ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

BÁO CÁO

ỨNG DỤNG CHIA SẺ GHI CHÚ

Lớp: Nhập môn Mã hóa - Mật mã

Thành viên:

Trần Thanh Thông - 23120091

Nguyễn Hải Đăng - 23120027

Lê Hải Đăng – 23122005

Mục lục

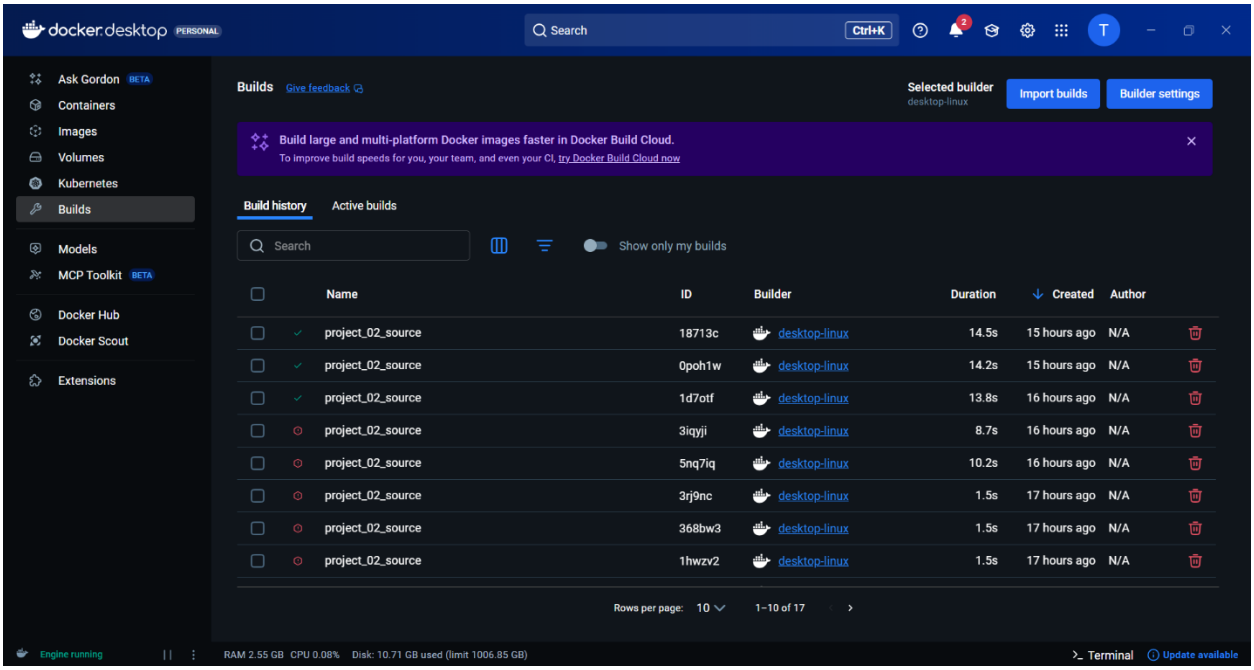
1. Tổng quan ứng dụng

1. 1. Mục tiêu

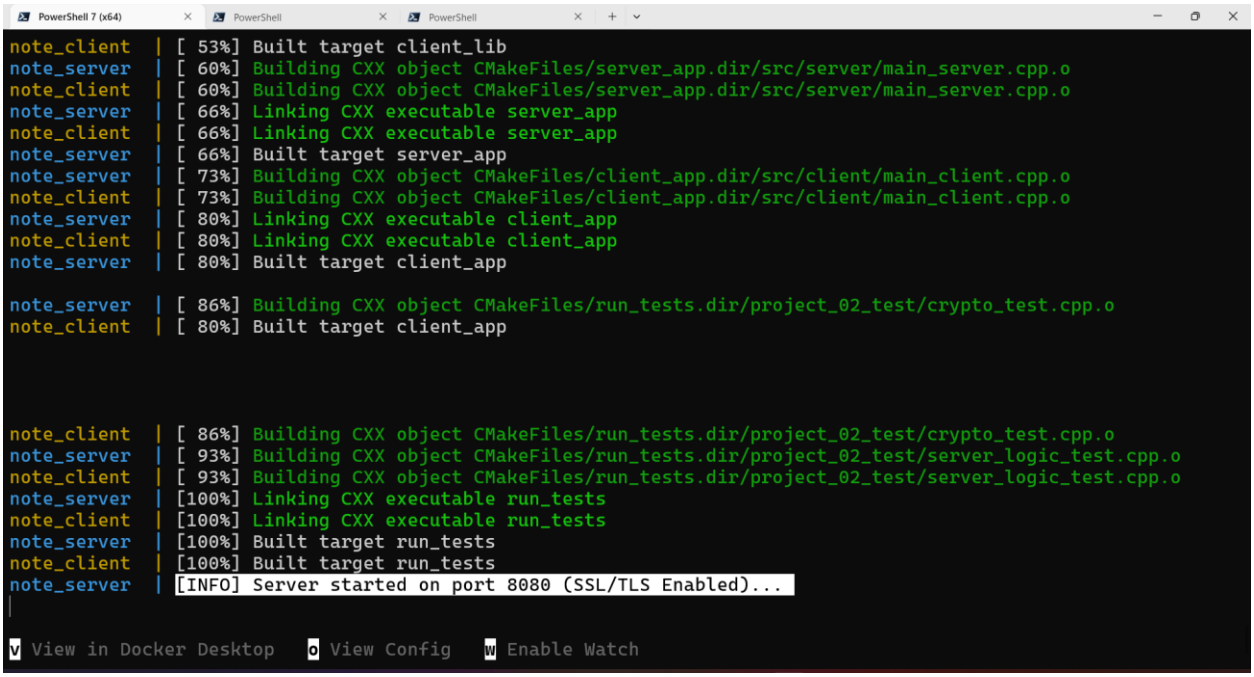
Mục tiêu chính của ứng dụng là xây dựng một hệ thống chia sẻ ghi chú và tập tin an toàn dựa trên mô hình Client-Server, đảm bảo rằng ngay cả máy chủ (Server) cũng không thể đọc được nội dung dữ liệu của người dùng.

1. 2. Cách chạy chương trình

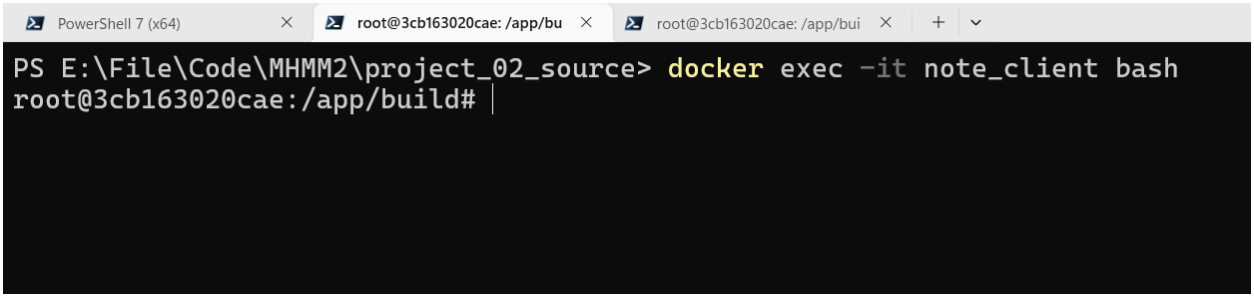
- Cài đặt Docker Desktop:
  - Hướng dẫn cài đặt Docker Desktop trên Window
  - Hướng dẫn cài đặt Docker Desktop trên MacOS
- Chạy Docker Desktop (đảm bảo phải góc trái dưới có chữ Engine Running)



