🔐 Secure File Sharing System – Project Report

Internship Task: Task 3 – Secure File Sharing System
Platform: VS CODE

Investigation Task Number: 3

TASK Code: FUTURE_CS_03

Client / Organization: FUTURE INTERNS

CIN ID (Cyber Inspection Number): FIT/OCT25/CS4249

Submission Date: 26/11/2025

Prepared By: Jeetesh Maurya

1. Project Overview

Secure File Sharing System ka main objective hai users ko ek **safe web portal** provide karna, jahan wo **confidential files upload aur download** kar sake securely.

Key features:

- Files encrypted using **AES-128** before storage.
- Encrypted files stored in **encrypted_files/** folder.
- Decrypted downloads stored in **downloads/** folder.
- Flash messages for **upload & download status**.
- Modern and user-friendly **UI for upload, download, and file listing**.
- Simulates real-world secure file sharing scenario (corporate, healthcare, legal).

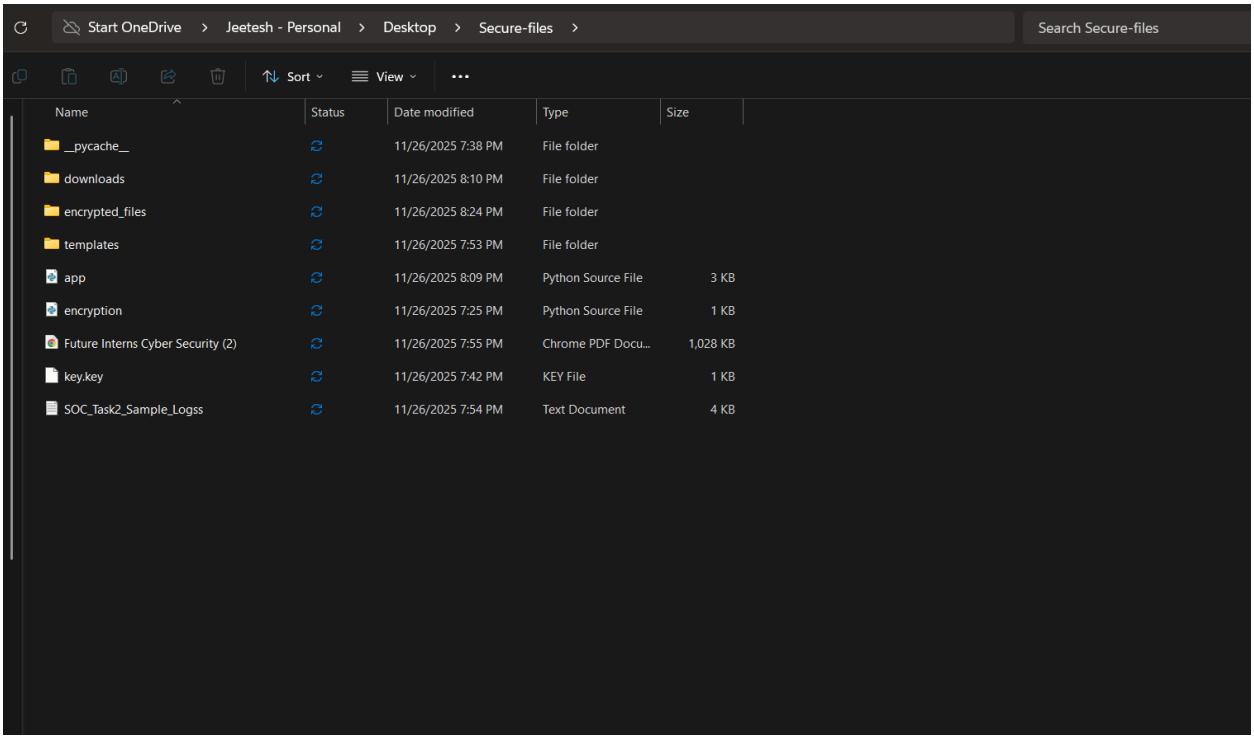
2. Tools & Technologies Used

Tool / Library	Purpose
Python 3.13	Backend development
Flask	Web framework
PyCryptodome	AES encryption & decryption
HTML / CSS / JS	Frontend for web portal
Git & GitHub	Version control and collaboration
Browser (Chrome/Edge)	Testing & UI interaction

