Ngô Ngọc Gia Bảo TÊN ĐỀ TÀI: XÂY DỰNG ỨNG DỤNG ĐO LƯỜNG VÀ ĐÁNH GIÁ THỰC PHẨM THÔNG MINH 2026

ĐẠI HỌC ĐÀ NẴNG

TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN TỐT NGHIỆP

NGÀNH: CÔNG NGHỆ THÔNG TIN

CHUYÊN NGÀNH: CÔNG NGHỆ PHẦN MỀM

ĐỀ TÀI:

XÂY DỰNG ỨNG DỤNG ĐO LƯỜNG VÀ ĐÁNH GIÁ THỰC PHẨM THÔNG MINH

Người hướng dẫn: **TS. NGUYỄN VĂN HIỆU**

Sinh viên thực hiện: **NGÔ NGỌC GIA BẢO**

Số thẻ sinh viên: **102190003**

Lớp**:** **19TCLC-DT1**

Đà Nẵng, 01/2026

TÓM TẮT

Tên đề tài: Xây dựng ứng dụng đo lường và đánh giá thực phẩm thông minh

Sinh viên thực hiện: Ngô Ngọc Gia Bảo

Số thẻ SV: 102190003 Lớp: 19TCLC_DT1

Đồ án này tập trung vào việc xây dựng ứng dụng mobile trên nền tảng Flutter để đo lường và đánh giá chất lượng thực phẩm thông minh. Ứng dụng kết nối với thiết bị đo quang phổ Ocean FX thông qua Bluetooth Low Energy (BLE) để thu thập dữ liệu quang phổ từ mẫu thực phẩm. Dữ liệu sau đó được gửi lên hệ thống backend FastAPI để phân tích bằng các mô hình AI/ML, bao gồm: (1) Xác định loại thực phẩm, (2) Xác định nồng độ các chất hóa học, và (3) Đánh giá mức độ an toàn dựa trên ngưỡng cho phép. Kết quả phân tích được hiển thị trực quan trên ứng dụng với các biểu đồ, bảng thống kê và lịch sử kiểm tra.

Ứng dụng được phát triển theo kiến trúc Clean Architecture với pattern BLoC để quản lý state, đảm bảo code dễ bảo trì và mở rộng. Hệ thống hỗ trợ đa ngôn ngữ (tiếng Việt, tiếng Anh), xác thực người dùng qua JWT token

ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA CÔNG NGHỆ THÔNG TIN

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

Họ tên sinh viên: Ngô Ngọc Gia Bảo Số thẻ sinh viên:102190003

Lớp: 19TCLC_DT1 Khoa:CNTT Ngành: Công Nghệ Thông Tin

1. *Tên đề tài đồ án: Xây dựng ứng dụng đo lường và đánh giá thực phẩm thông minh*

Xây dựng ứng dụng đo lường và đánh giá thực phẩm thông minh

2. *Đề tài thuộc diện: ☐ Có ký kết thỏa thuận sở hữu trí tuệ đối với kết quả thực hiện*

3. *Các số liệu và dữ liệu ban đầu:*

- *Thiết bị đo quang phổ Ocean FX (NIR spectrometer)*
- *Raspberry Pi với hệ điều hành Linux*
- *Thư viện python-seabreeze và SeaBreeze C driver*
- *Dữ liệu mẫu quang phổ từ các loại thực phẩm khác nhau*
- *API backend FastAPI với các mô hình AI/ML đã được huấn luyện*
- *WebSocket server để truyền dữ liệu real-time*
- *Tài liệu kỹ thuật về thiết bị Ocean FX và thư viện SeaBreeze*

4. *Nội dung các phần thuyết minh và tính toán:*

1. *Phân tích yêu cầu và thiết kế hệ thống*
2. *Thiết kế kiến trúc ứng dụng Flutter theo Clean Architecture*
3. *Triển khai các module chính: phân tích thực phẩm, dashboard, quản lý người dùng*
4. *Tích hợp với API backend và xử lý dữ liệu quang phổ*
5. *Kiểm thử và đánh giá ứng dụng*

5. *Các bản vẽ, đồ thị (ghi rõ các loại và kích thước bản vẽ):*

- *Sơ đồ kiến trúc hệ thống*
- *Sơ đồ luồng dữ liệu từ thiết bị đến ứng dụng*
- *Sơ đồ các màn hình chính của ứng dụng*
- *Biểu đồ kết quả phân tích thực phẩm*
- *Sơ đồ cấu trúc thư mục dự án*

6. *Họ tên người hướng dẫn:* Nguyễn Văn Hiệu

7. *Ngày giao nhiệm vụ đồ án: ……/……/201…..*

8. Ngày hoàn thành đồ án: ……/……/201…..

Đà Nẵng, ngày tháng năm 201

Trưởng Bộ môn …….. **Người hướng dẫn**

LỜI NÓI ĐẦU

Trong bối cảnh vấn đề an toàn thực phẩm đang trở thành mối quan tâm lớn của xã hội, việc phát triển các công cụ hỗ trợ kiểm tra và đánh giá chất lượng thực phẩm một cách nhanh chóng và chính xác là rất cần thiết. Công nghệ quang phổ hồng ngoại gần (NIR) đã được ứng dụng rộng rãi trong việc phân tích thành phần và chất lượng thực phẩm nhờ khả năng đo lường không phá hủy mẫu và cho kết quả nhanh chóng.

Đồ án này tập trung vào việc xây dựng ứng dụng mobile trên nền tảng Flutter để nhận dữ liệu quang phổ từ thiết bị đo quang phổ Ocean FX thông qua hệ thống Raspberry Pi và WebSocket server, sau đó sử dụng các mô hình AI/ML để phân tích và đánh giá chất lượng thực phẩm. Ứng dụng được thiết kế với giao diện thân thiện, dễ sử dụng, hỗ trợ người dùng trong việc kiểm tra nhanh chất lượng thực phẩm tại chỗ với khả năng nhận dữ liệu real-time.

Đồ án được thực hiện dưới sự hướng dẫn của thầy Nguyễn Văn Hiệu và sự hỗ trợ của team NIR. Em xin chân thành cảm ơn các thầy cô, bạn bè đã hỗ trợ và đóng góp ý kiến trong quá trình thực hiện đồ án.

CAM ĐOAN

Tôi xin cam đoan rằng đồ án tốt nghiệp này là công trình nghiên cứu của bản thân tôi dưới sự hướng dẫn của giảng viên hướng dẫn. Các số liệu, kết quả trình bày trong đồ án là trung thực và chưa được công bố trong bất kỳ công trình nào khác. Các tham khảo trong đồ án đều được trích dẫn và ghi rõ nguồn gốc.

Sinh viên thực hiện

{Chữ ký, họ và tên sinh viên}

MỤC LỤC

TÓM TẮT
NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP
LỜI NÓI ĐẦU
CAM ĐOAN
MỤC LỤC
DANH SÁCH CÁC BẢNG, HÌNH VẼ
DANH SÁCH CÁC KÝ HIỆU, CHỮ VIẾT TẮT
MỞ ĐẦU 1
1. Mục đích thực hiện đề tài 1
2. Mục tiêu đề tài 1
3. Phạm vi và đối tượng nghiên cứu 2
4. Phương pháp nghiên cứu 2
5. Cấu trúc của đồ án tốt nghiệp 3
Chương 1: TỔNG QUAN VỀ CÔNG NGHỆ VÀ THIẾT BỊ 4
1.1. Tổng quan về công nghệ quang phổ hồng ngoại gần (NIR) 4
1.1.1. Nguyên lý hoạt động của công nghệ NIR 4
1.1.2. Dữ liệu quang phổ và định dạng file 4
1.2. Thiết bị đo quang phổ Ocean FX và hệ thống Raspberry Pi 5
1.2.1. Giới thiệu về thiết bị Ocean FX 5
1.2.2. Thông số kỹ thuật 6
1.2.2. Hệ thống Raspberry Pi và SeaBreeze 6
1.3. Hệ thống Raspberry Pi và SeaBreeze 8
1.3.1. Raspberry Pi 8
1.3.2. Thư viện SeaBreeze 8
1.3.3. Thư viện python-seabreeze 9
1.4. Công nghệ WebSocket 10
1.4.1. Tổng quan về WebSocket 10
1.4.2. WebSocket trong Flutter 11
1.5. Framework Flutter và ngôn ngữ Dart 11
1.5.1. Giới thiệu về Flutter 11
1.5.2. Ngôn ngữ Dart 12
1.6. Kiến trúc Clean Architecture và BLoC Pattern 13
1.5.1. Clean Architecture 13
1.5.2. BLoC Pattern 14
Chương 2: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG 15
2.1. Phân tích yêu cầu hệ thống 16
2.1.1. Yêu cầu chức năng 16
2.1.2. Sơ đồ use-case 17
2.1.3. Biểu đồ Sequence 30
2.1.4. Quy trình phân tích thực phẩm bằng AI 36

2.1.2. Yêu cầu phi chức năng 39
2.2. Thiết kế kiến trúc ứng dụng 40
2.2.1. Kiến trúc tổng thể 40
2.2.2. Cấu trúc thư mục dự án 41
Bảng 2.7: Cấu trúc thư mục dự án 42
2.2.3. Luồng dữ liệu từ thiết bị đến ứng dụng 43
2.3. Thiết kế các module chính 45
2.3.1. Module phân tích thực phẩm 45
2.3.2. Module Dashboard 47
2.4. Thiết kế giao diện người dùng 50
2.4.1. Các màn hình chính 50
2.4.2. Design System 52
2.5. Thiết kế tích hợp với API backend 53
2.5.1. API Endpoints 53
2.5.2. Request/Response Format 54
2.5.3. Error Handling 55
Chương 3: TRIỂN KHAI VÀ KẾT QUẢ 56
3.1. Môi trường phát triển và công cụ 57
3.2. Triển khai module phân tích thực phẩm 58
3.2.1. Parse file dữ liệu quang phổ 58
3.2.2. Tích hợp với API phân tích 58
3.2.3. Xử lý và hiển thị kết quả 59
3.3. Triển khai module Dashboard 59
3.3.1. Dashboard Screen 59
3.3.2. Tích hợp API Dashboard 60
3.4. Triển khai các tính năng phụ trợ 60
3.4.1. Đa ngôn ngữ (i18n) 60
3.4.2. Quản lý routing 61
3.4.3. Dependency Injection 61
3.4.4. Xác thực và bảo mật 62
3.5. Triển khai module quản lý người dùng (Admin) 62
3.5.1. Danh sách người dùng 62
3.5.2. Chi tiết người dùng 62
3.6. Triển khai module quản lý mô hình AI (Admin) 63
3.7. Triển khai module cấu hình hệ thống (Admin) 63
3.8. Kết quả và đánh giá 63
3.8.1. Kết quả đạt được 63
3.8.2. Đánh giá hiệu năng 64
3.8.3. Hạn chế và hướng phát triển 77
KẾT LUẬN 78
TÀI LIỆU THAM KHẢO 80

PHỤ LỤC 81
Phụ lục A: Cấu trúc thư mục dự án 81
Phụ lục B: Một số đoạn code quan trọng 81
Phụ lục C: Hình ảnh giao diện ứng dụng 81

DANH SÁCH CÁC BẢNG, HÌNH VẼ

BẢNG

Bảng 1.1: Thông số kỹ thuật thiết bị Ocean FX

Bảng 2.1: Use Case mô tả đăng nhập

Bảng 2.2: Use Case mô tả phân tích thực phẩm

Bảng 2.3: Use Case mô tả dashboard và thống kê

Bảng 2.4: Use Case mô tả quản lý người dùng

Bảng 2.5: Use Case mô tả quản lý mô hình AI

Bảng 2.6: Use Case mô tả cấu hình hệ thống

Bảng 2.7: Cấu trúc thư mục dự án Flutter

Bảng 3.1: Kết quả kiểm thử các chức năng chính

HÌNH VẼ

Hình 1.1: Nguyên lý hoạt động của công nghệ NIR

Hình 1.2: Thiết bị đo quang phổ Ocean FX

Hình 2.1: Sơ đồ kiến trúc tổng thể hệ thống

Hình 2.2: Sơ đồ luồng dữ liệu từ thiết bị đến ứng dụng

Hình 2.3: Sơ đồ kiến trúc Clean Architecture

Hình 2.4: Sơ đồ BLoC Pattern

Hình 3.1: Màn hình Login

Hình 3.2: Màn hình Dashboard

Hình 3.3: Màn hình Chi tiết phân tích thực phẩm

Hình 3.4: Menu quản lý

Hình 3.5: Trang xem thông tin tài khoản

Hình 3.6: Trang xem danh sách model AI

Hình 3.7: Màn hình quản lý user (Admin)

Hình 3.8: Màn hình chi tiết quản lý user (Admin)

Hình 3.9: Màn hình xem thông tin model AI đang sử dụng

Hình 3.10: Màn hình chỉnh sửa thông tin model AI đang sử dụng

Hình 3.11: Màn hình xem lịch sử phân tích

Hình 3.12: Màn hình xem chi tiết lịch sử phân tích

DANH SÁCH CÁC KÝ HIỆU, CHỮ VIẾT TẮT

KÝ HIỆU:

- λ: Bước sóng (wavelength)
- I: Cường độ ánh sáng (intensity)
- mg/kg: Milligram trên kilogram (đơn vị nồng độ)

CHỮ VIẾT TẮT:

- **NIR:** Near Infrared (Hồng ngoại gần)
- **USB:** Universal Serial Bus
- **API:** Application Programming Interface
- **JWT:** JSON Web Token
- **UI:** User Interface (Giao diện người dùng)
- **UX:** User Experience (Trải nghiệm người dùng)
- **BLoC:** Business Logic Component
- **MVVM:** Model-View-ViewModel
- **REST:** Representational State Transfer
- **HTTP:** HyperText Transfer Protocol
- **HTTPS:** HTTP Secure
- **WSS:** WebSocket Secure
- **JSON:** JavaScript Object Notation
- **CSV:** Comma-Separated Values
- **SDK:** Software Development Kit
- **IDE:** Integrated Development Environment
- **OOP:** Object-Oriented Programming

- **SOLID:** Single Responsibility, Open-Closed, Liskov Substitution, Interface Segregation, Dependency Inversion
- **IoT:** Internet of Things
- **TCP:** Transmission Control Protocol

MỞ ĐẦU

1. Mục đích thực hiện đề tài

Vấn đề an toàn thực phẩm đang là mối quan tâm hàng đầu của người tiêu dùng và các cơ quan quản lý. Việc kiểm tra chất lượng thực phẩm truyền thống thường tốn nhiều thời gian, chi phí và yêu cầu thiết bị phòng thí nghiệm phức tạp. Công nghệ quang phổ hồng ngoại gần (NIR) đã được chứng minh là một giải pháp hiệu quả để phân tích nhanh thành phần và chất lượng thực phẩm mà không cần phá hủy mẫu.

Mục đích của đồ án này là xây dựng một ứng dụng mobile trên nền tảng Flutter để người dùng có thể thực hiện việc phân tích thực phẩm một cách dễ dàng đảm bảo được thực phẩm an toàn khi sử dụng. Đảm bảo được sức khỏe của người dùng.

2. Mục tiêu đề tài

Đồ án hướng tới việc xây dựng một ứng dụng mobile bằng Flutter nhằm phục vụ cho việc đo lường và đánh giá chất lượng thực phẩm một cách trực quan và dễ sử dụng. Ứng dụng được phát triển với khả năng kết nối WebSocket để nhận dữ liệu quang phổ theo thời gian thực từ Raspberry Pi, sau đó tiến hành phân tích dữ liệu này và hiển thị kết quả đánh giá một cách chính xác, rõ ràng cho người dùng.

Ứng dụng được thiết kế theo hướng Clean Architecture để đảm bảo cấu trúc rõ ràng, dễ bảo trì và mở rộng trong tương lai. Hệ thống có khả năng kết nối và duy trì WebSocket ổn định, lắng nghe dữ liệu liên tục và tự động kết nối lại khi xảy ra gián đoạn. Ứng dụng cũng hỗ trợ phân tích thực phẩm thông qua hai hình thức là nhận dữ liệu trực tiếp từ WebSocket hoặc cho phép người dùng tải lên file dữ liệu quang phổ, sau đó gửi dữ liệu đến API để nhận kết quả xử lý.

Ngoài chức năng phân tích, ứng dụng cung cấp dashboard hiển thị các biểu đồ thống kê và lịch sử các lần kiểm tra, giúp người dùng dễ dàng theo dõi và so sánh kết quả. Hệ thống xác thực người dùng được triển khai bằng JWT nhằm đảm bảo an toàn thông tin, đồng thời ứng dụng hỗ trợ đa ngôn ngữ bao gồm tiếng Việt và tiếng Anh để phù hợp với nhiều đối tượng sử dụng.

3. Phạm vi và đối tượng nghiên cứu

Phạm vi nghiên cứu:

- Tập trung vào phát triển phần Frontend ứng dụng mobile trên nền tảng Flutter
- Kết nối với WebSocket server để nhận dữ liệu quang phổ real-time từ Raspberry Pi
- Tích hợp với API backend FastAPI đã có sẵn
- Ứng dụng chạy trên nền tảng Android và iOS

Đối tượng nghiên cứu:

- Framework Flutter và ngôn ngữ Dart
- Công nghệ WebSocket để truyền dữ liệu real-time
- Kiến trúc Clean Architecture và BLoC Pattern
- Thiết bị đo quang phổ Ocean FX và hệ thống Raspberry Pi
- Thư viện python-seabreeze và SeaBreeze C driver
- Công nghệ quang phổ hồng ngoại gần (NIR)
- Các mô hình AI/ML để phân tích dữ liệu quang phổ

Giới hạn của đồ án:

- Không bao gồm việc phát triển phần backend API (sử dụng API có sẵn)
- Không bao gồm việc phát triển phần Raspberry Pi và WebSocket server (sử dụng hệ thống có sẵn)
- Không bao gồm việc huấn luyện các mô hình AI/ML (sử dụng mô hình đã được huấn luyện)
- Không bao gồm việc phát triển phần cứng thiết bị Ocean FX

4. Phương pháp nghiên cứu

Đồ án được em thực hiện dựa trên việc kết hợp nghiên cứu lý thuyết và triển khai thực tế. Em tìm hiểu các công nghệ liên quan như Flutter, Dart, WebSocket, Clean Architecture, quang phổ NIR và thiết bị Ocean FX để làm nền tảng phát triển ứng dụng. Trên cơ sở đó, em xây dựng ứng dụng theo quy trình Agile, sử dụng các công cụ như Flutter SDK và Android Studio, đồng thời kiểm thử trên thiết bị Android, iOS và tích hợp với Raspberry Pi thông qua WebSocket. Cuối cùng, ứng dụng được đánh giá

thông qua kiểm thử chức năng, giao diện, hiệu năng và độ chính xác của kết quả phân tích.

5. Cấu trúc của đồ án tốt nghiệp

Đồ án được chia thành 3 chương chính:

Chương 1: TỔNG QUAN VỀ CÔNG NGHỆ VÀ THIẾT BỊ

Trình bày tổng quan về công nghệ quang phổ NIR, thiết bị Ocean FX, công nghệ WebSocket, framework Flutter và các kiến trúc phần mềm được sử dụng.

Chương 2: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

Phân tích yêu cầu hệ thống, thiết kế kiến trúc ứng dụng theo Clean Architecture, thiết kế các module chính và giao diện người dùng.

Chương 3: TRIỂN KHAI VÀ KẾT QUẢ

Trình bày quá trình triển khai các module chính của ứng dụng, kết quả đạt được và đánh giá ứng dụng.

Chương 1: TỔNG QUAN VỀ CÔNG NGHỆ VÀ THIẾT BỊ

1.1. Tổng quan về công nghệ quang phổ hồng ngoại gần (NIR)

1.1.1. Nguyên lý hoạt động của công nghệ NIR

Quang phổ hồng ngoại gần (Near Infrared - NIR) là một kỹ thuật phân tích quang phổ sử dụng dải bước sóng từ 780nm đến 2500nm. Công nghệ này dựa trên nguyên lý hấp thụ ánh sáng của các phân tử trong mẫu vật khi được chiếu sáng bằng ánh sáng hồng ngoại gần.

Khi ánh sáng NIR chiếu vào mẫu thực phẩm, các liên kết hóa học như C-H, O-H, N-H sẽ hấp thụ năng lượng ở các bước sóng đặc trưng. Mức độ hấp thụ phụ thuộc vào nồng độ và loại chất có trong mẫu. Bằng cách đo cường độ ánh sáng phản xạ hoặc truyền qua mẫu ở các bước sóng khác nhau, ta có thể xác định được thành phần và nồng độ các chất trong mẫu.

Ưu điểm của công nghệ NIR: Phương pháp này cho phép phân tích rất nhanh, chỉ mất từ vài giây đến vài phút để có kết quả. Quá trình đo không làm hư hỏng mẫu, không cần xử lý mẫu phức tạp và có thể áp dụng trực tiếp cho cả mẫu rắn lẫn mẫu lỏng. Ngoài ra, chi phí vận hành thấp và phương pháp này cũng thân thiện với môi trường.

