TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

ĐỒ ÁN MÔN HỌC
NHẬP MÔN CÔNG NGHỆ PHẦN MỀM

Tên đề tài: Phần mềm Note hỗ trợ học tập

Nhóm sinh viên thực hiện : Phạm Hùng Phong - 20225060
Nguyễn Quốc Thái - 20225083
Nguyễn Hoàng Phương - 20225070
Lê Minh Triết - 20220045
Nguyễn Thế Phong – 20224888

Lớp : 154021

HÀ NỘI, 12/2024

Mục lục

Chương 1 Giới thiệu đề tài

1.1 Đặt vấn đề

Trong môi trường học tập hiện đại, sinh viên thường phải tiếp cận với lượng kiến thức khổng lồ từ nhiều nguồn khác nhau như bài giảng, tài liệu, và các nguồn trực tuyến. Việc tổ chức, lưu trữ, và truy cập nhanh chóng các thông tin quan trọng trở thành một thách thức lớn. Những phương pháp ghi chép truyền thống như viết tay hoặc sử dụng tài liệu số đơn giản thường không tối ưu trong việc hỗ trợ sinh viên hệ thống hóa và tìm kiếm thông tin một cách hiệu quả.

Vì vậy, nhu cầu phát triển một ứng dụng ghi chú chuyên biệt cho học tập là rất cấp thiết. Ứng dụng này không chỉ giúp người dùng ghi chú nhanh chóng mà còn có khả năng phân loại, sắp xếp và phân tích thông tin học tập, giúp sinh viên tối ưu hoá quy trình học và chuẩn bị tốt hơn cho các kỳ thi.

1.2 Mục tiêu và phạm vi đề tài

Với bài toán giới thiệu ở phần 1.1, một số sản phẩm đơn giản như Notepad cũng có thể thực hiện yêu cầu này, nhưng nó không hỗ trợ chia sẻ giữa các người dùng và đồng bộ dữ liệu giữa các thiết bị

Do vậy, sản phẩm của nhóm được tạo ra sẽ nhắm tới mục tiêuxây dựng được một ứng dụng note hỗ trợ học tập có thể đồng bộ giữa các thiết bị. Ứng dụng sẽ có một số chức năng sau: với một số điểm đặc biệt: Đồng bộ dữ liệu giữa các máy với cùng một tài khoản; Xuất, nhập dữ liệu với định dạng PDF; Chia sẻ Note của mình với các tài khoản khác; Giao diện đơn giản, thân thiện, cung cấp trải nghiệm tuyệt vời cho người dùng

1.3 Định hướng giải pháp

Từ việc xác định rõ nhiệm vụ cần giải quyết ở phần 1.2, nhóm quyết định sẽ xây dựng phần mềm theo mô hình thác nước (waterfall) để phát triển phần mềm, hướng tới việc xây dựng ứng dụng chạy trên nền tảng desktop.

Với định hướng như vậy, nhóm quyết định lựa chọn ngôn ngữ Java với framework JavaFX để lập trình và sử dụng GitHub để quản lý phiên bản, hỗ trợ phát triển phần mềm. Kiến trúc phần mềm được lựa chọn là kiến trúc phân tầng Layered Architecture do tính rõ ràng và dễ

triển khai của nó. Đồng thời, cơ sở dữ liệu sẽ được quản lý trên hệ quản trị cơ sở dữ liệu MySQL do tính dễ sử dụng của hệ quản trị này.

Cuối cùng, đóng góp chính của bài tập lớn này là tạo ra một ứng dụng note, qua đó giúp nhóm sinh viên có dịp nắm chắc các kiến thức đã học trong môn Công nghệ phần mềm.

1.4 Bố cục đồ án

Phần còn lại của báo cáo đồ án tốt nghiệp này được tổ chức như sau.

Chương 2 trình bày tổng quan về các chức năng chính của ứng dụng, đưa ra các quy trình nghiệp vụ và luồng hoạt động của các chức năng chính trong ứng dụng. Đồng thời, ở chương này, nhóm cũng sẽ đưa ra các biểu đồ Use case và biểu đồ hoạt động để đặc tả cho một số Use case quan trọng.

Chương 3 trình bày về một số các mẫu thiết kế và công nghệ được sử dụng trong việc phát triển phần mềm. Đó là các mẫu thiết kế như DAO Pattern, Singleton Pattern và framework được sử dụng chính để phát triển ứng dụng này là JavaFX.

Chương 4 sẽ trình bày về các thiết kế hệ thống và thiết kế chi tiết cho ứng dụng. Chương này sẽ lần lượt các biểu đồ UML diễn tả hệ thống từ mức tổng quan tới chi tiết, thông qua các biểu đồ package và class. Đồng thời, chương này cũng sẽ trình bày về giao diện của ứng dụng.

Chương 5 sẽ trình bày về phần kết luận và đánh giá kết quả của bài tập lớn. Đồng thời, trong chương này nhóm cũng sẽ trình bày về một số hướng phát triển cho ứng dụng trong tương lai.

Chương 2 Khảo sát và phân tích yêu cầu

Chương này sẽ trình bày về các chức năng của ứng dụng từ việc phân tích yêu cầu phần mềm. Trong chương này, nhóm sẽ dùng các biểu đồ Use case Diagram và Activity để diễn tả các chức năng của phần mềm. Chương này chủ yếu tham khảo từ tài liệu SRS của nhóm Nhóm 17-SRSv1.2.pdf.

2.1 Khảo sát hiện trạng

Từ việc phân tích một số ứng dụng tương tự như Notation hay Evernote, nhóm đưa ra một vài chức năng chính cần phải phát triển cho ứng dụng: tạo/xóa/sửa các note; cung cấp các thành phần note dạng văn bản và khảo sát; lưu trữ và quản lý note bằng tài khoản; chia sẻ và đồng bộ các note với người dùng khác; quản lý thông tin cá nhân của người dùng; tìm kiếm và sắp xếp ghi chú; quản lý quyền chia sẻ của admin.

Ứng dụng sẽ không được tập trung phát triển các tính năng: đồng bộ hóa đa nền tảng (mobile và web), cung cấp các thành phần đa dạng hơn (video, âm thanh, hình ảnh).

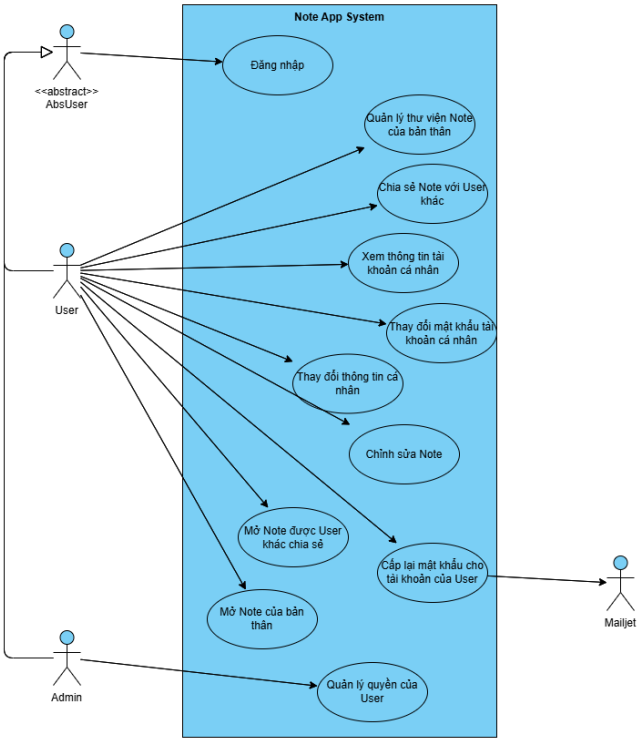
2.2 Tổng quan chức năng

2.2.1 Biểu đồ use case tổng quan

Hệ thống sẽ có 3 tác nhân tham gia, đó là User (những người sử dụng ứng dụng với mục đích là sử dụng note), Admin(những người sử dụng ứng dụng với mục đích là quản lý các User khác), Mailjet (hệ thống bên ngoài cung cấp API để gửi nhận mail).

Như đã trình bày trong phần 2.1, ứng dụng cung cấp cho User một số chức năng chính như: đăng nhập vào hệ thống, chỉnh sửa các Note, quản lý thư viện Note (tạo/xóa Note), quản lý tài khoán cá nhân, cấp lại mật khẩu mới, chia sẻ các Note với User khác.

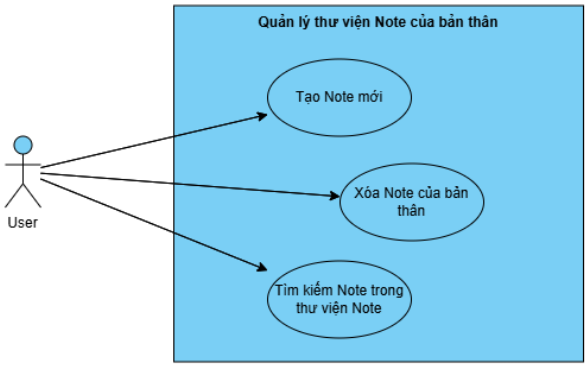
Ngoài ra, ứng dụng cung cấp cho Admin các chức năng khóa/mở khóa quyền chia sẻ của người dùng.



Hình 2-1. Biểu đồ Use case tổng quan

2.2.2 Biểu đồ use case phân rã Use case Quản lý thư viện Note của bản thân

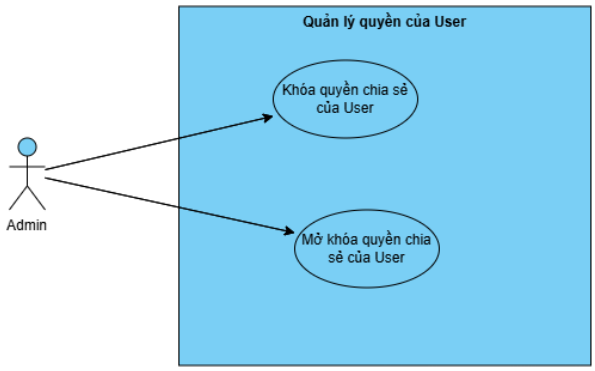
Trong UC này, User có thể thực hiện 3 chức năng cơ bản trong việc quản lý là tạo, xóa và tìm kiếm dựa trên tên và filter.



