

**BỘ CÔNG THƯƠNG**  
**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**

-----\*\*\*-----



**ĐỒ ÁN TỐT NGHIỆP**  
**NGÀNH KHOA HỌC MÁY TÍNH**

**ĐỀ TÀI: XÂY DỰNG ỨNG DỤNG DI ĐỘNG CHĂM SÓC  
SỨC KHỎE SỬ DỤNG FIREBASE VÀ FLUTTER**

**Giảng viên hướng dẫn : ThS. Trần Thanh Huân**

**Sinh viên thực hiện : Nguyễn Trọng Trường**

**Mã sinh viên : 2021600107**

**Hà Nội – 2025**

## MỤC LỤC

LỜI CẢM ƠN.....	iii
DANH MỤC CÁC THUẬT NGỮ, KÝ HIỆU VÀ CÁC CHỮ VIẾT TẮT....	iv
DANH MỤC HÌNH ẢNH.....	v
DANH MỤC BẢNG BIỂU.....	vii
LỜI MỞ ĐẦU.....	1
CHƯƠNG 1:TỔNG QUAN VỀ DỰ ÁN PHẦN MỀM.....	4
1.1. Cơ sở lý thuyết.....	4
1.1.1. Phát triển ứng dụng trên thiết bị di động.....	4
1.1.2. Phân tích thiết kế hệ thống.....	5
1.2. Ngôn ngữ mô hình hóa UML.....	6
1.3. Các công nghệ sử dụng trong đề tài.....	7
1.3.1. Dart.....	7
1.3.2. Flutter.....	9
1.4. Mô hình Model – View – ViewModel.....	10
1.5. Hệ quản trị cơ sở dữ liệu.....	12
1.5.1. Các khái niệm.....	12
1.5.2. Firebase.....	13
CHƯƠNG 2:PHÂN TÍCH THIẾT KẾ HỆ THỐNG.....	16
2.1. Khảo sát hệ thống.....	16
2.2. Mô tả bài toán.....	16
2.3. Xác định yêu cầu của hệ thống.....	17
2.4. Thiết kế hệ thống.....	18
2.4.1. Các tác nhân của hệ thống.....	18
2.4.2. Các use case trong hệ thống.....	18
2.4.3. Biểu đồ lớp.....	21
2.4.4. Biểu đồ hoạt động.....	21
2.4.5. Biểu đồ trình tự.....	29
2.4.6 Các yêu cầu về dữ liệu.....	33

2.6.7. Biểu đồ ERD.....	36
CHƯƠNG 3:CÀI ĐẶT VÀ KIỂM THỬ CHƯƠNG TRÌNH.....	37
3.1 Cài đặt môi trường.....	37
3.1.1 Các bước cài đặt chương trình.....	37
3.1.2 Lưu ý khi cài đặt chương trình.....	38
3.1.3 Giao diện hệ thống.....	39
3.2 Kiểm thử chương trình.....	69
3.3.1 Khái niệm.....	69
3.2.3. Các lĩnh vực kiểm thử hệ thống.....	70
3.2.4. Hoàn thành quá trình kiểm thử.....	70
3.3. Thiết kế các test case cho hệ thống.....	71
KẾT LUẬN .....	76
TÀI LIỆU THAM KHẢO.....	77

## **LỜI CẢM ƠN**

Lời đầu tiên cho phép em gửi lời cảm ơn sâu sắc tới các thầy cô trong Trường Công nghệ Thông tin và Truyền thông - Trường Đại học Công Nghiệp Hà Nội, những người đã hết mình truyền đạt và chỉ dẫn cho em những kiến thức, những bài học quý báu và bổ ích. Đặc biệt em xin được bày tỏ sự tri ân và xin chân thành cảm ơn giảng viên Thạc sĩ Trần Thanh Huân, người trực tiếp hướng dẫn, chỉ bảo em trong suốt quá trình học tập, nghiên cứu và hoàn thành được đồ án tốt nghiệp. Sau nữa, em xin gửi tình cảm sâu sắc tới gia đình và bạn bè vì đã luôn bên cạnh khuyến khích, động viên, giúp đỡ cả về vật chất lẫn tinh thần cho em trong suốt quy trình học tập để em hoàn thành tốt việc học tập của bản thân.

Trong quá trình nghiên cứu và làm đề tài, do năng lực, kiến thức, trình độ bản thân em còn hạn hẹp nên không tránh khỏi những thiếu sót và em mong mọi nhận được sự thông cảm và những góp ý từ quý thầy cô cũng như các bạn trong lớp.

Em xin trân trọng cảm ơn!

**Sinh viên thực hiện**  
Nguyễn Trọng Trường

## DANH MỤC CÁC THUẬT NGỮ, KÝ HIỆU VÀ CÁC CHỮ VIẾT TẮT

Ký hiệu	Giải thích
API	Giao diện lập trình ứng dụng
IDE	Môi trường phát triển tích hợp
Collection	Bộ sưu tập (trong Firestore – nơi chứa các tài liệu)
Document	Tài liệu (một bản ghi cụ thể trong Firestore)
UI	Giao diện người dùng (User Interface)
MVVM	Kiến trúc Model – View – ViewModel
OTP	One-Time Password – mã xác thực dùng một lần
Repository	Lớp xử lý truy xuất dữ liệu trong mô hình MVVM
ViewModel	Lớp trung gian giữa giao diện (View) và dữ liệu (Model)
LiveData	Thành phần trong Android Jetpack để theo dõi dữ liệu theo thời gian thực
Firebase	Nền tảng Backend-as-a-Service của Google
Firestore	Hệ quản trị cơ sở dữ liệu NoSQL dạng tài liệu của Firebase
UML	Unified Modeling Language – Ngôn ngữ mô hình hóa hợp nhất
BMI	Chỉ số khối cơ thể (Body Mass Index)
SDK	Android Package Kit (File cài đặt ứng dụng Android)
DBMS	Database Management System (Hệ quản trị cơ sở dữ liệu)
NoSQL	Not Only SQL (Cơ sở dữ liệu phi quan hệ)

## DANH MỤC HÌNH ẢNH

Hình 1.1: Ngôn ngữ Dart.....	7
Hình 1.2: Flutter.....	9
Hình 1.3: Firebase.....	13
Hình 2.1: Biểu đồ usecase tổng quan hệ thống.....	19
Hình 2.2: Biểu đồ lớp hệ thống.....	21
Hình 2.3: Biểu đồ hoạt động use case đăng ký.....	21
Hình 2.4: Biểu đồ hoạt động use case đăng nhập bằng email.....	22
Hình 2.5: Biểu đồ hoạt động use case đăng nhập bằng google.....	22
Hình 2.6: Biểu đồ hoạt động use case quản lý tài khoản cá nhân.....	23
Hình 2.7: Biểu đồ hoạt động use case theo dõi dinh dưỡng.....	24
Hình 2.8: Biểu đồ hoạt động use case theo dõi nước uống.....	25
Hình 2.9: Biểu đồ hoạt động use case đặt mục tiêu sức khoẻ.....	26
Hình 2.10: Biểu đồ hoạt động use case xem báo cáo sức khoẻ.....	26
Hình 2.11: Biểu đồ hoạt động use case xem nhật ký.....	27
Hình 2.12: Biểu đồ hoạt động use case quản lý món ăn.....	27
Hình 2.13: Biểu đồ hoạt động use case quản lý người dùng.....	28
Hình 2.14: Biểu đồ trình tự use case đăng ký tài khoản bằng email.....	29
Hình 2.15: Biểu đồ phân tích use case Đăng nhập bằng Email.....	29
Hình 2.16: Biểu đồ trình tự use case Đăng nhập bằng Google.....	30
Hình 2.17: Biểu đồ trình tự use case Quản lý tài khoản cá nhân.....	30
Hình 2.18: Biểu đồ trình tự use case Theo dõi dinh dưỡng.....	31
Hình 2.19: Biểu đồ trình tự use case Theo dõi nước uống.....	31
Hình 2.20: Biểu đồ trình tự use case đặt mục tiêu dinh dưỡng.....	32
Hình 2.21: Biểu đồ trình tự use case Xem báo cáo sinh dưỡng.....	32
Hình 2.22: Biểu đồ trình tự use case Xem nhật ký.....	32
Hình 2.23: Biểu đồ trình tự use case Quản lý món ăn.....	33
Hình 2.24: Biểu đồ trình tự use case Quản lý người dùng.....	33
Hình 2.25: Biểu đồ ERD.....	36
Hình 3.1: Build file APK bằng Android Studio.....	37
Hình 3.2: Giao diện màn hình Splash.....	39

Hình 3.3: Giao diện màn hình bắt đầu 1.....	40
Hình 3.4: Giao diện màn hình bắt đầu 2.....	41
Hình 3.5: Giao diện màn hình bắt đầu 3.....	42
Hình 3.6: Giao diện màn hình chào mừng 4.....	43
Hình 3.7: Giao diện đăng nhập.....	44
Hình 3.8: Giao diện đăng nhập bằng email.....	45
Hình 3.9: Giao diện đăng nhập bằng google.....	46
Hình 3.10: Giao diện đăng ký.....	47
Hình 3.11: Giao diện Recipes.....	48
Hình 3.12: Giao diện thêm món ăn.....	49
Hình 3.13: Giao diện chi tiết món ăn.....	50
Hình 3.14: Giao diện nhật ký dinh dưỡng.....	51
Hình 3.15: Giao diện xem nhật ký theo ngày.....	52
Hình 3.16: Giao diện thêm bữa ăn.....	53
Hình 3.17: Giao diện quản lý lượng nước.....	54
Hình 3.18: Giao diện báo cáo dinh dưỡng.....	55
Hình 3.19: Giao diện tài khoản.....	56
Hình 3.20: Giao diện chỉnh sửa hồ sơ.....	57
Hình 3.21: Giao diện cài đặt.....	58
Hình 3.22: Giao diện cài đặt tài khoản.....	59
Hình 3.23: Giao diện đổi email.....	60
Hình 3.24: Giao diện đổi mật khẩu.....	61
Hình 3.25: Giao diện thay đổi sáng tối.....	62
Hình 3.26: Giao diện thực đơn.....	63
Hình 3.27: Giao diện quản trị viên.....	64
Hình 3.28: Giao diện quản lý người dùng.....	65
Hình 3.29: Giao diện quản lý món ăn.....	66
Hình 3.30: Giao diện thêm món ăn vào danh sách.....	67
Hình 3.31: Giao diện chỉnh sửa món ăn.....	68

## **DANH MỤC BẢNG BIỂU**

Bảng 2.1: Food.....	34
Bảng 2.2: User.....	34
Bảng 2.3: ConsumedFoodLog.....	35
Bảng 2.4: DailyFoodLog.....	35
Bảng 2.5: Foodlog.....	36
Bảng 2.6: WaterLog.....	36
Bảng 3.1: Bảng lập kê hoạch kiểm thử.....	71
Bảng 3.2: Test case use case đăng nhập.....	72
Bảng 3.3: Test case use case đăng ký.....	72
Bảng 3.4: Test case use case quản lý hồ sơ cá nhân.....	73
Bảng 3.5: Test case use case thêm món ăn.....	73
Bảng 3.6: Test case use case quản lý thực phẩm.....	74
Bảng 3.7: Test case use case theo dõi nước uống.....	74
Bảng 3.8: Test case use case đặt mục tiêu sức khoẻ.....	75
Bảng 3.9: Test case use case xem báo cáo sức khoẻ.....	75
Bảng 3.10: Test case use case xem nhật ký.....	75



## LỜI MỞ ĐẦU

Trong xã hội hiện đại, việc quản lý chế độ ăn uống và theo dõi dinh dưỡng cá nhân trở nên ngày càng quan trọng. Tuy nhiên, nhiều người vẫn gặp khó khăn trong việc ghi chép thủ công lượng calo nạp vào, lượng nước uống hay theo dõi cân nặng, dẫn đến tình trạng ăn uống thiếu khoa học, thừa cân hoặc suy dinh dưỡng. Một số ít vẫn sử dụng phương pháp truyền thống như ghi vào sổ tay, excel hoặc google sheet. Cách làm này tuy hữu ích nhưng lại mất thời gian và thiếu tính linh động.

Trong khi đó, với sự bùng nổ của công nghệ và việc hầu hết mọi người đều sở hữu cho mình một chiếc điện thoại thông minh, thì việc có một ứng dụng hỗ trợ theo dõi dinh dưỡng và sức khỏe ngay trên điện thoại là rất cần thiết. Ứng dụng sẽ giúp người dùng dễ dàng ghi lại các bữa ăn, lượng nước tiêu thụ, theo dõi chỉ số BMI và lượng calo hằng ngày ở bất cứ đâu một cách đơn giản và tiện lợi hơn.

Vì vậy em quyết định thực hiện đồ án với đề tài “Xây dựng ứng dụng di động chăm sóc sức khoẻ sử dụng Firebase và Flutter”. Ứng dụng cho phép người dùng đăng ký tài khoản, nhập dữ liệu về cân nặng, chiều cao, bữa ăn, thức uống và theo dõi quá trình thay đổi theo thời gian. Qua đó giúp người dùng chủ động hơn trong việc kiểm soát chế độ ăn uống, duy trì vóc dáng, cải thiện sức khỏe và hướng tới một lối sống lành mạnh.

Phân tích và thiết kế hệ ứng dụng chăm sóc sức khoẻ đáp ứng nhu cầu người dùng hiện đại

- Phát triển giao diện người dùng bằng Flutter thân thiện, tương thích đa nền tảng và thiết bị.
- Tích hợp các tính năng quản lý thức ăn, calo, dinh dưỡng và báo cáo thống kê dinh dưỡng hàng ngày.
- Đảm bảo tính bảo mật và an toàn cho dữ liệu người dùng.

Đối tượng nghiên cứu của đề tài là các công nghệ, kiến trúc và giải pháp liên quan đến việc phát triển ứng dụng di động hỗ trợ quản lý chế độ ăn uống

và sức khỏe cá nhân. Cụ thể bao gồm: Nền tảng Flutter và ngôn ngữ Dart; Hệ sinh thái Firebase (Authentication, Firestore Database, Storage) phục vụ lưu trữ và đồng bộ dữ liệu; Các mô hình kiến trúc ứng dụng di động hiện đại (Provider, MVVM); Giải pháp tính toán và phân loại chỉ số BMI, theo dõi calo, protein, carb, chất béo và lượng nước tiêu thụ; Các chức năng bảo mật và xác thực người dùng; Quy trình quản lý thông tin cá nhân, thực phẩm, báo cáo và nhật ký dinh dưỡng trong ứng dụng.

Đề tài được nghiên cứu và triển khai trong khoảng thời gian 9 tuần, tập trung xây dựng một hệ thống ứng dụng di động phục vụ người dùng Việt Nam trong lĩnh vực: sức khỏe, thể hình, dinh dưỡng và công nghệ phát triển ứng dụng đa nền tảng.. Công nghệ: Tập trung vào nền tảng Flutter (ngôn ngữ Dart) kết hợp Firebase (Authentication, Firestore, Storage) để xây dựng backend không máy chủ, cùng với các thư viện hỗ trợ như Provider (quản lý trạng thái), để nâng cao trải nghiệm người dùng.

Kết thúc quá trình nghiên cứu với kết quả mong muốn đạt được là:

- Một ứng dụng di động hoàn chỉnh với đầy đủ chức năng cơ bản: đăng ký/đăng nhập, theo dõi dinh dưỡng, ghi nhật ký bữa ăn, tính toán BMI, báo cáo và thống kê sức khỏe
- Báo cáo chi tiết về quá trình phân tích, thiết kế và triển khai hệ thống
- Tài liệu hướng dẫn sử dụng và quản trị hệ thống
- Bộ mã nguồn hoàn thiện với các tính năng bảo mật được tích hợp
- Giải pháp kỹ thuật có thể mở rộng và phát triển thêm trong tương lai

## Cấu trúc của báo cáo

Báo cáo này là kết quả của một quá trình nghiên cứu và thực hành nghiêm túc, bao gồm các bước từ phân tích yêu cầu, thiết kế cơ sở dữ liệu, xây dựng giao diện, phát triển tính năng chính, đến kiểm thử và đánh giá hiệu quả của hệ thống. Sản phẩm cuối cùng không chỉ hướng đến việc số hóa trải nghiệm người dùng mà còn hỗ trợ người quản trị trong việc vận hành và quản lý hệ thống một cách hiệu quả. Nội dung đồ án gồm 3 chương:

## **Chương 1: Cơ sở lý thuyết**

Nêu các cơ sở lý thuyết và các công nghệ sẽ áp dụng vào hệ thống phần mềm sẽ triển khai, các ngôn ngữ lập trình và hệ quản trị cơ sở dữ liệu.

## **Chương 2: Phân tích và thiết kế hệ thống**

Thực hiện tìm hiểu và khảo sát hệ thống, từ đó trình bày các sơ đồ, mô hình use case, biểu đồ tuần tự... của các chức năng. Đồng thời, tiến hành thiết kế cơ sở dữ liệu cho hệ thống.

## **Chương 3: Cài đặt và kiểm thử hệ thống**

Từ những phân tích và thiết kế đã đưa ra, sau khi hoàn thiện xây dựng, áp dụng triển khai và kiểm thử các chức năng của hệ thống.

## CHƯƠNG 1: TỔNG QUAN VỀ DỰ ÁN PHẦN MỀM

### 1.1. Cơ sở lý thuyết

#### 1.1.1. Phát triển ứng dụng trên thiết bị di động

Phát triển ứng dụng trên thiết bị di động (Mobile Application Development) là quá trình xây dựng và triển khai phần mềm chạy trên các nền tảng di động như điện thoại thông minh và máy tính bảng. Đây là một lĩnh vực quan trọng trong phát triển phần mềm hiện đại, với yêu cầu cao về hiệu năng, giao diện thân thiện, khả năng phản hồi nhanh và tối ưu hóa tài nguyên hệ thống. Ứng dụng di động không chỉ hỗ trợ người dùng trong sinh hoạt và công việc mà còn đóng vai trò không thể thiếu trong các lĩnh vực như thương mại, giáo dục, tài chính, và chăm sóc sức khỏe.

Có nhiều cách tiếp cận trong phát triển ứng dụng di động, phổ biến nhất là phát triển ứng dụng native, ứng dụng hybrid, và ứng dụng cross-platform. Ứng dụng native được phát triển chuyên biệt cho từng hệ điều hành như Android (dùng Java/Kotlin) hoặc iOS (dùng Swift), mang lại hiệu suất cao và trải nghiệm tối ưu cho người dùng. Trong khi đó, hybrid và cross-platform (như Flutter, React Native) giúp giảm chi phí phát triển và rút ngắn thời gian triển khai bằng cách tái sử dụng mã nguồn trên nhiều nền tảng.

Một ứng dụng di động hiện đại thường được xây dựng theo mô hình kiến trúc MVVM (Model – View – ViewModel), giúp tách biệt rõ ràng giữa giao diện và xử lý dữ liệu, tăng tính linh hoạt và dễ bảo trì. Bên cạnh đó, các nền tảng dịch vụ như Firebase được sử dụng phổ biến để cung cấp các tính năng như xác thực người dùng, cơ sở dữ liệu thời gian thực, lưu trữ đám mây và thông báo đẩy – giúp đơn giản hóa quy trình phát triển mà vẫn đảm bảo hiệu quả.

Trong đề tài này, ứng dụng được phát triển trên Flutter – framework cross-platform do Google phát triển, cho phép triển khai trên cả Android và iOS. Dữ liệu người dùng và nhật ký dinh dưỡng được quản lý thông qua Firebase Firestore, hình ảnh được lưu trữ trên Firebase Storage, trong khi

chức năng xác thực người dùng sử dụng Firebase Authentication (hỗ trợ đăng nhập Email, Google). Quá trình thiết kế và phát triển được thực hiện với công cụ Android Studio và Visual Studio Code, giúp xây dựng hệ thống đáp ứng tốt yêu cầu thực tiễn, đồng thời rèn luyện kỹ năng phân tích yêu cầu, thiết kế và triển khai ứng dụng di động hiện đại.

### **1.1.2. Phân tích thiết kế hệ thống**

Phân tích và thiết kế hệ thống là giai đoạn đầu tiên và đóng vai trò then chốt trong quá trình phát triển phần mềm. Mục tiêu của bước này là hiểu rõ yêu cầu của người dùng, tổ chức các chức năng hợp lý, và xây dựng một mô hình hệ thống có thể phát triển được về sau. Trong quá trình phân tích, người phát triển tiến hành khảo sát thực tế hoặc thu thập yêu cầu từ người dùng thông qua bảng hỏi, phỏng vấn hoặc nghiên cứu tài liệu. Từ đó, các chức năng chính và dữ liệu liên quan sẽ được xác định và biểu diễn trực quan thông qua các sơ đồ như Use Case (sơ đồ trường hợp sử dụng), biểu đồ luồng dữ liệu (DFD), và sơ đồ quan hệ thực thể (ERD).

Sau giai đoạn phân tích là bước thiết kế hệ thống. Thiết kế hệ thống bao gồm cả thiết kế chức năng và thiết kế cơ sở dữ liệu. Thiết kế chức năng thể hiện cách hệ thống xử lý thông tin, tổ chức logic xử lý và giao tiếp giữa các thành phần trong hệ thống. Thiết kế cơ sở dữ liệu lại chú trọng đến cách lưu trữ dữ liệu hiệu quả, tránh dư thừa và đảm bảo toàn vẹn dữ liệu. Trong dự án này, kiến trúc phần mềm được triển khai theo mô hình MVC (Model - View - Controller) – một mô hình hiện đại và phổ biến trong các framework PHP như Laravel. MVC giúp phân chia trách nhiệm giữa các thành phần trong ứng dụng: phần Model quản lý dữ liệu và logic nghiệp vụ, View hiển thị giao diện người dùng, và Controller xử lý yêu cầu, điều phối giữa Model và View. Cách tiếp cận này giúp hệ thống dễ bảo trì, mở rộng, đồng thời tăng tính chuyên nghiệp trong tổ chức mã nguồn.

Thông qua việc phân tích và thiết kế hệ thống không chỉ nắm được cách xây dựng một hệ thống phần mềm bài bản mà còn rèn luyện được tư duy hệ thống, khả năng tổ chức và vận dụng kiến thức lý thuyết vào thực tiễn.

## **1.2. Ngôn ngữ mô hình hóa UML**

UML (Unified Modeling Language – Ngôn ngữ mô hình hóa hợp nhất) là một ngôn ngữ chuẩn được sử dụng để mô hình hóa, trực quan hóa, xây dựng và lập tài liệu cho các hệ thống phần mềm. UML được phát triển nhằm mục tiêu thống nhất các phương pháp biểu diễn phần mềm hướng đối tượng và hiện đã trở thành tiêu chuẩn công nghiệp trong thiết kế hệ thống phần mềm hiện đại.

Khác với các ngôn ngữ lập trình truyền thống, UML không nhằm mục đích mô tả chi tiết mã nguồn, mà tập trung mô hình hóa cấu trúc và hành vi của hệ thống ở mức trừu tượng. Việc sử dụng UML giúp các bên liên quan – bao gồm nhà phát triển, nhà phân tích hệ thống, kiểm thử viên và khách hàng – có thể hiểu được thiết kế hệ thống một cách trực quan, rõ ràng và thống nhất.

+ Mục tiêu của UML

- Hỗ trợ mô hình hóa toàn diện các cấu trúc và hành vi của hệ thống.
- Tạo ra ngôn ngữ chung trong nhóm phát triển phần mềm.
- Cung cấp công cụ hỗ trợ thiết kế, phân tích và kiểm thử hệ thống.
- Tạo tài liệu chính thức cho phần mềm trong suốt vòng đời phát triển.

+ Một số sơ đồ UML phổ biến bao gồm:

- *Biểu đồ lớp (Class Diagram)*: Mô tả cấu trúc tĩnh của hệ thống, thể hiện các lớp, thuộc tính, phương thức và mối quan hệ giữa các lớp như kế thừa, liên kết, phụ thuộc,...
- *Biểu đồ gói (Package Diagram)*: Giúp nhóm các lớp có liên quan vào một gói (package), thuận tiện cho việc tổ chức và quản lý mã nguồn.

- *Biểu đồ chức năng (Use Case Diagram)*: Mô tả chức năng của hệ thống. Biểu đồ này xác định các tác nhân (actor) và các trường hợp sử dụng (use case) mà hệ thống cần đáp ứng.
- *Biểu đồ hoạt động (Activity Diagram)*: Diễn tả luồng công việc hoặc các bước xử lý trong một quy trình nghiệp vụ cụ thể. Biểu đồ này thường dùng để mô hình hóa thuật toán, logic nghiệp vụ hoặc quy trình xử lý đơn hàng, đăng ký, thanh toán,...
- *Biểu đồ tuần tự (Sequence Diagram)*: Minh họa thứ tự tương tác giữa các đối tượng theo thời gian để thực hiện một chức năng cụ thể. Biểu đồ này giúp mô tả cách các đối tượng gọi lẫn nhau để hoàn thành nghiệp vụ.

