TRƯỜNG ĐẠI HỌC CẦN THƠ

TRƯỜNG CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG

BÁO CÁO DỰ ÁN CUỐI KỲ HỌC PHẦN PHÁT TRIỂN ỨNG DỤNG DI ĐỘNG

TÊN DỰ ÁN: XÂY DỰNG APP ĐỌC CÔNG THỨC NẤU ĂN

Link Github mã nguồn:
https://github.com/24-25Sem2-Courses/ct48401-project-Jolieboo.git

NHÓM HỌC PHẦN: **CT484 - 01**

GIẢNG VIÊN HƯỚNG DẪN: **TIẾN SĨ BÙI VÕ QUỐC BẢO**

Họ tên - MSSV Nhóm sinh viên thực hiện:

PHAN TRẦN THẢO DUY - B2111789

NGUYỄN PHÚ LÂM - B2105548

HỌC KÌ II, NH: 2024 - 2025

MỤC LỤC

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG

BẢNG PHÂN CÔNG CÔNG VIỆC

Họ và tên	MSSV	Tên công việc
Nguyễn Phú Lâm	B2105548	• Lên ý tưởng, thiết kế database • Phân chia công việc • Dựng khung dự án. • Xây dựng các trang: Hiển thị công thức nấu ăn, Yêu thích, Profile, Admin. • Chỉnh sửa và kiểm thử dự án. • Viết báo cáo • Có tham gia 100% các công việc được phân chia.
Phan Trần Thảo Duy	B2111789	• Thu thập nguồn hình, tìm kiếm tài liệu liên quan. • Xây dựng các trang: Loại món ăn, thêm mới món ăn, trang khởi động, icon dự án. • Thiết kế giao diện. • Viết báo cáo • Có tham gia 100% các công việc được phân chia.

NỘI DUNG

I. TỔNG QUAN

Ngày nay, với sự phát triển của xã hội, con người ngày càng bận rộn với những bữa ăn. Nhưng với nhiều gia đình, nấu ăn là một trong những việc quan trọng, nuôi sống sức khỏe bản thân của các thành viên.

Healthious - ứng dụng đọc công thức nấu ăn, là một nền tảng chia sẻ công thức nấu ăn được phát triển bằng **Flutter**, tích hợp với backend **PocketBase** để quản lý dữ liệu. Ứng dụng cho phép người dùng khám phá, thêm, và yêu thích các công thức nấu ăn, phân loại theo danh mục, cũng như xem chi tiết công thức với hình ảnh, nguyên liệu và các bước thực hiện.

Ngoài ra, ứng dụng hỗ trợ quản lý người dùng với vai trò **Admin** để phê duyệt công thức, cùng với các tính năng như đăng nhập/đăng xuất, quản lý hồ sơ cá nhân, và giao diện thân thiện với người dùng nhờ sử dụng Provider để quản lý trạng thái.

II. CHI TIẾT CHỨC NĂNG

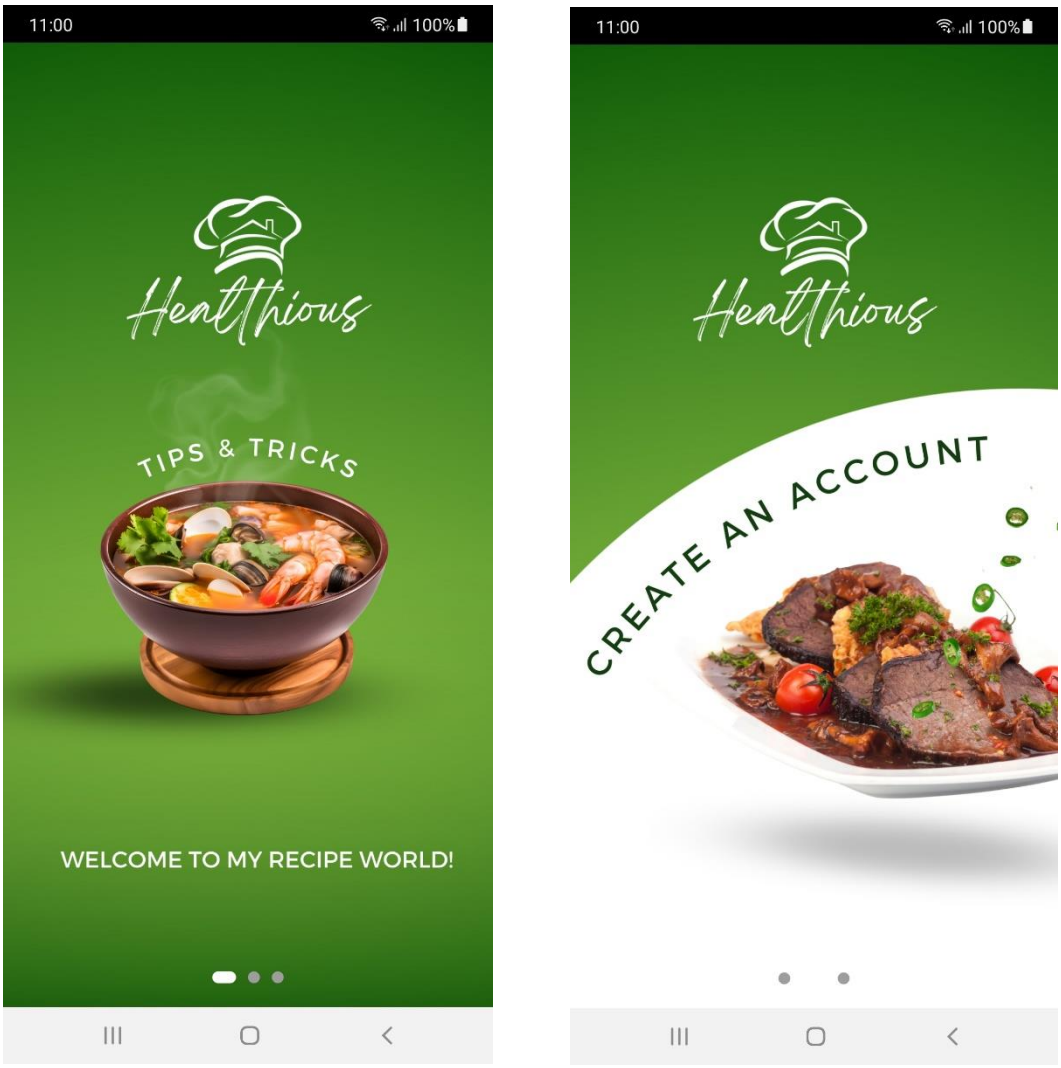
Các giao diện demo dưới đây được thực hiện trên 2 thiết bị android: Samsung Galaxy A50 (Điện thoại) và Samsung Tab S6lite (Tablet).

1. Giao diện khởi động

Giao diện này hiển thị khi lần đầu sử dụng ứng dụng, hiển thị chi tiết cách sử dụng ứng dụng trước khi bước vào giao diện đăng nhập.

Khởi động ứng dụng bằng cách nhấp vào biểu tượng đã được cài đặt phiên bản apk trên màn hình.

Các giao diện đều đã được responsive theo từng thiết bị.



Hình 1. Giao diện khởi động

1.1. Các widget sử dụng

Scaffold, PageView.builder, Image.asset, và AnimatedContainer (widget đặc biệt để tạo hiệu ứng cho chấm chỉ báo).

1.2. Thư viện/Plugin sử dụng

• **flutter/material.dart**: Thư viện cơ bản của Flutter, cung cấp các widget giao diện như Scaffold, PageView, Image.asset, AnimatedContainer, v.v.

• **shared_preferences**: Một plugin để lưu trữ dữ liệu cục bộ dưới dạng key-value đơn giản

1.3. Quản lý trạng thái

Không sử dụng giải pháp quản lý trạng thái chia sẻ.

1.4. Kiến trúc sơ lược

- Khi ứng dụng khởi động, MyApp kiểm tra trạng thái onboarding_completed (thường trong main.dart hoặc một màn hình khởi động khác, mặc dù đoạn code này không được hiển thị ở đây). Nếu onboarding_completed là false hoặc chưa được thiết lập, ứng dụng hiển thị OnboardingScreen.

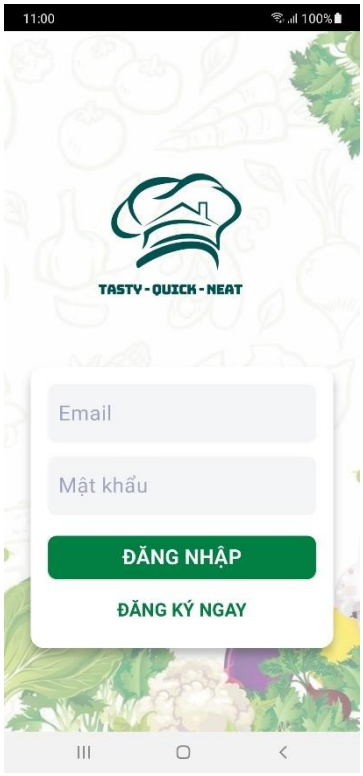
- Người dùng cuộn qua các trang bằng cách chạm (tap). Khi đến trang cuối cùng, ứng dụng gọi _completeOnboarding để lưu trạng thái và chuyển sang AuthScreen.

1.5. Lưu trữ dữ liệu

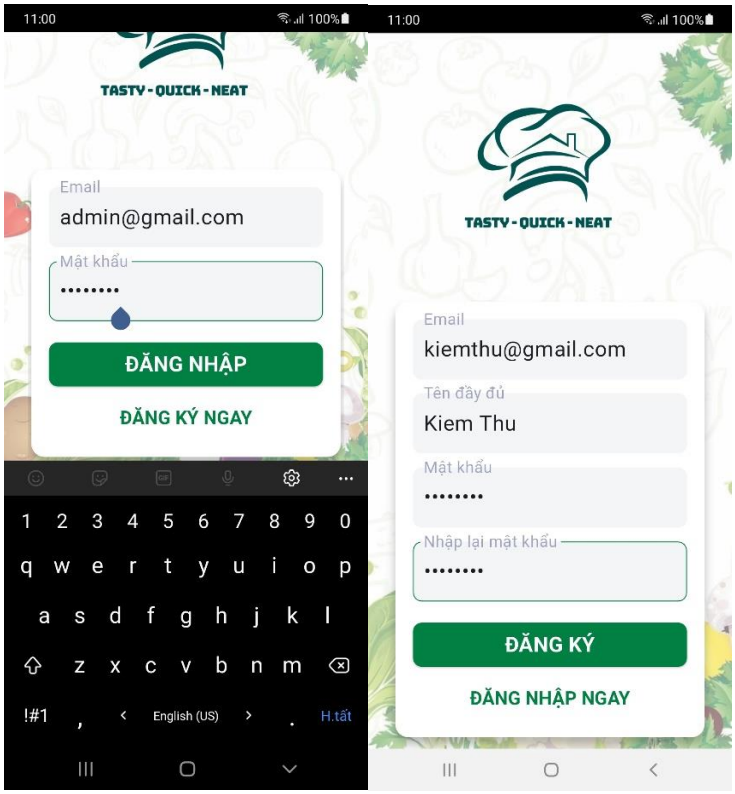
Dữ liệu được lưu trữ cục bộ trên thiết bị bằng shared_preferences.

2. Giao diện chứng thực

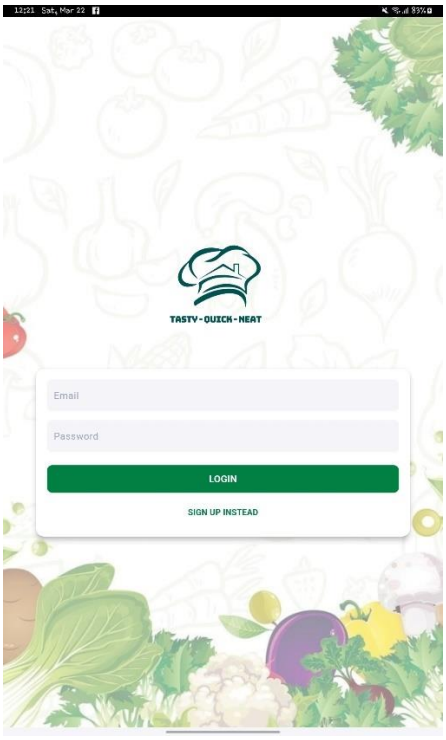
Cho phép người dùng có thể đăng nhập/ đăng ký trước khi bước vào sử dụng app.



Hình 2. Giao diện đăng nhập (di động)



Hình 3. Giao diện điền thông tin đăng ký (điện thoại)



Hình 4. Giao diện đăng nhập (tablet)

2.1. Các widget sử dụng

- **Trong AuthScreen:**

- Scaffold: Widget cơ bản để tạo cấu trúc giao diện với AppBar, body, v.v.
- Container: Để chứa nội dung và áp dụng DecorationImage làm hình nền.
- DecorationImage: Hiển thị hình nền (bg_login.jpg) với độ mờ (opacity: 0.5).
- Center: Căn giữa nội dung theo cả chiều ngang và dọc.
- SingleChildScrollView: Cho phép cuộn nội dung nếu nội dung vượt quá kích thước màn hình.
- Column: Sắp xếp các widget con theo chiều dọc (chứa AppBanner và AuthCard).
- SizedBox: Tạo khoảng cách giữa AppBanner và AuthCard.
- AppBanner: Một widget tùy chỉnh (không có trong thư viện Flutter mặc định) để hiển thị banner của ứng dụng.
- AuthCard: Một widget tùy chỉnh để hiển thị form đăng nhập/đăng ký.

- **Trong AuthCard:**

- Card: Tạo một thẻ với viền bo tròn và hiệu ứng nổi (elevation).
- Container: Để chứa nội dung của thẻ và áp dụng padding.
- Form: Widget để quản lý form và xử lý validation.
- SingleChildScrollView: Cho phép cuộn nội dung của form.
- Column: Sắp xếp các trường nhập liệu và nút theo chiều dọc.
- TextFormField: Các trường nhập liệu cho email, tên (nếu đăng ký), mật khẩu, và xác nhận mật khẩu.
- SizedBox: Tạo khoảng cách giữa các trường nhập liệu.
- ValueListenableBuilder: Widget đặc biệt để lắng nghe sự thay đổi của ValueNotifier (_isSubmitting) và cập nhật giao diện (hiển thị CircularProgressIndicator khi đang gửi dữ liệu).
- CircularProgressIndicator: Hiển thị vòng tròn tải khi đang xử lý đăng nhập/đăng ký.
- ElevatedButton: Nút để gửi form (đăng nhập hoặc đăng ký).
- TextButton: Nút để chuyển đổi giữa chế độ đăng nhập và đăng ký.
- Text: Hiển thị văn bản (nhãn, thông báo, v.v.).
- OutlineInputBorder: Tùy chỉnh viền của TextFormField.

- Widget đặc biệt: **ValueListenableBuilder**

2.2. Thư viện/ plugin sử dụng

Bảng 1. Thư viện/ plugin sử dụng trong widget đăng nhập

STT	Tên	Vai trò
1	provider	giúp quản lý trạng thái xác thực và chia sẻ AuthService giữa các widget.
2	pocketbase	cung cấp API để giao tiếp với backend, thực hiện các tác vụ như đăng nhập, đăng ký, và quản lý người dùng.
3	flutter/material.dart	cung cấp các widget giao diện cơ bản để xây dựng giao diện đăng nhập.

2.3. Quản lý chia sẻ trạng thái

- Sử dụng Provider trong chia sẻ trạng thái.

2.4. Kiến trúc code sơ lược

• **AuthScreen**: Là một StatelessWidget, chịu trách nhiệm hiển thị giao diện chính của màn hình đăng nhập/đăng ký. Nó sử dụng Scaffold để tạo cấu trúc giao diện, với hình nền và một Column chứa AppBanner và AuthCard.

• **AuthCard**: Là một StatefulWidget, quản lý trạng thái nội bộ của form (chế độ đăng nhập/đăng ký, dữ liệu nhập liệu, trạng thái gửi dữ liệu). Nó sử dụng Form để xử lý validation và các TextFormField để nhập dữ liệu.

• **AuthService**: Là một ChangeNotifier, quản lý trạng thái xác thực (người dùng hiện tại, vai trò admin) và giao tiếp với backend PocketBase để thực hiện các tác vụ như đăng nhập, đăng ký, đăng xuất, v.v.

2.5. Lưu trữ dữ liệu

• **Dịch vụ lưu trữ**: Sử dụng **PocketBase** làm backend, một cơ sở dữ liệu SQLite, hỗ trợ REST API để lưu trữ và truy xuất dữ liệu người dùng.

• **Không lưu trữ cục bộ**: Dữ liệu người dùng (email, mật khẩu, vai trò, v.v.) được lưu trữ trên server PocketBase, không lưu trữ cục bộ trên thiết bị.

Cấu trúc collection của users:

Bảng 2. Collection của users

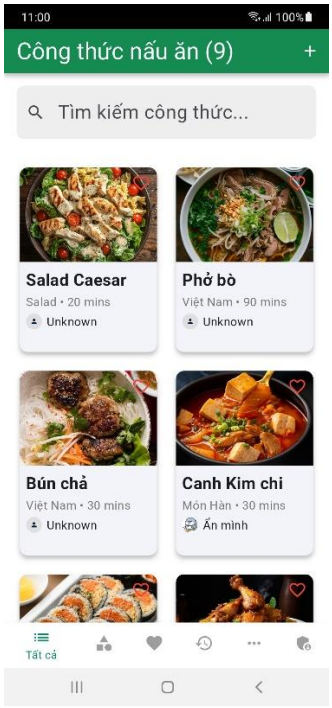
Thuộc tính	Kiểu dữ liệu	Mô tả
id	String	ID của người dùng (từ PocketBase).
name	String	Tên người dùng.
email	String	Email của người dùng.
avatarUrl	String?	URL ảnh đại diện (có thể null).
role	String	Vai trò của người dùng (mặc định là "user").
isActive	bool	Trạng thái kích hoạt tài khoản (mặc định true).
favorites	List<String>	Danh sách ID của các công thức yêu thích.

REST API:

- **Đăng ký người dùng:** POST /api/collections/users/records
- **Đăng nhập:** /api/collections/users/auth-with-password
- Lấy thông tin người dùng từ store (getUserFromStore)

3. Giao diện hiển thị danh sách các món

Sau khi đăng nhập vào sẽ hiển thị đầu tiên, hiển thị các món đã được phê duyệt theo dạng lưới, có thể dễ dàng quan sát.



Hình 5. Giao diện hiển thị tất cả các công thức (di động)

3.1. Widget sử dụng

• **Widget sử dụng**: Scaffold, AppBar, Consumer, Column, Padding, TextField, InputDecoration, Icon, Expanded, Center, Text, GridRecipes (tùy chỉnh), IconButton.

• **Widget đặc biệt**: Consumer (từ provider, lắng nghe thay đổi từ RecipeManager), GridRecipes (widget tùy chỉnh để hiển thị danh sách công thức dạng lưới).

3.2. Thư viện/plugin

• **flutter/material.dart**: Cung cấp widget giao diện (Scaffold, TextField, v.v.).

• **provider**: Quản lý trạng thái, cung cấp RecipeManager để cập nhật danh sách công thức.

• **pocketbase**: Giao tiếp với backend để lấy danh sách công thức và danh mục.

• **http**: Hỗ trợ gửi file (hình ảnh) khi thêm công thức/danh mục.

3.3. Quản lý trạng thái - Kiến trúc code

• **Giải pháp**: Provider (dùng RecipeManager để quản lý danh sách công thức và danh mục, thông báo thay đổi qua notifyListeners()).

• **Kiến trúc**:

- **RecipeService:** Giao tiếp với backend (PocketBase) để lấy/thêm/xóa công thức.
- **RecipeManager:** Quản lý trạng thái (danh sách công thức, danh mục), gọi RecipeService và thông báo thay đổi.
- **ListRecipesOverScreen:** Hiển thị giao diện, lắng nghe RecipeManager qua Consumer để cập nhật danh sách.

3.4. Lưu trữ dữ liệu

Tên collection: recipes.

Đọc/lưu trữ: Đọc và lưu trữ dữ liệu qua backend PocketBase.

Bảng 3. Collection recipes

Thuộc tính	Kiểu dữ liệu	Mô tả
id	String	ID của công thức (từ PocketBase).
title	String	Tên công thức.
description	String	Mô tả chi tiết về công thức.

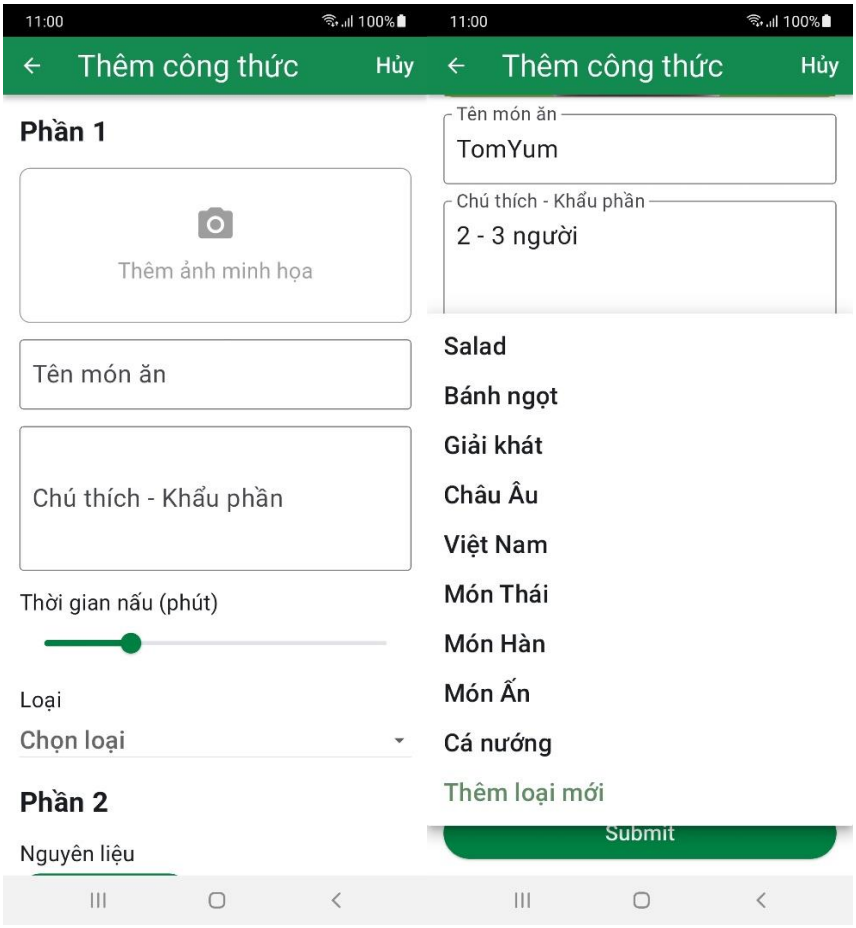
Thuộc tính	Kiểu dữ liệu	Mô tả
featuredImage	File?	Ảnh đại diện (file nội bộ, dùng khi upload).
imageUrl	String?	URL ảnh từ PocketBase.
ingredients	List<String>	Danh sách nguyên liệu.
steps	List<String>	Danh sách bước thực hiện.
category	String	ID danh mục công thức.
categoryName	String	Tên danh mục (ví dụ: "Món chính").
duration	int	Thời gian nấu (phút).
userId	String	ID của người tạo công thức.
userName	String	Tên người tạo công thức.
userAvatarUrl	String	URL ảnh đại diện của người tạo.
likes	int	Số lượt thích.
status	String	Trạng thái: "pending", "approved", "rejected".
isFavorite	bool	Đánh dấu yêu thích.

API (PocketBase):

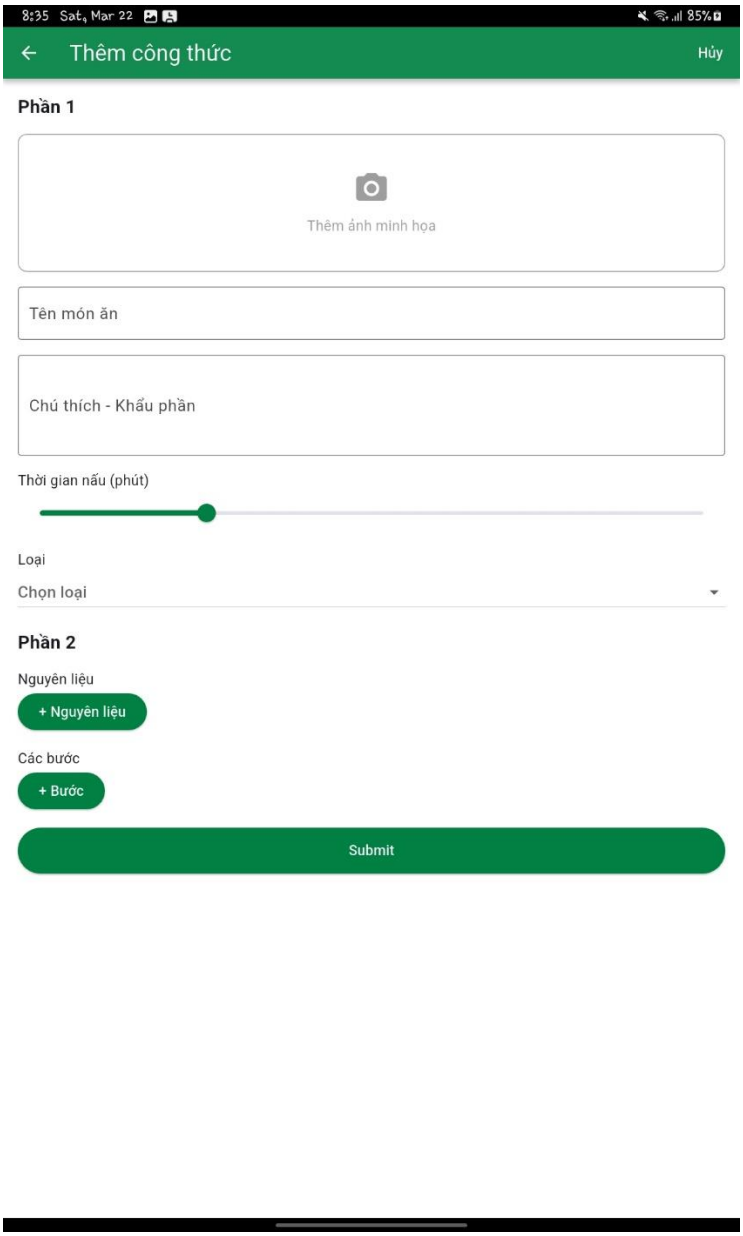
- **GET** /api/collections/recipes/records: Lấy danh sách công thức (đầu vào: filter, expand; đầu ra: danh sách JSON công thức).
- **GET** /api/collections/categories/records: Lấy danh sách danh mục (đầu ra: danh sách JSON danh mục).

