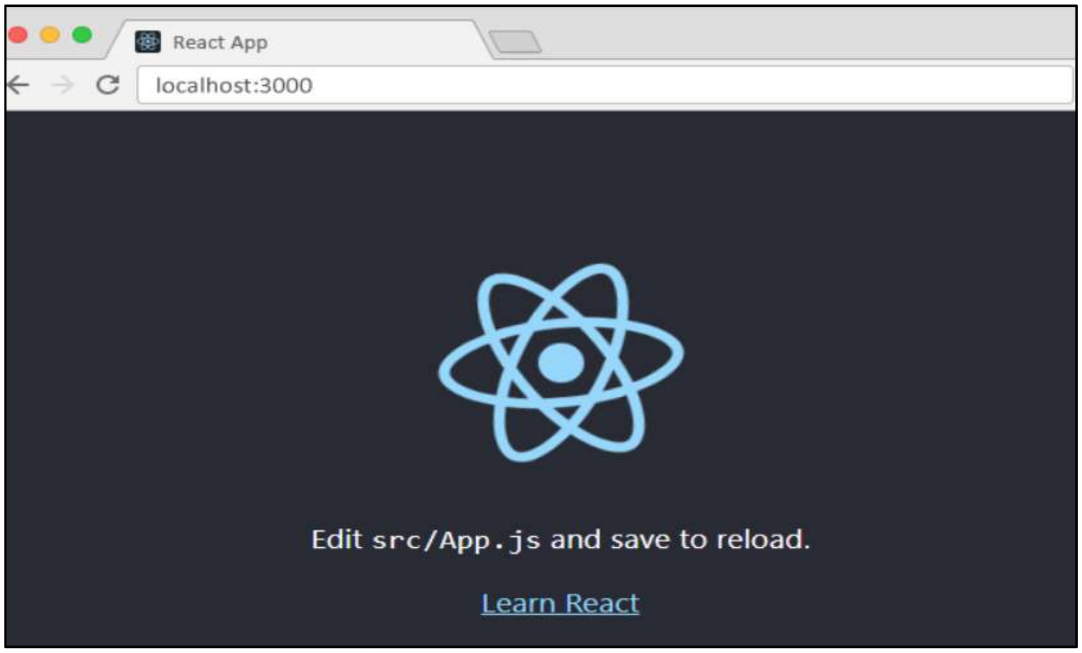
```
cd my-react-app
```

Run this command to run the React application my-react-app:

```
npm start
```

A new browser window will pop up with your newly created React App! If not, open your browser and type localhost:3000 in the address bar.

The result:



React App

localhost:3000

Edit src/App.js and save to reload.

Learn React

Step3: Modify the React Application

So far so good, but how do I change the content?

Look in the my-react-app directory, and you will find an src folder. Inside the src folder there is a file called App.js, open it and it will look like this:

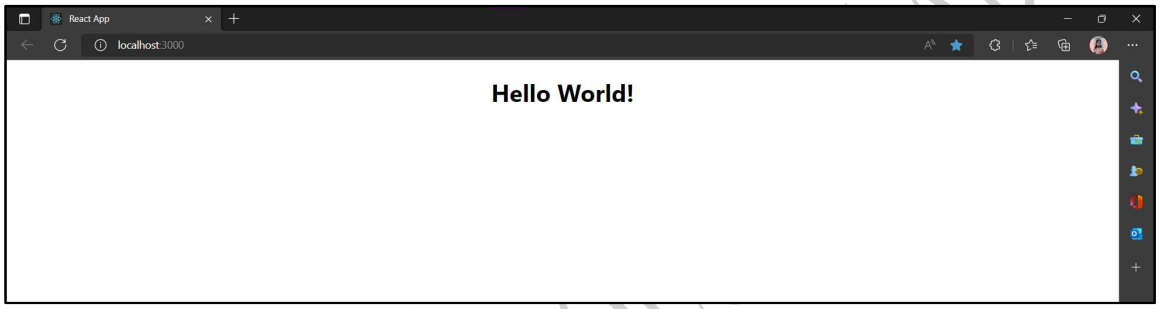
/myReactApp/src/App.js:

```
import logo from './logo.svg';
import './App.css';
function App() {
  return (
<div className="App">
<header className="App-header">
<imgsrc={logo} className="App-logo" alt="logo" />
<p>Edit <code>src/App.js</code> and save to reload. </p>
<a className="App-link"href="https://reactjs.org"target="_blank"rel="noopener noreferrer">
        Learn React</a>
</header>
</div>  );}
export default App;
```

Try changing the HTML content and save the file.
Notice that the changes are visible immediately after you save the file, you do not have to reload the browser!

```
function App() {
  return (
<div className="App">
<h1>Hello World!</h1>
</div>  );}
export default App;
```

OUTPUT:



Hello World!

RESUSLT:

Creating Hello World React Application is Successfully Completed.

Date		EXP.N0.		Page No.	

EXPERIMENT 4
React Application using JSX

AIM: To build a react application using jsx.

DESCRIPTION: JSX provides you to write HTML/XML-like structures (e.g., DOM-like tree structures) in the same file where you write JavaScript code, then preprocessor will transform these expressions into actual JavaScript code. Just like XML/HTML, JSX tags have a tag name, attributes, and children.

PROGRAM 1:

PROGRAM CODE:

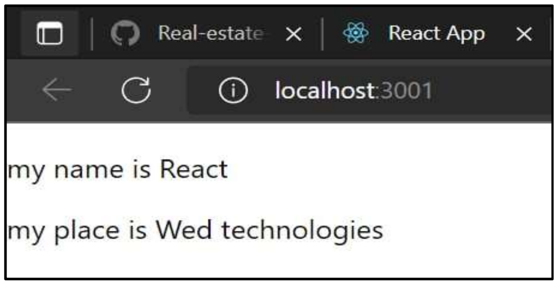
index.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>jsx</title>
</head>
<body>
<div id ="root"></div>
</body>
</html>
```

index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import reportWebVitals from './reportWebVitals';
var name = "React"
var myplace = "Wed technologies"
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
<div>
<p>my name is {name}</p>
<p>my place is {myplace}</p>
</div>
);
```

OUTPUT:



```
my name is React

my place is Wed technologies
```

Date		EXP.N0.		Page No.	

PROGRAM 2:

PROGRAM CODE:

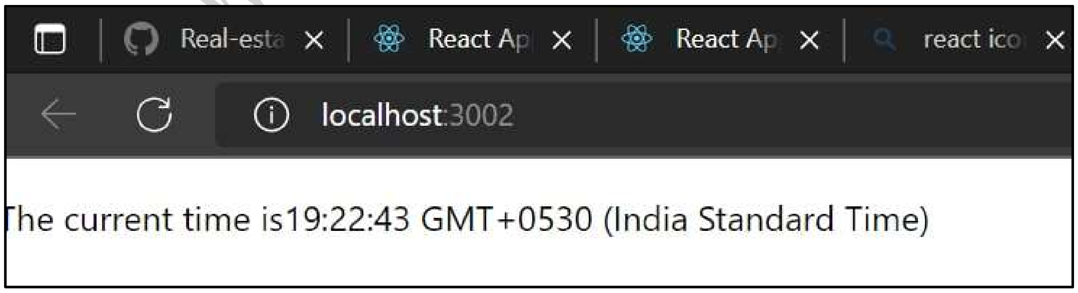
index.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>jsx2</title>
</head>
<body>
<div id="root"></div>

</body>
</html>
```

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'));
function getTime(){
  return new Date().toTimeString();
}
root.render(
<div>
<p>The current time is<span>{getTime()}</span></p>
</div>

)
```

OUTPUT:



```
The current time is19:22:43 GMT+0530 (India Standard Time)
```

RESULT:
React applicatom using JSX is done successfully

Date		EXP.NO.		Page No.	