UML không chỉ giúp định hình kiến trúc hệ thống rõ ràng mà còn là tài liệu hữu ích trong suốt vòng đời phát triển phần mềm, từ phân tích, thiết kế, lập trình đến kiểm thử và bảo trì.

### **1.3. Các công nghệ sử dụng trong đề tài**

#### **1.3.1. Dart**

Dart là ngôn ngữ lập trình mã nguồn mở, kiểu tĩnh, được phát triển bởi Google và là nền tảng chính để xây dựng ứng dụng với Flutter. Dart hỗ trợ cả lập trình hướng đối tượng lẫn lập trình hàm, đồng thời cung cấp cú pháp gọn gàng, dễ tiếp cận với những người đã quen thuộc với các ngôn ngữ như Java, JavaScript, C# hay Kotlin. Ngôn ngữ này được thiết kế nhằm mang lại hiệu năng cao, dễ dàng phát triển giao diện người dùng và có khả năng biên dịch sang mã máy gốc (native code) lẫn JavaScript để chạy trên nhiều nền tảng.



*Hình 1.1: Ngôn ngữ Dart*

## \* **Ưu điểm**

+Hỗ trợ đa nền tảng: Dart có thể biên dịch trực tiếp sang mã máy (AOT – Ahead of Time) cho Android/iOS, hoặc biên dịch sang JavaScript để chạy trên Web. Điều này cho phép viết một lần, triển khai nhiều nơi thông qua Flutter.

+Cú pháp rõ ràng, dễ học: Dart khá giống với C/Java/C#, vì vậy lập trình viên dễ dàng tiếp cận và sử dụng.

+Hiệu năng cao: Nhờ khả năng biên dịch AOT và JIT (Just in Time), ứng dụng Flutter viết bằng Dart chạy mượt mà, tốc độ khởi động nhanh, thích hợp cả với ứng dụng lớn.

+Hỗ trợ lập trình bất đồng bộ: Dart có async/await, Future, Stream, giúp xử lý các tác vụ bất đồng bộ (như gọi API, đọc ghi dữ liệu) một cách trực quan.

+Hệ sinh thái phong phú: Dart có kho thư viện pub.dev với hàng ngàn package hỗ trợ sẵn, giúp tăng tốc phát triển ứng dụng.

## \* **Nhược điểm**

+Phổ biến hạn chế: So với Java, JavaScript hay Python, Dart chưa phổ biến bằng, do đó cộng đồng và tài nguyên học tập ít hơn.

+Phụ thuộc mạnh vào Flutter: Dù Dart có thể dùng độc lập, nhưng hiện tại nó chủ yếu được sử dụng trong hệ sinh thái Flutter.

+Khả năng tích hợp với nền tảng khác: Dart không có mức độ tương thích ngược hay tích hợp sâu như Java (với JVM) hay Kotlin (với Java). Vì vậy khi cần dùng đến thư viện đặc thù cho nền tảng, lập trình viên phải viết native code (Java/Kotlin cho Android, Swift/Objective-C cho iOS) và kết nối thông qua “platform channel”.

### 1.3.2. Flutter



*Hình 1.2: Flutter*

Flutter là một framework phát triển ứng dụng đa nền tảng (cross-platform) do Google phát triển, ra mắt chính thức từ năm 2017. Flutter cho phép lập trình viên xây dựng ứng dụng di động, web và desktop từ cùng một mã nguồn duy nhất, nhờ đó tiết kiệm chi phí và rút ngắn thời gian triển khai so với cách phát triển native truyền thống.

Điểm nổi bật của Flutter là sử dụng ngôn ngữ lập trình Dart và cơ chế biên dịch trực tiếp sang mã máy (native code), mang lại hiệu năng cao và khả năng phản hồi nhanh. Flutter không dựa vào các thành phần giao diện có sẵn của hệ điều hành, mà tự render giao diện thông qua engine đồ họa Skia. Điều này giúp ứng dụng có giao diện đồng nhất trên nhiều nền tảng, đồng thời dễ dàng tùy chỉnh và thiết kế UI theo yêu cầu.

## **Ưu điểm của Flutter:**

- + Phát triển đa nền tảng: Một mã nguồn có thể chạy trên Android, iOS, Web, Windows, macOS và Linux.
- + Hiệu năng cao: Nhờ biên dịch AOT và engine Skia, ứng dụng Flutter chạy mượt mà, tốc độ khởi động nhanh.
- + UI linh hoạt và đẹp mắt: Flutter cung cấp hàng loạt widget dựng sẵn theo phong cách Material Design (Google) và Cupertino (Apple), đồng thời hỗ trợ dễ dàng tùy biến.
- + Hot Reload: Cho phép thay đổi giao diện hoặc logic và xem kết quả ngay lập tức mà không cần build lại toàn bộ ứng dụng.
- + Hệ sinh thái phong phú: Kho thư viện pub.dev có nhiều package hỗ trợ từ giao diện, cơ sở dữ liệu, API đến AI/ML.

## **Nhược điểm:**

- + Dung lượng ứng dụng lớn hơn native (do phải tích hợp engine riêng).
- + Thư viện chưa phong phú bằng native, một số tính năng đặc thù vẫn cần viết code native và kết nối qua platform channel.
- + Cộng đồng phát triển đang lớn mạnh nhưng vẫn nhỏ hơn so với các framework lâu đời như React Native.
- + Trong đợt tài này, Flutter được chọn làm công nghệ chính để phát triển ứng dụng theo dõi dinh dưỡng và sức khỏe. Nhờ khả năng phát triển nhanh, giao diện đẹp, dễ dàng mở rộng đa nền tảng và tích hợp tốt với Firebase (Authentication, Firestore, Storage), Flutter đáp ứng tốt yêu cầu của hệ thống.

### **1.4. Mô hình Model – View – ViewModel**

- Model:

- + Model là thành phần đại diện cho dữ liệu và logic kinh doanh của ứng dụng. Nó bao gồm các lớp chứa dữ liệu, các lớp quản lý dữ liệu (như repository), và các quy tắc xử lý dữ liệu.

+ Xử lý và quản lý dữ liệu, thường từ các nguồn như cơ sở dữ liệu, API, hoặc các dịch vụ khác.

- View:

+ Mô tả: View là thành phần đại diện cho giao diện người dùng. Trong Android, nó bao gồm các hoạt động (Activity), các mảnh (Fragment), và các thành phần giao diện khác.

+ Nhiệm vụ: Hiển thị dữ liệu cho người dùng và điều khiển các sự kiện người dùng (như click, nhập liệu) tới ViewModel.

- ViewModel:

+ ViewModel là lớp trung gian giữa View và Model. Nó chứa các logic trình bày dữ liệu và giữ trạng thái dữ liệu cho View.

+ Lấy dữ liệu từ Model, thực hiện các thao tác cần thiết và cung cấp dữ liệu đã được xử lý cho View. ViewModel cũng xử lý các sự kiện từ view và điều khiển luồng dữ liệu giữa Model và View.

- Cách hoạt động của MVVM:

+ **ViewModel cung cấp dữ liệu cho View:** View sẽ quan sát các đối tượng có kiểu dữ liệu là LiveData hoặc Observer trong ViewModel và tự động cập nhật khi dữ liệu thay đổi.

+ **ViewModel nhận sự kiện từ View:** Khi người dùng tương tác với giao diện, View sẽ gửi các sự kiện này tới ViewModel. ViewModel sẽ xử lý sự kiện, như thêm sửa xóa, cập nhật. Khi xử lý xong, View sẽ tự động được cập nhật.

+ **Model quản lý dữ liệu:** Model thực hiện các thao tác truy vấn dữ liệu, lưu trữ dữ liệu và các logic kinh doanh. Model không biết về sự tồn tại của View và ViewModel.

Trong Android thì thường sử dụng Livedata bởi Livedata có thể theo dõi vòng đời của đối tượng. Cụ thể nó chỉ thông báo đến các đối tượng, thành phần quan sát mà đối tượng, thành phần đó đang ở trạng thái hoạt động

## 1.5. Hệ quản trị cơ sở dữ liệu

### 1.5.1. Các khái niệm

Cơ sở dữ liệu (CSDL – Database) là một tập hợp dữ liệu có tổ chức, được lưu trữ và quản lý theo một cấu trúc nhất định, nhằm phục vụ cho việc truy xuất, cập nhật và quản lý thông tin một cách hiệu quả. Cơ sở dữ liệu thường được sử dụng trong các hệ thống thông tin, từ quy mô nhỏ (như trang web cá nhân) đến các hệ thống lớn (như ngân hàng, thương mại điện tử, chính phủ,...).

Dữ liệu trong cơ sở dữ liệu thường phản ánh các thực thể trong thế giới thực và mối quan hệ giữa chúng. Ví dụ, trong một hệ thống bán hàng, cơ sở dữ liệu có thể lưu trữ thông tin về khách hàng, sản phẩm, đơn hàng, và các giao dịch tương ứng.

Hệ quản trị cơ sở dữ liệu (Database Management System – DBMS) là phần mềm trung gian cho phép người dùng tương tác với cơ sở dữ liệu. DBMS thực hiện các nhiệm vụ như:

- Lưu trữ, truy xuất, cập nhật và xóa dữ liệu.
- Quản lý truy cập đồng thời từ nhiều người dùng.
- Đảm bảo tính toàn vẹn và nhất quán của dữ liệu.
- Cung cấp các công cụ bảo mật, sao lưu và phục hồi dữ liệu.

DBMS giúp người dùng không cần phải quan tâm đến cách dữ liệu được tổ chức và lưu trữ ở mức thấp, mà chỉ cần tập trung vào cách khai thác và sử dụng dữ liệu ở mức logic.

Có nhiều loại DBMS khác nhau, được phân loại dựa trên mô hình dữ liệu mà chúng sử dụng. Hai loại phổ biến nhất là:

- Hệ quản trị cơ sở dữ liệu quan hệ (Relational DBMS – RDBMS): Sử dụng mô hình bảng (table) để lưu trữ dữ liệu. Dữ liệu được tổ chức dưới dạng hàng (record) và cột (field), có khóa chính và khóa ngoại để liên kết giữa các bảng. Ví dụ: MySQL, PostgreSQL, Oracle, SQL Server.

- Hệ quản trị cơ sở dữ liệu phi quan hệ (NoSQL DBMS): Không sử dụng mô hình bảng, thay vào đó dữ liệu có thể được lưu dưới dạng document (tài liệu), key-value, column-family, hoặc graph. NoSQL thích hợp với các hệ thống yêu cầu linh hoạt về cấu trúc dữ liệu, quy mô lớn, hiệu năng cao. Ví dụ: MongoDB, Redis, Cassandra, Neo4j.

Trong phát triển phần mềm, DBMS đóng vai trò như trái tim của hệ thống, nơi lưu trữ và quản lý toàn bộ dữ liệu nghiệp vụ. Một hệ thống phần mềm hiệu quả không chỉ cần logic xử lý tốt, mà còn cần có cấu trúc dữ liệu khoa học và khả năng truy xuất tối ưu.

### **1.5.2. Firebase**

#### **\* Giới thiệu về FireBase**



*Hình 1.3: Firebase*

- Firebase là một nền tảng để phát triển ứng dụng di động và trang web, bao gồm các API đơn giản và mạnh mẽ mà không cần backend hay server.
- Google Firebase là một nền tảng toàn diện để xây dựng các ứng dụng web và di động, do Google vận hành và phát triển. Nó cung cấp nhiều công cụ và dịch vụ hữu ích, giúp đơn giản hóa quy trình xây dựng, phát triển, nâng cao chất lượng ứng dụng và thậm chí giúp kiểm tiền từ ứng dụng.
- Firebase cung cấp cả cơ sở hạ tầng back-end và thư viện phát triển phía client, giúp developer tiết kiệm thời gian, nguồn lực, tập trung hơn vào việc phát triển front- end của ứng dụng.

#### **\* Các dịch vụ của Firebase:**

- Firebase cung cấp hai cơ sở dữ liệu và Cloud Firestore và Realtime Database (RTDB), giúp lưu trữ và đồng bộ hóa dữ liệu trong thời gian thực trên máy khách, ngay cả khi người dùng ngoại tuyến.
- Cloud Firestore (hay Google Firestore): là một cơ sở dữ liệu NoSQL lưu trữ đám mây, cho phép developer lưu trữ, truy vấn và truy xuất dữ liệu có cấu trúc một cách linh hoạt và có thể mở rộng.
- Firebase Cloud Functions là một nền tảng dự trên điện toán đám mây không có máy chủ (serverless), cho phép developer chạy code back-end tùy chỉnh để phản hồi các sự kiện của người dùng.
- Firebase cung cấp các thư viện UI, back-end và SDK có sẵn để tạo các chức năng đăng ký, đăng nhập và quản lý danh tính của người dùng.
- Firebase Cloud Messaging (hay FCM) là dịch vụ nhắn tin đa nền tảng miễn phí. Nó cho phép chủ ứng dụng gửi tin nhắn (lên tới 4KB đối với nhắn tin tức thời) cho người dùng để thu hút và giữ chân họ.
- Developer có thể sử dụng FCM để gửi thông báo đẩy (push notification), cập nhật nội dung ứng dụng, ... ngay cả khi người không dùng ứng dụng.
- Trên đây là một vài dịch vụ Firebase cung cấp. Ngoài ra còn rất nhiều dịch vụ khác và Firebase cung cấp như Firebase Cloud Storage, Firebase Analytics (Phân tích),...

### \* Cơ sở dữ liệu NoSQL

Cơ sở dữ liệu NoSQL (Not Only SQL) là một loại cơ sở dữ liệu được thiết kế để xử lý một lượng lớn dữ liệu phi cấu trúc, không yêu cầu mô hình dữ liệu cố định và dễ dàng mở rộng theo chiều ngang. NoSQL thì chia ra làm 4 loại chính: **Key-value, Document, Column-Family, và Graph.**

#### **Đặc điểm:**

- Chấp nhận dư thừa dữ liệu.
- Dữ liệu dạng phi cấu trúc, phi quan hệ (non-relational).
- Dễ dàng mở rộng theo chiều ngang.
- Đa dạng hóa các kiểu cơ sở dữ liệu.

- Trạng thái dữ liệu và độc lập mỗi khi diễn ra sự thay đổi.
- Dữ liệu lưu trữ nhiều dạng linh hoạt (key-value, document, graphs).
- Đễ dàng thay đổi cấu trúc Cơ sở dữ liệu hơn so với SQL.

### **Ưu điểm:**

- Nhìn vào đặc điểm trên ta có thể thấy được các ưu điểm của loại cơ sở dữ liệu này. Ví dụ như linh hoạt về cấu trúc. Không cần các mối quan hệ ràng buộc phức tạp. Từ đó dễ dàng mở rộng theo chiều ngang.

- Việc lưu trữ dữ liệu trùng lặp (chấp nhận dư thừa dữ liệu). Giúp giảm số lượng truy vấn cần thiết để lấy ra dữ liệu, từ đó cải thiện được hiệu năng.

### **Nhược điểm:**

- Thiếu chuẩn hóa cơ sở dữ liệu: Việc chấp nhận dư thừa dữ liệu vừa là điểm mạnh nhưng cũng đồng thời là điểm yếu của NoSQL. Việc dư thừa có thể dẫn đến không nhất quán về dữ liệu trong database nếu như không được quản lý tốt.

- Hạn chế trong các truy vấn phức tạp khi phải thực hiện ở client, không thể sử dụng JOIN như SQL.

- Bảo mật của ứng dụng phụ thuộc vào rule được thiết lập có thể dẫn đến lỗ hổng bảo mật.

### **FireStore**

- Thiết kế và quản lý các cấu trúc dữ liệu phức tạp trong ứng dụng của bạn. FireStore là một dịch vụ cơ sở dữ liệu NoSQL, là một dịch vụ thuộc nền tảng Firebase. FireStore được thiết kế để hỗ trợ các ứng dụng thời gian thực với khả năng đồng bộ hóa dữ liệu mạnh mẽ và tính năng hỗ trợ offline.

- Trong Firebase FireStore, mỗi một collection tương đương với một bảng trong SQL, mỗi một bản ghi dữ liệu tương đương với một document. Tập hợp nhiều document lại sẽ thành một collection. Trong một collection có thể chứa các collection con khác, tùy theo cách thiết kế của người dùng.

- Document là các bản ghi riêng lẻ trong Firestore chứa các giá trị dữ liệu thực tế. Nói chung, tài liệu bao gồm các cặp khóa- giá trị được gọi là trường, mỗi trường có tên và giá trị tương ứng. Firestore hỗ trợ nhiều loại dữ liệu khác nhau cho các trường, bao gồm chuỗi, số, Boolean, mảng, map, ...

- Ngoài ra, trong một collection có thể lồng các collection khác. Cấu trúc lồng nhau này cho phép linh hoạt hơn trong quá trình thiết kế.

## CHƯƠNG 2: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

### 2.1. Khảo sát hệ thống

Ứng dụng chăm sóc sức khoẻ là một ứng dụng di động đa nền tảng (Android và iOS) được phát triển bằng Flutter, hướng đến người dùng có nhu cầu theo dõi chế độ ăn uống và sức khỏe cá nhân một cách hiệu quả và tiện lợi. Với giao diện hiện đại, thân thiện và trực quan, ứng dụng mang đến trải nghiệm theo dõi dinh dưỡng dễ dàng, phù hợp với nhiều đối tượng người dùng.

Ứng dụng cung cấp đầy đủ các tính năng thiết yếu như: ghi log món ăn hàng ngày (bao gồm calo, protein, carb, chất béo, trọng lượng, loại bữa ăn), theo dõi lượng nước uống, và tính toán chỉ số BMI để đánh giá tình trạng sức khỏe. Người dùng có thể dễ dàng thêm, chỉnh sửa hoặc xóa món ăn, cũng như quản lý hồ sơ cá nhân với các thông tin cơ bản như tuổi, cân nặng, chiều cao, mục tiêu calo và ảnh đại diện.

Bên cạnh đó, hệ thống còn hỗ trợ người dùng xem báo cáo trực quan dưới dạng biểu đồ, giúp phân tích chế độ ăn uống, theo dõi tiến trình tiêu thụ dinh dưỡng và lượng nước. Nhờ đó, người dùng có thể đưa ra những quyết định hợp lý nhằm duy trì hoặc cải thiện sức khỏe.

Dữ liệu được đồng bộ hóa qua tài khoản cá nhân, cho phép người dùng truy cập và quản lý sức khỏe của mình trên nhiều thiết bị khác nhau mà không lo mất dữ liệu. Bên cạnh đó, hệ thống còn đảm bảo tính bảo mật thông tin cá nhân và dữ liệu sức khỏe.

Với các tính năng đa dạng, hiện đại và khả năng mở rộng cao, ứng dụng hứa hẹn trở thành người bạn đồng hành đáng tin cậy, giúp người dùng chủ động quản lý dinh dưỡng, theo dõi sức khỏe và duy trì lối sống lành mạnh trên nền tảng di động.

## 2.2. Mô tả bài toán

Ứng dụng được xây dựng nhằm hỗ trợ người dùng theo dõi chế độ ăn uống và sức khỏe hằng ngày một cách khoa học và tiện lợi. Thay vì phải tự ghi nhớ lượng calo hay thành phần dinh dưỡng, hệ thống sẽ cung cấp danh mục món ăn được quản lý rõ ràng và chức năng tìm kiếm để người dùng dễ dàng lựa chọn.

Sau khi tạo tài khoản và đăng nhập, người dùng có thể theo dõi danh sách món ăn, thêm món ăn đã tiêu thụ vào nhật ký theo từng bữa, quản lý lượng calo, protein, carb, chất béo và lượng nước uống hằng ngày. Ứng dụng cho phép người dùng đặt mục tiêu (ví dụ calo hoặc lượng nước), hệ thống sẽ tự động tính toán mức khuyến nghị dựa trên thông tin cá nhân và hiển thị tiến trình thực hiện.

Ngoài ra, hệ thống còn cung cấp báo cáo trực quan bằng biểu đồ xu hướng calo, phân tích thành phần dinh dưỡng, cũng như khả năng chỉnh sửa thông tin cá nhân, từ đó đem lại sự thuận tiện, chính xác và an toàn trong quá trình theo dõi sức khỏe. Với giao diện thân thiện và tích hợp quản lý dữ liệu trên đám mây, người dùng có thể dễ dàng theo dõi và điều chỉnh thói quen sinh hoạt của mình một cách khoa học và hiệu quả hơn.

## 2.3. Xác định yêu cầu của hệ thống

1. Quản lý danh mục mua thực phẩm và thông tin liên quan:
2. Tài khoản người dùng và quản lý:
  - Đăng ký tài khoản với xác thực email/số điện thoại.
  - Đăng nhập/đăng xuất.
  - Quản lý thông tin cá nhân: họ tên, tuổi, cân nặng, chiều cao, ảnh đại diện.
  - Theo dõi chỉ số BMI, lượng calo tiêu thụ và lượng nước uống hằng ngày.
  - Quản lý lịch sử nhật ký ăn uống và hoạt động.
3. Tìm kiếm và gợi ý:

- Tìm kiếm món ăn theo tên.
- Bộ lọc linh hoạt theo loại bữa ăn ( sáng, trưa, tối, snack ).

4. Nhật ký bữa ăn và quản lý tiêu thụ:

- Ghi lại món ăn theo từng bữa với thông tin chi tiết dinh dưỡng.
- Quản lý lượng nước uống hằng ngày.
- Hiển thị tổng lượng calo và các chất dinh dưỡng (carb, protein, chất béo).

5. Đánh giá và báo cáo:

- Cung cấp báo cáo trực quan bằng biểu đồ theo ngày, tuần, tháng.
- Hiển thị tiến độ so với mục tiêu calo hoặc lượng nước

6. Quản lý lịch sử:

- Theo dõi món ăn đã tiêu thụ theo từng ngày.
- Quản lý lịch sử log thức ăn và nước uống để dễ dàng xem lại.
- Hỗ trợ chỉnh sửa hoặc xóa mục đã ghi trong nhật ký.

Các yêu cầu trên đảm bảo hệ thống không chỉ giúp người dùng theo dõi chế độ dinh dưỡng và sức khỏe một cách khoa học, mà còn có khả năng mở rộng, đem lại trải nghiệm thuận tiện, chính xác và an toàn.

## 2.4. Thiết kế hệ thống

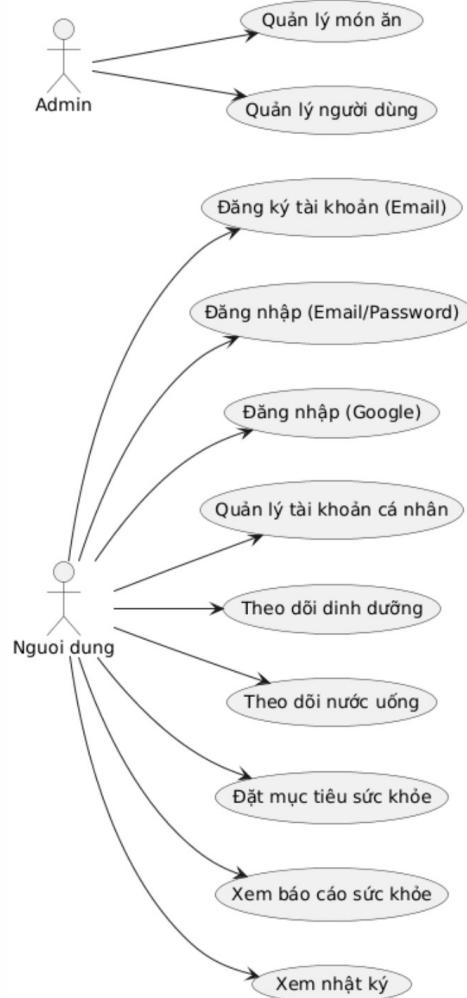
### 2.4.1. Các tác nhân của hệ thống

Quản trị viên (Administrator): Có quyền quản lý toàn bộ dữ liệu hệ thống như danh mục thực phẩm, thông tin người dùng, báo cáo thống kê. Quản trị viên có thể thêm, sửa, xóa dữ liệu, quản lý vai trò người dùng, theo dõi hoạt động sử dụng và đảm bảo hệ thống vận hành ổn định.

Khách hàng (User): Là người dùng có tài khoản trong hệ thống. Họ có thể đăng nhập, cập nhật thông tin cá nhân (tuổi, cân nặng, chiều cao), theo dõi chỉ số BMI, thêm món ăn đã tiêu thụ vào nhật ký, ghi nhận lượng nước uống hằng ngày. Người dùng cũng có thể xem lại lịch sử tiêu thụ, theo dõi tiến trình đạt mục tiêu calo/nước, và nhận báo cáo chi tiết về chế độ ăn uống của mình.

