SECURE FILE SHARING SYSTEM

CYBERSECURITY INTERNSHIP TASK-03

FUTURE INTERNS

Title: Secure File Sharing System

Intern Name: Akansha Bhagat

Date: 18/06/2025

INTRODUCTION

In today's digital landscape, the secure transfer of data has become a critical concern, especially in sectors like healthcare, law, and finance. As part of my cybersecurity internship under **Future Interns**, I undertook the task of designing and implementing a **Secure File Sharing System.** This system enables users to upload and download files securely while ensuring the confidentiality of data using encryption.

OBJECTIVE OF THE TASK

The main goals of the task were:

- To develop a basic but secure web portal for file sharing.
- To implement encryption logic for uploaded files using AES.
- To ensure files are safely decrypted only when downloaded.
- To securely manage and handle encryption keys using environment variables.
- To simulate real-world secure data sharing for internship deliverables.

TOOLS AND TECHNOLOGIES USED

Tool/Technology	**Purpose**
Python 3.12	Programming language for backend logic
Flask	Lightweight Python web framework
PyCryptodome	Library for AES encryption
Visual Studio Code	Code editor
.env File	For secure storage of encryption keys
Edge Browser	Used for interacting with the local app
Localhost (127.0.0.1:5000)	Hosting environment for testing the app

METHODOLOGY

The Secure File Sharing System was implemented through the following steps:

Step 1: Setting Up the Environment

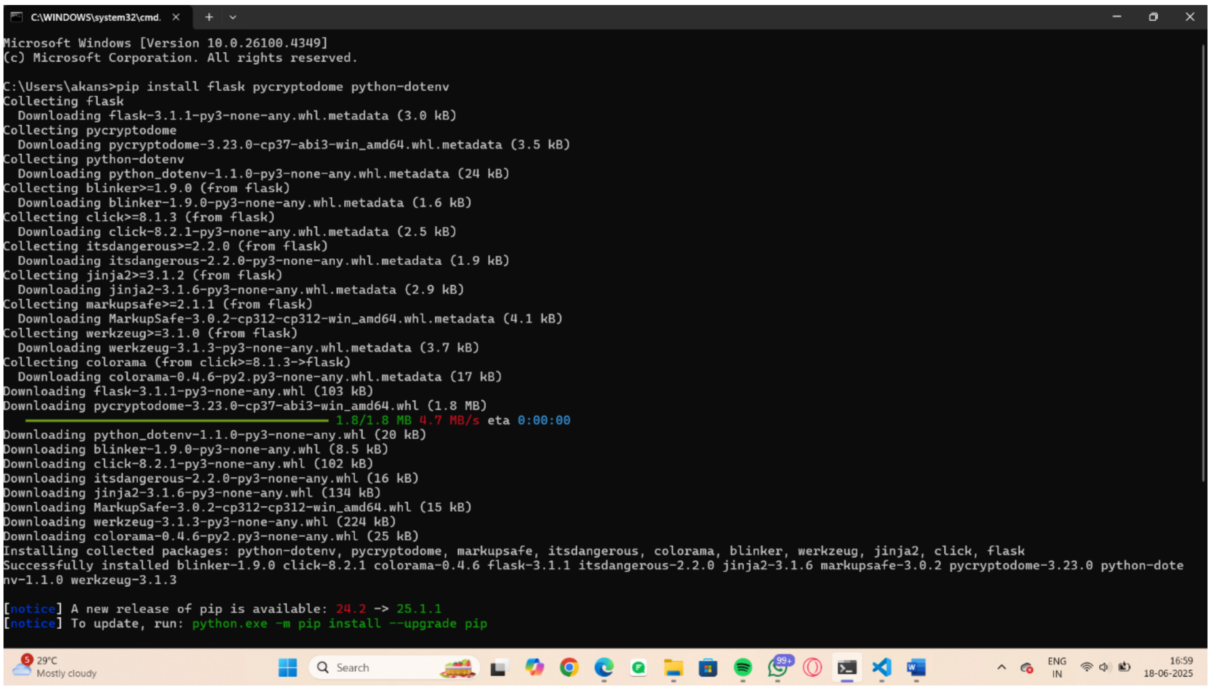
Python 3.12 and necessary libraries were installed.

