DEPARTMENT OF CSE
COURSE CODE: 23SDCS12A / 23SDCS12R
FULL STACK APPLICATION DEVELOPMENT

Date of the Session: ___/___/_____ **Time of The Session:______to_______**

LAB – 4 ➔ Implementation of multiple components as a "Single Page React App" with Redux State Management for Routing.

Prerequisites:
Having a basic understanding of HTML and CSS is valuable for building user interfaces and styling your React components.

>npm install redux

Exercise 1:

Create a single page simple React application with two routes: Home and About. Implement navigation links to switch between these routes using Redux.

Exercise 2:

Implement a nested routing structure in a React application using Redux. Create a parent route and two child routes that are rendered within the parent component. All these as a single page app.

index.html

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <link rel="icon" type="image/svg+xml" href="/vite.svg" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Vite + React</title>
  </head>
  <body>
    <div id="root"></div>
    <script type="module" src="/src/main.jsx"></script>
  </body>
</html>
```

❖ Watch The Video And Do In Visual Studio https://youtu.be/2Tb4Qe-yPIk?si=1QmPBsAhbByM2D7M

Page1.jsx

```
import './Style.css';
export default function Page1({store}){

   function navigation(event){
   store.dispatch({"type": "page", "data": event.currentTarget.getAttribute("val")})
   }
return(
   <div>
<a onClick={navigation} val={"Page1"}>Page 1</a>
<a onClick={navigation} val={"P2p1"}>Page 2</a>
<br/><br/><br/>
This is Page 1
   </div>
);

}
```

P2p1.jsx

```
import './Style.css';

export default function P2p1({store}){

   function navigation(event){
      store.dispatch({"type": "page","data": event.currentTarget.getAttribute("val")})


   }
return(
   <div>
<a onClick={navigation} val={"Page1"}>Page 1</a>
<a onClick={navigation} val={"P2p1"}>Page 2</a>
<br/><br/><br/>
<a onClick={navigation} val={"P2p1"}>P2 - P1</a>
<a onClick={navigation} val={"P2p2"}>P2 - P2</a>
<br/><br/><br/>
    This is Page 2  - Page 1
   </div>
);

}
```

P2p2.jsx

```
import './Style.css';

export default function P2p2({store}){

   function navigation(event){
      store.dispatch({"type": "page","data": event.currentTarget.getAttribute("val")})

   }
return(
   <div>
<a onClick={navigation} val={"Page1"}>Page 1</a>
<a onClick={navigation} val={"P2p1"}>Page 2</a>
<br/><br/><br/>
<a onClick={navigation} val={"P2p1"}>P2 - P1</a>
<a onClick={navigation} val={"P2p2"}>P2- P2</a>
<br/><br/><br/>
This is Page 2  - Page 2
   </div>
);

}
```

Content.jsx

```
import Page1 from "./Page1"
import P2p1 from "./P2p1"
import P2p2 from "./P2p2"

export default function Content({store}){

    function Page(){
        switch(store.getState()){
            case "Page1":
                return (<div> <Page1 store={store}/></div>)
                case "P2p1":
                return (<div> <P2p1 store={store}/></div>)
                case "P2p2":
                return (<div> <P2p2 store={store}/></div>)
                default:
                    return (<div> <Page1 store={store}/></div>)
        }
    }

    return(
        <div>
<Page />

        </div>
    )
}
```

NavReducer.jsx

```
export default function NavReducer (state="page1",action){

   switch(action.type){
      case"page":
      state = action.data;
      return state;
      default:
         return state;