Hình 2-2. Biểu đồ Use case phân ra cho UC Quản lý thư viện Note của bản thân

2.2.3 Biểu đồ use case phân ra Use case Quản lý quyền của User

UC này phân rã chức năng quản lý quyền các User được thực hiện bởi Admin.

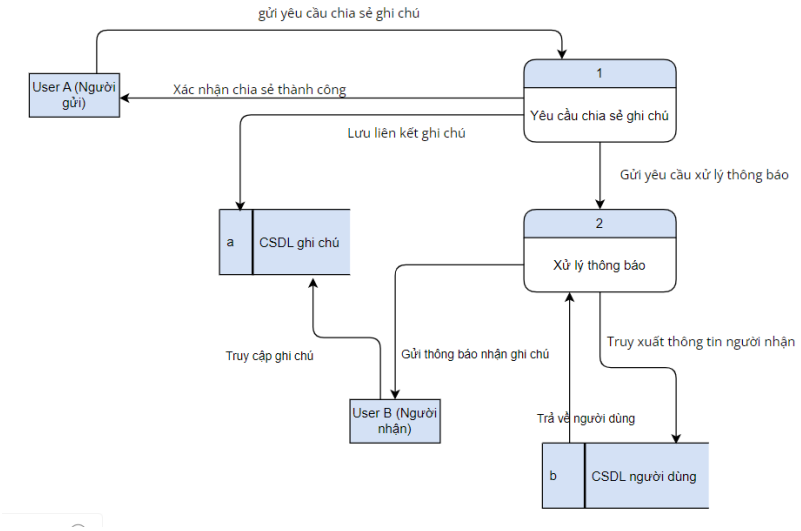


Hình 2-3. Biểu đồ Use case phân ra cho UC Quản lý quyền của User

2.2.4 Quy trình nghiệp vụ

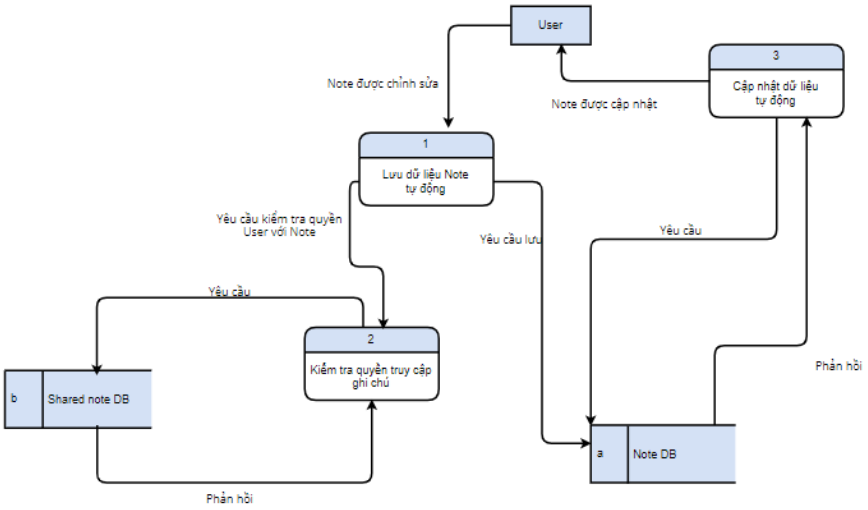
Trong phần này, nhóm sẽ trình bày về 2 quy trình nghiệp vụ quan trọng nhất của hệ thống, đó là quy trình chia sẻ Note và đồng bộ các Note được chia sẻ giữa nhiều người dùng. Các quy trình được diễn tả thông qua biểu đồ luồng dữ liệu (Data Flow Diagram).

Với quy trình chia sẻ Note, User cần đăng nhập, nhập tên người nhận và chọn Note cần chia sẻ. Sau đó phần mềm sẽ thêm Note được chia sẻ vào kho lưu trữ các Note được chia sẻ và gửi thông báo tới người nhận. Người nhận sau đó có thể sử dụng Note được chia sẻ.



Hình 2-4. Biểu đồ Data Flow cho quy trình nghiệp vụ chia sẻ Note

Với quy trình đồng bộ giữa các Note được chia sẻ giữa nhiều người dùng Mỗi khi User truy cập vào một Note được chia sẻ giữa nhiều người dùng thì: Thứ nhất, cứ sau mỗi thao tác chỉnh sửa, phần mềm sẽ tự động lưu thay đổi và cập nhật lên CSDL, đi kèm là việc kiểm tra quyền chỉnh sửa Note; thứ hai, cài đặt bộ thời gian để tự động cập nhật sự thay đổi của các User khác từ CSDL.\



Hình 2-5. Biểu đồ Data Flow cho quy trình nghiệp vụ đồng bộ Note được chia sẻ

2.3 Đặc tả chức năng

2.3.1 Đặc tả use case Tạo một Note mới cho tài khoản của User

Use Case "Tạo một Note mới cho tài khoản của User"

1. Mã use case

UC001

2. Mô tả ngắn gọn

User muốn tạo một ghi chú (Note) mới trong tài khoản của mình. Hệ thống cho phép User nhập nội dung và lưu trữ ghi chú này vào cơ sở dữ liệu của hệ thống.

3. Tác nhân

3.1 User

4. Tiền điều kiện

User đã đăng nhập vào tài khoản

5. Luồng sự kiện cơ sở

1. User nhập tiêu đề (header) của Note mới và ấn "Tạo Note".
2. Phần mềm kiểm tra xem tiêu đề đã tồn tại trong các Note được tạo bởi User chưa.
3. Phần mềm tạo một Note mới với tiêu đề này.
4. Phần mềm thêm Note mới vào danh sách các Note của User và hiển thị trên trang thư viện Note.
5. Phần mềm thông báo rằng Note mới được tạo thành công.
6. Kết thúc Use Case.

6. Luồng sự kiện thay thế

Bảng N-Các luồng sự kiện thay thế cho thứ tự UC Place

No	Vị trí	Điều kiện	Hành động	Vị trí quay lui
1.	Ở bước 2	Nếu tiêu đề đã tồn tại trong các Note của User	▪ Thông báo rằng tiêu đề đã tồn tại	Kết thúc UC.

7. Dữ liệu đầu vào

Bảng A- đặc tả dữ liệu đầu vào

No	Data fields	Description	Mandatory	Valid condition	Example
1.	Header	Tiêu đề của Note mới	Bắt buộc	Một chuỗi ký tự khác rỗng	MyNote

8. Dữ liệu đầu ra

Bảng đặc tả dữ liệu đầu ra

No	Data fields	Description	Display format	Example
1.	StatusMessage	Thông báo kết quả tạo Note	Chuỗi ký tự	Note MyNote đã được tạo thành công
2.	UpdatedNoteList	Danh sách các Note của User đã được cập nhật	Một danh sách các Note hiển thị trên giao diện	
3.	CreatedNote	Note vừa được tạo của User	Một Note	MyNote

9. Hậu điều kiện (nếu có)

Note mới được tạo thành công sẽ xuất hiện trong danh sách các Note của User và Note này được lưu trong CSDL.

Bảng 2-1. Đặc tả Use case Tạo một Note mới cho tài khoản của User

2.3.2 Đặc tả use case Chia sẻ Note

Use Case "Chia sẻ Note"

1. Mã use case

UC002

2. Mô tả ngắn gọn

Khi người dùng muốn chia sẻ Note của mình cho người dùng khác đọc hoặc chỉnh sửa, phần mềm sẽ kiểm tra các điều kiện cho phép người dùng có thể thực hiện thao tác trên.

3. Tác nhân

3.1 User

4. Tiền điều kiện :

User đã đăng nhập tài khoản.

5. Luồng sự kiện cơ sở

1. User chọn tính năng chia sẻ note
2. Phần mềm kiểm tra xem User có bị cấm quyền chia sẻ bởi Admin không?
3. User nhập username của người nhận và chọn Note được chia sẻ.
4. Phần mềm kiểm tra xem username của người nhận có tồn tại hay không.
5. User chọn loại chia sẻ (READ_ONLY hoặc CAN_EDIT) và ấn nút chia sẻ Note.
6. Phần mềm cập nhật danh sách Note được chia sẻ của người nhận trên CSDL.
7. Phần mềm thông báo chia sẻ thành công.
8. Kết thúc UC.

6. Luồng sự kiện thay thế

Bảng N-Các luồng sự kiện thay thế cho thứ tự UC Place

No	Vị trí	Điều kiện	Hành động	Vị trí quay lui
1.	Ở bước 2	Nếu User hiện tại bị cấm quyền chia sẻ bởi Admin của Note muốn chia sẻ	▪ Thông báo cho User Note đã bị cấm chia sẻ	Kết thúc UC.
2.	Ở bước 4	Nếu User nhập sai tên Username muốn chia sẻ	▪ Thông báo "Người nhận không tồn tại"	Quay lại bước 3.

7. Dữ liệu đầu vào

Bảng đặc tả dữ liệu đầu vào

No	Data fields	Description	Mandatory	Valid condition	Example
1.	Receiver	Tên đăng nhập của người nhận	Bắt buộc	Một chuỗi kí tự khác rỗng	Nhom17

8. Dữ liệu đầu ra

Bảng đặc tả dữ liệu đầu ra

No	Data fields	Description	Display format	Example
1.	StatusMessage	Thông báo kết quả chia sẻ Note	Chuỗi ký tự	"Note MyNote được chia sẻ tới User ABC thành công"
2.	ShareNote	Note được chia sẻ thành công	Một Note được chia sẻ	MyNote

9. Hậu điều kiện (nếu có)
Note được chia sẻ được thêm vào CSDL.

Bảng 2-2. Đặc tả Use case Chia sẻ Note

2.3.3 Đặc tả Use case Chỉnh sửa Note

Use Case "Chỉnh sửa Note"

1. Mã use case

UC003

2. Mô tả ngắn gọn

Khi người dùng muốn chỉnh sửa Note, hệ thống sẽ kiểm tra xem người dùng có quyền chỉnh sửa không. Nếu có quyền, người dùng sẽ được phép thực hiện thao tác chỉnh sửa Note.

