Module 6: Security-OAuth2

🚀 Overview

- What is OAuth2?
- OAuth2 Roles
- OAuth2 Flow with GitHub

Create a frontend React app to test the API

- The React app will have a simple UI for
  - user registration (using local database)
  - user login
  - displaying all users data
  - deleting a user by ID

What is OAuth2?

- An **authorization framework** that enables applications to obtain limited access to user accounts on an HTTP service
- **Delegated access:** Users can authorize third-party apps to access their resources without sharing credentials
- Commonly used for "Login with Google/GitHub/Facebook" features

OAuth2 Roles

- **Resource Owner:** The **user** who authorizes an application to access their resources.
- **Client:** The application (*frontend*) requesting access to the user's resources.
- **Authorization Server:** The server that authenticates the user.
- **Resource Server:**
  - *GitHub OAuth2 Server* acts as the resource server for user information
  - *Backend API* is a resource server if it provides protected endpoints that require access tokens

This OAuth2 Flow

- **GitHub** plays a dual role as both the Authorization Server (issuing access tokens) and Resource Server (providing user info)
- Your **Backend API** acts as an intermediary that exchanges tokens and then becomes a resource server for your application's protected endpoints

Implement OAuth2 with GitHub

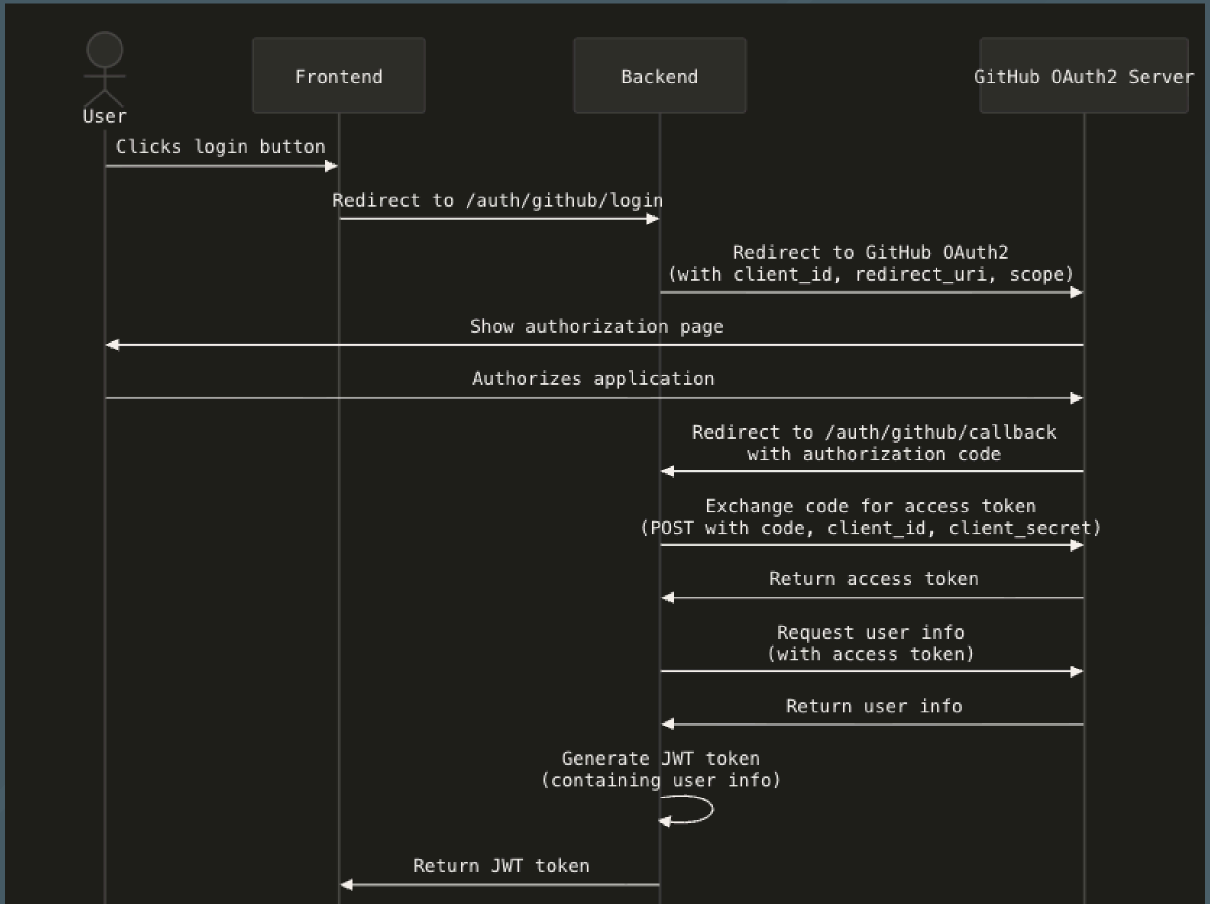
- Register OAuth app on GitHub
- Add GitHub OAuth endpoints to FastAPI
- Exchange code for token, get user info
- Match or create user in your DB
- Generate your app’s JWT token
- Redirect to frontend with token
- Store token in frontend and use it like local login

Register Your App with GitHub

- Go to: https://github.com/settings/developers
- Click "New OAuth App"
- Set:
  - Application name
  - Homepage URL: http://your-frontend.com
  - Authorization callback URL: http://your-backend.com/auth/github/callback
- GitHub gives you: `CLIENT_ID` and `CLIENT_SECRET`

Github OAuth2 login flow

GitHub OAuth2 flow:


User
Frontend
Backend
GitHub OAuth2 Server
Clicks login button
Redirect to /auth/github/login
Redirect to GitHub OAuth2 (with client_id, redirect_uri, scope)
Show authorization page
Authorizes application
Redirect to /auth/github/callback with authorization code
Exchange code for access token (POST with code, client_id, client_secret)
Return access token
Request user info (with access token)
Return user info
Generate JWT token (containing user info)
Return JWT token

OAuth2 Flow with GitHub

- **Frontend** button click --> Redirects to **Backend** (`/auth/github/login`)
- **Backend** (`/auth/github/login`) --> Redirects to **GitHub OAuth2 Server**
- **GitHub OAuth2 Server**: User authorizes --> Redirects to **Backend** with authorization code (`/auth/github/callback?code=...`)
- **Backend** exchanges `code` for `access token` from **GitHub**
- **Backend** retrieves user info from **GitHub** using `access token`
- **Backend** issues `JWT token` containing user info
- **Frontend** receives `JWT token` and stores it in local storage

GitHub OAuth2 flow

Link: https://claude.ai/public/artifacts/c4114f4b-b87a-4acf-adf6-37426cb9419b

Code Example: Backend - auth.py

- Endpoints

```
GET /auth/github/login
Redirects to GitHub's OAuth consent page.
```

```
GET /auth/github/callback
GitHub sends users here after login with a code.
```

- Utility function

```
get_or_create_user()
create or match a GitHub-authenticated user in your DB.
```

Code Example: Frontend - App.jsx

- Add a `Login with GitHub` Button

```
<a href="https://your-backend.com/auth/github/login">
  <button>Login with GitHub</button>
</a>
```

- Handle the Redirect (JWT Token)
  Create a page like `/oauth/callback` to extract the `JWT token` from the URL and store it

Homework

Link to homework

Section: **Practical exercises**

🎯 Remember

- What is OAuth2?
- OAuth2 Roles
- OAuth2 Flow with GitHub