Ứng dụng trong phân tích thực phẩm: Phương pháp này có thể được sử dụng để xác định hàm lượng nước, protein, chất béo và carbohydrate trong thực phẩm. Bên cạnh đó, nó còn hỗ trợ phát hiện các chất độc hại như thuốc trừ sâu hay chất bảo quản, giúp phân loại và nhận dạng các loại thực phẩm khác nhau, đồng thời đánh giá độ tươi và chất lượng của sản phẩm.

1.1.2. Dữ liệu quang phổ và định dạng file

Dữ liệu quang phổ được lưu trữ dưới dạng các cặp giá trị (wavelength, intensity), trong đó:

- **Wavelength (λ)**: Bước sóng ánh sáng, đơn vị nanomet (nm)
- **Intensity (I)**: Cường độ ánh sáng đo được tại bước sóng tương ứng

Trong dự án này, dữ liệu quang phổ từ thiết bị Ocean FX được lưu dưới định dạng file text (.txt) với cấu trúc:

Data from [Tên file].txt Node

Date: [Ngày tháng]

User: [Tên người dùng]

Spectrometer: [Số serial thiết bị]

Trigger mode: [Chế độ kích hoạt]

Integration Time (sec): [Thời gian tích hợp]

Scans to average: [Số lần quét trung bình]

Number of Pixels in Spectrum: [Số điểm dữ liệu]

>>>>>Begin Spectral Data<<<<<

[Wavelength] [Intensity]

316.194 2720.9

316.585 2716.26667

...

File mẫu trong dự án có 2136 điểm dữ liệu, dải bước sóng từ khoảng 316nm đến 1029nm.

1.2. Thiết bị đo quang phổ Ocean FX và hệ thống Raspberry Pi

1.2.1. Giới thiệu về thiết bị Ocean FX

Ocean FX là thiết bị đo quang phổ cầm tay của hãng Ocean Insight, được thiết kế để đo quang phổ trong dải NIR với độ chính xác cao. Thiết bị hỗ trợ kết nối qua USB, cho phép tích hợp với các hệ thống máy tính và Raspberry Pi để thu thập dữ liệu quang phổ.

Đặc điểm chính:

- Thiết kế nhỏ gọn, cầm tay
- Kết nối qua USB với máy tính/Raspberry Pi
- Thời gian đo nhanh

1.2.2. Thông số kỹ thuật

Thông số	Giá trị
Dải bước sóng	316nm - 1029nm
Số pixels	2136
Độ phân giải quang phổ	0.33(nm)
Thời gian tích hợp	0.013 giây (13ms)
Số lần quét trung bình	5
Giao tiếp	Usb
Nguồn điện	Usb hoặc pin tích hợp
Kích thước	Nhỏ gọn cầm tay

Bảng 1.1: Thông số kỹ thuật thiết bị Ocean FX

1.2.2. Hệ thống Raspberry Pi và SeaBreeze

Raspberry Pi làm trung gian:

Raspberry Pi đóng vai trò là máy tính trung gian giữa thiết bị Ocean FX và ứng dụng mobile. Raspberry Pi kết nối với Ocean FX qua USB và sử dụng thư viện SeaBreeze để điều khiển và thu thập dữ liệu từ thiết bị.

Kiến trúc phần mềm trên Raspberry Pi:

```
Ocean FX (USB)
   ↓
libusb (Linux USB library)
```

```
   ↓
SeaBreeze (C driver)
   ↓
python-seabreeze (Python wrapper)
   ↓
Python Application (Thu thập dữ liệu)
   ↓
WebSocket Client
   ↓
WebSocket Server
```

Thư viện SeaBreeze:

SeaBreeze là driver C mã nguồn mở được phát triển bởi Ocean Optics để điều khiển các thiết bị đo quang phổ của họ. SeaBreeze cung cấp API để:

- Kết nối và quản lý thiết bị qua USB
- Điều khiển các thông số đo (thời gian tích hợp, số lần quét trung bình)
- Thu thập dữ liệu quang phổ
- Đọc thông tin thiết bị (serial number, model, firmware version)

Thư viện python-seabreeze:

python-seabreeze là wrapper Python cho SeaBreeze C driver, cho phép sử dụng SeaBreeze từ Python một cách dễ dàng. Thư viện này cung cấp:

- Interface Python đơn giản để điều khiển thiết bị
- Tự động quản lý kết nối USB
- Xử lý lỗi và exception handling
- Hỗ trợ đa nền tảng (Linux, Windows, macOS)

Quy trình thu thập dữ liệu trên Raspberry Pi:

1. Kết nối Ocean FX với Raspberry Pi qua USB
2. Khởi tạo SeaBreeze và tìm thiết bị
3. Cấu hình thông số đo (integration time, scans to average)
4. Thu thập dữ liệu quang phổ (wavelength và intensity)
5. Format dữ liệu thành JSON
6. Gửi dữ liệu lên WebSocket server

1.3. Hệ thống Raspberry Pi và SeaBreeze

1.3.1. Raspberry Pi

Raspberry Pi là một máy tính nhỏ gọn, giá rẻ được sử dụng rộng rãi trong các dự án IoT và embedded systems. Trong dự án này, Raspberry Pi đóng vai trò là máy tính trung gian giữa thiết bị Ocean FX và ứng dụng mobile.

Vai trò của Raspberry Pi:

- Kết nối với Ocean FX qua USB
- Chạy Python application để điều khiển thiết bị
- Thu thập dữ liệu quang phổ từ thiết bị
- Kết nối với WebSocket server để gửi dữ liệu
- Xử lý và format dữ liệu trước khi gửi

Hệ điều hành:

- Raspberry Pi OS (Linux-based)
- Hỗ trợ libusb để giao tiếp với thiết bị USB
- Chạy Python runtime để thực thi ứng dụng

1.3.2. Thư viện SeaBreeze

SeaBreeze C Driver:

SeaBreeze là driver C mã nguồn mở được phát triển bởi Ocean Optics (nay là Ocean Insight) để điều khiển các thiết bị đo quang phổ của họ.

Cài đặt SeaBreeze trên Raspberry Pi:

```
# Clone SeaBreeze repository
git clone https://github.com/ap--/seabreeze.git
cd seabreeze

# Build và install
make
sudo make install
```

libusb:

SeaBreeze phụ thuộc vào libusb, một thư viện USB của Linux cho phép ứng dụng user-space giao tiếp với thiết bị USB mà không cần kernel driver. libusb cung cấp:

- API để enumerate và kết nối với thiết bị USB
- Đọc/ghi dữ liệu với thiết bị USB
- Quản lý USB descriptors và endpoints

1.3.3. Thư viện python-seabreeze

python-seabreeze là wrapper Python cho SeaBreeze C driver, cho phép sử dụng SeaBreeze từ Python một cách dễ dàng.

Cài đặt:

```
pip install seabreeze
```

Sử dụng cơ bản:

```
import seabreeze.spectrometers as sb

# Tìm và kết nối với thiết bị
devices = sb.list_spectrometers()
spec = sb.Spectrometer(devices[0])

# Cấu hình thông số đo
spec.integration_time_micros(13000)  # 13ms
spec.scans_to_average(5)

# Thu thập dữ liệu quang phổ
wavelengths = spec.wavelengths()
intensities = spec.intensities()

# Đóng kết nối
spec.close()
```

1.4. Công nghệ WebSocket

1.4.1. Tổng quan về WebSocket

WebSocket là một giao thức truyền thông hai chiều (bidirectional) cho phép client và server trao đổi dữ liệu real-time qua một kết nối TCP duy nhất. Khác với HTTP request-response truyền thống, WebSocket duy trì kết nối mở và cho phép cả hai phía gửi dữ liệu bất cứ lúc nào.

1.4.2. WebSocket trong Flutter

Trong dự án này, ứng dụng Flutter sử dụng WebSocket để nhận dữ liệu quang phổ real-time từ server. Flutter hỗ trợ WebSocket thông qua package web_socket_channel.

Các tính năng cần thiết:

- Kết nối với WebSocket server
- Lắng nghe và nhận dữ liệu real-time
- Xử lý reconnection khi mất kết nối
- Xử lý lỗi và timeout
- Đóng kết nối khi không cần thiết

Luồng hoạt động:

1. App Flutter kết nối với WebSocket server
2. Server nhận dữ liệu từ Raspberry Pi
3. Server broadcast dữ liệu đến tất cả client
4. App Flutter nhận dữ liệu và cập nhật UI
5. App có thể gửi commands đến server (nếu cần)

1.5. Framework Flutter và ngôn ngữ Dart

1.5.1. Giới thiệu về Flutter

Flutter là framework mã nguồn mở của Google để phát triển ứng dụng mobile đa nền tảng (iOS, Android, Web, Desktop). Flutter sử dụng ngôn ngữ Dart và có khả năng biên dịch thành mã máy (native code), đảm bảo hiệu năng cao.

Ưu điểm của Flutter:

- **Hot Reload**: Cập nhật UI ngay lập tức khi code thay đổi
- **Single Codebase**: Viết một lần, chạy trên nhiều nền tảng
- **Hiệu năng cao**: Biên dịch thành native code
- **Widget-based**: UI được xây dựng từ các widget có thể tái sử dụng
- **Rich ecosystem**: Nhiều package và plugin có sẵn
- **Material Design & Cupertino**: Hỗ trợ cả Material Design và iOS design

Version sử dụng trong dự án: Flutter 3.35.6, Dart SDK $^{3.9.2}$

1.5.2. Ngôn ngữ Dart

Dart là ngôn ngữ lập trình được phát triển bởi Google, được thiết kế đặc biệt cho phát triển ứng dụng client-side. Dart là ngôn ngữ hướng đối tượng, có kiểu tĩnh (statically typed) và hỗ trợ null safety.

Đặc điểm của Dart:

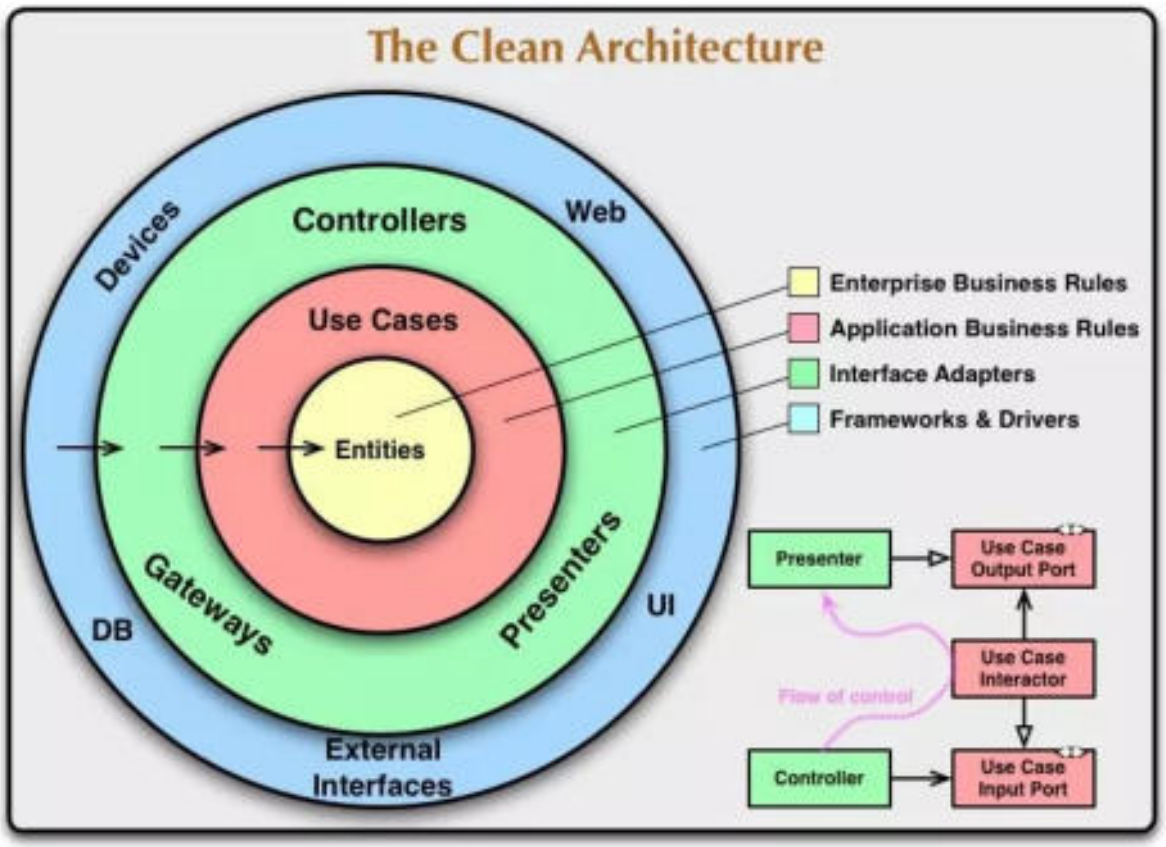
- **Null Safety**: Đảm bảo an toàn khi làm việc với giá trị null
- **Async/Await**: Hỗ trợ tốt cho lập trình bất đồng bộ
- **Stream**: Xử lý luồng dữ liệu hiệu quả

- **Mixins**: Cho phép tái sử dụng code
- **Generics**: Lập trình tổng quát

1.6. Kiến trúc Clean Architecture và BLoC Pattern

1.5.1. Clean Architecture

Clean Architecture là một kiến trúc phần mềm được đề xuất bởi Robert C. Martin (Uncle Bob), nhằm tách biệt các lớp của ứng dụng để dễ bảo trì, kiểm thử và mở rộng.



Các lớp trong Clean Architecture:

1. **Presentation Layer (Lớp trình bày)**
    a. Chứa UI, Widgets, BLoC
    b. Xử lý tương tác người dùng
    c. Hiển thị dữ liệu
2. **Domain Layer (Lớp nghiệp vụ)**
    a. Chứa Entities (đối tượng nghiệp vụ)

b. Chứa Use Cases (logic nghiệp vụ)

c. Chứa Repository Interfaces (giao diện repository)

3. **Data Layer (Lớp dữ liệu)**

a. Chứa Data Sources (API, Database, Local Storage)

b. Chứa Models (đối tượng dữ liệu)

c. Chứa Repository Implementations (triển khai repository)

Nguyên tắc Dependency Rule:

- Các lớp bên ngoài phụ thuộc vào các lớp bên trong
- Domain layer không phụ thuộc vào bất kỳ layer nào khác
- Data layer phụ thuộc vào Domain layer

Lợi ích:

- Dễ kiểm thử (có thể mock các dependency)
- Dễ bảo trì (thay đổi một layer không ảnh hưởng đến layer khác)
- Dễ mở rộng (thêm tính năng mới không cần sửa code cũ)
- Tái sử dụng code cao

1.5.2. BLoC Pattern

BLoC (Business Logic Component) là một pattern quản lý state được đề xuất bởi Google, đặc biệt phù hợp với Flutter. BLoC tách biệt logic nghiệp vụ khỏi UI, giúp code dễ kiểm thử và bảo trì.

Các thành phần của BLoC:

1. **Event**: Đại diện cho các hành động từ UI
2. **State**: Đại diện cho trạng thái hiện tại của ứng dụng
3. **BLoC**: Xử lý logic, nhận Event và emit State mới

Luồng hoạt động:

```
UI → Event → BLoC → State → UI
```

Ưu điểm:

1. Tách biệt logic và UI
2. Dễ kiểm thử (test BLoC độc lập)
3. Có thể tái sử dụng logic ở nhiều nơi
4. Dễ debug (có thể log tất cả events và states)

Chương 2: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

2.1. Phân tích yêu cầu hệ thống

2.1.1. Yêu cầu chức năng

Hệ thống đo lường và đánh giá thực phẩm thông minh bao gồm hai tác nhân chính:

- **Người dùng (User)**: Đối tượng sử dụng chính của hệ thống, có các chức năng phân tích thực phẩm, xem thống kê cá nhân và quản lý dữ liệu
- **Quản trị viên hệ thống (Admin)**: Có đầy đủ chức năng như User, đồng thời có thêm các chức năng quản lý hệ thống (quản lý người dùng, quản lý mô hình AI, xem thống kê tổng quan)

Người dùng (User)

- Quản lý tài khoản và xác thực (đăng nhập)
- Phân tích thực phẩm (nhận dữ liệu quang phổ từ file upload hoặc kết nối thiết bị, phân tích bằng AI/ML, xem kết quả và chi tiết)
- Dashboard và thống kê (xem số lượt kiểm tra, biểu đồ mức độ an toàn, biểu đồ phân tích theo loại thực phẩm, lịch sử kiểm tra với tìm kiếm và lọc)

Quản trị viên hệ thống (Admin)

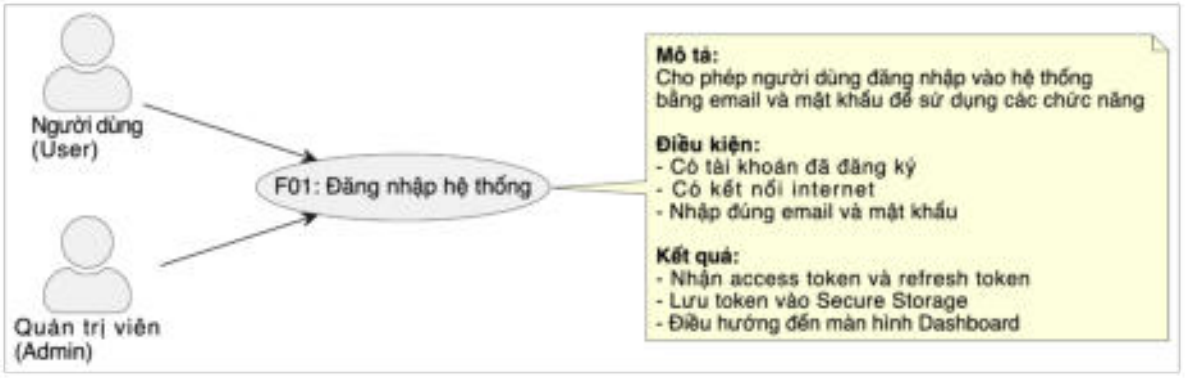
Admin có đầy đủ các chức năng như User, đồng thời có thêm các chức năng quản lý hệ thống:

- Quản lý tài khoản và xác thực
- Phân tích thực phẩm
- Dashboard và thống kê (bao gồm dashboard cá nhân và dashboard tổng quan hệ thống)
- Quản lý người dùng (xem danh sách, chi tiết, reset mật khẩu, khóa/mở khóa tài khoản)
- Quản lý mô hình AI (xem danh sách mô hình với thông tin cơ bản)

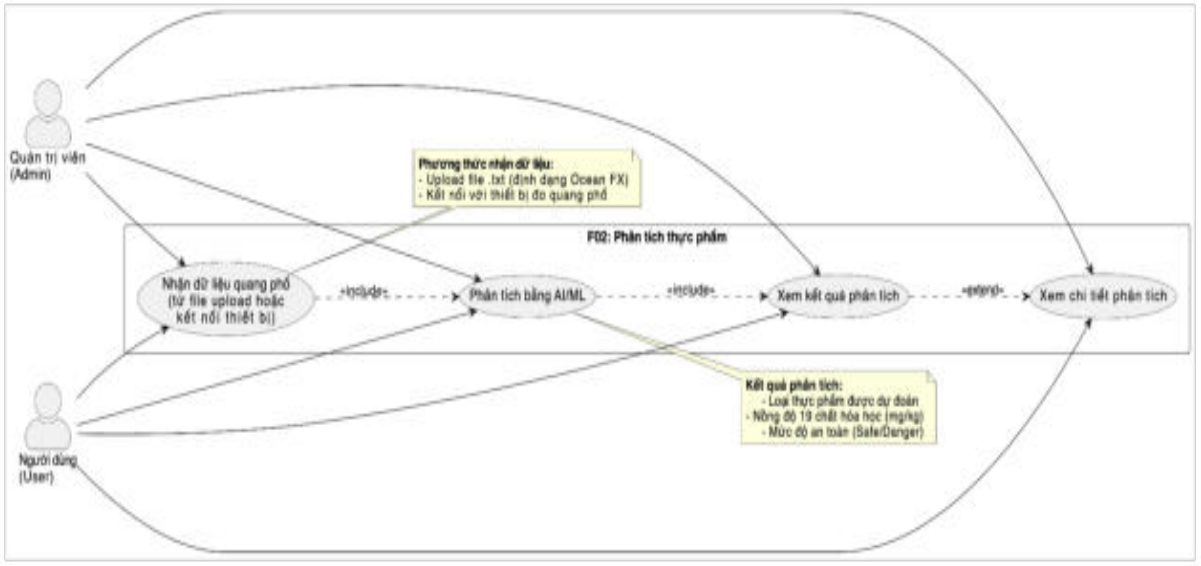
- Cấu hình hệ thống (xem mô hình AI hiện tại theo user, cập nhật ngưỡng 19 chất và các thông số hệ thống)

