Welcome - Session 4 Overview

Recap of Session 3 (State & Hooks)

Today’s Objectives:

- Why navigation is essential
- React Router (Web): routing principles, hooks, nested views
- React Navigation (Native): stack, tab, and drawer
- Real-life navigation examples: when to use what
- Nested routing & shared layouts
- Hook-based navigation logic

Why Navigation Matters

Breaks complex apps into logical, manageable screens

Enables modular UI: Dashboard, Profile, Settings, etc.

Key for SPAs and mobile apps to feel like native platforms

Real-life examples:

Web: /dashboard/settings/profile

Mobile: Home screen → tab bar → stack of inner pages

React Router (Web Navigation)

Documentation: https://reactrouter.com/en/main
Concepts:

- Declarative route mapping (<Route>)
- Nesting for shared layouts
- Navigation via <Link>
- Dynamic URLs with useParams
- Hook-based control with useNavigate, useLocation

Navigation in React Router

Defining routes:

```
import { BrowserRouter, Routes, Route } from 'react-router-dom';

<BrowserRouter>
  <Routes>
    <Route path="/" element={<Home />} />
    <Route path="/about" element={<About />} />
    <Route path="/contact" element={<Contact />} />
  </Routes>
</BrowserRouter>
```

Navigation in React Router

Navigating with links:

```
import { Link } from 'react-router-dom';

<nav>
  <Link to="/">Home</Link>
  <Link to="/about">About</Link>
  <Link to="/contact">Contact</Link>
</nav>
```

React Router Hooks in Depth

- useNavigate() – go to another route via code (e.g., after login)
- useLocation() – read current URL, pathname, search
- useParams() – extract values from dynamic routes
- useOutletContext() – share data across nested routes

```
const { id } = useParams();
const navigate = useNavigate();
navigate(`/user/${id}/edit`);
```

React Native Navigation Overview

Documentation: https://reactnavigation.org/

Main navigators:

Stack: linear history (e.g., Login → Profile → Details)

Tab: bottom navigation for main sections

Drawer: hidden side menu (great for settings, admin apps)

Use NavigationContainer as the root wrapper

When to Use Each Navigator

Stack: authentication flow, deep screen transitions
e.g., Product List → Product Detail → Checkout

Tab: user-facing apps with 3–5 main areas
e.g., Home, Explore, Profile, Settings

Drawer: content-heavy or utility-based navigation
e.g., Admin Panel, Side Menu, Preferences

You can combine them: Tab inside Stack or Drawer around Stack

Stack Navigation Logic (React Native)

```
<Stack.Navigator>
  <Stack.Screen name="Home" component={HomeScreen} />
  <Stack.Screen name="Details" component={DetailScreen} />
</Stack.Navigator>
```

Accessing route data:

```
const { id } = route.params;
navigation.navigate('Details', { id: 1 });
```

Tab Navigation (React Native)

```
<Tab.Navigator>
  <Tab.Screen name="Feed" component={FeedScreen} />
  <Tab.Screen name="Account" component={AccountScreen} />
</Tab.Navigator>
```

Persistent navigation for top-level sections

Automatically handles active tab highlighting

Drawer Navigation (React Native)

```
<Drawer.Navigator>
  <Drawer.Screen name="Dashboard" component={DashboardScreen} />
  <Drawer.Screen name="Preferences" component={PreferencesScreen} />
</Drawer.Navigator>
```

Common for apps with settings or admin features

Useful for hiding less frequently accessed screens

React Navigation Hooks

- useNavigation() ⟶ access navigator object
- useRoute() ⟶ get route info
- useFocusEffect() ⟶ trigger effects when screen is focused

Example:

```
const navigation = useNavigation();
navigation.setOptions({ title: 'Profile' });
```

Shared Layouts in Native & Web

React:

- Use <Outlet /> in layout components
- Wrap routes in a common layout

React Native:

- Create a wrapper screen (e.g., MainLayout) with tabs, headers
- Use conditional rendering or nested navigators

Recap & Q&A

Use React Router for web, React Navigation for mobile

Understand when to use Stack, Tab, Drawer, or Nested Routes

Hooks give full control over routing logic

Keep navigation declarative, composable, and modular