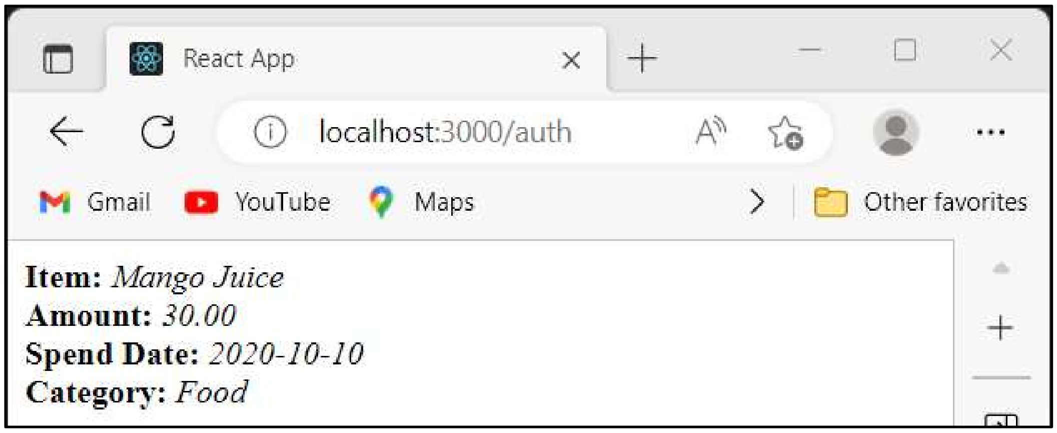
EXPERIMENT NO: 05
COMPONENTS

AIM: To implement component in react js.

DESCRIPTION:
React component is the building block of a React application.A React component represents a small chunk of user interface in a webpage. The primary job of a React component is to render its user interface and update it whenever its internal state is changed. In addition to rendering the UI, it manages the events belongs to its user interface

PROGRAM CODE:
ExpenseEntryItem.js:
```
import React from 'react';
class ExpenseEntryItem extends React.Component {
  render() {
    return (
      <div>
        <div><b>Item:</b><em>Mango Juice</em></div>
        <div><b>Amount:</b><em>30.00</em></div>
        <div><b>Spend Date:</b><em>2020-10-10</em></div>
        <div><b>Category:</b><em>Food</em></div>
      </div>
    );  }
}
export default ExpenseEntryItem;
```

index.js:
```
import React from 'react';
import ReactDOM from 'react-dom';
import ExpenseEntryItem from './components/ExpenseEntryItem'
ReactDOM.render(
  <React.StrictMode>
    <ExpenseEntryItem />
  </React.StrictMode>,
  document.getElementById('root')
);
```

OUTPUT:


```
Item: Mango Juice
Amount: 30.00
Spend Date: 2020-10-10
Category: Food
```

RESULT:
An application using component is executed successfully

Date		EXP.NO.		Page No.	

EXPERIMENT NO: 07
Properties(Props)

AIM: To build a React application using Properties(props).

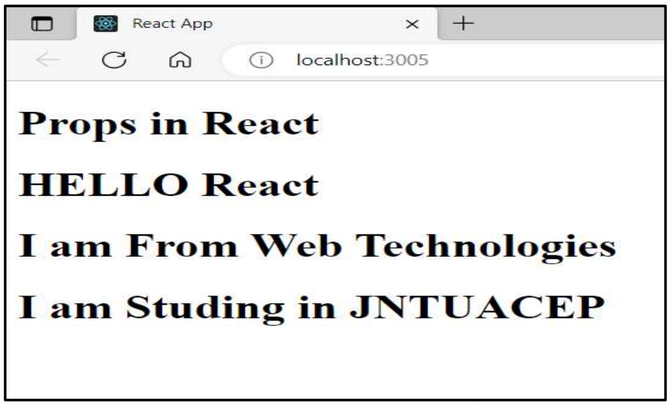
DESCRIPTION: Props stand for "**Properties**." They are **read-only** components. It is an object which stores the value of attributes of a tag and work similar to the HTML attributes. It gives a way to pass data from one component to other components. It is similar to function arguments. Props are passed to the component in the same way as arguments passed in a function.

PROGRAM 1:

PROGRAM CODE:
APP.js:
```
import logo from './logo.svg';
import Student from './Student'
function App(){
return(
<div className="APP">
<h1> Props in React   </h1>
<Student name = "React" place = "Web Technologies" college = "JNTUACEP"/>
</div>
  );
} export default App;
```

Student.js:
```
import React from 'react'
function Student (props){
    return(
      <div >
          <h1>HELLO {props.name}</h1>
          <h1>I am From {props.place}</h1>
          <h1>I am Studing in {props.college}</h1>

      </div>
    );
}
export default Student;
```

OUTPUT:


```
React App
localhost:3005

Props in React
HELLO React
I am From Web Technologies
I am Studing in JNTUACEP
```

Date		EXP.NO.		Page No.	

PROGRAM 2:

PROGRAM CODE:

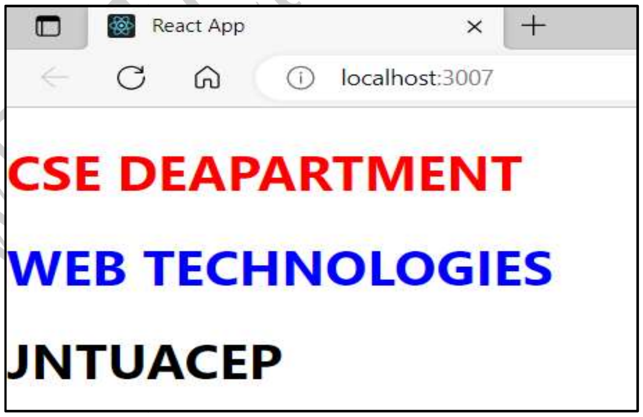
APP.js:

```
import React from 'react';
//import './APP.css'
import Student2 from './Student2';
function App1(){
   return(
      <div className='APP1'>
         <Student2 text="CSE DEAPARTMENT" color = 'red'/>
         <Student2 text="WEB TECHNOLOGIES" color = "blue"/>
         <Student2 text = "JNTUACEP" color = "black"/>
      </div>
   );
}
export default App1;
```

Student2.js:

```
import React from 'react';
function Student2(props){
   return(
      <div className='Student2'>
         <h1 style={{color:props.color}}>{props.text}</h1>
      </div>
   );
}
export default Student2;
```

OUTPUT:



CSE DEAPARTMENT

WEB TECHNOLOGIES

JNTUACEP

RESULT:
React js application using properties(Props) is done successfully

Date		EXP.N0.		Page No.	

EXPERIMENT NO:08
Event Management

AIM: To implement Event Handling in reactJs.

