Tổng hợp mã nguồn dự án: SmartBloodDonationAndroid

.gitignore

```
*.iml
.gradle
/local.properties
/.idea/caches
/.idea/libraries
/.idea/modules.xml
/.idea/workspace.xml
/.idea/navEditor.xml
/.idea/assetWizardSettings.xml
.DS_Store
/build
/captures
.externalNativeBuild
.cxx
local.properties
```

README.md

```
# SmartBloodDonation
```

SmartBloodDonation/
├── build.gradle.kts   // File build Gradle của project
├── settings.gradle.kts   // Khai báo các module của project
├── gradle/
│
├── app/   // Module chính, nơi ghép nối các module feature
│   ├── build.gradle.kts
│   └── src/main/
│   ├── java/com/smartblood/donation/ // <- Package name của app
│   │   ├── MainApplication.kt
│   │   ├── MainActivity.kt
│   │   ├─ di/
│   │   │   └── AppModule.kt
│   │   ├─ features/   // **THÊM MỚI: Chứa các màn hình của app**
│   │   │   └─ dashboard/
│   │   │   ├─ DashboardScreen.kt // **UI của Dashboard**
│   │   │   └─ DashboardViewModel.kt // **ViewModel của Dashboard**
```
```

```
│       │   ├─ navigation/          // Quản lý điều hướng toàn ứng dụng
│       │   │ ├─ AppNavHost.kt
│       │   │ ├─ BottomNavItem.kt    // **THÊM MỚI: Định nghĩa các mục cho Bottom Nav**
│       │   │ └─ Screen.kt
│       │   └─ ui/                 // **THÊM MỚI: Chứa các Composable dùng chung của app**
│       │     └─ MainScreen.kt       // **Màn hình chính chứa Bottom Nav và NavHost con**
│       └─── res/
│
│
├─── core/                         // Module lõi chứa code dùng chung
│   ├─── build.gradle.kts
│   └─── src/main/java/com/smartblood/core/
│       ├─── data/
│       │   ├─── local/
│       │   │   └─── AppDatabase.kt   // Lớp trừu tượng của Room DB
│       │   └─── network/
│       │       ├─── ApiClient.kt     // Cấu hình Retrofit, OkHttp
│       │       └─── AuthInterceptor.kt
│       ├─── domain/
│       │   └─── model/
│       │       └─── Result.kt       // Lớp Result wrapper chung (Success, Error)
│       ├─── ui/
│       │   ├─── components/         // Các Composable dùng chung toàn app
│       │   │   ├─── LoadingDialog.kt
│       │   │   ├─── ErrorMessage.kt
│       │   │   └─── PrimaryButton.kt
│       │   └─── theme/             // Theme, Color, Typography, Shape
│       │       ├─── Color.kt
│       │       ├─── Shape.kt
│       │       ├─── Theme.kt
│       │       └─── Type.kt
│       └─── util/                 // Các lớp tiện ích, extensions
│           ├─── Constants.kt
│           └─── extensions/
│               └─── StringExt.kt
│
├─── feature_auth/                 // Module tính năng: Xác thực
│   ├─── build.gradle.kts
│   └─── src/main/java/com/smartblood/auth/
│       ├─── data/
│       │   ├─── local/             // Dữ liệu cục bộ (ví dụ: lưu session token)
│       │   │   └─── AuthLocalDataSource.kt
│       │   ├─── mapper/            // Ánh xạ giữa DTO -> Domain Model
│       │   │   └── UserMapper.kt
│       │   ├── remote/
│       │   │   ├── AuthApiService.kt // Interface Retrofit/Firebase function
│       │   │   └── dto/           // Data Transfer Objects
│       │   │     ├── LoginRequestDto.kt
│       │   │     └── UserDto.kt
│       │   └── repository/
│       │     └── AuthRepositoryImpl.kt // Implement interface từ Domain
│       ├── domain/
│       │   ├── model/             // Model sạch, chỉ chứa logic nghiệp vụ
│       │   │   └── User.kt
│       │   ├── repository/
│       │   │   └── AuthRepository.kt  // Interface (Hợp đồng) cho repository
│       │   └── usecase/           // Các trường hợp sử dụng cụ thể
│       │     ├── LoginUseCase.kt
│       │     ├── RegisterUseCase.kt
│       │     └── PerformFaceAuthUseCase.kt
│       ├── di/                    // DI cho module auth
│       │   └── AuthModule.kt
│       └── ui/
│         ├── navigation/         // Điều hướng trong feature
│         │   └── AuthNavigation.kt
│         ├── login/
│         │   ├── LoginScreen.kt
│         │   ├── LoginViewModel.kt
│         │   └── LoginContract.kt // Định nghĩa State, Event, Effect
│         └── register/
│           ├── RegisterScreen.kt
│           ├── RegisterViewModel.kt
│           └── RegisterContract.kt
│         └─ splash/               // **THÊM MỚI: Màn hình Splash**
│           ├─ SplashScreen.kt
│           └─ SplashViewModel.kt
│
├── feature_profile/               // Module tính năng: Hồ sơ
│   ├── build.gradle.kts
│   └── src/main/java/com/smartblood/profile/
│       ├── data/
│       │   ├── mapper/
│       │   │   └── DonationHistoryMapper.kt
│       │   ├── remote/...
│       │   └── repository/
│       │     └── ProfileRepositoryImpl.kt
│      ├── domain/
│      │   ├── model/
│      │   │   ├── UserProfile.kt
│      │   │   └── DonationRecord.kt
│      │   ├── repository/
│      │   │   └── ProfileRepository.kt
│      │   └── usecase/
│      │     ├── GetUserProfileUseCase.kt
│      │     └── GetDonationHistoryUseCase.kt
│      │     └─ CalculateNextDonationDateUseCase.kt // **THÊM MỚI**
│      ├── di/
│      │   └── ProfileModule.kt
│      └── ui/
│        ├── navigation/
│        │   └── ProfileNavigation.kt
│        ├─ profile/              // **CẬP NHẬT: Cấu trúc lại cho gọn**
│        │  ├─ ProfileScreen.kt
│        │  ├─ ProfileViewModel.kt
│        │  └─ ProfileContract.kt
│        ├─ edit/                 // **CẬP NHẬT: Màn hình chỉnh sửa**
│        │  ├─ EditProfileScreen.kt
│        └─ history/              // **CẬP NHẬT: Cấu trúc lại**
│          ├─ DonationHistoryScreen.kt
│          └── DonationHistoryViewModel.kt
│
feature_map_booking/
└── src/main/java/com/smartblood/mapbooking/
   ├── data/
   │   ├── local/
   │   │   ├── dao/
   │   │   │   └── HospitalDao.kt          // Interface Room DAO cho Hospital
   │   │   └── entity/
   │   │     └── HospitalEntity.kt       // Bảng Hospital trong DB cục bộ để cache
   │   ├── mapper/
   │   │   ├── HospitalMapper.kt           // Chuyển đổi HospitalEntity/Dto -> Hospital
   │   │   └── AppointmentMapper.kt        // Chuyển đổi AppointmentDto -> Appointment
   │   ├── remote/
   │   │   ├── MapBookingApiService.kt     // Interface Retrofit/Firebase cho API bản đồ
   │   │   └── dto/
   │   │     ├── HospitalDto.kt          // DTO cho thông tin bệnh viện
   │   │     ├── AvailableSlotsDto.kt    // DTO cho các khung giờ còn trống
   │   │     └── BookingRequestDto.kt    // DTO để gửi yêu cầu đặt lịch
   │   └── repository/
   │     └── MapBookingRepositoryImpl.kt  // Triển khai repository, quyết định lấy dữ liệu
từ local/remote
   │
   ├── domain/
   │   ├── model/
   │   │   ├── Hospital.kt              // Model sạch của Bệnh viện
   │   │   ├── Appointment.kt            // Model sạch của Lịch hẹn
   │   │   └── TimeSlot.kt               // Model sạch của Khung giờ
   │   ├── repository/
   │   │   └── MapBookingRepository.kt     // Interface định nghĩa các hàm cần thiết
(getHospitals, bookAppointment,...)
   │   └── usecase/
   │     ├── GetNearbyHospitalsUseCase.kt // Use case lấy danh sách bệnh viện gần đây
   │     ├── GetHospitalDetailsUseCase.kt // Use case lấy chi tiết một bệnh viện
   │     ├── GetAvailableSlotsUseCase.kt  // Use case lấy các khung giờ trống
   │     └── BookAppointmentUseCase.kt    // Use case thực hiện đặt lịch hẹn
   │
   ├── di/
   │   └── MapBookingModule.kt           // Hilt module cung cấp Repository và Use Cases
   │
   └── ui/
       ├── navigation/
       │   └── MapBookingNavigation.kt     // Định nghĩa các route và hàm điều hướng cho
module
       ├── map/
       │   ├── components/
       │   │   ├── HospitalMarker.kt       // Composable cho marker trên bản đồ
       │   │   └── FilterBottomSheet.kt    // Composable cho bộ lọc
       │   ├── MapScreen.kt              // Màn hình chính hiển thị bản đồ
       │   ├── MapViewModel.kt            // ViewModel quản lý state bản đồ, danh sách bệnh
viện
       │   └── MapContract.kt             // Định nghĩa State, Event, Effect cho MapScreen
       ├── location_detail/
       │   ├── LocationDetailScreen.kt      // Màn hình hiển thị chi tiết một địa điểm
       │   └── LocationDetailViewModel.kt   // ViewModel lấy dữ liệu chi tiết
       └── booking/
           ├── components/
           │   ├── CalendarView.kt        // Composable cho giao diện lịch
           │   └── TimeSlotGrid.kt        // Composable cho lưới chọn giờ
           ├── BookingScreen.kt          // Màn hình đặt lịch
           └── BookingViewModel.kt        // ViewModel xử lý logic chọn ngày/giờ và đặt lịch
feature_emergency/
└── src/main/java/com/smartblood/emergency/
```

```
   ├── data/
   │   ├── mapper/
   │   │   └── BloodRequestMapper.kt       // Chuyển đổi BloodRequestDto -> BloodRequest
   │   ├── remote/
   │   │   ├── EmergencyApiService.kt      // Interface cho các API liên quan đến yêu cầu
khẩn cấp
   │   │   └── dto/
   │   │       ├── BloodRequestDto.kt      // DTO cho yêu cầu máu
   │   │       └── CreateRequestDto.kt     // DTO để tạo yêu cầu mới
   │   └── repository/
   │       └── EmergencyRepositoryImpl.kt   // Triển khai repository
   │
   ├── domain/
   │   ├── model/
   │   │   ├── BloodRequest.kt            // Model sạch cho yêu cầu máu
   │   │   └── RequestStatus.kt           // Enum cho trạng thái yêu cầu (PENDING, ACTIVE,
COMPLETED)
   │   ├── repository/
   │   │   └── EmergencyRepository.kt      // Interface repository
   │   └── usecase/
   │       ├── CreateEmergencyRequestUseCase.kt // Use case tạo yêu cầu khẩn cấp
   │       └── GetMyRequestsUseCase.kt      // Use case lấy danh sách các yêu cầu đã tạo
   │
   ├── di/
   │   └── EmergencyModule.kt             // Hilt module
   │
   └── ui/
       ├── navigation/
       │   └── EmergencyNavigation.kt      // Điều hướng trong module
       ├── create_request/
       │   ├── CreateRequestScreen.kt      // Màn hình form tạo yêu cầu
       │   ├── CreateRequestViewModel.kt    // ViewModel xử lý validation và gửi form
       │   └── CreateRequestContract.kt     // Định nghĩa State, Event, Effect
       └── manage_requests/
           ├── components/
           │   └── RequestListItem.kt      // Composable hiển thị một yêu cầu trong danh sách
           ├── ManageRequestsScreen.kt      // Màn hình danh sách các yêu cầu đã tạo
           └── ManageRequestsViewModel.kt  // ViewModel lấy và quản lý danh sách yêu cầu
feature_chatbot/
└── src/main/java/com/smartblood/chatbot/
   ├── data/
   │   ├── local/
   │   │   ├── dao/
   │   │   │   └── ChatMessageDao.kt       // Room DAO để lưu lịch sử chat
   │   │   └── entity/
   │   │       └── ChatMessageEntity.kt    // Bảng ChatMessage trong DB
   │   ├── mapper/
   │   │   └── ChatMessageMapper.kt        // Chuyển đổi giữa Entity/Dto và Model
   │   ├── remote/
   │   │   ├── ChatbotApiService.kt        // Interface API để giao tiếp với Dialogflow/Gemini
   │   │   └── dto/
   │   │       ├── ChatRequestDto.kt       // DTO gửi tin nhắn lên server
   │   │       └── ChatResponseDto.kt      // DTO nhận tin nhắn trả về
   │   └── repository/
   │       └── ChatbotRepositoryImpl.kt    // Triển khai repository, gửi tin nhắn và lưu lịch sử
   │
   ├── domain/
   │   ├── model/
   │   │   ├── ChatMessage.kt              // Model sạch cho một tin nhắn
   │   │   └── SenderType.kt               // Enum người gửi (USER, BOT)
   │   ├── repository/
   │   │   └── ChatbotRepository.kt        // Interface repository
   │   └── usecase/
   │       ├── SendMessageUseCase.kt       // Use case gửi một tin nhắn
   │       └── GetChatHistoryUseCase.kt    // Use case lấy lịch sử cuộc trò chuyện
   │
   ├── di/
   │   └── ChatbotModule.kt                // Hilt module
   │
   └── ui/
       ├── navigation/
       │   └── ChatbotNavigation.kt        // Điều hướng cho màn hình chat
       └── chat/
           ├── components/
           │   ├── ChatBubble.kt           // Composable cho bong bóng chat (gửi và nhận)
           │   ├── MessageInputField.kt    // Composable cho ô nhập tin nhắn
           │   └── TypingIndicator.kt      // Composable cho hiệu ứng "Bot is typing..."
           ├── ChatbotScreen.kt            // Màn hình chat chính
           ├── ChatbotViewModel.kt         // ViewModel quản lý danh sách tin nhắn, trạng thái đang gõ
           └── ChatbotContract.kt          // Định nghĩa State, Event, Effect
```

### **HƯỚNG DẪN CÀI ĐẶT VÀ CHẠY DỰ ÁN (PROJECT SETUP GUIDE)**

Quy trình này sẽ hướng dẫn bạn cách clone, cài đặt và chạy dự án **Smart Blood Donation** trên máy tính của bạn.

#### **Giai đoạn 0: Yêu Cầu Cần Có (Prerequisites)**

Trước khi bắt đầu, hãy đảm bảo máy tính của bạn đã cài đặt các công cụ sau:

1. **Git:** Hệ thống quản lý phiên bản. Nếu chưa có, bạn có thể tải tại git-scm.com.
2. **Android Studio:** Môi trường phát triển chính. Khuyến nghị sử dụng phiên bản mới nhất (Iguana 2023.2.1 hoặc mới hơn).
    *   Tải tại: developer.android.com/studio
    *   Trong quá trình cài đặt, hãy đảm bảo bạn đã chọn cài đặt **Android SDK**. Android Studio thường sẽ tự động cài đặt JDK (Java Development Kit) đi kèm, vì vậy bạn không cần cài đặt Java riêng.

#### **Giai đoạn 1: Lấy Mã Nguồn Dự Án (Cloning the Repository)**

Bạn cần sao chép (clone) mã nguồn từ GitHub về máy tính của mình.

1. **Lấy URL của Repository**
    *   Truy cập trang repository của dự án trên GitHub.
    *   Nhấn vào nút màu xanh lá **"<> Code"**.
    *   Chọn tab **HTTPS** và sao chép URL. (Ví dụ: `https://github.com/TenNguoiDung/SmartBloodDonation-Android.git`)

2. **Thực hiện Clone:**
    Bạn có thể dùng một trong hai cách sau:

    *   **Cách A: Dùng Terminal (Command Line)**
        ```bash
        # Mở Terminal (hoặc Git Bash trên Windows)
        # Di chuyển đến thư mục bạn muốn lưu dự án (ví dụ: D:\Projects)
        cd D:\Projects

        # Chạy lệnh clone với URL bạn đã sao chép
        git clone https://github.com/TenNguoiDung/SmartBloodDonation-Android.git

        # Di chuyển vào thư mục dự án vừa được tạo
        cd SmartBloodDonation-Android
        ```

  * **Cách B: Dùng Android Studio (Khuyến khích)**
    * Mở Android Studio.
    * Trên màn hình chào mừng, chọn **"Get from VCS"** (Lấy từ Hệ thống quản lý phiên bản).
    * Dán URL bạn đã sao chép vào ô **URL**.
    * Chọn thư mục trên máy tính của bạn ở ô **Directory**.
    * Nhấn **"Clone"**. Android Studio sẽ tự động tải dự án về và mở nó ra.

#### **Giai đoạn 2: Lần Mở Đầu Tiên và Đồng Bộ Hóa Gradle (First Open & Sync)**

Đây là bước tự động nhưng quan trọng nhất. Hãy kiên nhẫn.

1. **Mở Dự Án:**
  * Nếu bạn dùng cách B, dự án sẽ được mở tự động.
  * Nếu bạn dùng cách A, trong Android Studio, chọn **File -> Open** và trỏ đến thư mục `SmartBloodDonation-Android` bạn vừa clone về.

2. **Chờ Đợi Quá Trình Đồng Bộ Hóa Tự Động:**
  * Ngay khi dự án được mở, Android Studio sẽ bắt đầu một loạt các tác vụ nền. Bạn có thể theo dõi tiến trình ở thanh trạng thái dưới cùng bên phải.
  * **Điều gì đang xảy ra?**
    * Android Studio đọc file `gradle/wrapper/gradle-wrapper.properties` và thấy dự án yêu cầu **Gradle phiên bản 8.6**.
    * Nó sẽ **tự động tải về Gradle 8.6** (việc này có thể mất vài phút nếu đây là lần đầu bạn dùng phiên bản này).
    * Sau đó, Gradle sẽ đọc tất cả các file `build.gradle.kts`, `settings.gradle.kts`, và `gradle/libs.versions.toml`.
    * Nó sẽ **tải về tất cả các thư viện (dependencies)** và **plugins** được định nghĩa trong dự án.
    * Cuối cùng, nó sẽ lập chỉ mục (indexing) toàn bộ file trong dự án.

  LƯU Ý QUAN TRỌNG: **KHÔNG LÀM GÌ CẢ** cho đến khi tất cả các thanh tiến trình ở góc dưới bên phải biến mất và bạn không còn thấy thông báo "Syncing project..." hay "Gradle build running...". Việc can thiệp có thể làm hỏng quá trình cài đặt ban đầu.

#### **Giai đoạn 3: Build và Chạy Ứng Dụng**

Sau khi quá trình đồng bộ hoàn tất, bạn đã sẵn sàng để chạy ứng dụng.

1. **Chọn Thiết Bị Chạy:**
  * Ở thanh công cụ trên cùng, bạn sẽ thấy một danh sách thả xuống các thiết bị (thường có chữ 'app' bên cạnh).
  * **Nếu dùng máy thật:** Kết nối điện thoại của bạn với máy tính và bật chế độ **"USB

Debugging"** (Gỡ lỗi qua USB) trong Tùy chọn nhà phát triển.
   *   **Nếu dùng máy ảo:** Chọn một máy ảo có sẵn. Nếu chưa có, hãy vào **Tools -> Device Manager** để tạo một máy ảo mới (khuyến nghị API 34).

2. **Chạy Ứng Dụng:**
   *   Nhấn vào nút **Run 'app'** (biểu tượng hình tam giác màu xanh lá cây) ở thanh công cụ trên cùng.
   *   Gradle sẽ biên dịch toàn bộ dự án. Lần build đầu tiên có thể mất vài phút.
   *   Nếu không có lỗi, ứng dụng sẽ được cài đặt và tự động mở trên thiết bị bạn đã chọn.

#### **Giai đoạn 4: Xử Lý Các Vấn Đề Thường Gặp (Troubleshooting)**

Nếu bạn gặp lỗi trong quá trình build, hãy thử các bước sau theo thứ tự:

1. **Clean and Rebuild Project:**
   *   Vào **Build -> Clean Project**.
   *   Sau khi hoàn tất, vào **Build -> Rebuild Project**.

2. **Invalidate Caches / Restart (Giải pháp hiệu quả nhất):**
   *   Đây là cách giải quyết hầu hết các lỗi "kỳ lạ" của Gradle hoặc Android Studio.
   *   Vào **File -> Invalidate Caches...**
   *   Trong hộp thoại hiện ra, tick vào ô đầu tiên và nhấn **"Invalidate and Restart"**. Android Studio sẽ khởi động lại và dọn dẹp toàn bộ cache.

3. **Kiểm Tra Lại SDK Location:**
   *   Vào **File -> Project Structure... -> SDK Location**.
   *   Đảm bảo đường dẫn Android SDK là chính xác. Nếu không, hãy chọn lại.

build.gradle.kts

```
// Top-level build file where you can add configuration options common to all sub-projects/modules.
plugins {
   alias(libs.plugins.android.application) apply false
   alias(libs.plugins.kotlin.android) apply false
//    alias(libs.plugins.kotlin.compose) apply false
   alias(libs.plugins.android.library) apply false
   alias(libs.plugins.hilt) apply false
   alias(libs.plugins.ksp) apply false
   alias(libs.plugins.google.services) apply false
   alias(libs.plugins.firebase.crashlytics) apply false
```

```
    alias(libs.plugins.android.secrets.gradle.plugin) apply false
}
```

gradle.properties

```
# Project-wide Gradle settings.
# IDE (e.g. Android Studio) users:
# Gradle settings configured through the IDE *will override*
# any settings specified in this file.
# For more details on how to configure your build environment visit
# http://www.gradle.org/docs/current/userguide/build_environment.html
# Specifies the JVM arguments used for the daemon process.
# The setting is particularly useful for tweaking memory settings.
org.gradle.jvmargs=-Xmx2048m -Dfile.encoding=UTF-8
# When configured, Gradle will run in incubating parallel mode.
# This option should only be used with decoupled projects. For more details, visit
# https://developer.android.com/r/tools/gradle-multi-project-decoupled-projects
# org.gradle.parallel=true
# AndroidX package structure to make it clearer which packages are bundled with the
# Android operating system, and which are packaged with your app's APK
# https://developer.android.com/topic/libraries/support-library/androidx-rn
android.useAndroidX=true
# Kotlin code style for this project: "official" or "obsolete":
kotlin.code.style=official
# Enables namespacing of each library's R class so that its R class includes only the
# resources declared in the library itself and none from the library's dependencies,
# thereby reducing the size of the R class for that library
android.nonTransitiveRClass=true
```

gradlew

```
#!/bin/sh

#
# Copyright © 2015 the original authors.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#      https://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#
# SPDX-License-Identifier: Apache-2.0
#

##############################################################
##############
#
#   Gradle start up script for POSIX generated by Gradle.
#
#   Important for running:
#
#   (1) You need a POSIX-compliant shell to run this script. If your /bin/sh is
#       noncompliant, but you have some other compliant shell such as ksh or
#       bash, then to run this script, type that shell name before the whole
#       command line, like:
#
#           ksh Gradle
#
#       Busybox and similar reduced shells will NOT work, because this script
#       requires all of these POSIX shell features:
#         * functions;
#         * expansions «$var», «${var}», «${var:-default}», «${var+SET}»,
#           «${var#prefix}», «${var%suffix}», and «$( cmd )»;
#         * compound commands having a testable exit status, especially «case»;
#         * various built-in commands including «command», «set», and «ulimit».
#
#   Important for patching:
#
#   (2) This script targets any POSIX shell, so it avoids extensions provided
#       by Bash, Ksh, etc; in particular arrays are avoided.
#
#       The "traditional" practice of packing multiple parameters into a
#       space-separated string is a well documented source of bugs and security
#       problems, so this is (mostly) avoided, by progressively accumulating
#       options in "$@", and eventually passing that to Java.
#
#       Where the inherited environment variables (DEFAULT_JVM_OPTS, JAVA_OPTS,
#       and GRADLE_OPTS) rely on word-splitting, this is performed explicitly;
#       see the in-line comments for details.
#
#       There are tweaks for specific operating systems such as AIX, CygWin,
#       Darwin, MinGW, and NonStop.
```

```
#
#   (3) This script is generated from the Groovy template
#       https://github.com/gradle/gradle/blob/HEAD/platforms/jvm/plugins-
application/src/main/resources/org/gradle/api/internal/plugins/unixStartScript.txt
#       within the Gradle project.
#
#       You can find Gradle at https://github.com/gradle/gradle/.
#
#################################################################
###############

# Attempt to set APP_HOME

# Resolve links: $0 may be a link
app_path=$0

# Need this for daisy-chained symlinks.
while
    APP_HOME=${app_path%"${app_path##*/}"}  # leaves a trailing /; empty if no leading
path
    [ -h "$app_path" ]
do
    ls=$( ls -ld "$app_path" )
    link=${ls#*' -> '}
    case $link in             #(
      /*)   app_path=$link ;; #(
      *)    app_path=$APP_HOME$link ;;
    esac
done

# This is normally unused
# shellcheck disable=SC2034
APP_BASE_NAME=${0##*/}
# Discard cd standard output in case $CDPATH is set
(https://github.com/gradle/gradle/issues/25036)
APP_HOME=$( cd -P "${APP_HOME:-./}" > /dev/null && printf '%s\n' "$PWD" ) || exit

# Use the maximum available, or set MAX_FD != -1 to use that value.
MAX_FD=maximum

warn () {
    echo "$*"
} >&2
```

```
die () {
    echo
    echo "$*"
    echo
    exit 1
} >&2

# OS specific support (must be 'true' or 'false').
cygwin=false
msys=false
darwin=false
nonstop=false
case "$( uname )" in                #(
  CYGWIN* )         cygwin=true  ;; #(
  Darwin* )         darwin=true  ;; #(
  MSYS* | MINGW* )  msys=true    ;; #(
  NONSTOP* )        nonstop=true ;;
esac

CLASSPATH="\\\"\\\""


# Determine the Java command to use to start the JVM.
if [ -n "$JAVA_HOME" ] ; then
    if [ -x "$JAVA_HOME/jre/sh/java" ] ; then
        # IBM's JDK on AIX uses strange locations for the executables
        JAVACMD=$JAVA_HOME/jre/sh/java
    else
        JAVACMD=$JAVA_HOME/bin/java
    fi
    if [ ! -x "$JAVACMD" ] ; then
        die "ERROR: JAVA_HOME is set to an invalid directory: $JAVA_HOME

Please set the JAVA_HOME variable in your environment to match the
location of your Java installation."
    fi
else
    JAVACMD=java
    if ! command -v java >/dev/null 2>&1
    then
        die "ERROR: JAVA_HOME is not set and no 'java' command could be found in your
PATH.
```

```
Please set the JAVA_HOME variable in your environment to match the
location of your Java installation."
    fi
fi

# Increase the maximum file descriptors if we can.
if ! "$cygwin" && ! "$darwin" && ! "$nonstop" ; then
    case $MAX_FD in #(
      max*)
        # In POSIX sh, ulimit -H is undefined. That's why the result is checked to see if it
worked.
        # shellcheck disable=SC2039,SC3045
        MAX_FD=$( ulimit -H -n ) ||
            warn "Could not query maximum file descriptor limit"
    esac
    case $MAX_FD in  #(
      '' | soft) :;; #(
      *)
        # In POSIX sh, ulimit -n is undefined. That's why the result is checked to see if it worked.
        # shellcheck disable=SC2039,SC3045
        ulimit -n "$MAX_FD" ||
            warn "Could not set maximum file descriptor limit to $MAX_FD"
    esac
fi

# Collect all arguments for the java command, stacking in reverse order:
#   * args from the command line
#   * the main class name
#   * -classpath
#   * -D...appname settings
#   * --module-path (only if needed)
#   * DEFAULT_JVM_OPTS, JAVA_OPTS, and GRADLE_OPTS environment variables.

# For Cygwin or MSYS, switch paths to Windows format before running java
if "$cygwin" || "$msys" ; then
    APP_HOME=$( cygpath --path --mixed "$APP_HOME" )
    CLASSPATH=$( cygpath --path --mixed "$CLASSPATH" )

    JAVACMD=$( cygpath --unix "$JAVACMD" )

    # Now convert the arguments - kludge to limit ourselves to /bin/sh
    for arg do
```

```
        if
            case $arg in                                #(
             -*)   false ;;                         # don't mess with options #(
             /?*)  t=${arg#/} t=/${t%%/*}              # looks like a POSIX filepath
                  [ -e "$t" ] ;;                    #(
             *)    false ;;
            esac
        then
            arg=$( cygpath --path --ignore --mixed "$arg" )
        fi
        # Roll the args list around exactly as many times as the number of
        # args, so each arg winds up back in the position where it started, but
        # possibly modified.
        #
        # NB: a `for` loop captures its iteration list before it begins, so
        # changing the positional parameters here affects neither the number of
        # iterations, nor the values presented in `arg`.
        shift                   # remove old arg
        set -- "$@" "$arg"      # push replacement arg
    done
fi


# Add default JVM options here. You can also use JAVA_OPTS and GRADLE_OPTS to pass
JVM options to this script.
DEFAULT_JVM_OPTS='"-Xmx64m" "-Xms64m"'

# Collect all arguments for the java command:
#   * DEFAULT_JVM_OPTS, JAVA_OPTS, and optsEnvironmentVar are not allowed to contain
shell fragments,
#     and any embedded shellness will be escaped.
#   * For example: A user cannot expect ${Hostname} to be expanded, as it is an
environment variable and will be
#     treated as '${Hostname}' itself on the command line.

set -- \
        "-Dorg.gradle.appname=$APP_BASE_NAME" \
        -classpath "$CLASSPATH" \
        -jar "$APP_HOME/gradle/wrapper/gradle-wrapper.jar" \
        "$@"

# Stop when "xargs" is not available.
if ! command -v xargs >/dev/null 2>&1
```

```
then
    die "xargs is not available"
fi

# Use "xargs" to parse quoted args.
#
# With -n1 it outputs one arg per line, with the quotes and backslashes removed.
#
# In Bash we could simply go:
#
#   readarray ARGS < <( xargs -n1 <<<"$var" ) &&
#   set -- "${ARGS[@]}" "$@"
#
# but POSIX shell has neither arrays nor command substitution, so instead we
# post-process each arg (as a line of input to sed) to backslash-escape any
# character that might be a shell metacharacter, then use eval to reverse
# that process (while maintaining the separation between arguments), and wrap
# the whole thing up as a single "set" statement.
#
# This will of course break if any of these variables contains a newline or
# an unmatched quote.
#

eval "set -- $(
        printf '%s\n' "$DEFAULT_JVM_OPTS $JAVA_OPTS $GRADLE_OPTS" |
        xargs -n1 |
        sed ' s~[^-[:alnum:]+,./:=@_]~\\&~g; ' |
        tr '\n' ' '
    )" '"$@"'

exec "$JAVACMD" "$@"
```

