

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN



---

Báo cáo

**ĐỒ ÁN 1**

---

Môn học: An ninh máy tính

CSC15003\_22MMT

Sinh viên:

Lê Hoàng Đạt  
Nguyễn Hồ Đăng Duy  
Phạm Quang Duy

Giảng viên hướng dẫn:

Lê Giang Thanh  
Lê Hà Minh  
Phan Quốc Kỳ

## Mục lục

<b>1 Thông tin sinh viên</b>	<b>2</b>
<b>2 Giới thiệu</b>	<b>2</b>
<b>3 Phân công chi tiết</b>	<b>3</b>
3.1 Giai đoạn 1: Tìm hiểu và lên kế hoạch . . . . .	3
3.2 Giai đoạn 2: Triển khai chức năng . . . . .	3
3.3 Giai đoạn 3: Hoàn thành các chức năng và gộp mã . . . . .	4
3.4 Giai đoạn 4: Kiểm thử, viết báo cáo và quay video demo . . . . .	5
<b>4 Công nghệ sử dụng</b>	<b>6</b>
4.1 Ngôn ngữ lập trình . . . . .	6
4.2 Framework . . . . .	6
4.3 Thư viện và gói mở rộng . . . . .	6
4.4 Hệ quản trị cơ sở dữ liệu . . . . .	6
4.5 Công cụ và công nghệ hỗ trợ . . . . .	7
<b>5 Kiến trúc hệ thống</b>	<b>8</b>
5.1 Sơ đồ tổng thể . . . . .	8
5.2 Thư mục và các modules chính . . . . .	8
<b>6 Chi tiết chức năng và kỹ thuật bảo mật</b>	<b>9</b>
6.1 Đăng ký tài khoản người dùng . . . . .	9
6.2 Đăng nhập và Xác thực đa yếu tố (MFA) . . . . .	12
6.3 Quản lý khoá RSA cá nhân . . . . .	16
6.4 QR Code Public Key . . . . .	17
6.5 Cập nhật thông tin tài khoản . . . . .	18
6.6 Mã hoá tập tin gửi người khác . . . . .	23
6.7 Giải mã tập tin . . . . .	24
6.8 Ký số tập tin . . . . .	25
6.9 Xác minh chữ ký . . . . .	29
6.10 Phân quyền tài khoản . . . . .	33
6.11 Ghi log bảo mật . . . . .	34
6.12 Chia nhỏ tập tin lớn . . . . .	36
6.13 Kiểm tra trạng thái khoá . . . . .	37
6.14 Tìm kiếm public key . . . . .	38
6.15 Giới hạn đăng nhập . . . . .	41
6.16 Tùy chọn định dạng lưu file . . . . .	42
6.17 Khôi phục tài khoản . . . . .	43
<b>7 Kiểm thử ứng dụng</b>	<b>46</b>
<b>Tài liệu tham khảo</b>	<b>52</b>

## 1 Thông tin sinh viên

Nhóm gồm có 3 thành viên:

- 22127060 - Lê Hoàng Đạt - 22127060@student.hcmus.edu.vn
- 22127085 - Nguyễn Hồ Đăng Duy - 22127085@student.hcmus.edu.vn
- 22127088 - Phạm Quang Duy - 22127088@student.hcmus.edu.vn

## 2 Giới thiệu

**Secure Vault** là một ứng dụng mô phỏng hệ thống bảo mật theo mô hình thực tế, được xây dựng nhằm mục tiêu bảo vệ dữ liệu cá nhân và hỗ trợ truyền tin an toàn giữa người dùng. Ứng dụng tích hợp nhiều kỹ thuật bảo mật hiện đại như mã hóa RSA/AES, xác thực đa yếu tố (MFA), ký số – xác minh chữ ký, QR Code, cùng với cơ chế phân quyền, kiểm tra trạng thái khóa, ghi log bảo mật,... Người dùng có thể:

- Đăng ký, đăng nhập bảo mật với OTP/TOTP
- Tạo và quản lý cặp khóa RSA cá nhân
- Mã hóa và giải mã tập tin với AES + RSA
- Ký số tập tin và xác minh tính toàn vẹn
- Tạo và quét QR chứa public key để chia sẻ
- Theo dõi trạng thái khóa, phân quyền người dùng (admin/user)
- Khôi phục tài khoản khi quên mật khẩu

Hệ thống được xây dựng với ngôn ngữ `Python`, sử dụng `Flask` cho backend, kết hợp `MySQL/JSON` để lưu trữ dữ liệu, cùng các thư viện bảo mật như `pycryptodome`, `pyotp`, `qrcode`,...  
GitHub Repository: [https://github.com/YuD1405/Secure\\_Vault](https://github.com/YuD1405/Secure_Vault)

### 3 Phân công chi tiết

#### 3.1 Giai đoạn 1: Tìm hiểu và lên kế hoạch

Giai đoạn 1: Tìm hiểu và lên kế hoạch		
Thành viên	Nhiệm vụ	Thời hạn
Phạm Quang Duy	Tóm tắt thông tin đồ án, chia thành các modules	16/06
Phạm Quang Duy	Tạo repository github và chốt công nghệ	16/06
Phạm Quang Duy	Chia task	16/06
Cả nhóm	Chốt công nghệ	16/06

#### 3.2 Giai đoạn 2: Triển khai chức năng

QUẢN LÝ NGƯỜI DÙNG			
Thành viên	Yêu cầu	Nhiệm vụ	Thời hạn
Nguyễn Hồ Đăng Duy	1	Đăng ký người dùng	17/06 - 19/06
Nguyễn Hồ Đăng Duy	2	Đăng nhập và xác thực MFA	18/06 - 20/06
Phạm Quang Duy	5	Cập nhật thông tin tài khoản	20/06 - 21/06
Nguyễn Hồ Đăng Duy	10	Phân quyền Admin	20/06 - 23/06
Nguyễn Hồ Đăng Duy	15	Giới hạn login	19/06 - 20/06
Phạm Quang Duy	17	Khôi phục tài khoản	17/06 - 23/06
Nguyễn Hồ Đăng Duy	UI	Frontend: Form accounts, navigation	29/01 - 09/03

MÃ HÓA VÀ GIẢI MÃ			
Thành viên	Yêu cầu	Nhiệm vụ	Thời hạn
Lê Hoàng Đạt	3	Quản lí khóa	17/06 - 19/06
Lê Hoàng Đạt	4	Tạo và quét QR cho pub key	24/06 - 25/06
Lê Hoàng Đạt	6	Mã hóa tập tin gửi	20/06 - 21/06
Lê Hoàng Đạt	7	Giải mã tập tin	21/06 - 22/06
Lê Hoàng Đạt	12	Chia nhỏ tập tin khi mã hóa	23/06 - 24/06
Lê Hoàng Đạt	16	Tùy chọn định dạng lưu	21/06 - 22/06
Phạm Quang Duy	UI	Giao diện chức năng dashboard	17/06 - 25/06

SIGNING & LOGGING & UTILS			
Thành viên	Yêu cầu	Nhiệm vụ	Thời hạn
Phạm Quang Duy	8	Ký số	17/06 - 18/06
Phạm Quang Duy	9	Xác minh ký số	19/06 - 20/06
Phạm Quang Duy	11	Log bảo mật	21/06 - 22/06
Lê Hoàng Đạt	13	Kiểm tra trạng thái khóa	18/06 - 19/06
Phạm Quang Duy	14	Tìm pub Key	23/06 - 23/06
Nguyễn Hồ Đăng Duy	UI	Logging interface	23/06 - 27/06

### 3.3 Giai đoạn 3: Hoàn thành các chức năng và gộp mã

Giai đoạn 3: Hoàn thành các chức năng và merge code		
Thành viên	Nhiệm vụ	Thời hạn
Nguyễn Hồ Đăng Duy	Kết nối: Group 1: User management → main UI	01/07 - 03/07
Lê Hoàng Đạt	Kết nối: Group 2: Encrypt và Decrypt → main UI	03/07 - 06/07
Phạm Quang Duy	Kết nối: Group 3: Logging và Utils → main UI	06/07 - 08/07
Phạm Quang Duy	Kết nối toàn bộ chức năng (flow giữa các group chức năng)	08/07

### 3.4 Giai đoạn 4: Kiểm thử, viết báo cáo và quay video demo

Giai đoạn 4: Test chức năng, quay video và hoàn thiện report		
Thành viên	Nhiệm vụ	Thời hạn
Nguyễn Hồ Đăng Duy	Test toàn bộ flow chương trình	08/07 - 09/07
Cả nhóm	Chỉnh sửa sau test	09/07
Cả nhóm	Report	08/07 - 10/07
Cả nhóm	Record video	11/07 - 12/07

## 4 Công nghệ sử dụng

Dưới đây là danh sách các công nghệ và thư viện được sử dụng trong quá trình phát triển ứng dụng **Secure Vault**, một hệ thống mã phỏng bảo mật gồm xác thực đa yếu tố, mã hóa tập tin, ký số và phân quyền người dùng.

### 4.1 Ngôn ngữ lập trình

- **Python**: Ngôn ngữ chính được sử dụng cho toàn bộ backend của hệ thống.

### 4.2 Framework

- **Flask**: Framework web nhẹ, hỗ trợ xây dựng REST API, routing và tích hợp giao diện HTML.

### 4.3 Thư viện và gói mở rộng

- **flask**: Framework core cho backend Python.
- **flask-cors**: Cho phép giao tiếp giữa frontend và backend khác nguồn.
- **flask-mysqldb**: Kết nối MySQL với Flask.
- **pyotp**: Tạo và xác minh mã TOTP (Google Authenticator).
- **qrcode**: Tạo mã QR chứa public key hoặc URI xác thực.
- **pillow**: Hỗ trợ xử lý ảnh (dùng kết hợp với QR code).
- **cryptography**: Cung cấp các primitive bảo mật như AES, RSA, SHA-256.
- **python-dotenv**: Quản lý biến môi trường từ file .env.
- **logging (built-in)**: Ghi log các hành vi bảo mật, bao gồm đăng nhập, xác thực, mã hóa và các thao tác hệ thống khác
- **smtplib (built-in)**: Giao tiếp với SMTP server để gửi mã OTP qua email.
- **email.mime**: Dùng các class như MIMEText, MIMEMultipart để định dạng nội dung email HTML hoặc plain text.

### 4.4 Hệ quản trị cơ sở dữ liệu

- **MySQL**: Lưu trữ thông tin người dùng, khóa, mã OTP và log hành vi bảo mật.

#### 4.5 Công cụ và công nghệ hỗ trợ

- **Git:** Quản lý phiên bản mã nguồn.
- **.env file:** Lưu cấu hình bảo mật như SMTP, thông tin DB, app secret.
- **SMTP (Gmail App Password):** Gửi mã OTP qua email.
- **Jinja2:** Template engine giúp render giao diện HTML từ backend Flask.
- **HTML/CSS:** Xây dựng giao diện người dùng cơ bản.

## 5 Kiến trúc hệ thống

### 5.1 Sơ đồ tổng thể

### 5.2 Thư mục và các modules chính

Đồ án được tổ chức rõ ràng theo mô hình **Flask MVC** và phân chia modules theo chức năng bảo mật:

- `main.py` - Điểm khởi chạy của ứng dụng Flask
- `.env` - Tập tin cấu hình (thông tin DB, email, secret key,...)
- `data/` - Chứa dữ liệu hệ thống
  - `key_manage/` -
  - `keys/` -
  - `qr/`
- `flaskapi/` - Backend API
  - `app.py` - Cấu hình Flask app và đăng ký route
  - `routes/` - Chứa các tệp định nghĩa route theo nhóm chức năng
- `frontend/` - Các file HTML render giao diện với Jinja2
- `log/` - Ghi toàn bộ hành vi quan trọng: đăng nhập, mã hóa, ký số,... kèm thời gian, email, trạng thái
- `modules/` - Các modules chức năng chính
  - `auth/` - Hash passphrase, sinh OTP, xác thực MFA
  - `crypto/` - Mã hóa RSA/AES, ký số và xác minh chữ ký
  - `utils/` - Hàm tiện ích chung: log sự kiện, mail, QR,...
- `mySQL/` - Các chương trình để tạo database, table cho cơ sở dữ liệu MySQL
- `report/` - Báo cáo và video demo
- `README.md` - Hướng dẫn cài đặt và chạy thử đồ án. Mô tả các chức năng và công nghệ sử dụng.