   }
}

//action = {"type": "page","data": "page1"}
```

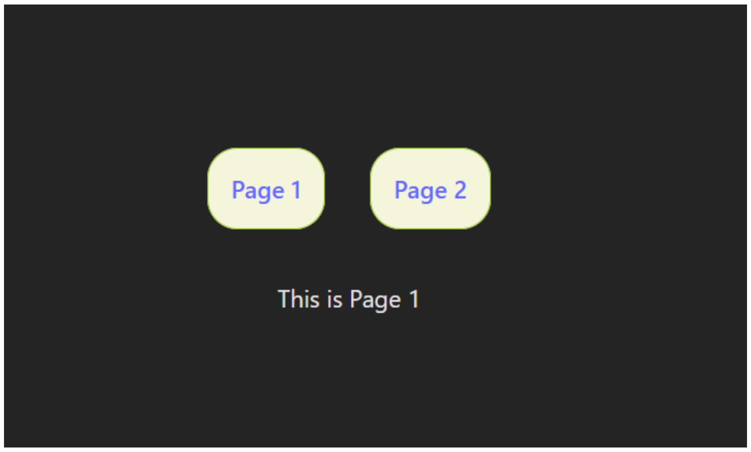
Style.css

```
a {
   border: 1px solid yellowgreen;
   padding: 15px;
   margin: 15px;
   background-color: beige;
   border-radius: 20px;
}
```

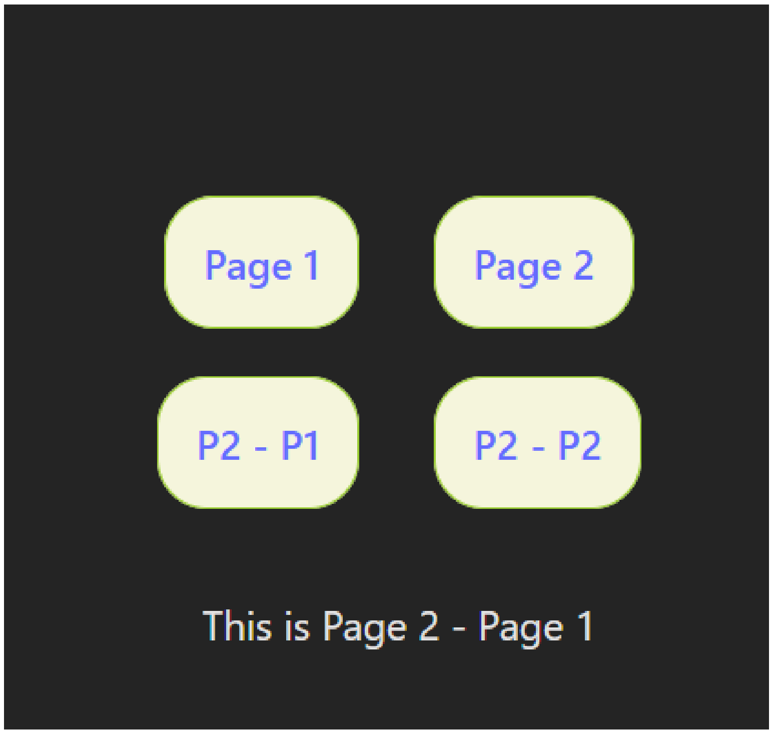
main.jsx

```
import { StrictMode } from "react";
import { createRoot } from "react-dom/client";
import "./index.css";
import App from './App.jsx'
import Content from './components/lab4/Content.jsx'
import { createStore } from 'redux'
import NavReducer from './components/lab4/reducer/NavReducer'
var store = createStore(NavReducer)



