Contents

1. Introduction

Welcome to our cutting-edge security platform, where advanced encryption meets innovative features to safeguard your data like never before. Our platform combines state-of-the-art encryption techniques with powerful tools such as face recognition for decryption, multi-cipher algorithms, network tracing, and geolocation tracking to provide unparalleled security and control over your sensitive information.

1.1 Technologies Used

- Streamlit
  - A framework for working with User interfaces of applications made with Python.
- Ipinfo
  - An API service provider for providing geolocations based on a user's public IP address.
- Postgres
  - An open source relational database.
  - Most popularly used with Python centric web-applications.
- Flask
  - A python framework to create servers to get and put information on.
- OpenCV
  - An open source Computer Vision library typically used for processing like Face detection through its pre-trained highly accurate models.
- Docker
  - A container management tool used to deal with containers, that can contain databases, systems virtually and without downloading into local device.
  - Here we have used it to run postgres database without physically downloading it into our device.

1.2 Platform Used

- Visual Studio Code - *IDE*
- Dbeaver - *Database viewer*

2. Functional Requirements

2.1 Authentication

1. Users can log in using one or more of the following methods:
   - Phone number
   - Username
   - Facial recognition

2.2 File Encryption

2. Users can choose any file from their device to encrypt.
3. Upon encryption, users must provide:
   - Name and password for the file
   - Unique identifiers (UUIDs) specifying the number of authorized devices that can access the file.

2.3 File Decryption

4. To decrypt a file, users must provide:
   - Name and password of the file
   - Facial recognition
   - Key for decryption
   - Verification that the computer is registered in the list of authorized UUIDs.

2.4 Access Control

5. The system maintains a list of UUIDs representing authorized devices.
6. Only devices with registered UUIDs can access encrypted files.

2.5 History Tracking 7. The system tracks access history, recording:

Geolocation information (accurate to the region)

Timestamp of each access event

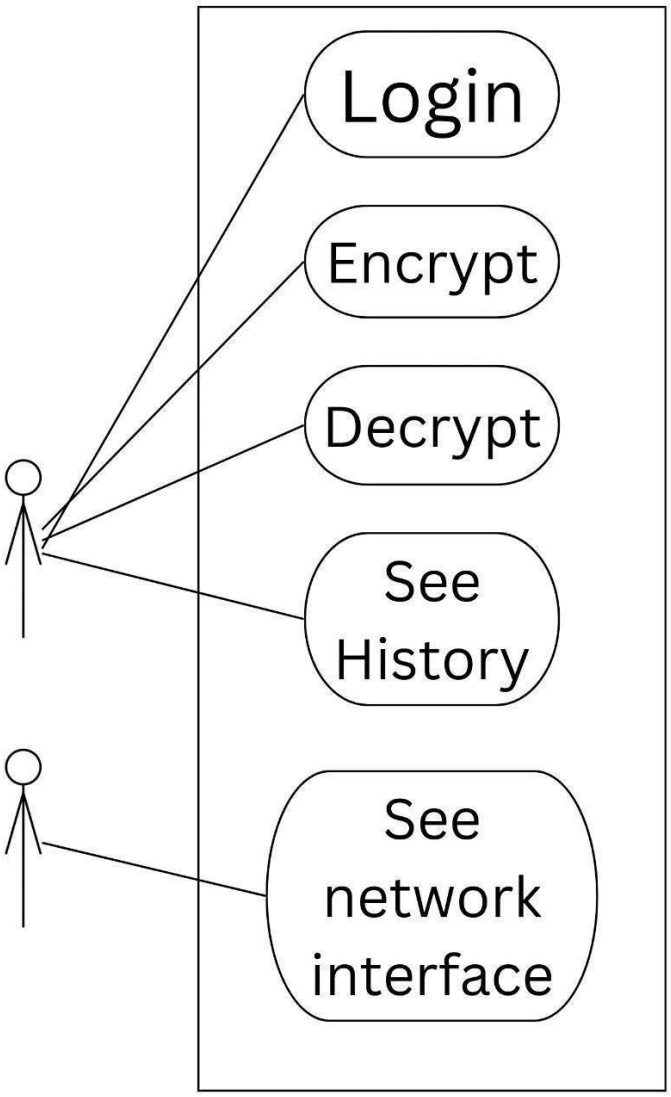
User identification (phone number, username, or facial recognition).

