Chào bạn,

Chắc chắn rồi. Việc cập nhật tài liệu kỹ thuật để nó phản ánh đúng trạng thái hiện tại của dự án là một công việc cực kỳ quan trọng. Dựa trên tất cả những thay đổi và cải tiến mà chúng ta đã cùng nhau thực hiện, tôi đã cập nhật lại toàn bộ tài liệu.

Các phần được cập nhật chi tiết bao gồm việc tích hợp hoàn chỉnh Riverpod, triển khai Repository Pattern, và áp dụng tầng Use Case. Lộ trình phát triển cũng được làm mới để phản ánh các mục tiêu tiếp theo.

Dưới đây là phiên bản mới, chi tiết và đầy đủ của tài liệu.

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

* **Phiên bản:** 2.0
* **Ngày:** 17 tháng 6 năm 2025
* **Tác giả:** MinVN

**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. Cấu trúc Thư mục
3. **Phân tích Chức năng Chi tiết**
   * 3.1. Quản lý Tủ đồ & Vật phẩm
   * 3.2. Gợi ý Trang phục bằng AI
   * 3.3. Xưởng Phối đồ & Trình duyệt Vật phẩm
   * 3.4. Thư viện Bộ đồ đã lưu
   * 3.5. Trang Cá nhân & Cài đặt
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. **Hướng dẫn Cài đặt & Môi trường Dev**
6. **Lộ trình Phát triển & Đề xuất**
7. **Phụ lục: Sơ đồ Cơ sở dữ liệu**

**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 MinCloset1111. Mục tiêu là làm cơ sở cho việc bảo trì, mở rộng và chuyển giao dự án cho các thành viên trong nhóm phát triển2222.

**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 minh3. Ứng dụng cho phép người dùng quản lý quần áo, tạo các bộ trang phục ảo, và nhận gợi ý phối đồ hàng ngày được cá nhân hóa bởi A.I4.

**2. Kiến trúc Hệ thống**

**2.1. Tổng quan**

Ứng dụng được xây dựng theo kiến trúc đa tầng rõ ràng, tuân thủ các nguyên tắc của Clean Architecture để tối ưu cho việc bảo trì, kiểm thử và mở rộng5. Luồng dữ liệu và logic chính tuân theo mô hình:

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

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

* **Công nghệ:** Dự án sử dụng **Riverpod** làm giải pháp quản lý trạng thái chính6.
* **Triển khai:**
  + StateNotifierProvider: Dùng để quản lý các trạng thái phức tạp và tương tác của người dùng (ví dụ: HomePage, ItemDetailPage, ProfilePage, OutfitBuilderPage)7777.
  + FutureProvider: Dùng để cung cấp dữ liệu bất đồng bộ một lần từ các Repository (ví dụ: closetsProvider, outfitsProvider)8.
  + .family modifier: Được sử dụng để tạo ra các provider có tham số, cho phép các phiên bản state độc lập với nhau (ví dụ: itemDetailProvider, itemFilterProvider).
  + .autoDispose modifier: Được sử dụng cho các provider có trạng thái không cần tồn tại lâu dài (ví dụ: các form, các bộ lọc) để tự động giải phóng bộ nhớ khi không còn được sử dụng.
* **Nguyên tắc:** Giao diện (Widgets) hoàn toàn mang tính khai báo. Chúng chỉ "theo dõi" (ref.watch) trạng thái và "gọi" (ref.read) các phương thức từ Notifier. Toàn bộ logic xử lý được đặt trong Notifier hoặc Use Case9.

**2.3. Tầng Repository**

* **Mục đích:** Tạo một lớp trừu tượng nằm giữa logic nghiệp vụ và nguồn dữ liệu, giúp logic không cần biết dữ liệu được lấy từ đâu (SQLite, API, Cache...). 10
* **Triển khai:** Mỗi domain dữ liệu có một Repository riêng: ClosetRepository11, ClothingItemRepository, OutfitRepository, WeatherRepository, SuggestionRepository.

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

* **Mục đích:** Đóng gói các quy trình nghiệp vụ phức tạp, đặc biệt là những quy trình cần điều phối nhiều Repository.
* **Triển khai:**
  + GetOutfitSuggestionUseCase: Chứa logic phức tạp để lấy thời tiết, lấy danh sách đồ, và gọi A.I để tạo gợi ý.
  + SaveOutfitUseCase: Chứa logic lưu một bộ đồ (lưu ảnh vào bộ nhớ và lưu metadata vào CSDL).

