One time view file transfer

Submitted by

Manvitha R Kabbathi

USN:1RVU22CSE101

School of Computer Science

RV University

Submitted to

SHEBA PARI

Designation: Professor

School of Computer Science

RV University

Submission date [07.04.25]

1. **Abstract**

This project implements a secure, one-time file transfer system where files can be seen only once after sending(similar to whatsapp one view). The system uses a hybrid encryption approach combining RSA and AES to ensure data confidentiality. It features a GUI-based Python client (macOS) and a multithreaded C server (Ubuntu) with MongoDB Atlas integration for user management and file tracking. After a successful send and decryption, the file is manually-deleted by pressing enter. This system demonstrates secure communication, one-time file view access enforcement, and clean encryption design.

1. **Introduction**

In a world of growing digital data exchange, ensuring secure, private, and controlled access to files is critical. One-time download systems are used in secure messaging, confidential data sharing, and password delivery. While this project is not IoT or Edge Computing-based, it shares similar concerns with those fields: security, data integrity, and efficient transfer between distributed systems.

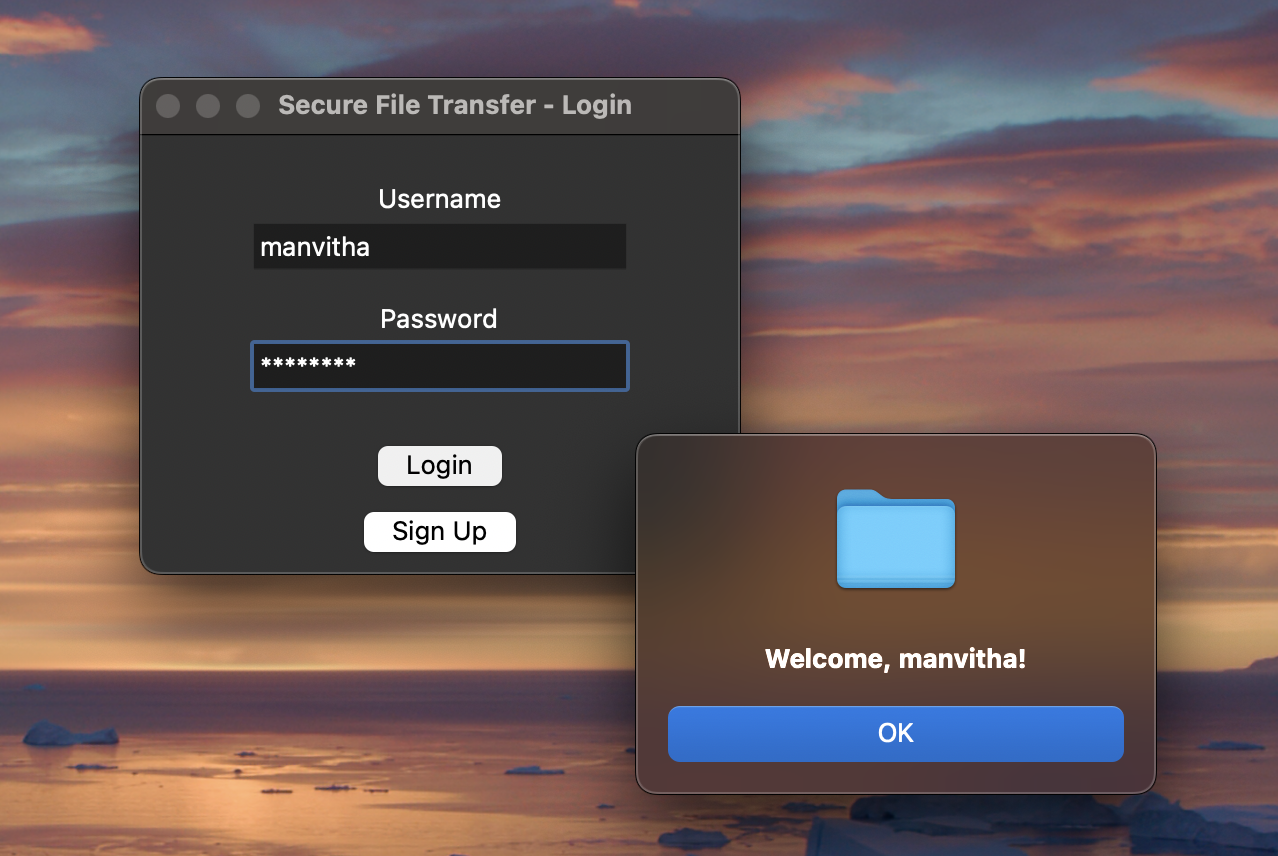
**Objectives:**

* Allow secure transfer of a file from one system to another
* Encrypt the file using AES and the key using RSA
* Enable one-time file view with manual deletion after access
* Track file and user activity via MongoDB
* Provide a simple GUI for usability

