ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
**TRƯỜNG ĐẠI HỌC BÁCH KHOA**
**KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH**


# ĐỒ ÁN TỔNG HỢP
# CÔNG NGHỆ PHẦN MỀM

# HỆ THỐNG FILE SHARING

**Nhóm: Backend Web**
**Giảng viên hướng dẫn: Lê Đình Thuận**


| **Họ và tên** | **MSSV** |
|---|---|
| Lê Thanh Huy | 2311159 |
| Nguyễn Đình Khôi | 2311681 |
| Đậu Minh Khôi | 2311659 |
| Cao Vũ Hoàng Long | 2311888 |
| Nguyễn Hoàng Long | 2311906 |
| Đặng Hải Sơn | 2312955 |


**TP. Hồ Chí Minh - 2025**

# PHÂN CÔNG NHIỆM VỤ

| STT | Họ và tên | MSSV | Nhóm | Công việc chính |
|---|---|---|---|---|
| 1 | Lê Thanh Huy | 2311159 | A | Database, API (Admin, File management) |
| 2 | Nguyễn Đình Khôi | 2311681 | A | Database, API (Admin, File management) |
| 3 | Đậu Minh Khôi | 2311659 | A | Database, API, Class Diagram |
| 4 | Cao Vũ Hoàng Long | 2311888 | B | API (Authentication), CI/CD |
| 5 | Nguyễn Hoàng Long | 2311906 | B | API (Authentication), CI/CD |
| 6 | Đặng Hải Sơn | 2312955 | B | API (Authentication), JWT, Usecase Diagram |

# MỤC LỤC

# 1. Tổng quan dự án

## 1.1. Mô tả dự án

Hệ thống **File Sharing and Management System** là một nền tảng web hỗ trợ người dùng tải lên, quản lý và chia sẻ tệp tin một cách an toàn, tiện lợi và linh hoạt.

Hệ thống được xây dựng theo kiến trúc dịch vụ REST với backend phát triển bằng Golang, giao tiếp thông qua API chuẩn hoá, và frontend dưới dạng ứng dụng web (SPA) do nhóm frontend triển khai.

Các đặc điểm chính của hệ thống bao gồm:

- Hệ thống cho phép người dùng tải tệp dưới hai hình thức: ẩn danh hoặc đăng nhập.
- Mỗi tệp được gán một `share token` duy nhất giúp người nhận có thể truy cập nhanh, đồng thời hỗ trợ các cơ chế bảo vệ như mật khẩu, whitelist theo email và giới hạn thời gian hiệu lực.
- Thông tin tệp (metadata), lịch sử truy cập và các chính sách hệ thống được quản lý tập trung trong cơ sở dữ liệu.
- Hệ thống hỗ trợ cơ chế dọn dẹp tệp tự động (cleanup) thông qua tác vụ cron với mã xác thực `X-Cron-Secret`.
- Tính năng thống kê lượt tải và xem lịch sử download hỗ trợ quá trình theo dõi và kiểm toán.

Mục tiêu tổng thể của dự án là xây dựng một nền tảng lưu trữ tệp hiệu quả, mở rộng tốt, đảm bảo bảo mật và dễ tích hợp với các dịch vụ phụ trợ như email service hoặc cloud storage trong tương lai.

## 1.2. Yêu cầu chức năng và phi chức năng

### 1.2.1. Yêu cầu chức năng

**FR1: Quản lý người dùng**

- **FR1.1:** Hệ thống cho phép người dùng đăng ký và đăng nhập thông qua email và mật khẩu; xác thực bằng JWT.
- **FR1.2:** Người dùng có thể kích hoạt và xác minh TOTP (Time-based One-Time Password) để tăng cường bảo mật cho tài khoản (2FA). Sau khi bật

TOTP, các lần đăng nhập tiếp theo sẽ yêu cầu mã 6 chữ số từ ứng dụng authenticator.

- **FR1.3:** Người dùng ẩn danh được phép tải lên tệp nhưng chỉ được upload file public (`isPublic=true`); không thể chỉnh sửa, quản lý hoặc xóa tệp sau đó.
- **FR1.4:** Người dùng đăng nhập có thể truy cập danh sách tệp cá nhân qua endpoint `/files/my`, xem chi tiết metadata qua `/files/{id}`, thay đổi bảo mật (password, whitelist), hoặc xóa tệp.
- **FR1.5:** Quản trị viên có thể truy cập các API quản lý hệ thống, bao gồm cập nhật chính sách (`/admin/policy`) và thực thi tác vụ cleanup (`/admin/cleanup`).

**FR2: Upload và chia sẻ tệp**

- **FR2.1:** Hệ thống hỗ trợ tải tệp bằng phương thức multipart thông qua endpoint `/files/upload`.
- **FR2.2:** Sau khi tải thành công, hệ thống sinh ra `share token` duy nhất và liên kết chia sẻ cho phép tải xuống qua endpoint `/files/{shareToken}/download`.
- **FR2.3:** Người dùng có thể thiết lập các thuộc tính:
  - Khoảng thời gian hiệu lực `availableFrom`, `availableTo` (validate theo system policy: $FROM < TO$, $TO$ không nằm trong quá khứ, tổng thời gian không vượt `maxValidityDays`).
  - Danh sách email được phép tải (whitelist `sharedWith`) - yêu cầu authenticated upload.
  - Mật khẩu bảo vệ (tối thiểu 8 ký tự) - yêu cầu authenticated upload.
  - Trạng thái công khai hoặc riêng tư (`isPublic`); anonymous upload luôn bị ép bằng true.
- **FR2.4:** Các tệp hết hạn sẽ không còn khả dụng và trả về mã lỗi HTTP 410. Tệp chưa đến thời gian hiệu lực (pending) trả về mã 423, trừ khi requester là owner (có thể preview để kiểm thử).

**FR3: Bảo mật và kiểm soát truy cập**

- **FR3.1:** Mật khẩu tệp được băm bằng bcrypt; mật khẩu user cũng được hash bằng bcrypt. Secret TOTP của user được lưu trữ an toàn trong cơ sở dữ liệu.
- **FR3.2:** Quy trình kiểm tra truy cập download bao gồm theo thứ tự: (1) trạng thái tệp (expired $\rightarrow$ 410, pending $\rightarrow$ 423), (2) kiểm tra whitelist (nếu

có `sharedWith` → yêu cầu JWT và email nằm trong danh sách), (3) yêu cầu mật khẩu (nếu file có password → phải gửi query parameter password).

- **FR3.3:** Người dùng ẩn danh không thể chỉnh sửa thông tin tệp đã tải lên; chỉ người dùng đăng nhập (owner) mới có quyền quản lý file của mình.
- **FR3.4:** Endpoint `/admin/cleanup` yêu cầu token admin hoặc header `X-Cron-Secret` (với cơ chế rotation định kỳ, rate limiting và logging đầy đủ).

**FR4: Quản lý vòng đời tệp**

- **FR4.1:** Metadata tệp được lưu trong cơ sở dữ liệu bao gồm thuộc tính (`fileName`, `fileSize`, `mimeType`), thời hạn (`availableFrom/To`, status: pending/active/expired), thông tin bảo mật (password hash, whitelist) và trạng thái.
- **FR4.2:** Người dùng có thể xem thống kê tệp qua `/files/{id}/stats`, bao gồm tổng lượt tải (`downloadCount`), số người download khác nhau (`uniqueDownloaders`) và thời điểm tải gần nhất (`lastDownloadedAt`). Anonymous uploads không có statistics.
- **FR4.3:** Hệ thống lưu lịch sử tải xuống chi tiết qua `/files/{id}/download-history` để phục vụ mục đích kiểm toán. Với anonymous download, chỉ ghi nhận "Anonymous" + timestamp + trạng thái, không lưu IP/User-Agent để đảm bảo privacy.
- **FR4.4:** Tệp có thể bị xóa bởi chủ sở hữu qua `DELETE /files/{id}` hoặc bị dọn dẹp tự động khi hết hạn thông qua `/admin/cleanup`.

### 1.2.2. Yêu cầu phi chức năng

**Hiệu năng**

- **NFR1:** Thời gian phản hồi đăng nhập không vượt quá 5 giây.
- **NFR2:** Mã TOTP phải được sinh và trả về (qua QR code) trong vòng 30 giây (tối đa 60 giây khi tải cao).
- **NFR3:** Thời gian tải lên tệp dưới 100 MB không vượt quá 30 giây.
- **NFR4:** Việc tải xuống hỗ trợ streaming HTTP đảm bảo ổn định với tệp dung lượng lớn, không chiếm nhiều RAM trên server.

**Bảo mật**

- **NFR5:** Toàn bộ giao tiếp giữa FE và BE sử dụng HTTPS (TLS 1.2 trở lên).
- **NFR6:** JWT có thời hạn và hỗ trợ refresh token (hoặc yêu cầu đăng nhập lại khi token hết hạn).
- **NFR7:** Share token được sinh ngẫu nhiên, chống dò quét, không thể đoán trước.
- **NFR8:** Mật khẩu user và file được hash bằng bcrypt; secret TOTP của user được lưu trữ an toàn; thông tin nhạy cảm không được log hoặc expose trong response.

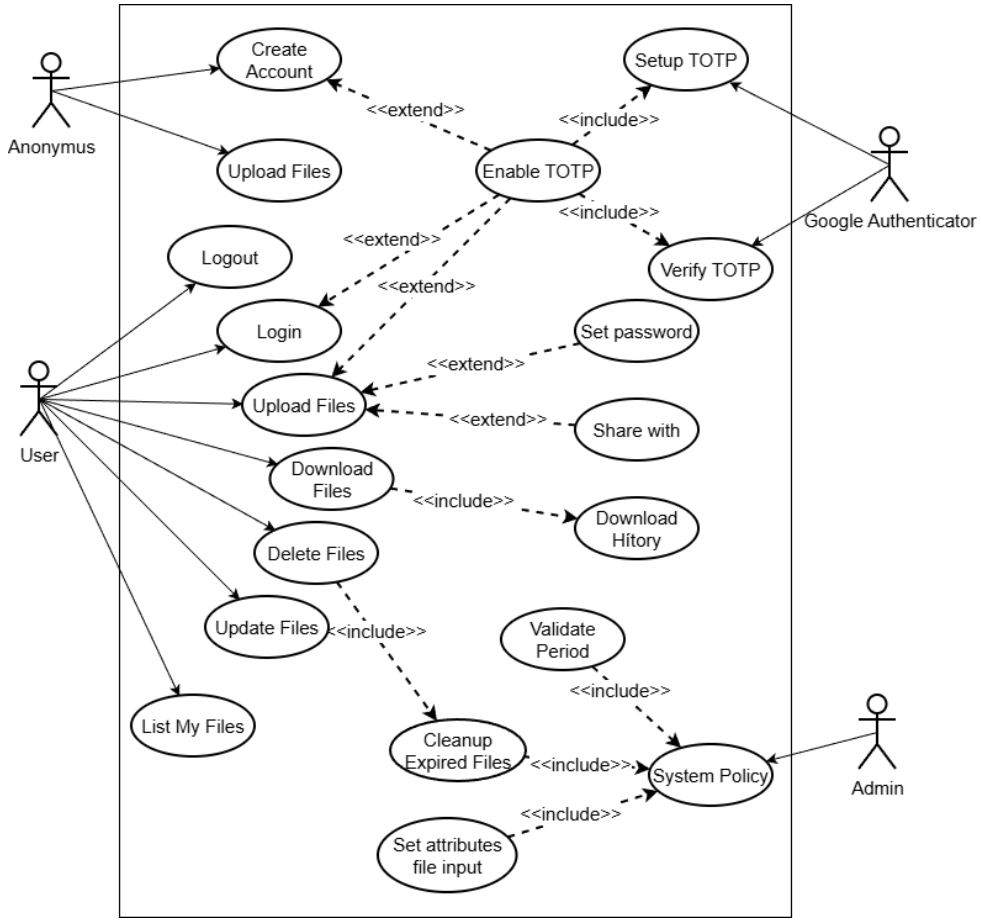
**Khả năng mở rộng và bảo trì**

- **NFR9:** Backend được thiết kế theo mô-đun, dễ mở rộng theo chức năng (ví dụ: tích hợp cloud storage, email service).
- **NFR10:** CSDL hỗ trợ đánh index và phân trang cho dữ liệu lớn (pagination với `page` và `limit`).
- **NFR11:** Sẵn sàng tích hợp cloud storage (S3, GCS) và email service (SMTP, provider bên thứ ba) trong tương lai.