Non-functional Requirements

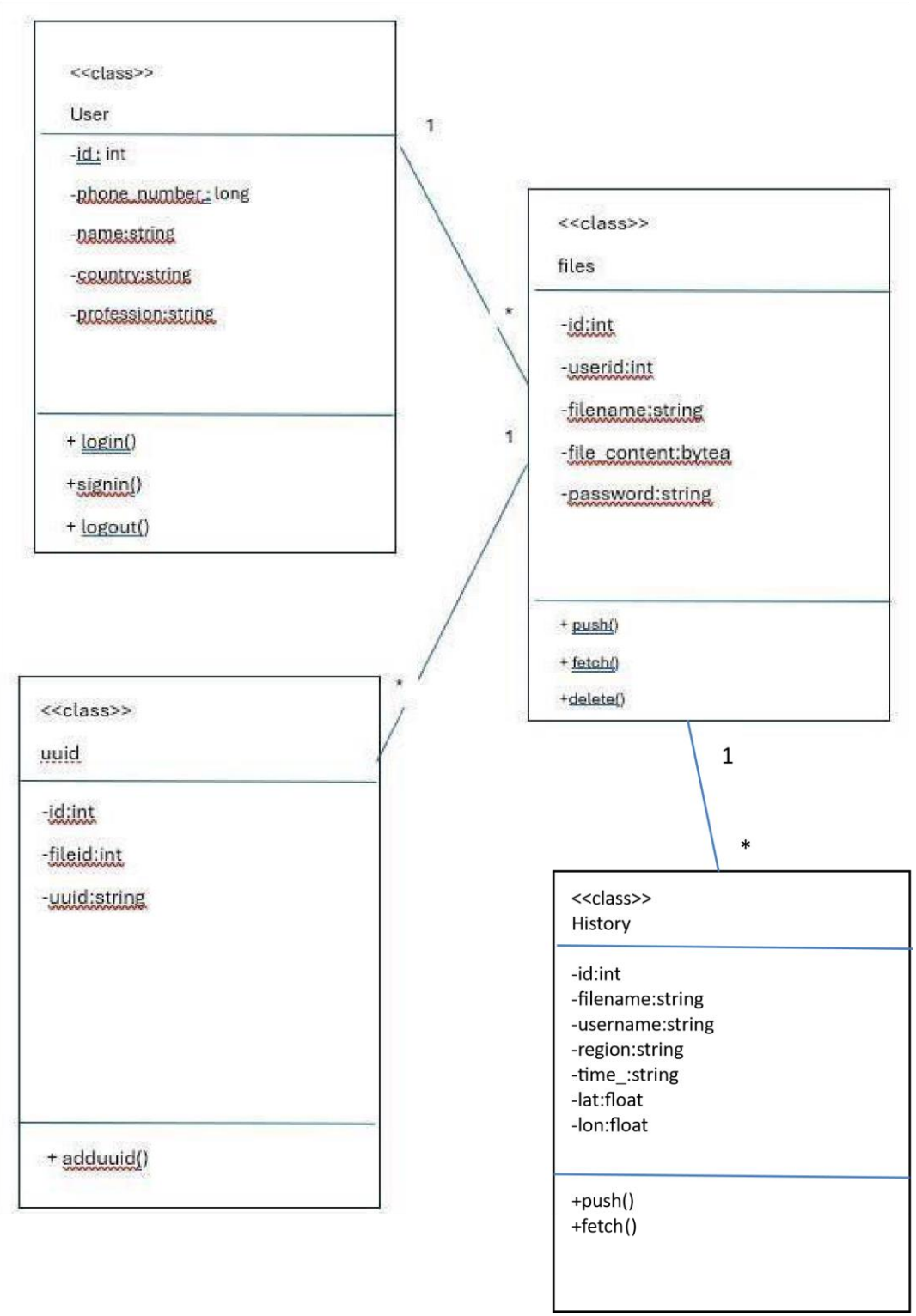
- **Security:** The system must employ robust encryption algorithms to ensure the confidentiality and integrity of encrypted files.
- **Usability:** The user interface should be intuitive and user-friendly, guiding users through the encryption and decryption process seamlessly.
- **Performance:** The system should perform encryption and decryption operations efficiently, minimizing processing time.
- **Reliability:** The system should be highly reliable, ensuring that access controls are enforced correctly and access history is accurately recorded.

