Chắc chắn rồi. Dựa trên các phân tích và quá trình chúng ta đã làm việc, tôi sẽ tổng hợp và phát triển thành một tài liệu kỹ thuật hoàn chỉnh theo chuẩn.

Tài liệu này sẽ có cấu trúc rõ ràng, chi tiết, phù hợp để chuyển giao cho một nhóm phát triển, giúp họ nhanh chóng nắm bắt và tiếp tục công việc.

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

* **Phiên bản:** 1.0
* **Ngày:** 17 tháng 6 năm 2025
* **Tác giả:** Gemini AI (Dựa trên mã nguồn được cung cấp)

**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. Lưu trữ Dữ liệu: Sqflite
   * 2.4. 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 (Outfit Builder)
   * 3.4. Thư viện Bộ đồ đã lưu
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**
   * 5.1. Yêu cầu
   * 5.2. Cài đặt
   * 5.3. Chạy Ứng dụng & Kiểm thử
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 MinCloset. Mục tiêu là làm cơ sở cho việc bảo trì, mở rộng và chuyển giao dự án cho các thành viên trong nhóm phát triển.

**1.2. Tổng quan Ứng dụng**

MinCloset là ứng dụng di động đa nền tảng (cross-platform) được xây dựng bằng Flutter, hoạt động như một trợ lý tủ đồ thông minh. 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.I.

**2. Kiến trúc Hệ thống**

**2.1. Tổng quan**

Ứng dụng được xây dựng theo kiến trúc phân lớp rõ ràng, tách biệt giữa Giao diện (UI), Logic Nghiệp vụ (Business Logic), và Tầng Dữ liệu (Data Layer) để tối ưu cho việc bảo trì và kiểm thử.

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

* **Công nghệ:** Dự án sử dụng **Riverpod** làm giải pháp quản lý trạng thái chính.
* **Triển khai:**
  + **FutureProvider**: Dùng để cung cấp dữ liệu bất đồng bộ một lần từ CSDL (ví dụ: closetsProvider, outfitsProvider).
  + **StateNotifierProvider**: Dùng để quản lý các trạng thái phức tạp và tương tác của người dùng. Mỗi màn hình có logic riêng sẽ có một cặp State (định nghĩa dữ liệu) và Notifier (xử lý logic). Ví dụ: HomePage, ItemDetailPage.
  + **Nguyên tắc:** Giao diện (Widgets) 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.

**2.3. Lưu trữ Dữ liệu (Data Persistence)**

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

**2.4. Cấu trúc Thư mục (lib)**

Cấu trúc thư mục được tổ chức để tối ưu cho khả năng mở rộng và tìm kiếm:

* screens/: Các màn hình chính.
* models/: Các lớp đối tượng dữ liệu (POCO/POJO).
* providers/, notifiers/, states/: Các file liên quan đến Riverpod.
* services/: Giao tiếp với các dịch vụ bên ngoài (APIs).
* widgets/: Các widget tái sử dụng.
* constants/: Các giá trị không đổi.
* helpers/: Các lớp trợ giúp, chủ yếu là DatabaseHelper.

**3. Phân tích Chức năng Chi tiết**

**3.1. Chức năng: Quản lý Tủ đồ & Vật phẩm**

* **Mô tả:** Người dùng có thể tạo/sửa/xóa tủ đồ. Trong mỗi tủ đồ, người dùng có thể thêm/sửa/xóa các vật phẩm quần áo.
* **Thành phần Kỹ thuật:**
  + **UI:** ClosetsPage, ClosetDetailPage, ItemDetailPage, AddItemScreen.
  + **State & Logic:**
    - closetsProvider, itemsInClosetProvider: Tải và hiển thị danh sách.
    - addItemProvider, itemDetailProvider: Quản lý trạng thái và logic cho từng form/màn hình chi tiết.
  + **Luồng Cập nhật:** Khi có thay đổi ở màn hình con (ví dụ ItemDetailPage), nó sẽ pop về một giá trị true. Màn hình cha (ClosetDetailPage) sẽ nhận giá trị này và gọi ref.invalidate() để làm mới lại dữ liệu từ CSDL, đảm bảo giao diện luôn đồng bộ.

**3.2. Chức năng: Gợi ý Trang phục bằng AI**

* **Mô tả:** Trang chủ cung cấp gợi ý trang phục dựa trên tủ đồ của người dùng và thời tiết hiện tại.
* **Thành phần Kỹ thuật:**
  + **Services:**
    - WeatherService: Gọi API OpenWeatherMap để lấy dữ liệu thời tiết.
    - SuggestionService: Soạn prompt và gọi API Google Gemini.
  + **State & Logic:** HomePageNotifier điều phối toàn bộ luồng hoạt động. Nó không tự động chạy khi vào trang mà chỉ kích hoạt khi người dùng yêu cầu, giúp tiết kiệm tài nguyên. Gợi ý gần nhất được lưu vào SharedPreferences để hiển thị ngay khi người dùng mở lại ứng dụng.
  + **Bảo mật:** API Keys được quản lý qua file .env và gói flutter\_dotenv.