**Chịu lỗi và UX**

- **NFR12:** Metadata và dữ liệu tệp (binary) được tách biệt, tránh mất dữ liệu metadata khi lỗi lưu trữ file binary.
- **NFR13:** Các thông báo lỗi tuân theo chuẩn JSON trong tài liệu OpenAPI (schema Error với `error`, `message`, `code`).
- **NFR14:** Dashboard hiển thị danh sách tệp qua `/files/my` với pagination, filter theo status, và summary statistics.
- **NFR15:** Hệ thống hỗ trợ owner preview file trong giai đoạn pending (bypass 423) để chủ file có thể kiểm thử link trước khi phát hành.
- **NFR17:** Hệ thống tự động xoá các tệp upload ẩn danh hoặc tệp tạm sau thời gian tồn tại vượt quá $x$ giờ.

## 1.3. Mô hình Use Case

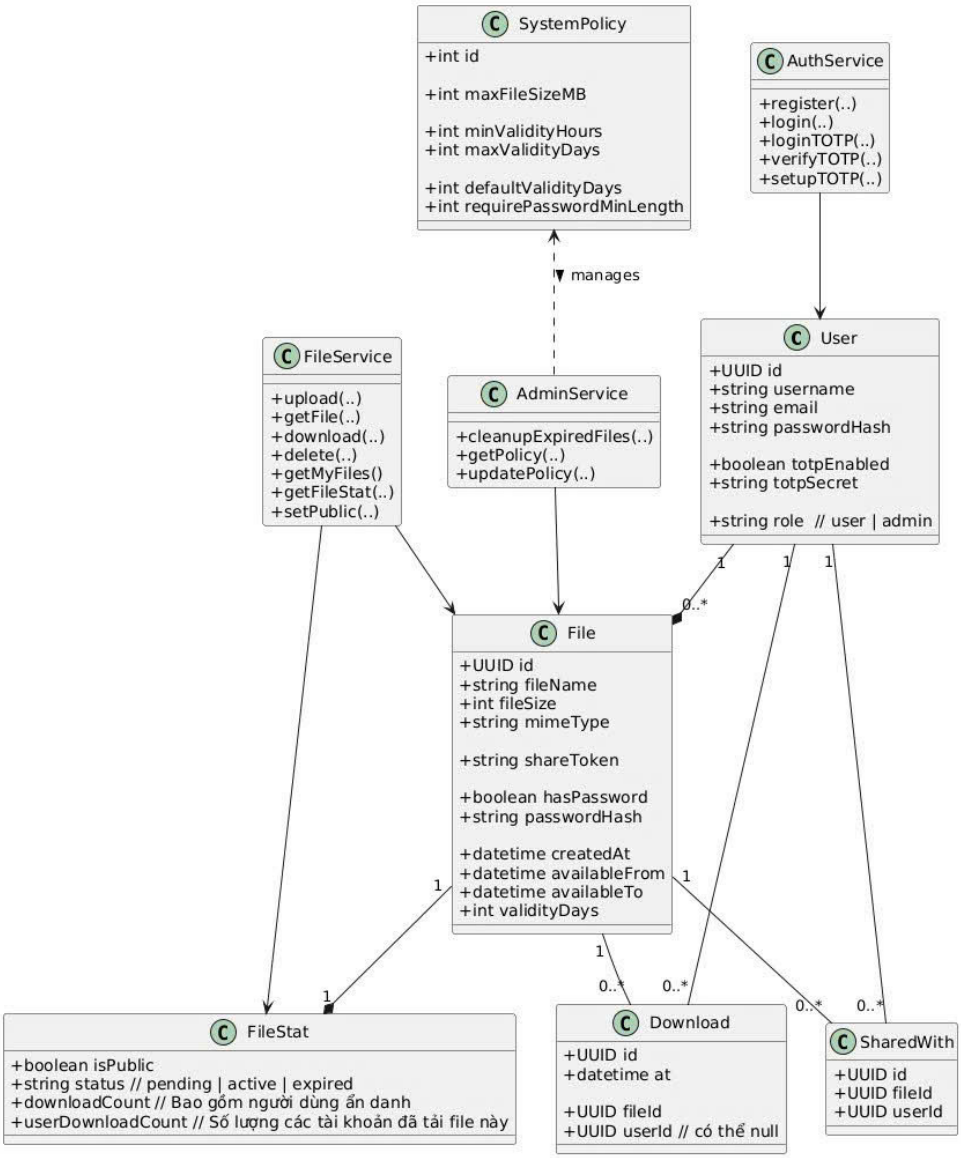



Hình 1.1: Sơ đồ Use Case của hệ thống chia sẻ file

Bảng 1.1: Bảng mô tả Use Case của hệ thống chia sẻ file

| **No.** | **Use-case** | **Job Description** |
|---|---|---|
| 1 | Create Account | Dùng để người dùng tạo tài khoản mới trong hệ thống. |
| 2 | Login | Cho phép người dùng đăng nhập để truy cập các chức năng của hệ thống. |
| 3 | Logout | Đăng xuất khỏi hệ thống, kết thúc phiên làm việc. |
| 4 | Enable TOTP | Bật tính năng xác thực hai lớp (2FA) cho tài khoản người dùng. |
| 5 | Setup TOTP | Sinh mã QR và secret key để người dùng đăng ký Google Authenticator. |
| 6 | Verify TOTP | Kiểm tra mã xác thực sáu chữ số được tạo từ ứng dụng xác thực. |
| 7 | Upload files | Cho phép người dùng hoặc khách ẩn danh tải file lên hệ thống và tạo link chia sẻ. |
| 8 | Set password | Thiết lập mật khẩu bảo vệ file để chỉ người có mật khẩu mới tải được. |
| 9 | Share with | Chia sẻ file cho danh sách người dùng cụ thể qua email. |
| 10 | Download files | Tải file về nếu thỏa mãn điều kiện truy cập. |
| 11 | Download History | Ghi lại lịch sử tải file. |
| 12 | Delete files | Cho phép chủ sở hữu file xóa file khỏi hệ thống. |
| 13 | Update files | Cập nhật thông tin hoặc thuộc tính của file. |
| 14 | List my files | Hiển thị danh sách file của người dùng. |
| 15 | Validate Period | Kiểm tra thời gian hiệu lực file. |
| 16 | Set attribute file input | Thiết lập thuộc tính file. |
| 17 | System Policy | Xem và quản lý cấu hình hệ thống. |
| 18 | Cleanup Expired Files | Xóa các file đã hết hạn. |

## 1.4. Mô hình Class




Hình 1.2: Sơ đồ Class của hệ thống chia sẻ file

Bảng 1.2: Bảng mô tả về Class diagram

| No. | Class | Giải thích |
|---|---|---|
| 1 | User | Lưu trữ thông tin về tài khoản người dùng. |
| 2 | File | Lưu các metadata về các file được tải lên. |
| 3 | FileStat | Chứa các thông tin về tình trạng và thống kê của file. |
| 4 | SharedWith | Cho biết file này được chia sẻ với các người dùng nào. |
| 5 | Download | Ghi nhận những lượt tải về (bao gồm tải về ẩn danh). |
| 6 | Services | Các interface (FileService, AdminService, AuthService) để quản lý, tương tác với File, User, FileStat. |
| 7 | System Policy | Các ràng buộc về tải lên và chia sẻ file. |

# 2. Đặc tả API (API Specification)

Chương này mô tả chi tiết các API endpoints của hệ thống chia sẻ file. API được thiết kế theo chuẩn RESTful và tuân thủ OpenAPI Specification 3.0.4.

***Ghi chú:*** *Để xem đầy đủ chi tiết về tất cả các API endpoints, request/response schemas và ví dụ, vui lòng tham khảo file* `docs/openapi.yaml` *trong mã nguồn dự án.*

## 2.1. Tổng quan API

### 2.1.1. Thông tin chung

Bảng 2.1: Thông tin chung về API

| **Thuộc tính** | **Giá trị** |
|---|---|
| Tên API | File Sharing System API |
| Phiên bản | 1.0.0 |
| Base URL (Development) | `http://localhost:8080` |
| Định dạng dữ liệu | JSON |
| Xác thực | Bearer Token (JWT) |

### 2.1.2. Phân nhóm API

API được phân thành 3 nhóm chính:

Bảng 2.2: Phân nhóm các API endpoints

| **Nhóm** | **Mô tả** | **Số endpoints** |
|---|---|---|
| Authentication | Đăng ký, đăng nhập, xác thực 2FA, đăng xuất | 7 |
| Files | Upload, download, preview, quản lý file | 10 |
| Admin | Quản lý hệ thống, cleanup, policy | 3 |

### 2.1.3. Xác thực và phân quyền

Hệ thống sử dụng JWT (JSON Web Token) cho xác thực:

- **Bearer Token**: Gửi trong header `Authorization: Bearer <token>`
- **Optional Auth**: Một số endpoint cho phép anonymous access (upload public, download public file)
- **Admin Token**: Các endpoint admin yêu cầu token đặc biệt từ biến môi trường `ADMIN_API_TOKEN`

## 2.2. Authentication APIs

### 2.2.1. Đăng ký tài khoản

Bảng 2.3: API Đăng ký tài khoản

| | |
|---|---|
| **Endpoint** | `POST /auth/register` |
| **Mô tả** | Tạo tài khoản người dùng mới |
| **Xác thực** | Không yêu cầu |
| **Content-Type** | `application/json` |

