LAB EXPERIMENT-3

ID:2200030222

NAME:M.Jashrutha

3. Experiment Title: Working with React-Router

Aim/Objective:

The aim of React Router is to provide a routing solution for single-page applications built with React. It allows developers to implement navigation and routing functionality in their React applications, enabling users to move between different views or components based on the URL.

Description:

React Router is a popular library in the React ecosystem that enables developers to implement routing functionality in their React applications. It allows for navigation and rendering of different components based on the URL, enabling users to move between different views or pages without requiring a full page reload. React Router provides a set of components and APIs that help in defining routes, handling navigation, and managing the history of the application.

Pre-Requisites:

React Router primarily deals with routing and navigation, having a basic understanding of HTML and CSS is valuable for building user interfaces and styling your React components.

Pre-Lab:

1. What is React Router, and what problem does it solve in React applications?

React Router is a standard library for routing in React. It enables the navigation among views of various components in a React Application, allows changing the browser URL, and keeps the UI in sync with the URL.

2. What is the purpose of the <Switch> component in React Router? How does it work?

The <Switch> component is used to render components only when the path will be matched. Otherwise, it returns to the not found component.

3. How can you pass parameters to a route in React Router? Explain how you can access these parameters in the component.

Passing parameters to routes:

1. Pass params to a route by putting them in an object as a second parameter to the navigation. navigate function: navigation. navigate('RouteName', { /* params go here */ })
2. Read the params in your screen component: route. params .

4. What is nested routing in React Router? Provide an example of how you can create nested routes.

We will continue working on the User component, because this is the place where we want to have the nested routing via tabs. Therefore, we will instantiate a new set of Link components (which will be our unstyled tabs) which navigate a user to their profile and their account.

5. How does React Router handle browser history and navigation? Explain the difference between <BrowserRouter> and <HashRouter>.

Both BrowserRouter and HashRouter components were introduced in React Router ver. 4 as subclasses of Router class. Simply, BrowserRouter syncs the UI with the current URL in your browser, This is done by the means of HTML-5 History API. On the other hand, HashRouter uses the Hash part of your URL to sync.

In-Lab:

Exercise 1: Create a simple React application with two routes: Home and About. Implement navigation links to switch between these routes using React Router.

Exercise 2: Implement a nested routing structure in a React application using React Router. Create a parent route and two child routes that are rendered within the parent component.

□ Procedure/Program:

1

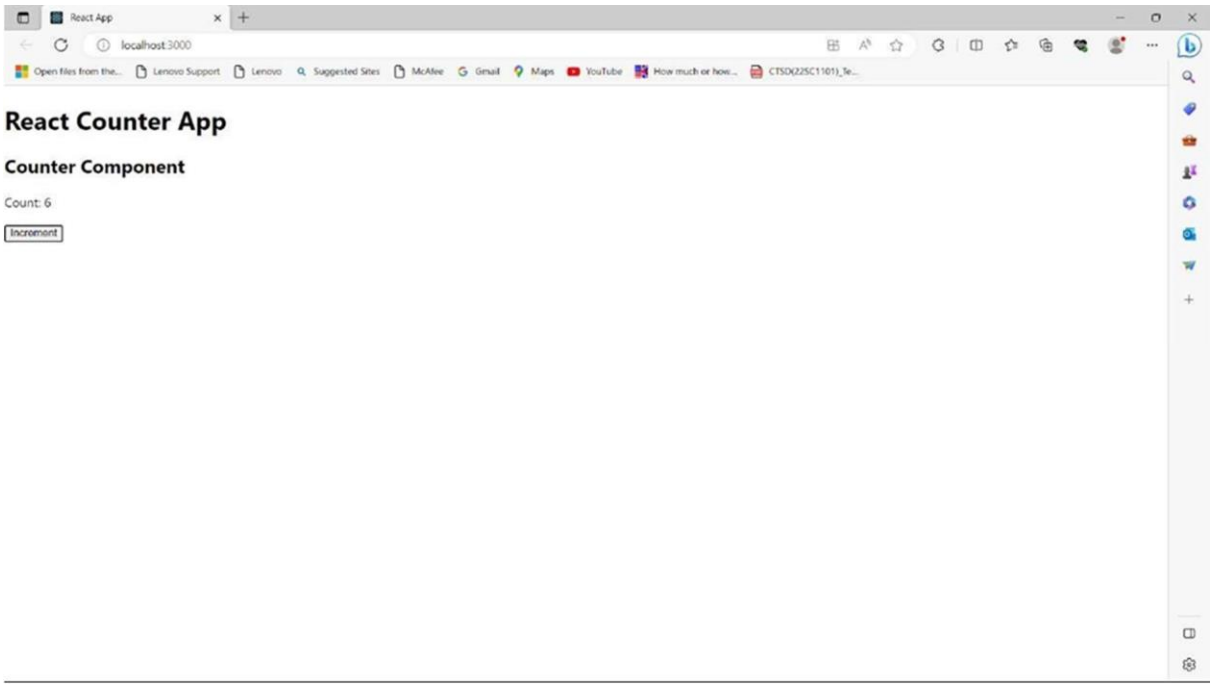
```
//app.js
import React from 'react'; I
mport Counter from './Counter';

const App = () => {
return (
<div>
<h1>React Counter App</h1>
<Counter initialCount={0} />
</div>
);
};

export default App;

//counter.js
import React, { useState } from 'react';

const Counter = ({ initialCount }) => {
const [count, setCount] = useState(initialCount);

const handleIncrement = () => { setCount(count + 1);
};

return (
<div>
<h2>Counter Component</h2>
<p>Count: {count}</p>
<button onClick={handleIncrement}>Increment</button>
</div>
);
};

export default Counter;
```