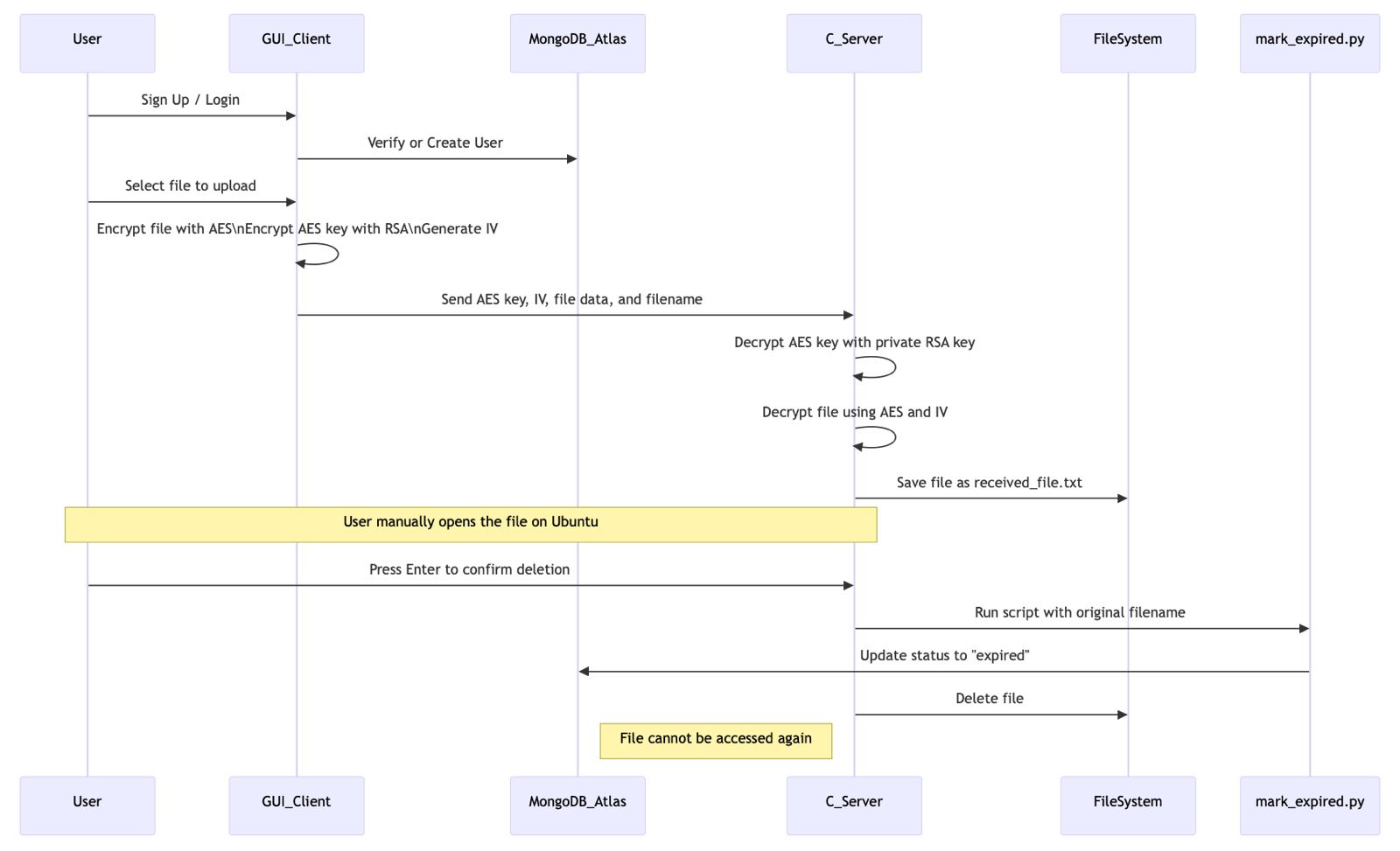
1. **System Overview**

**System Architecture:**

* **Client (macOS)**: Python GUI for login, signup, file selection, and upload.

*Fig1: User successfully login after signup*

* **Server (Ubuntu)**: C program that receives, decrypts, saves, and deletes files.
* **MongoDB Atlas**: Cloud database to track users and file status.

