Don Bosco Institute of Technology
Kurla (W), Mumbai 400 070

Internet Programming

Name: Parth Pravin Shikhare

Class: TE-IT

Roll No.: 53

Date: 26/09/25

Experiment No.: 9

Title:

To Design a web page using ReactJS Hooks

Problem Definition:

Create a web page using ReactJS implementing Hooks with functionalities involving ReactJS Forms, Events, Routers, Refs, and Keys.

Pre-requisites:

1. Basic knowledge of ReactJS components, JSX, and props
2. Understanding of functional components in React
3. Node.js and npm installed for React environment
4. Familiarity with ES6 features and basic DOM manipulation
5. Basic knowledge of React Router and React Hooks

Theory:

React Hooks are special functions that let you 'hook into' React features such as state and lifecycle methods from function components. Hooks allow writing stateful logic without using classes. Common hooks include useState, useEffect, useRef, useContext, and useReducer.

• useState: Manages state in functional components.
• useEffect: Handles side effects such as fetching data, subscriptions, or manual DOM manipulations.
• useRef: Creates a reference to DOM elements or mutable variables.
• useContext: Accesses context values directly without using props.
• useReducer: Manages complex state logic similar to Redux.

React Router allows for navigation between different components or pages. Refs are used to directly access DOM nodes, while Keys help React identify elements during rendering for optimized updates.

Procedure:

1. Install Node.js and verify npm installation.
2. Create a new React project using Create React App.
3. Install React Router DOM using npm.
4. Create different components for pages and navigation.
5. Implement useState and useEffect for managing state and side effects.
6. Use useRef to access form elements directly.
7. Implement form validation and event handling.
8. Use Router, Route, and Link for navigation between components.
9. Test the web page in a browser to ensure correct functionality.
10. Document the output and results.

Program:

About.jsx

```
export default function About() {
  return (
    <div className="p-6">
      <h2 className="text-2xl font-semibold">About</h2>
      <p className="mt-2 text-gray-700">
        Small demo app to teach common React patterns.
      </p>
    </div>
  );
}
```

Nav.jsx

```
import { Link } from "react-router-dom";

export default function Nav() {
  return (
    <nav className="p-4 bg-gradient-to-r from-sky-500 to-indigo-600 text-white flex gap-
4">
      <Link to="/" className="font-semibold hover:underline">Home</Link>
      <Link to="/form" className="font-semibold hover:underline">Form</Link>
      <Link to="/list" className="font-semibold hover:underline">List (Keys)</Link>
      <Link to="/refs" className="font-semibold hover:underline">Refs</Link>
      <Link to="/about" className="font-semibold hover:underline">About</Link>
    </nav>
  );
}
```

Home.jsx

```
export default function Home() {
 return (
  <div className="p-6">
   <h1 className="text-3xl font-bold mb-2">React Hooks & Features Demo</h1>
   <p className="text-gray-700">
    This sample app demonstrates hooks, forms, events, router, refs, and list keys.
   </p>
  </div>
 );
}
```

