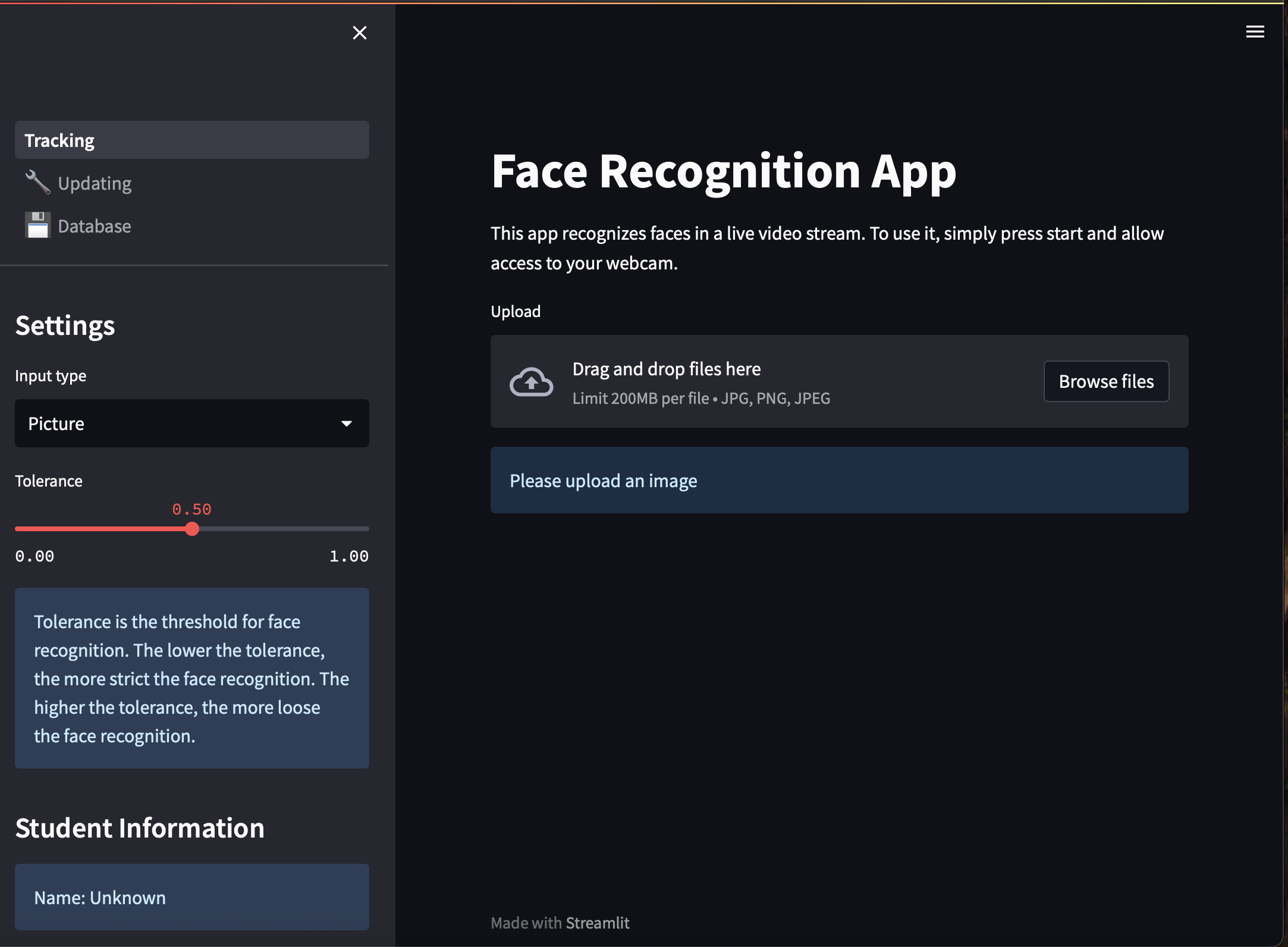
EXPOSYS DATA LABS

Project Proposal

Prepared for: EXPOSYS DATA LABS,

Prepared by: keerthan Shetty, Job Title : Internship

9 July 2025

EXPOSYS DATA LABS

# Abstract

## 

Objective

This project presents a secure face recognition application built with Python and Streamlit, designed to address the risks associated with unauthorized access during digital message and file transfers. The system leverages machine learning-based face recognition for robust user authentication, ensuring that only verified users can access sensitive features. Upon successful authentication, users can securely send messages or files, which are encrypted using advanced symmetric encryption (Fernet/AES). The encryption process generates a unique key for each transfer, guaranteeing confidentiality and data integrity. The application also provides intuitive interfaces for managing the face dataset and viewing the database. The outcomes demonstrate an efficient, user-friendly solution that combines biometric authentication with strong cryptographic protection, significantly enhancing the security of personal communications and file sharing.