2

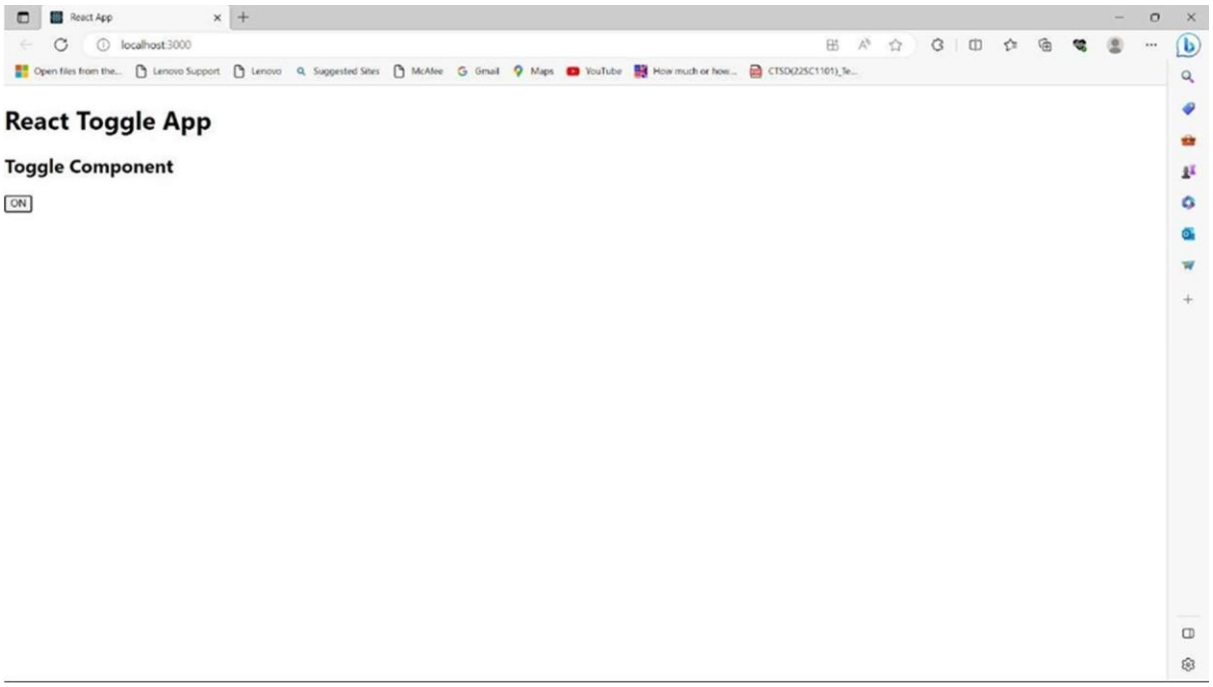
```
// app.js
import React from 'react';
import Toggle from './Toggle';

const App = () => { return (
<div>
<h1>React Toggle App</h1>
<Toggle />
</div>
);
};

export default App;

//toggle.js
import React, { useState } from 'react';

const Toggle = () => {
const [isOn, setIsOn] = useState(false);

const handleToggle = () => { setIsOn((prevIsOn) => !prevIsOn);
};

return (
<div>
<h2>Toggle Component</h2>
<button onClick={handleToggle}>
{isOn ? "ON" : "OFF"}
</button>
</div>
);
};

export default Toggle;
```