FormPage.jsx

```
import { useState } from "react";
import { useNavigate } from "react-router-dom";

export default function FormPage() {
 const [form, setForm] = useState({ name: "", email: "", age: "" });
 const [errors, setErrors] = useState({});
 const [submittedData, setSubmittedData] = useState(null);
 const navigate = useNavigate();

 function validate(values) {
  const e = {};
  if (!values.name.trim()) e.name = "Name is required";
  if (!values.email.includes("@")) e.email = "Must be a valid email";
  const ageNum = Number(values.age);
  if (!values.age || Number.isNaN(ageNum) || ageNum <= 0)
   e.age = "Enter a valid age";
  return e;
 }

 function handleChange(e) {
  const { name, value } = e.target;
  setForm((prev) => ({ ...prev, [name]: value }));
 }

 function handleSubmit(e) {
  e.preventDefault();
  const v = validate(form);
  setErrors(v);
  if (Object.keys(v).length === 0) {
   setSubmittedData(form);
   setTimeout(() => navigate("/list"), 500);
  }
 }
```

```
function handleReset() {
  setForm({ name: "", email: "", age: "" });
  setErrors({});
  setSubmittedData(null);
}

return (
  <div className="p-6 max-w-xl">
    <h2 className="text-2xl font-semibold mb-4">Controlled Form (Hooks + Events)</h2>
    <form onSubmit={handleSubmit} onReset={handleReset} className="space-y-4">
      <div>
        <label className="block font-medium">Name</label>
        <input
          name="name"
          value={form.name}
          onChange={handleChange}
          className="mt-1 p-2 border rounded w-full"
          placeholder="Your name"
        />
        {errors.name && <div className="text-red-600 mt-1">{errors.name}</div>}
      </div>

      <div>
        <label className="block font-medium">Email</label>
        <input
          name="email"
          value={form.email}
          onChange={handleChange}
          className="mt-1 p-2 border rounded w-full"
          placeholder="you@example.com"
        />
        {errors.email && <div className="text-red-600 mt-1">{errors.email}</div>}
      </div>

      <div>
        <label className="block font-medium">Age</label>
        <input
          name="age"
          value={form.age}
          onChange={handleChange}
          className="mt-1 p-2 border rounded w-full"
          placeholder="Age"
          type="number"
        />
        {errors.age && <div className="text-red-600 mt-1">{errors.age}</div>}
      </div>

      <div className="flex gap-2">
        <button type="submit" className="px-4 py-2 bg-indigo-600 text-white rounded">
          Submit
```

```
        </button>
        <button type="reset" className="px-4 py-2 bg-gray-200 rounded">
          Reset
        </button>
      </div>
    </form>

    {submittedData && (
      <div className="mt-6 p-4 border rounded bg-green-50">
        <h3 className="font-semibold">Submitted</h3>
        <pre className="text-sm">{JSON.stringify(submittedData, null, 2)}</pre>
      </div>
    )}
  </div>
 );
}
```

ListPage.jsx

```
import { useState } from "react";
import { Link, Routes, Route } from "react-router-dom";
import ListItemDetail from "./ListItemDetail";

let idCounter = 2;

export default function ListPage() {
 const [items, setItems] = useState([
  { id: 1, text: "Learn React" },
  { id: 2, text: "Build Projects" },
 ]);
 const [text, setText] = useState("");

 function addItem() {
  if (!text.trim()) return;
  setItems([{ id: ++idCounter, text }, ...items]);
  setText("");
 }

 function removeItem(id) {
  setItems(items.filter((i) => i.id !== id));
 }

 return (
  <div className="p-6 max-w-2xl">
   <h2 className="text-2xl font-semibold mb-4">List & Keys</h2>
   <div className="flex gap-2">
    <input
     value={text}
     onChange={(e) => setText(e.target.value)}
     className="p-2 border rounded flex-1"
```

```
        placeholder="Add new item"
       />
       <button onClick={addItem} className="px-4 py-2 bg-sky-600 text-white rounded">
        Add
       </button>
      </div>

      <ul className="mt-4 space-y-2">
       {items.map((item) => (
        <li key={item.id} className="flex justify-between items-center p-3 border rounded">
         <Link to={`/list/${item.id}`} className="font-medium hover:underline">
          {item.text}
         </Link>
         <button
          onClick={() => removeItem(item.id)}
          className="px-2 py-1 bg-red-500 text-white rounded"
         >
          Remove
         </button>
        </li>
       ))}
      </ul>

      <Routes>
       <Route path=":id" element={<ListItemDetail items={items} />} />
      </Routes>
    </div>
  );
}
```