**Request Body:**

Listing 2.1: Request đăng ký tài khoản

```
{
  "username": "nam123",
  "email": "nam@example.com",
  "password": "passwordtest"
}
```

**Response thành công (200):**

Listing 2.2: Response đăng ký thành công

```
{
  "message": "User registered successfully",
  "userId": "550e8400-e29b-41d4-a716-446655440000"
}
```

**Các mã lỗi:**

- **400 Bad Request**: Thiếu hoặc sai định dạng trường bắt buộc
- **409 Conflict**: Email hoặc username đã tồn tại

### 2.2.2. Đăng nhập

Bảng 2.4: API Đăng nhập

| | |
|---|---|
| **Endpoint** | `POST /auth/login` |
| **Mô tả** | Đăng nhập để lấy JWT token |
| **Xác thực** | Không yêu cầu |
| **Content-Type** | `application/json` |

**Request Body:**

Listing 2.3: Request đăng nhập

```
{
  "email": "nam@example.com",
  "password": "passwordtest"
}
```

**Response thành công - Không có 2FA (200):**

Listing 2.4: Response đăng nhập thành công

```
{
  "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "user": {
    "id": "550e8400-e29b-41d4-a716-446655440000",
    "username": "nam123",
    "email": "nam@example.com"
  }
}
```

**Response yêu cầu TOTP - Có 2FA (200):**

Listing 2.5: Response yêu cầu xác thực TOTP

```
{
  "requireTOTP": true,
  "cid": "8d4f3bb1-2f52-4a76-b951-7c21ef991abc",
  "message": "TOTP verification required"
}
```

**Các mã lỗi:**

- **401 Unauthorized**: Sai email hoặc password

### 2.2.3. Xác thực TOTP

Bảng 2.5: API Xác thực TOTP

| | |
|---|---|
| **Endpoint** | `POST /auth/login/totp` |
| **Mô tả** | Hoàn tất đăng nhập với mã TOTP 6 chữ số |
| **Xác thực** | Không yêu cầu (sử dụng CID từ bước trước) |
| **Content-Type** | `application/json` |

**Request Body:**

Listing 2.6: Request xác thực TOTP

```
{
  "cid": "8d4f3bb1-2f52-4a76-b951-7c21ef991abc",
  "code": "123456"
}
```

**Response thành công (200):**

Listing 2.7: Response xác thực TOTP thành công

```
{
  "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "user": {
    "id": "550e8400-e29b-41d4-a716-446655440000",
    "username": "nam123",
    "email": "nam@example.com"
  }
}
```

**Các mã lỗi:**

- **401 Unauthorized**: Mã TOTP không hợp lệ hoặc CID đã hết hạn

### 2.2.4. Thiết lập TOTP

Bảng 2.6: API Thiết lập TOTP

| **Endpoint** | `POST /auth/totp/setup` |
|---|---|
| **Mô tả** | Tạo TOTP secret và QR code để thiết lập 2FA |
| **Xác thực** | Bearer Token (bắt buộc) |

**Response thành công (200):**

Listing 2.8: Response thiết lập TOTP

```
{
  "message": "TOTP secret generated",
  "totpSetup": {
    "secret": "NB2W45DFOIZA====",
    "qrCode": "..."
  }
}
```

### 2.2.5. Xác minh và kích hoạt TOTP

Bảng 2.7: API Xác minh TOTP

| | |
|---|---|
| **Endpoint** | `POST /auth/totp/verify` |
| **Mô tả** | Xác minh mã TOTP để kích hoạt 2FA cho tài khoản |
| **Xác thực** | Bearer Token (bắt buộc) |
| **Content-Type** | `application/json` |

#### Request Body:

Listing 2.9: Request xác minh TOTP

```
{
  "code": "123456"
}
```

#### Response thành công (200):

Listing 2.10: Response kích hoạt TOTP thành công

```
{
  "message": "TOTP verified successfully",
  "totpEnabled": true
}
```

### 2.2.6. Đăng xuất

Bảng 2.8: API Đăng xuất

| | |
|---|---|
| **Endpoint** | `POST /auth/logout` |
| **Mô tả** | Đăng xuất và vô hiệu hóa token |
| **Xác thực** | Bearer Token (bắt buộc) |

#### Response thành công (200):

Listing 2.11: Response đăng xuất thành công

```
{
  "message": "User logged out"
}
```

### 2.2.7. Lấy thông tin người dùng

Bảng 2.9: API Lấy thông tin người dùng

| | |
|---|---|
| **Endpoint** | `GET /user` |
| **Mô tả** | Lấy thông tin profile của user hiện tại |
| **Xác thực** | Bearer Token (bắt buộc) |

#### Response thành công (200):

Listing 2.12: Response thông tin người dùng

```
{
  "user": {
    "id": "550e8400-e29b-41d4-a716-446655440000",
    "username": "nam123",
    "email": "nam@example.com",
    "role": "user",
    "totpEnabled": true
  }
}
```

## 2.3. File Management APIs

### 2.3.1. Upload file

Bảng 2.10: API Upload file

| | |
|---|---|
| **Endpoint** | `POST /files/upload` |
| **Mô tả** | Upload file mới và tạo share link |
| **Xác thực** | Optional (anonymous cho public files) |
| **Content-Type** | `multipart/form-data` |

#### Request Parameters:

Bảng 2.11: Các tham số upload file

| Tham số | Kiểu | Bắt buộc | Mô tả |
|---|---|---|---|
| file | Binary | Có | File cần upload |
| isPublic | Boolean | Không | true = public, false = private |
| password | String | Không | Mật khẩu bảo vệ (tối thiểu 8 ký tự) |
| availableFrom | DateTime | Không | Thời điểm bắt đầu hiệu lực |
| availableTo | DateTime | Không | Thời điểm hết hạn |
| sharedWith | Array[String] | Không | Danh sách email được chia sẻ |

#### Response thành công (201):

Listing 2.13: Response upload thành công

```
{
  "success": true,
  "message": "File uploaded successfully",
  "file": {
    "id": "550e8400-e29b-41d4-a716-446655440000",
    "fileName": "report.pdf",
    "shareToken": "a1b2c3d4e5f6g7h8",
    "isPublic": false
  }
```

```
}
```

**Các mã lỗi:**

- **400 Bad Request**: Thiếu file, password yếu, hoặc cấu hình không hợp lệ
- **401 Unauthorized**: Private upload yêu cầu authentication
- **413 Payload Too Large**: File vượt quá giới hạn kích thước

### 2.3.2. Lấy danh sách file của tôi

Bảng 2.12: API Lấy danh sách file

| **Endpoint** | `GET /files/my` |
|---|---|
| **Mô tả** | Lấy danh sách file do user hiện tại upload |
| **Xác thực** | Bearer Token (bắt buộc) |

**Query Parameters:**

Bảng 2.13: Các tham số query cho danh sách file

| **Tham số** | **Kiểu** | **Mặc định** | **Mô tả** |
|---|---|---|---|
| status | String | all | Lọc: active, expired, pending, all |
| page | Integer | 1 | Số trang |
| limit | Integer | 20 | Số file mỗi trang (max 100) |
| sortBy | String | createdAt | Sắp xếp: createdAt, fileName |
| order | String | desc | Thứ tự: asc, desc |

**Response thành công (200):**

Listing 2.14: Response danh sách file

```
{
  "files": [
    {
      "id": "550e8400-e29b-41d4-a716-446655440001",
      "fileName": "document.pdf",
      "shareToken": "a1b2c3d4e5f6g7h8",
      "status": "active",
      "createdAt": "2025-11-19T10:00:00Z"
    }
  ],
  "pagination": {
    "currentPage": 1,
    "totalPages": 3,
    "totalFiles": 42,
    "limit": 20
```

```
  },
  "summary": {
    "activeFiles": 28,
    "pendingFiles": 5,
    "expiredFiles": 9
  }
}
```

### 2.3.3. Lấy thông tin file theo ID

Bảng 2.14: API Lấy thông tin file theo ID

| | |
|---|---|
| **Endpoint** | `GET /files/info/{id}` |
| **Mô tả** | Lấy thông tin chi tiết file (chỉ owner hoặc admin) |
| **Xác thực** | Bearer Token (bắt buộc) |

**Response thành công (200):**

Listing 2.15: Response thông tin file chi tiết

```
{
  "file": {
    "id": "550e8400-e29b-41d4-a716-446655440000",
    "fileName": "contract.pdf",
    "fileSize": 2048576,
    "mimeType": "application/pdf",
    "shareToken": "a1b2c3d4e5f6g7h8",
    "isPublic": false,
    "hasPassword": true,
    "availableFrom": "2025-11-10T00:00:00Z",
    "availableTo": "2025-11-17T00:00:00Z",
    "status": "active",
    "hoursRemaining": 120.5,
    "sharedWith": ["user1@example.com", "user2@example.com"],
    "owner": {
      "id": "550e8400-e29b-41d4-a716-446655440001",
      "username": "nam123",
      "email": "nam@example.com"
    },
    "createdAt": "2025-11-10T10:00:00Z"
  }
}
```

### 2.3.4. Lấy thông tin file theo Share Token

Bảng 2.15: API Lấy thông tin file theo Share Token

| | |
|---|---|
| **Endpoint** | `GET /files/{shareToken}` |
| **Mô tả** | Lấy metadata cơ bản của file (public endpoint) |
| **Xác thực** | Không yêu cầu |

**Response thành công (200):**

Listing 2.16: Response thông tin file public

```
{
  "file": {
    "id": "550e8400-e29b-41d4-a716-446655440000",
    "fileName": "contract.pdf",
    "shareToken": "a1b2c3d4e5f6g7h8",
    "status": "active",
    "isPublic": false,
    "hasPassword": true,
    "fileSize": 2048576,
    "mimeType": "application/pdf"
  }
}
```

**Các mã lỗi:**

- **404 Not Found**: Share token không tồn tại
- **410 Gone**: File đã hết hạn
- **423 Locked**: File chưa đến thời gian hiệu lực

### 2.3.5. Tải file

Bảng 2.16: API Tải file

| | |
|---|---|
| **Endpoint** | `GET /files/{shareToken}/download` |
| **Mô tả** | Download file (có thể yêu cầu password) |
| **Xác thực** | Optional (yêu cầu cho private files) |

**Query/Header Parameters:**

- `password` (query): Mật khẩu file (nếu có)
- `Authorization` (header): Bearer token (cho private files)

**Response thành công (200):**

Listing 2.17: Response headers khi download

```
Content-Type: application/pdf
Content-Length: 2048576
Content-Disposition: attachment; filename="report.pdf"

[Binary content]
```

**Các mã lỗi:**

- **401 Unauthorized**: Thiếu password hoặc sai password
- **403 Forbidden**: Không có quyền truy cập
- **404 Not Found**: File không tồn tại
- **410 Gone**: File đã hết hạn
- **423 Locked**: File chưa đến thời gian hiệu lực

### 2.3.6. Xem trước file

Bảng 2.17: API Xem trước file

| **Endpoint** | `GET /files/{shareToken}/preview` |
|---|---|
| **Mô tả** | Preview file trong browser (inline display) |
| **Xác thực** | Optional (yêu cầu cho private files) |