3. System Design

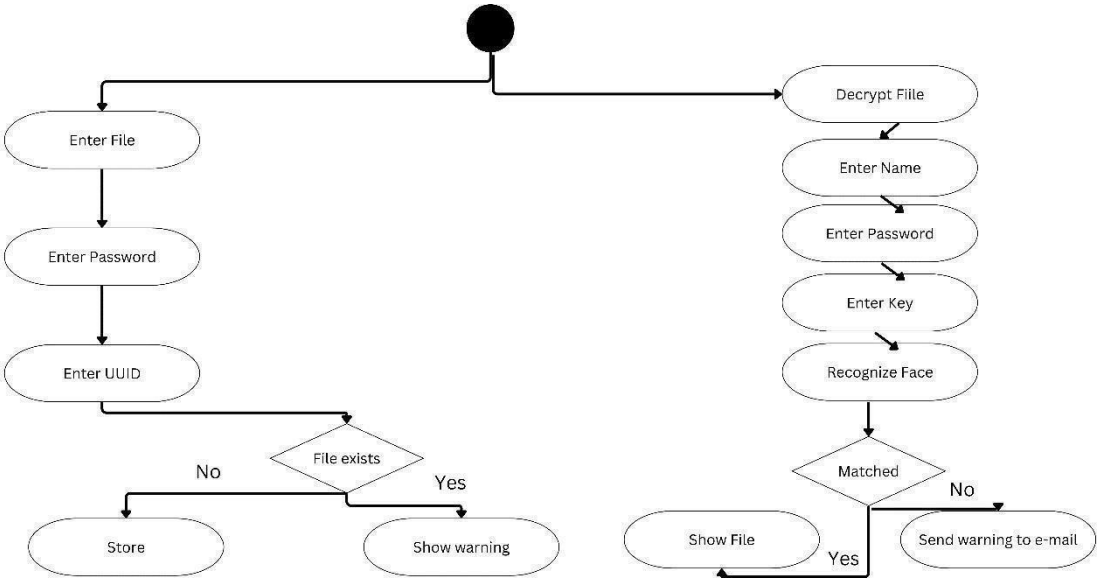
3.1 Use Case Diagram



3.2 Class Diagram

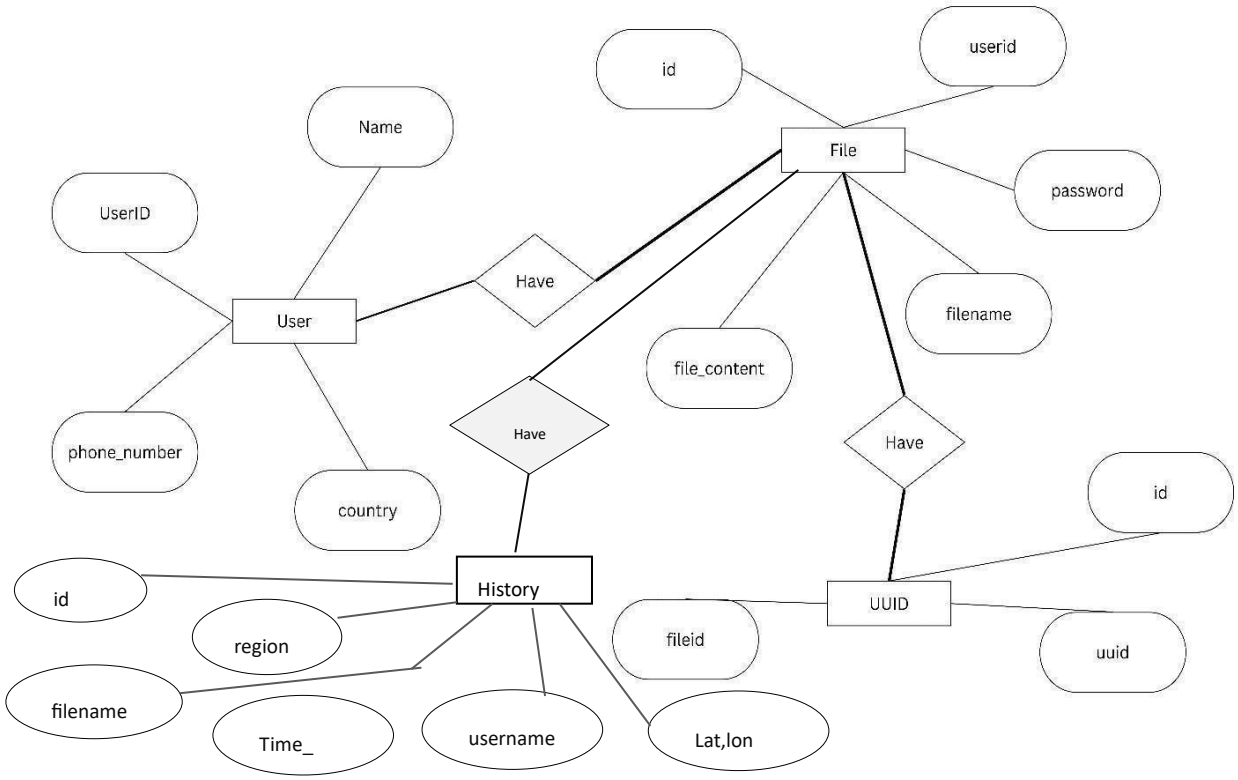


3.3 Activity Diagram



3.4 Sequence Diagram

3.5 ER Diagram



3.6 Data Dictionary

3.6.1 Users Table

Field Name	Data Type	Field Length	Constraint	Description
Userid	Int		Primary	User id
Name	String	255		User name
Phone_number	Long			Mobile number
country	String	255		Country name

3.6.2 Files Table

Field Name	Data Type	Field Length	Constraint	Description
id	Int		primary	File id
Userid	Int			File's user
password	String	255		File password
filename	String	255		File name
File_content	bytea			File content