### 2.4.2. Các use case trong hệ thống

\* Sơ đồ use case tổng quan hệ thống



Hình 2.4: Biểu đồ usecase tổng quan hệ thống

#### Quyền Hạn Khách Hàng:

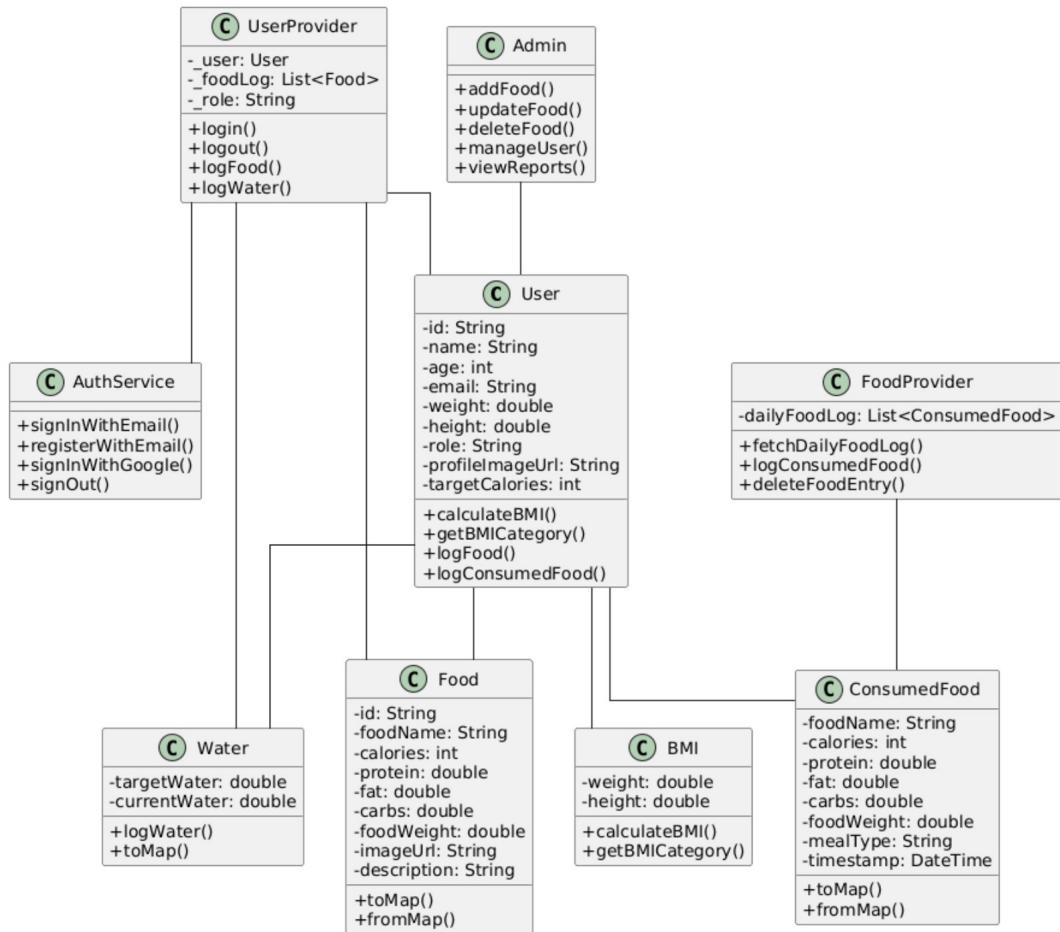
- Đăng ký & xác thực: Tạo tài khoản cá nhân bằng email hoặc Google để trở thành người dùng chính thức.
- Đăng nhập/khôi phục mật khẩu: Truy cập tài khoản đã đăng ký, hỗ trợ khôi phục mật khẩu khi quên.
- Tìm kiếm & duyệt món ăn: Tìm kiếm thực phẩm theo tên, duyệt danh mục món ăn kèm calo và dinh dưỡng.
- Quản lý nhật ký: Thêm món ăn vào nhật ký bữa ăn (sáng, trưa, tối, snack), ghi nhận lượng nước uống.

- Cá nhân hóa thông tin: Cập nhật họ tên, tuổi, cân nặng, chiều cao, ảnh đại diện.
- Theo dõi sức khỏe: Tự động tính toán và hiển thị chỉ số BMI, tiến trình đạt mục tiêu calo/nước.
- Báo cáo và thống kê: Xem báo cáo theo tháng
- Quản lý dữ liệu cá nhân: Xem lại lịch sử món ăn đã ghi, chỉnh sửa hoặc xóa thông tin trong nhật ký.

### **Quyền Hạn Quản Trị Viên:**

- Đăng nhập quản trị: Truy cập vào hệ thống bằng tài khoản có quyền quản lý.
- Quản lý tài khoản người dùng: Xem thông tin chi tiết, cập nhật hoặc vô hiệu hóa tài khoản khi cần thiết.
- Quản lý thực phẩm: Thêm món ăn mới, chỉnh sửa thông tin (calo, protein, carb, chất béo, hình ảnh), xóa món ăn không còn sử dụng.
- Quản lý nhật ký dinh dưỡng: Theo dõi dữ liệu nhật ký của người dùng để đảm bảo tính chính xác và thống nhất.
- Quản lý danh mục & phân loại: Quản lý danh mục món ăn theo nhóm (ví dụ: cơm, rau, thịt, đồ uống).
- Báo cáo và thống kê: Xem số liệu về người dùng, lượng calo tiêu thụ trung bình, xu hướng dinh dưỡng và các thống kê khác để phục vụ nghiên cứu và cải tiến hệ thống.

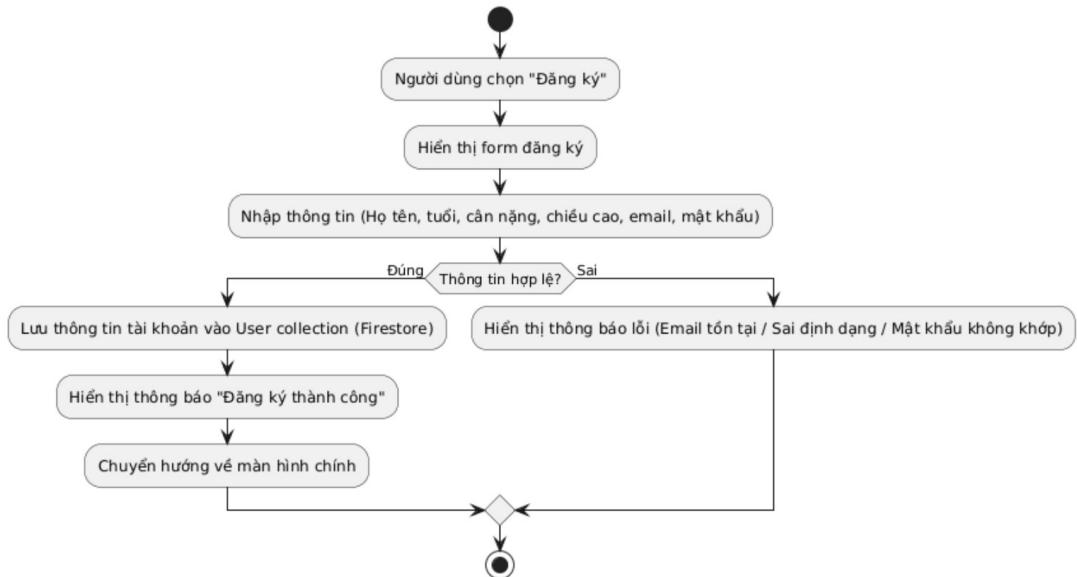
### 2.4.3. Biểu đồ lớp



Hình 2.5: Biểu đồ lớp hệ thống

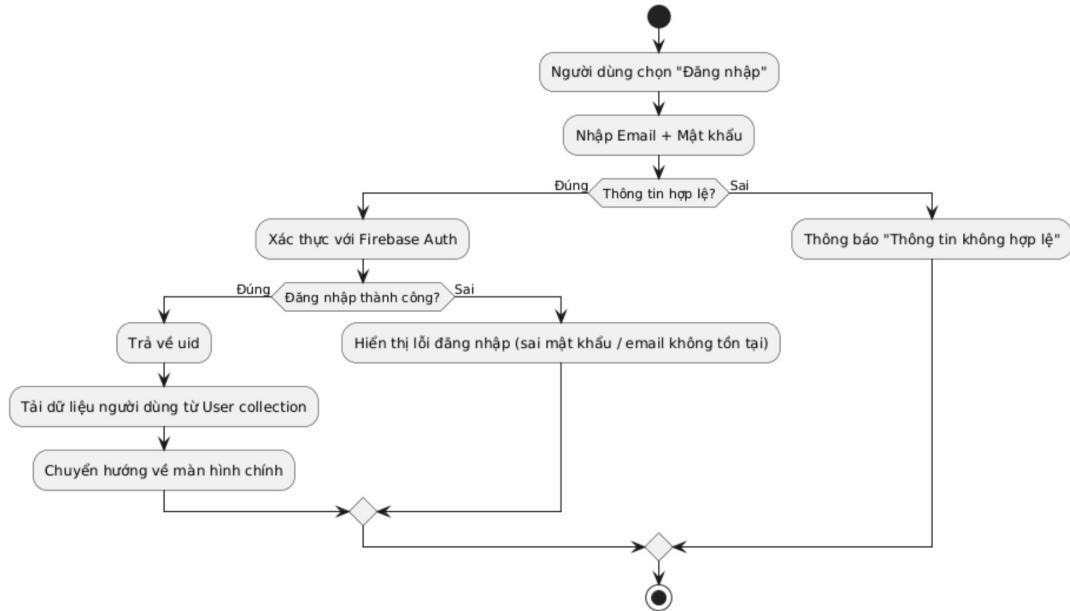
### 2.4.4. Biểu đồ hoạt động

\* Use case Đăng ký



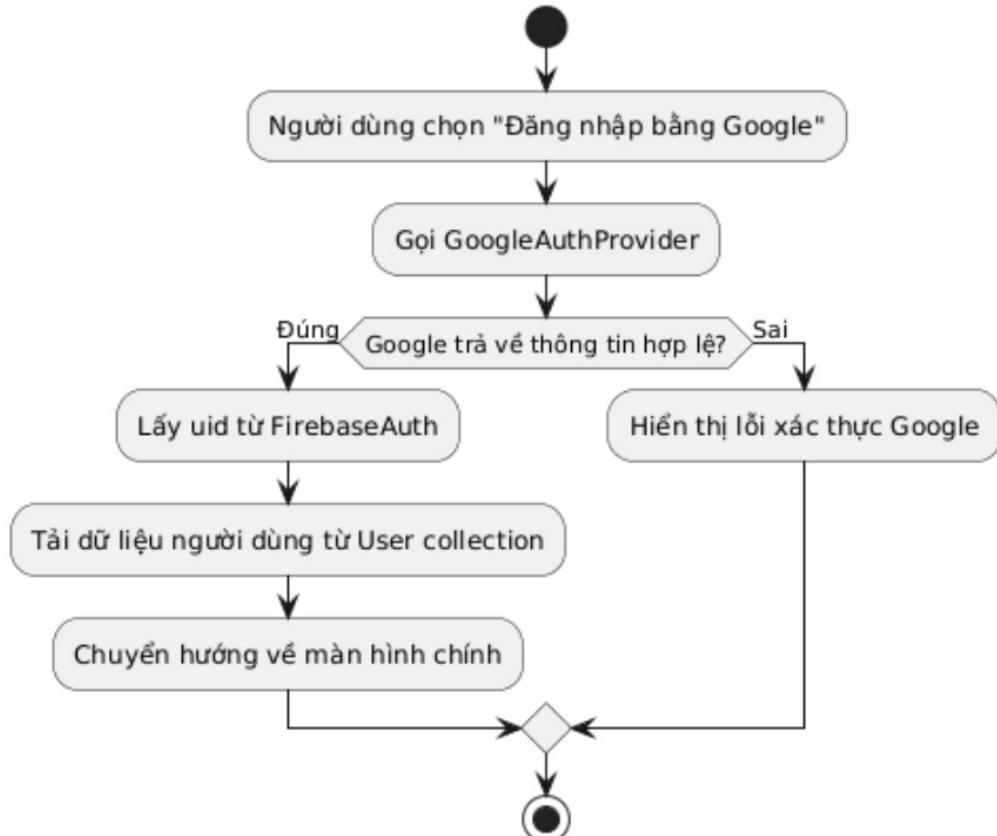
Hình 2.6: Biểu đồ hoạt động use case đăng ký

\* Use case Đăng nhập bằng Email



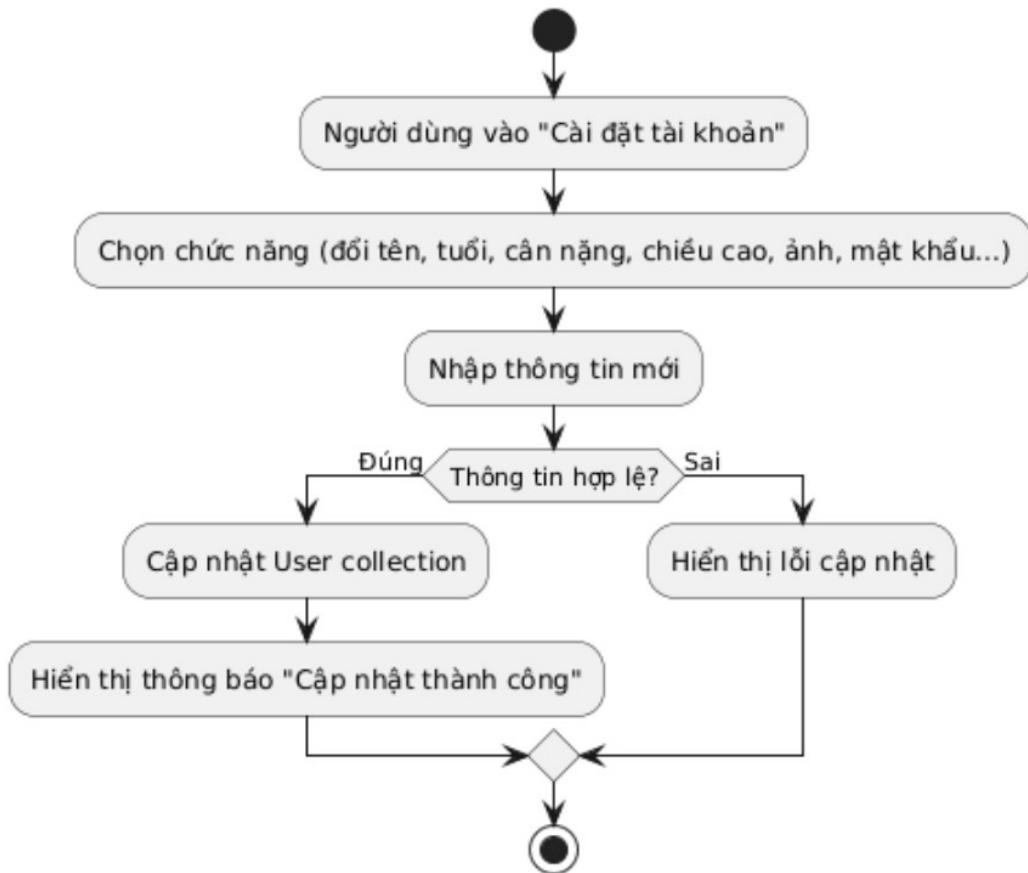
Hình 2.7: Biểu đồ hoạt động use case đăng nhập bằng email

\* Use case Đăng nhập bằng Google



Hình 2.8: Biểu đồ hoạt động use case đăng nhập bằng google

\* Use case Quản lý tài khoản cá nhân



Hình 2.9: Biểu đồ hoạt động use case quản lý tài khoản cá nhân

\* Use case Theo dõi dinh dưỡng



Hình 2.10: Biểu đồ hoạt động use case theo dõi dinh dưỡng

\* Use case Theo dõi nước uống



Hình 2.11: Biểu đồ hoạt động use case theo dõi nước uống

\* Use case Đặt mục tiêu sức khoẻ



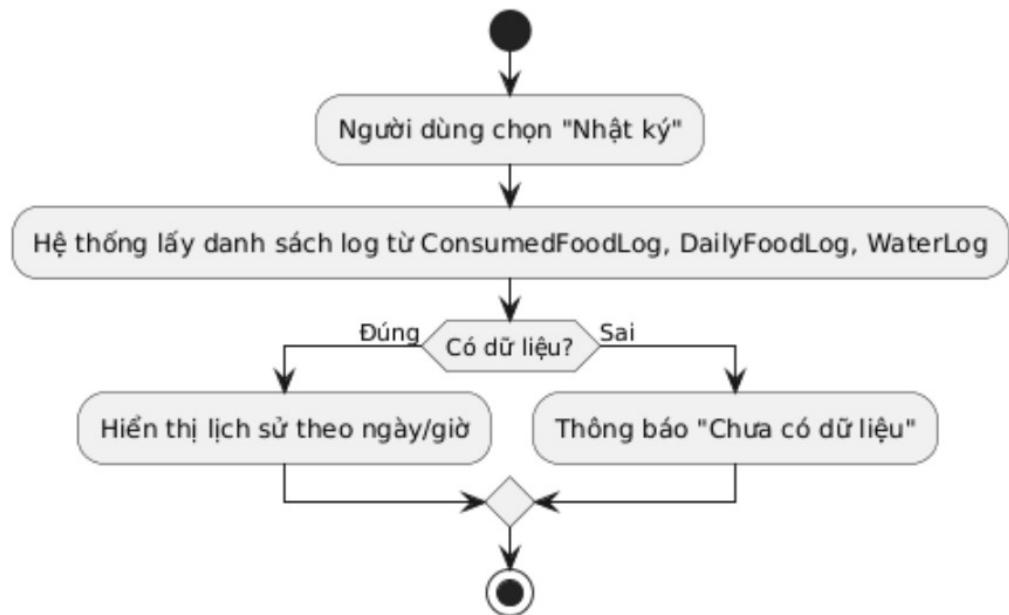
Hình 2.12: Biểu đồ hoạt động use case đặt mục tiêu sức khỏe

\* Use case Xem báo cáo sức khoẻ



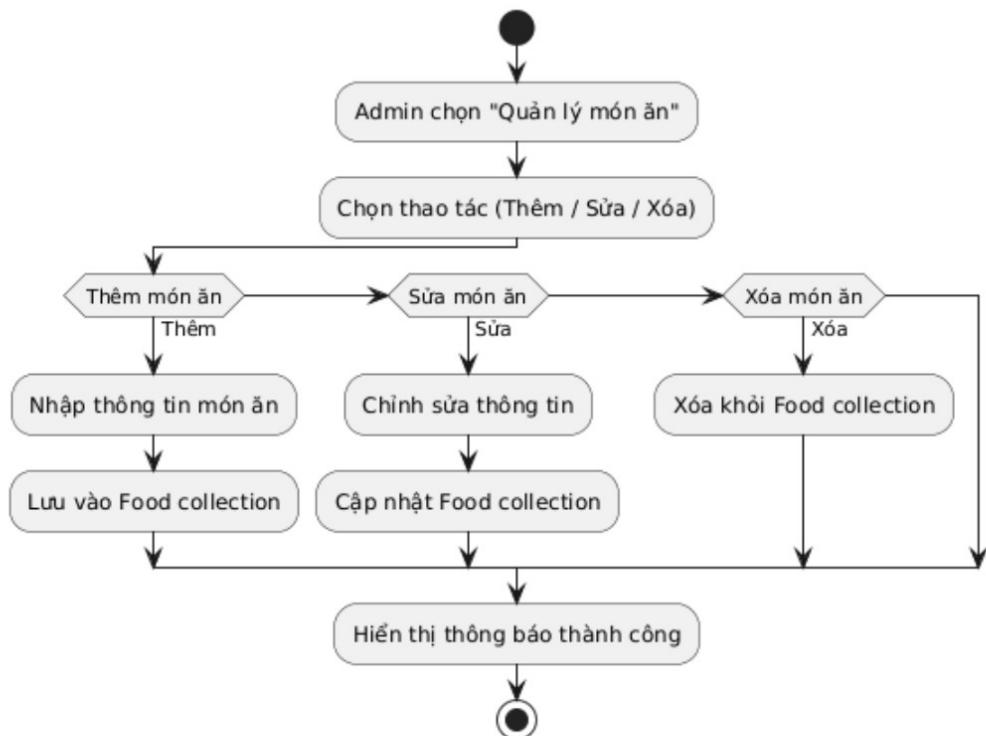
Hình 2.13: Biểu đồ hoạt động use case xem báo cáo sức khỏe

\* Use case Xem nhật ký



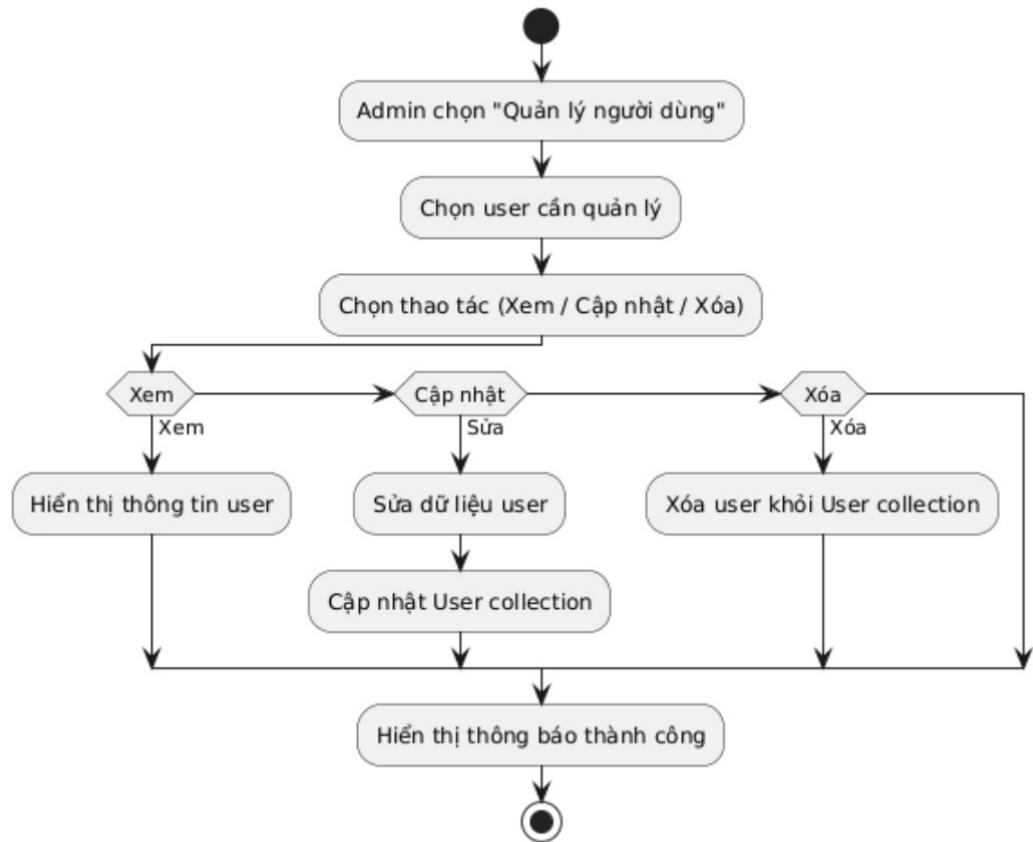
Hình 2.14: Biểu đồ hoạt động use case xem nhật ký