**Khác biệt so với Download:**

- Download: `Content-Disposition: attachment` → Tải file về
- Preview: `Content-Disposition: inline` → Hiển thị trong browser

### 2.3.7. Xóa file

Bảng 2.18: API Xóa file

| **Endpoint** | `DELETE /files/info/{id}` |
|---|---|
| **Mô tả** | Xóa file (chỉ owner hoặc admin) |
| **Xác thực** | Bearer Token (bắt buộc) |

**Response thành công (200):**

Listing 2.18: Response xóa file thành công

```
{
  "message": "File deleted successfully",
  "fileId": "550e8400-e29b-41d4-a716-446655440000"
}
```

### 2.3.8. Lấy thống kê download

Bảng 2.19: API Lấy thống kê download

| **Endpoint** | `GET /files/stats/{id}` |
|---|---|
| **Mô tả** | Lấy statistics của file (chỉ owner hoặc admin) |
| **Xác thực** | Bearer Token (bắt buộc) |

**Response thành công (200):**

Listing 2.19: Response thống kê download

```
{
  "fileId": "550e8400-e29b-41d4-a716-446655440000",
  "fileName": "presentation.pdf",
  "statistics": {
    "downloadCount": 45,
    "uniqueDownloaders": 12,
    "lastDownloadedAt": "2025-11-19T14:30:00Z",
    "createdAt": "2025-11-10T10:00:00Z"
  }
}
```

### 2.3.9. Lấy lịch sử download

Bảng 2.20: API Lấy lịch sử download

| **Endpoint** | `GET /files/download-history/{id}` |
|---|---|
| **Mô tả** | Lấy download history chi tiết (chỉ owner hoặc admin) |
| **Xác thực** | Bearer Token (bắt buộc) |

**Response thành công (200):**

Listing 2.20: Response lịch sử download

```
{
  "fileId": "550e8400-e29b-41d4-a716-446655440000",
  "fileName": "presentation.pdf",
  "history": [
    {
      "id": "650e8400-e29b-41d4-a716-446655440001",
      "downloader": {
        "username": "tranthib",
        "email": "tranthib@example.com"
      },
      "downloadedAt": "2025-11-19T14:30:00Z",
      "downloadCompleted": true
```

```
    },
    {
      "id": "650e8400-e29b-41d4-a716-446655440002",
      "downloader": null,
      "downloadedAt": "2025-11-19T10:15:00Z",
      "downloadCompleted": true
    }
  ],
  "pagination": {
    "currentPage": 1,
    "totalPages": 1,
    "totalRecords": 2,
    "limit": 50
  }
}
```


## 2.4. Admin APIs

### 2.4.1. Dọn dẹp file hết hạn

Bảng 2.21: API Dọn dẹp file hết hạn

| | |
|---|---|
| **Endpoint** | `POST /admin/cleanup` |
| **Mô tả** | Xóa các file đã hết hạn khỏi hệ thống |
| **Xác thực** | Admin Token hoặc X-Cron-Secret |

**Response thành công (200):**

Listing 2.21: Response cleanup thành công


```
{
  "message": "Expired files removed",
  "deletedFiles": 12,
  "timestamp": "2025-11-19T10:00:00Z"
}
```


### 2.4.2. Lấy cấu hình hệ thống

Bảng 2.22: API Lấy cấu hình hệ thống

| | |
|---|---|
| **Endpoint** | `GET /admin/policy` |
| **Mô tả** | Lấy system policy hiện tại |
| **Xác thực** | Admin Token (bắt buộc) |

**Response thành công (200):**

Listing 2.22: Response cấu hình hệ thống

```
{
  "id": 1,
  "maxFileSizeMB": 50,
  "minValidityHours": 1,
  "maxValidityDays": 30,
  "defaultValidityDays": 7,
  "requirePasswordMinLength": 8
}
```

### 2.4.3. Cập nhật cấu hình hệ thống

Bảng 2.23: API Cập nhật cấu hình hệ thống

| | |
|---|---|
| **Endpoint** | `PATCH /admin/policy` |
| **Mô tả** | Cập nhật system policy |
| **Xác thực** | Admin Token (bắt buộc) |
| **Content-Type** | `application/json` |

**Request Body:**

Listing 2.23: Request cập nhật policy

```
{
  "maxFileSizeMB": 100,
  "maxValidityDays": 14,
  "defaultValidityDays": 5
}
```

**Response thành công (200):**

Listing 2.24: Response cập nhật policy thành công

```
{
  "message": "Policy updated",
  "policy": {
    "id": 1,
    "maxFileSizeMB": 100,
    "minValidityHours": 1,
    "maxValidityDays": 14,
    "defaultValidityDays": 5,
    "requirePasswordMinLength": 8
  }
}
```

## 2.5. Mã lỗi HTTP phổ biến

Bảng 2.24: Bảng mã lỗi HTTP và ý nghĩa

| **Mã** | **Tên** | **Mô tả** |
|---|---|---|
| 200 | OK | Thành công |
| 201 | Created | Tạo mới thành công (upload file) |
| 400 | Bad Request | Dữ liệu đầu vào không hợp lệ |
| 401 | Unauthorized | Chưa xác thực hoặc sai thông tin |
| 403 | Forbidden | Không có quyền truy cập |
| 404 | Not Found | Không tìm thấy resource |
| 409 | Conflict | Xung đột dữ liệu (email đã tồn tại) |
| 410 | Gone | File đã hết hạn |
| 413 | Payload Too Large | File vượt quá giới hạn |
| 423 | Locked | File chưa đến thời gian hiệu lực |
| 429 | Too Many Requests | Vượt quá giới hạn tần suất |
| 500 | Internal Server Error | Lỗi server |

## 2.6. Cấu trúc Response lỗi

Tất cả các response lỗi đều có cấu trúc thống nhất:

Listing 2.25: Cấu trúc response lỗi

```
{
  "error": "Error Type",
  "message": "Chi tiet loi"
}
```

Một số response lỗi có thêm thông tin bổ sung:

Listing 2.26: Response lỗi file hết hạn

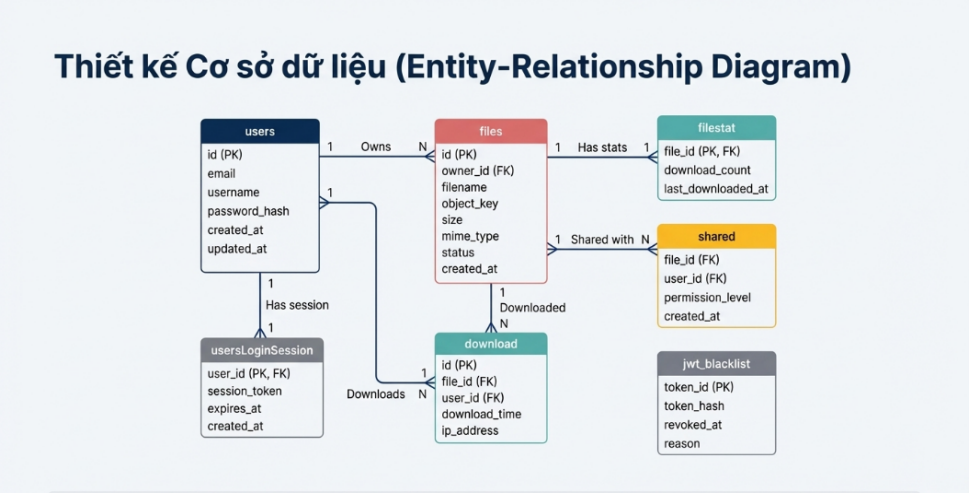
```
{
  "error": "File expired",
  "expiredAt": "2025-11-19T16:20:18.619Z"
}
```

Listing 2.27: Response lỗi file chưa đến hạn

```
{
  "error": "File not yet available",
  "availableFrom": "2025-11-20T10:00:00Z",
  "hoursUntilAvailable": 6
}
```

# 3. Phân tích và Thiết kế hệ thống

## 3.1. Thiết kế Cơ sở dữ liệu




Hình 3.1: Thiết kế cơ sơ dữ liệu (Entity-Relationship diagram

## 3.2. Kiến trúc Hệ thống


Hình 3.2: Kiến trúc tổng thể của hệ thống

# 4. Thiết kế các luồng xử lý chính

Phần này mô tả logic chi tiết của các chức năng nghiệp vụ quan trọng, đảm bảo tính bảo mật và tuân thủ các quy tắc nghiệp vụ. Mỗi luồng xử lý được phân tích kèm theo cấu trúc request/response cụ thể.

***Ghi chú:*** *Để xem chi tiết đầy đủ về tất cả các API endpoints, request/response schemas và ví dụ, vui lòng tham khảo file* `docs/openapi.yaml` *trong mã nguồn dự án.*

## 4.1. Quy trình Đăng nhập và Xác thực 2 lớp (Authentication Flow)

### 4.1.1. Mô tả tổng quan

Hệ thống hỗ trợ hai chế độ đăng nhập: đăng nhập thông thường (email + password) và đăng nhập với xác thực hai lớp (2FA) sử dụng TOTP (Time-based One-Time Password) theo chuẩn RFC 6238.

### 4.1.2. Luồng đăng nhập không có 2FA

#### 4.1.2.1. Bước 1: Client gửi yêu cầu đăng nhập

**Request:**

Listing 4.1: Request đăng nhập

```
POST /auth/login
Content-Type: application/json

{
  "email": "nam@example.com",
  "password": "passwordtest"
}
```

**Mô tả xử lý:**

- Server nhận request và thực hiện chuẩn hóa email: loại bỏ khoảng trắng thừa và chuyển về chữ thường.
- Truy vấn database để tìm user theo email.
- Nếu không tìm thấy user hoặc email không tồn tại trong hệ thống, server trả về lỗi xác thực.

#### 4.1.2.2. Bước 2: Xác thực mật khẩu

**Mô tả xử lý:**

- Sử dụng hàm `bcrypt.CompareHashAndPassword()` để so sánh password người dùng nhập với hash đã lưu trong database.
- Bcrypt được chọn vì tính chất one-way hash và khả năng chống brute-force với cost factor có thể điều chỉnh.
- Nếu password không khớp, server trả về lỗi xác thực giống như trường hợp email không tồn tại (để tránh tiết lộ thông tin user có tồn tại hay không).

