Secure File Sharing System with AES Encryption

Name: Girija .S

Internship Domain: Cybersecurity

Internship at:[Future interns]

Date of Submission:[13-07-2025]

🔗 GitHub Repo link: https://github.com/girija031-s/FUTURE\_CS\_03.git

1 **Introduction**

In a world where data exchange is constant, securing sensitive information during transmission has become a top priority. Unauthorized access, data interception, and file tampering are serious concerns in file sharing workflows.

This internship project addresses those threats by designing and developing a **Flask-based secure file sharing system**. The platform ensures:

* 🔐 Confidentiality via AES encryption
* ⚙️ Controlled file handling through upload filtering and routing
* 📊 Traceability via operation logs and timestamps
* 🖥️ User-friendly interface with styling elements

The system was built for educational and practical use cases, demonstrating backend integration with strong encryption and real-time event tracking.

2 **Objective**

- ✅ Build a Flask-based backend for secure file sharing

- 🔐 Implement AES encryption for confidentiality

- 🔍 Log all file operations for forensic traceability

- 🛡 Apply key management and audit principles

3 Technologies Used

|  |  |
| --- | --- |
| Technology | Purpose |
| Flask | Python web framework for routing and UI rendering |
| PyCryptodome | AES encryption and decryption (CFB mode) |
| HTML & CSS | Frontend structure and styling |
| Git & GitHub | Version control, remote hosting of codebase |
| Postman / cURL | Testing backend endpoints |

4 Core Functional Features

system was built to handle encryption-based file transfer with care, precision, and usability:

* 🗂️ **File Upload Interface**
  + Supports secure upload of .txt, .pdf, .jpg, .png
  + Files are filtered for safe types before processing
* 🔒 **AES-CFB Encryption**
  + Uploaded files encrypted with AES using per-file IVs
  + AES key stored in .env file and excluded from version control
* 📥 **Secure Download Mechanism**
  + Decryption route built with Flask to restore original content
  + Users can download decrypted files via browser
* 🧾 **Audit Logging with event\_log.csv**
  + Every action (upload/download) is logged with timestamp, filename, status
  + Logs aid in monitoring and future forensic traceability
* 🎨 **Styled User Interface**
  + Footer branding and clean layout

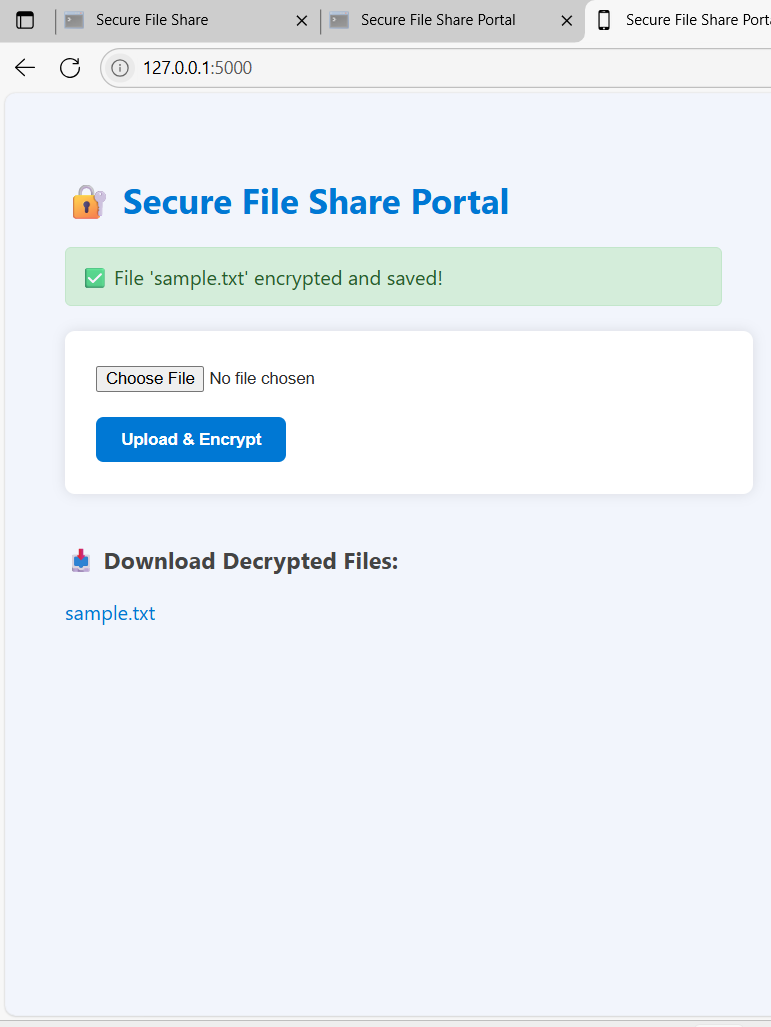
🔐 5 Encryption Methodology

To ensure data confidentiality and prevent unauthorized file access, the system uses the **Advanced Encryption Standard (AES)** in **Cipher Feedback (CFB)** mode:

* **AES-CFB Mode**:
  + Stream-like encryption for variable-length files
  + IV (Initialization Vector) generated dynamically per upload
  + AES key stored securely in .env, never hardcoded
* **Key Management**:
  + Key loaded via os.getenv() at runtime
  + .env file is excluded from GitHub through .gitignore
* **Encrypted File Handling**:
* Encrypted files given a .enc extension
* Stored in the uploads/ directory
* Original file names retained in metadata (if extended)

🧭 6 Security Measures & Controls

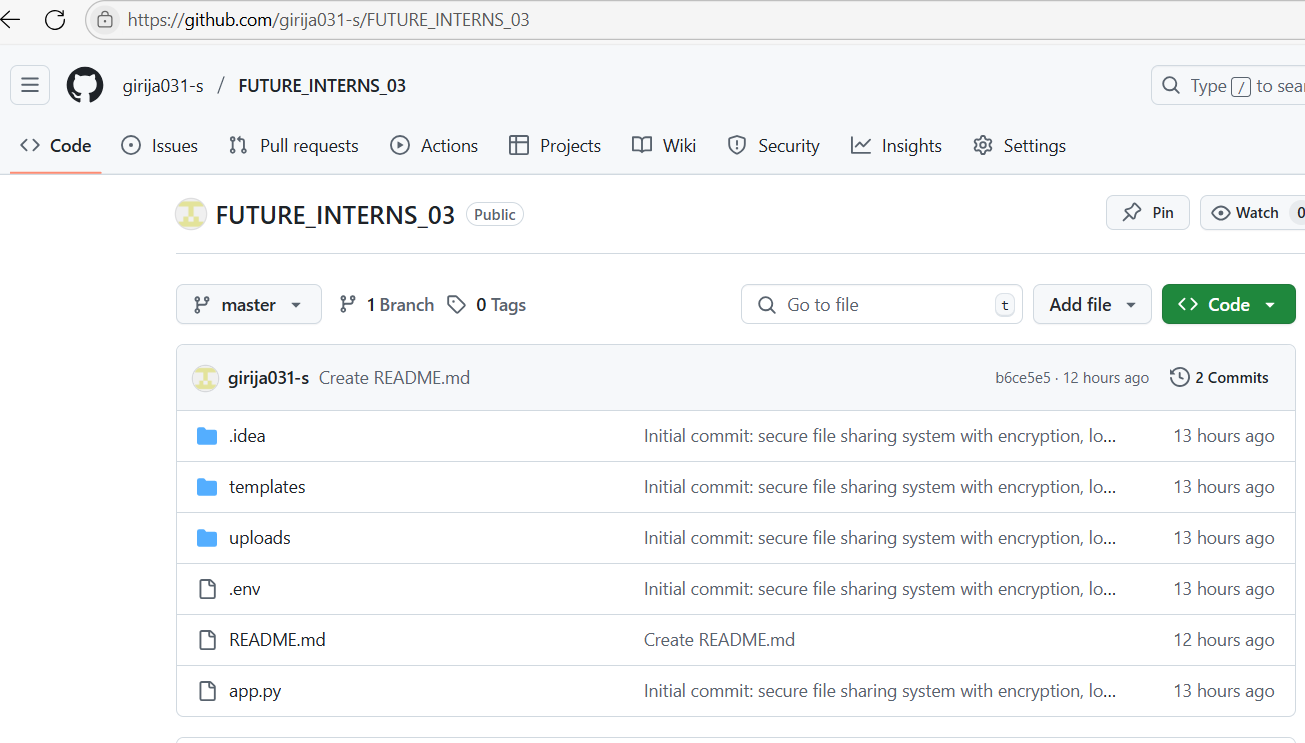
* **Upload Filtering**  
  Only .txt, .pdf, .png, and .jpg files are accepted—reducing malware risk
* **.gitignore Configuration**  
  Blocks .env, event\_log.csv, and uploads/ directory from being committed to GitHub



**Decrypted File Ready for Download via Secure Flask Route**

🔗 7 GitHub Repository Overview

secure file sharing system is version-controlled using Git and hosted publicly on GitHub for transparency, collaboration, and professional presentation.