gradlew.bat

```
@rem
@rem Copyright 2015 the original author or authors.
@rem
@rem Licensed under the Apache License, Version 2.0 (the "License");
@rem you may not use this file except in compliance with the License.
@rem You may obtain a copy of the License at
@rem
@rem      https://www.apache.org/licenses/LICENSE-2.0
@rem
```

```
@rem Unless required by applicable law or agreed to in writing, software
@rem distributed under the License is distributed on an "AS IS" BASIS,
@rem WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
@rem See the License for the specific language governing permissions and
@rem limitations under the License.
@rem
@rem SPDX-License-Identifier: Apache-2.0
@rem

@if "%DEBUG%"=="" @echo off
@rem
##################################################################
###########
@rem
@rem  Gradle startup script for Windows
@rem
@rem
##################################################################
###########

@rem Set local scope for the variables with windows NT shell
if "%OS%"=="Windows_NT" setlocal

set DIRNAME=%~dp0
if "%DIRNAME%"=="" set DIRNAME=.
@rem This is normally unused
set APP_BASE_NAME=%~n0
set APP_HOME=%DIRNAME%

@rem Resolve any "." and ".." in APP_HOME to make it shorter.
for %%i in ("%APP_HOME%") do set APP_HOME=%%~fi

@rem Add default JVM options here. You can also use JAVA_OPTS and GRADLE_OPTS to
pass JVM options to this script.
set DEFAULT_JVM_OPTS="-Xmx64m" "-Xms64m"

@rem Find java.exe
if defined JAVA_HOME goto findJavaFromJavaHome

set JAVA_EXE=java.exe
%JAVA_EXE% -version >NUL 2>&1
if %ERRORLEVEL% equ 0 goto execute
```

```
echo. 1>&2
echo ERROR: JAVA_HOME is not set and no 'java' command could be found in your PATH.
1>&2
echo. 1>&2
echo Please set the JAVA_HOME variable in your environment to match the 1>&2
echo location of your Java installation. 1>&2

goto fail

:findJavaFromJavaHome
set JAVA_HOME=%JAVA_HOME:"=%
set JAVA_EXE=%JAVA_HOME%/bin/java.exe

if exist "%JAVA_EXE%" goto execute

echo. 1>&2
echo ERROR: JAVA_HOME is set to an invalid directory: %JAVA_HOME% 1>&2
echo. 1>&2
echo Please set the JAVA_HOME variable in your environment to match the 1>&2
echo location of your Java installation. 1>&2

goto fail

:execute
@rem Setup the command line

set CLASSPATH=


@rem Execute Gradle
"%JAVA_EXE%" %DEFAULT_JVM_OPTS% %JAVA_OPTS% %GRADLE_OPTS% "-
Dorg.gradle.appname=%APP_BASE_NAME%" -classpath "%CLASSPATH%" -jar
"%APP_HOME%\gradle\wrapper\gradle-wrapper.jar" %*

:end
@rem End local scope for the variables with windows NT shell
if %ERRORLEVEL% equ 0 goto mainEnd

:fail
rem Set variable GRADLE_EXIT_CONSOLE if you need the _script_ return code instead of
rem the _cmd.exe /c_ return code!
set EXIT_CODE=%ERRORLEVEL%
if %EXIT_CODE% equ 0 set EXIT_CODE=1
if not ""=="%GRADLE_EXIT_CONSOLE%" exit %EXIT_CODE%
exit /b %EXIT_CODE%

:mainEnd
if "%OS%"=="Windows_NT" endlocal

:omega
```

local.properties

```
## This file is automatically generated by Android Studio.
# Do not modify this file -- YOUR CHANGES WILL BE ERASED!
#
# This file should *NOT* be checked into Version Control Systems,
# as it contains information specific to your local configuration.
#
# Location of the SDK. This is only used by Gradle.
# For customization when using a Version Control System, please read the
# header note.
sdk.dir=C\:\\Users\\ADMIN\\AppData\\Local\\Android\\Sdk
MAPS_API_KEY=AIzaSyBS6lGj7CsMDE5O9bMEf3I3anmfn34OBlA
CLOUDINARY_CLOUD_NAME=dotq3sk7l
CLOUDINARY_API_KEY=943654194677826
CLOUDINARY_API_SECRET=6G4876Xf-UgHaHD7762Q3JtkOiI
```

settings.gradle.kts

```
pluginManagement {
  repositories {
    google {
      content {
        includeGroupByRegex("com\\.android.*")
        includeGroupByRegex("com\\.google.*")
        includeGroupByRegex("androidx.*")
      }
    }
    mavenCentral()
    gradlePluginPortal()
  }
}
dependencyResolutionManagement {
  repositoriesMode.set(RepositoriesMode.FAIL_ON_PROJECT_REPOS)
  repositories {
    google()
    mavenCentral()
    }
}

rootProject.name = "SmartBloodDonationAndroid"
include(":app")
include(":core")
include(":feature_auth")
include(":feature_profile")
include(":feature_map_booking")
include(":feature_emergency")
include(":feature_chatbot")
```

app/.gitignore

```
/build
# Google Services file
google-services.json
```

app/build.gradle.kts

```
plugins {
    alias(libs.plugins.android.secrets.gradle.plugin)
    alias(libs.plugins.android.application)
    alias(libs.plugins.kotlin.android)
    alias(libs.plugins.kotlin.compose.compiler)
    alias(libs.plugins.ksp)
    alias(libs.plugins.google.services)
    alias(libs.plugins.firebase.crashlytics)
    alias(libs.plugins.hilt)
//    alias(libs.plugins.maps.secrets)
}

android {
    namespace = "com.example.smartblood"
    compileSdk = 34

    defaultConfig {
        applicationId = "com.smartblood.donation"
        minSdk = 24
        targetSdk = 34
        versionCode = 1
        versionName = "1.0"

        testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            isMinifyEnabled = false
            proguardFiles(
                getDefaultProguardFile("proguard-android-optimize.txt"),
                "proguard-rules.pro"
            )
        }
    }
    compileOptions {
        sourceCompatibility = JavaVersion.VERSION_11
        targetCompatibility = JavaVersion.VERSION_11
    }
    kotlinOptions {
        jvmTarget = "11"
    }
    buildFeatures {
        compose = true
        // Thêm dòng này để truy cập BuildConfig
        buildConfig = true
    }
    // Thêm khối packagingOptions nếu chưa có
    packagingOptions {
        resources {
            excludes += "/META-INF/{AL2.0,LGPL2.1}"
        }
    }
}

dependencies {
    // Core & UI
    implementation(libs.androidx.core.ktx)
    implementation(libs.androidx.lifecycle.runtime.ktx)
    implementation(libs.androidx.activity.compose)
    implementation(platform(libs.androidx.compose.bom))
    implementation(libs.androidx.compose.ui)
    implementation(libs.androidx.compose.ui.graphics)
    implementation(libs.androidx.compose.ui.tooling.preview)
    implementation(libs.androidx.compose.material3)

    implementation("androidx.compose.material:material-icons-extended:1.6.1")
    // ------------------------------------------------------------------------------------

    // Hilt
    implementation(libs.hilt.android)
    implementation(libs.androidx.viewbinding)
    implementation(libs.transport.backend.cct)
    ksp(libs.hilt.compiler)

    // Dependencies cho các feature module
    implementation(project(":core"))
    implementation(project(":feature_auth"))
    implementation(project(":feature_profile"))
    implementation(project(":feature_map_booking"))
    implementation(project(":feature_emergency"))
    implementation(project(":feature_chatbot"))

    implementation(libs.androidx.navigation.compose)
    implementation(libs.androidx.hilt.navigation.compose)

    // Test
    testImplementation(libs.junit)
    androidTestImplementation(libs.androidx.junit)
    androidTestImplementation(libs.androidx.espresso.core)
    androidTestImplementation(platform(libs.androidx.compose.bom))
    androidTestImplementation(libs.androidx.compose.ui.test.junit4)
    debugImplementation(libs.androidx.compose.ui.tooling)
    debugImplementation(libs.androidx.compose.ui.test.manifest)

    //TrackAsia
    implementation(libs.trackasia.sdk)
    implementation(libs.trackasia.annotation.plugin)

    implementation(libs.accompanist.navigation.animation)
    implementation("com.cloudinary:cloudinary-android:2.4.0")
}
```

app/google-services.json

```
{
  "project_info": {
    "project_number": "731740765779",
    "project_id": "smart-blood-donation-2911",
    "storage_bucket": "smart-blood-donation-2911.firebasestorage.app"
  },
  "client": [
    {
      "client_info": {
        "mobilesdk_app_id": "1:731740765779:android:12b089b55854767d6150a6",
        "android_client_info": {
          "package_name": "com.smartblood.donation"
        }
      },
      "oauth_client": [],
      "api_key": [
        {
          "current_key": "AIzaSyA55BbyQtArvpqUJr2kbOCknqQymZTdMfE"
        }
      ],
      "services": {
        "appinvite_service": {
          "other_platform_oauth_client": []
        }
      }
    }
  ],
  "configuration_version": "1"
}
```

app/proguard-rules.pro

```
# Add project specific ProGuard rules here.
# You can control the set of applied configuration files using the
# proguardFiles setting in build.gradle.
#
# For more details, see
#   http://developer.android.com/guide/developing/tools/proguard.html

# If your project uses WebView with JS, uncomment the following
# and specify the fully qualified class name to the JavaScript interface
# class:
#-keepclassmembers class fqcn.of.javascript.interface.for.webview {
#   public *;
#}

# Uncomment this to preserve the line number information for
# debugging stack traces.
#-keepattributes SourceFile,LineNumberTable
```

```
# If you keep the line number information, uncomment this to
# hide the original source file name.
#-renamesourcefileattribute SourceFile
```

app/src/androidTest/java/com/example/smartblood/ExampleInstrumentedTest.kt

```
package com.example.smartblood

import androidx.test.platform.app.InstrumentationRegistry
import androidx.test.ext.junit.runners.AndroidJUnit4

import org.junit.Test
import org.junit.runner.RunWith

import org.junit.Assert.*

/**
 * Instrumented test, which will execute on an Android device.
 *
 * See [testing documentation](http://d.android.com/tools/testing).
 */
@RunWith(AndroidJUnit4::class)
class ExampleInstrumentedTest {
    @Test
    fun useAppContext() {
        // Context of the app under test.
        val appContext = InstrumentationRegistry.getInstrumentation().targetContext
        assertEquals("com.example.smartblooddonationandroid", appContext.packageName)
    }
}
```

app/src/main/AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <application
        android:name="com.smartblood.donation.MainApplication"
        android:enableOnBackInvokedCallback="true"
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.SmartBloodDonationAndroid">
        <activity
            android:name="com.smartblood.donation.MainActivity"
            android:exported="true"
            android:label="@string/app_name"
            android:theme="@style/Theme.SmartBloodDonationAndroid">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

app/src/main/assets/local.properties

```
CLOUDINARY_CLOUD_NAME=dotq3sk7l
CLOUDINARY_API_KEY=943654194677826
CLOUDINARY_API_SECRET=6G4876Xf-UgHaHD7762Q3JtkOiI
```

app/src/main/java/com/smartblood/donation/MainActivity.kt

```
package com.smartblood.donation

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Surface
import androidx.compose.ui.Modifier
import com.smartblood.donation.theme.MainScreen
import com.smartblood.donation.theme.SmartBloodDonationAndroidTheme
import dagger.hilt.android.AndroidEntryPoint

@AndroidEntryPoint
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            SmartBloodDonationAndroidTheme {
                // A surface container using the 'background' color from the theme
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colorScheme.background
                ) {
                    // FIX: Gọi MainScreen để hiển thị giao diện chính (bao gồm BottomBar)
                    MainScreen()
                }
            }
        }
    }
}
```

app/src/main/java/com/smartblood/donation/MainApplication.kt

```
package com.smartblood.donation

import android.app.Application
import android.util.Log
import com.cloudinary.android.MediaManager
import com.trackasia.android.TrackAsia // <-- Thêm TrackAsia
import java.io.InputStream
import java.util.Properties
import dagger.hilt.android.HiltAndroidApp

@HiltAndroidApp
class MainApplication : Application() {
    override fun onCreate() {
        super.onCreate()
        val properties = Properties()
        try {
            val inputStream: InputStream = assets.open("local.properties")
            properties.load(inputStream)
        } catch (e: Exception) {
            Log.e("MainApplication", "Could not read local.properties file", e)
        }
        val cloudName = properties.getProperty("CLOUDINARY_CLOUD_NAME", "")
        val apiKey = properties.getProperty("CLOUDINARY_API_KEY", "")
        val apiSecret = properties.getProperty("CLOUDINARY_API_SECRET", "")
        if (cloudName.isEmpty() || apiKey.isEmpty() ) {
            Log.e("MainApplication", "Cloudinary credentials are not set in local.properties")
        }
        val config = mapOf(
            "cloud_name" to cloudName,
            "api_key" to apiKey,
            "api_secret" to apiSecret
        )
        MediaManager.init(this, config)


        TrackAsia.getInstance(applicationContext)
    }
}
```

app/src/main/java/com/smartblood/donation/dashboard/DashboardScreen.kt

```
package com.smartblood.donation.dashboard

import androidx.compose.foundation.background
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.items
import androidx.compose.foundation.shape.CircleShape
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.filled.*
import androidx.compose.material.icons.outlined.AccessTime
import androidx.compose.material.icons.outlined.LocalHospital
import androidx.compose.material.icons.outlined.WaterDrop
import androidx.compose.material3.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.clip
import androidx.compose.ui.graphics.Brush
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.graphics.vector.ImageVector
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.style.TextOverflow
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.hilt.navigation.compose.hiltViewModel
import com.smartblood.core.domain.model.BloodRequest
import com.smartblood.core.ui.theme.PrimaryRed
import com.smartblood.core.ui.theme.PrimaryRedDark
import java.text.SimpleDateFormat
import java.util.Locale
```

```
@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun DashboardScreen(
    viewModel: DashboardViewModel = hiltViewModel()
) {
    val state by viewModel.state.collectAsState()
    val snackbarHostState = remember { SnackbarHostState() }

    // State quản lý Dialog xác nhận
    var showConfirmDialog by remember { mutableStateOf(false) }
    var selectedRequest by remember { mutableStateOf<BloodRequest?>(null) }
    var selectedVolume by remember { mutableStateOf("350") } // Mặc định 350ml

    // Xử lý hiển thị thông báo thành công
    LaunchedEffect(state.pledgeSuccess) {
        if (state.pledgeSuccess) {
            snackbarHostState.showSnackbar("Cảm ơn nghĩa cử cao đẹp của bạn! Vui lòng kiểm
tra lịch hẹn trong Hồ sơ.")
            viewModel.onEvent(DashboardEvent.OnPledgeSuccessMessageShown)
        }
    }

    // Xử lý hiển thị các lỗi khác
    LaunchedEffect(state.error) {
        state.error?.let {
            snackbarHostState.showSnackbar("Lỗi: $it")
        }
    }

    Scaffold(
        containerColor = Color(0xFFF8F9FA), // Màu nền xám rất nhạt cho toàn màn hình
        snackbarHost = { SnackbarHost(snackbarHostState) }
    ) { paddingValues ->
        val isLoading = state.isLoadingProfile || state.isLoadingRequests

        if (isLoading && state.displayableEmergencyRequests.isEmpty()) {
            Box(
                modifier = Modifier
                    .fillMaxSize()
                    .padding(paddingValues),
                contentAlignment = Alignment.Center
            ) {
                CircularProgressIndicator(color = PrimaryRed)
            }
        } else {
            Box(modifier = Modifier.fillMaxSize()) {
                Column(
                    modifier = Modifier
                        .fillMaxSize()
                        .padding(paddingValues)
                ) {
                    // 1. Phần Header: Thông tin cá nhân
                    HomeHeaderSection(
                        userName = state.userName,
                        bloodType = state.bloodType,
                        nextDonationMessage = state.nextDonationMessage
                    )

                    Spacer(modifier = Modifier.height(16.dp))

                    // 2. Tiêu đề danh sách
                    PaddingTextTitle(text = "Cần máu khẩn cấp")

                    // 3. Danh sách yêu cầu
                    Box(modifier = Modifier.weight(1f)) {
                        if (state.displayableEmergencyRequests.isEmpty()) {
                            EmptyStateView()
                        } else {
                            LazyColumn(
                                contentPadding = PaddingValues(horizontal = 16.dp, vertical = 8.dp),
                                verticalArrangement = Arrangement.spacedBy(16.dp)
                            ) {
                                items(state.displayableEmergencyRequests, key = { it.id }) { request ->
                                    EnhancedEmergencyRequestCard(
                                        request = request,
                                        isPledging = state.isPledging,
                                        isEligible = state.isEligibleToDonate,
                                        daysToWait = state.daysToWait,
                                        onAcceptClick = {
                                            // KHI CLICK NÚT TRÊN CARD:
                                            // 1. Lưu request đang chọn
                                            selectedRequest = request
                                            // 2. Reset volume về mặc định hoặc theo request yêu cầu (nếu có)
                                            val preferred = request.preferredVolume.replace("ml", "").trim()
                                            selectedVolume = if (preferred.isNotEmpty() && preferred.all { char -
> char.isDigit() }) preferred else "350"
```

```
                    // 3. Hiển thị Dialog
                    showConfirmDialog = true
                  }
                )
              }
            }
          }
        }
      }

      // --- DIALOG XÁC NHẬN (Đặt ở đây để đè lên nội dung) ---
      if (showConfirmDialog && selectedRequest != null) {
        AlertDialog(
          onDismissRequest = { showConfirmDialog = false },
          title = {
            Text(
              text = "Xác nhận hiến máu",
              style = MaterialTheme.typography.titleLarge,
              fontWeight = FontWeight.Bold
            )
          },
          text = {
            Column {
              Text(
                text = "Bạn đang đăng ký hiến máu cho:",
                style = MaterialTheme.typography.bodyMedium
              )
              Text(
                text = selectedRequest!!.hospitalName,
                style = MaterialTheme.typography.bodyLarge,
                fontWeight = FontWeight.Bold,
                color = PrimaryRed
              )
              Spacer(modifier = Modifier.height(16.dp))

              Text(
                text = "Chọn dung tích hiến:",
                style = MaterialTheme.typography.titleMedium,
                fontWeight = FontWeight.SemiBold
              )
              Spacer(modifier = Modifier.height(8.dp))

              // Hàng nút chọn dung tích
```

```
                    Row(
                        horizontalArrangement = Arrangement.spacedBy(8.dp),
                        modifier = Modifier.fillMaxWidth()
                    ) {
                        val volumes = listOf("250", "350", "450")
                        volumes.forEach { vol ->
                            val isSelected = selectedVolume == vol
                            FilterChip(
                                selected = isSelected,
                                onClick = { selectedVolume = vol },
                                label = {
                                    Text(
                                        text = "${vol}ml",
                                        fontWeight = if (isSelected) FontWeight.Bold else
FontWeight.Normal
                                    )
                                },
                                leadingIcon = if (isSelected) {
                                    { Icon(Icons.Default.Check, contentDescription = null, modifier =
Modifier.size(16.dp)) }
                                } else null
                            )
                        }
                    }

                    Spacer(modifier = Modifier.height(8.dp))
                    Text(
                        text = "Bệnh viện khuyến khích: ${selectedRequest!!.preferredVolume}",
                        style = MaterialTheme.typography.bodySmall,
                        color = Color.Gray,
                        fontStyle = androidx.compose.ui.text.font.FontStyle.Italic
                    )
                }
            },
            confirmButton = {
                Button(
                    onClick = {
                        showConfirmDialog = false
                        // Gửi sự kiện kèm ID và Volume đã chọn
                        viewModel.onEvent(
                            DashboardEvent.OnAcceptRequestClicked(
                                requestId = selectedRequest!!.id,
                                volume = "${selectedVolume}ml"
```

```
                                        )
                                    )
                                },
                                colors = ButtonDefaults.buttonColors(containerColor = PrimaryRed)
                            ) {
                                Text("Xác nhận đăng ký")
                            }
                        },
                        dismissButton = {
                            OutlinedButton(onClick = { showConfirmDialog = false }) {
                                Text("Hủy bỏ")
                            }
                        }
                    )
                }
            }
        }
    }
}

@Composable
fun HomeHeaderSection(
    userName: String,
    bloodType: String,
    nextDonationMessage: String
) {
    Box(
        modifier = Modifier
            .fillMaxWidth()
            .clip(RoundedCornerShape(bottomStart = 24.dp, bottomEnd = 24.dp))
            .background(
                Brush.verticalGradient(
                    colors = listOf(PrimaryRed, PrimaryRedDark)
                )
            )
            .padding(start = 20.dp, end = 20.dp, top = 20.dp, bottom = 30.dp)
    ) {
        Row(
            verticalAlignment = Alignment.CenterVertically,
            modifier = Modifier.fillMaxWidth()
        ) {
            // Avatar / Icon
            Surface(
                shape = CircleShape,
                color = Color.White.copy(alpha = 0.2f),
                modifier = Modifier.size(56.dp)
            ) {
                Box(contentAlignment = Alignment.Center) {
                    Icon(
                        imageVector = Icons.Default.Person,
                        contentDescription = null,
                        tint = Color.White,
                        modifier = Modifier.size(32.dp)
                    )
                }
            }

            Spacer(modifier = Modifier.width(16.dp))

            // Text Info
            Column(modifier = Modifier.weight(1f)) {
                Text(
                    text = "Xin chào, $userName",
                    style = MaterialTheme.typography.titleLarge,
                    color = Color.White,
                    fontWeight = FontWeight.Bold
                )
                Spacer(modifier = Modifier.height(4.dp))

                // Badge nhóm máu
                Row(verticalAlignment = Alignment.CenterVertically) {
                    Icon(
                        imageVector = Icons.Outlined.WaterDrop,
                        contentDescription = null,
                        tint = Color.White.copy(alpha = 0.9f),
                        modifier = Modifier.size(14.dp)
                    )
                    Spacer(modifier = Modifier.width(4.dp))
                    Text(
                        text = "Nhóm máu: $bloodType",
                        style = MaterialTheme.typography.bodyMedium,
                        color = Color.White.copy(alpha = 0.9f)
                    )
                }

                // Thông báo ngày hiến tiếp theo
```

```
                Spacer(modifier = Modifier.height(4.dp))
                Row(verticalAlignment = Alignment.CenterVertically) {
                    Icon(
                        imageVector = Icons.Outlined.AccessTime,
                        contentDescription = null,
                        tint = Color.White.copy(alpha = 0.9f),
                        modifier = Modifier.size(14.dp)
                    )
                    Spacer(modifier = Modifier.width(4.dp))
                    Text(
                        text = nextDonationMessage,
                        style = MaterialTheme.typography.bodySmall,
                        color = Color.White.copy(alpha = 0.9f),
                        maxLines = 1,
                        overflow = TextOverflow.Ellipsis
                    )
                }
            }
        }
    }
}

@Composable
fun PaddingTextTitle(text: String) {
    Text(
        text = text,
        style = MaterialTheme.typography.titleMedium,
        fontWeight = FontWeight.Bold,
        color = Color.Black.copy(alpha = 0.8f),
        modifier = Modifier.padding(horizontal = 16.dp)
    )
}

@Composable
fun EnhancedEmergencyRequestCard(
    request: BloodRequest,
    isPledging: Boolean,
    isEligible: Boolean,
    daysToWait: Long,
    onAcceptClick: () -> Unit
) {
    val dateFormat = remember { SimpleDateFormat("dd/MM/yyyy", Locale.getDefault()) }
```

```
Card(
    modifier = Modifier.fillMaxWidth(),
    shape = RoundedCornerShape(16.dp),
    elevation = CardDefaults.cardElevation(defaultElevation = 4.dp),
    colors = CardDefaults.cardColors(containerColor = Color.White)
) {
    Column(modifier = Modifier.padding(16.dp)) {
        // Header Card: Nhóm máu và Số lượng
        Row(
            modifier = Modifier.fillMaxWidth(),
            horizontalArrangement = Arrangement.SpaceBetween,
            verticalAlignment = Alignment.CenterVertically
        ) {
            Row(verticalAlignment = Alignment.CenterVertically) {
                // Icon giọt máu lớn
                Icon(
                    imageVector = Icons.Default.WaterDrop,
                    contentDescription = null,
                    tint = PrimaryRed,
                    modifier = Modifier.size(28.dp)
                )
                Spacer(modifier = Modifier.width(8.dp))
                Text(
                    text = "Nhóm ${request.bloodType}",
                    style = MaterialTheme.typography.titleLarge,
                    fontWeight = FontWeight.ExtraBold,
                    color = PrimaryRed
                )
            }
            // Badge số lượng
            Surface(
                color = PrimaryRed.copy(alpha = 0.1f),
                shape = RoundedCornerShape(8.dp)
            ) {
                Text(
                    text = "${request.quantity} đơn vị",
                    style = MaterialTheme.typography.labelLarge,
                    color = PrimaryRedDark,
                    modifier = Modifier.padding(horizontal = 12.dp, vertical = 6.dp),
                    fontWeight = FontWeight.Bold
                )
            }
        }
```

```
Divider(
   modifier = Modifier.padding(vertical = 12.dp),
   color = Color.LightGray.copy(alpha = 0.3f)
)

// Body Card: Thông tin bệnh viện
InfoRowWithIcon(
   icon = Icons.Outlined.LocalHospital,
   text = request.hospitalName,
   color = Color.Black.copy(alpha = 0.8f),
   isBold = true
)

Spacer(modifier = Modifier.height(8.dp))

// --- HIỂN THỊ DUNG TÍCH YÊU CẦU ---
Row(
   modifier = Modifier.fillMaxWidth(),
   horizontalArrangement = Arrangement.SpaceBetween
) {
   InfoRowWithIcon(
      icon = Icons.Outlined.AccessTime,
      text = "Ngày: ${dateFormat.format(request.createdAt)}",
      color = Color.Gray
   )

   // Badge dung tích yêu cầu
   Surface(
      color = Color(0xFFE3F2FD), // Màu xanh nhạt
      shape = RoundedCornerShape(4.dp)
   ) {
      Text(
         text = "Yêu cầu: ${request.preferredVolume}",
         style = MaterialTheme.typography.labelMedium,
         color = Color(0xFF1565C0), // Màu xanh đậm
         fontWeight = FontWeight.Bold,
         modifier = Modifier.padding(horizontal = 8.dp, vertical = 4.dp)
      )
   }
}

Spacer(modifier = Modifier.height(16.dp))
```

```
// Footer: Nút hành động
if (isEligible) {
  Button(
    onClick = onAcceptClick,
    enabled = !isPledging,
    modifier = Modifier
      .fillMaxWidth()
      .height(48.dp),
    colors = ButtonDefaults.buttonColors(
      containerColor = PrimaryRed,
      disabledContainerColor = PrimaryRed.copy(alpha = 0.5f)
    ),
    shape = RoundedCornerShape(12.dp)
  ) {
    if (isPledging) {
      CircularProgressIndicator(
        modifier = Modifier.size(20.dp),
        color = Color.White,
        strokeWidth = 2.dp
      )
      Spacer(modifier = Modifier.width(8.dp))
      Text("Đang xử lý...")
    } else {
      Text("Tôi muốn hiến máu", fontWeight = FontWeight.Bold)
    }
  }
} else {
  // HIỂN THỊ CẢNH BÁO NẾU KHÔNG ĐỦ ĐIỀU KIỆN
  Surface(
    color = Color.Gray.copy(alpha = 0.1f),
    shape = RoundedCornerShape(12.dp),
    border = androidx.compose.foundation.BorderStroke(1.dp,
Color.Gray.copy(alpha = 0.5f)),
    modifier = Modifier.fillMaxWidth()
  ) {
    Row(
      modifier = Modifier.padding(12.dp),
      verticalAlignment = Alignment.CenterVertically
    ) {
      Icon(Icons.Default.Info, null, tint = Color.Gray)
      Spacer(modifier = Modifier.width(8.dp))
      Text(
                    text = "Bạn cần chờ thêm $daysToWait ngày để hồi phục.",
                    style = MaterialTheme.typography.bodyMedium,
                    color = Color.DarkGray
                )
            }
        }
      }
    }
  }
}

@Composable
fun InfoRowWithIcon(
  icon: ImageVector,
  text: String,
  color: Color,
  isBold: Boolean = false
) {
  Row(verticalAlignment = Alignment.Top) {
    Icon(
      imageVector = icon,
      contentDescription = null,
      tint = Color.Gray,
      modifier = Modifier
        .size(18.dp)
        .padding(top = 2.dp) // Căn chỉnh nhẹ với dòng đầu tiên của text
    )
    Spacer(modifier = Modifier.width(12.dp))
    Text(
      text = text,
      style = MaterialTheme.typography.bodyMedium,
      color = color,
      fontWeight = if (isBold) FontWeight.SemiBold else FontWeight.Normal,
      lineHeight = 20.sp
    )
  }
}

@Composable
fun EmptyStateView() {
  Column(
    modifier = Modifier.fillMaxSize(),
    verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        Icon(
            imageVector = Icons.Default.ThumbUp,
            contentDescription = null,
            tint = Color.Gray.copy(alpha = 0.3f),
            modifier = Modifier.size(64.dp)
        )
        Spacer(modifier = Modifier.height(16.dp))
        Text(
            text = "Hiện chưa có yêu cầu khẩn cấp nào.",
            style = MaterialTheme.typography.bodyLarge,
            color = Color.Gray
        )
        Text(
            text = "Tuyệt vời! Mọi người đều đang khỏe mạnh.",
            style = MaterialTheme.typography.bodySmall,
            color = Color.Gray.copy(alpha = 0.7f)
        )
    }
}
```

