Chào bạn! Tôi là MobileAppBuilder, rất vui được hỗ trợ bạn xây dựng ứng dụng di động. Tôi đã xem qua toàn bộ mã nguồn ứng dụng MinCloset của bạn và nắm được cấu trúc, các chức năng chính mà bạn đã triển khai. Ứng dụng của bạn được xây dựng rất bài bản với **Clean Architecture** và sử dụng **Riverpod** cho quản lý trạng thái, cùng với nhiều tính năng thông minh tích hợp AI.

Dựa trên mã nguồn hiện tại, tôi đã cập nhật và viết lại nội dung trong tài liệu kỹ thuật của bạn để phản ánh chính xác nhất tình trạng của ứng dụng.

## Tài liệu Kỹ thuật – Ứng dụng MinCloset

Phiên bản: 4.0

Ngày: 03 tháng 7 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 MinCloset1. Mục tiêu là làm cơ sở cho việc bảo trì, mở rộng và chuyển giao dự án trong tương lai2.

#### 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 minh3. 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.)4.

### 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 dàng5. Luồng dữ liệu và sự phụ thuộc tuân theo mô hình6:

UI (Giao diện) → Notifier (Logic Giao diện) → Use Case (Logic Nghiệp vụ) → Repository (Truy cập Dữ liệu) → Data Source (Nguồn Dữ liệu) 7

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)8.

#### 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)9. Các loại provider được sử dụng bao gồm10:

* **StateNotifierProvider**: Dùng cho các trạng thái phức tạp có logic đi kèm (ví dụ: HomePageNotifier, ItemDetailNotifier, ClosetsPageNotifier)11.
* **FutureProvider**: Dùng để xử lý các tác vụ bất đồng bộ và cung cấp dữ liệu từ CSDL (ví dụ: closetsProvider, recentItemsProvider, itemsInClosetProvider)12.
* **Provider**: Dùng để cung cấp các đối tượng không thay đổi, như các lớp Repository hoặc Service (ví dụ: clothingItemRepositoryProvider, weatherServiceProvider)13.
* **StateProvider**: Dùng cho các trạng thái đơn giản (ví dụ: mainScreenIndexProvider để theo dõi tab đang được chọn, itemChangedTriggerProvider làm cơ chế kích hoạt sự kiện)14.

#### 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 lý15:

* **constants**: Chứa các giá trị hằng số (ví dụ: AppOptions định nghĩa danh mục, màu sắc, PromptStrings cho các câu lệnh AI)16.
* **domain**: Tầng nghiệp vụ, chứa các models và use\_cases, cũng như định nghĩa failures (các loại lỗi) và type\_defs (kiểu dữ liệu tùy chỉnh như FutureEither)17.
* **helpers**: Các lớp hỗ trợ, như db\_helper.dart (tương tác SQLite), image\_helper.dart (xử lý ảnh thu nhỏ), dialog\_helpers.dart (hiển thị dialog động), l10n\_helper.dart (hỗ trợ đa ngôn ngữ)18.
* **models**: Các lớp mô hình dữ liệu (ví dụ: ClothingItem, Outfit, Closet, Quest, Achievement, Badge, NotificationType)19.
* **notifiers**: Các lớp StateNotifier quản lý logic và trạng thái của từng màn hình hoặc chức năng (HomePageNotifier, ItemDetailNotifier, CalendarNotifier, ClosetInsightsNotifier, QuestMascotNotifier)20.
* **providers**: Nơi định nghĩa và khởi tạo các providers của Riverpod, nhóm theo chức năng (database\_providers, repository\_providers, service\_providers, event\_providers, ui\_providers, flow\_providers)21.
* **repositories**: Tầng truy cập dữ liệu, làm cầu nối giữa Use Case và Data Source, trả về kết quả dưới dạng Either<Failure, T>22.
* **screens**: Chứa các widget giao diện, được chia thành pages (các trang chính của MainScreen) và các màn hình con (ví dụ: ItemDetailScreen, EditProfileScreen, SettingsPage)23.
* **services**: Các dịch vụ bên ngoài (gọi API AI, API thời tiết, dịch vụ thông báo cục bộ, chuyển đổi đơn vị, xử lý thành tích)24.
* **states**: Các lớp trạng thái bất biến, được sử dụng bởi các Notifier (ví dụ: HomePageState, ItemDetailState, ProfilePageState)25.
* **theme**: Định nghĩa giao diện chung cho ứng dụng (app\_theme.dart)26.
* **widgets**: Các widget có thể tái sử dụng (ActionCard, CategorySelector, MultiSelectChipField, StatsPieChart, NotificationBanner, QuestMascot)27.