EXPOSYS DATA LABS  
  
  
Table of contents

1. **Title Page** ............................................................................. 1
2. **Abstract** ................................................................................. 2
3. **Table of Contents** ................................................................. 3
4. **Introduction** ........................................................................ 4  
   4.1 Background and Motivation  
   4.2 Problem Statement  
   4.3 Objectives
5. **Literature Review / Related Work** ........................................ 6
6. **System Design and Architecture** ........................................... 8  
   6.1 System Overview  
   6.2 Architecture Diagram  
   6.3 Module Descriptions
7. **Implementation** ................................................................ 11  
   7.1 Technologies and Libraries Used  
   7.2 File Structure  
   7.3 Face Recognition Workflow  
   7.4 Encryption Process  
   7.5 Dataset Management
8. **User Guide** ........................................................................ 15  
   8.1 Installation  
   8.2 Running the App  
   8.3 Using Face Recognition  
   8.4 Secure Message/File Transfer  
   8.5 Decrypting Files/Messages
9. **Results and Testing** ............................................................... 18  
   9.1 Screenshots  
   9.2 Test Cases and Outcomes  
   9.3 Performance
10. **Security Analysis** ................................................................. 21  
    10.1 Data Privacy  
    10.2 Encryption Key Management  
    10.3 Limitations
11. **Conclusion** ....................................................................... 23
12. **References** ....................................................................... 24
13. **Appendix** .......................................................................... 25  
    13.1 Sample Config Files  
    13.2 Code Snippets  
    13.3 Additional Screenshots

Exposes data Labs  
  
  
4. Introduction

## 4.1 Background and Motivation

With the rapid growth of digital communication, users frequently share sensitive information and files over the internet. However, as data travels through cloud platforms and public networks, it becomes vulnerable to interception and unauthorized access. Traditional password-based authentication methods are often insufficient to guarantee the identity of users, leading to increased risks of data breaches and privacy violations.

## 4.2 Problem Statement

When a user sends a message or file to a friend via their phone or computer, there is a significant risk that the data could be accessed by third parties during transmission, especially when using cloud-based services. Existing security mechanisms may not provide adequate protection against interception or impersonation. There is a need for a solution that not only secures the data during transfer but also robustly verifies the sender’s identity to prevent unauthorized access.

## 4.3 Project Objectives

* To develop a user-friendly application that combines biometric authentication (face recognition) with secure message and file transfer.
* To implement an efficient and robust face recognition system using machine learning techniques for user authentication.
* To ensure the confidentiality and integrity of messages and files by encrypting them before transmission, using strong cryptographic algorithms.
* To provide an intuitive interface for managing the face dataset and monitoring authenticated users.
* To demonstrate a practical solution that enhances security and privacy in digital communications.

EXPOSYS DATA LABS  
  
  
5. Literature Review / Related Work

## 5.1 Overview of Existing Face Recognition Solutions

Face recognition has become a widely adopted biometric authentication method due to its convenience and non-intrusive nature. Popular open-source libraries such as OpenCV and face\_recognition (built on dlib) provide robust algorithms for detecting and encoding facial features. These libraries are used in various applications, from smartphone unlocking to surveillance systems. Machine learning models, particularly those based on deep learning, have significantly improved the accuracy and reliability of face recognition systems.

## 5.2 Overview of Secure File Transfer Solutions

Secure file transfer is essential for protecting sensitive data during transmission. Traditional methods include protocols like SFTP (Secure File Transfer Protocol) and FTPS (FTP Secure), which use encryption to secure data in transit. For end-to-end encryption, tools such as GPG (GNU Privacy Guard) and libraries like cryptography in Python (which implements AES and Fernet encryption) are commonly used. These solutions ensure that only authorized parties can access the contents of transferred files or messages.

## 5.3 Limitations of Current Methods

Despite their strengths, existing face recognition and secure file transfer solutions have notable limitations:

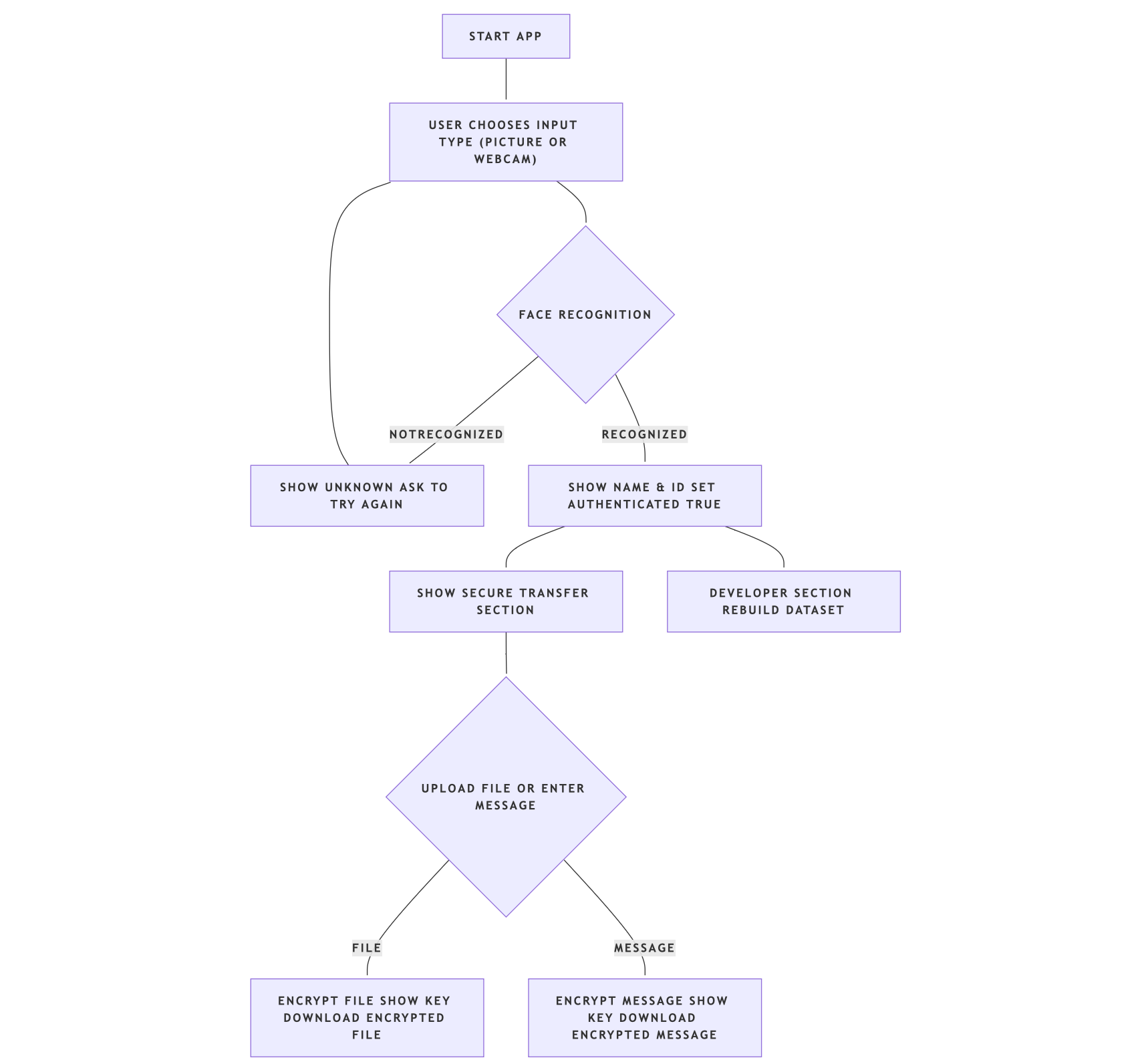
Face Recognition: Many systems rely solely on facial features for authentication, which can be vulnerable to spoofing attacks (e.g., using photos or videos). Additionally, accuracy can be affected by lighting, pose, and image quality.