app/src/main/java/com/smartblood/donation/dashboard/DashboardViewModel.kt

```
package com.smartblood.donation.dashboard

import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import com.smartblood.core.domain.model.BloodRequest
import com.example.feature_emergency.domain.usecase.AcceptEmergencyRequestUseCase
import
com.example.feature_emergency.domain.usecase.GetActiveEmergencyRequestsUseCase
import com.example.feature_emergency.domain.usecase.GetMyPledgedRequestsUseCase
import com.example.feature_profile.domain.usecase.GetUserProfileUseCase
import com.smartblood.profile.domain.usecase.CalculateNextDonationDateUseCase
import dagger.hilt.android.lifecycle.HiltViewModel
import kotlinx.coroutines.flow.*
import kotlinx.coroutines.launch
import javax.inject.Inject

// Lớp Event để giao tiếp từ UI -> ViewModel
sealed class DashboardEvent {
    data class OnAcceptRequestClicked(val requestId: String, val volume: String) :
```

```
DashboardEvent()
    object OnPledgeSuccessMessageShown : DashboardEvent()
}

// State được cập nhật hoàn chỉnh
data class DashboardState(
    val isLoadingProfile: Boolean = true,
    val isLoadingRequests: Boolean = true,
    val isPledging: Boolean = false,
    val userName: String = "",
    val bloodType: String = "N/A",
    val nextDonationMessage: String = "",
    val displayableEmergencyRequests: List<BloodRequest> = emptyList(),
    val error: String? = null,
    val pledgeSuccess: Boolean = false,
    val isEligibleToDonate: Boolean = true, // Mặc định cho phép
    val daysToWait: Long = 0 // Số ngày cần chờ
)

@HiltViewModel
class DashboardViewModel @Inject constructor(
    private val getUserProfileUseCase: GetUserProfileUseCase,
    private val calculateNextDonationDateUseCase: CalculateNextDonationDateUseCase,
    private val getActiveEmergencyRequestsUseCase: GetActiveEmergencyRequestsUseCase,
    private val acceptEmergencyRequestUseCase: AcceptEmergencyRequestUseCase,
    private val getMyPledgedRequestsUseCase: GetMyPledgedRequestsUseCase
) : ViewModel() {

    private val _state = MutableStateFlow(DashboardState())
    val state = _state.asStateFlow()

    init {
        // Bắt đầu lắng nghe dữ liệu ngay khi ViewModel được tạo
        listenToDashboardData()
    }

    fun onEvent(event: DashboardEvent) {
        when (event) {
            is DashboardEvent.OnAcceptRequestClicked -> acceptRequest(event.requestId,
event.volume)
            DashboardEvent.OnPledgeSuccessMessageShown -> _state.update {
it.copy(pledgeSuccess = false, error = null) }
        }
```

```
}

private fun listenToDashboardData() {
   // 1. Lắng nghe thông tin User Profile
   viewModelScope.launch {
      _state.update { it.copy(isLoadingProfile = true) }
      val profileResult = getUserProfileUseCase() // Lấy thông tin user một lần
      profileResult.onSuccess { userProfile ->
         val (eligible, days) = checkEligibility(userProfile.lastDonationDate)
         _state.update {
            it.copy(
               isLoadingProfile = false,
               userName = userProfile.fullName,
               bloodType = userProfile.bloodType ?: "N/A",
               nextDonationMessage = calculateNextDonationDateUseCase(userProfile)
            )
         }
      }.onFailure { error ->
         _state.update { it.copy(isLoadingProfile = false, error = error.message) }
      }
   }

   // 2. Lắng nghe và kết hợp dữ liệu từ hai nguồn Flow
   viewModelScope.launch {
      _state.update { it.copy(isLoadingRequests = true) }

      // Lấy Flow của các yêu cầu đang hoạt động
      val activeRequestsFlow = getActiveEmergencyRequestsUseCase()

      // Lấy Flow của các yêu cầu user đã chấp nhận
      val pledgedRequestsFlow = getMyPledgedRequestsUseCase()

      // Dùng `combine` để tự động tính toán lại danh sách hiển thị
      // mỗi khi một trong hai flow trên có dữ liệu mới
      combine(activeRequestsFlow, pledgedRequestsFlow) { activeResult, pledgedResult ->

         // Xử lý lỗi từ các Flow
         val errorMessage = activeResult.exceptionOrNull()?.message
            ?: pledgedResult.exceptionOrNull()?.message
         if (errorMessage != null) {
            _state.update { it.copy(isLoadingRequests = false, error = errorMessage) }
            return@combine
         }
```

```
            val activeRequests = activeResult.getOrNull() ?: emptyList()
            val pledgedRequestIds = pledgedResult.getOrNull()?.map { it.id }?.toSet() ?:
emptySet()

            // Lọc ra danh sách cần hiển thị
            val displayableRequests = activeRequests.filter { it.id !in pledgedRequestIds }

            _state.update {
               it.copy(
                  isLoadingRequests = false,
                  displayableEmergencyRequests = displayableRequests
               )
            }

         }.catch { e ->
            // Bắt các lỗi không mong muốn từ Flow
            _state.update { it.copy(isLoadingRequests = false, error = e.message) }
         }.collect() // Bắt đầu lắng nghe
      }
   }
   private fun checkEligibility(lastDateStr: String?): Pair<Boolean, Long> {
      if (lastDateStr.isNullOrBlank()) return Pair(true, 0)

      try {
         val dateFormat = java.text.SimpleDateFormat("dd/MM/yyyy",
java.util.Locale.getDefault())
         val lastDate = dateFormat.parse(lastDateStr) ?: return Pair(true, 0)

         val calendar = java.util.Calendar.getInstance()
         calendar.time = lastDate
         calendar.add(java.util.Calendar.DAY_OF_YEAR, 84) // Cộng 84 ngày

         val eligibleDate = calendar.time
         val today = java.util.Date()

         if (eligibleDate.after(today)) {
            val diff = eligibleDate.time - today.time
            val days = java.util.concurrent.TimeUnit.MILLISECONDS.toDays(diff) + 1
            return Pair(false, days)
         }
      } catch (e: Exception) {
         return Pair(true, 0) // Lỗi format thì cho phép (hoặc chặn tùy logic)
```

```
        }
        return Pair(true, 0)
    }
    private fun acceptRequest(requestId: String, volume: String) {
        viewModelScope.launch {
            _state.update { it.copy(isPledging = true, error = null) }

            val profileResult = getUserProfileUseCase()
            // ... (Phần kiểm tra profile giữ nguyên) ...
            val userProfile = profileResult.getOrNull()
            if (userProfile == null) {
                _state.update { it.copy(isPledging = false, error = "Không tìm thấy hồ sơ người
dùng.") }
                return@launch
            }

            // Gọi UseCase với volume
            val acceptResult = acceptEmergencyRequestUseCase(requestId, userProfile, volume)

            acceptResult.onSuccess {
                _state.update { it.copy(isPledging = false, pledgeSuccess = true) }
            }.onFailure { error ->
                _state.update { it.copy(isPledging = false, error = "Lỗi: ${error.message}") }
            }
        }
    }
}
```

app/src/main/java/com/smartblood/donation/navigation/AppNavHost.kt

```
package com.smartblood.donation.navigation

import androidx.compose.animation.ExperimentalAnimationApi
import androidx.compose.foundation.layout.PaddingValues
import androidx.compose.foundation.layout.padding
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.navigation.NavHostController
import androidx.navigation.NavType
import androidx.navigation.navArgument
import com.example.feature_auth.ui.navigation.authGraph
import com.example.feature_auth.ui.splash.SplashScreen
import com.example.feature_emergency.ui.history.EmergencyHistoryScreen
import com.example.feature_map_booking.domain.ui.booking.BookingScreen
import com.example.feature_map_booking.domain.ui.location_detail.LocationDetailScreen
import com.example.feature_map_booking.domain.ui.map.MapScreen
import com.example.feature_profile.ui.DonationHistoryScreen
import com.example.feature_profile.ui.EditProfileScreen
import com.example.feature_profile.ui.ProfileScreen
import com.smartblood.donation.dashboard.DashboardScreen
import com.google.accompanist.navigation.animation.AnimatedNavHost
import com.google.accompanist.navigation.animation.composable

@OptIn(ExperimentalAnimationApi::class)
@Composable
fun AppNavHost(
  navController: NavHostController,
  paddingValues: PaddingValues
) {
  AnimatedNavHost(
    navController = navController,
    startDestination = Screen.SPLASH, // Bắt đầu từ Splash
    modifier = Modifier.padding(paddingValues)
  ) {
    // 1. Đăng ký luồng Authentication
    authGraph(
      navController = navController,
      onLoginSuccess = {
        // Khi đăng nhập thành công, AppNavHost sẽ điều hướng đến Dashboard
        navController.navigate(BottomNavItem.Dashboard.route) {
          // Xóa sạch lịch sử màn hình Login/Splash để không back lại được
          popUpTo(0) {
            inclusive = true
          }
          launchSingleTop = true
        }
      }
    )

    // 2. Màn hình Splash
    composable(Screen.SPLASH) {
      SplashScreen(
        navigateToLogin = {
          navController.navigate(Graph.AUTHENTICATION) {
            popUpTo(Screen.SPLASH) { inclusive = true }
          }
        },
        navigateToDashboard = {
            navController.navigate(BottomNavItem.Dashboard.route) {
                popUpTo(Screen.SPLASH) { inclusive = true }
            }
        }
    )
}
composable("emergency_history_standalone") {
    // Lúc này IDE sẽ hết báo lỗi vì nó đã được sử dụng ở đây
    EmergencyHistoryScreen(
        onNavigateBack = { navController.popBackStack() }
    )
}
// 3. Màn hình Dashboard
composable(route = BottomNavItem.Dashboard.route) {
    DashboardScreen()
}

// 4. Luồng Bản đồ & Đặt lịch
composable(route = BottomNavItem.Map.route) {
    MapScreen(
        onNavigateToLocationDetail = { hospitalId ->
            navController.navigate("location_detail/$hospitalId")
        }
    )
}
composable(
    route = "location_detail/{hospitalId}",
    arguments = listOf(navArgument("hospitalId") { type = NavType.StringType })
) {
    LocationDetailScreen(
        onNavigateToBooking = { hospitalId, hospitalName ->
            navController.navigate("booking/$hospitalId/$hospitalName")
        },
        onNavigateBack = { navController.popBackStack() }
    )
}
composable(
    route = "booking/{hospitalId}/{hospitalName}",
    arguments = listOf(
        navArgument("hospitalId") { type = NavType.StringType },
        navArgument("hospitalName") { type = NavType.StringType }
    )
    ) {
      BookingScreen(
        onBookingSuccess = { navController.popBackStack(BottomNavItem.Map.route,
inclusive = false) },
        onNavigateBack = { navController.popBackStack() }
      )
    }

    // 5. Luồng Hồ sơ (Profile)
    composable(BottomNavItem.Profile.route) {
      ProfileScreen(
        // Tham số 1: Chuyển trang Edit Profile
        onNavigateToEditProfile = {
          navController.navigate(Screen.EDIT_PROFILE)
        },
        // Tham số 2: Chuyển trang Lịch sử
        onNavigateToDonationHistory = {
          navController.navigate(Screen.DONATION_HISTORY)
        },
        // Tham số 3: Đăng xuất -> Về trang Login
        onNavigateToLogin = {
          navController.navigate(Graph.AUTHENTICATION) {
            popUpTo(Graph.MAIN) { inclusive = true }
            launchSingleTop = true
          }
        }
      )
    }

    composable(Screen.EDIT_PROFILE) {
      EditProfileScreen(onNavigateBack = { navController.popBackStack() })
    }

    composable(Screen.DONATION_HISTORY) {
      DonationHistoryScreen(onNavigateBack = { navController.popBackStack() })
    }
  }
}
```

app/src/main/java/com/smartblood/donation/navigation/BottomNavItem.kt

```
//D:\SmartBloodDonationAndroid\app\src\main\java\com\example\smartblooddonatio
nandroid\navigation\BottomNavItem.kt
package com.smartblood.donation.navigation
```

```
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.filled.Home
import androidx.compose.material.icons.filled.LocationOn
import androidx.compose.material.icons.filled.Person
import androidx.compose.ui.graphics.vector.ImageVector

sealed class BottomNavItem(val route: String, val title: String, val icon: ImageVector) {
   object Dashboard : BottomNavItem("dashboard", "Trang chủ", Icons.Default.Home)
   object Map : BottomNavItem("map", "Bản đồ", Icons.Default.LocationOn)
   object Profile : BottomNavItem("profile", "Hồ sơ", Icons.Default.Person)
   // Thêm các mục khác sau này: Map, Emergency...
}
```

app/src/main/java/com/smartblood/donation/navigation/Screen.kt

```
//app/src/main/java/com/smartblood/donation/navigation/Screen.kt
package com.smartblood.donation.navigation

import com.example.feature_auth.ui.navigation.AUTH_GRAPH_ROUTE

// Định nghĩa các "địa chỉ" cho các màn hình
object Screen {
   const val SPLASH = "splash"
   const val DASHBOARD = "dashboard"
   const val EDIT_PROFILE = "edit_profile"
   const val DONATION_HISTORY = "donation_history"
   // Thêm các màn hình khác ở đây...
}

object Graph {
   const val ROOT = "root_graph"
   const val AUTHENTICATION = AUTH_GRAPH_ROUTE // Sử dụng lại route đã định nghĩa ở
feature_auth
   const val MAIN = "main_graph_route"
}
```

app/src/main/java/com/smartblood/donation/theme/Color.kt

```
package com.smartblood.donation.theme

import androidx.compose.ui.graphics.Color

val Purple80 = Color(0xFFD0BCFF)
val PurpleGrey80 = Color(0xFFCCC2DC)
val Pink80 = Color(0xFFEFB8C8)

val Purple40 = Color(0xFF6650a4)
val PurpleGrey40 = Color(0xFF625b71)
val Pink40 = Color(0xFF7D5260)
```

app/src/main/java/com/smartblood/donation/theme/MainScreen.kt

```
package com.smartblood.donation.theme

import androidx.compose.animation.ExperimentalAnimationApi
import androidx.compose.material3.*
import androidx.compose.runtime.Composable
import androidx.compose.runtime.getValue
import androidx.navigation.NavDestination.Companion.hierarchy
import androidx.navigation.NavGraph.Companion.findStartDestination
import androidx.navigation.compose.currentBackStackEntryAsState
import com.smartblood.donation.navigation.AppNavHost
import com.smartblood.donation.navigation.BottomNavItem
import com.google.accompanist.navigation.animation.rememberAnimatedNavController

@OptIn(ExperimentalAnimationApi::class)
@Composable
fun MainScreen() {
  val navController = rememberAnimatedNavController()

  // Danh sách các màn hình cần hiện Bottom Bar
  val navItems = listOf(
    BottomNavItem.Dashboard,
    BottomNavItem.Map,
    BottomNavItem.Profile,
  )

  // Lấy thông tin màn hình hiện tại
  val navBackStackEntry by navController.currentBackStackEntryAsState()
  val currentDestination = navBackStackEntry?.destination

  // --- LOGIC KIỂM TRA: Chỉ hiện BottomBar nếu màn hình hiện tại nằm trong danh sách
navItems ---
  val isBottomBarVisible = navItems.any { it.route == currentDestination?.route }
  // ---------------------------------------------------------------------------------------------

  Scaffold(
        bottomBar = {
            // Chỉ hiển thị nếu biến kiểm tra là true
            if (isBottomBarVisible) {
                NavigationBar {
                    navItems.forEach { screen ->
                        NavigationBarItem(
                            icon = { Icon(screen.icon, contentDescription = screen.title) },
                            label = { Text(screen.title) },
                            selected = currentDestination?.hierarchy?.any { it.route == screen.route } ==
true,
                            onClick = {
                                navController.navigate(screen.route) {
                                    popUpTo(navController.graph.findStartDestination().id) {
                                        saveState = true
                                    }
                                    launchSingleTop = true
                                    restoreState = true
                                }
                            }
                        )
                    }
                }
            }
        }
    ) { innerPadding ->
        // Nếu BottomBar bị ẩn, padding bottom sẽ là 0, giúp Splash full màn hình
        AppNavHost(
            navController = navController,
            paddingValues = innerPadding
        )
    }
}
```

app/src/main/java/com/smartblood/donation/theme/Theme.kt

```
package com.smartblood.donation.theme

import android.os.Build
import androidx.compose.foundation.isSystemInDarkTheme
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.darkColorScheme
import androidx.compose.material3.dynamicDarkColorScheme
import androidx.compose.material3.dynamicLightColorScheme
import androidx.compose.material3.lightColorScheme
import androidx.compose.runtime.Composable
import androidx.compose.ui.platform.LocalContext

private val DarkColorScheme = darkColorScheme(
    primary = Purple80,
    secondary = PurpleGrey80,
    tertiary = Pink80
)

private val LightColorScheme = lightColorScheme(
    primary = Purple40,
    secondary = PurpleGrey40,
    tertiary = Pink40

    /* Other default colors to override
    background = Color(0xFFFFFBFE),
    surface = Color(0xFFFFFBFE),
    onPrimary = Color.White,
    onSecondary = Color.White,
    onTertiary = Color.White,
    onBackground = Color(0xFF1C1B1F),
    onSurface = Color(0xFF1C1B1F),
    */
)

@Composable
fun SmartBloodDonationAndroidTheme(
    darkTheme: Boolean = isSystemInDarkTheme(),
    // Dynamic color is available on Android 12+
    dynamicColor: Boolean = true,
    content: @Composable () -> Unit
) {
    val colorScheme = when {
        dynamicColor && Build.VERSION.SDK_INT >= Build.VERSION_CODES.S -> {
            val context = LocalContext.current
            if (darkTheme) dynamicDarkColorScheme(context) else
dynamicLightColorScheme(context)
        }

        darkTheme -> DarkColorScheme
        else -> LightColorScheme
    }
    MaterialTheme(
        colorScheme = colorScheme,
        typography = Typography,
        content = content
    )
}
```

app/src/main/java/com/smartblood/donation/theme/Type.kt

```
package com.smartblood.donation.theme

import androidx.compose.material3.Typography
import androidx.compose.ui.text.TextStyle
import androidx.compose.ui.text.font.FontFamily
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.sp

// Set of Material typography styles to start with
val Typography = Typography(
    bodyLarge = TextStyle(
        fontFamily = FontFamily.Default,
        fontWeight = FontWeight.Normal,
        fontSize = 16.sp,
        lineHeight = 24.sp,
        letterSpacing = 0.5.sp
    )
    /* Other default text styles to override
    titleLarge = TextStyle(
        fontFamily = FontFamily.Default,
        fontWeight = FontWeight.Normal,
        fontSize = 22.sp,
        lineHeight = 28.sp,
        letterSpacing = 0.sp
    ),
    labelSmall = TextStyle(
        fontFamily = FontFamily.Default,
        fontWeight = FontWeight.Medium,
        fontSize = 11.sp,
        lineHeight = 16.sp,
        letterSpacing = 0.5.sp
    )
    */
)
```

app/src/main/res/drawable/ic_launcher_background.xml

```
<?xml version="1.0" encoding="utf-8"?>
<vector xmlns:android="http://schemas.android.com/apk/res/android"
   android:width="108dp"
   android:height="108dp"
   android:viewportWidth="108"
   android:viewportHeight="108">
   <path
      android:fillColor="#3DDC84"
      android:pathData="M0,0h108v108h-108z" />
   <path
      android:fillColor="#00000000"
      android:pathData="M9,0L9,108"
      android:strokeWidth="0.8"
      android:strokeColor="#33FFFFFF" />
   <path
      android:fillColor="#00000000"
      android:pathData="M19,0L19,108"
      android:strokeWidth="0.8"
      android:strokeColor="#33FFFFFF" />
   <path
      android:fillColor="#00000000"
      android:pathData="M29,0L29,108"
      android:strokeWidth="0.8"
      android:strokeColor="#33FFFFFF" />
   <path
      android:fillColor="#00000000"
      android:pathData="M39,0L39,108"
      android:strokeWidth="0.8"
      android:strokeColor="#33FFFFFF" />
   <path
      android:fillColor="#00000000"
      android:pathData="M49,0L49,108"
      android:strokeWidth="0.8"
      android:strokeColor="#33FFFFFF" />
   <path
      android:fillColor="#00000000"
      android:pathData="M59,0L59,108"
      android:strokeWidth="0.8"
      android:strokeColor="#33FFFFFF" />
   <path
      android:fillColor="#00000000"
      android:pathData="M69,0L69,108"
```

```
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M79,0L79,108"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M89,0L89,108"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M99,0L99,108"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M0,9L108,9"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M0,19L108,19"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M0,29L108,29"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M0,39L108,39"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M0,49L108,49"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
```

```
    android:fillColor="#00000000"
    android:pathData="M0,59L108,59"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M0,69L108,69"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M0,79L108,79"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M0,89L108,89"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M0,99L108,99"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M19,29L89,29"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M19,39L89,39"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M19,49L89,49"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M19,59L89,59"
    android:strokeWidth="0.8"
```

```
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M19,69L89,69"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M19,79L89,79"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M29,19L29,89"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M39,19L39,89"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M49,19L49,89"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M59,19L59,89"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M69,19L69,89"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M79,19L79,89"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
</vector>
```

app/src/main/res/drawable/ic_launcher_foreground.xml

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:width="108dp"
    android:height="108dp"
    android:viewportWidth="512"
    android:viewportHeight="512">
  <group android:scaleX="0.46"
      android:scaleY="0.46"
      android:translateX="138.24"
      android:translateY="138.24">
    <path
        android:pathData="M453.81,333.05H260.89c-29.86,0 -54.15,-24.29 -54.15,-
54.15V54.15C206.74,24.29 231.03,0 260.89,0h192.92c29.86,0 54.15,24.29
54.15,54.15v224.75C507.96,308.76 483.67,333.05 453.81,333.05zM260.89,28.59c-14.09,0
-25.56,11.46 -25.56,25.56v224.75c0,14.09 11.46,25.56 25.56,25.56h192.92c14.09,0 25.56,-
11.46 25.56,-25.56V54.15c0,-14.09 -11.47,-25.56 -25.56,-25.56H260.89z"
        android:fillColor="#B3404A"/>
    <path
        android:pathData="M376.36,411.22h-38.03c-20.4,0 -37.01,-16.6 -37.01,-37.01v-
47.75c0,-7.9 6.4,-14.3 14.3,-14.3s14.3,6.4 14.3,14.3v47.75c0,4.64 3.77,8.41
8.41,8.41h38.03c4.64,0 8.41,-3.77 8.41,-8.41v-12.52c0,-7.9 6.4,-14.3 14.3,-14.3c7.89,0
14.3,6.4 14.3,14.3v12.52C413.37,394.62 396.77,411.22 376.36,411.22z"
        android:fillColor="#B3404A"/>
    <path
        android:pathData="M293.37,119h-72.32c-7.89,0 -14.3,-6.4 -14.3,-14.3s6.4,-14.3 14.3,-
14.3h72.32c7.89,0 14.3,6.4 14.3,14.3S301.27,119 293.37,119z"
        android:fillColor="#B3404A"/>
    <path
        android:pathData="M293.37,194.64h-72.32c-7.89,0 -14.3,-6.4 -14.3,-14.3s6.4,-14.3
14.3,-14.3h72.32c7.89,0 14.3,6.4 14.3,14.3S301.27,194.64 293.37,194.64z"
        android:fillColor="#B3404A"/>
    <path
        android:pathData="M321.19,270.28H221.05c-7.89,0 -14.3,-6.4 -14.3,-14.3c0,-7.9 6.4,-
14.3 14.3,-14.3h100.14c7.89,0 14.3,6.4 14.3,14.3C335.49,263.88 329.09,270.28
321.19,270.28z"
        android:fillColor="#B3404A"/>
    <path
        android:pathData="M283.62,234.28V104.7h147.46v129.58c0,12.09 -9.8,21.89 -
21.89,21.89H305.51C293.42,256.17 283.62,246.37 283.62,234.28z"
        android:fillColor="#F4B2B0"/>
    <path
        android:pathData="M409.19,270.47H305.51c-19.95,0 -36.19,-16.23 -36.19,-
36.19V104.7c0,-7.9 6.4,-14.3 14.3,-14.3h147.46c7.89,0 14.3,6.4
```

```
14.3,14.3v129.58C445.38,254.23 429.14,270.47
409.19,270.47zM297.92,119v115.28c0,4.19 3.41,7.6 7.6,7.6h103.67c4.19,0 7.6,-3.41 7.6,-
7.6V119H297.92z"
        android:fillColor="#B3404A"/>
    <path
        android:pathData="M305.62,512h-158.3c-37.2,0 -67.46,-30.26 -67.46,-67.46v-11.26c0,-
7.9 6.4,-14.3 14.3,-14.3s14.3,6.4 14.3,14.3v11.26c0,21.43 17.43,38.87
38.87,38.87h158.3c21.43,0 38.87,-17.43 38.87,-38.87v-43.05c0,-7.9 6.4,-14.3 14.3,-
14.3c7.89,0 14.3,6.4 14.3,14.3v43.05C373.08,481.74 342.82,512 305.62,512z"
        android:fillColor="#B3404A"/>
    <path
        android:pathData="M94.16,380.38c-7.89,0 -14.3,-6.4 -14.3,-14.3V216.9c0,-7.9 6.4,-14.3
14.3,-14.3s14.3,6.4 14.3,14.3v149.18C108.45,373.98 102.05,380.38 94.16,380.38z"
        android:fillColor="#B3404A"/>
    <path
        android:pathData="M127.42,166.53c-19.86,0 -28.77,11.13 -32.28,17.67c-0.44,0.82 -
1.64,0.82 -2.07,-0c-3.49,-6.54 -12.35,-17.67 -32.21,-17.67c-22.62,0 -42.52,15.44 -
42.52,43.64c0,11.41 2.33,21.19 8.78,32.1C40.27,264.5 83.6,293.32 92.5,299c0.98,0.63
2.23,0.63 3.21,0c8.91,-5.65 52.24,-34.33 65.5,-56.74c6.45,-10.91 8.76,-20.69 8.76,-
32.1C169.98,181.97 150.05,166.53 127.42,166.53z"
        android:fillColor="#F4B2B0"/>
    <path
        android:pathData="M94.11,313.77L94.11,313.77c-3.3,0 -6.52,-0.94 -9.3,-2.72c-9.01,-
5.75 -54.76,-35.75 -69.99,-61.51c-7.45,-12.59 -10.77,-24.74 -10.77,-39.38c0,-33.57 23.9,-
57.94 56.82,-57.94c15.35,0 26.04,5.19 33.26,11.16c7.23,-5.96 17.94,-11.16 33.3,-
11.16c32.94,0 56.85,24.37 56.85,57.94c0,14.66 -3.32,26.8 -10.76,39.38c-15.34,25.93 -
61.12,55.81 -70.14,61.53C100.6,312.84 97.39,313.77 94.11,313.77zM60.86,180.82c-13.02,0
-28.23,7.69 -28.23,29.35c0,9.48 1.97,16.67 6.79,24.82c8.83,14.92 36.88,36.03
54.7,47.99c17.83,-11.91 45.9,-32.96 54.79,-47.99c4.81,-8.13 6.77,-15.32 6.77,-24.82c0,-
21.66 -15.22,-29.35 -28.26,-29.35c-9.45,0 -16.07,3.41 -19.68,10.13c-2.71,5.03 -7.93,8.16 -
13.64,8.16c-5.73,0 -10.96,-3.13 -13.65,-8.18C76.88,184.22 70.28,180.82 60.86,180.82z"
        android:fillColor="#B3404A"/>
  </group>
</vector>
```

app/src/main/res/mipmap-anydpi-v26/ic_launcher.xml

