Secure File Transfer website

Project Title:

Secure File Sharing System Using AES Encryption

Intern Name:

Kanishka Mishra

Internship Program:

Cyber Security Internship – Future Interns

Task Number:

Task 03 – Secure File Sharing System

Tools Used:

Python Flask, PyCryptodome, HTML, CSS, JavaScript

Date:-

27th December 2025

My Github Link: -

knishka-pixel/FUTURE_CS_01: Cyber Security Internship Tasks – Future Interns

INTRODUCTION

The increasing reliance on digital communication and data storage has made secure file transfer an essential requirement for organizations and individuals. Sensitive information such as legal records, healthcare data, financial documents, and corporate files must be protected from unauthorized access.

In this project, a Secure File Sharing System was developed using Python Flask and AES encryption. The system allows users to upload files, which are encrypted before storage, and securely decrypt them during download. This project simulates real-world secure file handling processes used in industries where confidentiality and data protection are critical.

The goal of the project is to understand secure encryption implementation, backend development, cryptography basics, and secure key handling practices.

OBJECTIVE

The main objectives of this project are:

• To develop a secure web-based file upload and download platform
• To implement AES encryption for data confidentiality
• To ensure files remain protected during storage (data-at-rest security)
• To securely decrypt files during download
• To understand encryption key handling and security concepts
• To gain practical exposure to secure backend development

This project also helps in understanding how real-world secure systems are designed and implemented in cybersecurity environments.

TOOLS & TECHNOLOGIES

Programming Language:
Python

Framework:
Flask (Micro-web framework)

