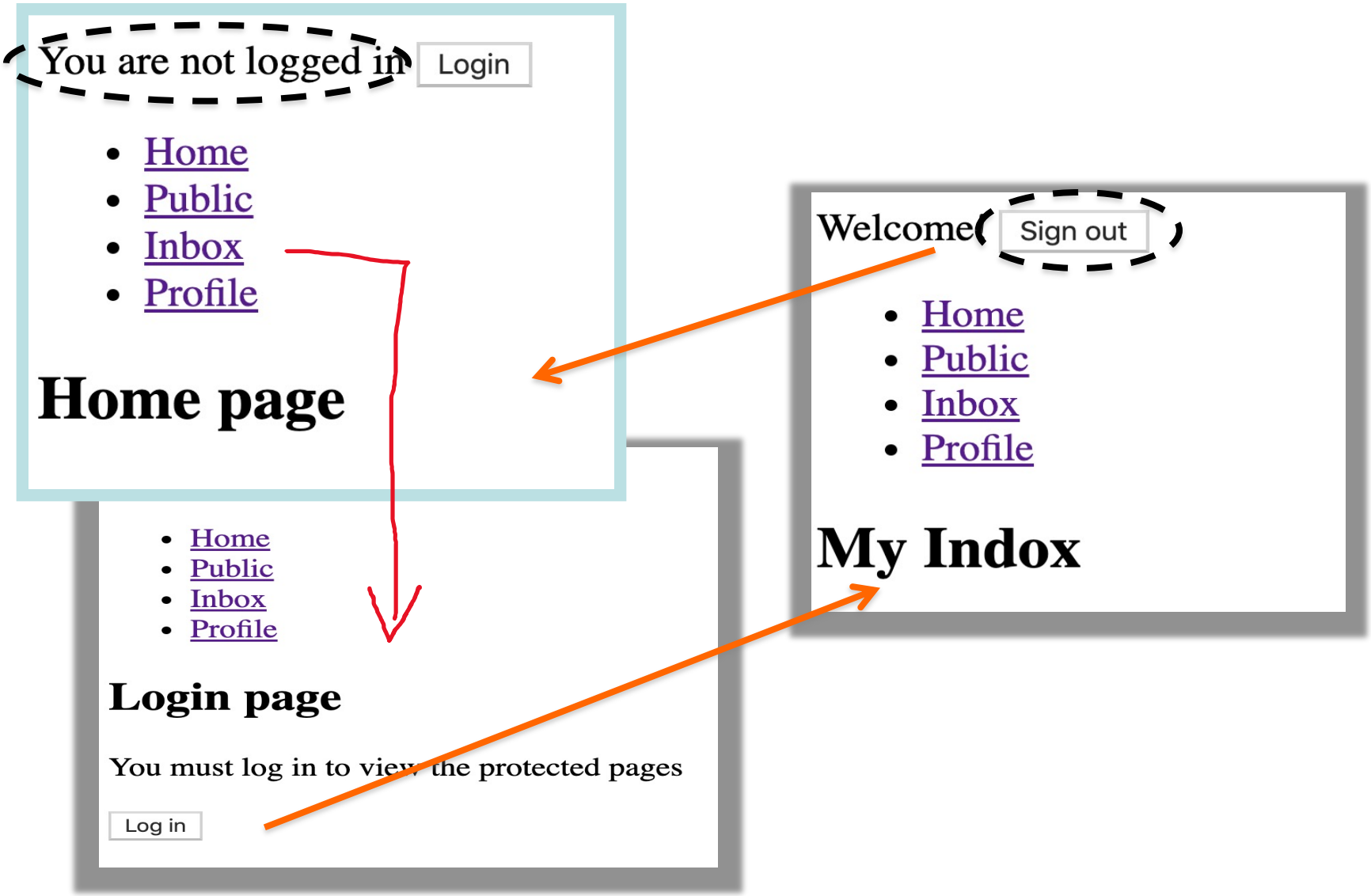
Use Case:

Authentication and
Protected/Private Routes

Objective


You are not logged in
Login
Home
Public
Inbox
Profile
Home page
Welcome
Sign out
Home
Public
Inbox
Profile
My Indox
Home
Public
Inbox
Profile
Login page
You must log in to view the protected pages
Log in

Solution outline.

