TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN
BỘ MÔN CÔNG NGHỆ TRI THỨC
NHẬP MÔN MÃ HÓA - MẬT MÃ

BÁO CÁO ĐỒ ÁN 02.
ỨNG DỤNG CHIA SẺ GHI CHÚ

CHUNG GIA KHÁNH – 23120282
TRƯƠNG SỸ KHÁNH – 23120284
MÃ TUYẾT NGÂN – 23120302
TRƯƠNG THÁI NGUYÊN – 23120308

Tp. Hồ Chí Minh, tháng 11/**2025**

Mục lục

I. Giới thiệu đồ án 3

1.Mục tiêu ứng dụng: 3

2. Cách chạy chương trình từ mã nguồn nộp kèm: 3

3. Các chức năng đã triển khai và tính năng nâng cao 4

II. Thiết kế và kiến trúc 4

1.Mô tả kiến trúc hệ thống và mục đích thiết kế 4

2. Các thành phần chính 6

3. Công nghệ sử dụng. 7

III. Chi tiết cài đặt 7

1 Quy trình Mã hóa & Chia sẻ (ECDH + AES) - phần cốt lõi của ứng dụng. 8

2. Tối ưu hóa (Optimization) 8

IV. Thách thức và giải pháp: Trình bày các vấn đề đã gặp và cách giải quyết. . 8

1 Chuyển đổi từ RSA sang ECDH 8

2 Bảo mật Mật khẩu 9

3 Đồng bộ hóa (Concurrency) 9

V. Kiểm thử 9

2. Kết quả kiểm thử 9

2.1. Automated Testing 9

2.2. Manual Test 10

VI. Cải tiến tương lai: Đề xuất các cải tiến hoặc tính năng bổ sung. 10

1. Cải tiến về Bảo mật 10

2. Cải tiến về Hiệu năng 11

3. Cải tiến về Tính năng 11

4. Cải tiến về Monitoring & Operations 11
5. Cải tiến về User Experience 11
6. Cải tiến về Scalability 12

I. Giới thiệu đồ án

1.Mục tiêu ứng dụng:

Xây dựng một nền tảng chia sẻ ghi chú an toàn, đảm bảo tính riêng tư của dữ liệu thông qua cơ chế mã hóa phía client. Máy chủ đóng vai trò lưu trữ "mù", không thể đọc được nội dung thực tế của ghi chú.

2. Cách chạy chương trình từ mã nguồn nộp kèm:

2.1. Cài đặt

Yêu cầu: Go 1.22+.

1. Xóa server.db và các file *.pem cũ (nếu có).
2. Chạy Server:

go run ./cmd/server/main.go

3. Chạy Client:

go run ./cmd/client/main.go

Hướng Dẫn Sử Dụng (Client CLI)

Sau khi chạy Client (go run ./cmd/client/main.go), bạn sẽ thấy các menu sau tùy thuộc vào trạng thái đăng nhập.

Menu Chính (Chưa đăng nhập)

1. **Đăng nhập**: Dùng tài khoản đã có để vào hệ thống.
2. **Đăng ký**: Tạo tài khoản mới. Hệ thống sẽ tự động sinh cặp khóa Public/Private (lưu tại file username.pem) phục vụ cho việc mã hóa/giải mã.
3. **Tải từ Link**: Tải ghi chú từ đường dẫn chia sẻ công khai (không cần tài khoản).
4. **Thoát**: Đóng ứng dụng.

Menu Người Dùng (Đã đăng nhập)

Sau khi đăng nhập thành công, bạn có thể thực hiện các chức năng:

1. **Tạo ghi chú**: Upload và mã hóa file.
    - Nhập tiêu đề ghi chú.
    - Nhập đường dẫn file (VD: C:\tailieu\secret.txt).
2. **Liệt kê ghi chú**: Xem danh sách tất cả ghi chú bạn sở hữu hoặc được chia sẻ.

3. **Xem ghi chú**: Giải mã và tải nội dung ghi chú về máy.
    - Cần nhập Note ID (lấy từ chức năng liệt kê).
4. **Chia sẻ ghi chú**: Chia sẻ quyền truy cập cho người dùng khác trong hệ thống.
    - Cần Note ID và Tên người nhận.
5. **Chia sẻ qua Link**: Tạo URL chia sẻ công khai (có chứa Token và Key giải mã).
6. **Tải từ Link**: Tải ghi chú từ Link chia sẻ (tương tự chức năng ở menu chính).
7. **Xóa ghi chú**: Xóa ghi chú khỏi server (chỉ dành cho chủ sở hữu).
8. **Đăng xuất**.
9. **Thoát**.