3. Folder Structure

Secure-files/

```
|
├── app.py              # Main Flask application
├── encryption.py       # AES encryption & decryption functions
├── key.key             # AES key (auto-generated)
├── encrypted_files/    # Folder to store encrypted files
├── downloads/          # Folder to store decrypted downloaded files
└── templates/
    ├── upload.html     # Upload & download UI
    └── files_list.html # Table view of uploaded files with download buttons
```

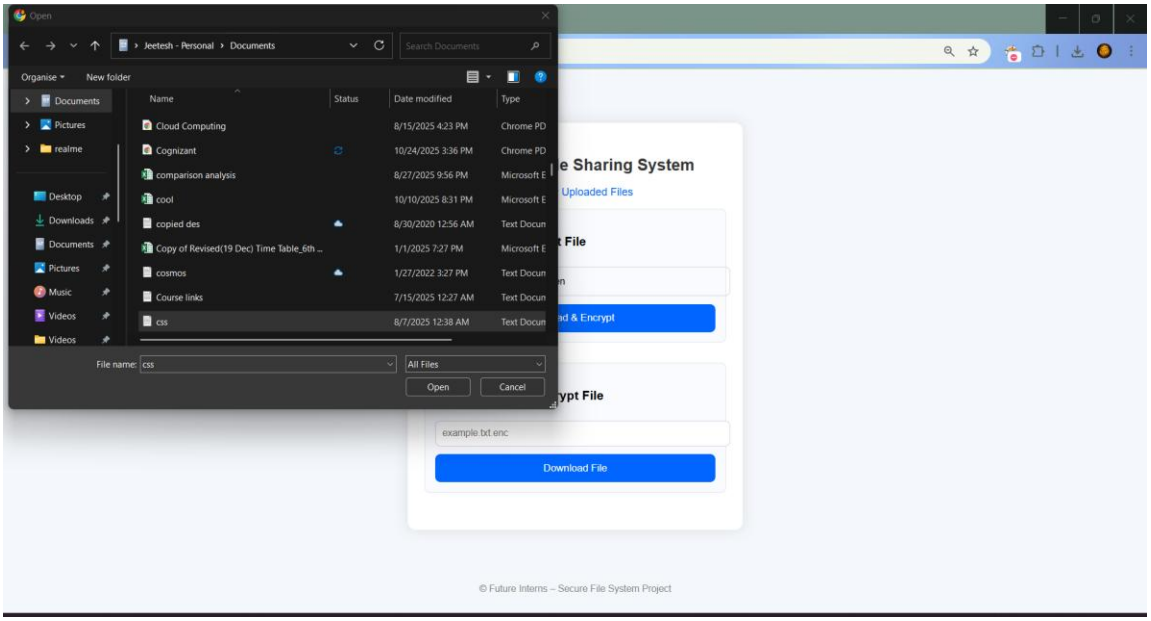


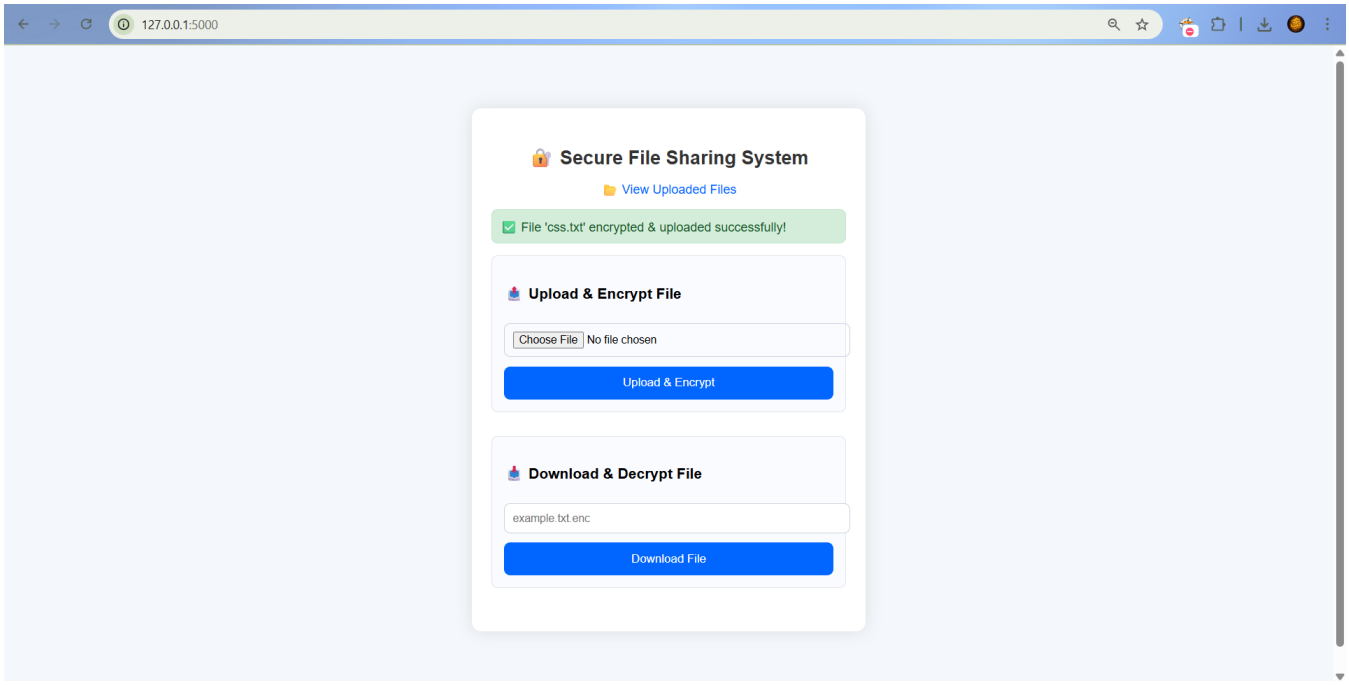
4. Functional Workflow

4.1 Upload & Encrypt

1. User selects a file and clicks **Upload & Encrypt**.
2. Flask reads file bytes → encrypts using **AES EAX mode**.
3. Encrypted file saved in **encrypted_files/** folder with .enc extension.
4. Flash message displayed on same page:

   File 'filename' encrypted & uploaded successfully.