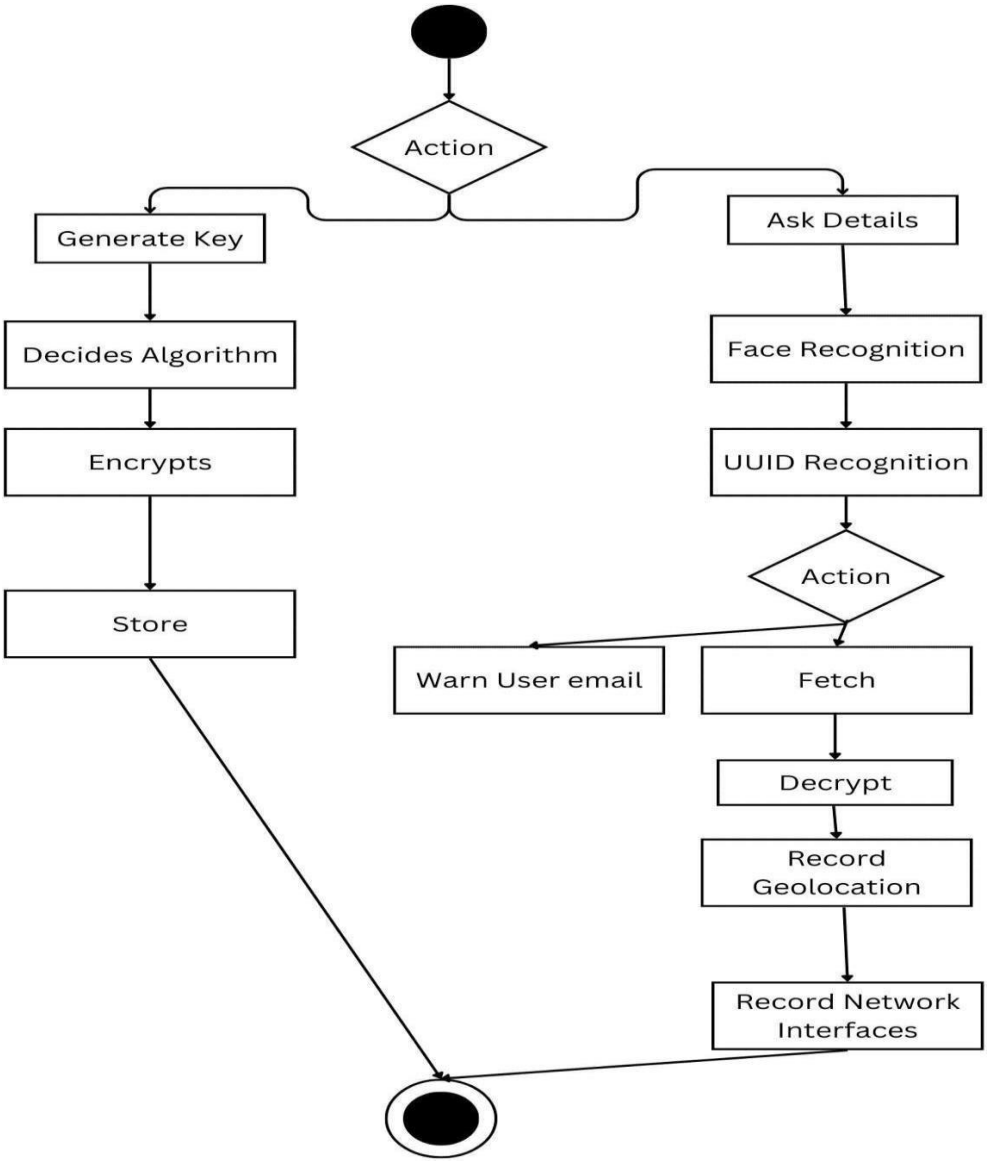
3.6.3 UUID Table

Field Name	Data Type	Field Length	Constraint	Description
id	Int		primary	Uuid id
fileid	Int			Uuid file
Uuid	String	255		Uuid

3.6.4 History Table

Field Name	Data Type	Field Length	Constraint	Description
id	Int		primary	History id
filename	string	255		History for file
username	String	255		User for history
region	String	255		Geolocation
Time_	String	255		Time accessed
lat	float			Latitude
lon	float			Longitude

3.7 Flow Chart



4 Implementation

For the implementation of the project, we have chosen to develop it using the Python programming language along with several libraries and frameworks to fulfill the requirements. Below are the key components and libraries used in the implementation:

4.1 Technologies and Libraries Used:

- **Python:** The core programming language used for implementing the backend logic and functionality of the system.
- **Flask:** Flask is used to develop the web application and RESTful APIs required for user interaction and data transmission.
- **OpenCV:** OpenCV library is utilized for facial recognition functionality, allowing users to log in using facial authentication.
- **UUID:** The UUID module in Python is used for generating unique identifiers (UUIDs) for authorized devices, enabling access control.

4.2 Modules and Brief Description:

Authentication Module:

- **Login with Phone Number:** Users can log in using their phone numbers, leveraging Flask's session management for user authentication.
- **Login with Username:** Alternatively, users can log in using their usernames, with authentication handled by Flask's authentication mechanisms.
- **Facial Recognition:** For advanced security, users have the option to log in using facial recognition. OpenCV is utilized for facial detection and recognition.