3. Các chức năng đã triển khai và tính năng nâng cao.

- Xác thực an toàn: Đăng ký/Đăng nhập với mật khẩu được bảo vệ bởi Salt và SHA-256. Quản lý phiên bằng JWT.
- Mã hóa đầu-cuối (E2EE):
    - Mỗi file được mã hóa bằng một khóa AES ngẫu nhiên riêng biệt.
    - Khóa AES được bảo vệ bằng cơ chế trao đổi khóa Diffie-Hellman (ECDH X25519)
- Chia sẻ linh hoạt:
    - Chia sẻ cho người dùng cụ thể trong hệ thống.
    - Chia sẻ công khai qua link (sử dụng Token và Fragment URL để bảo mật khóa).
- Quản lý ghi chú:
    - Upload/Download file mã hóa.
    - Xóa ghi chú.
    - Tự động kiểm tra thời gian hết hạn.

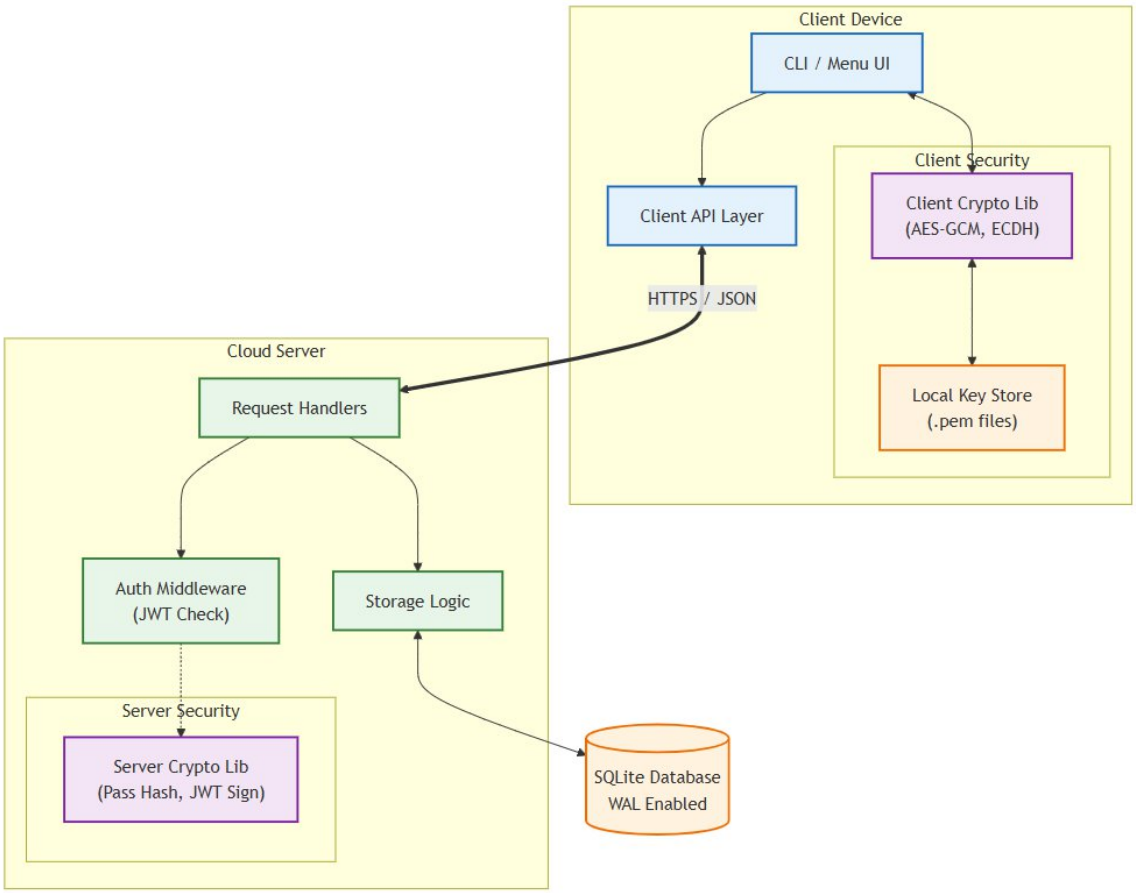
II. Thiết kế và kiến trúc

1.Mô tả kiến trúc hệ thống và mục đích thiết kế

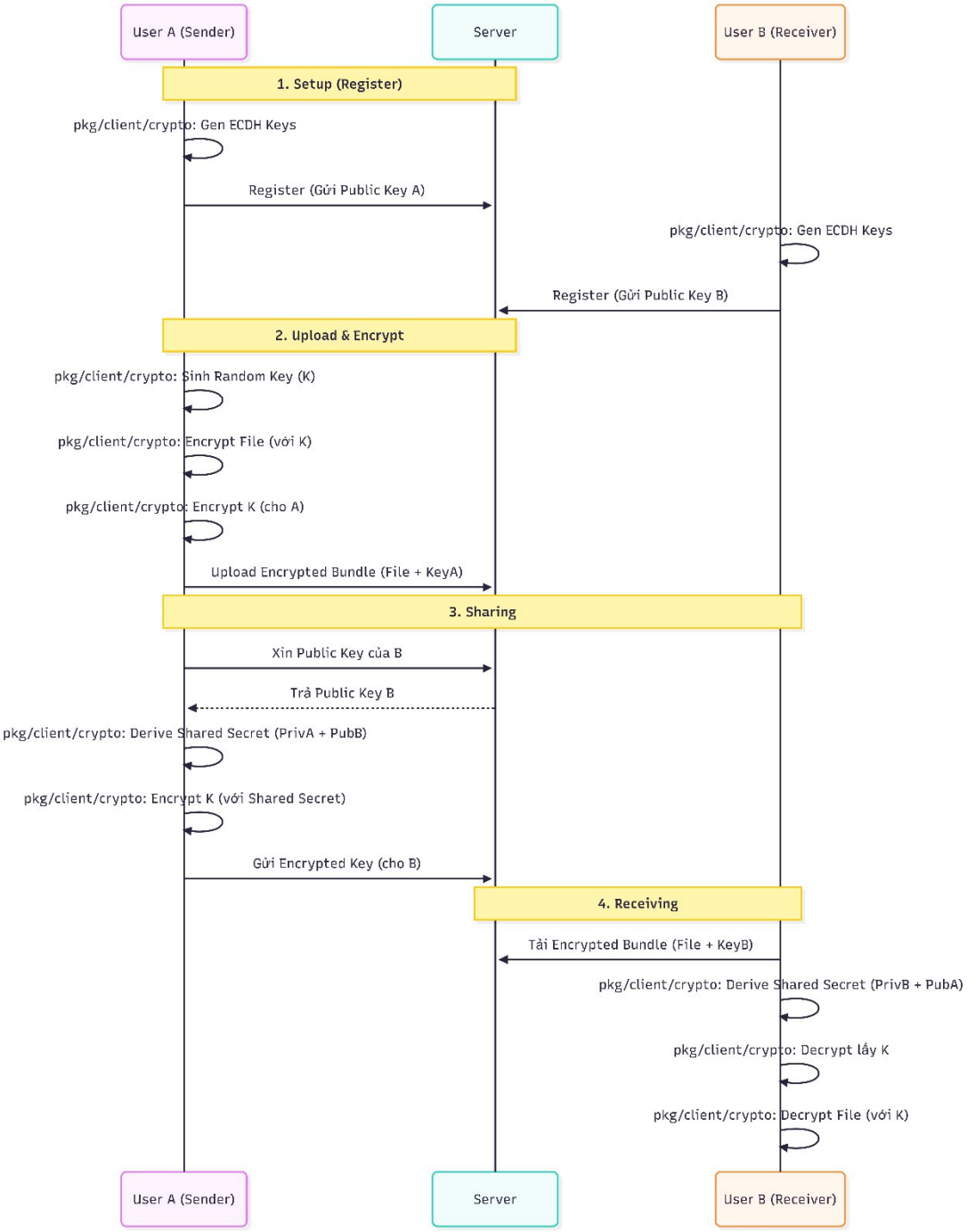
Hệ thống được xây dựng với mục tiêu tối thượng là **Bảo mật** và **Quyền riêng tư**.

- **Zero-Knowledge**: Server không bao giờ biết nội dung file gốc hay khóa mã hóa file.
- **Client-Centric**: Mọi tác vụ mã hóa/giải mã đều diễn ra tại Client.