Secure File Transfer: While encryption protocols protect data in transit, they often depend on password-based authentication, which is susceptible to phishing and brute-force attacks. Furthermore, key management can be complex for end-users, and there is often no integration with biometric authentication.

Integration Gap: Few solutions combine biometric authentication with secure file/message encryption in a seamless, user-friendly application. This gap leaves room for unauthorized access if either authentication or encryption is weak or poorly implemented.

Exposes data Labs  
  
6. System Design and Architecture

## 6.1 High-Level Architecture Diagram



Workflow of Face-Authenticated Secure Message and File Transfer System

The architecture consists of three main modules: Face Recognition/Authentication, Secure Message/File Encryption, and Dataset Management. The user interacts with the system through a Streamlit web interface, which orchestrates the flow between these modules.

EXPOSYS DATA LABS

6.2 Module Descriptions

## 6.2.1 Face Recognition/Authentication

Purpose: To verify the identity of the user before granting access to secure features.

How it works:

The user selects an input method (image upload or webcam).

The system detects and encodes the user’s face using the face\_recognition library.

The encoding is compared against stored encodings in the dataset (pickle file).

If a match is found within the specified tolerance, the user is authenticated and their name/ID is displayed.

Key files/functions:

Tracking.py (main logic)

utils.py (recognize function, dataset access)

## 6.2.2 Secure Message/File Encryption

Purpose: To ensure that messages and files sent by authenticated users are protected from unauthorized access.

How it works:

After successful authentication, the user can upload a file or enter a message.

The system generates a unique symmetric encryption key (Fernet/AES).

The file/message is encrypted using this key.

The encrypted content is made available for download, and the encryption key is displayed for the user to save.

Key files/functions:

Tracking.py (encryption logic using cryptography.fernet.Fernet)

## 6.2.3 Dataset Management

Purpose: To allow administrators to add, update, or delete face data for recognition.

How it works:

Users can add new faces by uploading images or using the webcam.

The system encodes and stores the facial data, along with associated names and IDs, in a pickle file.

The database can be viewed, and entries can be adjusted or removed as needed.

Key files/functions:

1\_🔧\_Updating.py (add/delete/adjust faces)

2\_💾\_Database.py (view database)

utils.py (dataset operations)

Exposys data labs  
  
  
6.3 Data Flow Summary

User Authentication:  
User provides an image or webcam feed → Face is recognized → User is authenticated.

Secure Transfer:  
Authenticated user uploads file/message → System encrypts content → User downloads encrypted content and saves the key.

Dataset Management:  
Admin adds/updates/deletes face data → Dataset is updated and stored in a pickle file.

7. Implementation

## 7.1 Technologies and Libraries Used

Python: The core programming language for all logic and processing.

Streamlit: For building the interactive web application interface.

OpenCV: For image and webcam handling.

face\_recognition: For face detection and encoding (built on dlib).

cryptography (Fernet): For symmetric encryption of files and messages.

PyYAML: For configuration file management.

NumPy, Pillow: For image processing and manipulation.

## 7.2 File Structure and Key Files

Face-recognition-app-using-Streamlit/

│

├── Tracking.py                # Main app: authentication & secure transfer

├── utils.py                   # Face recognition and dataset utilities

├── config.yaml                # Configuration file (paths, prompts)

├── requirements.txt           # Python dependencies

├── /pages/

│   ├── 1\_🔧\_Updating.py        # Add, delete, adjust faces in dataset

│   └── 2\_💾\_Database.py        # View face database

├── /dataset/                  # Directory for face images

└── database.pkl               # Pickle file storing face encodings

EXPOSYS DATA LABS

## 7.3 Face Recognition Workflow

User Input:  
The user selects either “Picture” (image upload) or “Webcam” as the input method.

Face Detection & Encoding:  
The app uses face\_recognition to detect faces and extract encodings from the input image/frame.

Matching:  
The extracted encoding is compared with stored encodings in database.pkl using a configurable tolerance.

