**BỘ KHOA HỌC VÀ CÔNG NGHỆ**

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

-----------------------------



**BÁO CÁO   
ĐỒ ÁN MÔN HỌC**

**ĐỀ TÀI: XÂY DỰNG CALORIE TRACKING APP**

**Môn học:** Phát triển ứng dụng cho các thiết bị di động

**Giảng viên hướng dẫn:** Nguyễn Trung Hiếu

**Thực hiện bởi nhóm sinh viên, bao gồm:**

1. Trần Quang Hùng N22DCPT035 <Trưởng nhóm>
2. Vũ Quang Long N22DCPT052 <Thành viên>

**TP.HCM, tháng** **12 / 2025**

# MỤC LỤC

[MỤC LỤC 2](#_Toc217913531)

[DANH SÁCH HÌNH, BẢNG 4](#_Toc217913532)

[TÓM TẮT 5](#_Toc217913533)

[CHƯƠNG I. TỔNG QUAN ĐỒ ÁN 6](#_Toc217913534)

[1.1. Lý do chọn đề tài 6](#_Toc217913535)

[1.2. Mục tiêu của đề tài 7](#_Toc217913536)

[1.3. Phạm vi nghiên cứu và triển khai 7](#_Toc217913537)

[1.4. Phương pháp nghiên cứu 8](#_Toc217913538)

[CHƯƠNG II. CƠ SỞ LÝ THUYẾT VÀ CÔNG NGHỆ SỬ DỤNG 10](#_Toc217913539)

[2.1. Tổng quan về kiến trúc MVVM 10](#_Toc217913540)

[2.2. Nền tảng Android và ngôn ngữ Java 11](#_Toc217913541)

[2.3. Android Jetpack Components 13](#_Toc217913542)

[2.4. Room Database và SQLite 15](#_Toc217913543)

[2.5. Material Design 3 18](#_Toc217913544)

[2.6. Các thư viện bên thứ ba 20](#_Toc217913545)

[2.7. Công cụ phát triển 21](#_Toc217913546)

[2.8. Bảng tổng hợp công nghệ 23](#_Toc217913547)

[CHƯƠNG III. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG 24](#_Toc217913548)

[3.1. Phân tích yêu cầu 24](#_Toc217913549)

[3.2. Biểu đồ Use Case 26](#_Toc217913550)

[3.3. Thiết kế kiến trúc hệ thống 31](#_Toc217913551)

[3.4. Thiết kế cơ sở dữ liệu 34](#_Toc217913552)

[3.5. Thiết kế các thuật toán tính toán 38](#_Toc217913553)

[3.6. Sequence Diagrams 44](#_Toc217913554)

[3.7. Tổng kết chương 48](#_Toc217913555)

[CHƯƠNG IV. THIẾT KẾ GIAO DIỆN (UI/UX) 49](#_Toc217913556)

[4.1. Nguyên tắc thiết kế UI/UX 49](#_Toc217913557)

[4.2. Hệ thống màu sắc (Color System) 50](#_Toc217913558)

[4.3. Typography và Spacing 54](#_Toc217913559)

[4.4. Thiết kế các màn hình chính 58](#_Toc217913560)

[4.5. Thiết kế Components 64](#_Toc217913561)

[4.6. Navigation Flow 70](#_Toc217913562)

[4.7. Responsive Design 73](#_Toc217913563)

[4.8. Dark Mode Support 73](#_Toc217913564)

[4.9. Screenshots và Wireframes 75](#_Toc217913565)

[4.10. Tổng kết thiết kế UI/UX 77](#_Toc217913566)

[CHƯƠNG V: XÂY DỰNG VÀ CÀI ĐẶT CHƯƠNG TRÌNH 78](#_Toc217913567)

[5.1. Môi trường phát triển 78](#_Toc217913568)

[5.2. Luồng hoạt động chính 79](#_Toc217913569)

[5.3. Cài đặt các thành phần chính 81](#_Toc217913570)

[5.4. Tích hợp API và Caching 86](#_Toc217913571)

[5.5. Hướng dẫn cài đặt 87](#_Toc217913572)

[CHƯƠNG VI. TỔNG KẾT VÀ HƯỚNG PHÁT TRIỂN 88](#_Toc217913573)

[6.1. Kết quả đạt được 88](#_Toc217913574)

[6.2. Ưu điểm và hạn chế 88](#_Toc217913575)

[6.3. Hướng phát triển 89](#_Toc217913576)

[6.4. Kết luận 90](#_Toc217913577)

[TÀI LIỆU THAM KHẢO 91](#_Toc217913578)

# DANH SÁCH HÌNH, BẢNG

# BẢNG PHÂN CÔNG NHIỆM VỤ

|  |  |  |  |
| --- | --- | --- | --- |
| **Tên sinh viên** | **MSSV** | **Nhiệm vụ** | **Mức độ hoàn thành** |
| Trần Quang Hùng | N22DCPT035 | - Phân tích yêu cầu và thiết kế hệ thống  - Xây dựng Data Layer: Room Database, Entities, DAOs, Repositories  - Tích hợp USDA API và caching strategy  - Viết báo cáo Chương 1, 3, 5  - Kiểm thử và sửa lỗi | 100% |
| Vũ Quang Long | N22DCPT052 | -Nghiên cứu cơ sở lý thuyết dinh dưỡng  - Xây dựng UI Layer: Activities, Fragments, Adapters  - Thiết kế giao diện và tích hợp biểu đồ MPAndroidChart  - Viết báo cáo Chương 2, 4, 6  - Tổng hợp tài liệu tham khảo | 100% |

**THUẬN LỢI**

- Tài liệu Android Developers đầy đủ, cập nhật thường xuyên

- Thư viện mã nguồn mở (MPAndroidChart, Room, Volley) có cộng đồng hỗ trợ lớn

- USDA API miễn phí với cơ sở dữ liệu thực phẩm phong phú

- Android Studio hỗ trợ tốt việc phát triển và debug

**KHÓ KHĂN**

- Thiết kế database phức tạp, phải thay đổi schema nhiều lần

- Xử lý bất đồng bộ (Room, API) gây nhiều lỗi ban đầu

- Thiếu cơ sở dữ liệu thực phẩm Việt Nam chuẩn, phải nhập liệu thủ công

- Customize biểu đồ MPAndroidChart khó do documentation không đầy đủ

- Thời gian hạn chế, phải cân bằng với việc thực hiện đồ án môn khác

# CHƯƠNG I. TỔNG QUAN ĐỒ ÁN

## 1.1. Lý do chọn đề tài

Trong bối cảnh xã hội hiện đại, vấn đề sức khỏe và dinh dưỡng ngày càng được quan tâm. Theo thống kê của Tổ chức Y tế Thế giới (WHO), tỷ lệ béo phì và các bệnh liên quan đến chế độ ăn uống không lành mạnh đang gia tăng đáng kể, đặc biệt tại các quốc gia đang phát triển như Việt Nam. Việc kiểm soát lượng calo tiêu thụ hàng ngày được xem là một trong những phương pháp hiệu quả nhất để duy trì cân nặng hợp lý và phòng ngừa các bệnh mãn tính.

Tuy nhiên, các ứng dụng theo dõi calo hiện có trên thị trường chủ yếu được phát triển cho người dùng quốc tế, với cơ sở dữ liệu thực phẩm phương Tây. Điều này gây khó khăn cho người dùng Việt Nam khi muốn tra cứu thông tin dinh dưỡng của các món ăn truyền thống như phở, bún chả, cơm tấm hay bánh mì. Ngoài ra, giao diện tiếng Anh cũng là rào cản đối với nhiều người dùng.

Xuất phát từ thực tế trên, đề tài "Xây dựng ứng dụng Android theo dõi calo - TrackingCaloApp" được thực hiện nhằm cung cấp một giải pháp phù hợp với người dùng Việt Nam, với các đặc điểm:

• Cơ sở dữ liệu hơn 50 món ăn Việt Nam phổ biến với thông tin dinh dưỡng chính xác

• Giao diện hoàn toàn bằng tiếng Việt, thân thiện và dễ sử dụng

• Tích hợp các công thức tính toán khoa học (BMR, TDEE, BMI) theo tiêu chuẩn châu Á

• Hỗ trợ theo dõi cả thực phẩm tiêu thụ và bài tập thể dục

## 1.2. Mục tiêu của đề tài

**1.2.1. Mục tiêu tổng quát**

Xây dựng ứng dụng di động trên nền tảng Android giúp người dùng Việt Nam theo dõi lượng calo tiêu thụ và đốt cháy hàng ngày, từ đó hỗ trợ việc quản lý cân nặng và duy trì lối sống lành mạnh.

**1.2.2. Mục tiêu cụ thể**

**Về mặt chức năng:**

• Cho phép người dùng ghi nhận thực phẩm tiêu thụ theo từng bữa ăn (sáng, trưa, tối, bữa phụ)

• Cho phép người dùng ghi nhận các hoạt động thể dục với nhiều loại bài tập khác nhau

• Tự động tính toán lượng calo, protein, carbohydrate và chất béo dựa trên khối lượng thực phẩm

• Tính toán mục tiêu calo cá nhân hóa dựa trên thông tin người dùng (tuổi, giới tính, chiều cao, cân nặng, mức độ hoạt động)

• Hiển thị báo cáo và biểu đồ thống kê trực quan

• Theo dõi tiến độ cân nặng theo thời gian

**Về mặt kỹ thuật:**

• Áp dụng kiến trúc MVVM (Model-View-ViewModel) kết hợp Repository Pattern

• Sử dụng Room Database để lưu trữ dữ liệu local hiệu quả

• Tích hợp API bên ngoài (USDA FoodData Central) để mở rộng cơ sở dữ liệu thực phẩm

• Thiết kế giao diện theo chuẩn Material Design 3

• Đảm bảo hiệu năng và trải nghiệm người dùng mượt mà

## 1.3. Phạm vi nghiên cứu và triển khai

**1.3.1. Phạm vi nghiên cứu**

* Nghiên cứu các công thức tính toán dinh dưỡng: BMR (Basal Metabolic Rate) theo công thức Mifflin-St Jeor, TDEE (Total Daily Energy Expenditure), BMI (Body Mass Index) theo tiêu chuẩn châu Á
* Nghiên cứu kiến trúc ứng dụng Android hiện đại với các thành phần Jetpack
* Nghiên cứu thiết kế cơ sở dữ liệu quan hệ cho ứng dụng theo dõi dinh dưỡng
* Nghiên cứu tích hợp API RESTful trong ứng dụng Android

**1.3.2. Phạm vi triển khai**

**Đối tượng người dùng:**

• Người dùng Việt Nam có nhu cầu theo dõi chế độ ăn uống

• Người muốn giảm cân, tăng cân hoặc duy trì cân nặng

• Người tập thể dục, thể thao muốn kiểm soát dinh dưỡng

**Nền tảng:**

• Hệ điều hành: Android 8.0 (API level 27) trở lên

• Ngôn ngữ lập trình: Java 11

• IDE: Android Studio

**Giới hạn:**

• Ứng dụng hoạt động offline với dữ liệu local, tích hợp API khi có kết nối mạng

• Chưa hỗ trợ đồng bộ dữ liệu đa thiết bị

• Chưa tích hợp với các thiết bị đeo thông minh (smartwatch, fitness tracker)

## 1.4. Phương pháp nghiên cứu

**1.4.1. Phương pháp nghiên cứu lý thuyết**

• Thu thập và phân tích tài liệu về dinh dưỡng học, các công thức tính toán calo được công nhận quốc tế

• Nghiên cứu tài liệu chính thức của Google về phát triển ứng dụng Android (Android Developers Documentation)

• Tham khảo các best practices về kiến trúc phần mềm và design patterns

**1.4.2. Phương pháp phân tích và thiết kế**

• Phân tích yêu cầu chức năng và phi chức năng của hệ thống

• Thiết kế kiến trúc theo mô hình MVVM với các layer rõ ràng

• Thiết kế cơ sở dữ liệu với Entity Relationship Diagram (ERD)

• Thiết kế giao diện người dùng với wireframes và mockups

**1.4.3. Phương pháp thực nghiệm**

• Xây dựng ứng dụng theo phương pháp phát triển lặp (Iterative Development)

• Kiểm thử chức năng trên nhiều thiết bị và phiên bản Android khác nhau

• Thu thập phản hồi và cải tiến liên tục

**1.4.4. Công cụ và môi trường phát triển**

|  |  |  |
| --- | --- | --- |
| Công cụ | Phiên bản | Mục đích |
| Android Studio | Hedgehog 2023.1.1+ | IDE phát triển |
| Java | 11 | Ngôn ngữ lập trình |
| Gradle | 8.0+ | Build system |
| Room Database | 2.6.1 | Local database |
| Material Design | 3 (1.12.0) | UI Components |
| MPAndroidChart | 3.1.0 | Biểu đồ thống kê |
| Volley | 1.2.1 | Network requests |
| Git | - | Version control |

# CHƯƠNG II. CƠ SỞ LÝ THUYẾT VÀ CÔNG NGHỆ SỬ DỤNG

## 2.1. Tổng quan về kiến trúc MVVM

**2.1.1. Giới thiệu mô hình MVVM**

MVVM (Model-View-ViewModel) là một mô hình kiến trúc phần mềm được Google khuyến nghị sử dụng cho phát triển ứng dụng Android. Mô hình này giúp tách biệt rõ ràng giữa giao diện người dùng (UI) và logic xử lý dữ liệu, từ đó tăng khả năng bảo trì, kiểm thử và mở rộng ứng dụng.

**Các thành phần chính của MVVM bao gồm:**

• Model: Đại diện cho dữ liệu và business logic của ứng dụng. Trong TrackingCaloApp, Model bao gồm các Entity classes (Food, FoodEntry, Workout, WorkoutEntry, User, WeightLog) và các Repository classes.

• View: Chịu trách nhiệm hiển thị dữ liệu và nhận tương tác từ người dùng. Trong Android, View được triển khai thông qua Activity và Fragment.

• ViewModel: Đóng vai trò trung gian giữa View và Model. ViewModel chứa logic xử lý UI, quản lý trạng thái và cung cấp dữ liệu cho View thông qua LiveData.

**2.1.2. Repository Pattern**

Repository Pattern được sử dụng kết hợp với MVVM để tạo một lớp trừu tượng giữa ViewModel và các nguồn dữ liệu (local database, remote API). Điều này giúp:

• Tách biệt logic truy xuất dữ liệu khỏi ViewModel

• Dễ dàng thay đổi nguồn dữ liệu mà không ảnh hưởng đến các layer khác

• Hỗ trợ caching và đồng bộ dữ liệu giữa local và remote

**2.1.3. Luồng dữ liệu trong ứng dụng**

Luồng dữ liệu trong TrackingCaloApp tuân theo nguyên tắc một chiều (Unidirectional Data Flow):

1. User Action: Người dùng tương tác với UI (click button, nhập text)

2. UI Layer: Activity/Fragment xử lý event và gọi Repository

3. Repository: Thực hiện business logic, gọi DAO

4. Database: Room thực hiện query, trả về LiveData

5. LiveData: Tự động notify UI khi data thay đổi

6. UI Update: UI tự động cập nhật với data mới

## 2.2. Nền tảng Android và ngôn ngữ Java

**2.2.1. Nền tảng Android**

Android là hệ điều hành di động phổ biến nhất thế giới, được phát triển bởi Google. TrackingCaloApp được xây dựng trên nền tảng Android với các thông số kỹ thuật:

• Compile SDK: 36 (Android 14)

• Minimum SDK: 27 (Android 8.1 Oreo)

• Target SDK: 36 (Android 14)

Việc chọn minSdk 27 đảm bảo ứng dụng có thể chạy trên khoảng 95% thiết bị Android hiện tại, đồng thời tận dụng được các API mới nhất.

**2.2.2. Ngôn ngữ Java 11**

Java là ngôn ngữ lập trình chính thức được hỗ trợ cho phát triển Android. TrackingCaloApp sử dụng Java 11 với các tính năng:

• Type Inference với var keyword

• String methods mới (isBlank, strip, repeat)

• Collection factory methods

• Optional improvements

Cấu hình Java trong build.gradle.kts:

|  |
| --- |
| compileOptions {      sourceCompatibility = JavaVersion.VERSION\_11      targetCompatibility = JavaVersion.VERSION\_11  } |

**2.2.3. Kiến trúc Single Activity**

TrackingCaloApp áp dụng kiến trúc Single Activity với Multiple Fragments. MainActivity đóng vai trò container chính, các màn hình được triển khai dưới dạng Fragment và chuyển đổi thông qua Bottom Navigation.

Ưu điểm của kiến trúc này:

• Quản lý navigation tập trung

• Chia sẻ dữ liệu giữa các màn hình dễ dàng

• Giảm overhead khi chuyển màn hình

• Hỗ trợ animation mượt mà hơn

## 2.3. Android Jetpack Components

**2.3.1. Room Database**

Room là thư viện persistence của Android Jetpack, cung cấp abstraction layer trên SQLite. Room giúp đơn giản hóa việc làm việc với database thông qua:

• Entity: Đại diện cho bảng trong database

• DAO (Data Access Object): Định nghĩa các phương thức truy vấn

• Database: Singleton class quản lý kết nối database

Trong TrackingCaloApp, Room Database được cấu hình với 6 entities:

|  |
| --- |
| @Database(      entities = {Food.class, FoodEntry.class, Workout.class,                  WorkoutEntry.class, WeightLog.class, User.class},      version = 4,      exportSchema = false  )  public abstract class AppDatabase extends RoomDatabase {      public abstract FoodDao foodDao();      public abstract FoodEntryDao foodEntryDao();      public abstract WorkoutDao workoutDao();      public abstract WorkoutEntryDao workoutEntryDao();      public abstract WeightLogDao weightLogDao();      public abstract UserDao userDao();  } |

Room hỗ trợ Migration để cập nhật schema database mà không mất dữ liệu người dùng. TrackingCaloApp đã triển khai 3 migrations (1→2, 2→3, 3→4) để thêm các tính năng mới.

**2.3.2. LiveData**

LiveData là observable data holder class, lifecycle-aware. LiveData tự động quản lý subscription dựa trên lifecycle của Activity/Fragment, giúp:

• Tránh memory leaks

• Không crash khi Activity bị destroy

• Tự động cập nhật UI khi data thay đổi

• Chia sẻ resources giữa các observers

Ví dụ sử dụng LiveData trong TrackingCaloApp:

|  |
| --- |
| LiveData<Float> consumedLiveData = foodEntryRepository.getTotalCaloriesByDate(startOfDay, endOfDay);  consumedLiveData.observe(getViewLifecycleOwner(), consumed -> {      tvCaloriesConsumed.setText(String.valueOf((int) consumed));  }); |

**2.3.3. ViewModel**

ViewModel là class được thiết kế để lưu trữ và quản lý UI-related data theo lifecycle-aware manner. ViewModel tồn tại qua configuration changes (như xoay màn hình), giúp:

• Tách biệt UI logic khỏi Activity/Fragment

• Dữ liệu không bị mất khi configuration change

• Dễ dàng unit test

Ví dụ ProfileViewModel trong TrackingCaloApp:

|  |
| --- |
| public class ProfileViewModel extends AndroidViewModel {  private final MutableLiveData<UserInfo> userInfo = new MutableLiveData<>();  private final LiveData<List<WeightLog>> weightLogs;    public ProfileViewModel(@NonNull Application application) {  super(application);  userPreferences = new UserPreferences(application);  weightLogRepository = new WeightLogRepository(application);  weightLogs = weightLogRepository.getLast30DaysLogs();  loadUserInfo();  }  } |

**2.3.4. ViewPager2**

ViewPager2 là phiên bản cải tiến của ViewPager, được xây dựng trên RecyclerView. TrackingCaloApp sử dụng ViewPager2 cho:

• DiaryFragment: Chuyển đổi giữa tab Food Entries và Workout Entries

• AddFragment: Chuyển đổi giữa tab Thêm Food và Thêm Workout

Ưu điểm của ViewPager2:

• Hỗ trợ vertical orientation

• RTL (Right-to-Left) support

• Modifiable Fragment collections

• DiffUtil support

## 2.4. Room Database và SQLite

**2.4.1. Tổng quan về SQLite**

SQLite là hệ quản trị cơ sở dữ liệu quan hệ nhúng (embedded), được tích hợp sẵn trong Android. SQLite có các đặc điểm:

• Serverless: Không cần cài đặt server riêng

• Zero-configuration: Không cần cấu hình phức tạp

• Self-contained: Toàn bộ database nằm trong một file

• Transactional: Hỗ trợ ACID transactions

**2.4.2. Cấu trúc Database trong TrackingCaloApp**

Database của ứng dụng bao gồm 6 bảng chính:

**Bảng foods:**

• id (INTEGER, PRIMARY KEY)

• name (TEXT) - Tên thực phẩm

• calories (REAL) - Calo/100g

• protein (REAL) - Protein/100g

• carbs (REAL) - Carbohydrate/100g

• fat (REAL) - Chất béo/100g

• category (TEXT) - Danh mục

• isCustom (INTEGER) - User tự tạo?

• apiId (INTEGER) - ID từ API

• apiSource (TEXT) - Nguồn API

• cachedAt (INTEGER) - Thời điểm cache

**Bảng food\_entries:**

• id (INTEGER, PRIMARY KEY)

• foodId (INTEGER, FOREIGN KEY) - Liên kết với foods

• quantity (REAL) - Khối lượng (gram)

• mealType (INTEGER) - Loại bữa ăn (0-3)

• date (INTEGER) - Timestamp

• totalCalories, totalProtein, totalCarbs, totalFat (REAL)

**Bảng workouts:**

• id (INTEGER, PRIMARY KEY)

• name (TEXT) - Tên bài tập

• caloriesPerUnit (REAL) - Calo/đơn vị

• unit (TEXT) - Đơn vị (phút, km, lần)

• category (TEXT) - Loại bài tập

• isCustom (INTEGER)

**Bảng workout\_entries:**

• id (INTEGER, PRIMARY KEY)

• workoutId (INTEGER, FOREIGN KEY)

• quantity (REAL)

• duration (INTEGER)

• date (INTEGER)

• caloriesBurned (REAL)

• note (TEXT)

**Bảng weight\_logs:**

• id (INTEGER, PRIMARY KEY)

• weight (REAL) - Cân nặng

• timestamp (INTEGER) - Thời điểm ghi

• note (TEXT)

**Bảng users:**

• id (INTEGER, PRIMARY KEY)

• username (TEXT, UNIQUE)

• passwordHash (TEXT)

• createdAt (INTEGER)

**2.4.3. Indexing và Performance**

Để tối ưu hiệu năng truy vấn, TrackingCaloApp sử dụng các index:

• Index trên foodId trong food\_entries

• Index trên date trong food\_entries và workout\_entries

• Index trên workoutId trong workout\_entries

• Unique index trên username trong users

• Index trên timestamp trong weight\_logs

## 2.5. Material Design 3

**2.5.1. Giới thiệu Material Design 3**

Material Design 3 (Material You) là hệ thống thiết kế mới nhất của Google, tập trung vào:

• Dynamic Color: Màu sắc tự động thích ứng với wallpaper

• Personalization: Cá nhân hóa trải nghiệm người dùng

• Accessibility: Tăng cường khả năng tiếp cận

TrackingCaloApp sử dụng Material Design 3 với theme base:

|  |
| --- |
| <style name="Base.Theme.TrackingCaloApp" parent="Theme.Material3.DayNight.NoActionBar">  <item name="colorPrimary">@color/primary</item>  <item name="colorPrimaryContainer">@color/primary\_container</item>  <item name="colorSecondary">@color/secondary</item>  ...  </style> |

**2.5.2. Các Component chính**

MaterialButton: Nút bấm với nhiều style (Filled, Outlined, Text)

• Corner radius: 12dp

• Ripple effect khi click

• Support icon và text

MaterialCardView: Card container với elevation và corner radius

• Corner radius: 16dp

• Stroke width: 1dp

• Background color theo theme

TextInputLayout: Input field với floating label

• Outlined style với corner radius 12dp

• Error state support

• Helper text và character counter

ChipGroup và Chip: Filter và selection chips

• Corner radius: 20dp

• Checked state với màu khác biệt

• Support single và multiple selection

BottomNavigationView: Navigation bar ở bottom

• 4 items: Home, Diary, Add, Profile

• Active indicator với primary container color

• Label visibility: labeled

**2.5.3. Color System**

TrackingCaloApp sử dụng color system của Material 3:

• Primary: #4CAF50 (Green) - Màu chủ đạo, đại diện cho sức khỏe

• Secondary: #FF9800 (Orange) - Màu phụ, đại diện cho năng lượng

• Surface: #FFFFFF (Light) / #1C1B1F (Dark)

• Background: #F5F5F5 (Light) / #121212 (Dark)

• Error: #B00020 - Màu cảnh báo lỗi

## 2.6. Các thư viện bên thứ ba

**2.6.1. MPAndroidChart (v3.1.0)**

MPAndroidChart là thư viện biểu đồ mạnh mẽ cho Android, hỗ trợ nhiều loại chart. TrackingCaloApp sử dụng:

• LineChart: Hiển thị xu hướng calo theo ngày

• BarChart: So sánh calo theo bữa ăn

• PieChart: Phân bố macro nutrients (Protein, Carbs, Fat)

**Tính năng nổi bật:**

• Animation khi load data

• Touch interaction (zoom, scroll)

• Customizable appearance

• Legend và description support

**2.6.2. Volley (v1.2.1)**

Volley là HTTP library của Google, được sử dụng để gọi API. Trong TrackingCaloApp, **Volley được dùng để:**

• Gọi USDA FoodData Central API để tìm kiếm thực phẩm

• Xử lý JSON response

• Quản lý request queue

**Ưu điểm của Volley:**

• Automatic scheduling of network requests

• Multiple concurrent network connections

• Request prioritization

• Cancellation request API

## 2.7. Công cụ phát triển

**2.7.1. Android Studio**

Android Studio là IDE chính thức cho phát triển Android, được xây dựng trên IntelliJ IDEA. Phiên bản sử dụng: Hedgehog 2023.1.1 trở lên.