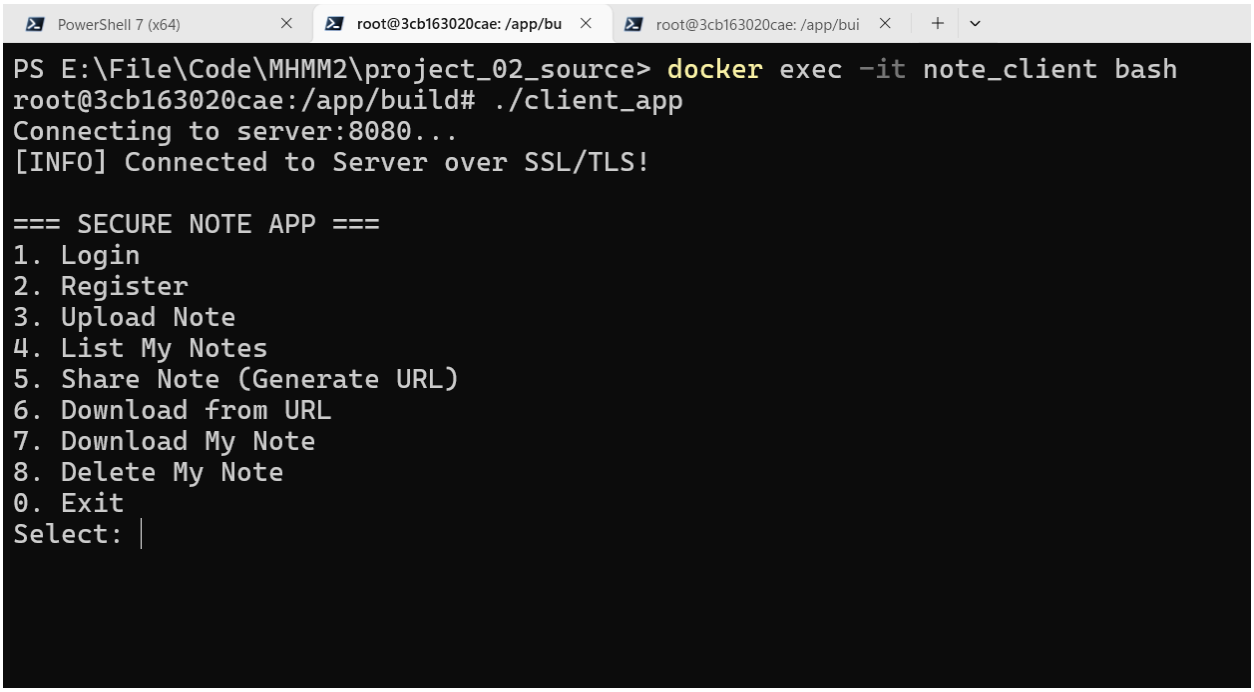
- Mở một cửa sổ terminal tại thư mục **project_02_source** và chạy lệnh **docker-compose up –build.** Nếu thấy dòng **[INFO] Server started on port 8080 (SSL/TLS Enabled)...** như hình bên dưới là đã build thành công và server cũng đã chạy.



- Sau khi chạy thành công, có thể mở các cửa sổ terminal khác trong **project_02_source** để chạy ứng dụng (mỗi một cửa sổ có thể đăng nhập tài khoản khác nhau để test khả năng chia sẻ).
- Tại các terminal, chạy lệnh **docker exec -it note_client bash** để chạy client. Kết quả như hình bên dưới.



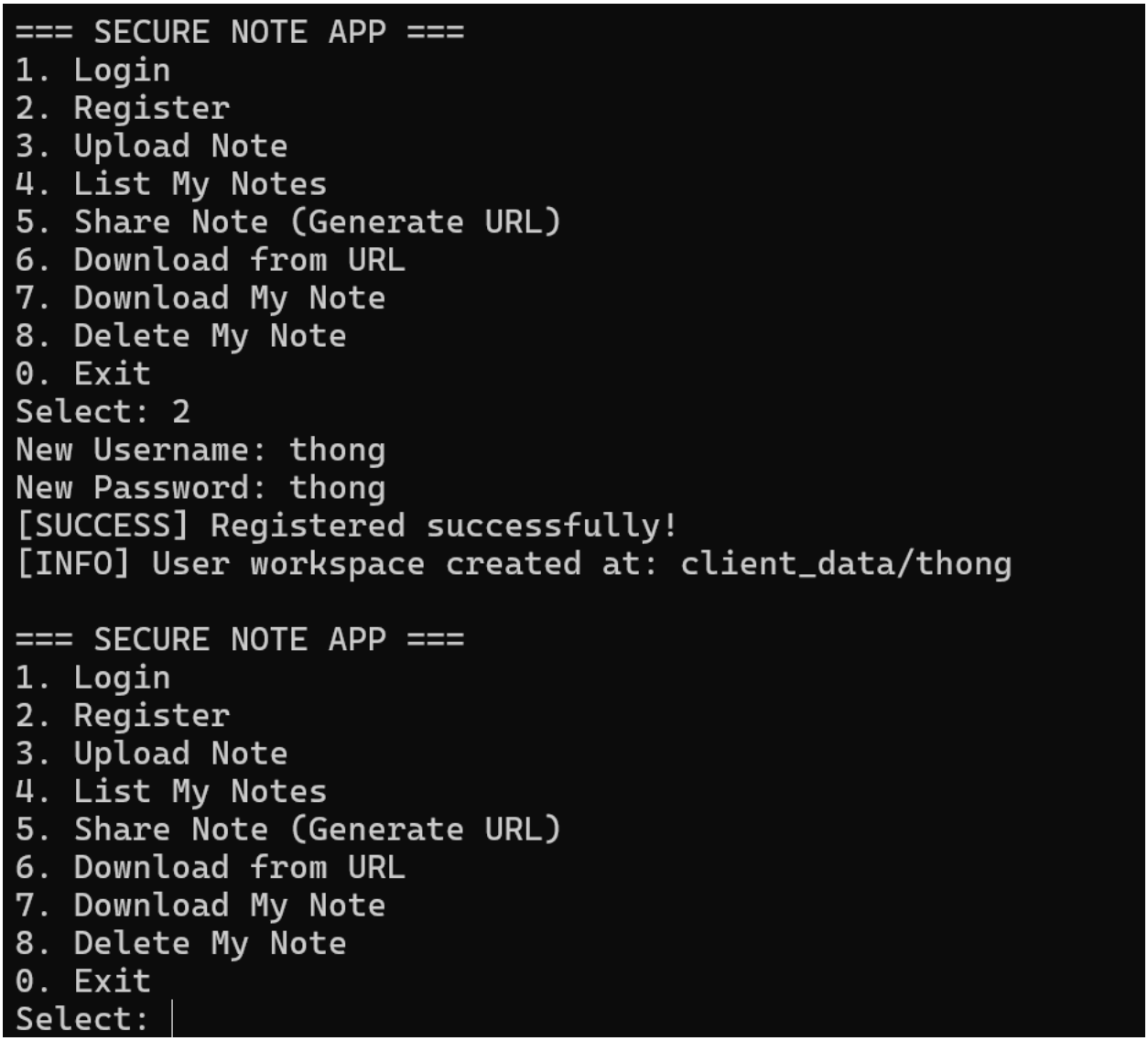
- Sau khi thực hiện xong lệnh **docker exec -it note_client bash** thì chạy lệnh **./client_app** để chạy app client.