## 6 Chi tiết chức năng và kỹ thuật bảo mật

### 6.1 Đăng ký tài khoản người dùng

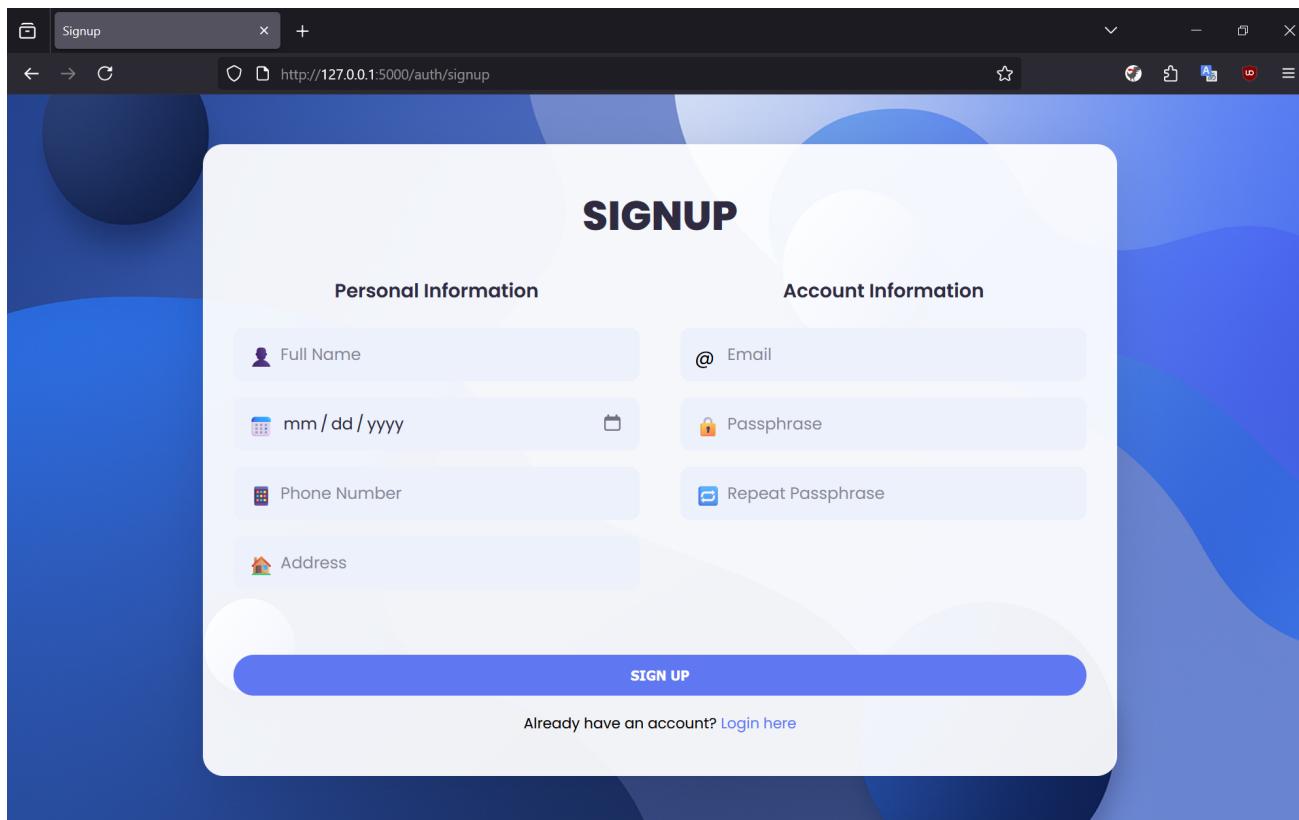
#### Mục tiêu

Cho phép người dùng tạo tài khoản bằng cách điền đầy đủ thông tin cá nhân và **passphrase** (đảm bảo đủ an toàn). Thông tin sau đó được kiểm tra tính hợp lệ, hash **passphrase** và lưu vào cơ sở dữ liệu. Đồng thời tạo **recovery key** và cung cấp cho người dùng.

#### Giao diện

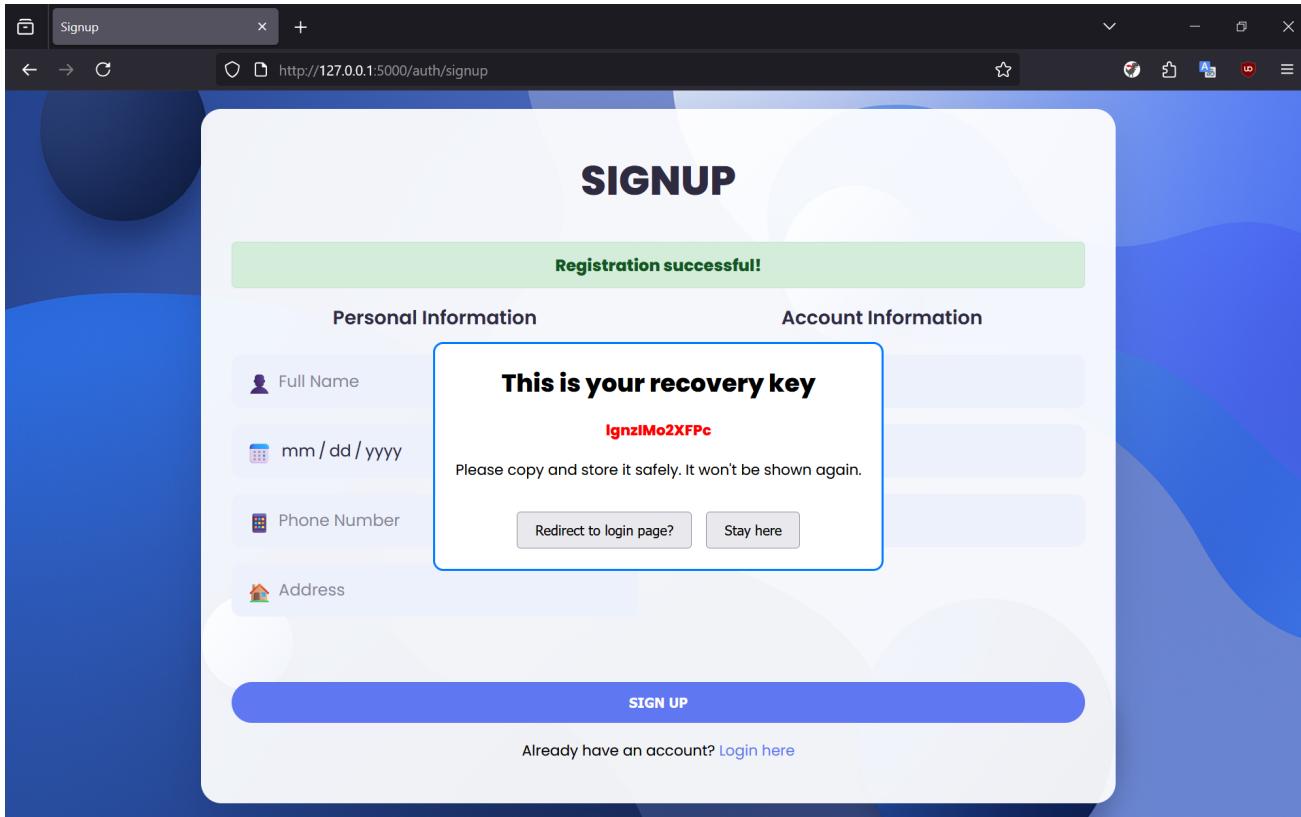
Giao diện `/auth/signup` là form HTML bao gồm các trường:

- Email
- Họ tên
- Ngày sinh
- Số điện thoại
- Địa chỉ
- Passphrase và xác nhận passphrase



Hình 1: Giao diện Signup

Khi đăng ký thành công, hệ thống sẽ hiển thị mã khôi phục tài khoản, yêu cầu người dùng lưu lại mã này để phục hồi nếu mất mật khẩu.



Hình 2: Giao diện popup recovery key

### Quy trình thực hiện

1. Người dùng nhập thông tin và submit form → POST `/signup`
2. Flask gọi hàm `register_user(request.form)` trong `logic.py`
3. Thông tin được kiểm tra và chuẩn hóa:
  - Email hợp lệ (`is_valid_email`)
  - Ngày sinh đúng định dạng (`is_valid_date`)
  - Số điện thoại gồm đúng 10 chữ số (`is_valid_phone`)
  - Passphrase đủ mạnh (`is_strong_passphrase`)
4. Nếu hợp lệ:
  - Sinh `salt` ngẫu nhiên
  - Băm `passphrase` kết hợp với salt bằng `SHA-256`
  - Sinh `recovery_code` để dùng cho khôi phục

- Sinh `mfa_secret` phục vụ TOTP trong MFA
5. Thực hiện `INSERT` bản ghi vào bảng `users` trong CSDL
  6. Ghi log hành động: `log_user_action(email, "Register", "Success")`
  7. Trả về kết quả thành công và hiển thị mã khôi phục

## Chi tiết kỹ thuật và thư viện bảo mật

### 1. Hashing passphrase với Salt

Thư viện: `hashlib, os`

Kỹ thuật:

- Salt được sinh bằng `os.urandom(16).hex()` → đảm bảo tính ngẫu nhiên mạnh.
- `passphrase` được hash bằng `SHA-256(passphrase + salt)` trước khi lưu xuống DB.

### 2. Validator kiểm tra đầu vào

Thư viện: `re, datetime`

Kỹ thuật:

- Regex kiểm tra định dạng email, số điện thoại.
- Kiểm tra `passphrase` phải ít nhất 8 ký tự, chứa chữ hoa, số, ký hiệu.
- Loại bỏ ký tự nguy hiểm khỏi input (`sanitize_input()`) → giảm rủi ro XSS / SQLi / Code Injection.

### 3. Tạo mã khôi phục

Thư viện: `random, string`

Kỹ thuật:

- Sinh chuỗi ngẫu nhiên gồm chữ + số, dùng 1 lần.
- Người dùng được yêu cầu lưu lại → dùng khi mất `passphrase`.
- Được lưu trong DB (`users.recovery_code`) và xóa sau khi reset thành công.

## 6.2 Đăng nhập và Xác thực đa yếu tố (MFA)

### Mục tiêu

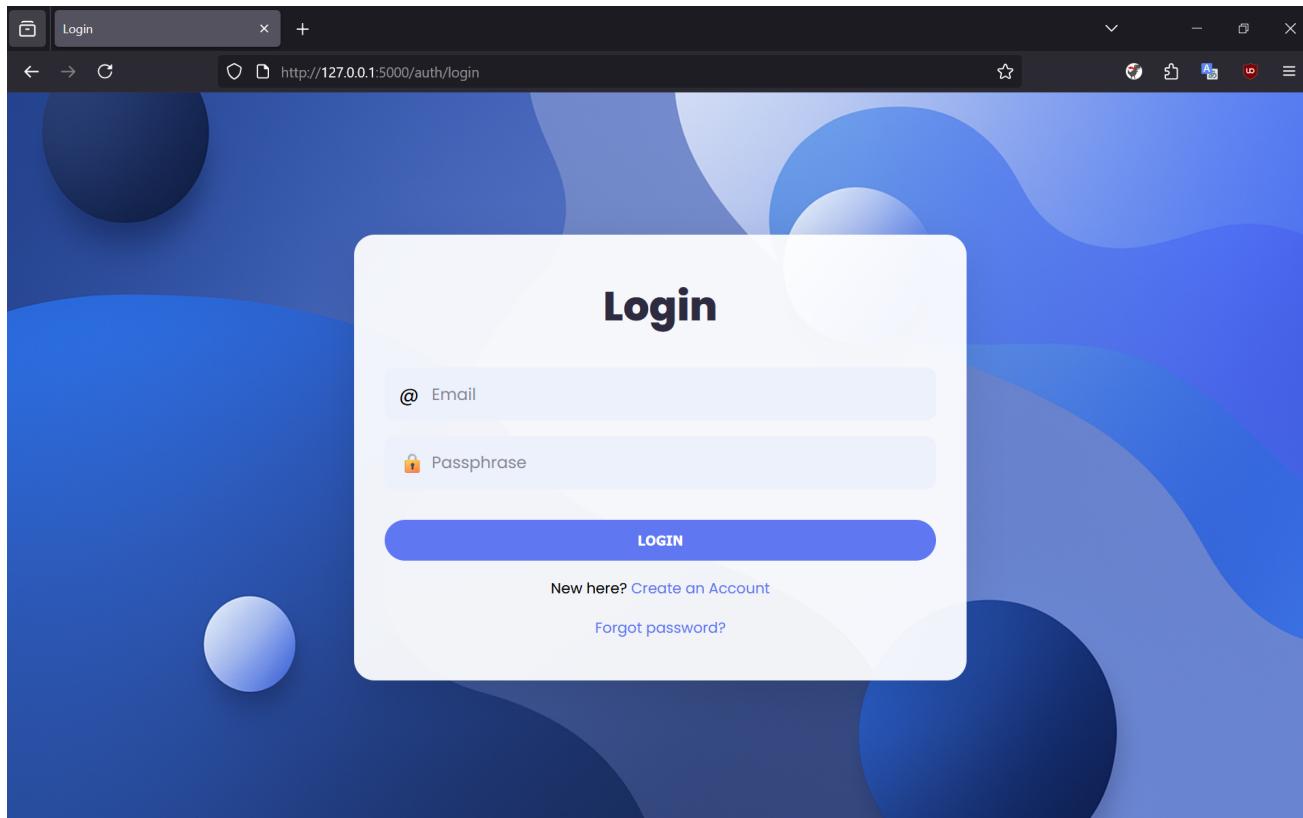
Cho phép người dùng đăng nhập bằng cách nhập `email` và `passphrase`. Nếu thông tin hợp lệ, hệ thống sẽ yêu cầu xác minh OTP qua email hoặc mã TOTP (Google Authenticator). Hệ thống hỗ trợ các biện pháp bảo mật nâng cao:

- Bảo vệ bằng xác thực 2 yếu tố
- Tự động khóa tài khoản 5 phút sau 5 lần đăng nhập sai
- Cho phép admin khóa tài khoản thủ công
- Ghi log toàn bộ hành vi đăng nhập

### Giao diện

Giao diện `/auth/login` là form HTML bao gồm 2 trường:

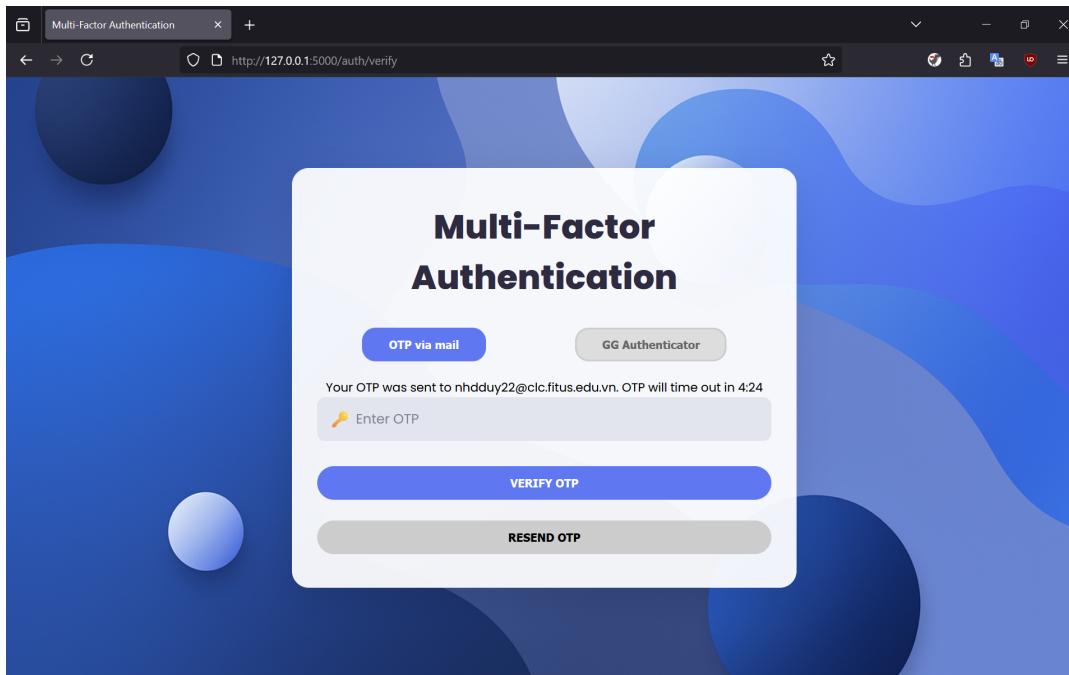
- Email
- Passphrase



Hình 3: Giao diện Login

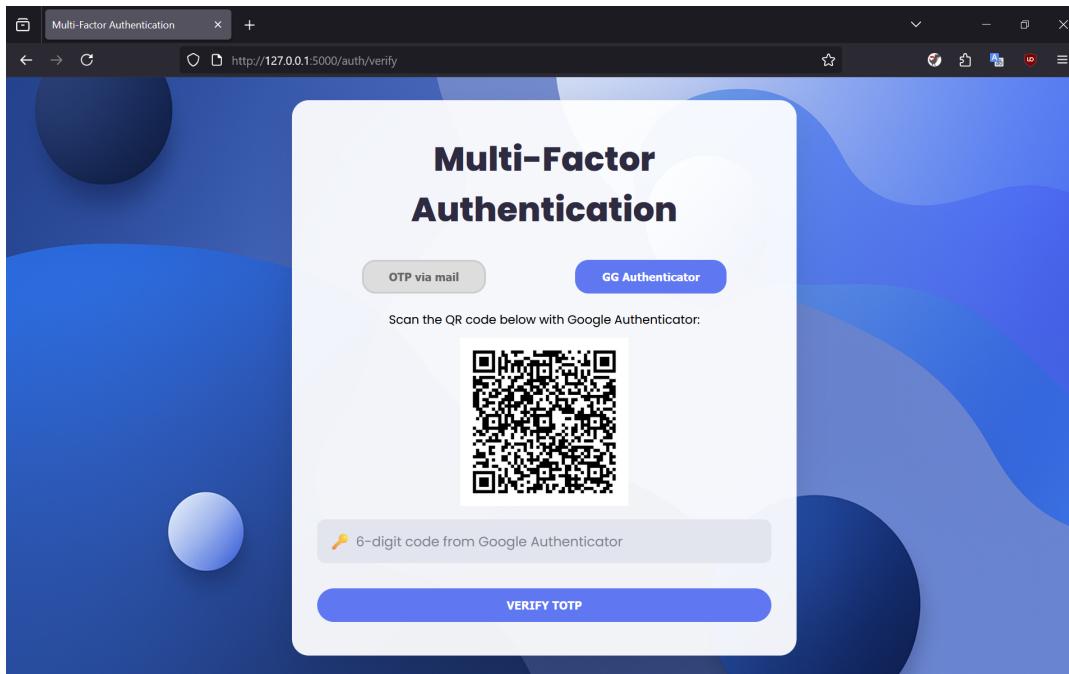
Sau khi đăng nhập thành công, người dùng được chuyển đến `/auth/verify` để xác thực MFA bằng 1 trong 2 phương thức:

## 1. OTP gửi qua email (mặc định)



Hình 4: Xác thực bằng OTP gửi qua mail

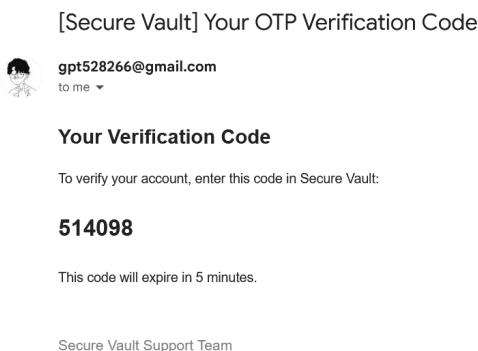
## 2. Mã TOTP (QR code cho Google Authenticator)



Hình 5: Xác thực bằng TOTP qua QR code

## Quy trình thực hiện

1. Người dùng submit form `/auth/login` với `email` và `passphrase`.
2. Flask gọi hàm `process_login(email, passphrase)` trong `logic.py`
3. Chương trình thực hiện:
  - Kiểm tra người dùng tồn tại hay không.
  - So sánh `hash(passphrase + salt)`.
  - Nếu sai, tăng `failed_attempts` và cập nhật `last_failed_login`.
  - Nếu sai hơn 5 lần trong 2 phút → Gán `is_locked = 1`, trả về trạng thái **locked**.
  - Nếu tài khoản bị khóa bởi admin (`last_failed_login = null` và `is_locked = 1`) → trả về **locked by admin**.
  - Nếu các thông tin đăng nhập đúng → Reset `failed_attempts`, ghi log, chuyển đến bước xác thực MFA.
4. Người dùng được chuyển đến `/auth/verify`, chọn xác thực OTP qua email hoặc quét QR TOTP (mặc định là gửi OTP qua email đã đăng ký).
5. Nếu chọn OTP:
  - Gọi `generate_and_send_otp(email)` → sinh mã 6 chữ số, gửi qua email, lưu DB kèm thời gian hết hạn.



Hình 6: Cấu trúc mail

- Kiểm tra bằng `verify_otp_code(email, input_code)`.
6. Nếu chọn TOTP:
  - Sinh mã QR từ `generate_qr_code(email)` → dùng cho ứng dụng Google Authenticator.
  - Kiểm tra bằng `verify_totp_code(email, input_code)`.
7. Nếu đúng → Lưu `session['user_id']` và chuyển đến `auth/dashboard`

## Chi tiết kỹ thuật và thư viện bảo mật

### 1. Hashing passphrase với Salt

Thư viện: `hashlib`

Kỹ thuật:

- Salt đã được sinh bằng `os.urandom(16).hex()` →.
- Kiểm tra `passphrase` được người dùng nhập vào sau khi hash có giống với chuỗi được lưu trong DB hay không.

### 2. Gửi mã OTP qua email

Thư viện: `random, datetime, smtplib, email`

Kỹ thuật:

- Sinh mã 6 chữ số ngẫu nhiên.
- Lưu `expires_at = now + 5 minutes`
- Dùng `email.mime.text` và `email.mime.multipart` để format tin nhắn gửi email.
- Tạo tài khoản gmail đã xác thực 2 yếu tố và tiến hành tạo `app password` và lưu vào `.env` với cấu trúc như sau

<sup>1</sup> `SMTP_USER=<sender mail>`

<sup>2</sup> `SMTP_PASS=<app password>`

- OTP được tạo sẽ lưu vào DB và gửi bằng SMTP → người dùng cần kiểm tra email để nhận OTP.
- Người dùng nhập mã OTP → Server sẽ lấy bản `otp_code` mới nhất và kiểm tra xem mã có đúng và còn hạn hay không → Nếu hợp lệ, người dùng xác thực thành công

### 3. TOTP

Thư viện: `pyotp, qrcode, base64`

Kỹ thuật:

- Mỗi tài khoản có một `mfa_secret` (chuỗi `base32`) sinh ngẫu nhiên và được lưu trong bảng `users`.
- Dùng `pyotp.totp.TOTP(mfa_secret).provisioning_uri()` để tạo ra URI định dạng chuẩn TOTP.
- Sinh ảnh QR code từ URI trên.
- Người dùng sử dụng Google Authenticator để quét mã trên.
- Người dùng nhập 6 số do app tạo ra mỗi 30 giây.
- Server sẽ dùng lại `mfa_secret` từ DB để sinh lại mã đúng tại thời điểm đó. Nếu mã số giống nhau thì người dùng xác thực thành công.

### 6.3 Quản lý khoá RSA cá nhân

Mục tiêu

Giao diện

Quy trình thực hiện

Chi tiết kỹ thuật và thư viện bảo mật

## 6.4 QR Code Public Key

Mục tiêu

Giao diện

Quy trình thực hiện

Chi tiết kỹ thuật và thư viện bảo mật

## 6.5 Cập nhật thông tin tài khoản

### Mục tiêu

Chức năng cập nhật tài khoản cho phép người dùng chỉnh sửa thông tin cá nhân (họ tên, ngày sinh, địa chỉ, số điện thoại), cũng như thay đổi mật khẩu (passphrase). Việc này đảm bảo:

- Người dùng có thể chủ động thay đổi thông tin cá nhân khi cần.
- Cho phép cập nhật passphrase và tự động mã hóa lại khóa RSA bằng passphrase mới.
- Đảm bảo tính bảo mật và tính nhất quán dữ liệu khóa trong hệ thống.

### Giao diện

Giao diện tại /render\_update\_account bao gồm:

- Form nhập các trường: Họ tên, ngày sinh, địa chỉ, số điện thoại.
- Tùy chọn nhập mật khẩu hiện tại và mật khẩu mới nếu muốn đổi passphrase.
- Nút Save changes sẽ gửi POST đến API /update\_account và hiển thị thông báo kết quả.

The screenshot shows a web-based account update interface. On the left, a vertical sidebar features icons for navigation (home, back, forward), security (key, lock), and user management (user icon). The main content area has a title 'Update Account Information'. It contains several input fields: 'Full Name' (abc), 'Date of Birth' (11/11/1111), 'Address' (nhà của 1), 'Phone Number' (1111111111), and an 'Email (read-only)' field (1@gmail.com). Below these fields is a 'Change Passphrase' section with 'Old Passphrase' and 'New Passphrase' input fields. At the bottom right is a green 'Save Changes' button. The overall design is clean and modern.

Hình 7: Giao diện cập nhật thông tin

### Quy trình thực hiện

**Bước 1 - Người dùng nhập thông tin** Người dùng điền thông tin cần thay đổi, có thể chọn thay đổi passphrase hoặc không.

**Bước 2 - Gửi yêu cầu cập nhật** Form gửi POST đến route /update\_account, kèm theo session để xác định người dùng.

**Bước 3 - Kiểm tra và xử lý** Server thực hiện:

- Kiểm tra dữ liệu đầu vào và định dạng.
- Kiểm tra passphrase cũ nếu có yêu cầu đổi mật khẩu.
- Hash passphrase mới và cập nhật.
- Nếu đổi mật khẩu, khóa riêng RSA sẽ được giải mã bằng passphrase cũ và mã hóa lại bằng passphrase mới.

**Bước 4 - Ghi log và phản hồi** Ghi log cập nhật tài khoản và trả kết quả thành công hoặc thất bại cho giao diện frontend.

The screenshot shows a user interface for updating account information. On the left is a vertical sidebar with icons for home, key, lock, file, settings, and user. The main area has a title 'Update Account Information'. It contains fields for 'Full Name' (Phạm Quang Duy), 'Date of Birth' (11/11/1111), 'Address' (nhà của 1), 'Phone Number' (1111111111) and 'Email (read-only)' (1@gmail.com). Below these are sections for 'Change Passphrase' with 'Old Passphrase' and 'New Passphrase' fields, and a 'Save Changes' button. A green notification bar at the top right says 'Account information has been updated!'. The bottom right corner of the page has three small circular icons.

Hình 8: Giao diện cập nhật thông tin thành công

### Chi tiết kỹ thuật và thư viện bảo mật

- Kiểm tra thông tin đầu vào**
  - Sử dụng các hàm xác thực định dạng: `is_valid_email()`, `is_valid_date()`, `is_valid_phone()`.
  - Bắt buộc điền đầy đủ thông tin: email, họ tên, ngày sinh, địa chỉ, số điện thoại.
- Kiểm tra và cập nhật passphrase**
  - Nếu có ý định đổi passphrase, yêu cầu nhập cả pass cũ và pass mới.
  - Pass mới phải khác pass cũ và đạt yêu cầu bảo mật qua hàm `is_strong_passphrase()`.
  - So sánh hash passphrase cũ với DB bằng `hash_with_salt()`.
  - Nếu hợp lệ, hash passphrase mới và cập nhật vào DB.
- Re-encrypt RSA key với passphrase mới**
  - Hàm `re_encrypt_private_key_with_new_passphrase()` dùng để giải mã khóa RSA bằng passphrase cũ và mã hóa lại bằng passphrase mới.
  - Nếu lỗi trong quá trình này, cập nhật DB sẽ được rollback và trả lỗi.
  - Passphrase mới được cập nhật vào session để sử dụng cho các thao tác sau.

```
--- [BẮT ĐẦU QUÁ TRÌNH TÁI MÃ HOÁ PRIVATE KEY CHO 1@gmail.com] ---  
Giải mã thành công private key bằng passphrase cũ.  
Mã hoá lại thành công private key bằng passphrase mới.  
Đã cập nhật thành công file key_1.json.  
127.0.0.1 - - [11/Jul/2025 10:21:59] "POST /auth/update_account HTTP/1.1" 200 -
```

Hình 9: Mã hóa passphrase mới thành công

**4. Cập nhật recovery key (khóa khôi phục)** Để đảm bảo người dùng có thể khôi phục khóa riêng sau khi quên passphrase, hệ thống sử dụng một **recovery key** (AES key phụ) được mã hóa bằng passphrase hiện tại. Khi người dùng thay đổi passphrase, recovery key cần được giải mã và mã hóa lại như sau:

- **Bước 1: Giải mã recovery key bằng passphrase cũ**
  - Truy vấn trường `encrypted_recovery_key` từ bảng `users`.
  - Derive khóa AES từ passphrase cũ bằng hàm `derive_aes_key(pass1, salt)`.
  - Giải mã recovery key hiện tại bằng AES.
- **Bước 2: Mã hóa lại recovery key bằng passphrase mới**
  - Derive AES key mới từ passphrase mới: `derive_aes_key(pass2, salt)`.
  - Mã hóa recovery key vừa giải mã bằng AES key mới.
  - Lưu bản mã hóa mới vào trường `encrypted_recovery_key`.
- **Lưu ý bảo mật:**
  - Nếu giải mã hoặc mã hóa recovery key thất bại, toàn bộ quá trình cập nhật sẽ bị rollback và trả lỗi.
  - Mọi thao tác đều được log bằng `log_user_action(..., "Fail", ...)` hoặc `log_internal_error` với mức `error`.

