Experiment No – 6

Aim

To implement an application using React JS

Theory:

React Js

ReactJS is a declarative, efficient, and flexible JavaScript library for building reusable UI components. It is an open-source, component-based front-end library which is responsible only for the view layer of the application. It was initially developed and maintained by Facebook and later used in its products like WhatsApp & Instagram.

A ReactJS application is made up of multiple components, each component responsible for outputting a small, reusable piece of HTML code. The components are the heart of all React applications. These Components can be nested with other components to allow complex applications to be built of simple building blocks. ReactJS uses virtual DOM based mechanism to fill data in HTML DOM. The virtual DOM works fast as it only changes individual DOM elements instead of reloading complete DOM every time.

To create React app, we write React components that correspond to various elements. We organize these components inside higher level components which define the application structure. For example, we take a form that consists of many elements like input fields, labels, or buttons. We can write each element of the form as React components, and then we combine it into a higher-level component, i.e., the form component itself. The form components would specify the structure of the form along with elements inside of it.

React create-react-app

Starting a new React project is very complicated, with so many build tools. It uses many dependencies, configuration files, and other requirements such as Babel, Webpack, ESLint before writing a single line of React code. Create React App CLI tool removes all that complexities and makes React app simple. For this, you need to install the package using NPM, and then run a few simple commands to get a new React project.

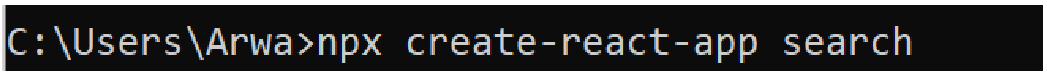
To create a React Project using create-react-app, you need to have installed the following things in your system.

- Node version >= 8.10
- NPM version >= 5.6

We can install React using npm package manager by using the following command. There is no need to worry about the complexity of React installation. The create-react-app npm package manager will manage everything, which needed for React project.

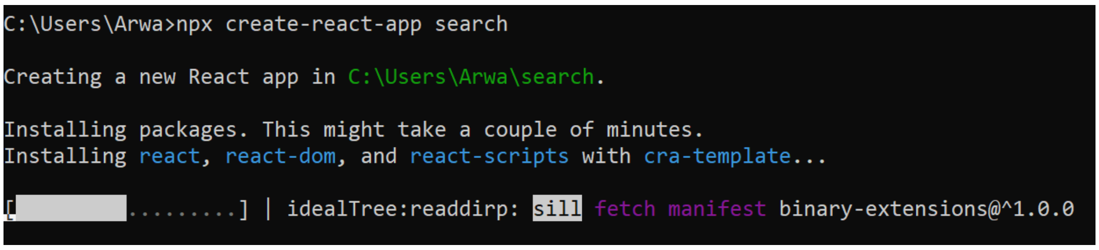
C:\Users\Arwa> npm install -g create-react-app

Once the React installation is successful, we can create a new React project using create-react-app command.



```
C:\Users\Arwa>npx create-react-app search
```