*Sơ đồ thiết kế



*Sơ đồ luồng hoạt động



2. Các thành phần chính

1. **Client Application**:
    - **Giao diện (UI)**: Cung cấp menu dòng lệnh (CLI) để người dùng tương tác (Đăng ký, Đăng nhập, Gửi/Nhận file).

- **Client API Layer**: Module chịu trách nhiệm đóng gói dữ liệu và gửi các HTTP Request tới Server.
- **Client Crypto Module**: Thành phần quan trọng nhất, thực hiện mã hóa AES-256 nội dung file và trao đổi khóa ECDH. Đảm bảo dữ liệu rời khỏi máy người dùng luôn ở dạng mã hóa.

2. **Server Application:**
   - **Request Handlers**: Tiếp nhận các yêu cầu từ Client, kiểm tra tính hợp lệ của dữ liệu đầu vào.
   - **Authentication Middleware**: Xác thực người dùng thông qua JWT Token trước khi cho phép truy cập tài nguyên.
   - **Server Crypto Module**: Chỉ thực hiện các tác vụ của Server như băm mật khẩu (Hashing) và ký Token. Không dính dáng đến khóa giải mã file.
   - **Storage Layer**: Tương tác trực tiếp với cơ sở dữ liệu SQLite, thực hiện các truy vấn tối ưu hóa (WAL Mode).
3. **Database (Cơ sở dữ liệu)**:
   - **SQLite**: Lưu trữ bền vững thông tin người dùng, metadata của ghi chú và các khối dữ liệu (BLOB) đã mã hóa.

3. Công nghệ sử dụng.

- Ngôn ngữ: Go (Golang) phiên bản 1.22+.
- Cơ sở dữ liệu: SQLite (thư viện modernc.org/sqlite v1.33.1 - Pure Go driver).
- Xác thực:
  - JWT: github.com/golang-jwt/jwt/v5 để tạo và xác thực token Bearer.
  - Password Hashing: SHA-256 kết hợp với Salt (16 bytes random hex) tự xây dựng.
- Mật mã học (Cryptography):
  - AES-GCM (256-bit): Dùng để mã hóa nội dung file (Thư viện chuẩn crypto/aes, crypto/cipher).
  - ECDH (Curve25519/X25519): Trao đổi khóa an toàn (Thư viện chuẩn crypto/ecdh).
  - KDF: SHA-256 để dẫn xuất khóa từ Shared Secret (Thư viện chuẩn crypto/sha256).
  - Random: crypto/rand (CSPRNG) để tạo Salt, IV, và Key.

III. Chi tiết cài đặt

1 Quy trình Mã hóa & Chia sẻ (ECDH + AES) - phần cốt lõi của ứng dụng.

- Tạo khóa (Registration):
  - Client tạo cặp khóa ECDH X25519 (PrivKey_A, PubKey_A).
  - Gửi PubKey_A lên Server lưu trữ. PrivKey_A lưu bí mật tại file local (username.pem).
- Upload & Mã hóa (Create Note):
  - Tạo khóa ngẫu nhiên K_File (32 bytes).
  - Mã hóa File:
  - Để chính mình đọc lại được, Client A lấy PubKey_A (của chính mình), kết hợp PrivKey_A → SharedSecret (thực ra là ECDH với chính mình hoặc derived key).
  - Mã hóa khóa file:
  - Gửi
- Chia sẻ cho B (Share Note):
  - Client A tải PubKey_B từ Server.
  - Client A tính SharedSecret_AB = ECDH(PrivKey_A, PubKey_B).
  - Client A giải mã K_File (dùng khóa của mình).
  - Client A mã hóa K_File bằng SharedSecret_AB:
  - Gửi K_{EncB} lên Server cho B.

2. Tối ưu hóa (Optimization)

- SQLite (Pure Go): Sử dụng driver không cần CGO (modernc.org/sqlite) giúp việc biên dịch và chạy trên Windows dễ dàng hơn, không cần cài GCC.
- Write-Ahead Logging (WAL): Cấu hình PRAGMA journal_mode=WAL; giúp tăng hiệu năng xử lý đồng thời, cho phép đọc/ghi song song.
- Connection Pooling: Sử dụng DSN parameters (busy_timeout=5000) để quản lý timeout kết nối hiệu quả, tránh lỗi "database locked" khi tải cao.
- Indexing: Đánh chỉ mục cho các trường owner_id và share_token để tăng tốc độ truy vấn.
- JWT Caching: Server xác thực Stateless, không cần query DB để check session ID mỗi lần req (tuy nhiên vẫn check user existence).

