

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC CẦN THƠ
TRƯỜNG CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG



**BÁO CÁO ĐỒ ÁN
ĐỀ TÀI**
**XÂY DỰNG ỦNG DỤNG THẺ GHI NHỚ
FLEXIFLASH**

TÊN HỌC PHẦN: PHÁT TRIỂN ỦNG DỤNG DI ĐỘNG
MÃ HỌC PHẦN: CT484-01

Giảng viên giảng dạy:
TS Bùi Võ Quốc Bảo

Sinh viên thực hiện:
Trần Yên Nhi - B2111857
Lê Trương Ngọc Dyên - B2105569

Cần Thơ, 01/2024

MỤC LỤC

MỤC LỤC	2
LINK GITHUB MÃ NGUỒN:	5
LINK YOUTUBE CHẠY DEMO	5
I. TỔNG QUAN	5
1. Miêu tả ứng dụng.....	5
2. Phân công công việc.....	5
II. CHI TIẾT CHỨC NĂNG	5
1. Giao diện Đăng nhập.....	5
2. Giao diện Đăng ký.....	7
3. Giao diện Trang chủ	9
4. Giao diện Trang tài khoản người dùng	12
5. Giao diện Trang thêm bộ thẻ mới và những bộ thẻ được tạo gần đây.....	14
6. Giao diện thêm, sửa bộ thẻ.....	15
7. Giao diện thêm, sửa flashcard	19
8. Giao diện Trang một bộ thẻ	23
9. Giao diện Trang một flashcard.....	26
10. Giao diện Chi tiết một flashcard	28
11. Giao diện Các flashcard được đánh dấu.....	30
12. Giao diện Trả lời câu hỏi.....	31
13. Giao diện chỉnh sửa email	34
14. Giao diện chỉnh sửa password.....	37
15. Giao diện Các bộ thẻ được yêu thích	40
16. Giao diện danh sách chỉnh sửa các flashcard	42
TÀI LIỆU THAM KHẢO	47

DANH MỤC HÌNH

Hình 1 Giao diện đăng nhập.....	6
Hình 2 Giao diện đăng ký	8
Hình 3 Giao diện trang chủ	10
Hình 4 Lọc chủ đề bộ thẻ	10
Hình 5 Tìm kiếm bộ thẻ	11
Hình 6 Giao diện tài khoản	13
Hình 7 Chuyển ché độ tối.....	13
Hình 8 Giao diện thêm bộ thẻ và bộ thẻ được tạo gần đây	14
Hình 9 Giao diện thêm bộ thẻ mới.....	16
Hình 10 Giao diện khi nhập nội dung thêm bộ thẻ mới.....	16
Hình 11 Giao diện chỉnh sửa bộ thẻ.....	17
Hình 12 Chính sửa ảnh nền của thẻ	17
Hình 13 Sau khi chỉnh sửa thành công	18
Hình 14 Giao diện thêm flashcard	20
Hình 15 Cảnh báo nội dung không hợp lệ	20
Hình 16 Thêm một flashcard.....	21
Hình 17 Thêm một flahscard không có mô tả	21
Hình 18 Thông báo đã thêm tất cả flashcard	22
Hình 19 Giao diện chỉnh sửa một flahscard.....	22
Hình 20 Giao diện chi tiết một bộ thẻ	24
Hình 21 Yêu thích bộ thẻ	24
Hình 22 Thông báo xác nhận xóa bộ thẻ	25
Hình 23 Giao diện chỉnh sửa bộ thẻ.....	25
Hình 24 Giao diện hiển thị một flashcard	27
Hình 25 Đánh dấu một flahscard	27
Hình 26 Giao diện chi tiết một flahscard	29
Hình 27 Danh sách flashcard được đánh dấu của một bộ thẻ	30
Hình 28 Giao diện tất cả flashcard được đánh dấu	30
Hình 29 Giao diện trả lời câu hỏi	32
Hình 30 Trả lời câu hỏi sai.....	32
Hình 31 Trả lời câu hỏi đúng	33
Hình 32 Thông báo kết quả	33
Hình 33 Giao diện chỉnh sửa email.....	35
Hình 34 Cảnh báo email không hợp lệ.....	35
Hình 35 Thông báo chỉnh sửa email thành công.....	36
Hình 36 Giao diện đổi mật khẩu	38
Hình 37 Cảnh báo mật khẩu không khớp.....	38
Hình 38 Thông báo mật khẩu không đúng.....	39
Hình 39 Thông báo đổi mật khẩu thành công	39
Hình 40 Giao diện danh sách các bộ thẻ được yêu thích	41
Hình 41 Giao diện danh sách chỉnh sửa flashcard của 1 bộ thẻ.....	43
Hình 42 Chính sửa một flahscard.....	43

Hình 43 Sau khi chỉnh sửa	44
Hình 44 Thông báo xác nhận xóa một flashcard	45
Hình 45 Xóa flashcard thành công.....	45

LINK GITHUB MÃ NGUỒN:

<https://github.com/24-25Sem2-Courses/ct48401-project-DMonkey2943>

LINK YOUTUBE CHẠY DEMO

<https://www.youtube.com/watch?v=n7tsWDWMILg>

I. TỔNG QUAN

1. Miêu tả ứng dụng

Theo nhiều nghiên cứu, hình ảnh được nén bộ ghi nhớ nhanh hơn và giúp kích thích nhiều giác quan tăng cường khả năng ghi nhớ. Do đó ứng dụng FlexiFlash được thiết kế với sự kết hợp hình ảnh trong từng thẻ ghi nhớ nhằm tối ưu hóa quá trình học tập và ghi nhớ thông tin.

FlexiFlash cho phép người dùng tự do sáng tạo và thiết kế các bộ thẻ ghi nhớ cá nhân. Ứng dụng cung cấp tính năng cho phép người dùng mô tả chi tiết từng thẻ đọc nội dung thẻ, điều này giúp việc học trở nên thú vị và linh động. Với FlexiFlash, việc học tập trở nên dễ dàng và hiệu quả hơn bao giờ hết, giúp người dùng nắm bắt kiến thức và ghi nhớ lâu dài.

Ứng dụng lưu trữ và quản lý dữ liệu thông qua API được xây dựng bằng Laravel và phần mềm phát triển môi trường cục bộ Laragon

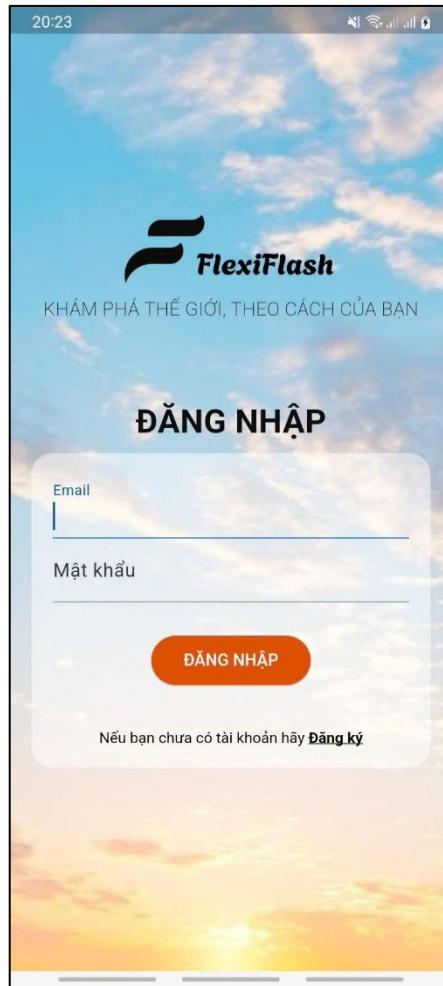
2. Phân công công việc

Thành viên	Công việc	Mức độ hoàn thành
Trần Yên Nhi	<ul style="list-style-type: none">- Xây dựng Front-end- Viết báo cáo	100%
Lê Trương Ngọc Duyên	<ul style="list-style-type: none">- Xây dựng Back-end- Viết báo cáo	100%

II. CHI TIẾT CHỨC NĂNG

1. Giao diện Đăng nhập

Giao diện đăng nhập gồm 2 ô nhập liệu Email và Mật khẩu để người dùng nhập thông tin đăng nhập



Hình 1 Giao diện đăng nhập

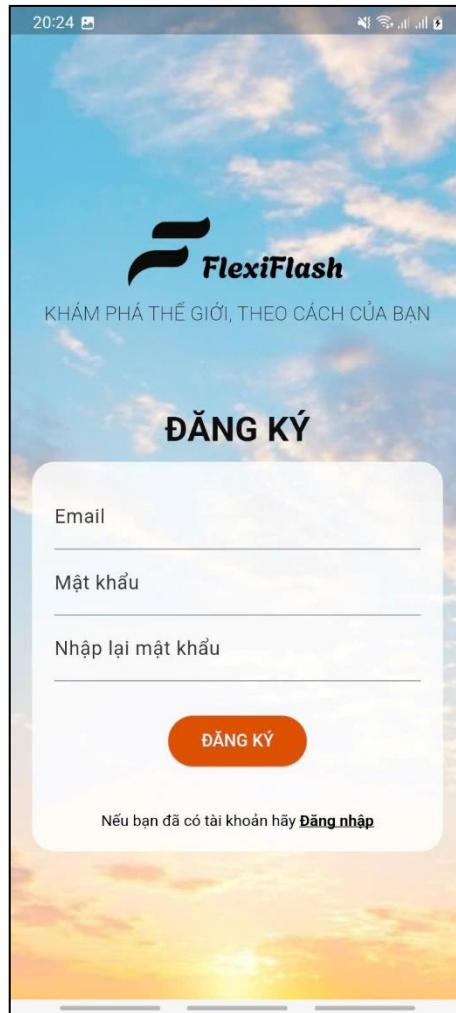
Chi tiết cài đặt:

- Các widget được sử dụng:
 - Scaffold: cung cấp cấu trúc cơ bản cho giao diện.
 - Image.asset(): là phương thức giúp hiển thị ảnh logo từ thư mục asset.
 - Stack: xếp chồng các widget trong giao diện.
 - SingleChildScrollView: cho phép cuộn khi bàn phím xuất hiện làm tràn nội dung.
 - SafeArea: tránh các khu vực nguy hiểm để giao diện hiển thị đúng.
 - Container: dùng để chứa và căn chỉnh các widget.
 - SizedBox: Tạo khoảng cách giữa các widget với kích thước cố định.
 - Column: giúp các widget con được xếp theo chiều dọc.
 - Text: Hiển thị các chuỗi văn bản, ví dụ như tiêu đề và slogan.
 - Form: quản lý và xác thực dữ liệu nhập từ người dùng.
 - TextFormField: nhận dữ liệu nhập từ người dùng bao gồm email và mật khẩu.

- RichText và TextSpan: giúp hiển thị văn bản với nhiều định dạng như gạch dưới và in đậm “Đăng nhập”.
- ElevatedButton: là một nút bấm có hiệu ứng nâng cao.
- Các widget đặc biệt:
 - TapGestereCognizer: trình nhận diện cử chỉ và xử lý sự kiện tap.
 - CircularProgressIndicator: hiển thị tiến trình dạng tròn khi thông tin đăng nhập đang được xử lý.
 - Opacity: điều chỉnh độ trong suốt của widger chứa ảnh nền
- Provider là plugin được sử dụng để quản lý trạng thái cùng ChangeNotifier.
- Giao diện sử dụng giải pháp quản lý chia sẻ ValueListenableBuilder để lắng nghe người dùng có submit hay chưa. AuthManger là một ChangeNotifier cũng được sử dụng để gọi hàm login.
- Khi submit thông tin đăng nhập sẽ được gửi bằng phương thức POST đến API '/login' để xử lý. Nếu thông tin đăng nhập đúng, email và id của người dùng được trả về, đồng thời nhận một token để truy cập ứng dụng. Nếu thông tin đăng nhập sau một thông báo sẽ hiển thị.

2. Giao diện Đăng ký

Giao diện đăng ký gồm 3 ô nhập liệu là Email, Mật khẩu và Nhập lại mật khẩu sau khi người dùng nhập thông tin và thực hiện đăng ký thành công sẽ chuyển sang trang đăng nhập.



Hình 2 Giao diện đăng ký

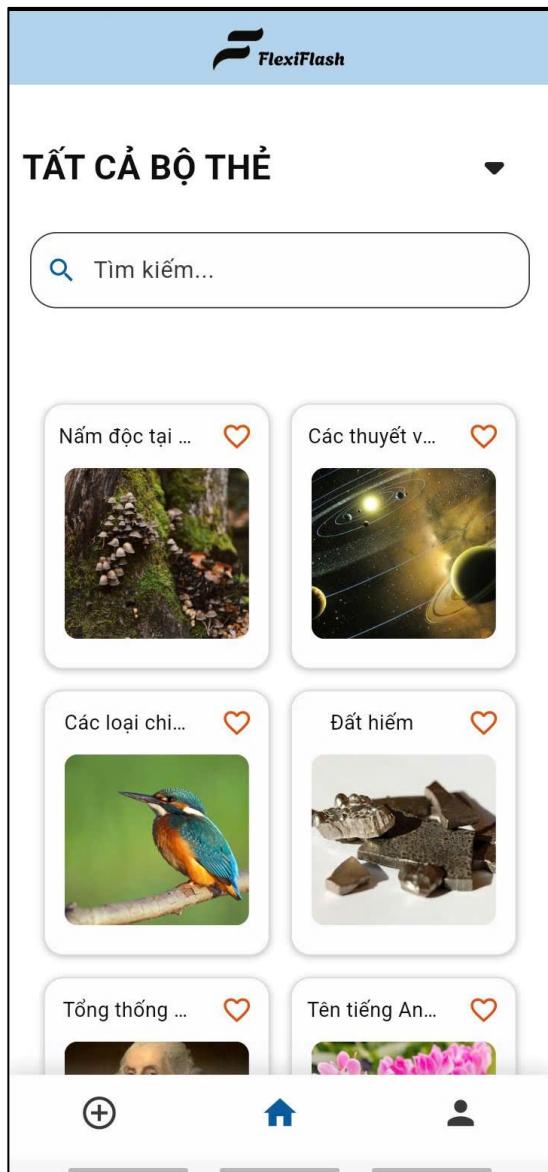
Chi tiết cài đặt:

- Bảng users trong cơ sở dữ liệu có các trường sau:
 - id: id của tài khoản.
 - email: email của tài khoản.
 - password: mật khẩu của tài khoản.
 - remember_token: token được ghi nhớ
 - created_at: thời gian tạo tài khoản.
 - updated_at: thời gian cập nhật tài khoản.
- Các widget được sử dụng:
 - Scaffold: cung cấp cấu trúc cơ bản cho giao diện.
 - Image.asset(): là phương thức giúp hiển thị ảnh logo từ thư mục asset.
 - Stack: xếp chồng các widget trong giao diện.
 - SingleChildScrollView: cho phép cuộn khi bàn phím xuất hiện làm tràn nội dung.
 - SafeArea: tránh các khu vực nguy hiểm để giao diện hiển thị đúng.
 - Container: dùng để chứa và căn chỉnh các widget.