**Response khi thất bại:**

Listing 4.2: Response lỗi xác thực

```
HTTP Status: 401 Unauthorized

{
  "error": "Unauthorized",
  "message": "Invalid email or password"
}
```

#### 4.1.2.3. Bước 3: Tạo Access Token và trả về

**Mô tả xử lý:**

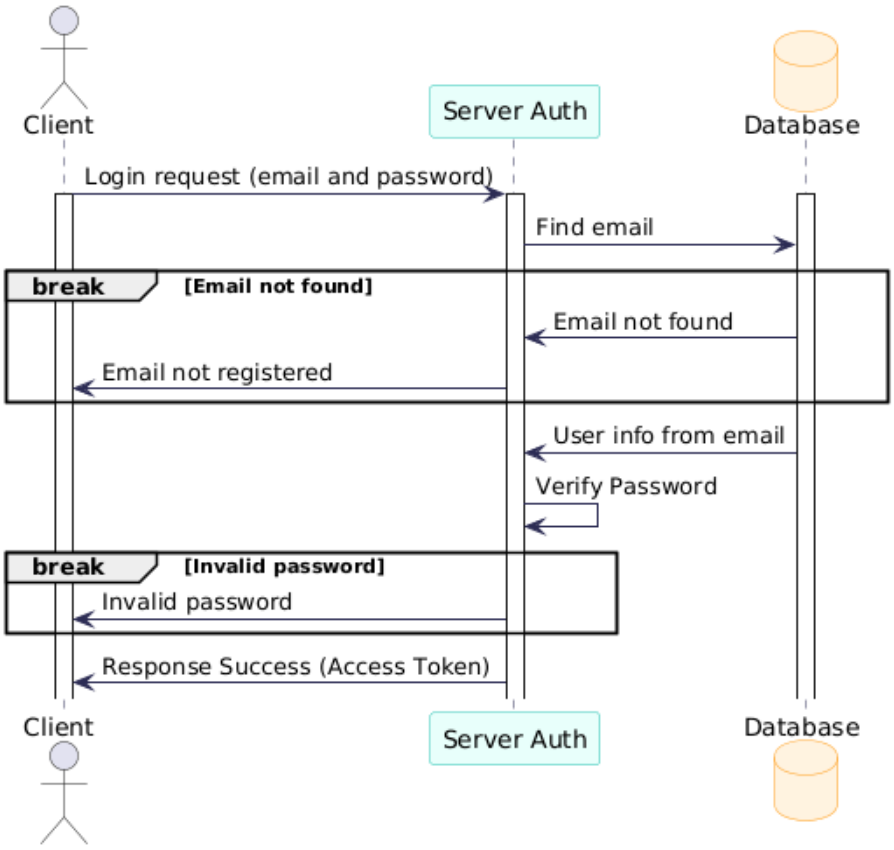
- Kiểm tra cờ `EnableTOTP` của user. Trong trường hợp này, `EnableTOTP = false`.
- Tạo JWT Access Token chứa các claims: `userId`, `email`, `role`, `exp` (thời điểm hết hạn).
- Token được ký bằng secret key của server, đảm bảo không thể giả mạo.

**Response thành công:**

Listing 4.3: Response đăng nhập thành công

```
HTTP Status: 200 OK

{
  "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "user": {
    "id": "550e8400-e29b-41d4-a716-446655440000",
    "username": "nam123",
    "email": "nam@example.com"
  }
}
```

**Sequence diagram:**




Hình 4.1: Sơ đồ trình tự đăng nhập không TOTP

### 4.1.3. Luồng đăng nhập có xác thực 2 lớp (2FA)

Khi người dùng đã kích hoạt TOTP, quy trình đăng nhập chia thành hai giai đoạn:

#### 4.1.3.1. Giai đoạn 1: Xác thực thông tin cơ bản

**Request:**

Listing 4.4: Request đăng nhập giai đoạn 1

```
POST /auth/login
Content-Type: application/json

{
  "email": "nam@example.com",
  "password": "passwordtest"
}
```

**Mô tả xử lý:**

- Server thực hiện xác thực email và password giống như luồng không có 2FA.
- Sau khi xác thực thành công, hệ thống phát hiện `EnableTOTP = true`.

- Thay vì tạo Access Token ngay, hệ thống tạo một Challenge ID (CID) sử dụng UUID version 1.
- UUID v1 được chọn vì nó chứa timestamp trong chính cấu trúc ID, cho phép kiểm tra thời gian hết hạn mà không cần lưu thêm trường thời gian vào database.
- CID được lưu vào bảng session tạm thời (`users_login_session`) cùng với userId.

**Response yêu cầu TOTP:**

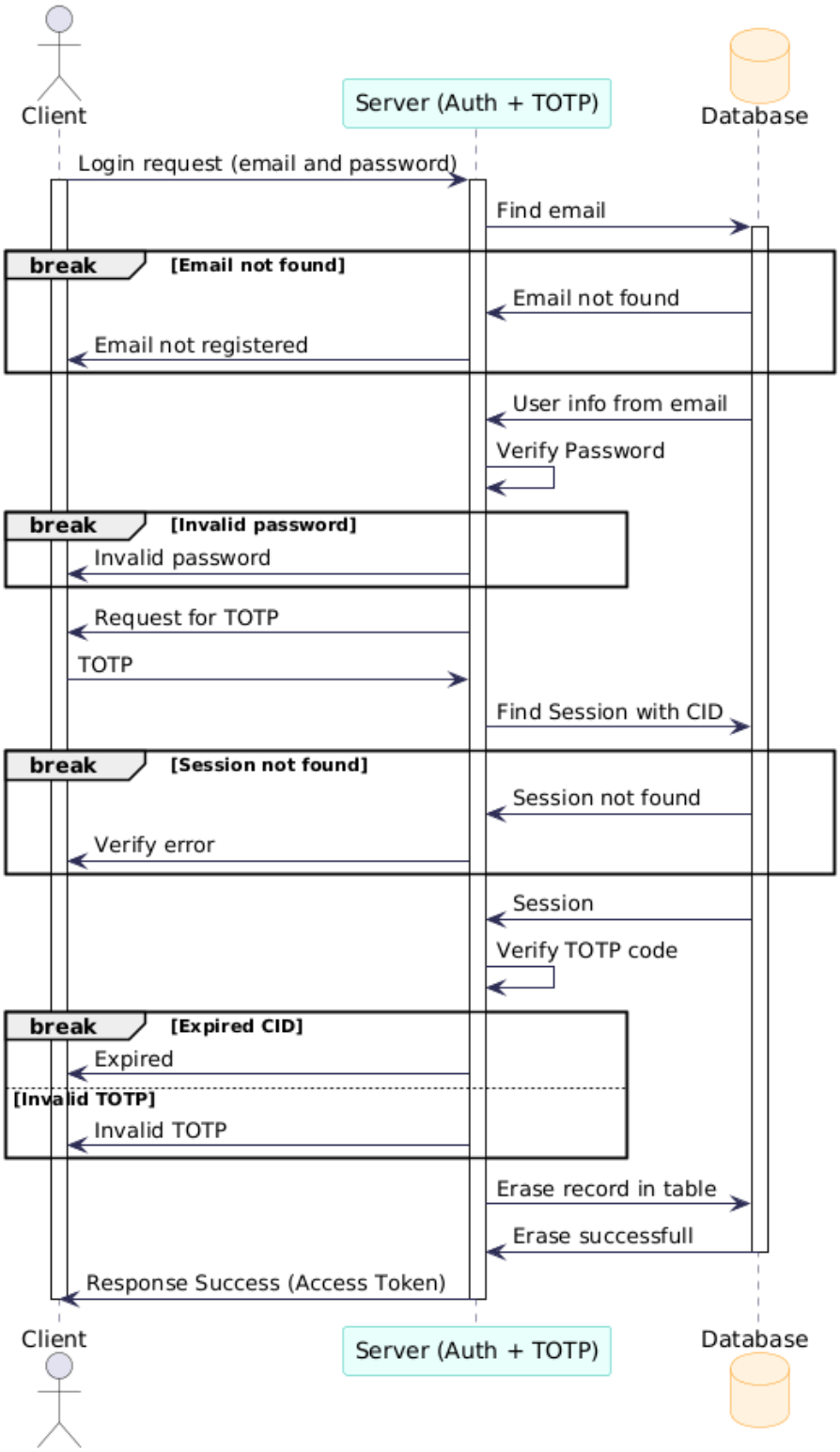
Listing 4.5: Response yêu cầu xác thực TOTP

```
HTTP Status: 200 OK

{
  "requireTOTP": true,
  "cid": "8d4f3bb1-2f52-4a76-b951-7c21ef991abc",
  "message": "TOTP verification required"
}
```

**Phân tích response:**

- `requireTOTP: true`: Báo cho client biết cần tiếp tục với bước xác thực TOTP.
- `cid`: Challenge ID để sử dụng trong bước tiếp theo. CID này có thời hạn 5 phút.
- Client lưu tạm CID và hiển thị form nhập mã TOTP cho user.

**Sequence diagram:**




Hình 4.2: Sơ đồ trình tự đăng nhập có TOTP

#### 4.1.3.2. Giai đoạn 2: Xác thực mã TOTP

**Request:**

Listing 4.6: Request xác thực TOTP

```
POST /auth/login/totp
Content-Type: application/json

{
  "cid": "8d4f3bb1-2f52-4a76-b951-7c21ef991abc",
  "code": "123456"
}
```

**Mô tả xử lý chi tiết:**

1. **Tìm session theo CID**: Hệ thống truy vấn bảng `users_login_session` để tìm record tương ứng với CID. Nếu không tìm thấy (CID không hợp lệ hoặc đã bị xóa), trả về lỗi.
2. **Lấy thông tin user**: Từ userId trong session, truy xuất thông tin user bao gồm `SecretTOTP` - secret key đã được lưu khi user thiết lập 2FA.
3. **Validate mã TOTP**: Sử dụng thư viện `pquerna/otp` để validate mã 6 chữ số. Thuật toán TOTP tính toán mã dựa trên:
   - Secret key của user
   - Thời gian hiện tại (chia thành các khoảng 30 giây)
   - Mã chỉ có hiệu lực trong khoảng thời gian 30 giây đó (với một số tolerance cho clock skew).
4. **Kiểm tra CID hết hạn**: Parse UUID v1 để lấy timestamp và so sánh với thời gian hiện tại. Nếu quá 5 phút (300 giây), CID được coi là hết hạn.
5. **Cleanup session**: Xóa record trong bảng `users_login_session` ngay sau khi xác thực thành công để ngăn replay attack.
6. **Tạo Access Token**: Tạo JWT Access Token giống như luồng đăng nhập thông thường.