Các tính năng chính:

• Layout Editor: Thiết kế UI trực quan

• Database Inspector: Xem và chỉnh sửa Room database

• Profiler: Phân tích performance, memory, network

• Logcat: Debug và xem logs

• Emulator: Test trên nhiều thiết bị ảo

**2.7.2. Gradle Build System**

Gradle là build system chính thức cho Android, sử dụng Kotlin DSL (build.gradle.kts). Phiên bản: 8.0+

Cấu hình dependencies trong TrackingCaloApp:

|  |
| --- |
| dependencies {  // AndroidX Core  implementation(libs.appcompat) // 1.7.1  implementation(libs.material) // 1.13.0  implementation(libs.constraintlayout) // 2.2.1    // Room Database  implementation(libs.room.runtime) // 2.6.1  annotationProcessor(libs.room.compiler)    // Lifecycle  implementation(libs.lifecycle.viewmodel) // 2.7.0  implementation(libs.lifecycle.livedata)    // UI Components  implementation(libs.recyclerview) // 1.3.2  implementation(libs.cardview) // 1.0.0  implementation(libs.viewpager2) // 1.1.0    // Third-party  implementation("com.github.PhilJay:MPAndroidChart:v3.1.0")  implementation(libs.volley) // 1.2.1  } |

**2.7.3. Git Version Control**

Git được sử dụng để quản lý source code với các best practices:

• Feature branches cho mỗi tính năng mới

• Commit messages rõ ràng và có ý nghĩa

• .gitignore để loại bỏ các file không cần thiết

## 2.8. Bảng tổng hợp công nghệ

|  |  |  |
| --- | --- | --- |
| Công nghệ | Phiên bản | Mục đích |
| Android Studio | Hedgehog+ | IDE phát triển |
| Java | 11 | Ngôn ngữ lập trình |
| Gradle | 8.0+ | Build system |
| Android SDK | 27-36 | Target platform |
| Room Database | 2.6.1 | Local database |
| LiveData | 2.7.0 | Reactive data |
| ViewModel | 2.7.0 | UI state management |
| Material Design | 1.13.0 | UI components |
| RecyclerView | 1.3.2 | List display |
| ViewPager2 | 1.1.0 | Swipeable views |
| MPAndroidChart | 3.1.0 | Charts |
| Volley | 1.2.1 | HTTP client |
| Git | - | Version control |

# CHƯƠNG III. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

## 3.1. Phân tích yêu cầu

**3.1.1. Yêu cầu chức năng**

Ứng dụng TrackingCaloApp được thiết kế để đáp ứng các yêu cầu chức năng sau:

**a) Quản lý tài khoản người dùng**

- Đăng ký tài khoản mới với username và password

- Đăng nhập với xác thực local (Room Database)

- Lưu trạng thái đăng nhập (Remember Me)

- Thiết lập thông tin cá nhân ban đầu (Onboarding)

**b) Quản lý thông tin cá nhân**

- Nhập và lưu trữ: tên, tuổi, giới tính, chiều cao, cân nặng

- Chọn mức độ hoạt động (5 cấp độ từ ít vận động đến rất năng động)

- Đặt mục tiêu cân nặng (giảm cân, duy trì, tăng cân)

- Tự động tính toán mục tiêu calo hàng ngày

**c) Theo dõi thực phẩm**

- Tìm kiếm thực phẩm từ database có sẵn (50+ món Việt Nam)

- Thêm thực phẩm tùy chỉnh (custom food)

- Ghi nhận bữa ăn theo 4 loại: sáng, trưa, tối, ăn vặt

- Nhập khối lượng (gram) và tự động tính calo, protein, carbs, fat

- Xem lịch sử ăn uống theo ngày

**d) Theo dõi bài tập**

- Tìm kiếm bài tập từ database có sẵn (27+ bài tập)

- Thêm bài tập tùy chỉnh (custom workout)

- Ghi nhận thời gian/số lượng tập luyện

- Tự động tính calo đốt cháy

- Xem lịch sử tập luyện theo ngày

**e) Theo dõi cân nặng**

- Ghi nhận cân nặng theo thời gian

- Xem biểu đồ xu hướng cân nặng

- Tính toán tiến độ đạt mục tiêu

**f) Dashboard và báo cáo**

- Hiển thị tổng quan calo trong ngày (ăn vào, đốt cháy, còn lại)

- Biểu đồ phân bố macro nutrients (Protein, Carbs, Fat)

- Hoạt động gần đây

- Tính toán và hiển thị BMI

**3.1.2. Yêu cầu phi chức năng**

**a) Hiệu năng**

- Thời gian khởi động ứng dụng dưới 3 giây

- Truy vấn database phản hồi dưới 100ms

- Hỗ trợ lưu trữ hàng nghìn entries mà không giảm hiệu năng

**b) Khả năng sử dụng**

- Giao diện tiếng Việt hoàn toàn

- Thiết kế theo Material Design 3

- Hỗ trợ Dark Mode

- Navigation đơn giản với Bottom Navigation Bar

**c) Độ tin cậy**

- Dữ liệu được lưu trữ local, không mất khi offline

- Hỗ trợ database migration khi cập nhật phiên bản

- Xử lý lỗi graceful với thông báo rõ ràng

**d) Bảo mật**

- Mật khẩu được hash trước khi lưu trữ

- Dữ liệu người dùng được lưu riêng biệt

- Không truyền dữ liệu nhạy cảm qua mạng

**e) Khả năng mở rộng**

- Kiến trúc MVVM cho phép thêm tính năng dễ dàng

- Repository pattern hỗ trợ thêm nguồn dữ liệu mới

- Modular code structure

**f) Tương thích**

- Hỗ trợ Android 8.1 (API 27) trở lên

- Tương thích với khoảng 95% thiết bị Android hiện tại

- Responsive layout cho nhiều kích thước màn hình

## 3.2. Biểu đồ Use Case

**3.2.1. Danh sách Actor**

Hệ thống có một actor chính:

- Người dùng (User): Người sử dụng ứng dụng để theo dõi calo và sức khỏe

**3.2.2. Danh sách Use Case**

**UC01: Đăng ký tài khoản**

- Actor: Người dùng

- Mô tả: Người dùng tạo tài khoản mới với username và password

- Tiền điều kiện: Chưa có tài khoản

- Hậu điều kiện: Tài khoản được tạo trong database

**UC02: Đăng nhập**

- Actor: Người dùng

- Mô tả: Người dùng đăng nhập vào hệ thống

- Tiền điều kiện: Đã có tài khoản

- Hậu điều kiện: Truy cập được các chức năng của ứng dụng

**UC03: Thiết lập thông tin cá nhân (Onboarding)**

- Actor: Người dùng

- Mô tả: Nhập thông tin cá nhân và mục tiêu sức khỏe

- Tiền điều kiện: Đã đăng nhập, chưa hoàn thành onboarding

- Hậu điều kiện: Thông tin được lưu, mục tiêu calo được tính toán

**UC04: Thêm thực phẩm vào nhật ký**

- Actor: Người dùng

- Mô tả: Ghi nhận thực phẩm đã ăn trong ngày

- Tiền điều kiện: Đã đăng nhập

- Hậu điều kiện: Food entry được lưu vào database

**UC05: Thêm bài tập vào nhật ký**

- Actor: Người dùng

- Mô tả: Ghi nhận bài tập đã thực hiện

- Tiền điều kiện: Đã đăng nhập

- Hậu điều kiện: Workout entry được lưu vào database

**UC06: Xem nhật ký ăn uống**

- Actor: Người dùng

- Mô tả: Xem danh sách thực phẩm đã ăn theo ngày

- Tiền điều kiện: Đã đăng nhập

- Hậu điều kiện: Hiển thị danh sách food entries

**UC07: Xem nhật ký tập luyện**

- Actor: Người dùng

- Mô tả: Xem danh sách bài tập đã thực hiện theo ngày

- Tiền điều kiện: Đã đăng nhập

- Hậu điều kiện: Hiển thị danh sách workout entries

**UC08: Xem dashboard**

- Actor: Người dùng

- Mô tả: Xem tổng quan calo và tiến độ trong ngày

- Tiền điều kiện: Đã đăng nhập

- Hậu điều kiện: Hiển thị thông tin tổng hợp

**UC09: Ghi nhận cân nặng**

- Actor: Người dùng

- Mô tả: Cập nhật cân nặng hiện tại

- Tiền điều kiện: Đã đăng nhập

- Hậu điều kiện: Weight log được lưu vào database

**UC10: Xem hồ sơ cá nhân**

- Actor: Người dùng

- Mô tả: Xem và chỉnh sửa thông tin cá nhân

- Tiền điều kiện: Đã đăng nhập

- Hậu điều kiện: Thông tin được hiển thị/cập nhật

**UC11: Tạo thực phẩm tùy chỉnh**

- Actor: Người dùng

- Mô tả: Thêm thực phẩm mới vào database

- Tiền điều kiện: Đã đăng nhập

- Hậu điều kiện: Custom food được lưu vào database

**UC12: Tạo bài tập tùy chỉnh**

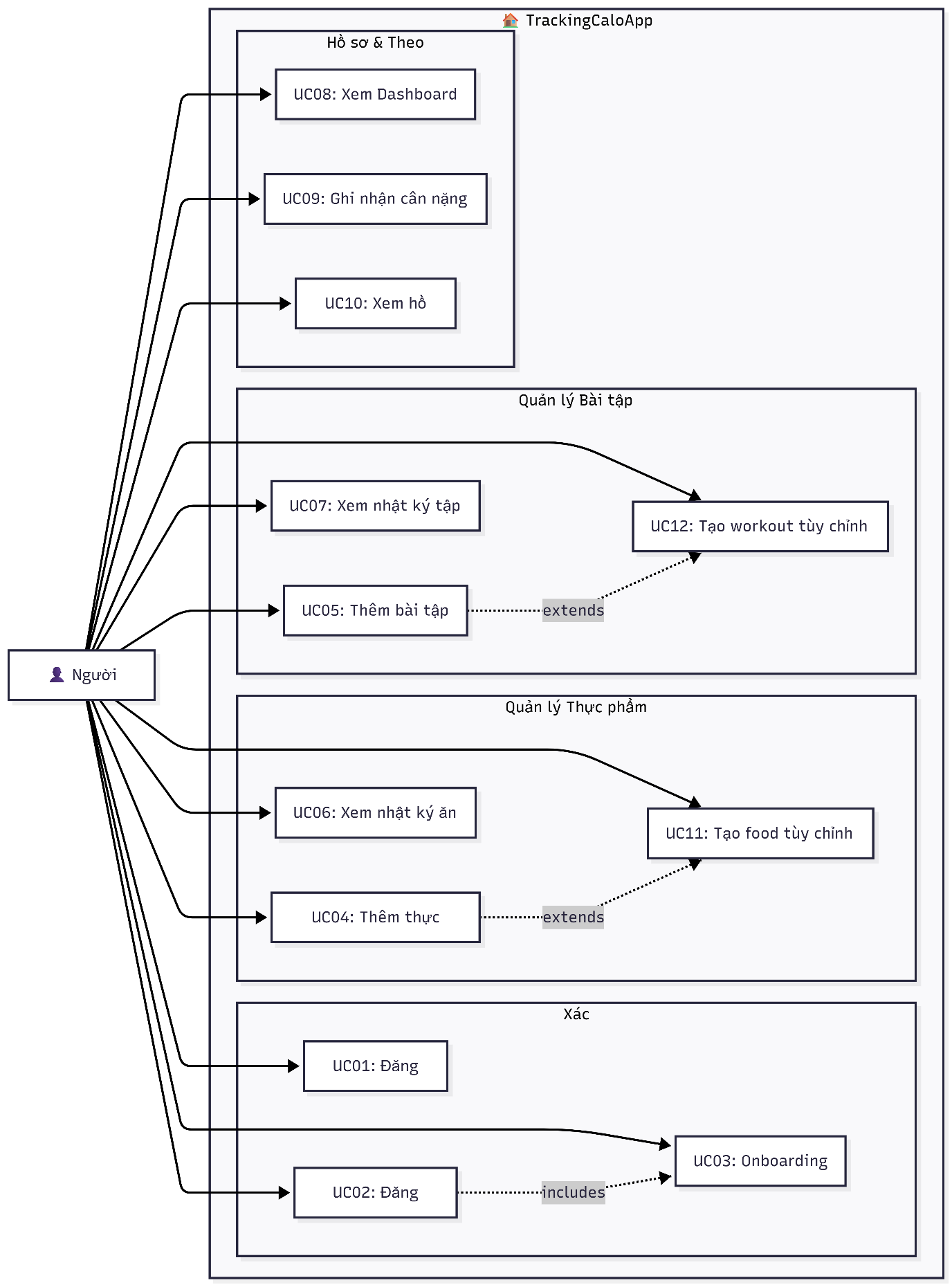
- Actor: Người dùng

- Mô tả: Thêm bài tập mới vào database

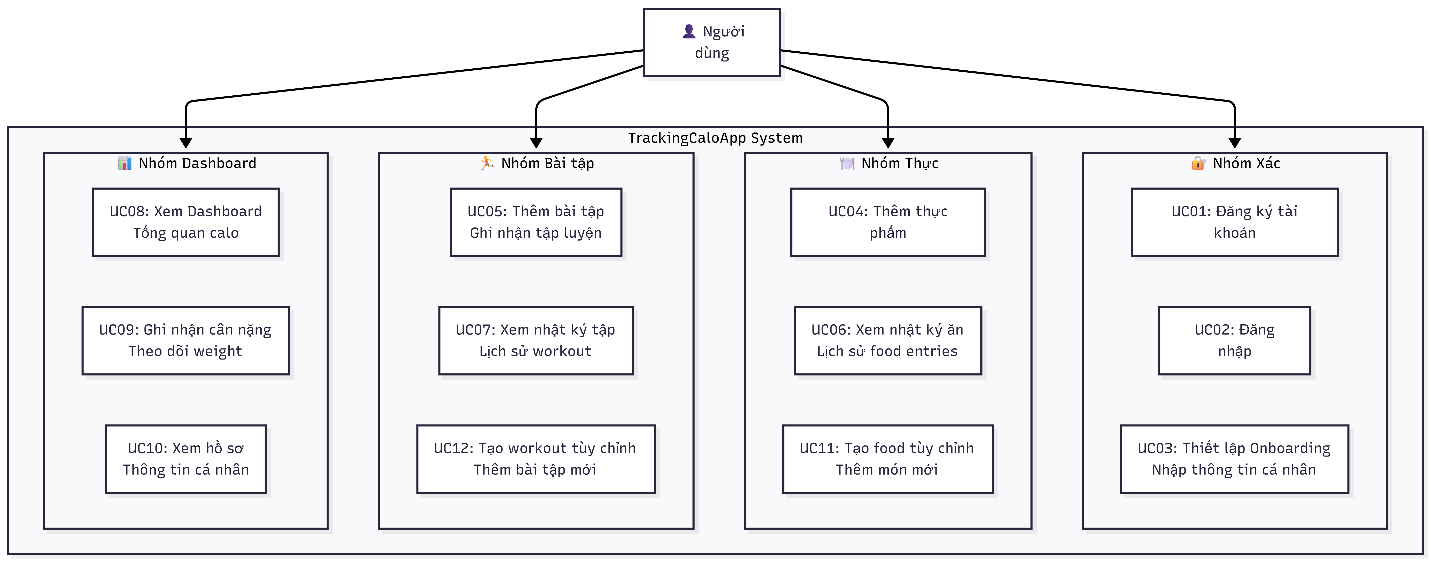
- Tiền điều kiện: Đã đăng nhập

- Hậu điều kiện: Custom workout được lưu vào database

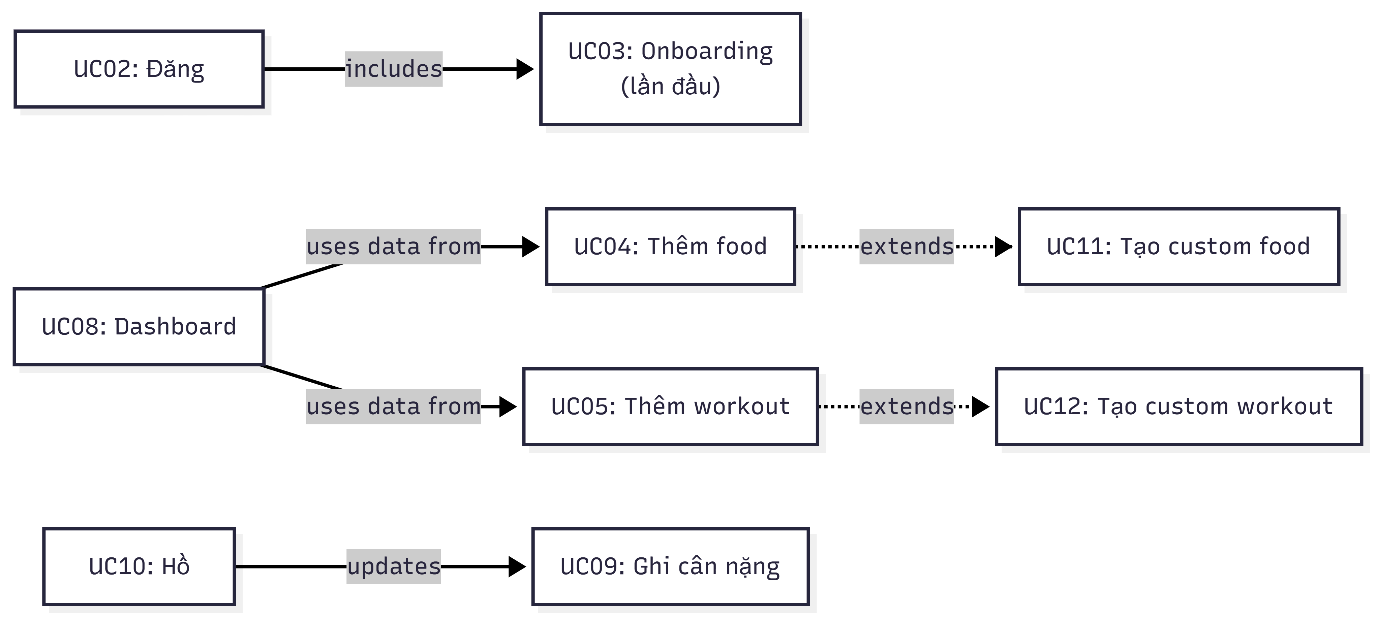
**3.2.3. Sơ đồ Use Case**



**Sơ đồ Use Case chi tiết theo nhóm chức năng:**



**Quan hệ giữa các Use Case:**

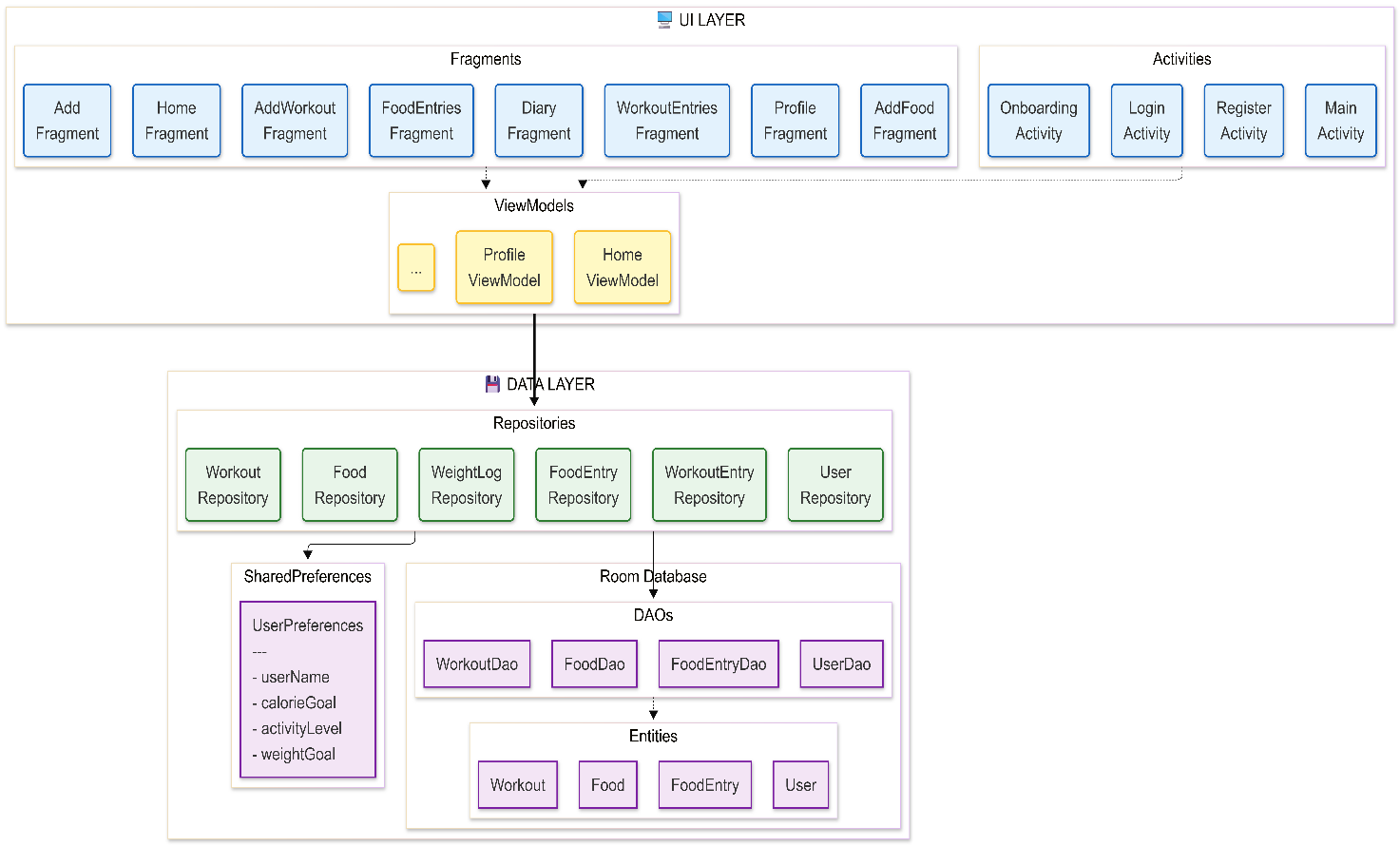


## 3.3. Thiết kế kiến trúc hệ thống

**3.3.1. Mô hình kiến trúc MVVM**

TrackingCaloApp được xây dựng theo mô hình MVVM (Model-View-ViewModel) kết hợp với Repository Pattern. Kiến trúc này được Google khuyến nghị cho phát triển ứng dụng Android.

