**Student Have To Prepare Report In Format**

1. Add Task Description
2. Attach Screenshot Of Output.
3. Describe Widget/Algorithm Used In Task
4. Add Report In Your Task Zip File

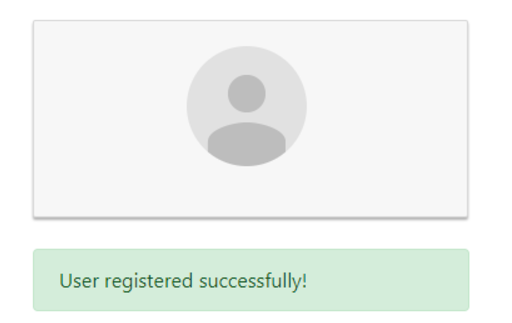
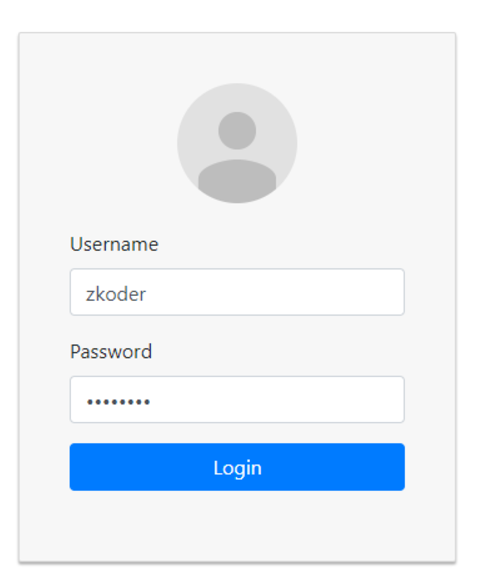
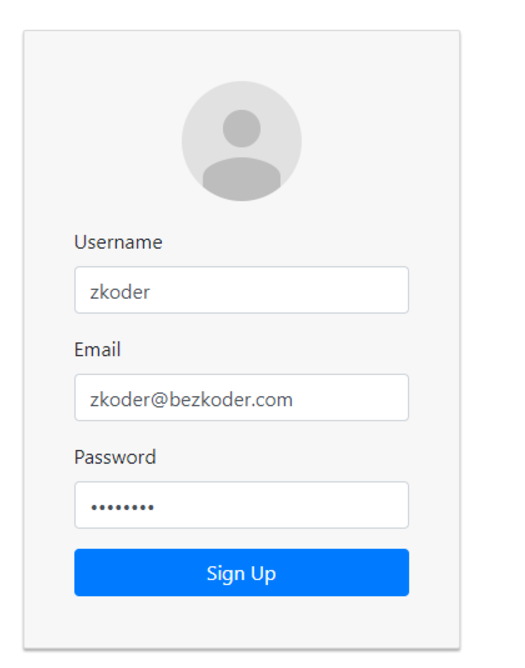
**Sample Example :**

1. **Task Description**

This project implements a **Redux store for managing user authentication** using **Redux Toolkit**. It includes:

* **Authentication State Management**: Tracks isAuthenticated, user, and token in the Redux store.
* **Actions**: Provides login and logout actions to update authentication state.
* **Components**: Includes a Login component to simulate user login and a LogoutButton component for user logout.
* **State Access**: Uses useSelector to display authentication status and user information dynamically (e.g., in a Dashboard component).
* **Store Configuration**: Combines reducers into a central store and integrates with the app using the Provider component.

1. **Task Output Screenshot**



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

* **1. Widgets (UI Components)**
* **Login Form**
* **Purpose**: Collects user credentials (username and password).
* **Implementation**:
* Controlled components (<input> fields) for user input.
* Login button dispatches the login action with mock or real API response.
* **Logout Button**
* **Purpose**: Allows users to log out of the application.
* **Implementation**:
* A button that triggers the logout action to reset the authentication state.
* **Protected Route (Optional)**
* **Purpose**: Ensures certain pages are only accessible to authenticated users.
* **Implementation**:
* A conditional rendering component that checks the isAuthenticated flag and redirects unauthenticated users.
* **Dashboard Component**
* **Purpose**: Displays personalized content based on the authenticated user's information.
* **Implementation**:
* Fetches and displays user details from the Redux store using useSelector.
* **Navigation Bar**
* **Purpose**: Dynamically updates links and buttons based on authentication status.
* **Implementation**:
* Conditional rendering to show login/register links for unauthenticated users or dashboard/logout links for authenticated users.
* **2. Algorithms**
* **Authentication State Management**
* **Purpose**: Centralized control of user authentication state (isAuthenticated, user, and token).
* **Implementation**:
* Uses Redux Toolkit’s createSlice to define initial state, reducers, and actions.
* **Login Logic**
* **Purpose**: Handles user login process.
* **Implementation**:
* Captures user input.
* Simulates or integrates with an API to validate credentials.
* Updates the Redux store with user details and token on success.
* **Logout Logic**
* **Purpose**: Handles user logout process.
* **Implementation**:
* Clears the authentication state in the Redux store by resetting isAuthenticated, user, and token.
* **Token Validation (Optional)**
* **Purpose**: Ensures that the stored token is valid and active.
* **Implementation**:
* Checks the token’s expiration or sends a request to the server for validation (can be added later).
* **Dynamic Navigation**
* **Purpose**: Dynamically updates the navigation bar to reflect the current user state.
* **Implementation**:
* Uses conditional rendering based on the isAuthenticated state.
* **Protected Routing**
* **Purpose**: Restricts access to certain pages based on the user’s authentication status.
* **Implementation**:
* Custom logic in a higher-order component or route wrapper to check the isAuthenticated flag and redirect unauthenticated users.