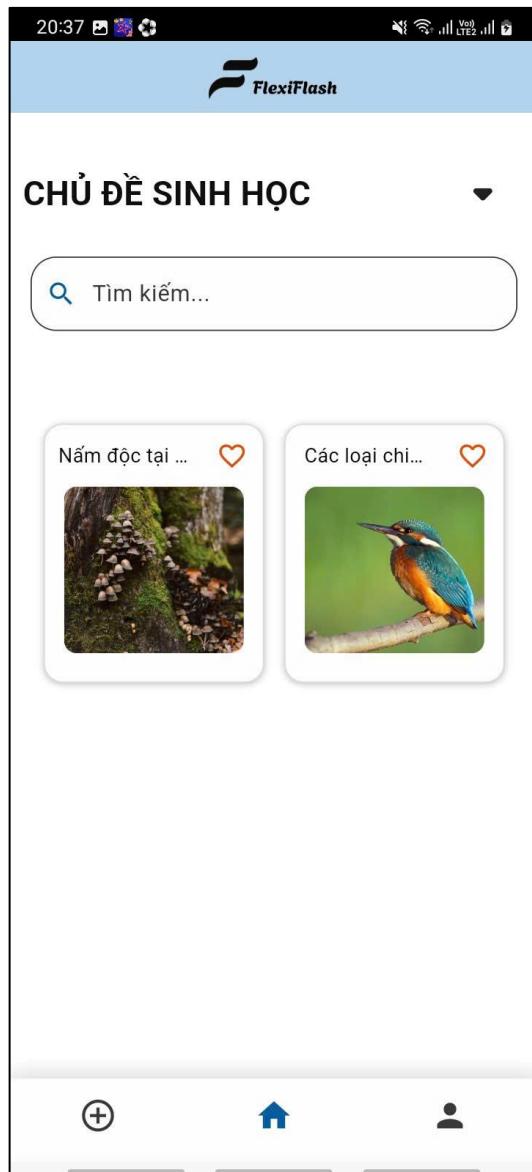
- SizedBox: Tạo khoảng cách giữa các widget với kích thước cố định.
 - Column: giúp các widget con được xếp theo chiều dọc.
 - Text: Hiển thị các chuỗi văn bản, ví dụ như tiêu đề và slogan.
 - Form: quản lý và xác thực dữ liệu nhập từ người dùng.
 - TextFormField: nhận dữ liệu nhập từ người dùng bao gồm email và mật khẩu.
 - RichText và TextSpan: giúp hiển thị văn bản với nhiều định dạng như gạch dưới và in đậm “Đăng nhập”.
 - ElevatedButton: là một nút bấm có hiệu ứng nâng cao.
- Các widget đặt biệt:
- TapGestereCognizer: trình nhận diện cử chỉ và xử lý sự kiện tap.
 - CircularProgressIndicator: hiển thị tiến trình dạng tròn khi thông tin đăng nhập đang được xử lý.
 - Opacity: điều chỉnh độ trong suốt của widger chứa ảnh nền
- Provider là plugin được sử dụng để quản lý trạng thái cùng ChangeNotifier.
- AuthManger là một ChangeNotifier được dùng để gọi hàm register.
- Khi submit thông tin đăng ký được gửi bằng phương thức POST đến API ‘/register’ để thêm user vào bảng users với các trường id, email, password.

3. Giao diện Trang chủ

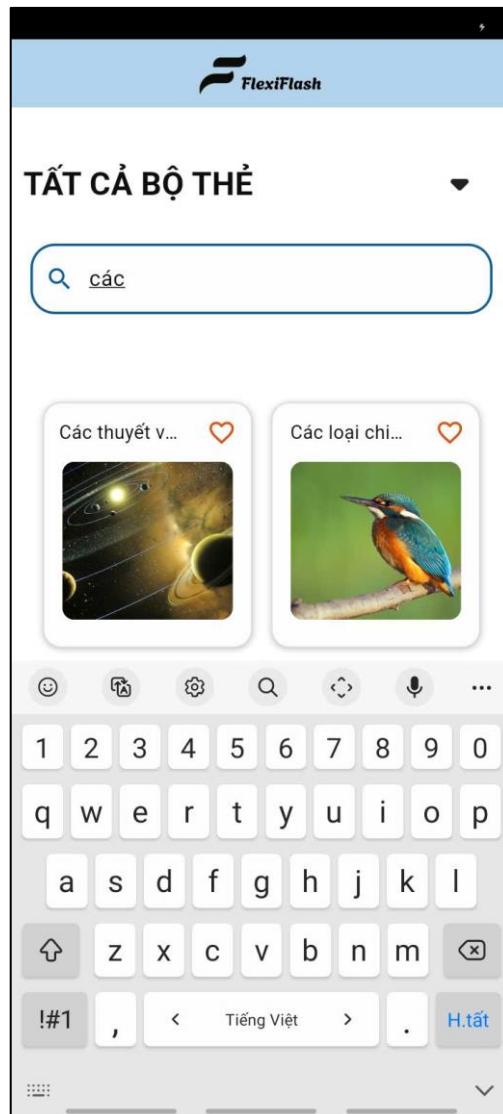
Giao diện trang chủ khi vừa đăng nhập vào sẽ hiển thị toàn bộ các bộ thẻ. Người dùng có thể thực hiện tìm kiếm bộ thẻ hoặc dùng menu để lọc ra thẻ loại các bộ thẻ.



Hình 3 Giao diện trang chủ



Hình 4 Lọc chủ đề bộ thẻ



Hình 5 Tìm kiếm bộ thẻ

Chi tiết cài đặt:

- Các widger được sử dụng:
 - Scaffold: cung cấp cấu trúc cơ bản cho giao diện.
 - AppBar: Hiển thị logo.
 - Image.asset(): là phương thức giúp hiển thị ảnh logo từ thư mục asset.
 - Stack: xếp chồng các widget trong giao diện.
 - Container: dùng để chứa và căn chỉnh các widget.
 - Center: Căn giữa các widget con bên trong.
 - SizedBox: Tạo khoảng cách giữa các widget với kích thước cố định.
 - Column: giúp các widget con được xếp theo chiều dọc.
 - Text: Hiển thị các chuỗi văn bản.
 - TextField: nhận dữ liệu nhập từ người dùng từ ô tìm kiếm.
 - Padding: Tạo khoảng cách giữa các phần tử con và biên của widget.

- Expanded: giúp mở rộng widger con chiếm lấy tối đa không gian có sẵn.
 - IconButton: widget kết hợp icon trái tim và nút bấm
- Các widget đặc biệt được sử dụng:
- CustomProgressIndicator là widget tự xây dựng giúp tùy chỉnh để hiển thị đang trong quá trình lấy các bộ thẻ.
 - PopupMenuButton hiển thị danh sách tùy chọn loại bộ thẻ dưới dạng pop-up.
 - FilterMenu: widget tự xây dựng giúp lọc hiển thị menu chọn loại.
 - TopicName: widget tự xây dựng giúp hiển thị tiêu đề theo loại được chọn.
 - FutureBuilder: hiển thị dữ liệu bất đồng bộ là các bộ thẻ được lấy.
 - GridDeck: widget tự xây dựng giúp chia các deck thành lưới phù hợp.
 - GridDeckItem: widget tự xây dựng giúp hiển thị một bộ thẻ trong.
 - BottomNavBar: widget tự xây dựng giúp hiển thị thanh điều hướng bên dưới.
- Các plugin sử dụng:
- provider là plugin được sử dụng để quản lý trạng thái cùng ChangeNotifier.
 - lottie: giúp hiển thị animation dạng JSON được gọi trong CircularProgressIndicator.
 - flex_color_scheme: tạo giao diện Material Theme dễ dàng.
- DeckManager là một ChangeNotifier được sử dụng để lấy danh sách các bộ thẻ có trong cơ sở dữ liệu thông qua phương thức GET đến API '/deck'. Khi thực hiện thích hoặc bỏ thích một bộ thẻ, hàm update sẽ được gọi thông qua phương thức POST gửi yêu cầu thay đổi trạng thái yêu thích đến API '/deck/\${deckId}'.

4. Giao diện Trang tài khoản người dùng

Giao diện tài khoản người dùng giúp người dùng quản lý tài khoản, bộ thẻ yêu thích, bộ thẻ của mình, tùy chỉnh chế độ sáng tối và đăng xuất khỏi tài khoản.



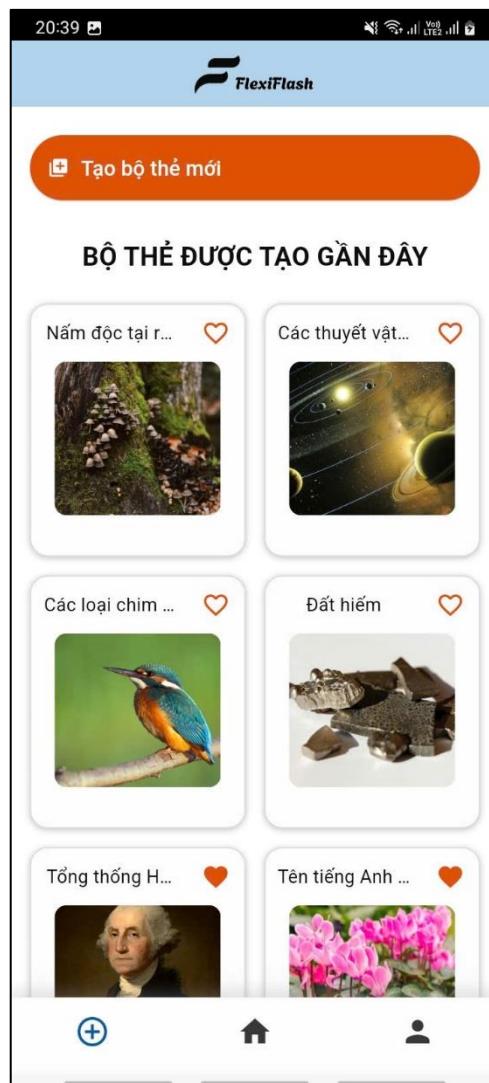
Chi tiết cài đặt:

- Các widget sử dụng:
 - Scaffold: cung cấp cấu trúc cơ bản cho giao diện.
 - Image.asset(): là phương thức giúp hiển thị ảnh logo từ thư mục asset.
 - AppBar: Hiển thị logo.
 - Column: giúp các widget con được xếp theo chiều dọc.
 - Padding: Tạo khoảng cách giữa các phần tử con và biên của widget.
 - SizedBox: Tạo khoảng cách giữa các widget với kích thước cố định.
 - Text: Hiển thị các chuỗi văn bản.
 - Row: hiển thị các widget được xếp theo chiều ngang.
- Các widget đặc biệt được sử dụng:
 - BotNavBar: widget tự xây dựng giúp hiển thị thanh điều hướng bên dưới.

- WarningButton: widget tự xây dựng giúp hiển thị button màu đỏ.
- LongButton: : widget tự xây dựng giúp hiển thị các button dài màu secondary của hệ thống.
- provider là plugin được sử dụng để quản lý trạng thái cùng ChangNotifier.
- AuthManager gọi hàm logout để đăng xuất khỏi tài khoản và xóa token bằng phương thức POST thông qua API '/logout'.

5. Giao diện Trang thêm bộ thẻ mới và những bộ thẻ được tạo gần đây.

Giao diện thêm này đặc biệt có một nút bấm trên cùng để người dùng thực hiện tạo bộ thẻ mới. Bên dưới nút bấm là các bộ thẻ được tạo cách đây 7 ngày trước.