File Encryption Module:

- **Choose File for Encryption:** Users can select any file from their device for encryption, with the encryption process handled by our multi-cipher algorithm.
- **UUID Specification:** During encryption, users can specify the number of authorized devices that can access the encrypted file, with UUIDs generated using Python's UUID module.
- **Name and Password Protection:** Users must provide a name and password for the encrypted file, ensuring secure access.

File Decryption Module:

- **Decrypt File:** To decrypt a file, users must provide the name and password of the file, along with facial recognition for added security.
- **Access Control:** Access to decrypted files is restricted to authorized devices with matching UUIDs.

History Tracking Module:

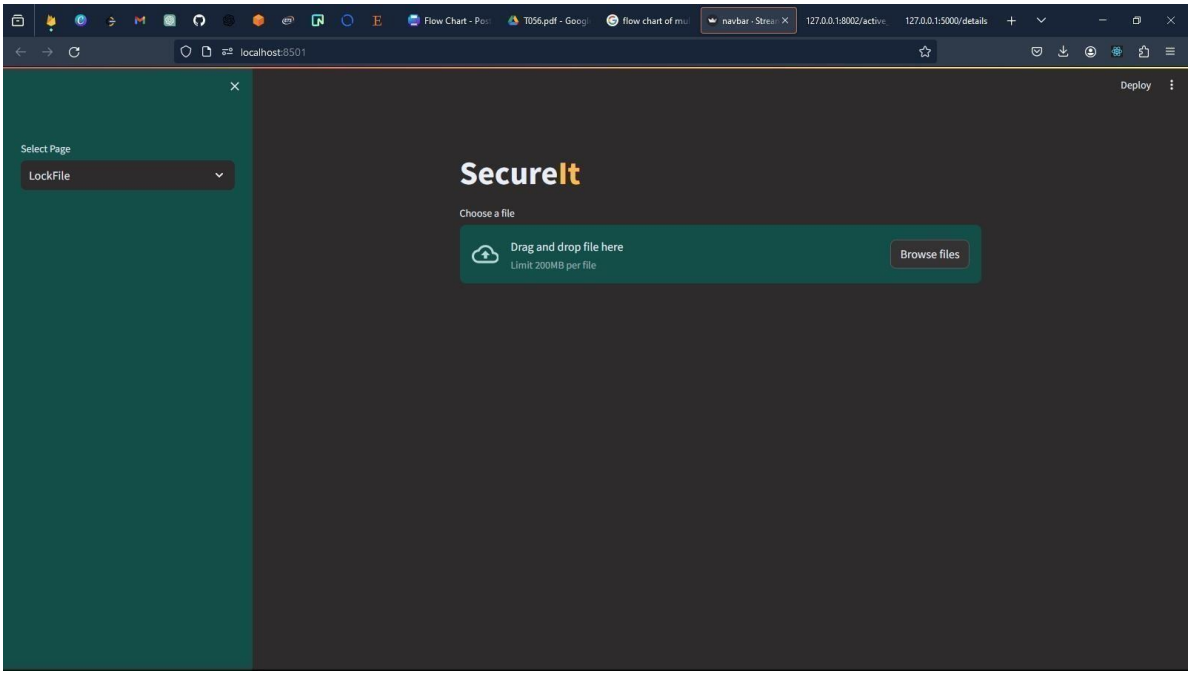
- **Access History:** The system maintains a log of access history, recording geolocation information and timestamps for each access event.
- **User Identification:** Access events are associated with user identification, such as phone number, username, or facial recognition data.