**Sơ đồ kiến trúc tổng quan:**



**3.3.2. Luồng dữ liệu (Data Flow)**

Luồng dữ liệu trong ứng dụng tuân theo nguyên tắc một chiều (Unidirectional Data Flow):



Chi tiết các bước:

**Bước 1: User Action**

- Người dùng tương tác với UI (click button, nhập text, chọn item)

**Bước 2: UI Layer xử lý**

- Fragment/Activity nhận event

- Gọi phương thức tương ứng trong Repository

**Bước 3: Repository xử lý**

- Thực hiện business logic

- Gọi DAO để thao tác với database

**Bước 4: Database thực thi**

- Room Database thực hiện query SQL

- Trả về kết quả dưới dạng LiveData

**Bước 5: LiveData notify**

- LiveData tự động thông báo cho observers khi data thay đổi

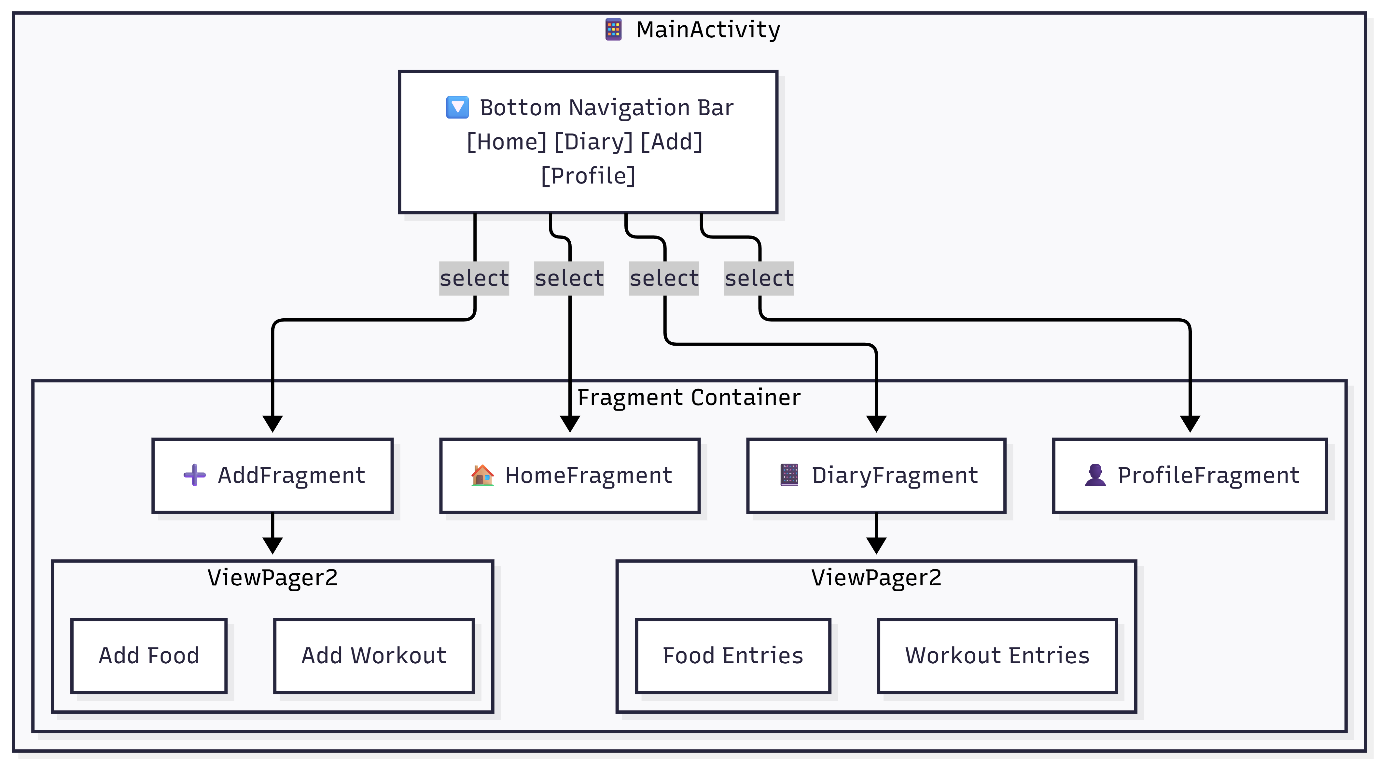
**Bước 6: UI Update**

- UI tự động cập nhật với data mới

- Không cần gọi thủ công

**3.3.3. Kiến trúc Single Activity**

Ứng dụng sử dụng kiến trúc Single Activity với Multiple Fragments:



**Ưu điểm của kiến trúc này:**

- Quản lý navigation tập trung

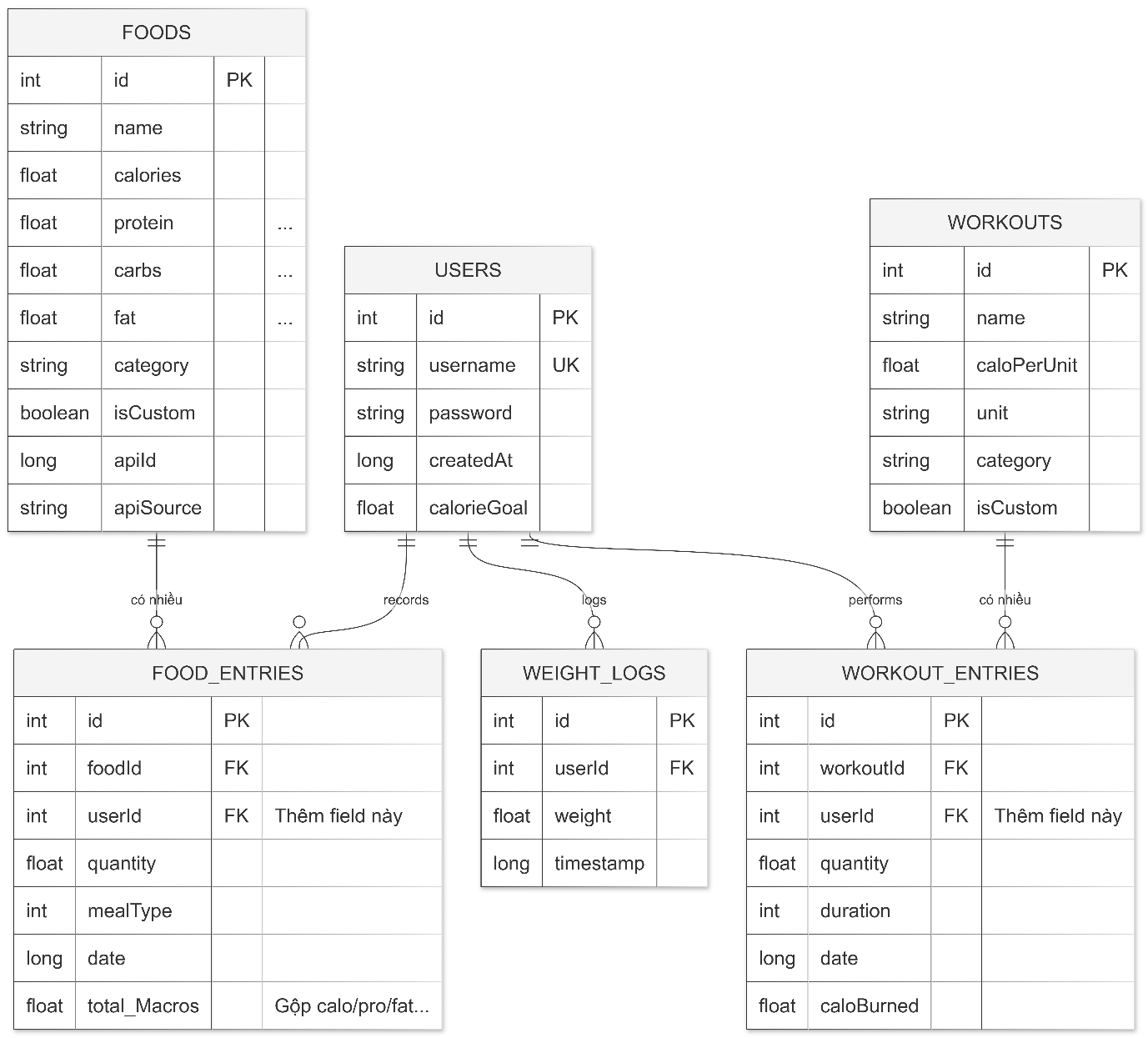
- Chia sẻ dữ liệu giữa các màn hình dễ dàng

- Giảm overhead khi chuyển màn hình

- Hỗ trợ animation mượt mà hơn

## 3.4. Thiết kế cơ sở dữ liệu

**3.4.1. Sơ đồ quan hệ thực thể (ERD)**



**3.4.2. Chi tiết các bảng**

**BẢNG FOODS (Thực phẩm)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Cột** | **Kiểu dữ liệu** | **Ràng buộc (Constraint)** | **Mô tả** |
| **id** | INTEGER | PK, AUTO | Khóa chính, tự động tăng |
| **name** | TEXT | NOT NULL | Tên thực phẩm (VD: "Phở bò") |
| **calories** | REAL | NOT NULL | Lượng Calo trên 100g |
| **protein** | REAL | NOT NULL | Lượng Protein (g) trên 100g |
| **carbs** | REAL | NOT NULL | Lượng Carbohydrate (g) trên 100g |
| **fat** | REAL | NOT NULL | Lượng Chất béo (g) trên 100g |
| **category** | TEXT | NULL | Danh mục (cơm, thịt, rau, ...) |
| **isCustom** | INTEGER | DEFAULT 0 | 1 = User tự tạo, 0 = Dữ liệu hệ thống |
| **apiId** | INTEGER | NULL | ID tham chiếu từ FatSecret API |
| **apiSource** | TEXT | NULL | Nguồn API (VD: "fatsecret" hoặc null) |
| **cachedAt** | INTEGER | DEFAULT 0 | Thời gian (Timestamp) cache từ API |

**BẢNG FOOD\_ENTRIES (Nhật ký ăn uống)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Cột** | **Kiểu dữ liệu** | **Ràng buộc (Constraint)** | **Mô tả** |
| **id** | INTEGER | PK, AUTO | Khóa chính, tự động tăng |
| **foodId** | INTEGER | FK, INDEX | Khóa ngoại tham chiếu đến bảng FOODS |
| **quantity** | REAL | NOT NULL | Khối lượng thực phẩm (gram) |
| **mealType** | INTEGER | NOT NULL | Loại bữa ăn: 0=Sáng, 1=Trưa, 2=Tối, 3=Bữa phụ (Snack) |
| **date** | INTEGER | INDEX | Thời gian lưu trữ (Timestamp milliseconds) |
| **totalCalories** | REAL | NOT NULL | Tính toán: (food.calories \* quantity) / 100 |
| **totalProtein** | REAL | NOT NULL | Tính toán: (food.protein \* quantity) / 100 |
| **totalCarbs** | REAL | NOT NULL | Tính toán: (food.carbs \* quantity) / 100 |
| **totalFat** | REAL | NOT NULL | Tính toán: (food.fat \* quantity) / 100 |

Foreign Key: foodId REFERENCES foods(id) ON DELETE CASCADE

**BẢNG WORKOUTS (Bài tập)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Cột** | **Kiểu dữ liệu** | **Ràng buộc (Constraint)** | **Mô tả** |
| **id** | INTEGER | PK, AUTO | Khóa chính, tự động tăng |
| **name** | TEXT | NOT NULL | Tên bài tập (VD: "Chạy bộ", "Đẩy tạ") |
| **caloriesPerUnit** | REAL | NOT NULL | Lượng Calo tiêu thụ trên mỗi đơn vị tính |
| **unit** | TEXT | NOT NULL | Đơn vị đo lường (VD: "phút", "km", "lần") |
| **category** | TEXT | NULL | Phân loại bài tập (Cardio, Strength, Flexibility...) |
| **isCustom** | INTEGER | DEFAULT 0 | 1 = User tự tạo, 0 = Dữ liệu mặc định của hệ thống |

**BẢNG WORKOUT\_ENTRIES (Nhật ký tập luyện)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Cột** | **Kiểu dữ liệu** | **Ràng buộc (Constraint)** | **Mô tả** |
| **id** | INTEGER | PK, AUTO | Khóa chính, tự động tăng |
| **workoutId** | INTEGER | FK, INDEX | Khóa ngoại tham chiếu đến bảng WORKOUTS |
| **quantity** | REAL | NOT NULL | Số lượng thực hiện (tùy theo đơn vị: phút, km, lần) |
| **duration** | INTEGER | NOT NULL | Tổng thời gian tập luyện thực tế (phút) |
| **date** | INTEGER | INDEX | Thời điểm tập (Timestamp milliseconds) |
| **caloriesBurned** | REAL | NOT NULL | Tính toán: workout.caloriesPerUnit \* quantity |
| **note** | TEXT | NULL | Ghi chú thêm (tùy chọn) |

Foreign Key: workoutId REFERENCES workouts(id) ON DELETE CASCADE

**BẢNG USERS (Người dùng)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Cột** | **Kiểu dữ liệu** | **Ràng buộc (Constraint)** | **Mô tả** |
| **id** | INTEGER | PK, AUTO | Khóa chính, tự động tăng |
| **username** | TEXT | UNIQUE | Tên đăng nhập (Duy nhất, không trùng lặp) |
| **passwordHash** | TEXT | NOT NULL | Mật khẩu đã được mã hóa (Không lưu text gốc) |
| **createdAt** | INTEGER | NOT NULL | Thời gian tạo tài khoản (Timestamp) |

**BẢNG WEIGHT\_LOGS (Lịch sử cân nặng)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Cột** | **Kiểu dữ liệu** | **Ràng buộc** | **Mô tả** |
| **id** | INTEGER | PK, AUTO | Khóa chính, tự động tăng |
| **weight** | REAL | NOT NULL | Cân nặng ghi nhận (kg) |
| **timestamp** | INTEGER | INDEX | Thời điểm cân (Timestamp) |
| **note** | TEXT | NULL | Ghi chú thêm (tùy chọn) |

**3.4.3. Indexing và tối ưu hiệu năng**

**Các index được tạo để tối ưu truy vấn:**

- Index trên foodId trong food\_entries: Tăng tốc JOIN với bảng foods

- Index trên date trong food\_entries: Tăng tốc query theo ngày

- Index trên workoutId trong workout\_entries: Tăng tốc JOIN với bảng workouts

- Index trên date trong workout\_entries: Tăng tốc query theo ngày

- Unique index trên username trong users: Đảm bảo username duy nhất

- Index trên timestamp trong weight\_logs: Tăng tốc query lịch sử cân nặng

**3.4.4. Database Migration**

Ứng dụng hỗ trợ migration để cập nhật schema mà không mất dữ liệu:

Migration 1 -> 2: Thêm trường isCustom cho foods và workouts

Migration 2 -> 3: Thêm bảng weight\_logs cho tính năng theo dõi cân nặng

Migration 3 -> 4: Thêm bảng users và các trường API integration

## 3.5. Thiết kế các thuật toán tính toán

**3.5.1. Thuật toán tính BMR (Basal Metabolic Rate)**

BMR là tỷ lệ trao đổi chất cơ bản - lượng calo cơ thể cần để duy trì các chức năng sống cơ bản khi nghỉ ngơi hoàn toàn.

**Công thức Mifflin-St Jeor (được sử dụng trong ứng dụng):**

Nam: BMR = (10 x weight) + (6.25 x height) - (5 x age) + 5

Nữ: BMR = (10 x weight) + (6.25 x height) - (5 x age) - 161

**Trong đó:**

- weight: Cân nặng (kg)

- height: Chiều cao (cm)

- age: Tuổi (năm)

**Ví dụ:** Nam, 25 tuổi, 70kg, 175cm

BMR = (10 x 70) + (6.25 x 175) - (5 x 25) + 5

= 700 + 1093.75 - 125 + 5

= 1673.75 calo/ngày

**Code implementation:**

|  |
| --- |
| public static float calculateBMR(float weight, float height, int age, boolean isMale) {  float bmr = (10 \* weight) + (6.25f \* height) - (5 \* age);  if (isMale) {  bmr += 5;  } else {  bmr -= 161;  }  return bmr;  **}** |

**3.5.2. Thuật toán tính TDEE (Total Daily Energy Expenditure)**

TDEE là tổng năng lượng tiêu hao hàng ngày, bao gồm BMR và hoạt động thể chất.

Công thức: TDEE = BMR x Activity Multiplier

Bảng hệ số hoạt động (Activity Multiplier):

|  |  |  |
| --- | --- | --- |
| **Mức độ hoạt động** | **Hệ số** | **Mô tả chi tiết** |
| Ít vận động | 1.2 | Công việc văn phòng, ít đi lại, gần như không tập thể dục. |
| Nhẹ nhàng | 1.375 | Tập thể dục nhẹ nhàng (đi bộ, yoga nhẹ) khoảng 1-3 ngày/tuần. |
| Trung bình | 1.55 | Tập thể dục cường độ vừa phải khoảng 3-5 ngày/tuần. |
| Năng động | 1.725 | Tập thể dục cường độ cao hoặc chơi thể thao 6-7 ngày/tuần. |
| Rất năng động | 1.9 | Tập luyện rất nặng (2 lần/ngày) hoặc làm công việc lao động chân tay nặng nhọc. |

**Ví dụ:** BMR = 1673.75, mức độ trung bình

TDEE = 1673.75 x 1.55 = 2594.31 calo/ngày

Code implementation:

|  |
| --- |
| public static float calculateTDEE(float bmr, float activityMultiplier) {  return bmr \* activityMultiplier;  **}** |

**3.5.3. Thuật toán tính mục tiêu calo hàng ngày**

Dựa trên TDEE và mục tiêu cân nặng:

|  |  |  |
| --- | --- | --- |
| **Mục tiêu cá nhân** | **Mức điều chỉnh (Calo/ngày)** | **Công thức tính toán** |
| Giảm cân | -500 calo/ngày | Daily Goal = TDEE - 500 |
| Duy trì cân nặng | 0 (Giữ nguyên) | Daily Goal = TDEE |
| Tăng cân | +500 calo/ngày | Daily Goal = TDEE + 500 |

Lưu ý: Giảm/tăng 500 calo/ngày tương đương khoảng 0.5kg/tuần

Code implementation:

|  |
| --- |
| public static int calculateDailyCalorieGoal(float tdee, String weightGoal) {  float adjustment = 0;  switch (weightGoal) {  case "lose":  adjustment = -500;  break;  case "maintain":  adjustment = 0;  break;  case "gain":  adjustment = 500;  break;  }  return Math.round(tdee + adjustment);  } |

**3.5.4. Thuật toán tính mục tiêu calo với cân nặng mục tiêu**

Khi người dùng đặt cân nặng mục tiêu cụ thể:

Hằng số: 1kg mỡ cơ thể ≈ 7700 calo

**Công thức:**

**1. Tính tổng calo cần thay đổi:**

totalCalorieChange = |currentWeight - targetWeight| x 7700

**2. Tính calo điều chỉnh mỗi ngày:**

dailyAdjustment = totalCalorieChange / daysToGoal

**3. Tính mục tiêu calo:**

Nếu giảm cân: dailyGoal = TDEE - dailyAdjustment

Nếu tăng cân: dailyGoal = TDEE + dailyAdjustment

**Giới hạn an toàn:**

- Nam: Tối thiểu 1500 calo/ngày

- Nữ: Tối thiểu 1200 calo/ngày

- Tốc độ giảm cân an toàn: 0.25 - 1.0 kg/tuần

- Tốc độ tăng cân an toàn: 0.25 - 0.5 kg/tuần

**3.5.5. Thuật toán tính BMI (Body Mass Index)**

**Công thức: BMI = weight / (height in meters)^2**

Trong đó:

- weight: Cân nặng (kg)

- height: Chiều cao (m)

**Phân loại BMI theo tiêu chuẩn châu Á (WHO 2004):**

|  |  |  |
| --- | --- | --- |
| **Phân loại** | **Chỉ số BMI (kg/m2)** | **Nguy cơ sức khỏe** |
| **Thiếu cân** | < 18.5 | Thấp (nhưng nguy cơ các bệnh khác cao) |
| **Bình thường** | 18.5 - 22.9 | Trung bình |
| **Thừa cân** | 23.0 - 24.9 | Tăng nhẹ |
| **Tiền béo phì** | 25.0 - 29.9 | Tăng vừa |
| **Béo phì độ I** | 30.0 - 34.9 | Cao |
| **Béo phì độ II** | 35.0 - 39.9 | Rất cao |
| **Béo phì độ III** | ≥ 40.0 | Nguy hiểm |

Code implementation:

|  |
| --- |
| public static float calculateBMI(float weight, float height) {  float heightInMeters = height / 100f;  return weight / (heightInMeters \* heightInMeters);  }  public static String getBMICategory(float bmi) {  if (bmi < 18.5f) return "Thiếu cân";  else if (bmi < 23f) return "Bình thường";  else if (bmi < 25f) return "Thừa cân";  else if (bmi < 30f) return "Tiền béo phì";  else if (bmi < 35f) return "Béo phì độ I";  else if (bmi < 40f) return "Béo phì độ II";  else return "Béo phì độ III";  } |

**3.5.6. Thuật toán tính cân nặng lý tưởng**

**Công thức Devine:**

Nam: IdealWeight = 50 + 2.3 x (height in inches - 60)

Nữ: IdealWeight = 45.5 + 2.3 x (height in inches - 60)

Trong đó: 1 inch = 2.54 cm

Code implementation:

|  |
| --- |
| public static float calculateIdealWeight(float height, boolean isMale) {  float heightInInches = height / 2.54f;  if (isMale) {  return 50 + 2.3f \* (heightInInches - 60);  } else {  return 45.5f + 2.3f \* (heightInInches - 60);  }  } |

**3.5.7. Thuật toán tính calo thực phẩm**

**Công thức tính calo từ khối lượng thực phẩm:**

totalCalories = (food.calories x quantity) / 100

totalProtein = (food.protein x quantity) / 100

totalCarbs = (food.carbs x quantity) / 100

totalFat = (food.fat x quantity) / 100

**Trong đó:**

- food.calories/protein/carbs/fat: Giá trị per 100g

- quantity: Khối lượng thực tế (gram)

**3.5.8. Thuật toán tính calo đốt cháy**

**Công thức:**

caloriesBurned = workout.caloriesPerUnit x quantity

Trong đó:

- workout.caloriesPerUnit: Calo đốt per đơn vị (phút/km/lần)

