# **State Management (Redux, Redux-Toolkit or Recoil)**

**Question 1:** What is Redux, and why is it used in React applications? Explain the coreconcepts of actions, reducers, and the store

**Ans. Redux** is a state management library often used with React applications. It provides a predictable way to manage and centralize the application's state, making it easier to build, debug, and maintain complex apps.

Redux allows the application state to be stored in a single source of truth called the **store**, making state updates predictable and ensuring that the app behaves consistently.

**Concepts of Redux**

1. **Actions**:  
   Actions are plain JavaScript objects that describe what should be done. They have a mandatory type property (a string) and an optional payload property to pass data.
2. **Reducers:**  
   Reducers are pure functions that specify how the application's state should change in response to an action. They take the current state and an action as arguments and return a new state.
3. **Store**:   
   The store is a single JavaScript object that holds the entire state of the application. It is created using the createStore function from Redux.

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

**Ans. Recoil** simplifies state management in React compared to **Redux** by offering a more React-centric approach with minimal boilerplate. Here are the key points of comparison:

1. **State Granularity**:
   * **Recoil** uses **atoms** for small, independent units of state, which can be updated individually.
   * **Redux** uses a **single, global store**, requiring actions and reducers for every state change.
2. **Boilerplate**:
   * **Recoil** requires less boilerplate: just define **atoms** and **selectors**.
   * **Redux** requires defining **actions**, **reducers**, and middleware.
3. **Reactivity**:
   * **Recoil** automatically tracks dependencies between atoms and selectors, re-rendering only the components that use the updated state.
   * **Redux** re-renders components connected to the global store, potentially causing unnecessary updates.
4. **Async Handling**:
   * **Recoil** uses **selectors** to handle asynchronous logic natively.
   * **Redux** requires middleware like **Thunk** for async handling.