1. 3. Chi tiết các chức năng

1.3.1. Register

Khi chọn chức năng Register, người dùng sẽ được yêu cầu nhập vào username và password. Sau khi tạo xong chương trình sẻ tạo ra một folder **client_data/<username>**, thư mục này sẽ được ánh xạ sang máy thật tại đường dẫn **client_data/<username>**. Trong **client_data/<username>** có chứa hai folder **data** và **downloads**. Folder data là folder chứa các data chính của user đó, có thể thêm file từ máy thật vào đây để upload note. Fodler **downloads** sẽ chứa tất cả các file tải về.



1.3.2. Login

Nhập username và password để đăng nhập.

```
=== SECURE NOTE APP ===
1. Login
2. Register
3. Upload Note
4. List My Notes
5. Share Note (Generate URL)
6. Download from URL
7. Download My Note
8. Delete My Note
0. Exit
Select: 1
Username: thong
Password: thong
[SUCCESS] Logged in! Token saved.
[INFO] User workspace created at: client_data/thong
[INFO] Private key loaded for E2EE.

=== SECURE NOTE APP ===
1. Login
2. Register
3. Upload Note
4. List My Notes
5. Share Note (Generate URL)
6. Download from URL
7. Download My Note
8. Delete My Note
0. Exit
Select:
```

1.3.3. Upload Note

Để có thể upload file, có thể thêm file vào thư mục **client_data/<username>/data/** trên máy thật. Khi Upload Note chương trình sẽ hiện đường dẫn folder data để người dùng dễ copy đường dẫn.

```
=== SECURE NOTE APP ===
1. Login
2. Register
3. Upload Note
4. List My Notes
5. Share Note (Generate URL)
6. Download from URL
7. Download My Note
8. Delete My Note
0. Exit
Select: 3
Your workspace is at client_data/thong/data/

--- FILES IN: client_data/thong/data/ ---
- thong.priv (32 bytes)
-----------------------------------
Enter filename path to upload (e.g. client_data/thong/data/a.txt): client_data/thong/data/thong.priv
[SUCCESS] Note uploaded with ID: 49YBn8upbn07

=== SECURE NOTE APP ===
1. Login
2. Register
3. Upload Note
4. List My Notes
5. Share Note (Generate URL)
6. Download from URL
7. Download My Note
8. Delete My Note
0. Exit
Select:
```

1.3.4. List My Notes

```
=== SECURE NOTE APP ===
1. Login
2. Register
3. Upload Note
4. List My Notes
5. Share Note (Generate URL)
6. Download from URL
7. Download My Note
8. Delete My Note
0. Exit
Select: 4

=== MY WORKSPACE ===
Note ID         | Filename                       | Upload Time
-------------------------------------------------------------------------------
49YBn8upbn07    | thong.priv                     | Thu Dec 11 08:59:56 2025
-------------------------------------------------------------------------------
```

1.3.5. Share Note (Generate URL)

Chương trình hiện ra danh sách các note, và người dùng sẽ nhập noteid, username của người nhận (cần phải tạo một user khác để chia sẻ). Nhập thời hiệu lực của link, số lần tải tối đa. Sao chép URL và sang terminal khác đăng nhập và chọn **Download from URL**

```
3. Upload Note
4. List My Notes
5. Share Note (Generate URL)
6. Download from URL
7. Download My Note
8. Delete My Note
0. Exit
Select: 5

=== MY WORKSPACE ===
Note ID         | Filename                       | Upload Time
-------------------------------------------------------------------------------
49YBn8upbn07    | thong.priv                     | Thu Dec 11 08:59:56 2025
-------------------------------------------------------------------------------

=== SHARE NOTE (End-to-End Encrypted) ===
Enter Note ID to share: 49YBn8upbn07
Enter Recipient Username: test
Link expiration (hours): 1
Max views (e.g. 1): 2
[INFO] Encrypting key for test...

[SUCCESS] Encrypted Link Created!
--------------------------------------------------
URL: securenote://ihr2zRzlFdlz
--------------------------------------------------
Settings: Expire in 1h, Max views: 2
Send this URL to user 'test'.
```

1.3.6. Download from URL

Nhập URL và tải về.

```
=== SECURE NOTE APP ===
1. Login
2. Register
3. Upload Note
4. List My Notes
5. Share Note (Generate URL)
6. Download from URL
7. Download My Note
8. Delete My Note
0. Exit
Select: 6
Paste the URL (securenote://...): securenote://ihr2zRzlFdlz
[INFO] Note sent by: thong
[SUCCESS] Decrypted & Saved to: client_data/test/downloads/thong.priv
```

1.3.7. Download My Note

Nhập Noteid và tải vào downloads.

```
=== SECURE NOTE APP ===
1. Login
2. Register
3. Upload Note
4. List My Notes
5. Share Note (Generate URL)
6. Download from URL
7. Download My Note
8. Delete My Note
0. Exit
Select: 7

=== MY WORKSPACE ===
Note ID         | Filename                       | Upload Time
--------------------------------------------------------------------------------
49YBn8upbn07    | thong.priv                     | Thu Dec 11 08:59:56 2025
--------------------------------------------------------------------------------
Enter Note ID: 49YBn8upbn07
[SUCCESS] File saved as: client_data/thong/downloads/thong.priv
```

1.3.8. Delete My Note

Nhập Noteid và xác nhận xóa

```
1. Login
2. Register
3. Upload Note
4. List My Notes
5. Share Note (Generate URL)
6. Download from URL
7. Download My Note
8. Delete My Note
0. Exit
Select: 8

=== MY WORKSPACE ===
Note ID         | Filename                       | Upload Time
--------------------------------------------------------------------------------
49YBn8upbn07    | thong.priv                     | Thu Dec 11 08:59:56 2025
--------------------------------------------------------------------------------

=== DELETE NOTE ===
Enter Note ID to delete: 49YBn8upbn07
Are you sure you want to delete 49YBn8upbn07? (y/n): y
[SUCCESS] Note deleted from server.
[INFO] Removed encryption key from local storage.
```

2. Thiết kế và kiến trúc

2. 1. Công nghệ sử dụng

Dự án được phát triển dựa trên ngôn ngữ C++ chuẩn, kết hợp với các thư viện mã nguồn mở uy tín về bảo mật và xử lý dữ liệu. Môi trường triển khai được tiêu chuẩn hóa thông qua Containerization.

2.1.1. Ngôn ngữ và trình biên dịch

- **Ngôn ngữ:** C++17 (Sử dụng các tính năng hiện đại như std::filesystem, std::optional, smart pointers).
- **Trình biên dịch (Compiler):**
  - **GCC** (GNU Compiler Collection) phiên bản 9.4+ (Mặc định trên môi trường Docker/Linux).