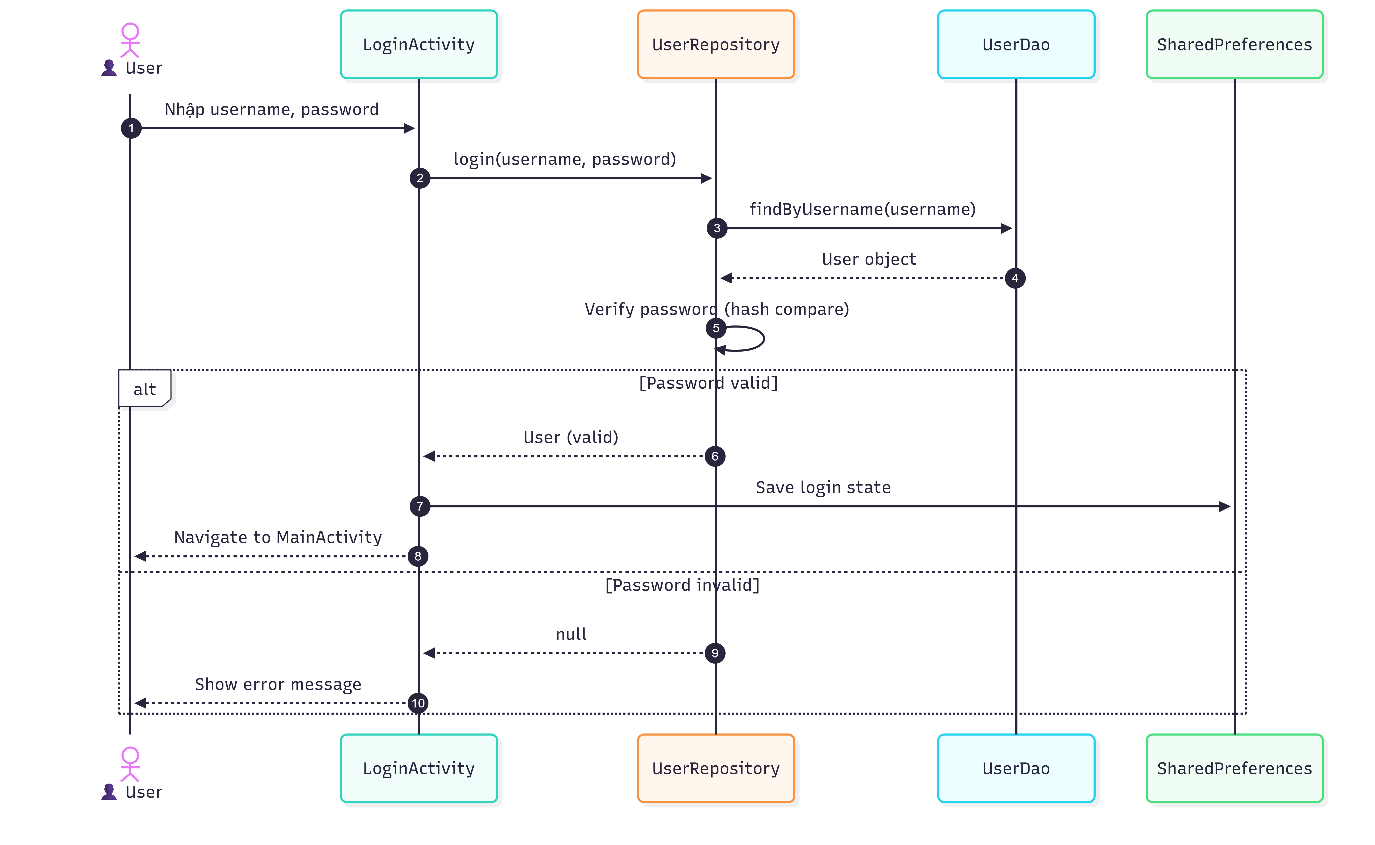
- quantity: Số lượng thực hiện

Ví dụ: Chạy bộ 30 phút, caloriesPerUnit = 10 cal/phút

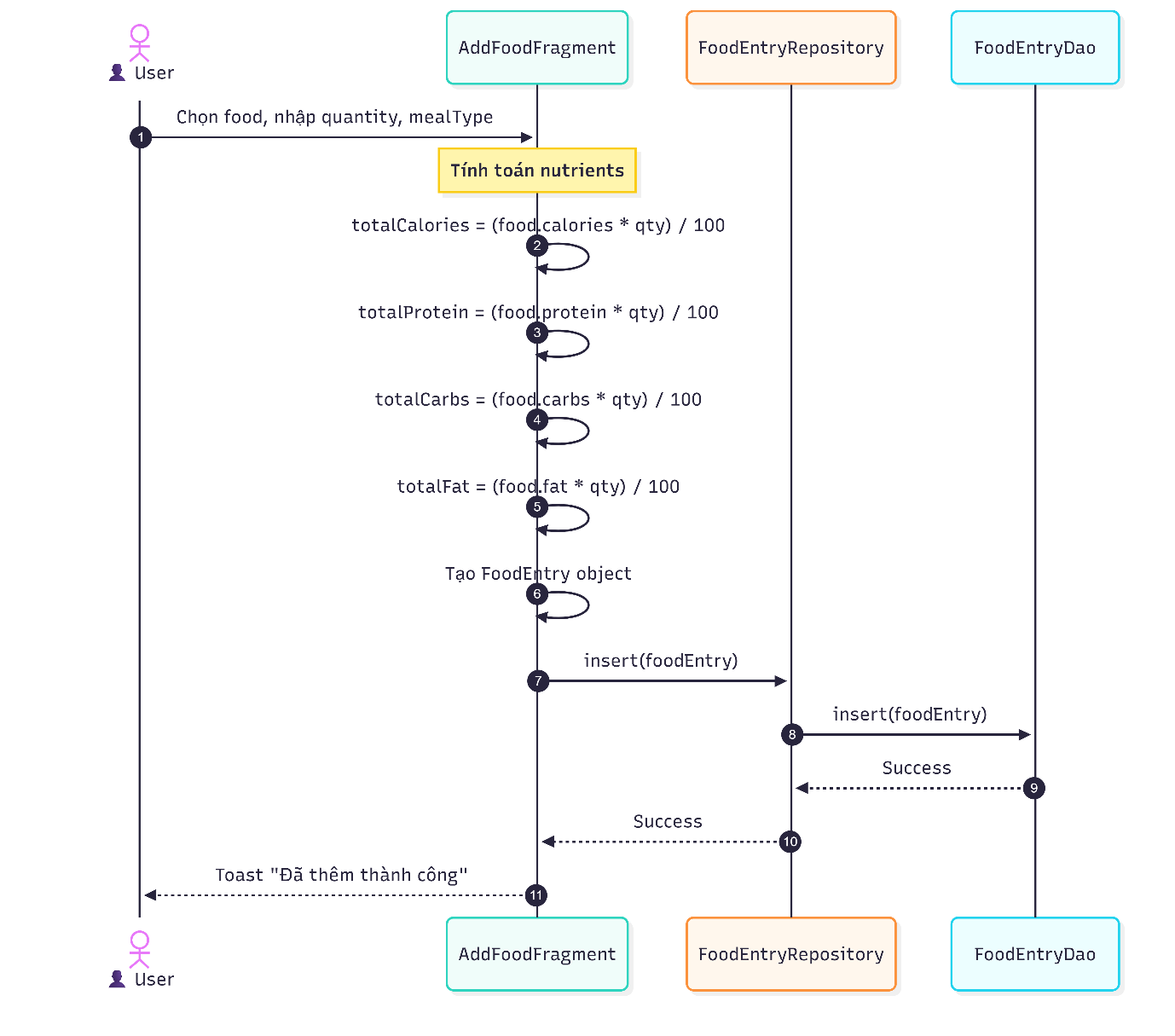
caloriesBurned = 10 x 30 = 300 calo

## 3.6. Sequence Diagrams

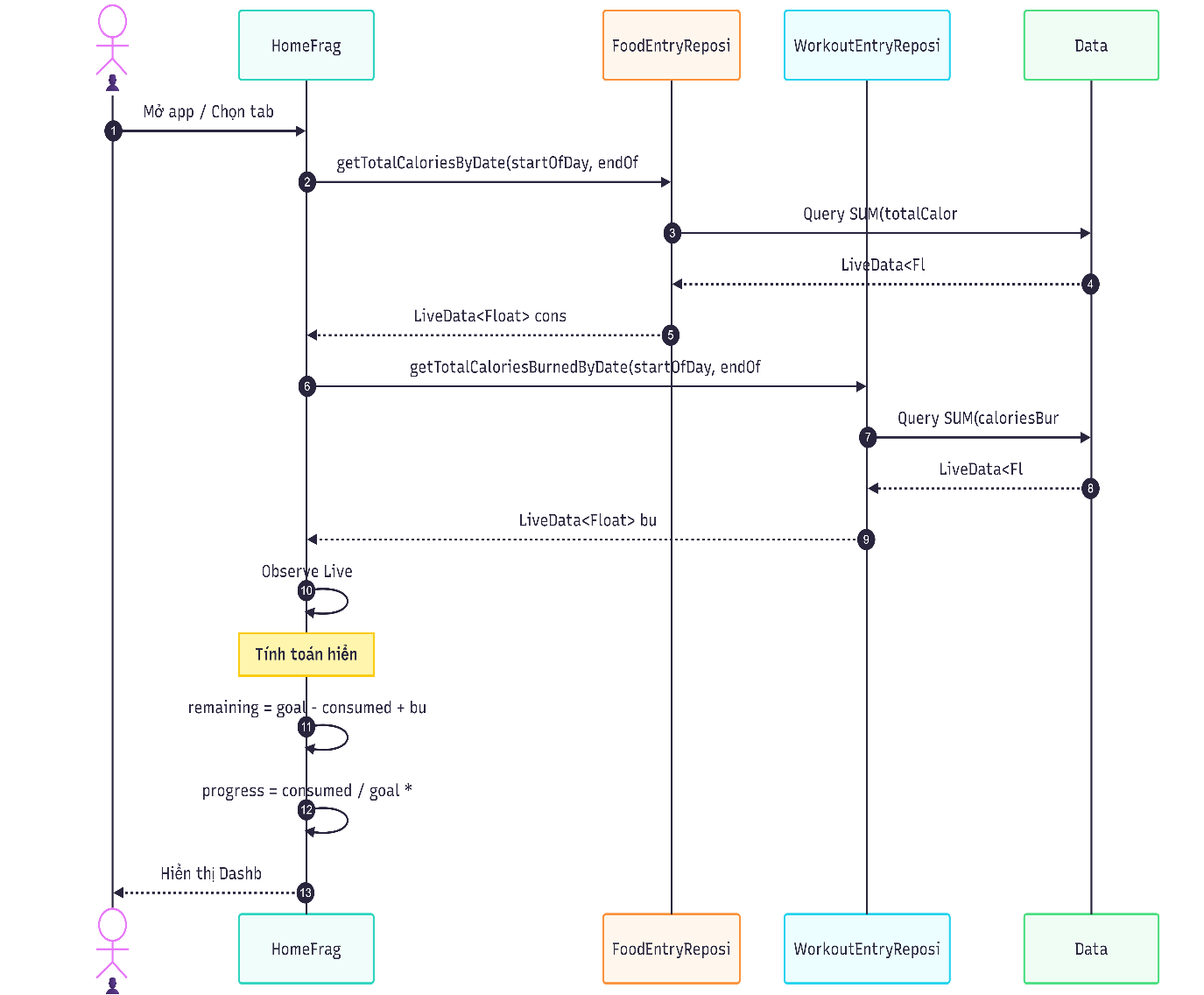
**3.6.1. Sequence Diagram: Đăng nhập**



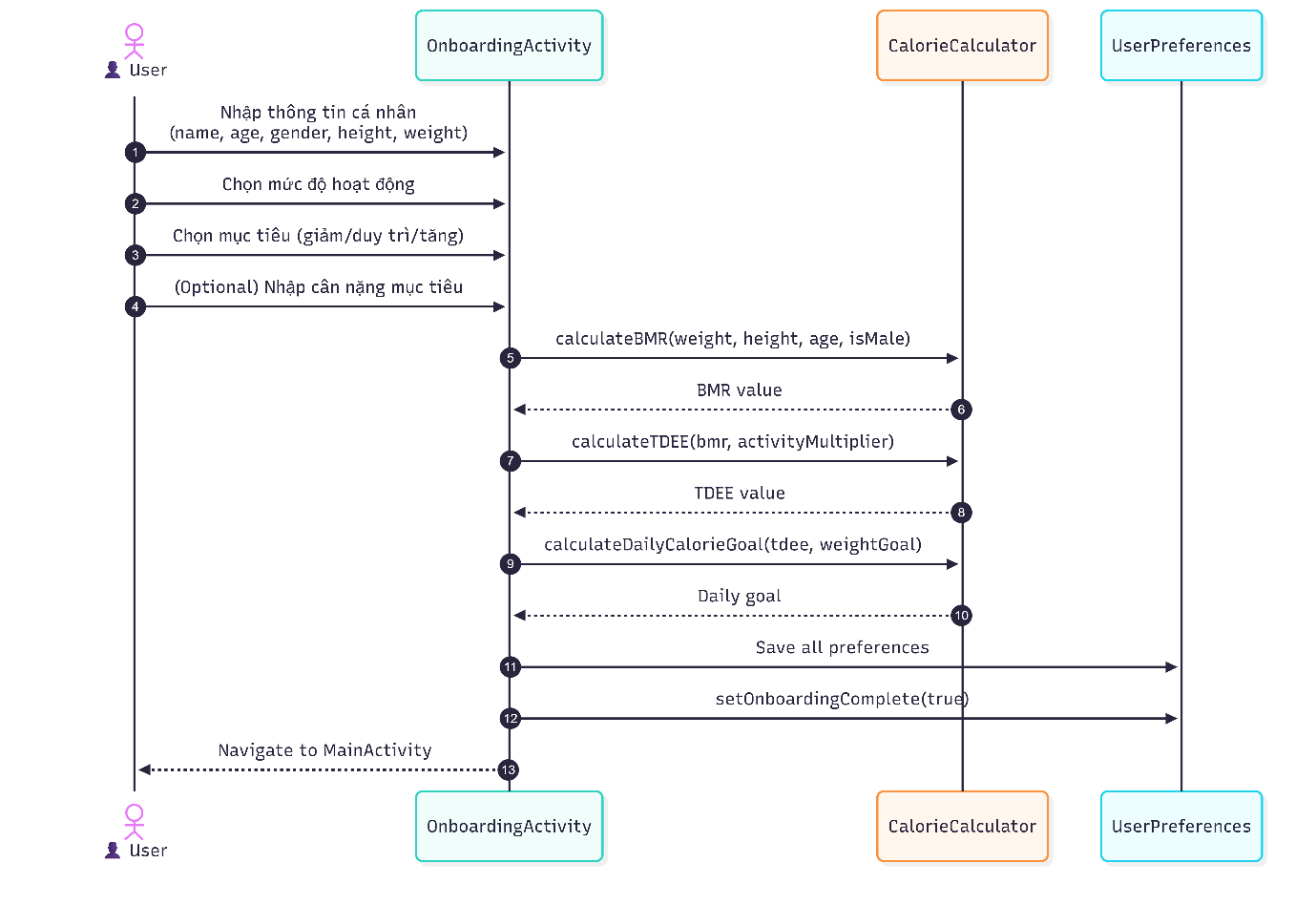
**3.6.2. Sequence Diagram: Thêm thực phẩm vào nhật ký**



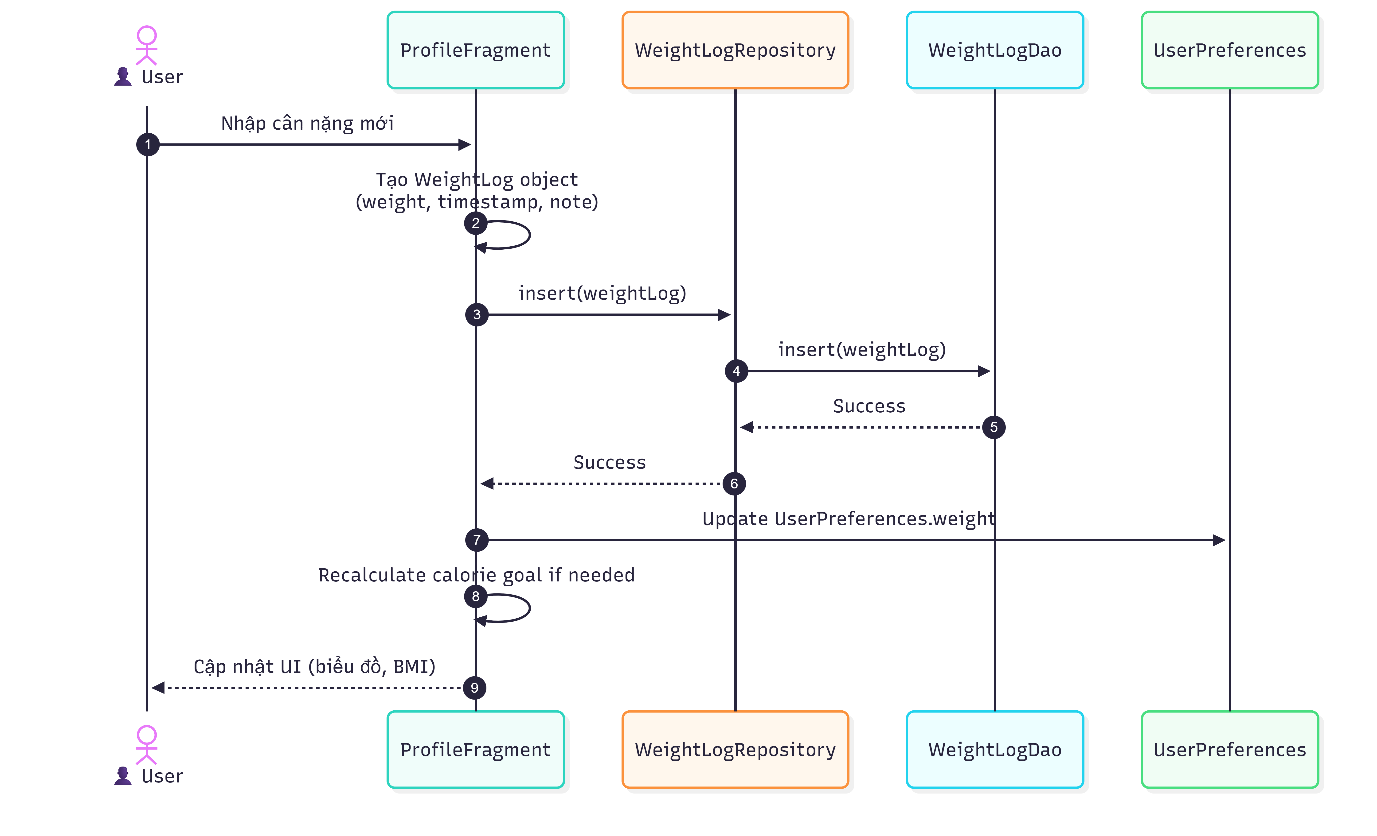
**3.6.3. Sequence Diagram: Xem Dashboard (HomeFragment)**



**3.6.4. Sequence Diagram: Onboarding (Thiết lập ban đầu)**



**3.6.5. Sequence Diagram: Ghi nhận cân nặng**



## 3.7. Tổng kết chương

Chương 3 đã trình bày chi tiết về phân tích và thiết kế hệ thống TrackingCaloApp:

Phân tích yêu cầu: Xác định 12 use cases chính và các yêu cầu phi chức năng về hiệu năng, bảo mật, khả năng mở rộng.

Thiết kế kiến trúc: Áp dụng mô hình MVVM với Repository Pattern, kiến trúc Single Activity với Multiple Fragments.

Thiết kế cơ sở dữ liệu: 6 bảng chính (foods, food\_entries, workouts, workout\_entries, users, weight\_logs) với các quan hệ và index tối ưu.

Thiết kế thuật toán: Các công thức tính BMR (Mifflin-St Jeor), TDEE, BMI, mục tiêu calo dựa trên cơ sở khoa học dinh dưỡng.

Sequence diagrams: Mô tả luồng xử lý cho các use cases quan trọng như đăng nhập, thêm thực phẩm, xem dashboard, onboarding.

Thiết kế này đảm bảo ứng dụng có kiến trúc rõ ràng, dễ bảo trì và mở rộng trong tương lai.