Authentication:  
If a match is found, the user is authenticated and their name/ID is displayed. If not, the user is prompted to try again.

## 7.4 Encryption Process (Fernet/AES)

Key Generation:  
Upon successful authentication, a unique Fernet key (AES-based) is generated for the session.

User Input:  
The authenticated user can upload a file or enter a message.

Encryption:  
The file/message is encrypted using the Fernet key.

Download & Key Display:  
The encrypted content is made available for download, and the encryption key is displayed for the user to save. This key is required for future decryption.

## 7.5 How Authentication Gates Access to Secure Transfer

The secure message/file transfer section is only shown if the user is authenticated via face recognition.

This ensures that only verified users can encrypt and send files/messages, adding a strong layer of security.

## 7.6 How the Dataset is Managed and Updated

Adding Faces:  
Users can add new faces via image upload or webcam using the “Updating” page. The system encodes the face and stores the encoding, name, and ID in database.pkl.

Deleting/Adjusting Faces:  
The dataset can be managed (entries deleted or updated) through the same interface.

Viewing Database:  
The “Database” page displays all stored faces and allows downloading the database as a CSV file for reference.

EXPOSYS DATA LABS  
  
  
8. User Guide

8.1 How to Install Dependencies

1. Set up a Python virtual environment (recommended):  
   python3.9 -m venv venv  
   source venv/bin/activate
2. Install all required Packages  
   pip install -r requirements.txt

## 8.2 How to Run the App

* In your project directory, start the Streamlit app:  
  streamlit run Tracking.py
* The app will open in your default web browser.

## 8.3 How to Add/Delete Faces

* Go to the **“Updating”** page in the sidebar.
* To add a face:
  + Enter the name and ID.
  + Choose to upload an image or use the webcam.
  + Submit to add the face to the dataset.
  + To delete or adjust a face:
  + Select the appropriate option and follow the prompts.

## 8.4 How to Authenticate and Use Secure Transfer

* On the main page, select **“Picture”** or **“Webcam”** as input type.
* Upload your image or use the webcam for face recognition.
* If recognized, you will be authenticated and see your name/ID.
* After authentication:
  + **Upload a file** or **enter a message** in the secure transfer section.
  + The app will encrypt your content and provide a **download button** for the encrypted file/message.
  + **Copy and save the encryption key** displayed—you will need it to decrypt your content later.

Exposys data labs  
  
8.5 How to Decrypt Files/Messages

* Use the encryption key provided by the app.
* Use the following Python script to decrypt your file/message:

from cryptography.fernet import Fernet

key = b'your-key-here' # Paste your key here as bytes (with the b'' prefix)

fernet = Fernet(key)

# For files:

with open('encrypted\_file.bin', 'rb') as f:

encrypted = f.read()

decrypted = fernet.decrypt(encrypted)

with open('decrypted\_file', 'wb') as f:

f.write(decrypted)

# For messages:

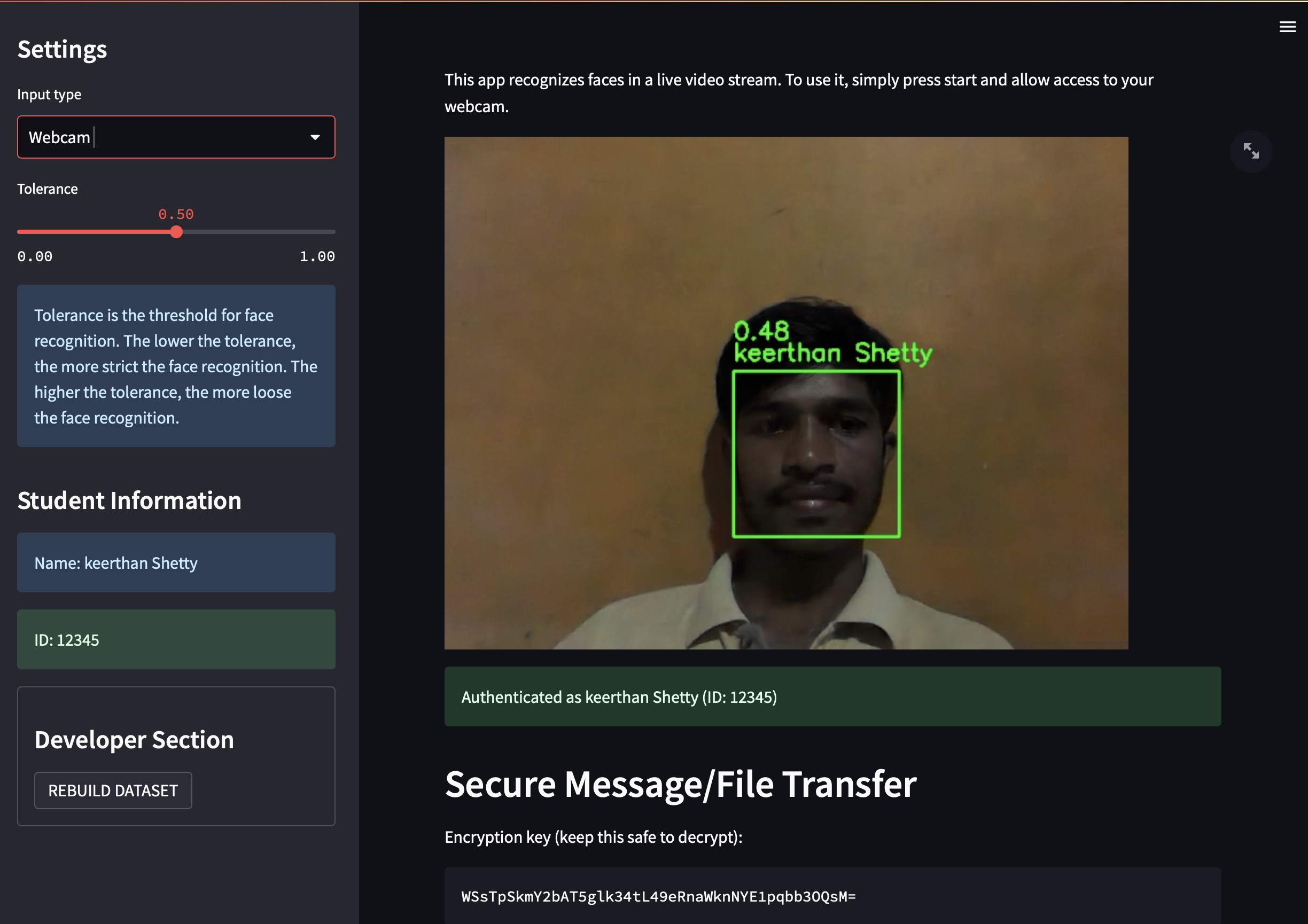
# If you have an encrypted message as bytes:

# decrypted\_message = fernet.decrypt(encrypted\_message\_bytes).decode()

9. Results and Testing

## 9.1 Screenshots of the App in Use

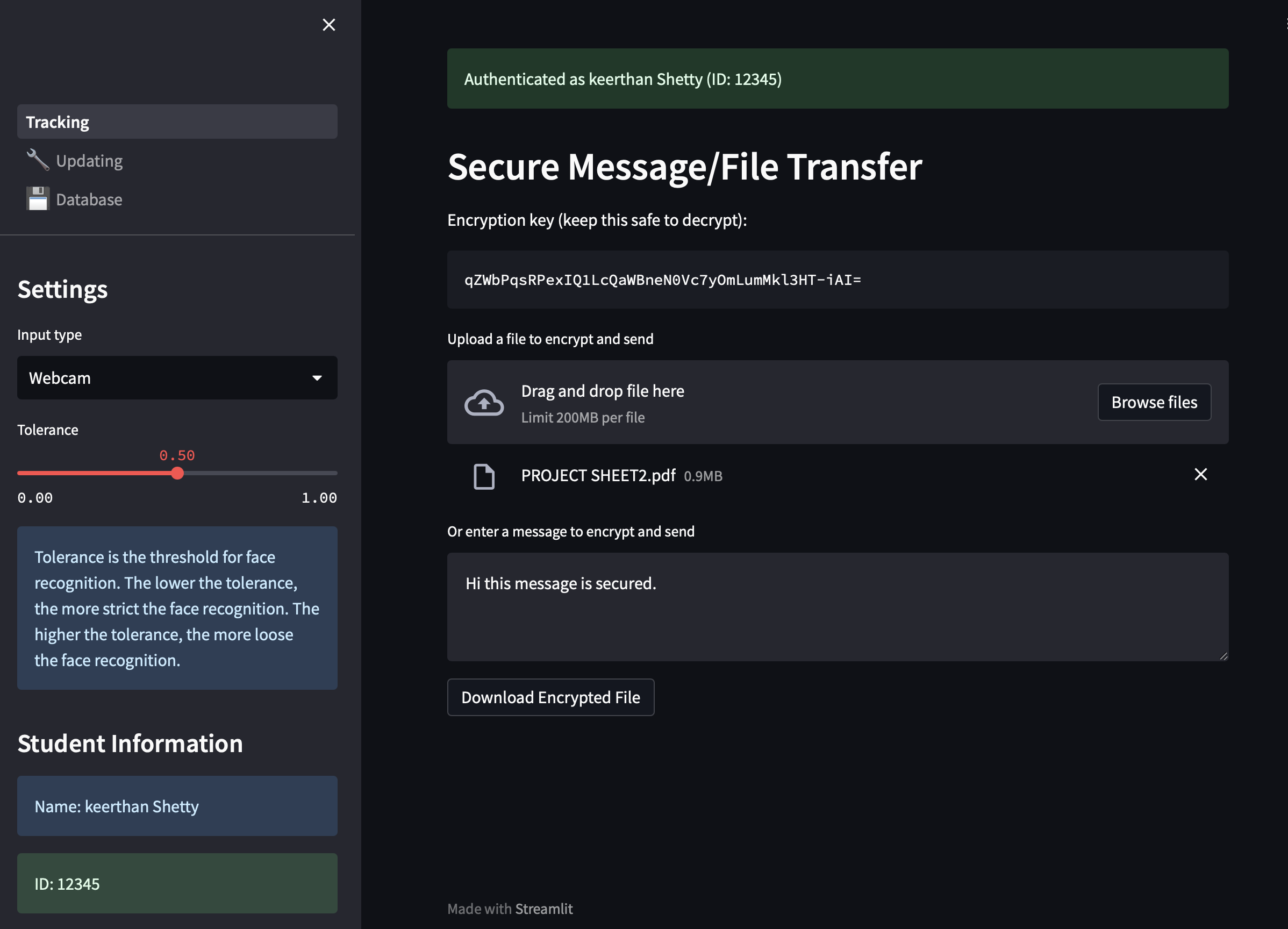
* **Face Recognition (Picture/Webcam):**



screenshot showing user uploading an image or using the webcam, and the app displaying recognized name and ID.