DESCRIPTION: Event Management enables the user to interact with the application. React event handling is very similar to DOM events with little changes. React supports all events available in a web application

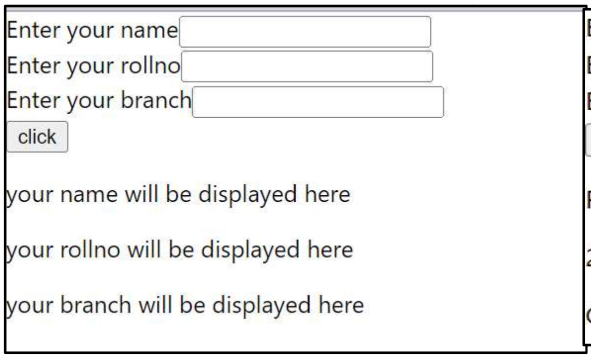
PROGRAM CODE:

App.js:

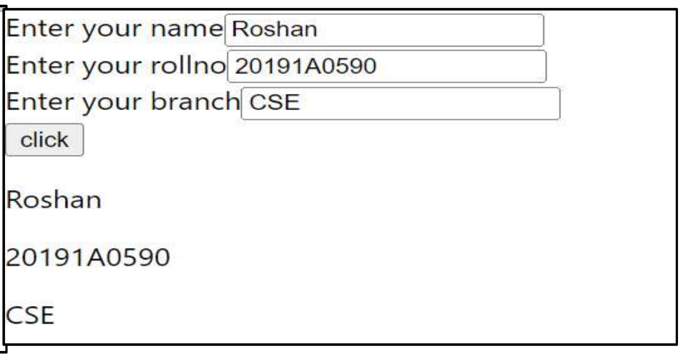
```
import React, { useState } from "react"
function App(){
  const [enname,setname]=useState("your name will be displayed here")
  const [enrno,setrno]=useState("your rollno will be displayed here")
  const [enbranch,setbranch]=useState("your branch will be displayed here")
  function events(){
    setname(document.getElementById("nameip").value)
    setrno(document.getElementById("rnip").value)
    setbranch(document.getElementById("bip").value)
  }
  return(
    <div>
      <label>Enter your name</label>
      <input type="text"id="nameip"/><br></br>
      <label>Enter your rollno</label>
      <input type="text"id="rnip"/><br></br>
      <label>Enter your branch</label>
      <input type="text"id="bip"/><br></br>
      <button onClick={events}>click</button>
      <p id="display1">{enname}</p>
      <p id="display2">{enrno}</p>
      <p id="display3">{enbranch}</p>
    </div>
  );
}export default App;
```

OUTPUT:

Before entering the details:After entering the details:



```
Enter your name
Enter your rollno
Enter your branch
click

your name will be displayed here

your rollno will be displayed here

your branch will be displayed here
```



```
Enter your name Roshan
Enter your rollno 20191A0590
Enter your branch CSE
click

Roshan

20191A0590

CSE
```

RESULT:

Implementing Event Handling using reactJs is done successfully.

EXPERIMENT NO:09
STATE MANAGEMENT

AIM: To implement state management in reactjs

DESCRIPTION:
State management is the core of any modern web application, as it determines what data is displayed on the screen during the app usage session while the user interacts with it.
For example, ticking checkboxes in online surveys, adding products to a shopping cart on an e-commerce store, or selecting audio from a playlist in a music player is possible thanks to state management keeping track of each action the user makes.

PROGRAM CODE:
AddMovie.js:
```
import React, { useState, useContext } from "react";
import { MovieContext } from "./MovieContext";
constAddMovie = () => {
const [name, setName] = useState("");
const [price, setPrice] = useState("");
const [movies, setMovies] = useContext(MovieContext);
constupdateName = e => {
setName(e.target.value);
  };
constupdatePrice = e => {
setPrice(e.target.value);
  };
constaddMovie = e => {
e.preventDefault();
setMovies(preMovies => [...preMovies, { name: name, price: price }]);
  };
  return (
<form onSubmit={addMovie}>
<input type="text" name="name" value={name} onChange={updateName} />
<input type="text" name="price" value={price} onChange={updatePrice} />
<button>Submit</button>
</form>
  );};
export default AddMovie;
```

App.js:
```
import "./styles.css";
import React from 'react';
import MovieList from './Movielist';
import Nav from './Nav';
import AddMovie from './AddMovie';
import {MovieProvider} from './MovieContext';
function App() {
  return (
<MovieProvider>
<div className="App">
<Nav/>
<AddMovie/>
<MovieList/>
</div>
</MovieProvider>
```

```
  );}
export default App;
```

Movie.js:

```
import React from 'react';
const Movie = ({name,price}) => {
return(
<div>
<h3>{name}</h3>
<p>{price}</p>
</div> );}
export default Movie;
```

MovieContest.js:

```
import React, { useState, createContext } from "react";
import Nav from "./Nav";
export constMovieContext = createContext();
export constMovieProvider = (props) => {
const [movies, setMovies] = useState([
{ name: "Harry Potter",
     price: "$10",
     id: "23124"  },
{ name: "Game of Thrones",
     price: "$10",
     id: "23127"  },
{ name: "Inception",
     price: "$10",
     id: "23123"  } ]);
  return (
<MovieContext.Provider value={[movies, setMovies]}>
      {props.children}
</MovieContext.Provider> );};
```

Movielist.js:

```
import React,{useState} from 'react';
import Movie from './Movie';
import {MovieContext} from './MovieContext';
constMovielist = () => {
const[movies,setMovies]=useState([
{ name:'Harry Potter',
     price:'$10',
     id:'23124'  },
{ name:'Game of Thrones',
     price:'$10',
     id:'23127' },
{  name:'Inception',
     price:'$10',
     id:'23123'  }  ]);
return(
<div>
    {movies.map(movie => (
<Movie name={movie.name} price={movie.price} key={movie.id}/>))}
</div> );}
export default Movielist;Nav.js:
import React, { useContext } from 'react';
```

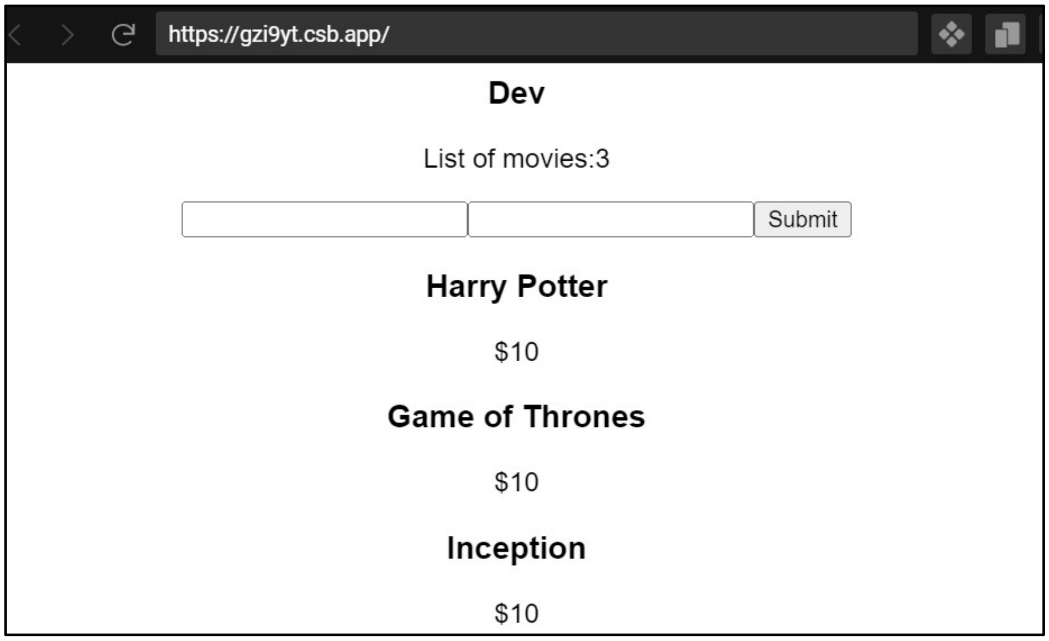
Date		EXP.N0.		Page No.	

```
import {MovieContext} from './MovieContext';
const Nav = () => {
const [movies,setMovies]=useContext(MovieContext);
return(
<div>
<h3>Dev</h3>
<p>List of movies:{movies.length}</p>
</div>  );}
export default Nav;
```

index.js:

```
import { StrictMode } from "react";
import { createRoot } from "react-dom/client";
import App from "./App";
constrootElement = document.getElementById("root");
const root = createRoot(rootElement);
root.render(
<StrictMode><App /></StrictMode>);
```

OUTPUT:



```
Dev

List of movies:3

Harry Potter
$10
Game of Thrones
$10
Inception
$10
```

RESULT:

An application for state management is executed successfully.