#### 2.4. Thư viện & Dependencies chính

File

pubspec.yaml định nghĩa các thư viện bên thứ ba quan trọng được sử dụng28:

* **Quản lý trạng thái**: flutter\_riverpod, hooks\_riverpod, equatable29.
* **Lưu trữ dữ liệu**: sqflite, path\_provider (CSDL SQLite), shared\_preferences (cài đặt đơn giản)30.
* **Trí tuệ nhân tạo**: google\_generative\_ai (tương tác với Gemini AI), flutter\_dotenv (quản lý biến môi trường)31.
* **Xử lý ảnh & UI/UX**:
  + pro\_image\_editor: Thư viện chính cho chức năng "Xưởng phối đồ" và chỉnh sửa ảnh32.
  + image\_picker, image\_background\_remover, image: Chọn, tách nền, và xử lý ảnh (tạo thumbnail, kiểm tra alpha)33.
  + flutter\_animate: Tạo hiệu ứng động phong phú cho UI34.
  + intl: Hỗ trợ định dạng ngày tháng, số, tiền tệ và đa ngôn ngữ35.
  + webview\_flutter: Hiển thị nội dung web (chính sách bảo mật, điều khoản)36.
  + table\_calendar: Hiển thị lịch37.
  + fl\_chart: Vẽ biểu đồ thống kê38.
  + dotted\_border: Tạo đường viền nét đứt cho placeholder39.
  + material\_symbols\_icons, font\_awesome\_flutter: Bộ icon mở rộng40.
* **Chia sẻ**: share\_plus (chia sẻ ảnh, văn bản)41.
* **Dịch vụ ngoài**:
  + geolocator, geocoding: Lấy vị trí cho chức năng thời tiết42.
  + http: Thực hiện các cuộc gọi mạng đến API43.
  + flutter\_local\_notifications, timezone, flutter\_timezone: Quản lý thông báo cục bộ44.
  + permission\_handler: Yêu cầu quyền truy cập của thiết bị45.
* **Giám sát lỗi**: sentry\_flutter46.

### 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, name, category, color, imagePath, thumbnailPath (đường dẫn ảnh thu nhỏ), closetId, season, occasion, material, pattern, isFavorite, price, và notes47.
  + Closet: Đại diện cho một tủ đồ, chứa id, name, iconName, và colorHex48.
  + Outfit: Đại diện cho một bộ đồ, chứa id, name, imagePath, thumbnailPath, danh sách ID vật phẩm (itemIds), trạng thái isFixed (bộ đồ cố định), và lastWornDate (ngày mặc lần cuối)49.
  + WearLog: Ghi lại mỗi lần một vật phẩm hoặc bộ đồ được mặc.
  + Quest, Achievement, Badge: Mô hình hóa hệ thống nhiệm vụ và thành tích.
  + CitySuggestion, NotificationSettings: Các mô hình dữ liệu khác.
* **Data Sources**:
  + DatabaseHelper (db\_helper.dart): Là một lớp singleton quản lý việc khởi tạo CSDL SQLite, tạo bảng (bao gồm các cột thumbnailPath và lastWornDate mới), và thực hiện các thao tác CRUD (tạo, đọc, cập nhật, xóa) cho closets, clothing\_items, outfits, và wear\_log50.
  + 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, trạng thái hoàn thành onboarding/permissions, và tiến độ nhiệm vụ51.

#### 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 khai52. Các phương thức trong Repository thường trả về

FutureEither<Failure, T> để xử lý kết quả thành công hoặc thất bại một cách nhất quán.

* 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, hỗ trợ lọc và tìm kiếm vật phẩm, xử lý đa chọn và di chuyển vật phẩm, cập nhật lastWornDate cho outfit)53.
* WeatherRepository, SuggestionRepository, CityRepository: Đóng gói các cuộc gọi đến WeatherService và SuggestionService, bao gồm cả chức năng tìm kiếm thành phố54.
* WearLogRepository: Quản lý nhật ký mặc đồ, thêm/xóa log và lấy log theo khoảng thời gian.
* SettingsRepository: Đóng gói việc lưu/đọc các cài đặt người dùng từ SharedPreferences.
* QuestRepository, AchievementRepository: Cung cấp các phương thức để tương tác với QuestService và AchievementService.