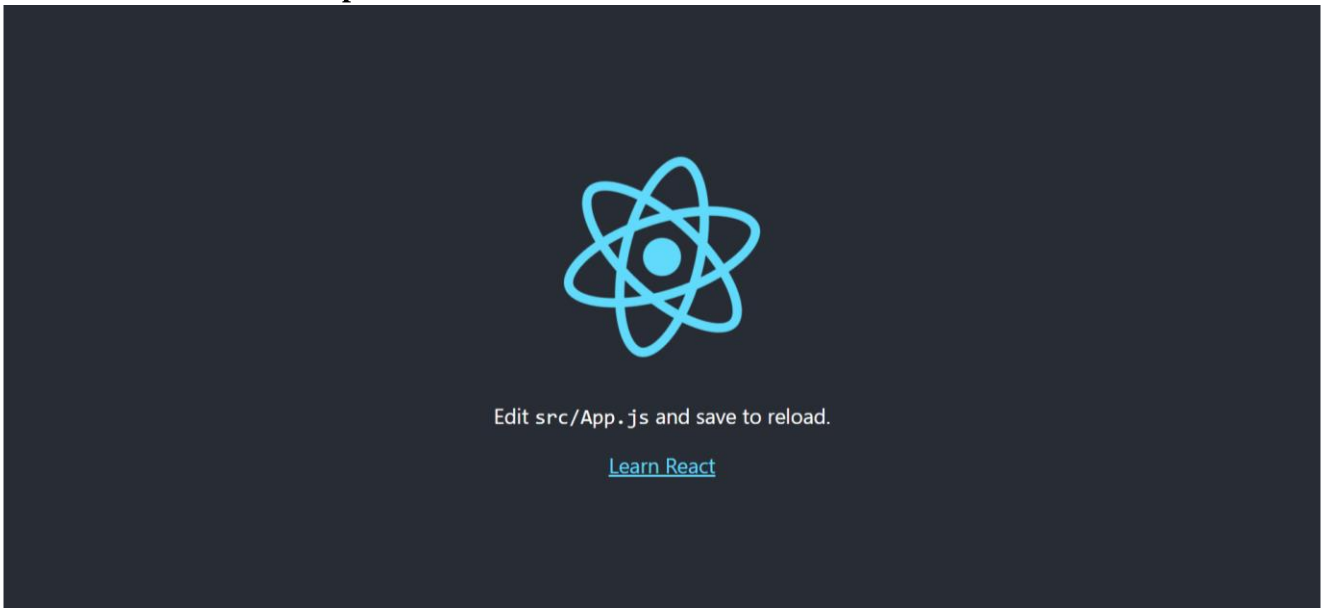
The above command will take some time to install the React and create a new project with the name "search." Now, we can see the terminal as like below.



```
C:\Users\Arwa>npx create-react-app search

Creating a new React app in C:\Users\Arwa\search.

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

[█████████.........] | idealTree:readdirp: sill fetch manifest binary-extensions@^1.0.0
```

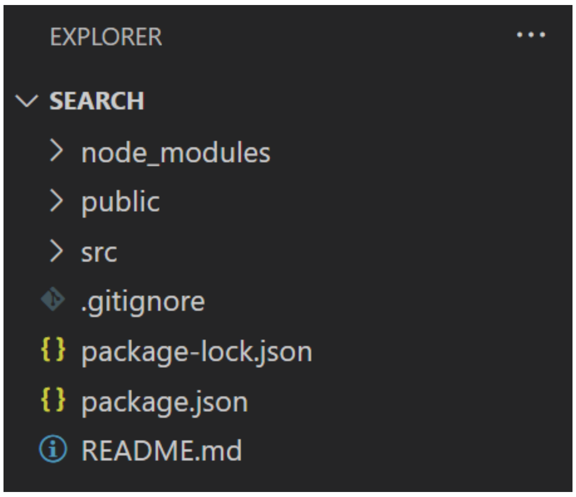
The above screen tells that the React project is created successfully on our system. Now, we need to start the server so that we can access the application on the browser. Type the following command in the terminal window.

C:\Users\Arwa>cd search
C:\Users\Arwa\search>npm start



Edit src/App.js and save to reload.

Learn React



```
EXPLORER ...
∨ SEARCH
  > node_modules
  > public
  > src
    .gitignore
  {} package-lock.json
  {} package.json
  ⓘ README.md
```