EXPERIMENT NO:13
Animations

AIM: To build a React application using Animations

DESCRIPTION:
As front-end developer, a popular request you might get from your clients is to implement stunning animation effects on page scroll. **AOS** (Animate on Scroll) is the most popular library for the purpose of making this task easier for us.
Framer Motion is a production-ready motion library for React from Framer.
It's simple yet powerful, allowing you to express complex user interactions with robust, semantic markup.
The <motion /> component
The core of the library is the motion component. Think of it as a plain HTML or SVG element, supercharged with animation capabilities.

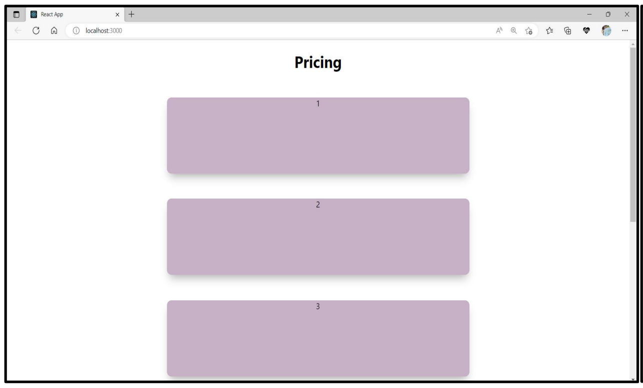
PROGRAM CODE:
App.js
```
import './App.css';
import React, { Component } from 'react'
import Pricing from './components/Pricing'
class App extends Component {
   render() {
      return (
         <div className="App">
            <Pricing /></div>)}}
export default App
```

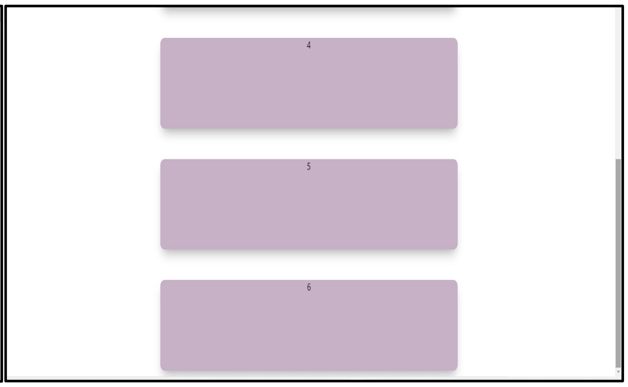
Pricing.js
```
import React, { useEffect } from "react";
import AOS from "aos";
import "aos/dist/aos.css"
import "./Boxes.css";
const Pricing=()=>{
useEffect(()=>{
AOS.init({ duration : 2000});    }, []);
return(
<div className="App">
<h1>Pricing</h1>
<div className="grids">
<div className="boxes">1</div>
<div className="boxes">2</div>
<div data-aos="fade-up" className="boxes">3</div>
<div data-aos="fade-left" className="boxes">4</div>
<div data-aos="fade-right" className="boxes">5</div>
<div data-aos="flip-right" className="boxes">6</div>
</div>
</div>  );}};
export default Pricing;
```

Boxes.css
```
.boxes
{  margin:10px;
   margin-top: 50px;
```

```
    background-color: rgb(198,177,198);
    overflow: hidden;
    height: 150px;
    max-height: 35vh;
    box-shadow: 0px 12px 18px -6px rgb(0,0,0,0.3);
    border-radius: 10px 10px10px 10px;}
.grids{
    max-width: 50%;
    margin: auto;}
```

OUTPUT:





PROGRAM2:

App.js

```
import './App.css';
import React, { Component } from 'react'
import Framer from './components/Framer'
class App extends Component {
    render() {
        return (
            <div className="App">
             <Framer />
            </div>       )  }}
export default App
```

Framer.js

```
import React from "react";
import "./Framer.css";
import {motion} from "framer-motion"
function Framer(){
    return(
      <div className="b">
        <motion.div animate={{scale:2 }} className="d"></motion.div>
     <motion.h2 animate={{color:'#111',x:-100,  fontSize:100}} className="d1">Welcome</motion.h2>
        <motion.button animate={{rotate:180 ,x:-300,y:-100}} className="b1">Submit</motion.button>
      </div>    );}
export default Framer;
```

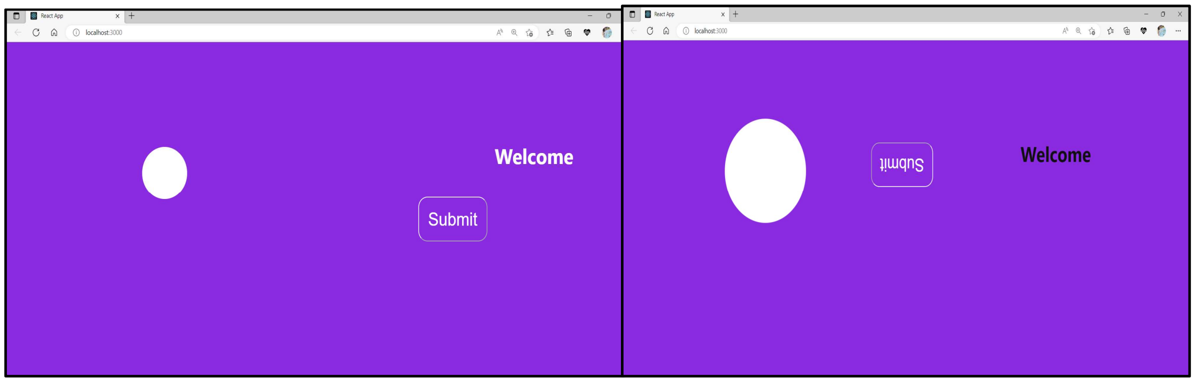
Framer.css

```
.d{ width:100px;
    height: 100px;
    border-radius: 50%;
    background-color: white;
```

Date		EXP.NO.		Page No.	

```
    margin-top: 200px;
    margin-left: 300px;}
body{  background-color: blueviolet;
    color: white;}
.d1{font-size: 40px;
    margin-left: 940px;
    margin-top: -110px;
    color:white;
}
.b1{
    margin-left: 580px;
    margin-top: 20px;
   background-color: blueviolet;
   border-radius: 20px;
   font-size: 35px;
   padding: 20px;
   color:white;
   border-color: white;
}
```

OUTPUT:



RESULT:
An animation in reactjs is completed successfully.

EXPERIMENT:10

Http Client Programming

AIM: To implement Http Client Programming in reactjs.

DESCRIPTION:
Http client programming enables the application to connect and fetch data from http server through JavaScript. It reduces the data transfer between client and server as it fetches only the required data instead of the whole design and subsequently improves the network speed. React does not provide it's own http programming api but it supports browser's built-in *fetch()* api as well as third party client library like axios to do client side programming.