3. Tác nhân

3.1 User

4. Tiền điều kiện :

User đã đăng nhập vào tài khoản

5. Luồng sự kiện cơ sở

1. User chuyển sang giao diện chỉnh sửa Note
2. Phần mềm kiểm tra quyền của User đối với Note này. Nếu User được cấp quyền chỉnh sửa thành phần này, tiếp tục.
3. User chỉnh sửa thành phần Note.
4. User hoàn thành chỉnh sửa thành phần Note.
5. Phần mềm tự động lưu lại chỉnh sửa của User.
6. Nếu thành công, phần mềm thông báo "Chỉnh sửa đã được lưu" cho User.
7. Kết thúc UC

6. Luồng sự kiện thay thế

Bảng N-Các luồng sự kiện thay thế cho thứ tự UC Place

No	Vị trí	Điều kiện	Hành động	Vị trí quay lui
1.	Ở bước 2	Nếu User không có quyền chỉnh sửa Note	▪ Thông báo cho User "Bạn chỉ có quyền xem Note này, không thể chỉnh sửa"	Quay trở lại bước 1
2.	Ở bước 6	Nếu hệ thống không thể lưu lại chỉnh sửa do lỗi hoặc sự cố	• Thông báo cho User "Lưu không thành công" và quay lại giao diện chỉnh sửa để thử lại	Quay trở lại bước 4

7. Dữ liệu đầu vào

Bảng đặc tả dữ liệu đầu vào

No	Data fields	Description	Mandatory	Valid condition	Example
1.					

8. Dữ liệu đầu ra

Bảng đặc tả dữ liệu đầu ra

No	Data fields	Description	Display format	Example
1.	StatusMessage	Thông báo kết quả lưu Note	Chuỗi ký tự	" Chỉnh sửa đã được lưu "
2.	Note	Note đã được lưu	Một Note	" Lưu không thành công "

9. Hậu điều kiện (nếu có)

- Hệ thống lưu lại lịch sử chỉnh sửa cho Note vào CSDL để có thể truy xuất nếu cần thiết.

Bảng 2-3. Đặc tả Use case chỉnh sửa Note

2.3.4 Đặc tả Use case Mở Note được người khác chia sẻ

Use Case "Mở Note được người khác chia sẻ"

1. Mã use case

UC004

2. Mô tả ngắn gọn

User muốn xem hoặc chỉnh sửa note. Tùy theo quyền được cấp cho user thì user có thể chỉ xem hoặc có thể chỉnh sửa note. Phần mềm tự động cập nhật nội dung note khi user chỉnh sửa.

3. Tác nhân

3.1 User

4. Tiền điều kiện

User đã đăng nhập tài khoản.

5. Luồng sự kiện cơ sở

1. User chọn Note cần mở.
2. Phần mềm lấy thông tin của Note tương ứng
3. Phần mềm hiển thị dữ liệu trên trang Edit
4. Phần mềm kiểm tra loại chia sẻ là chỉ đọc hay có thể Edit.
5. Phần mềm đặt Note vào trạng thái tự động cập nhật nội dung từ các User khác.
6. Kết thúc UC.

6. Luồng sự kiện thay thế

Bảng N-Các luồng sự kiện thay thế cho thứ tự UC Place

No	Vị trí	Điều kiện	Hành động	Vị trí quay lui
1.	Ở bước 3	Nếu thất bại	▪ Thông báo lỗi không mở được Note	Đến bước 6.
2.	Ở bước 4	Nếu là loại chỉ đọc	▪ Khóa tất cả các quyền chỉnh sửa thành phần Note	Đến bước 5.

7. Dữ liệu đầu vào

Bảng A- đặc tả dữ liệu đầu vào

No	Data fields	Description	Mandatory	Valid condition	Example

8. Dữ liệu đầu ra

Bảng đặc tả dữ liệu đầu ra

No	Data fields	Description	Display format	Example
1.	Note	Note được mở	Một Note	MyNote

9. Hậu điều kiện (nếu có)

- Các thông tin của Note được hiển thị trên trang Edit Note.

Bảng 2-4. Đặc tả Use case Mở Note được người khác chia sẻ

2.3.5 Đặc tả Use case Cấp lại mật khẩu mới cho tài khoản User

Use Case "Cấp lại mật khẩu mới cho tài khoản của User"

1. Mã use case

UC005

2. Mô tả ngắn gọn

Khi quên mật khẩu, User muốn phần mềm cấp lại một mật khẩu mới cho mình. Phần mềm sẽ sử dụng xác thực qua SMS điện thoại để cấp lại mật khẩu cho User.

3. Tác nhân

3.1 User

3.2 Mailjet

4. Tiền điều kiện

Không có

5. Luồng sự kiện cơ sở

1. User ấn quên mật khẩu

2. Hệ thống hiển thị trang cấp lại mật khẩu
3. User nhập username (xem bảng A)
4. Hệ thống kiểm tra username và địa chỉ email xác thực của username
5. Hệ thống tạo mã xác thực ngẫu nhiên (tồn tại 3 phút).
6. Hệ thống yêu cầu Atomic SMS Sender API gửi mã xác nhận tới địa chỉ email nhận mã.
7. User nhập mã xác thực.
8. Hệ thống kiểm tra mã xác thực và chuyển tới trang đặt lại mật khẩu.
9. User nhập mật khẩu mới.
10. Hệ thống lưu mật khẩu mới vào CSDL và chuyển tới trang đăng nhập.
11. Kết thúc UC.

6. Luồng sự kiện thay thế

Bảng N-Các luồng sự kiện thay thế cho thứ tự UC Place

No	Vị trí	Điều kiện	Hành động	Vị trí quay lui
1.	Ở bước 4	Nếu không tồn tại tài khoản có username được nhập hoặc username không có email xác thực	▪ Thông báo User nhập sai thông tin.	Quay lại bước 3.
2.	Ở bước 8	Nếu mã xác thực bị sai	▪ Thông báo nhập sai mã xác thực	Quay lại bước 7.
3.	Ở bước 8	Nếu mã xác thực bị hết hạn	▪ Thông báo mã xác thực đã hết hạn	Quay lại bước 5.

7. Dữ liệu đầu vào

Bảng A- đặc tả dữ liệu đầu vào

No	Data fields	Description	Mandatory	Valid condition	Example
1.	Username	Tên đăng nhập của User	Bắt buộc	Một chuỗi ký tự khác rỗng	Nhom17
2.	VerifyCode	Mã xác thực	Bắt buộc	Chuỗi gồm 6 số	123456

8. Dữ liệu đầu ra

Bảng đặc tả dữ liệu đầu ra

No	Data fields	Description	Display format	Example
1.	StatusMessage	Thông báo cấp lại mật khẩu thành công	Chuỗi ký tự	"Mật khẩu của bạn dã được cập nhật thành công"
2.	NewPassword	Mật khẩu mới của người dùng	Chuỗi ký tự được mã hóa	Acd1@2ef

9. Hậu điều kiện (nếu có) Mật khẩu mới được cập nhật trên CSDL và User có thể dùng mật khẩu này để đăng nhập

Bảng 2-5. Đặc tả Use case Cấp lại mật khẩu mới cho tài khoản của User

2.4 Yêu cầu phi chức năng

Về việc thông báo lỗi, trong các chuỗi sự kiện của các use case, tất cả các bước có thao tác với CSDL, nếu có lỗi trong quá trình kết nối hoặc thao tác, cần có thông báo lỗi tương ứng để tác nhân biết là lỗi liên quan đến CSDL chứ không liên quan tới lỗi của người dùng.

Các use case do Admin và Usser thì Khách cần đăng nhập với vai trò tương ứng.

Về định dạng hiển thị, các định dạng chung là: số và chữ căn trái, sử dụng font là System màu đen, nền trắng.

Về thời gian phản hồi, hệ thống phản hồi nhiều nhất trong vòng 3 giây cho các yêu cầu của người dùng.

Về khả năng chịu tải, trước hết hệ thống hỗ trợ tối thiểu 300-400 người dùng mà không làm giảm hiệu năng.

Về khả năng khôi phục, trong trường hợp bị lỗi, đảm bảo khả năng khôi phục hoạt động bình thường trong vòng 2 giờ.

Về mã hóa dữ liệu, mã hóa tất cả dữ liệu cá nhân nhạy cảm (mật khẩu) của người dùng.

Về việc coding và bảo trì, cần tuân thủ 2 chỉ số: Cyclomatic Complexity nhiều nhất là 12, Depth cho mỗi hàm nhiều nhất là 6.

Vậy với các yêu cầu chức năng như vậy, việc sử dụng các công nghệ hợp lý là quan trọng. Các công nghệ mà nhóm sử dụng sẽ được trình bày trong chương sau.

Chương 3 Công nghệ sử dụng

Trong chương trước, nhóm đã trình bày về các chức năng tổng quan cũng như đặc tả một số chức năng quan trọng của ứng dụng. Ở chương này, nhóm sẽ trình bày về một số công nghệ được sử dụng, trong đó có các thư viện quan trọng và một số lý thuyết về các mẫu thiết kế (Design Pattern) được sử dụng

3.1 Thư viện JavaFX

Với yêu cầu phát triển phần mềm chạy trên nền tảng desktop, cộng thêm nhóm quyết định lựa chọn ngôn ngữ lập trình chính là Java, thư viện JavaFX là lựa chọn hàng đầu của nhóm trong việc hỗ trợ thiết kế giao diện người dùng (GUI) và xử lý sự kiện.

JavaFX là cơ sở phát triển ứng dụng đa nền tảng và giao diện người dùng đồ hoạ (GUI) dành cho ngôn ngữ lập trình Java. Được giới thiệu vào năm 2007 bởi Sun Microsystems (sau này là Oracle), JavaFX đã phát triển thành bộ công cụ mạnh mẽ cho việc xây dựng các ứng dụng có giao diện người dùng đẹp mắt và khả năng tương tác cao trong ngôn ngữ Java.

Từ JDK 11 trở lên, Oracle quyết định loại bỏ JavaFX khỏi bản phân phối của JDK theo nguyện vọng mở cửa cho sự phát triển cộng đồng mở rộng. Tính đến thời điểm hiện tại, JavaFX tiếp tục được phát triển và hỗ trợ bởi cộng đồng Java và các tổ chức khác nhau.

Một số ưu điểm của JavaFX có thể kể đến như: hỗ trợ FXML, một ngôn ngữ đánh dấu dựa trên XML cung cấp cơ sở tiếp cận dễ dàng hơn cho việc xây dựng, quản lý giao diện người dùng; hỗ trợ giao diện kéo thả bằng Scene Builder; tích hợp điều khiển GUI đa dạng thông qua API và FXML; mang đến giải pháp tiếp cận CSS để hỗ trợ thiết kế giao diện người dùng; etc.