Hình 8 Giao diện thêm bộ thẻ và bộ thẻ được tạo gần đây

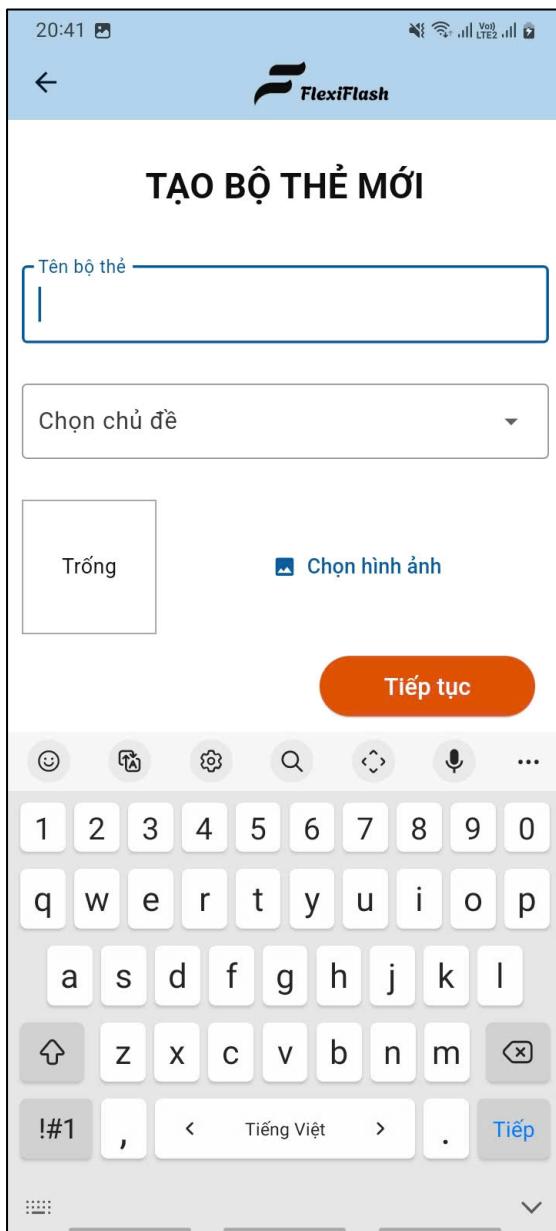
Chi tiết cài đặt:

- Các widget được sử dụng:

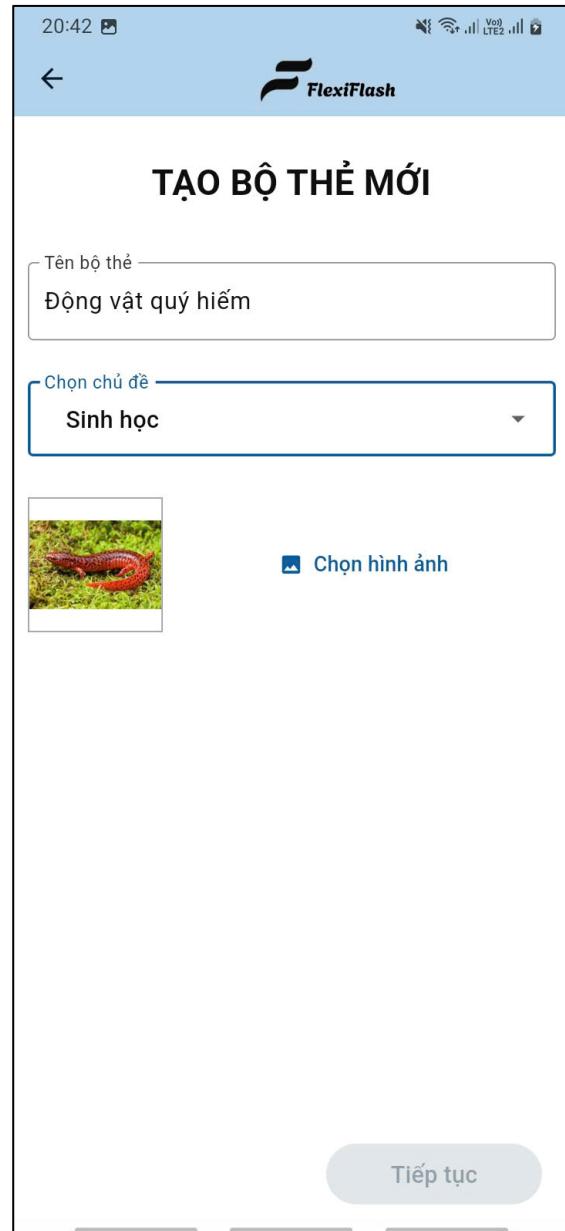
- Scaffold: cung cấp cấu trúc cơ bản cho giao diện.
- Image.asset(): là phương thức giúp hiển thị ảnh logo từ thư mục asset.
- AppBar: Hiển thị logo.
- Column: giúp các widget con được xếp theo chiều dọc.
- Padding: Tạo khoảng cách giữa các phần tử con và biên của widget.
- SizedBox: Tạo khoảng cách giữa các widget với kích thước cố định.
- Text: Hiển thị các chuỗi văn bản.
- Row: hiển thị các widget được xếp theo chiều ngang.
- Các widget đặc biệt được sử dụng:
 - BotNavBar: widget tự xây dựng giúp hiển thị thanh điều hướng bên dưới.
 - LongButton: : widget tự xây dựng giúp hiển thị các button dài màu secondary của hệ thống.
 - FutureBuilder: hiển thị dữ liệu bất đồng bộ là các bộ thẻ được lấy.
 - GridDeck: widget tự xây dựng giúp chia các deck thành lưới phù hợp.
 - GridDeckItem: widget tự xây dựng giúp hiển thị một bộ thẻ trong.
- provider là plugin được sử dụng để quản lý trạng thái cùng ChangeNotifier.
- DeckManager là một ChangeNotifier được sử dụng để lấy danh sách các bộ thẻ có trong cơ sở dữ liệu thông qua phương thức GET đến API '/deck'. Các bộ thẻ này được lọc qua hàm .where() với điều kiện thuộc tính createdAt của bộ thẻ nằm trong 7 ngày gần đây.

6. Giao diện thêm, sửa bộ thẻ

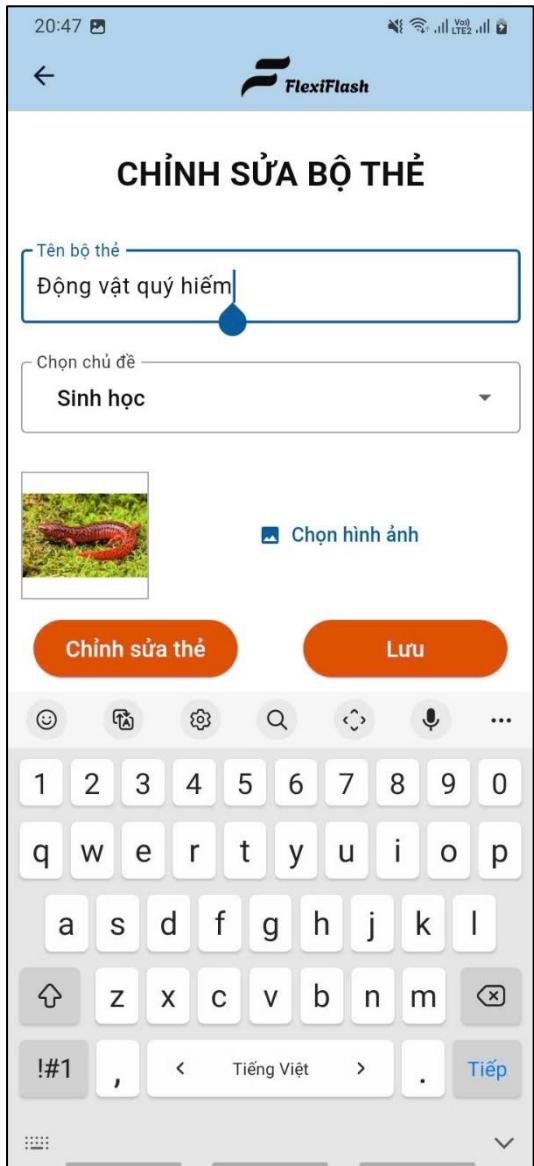
Tại giao diện này, người dùng nhập các thông tin của bộ thẻ để thêm hoặc sửa bộ thẻ với điều kiện là tất cả nội dung không được bỏ trống.



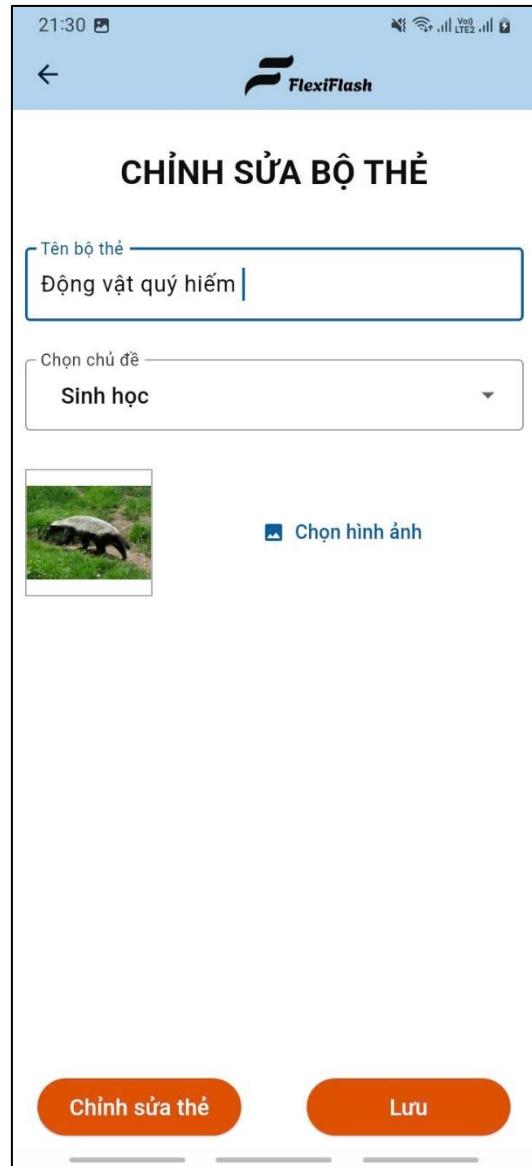
Hình 9 Giao diện thêm bộ thẻ mới



Hình 10 Giao diện khi nhập nội dung thêm bộ thẻ mới



Hình 11 Giao diện chỉnh sửa bộ thẻ



Hình 12 Chỉnh sửa ảnh nền của thẻ



Hình 13 Sau khi chỉnh sửa thành công

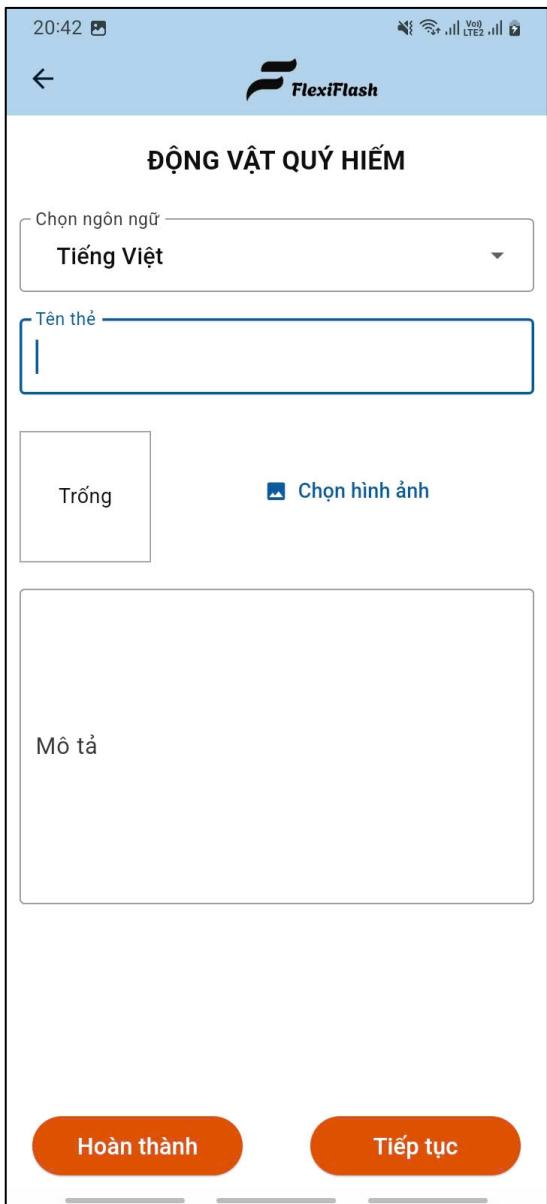
Chi tiết cài đặt:

- Bảng dekes trong cơ sở dữ liệu có các trường sau:
 - id: id của bộ thẻ.
 - title: tên bộ thẻ.
 - type: chủ đề của bộ thẻ.
 - imageBg: ảnh nền.
 - isFavorite: trạng thái yêu thích của bộ thẻ.
 - userId: id người tạo bộ thẻ.
 - created_at: thời gian tạo.
 - updated_at: thời gian cập nhật

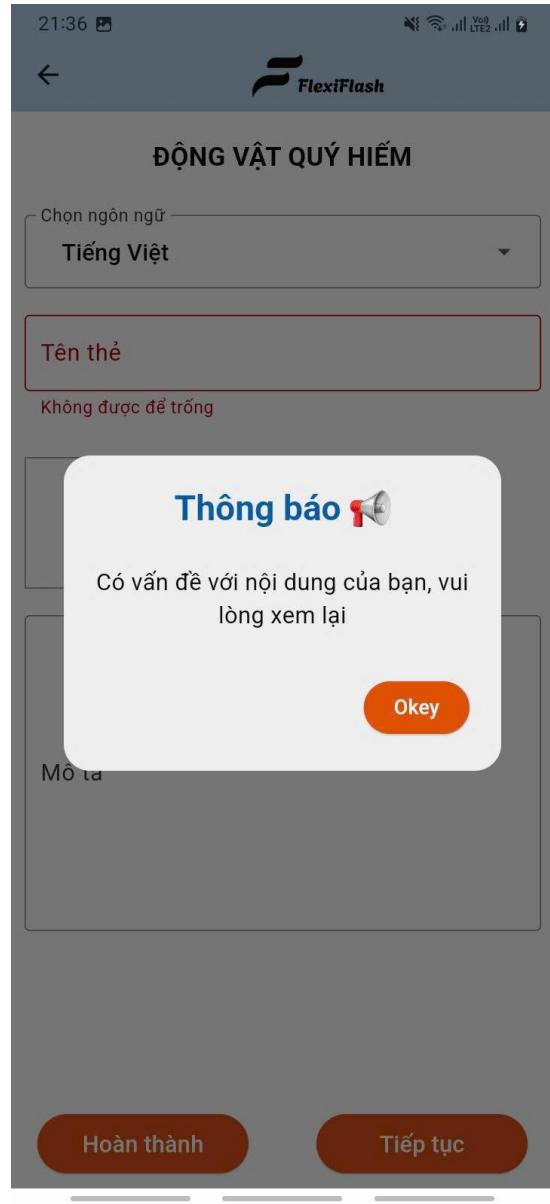
- Các widget trong giao diện:
 - Scaffold: cung cấp cấu trúc cơ bản cho giao diện.
 - Image.asset(): là phương thức giúp hiển thị ảnh logo từ thư mục asset.
 - AppBar: Hiển thị logo.
 - Column: giúp các widget con được xếp theo chiều dọc.
 - Padding: Tạo khoảng cách giữa các phần tử con và biên của widget.
 - SizedBox: Tạo khoảng cách giữa các widget với kích thước cố định.
 - Text: Hiển thị các chuỗi văn bản.
 - Row: hiển thị các widget được xếp theo chiều ngang.
 - Expanded: giúp mở rộng kết quả chiếm lấy tối đa không gian có sẵn.
 - Form: quản lý thông tin nhập từ người dùng.
 - ListView: giúp hiển thị danh sách các widget.
 - TextFormField: nhận dữ liệu nhập từ người dùng.
 - DropdownButtonFormField tạo menu thả xuống trong form.
- Các widget đặc biệt trong giao diện:
 - ShortButton: là button tự xây dựng có màu secondary của ứng dụng.
- plugin được sử dụng:
 - provider là plugin được sử dụng để quản lý trạng thái cùng ChangeNotifier
 - image_picker là plugin giúp chọn ảnh từ thư viện điện thoại.
- DecksManager gọi hàm addDeck để thêm bộ thẻ nhờ phương thức POST gửi đến API '/decks'
- Giao diện cũng hỗ trợ chỉnh sửa bộ thẻ, khi đó dữ liệu sẽ được gửi bằng phương thức PATCH (nếu có hình ảnh) hoặc POST (nếu không có hình ảnh) đến API '/decks/\${deck.id}'