**2.5. Lưu trữ Dữ liệu**

* **CSDL Chính:** Dữ liệu có cấu trúc (tủ đồ, vật phẩm, bộ đồ) được lưu trữ trong CSDL SQLite cục bộ thông qua gói sqflite12121212.
* **Lưu trữ Đơn giản:** Dữ liệu nhỏ, tạm thời (gợi ý A.I gần nhất, cài đặt người dùng) được lưu bằng shared\_preferences13131313.

**2.6. Cấu trúc Thư mục (lib)**

Cấu trúc thư mục được tổ chức theo chức năng và tầng kiến trúc:

* screens/pages/: Các màn hình chính.
* models/: Các lớp đối tượng dữ liệu (POCO)14.
* states/, notifiers/, providers/: Các file liên quan đến Riverpod.
* domain/use\_cases/: Chứa các lớp logic nghiệp vụ chính.
* repositories/: Chứa các lớp Repository.
* services/: Giao tiếp với các dịch vụ bên ngoài (API)15.
* widgets/: Các widget có thể tái sử dụng16.
* helpers/: Chứa DatabaseHelper.

**3. Phân tích Chức năng Chi tiết**

**3.1. Quản lý Tủ đồ & Vật phẩm**

* **Mô tả:** Cho phép người dùng tạo/sửa/xóa tủ đồ 17 và các vật phẩm bên trong18.
* **Thành phần Kỹ thuật:**
  + **UI:** ClosetsPage được thiết kế với 2 tab: "Tất cả vật phẩm" và "Theo Tủ đồ". Tab "Tất cả vật phẩm" sử dụng widget ItemBrowserView dùng chung. ClosetDetailPage hiển thị các vật phẩm trong một tủ đồ cụ thể.
  + **State & Logic:** closetsProvider tải danh sách tủ đồ. ItemFilterNotifier (itemFilterProvider.family('closetsPage')) quản lý việc tìm kiếm và lọc trong tab "Tất cả vật phẩm". Các hành động CRUD đều được thực hiện thông qua các Repository tương ứng.
  + **Luồng Cập nhật:** Sử dụng Navigator.pop(true) và ref.invalidate() để đảm bảo dữ liệu luôn đồng bộ giữa các màn hình sau khi sửa hoặc xóa19.

**3.2. Gợi ý Trang phục bằng AI**

* **Mô tả:** Trang chủ cung cấp gợi ý trang phục dựa trên tủ đồ và vị trí của người dùng20.
* **Thành phần Kỹ thuật:**
  + **Logic:** Logic được đóng gói trong GetOutfitSuggestionUseCase. Use Case này sẽ điều phối WeatherRepository và ClothingItemRepository để lấy dữ liệu đầu vào, sau đó gọi SuggestionRepository để gửi prompt đến Google Gemini21. Tên thành phố được xác định tự động bằng geolocator/geocoding hoặc do người dùng cài đặt, không còn bị gán cứng.
  + **State:** HomePageNotifier gọi Use Case và cập nhật HomePageState. Nó chỉ tải dữ liệu khi người dùng bấm nút yêu cầu, và lưu gợi ý gần nhất vào SharedPreferences để hiển thị khi khởi động lại22.
  + **Bảo mật:** API Keys được quản lý qua file .env và gói flutter\_dotenv23.

**3.3. Xưởng Phối đồ & Trình duyệt Vật phẩm**

* **Mô tả:** Canvas sáng tạo cho phép người dùng kéo thả các vật phẩm để tạo bộ đồ ảo24. Phần chọn đồ đã được nâng cấp với chức năng tìm kiếm và lọc nâng cao.
* **Thành phần Kỹ thuật:**
  + **UI:** OutfitBuilderPage sử dụng Stack làm canvas 25 và widget ClothingSticker tùy chỉnh để xử lý tương tác26.
  + **Thành phần dùng chung:** Phần chọn đồ ở dưới cùng giờ đây sử dụng widget tái sử dụng ItemBrowserView, cung cấp giao diện tìm kiếm và lọc nhất quán với trang Tủ đồ.
  + **State & Logic:** OutfitBuilderNotifier quản lý trạng thái của canvas (các sticker, sticker được chọn). ItemFilterNotifier (itemFilterProvider.family('outfitBuilderPage')) quản lý trạng thái của ItemBrowserView.
  + **Lưu trữ:** SaveOutfitUseCase xử lý logic lưu: dùng gói screenshot 27 để chụp ảnh, path\_provider 28 để lấy đường dẫn, và OutfitRepository để lưu metadata vào CSDL29.