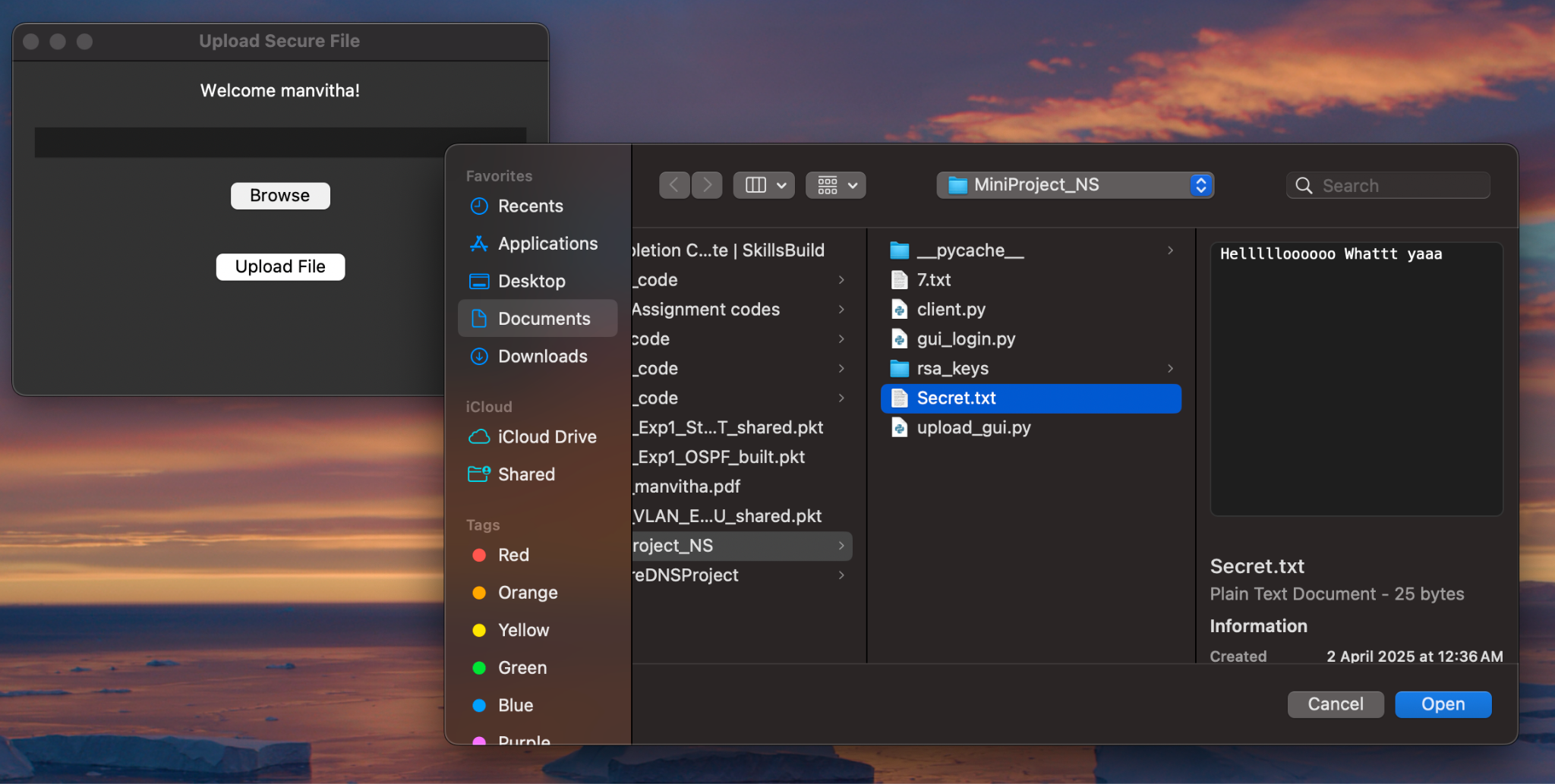


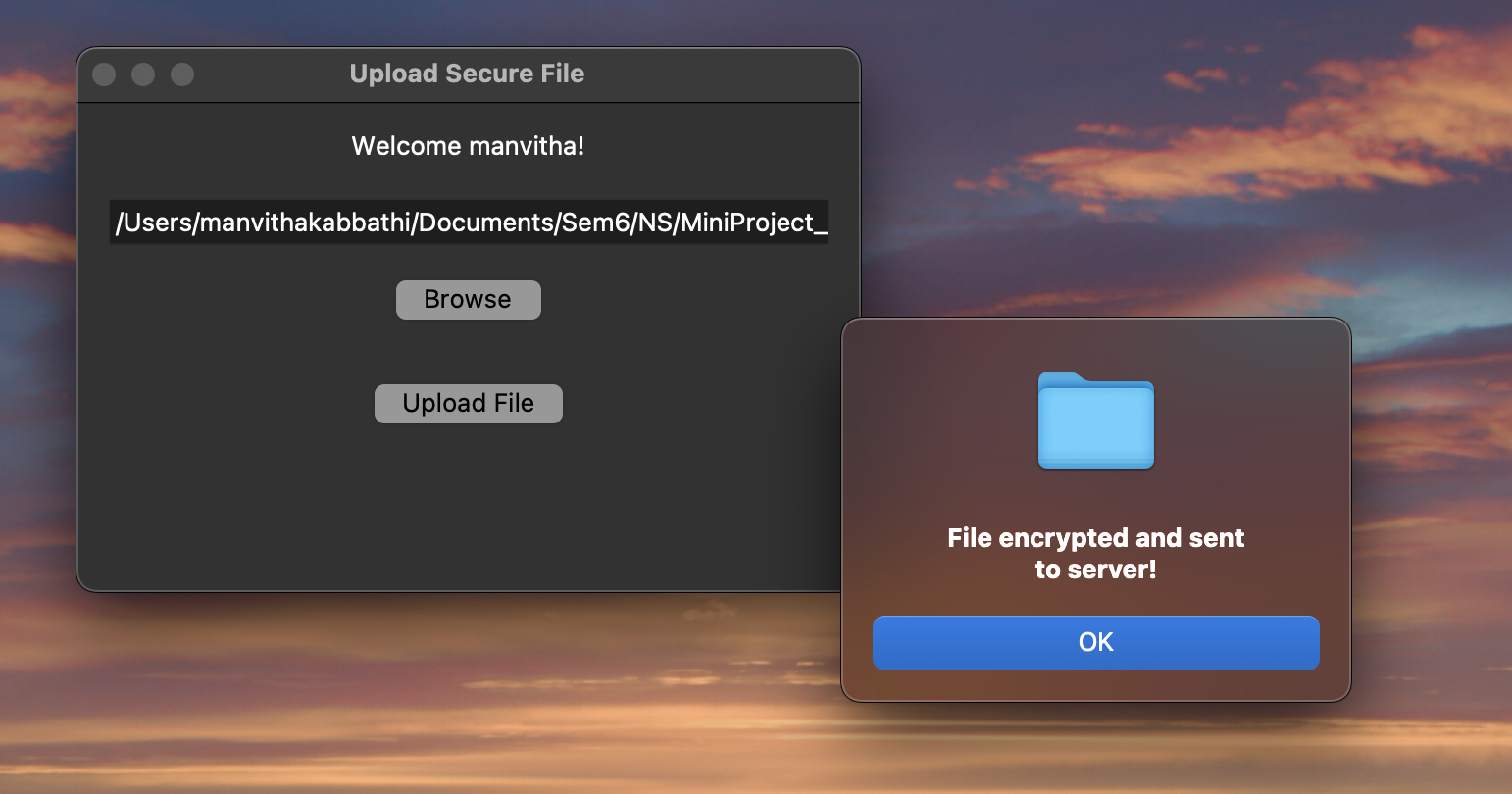
*Fig2: System Architecture*

1. **Design:**

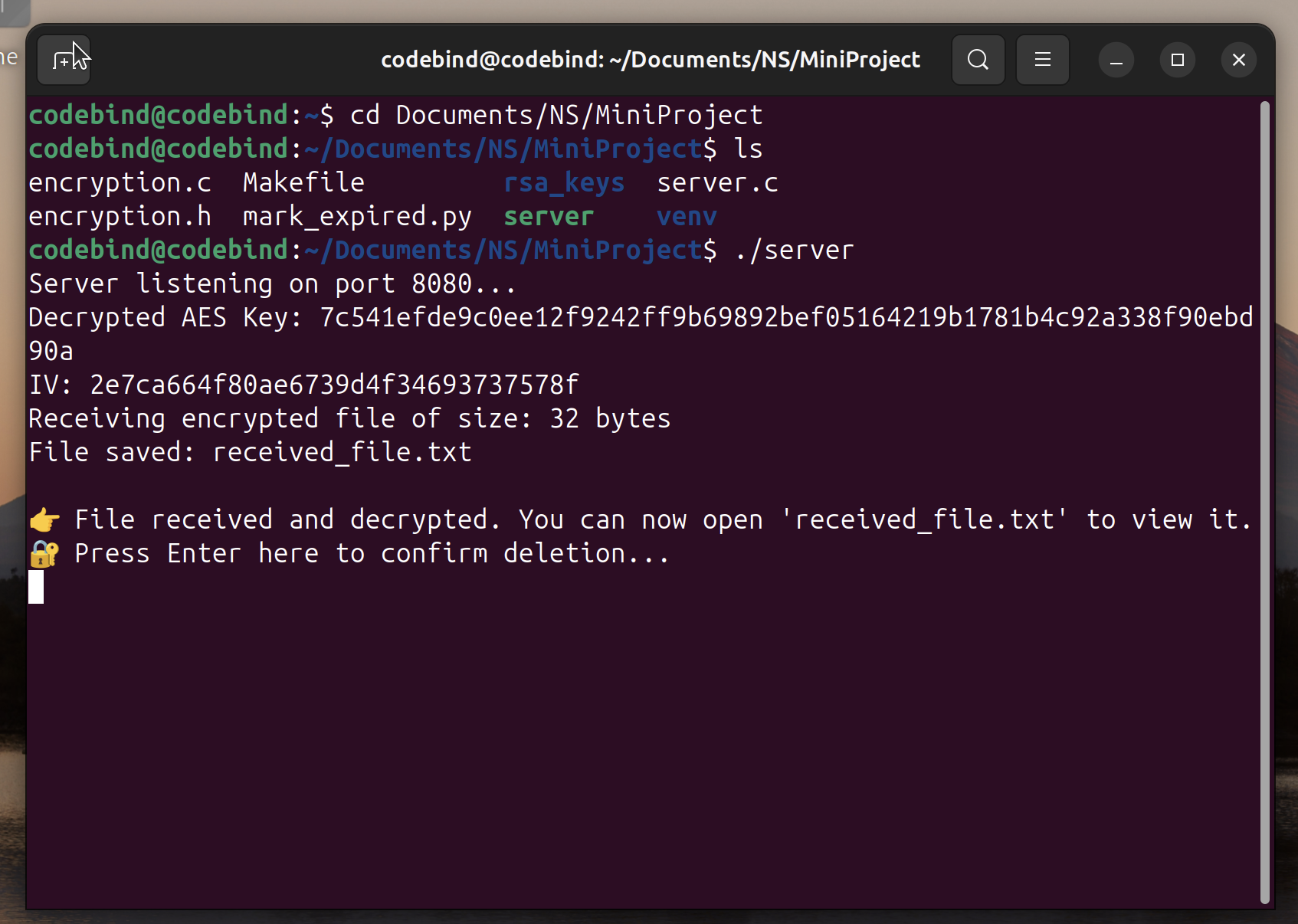
#### Major Components-

#### GUI Client (Python Tkinter): Authenticates users, lets them select files for upload

*fig3: Selecting files using Tinker from the client side*

*fig4: successfully uploading the encrypted file to the sever*

* **Encryption Module (Python)**: Uses AES (CBC) for file encryption, RSA for key encryption
* **C Server**:
  + Accepts TCP connections