PROGRAM CODE:
HTTP GET REQUEST

```
import './App.css';
import React, { Component } from 'react'
import PostList from './components/PostList'
class App extends Component {
        render() {
                return (
                        <div className="App">
                        <PostList />    </div> )}}
export default App
```

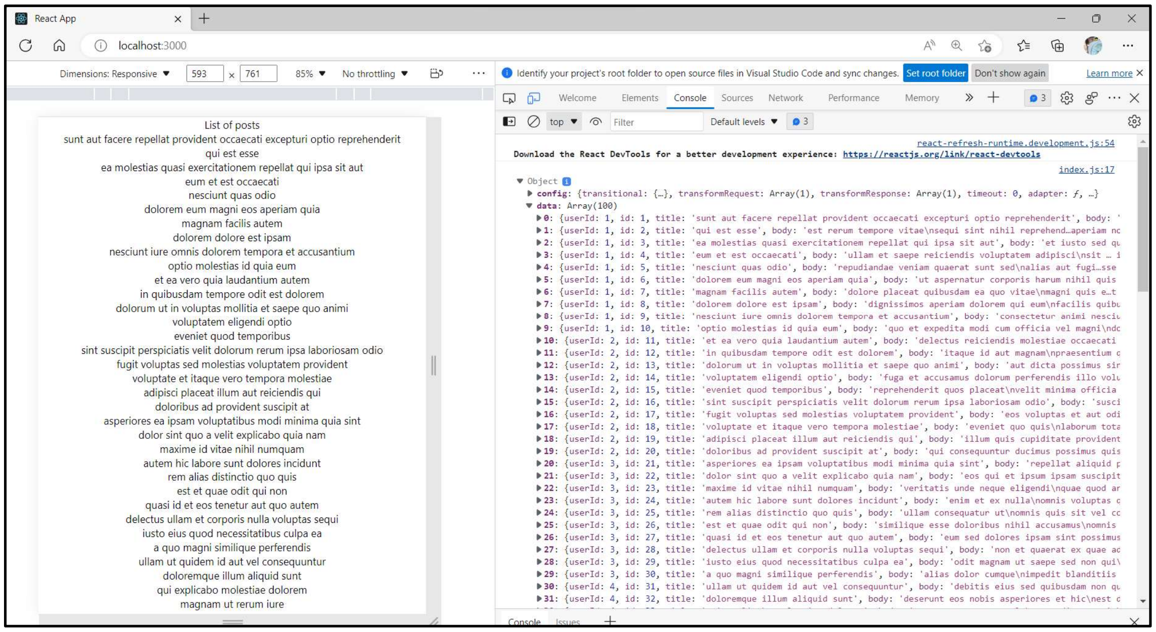
PostList.js

```
import React, { Component } from 'react'
import axios from 'axios'
class PostList extends Component {
   constructor(props) {
      super(props)
      this.state = {
     posts: [],
     errorMsg: ''       }    }
   componentDidMount() {
      axios
         .get('https://jsonplaceholder.typicode.com/posts')
         .then(response => {
            console.log(response)
            this.setState({ posts: response.data })
         })
         .catch(error => {
      console.log(error)
      this.setState({errorMsg: 'Error retrieving data'})
         })    }
   render() {
      const{ posts, errorMsg } = this.state
      return (
         <div>  List of posts
            {posts.length
               ? posts.map(post =><div key={post.id}>{post.title}</div>)
: null}
      {errorMsg ?<div>{errorMsg}</div> : null}
</div>
)
 }}
```

Date		EXP.N0.		Page No.	

```
export default PostList
```

OUTPUT:



```
List of posts
sunt aut facere repellat provident occaecati excepturi optio reprehenderit
qui est esse
ea molestias quasi exercitationem repellat qui ipsa sit aut
eum et est occaecati
nesciunt quas odio
dolorem eum magni eos aperiam quia
magnam facilis autem
dolorem dolore est ipsam
nesciunt iure omnis dolorem tempora et accusantium
optio molestias id quia eum
et ea vero quia laudantium autem
in quibusdam tempore odit est dolorem
dolorum ut in voluptas mollitia et saepe quo animi
voluptatem eligendi optio
eveniet quod temporibus
sint suscipit perspiciatis velit dolorum rerum ipsa laboriosam odio
fugit voluptas sed molestias voluptatem provident
voluptate et itaque vero tempora molestiae
adipisci placeat illum aut reiciendis qui
doloribus ad provident suscipit at
asperiores ea ipsam voluptatibus modi minima quia sint
dolor sint quo a velit explicabo quia nam
maxime id vitae nihil numquam
autem hic labore sunt dolores incidunt
rem alias distinctio quo quis
est et quae odit qui non
quasi id et eos tenetur aut quo autem
delectus ullam et corporis nulla voluptas sequi
iusto eius quod necessitatibus culpa ea
a quo magni similique perferendis
ullam ut quidem id aut vel consequuntur
doloremque illum aliquid sunt
qui explicabo molestiae dolorem
magnam ut rerum iure
```

```
                                                                    react-refresh-runtime.development.js:54
Download the React DevTools for a better development experience: https://reactjs.org/link/react-devtools
                                                                                             index.js:17
▼ Object
  ▶ config: {transitional: {…}, transformRequest: Array(1), transformResponse: Array(1), timeout: 0, adapter: ƒ, …}
  ▼ data: Array(100)
    ▶0: {userId: 1, id: 1, title: 'sunt aut facere repellat provident occaecati excepturi optio reprehenderit', body: '
    ▶1: {userId: 1, id: 2, title: 'qui est esse', body: 'est rerum tempore vitae\nsequi sint nihil reprehend…aperiam nc
    ▶2: {userId: 1, id: 3, title: 'ea molestias quasi exercitationem repellat qui ipsa sit aut', body: 'et iusto sed qu
    ▶3: {userId: 1, id: 4, title: 'eum et est occaecati', body: 'ullam et saepe reiciendis voluptatem adipisci\nsit … i
    ▶4: {userId: 1, id: 5, title: 'nesciunt quas odio', body: 'repudiandae veniam quaerat sunt sed\nalias aut fugi…sse
    ▶5: {userId: 1, id: 6, title: 'dolorem eum magni eos aperiam quia', body: 'ut aspernatur corporis harum nihil quis
    ▶6: {userId: 1, id: 7, title: 'magnam facilis autem', body: 'dolore placeat quibusdam ea quo vitae\nmagni quis e…t
    ▶7: {userId: 1, id: 8, title: 'dolorem dolore est ipsam', body: 'dignissimos aperiam dolorem qui eum\nfacilis quibu
    ▶8: {userId: 1, id: 9, title: 'nesciunt iure omnis dolorem tempora et accusantium', body: 'consectetur animi nesciu
    ▶9: {userId: 1, id: 10, title: 'optio molestias id quia eum', body: 'quo et expedita modi cum officia vel magni\ndc
    ▶10: {userId: 2, id: 11, title: 'et ea vero quia laudantium autem', body: 'delectus reiciendis molestiae occaecati
    ▶11: {userId: 2, id: 12, title: 'in quibusdam tempore odit est dolorem', body: 'itaque id aut magnam\npraesentium c
    ▶12: {userId: 2, id: 13, title: 'dolorum ut in voluptas mollitia et saepe quo animi', body: 'aut dicta possimus sir
    ▶13: {userId: 2, id: 14, title: 'voluptatem eligendi optio', body: 'fuga et accusamus dolorum perferendis illo volu
    ▶14: {userId: 2, id: 15, title: 'eveniet quod temporibus', body: 'reprehenderit quos placeat\nvelit minima officia
    ▶15: {userId: 2, id: 16, title: 'sint suscipit perspiciatis velit dolorum rerum ipsa laboriosam odio', body: 'susci
    ▶16: {userId: 2, id: 17, title: 'fugit voluptas sed molestias voluptatem provident', body: 'eos voluptas et aut odi
    ▶17: {userId: 2, id: 18, title: 'voluptate et itaque vero tempora molestiae', body: 'eveniet quo quis\nlaborum tota
    ▶18: {userId: 2, id: 19, title: 'adipisci placeat illum aut reiciendis qui', body: 'illum quis cupiditate provident
    ▶19: {userId: 2, id: 20, title: 'doloribus ad provident suscipit at', body: 'qui consequuntur ducimus possimus quis
    ▶20: {userId: 3, id: 21, title: 'asperiores ea ipsam voluptatibus modi minima quia sint', body: 'repellat aliquid p
    ▶21: {userId: 3, id: 22, title: 'dolor sint quo a velit explicabo quia nam', body: 'eos qui et ipsum ipsam suscipit
    ▶22: {userId: 3, id: 23, title: 'maxime id vitae nihil numquam', body: 'veritatis unde neque eligendi\nquae quod ar
    ▶23: {userId: 3, id: 24, title: 'autem hic labore sunt dolores incidunt', body: 'enim et ex nulla\nomnis voluptas c
    ▶24: {userId: 3, id: 25, title: 'rem alias distinctio quo quis', body: 'ullam consequatur ut\nomnis quis sit vel cc
    ▶25: {userId: 3, id: 26, title: 'est et quae odit qui non', body: 'similique esse doloribus nihil accusamus\nomnis
    ▶26: {userId: 3, id: 27, title: 'quasi id et eos tenetur aut quo autem', body: 'eum sed dolores ipsam sint possimus
    ▶27: {userId: 3, id: 28, title: 'delectus ullam et corporis nulla voluptas sequi', body: 'non et quaerat ex quae ac
    ▶28: {userId: 3, id: 29, title: 'iusto eius quod necessitatibus culpa ea', body: 'odit magnam ut saepe sed non qui\
    ▶29: {userId: 3, id: 30, title: 'a quo magni similique perferendis', body: 'alias dolor cumque\nimpedit blanditiis
    ▶30: {userId: 4, id: 31, title: 'ullam ut quidem id aut vel consequuntur', body: 'debitis eius sed quibusdam non qu
    ▶31: {userId: 4, id: 32, title: 'doloremque illum aliquid sunt', body: 'deserunt eos nobis asperiores et hic\nest c
```