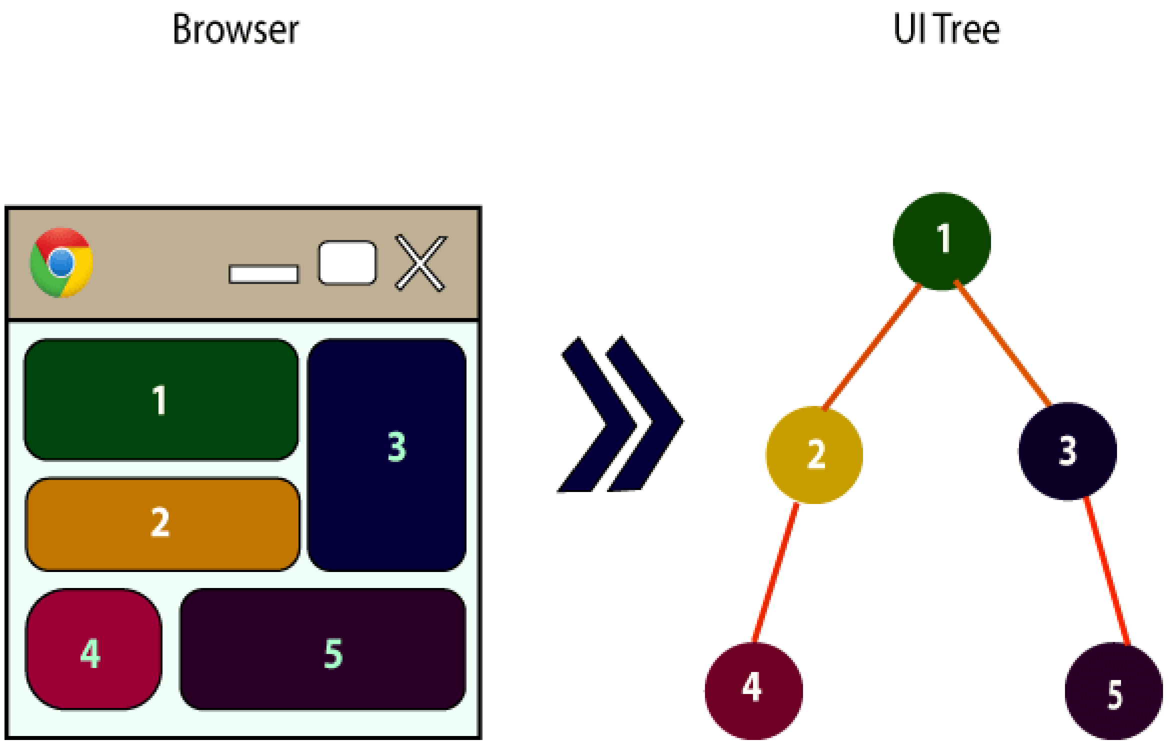
In React application, there are several files and folders in the root directory. Some of them are as follows:

- **node_modules:** It contains the React library and any other third-party libraries needed.
- **public:** It holds the public assets of the application. It contains the index.html where React will mount the application by default on the <div id="root"></div> element.

- **src:** It contains the App.css, App.js, App.test.js, index.css, index.js, and serviceWorker.js files. Here, the App.js file always responsible for displaying the output screen in React.
- **package-lock. json:** It is generated automatically for any operations where npm package modifies either the node_modules tree or package. json. It cannot be published. It will be ignored if it finds any other place rather than the top-level package.
- **package. json:** It holds various metadata required for the project. It gives information to npm, which allows to identify the project as well as handle the projects dependencies.
- **README.md:** It provides the documentation to read about React topics.

React Components

Earlier, the developers write more than thousands of lines of code for developing a single page application. These applications follow the traditional DOM structure, and making changes in them was a very challenging task. If any mistake found, it manually searches the entire application and update accordingly. The component-based approach was introduced to overcome an issue. In this approach, the entire application is divided into a small logical group of code, which is known as components.
A Component is considered as the core building blocks of a React application. It makes the task of building UIs much easier. Each component exists in the same space, but they work independently from one another and merge all in a parent component, which will be the final UI of your application.
Every React component have their own structure, methods as well as APIs. They can be reusable as per your need. For better understanding, consider the entire UI as a tree. Here, the root is the starting component, and each of the other pieces becomes branches, which are further divided into sub-branches.



React State

The state is an updatable structure that is used to contain data or information about the component. The state in a component can change over time. The change in state over time can happen as a response to user action or system event. A component with the state is known as stateful components. It is the heart of the react component which determines the behavior of the component and how it will render. They are also responsible for making a component dynamic and interactive.