  + Decrypts received content
  + Saves decrypted file
  + Waits for user confirmation

*fig5: Server side receives the received.txt file and decrypts its successfully*

* + Deletes file from the server
* **MongoDB Atlas**: Tracks user signups and logins

#### Component Interaction-

* Python client sends encrypted data over socket
* C server listens and spawns a thread per connection

1. **Security Features:**

* **AES Encryption (CBC Mode)**: Encrypts file content securely
* **RSA Encryption**: Ensures only the server can decrypt the AES key
* **One-Time Access Control**: File is auto-deleted after view
* **MongoDB**: User signups and logins management
* **Threaded Server**: Handles multiple users independently

1. **System Requirements**

#### Platform

* + **Server**: Ubuntu 22.04 LTS (GCC, OpenSSL, Python 3.12)
  + **Client:** macOS (Python 3.11, GUI)
  + **Database**: MongoDB Atlas (Cloud)

#### System Dependencies

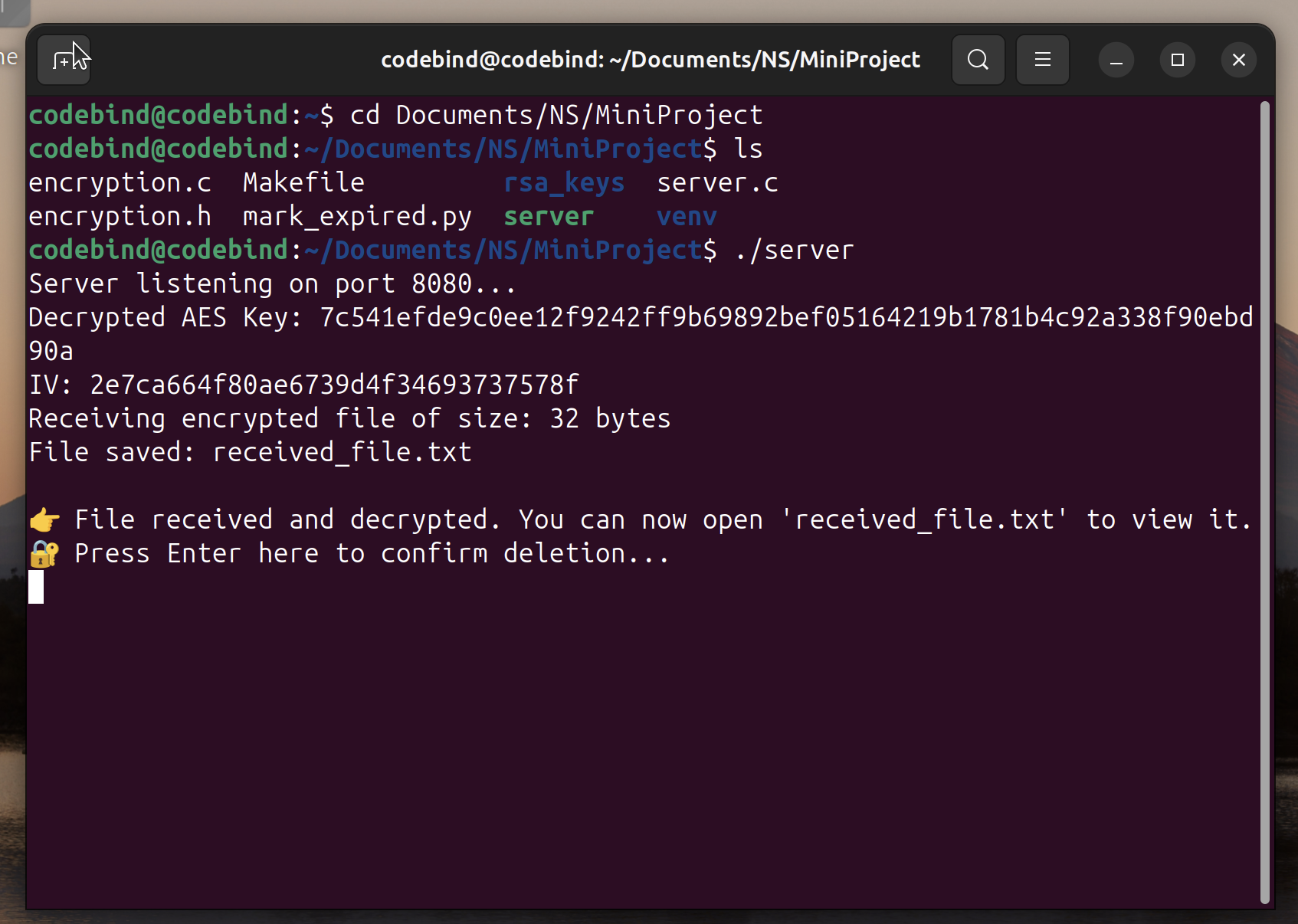
* Internet access for MongoDB Atlas
* Python environment (with venv support) on both client and server