* 🟢 **Repository Name**: FUTURE\_CS\_03
* 🔗 **GitHub Link**: https://github.com/girija031-s/FUTURE\_INTERNS\_03.git 
* 📦 **Hosting**: All project files including code, templates, logs, and documentation
* 🧾 **README.md**: Includes encryption overview, usage instructions, author details
* 🔐 **Security First**: Sensitive directories (like uploads/ and .env) are protected via .gitignore
* 📄 **Documentation**: Everything needed to replicate or review the project is hosted clearly

🧱 8 Folder & File Structure

The project follows a clean and modular layout:

secure\_file\_share

├── app.py # Main Flask application with routes

├── templates/

│ └── index.html # Upload/download UI interface

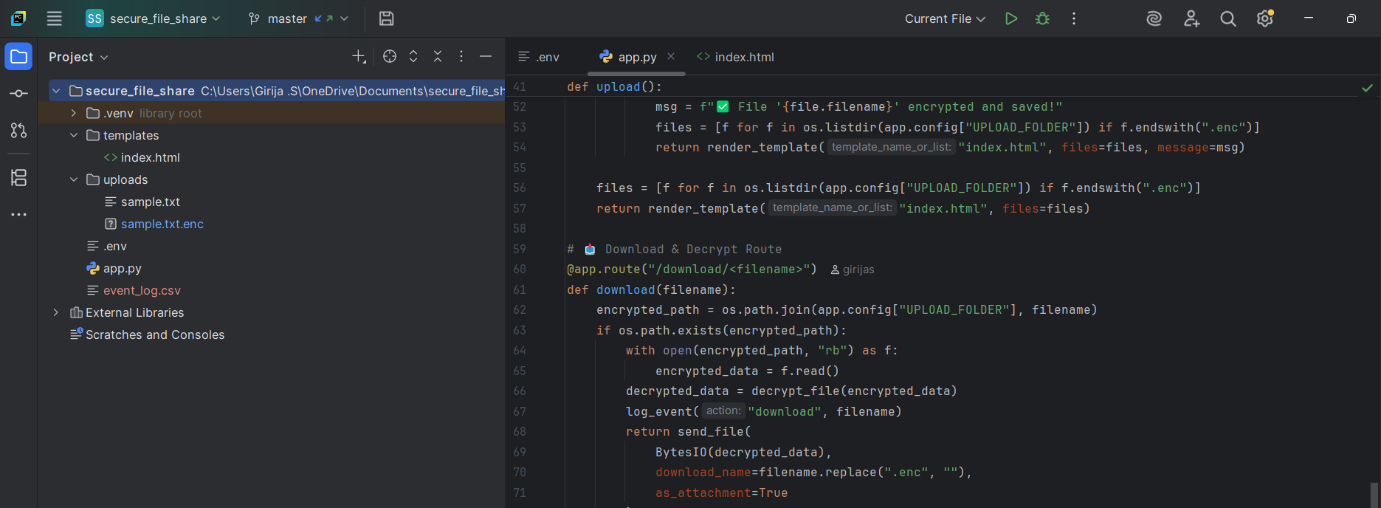
├── uploads/ # Encrypted files (excluded via .gitignore)

├── event\_log.csv # Logs each file action with timestamp

├── .env # Stores AES encryption key (excluded)

├── .gitignore # Prevents sensitive files from uploading

├── README.md # Project summary and instructions



PyCharm or GitHub view of the full folder layout

✅ **Modular Design**: Backend and frontend elements are separated

🔐 **Security First**: Sensitive directories (like uploads/ and .env) are protected via .gitignore

🎯 9 Project Outcome

This project successfully delivers a functional and secure file sharing system that uses AES encryption to protect uploaded content. Key achievements include:

* Developed a Flask-based upload and download portal
* Integrated AES-CFB encryption and safe decryption routing
* Captured activity logs using event\_log.csv for audit and traceability
* Implemented UI styling and hosted the codebase on GitHub with full documentation

The result is a scalable, professional-grade backend system for securely exchanging files—suitable for academic use, demos, or further development.

🔍 10 Personal Learnings

Through Task 3, I deepened my understanding of:

* Cryptographic techniques using Python and PyCryptodome
* Secure key handling and .env management
* Routing logic in Flask for encrypted file streaming
* Debugging Python applications and resolving Git conflicts
* Structuring technical reports and presenting secure workflows