#### 3.3. Tầng Nghiệp vụ (Domain Layer)

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

Use Cases55. Các Use Case thường không phụ thuộc vào tầng Presentation hoặc Data Source trực tiếp, mà thông qua các interfaces của Repository.

* AnalyzeItemUseCase: Nhận một file ảnh, gọi đến ClassificationService để phân tích và trả về các thuộc tính của vật phẩm (tên, danh mục, màu sắc, chất liệu, họa tiết)56.
* GetOutfitSuggestionUseCase: Điều phối việc lấy thông tin thời tiết (tự động hoặc thủ công), danh sách vật phẩm và bộ đồ cố định, sau đó gọi đến SuggestionService để tạo ra một gợi ý trang phục hoàn chỉnh từ AI, bao gồm cả validation dữ liệu đầu vào57.
* SaveOutfitUseCase: Xử lý việc lưu ảnh của bộ đồ (bao gồm cả ảnh thu nhỏ PNG), lưu thông tin vào CSDL, và kiểm tra các ràng buộc cho "bộ đồ cố định"58.
* ValidateItemNameUseCase, ValidateRequiredFieldsUseCase: Các Use Case dùng để xác thực dữ liệu đầu vào, ví dụ ValidateItemNameUseCase kiểm tra tên vật phẩm có bị trùng không, ValidateRequiredFieldsUseCase kiểm tra các trường bắt buộc59.
* DeleteMultipleItemsUseCase, MoveMultipleItemsUseCase: Xử lý logic xóa và di chuyển nhiều vật phẩm.
* GetClosetInsightsUseCase: Tổng hợp dữ liệu từ các Repository để tính toán và trả về các thống kê chi tiết về tủ đồ (giá trị, tần suất mặc, chi phí mỗi lần mặc, vật phẩm bị lãng quên).

#### 3.4. Tầng Trình bày (Presentation Layer)

* **States**: Các lớp trạng thái (ví dụ: HomePageState, OutfitBuilderState, ItemDetailState, ProfilePageState, ClosetInsightsState) là các lớp bất biến chứa dữ liệu để vẽ nên một màn hình60.
* **Notifiers**: Các lớp StateNotifier (ví dụ: HomePageNotifier, ItemDetailNotifier, BatchAddItemNotifier, OutfitsHubNotifier, CalendarNotifier, ClosetInsightsNotifier, ProfilePageNotifier, QuestMascotNotifier, NotificationSettingsNotifier) chứa logic giao diện, nhận tương tác từ người dùng, gọi Use Case/Repository, và tạo ra một State mới để cập nhật UI61. Chúng cũng sử dụng

StateProvider như một "kênh sự kiện" (itemChangedTriggerProvider, completedQuestProvider, unlockedAchievementProvider, itemDetailErrorProvider, batchItemDetailErrorProvider, profileChangedTriggerProvider) để thông báo các thay đổi giữa các notifier không liên quan trực tiếp đến state.

* **UI (Screens/Widgets)**:
  + MainScreen: Widget chính chứa NavigationBar và quản lý việc chuyển đổi giữa 4 trang chính (HomePage, ClosetsPage, OutfitsHubPage, ProfilePage). Nó cũng tích hợp

ShowCaseWidget cho tutorial và GlobalUiScope cho các UI toàn cục62.

* + OutfitBuilderPage: Màn hình "Xưởng phối đồ" nơi người dùng có thể tự do kết hợp các vật phẩm trên canvas (ProImageEditor)63.
  + ItemDetailScreen / BatchAddItemScreen: Màn hình thêm/sửa vật phẩm, hỗ trợ phân loại AI và chỉnh sửa ảnh.
  + ClosetsPage / ClosetDetailPage: Quản lý danh sách tủ đồ và vật phẩm bên trong, hỗ trợ đa chọn và các hành động hàng loạt.
  + CalendarPage: Nhật ký phong cách, cho phép ghi lại outfit đã mặc và xem lịch sử.
  + QuestsPage / BadgeDetailPage: Hiển thị nhiệm vụ và thành tích.
  + SettingsPage / EditProfileScreen: Quản lý cài đặt và thông tin cá nhân, hỗ trợ các đơn vị đo lường và định dạng số khác nhau.
  + AnalysisLoadingScreen: Màn hình loading khi AI đang phân tích ảnh.
  + BackgroundRemoverPage, ImageEditorScreen, AvatarCropperScreen: Các màn hình hỗ trợ chỉnh sửa ảnh.
  + Các widget tái sử dụng: ActionCard, RecentItemCard, SectionHeader, StatsOverviewCard, StatsPieChart, MultiSelectChipField, CategorySelector, NotificationBanner, QuestMascot, SpeechBubble.