var customer = () => createRoot(document.getElementById("root")).render(
  <StrictMode>
   <Content store={store}/>
  </StrictMode>,
)
customer()
store.subscribe(customer)
```

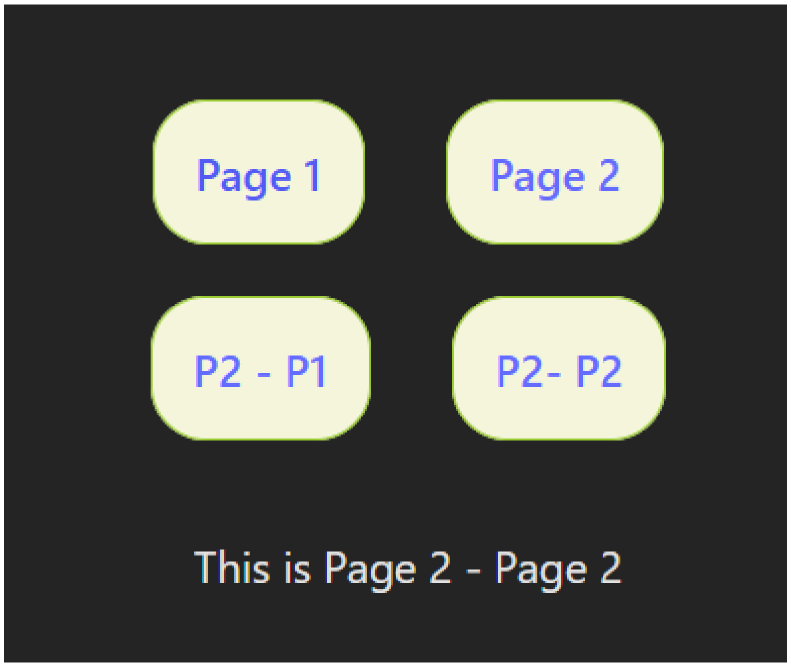
Output



This is Page 1



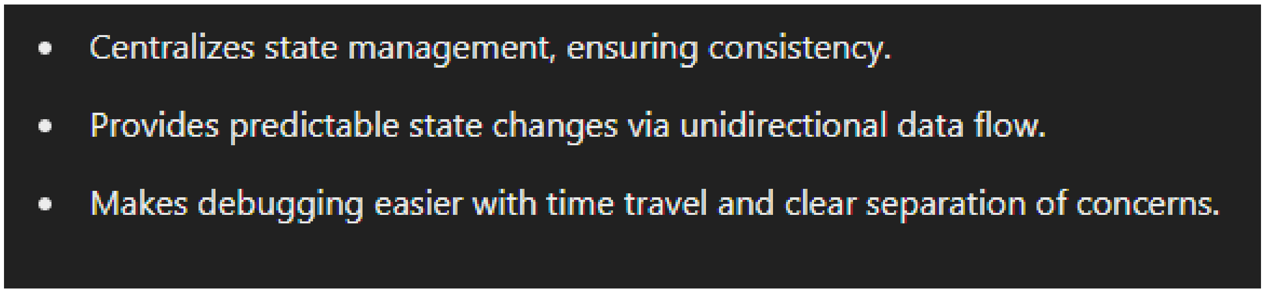
This is Page 2 - Page 1



This is Page 2 - Page 2

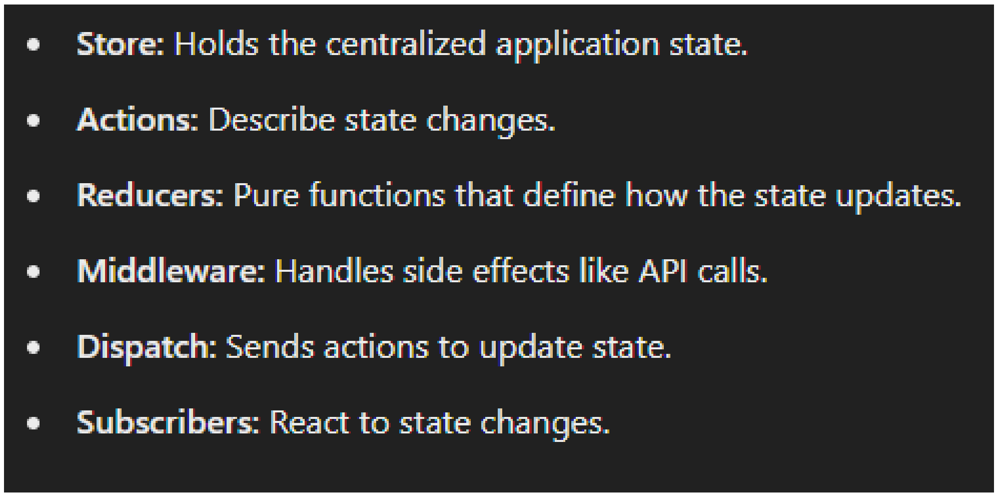
VIVA QUESTIONS:

1. How does Redux help in managing the state of an application?



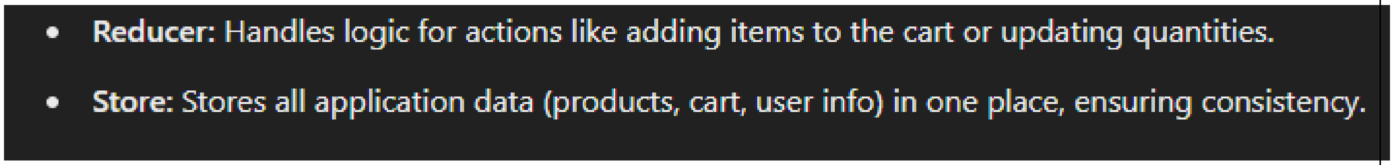
- Centralizes state management, ensuring consistency.
- Provides predictable state changes via unidirectional data flow.
- Makes debugging easier with time travel and clear separation of concerns.

2. List the key components of Redux and their roles?



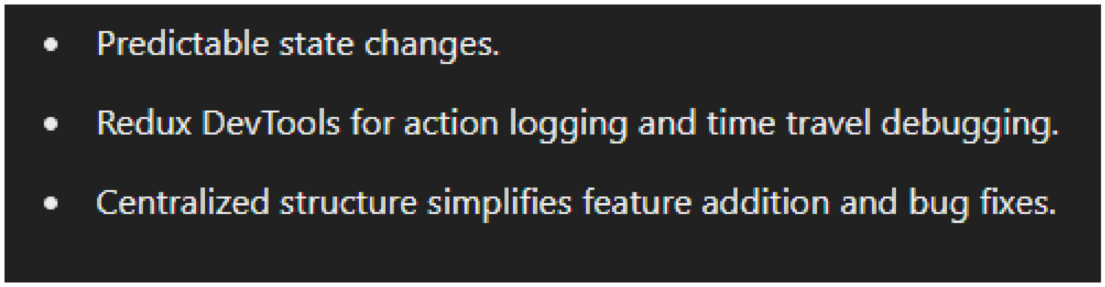
- **Store:** Holds the centralized application state.
- **Actions:** Describe state changes.
- **Reducers:** Pure functions that define how the state updates.
- **Middleware:** Handles side effects like API calls.
- **Dispatch:** Sends actions to update state.