A state must be kept as simple as possible. It can be set by using the setState() method and calling setState() method triggers UI updates. A state represents the component's local state or information. It can only be accessed or modified inside the component or by the component directly. To set an initial state before any interaction occurs, we need to use the getInitialState() method.
For example, if we have five components that need data or information from the state, then we need to create one container component that will keep the state for all of them.

Defining State

To define a state, you have to first declare a default set of values for defining the component's initial state. To do this, add a class constructor which assigns an initial state using this.state. The 'this.state' property can be rendered inside render() method.

React Props

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.
Props are immutable so we cannot modify the props from inside the component. Inside the components, we can add attributes called props. These attributes are available in the component as this.props and can be used to render dynamic data in our render method.
When you need immutable data in the component, you have to add props to reactDom.render() method in the main.js file of your ReactJS project and used it inside the component in which you need.

Install Material-UI's source files via npm. We take care of injecting the CSS needed.

```
$ npm install @material-ui/core
```

Code:

Search.js

```
import React,{Component} from 'react';
import axios from 'axios';
import ImageResults from "../imageResults/imageResults";

class Search extends Component
{
   state={
      searchText:'',
      apiUrl:'https://pixabay.com/api',
      apiKey:'12733053-0141b92b419134a72765749a2',
      images:[]
   }
   onTextChange=e=>{
      this.setState({[e.target.name]:e.target.value},()=>{
         axios
         .get(

`${this.state.apiUrl}/?key=${this.state.apiKey}&q=${this.state.searchText}&image_type=photo&safe
search=true`
         )
         .then(res=>this.setState({images:res.data.hits}))
```

```
            .catch(err=>console.log(err));
        })
    }
    render()
    {
        console.log(this.state.images);
        return(
            <div>
            <input type="text" style=
            {{backgroundColor:'black',
             color : 'white',
             marginLeft:570,
             marginTop:100,
             paddingTop:20,
             paddingLeft:70,
             fontSize:30,
             borderTopStyle:"hidden",
             borderLeftStyle:"hidden",
             borderRightStyle:"hidden",
             outline:"none",
             borderBottomStyle:"groove",
             }}
             placeholder="Search for images"
             name="searchText"
             value={this.state.searchText}
             onChange={this.onTextChange}
             ></input>
             {this.state.images.length>0?(<ImageResults images={this.state.images}/>):null}
        </div>
        )
    }
}

export default Search;
```

imageResult.js

```
import React,{Component} from 'react';
import PropTypes from 'prop-types';
import { GridList } from '@material-ui/core';
import { GridListTile } from '@material-ui/core';

class ImageResults extends Component{

    state = {
        open:false,
        currentImg:''
    }
    render()
    {
        let imageList;
        const {images}= this.props

        if(images)
        {
            imageList=(
                <GridList cols={4}>
                    {
                        images.map(img=>(
                            <GridListTile
                                title={img.tags}
                                key = {img.id}
                            >
                            <img src={img.largeImageURL} alt="" />
                            </GridListTile>
                        ))
                    }

                </GridList>
            )

        }
        else
        {
            imageList=null;
        }

        return(
            <div style={{marginLeft:50, marginRight:50, marginTop:90}}>
                {imageList}

                <img src={this.state.currentImg} alt="" style={{width:'100%'}}></img>
            </div>
        )
    }
}
ImageResults.propTypes=
{
    images:PropTypes.array.isRequired
}

export default ImageResults;
```