□ Data and Results:

1



2



Sample VIVA-VOCE Questions (In-Lab):

1. How can you pass parameters to a route using React Router?

React Router, you can pass parameters to a route using the URL path. These parameters are known as "URL parameters" or "route parameters." React Router provides a way to define placeholders in the route's path, and these placeholders will be used to extract the parameters from the URL

2. How can you handle 404 errors or create a fallback route using React Router?

To handle 404 errors or create a fallback route (also known as a "404 Not Found" page) using React Router, you can define a catch-all route at the end of your route configuration. This route will match any URL that doesn't match any of the previously defined routes, allowing you to handle cases where the user accesses a non-existent page.

3. How can you navigate between routes programmatically using React Router?

In React Router, you can navigate between routes programmatically using the history object provided by React Router. The history object allows you to manipulate the browser history and change the current URL, which triggers the rendering of the corresponding route.

4. How can you protect routes and implement authentication/authorization using React Router?

To protect routes and implement authentication/authorization using React Router, you can follow these general steps:

1. Set Up Authentication: Implement authentication in your application using a login system, such as username/password login or OAuth-based login (e.g., using third-party providers like Google, Facebook, etc.). When a user logs in successfully, you'll typically receive an authentication token or session data from the server.

2. Create a ProtectedRoute Component: Build a custom ProtectedRoute component that will act as a wrapper around the routes that require authentication. This component will check if the user is authenticated and redirect them to the login page if they are not.

5. What are the different types of routers provided by React Router?

React Router provides three types of routers:

- BrowserRouter: This router uses HTML5 history API to handle navigation. It provides clean URLs without the need for hash fragments (#) in the URLs. It is the most commonly used router for web applications when the server is configured to handle dynamic URLs.
- HashRouter: This router uses the hash portion of the URL (after the #) to handle navigation. It is useful when you need to support older browsers that do not fully support HTML5 history API. Hash-based URLs are easier to set up and work in a wider range of environments.
- MemoryRouter: This router does not read or write to the browser's URL. Instead, it keeps the current location in memory. It is primarily used for testing and non-browser environments like React Native.

Post-Lab:

Question 1: How can you create a dynamic route in React Router that accepts a parameter and renders different content based on that parameter?

Question 2: How can you implement nested routes in React Router to create a hierarchical navigation structure?

Procedure/Program:

1

```
//App.js
import React from 'react';
import { BrowserRouter as Router, Switch, Route } from 'react-router-dom';
import Home from './Home';
```

```
import UserProfile from './UserProfile';
import NotFound from './NotFound';
function App() { return ( <Router>
<Switch>
<Route exact path="/" component={Home} />
<Route path="/user/:userId" component={UserProfile} />
<Route component={NotFound} />
</Switch>
</Router>
);
} export default App;

//Home.js
import React from 'react';
const Home = () => { return <div>Welcome to the Home page!</div>;
};
export default Home;

// NotFound.js
import React from 'react';
const NotFound = () => {
return <div>Page not found!</div>;
};
export default NotFound;

//UserProfile.js
import React from 'react';
const UserProfile = (props) => {
const userId = props.match.params.userId;

// Fetch user data or do any other logic based on the userId


return (
<div>
<h2>User Profile</h2>
<p>User ID: {userId}</p>
{/* Render user-specific content based on the userId */}
</div>
);
}; export default UserProfile;

// UserList.js
import React from 'react';
import { Link } from 'react-router-dom';
const UserList = () => { const users = [
{ id: 1, name: 'John' },
{ id: 2, name: 'Jane' }, // Add more users here...
]; return (
<div>
<h2>User List</h2>
<ul>
{users.map((user) => (
<li key={user.id}>
<Link to={`/user/${user.id}`}>{user.name}</Link>
</li>
```

```
))}
</ul>
</div>
);
}; export default UserList;
```