4. Chức năng thêm mới công thức

Chức năng này cho phép người dùng thêm mới một công thức nấu ăn do chính mình sáng tác, nhưng phải chờ đợi phê duyệt từ admin rồi mới được hiển thị.



Hình 6. Thêm mới công thức nấu ăn (di động)



Hình 7. Thêm mới công thức (tablet)

4.1. Các widget sử dụng

- **Widget sử dụng**: Scaffold, AppBar, FutureBuilder, SingleChildScrollView, Column, GestureDetector, Container, Image.file, TextField, Slider, Consumer, DropdownButton, ListTile, IconButton, ElevatedButton, AlertDialog, TextButton, Center, CircularProgressIndicator, SnackBar.

- **Widget đặc biệt**: Consumer (từ provider, lắng nghe RecipeManager), Slider (điều chỉnh thời gian nấu), ImagePicker (chọn ảnh từ thư viện).

4.2. Các thư viện/plugin

- **flutter/material.dart**: Cung cấp widget giao diện (Scaffold, TextField, v.v.).
- **provider**: Quản lý trạng thái, cung cấp RecipeManager và AuthService.
- **image_picker**: Chọn ảnh từ thư viện để làm ảnh bìa công thức.
- **dart:io**: Xử lý file ảnh (File).

4.3. Giải pháp quản lý trạng thái – Kiến trúc code

- **Giải pháp**: Provider (dùng RecipeManager để quản lý danh mục và thêm công thức, AuthService để kiểm tra đăng nhập).
- **Kiến trúc**:

- **AuthService:** Kiểm tra trạng thái đăng nhập.
- **RecipeManager:** Quản lý danh mục và thêm công thức, gọi backend qua **RecipeService.**
- **AddRecipeScreen:** Hiển thị form nhập liệu, gọi RecipeManager để thêm công thức

4.4. Lưu trữ dữ liệu:

Có lưu trữ dữ liệu và sử dụng collection tương tự với chức năng hiển thị danh sách món.

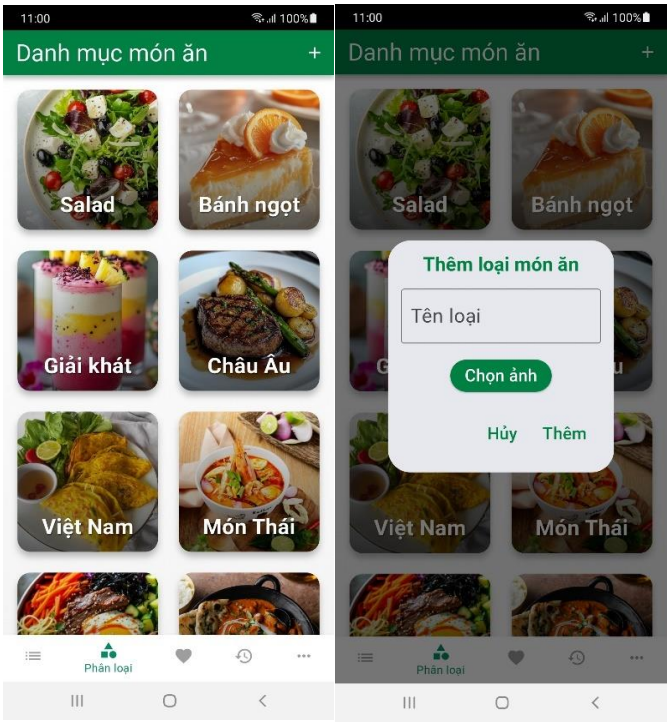
API (PocketBase):

- **POST** /api/collections/recipes/records: Thêm công thức (đầu vào: JSON công thức + file ảnh; đầu ra: JSON công thức mới).

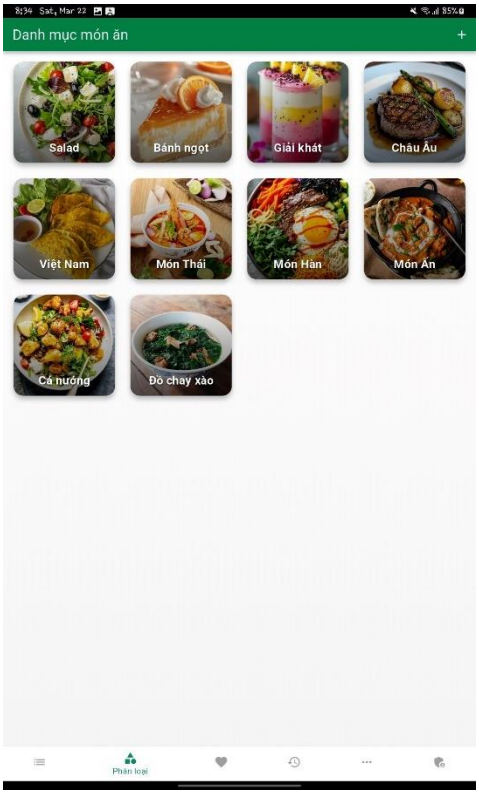
- **GET** /api/collections/categories/records: Lấy danh sách danh mục (đầu ra: danh sách JSON danh mục).

5. Hiển thị danh mục món và thêm mới

Giao diện này cho phép hiển thị các món đã được phân theo loại sẵn, có thể dễ dàng lựa chọn tùy sở thích và thêm một loại mới.



Hình 8. Hiển thị danh mục món và thêm mới



Hình 9. Danh mục món (Tablet)

5.1. Các widget sử dụng

• **Widget sử dụng**: Scaffold, AppBar, Consumer, Container, GridView.builder, GestureDetector, Card, ClipRRect, Image.network, Stack, Positioned, Text, IconButton, AlertDialog, TextField, ElevatedButton, TextButton, Center, CircularProgressIndicator, SnackBar.

• **Widget đặc biệt**: Consumer (từ provider, lắng nghe CategoryManager), GridView.builder (hiển thị danh sách dạng lưới), ImagePicker (chọn ảnh từ thư viện).

5.2. Thư viện/ plugin

• **flutter/material.dart**: Cung cấp widget giao diện (Scaffold, GridView, v.v.).

• **provider**: Quản lý trạng thái, cung cấp CategoryManager.

• **image_picker**: Chọn ảnh từ thư viện để làm ảnh đại diện danh mục.

• **dart:io**: Xử lý file ảnh (File).

5.3. Giải pháp xử lý trạng thái – Kiến trúc code

• **Giải pháp**: Provider (dùng CategoryManager để quản lý danh mục, thông báo thay đổi qua notifyListeners()).

• **Kiến trúc**:

- CategoryService: Giao tiếp với backend (PocketBase) để lấy/thêm danh mục.
- CategoryManager: Quản lý danh sách danh mục, gọi CategoryService.
- CategoryScreen: Hiển thị danh sách danh mục, gọi CategoryManager để thêm danh mục.

5.4. Lưu trữ dữ liệu

Lưu trữ dữ liệu qua backend PocketBase.

Collection sử dụng category:

Bảng 4. Collection categories

Thuộc tính	Kiểu dữ liệu	Mô tả
id	String	ID của danh mục (từ PocketBase).
name	String	Tên danh mục.
featuredImage	File?	Ảnh đại diện (nếu ảnh được chọn từ thiết bị).

Thuộc tính	Kiểu dữ liệu	Mô tả
imageUrl	String	URL ảnh từ PocketBase.

API (PocketBase):

- **GET** /api/collections/categories/records: Lấy danh sách danh mục (đầu ra: danh sách JSON danh mục).
- **POST** /api/collections/categories/records: Thêm danh mục (đầu vào: JSON danh mục + file ảnh; đầu ra: JSON danh mục mới).

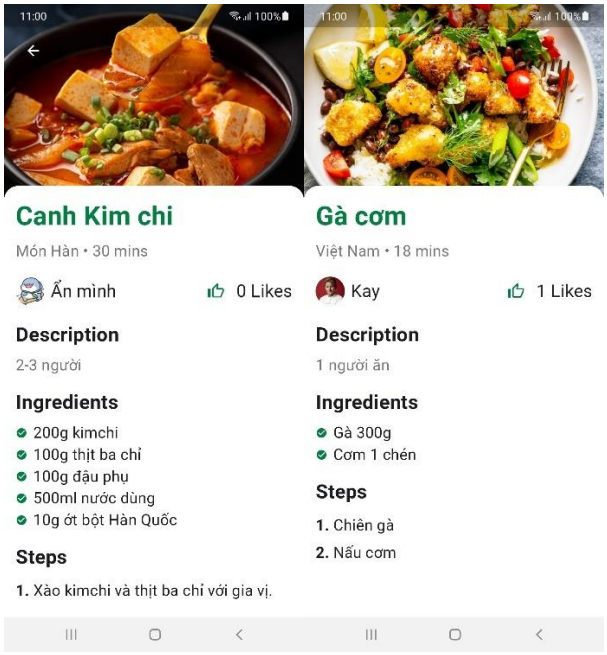
6. Giao diện hiển thị chi tiết công thức

Hiển thị chi tiết cách nấu, quy trình nấu và người nấu món đó.

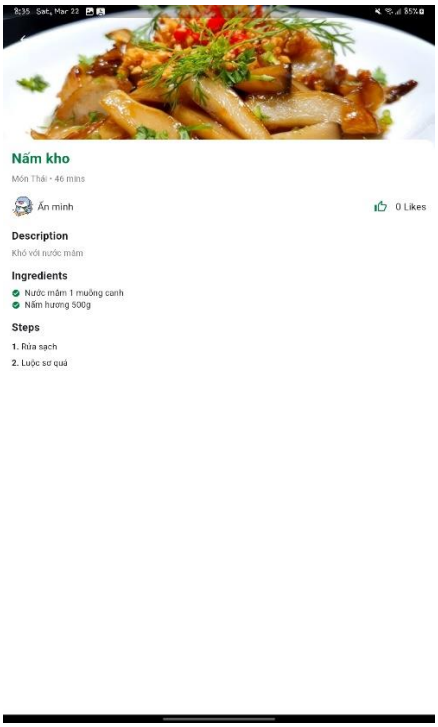
6.1. Các widget sử dụng

• **Widget sử dụng**: Scaffold, Stack, Positioned, Image.network, IconButton, Container, SingleChildScrollView, Column, Text, Row, CircleAvatar, Consumer, Icon, Expanded, Padding, SnackBar.

• **Widget đặc biệt**: Consumer (từ provider, lắng nghe RecipeManager để cập nhật trạng thái thích).



Hình 10. Chi tiết công thức nấu ăn



Hình 11. Giao diện chi tiết (tablet)

Hình 12. Giao diện sau khi đã thêm mới thành công

6.2. Thư viện/ plugin

• **flutter/material.dart**: Cung cấp widget giao diện (Scaffold, Image.network, v.v.).

• **provider**: Quản lý trạng thái, cung cấp RecipeManager để xử lý thích công thức và lưu công thức đã xem.

6.3. Giải pháp quản lý trạng thái - Kiến trúc code

• **Giải pháp**: Provider (dùng RecipeManager để quản lý trạng thái thích và danh sách đã xem).

• **Kiến trúc**:

- **RecipeManager:** Quản lý trạng thái (thích công thức, danh sách đã xem), gọi backend qua RecipeService.
- **RecipeDetailScreen:** Hiển thị chi tiết công thức, gọi RecipeManager để xử lý thích và lưu công thức đã xem.

6.4. Lưu trữ dữ liệu

Lưu trữ dữ liệu qua backend PocketBase, sử dụng collection: recipes, category đã mô tả.

API (PocketBase):

PUT /api/collections/recipes/records/{id}: Cập nhật số lượt thích (đầu vào: JSON với likes và likedBy; đầu ra: JSON công thức đã cập nhật).

7. Giao diện hiển thị món yêu thích

Giao diện này hiển thị các món đã được nhấn yêu thích (theo từng người dùng).

7.1. Các widget sử dụng

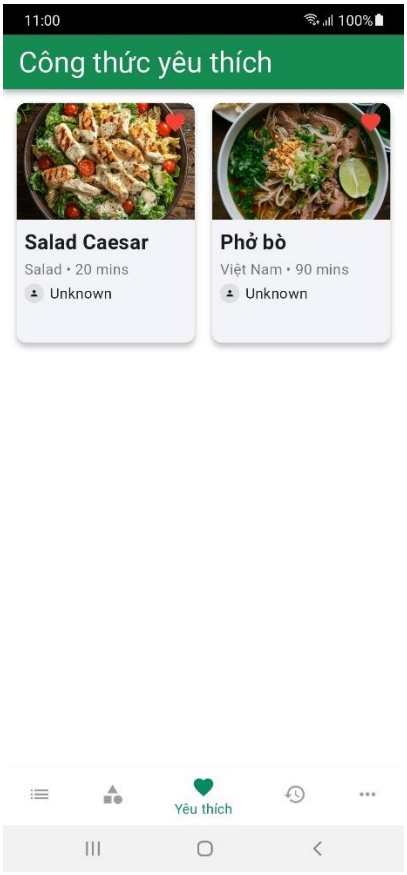
• **Widget sử dụng**: Scaffold, AppBar, FutureBuilder, GridView.builder, Center, CircularProgressIndicator, Text, RecipeItem (tùy chỉnh).

• **Widget đặc biệt**: FutureBuilder (xử lý dữ liệu bất đồng bộ từ fetchFavoriteRecipes), RecipeItem (widget tùy chỉnh hiển thị công thức).

7.2 Thư viện và plugin

• **flutter/material.dart**: Cung cấp widget giao diện (Scaffold, GridView, v.v.).

• **provider**: Quản lý trạng thái, cung cấp RecipeManager để lấy danh sách công thức yêu thích.



Hình 13. Hiển thị yêu thích

7.3. Giải pháp quản lý trạng thái – Kiến trúc code

• **Giải pháp**: Provider (dùng RecipeManager để quản lý danh sách công thức yêu thích).

• **Kiến trúc**:

- **RecipeManager:** Quản lý danh sách công thức yêu thích, gọi RecipeService để lấy dữ liệu.
- **FavoriteScreen:** Hiển thị danh sách công thức yêu thích, gọi RecipeManager để lấy dữ liệu.

Điều hướng: Từ MainScreen, tab "Yêu thích" (index 2) dẫn đến FavoriteScreen.

7.4. Lưu trữ dữ liệu

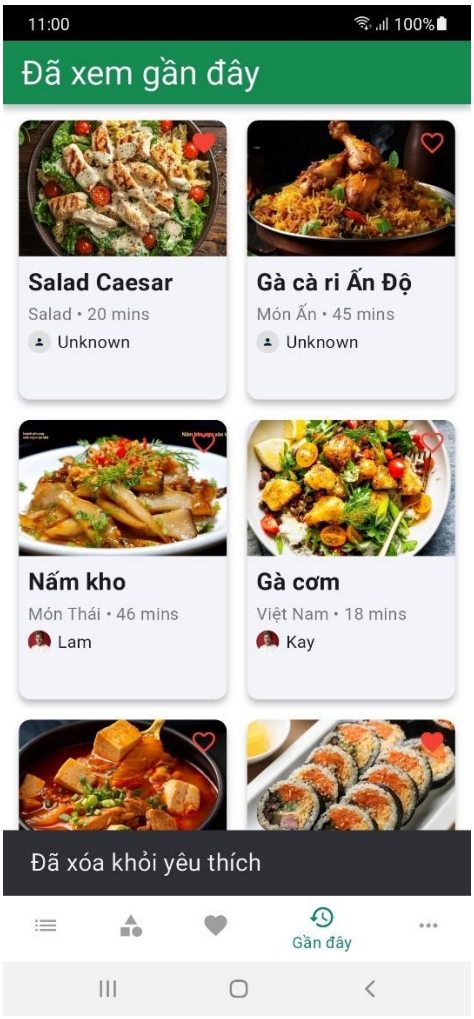
Sử dụng Backend Pocketbase, collection recipes.

API (PocketBase):

GET /api/collections/recipes/records: Lấy danh sách công thức (lọc isFavorite = true trong RecipeManager).

8. Giao diện các món đã xem gần đây

Tại đây có thể xem lại các món gần đây đã xem ở chế độ chi tiết, giúp người dùng có thể dễ dàng xem lại và cũng có thể cho vào danh sách yêu thích.



8.1. Widget sử dụng

- **Widget sử dụng**: Scaffold, AppBar, FutureBuilder, GridView.builder, Center, CircularProgressIndicator, Text, RecipeItem (tùy chỉnh).