2.1.2. Sơ đồ use-case

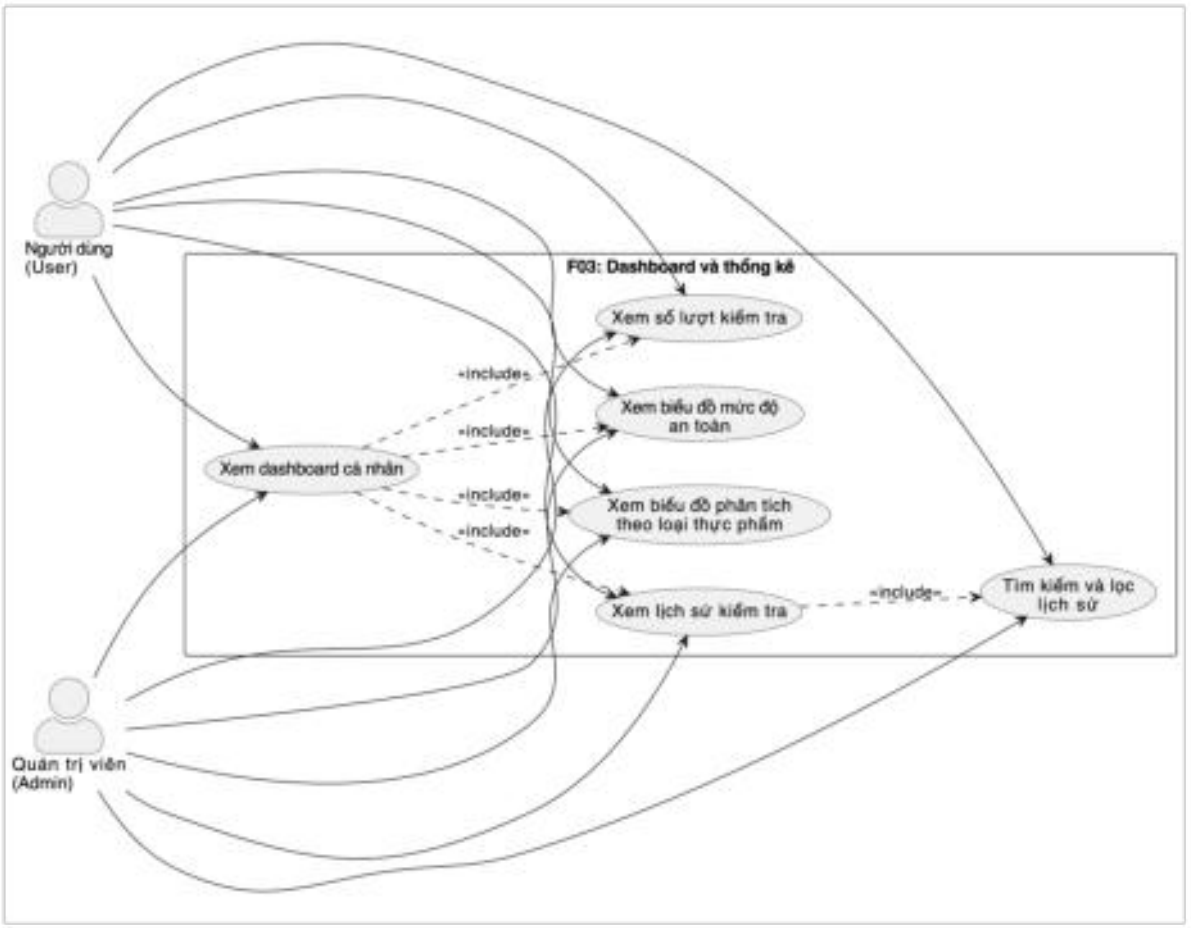
Để minh họa rõ ràng các chức năng của hệ thống, sơ đồ use-case được xây dựng với hai tác nhân chính: Người dùng (User) và Quản trị viên (Admin). Mỗi chức năng (F01-F06) được trình bày trong một sơ đồ use-case riêng biệt.



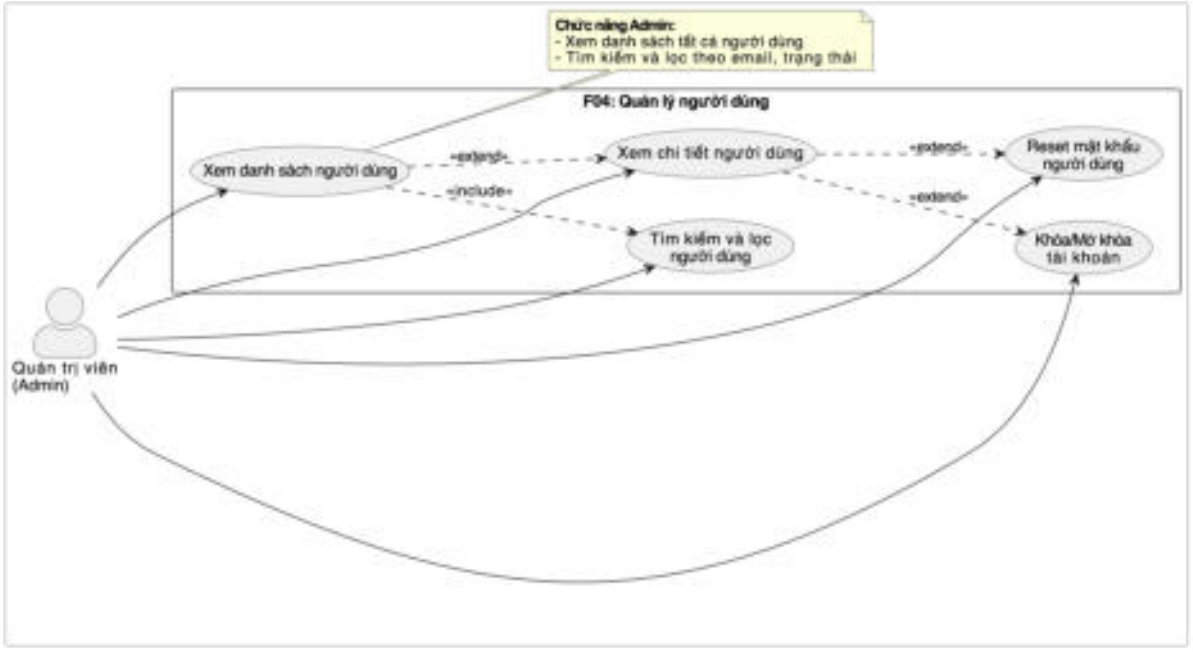
Hình 2.5: Sơ đồ use-case - F01: Đăng nhập



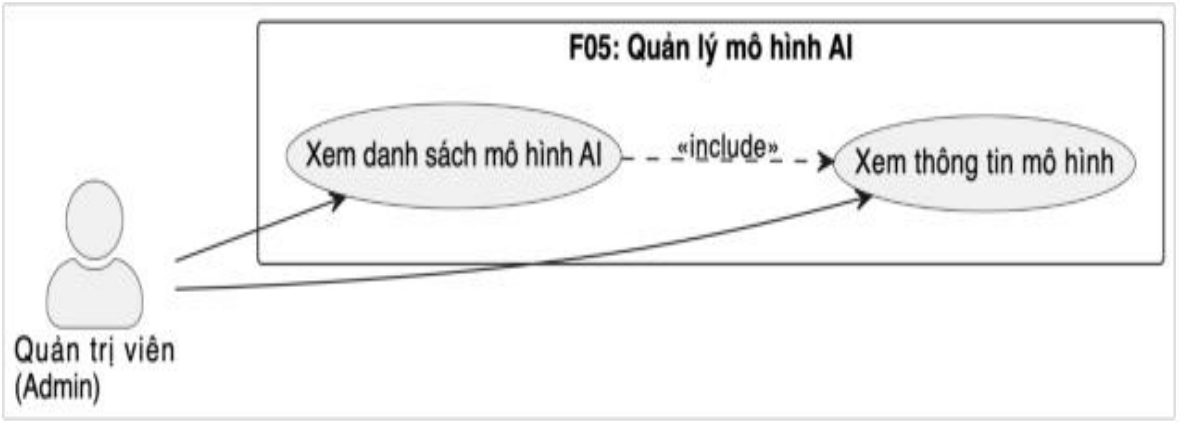
Hình 2.6: Sơ đồ use-case - F02: Phân tích thực phẩm



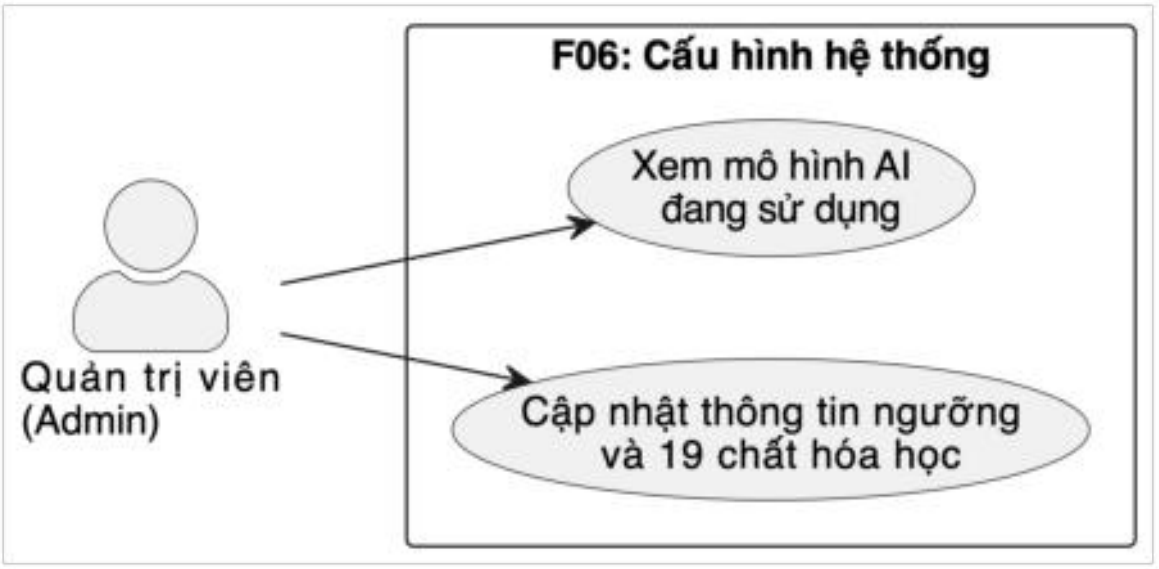
Hình 2.7: Sơ đồ use-case - F03: Dashboard và thống kê



Hình 2.8: Sơ đồ use-case - F04: Quản lý người dùng



Hình 2.9: Sơ đồ use-case - F05: Quản lý mô hình AI



Hình 2.10: Sơ đồ use-case - F06: Cấu hình hệ thống

Mô tả chi tiết các chức năng:

Chi tiết mô tả từng chức năng (F01-F06) được trình bày dưới dạng bảng như sau:

F01: Đăng nhập

Thuộc tính	Mô tả
Tên tính năng	Đăng nhập hệ thống
Mô tả & Mục đích	Cho phép người dùng đăng nhập vào hệ thống bằng email và mật khẩu để sử dụng các chức năng
Người dùng	User, Admin
Điều kiện thực hiện	- Có tài khoản đã đăng ký - Có kết nối internet - Nhập đúng email và mật khẩu
Quy tắc nghiệp vụ	- Token JWT có thời hạn, tự động refresh khi hết hạn- Lưu trữ token an toàn bằng Secure Storage
Các trạng thái	- Loading: Đang xử lý đăng nhập - Thành công: Chuyển đến màn hình chính - Lỗi: Hiển thị thông báo lỗi (sai mật khẩu, tài khoản không tồn tại, tài khoản bị khóa) - Trống: Form đăng nhập
Kết quả	- Nhận access token và refresh token - Lưu token vào Secure Storage - Điều hướng đến màn hình Dashboard
Luồng xử lý	1. Người dùng nhập email và mật khẩu

	2. Validate dữ liệu đầu vào 3. Gửi request đến API /auth/token 4. Nhận token và lưu trữ 5. Điều hướng đến màn hình chính

Bảng 2.1 Use Case mô tả đăng nhập

F02: Phân tích thực phẩm

Thuộc tính	Mô tả
Tên tính năng	Phân tích thực phẩm
Mô tả & Mục đích	Cho phép người dùng nhận dữ liệu quang phổ (từ file upload hoặc kết nối thiết bị), phân tích bằng AI/ML để xác định loại thực phẩm, nồng độ các chất hóa học và mức độ an toàn, xem kết quả phân tích và chi tiết
Người dùng	User, Admin
Điều kiện thực hiện	- Đã đăng nhập - Có kết nối internet - Có dữ liệu quang phổ (từ file upload hoặc kết nối thiết bị) - Dữ liệu hợp lệ (2136 điểm dữ liệu)
Quy tắc nghiệp vụ	- **Nhận dữ liệu:** Có thể upload file .txt hoặc kết nối với thiết bị để nhận dữ liệu quang phổ - **Upload file:** Chỉ chấp nhận file .txt, có

	định dạng Ocean FX, kích thước ≤ 10MB, parse và validate dữ liệu - **Kết nối thiết bị:** Kết nối với thiết bị đo quang phổ để nhận dữ liệu real-time, tự động reconnect khi mất kết nối, hiển thị trạng thái kết nối - **Phân tích:** Dữ liệu phải có đủ 2136 điểm intensity, gửi request đến API `/ai/predict`, thời gian phân tích tối đa 30 giây, tự động retry nếu lỗi mạng (tối đa 3 lần) - **Hiển thị kết quả:** Loại thực phẩm, nồng độ các chất (mg/kg), mức độ an toàn (Safe/Danger), ngưỡng cho phép, biểu đồ quang phổ
Các trạng thái	- **Nhận dữ liệu:** Đang kết nối thiết bị/đang chọn file, đã kết nối/đã chọn file, đang nhận dữ liệu/đang parse, ngắt kết nối/lỗi file - **Phân tích:** Loading (đang gửi request), Thành công (nhận kết quả), Lỗi (mạng/API/dữ liệu không hợp lệ), Timeout - **Hiển thị:** Có dữ liệu (hiển thị kết quả), Trống (chưa có kết quả), Loading (đang tải chi tiết)
Kết quả	- Nhận dữ liệu thành công (từ file hoặc thiết bị) - Dữ liệu quang phổ được trích xuất và validate - Loại thực phẩm được dự đoán với xác

	suất - Nồng độ 19 chất hóa học (mg/kg) - Mức độ an toàn (Safe/Danger) - Biểu đồ quang phổ và bảng so sánh với ngưỡng cho phép
Luồng xử lý	1. Người dùng chọn phương thức nhận dữ liệu (upload file hoặc kết nối thiết bị) 2. Nếu kết nối thiết bị: Kết nối với thiết bị, nhận dữ liệu real-time, parse và validate 3. Nếu upload file: Chọn file, validate định dạng và kích thước, parse dữ liệu 4. Người dùng nhấn "Phân tích" 5. Validate dữ liệu quang phổ (2136 điểm) 6. Gửi request đến API `/ai/predict` 7. Nhận response và xử lý kết quả 8. Hiển thị kết quả phân tích với biểu đồ và bảng 9. Cho phép xem chi tiết (biểu đồ quang phổ, bảng nồng độ chất)

Bảng 2.2 Use Case mô tả phân tích thực phẩm

F03: Dashboard và thống kê

Thuộc tính	Mô tả
Tên tính năng	Dashboard và thống kê

Mô tả & Mục đích	Hiển thị tổng quan thống kê cá nhân (User/Admin) hoặc tổng quan hệ thống (Admin), bao gồm số lượt kiểm tra, biểu đồ mức độ an toàn, biểu đồ phân tích theo loại thực phẩm, lịch sử kiểm tra với tìm kiếm và lọc
Người dùng	User, Admin
Điều kiện thực hiện	- Đã đăng nhập - Có kết nối internet
Quy tắc nghiệp vụ	- **Dashboard cá nhân:** Lấy dữ liệu từ API `/dashboard/statistics`, hiển thị thống kê dựa trên lịch sử kiểm tra cá nhân, tự động refresh khi có dữ liệu mới- **Dashboard tổng quan (Admin):** Lấy dữ liệu từ API `/dashboard/admin/statistics`, hiển thị thống kê tổng thể của toàn hệ thống, chỉ Admin có quyền truy cập - **Lịch sử kiểm tra:** Lấy từ API `/dashboard/history`, sắp xếp theo thời gian mới nhất, hỗ trợ tìm kiếm và lọc theo loại thực phẩm, mức độ an toàn, phân trang
Các trạng thái	- Loading: Đang tải dữ liệu - Có dữ liệu: Hiển thị dashboard/thống kê/lịch sử - Trống: Chưa có lịch sử kiểm tra - Lỗi: Không có quyền truy cập (Admin dashboard) hoặc lỗi API
Kết quả	- Hiển thị dashboard với các chỉ số KPI

	(số lượt kiểm tra, số mô hình AI - Admin)- Biểu đồ mức độ an toàn (Safe/Danger) - Biểu đồ phân tích theo loại thực phẩm - Lịch sử kiểm tra gần đây với tìm kiếm và lọc - Cho phép xem chi tiết từng lần kiểm tra
Luồng xử lý	1. Người dùng vào màn hình Dashboard 2. Kiểm tra quyền (User/Admin) 3. Gọi API lấy thống kê (cá nhân hoặc tổng quan) 4. Hiển thị các KPI cards và biểu đồ 5. Hiển thị lịch sử kiểm tra gần đây 6. Cho phép tìm kiếm và lọc lịch sử 7. Cho phép xem chi tiết từng bản ghi

Bảng 2.3 Use Case mô tả dashboard và thống kê

F04: Quản lý người dùng

Thuộc tính	Mô tả
Tên tính năng	Quản lý người dùng
Mô tả & Mục đích	Cho phép Admin xem danh sách tất cả người dùng, xem chi tiết thông tin người dùng (hồ sơ, lịch sử hoạt động), reset mật khẩu và khóa/mở khóa tài khoản người dùng

Người dùng	Admin
Điều kiện thực hiện	- Đã đăng nhập với quyền Admin- Có kết nối internet
Quy tắc nghiệp vụ	- **Xem danh sách:** Lấy dữ liệu từ API /users/, hiển thị với phân trang, hỗ trợ tìm kiếm và lọc theo email, trạng thái- **Xem chi tiết:** Lấy dữ liệu từ API /users/:id, hiển thị đầy đủ thông tin: email, ngày tạo, trạng thái, lịch sử hoạt động- **Reset mật khẩu:** Gửi request đến API /users/:id/reset-password, mật khẩu mới được tạo tự động và gửi qua email, yêu cầu xác nhận trước khi reset- **Khóa/Mở khóa:** Gửi request đến API /users/:id/status, khóa tài khoản thì người dùng không thể đăng nhập, yêu cầu xác nhận trước khi thực hiện
Các trạng thái	- Loading: Đang tải danh sách/thông tin - Có dữ liệu: Hiển thị danh sách/thông tin chi tiết - Trống: Không có người dùng nào - Xác nhận: Hiển thị dialog xác nhận (reset password, khóa/mở khóa) - Thành công: Thông báo đã thực hiện thành công - Lỗi: Không có quyền hoặc lỗi API
Kết quả	- Hiển thị danh sách người dùng với thông tin cơ bản - Hiển thị thông tin chi tiết người dùng

	và lịch sử hoạt động - Mật khẩu mới được tạo và gửi đến email người dùng (reset password) - Trạng thái tài khoản được cập nhật (khóa/mở khóa)
Luồng xử lý	1. Admin vào màn hình quản lý người dùng 2. Kiểm tra quyền Admin 3. Gọi API lấy danh sách người dùng 4. Hiển thị danh sách với tìm kiếm/lọc 5. Admin chọn một người dùng để xem chi tiết 6. Gọi API lấy chi tiết người dùng 7. Hiển thị thông tin chi tiết và lịch sử hoạt động 8. Admin có thể thực hiện: reset mật khẩu, khóa/mở khóa tài khoản 9. Xác nhận và gửi request đến API 10. Nhận kết quả và thông báo

Bảng 2.4 Use Case mô tả quản lý người dùng

F05: Quản lý mô hình AI

Thuộc tính	Mô tả
Tên tính năng	Xem danh sách mô hình AI
Mô tả & Mục đích	Cho phép Admin xem danh sách tất cả các mô hình AI trong hệ thống với các

	thông tin cơ bản như tên, phiên bản, trạng thái (active/inactive)
Người dùng	Admin
Điều kiện thực hiện	- Đã đăng nhập với quyền Admin- Có kết nối internet
Quy tắc nghiệp vụ	- **Xem danh sách:** Lấy dữ liệu từ API /training_models/, hiển thị với thông tin: tên mô hình, phiên bản, trạng thái (active/inactive), độ chính xác (nếu có), hỗ trợ lọc theo trạng thái- Chỉ hiển thị danh sách, không có chức năng xem chi tiết hay cập nhật trạng thái
Các trạng thái	- Loading: Đang tải danh sách - Có dữ liệu: Hiển thị danh sách mô hình - Trống: Không có mô hình nào - Lỗi: Lỗi API hoặc không có quyền
Kết quả	- Hiển thị danh sách mô hình AI với thông tin cơ bản (tên, phiên bản, trạng thái, độ chính xác)
Luồng xử lý	1. Admin vào màn hình quản lý mô hình AI 2. Kiểm tra quyền Admin 3. Gọi API lấy danh sách mô hình 4. Hiển thị danh sách với thông tin cơ bản và lọc theo trạng thái