ListItemDetail.jsx

```
import { useParams } from "react-router-dom";

export default function ListItemDetail({ items }) {
  const { id } = useParams();
  const item = items.find((it) => String(it.id) === id);

  if (!item) return <div className="p-4">Item not found.</div>;

  return (
    <div className="mt-4 p-4 border rounded bg-gray-50">
      <h3 className="font-semibold">Item detail</h3>
      <p>ID: {item.id}</p>
      <p>Text: {item.text}</p>
    </div>
  );
}
```

RefsPage.jsx

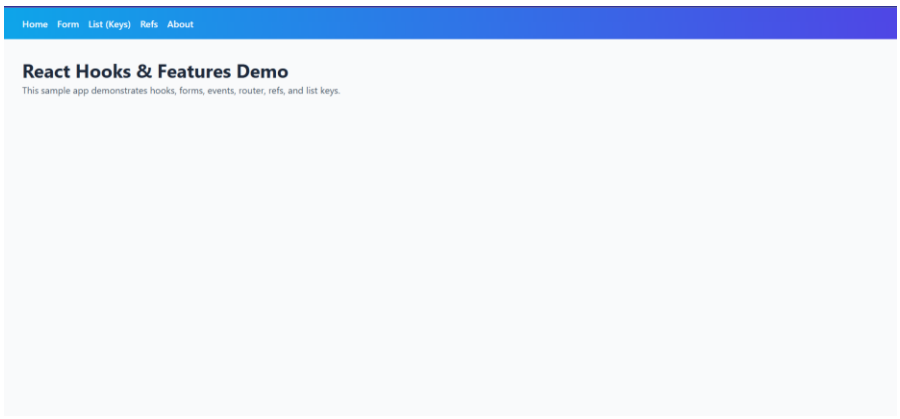
```
import { useRef, useState, useEffect } from "react";

export default function RefsPage() {
 const inputRef = useRef(null);
 const [value, setValue] = useState("");
 const [log, setLog] = useState([]);

 useEffect(() => {
  inputRef.current?.focus();
 }, []);

 function handleRead() {
  const v = inputRef.current?.value || "";
  setLog((prev) => [`Read via ref: "${v}"`, ...prev].slice(0, 10));
 }

 function handleUpdateState() {
  setValue(inputRef.current?.value || "");
 }

 return (
  <div className="p-6 max-w-xl">
   <h2 className="text-2xl font-semibold mb-4">Refs Demo</h2>
   <input
    ref={inputRef}
    className="p-2 border rounded w-full"
    placeholder="Type something..."
    defaultValue={value}
   />
   <div className="mt-3 flex gap-2">
    <button onClick={handleRead} className="px-4 py-2 bg-indigo-600 text-white
rounded">
     Read via ref
    </button>
    <button onClick={handleUpdateState} className="px-4 py-2 bg-gray-200 rounded">
     Sync to state
    </button>
   </div>

   <div className="mt-4">
    <h4 className="font-medium">State value</h4>
    <div className="p-2 border rounded mt-1">{value || <span className="text-gray-
400">(empty)</span>}</div>
   </div>

   <div className="mt-4">
    <h4 className="font-medium">Action log</h4>
```

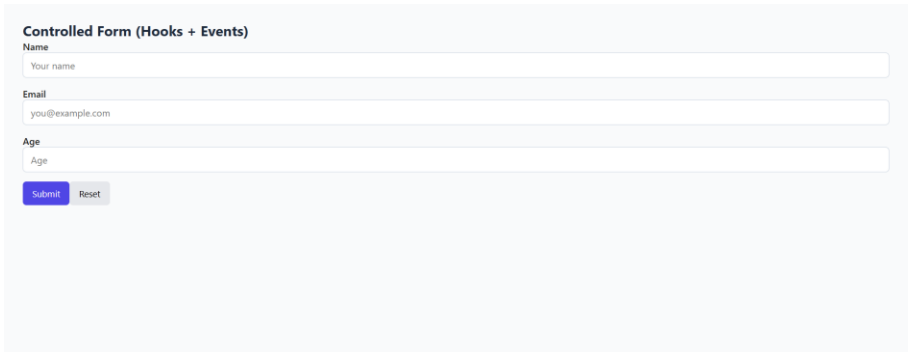
```
      <ul className="mt-2 list-disc pl-5 space-y-1">
       {log.map((l, i) => (
        <li key={i}>{l}</li>
       ))}
      </ul>
     </div>
    </div>
  );
}
```