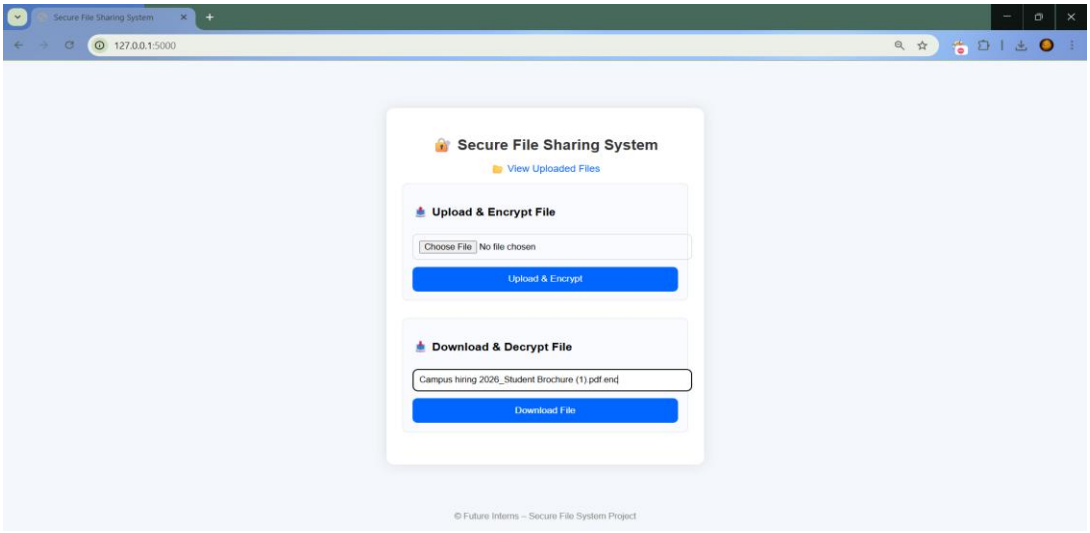


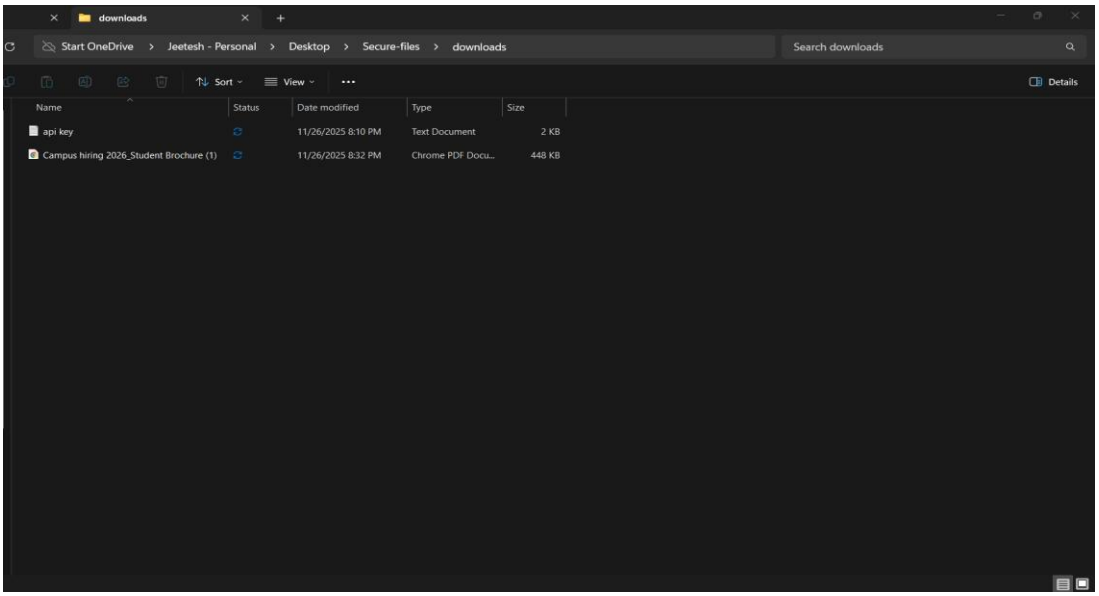
4.2 Download & Decrypt

1. User clicks download button (from files list or input field).
2. Flask reads encrypted file → decrypts using AES key.
3. Decrypted file saved in **downloads/** folder.
4. Flash message displayed:

   File 'filename' decrypted & downloaded successfully.