Mặc dù có một vài nhược điểm về hiệu suất hàng đầu, JavaFX vẫn là lựa chọn hàng đầu của nhóm trong khi thiết kế giao diện người dùng.

3.2 Một vài mẫu thiết kế (Design Pattern) được sử dụng

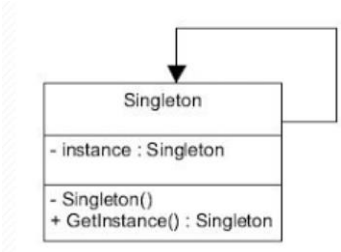
Mẫu thiết kế (Design Patterns) là một kỹ thuật trong lập trình hướng đối tượng, cung cấp các "mẫu thiết kế", giải pháp để giải quyết các vấn đề chung, thường gặp trong lập trình. Các vấn đề mà bạn gặp phải có thể bạn sẽ tự nghĩ ra cách giải quyết nhưng có thể nó chưa phải

là tối ưu. Design Pattern giúp bạn giải quyết vấn đề một cách tối ưu nhất, cung cấp cho bạn các giải pháp trong lập trình OOP.

Có 3 nhóm Design Pattern chính là khởi tạo (Creational Pattern), cấu trúc (Structural Pattern) và hành vi (Behavioral Pattern). Phần dưới đây sẽ chỉ trình bày về 2 Design Pattern chính sẽ được sử dụng để triển khai một cách khoa học các chức năng của ứng dụng.

3.2.1 Singleton Pattern

Singleton đảm bảo chỉ duy nhất một thể hiện (instance) được tạo ra và nó sẽ cung cấp cho bạn một method để có thể truy xuất được thể hiện duy nhất đó mọi lúc mọi nơi trong chương trình.



Hình 3-1. Minh họa Singleton Pattern

Có nhiều cách để thực thi Singleton. Trong việc triển khai thiết kế ứng dụng này, do việc thiết kế Server là đa luồng, nên khi thiết kế Singleton cũng phải đảm bảo thể hiện sẽ được truy nhập hợp lý giữa các luồng. Vì vậy, cách thực thi được chọn là Bill Pugh Singleton Implementation.

Với cách làm này bạn sẽ tạo ra static nested class với vai trò 1 Helper. Khi Singleton được tải vào bộ nhớ thì SingletonHelper chưa được tải vào. Nó chỉ được tải khi và chỉ khi phương thức gọi thể hiện được gọi tới. Với cách này tránh được lỗi cơ chế khởi tạo instance của Singleton trong Multi-Thread. Chi tiết sẽ được trình bày trong chương 4.

Ngoài ra, có thể sử dụng cách thực thi Thread Safe Singleton để thực thi Singleton.

3.2.2 DAO Pattern

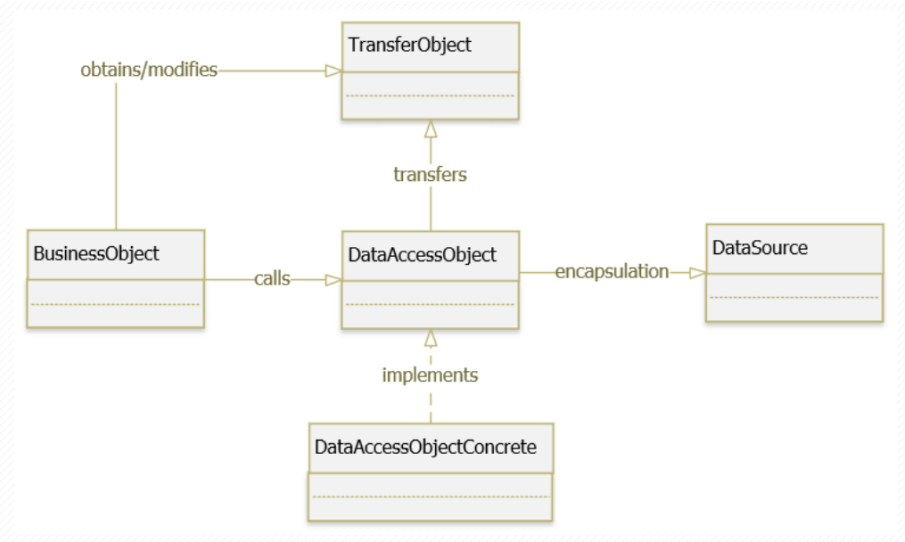
Data Access Object (DAO) Pattern là một trong những Pattern thuộc nhóm cấu trúc. Mẫu thiết kế DAO được sử dụng để phân tách logic lưu trữ dữ liệu trong một lớp riêng biệt. Theo cách này, các service được che dấu về cách các hoạt động cấp thấp để truy cập cơ sở dữ liệu được thực hiện.

Ý tưởng là thay vì có logic giao tiếp trực tiếp với cơ sở dữ liệu, hệ thống file, dịch vụ web hoặc bất kỳ cơ chế lưu trữ nào mà ứng dụng cần sử dụng, chúng ta sẽ để logic này sẽ giao tiếp với lớp trung gian DAO. Lớp DAO này sau đó giao tiếp với hệ thống lưu trữ, hệ quản

trị CSDL như thực hiện các công việc liên quan đến lưu trữ và truy vấn dữ liệu (tìm kiếm, thêm, xóa, sửa,…).

DAO Pattern dựa trên các nguyên tắc thiết kế abstraction và encapsulation. Nó bảo vệ phần còn lại của ứng dụng khỏi mọi thay đổi trong lớp lưu trữ, ví dụ: thay đổi database từ Oracle sang MySQL, thay đổi công nghệ lưu trữ từ file sang database.

Mô hình của DAO Pattern được triển khai như sau (chi tiết hơn có thể xem thiết kế các package dao ở chương 4).



Hình 3-2. Minh họa DAO Pattern

Chương này đã trình bày cơ bản các lý thuyết quan trọng cần thiết cho việc thiết kế ứng dụng. Chi tiết việc thiết kế này sẽ được trình bày trong chương sau - Chương 4.

Chương 4 Phát triển và triển khai ứng dụng

Trong các chương trước, ta đã trình bày về các chức năng chính của hệ thống và các lý thuyết cơ sở để triển khai các chức năng đó. Chương này sẽ trình bày việc triển khai và thiết kế chi tiết của ứng dụng.

4.1 Thiết kế kiến trúc

4.1.1 Lựa chọn kiến trúc phần mềm

Với các yêu cầu về chức năng như đã trình bày trong Chương 2, cùng với việc phân tích các khía cạnh yêu cầu, nhóm quyết định lựa chọn kiến trúc phần mềm là Layered Architecture (kiến trúc phân tầng) cùng với một số cải tiến.

Layered architecture là kiểu kiến trúc được tổ chức theo chiều dọc bao gồm nhiều tầng xếp chồng lên nhau. Tùy vào độ phức tạp của dự án mà kiến trúc có số lượng tầng và chức năng của mỗi tầng khác nhau.

Thường thì Layered Architecture gồm 4 tầng chính theo thứ tự từ trên xuống dưới như sau: Presentation – chứa các xử lý tương tác với người dùng, gồm các GUI và cách thức xử lý dữ liệu và trung chuyển dữ liệu, Application (Business Layer) – tầng nghiệp vụ, cung cấp các nghiệp vụ và xử lý logic của hệ thống, Presistence – tầng liên kết dữ liệu, chịu trách nhiệm tương tác với tầng database, có thể kể đến như quản lý connection pool và thực hiện các cầu truy vấn được yêu cầu bởi tầng trên, Database - là các dịch vụ lưu trữ như Mysql, MongoDb cung cấp các phương thức để thêm, xóa, sửa, truy vấn dữ liệu.

Thông thường Layerd Architecture triển khai theo hướng Closed Layer, bắt buộc một request từ tầng trên phải gọi trực tiếp xuống tầng kế tiếp nó (tức là phải đi theo đúng thứ tự Presentation, Application, Presistence, Database). Một số trường hợp có thể triển khai theo Opene Layer, nghĩa là request có thể bỏ qua tầng (ví dụ từ tầng Presentation có thể đi thẳng tới Presistence).

Layered Architecture có ưu điểm rõ nhất là tính đơn giản và dễ triển khai của nó, đồng thời cũng làm giảm sử phụ thuộc giữa các tầng, khi hướng tới việc sự thay đổi ở một tầng sẽ không làm ảnh hưởng nhiều tới sự thay đổi của các tầng khác.

Tuy vậy, kiến trúc này tồn tại một vài hạn chế về khả năng mở rộng hoặc chịu lỗi. Điều này xảy ra khi tất cả xử lý của các tầng nằm trong một khối lớn, lợi ích của kiến trúc này có thể kể đến như giảm thiểu overhead giao tiếp giữa các component, đơn giản hoá việc xử lý transaction, etc.. nhưng ngược lại, nếu muốn scale một layer nào đó, ta phải scale toàn bộ khối monolithic này, mặc dù các layer khác hoàn toàn đáp ứng được capacity của hệ thống. Do đó scaling trở nên khó khăn, thường không hiệu quả và không tận dụng tối đa tài nguyên. Thêm nữa, vì đây là một kiến trúc hướng về kỹ thuật, nên việc phân chia nghiệp vụ là không rõ ràng.

Do vậy, nhóm có một vài chỉnh sửa trong kiến trúc này, đó là từ tầng nghiệp vụ trở xuống, các tầng Application, Presistence và Database sẽ được chia vào các domain (nghiệp vụ) tương ứng, trong bối cảnh của ứng dụng này là 2 domain chính (user và note). Điều này giúp nhóm thu được một số lợi ích sau:

Thứ nhất, việc tách biệt các tầng kỹ thuật (Layered Architecture) và các lĩnh vực nghiệp vụ (Domain) giúp hệ thống trở nên mô-đun hơn, cho phép các phần của hệ thống được phát triển và bảo trì độc lập.

Thứ hai, dễ bảo trì và mở rộng hơn. Technical Partition (Phân chia kỹ thuật) của kiến trúc Layered Architecture giúp tách biệt các tầng kỹ thuật, làm cho việc thay thế hoặc nâng cấp các phần của hệ thống trở nên dễ dàng. Còn Domain Partition (Phân chia lĩnh vực) giúp phản ánh chính xác các quy trình nghiệp vụ, dễ dàng mở rộng hoặc thay đổi để đáp ứng các yêu cầu kinh doanh mới.

