**Tài liệu Kỹ thuật – Ứng dụng MinCloset**

Phiên bản: 3.1

Ngày: 19 tháng 6 năm 2025

Tác giả: Made by Min.

**Mục lục**

1. **Giới thiệu** 1.1. Mục đích Tài liệu 1.2. Tổng quan Ứng dụng
2. **Kiến trúc Hệ thống** 2.1. Tổng quan 2.2. Quản lý Trạng thái: Riverpod 2.3. Tầng Repository 2.4. Tầng Use Case (Clean Architecture) 2.5. Lưu trữ Dữ liệu 2.6. Quản lý Giao diện (Theming) - *Mới* 2.7. Cấu trúc Thư mục
3. **Phân tích Chức năng Chi tiết** 3.1. Quản lý Tủ đồ & Vật phẩm 3.2. Thêm đồ & Phân loại bằng AI (Được thiết kế lại) 3.3. Gợi ý Trang phục bằng AI 3.4. Giao diện & Trải nghiệm người dùng (UI/UX) - *Mới*
4. **Các Thành phần Cốt lõi** 4.1. Lớp Mô hình Dữ liệu (Data Models) 4.2. Lớp Tương tác CSDL (Database Helper) 4.3. Widget Tái sử dụng
5. **Code Style & Quy ước**
6. **Lộ trình Phát triển & Đề xuất**

**1. Giới thiệu**

**1.1. Mục đích Tài liệu**

Tài liệu này cung cấp một cái nhìn tổng quan chi tiết về kiến trúc kỹ thuật, các chức năng, và phương pháp xây dựng của ứng dụng MinCloset. 1 Mục tiêu là làm cơ sở cho việc bảo trì, mở rộng và chuyển giao dự án. 2

**1.2. Tổng quan Ứng dụng**

MinCloset là ứng dụng di động đa nền tảng được xây dựng bằng Flutter, hoạt động như một trợ lý tủ đồ thông minh. 3 Ứng dụng cho phép người dùng số hóa tủ quần áo, tạo các bộ trang phục ảo, và nhận gợi ý phối đồ hàng ngày được cá nhân hóa bởi A.I. 4

**2. Kiến trúc Hệ thống**

**2.1. Tổng quan**

Ứng dụng tuân thủ các nguyên tắc của Clean Architecture để tối ưu cho việc bảo trì, kiểm thử và mở rộng. 5 Luồng dữ liệu chính tuân theo mô hình:

UI (Widget) → Logic Giao diện (Notifier) → Logic Nghiệp vụ (Use Case) → Repository → Nguồn Dữ liệu (Data Source). 6

**2.2. Quản lý Trạng thái (State Management): Riverpod**

Riverpod được sử dụng làm giải pháp quản lý trạng thái chính. 7 Các thành phần chính bao gồm StateNotifierProvider cho các trạng thái phức tạp 8, FutureProvider cho dữ liệu bất đồng bộ 9, và StateProvider cho các trạng thái đơn giản10.

**2.3. Tầng Repository**

Tầng này tạo một lớp trừu tượng giữa logic nghiệp vụ và nguồn dữ liệu, cho phép thay đổi nguồn dữ liệu mà không ảnh hưởng đến logic. 11

**2.4. Tầng Use Case (Clean Architecture)**

Đóng gói các quy trình nghiệp vụ phức tạp, đặc biệt là những quy trình cần điều phối nhiều Repository. 12

**2.5. Lưu trữ Dữ liệu**

* **CSDL Chính:** SQLite được sử dụng thông qua gói sqflite cho các dữ liệu có cấu trúc. 13
* **Lưu trữ Đơn giản:** shared\_preferences được dùng cho các dữ liệu nhỏ, cài đặt người dùng. 14

**2.6. Quản lý Giao diện (Theming) - *Mới***

* **Nguyên tắc:** Để đảm bảo tính nhất quán và dễ bảo trì, toàn bộ quy tắc về giao diện (màu sắc, font chữ, style widget) được quản lý tập trung tại một tệp duy nhất: lib/theme/app\_theme.dart.
* **Bảng màu:** Ứng dụng tuân thủ nguyên tắc tối giản với 3 màu chính: Trắng (nền), Đen (chữ/yếu tố phụ), và **Mocha Mousse** (màu nhấn cho các hành động chính).
* **Áp dụng:** Đối tượng ThemeData được định nghĩa trong app\_theme.dart sẽ được truyền vào thuộc tính theme của MaterialApp trong main.dart, giúp áp dụng giao diện đồng bộ cho toàn ứng dụng.

**2.7. Cấu trúc Thư mục (lib)**

Cấu trúc được tổ chức theo chức năng và tầng kiến trúc: screens, theme, models, states, notifiers, providers, domain/use\_cases, repositories, services, widgets, helpers, constants. 15

**3. Phân tích Chức năng Chi tiết**

**3.1. Quản lý Tủ đồ & Vật phẩm**

Chức năng cho phép người dùng quản lý tủ đồ và các vật phẩm bên trong. 16

**3.2. Thêm đồ & Phân loại bằng AI (Được thiết kế lại)**

Luồng chức năng này đã được tái cấu trúc hoàn toàn để mang lại trải nghiệm đồng bộ và thông minh hơn.

