

Secure File Storage using Google Drive API

Abstract

This project demonstrates the implementation of a secure file storage system utilizing the Google Drive API. The goal of this system is to provide users with a reliable and secure way to store their files in the cloud while ensuring data privacy and access control. The system allows users to upload, retrieve, and manage files on Google Drive, with encryption and authentication mechanisms to safeguard the data. Google Drive's API is leveraged to interact with the cloud storage, and OAuth 2.0 is used for secure authentication. This report details the architecture, implementation, and security considerations for the system.

Author

Hyder Presswala

Roll No: **16010122151**

1. Introduction

- **Overview:** In today's digital age, data security is one of the most critical concerns for individuals and organizations. Cloud storage services, such as Google Drive, offer a convenient way to store and access files, but they must ensure robust security measures to protect sensitive data. This project implements a secure file storage system using the Google Drive API, focusing on secure file upload, retrieval, and encryption.
- **Objective:** The primary goal is to develop a secure file storage solution that utilizes Google Drive as a cloud platform and implements encryption for data security. Additionally, OAuth 2.0 authentication ensures secure access control for users.

2. Google Drive API and OAuth 2.0

- **Google Drive API:** Google Drive API provides a programmatic interface to interact with Google Drive, enabling users to upload, download, and manage files stored in their Google Drive account. It supports a wide range of file types and integrates well with other Google services.
- **OAuth 2.0 Authentication:** OAuth 2.0 is employed to authenticate users securely without requiring them to share their Google account credentials. It provides a token-based authentication mechanism that ensures that only authorized users can access their files.

3. System Architecture

- **Overview:** The system architecture consists of three major components:
 1. **User Interface:** A web-based or desktop interface where users can upload, download, and manage their files.
 2. **Google Drive Integration:** The backend of the system interacts with Google Drive via the Google Drive API, handling file uploads, downloads, and management tasks.

3. **Security Layer:** Encryption techniques (e.g., AES encryption) are applied to the files before upload to ensure their security. Decryption is performed when files are retrieved.
- **Flow of Data:**
 - User authenticates using OAuth 2.0.
 - User uploads a file, which is encrypted before being stored on Google Drive.
 - The file can be retrieved by the user, decrypted on the backend, and delivered securely.

4. File Encryption

- **Encryption Mechanism:** To ensure data privacy, files are encrypted using AES (Advanced Encryption Standard) before they are uploaded to Google Drive. This means that even if someone gains unauthorized access to the cloud storage, they will not be able to view the contents of the files without the decryption key.
- **Decryption Process:** When the user requests to download a file, it is decrypted before being sent to the user. This ensures that the data remains secure during storage and transfer.

5. Security Features

- **Authentication:** The system uses OAuth 2.0 for secure user authentication, ensuring that only authorized users can interact with their files.
- **Data Encryption:** Files are encrypted before being uploaded to Google Drive and decrypted when retrieved, ensuring that sensitive data is protected at all times.
- **Access Control:** The system ensures that users can only access their own files, with appropriate access restrictions in place.

6. Implementation

- **Frontend:** The frontend can be built using any web framework (such as Flask, React, or others) to allow users to interact with the system.
- **Backend:** The backend is responsible for handling file uploads and downloads, encryption and decryption, and interaction with the Google Drive API.
- **Google Drive API Integration:** The backend uses the Google Drive API to upload files to the user's Google Drive account and manage file metadata.
- **Code Structure:** The code is divided into several modules:
 - **Authentication:** Handles OAuth 2.0 authentication and token management.
 - **Encryption:** Manages encryption and decryption of files.
 - **File Management:** Manages file upload, download, and metadata handling using the Google Drive API.