2.1.2. Thư viện và Framework

- **OpenSSL (Phiên bản 3.0):** Thư viện cốt lõi cung cấp các thuật toán mã hóa (AES-256-GCM, SHA-256), trao đổi khóa (Diffie-Hellman/ECDH) và thiết lập kênh truyền bảo mật (TLS/SSL).
- **nlohmann/json (Phiên bản 3.11.2):** Thư viện xử lý định dạng JSON, dùng để đóng gói metadata, cấu trúc gói tin (packet) gửi nhận giữa Client và Server, và lưu trữ dữ liệu cấu hình.
- **Google Test (GTest) (Phiên bản 1.14.0):** Framework kiểm thử đơn vị (Unit Test) để đảm bảo tính chính xác của các module Logic và Crypto.
- **POSIX Threads (Pthreads) / std::thread:** Xử lý đa luồng cho Server để phục vụ nhiều Client đồng thời.

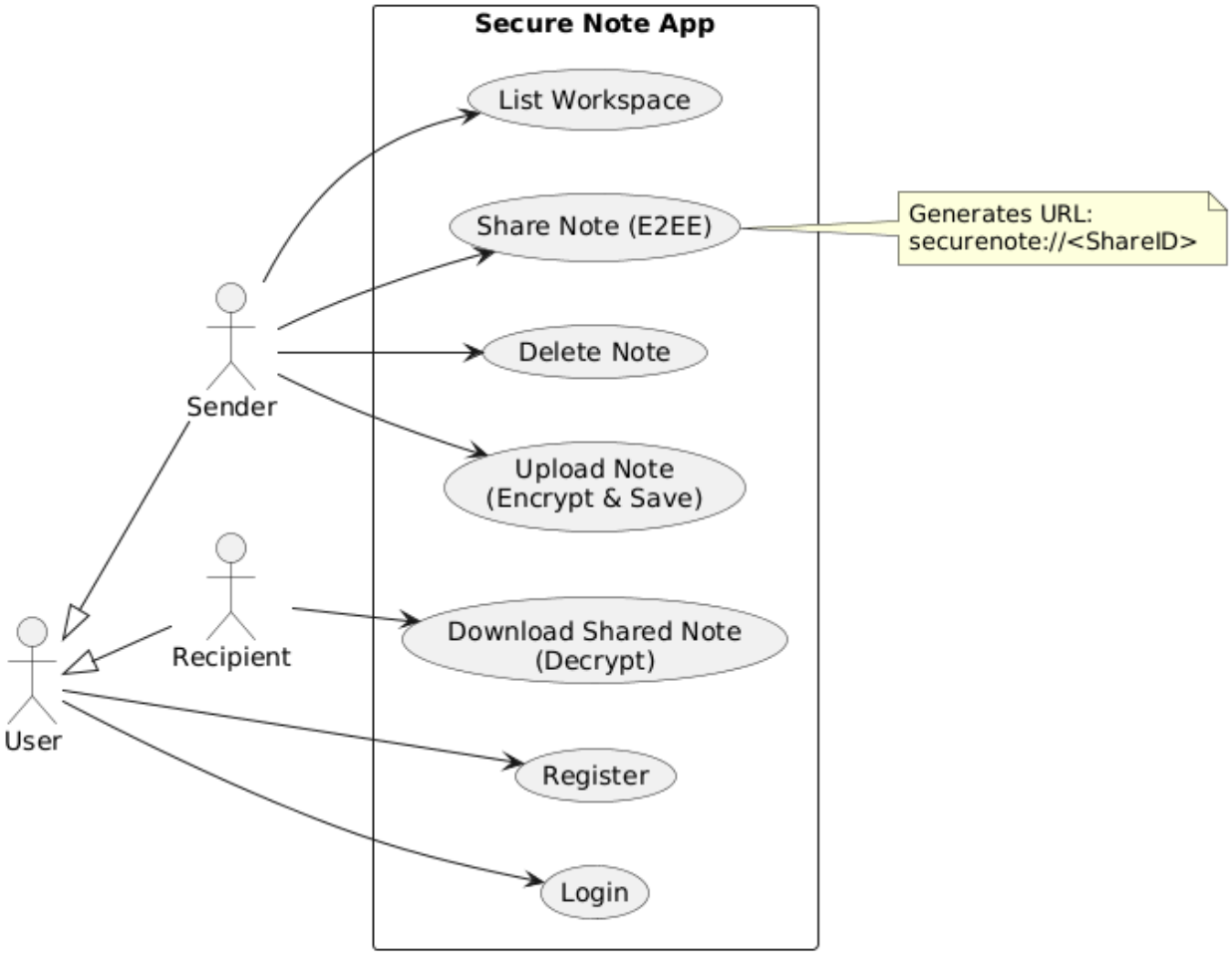
2.1.3. Công cụ quản lý và triển khai

- **CMake (Phiên bản 3.10+):** Hệ thống build system đa nền tảng, quản lý quy trình biên dịch và liên kết thư viện.

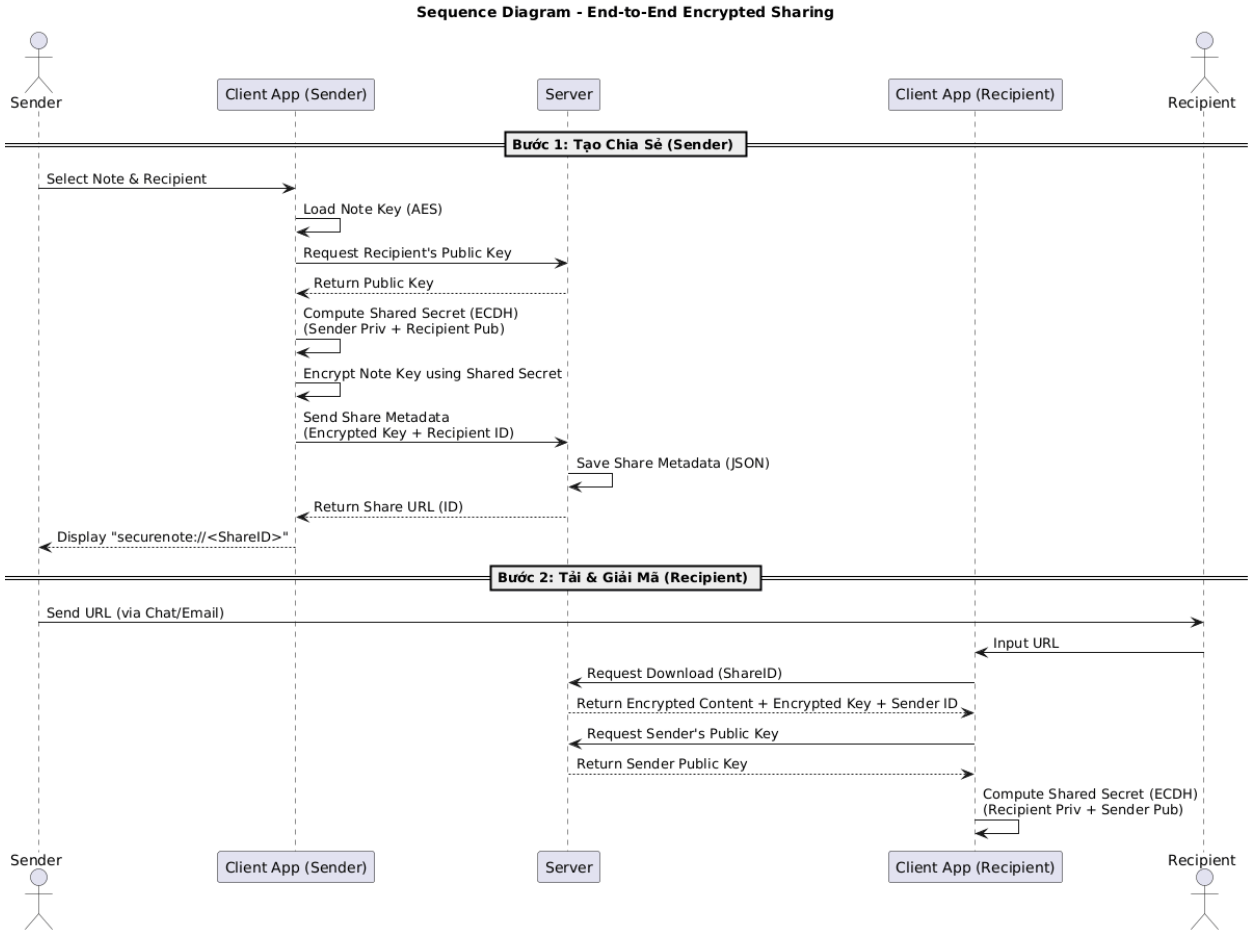
- **Docker & Docker Compose:** Đóng gói toàn bộ ứng dụng và môi trường vào Container, đảm bảo có thể chạy đa nền tảng và có thể tách biệt server và client nếu muốn.