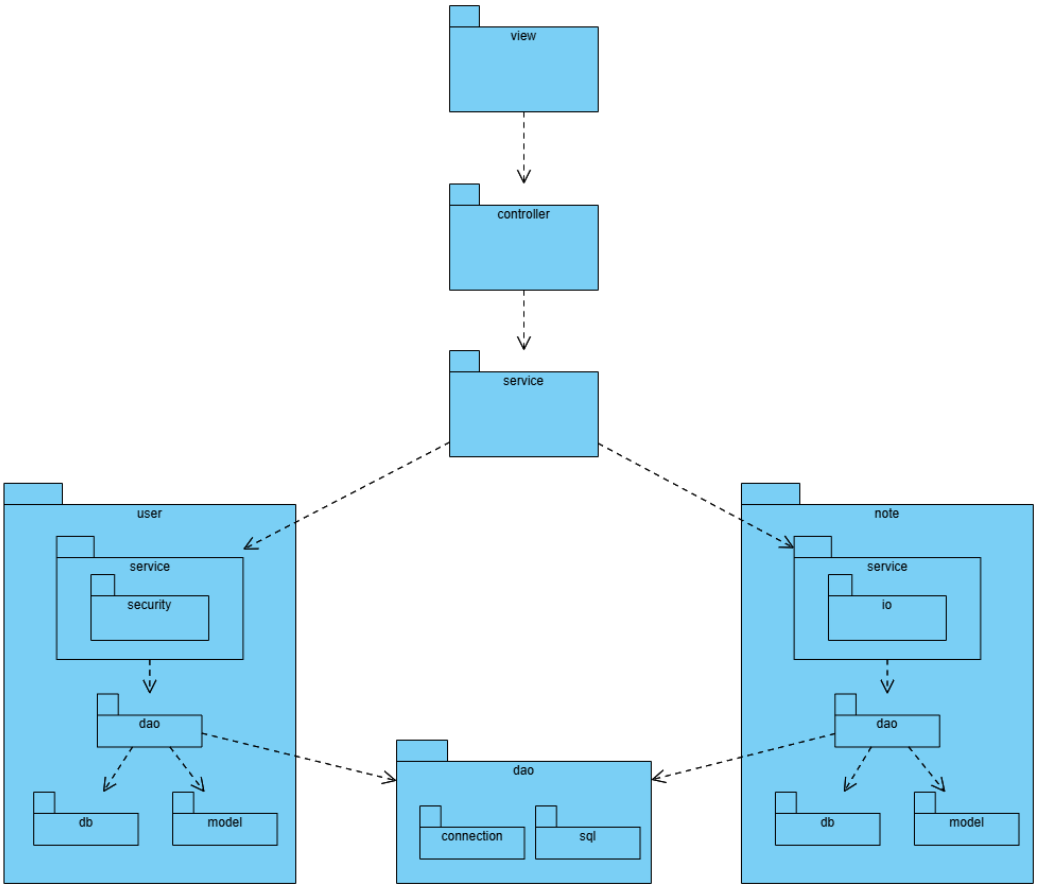
Thứ ba, kết hợp cả hai phương pháp theo kỹ thuật và nghiệp vụ giúp nhóm dễ dàng phối hợp với nhau hơn, vì nhóm có thể làm việc độc lập trên các tầng kỹ thuật hoặc các lĩnh vực cụ thể mà không gây xung đột.

Tuy vậy, cách thiết kế này cũng tiềm ẩn các rủi ro liên quan tới việc quản lý sự phụ thuộc và sự phức tạp của mã nguồn, đồng thời các thiết kế phân tầng cho mỗi nghiệp vụ phải nhất quán.

Cuối cùng, sau khi đánh giá và phân tích, nhóm vẫn quyết định lựa chọn Layered Architecture với chỉnh sửa phân rã theo domain ở tầng Application trở xuống.

4.1.2 Thiết kế tổng quan

Thiết kế tổng quan của ứng dụng được thể hiện trong biểu đồ package bên dưới



Hình 4-1. Biểu đồ package cho kiến trúc tổng quan của ứng dụng

Mỗi package đảm nhận vai trò riêng biệt về kỹ thuật, theo đúng kiến trúc phân tầng, đồng thời cũng thể hiện sự cải tiến của nhóm ở các tầng dưới.

Đầu tiên là package view, nó chứa các giao diện người dùng viết bằng FXML. Tiếp theo là package controller chứa các đối tượng Controller để cung cấp điều khiển và xử lý sự kiện trên giao diện.

Package service được sử dụng để tích hợp các service từ 2 package user và note, giúp giảm sự phụ thuộc giữa controller và các dịch vụ này.

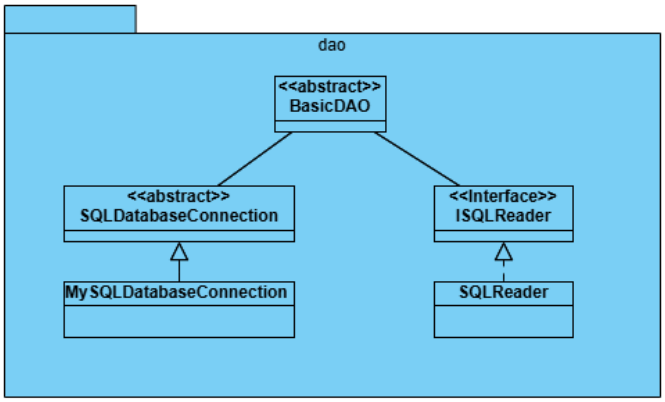
Các tầng dưới trong kiến trúc phân tầng được chia vào 2 package là user (xử lý nghiệp vụ liên quan tới tài khoản và thông tin cá nhân người dùng) và note (xử lý nghiệp vụ liên quan tới note). Ở mỗi package trong 2 package này, package service chứa các class cung cấp các nghiệp vụ liên quan, package dao chứa các class để truy xuất vào thao tác với cơ sở dữ liệu tương ứng của mỗi nghiệp vụ, package model chứa các class đại diện cho các dữ liệu được chuyển đổi từ CSDL, và package db chứa các câu lệnh, truy vấn được phép sử dụng với cơ sở dữ liệu.

Ngoài ra, package dao cung cấp một vài class với mục đích kết nối tới hệ quản trị cơ sở dữ liệu và cung cấp các phương thức để đọc ghi dữ liệu từ file SQL. Hai package dao của user và note sẽ sử dụng các dịch vụ được cung cấp từ package này để lấy connection và đọc câu lệnh sql.

4.1.3 Thiết kế chi tiết các chức năng

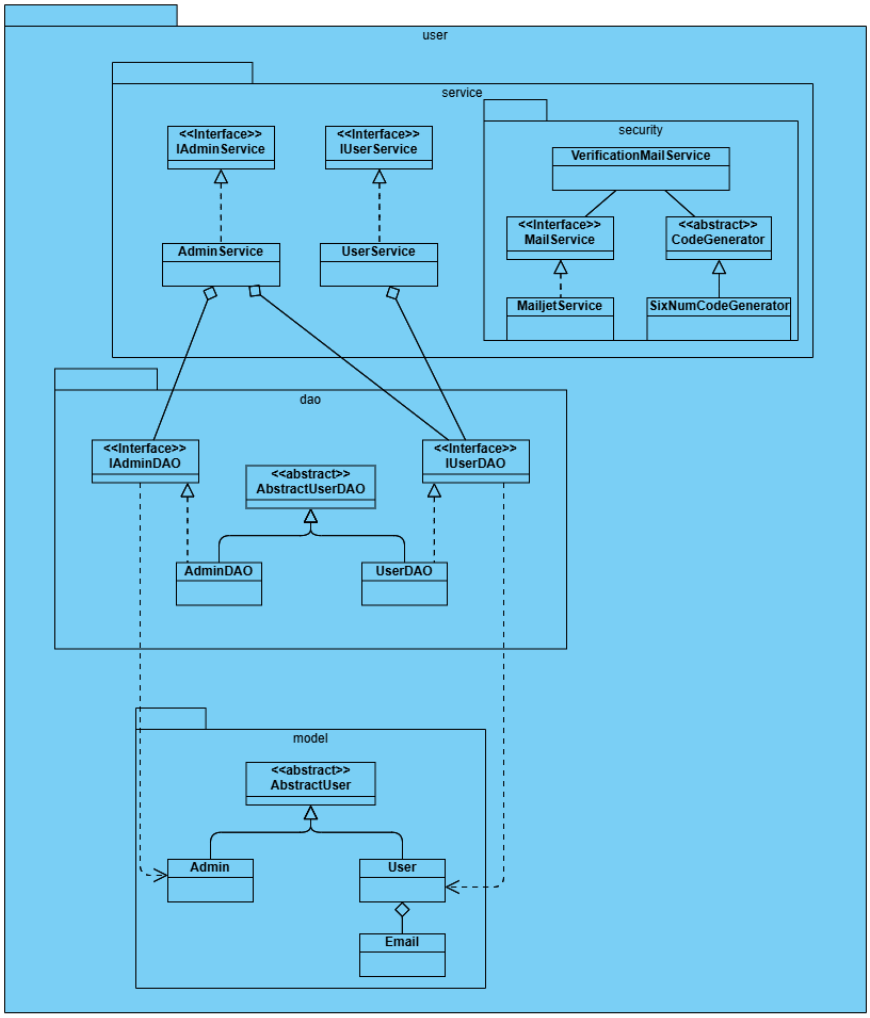
Trong phần này, nhóm sẽ trình bày thiết kế chi tiết cho từng package.

Đầu tiên là package dao. Package này gồm 1 abstract class chính là BasicDAO, nó cung cấp các phương thức mà một DAO bắt buộc cần phải có, đó là lấy connection tới CSDL và có thể đọc file SQL. Hai phương thức tương ứng này sẽ lần lượt sử dụng các abstract class và interface là SQLDatabaseConnection và ISQLReader.