- **Not native to React Router.**
- **We need a custom solution.**
- Solution outline**: A clear, declarative style for declare the views/pages that require authentication:**

```
<Routes>
  <Route path="/public" element={<PublicPage />} />
  <Route path="/login" element={<LoginPage />} />
  <Route index element={<HomePage />} />
  <Route path="/inbox"element={
      <ProtectedRoute>
        <Inbox />
      </ProtectedRoute>
    }
  />
  <Route path="/profile" element={
      <ProtectedRoute>
        <Profile />
      </ProtectedRoute>
    }
  />
  <Route path="*" element={<Navigate to="/" replace />} />
</Routes>
```

Solution elements.

- **Solution features:**

  1. **React Context to store current authenticated user token.**
  2. **Programmatic navigation - to redirect unauthenticated user to login page.**
  3. **Remember user's intent before forced authentication.**

Implementation

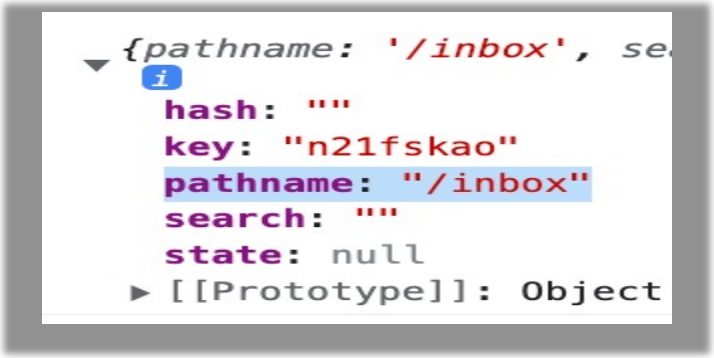
- **Solution elements: The AuthContext.**

```
 4    export const AuthContext = createContext(null);
 5
 6    const AuthContextProvider = ({ children }) => {
 7      const [token, setToken] = useState(null);
 8      const location = useLocation();
 9      const navigate = useNavigate();
10
11      const authenticate = (username, password) => {
12        setTimeout(() => {
13          setToken("2342f2f1d131rf12");
14          const origin = location.state?.intent || "/";
15          navigate(origin);
16        }, 250);
17      };
18
19  >   const signout = () => {…
22      };
23
24      return (
25        <AuthContext.Provider
26          value={{ token, authenticate, signout }}
27        >
28          {children}
29        </AuthContext.Provider>
30      );
31    };
```

Implementation

- **Solution elements (Contd.): <ProtectedRoute />**

```
<Route path="/inbox"element={
    <ProtectedRoute>
      <Inbox />
    </ProtectedRoute>
  }
/>
```




```
5   const ProtectedRoute = ({ children }) => {
6     const { token } = useContext(AuthContext);
7     const location = useLocation();
8     // console.log(location)
9     if (!token) {
10      return <Navigate to={"/login"} replace
11                       state={{ intent: location.pathname }} />;
12    }
13
14    return children;
15  };
16
```

