

React Practical Assessment

Task 1 – Authentication Navbar (React Bootstrap + Global State)

Objective

To understand how global authentication state controls the user interface using React Bootstrap, props, and conditional rendering.

Problem Statement

Create a React application and install React Bootstrap. Build a navigation bar that changes its content based on whether the user is logged in or not.

Requirements

1. Install React Bootstrap and import Bootstrap CSS globally.
2. Create a global authentication context named AuthContext.
3. The context must store:
 - o isLoggedIn (boolean)
 - o user object (example: { name: "Student" })
 - o login() function
 - o logout() function
4. Create a Navbar component using React Bootstrap.
5. When the user is logged out:
 - o Show a **Login** button.
6. When the user is logged in:
 - o Show **Welcome, UserName**
 - o Show a **Logout** button.
7. The Navbar must receive all required data through **props**, not directly from AuthContext.
8. The UI must update automatically when login or logout is performed.

Task 2 – Login Page with Validation and Navigation

Objective

To practice form handling, state management, conditional rendering, and navigation.

Problem Statement

Create a Login page using React Bootstrap that validates user input and redirects the user after successful login.

Requirements

1. Create a Login page using React Bootstrap Form, Form.Control, and Button.
 2. Use useState to store email and password values.
 3. Disable the Login button if:
 - o Email is empty, or
 - o Password is empty.
 4. On clicking Login:
 - o Update the global authentication state.
 - o Navigate the user to /dashboard.
 5. Show an error message conditionally if login requirements are not met.
-

Task 3 – Protected Routes Implementation

Objective

To restrict access to certain routes based on authentication status.

Problem Statement

Protect the dashboard route so that only authenticated users can access it.

Requirements

1. Create a ProtectedRoute component.
 2. If the user is not logged in:
 - o Redirect the user to /login.
 3. Protect the /dashboard route using ProtectedRoute.
 4. Display a React Bootstrap Alert message when access is denied.
-

Task 4 – Dashboard with Nested Routes

Objective

To understand nested routing and layout reuse.

Problem Statement

Create a dashboard layout with a sidebar and nested routes.

Requirements

1. Create a /dashboard layout using React Bootstrap.
2. Add a sidebar that is always visible on the dashboard.

3. Create nested routes:
 - o /dashboard/profile
 - o /dashboard/settings
 4. Use <Outlet /> to render nested components.
 5. Pass logged-in user information to child components using props.
-

Task 5 – Profile Page with API Call

Objective

To use useEffect for data fetching and conditional UI rendering.

Problem Statement

Fetch and display user profile information when the profile page loads.

Requirements

1. Fetch profile data from a mock API or static JSON when the page loads.
 2. Use useEffect to perform the API call.
 3. Show a loading spinner while data is being fetched.
 4. Display the profile data inside a React Bootstrap Card.
-

Task 6 – Global Cart State Using Context

Objective

To manage shared application state using React Context.

Problem Statement

Create a global shopping cart that can be accessed by multiple components.

Requirements

1. Create a CartContext.
2. The context must store:
 - o cartItems
 - o addCartItem(item)
 - o removeFromCart(itemId)
3. Use the cart state in:
 - o ProductList
 - o CartSummary

-
4. Display “Cart is empty” when there are no items in the cart.

Task 7 – Product List with Conditional UI

Objective

To understand reusable components, props, and conditional rendering.

Problem Statement

Display a list of products using a reusable product card component.

Requirements

1. Create a reusable ProductCard using React Bootstrap Card.
2. Pass the following props:
 - o name
 - o price
 - o isAvailable
3. Disable the **Add to Cart** button when the product is not available.
4. Display a badge:
 - o “Available” if in stock
 - o “Out of Stock” if not available.

Task 8 – Dynamic Route with Parameters

Objective

To work with dynamic routes and data fetching.

Problem Statement

Display product details based on a dynamic route parameter.

Requirements

1. Create a route: /dashboard/product/:id.
2. Read the product ID using useParams.
3. Fetch product details based on the ID using useEffect.
4. Display product details using React Bootstrap components.

Task 9 – Role-Based UI Rendering

Objective

To implement real-world role-based UI logic.

Problem Statement

Show or hide UI elements based on the user's role.

Requirements

1. The user object must contain a role property (admin or user).
 2. Store the role in AuthContext.
 3. Display an **Admin Panel** link only if the user role is admin.
 4. Hide admin-related UI for normal users.
-

Task 10 – Simple Secure Notes Application

Objective

To integrate authentication, routing, global state, and conditional rendering in one simple application.

Problem Statement

Create a Secure Notes Application where users can log in and manage personal notes.

Application Requirements

Authentication

1. Create an AuthContext.
2. Store:
 - o isLoggedIn
 - o user (name only)
 - o login() and logout()

Routing

1. Create the following routes:
 - o /login
 - o /notes
2. Protect the /notes route.
3. Redirect unauthenticated users to /login.

User Interface

1. Use React Bootstrap components:
 - o Navbar

- o Form
 - o Button
 - o Card
2. Maintain a clean and simple layout.

Notes State

1. Create a NotesContext.
2. Store:
 - o notes array
 - o addNote(note) function

Notes Page

1. Fetch initial notes using useEffect (static or mock data).
2. Display notes using Bootstrap Cards.
3. Pass note data to components using props.
4. Display “No notes available” when the notes list is empty.

Conditional Rendering

1. Navbar must show Login or Logout based on authentication state.
2. Notes page must be visible only when the user is logged in.

Expected User Flow

1. Application loads and shows Login page.
2. User logs in successfully.
3. User is redirected to Notes page.
4. User adds a note.
5. Notes list updates immediately.
6. User logs out and returns to Login page.