These include:

- flask
- pycryptodome
- python-dotenv

The command used was:

```
pip install flask pycryptodome python-dotenv
```



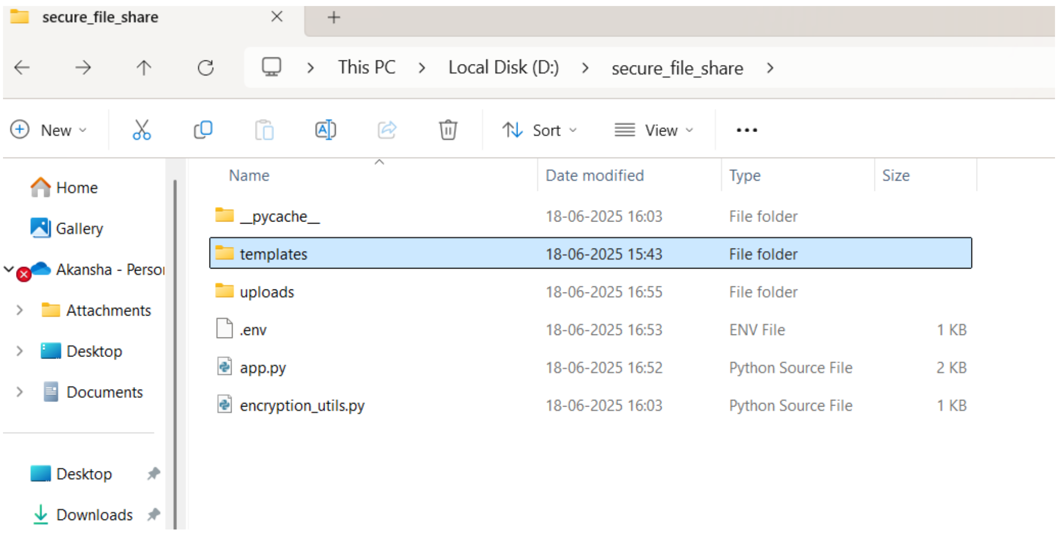
```
Microsoft Windows [Version 10.0.26100.4349]
(c) Microsoft Corporation. All rights reserved.

C:\Users\akans>pip install flask pycryptodome python-dotenv
Collecting flask
  Downloading flask-3.1.1-py3-none-any.whl.metadata (3.0 kB)
Collecting pycryptodome
  Downloading pycryptodome-3.23.0-cp37-abi3-win_amd64.whl.metadata (3.5 kB)
Collecting python-dotenv
  Downloading python_dotenv-1.1.0-py3-none-any.whl.metadata (24 kB)
Collecting blinker>=1.9.0 (from flask)
  Downloading blinker-1.9.0-py3-none-any.whl.metadata (1.6 kB)
Collecting click>=8.1.3 (from flask)
  Downloading click-8.2.1-py3-none-any.whl.metadata (2.5 kB)
Collecting itsdangerous>=2.2.0 (from flask)
  Downloading itsdangerous-2.2.0-py3-none-any.whl.metadata (1.9 kB)
Collecting jinja2>=3.1.2 (from flask)
  Downloading jinja2-3.1.6-py3-none-any.whl.metadata (2.9 kB)
Collecting markupsafe>=2.1.1 (from flask)
  Downloading MarkupSafe-3.0.2-cp312-cp312-win_amd64.whl.metadata (4.1 kB)
Collecting werkzeug>=3.1.0 (from flask)
  Downloading werkzeug-3.1.3-py3-none-any.whl.metadata (3.7 kB)
Collecting colorama (from click>=8.1.3->flask)
  Downloading colorama-0.4.6-py2.py3-none-any.whl.metadata (17 kB)
Downloading flask-3.1.1-py3-none-any.whl (103 kB)
Downloading pycryptodome-3.23.0-cp37-abi3-win_amd64.whl (1.8 MB)
   ---------------------------------------- 1.8/1.8 MB 4.7 MB/s eta 0:00:00
Downloading python_dotenv-1.1.0-py3-none-any.whl (20 kB)
Downloading blinker-1.9.0-py3-none-any.whl (8.5 kB)
Downloading click-8.2.1-py3-none-any.whl (102 kB)
Downloading itsdangerous-2.2.0-py3-none-any.whl (16 kB)
Downloading jinja2-3.1.6-py3-none-any.whl (134 kB)
Downloading MarkupSafe-3.0.2-cp312-cp312-win_amd64.whl (15 kB)
Downloading werkzeug-3.1.3-py3-none-any.whl (224 kB)
Downloading colorama-0.4.6-py2.py3-none-any.whl (25 kB)