**5. Ghi log và bảo mật** • Mọi thao tác đều được ghi bằng `log_user_action()` với trạng thái và chi tiết lỗi.

- Lỗi truy cập không có session sẽ chuyển hướng về trang đăng nhập.
- Dữ liệu passphrase luôn được hash bằng salt, không lưu dưới dạng thô.

**6. Xử lý lỗi** • Nếu thiếu session → lỗi 401 hoặc redirect về trang đăng nhập.

- Nếu định dạng sai hoặc passphrase cũ không đúng → trả lỗi rõ ràng.
- Nếu không có thay đổi gì → trả thông báo: "No changes were made."
- Nếu cập nhật thành công nhưng RSA re-encryption thất bại → rollback và log lỗi mức độ `error`.

The screenshot shows the 'Update Account Information' page. On the left is a vertical sidebar with icons for home, key, lock, file, list, grid, user, and export. The main area has fields for Full Name (Phạm Quang Duy), Date of Birth (11/11/1111), Address (nhà cửa 1), Phone Number (1111111111), and Email (read-only) (1@gmail.com). Below these is a 'Change Passphrase' section with 'Old Passphrase' and 'New Passphrase' fields, both containing four dots. A red error box at the top right says: 'New passphrase is too weak. It must contain at least 8 characters, an uppercase letter, a number, and a special symbol.' A green 'Save Changes' button is at the bottom.

Hình 10: Thông báo lỗi khi kiểm tra đầu vào

This screenshot is identical to Figure 10, showing the 'Update Account Information' page. The sidebar and form fields are the same. However, the red error box at the top right now says: 'Current passphrase is incorrect.' The 'Save Changes' button is still present at the bottom.

Hình 11: Thông báo lỗi khi kiểm tra đầu vào

The screenshot shows a user interface for updating account information. On the left is a vertical sidebar with icons for Home, Key, Lock, Address, Phone, Email, and User. The main area has a title 'Update Account Information'. It contains fields for 'Full Name' (Phạm Quang Duy), 'Date of Birth' (11/11/1111), 'Address' (nhà của 1), 'Phone Number' (1111111111), and 'Email (read-only)' (1@gmail.com). Below these are sections for 'Change Passphrase' with 'Old Passphrase' and 'New Passphrase' fields, both currently empty. A green 'Save Changes' button is at the bottom right. In the top right corner, there is a blue box stating 'No changes were made.'

Hình 12: Thông báo khi không có thay đổi

## 6.6 Mã hoá tập tin gửi người khác

Mục tiêu

Giao diện

Quy trình thực hiện

Chi tiết kỹ thuật và thư viện bảo mật

## 6.7 Giải mã tập tin

Mục tiêu

Giao diện

Quy trình thực hiện

Chi tiết kỹ thuật và thư viện bảo mật

## 6.8 Ký số tập tin

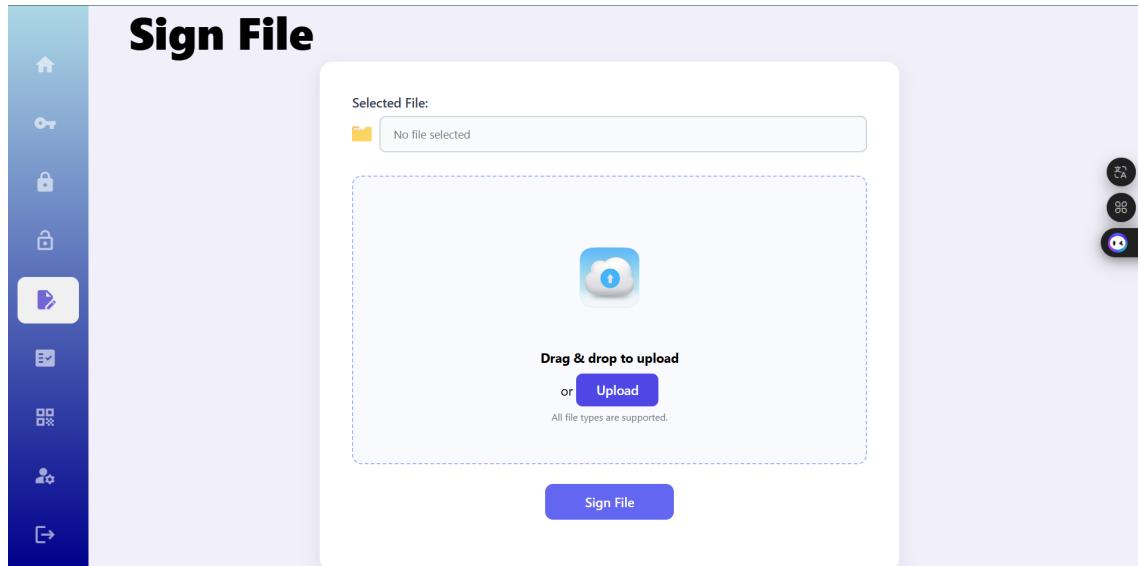
### Mục tiêu

Chức năng ký số cho phép người dùng xác minh tính **toàn vẹn** và **nguồn gốc xác thực** của tập tin. Người gửi sử dụng private key RSA để ký tập tin, từ đó sinh ra một file chữ ký **.sig**. Người nhận có thể xác minh bằng public key tương ứng.

### Giao diện

Giao diện `/utils/sign_file` cho phép:

- Chọn file bất kỳ để ký
- Submit và tải về file chữ ký **.sig**
- Hiển thị thông báo (toast) khi ký thành công hoặc lỗi

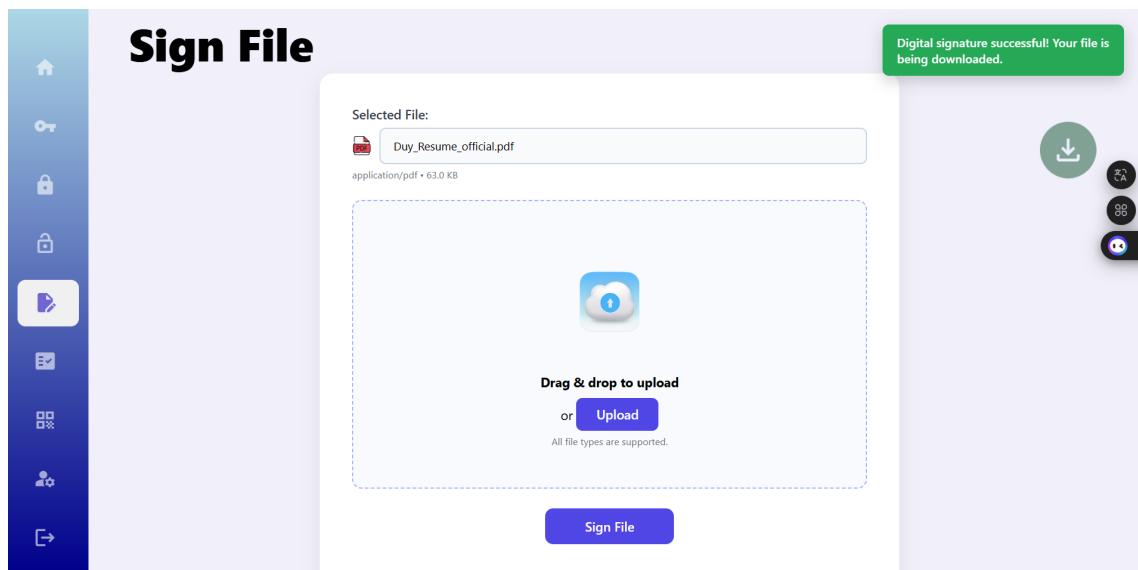


Hình 13: Giao diện ký số và nút tải file chữ ký

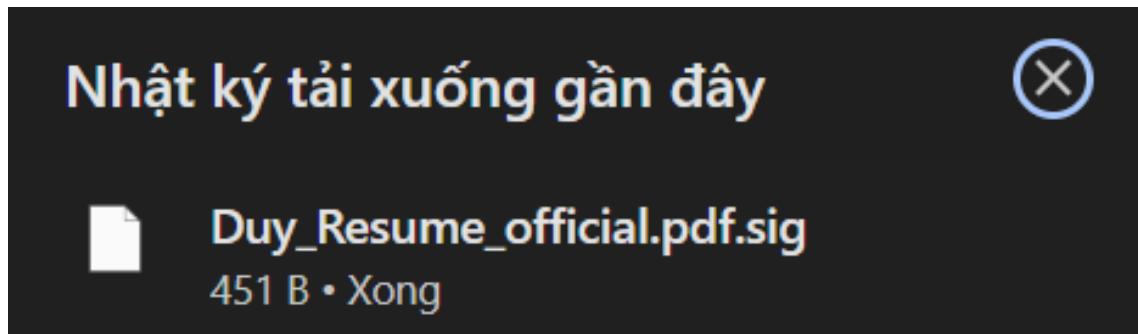
### Quy trình thực hiện

**Bước 1 - Frontend:** Người dùng chọn file và submit form. File được gửi lên server bằng `fetch()` qua `FormData`:

- Gửi POST đến `/utils/sign_file`
- Header `Content-Type` là `multipart/form-data`
- Khi server trả về file **.sig**, frontend tạo link download động và tự động tải về



Hình 14: Ký số thành công



Hình 15: Tự động tải file chữ ký

**Bước 2 - Flask Route:** Route `/sign_file` kiểm tra:

1. Đã đăng nhập và có session
2. Có passphrase trong session để giải mã khóa riêng
3. Tìm salt người dùng → derive AES key → giải mã private key
4. Gọi hàm `digital_sign_file(file, private_key)`
5. Trả về file `.sig` cho trình duyệt dưới dạng `application/octet-stream`

**Bước 3 - Xử lý ký số (Modules):** Hàm `digital_sign_file()` thực hiện:

1. Đọc toàn bộ nội dung file → tính hash SHA-256
2. Ký bằng `private_key.sign(...)` sử dụng PKCS1v15 + SHA-256
3. Base64 hóa chữ ký → tạo JSON `{"filename": ..., "signature": ..., "timestamp": ...}`

4. Ghi file chữ ký .sig vào thư mục data/signature/

```
{
  "timestamp": "2025-07-11T10:51:36.036912",
  "filename": "Duy_Resume_official.pdf",
  "signature": "10gcX9kkDu3nSwp4p2TBjJ7edvggYOa5l+GLS3b1HNC1iVMz8bZdOPePJ618koSxn6W7bbCRTNsE9rAs/"
}
```

Hình 16: Ghi file chữ ký

### Chi tiết kỹ thuật và thư viện bảo mật

**1. Ký số SHA-256 + RSA:** Quá trình ký số bắt đầu bằng việc băm nội dung tập tin bằng SHA-256, sau đó dùng khóa riêng RSA để tạo chữ ký số.

- Dùng `hashlib.sha256(file_bytes).digest()` để tính hash
- Dùng `cryptography.hazmat.primitives.asymmetric.rsa` để ký
- Padding: `PKCS1v15()`, Hash: `SHA256()`

**2. Mã hóa và giải mã private key:** Private key được bảo vệ bằng AES sử dụng passphrase người dùng. Khi cần ký, passphrase được lấy từ session để giải mã khóa.

- Private key được AES hóa bằng passphrase người dùng khi tạo khóa
- Để ký, passphrase được lấy từ session và dùng để giải mã khóa

**3. Ghi log:** Hệ thống ghi lại đầy đủ mọi thao tác ký dưới dạng log bảo mật – gồm cả hành động của người dùng và sự kiện nội bộ hệ thống.

- **Log người dùng:** sử dụng `log_user_action()` ghi lại email, hành động, trạng thái và lỗi nếu có
- **Log nội bộ:** `log_internal_event()` lưu các sự kiện kỹ thuật như “Signed filename successfully...”

