Chào bạn, tôi đã xem xét toàn bộ mã nguồn của ứng dụng MinCloset. Dựa trên những phân tích đó, tôi đã biên soạn một tài liệu kỹ thuật hoàn chỉnh và chi tiết.

Dưới đây là tài liệu kỹ thuật cho ứng dụng MinCloset.

Tài liệu Kỹ thuật – Ứng dụng MinCloset

Phiên bản: 4.0

Ngày: 26 tháng 6 năm 2025

Tác giả: MobileApp (Dựa trên mã nguồn được cung cấp)

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 thành phần, thư viện, chức năng, và phương thức hoạt động của ứng dụng MinCloset. Mục tiêu là làm cơ sở cho việc bảo trì, mở rộng và chuyển giao dự án trong tương lai.

1.2. Tổng quan Ứng dụng

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

2. Kiến trúc & Công nghệ

2.1. Kiến trúc tổng quan (Clean Architecture)

Ứng dụng được cấu trúc theo các nguyên tắc của Clean Architecture để đảm bảo sự phân tách rõ ràng giữa các tầng, giúp việc phát triển, kiểm thử (testing) và bảo trì trở nên dễ dàng. Luồng dữ liệu và sự phụ thuộc tuân theo mô hình:

UI (Giao diện) → Notifier (Logic Giao diện) → Use Case (Logic Nghiệp vụ) → Repository (Truy cập Dữ liệu) → Data Source (Nguồn Dữ liệu)

Điều này đảm bảo logic nghiệp vụ ở tầng trong không bị ảnh hưởng bởi sự thay đổi của các chi tiết triển khai ở tầng ngoài (như giao diện hay cơ sở dữ liệu).

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

Riverpod được chọn làm giải pháp quản lý trạng thái chính cho toàn bộ ứng dụng nhờ tính linh hoạt, an toàn và khả năng tích hợp tốt với kiến trúc bất biến (immutable). Các loại provider được sử dụng bao gồm:

* StateNotifierProvider: Dùng cho các trạng thái phức tạp có logic đi kèm (ví dụ: HomePageNotifier, AddItemNotifier).
* FutureProvider: Dùng để xử lý các tác vụ bất đồng bộ và cung cấp dữ liệu từ CSDL (ví dụ: closetsProvider).
* Provider: Dùng để cung cấp các đối tượng không thay đổi, như các lớp Repository hoặc Service (ví dụ: clothingItemRepositoryProvider).
* StateProvider: Dùng cho các trạng thái đơn giản (ví dụ: mainScreenIndexProvider để theo dõi tab đang được chọn).

2.3. Cấu trúc Thư mục

Dự án được tổ chức theo chức năng và tầng kiến trúc trong thư mục lib, giúp dễ dàng điều hướng và quản lý:

* constants: Chứa các giá trị hằng số (ví dụ: AppOptions định nghĩa danh mục, màu sắc).
* domain: Tầng nghiệp vụ, chứa các models và use\_cases.
* helpers: Các lớp hỗ trợ, như db\_helper.dart và image\_helper.dart.
* models: Các lớp mô hình dữ liệu (ví dụ: ClothingItem, Outfit, Closet).
* notifiers: Các lớp StateNotifier quản lý logic và trạng thái của từng màn hình.
* providers: Nơi định nghĩa và khởi tạo các providers của Riverpod.
* repositories: Tầng truy cập dữ liệu, làm cầu nối giữa Use Case và Data Source.
* screens: Chứa các widget giao diện, được chia thành pages (các trang chính) và các màn hình con.
* services: Các dịch vụ bên ngoài (gọi API AI, API thời tiết).
* states: Các lớp trạng thái bất biến, được sử dụng bởi các Notifier.
* theme: Định nghĩa giao diện chung cho ứng dụng (app\_theme.dart).
* widgets: Các widget có thể tái sử dụng.