2. 2. Kiến trúc hệ thống

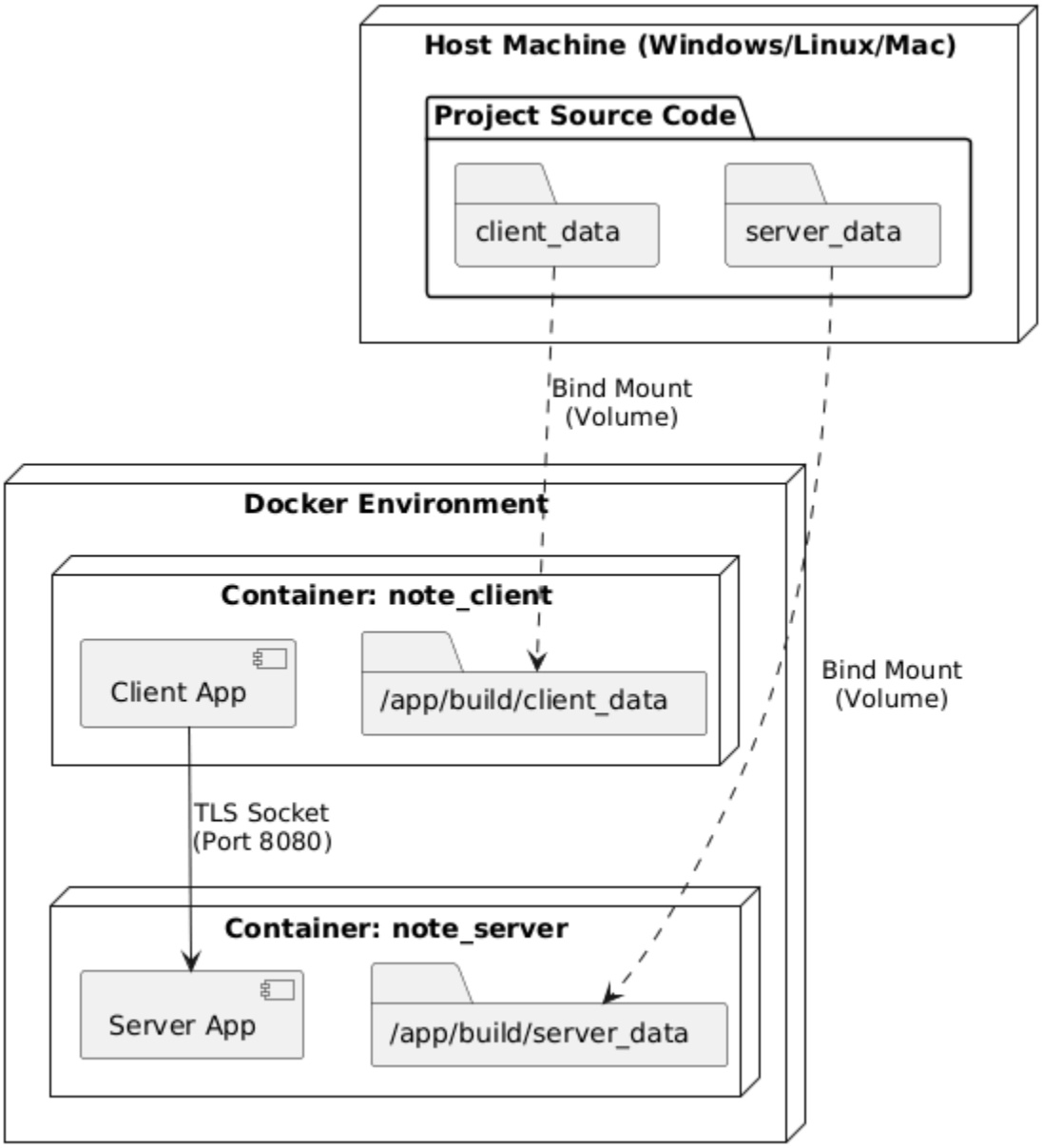
2.2.1. Use case diagram



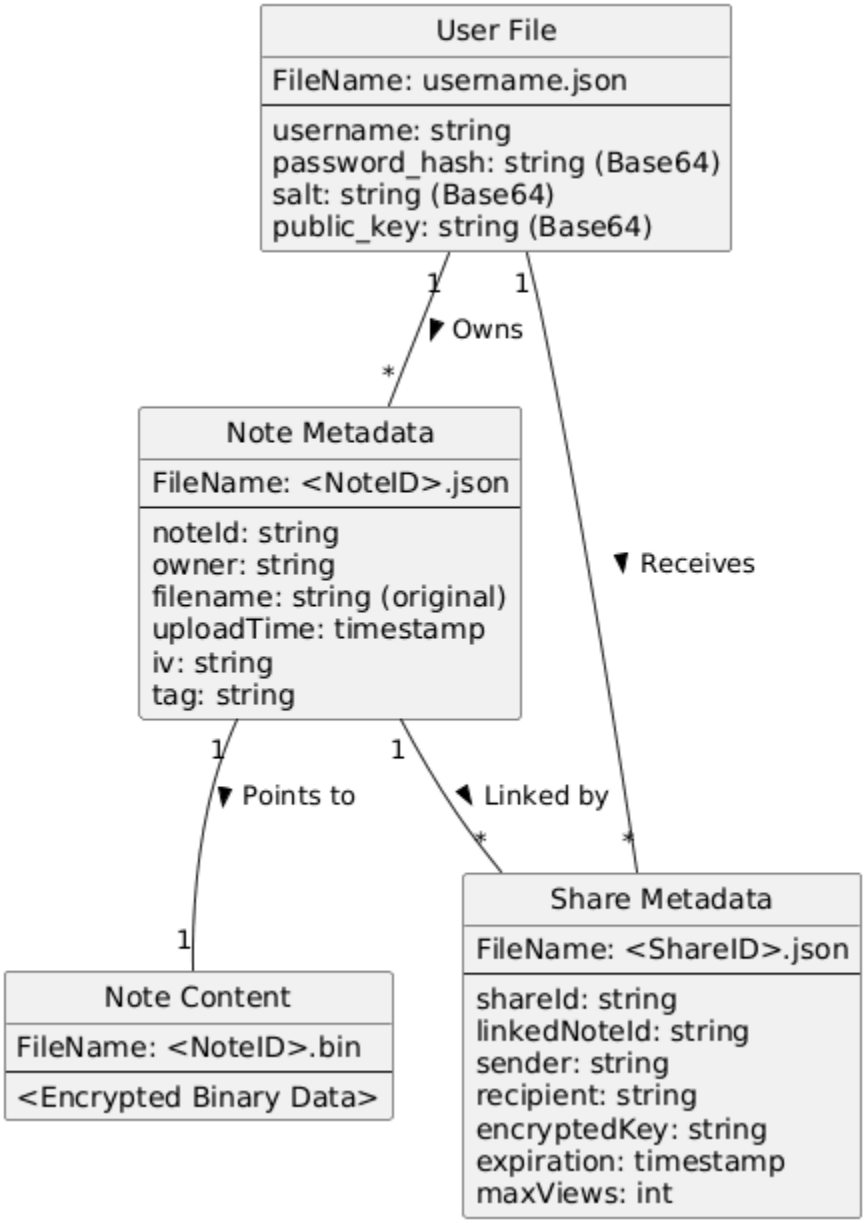
2.2.2. Sơ đồ luồng hoạt động



2.2.3. Deploy diagram



2.2.4. Cấu trúc lưu trữ



2. 3. Mục đích thiết kế

2.3.1. Bảo mật đầu cuối (End-to-End Encryption - E2EE):

Đảm bảo rằng **chỉ người gửi và người nhận đích danh** mới có thể đọc được nội dung ghi chú.

Ngăn chặn Server (hoặc kẻ tấn công chiếm quyền kiểm soát Server) đọc được nội dung gốc. Server chỉ đóng vai trò là kho lưu trữ dữ liệu đã mã hóa và trung chuyển thông tin.

2.3.2. Tính toàn vẹn và xác thực dữ liệu:

Sử dụng chế độ mã hóa AES-GCM (Galois/Counter Mode) để không chỉ mã hóa dữ liệu mà còn tạo ra thẻ xác thực (Authentication Tag). Điều này giúp Client phát hiện ngay lập tức nếu dữ liệu bị can thiệp hoặc sửa đổi trên đường truyền hoặc tại nơi lưu trữ.

2.3.3. Kiểm soát truy cập linh hoạt:

Hệ thống không sử dụng một khóa chung cho tất cả. Mỗi ghi chú có một khóa riêng (Note Key).

Khi chia sẻ, Note Key được mã hóa bằng khóa bí mật chung (Shared Secret) được sinh ra từ giao thức Diffie-Hellman giữa hai người dùng cụ thể. Điều này đảm bảo tính riêng tư tuyệt đối cho từng phiên chia sẻ.

2.3.4. Khả năng mở rộng và độc lập:

Thiết kế tách biệt hoàn toàn giữa Logic xử lý (Business Logic) và Giao diện/Mạng (UI/Network) giúp dễ dàng nâng cấp, bảo trì và kiểm thử (Unit Testing) từng thành phần riêng biệt.

2. 4. Các thành phần chính

2.4.1. Phía Client (Client Application)

ClientApp (UI & Flow Controller):

Là giao diện dòng lệnh (CLI) tương tác với người dùng.

Điều phối luồng hoạt động: Nhận lệnh từ user -> Gọi CryptoManager để xử lý dữ liệu -> Gọi Network để gửi đi.

Quản lý LocalKeyStore (kho khóa cục bộ) để lưu trữ khóa giải mã của các file do chính user sở hữu.

CryptoManager (Client-side):

Thực hiện toàn bộ các tác vụ mật mã nặng tại máy khách.

Sinh khóa ngẫu nhiên (CSPRNG), mã hóa file (AES-256), tạo cặp khóa bất đối xứng (Diffie-Hellman).

Đảm bảo dữ liệu rời khỏi Client là dữ liệu rác (đã mã hóa).

Local Storage (File System):

Lưu trữ private key của người dùng.

