THEORY EXERCISE

14. State Management (Redux, Redux-Toolkit or Recoil)

Question 1: What is Redux, and why is it used in React applications?

Redux is a **state management library** used to manage and centralize application state in React apps.

✓ **Why Redux is used:**

- React only handles **local component state**.
- Redux provides a **single global state** (called **store**) that all components can access.
- Helps avoid **prop drilling** (passing props multiple levels deep).
- Useful in **large apps** where many components need to share state.

✓ **Core Concepts:**

1. **Actions**
    - Plain JavaScript objects that describe **what happened**.
    - Must have a type property.
    - Example:

```
{ type: 'INCREMENT' }
```

2. **Reducers**
    - Functions that take the **current state** and an **action**, then return a **new state**.
    - Pure functions (no side effects).
    - Example:

```
function counterReducer(state = { count: 0 }, action) {
  switch (action.type) {
    case 'INCREMENT':
      return { count: state.count + 1 };
    default:
      return state;
  }
}
```

3. **Store**
   - The **single source of truth**.
   - Holds the app’s state, allows access via getState(), and updates via dispatch(action).
   - Example:

```
import { createStore } from 'redux';

const store = createStore(counterReducer);
```

Question 2: How does Recoil simplify state management in React compared to Redux?

Recoil is a state management library for React built by Facebook, designed to be more **reactive and simpler** than Redux.

- ➢ **How Recoil simplifies things:**

Feature	**Redux**	**Recoil**
Boilerplate	More code: actions, reducers, store	Minimal code, more React-like
Async handling	Requires middleware (e.g., redux-thunk)	Built-in support using selectors
Global vs local state	Mostly global (central store)	Can create atom-like local state that’s sharable
Setup	Needs Provider, store configuration	Lightweight setup, no store needed
Learning curve	Steeper	Easier for beginners already using React hooks

- ✓ **Recoil Core Concepts:**

1. **Atoms** – units of state (like useState) but globally accessible.
2. **Selectors** – derived or computed state (like useMemo but for atoms).
3. **useRecoilState** – similar to useState but works with atoms.

- ✓ **Example:**

```
// counterAtom.js
```

```
import { atom } from 'recoil';

export const counterState = atom({

  key: 'counterState',

  default: 0,

});

// Counter.jsx

import { useRecoilState } from 'recoil';

import { counterState } from './counterAtom';

function Counter() {

  const [count, setCount] = useRecoilState(counterState);

  return (

    <>

      <h1>{count}</h1>

      <button onClick={() => setCount(count + 1)}>Add</button>

    </>

  );

}
```