Scopio Backend Overview (Django + Postgres)

This doc summarizes how the backend works, local setup details (localhost), and what to change when deploying.

Project Summary

- Framework: Django + Django REST Framework
- Database: PostgreSQL (Docker)
- Auth: JWT (SimpleJWT) with token blacklist; Google OAuth (Django Allauth)
- Admin UI: pgAdmin (Docker) for DB browsing

Django Details (What's Implemented)

- Installed apps (see Backend/main/settings.py):
  - `rest_framework`
  - `corsheaders`
  - `allauth`, `allauth.account`, `allauth.socialaccount`, `allauth.socialaccount.providers.google`
  - `rest_framework_simplejwt.token_blacklist`
  - your project apps: `api`, `glogin`, etc.
- REST framework auth/permissions:
  - `JWTAuthentication` is enabled; default permission `IsAuthenticated` (see `REST_FRAMEWORK` in Backend/main/settings.py)
- Token blacklist:
  - Enabled via `rest_framework_simplejwt.token_blacklist` and migrations (DB tables `token_blacklist_*`).
- Allauth + Google:
  - `SITE_ID = 1` and Google provider configured in `SOCIALACCOUNT_PROVIDERS` (see Backend/main/settings.py).
- URLs & Apps:
  - Project URLs: Backend/main/urls.py
  - API app: endpoints in Backend/api/urls.py, logic in Backend/api/views.py, serializers in Backend/api/serializers.py, models in Backend/api/models.py
  - Google auth flows: Backend/glogin

Simple Flows

- Login: backend returns access (short-lived) + refresh (longer-lived) tokens.
- Authenticated calls: frontend sends `Authorization: Bearer <access_token>`.
- Refresh: when access expires, frontend uses refresh token to get a new access.
- Logout/rotation: old refresh tokens are blacklisted; can't be reused.
- Google OAuth: user consents at Google; backend verifies and issues same JWT pair.

Database & pgAdmin Details

- Compose services: see docker-compose.yaml
    - `postgres_db` (Postgres 16) with healthcheck
    - `pgadmin4` (depends on Postgres health)
    - Volumes: `postgres_data` (DB), `pgadmin_data` (pgAdmin state)
    - Server auto-config: pgadmin/servers.json
- Connections from your computer (host):
    - Host: `localhost`
    - Port: `5432`
    - Database: `scopio`
    - User: `admin`
    - Password: `Vishalashvafalin`
- pgAdmin (container UI): http://localhost:5050
    - Login: email `scopioedutech@gmail.com`, password `Vishalashvafalin`
    - Inside pgAdmin, use Host: `postgres_db` (Docker network hostname), Port: `5432`, DB: `scopio`, User: `admin`, Password: `Vishalashvafalin`
- Databases you'll see:
    - `postgres`: default maintenance DB created by Postgres
    - `scopio`: application DB used by Django (tables in `public` schema)

Local Development (localhost)

- Start services:

```
cd D:\Scopio
docker compose up -d
```

- Apply migrations:

```
cd D:\Scopio\Backend
..\benv\Scripts\python.exe manage.py migrate
```

- Run dev server (http://127.0.0.1:8000/):

```
..\benv\Scripts\python.exe manage.py runserver
```

Useful Diagnostics

- DB readiness:

```
docker exec postgres_db pg_isready -U admin -d scopio
```

- List tables:

```
docker exec postgres_db psql -U admin -d scopio -c "\\dt"
```

- Check host port (PowerShell):

```
Test-NetConnection localhost -Port 5432
```

Deployment Changes (what to change later)

- Database host:
  - Local: `DB_HOST=localhost`
  - Deploy: set `DB_HOST` to your managed DB host/IP (port likely `5432`)
- Secrets & config to environment:
  - `SECRET_KEY`, DB creds, OAuth keys, CORS, token lifetimes (read from env in Backend/main/settings.py)
- Security:
  - `DEBUG=False`; set `ALLOWED_HOSTS` to your domain(s)
  - `CORS_ALLOWED_ORIGINS` to your frontend domain(s)
  - HTTPS in production; prefer HttpOnly, Secure cookies for refresh tokens
- Server process & static files:
  - Use Gunicorn/Uvicorn+Daphne behind Nginx/Apache
  - Run `collectstatic` and serve static assets via web server or storage
- Admin tools:
  - Keep pgAdmin private; don't expose DB admin UI publicly

Where to Look in Code (quick links)

- Settings & auth: Backend/main/settings.py
- URLs: Backend/main/urls.py
- API: Backend/api → models, serializers, views, urls
- Google OAuth: Backend/glogin