### Tài liệu Kỹ thuật – Ứng dụng MinCloset

Phiên bản: 1.0

Ngày: 19 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 & Offline-First)

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/Service (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.

Ngoài ra, ứng dụng được xây dựng theo kiến trúc **Offline-First**, ưu tiên trải nghiệm ngoại tuyến. Toàn bộ dữ liệu người dùng được lưu và xử lý trực tiếp trên thiết bị (SQLite), đảm bảo ứng dụng hoạt động nhanh và đầy đủ chức năng ngay cả khi không có kết nối mạng. Các dịch vụ đám mây (Firebase) đóng vai trò là một lớp thứ cấp cho việc sao lưu và phục hồi dữ liệu theo yêu cầu của người dùng.

##### 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, backupServiceProvider)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 lý 15:

* **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, auth\_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 sao lưu/phục hồi)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**:
  + **Cục bộ**: sqflite, path\_provider (CSDL SQLite), shared\_preferences (cài đặt đơn giản)30.
  + **Đám mây**: firebase\_core, firebase\_auth, cloud\_firestore, firebase\_storage, google\_sign\_in (Xác thực và sao lưu/phục hồi).
* **Trí tuệ nhân tạo**: google\_generative\_ai (tương tác với Gemini AI)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 47,

Closet 48,

Outfit 49,

WearLog 50,

Quest, Achievement, Badge 51,

CitySuggestion, NotificationSettings52.

* **Data Sources**:
  + **DatabaseHelper (SQLite)**: Là nguồn dữ liệu chính (source of truth) cho mọi hoạt động hàng ngày của ứng dụng, quản lý việc khởi tạo CSDL, tạo bảng và thực hiện các thao tác CRUD cho toàn bộ dữ liệu người dùng53.
  + **SharedPreferences**: Lưu trữ các cài đặt của người dùng như thông tin cá nhân, lựa chọn ngôn ngữ, cài đặt vị trí, và trạng thái hoàn thành onboarding/permissions54.
  + **Firebase (Firestore & Storage)**: Đóng vai trò là nguồn dữ liệu từ xa (remote data source) cho mục đích sao lưu và phục hồi.

##### 3.2. Tầng Repository

Tầng này đóng vai trò là một API dữ liệu, trừu tượng hóa nguồn gốc dữ liệu55.

* **AuthRepository**: Lớp mới, chịu trách nhiệm giao tiếp với Firebase Authentication, xử lý logic đăng nhập với Google và đăng xuất. Cung cấp một Stream để theo dõi trạng thái đăng nhập của người dùng.
* **Các Repository khác**: ClosetRepository, ClothingItemRepository, OutfitRepository, WeatherRepository, SuggestionRepository, v.v. cung cấp các phương thức để tương tác với các Data Source tương ứng, trả về kết quả dưới dạng Either<Failure, T> 56.

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

Tầng này chứa logic nghiệp vụ cốt lõi57.

* **Services**:
  + **BackupService**: Chứa logic phức tạp cho việc sao lưu, điều phối việc đọc dữ liệu từ các repository SQLite, tải file ảnh lên Cloud Storage, và ghi metadata lên Firestore.
  + **RestoreService**: Chứa logic phức tạp cho việc phục hồi, điều phối việc xóa dữ liệu cục bộ, tải metadata từ Firestore, tải ảnh từ Storage, và ghi lại toàn bộ dữ liệu vào SQLite.
* **Use Cases**: AnalyzeItemUseCase, GetOutfitSuggestionUseCase, SaveOutfitUseCase, và các use case xác thực, xóa, di chuyển dữ liệu vẫn giữ vai trò xử lý các logic nghiệp vụ cụ thể và độc lập 58.

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

* **States**: Các lớp trạng thái bất biến (HomePageState, ItemDetailState, ProfilePageState, v.v.) chứa dữ liệu để vẽ nên giao diện59.
* **Notifiers**: Các lớp StateNotifier (HomePageNotifier, ItemDetailNotifier, ProfilePageNotifier, v.v.) chứa logic giao diện, nhận tương tác từ người dùng, gọi Use Case/Service/Repository, và tạo ra một State mới để cập nhật UI60. Chúng cũng sử dụng các

StateProvider như một "kênh sự kiện" (itemChangedTriggerProvider, completedQuestProvider) để thông báo các thay đổi giữa các notifier61.

