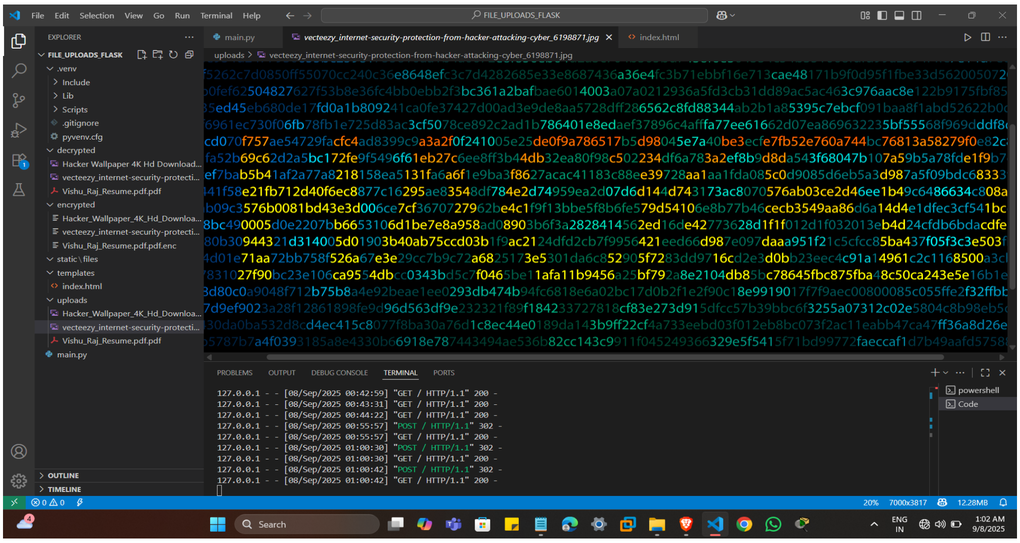
Cybersecurity Internship Project Report

Project Title: Secure File Sharing System

This report documents my work during my cybersecurity internship with Future Intern. The project undertaken was titled **Secure File Sharing System**. The objective of this project was to design and implement a secure file sharing mechanism using Flask (Python) and AES encryption to ensure confidentiality and integrity of files.

Project Overview:
The Secure File Sharing System allows users to upload files through a web interface. Once uploaded, the files are automatically encrypted using the AES encryption algorithm, stored securely in an encrypted folder, and can later be decrypted for safe retrieval. This system demonstrates how cryptographic methods can be integrated into web applications to protect sensitive data from unauthorized access.

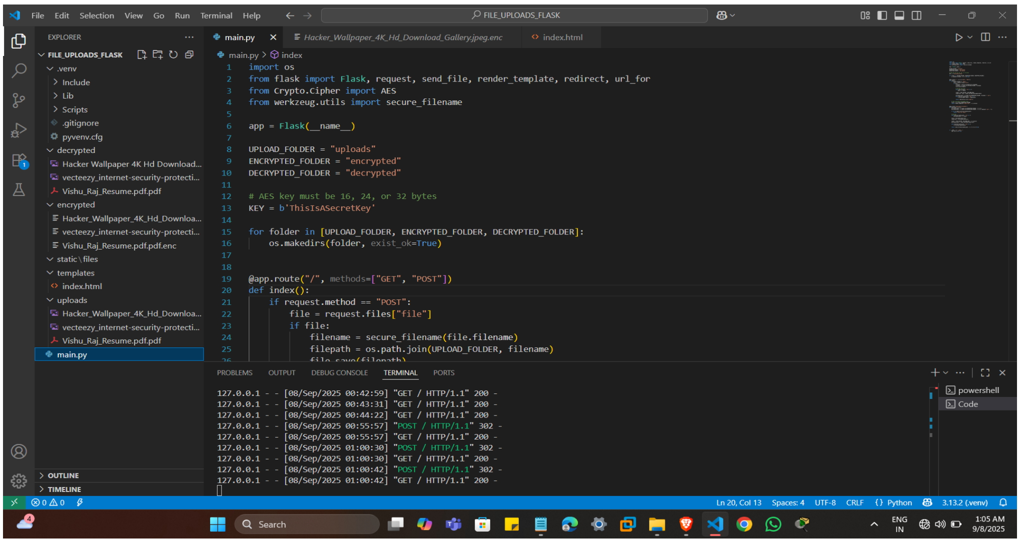
Technology Stack: - Python (Flask Framework) - AES Encryption (Crypto.Cipher library) - HTML (Frontend Interface) - VS Code for Development - Localhost Testing via Flask Server