Bảng 2.5 Use Case mô tả quản lý mô hình AI

F06: Cấu hình hệ thống

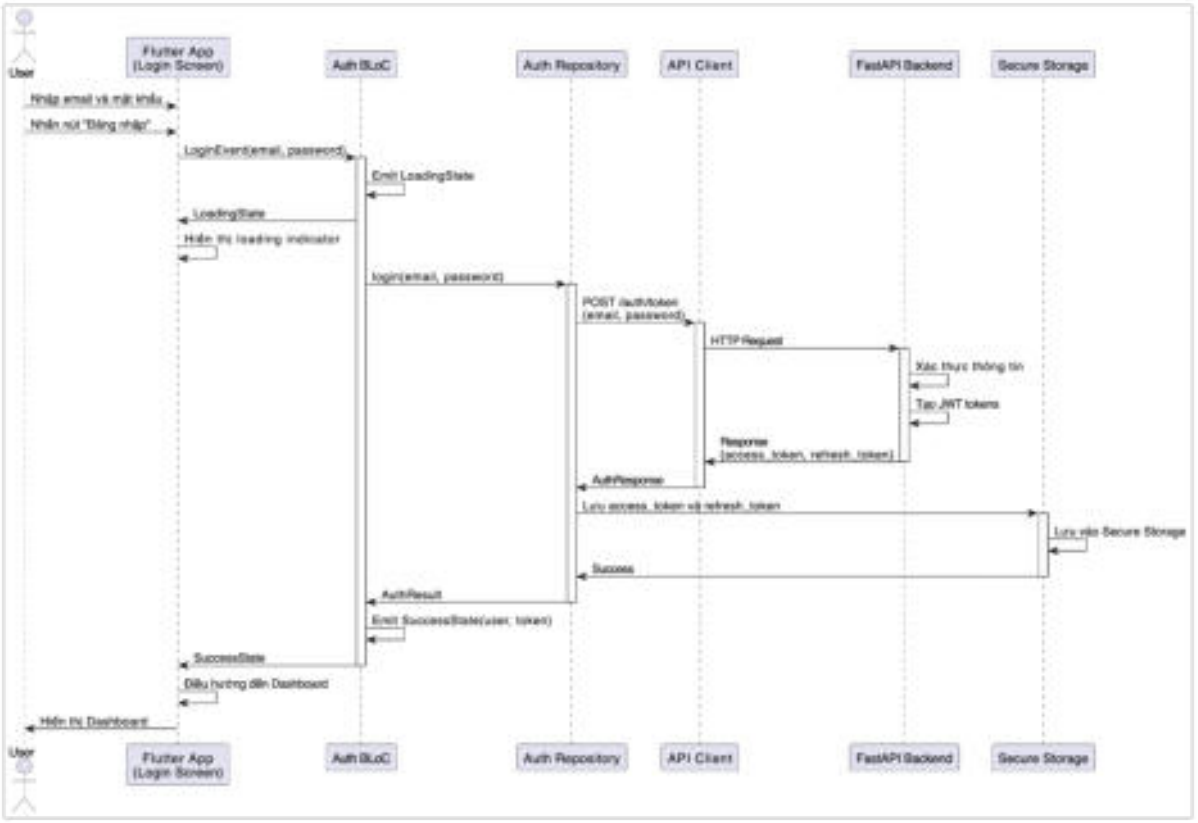
Thuộc tính	Mô tả
Tên tính năng	Cấu hình hệ thống
Mô tả & Mục đích	Cho phép Admin xem mô hình AI hiện tại được sử dụng , và cập nhật các thông số cấu hình hệ thống bao gồm ngưỡng cho phép của 19 chất hóa học cho các loại thực phẩm.
Người dùng	Admin
Điều kiện thực hiện	- Đã đăng nhập với quyền Admin - Có kết nối internet
Quy tắc nghiệp vụ	- Chỉ Admin mới có quyền thay đổi cấu hình - Xem mô hình AI hiện tại - Cập nhật ngưỡng cho phép của 19 chất hóa học (mg/kg)
Các trạng thái	- Có dữ liệu: Hiển thị form cấu hình với các giá trị hiện tại - Loading: Đang tải/lưu cấu hình - Thành công: Thông báo đã lưu cấu hình - Lỗi: Lỗi validation (giá trị không hợp lệ) hoặc lỗi API
Kết quả	- Hiển thị mô hình AI hiện tại dùng cho việc phân tích - Hiển thị form cấu hình với ngưỡng 19

	chất hóa học và các thông số khác - Cấu hình hệ thống được cập nhật - Các thay đổi được áp dụng ngay lập tức cho các phân tích tiếp theo
Luồng xử lý	1. Admin vào màn hình cấu hình hệ thống 2. Gọi API lấy mô hình AI hiện tại 4. Gọi API lấy cấu hình hiện tại (ngưỡng 19 chất, các thông số khác) 5. Hiển thị form cấu hình với các giá trị hiện tại 6. Admin chỉnh sửa các thông số (ngưỡng chất, v.v.) 7. Validate các giá trị 8. Xác nhận và gửi request cập nhật 9. Validate và cập nhật cấu hình thành công

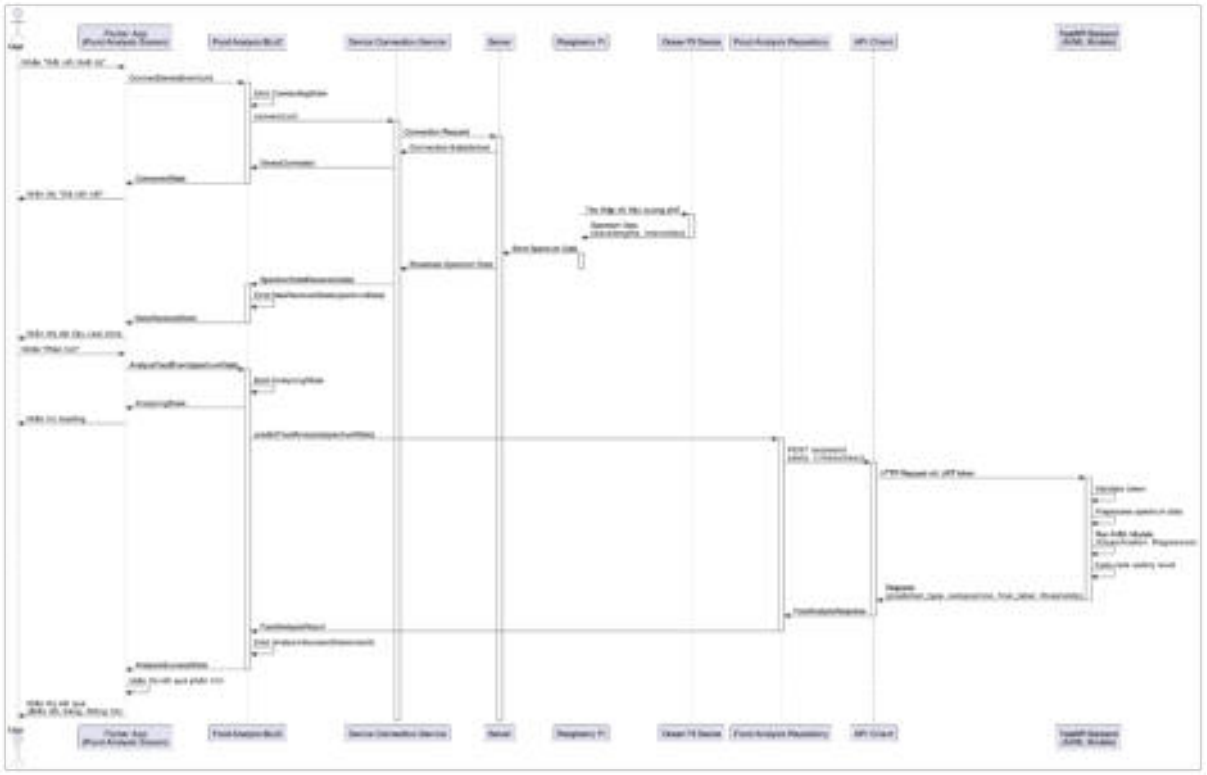
Bảng 2.6 Use Case mô tả cấu hình hệ thống

2.1.3. Biểu đồ Sequence

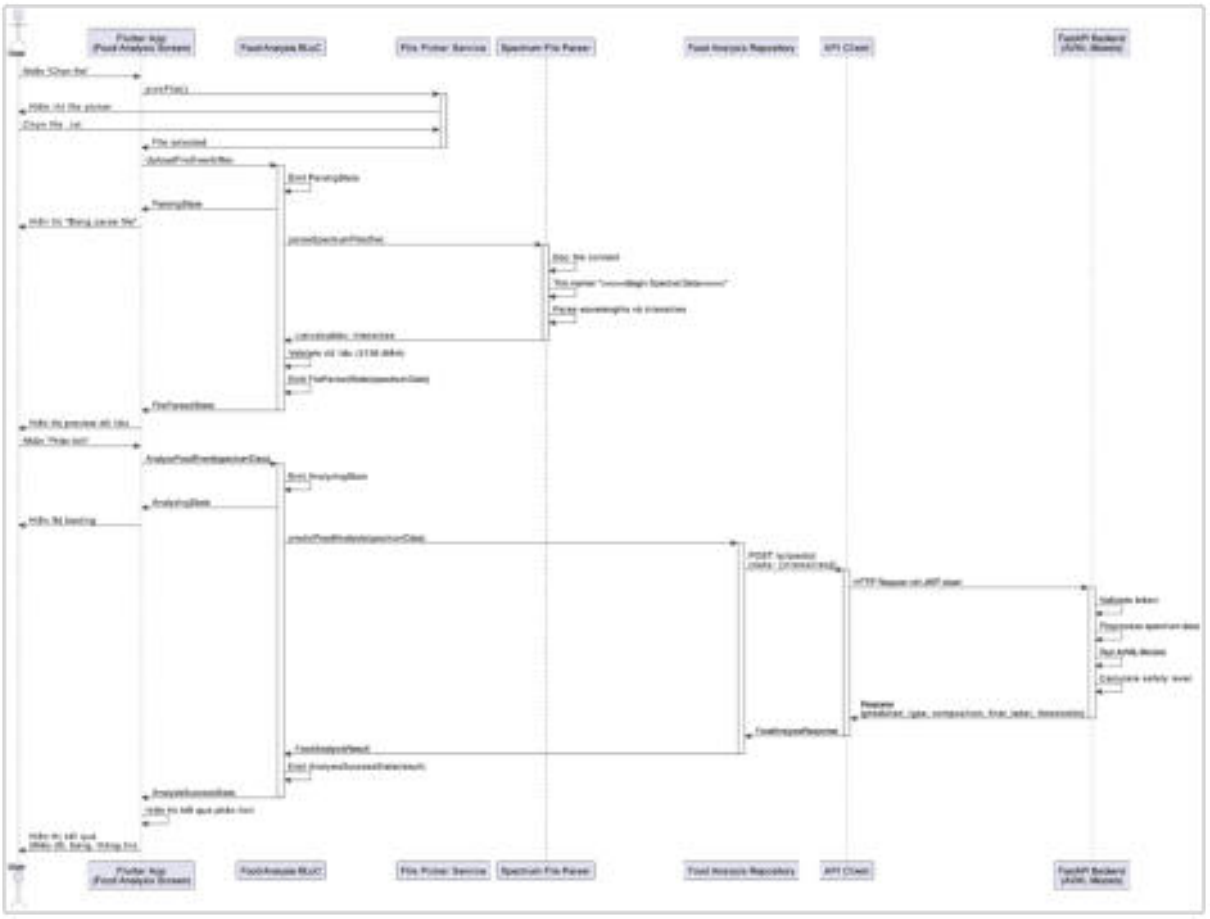
Biểu đồ Sequence mô tả luồng tương tác giữa các thành phần trong hệ thống theo thời gian. Dưới đây là các biểu đồ Sequence cho các luồng chính:



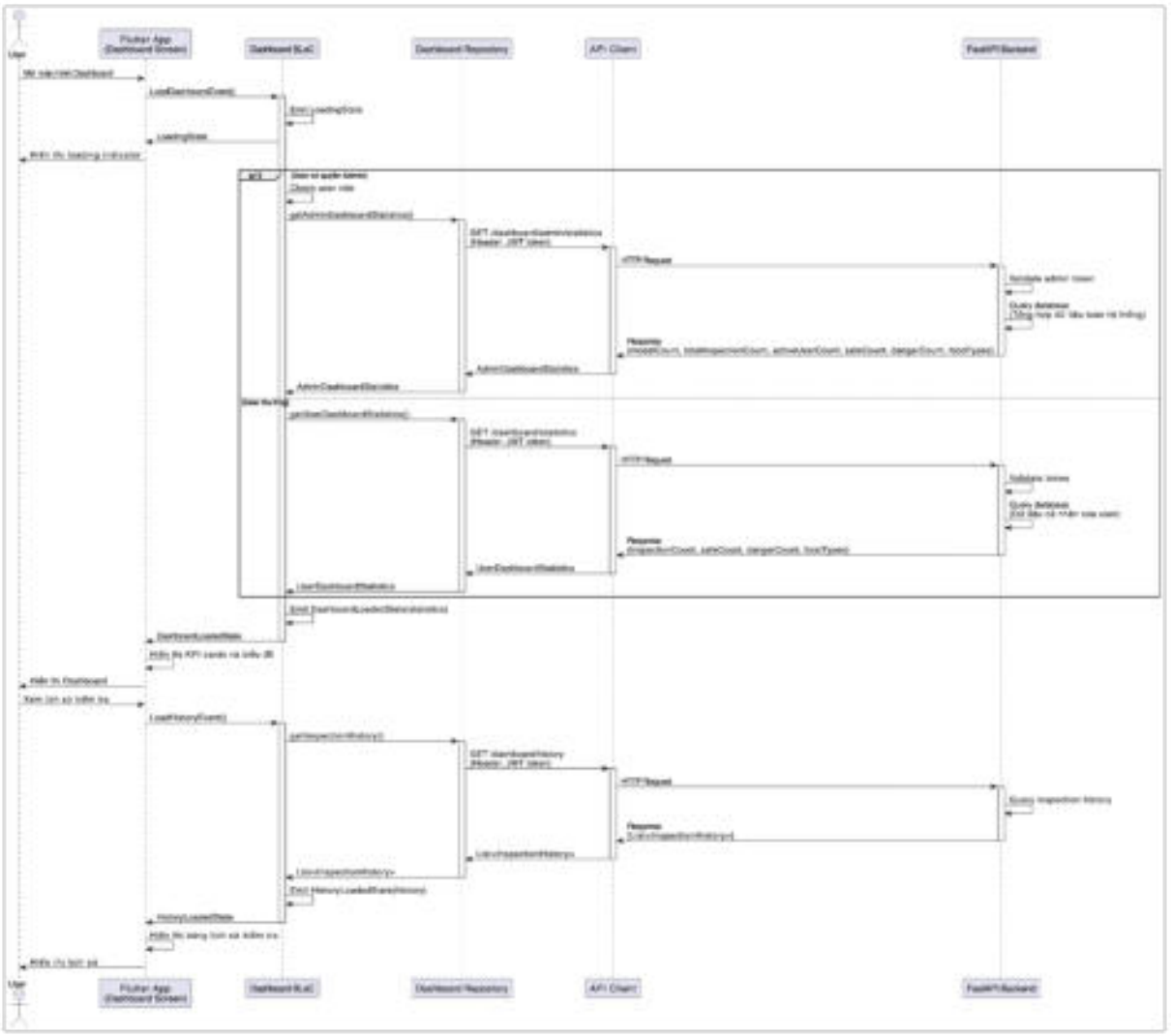
Hình 2.11: Sequence Diagram - Luồng đăng nhập



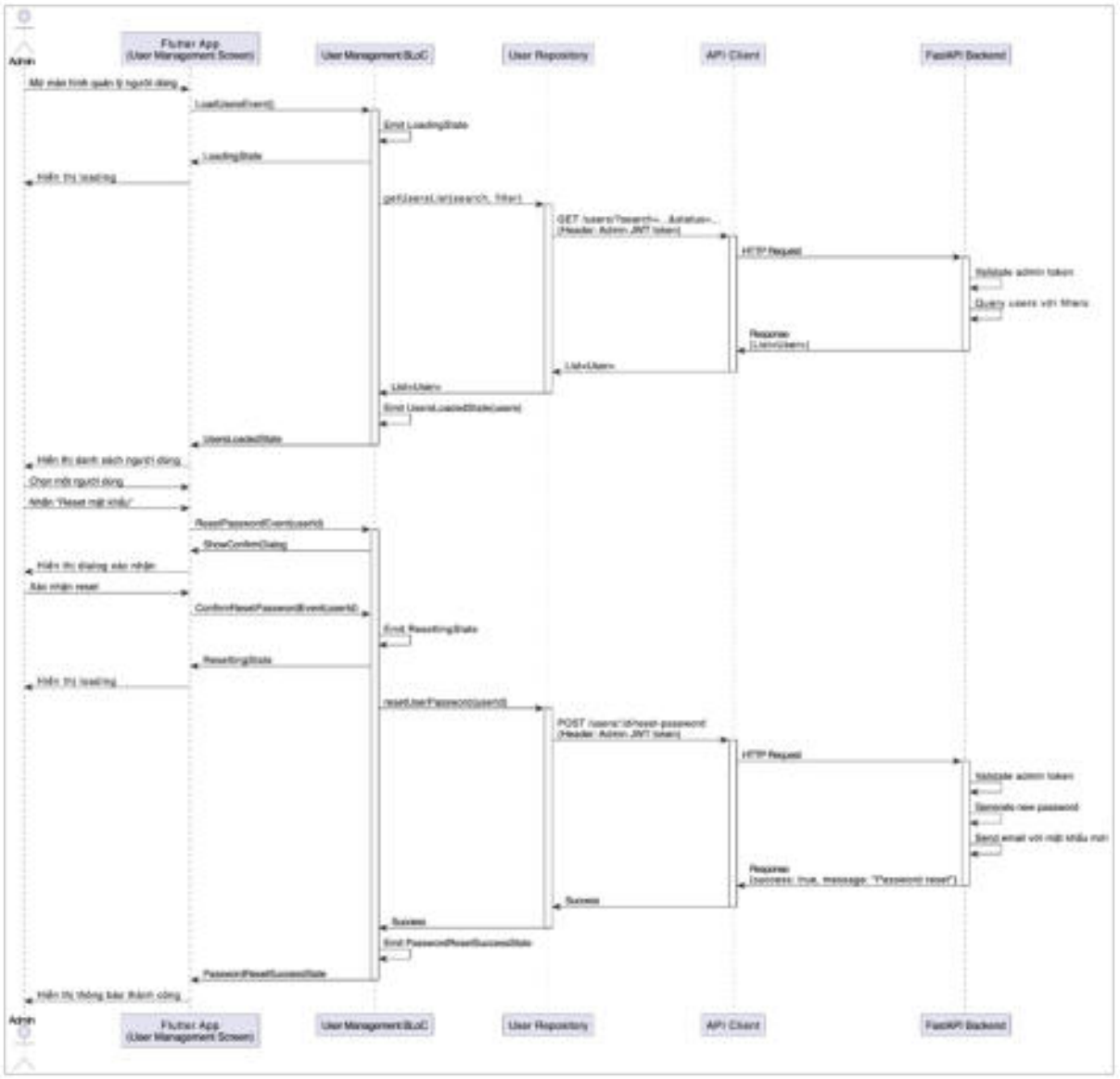
Hình 2.12: Sequence Diagram - Luồng phân tích thực phẩm (Nhận dữ liệu qua kết nối thiết bị)



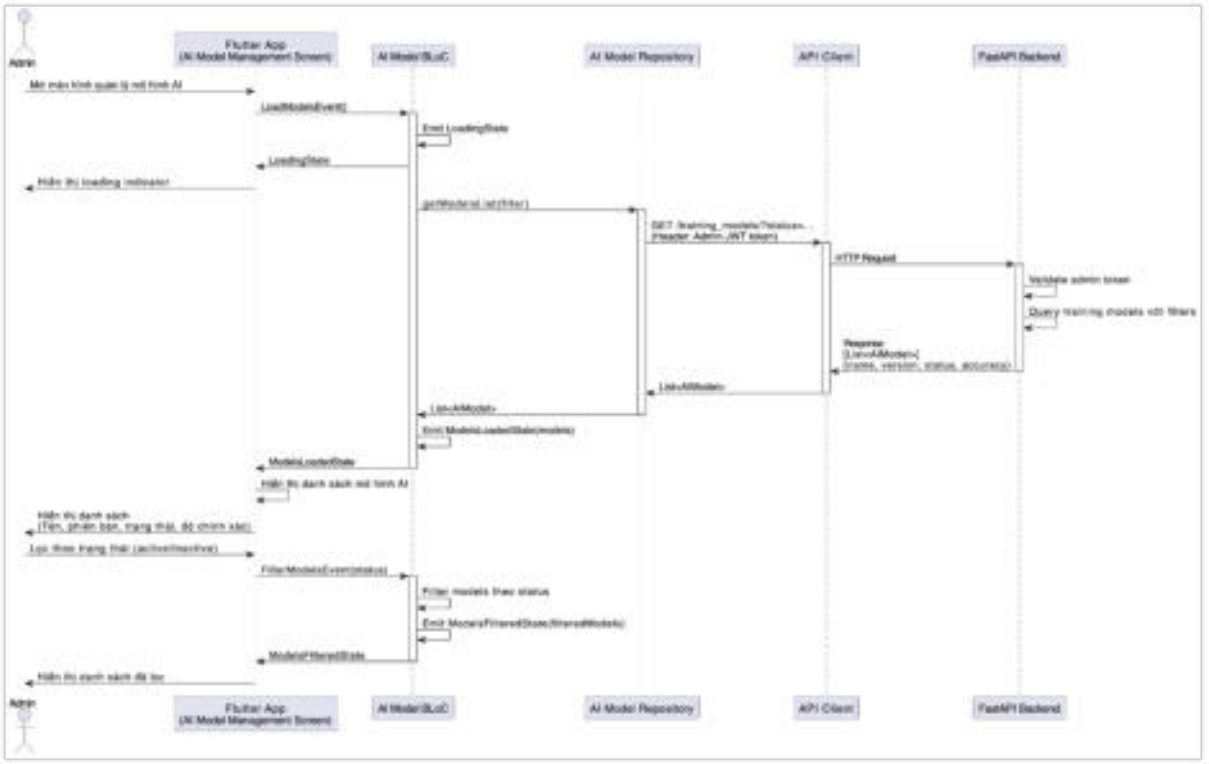
Hình 2.13: Sequence Diagram - Luồng phân tích thực phẩm (Nhận dữ liệu qua Upload File)