IV. Thách thức và giải pháp: Trình bày các vấn đề đã gặp và cách giải quyết.

1 Chuyển đổi từ RSA sang ECDH

- Vấn đề: RSA mã hóa trực tiếp được khóa nhỏ, nhưng ECDH chỉ tạo ra Shared Secret chứ không mã hóa trực tiếp.

- Giải pháp: Sử dụng cơ chế Key Wrapping. Dùng ECDH để tạo Shared Secret, sau đó Hash(SharedSecret) để làm khóa AES dùng để mã hóa cái "File Key". Đây là mô hình Hybrid Encryption chuẩn.

2 Bảo mật Mật khẩu

- Vấn đề: SHA-256 thuần túy dễ bị tấn công bởi Rainbow Table.
- Giải pháp: Triển khai Salt. Mỗi user có một chuỗi Salt ngẫu nhiên 16-byte lưu trong DB. Khi hash, chuỗi này được nối vào password.

3 Đồng bộ hóa (Concurrency)

- Vấn đề: SQLite mặc định lock toàn bộ database khi ghi, gây lỗi khi stress test nhiều user.
- Giải pháp:
  - Chuyển sang WAL Mode (Write-Ahead Logging).
  - Cấu hình Busy Timeout để driver tự động chờ (backoff) thay vì fail ngay.

V. Kiểm thử

1. Phương pháp kiểm thử: Kết hợp Automated Testing (Go test framework) và Manual Testing (CLI Client).

2. Kết quả kiểm thử

2.1. Automated Testing

Authentication Tests (auth_test.go)	Đăng ký thành công với ECDH key pair	PASS
	Đăng ký trùng username → 409 Conflict	PASS
	Validate mật khẩu yếu (5 sub-tests: độ dài, chữ hoa/thường, số, ký tự đặc biệt)	PASS
	Invalid JSON → 400 Bad Request	PASS
	Đăng nhập thành công → JWT token	PASS
	Đăng nhập sai credentials → 401 Unauthorized	PASS
	Password được hash (SHA-256 + Salt) trong DB	PASS
	Invalid/Empty token → 401 Unauthorized	PASS
	Concurrent registration (thread-safe)	PASS
Encryption Tests (encryption_test.go)	AES-GCM encryption/decryption integrity	PASS
	Validate AES key size (256-bit)	PASS
	IV uniqueness (mỗi lần encrypt khác nhau)	PASS
E2E Encryption Tests (e2e_encryption_test.go)	End-to-end note encryption workflow	PASS
	Shared note có encrypted keys riêng cho mỗi user	PASS

	Multiple users share & decrypt cùng 1 note	PASS
	Key rotation (re-encrypt với key mới)	PASS
Access Control Tests (access_control_test.go)	User A share note cho User B → B decrypt thành công	PASS
	Generate share link với token	PASS
	Access public note qua share link (không cần login)	PASS
	Expired share link → 403 Forbidden	PASS
	Unauthorized user access → 403 Forbidden	PASS
Integration Tests (integration_test.go)	Full workflow: Register → Login → Upload → Share → Download	PASS
	Concurrent note creation (20 notes đồng thời, race condition test)	PASS
	Stress test (10 users × 5 notes = 50 operations, concurrency + retry logic)	PASS

2.2. Manual Test

Authentication Tests	Thành công tạo User mới, file .pem được lưu Đăng nhập sai pass/username trả về lỗi 401 Đăng nhập đúng trả về JWT Token	PASS
Encryption Tests	Upload file text/binary, tải về giải mã trùng khớp SHA-256 với file gốc Không có Private Key (.pem) không thể giải mã	PASS
	Concurrent note creation (20 notes đồng thời, race condition test)	PASS
Share	Tạo link, mở trên client khác (chọn menu 3) tải thành công mà không cần login	PASS
	User A chia sẻ cho User B. User B đăng nhập, tải file và giải mã thành công User C (không được share) truy cập bị lỗi 403	PASS

VI. Cải tiến tương lai: Đề xuất các cải tiến hoặc tính năng bổ sung.

1. Cải tiến về Bảo mật

Bảo mật là yếu tố then chốt để bảo vệ dữ liệu nhạy cảm của người dùng. Việc chỉ sử dụng username/password hiện tại chưa đủ trước các tấn công hiện đại. Triển khai Multi-Factor Authentication (MFA) giúp tăng cường bảo vệ tài khoản bằng cách yêu cầu người dùng cung cấp thêm mã xác thực thời gian thực (TOTP), giảm rủi ro khi password bị lộ. Song song, Audit Logging cung cấp khả năng theo

dõi chi tiết mọi hành động quan trọng, giúp phát hiện bất thường, hỗ trợ điều tra sự cố bảo mật. Cuối cùng, Rate Limiting nâng cao ngăn chặn brute-force và DDoS attack, bảo vệ hệ thống khỏi lưu lượng bất thường, đồng thời có thể tùy chỉnh quota cho từng nhóm người dùng, cân bằng giữa bảo mật và trải nghiệm.