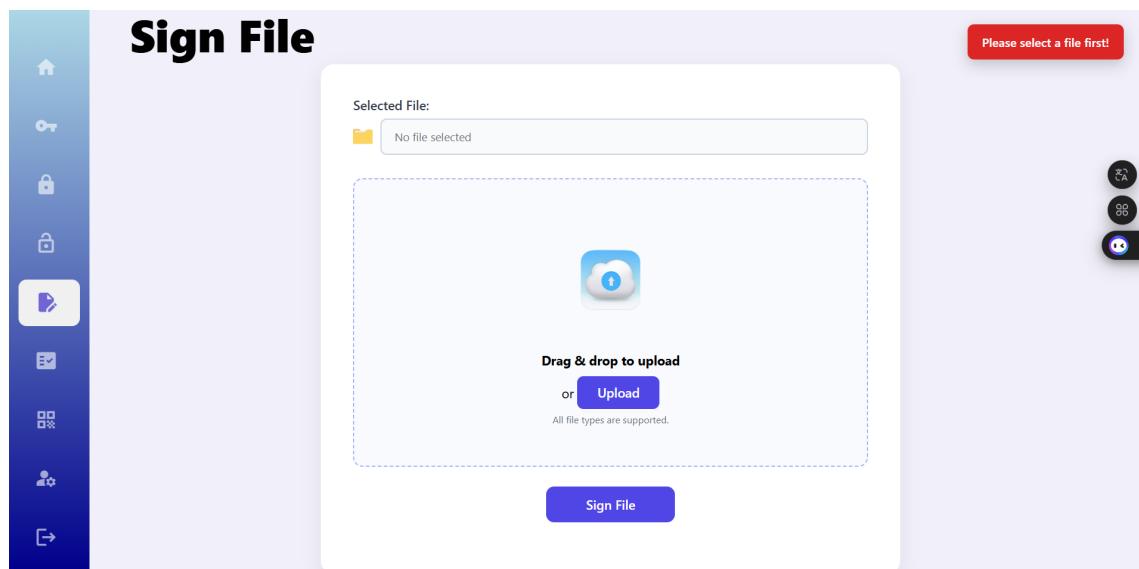
**4. Bảo mật tổng thể:** Quy trình ký số đảm bảo private key không bao giờ rời khỏi server, ngăn rò rỉ khóa, đồng thời có thể truy vết mọi hoạt động.

- Tránh truyền private key về phía client – xử lý toàn bộ phía server
- Tất cả chữ ký lưu kèm timestamp và filename để tăng khả năng truy vết
- Chỉ người dùng đã đăng nhập mới được phép truy cập chức năng ký

**5. Xử lý lỗi và báo lỗi chi tiết:** Hệ thống đảm bảo mọi tình huống lỗi trong quá trình ký số đều được kiểm tra, phản hồi rõ ràng cho frontend, đồng thời ghi log đầy đủ để phục vụ giám sát bảo mật.

- Kiểm tra session: Nếu chưa đăng nhập → trả lỗi 401 với thông báo "You must be logged in to access this page.".

- Kiểm tra file đầu vào: Nếu không có file hoặc tên file rỗng → trả lỗi 400: "No file provided.".
- Kiểm tra passphrase: Nếu thiếu → trả lỗi 401: "Passphrase not found.".
- Kiểm tra lỗi phát sinh khi đọc salt, giải mã khóa riêng hoặc ký file:
  - Nếu bất kỳ bước nào bị lỗi → trả lỗi 500 kèm thông báo: "Error during digital signing: <error>".
  - Toàn bộ lỗi được ghi log với level="error" để phục vụ phân tích.
- Ghi log thất bại bằng log\_user\_action(..., "Fail", ...) với lý do chi tiết cho từng trường hợp.



Hình 17: Báo lỗi nếu chưa chọn file

## 6.9 Xác minh chữ ký

### Mục tiêu

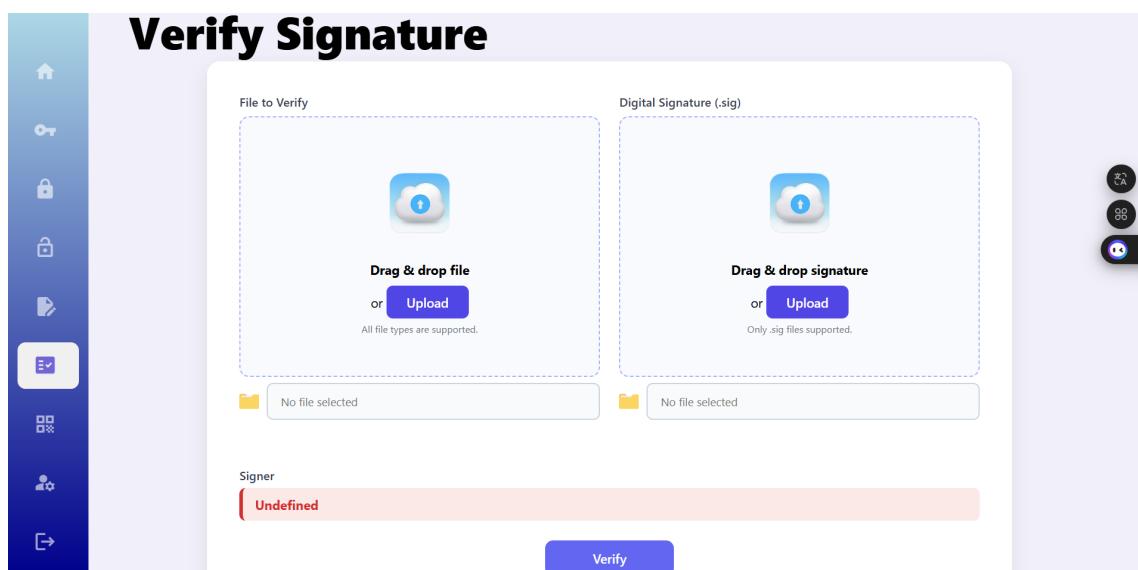
Chức năng xác minh chữ ký cho phép người dùng kiểm tra tính toàn vẹn và nguồn gốc của một tập tin bất kỳ thông qua file chữ ký ‘.sig’ do người gửi tạo ra. Việc xác minh này giúp đảm bảo:

- Tập tin không bị chỉnh sửa sau khi ký.
- Người nhận có thể xác định chính xác ai là người ký.
- Thời điểm ký được xác minh và hiển thị lại rõ ràng.

### Giao diện

Giao diện xác minh tại trang /crypto/verify bao gồm:

- Trường upload 2 file: tập tin gốc cần xác minh và file chữ ký ‘.sig’.
- Nút “Xác minh chữ ký” gửi yêu cầu lên server bằng `fetch()`.
- Kết quả xác minh hiển thị thông tin người ký và thời điểm ký nếu hợp lệ.
- Toast thông báo xác minh thành công hoặc thất bại.



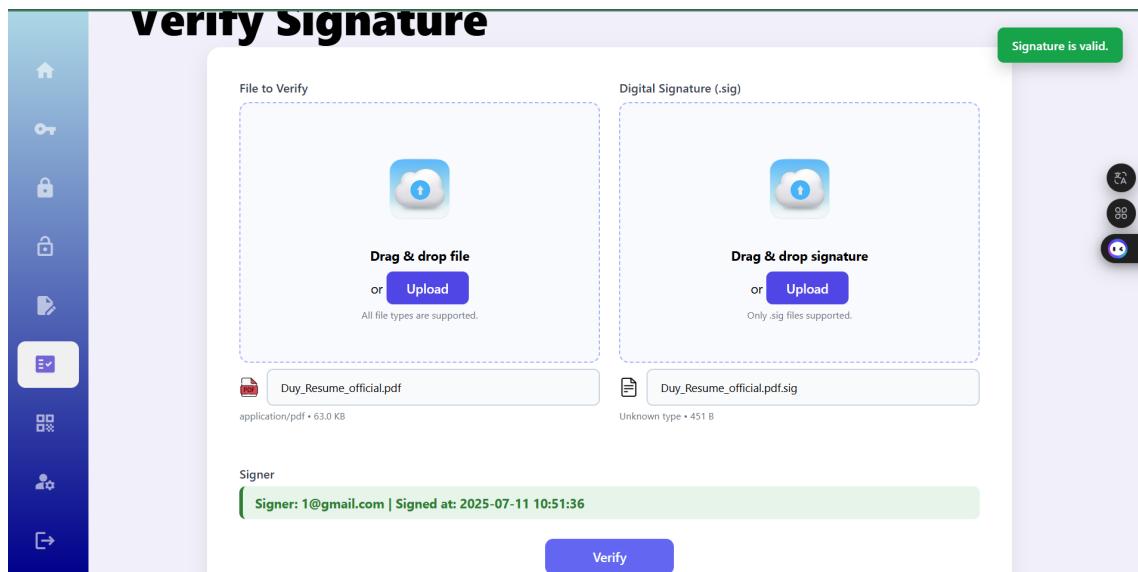
Hình 18: Giao diện xác minh ký số

### Quy trình thực hiện

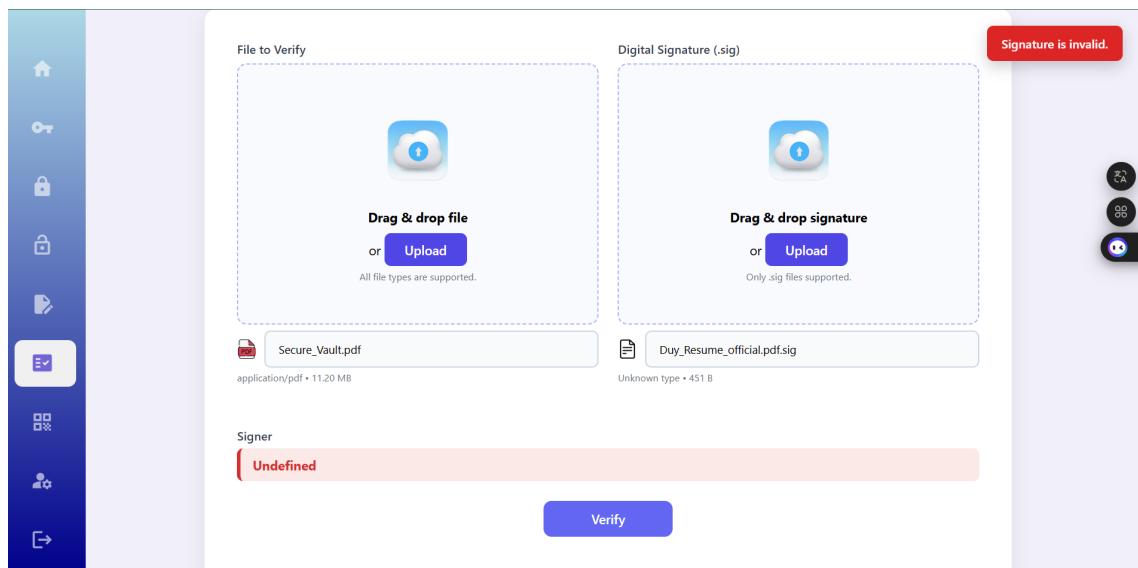
**Bước 1 - Frontend:** Người dùng chọn tập tin và file chữ ký ‘.sig’, sau đó gửi lên server bằng `FormData`.

- Gửi POST đến `/utils/verify_signature`.
- Gửi kèm 2 trường: `file_to_verify` và `signature`.

- Nếu kết quả hợp lệ → hiển thị tên người ký và thời điểm ký.



Hình 19: Xác minh ký số thành công



Hình 20: Chữ ký không được xác minh

**Bước 2 - Flask Route:** Route xử lý tại /verify\_signature.

- Kiểm tra session và sự tồn tại của cả 2 file.
- Đọc nội dung file chữ ký và parse JSON.
- Đọc danh sách public key từ file contact\_public\_key.json.
- Gọi hàm verify\_signature(...) để xác minh chữ ký.

5. Ghi log kết quả (thành công hoặc thất bại) và trả về thông tin người ký.

**Bước 3 - Xử lý xác minh (Modules):** Hàm `verify_signature()` thực hiện:

1. Tính hash SHA-256 của tập tin gốc.
2. Dò từng public key trong danh sách để thử xác minh.
3. Nếu khớp → trả về email người ký và thời gian.
4. Nếu không khớp → trả lỗi “Signature is invalid”.

#### Chi tiết kỹ thuật và thư viện bảo mật

**1. Phân tích file chữ ký:** File chữ ký ‘.sig’ là JSON chứa chữ ký (dạng base64) và timestamp. Hệ thống decode và parse dữ liệu để chuẩn bị xác minh.

- Đọc bằng `.read().decode('utf-8')` và xử lý JSON với `json.loads(...)`.
- Giải mã chữ ký với `base64.b64decode(...)`.

**2. Duyệt danh sách public key:** Server đọc file `contact_public_key.json` chứa public key của các người gửi.

- Public key lưu dưới dạng PEM, được parse bằng `serialization.load_pem_public_key()`.
- Với mỗi key, thử gọi `public_key.verify(...)` để kiểm tra.

**3. Tính hash và xác minh:** Server tính hash SHA-256 của file gốc và dùng RSA-PKCS1v15 để xác minh chữ ký.

- Hash bằng: `hashlib.sha256(file_data).digest()`.
- Padding: `PKCS1v15()`, Hash: `SHA256()`.
- Nếu đúng → xác minh thành công và log email + timestamp.

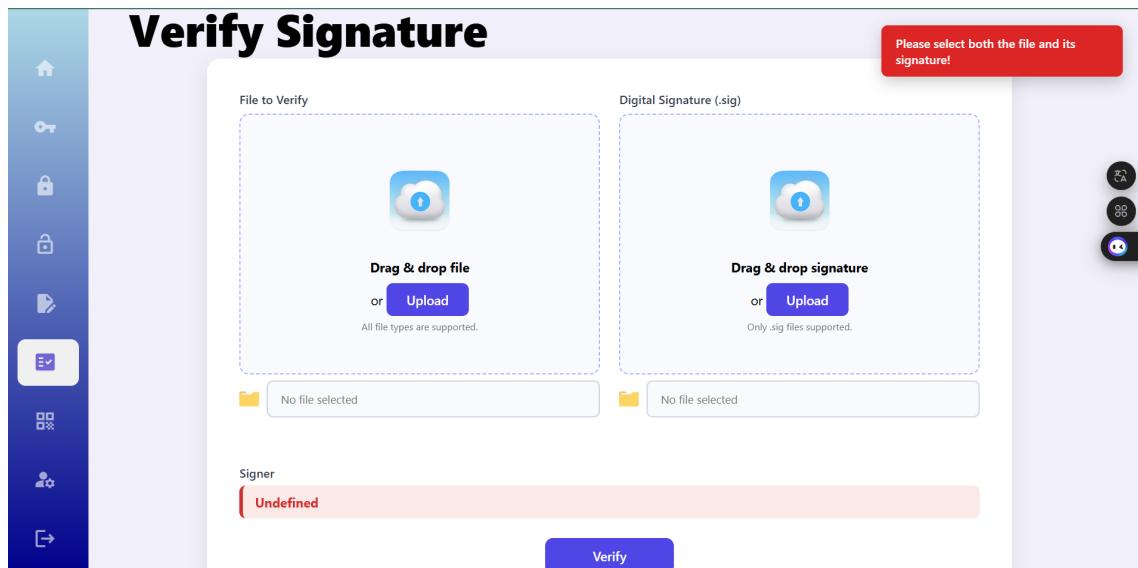
**4. Xử lý lỗi và bảo mật:** Hệ thống có các bước kiểm tra và log lỗi đầy đủ:

- Báo lỗi nếu thiếu file, sai định dạng hoặc không tìm thấy key.
- Ghi log bằng `log_user_action()` và `log_internal_event()` theo mức độ `info`, `warning`, hoặc `error`.
- Không lưu lại nội dung file, chỉ log metadata để đảm bảo riêng tư.