7. Giao diện thêm, sửa flashcard

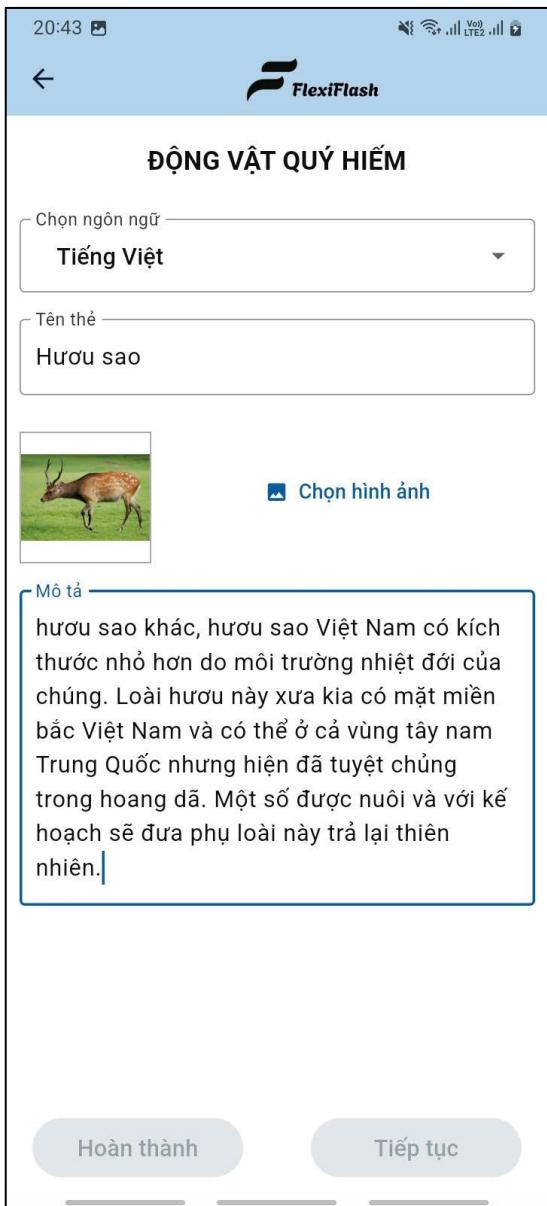
Sau khi thêm bộ thẻ, người dùng thực hiện thêm lần lượt các flashcard. Đây cũng là giao diện chỉnh sửa từng flashcard được chọn từ danh sách chỉnh sửa.



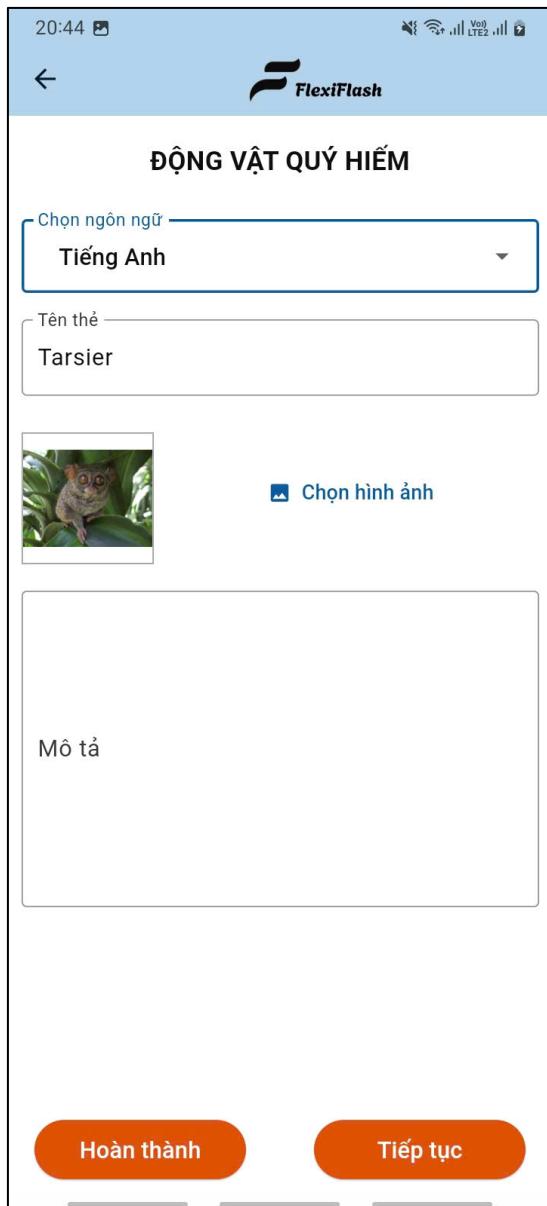
Hình 14 Giao diện thêm flashcard



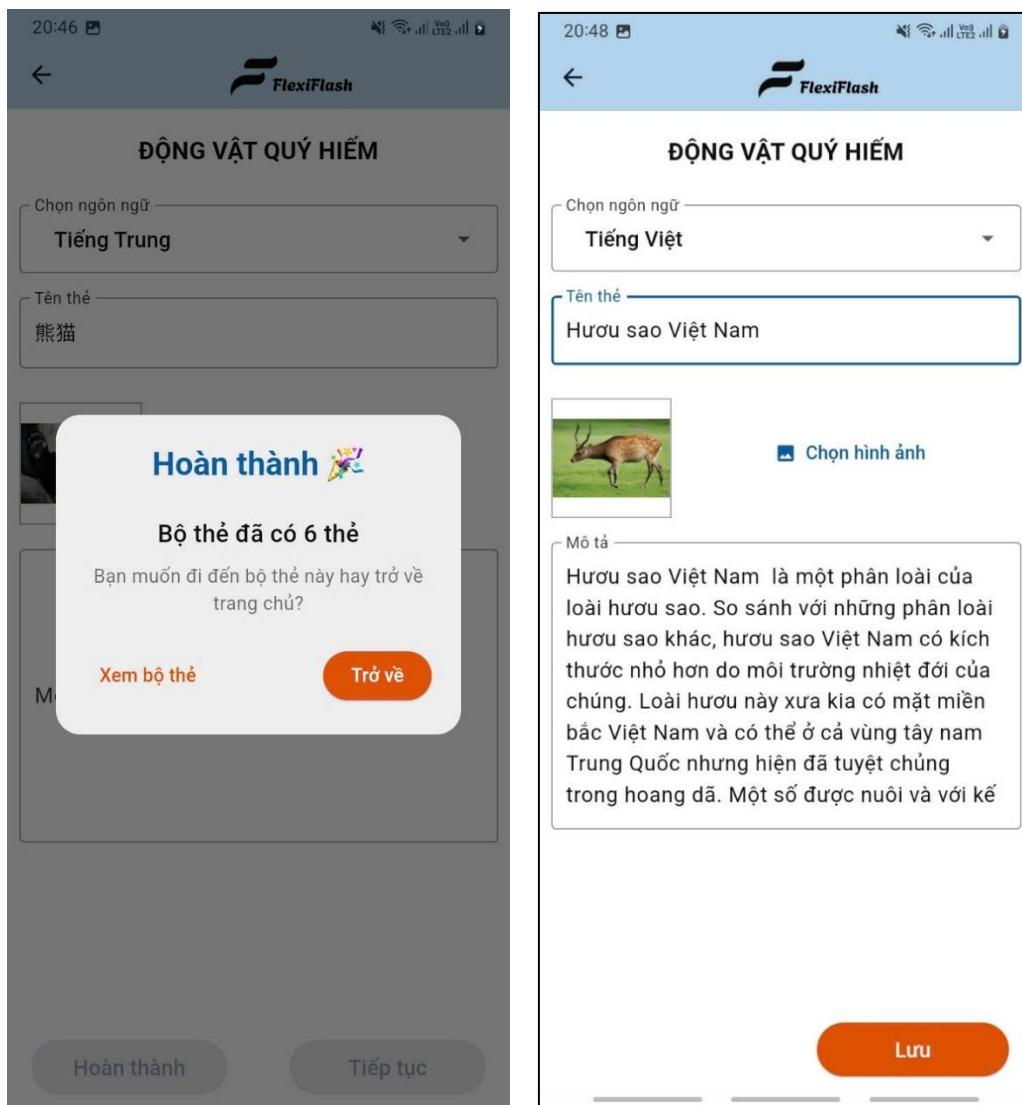
Hình 15 Cảnh báo nội dung không hợp lệ



Hình 16 Thêm một flashcard



Hình 17 Thêm một flahscard không có mô tả



Hình 18 Thông báo đã thêm tất cả flashcard

Hình 19 Giao diện chỉnh sửa một flashcard

Chi tiết cài đặt:

- Bảng flashcards có các trường như sau:
 - id: id của flashcard
 - text: tên của flashcard
 - imgUrl: địa chỉ url chứa hình ảnh của flashcard.
 - description: mô tả của flashcard.
 - isMarked: trạng thái đánh dấu của flashcard
 - deckId: id của bộ thẻ chứa flashcard
 - language: ngôn ngữ của flashcard được sử dụng để đọc tên và mô tả.
 - create_at: thời gian tạo flashcard.
 - update_at: thời gian chỉnh sửa flashcard
- Các widget trong giao diện:

- Scaffold: cung cấp cấu trúc cơ bản cho giao diện.
 - Image.asset(): là phương thức giúp hiển thị ảnh logo từ thư mục asset.
 - AppBar: Hiển thị logo.
 - Column: giúp các widget con được xếp theo chiều dọc.
 - Padding: Tạo khoảng cách giữa các phần tử con và biên của widget.
 - SizedBox: Tạo khoảng cách giữa các widget với kích thước cố định.
 - Text: Hiển thị các chuỗi văn bản.
 - Row: hiển thị các widget được xếp theo chiều ngang.
 - Expanded: giúp mở rộng kết quả chiêm lấy tối đa không gian có sẵn.
 - Form: quản lý thông tin nhập từ người dùng.
 - TextFormField: nhận dữ liệu nhập từ người dùng.
 - DropdownButtonFormField tạo menu thả xuống trong form.
 - ListView: giúp hiển thị danh sách các widget.
- Các widget đặc biệt trong giao diện:
 - ShortButton: là button tự xây dựng có màu secondary của ứng dụng.
 - plugin được sử dụng:
 - provider là plugin được sử dụng để quản lý trạng thái cùng ChangeNotifier
 - image_picker là plugin giúp chọn ảnh từ thư viện điện thoại.
 - FlashcardManager gọi hàm addFlashcard để thêm flashcard vào bộ thẻ bằng phương thức POST gửi đến API '/decks/\$deckId/flashcards'.
 - FlashManager gọi hàm updateFlashcard để chỉnh sửa flashcard bằng phương thức PATCH (nếu có hình ảnh) hoặc POST (nếu không có hình ảnh) đến API '/decks/\$deckId/flashcards/\${flashcard.id}'

8. Giao diện Trang một bộ thẻ

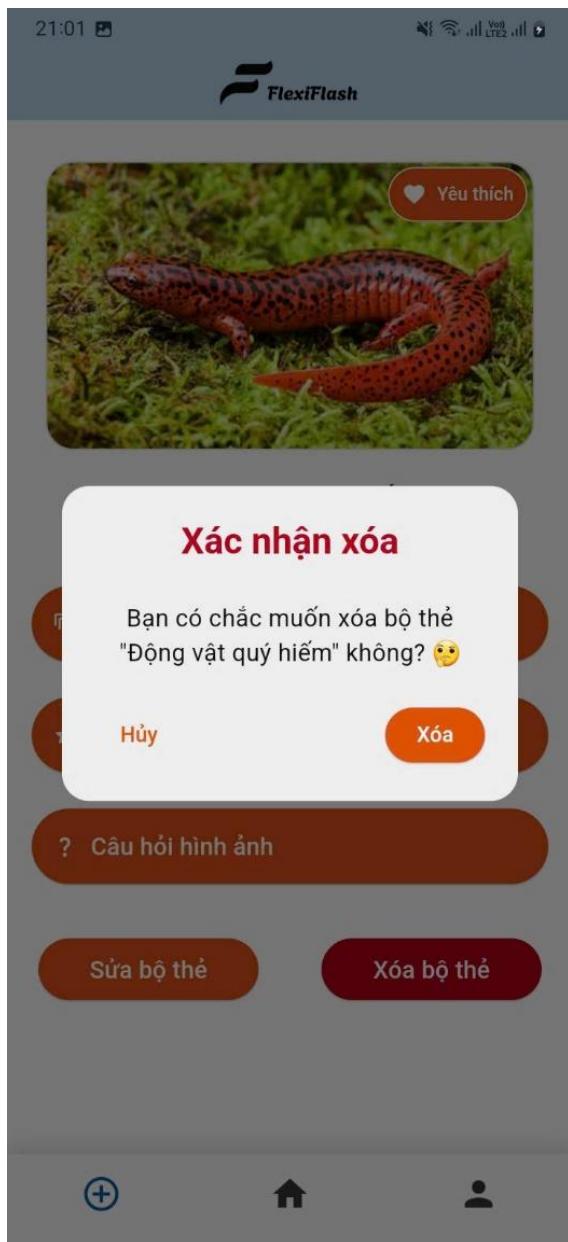
Giao diện này hiển thị hình ảnh, tên và trạng thái yêu thích của bộ thẻ được chọn. Người dùng có thể thực hiện các tác vụ với bộ thẻ được chọn như xem tất cả thẻ, xem thẻ yêu thích, trả lời câu hỏi, và sửa, xóa.