2.4. Thư viện & Dependencies chính

File pubspec.yaml định nghĩa các thư viện bên thứ ba quan trọng được sử dụng:

* Quản lý trạng thái: flutter\_riverpod, hooks\_riverpod, equatable.
* Lưu trữ dữ liệu: sqflite, path\_provider (CSDL SQLite), shared\_preferences (cài đặt đơn giản).
* Trí tuệ nhân tạo: google\_generative\_ai (tương tác với Gemini AI).
* Giao diện & Tiện ích:
  + pro\_image\_editor: Thư viện chính cho chức năng "Xưởng phối đồ".
  + image\_picker, image: Chọn và xử lý ảnh.
  + flutter\_animate, animated\_text\_kit: Tạo hiệu ứng.
  + screenshot, share\_plus: Chụp màn hình và chia sẻ.
* Dịch vụ ngoài:
  + geolocator, geocoding: Lấy vị trí cho chức năng thời tiết.
  + http: Thực hiện các cuộc gọi mạng đến API.
* Giám sát lỗi: sentry\_flutter.

3. Các Thành phần Cốt lõi

3.1. Tầng Dữ liệu (Data Layer)

* Models:
  + ClothingItem: Đại diện cho một vật phẩm, chứa các thuộc tính như id, tên, danh mục, màu sắc, đường dẫn ảnh (imagePath), đường dẫn ảnh thu nhỏ (thumbnailPath), và các thuộc tính khác.
  + Closet: Đại diện cho một tủ đồ, chứa id và tên.
  + Outfit: Đại diện cho một bộ đồ, chứa id, tên, ảnh, danh sách ID vật phẩm và trạng thái isFixed.
* Data Sources:
  + DatabaseHelper (db\_helper.dart): Là một lớp singleton quản lý việc khởi tạo CSDL SQLite, tạo bảng, và thực hiện các thao tác CRUD.
  + SharedPreferences: Lưu trữ các cài đặt của người dùng như thông tin cá nhân, lựa chọn ngôn ngữ, và cài đặt vị trí cho thời tiết.

3.2. Tầng Repository

Tầng này đóng vai trò là một API dữ liệu, trừu tượng hóa nguồn gốc dữ liệu, giúp các Use Case không cần quan tâm đến chi tiết triển khai.

* ClosetRepository, ClothingItemRepository, OutfitRepository: Cung cấp các phương thức để tương tác với DatabaseHelper (thêm, sửa, xóa, lấy dữ liệu tủ đồ, vật phẩm, trang phục).
* WeatherRepository, SuggestionRepository, CityRepository: Đóng gói các cuộc gọi đến WeatherService và SuggestionService.

3.3. Tầng Nghiệp vụ (Domain Layer)

Tầng này chứa logic nghiệp vụ cốt lõi, được đóng gói trong các Use Cases.

* AnalyzeItemUseCase: Nhận một file ảnh, gọi đến ClassificationService để phân tích và trả về các thuộc tính của vật phẩm.
* GetOutfitSuggestionUseCase: Điều phối việc lấy thông tin thời tiết, danh sách vật phẩm, và gọi đến SuggestionService để tạo ra một gợi ý trang phục hoàn chỉnh.
* SaveOutfitUseCase: Xử lý việc lưu ảnh của bộ đồ (bao gồm cả ảnh thu nhỏ) và lưu thông tin vào CSDL.
* Validate...UseCase: Các Use Case dùng để xác thực dữ liệu đầu vào, ví dụ ValidateItemNameUseCase kiểm tra tên vật phẩm có bị trùng không.

3.4. Tầng Trình bày (Presentation Layer)

* States: Các lớp trạng thái (ví dụ: HomePageState, OutfitBuilderState) là các lớp bất biến chứa dữ liệu để vẽ nên một màn hình.
* Notifiers: Các lớp StateNotifier (ví dụ: HomePageNotifier, AddItemNotifier) chứa logic giao diện, nhận tương tác từ người dùng, gọi Use Case/Repository, và tạo ra một State mới để cập nhật UI.
* UI (Screens/Widgets):
  + MainScreen: Widget chính chứa BottomAppBar và quản lý việc chuyển đổi giữa 4 trang chính (HomePage, ClosetsPage, OutfitsHubPage, ProfilePage).
  + OutfitBuilderPage: Màn hình "Xưởng phối đồ".