**Response thành công:**

Listing 4.7: Response xác thực TOTP thành công

```
HTTP Status: 200 OK

{
```

```
  "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "user": {
    "id": "550e8400-e29b-41d4-a716-446655440000",
    "username": "nam123",
    "email": "nam@example.com"
  }
}
```

**Response khi TOTP sai:**

Listing 4.8: Response lỗi TOTP

```
HTTP Status: 401 Unauthorized

{
  "error": "Unauthorized",
  "message": "Invalid or expired TOTP code"
}
```

**Response khi CID hết hạn:**

Listing 4.9: Response CID hết hạn

```
HTTP Status: 401 Unauthorized

{
  "error": "Unauthorized",
  "message": "CID has expired"
}
```

### 4.1.4. Thiết lập xác thực 2 lớp

Người dùng có thể kích hoạt 2FA thông qua hai API:

#### 4.1.4.1. Bước 1: Tạo TOTP secret

**Request:**

Listing 4.10: Request tạo TOTP secret

```
POST /auth/totp/setup
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

**Mô tả xử lý:**

- Yêu cầu user đã đăng nhập (có Bearer token hợp lệ).
- Tạo TOTP secret sử dụng thuật toán tiêu chuẩn với issuer là "File Sharing" và account name là username.

- Tạo OTP URL theo định dạng: `otpauth://totp/File%20Sharing:username?secret=X`
- Encode URL thành QR code PNG 256x256 pixels, sau đó encode thành base64.
- Lưu secret vào database (chưa enable TOTP).

**Response thành công:**

Listing 4.11: Response tạo TOTP secret thành công

```
HTTP Status: 200 OK

{
  "message": "TOTP secret generated",
  "totpSetup": {
    "secret": "NB2W45DFOIZA====",
    "qrCode": "..."
  }
}
```

**Phân tích response:**

- `secret`: Backup code dạng Base32, user nên lưu lại phòng trường hợp mất điện thoại.
- `qrCode`: Data URL chứa hình ảnh QR code, có thể hiển thị trực tiếp trong thẻ `<img>`.

#### 4.1.4.2. Bước 2: Xác minh và kích hoạt

**Request:**

Listing 4.12: Request xác minh TOTP

```
POST /auth/totp/verify
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
Content-Type: application/json

{
  "code": "123456"
}
```

**Mô tả xử lý:**

- User quét QR code bằng ứng dụng authenticator (Google Authenticator, Authy, Microsoft Authenticator...).
- App sẽ hiển thị mã 6 số thay đổi mỗi 30 giây.

- User nhập mã vào form và gửi request.
- Server validate mã với secret đã lưu.
- Nếu đúng, cập nhật `EnableTOTP = true` cho user.

**Response thành công:**

Listing 4.13: Response kích hoạt TOTP thành công

```
HTTP Status: 200 OK

{
  "message": "TOTP verified successfully",
  "totpEnabled": true
}
```

**Response khi mã sai:**

Listing 4.14: Response lỗi mã TOTP

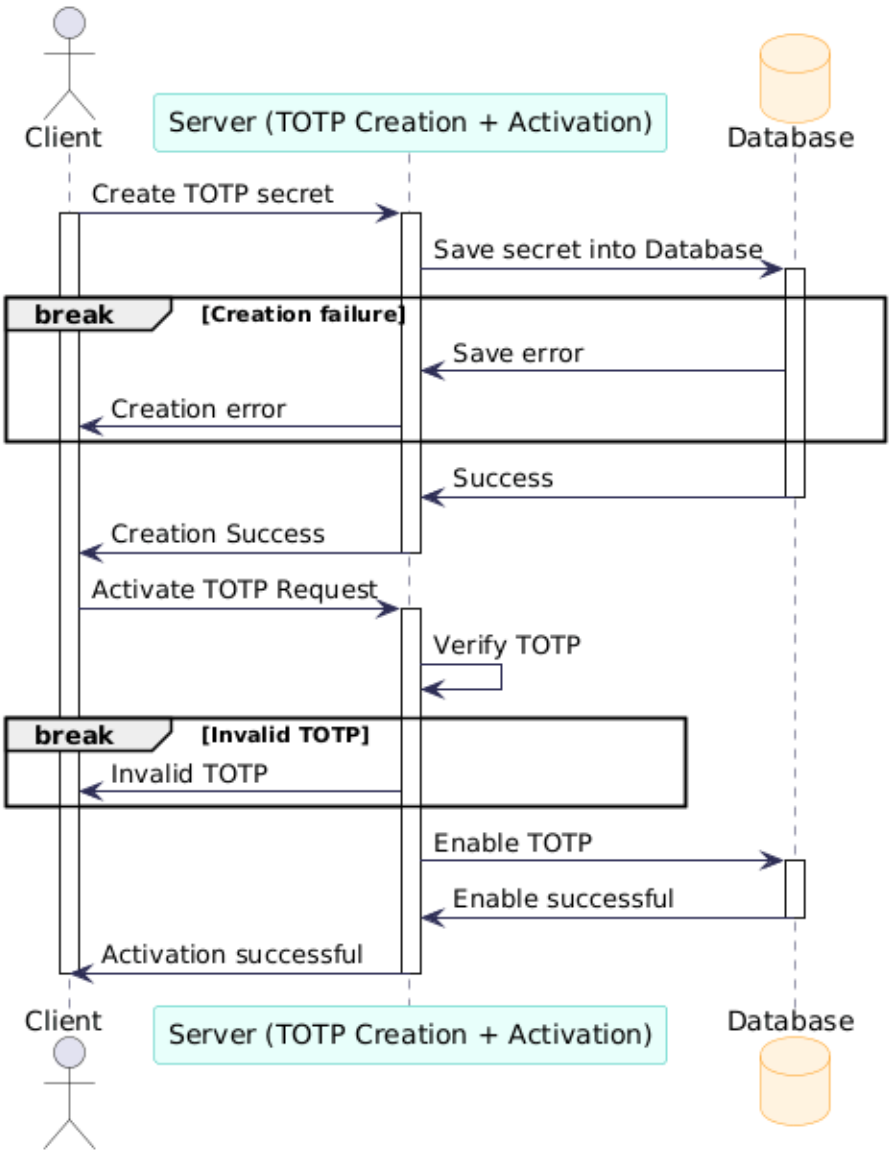
```
HTTP Status: 400 Bad Request

{
  "error": "Invalid TOTP code",
  "message": "The provided code is incorrect or expired"
}
```

**Sequence diagram:**

### 4.1.5. Các biện pháp bảo mật

Bảng 4.1: Các biện pháp bảo mật trong quy trình đăng nhập

| **Biện pháp** | **Mô tả chi tiết** | **Mục đích** |
|---|---|---|
| Bcrypt hash | Password được hash với bcrypt sử dụng cost factor mặc định (10). Việc verify một password mất khoảng 100ms, làm brute-force attack trở nên không khả thi. | Bảo vệ password |
| UUID v1 cho CID | UUID v1 chứa timestamp trong 60 bit đầu, cho phép extract thời gian tạo mà không cần query database. | Kiểm tra expiry hiệu quả |
| Session TTL 5 phút | CID chỉ có hiệu lực 5 phút từ khi tạo. Sau 5 phút, user phải bắt đầu lại từ bước nhập email/password. | Ngăn brute-force TOTP |
| TOTP RFC 6238 | Mã thay đổi mỗi 30 giây dựa trên thời gian UTC. Server cho phép tolerance 1 khoảng thời gian (±30s) để xử lý clock skew. | Xác thực hai yếu tố |
| Token Blacklist | Khi logout, Access Token được thêm vào Redis blacklist với TTL = thời gian còn lại của token. AuthMiddleware kiểm tra blacklist trước khi chấp nhận token. | Vô hiệu hóa token sau logout |
| Xóa session sau verify | Record trong `users_login_session` bị xóa ngay sau khi TOTP verify thành công. | Ngăn replay attack |




Hình 4.3: Sơ đồ trình tự thiết lập TOTP

## 4.2. Quy trình Tải lên tệp tin (File Upload Logic)

### 4.2.1. Mô tả tổng quan

Hệ thống hỗ trợ cả upload có xác thực (authenticated) và upload ẩn danh (anonymous). File được lưu trữ với các tùy chọn bảo mật như mật khẩu, thời hạn hiệu lực, và chia sẻ riêng tư cho người dùng cụ thể.

### 4.2.2. Bước 1: Client gửi request upload

**Request (Authenticated upload - file private):**

Listing 4.15: Request upload file có xác thực

```
POST /files/upload
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
Content-Type: multipart/form-data

------WebKitFormBoundary
Content-Disposition: form-data; name="file"; filename="report
   .pdf"
Content-Type: application/pdf

[Binary content of file]
------WebKitFormBoundary
Content-Disposition: form-data; name="isPublic"

false
------WebKitFormBoundary
Content-Disposition: form-data; name="password"

mysecretpass123
------WebKitFormBoundary
Content-Disposition: form-data; name="availableFrom"

2025-01-10T00:00:00Z
------WebKitFormBoundary
Content-Disposition: form-data; name="availableTo"

2025-01-17T23:59:59Z
------WebKitFormBoundary
Content-Disposition: form-data; name="sharedWith"

["user1@example.com", "user2@example.com"]
```

```
------WebKitFormBoundary--
```

**Request (Anonymous upload - file public):**

Listing 4.16: Request upload file ẩn danh

```
POST /files/upload
Content-Type: multipart/form-data

------WebKitFormBoundary
Content-Disposition: form-data; name="file"; filename="image.
    png"
Content-Type: image/png

[Binary content of file]
------WebKitFormBoundary
Content-Disposition: form-data; name="isPublic"

true
------WebKitFormBoundary--
```

**Các tham số trong request:**

Bảng 4.2: Các tham số trong request upload

| Tham số | Kiểu | Bắt buộc | Mô tả |
|---|---|---|---|
| file | Binary | Có | File cần upload |
| isPublic | Boolean | Không | `true` = public, `false` = private. Mặc định: `true` cho anonymous |
| password | String | Không | Mật khẩu bảo vệ file, tối thiểu 8 ký tự |
| availableFrom | DateTime | Không | Thời điểm bắt đầu hiệu lực (ISO 8601) |
| availableTo | DateTime | Không | Thời điểm hết hạn (ISO 8601) |
| sharedWith | Array[String] | Không | Danh sách email được chia sẻ |

### 4.2.3. Bước 2: Middleware xác thực (Optional Authentication)

**Mô tả xử lý:**

- Request đi qua middleware `AuthMiddlewareUpload` - đây là middleware đặc biệt không yêu cầu bắt buộc phải có token.
- Nếu có header `Authorization: Bearer <token>`:
  - Kiểm tra token có trong blacklist không.
  - Parse và validate JWT.
  - Extract `userId` từ claims và set vào context.

- Nếu không có token hoặc token không hợp lệ:
  - Không trả lỗi, request tiếp tục xử lý như anonymous user.
  - `userId` trong context sẽ là `nil`.

### 4.2.4. Bước 3: Validation đầu vào

**Mô tả xử lý:**

1. **Kiểm tra file tồn tại:**

   Nếu không có file trong form-data, trả về lỗi.

   Listing 4.17: Response lỗi thiếu file

```
HTTP Status: 400 Bad Request

{
  "error": "Validation error",
  "message": "File is required"
}
```

2. **Kiểm tra kích thước file:**

   So sánh `fileHeader.Size` với `Policy.MaxFileSizeMB * 1024 * 1024`. Giới hạn mặc định: 100MB.

   Listing 4.18: Response lỗi file quá lớn

```
HTTP Status: 413 Payload Too Large

{
  "error": "Payload too large",
  "message": "File size exceeds the system limit"
}
```

3. **Kiểm tra password (nếu có):**

   Password phải có độ dài tối thiểu 8 ký tự.

   Listing 4.19: Response lỗi password yếu