**5. Xử lý lỗi và báo lỗi chi tiết:** Hệ thống được thiết kế để phát hiện và phản hồi rõ ràng với các lỗi có thể xảy ra trong quá trình xác minh chữ ký, đồng thời ghi log đầy đủ theo từng tình huống cụ thể.

- **Kiểm tra đăng nhập:** Nếu session không tồn tại, trả về mã lỗi 401 cùng thông báo: “`You must be logged in to access this page.`”.
- **Thiếu file:** Nếu người dùng không gửi cả file cần xác minh hoặc file chữ ký → trả lỗi 400 kèm thông báo: “`No file or signature provided`”.
- **Tên file trống:** Nếu `file.filename == ""` → trả lỗi 400 với thông báo: “`Tên file trống`”.

- **Lỗi định dạng chữ ký:**
  - Nếu không đọc hoặc decode được nội dung file chữ ký JSON → trả lỗi 400, thông báo: "Invalid signature file format".
  - Log lỗi này ở mức error vì có thể liên quan đến file bị sửa hoặc không hợp lệ.
- **Thiếu danh sách public key:**
  - Nếu không tồn tại file contact\_public\_key.json → trả lỗi 400 với thông báo: "Public key does not exist.".
  - Lỗi được log lại kèm tên người dùng để hỗ trợ debug.
- **Chữ ký không hợp lệ:**
  - Nếu không có public key nào xác minh được chữ ký → trả lỗi 400 với thông báo: "Signature is invalid.".
  - Thông tin file được log kèm để phục vụ điều tra.
- **Log chi tiết:** Mỗi lỗi đều được ghi bằng log\_user\_action(...) với trạng thái "Fail" và ghi rõ lý do, đồng thời có thể bổ sung thêm log\_internal\_event() để phân tích kỹ thuật.



Hình 21: Kiểm tra có đủ file đầu vào

## 6.10 Phân quyền tài khoản

Mục tiêu

Giao diện

Quy trình thực hiện

Chi tiết kỹ thuật và thư viện bảo mật

## 6.11 Ghi log bảo mật

### Mục tiêu

Chức năng ghi log bảo mật nhằm đảm bảo toàn bộ hành động của người dùng và các sự kiện nội bộ trong hệ thống đều được theo dõi, ghi nhận và lưu trữ dưới dạng thống nhất. Việc này phục vụ:

- Truy vết các hành động quan trọng: đăng nhập, ký số, xác minh,...
- Giám sát lỗi bảo mật và cảnh báo bất thường.
- Làm bằng chứng trong điều tra sự cố hoặc kiểm toán.

### Giao diện

Giao diện tại trang /admin\_dashboard cung cấp:

- Bảng hiển thị đầy đủ log: thời gian, mức độ (INFO, WARNING, ERROR), email người dùng, hành động, trạng thái và chi tiết.
- Tính năng tìm kiếm realtime và nút Reload để làm mới dữ liệu.
- Dữ liệu được render động thông qua fetch X-Requested-With = XMLHttpRequest.

#	Level	Time	Email	Action	Status	Message
1	INFO	2025-07-09 10:54:14	yuddyd oo1405 @gmail. com	Logout	Success	Logout successfully
2	INFO	2025-07-09 14:13:28	yuddyd oo1405 @gmail. com	Login	Success	Login successfully
3	INFO	2025-07-09 14:13:32	yuddyd oo1405 @gmail. com	Send OTP	Success	OTP sent via email

Hình 22: Bảng log của admin

### Quy trình thực hiện

- Người dùng thực hiện hành động** Mỗi khi người dùng thao tác (ví dụ: đăng nhập, ký file, xác minh), hàm log\_user\_action() được gọi để ghi log với thông tin đầy đủ: email, hành động, trạng thái, chi tiết và mức độ severity.

```
[2025-07-09 10:54:14] [INFO] [yuddydoo1405@gmail.com] Action=Logout | Status=Success | Details=Logout successfully
[2025-07-09 14:13:28] [INFO] [yuddydoo1405@gmail.com] Action=Login | Status=Success | Details=Login successfully
[2025-07-09 14:13:32] [INFO] [yuddydoo1405@gmail.com] Action=Send OTP | Status=Success | Details=OTP sent via email
[2025-07-09 14:13:50] [INFO] [admin@fitus.edu.vn] Action=Login | Status=Success | Details=Login successfully
[2025-07-09 14:14:02] [INFO] [admin@fitus.edu.vn] Action=Send OTP | Status=Success | Details=OTP sent via email
[2025-07-09 14:14:18] [INFO] [admin@fitus.edu.vn] Action=TOTP Verify | Status=Success
[2025-07-09 14:13:50] [INFO] [admin@fitus.edu.vn] Action=Login | Status=Success | Details=Login successfully
[2025-07-09 14:14:02] [INFO] [admin@fitus.edu.vn] Action=Send OTP | Status=Success | Details=OTP sent via email
[2025-07-09 14:14:18] [INFO] [admin@fitus.edu.vn] Action=TOTP Verify | Status=Success\[2025-07-09 15:05:33] [INFO] [admin@fitus.edu.vn] Action=Login | Status=Success | Details=Login successfully
[2025-07-09 17:31:42] [INFO] [yuddydoo1405@gmail.com] Action=Login | Status=Success | Details=Login successfully
[2025-07-09 17:31:46] [INFO] [yuddydoo1405@gmail.com] Action=Send OTP | Status=Success | Details=OTP sent via email
[2025-07-09 17:31:58] [WARNING] [yuddydoo1405@gmail.com] Action=OTP Verify | Status=Fail | Details=Invalid or expired OTP
[2025-07-09 17:32:07] [INFO] [yuddydoo1405@gmail.com] Action=OTP Verify | Status=Success
[2025-07-09 17:32:10] [INFO] [yuddydoo1405@gmail.com] Action=Get User Info | Status=Success | Details=User profile returned
[2025-07-09 17:33:31] [INFO] [yuddydoo1405@gmail.com] Action=Get User Info | Status=Success | Details=User profile returned
[2025-07-09 17:34:35] [INFO] [yuddydoo1405@gmail.com] Action=Get User Info | Status=Success | Details=User profile returned
[2025-07-09 17:37:58] [INFO] [yuddydoo1405@gmail.com] Action=Get User Info | Status=Success | Details=User profile returned
[2025-07-09 17:38:08] [INFO] [yuddydoo1405@gmail.com] Action=Get User Info | Status=Success | Details=User profile returned
[2025-07-09 17:39:04] [INFO] [yuddydoo1405@gmail.com] Action=Get User Info | Status=Success | Details=User profile returned
[2025-07-09 17:39:05] [WARNING] [yuddydoo1405@gmail.com] Action=Update Account Info | Status=Fail | Details>No changes were made.
[2025-07-09 17:39:07] [WARNING] [yuddydoo1405@gmail.com] Action=Update Account Info | Status=Fail | Details>No changes were made.
[2025-07-09 17:39:08] [WARNING] [yuddydoo1405@gmail.com] Action=Update Account Info | Status=Fail | Details>No changes were made.
[2025-07-09 17:39:08] [WARNING] [yuddydoo1405@gmail.com] Action=Update Account Info | Status=Fail | Details>No changes were made.
[2025-07-10 07:53:24] [INFO] [1@gmail.com] Action/Register | Status=Success | Details=Registration successful!
[2025-07-10 07:53:57] [INFO] [1@gmail.com] Action=Login | Status=Success | Details=Login successfully
[2025-07-10 07:54:00] [INFO] [1@gmail.com] Action=Send OTP | Status=Success | Details=OTP sent via email
```

Hình 23: Log cho người dùng - Routes

**2. Hệ thống nội bộ phát sinh sự kiện** Các module nội bộ (như mã hóa, xác minh chữ ký) có thể gọi log\_internal\_event() để ghi lại log kỹ thuật chi tiết.

```
1 [2025-07-10 07:54:37] [INFO] [digital_signature]: Signed Duy_Resume_official.pdf successfully and saved to data/signature\Duy_Resume_official.pdf.sig.
[2025-07-10 07:56:39] [INFO] [digital_signature]: Verified signature for Duy_Resume_official.pdf successfully.
[2025-07-10 07:59:23] [INFO] [digital_signature]: Verified signature for Duy_Resume_official.pdf successfully.
[2025-07-11 10:50:51] [INFO] [digital_signature]: Signed Duy_Resume_official.pdf successfully and saved to data/signature\Duy_Resume_official.pdf.sig.
[2025-07-11 10:50:59] [INFO] [digital_signature]: Signed Duy_Resume_official.pdf successfully and saved to data/signature\Duy_Resume_official.pdf.sig.
[2025-07-11 10:51:15] [INFO] [digital_signature]: Signed Duy_Resume_official.pdf successfully and saved to data/signature\Duy_Resume_official.pdf.sig.
[2025-07-11 10:51:36] [INFO] [digital_signature]: Signed Duy_Resume_official.pdf successfully and saved to data/signature\Duy_Resume_official.pdf.sig.
[2025-07-11 11:02:56] [INFO] [digital_signature]: Verified signature for Duy_Resume_official.pdf successfully.
[2025-07-11 11:03:02] [INFO] [digital_signature]: Verified signature for Duy_Resume_official.pdf successfully.
[2025-07-11 11:03:40] [WARNING] [digital_signature]: Failed to verify signature for AIoT-Group5-Slide.pptx.pdf.
[2025-07-11 11:03:44] [WARNING] [digital_signature]: Failed to verify signature for AIoT-Group5-Slide.pptx.pdf.
[2025-07-11 11:03:46] [WARNING] [digital_signature]: Failed to verify signature for AIoT-Group5-Slide.pptx.pdf.
[2025-07-11 11:04:23] [WARNING] [digital_signature]: Failed to verify signature for Secure_Vault.pdf.
[2025-07-11 11:06:49] [WARNING] [digital_signature]: Failed to verify signature for Secure_Vault.pdf.
[2025-07-11 11:07:06] [WARNING] [digital_signature]: Failed to verify signature for Secure_Vault.pdf.
[2025-07-11 11:07:36] [WARNING] [digital_signature]: Failed to verify signature for Secure_Vault.pdf.
```

Hình 24: Log cho debug - Modules

**3. Lưu vào file log** Log người dùng và log nội bộ để debug sẽ được lưu vào các file riêng

- Log người dùng được ghi vào log/security.log
- Log module nội bộ được ghi vào log/debug\_log.log

**4. Giao diện đọc log** Khi người dùng truy cập trang /admin\_dashboard, Flask route đọc và phân tích nội dung từ file log, chuyển thành danh sách JSON và hiển thị trên giao diện.

### Chi tiết kỹ thuật và thư viện bảo mật

**1. Hàm log người dùng** log\_user\_action(email, action, status, details, level) ghi log theo format thống nhất:

- Ví dụ log: [2025-07-09 10:15:23] [INFO] [user@example.com] Action=Sign File | Status=Success | Details=file=report.pdf

- Sử dụng thư viện chuẩn logging, ghi vào log/security.log

**2. Log nội bộ hệ thống** log\_internal\_event(module, message, level) dùng cho debug các module như crypto, xác minh,...

- Ví dụ: [crypto]: Signature verified successfully.
- Ghi vào log/debug\_log.log với format chi tiết hơn để phục vụ debug.

**3. Route hiển thị log** Route /log\_security thực hiện:

- Đọc file log/security.log, tách thành các trường: timestamp, level, user, action, status, details
- Trả về JSON nếu là AJAX, hoặc render giao diện nếu là truy cập thường

**4. Mức độ log hỗ trợ** • INFO – hành động thành công hoặc hợp lệ

- WARNING – thao tác sai, lỗi thường gặp
- ERROR – lỗi hệ thống hoặc dữ liệu bất thường
- DEBUG – dành cho log kỹ thuật nội bộ (chỉ module log mới dùng)

**5. Định dạng và chuẩn hóa log** Mọi log đều tuân thủ format:

[timestamp] [LEVEL] [email] Action=... | Status=... | Details=...

Điều này giúp dễ dàng phân tích bằng tool, lọc log, và hỗ trợ audit.

## 6.12 Chia nhỏ tập tin lớn

Mục tiêu

Giao diện

Quy trình thực hiện

Chi tiết kỹ thuật và thư viện bảo mật

## 6.13 Kiểm tra trạng thái khoá

Mục tiêu

Giao diện

Quy trình thực hiện

Chi tiết kỹ thuật và thư viện bảo mật

## 6.14 TÌM KIẾM PUBLIC KEY

### Mục tiêu

Chức năng này cho phép người dùng tìm kiếm trong danh sách các public key đã lưu (của các liên hệ khác). Việc quản lý và tìm nhanh public key giúp người dùng dễ dàng sử dụng trong quá trình mã hóa, xác minh chữ ký, và chia sẻ an toàn.