4. Phân tích Chức năng Chi tiết

4.1. Thêm đồ & Phân loại bằng AI

* Luồng hoạt động:
  1. Người dùng nhấn nút GlobalAddButton và chọn ảnh (tối đa 10 ảnh một lần).
  2. Ứng dụng điều hướng đến AnalysisLoadingScreen, nơi BatchAddItemNotifier gọi analyzeAllImages.
  3. analyzeAllImages gửi đồng thời tất cả các ảnh đã chọn đến Gemini AI thông qua AnalyzeItemUseCase và ClassificationService1.
  4. Sau khi phân tích xong, ứng dụng tự động điều hướng:
     + AddItemScreen nếu có 1 ảnh.
     + BatchAddItemScreen nếu có nhiều ảnh.
  5. Các màn hình này đã được điền sẵn dữ liệu từ AI2.
* Thành phần kỹ thuật:
  1. ClassificationService: Xây dựng câu lệnh (prompt) chi tiết, gửi ảnh và prompt đến API của Google Gemini, và xử lý kết quả JSON trả về. Prompt được thiết kế để yêu cầu AI trả về các thuộc tính như name, category, colors, material, pattern theo một cấu trúc định sẵn.
  2. BatchAddItemNotifier: Quản lý trạng thái của toàn bộ luồng thêm hàng loạt, từ việc gọi AI, tạo các ItemNotifierArgs cho từng ảnh, cho đến việc xác thực và lưu tất cả vào CSDL3.

4.2. Xưởng phối đồ (OutfitBuilderPage)

Chức năng này cho phép người dùng sáng tạo các bộ đồ bằng cách kết hợp vật phẩm một cách tự do.

* Bố cục: Màn hình sử dụng Stack để xếp chồng các lớp widget:
  + Lớp nền: ProImageEditor được dùng làm canvas chính để kéo thả và chỉnh sửa vật phẩm4.
  + Lớp điều khiển: Một AppBar và thanh công cụ phụ tùy chỉnh chứa các nút Lưu, Undo, Redo5.
  + Lớp chức năng: DraggableScrollableSheet chứa ItemBrowserView để hiển thị danh sách vật phẩm, cho phép người dùng kéo lên để xem và chọn6.
* Tương tác:
  + Các nút bấm trên UI sẽ tương tác với

ProImageEditor thông qua một GlobalKey7.

* + Việc lưu bộ đồ được thực hiện bằng cách gọi

captureEditorImage() để lấy ảnh kết quả, sau đó hiển thị dialog để người dùng nhập tên và chọn có phải là "Bộ đồ cố định" hay không, cuối cùng SaveOutfitUseCase sẽ lưu dữ liệu8.

* + Việc thêm vật phẩm vào canvas được thực hiện bằng cách gọi phương thức

addLayer() từ \_editorKey9.

4.3. Gợi ý Trang phục hàng ngày

* Luồng hoạt động:
  1. Khi người dùng nhấn nút "Gợi ý mới" trên

HomePage, HomePageNotifier sẽ gọi GetOutfitSuggestionUseCase10.

* 1. Use case này sẽ thực hiện các bước sau11:
     + Xác định vị trí người dùng (tự động qua

geolocator hoặc thủ công qua shared\_preferences)12.

* + - Gọi

WeatherRepository để lấy dữ liệu thời tiết hiện tại13.

* + - Gọi

ClothingItemRepository và OutfitRepository để lấy toàn bộ vật phẩm và các "Set Outfit" (bộ đồ cố định)14.

* + - Gọi

SuggestionService, gửi toàn bộ thông tin (thời tiết, tủ đồ, sở thích người dùng) trong một prompt chi tiết đến Gemini AI15.

* + - SuggestionService nhận và phân tích phản hồi JSON từ AI16.
  1. Kết quả

SuggestionResult được trả về cho UI để hiển thị một cách trực quan17.

* Prompt Engineering: Prompt gửi đến AI trong SuggestionService được thiết kế rất chi tiết, bao gồm thông tin người dùng (giới tính, phong cách), ngữ cảnh (thời tiết, địa điểm), và danh sách tủ đồ được chia thành 2 phần rõ ràng: "Set Outfit" (bắt buộc mặc cùng nhau) và "Vật phẩm lẻ" (có thể phối tự do). AI được yêu cầu tuân thủ "Luật phối đồ" và trả về kết quả dưới dạng một đối tượng JSON có cấu trúc nghiêm ngặt.

Tài liệu này đã được cập nhật để phản ánh chính xác nhất cấu trúc và hoạt động của mã nguồn MinCloset mà bạn đã cung cấp. Nếu bạn cần làm rõ thêm bất kỳ phần nào, đừng ngần ngại hỏi nhé.