### 4. Phân tích Chức năng Chi tiết

#### 4.1. Thêm đồ & Phân loại bằng AI

* **Luồng hoạt động**:
  1. Người dùng nhấn nút thêm và chọn ảnh (tối đa 10 ảnh một lần)64.
  2. Ứng dụng điều hướng đến

AnalysisLoadingScreen, nơi BatchAddItemNotifier gọi analyzeAllImages65.

* 1. analyzeAllImages gửi đồng thời tất cả các ảnh đã chọn đến Gemini AI thông qua AnalyzeItemUseCase và ClassificationService. Trong quá trình này,

AnalysisLoadingScreen hiển thị một vòng xoay vô định (indefinite progress indicator) thay vì thanh tiến trình, cùng với thông báo trạng thái "Preparing images..." hoặc "Pre-filling information..."66.

* 1. Sau khi phân tích xong, ứng dụng tự động điều hướng67:
     + ItemDetailScreen nếu có 1 ảnh68.
     + BatchAddItemScreen nếu có nhiều ảnh69.
  2. Các màn hình này đã được điền sẵn dữ liệu từ AI, cho phép người dùng xem lại và chỉnh sửa trước khi lưu70.
* **Thành phần kỹ thuật**:
  1. ClassificationService: Xây dựng câu lệnh (prompt) chi tiết (có hỗ trợ đa ngôn ngữ thông qua PromptStrings.localized), gửi ảnh và prompt đến API của Google Gemini, và xử lý kết quả JSON trả về. Nó xử lý các lỗi mạng và lỗi server thông qua

Either71.

* 1. Prompt được thiết kế để yêu cầu AI trả về các thuộc tính như

name, category, colors, material, pattern theo một cấu trúc JSON định sẵn72.

* 1. BatchAddItemNotifier: Quản lý trạng thái của toàn bộ luồng thêm hàng loạt, từ việc gọi AI song song, tạo các ItemDetailNotifierArgs cho từng ảnh, cho đến việc xác thực (ValidateRequiredFieldsUseCase, ValidateItemNameUseCase) và lưu tất cả vật phẩm vào CSDL thông qua insertBatchItems của ClothingItemRepository. Nó cũng xử lý các thông báo lỗi hoặc thành công thông qua

notificationServiceProvider73.

#### 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 do74.

* **Bố cục**: Màn hình sử dụng Stack để xếp chồng các lớp widget75:
  + **Lớp nền**: ProImageEditor được dùng làm canvas chính để kéo thả và chỉnh sửa vật phẩm, với khả năng thay đổi ảnh nền (\_pickBackgroundImage)76. Nền mặc định là một ảnh trắng, đảm bảo các vật phẩm hiển thị rõ ràng.
  + **Lớp điều khiển**: Một AppBar và thanh công cụ phụ tùy chỉnh chứa các nút Lưu, Undo, Redo77.
  + **Lớp chức năng**: DraggableScrollableSheet chứa ItemBrowserView để hiển thị danh sách vật phẩm (lấy từ itemFilterProvider với ID riêng \_itemBrowserProviderId), cho phép người dùng kéo lên để xem và chọn. Nó cũng hiển thị số lượng item hiện có trên canvas. Khi một item được thêm vào canvas,

\_recalculateItemCounts sẽ cập nhật số lượng78.

* **Tương tác**:
  + Các nút bấm trên UI sẽ tương tác với

ProImageEditor thông qua một GlobalKey79.

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

addLayer() từ \_editorKey, đồng thời lưu trữ ClothingItem tương ứng vào \_itemsOnCanvas80.