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

ShowCaseWidget cho tutorial và GlobalUiScope cho các UI toàn cục 62.

* + **OnboardingScreen**: Được cập nhật để cung cấp hai luồng cho người dùng mới: (1) Nhập tên để sử dụng cục bộ hoặc (2) Đăng nhập bằng tài khoản Google. Nếu người dùng đăng nhập và có bản sao lưu, ứng dụng sẽ đề xuất phục hồi dữ liệu.
  + **EditProfileScreen**: Giờ đây là trung tâm quản lý tài khoản. Nó hiển thị giao diện động: nếu chưa đăng nhập, sẽ có nút đăng nhập; nếu đã đăng nhập, sẽ hiển thị email, thông tin sao lưu và các nút chức năng "Sao lưu", "Phục hồi", "Đăng xuất".
  + Các màn hình và widget khác vẫn giữ vai trò như đã mô tả trước đây 63.

#### 4. Phân tích Chức năng Chi tiết

##### 4.1. Xác thực, Sao lưu & Phục hồi

Đây là một tính năng mới, được xây dựng theo kiến trúc **"Offline-First with Manual Backup/Restore"**.

* **Kiến trúc:** Ứng dụng ưu tiên hoạt động ngoại tuyến. SQLite là nguồn dữ liệu chính cho mọi thao tác hàng ngày, đảm bảo hiệu suất và khả năng truy cập không phụ thuộc vào mạng. Firebase chỉ đóng vai trò là một dịch vụ sao lưu/phục hồi khi người dùng có tài khoản và chủ động yêu cầu.
* **Luồng Xác thực:**
  1. Người dùng có thể bỏ qua việc đăng nhập ở màn hình Onboarding để dùng ứng dụng ngay lập tức với dữ liệu lưu trên máy.
  2. Họ cũng có thể đăng nhập bằng Google ở màn hình Onboarding hoặc sau này trong trang EditProfileScreen.
  3. Trạng thái đăng nhập được quản lý toàn cục bởi authStateChangesProvider.
* **Luồng Sao lưu (Backup Flow):**
  1. Người dùng đã đăng nhập nhấn nút "Backup" trên trang EditProfileScreen.
  2. BackupService được kích hoạt. Nó gửi lệnh xóa toàn bộ dữ liệu cũ của người dùng này trên Cloud Storage và Firestore để đảm bảo bản sao lưu mới là một bản "phản chiếu" chính xác.
  3. Service đọc toàn bộ dữ liệu (items, outfits, closets, wear logs) từ các repository SQLite.
  4. Với mỗi vật phẩm/trang phục, nó tải file ảnh từ đường dẫn cục bộ lên thư mục riêng của người dùng trên Cloud Storage.
  5. Sau khi có URL ảnh mới từ cloud, nó ghi toàn bộ dữ liệu metadata (với URL ảnh mới) lên các sub-collection tương ứng trong document của người dùng trên Firestore.
  6. Cuối cùng, nó cập nhật một dấu thời gian lastBackup.
* **Luồng Phục hồi (Restore Flow):**
  1. Người dùng nhấn "Restore" trên EditProfileScreen hoặc trong luồng Onboarding.
  2. Ứng dụng hiển thị một hộp thoại cảnh báo việc ghi đè dữ liệu.
  3. RestoreService được kích hoạt. Nó xóa sạch toàn bộ dữ liệu cục bộ trong SQLite và các thư mục ảnh.
  4. Service tải toàn bộ metadata từ các sub-collection của người dùng trên Firestore.
  5. Với mỗi bản ghi metadata, nó dùng URL ảnh để tải file ảnh từ Cloud Storage về và lưu vào bộ nhớ cục bộ của thiết bị.
  6. Cuối cùng, nó ghi toàn bộ metadata đã tải về vào SQLite, đồng thời cập nhật các trường imagePath bằng đường dẫn file cục bộ mới.
  7. Ứng dụng sau đó được làm mới để hiển thị dữ liệu vừa được phục hồi.

