

SECURE FILE SHARING SYSTEM

CYBERSECURITY INTERNSHIP TASK-03

FUTURE INTERNS

Title: Secure File Sharing System

Intern Name: Anush N

Date: 15/07/2025

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

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Anush> pip install flask
Requirement already satisfied: flask in c:\users\anush\appdata\local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (3.0.3)
Requirement already satisfied: Werkzeug>=3.0.0 in c:\users\anush\appdata\local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (from flask) (3.0.3)
Requirement already satisfied: Jinja2>=3.1.2 in c:\users\anush\appdata\local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (from flask) (3.1.4)
Requirement already satisfied: itsdangerous>=2.1.2 in c:\users\anush\appdata\local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (from flask) (2.2.0)
Requirement already satisfied: click>=8.1.3 in c:\users\anush\appdata\local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (from flask) (8.1.7)
Requirement already satisfied: blinker>=1.6.2 in c:\users\anush\appdata\local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (from flask) (1.8.2)
Requirement already satisfied: colorama in c:\users\anush\appdata\local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (from click>=8.1.3->flask) (0.4.6)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\anush\appdata\local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (from Jinja2>=3.1.2->flask) (2.1.5)

[notice] A new release of pip is available: 24.1.1 -> 25.1.1
[notice] To update, run: C:\Users\Anush\AppData\Local\Microsoft\WindowsApps\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\python.exe -m pip install --upgrade pip
PS C:\Users\Anush> |
```

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_utils.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.

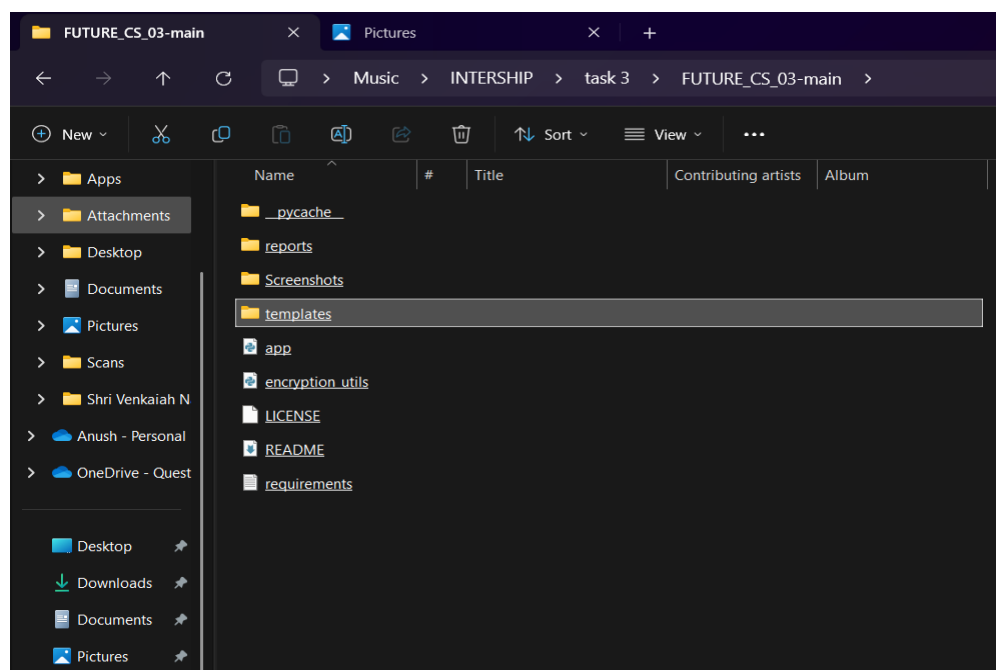


Figure: Project Structure for Secure File Sharing

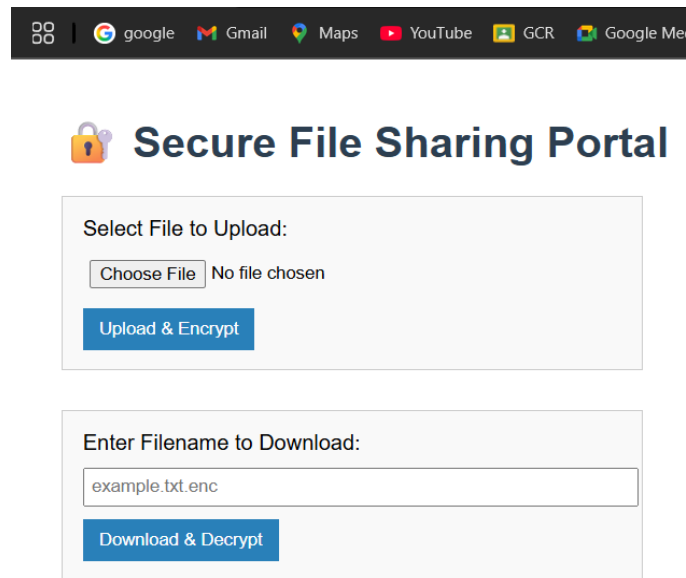
Step 3: Developing AES Encryption Logic

Using the PyCryptodome library, AES encryption was implemented in ECB mode. File 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)



The screenshot shows a web browser window with a dark header bar containing navigation links: google, Gmail, Maps, YouTube, GCR, and Google Me. Below the header is the title "Secure File Sharing Portal" with a padlock icon. The main content area contains two forms. The first form, titled "Select File to Upload:", has a "Choose File" button and a "No file chosen" status. Below it is a blue "Upload & Encrypt" button. The second form, titled "Enter Filename to Download:", has a text input field containing "example.txt.enc" and a blue "Download & Decrypt" button.

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.

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.