**Q.1. What is Redux?**

Ans => Redux is a state management library for JavaScript applications, often used with React but also with other libraries or vanilla JavaScript. It helps manage the state (data) of an application in a centralized store, making it easier to keep the state predictable and consistent

**Q.2. What is Redux Thunk used for?**

**Ans =>**Redux Thunk is a middleware for Redux that allows you to write action creators that return a function instead of an action object. This function can be used to perform asynchronous operations (like fetching data from an API) or perform delayed state updates.

Why Use Redux Thunk

In a standard Redux flow, action creators can only return plain objects, which makes handling asynchronous logic difficult. With Redux Thunk, you can:

Dispatch multiple actions: A thunk function can dispatch multiple actions, allowing you to handle different states of an async operation (e.g., loading, success, error).

Access the current state: The thunk function receives dispatch and getState as arguments, which lets you read the current state and conditionally dispatch actions.

Perform side effects: It enables you to perform side effects like data fetching, set timers, or call other APIs.

**Q.3. What is Pure Component? When to use Pure Component over Component?**

Ans => Pure Component is a class component that automatically performs a shallow comparison of its props and state to determine whether a re-render is necessary. It is a subclass of React.PureComponent, which is similar to React.Component, but with an optimized shouldComponentUpdate method.

Use Pure Component:

When the Props and State are Immutable: If you are using immutable data (i.e., data that does not change), a Pure Component can help optimize the re-rendering process.

When Performance Optimization is Needed: If your component re-renders frequently but the props or state do not change often, using a Pure Component can help improve performance.

For Simple Shallow Comparison Scenarios: When your state or props are simple data types (strings, numbers, booleans), a shallow comparison is usually sufficient.

**Q.4. What is the second argument that can optionally be passed tosetState and what is its purpose?**

Ans => The second argument that can optionally be passed to setState in React is a callback function. This callback function is executed after the state has been updated and the component has been re-rendered.

Second Argument:

Execute Code After State Update: Since setState is asynchronous, the callback allows you to perform actions that depend on the updated state after the state change has been applied.

Handle Side Effects: It can be used to trigger side effects, such as making API calls, logging data, or updating external libraries that rely on the latest state.

Ensure Code Runs After Re-render: The callback ensures that the code will only execute once the component has been re-rendered with the updated state, avoiding situations where code runs before the state change takes effect.