\* Use case Quản lý món ăn



Hình 2.15: Biểu đồ hoạt động use case quản lý món ăn

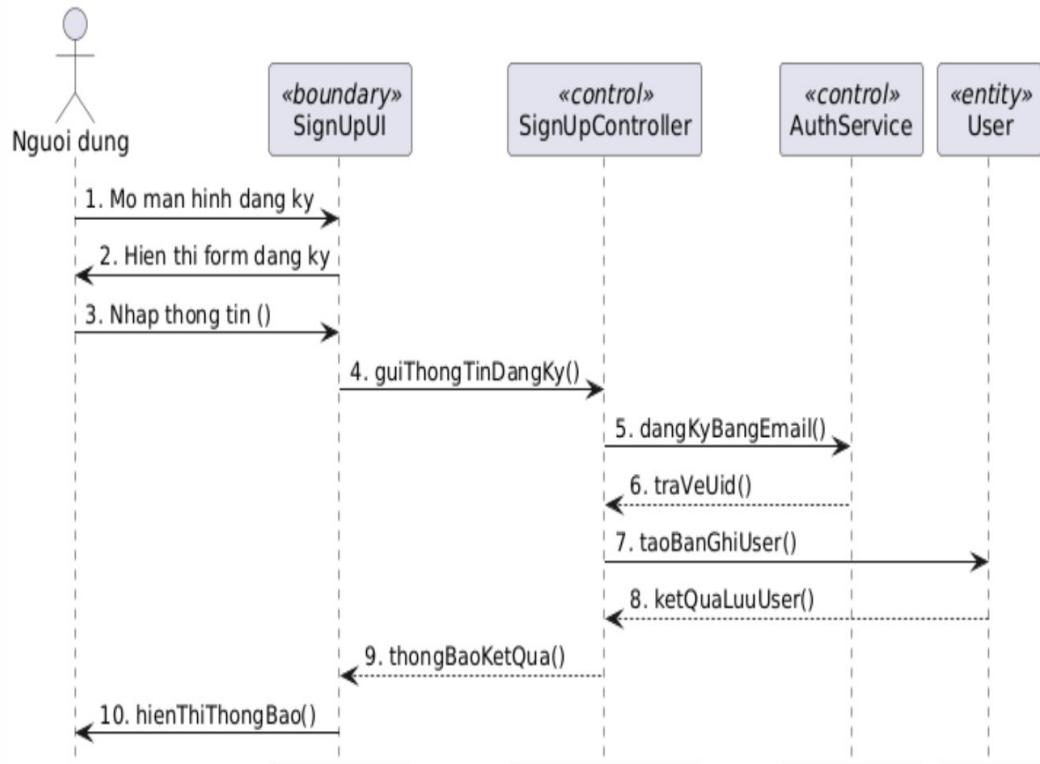
\* Use case Quản lý người dùng



Hình 2.16: Biểu đồ hoạt động use case quản lý người dùng

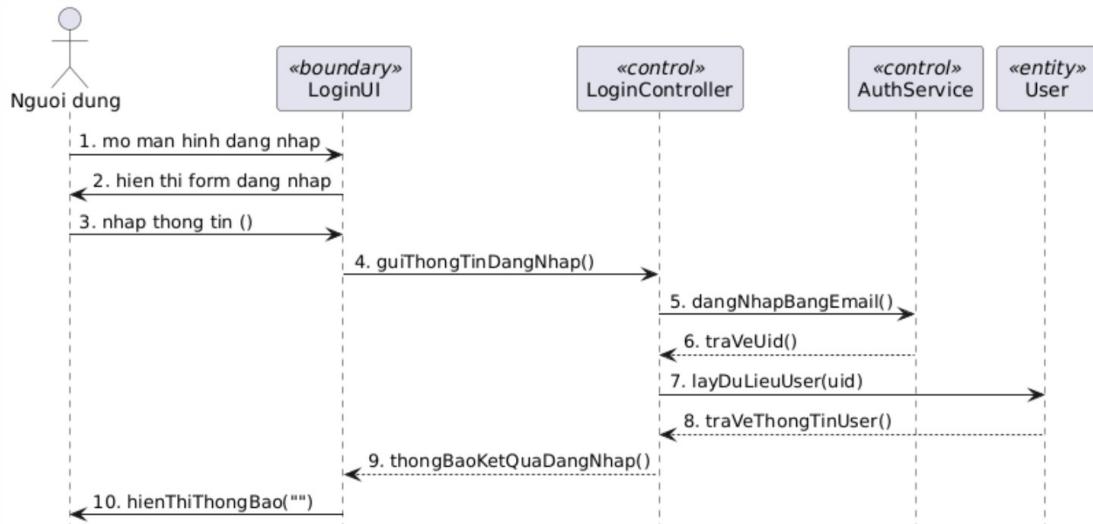
### 2.4.5. Biểu đồ trình tự

\* Biểu đồ trình tự use case đăng ký tài khoản bằng email



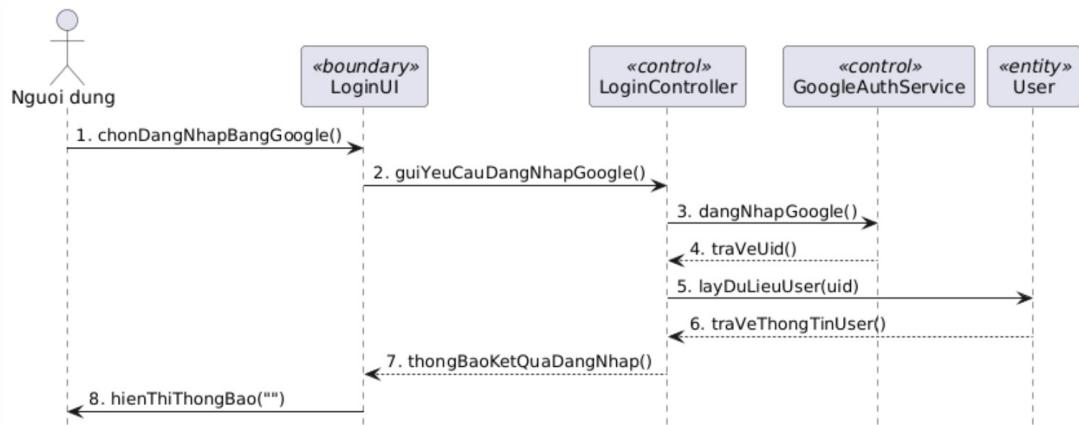
Hình 2.17: Biểu đồ trình tự use case đăng ký tài khoản bằng email

\* Biểu đồ trình tự use case Đăng nhập bằng Email



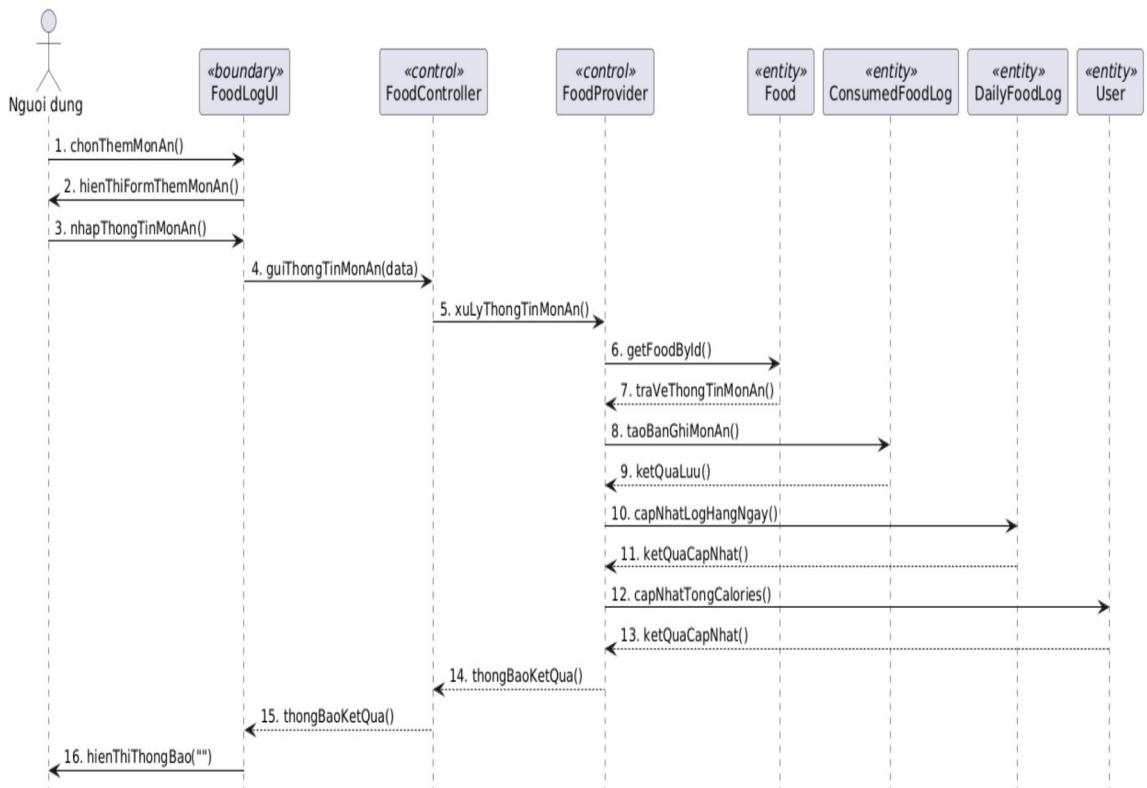
Hình 2.18: Biểu đồ phân tích use case Đăng nhập bằng Email

\*Biểu đồ trình tự use case Đăng nhập bằng Google



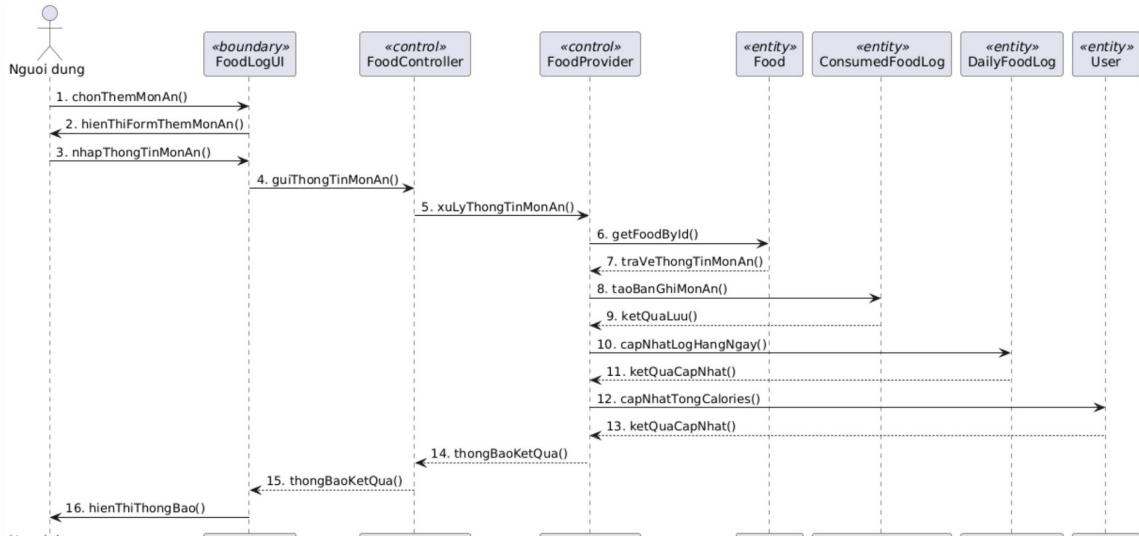
Hình 2.19: Biểu đồ trình tự use case Đăng nhập bằng Google

\*Biểu đồ trình tự use case Quản lý tài khoản cá nhân



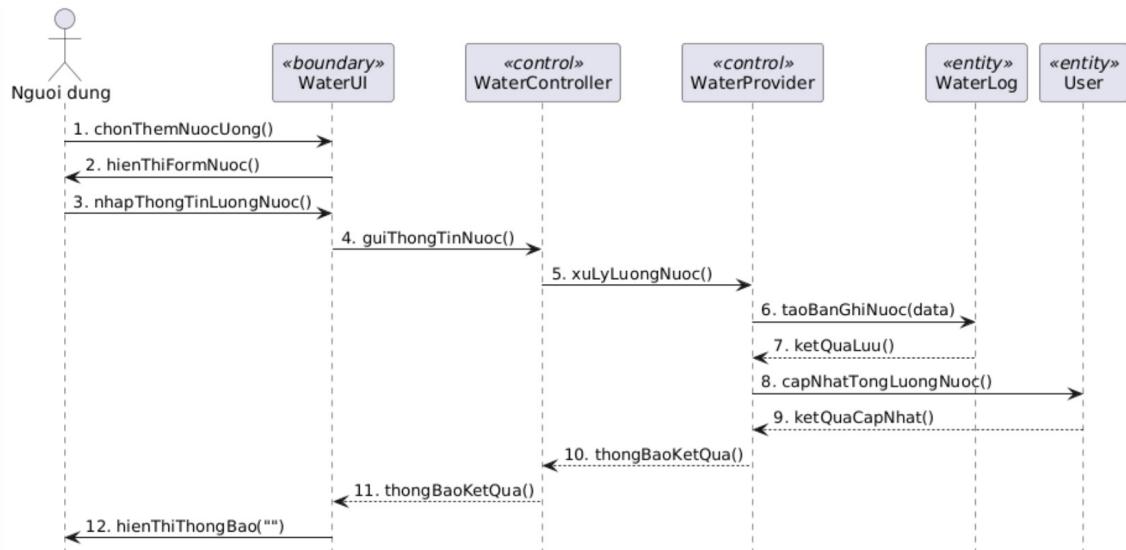
Hình 2.20: Biểu đồ trình tự use case Quản lý tài khoản cá nhân

\*Biểu đồ trình tự use case Theo dõi dinh dưỡng



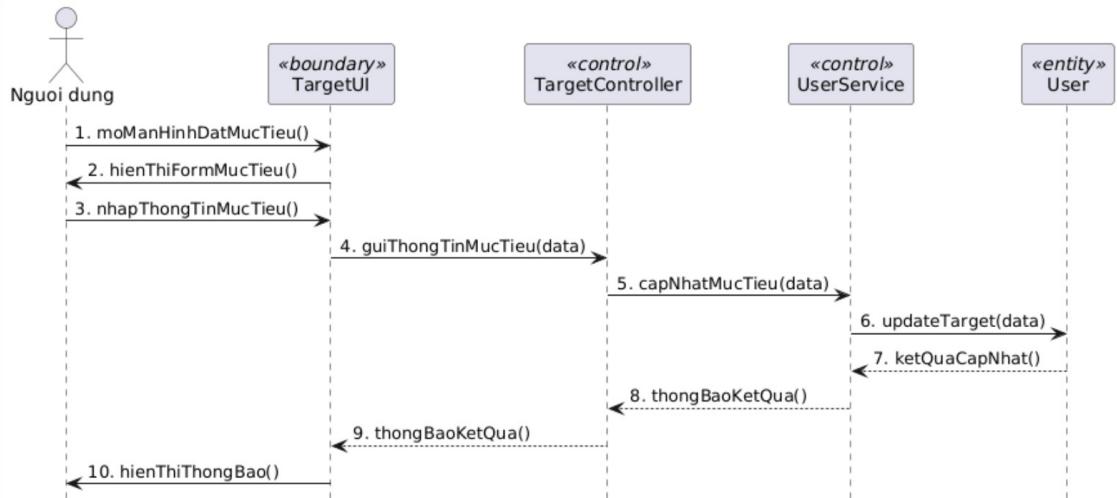
Hình 2.21: Biểu đồ trình tự use case Theo dõi dinh dưỡng

\*Biểu đồ trình tự use case Theo dõi nước uống



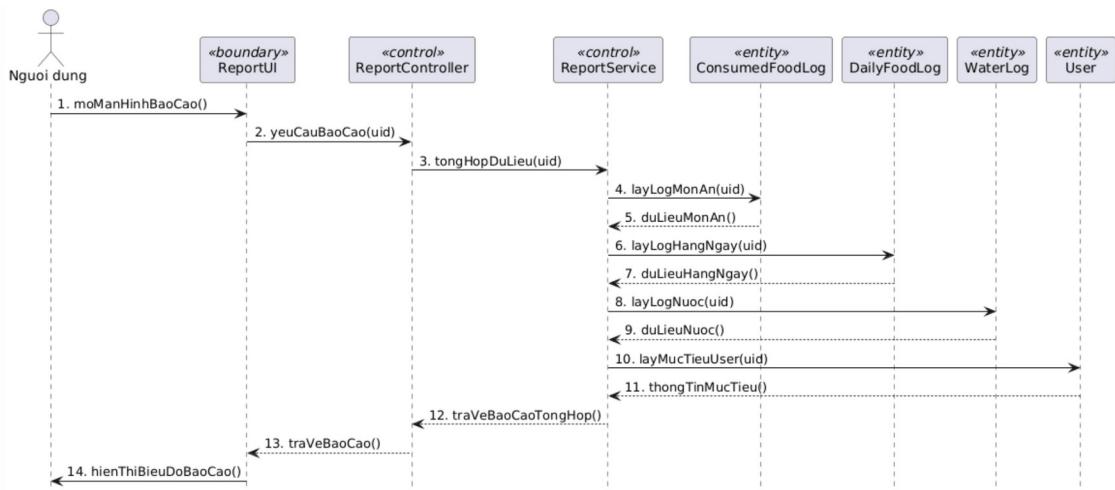
Hình 2.22: Biểu đồ trình tự use case Theo dõi nước uống

\*Biểu đồ trình tự use case Đặt mục tiêu dinh dưỡng



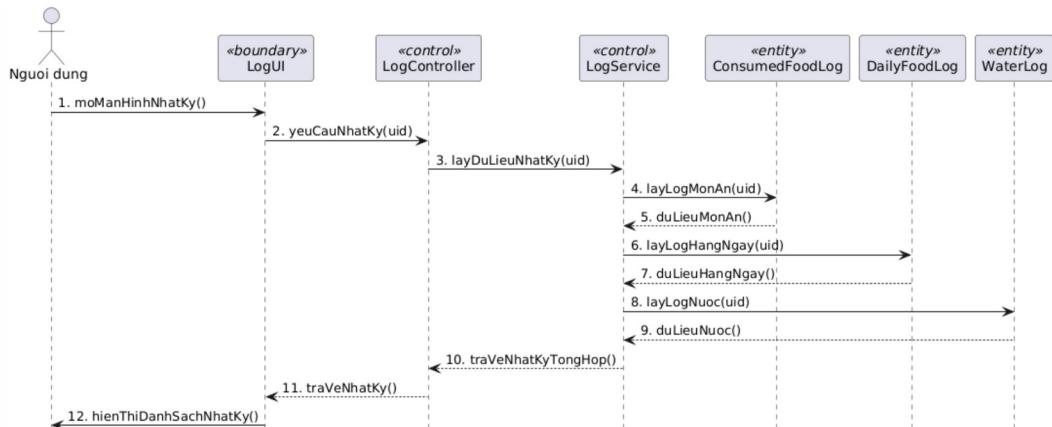
Hình 2.23: Biểu đồ trình tự use case đặt mục tiêu dinh dưỡng

\*Biểu đồ trình tự use case Xem báo cáo sinh dưỡng



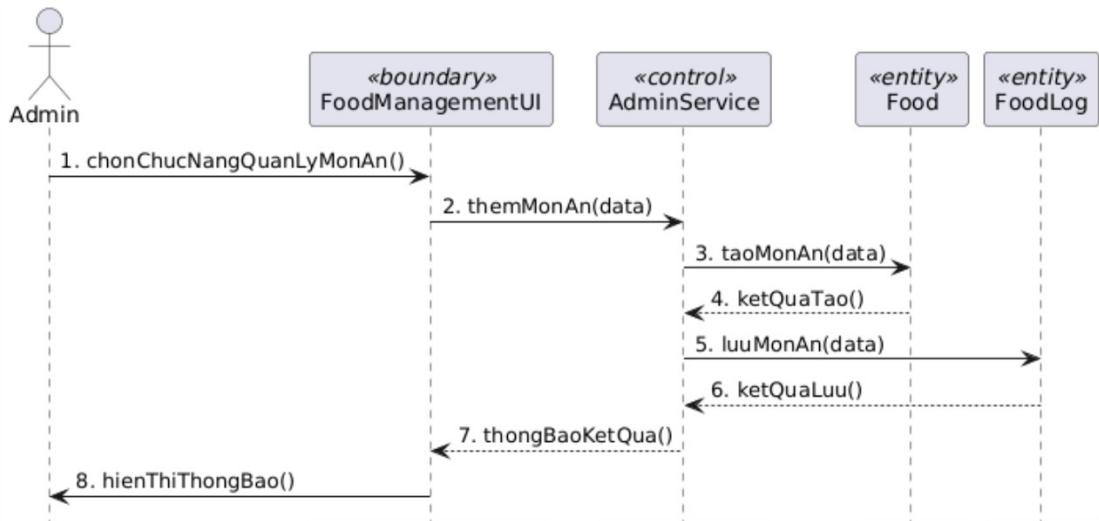
Hình 2.24: Biểu đồ trình tự use case Xem báo cáo sinh dưỡng

\*Biểu đồ trình tự use case Xem nhật ký



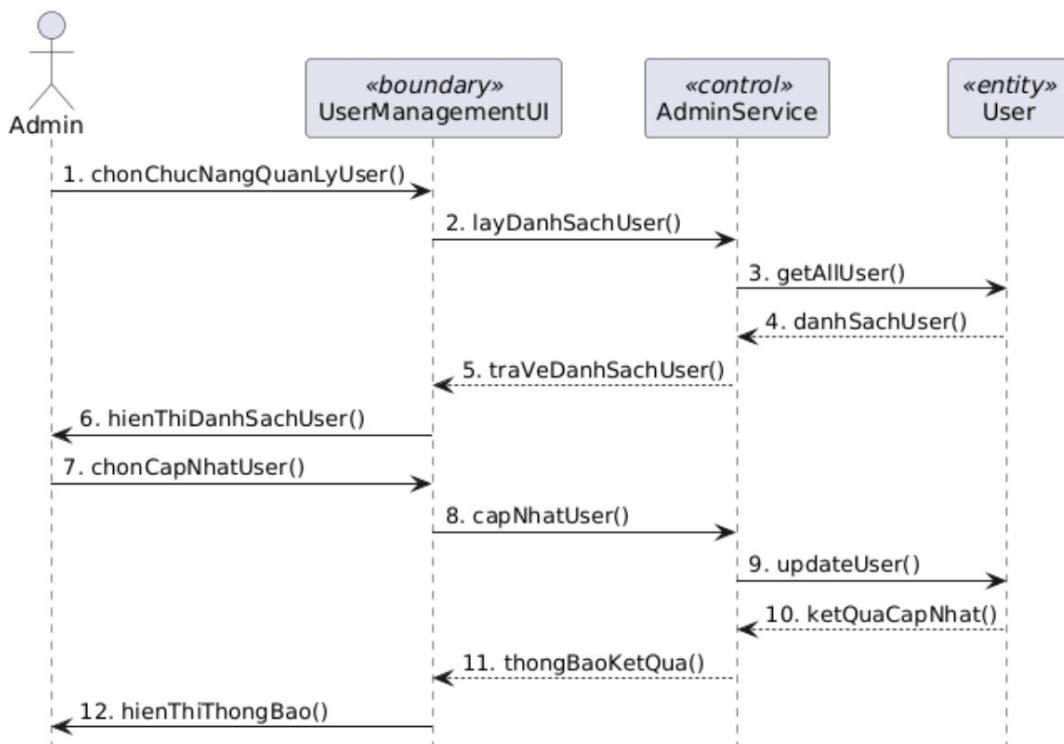
Hình 2.25: Biểu đồ trình tự use case Xem nhật ký

\*Biểu đồ trình tự use case Quản lý món ăn



Hình 2.26: Biểu đồ trình tự use case Quản lý món ăn

\*Biểu đồ trình tự use case Quản lý người dùng



Hình 2.27: Biểu đồ trình tự use case Quản lý người dùng

#### 2.4.6 Các yêu cầu về dữ liệu

Cơ sở dữ liệu sử dụng Firebase FireStore là NoSQL. Mỗi một bản ghi (record) trong FireStore sẽ tương ứng với một document. Tập hợp các document sẽ tạo thành một collection.