Installing collected packages: python-dotenv, pycryptodome, markupsafe, itsdangerous, colorama, blinker, werkzeug, jinja2, click, flask
Successfully installed blinker-1.9.0 click-8.2.1 colorama-0.4.6 flask-3.1.1 itsdangerous-2.2.0 jinja2-3.1.6 markupsafe-3.0.2 pycryptodome-3.23.0 python-dote
nv-1.1.0 werkzeug-3.1.3

[notice] A new release of pip is available: 24.2 -> 25.1.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

Figure: Libraries Installed Successfully

Step 2: Creating the Project Structure

A dedicated project folder was created named **secure_file_share.** Inside this folder, the following components were added:

- **app.py** – The main Flask backend handling routes and requests.
- **encryption_utlis.py** – Handles AES encryption and decryption logic.
- **.env** – Stores the 16-byte secret key securely.
- **Templates/index.html** – User interface for uploading and downloading files.



Name	Date modified	Type	Size
__pycache__	18-06-2025 16:03	File folder	
templates	18-06-2025 15:43	File folder	
uploads	18-06-2025 16:55	File folder	
.env	18-06-2025 16:53	ENV File	1 KB
app.py	18-06-2025 16:52	Python Source File	2 KB
encryption_utils.py	18-06-2025 16:03	Python Source File	1 KB

Figure: Project Structure for Secure File Sharing

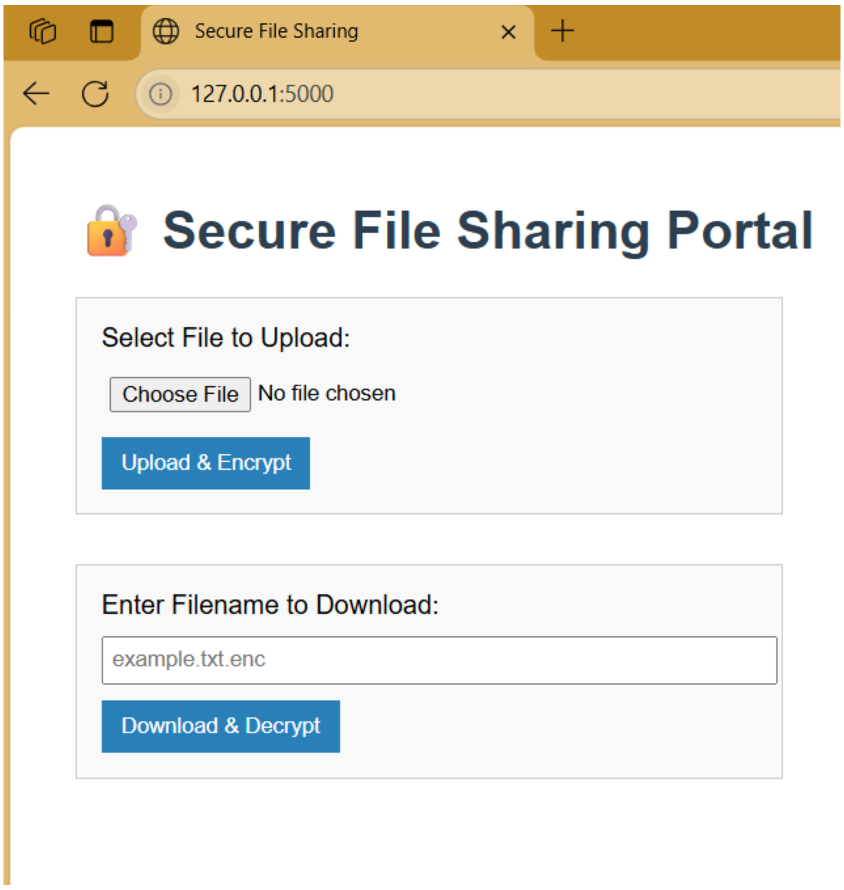
Step 3: Developing AES Encryption Logic

Using the PyCryptodome library, AES encryption was implemented in ECB mode. Files uploaded via the web interface were encrypted before saving using a secure key. During download, the encrypted files were decrypted in real-time using the same key.

Step 4: Building the Web Interface

A simple, user-friendly HTML form was created that allows:

- File selection and upload
- Encryption of files before saving
- File selection and download (with decryption)



Secure File Sharing Portal

Select File to Upload:

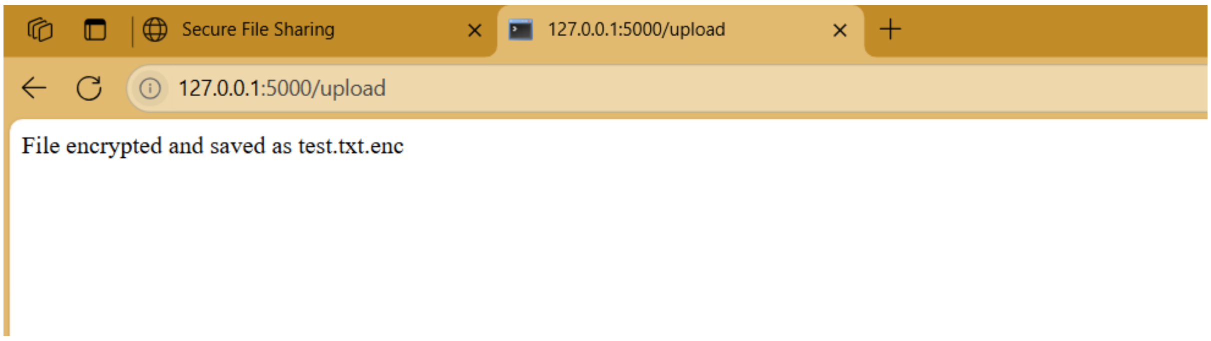
Enter Filename to Download:

Figure: Upload Form in the Web Interface

Step 5: Testing the Application