# CHƯƠNG IV. THIẾT KẾ GIAO DIỆN (UI/UX)

## 4.1. Nguyên tắc thiết kế UI/UX

**4.1.1. Triết lý thiết kế**

TrackingCaloApp được thiết kế theo triết lý "Simple, Clean, Intuitive" với các nguyên tắc cốt lõi:

**a) Đơn giản hóa (Simplicity)**

- Loại bỏ các yếu tố không cần thiết

- Tập trung vào chức năng chính: theo dõi calo

- Giảm thiểu số bước để hoàn thành một tác vụ

**b) Nhất quán (Consistency)**

- Sử dụng hệ thống màu sắc thống nhất

- Các component có style đồng bộ

- Pattern tương tác giống nhau trên toàn ứng dụng

**c) Phản hồi trực quan (Visual Feedback)**

- Hiển thị trạng thái rõ ràng (loading, success, error)

- Animation mượt mà khi chuyển đổi

- Ripple effect khi tương tác

**d) Khả năng tiếp cận (Accessibility)**

- Contrast ratio đạt chuẩn WCAG 2.1

- Touch target tối thiểu 48dp

- Hỗ trợ content description cho screen reader

**4.1.2. Design System**

**Ứng dụng tuân theo Material Design 3 (Material You) của Google với các đặc điểm:**

- Dynamic color theming

- Rounded corners với các cấp độ khác nhau

- Elevation và shadow tinh tế

- Typography scale chuẩn Material 3

## 4.2. Hệ thống màu sắc (Color System)

**4.2.1. Bảng màu chính (Primary Palette)**

**Ứng dụng sử dụng bảng màu Material 3 với các nhóm màu sau:**

**PRIMARY COLORS - Fresh Green (Sức khỏe/Wellness)**

|  |  |  |
| --- | --- | --- |
| **Tên biến (Name)** | **Mã màu (Hex)** | **Mục đích sử dụng** |
| **primary** | #10B981 | Màu chính, dùng cho Button, Link, các điểm nhấn quan trọng. |
| **primary\_dark** | #059669 | Màu nền thanh trạng thái (Status bar), trạng thái nút khi nhấn (Pressed). |
| **primary\_light** | #D1FAE5 | Màu nền nhẹ, huy hiệu (Badges), trạng thái Hover. |
| **primary\_container** | #ECFDF5 | Màu nền cho các khối chứa (Container), Card background. |
| **on\_primary** | #FFFFFF | Màu chữ/icon hiển thị trên nền primary. |
| **on\_primary\_container** | #065F46 | Màu chữ/icon hiển thị trên nền primary\_container. |

**SECONDARY COLORS - Warm Orange (Năng lượng/Hoạt động)**

|  |  |  |
| --- | --- | --- |
| **Tên biến (Name)** | **Mã màu (Hex)** | **Mục đích sử dụng** |
| **secondary** | #F59E0B | Màu nhấn (Accent), cảnh báo nhẹ, các nút hành động phụ (FAB). |
| **secondary\_dark** | #D97706 | Trạng thái nút khi nhấn (Pressed). |
| **secondary\_light** | #FEF3C7 | Màu nền nhẹ để làm nổi bật thông tin phụ. |
| **secondary\_container** | #FFFBEB | Màu nền cho các khối chứa phụ (Secondary Container). |

**TERTIARY COLORS - Cool Blue (Thông tin)**

|  |  |  |
| --- | --- | --- |
| **Tên biến (Name)** | **Mã màu (Hex)** | **Mục đích sử dụng** |
| **tertiary** | #3B82F6 | Hiển thị thông tin chung, đường link, thanh tiến trình (progress bar). |
| **tertiary\_dark** | #2563EB | Trạng thái nhấn hoặc các yếu tố cần độ tương phản cao hơn. |
| **tertiary\_light** | #DBEAFE | Màu nền nhẹ cho các thông báo tin tức (Info alerts). |

**4.2.2. Màu ngữ nghĩa (Semantic Colors)**

**CALORIE COLORS - Màu theo ngữ cảnh calo**

|  |  |  |
| --- | --- | --- |
| **Tên biến (Resource Name)** | **Mã màu (Hex)** | **Mục đích sử dụng** |
| **calories\_consumed** | #EF4444 | Biểu thị lượng Calo nạp vào (Màu đỏ - mang ý nghĩa cảnh báo/tăng tích lũy). |
| **calories\_burned** | #10B981 | Biểu thị lượng Calo đốt cháy (Màu xanh - tích cực). |
| **calories\_remaining** | #3B82F6 | Biểu thị phần Calo còn lại được phép ăn (Màu xanh dương - trung tính). |
| **calories\_net** | #8B5CF6 | Biểu thị Calo Net = (Nạp vào - Đốt cháy). |

**MEAL TYPE COLORS - Màu theo loại bữa ăn**

|  |  |  |
| --- | --- | --- |
| **Tên biến (Resource Name)** | **Mã màu (Hex)** | **Mục đích sử dụng** |
| **breakfast** | #F97316 | Bữa sáng (Cam - năng lượng khởi đầu ngày mới). |
| **lunch** | #06B6D4 | Bữa trưa (Cyan - sự tươi mát, cân bằng). |
| **dinner** | #8B5CF6 | Bữa tối (Tím - sự thư giãn, kết thúc ngày). |
| **snack** | #84CC16 | Ăn vặt (Xanh lá mạ - nhẹ nhàng, healthy). |

**WORKOUT CATEGORY COLORS - Màu theo loại bài tập**

|  |  |  |
| --- | --- | --- |
| **Tên biến (Resource Name)** | **Mã màu (Hex)** | **Mục đích sử dụng** |
| **cardio** | #EF4444 | Cardio/Tim mạch (Đỏ - thể hiện nhịp tim cao, cường độ mạnh). |
| **strength** | #6366F1 | Strength/Sức mạnh (Indigo - sự rắn rỏi, tập tạ). |
| **flexibility** | #14B8A6 | Flexibility/Dẻo dai (Teal - sự linh hoạt, Yoga). |

**NUTRITION COLORS - Màu dinh dưỡng**

|  |  |  |
| --- | --- | --- |
| **Tên biến (Resource Name)** | **Mã màu (Hex)** | **Mục đích sử dụng** |
| **protein** | #EC4899 | Chất đạm (Hồng - thường dùng trong các app thể hình cho cơ bắp). |
| **carbs** | #F59E0B | Tinh bột (Vàng cam - nguồn năng lượng chính). |
| **fat** | #A855F7 | Chất béo (Tím - năng lượng dự trữ). |

**4.2.3. Màu Surface và Background**

**SURFACE COLORS**

|  |  |  |
| --- | --- | --- |
| **Tên biến (Resource Name)** | **Mã màu (Hex)** | **Mục đích sử dụng** |
| **background** | #FAFAFA | Màu nền tổng thể của ứng dụng (Off-white), giúp giảm mỏi mắt so với trắng tinh. |
| **surface** | #FFFFFF | Nền của các thẻ (Card), hộp thoại (Dialog), menu (Trắng tuyệt đối để nổi bật trên nền background). |
| **surface\_variant** | #F5F5F5 | Nền phụ, dùng cho các khối ít quan trọng hơn hoặc làm nền phân cách. |
| **surface\_container** | #F3F4F6 | Nền cho các khối chứa (Container) bao quanh nội dung. |
| **outline** | #E5E7EB | Viền của thẻ (Card Border), đường kẻ phân cách (Divider). |

**TEXT COLORS**

|  |  |  |
| --- | --- | --- |
| **Tên biến (Resource Name)** | **Mã màu (Hex)** | **Mục đích sử dụng** |
| **text\_primary** | #111827 | **Văn bản chính:** Tiêu đề, nội dung quan trọng nhất (Gần như đen nhưng dịu hơn). |
| **text\_secondary** | #6B7280 | **Văn bản phụ:** Mô tả ngắn, thông tin bổ sung (Màu xám trung tính). |
| **text\_tertiary** | #9CA3AF | **Văn bản cấp 3:** Các gợi ý (Hint), label ít quan trọng. |
| **text\_hint** | #D1D5DB | **Văn bản giữ chỗ:** Placeholder trong ô nhập liệu (Input field). |

**STATUS COLORS**

|  |  |  |
| --- | --- | --- |
| **Tên biến (Resource Name)** | **Mã màu (Hex)** | **Mục đích sử dụng** |
| **success** | #10B981 | Thông báo thành công (Hoàn thành mục tiêu, lưu dữ liệu xong). |
| **warning** | #F59E0B | Cảnh báo (Chưa lưu, sắp vượt quá calo). |
| **error** | #EF4444 | Báo lỗi (Mất mạng, nhập sai dữ liệu, lỗi hệ thống). |
| **info** | #3B82F6 | Thông tin (Thông báo cập nhật, hướng dẫn). |

## 4.3. Typography và Spacing

**4.3.1. Typography Scale**

Ứng dụng sử dụng font family sans-serif (Roboto) với các cấp độ:

**HEADLINE STYLES**

|  |  |  |  |
| --- | --- | --- | --- |
| **Style Name** | **Kích thước (Size)** | **Độ đậm (Weight)** | **Mục đích sử dụng** |
| **Headline.Large** | 32sp | Medium | Tiêu đề trang chính (Screen Title). |
| **Headline.Medium** | 28sp | Medium | Tiêu đề phân đoạn lớn (Section Header). |
| **Headline.Small** | 24sp | Medium | Tiêu đề của các thẻ lớn (Card Title). |
| **Greeting** | 26sp | Bold | Lời chào đặc biệt (VD: *"Xin chào, [Name]! 👋"*). |

**TITLE STYLES**

|  |  |  |  |
| --- | --- | --- | --- |
| **Style Name** | **Kích thước (Size)** | **Độ đậm (Weight)** | **Mục đích sử dụng** |
| **Title.Large** | 22sp | Medium | Tiêu đề lớn trong luồng nội dung. |
| **Title.Medium** | 16sp | Medium | Tiêu đề thẻ con, tiêu đề section nhỏ. |
| **Title.Small** | 14sp | Medium | Tiêu đề danh sách (List item title). |

**BODY STYLES**

|  |  |  |  |
| --- | --- | --- | --- |
| **Style Name** | **Kích thước (Size)** | **Độ đậm (Weight)** | **Mục đích sử dụng** |
| **Body.Large** | 16sp | Normal (Regular) | Nội dung chính, đoạn văn mô tả dài. |
| **Body.Medium** | 14sp | Normal (Regular) | Nội dung phụ, thông tin bổ sung. |
| **Body.Small** | 12sp | Normal (Regular) | Chú thích (Caption), văn bản gợi ý (Hint). |

**LABEL STYLES**

|  |  |  |  |
| --- | --- | --- | --- |
| **Style Name** | **Kích thước (Size)** | **Độ đậm (Weight)** | **Mục đích sử dụng** |
| **Label.Large** | 14sp | Medium | Văn bản trong nút bấm (Button Text), Tabs. |
| **Label.Medium** | 12sp | Medium | Văn bản trong Chip, huy hiệu (Badges). |
| **Label.Small** | 11sp | Medium | Nhãn rất nhỏ, chú thích icon. |

**CALORIE DISPLAY STYLES**

|  |  |  |  |
| --- | --- | --- | --- |
| **Style Name** | **Kích thước (Size)** | **Độ đậm (Weight)** | **Mục đích sử dụng** |
| **Calorie.Large** | 48sp | Bold | **Số Calo tổng (Hero Display):** Hiển thị ở dashboard chính. |
| **Calorie.Medium** | 24sp | Bold | **Số Calo thẻ:** Hiển thị tổng calo của từng bữa ăn. |
| **Calorie.Small** | 18sp | Bold | **Số Calo mục:** Hiển thị calo của từng món ăn/bài tập. |

**4.3.2. Spacing System**

Ứng dụng sử dụng hệ thống spacing 4dp base:

**SPACING VALUES**

|  |  |  |
| --- | --- | --- |
| **Tên biến (Resource Name)** | **Giá trị** | **Mục đích sử dụng** |
| **spacing\_xxs** | 2dp | Khoảng cách rất nhỏ (dùng cho các chi tiết tinh chỉnh). |
| **spacing\_xs** | 4dp | Khoảng cách nhỏ (giữa icon và text). |
| **spacing\_sm** | 8dp | Khoảng cách tiêu chuẩn giữa các phần tử nhỏ (Standard padding). |
| **spacing\_md** | 12dp | Khoảng cách trung bình. |
| **spacing\_base** | 16dp | **Padding chuẩn** cho mép màn hình, bên trong thẻ (Card). |
| **spacing\_lg** | 20dp | Khoảng cách lớn. |
| **spacing\_xl** | 24dp | Khoảng cách rất lớn (giữa các nhóm nội dung). |
| **spacing\_2xl** | 32dp | Khoảng cách phân chia các Section (Phần) lớn. |

**4.3.3. Corner Radius**

**RADIUS VALUES**

|  |  |  |
| --- | --- | --- |
| **Tên biến (Resource Name)** | **Giá trị** | **Mục đích sử dụng** |
| **radius\_sm** | 8dp | Dùng cho các thành phần nhỏ: Chips, Badges, Tag. |
| **radius\_base** | 12dp | Dùng cho: Nút bấm (Button), Ô nhập liệu (Input Field). |
| **radius\_lg** | 16dp | Dùng cho: Thẻ (Card), Hộp thoại (Dialog). |
| **radius\_xl** | 20dp | Dùng cho: Thẻ lớn, Bottom Sheet (Menu trượt từ dưới lên). |
| **radius\_full** | 50% | Dùng cho: Ảnh đại diện (Avatar) tròn, nút tròn. |

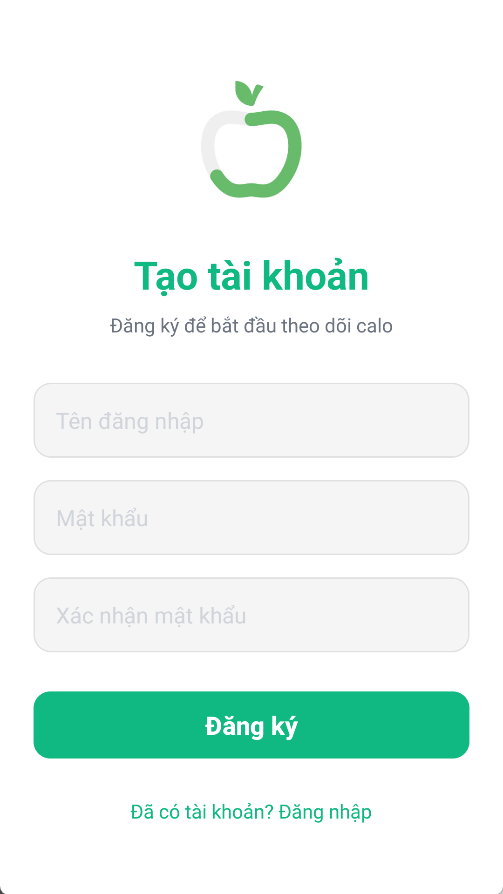
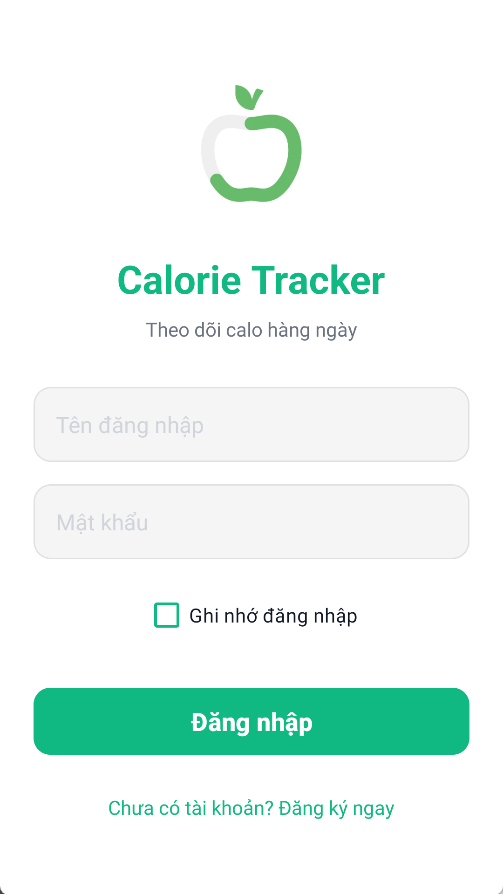
**4.3.4. Elevation**

**ELEVATION VALUES**

|  |  |  |
| --- | --- | --- |
| **Tên biến (Resource Name)** | **Giá trị** | **Mục đích sử dụng** |
| **elevation\_none** | 0dp | Thẻ phẳng (Flat Cards) thường dùng kèm với đường viền (Stroke). |
| **elevation\_sm** | 2dp | Đổ bóng nhẹ: Thẻ nội dung thông thường. |
| **elevation\_md** | 4dp | Đổ bóng vừa: Nút bấm nổi (Raised Button), Card nổi bật. |
| **elevation\_lg** | 8dp | Đổ bóng lớn: Thanh điều hướng dưới (Bottom Nav), Nút hành động nổi (FAB). |

## 4.4. Thiết kế các màn hình chính

**4.4.1. Màn hình Login (LoginActivity)**



**Mô tả:** Màn hình đăng nhập cho người dùng đã có tài khoản

**Cấu trúc layout:**

- Logo và tên ứng dụng (center aligned)

- TextInputLayout cho Username (outlined style)

- TextInputLayout cho Password (outlined style, password toggle)

- Checkbox "Ghi nhớ đăng nhập"

- Button "Đăng nhập" (primary style, full width)

- Link "Chưa có tài khoản? Đăng ký"

**Đặc điểm thiết kế:**

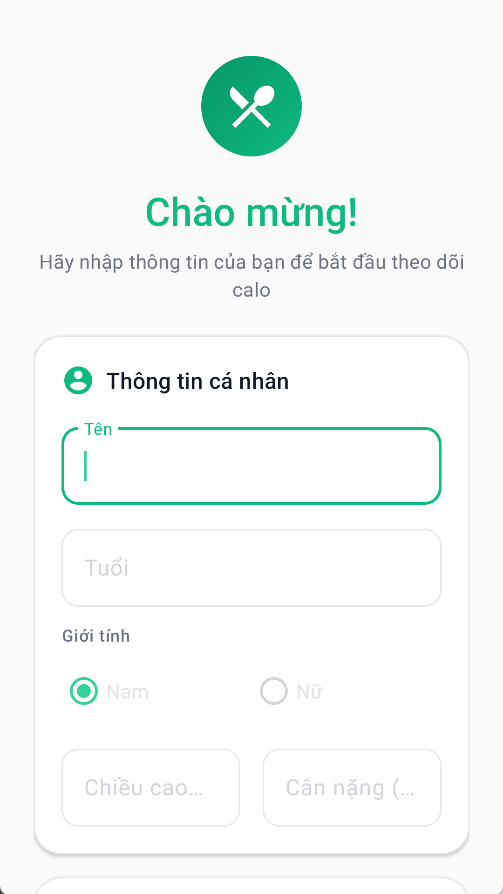
- Background: @color/background (#FAFAFA)

- Input fields: Outlined style với corner radius 12dp

- Button: Primary color (#10B981), height 56dp

- Vertical spacing: 16dp giữa các elements

**4.4.2. Màn hình Onboarding (OnboardingActivity)**



**Mô tả:** Màn hình thiết lập thông tin cá nhân lần đầu

**Cấu trúc layout:**

- Progress indicator (step 1/4, 2/4, ...)

- Title và description cho mỗi bước

- Input fields tương ứng với từng bước:

+ Bước 1: Tên, tuổi, giới tính

+ Bước 2: Chiều cao, cân nặng

+ Bước 3: Mức độ hoạt động

+ Bước 4: Mục tiêu cân nặng

- Button "Tiếp tục" / "Hoàn thành"

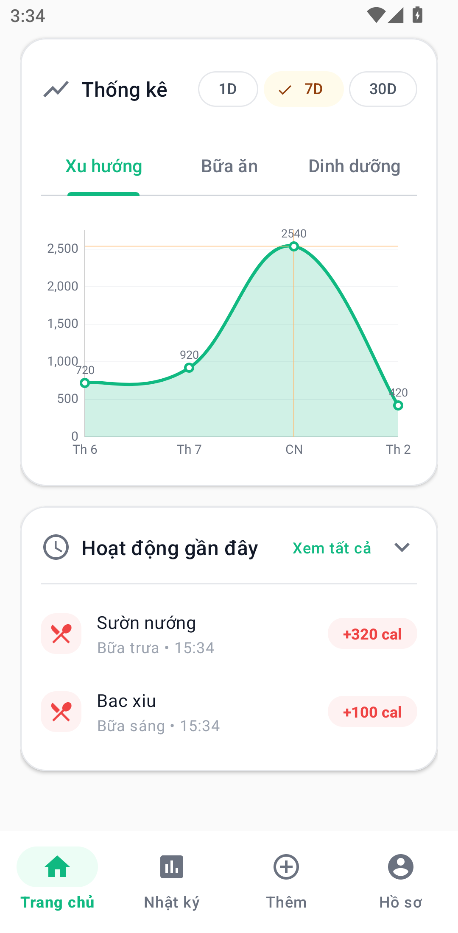
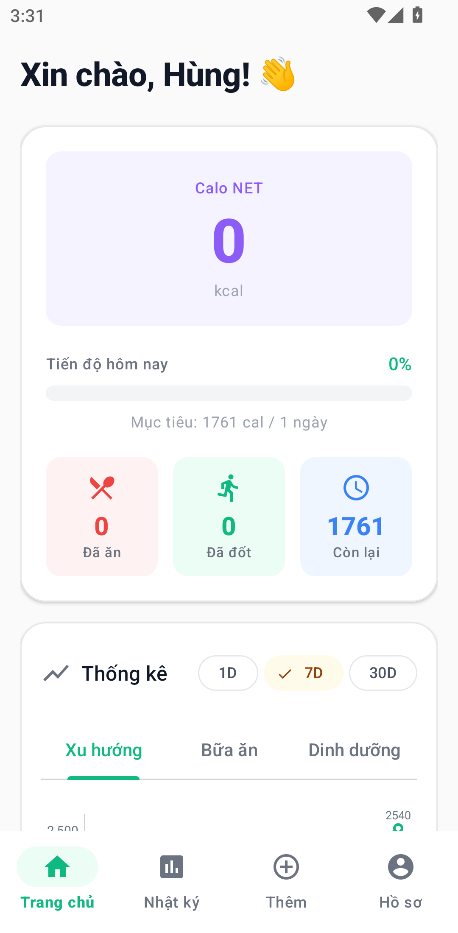
**Đặc điểm thiết kế:**

- Sử dụng ViewPager2 cho smooth transitions

- Progress bar với primary color

- Dropdown menus cho selections (ExposedDropdownMenu style)

**4.4.3. Màn hình Home (HomeFragment)**



**Mô tả:** Dashboard chính hiển thị tổng quan calo trong ngày

**Cấu trúc layout (từ trên xuống):**

**a) Header Section**

- Greeting text: "Xin chào! 👋" (TextAppearance.App.Greeting)

- Sử dụng emoji để tạo cảm giác thân thiện

**b) Hero Card - Tổng quan Calo**

- NET Calories display (hero style, 48sp, màu tím #8B5CF6)

- Progress bar với OverflowProgressBar custom view

- Hiển thị phần trăm tiến độ

- Warning message khi vượt mục tiêu

**c) Stats Row (3 columns)**

- Ăn vào (consumed): Icon + số + label, màu đỏ #EF4444

- Đốt cháy (burned): Icon + số + label, màu xanh #10B981

- Còn lại (remaining): Icon + số + label, màu xanh dương #3B82F6

**d) Statistics Card - Biểu đồ**

- ChipGroup để chọn period (1D, 7D, 30D)

- TabLayout để chọn loại chart (Xu hướng, Bữa ăn, Dinh dưỡng)

- ViewFlipper chứa 3 loại chart:

+ LineChart: Xu hướng calo theo thời gian

+ BarChart: Phân bố calo theo bữa ăn

+ PieChart: Tỷ lệ dinh dưỡng (Protein, Carbs, Fat)

**e) Recent Activities Card**

- Header với icon và "Xem tất cả" link

- Expand/Collapse functionality

- RecyclerView hiển thị hoạt động gần đây

- Empty state khi chưa có dữ liệu

**Đặc điểm thiết kế:**

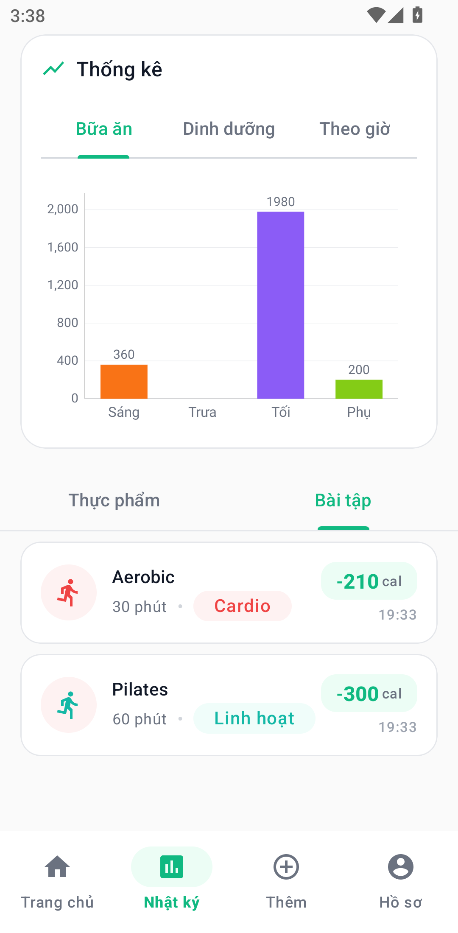
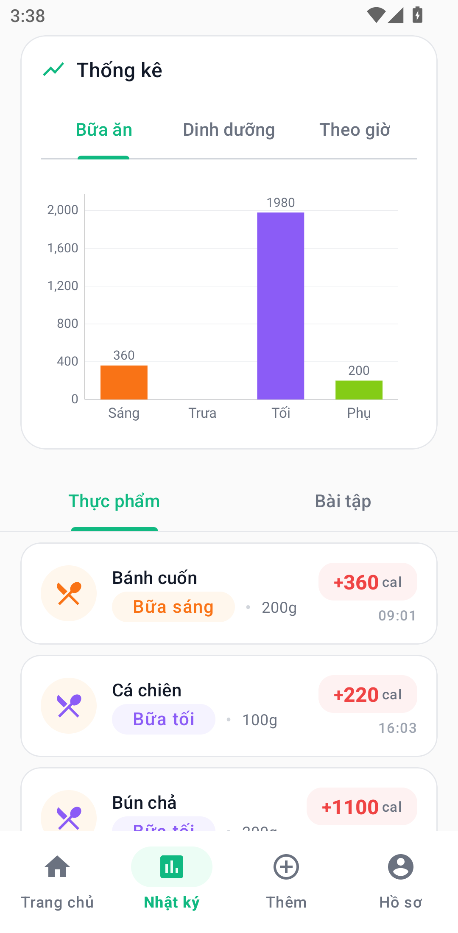
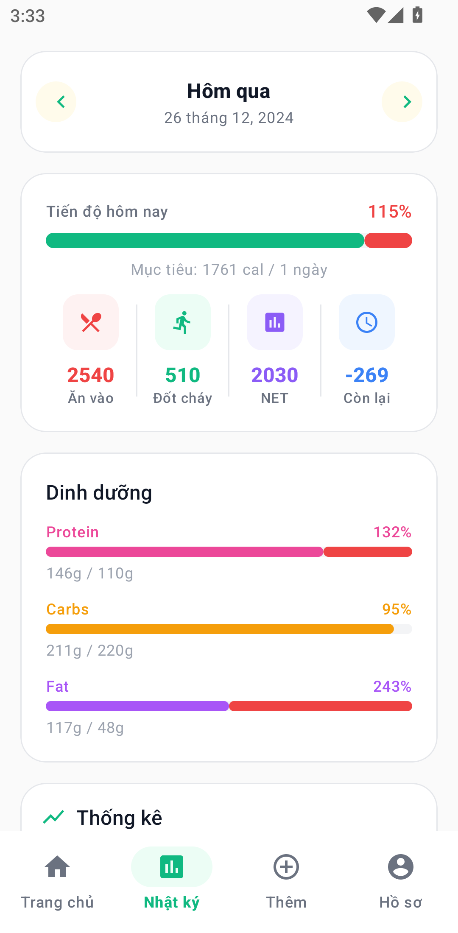
- NestedScrollView cho smooth scrolling