5 Testing

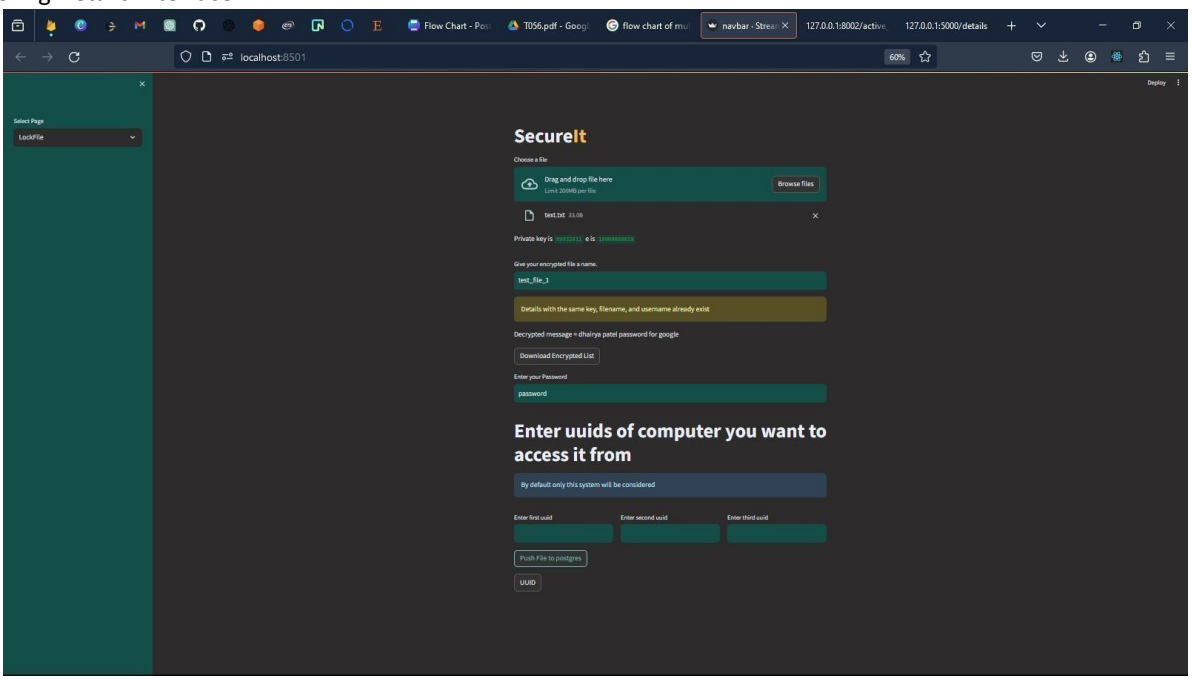
Ser. No.	Input	Output	Expected	Results
1	Wrong Credential Login	Displayed Warning	Display Warning	Pass
2	Correct Login	Entered Web-app	Enter Web-app	Pass
3	Wrong Password for File	Sent email warning	Send email warning	Pass
4	No Face Detected	Don't Display File	Don't Display File	Pass
5	Face Detected	Decrypt File	Decrypt File	Pass

