Deployment Plan for Production (Professional Tone)

I would like to outline the steps I plan to follow to deploy the File-Share application to a secure and scalable production environment:

---

1. Environment Setup

I will deploy the application on a cloud-based Linux server such as **AWS EC2**, **Render**, or **DigitalOcean**, using **Ubuntu 22.04 LTS**. This server will be secured and configured with necessary software, including:

- Python 3.x and pip
- Virtual environment for dependency isolation
- Gunicorn as the WSGI HTTP server
- Nginx as a reverse proxy
- PostgreSQL (or another preferred database system)

---

2. Secure Code Deployment

I will pull the latest version of the code from GitHub using:

git clone https://github.com/Varhneyprachi/Secure-File-system.git

Inside the server, I will set up a Python virtual environment and install all dependencies using:

python3 -m venv venv

source venv/bin/activate

pip install -r requirements.txt

---

3. Configuration Management

Sensitive credentials like database URLs, secret keys, and JWT tokens will be stored in a **.env file** and securely loaded into the application using libraries like python-dotenv.

---

4. Application Deployment

The app will be launched using **Gunicorn**, ensuring high performance in a production setting:

bash

CopyEdit

gunicorn app:app --bind 0.0.0.0:8000

Nginx will be configured as a reverse proxy to handle incoming HTTP/HTTPS requests and forward them to Gunicorn. SSL will be enabled using **Let's Encrypt** for secure HTTPS communication.

---

5. File Security

Uploaded files will be stored in a non-public directory outside the root path, ensuring they are not directly accessible without proper authentication. Only authorized Ops Users will be able to upload files, and file types will be strictly validated (e.g., .docx, .pptx, .xlsx).

---

6. Continuous Integration and Deployment (CI/CD)

To ensure efficient deployment and consistency, I will configure **GitHub Actions** for CI/CD. This pipeline will:

- Run tests and linting on every push
- Deploy automatically to the production server on successful build
- Notify on build/deploy failures

---

7. Monitoring and Logging

To maintain system health and track errors, I will integrate:

- **Logging** using standard Python logging with rotation
- **Monitoring** using services like UptimeRobot or Prometheus
- **Crash/Error tracking** using tools like **Sentry** or similar

---

8. Docker (Optional for Portability)

For easier replication and team collaboration, I can containerize the application using **Docker** and manage services with **Docker Compose**, which is especially useful for local testing and staging.

This approach ensures the system is **secure**, **scalable**, and **easy to maintain** in a production environment. I would be happy to adjust or enhance the deployment strategy based on specific company infrastructure or DevOps preferences.