```
<?xml version="1.0" encoding="utf-8"?>
<adaptive-icon xmlns:android="http://schemas.android.com/apk/res/android">
    <background android:drawable="@color/ic_launcher_background"/>
    <foreground android:drawable="@drawable/ic_launcher_foreground"/>
</adaptive-icon>
```

app/src/main/res/mipmap-anydpi-v26/ic_launcher_round.xml

```
<?xml version="1.0" encoding="utf-8"?>
<adaptive-icon xmlns:android="http://schemas.android.com/apk/res/android">
  <background android:drawable="@color/ic_launcher_background"/>
  <foreground android:drawable="@drawable/ic_launcher_foreground"/>
</adaptive-icon>
```

app/src/main/res/values/colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <color name="purple_200">#FFBB86FC</color>
  <color name="purple_500">#FF6200EE</color>
  <color name="purple_700">#FF3700B3</color>
  <color name="teal_200">#FF03DAC5</color>
  <color name="teal_700">#FF018786</color>
  <color name="black">#FF000000</color>
  <color name="white">#FFFFFFFF</color>
</resources>
```

app/src/main/res/values/ic_launcher_background.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <color name="ic_launcher_background">#FFFFFF</color>
</resources>
```

app/src/main/res/values/strings.xml

```
<resources>
  <string name="app_name">SmartBloodDonationAndroid</string>
</resources>
```

app/src/main/res/values/themes.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

  <style name="Theme.SmartBloodDonationAndroid"
parent="android:Theme.Material.Light.NoActionBar" />
</resources>
```

app/src/main/res/xml/backup_rules.xml

```
<?xml version="1.0" encoding="utf-8"?><!--
  Sample backup rules file; uncomment and customize as necessary.
  See https://developer.android.com/guide/topics/data/autobackup
  for details.
  Note: This file is ignored for devices older than API 31
  See https://developer.android.com/about/versions/12/backup-restore
-->
<full-backup-content>
   <!--
  <include domain="sharedpref" path="."/>
  <exclude domain="sharedpref" path="device.xml"/>
-->
</full-backup-content>
```

app/src/main/res/xml/data_extraction_rules.xml

```
<?xml version="1.0" encoding="utf-8"?><!--
  Sample data extraction rules file; uncomment and customize as necessary.
  See https://developer.android.com/about/versions/12/backup-restore#xml-changes
  for details.
-->
<data-extraction-rules>
    <cloud-backup>
        <!-- TODO: Use <include> and <exclude> to control what is backed up.
        <include .../>
        <exclude .../>
        -->
    </cloud-backup>
    <!--
    <device-transfer>
        <include .../>
        <exclude .../>
    </device-transfer>
    -->
</data-extraction-rules>
```

app/src/test/java/com/example/smartblood/ExampleUnitTest.kt

```
package com.example.smartblood

import org.junit.Test

import org.junit.Assert.*

/**
 * Example local unit test, which will execute on the development machine (host).
 *
 * See [testing documentation](http://d.android.com/tools/testing).
 */
class ExampleUnitTest {
    @Test
    fun addition_isCorrect() {
        assertEquals(4, 2 + 2)
    }
}
```

core/.gitignore

```
/build
```

core/build.gradle.kts

```
// D:\SmartBloodDonationAndroid\core\build.gradle.kts

plugins {
    // Sử dụng plugin cho thư viện Android
    alias(libs.plugins.android.library)
    // Plugin cho Kotlin
    alias(libs.plugins.kotlin.android)
    // Plugin cho KSP (để Hilt và Room hoạt động)
    alias(libs.plugins.ksp)
    alias(libs.plugins.kotlin.compose.compiler)
    alias(libs.plugins.android.secrets.gradle.plugin)
}

android {
    namespace = "com.smartblood.core" // Đổi thành namespace của dự án
    compileSdk = 34

    defaultConfig {
        minSdk = 24
        testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
        consumerProguardFiles("consumer-rules.pro")
    }

    buildTypes {
        release {
            isMinifyEnabled = false
            proguardFiles(
                getDefaultProguardFile("proguard-android-optimize.txt"),
                "proguard-rules.pro"
            )
        }
    }

    compileOptions {
        sourceCompatibility = JavaVersion.VERSION_1_8 // Sử dụng 1.8 là đủ và phổ biến
        targetCompatibility = JavaVersion.VERSION_1_8
    }

    kotlinOptions {
        jvmTarget = "1.8"
    }

    // Bật tính năng Jetpack Compose
    buildFeatures {
        compose = true
        buildConfig = true
    }

}

dependencies {
    // Sử dụng bí danh từ libs.versions.toml để nhất quán

    // Core Android KTX
    implementation(libs.androidx.core.ktx)

    // Jetpack Compose
    implementation(platform(libs.androidx.compose.bom)) // BoM quản lý phiên bản
    implementation(libs.androidx.compose.ui)
    implementation(libs.androidx.compose.ui.graphics)
    implementation(libs.androidx.compose.ui.tooling.preview)
    implementation(libs.androidx.compose.material3)

    // Dependency Injection - Hilt
    implementation(libs.hilt.android)
    ksp(libs.hilt.compiler)

    // Local Database - Room
    implementation(libs.androidx.room.runtime)
    implementation(libs.androidx.room.ktx)
    ksp(libs.androidx.room.compiler)

    // Remote - Firebase
    implementation(platform(libs.firebase.bom)) // BoM quản lý phiên bản
    implementation(libs.firebase.auth.ktx)
    implementation(libs.firebase.firestore.ktx)
    implementation(libs.firebase.storage.ktx)
    implementation(libs.firebase.messaging.ktx)
    implementation(libs.firebase.crashlytics.ktx)
    implementation(libs.play.services.auth) // Google Sign-In

    // Asynchronous - Coroutines
    implementation(libs.kotlinx.coroutines.core)
    implementation(libs.kotlinx.coroutines.android)

    // Networking (Để dành cho tương lai)
    implementation(libs.retrofit)
    implementation(libs.converter.gson)
    implementation(libs.logging.interceptor)

    // Testing
    testImplementation(libs.junit)
    androidTestImplementation(libs.androidx.junit)
    androidTestImplementation(libs.androidx.espresso.core)
    androidTestImplementation(platform(libs.androidx.compose.bom))
    debugImplementation(libs.androidx.compose.ui.tooling)

    implementation("com.cloudinary:cloudinary-android:2.4.0")

}
```

core/consumer-rules.pro

[File rỗng]

core/proguard-rules.pro

```
# Add project specific ProGuard rules here.
# You can control the set of applied configuration files using the
# proguardFiles setting in build.gradle.
#
# For more details, see
#   http://developer.android.com/guide/developing/tools/proguard.html

# If your project uses WebView with JS, uncomment the following
# and specify the fully qualified class name to the JavaScript interface
# class:
#-keepclassmembers class fqcn.of.javascript.interface.for.webview {
#   public *;
#}

# Uncomment this to preserve the line number information for
# debugging stack traces.
#-keepattributes SourceFile,LineNumberTable
```

```
# If you keep the line number information, uncomment this to
# hide the original source file name.
#-renamesourcefileattribute SourceFile
```

core/src/androidTest/java/com/example/core/ExampleInstrumentedTest.kt

```
package com.example.core

import androidx.test.platform.app.InstrumentationRegistry
import androidx.test.ext.junit.runners.AndroidJUnit4

import org.junit.Test
import org.junit.runner.RunWith

import org.junit.Assert.*

/**
 * Instrumented test, which will execute on an Android device.
 *
 * See [testing documentation](http://d.android.com/tools/testing).
 */
@RunWith(AndroidJUnit4::class)
class ExampleInstrumentedTest {
    @Test
    fun useAppContext() {
        // Context of the app under test.
        val appContext = InstrumentationRegistry.getInstrumentation().targetContext
        assertEquals("com.example.core.test", appContext.packageName)
    }
}
```

core/src/main/AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android">

</manifest>
```

core/src/main/java/com/smartblood/core/data/local/AppDatabase.kt

```
//// core/src/main/java/com/smartblood/core/data/local/AppDatabase.kt
//
//package com.smartblood.core.data.local
//
//import androidx.room.Database
//import androidx.room.RoomDatabase
```

```
//
//// TODO: Thêm các class Entity của bạn vào mảng entities = [...]
//// Ví dụ: @Database(entities = [UserEntity::class], version = 1, exportSchema = false)
//@Database(entities = [], version = 1, exportSchema = false)
//abstract class AppDatabase : RoomDatabase() {
//    // TODO: Khai báo các abstract fun cho các DAO của bạn
//    // Ví dụ: abstract fun userDao(): UserDao
//
//    companion object {
//        const val DATABASE_NAME = "smartblood_db"
//    }
//}
```

core/src/main/java/com/smartblood/core/data/remote/ApiClient.kt

```
// core/src/main/java/com/smartblood/core/data/remote/ApiClient.kt

// (Để trống file này nếu bạn quyết định chỉ dùng Firebase SDK trực tiếp.
// Nhưng việc tạo module Hilt cho nó vẫn là một ý hay để chuẩn bị cho tương lai.)
// Chúng ta sẽ định nghĩa nó trong Hilt module bên dưới.
```

core/src/main/java/com/smartblood/core/di/DatabaseModule.kt

```
// core/src/main/java/com/smartblood/core/di/DatabaseModule.kt

package com.smartblood.core.di

import android.content.Context
import androidx.room.Room
//import com.smartblood.core.data.local.AppDatabase
import dagger.Module
import dagger.Provides
import dagger.hilt.InstallIn
import dagger.hilt.android.qualifiers.ApplicationContext
import dagger.hilt.components.SingletonComponent
import javax.inject.Singleton

@Module
@InstallIn(SingletonComponent::class)
object DatabaseModule {

//    @Provides
//    @Singleton
//    fun provideAppDatabase(@ApplicationContext context: Context): AppDatabase {
//        return Room.databaseBuilder(
//            context,
//          AppDatabase::class.java,
//          AppDatabase.DATABASE_NAME
//      ).fallbackToDestructiveMigration().build()
//   }

   // TODO: Cung cấp các DAO ở đây
   // Ví dụ:
   // @Provides
   // @Singleton
   // fun provideUserDao(appDatabase: AppDatabase): UserDao {
   //     return appDatabase.userDao()
   // }
}
```

core/src/main/java/com/smartblood/core/di/FirebaseModule.kt

```
// core/src/main/java/com/smartblood/core/di/FirebaseModule.kt

package com.smartblood.core.di

import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.auth.ktx.auth
import com.google.firebase.firestore.FirebaseFirestore
import com.google.firebase.firestore.ktx.firestore
import com.google.firebase.ktx.Firebase
import com.google.firebase.storage.FirebaseStorage
import com.google.firebase.storage.ktx.storage
import dagger.Module
import dagger.Provides
import dagger.hilt.InstallIn
import dagger.hilt.components.SingletonComponent
import javax.inject.Singleton

@Module
@InstallIn(SingletonComponent::class)
object FirebaseModule {

   @Provides
   @Singleton
   fun provideFirebaseAuth(): FirebaseAuth = Firebase.auth

   @Provides
   @Singleton
   fun provideFirebaseFirestore(): FirebaseFirestore = Firebase.firestore
```

```
    @Provides
    @Singleton
    fun provideFirebaseStorage(): FirebaseStorage = Firebase.storage

}
```

core/src/main/java/com/smartblood/core/di/NetworkModule.kt

```
// core/src/main/java/com/smartblood/core/di/NetworkModule.kt

package com.smartblood.core.di

import com.google.gson.GsonBuilder
import dagger.Module
import dagger.Provides
import dagger.hilt.InstallIn
import dagger.hilt.components.SingletonComponent
import okhttp3.OkHttpClient
import okhttp3.logging.HttpLoggingInterceptor
import retrofit2.Retrofit
import retrofit2.converter.gson.GsonConverterFactory
import java.util.concurrent.TimeUnit
import javax.inject.Singleton

@Module
@InstallIn(SingletonComponent::class)
object NetworkModule {

    private const val BASE_URL = "https://your.future.api.com/"

    @Provides
    @Singleton
    fun provideOkHttpClient(): OkHttpClient {
        return OkHttpClient.Builder()
            .addInterceptor(HttpLoggingInterceptor().apply {
                // Chỉ log khi ở chế độ debug
                level = HttpLoggingInterceptor.Level.BODY
            })
            .connectTimeout(30, TimeUnit.SECONDS)
            .readTimeout(30, TimeUnit.SECONDS)
            .build()
    }
    @Provides
    @Singleton
    fun provideRetrofit(okHttpClient: OkHttpClient): Retrofit {
        return Retrofit.Builder()
            .baseUrl(BASE_URL)
            .client(okHttpClient)
            .addConverterFactory(GsonConverterFactory.create(GsonBuilder().create()))
            .build()
    }
}
```

core/src/main/java/com/smartblood/core/domain/model/Appointment.kt

```
package com.smartblood.core.domain.model

//
feature_map_booking/src/main/java/com/smartblood/mapbooking/domain/model/Appo
intment.kt

import java.util.Date

data class Appointment(
    val id: String = "",
    val userId: String = "",
    val hospitalId: String = "",
    val hospitalName: String = "",
    val hospitalAddress: String = "",
    val dateTime: Date = Date(),
    val status: String = "PENDING", // PENDING, CONFIRMED, CANCELED, COMPLETED
    val registeredVolume: String = "350ml",

    // --- CÁC TRƯỜNG MỚI ---
    val actualVolume: String? = null, // Dung tích thực tế đã hiến (VD: "350ml")
    val labResult: LabResult? = null  // Kết quả xét nghiệm đính kèm
)
```

core/src/main/java/com/smartblood/core/domain/model/BloodRequest.kt

```
package com.smartblood.core.domain.model

import java.util.Date

data class BloodRequest(
    val id: String = "",
    val bloodType: String = "",
    val hospitalId: String = "",
    val hospitalName: String = "",
    val priority: String = "Trung bình",
    val quantity: Int = 0,
    val status: String = "ĐANG HOẠT ĐỘNG",
    val createdAt: Date = Date(),
    val preferredVolume: String = "350ml",
    val userPledgedDate: Date? = null

)
```

core/src/main/java/com/smartblood/core/domain/model/Donor.kt

```
package com.smartblood.core.domain.model

import java.util.Date

// Model này chứa thông tin người hiến máu sẽ được lưu vào sub-collection
data class Donor(
    val userId: String = "",
    val userName: String = "",
    val userAge: Int = 0,
    val userGender: String = "Không xác định",
    val userPhone: String = "",
    val userBloodType: String = "",
    val requestedBloodType: String = "",
    val pledgedAt: Date = Date(),
    val status: String = "Pending",
    val pledgedVolume: String = "",
    val rejectionReason: String? = null

)
```

core/src/main/java/com/smartblood/core/domain/model/Hospital.kt

```
//
feature_map_booking/src/main/java/com/smartblood/mapbooking/domain/model/Hospi
tal.kt
package com.smartblood.core.domain.model

import com.google.firebase.firestore.GeoPoint

data class Hospital(
    val id: String = "",
    val name: String = "",
    val address: String = "",
    val location: GeoPoint? = null,
    val phone: String = "",
    val workingHours: String = "",
    val availableBloodTypes: List<String> = emptyList(),
    val inventory: Map<String, Int> = emptyMap() // Map<Nhóm máu, Số lượng>

)
```

core/src/main/java/com/smartblood/core/domain/model/LabResult.kt

```
package com.smartblood.core.domain.model

import java.util.Date

/**
 * Model chứa kết quả xét nghiệm (Phiên bản Public cho User App).
 * Lưu ý: Các trường nội bộ như 'screeningStatus' hay 'confirmedBloodType'
 * KHÔNG được đưa vào đây để đảm bảo người dùng chỉ thấy thông tin được phép.
 */
data class LabResult(
    val documentUrl: String? = null, // Link file PDF/Ảnh kết quả xét nghiệm
    val conclusion: String? = null,  // Lời dặn của bác sĩ hoặc kết luận công khai
    val recordedAt: Date? = null     // Thời gian ghi nhận kết quả
)
```

core/src/main/java/com/smartblood/core/domain/model/TimeSlot.kt

```
//
feature_map_booking/src/main/java/com/smartblood/mapbooking/domain/model/Time
Slot.kt
package com.smartblood.core.domain.model
data class TimeSlot(
    val time: String = "", // e.g., "09:00"
    val isAvailable: Boolean = true
)
```

core/src/main/java/com/smartblood/core/domain/model/UserProfile.kt

```
package com.smartblood.core.domain.model

import java.util.Date

data class UserProfile(
    val uid: String = "",
    val email: String = "",
    val fullName: String = "",
    val phoneNumber: String? = null,
    val bloodType: String? = null,
    val avatarUrl: String? = null,
    val dateOfBirth: String? = null, // Lưu ý: Nếu Web lưu string thì để String
    val gender: String? = null,

    // --- SỬA Ở ĐÂY ---
    // Đổi từ Date? sang String? để khớp với dữ liệu "14/12/2025" trên Firestore
    val lastDonationDate: String? = null,

    val donationCount: Int = 0,
    val lastDonationType: String? = null,
    val isPriority: Boolean = false,
    val status: String = "Active"
)
```

core/src/main/java/com/smartblood/core/storage/data/repository/StorageRepositoryImpl.kt

```
package com.smartblood.core.storage.data.repository

import android.net.Uri
import android.util.Log
import com.cloudinary.android.MediaManager
import com.cloudinary.android.callback.ErrorInfo
import com.cloudinary.android.callback.UploadCallback // <<-- SỬ DỤNG UploadCallback
import com.smartblood.core.storage.domain.repository.StorageRepository
import kotlinx.coroutines.suspendCancellableCoroutine
import javax.inject.Inject
import kotlin.Result
import kotlin.coroutines.resume

class StorageRepositoryImpl @Inject constructor() : StorageRepository {

    override suspend fun uploadImage(uri: Uri, path: String): Result<String> {
        return suspendCancellableCoroutine { continuation ->
            // Sử dụng path làm public_id để có thể ghi đè ảnh đại diện cũ
            // Cloudinary không khuyến khích dùng '/' trong public_id
            val publicId = path

            MediaManager.get().upload(uri)
                .option("public_id", publicId)
                .option("overwrite", true) // Ghi đè file cũ nếu có cùng public_id
                .callback(object : UploadCallback { // <<-- SỬ DỤNG UploadCallback
                    override fun onStart(requestId: String) {
                        Log.d("CloudinaryUpload", "Upload started: $requestId")
                    }

                    override fun onProgress(requestId: String, bytes: Long, totalBytes: Long) {
                        // Optional: Handle progress
                    }

                    override fun onSuccess(requestId: String, resultData: Map<*, *>?) {
                        val secureUrl = resultData?.get("secure_url") as? String
                        if (secureUrl != null) {
                            Log.d("CloudinaryUpload", "Upload success: $secureUrl")
                            continuation.resume(Result.success(secureUrl))
                        } else {
                            Log.e("CloudinaryUpload", "Upload error: URL is null")
                            continuation.resume(Result.failure(Exception("Upload successful but URL
not found")))
                        }
                    }

                    override fun onError(requestId: String, error: ErrorInfo?) {
                        val errorMessage = error?.description ?: "Unknown upload error"
                        Log.e("CloudinaryUpload", "Upload error: $errorMessage")
                        continuation.resume(Result.failure(Exception(errorMessage)))
                    }

                    override fun onReschedule(requestId: String, error: ErrorInfo?) {
                        // Not typically used
                    }
                }).dispatch() // <<-- SỬ DỤNG dispatch()
        }
    }
}
```

core/src/main/java/com/smartblood/core/storage/di/StorageModule.kt

```
package com.smartblood.core.storage.di

// storage/di/StorageModule.kt

import com.smartblood.core.storage.data.repository.StorageRepositoryImpl
import com.smartblood.core.storage.domain.repository.StorageRepository
import dagger.Binds
import dagger.Module
import dagger.hilt.InstallIn
import dagger.hilt.components.SingletonComponent
import javax.inject.Singleton

@Module
@InstallIn(SingletonComponent::class)
abstract class StorageModule {
    @Binds
    @Singleton
    abstract fun bindStorageRepository(impl: StorageRepositoryImpl): StorageRepository
}
```

core/src/main/java/com/smartblood/core/storage/domain/repository/StorageRepository.kt

```
package com.smartblood.core.storage.domain.repository

import android.net.Uri
import kotlin.Result

interface StorageRepository {
    suspend fun uploadImage(uri: Uri, path: String): Result<String>
}
```

core/src/main/java/com/smartblood/core/storage/domain/usecase/UploadImageUseCase.kt

```
package com.smartblood.core.storage.domain.usecase

import android.net.Uri
import com.smartblood.core.storage.domain.repository.StorageRepository
import javax.inject.Inject

class UploadImageUseCase @Inject constructor(
    private val repository: StorageRepository
) {
    suspend operator fun invoke(uri: Uri, path: String) = repository.uploadImage(uri, path)
}
```

core/src/main/java/com/smartblood/core/ui/components/LoadingDialog.kt

```
// core/src/main/java/com/smartblood/core/ui/components/LoadingDialog.kt

package com.smartblood.core.ui.components

import androidx.compose.foundation.background
import androidx.compose.foundation.layout.Box
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.layout.size
import androidx.compose.material3.CircularProgressIndicator
import androidx.compose.material3.MaterialTheme
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.unit.dp
import androidx.compose.ui.window.Dialog
import androidx.compose.ui.window.DialogProperties

@Composable
fun LoadingDialog(isLoading: Boolean) {
    if (isLoading) {
        Dialog(
            onDismissRequest = { /* Không cho phép dismiss */ },
            properties = DialogProperties(dismissOnBackPress = false, dismissOnClickOutside =
false)
        ) {
            Box(
                modifier = Modifier
                    .size(100.dp)
                    .background(
                        color = MaterialTheme.colorScheme.surface,
                        shape = MaterialTheme.shapes.large
                    ),
                contentAlignment = Alignment.Center
            ) {
                CircularProgressIndicator()
            }
        }
    }
}
```

core/src/main/java/com/smartblood/core/ui/components/PrimaryButton.kt

```
// core/src/main/java/com/smartblood/core/ui/components/PrimaryButton.kt

package com.smartblood.core.ui.components

import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.height
import androidx.compose.material3.Button
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.unit.dp

@Composable
fun PrimaryButton(
    text: String,
    onClick: () -> Unit,
    modifier: Modifier = Modifier,
    enabled: Boolean = true
) {
    Button(
        onClick = onClick,
        modifier = modifier
            .fillMaxWidth()
            .height(50.dp),
        shape = MaterialTheme.shapes.medium,
        enabled = enabled
    ) {
        Text(
            text = text,
            style = MaterialTheme.typography.labelLarge
        )
    }
}
```

core/src/main/java/com/smartblood/core/ui/theme/Color.kt

```
// core/src/main/java/com/smartblood/core/ui/theme/Color.kt

package com.smartblood.core.ui.theme

import androidx.compose.ui.graphics.Color

// Bảng màu chính theo chủ đề
val PrimaryRed = Color(0xFFD32F2F)      // Màu đỏ máu, mạnh mẽ, kêu gọi hành động
val PrimaryRedLight = Color(0xFFFF6659)
val PrimaryRedDark = Color(0xFF9A0007)

val AccentBlue = Color(0xFF1976D2)      // Màu xanh y tế, tin cậy, an toàn
val AccentBlueLight = Color(0xFF63A4FF)
val AccentBlueDark = Color(0xFF004BA0)

// Bảng màu phụ trợ
```

```
val TextPrimary = Color(0xFF212121)    // Màu chữ chính trên nền sáng
val TextSecondary = Color(0xFF757575)  // Màu chữ phụ, chú thích
val White = Color(0xFFFFFFFF)
val LightGray = Color(0xFFF5F5F5)      // Màu nền nhẹ nhàng
val SuccessGreen = Color(0xFF388E3C)    // Màu cho thông báo thành công
val ErrorRed = Color(0xFFD32F2F)       // Màu cho thông báo lỗi
```

core/src/main/java/com/smartblood/core/ui/theme/Shape.kt

```
// core/src/main/java/com/smartblood/core/ui/theme/Shape.kt

package com.smartblood.core.ui.theme

import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material3.Shapes
import androidx.compose.ui.unit.dp

val AppShapes = Shapes(
   small = RoundedCornerShape(4.dp),   // Dùng cho các component nhỏ như chip, tag
   medium = RoundedCornerShape(8.dp),  // Dùng cho Card, Button, Input Field
   large = RoundedCornerShape(16.dp)   // Dùng cho Dialog, Bottom Sheet
)
```

core/src/main/java/com/smartblood/core/ui/theme/Theme.kt

```
// core/src/main/java/com/smartblood/core/ui/theme/Theme.kt

package com.smartblood.core.ui.theme

import android.app.Activity
import androidx.compose.foundation.isSystemInDarkTheme
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.lightColorScheme
import androidx.compose.runtime.Composable
import androidx.compose.runtime.SideEffect
import androidx.compose.ui.graphics.toArgb
import androidx.compose.ui.platform.LocalView
import androidx.core.view.WindowCompat

// Đồ án này tập trung vào light theme để đơn giản hóa, nhưng cấu trúc đã sẵn sàng cho
dark theme
private val LightColorScheme = lightColorScheme(
   primary = PrimaryRed,
   secondary = AccentBlue,
   tertiary = AccentBlueDark,
   background = White,
   surface = White,
   onPrimary = White,
   onSecondary = White,
   onTertiary = White,
   onBackground = TextPrimary,
   onSurface = TextPrimary,
   error = ErrorRed
)

@Composable
fun SmartBloodTheme(
   darkTheme: Boolean = isSystemInDarkTheme(),
   content: @Composable () -> Unit
) {
   val colorScheme = LightColorScheme // Hiện tại chỉ dùng LightColorScheme
   val view = LocalView.current
   if (!view.isInEditMode) {
       SideEffect {
           val window = (view.context as Activity).window
           window.statusBarColor = colorScheme.primary.toArgb()
           WindowCompat.getInsetsController(window, view).isAppearanceLightStatusBars =
darkTheme
       }
   }

   MaterialTheme(
       colorScheme = colorScheme,
       typography = AppTypography,
       shapes = AppShapes,
       content = content
   )
}
```

core/src/main/java/com/smartblood/core/ui/theme/Type.kt

```
// core/src/main/java/com/smartblood/core/ui/theme/Type.kt

package com.smartblood.core.ui.theme

import androidx.compose.material3.Typography
import androidx.compose.ui.text.TextStyle
import androidx.compose.ui.text.font.FontFamily
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.sp
```

```
// (Bạn có thể thêm các font chữ custom vào đây nếu muốn)

val AppTypography = Typography(
    displayLarge = TextStyle(
        fontFamily = FontFamily.Default,
        fontWeight = FontWeight.Bold,
        fontSize = 30.sp,
        lineHeight = 36.sp,
        letterSpacing = 0.sp
    ),
    headlineMedium = TextStyle(
        fontFamily = FontFamily.Default,
        fontWeight = FontWeight.SemiBold,
        fontSize = 24.sp,
        lineHeight = 28.sp,
        letterSpacing = 0.sp
    ),
    bodyLarge = TextStyle(
        fontFamily = FontFamily.Default,
        fontWeight = FontWeight.Normal,
        fontSize = 16.sp,
        lineHeight = 24.sp,
        letterSpacing = 0.5.sp
    ),
    bodyMedium = TextStyle(
        fontFamily = FontFamily.Default,
        fontWeight = FontWeight.Normal,
        fontSize = 14.sp,
        lineHeight = 20.sp,
        letterSpacing = 0.25.sp
    ),
    labelLarge = TextStyle(
        fontFamily = FontFamily.Default,
        fontWeight = FontWeight.Medium,
        fontSize = 14.sp,
        lineHeight = 20.sp,
        letterSpacing = 0.1.sp
    )
)
```

core/src/test/java/com/example/core/ExampleUnitTest.kt