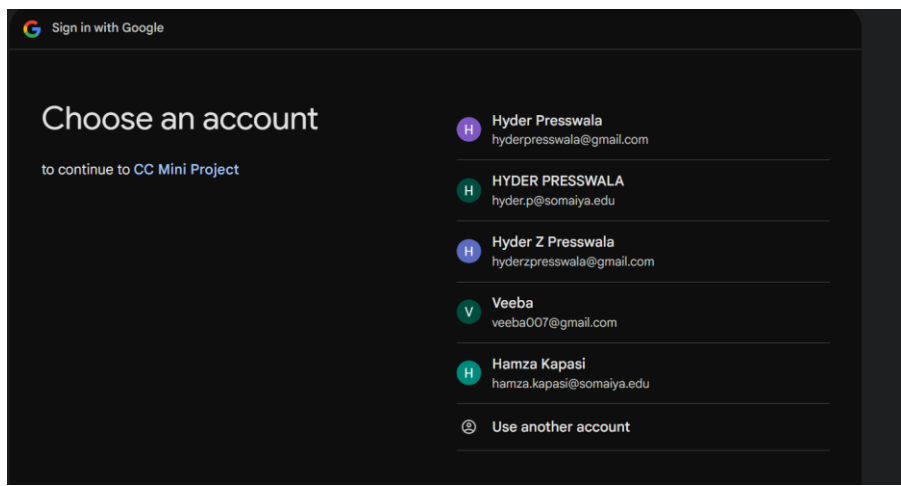
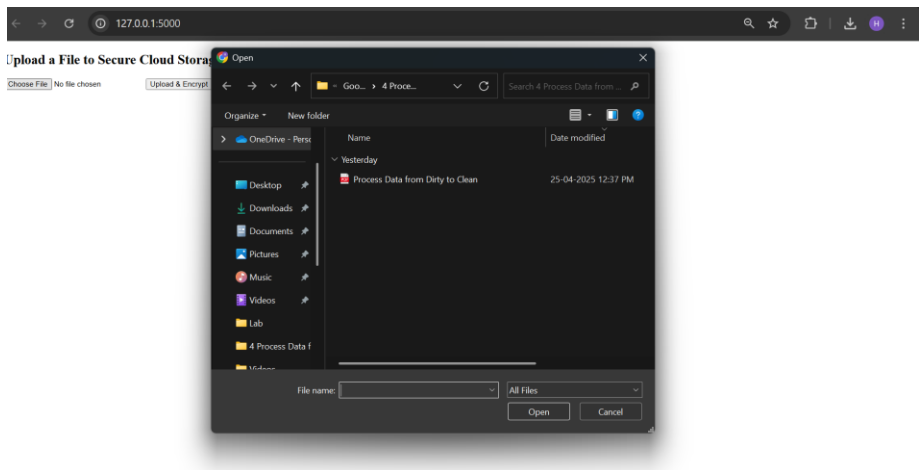
```
hyder@HyderPresswala MINGW64 ~/Downloads/Cloud Computing/Lab/Mini Project
$ python app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with watchdog (windowsapi)
* Debugger is active!
* Debugger PIN: 251-305-050
```




Upload a File to Secure Cloud Storage


Choose File No file chosen


Upload & Encrypt




 Sign in with Google

CC Mini Project wants access to your Google Account

 hyderpresswala@gmail.com

 CC Mini Project already has some access
See the [1 service](#) to which CC Mini Project has some access.

Make sure that you trust CC Mini Project

 [Learn why you're not seeing links to CC Mini Project's Privacy Policy or Terms of Service](#)





Review CC Mini Project's Privacy Policy and Terms of Service to understand how CC Mini Project will process and protect your data.

To make changes at any time, go to your [Google Account](#).

[Learn how Google helps you share data safely.](#)

