1. What is FLUX?

Ans: Flux is a pattern for managing application state and data flow in a predictable way, primarily used in single-page applications (SPAs). Flux is often compared to Redux, which is another state management library that builds on the ideas of Flux but with a more centralized approach (a single store). Redux simplifies the Flux pattern by using a single global store and pure functions (reducers) to manage state updates.

Flux also has concepts like: Action, Dispatcher, Store, View.

2. What is Redux? How do you use it with React components?

Ans: Redux is a state management library that helps keep track of the state in your application by using a single store. It works with actions (which describe what happened) and reducers (which define how the state changes) to manage the state in a predictable way. In a React app, you can connect Redux to your components using the react-redux library, where useSelector helps you access the state, and useDispatch lets you trigger actions to update the state. Redux uses a Provider component from the react-redux library to wrap your React app and make the Redux store available to all components.

3. What is a reducer?

Ans: A reducer is a pure function in Redux that takes the current state and an action as inputs and returns a new state based on the action type. It specifies how the state of the application should change in response to a given action, without modifying the original state.

4. How do you choose between ContextAPI and Redux for global state management?

Ans: Use Context API for simpler, smaller applications, and Redux for larger, more complex applications where robust state management is needed.

Context API is great for sharing state between components without prop drilling and works well for managing basic global state like themes, user authentication, or settings. But can lead to performance issues if used for large or frequently updated state because it causes all components within the context to re-render when the state changes.

Redux can be used when the app has a more complex or large-scale state that involves frequent updates, multiple slices of state, or requires features like time-travel debugging, middleware, or advanced state logic. It's ideal for applications that need predictable and centralized state management across many components.

5. What is redux thunk and why do you want to use it?

Ans: Redux Thunk is a middleware that lets you handle asynchronous operations in Redux, like fetching data from an API. Instead of just returning an action object, it allows you to return a function that can delay the action until the operation is complete, or even dispatch multiple actions. It's useful when you need to manage tasks like API calls directly within Redux, keeping everything organized and in sync with your state.