```
package com.example.core

import org.junit.Test

import org.junit.Assert.*

/**
 * Example local unit test, which will execute on the development machine (host).
 *
 * See [testing documentation](http://d.android.com/tools/testing).
 */
class ExampleUnitTest {
    @Test
    fun addition_isCorrect() {
        assertEquals(4, 2 + 2)
    }
}
```

feature_auth/.gitignore

```
/build
```

feature_auth/build.gradle.kts

```
plugins {
    alias(libs.plugins.android.library)
    alias(libs.plugins.kotlin.android)
    alias(libs.plugins.kotlin.compose.compiler)
    alias(libs.plugins.ksp)
//    alias(libs.plugins.google.services)
//    alias(libs.plugins.firebase.crashlytics)

}

android {
    namespace = "com.example.feature_auth"
    compileSdk = 34

    defaultConfig {
        minSdk = 24

        testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
        consumerProguardFiles("consumer-rules.pro")
    }
    buildTypes {

        release {
            isMinifyEnabled = false
            proguardFiles(
                getDefaultProguardFile("proguard-android-optimize.txt"),
                "proguard-rules.pro"
            )
        }
    }
    compileOptions {
        sourceCompatibility = JavaVersion.VERSION_11
        targetCompatibility = JavaVersion.VERSION_11
    }
    kotlinOptions {
        jvmTarget = "11"
    }
}

dependencies {
    implementation(project(":core"))
    // Core Android KTX
    implementation(libs.androidx.core.ktx)

    implementation("androidx.compose.material:material-icons-extended:1.6.1")

    // Jetpack Compose
    implementation(platform(libs.androidx.compose.bom)) // BoM quản lý phiên bản
    implementation(libs.androidx.compose.ui)
    implementation(libs.androidx.compose.ui.graphics)
    implementation(libs.androidx.compose.ui.tooling.preview)
    implementation(libs.androidx.compose.material3)

    // Dependency Injection - Hilt
    implementation(libs.hilt.android)
    ksp(libs.hilt.compiler)
    implementation(libs.androidx.hilt.navigation.compose)
    implementation(libs.androidx.lifecycle.viewmodel.compose)
    implementation(libs.androidx.lifecycle.runtime.compose)

    // Local Database - Room
    implementation(libs.androidx.room.runtime)
    implementation(libs.androidx.room.ktx)
    ksp(libs.androidx.room.compiler)

    // Remote - Firebase
    implementation(platform(libs.firebase.bom)) // BoM quản lý phiên bản
    implementation(libs.firebase.auth.ktx)
    implementation(libs.firebase.firestore.ktx)
    implementation(libs.firebase.storage.ktx)
    implementation(libs.firebase.messaging.ktx)
    implementation(libs.firebase.crashlytics.ktx)
    implementation(libs.play.services.auth) // Google Sign-In

    // Asynchronous - Coroutines
    implementation(libs.kotlinx.coroutines.core)
    implementation(libs.kotlinx.coroutines.android)

    // Networking (Để dành cho tương lai)
    implementation(libs.retrofit)
    implementation(libs.converter.gson)
    implementation(libs.logging.interceptor)

    // Testing
    testImplementation(libs.junit)
    androidTestImplementation(libs.androidx.junit)
    androidTestImplementation(libs.androidx.espresso.core)
    androidTestImplementation(platform( libs.androidx.compose.bom))
    debugImplementation(libs.androidx.compose.ui.tooling)


}
```

feature_auth/consumer-rules.pro

[File rỗng]

feature_auth/proguard-rules.pro

```
# Add project specific ProGuard rules here.
# You can control the set of applied configuration files using the
# proguardFiles setting in build.gradle.
#
# For more details, see
#   http://developer.android.com/guide/developing/tools/proguard.html

# If your project uses WebView with JS, uncomment the following
# and specify the fully qualified class name to the JavaScript interface
# class:
#-keepclassmembers class fqcn.of.javascript.interface.for.webview {
#   public *;
#}

# Uncomment this to preserve the line number information for
# debugging stack traces.
#-keepattributes SourceFile,LineNumberTable

# If you keep the line number information, uncomment this to
# hide the original source file name.
#-renamesourcefileattribute SourceFile
```

feature_auth/src/androidTest/java/com/example/feature_auth/ExampleInstrumentedTest.kt

```
package com.example.feature_auth

import androidx.test.platform.app.InstrumentationRegistry
import androidx.test.ext.junit.runners.AndroidJUnit4

import org.junit.Test
import org.junit.runner.RunWith

import org.junit.Assert.*

/**
 * Instrumented test, which will execute on an Android device.
 *
 * See [testing documentation](http://d.android.com/tools/testing).
 */
@RunWith(AndroidJUnit4::class)
class ExampleInstrumentedTest {
    @Test
    fun useAppContext() {
        // Context of the app under test.
        val appContext = InstrumentationRegistry.getInstrumentation().targetContext
        assertEquals("com.example.feature_auth.test", appContext.packageName)
    }
}
```

feature_auth/src/main/AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android">
```

</manifest>

feature_auth/src/main/java/com/example/feature_auth/data/repository/AuthRepositoryImpl.kt

```
//
feature_auth/src/main/java/com/smartblood/auth/data/repository/AuthRepositoryImpl.
kt

package com.example.feature_auth.data.repository

import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.firestore.FirebaseFirestore
import com.example.feature_auth.domain.model.User
import com.example.feature_auth.domain.repository.AuthRepository
import kotlinx.coroutines.tasks.await
import javax.inject.Inject
import kotlin.Result

class AuthRepositoryImpl @Inject constructor(
  private val auth: FirebaseAuth,
  private val firestore: FirebaseFirestore
) : AuthRepository {

  override fun isUserAuthenticated(): Boolean {
    return auth.currentUser != null
  }

  override suspend fun loginWithEmail(email: String, password: String): Result<User> {
    return try {
      // Bước 1: Xác thực người dùng với Firebase Authentication
      val authResult = auth.signInWithEmailAndPassword(email, password).await()
      val firebaseUser = authResult.user

      if (firebaseUser != null) {
        // Bước 2: Lấy thông tin người dùng từ Cloud Firestore
        // Dùng UID từ kết quả xác thực để truy vấn đúng document.
        val userDocument =
firestore.collection("users").document(firebaseUser.uid).get().await()

        // Chuyển đổi DocumentSnapshot từ Firestore thành đối tượng User của chúng ta.
        val user = userDocument.toObject(User::class.java)
        if (user != null) {
          Result.success(user) // Trả về đối tượng User nếu thành công
        } else {
          // Trường hợp hiếm gặp: Xác thực thành công nhưng không tìm thấy bản ghi user
trong Firestore
          // (có thể do lỗi khi đăng ký hoặc dữ liệu bị xóa thủ công).
          Result.failure(Exception("Không tìm thấy dữ liệu người dùng trong cơ sở dữ
liệu."))
        }
      } else {
        Result.failure(Exception("Không xác thực được người dùng."))
      }
    } catch (e: Exception) {
      Result.failure(e)
    }
  }

  override suspend fun registerUser(fullName: String, email: String, password: String):
Result<Unit> {
    return try {
      // Bước 1: Tạo user trong Firebase Authentication
      val authResult = auth.createUserWithEmailAndPassword(email, password).await()
      val firebaseUser = authResult.user

      if (firebaseUser != null) {
        // Bước 2: Tạo đối tượng User để lưu vào Firestore
        val user = User(
          uid = firebaseUser.uid,
          email = email,
          fullName = fullName
        )

        // Bước 3: Lưu đối tượng User vào collection "users" trong Firestore
        // với document ID chính là UID của người dùng.
        firestore.collection("users").document(firebaseUser.uid).set(user).await()

        Result.success(Unit)
      } else {
        Result.failure(Exception("Failed to create user."))
      }
    } catch (e: Exception) {
      Result.failure(e)
    }
    }
}
```

feature_auth/src/main/java/com/example/feature_auth/di/AuthModule.kt

```
// feature_auth/src/main/java/com/smartblood/auth/di/AuthModule.kt

package com.smartblood.auth.di

import com.example.feature_auth.data.repository.AuthRepositoryImpl
import com.example.feature_auth.domain.repository.AuthRepository
import dagger.Binds
import dagger.Module
import dagger.hilt.InstallIn
import dagger.hilt.components.SingletonComponent
import javax.inject.Singleton

@Module
@InstallIn(SingletonComponent::class)
abstract class AuthModule {

    @Binds
    @Singleton
    abstract fun bindAuthRepository(
        authRepositoryImpl: AuthRepositoryImpl
    ): AuthRepository
}
```

feature_auth/src/main/java/com/example/feature_auth/domain/model/User.kt

```
// feature_auth/src/main/java/com/smartblood/auth/domain/model/User.kt

package com.example.feature_auth.domain.model

data class User(
    val uid: String = "",
    val email: String = "",
    val fullName: String = ""
    // Thêm các trường khác sau này, ví dụ:
    // val bloodType: String? = null,
    // val phoneNumber: String? = null
)
```

feature_auth/src/main/java/com/example/feature_auth/domain/repository/AuthRepository.kt

```
//
feature_auth/src/main/java/com/smartblood/auth/domain/repository/AuthRepository.kt

package com.example.feature_auth.domain.repository

// Sử dụng Result của Kotlin để đóng gói thành công hoặc lỗi một cách an toàn
import com.example.feature_auth.domain.model.User
import kotlin.Result

interface AuthRepository {
   fun isUserAuthenticated(): Boolean

   /**
    * Thực hiện đăng nhập bằng email và mật khẩu.
    * @return Result.success(Unit) nếu thành công, Result.failure(Exception) nếu thất bại.
    */
   suspend fun loginWithEmail(email: String, password: String): Result<User>
   suspend fun registerUser(fullName: String, email: String, password: String): Result<Unit>
}
```

feature_auth/src/main/java/com/example/feature_auth/domain/usecase/CheckUserAuthenticationUseCase.kt

```
//
feature_auth/src/main/java/com/smartblood/auth/domain/usecase/CheckUserAuthenticationUseCase.kt

package com.example.feature_auth.domain.usecase

import com.example.feature_auth.domain.repository.AuthRepository
import javax.inject.Inject

class CheckUserAuthenticationUseCase @Inject constructor(
   private val repository: AuthRepository
) {
   operator fun invoke(): Boolean {
      return repository.isUserAuthenticated()
   }
}
```

feature_auth/src/main/java/com/example/feature_auth/domain/usecase/LoginUseCase.kt

```
// feature_auth/src/main/java/com/smartblood/auth/domain/usecase/LoginUseCase.kt

package com.example.feature_auth.domain.usecase

import com.example.feature_auth.domain.model.User
import com.example.feature_auth.domain.repository.AuthRepository
import javax.inject.Inject

class LoginUseCase @Inject constructor(
    private val repository: AuthRepository
) {
    suspend operator fun invoke(email: String, password: String): Result<User> {
        // Có thể thêm logic kiểm tra dữ liệu đầu vào ở đây
        if (email.isBlank() || password.isBlank()) {
            return Result.failure(IllegalArgumentException("Email and password cannot be
empty."))
        }
        return repository.loginWithEmail(email, password)
    }
}
```

feature_auth/src/main/java/com/example/feature_auth/domain/usecase/RegisterUseCase.kt

```
//
feature_auth/src/main/java/com/smartblood/auth/domain/usecase/RegisterUseCase.kt

package com.example.feature_auth.domain.usecase

import com.example.feature_auth.domain.repository.AuthRepository
import javax.inject.Inject

class RegisterUseCase @Inject constructor(
    private val repository: AuthRepository
) {
    suspend operator fun invoke(fullName: String, email: String, password: String):
Result<Unit> {
        if (fullName.isBlank() || email.isBlank() || password.length < 6) {
            return Result.failure(IllegalArgumentException("Vui lòng điền đầy đủ thông tin. Mật
khẩu phải có ít nhất 6 ký tự."))
        }
        return repository.registerUser(fullName, email, password)
    }
}
```

feature_auth/src/main/java/com/example/feature_auth/ui/login/LoginContract.kt

```
// feature_auth/src/main/java/com/smartblood/auth/ui/login/LoginContract.kt

package com.example.feature_auth.ui.login

// Định nghĩa trạng thái của màn hình
data class LoginState(
    val email: String = "",
    val password: String = "",
    val isLoading: Boolean = false,
    val error: String? = null,
    val loginSuccess: Boolean = false
)

// Định nghĩa các sự kiện mà người dùng có thể tạo ra
sealed class LoginEvent {
    data class OnEmailChanged(val email: String) : LoginEvent()
    data class OnPasswordChanged(val password: String) : LoginEvent()
    object OnLoginClicked : LoginEvent()
    object OnErrorDismissed : LoginEvent()
}
```

feature_auth/src/main/java/com/example/feature_auth/ui/login/LoginScreen.kt

```
package com.example.feature_auth.ui.login

import androidx.compose.foundation.background
import androidx.compose.foundation.clickable
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.rememberScrollState
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.foundation.text.KeyboardOptions
import androidx.compose.foundation.verticalScroll
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.filled.Email
import androidx.compose.material.icons.filled.Favorite
import androidx.compose.material.icons.filled.Lock
import androidx.compose.material.icons.filled.Visibility
import androidx.compose.material.icons.filled.VisibilityOff
import androidx.compose.material3.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.input.ImeAction
import androidx.compose.ui.text.input.KeyboardCapitalization
import androidx.compose.ui.text.input.KeyboardType
import androidx.compose.ui.text.input.PasswordVisualTransformation
import androidx.compose.ui.text.input.VisualTransformation
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.hilt.navigation.compose.hiltViewModel

@Composable
fun LoginScreen(
    viewModel: LoginViewModel = hiltViewModel(),
    navigateToDashboard: () -> Unit,
    navigateToRegister: () -> Unit,
) {
    val state by viewModel.state.collectAsState()
    var passwordVisible by remember { mutableStateOf(false) }
    val scrollState = rememberScrollState()

    LaunchedEffect(state.loginSuccess) {
        if (state.loginSuccess) {
            navigateToDashboard()
        }
    }

    if (state.error != null) {
        AlertDialog(
            onDismissRequest = { viewModel.onEvent(LoginEvent.OnErrorDismissed) },
            title = { Text("Đăng nhập thất bại") },
            text = { Text(state.error!!) },
            confirmButton = {
                TextButton(onClick = { viewModel.onEvent(LoginEvent.OnErrorDismissed) }) {
                    Text("Đóng")
                }
            }
        )
    }
```

```
Box(modifier = Modifier.fillMaxSize()) {
  Column(
    modifier = Modifier
      .fillMaxSize()
      .padding(24.dp)
      .verticalScroll(scrollState),
    horizontalAlignment = Alignment.CenterHorizontally,
    verticalArrangement = Arrangement.Center
  ) {
    Icon(
      imageVector = Icons.Default.Favorite,
      contentDescription = "Logo",
      tint = MaterialTheme.colorScheme.primary,
      modifier = Modifier.size(80.dp)
    )
    Spacer(modifier = Modifier.height(16.dp))
    Text(
      text = "Chào mừng trở lại!",
      style = MaterialTheme.typography.headlineMedium,
      fontWeight = FontWeight.Bold,
      color = MaterialTheme.colorScheme.onBackground
    )
    Text(
      text = "Đăng nhập để tiếp tục hành trình nhân ái",
      style = MaterialTheme.typography.bodyMedium,
      color = Color.Gray
    )

    Spacer(modifier = Modifier.height(32.dp))

    // --- EMAIL FIELD (ĐÃ CẬP NHẬT) ---
    OutlinedTextField(
      value = state.email,
      onValueChange = { viewModel.onEvent(LoginEvent.OnEmailChanged(it)) },
      label = { Text("Email") },
      placeholder = { Text("nhapemail@example.com") },
      leadingIcon = { Icon(Icons.Default.Email, contentDescription = null) },
      modifier = Modifier.fillMaxWidth(),
      shape = RoundedCornerShape(12.dp),
      singleLine = true,
      // Cấu hình bàn phím để gợi ý Email
      keyboardOptions = KeyboardOptions(
            capitalization = KeyboardCapitalization.None, // Không viết hoa
            autoCorrect = false,                    // Tắt sửa lỗi chính tả
            keyboardType = KeyboardType.Email,           // Kiểu bàn phím Email
            imeAction = ImeAction.Next
        )
    )
    // --------------------------------

    Spacer(modifier = Modifier.height(16.dp))

    OutlinedTextField(
        value = state.password,
        onValueChange = { viewModel.onEvent(LoginEvent.OnPasswordChanged(it)) },
        label = { Text("Mật khẩu") },
        leadingIcon = { Icon(Icons.Default.Lock, contentDescription = null) },
        trailingIcon = {
            val image = if (passwordVisible) Icons.Filled.Visibility else
Icons.Filled.VisibilityOff
            IconButton(onClick = { passwordVisible = !passwordVisible }) {
                Icon(imageVector = image, contentDescription = "Toggle Password")
            }
        },
        visualTransformation = if (passwordVisible) VisualTransformation.None else
PasswordVisualTransformation(),
        modifier = Modifier.fillMaxWidth(),
        shape = RoundedCornerShape(12.dp),
        singleLine = true,
        keyboardOptions = KeyboardOptions(
            keyboardType = KeyboardType.Password,
            imeAction = ImeAction.Done
        )
    )

    Row(
        modifier = Modifier.fillMaxWidth(),
        horizontalArrangement = Arrangement.End
    ) {
        TextButton(onClick = { /* TODO: Forgot Password */ }) {
            Text("Quên mật khẩu?", color = MaterialTheme.colorScheme.primary)
        }
    }

    Spacer(modifier = Modifier.height(16.dp))
```

```
        Button(
            onClick = { viewModel.onEvent(LoginEvent.OnLoginClicked) },
            modifier = Modifier
                .fillMaxWidth()
                .height(50.dp),
            shape = RoundedCornerShape(12.dp),
            enabled = !state.isLoading
        ) {
            Text(
                text = "Đăng nhập",
                fontSize = 16.sp,
                fontWeight = FontWeight.Bold
            )
        }

        Spacer(modifier = Modifier.height(24.dp))

        Row(
            verticalAlignment = Alignment.CenterVertically,
            horizontalArrangement = Arrangement.Center
        ) {
            Text("Chưa có tài khoản? ", color = Color.Gray)
            Text(
                text = "Đăng ký ngay",
                color = MaterialTheme.colorScheme.primary,
                fontWeight = FontWeight.Bold,
                modifier = Modifier.clickable { navigateToRegister() }
            )
        }
    }

    if (state.isLoading) {
        Box(
            modifier = Modifier
                .fillMaxSize()
                .background(Color.Black.copy(alpha = 0.5f))
                .clickable(enabled = false) {},
            contentAlignment = Alignment.Center
        ) {
            Card(
                shape = RoundedCornerShape(16.dp),
                colors = CardDefaults.cardColors(containerColor = Color.White)
                ) {
                    Column(
                        modifier = Modifier.padding(24.dp),
                        horizontalAlignment = Alignment.CenterHorizontally
                    ) {
                        CircularProgressIndicator()
                        Spacer(modifier = Modifier.height(16.dp))
                        Text("Đang đăng nhập...", style = MaterialTheme.typography.bodyMedium)
                    }
                }
            }
        }
    }
}
```

feature_auth/src/main/java/com/example/feature_auth/ui/login/LoginViewModel.kt

```
package com.example.feature_auth.ui.login

import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import com.example.feature_auth.domain.usecase.LoginUseCase
import dagger.hilt.android.lifecycle.HiltViewModel
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.asStateFlow
import kotlinx.coroutines.flow.update
import kotlinx.coroutines.launch
import javax.inject.Inject

@HiltViewModel
class LoginViewModel @Inject constructor(
    private val loginUseCase: LoginUseCase
) : ViewModel() {
    private val _state = MutableStateFlow(LoginState())
    val state = _state.asStateFlow()

    fun onEvent(event: LoginEvent) {
        when (event) {
            is LoginEvent.OnEmailChanged -> {
                _state.update { it.copy(email = event.email) }
            }
            is LoginEvent.OnPasswordChanged -> {
                _state.update { it.copy(password = event.password) }
            }
            LoginEvent.OnLoginClicked -> {
                login()
            }
            LoginEvent.OnErrorDismissed -> {
                _state.update { it.copy(error = null) }
            }
        }
    }

    private fun login() {
        viewModelScope.launch {
            _state.update { it.copy(isLoading = true, error = null) }
            val result = loginUseCase(state.value.email, state.value.password)
            result.onSuccess {
                _state.update { it.copy(isLoading = false, loginSuccess = true) }
            }.onFailure { exception ->
                // --- SỬA: Dịch lỗi sang tiếng Việt ---
                val errorMessage = when {
                    exception.message?.contains("badly formatted") == true -> "Định dạng email
không hợp lệ."
                    exception.message?.contains("user-not-found") == true ||
exception.message?.contains("There is no user") == true -> "Tài khoản không tồn tại."
                    exception.message?.contains("wrong-password") == true ||
exception.message?.contains("INVALID_LOGIN_CREDENTIALS") == true -> "Sai mật khẩu
hoặc email."
                    exception.message?.contains("network error") == true -> "Lỗi kết nối mạng. Vui
lòng kiểm tra lại."
                    else -> "Đăng nhập thất bại: ${exception.message}"
                }
                // ------------------------------------
                _state.update {
                    it.copy(isLoading = false, error = errorMessage)
                }
            }
        }
    }
}
```

feature_auth/src/main/java/com/example/feature_auth/ui/navigation/AuthNavigation.kt

```
package com.example.feature_auth.ui.navigation
```

```
import androidx.navigation.NavGraphBuilder
import androidx.navigation.NavHostController
import androidx.navigation.compose.composable
import androidx.navigation.navigation
import com.example.feature_auth.ui.login.LoginScreen
import com.example.feature_auth.ui.register.RegisterScreen

const val AUTH_GRAPH_ROUTE = "auth_graph"

// SỬA: Thêm tham số onLoginSuccess vào hàm này
fun NavGraphBuilder.authGraph(
    navController: NavHostController,
    onLoginSuccess: () -> Unit // <--- THÊM THAM SỐ NÀY
) {
    navigation(
        startDestination = AuthScreen.Login.route,
        route = AUTH_GRAPH_ROUTE
    ) {
        composable(route = AuthScreen.Login.route) {
            LoginScreen(
                navigateToDashboard = {
                    // Gọi callback được truyền vào thay vì tự navigate
                    onLoginSuccess()
                },
                navigateToRegister = {
                    navController.navigate(AuthScreen.Register.route)
                }
            )
        }

        composable(route = AuthScreen.Register.route) {
            RegisterScreen(
                navigateToDashboard = {
                    // Gọi callback được truyền vào
                    onLoginSuccess()
                },
                navigateBack = {
                    navController.popBackStack()
                }
            )
        }
    }
}
```

feature_auth/src/main/java/com/example/feature_auth/ui/navigation/AuthScreen.kt

```
//D:\SmartBloodDonationAndroid\feature_auth\src\main\java\com\example\feature_auth\ui\navigation\AuthScreen.kt
package com.example.feature_auth.ui.navigation


// Định nghĩa các route cụ thể bên trong luồng xác thực
sealed class AuthScreen(val route: String) {
   object Login : AuthScreen("login_screen")
   object Register : AuthScreen("register_screen")
   // Thêm các màn hình khác nếu có, ví dụ:
   // object ForgotPassword : AuthScreen("forgot_password_screen")
   // object FaceAuthGuide : AuthScreen("face_auth_guide_screen")
}
```

feature_auth/src/main/java/com/example/feature_auth/ui/register/RegisterContract.kt

```
// feature_auth/src/main/java/com/smartblood/auth/ui/register/RegisterContract.kt

package com.example.feature_auth.ui.register

// Định nghĩa trạng thái của màn hình
data class RegisterState(
   val fullName: String = "",
   val email: String = "",
   val password: String = "",
   val isLoading: Boolean = false,
   val error: String? = null,
   val registrationSuccess: Boolean = false
)

// Định nghĩa các sự kiện mà người dùng có thể tạo ra
sealed class RegisterEvent {
   data class OnFullNameChanged(val fullName: String) : RegisterEvent()
   data class OnEmailChanged(val email: String) : RegisterEvent()
   data class OnPasswordChanged(val password: String) : RegisterEvent()
   object OnRegisterClicked : RegisterEvent()
   object OnErrorDismissed : RegisterEvent()
}
```

feature_auth/src/main/java/com/example/feature_auth/ui/register/RegisterScreen.kt

```
package com.example.feature_auth.ui.register

import androidx.compose.foundation.background
import androidx.compose.foundation.clickable
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.rememberScrollState
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.foundation.text.KeyboardOptions
import androidx.compose.foundation.verticalScroll
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.automirrored.filled.ArrowBack
import androidx.compose.material.icons.filled.*
import androidx.compose.material3.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.input.ImeAction
import androidx.compose.ui.text.input.KeyboardCapitalization
import androidx.compose.ui.text.input.KeyboardType
import androidx.compose.ui.text.input.PasswordVisualTransformation
import androidx.compose.ui.text.input.VisualTransformation
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.hilt.navigation.compose.hiltViewModel

@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun RegisterScreen(
  viewModel: RegisterViewModel = hiltViewModel(),
  navigateToDashboard: () -> Unit,
  navigateBack: () -> Unit,
) {
  val state by viewModel.state.collectAsState()
  var passwordVisible by remember { mutableStateOf(false) }
  val scrollState = rememberScrollState()

  LaunchedEffect(state.registrationSuccess) {
    if (state.registrationSuccess) {
      navigateToDashboard()
    }
}

if (state.error != null) {
    AlertDialog(
        onDismissRequest = { viewModel.onEvent(RegisterEvent.OnErrorDismissed) },
        title = { Text("Đăng ký thất bại") },
        text = { Text(state.error!!) },
        confirmButton = {
            TextButton(onClick = { viewModel.onEvent(RegisterEvent.OnErrorDismissed) }) {
                Text("Thử lại")
            }
        }
    )
}

Scaffold(
    topBar = {
        TopAppBar(
            title = { },
            navigationIcon = {
                IconButton(onClick = navigateBack) {
                    Icon(Icons.AutoMirrored.Filled.ArrowBack, contentDescription = "Back")
                }
            },
            colors = TopAppBarDefaults.topAppBarColors(containerColor = Color.Transparent)
        )
    }
) { paddingValues ->
    Box(modifier = Modifier.fillMaxSize()) {
        Column(
            modifier = Modifier
                .fillMaxSize()
                .padding(paddingValues)
                .padding(horizontal = 24.dp)
                .verticalScroll(scrollState),
            horizontalAlignment = Alignment.CenterHorizontally
        ) {
            Spacer(modifier = Modifier.height(10.dp))

            Text(
                text = "Tạo tài khoản",
                style = MaterialTheme.typography.headlineLarge,
                fontWeight = FontWeight.Bold,
                color = MaterialTheme.colorScheme.primary
            )
            Text(
                text = "Tham gia cộng đồng hiến máu ngay hôm nay",
                style = MaterialTheme.typography.bodyMedium,
                color = Color.Gray,
                modifier = Modifier.padding(top = 8.dp, bottom = 32.dp)
            )

            // Full Name
            OutlinedTextField(
                value = state.fullName,
                onValueChange = { viewModel.onEvent(RegisterEvent.OnFullNameChanged(it))
    },
                label = { Text("Họ và tên") },
                leadingIcon = { Icon(Icons.Default.Person, contentDescription = null) },
                modifier = Modifier.fillMaxWidth(),
                shape = RoundedCornerShape(12.dp),
                singleLine = true,
                keyboardOptions = KeyboardOptions(
                    imeAction = ImeAction.Next,
                    keyboardType = KeyboardType.Text,
                    capitalization = KeyboardCapitalization.Words // Viết hoa chữ cái đầu mỗi từ
                )
            )

            Spacer(modifier = Modifier.height(16.dp))

            // --- EMAIL FIELD (ĐÃ CẬP NHẬT) ---
            OutlinedTextField(
                value = state.email,
                onValueChange = { viewModel.onEvent(RegisterEvent.OnEmailChanged(it)) },
                label = { Text("Email") },
                leadingIcon = { Icon(Icons.Default.Email, contentDescription = null) },
                modifier = Modifier.fillMaxWidth(),
                shape = RoundedCornerShape(12.dp),
                singleLine = true,
                // Cấu hình bàn phím để gợi ý Email
                keyboardOptions = KeyboardOptions(
                    capitalization = KeyboardCapitalization.None,
                    autoCorrect = false,
                    keyboardType = KeyboardType.Email,
                    imeAction = ImeAction.Next
                )
            )
            // --------------------------------

            Spacer(modifier = Modifier.height(16.dp))

            // Password
            OutlinedTextField(
                value = state.password,
                onValueChange = { viewModel.onEvent(RegisterEvent.OnPasswordChanged(it))
    },
                label = { Text("Mật khẩu") },
                leadingIcon = { Icon(Icons.Default.Lock, contentDescription = null) },
                trailingIcon = {
                    val image = if (passwordVisible) Icons.Filled.Visibility else
    Icons.Filled.VisibilityOff
                    IconButton(onClick = { passwordVisible = !passwordVisible }) {
                        Icon(imageVector = image, contentDescription = "Toggle Password")
                    }
                },
                visualTransformation = if (passwordVisible) VisualTransformation.None else
    PasswordVisualTransformation(),
                modifier = Modifier.fillMaxWidth(),
                shape = RoundedCornerShape(12.dp),
                singleLine = true,
                keyboardOptions = KeyboardOptions(
                    keyboardType = KeyboardType.Password,
                    imeAction = ImeAction.Done
                )
            )

            Text(
                text = "Mật khẩu phải có ít nhất 6 ký tự",
                style = MaterialTheme.typography.labelSmall,
                color = Color.Gray,
                modifier = Modifier.fillMaxWidth().padding(start = 8.dp, top = 4.dp)
            )

            Spacer(modifier = Modifier.height(32.dp))

            Button(
                onClick = { viewModel.onEvent(RegisterEvent.OnRegisterClicked) },
          modifier = Modifier
            .fillMaxWidth()
            .height(50.dp),
          shape = RoundedCornerShape(12.dp),
          enabled = !state.isLoading
        ) {
          Text(
            text = "Đăng ký",
            fontSize = 16.sp,
            fontWeight = FontWeight.Bold
          )
        }

        Spacer(modifier = Modifier.height(24.dp))

        Row(
          verticalAlignment = Alignment.CenterVertically,
          horizontalArrangement = Arrangement.Center
        ) {
          Text("Đã có tài khoản? ", color = Color.Gray)
          Text(
            text = "Đăng nhập",
            color = MaterialTheme.colorScheme.primary,
            fontWeight = FontWeight.Bold,
            modifier = Modifier.clickable { navigateBack() }
          )
        }

        Spacer(modifier = Modifier.height(50.dp))
      }
    }

    if (state.isLoading) {
      Box(
        modifier = Modifier
          .fillMaxSize()
          .background(Color.Black.copy(alpha = 0.5f))
          .clickable(enabled = false) {},
        contentAlignment = Alignment.Center
      ) {
        Card(
          shape = RoundedCornerShape(16.dp),
          colors = CardDefaults.cardColors(containerColor = Color.White)
```

```
            ) {
                Column(
                    modifier = Modifier.padding(24.dp),
                    horizontalAlignment = Alignment.CenterHorizontally
                ) {
                    CircularProgressIndicator()
                    Spacer(modifier = Modifier.height(16.dp))
                    Text("Đang tạo tài khoản...", style = MaterialTheme.typography.bodyMedium)
                }
            }
        }
    }
  }
}
```

