● useContext vs Redux.

If your application is small to medium-sized and has relatively simple state management needs, useContext might be sufficient and more straightforward.

If your application is large, has complex state management requirements, or you foresee the need for features like time-travel debugging, middleware, or persisting state to local storage, then Redux might be a better choice.

It's also worth mentioning that the use of these tools is not mutually exclusive. In fact, some applications use both useContext for local state management within components and Redux for managing global application state.

● Advantage of using Redux Toolkit over Redux.

Redux Toolkit is a set of utility functions and conventions that simplify and streamline the process of working with Redux. It provides a more opinionated and efficient way of writing Redux code, addressing some of the common pain points and boilerplate associated with traditional Redux setups.

● Explain Dispatcher.

In the context of state management libraries like Redux, the term "dispatcher" refers to a concept or a part of the system responsible for dispatching actions to the store

● Explain Reducer.

Reducers are pure functions that specify how the state should change in response to an action. They take the current state and an action as parameters and return the new state

● Explain slice.

The createSlice function is part of the Redux Toolkit and is used to define a Redux slice. It takes an object with several properties, including name, initialState, reducers, and optionally extraReducers

● Explain selector.

It to subscribe the reducers object to load/retrieve to action/iteams

● Explain createSlice and the configuration it takes.

createSlice is a function provided by Redux Toolkit that simplifies the process of creating Redux slices. A slice in Redux represents a portion of the application state along with the related logic for updating that state. The createSlice function combines the definition of action creators, action types, and reducers into a single unit, reducing boilerplate code and making Redux code more concise and readable.