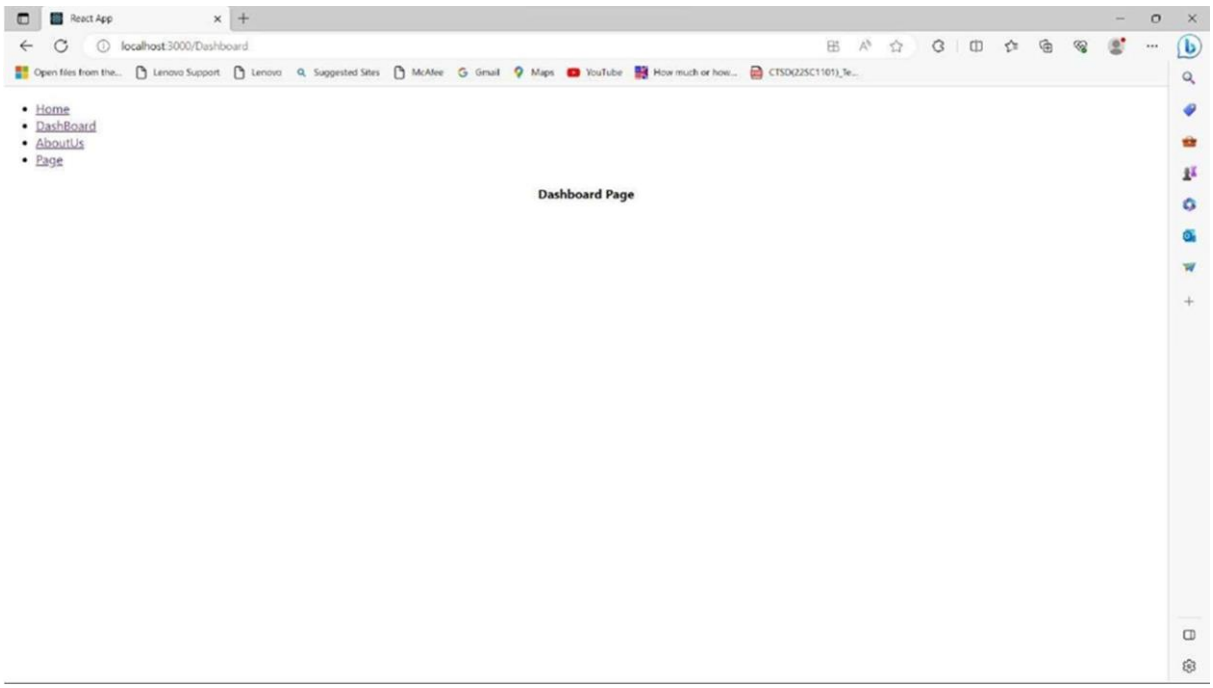
2

```
// App.js
import { BrowserRouter as Router, Route, Switch } from 'react-router-dom';
import Home from './Home';
import About from './About';
import Contact from './Contact';
const App = () => { return ( <Router>
<Switch>
<Route exact path="/" component={Home} />
<Route path="/about" component={About} />
<Route path="/contact" component={Contact} />
</Switch>
</Router>
);
};
export default App;

// Home.js
import { Link } from 'react-router-dom';
const Home = () => { return (
<div>
<h2>Home</h2>
<Link to="/about">About</Link>
<Link to="/contact">Contact</Link>
</div>
);
}; export default Home;
// About.js import { Route, Link } from 'react-router-dom'; import Team from
'./Team'; import History from './History';
const About = () => { return (
<div>
<h2>About</h2>
<ul>
<li>
<Link to="/about/team">Team</Link>
</li>
<li>
<Link to="/about/history">History</Link>
</li>
</ul>

<Route path="/about/team" component={Team} />
<Route path="/about/history" component={History} />
</div>
);
};
export default About; // Team.js const Team = () => { return <div>Our
Team</div>;
```

```
}; export default Team; // History.js const History = () =>
{ return <div>Our History</div>;
};
export default History;


// Contact.js
const Contact = () => { return <div>Contact Us</div>;
};
export default Contact;
```

□ Data and Results:

1



2


HOME
COURSES
You are in the Home page!
URL: localhost:3000/