Hình 2.14: Sequence Diagram - Luồng xem Dashboard



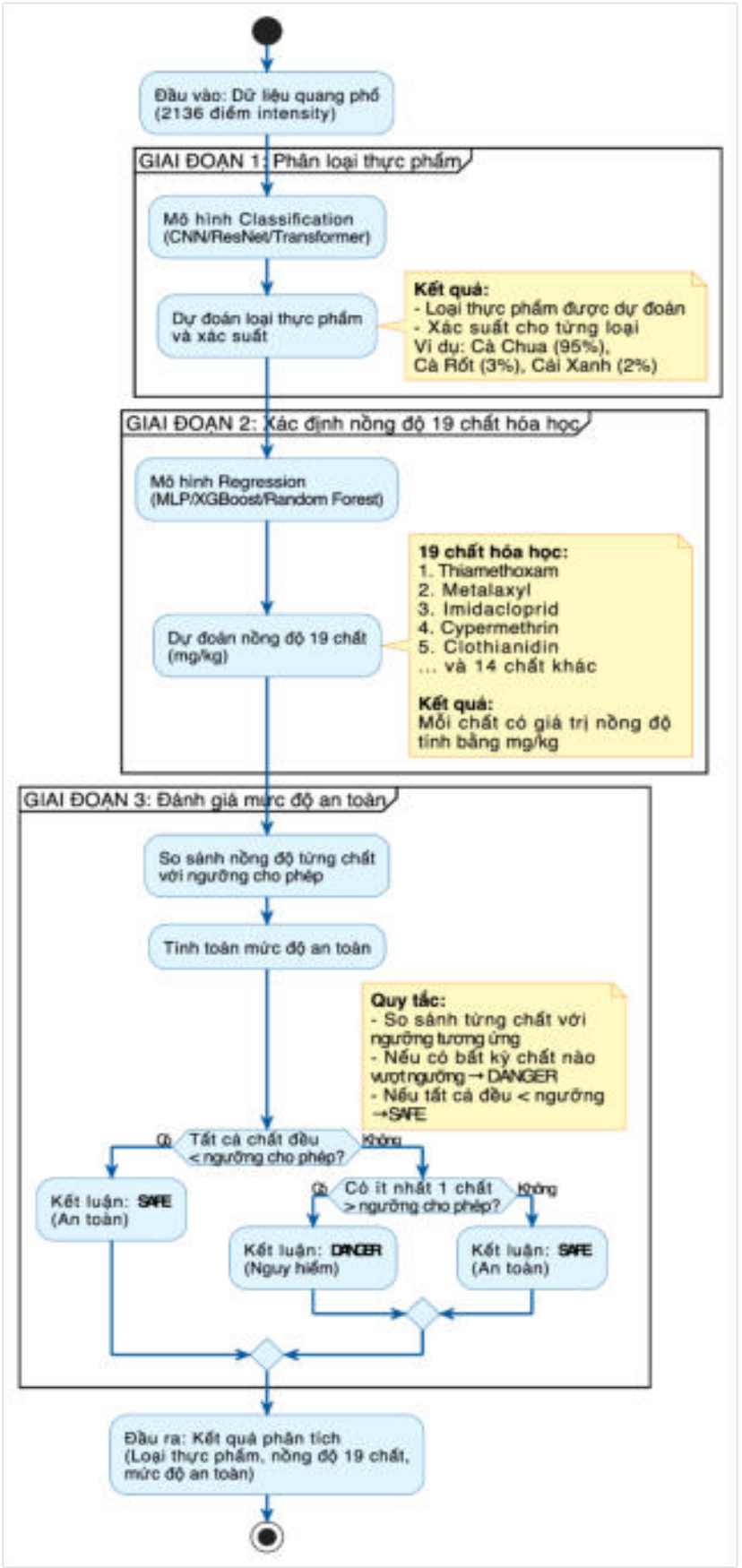
Hình 2.15: Sequence Diagram - Luồng quản lý người dùng (Admin)



Hình 2.16: Sequence Diagram - Luồng xem danh sách mô hình AI (Admin)

2.1.4. Quy trình phân tích thực phẩm bằng AI

Hệ thống sử dụng các mô hình AI/ML để phân tích dữ liệu quang phổ và đánh giá chất lượng thực phẩm thông qua 3 giai đoạn chính:



Hình 2.17: Sơ đồ quy trình phân tích thực phẩm bằng AI

Mô tả chi tiết các giai đoạn:

1. **Giai đoạn 1: Phân loại thực phẩm**
    a. **Mô hình sử dụng:** Classification Model (CNN, ResNet, hoặc Transformer)
    b. **Đầu vào:** Dữ liệu quang phổ gồm 2136 điểm intensity
    c. **Đầu ra:** Loại thực phẩm được dự đoán cùng với xác suất cho từng loại
    d. **Ví dụ:** Cà Chua (95%), Cà Rốt (3%), Cải Xanh (2%)
2. **Giai đoạn 2: Xác định nồng độ 19 chất hóa học**
    a. **Mô hình sử dụng:** Regression Model (MLP, XGBoost, hoặc Random Forest)
    b. **Đầu vào:** Dữ liệu quang phổ và loại thực phẩm đã được phân loại
    c. **Đầu ra:** Nồng độ của 19 chất hóa học tính bằng mg/kg
    d. **19 chất hóa học:** Thiamethoxam, Metalaxyl, Imidacloprid, Cypermethrin, Clothianidin và 14 chất khác
3. **Giai đoạn 3: Đánh giá mức độ an toàn**
    a. **Quy trình:** So sánh nồng độ từng chất với ngưỡng cho phép tương ứng
    b. **Quy tắc đánh giá:**
        i. Nếu tất cả các chất đều nhỏ hơn ngưỡng cho phép → **SAFE** (An toàn)
        ii. Nếu có ít nhất một chất vượt quá ngưỡng cho phép → **DANGER** (Nguy hiểm)
    c. **Đầu ra:** Mức độ an toàn cuối cùng (SAFE hoặc DANGER)

Kết quả cuối cùng: Hệ thống trả về thông tin đầy đủ bao gồm loại thực phẩm, nồng độ 19 chất hóa học, và mức độ an toàn để hiển thị cho người dùng.

2.1.2. Yêu cầu phi chức năng

Hiệu năng:

- Thời gian nhận dữ liệu quang phổ: < 3 giây
- Thời gian phân tích dữ liệu: < 5 giây (tùy thuộc vào API)
- Ứng dụng khởi động: < 3 giây
- Hỗ trợ file dữ liệu quang phổ lên đến 10MB

Bảo mật:

- Lưu trữ token an toàn bằng Secure Storage
- HTTPS cho tất cả API calls
- Refresh token tự động khi access token hết hạn
- Xác thực người dùng khi kết nối thiết bị (nếu hệ thống yêu cầu)

Khả năng sử dụng:

- Giao diện thân thiện, dễ sử dụng
- Hỗ trợ đa ngôn ngữ (Việt, Anh)
- Responsive design cho nhiều kích thước màn hình
- Thông báo lỗi rõ ràng, dễ hiểu

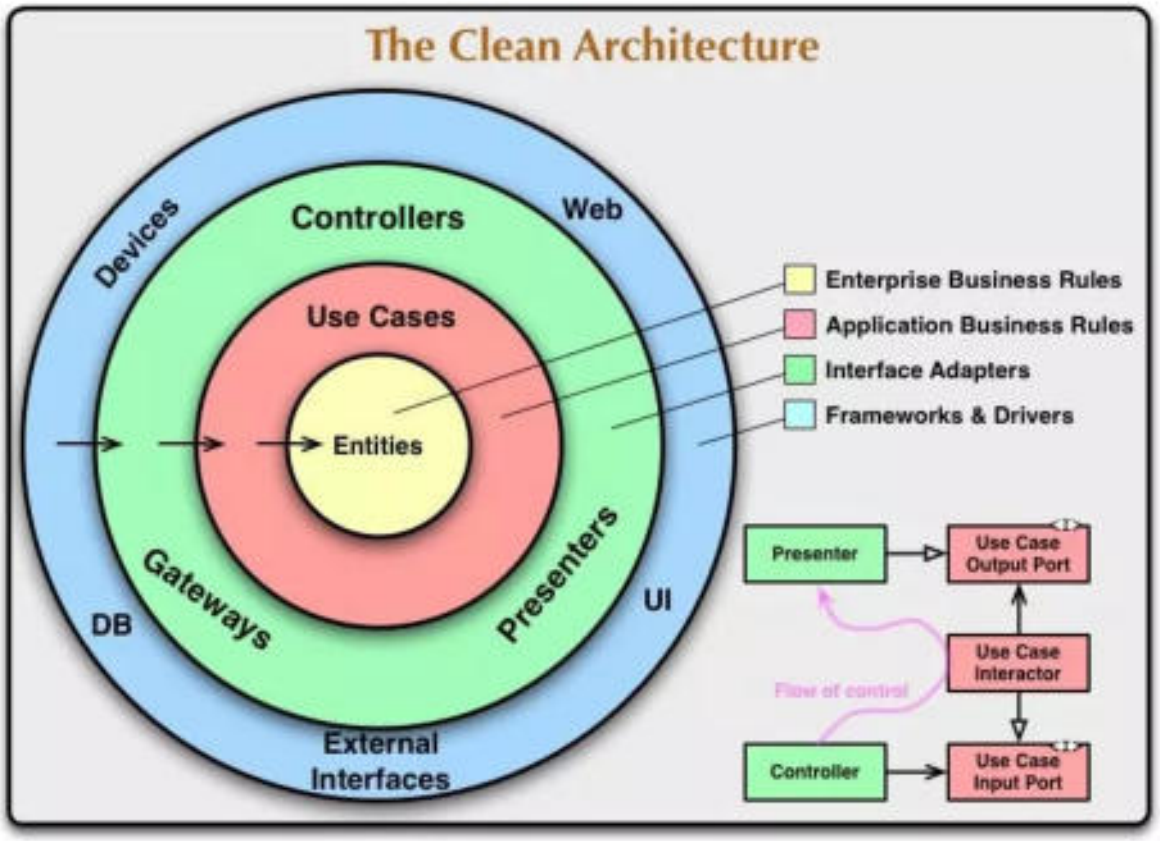
Khả năng mở rộng:

- Kiến trúc Clean Architecture dễ thêm tính năng mới
- Code modular, dễ bảo trì
- Hỗ trợ nhiều môi trường (development, staging, production)

2.2. Thiết kế kiến trúc ứng dụng

2.2.1. Kiến trúc tổng thể

Ứng dụng được thiết kế theo kiến trúc Clean Architecture:



Hình 2.1: Sơ đồ kiến trúc Clean Architecture

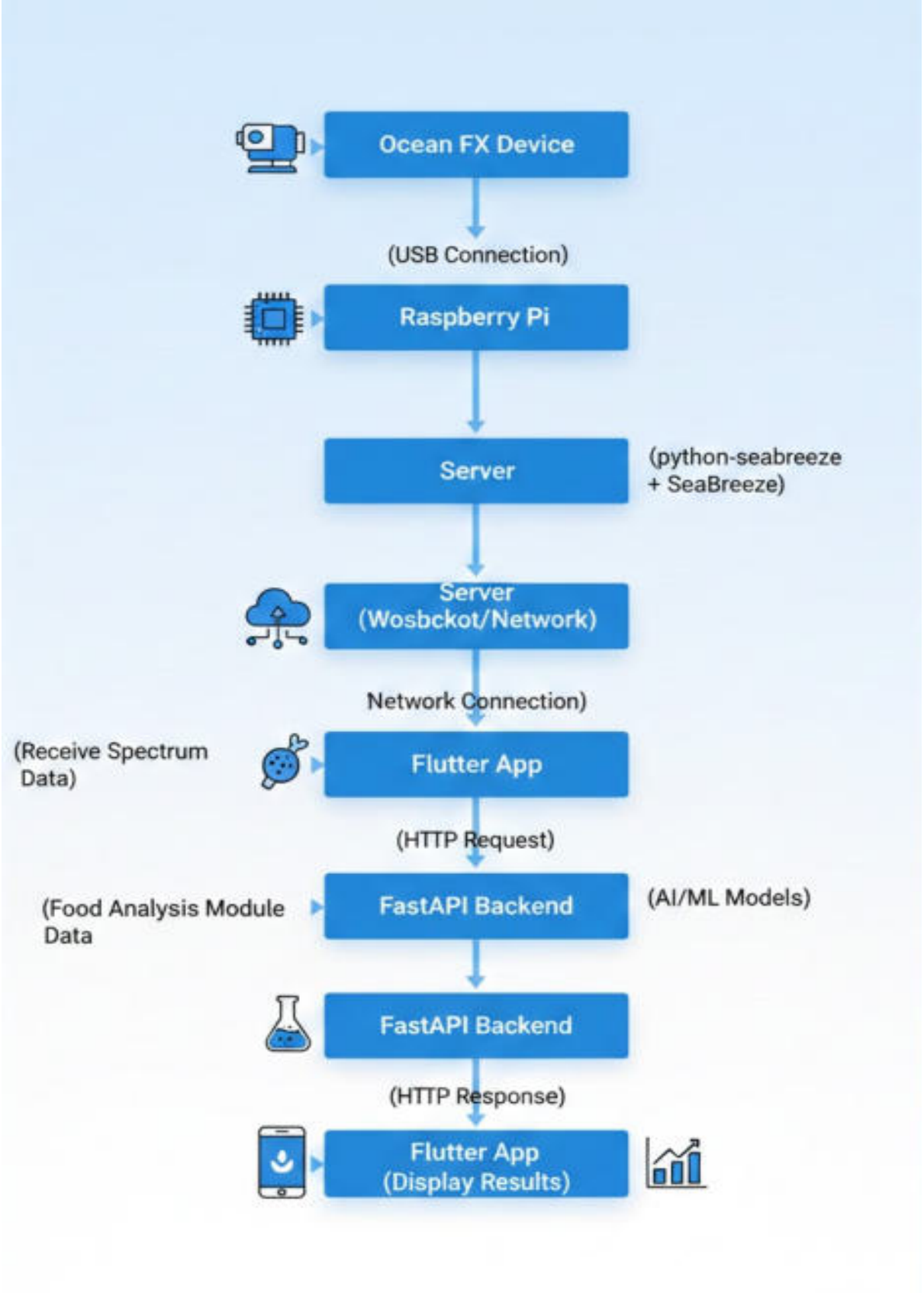
2.2.2. Cấu trúc thư mục dự án

```
lib/
├── core/                    # Core functionality
│   ├── base/                # Base classes (BLoC, State, Event)
│   ├── config/              # App configuration
│   ├── extensions/          # Extension methods
│   ├── injection/           # Dependency injection
│   ├── network/             # Network layer (API clients, interceptors)
│   ├── router/              # Navigation routing
│   ├── services/            # Core services (Device Connection, Storage)
│   ├── shared/              # Shared components, styles
│   └── utils/               # Utility functions
│
├── features/                # Feature modules
│   ├── food_analysis/       # Food analysis feature
│   │   ├── data/            # Data layer
│   │   │   ├── datasources/     # API data sources
│   │   │   ├── models/          # Data models
│   │   │   ├── repositories/    # Repository implementations
│   │   │   └── services/        # Data services
│   │   ├── domain/          # Domain layer
│   │   │   ├── entities/        # Business entities
│   │   │   ├── repositories/    # Repository interfaces
│   │   │   └── usecases/        # Use cases
│   │   └── presentation/        # Presentation layer
```

```
│   │       └── pages/            # UI pages
│   │
│   ├── home_page/                # Home/Dashboard feature
│   ├── login/                    # Authentication feature
│   ├── account_settings/         # User settings feature
│   └── ...
│
└── main.dart                     # App entry point
```

Bảng 2.7: Cấu trúc thư mục dự án

2.2.3. Luồng dữ liệu từ thiết bị đến ứng dụng



Hình 2.2: Sơ đồ luồng dữ liệu

Các bước chi tiết:

1. **Thu thập dữ liệu từ Ocean FX:**
    a. Raspberry Pi kết nối với Ocean FX qua USB
    b. Python application sử dụng python-seabreeze để điều khiển thiết bị
    c. Thu thập dữ liệu quang phổ (wavelength và intensity)
    d. Format dữ liệu và truyền lên server
2. **Truyền dữ liệu từ server đến ứng dụng:**
    a. Raspberry Pi gửi dữ liệu quang phổ lên server
    b. Server truyền dữ liệu đến ứng dụng Flutter
    c. Flutter app nhận dữ liệu quang phổ
3. **Xử lý dữ liệu trong Flutter App:**
    a. Nhận dữ liệu quang phổ từ server hoặc từ file upload
    b. Parse và validate dữ liệu (kiểm tra format, số lượng điểm dữ liệu)
    c. Lưu trữ dữ liệu tạm thời hoặc hiển thị trực tiếp
    d. Hoặc người dùng có thể chọn file dữ liệu quang phổ (.txt) từ thiết bị
4. **Gửi lên API để phân tích:**
    a. Tạo request với dữ liệu quang phổ (mảng các giá trị intensity)
    b. Gửi POST request đến endpoint `/ai/predict`
    c. Xử lý authentication (JWT token)
5. **Nhận và xử lý kết quả:**
    a. Nhận response từ API chứa:
        i. `predicted_type`: Loại thực phẩm
        ii. `predicted_type_probabilities`: Xác suất các loại
        iii. `predicted_composition`: Nồng độ các chất hóa học
        iv. `final_label`: Kết luận (Safe/Danger)
        v. `thresholds`: Ngưỡng cho phép các chất
        vi. `food_information`: Thông tin về thực phẩm

b. Map response thành domain entities

c. Tính toán mức độ an toàn (%)

d. Hiển thị kết quả trên UI

2.3. Thiết kế các module chính

2.3.1. Module phân tích thực phẩm

Module phân tích thực phẩm là module chính của ứng dụng, bao gồm các chức năng: nhận dữ liệu quang phổ (từ file upload hoặc kết nối thiết bị), parse và validate dữ liệu, gửi lên API để phân tích, và hiển thị kết quả.