##### 4.2. Thêm đồ & Phân loại bằng AI

* **Luồng hoạt động**: Người dùng chọn ảnh (tối đa 10). Ứng dụng hiển thị màn hình AnalysisLoadingScreen và gửi đồng thời tất cả ảnh đến Gemini AI để phân tích. Sau khi hoàn tất, ứng dụng tự động điều hướng đến

ItemDetailScreen (1 ảnh) hoặc BatchAddItemScreen (nhiều ảnh) với các thông tin đã được điền sẵn 64.

* **Thành phần kỹ thuật**: ClassificationService xây dựng prompt chi tiết và gửi đến Gemini AI. BatchAddItemNotifier quản lý toàn bộ luồng, từ việc gọi AI song song đến việc xác thực và lưu hàng loạt vào CSDL.

##### 4.3. Xưởng phối đồ (OutfitBuilderPage)

* **Chức năng**: Cho phép người dùng sáng tạo các bộ đồ bằng cách kết hợp vật phẩm một cách tự do trên một canvas.
* **Bố cục**: Sử dụng Stack để xếp chồng ProImageEditor (làm canvas chính), các thanh công cụ, và một DraggableScrollableSheet chứa ItemBrowserView để người dùng chọn vật phẩm 65.
* **Tương tác**: Các nút bấm tương tác với ProImageEditor thông qua một GlobalKey. Việc lưu bộ đồ sẽ chụp lại ảnh từ canvas, hiển thị

\_SaveOutfitDialog để nhập tên, và sau đó SaveOutfitUseCase sẽ xử lý việc lưu trữ 66.

##### 4.4. Gợi ý Trang phục hàng ngày

* **Luồng hoạt động**: HomePageNotifier gọi GetOutfitSuggestionUseCase. Use case này sẽ lấy vị trí, thời tiết, toàn bộ tủ đồ của người dùng, thực hiện validation, và cuối cùng gọi

SuggestionService để gửi một prompt chi tiết đến Gemini AI 67.

* **Thành phần kỹ thuật**: Prompt Engineering đóng vai trò quan trọng. Prompt được thiết kế chi tiết để AI hiểu rõ ngữ cảnh, thông tin người dùng, và danh sách tủ đồ (chia thành "Set Outfit" và "Vật phẩm lẻ") để đưa ra gợi ý tốt nhất và trả về kết quả dưới dạng JSON có cấu trúc 68.

#### 5. Các Chức năng & Cải tiến khác

* **5.1. Quản lý Tủ đồ**: Hỗ trợ đa chọn để xóa/di chuyển vật phẩm hàng loạt. Cho phép thêm/sửa tủ đồ với icon và màu sắc tùy chỉnh 69.
* 5.2. Nhật ký Phong cách

: Ghi lại trang phục đã mặc, hiển thị tóm tắt hàng tuần trên trang chủ 70.

* **5.3. Thành tích & Nhiệm vụ**: Hệ thống nhiệm vụ theo dõi tiến độ người dùng, mở khóa thành tích và huy hiệu. Một mascot trợ lý thông báo các sự kiện quan trọng 71.
* 5.4. Cài đặt & Hồ sơ

: Tùy chỉnh đơn vị đo lường, định dạng số, quản lý vị trí, ảnh đại diện, và cài đặt thông báo 72.

* **5.5. Quản lý lỗi & Thông báo chung**: Sử dụng Either<Failure, T> và NotificationService để xử lý lỗi một cách nhất quán.

GlobalUiScope hiển thị các UI toàn cục 73.

* 5.6. UI/UX & Hiệu ứng

: Sử dụng AnimatePageRoute và thư viện flutter\_animate để tạo các hiệu ứng chuyển trang và hoạt ảnh phong phú, mang lại trải nghiệm người dùng mượt mà và hiện đại 74.