- Cards với stroke 1dp và corner radius 16dp

- Sử dụng MPAndroidChart library cho biểu đồ

- Padding bottom để tránh bị che bởi Bottom Navigation

**4.4.4. Màn hình Diary (DiaryFragment)**



**Mô tả:** Nhật ký ăn uống và tập luyện theo ngày

**Cấu trúc layout:**

**a) Date Selector Card**

- Button Previous Day (chevron left)

- Date display (center): "Hôm nay" + "26 tháng 12, 2024"

- Button Next Day (chevron right)

- Clickable để mở DatePicker

**b) Summary Card**

- Progress bar với phần trăm

- Stats row: Ăn vào | Đốt cháy | NET | Còn lại

- Mỗi stat có icon container với background color tương ứng

**c) Macro Nutrients Card**

- Title "Dinh dưỡng"

- 3 progress bars cho Protein, Carbs, Fat

- Mỗi macro có: label, percent, progress bar, value (0g / 0g)

- Màu sắc riêng biệt cho từng macro

**d) Chart Card**

- TabLayout: Bữa ăn | Dinh dưỡng | Theo giờ

- ViewFlipper với BarChart, PieChart, LineChart

**e) Entries TabLayout**

- Tab "Thực phẩm" và "Bài tập"

- ViewPager2 chứa FoodEntriesFragment và WorkoutEntriesFragment

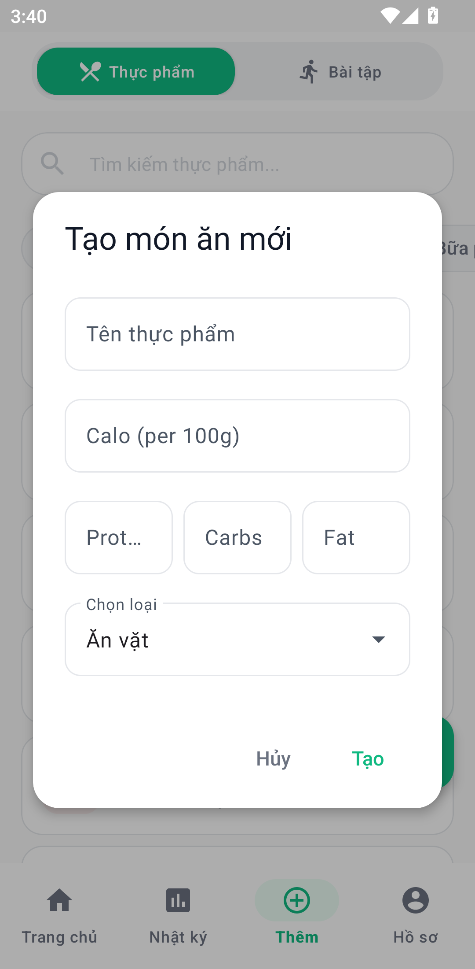
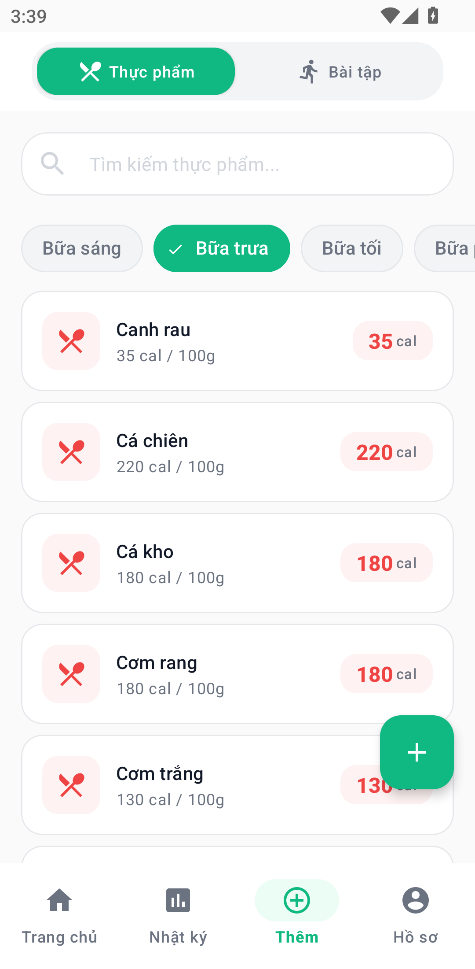
**Đặc điểm thiết kế:**

- Date navigation trực quan với buttons

- Màu sắc semantic cho từng loại thông tin

- Grouped entries theo meal type

**4.4.5. Màn hình Add (AddFragment)**



**Mô tả:** Màn hình thêm thực phẩm hoặc bài tập mới

**Cấu trúc layout:**

- TabLayout với 2 tabs: "Thực phẩm" và "Bài tập"

- ViewPager2 chứa AddFoodFragment và AddWorkoutFragment

**AddFoodFragment:**

- SearchView để tìm kiếm thực phẩm

- ChipGroup để filter theo meal type (Sáng, Trưa, Tối, Snack)

- Button "Tạo món mới" để thêm custom food

- RecyclerView hiển thị danh sách thực phẩm

- Empty state khi không có kết quả

**AddWorkoutFragment:**

- SearchView để tìm kiếm bài tập

- ChipGroup để filter theo category (Cardio, Strength, Flexibility)

- Button "Tạo bài tập mới" để thêm custom workout

- RecyclerView hiển thị danh sách bài tập

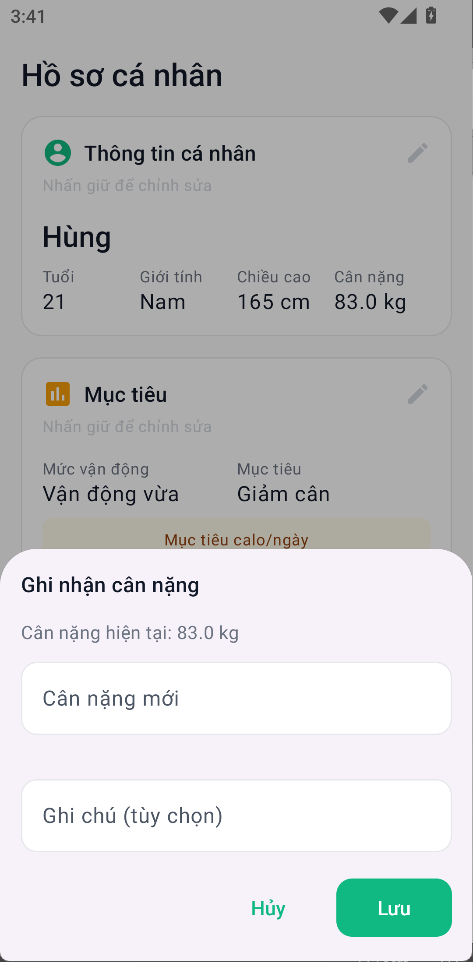
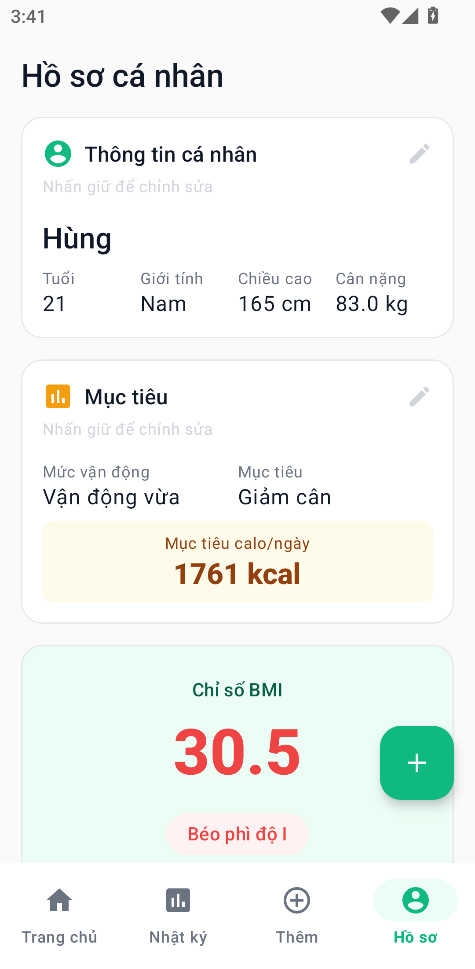
**Đặc điểm thiết kế:**

- Chips với màu sắc tương ứng meal type/workout category

- Search với debounce để tối ưu performance

- Smooth animations khi filter

**4.4.6. Màn hình Profile (ProfileFragment)**



**Mô tả:** Quản lý thông tin cá nhân và mục tiêu

**Cấu trúc layout:**

**a) Header**

- Title "Hồ sơ" (Headline.Small)

**b) User Info Card**

- Icon + Title "Thông tin cá nhân"

- TextInputLayout: Tên

- Row: Tuổi | Giới tính (dropdown)

- Row: Chiều cao (cm) | Cân nặng (kg)

**c) Goals Card**

- Icon + Title "Mục tiêu"

- Dropdown: Mức độ hoạt động

- Dropdown: Mục tiêu cân nặng

- TextInput: Mục tiêu calo hàng ngày

- Button "Tính tự động"

**d) BMI Card (Hero Style)**

- Background: primary\_container (#ECFDF5)

- Label "Chỉ số BMI"

- BMI value (48sp, bold)

- Chip hiển thị category (Bình thường, Thừa cân, ...)

**e) Action Buttons**

- Button "Lưu" (primary style)

- Divider

- Button "Đăng xuất" (outlined, error color)

**Đặc điểm thiết kế:**

- Cards với stroke và no elevation

- BMI card nổi bật với background color

- Dropdown menus sử dụng ExposedDropdownMenu style

## 4.5. Thiết kế Components

**4.5.1. MaterialCardView**

Ứng dụng sử dụng 3 styles cho cards:

**Widget.App.CardView (Default)**

- cornerRadius: 16dp

- elevation: 0dp

- backgroundColor: @color/surface

- strokeWidth: 1dp

- strokeColor: @color/outline

**Widget.App.CardView.Elevated**

- cornerRadius: 16dp

- elevation: 2dp

- backgroundColor: @color/surface

**Widget.App.CardView.Outlined**

- cornerRadius: 16dp

- elevation: 0dp

- strokeWidth: 1dp

- strokeColor: @color/outline

**4.5.2. Buttons**

**Widget.App.Button (Primary)**

- cornerRadius: 12dp

- backgroundColor: @color/primary (#10B981)

- textColor: white

- textAllCaps: false

- paddingVertical: 12dp

**Widget.App.Button.Outlined**

- cornerRadius: 12dp

- strokeColor: @color/primary

- textColor: @color/primary

- backgroundColor: transparent

**Widget.App.Button.Text**

- textColor: @color/primary

- backgroundColor: transparent

- rippleColor: @color/ripple\_primary

**4.5.3. TextInputLayout**

**Widget.App.TextInputLayout (Outlined)**

- boxCornerRadius: 12dp (all corners)

- boxStrokeColor: color state list

- hintTextColor: @color/primary

- boxBackgroundColor: @color/surface

**Widget.App.TextInputLayout.ExposedDropdownMenu**

- Kế thừa từ Outlined style

- Sử dụng cho dropdown selections

**4.5.4. Chips**

**Widget.App.Chip (Filter)**

- cornerRadius: 20dp (pill shape)

- minHeight: 36dp

- textSize: 14sp

- strokeWidth: 0dp

- backgroundColor: color state list (selected/unselected)

**Chip Colors theo context:**

- Meal type chips: Màu tương ứng (breakfast, lunch, dinner, snack)

- Workout category chips: Màu tương ứng (cardio, strength, flexibility)

- Period chips: Primary color khi selected

**4.5.5. Bottom Navigation**

**Widget.App.BottomNavigation**

- backgroundColor: @color/surface

- minHeight: 64dp

- elevation: 8dp

- labelVisibilityMode: labeled

- itemIconTint: color state list

- itemTextColor: color state list

- activeIndicatorStyle: primary\_container background

**Menu items:**

- Home (ic\_home)

- Diary (ic\_diary)

- Add (ic\_add)

- Profile (ic\_profile)

**4.5.6. TabLayout**

**Widget.App.TabLayout**

- indicatorColor: @color/primary

- indicatorHeight: 3dp

- indicatorFullWidth: false

- selectedTextColor: @color/primary

- textColor: @color/text\_secondary

- rippleColor: @color/ripple\_primary

**4.5.7. Progress Bars**

**OverflowProgressBar (Custom View)**

- Hiển thị progress với khả năng overflow (vượt 100%)

- normalColor: @color/progress\_normal (#10B981)

- overflowColor: @color/progress\_over (#EF4444)

- backgroundColor: @color/progress\_track (#F3F4F6)

- cornerRadius: full rounded

- Marker line tại vị trí 100%

**Macro Progress Bars**

- Height: 8dp (thick style)

- Màu riêng cho từng macro:

+ Protein: #EC4899 (pink)

+ Carbs: #F59E0B (orange)

+ Fat: #A855F7 (purple)

**4.5.8. RecyclerView Items**

**item\_food.xml - Hiển thị thực phẩm trong danh sách**

- Icon container với background màu consumed

- Food name (Title.Small)

- Food info: "350 cal / 100g" (Body.Small)

- Calorie badge với background

**item\_food\_entry.xml - Hiển thị entry trong nhật ký**

- Meal type indicator (icon với màu tương ứng)

- Food name

- Meal type chip + quantity

- Calorie badge với dấu "+"

- Time display

**item\_workout.xml - Hiển thị bài tập trong danh sách**

- Icon container với background màu burned

- Workout name

- Workout info: "10 cal/phút"

- Calorie badge

**item\_workout\_entry.xml - Hiển thị workout entry**

- Category indicator

- Workout name

- Duration/quantity

- Calorie badge với dấu "-"

- Time display

**4.5.9. Dialogs**

**Theme.App.Dialog**

- cornerRadius: 20dp

- backgroundColor: @color/surface

- Button styles: Text buttons với primary/secondary colors

**Dialog layouts trong ứng dụng:**

- dialog\_add\_food\_entry.xml: Thêm food entry với quantity và meal type

- dialog\_add\_workout\_entry.xml: Thêm workout entry với duration

- dialog\_create\_custom\_food.xml: Tạo thực phẩm tùy chỉnh

- dialog\_create\_custom\_workout.xml: Tạo bài tập tùy chỉnh

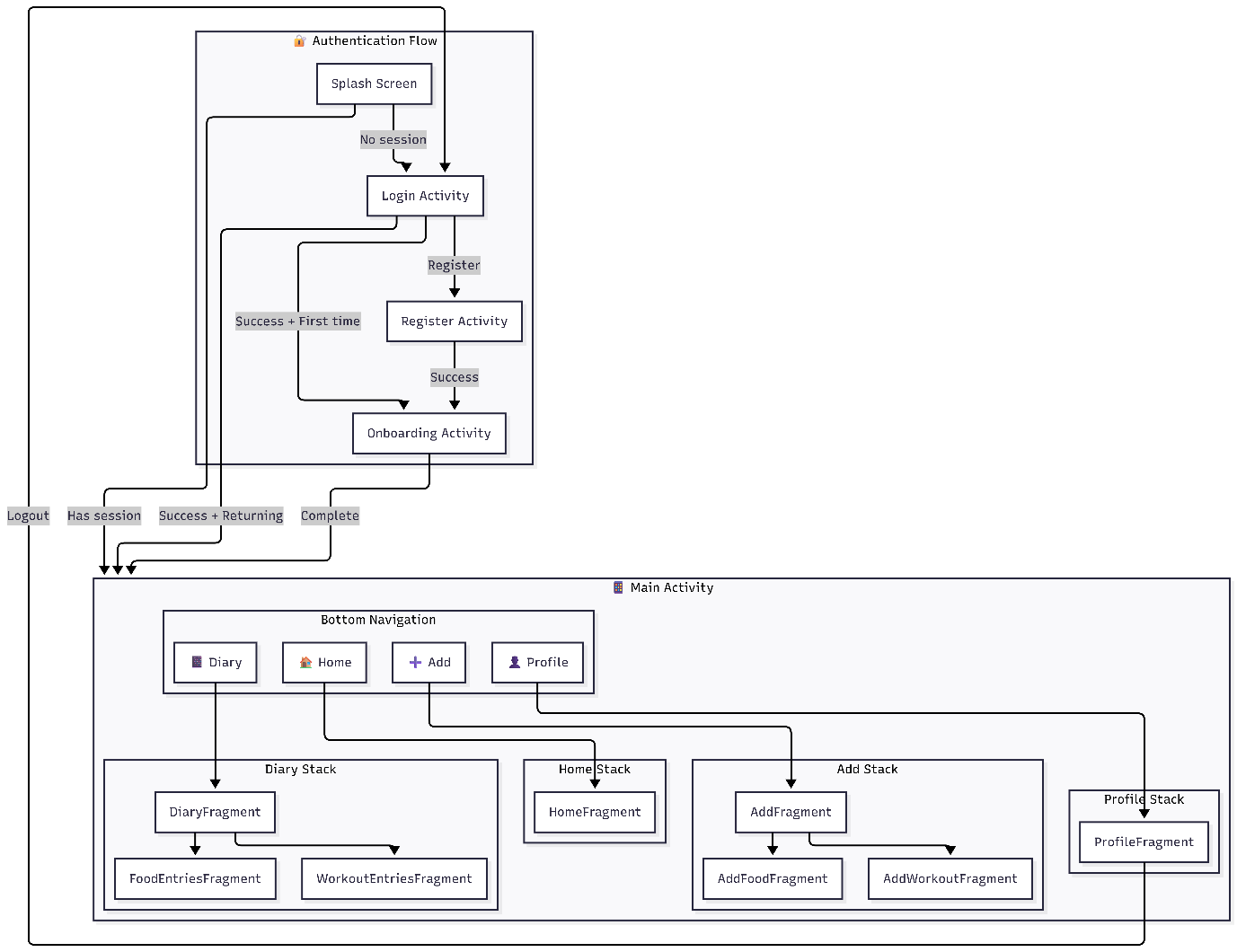
- dialog\_edit\_goals.xml: Chỉnh sửa mục tiêu

- dialog\_edit\_personal\_info.xml: Chỉnh sửa thông tin cá nhân

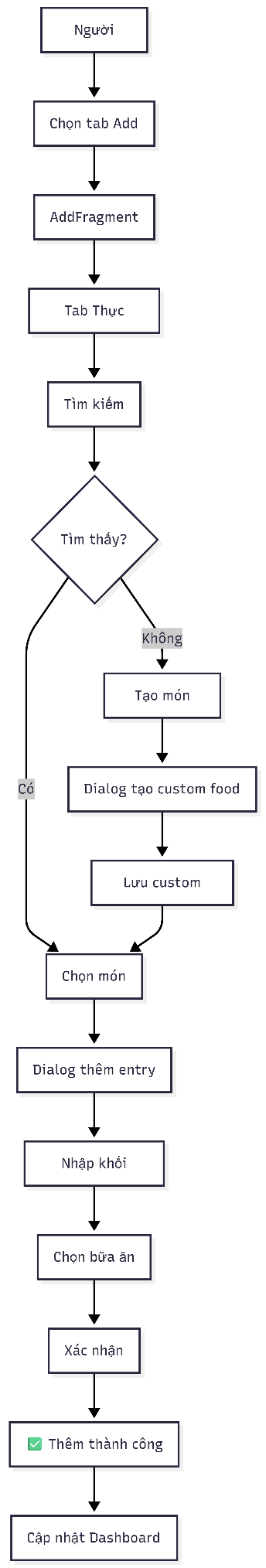
- dialog\_quick\_weight\_log.xml: Ghi nhận cân nặng nhanh

## 4.6. Navigation Flow

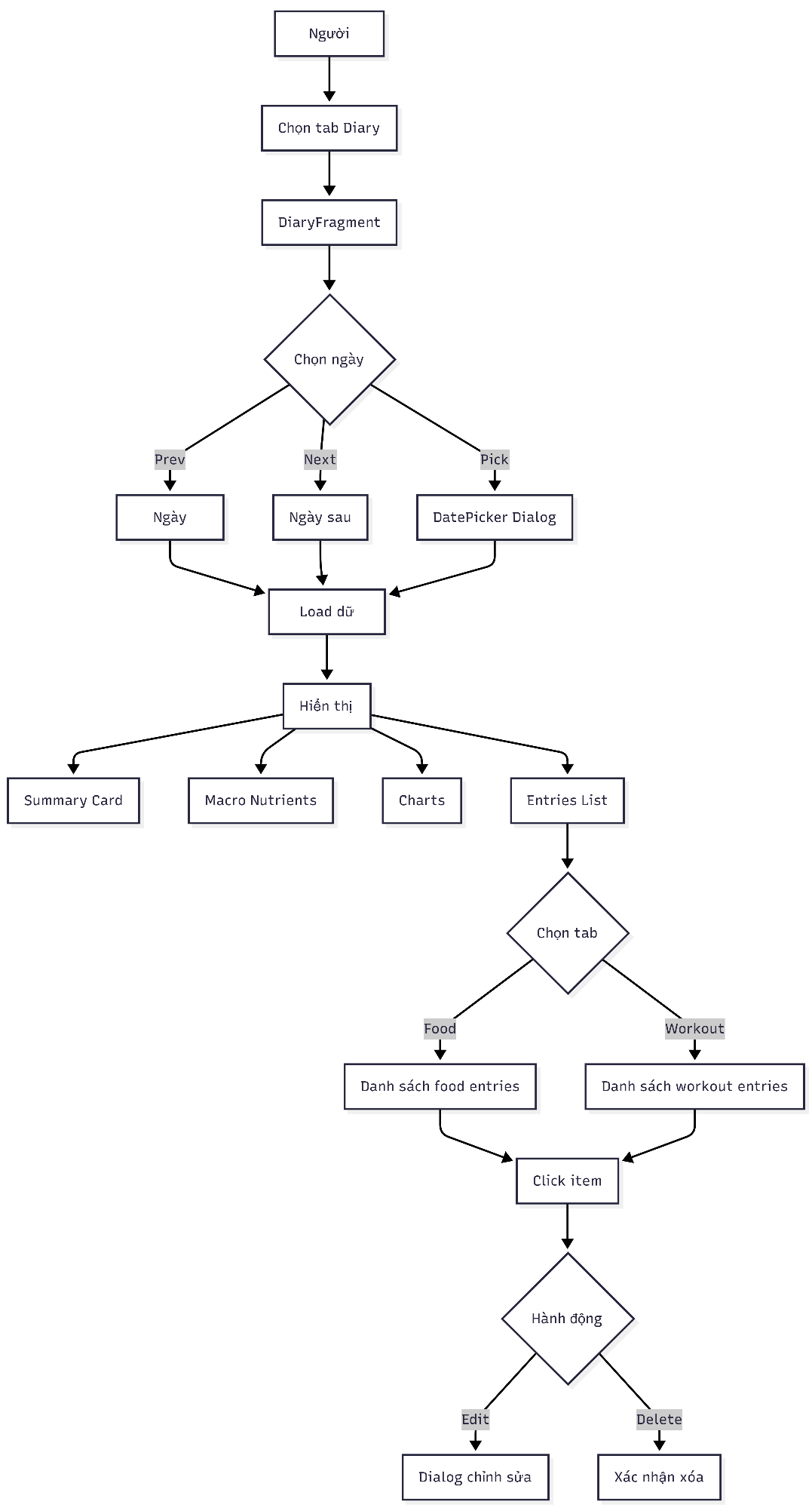
**4.6.1. Sơ đồ Navigation tổng quan**



**4.6.2. Sơ đồ User Flow - Thêm thực phẩm**



**4.6.3. Sơ đồ User Flow - Xem nhật ký**



## 4.7. Responsive Design

**4.7.1. Hỗ trợ nhiều kích thước màn hình**

**Ứng dụng được thiết kế responsive với các nguyên tắc:**

**a) Flexible Layouts**

- Sử dụng ConstraintLayout và LinearLayout với weights

- Tránh hard-coded dimensions

- Sử dụng match\_parent và wrap\_content phù hợp

**b) Dimension Resources**

- Định nghĩa dimensions trong dimens.xml

- Có thể override cho các screen sizes khác nhau

- Sử dụng sp cho text, dp cho spacing

**c) Scrollable Content**

- NestedScrollView cho các màn hình có nhiều nội dung

- RecyclerView với nestedScrollingEnabled=false khi cần

- fillViewport=true để đảm bảo content fill screen

**4.7.2. Minimum Touch Targets**

**Theo Material Design guidelines:**

- Minimum touch target: 48dp x 48dp

- Buttons: minHeight 48dp

- Icon buttons: 48dp x 48dp

- List items: minHeight 56dp

## 4.8. Dark Mode Support

**4.8.1. Cấu hình Theme**

Ứng dụng sử dụng Theme.Material3.DayNight.NoActionBar làm parent theme, cho phép tự động chuyển đổi giữa Light và Dark mode theo system settings.

