NEED OF CENTRALIZED STATE MANAGEMENT:

Centralized state management is the practice of storing and managing the state of an application in a single location, often referred to as a "store". This approach is useful in larger applications where multiple components need access to the same state or where state needs to be shared between different parts of the application.

Redux is a popular library for implementing centralized state management in JavaScript applications, particularly in React.js. It simplifies state handling by providing a predictable state container and a set of rules for updating that state.

Here's a simplified example to demonstrate how Redux simplifies state handling:

1. Define the initial state: In Redux, the initial state is defined as a plain JavaScript object.
2. Create actions: Actions are plain JavaScript objects that describe an event or intention to change the state. They typically have a **type** property that specifies the type of action and optional payload data.
3. Create reducers: Reducers are pure functions that take the current state and an action as parameters, and return a new state based on the action type. They should not mutate the state directly, but rather create a new state object.
4. Create the store: The store holds the application state and provides methods to dispatch actions and subscribe to state changes.
5. Use the store in components: Components can access the state from the store using selectors, and dispatch actions to update the state. Redux provides a **connect** function or hooks like **useSelector** and **useDispatch** to simplify this integration.

The benefits of using Redux for state handling in larger applications include:

- **Centralized state**: Redux provides a single source of truth for the application state, making it easier to understand and modify.
- **Predictable state changes**: Redux enforces a strict unidirectional data flow, making it easier to understand how the state changes in response to actions.
- **Debugging and time-traveling**: Redux keeps a log of dispatched actions, allowing for easy debugging and the ability to go back and forth in time to inspect the state at different points.
- **Testability**: Redux promotes the use of pure functions, making it easier to test reducers and other parts of the application that rely on the state.

Overall, Redux simplifies state handling in larger applications by providing a clear structure and set of rules for managing and updating the state.