1. **Open-source libraries and tools**

| **Tool** | **Version** | **Use** |
| --- | --- | --- |
| Python | 3.11 / 3.12 | GUI, encryption, DB update |
| PyMongo | 4.7.0 | MongoDB Atlas communication |
| OpenSSL | 1.1.1f | AES/RSA in C |
| Tkinter | Standard | GUI creation |
| GCC | 11.4.0 | Compiling C server |
| MongoDB Atlas | Cloud | Secure file metadata & auth |

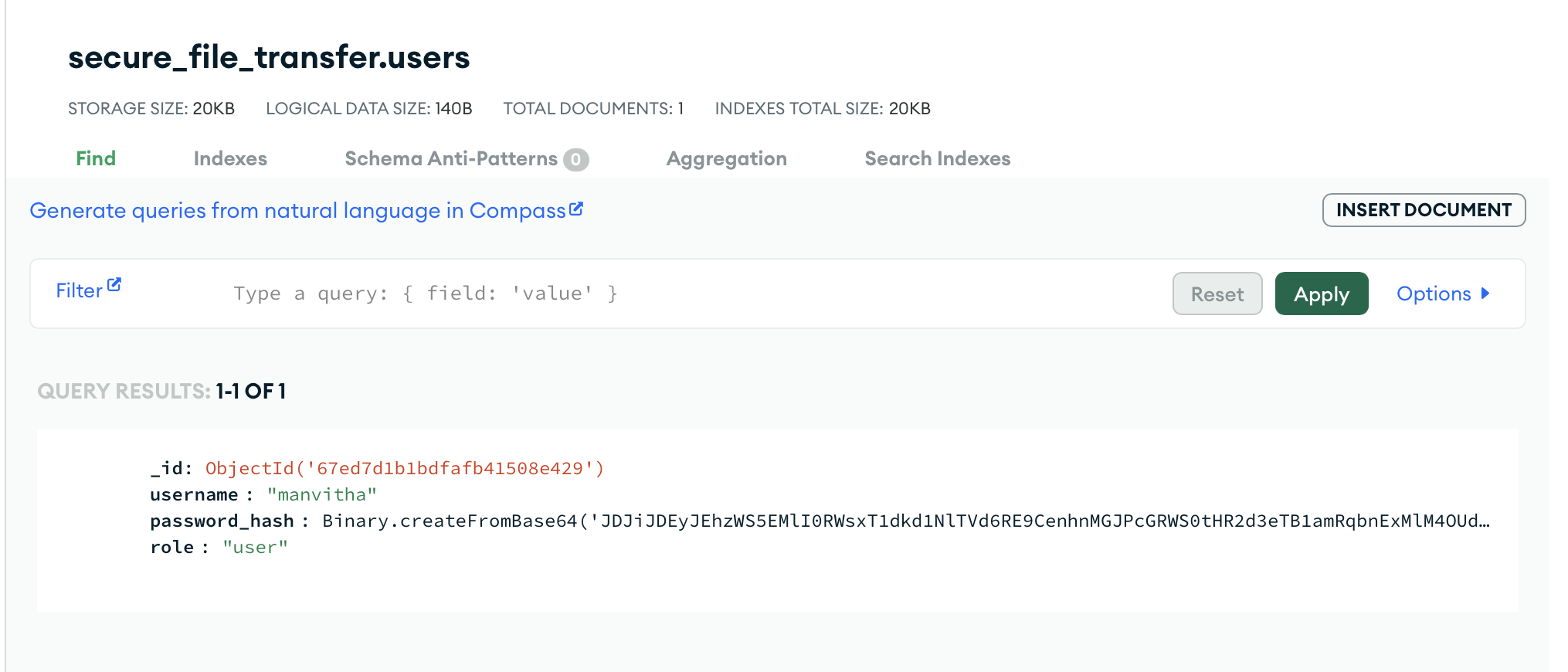
1. **Implementation and Testing**

* Tested file uploads from macOS → Ubuntu
* Verified AES and RSA encryption with test files