Lưu trữ các file đã tải về và giải mã (downloads/).

2.4.2. Phía Server (Server Application)

Server (Network Handler):

Lắng nghe kết nối qua cổng bảo mật (TLS Socket).

Phân luồng (Threading) để xử lý nhiều kết nối đồng thời.

Điều hướng các gói tin (Packet Dispatcher) tới các Manager tương ứng dựa trên CommandType.

AuthManager:

Xử lý đăng ký và đăng nhập.

Lưu trữ mật khẩu dưới dạng Hash an toàn (PBKDF2 với Salt).

Quản lý Public Key của người dùng để phục vụ việc chia sẻ.

NoteManager:

Quản lý việc lưu trữ vật lý các file mã hóa (server_data/note/).

Quản lý Metadata (JSON) của ghi chú (Owner, ID, Timestamp).

Xử lý logic chia sẻ: Tạo và kiểm tra các liên kết chia sẻ (Share Link), kiểm soát thời hạn (expiration) và số lượt xem (max views).

CryptoManager (Server-side):

Phiên bản rút gọn dùng cho Server, chủ yếu thực hiện các tác vụ Hashing mật khẩu và sinh Token phiên làm việc.

3. Chi tiết cài đặt

3. 1. Lưu trữ dữ liệu riêng biệt

Kỹ thuật: Áp dụng mẫu thiết kế Constructor Injection cho các lớp quản lý (AuthManager, NoteManager). Thay vì sử dụng đường dẫn thư mục cứng, các lớp này nhận đường dẫn gốc (root path) thông qua Constructor.

Lý do: Để tách biệt môi trường chạy thật và môi trường kiểm thử. Trong quá trình code các file tạo ra trong lúc kiểm thử có thể khó kiểm soát.

Kết quả thực tiễn: Cho phép chạy Unit Test song song với Server đang hoạt động mà không gây xung đột dữ liệu hoặc vô tình xóa mất dữ liệu người dùng thật.

3. 2. Tối ưu hóa Docker Layer Caching

Kỹ thuật: Trong Dockerfile và quy trình build, mã nguồn của các thư viện phụ thuộc (Google Test, nlohmann/json) được cấu hình để tải về và cache lại trước khi copy mã nguồn chính vào.

Kết quả thực tiễn: Giảm thời gian build lại từ vài phút xuống còn vài giây khi chỉ có thay đổi nhỏ trong mã nguồn, do Docker không phải tải và biên dịch lại các thư viện không thay đổi.

4. Thách thức và giải pháp

4. 1. Thách thức về đường dẫn và quyền hạn trong Docker

- Vấn đề:

Khi chạy trên Windows và ánh xạ (mount) thư mục mã nguồn vào Linux Container, thường xuyên gặp lỗi do cấu hình không đúng.