* + Việc lưu bộ đồ được thực hiện bằng cách gọi captureEditorImage() từ \_editorKey để lấy ảnh kết quả. Sau đó, một \_SaveOutfitDialog sẽ hiển thị để người dùng nhập tên và chọn có phải là "Bộ đồ cố định" hay không. Cuối cùng, SaveOutfitUseCase sẽ lưu dữ liệu. Nếu là "Bộ đồ cố định", OutfitBuilderNotifier sẽ kiểm tra trùng lặp với các item trong bộ đồ cố định khác trước khi cho phép lưu. Sau khi lưu thành công,

outfitsHubProvider sẽ được invalidate để làm mới danh sách outfit, và một thông báo thành công sẽ hiển thị81.

#### 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 GetOutfitSuggestionUseCase82.

* 1. Use case này sẽ thực hiện các bước sau83:
     + Xác định vị trí người dùng (tự động qua geolocator hoặc thủ công qua shared\_preferences thông qua SettingsRepository). Nếu dữ liệu vị trí thủ công bị thiếu hoặc quyền vị trí bị từ chối, một

Failure rõ ràng sẽ được trả về thay vì dữ liệu mặc định84.

* + - Gọi

WeatherRepository để lấy dữ liệu thời tiết hiện tại dựa trên vị trí đã xác định85.

* + - Gọi

ClothingItemRepository và OutfitRepository để lấy toàn bộ vật phẩm và các "Set Outfit" (bộ đồ cố định)86.

* + - Thực hiện validation: Kiểm tra xem người dùng có đủ số lượng vật phẩm tối thiểu (ví dụ: ít nhất 3 áo và 3 quần/váy) để nhận gợi ý không. Nếu không đủ, một

GenericFailure sẽ được trả về87.

* + - Gọi

SuggestionService, gửi toàn bộ thông tin (thời tiết, tủ đồ, sở thích người dùng, mục đích) trong một prompt chi tiết đến Gemini AI88.

* + - SuggestionService nhận và phân tích phản hồi JSON từ AI, xử lý lỗi API và mạng.
  1. Kết quả SuggestionResult (bao gồm tên outfit, lý do, và danh sách các item được đề xuất) được trả về cho UI để hiển thị một cách trực quan. Nếu có lỗi,

NotificationService sẽ hiển thị banner thông báo89.

* 1. Khi có gợi ý mới, HomePageNotifier sẽ cập nhật backgroundImagePath dựa trên icon thời tiết hiện tại thông qua WeatherImageService. Người dùng có thể nhấn nút Refresh để thay đổi ảnh nền90.
* **Thành phần kỹ thuật**:
  1. Prompt Engineering: Prompt gửi đến AI trong SuggestionService được thiết kế rất chi tiết (có hỗ trợ đa ngôn ngữ), bao gồm thông tin người dùng (giới tính, phong cách, màu yêu thích từ ProfilePageState), ngữ cảnh (thời tiết, địa điểm, mục đích), 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)91.
  2. AI được yêu cầu tuân thủ "Luật phối đồ" và trả về kết quả dưới dạng một đối tượng JSON có cấu trúc nghiêm ngặt92.

### 5. Các Chức năng & Cải tiến khác

#### 5.1. Quản lý Tủ đồ (ClosetsPage, ClosetDetailPage, ClosetFormScreen)

* **Đa chọn & Thao tác hàng loạt**: Các tab "All Items" và "Closet Detail" trong ClosetsPage và ClosetDetailPage hỗ trợ chế độ đa chọn (isMultiSelectMode). Người dùng có thể chọn nhiều vật phẩm để xóa hoặc di chuyển sang tủ đồ khác. Các hành động này được xử lý bởi ItemFilterNotifier và ClosetDetailNotifier, sử dụng DeleteMultipleItemsUseCase và MoveMultipleItemsUseCase để thực hiện giao dịch CSDL hiệu quả.
* **Thêm/Sửa tủ đồ**: ClosetFormScreen cho phép thêm tủ đồ mới hoặc chỉnh sửa tủ đồ hiện có, bao gồm cả việc chọn icon và màu sắc riêng cho từng tủ. Khi thêm tủ đồ mới, ứng dụng sẽ tự động chuyển người dùng đến tab "By Closet" và cuộn xuống tủ đồ vừa tạo.
* **Giới hạn tủ đồ**: Ứng dụng giới hạn số lượng tủ đồ tối đa là 10.