### \*Thiết kế các collection

a) Quản lý danh sách món ăn

*Bảng 2.1: Food*

Tên trường	Kiểu dữ liệu
Calories	Number
carbs	Number
description	String
fat	Number
foodName	String
foodWeigh	Number
ImageUrl	String
protein	Number

b) Quản lý danh sách người dùng

*Bảng 2.2: User*

Tên trường	Kiểu dữ liệu
age	Number
bmi	Number
email	String
foodlog	String
Height	Number
id	String
lastReset	Timestamp
Name	String
profileImageUrl	String
role	String
targetCalories	Number
totalCalories	Number
totalWaterIntake	Number

waterLog	Map
weight	Number
consumedFoodLog	Collection
dailyFoodLog	Collection

c) Quản lý món ăn tiêu thụ

Bảng 2.3: ConsumedFoodLog

Tên trường	Kiểu dữ liệu
calories	Number
carbs	Number
fat	Number
foodName	String
foodWight	Number
mealType	String
protein	Number
timestamp	timestamp

d) Quản lý món ăn theo ngày

Bảng 2.4: DailyFoodLog

Tên trường	Kiểu dữ liệu
calories	Number
carbs	Number
fat	Number
foodName	String
foodWight	Number
mealType	String
protein	Number
timestamp	timestamp

## e) Quản lý nhật ký

Bảng 2.5: Foodlog

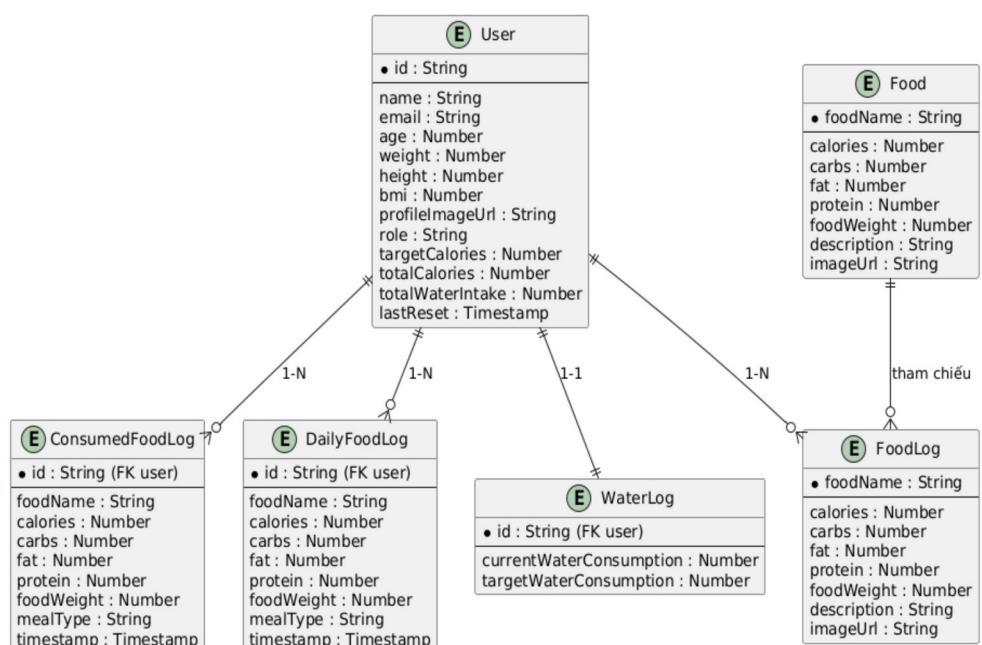
Tên trường	Kiểu dữ liệu
calories	Number
carbs	Number
description	String
fat	Number
foodName	String
foodWight	Number
ImageUrl	String
protein	Number

## f) Quản lý nước uống

Bảng 2.6: WaterLog

Tên trường	Kiểu dữ liệu
currentWaterConsumption	Number
targetWaterConsumption	Number

## 2.6.7. Biểu đồ ERD



Hình 2.28: Biểu đồ ERD

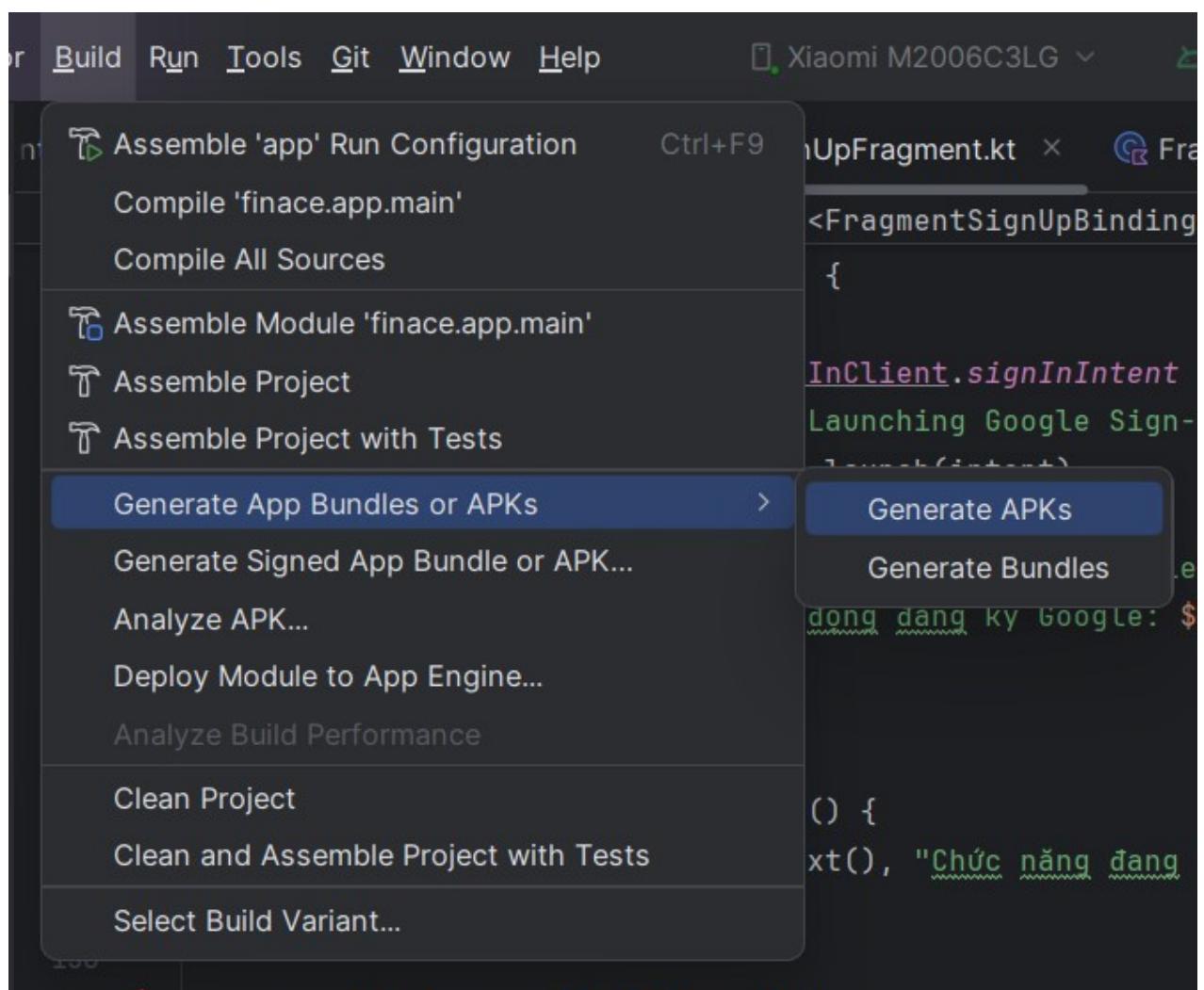
## CHƯƠNG 3: CÀI ĐẶT VÀ KIỂM THỬ CHƯƠNG TRÌNH

### 3.1 Cài đặt môi trường

#### 3.1.1 Các bước cài đặt chương trình

##### Bước 1: Build file APK từ Android Studio

- Mở project trong Android Studio.
- Trên thanh menu, chọn: Build → Build Bundle(s) / APK(s) → Build APK(s).



Hình 3.29: Build file APK bằng Android Studio

- Chờ quá trình build hoàn tất. Sau khi hoàn tất, một thông báo sẽ hiện ra ở góc dưới bên phải cho phép bạn định vị file `apk` vừa tạo.
- Tìm file APK tại đường dẫn: `.../app/build/outputs/apk/debug/app-

debug.apk` hoặc tương đương tùy theo cấu hình project.

### **Bước 2:** Cho phép cài đặt từ nguồn không xác định trên thiết bị Android

- Trên điện thoại Android, mở "Cài đặt" (Settings).
- Vào mục "Bảo mật và quyền riêng tư" (hoặc "An toàn và bảo mật" tùy phiên bản hệ điều hành).
- Chọn "Cài đặt ứng dụng từ nguồn không xác định".
- Bật quyền cho phép trình quản lý tệp (hoặc trình duyệt, ứng dụng dùng để mở file APK) được cài đặt ứng dụng từ nguồn bên ngoài Google Play.

### **Bước 3:** Cài đặt ứng dụng

- Chép file APK sang điện thoại hoặc gửi qua email/Zalo/Google Drive, sau đó tải về điện thoại.
- Truy cập file APK từ trình quản lý tệp trên điện thoại và chọn "Cài đặt" (Install).
- Hệ thống sẽ tự động cài đặt ứng dụng. Sau khi hoàn tất, bạn có thể mở ứng dụng trực tiếp từ màn hình chính.

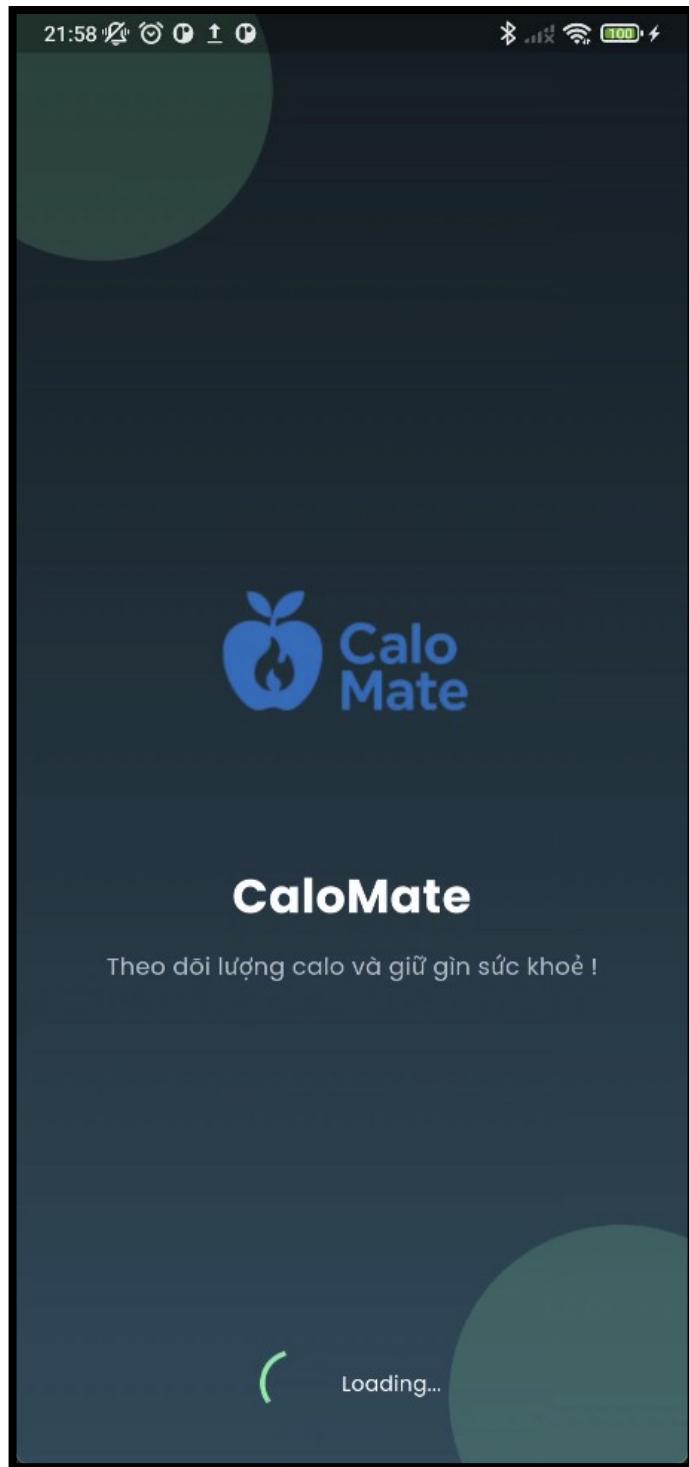
#### **3.1.2 Lưu ý khi cài đặt chương trình**

- Nếu sử dụng thiết bị Android từ Android 8.0 (Oreo) trở lên, quyền “cài đặt từ nguồn không xác định” sẽ được cấp riêng cho từng ứng dụng (ví dụ: Chrome, File Manager...).
- Trong quá trình cài đặt, có thể cần xác nhận thêm bằng vân tay, mã PIN hoặc mật khẩu bảo mật nếu thiết bị có thiết lập bảo mật hệ thống.
- Nếu thiết bị báo lỗi “Ứng dụng không được cài đặt”, hãy kiểm tra xem phiên bản APK có tương thích với phiên bản Android hiện tại hay không, hoặc thử gỡ bản cũ trước khi cài lại.
- Một số dòng máy Android có thể ẩn phần tùy chọn cài đặt từ nguồn không xác định – có thể tìm bằng cách truy cập Cài đặt → Tìm kiếm → "Không rõ nguồn gốc" để bật tùy chọn nhanh.

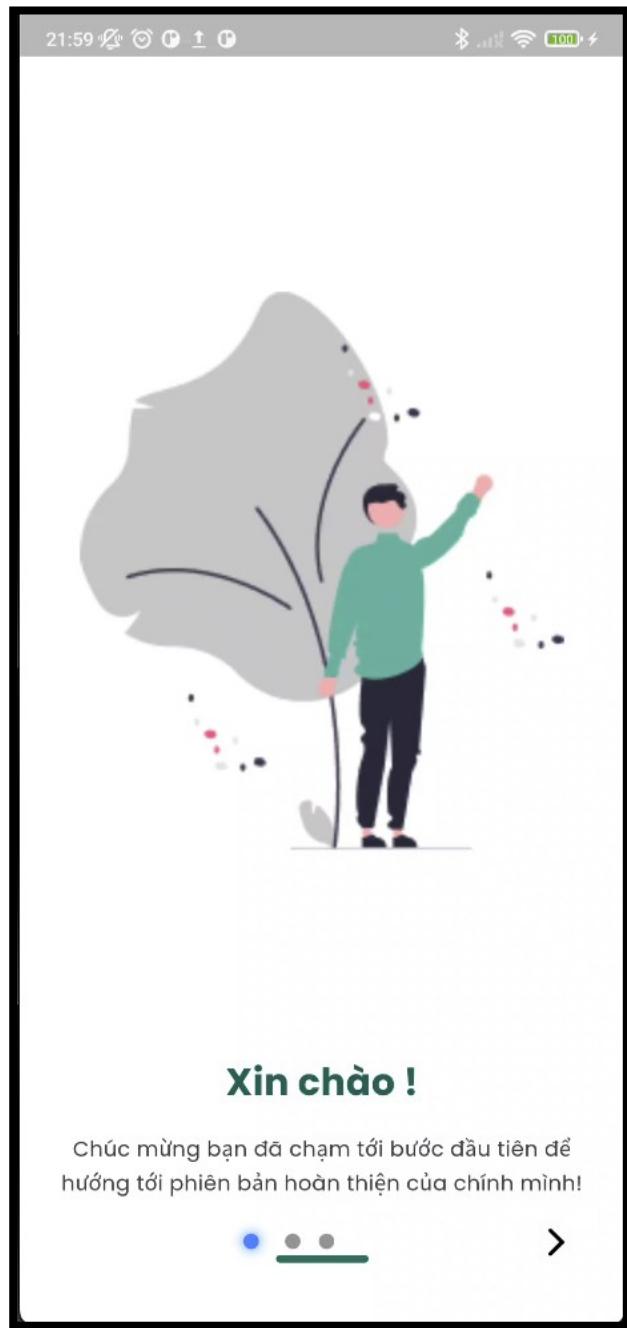
- Sau khi cài đặt thành công, ứng dụng sẽ hiển thị biểu tượng trên màn hình chính. Nếu không thấy, nên khởi động lại thiết bị hoặc kiểm tra trong Danh sách ứng dụng đã cài đặt.

### 3.1.3 Giao diện hệ thống

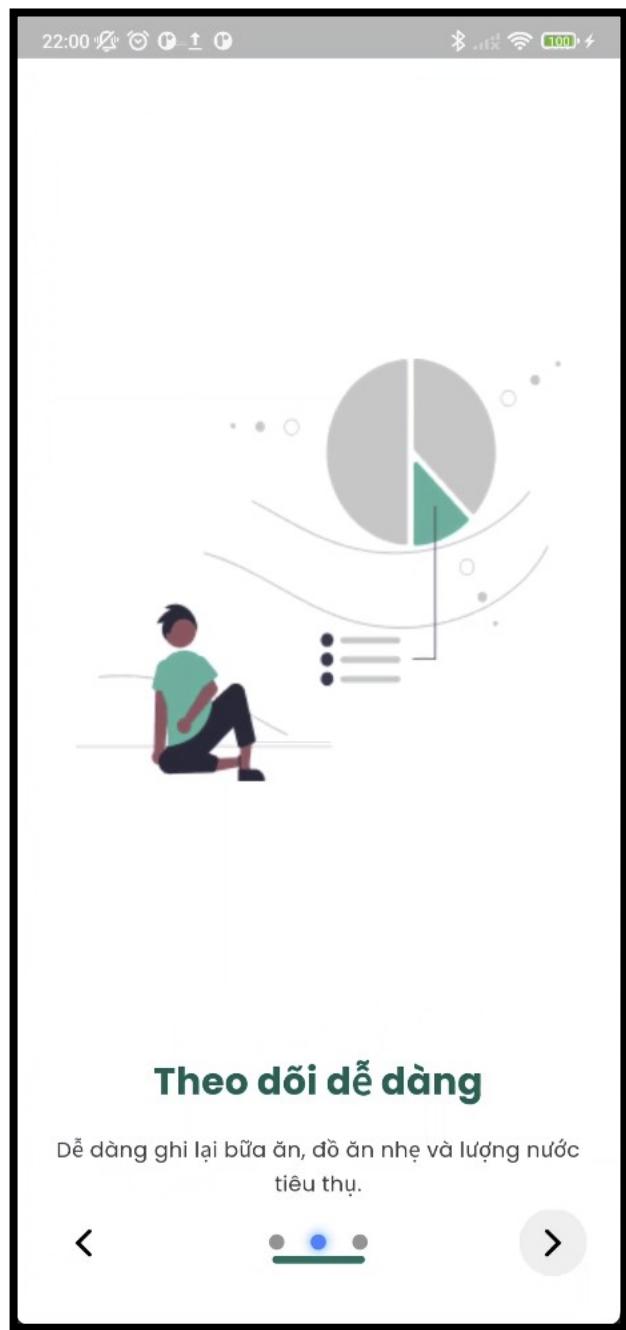
- \* Giao diện bắt đầu



Hình 3.30: Giao diện màn hình Splash



Hình 3.31: Giao diện màn hình bắt đầu I



Hình 3.32: Giao diện màn hình bắt đầu 2



Hình 3.33: Giao diện màn hình bắt đầu 3

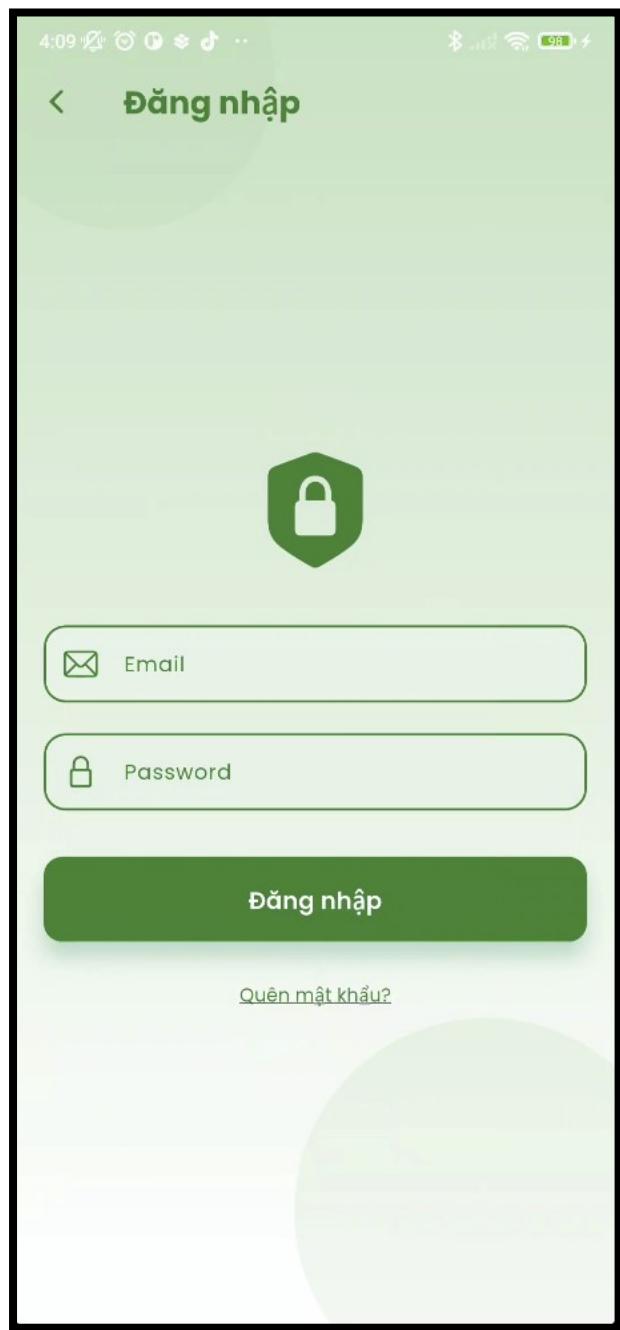


Hình 3.34: Giao diện màn hình chào mừng 4

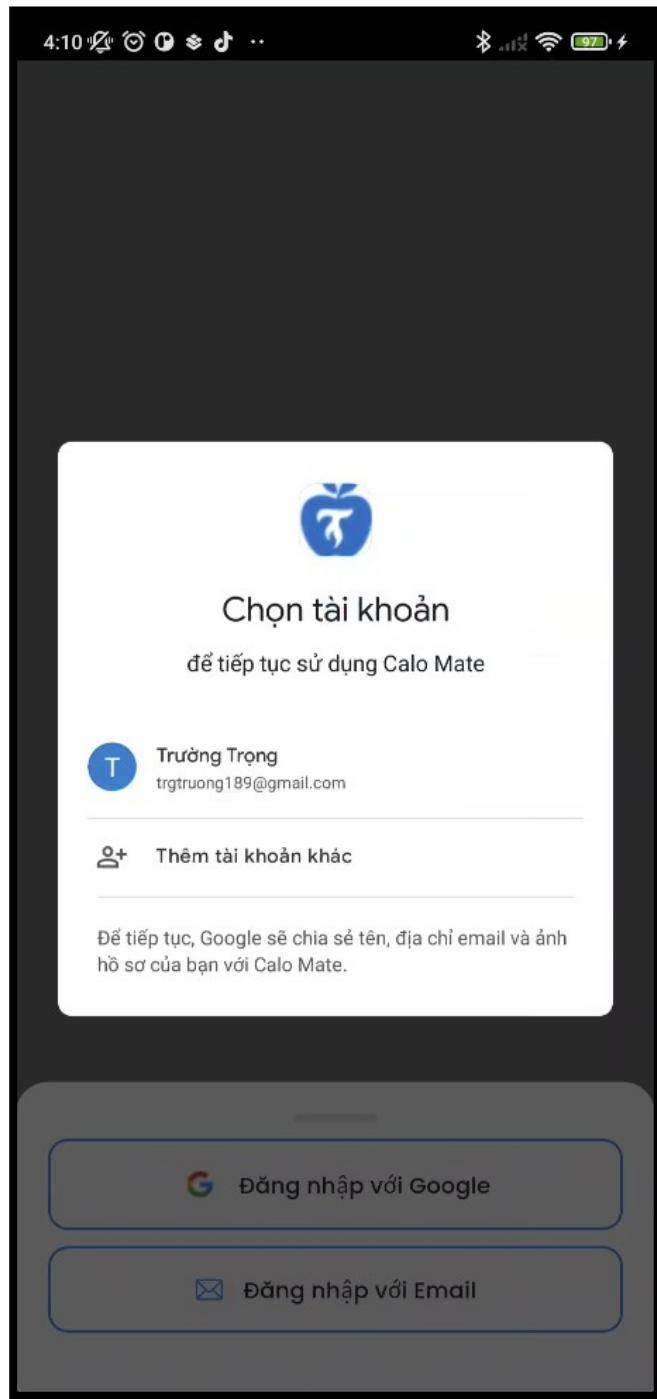
\* Giao diện đăng nhập



Hình 3.35: Giao diện đăng nhập



Hình 3.36: Giao diện đăng nhập bằng email



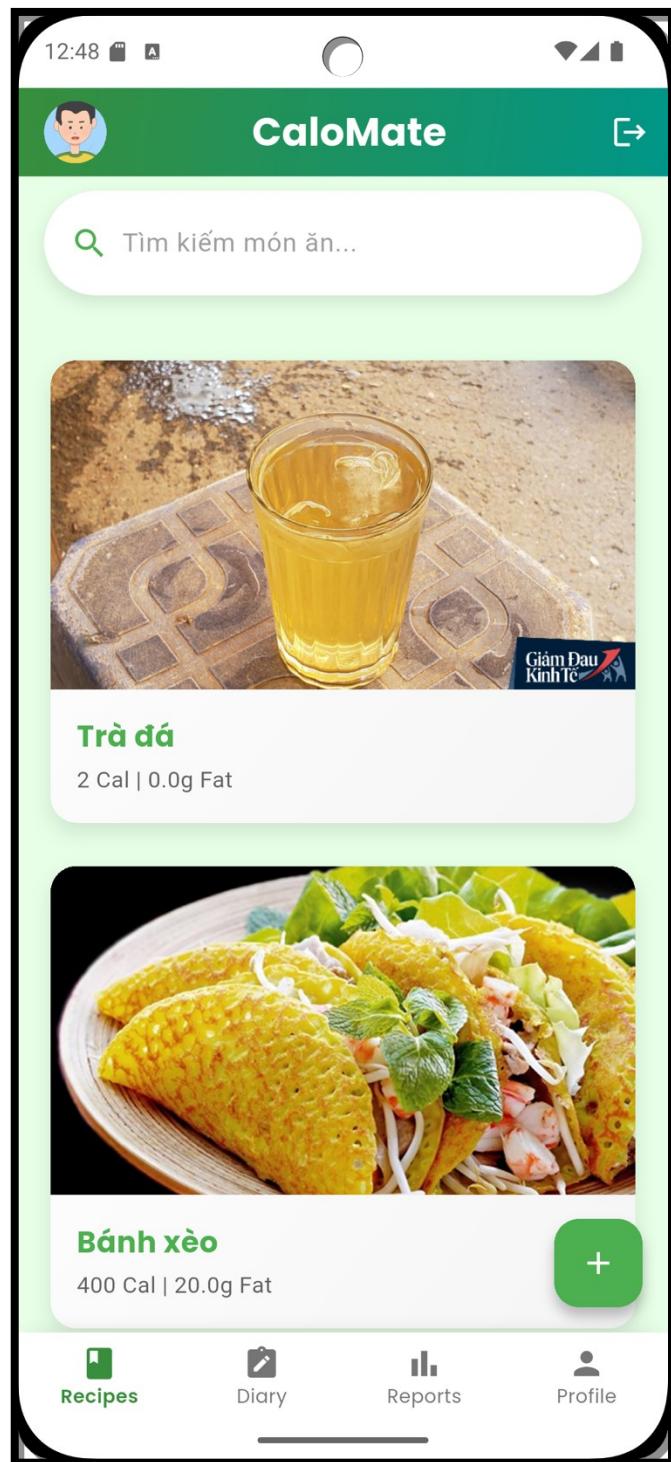
Hình 3.37: Giao diện đăng nhập bằng google