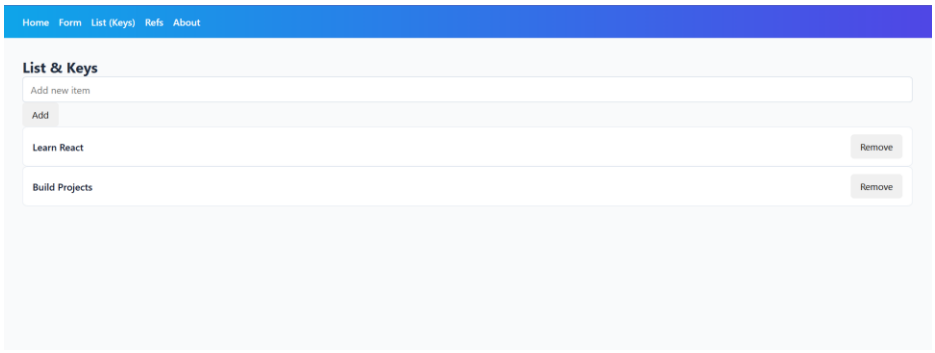
App.jsx

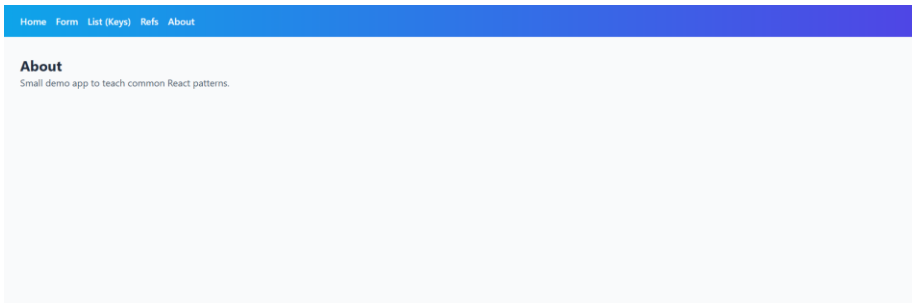
```
import { BrowserRouter as Router, Routes, Route } from "react-router-dom";
import Nav from "./components/Nav";
import Home from "./components/Home";
import About from "./components/About";
import FormPage from "./components/FormPage";
import ListPage from "./components/ListPage";
import RefsPage from "./components/RefsPage";

export default function App() {
  return (
    <Router>
     <div className="min-h-screen bg-gray-50">
      <Nav />
      <main>
       <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/form" element={<FormPage />} />
        <Route path="/list/*" element={<ListPage />} />
        <Route path="/refs" element={<RefsPage />} />
        <Route path="/about" element={<About />} />
        <Route path="*" element={<div className="p-6">Page not found</div>} />
       </Routes>
      </main>
     </div>
    </Router>
  );
}
```

Output:


Home Form List (Keys) Refs About
React Hooks & Features Demo
This sample app demonstrates hooks, forms, events, router, refs, and list keys.


Controlled Form (Hooks + Events)
Name
Your name
Email
you@example.com
Age
Age
Submit
Reset


Home Form List (Keys) Refs About
List & Keys
Add new item
Add
Learn React
Remove
Build Projects
Remove



Result:

Successfully designed and implemented a web page using ReactJS Hooks, including Forms, Events, Routers, Refs, and Keys.

References:

1. https://react.dev/
2. https://reactrouter.com/
3. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference