

**ĐẠI HỌC BÁCH KHOA HÀ NỘI**  
**TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**



**BÁO CÁO BÀI TẬP LỚN**  
**LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG**

**Giảng viên hướng dẫn: TS. Trần Thế Hùng**

**Đề tài: Web học tập flashcard**

**Lớp: 156772 – IT3103**

**Nhóm: 03**

Họ và Tên	MSSV	Email	Lớp
Đỗ Xuân Quý	20235818	quy.dx235818@sis.hust.edu.vn	Việt Nhật 02-K68
Ma Ngọc Thắng	20235829	thang.mn235829@sis.hust.edu.vn	Việt Nhật 02-K68
Vì Hùng Vương	20235881	vuong.vh235881@sis.hust.edu.vn	Việt Nhật 02-K68
Nguyễn Thị Khánh Vân	20235869	van.ntk235869@sis.hust.edu.vn	Việt Nhật 02-K68
Nguyễn Đức Thịnh	20235841	thinh.nd235841@sis.hust.edu.vn	Việt Nhật 02-K68
Nguyễn Quốc Cường	20235667	cuong.nq235667@sis.hust.edu.vn	Việt Nhật 02-K68

***Hà Nội, ngày 19 tháng 6 năm 2025***

**Lời cảm ơn**

Nhóm xin chân thành cảm ơn **TS. Trần Thế Hùng** đã tận tình hướng dẫn và hỗ trợ trong suốt quá trình thực hiện đề tài. Nhờ sự định hướng rõ ràng và những góp ý chuyên môn từ thầy, nhóm đã có cơ hội áp dụng kiến thức đã học vào xây dựng một sản phẩm cụ thể. Nhóm cũng cảm ơn nhà trường đã tạo điều kiện học tập và phát triển dự án. Cuối cùng, xin cảm ơn các thành viên trong nhóm đã phối hợp chặt chẽ và làm việc nghiêm túc để hoàn thành bài tập lớn đúng tiến độ.

## Tóm tắt

Đề tài tập trung xây dựng một website học từ vựng bằng flashcard nhằm hỗ trợ người học ghi nhớ hiệu quả thông qua thuật toán lặp lại ngắt quãng (SRS) và tích hợp trí tuệ nhân tạo (AI). Hệ thống cho phép người dùng:

- Tạo bộ thẻ, thêm/sửa/xoá nội dung thẻ.
- Tùy chỉnh trải nghiệm học tập, luyện tập theo tiến độ cá nhân, cũng như tìm kiếm và chia sẻ tài nguyên học.

Về kiến trúc tổng quan của sản phẩm:

- Backend được xây dựng bằng Python có tích hợp AI hỗ trợ sinh nội dung thẻ và giải đáp thắc mắc thông qua chatbot, sử dụng API của Google Gemini.
- Phần xử lý nghiệp vụ và truy vấn cơ sở dữ liệu (database, CSDL) được thực hiện bằng Java với thiết kế hướng đối tượng cùng hệ thống Query Builder tự phát triển.
- Dữ liệu được lưu trữ trong PostgreSQL và giao diện được thiết kế bằng NiceGUI – một framework dựa trên FastAPI.

Kết quả cho thấy hệ thống đáp ứng đầy đủ các chức năng đặt ra, hoạt động ổn định và có khả năng mở rộng, như tích hợp hình ảnh, âm thanh, thêm dạng bài luyện tập khác và chế độ học offline. Đề tài mang tính ứng dụng cao trong thực tế học tập, đồng thời giúp các thành viên củng cố kiến thức về lập trình hướng đối tượng, thiết kế hệ thống, xử lý dữ liệu, tích hợp AI và nâng cao kỹ năng làm việc nhóm trong dự án phần mềm hoàn chỉnh.

# Mục lục

1.1	Mô tả bài toán .....	5
1.2	Quy trình nghiệp vụ.....	5
1.3	Chức năng hệ thống.....	5
2	Công nghệ & thư viện sử dụng .....	7
2.1	Danh sách công nghệ:.....	7
2.2	Cài đặt môi trường.....	7
2.3	Công nghệ API của Google Gemini trong Dự án .....	8
3	Thiết kế hệ thống.....	10
3.1	Biểu đồ Use Case .....	10
3.2	Biểu đồ lớp .....	11
3.3	Biểu đồ ERD .....	15
4	Cấu trúc chương trình .....	16
4.1	Kiến trúc tổng quan của hệ thống.....	16
4.2	Các lớp .....	16
4.3	Mối liên hệ giữa các lớp.....	17
4.4	Các nguyên lý OOP.....	19
4.5	Tương tác cơ sở Dữ liệu và Query Builder:.....	23
4.6	Phương pháp tính thời gian (số ngày) đến lần ôn tập tiếp theo - quản lý tiến độ học tập .....	26
4.7	Demo .....	30
5	Kết luận.....	36
	Tài liệu tham khảo .....	38

Đánh giá thành viên:

Họ và Tên	Công việc được giao	Mức độ hoàn thiện
Đỗ Xuân Quý	Nhóm trưởng, giao nhiệm vụ, lựa chọn công nghệ và quán xuyến tiến độ nhóm Thiết kế phần front-end bằng NiceGUI và xây dựng server Python Bản luận thiết kế hệ thống, thiết kế mô hình OOP	Xuất sắc
Ma Ngọc Thắng	Thiết kế chính prompt sinh data cho mặt sau của thẻ Nghiên cứu phương pháp SRS và xây dựng cách tính deadline ôn tập cho thẻ	Tốt
Vì Hùng Vương	Nghiên cứu, kiểm thử Front_end bằng JavaFX, NiceGUI. Xây dựng SQL Query dạng thô để xây dựng thành SQL Query Builder Nghiên cứu 4 tính chất của OOP và cách xây dựng ORM, Query Builder để làm báo cáo và slide	Xuất sắc
Nguyễn Thị Khánh Vân	Nghiên cứu cơ sở dữ liệu Viết chính báo cáo và làm slide trình bày sản phẩm	Tốt
Nguyễn Đức Thịnh	Nghiên cứu cơ sở dữ liệu Vẽ sơ đồ UseCase và sơ đồ các lớp	Xuất sắc
Nguyễn Quốc Cường	Đánh giá và kiểm thử prompt sinh data cho mặt sau của thẻ Hỗ trợ làm báo cáo và slide	Trung bình

## 1 Giới thiệu đề tài

## 1.1 Mô tả bài toán

Hệ thống học từ vựng qua flashcard này được xây dựng nhằm hỗ trợ người học ngoại ngữ ghi nhớ từ vựng hiệu quả hơn thông qua cơ chế lặp lại ngắt quãng (SRS).

Người dùng có thể tạo bộ từ vựng cá nhân, luyện tập thông minh, theo dõi tiến độ ghi nhớ từng từ và tùy chỉnh trải nghiệm học tập với các cài đặt cá nhân như ngôn ngữ, tốc độ học và font chữ.

Hệ thống còn cho phép chia sẻ hoặc bảo mật các bộ thẻ thông qua hệ thống tag như "public" hoặc "private", giúp người học dễ dàng tìm kiếm hoặc chia sẻ tài nguyên phù hợp.

## 1.2 Quy trình nghiệp vụ

- **Người dùng đăng ký / đăng nhập:** Nhập thông tin email, mật khẩu để tạo tài khoản hoặc truy cập hệ thống
- **Tạo bộ từ vựng (deck):** Người dùng nhập tên deck, tên deck, chọn ngôn ngữ, chọn tag (public/private, có thể thêm nhiều tag)
- **Thêm thẻ từ vựng (card):**
  - Mỗi deck có thể chứa nhiều card
  - Mỗi flashcard gồm: mặt trước (từ vựng hoặc ngữ pháp), mặt sau (nghĩa hoặc giải thích)
  - Người dùng có thể chỉnh sửa hoặc xóa card
- **Luyện tập học từ:**
  - Hệ thống hiển thị flashcard ngẫu nhiên theo thuật toán SRS
  - Hệ thống lưu lại vào tiến độ học tập
- **Theo dõi tiến độ:** Người dùng có thể xem tiến trình học của mình (đã học bao nhiêu từ, tần suất nhớ, v.v.)
- **Tùy chỉnh cài đặt:** Người dùng có thể điều chỉnh font chữ, tốc độ học,...
- **Tìm kiếm và chia sẻ deck:**
  - ❖ Tìm kiếm deck theo tag
  - ❖ Deck public có thể được người khác tìm thấy và học nhưng không thể sửa nếu không phải là tác giả

## 1.3 Chức năng hệ thống

### a. Chức năng dành cho người dùng

Người dùng có thể đăng ký hoặc đăng nhập vào hệ thống để tạo các deck, đặt tên, chọn ngôn ngữ và thiết lập chế độ công khai hoặc riêng tư.

Trong mỗi deck, họ có thể thêm, sửa, xóa các card. Trong đó card bao gồm mặt trước và mặt sau. Các deck có thể được gán tag từ danh sách có sẵn nhằm phân loại và tổ chức hợp lý.

Hệ thống hỗ trợ chế độ luyện tập dạng flashcard, nơi người dùng tự đánh giá mức độ ghi nhớ từng từ. Thông tin này được lưu lại để đề xuất thời điểm ôn tập phù hợp theo thuật toán.

Người dùng có thể theo dõi tiến độ học của từng từ và cấu hình tốc độ học cá nhân như số thẻ mỗi ngày. Ngoài ra, họ có thể tìm kiếm deck theo tag để dễ dàng quản lý và truy cập nội dung học tập.








## **b. Chức năng hệ thống**

Hệ thống có nhiệm vụ lưu trữ toàn bộ dữ liệu liên quan đến người dùng, bộ từ vựng, thẻ từ, tag, tiến độ học và cài đặt học tập cá nhân. Khi người dùng luyện tập, hệ thống tự động áp dụng thuật toán lặp lại ngắt quãng để xác định và đề xuất các thẻ cần ôn tập, giúp cải thiện khả năng ghi nhớ.

Khi tạo deck, nếu người dùng không chọn chế độ công khai, hệ thống sẽ tự gán trạng thái riêng tư, đồng thời đảm bảo mỗi deck đều có ít nhất một tag để đáp ứng yêu cầu phân loại và giúp người dùng dễ dàng trong việc tìm kiếm bộ thẻ.

## 2 Công nghệ & thư viện sử dụng

### 2.1 Danh sách công nghệ:

Tên công nghệ	Mục đích sử dụng	Ưu điểm
Astah UML 	Vẽ sơ đồ mô hình hóa hệ thống như sơ đồ lớp, Use Case	Giao diện dễ dùng; hỗ trợ nhiều loại sơ đồ UML; xuất file hình ảnh nhanh chóng
PostgreSQL 	Quản lý cơ sở dữ liệu lưu trữ người dùng, bộ thẻ, thẻ từ, tiến độ học, cài đặt	Hệ quản trị CSDL mã nguồn mở mạnh mẽ; hỗ trợ tốt truy vấn phức tạp; đảm bảo tính toàn vẹn
Java 	Xây dựng logic nghiệp vụ và truy vấn cơ sở dữ liệu	Mạnh mẽ, là ngôn ngữ thuần OOP, phù hợp với nội dung học phần, dễ mở rộng
Python 	Xử lý backend, tích hợp AI (LLM) và kết nối với cơ sở dữ liệu	Cú pháp ngắn gọn; hỗ trợ nhiều thư viện mạnh mẽ (AI, DB, GUI...)
NiceGUI 	Thiết kế giao diện web dựa trên Python	Dễ sử dụng, đặc biệt đối với người mới
GitHub 	Quản lý mã nguồn, phối hợp làm việc nhóm, theo dõi lịch sử	Quản lý source code dễ dàng, cho phép nhiều người cùng làm việc trên 1 dự án; tính năng pull request
Gemini 	Mô hình ngôn ngữ lớn (LLM) của Google	Tự động sinh mặt sau của thẻ và xây dựng chatbot trợ lý ảo thông minh cho web.

### 2.2 Cài đặt môi trường

Để triển khai và chạy được hệ thống, cần thiết lập các công cụ và thư viện sau:

#### Môi trường phát triển

##### - IDE:

- Java: IntelliJ IDEA hoặc Eclipse, JDK 17+
- Python: Phiên bản 3.10+, phát triển trên Visual Studio Code

- **Thư viện Py4J:** hỗ trợ kết nối giữa Java và Python
- **Trình duyệt:** Google Chrome, Cốc Cốc,....

**Kết nối AI:** Sử dụng API Gemini từ Google

- Cần có API key hợp lệ trong ProtectedData.py
- Thư viện dùng: google-generativeai

**Cơ sở dữ liệu:** PostgreSQL 14+, một hệ quản trị cơ sở dữ liệu mã nguồn mở mạnh mẽ, được sử dụng trong nhiều ứng dụng thực tế.

### Giao diện người dùng

- **Framework (NiceGUI):** là một thư viện Python mã nguồn mở dùng để viết giao diện người dùng đồ họa chạy trên trình duyệt. Thư viện này dễ học, dễ làm quen trong khi vẫn cung cấp tùy chọn cho các tùy chỉnh nâng cao. NiceGUI tuân theo triết lý "**backend-first**": nó xử lý tất cả các chi tiết phát triển web. Bạn có thể tập trung vào việc viết mã Python.
- NiceGUI render code python -> Vue JS

### 2.3 Công nghệ API của Google Gemini trong Dự án

- Trong khi từ điển thông thường là một công cụ tra cứu hữu ích, dự án Flashcard của chúng tôi, với sự tích hợp mạnh mẽ của LLM (Google Gemini), đã vượt xa một cuốn từ điển truyền thống, mang lại trải nghiệm học tập và tương tác năng động hơn.
- **So sánh LLM với từ điển:**

Tính năng	Từ điển	Flashcard
Vai trò	Tra cứu thụ động	Học tập chủ động, tương tác
Nội dung	Ngắn gọn súc tích, nhưng không cá nhân hóa	Phong phú, chi tiết, đa dạng
Tạo nội dung	Không có	Tạo nội dung tự động
Tính cập nhật	Không có	Tính cập nhật linh hoạt: học hỏi và cải thiện liên tục

### - Sinh data cho Card\_back:

- **Mục tiêu:** Tự động tạo nội dung của Card\_back với class BackFlashcardPrompt.
- **Cơ chế:**



- + Nhận thông tin đầu vào Card\_front.
  - + Tạo prompt.
  - + Gọi API Gemini để sinh data.
  - + Nội dung Card\_back nhận được sẽ được lưu trữ vào DB.
  - + Hiện thị nội dung đó trên giao diện người dùng.
- **Điểm mạnh:** LLM có khả năng phân tích và tổng hợp thông tin từ hàng tỷ văn bản, cho phép nó tạo ra các giải thích chi tiết, chính xác và có ngữ cảnh cho mặt sau của flashcard. Nó không chỉ đơn thuần định nghĩa thuật ngữ mà còn cung cấp các ví dụ đa dạng, trường hợp sử dụng, và thông tin liên quan, giúp người học hiểu sâu hơn thay vì chỉ ghi nhớ.

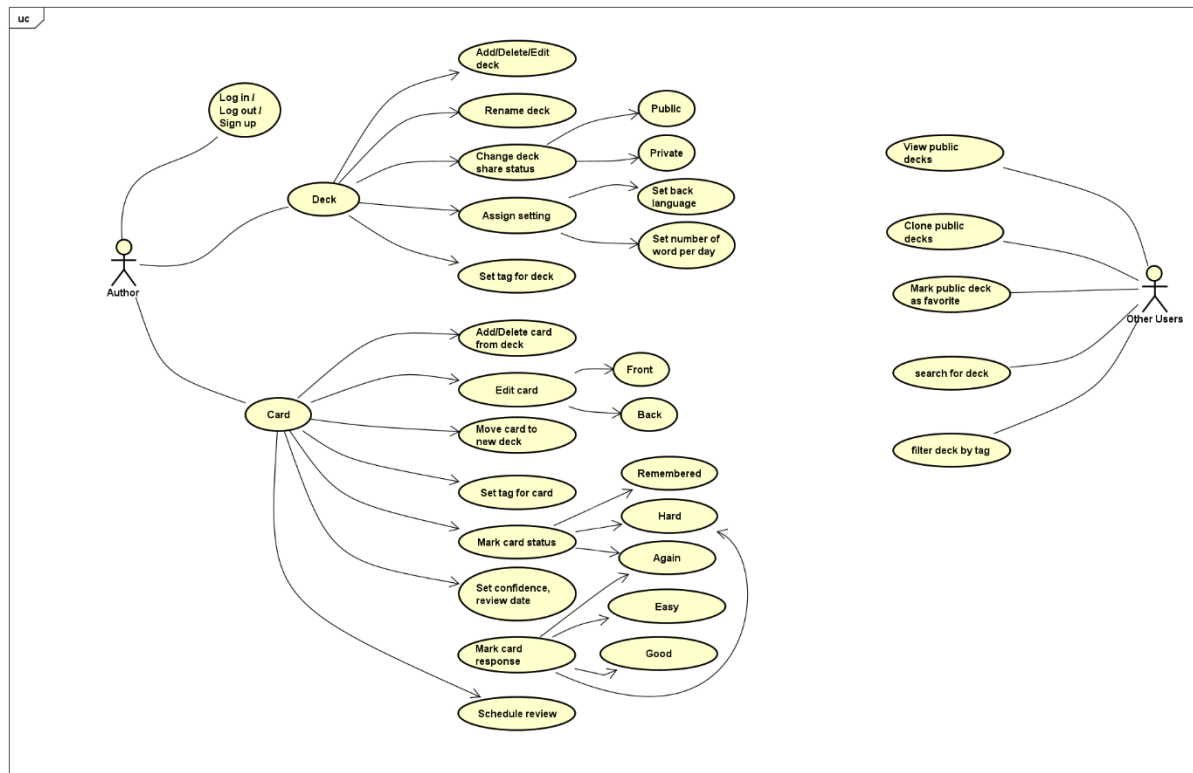
**CHÚ Ý:** Chỉ tác giả của bộ thẻ mới có thể sinh mặt sau của thẻ theo cách này.

#### - Chatbot:

- **Mục tiêu:** trợ lý ảo thông minh cho web. Người dùng có thể tương tác để đặt câu hỏi, tìm kiếm,.....
- **Cơ chế:**
  - + Người dùng nhập nội dung.
  - + Tầng backend sẽ gọi tới Gemini
  - + Gemini xử lý nội dung và trả data về cho tầng backend.
  - + Hiện thị data vừa tạo ra khung chat.
- **Điểm mạnh:** Khả năng hiểu các câu hỏi phức tạp, mơ hồ hoặc có nhiều ý nghĩa của người dùng. Gemini có thể diễn giải ý định của người dùng ngay cả khi câu hỏi được đặt một cách tự nhiên, không theo khuôn mẫu. Nó cũng tạo ra các phản hồi trôi chảy, logic và phù hợp với ngữ cảnh của cuộc trò chuyện, làm cho tương tác trở nên tự nhiên và hiệu quả.

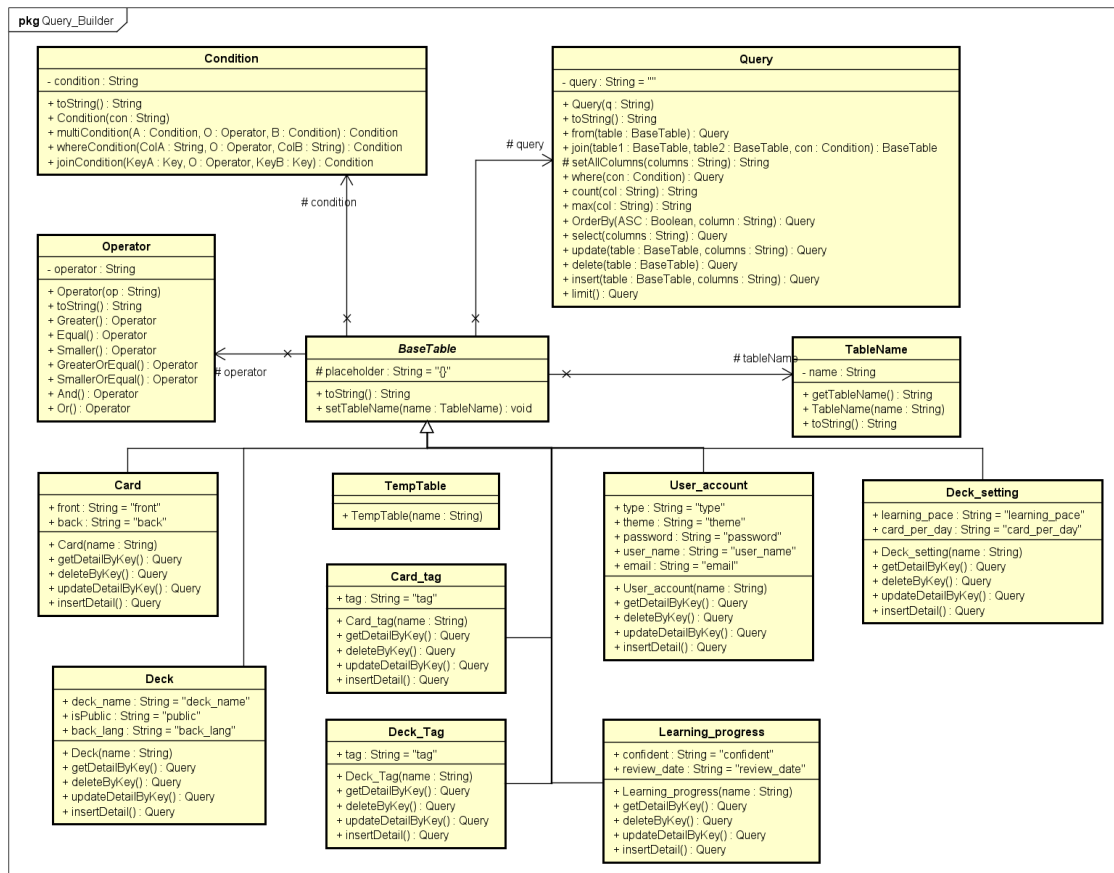
# 3 Thiết kế hệ thống

## 3.1 Biểu đồ Use Case

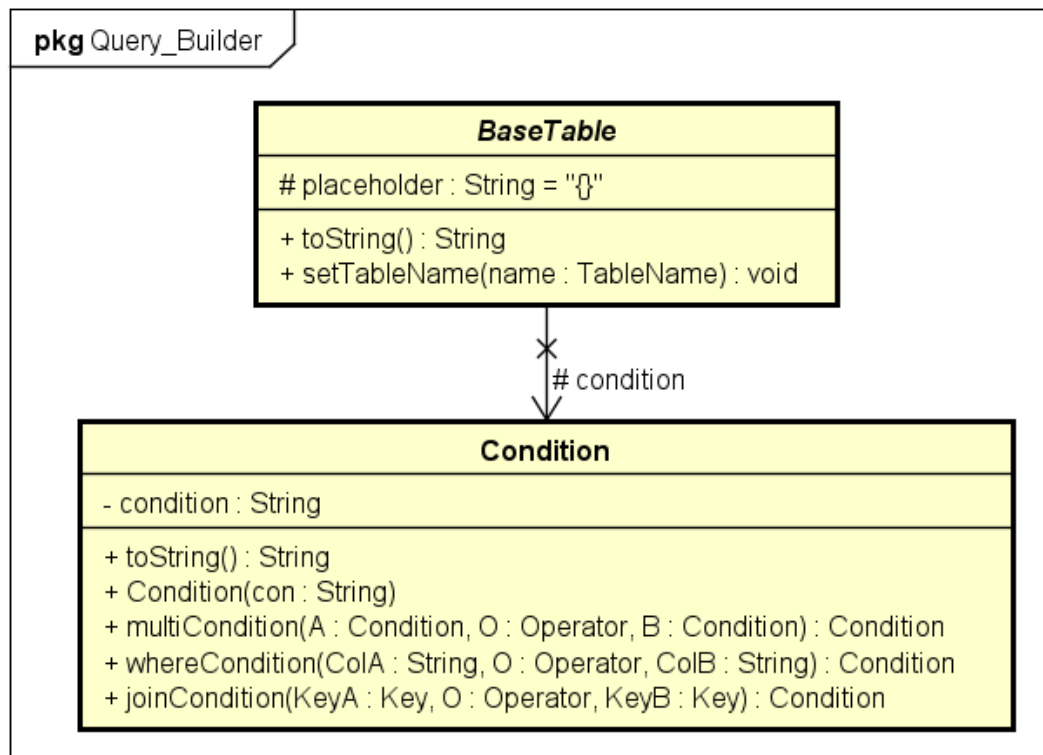


## 3.2 Biểu đồ lớp

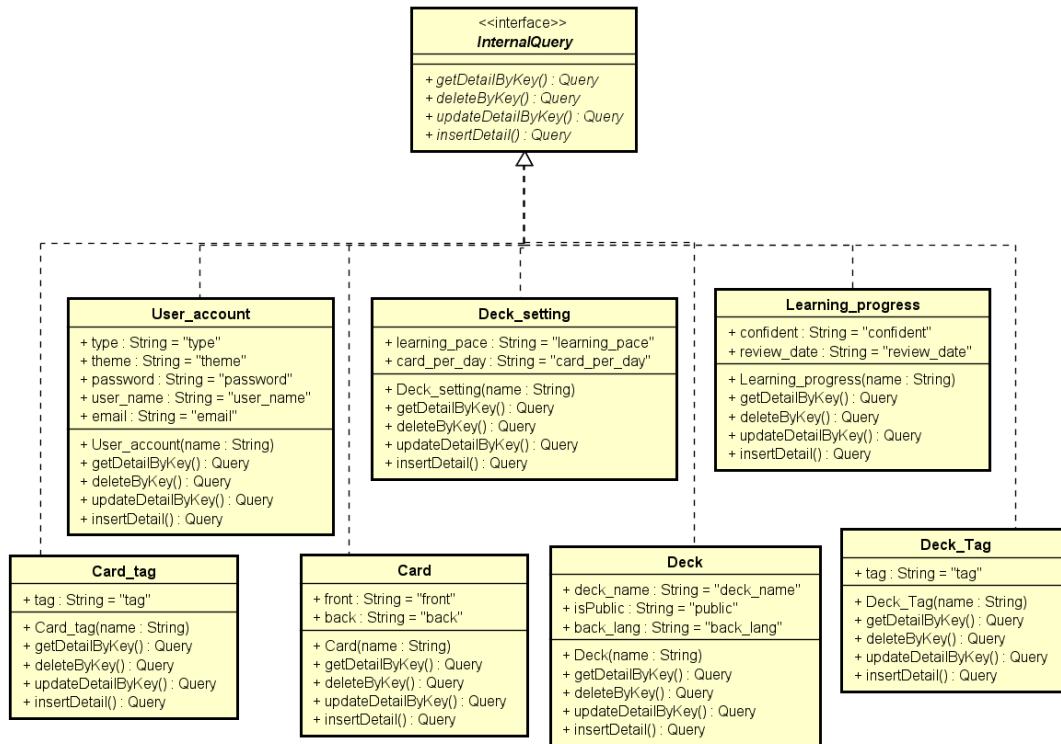
Các biểu đồ lớp trong package Query\_Bulider:



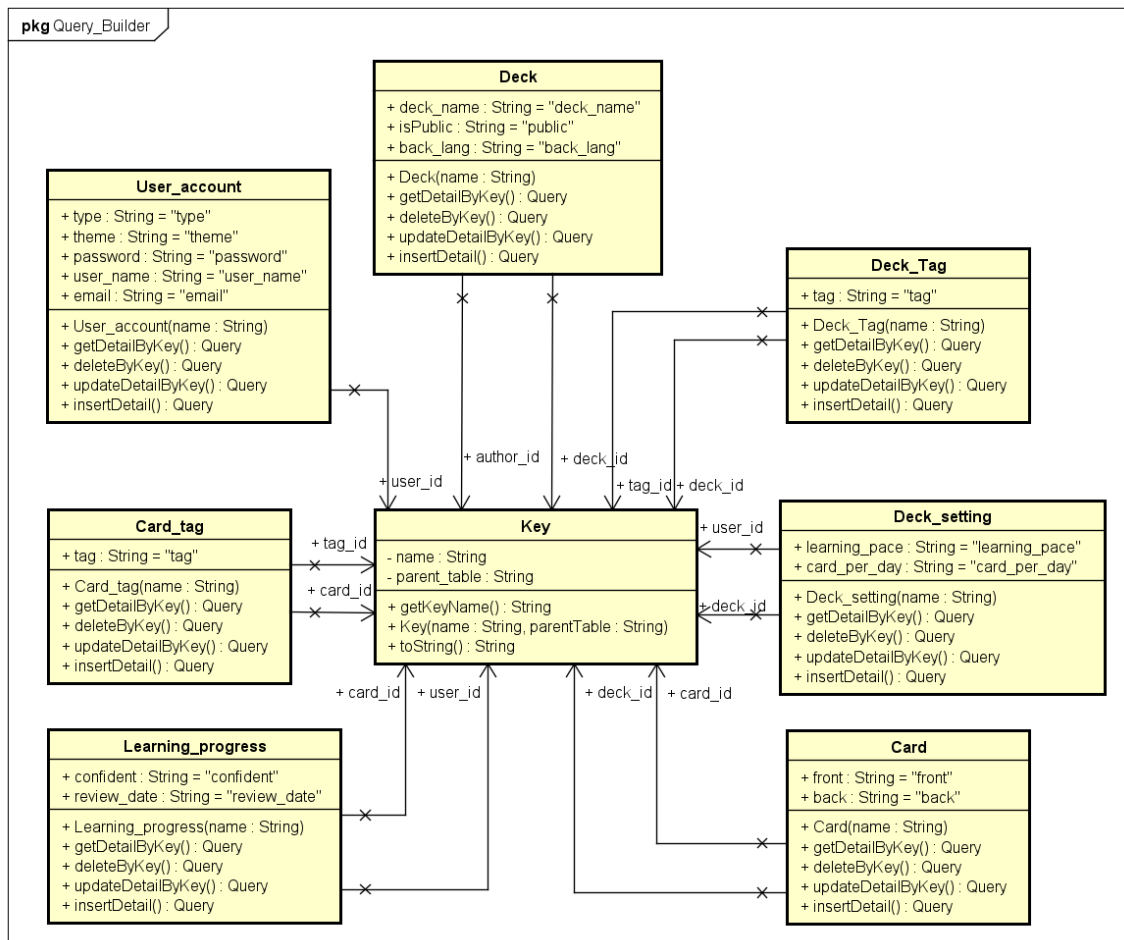
Class: BaseTable



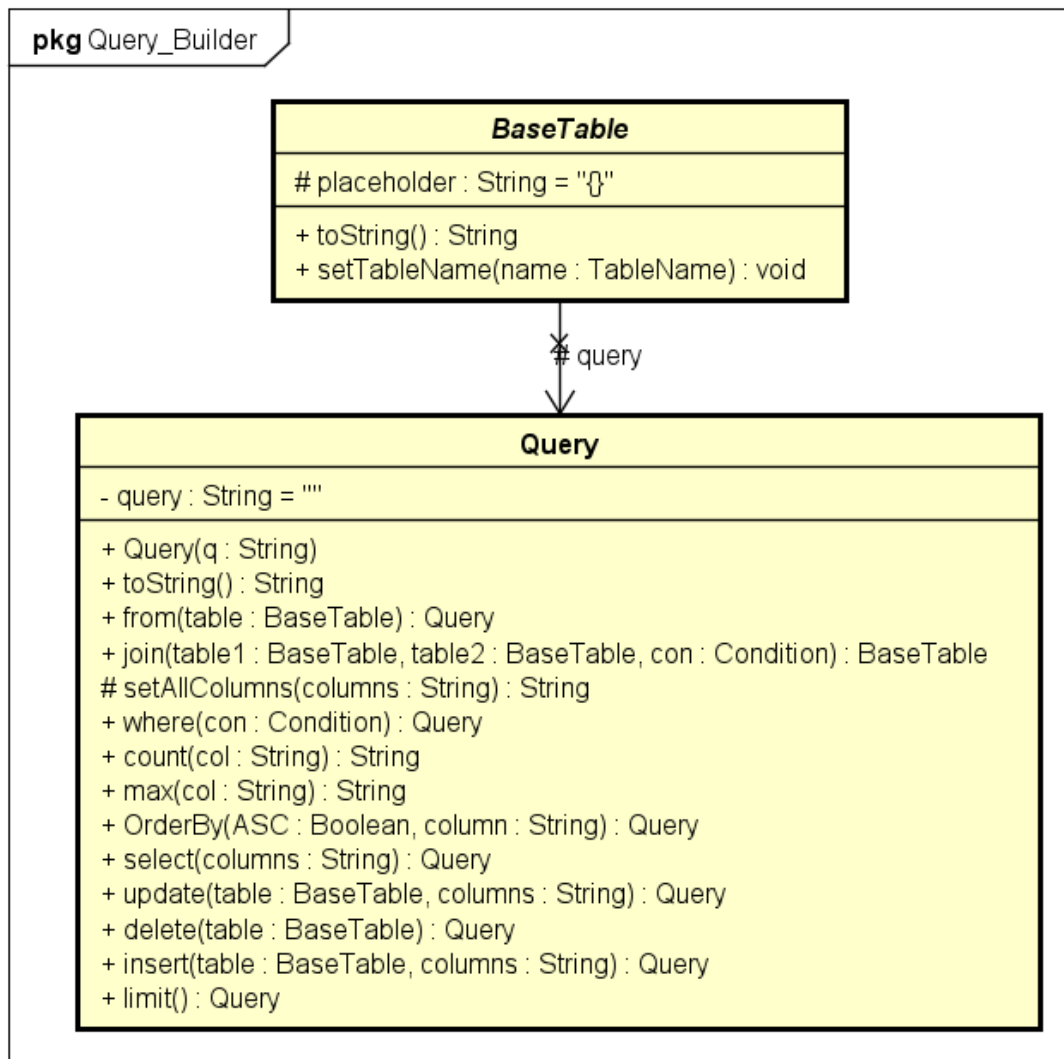
Class: Condition



## Interface: InternalQuery

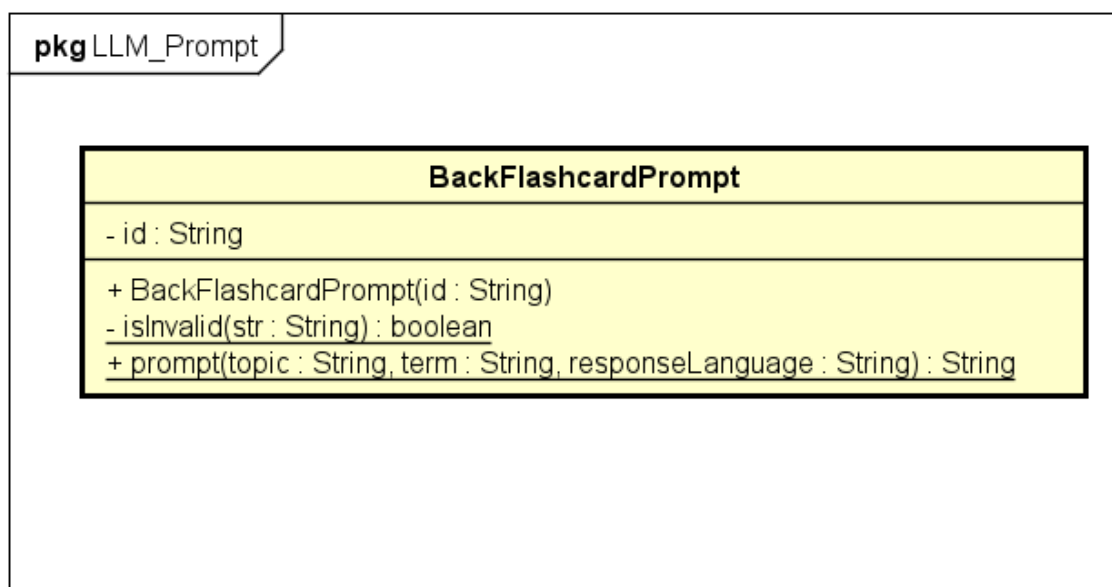


Class: Key



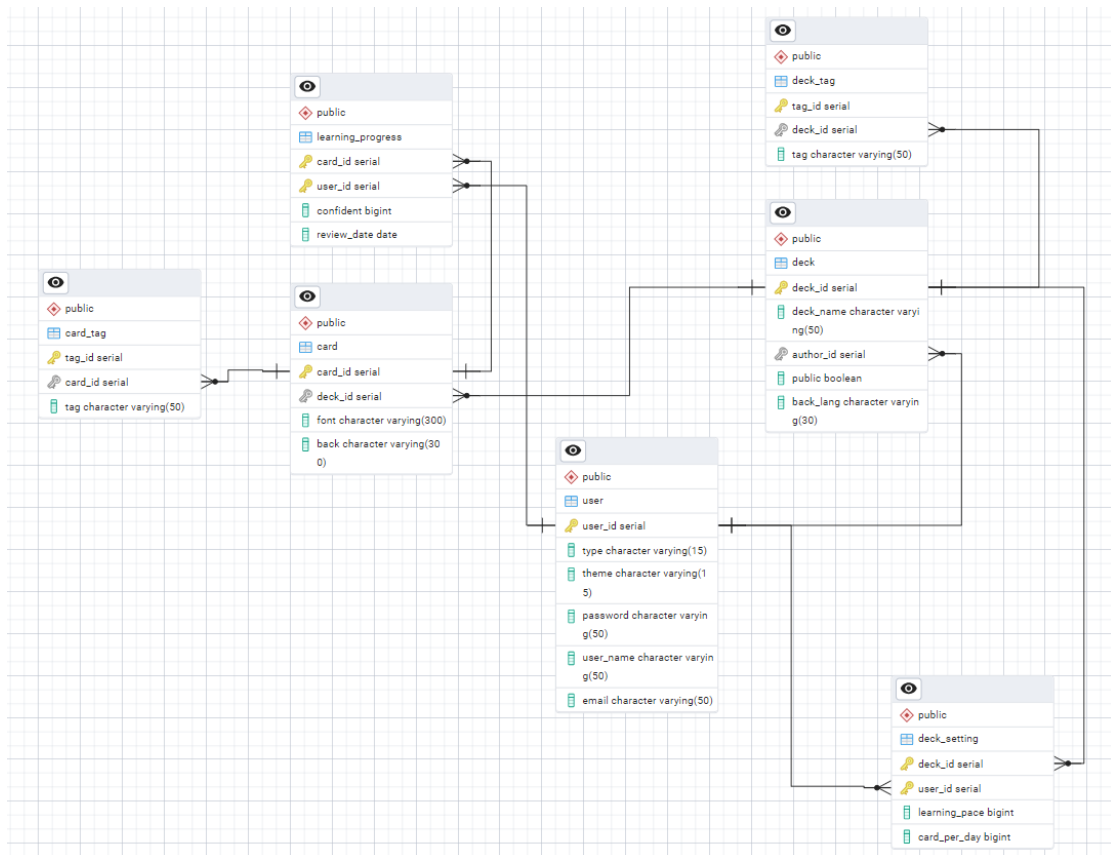
Class: Query

Biểu đồ lớp trong package LLM\_Prompt:



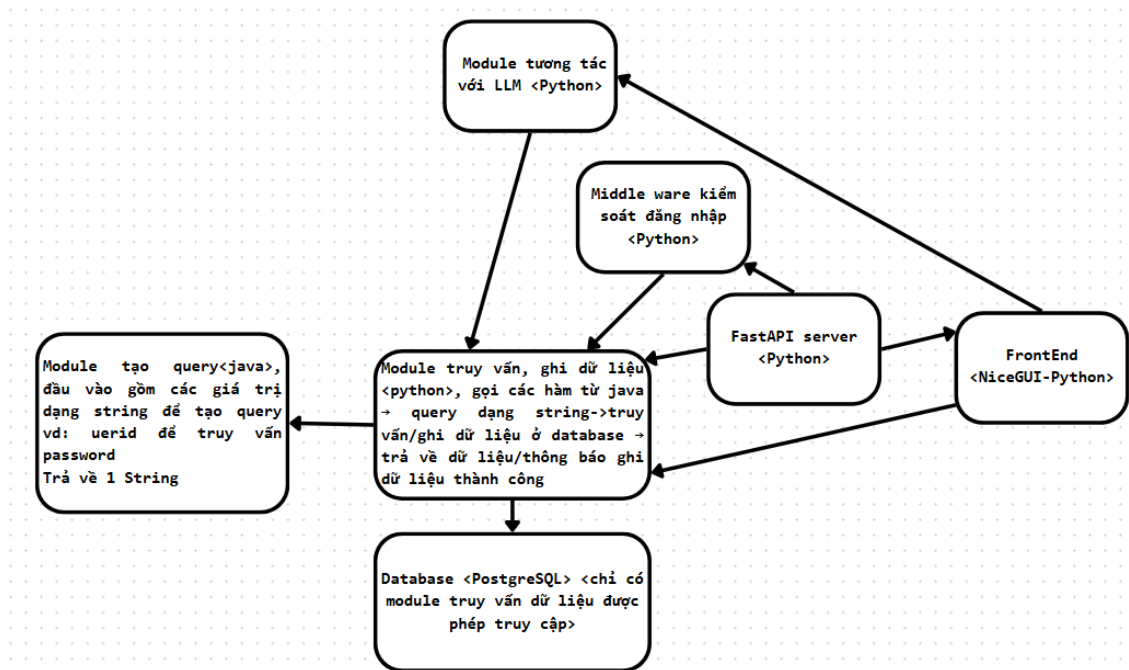
Class: BackFlashcardPrompt

### 3.3 Biểu đồ ERD

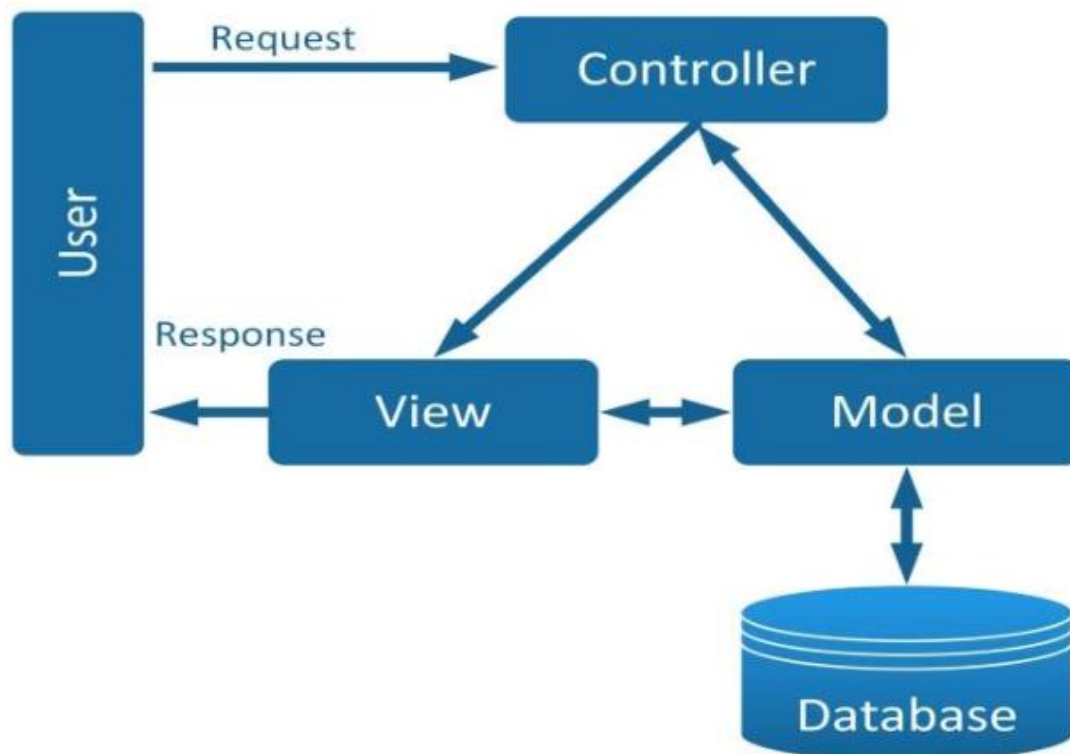


## 4 Cấu trúc chương trình

### 4.1 Kiến trúc tổng quan của hệ thống



Kiến trúc của hệ thống flashcard là tiêu biểu cho kiến trúc MVC (Model-View-Controller).



### 4.2 Các lớp

- a. **Package Query\_Builder**: Xây dựng truy vấn SQL
  - **Query.java**: Lớp xây dựng câu lệnh SQL.



- BaseTable.java: Lớp trừu tượng cho các bảng.
  - InternalQuery.java: Giao diện định nghĩa hành vi CRUD của riêng table đó
  - Condition, Operator.java, Key.java, TableName.java: Các lớp xây dựng điều kiện và quản lý định danh.
  - **Card.java, Deck.java, ....**: Các lớp đại diện cho từng Table.Package LLM\_Prompt:
- b. BackFlashcardPrompt.: Lớp tạo prompt sinh back\_card cho flashcard.

## 4.3 Môi liên hệ giữa các lớp

### a. Quan hệ Kế thừa (Inheritance):

❖ **BaseTable (Lớp Cha Trừu tượng)**: Đây là lớp cha trừu tượng (abstract class) của tất cả các lớp đại diện cho các bảng trong cơ sở dữ liệu. Nó định nghĩa các thuộc tính và công cụ chung mà mọi bảng cần có để xây dựng truy vấn.

- Các thuộc tính chung: query (đối tượng Query), condition (đối tượng Condition), operator (đối tượng Operator), placeholder (chuỗi ký tự đại diện), và tableName (đối tượng TableName).
- Phương thức chung: toString() (được ghi đè để trả về tên bảng thông qua tableName.toString()).

### ❖ Các Lớp Bảng Cụ thể (Lớp Con):

- Card, Deck, User\_account, Card\_tag, Deck\_setting, Deck\_Tag, Learning\_progress, TempTable là các lớp con **kế thừa (extends)** từ BaseTable.
- Mỗi lớp con định nghĩa các thuộc tính riêng đại diện cho các cột của bảng tương ứng (ví dụ: front, back trong Card; deck\_name, isPublic trong Deck).
- Các lớp con này cũng khởi tạo các đối tượng Key riêng cho các khóa chính/ngoại của chúng, liên kết với tên bảng của chính nó.
- **Ví dụ**: Card extends BaseTable nghĩa là một đối tượng Card cũng "là một loại" BaseTable và tự động có các thành phần như query, condition, operator để sử dụng.

```
public class Card extends BaseTable
```

❖ **TempTable**: Đây là một lớp con đặc biệt của BaseTable, được sử dụng để đại diện cho các bảng tạm thời hoặc kết quả của các phép nối (JOIN), cho phép chúng được xử lý như các bảng thông thường trong quá trình xây dựng truy vấn.

b. **Quan hệ Triển khai Giao diện (Interface Implementation):**

❖ **InternalQuery (Giao diện):** Giao diện này định nghĩa một "hợp đồng" chuẩn cho các thao tác truy vấn cơ bản (CRUD) trên các bảng. Nó bao gồm các phương thức trừu tượng như `getDetailByKey()`, `deleteByKey()`, `updateDetailByKey()`, `insertDetail()`.

❖ **Các Lớp Bảng Cụ thể (implements InternalQuery):**

- Tất cả các lớp bảng cụ thể (`Card`, `Deck`, `User_account`, `Card_tag`, `Deck_setting`, `Deck_Tag`, `Learning_progress`) đều **triển khai (implements)** giao diện `InternalQuery`.
- Điều này buộc mỗi lớp phải cung cấp một cài đặt cụ thể cho từng phương thức trong giao diện, tùy thuộc vào cấu trúc và logic của bảng đó.
- **Ví dụ:** Cả `Card` và `Deck` đều có phương thức `getDetailByKey()`, nhưng cách thức chúng tạo ra câu lệnh SQL để lấy chi tiết sẽ khác nhau, phù hợp với cấu trúc cột của từng bảng.

```
@Override
public Query getDetailByKey() {
    return query.select(this.front,this.back)
                .from(this)
                .where(condition.whereCondition(this.card_id.toString(),operator.Equal(),placeholder));
}
```

```
@Override
public Query getDetailByKey() {
    return query.select(this.deck_name,this.isPublic,this.back_lang,this.author_id.toString())
                .from(this)
                .where(condition.whereCondition(this.deck_id.toString(),operator.Equal(),placeholder));
}
```

c. **Quan hệ Phụ thuộc/Sử dụng (Dependency/Usage):**

❖ **BaseTable sử dụng Query, Condition, Operator, TableName:**

- Trong `BaseTable`, các đối tượng `query`, `condition`, `operator` được khởi tạo và cung cấp cho các lớp con. Điều này cho phép các lớp bảng sử dụng chúng để xây dựng các phần của câu lệnh SQL.
- Mỗi `BaseTable` cũng có một đối tượng `TableName` để quản lý tên bảng.

❖ **Các Lớp Bảng sử dụng Key:**

- Các lớp bảng cụ thể (ví dụ Card, Deck) tạo và quản lý các đối tượng Key để đại diện cho các khóa chính và khóa ngoại của chúng.
- Các đối tượng Key này sau đó được sử dụng trong các phương thức của Query hoặc Condition để xây dựng điều kiện truy vấn chính xác (ví dụ: card\_id.toString()).

#### ❖ Query sử dụng BaseTable, Condition, Key:

- Các phương thức của lớp Query (như from(), update(), delete(), insert(), join(), where()) nhận các đối tượng BaseTable và Condition làm tham số để xây dựng chuỗi SQL.
- Phương thức joinCondition() trong Condition nhận các đối tượng Key để tạo điều kiện nối bảng.

#### ❖ Condition sử dụng Operator, Key và String:

- Lớp Condition có các phương thức như multiCondition(), whereCondition(), joinCondition() để xây dựng các điều kiện. Các phương thức này nhận vào các đối tượng Condition, Operator, String (tên cột) hoặc Key để kết hợp và tạo ra chuỗi điều kiện SQL.

#### ❖ Key sử dụng TableName (một cách gián tiếp):

- Đối tượng Key chứa parent\_table (là tên bảng dạng String), được sử dụng trong phương thức toString() của Key để tạo ra chuỗi có dạng "TableName.column\_name". Mặc dù parent\_table là String nhưng nó thường được lấy từ tên của đối tượng TableName liên quan.

## 4.4 Các nguyên lý OOP

### ❖ Kế thừa

- **Lớp BaseTable là lớp cha:** Tất cả các lớp đại diện cho các bảng trong cơ sở dữ liệu (như Card, Deck, User\_account, Deck\_setting, Card\_tag, Deck\_Tag, Learning\_progress, và TempTable) đều **kế thừa** từ lớp trừu tượng BaseTable (Kế thừa **query, condition, .....**).

```
public class Card extends BaseTable
```

```
public class Deck extends BaseTable
```

```
public class Deck_Tag extends BaseTable
```

```
public class Card_tag extends BaseTable
```

- **BaseTable:**

```
public abstract class BaseTable{
    protected Query query=new Query(q:"");
    protected Condition condition=new Condition(con:"");
    protected Operator operator=new Operator(op:"");
    protected String placeholder="{}";
    protected TableName tableName;
```

- Ví dụ lớp Card kế thừa Query, Condition,... từ lớp BaseTable:

```
public class Card extends BaseTable
@Override
public Query getDetailByKey() {
    return query.select(this.front,this.back)
        .from(this)
        .where(condition.whereCondition(this.card_id.toString(),operator.Equal(),placeholder));
}
```

- **Ưu điểm:**
  - Tái sử dụng mã: Tránh lặp lại code định nghĩa (query, condition, tableName) cho mỗi class table.
  - Cấu trúc: tạo cây phân cấp rõ ràng, dễ hiểu, dễ mở rộng khi có table mới.

## ❖ Đóng gói

- **Sử dụng private và protected:** Các biến quan trọng như query trong lớp Query, condition trong lớp Condition, name trong TableName, name và parent\_table trong Key đều được khai báo là **private** hoặc **protected**. Các thuộc tính này sẽ được thiết lập và gọi qua Getter và Setter.

```

public class Key{
    private final String name;
    private final String parent_table;
    public String getKeyName() {
        return this.name;
    }
    public Key(String name, String parentTable){
        this.name=name;
        this.parent_table = parentTable;
    }
    @Override
    public String toString(){
        return this.parent_table+"."+this.name;
    }

    public String toInsert(){
        return this.name;
    }
}

```

```

public class Query {
    private String query="";
    public Query(String q){
        this.query=q;
    }
    @Override
    public String toString(){
        return this.query;
    }
}

```

- **Ưu điểm:**
  - Bảo vệ dữ liệu: Ngăn chặn việc thay đổi dữ liệu một cách không hợp lệ từ bên ngoài lớp.
  - Dễ bảo trì: Cho phép thay đổi cài đặt bên trong lớp mà không ảnh hưởng đến các phần khác của hệ thống, miễn là giao diện công khai không thay đổi

## ❖ Đa hình

- **Thông qua interface InternalQuery:** Các đối tượng của các lớp bảng khác nhau (Card, Deck, User\_account, ....) đều triển khai giao diện InternalQuery với các hàm **getDetailByKey(), deleteByKey(), updateDetailByKey(), insertDetail()** được tính chỉnh theo từng đặc tính của mỗi table.
- **Minh chứng:** Mỗi lớp (Card, Deck, User\_account) cung cấp một phiên bản riêng của cùng một phương thức (getDetailByKey()).

```
@Override
public Query getDetailByKey() {
    return query.select(this.front,this.back)
        .from(this)
        .where(condition.whereCondition(this.card_id.toString(),operator.Equal(),placeholder));
}
```

```
@Override
public Query getDetailByKey() {
    return query.select(this.deck_name,this.isPublic,this.back_lang,this.author_id.toString())
        .from(this)
        .where(condition.whereCondition(this.deck_id.toString(),operator.Equal(),placeholder));
}
```

- **Ưu điểm:**

- Linh hoạt: Cho phép viết mã tổng quát hơn, không cần biết chính xác loại đối tượng cụ thể tại thời điểm biên dịch.
- Mở rộng dễ dàng: Khi thêm bảng mới, chỉ cần triển khai InternalQuery mà không làm thay đổi các phần code khác.

## ❖ Trừu tượng

- **Lớp trừu tượng BaseTable:**

- Ẩn chi tiết: BaseTable định nghĩa các thuộc tính và công cụ chung (query, condition, operator), đó là những công cụ cơ bản cho bất cứ thao tác truy vấn nào, không ép buộc các lớp con phải có những cột cụ thể nào hay cách chúng thực hiện CRUD.
- Tập trung vào "is a table": Người dùng chỉ cần biết BaseTable là nền tảng cho một "bảng", không cần biết chi tiết bên trong của các bảng cụ thể

```
public abstract class BaseTable{
    protected Query query=new Query(q:"");
    protected Condition condition=new Condition(con:"");
    protected Operator operator=new Operator(op:"");
    protected String placeholder="{}";
    protected TableName tableName;
```

- **Interface InternalQuery:**

- Chỉ định nghĩa hành vi: trừu tượng hóa mọi hành vi cốt lõi mà mọi lớp table phải có. InternalQuery chỉ cho biết "một bảng có thể thực hiện getDetailByKey()" nhưng

không tiết lộ cách `getDetailByKey()` được cài đặt trong Card hay Deck.

- Ẩn phức tạp: Người dùng chỉ cần biết rằng họ có thể yêu cầu lấy chi tiết bằng khóa, mà không cần bận tâm đến việc câu lệnh SQL được tạo ra như thế nào.

```
public interface InternalQuery {  
    public Query getDetailByKey();  
    public Query deleteByKey();  
    public Query updateDetailByKey();  
    public Query insertDetail();  
}
```

- **Lớp Query:** Khi sử dụng `query.select(...).from(...).where(...)`, không cần biết Query đã nối chuỗi SQL như thế nào bên trong mà nó sẽ tự tạo ra câu lệnh Query mong.

```
public BaseTable join(BaseTable table1, BaseTable table2, Condition con){  
  
    return new TempTable(table1.toString() + " JOIN " + table2.toString() + " ON " + con.toString());  
}
```

#### 4.5 Tương tác cơ sở Dữ liệu và Query Builder:

- ❖ Để thực hiện các truy vấn đến CSDL của hệ thống, đặc biệt là khi cần tự động hoá việc truy vấn hàng loạt, nhóm đứng trước việc lựa chọn 1 trong 2 phương pháp: **Query Builder** và **ORM**.

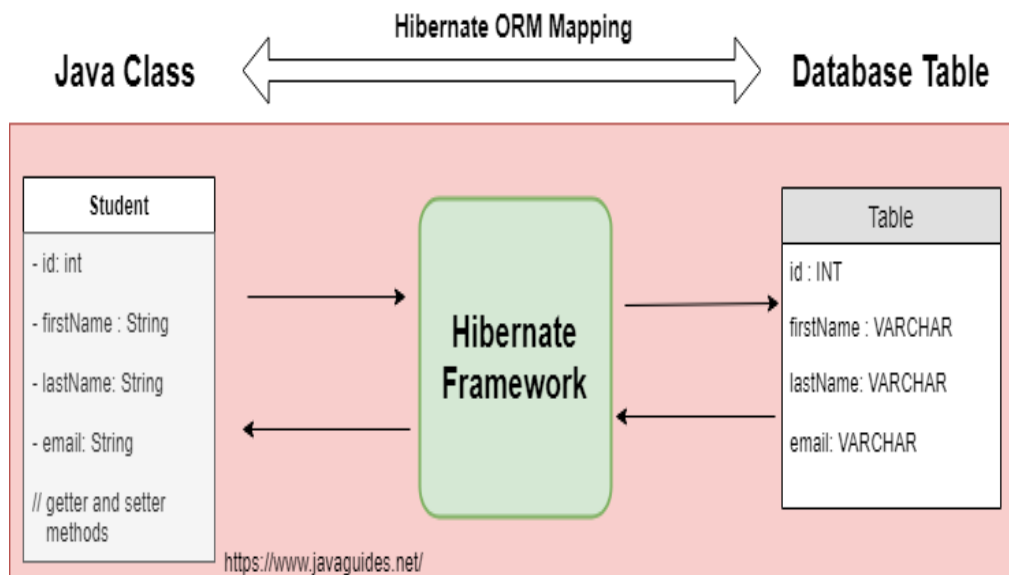
- ❖ **ORM:**

- **Mục tiêu chính của ORM:** Cung cấp một lớp trừu tượng (abstraction layer) giữa ứng dụng hướng đối tượng và cơ sở dữ liệu quan hệ. Nó cho phép thao tác với dữ liệu trong cơ sở dữ liệu thông qua các đối tượng Java (model objects) thay vì phải viết các câu lệnh SQL trực tiếp.
- **Các tính năng cốt lõi của ORM:**
  - **Ánh xạ đối tượng-quan hệ:** Tự động ánh xạ các bảng cơ sở dữ liệu thành các lớp Java và các hàng (rows) thành các đối tượng (objects).
  - **Thao tác dữ liệu qua đối tượng:** Cho phép thực hiện CRUD (Create, Read, Update, Delete) bằng cách gọi các phương thức trên đối tượng Java, không phụ thuộc vào mô hình dữ liệu cụ thể (schema).
  - **Quản lý phiên (Session Management):** Xử lý kết nối, giao dịch, và trạng thái của các đối tượng.



- **Lazy Loading/Eager Loading:** Quản lý cách tải dữ liệu liên quan.
- **Caching:** Tối ưu hóa hiệu suất truy vấn.

### Các frame work sử dụng ORM:



### ❖ Query Builder:

- Trừu tượng hóa thao tác DB:
  - Không cần viết chuỗi SQL trực tiếp, ta sử dụng các phương thức của lớp Query(select, from, where, update, delete, insert) để xây dựng truy vấn.
  - Các class table chứa các phương thức CRUD (getDetailByKey(), deleteKey(),...) được định nghĩa trong interface InternalQuery. Những phương thức này tự động tạo ra chuỗi SQL cần thiết cho các thao tác trên table đó.
  - Khi ta gọi các phương thức như myCard, getDetailByKey(), thì nó trả về một Query chứa SQL, thay vì phải tự viết truy vấn cứng như "Select front, back FROM Card WHERE card\_id=..."
- Đại diện table dưới dạng class:



- Mỗi table trong DB được đại diện dưới class Java riêng.
- Các cột trong bảng được định nghĩa dưới dạng các thuộc tính của Java.
- Ta sẽ có đối tượng Java để tham chiếu đến một table và các column của nó
- **Quản lý các Key:**
  - Class Key được sử dụng để đại diện cho các khóa chính, khóa ngoài, bao gồm cả tên cột và tên table\_parent. Điều này có ích khi xây dựng condition JOIN(...).
  - Điều này giúp quản lý các mối quan hệ giữa các bảng ở mức độ đối tượng, một phần giống chức năng của ORM.

### Các frame work sử dụng Query Builder:



### ❖ So sánh Query Builder và ORM:

- Điểm chung:
  - Điều tra về các yêu cầu CRUD biểu diễn dưới dạng truy vấn SQL.
  - Điều cho phép tự động hoá truy vấn.
- Điểm riêng của Query Builder:
  - **Trừu tượng hoá trên cú pháp SQL**
  - Thích hợp với các schema nhiều thực thể
  - Tối ưu hiệu suất
  - Cần kỹ năng SQL và hiểu về CSDL.
- Điểm riêng của ORM:
  - **Trừu tượng hoá trên các bảng, bản ghi và thao tác CRUD**
  - Thường dành cho các schema có 1 hoặc ít thực thể
  - Rất phù hợp với OOP nhưng rất khó triển khai
  - Phù hợp với người không có (hoặc rất ít) kinh nghiệm làm việc với CSDL.

#### ❖ Giải pháp lựa chọn

- Nhóm phát triển lựa chọn xây dựng một Custom SQL Query Builder, vì tính đơn giản, dễ tiếp cận, cho phép linh hoạt trong thiết kế, quá trình triển khai phù hợp với người học OOP bằng Java và CSDL.
- So với ORM: Kiến trúc ORM rất phức tạp và khó có thể thiết kế tốt, nhất là với những người chưa có đủ kinh nghiệm.

#### ❖ Mục tiêu dự án:

- Dự án tập trung vào xây dựng chuỗi câu lệnh SQL (Query Builder) một cách có cấu trúc và dễ đọc.
- Các lớp bảng: đại diện cho các bảng DataBase. Chúng chủ yếu chứa các tên cột( ví dụ: front, back,...) và các đối tượng để dễ dàng tham chiếu khi xây dựng truy vấn
- Trong interface InternalQuery: Giao diện định nghĩa các hoạt động của CRUD (Create, Read, Update, Delete), và các lớp bảng triển khai. Các phương thức này trả về đối tượng Query( chuỗi SQL), chứ không tự động thực hiện thao tác dữ liệu hay trả về đối tượng được điền từ DB.
- **Ưu điểm:**
  - Kiểm soát với SQL tốt hơn.
  - Độc lập ORM cụ thể.
  - Giảm SQL injection

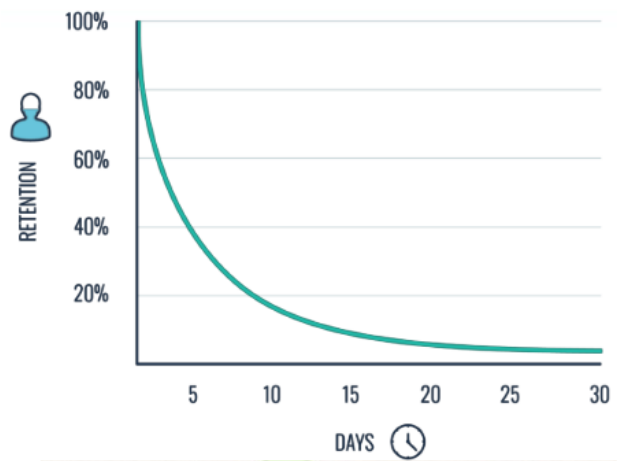
#### ❖ Kết quả:

- Hệ thống đã thành công trong việc trừu tượng hóa quá trình tạo SQL, tăng cường tính an toàn và khả năng bảo trì mã nguồn.

## 4.6 Phương pháp tính thời gian (số ngày) đến lần ôn tập tiếp theo - quản lý tiến độ học tập

### 4.6.1. Sơ lược về phương pháp học SRS

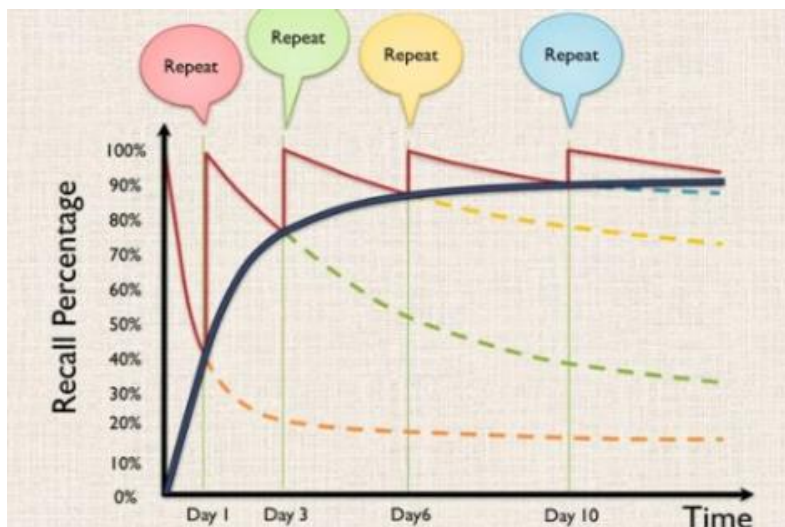
Một trong những vấn đề mà nhiều người học các môn học, từ vựng, ngữ pháp của một ngoại ngữ gặp phải, là: Sau khi học một kiến thức mới, khả năng ghi nhớ của chúng ta sẽ giảm dần theo thời gian nếu không được ôn tập. Đường cong lãng quên minh họa tốc độ mất mát thông tin này.



Phương pháp học tập Spaced Repetition System (SRS) được đưa ra, theo đó người học ôn tập tài liệu, thường dưới dạng các flashcard, theo các khoảng thời gian có hệ thống.

- + Ban đầu, khoảng thời gian giữa các lần ôn tập tương đối ngắn (khoảng 1 ngày đến 1 tuần)

- + Về sau khi người học đã ghi nhớ kiến thức, khoảng thời gian trên kéo dài ra (có thể là 1 tháng, 6 tháng, 1-2 năm).



- + Đồng lại, khi người học quên kiến thức, cần gấp rút ôn lại, khoảng thời gian tới lần ôn tiếp theo sẽ rút ngắn trở lại.

Phương pháp SRS, dựa trên cơ sở lý thuyết về đường cong quên lãng, là phương pháp học tập hiệu quả giúp người học nhớ lâu kiến thức. Đây là một phương pháp được chính Anki, ứng dụng học tập với flashcard và quản lý flashcard hàng đầu thế giới, quan tâm và triển khai.

Trong phần này, nhóm phát triển đã nghiên cứu cách xác định thời gian ôn tập tiếp theo cho mỗi thẻ ở một số nghiên cứu, bao gồm cả ứng dụng của Anki, và xây dựng 1 công thức riêng cho website của nhóm.

#### 4.6.2. Quy trình học trong ứng dụng

Nhóm xác định quy trình học của người học trên 1 bộ thẻ như sau:

- Người học lựa chọn 1 bộ thẻ (deck)
- Hệ thống lựa chọn 1 số thẻ trong deck cho người dùng
- Người học tiến hành học các thẻ đã được chọn
- Với mỗi từ, người học đánh giá mức độ ghi nhớ về từ vựng đó trên 4 mức: AGAIN, HARD, GOOD và EASY, và lưu tiến trình học
- Căn cứ vào kết quả học, hệ thống cập nhật ngày ôn tập tiếp theo

Một số câu hỏi đặt ra từ quy trình này:

- Cách lựa chọn các thẻ sẽ được học trong ngày hôm nay?
- Cách cập nhật ngày ôn tập tiếp theo cho các thẻ?

#### 4.6.3. Mức độ ghi nhớ

Người học đánh giá mức độ ghi nhớ cho từng thẻ (trong số các thẻ mà hệ thống cấp cho người học). Có 4 lựa chọn tương ứng 4 giá trị điểm (trên thang 4, tương tự GPA ở bậc Đại học)

Mức độ	Giá trị điểm
AGAIN	0
HARD	2
GOOD	3
EASY	4

Mức độ ghi nhớ dưới dạng điểm cho 1 lần học của 1 bộ thẻ được xác định bằng **TRUNG BÌNH** giá trị điểm của tất cả các thẻ mà người học đã học (và để đơn giản, hiện tại nhóm coi các thẻ có trọng số bằng nhau).

#### 4.6.4. Tốc độ học và Điểm tự tin (Learning pace and Confidence points)

##### a. Khái niệm

**Điểm tự tin** đánh giá mức độ tự tin của người dùng trên 1 thẻ trong suốt quá trình học vừa qua. Điểm tự tin giao động từ 0 đến 100 và là số nguyên.

**Tốc độ học** xác định mức độ nhanh hay chậm của toàn quá trình học. Cụ thể, tốc độ học sẽ là số điểm tự tin tăng lên sau 1 lần học, nếu điểm ghi nhớ trong lần học đó đủ cao.

##### b. Triển khai

Điểm tự tin C được cập nhật sau mỗi lần học cho các thẻ đã học tùy vào:

- Mức độ ghi nhớ dưới dạng điểm trong 1 lần học (M)
- Tốc độ học của bộ thẻ (L)

Điểm ghi nhớ (M)	Điểm tự tin được cập nhật (C')
------------------	--------------------------------

$3.6 \leq M \leq 4$	$C' = C + L$
$3.2 \leq M < 3.6$	$C' = C + \left\lfloor \frac{L}{2} \right\rfloor$
$2.5 \leq M < 3.2$	$C' = C$
$2.0 \leq M < 2.5$	$C' = \left\lfloor \frac{3C}{4} \right\rfloor$
$1.5 \leq M < 2.0$	$C' = \left\lfloor \frac{C}{2} \right\rfloor$
$1.0 \leq M < 1.5$	$C' = \left\lfloor \frac{C}{4} \right\rfloor$
$0.0 \leq M < 1.0$	$C' = 0$

Trong đó  $[x]$  là số nguyên lớn nhất không vượt quá  $x$ .

Sau mỗi lần học và cập nhật điểm tự tin  $C$ , giá trị điểm mới cũng xác định **Số ngày trước khi người dùng cần phải ôn tập lại  $D$** . Giá trị  $D$  sẽ được cập nhật trên các thẻ đã học sau khi kết thúc 1 lần học.  $D$  được tính như sau:

$$D = \frac{(e - 1)^{\frac{C}{5\pi}}}{(e - 1)^{20\pi}} \cdot 730$$

Để giảm khối lượng tính toán lặp lại, nhóm phát triển sẽ tính toán các giá trị của  $D$  với mỗi  $C$  từ 0 đến 100 và lưu vào 1 mảng Python để tham chiếu.

### c. Cơ sở triển khai

#### i. Phương pháp SRS

Về nguyên tắc của phương pháp SRS,  $D$  sẽ tăng lên khi người dùng đã tự tin, nhớ rõ từ vựng và sẽ giảm nếu thông qua 1 lần học, nhận thấy người dùng đã không còn tự tin về từ vựng đó như trước nữa.

Như vậy, ta nên xây dựng  $D$  là một đại lượng tăng theo  $C$ , và  $C$  sẽ được cập nhật tùy vào kết quả học tập ở 1 lần học.

#### ii. Một số nghiên cứu triển khai phương pháp SRS

[Nghiên cứu của Paul Pimsleur](#) là một trong các nghiên cứu về đường cong quên lãng và việc nhắc lại để khắc sâu kiến thức hơn.

Thay vì đưa ra các khoảng thời gian cụ thể, ông đề xuất phương án là đặt ra các khoảng thời gian tăng xấp xỉ theo cấp số nhân, mặc dù hệ số nhân cụ thể phụ thuộc vào tốc độ học và độ khó của kiến thức.

Ví dụ do Pimsleur đưa ra trong nghiên cứu là: Với hệ số nhân bằng 5, các khoảng thời gian giữa các lần ôn tập (một cách xấp xỉ): 5 giây  $\rightarrow$  25 giây  $\rightarrow$  2 phút  $\rightarrow$  10 phút  $\rightarrow$  1 giờ  $\rightarrow$  5 giờ  $\rightarrow$  1 ngày  $\rightarrow$  5 ngày  $\rightarrow$  25 ngày  $\rightarrow$  4 tháng  $\rightarrow$  2 năm.

Từ nghiên cứu trên, nhóm phát triển mong muốn:

- Bổ sung thêm tham số tốc độ học ( $L$ ) vào mỗi bộ thẻ.

- Khi C tăng thì D tăng theo hàm mũ của C.

Tuy nhiên:

- Không phải người học nào cũng có điều kiện học vào nhiều hơn 1 buổi trong 1 ngày, và để cho thuận tiện, D sẽ được tính bằng ngày thay vì chính xác hơn đến giờ hay phút.
- Khả năng ghi nhớ lâu của con người cũng có giới hạn. Con người không thể nhớ 1 khối lượng kiến thức nào đó quá giới hạn này nếu không được tiếp xúc với nó bằng bất kỳ giác quan nào. Do đó, cả 2 đại lượng C và D đều có giới hạn.

### iii. Một số giá trị đặc biệt

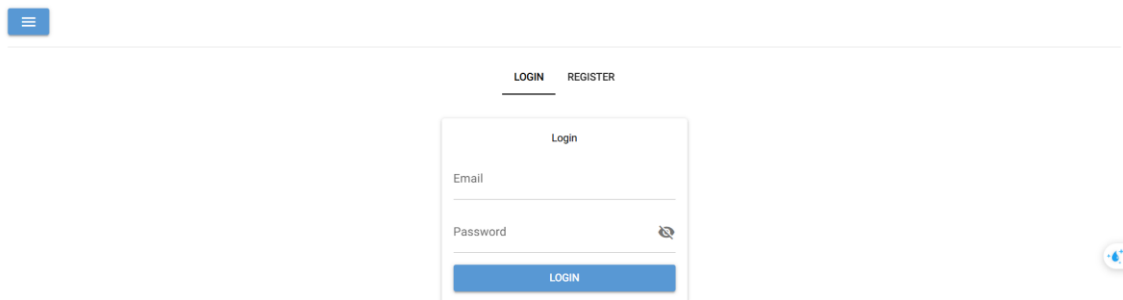
Với  $C = 0$ , có thể coi như người dùng đã quên sạch từ vựng đã cho, ta cần cho người học ôn ngay trong ngày hôm đó (hoặc cùng lắm ngày mai) do đó  $D \leq 1$ . Ở hệ thống này ta cho  $D = 0$ .

Với giá trị tối đa của C, tức là  $C = 100$ , ta cho  $D = 730$ , người dùng sẽ phải ôn tập sau muộn nhất 2 năm (khoảng thời gian tương tự hạn sử dụng của nhiều chứng chỉ ngoại ngữ và học thuật ở Việt Nam và trên thế giới như IELTS, TOEIC, TOEFL, TSA, HSA, v.v...).

## 4.7 Demo

- ❖ Dành cho người dùng có VIP (Premium): Dùng được AI-Generator

- **Login:**



The screenshot shows a web interface with a blue hamburger menu icon in the top left. Below it, there are two tabs: 'LOGIN' (selected) and 'REGISTER'. The main content area contains a 'Login' form with the following elements:

- A title 'Login' centered at the top of the form.
- An 'Email' input field.
- A 'Password' input field with a toggle icon (an eye) to the right.
- A blue 'LOGIN' button at the bottom of the form.

- **Register:**

≡

LOGINREGISTER

Register

Your name

Email

Password

Confirm password

Account type:Account type:public

REGISTER

- **Create Deck:** tạo một bộ thẻ của bản thân.
  - Public: Công khai cho các user khác có thể sử dụng
  - Private: Chỉ user sở hữu mới sử dụng được.

≡MY DECKSMY CARDSASK AIMY LEARNING STATUS

Hello VH Vương

Light Mode

Dark Mode

Account type: premium

Logout →

This is my deck tab

CREATE NEW DECK

Title: N5 Vocab for everyoneBack language: English

Learning pace: 5Card per day: 5

Tags: JLPT N5, Vocabulary

DECK SETTINGUSE THIS DECK

Title: Ngữ pháp N3 cho mọi ngườiBack language: English

Learning pace: 5Card per day: 5

Tags: Grammar, JLPT N3

DECK SETTINGUSE THIS DECK

Title: N4 vocabBack language: VietnameseView type: publicAuthor: nguyen duc thinh

Learning pace: 5Card per day: 5

Tags: JLPT N4, Vocabulary

DECK SETTINGUSE THIS DECK

Create new deck

TitleBasic English

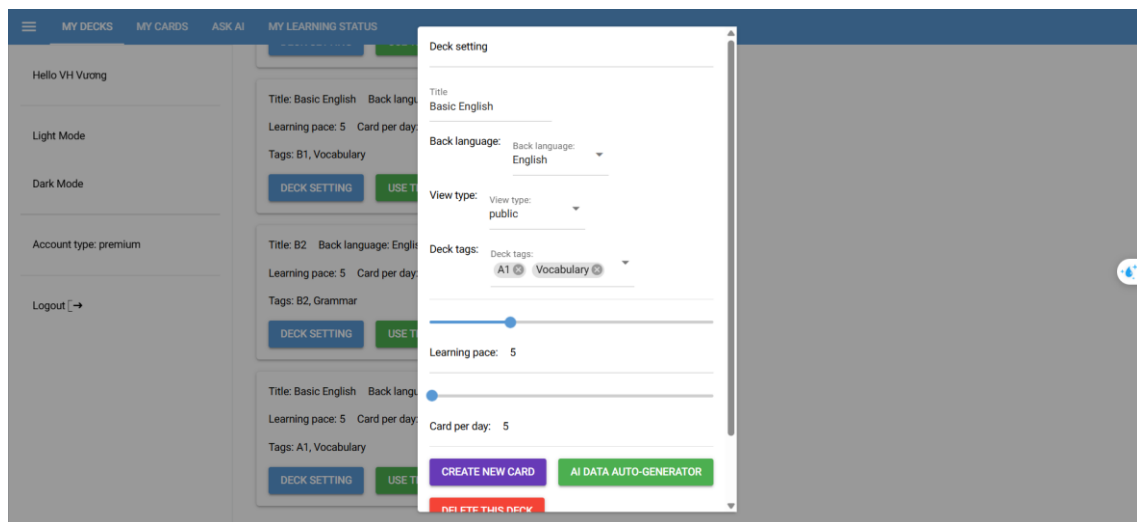
Back language:Back language:English

View type:View type:public

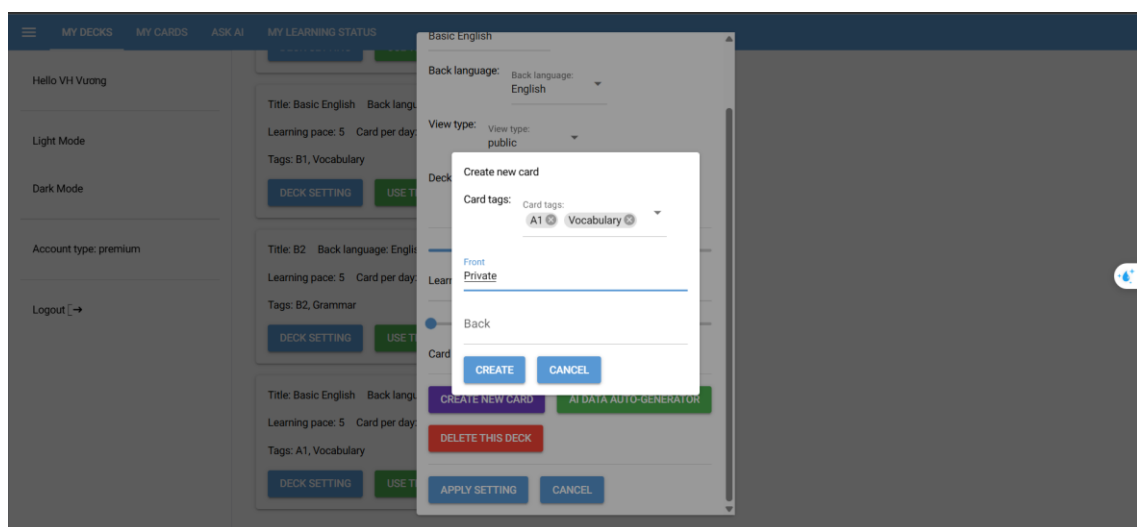
Deck tags:Deck tags:VocabularyA1

CREATECANCEL

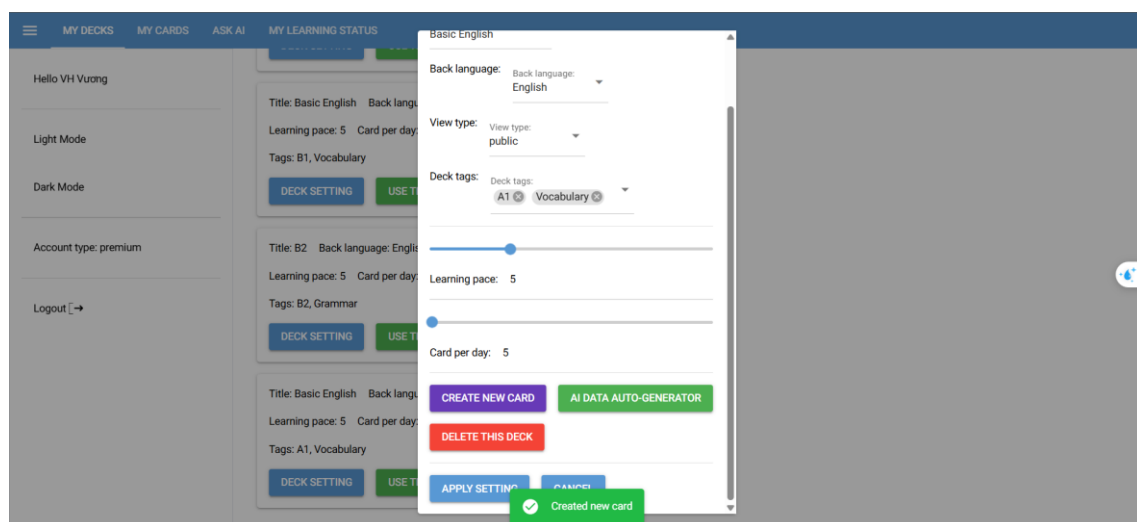
- **Deck\_setting:**



- Create Card: Tạo Card cho Deck, chỉ cần nhập vào từ khóa cần học.

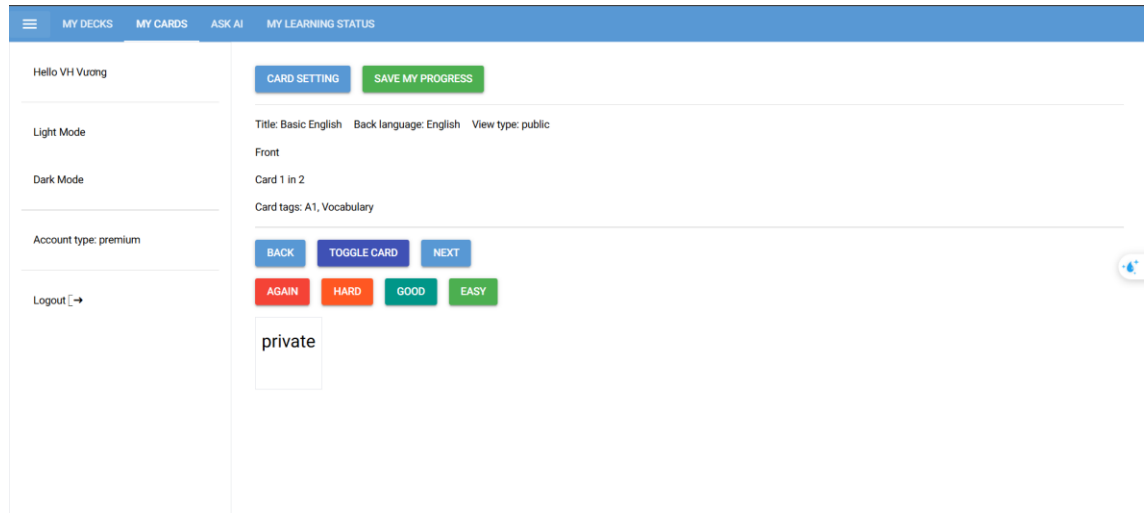


- AI Data Auto-Generator: Tìm kiếm Card\_front có Card\_back trống để sinh data cho Card\_back.

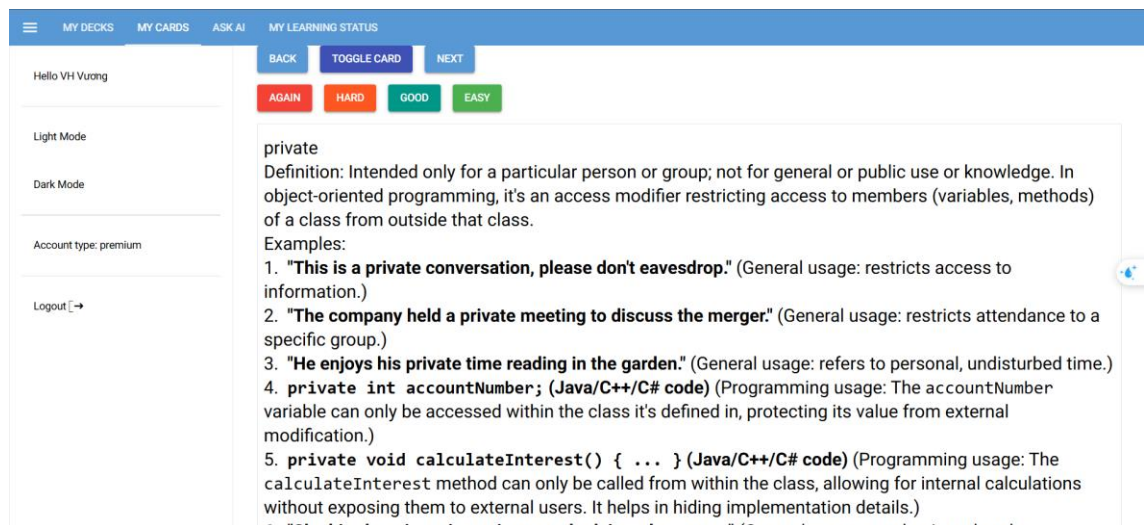




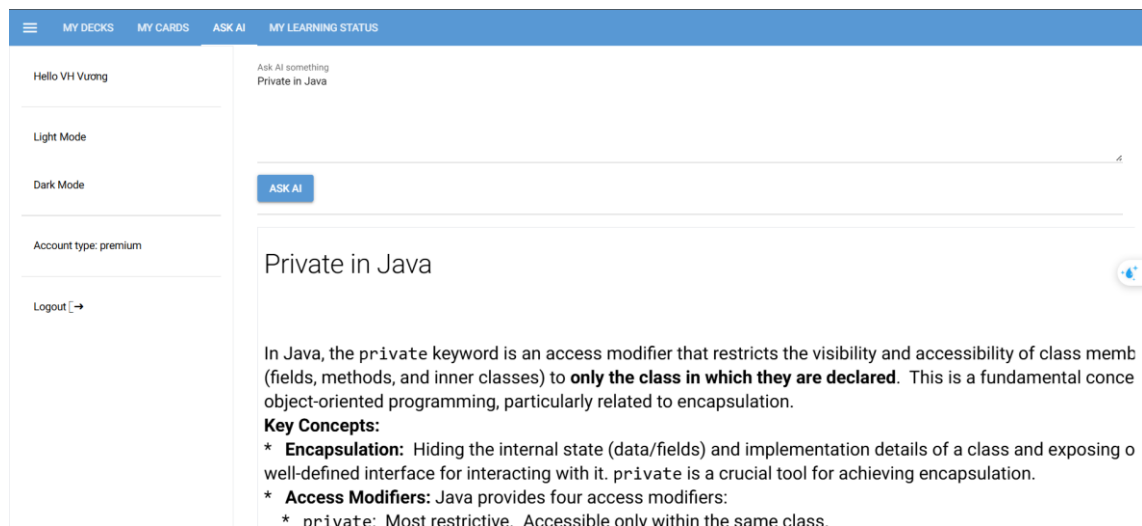
- **Card\_front:** Card\_front đã được thêm vào Deck



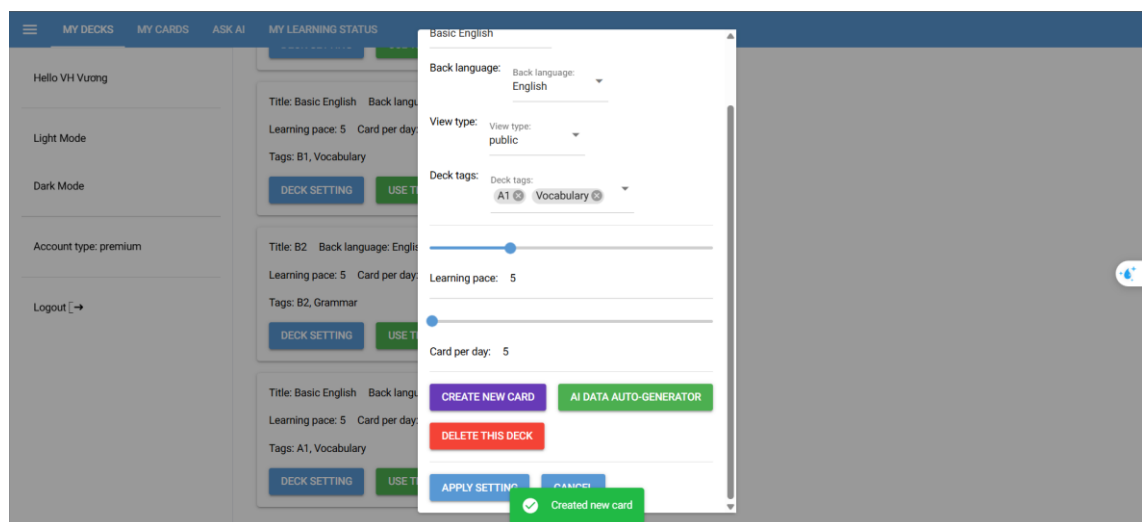
- **Card\_back:** data tự động sinh ra sau khi click “AI DATA AUTO-GENERATOR



- **AI\_ask:**

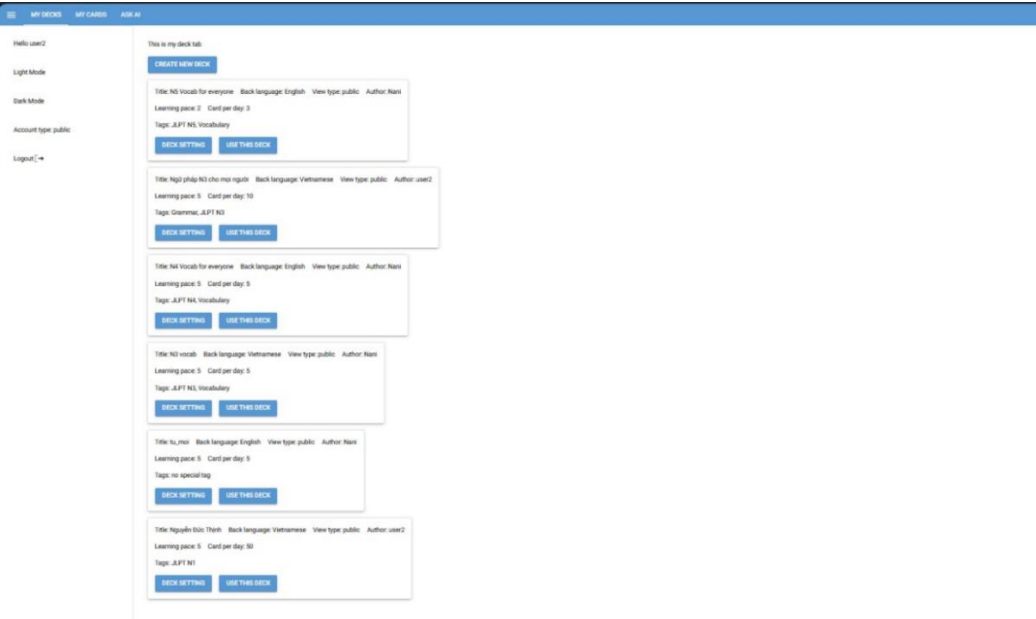


- **Learning Pace:** Đưa ra những Card cần phải học trước, hiển thị lên đầu Deck. Dữ liệu này dựa vào confident (AGAIN, HARD, GOOD, EASY) có trong mỗi Card. Learning Pace càng nhỏ thì Card ấy sẽ lặp lại nhiều lần, và ngược lại.
- **Card Per Day:** Đưa ra số Card mà User muốn học trong Deck.

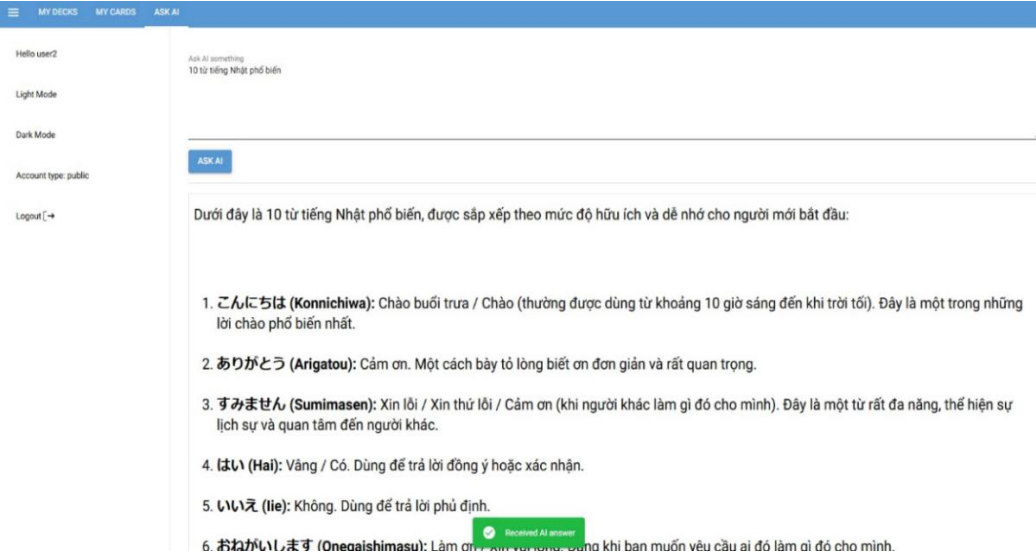


❖ Dành cho người dùng thường: Không thể dùng AI-Generator để sinh data cho Card\_back

- **Home\_Page:**



• AI\_ask:



## 5 Kết luận

### 5.1 Những điểm đạt được

Hệ thống đã hoàn thiện các chức năng cốt lõi theo đúng yêu cầu đề ra:

- Người dùng có thể dễ dàng đăng ký, đăng nhập, tạo bộ từ vựng (deck), thêm/sửa/xóa thẻ card và luyện tập ghi nhớ từ vựng thông qua chế độ flashcard.
- Việc áp dụng thuật toán lặp lại ngắt quãng để tính toán và đề xuất những thẻ cần ôn tập giúp tối ưu hóa hiệu quả ghi nhớ và theo dõi tiến độ học tập cá nhân.
- Trang web còn được tích hợp với AI có vai trò như một công cụ tra cứu thông minh. Người dùng có thể nhập bất kỳ từ hay cụm từ nào để nhận được giải nghĩa chính xác, ví dụ minh họa, cách dùng trong ngữ cảnh, cũng như phân biệt các từ dễ gây nhầm lẫn. Nhờ đó, việc học từ trở nên trực quan, hiệu quả và sâu sắc hơn.
- Về mặt giao diện, hệ thống được xây dựng với tiêu chí đơn giản, trực quan và thân thiện với người dùng, giúp người học dễ dàng thao tác dù không am hiểu công nghệ. Việc chọn tag từ danh sách có sẵn cũng giúp hạn chế lỗi chính tả và tăng khả năng phân loại, tìm kiếm.
- Đảm bảo được các yêu cầu nghiệp vụ như: mỗi deck phải có ít nhất một tag, tự động gán tag private nếu người dùng không chọn public, kiểm soát quyền truy cập dựa trên trạng thái deck (public/private), và hỗ trợ cấu hình học tập cá nhân trên từng deck.

### 5.2 Những điểm hạn chế

Mặc dù hệ thống đã đáp ứng được các chức năng cốt lõi, vẫn còn tồn tại một số điểm hạn chế cần được khắc phục trong các giai đoạn phát triển tiếp theo.

- **Hình thức luyện tập còn khá đơn giản:** Chưa được mở rộng sang các dạng khác như kiểm tra trắc nghiệm, điền từ vào chỗ trống, nối từ hoặc các bài kiểm tra định kỳ
- **Khả năng tùy biến nội dung của hệ thống còn hạn chế:** Chưa thể tạo thẻ từ vựng có kèm hình ảnh hoặc âm thanh – một yếu tố quan trọng đối với những người học ngoại ngữ, đặc biệt là trẻ em hoặc những người có xu hướng học qua thính giác
- **Chưa hỗ trợ chế độ học ngoại tuyến:** Gây bất tiện cho người dùng trong những tình huống mạng yếu hoặc không ổn định. Việc bổ sung tính năng học offline trong tương lai sẽ góp phần nâng cao trải nghiệm học tập, đảm bảo tính linh hoạt và tiện dụng hơn cho người học

### 5.3 Định hướng phát triển

Trong thời gian tới, hệ thống học từ vựng sẽ tiếp tục được mở rộng nhằm nâng cao trải nghiệm người dùng và hiệu quả học tập.

- **Đa dạng hóa hình thức luyện tập:** Ngoài chế độ flashcard hiện tại, hệ thống sẽ được bổ sung các phương thức học mới như trắc nghiệm, điền từ vào chỗ trống, kéo thả ghép từ và các bài kiểm tra định kỳ giúp người học củng cố kiến thức toàn diện và hứng thú hơn trong quá trình học.
- **Hỗ trợ tích hợp đa phương tiện:** Hệ thống sẽ cho phép người dùng thêm hình ảnh, âm thanh hoặc video vào thẻ từ. Việc này sẽ đặc biệt hữu ích cho việc học ngôn ngữ, giúp tăng khả năng ghi nhớ bằng trực quan và thính giác.
- **Xây dựng chế độ học offline:** Cho phép người dùng học tập ngay cả khi không có kết nối internet. Dữ liệu sẽ được đồng bộ khi có mạng trở lại, đảm bảo liên mạch trong tiến trình học. Và tính năng cá nhân hóa sẽ được nâng cao với khả năng gợi ý từ vựng theo cấp độ.

Nhóm phát triển rất mong nhận được sự giúp đỡ từ thầy, các nhóm còn lại, và tất cả mọi người sử dụng, nghiên cứu, theo dõi tiến độ của sản phẩm, để có thể phát triển hệ thống tiến tới một sản phẩm hoàn chỉnh.

Nhóm tin rằng, hệ thống hỗ trợ học tập dựa trên flashcard sẽ là trợ thủ đắc lực dành cho học sinh, sinh viên, giúp học sinh, sinh viên vượt qua những trở ngại trong việc nhớ và hiểu từ vựng, ngữ pháp khi học ngôn ngữ, khái niệm trong các môn học.

**XIN CHÂN THÀNH CẢM ƠN!**

## Tài liệu tham khảo

1. [Tìm hiểu Eloquent Model và Query Builder](#)
2. [Raw SQL vs Query Builder vs ORM](#)
3. [Diễn giải chi tiết về OOP \(Java\)](#)
4. [Object Relational Mapping](#)
5. [Spring Boot JPA Native Query with Example - GeeksforGeeks](#)
6. [Viết SQL trong Java với JOOQ](#)
7. [Anki Algorithm Explained in 5 minutes! - YouTube](#)
8. [Tìm hiểu tính đa hình trong Java](#)
9. [NiceGUI Documentation](#)
10. [sql - Laravel Eloquent ORM vs Query builder which one better? - Stack Overflow](#)
11. [Pimsleur, Paul \(February 1967\). "A Memory Schedule". The Modern Language Journal.](#)