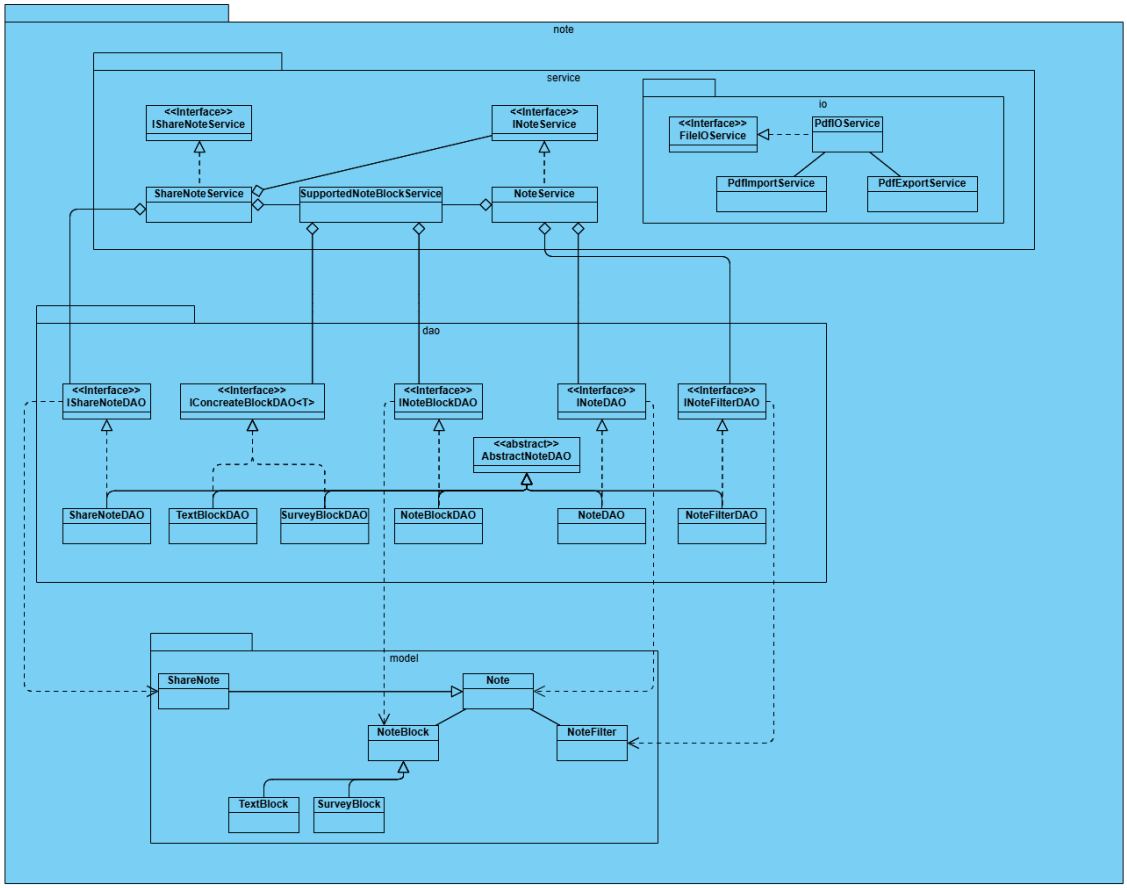
Hình 4-2. Biểu đồ class cho package dao

Thứ hai là package user. Như đã trình bày, package này chứa tất cả những gì liên quan tới nghiệp vụ của User. Bắt đầu từ User là các model thể hiện các đối tượng có thể tham gia vào nghiệp vụ của User, đó là User và Admin. Kế đến là các DAO tương ứng với từng model một. Cuối cùng là các service ứng với từng đối tượng User hay Admin. Chú ý rằng liên kết và sự phụ thuộc giữa các tầng đều qua interface tương ứng. Xem biểu đồ class ở hình sau



Hình 4-3. Biểu đồ class cho package user

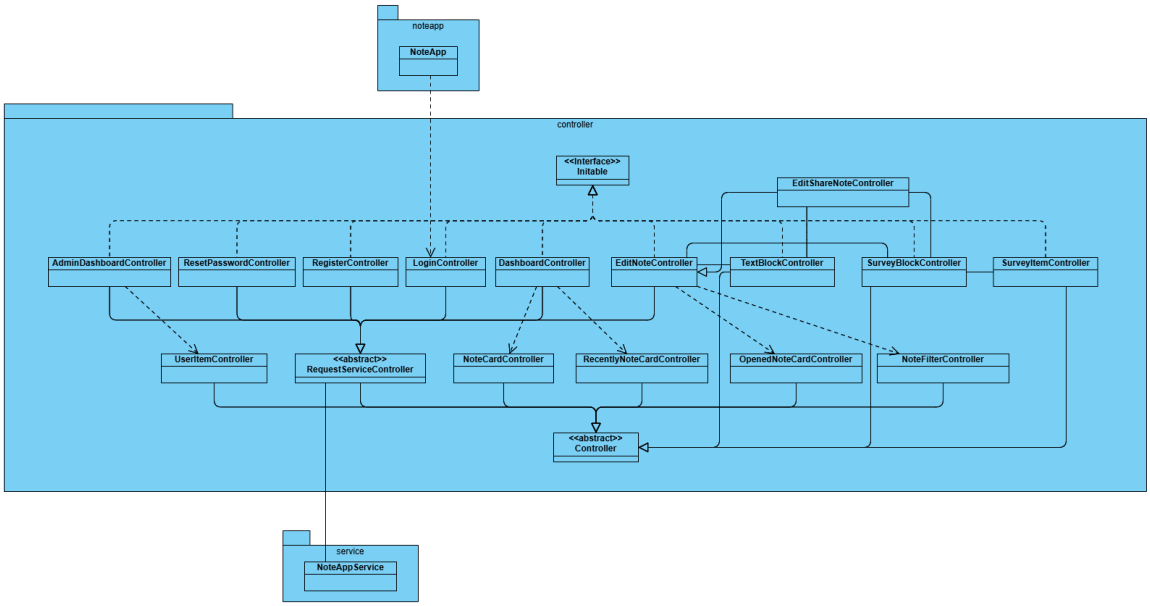
Tương tự như package user, package note cũng gồm các package con với vai trò tương tự. Ở tầng service, do các nghiệp vụ liên quan tới blocks (các thành phần của note) khá phức tạp nên được nhóm tách ra thành một class riêng xử lý các nghiệp vụ này. Các nghiệp vụ chính liên quan tới note và sharenote sẽ sử dụng các nghiệp vụ mà class này cung cấp.



Hình 4-4. Biểu đồ class cho package note

Cuối cùng là 2 tầng logic cao nhất của ứng dụng, đó là tầng controller và service. Về package service, nó sẽ chứa class NoteAppService kết tập các service của User và Note thông qua các interface (hiện có 6 là IUserService, IAdminService, INoteService, IShareNoteService, FileIOService, VerificationMailService).

Package controller sẽ bao gồm một abstract class Controller cung cấp những gì cơ bản nhất một controller phải có, đó là tọa độ và di chuyển, load file giao diện FXML. Ngoài ra, còn có abstract class RequestServiceController đại diện cho các controller phải sử dụng service từ tầng service, cùng với interface Initable định nghĩa phương thức init() để khởi tạo giao diện trước khi hiển thị. Cuối cùng là Main class của ứng dụng, nó sẽ gọi tới GUI Login đầu tiên. Chi tiết có thể xem trong biểu đồ class sau



Hình 4-5. Biểu đồ class cho package service, controller

4.2 Thiết kế chi tiết

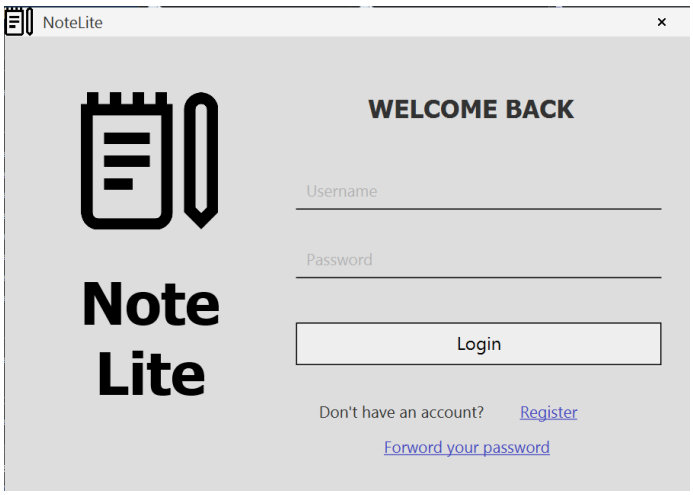
4.2.1 Thiết kế giao diện

Về giao diện của ứng dúng, thứ nhất về kích thước màn hình, màn hình trang login có kích thước 600 x 425, màn hình trang register có kích thước 400 x 600, màn hình trang reset password có kích thước 400 x 550, màn hình trang dashboard (cho cả user và admin) có kích thước 1020 x 630, màn hình cho trang edit note có kích thước 810 x 640.

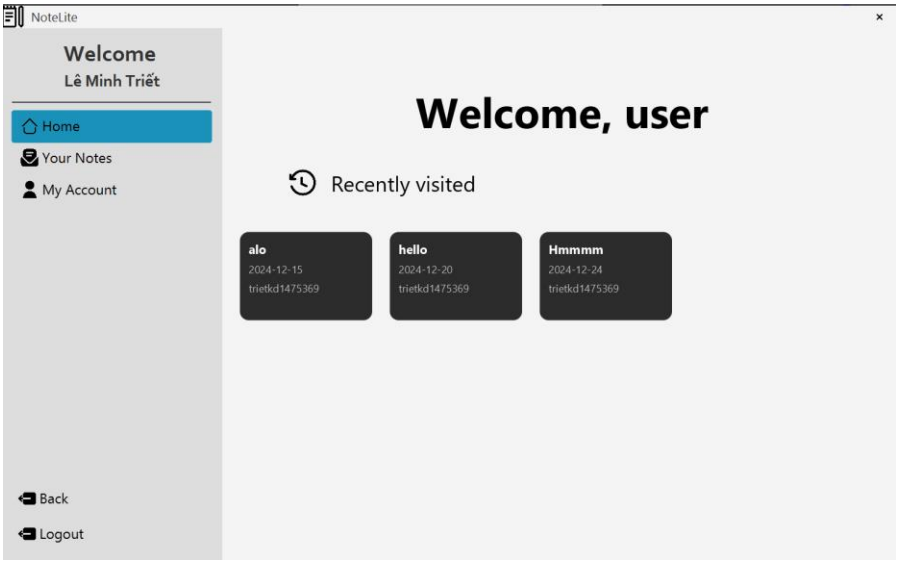
Về số lượng màu sắc hỗ trợ và phối màu, ứng dụng sử dụng các màu sắc chính sau cho các thành phần và các note (#ccc, #ddd, #eee, #000, #fff). Các nút chức năng trong ứng dụng đều có hiệu ứng đổi sang màu tương phản khi ấn (trừ trường hợp nút close đổi sang màu đỏ), đồng thời đều có boundary màu đen bao quanh mỗi nút. Các field điền dữ liệu cũng được thống nhất màu là trong suốt với container chứa nó, đồng thời có boundary ở bottom cho mỗi field.

