OAuth2 Authorization Server – Project Documentation

1. Setup: Django OAuth Toolkit as Authorization Server

- Added `oauth2_provider` to INSTALLED_APPS
- Added `o/` routes in urls.py
- Run migrations to create tables
- Management command `create_oauth_app` provisions default client

2. Implemented OAuth2 Flows

- Authorization Code Grant with PKCE implemented
- Refresh Token flow supported
- Confidential clients can exchange tokens via backend

3. Token-Based Authentication

- Tokens issued at `/o/token/` include access & refresh tokens
- Expiry configurable in `.env`
- Refresh tokens rotated for security

4. Secure Sessions & User Roles

- Custom User model extends AbstractUser with `role`
- Sessions secured via CSRF, HSTS, and secure cookies
- Roles API available

5. Provided APIs

- /o/token/: Token issuance
- /api/userinfo/: Get user info
- /api/logout/: Logout user
- /api/roles/: Fetch roles
- /api/validate-token/: Validate token

6. Setup & Client Integration

Server setup:
git clone
cd final
python -m venv .venv
source .venv/bin/activate
pip install -r requirements.txt
python manage.py migrate

python manage.py seed_demo_users
python manage.py create_oauth_app
python manage.py runserver

Demo users:
admin / adminpass
alice / alicepass

React client:
cd react-client
npm install
npm start

7. Security Best Practices

- PKCE enforced for SPA clients
- HTTPS recommended in production
- Rotate refresh tokens enabled
- Secrets managed via .env
- CSRF middleware enabled

Project File Structure

```
Oauth-Project/
  auth_server/
    settings.py
    urls.py
    wsgi.py
  users/
    models.py
    views.py
    management/commands/create_oauth_app.py
  client_backend/
    views.py
  react-client/
    src/
      App.js
      config.js
  manage.py
  requirements.txt
```

Example: React Config.js

```
export const OAUTH_CLIENT_ID = "your_client_id_here";
export const REDIRECT_URI = "http://localhost:3000/callback";
export const AUTH_SERVER_URL = "http://localhost:8000";
```