- **Widget đặc biệt**: FutureBuilder (xử lý dữ liệu bất đồng bộ từ fetchRecentlyViewedRecipes), RecipeItem (widget tùy chỉnh hiển thị công thức).

Bảng mô tả widget:

Bảng 5. Bảng mô tả widget hiển thị công thức đã xem gần đây

Thành phần	Mô tả	Widget chính
Thanh tiêu đề (AppBar)	Hiển thị tiêu đề "Đã xem gần đây".	AppBar, Text
Danh sách đã xem	Hiển thị danh sách công thức đã xem dạng lưới.	GridView.builder, RecipeItem
Trạng thái tải	Hiển thị vòng tròn tải khi đang lấy dữ liệu.	CircularProgressIndicator
Thông báo rỗng	Hiển thị "Chưa có công thức nào được xem gần đây" nếu danh sách rỗng.	Center, Text

8.2. Thư viện/ plugin

• **flutter/material.dart**: Cung cấp widget giao diện (Scaffold, GridView, v.v.).

• **provider**: Quản lý trạng thái, cung cấp RecipeManager để lấy danh sách công thức đã xem.

8.3. Quản lý trạng thái – Kiến trúc code

• **Giải pháp**: Provider (dùng RecipeManager để quản lý danh sách công thức đã xem).

• **Kiến trúc**:

- **RecipeManager:** Quản lý danh sách công thức đã xem (lưu trữ recentlyViewedIds trong bộ nhớ), gọi RecipeService để lấy dữ liệu.
- **RecentlyViewedScreen:** Hiển thị danh sách công thức đã xem, gọi
- **RecipeManager** để lấy dữ liệu.

Điều hướng: Từ MainScreen, tab "Gần đây" (index 3) dẫn đến RecentlyViewedScreen.

8.4. Lưu trữ dữ liệu

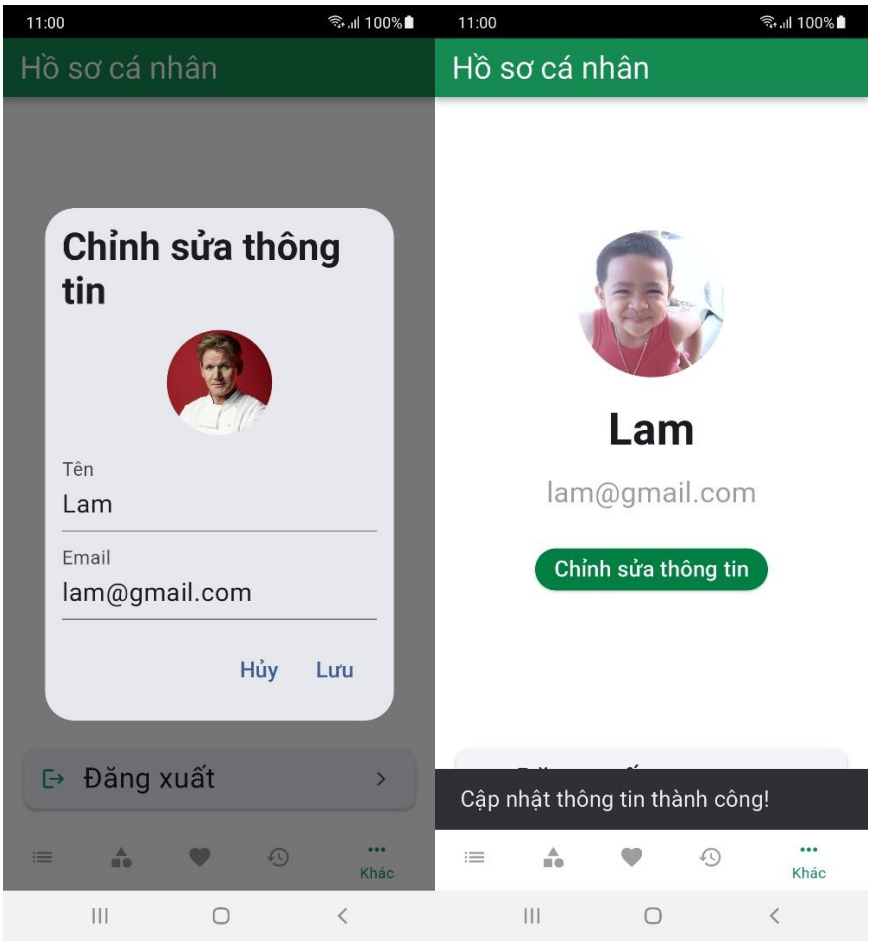
Lưu trữ qua pocketbase, sử dụng collection: recipe đã mô tả ở ***Bảng 3***.

API (PocketBase):

GET /api/collections/recipes/records: Lấy danh sách công thức (lọc theo recentlyViewedIds trong RecipeManager).

9. Giao diện Profile – chỉnh sửa thông tin

Giao diện này cho phép xem lại và điều chỉnh thông tin, cập nhật ảnh đại diện.

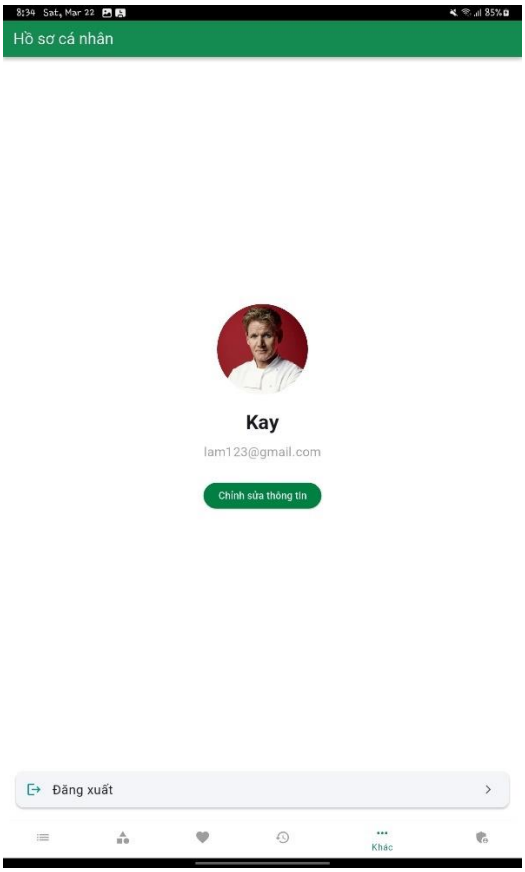


Hình 14. Trang hồ sơ cá nhân (di động)

9.1. Widget sử dụng

• **Widget sử dụng**: Scaffold, AppBar, Consumer, SingleChildScrollView, Column, GestureDetector, CircleAvatar, Text, ElevatedButton, AlertDialog, TextField, TextButton, Center, CircularProgressIndicator, SnackBar, Card, ListTile, Icon.

• **Widget đặc biệt**: Consumer (từ provider, lắng nghe ProfileManager), ImagePicker (chọn ảnh từ thư viện).



Hình 15. Giao diện hồ sơ (tablet)

Bảng mô tả widget:

Bảng 6. Bảng mô tả widget chỉnh sửa thông tin hồ sơ

Thành phần	Mô tả	Widget chính
Thanh tiêu đề (AppBar)	Hiển thị tiêu đề "Hồ sơ cá nhân".	AppBar, Text
Ảnh đại diện	Hiển thị ảnh đại diện, có thể chọn ảnh mới.	CircleAvatar, GestureDetector
Thông tin cá nhân	Hiển thị tên và email người dùng.	Text
Nút chỉnh sửa	Mở dialog để chỉnh sửa thông tin (tên, email, ảnh).	ElevatedButton, AlertDialog
Nút đăng xuất	Nút đăng xuất với xác nhận qua dialog.	Card, ListTile, AlertDialog

9.2. Thư viện/ plugin

- **flutter/material.dart**: Cung cấp widget giao diện (Scaffold, CircleAvatar, v.v.).
- **provider**: Quản lý trạng thái, cung cấp ProfileManager và AuthService.
- **image_picker**: Chọn ảnh từ thư viện để cập nhật ảnh đại diện.
- **dart:io**: Xử lý file ảnh (File).

9.3. Quản lý trạng thái – Kiến trúc code:

- **Giải pháp**: Provider (dùng ProfileManager để quản lý thông tin hồ sơ, AuthService để xử lý đăng xuất).
- **Kiến trúc**:

- **ProfileService:** Giao tiếp với backend (PocketBase) để lấy/cập nhật thông tin hồ sơ.
- **ProfileManager:** Quản lý trạng thái hồ sơ, gọi ProfileService.
- **ProfileScreen:** Hiển thị thông tin hồ sơ, gọi ProfileManager để cập nhật.
- **OtherScreen:** Bao gồm ProfileScreen và nút đăng xuất, điều hướng từ MainScreen (tab "Khác").

9.4. Lưu trữ dữ liệu

- Lưu trữ qua Pocketbase, collection: users đã được mô tả.

- Các API:

API (PocketBase):

- **GET** /api/collections/users/records/{id}: Lấy thông tin hồ sơ (đầu ra: JSON người dùng).
- **PUT** /api/collections/users/records/{id}: Cập nhật thông tin (đầu vào: JSON với name, email; đầu ra: JSON người dùng đã cập nhật).
- **POST** /api/collections/users/records/{id}/files: Tải ảnh đại diện (đầu vào: file ảnh; đầu ra: URL ảnh).

10. Giao diện quản lý

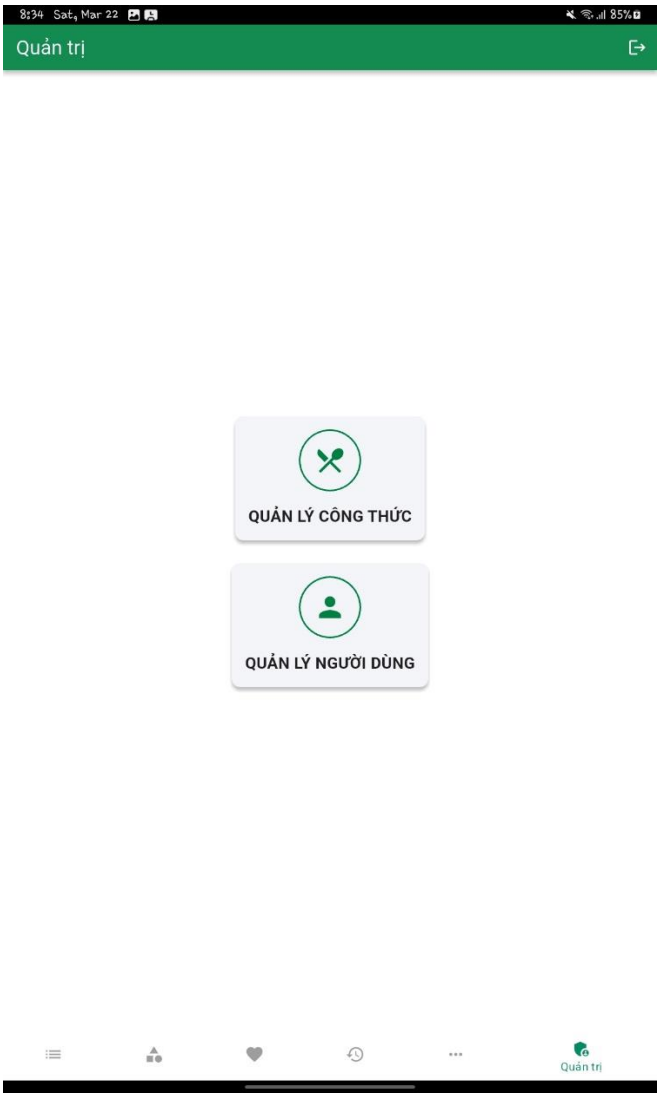
Giao diện này khi đã đăng nhập vào tài khoản quản trị viên, xuất hiện thêm icon ở Main Screen để quản trị có thể quan sát qua lại các quá trình điều khiển.

Hình 16. Giao diện quản lý (di động)

10.1. Widget sử dụng

- **Widget sử dụng**: Scaffold, AppBar, IconButton, Center, Column, InkWell, Card, Padding, Container, Icon, Text, AlertDialog, TextButton, CircularProgressIndicator, SnackBar.

- **Widget đặc biệt**: Không có widget đặc biệt, chủ yếu sử dụng các widget cơ bản của Flutter.



Hình 17. Giao diện quản lý (Tablet)

Bảng mô tả:

Bảng 7. Bảng mô tả widget của giao diện quản lý của Admin

Thành phần	Mô tả	Widget chính
Thanh tiêu đề (AppBar)	Hiển thị tiêu đề "Admin Dashboard" và nút đăng xuất.	AppBar, IconButton
Card quản lý công thức	Hiển thị tùy chọn "Recipe Management", điều hướng đến màn hình quản lý công thức.	InkWell, Card, Icon
Card quản lý người dùng	Hiển thị tùy chọn "User Management", điều hướng đến màn hình quản lý người	InkWell, Card, Icon

Thành phần	Mô tả	Widget chính
	dùng.	
Xác nhận đăng xuất	Dialog xác nhận đăng xuất với nút hủy và đăng xuất.	AlertDialog, TextButton

10.2. Thư viện/ plugin

• **flutter/material.dart**: Cung cấp widget giao diện (Scaffold, Card, v.v.).

• **provider**: Quản lý trạng thái, cung cấp AuthService để xử lý đăng xuất.

10.3. Quản lý trạng thái – Kiến trúc code

• **Giải pháp**: Provider (dùng AuthService để xử lý đăng xuất).

• **Kiến trúc**:

- **AuthService:** Quản lý trạng thái đăng nhập/đăng xuất.
- **AdminDashboard:** Hiển thị giao diện quản trị, điều hướng đến các màn hình quản lý công thức và người dùng.

Điều hướng: Từ MainScreen, tuy nhiên cần yêu cầu vào vai trò admin.

10.4. Lưu trữ dữ liệu

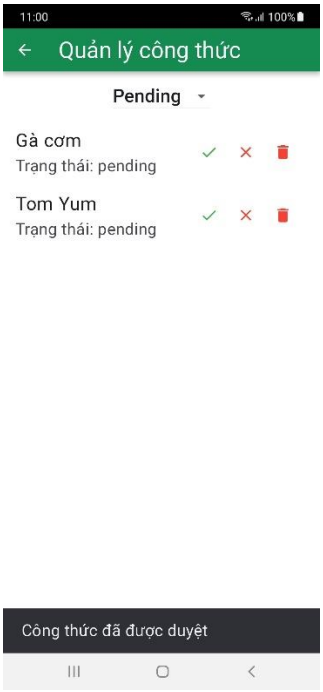
- **Đọc/lưu trữ**: Không trực tiếp đọc/lưu trữ dữ liệu trong màn hình này (chỉ xử lý đăng xuất).
- **Cấu trúc dữ liệu**: Không áp dụng (màn hình này không hiển thị dữ liệu từ backend).

• **API (PocketBase)**:

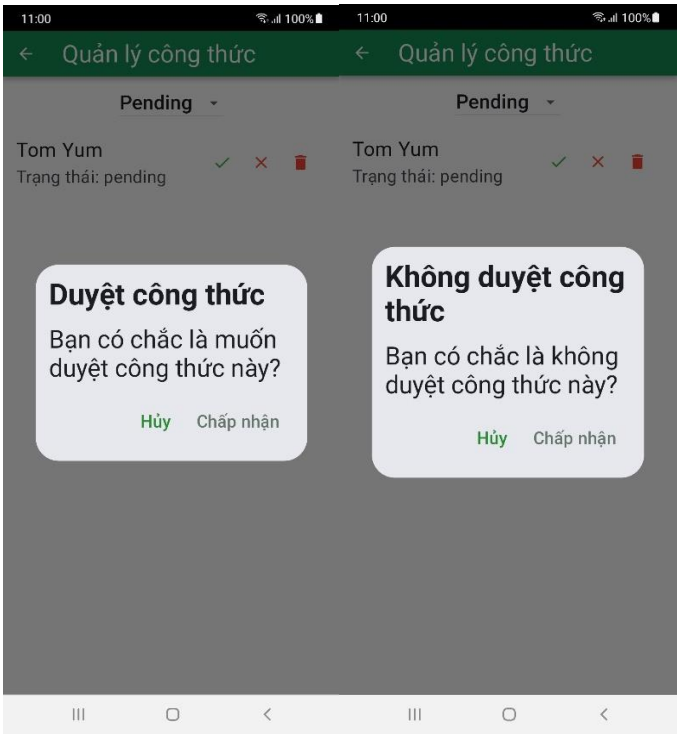
POST /api/auth/logout: Đăng xuất (được gọi từ AuthService.logout()).

11. Giao diện quản lý công thức được yêu cầu thêm.

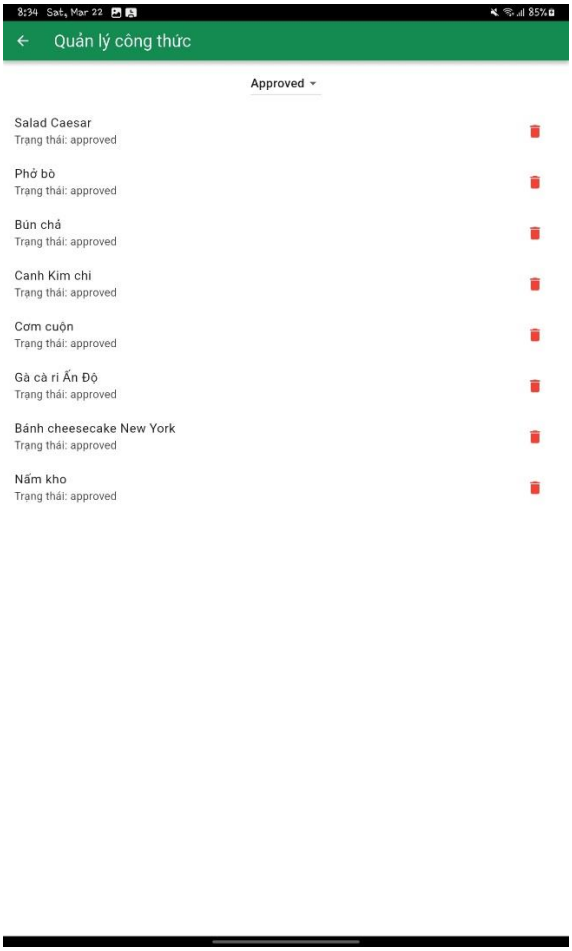
Cho phép admin quản lý lại tất cả công thức được yêu cầu xuất hiện lên hệ thống.



Hình 18. Quản lý công thức



Hình 19. Phê duyệt công thức



Hình 20. Giao diện quản lý (tablet)

11.1. Widget sử dụng

- **Widget sử dụng:** Scaffold, AppBar, Column, DropdownButton, Expanded, FutureBuilder, Consumer, ListView.builder, ListTile, IconButton, AlertDialog, TextButton, Center, CircularProgressIndicator, SnackBar.
- **Widget đặc biệt:** FutureBuilder (xử lý dữ liệu bất đồng bộ), Consumer (từ provider, lắng nghe RecipeManager).

Bảng mô tả:

Bảng 8. Bảng mô tả widget của giao diện quản lý công thức được yêu cầu thêm

Thành phần	Mô tả	Widget chính
Thanh tiêu đề (AppBar)	Hiển thị tiêu đề "Recipe Management".	AppBar, Text
Lọc trạng thái	Dropdown để lọc công thức	DropdownButton

Thành phần	Mô tả	Widget chính
	theo trạng thái (pending, approved, rejected).	
Danh sách công thức	Hiển thị danh sách công thức với tiêu đề, trạng thái, và nút hành động.	ListView.builder, ListTile
Nút hành động	Nút phê duyệt, từ chối (cho trạng thái pending), và xóa công thức.	IconButton
Xác nhận hành động	Dialog xác nhận hành động (phê duyệt, từ chối, xóa).	AlertDialog

11.2. Thư viện/ plugin

- **flutter/material.dart**: Cung cấp widget giao diện (Scaffold, ListView, v.v.).
- **provider**: Quản lý trạng thái, cung cấp RecipeManager để lấy và cập nhật công thức.

11.3. Quản lý trạng thái – Kiến trúc code:

• **Giải pháp**: Provider (dùng RecipeManager để quản lý danh sách công thức).

• **Kiến trúc**:

RecipeManager: Quản lý danh sách công thức, gọi RecipeService để lấy/cập nhật/xóa công thức.

RecipeManagementScreen: Hiển thị danh sách công thức, gọi RecipeManager để xử lý dữ liệu.

Điều hướng: Từ AdminDashboard dẫn đến RecipeManagementScreen

11.4. Lưu trữ dữ liệu

Lưu trữ qua Pocketbase.

Collections: recipes.

API:

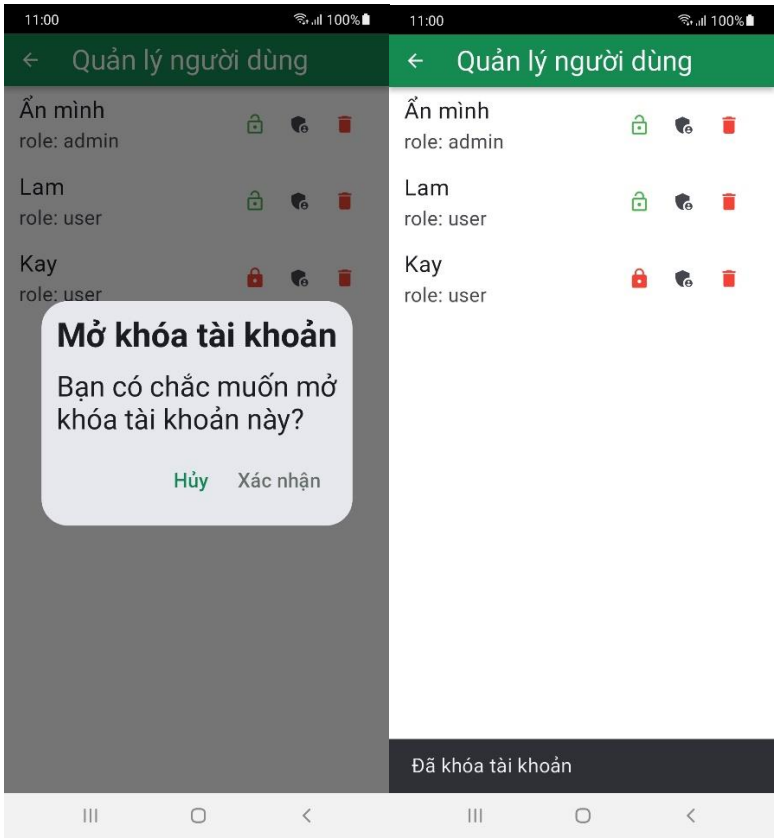
• **GET** /api/collections/recipes/records?filter=status="pending": Lấy danh sách công thức (đầu ra: danh sách JSON công thức).

• **PUT** /api/collections/recipes/records/{id}: Cập nhật trạng thái công thức (đầu vào: JSON với status).

• **DELETE** /api/collections/recipes/records/{id}: Xóa công thức.

12. Giao diện quản lý người dùng

Giao diện này cho phép tài khoản có quyền admin khóa tài khoản, xóa tài khoản user và cấp thêm vai trò mới.



Hình 21. Giao diện quản lý người dùng

12.1. Widget sử dụng

• **Widget sử dụng**: Scaffold, AppBar, FutureBuilder, ListView.builder, ListTile, IconButton, AlertDialog, TextButton, Center, CircularProgressIndicator, SnackBar.

• **Widget đặc biệt**: FutureBuilder (xử lý dữ liệu bất đồng bộ từ fetchUsers).

Bảng mô tả:

Bảng 9. Bảng mô tả widget của giao diện quản lý người dùng

Thành phần	Mô tả	Widget chính
Thanh tiêu đề (AppBar)	Hiển thị tiêu đề "User Management".	AppBar, Text
Danh sách người dùng	Hiển thị danh sách người dùng với tên, email, vai trò, trạng thái.	ListView.builder, ListTile
Nút hành động	Nút khóa/mở khóa, thay đổi vai trò, và xóa người dùng.	IconButton
Xác nhận hành động	Dialog xác nhận hành động (khóa, mở khóa, thay đổi vai trò, xóa).	AlertDialog, TextButton
Thông báo rỗng	Hiển thị "Không có người dùng nào" nếu danh sách rỗng.	Center, Text

12.2. Plugin/ thư viện

- **flutter/material.dart**: Cung cấp widget giao diện (Scaffold, ListView, v.v.).
- **provider**: Quản lý trạng thái, cung cấp AuthService để lấy và quản lý người dùng.

12.3. Quản lý trạng thái/ chia sẻ:

- **Giải pháp**: Provider (dùng AuthService để quản lý danh sách người dùng).
- **Kiến trúc**:

AuthService: Quản lý danh sách người dùng, gọi backend để lấy/cập nhật/xóa người dùng.

UserManagementScreen: Hiển thị danh sách người dùng, gọi AuthService để xử lý dữ liệu.

Điều hướng: Từ AdminDashboard dẫn đến UserManagementScreen.

12.4. Lưu trữ dữ liệu

Lưu trữ qua Pocketbase

API:

- **GET** /api/collections/users/records: Lấy danh sách người dùng (đầu ra: danh sách JSON người dùng).
- **PUT** /api/collections/users/records/{id}: Cập nhật trạng thái hoặc vai trò (đầu vào: JSON với isActive hoặc role).
- **DELETE** /api/collections/users/records/{id}: Xóa người dùng.

Hình 22. Quản lý người dùng (tablet)

PHỤ LỤC

DEMO DỰ ÁN

Đồ án đã được build apk và được thực hiện trên thiết bị android. Link youtube

https://youtube.com/watch?v=azW1M0HgQFQ&si=sNJnbv5uLO_nFYYJ