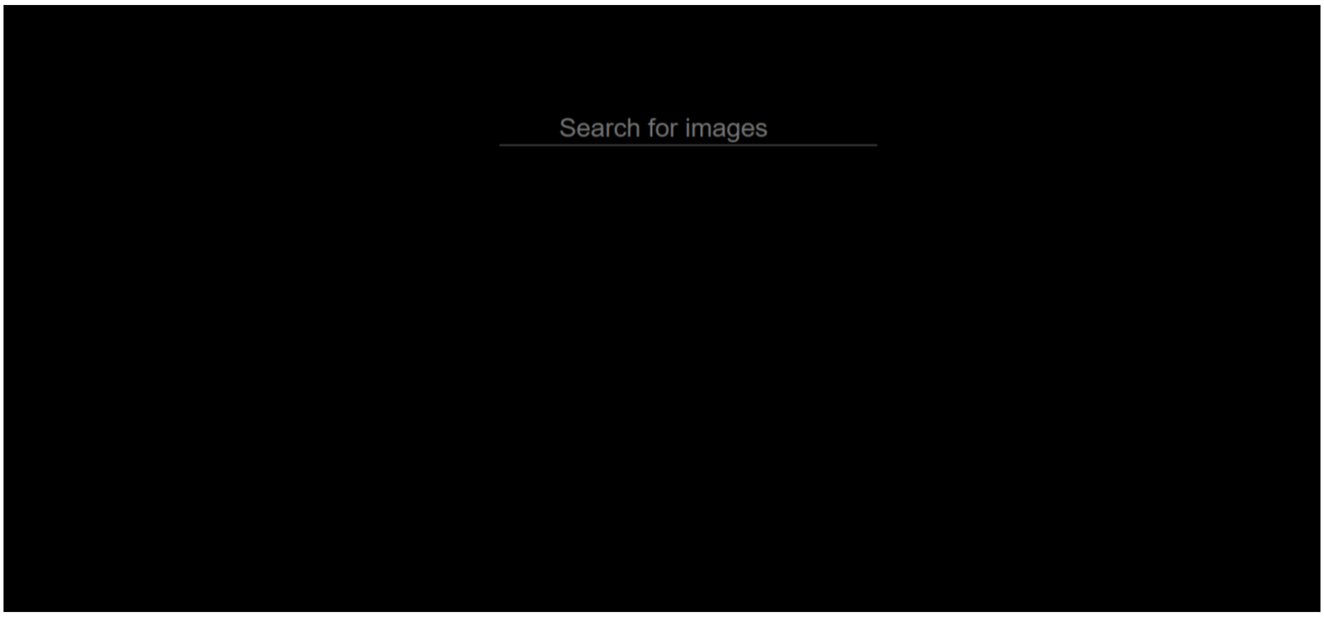
App.css

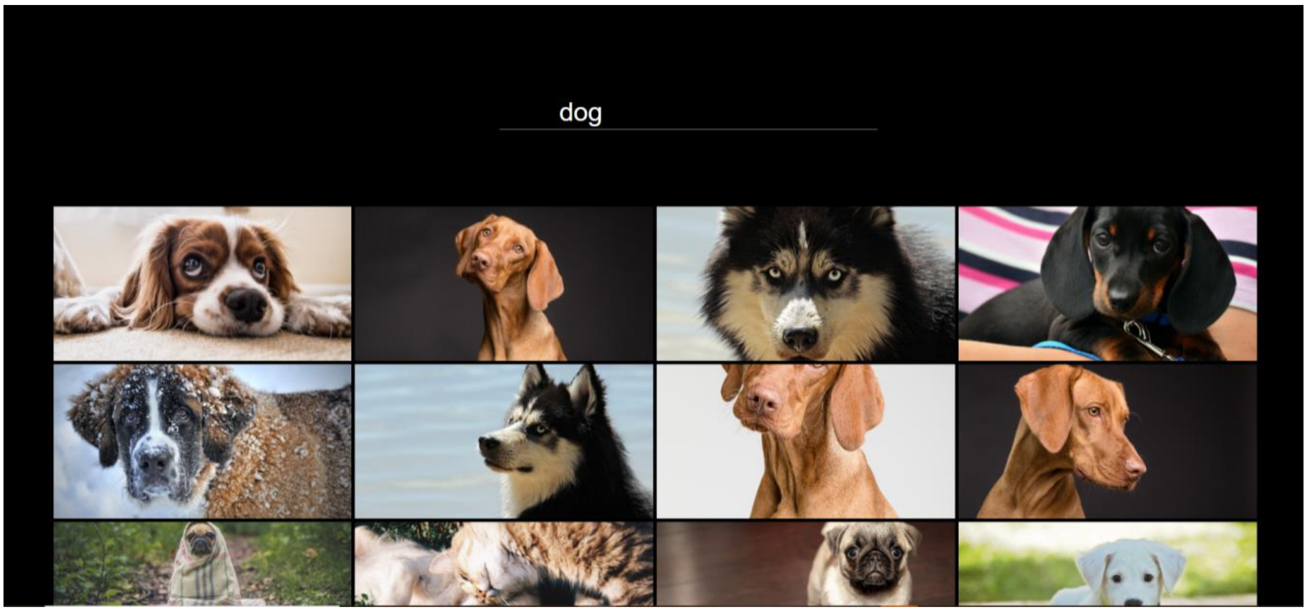
```
body{
  background-color: black;
}
```