feature_auth/src/main/java/com/example/feature_auth/ui/register/RegisterViewModel.kt

```
package com.example.feature_auth.ui.register

import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import com.example.feature_auth.domain.usecase.RegisterUseCase
import dagger.hilt.android.lifecycle.HiltViewModel
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.asStateFlow
import kotlinx.coroutines.flow.update
import kotlinx.coroutines.launch
import javax.inject.Inject

@HiltViewModel
class RegisterViewModel @Inject constructor(
  private val registerUseCase: RegisterUseCase
) : ViewModel() {
  private val _state = MutableStateFlow(RegisterState())
  val state = _state.asStateFlow()

  fun onEvent(event: RegisterEvent) {
    when (event) {
      is RegisterEvent.OnFullNameChanged -> _state.update { it.copy(fullName =
event.fullName) }
      is RegisterEvent.OnEmailChanged -> _state.update { it.copy(email = event.email) }
      is RegisterEvent.OnPasswordChanged -> _state.update { it.copy(password =
event.password) }
```

```
            RegisterEvent.OnRegisterClicked -> register()
            RegisterEvent.OnErrorDismissed -> _state.update { it.copy(error = null) }
        }
    }

    private fun register() {
        viewModelScope.launch {
            _state.update { it.copy(isLoading = true, error = null) }
            val currentState = state.value
            val result = registerUseCase(
                fullName = currentState.fullName,
                email = currentState.email,
                password = currentState.password
            )
            result.onSuccess {
                _state.update { it.copy(isLoading = false, registrationSuccess = true) }
            }.onFailure { exception ->
                // --- SỬA: Dịch lỗi sang tiếng Việt ---
                val errorMessage = when {
                    exception.message?.contains("email-already-in-use") == true -> "Email này đã
được sử dụng."
                    exception.message?.contains("badly formatted") == true -> "Định dạng email
không hợp lệ."
                    exception.message?.contains("Password should be at least") == true -> "Mật khẩu
phải có ít nhất 6 ký tự."
                    exception.message?.contains("network error") == true -> "Lỗi kết nối mạng."
                    else -> "Đăng ký thất bại: ${exception.message}"
                }
                // -----------------------------------
                _state.update {
                    it.copy(isLoading = false, error = errorMessage)
                }
            }
        }
    }
}
```

feature_auth/src/main/java/com/example/feature_auth/ui/splash/SplashScreen.kt

```
// D:\...\feature_auth\src\main\java\com\smartblood\auth\ui\splash\SplashScreen.kt

package com.example.feature_auth.ui.splash
```

```
// --- CÁC IMPORT MỚI CẦN THÊM ---
import androidx.compose.animation.core.Animatable
import androidx.compose.animation.core.RepeatMode
import androidx.compose.animation.core.animateDpAsState
import androidx.compose.animation.core.infiniteRepeatable
// --------------------------------

import androidx.compose.animation.core.animateFloatAsState
import androidx.compose.animation.core.tween
import androidx.compose.foundation.background
import androidx.compose.foundation.layout.Arrangement
import androidx.compose.foundation.layout.Box
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.Spacer
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.height
import androidx.compose.foundation.layout.offset // --- IMPORT MỚI ---
import androidx.compose.foundation.layout.size
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.filled.Favorite
import androidx.compose.material3.Icon
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.runtime.LaunchedEffect
import androidx.compose.runtime.collectAsState
import androidx.compose.runtime.getValue
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.runtime.setValue
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.alpha
import androidx.compose.ui.draw.scale // --- IMPORT MỚI ---
import androidx.compose.ui.graphics.Brush
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.Dp // --- IMPORT MỚI ---
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.hilt.navigation.compose.hiltViewModel
import com.smartblood.core.ui.theme.PrimaryRed
import com.smartblood.core.ui.theme.PrimaryRedDark
import com.smartblood.core.ui.theme.SmartBloodTheme
import kotlinx.coroutines.delay

@Composable
fun SplashScreen(
   viewModel: SplashViewModel = hiltViewModel(),
   navigateToLogin: () -> Unit,
   navigateToDashboard: () -> Unit
) {
   val isAuthenticated by viewModel.isAuthenticated.collectAsState()
   var startAnimation by remember { mutableStateOf(false) } // <--- Đổi tên biến alphaAnim
thành startAnimation để rõ nghĩa hơn

   // Kích hoạt animation và xử lý điều hướng
   LaunchedEffect(key1 = true) {
      startAnimation = true
      delay(3000L) // Tăng thời gian chờ lên 3 giây để xem animation rõ hơn

      // Sau khi chờ, nếu chưa có kết quả xác thực thì mặc định đi đến login
      if (isAuthenticated == null) {
         navigateToLogin()
      }
   }

   // Thêm một LaunchedEffect khác để xử lý điều hướng ngay khi có kết quả
   LaunchedEffect(key1 = isAuthenticated) {
      if (isAuthenticated != null) {
         if (isAuthenticated == true) {
            navigateToDashboard()
         } else {
            navigateToLogin()
         }
      }
   }

   // Gọi giao diện Splash và truyền vào công tắc animation
   SplashContent(startAnimation = startAnimation)
}

// Tách riêng phần giao diện để dễ dàng preview và tái sử dụng
@Composable
fun SplashContent(startAnimation: Boolean) { // <--- THAM SỐ ĐÃ THAY ĐỔI
```

```
// --- NÂNG CẤP 1: ANIMATION CHO HIỆU ỨNG "BƠM TIM" ---
val scale = remember { Animatable(1f) }
LaunchedEffect(startAnimation) {
   if (startAnimation) {
      scale.animateTo(
         targetValue = 1.1f, // Phóng to 10%
         animationSpec = infiniteRepeatable(
            animation = tween(durationMillis = 700), // Thời gian 1 nhịp
            repeatMode = RepeatMode.Reverse // Lặp ngược lại (phóng to -> thu nhỏ)
         )
      )
   }
}

// --- NÂNG CẤP 2: ANIMATION CHO HIỆU ỨNG "TRƯỢT LÊN" ---
val offsetY: Dp by animateDpAsState(
   targetValue = if (startAnimation) 0.dp else 100.dp,
   animationSpec = tween(durationMillis = 1500),
   label = "offset_animation"
)

// --- GIỮ NGUYÊN: ANIMATION CHO HIỆU ỨNG "MỜ DẦN" ---
val alpha: Float by animateFloatAsState(
   targetValue = if (startAnimation) 1f else 0f,
   animationSpec = tween(durationMillis = 1500),
   label = "alpha_animation"
)

Box(
   modifier = Modifier
      .fillMaxSize()
      .background(
         Brush.verticalGradient(
            colors = listOf(
               PrimaryRed,
               PrimaryRedDark
            )
         )
      ),
   contentAlignment = Alignment.Center
) {
   Column(
      horizontalAlignment = Alignment.CenterHorizontally,
            verticalArrangement = Arrangement.Center,
            modifier = Modifier.alpha(alpha) // Vẫn áp dụng hiệu ứng mờ dần
        ) {
            // NÂNG CẤP: Áp dụng hiệu ứng "Bơm Tim" vào Icon
            Icon(
                imageVector = Icons.Default.Favorite,
                contentDescription = "App Logo",
                modifier = Modifier
                    .size(120.dp)
                    .scale(scale.value), // <-- Dùng giá trị scale từ animation
                tint = Color.White
            )

            Spacer(modifier = Modifier.height(24.dp))

            // NÂNG CẤP: Áp dụng hiệu ứng "Trượt Lên" vào Text
            Text(
                modifier = Modifier.offset(y = offsetY), // <-- Dùng giá trị offset từ animation
                text = "Smart Blood Donation",
                color = Color.White,
                fontSize = 28.sp,
                fontWeight = FontWeight.Bold
            )
        }
    }
}

// Hàm Preview để xem trước giao diện ngay trong Android Studio
@Preview(showBackground = true, name = "Splash Screen Preview")
@Composable
fun SplashScreenPreview() {
    SmartBloodTheme {
        // Thay đổi tham số để xem trước trạng thái animation đã bắt đầu
        SplashContent(startAnimation = true)
    }
}
```

feature_auth/src/main/java/com/example/feature_auth/ui/splash/SplashViewModel.kt

```
// feature_auth/src/main/java/com/smartblood/auth/ui/splash/SplashViewModel.kt

package com.example.feature_auth.ui.splash
```

```
import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import com.example.feature_auth.domain.usecase.CheckUserAuthenticationUseCase
import dagger.hilt.android.lifecycle.HiltViewModel
import kotlinx.coroutines.delay
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.asStateFlow
import kotlinx.coroutines.launch
import javax.inject.Inject

@HiltViewModel
class SplashViewModel @Inject constructor(
    private val checkUserAuthenticationUseCase: CheckUserAuthenticationUseCase
) : ViewModel() {

    private val _isAuthenticated = MutableStateFlow<Boolean?>(null)
    val isAuthenticated = _isAuthenticated.asStateFlow()

    init {
        checkAuthentication()
    }

    private fun checkAuthentication() {
        viewModelScope.launch {
            // Thêm một khoảng trễ nhỏ (ví dụ 2 giây) để người dùng có thể thấy splash screen
            // Điều này cũng cho Firebase SDK thời gian để khởi tạo và kiểm tra trạng thái đăng
nhập.
            delay(2000L)
            _isAuthenticated.value = checkUserAuthenticationUseCase()
        }
    }
}
```

feature_auth/src/test/java/com/example/feature_auth/ExampleUnitTest.kt

```
package com.example.feature_auth

import org.junit.Test

import org.junit.Assert.*

/**
 * Example local unit test, which will execute on the development machine (host).
 *
 * See [testing documentation](http://d.android.com/tools/testing).
 */
class ExampleUnitTest {
    @Test
    fun addition_isCorrect() {
        assertEquals(4, 2 + 2)
    }
}
```

feature_chatbot/.gitignore

```
/build
```

feature_chatbot/build.gradle.kts

```
plugins {
    alias(libs.plugins.android.library)
    alias(libs.plugins.kotlin.android)
    alias(libs.plugins.kotlin.compose.compiler)

}

android {
    namespace = "com.example.feature_chatbot"
    compileSdk = 34

    defaultConfig {
        minSdk = 24

        testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
        consumerProguardFiles("consumer-rules.pro")
    }

    buildTypes {
        release {
            isMinifyEnabled = false
            proguardFiles(
                getDefaultProguardFile("proguard-android-optimize.txt"),
                "proguard-rules.pro"
            )
        }
    }
    compileOptions {
        sourceCompatibility = JavaVersion.VERSION_11
        targetCompatibility = JavaVersion.VERSION_11
    }
    kotlinOptions {
        jvmTarget = "11"
    }
}

dependencies {
    implementation(project(":core"))
    implementation(libs.androidx.core.ktx)
//    implementation(libs.androidx.appcompat)
//    implementation(libs.material)
    testImplementation(libs.junit)
    androidTestImplementation(libs.androidx.junit)
    androidTestImplementation(libs.androidx.espresso.core)
}
```

feature_chatbot/consumer-rules.pro

[File rỗng]

feature_chatbot/proguard-rules.pro

```
# Add project specific ProGuard rules here.
# You can control the set of applied configuration files using the
# proguardFiles setting in build.gradle.
#
# For more details, see
#   http://developer.android.com/guide/developing/tools/proguard.html

# If your project uses WebView with JS, uncomment the following
# and specify the fully qualified class name to the JavaScript interface
# class:
#-keepclassmembers class fqcn.of.javascript.interface.for.webview {
#   public *;
#}

# Uncomment this to preserve the line number information for
# debugging stack traces.
#-keepattributes SourceFile,LineNumberTable

# If you keep the line number information, uncomment this to
# hide the original source file name.
#-renamesourcefileattribute SourceFile
```

feature_chatbot/src/androidTest/java/com/example/feature_chatbot/ExampleInstrumentedTest.kt

```
package com.example.feature_chatbot

import androidx.test.platform.app.InstrumentationRegistry
import androidx.test.ext.junit.runners.AndroidJUnit4

import org.junit.Test
import org.junit.runner.RunWith

import org.junit.Assert.*

/**
 * Instrumented test, which will execute on an Android device.
 *
 * See [testing documentation](http://d.android.com/tools/testing).
 */
@RunWith(AndroidJUnit4::class)
class ExampleInstrumentedTest {
    @Test
    fun useAppContext() {
        // Context of the app under test.
        val appContext = InstrumentationRegistry.getInstrumentation().targetContext
        assertEquals("com.example.feature_chatbot.test", appContext.packageName)
    }
}
```

feature_chatbot/src/main/AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android">

</manifest>
```

feature_chatbot/src/test/java/com/example/feature_chatbot/ExampleUnitTest.kt

```
package com.example.feature_chatbot

import org.junit.Test

import org.junit.Assert.*

/**
 * Example local unit test, which will execute on the development machine (host).
 *
 * See [testing documentation](http://d.android.com/tools/testing).
 */
class ExampleUnitTest {
   @Test
   fun addition_isCorrect() {
      assertEquals(4, 2 + 2)
   }
}
```

feature_emergency/.gitignore

```
/build
```

feature_emergency/build.gradle.kts

```
plugins {
   alias(libs.plugins.android.library)
   alias(libs.plugins.kotlin.android)
   alias(libs.plugins.kotlin.compose.compiler)
   alias(libs.plugins.ksp)
}

android {
   namespace = "com.example.feature_emergency"
   compileSdk = 34

   defaultConfig {
      minSdk = 24

      testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
      consumerProguardFiles("consumer-rules.pro")
   }

   buildTypes {
      release {
         isMinifyEnabled = false
         proguardFiles(
            getDefaultProguardFile("proguard-android-optimize.txt"),
            "proguard-rules.pro"
         )
      }
   }
   compileOptions {
      sourceCompatibility = JavaVersion.VERSION_11
      targetCompatibility = JavaVersion.VERSION_11
```

```
    }
    kotlinOptions {
        jvmTarget = "11"
    }
}

dependencies {
    ksp(libs.hilt.compiler)
    implementation(project(":core"))
    implementation(libs.androidx.core.ktx)
    implementation(libs.firebase.firestore)
    implementation(libs.firebase.auth)
    implementation("androidx.compose.material:material-icons-extended:1.7.0")

    // Remote - Firebase
    // BoM để quản lý phiên bản các thư viện Firebase
    implementation(platform(libs.firebase.bom))
    // Thư viện cần thiết cho xác thực (lấy currentUser)
    implementation(libs.firebase.auth.ktx)
    // Thư viện cần thiết cho Firestore (hàm .toObject, .id, .collection, ...)
    implementation(libs.firebase.firestore.ktx)

    // Cần thiết cho các hàm coroutine của Firebase như .await()
    implementation(libs.kotlinx.coroutines.play.services)
    // Dòng hilt-android không cần thiết vì đã có trong core, nhưng để cũng không sao
    // implementation(libs.hilt.android)

    // THAY ĐỔI Ở ĐÂY
    // implementation(libs.androidx.material3) // <--- XÓA HOẶC CHÚ THÍCH DÒNG NÀY

    // THAY BẰNG CÁCH SỬ DỤNG BOM (Bill of Materials)
    implementation(platform(libs.androidx.compose.bom)) // Quan trọng: Khai báo BOM
    implementation(libs.androidx.compose.ui)
    implementation(libs.androidx.compose.material3) // Bây giờ dòng này sẽ hoạt động
    implementation(libs.androidx.compose.ui.tooling.preview)

    // ... các dependency khác
    testImplementation(libs.junit)
    androidTestImplementation(libs.androidx.junit)
    androidTestImplementation(libs.androidx.espresso.core)
    implementation(libs.androidx.hilt.navigation.compose)

}
```

feature_emergency/consumer-rules.pro

[File rỗng]

feature_emergency/proguard-rules.pro

```
# Add project specific ProGuard rules here.
# You can control the set of applied configuration files using the
# proguardFiles setting in build.gradle.
#
# For more details, see
#   http://developer.android.com/guide/developing/tools/proguard.html

# If your project uses WebView with JS, uncomment the following
# and specify the fully qualified class name to the JavaScript interface
# class:
#-keepclassmembers class fqcn.of.javascript.interface.for.webview {
#   public *;
#}

# Uncomment this to preserve the line number information for
# debugging stack traces.
#-keepattributes SourceFile,LineNumberTable

# If you keep the line number information, uncomment this to
# hide the original source file name.
#-renamesourcefileattribute SourceFile
```

feature_emergency/src/androidTest/java/com/example/feature_emergency/ExampleInstrumentedTest.kt

```
package com.example.feature_emergency

import androidx.test.platform.app.InstrumentationRegistry
import androidx.test.ext.junit.runners.AndroidJUnit4

import org.junit.Test
import org.junit.runner.RunWith

import org.junit.Assert.*

/**
 * Instrumented test, which will execute on an Android device.
 *
 * See [testing documentation](http://d.android.com/tools/testing).
 */
```

```
@RunWith(AndroidJUnit4::class)
class ExampleInstrumentedTest {
    @Test
    fun useAppContext() {
        // Context of the app under test.
        val appContext = InstrumentationRegistry.getInstrumentation().targetContext
        assertEquals("com.example.feature_emergency.test", appContext.packageName)
    }
}
```

feature_emergency/src/main/AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android">

</manifest>
```

feature_emergency/src/main/java/com/example/feature_emergency/data/dto/BloodRequestDto.kt

```
package com.example.feature_emergency.data.dto


import com.smartblood.core.domain.model.BloodRequest
import com.google.firebase.Timestamp

// Lớp DTO này khớp với cấu trúc trên Firestore
data class BloodRequestDto(
    val bloodType: String = "",
    val hospitalId: String = "",
    val hospitalName: String = "",
    val priority: String = "Trung bình",
    val quantity: Int = 0,
    val status: String = "ĐANG HOẠT ĐỘNG",
    val createdAt: Timestamp = Timestamp.now(),
    val preferredVolume: String = "350ml"

)

// Hàm chuyển đổi từ DTO sang Domain Model
fun BloodRequestDto.toDomain(id: String): BloodRequest {
    return BloodRequest(
        id = id,
        bloodType = this.bloodType,
        hospitalId = this.hospitalId,
        hospitalName = this.hospitalName,
        priority = this.priority,
        quantity = this.quantity,
        status = this.status,
        createdAt = this.createdAt.toDate(), // Việc chuyển đổi diễn ra an toàn ở đây
        preferredVolume = this.preferredVolume

    )
}
```

feature_emergency/src/main/java/com/example/feature_emergency/data/repository/EmergencyRepositoryImpl.kt

```
package com.example.feature_emergency.data.repository


import com.example.feature_emergency.data.dto.BloodRequestDto
import com.example.feature_emergency.data.dto.toDomain
import com.example.feature_emergency.domain.model.EmergencyDonationRecord
import com.smartblood.core.domain.model.BloodRequest
import com.smartblood.core.domain.model.Donor
import com.example.feature_emergency.domain.repository.EmergencyRepository
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.firestore.FirebaseFirestore
import com.google.firebase.firestore.Query
import com.google.firebase.firestore.toObject
import kotlinx.coroutines.async
import kotlinx.coroutines.awaitAll
import kotlinx.coroutines.channels.awaitClose
import kotlinx.coroutines.coroutineScope
import kotlinx.coroutines.flow.Flow
import kotlinx.coroutines.flow.callbackFlow
import kotlinx.coroutines.launch
import kotlinx.coroutines.tasks.await
import java.util.Date
import javax.inject.Inject
import kotlin.Result

/**
 * Lớp này triển khai các phương thức từ EmergencyRepository.
 * Nó chịu trách nhiệm lấy và gửi dữ liệu liên quan đến yêu cầu khẩn cấp từ Firebase
Firestore.
 * @param firestore Instance của FirebaseFirestore để tương tác với database.
 * @param auth Instance của FirebaseAuth để lấy thông tin người dùng hiện tại.
 */
class EmergencyRepositoryImpl @Inject constructor(
   private val firestore: FirebaseFirestore,
   private val auth: FirebaseAuth
) : EmergencyRepository {

   override fun getMyPledgedRequests(): Flow<Result<List<BloodRequest>>> =
callbackFlow {
      val userId = auth.currentUser?.uid
      if (userId == null) {
         trySend(Result.failure(Exception("Người dùng chưa đăng nhập.")))
         close()
         return@callbackFlow
      }

      // Query vào sub-collection "donors"
      val query = firestore.collectionGroup("donors").whereEqualTo("userId", userId)

      val listenerRegistration = query.addSnapshotListener { snapshot, error ->
         if (error != null) {
            trySend(Result.failure(error))
            return@addSnapshotListener
         }
         if (snapshot == null || snapshot.isEmpty) {
            trySend(Result.success(emptyList()))
            return@addSnapshotListener
         }

         // 1. Tạo Map lưu trữ: ParentID -> PledgedTime (Thời gian chấp nhận)
         // Lấy pledgedAt từ document donor
         val pledgedMap = snapshot.documents.associate { doc ->
            val parentId = doc.reference.parent.parent?.id ?: ""
            val pledgedAt = doc.getDate("pledgedAt") ?: Date()
            parentId to pledgedAt
         }

         val parentRequestRefs = snapshot.documents.mapNotNull {
it.reference.parent.parent }

         if (parentRequestRefs.isEmpty()) {
            trySend(Result.success(emptyList()))
            return@addSnapshotListener
         }
```

```
            launch {
                try {
                    val requestSnapshots = parentRequestRefs.map { it.get().await() }

                    val bloodRequests = requestSnapshots.mapNotNull { doc ->
                        val request = doc.toObject<BloodRequestDto>()?.toDomain(id = doc.id)

                        // 2. Gán thời gian chấp nhận vào Model
                        // Lấy thời gian từ Map đã tạo ở bước 1
                        val pledgedTime = pledgedMap[doc.id]

                        request?.copy(userPledgedDate = pledgedTime)
                    }

                    // 3. SẮP XẾP: Mới nhất lên đầu (Dựa vào userPledgedDate)
                    val sortedRequests = bloodRequests.sortedByDescending {
                        it.userPledgedDate ?: it.createdAt
                    }

                    trySend(Result.success(sortedRequests))
                } catch (e: Exception) {
                    trySend(Result.failure(e))
                }
            }
        }
        awaitClose { listenerRegistration.remove() }
    }
    override suspend fun getEmergencyDonationHistory():
Result<List<EmergencyDonationRecord>> {
        return try {
            val userId = auth.currentUser?.uid
                ?: return Result.failure(Exception("Người dùng chưa đăng nhập."))

            // 1. Dùng Collection Group để lấy tất cả docs trong các sub-collection 'donors'
            // mà có userId trùng với user hiện tại.
            val querySnapshot = firestore.collectionGroup("donors")
                .whereEqualTo("userId", userId)
                .get()
                .await()

            // 2. Xử lý bất đồng bộ để fetch thông tin Hospital từ Parent Document
            val historyList = coroutineScope {
```

```
        querySnapshot.documents.map { donorDoc ->
          async {
            // Lấy reference đến document cha
            val parentRef = donorDoc.reference.parent.parent
            var hospitalName = "Không xác định"

            if (parentRef != null) {
              val parentSnap = parentRef.get().await()
              hospitalName = parentSnap.getString("hospitalName") ?: "Không xác định"
            }

            // --- XỬ LÝ MAP LAB RESULT TỪ FIRESTORE ---
            // Firestore lưu object dưới dạng Map<String, Any>
            val labResultMap = donorDoc.get("labResult") as? Map<String, Any>
            val labResult = if (labResultMap != null) {
              com.smartblood.core.domain.model.LabResult(
                documentUrl = labResultMap["documentUrl"] as? String,
                conclusion = labResultMap["conclusion"] as? String,
                // Kiểm tra kỹ kiểu dữ liệu của recordedAt
                recordedAt = when (val rawDate = labResultMap["recordedAt"]) {
                  is com.google.firebase.Timestamp -> rawDate.toDate() // Chuẩn
Firestore
                  is Date -> rawDate // Trường hợp hiếm
                  else -> null
                }
              )
            } else {
              null
            }


            // Map dữ liệu vào Model
            EmergencyDonationRecord(
              id = donorDoc.id,
              requestId = parentRef?.id ?: "",
              hospitalName = hospitalName,
              pledgedAt = donorDoc.getDate("pledgedAt") ?: Date(),
              status = donorDoc.getString("status") ?: "Pending",
              userBloodType = donorDoc.getString("userBloodType") ?: "",
              certificateUrl = donorDoc.getString("certificateUrl"),
              rating = donorDoc.getLong("rating")?.toInt() ?: 0,
              review = donorDoc.getString("review"),
                    // Gán kết quả xét nghiệm vừa map được
                    labResult = labResult,
                    rejectionReason = donorDoc.getString("rejectionReason")
                )
            }
        }.awaitAll()
    }

    val sortedList = historyList.sortedByDescending { it.pledgedAt }

    // Sắp xếp theo ngày mới nhất
    Result.success(sortedList)
  } catch (e: Exception) {
    Result.failure(e)
  }
}

override suspend fun acceptEmergencyRequest(requestId: String, donorInfo: Donor):
Result<Unit> {
    return try {
        val userId = auth.currentUser?.uid
            ?: return Result.failure(Exception("Người dùng chưa đăng nhập."))

        // Tạo một document mới trong sub-collection với ID là userId
        firestore.collection("blood_requests")
            .document(requestId)
            .collection("donors")
            .document(userId)
            .set(donorInfo)
            .await()

        Result.success(Unit)
    } catch (e: Exception) {
        Result.failure(e)
    }
}
/**
 * Triển khai phương thức tạo một yêu cầu khẩn cấp mới (logic cũ của bạn).
 */
override suspend fun createEmergencyRequest(request: BloodRequest): Result<Unit> {
    // Đây là nơi bạn sẽ triển khai logic để tạo request mới, nếu cần.
    // Ví dụ:
    return try {
        firestore.collection("blood_requests").add(request).await()
        Result.success(Unit)
    } catch (e: Exception) {
        Result.failure(e)
    }
}

/**
 * Triển khai phương thức lấy các yêu cầu do người dùng hiện tại tạo (logic cũ của bạn).
 */
override suspend fun getMyRequests(): Result<List<BloodRequest>> {
    // Đây là nơi bạn sẽ triển khai logic để lấy các request của người dùng, nếu cần.
    // Ví dụ:
    return try {
        val currentUser = auth.currentUser ?: return Result.failure(Exception("Người dùng
chưa đăng nhập"))
        val snapshot = firestore.collection("blood_requests")
            .whereEqualTo("creatorId", currentUser.uid) // Giả sử có trường creatorId
            .get()
            .await()
        val requests = snapshot.toObjects(BloodRequest::class.java)
        Result.success(requests)
    } catch (e: Exception) {
        Result.failure(e)
    }
}

// --- TRIỂN KHAI PHƯƠNG THỨC MỚI ---

/**
 * Lấy tất cả các yêu cầu máu đang hoạt động từ Firestore.
 * Các yêu cầu được sắp xếp theo ngày tạo mới nhất.
 */
override fun getActiveEmergencyRequests(): Flow<Result<List<BloodRequest>>> =
callbackFlow {
    val query = firestore.collection("blood_requests")
        .whereEqualTo("status", "ĐANG HOẠT ĐỘNG")
        .orderBy("createdAt", Query.Direction.DESCENDING)

    val listener = query.addSnapshotListener { snapshot, error ->
        if (error != null) {
            trySend(Result.failure(error))
            return@addSnapshotListener
        }
        if (snapshot != null) {
            val requests = snapshot.documents.mapNotNull { doc ->
                doc.toObject<BloodRequestDto>()?.toDomain(id = doc.id)
            }
            trySend(Result.success(requests))
        }
    }
    // Hủy listener khi Flow bị hủy
    awaitClose { listener.remove() }
}



}
```