HTTP POST REQUEST

App.js

```
import './App.css';
import React, { Component } from 'react'
import PostForm from './components/PostForm'
class App extends Component {
        render() {
                return (
                        <div className="App">
                        <PostForm />
                        </div>          )          }}
export default App
```

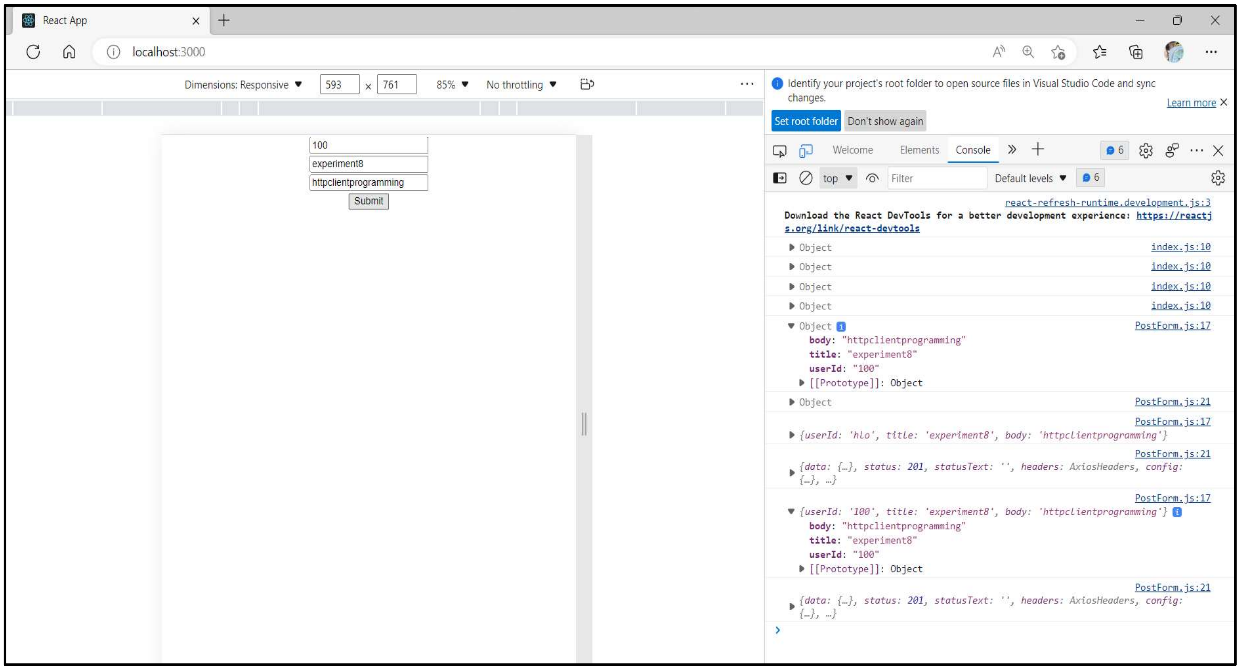
PostForm.js

```
import React, { Component } from 'react'
import axios from 'axios'
class PostForm extends Component {
   constructor(props) {
      super(props)
      this.state = {
         userId: '',
         title: '',
         body: ''       }    }
   changeHandler = e => {
      this.setState({ [e.target.name]: e.target.value })
   }
   submitHandler = e => {
      e.preventDefault()
      console.log(this.state)
      axios
         .post('https://jsonplaceholder.typicode.com/posts', this.state)
         .then(response => {
```

Date		EXP.NO.		Page No.	

```
          })
          .catch(error => {
             console.log(error)           })    }
   render() {
      const{ userId, title, body } = this.state
      return (
         <div>
            <form onSubmit={this.submitHandler}>
               <div>
                  <input type="text" name="userId" value={userId} onChange={this.changeHandler} />
               </div>
               <div>
                  <input type="text" name="title" value={title} onChange={this.changeHandler} />
               </div>
               <div>
                  <input type="text" name="body" value={body} onChange={this.changeHandler} />
               </div>
               <button type="submit">Submit</button>
            </form>
         </div>        )    }}
export default PostForm
```

OUTPUT:



Identify your project's root folder to open source files in Visual Studio Code and sync changes.

```
                                              react-refresh-runtime.development.js:3
Download the React DevTools for a better development experience: https://reactjs.org/link/react-devtools
▶ Object                                                              index.js:10
▶ Object                                                              index.js:10
▶ Object                                                              index.js:10
▶ Object                                                              index.js:10
▼ Object                                                           PostForm.js:17
    body: "httpclientprogramming"
    title: "experiment8"
    userId: "100"
  ▶ [[Prototype]]: Object
▶ Object                                                           PostForm.js:21
                                                                   PostForm.js:17
▶ {userId: 'hlo', title: 'experiment8', body: 'httpclientprogramming'}
                                                                   PostForm.js:21
▶ {data: {…}, status: 201, statusText: '', headers: AxiosHeaders, config:
  {…}, …}
                                                                   PostForm.js:17
▼ {userId: '100', title: 'experiment8', body: 'httpclientprogramming'}
    body: "httpclientprogramming"
    title: "experiment8"
    userId: "100"
  ▶ [[Prototype]]: Object
                                                                   PostForm.js:21
▶ {data: {…}, status: 201, statusText: '', headers: AxiosHeaders, config:
  {…}, …}
›
```

RESULT:

An application for Http client programming is executed successfully.

EXPERIMENT:11
Routing

AIM: To implement Routing in reactjs.

DESCRIPTION:
Routing is a process in which a user is directed to different pages based on their action or request. ReactJS Router is mainly used for developing Single Page Web Applications. React Router is used to define multiple routes in the application.