- Giải pháp:

Cấu hình lại docker-compose.yml để tách biệt thư mục build (/app/build) ra khỏi cơ chế đồng bộ volume của Windows (sử dụng Anonymous Volume cho /app/build).

Chỉ định rõ ràng working_dir và sử dụng lệnh rm chọn lọc (chỉ xóa cache CMake) thay vì rm -rf * toàn bộ thư mục.

4. 2. Thách thức về Kiểm thử Logic Server

Vấn đề: Các hàm logic của Server (SaveNote, Register) thao tác trực tiếp với File System. Việc viết Test Case gặp khó khăn khi phải dọn dẹp file rác sau mỗi lần chạy test để tránh ảnh hưởng đến kết quả lần sau.

Giải pháp: Sử dụng thư viện std::filesystem để tạo môi trường Sandbox tạm thời (server_data_test/) trong SetUp() và dọn dẹp trong TearDown() của Google Test Framework. Đảm bảo môi trường test luôn sạch (clean state).

5. Kiểm thử

5. 1. Phương pháp và công cụ

Framework: Sử dụng Google Test (GTest) - Thư viện kiểm thử C++ tiêu chuẩn công nghiệp.

Môi trường: Test được chạy trực tiếp bên trong Docker Container của Server để đảm bảo môi trường giống hệt môi trường Production.

Build System: Tích hợp enable_testing() và add_test() trong CMake để tự động phát hiện và biên dịch test case.

5. 2. Các kịch bản kiểm thử

5.2.1. Xác thực

Tên kịch bản	Mô tả chi tiết	Kết quả mong đợi
User Registration (Happy Path)	Đăng ký tài khoản mới với Username chưa tồn tại và Password hợp lệ.	Đăng ký thành công, file dữ liệu người dùng được tạo trên Server.
Duplicate User Prevention	Cố gắng đăng ký một tài khoản với Username đã tồn tại trong hệ thống.	Đăng ký thất bại, hệ thống báo lỗi "User already exists".
Login Success	Đăng nhập với Username và Password đúng.	Đăng nhập thành công, Server trả về Token phiên làm việc (Session Token).
Login Failure	Đăng nhập với mật khẩu sai hoặc Username không tồn tại.	Đăng nhập thất bại, không có Token nào được cấp.
Token Validation	Gửi yêu cầu kèm theo Token hợp lệ và Token giả mạo.	Token hợp lệ trả về đúng Username. Token giả mạo bị từ chối.

5.2.2. Mã hóa/giải mã

Tên kịch bản	Mô tả chi tiết	Kết quả mong đợi
AES-GCM Encryption/Decryption	Thực hiện mã hóa một chuỗi văn bản bất kỳ, sau đó giải mã bằng đúng khóa (Key) và vector khởi tạo (IV).	Dữ liệu sau khi giải mã phải trùng khớp hoàn toàn với dữ liệu gốc.
Data Integrity Check (Tamper Resistance)	Mã hóa dữ liệu, sau đó cố tình thay đổi 1 bit bất kỳ trong chuỗi mã hóa (ciphertext) hoặc thẻ xác thực (auth tag) trước khi giải mã.	Hàm giải mã phải trả về lỗi (false). Hệ thống từ chối giải mã dữ liệu bị can thiệp.
Wrong Key Decryption	Cố gắng giải mã một gói tin hợp lệ bằng một khóa sai (khóa ngẫu nhiên khác).	Hàm giải mã trả về lỗi.
ECDH Key Exchange Correctness	Mô phỏng hai người dùng (Alice và Bob). Alice dùng PrivateKey của mình và PublicKey của Bob (và ngược lại) để tính toán khóa bí mật chung (Shared Secret).	Khóa chung do Alice tính và Bob tính phải giống hệt nhau (Bit-exact match).

Secure RNG Uniqueness	Gọi hàm sinh số ngẫu nhiên an toàn (CSPRNG) liên tiếp nhiều lần.	Các chuỗi byte sinh ra không được trùng lặp.
Password Hashing (PBKDF2)	Hash mật khẩu với Salt ngẫu nhiên và kiểm tra lại (Verify).	Quá trình verify trả về true với đúng mật khẩu và false với mật khẩu sai.

5.2.3. Giới hạn truy cập

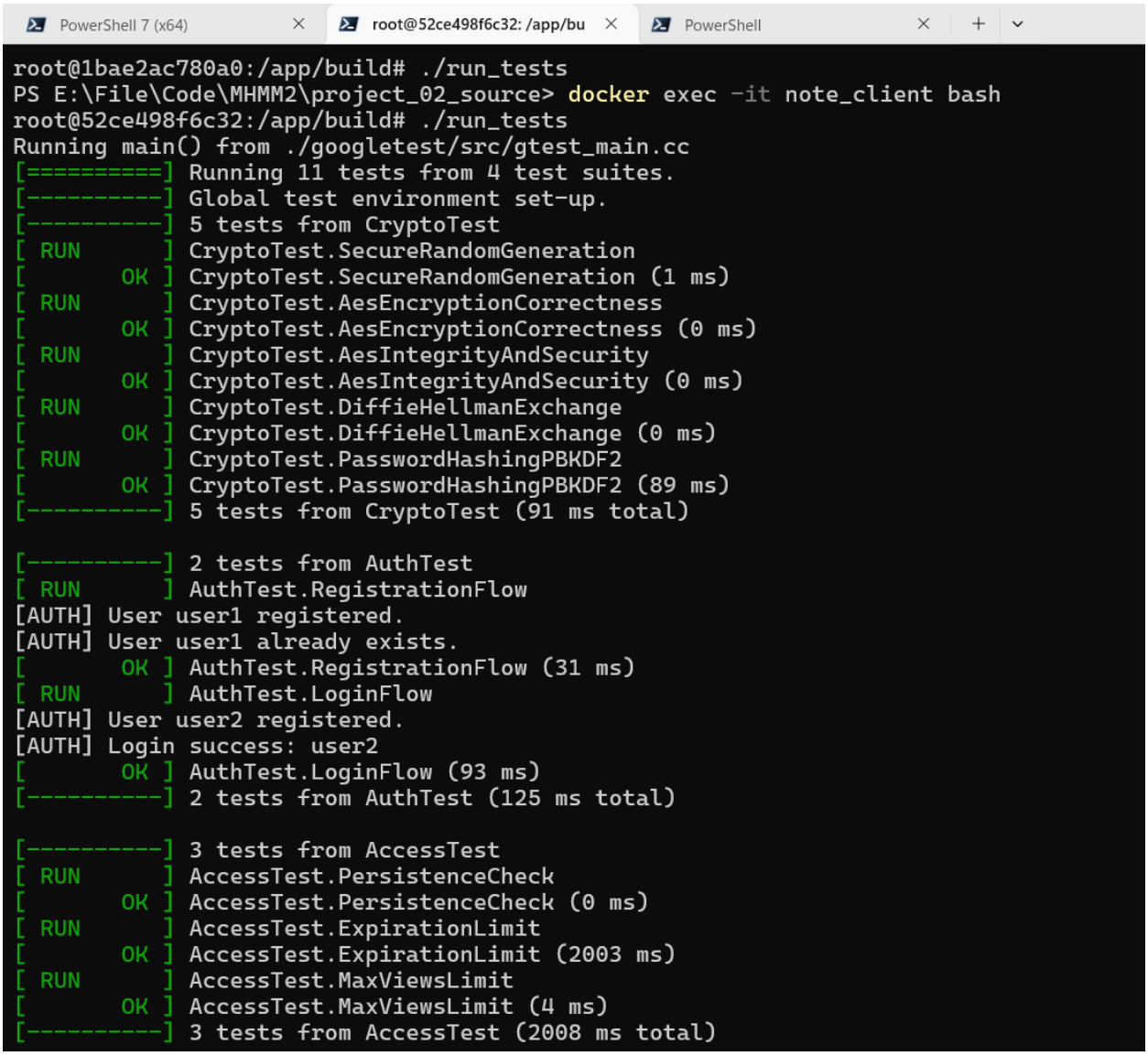
Tên kịch bản	Mô tả chi tiết	Kết quả mong đợi
Persistence Check	Thực hiện lưu một ghi chú (Save Note) lên Server.	File vật lý (.bin và .json) phải được tạo ra và tồn tại trên ổ cứng của Server.
Expiration Time Enforcement	Tạo liên kết chia sẻ có thời hạn 1 giây. Chờ 2 giây sau đó thử truy cập lại liên kết này.	Truy cập thất bại. Server tự động xóa file metadata chia sẻ đã hết hạn.
Max Views Limit Enforcement	Tạo liên kết chia sẻ với giới hạn 2 lượt xem. Thực hiện truy cập lần 1, lần 2 và lần 3.	Lần 1, 2 thành công (file còn tồn tại). Lần 3 thất bại (Server đã xóa file sau khi hết lượt xem thứ 2).