**4.8.2. Color Resources cho Dark Mode**

**Các màu được định nghĩa trong:**

- values/colors.xml: Light mode colors

- values-night/colors.xml: Dark mode colors (nếu cần override)

**Nguyên tắc Dark Mode:**

- Background: Chuyển từ light (#FAFAFA) sang dark (#121212)

- Surface: Chuyển từ white sang dark gray

- Text: Đảo ngược contrast

- Primary colors: Giữ nguyên hoặc điều chỉnh brightness

**4.8.3. Status Bar và Navigation Bar**

**Cấu hình trong theme:**

- android:statusBarColor: Theo background color

- android:navigationBarColor: Theo surface color

- android:windowLightStatusBar: true (light mode) / false (dark mode)

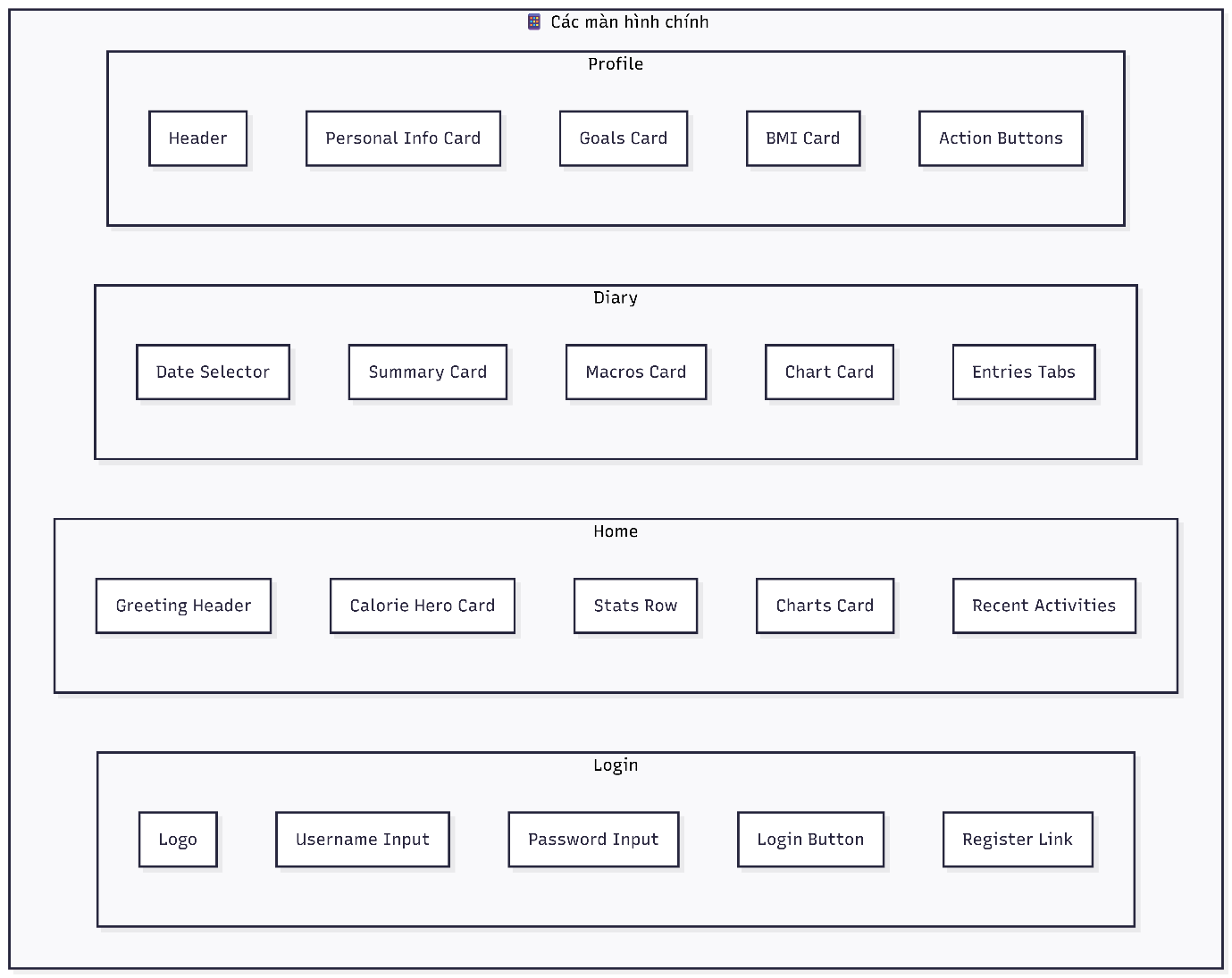
## 4.9. Screenshots và Wireframes

**4.9.1. Danh sách Screenshots**

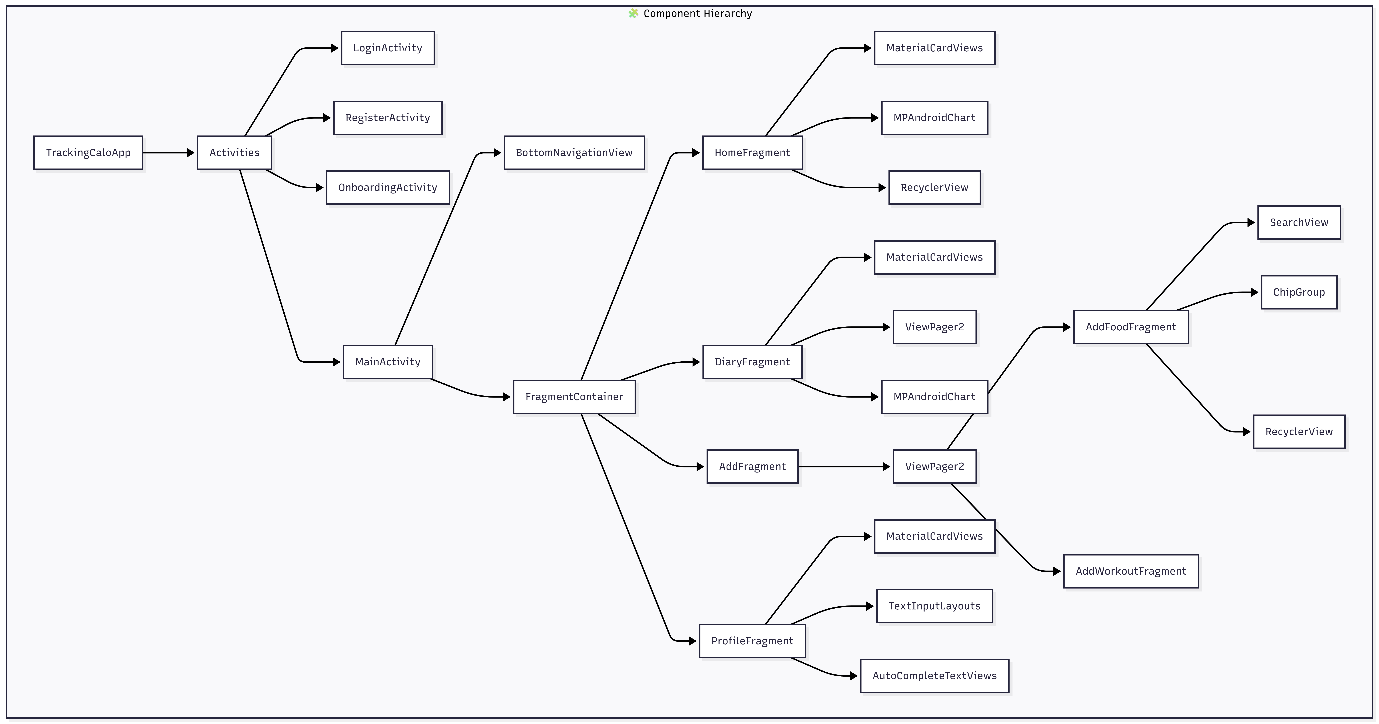
**Ứng dụng có các screenshots minh họa tại thư mục docs/screenshots/:**

|  |  |
| --- | --- |
| **Tên tập tin (File Name)** | **Mô tả chi tiết** |
| **home.png** | Màn hình chính (Home) hiển thị Dashboard tổng quan về Calo và tiến độ trong ngày. |
| **diary.png** | Màn hình Nhật ký (Diary) liệt kê chi tiết các bữa ăn và lịch sử tập luyện. |
| **add\_food.png** | Màn hình chức năng Thêm thực phẩm (Add Food) và tìm kiếm món ăn. |
| **onboarding.png** | Màn hình Onboarding (Chào mừng) giúp người dùng thiết lập chỉ số cơ thể ban đầu. |

**4.9.2. Wireframe tổng quan các màn hình**



**4.9.3. Component Hierarchy**



## 4.10. Tổng kết thiết kế UI/UX

**4.10.1. Điểm mạnh của thiết kế**

**a) Tuân thủ Material Design 3**

- Sử dụng đầy đủ các components chuẩn của Material 3

- Color system với semantic colors

- Typography scale nhất quán

- Shape system với rounded corners

**b) User Experience tốt**

- Navigation đơn giản với Bottom Navigation

- Visual feedback rõ ràng (colors, progress bars)

- Empty states được thiết kế

- Error handling với messages thân thiện

**c) Accessibility**

- Contrast ratio đạt chuẩn

- Touch targets đủ lớn

- Content descriptions cho icons

**d) Performance**

- Sử dụng RecyclerView cho lists

- Lazy loading với ViewPager2

- Efficient layouts (ConstraintLayout)

**4.10.2. Các cải tiến có thể thực hiện**

- Thêm animations và transitions

- Implement full Dark Mode support

- Thêm onboarding tutorial

- Hỗ trợ landscape orientation

- Thêm widgets cho home screen

# CHƯƠNG V: XÂY DỰNG VÀ CÀI ĐẶT CHƯƠNG TRÌNH

## 5.1. Môi trường phát triển

**5.1.1. Công cụ phát triển**

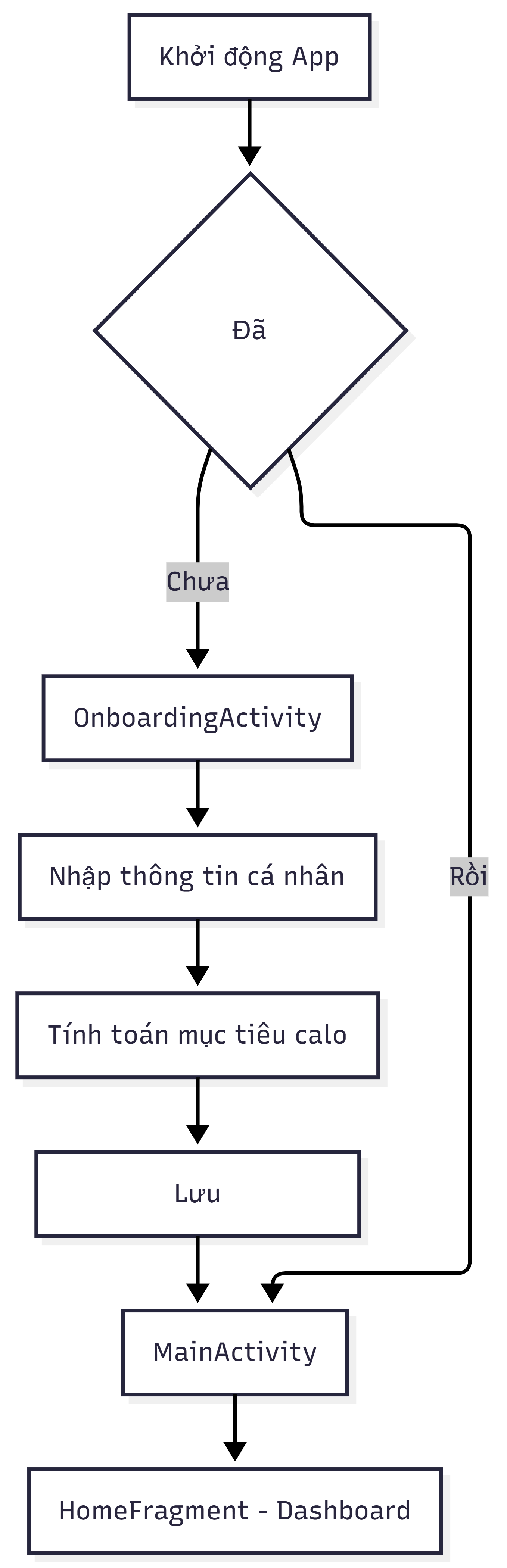
|  |  |  |
| --- | --- | --- |
| **Công cụ** | **Phiên bản** | **Mục đích** |
| Android Studio | Hedgehog 2023.1.1 | IDE chính cho phát triển Android |
| JDK | 11 | Java Development Kit |
| Android SDK | 36 (Android 14) | Target SDK |
| Android SDK Min | 27 (Android 8.1) | Minimum SDK hỗ trợ |
| Gradle | 8.12.0 | Build automation tool |
| Git | 2.x | Version control system |

**5.1.1. Thư viện và Dependencies**

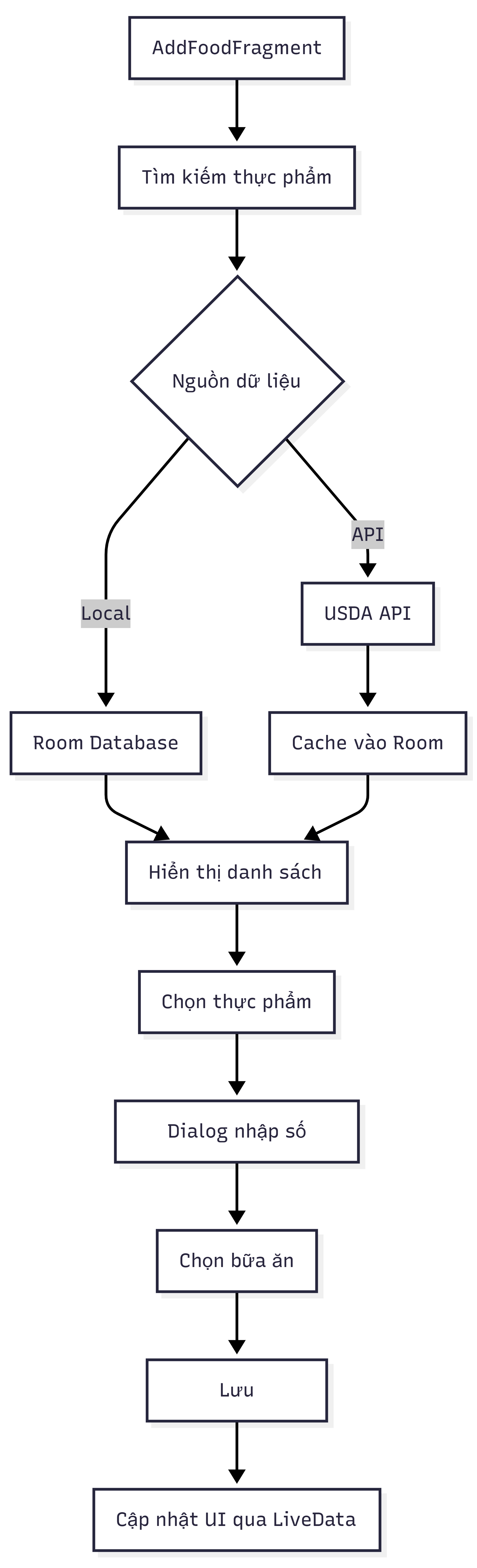
|  |  |  |
| --- | --- | --- |
| **Thư viện** | **Phiên bản** | **Mục đích** |
| AndroidX AppCompat | 1.7.1 | Backward compatibility |
| Material Design | 1.13.0 | UI components theo Material Design |
| Room Database | 2.6.1 | Local database với SQLite |
| Lifecycle ViewModel | 2.7.0 | Quản lý UI state |
| Lifecycle LiveData | 2.7.0 | Observable data holder |
| RecyclerView | 1.3.2 | Hiển thị danh sách hiệu quả |
| CardView | 1.0.0 | Card UI component |
| ViewPager2 | 1.1.0 | Swipeable views |
| MPAndroidChart | 3.1.0 | Biểu đồ thống kê |
| Volley | 1.2.1 | HTTP client cho API calls |

## 5.2. Luồng hoạt động chính

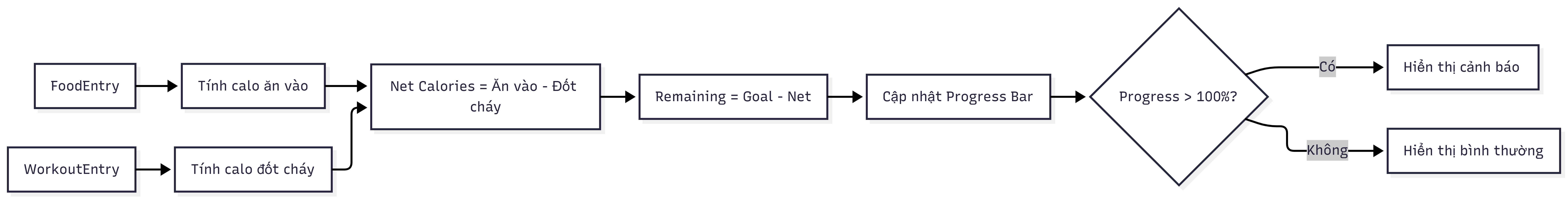
**5.2.1. Luồng khởi động ứng dụng**



**5.2.2. Luồng thêm thực phẩm**



**5.2.3. Luồng tính toán calo**



## 5.3. Cài đặt các thành phần chính

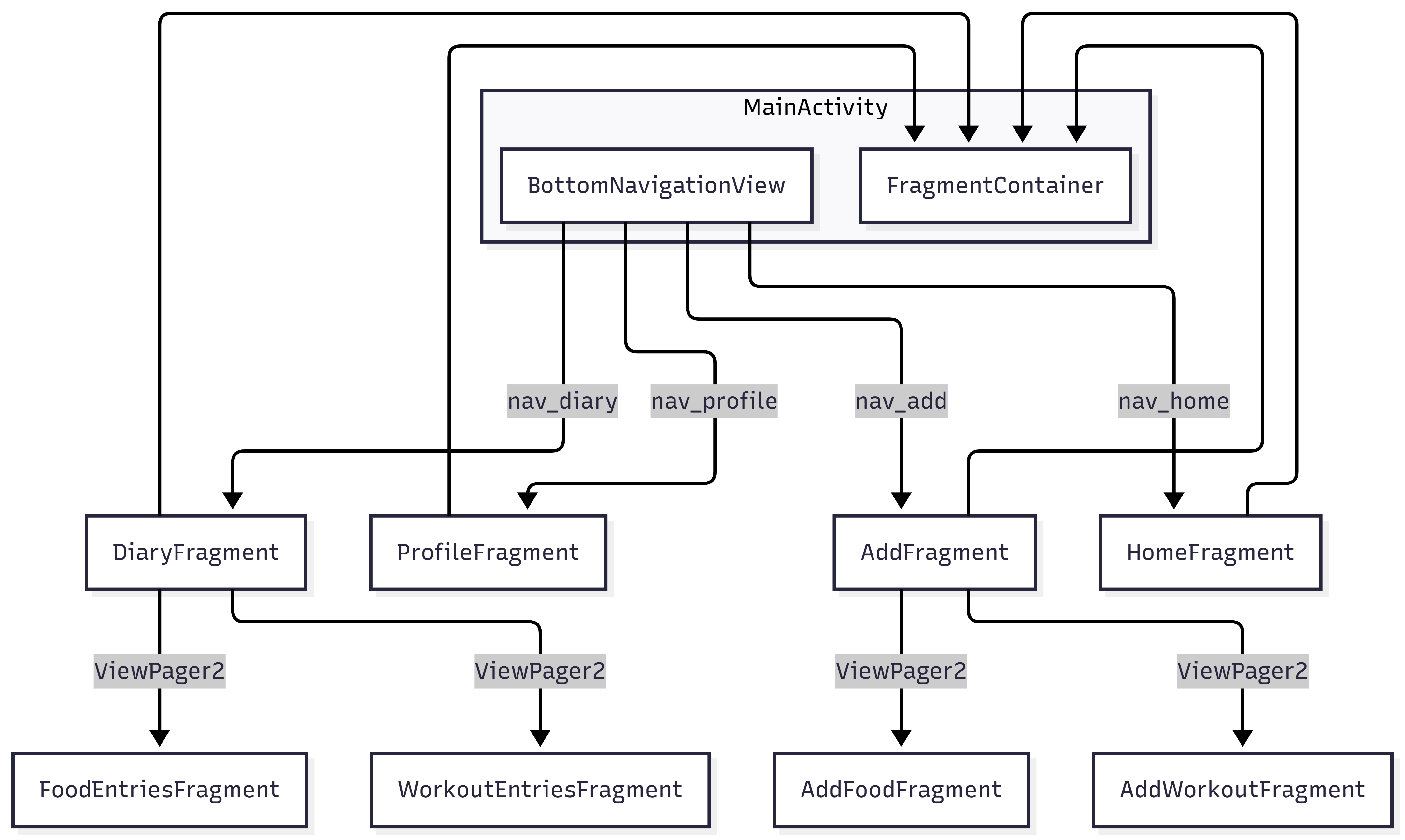
**5.3.1. Room Database - Singleton Pattern**

|  |
| --- |
| @Database(entities = {Food.class, FoodEntry.class, Workout.class,  WorkoutEntry.class, WeightLog.class, User.class},  version = 4)  public abstract class AppDatabase extends RoomDatabase {  private static volatile AppDatabase INSTANCE;    public static AppDatabase getDatabase(Context context) {  if (INSTANCE == null) {  synchronized (AppDatabase.class) {  if (INSTANCE == null) {  INSTANCE = Room.databaseBuilder(context, AppDatabase.class,  "tracking\_calo\_database")  .fallbackToDestructiveMigration()  .build();  }  }  }  return INSTANCE;  }    // DAOs  public abstract FoodDao foodDao();  public abstract FoodEntryDao foodEntryDao();  public abstract WorkoutDao workoutDao();  public abstract WorkoutEntryDao workoutEntryDao();  public abstract WeightLogDao weightLogDao();  public abstract UserDao userDao();  } |