Encryption Library:
PyCryptodome – Advanced cryptography library supporting AES

Frontend:
HTML, CSS, JavaScript

Environment:
Windows 10 / 11

Command Line Tool:
CMD / PowerShell

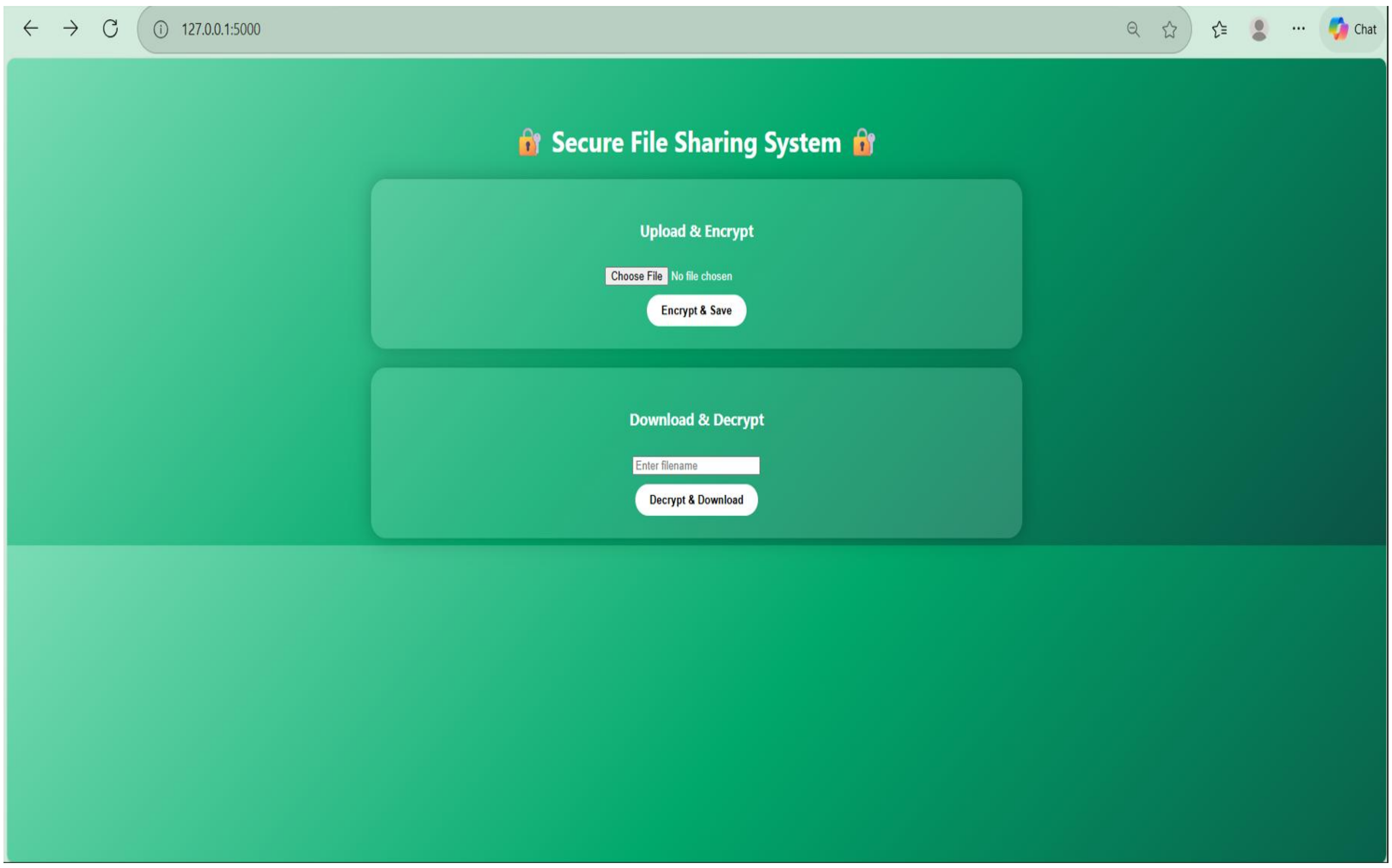
Version Control:
GitHub

SYSTEM ARCHITECTURE

The system consists of a client interface where the user uploads a file. Upon upload, the backend generates a secure encryption key and encrypts the file using the AES algorithm. The encrypted file is stored on the server. When the user requests a download, the system decrypts the file using the same key and returns the original file.

Workflow Steps:

1. User selects a file
2. File is uploaded to the Flask server
3. AES encryption is applied
4. Encrypted file is stored
5. User downloads file
6. AES decryption restores original content

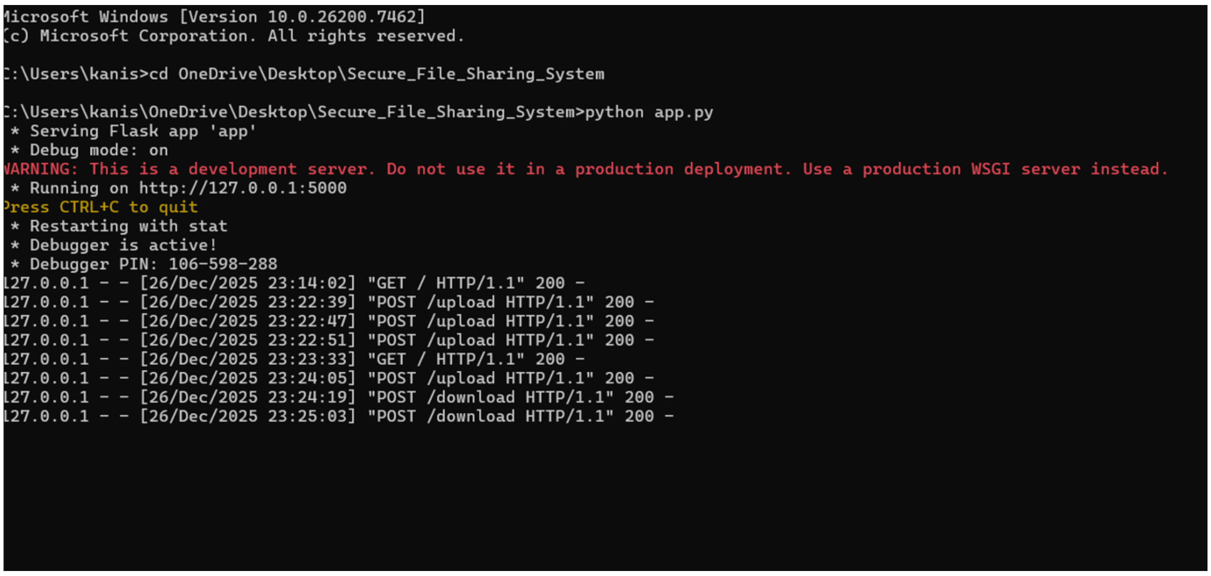

127.0.0.1:5000
Chat
🔐 Secure File Sharing System 🔐
Upload & Encrypt
Choose File No file chosen
Encrypt & Save
Download & Decrypt
Enter filename
Decrypt & Download

IMPLEMENTATION

The project was implemented by creating a Flask backend for handling file uploads and downloads. AES encryption from PyCryptodome was used to encrypt files securely. The user interface was developed using HTML and CSS to create a simple and user-friendly design.