FILE_UPLOADS_FLASK
main.py
vecteezy_internet-security-protection-from-hacker-attacking-cyber_6198871.jpg
index.html
uploads > vecteezy_internet-security-protection-from-hacker-attacking-cyber_6198871.jpg
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
powershell
Code

```
f5262c7d0850ff55070cc240c36e8648efc3c7d4282685e33e8687436a36e4fc3b71ebbf16e713cae48171b9f0d95f1fbe33d562005072
b0fef62504827627f53b8e36fc4bb0ebb2f3bc361a2bafbae6014003a07a0212936a5fd3cb31dd89ac5ac463c976aac8e122b9175fbf85
35ed45eb680de17fd0a1b809241ca0fe37427d00ad3e9de8aa5728dff286562c8fd88344ab2b1a85395c7ebcf091baa8f1abd52622b0c
6961ec730f06fb78fb1e725d83ac3cf5078ce892c2ad1b786401e8edaef37896c4afffa77ee61662d07ea869632235bf55568f969dddf8
cd070f757ae54729facfc4ad8399c9a3a2f0f241005e25de0f9a786517b5d98045e7a40be3ecfe7fb52e760a744bc76813a58279f0e82c
fa52b69c62d2a5bc172fe9f5496f61eb27c6ee8ff3b44db32ea80f98c502234df6a783a2ef8b9d8da543f68047b107a59b5a78fde1f9b7
ef7bab5b41af2a77a8218158ea5131fa6a6f1e9ba3f8627acac41183c88ee39728aa1aa1fda085c0d9085d6eb5a3d987a5f09bdc6833
41f58e21fb712d40f6ec8877c16295ae83548df784e2d74959ea2d07d6d144d743173ac8070576ab03ce2d46ee1b49c6486634c808a
b09c3576b0081bd43e3d006ce7cf3670727962be4c1f9f13bbe5f8b6fe579d54106e8b77b46cecb3549aa86d6a14d4e1dfec3cf541bc
8bc490005d0e2207bb6653106d1be7e8a958ad08903b6f3a2828414562ed16de42773628d1f1f012d1f032013eb4d24cfdb6bdacdfe
80b30944321d314005d01903b40ab75ccd03b1f9ac2124dfd2cb7f9956421eed66d987e097daaa951f21c5cfcc85ba437f05f3c3e503f
4d01e71aa72bb758f526a67e3e29cc7b9c72a6825173e5301da6c852905f7283dd9716cd2e3d0bb23eec4c91a14961c2c1168500a3c
7831027f90bc23e106ca9554dbcc0343bd5c7f0465be11afa11b9456a25bf792a8e2104db85bc78645fbc875fba48c50ca243e5e16b1e
3d80c0a9048f712b75b8a4e92beae1ee0293db474b94fc6818e6a02bc17d0b2f1e2f90c18e9919017f7f9aec00800085c055ffe2f32ffbb
7d9ef9023a28f12861898fe9d96d563df9e232321f89f184233727818cf83e273d915dfcc57b39bbc6f3255a07312c02e5804c8b98eb5
30da0ba532d8cd4ec415c8077f8ba30a76d1c8ec44e0189da143b9ff22cf4a733eebd03f012eb8bc073f2ac11eabb47ca47ff36a8d26e
5787b7a4f0393185a8e4330b66918e787443494ae536b82cc143c9911f045249366329e5f5415f71bd99772faeccaf1d7b49aafd5758
```

```
127.0.0.1 - - [08/Sep/2025 00:42:59] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [08/Sep/2025 00:43:31] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [08/Sep/2025 00:44:22] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [08/Sep/2025 00:55:57] "POST / HTTP/1.1" 302 -
127.0.0.1 - - [08/Sep/2025 00:55:57] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [08/Sep/2025 01:00:30] "POST / HTTP/1.1" 302 -
127.0.0.1 - - [08/Sep/2025 01:00:30] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [08/Sep/2025 01:00:42] "POST / HTTP/1.1" 302 -
127.0.0.1 - - [08/Sep/2025 01:00:42] "GET / HTTP/1.1" 200 -
```