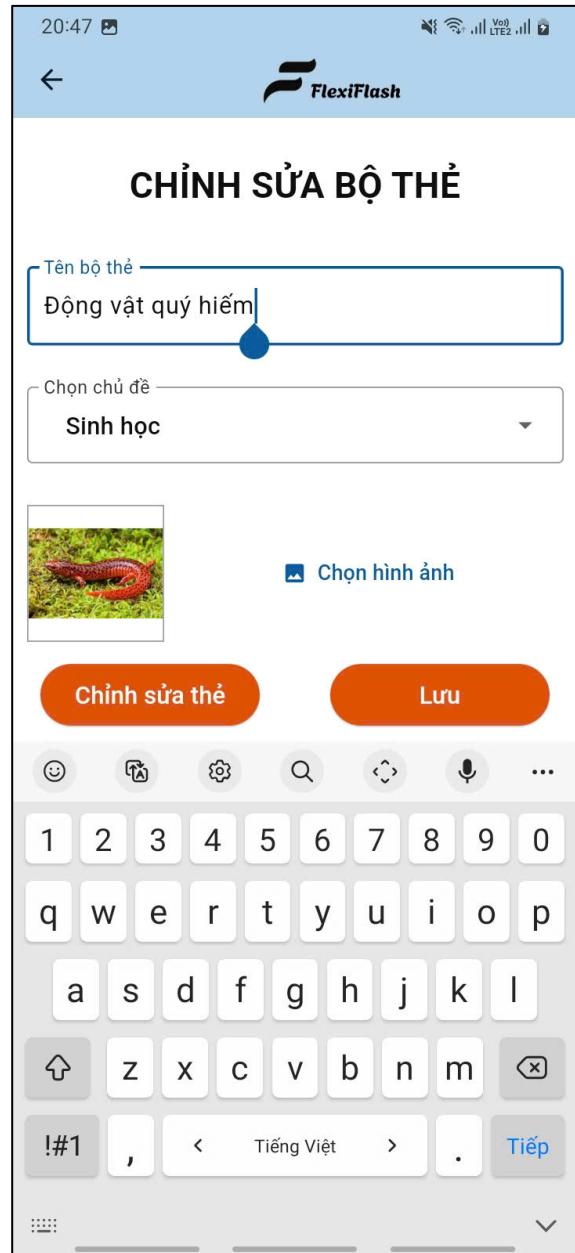
Hình 20 Giao diện chi tiết một bộ thẻ



Hình 21 Yêu thích bộ thẻ



Hình 22 Thông báo xác nhận xóa bộ thẻ



Hình 23 Giao diện chỉnh sửa bộ thẻ

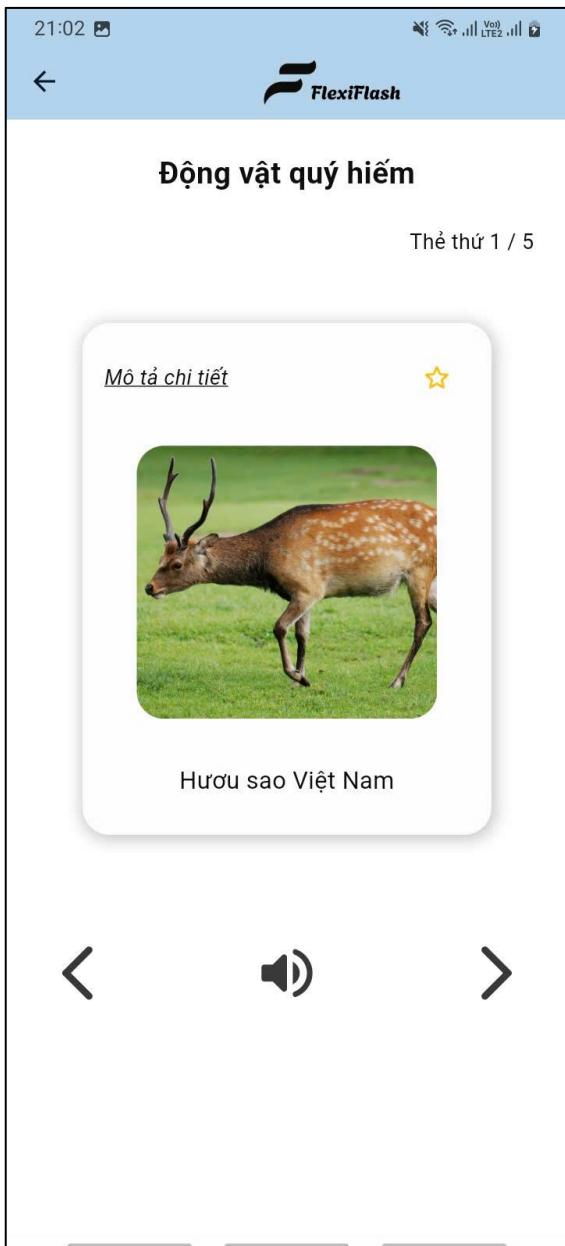
Chi tiết cài đặt:

- Các widget được sử dụng:
 - Scaffold: cung cấp cấu trúc cơ bản cho giao diện.
 - Image.asset(): là phương thức giúp hiển thị ảnh logo từ thư mục asset.
 - AppBar: Hiển thị logo.
 - Column: giúp các widget con được xếp theo chiều dọc.

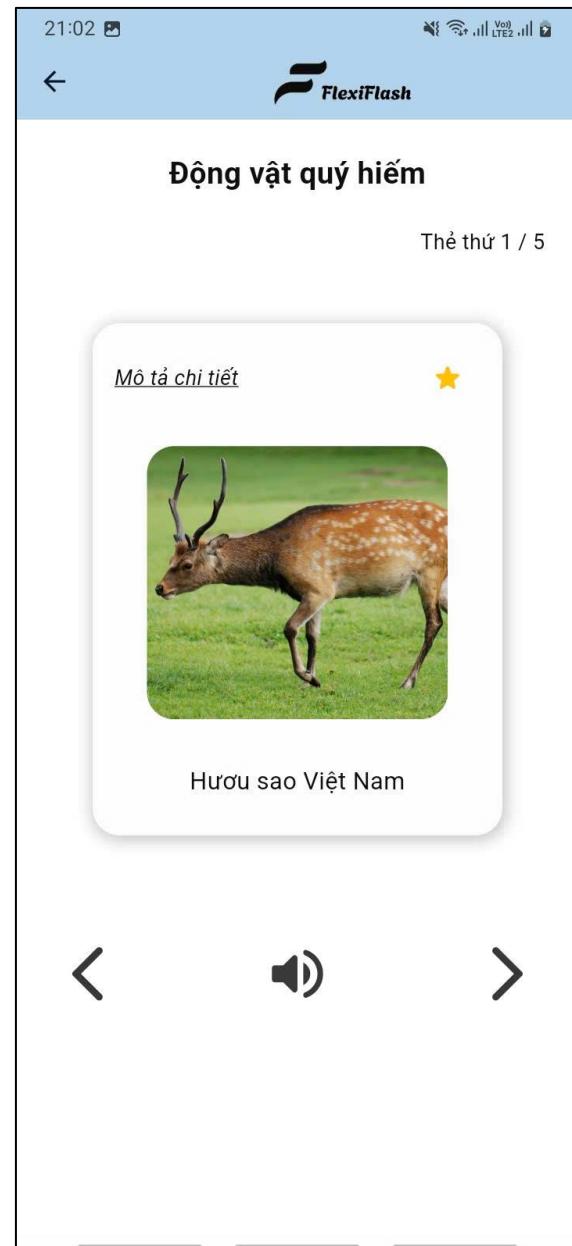
- Padding: Tạo khoảng cách giữa các phần tử con và biên của widget.
- SizedBox: Tạo khoảng cách giữa các widget với kích thước cố định.
- Text: Hiển thị các chuỗi văn bản như tên của bộ thẻ.
- Row: hiển thị các widget được xếp theo chiều ngang.
- Stack: xếp chồng các widget trong giao diện.
- Positioned: định vị widget FavorIcon bên trong Stack để nó nằm trên hình ảnh.
 - Các widget đặc biệt được sử dụng:
- BotNavBar: widget tự xây dựng giúp hiển thị thanh điều hướng bên dưới.
- ShapeDecoration tạo kiểu trang trí chứa hình ảnh và viền ngoài.
- LongButton: : widget tự xây dựng giúp hiển thị các button dài màu secondary của hệ thống.
- WarningButton: widget tự xây dựng giúp hiển thị button màu đỏ.
- FavorButton: widget tự xây dựng giúp hiển thị và tương tác trạng thái yêu thích.
- provider là plugin được sử dụng để quản lý trạng thái cùng ChangNotifier
- Khi thực hiện thích hoặc bỏ thích một bộ thẻ, hàm update sẽ được gọi thông qua phương thức POST gửi yêu cầu thay đổi trạng thái yêu thích đến API '/deck/\${deckId}'.
- Khi người dùng muốn xóa một flashcard, yêu cầu sẽ được gửi bằng phương thức DELETE đến API '/decks/\$id'.

9. Giao diện Trang một flashcard

Giao diện này hiển thị một flashcard trong bộ thẻ được chọn. Người dùng có thể di chuyển lần lượt qua lại giữa các flashcard, nghe tên của từng flashcard, yêu thích hoặc xem mô tả chi tiết của flahscard.



Hình 24 Giao diện hiển thị một flashcard



Hình 25 Đánh dấu một flahscard

Chi tiết cài đặt:

- Các widget được sử dụng:
 - Scaffold: cung cấp cấu trúc cơ bản cho giao diện.
 - Image.asset(): là phương thức giúp hiển thị ảnh logo từ thư mục asset.
 - AppBar: Hiển thị logo.
 - Column: giúp các widget con được xếp theo chiều dọc.

- Padding: Tạo khoảng cách giữa các phần tử con và biên của widget.
- SizedBox: Tạo khoảng cách giữa các widget với kích thước cố định.
- Text: Hiển thị các chuỗi văn bản như tên của thẻ.
- Row: hiển thị các widget được xếp theo chiều ngang.
- IconButton: widget kế hợp giữa các icon và nút bấm
- Container: dùng để chứa và căn chỉnh các widget.
- Các widget đặc biệt được sử dụng:
 - CustomProgressIndicator là widget tự xây dựng để biểu thị đang trong quá trình lấy các thẻ.
 - FlashCard: widget tự xây dựng giúp hiển thị lần lượt các thẻ.
- Các plugin sử dụng:
 - provider là plugin được sử dụng để quản lý trạng thái cùng ChangeNotifier.
 - flutter_tts: giúp đọc văn bản thành giọng nói, hỗ trợ nhiều ngôn ngữ và tùy chỉnh tốc độ đọc.
 - lottie: giúp hiển thị animation dạng JSON được gọi trong CircularProgressIndicator.
- Quản lý trạng thái hiển thị từng flashcard như sau: các flashcards được lấy ra tương ứng với một index, khi chuyển tới thẻ tiếp theo index sẽ tăng lên 1, khi lùi index sẽ giảm 1. Phép tính luôn chia lấy dư với độ dài của danh sách flashcards để các flashcards được xoay vòng
- FlashcardManager là một ChangeNotifier được sử dụng để lấy tất cả flashcard của một bộ thẻ thông qua phươn thức GET gửi đến API ‘dekks/\$deckId/flashcards’

10.Giao diện Chi tiết một flashcard

Giao diện này hiển thị chi tiết của một flashcard được chọn bao gồm tên, hình ảnh và mô tả chi tiết hỗ trợ chuyển văn bản thành lời nói.



Hình 26 Giao diện chi tiết một flahscard

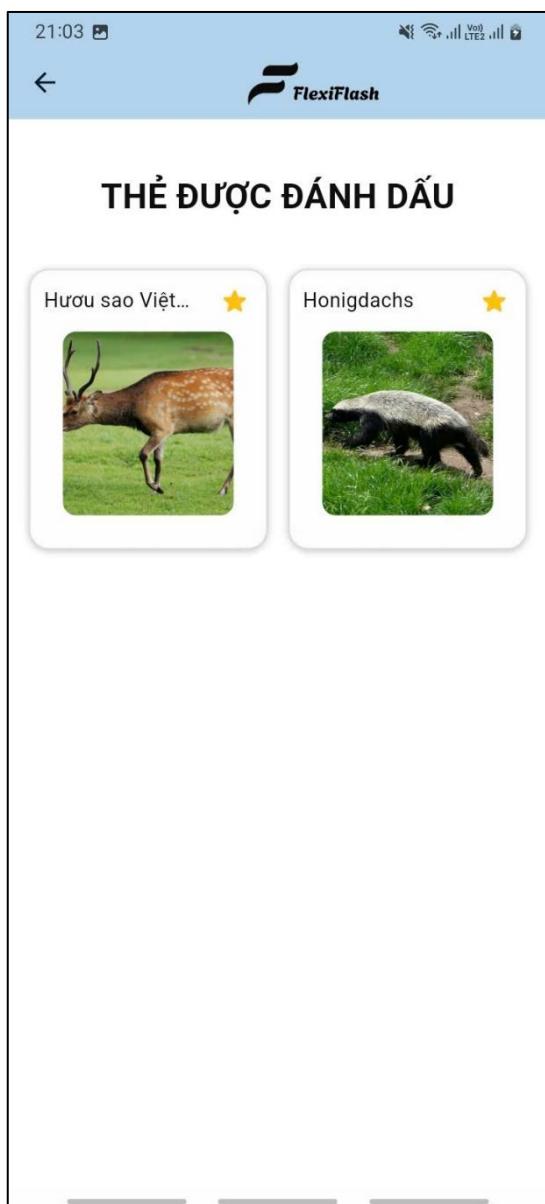
Chi tiết cài đặt:

- Các widget được sử dụng:
 - Scaffold: cung cấp cấu trúc cơ bản cho giao diện.
 - Image.asset(): là phương thức giúp hiển thị ảnh logo từ thư mục asset.
 - AppBar: Hiển thị logo.
 - Column: giúp các widget con được xếp theo chiều dọc.
 - Container: dùng để chứa và căn chỉnh các widget.
 - Padding: Tạo khoảng cách giữa các phần tử con và biên của widget.
 - SizedBox: Tạo khoảng cách giữa các widget với kích thước cố định.
 - Text: Hiển thị các chuỗi văn bản.
 - Row: hiển thị các widget được xếp theo chiều ngang.
 - Column: giúp các widget con được xếp theo chiều dọc.
 - IconButton: widget kế hợp giữa các icon và nút bấm.
- Các widget đặc biệt được sử dụng:

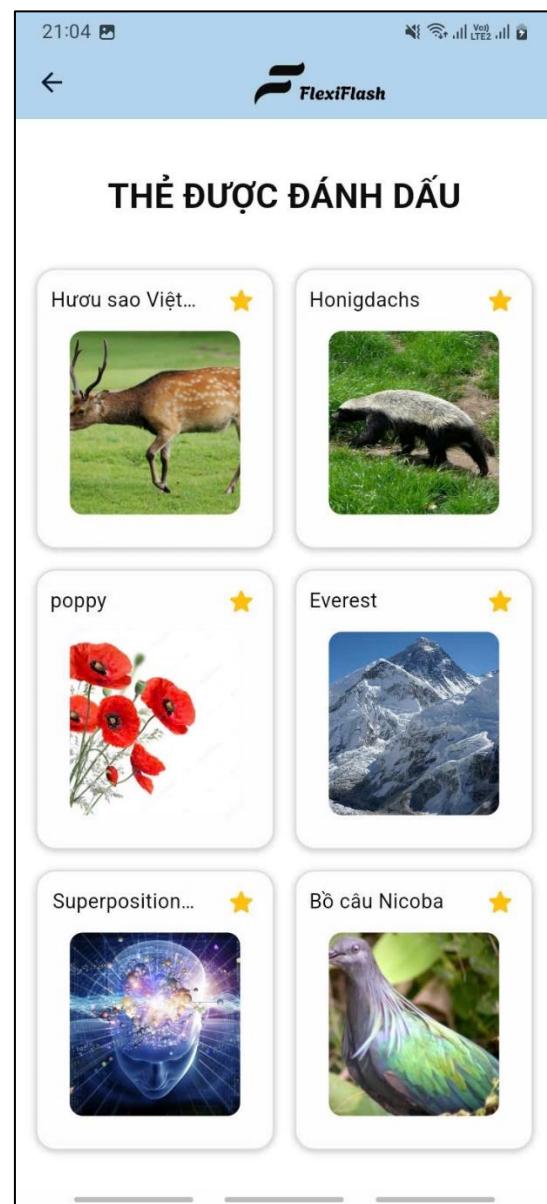
- ShapeDecoration tạo kiểu trang trí chứa hình ảnh và viền ngoài.
- Các plugin sử dụng:
 - provider là plugin được sử dụng để quản lý trạng thái cùng ChangNotifier
 - flutter_tts: giúp đọc văn bản thành giọng nói, hỗ trợ nhiều ngôn ngữ và tùy chỉnh tốc độ đọc.
- AuthManager được gọi để sử hàm speak đọc miêu tả của flashcard.

11. Giao diện Các flashcard được đánh dấu

Giao diện hiển thị danh sách các flashcard được đánh dấu của một bộ thẻ hay tất cả flahscard được người dùng đánh dấu.



Hình 27 Danh sách flashcard được đánh dấu
của một bộ thẻ



Hình 28 Giao diện tất cả flashcard được đánh dấu

Chi tiết cài đặt:

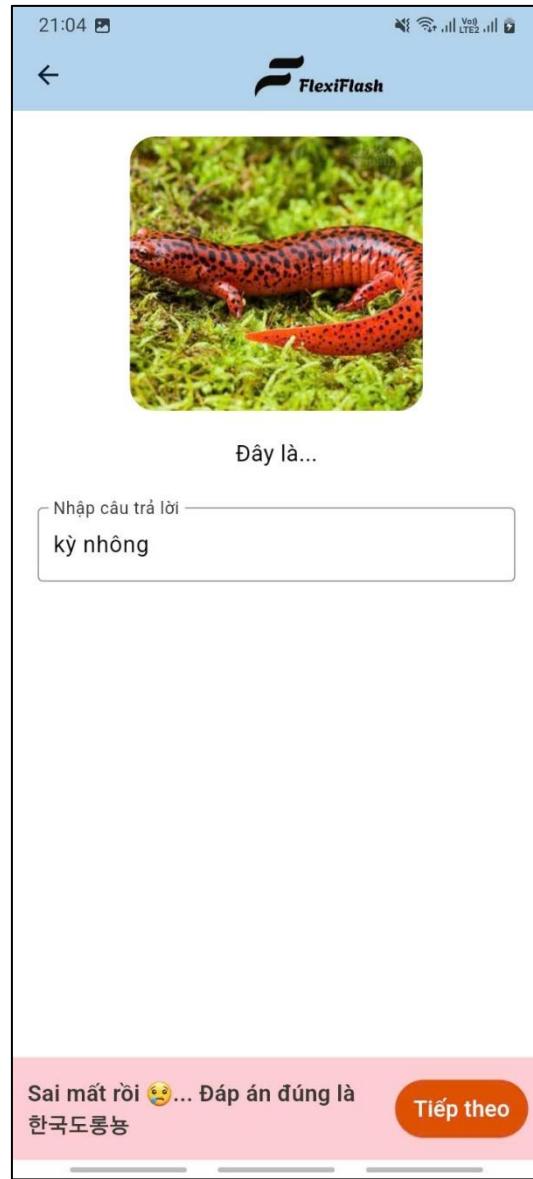
- Các widget được sử dụng:
 - Scaffold: cung cấp cấu trúc cơ bản cho giao diện.
 - Image.asset(): là phương thức giúp hiển thị ảnh logo từ thư mục asset.
 - AppBar: Hiển thị logo.
 - Column: giúp các widget con được xếp theo chiều dọc.
 - Container: dùng để chứa và căn chỉnh các widget.
 - Padding: Tạo khoảng cách giữa các phần tử con và biên của widget.
 - SizedBox: Tạo khoảng cách giữa các widget với kích thước cố định.
 - Text: Hiển thị các chuỗi văn bản như tiêu đề widget.
 - Row: hiển thị các widget được xếp theo chiều ngang.
 - IconButton: widget kế hợp giữa các icon và nút bấm.
 - SingleChildScrollView: cho phép cuộn khi nội dung quá dài làm tràn màn hình.
- Các widget đặc biệt được sử dụng:
 - FutureBuilder: hiển thị dữ liệu bất đồng bộ là các flashcard được lấy.
 - FlashcardGrid: widget tự xây dựng giúp căn chỉnh danh sách flashcard theo lối.
 - FlashcardGridItem: widget tự xây dựng giúp hiển thị nội dung của một widget
 - CustomProgressIndicator là widget tự xây dựng để biểu thị đang trong quá trình lấy các thẻ
- Các plugin sử dụng:
 - provider là plugin được sử dụng để quản lý trạng thái cùng ChangeNotifier
 - lottie: giúp hiển thị animation dạng JSON được gọi trong CircularProgressIndicator.
- FlashcardManager là một ChangeNotifier được sử dụng để lấy flashcard của một bộ thẻ thông qua phương thức GET gửi đến API ‘dekks/\$deckId/flashcards’, sau đó dùng hàm .where để so sánh trường isMarked của từng flashcard để lọc và hiển thị flashcard được đánh dấu.
- FlashcardManager cũng dùng hàm fetchMarkedFlashCards để lấy tất cả flashcard được đánh dấu thông qua phương thức GET gửi đến API ‘/marked-flashcards’

12. Giao diện Trả lời câu hỏi

Giao diện gồm các flashcard xuất hiện ngẫu nhiên với hình ảnh, yêu cầu người dùng nhập đúng tên của flashcard đó. Khi nhấn “Trả lời” đáp án sẽ hiện ra, khi đến flashcard cuối cùng người dùng nhận một thông báo kết quả.



Hình 29 Giao diện trả lời câu hỏi



Hình 30 Trả lời câu hỏi sai



Hình 31 Trả lời câu hỏi đúng



Hình 32 Thông báo kết quả

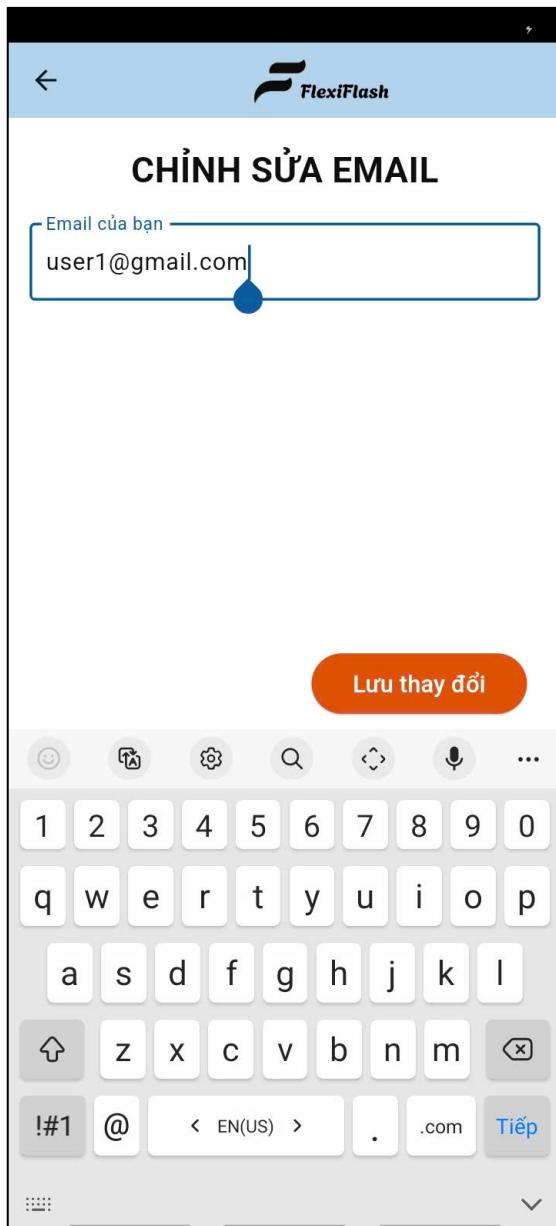
Chi tiết cài đặt:

- Các widget được sử dụng:
 - Scaffold: cung cấp cấu trúc cơ bản cho giao diện.
 - Image.asset(): là phương thức giúp hiển thị ảnh logo từ thư mục asset.
 - AppBar: Hiển thị logo.
 - Column: giúp các widget con được xếp theo chiều dọc.
 - Container: dùng để chứa và căn chỉnh các widget.
 - Padding: Tạo khoảng cách giữa các phần tử con và biên của widget.
 - SizedBox: Tạo khoảng cách giữa các widget với kích thước cố định.

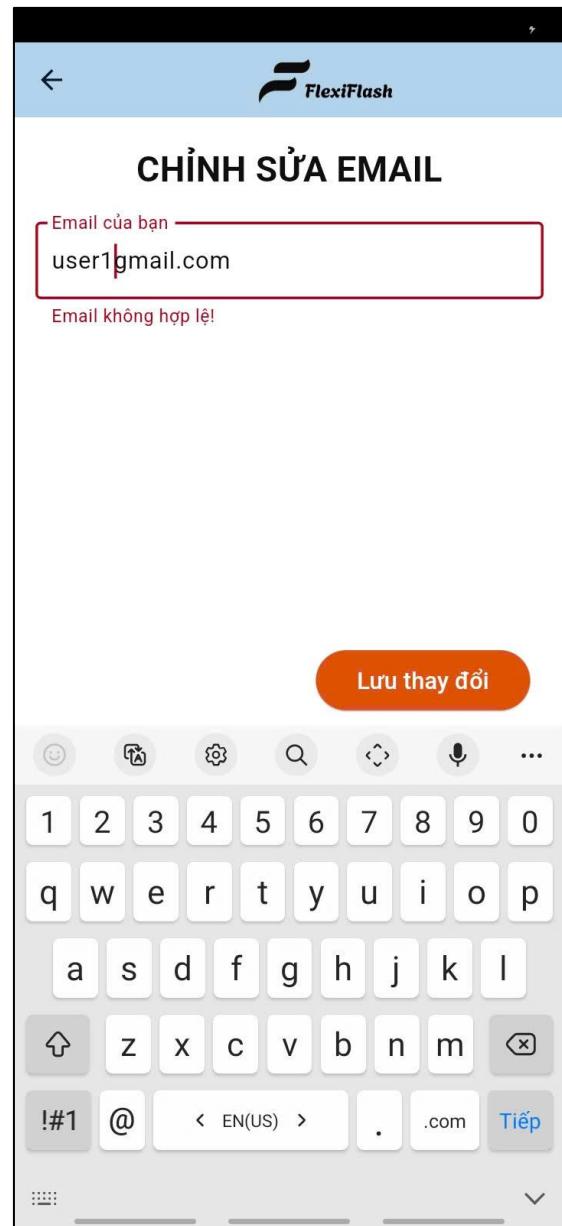
- Text: Hiển thị các chuỗi văn bản.
- Row: hiển thị các widget được xếp theo chiều ngang.
- IconButton: widget kết hợp giữa các icon và nút bấm.
- SingleChildScrollView: cho phép cuộn khi bàn phím xuất hiện làm tràn màn hình.
- TextField: nhận dữ liệu nhập vào ô trả lời.
- Expanded: giúp mở rộng kết quả chiếm lấy tối đa không gian có sẵn.
- Các widget đặc biệt được sử dụng:
 - PieChart: biểu đồ tỷ lệ các thành phần dưới dạng biểu đồ tròn.
 - PieChartData: giúp vẽ biểu đồ trong từ các sections.
 - PieChartSectionData: nắm tổng sections, nó là 1 trong những thành phần của biểu đồ tròn.
 - CustTextButton, CusFillButton: là widget tự xây dựng hiển thị các button.
- Các plugin sử dụng:
 - provider là plugin được sử dụng để quản lý trạng thái cùng ChangeNotifier
 - fl_char: là plugin hỗ trợ vẽ các biểu đồ.
- Quản lý trạng thái hiển thị đáp án và kết quả: một biến _feedbackMessage ban đầu được gán là null, button “Trả lời” được hiển thị. Khi người dùng trả lời, hàm _checkAnswer được gọi để kiểm tra đáp án, nếu câu trả lời là đúng thì _feedbackMessage được cập nhật trạng thái và được hiển thị. Khi đến câu tiếp theo trạng thái của feedbackMessage quay lại là null. Khi đến câu hỏi cuối cùng, tức là index hiện tại nhỏ độ dài danh sách flashcards -1, thông báo kết quả sẽ được hiển thị.
- FlashcardManager là một ChangeNotifier được sử dụng để lấy flashcard của một bộ thẻ thông qua phương thức GET gửi đến API ‘dekks/\$deckId/flashcards’, sau đó xáo trộn các flashcards này để hiển thị câu hỏi