```
HTTP Status: 400 Bad Request

{
  "error": "Validation error",
  "message": "Password must be at least 8 characters long"
}
```

4. **Kiểm tra quy tắc anonymous upload:**

   Nếu không có token (anonymous) và `isPublic = false` hoặc có `sharedWith`, trả về lỗi.

Listing 4.20: Response lỗi anonymous upload private

```
HTTP Status: 401 Unauthorized

{
  "error": "Unauthorized",
  "message": "Private uploads require authentication"
}
```

5. **Kiểm tra xung đột public/sharedWith:**

   File public không được có danh sách `sharedWith`.

Listing 4.21: Response lỗi xung đột

```
HTTP Status: 400 Bad Request

{
  "error": "Bad request",
  "message": "Public files are not allowed to have a
    whitelist"
}
```

### 4.2.5. Bước 4: Tính toán thời hạn hiệu lực (Validity Period)

**Mô tả xử lý:**

Hệ thống tính toán `availableFrom` và `availableTo` theo logic sau:

Bảng 4.3: Logic tính toán thời hạn hiệu lực

| **availableFrom** | **availableTo** | **Kết quả** |
|---|---|---|
| Có giá trị | Có giá trị | Sử dụng trực tiếp cả hai giá trị |
| Không có | Có giá trị | from = thời điểm hiện tại |
| Có giá trị | Không có | to = from + DefaultValidityDays |
| Không có | Không có | from = now, to = now + DefaultValidityDays |

**Validation:**

- `availableFrom < availableTo`: Thời điểm bắt đầu phải trước thời điểm kết thúc.
- Khoảng thời gian $\geq$ `MinValidityHours` (mặc định: 1 giờ).
- Khoảng thời gian $\leq$ `MaxValidityDays` (mặc định: 365 ngày).

**Response lỗi validation thời gian:**

Listing 4.22: Response lỗi thời gian không hợp lệ

```
HTTP Status: 400 Bad Request

{
  "error": "Validation error",
  "message": "AvailableFrom cannot be after AvailableTo"
}
```

### 4.2.6. Bước 5: Tạo metadata cho file

**Mô tả xử lý:**

Bảng 4.4: Metadata được tạo cho file

| Field | Giá trị | Mô tả |
|---|---|---|
| fileId | `uuid.New().String()` | UUID v4 ngẫu nhiên |
| shareToken | `GenerateRandomString(16)` | Chuỗi 16 ký tự alphanumeric |
| storageName | fileId | Tên file trên storage = UUID |
| fileName | `fileHeader.Filename` | Tên gốc từ client |
| fileSize | `fileHeader.Size` | Kích thước bytes |
| mimeType | `Content-Type` header | MIME type từ header |
| ownerId | userId hoặc nil | Owner (nil nếu anonymous) |
| passwordHash | `bcrypt.GenerateFromPassword` | Hash của password (nếu có) |

### 4.2.7. Bước 6: Lưu file vật lý

**Mô tả xử lý:**

- Gọi `storage.SaveFile(fileHeader, storageName)`.
- File được lưu với tên là UUID thay vì tên gốc.

**Lý do sử dụng UUID làm storage name:**

- Tránh trùng lặp khi nhiều user upload file cùng tên.
- Ẩn thông tin tên file gốc trên hệ thống lưu trữ.
- Ngăn chặn path traversal attack (ví dụ: tên file chứa `../`).
- Dễ dàng mapping giữa storage và database.

### 4.2.8. Bước 7: Lưu metadata vào database

**Mô tả xử lý:**

- Gọi `fileRepo.CreateFile(ctx, newFile)` để lưu record vào database.

**Cơ chế Rollback:**

- Nếu việc lưu database thất bại (ví dụ: constraint violation, connection error), hệ thống tự động xóa file vật lý đã lưu ở bước trước.
- Đảm bảo không có file "mồ côi" (orphan files) - file tồn tại trên storage nhưng không có metadata trong database.

### 4.2.9. Bước 8: Xử lý chia sẻ (nếu có)

**Mô tả xử lý:**

- Nếu request có danh sách `sharedWith`, gọi `sharedRepo.ShareFileWithUsers(ctx, fileId, emails)`.
- Lưu các record vào bảng `file_shares`, liên kết `fileId` với các `userId` tương ứng với email.

### 4.2.10. Bước 9: Trả về kết quả

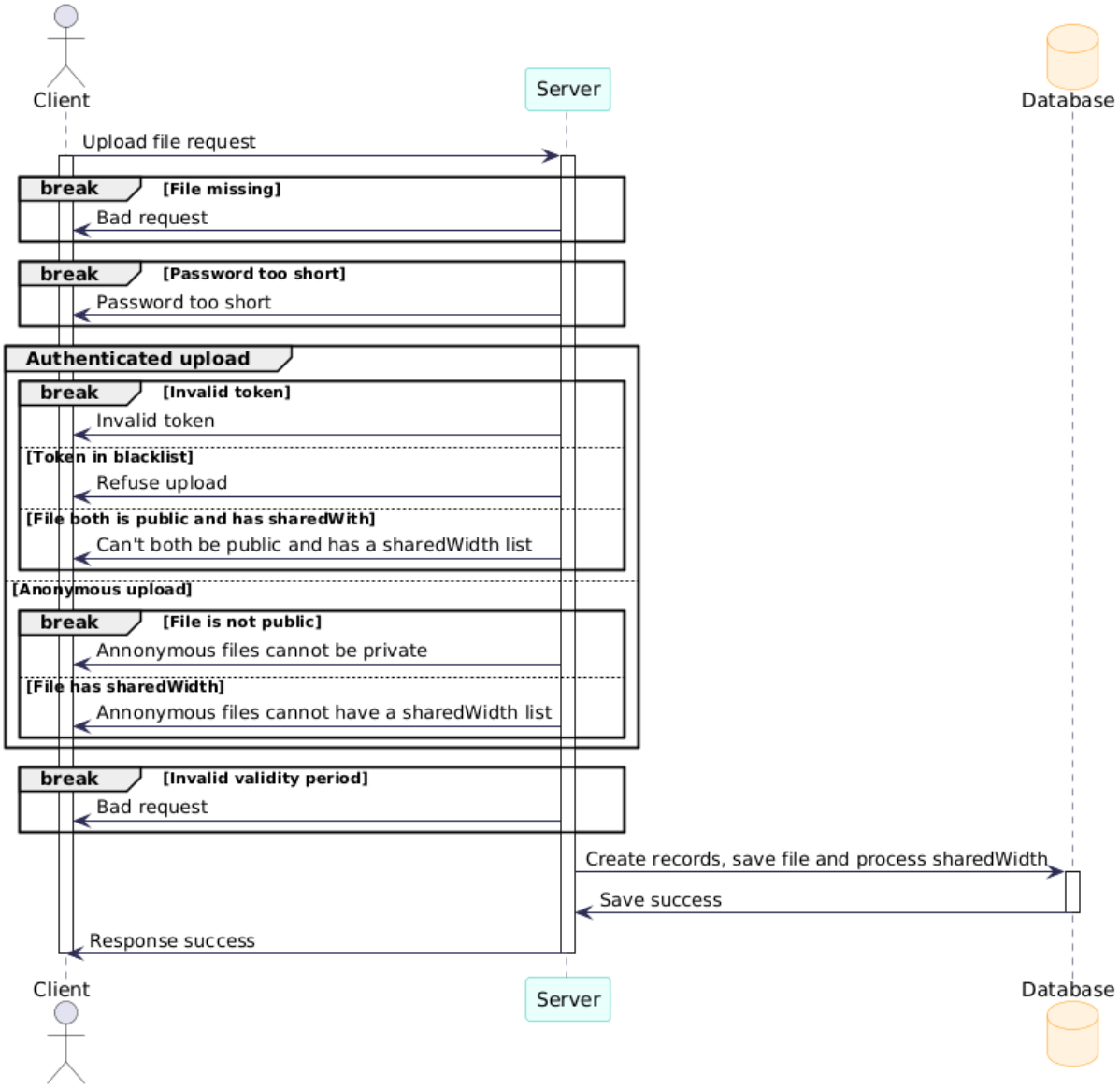
**Response thành công:**

Listing 4.23: Response upload thành công

```
HTTP Status: 201 Created

{
  "success": true,
  "message": "File uploaded successfully",
  "file": {
    "id": "550e8400-e29b-41d4-a716-446655440000",
    "fileName": "report.pdf",
    "shareToken": "a1b2c3d4e5f6g7h8",
    "isPublic": false
  }
}
```

**Phân tích response:**

- `id`: UUID của file, dùng cho các thao tác quản lý (xem chi tiết, xóa).
- `fileName`: Tên file gốc để hiển thị cho user.
- `shareToken`: Token 16 ký tự để tạo link chia sẻ.
- `isPublic`: Xác nhận chế độ truy cập đã được set.




Hình 4.4: Sơ đồ trình tự tải tập tin

### 4.2.11. Quy tắc nghiệp vụ quan trọng

Bảng 4.5: Quy tắc nghiệp vụ cho upload file

| **Quy tắc** | **Điều kiện** | **Kết quả** |
|---|---|---|
| Anonymous Upload | Không có token | File bắt buộc `isPublic = true`, không có `sharedWith` |
| Private Upload | Có token | File có thể `isPublic = false` và/hoặc có `sharedWith` |
| Public + SharedWith | Xung đột | Lỗi 400: "Public files are not allowed to have a whitelist" |
| Password Protection | Bất kỳ | Password $\geq$ 8 ký tự, được hash bằng bcrypt |

**Sequence diagram:**

## 4.3. Quy trình Tải xuống bảo mật (Secure Download Logic)

### 4.3.1. Mô tả tổng quan

Quy trình tải xuống được bảo vệ bởi nhiều lớp bảo mật: kiểm tra quyền truy cập, xác thực mật khẩu file, kiểm tra trạng thái thời hạn, và ghi lại lịch sử download để audit.

### 4.3.2. Bước 1: Client gửi request download

**Request (File public, không có password):**

Listing 4.24: Request download file public

```
GET /files/a1b2c3d4e5f6g7h8/download
```

**Request (File có password):**

Listing 4.25: Request download file có password

```
GET /files/a1b2c3d4e5f6g7h8/download?password=mysecretpass123
```

**Request (File private với authentication):**

Listing 4.26: Request download file private

```
GET /files/a1b2c3d4e5f6g7h8/download
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

### 4.3.3. Bước 2: Xác định danh tính người dùng

**Mô tả xử lý:**

- Request đi qua middleware `AuthMiddlewareUpload` (optional authentication).
- Nếu có token hợp lệ: extract `userId` và set vào context.
- Nếu không có token hoặc token không hợp lệ: `userId = ""` (anonymous).

### 4.3.4. Bước 3: Truy xuất thông tin file

**Mô tả xử lý:**

- Gọi `fileRepo.GetFileByToken(ctx, shareToken)`.
- Nếu không tìm thấy file, trả về lỗi 404.