- **Subscribers:** React to state changes.

3. What is the role of reducer and store in any e-commerce application.



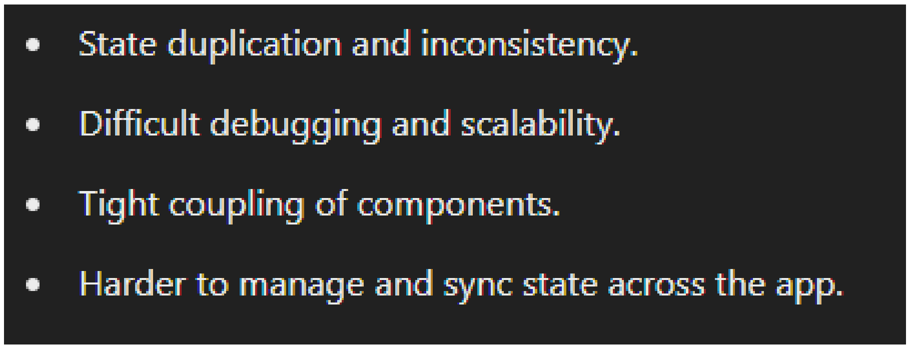
- **Reducer:** Handles logic for actions like adding items to the cart or updating quantities.
- **Store:** Stores all application data (products, cart, user info) in one place, ensuring consistency.

4. What are the benefits of using Redux in terms of debugging and maintaining application state?



- Predictable state changes.
- Redux DevTools for action logging and time travel debugging.
- Centralized structure simplifies feature addition and bug fixes.

5. What is drawback developer need to face without redux?



- State duplication and inconsistency.
- Difficult debugging and scalability.
- Tight coupling of components.
- Harder to manage and sync state across the app.

(For Evaluator's use only)

Comment of the Evaluator (if Any)	Evaluator's Observation
	Marks Secured ________out of 50 Full Name of the Evaluator: Signature of the Evaluator Date of Evaluation: