Authentication

1. Create Custom Hooks for Authentication

You can create custom hooks to handle authentication logic, such as storing and retrieving user information from localStorage.

Create a Custom Hook for Authentication

Create a file named useAuth.js inside the hooks folder (you may need to create this folder if it doesn't exist):

javascript Copy code

```javascript
// hooks/useAuth.js
import { useState, useEffect } from 'react';

const AUTH_TOKEN_KEY = 'authToken';

export function useAuth() {
  const [user, setUser] = useState(null);

  useEffect(() => {
    const token = localStorage.getItem(AUTH_TOKEN_KEY);
    if (token) {
      // Assume the token contains user info or decode it to get user info
      // For example, you might use a library like jwt-decode here
      setUser({ token }); // You would set user details here
    }
  }, []);

  const login = (token) => {
    localStorage.setItem(AUTH_TOKEN_KEY, token);
    setUser({ token });
  };

  const logout = () => {
    localStorage.removeItem(AUTH_TOKEN_KEY);
    setUser(null);
  };

  return { user, login, logout };
}
```

2. Implement Login and Logout Pages

Create simple login and logout pages that use this hook.

Login Page

Create a file at pages/login.js:

javascript Copy code

```javascript
// pages/login.js
import { useState } from 'react';
import { useAuth } from '../hooks/useAuth';

export default function LoginPage() {
  const [username, setUsername] = useState('');
  const [password, setPassword] = useState('');
  const { login } = useAuth();

  const handleSubmit = async (e) => {
    e.preventDefault();
    // Replace this with your authentication logic
    const token = 'fake-jwt-token'; // This should be replaced with an actual token from y
    login(token);
    window.location.href = '/'; // Redirect after login
  };

  return (
    <form onSubmit={handleSubmit}>
      <label>
        Username:
        <input
          type="text"
          value={username}
          onChange={(e) => setUsername(e.target.value)}
        />
      </label>
      <label>
        Password:
        <input
          type="password"
          value={password}
          onChange={(e) => setPassword(e.target.value)}
        />
      </label>
      <button type="submit">Login</button>
    </form>
  );
}
```

Logout Page

Create a file at pages/logout.js:

```javascript
// pages/logout.js
import { useEffect } from 'react';
import { useAuth } from '../hooks/useAuth';

export default function LogoutPage() {
  const { logout } = useAuth();

  useEffect(() => {
    logout();
    window.location.href = '/'; // Redirect after logout
  }, [logout]);

  return <p>Logging out...</p>;
}
```

3. Protect Routes

To protect routes, you can use the useAuth hook in your components to check if a user is logged in.

Protected Page

Create a file at pages/protected.js:

```javascript
// pages/protected.js
import { useAuth } from '../hooks/useAuth';
import { useEffect } from 'react';

export default function ProtectedPage() {
  const { user } = useAuth();

  useEffect(() => {
    if (!user) {
      window.location.href = '/login'; // Redirect to login if not authenticated
    }
  }, [user]);

  if (!user) {
    return <p>Loading...</p>; // You might want to show a loading spinner or message here
  }

  return (
    <div>
      <h1>Protected Page</h1>
      <p>Welcome, you are logged in!</p>
    </div>
  );
}
```


4. Add Authentication Check in _app.js (Optionally)

To apply authentication checks across all pages, you can add logic to _app.js.

Modify _app.js

```javascript
// pages/_app.js
import { useAuth } from '../hooks/useAuth';

function MyApp({ Component, pageProps }) {
  const { user } = useAuth();

  return (
    <Component {...pageProps} />
  );
}

export default MyApp;
```

Summary

1. **Create a custom hook useAuth** to manage authentication with localStorage.
2. **Build login and logout pages** that utilize the useAuth hook.
3. **Protect routes** by checking authentication status using the useAuth hook.
4. **Optionally, apply authentication logic** globally in _app.js if needed.