PROGRAM CODE:
App.js
```
import React from 'react'
import './App.css'
import Navbar from './Navbar'
import Home from './Home'
import Dashboard from './Dashboard'
import About from './About'
import {BrowserRouter,Routes,Route} from 'react-router-dom'
function App() {
                return (
                        <div>
                                <BrowserRouter>
                                <Navbar />
                                        <Routes>
                                                <Route path="/" element={<Home />}/>
                                                <Route path="/Dashboard" element={<Dashboard />}/>
                                                <Route path="/about" element={<About />}/>
                                        </Routes>
                                </BrowserRouter>
                        </div>          )       }

export default App
```

Navbar.js
```
import React from 'react'
import {Link} from 'react-router-dom'
import './Navbar.css'
const Navbar = () => {
  return (
   <div>
     <ul class="nav">
       <Link to="/"><li>Home&nbsp;</li></Link>
       <Link to="/dashboard"><li>Dashboard</li></Link>
       <Link to="/about"><li>About</li></Link>
     </ul>
   </div>  )}
export default Navbar
```

Home.js
```
import React from 'react'
const Home = () => {
  return (
   <div><center>
     <h4>Welcome to home page</h4>
```

```
        <p>this is home page content</p>
        </center>
    </div>  )}
export default Home
```

Dashboard,js

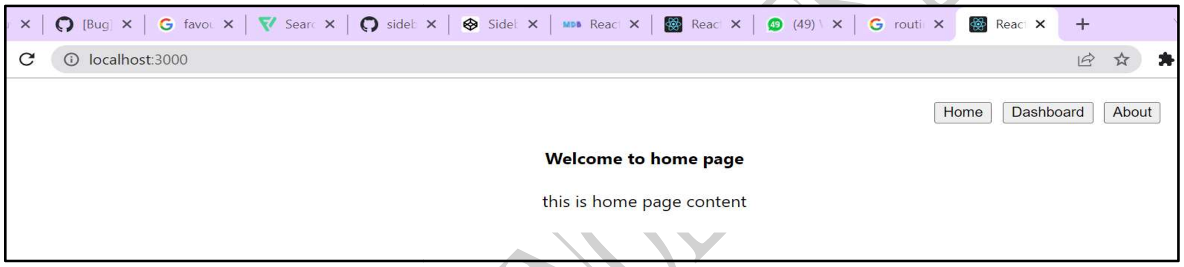
```
import React from 'react'
const Dashboard = () => {
  return (
    <div><center>Dashboard page content</center></div>  )}
export default Dashboard
```

About,js

```
import React from 'react'
const About = () => {
  return (
    <div><center>About page content</center></div>  )}
export default About
```

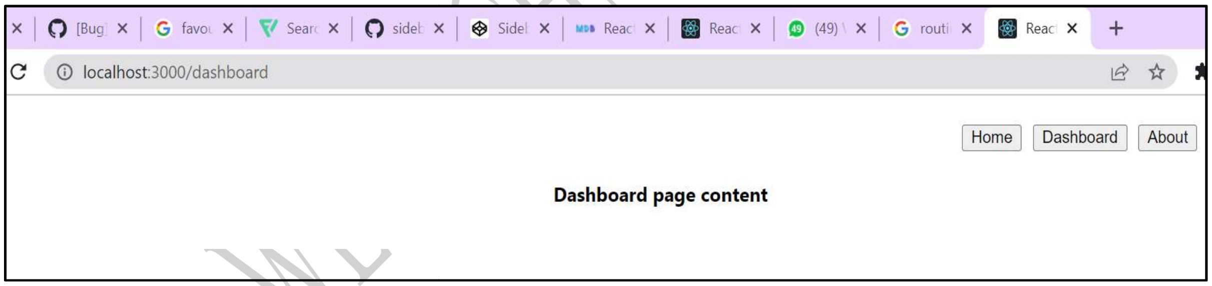
OUTPUT:

1. when we click on the home button in navbar



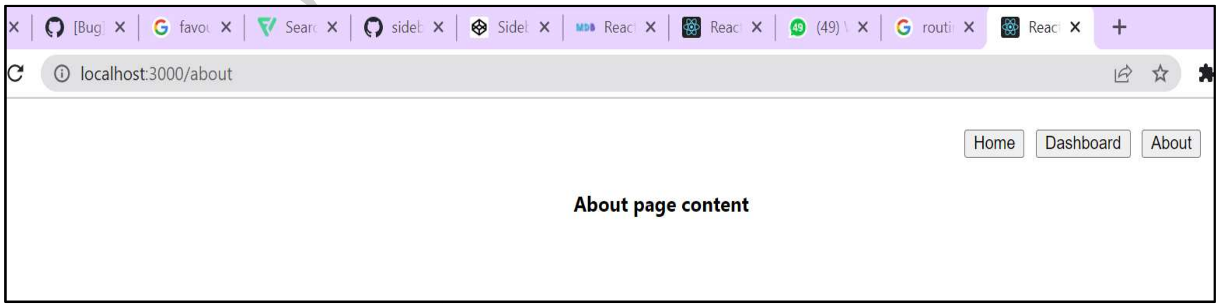
```
Welcome to home page

this is home page content
```

2. when we click on the Dashboard button in navbar



```
Dashboard page content
```

3. when we click on the About button in nav bar



```
About page content
```

RESULT:

React application using routing is done successfully

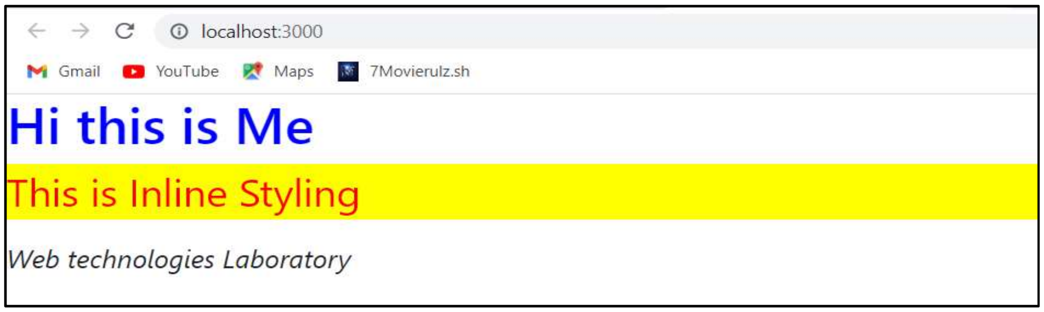
Date		EXP.NO.		Page No.	

EXPERIMENT:06
Styling

AIM: To implement Styling in reactjs.

DESCRIPTION:
In general, React allows component to be styled using CSS class through className attribute. Since, the React JSX supports JavaScript expression, a lot of common CSS methodology can be used. Some of the top options are as follows
1.Inline Styling
2.CSS Stylesheet
3.styled Components
4.SASS Stylesheet

PROGRAM CODE:
INLINE STYLING
```
function App() {
return(
    <div>
      <h1 style={{color:"blue"}}>Hi this is Me</h1>
      <p style={{color:"red" , fontSize:"30px" , background:"yellow"}}>This is Inline Styling</p>
      <p style={{fontStyle:"italic" , fontSize:"20px"}}>Web technologies Laboratory</p>
    </div>
  );}
export default App;
```

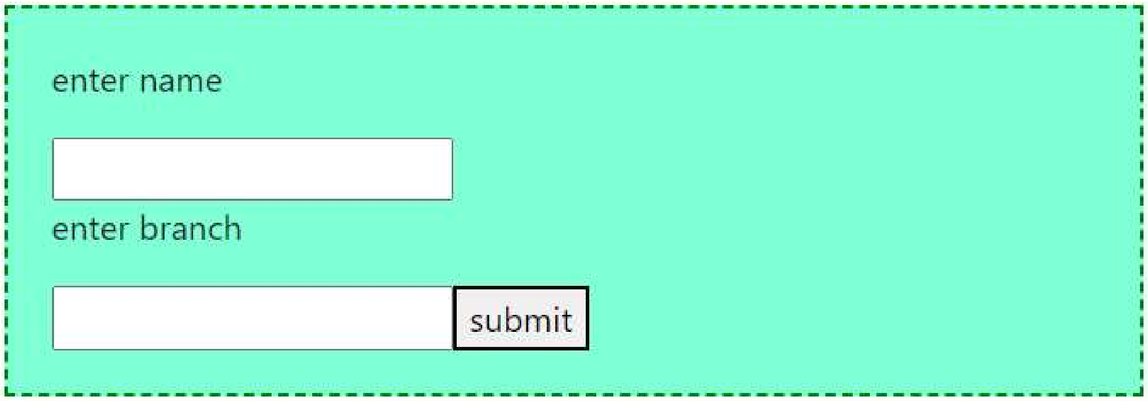
OUTPUT:


