Experiment No. 5

Aim :- Design a web page using Bootstrap's Components Grid system forms, Button, Navbar, Bread Crumb, Jumbotron

Theory :→ Bootstrap is a popular open-source frontend-framework that helps developers create responsive, mobile first websites quickly & easily. it provide a collection of CSS & JS components.

Components in Bootstrap :-

1) Grid system :- It is a core layout structure. it allows you to create complex layout through a series of rows & columns. it is based on a 12-column layout, where you can define how many col a particular content element should occupy.

2) Forms :- It provide a set of components for creating forms. They are used for collecting user input & can include a variety of elements like text fields, radiobutton & submit buttons.

3) Buttons :→ They are styled using pre-defined classes which make it easy to create different type of buttons. You can create primary secondary, succes, wraning buttons & more.

(4) Navbar :- Navbar is a responsive navigation header that include support for branding navigation & more. it can adopt to various screen size, ensuring that your site's nav is accessible.

(5) Breadcrumb :- They are a navigation aid that shows the user's current location within website's hierarchy. it provides links back to each previous page the user navigated through.

(6) Jumbotron :- It is a large attention grabbing component used to showcase key content or information. it is typically used at the top of a page to highlight important messages or calls to action.

Conclusion :- Hence, we have successfully implemented a webpage design using Bootstrap components.

Experiment No-6

Aim:→ Write a Program which demonstrate state & props in react.

Theory:→ In react state & props are two core concepts that allow components to manage and share data.

① **State:→** State is mutable and can change over time

→ It is used to store data that can change within a component.
→ A change in state causes the component to re-render.
→ Unlike props, state are managed within the component.

ex→
```
import React, {useState} from 'react'
function component () {
    const [statevariable, setstatevariable] =
                    usestate(initialvalue);
    const updatestate = () => {
        setstatevariable(newValue);
    };
    return(
        <button onclick={updatestate}>update
            state </button>
    );
}
```

props :→ props are read-only and immutable.

→ They allow passing data from a parent component to child components.

→ The child component cannot modify props

→ props are passed into component like attributes in HTML.

ex→ //Parent component.js

```
<Child component
    propName = {propsValue} />
```

```
// Child Componect.JS
function childComponent (props){
    return <div> {props.propName}</
    div>;
}
```

Conclusion :→ Hence, We have sucessfully implemented a program to demonstrate state & props in React.

Experiment No-7

Aim :→ Design a registration form using form element in React Js along with appropiate validations

Theory :→ Following are the React Js form element & their valid-actions.

① <input> :→ This is the most versatile form element used for various input types like text, email, password etc.

- Validations :→ input feild involves checking the type of input ensuring the field is not empty & matching a specific pattern if necessary.

Example → validating a text field could involve ensuring that it Con-tains no special characters.

② <textarea> : it is a similar to the input element but used for longer text entries such as description or comments.

Validations :→
it can include ensuring the text area is not empty. checking for character limits, or restricting the use of ocrtain characters.

③ '<select>' :- This dropdown form element. allows user to select from a predefined list of options.

④ '<radiobox>' :→ Radio buttons in form allow user to select only one option from a group.

validations :→ it involves ensuring that atleast one radio button is selected in a group.

→ it is important where a choice between multiple exclusive options is required.

⑤ '<button>' :→ A button element is typically used to submit the form. while it does not hold data, it plays a crucial role in trigering form validation before submission if any form validation rules are violated, the form can be prevented from submitting.

⑥ '<fieldset>' : This element group multiple form controls togethers such as grouping related checkboxes or radio buttons

⑦ '<tables>' :→ Lables are used to associate text discriptions with input element. Though not directly involved in validation. labled ensure users understand the purpose of each field.

VASANTDADA PATIL PRATISHTHAN'S
COLLEGE OF ENGINEERING & VISUAL ARTS

Page No.

(6) <datalist>:→ This element provides a list of pre-defined option for an <input> field, offering suggestion as the user types.

Conclusion:→ Hence we have successfully implemented a registration form using form elements in React JS along with appropriate validations.

Experiment No-8

Aim :- Write a program which demonstrates React Router & single page Application.

Theory :→

ReactRouter :→ React Router is a powerful library for routing in React application. in single page Application (SPA), there is only one HTML page, and routing allows navigation b/w different "pages" without refreshing the browser. This results in faster navigation and a more seamless user experience.

React Router enables :-

- Client-side routing to different views (components)
- Declarative routing in JSX using the Route component.
- URL parameter, navigation, and dynamic rendering of components based on the route.

It is typically SPA, components are rendered based on the current URL, but the browser doesn't make a new Request to the server; instead it loads the new view dynamically using Javascript.

React Router enables building SPAs by allowing clint side navigation.

→ Syntax involves wrapping the app with BrowserRouter and defining Route components inside Routes.

→ Dynamic Routing is possible with useParams, and programmatic navigation can be done with UseNaviget

Syntax ex→

```
import { BrowserRouter as Router, Routes, Route,
                                    Link } from
                    "react-router-dom";
function App() {
  return(
      <Router>
        <nav>
          <ul>
            <li><Link to="/">Home
            </Link> </li>
            <li><Link to=""> About
                  </Link> </li>
        <Routes>
          <Route path="/" element=
                  {<Home/>} />
          <Route path="/" element=
            {<About/>} />
        </Routes>
      </Router>
  );
}
```

```
function Home() {
    return <h2> Home page </h2>
}
function About() {
    return <h2> About page </h2>
}
```

Conclusion :→ Hence, we have susscefully implemented a program to demonstrate React Router and Single page Application (SPA).

Experiment No-9

Aim:- Write a program which demonstrates Hooks (useEffect) refs in React JS.

Theory :- Hooks in React are special function that let you "hook into" React state and lifecycle features within function components. They were introduced in React 16.8 and provide a way to use state, lifecycle methods, and other React features without needing class component.

useEffect :- This hook lets you perform side effect in function components. it run after the Component renders and can be use to handle things like data fetching, subscription, and manually changing the DOM. you can also control when the effect runs by passing dependencies.

useref :- This hooks allow you to persist values across renders persist values across render without causing re-renders. it typically used to access, DOM elements directly, store mutable values, and keep values that don't tigger a re-render when updated.

Syntax:→

```
useEffect(() => {
    console.log('component rendered');

    return() => {
        console.log("Hello");
    };
}, [count];
```

(ii) Use ref

Syntax →

```
const myRef =
    useRef(initialvalue);

myRef.current;
```

Conclusion:→ Hence, we have succesfully implemented a program that demostraff Hooks useEffect / ref in React.