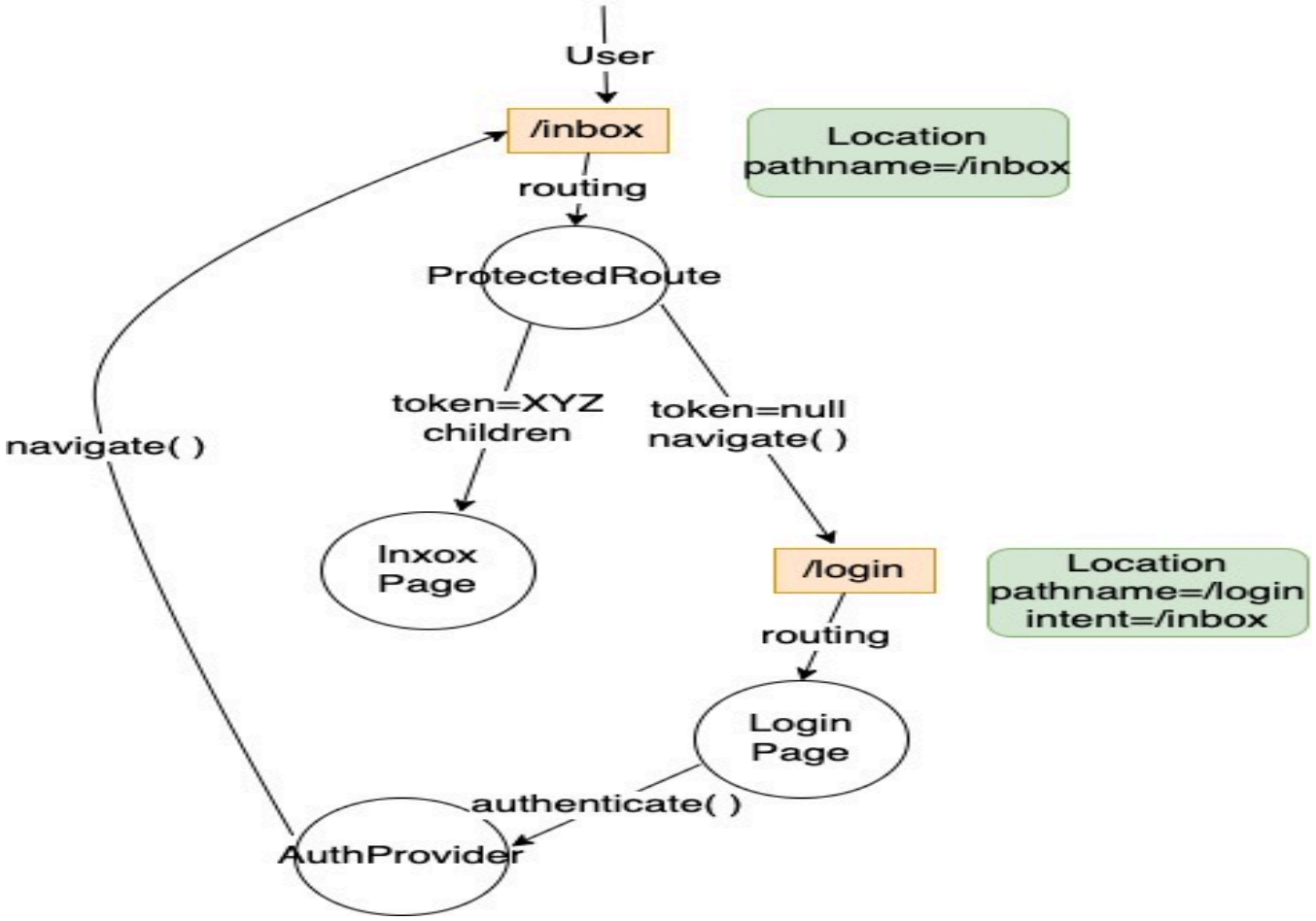
Implementation

- **Solution elements (Contd.): The Login Page.**

```
4   const LoginPage = () => {
5     const {authenticate } = useContext(AuthContext);
6
7     const login = () => {
8       const password = Math.random().toString(36).substring(7);
9       authenticate('user1', password);
10    };      You, 3 weeks ago • Upgrade to React Router v6 – Major
11
12    return (
13      <>
14        <h2>Login page</h2>
15        <p>You must log in to view the protected pages </p>
16        {/* Login web form  */}
17        <button onClick={login}>Submit</button>
18      </>
19      )
20  };
```

Implementation - Flow of control.

When an unauthenticated user tries to access /inbox



The optional chaining operator (?.)

- **The optional chaining operator (?.) accesses an object's property or calls a function. If the object is undefined or null, it returns undefined instead of throwing an error.**

```
1  let var1 = {}  // Empty object
2  let var2 = var1.foo // undefined
3  let var3 = var1.foo.bar // Runtime ERROR
4  let var4 = var1.foo?.bar // undefined
5  let var5 = var1.foo?.bar?.baz // undefined
6
7  var1 = {foo: {bar: 10}}
8  var4 = var1.foo?.bar // 10
9
```

The code archive.

- **The archive contains two versions of the use case. Version 2 uses the Promise model for handling asynchronous communication in JavaScript – used later in this module.**