Chức năng:

1. **Nhận dữ liệu quang phổ:**
    a. Upload file dữ liệu quang phổ (.txt) từ thiết bị
    b. Hoặc kết nối với thiết bị đo quang phổ để nhận dữ liệu real-time
    c. Xử lý reconnection tự động khi mất kết nối (nếu kết nối thiết bị)
    d. Quản lý và hiển thị trạng thái kết nối

- Xử lý lỗi và timeout

1. **Xử lý dữ liệu:**
    a. Parse file dữ liệu quang phổ
    b. Validate dữ liệu quang phổ
2. **Phân tích và hiển thị kết quả:**
    a. Gửi dữ liệu lên API để phân tích
    b. Xử lý và hiển thị kết quả phân tích

Các class chính:

```
// Service để nhận dữ liệu quang phổ từ thiết bị
class DeviceConnectionService {
```

```
  // Kết nối với thiết bị đo quang phổ
  Future<void> connect(String deviceUrl);

  // Ngắt kết nối
  Future<void> disconnect();

  // Lắng nghe dữ liệu quang phổ
  Stream<SpectrumData> get spectrumDataStream;

  // Kiểm tra trạng thái kết nối
  bool get isConnected;
}

// Domain Layer
class FoodAnalysisResult {
  final String id;
  final String predictedType;
  final Map<String, double> predictedTypeProbabilities;
  final Map<String, double> predictedComposition;
  final String finalLabel;
  final Map<String, double> thresholds;
  final String? foodInformation;

  bool get isSafe => finalLabel.toLowerCase() == 'safe';
}
```

```
// Data Layer
class SpectrumFileParserService {
  Future<List<double>> parseSpectrumFile(File file);
}

class FoodAnalysisRepository {
  Future<FoodAnalysisResult> predictFoodAnalysis(List<double> spectrumData);
}
```

Luồng xử lý phân tích:

1. **Nhận dữ liệu:**
    a. Từ kết nối thiết bị: Nhận dữ liệu real-time qua DeviceConnectionService
    b. Hoặc từ file: Người dùng chọn file .txt và SpectrumFileParserService parse file để lấy mảng intensity
2. **Xử lý và phân tích:**
    a. Validate dữ liệu quang phổ
    b. FoodAnalysisRepository tạo request và gửi lên API
    c. Map response thành FoodAnalysisResult
3. **Hiển thị kết quả:**
    a. Tính toán mức độ an toàn dựa trên nồng độ và ngưỡng
    b. Hiển thị kết quả trên UI với biểu đồ và bảng thống kê

2.3.2. Module Dashboard

Dashboard cá nhân (dành cho User và Admin):

Chức năng:

- Hiển thị thống kê tổng quan của người dùng hiện tại (User và Admin đều có thể xem thống kê cá nhân)
- Biểu đồ phân tích dựa trên lịch sử kiểm tra cá nhân

- Lịch sử kiểm tra của bản thân

Các component:

- KpiCard: Hiển thị các chỉ số KPI (số lượt kiểm tra của bản thân)
- SafetyLevelChart: Biểu đồ phân bố mức độ an toàn (dựa trên lịch sử cá nhân)
- FoodSafetyChart: Biểu đồ phân tích theo loại thực phẩm đã kiểm tra
- InspectionHistoryTable: Bảng lịch sử kiểm tra của bản thân

Dashboard tổng quan hệ thống (chỉ dành cho Admin):

Chức năng:

- Hiển thị thống kê tổng quan của toàn hệ thống
- Biểu đồ phân tích dựa trên dữ liệu của tất cả người dùng
- Lịch sử kiểm tra của tất cả người dùng
- Thống kê số lượng người dùng hoạt động

Các component:

- KpiCard: Hiển thị các chỉ số KPI (số mô hình AI, tổng số lượt kiểm tra, số người dùng)
- SafetyLevelChart: Biểu đồ phân bố mức độ an toàn tổng thể
- FoodSafetyChart: Biểu đồ phân tích theo loại thực phẩm tổng thể
- InspectionHistoryTable: Bảng lịch sử kiểm tra của tất cả người dùng
- UserActivityChart: Biểu đồ thống kê số lượt kiểm tra theo thời gian

Dữ liệu:

```
// Dashboard statistics cá nhân (cho User và Admin)
class UserDashboardStatistics {
```

```
  final int inspectionCount;  // Số lượt kiểm tra của bản thân
  final int safeCount;
  final int dangerCount;
  final List<FoodTypeStatistics> foodTypes;  // Dựa trên lịch sử cá nhân
}

// Dashboard statistics tổng quan hệ thống (chỉ cho Admin)
class AdminDashboardStatistics {
  final int modelCount;  // Số mô hình AI
  final int totalInspectionCount;  // Tổng số lượt kiểm tra
  final int activeUserCount;  // Số người dùng hoạt động
  final int safeCount;
  final int dangerCount;
  final List<FoodTypeStatistics> foodTypes;  // Tổng thể
}

class InspectionHistory {
  final String id;
  final String foodName;
  final double safetyLevel;
  final String conclusion;
  final DateTime createdAt;
  final String? userId;  // Để Admin có thể xem ai đã kiểm tra (chỉ có trong dashboard tổng quan)
  final String? userName;  // Tên người dùng (cho Admin, chỉ có trong dashboard tổng quan)
}
```

2.4. Thiết kế giao diện người dùng

2.4.1. Các màn hình chính

Màn hình chung (dùng cho cả User và Admin):

1. **Màn hình Splash**
    a. Hiển thị logo ứng dụng
    b. Kiểm tra trạng thái đăng nhập và quyền người dùng
    c. Điều hướng đến màn hình phù hợp theo vai trò
2. **Màn hình Đăng nhập**
    a. Form đăng nhập (email, password)
    b. Link quên mật khẩu
    c. Link đăng ký (chỉ dành cho User)

Màn hình dành cho User:

1. **Màn hình Dashboard (Home) - User**
    a. KPI cards (số lượt kiểm tra của bản thân)
    b. Biểu đồ mức độ an toàn (dựa trên lịch sử kiểm tra cá nhân)
    c. Biểu đồ phân tích theo loại thực phẩm đã kiểm tra
    d. Bảng lịch sử kiểm tra của bản thân
    e. Nút "Kiểm tra thực phẩm"
2. **Màn hình Kết quả phân tích**
    a. Tên thực phẩm
    b. Mức độ an toàn (%)
    c. Kết luận (An toàn/Nguy hiểm)
    d. Biểu đồ quang phổ
    e. Bảng nồng độ các chất hóa học
    f. Thông tin về thực phẩm

3. **Màn hình Cài đặt tài khoản**
    a. Thông tin cá nhân
    b. Đổi mật khẩu
    c. Cài đặt ngôn ngữ
    d. Đăng xuất

Màn hình dành cho Admin *(Admin có đầy đủ các màn hình như User, cộng thêm các màn hình quản lý hệ thống):*

1. **Màn hình Dashboard cá nhân - Admin**
    a. KPI cards (số lượt kiểm tra của bản thân)
    b. Biểu đồ mức độ an toàn (dựa trên lịch sử kiểm tra cá nhân)
    c. Biểu đồ phân tích theo loại thực phẩm đã kiểm tra cá nhân
    d. Bảng lịch sử kiểm tra của bản thân
    e. Nút "Kiểm tra thực phẩm"
    f. Tab/Menu chuyển sang Dashboard tổng quan hệ thống
2. **Màn hình Kết quả phân tích**
    a. Tên thực phẩm
    b. Mức độ an toàn (%)
    c. Kết luận (An toàn/Nguy hiểm)
    d. Biểu đồ quang phổ
    e. Bảng nồng độ các chất hóa học
    f. Thông tin về thực phẩm
3. **Màn hình Dashboard tổng quan hệ thống - Admin** *(Chỉ dành cho Admin)*
    a. KPI cards (số mô hình AI, tổng số lượt kiểm tra, số người dùng)
    b. Biểu đồ mức độ an toàn tổng thể (tất cả người dùng)
    c. Biểu đồ phân tích theo loại thực phẩm của toàn hệ thống
    d. Thống kê số lượt kiểm tra theo thời gian
    e. Bảng lịch sử kiểm tra của tất cả người dùng

f. Menu điều hướng đến các chức năng quản lý

4. **Màn hình Quản lý người dùng** *(Chỉ dành cho Admin)*
    a. Danh sách tất cả người dùng với tìm kiếm và lọc
    b. Chi tiết thông tin người dùng (hồ sơ, lịch sử hoạt động)
    c. Nút reset mật khẩu
    d. Nút khóa/mở khóa tài khoản
5. **Màn hình Quản lý mô hình AI** *(Chỉ dành cho Admin)*
    a. Danh sách tất cả mô hình AI với thông tin cơ bản (tên, phiên bản, trạng thái active/inactive, độ chính xác)
    b. Hỗ trợ lọc theo trạng thái (active/inactive)
6. **Màn hình Cài đặt tài khoản Admin**
    a. Thông tin cá nhân
    b. Đổi mật khẩu
    c. Cài đặt ngôn ngữ
    d. Đăng xuất

2.4.2. Design System

Màu sắc:

- Primary: Blue (kết nối, hành động chính)
- Success: Green (an toàn, thành công)
- Danger: Red (nguy hiểm, cảnh báo)
- Warning: Orange (cảnh báo)
- Background: Grey[100] (nền)

Typography:

- Font: NotoSans (Regular, Medium, SemiBold, Bold)
- Sizes: 10, 12, 13, 14, 16, 18, 20, 24

Spacing:

- Sử dụng flutter_screenutil để responsive
- Padding/Margin: 8, 12, 16, 24, 32

Components:

- DefaultButtonWidget: Button chuẩn với các variants (filled, outlined)
- AppTextField: Text field với validation
- KpiCard: Card hiển thị KPI
- SafetyLevelChart: Biểu đồ mức độ an toàn
- FoodSafetyChart: Biểu đồ phân tích thực phẩm

2.5. Thiết kế tích hợp với API backend

2.5.1. API Endpoints

Authentication (User và Admin):

- POST /auth/token: Đăng nhập, nhận access token và refresh token
- POST /auth/refresh: Refresh access token
- GET /auth/profile: Lấy thông tin người dùng hiện tại

Food Analysis (User và Admin):

- POST /ai/predict: Phân tích dữ liệu quang phổ (yêu cầu authentication)

Dashboard:

Dành cho User và Admin (Dashboard cá nhân):

- GET /dashboard/statistics: Lấy thống kê tổng quan của người dùng hiện tại (User và Admin đều có thể truy cập để xem thống kê cá nhân)
- GET /dashboard/history: Lấy lịch sử kiểm tra của người dùng hiện tại (User và Admin đều có thể truy cập để xem lịch sử cá nhân)

Chỉ dành cho Admin (Dashboard tổng quan hệ thống):

User Management (Chỉ dành cho Admin):

- GET /users/: Danh sách tất cả người dùng (yêu cầu quyền admin)
- GET /users/:id: Chi tiết người dùng (yêu cầu quyền admin)
- POST /users/:id/reset-password: Reset mật khẩu cho người dùng (yêu cầu quyền admin)
- PUT /users/:id/status: Cập nhật trạng thái người dùng (khóa/mở khóa) (yêu cầu quyền admin)

Model Management (Chỉ dành cho Admin):

- GET /training_models/: Danh sách mô hình AI với thông tin cơ bản (tên, phiên bản, trạng thái, độ chính xác) (yêu cầu quyền admin)

2.5.2. Request/Response Format

Food Analysis Request:

```
{
  "data": [2720.9, 2716.27, 2705.87, ...]
}
```

Food Analysis Response:

```
{
  "id": "68efe45aee19d5ace812d6cd",
  "predicted_type": "Cà Chua",
  "predicted_type_probabilities": {
    "Cà Chua": 0.95,
    "Cà Rốt": 0.03,
    "Cải Xanh": 0.02
```

```
  },
  "predictd_composition": {
    "Thiamethoxam": 0.24,
    "Metalaxyl": 0.08,
    "Imidaclopird": 0.02,
    "Cypermethrin": 0.02,
    "Chlothianidin": 0.01
  },
  "final_label": "Safe",
  "thresholds": {
    "Thiamethoxam": 0.5,
    "Metalaxyl": 0.3,
    "Imidaclopird": 0.2,
    "Cypermethrin": 0.2,
    "Chlothianidin": 0.2
  },
  "food_information": "Cà chua là một loại quả mọng..."
}
```


2.5.3. Error Handling

Các loại lỗi:

- Network errors: Không có kết nối internet
- API errors: Lỗi từ server (400, 401, 403, 404, 500)
- Validation errors: Dữ liệu không hợp lệ
- Connection errors: Lỗi kết nối thiết bị, mất kết nối
- Data errors: Dữ liệu quang phổ không hợp lệ

Xử lý lỗi:

- Hiển thị thông báo lỗi rõ ràng cho người dùng
- Retry mechanism cho network errors
- Auto reconnect khi mất kết nối thiết bị
- Auto refresh token khi 401
- Log errors để debug

Chương 3: TRIỂN KHAI VÀ KẾT QUẢ

3.1. Môi trường phát triển và công cụ

Công cụ/Công nghệ	Phiên bản/Mô tả chi tiết	Vai trò
IDE/Editor	Visual Studio Code (VS Code)	Môi trường phát triển tích hợp (IDE) chính cho Flutter và Dart.
Framework Frontend	Flutter 3.35.6	Framework đa nền tảng để xây dựng ứng dụng Mobile (iOS và Android).
Ngôn ngữ Backend	Python 3.9	Ngôn ngữ được sử dụng để phát triển ứng dụng trên Raspberry Pi (thu thập dữ liệu) và API Backend (phân tích AI/ML).
Quản lý phiên bản	Git	Hệ thống quản lý mã nguồn, được sử dụng để theo dõi các thay đổi và hợp tác trong quá trình phát triển.
Kiểm thử API	Postman	Công cụ được sử dụng để kiểm thử các API endpoint của hệ thống Backend (FastAPI) trong quá trình phát triển và tích hợp.

3.1.2. Môi trường phần cứng và triển khai

Hạng mục	Chi tiết	Vai trò
Thiết bị trung gian	Raspberry Pi 3+	Máy tính nhúng làm trung gian, kết nối với thiết bị Ocean FX và chạy ứng dụng Python để truyền dữ liệu quang phổ lên server qua WebSocket.
Nền tảng kiểm thử	TestFlight (iOS)	Ứng dụng đã được xây dựng và triển khai lên nền tảng TestFlight để thực hiện kiểm thử trên các thiết bị iOS thực tế.

3.2. Triển khai module phân tích thực phẩm

Module phân tích thực phẩm là module chính của ứng dụng, bao gồm các chức năng: nhận dữ liệu quang phổ (từ file upload hoặc kết nối thiết bị), parse và validate dữ liệu, gửi lên API để phân tích, và hiển thị kết quả.

3.2.1. Parse file dữ liệu quang phổ

Triển khai:

- Tạo service SpectrumFileParserService để xử lý việc parse file dữ liệu quang phổ
- Service đọc file .txt theo định dạng Ocean FX, tìm marker ">>>>>Begin Spectral Data<<<<<" để xác định vị trí bắt đầu dữ liệu
- Parse các dòng dữ liệu dạng tab-separated (wavelength và intensity), trích xuất mảng các giá trị intensity
- Validate dữ liệu để đảm bảo có đủ 2136 điểm dữ liệu như yêu cầu

Xử lý lỗi:

- Kiểm tra định dạng file hợp lệ
- Xử lý trường hợp file không đúng format
- Thông báo lỗi rõ ràng cho người dùng nếu file không hợp lệ

3.2.2. Tích hợp với API phân tích

Kiến trúc triển khai theo Clean Architecture:

1. **Data Layer:**
    - Tạo FoodAnalysisDataSource interface và implementation
    - Sử dụng AuthApiClient để gửi POST request đến endpoint /ai/predict
    - Request body chứa mảng intensity dữ liệu quang phổ
    - Map response JSON thành FoodAnalysisResponse model
2. **Domain Layer:**
    - Tạo FoodAnalysisRepository interface định nghĩa các phương thức cần thiết
    - Tạo FoodAnalysisResult entity chứa kết quả phân tích
    - Tạo PredictFoodAnalysisUseCase để thực hiện logic nghiệp vụ

3. **Presentation Layer:**
    - Tạo FoodAnalysisBloc để quản lý state của màn hình phân tích
    - Xử lý các events: upload file, kết nối thiết bị, phân tích, xem kết quả
    - Emit các states: loading, success, error

Quy trình xử lý:

- Người dùng chọn file hoặc kết nối thiết bị → Parse dữ liệu → Validate → Gửi lên API → Nhận kết quả → Hiển thị

3.2.3. Xử lý và hiển thị kết quả

Chuyển đổi dữ liệu:

- Map FoodAnalysisResponse từ API thành FoodAnalysisResult entity
- Tính toán mức độ an toàn (%) dựa trên việc so sánh nồng độ các chất với ngưỡng cho phép
- Tạo danh sách ChemicalConcentration chứa thông tin từng chất: tên, nồng độ, ngưỡng

Hiển thị kết quả:

- Hiển thị loại thực phẩm được dự đoán với xác suất
- Hiển thị mức độ an toàn dưới dạng phần trăm và kết luận (Safe/Danger)
- Hiển thị bảng nồng độ 19 chất hóa học với so sánh ngưỡng
- Hiển thị biểu đồ quang phổ (nếu có)
- Hiển thị thông tin về thực phẩm

3.3. Triển khai module Dashboard

3.3.1. Dashboard Screen

Cấu trúc màn hình:

- Dashboard được chia thành các section: KPI Cards, Biểu đồ, Bảng lịch sử
- Sử dụng DashboardBloc để quản lý state và fetch dữ liệu từ API
- Hỗ trợ pull-to-refresh để cập nhật dữ liệu

Các component chính:

1. **KPI Cards:**

- Hiển thị các chỉ số quan trọng: số lượt kiểm tra, số mô hình AI (Admin)
- Mỗi card có icon và màu sắc riêng để dễ phân biệt
- Responsive design với flutter_screenutil

2. **Safety Level Chart:**
   - Sử dụng thư viện fl_chart để vẽ biểu đồ Pie Chart
   - Hiển thị phân bố mức độ an toàn (Safe/Danger) dựa trên dữ liệu thống kê
   - Màu xanh cho Safe, màu đỏ cho Danger
3. **Food Safety Chart:**
   - Biểu đồ cột hoặc bar chart hiển thị số lượt phân tích theo từng loại thực phẩm
   - Giúp người dùng nắm được xu hướng kiểm tra các loại thực phẩm
4. **Inspection History Table:**
   - Bảng hiển thị lịch sử kiểm tra gần đây
   - Các cột: Tên thực phẩm, Mức độ an toàn, Kết luận, Ngày kiểm tra
   - Hỗ trợ click vào từng row để xem chi tiết
   - Hỗ trợ phân trang nếu có nhiều dữ liệu

3.3.2. Tích hợp API Dashboard

API Endpoints:

- GET /dashboard/statistics: Lấy thống kê cá nhân (User và Admin)
- GET /dashboard/history: Lấy lịch sử kiểm tra cá nhân
- GET /dashboard/admin/statistics: Lấy thống kê tổng quan hệ thống (chỉ Admin)

Xử lý dữ liệu:

- Map response từ API thành domain entities
- Xử lý trường hợp dữ liệu trống
- Cache dữ liệu để tối ưu hiệu năng

3.4. Triển khai các tính năng phụ trợ

3.4.1. Đa ngôn ngữ (i18n)

Triển khai:

- Sử dụng thư viện easy_localization để quản lý đa ngôn ngữ
- Hỗ trợ 3 ngôn ngữ: Tiếng Việt (mặc định), Tiếng Anh
- Các file translation được lưu trong assets/i18n/ với format JSON
- Khởi tạo EasyLocalization trong main.dart với fallback locale là tiếng Việt

Sử dụng:

- Sử dụng extension .tr() trên string để tự động dịch theo locale hiện tại
- Cho phép người dùng thay đổi ngôn ngữ trong Settings
- Lưu preference ngôn ngữ để giữ nguyên khi mở lại ứng dụng

3.4.2. Quản lý routing

Triển khai:

- Sử dụng go_router để quản lý navigation và routing
- Định nghĩa các routes chính: splash, login, home, food-analysis, dashboard, user-management, model-management, system-settings
- Hỗ trợ deep linking và navigation với parameters
- Xử lý authentication guard để bảo vệ các routes yêu cầu đăng nhập

Cấu trúc routes:

- Routes công khai: splash, login
- Routes yêu cầu authentication: home, dashboard, food-analysis
- Routes yêu cầu quyền Admin: user-management, model-management, system-settings

3.4.3. Dependency Injection

Triển khai:

- Sử dụng get_it làm service locator
- Sử dụng injectable và injectable_generator để tự động generate code đăng ký dependencies
- Định nghĩa các dependency types: Singleton, LazySingleton, Factory
- Khởi tạo dependencies trong configureDependencies() function

Các loại dependency:

- **Singleton**: Services, Repositories (một instance cho toàn bộ app)
- **LazySingleton**: Tạo instance khi cần thiết lần đầu
- **Factory**: Tạo instance mới mỗi lần được yêu cầu (Use Cases)

3.4.4. Xác thực và bảo mật

JWT Token Management:

- Lưu trữ access token và refresh token trong SecureStorage
- Tự động refresh token khi access token hết hạn thông qua RefreshTokenInterceptor
- Xử lý logout và clear tokens khi người dùng đăng xuất

API Interceptors:

- AuthInterceptor: Tự động thêm JWT token vào header của mọi request
- RefreshTokenInterceptor: Tự động refresh token khi nhận 401 response
- ErrorInterceptor: Xử lý và format lỗi từ API

3.5. Triển khai module quản lý người dùng (Admin)

3.5.1. Danh sách người dùng

Triển khai:

- Tạo UserManagementBloc để quản lý state của màn hình quản lý người dùng
- Fetch danh sách người dùng từ API /users/ với hỗ trợ pagination
- Hỗ trợ tìm kiếm và lọc theo email, trạng thái
- Hiển thị danh sách dưới dạng table hoặc list với thông tin cơ bản: email, tên, role, trạng thái

Chức năng:

- Xem chi tiết người dùng: Navigate đến màn hình chi tiết khi click vào một user
- Reset mật khẩu: Gửi request đến API /users/:id/reset-password, hiển thị dialog xác nhận
- Khóa/Mở khóa tài khoản: Gửi request đến API để cập nhật trạng thái user

3.5.2. Chi tiết người dùng

Triển khai:

- Fetch thông tin chi tiết từ API /users/:id
- Hiển thị đầy đủ thông tin: email, tên, role, ngày tạo, trạng thái, lịch sử hoạt động
- Các nút hành động: Reset mật khẩu, Khóa/Mở khóa tài khoản

3.6. Triển khai module quản lý mô hình AI (Admin)

Triển khai:

- Tạo ModelManagementBloc để quản lý state
- Fetch danh sách mô hình từ API /training_models/ với pagination
- Hiển thị thông tin cơ bản: tên mô hình, phiên bản, trạng thái (active/inactive), độ chính xác
- Hỗ trợ lọc theo trạng thái (active/inactive)
- Chỉ hiển thị danh sách, không có chức năng xem chi tiết hay cập nhật

3.7. Triển khai module cấu hình hệ thống (Admin)

Triển khai:

- Tạo SystemSettingsBloc để quản lý state
- Fetch cấu hình hiện tại từ API /settings/
- Hiển thị form cấu hình với các trường:
    - Mô hình classification đang sử dụng
    - Ngưỡng classification
    - Danh sách mô hình regression cho từng loại thực phẩm
    - Ngưỡng an toàn cho 19 chất hóa học
- Hỗ trợ cập nhật cấu hình: Validate dữ liệu, gửi request đến API, hiển thị thông báo thành công/lỗi

3.8. Kết quả và đánh giá

3.8.1. Kết quả đạt được

1. Hoàn thành các module chính:

- Module phân tích thực phẩm: Parse file quang phổ, tích hợp API, hiển thị kết quả

- Module Dashboard: Hiển thị thống kê, biểu đồ, lịch sử kiểm tra
- Module quản lý người dùng: Đăng nhập, đăng ký, quản lý tài khoản
- Module quản lý mô hình AI (Admin): Xem danh sách mô hình với thông tin cơ bản

2. Tính năng đã triển khai:

- Đa ngôn ngữ (Tiếng Việt, Tiếng Anh)
- Xác thực người dùng với JWT token
- Refresh token tự động
- Responsive UI với flutter_screenutil
- Xử lý lỗi và loading states

3. Kiến trúc và code quality:

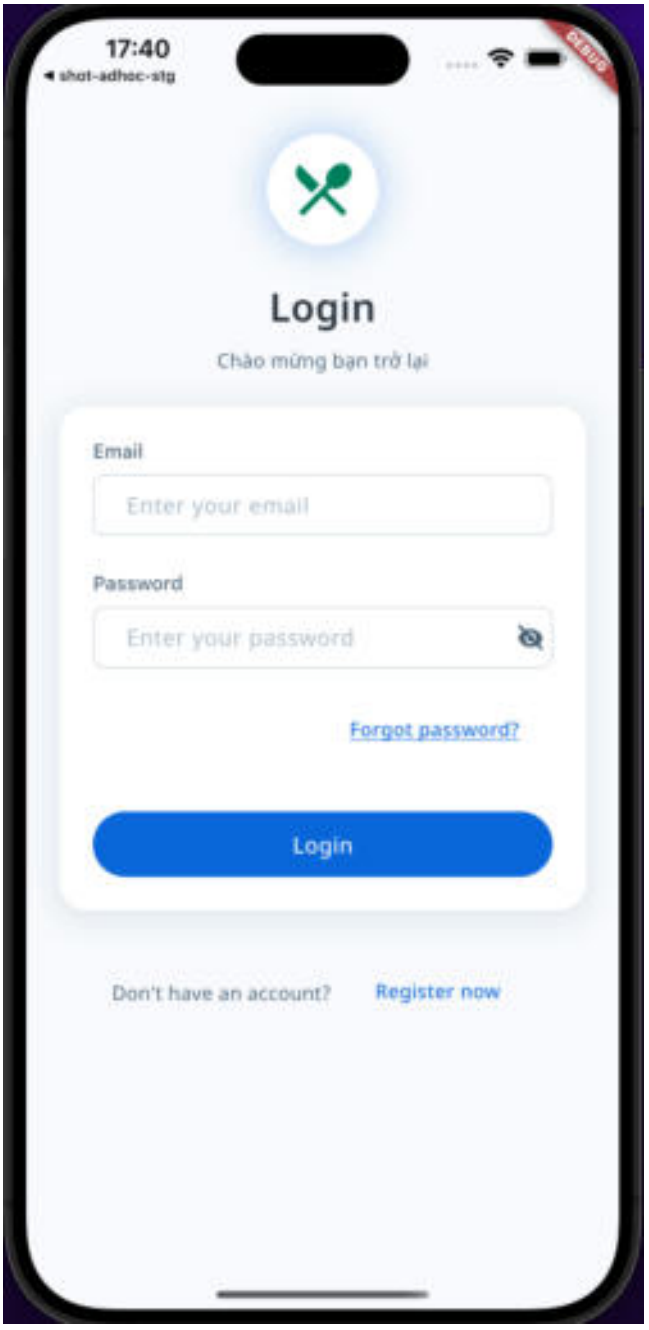
- Clean Architecture với 3 lớp rõ ràng (Presentation, Domain, Data)
- BLoC Pattern cho state management
- Dependency Injection với GetIt và Injectable
- Code được tổ chức theo feature modules
- Separation of concerns tốt

3.8.2. Đánh giá hiệu năng

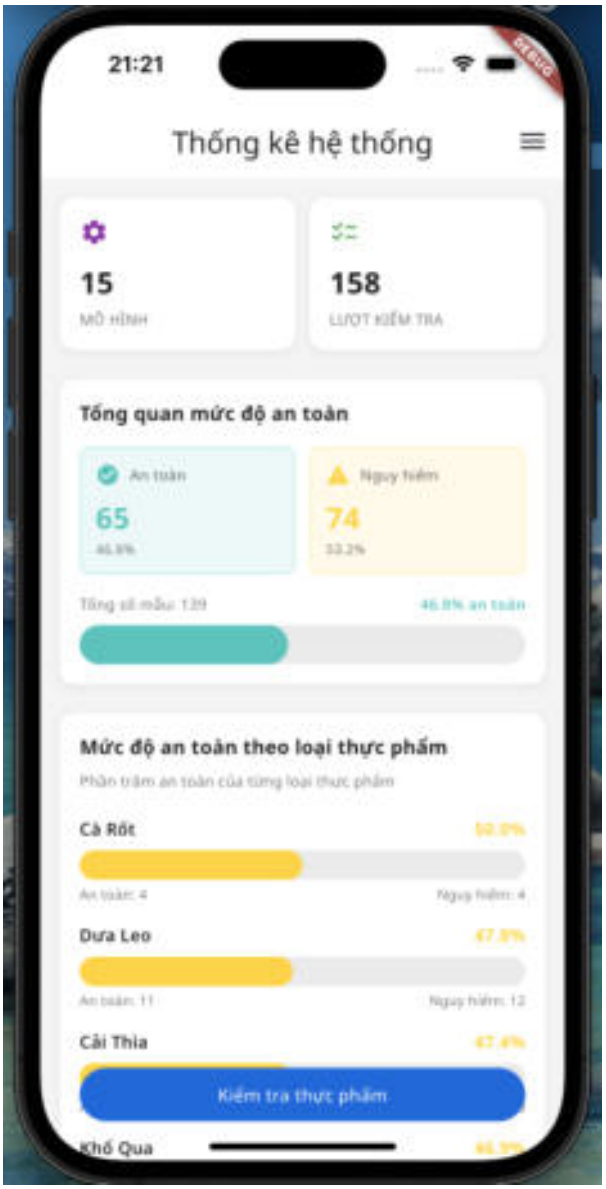
Thời gian phản hồi:

- Parse file quang phổ (2136 điểm): < 100ms
- Gửi request API và nhận response: 2-5 giây (tùy thuộc vào server)
- Hiển thị kết quả trên UI: < 50ms
- Khởi động ứng dụng: < 3 giây

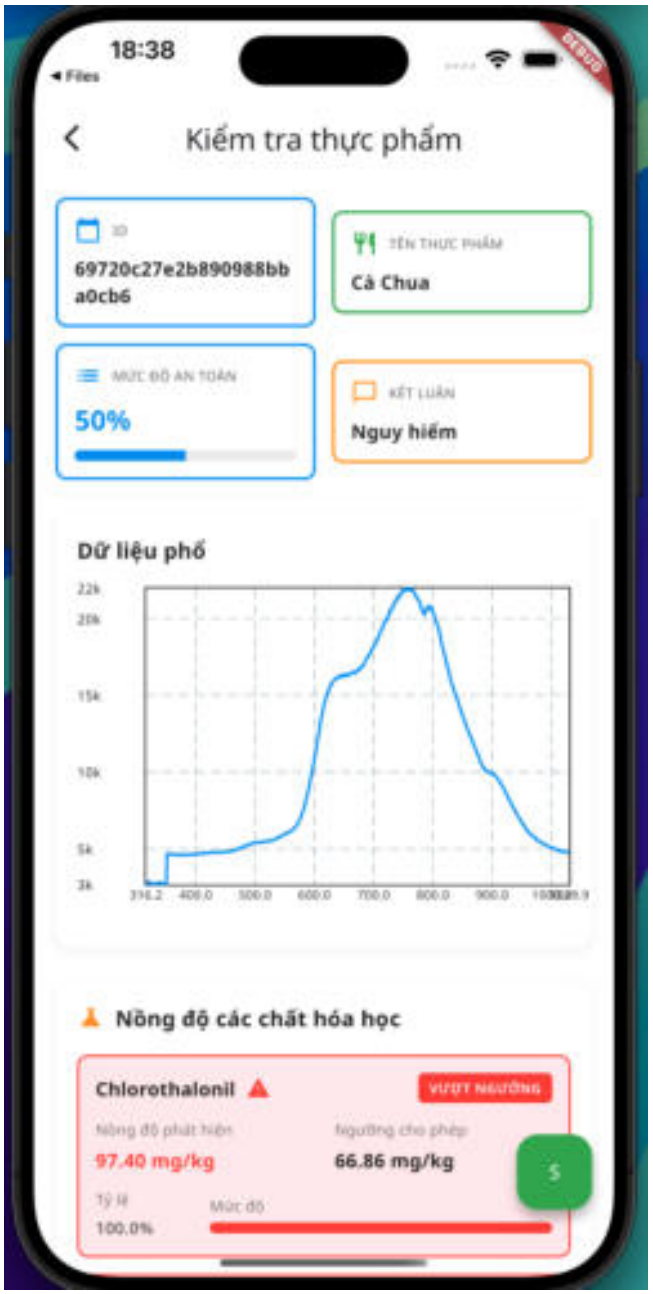
Kết quả thực tế



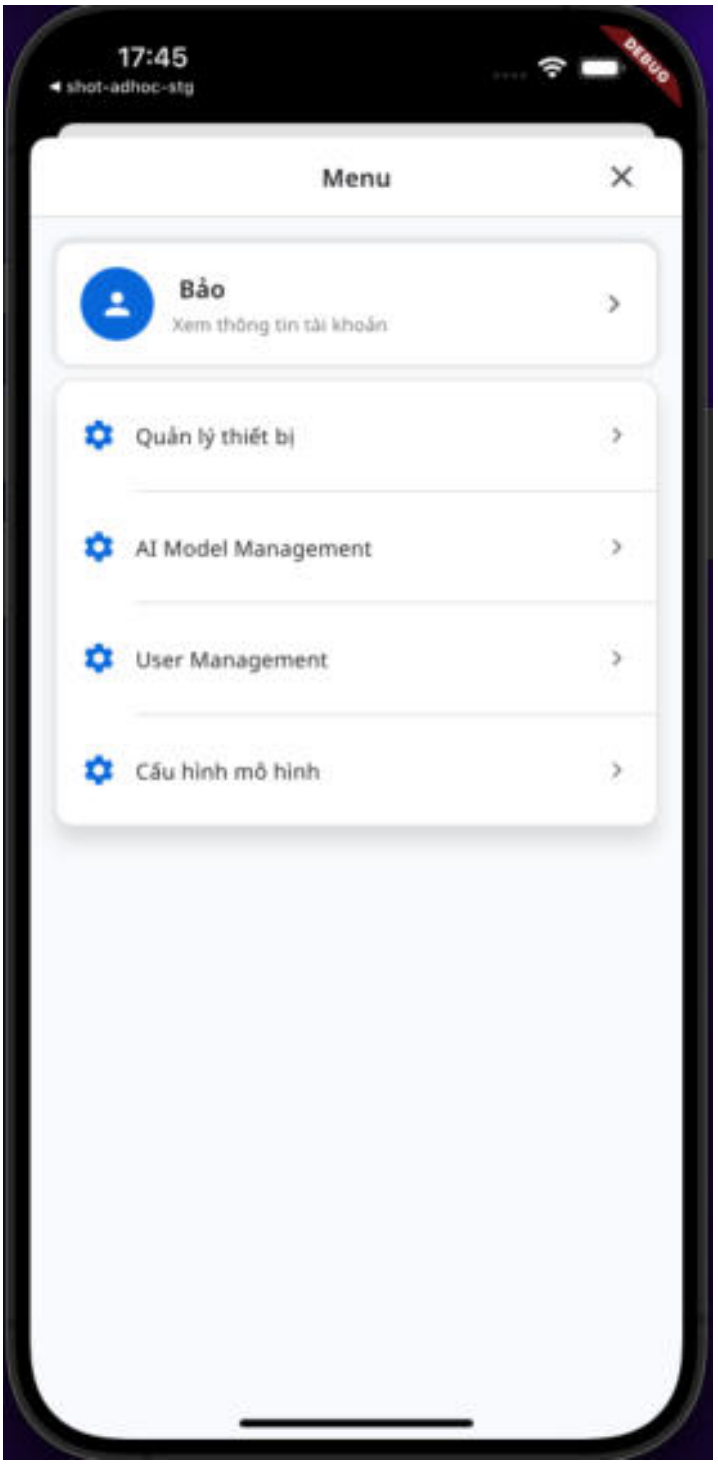
Hình 3.1.Màn hình Login



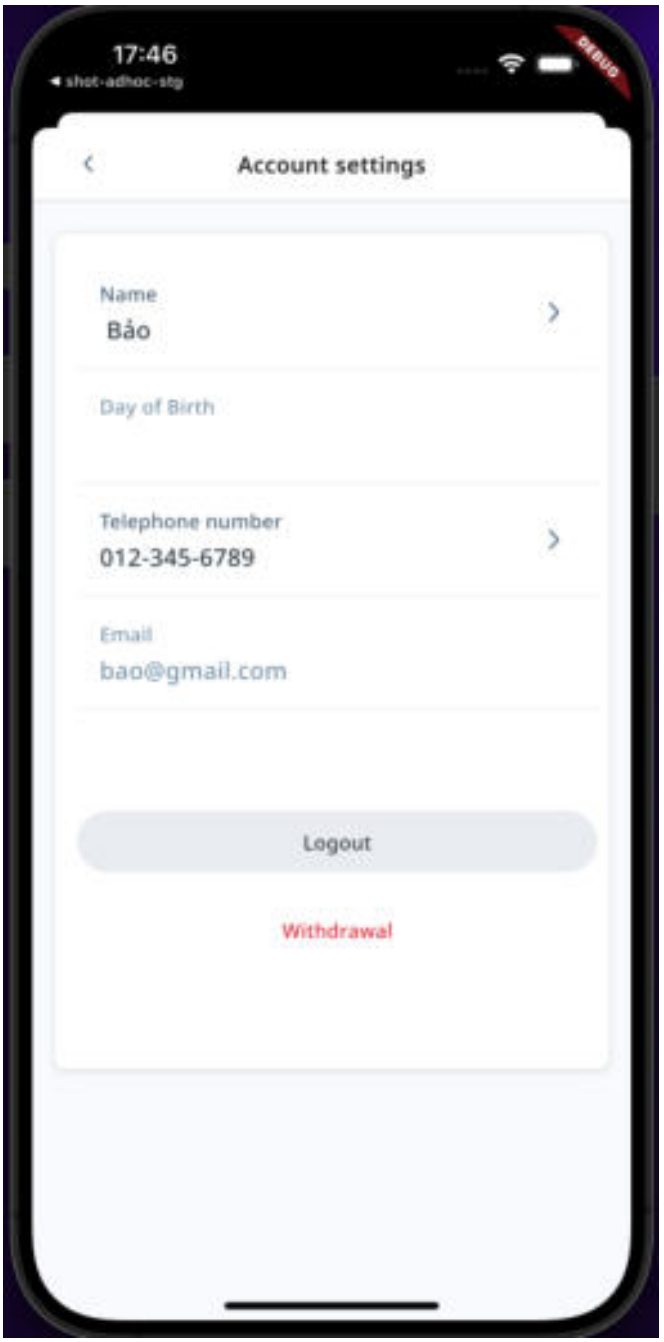
Hình 3.2.Màn hình Dashboard



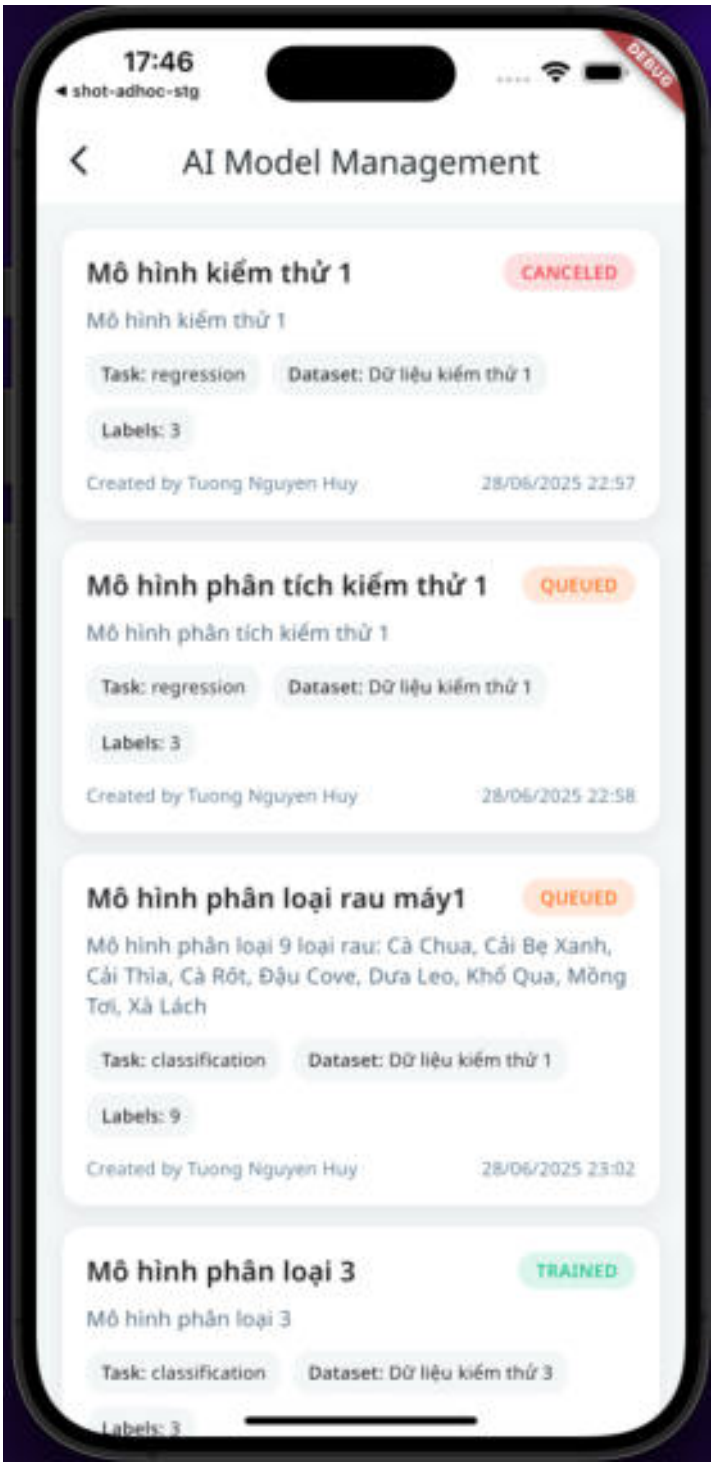
Hình 3.3.Màn hình Chi tiết phân tích thực phẩm