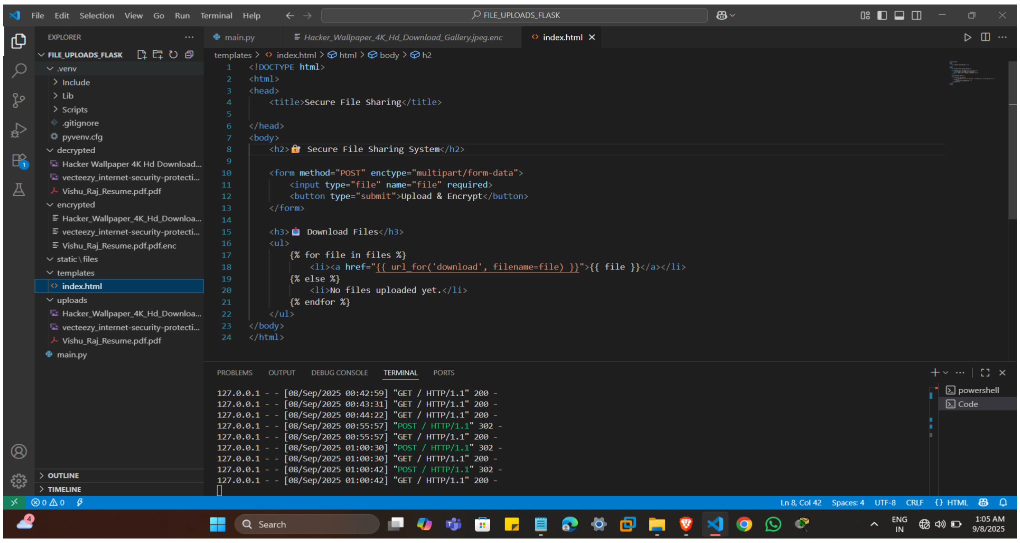
This image shows the encrypted hexadecimal representation of the uploaded files after AES encryption. The encryption ensures data confidentiality.



```python
import os
from flask import Flask, request, send_file, render_template, redirect, url_for
from Crypto.Cipher import AES
from werkzeug.utils import secure_filename

app = Flask(__name__)

UPLOAD_FOLDER = "uploads"
ENCRYPTED_FOLDER = "encrypted"
DECRYPTED_FOLDER = "decrypted"

# AES key must be 16, 24, or 32 bytes
KEY = b'ThisIsASecretKey'

for folder in [UPLOAD_FOLDER, ENCRYPTED_FOLDER, DECRYPTED_FOLDER]:
    os.makedirs(folder, exist_ok=True)


@app.route("/", methods=["GET", "POST"])
def index():
    if request.method == "POST":
        file = request.files["file"]
        if file:
            filename = secure_filename(file.filename)
            filepath = os.path.join(UPLOAD_FOLDER, filename)
```

```
127.0.0.1 - - [08/Sep/2025 00:42:59] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [08/Sep/2025 00:43:31] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [08/Sep/2025 00:44:22] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [08/Sep/2025 00:55:57] "POST / HTTP/1.1" 302 -
127.0.0.1 - - [08/Sep/2025 00:55:57] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [08/Sep/2025 01:00:30] "POST / HTTP/1.1" 302 -
127.0.0.1 - - [08/Sep/2025 01:00:30] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [08/Sep/2025 01:00:42] "POST / HTTP/1.1" 302 -
127.0.0.1 - - [08/Sep/2025 01:00:42] "GET / HTTP/1.1" 200 -
```