### Giao diện

Giao diện tại tab `Owned Public Keys` trong dashboard bao gồm:

- Bảng hiển thị danh sách các public key đã lưu: tên người gửi, email, timestamp và public key.
- Ô tìm kiếm theo email hoặc public key (tìm mờ, không phân biệt hoa thường).
- Dữ liệu được lấy động từ API `/owned_keys`.

The screenshot shows a dashboard interface titled 'QR Code'. On the left is a sidebar with icons for Home, Keys, Locks, QR Codes, and Contacts. The main area has tabs for 'My QR Info', 'Scan Other', and 'Owned Keys' (which is selected). Below the tabs is a section titled 'Owned Public Keys' with the sub-instruction: 'Below is the list of public keys you have scanned and saved.' A search bar is present with the placeholder 'Search by email or public key...'. A table lists the saved keys:

#	Email	Creation date	Expiry date	Public Key	Status
1	1@gmail.com	2025-07-10	2025-10-08	MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIIBCgK...	Valid

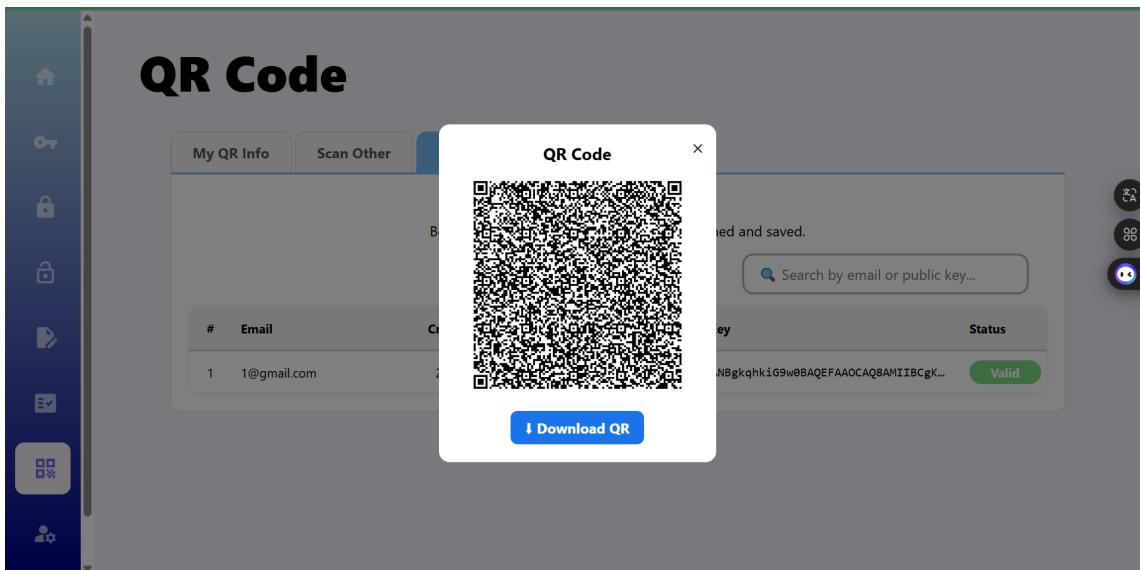
Hình 25: Danh sách contact - public key đã lưu

### Quy trình thực hiện

- Truy cập tab Owned Keys** Người dùng đăng nhập và truy cập trang dashboard → phần "Owned Public Keys" sẽ gửi request GET đến API `/owned_keys`.
- API lấy dữ liệu** Flask route `/owned_keys` thực hiện:
  - Kiểm tra phiên đăng nhập.
  - Đọc file `contact_public_key.json` theo thư mục cá nhân người dùng.
  - Chuyển danh sách public key về dạng JSON và trả về frontend.
- Hiển thị và tìm kiếm**
  - JavaScript tạo bảng dữ liệu từ kết quả JSON trả về.
  - Khi người dùng gõ từ khóa vào ô tìm kiếm, hàm `filterPublicKeys()` lọc các hàng phù hợp theo email hoặc chuỗi public key.

## Chi tiết kỹ thuật và thư viện bảo mật

1. **Cấu trúc lưu public key**
  - Mỗi người dùng có một file riêng: <user\_email>/contact\_public\_key.json
  - Dạng JSON: {"abc@example.com": {"name": "...", "public\_key\_pem": "..."}}
2. API trả dữ liệu public key /owned\_keys là route GET:
  - Nếu chưa đăng nhập → trả lỗi 401 Unauthorized.
  - Nếu file danh bạ không tồn tại → trả danh sách rỗng.
  - Ghi log trạng thái truy vấn bằng log\_user\_action(...) với số lượng public key tìm được.
3. **Tìm kiếm phía client**
  - Hàm filterPublicKeys() lọc dữ liệu theo từ khóa không phân biệt hoa thường.
  - Kiểm tra xem từ khóa xuất hiện trong email hoặc chuỗi public key PEM.
  - Lọc trực tiếp trên DOM → không gọi lại API khi tìm kiếm.
4. **Hiển thị QR code public key** Hệ thống hỗ trợ hiển thị mã QR của public key khi người dùng click vào một contact cụ thể trong bảng danh sách. Điều này giúp việc chia sẻ public key giữa người dùng dễ dàng và nhanh chóng hơn (qua quét mã bằng điện thoại, Google Authenticator,...).
  - Giao diện sử dụng sự kiện onClick hoặc nút View QR gắn với mỗi hàng trong bảng.
  - Khi người dùng bấm chọn, mã public key PEM tương ứng sẽ được gửi đến route /utils/generate\_qr (hoặc sinh trực tiếp frontend).
  - QR code được sinh từ chuỗi PEM hoặc JSON chứa các thông tin liên hệ và khóa công khai.
  - Kết quả được hiển thị trong một modal hoặc khung bên cạnh dòng đã chọn.
  - Mã QR cho phép người khác dễ dàng quét để lưu lại public key → hỗ trợ xác thực và mã hóa an toàn.



Hình 26: Pop-up QR của contact đã lưu

**5. Bảo mật truy cập** Chỉ người dùng đã đăng nhập mới được phép truy vấn danh sách public key đã lưu.

- File JSON được lưu riêng trong thư mục người dùng → cô lập dữ liệu từng tài khoản.

## 6.15 Giới hạn đăng nhập

Mục tiêu

Giao diện

Quy trình thực hiện

Chi tiết kỹ thuật và thư viện bảo mật

## 6.16 Tùy chọn định dạng lưu file

Mục tiêu

Giao diện

Quy trình thực hiện

Chi tiết kỹ thuật và thư viện bảo mật

## 6.17 Khôi phục tài khoản

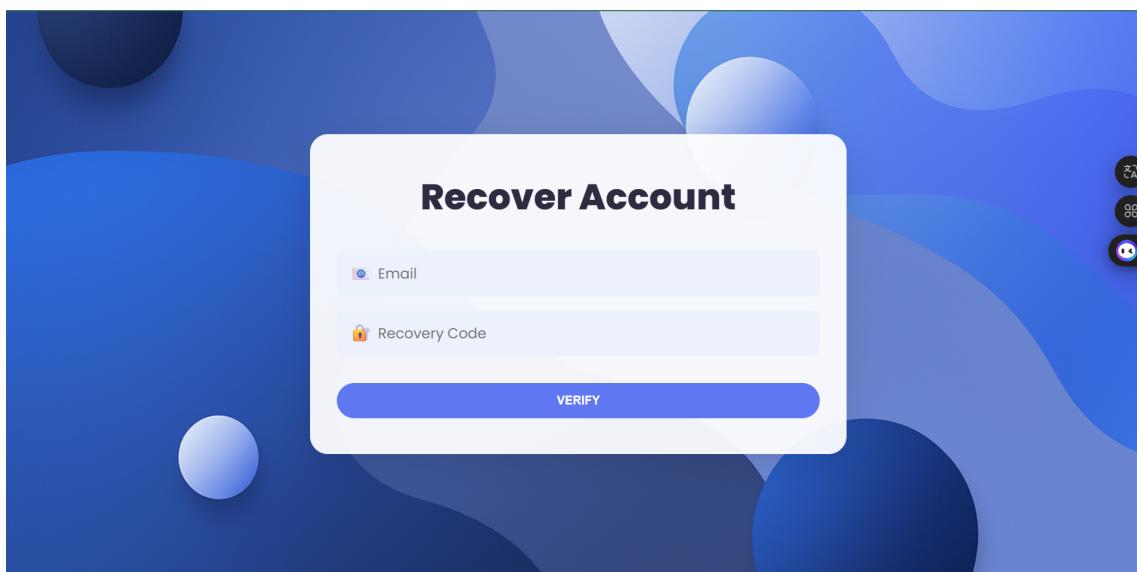
### Mục tiêu

Chức năng khôi phục tài khoản cho phép người dùng đặt lại mật khẩu (passphrase) khi không còn nhớ mật khẩu cũ. Khác với đặt lại mật khẩu thông thường, hệ thống vẫn đảm bảo khôi phục được private key đã mã hóa trước đó – nhờ sử dụng một **recovery code** (AES key phụ) đã được lưu trữ an toàn từ khi đăng ký.

### Giao diện

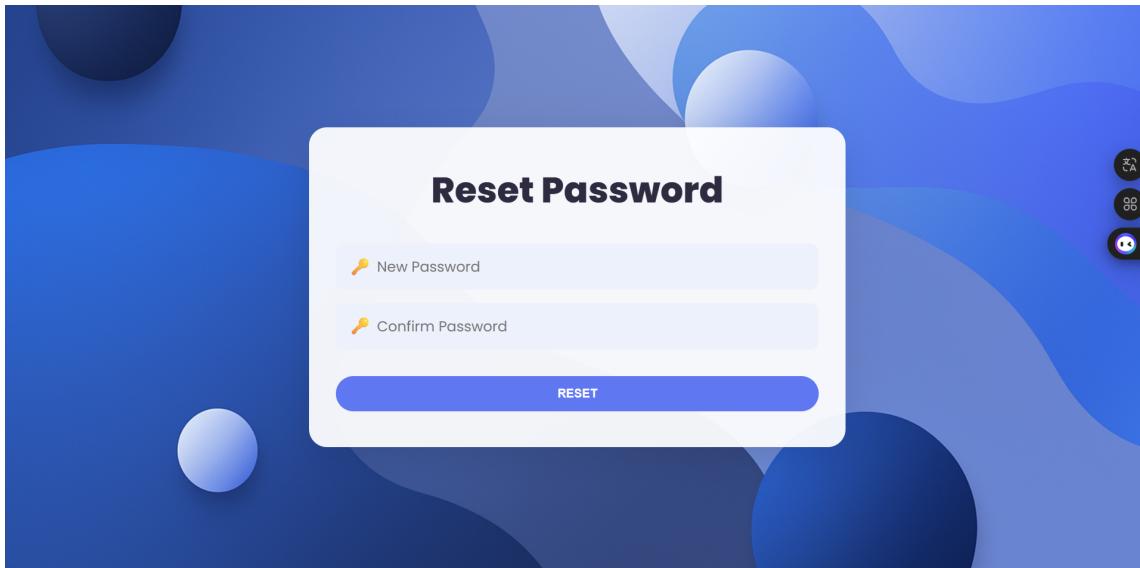
Giao diện khôi phục gồm 2 bước:

- **Bước 1:** Người dùng nhập email và recovery code để xác minh quyền sở hữu tài khoản.



Hình 27: Nhập mã khôi phục

- **Bước 2:** Nếu thành công, hệ thống cho phép nhập mật khẩu mới (passphrase), sau đó sẽ mã hóa lại khóa riêng và recovery key.



Hình 28: Nhập mật khẩu mới

### Quy trình thực hiện

#### Bước 1 - Xác minh recovery code

- Người dùng nhập recovery code → hệ thống derive AES key từ recovery code đó.
- Dùng AES key này để giải mã file `key_recovery` chứa private key hiện tại.
- Nếu giải mã thành công → recovery code hợp lệ.

#### Bước 2 - Nhập mật khẩu mới

- Người dùng nhập passphrase mới.
- Hệ thống mã hóa lại private key hiện tại bằng AES key derive từ passphrase mới.
- Đồng thời, mã hóa lại recovery code bằng AES key derive từ passphrase mới → lưu vào cột `encrypted_recovery_code` trong database.

#### Bước 3 - Cập nhật cơ sở dữ liệu

- Hash passphrase mới và cập nhật vào cột `hashed_passphrase`.
- Ghi log hành động khôi phục với trạng thái thành công hoặc thất bại.

### Chi tiết kỹ thuật và thư viện bảo mật

#### 1. Recovery code

Recovery code là một chuỗi ngẫu nhiên sinh ra khi đăng ký, dùng để derive AES key tạm thời cho mục đích khôi phục khóa.

- Được dùng để giải mã private key hiện tại khi muốn reset password.
- Được mã hóa bằng AES key derive từ passphrase và lưu dưới database.

#### 2. Kiểm tra tính hợp lệ của recovery code

- Dùng recovery code derive AES key: `derive_aes_key(recovery_code)`
- Giải mã file `key_recovery` (chứa private key mã hóa) bằng AES key này.
- Nếu giải mã được → xác minh thành công.

#### 3. Đặt lại mật khẩu và mã hóa lại dữ liệu

- Derive AES key từ passphrase mới.

- Mã hóa lại private key bằng AES key mới.
- Mã hóa recovery code bằng AES key mới và lưu vào `encrypted_recovery_code`.
- Hash passphrase mới với salt → cập nhật vào bảng `users`.

**4. Xử lý lỗi và báo lỗi chi tiết** Hệ thống đảm bảo mọi lỗi trong quá trình khôi phục tài khoản đều được phát hiện, phản hồi rõ ràng cho người dùng, đồng thời ghi log bảo mật để hỗ trợ điều tra.