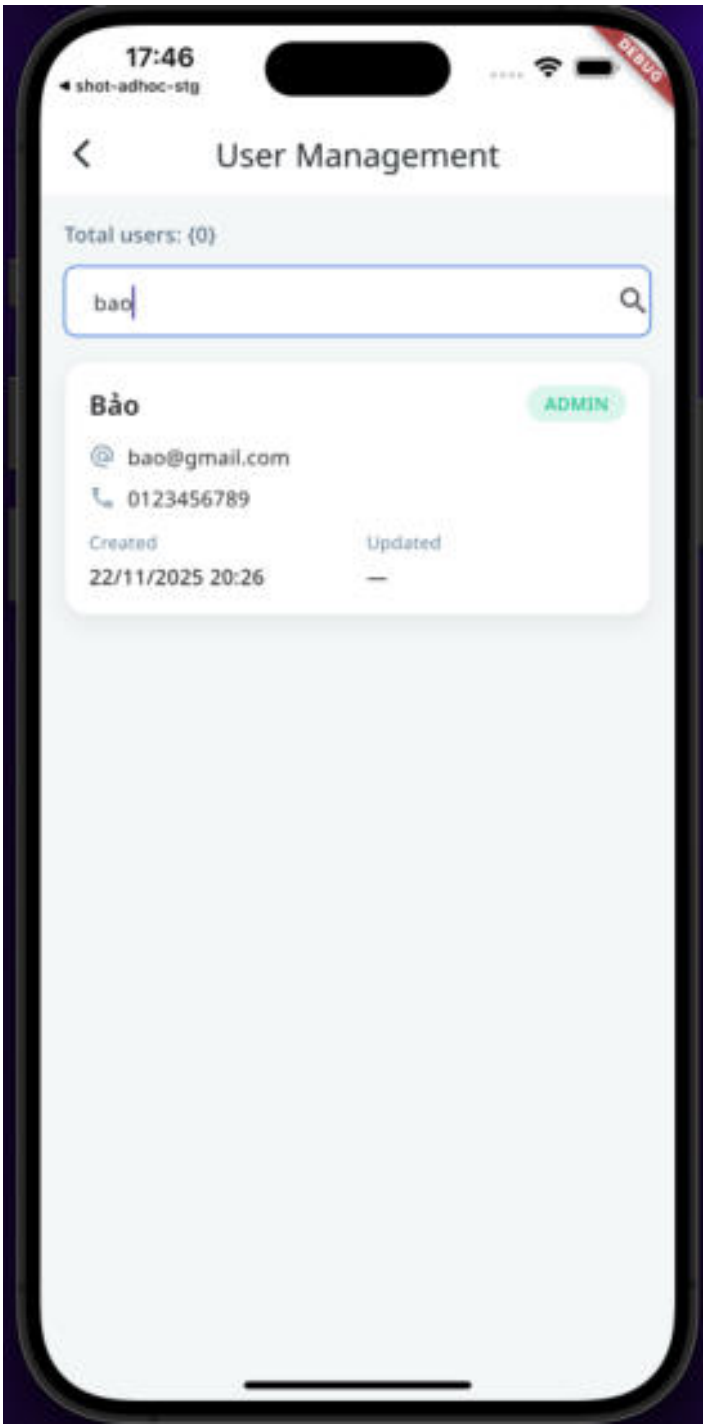
Hình 3.4.Menu quản lý



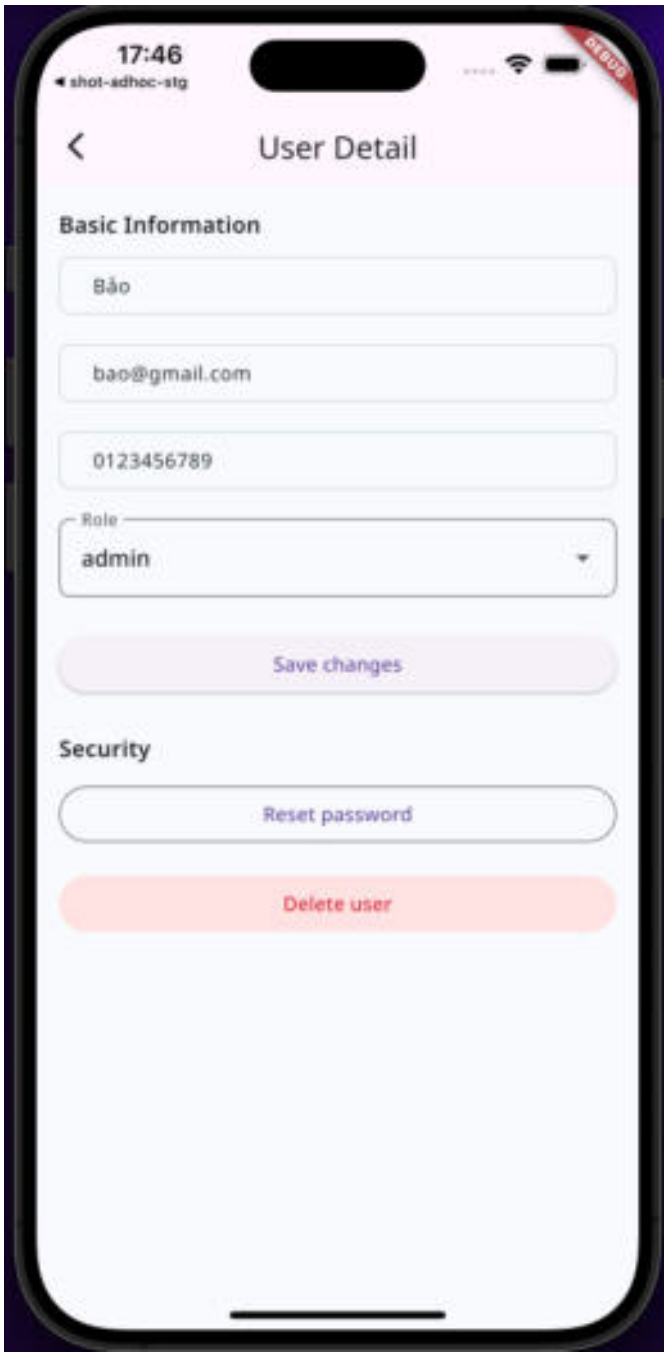
Hình 3.5.Trang xem thông tin tài khoản



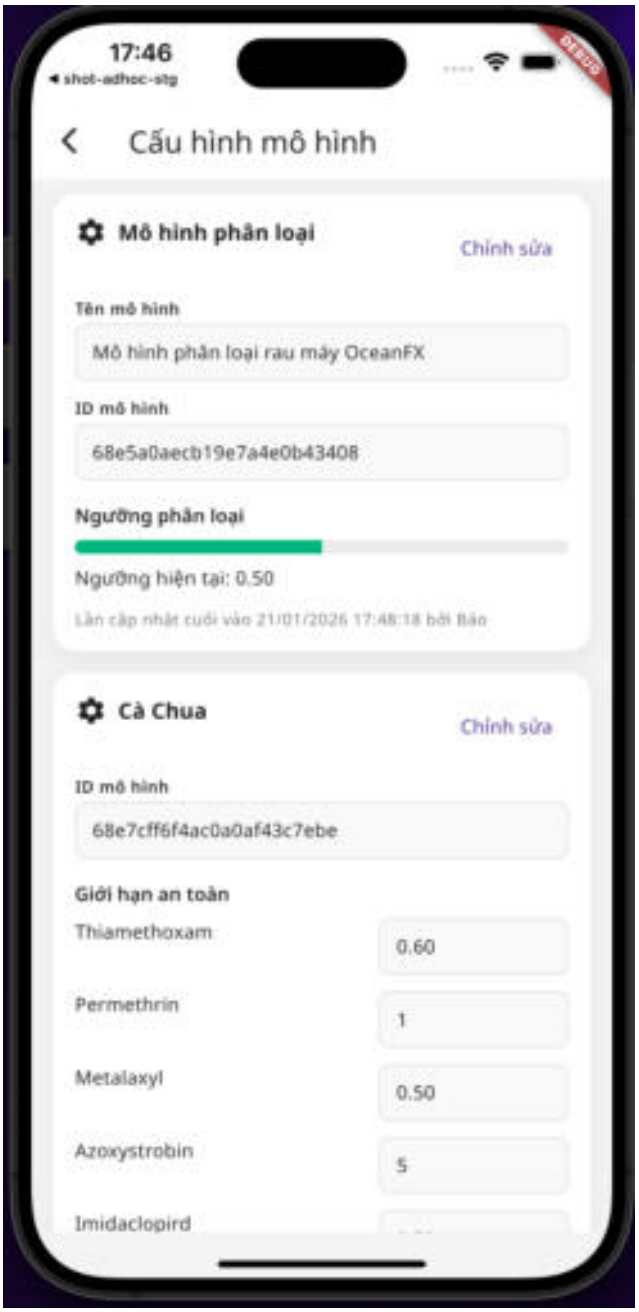
Hình 3.6.Trang xem danh sách model AI



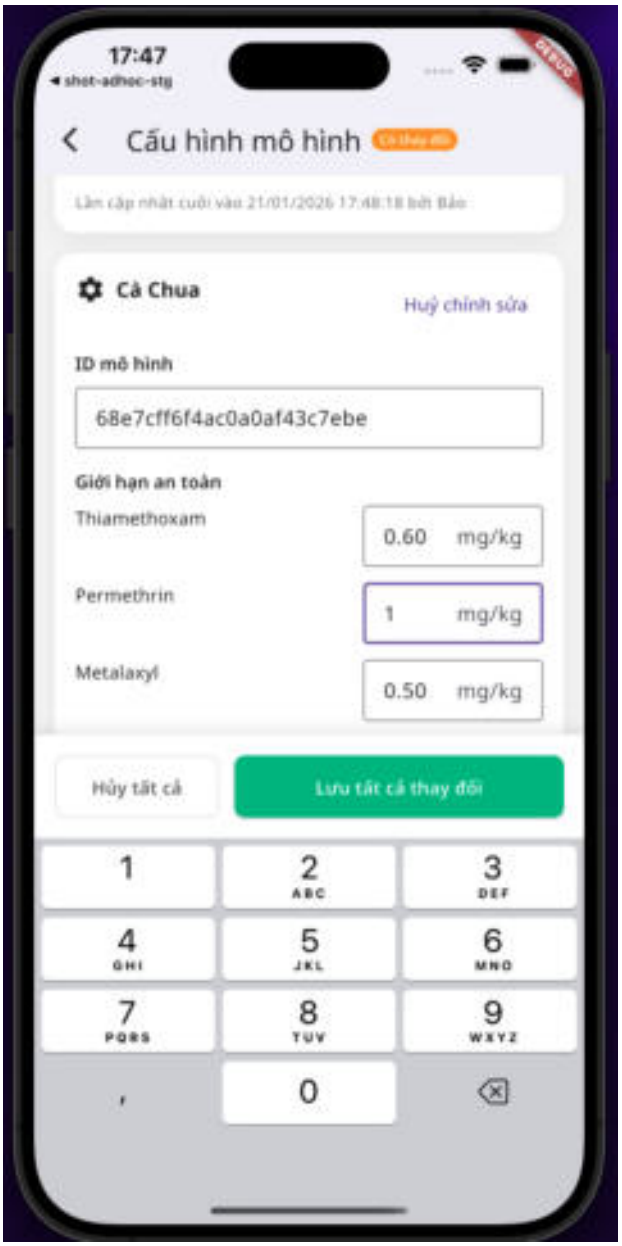
Hình 3.7.Màn hình quản lý user (Admin)



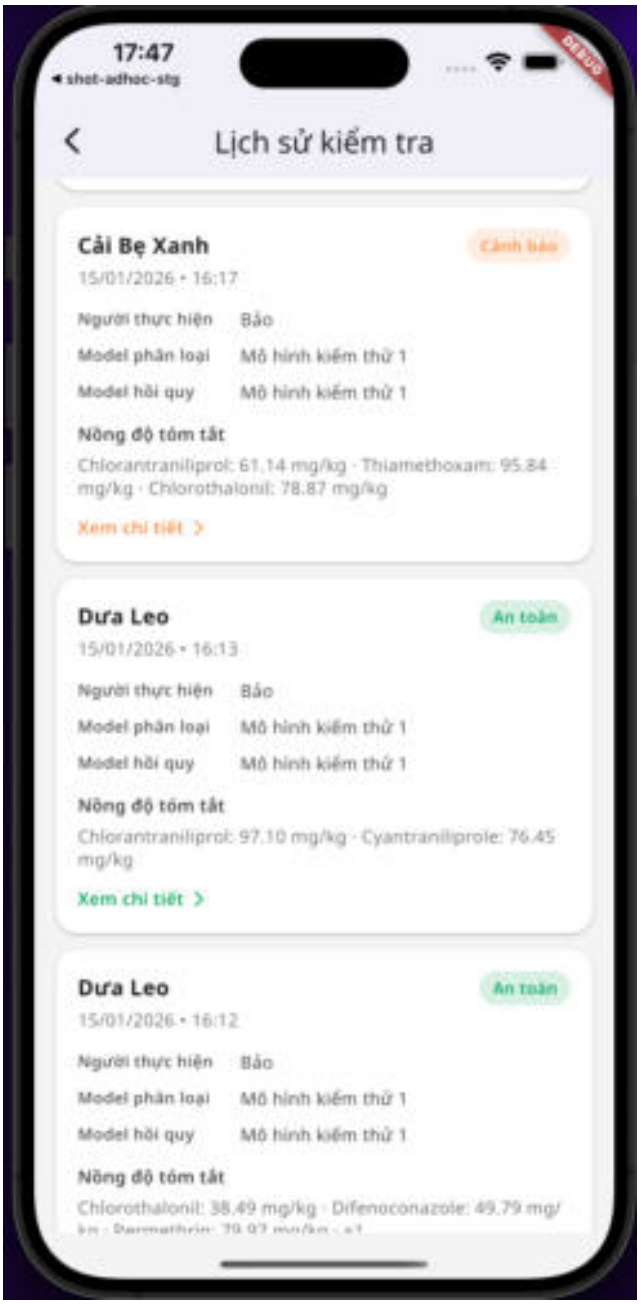
Hình 3.8.Màn hình chi tiết quản lý user (Admin)



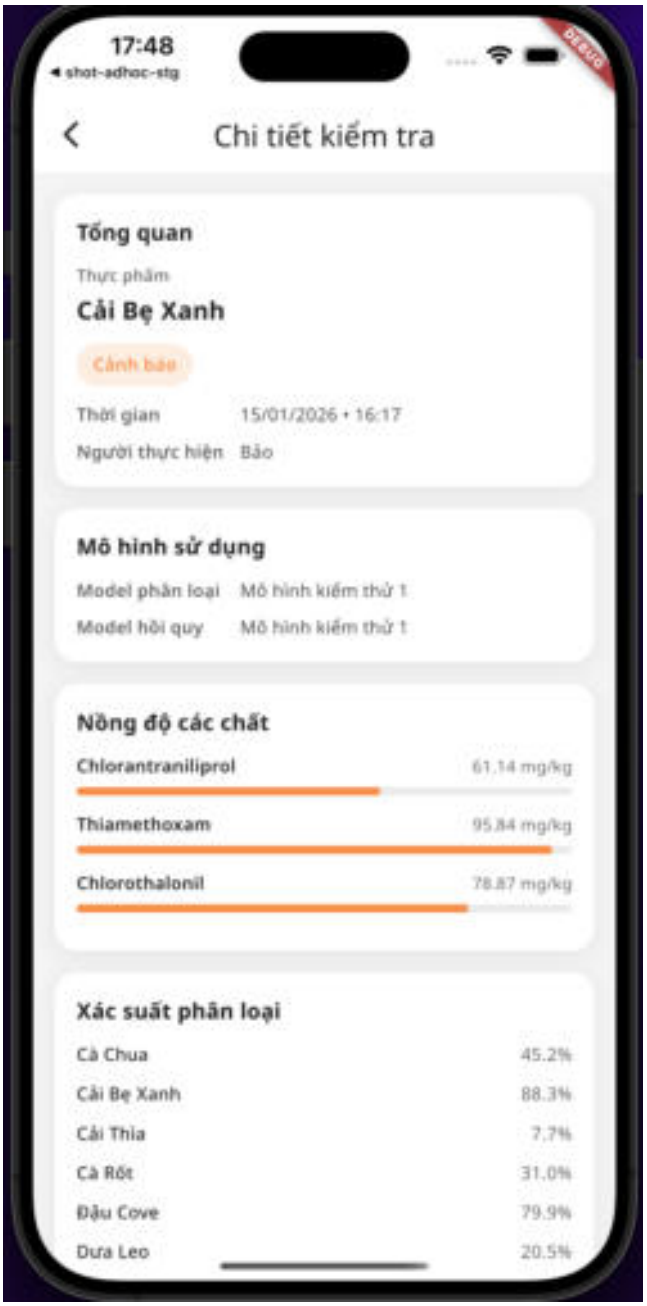
Hình 3.9.Màn hình xem thông tin model AI đang sử dụng



Hình 3.10.Màn hình chỉnh sửa thông tin model AI đang sử dụng



Hình 3.11.Màn hình xem lịch sử phân tích



Hình 3.12.Màn hình xem chi tiết lịch sử phân tích

3.8.3. Hạn chế và hướng phát triển

Hạn chế hiện tại:

1. **Kết nối thiết bị chưa ổn định:**
    - Module kết nối thiết bị đo quang phổ Ocean FX đã được thiết kế (theo kiến trúc hệ thống), tuy nhiên việc tích hợp kết nối real-time với hệ thống Raspberry Pi và WebSocket server thực tế đang gặp trục trặc do thiết bị Ocean FX đang gặp sự cố về driver (drive). Do đó, kết nối vẫn chưa ổn định và nhóm đang chờ sự hỗ trợ kỹ thuật từ hãng sản xuất/nhà cung cấp để khắc phục.

      Giải pháp dùng tạp output file để phân tích

Hướng phát triển trong tương lai:

1. **Hoàn thiện kết nối thiết bị mượt mà:**
    - Tích hợp với Raspberry Pi vào thiết bị Ocean FX một cách ổn định, mượt mà.
2. **Tính năng mới:**
    - So sánh kết quả phân tích giữa các lần kiểm tra
    - Export báo cáo PDF
    - Chia sẻ kết quả qua email/social media
    - Thống kê xu hướng an toàn thực phẩm theo thời gian
3. **Cải thiện UI/UX:**
    - Thêm animations và transitions mượt mà hơn
    - Dark mode
    - Tùy chỉnh theme
4. **Testing:**
    - Unit tests cho các use cases và repositories
    - Widget tests cho các components
    - Integration tests cho các luồng chính

KẾT LUẬN

Đồ án "Xây dựng ứng dụng đo lường và đánh giá thực phẩm thông minh" đã hoàn thành việc phát triển ứng dụng mobile Flutter với các tính năng chính:

1. **Phân tích thực phẩm:** Ứng dụng có khả năng nhận dữ liệu quang phổ từ file upload hoặc kết nối thiết bị, gửi lên API backend để phân tích bằng các mô hình AI/ML, và hiển thị kết quả một cách trực quan với các biểu đồ và bảng thống kê.
2. **Dashboard thống kê:** Cung cấp cái nhìn tổng quan về hệ thống với các chỉ số KPI, biểu đồ phân bố mức độ an toàn, và lịch sử kiểm tra.
3. **Quản lý người dùng:** Hệ thống xác thực hoàn chỉnh với JWT token, refresh token tự động, và quản lý tài khoản.
4. **Kiến trúc tốt:** Ứng dụng được phát triển theo Clean Architecture và BLoC Pattern, đảm bảo code dễ bảo trì, dễ mở rộng và dễ kiểm thử.

Đóng góp của đồ án:

- Cung cấp một giải pháp công nghệ để kiểm tra và đánh giá chất lượng thực phẩm một cách nhanh chóng và tiện lợi
- Ứng dụng công nghệ quang phổ NIR và AI/ML vào thực tế
- Tạo nền tảng để phát triển thêm các tính năng trong tương lai

Hướng phát triển:

Trong tương lai, đồ án có thể được mở rộng với các tính năng như:

- Hoàn thiện kết nối thiết bị để nhận dữ liệu real-time từ Raspberry Pi
- Thêm các tính năng thống kê và báo cáo nâng cao
- Tích hợp với các thiết bị đo quang phổ khác

TÀI LIỆU THAM KHẢO

[1] Flutter Team. (2024). *Flutter Documentation*. Truy cập từ: https://docs.flutter.dev/
[2] pub.dev. (2024). *pub.dev - Flutter Package Repository*. Truy cập từ: https://pub.dev/
Các thư viện Flutter được sử dụng trong dự án có thể được tìm kiếm và tra cứu tại trang pub.dev, bao gồm: flutter_bloc, go_router, dio, freezed, json_serializable, easy_localization, fl_chart, flutter_secure_storage, file_picker, get_it, injectable, flutter_screenutil, lottie và các thư viện khác.
[3] Ocean Insight. (2024). *Ocean FX Spectrometer - Product Information*. Truy cập từ: https://www.oceaninsight.com/products/spectrometers/modular-spectrometers/ocean-fx/
[4] SeaBreeze Project. (2024). *SeaBreeze - Open Source Spectrometer Driver*. Truy cập từ: https://github.com/ap--/seabreeze
[5] python-seabreeze Documentation. (2024). *python-seabreeze Documentation*. Truy cập từ: https://python-seabreeze.readthedocs.io/
[6] Reso Coder. (2024). *Flutter Clean Architecture Tutorial*. Truy cập từ: https://resocoder.com/flutter-clean-architecture-tdd/
[7] BLoC Pattern. (2024). *BLoC Pattern Documentation*. Truy cập từ: https://bloclibrary.dev/
[8] Workman, J., & Weyer, L. (2012). *Practical Guide to Interpretive Near-Infrared Spectroscopy*. CRC Press.
[9] MDN Web Docs. (2024). *WebSocket API Documentation*. Truy cập từ: https://developer.mozilla.org/en-US/docs/Web/API/WebSocket
[10] FastAPI. (2024). *FastAPI Documentation*. Truy cập từ: https://fastapi.tiangolo.com/

PHỤ LỤC

Phụ lục A: Cấu trúc thư mục dự án

[Cấu trúc thư mục đã được trình bày trong Chương 2]

Phụ lục B: Một số đoạn code quan trọng

[Các đoạn code đã được trình bày trong Chương 3]

Phụ lục C: Hình ảnh giao diện ứng dụng

[Các hình ảnh màn hình sẽ được thêm vào]