13. Giao diện chỉnh sửa email

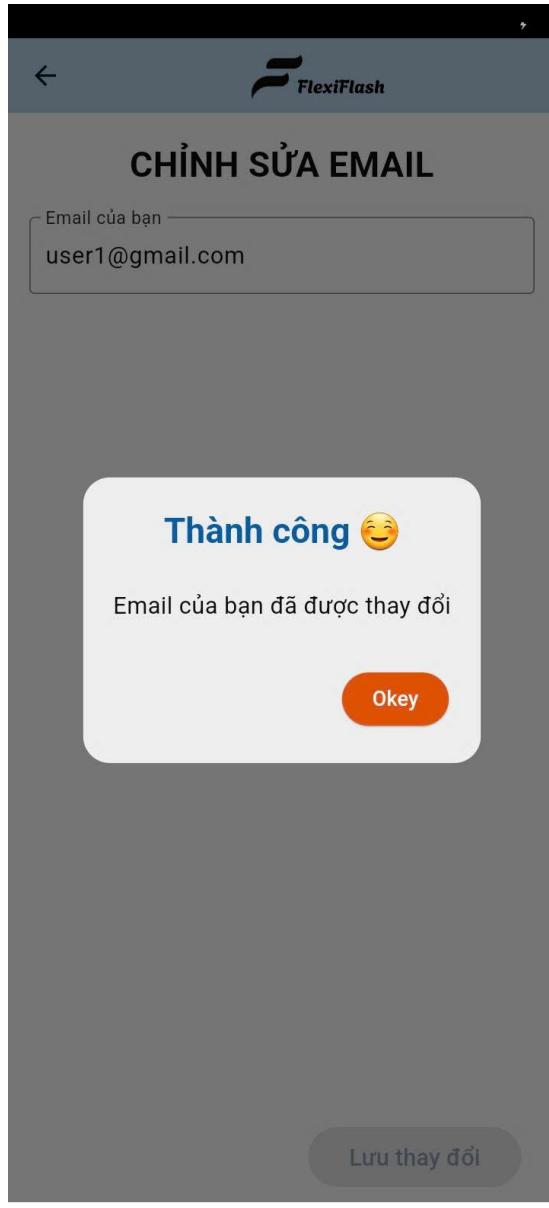
Giao diện chỉnh sửa email gồm có 1 ô nhập liệu hiển thị email hiện tại. Người dùng thực hiện chỉnh sửa và lưu email sau khi chỉnh sửa với điều kiện email mới phải hợp lệ và không rỗng.



Hình 33 Giao diện chỉnh sửa email



Hình 34 Cảnh báo email không hợp lệ



Hình 35 Thông báo chỉnh sửa email thành công

Chi tiết cài đặt:

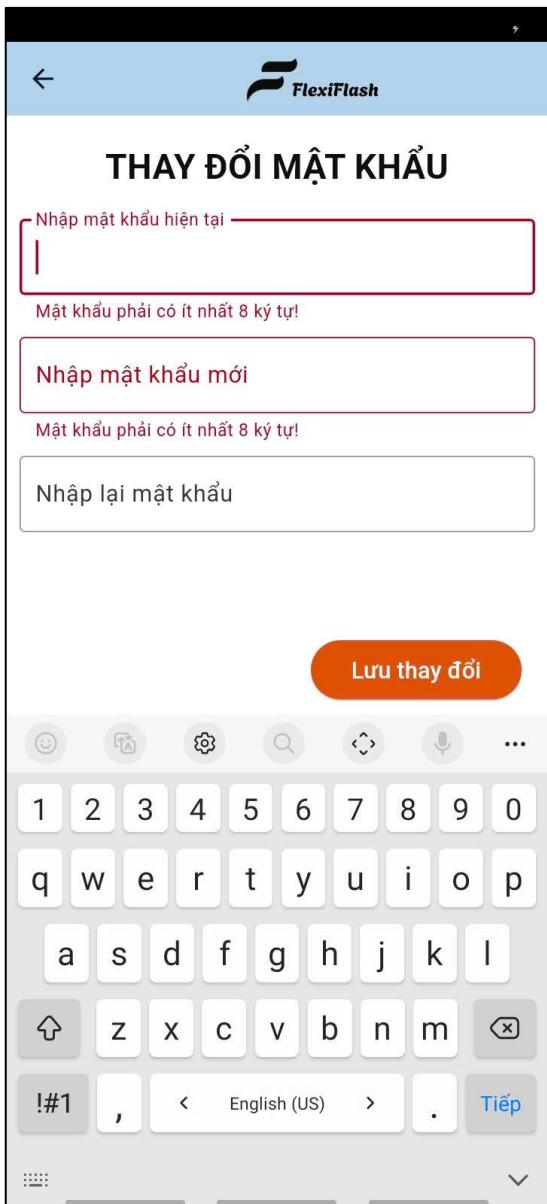
- Các widget được sử dụng:
 - Scaffold: cung cấp cấu trúc cơ bản cho giao diện.
 - Image.asset(): là phương thức giúp hiển thị ảnh logo từ thư mục asset.
 - AppBar: Hiển thị logo.
 - Column: giúp các widget con được xếp theo chiều dọc.
 - Container: dùng để chứa và căn chỉnh các widget.
 - Padding: Tạo khoảng cách giữa các phần tử con và biên của widget.
 - SizedBox: Tạo khoảng cách giữa các widget với kích thước cố định.
 - Text: Hiển thị các chuỗi văn bản.

- Row: hiển thị các widget được xếp theo chiều ngang.
- Expanded: giúp mở rộng kết quả chiếm lấy tối đa không gian có sẵn.
- Form: quản lý dữ liệu nhập từ người dùng.
- TextFormField: hiển thị email cũ cho người dùng chỉnh sửa.
- Các widget đặc biệt trong giao diện:
 - ShortButton: là button tự xây dựng có màu secondary của ứng dụng.
- Các plugin sử dụng:
 - provider là plugin được sử dụng để quản lý trạng thái cùng ChangNotifier
- AuthManager gọi hàm changeEmail gửi email đã thay đổi bằng phương thức PATCH đến API '/changeEmail'.

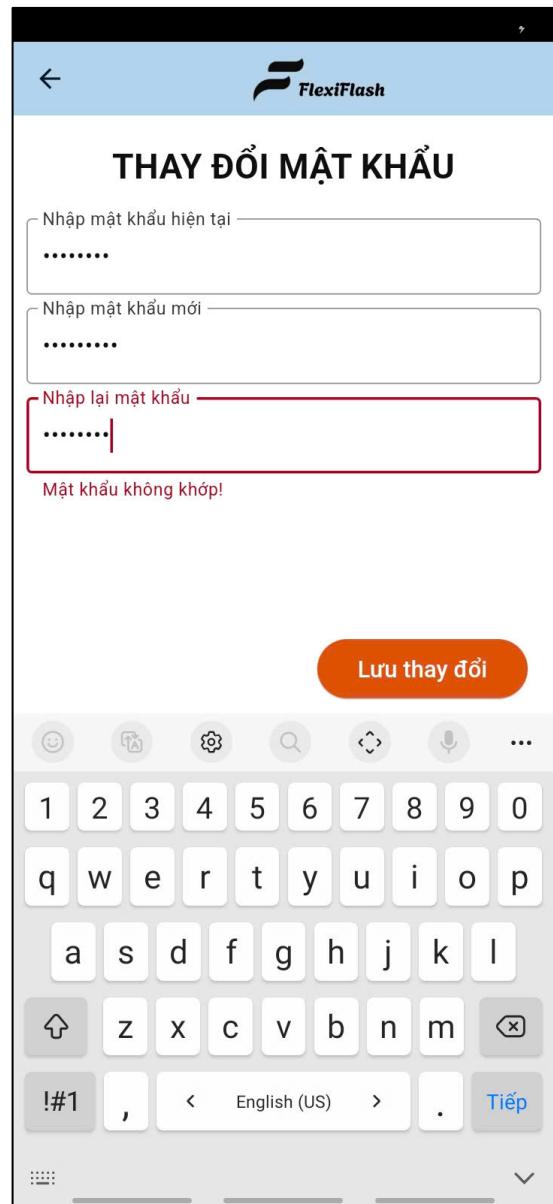
14. Giao diện chỉnh sửa password

Giao diện gồm các trường mật khẩu cũ, mật khẩu mới và nhập lại mật khẩu mới.

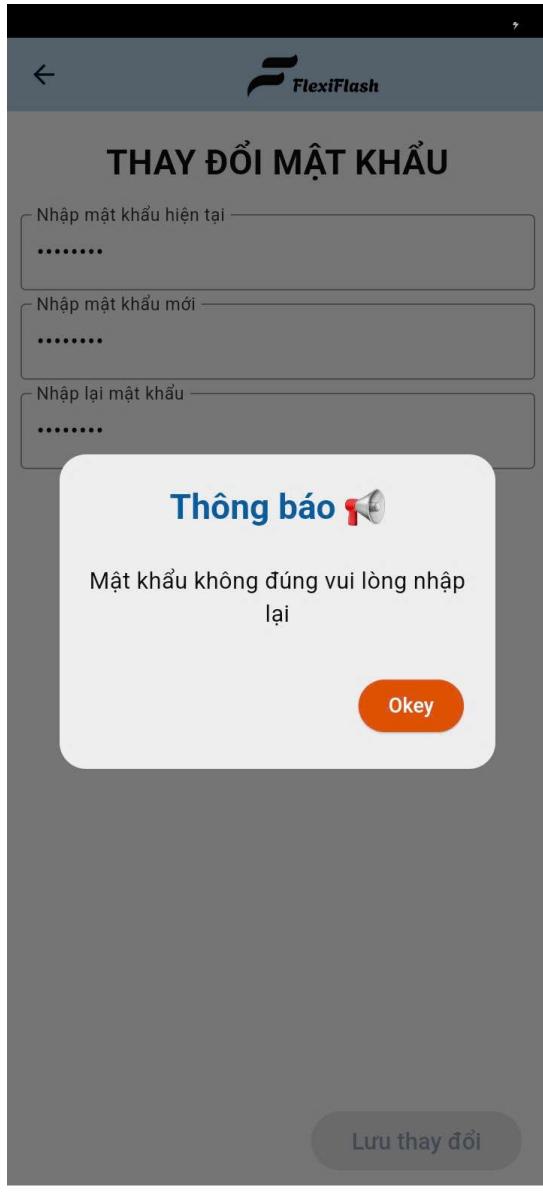
Người dùng tiến hành nhập đầy đủ thông tin để thay đổi mật khẩu với điều kiện mật khẩu mới nhiều hơn 8 ký tự và không rỗng.