Các thông điệp phản hồi sẽ được hiện ra với người dùng thông qua các dialog ở chính giữa màn hình người dùng, đè lên giao diện hiện tại của ứng dụng.

Sau đây là một số hình ảnh về màn hình giao diện được thiết kế.



Hình 4-6. Thiết kế giao diện trang login



Hình 4-7. Thiết kế giao diện trang dashboard cho user

4.2.2 Thiết kế lớp

Trong phần này, nhóm sẽ thiết kế luồng truyền thông điệp giữa các đối tượng tham gia vào 2 Use case Chia sẻ Note và Mở Note.

Trước hết, ta có một vài model quan trọng được sử dụng, đó là User, Note, ShareNote. Trước hết là User, một class kế thừa từ Abstract User

```
public abstract class AbstractUser {
	protected String username;
	protected String password;
	//getter and setter
}
public class User extends AbstractUser {
	private String name;
```

```
    private Date birthday;
    private String school;
    private Gender gender;
    private Email email;
    private boolean locked;

    /**
     * Các giới tính có thể có của User
     */
    public static enum Gender {
        MALE, FEMALE, OTHER;
    }
    //Constructor, getter, setter
}
```

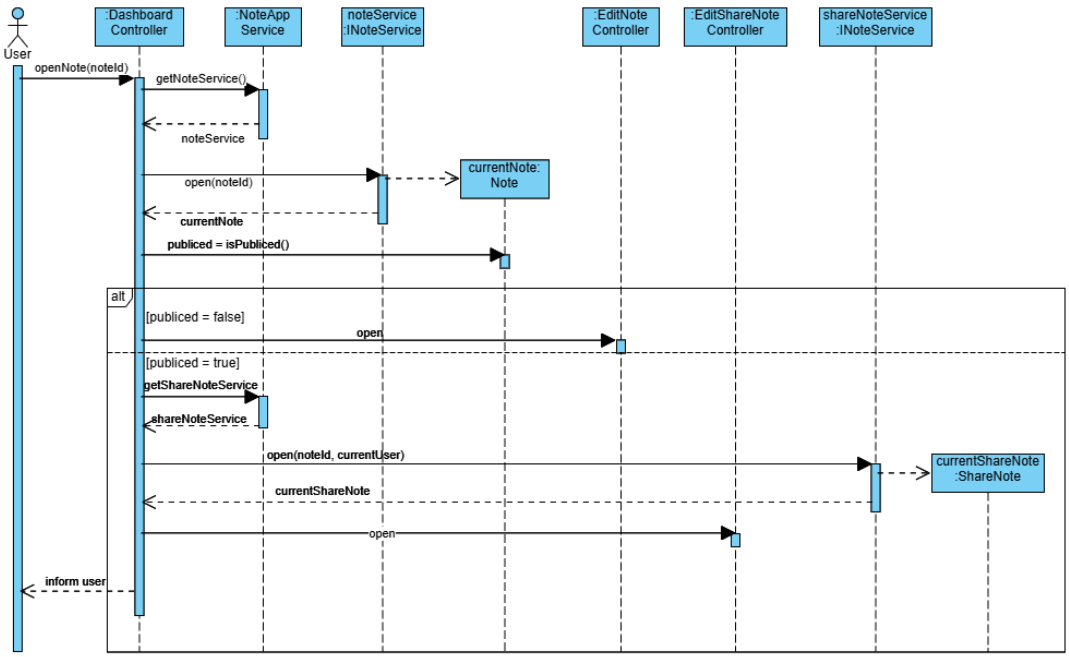
Sau đó là Note và ShareNote

```
public class Note {
    private int id;
    private String author;
    private String header;
    private List<NoteBlock> blocks;
    private Date lastModifiedDate;
    private List<NoteFilter> filters;
    private boolean publiced;
    //Constructor, getter, setter
}
public class ShareNote extends Note {
    private String editor;
    private ShareType shareType;
    private Map<Integer, List<NoteBlock>> otherEditorBlocks;

    /**
     * Định nghĩa các kiểu Share
     */
    public static enum ShareType {
        READ_ONLY, CAN_EDIT;
    }
    //Constructor, getter, setter
}
```

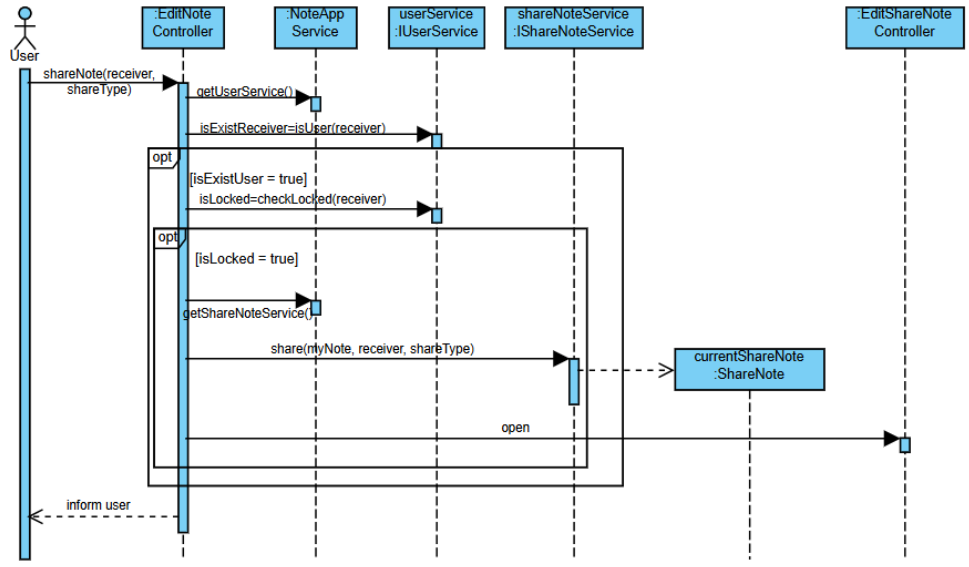
Tiếp theo, nhóm sẽ trình bày về luồng truyền thông điệp của UC Mở một Note. Chú ý biểu đồ Sequence Diagram dưới đây chỉ thể hiện luồng truyền thông điệp ở mức tầng nghiệp vụ. Trong thực tế, khi gọi service open(noteId) của NoteService, vẫn sẽ có các object của các DAO tham gia vào (được gọi từ NoteService). Tuy nhiên, do mục đích của biểu đồ này chỉ

là trình bày về luồng thông điệp chính nên nhóm sẽ không thêm vào biểu đồ này các object đó để biểu đồ rõ ràng hơn.



Hình 4-8. Biểu đồ Sequence biểu diễn luồng truyền thông điệp của UC Mở Note

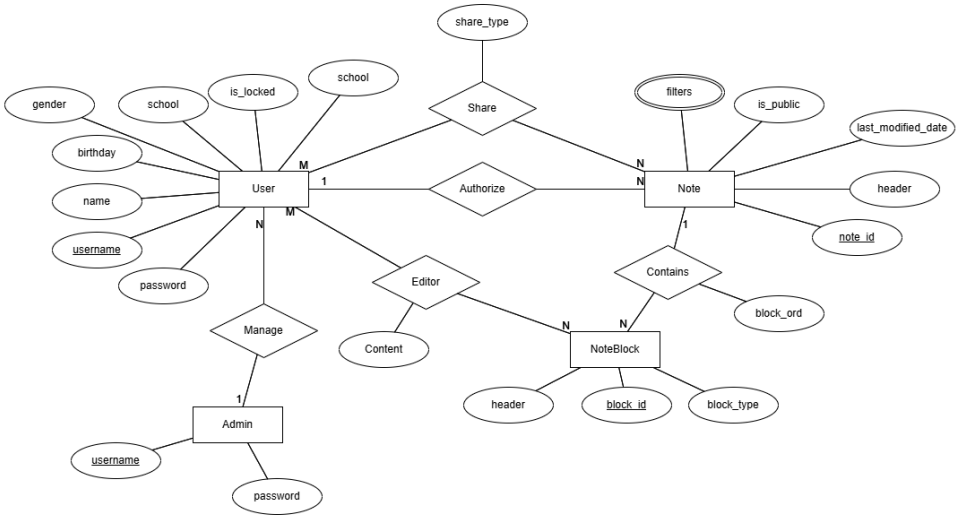
Tương tự với UC Chia sẻ Note, nhóm cũng sẽ chỉ trình bày ở mức service.



Hình 4-9. Biểu đồ Sequence cho luồng truyền thông điệp của UC Chia sẻ Note

4.2.3 Thiết kế cơ sở dữ liệu

Trước hết, nhóm sử dụng cách tiếp cận theo thực thể - liên kết, từ đó vẽ được sơ đồ thực thể liên kết sau



Hình 4-10. Biểu đồ thực thể - liên kết cho cơ sở dữ liệu của ứng dụng

Chú ý rằng một NoteBlock có thể có dạng Text và Survey, tuy nhiên do nội dung của Text và Survey được lưu trữ theo các dạng dữ liệu khác nhau nên khi xây dựng cơ sở dữ liệu, nhóm sẽ xây dựng 2 bảng tương ứng.

Cuối cùng, từ việc phân tích các mối quan hệ một – nhiều, nhiều – nhiều và các thuộc tính đa trị, nhóm xây dựng cơ sở dữ liệu trên hệ quản trị CSDL MySQL với các bảng dữ liệu sau:

Bảng thứ nhất là admins(**username**, password). Bảng thứ hai là users(**username**, password, name, birthday, gender, school, is_locked, *admin*), trong đó admin là khóa ngoài đến bảng admins chỉ admin manage user tương ứng. Bảng thứ ba là notes(**note_id**, *author*, header, last_modified_date, is_public) trong đó author là khóa ngoài đến bảng users chỉ user sở hữu note này. Bảng thứ 4 là note_filters(***note_id***, **filter**). Bảng thứ 5 là sharenotes(***note_id***, ***editor***, share_type), trong đó note_id và editor là khóa ngoài tới notes và users để chỉ xem sharenote này có note gốc là note nào và bây giờ đang được sửa bởi editor nào. Bảng thứ 6 là note_blocks(**block_id**, *note_id*, header, block_type, block_ord). Bảng thứ 7 là text_blocks(***block_id***, ***editor***, content). Bảng thứ 8 là survey_blocks(***block_id***, ***editor***, survey_map).