- Encryption occurs before saving the file
- A unique key is generated
- Files are decrypted only when requested

This ensures secure handling of data at rest.



```
icrosoft Windows [Version 10.0.26200.7462]
c) Microsoft Corporation. All rights reserved.

:\Users\kanis>cd OneDrive\Desktop\Secure_File_Sharing_System

:\Users\kanis\OneDrive\Desktop\Secure_File_Sharing_System>python app.py
 * Serving Flask app 'app'
 * Debug mode: on
ARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
ress CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 106-598-288
27.0.0.1 - - [26/Dec/2025 23:14:02] "GET / HTTP/1.1" 200 -
27.0.0.1 - - [26/Dec/2025 23:22:39] "POST /upload HTTP/1.1" 200 -
27.0.0.1 - - [26/Dec/2025 23:22:47] "POST /upload HTTP/1.1" 200 -
27.0.0.1 - - [26/Dec/2025 23:22:51] "POST /upload HTTP/1.1" 200 -
27.0.0.1 - - [26/Dec/2025 23:23:33] "GET / HTTP/1.1" 200 -
27.0.0.1 - - [26/Dec/2025 23:24:05] "POST /upload HTTP/1.1" 200 -
27.0.0.1 - - [26/Dec/2025 23:24:19] "POST /download HTTP/1.1" 200 -
27.0.0.1 - - [26/Dec/2025 23:25:03] "POST /download HTTP/1.1" 200 -
```

TESTING & RESULTS

The system was tested by uploading multiple files including:

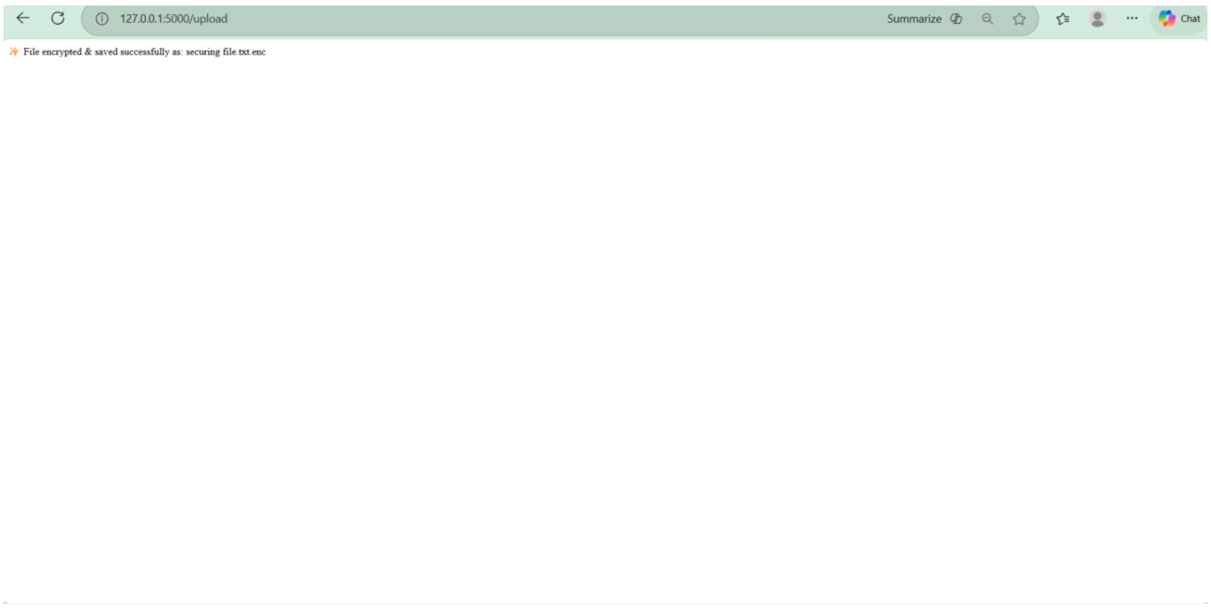
• Text files

• Images

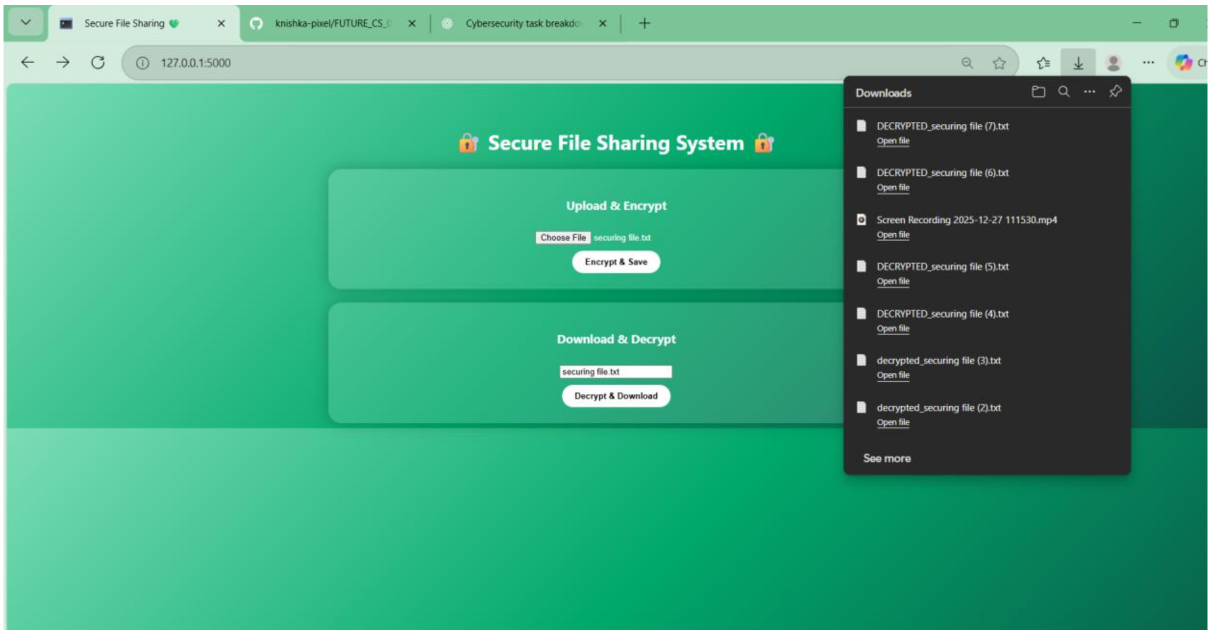
• Documents

Each uploaded file was successfully encrypted and stored securely. During download, the decryption process restored the original file content accurately.

This verified that the encryption and decryption processes worked correctly without data loss.



```
✨ File encrypted & saved successfully as: securing file.txt.enc
```


Secure File Sharing
knishka-pixel/FUTURE_CS_
Cybersecurity task breakdo
127.0.0.1:5000
Secure File Sharing System
Upload & Encrypt
Choose File
securing file.txt
Encrypt & Save
Download & Decrypt
securing file.txt
Decrypt & Download
Downloads
DECRYPTED_securing file (7).txt
Open file
DECRYPTED_securing file (6).txt
Open file
Screen Recording 2025-12-27 111530.mp4
Open file
DECRYPTED_securing file (5).txt
Open file
DECRYPTED_securing file (4).txt
Open file
decrypted_securing file (3).txt
Open file
decrypted_securing file (2).txt
Open file
See more

SECURITY CONSIDERATIONS

To improve security in real-world deployment, the following measures are recommended:

• Secure key storage solutions such as environment variables or vaults
• HTTPS encryption during transfer
• Authentication & user access control
• Logging & monitoring
• Secure server deployment
• Preventing unauthorized file uploads
• Limiting allowed file types

This ensures strong data protection.

CONCLUSION

This project successfully demonstrated how AES encryption can be implemented to build a secure file-sharing system. The system ensures that sensitive files remain confidential during storage and retrieval. Through this project, I gained hands-on experience in:

- Flask backend development
- Web security concepts
- AES encryption
- Secure file handling
- Practical cybersecurity implementation

This project closely simulates real-world secure systems used in modern organizations.