2. Cải tiến về Hiệu năng

Hiệu năng tốt đảm bảo trải nghiệm người dùng mượt mà và khả năng chịu tải của hệ thống. Việc thêm Caching Layer sử dụng Redis giúp giảm tải database, cải thiện thời gian phản hồi từ hàng trăm mili giây xuống hàng chục mili giây. Database Optimization bằng indexing, partitioning và giám sát slow query giúp tối ưu truy vấn trên các bảng lớn, tránh bottleneck. Async Processing tách các tác vụ nặng ra khỏi luồng chính, cho phép xử lý song song trong background queue, vừa tăng tốc phản hồi cho người dùng vừa mở rộng khả năng scale khi lượng dữ liệu hoặc thông báo tăng lên.

3. Cải tiến về Tính năng

Mở rộng tính năng làm tăng giá trị và trải nghiệm người dùng. Note Versioning giúp lưu lịch sử chỉnh sửa, cho phép rollback và quản lý conflict khi nhiều người cùng chỉnh sửa. Advanced Sharing cung cấp quyền chi tiết (read, write, reshare) giúp kiểm soát chia sẻ dữ liệu linh hoạt. Full-Text Search với engine như Elasticsearch cho phép tìm kiếm toàn nội dung note, bao gồm fuzzy search, nâng cao khả năng tìm kiếm của người dùng. Thêm File Attachments tăng khả năng lưu trữ tài liệu đa phương tiện, với các biện pháp bảo mật và tối ưu lưu trữ, phù hợp với nhu cầu làm việc đa dạng.

4. Cải tiến về Monitoring & Operations

Hoạt động giám sát và vận hành chặt chẽ giúp hệ thống ổn định và giảm downtime. Metrics & Alerting theo dõi chỉ số hiệu suất và cảnh báo khi vượt ngưỡng, giúp phát hiện vấn đề sớm. Health Checks đảm bảo hệ thống phân phối traffic chính xác, kết hợp auto-healing để xử lý instance gặp sự cố. Backup & Recovery bảo vệ dữ liệu, hỗ trợ restore nhanh chóng khi có sự cố, bao gồm point-in-time recovery và lưu trữ đa vùng, giảm rủi ro mất mát dữ liệu do thảm họa.

5. Cải tiến về User Experience

r như Notion hoặc Obsidian.Trải nghiệm người dùng là yếu tố then chốt giữ chân và tăng tương tác. Mobile App với offline mode và biometric authentication cho phép người dùng thao tác mọi lúc, mọi nơi, tăng tính tiện lợi. Real-time Collaboration giúp nhiều người cùng chỉnh sửa note đồng thời, sử dụng công nghệ như WebSocket, Operational Transform hoặc CRDT để tránh conflict, tạo cảm giác tương tác tức thời như Google Docs. Rich Text Editor nâng cao khả năng trình bày

nội dung, hỗ trợ Markdown, code syntax, hình ảnh và bảng biểu, đáp ứng nhu cầu đa dạng từ casual users đến technical users.

6. Cải tiến về Scalability

Khả năng mở rộng giúp hệ thống phục vụ số lượng lớn người dùng mà không ảnh hưởng hiệu năng. Microservices Architecture tách chức năng thành các service độc lập, cho phép scale riêng từng phần, tăng tính linh hoạt, giảm rủi ro lỗi lan truyền. Horizontal Scaling thêm instances để cân bằng tải, kết hợp auto-scaling và session lưu trong Redis giúp hệ thống thích ứng với lưu lượng thay đổi. CDN Integration tối ưu phân phối nội dung và API response, giảm latency cho người dùng toàn cầu, đồng thời cung cấp lớp bảo vệ DDoS miễn phí, nâng cao trải nghiệm và độ ổn định của hệ thống.

Tham khảo