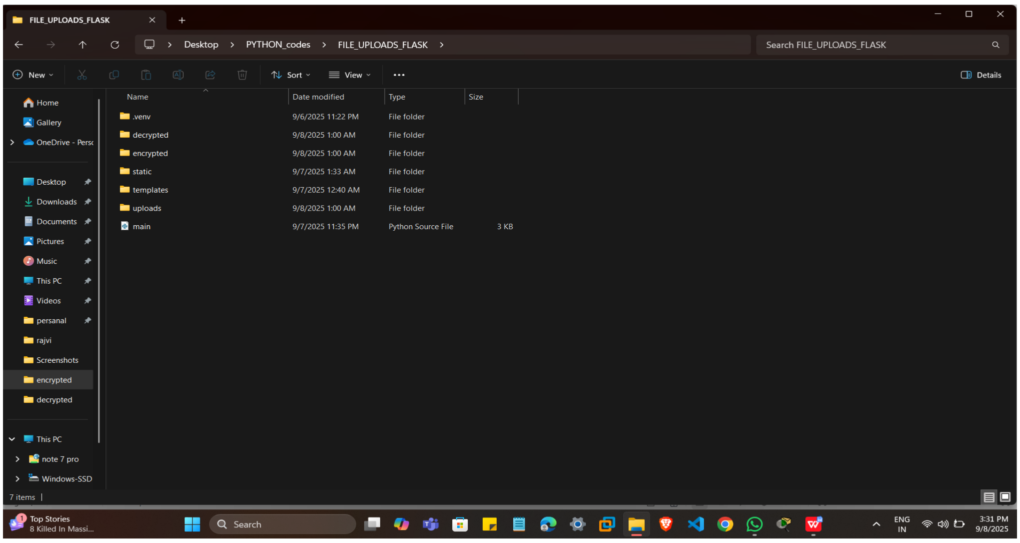
This screenshot displays the main Flask backend (main.py) where the AES encryption, file upload, and storage functionalities are implemented.



```html
<!DOCTYPE html>
<html>
<head>
    <title>Secure File Sharing</title>

</head>
<body>
    <h2>🔐 Secure File Sharing System</h2>

    <form method="POST" enctype="multipart/form-data">
        <input type="file" name="file" required>
        <button type="submit">Upload & Encrypt</button>
    </form>

    <h3>📥 Download Files</h3>
    <ul>
        {% for file in files %}
            <li><a href="{{ url_for('download', filename=file) }}">{{ file }}</a></li>
        {% else %}
            <li>No files uploaded yet.</li>
        {% endfor %}
    </ul>
</body>
</html>
```

```
127.0.0.1 - - [08/Sep/2025 00:42:59] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [08/Sep/2025 00:43:31] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [08/Sep/2025 00:44:22] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [08/Sep/2025 00:55:57] "POST / HTTP/1.1" 302 -
127.0.0.1 - - [08/Sep/2025 00:55:57] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [08/Sep/2025 01:00:30] "POST / HTTP/1.1" 302 -
127.0.0.1 - - [08/Sep/2025 01:00:30] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [08/Sep/2025 01:00:42] "POST / HTTP/1.1" 302 -
127.0.0.1 - - [08/Sep/2025 01:00:42] "GET / HTTP/1.1" 200 -
```