\* Giao diện đăng ký



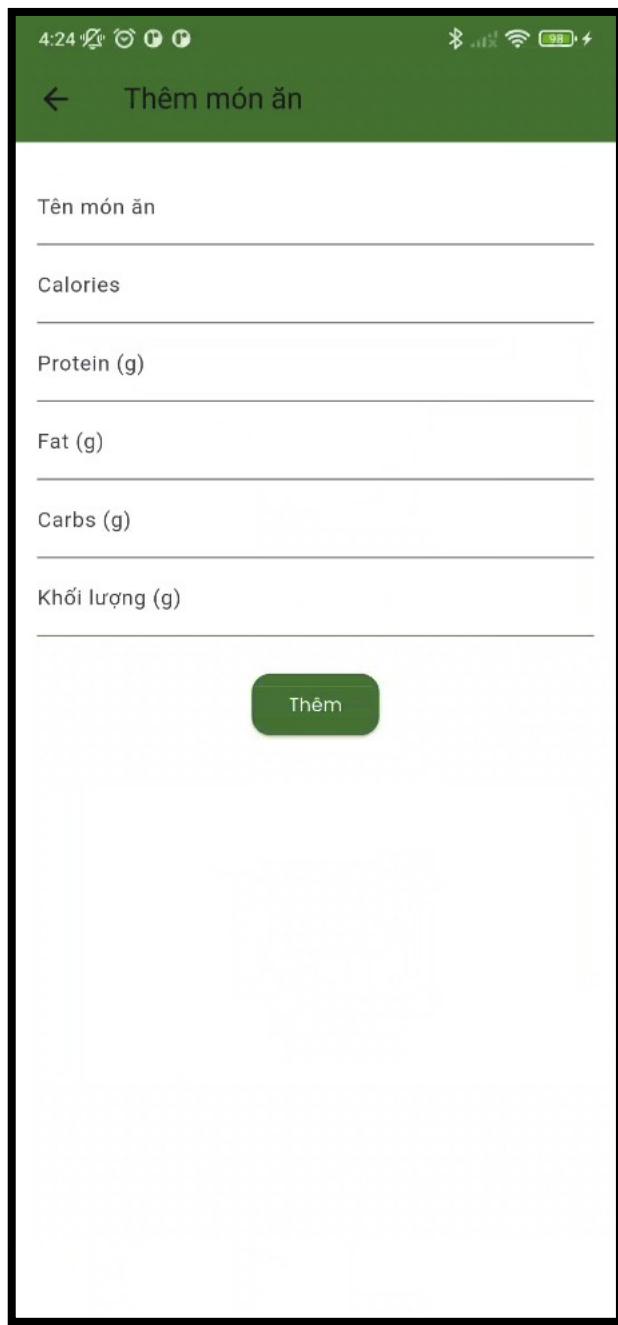
Hình 3.38: Giao diện đăng ký

\* Giao diện Recipes



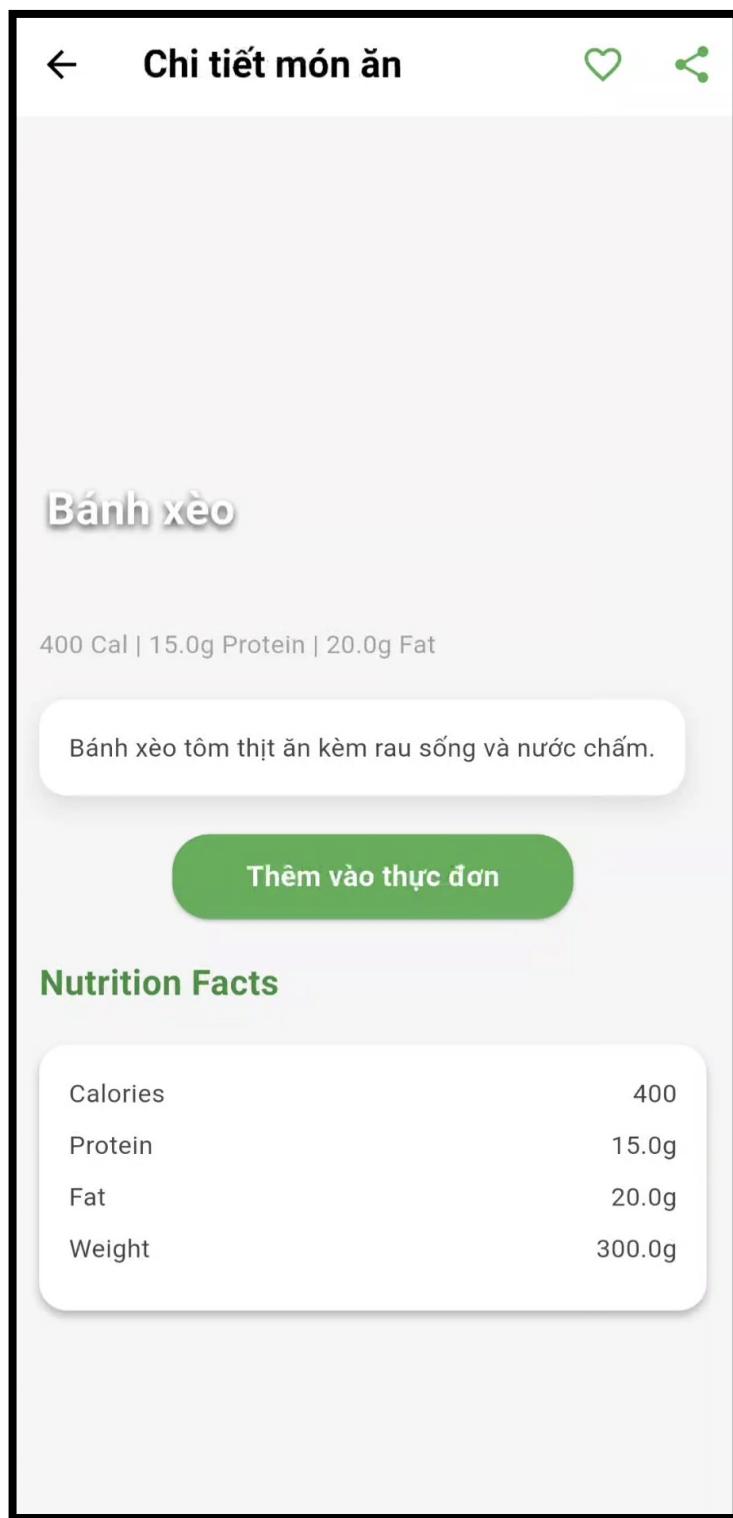
Hình 3.39: Giao diện Recipes

\* Giao diện thêm món ăn



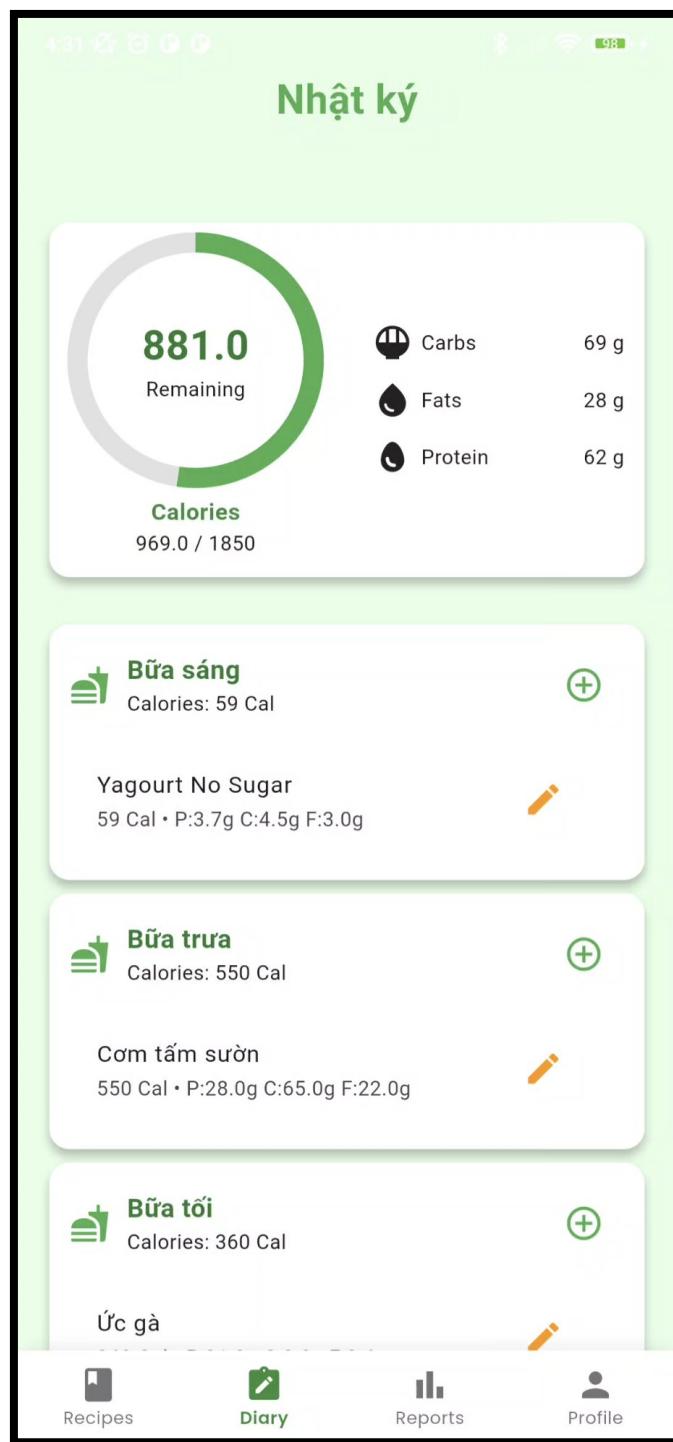
Hình 3.40: Giao diện thêm món ăn

\* Giao diện chi tiết món ăn



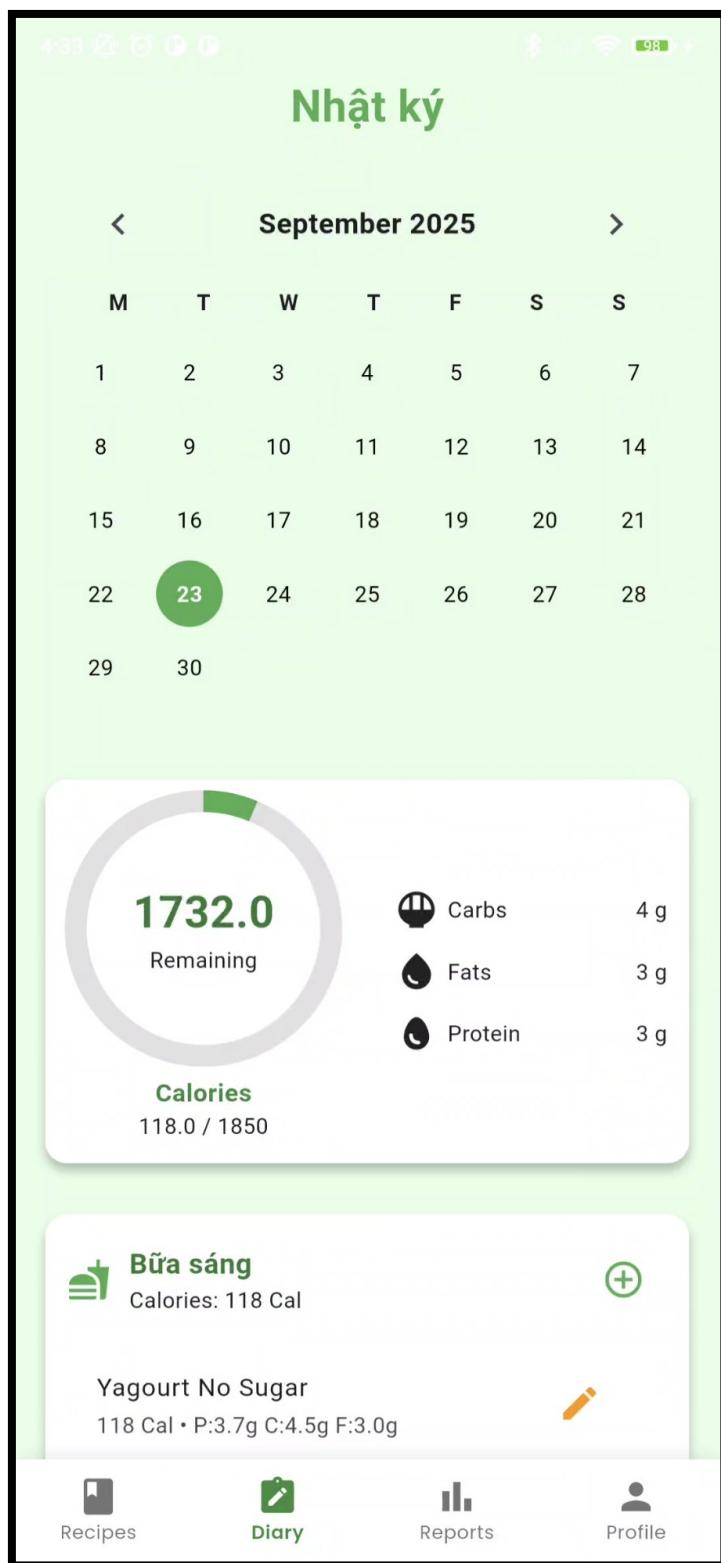
Hình 3.41: Giao diện chi tiết món ăn

\* Giao diện nhật ký dinh dưỡng



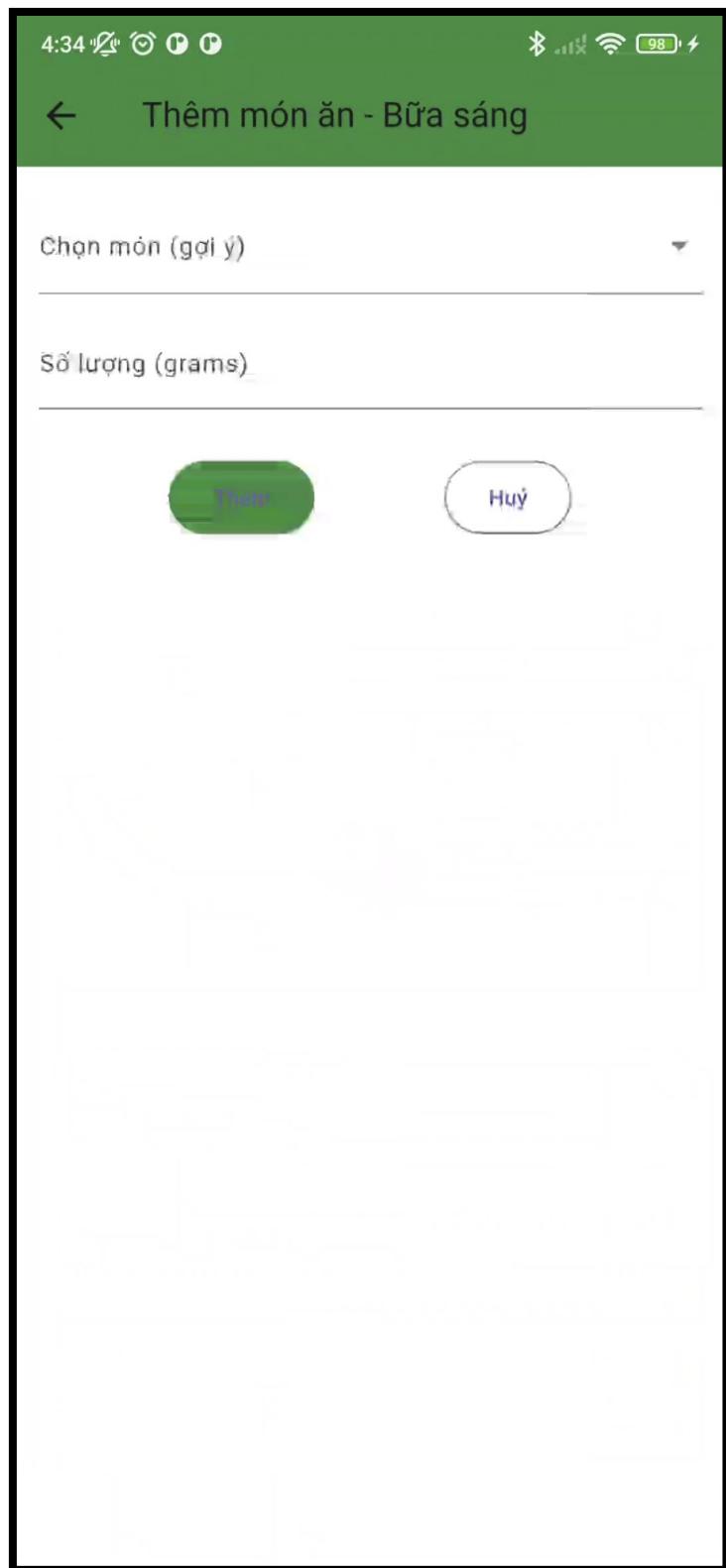
Hình 3.42: Giao diện nhật ký dinh dưỡng

\* Giao diện xem nhật ký theo ngày



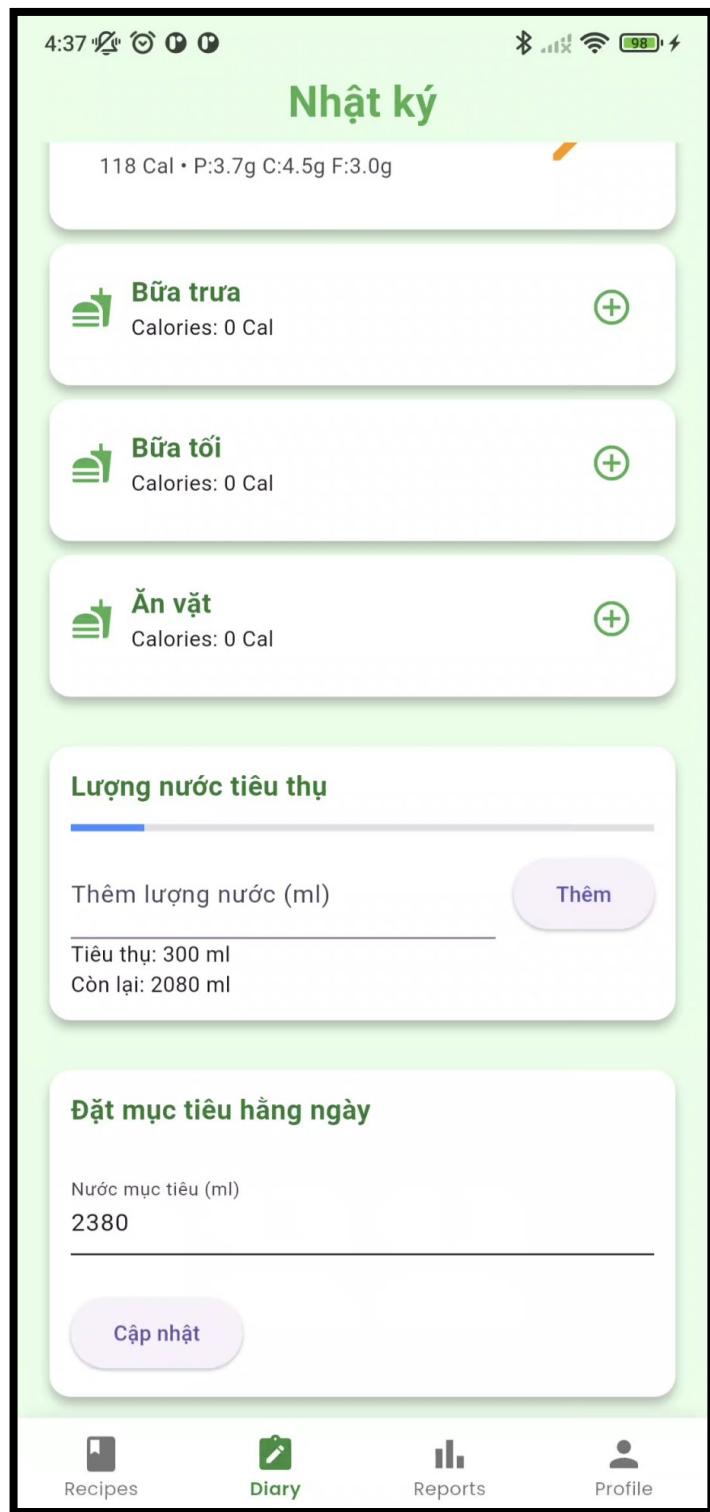
Hình 3.43: Giao diện xem nhật ký theo ngày