**5.3.2. Repository Pattern - Hybrid Search**

|  |
| --- |
| public class FoodRepository {  private final FoodDao foodDao;  private UsdaApiService apiService;    // Tìm kiếm kết hợp Local + API  public void searchHybrid(String query, HybridSearchCallback callback) {  if (query.length() >= 3) {  apiService.searchFoods(query, 20, new SearchCallback() {  @Override  public void onSuccess(List<Food> apiFoods) {  // Cache vào Room  for (Food food : apiFoods) {  if (foodDao.getFoodByApiId(food.getApiId()) == null) {  foodDao.insert(food);  }  }  callback.onApiResults(apiFoods);  }  });  }  }  } |

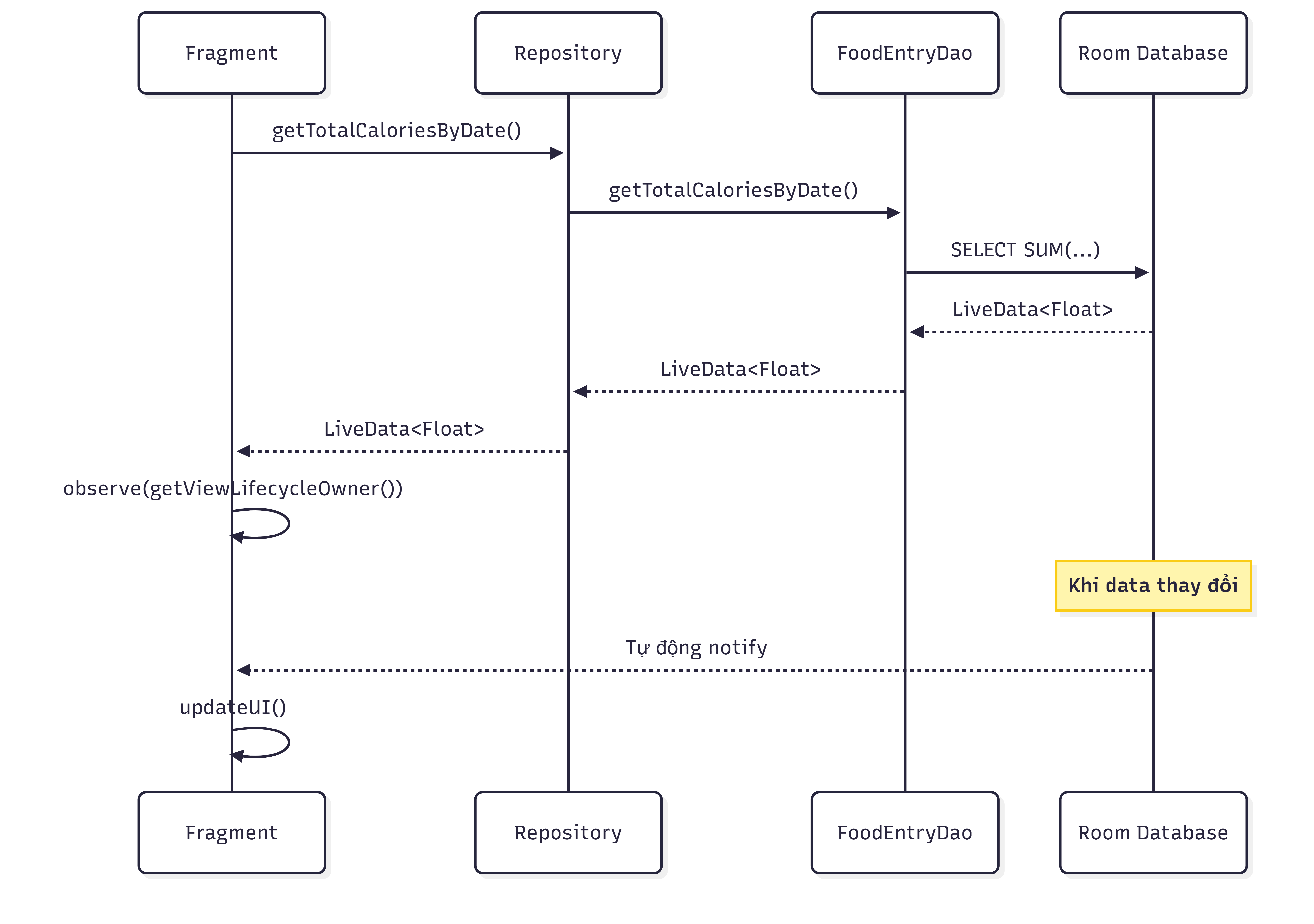
**5.3.3. Fragment Navigation với Bottom Navigation**



**Kỹ thuật giữ Fragment state:**

|  |
| --- |
| private void setupFragments() {  fm.beginTransaction()  .add(R.id.fragmentContainer, profileFragment, TAG\_PROFILE).hide(profileFragment)  .add(R.id.fragmentContainer, addFragment, TAG\_ADD).hide(addFragment)  .add(R.id.fragmentContainer, diaryFragment, TAG\_DIARY).hide(diaryFragment)  .add(R.id.fragmentContainer, homeFragment, TAG\_HOME)  .commit();  activeFragment = homeFragment;  }  private void switchFragment(Fragment fragment) {  getSupportFragmentManager().beginTransaction()  .hide(activeFragment)  .show(fragment)  .commit();  activeFragment = fragment;  } |

**5.3.4. LiveData - Reactive UI Pattern**



Sử dụng trong Fragment:

|  |
| --- |
| foodEntryRepository.getTotalCaloriesByDate(startOfDay, endOfDay)  .observe(getViewLifecycleOwner(), consumed -> {  tvCaloriesConsumed.setText(String.valueOf((int) consumed));  updateNetCalories();  }); |

**5.3.5. Tính toán calo - CalorieCalculator**

Công thức Mifflin-St Jeor:

|  |
| --- |
| public static float calculateBMR(float weight, float height, int age, boolean isMale) {  float bmr = (10 \* weight) + (6.25f \* height) - (5 \* age);  return isMale ? bmr + 5 : bmr - 161;  } |

TDEE = BMR × Activity Multiplier:

|  |
| --- |
| public static float calculateTDEE(float bmr, float activityMultiplier) {  return bmr \* activityMultiplier;  } |

Mục tiêu calo theo weight goal:

|  |
| --- |
| public static int calculateDailyCalorieGoal(float tdee, String weightGoal) {  switch (weightGoal) {  case "lose": return Math.round(tdee - 500); // -0.5kg/tuần  case "gain": return Math.round(tdee + 500); // +0.5kg/tuần  default: return Math.round(tdee); // Duy trì  }  } |

BMI theo tiêu chuẩn châu Á:

|  |
| --- |
| public static String getBMICategory(float bmi) {  if (bmi < 18.5f) return "Thiếu cân";  if (bmi < 23f) return "Bình thường";  if (bmi < 25f) return "Thừa cân";  return "Béo phì";  } |

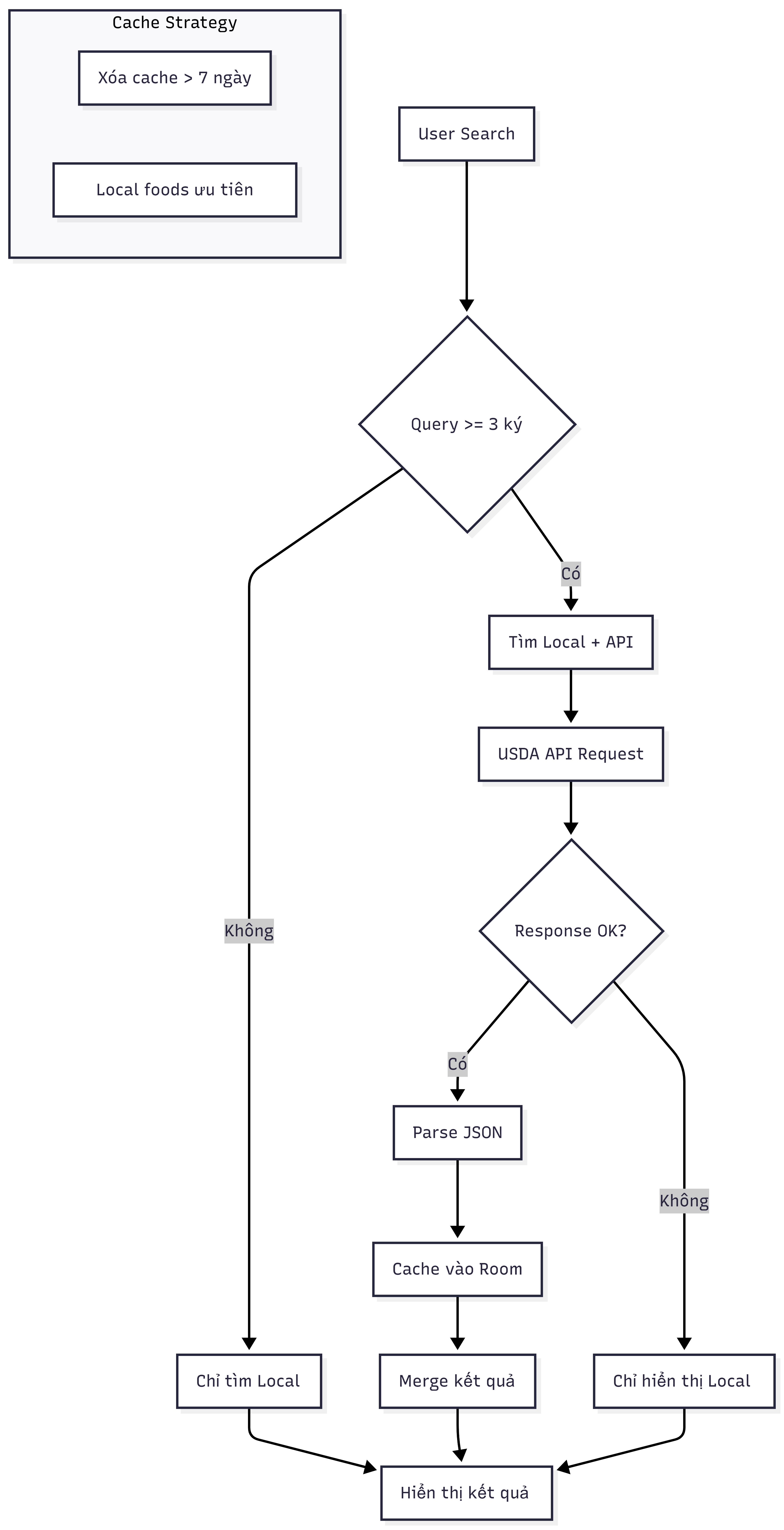
**5.3.6. Biểu đồ thống kê - ChartHelper**



**Setup và update chart:**

|  |
| --- |
| public static void setupLineChart(LineChart chart, Context ctx) {  chart.getDescription().setEnabled(false);  chart.setTouchEnabled(true);  chart.setDragEnabled(true);  chart.getXAxis().setPosition(XAxis.XAxisPosition.BOTTOM);  chart.getAxisRight().setEnabled(false);  }  public static void updateLineChartData(LineChart chart, List<DailyCalorieSum> data) {  List<Entry> entries = new ArrayList<>();  for (int i = 0; i < data.size(); i++) {  entries.add(new Entry(i, data.get(i).getTotalCalories()));  }    LineDataSet dataSet = new LineDataSet(entries, "Calories");  dataSet.setMode(LineDataSet.Mode.CUBIC\_BEZIER);  dataSet.setDrawFilled(true);    chart.setData(new LineData(dataSet));  chart.animateX(500);  } |

## 5.4. Tích hợp API và Caching



**USDA API Service:**

|  |
| --- |
| public void searchFoods(String query, int pageSize, SearchCallback callback) {  String url = BASE\_URL + "foods/search?api\_key=" + API\_KEY  + "&query=" + Uri.encode(query);    JsonObjectRequest request = new JsonObjectRequest(url, null,  response -> callback.onSuccess(UsdaParser.parseFoods(response)),  error -> callback.onError(error.getMessage())  );  requestQueue.add(request);  } |

## 5.5. Hướng dẫn cài đặt

**5.5.1. Yêu cầu hệ thống**

* Android 8.1 (API 27) trở lên
* RAM: 2GB+
* Bộ nhớ: 50MB

**5.5.2. Các bước cài đặt**

**1. Clone repository**

git clone https://github.com/username/TrackingCaloApp.git

**2. Cấu hình API (tùy chọn)**

Tạo file local.properties, thêm: USDA\_API\_KEY=your\_api\_key

**3. Build APK**

./gradlew assembleRelease

Output: app/build/outputs/apk/release/app-release.apk

# CHƯƠNG VI. TỔNG KẾT VÀ HƯỚNG PHÁT TRIỂN

## 6.1. Kết quả đạt được

Sau quá trình nghiên cứu, phân tích và xây dựng, đề tài đã hoàn thành ứng dụng TrackingCaloApp - một ứng dụng Android hỗ trợ người dùng theo dõi lượng calo nạp vào và tiêu hao hàng ngày. Ứng dụng đã đáp ứng được các mục tiêu đề ra ban đầu và mang lại những kết quả cụ thể như sau.

Về mặt chức năng, ứng dụng cho phép người dùng ghi nhận thực phẩm đã ăn với đầy đủ thông tin dinh dưỡng bao gồm calo, protein, carbohydrate và chất béo. Người dùng có thể phân loại thực phẩm theo bữa ăn (sáng, trưa, tối, phụ) và theo dõi lượng calo tiêu hao thông qua các bài tập thể dục. Hệ thống tự động tính toán mục tiêu calo cá nhân hóa dựa trên công thức Mifflin-St Jeor, một trong những công thức được đánh giá chính xác nhất hiện nay trong lĩnh vực dinh dưỡng.

Về mặt kỹ thuật, ứng dụng được xây dựng theo kiến trúc MVVM kết hợp Repository Pattern, đảm bảo code clean, dễ bảo trì và mở rộng. Việc sử dụng Room Database với LiveData giúp UI tự động cập nhật khi dữ liệu thay đổi, mang lại trải nghiệm mượt mà cho người dùng. Tính năng Hybrid Search kết hợp cơ sở dữ liệu local với USDA FoodData Central API giúp mở rộng đáng kể nguồn dữ liệu thực phẩm, cho phép người dùng tìm kiếm hàng nghìn loại thực phẩm khác nhau.

Về mặt giao diện, ứng dụng tuân thủ Material Design 3 với thiết kế hiện đại, trực quan và thân thiện với người dùng. Các biểu đồ thống kê (Line Chart, Bar Chart, Pie Chart) giúp người dùng dễ dàng theo dõi xu hướng dinh dưỡng theo thời gian. Tính năng OverflowProgressBar cho phép hiển thị trực quan khi người dùng vượt quá mục tiêu calo, kèm theo cảnh báo phù hợp.

## 6.2. Ưu điểm và hạn chế

**6.2.1. Ưu điểm**

Ứng dụng TrackingCaloApp sở hữu nhiều ưu điểm nổi bật. Thứ nhất, việc tính toán mục tiêu calo được cá nhân hóa hoàn toàn dựa trên thông tin cơ thể, mức độ hoạt động và mục tiêu cân nặng của từng người dùng, thay vì áp dụng một con số cố định cho tất cả. Điều này giúp người dùng có được mục tiêu phù hợp và thực tế hơn.

Thứ hai, giao diện người dùng được thiết kế đơn giản, dễ sử dụng với luồng thao tác logic. Người dùng có thể nhanh chóng ghi nhận bữa ăn chỉ với vài thao tác chạm. Tính năng tìm kiếm thực phẩm hoạt động nhanh và chính xác, hỗ trợ cả tiếng Việt lẫn tiếng Anh.

Thứ ba, ứng dụng hoạt động hoàn toàn offline với cơ sở dữ liệu local, chỉ cần kết nối internet khi muốn tìm kiếm thực phẩm từ API bên ngoài. Dữ liệu từ API được cache lại để sử dụng offline trong những lần sau.

Thứ tư, hệ thống biểu đồ thống kê đa dạng giúp người dùng có cái nhìn tổng quan về thói quen ăn uống, từ đó điều chỉnh chế độ dinh dưỡng phù hợp hơn.

**6.2.2. Hạn chế**

Bên cạnh những ưu điểm, ứng dụng vẫn còn một số hạn chế cần được cải thiện. Đầu tiên, cơ sở dữ liệu thực phẩm Việt Nam có sẵn trong ứng dụng còn hạn chế, chủ yếu tập trung vào các món ăn phổ biến. Người dùng cần tự thêm các món ăn đặc trưng vùng miền hoặc món ăn gia đình.

Thứ hai, ứng dụng chưa có tính năng đồng bộ dữ liệu lên cloud, nghĩa là khi người dùng đổi điện thoại hoặc cài đặt lại ứng dụng, dữ liệu sẽ bị mất. Đây là một hạn chế đáng kể cần được khắc phục trong các phiên bản tiếp theo.

Thứ ba, tính năng nhận diện thực phẩm qua hình ảnh chưa được tích hợp, người dùng vẫn phải nhập liệu thủ công. Điều này có thể gây bất tiện khi người dùng ăn những món ăn phức tạp với nhiều thành phần.

Cuối cùng, ứng dụng chưa có tính năng xã hội như chia sẻ tiến độ, thách thức bạn bè hay tham gia cộng đồng, những yếu tố có thể giúp tăng động lực cho người dùng.

## 6.3. Hướng phát triển

Dựa trên những hạn chế đã nhận diện và xu hướng phát triển của các ứng dụng sức khỏe hiện đại, nhóm đề xuất các hướng phát triển sau cho ứng dụng TrackingCaloApp.

Về tính năng đồng bộ và backup, việc tích hợp Firebase Authentication và Cloud Firestore sẽ cho phép người dùng đăng nhập bằng tài khoản Google và đồng bộ dữ liệu lên cloud. Điều này không chỉ giúp bảo vệ dữ liệu mà còn cho phép người dùng sử dụng ứng dụng trên nhiều thiết bị khác nhau.

Về nhận diện thực phẩm bằng AI, việc tích hợp các mô hình machine learning như TensorFlow Lite hoặc sử dụng các API nhận diện hình ảnh sẽ giúp người dùng chỉ cần chụp ảnh món ăn, ứng dụng sẽ tự động nhận diện và ước tính lượng calo. Đây là tính năng được nhiều ứng dụng hàng đầu như MyFitnessPal, Lose It! đã triển khai thành công.

Về mở rộng cơ sở dữ liệu, việc hợp tác với các chuyên gia dinh dưỡng để xây dựng cơ sở dữ liệu thực phẩm Việt Nam đầy đủ hơn là rất cần thiết. Ngoài ra, có thể tích hợp thêm các API khác như Nutritionix, Edamam để đa dạng hóa nguồn dữ liệu.

Về tính năng xã hội, việc xây dựng cộng đồng người dùng với các tính năng như chia sẻ công thức nấu ăn healthy, thách thức giảm cân theo nhóm, bảng xếp hạng sẽ giúp tăng tương tác và động lực cho người dùng.

Về tích hợp thiết bị đeo, việc kết nối với các thiết bị như smartwatch, fitness tracker thông qua Google Fit API hoặc Samsung Health SDK sẽ giúp tự động ghi nhận lượng calo tiêu hao từ các hoạt động hàng ngày, mang lại độ chính xác cao hơn.

Về cá nhân hóa nâng cao, việc áp dụng machine learning để phân tích thói quen ăn uống và đưa ra gợi ý thực phẩm phù hợp, cảnh báo khi phát hiện xu hướng ăn uống không lành mạnh sẽ nâng cao giá trị của ứng dụng.

## 6.4. Kết luận

Đề tài "Xây dựng ứng dụng theo dõi calo trên nền tảng Android" đã hoàn thành các mục tiêu đề ra, tạo ra một sản phẩm phần mềm có tính ứng dụng thực tiễn cao trong việc hỗ trợ người dùng quản lý chế độ dinh dưỡng hàng ngày.

Trong quá trình thực hiện đề tài, nhóm đã có cơ hội áp dụng và củng cố kiến thức về phát triển ứng dụng Android với ngôn ngữ Java, làm quen với các công nghệ hiện đại như Room Database, LiveData, ViewModel theo kiến trúc MVVM. Bên cạnh đó, nhóm cũng học được cách tích hợp API bên ngoài, xử lý dữ liệu bất đồng bộ và thiết kế giao diện người dùng theo chuẩn Material Design.

Mặc dù còn một số hạn chế, ứng dụng TrackingCaloApp đã đặt nền móng vững chắc cho việc phát triển thêm các tính năng nâng cao trong tương lai. Với xu hướng ngày càng quan tâm đến sức khỏe và dinh dưỡng của người dân, ứng dụng có tiềm năng phát triển thành một công cụ hữu ích, đồng hành cùng người dùng trong hành trình xây dựng lối sống lành mạnh.

# TÀI LIỆU THAM KHẢO

**A. Tài liệu tiếng Việt**

[1] Nguyễn Văn Hiệp (2022), Lập trình Android cơ bản đến nâng cao, NXB Khoa học và Kỹ thuật, Hà Nội.

[2] Trần Đình Quế (2021), Phát triển ứng dụng di động với Android Studio, NXB Đại học Quốc gia TP.HCM.

[3] Viện Dinh dưỡng Quốc gia (2017), Bảng thành phần thực phẩm Việt Nam, NXB Y học, Hà Nội.

[4] Bộ Y tế Việt Nam (2016), Hướng dẫn chế độ ăn và hoạt động thể lực cho người trưởng thành Việt Nam, Quyết định số 2892/QĐ-BYT.

**B. Tài liệu tiếng Anh**

[5] Mifflin, M.D., St Jeor, S.T., et al. (1990), "A new predictive equation for resting energy expenditure in healthy individuals", The American Journal of Clinical Nutrition, 51(2), pp. 241-247.

[6] Google Developers (2024), Android Developers Documentation, https://developer.android.com/docs (truy cập ngày 15/12/2024).

[7] Google Developers (2024), Room Persistence Library, https://developer.android.com/training/data-storage/room (truy cập ngày 15/12/2024).

[8] Google Developers (2024), Guide to App Architecture, https://developer.android.com/topic/architecture (truy cập ngày 15/12/2024).

[9] Material Design (2024), Material Design 3 Guidelines, https://m3.material.io/ (truy cập ngày 15/12/2024).

[10] U.S. Department of Agriculture (2024), FoodData Central API Documentation, https://fdc.nal.usda.gov/api-guide.html (truy cập ngày 15/12/2024).

[11] PhilJay (2024), MPAndroidChart Documentation, https://github.com/PhilJay/MPAndroidChart (truy cập ngày 15/12/2024).

[12] Android Developers (2024), Volley Overview, https://developer.android.com/training/volley (truy cập ngày 15/12/2024).

[13] JetBrains (2024), Kotlin for Android Developers, https://kotlinlang.org/docs/android-overview.html (truy cập ngày 15/12/2024).