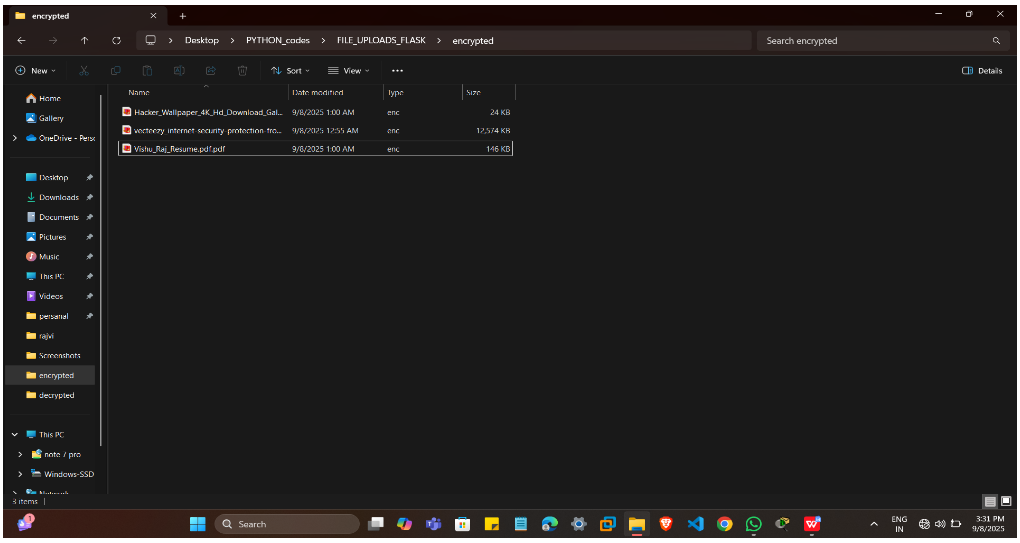
This screenshot shows the HTML template (index.html) that provides the frontend interface for users to upload files securely.



Desktop > PYTHON_codes > FILE_UPLOADS_FLASK

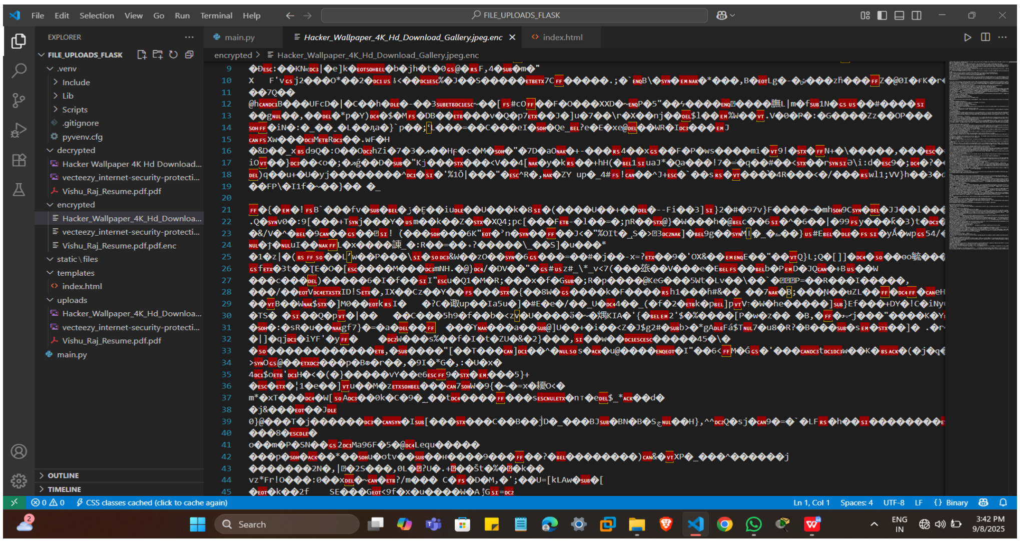
Name	Date modified	Type	Size
.venv	9/6/2025 11:22 PM	File folder	
decrypted	9/8/2025 1:00 AM	File folder	
encrypted	9/8/2025 1:00 AM	File folder	
static	9/7/2025 1:33 AM	File folder	
templates	9/7/2025 12:40 AM	File folder	
uploads	9/8/2025 1:00 AM	File folder	
main	9/7/2025 11:35 PM	Python Source File	3 KB

This image displays the folder structure of the project including uploads, encrypted, and decrypted directories for file management.