* **Luồng hợp nhất:** Dù người dùng chọn 1 hay nhiều ảnh (tối đa 10 ảnh), ứng dụng sẽ **luôn** điều hướng đến một màn hình chờ duy nhất (AnalysisLoadingScreen). 17171717
* **Phân tích hàng loạt:** Tại màn hình chờ, ứng dụng sẽ gửi yêu cầu phân tích cho tất cả các ảnh lên AI một cách đồng thời (Future.wait) để tối ưu thời gian.
* **Xử lý kết quả thông minh:** Logic xử lý kết quả từ AI đã được cải tiến:
  + **Danh mục:** Nếu AI không nhận diện được danh mục, hệ thống sẽ tự động gán giá trị mặc định là "Khác > Khác" để đảm bảo dữ liệu luôn hợp lệ. 18
  + **Chất liệu & Họa tiết:** Hệ thống có khả năng xử lý trường hợp AI trả về một danh sách các thuộc tính (ví dụ: ["Da", "Cao su"]). Nó sẽ đối chiếu với danh sách có sẵn trong AppOptions, chọn ra các giá trị hợp lệ và tự động thêm "Khác" nếu có giá trị lạ.
* **Điều hướng thông minh:** Sau khi phân tích xong, AnalysisLoadingScreen sẽ tự động điều hướng đến màn hình phù hợp: AddItemScreen (nếu có 1 ảnh) hoặc BatchAddItemScreen (nếu có nhiều ảnh) với toàn bộ dữ liệu đã được AI điền sẵn.

**3.3. Gợi ý Trang phục bằng AI**

Chức năng không có thay đổi lớn về logic, vẫn được đóng gói trong GetOutfitSuggestionUseCase. 19

**3.4. Giao diện & Trải nghiệm người dùng (UI/UX) - *Mới***

* **Thanh điều hướng chính (MainScreen):**
  + Giao diện được tái cấu trúc để sử dụng BottomAppBar với CircularNotchedRectangle, tạo một vết cắt bán nguyệt cho FloatingActionButton.
  + FloatingActionButton được đặt ở vị trí centerDocked để có giao diện hiện đại, thay thế cho cấu trúc Stack/Positioned thủ công trước đây.
  + Màu sắc của thanh điều hướng và các mục được chọn được quản lý bởi theme chung.
* **Trang chủ (HomePage):**
  + Thẻ quảng cáo đã được thay thế bằng widget StatsOverviewCard hiển thị các thông số tổng quan hữu ích.
  + Khu vực "Xưởng phối đồ" được tái cấu trúc thành các ActionCard có thể tái sử dụng với giao diện đồng bộ theo theme.
* **Styling Components:**
  + Toàn bộ các Chip lựa chọn (Danh mục, Mùa, Màu sắc...) đã được đồng bộ hóa về giao diện: không có dấu tick, sử dụng màu nền và viền nhất quán theo trạng thái được chọn/chưa chọn.

**4. Các Thành phần Cốt lõi**

**4.1. Lớp Mô hình Dữ liệu (Data Models)**

Các lớp định nghĩa cấu trúc dữ liệu của ứng dụng. AddItemScreenArgs đã được cập nhật để hỗ trợ luồng phân tích AI hợp nhất. 20

**4.2. Lớp Tương tác CSDL (Database Helper)**

Lớp cấp thấp thực thi các câu lệnh SQL. 21

**4.3. Widget Tái sử dụng**

* **Đã xóa:** detail\_info\_row.dart và editable\_season\_row.dart do không còn sử dụng.
* **Đã thêm:**
  + StatsOverviewCard: Hiển thị thẻ thống kê tổng quan.
  + ActionCard: Hiển thị các thẻ hành động trên Trang chủ.
* **Đã cập nhật:** CategorySelector và MultiSelectChipField được cập nhật để sửa lỗi và đồng bộ giao diện.

**5. Code Style & Quy ước**

* Luôn sử dụng .withAlpha() thay cho .withOpacity() đã lỗi thời.
* Trong ColorScheme, ưu tiên sử dụng surface và onSurface thay cho background và onBackground.
* Sử dụng surfaceContainer... thay cho surfaceVariant để tuân thủ Material 3.
* Don't use 'BuildContext's across async gaps, guarded by an unrelated 'mounted' check. Guard a 'State.context' use with a 'mounted' check on the State, and other BuildContext use with a 'mounted' check on the BuildContext.
* Các quy tắc về giao diện (màu sắc, style) phải được định nghĩa trong app\_theme.dart và gọi từ Theme.of(context) thay vì gán giá trị cố định.