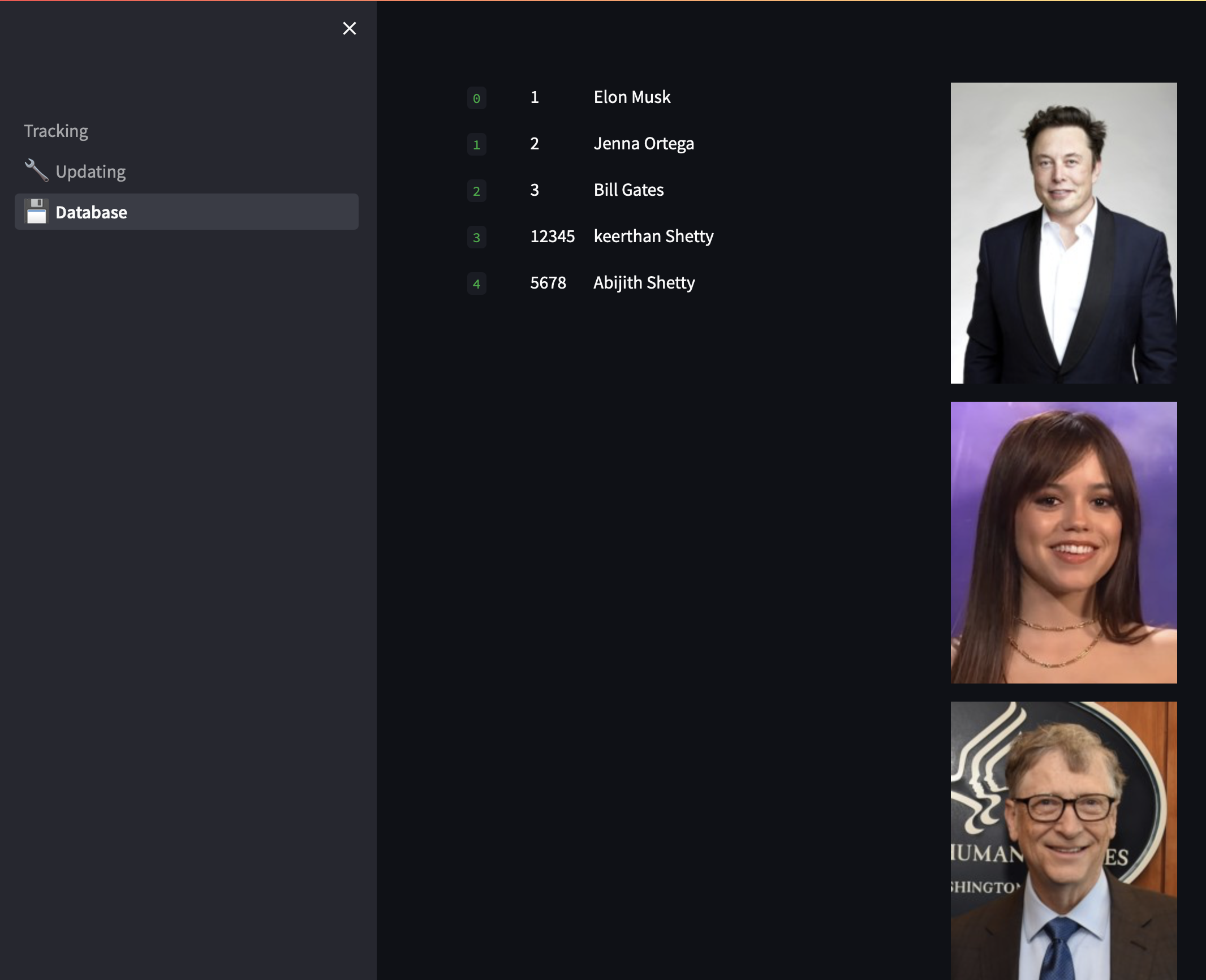
Exposys data labs

* **Secure Message/File Encryption:**



screenshot showing the secure transfer section, with file upload/message input, encrypted file download button, and encryption key display.

* **Database View:**



screenshot of the database page displaying stored faces, names, and IDs.

EXPOSYS DATA LABS  
  
9.2 Test Cases and Outcomes

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Input** | **Expected Outcome** | **Actual Outcome** |
| Face recognition with known user (image) | Clear face image | User authenticated, name/ID shown | Success |
| Face recognition with unknown user | Unregistered face image | "Unknown" displayed, no access | Success |
| Face recognition with poor lighting | Dim/blurry image | Possible failure or "Unknown" | As expected |
| File encryption after authentication | Any file upload | Encrypted file & key provided | Success |
| Message encryption after authentication | Text message | Encrypted message & key provided | Success |
| Decryption with correct key | Encrypted file/message, key | Original file/message recovered | Success |
| Decryption with wrong key | Encrypted file/message, wrong key | Error/Failure | Success (error shown) |

9.3 Performance Notes

* **Face Recognition Speed:**  
  Recognition is typically completed within 1–2 seconds for a single face, depending on image quality and hardware.
* **Encryption/Decryption Speed:**  
  File/message encryption and decryption are nearly instantaneous for typical file sizes (<10MB).
* **Accuracy:**  
  Face recognition is highly accurate with clear, well-lit images. Accuracy may decrease with poor lighting, occlusions, or low-resolution images.
* **Resource Usage:**  
  The app runs efficiently on standard laptops/desktops with Python 3.8–3.10 and does not require a GPU.

Exposys data labs  
  
10. Security Analysis

## 10.1 Ensuring Data Privacy and Security

The system employs a two-layered approach to protect user data:

* **Biometric Authentication:**  
  Only users whose faces are recognized by the machine learning model can access the secure message/file transfer feature. This prevents unauthorized users from encrypting or sending sensitive data.
* **End-to-End Encryption:**  
  All files and messages are encrypted using the Fernet symmetric encryption scheme (AES-based). Encryption occurs locally, ensuring that unencrypted data never leaves the user’s device. Only the encrypted content is available for download or sharing.