#### 5.2. Nhật ký Phong cách (CalendarPage)

* **Ghi lại trang phục**: Người dùng có thể chọn một ngày trên lịch và ghi lại các vật phẩm hoặc bộ đồ đã mặc.
* **Hiển thị hàng tuần**: HomePage có một WeeklyPlanner hiển thị các outfit đã mặc trong tuần hiện tại.
* **Đa chọn & Xóa log**: CalendarPage cho phép đa chọn các log mặc đồ để xóa nhiều bản ghi cùng lúc.

#### 5.3. Thành tích & Nhiệm vụ (QuestsPage, BadgeDetailPage, QuestMascot)

* **Hệ thống nhiệm vụ**: Ứng dụng có một hệ thống nhiệm vụ theo dõi tiến độ của người dùng (ví dụ: thêm item, nhận gợi ý, tạo outfit). Các nhiệm vụ có thể bị khóa và yêu cầu hoàn thành nhiệm vụ trước đó.
* **Huy hiệu & Thành tích**: Khi một nhóm nhiệm vụ được hoàn thành, người dùng sẽ mở khóa một thành tích và nhận được một huy hiệu tương ứng.
* **Mascot trợ lý**: Một mascot dễ thương (QuestMascot) hiển thị trên màn hình chính, thông báo khi có nhiệm vụ mới hoặc nhiệm vụ hoàn thành. Khi nhấn vào mascot, nó sẽ điều hướng đến trang Nhiệm vụ.
* **Dialog chúc mừng**: Một dialog đặc biệt (AchievementUnlockedDialog) sẽ hiển thị với hiệu ứng đẹp mắt khi một thành tích mới được mở khóa.

#### 5.4. Cài đặt & Hồ sơ (SettingsPage, EditProfileScreen)

* **Cài đặt đơn vị & định dạng số**: Người dùng có thể tùy chỉnh đơn vị cho chiều cao (cm/ft), cân nặng (kg/lbs), nhiệt độ (°C/°F), tiền tệ (VND/USD/EUR), và định dạng số thập phân (dấu chấm/dấu phẩy).
* **Quản lý vị trí**: Người dùng có thể chọn tự động phát hiện vị trí hoặc chọn thủ công một thành phố để lấy dữ liệu thời tiết.
* **Ảnh đại diện**: Người dùng có thể chụp ảnh hoặc chọn từ album để đặt làm ảnh đại diện, sau đó cắt ảnh thành hình tròn.
* **Cài đặt hiển thị**: Bật/tắt ảnh nền thời tiết và hiển thị mascot.
* **Thông báo cục bộ**: Quản lý việc bật/tắt thông báo hàng ngày để nhắc nhở cập nhật nhật ký thời trang.

#### 5.5. Quản lý lỗi & Thông báo chung (NotificationService, GlobalUiScope)

* **Xử lý lỗi nhất quán**: Các lớp Repository và Use Case trả về Either<Failure, T>, cho phép Notifier xử lý lỗi một cách tập trung và hiển thị các thông báo banner thống nhất qua NotificationService.
* **GlobalUiScope**: Widget này được đặt ở tầng cao nhất của cây widget để hiển thị các thành phần UI toàn cục như NotificationBanner và QuestMascot, đảm bảo chúng luôn nằm trên các màn hình khác.

#### 5.6. UI/UX & Hiệu ứng

* **Hoạt ảnh điều hướng**: Các màn hình quan trọng sử dụng AnimatePageRoute để tạo hiệu ứng chuyển trang mượt mà (trượt ngang và mờ dần).
* **Hiệu ứng chung**: Thư viện flutter\_animate được sử dụng rộng rãi để tạo các hiệu ứng fadeIn, slide, scale, shimmer cho các thành phần UI như thẻ vật phẩm, biểu đồ, dialog.
* **Dialog tùy chỉnh**: showAnimatedDialog cung cấp hiệu ứng zoom và fade khi hiển thị dialog.
* **Giao diện thân thiện**: Sử dụng ThemeData để định nghĩa màu sắc, font chữ, và kiểu dáng cho toàn bộ ứng dụng, đảm bảo sự nhất quán và hấp dẫn.

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 cấp93. Nếu bạn cần làm rõ thêm bất kỳ phần nào, đừng ngần ngại hỏi nhé94!