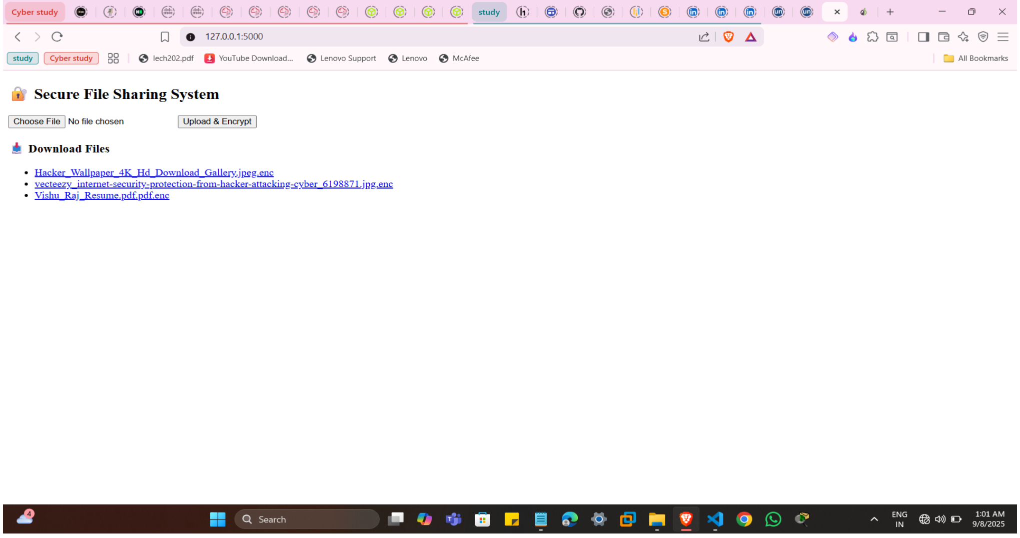
Name	Date modified	Type	Size
Hacker_Wallpaper_4K_Hd_Download_Gal...	9/8/2025 1:00 AM	enc	24 KB
vecteezy_internet-security-protection-fro...	9/8/2025 12:55 AM	enc	12,574 KB
Vishu_Raj_Resume.pdf.pdf	9/8/2025 1:00 AM	enc	146 KB

This screenshot shows the encrypted files stored with '.enc' extension, ensuring they cannot be read without proper decryption.



File Edit Selection View Go Run Terminal Help
FILE_UPLOADS_FLASK
EXPLORER
main.py
Hacker_Wallpaper_4K_Hd_Download_Gallery.jpeg.enc
index.html
encrypted > Hacker_Wallpaper_4K_Hd_Download_Gallery.jpeg.enc
FILE_UPLOADS_FLASK
.venv
Include
Lib
Scripts
.gitignore
pyvenv.cfg
decrypted
Hacker Wallpaper 4K Hd Download...
vecteezy_internet-security-protecti...
Vishu_Raj_Resume.pdf.pdf
encrypted
Hacker_Wallpaper_4K_Hd_Downloa...
vecteezy_internet-security-protecti...
Vishu_Raj_Resume.pdf.pdf.enc
static\files
templates
index.html
uploads
Hacker_Wallpaper_4K_Hd_Downloa...
vecteezy_internet-security-protecti...
Vishu_Raj_Resume.pdf.pdf
main.py
OUTLINE
TIMELINE
CSS classes cached (click to cache again)
Ln 1, Col 1 Spaces: 4 UTF-8 LF Binary

This image displays the binary content of an encrypted file, which appears as unreadable characters ensuring security.



🔐 Secure File Sharing System

Choose File No file chosen

Upload & Encrypt

📥 Download Files

- Hacker_Wallpaper_4K_Hd_Download_Gallery.jpeg.enc
- vecteezy_internet-security-protection-from-hacker-attacking-cyber_6198871.jpg.enc
- Vishu_Raj_Resume.pdf.pdf.enc

This screenshot shows the web interface running on localhost where users can upload files and access encrypted files for download.

Conclusion:

The Secure File Sharing System project successfully demonstrated the integration of cryptographic techniques into web applications. This internship provided valuable hands-on experience in applying cybersecurity principles such as encryption, data confidentiality, and secure file transfer. The system can be further extended by adding user authentication, role-based access control, and deployment on a cloud platform for real-world use cases.