**3.3. Chức năng: Xưởng Phối đồ (Outfit Builder)**

* **Mô tả:** Canvas sáng tạo cho phép người dùng kéo thả các vật phẩm để tạo bộ đồ ảo.
* **Thành phần Kỹ thuật:**
  + **UI:** OutfitBuilderPage sử dụng Stack làm canvas. Widget ClothingSticker tùy chỉnh sử dụng GestureDetector để xử lý các thao tác di chuyển, xoay, và thay đổi kích thước.
  + **Lưu trữ:** Gói screenshot được dùng để chụp lại canvas. File ảnh được lưu vào bộ nhớ thiết bị bằng path\_provider. Metadata được lưu vào bảng outfits trong CSDL.
  + **State Management:** *(Hiện trạng)* Chức năng này vẫn đang dùng StatefulWidget và setState. Đây là mục tiêu cần refactor sang Riverpod để đảm bảo tính nhất quán.

**3.4. Chức năng: Thư viện Bộ đồ đã lưu**

* **Mô tả:** Hiển thị thư viện các bộ đồ đã tạo, cho phép xem, xóa và chia sẻ.
* **Thành phần Kỹ thuật:**
  + **UI:** SavedOutfitsPage sử dụng GridView.
  + **State & Logic:** outfitsProvider (FutureProvider) tải danh sách bộ đồ từ CSDL.
  + **Chia sẻ:** Sử dụng gói share\_plus.

**4. Các Thành phần Cốt lõi**

* **Data Models (lib/models/):** Các lớp ClothingItem, Closet, Outfit định nghĩa cấu trúc dữ liệu. ClothingItem kế thừa từ Equatable để đảm bảo việc so sánh và phát hiện thay đổi trong Riverpod là chính xác và đáng tin cậy.
* **Database Helper (lib/helpers/db\_helper.dart):** Lớp cấp thấp chịu trách nhiệm thực thi các câu lệnh SQL thô (CRUD) với CSDL sqflite.
* **Reusable Widgets (lib/widgets/):** Các widget như MultiSelectChipField và SectionHeader được tạo ra để tái sử dụng, giúp giao diện nhất quán và giảm lặp code.

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

1. **Yêu cầu:** Flutter SDK (phiên bản 3.22 trở lên được khuyến nghị).
2. **Cài đặt:** Chạy flutter pub get từ thư mục gốc để cài đặt tất cả các phụ thuộc.
3. **Biến môi trường:**
   * Tạo một file tên là .env ở thư mục gốc của dự án.
   * Thêm các khóa API vào file này:
   * GEMINI\_API\_KEY=YOUR\_GEMINI\_API\_KEY
   * OPENWEATHER\_API\_KEY=YOUR\_OPENWEATHER\_API\_KEY
4. **Chạy Ứng dụng:** flutter run
5. **Chạy Kiểm thử:** flutter test

**6. Lộ trình Phát triển & Đề xuất**

1. **Hoàn thiện Refactor Riverpod:** Ưu tiên hàng đầu là refactor màn hình OutfitBuilderPage sang StateNotifierProvider để đồng bộ kiến trúc toàn ứng dụng.
2. **Triển khai Repository Pattern:** Tạo một tầng Repository nằm giữa Notifier và DatabaseHelper. Ví dụ, tạo ClosetRepository chịu trách nhiệm mọi logic liên quan đến tủ đồ. Điều này giúp trừu tượng hóa nguồn dữ liệu và tăng cường khả năng kiểm thử.
3. **Xây dựng Design System:** Chuẩn hóa các style (màu sắc, nút bấm, thẻ...) bằng ThemeData và ThemeExtension để dễ dàng quản lý và thay đổi giao diện.

**7. Phụ lục: Sơ đồ Cơ sở dữ liệu**

Ứng dụng sử dụng 3 bảng chính trong CSDL sqflite:

* **Bảng closets:**
  + id (TEXT, PRIMARY KEY)
  + name (TEXT)
* **Bảng clothing\_items:**
  + id (TEXT, PRIMARY KEY)
  + name (TEXT)
  + category (TEXT)
  + color (TEXT)
  + imagePath (TEXT)
  + closetId (TEXT, Foreign Key tham chiếu đến closets.id)
  + season, occasion, material, pattern (TEXT, nullable)
* **Bảng outfits:**
  + id (TEXT, PRIMARY KEY)
  + name (TEXT)
  + imagePath (TEXT)
  + itemIds (TEXT, chuỗi các ID của clothing\_items, ngăn cách bởi dấu phẩy)