**Redux Auth Routing**

1. **Task Description**

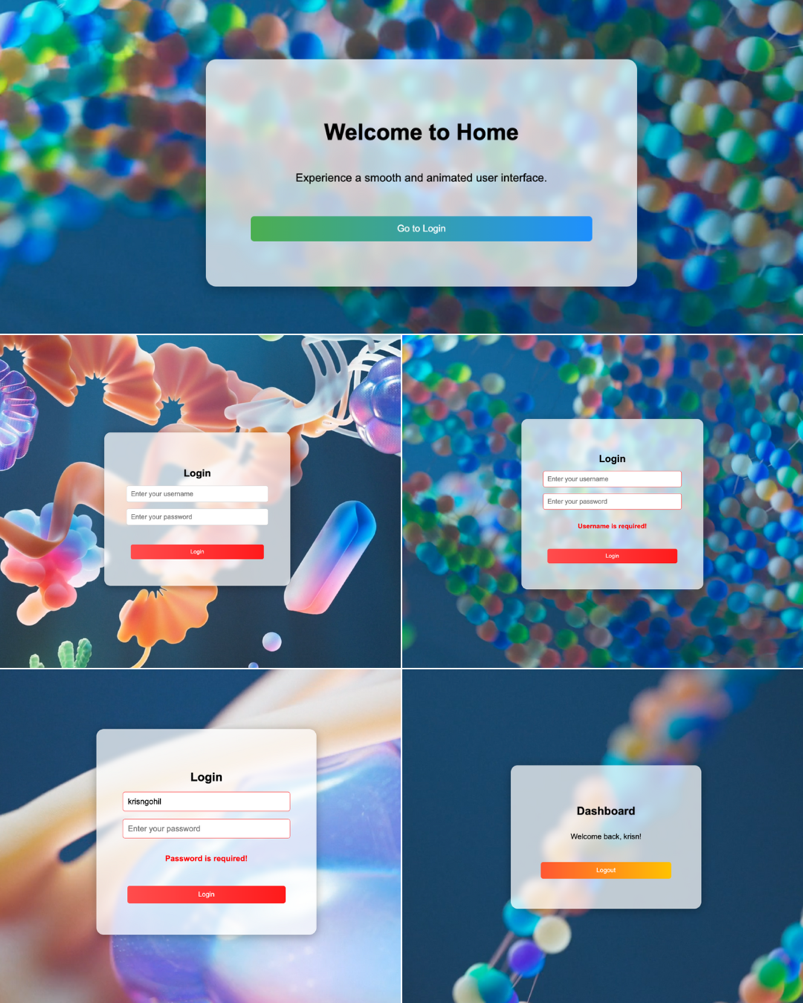
The task involves creating a user authentication system using Redux for managing login and logout functionality. The application features routes and a protected dashboard page that requires user authentication. The user can log in using their username and password, and after successful authentication, they are redirected to the dashboard. If the user tries to access the dashboard without logging in, they are redirected to the login page.

The project also involves integrating animations and transitions on various pages such as the login page, home page, and dashboard.

**Key Features:**

* Login and logout functionality using Redux
* Conditional rendering based on authentication status
* Protected route implementation for the dashboard
* Animated transitions for smooth user experience
* Basic validation for username and password inputs

1. **Task Output Screenshot**



1. **Widget/Algorithm Used In Task**

* **React Router**: Used for routing between different pages (Home, Login, Dashboard, and NotFound). It helps manage navigation between routes and controls protected routes based on authentication status.
* **Redux**: The createSlice method from Redux Toolkit is used to manage authentication state. It stores whether the user is logged in and the user's data.
* **Reducers**: login to set the user as authenticated and logout to clear the authentication state.
* **Selectors**: Used to retrieve the current authentication state from the Redux store.
* **useState**: Used to manage state for username, password, and error messages on the login form.
* **useDispatch**: Used to dispatch actions like login and logout to the Redux store.
* **useNavigate**: A hook from React Router used for navigating programmatically between pages.
* **useSelector**: A hook to select data from the Redux store, such as whether the user is authenticated and user details.
* **Validation Algorithm**: Basic validation checks are added to ensure the username is at least 5 characters long and only contains alphanumeric characters or underscores. Passwords must be at least 6 characters long.
* **CSS Animations**: Animation classes like fade-in, bounce-in, and btn-error-gradient were used to add visual effects on elements such as login input fields and buttons.
* **ProtectedRoute Component**: This component checks the authentication state and either renders the children or redirects to the home page if the user is not authenticated.