Approach Used

Objective

The goal of the project is to build a secure, role-based backend system where:

- **OPS users** can upload files to the system
- **CLIENT users** can download those files — but only through an encrypted, secure download link
- All user actions are protected by **authentication tokens**

Development Approach

1. Backend Setup

The project was built using the Django framework with Django REST Framework (DRF) for API creation. A virtual environment was set up to manage dependencies and isolate the project environment.

2. User Role Management

A custom user model was used to introduce role-based functionality. Each user, upon registration, is assigned a role: either **OPS** or **CLIENT**. This role determines what actions they can perform in the system.

3. Authentication System

Djoser was integrated to handle user registration, token-based login, and account activation. Email-based account activation was simulated using Django's console email backend. Authentication tokens were used to protect all APIs, ensuring only authorized users could access protected endpoints.

4. File Upload (OPS Role)

OPS users are allowed to upload files. A dedicated file upload endpoint was built, restricted to authenticated users with the OPS role. Each file is stored with a reference to the user who uploaded it and the upload timestamp.

5. Secure Link Generation (CLIENT Role)

CLIENT users are not allowed to access files directly. Instead, they must request a **signed download link** for a specific file. This link is generated using a cryptographic signing method that ensures the file ID cannot be tampered with.

6. File Download with Access Control

Using the signed URL, a CLIENT user can download the requested file. The system verifies the link's authenticity and ensures that only a CLIENT user can access it. Any tampering or role mismatch results in access being denied.

7. API Routing and Access Control

API endpoints were organized based on functionality (authentication, upload, download). Permission checks were implemented in each view to restrict access based on user role and authentication status.

8. Testing with Postman

All API endpoints were thoroughly tested using Postman. This included:

- Registering users with different roles
- Activating accounts
- Logging in and generating tokens
- Uploading files (OPS)
- Generating download links (CLIENT)
- Downloading files securely using those links

A Postman collection was prepared for submission.

Thank You!