React – Applying Redux

□ Theory Assignment

Q1: What is Redux in React?

Answer:
Redux is a **state management library** for JavaScript applications. It helps manage the state of an entire application in a **single centralized store**, making data flow predictable and easier to debug.

In React, Redux is often used when multiple components need access to the same shared state — instead of passing props manually through many component levels.

Q2: Why do we use Redux in React?

Answer:
Redux is used to:

- Manage **complex application state** efficiently.
- Maintain a **single source of truth** (the store).
- Simplify **data flow** across components.
- Make applications **easier to debug** and **test**.
- Avoid **prop drilling** (passing data through multiple nested components).

Q3: What are the core principles of Redux?

Answer:

1. **Single Source of Truth** – All application state is stored in a single object called the *store*.
2. **State is Read-Only** – You cannot modify the state directly; you must dispatch an action.
3. **Changes are Made with Pure Functions** – Reducers specify how the state changes in response to actions.

Q4: What are the main components of Redux?

Component	Description
Store	Holds the entire state tree of the application.
Action	An object describing what happened. Example: { type: 'ADD_TODO', payload: 'Learn Redux' }
Reducer	A pure function that takes the current state and an action, and returns the new state.
Dispatch	A method to send actions to the store.
Selector	A function that retrieves specific data from the store.

Q5: How does data flow work in Redux?

Redux Data Flow:

1. The **user interacts** with the UI.
2. A **Redux action** is **dispatched**.
3. The **reducer** handles the action and updates the **store**.
4. The **React components** subscribed to the store **re-render automatically**.

⚙ **Practical / Lab Assignment**

Task 1: Setup Redux in a React App

Steps:

1. Create a React app:
2. npx create-react-app redux-demo
3. cd redux-demo
4. Install Redux and React Redux:
5. npm install redux react-redux

Task 2: Create Redux Store and Reducer

File: src/redux/store.js

```
import { createStore } from 'redux';
import counterReducer from './counterReducer';

const store = createStore(counterReducer);
export default store;
```

File: src/redux/counterReducer.js

```
const initialState = {
  count: 0
};

function counterReducer(state = initialState, action) {
  switch (action.type) {
    case 'INCREMENT':
      return { ...state, count: state.count + 1 };
    case 'DECREMENT':
      return { ...state, count: state.count - 1 };
    default:
      return state;
  }
}

export default counterReducer;
```

Task 3: Create Action Creators

File: src/redux/actions.js

```
export const increment = () => ({ type: 'INCREMENT' });

export const decrement = () => ({ type: 'DECREMENT' });
```

Task 4: Connect Redux Store with React

File: src/index.js

```
import React from 'react';

import ReactDOM from 'react-dom/client';

import App from './App';

import { Provider } from 'react-redux';

import store from './redux/store';


const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

  <Provider store={store}>

    <App />

  </Provider>

);
```

Task 5: Use Redux State and Actions in Component

File: src/App.js

```
import React from 'react';

import { useSelector, useDispatch } from 'react-redux';
```

```
import { increment, decrement } from './redux/actions';


function App() {
  const count = useSelector(state => state.count);
  const dispatch = useDispatch();


  return (
    <div style={{ textAlign: "center", marginTop: "50px" }}>
      <h2>React Redux Counter Example</h2>
      <h3>Count: {count}</h3>
      <button onClick={() => dispatch(increment())}>Increment</button>
      <button onClick={() => dispatch(decrement())}>Decrement</button>
    </div>
  );
}


export default App;
```