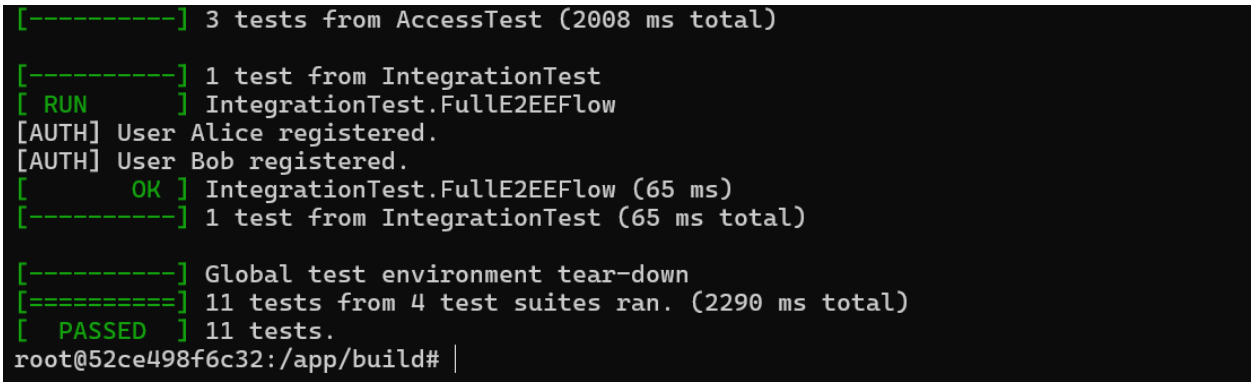
5.2.4. Mã hóa đầu-cuối

Tên kịch bản	Mô tả chi tiết	Kết quả mong đợi
Full E2EE Sharing Flow	1. Alice lấy Public Key của Bob từ Server. 2. Alice tính Shared Secret và mã hóa khóa file (Note Key). 3. Alice gửi metadata lên Server. 4. Bob tải metadata về. 5. Bob dùng Private Key của mình để giải mã ra Note Key. 5. Bob dùng Note Key giải mã nội dung file.	Bob giải mã thành công và nhận được chính xác nội dung file gốc mà Alice đã gửi. Khóa Note Key sau khi Bob giải mã phải trùng khớp với khóa gốc của Alice.

5. 3. Kết quả kiểm thử

Tại các terminal sau khi chạy **docker exec -it note_client bash,** chạy lệnh **./run_tests** để chạy test. Kết quả 100% thành công.


PowerShell 7 (x64)
root@52ce498f6c32: /app/bu
PowerShell
root@1bae2ac780a0:/app/build# ./run_tests
PS E:\File\Code\MHMM2\project_02_source> docker exec -it note_client bash
root@52ce498f6c32:/app/build# ./run_tests
Running main() from ./googletest/src/gtest_main.cc
[==========] Running 11 tests from 4 test suites.
[----------] Global test environment set-up.
[----------] 5 tests from CryptoTest
[RUN] CryptoTest.SecureRandomGeneration
[OK] CryptoTest.SecureRandomGeneration (1 ms)
[RUN] CryptoTest.AesEncryptionCorrectness
[OK] CryptoTest.AesEncryptionCorrectness (0 ms)
[RUN] CryptoTest.AesIntegrityAndSecurity
[OK] CryptoTest.AesIntegrityAndSecurity (0 ms)
[RUN] CryptoTest.DiffieHellmanExchange
[OK] CryptoTest.DiffieHellmanExchange (0 ms)
[RUN] CryptoTest.PasswordHashingPBKDF2
[OK] CryptoTest.PasswordHashingPBKDF2 (89 ms)
[----------] 5 tests from CryptoTest (91 ms total)
[----------] 2 tests from AuthTest
[RUN] AuthTest.RegistrationFlow
[AUTH] User user1 registered.
[AUTH] User user1 already exists.
[OK] AuthTest.RegistrationFlow (31 ms)
[RUN] AuthTest.LoginFlow
[AUTH] User user2 registered.
[AUTH] Login success: user2
[OK] AuthTest.LoginFlow (93 ms)
[----------] 2 tests from AuthTest (125 ms total)
[----------] 3 tests from AccessTest
[RUN] AccessTest.PersistenceCheck
[OK] AccessTest.PersistenceCheck (0 ms)
[RUN] AccessTest.ExpirationLimit
[OK] AccessTest.ExpirationLimit (2003 ms)
[RUN] AccessTest.MaxViewsLimit
[OK] AccessTest.MaxViewsLimit (4 ms)
[----------] 3 tests from AccessTest (2008 ms total)


[----------] 3 tests from AccessTest (2008 ms total)
[----------] 1 test from IntegrationTest
[RUN] IntegrationTest.FullE2EEFlow
[AUTH] User Alice registered.
[AUTH] User Bob registered.
[OK] IntegrationTest.FullE2EEFlow (65 ms)
[----------] 1 test from IntegrationTest (65 ms total)
[----------] Global test environment tear-down
[==========] 11 tests from 4 test suites ran. (2290 ms total)
[PASSED] 11 tests.
root@52ce498f6c32:/app/build#

6. Cải tiến tương lai

Hỗ trợ chia sẻ nhóm người dùng.

Tích hợp thêm chữ ký số để xác minh danh tính người gửi. Sử dụng RSA hoặc ECDSA để ký vào tin nhắn, đảm bảo tính chống chối bỏ (Non-repudiation) - chứng minh chính xác ai là người gửi.

Hỗ trợ mã hóa file dung lượng lớn theo luồng (streaming).

Sử dụng cơ sở dữ liệu như MySQL, Postgresql,… thay vì lưu thông itn vào file.

Client có giao diện.