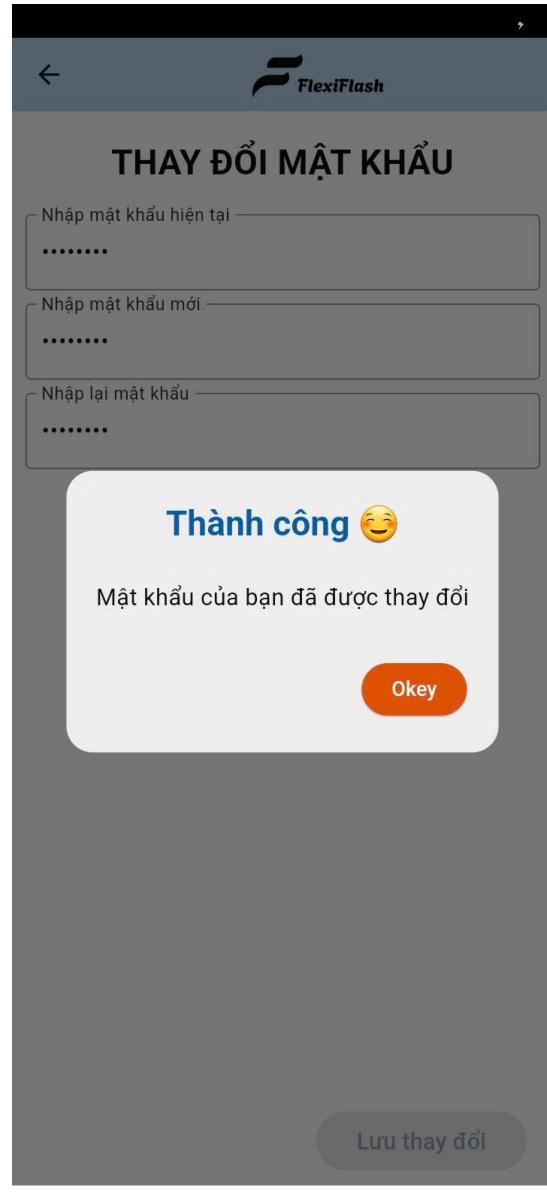
Hình 36 Giao diện đổi mật khẩu



Hình 37 Cảnh báo mật khẩu không khớp



Hình 38 Thông báo mật khẩu không đúng



Hình 39 Thông báo đổi mật khẩu thành công

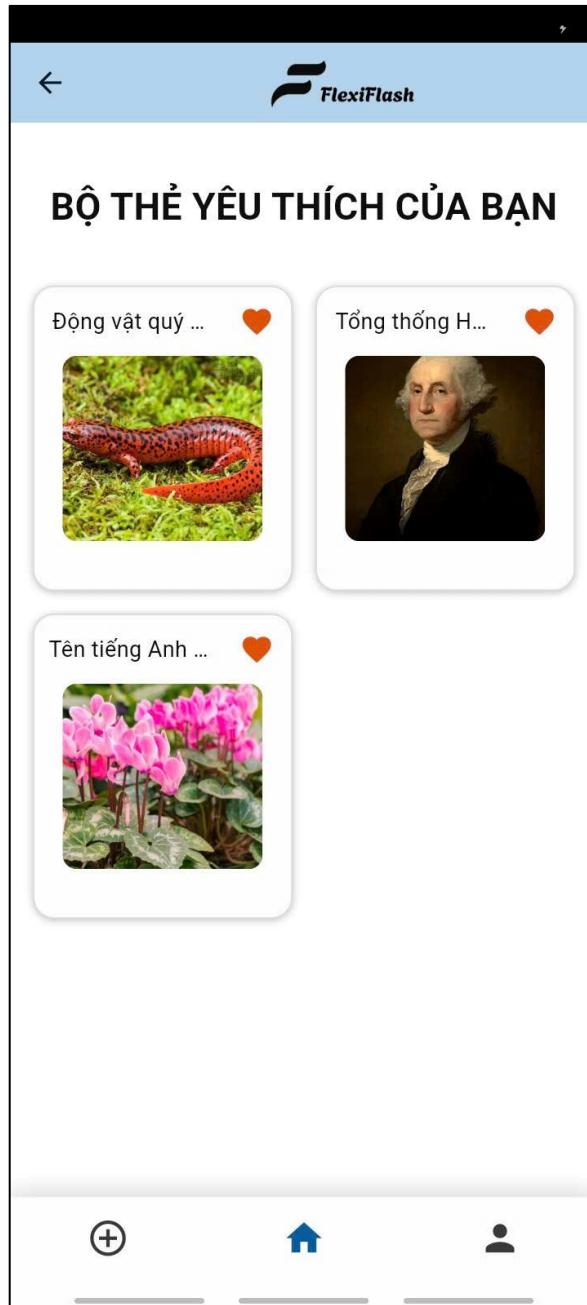
Chi tiết cài đặt:

- Các widget được sử dụng:
 - Scaffold: cung cấp cấu trúc cơ bản cho giao diện.
 - Image.asset(): là phương thức giúp hiển thị ảnh logo từ thư mục asset.
 - AppBar: Hiển thị logo.
 - Column: giúp các widget con được xếp theo chiều dọc.
 - Container: dùng để chứa và căn chỉnh các widget.
 - Padding: Tạo khoảng cách giữa các phần tử con và biên của widget.

- SizedBox: Tạo khoảng cách giữa các widget với kích thước cố định.
- Text: Hiển thị các chuỗi văn bản.
- Row: hiển thị các widget được xếp theo chiều ngang.
- Expanded: giúp mở rộng kết quả chiêm lấy tối đa không gian có sẵn.
- Form: quản lý dữ liệu nhập từ người dùng.
- TextFormField: hiển thị email cũ cho người dùng chỉnh sửa.
- Các widget đặc biệt trong giao diện:
 - ShortButton: là button tự xây dựng có màu secondary của ứng dụng.
- Các plugin sử dụng:
 - provider là plugin được sử dụng để quản lý trạng thái cùng ChangNotifier
- AuthManager gọi hàm changePass gửi dữ liệu bằng phương thức PATCH đến API '/changePassword'.

15. Giao diện Các bộ thẻ được yêu thích

Giao diện hiển thị danh sách các thẻ ở trạng thái yêu thích.



Hình 40 Giao diện danh sách các bộ thẻ được yêu thích

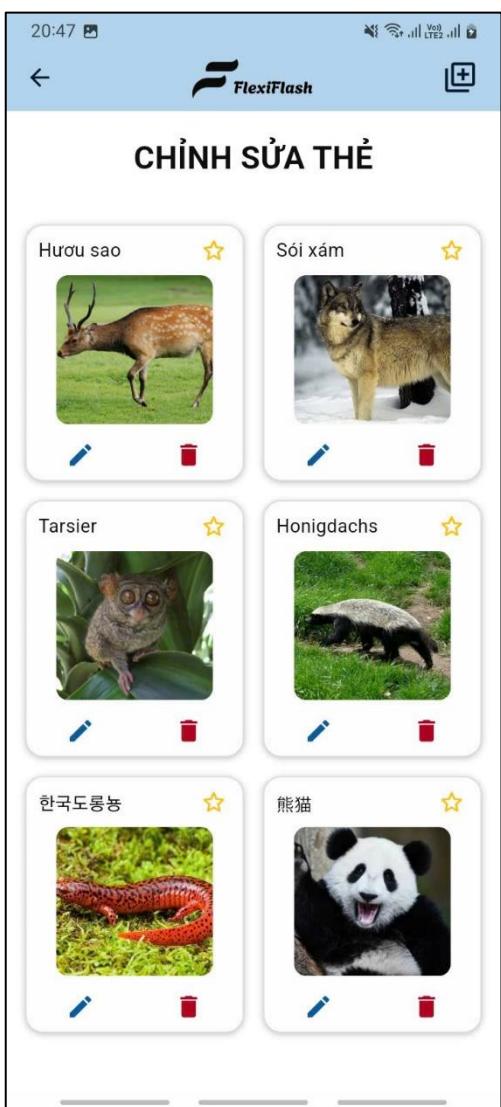
Chi tiết cài đặt:

- Các widget được sử dụng:
 - Scaffold: cung cấp cấu trúc cơ bản cho giao diện.
 - Image.asset(): là phương thức giúp hiển thị ảnh logo từ thư mục asset.
 - AppBar: Hiển thị logo.
 - Column: giúp các widget con được xếp theo chiều dọc.
 - Container: dùng để chứa và căn chỉnh các widget.
 - Padding: Tạo khoảng cách giữa các phần tử con và biên của widget.
 - SizedBox: Tạo khoảng cách giữa các widget với kích thước cố định.

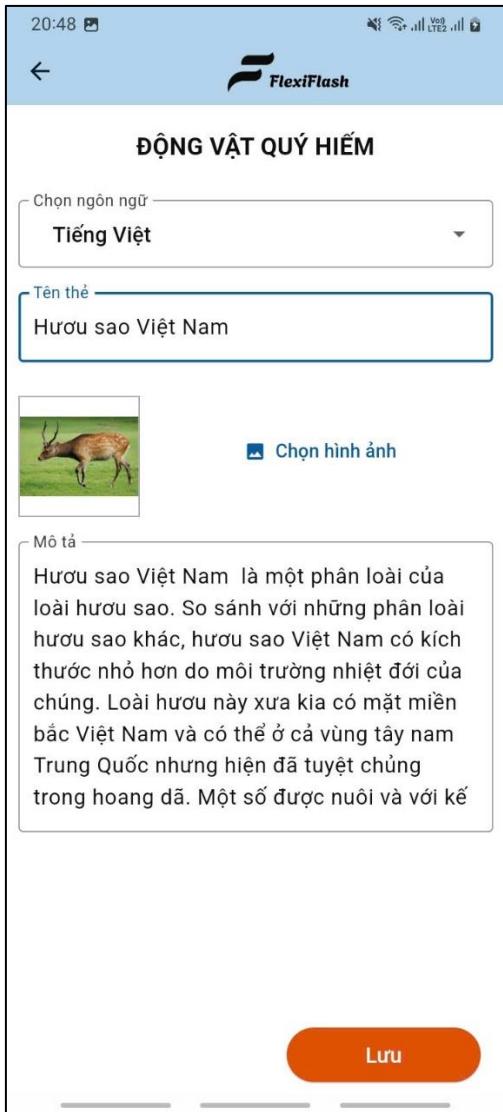
- Text: Hiển thị các chuỗi văn bản.
- Row: hiển thị các widget được xếp theo chiều ngang.
- Expanded: giúp mở rộng kết quả chiếm lấy tối đa không gian có sẵn.
- IconButton: widget kế hợp icon trái tim và nút bấm
- Các widget đặc biệt được sử dụng:
 - CustomProgressIndicator là widget tự xây dựng giúp tùy chỉnh để hiển thị đang trong quá trình lấy các bộ thẻ.
 - FutureBuilder: hiển thị dữ liệu bất đồng bộ là các bộ thẻ được lấy.
 - GridDeck: widget tự xây dựng giúp chia các deck thành lưới phù hợp.
 - GridDeckItem: widget tự xây dựng giúp hiển thị một thẻ trong.
- Các plugin sử dụng:
 - provider là plugin được sử dụng để quản lý trạng thái cùng ChangeNotifier.
 - lottie: giúp hiển thị animation dạng JSON được gọi trong CircularProgressIndicator.
- DeckManager được sử dụng để lấy danh sách các bộ thẻ có trong cơ sở dữ liệu thông qua phương thức GET đến API '/deck', sau đó dùng hàm .where() với điều kiện là trường isFavorite của bộ thẻ là true sẽ được hiển thị.

16. Giao diện danh sách chỉnh sửa các flashcard

Danh sách hiển thị các flashcard đi kèm với các tác vụ xóa, chỉnh sửa từng flashcard. Trên thanh App Bar có button icon hỗ trợ việc thêm flashcard cho bộ thẻ



Hình 41 Giao diện danh sách chỉnh sửa flashcard của 1 bộ thẻ



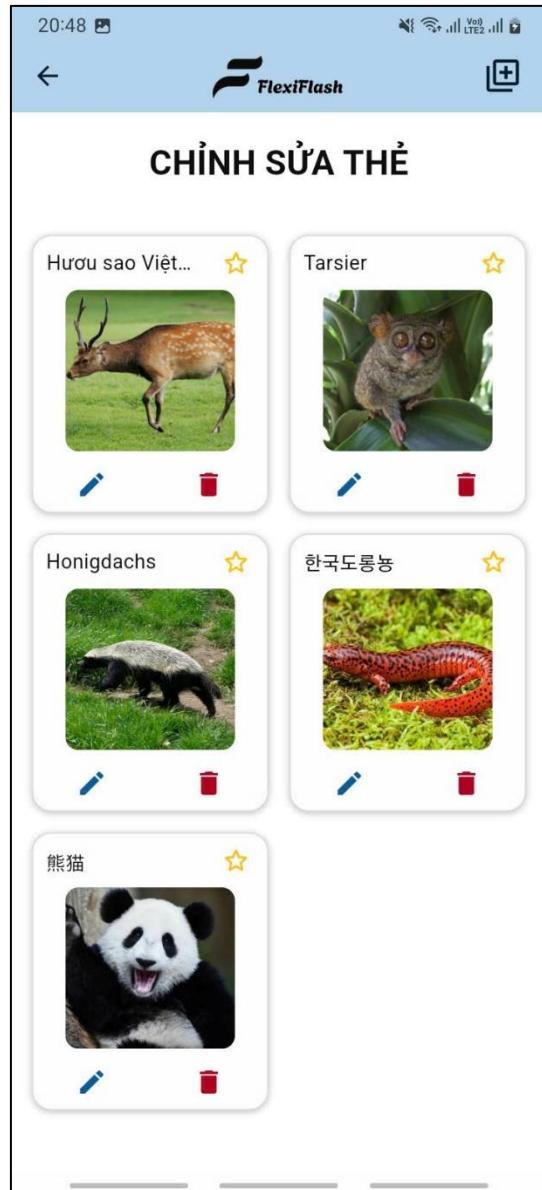
Hình 42 Chỉnh sửa một flahscard



Hình 43 Sau khi chỉnh sửa



Hình 44 Thông báo xác nhận xóa một flashcard



Hình 45 Xóa flashcard thành công

Chi tiết cài đặt:

- Các widget được sử dụng:
 - Scaffold: cung cấp cấu trúc cơ bản cho giao diện.
 - Image.asset(): là phương thức giúp hiển thị ảnh logo từ thư mục asset.
 - AppBar: Hiển thị logo.
 - Column: giúp các widget con được xếp theo chiều dọc.
 - Container: dùng để chứa và căn chỉnh các widget.
 - Padding: Tạo khoảng cách giữa các phần tử con và biên của widget.
 - SizedBox: Tạo khoảng cách giữa các widget với kích thước cố định.

- Text: Hiển thị các chuỗi văn bản như tiêu đề widget.
- Row: hiển thị các widget được xếp theo chiều ngang.
- IconButton: widget kết hợp giữa các icon và nút bấm.
- SingleChildScrollView: cho phép cuộn khi nội dung quá dài làm tràn màn hình.
- Các widget đặc biệt được sử dụng:
 - FutureBuilder: hiển thị dữ liệu bất đồng bộ là các flashcard được lấy.
 - FlashcardGrid: widget tự xây dựng giúp căn chỉnh danh sách flashcard theo lưới.
 - FlashcardGridItem: widget tự xây dựng giúp hiển thị nội dung của một widget
 - CircularProgressIndicator là widget tự xây dựng để hiển thị đang trong quá trình lấy các thẻ
- Các plugin sử dụng:
 - provider là plugin được sử dụng để quản lý trạng thái cùng ChangeNotifier
 - lottie: giúp hiển thị animation dạng JSON được gọi trong CircularProgressIndicator.
- FlashcardManager là một ChangeNotifier được sử dụng để lấy flashcard của một bộ thẻ thông qua phương thức GET gửi đến API 'decks/\$deckId/flashcards'.
- Khi người dùng muốn xóa một flashcard, yêu cầu sẽ được gửi bằng phương thức DELETE đến API '/decks/\$deckId/flashcards/\$flashcardId'.

TÀI LIỆU THAM KHẢO

1. Tài liệu học phần CT484 của TS Bùi Võ Quốc Bảo.
2. Tài liệu tham khảo Flutter: <https://api.flutter.dev/index.html>
3. Tài liệu tham khảo Dart: <https://dart.dev/docs>
4. Tài liệu tham khai Laravel: <https://laravel.com/docs/12.x/readme>
5. Tài liệu tham khai Laragon: <https://laragon.org/docs/>