4.3 Xây dựng ứng dụng

4.3.1 Thư viện và công cụ sử dụng

Mục đích	Công cụ	Địa chỉ URL
IDE lập trình	Apache Netbeans IDE 21	https://netbeans.apache.org/
Hệ quản trị CSDL	MySQL 8.0	https://www.mysql.com/
Driver hỗ trợ JDBC tới MySQL	MySQL Connector J 8.3.0	https://dev.mysql.com/downloads/connector/j/
API hỗ trợ gửi email	Mailjet Client 5.2.5	https://github.com/mailjet/mailjet-apiv3-java/
API hỗ trợ đọc ghi PDF	itextpdf 5.5.13.3	https://itextpdf.com/
Bộ công cụ lập trình giao diện	JavaFX SDK 21	https://gluonhq.com/
Công cụ hỗ trợ thiết kế giao diện bằng kéo thả	Scene Builder 21	https://gluonhq.com/

Bảng 4-1. Danh sách thư viện và công cụ sử dụng

4.3.2 Kết quả đạt được

Ứng dụng được đóng gói thành file NoteApp.jar, sẽ chạy trên máy của người dùng để cung cấp các dịch vụ.

Sau đây là một số kết quả đạt được về mã nguồn

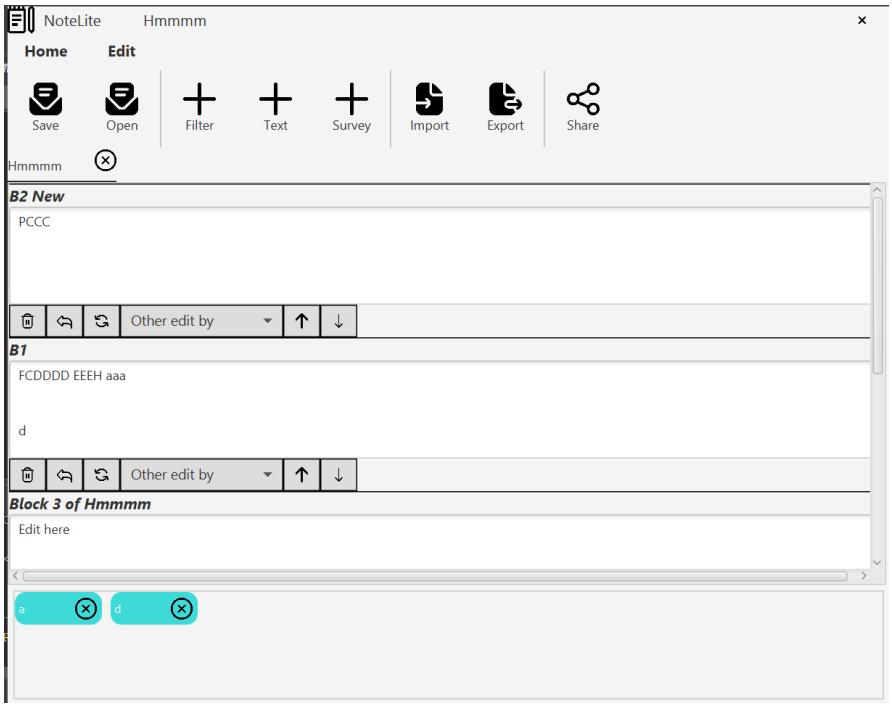
Thước đo	Kết quả
Tổng số dòng	8163
Số lượng dòng code	4598
Tỉ lệ comments	19.7%
Tổng số files mã nguồn (.java)	78

Tổng số lớp	105
Số lượng phương thức trung bình trên 1 class	5.6
Số lượng dòng code trung bình trên 1 method	4.76
Độ phức tạp lớn nhất (Max Complextity)	11
Dung lượng toàn bộ mã nguồn	421KB

Bảng 4-2. Kết quả đạt được về mã nguồn

4.3.3 Minh hoạ các chức năng chính

Trong phần này, nhóm sẽ minh họa cho 2 chức năng chính của ứng dụng, đó là chức năng edit một note được chia sẻ và chia sẻ note

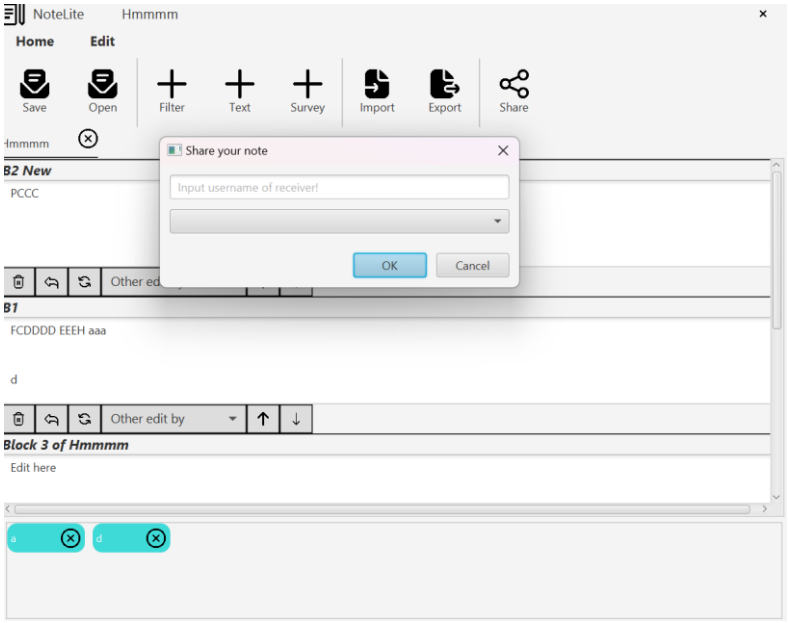


Hình 4-11. Màn hình edit Note

Phía trên là màn hình chính cho trang edit. Phần chính của trang này sẽ gồm các block dạng text hay survey. Với mỗi block dạng text, người dùng có thể nhận thông báo và xem chỉnh sửa của người khác qua “Other Edit By” box. Chúng sẽ được tự động cập nhật.

Tiếp theo là chức năng chia sẻ Note. Trên màn hình Edit Note ở hình 4-11, khi người dùng ấn vào nút Share, một Dialog sẽ xuất hiện đè lên và yêu cầu người dùng nhập các thông tin

cần thiết. Nếu người dùng nhập xong và nhấn OK thì nó sẽ gọi service share để share note này tới receiver tương ứng.



Hình 4-12. Dialog xuất hiện khi ấn nút Share

4.4 Triển khai

Ứng dụng của nhóm được triển khai thử nghiệm trên máy local nơi chứa Cơ sở dữ liệu.

Cuối cùng, để đánh giá lại các giải pháp cũng như kết quả đạt được sau khi phát triển ứng dụng, chương 5 sẽ tổng kết lại những điều này.

Chương 5 Kết luận và hướng phát triển

Trong 3 chương 2,3,4, nhóm đã lần lượt trình bày về các chức năng, các công nghệ và các giải pháp thiết kế. Trong phần này, nhóm sẽ tổng hợp lại một số ý quan trọng của các điều trên.

5.1 Kết luận

Thứ nhất, về việc thiết kế hệ thống và tổ chức mã nguồn, ứng dụng được xây dựng dựa trên 5 nguyên lý chính của lập trình hướng đối tượng (SOLID). Các giải pháp được đề ra để thiết kế ứng dụng trong chương 4 đều để đảm bảo 5 nguyên lý này.

Ở trong chương 4, mỗi package đại diện cho một module xử lý hầu hết đều có một interface. Interface này để đảm bảo giao tiếp giữa các module là thông qua interface, chứ không phải implementation của nó. Điều này giúp đảm bảo nguyên lý đóng mở (OCP) và nguyên lý đảo ngược phụ thuộc (DIP). Chúng giúp tăng tính tái sử dụng của code.

Ngoài ra, việc chỉnh sửa lại kiến trúc phân tầng cho phù hợp với các nghiệp vụ của ứng dụng cũng giúp mã nguồn hoạt động hiệu quả hơn, khi một tầng không phải đảm nhiệm quá nhiều nghiệp vụ.

Đồng thời, độ phức tạp của mỗi hàm không quá cao, do sau khi hoàn thành phần lớn mã nguồn của dự án, nhóm đã tiến hành phân tích độ phức tạp và tách hàm để đơn giản hóa các hàm phức tạp hơn.

Thứ hai, về kết quả thực hiện các chức năng, tất cả các chức năng quan trọng đã đề ra đều hoạt động tốt, kiểm tra được dữ liệu đầu vào, thông báo lỗi. Cho đến nay, ứng dụng của nhóm phần lớn vẫn hoạt động bình thường, chưa xảy ra lỗi. Tuy nhiên, có thể khi triển khai trên quy mô lớn hơn sẽ có một số vấn đề xảy ra do lưu lượng truy cập CSDL,…

5.2 Hướng phát triển

Sau sự tìm hiểu các tài liệu và không ngừng hoàn thiện, ứng dụng đã được hoàn thiện các chức năng chính và hoạt động thử nghiệm hiệu quả.

Ứng dụng đã thực hiện được các mục tiêu ban đầu của bài tập lớn, vận dụng được các kiến thức đã học để hoàn thiện, thiết kế.

Ứng dụng còn có thể phát triển nhiều trong tương lai vì còn nhiều chức năng chưa được khai thác, ví dụ như vòng bạn bè, quảng bá Note, tìm các tài liệu liên quan đến chủ đề của Note qua các nguồn Internet, thêm các thành phần Note đa dạng khác như hình ảnh, video, âm thanh, liên kết,…

Như vậy, mặc dù vẫn còn nhiều thiếu sót như hiệu năng của ứng dụng, chưa phục vụ được các service cao hơn nhưng nhìn chung, ứng dụng đã đáp ứng được các yêu cầu đề ra và được nhóm hoàn thiện một cách hiệu quả.