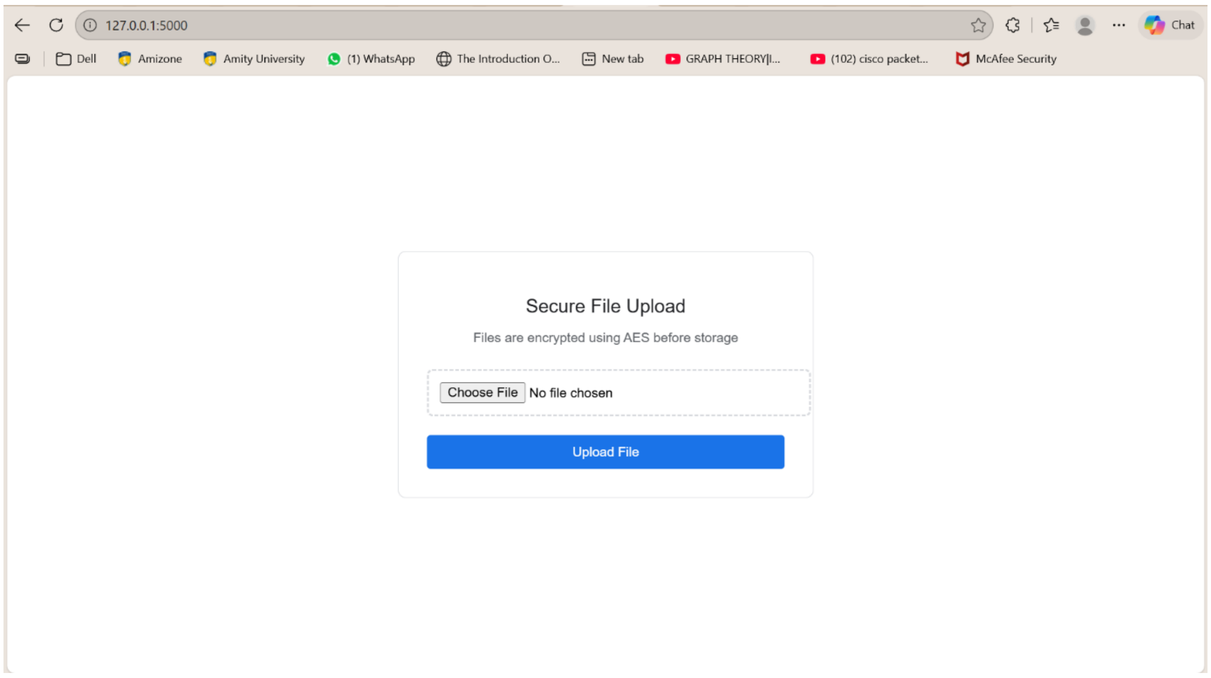
# Secure File Sharing System

- Security Overview

## 1. Introduction

This project implements a secure file sharing system using Python Flask with a focus on protecting sensitive files during storage and transfer. The primary security objective is to ensure that uploaded files are not stored in plaintext and are protected against unauthorized access using encryption techniques.

- **Screenshots of the GUI:-**



Secure File Upload

Files are encrypted using AES before storage

## 2. Encryption Method Used

The system uses Advanced Encryption Standard (AES), which is a widely accepted symmetric encryption algorithm.

- AES Key Size: 256-bit
- Encryption Mode: CBC (Cipher Block Chaining)
- Initialization Vector (IV): Randomly generated for each file

AES was chosen because it is fast, secure, and commonly used in real-world applications to protect sensitive data**.**

## 3. File Encryption Process:

When a user uploads a file:

1. The file content is read by the Flask application.
2. The data is encrypted using AES-256 before being saved.

3. A random IV is generated to ensure that identical files produce different encrypted outputs.
4. Only the encrypted version of the file (.enc) is stored on disk.

At no point is the plaintext file stored permanently on the server.

### 4. File Decryption Process

When a user downloads a file:

1. The encrypted file is read from storage.
2. The file is decrypted using the same AES secret key.
3. The original file is returned to the user in its correct format.

This ensures secure access while maintaining data confidentiality.

### 5. Key Management

A single AES secret key is generated and stored in a local file (secret.key).

- The key is generated once using a secure random generator.
- The key file is not exposed to users.

## 6. Security Benefits

- Files are encrypted at rest, preventing unauthorized access to stored data.
- Even if the server storage is compromised, encrypted files remain unreadable.
- Random IV usage prevents pattern-based attacks.
- Plaintext files are never stored permanently on disk.

## 7. Limitations

- The system does not include user authentication or access control.
- The encryption key is stored locally rather than in a dedicated key management service.
- Decrypted files are temporarily written to disk during download.

## 8. Conclusion

This secure file sharing system demonstrates the practical use of AES encryption to protect files during storage and retrieval. By encrypting files before saving and decrypting them only when required, the system ensures confidentiality and integrity of user data. The project provides a strong foundation for understanding secure file handling and basic cryptographic practices.