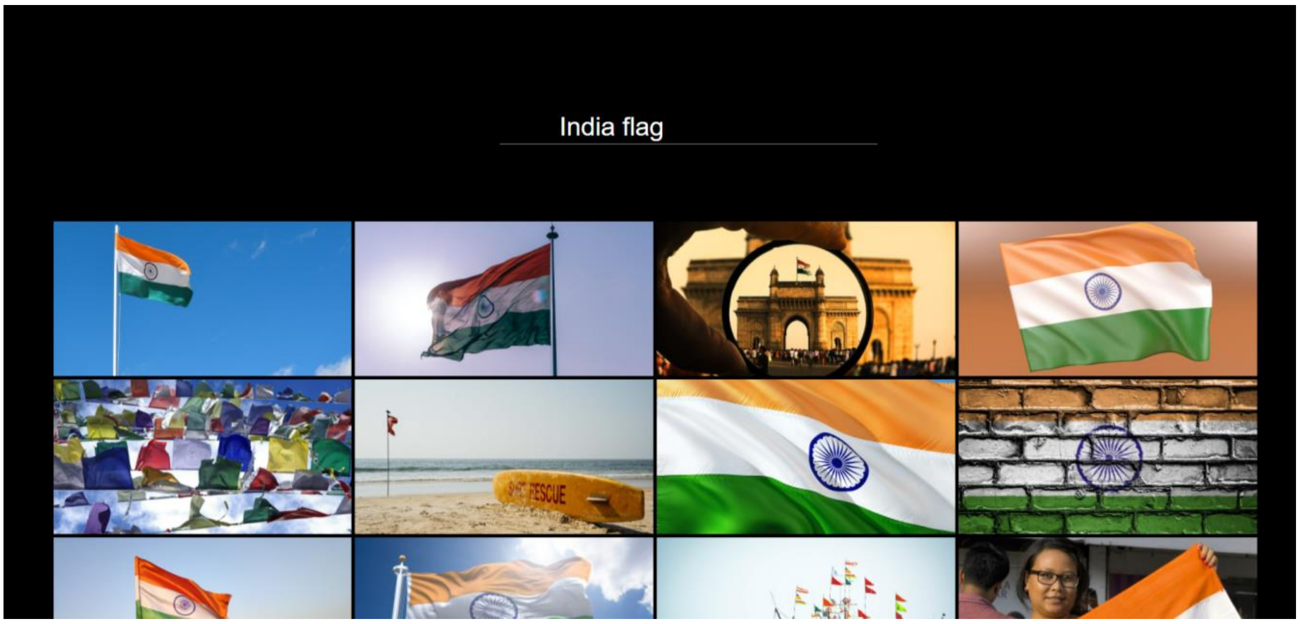
App.js

```
import React, { Component } from 'react';
import Search from "./components/search/Search";
import './App.css';
import { MuiThemeProvider, createMuiTheme } from '@material-ui/core/styles'; // v1.x

class App extends Component
{
  render()
  {
    return(
      <MuiThemeProvider>
        <div>
        <Search />
      </div>
      </MuiThemeProvider>
    )
  }
}

export default App;
```

OUTPUT




dog


India flag