**Response lỗi:**

Listing 4.27: Response file không tồn tại

```
HTTP Status: 404 Not Found

```

```
{
  "error": "Not found",
  "message": "File not found"
}
```

### 4.3.5. Bước 4: Xác định trạng thái thời hạn

**Mô tả xử lý:**

Dựa trên thời điểm hiện tại (`now`) và thông tin `availableFrom`, `availableTo`:

- Nếu `now < availableFrom`: `status = "pending"` (Chưa tới thời gian phát hành)
- Nếu `now > availableTo`: `status = "expired"` (Đã hết hạn)
- Ngược lại: `status = "active"` (Đang hoạt động bình thường)

### 4.3.6. Bước 5: Kiểm tra quyền truy cập (Authorization)

**Mô tả xử lý:**

Hệ thống kiểm tra theo thứ tự ưu tiên:

1. **Kiểm tra vai trò Admin:** Nếu `requester.Role == "admin"` → Cho phép truy cập tất cả file trong mọi trạng thái.
2. **Kiểm tra Owner:** Nếu `file.OwnerId == userId` → Cho phép truy cập kể cả khi file đã expired hoặc pending.
3. **Kiểm tra file Public:** Nếu `file.IsPublic == true`:
   - Nếu `status == "active"` → Cho phép truy cập.
   - Nếu `status == "expired"` hoặc `"pending"` → Từ chối.
4. **Kiểm tra SharedWith list:** Nếu `userId` có trong danh sách `sharedWith`:
   - Nếu `status == "active"` → Cho phép truy cập.
   - Nếu `status != "active"` → Từ chối.
5. **Không thỏa mãn điều kiện nào:** Trả về lỗi 403 Forbidden.

**Response lỗi quyền truy cập:**

Listing 4.28: Response không có quyền truy cập

```
HTTP Status: 403 Forbidden

```

```
{
  "error": "Forbidden",
  "message": "You don't have permission to access this file"
}
```

### 4.3.7. Bước 6: Kiểm tra trạng thái thời hạn cho non-owner

**Mô tả xử lý:**

Với những người không phải owner hoặc admin:

**Trường hợp file EXPIRED:**

Listing 4.29: Response file hết hạn

```
HTTP Status: 410 Gone

{
  "error": "File expired",
  "expiredAt": "2025-01-17T23:59:59Z"
}
```

**Trường hợp file PENDING:**

Listing 4.30: Response file chưa tới hạn

```
HTTP Status: 423 Locked

{
  "error": "File not yet available",
  "availableFrom": "2025-01-20T10:00:00Z",
  "hoursUntilAvailable": 48.5
}
```

**Phân tích response PENDING:**

- `availableFrom`: Thời điểm file sẽ khả dụng.
- `hoursUntilAvailable`: Số giờ còn lại, giúp client có thể hiển thị countdown.

### 4.3.8. Bước 7: Xác thực mật khẩu file

**Mô tả xử lý:**

Nếu file có `hasPassword = true`:

1. Kiểm tra request có cung cấp password không (qua query parameter).
2. Nếu không có password:

Listing 4.31: Response yêu cầu password

```
HTTP Status: 401 Unauthorized

{
  "error": "Password required",
  "message": "This file is password protected"
}
```

3. Nếu có password, sử dụng `bcrypt.CompareHashAndPassword()` để so sánh.
4. Nếu password không khớp:

Listing 4.32: Response sai password

```
HTTP Status: 401 Unauthorized

{
  "error": "Incorrect password",
  "message": "The file password is incorrect"
}
```

### 4.3.9. Bước 8: Đọc file từ storage

**Mô tả xử lý:**

- Gọi `storage.GetFile(fileId)` để lấy file stream.
- `fileId` chính là `storageName` được lưu khi upload.

### 4.3.10. Bước 9: Ghi lịch sử download

**Mô tả xử lý:**

Trước khi trả file về cho client, hệ thống ghi lại bản ghi download history:

Bảng 4.6: Dữ liệu lịch sử download

| Field | Giá trị |
|---|---|
| fileId | ID của file được download |
| userId | ID người download (`nil` nếu anonymous) |
| downloadedAt | Timestamp hiện tại |

**Mục đích:**

- Audit: Owner có thể xem ai đã tải file của mình.
- Thống kê: Đếm số lượt download, unique downloaders.
- Không lưu IP/User-Agent để bảo vệ quyền riêng tư người dùng.

### 4.3.11. Bước 10: Trả file về client

**Response thành công (Download):**

Listing 4.33: Response download file

```
HTTP/1.1 200 OK
Content-Type: application/pdf
Content-Length: 2048576
Content-Disposition: attachment; filename="report.pdf"

[Binary content of file]
```

**Response thành công (Preview):**

Listing 4.34: Response preview file

```
HTTP/1.1 200 OK
Content-Type: application/pdf
Content-Length: 2048576
Content-Disposition: inline; filename="report.pdf"

[Binary content of file]
```

**Sự khác biệt:**

- `/download`: Header `Content-Disposition: attachment` → Browser sẽ download file.
- `/preview`: Header `Content-Disposition: inline` → Browser sẽ cố gắng hiển thị trực tiếp (PDF, image, video...).

### 4.3.12. Ma trận quyền truy cập

Bảng 4.7: Ma trận quyền truy cập theo vai trò và trạng thái file

| **Vai trò** | **File ACTIVE** | **File PENDING** | **File EXPIRED** |
|---|---|---|---|
| Admin | ✓Cho phép | ✓Cho phép | ✓Cho phép |
| Owner | ✓Cho phép | ✓Cho phép | ✓Cho phép |
| Shared User | ✓Cho phép | × 423 Locked | × 410 Gone |
| Public User | ✓Cho phép | × 423 Locked | × 410 Gone |
| Stranger (file private) | × 403 Forbidden | × 403 Forbidden | × 403 Forbidden |

### 4.3.13. Tổng hợp các lớp bảo mật

Bảng 4.8: Các lớp bảo mật trong quy trình download

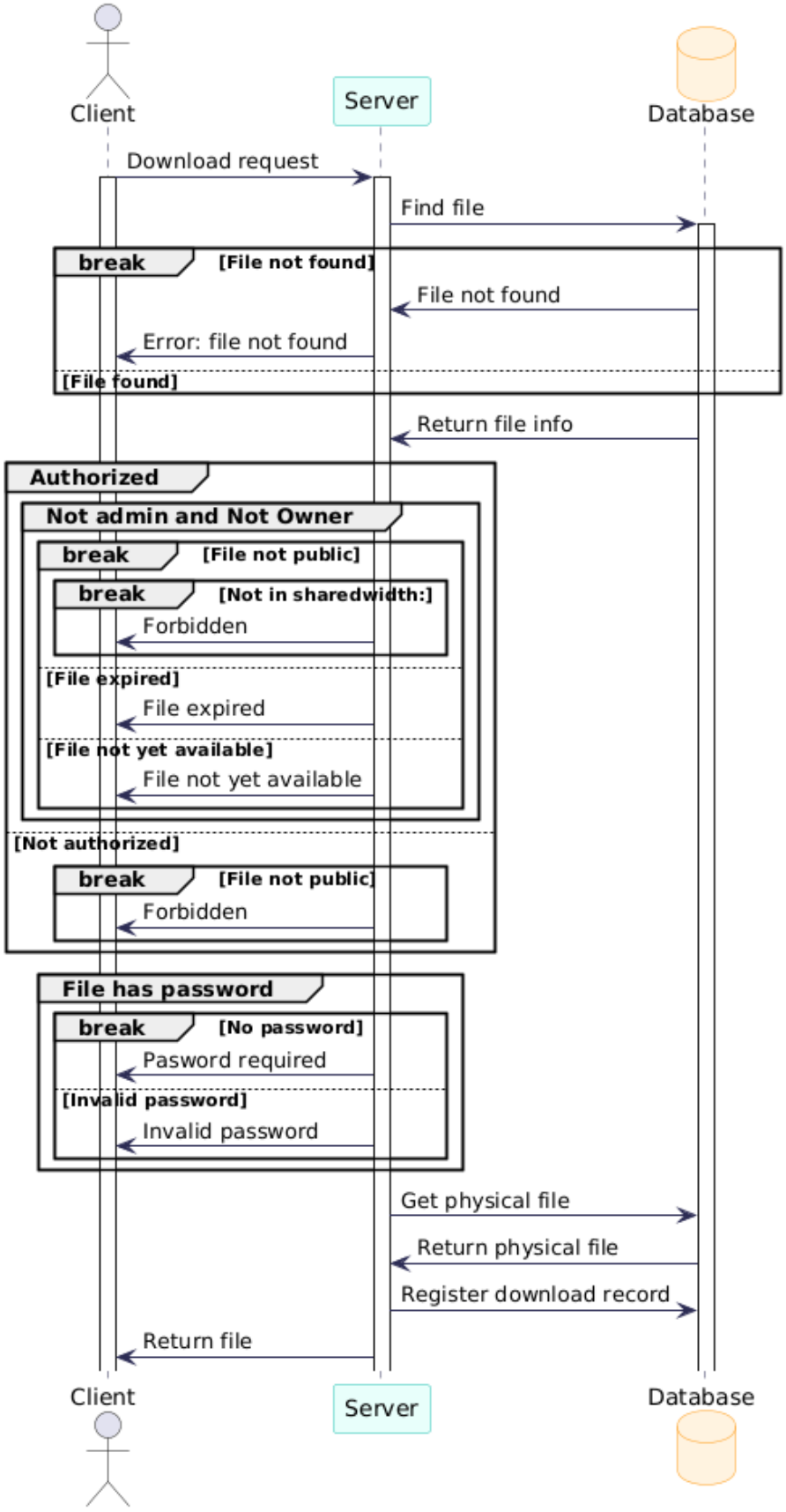
| **Lớp** | **Mô tả** | **HTTP Status khi thất bại** |
|---|---|---|
| 1 | File Existence - Kiểm tra file tồn tại | 404 Not Found |
| 2 | Authorization - Kiểm tra quyền (admin/owner/shared/public) | 403 Forbidden |
| 3 | Time Validation - Kiểm tra trạng thái thời hạn | 410 Gone / 423 Locked |
| 4 | Password - Xác thực mật khẩu file | 401 Unauthorized |
| 5 | Audit Log - Ghi lịch sử download | N/A (không block) |

**Sequence diagram:**

### 4.3.14. Tổng hợp các trường hợp lỗi

Bảng 4.9: Tổng hợp các trường hợp lỗi trong quy trình download

| **Trường hợp** | **HTTP Status** | **Error Message** |
|---|---|---|
| ShareToken không tồn tại | 404 | "File not found" |
| File private, không đăng nhập | 403 | "You don't have permission to access this file" |
| File private, không trong sharedWith | 403 | "You don't have permission to access this file" |
| File đã hết hạn (non-owner) | 410 | "File expired" |
| File chưa tới hạn (non-owner) | 423 | "File not yet available" |
| File có password, không cung cấp | 401 | "This file is password protected" |
| File có password, sai password | 401 | "The file password is incorrect" |




Hình 4.5: Sơ đồ trình tự tải xuống tập tin