The system was tested by uploading and downloading files. The final decrypted file was validated to ensure the integrity of content. Errors were handled gracefully, and key mismatch issues were resolved using .env secrets.

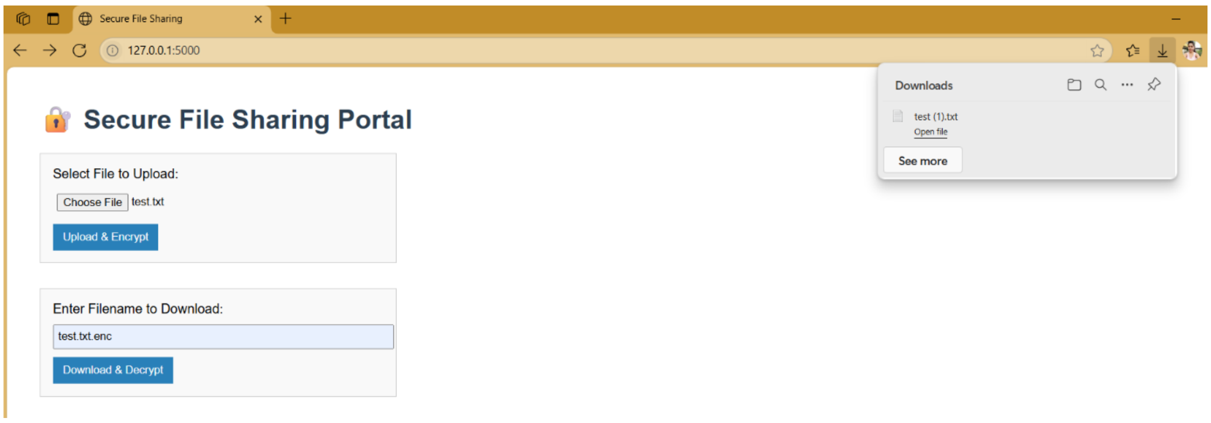
Encryption Message:



File encrypted and saved as test.txt.enc

Figure: Console Message Showing Successful Encryption

Download and Decryption Confirmation:



Secure File Sharing Portal

Select File to Upload:

test.txt

Enter Filename to Download:

test.txt.enc

Figure: Downloaded Decrypted File with Original Content

SECURITY MEASURES IMPLEMENTED

The following security features were incorporated in the system:

- **AES Encryption (128-bit):** All files are encrypted using a secure key.
- **Key Management via. env**: sensitive keys were stored separately from the source code.
- **Encrypted Storage**: Files are never stored in plain format on the server.
- **Download-side Decryption**: Decryption only happens when the user downloads the file.
- **Separation of Concerns**: Encryption logic and routing were handled in separate files.

LEARNINGS AND SKILLS GAINED

This project helped me to gain practical knowledge in the following areas:

- Implementing real-world encryption using **AES**
- Securely handling sensitive information like encryption keys
- Developing a full-stack **Flask application**
- Understanding **file stream encryption/decryption**
- **Debugging errors** related to key mismatches and Flask routes
- Managing environment configuration with .env securely

CONCLUSION

The Secure File Sharing System successfully meets its goal of providing encrypted upload and download functionalities in a simple web interface. It adheres to secure development practices and demonstrates how critical concepts like encryption, key management, and secure architecture can be implemented using beginner-friendly tools.

This task has enhanced my understanding of how web security and cryptography work together in real-world scenarios and has improved my confidence in developing secure applications.