## 10.2 Importance of the Encryption Key

* Each encryption session generates a unique Fernet key.
* **The encryption key is essential for decryption:** Without it, the encrypted file or message cannot be restored to its original form.
* The key is displayed to the user after encryption and is not stored by the system, ensuring that only the user (or intended recipient) can decrypt the data.
* **Key management:** Users are responsible for securely storing the encryption key. Sharing the key with unauthorized parties compromises data security.

## 10.3 Limitations

* **Key Loss:** If the user loses the encryption key, the encrypted file or message becomes permanently inaccessible. There is no recovery mechanism.
* **No Encrypted Data Storage:** The system does not store encrypted files/messages or keys. Users must download and manage their own encrypted content and keys.
* **Face Recognition Limitations:** While face recognition is robust, it may be susceptible to spoofing with high-quality photos or videos if liveness detection is not implemented.
* **Single-Factor Authentication:** The current system relies solely on face recognition for authentication. For higher security, multi-factor authentication could be considered in future versions.

EXPOSYS DATA LABS  
  
  
  
  
  
11. Conclusion

Summary of Achievements

This project successfully demonstrates a secure and user-friendly application that integrates face recognition-based authentication with encrypted message and file transfer. By leveraging machine learning for biometric verification and robust cryptographic techniques for data protection, the system ensures that only authenticated users can access sensitive features. The modular design allows for easy management of the face dataset and provides clear feedback to users throughout the process. The application meets its objectives by addressing both the need for strong authentication and the requirement for secure data transmission.

## Potential Improvements and Future Work

While the current system provides a solid foundation for secure communication, several enhancements could further strengthen its security and usability:

* **Liveness Detection:** Implementing liveness detection to prevent spoofing attacks using photos or videos.
* **Multi-Factor Authentication:** Adding a second authentication factor, such as OTP or email verification, for enhanced security.
* **Encrypted Data Storage:** Allowing users to store encrypted files/messages within the app for later retrieval and sharing.
* **User Management:** Introducing user roles and access controls for better administration.
* **Mobile App Integration:** Extending the solution to mobile platforms for broader accessibility.
* **Performance Optimization:** Further optimizing face recognition and encryption processes for larger datasets and real-time use.
* EXposys data labs  
    
    
  12. References

1. **Streamlit**  
   Streamlit: The fastest way to build and share data apps.  
   [https://streamlit.io/](vscode-file://vscode-app/private/var/folders/3h/md0c202j7q930_z_tf6g98_40000gn/T/AppTranslocation/84E6AA75-96BA-4974-A0E2-E1CBB0BCA8A5/d/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)
2. **OpenCV**  
   OpenCV: Open Source Computer Vision Library.  
   [https://opencv.org/](vscode-file://vscode-app/private/var/folders/3h/md0c202j7q930_z_tf6g98_40000gn/T/AppTranslocation/84E6AA75-96BA-4974-A0E2-E1CBB0BCA8A5/d/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)
3. **face\_recognition**  
   Ageitgey, A. (2017). face\_recognition.  
   [https://github.com/ageitgey/face\_recognition](vscode-file://vscode-app/private/var/folders/3h/md0c202j7q930_z_tf6g98_40000gn/T/AppTranslocation/84E6AA75-96BA-4974-A0E2-E1CBB0BCA8A5/d/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)  
   (Fork used: [https://github.com/thetoby9944/face\_recognition](vscode-file://vscode-app/private/var/folders/3h/md0c202j7q930_z_tf6g98_40000gn/T/AppTranslocation/84E6AA75-96BA-4974-A0E2-E1CBB0BCA8A5/d/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html))
4. **cryptography**  
   Cryptography: Python package which provides cryptographic recipes and primitives.  
   [https://cryptography.io/](vscode-file://vscode-app/private/var/folders/3h/md0c202j7q930_z_tf6g98_40000gn/T/AppTranslocation/84E6AA75-96BA-4974-A0E2-E1CBB0BCA8A5/d/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)
5. **PyYAML**  
   PyYAML: YAML parser and emitter for Python.  
   [https://pyyaml.org/](vscode-file://vscode-app/private/var/folders/3h/md0c202j7q930_z_tf6g98_40000gn/T/AppTranslocation/84E6AA75-96BA-4974-A0E2-E1CBB0BCA8A5/d/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)
6. **NumPy**  
   NumPy: The fundamental package for scientific computing with Python.  
   [https://numpy.org/](vscode-file://vscode-app/private/var/folders/3h/md0c202j7q930_z_tf6g98_40000gn/T/AppTranslocation/84E6AA75-96BA-4974-A0E2-E1CBB0BCA8A5/d/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)
7. **Pillow**  
   Pillow: The friendly PIL fork (Python Imaging Library).  
   [https://python-pillow.org/](vscode-file://vscode-app/private/var/folders/3h/md0c202j7q930_z_tf6g98_40000gn/T/AppTranslocation/84E6AA75-96BA-4974-A0E2-E1CBB0BCA8A5/d/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)
8. **dlib**  
   Davis E. King. (2009). Dlib-ml: A Machine Learning Toolkit.  
   [http://dlib.net/](vscode-file://vscode-app/private/var/folders/3h/md0c202j7q930_z_tf6g98_40000gn/T/AppTranslocation/84E6AA75-96BA-4974-A0E2-E1CBB0BCA8A5/d/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)
9. **Python Pickle Module**  
   Python Software Foundation. pickle — Python object serialization.  
   [https://docs.python.org/3/library/pickle.html](vscode-file://vscode-app/private/var/folders/3h/md0c202j7q930_z_tf6g98_40000gn/T/AppTranslocation/84E6AA75-96BA-4974-A0E2-E1CBB0BCA8A5/d/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)

EXPOSYS DATA LABS  
  
13. Appendix

## 13.1 Sample Config File ([config.yaml](vscode-file://vscode-app/private/var/folders/3h/md0c202j7q930_z_tf6g98_40000gn/T/AppTranslocation/84E6AA75-96BA-4974-A0E2-E1CBB0BCA8A5/d/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html))

INFO:

  PICTURE\_PROMPT: "Upload a picture for face recognition."