**3.4. Thư viện Bộ đồ đã lưu**

* **Mô tả:** Trang OutfitsHubPage hiển thị thư viện các bộ đồ đã tạo, cho phép xem, xóa, chia sẻ và điều hướng đến "Xưởng phối đồ". 30
* **Thành phần Kỹ thuật:**
  + **UI:** Sử dụng GridView để hiển thị31. Một FloatingActionButton được dùng để điều hướng đến OutfitBuilderPage.
  + **State & Logic:** outfitsProvider (FutureProvider) tải danh sách bộ đồ thông qua OutfitRepository32.
  + **Chia sẻ:** Sử dụng gói share\_plus33.

**3.5. Trang Cá nhân & Cài đặt**

* **Mô tả:** Hiển thị thông tin người dùng, các thống kê về tủ đồ, và các cài đặt ứng dụng.
* **Thành phần Kỹ thuật:**
  + **UI:** ProfilePage, SettingsPage, và dialog CityPreferenceDialog.
  + **State & Logic:** ProfilePageNotifier tải và quản lý tất cả dữ liệu cho trang này (thông tin người dùng từ SharedPreferences và dữ liệu thống kê từ các repository).
  + **Thống kê:** Dữ liệu được tính toán trong Notifier và hiển thị bằng các Text widget và biểu đồ từ gói fl\_chart.
  + **Cài đặt Vị trí:** Chức năng chọn thành phố có khả năng xác thực tên thành phố bằng cách gọi thử API thời tiết thông qua WeatherRepository.

**4. Các Thành phần Cốt lõi**

* **Data Models (lib/models/):** Các lớp ClothingItem34, Closet, Outfit, OutfitFilter định nghĩa cấu trúc dữ liệu. Việc sử dụng Equatable 35 giúp Riverpod phát hiện thay đổi trạng thái một cách chính xác.
* **Database Helper (lib/helpers/db\_helper.dart):** Lớp cấp thấp, là chi tiết triển khai của tầng Repository, chịu trách nhiệm thực thi các câu lệnh SQL36.
* **Reusable Widgets (lib/widgets/):** Các widget như MultiSelectChipField, ItemBrowserView, StatsPieChart được tạo ra để tái sử dụng, giúp giao diện nhất quán và giảm lặp code37.

**5. Hướng dẫn Cài đặt & Môi trường Dev**

* **Yêu cầu:** Flutter SDK (phiên bản 3.22+ được khuyến nghị)38.
* **Cài đặt:** Chạy flutter pub get39.
* **Biến môi trường:** Tạo file .env ở thư mục gốc và cung cấp các khóa API40.
* **Chạy/Test:** flutter run, flutter test.

**6. Lộ trình Phát triển & Đề xuất**

Kiến trúc hiện tại đã rất vững chắc. Các bước phát triển tiếp theo có thể tập trung vào việc xây dựng tính năng mới:

1. **Lịch mặc đồ (Outfit Calendar):** Cho phép người dùng lên kế hoạch trang phục cho tương lai.
2. **Thống kê Nâng cao:** Bổ sung các chỉ số như "Chi phí mỗi lần mặc" (Cost Per Wear).
3. **Đồng bộ Đám mây:** Sử dụng Firebase (Auth, Firestore, Storage) để sao lưu và đồng bộ dữ liệu người dùng. Kiến trúc Repository hiện tại đã sẵn sàng cho việc tích hợp một RemoteDataSource.

**7. Phụ lục: Sơ đồ Cơ sở dữ liệu**

Ứng dụng sử dụng 3 bảng chính trong CSDL sqflite: closets, clothing\_items, outfits với các trường như đã định nghĩa trong DatabaseHelper.\_createDB41.