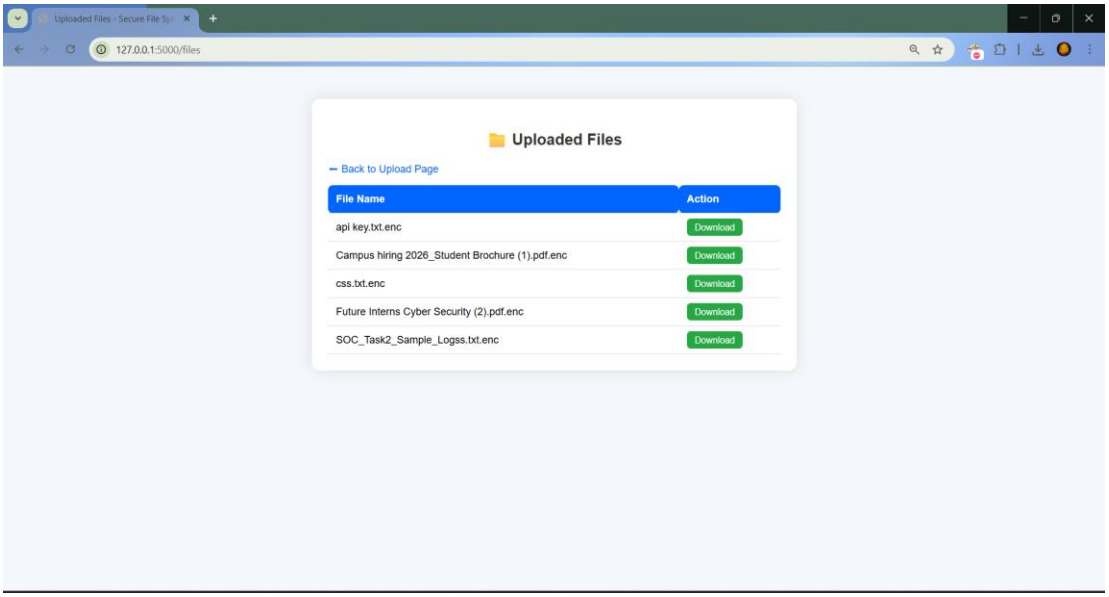
5. Browser prompts download of the decrypted file.





4.3 Files List Page

- Displays all uploaded files from **encrypted_files/** folder in a table.
- Each file has a **Download** button for secure decryption and download.
- Clean, responsive, and styled UI.



5. Encryption & Security Details

- **Algorithm:** AES-128 in EAX mode (ensures confidentiality + integrity).
- **Key Management:** AES key stored in **key.key** file (16 bytes).
- **File Handling:**
  - Encrypted files stored in `encrypted_files/`
  - Decrypted downloads stored in `downloads/`
- **Flash Messages:** Gives feedback on upload/download without exposing sensitive info.
- **Development Safety:** Flask debug mode used for testing, not for production.

6. User Interface

6.1 Upload Page

- Upload & Encrypt Section
- Download & Decrypt Section
- Link to **Files List Page**

- Flash messages displayed dynamically on same page

6.2 Files List Page

- Table view of all uploaded files
- Each row: File name + Download button
- Stylish, professional UI with hover effects

7. Testing

Test Case	Result
Upload single small file	Success, encrypted in folder, flash message displayed
Upload multiple files	Success, all encrypted
Download encrypted file	Success, decrypted in downloads folder
Download non-existent file	Flash message shown: File not found
Files List page	Shows all uploaded files with download buttons

8. Challenges & Solutions

Challenge	Solution
Flask TemplateNotFound	Created `templates/` folder with correct file names
Decrypted files saved in root folder	Added `downloads/` folder to store decrypted files separately
Upload feedback not visible	Implemented Flask **flash messages** for same-page success notifications
Ensuring encryption & decryption integrity	Used AES EAX mode for combined confidentiality & integrity

9. Key Learnings

- Flask web development fundamentals
- Implementing AES encryption/decryption using PyCryptodome
- Flash messages for dynamic UI feedback
- Secure file handling & folder organization
- Version control (GitHub) for project management

---

10. Future Improvements

- Auto-delete decrypted files after download for extra security
- File size/type restrictions to prevent malicious uploads
- User authentication to restrict file access
- Cloud integration for multi-user secure file sharing

---

11. Conclusion

The **Secure File Sharing System** successfully demonstrates secure file handling, AES encryption, and a clean web interface. It simulates real-world secure data sharing applications and can be extended further with authentication, cloud storage, and additional security features.