feature_emergency/src/main/java/com/example/feature_emergency/di/EmergencyModule.kt

```
package com.example.feature_emergency.di

import com.example.feature_emergency.data.repository.EmergencyRepositoryImpl
import com.example.feature_emergency.domain.repository.EmergencyRepository
import dagger.Binds
import dagger.Module
import dagger.hilt.InstallIn
import dagger.hilt.components.SingletonComponent
import javax.inject.Singleton

@Module
@InstallIn(SingletonComponent::class)
abstract class EmergencyModule {

    @Binds
    @Singleton
    abstract fun bindEmergencyRepository(
        emergencyRepositoryImpl: EmergencyRepositoryImpl
    ): EmergencyRepository

}
```

feature_emergency/src/main/java/com/example/feature_emergency/domain/model/EmergencyDonationRecord.kt

```
package com.example.feature_emergency.domain.model

import com.smartblood.core.domain.model.LabResult
import java.util.Date

data class EmergencyDonationRecord(
    val id: String = "",
    val requestId: String = "",
    val hospitalName: String = "",
    val pledgedAt: Date = Date(),
    val status: String = "",
    val userBloodType: String = "",
    // Các field sau khi hoàn thành
    val certificateUrl: String? = null,
    val rating: Int = 0,
    val review: String? = null,
    val labResult: LabResult? = null,
    val rejectionReason: String? = null
)
```

feature_emergency/src/main/java/com/example/feature_emergency/domain/repository/EmergencyRepository.kt

```
package com.example.feature_emergency.domain.repository

import com.example.feature_emergency.domain.model.EmergencyDonationRecord
import com.smartblood.core.domain.model.BloodRequest
import com.smartblood.core.domain.model.Donor
import kotlinx.coroutines.flow.Flow
import kotlin.Result

interface EmergencyRepository {
    // Giữ lại các hàm cũ nếu cần
    suspend fun createEmergencyRequest(request: BloodRequest): Result<Unit>
    suspend fun getMyRequests(): Result<List<BloodRequest>>

    // **HÀM MỚI**: Lấy tất cả các yêu cầu khẩn cấp đang hoạt động
    fun getActiveEmergencyRequests(): Flow<Result<List<BloodRequest>>>
    suspend fun acceptEmergencyRequest(requestId: String, donorInfo: Donor):
Result<Unit>
    fun getMyPledgedRequests(): Flow<Result<List<BloodRequest>>>
```

```
  suspend fun getEmergencyDonationHistory(): Result<List<EmergencyDonationRecord>>
}
```

feature_emergency/src/main/java/com/example/feature_emergency/domain/usecase/AcceptEmergencyRequestUseCase.kt

```
package com.example.feature_emergency.domain.usecase

import com.smartblood.core.domain.model.Donor
import com.example.feature_emergency.domain.repository.EmergencyRepository
import com.smartblood.core.domain.model.UserProfile // Import từ feature_profile

import javax.inject.Inject

class AcceptEmergencyRequestUseCase @Inject constructor(
  private val emergencyRepository: EmergencyRepository
) {
  suspend operator fun invoke(requestId: String, userProfile: UserProfile, volume: String): 
Result<Unit> {
    // Kiểm tra thông tin cần thiết từ profile
    if (userProfile.uid.isBlank() || userProfile.fullName.isBlank()) {
      return Result.failure(Exception("Thông tin người dùng không đầy đủ."))
    }

    // Tạo đối tượng Donor từ UserProfile
    val donorInfo = Donor(
      userId = userProfile.uid,
      userName = userProfile.fullName,
      userPhone = userProfile.phoneNumber ?: "",
      userBloodType = userProfile.bloodType ?: "N/A",
      // pledgedAt và status sẽ dùng giá trị mặc định
      pledgedVolume = volume,
      status = "Pending"



    )

    // Gọi phương thức từ repository
    return emergencyRepository.acceptEmergencyRequest(requestId, donorInfo)
  }
}
```

feature_emergency/src/main/java/com/example/feature_emergency/domain/usecase/GetActiveEmergencyRequestsUseCase.kt

```
package com.example.feature_emergency.domain.usecase

import com.example.feature_emergency.domain.repository.EmergencyRepository
import javax.inject.Inject

class GetActiveEmergencyRequestsUseCase @Inject constructor(
   private val repository: EmergencyRepository
) {
   operator fun invoke() = repository.getActiveEmergencyRequests()
}
```

feature_emergency/src/main/java/com/example/feature_emergency/domain/usecase/GetActiveRequestsUseCase.kt

```
//package com.example.feature_emergency.domain.usecase
//
//import com.example.feature_emergency.domain.repository.EmergencyRepository
//import javax.inject.Inject
//
//class GetActiveRequestsUseCase @Inject constructor(
//    private val repository: EmergencyRepository
//) {
//    suspend operator fun invoke() = repository.getActiveRequests()
//}
```

feature_emergency/src/main/java/com/example/feature_emergency/domain/usecase/GetEmergencyHistoryUseCase.kt

```
package com.example.feature_emergency.domain.usecase

import com.example.feature_emergency.domain.repository.EmergencyRepository
import javax.inject.Inject

class GetEmergencyHistoryUseCase @Inject constructor(
   private val repository: EmergencyRepository
) {
   suspend operator fun invoke() = repository.getEmergencyDonationHistory()
}
```

feature_emergency/src/main/java/com/example/feature_emergency/domain/usecase/GetMyPledgedRequestsUseCase.kt

```
package com.example.feature_emergency.domain.usecase

import com.example.feature_emergency.domain.repository.EmergencyRepository
import javax.inject.Inject

class GetMyPledgedRequestsUseCase @Inject constructor(
    private val repository: EmergencyRepository
) {
    operator fun invoke() = repository.getMyPledgedRequests()
}
```

feature_emergency/src/main/java/com/example/feature_emergency/ui/EmergencyListContract.kt

```
package com.example.feature_emergency.ui

import com.smartblood.core.domain.model.BloodRequest

data class EmergencyListState(
    val isLoading: Boolean = true,
    val requests: List<BloodRequest> = emptyList(),
    val error: String? = null,
    val acceptSuccess: Boolean = false
)

sealed class EmergencyListEvent {
    data class OnAcceptClick(val request: BloodRequest) : EmergencyListEvent()
    object OnDialogDismiss : EmergencyListEvent()
}
```

feature_emergency/src/main/java/com/example/feature_emergency/ui/history/EmergencyHistoryScreen.kt

```
package com.example.feature_emergency.ui.history

import androidx.compose.foundation.BorderStroke
import androidx.compose.foundation.background
import androidx.compose.foundation.border
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.items
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.automirrored.filled.ArrowBack
import androidx.compose.material.icons.filled.Description
import androidx.compose.material.icons.filled.Info
import androidx.compose.material.icons.filled.Star
import androidx.compose.material.icons.filled.VerifiedUser
import androidx.compose.material.icons.outlined.StarBorder
import androidx.compose.material3.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.platform.LocalUriHandler
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.hilt.navigation.compose.hiltViewModel
import com.example.feature_emergency.domain.model.EmergencyDonationRecord
import java.text.SimpleDateFormat
import java.util.Locale

@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun EmergencyHistoryScreen(
    viewModel: EmergencyHistoryViewModel = hiltViewModel(),
    onNavigateBack: () -> Unit
) {
    val state by viewModel.state.collectAsState()

    Scaffold(
        topBar = {
            TopAppBar(
                title = { Text("Lịch sử hiến máu khẩn cấp") },
                navigationIcon = {
                    IconButton(onClick = onNavigateBack) {
                        Icon(Icons.AutoMirrored.Filled.ArrowBack, "Back")
                    }
                }
            )
        }
    ) { padding ->
        if (state.isLoading) {
            Box(Modifier.fillMaxSize(), contentAlignment = Alignment.Center) {
                CircularProgressIndicator()
            }
        } else if (state.history.isEmpty()) {
            Box(Modifier.fillMaxSize(), contentAlignment = Alignment.Center) {
                Text("Chưa có lịch sử hiến máu khẩn cấp.")
            }
        } else {
            LazyColumn(
                modifier = Modifier.padding(padding).padding(16.dp),
                verticalArrangement = Arrangement.spacedBy(12.dp)
            ) {
                items(state.history) { record ->
                    EmergencyHistoryItem(record = record)
                }
            }
        }
    }
}

/**
 * Component hiển thị item lịch sử khẩn cấp.
 * Được public để DonationHistoryScreen (Module Profile) có thể gọi dùng chung.
 */
@Composable
fun EmergencyHistoryItem(record: EmergencyDonationRecord) {
    val dateFormat = remember { SimpleDateFormat("HH:mm - dd/MM/yyyy",
Locale.getDefault()) }
    val uriHandler = LocalUriHandler.current

    // --- 1. CHUẨN HÓA DỮ LIỆU ĐẦU VÀO ---
    // Loại bỏ khoảng trắng thừa và viết hoa toàn bộ để so sánh chính xác
    val normalizedStatus = record.status.trim().uppercase()

    // Kiểm tra lý do từ chối (null safety)
    val hasRejectionReason = !record.rejectionReason.isNullOrBlank()

    // --- 2. XÁC ĐỊNH TRẠNG THÁI ---
    // Logic ưu tiên: Nếu có lý do từ chối -> Chắc chắn là bị từ chối
    val isRejected = hasRejectionReason ||
            normalizedStatus == "REJECTED" ||
            normalizedStatus == "CANCELLED" ||
            normalizedStatus == "BỊ TỪ CHỐI"

    val (statusColor, statusBgColor, statusText) = when {
        // Trạng thái Hoàn thành
        normalizedStatus == "COMPLETED" || normalizedStatus == "ĐÃ HIẾN" ->
            Triple(Color(0xFF4CAF50), Color(0xFFE8F5E9), "Đã hiến")

        // Trạng thái Từ chối (Check status HOẶC có lý do)
        isRejected ->
            Triple(Color(0xFFD32F2F), Color(0xFFFFEBEE), "Đã bị từ chối")

        // Trạng thái Chờ (Pending, Pledged)
        normalizedStatus == "PENDING" || normalizedStatus == "PLEDGED" ->
            Triple(Color(0xFFFF9800), Color(0xFFFFF3E0), "Đang chờ duyệt")

        // Trường hợp lạ
        else -> Triple(Color.Gray, Color(0xFFF5F5F5), record.status)
    }

    // Viền đỏ nếu bị từ chối
    val cardBorder = if (isRejected) BorderStroke(1.dp, Color(0xFFFFCDD2)) else null

    Card(
        modifier = Modifier.fillMaxWidth(),
        elevation = CardDefaults.cardElevation(defaultElevation = if (isRejected) 0.dp else
2.dp),
        colors = CardDefaults.cardColors(containerColor = Color.White),
        shape = RoundedCornerShape(12.dp),
        border = cardBorder
    ) {
        Column(modifier = Modifier.padding(16.dp)) {
            // --- HEADER ---
            Row(verticalAlignment = Alignment.Top) {
                Column(modifier = Modifier.weight(1f)) {
                    Text(
                        text = record.hospitalName,
                        style = MaterialTheme.typography.titleMedium,
                        fontWeight = FontWeight.Bold
                    )
                    // Ẩn chữ "Yêu cầu khẩn cấp" nếu bị từ chối để đỡ rối mắt (hoặc giữ lại tùy ý)
                    if (!isRejected) {
                        Text(
                            text = "Yêu cầu khẩn cấp",
                            style = MaterialTheme.typography.bodySmall,
                            color = Color.Red
                        )
                }
            }
            // Badge trạng thái
            Surface(
                color = statusBgColor,
                shape = RoundedCornerShape(16.dp)
            ) {
                Text(
                    text = statusText,
                    color = statusColor,
                    style = MaterialTheme.typography.labelSmall,
                    modifier = Modifier.padding(horizontal = 8.dp, vertical = 4.dp),
                    fontWeight = FontWeight.Bold
                )
            }
        }

        HorizontalDivider(
            modifier = Modifier.padding(vertical = 12.dp),
            color = Color.LightGray.copy(alpha = 0.3f)
        )

        // --- THÔNG TIN CHI TIẾT ---
        Row(
            horizontalArrangement = Arrangement.SpaceBetween,
            modifier = Modifier.fillMaxWidth()
        ) {
            Text("Thời gian chấp nhận:", color = Color.Gray, style =
MaterialTheme.typography.bodyMedium)
            Text(dateFormat.format(record.pledgedAt), fontWeight = FontWeight.SemiBold,
style = MaterialTheme.typography.bodyMedium)
        }
        Spacer(modifier = Modifier.height(4.dp))
        Row(
            horizontalArrangement = Arrangement.SpaceBetween,
            modifier = Modifier.fillMaxWidth()
        ) {
            Text("Nhóm máu hiến:", color = Color.Gray, style =
MaterialTheme.typography.bodyMedium)
            Text(record.userBloodType, fontWeight = FontWeight.Bold, color = Color.Red, style
= MaterialTheme.typography.bodyMedium)
        }
        // --- HIỂN THỊ LÝ DO TỪ CHỐI (QUAN TRỌNG) ---
        if (isRejected && hasRejectionReason) {
          Spacer(modifier = Modifier.height(12.dp))
          Column(
            modifier = Modifier
              .fillMaxWidth()
              .background(Color(0xFFFFF5F5), RoundedCornerShape(8.dp)) // Nền đỏ rất
nhạt
              .border(1.dp, Color(0xFFFFCDD2), RoundedCornerShape(8.dp))
              .padding(12.dp)
          ) {
            Row(verticalAlignment = Alignment.CenterVertically) {
              Icon(
                imageVector = Icons.Default.Info,
                contentDescription = null,
                tint = Color(0xFFD32F2F),
                modifier = Modifier.size(16.dp)
              )
              Spacer(modifier = Modifier.width(8.dp))
              Text(
                text = "Lý do từ chối:",
                style = MaterialTheme.typography.labelLarge,
                fontWeight = FontWeight.Bold,
                color = Color(0xFFD32F2F)
              )
            }
            Spacer(modifier = Modifier.height(4.dp))
            Text(
              text = record.rejectionReason ?: "",
              style = MaterialTheme.typography.bodyMedium,
              color = Color(0xFFB71C1C), // Đỏ sẫm
              fontStyle = androidx.compose.ui.text.font.FontStyle.Italic
            )
          }
        }

        // --- HIỂN THỊ LAB RESULT ---
        // Chỉ hiện nếu KHÔNG bị từ chối
        if (!isRejected && record.labResult != null) {
          Spacer(modifier = Modifier.height(12.dp))
          Column(
            modifier = Modifier
              .fillMaxWidth()
              .background(Color(0xFFE3F2FD).copy(alpha = 0.5f),
RoundedCornerShape(8.dp))
              .padding(12.dp)
          ) {
            Text(
              text = "Kết quả xét nghiệm & Lời dặn:",
              style = MaterialTheme.typography.labelMedium,
              fontWeight = FontWeight.Bold,
              color = Color(0xFF1565C0)
            )

            val conclusion = record.labResult.conclusion
            if (!conclusion.isNullOrEmpty()) {
              Spacer(modifier = Modifier.height(4.dp))
              Text(
                text = conclusion,
                style = MaterialTheme.typography.bodyMedium,
                color = Color.Black.copy(alpha = 0.8f)
              )
            }

            val docUrl = record.labResult.documentUrl
            if (!docUrl.isNullOrEmpty()) {
              Spacer(modifier = Modifier.height(8.dp))
              Button(
                onClick = { uriHandler.openUri(docUrl) },
                modifier = Modifier.fillMaxWidth().height(36.dp),
                colors = ButtonDefaults.buttonColors(
                  containerColor = Color(0xFF2196F3),
                  contentColor = Color.White
                ),
                contentPadding = PaddingValues(0.dp),
                shape = RoundedCornerShape(6.dp)
              ) {
                Icon(Icons.Default.Description, null, modifier = Modifier.size(14.dp))
                Spacer(modifier = Modifier.width(6.dp))
                Text("Xem file kết quả (PDF)", style =
MaterialTheme.typography.labelMedium)
              }
            }
          }
        }
        // --- ĐÁNH GIÁ ---
        // Chỉ hiện khi đã hoàn thành hoặc có rating
        if ((normalizedStatus == "COMPLETED" || normalizedStatus == "ĐÃ HIẾN") ||
record.rating > 0) {
          Spacer(modifier = Modifier.height(12.dp))
          Row(
            verticalAlignment = Alignment.CenterVertically,
            modifier = Modifier
              .background(Color(0xFFFFF8E1), RoundedCornerShape(8.dp))
              .padding(8.dp)
              .fillMaxWidth()
          ) {
            Column {
              Row(verticalAlignment = Alignment.CenterVertically) {
                Text("Đánh giá:", style = MaterialTheme.typography.labelMedium,
fontWeight = FontWeight.Bold)
                Spacer(modifier = Modifier.width(8.dp))
                repeat(5) { index ->
                  Icon(
                    imageVector = if (index < record.rating) Icons.Filled.Star else
Icons.Outlined.StarBorder,
                    contentDescription = null,
                    tint = Color(0xFFFFC107),
                    modifier = Modifier.size(14.dp)
                  )
                }
              }
              if (!record.review.isNullOrBlank()) {
                Spacer(modifier = Modifier.height(2.dp))
                Text(
                  text = "\"${record.review}\"",
                  style = MaterialTheme.typography.bodySmall,
                  fontStyle = androidx.compose.ui.text.font.FontStyle.Italic,
                  color = Color.DarkGray
                )
              }
            }
          }
        }

        // --- CHỨNG NHẬN ---
        // Chỉ hiện khi KHÔNG bị từ chối
        if (!isRejected && !record.certificateUrl.isNullOrBlank()) {
```

```
                Spacer(modifier = Modifier.height(12.dp))
                OutlinedButton(
                    onClick = { uriHandler.openUri(record.certificateUrl) },
                    modifier = Modifier.fillMaxWidth().height(40.dp),
                    colors = ButtonDefaults.outlinedButtonColors(
                        contentColor = Color(0xFF388E3C)
                    ),
                    border = BorderStroke(1.dp, Color(0xFF388E3C).copy(alpha = 0.5f)),
                    shape = RoundedCornerShape(8.dp)
                ) {
                    Icon(Icons.Default.VerifiedUser, null, modifier = Modifier.size(16.dp))
                    Spacer(modifier = Modifier.width(8.dp))
                    Text("Chứng Nhận Hiến Máu", style = MaterialTheme.typography.labelLarge)
                }
            }
        }
    }
}
```

feature_emergency/src/main/java/com/example/feature_emergency/ui/history/EmergencyHistoryViewModel.kt

```
package com.example.feature_emergency.ui.history

import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import com.example.feature_emergency.domain.model.EmergencyDonationRecord
import com.example.feature_emergency.domain.usecase.GetEmergencyHistoryUseCase
import dagger.hilt.android.lifecycle.HiltViewModel
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.asStateFlow
import kotlinx.coroutines.flow.update
import kotlinx.coroutines.launch
import javax.inject.Inject

data class EmergencyHistoryState(
    val isLoading: Boolean = true,
    val history: List<EmergencyDonationRecord> = emptyList(),
    val error: String? = null
)

@HiltViewModel
class EmergencyHistoryViewModel @Inject constructor(
    private val getEmergencyHistoryUseCase: GetEmergencyHistoryUseCase
) : ViewModel() {

    private val _state = MutableStateFlow(EmergencyHistoryState())
    val state = _state.asStateFlow()

    init {
        loadHistory()
    }

    private fun loadHistory() {
        viewModelScope.launch {
            _state.update { it.copy(isLoading = true) }
            getEmergencyHistoryUseCase()
                .onSuccess { list ->
                    _state.update { it.copy(isLoading = false, history = list) }
                }
                .onFailure { error ->
                    _state.update { it.copy(isLoading = false, error = error.message) }
                }
        }
    }
}
```

feature_emergency/src/test/java/com/example/feature_emergency/ExampleUnitTest.kt

```
package com.example.feature_emergency

import org.junit.Test

import org.junit.Assert.*

/**
 * Example local unit test, which will execute on the development machine (host).
 *
 * See [testing documentation](http://d.android.com/tools/testing).
 */
class ExampleUnitTest {
    @Test
    fun addition_isCorrect() {
        assertEquals(4, 2 + 2)
    }
}
```

feature_map_booking/.gitignore

```
/build
```

feature_map_booking/build.gradle.kts

```
plugins {
    alias(libs.plugins.android.library)
    alias(libs.plugins.kotlin.android)
    alias(libs.plugins.kotlin.compose.compiler)
    alias(libs.plugins.hilt)
    alias(libs.plugins.ksp)
    // THÊM MỚI: Plugin để quản lý secrets, bao gồm API key
//    alias(libs.plugins.maps.secrets)
}

android {
    namespace = "com.smartblood.mapbooking"
    compileSdk = 34

    defaultConfig {
        minSdk = 24
        testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
        consumerProguardFiles("consumer-rules.pro")
    }

    buildTypes {
        release {
            isMinifyEnabled = false
            proguardFiles(
                getDefaultProguardFile("proguard-android-optimize.txt"),
                "proguard-rules.pro"
            )
        }
    }
    compileOptions {
        sourceCompatibility = JavaVersion.VERSION_11
        targetCompatibility = JavaVersion.VERSION_11
    }
    kotlinOptions {
        jvmTarget = "11"
    }
    buildFeatures {
        compose = true
    }
}

dependencies {
    implementation(project(":core"))
    implementation(libs.androidx.core.ktx)

    // THÊM MỚI: Google Maps & Location Services
    implementation(libs.play.services.maps)
    implementation(libs.maps.compose) // Google Maps for Jetpack Compose
    implementation(libs.play.services.location)
    implementation(libs.trackasia.sdk)
    implementation(libs.trackasia.annotation.plugin)

    // Hilt
    implementation(libs.hilt.android)
    implementation(libs.firebase.auth)
    ksp(libs.hilt.compiler)
    implementation(libs.androidx.hilt.navigation.compose)

    // Lifecycle & Coroutines
    implementation(libs.androidx.lifecycle.runtime.ktx)
    implementation(libs.androidx.lifecycle.viewmodel.compose)
    implementation(libs.kotlinx.coroutines.core)
    implementation(libs.kotlinx.coroutines.android)
    // Cần thiết để coroutines hoạt động với Task API của Firebase/Play Services
    implementation(libs.kotlinx.coroutines.play.services)

    // Firebase
    implementation(platform(libs.firebase.bom))
    implementation(libs.firebase.firestore.ktx)

    // Jetpack Compose
    implementation(libs.androidx.compose.ui)
    implementation(libs.androidx.compose.material3)
    implementation(libs.androidx.compose.ui.tooling.preview)
    debugImplementation(libs.androidx.compose.ui.tooling)
    implementation(libs.androidx.compose.material.icons.extended)

    // Testing
    testImplementation(libs.junit)
    androidTestImplementation(libs.androidx.junit)
    androidTestImplementation(libs.androidx.espresso.core)
}
```

feature_map_booking/consumer-rules.pro

[File rỗng]

feature_map_booking/proguard-rules.pro

```
# Add project specific ProGuard rules here.
# You can control the set of applied configuration files using the
# proguardFiles setting in build.gradle.
#
# For more details, see
#   http://developer.android.com/guide/developing/tools/proguard.html

# If your project uses WebView with JS, uncomment the following
# and specify the fully qualified class name to the JavaScript interface
# class:
#-keepclassmembers class fqcn.of.javascript.interface.for.webview {
#   public *;
#}

# Uncomment this to preserve the line number information for
# debugging stack traces.
#-keepattributes SourceFile,LineNumberTable

# If you keep the line number information, uncomment this to
# hide the original source file name.
#-renamesourcefileattribute SourceFile
```

feature_map_booking/src/androidTest/java/com/example/feature_map_booking/ExampleInstrumentedTest.kt

```
package com.example.feature_map_booking

import androidx.test.platform.app.InstrumentationRegistry
import androidx.test.ext.junit.runners.AndroidJUnit4

import org.junit.Test
import org.junit.runner.RunWith

import org.junit.Assert.*

/**
 * Instrumented test, which will execute on an Android device.
 *
 * See [testing documentation](http://d.android.com/tools/testing).
 */
```

```kotlin
@RunWith(AndroidJUnit4::class)
class ExampleInstrumentedTest {
    @Test
    fun useAppContext() {
        // Context of the app under test.
        val appContext = InstrumentationRegistry.getInstrumentation().targetContext
        assertEquals("com.example.feature_map_booking.test", appContext.packageName)
    }
}
```

feature_map_booking/src/main/AndroidManifest.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android">

</manifest>
```

feature_map_booking/src/main/java/com/example/feature_map_booking/domain/data/dto/HospitalDto.kt

```kotlin
package com.example.feature_map_booking.domain.data.dto

import com.google.firebase.firestore.PropertyName

data class HospitalDto(
    val name: String = "",
    val address: String = "",
    val status: String = "",
    val licenseUrl: String = "",

    // Trong ảnh Firestore, location là một Map chứa lat/lng
    val location: Map<String, Double> = emptyMap(),

    // Inventory chứa số lượng máu (A+: 40, B+: 100...)
    val inventory: Map<String, Int> = emptyMap(),

    // Các trường này có thể chưa có trên Web, nên để default
    val phone: String = "",
    val workingHours: String = ""
)
```

feature_map_booking/src/main/java/com/example/feature_map_booking/domain/data/mapper/HospitalMapper.kt

```kotlin
package com.example.feature_map_booking.domain.data.mapper
```

```
import com.example.feature_map_booking.domain.data.dto.HospitalDto
import com.google.firebase.firestore.GeoPoint
import com.smartblood.core.domain.model.Hospital

fun HospitalDto.toDomain(id: String): Hospital {
    val lat = this.location["lat"] ?: 0.0
    val lng = this.location["lng"] ?: 0.0
    val geoPoint = GeoPoint(lat, lng)

    val availableTypes = this.inventory.filter { it.value > 0 }.map { it.key }

    // Xử lý hiển thị nếu dữ liệu trên Firebase bị thiếu
    val displayPhone = if (this.phone.isNotBlank()) this.phone else "Đang cập nhật SĐT"
    val displayHours = if (this.workingHours.isNotBlank()) this.workingHours else "Giờ hành chính"

    return Hospital(
        id = id,
        name = this.name,
        address = this.address,
        location = geoPoint,
        phone = displayPhone,
        workingHours = displayHours,
        availableBloodTypes = availableTypes,
        inventory = this.inventory

    )
}
```

feature_map_booking/src/main/java/com/example/feature_map_booking/domain/data/repository/MapBookingRepositoryImpl.kt

```
package com.example.feature_map_booking.domain.data.repository
// feature_map_booking/src/main/java/com/smartblood/mapbooking/data/repository/MapBookingRepositoryImpl.kt

import com.example.feature_map_booking.domain.data.dto.HospitalDto
import com.example.feature_map_booking.domain.data.mapper.toDomain
import com.example.feature_map_booking.domain.repository.MapBookingRepository
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.firestore.FirebaseFirestore
import com.smartblood.core.domain.model.Appointment
import com.smartblood.core.domain.model.Hospital
import com.smartblood.core.domain.model.TimeSlot
import kotlinx.coroutines.channels.awaitClose
import kotlinx.coroutines.flow.Flow
import kotlinx.coroutines.flow.callbackFlow
import kotlinx.coroutines.tasks.await
import java.util.*
import javax.inject.Inject
import kotlin.Result

class MapBookingRepositoryImpl @Inject constructor(
  private val firestore: FirebaseFirestore,
  private val auth: FirebaseAuth
) : MapBookingRepository {

  // --- CẬP NHẬT: Lấy danh sách bệnh viện thật từ Firestore ---
  override suspend fun getNearbyHospitals(lat: Double, lng: Double, radiusKm: Double):
Result<List<Hospital>> {
    return try {
      // Lấy các bệnh viện có status là "Đã duyệt" (theo như trong ảnh console)
      val snapshot = firestore.collection("hospitals")
        .whereEqualTo("status", "Đã duyệt")
        .get()
        .await()

      val hospitals = snapshot.documents.mapNotNull { doc ->
        // Convert document sang DTO rồi sang Domain Model
        doc.toObject(HospitalDto::class.java)?.toDomain(doc.id)
      }

      // TODO: (Nâng cao) Có thể lọc theo khoảng cách radiusKm ở đây nếu muốn
      // Hiện tại trả về toàn bộ danh sách đã duyệt
      Result.success(hospitals)
    } catch (e: Exception) {
      Result.failure(e)
    }
  }

  // --- CẬP NHẬT: Lấy chi tiết bệnh viện thật ---
  override suspend fun getHospitalDetails(hospitalId: String): Result<Hospital> {
    return try {
      val document = firestore.collection("hospitals")
        .document(hospitalId)
        .get()
```

```
            .await()

        val hospitalDto = document.toObject(HospitalDto::class.java)

        if (hospitalDto != null) {
            Result.success(hospitalDto.toDomain(document.id))
        } else {
            Result.failure(Exception("Hospital not found"))
        }
    } catch (e: Exception) {
        Result.failure(e)
    }
}

// ... Giữ nguyên các hàm getAvailableSlots, bookAppointment, getMyAppointments ...
// (Các hàm này đã viết đúng logic Firestore ở file cũ của bạn, không cần sửa)

override suspend fun getAvailableSlots(hospitalId: String, date: Date):
Result<List<TimeSlot>> {
    // Logic cũ của bạn ok
    return try {
        val calendar = Calendar.getInstance().apply { time = date }
        calendar.set(Calendar.HOUR_OF_DAY, 0); calendar.set(Calendar.MINUTE, 0);
calendar.set(Calendar.SECOND, 0)
        val startOfDay = calendar.time
        calendar.set(Calendar.HOUR_OF_DAY, 23); calendar.set(Calendar.MINUTE, 59);
calendar.set(Calendar.SECOND, 59)
        val endOfDay = calendar.time

        val appointmentsSnapshot = firestore.collection("appointments")
            .whereEqualTo("hospitalId", hospitalId)
            .whereGreaterThanOrEqualTo("dateTime", startOfDay)
            .whereLessThanOrEqualTo("dateTime", endOfDay)
            .get().await()

        val bookedHours = appointmentsSnapshot.documents.mapNotNull {
            it.toObject(Appointment::class.java)?.dateTime?.let { bookedDate ->
                Calendar.getInstance().apply { time = bookedDate }.get(Calendar.HOUR_OF_DAY)
            }
        }

        val allSlots = mutableListOf<TimeSlot>()
        val START_HOUR = 7 // Cập nhật theo giờ làm việc thực tế
        val END_HOUR = 16

        for (hour in START_HOUR until END_HOUR) {
            val timeString = String.format("%02d:00", hour)
            val isBooked = bookedHours.contains(hour)
            allSlots.add(TimeSlot(time = timeString, isAvailable = !isBooked))
        }
        Result.success(allSlots)
    } catch (e: Exception) {
        Result.failure(e)
    }
}

override suspend fun bookAppointment(hospitalId: String, dateTime: Date, volume:
String): Result<Unit> {
    val currentUser = auth.currentUser ?: return Result.failure(Exception("User not
authenticated."))