- **Thiếu email hoặc recovery code:**
  - Nếu không gửi đủ thông tin đầu vào → trả lỗi 400 với thông báo: "Missing recovery information".
- **Recovery code không hợp lệ:**
  - Nếu recovery code không trùng khớp hoặc giải mã private key thất bại → trả lỗi 400, thông báo: "Invalid recovery code".
  - Lỗi này thường do nhập sai hoặc đã đổi passphrase trước đó mà chưa cập nhật lại recovery key.
- **Mật khẩu mới không hợp lệ:**
  - Nếu passphrase mới không đủ mạnh (dưới 8 ký tự, không có ký tự đặc biệt, chữ hoa, số) → từ chối và trả thông báo cụ thể.
- **Lỗi mã hóa lại private key hoặc recovery key:**
  - Nếu trong quá trình mã hóa lại private key bằng passphrase mới xảy ra lỗi → rollback thao tác DB và trả về lỗi "Re-encrypt RSA failed".
  - Nếu mã hóa lại recovery key lỗi → báo lỗi "Failed to encrypt recovery key" và không cập nhật thông tin.
- **Lỗi cập nhật CSDL:**
  - Nếu xảy ra lỗi khi cập nhật passphrase hoặc encrypted recovery key trong database → trả lỗi "Database error" và log ở mức `error`.
- **Log đầy đủ:**
  - Tất cả lỗi đều được ghi lại bằng `log_user_action(...)` với trạng thái "Fail" và nội dung chi tiết.
  - Các lỗi hệ thống (mã hóa thất bại, không đọc được recovery file, lỗi DB) được log với `level = "error"`.

## 7 Kiểm thử ứng dụng

Tất cả hình ảnh kết quả kiểm thử được cập nhật đầy đủ trong drive sau

### 6.1 Đăng ký tài khoản người dùng (/signup)

Mô tả kiểm thử	Input	Kết quả mong đợi
Đăng ký thành công	Email hợp lệ, Passphrase mạnh, các trường đầy đủ	Thông báo "Registration successful", pop-up <code>recovery_code</code>
Email sai định dạng	<code>abc@.com</code>	Báo lỗi "Invalid email format"
Passphrase yếu	abc12345	Báo lỗi "Passphrase too weak"
Trùng email đã đăng ký	Email đã có trong DB	Báo lỗi "Account already exists"
XSS/SQL injection	<code>&lt;script&gt;, " OR "1"="1</code>	Bị sanitize, không lỗi hệ thống

### 6.2 Đăng nhập và xác thực đa yếu tố (/login → /verify)

Mô tả kiểm thử	Input	Kết quả mong đợi
Đăng nhập đúng pass → chờ xác thực OTP	Email và passphrase đúng	Chuyển hướng tới trang <code>/verify</code> để xác thực OTP hoặc TOTP
Sai pass < 5 lần	Email đúng, passphrase sai	Báo lỗi "Wrong email or password" và tăng <code>failed_attempts</code>
Sai pass 5 lần	Nhập sai liên tiếp 5 lần	Khóa tài khoản 5 phút, thông báo thời gian chờ
OTP hợp lệ trong thời gian	Mã OTP 6 chữ số từ email (trong vòng 5 phút)	Xác thực thành công, chuyển đến <code>/dashboard</code> hoặc <code>/admin_dashboard</code>
OTP sai hoặc hết hạn	Mã OTP sai hoặc quá hạn 5 phút	Báo lỗi "Invalid or expired OTP"
TOTP đúng từ Google Authenticator	Mã 6 chữ số từ ứng dụng	Xác thực thành công, chuyển trang chính
TOTP sai	Nhập sai mã TOTP	Báo lỗi "Invalid TOTP code"

### 6.3 Quản lý khóa RSA cá nhân (/manage\_keys)

Mô tả kiểm thử	Input	Kết quả mong đợi
Tạo cặp khóa thành công	Bấm nút "Tạo khóa RSA" sau khi đăng nhập	Sinh file private và public '.pem', lưu DB với ngày tạo và hạn 90 ngày
Kiểm tra trạng thái khóa	Tài khoản đã có khóa	Hiển thị trạng thái: Còn hạn / Gần hết hạn / Hết hạn
Giải mã private key thành công	Passphrase đúng	Mở được private key để ký / giải mã
Giải mã private key thất bại	Passphrase sai	Báo lỗi "Unable to decrypt private key"

### 6.4 QR Code Public Key (/utils/qr)

Mô tả kiểm thử	Input	Kết quả mong đợi
Tạo QR thành công	Email có public key	Tạo mã QR base64, lưu file PNG
Quét QR thành công	File ảnh QR đúng định dạng	Hiển thị: email, public key, ngày tạo
Quét file ảnh sai định dạng	PNG không chứa mã QR hoặc bị lỗi	Thông báo "Không thể đọc mã QR"

### 6.5 Cập nhật thông tin tài khoản (/update\_account)

Mô tả kiểm thử	Input	Kết quả mong đợi
Cập nhật thông tin thành công	Họ tên, địa chỉ, SĐT đúng định dạng	Lưu thay đổi thành công, reload dữ liệu
Đổi passphrase thành công	Pass cũ đúng, pass mới đủ mạnh	Passphrase thay đổi, AES key tự cập nhật
Pass cũ sai	Nhập sai passphrase hiện tại	Báo lỗi "Passphrase hiện tại không đúng"
Pass mới yếu	Mới <8 ký tự hoặc không đủ yêu cầu	Báo lỗi "Passphrase mới quá yếu"

## 6.6 Mã hoá tập tin gửi người khác (/crypto/encrypt)

Mô tả kiểm thử	Input	Kết quả mong đợi
Mã hoá thành công (gộp file)	Chọn file + email người nhận có public key	Tạo file <code>.enc</code> chứa dữ liệu mã hoá và metadata
Mã hoá thành công (tách file)	Chọn lưu dạng tách	Sinh file <code>.enc</code> và <code>.key</code> riêng biệt
Người nhận không có public key	Nhập email chưa có key lưu trữ	Báo lỗi "Không tìm thấy public key của người nhận"
Metadata đúng định dạng	Sau khi mã hoá	Metadata gồm người gửi, thời gian, thuật toán, định dạng

## 6.7 Giải mã tập tin (/crypto/decrypt)

Mô tả kiểm thử	Input	Kết quả mong đợi
Giải mã thành công (file gộp)	File <code>.enc</code> gộp và passphrase đúng	Giải mã thành công, hiển thị file gốc và metadata
Giải mã thành công (file tách)	<code>.enc</code> + <code>.key</code> + passphrase đúng	Khôi phục file gốc đúng nội dung
Sai passphrase / khóa sai	Passphrase không đúng hoặc thiếu <code>.key</code>	Báo lỗi "Decryption failed" hoặc "Unable to decrypt key"
Tự động nhận dạng định dạng file	Dù là gộp hay tách	Tự động phân tích đúng định dạng và giải mã phù hợp

## 6.8 Ký số tập tin (/crypto/sign)

Mô tả kiểm thử	Input	Kết quả mong đợi
Ký số thành công	Chọn file bất kỳ + passphrase đúng	Tạo file chữ ký <code>.sig</code> theo định dạng SHA-256 + RSA
Thiếu private key hoặc passphrase sai	Không giải mã được private key	Báo lỗi "Không thể ký tập tin"
Log ký số đúng	Sau ký thành công	Ghi log: email người ký, thời gian ký, file đã ký

## 6.9 Xác minh chữ ký (/crypto/verify)

Mô tả kiểm thử	Input	Kết quả mong đợi
Xác minh đúng chữ ký	File gốc + file <code>.sig</code> đúng	Hiển thị email người ký, ngày ký, thông báo hợp lệ
Chữ ký sai hoặc không khớp	File bị thay đổi hoặc <code>.sig</code> giả mạo	Báo lỗi "Invalid signature" hoặc "Verification failed"
Không có public key người ký	Người ký chưa có key trong danh sách	Báo lỗi "Không tìm thấy public key phù hợp"

## 6.10 Phân quyền tài khoản

Mô tả kiểm thử	Input	Kết quả mong đợi
Admin truy cập trang quản lý	Người dùng có role = admin	Hiển thị danh sách tài khoản, log, nút khóa/mở tài khoản
Người thường truy cập trang admin	role = user	Báo lỗi "Access denied" và chuyển hướng về login
Khoá / mở tài khoản người dùng	Bấm nút khóa/mở trong giao diện admin	Cập nhật trạng thái khóa, lưu log thao tác

## 6.11 Ghi log bảo mật (security.log hoặc bảng log\_activity)

Mô tả kiểm thử	Input	Kết quả mong đợi
Ghi log đăng nhập thành công	Email + passphrase đúng	Log sự kiện "Login success" với trạng thái "Pending MFA"
Ghi log đăng nhập sai	Nhập sai passphrase	Ghi vào log với thông tin thất bại, timestamp, email
Ghi log ký số / cập nhật / mã hoá	Thao tác thành công các chức năng	Ghi đúng hành vi người dùng vào log theo chuẩn đã định

## 6.12 Chia nhỏ tập tin lớn khi mã hóa (>5MB)

Mô tả kiểm thử	Input	Kết quả mong đợi
File >5MB được chia nhỏ	Upload file >5MB	Hệ thống chia thành block 1MB, mã hóa từng block bằng AES-GCM
File nhỏ <5MB không chia	Upload file 2MB	Hệ thống mã hóa nguyên khối, không chia block
Kiểm tra toàn vẹn sau khi gộp lại	Giải mã file đã chia	Nội dung khôi phục đúng, không mất dữ liệu

## 6.13 Kiểm tra trạng thái khóa (/manage\_keys)

Mô tả kiểm thử	Input	Kết quả mong đợi
Khóa còn hạn	Ngày tạo < 60 ngày	Hiển thị trạng thái “Còn hạn”
Khóa gần hết hạn	Ngày tạo 80–89 ngày	Hiển thị “Gần hết hạn”
Khóa hết hạn	Ngày tạo > 90 ngày	Hiển thị “Hết hạn”, cho phép gia hạn hoặc tạo mới

## 6.14 Tìm kiếm public key (/keys/search)

Mô tả kiểm thử	Input	Kết quả mong đợi
Tìm thấy public key	Nhập email có public key trong DB	Hiển thị: email, QR code, ngày tạo, hạn dùng
Không tìm thấy public key	Email không tồn tại trong DB	Thông báo “Không tìm thấy khóa công khai”

## 6.15 Giới hạn số lần đăng nhập (/login)

Mô tả kiểm thử	Input	Kết quả mong đợi
Sai 5 lần liên tiếp	Nhập sai passphrase 5 lần trong 2 phút	Khoá tài khoản trong 5 phút, hiển thị thông báo thời gian chờ
Sau 5 phút thử lại	Đợi hết thời gian khoá	Tài khoản được mở lại và đăng nhập thành công nếu đúng pass
Log mỗi lần sai	Nhập sai liên tiếp	Ghi log với timestamp, lý do và email liên quan

## 6.16 Tùy chọn định dạng lưu file (/crypto/encrypt)

Mô tả kiểm thử	Input	Kết quả mong đợi
Lưu dạng gộp file	Chọn tùy chọn "Gộp .enc" khi mã hoá	Sinh 1 file duy nhất chứa toàn bộ dữ liệu và metadata
Lưu dạng tách file	Chọn "Tách .enc và .key" khi mã hoá	Sinh 2 file riêng biệt: dữ liệu mã hoá và AES key
Tự động nhận dạng khi giải mã	Upload file (gộp hoặc tách) khi giải mã	Tự động phân tích định dạng và xử lý phù hợp

## 6.17 Khôi phục tài khoản (/recover\_account, /verify\_recovery)

Mô tả kiểm thử	Input	Kết quả mong đợi
Khôi phục thành công	Nhập đúng recovery code hiển thị sau đăng ký	Cho phép đổi passphrase mới
Mã khôi phục sai hoặc hết hiệu lực	Mã recovery không đúng hoặc đã dùng	Báo lỗi "Invalid recovery code"
Đổi pass thành công sau xác minh	Pass mới đủ mạnh	Lưu pass mới, cập nhật salt mới

## Tài liệu

- [1] Lanchec, S. (2024, June 5). User Authentication: a guide for developers. Forest Admin Blog. <https://www.forestadmin.com/blog/user-authentication-a-guide-for-developers/>
- [2] The five different types of authentication — WorkOS. (n.d.). WorkOS. <https://workos.com/blog/the-five-different-types-of-authentication>
- [3] What is Authentication? Definition and uses - Auth0. (n.d.). Auth0. <https://auth0.com/intro-to-iam/what-is-authentication>
- [4] GeeksforGeeks. (2024, June 6). What is User Authentication, and Why is it Important? GeeksforGeeks. <https://www.geeksforgeeks.org/what-is-user-authentication-and-why-is-it-important/>
- [5] Frontegg. (2025, January 9). Authentication: What It is and How It Works | Frontegg. Frontegg. [https://frontegg.com/blog/authentication#API\\_Authentication\\_Methods](https://frontegg.com/blog/authentication#API_Authentication_Methods)
- [6] Sakshyam Shah. (2022, February 25). 6 Authentication best practices. Teleport. <https://goteleport.com/blog/authentication-best-practices/>
- [7] Intel® AMT SDK implementation and Reference Guide. (n.d.). [https://software.intel.com/sites/manageability/AMT\\_Implementation\\_and\\_Reference\\_Guide/default.htm?url=WordDocuments%2Fintroductiontokerberosauthentication.htm](https://software.intel.com/sites/manageability/AMT_Implementation_and_Reference_Guide/default.htm?url=WordDocuments%2Fintroductiontokerberosauthentication.htm)
- [8] Author, V., & Author, V. (2024, October 8). What is Kerberos? Kerberos Authentication Explained. VAADATA - Ethical Hacking Services. <https://www.vaadata.com/blog/what-is-kerberos-kerberos-authentication-explained/>
- [9] Wikipedia contributors. (2025, February 8). Kerberos (protocol). Wikipedia. [https://en.wikipedia.org/wiki/Kerberos\\_\(protocol\)](https://en.wikipedia.org/wiki/Kerberos_(protocol))