*fig6: Encrypted files are decrypted using AES and RSA keys*

* Checked logins/signups and MongoDB updates



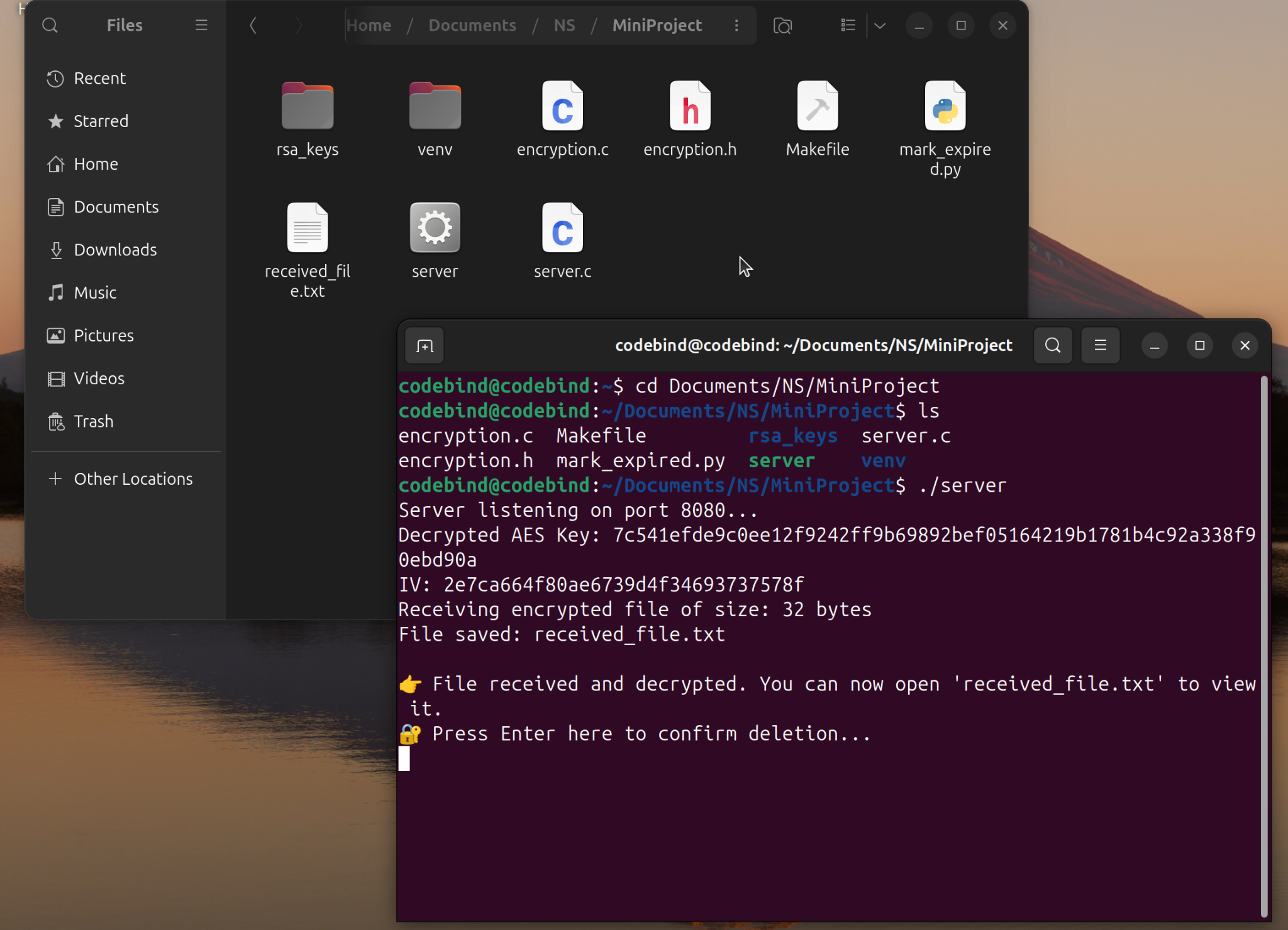
*fig7:User logs*

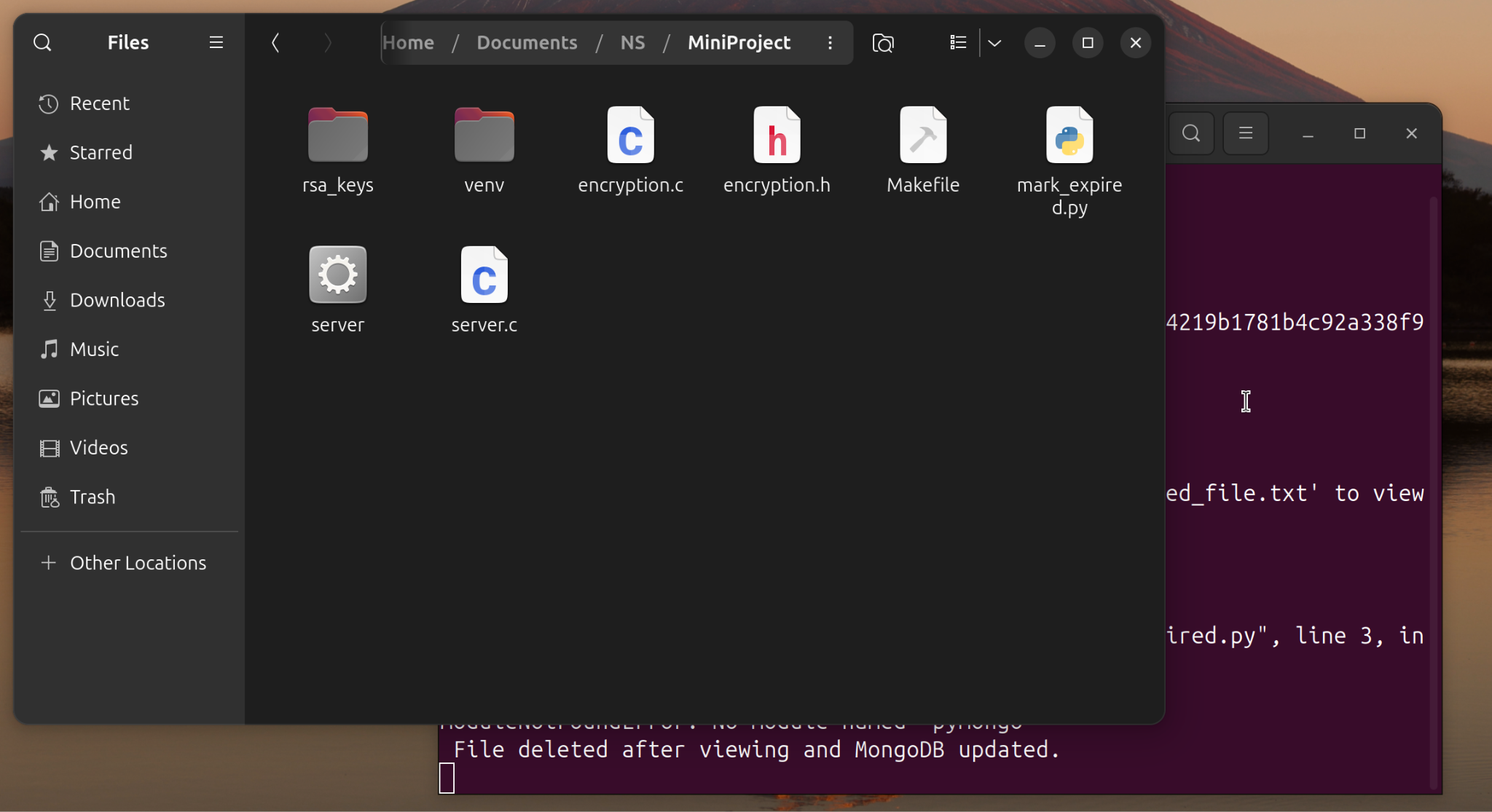
* Validated GUI login/signup flow
* Tested edge cases like wrong password
* Debugged and confirmed venv + pymongo on server

1. **Results**

**Final features supported:**

* GUI-based file upload with login
* AES + RSA hybrid encryption
* Secure socket communication
* Multithreaded C server handling uploads
* File saved, decrypted, then deleted after user review

*fig8: Server has the received.txt file and is waiting for the user confirmation to delete it.*

*fig9: received.txt file is deleted from the server after receiving user confirmation*

* Works across different OS (macOS → Ubuntu)

#### Future Extensions

* Add download GUI with automatic pull from server
* Integrate email OTP verification
* Full admin dashboard for file/user tracking

1. **Conclusion**

This project successfully demonstrates a secure file transfer system with one-time download enforcement. We combined Python, C, encryption standards, and MongoDB into a robust and practical solution. We learned how to integrate hybrid encryption, manage sockets across platforms, use GUI interfaces for real-world usability, and leverage cloud databases for metadata tracking. This system can be extended into a real-world SaaS tool or private file-sharing platform with minimal changes.

\*\*\*\*\*\*\*