FUTURE_CS_03 — Secure File Upload/Download Portal with AES Encryption

Objective

Build a secure web portal to allow encrypted file upload and download using AES-256 encryption. The main goal is to preserve file confidentiality during both storage and transmission.

Tech Stack and Tools Used

- Python Flask (backend framework)
- AES-256 CBC Mode using PyCryptodome
- Bootstrap 5 (frontend UI)
- Git & GitHub (version control)
- Render.com (for live deployment)

Security Features Implemented

- AES Encryption (256-bit)
- CBC Mode with random IV per file
- PKCS#7 Padding for block completion
- Environment Variable used to store encryption key securely
- Encrypted storage of files in uploads/
- Decryption on download only, stored temporarily in `decrypted/`
- HTTPS deployment on Render

How to Run-Commands

git clone https://github.com/sachinsree47/FUTURE_CS_03.git

cd FUTURE_CS_03

pip install -r requirements.txt

python app.py

Then go to 👉 **http://localhost:5000**

You'll be able to upload any file → it gets encrypted → stored → downloadable → decrypted only when needed.

Learnings and Key Outcomes

- Real-world application of AES cryptography
- Key management via environment variables
- Flask routing, secure file handling, folder isolation
- Deployed production-grade app on Render
- Secure development with OWASP best practices

Limitations & Future Enhancements

- No user login/authentication
- No file type/size filtering
- No expiration on stored files

Recommended Upgrades:

- Add login & role-based access
- Auto-deletion cron jobs
- Validate file extensions & size
- Encrypted filename obfuscation
- Log activity for audit trail

Conclusion

This internship task blends cryptography, secure programming, and full-stack development in one project. It helped build a secure application while understanding the real-world importance of encryption, key secrecy, and ethical securedesign.

Prepared by : Safik Rahman