\* Giao diện thêm bữa ăn



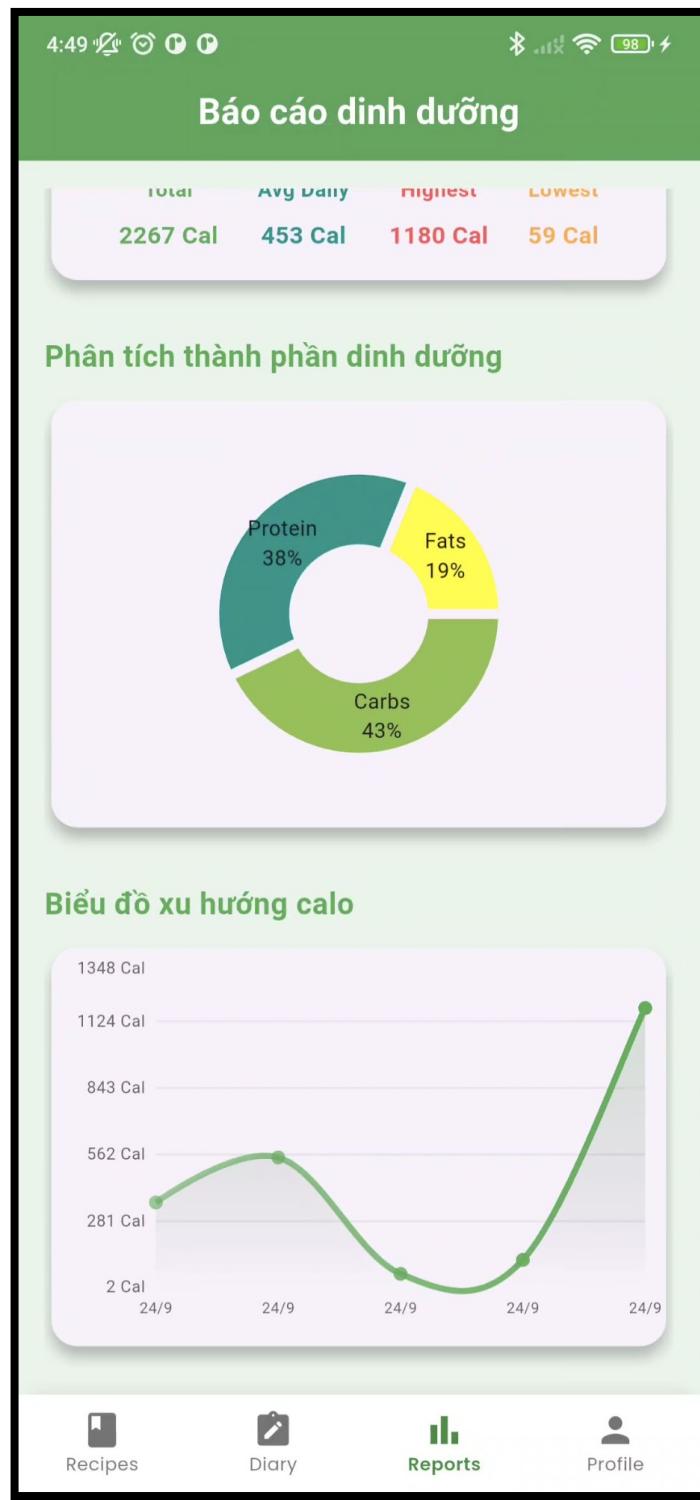
Hình 3.44: Giao diện thêm bữa ăn

\* Giao diện quản lý lượng nước



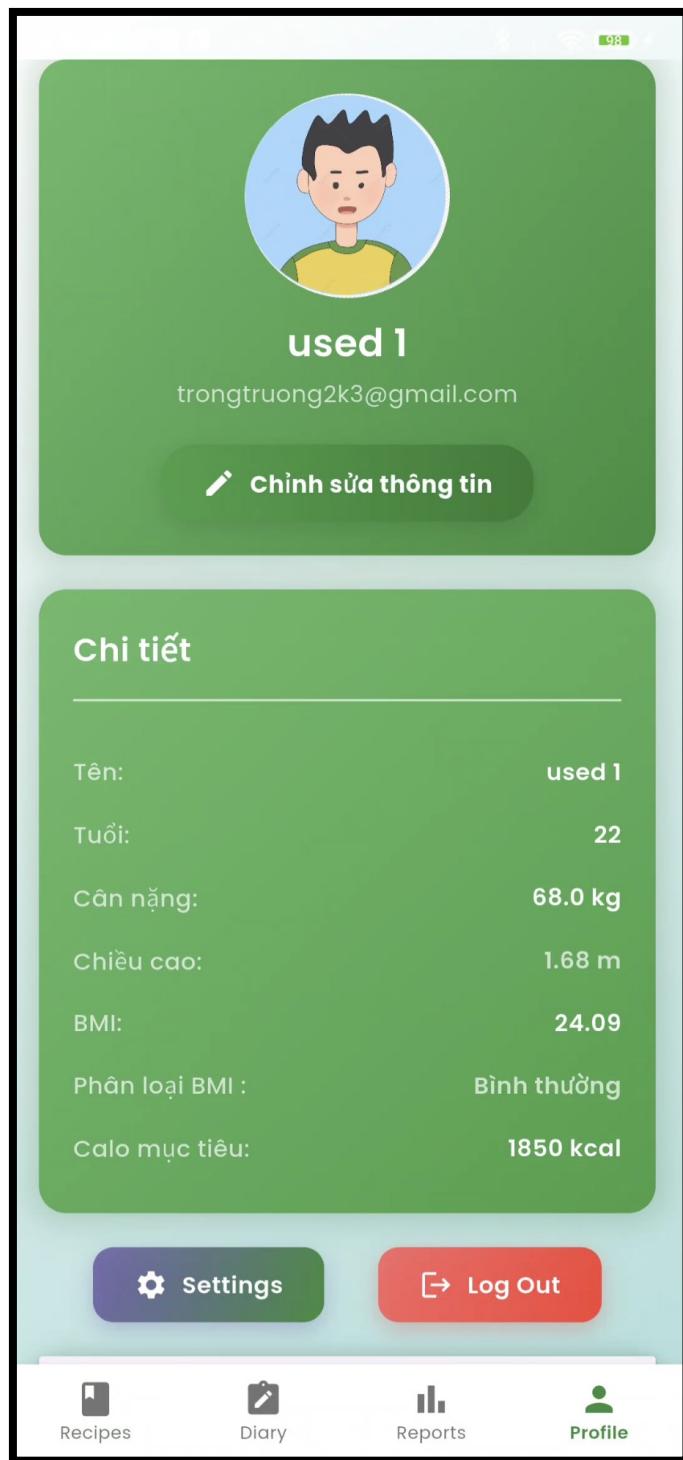
Hình 3.45: Giao diện quản lý lượng nước

\* Giao diện báo cáo dinh dưỡng



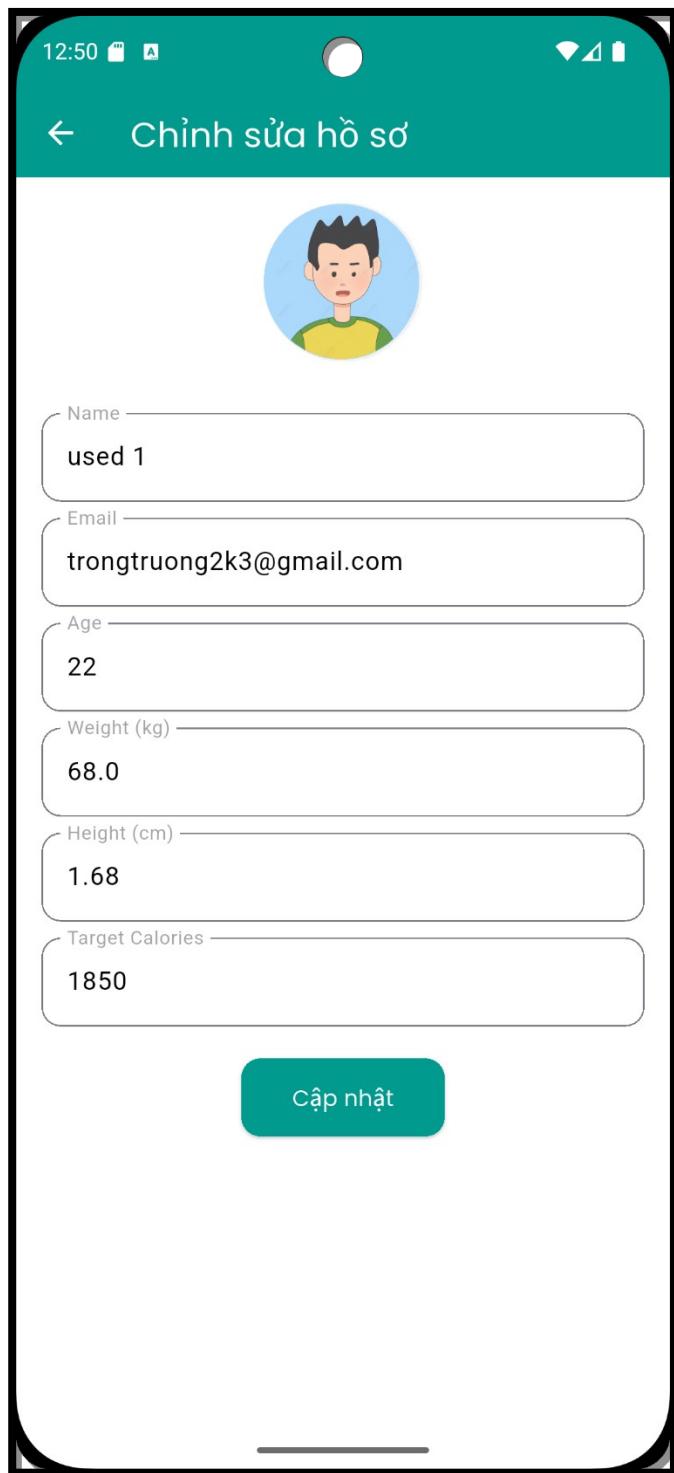
Hình 3.46: Giao diện báo cáo dinh dưỡng

\* Giao diện tài khoản



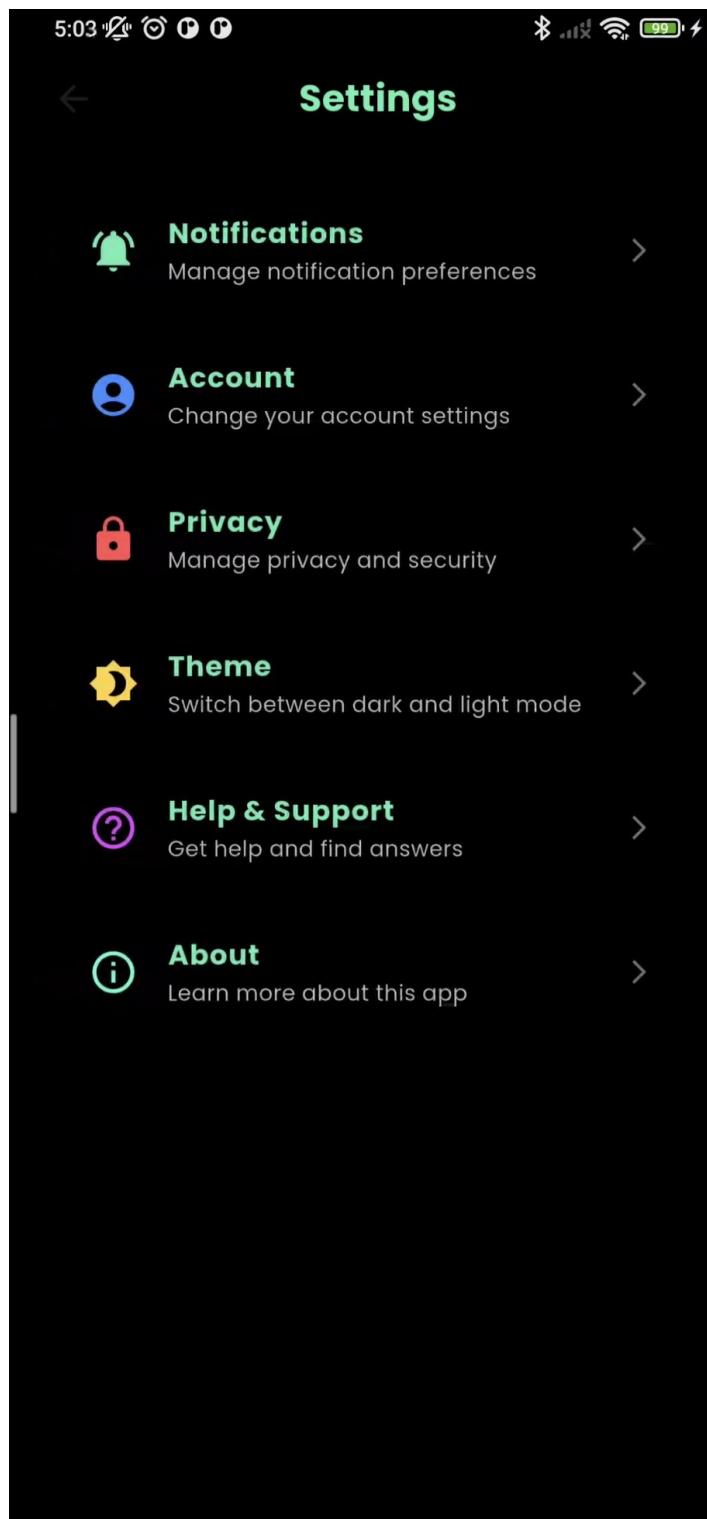
Hình 3.47: Giao diện tài khoản

\* Giao diện chỉnh sửa hồ sơ



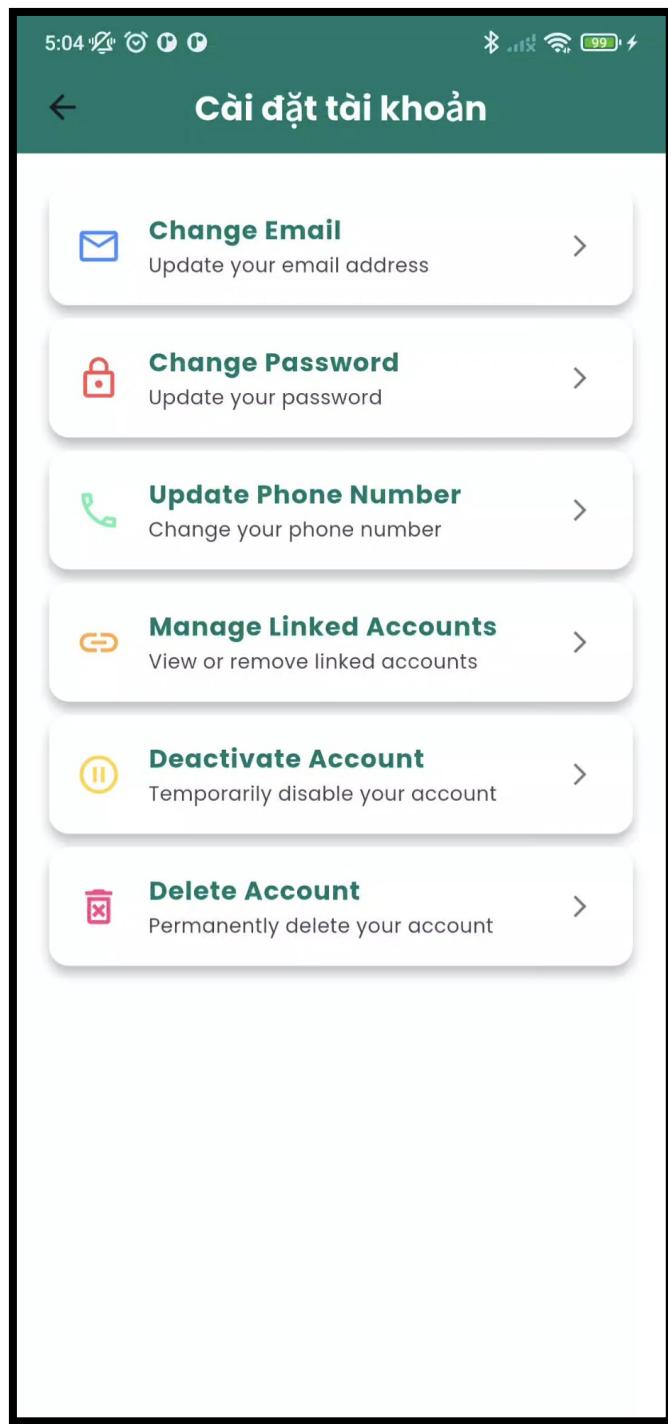
Hình 3.48: Giao diện chỉnh sửa hồ sơ

\* Giao diện cài đặt



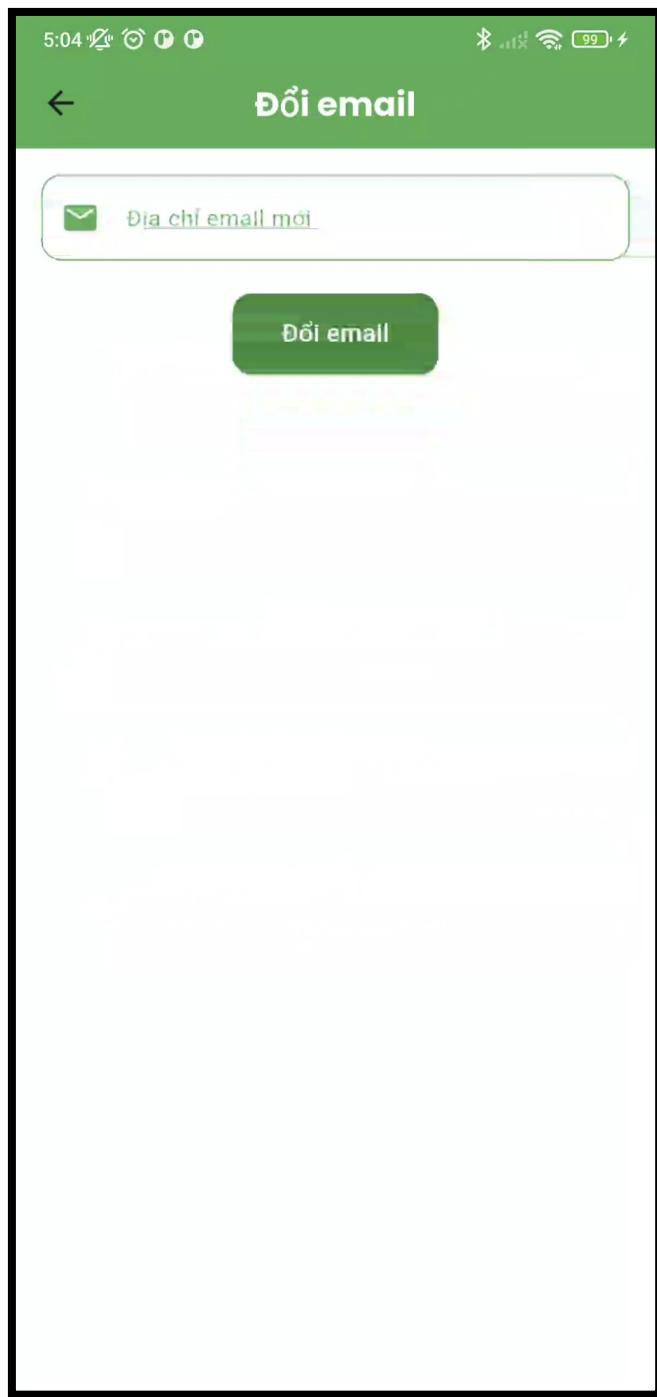
Hình 3.49: Giao diện cài đặt

\* Giao diện cài đặt tài khoản



Hình 3.50: Giao diện cài đặt tài khoản

\* Giao diện thay đổi email



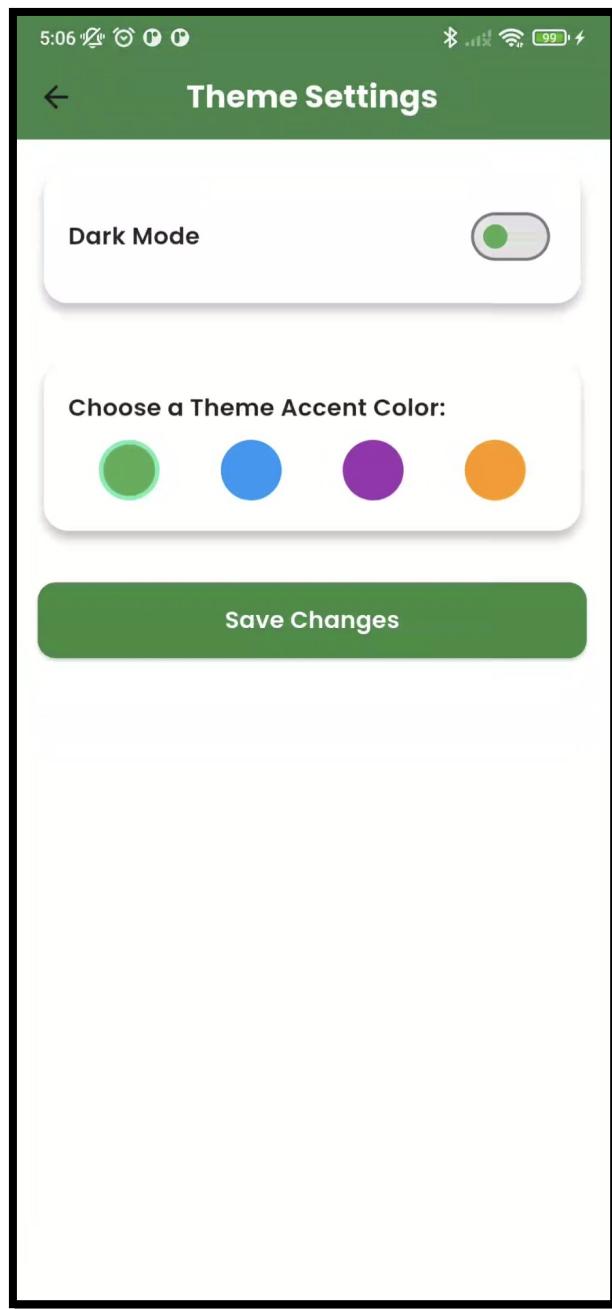
Hình 3.51: Giao diện đổi email

- \* Giao diện thay đổi mật khẩu



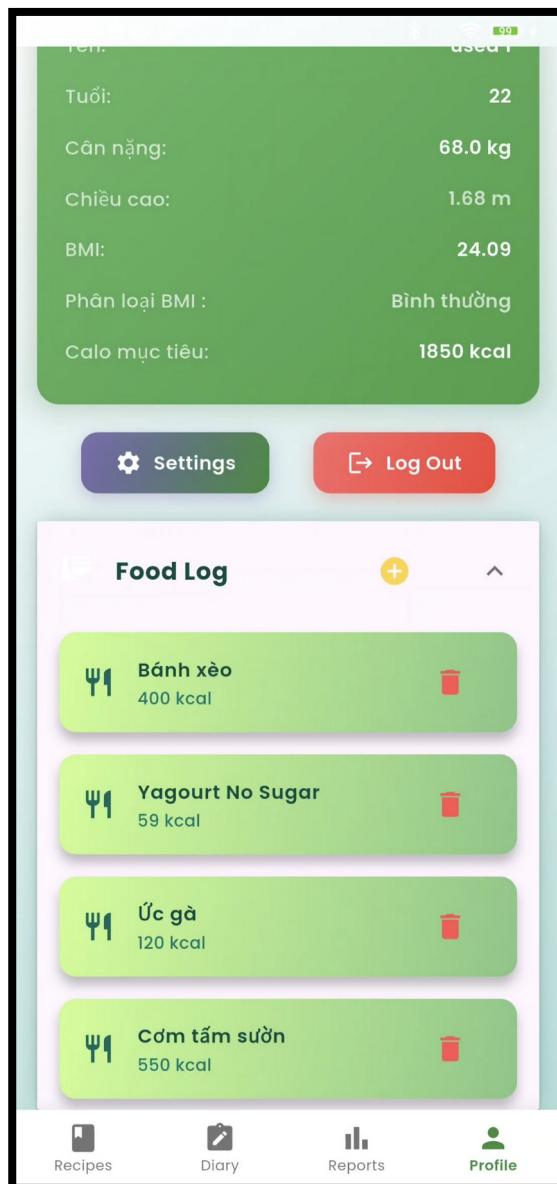
Hình 3.52: Giao diện đổi mật khẩu

- \* Giao diện thay đổi sáng tối



Hình 3.53: Giao diện thay đổi sáng tối

\* Giao diện thực đơn



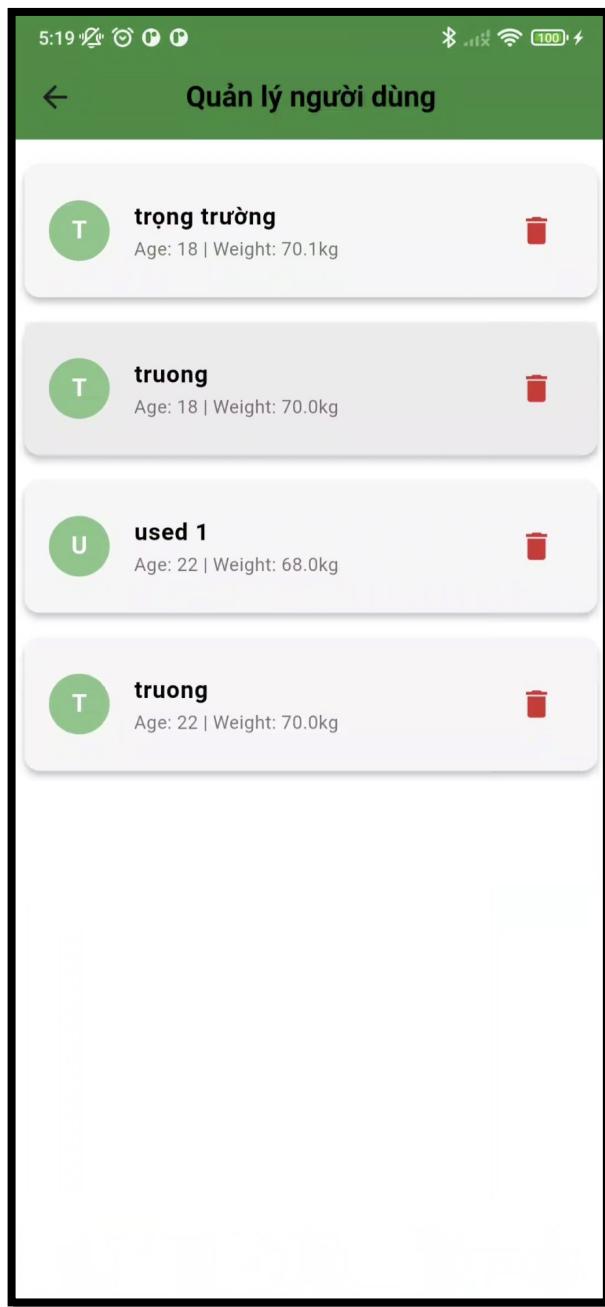
Hình 3.54: Giao diện thực đơn

\* Giao diện quản trị viên



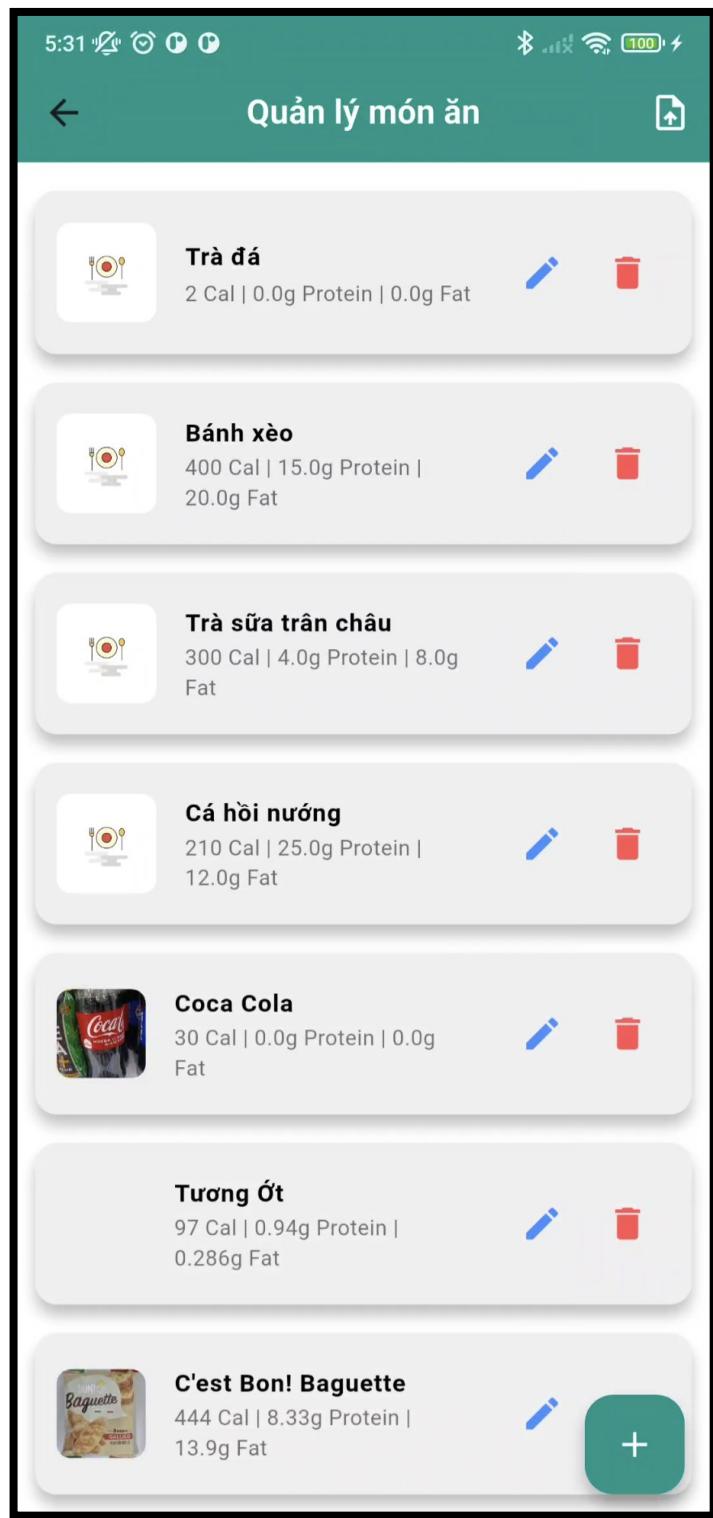
Hình 3.55: Giao diện quản trị viên

\* Giao diện quản lý người dùng



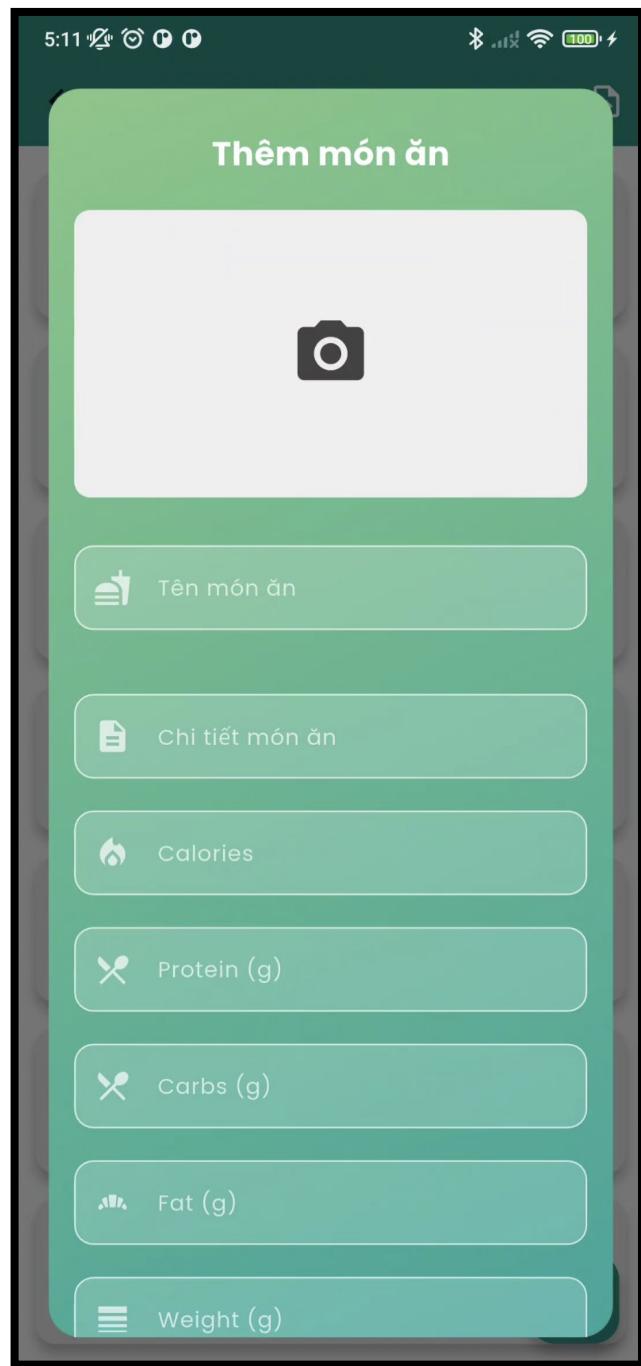
Hình 3.56: Giao diện quản lý người dùng

\* Giao diện quản lý món ăn



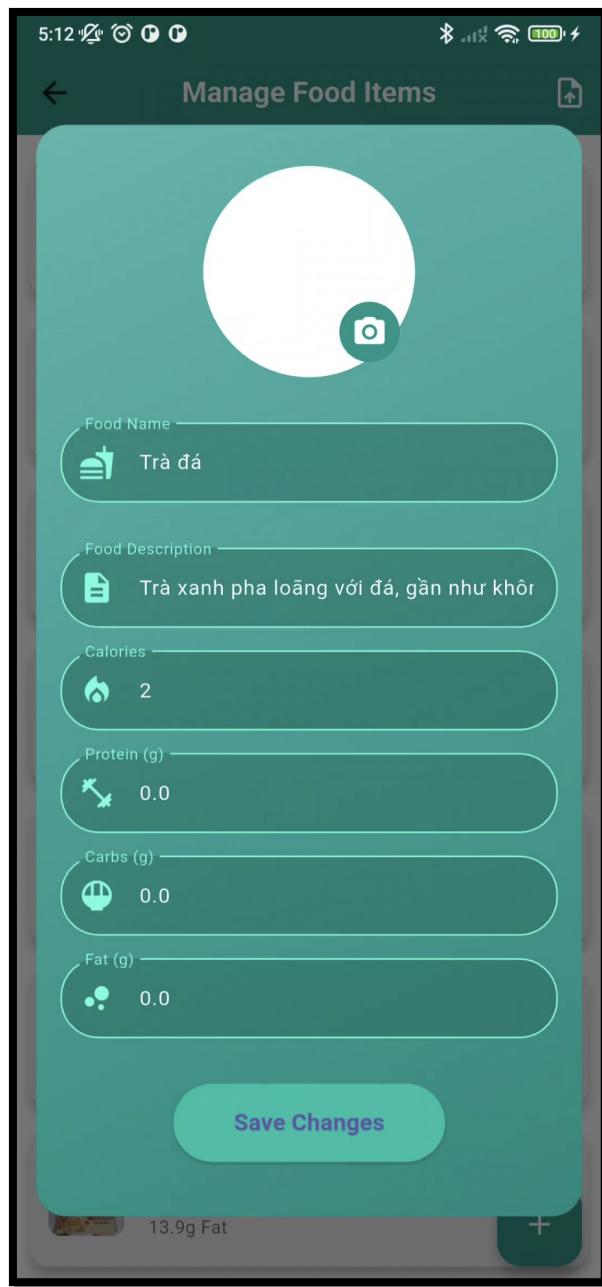
Hình 3.57: Giao diện quản lý món ăn

\* Giao diện thêm món ăn vào danh sách



Hình 3.58: Giao diện thêm món ăn vào danh sách

\* Giao diện chỉnh sửa món ăn



Hình 3.59: Giao diện chỉnh sửa món ăn

## 3.2 Kiểm thử chương trình

### 3.3.1 Khái niệm

Kiểm thử phần mềm (Software Testing) là một quá trình có hệ thống nhằm phát hiện lỗi, đảm bảo phần mềm hoạt động chính xác, đầy đủ và tuân thủ theo các yêu cầu đã được đặt ra trong tài liệu đặc tả kỹ thuật hoặc từ phía khách hàng. Đây không chỉ là công cụ tìm lỗi, mà còn là phương pháp đánh giá chất lượng phần mềm một cách khách quan, giúp giảm thiểu rủi ro khi triển khai hệ thống vào môi trường thực tế.

Trong quy trình kiểm thử, một giai đoạn đặc biệt quan trọng là kiểm thử hệ thống (System Testing). Đây là bước kiểm tra tổng thể toàn bộ hệ thống sau khi đã hoàn tất việc tích hợp các thành phần, nhằm đánh giá phần mềm có đáp ứng đúng các yêu cầu chức năng và phi chức năng đã đề ra hay không. Phương pháp chủ yếu được sử dụng trong giai đoạn này là kiểm thử hộp đen (Black-box Testing) – nghĩa là kiểm thử dựa trên đầu vào và đầu ra mà không cần quan tâm đến mã nguồn hay cấu trúc nội bộ của phần mềm.

### 3.3.2. Đặc điểm của kiểm thử hệ thống:

- Áp dụng kiểm thử hộp đen: Tập trung kiểm tra đầu vào – đầu ra mà không xét đến cấu trúc nội bộ.
- Thực hiện sau kiểm thử tích hợp: Là giai đoạn tiếp theo sau khi các module đã được tích hợp thành một hệ thống hoàn chỉnh.
- Đánh giá tổng thể: Kiểm tra đầy đủ chức năng chính, phụ và các yếu tố phi chức năng như hiệu suất, bảo mật, khả năng sử dụng,...
- Góc nhìn người dùng cuối: Mô phỏng trải nghiệm thực tế từ phía người sử dụng để đảm bảo tính thân thiện và phù hợp.
- Kiểm tra sự đầy đủ và đúng đắn: Đảm bảo hệ thống thực sự thỏa mãn các yêu cầu đã được đề xuất ban đầu.
- Môi trường tương đương thực tế: Thường thực hiện trong môi trường gần giống hoặc tương đồng với môi trường triển khai thật.

### 3.2.3. Các lĩnh vực kiểm thử hệ thống

Kiểm thử hệ thống không chỉ giới hạn ở chức năng phần mềm mà còn bao gồm nhiều khía cạnh khác, cụ thể:

- **Hiệu suất:** Đảm bảo hệ thống hoạt động ổn định, đúng yêu cầu và không phát sinh lỗi trong điều kiện sử dụng thực tế.
- **Bảo mật:** Ngăn chặn truy cập trái phép, bảo vệ dữ liệu người dùng và hệ thống khỏi các rủi ro bảo mật.
- **Giao diện:** Kiểm tra sự đầy đủ, chính xác và ổn định của giao diện người dùng theo yêu cầu thiết kế.
- **Khả năng cài đặt:** Đảm bảo phần mềm có thể được triển khai vào môi trường thật mà không phát sinh lỗi hoặc khó khăn.
- **Kiểm thử tải và chịu tải (Load/Stress Testing):** Đánh giá khả năng hệ thống xử lý các tình huống tải nặng, đảm bảo không bị quá tải hoặc sập hệ thống.

### 3.2.4. Hoàn thành quá trình kiểm thử

Để kiểm thử hệ thống đạt hiệu quả cao, quy trình cần được xây dựng và tuân thủ chặt chẽ. Một quy trình kiểm thử hệ thống hoàn chỉnh thường bao gồm các bước sau:

- **Lập kế hoạch kiểm thử (Test Plan):** Xác định phạm vi, mục tiêu, chiến lược kiểm thử, tiêu chí thành công/thất bại, lựa chọn công cụ và phân công nhân sự.
- **Thiết kế test case (Test Case):** Dựa trên tài liệu yêu cầu và các kịch bản sử dụng (use case), xây dựng các tình huống kiểm thử cụ thể bao gồm chức năng, giao diện và phi chức năng.
- **Chuẩn bị dữ liệu kiểm thử (Test Data):** Tạo hoặc lựa chọn dữ liệu đầu vào phù hợp cho các test case để đảm bảo việc kiểm thử phản ánh đúng các kịch bản sử dụng thực tế.
- **Thực thi kiểm thử (Test Execution):** Triển khai các test case trên hệ thống, ghi nhận kết quả và phát hiện lỗi (nếu có).

- **Báo cáo và xử lý lỗi (Defect Reporting & Fixing):** Ghi nhận chi tiết các lỗi phát hiện, chuyển đến nhóm phát triển để khắc phục và phối hợp xác minh sau sửa lỗi.
- **Kiểm thử lại (Retesting):** Sau khi lỗi được sửa, lặp lại các test case liên quan để đảm bảo hệ thống không còn tồn tại vấn đề.

### 3.2.5. Lý do thực hiện Kiểm thử hệ thống

Kiểm thử hệ thống đóng vai trò thiết yếu trong quá trình phát triển phần mềm, với các lợi ích nổi bật:

- **Nâng cao chất lượng sản phẩm:** Giúp phát hiện và loại bỏ lỗi, sự cố trước khi hệ thống được triển khai chính thức.
- **Đảm bảo phần mềm đúng yêu cầu:** Xác minh phần mềm thực hiện đúng các chức năng đã cam kết với người dùng.
- **Phát hiện lỗi sớm:** Giảm chi phí sửa lỗi bằng cách phát hiện vấn đề ở giai đoạn sớm.
- **Tăng khả năng tích hợp:** Kiểm tra các thành phần trong hệ thống có thể phối hợp và hoạt động trơn tru với nhau.
- **Tăng cường bảo mật:** Giúp phát hiện lỗ hổng, từ đó giảm nguy cơ bị khai thác.
- **Đảm bảo khả năng triển khai:** Xác minh rằng hệ thống có thể được vận hành ổn định trong môi trường thực tế.

### 3.3. Thiết kế các test case cho hệ thống

#### 1. Lịch trình công việc

Bảng 3.7: Bảng lập kế hoạch kiểm thử

Mốc công việc	Sản phẩm	Thời gian	Bắt đầu	Kết thúc
Lập kế hoạch kiểm thử	Test plan	2 ngày	01/09/2025	03/09/2025
Xem lại các tài liệu	Test plan	2 ngày	04/09/2025	06/09/2025
Thiết kế các testcase	Test case	1 ngày	07/09/2025	07/09/2025

Viết các testcase	Test case	2 ngày	08/09/2025	10/09/2025
Xem lại các testcase	Test case	1 ngày	11/09/2025	11/09/2025
Thực thi các testcase	Test case	1 ngày	12/09/2025	12/09/2025
Ghi nhận và đánh giá kết quả kiểm thử	Test report	2 ngày	13/09/2025	15/09/2025

## 2. Kết quả kiểm thử

### ***Use case: Đăng nhập***

Bảng 3.8: Test case use case đăng nhập

