Task 7: Role-Based Project Access & Production Deployment

Objective

This task focuses on advanced role-based access control, user onboarding flows, and production deployment of a Django project.

By completing this task, you will learn how to:

- Design multiple user roles with different permission levels inside a project
- Control what users can see and do based on their role and verification status
- Implement post-registration data completion workflows
- Promote users between roles using Django Admin
- Deploy a Django application in a production-like environment using Gunicorn + Nginx
- Understand how WSGI servers work and how Django communicates with Nginx

This task extends the project from Tasks 5 and 6, but introduces multi-level roles inside projects, not just global permissions.

Project Concept

You are building a Project Management Platform where users participate in projects with different roles and trust levels.

Not all developers are equal:

- Some can only view projects
- Some are verified/authorized and can create new projects
- Admins can manage everything

Questions to Explore

(Permissions have covered in previous tasks. This task is more practical mini project)

Deployment & Production

1. What is WSGI and why does Django need it?
2. What problem do Gunicorn and uWSGI solve?

3. What is the role of Nginx in front of Gunicorn?
4. Why should Django not serve static files in production?
5. What responsibilities belong to:
    a. Django
    b. Gunicorn
    c. Nginx
6. What security and performance benefits does this architecture provide?

Roles & Permission Levels

You must implement 4 roles, for example

Required Roles

- Admin
- verified Developer
- Developer
- Customer

they should be implemented by roles/groups. for example, do not just use a Boolean field for verified Developer.

In registration you should ask them that they want to be customer or developer.

The exact naming is flexible. The behavior matters, not the name.

Permission Rules

- **Admin**
    - Full access to all projects
    - Change user roles (via Django Admin if it is hard)
    - View and edit all profiles
- **Authorized Developer**
    - Can see list of all developers
    - Create new projects
    - Edit their own projects

    - View all projects
- **Developer**
    - see list of all developers
    - View all projects
    - Can not edit anything
    - Can only have one project.
    - Must complete extra profile information to unlock more permissions
- **Customer**
    - See list of developers
    - View project list only

Note: By see list I mean list of the objects and their detail view not just simple list.

Deliverables

Authentication & Registration

- User registration and login must be implemented
- At first developers considered unverified until they **Complete Your Profile and confirm they phone number (fake input).**
- Registration must create:
    - User
    - Profile (automatically)
    - Default role/group

  You may want to use signals!?

Profile Completion & Verification

- Extend the Profile model with additional required fields (e.g.:
    - national_id / brief description/ profile image)

- Developers cannot access certain features until profile is completed
- In real word, normal users should be verified to unlock some features but we want to keep it simple so normal users just can see list of things.
- The phone verification for developers should implement correctly but instead of sending the code print it in the console. (use random number and save it somewhere. You may want to use redis !?)

Views and models:

Add some filters to developer and project list:

- Can filter based on developer
- Can filter based on project
- Can search specific username
- Can search specific project name

Models like project should follow previous tasks instructions.

Deployment (Mandatory)

You must deploy the project on a Linux VM using:

- Gunicorn as the WSGI server
- Nginx as the reverse proxy
- Static and media files served by Nginx (all statics should be loaded correctly)
- Django running with DEBUG = False
- Provide:
  - gunicorn service file
  - Final Nginx configuration file connected to your project

Bonus (Optional)

- Add audit logging for project

Evaluation Criteria

- Correct role-based permission enforcement
- Clean separation of concerns
- Proper use of Django Groups & Permissions
- Secure onboarding flow
- Correct production deployment architecture
- Understanding why, not just how