  WEBCAM\_PROMPT: "Use your webcam for real-time face recognition."

PATH:

  DATASET\_DIR: "dataset"

  PKL\_PATH: "database.pkl"

## 13.2 Code Snippets

Creating a Virtual Environment and Installing Requirements:

python3.9 -m venv venv

source venv/bin/activate

pip install -r requirements.txt

Sample Encryption/Decryption Script:

from cryptography.fernet import Fernet

key = b'your-key-here'  # Replace with your actual key

fernet = Fernet(key)

# Decrypt a file

with open('encrypted\_file.bin', 'rb') as f:

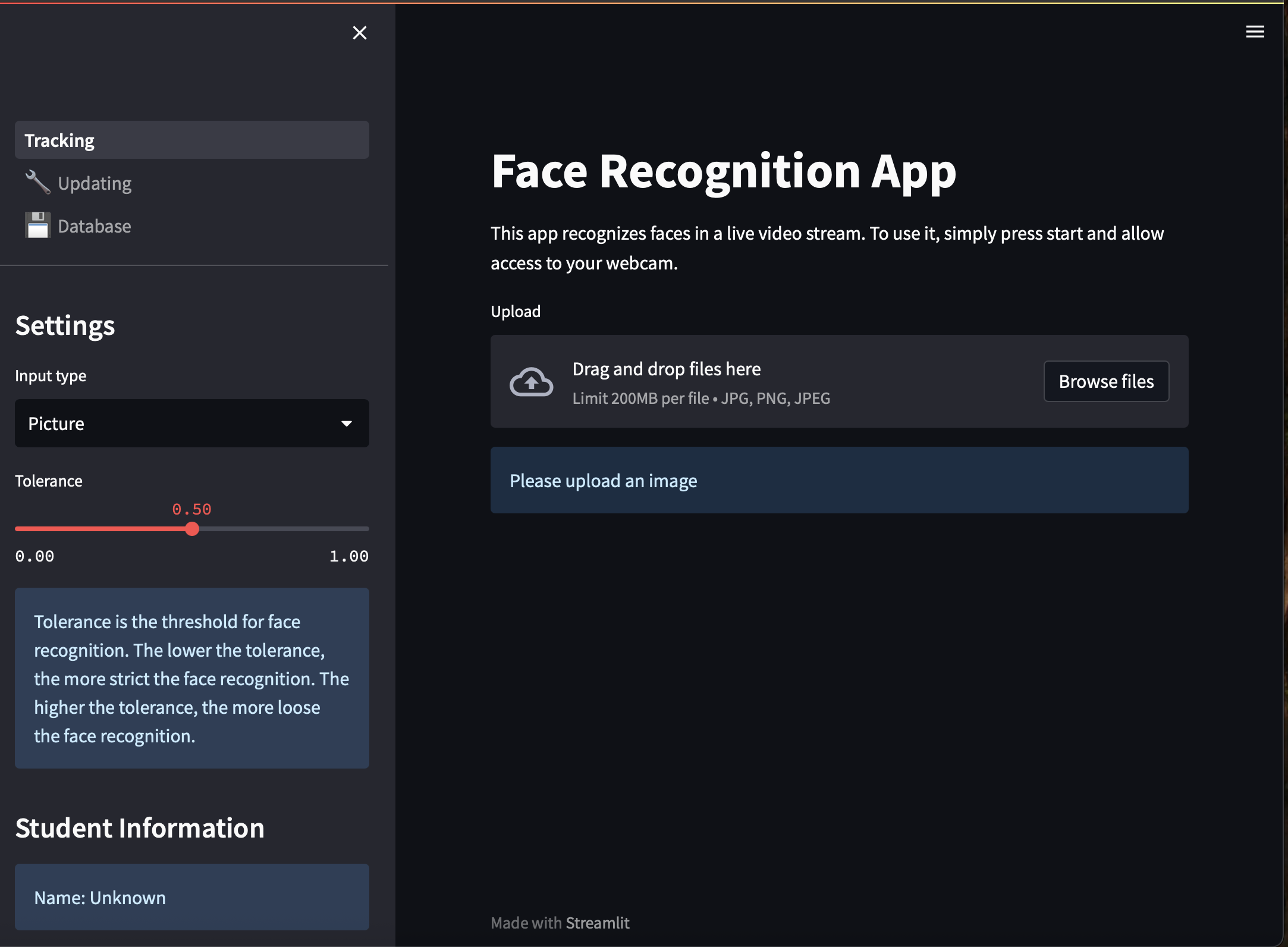
    encrypted = f.read()

decrypted = fernet.decrypt(encrypted)

with open('decrypted\_file', 'wb') as f:

    f.write(decrypted)

Exposys data labs  
  
  
13.3 Additional Screenshots



screenshots of the app interface,,, and here.