STT	Mô tả	Dữ liệu đầu vào	Bước thực hiện	Kết quả mong đợi	Trạng thái
TC01	Đăng nhập thành công	Email hợp lệ, mật khẩu đúng	Nhập email và mật khẩu > Submit	Chuyển đến trang chủ người dùng	Đạt
TC02	Sai mật khẩu	Email hợp lệ, mật khẩu sai	Nhập email và mật khẩu > Submit	Hiển thị thông báo “Mật khẩu không đúng”	Đạt
TC03	Email không tồn tại	Email không tồn tại	Nhập email và mật khẩu > Submit	Hiển thị “Tài khoản không tồn tại”	Đạt
TC04	Để trống trường dữ liệu	Trống email hoặc mật khẩu	Submit không điền dữ liệu	Hiển thị cảnh báo “Vui lòng điền đầy đủ thông tin”	Đạt

### ***Use case: Đăng ký***

Bảng 3.9: Test case use case đăng ký

STT	Mô tả	Dữ liệu đầu vào	Bước thực hiện	Kết quả mong đợi	Trạng thái
TC01	Đăng ký thành công	Tên, email, mật khẩu hợp lệ	Điền form đăng ký > Submit	Chuyển đến trang đăng nhập và hiển thị thông báo đăng ký thành công	Đạt
TC02	Email đã tồn tại	Email đã có trong hệ	Điền form > Submit	Hiển thị thông báo “Email đã được	Đạt

		thông		đăng ký”	
TC03	Mật khẩu không khớp xác nhận	Mật khẩu và xác nhận không trùng	Điền form > Submit	Hiển thị lỗi “Mật khẩu không khớp”	Đạt
TC04	Trường thông tin thiếu	Bỏ trống một hoặc nhiều trường	Submit	Hiển thị lỗi bắt buộc điền	Đạt

### Use case: Quản lý hồ sơ cá nhân

Bảng 3.10: Test case use case quản lý hồ sơ cá nhân

STT	Mô tả	Dữ liệu đầu vào	Bước thực hiện	Kết quả mong đợi	Trạng thái
TC01	Cập nhật hồ sơ thành công	Thông tin người dùng đúng	Mở hồ sơ > Sửa dữ liệu > Lưu	Hiển thông tin người dùng mới	Đạt
TC02	Thiếu dữ liệu bắt buộc	Để trống tên người dùng	Mở hồ sơ > Sửa dữ liệu > Lưu	Thông báo lỗi	Đạt
TC03	Cập nhật thất bại do mất mạng	Thông tin người dùng đúng	Mở hồ sơ > Sửa dữ liệu > Lưu	Thông báo cập nhật thất bại	Đạt

### Use case: Thêm món ăn

Bảng 3.11: Test case use case thêm món ăn

STT	Mô tả	Dữ liệu đầu vào	Bước thực hiện	Kết quả mong đợi	Trạng thái
TC01	Thêm món ăn thành công	Thông tin món ăn đúng	Mở”Thêm món ăn” > nhập dữ liệu > Lưu	Thêm món ăn thành công	Đạt
TC02	Thiếu dữ liệu bắt buộc	Để trống tên món ăn	Mở hồ sơ > Sửa dữ liệu > Lưu	Thông báo lỗi	Đạt
TC03	Dữ liệu sai định dạng	Thông tin calo là chữ cái	Mở hồ sơ > Sửa dữ liệu > Lưu	Thông báo lỗi	Đạt

### Use case: Quản lý thực phẩm

Bảng 3.12: Test case use case quản lý thực phẩm

STT	Mô tả	Dữ liệu đầu vào	Bước thực hiện	Kết quả mong đợi	Trạng thái
TC01	Tìm món ăn	Từ khoá “Cơm”	Mở”Tìm kiếm món ăn” > nhập từ khoá” Cơm” > nhấn tìm kiếm	Hiển thị danh sách món ăn có từ “Cơm”	Đạt
TC02	Xem chi tiết món ăn	foodName:Phở bò	Mở danh sách món ăn > Chọn món Phở bò	Thông báo lỗi	Đạt

**Use case: Theo dõi nước uống**

Bảng 3.13: Test case use case theo dõi nước uống

STT	Mô tả	Dữ liệu đầu vào	Bước thực hiện	Kết quả mong đợi	Trạng thái
TC01	Thêm nước thành công	500	Nhập 500> Lưu	Thanh nước tăng 500	Đạt
TC02	Đạt mục tiêu	2000	Nhập 2000> Lưu	Thông báo đạt mục tiêu	Đạt
TC03	Dữ liệu sai định dạng	abc	Nhập abc> Lưu	Thông báo lỗi	Đạt

### **Use case: Đặt mục tiêu sức khoẻ**

Bảng 3.14: Test case use case đặt mục tiêu sức khoẻ

STT	Mô tả	Dữ liệu đầu vào	Bước thực hiện	Kết quả mong đợi	Trạng thái
TC01	Đặt mục tiêu thành công	2000	Nhập 2000> Lưu	Mục tiêu hiện 2000 calories	Đạt
TC02	Nhập sai dữ liệu	abc	Nhập abc> Lưu	Thông báo lỗi	Đạt

### **Use case: Xem báo cáo sức khoẻ**

Bảng 3.15: Test case use case xem báo cáo sức khoẻ

STT	Mô tả	Dữ liệu đầu vào	Bước thực hiện	Kết quả mong đợi	Trạng thái
TC01	Xem báo cáo có dữ liệu	Có dữ liệu	Người dùng thêm món ăn	Hiển thị biểu đồ	Đạt
TC02	Không có dữ liệu	None	Người dùng chưa thêm món ăn	Hiển thị chưa có dữ liệu	Đạt
TC03	Lỗi kết nối	Có dữ liệu	Người dùng tắt mạng, wifi	Hiển thị "Không thể tải báo cáo"	Đạt

### **Use case: Xem nhật ký**

Bảng 3.16: Test case use case xem nhật ký

STT	Mô tả	Dữ liệu đầu vào	Bước thực hiện	Kết quả mong đợi	Trạng thái
TC01	Xem nhật ký món ăn	Có dữ liệu	Người dùng thêm món ăn	Danh sách hiển thị đúng	Đạt
TC02	Xem nhật ký theo ngày	Có dữ liệu	Bấm vào ngày muốn xem	Danh sách hiển thị đúng	Đạt

## KẾT LUẬN

Qua quá trình nghiên cứu, thiết kế và triển khai, đề tài "Xây dựng ứng dụng di động chăm sóc sức khoẻ sử dụng Firebase và Flutter" đã đạt được những kết quả tích cực và hoàn thành các mục tiêu đề ra ban đầu. Ứng dụng đã được phát triển với đầy đủ các chức năng cốt lõi của hệ thống . Người dùng có thể đăng ký, đăng nhập, thêm món ăn, xem báo cáo, xem nhật ký, chỉnh sửa thông tin. Ứng dụng cũng cung cấp giao diện quản trị cho phép quản lý món ăn, người dùng. Việc sử dụng Dart đã tạo nên ứng dụng có hiệu suất cao. Firebase mang lại sự linh hoạt trong lưu trữ dữ liệu, cung cấp cơ chế xác thực an toàn, trong khi Flutter tạo giao diện người dùng hiện đại và thân thiện.

Hệ thống vẫn còn một số điểm cần cải thiện như tối ưu giao diện người dùng, bổ sung các biện pháp mã hóa dữ liệu và xác thực hai yếu tố (2FA). Hiệu suất cần được nâng cao thông qua tối ưu hóa truy vấn cơ sở dữ liệu và implement caching. Ngoài ra, cần tích hợp thêm các tính năng dinh dưỡng. Đề tài không chỉ mang ý nghĩa học thuật mà còn có giá trị thực tiễn cao. Việc ứng dụng thành công các công nghệ hiện đại như Dart, Flutter, Firebase vào xây dựng hệ thống thương mại điện tử đã góp phần nâng cao hiểu biết và kỹ năng phát triển phần mềm. Đồng thời, sản phẩm tạo ra có thể được ứng dụng trong thực tế, đáp ứng nhu cầu chăm sóc dinh dưỡng của người dùng.

Ứng dụng đã xây dựng được nền tảng vững chắc, có khả năng mở rộng và phát triển thành một hệ thống quản lý sức khoẻ chuyên nghiệp trong tương lai. Điều này không chỉ góp phần vào việc số hóa hoạt động chăm sóc sức khoẻ mà còn mở ra cơ hội ứng dụng công nghệ thông tin vào giải quyết các bài toán thực tế trong kỷ nguyên chuyển đổi số.

## TÀI LIỆU THAM KHẢO

### **Tiếng Việt**

- [1] L. N. Tiến, *Firebase và ứng dụng trong phát triển di động*, ĐH CNTT TP.HCM, 2021.
- [2] N. H. Linh, “Phân tích hệ thống quản lý chi tiêu cá nhân,” Báo cáo môn học, ĐH Bách Khoa Hà Nội, 2022.
- [3] N. V. Hiếu, *Lập trình Android cơ bản đến nâng cao*, NXB Hồng Đức, 2020
- [4] T. C. Thành, *Kỹ thuật phần mềm và kiểm thử*, NXB Đại học Quốc gia Hà Nội, 2018.

### **Tiếng Anh**

- [5] M. Fowler, *Patterns of Enterprise Application Architecture*, Addison-Wesley, 2002.
- [6] J. Smith, *Mastering Kotlin for Android Development*, Packt Publishing, 2019.
- [7] K. Sriraman, *MVVM in Android Development*, Apress, 2021.

### **Website**

- [8] <https://developer.android.com/>( Truy cập lần cuối 22/09/2025)
- [9] <https://flutterflow.io/>( Truy cập lần cuối 22/09/2025)
- [10] <https://www.figma.com/community>(Truy cập lần cuối 22/09/2025)
- [11] <https://firebase.google.com/>( Truy cập lần cuối 22/09/2025)
- [12] <https://www.mongodb.com/developer/products/mongodb/schema-design-best-practices>( Truy cập lần cuối 22/09/2025)