6. Screenshots

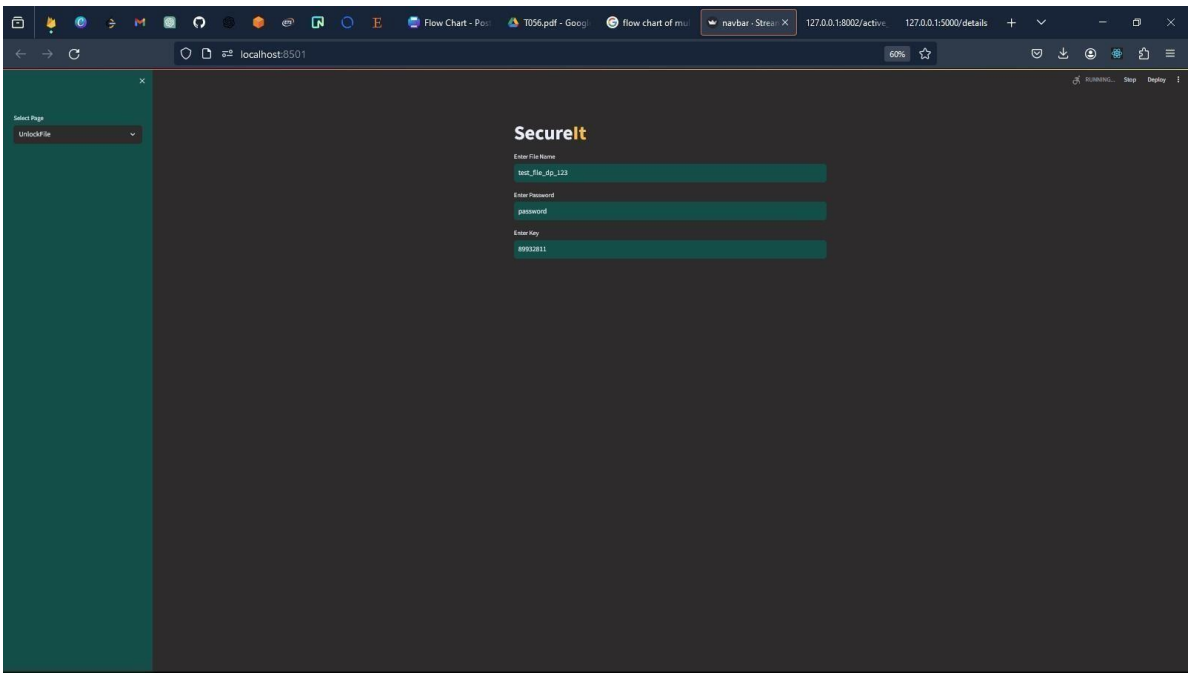
LockFile Interface



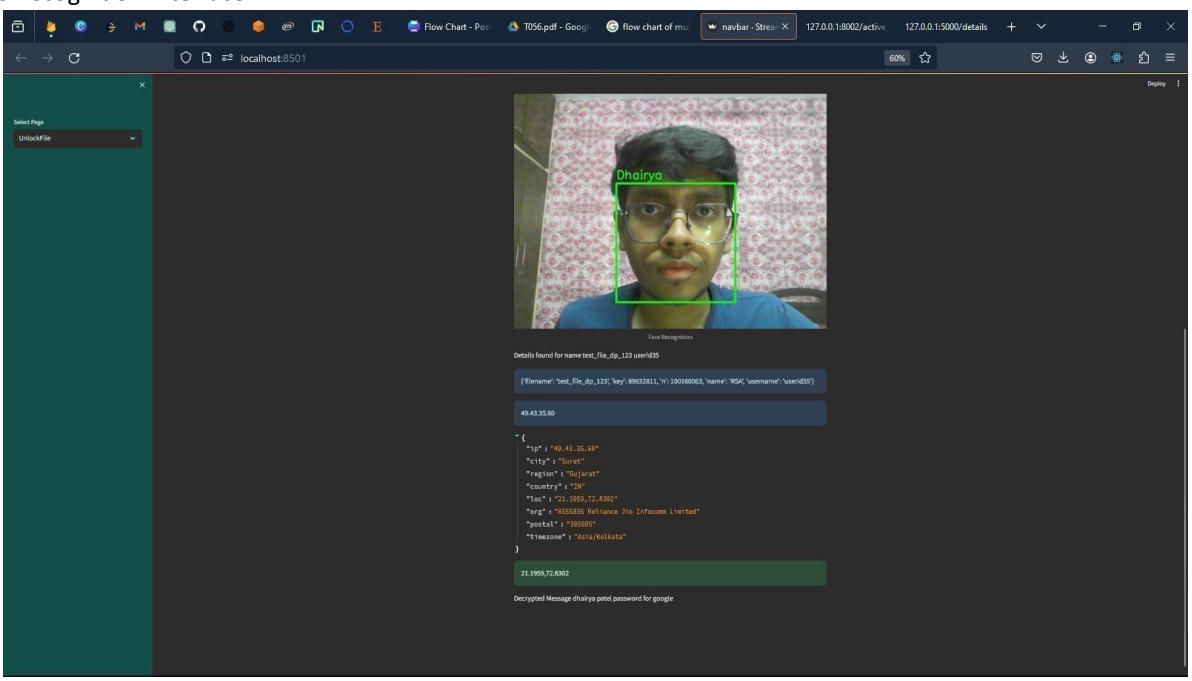
Entering Details Interface



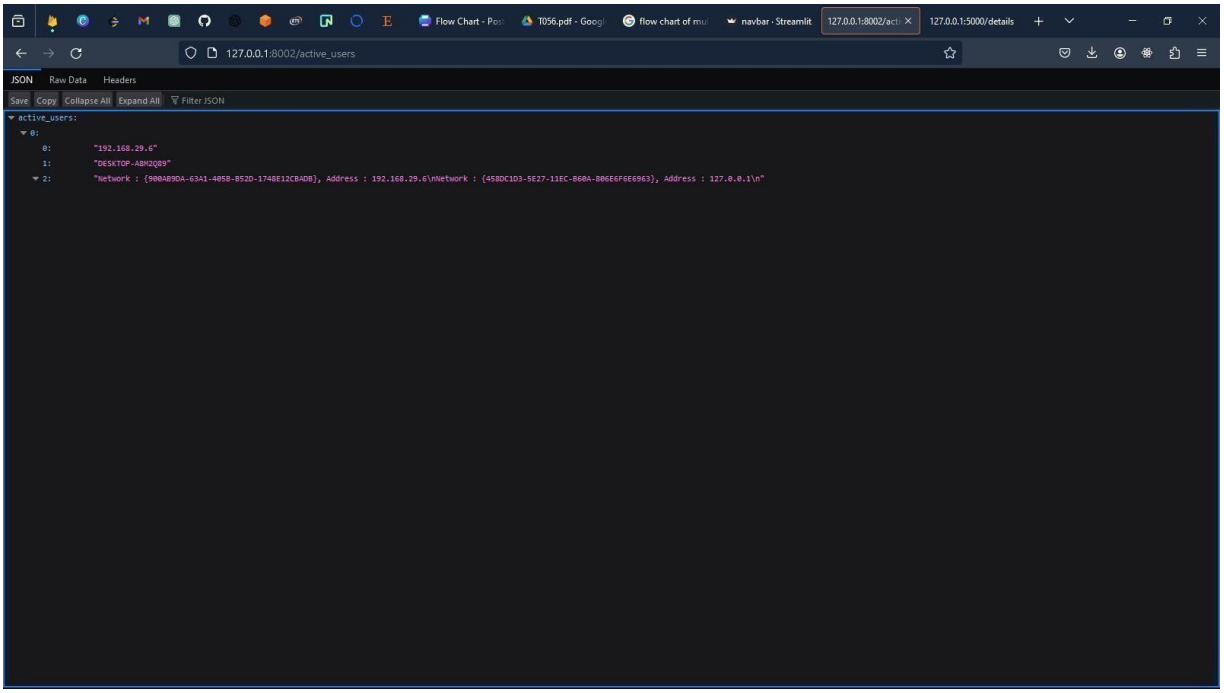
UnlockFile Interface



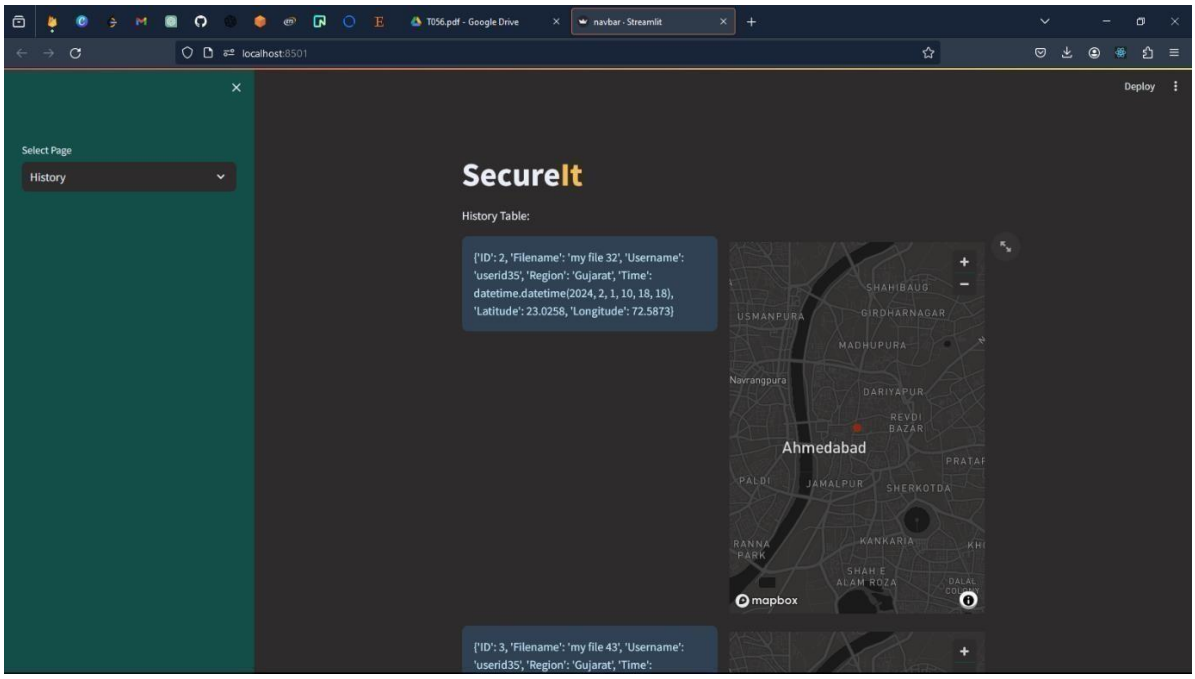
Face Recognition Interface



Seeing Active Network Interfaces



Seeing Geolocation Interface



7. Conclusion

This project aims to develop a comprehensive file encryption and decryption system that prioritizes security and user accessibility. It enables users to encrypt files with advanced authentication mechanisms such as login with phone numbers, usernames, or facial recognition. Additionally, users can specify access control parameters, including the number of authorized devices and unique identifiers (UUIDs) for secure access.

The implementation utilizes Python along with Flask for web development, OpenCV for facial recognition, and custom cipher systems for cryptographic operations, along with IPinfo for geolocation fetching. Through this combination of technologies and libraries, the system ensures robust security measures while providing a user-friendly interface for file encryption and decryption.

One of the main advantages of this system is its ability to simplify the encryption and decryption process, making it accessible to users with varying levels of technical expertise. Furthermore, the system enhances data security by incorporating facial recognition and access control features, ensuring that only authorized users and devices can access encrypted files.

In conclusion, the implemented application successfully fulfills the project requirements and demonstrates the effectiveness of using Python-based technologies for developing secure file encryption and decryption systems.

What we have learned –

Cryptography – How to implement cryptography algorithms and it's mathematical challenges.
Network security – Dealing with network interfaces and IP addresses.
Servers and API – How to host servers in Python and exchange information in it.
Postgres – How to integrate with any application.
Docker – How to connect to a database virtually. Opencv
– How to identify and detect faces

8. Limitations and Future Extensions

Currently, we are bounded by the interface to provide some industrial level interfaces and system. Due to which there is problem in generating continuous random keys, as the interfaces reloads every time any change is detected.

In future, we might take a robust framework for developing UI such as React/Angular. Also to fully use IP Info services and to host our postgres database, we need to invest some capital.

References

https://pypi.org/project/netifaces/
https://www.geeksforgeeks.org/rsa-algorithm-cryptography/amp/
https://opencv.org/opencv-facerecognition/