Hi this is Me
This is Inline Styling
Web technologies Laboratory

USING CSS:

App.css
```
.myform{
  border-style: dashed;
  border-width: 2px;
  border-color: green;
  background-color: aquamarine;
  padding: 20px;
  margin-left: 500px;
  margin-right: 500px;
  margin-top: 100px;
}
```
App.js
```
import './App.css';
function App() {
  return(
      <p>enter name</p>
      <input type="text" />
      <p>enter branch</p>
      <input type="text" />
      <button>submit</button><br/>
    </div>
  );
}
export default App;
```

OUTPUT:



USING STYLED COMPONENTS:

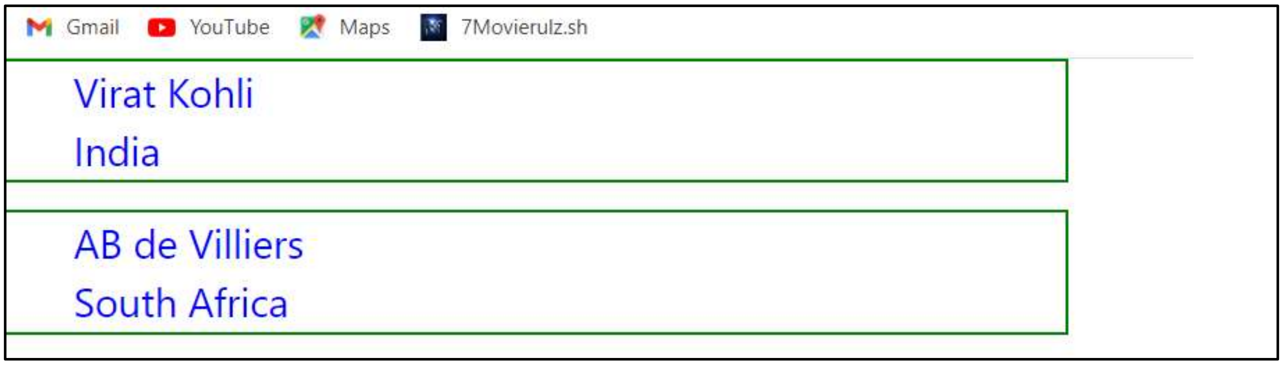
Styling.js

```
import React from "react";
import styled from 'styled-components';
const Li = styled.li`
  color : blue;
  font-size : 23px`
constUl = styled.ul`
  border : 2px solid green;
  width: 40%;
  list-style-type:none`
function Styling(props){
    return(
      <Ul>
        <Li>{props.name}</Li>
        <Li>{props.country}</Li>
      </Ul>  );}
export default Styling;
```

app.js

```
import Styling from './Styling';
function App() {
  return(
    <div>
      <Styling name="Virat Kohli" country="India"/>
      <Styling name="AB de Villiers" country="South Africa" />
    </div>  );}
export default App;
```

OUTPUT:



```
Virat Kohli
India

AB de Villiers
South Africa
```

USING SASS STYLE SHEET:

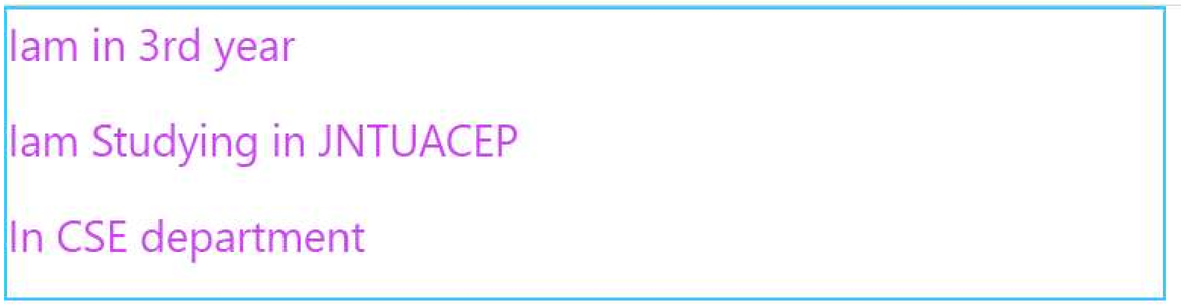
Stylenew.scss

```
$purple: #d142f5;
$pale-blue-color: #42c8f5;
 .list{
    border: 2px solid $pale-blue-color;
    width: 40%;
    list-style-type: none;    }
 .details{ color: $purple;
    font-size: 23px;}
```

App.js

```
import './Stylenew.scss';
function App() {
  return(
    <div>
      <div className='list'>
        <p className='details'>Iam in 3rd year</p>
        <p className='details'>Iam Studying in JNTUACEP</p>
        <p className='details'>In CSE department</p>
      </div>    </div>  );}
export default App;
```

OUTPUT:



```
Iam in 3rd year

Iam Studying in JNTUACEP

In CSE department
```

RESULT:
An application using various ways of styling is executed successfully.

EXPERIMENT NO: 12

ReactJS – Redux

AIM : Implementing a Redux library using React.

DESCRIPTION :
Redux is a predictable state container designed to help you write JavaScript apps that behave consistently across client, server, and native environments, and are easy to test. While it's mostly used as a state management tool with React, you can use it with any other JavaScript framework or library. React Redux is the official React binding for Redux. It allows React components to read data from a Redux Store, and dispatch Actions to the Store to update data.

PROGRAM CODE :
index.js:

```
import { StrictMode } from "react";
import ReactDOM from 'react-dom';
import App from "./App";
import { Store } from "./app/Store";
import { Provider } from "react-redux";
ReactDOM.render(
 <StrictMode>
 <Provider store={Store}>
 <App />
 </Provider>
 </StrictMode>,
 document.getElementById('root')
);
```

App.js:

```
import "./styles.css";
import Counter from "./features/counter/Counter";
export default function App() {
 return (
 <main className="App">
 <Counter />
 </main>
 );
}
```

Store.js:

```
import { configureStore } from "@reduxjs/toolkit";
import counterReducer from "../features/counter/counterSlice";
export const Store = configureStore({
 reducer: {
 counter: counterReducer,
 }
})
```

counterSlice.js:

```
import { createSlice } from "@reduxjs/toolkit";
const initialState = {
 count : 0
}
export const counterSlice = createSlice({
 name: 'counter',
 initialState,
 reducers: {
```

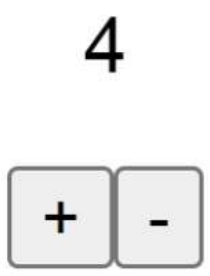
Date		EXP.N0.		Page No.	

```
 increment: (state) => {
 state.count += 1;
 },
 decrement: (state) => {
state.count -= 1;
 }
 }
});
export const { increment, decrement } = counterSlice.actions;
export default counterSlice.reducer;
```

Counter.js:

```
import { useSelector, useDispatch } from "react-redux";
import { increment, decrement } from "./counterSlice";
const Counter = () => {
 const count = useSelector((state) => state.counter.count);
 const dispatch = useDispatch();
 return (
 <section>
 <p>{count}</p>
 <div>
 <button onClick={() => dispatch(increment())}>+</button>
 <button onClick={() => dispatch(decrement())}>-</button>
</div>
 </section>
 )
}
export default Counter
```

OUTPUT:



RESULT:
Implementation of Redux has been executed successfully.