Cancel

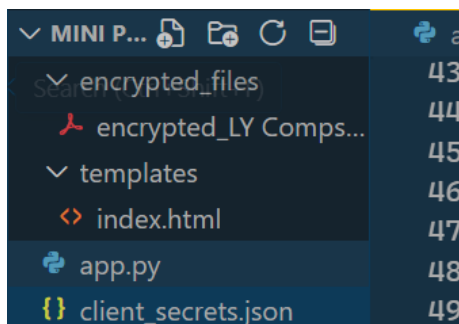
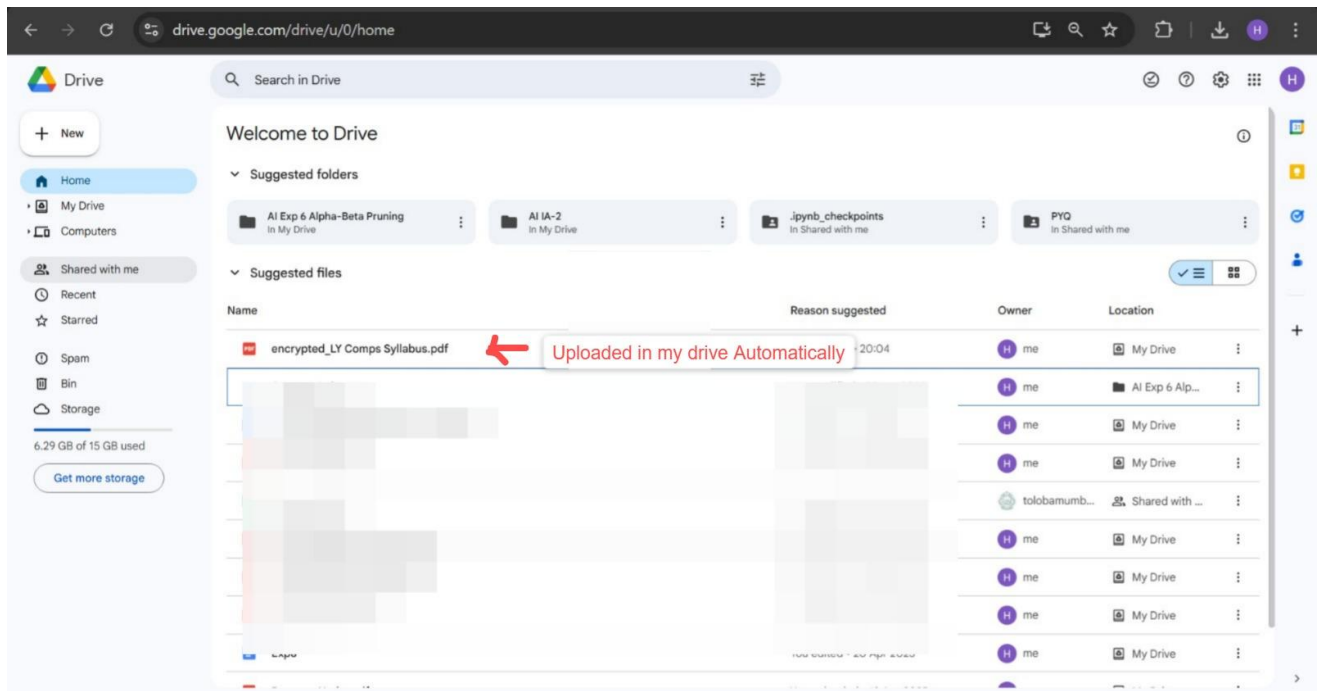
Continue

    localhost:8080/?code=4/0Ab_5qlkL17FpFpY-lLaB1swfFk3-p7UyIIbN0OLMVlb2wD3PKolob7e1L9lyqntbc4xQ&scope=ht

The authentication flow has completed.

    127.0.0.1:5000/upload

File "LY Comps Syllabus.pdf" encrypted and uploaded to Google Drive successfully.



File Saved in locally too

```
hyder@HyderPresswala MINGW64 ~/Downloads/Cloud Computing/Lab/Mini Project
$ python decrypt_file.py
Enter the encrypted file name (with extension): encrypted_LY Comps Syllabus.pdf
File decrypted successfully! Saved as decrypted_encrypted_LY Comps Syllabus.pdf
hyder@HyderPresswala MINGW64 ~/Downloads/Cloud Computing/Lab/Mini Project
$
```



7. Challenges and Solutions

- **Challenge 1 - OAuth Authentication:** The implementation of OAuth 2.0 can be tricky, especially handling the refresh tokens for long-term access. This was solved by using the appropriate libraries provided by Google and ensuring token management was done securely.
- **Challenge 2 - File Encryption/Decryption:** Ensuring that encryption/decryption is done efficiently without introducing delays in the upload/download process was a challenge. This was addressed by using optimized libraries and performing encryption operations asynchronously.

8. Conclusion

This project successfully demonstrates a secure file storage system using the Google Drive API. By combining Google Drive's cloud storage capabilities with strong encryption mechanisms and secure authentication, users can store their files in the cloud with the assurance that their data is protected from unauthorized access. The system offers a simple and efficient solution for secure file storage, making use of widely-used technologies and best practices in security.

9. Future Work

- **Expanded Encryption Options:** Implement additional encryption algorithms and allow users to choose the level of encryption they prefer.
- **Advanced User Management:** Implement features such as multi-user access control and collaborative file sharing with varying permission levels.
- **Improved User Interface:** Enhance the frontend to provide a more intuitive user experience and support mobile devices.

10. References

- Google Drive API Documentation: <https://developers.google.com/drive>
- OAuth 2.0 Documentation: <https://developers.google.com/identity/protocols/oauth2>
- AES Encryption in Python: <https://www.pycryptodome.org/src/cipher/aes>