    // ... (Giữ nguyên các đoạn code lấy thông tin bệnh viện ở trên) ...
    val hospitalResult = getHospitalDetails(hospitalId)
    if (hospitalResult.isFailure) {
        return Result.failure(hospitalResult.exceptionOrNull() ?: Exception("Unknown error
finding hospital."))
    }
    val hospital = hospitalResult.getOrNull() ?: return Result.failure(Exception("Hospital
data is null."))

    return try {
        val appointmentId = firestore.collection("appointments").document().id

        val appointment = Appointment(
            id = appointmentId,
            userId = currentUser.uid,
            hospitalId = hospitalId,
            hospitalName = hospital.name,
            hospitalAddress = hospital.address,
            dateTime = dateTime,

            // --- SỬA Ở ĐÂY: Đổi từ "CONFIRMED" thành "PENDING" ---
            status = "PENDING",
            // "PENDING" nghĩa là Chờ duyệt. Admin trên web sẽ bấm duyệt để chuyển thành
"CONFIRMED"
            registeredVolume = volume,
        )

firestore.collection("appointments").document(appointmentId).set(appointment).await()
        Result.success(Unit)
    } catch (e: Exception) {
        Result.failure(e)
    }
}

override fun getMyAppointments(): Flow<Result<List<Appointment>>> = callbackFlow {
    val userId = auth.currentUser?.uid
    if (userId == null) {
        trySend(Result.failure(Exception("Người dùng chưa đăng nhập.")))
        close()
        return@callbackFlow
    }

    val query = firestore.collection("appointments")
        .whereEqualTo("userId", userId)
        .orderBy("dateTime", com.google.firebase.firestore.Query.Direction.DESCENDING)

    val listener = query.addSnapshotListener { snapshot, error ->
        if (error != null) {
            trySend(Result.failure(error))
            return@addSnapshotListener
        }

        if (snapshot != null) {
            // --- SỬA LỖI CRASH: Map dữ liệu thủ công thay vì dùng .toObjects() ---
            val appointments = snapshot.documents.map { doc ->

                // 1. Xử lý an toàn cho LabResult
                val labResultMap = doc.get("labResult") as? Map<String, Any>
                val labResult = if (labResultMap != null) {
                    com.smartblood.core.domain.model.LabResult(
                        documentUrl = labResultMap["documentUrl"] as? String,
                        conclusion = labResultMap["conclusion"] as? String,
                        // Xử lý recordedAt: Chấp nhận Timestamp hoặc Date, bỏ qua nếu là
HashMap lỗi
                        recordedAt = when (val rawDate = labResultMap["recordedAt"]) {
                            is com.google.firebase.Timestamp -> rawDate.toDate()
                is java.util.Date -> rawDate
                else -> null // Bỏ qua nếu dữ liệu sai định dạng (HashMap)
              }
            )
          } else {
            null
          }

          // 2. Map thủ công các trường của Appointment
          Appointment(
            id = doc.id,
            userId = doc.getString("userId") ?: "",
            hospitalId = doc.getString("hospitalId") ?: "",
            hospitalName = doc.getString("hospitalName") ?: "",
            hospitalAddress = doc.getString("hospitalAddress") ?: "",
            dateTime = doc.getDate("dateTime") ?: java.util.Date(),
            status = doc.getString("status") ?: "PENDING",

            // Map thêm các trường mới
            actualVolume = doc.getString("actualVolume"),
            registeredVolume = doc.getString("registeredVolume") ?: "350ml",
            labResult = labResult
          )
        }
        trySend(Result.success(appointments))
      }
    }
    awaitClose { listener.remove() }
  }
}
```

feature_map_booking/src/main/java/com/example/feature_map_booking/domain/di/MapBookingModule.kt

```
package com.example.feature_map_booking.domain.di
//
feature_map_booking/src/main/java/com/smartblood/mapbooking/di/MapBookingModule.kt

import
com.example.feature_map_booking.domain.data.repository.MapBookingRepositoryImpl
import com.example.feature_map_booking.domain.repository.MapBookingRepository
import dagger.Binds
import dagger.Module
import dagger.hilt.InstallIn
import dagger.hilt.components.SingletonComponent
import javax.inject.Singleton

@Module
@InstallIn(SingletonComponent::class)
abstract class MapBookingModule {

   @Binds
   @Singleton
   abstract fun bindMapBookingRepository(
      mapBookingRepositoryImpl: MapBookingRepositoryImpl
   ): MapBookingRepository
}
```

feature_map_booking/src/main/java/com/example/feature_map_booking/domain/repository/MapBookingRepository.kt

```
package com.example.feature_map_booking.domain.repository

//
feature_map_booking/src/main/java/com/smartblood/mapbooking/domain/repository/
MapBookingRepository.kt

import com.smartblood.core.domain.model.Appointment
import com.smartblood.core.domain.model.Hospital
import com.smartblood.core.domain.model.TimeSlot
import kotlinx.coroutines.flow.Flow
import java.util.Date
import kotlin.Result

interface MapBookingRepository {
   suspend fun getNearbyHospitals(lat: Double, lng: Double, radiusKm: Double):
Result<List<Hospital>>
   suspend fun getHospitalDetails(hospitalId: String): Result<Hospital>
   suspend fun getAvailableSlots(hospitalId: String, date: Date): Result<List<TimeSlot>>
   suspend fun bookAppointment(hospitalId: String, dateTime: Date, volume: String):
Result<Unit>


   fun getMyAppointments(): Flow<Result<List<Appointment>>>

}
```

feature_map_booking/src/main/java/com/example/feature_map_booking/domain/ui/booking/BookingContract.kt

```
package com.example.feature_map_booking.domain.ui.booking

//
feature_map_booking/src/main/java/com/smartblood/mapbooking/ui/booking/BookingContract.kt

import com.smartblood.core.domain.model.TimeSlot
import java.util.Date

data class BookingState(
   val hospitalId: String = "",
   val hospitalName: String = "",
   val isLoadingSlots: Boolean = false,
   val isBooking: Boolean = false,
   val selectedDate: Date = Date(),
   val selectedVolume: String = "350ml",
   val timeSlots: List<TimeSlot> = emptyList(),
   val error: String? = null,
   val bookingSuccess: Boolean = false
)

sealed class BookingEvent {
   data class OnDateSelected(val date: Date) : BookingEvent()
   data class OnVolumeSelected(val volume: String) : BookingEvent()

   data class OnSlotSelected(val time: String) : BookingEvent()
   object OnConfirmBooking : BookingEvent()
}
```

feature_map_booking/src/main/java/com/example/feature_map_booking/domain/ui/booking/BookingScreen.kt

```
package com.example.feature_map_booking.domain.ui.booking

import android.app.DatePickerDialog
import android.widget.DatePicker
import androidx.compose.foundation.BorderStroke
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.lazy.grid.GridCells
import androidx.compose.foundation.lazy.grid.LazyVerticalGrid
import androidx.compose.foundation.lazy.grid.items
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.automirrored.filled.ArrowBack
import androidx.compose.material3.*
import androidx.compose.runtime.*
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.hilt.navigation.compose.hiltViewModel
import com.smartblood.core.domain.model.TimeSlot
import java.text.SimpleDateFormat
import java.util.Calendar
import java.util.Locale

@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun BookingScreen(
    viewModel: BookingViewModel = hiltViewModel(),
    onBookingSuccess: () -> Unit,
    onNavigateBack: () -> Unit
) {
    val state by viewModel.state.collectAsState()
    val snackbarHostState = remember { SnackbarHostState() }
    val context = LocalContext.current
    val calendar = Calendar.getInstance()

    // Date Picker Dialog
    val datePickerDialog = remember {
        DatePickerDialog(
            context,
            { _: DatePicker, year: Int, month: Int, dayOfMonth: Int ->
                calendar.set(year, month, dayOfMonth)
                viewModel.onEvent(BookingEvent.OnDateSelected(calendar.time))
            },
            calendar.get(Calendar.YEAR),
            calendar.get(Calendar.MONTH),
            calendar.get(Calendar.DAY_OF_MONTH)
        ).apply {
            datePicker.minDate = System.currentTimeMillis()
        }
    }
```

```
// Effects
LaunchedEffect(state.bookingSuccess) {
  if (state.bookingSuccess) {
    onBookingSuccess()
  }
}

LaunchedEffect(state.error) {
  state.error?.let {
    snackbarHostState.showSnackbar(it)
    viewModel.clearError()
  }
}

// UI
Scaffold(
  snackbarHost = { SnackbarHost(snackbarHostState) },
  topBar = {
    TopAppBar(
      title = { Text("Đặt lịch hẹn") },
      navigationIcon = {
        IconButton(onClick = onNavigateBack) {
          Icon(
            imageVector = Icons.AutoMirrored.Filled.ArrowBack,
            contentDescription = "Back"
          )
        }
      }
    )
  }
) { paddingValues ->
  Column(
    modifier = Modifier
      .padding(paddingValues)
      .padding(16.dp)
      .fillMaxSize()
  ) {
    // Tên bệnh viện
    Text(
      text = state.hospitalName,
      style = MaterialTheme.typography.titleLarge,
      fontWeight = FontWeight.Bold
    )
```

```
        Spacer(modifier = Modifier.height(16.dp))

        // Chọn ngày
        val dateFormat = remember { SimpleDateFormat("dd/MM/yyyy",
Locale.getDefault()) }
        Text("Chọn ngày: ${dateFormat.format(state.selectedDate)}")
        Spacer(modifier = Modifier.height(8.dp))
        Button(onClick = { datePickerDialog.show() }) {
          Text("Đổi ngày")
        }

        Spacer(modifier = Modifier.height(24.dp))

        // Chọn khung giờ
        Text("Chọn khung giờ:", style = MaterialTheme.typography.titleMedium)
        Spacer(modifier = Modifier.height(8.dp))

        if (state.isLoadingSlots) {
          CircularProgressIndicator()
        } else {
          // Biến tạm để lưu giờ đang chọn trên UI (ViewModel đã quản lý nhưng dùng local
state để highlight nhanh)
          var selectedTime by remember { mutableStateOf<String?>(null) }

          LazyVerticalGrid(
            columns = GridCells.Adaptive(minSize = 100.dp),
            verticalArrangement = Arrangement.spacedBy(8.dp),
            horizontalArrangement = Arrangement.spacedBy(8.dp),
            // Giới hạn chiều cao cho Grid để không chiếm hết màn hình
            modifier = Modifier.height(200.dp)
          ) {
            items(state.timeSlots) { slot ->
              // Gọi hàm TimeSlotItem được định nghĩa ở dưới cùng file
              TimeSlotItem(
                timeSlot = slot,
                isSelected = slot.time == selectedTime,
                onSelect = { time ->
                  selectedTime = time
                  viewModel.onEvent(BookingEvent.OnSlotSelected(time))
                }
              )
            }
        }
    }

    Spacer(modifier = Modifier.height(24.dp))

    // --- CHỌN DUNG TÍCH (PHẦN MỚI) ---
    Text("Đăng ký lượng máu hiến:", style = MaterialTheme.typography.titleMedium,
fontWeight = FontWeight.Bold)
    Spacer(modifier = Modifier.height(8.dp))

    Row(horizontalArrangement = Arrangement.spacedBy(12.dp)) {
        val volumes = listOf("250ml", "350ml", "450ml")
        volumes.forEach { volume ->
            val isSelected = state.selectedVolume == volume
            OutlinedButton(
                onClick = { viewModel.onEvent(BookingEvent.OnVolumeSelected(volume)) },
                colors = if (isSelected)
                    ButtonDefaults.outlinedButtonColors(
                        containerColor = MaterialTheme.colorScheme.primary.copy(alpha = 0.1f),
                        contentColor = MaterialTheme.colorScheme.primary
                    )
                else
                    ButtonDefaults.outlinedButtonColors(),
                border = if (isSelected)
                    BorderStroke(2.dp, MaterialTheme.colorScheme.primary)
                else
                    BorderStroke(1.dp, Color.Gray)
            ) {
                Text(text = volume, fontWeight = if (isSelected) FontWeight.Bold else
FontWeight.Normal)
            }
        }
    }
    // --------------------------------

    Spacer(modifier = Modifier.weight(1f))

    // Nút Xác nhận
    Button(
        onClick = { viewModel.onEvent(BookingEvent.OnConfirmBooking) },
        enabled = !state.isBooking,
        modifier = Modifier
            .fillMaxWidth()
            .height(50.dp)
        ) {
            if (state.isBooking) {
                CircularProgressIndicator(modifier = Modifier.size(24.dp), color =
MaterialTheme.colorScheme.onPrimary)
            } else {
                Text("Xác nhận")
            }
        }
    }
  }
}

// --- ĐÂY LÀ HÀM BỊ THIẾU DẪN ĐẾN LỖI UNRESOLVED REFERENCE ---
@Composable
fun TimeSlotItem(
  timeSlot: TimeSlot,
  isSelected: Boolean,
  onSelect: (String) -> Unit
) {
  val colors = ButtonDefaults.outlinedButtonColors(
    containerColor = if (isSelected) MaterialTheme.colorScheme.primary else
MaterialTheme.colorScheme.surface,
    contentColor = if (isSelected) MaterialTheme.colorScheme.onPrimary else
MaterialTheme.colorScheme.primary
  )

  OutlinedButton(
    onClick = { onSelect(timeSlot.time) },
    enabled = timeSlot.isAvailable,
    colors = colors,
    border = BorderStroke(1.dp, MaterialTheme.colorScheme.primary)
  ) {
    Text(text = timeSlot.time)
  }
}
```

feature_map_booking/src/main/java/com/example/feature_map_booking/domain/ui/booking/BookingViewModel.kt

```
package com.example.feature_map_booking.domain.ui.booking

//
feature_map_booking/src/main/java/com/smartblood/mapbooking/ui/booking/BookingV
```

iewModel.kt

```
import androidx.lifecycle.SavedStateHandle
import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import com.example.feature_map_booking.domain.usecase.BookAppointmentUseCase
import com.example.feature_map_booking.domain.usecase.GetAvailableSlotsUseCase
import dagger.hilt.android.lifecycle.HiltViewModel
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.asStateFlow
import kotlinx.coroutines.flow.update
import kotlinx.coroutines.launch
import java.util.Calendar
import java.util.Date
import javax.inject.Inject

@HiltViewModel
class BookingViewModel @Inject constructor(
    private val getAvailableSlotsUseCase: GetAvailableSlotsUseCase,
    private val bookAppointmentUseCase: BookAppointmentUseCase,
    savedStateHandle: SavedStateHandle
) : ViewModel() {

    private val _state = MutableStateFlow(BookingState())
    val state = _state.asStateFlow()

    private val hospitalId: String = checkNotNull(savedStateHandle["hospitalId"])
    private var selectedTime: String? = null

    init {
        _state.update { it.copy(hospitalId = hospitalId, hospitalName =
savedStateHandle["hospitalName"] ?: "") }
        fetchSlotsForDate(Date())
    }
    fun clearError() {
        _state.update { it.copy(error = null) }
    }

    fun onEvent(event: BookingEvent) {
        when (event) {
            is BookingEvent.OnDateSelected -> {
                _state.update { it.copy(selectedDate = event.date) }
                fetchSlotsForDate(event.date)
            }
            is BookingEvent.OnSlotSelected -> {
                selectedTime = event.time
            }
            is BookingEvent.OnVolumeSelected -> {
                _state.update { it.copy(selectedVolume = event.volume) }
            }

            is BookingEvent.OnConfirmBooking -> {
                confirmBooking()
            }

        }
    }

    private fun fetchSlotsForDate(date: Date) {
        viewModelScope.launch {
            _state.update { it.copy(isLoadingSlots = true, error = null) } // Xóa lỗi cũ khi fetch lại
            getAvailableSlotsUseCase(hospitalId, date)
                .onSuccess { slots ->
                    _state.update { it.copy(isLoadingSlots = false, timeSlots = slots) }
                }
                .onFailure { error ->
                    _state.update { it.copy(isLoadingSlots = false, error = error.message) }
                }
        }
    }

    private fun confirmBooking() {
        // 1. Lấy giờ đã chọn. Nếu chưa chọn, báo lỗi và dừng lại.
        val timeString = selectedTime ?: run {
            _state.update { it.copy(error = "Vui lòng chọn một khung giờ.") }
            return
        }

        viewModelScope.launch {
            _state.update { it.copy(isBooking = true, error = null) }

            try {
                // 2. Tạo đối tượng Calendar và đặt ngày tháng năm từ state
                val calendar = Calendar.getInstance().apply {
                    time = _state.value.selectedDate
                    // 3. Phân tích chuỗi giờ (vd: "14:00") để lấy giờ và phút
                    val (hour, minute) = timeString.split(":").map { it.toInt() }

                    // 4. Cập nhật giờ, phút, giây cho Calendar
                    set(Calendar.HOUR_OF_DAY, hour)
                    set(Calendar.MINUTE, minute)
                    set(Calendar.SECOND, 0)
                    set(Calendar.MILLISECOND, 0)
                }

                // 5. Lấy ra đối tượng Date cuối cùng đã được kết hợp
                val finalDateTime = calendar.time
                val volumeToBook = _state.value.selectedVolume

                // 6. Gọi UseCase để đặt lịch
                bookAppointmentUseCase(hospitalId, finalDateTime, volumeToBook)
                    .onSuccess {
                        _state.update { it.copy(isBooking = false, bookingSuccess = true) }
                    }
                    .onFailure { error ->
                        _state.update { it.copy(isBooking = false, error = error.message) }
                    }
            } catch (e: NumberFormatException) {
                // 7. Bắt lỗi nếu định dạng giờ không hợp lệ
                _state.update { it.copy(isBooking = false, error = "Định dạng giờ không hợp lệ.") }
            }
        }
    }
  }
```

feature_map_booking/src/main/java/com/example/feature_map_booking/domain/ui/location_detail/LocationDetailContract.kt

```
package com.example.feature_map_booking.domain.ui.location_detail

// feature_map_booking/src/main/java/com/smartblood/mapbooking/ui/location_detail/LocationDetailContract.kt

import com.smartblood.core.domain.model.Hospital

data class LocationDetailState(
  val isLoading: Boolean = true,
  val hospital: Hospital? = null,
    val error: String? = null
)
```

feature_map_booking/src/main/java/com/example/feature_map_booking/domain/ui/location_detail/LocationDetailScreen.kt

```
package com.example.feature_map_booking.domain.ui.location_detail


//
feature_map_booking/src/main/java/com/smartblood/mapbooking/ui/location_detail/LocationDetailScreen.kt


import androidx.compose.foundation.BorderStroke
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.rememberScrollState
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.foundation.verticalScroll
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.automirrored.filled.ArrowBack
import androidx.compose.material.icons.filled.Phone
import androidx.compose.material.icons.filled.Schedule
import androidx.compose.material.icons.filled.Business
import androidx.compose.material.icons.filled.Bloodtype
import androidx.compose.material.icons.filled.Warning
import androidx.compose.material3.*
import androidx.compose.runtime.Composable
import androidx.compose.runtime.collectAsState
import androidx.compose.runtime.getValue
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.graphics.vector.ImageVector
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.hilt.navigation.compose.hiltViewModel
import com.smartblood.core.domain.model.Hospital

@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun LocationDetailScreen(
    viewModel: LocationDetailViewModel = hiltViewModel(),
    onNavigateToBooking: (hospitalId: String, hospitalName: String) -> Unit,
    onNavigateBack: () -> Unit
) {
    val state by viewModel.state.collectAsState()

    Scaffold(
        topBar = {
            TopAppBar(
                title = { Text("Chi tiết địa điểm") },
                navigationIcon = {
                    IconButton(onClick = onNavigateBack) {
                        Icon(
                            imageVector = Icons.AutoMirrored.Filled.ArrowBack,
                            contentDescription = "Back"
                        )
                    }
                }
            )
        }
    ) { paddingValues ->
        Box(
            modifier = Modifier
                .fillMaxSize()
                .padding(paddingValues)
        ) {
            if (state.isLoading) {
                CircularProgressIndicator(modifier = Modifier.align(Alignment.Center))
            } else if (state.hospital != null) {
                HospitalDetails(
                    hospital = state.hospital!!,
                    onBookAppointment = {
                        onNavigateToBooking(state.hospital!!.id, state.hospital!!.name)
                    }
                )
            } else {
                Text(
                    text = state.error ?: "Không thể tải thông tin bệnh viện.",
                    modifier = Modifier.align(Alignment.Center)
                )
            }
        }
    }
}
```

```
@Composable
fun HospitalDetails(hospital: Hospital, onBookAppointment: () -> Unit) {
   Column(
      modifier = Modifier
         .fillMaxSize()
         .padding(16.dp)
         .verticalScroll(rememberScrollState()), // Thêm scroll vì nội dung có thể dài
      verticalArrangement = Arrangement.spacedBy(16.dp)
   ) {
      Text(
         text = hospital.name,
         style = MaterialTheme.typography.headlineMedium,
         fontWeight = FontWeight.Bold
      )

      Divider(color = Color.LightGray.copy(alpha = 0.5f))

      // --- THÊM MỚI: Section Kho Máu ---
      InventorySection(inventory = hospital.inventory)

      Divider(color = Color.LightGray.copy(alpha = 0.5f))

      InfoRow(icon = Icons.Filled.Business, text = hospital.address)
      InfoRow(icon = Icons.Filled.Phone, text = hospital.phone)
      InfoRow(icon = Icons.Filled.Schedule, text = hospital.workingHours)

      Spacer(modifier = Modifier.weight(1f))

      Button(
         onClick = onBookAppointment,
         modifier = Modifier
            .fillMaxWidth()
            .height(50.dp),
         colors = ButtonDefaults.buttonColors(containerColor =
MaterialTheme.colorScheme.primary)
      ) {
         Text(text = "Đặt lịch hiến máu", fontSize = 16.sp, fontWeight = FontWeight.Bold)
      }
   }
}

@OptIn(ExperimentalLayoutApi::class) // Cần OptIn cho FlowRow
```

```
@Composable
fun InventorySection(inventory: Map<String, Int>) {
  Column(modifier = Modifier.fillMaxWidth()) {
    Row(verticalAlignment = Alignment.CenterVertically) {
      Icon(
        imageVector = Icons.Default.Bloodtype,
        contentDescription = null,
        tint = MaterialTheme.colorScheme.primary
      )
      Spacer(modifier = Modifier.width(8.dp))
      Text(
        text = "Tình trạng kho máu",
        style = MaterialTheme.typography.titleMedium,
        fontWeight = FontWeight.Bold
      )
    }

    Spacer(modifier = Modifier.height(12.dp))

    if (inventory.isEmpty()) {
      Text("Chưa có dữ liệu tồn kho.", style = MaterialTheme.typography.bodyMedium,
color = Color.Gray)
    } else {
      // Sử dụng FlowRow để các item tự xuống dòng
      FlowRow(
        modifier = Modifier.fillMaxWidth(),
        horizontalArrangement = Arrangement.spacedBy(8.dp),
        verticalArrangement = Arrangement.spacedBy(8.dp)
      ) {
        // Sắp xếp để nhóm máu thiếu (critical) lên đầu
        val sortedInventory = inventory.entries.sortedBy { it.value }

        sortedInventory.forEach { (type, count) ->
          val isCritical = count < 5 // Ngưỡng báo động đỏ

          val backgroundColor = if (isCritical) Color(0xFFFFEBEE) else Color(0xFFE8F5E9)
          val borderColor = if (isCritical) Color(0xFFEF5350) else Color(0xFF66BB6A)
          val textColor = if (isCritical) Color(0xFFC62828) else Color(0xFF2E7D32)
          val statusText = if (isCritical) "Thiếu ($count)" else "Ổn ($count)"

          Surface(
            color = backgroundColor,
            shape = RoundedCornerShape(8.dp),
                border = BorderStroke(1.dp, borderColor)
            ) {
                Row(
                    modifier = Modifier.padding(horizontal = 12.dp, vertical = 8.dp),
                    verticalAlignment = Alignment.CenterVertically
                ) {
                    Text(
                        text = type,
                        style = MaterialTheme.typography.bodyLarge,
                        fontWeight = FontWeight.Bold,
                        color = textColor
                    )
                    Spacer(modifier = Modifier.width(6.dp))
                    Text(
                        text = statusText,
                        style = MaterialTheme.typography.bodySmall,
                        color = textColor.copy(alpha = 0.8f)
                    )
                    if (isCritical) {
                        Spacer(modifier = Modifier.width(4.dp))
                        Icon(
                            imageVector = Icons.Default.Warning,
                            contentDescription = "Critical",
                            tint = textColor,
                            modifier = Modifier.size(14.dp)
                        )
                    }
                }
            }
        }
    }
  }
 }
}


@Composable
fun InfoRow(icon: ImageVector, text: String) {
    Row(
        verticalAlignment = Alignment.CenterVertically,
        horizontalArrangement = Arrangement.spacedBy(16.dp)
    ) {
        Icon(imageVector = icon, contentDescription = null, tint =
MaterialTheme.colorScheme.primary)
        Text(text = text, style = MaterialTheme.typography.bodyLarge)
    }
}
```

feature_map_booking/src/main/java/com/example/feature_map_booking/domain/ui/location_detail/LocationDetailViewModel.kt

```
package com.example.feature_map_booking.domain.ui.location_detail

//
feature_map_booking/src/main/java/com/smartblood/mapbooking/ui/location_detail/LocationDetailViewModel.kt

import androidx.lifecycle.SavedStateHandle
import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import com.example.feature_map_booking.domain.usecase.GetHospitalDetailsUseCase
import dagger.hilt.android.lifecycle.HiltViewModel
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.asStateFlow
import kotlinx.coroutines.flow.update
import kotlinx.coroutines.launch
import javax.inject.Inject

@HiltViewModel
class LocationDetailViewModel @Inject constructor(
    private val getHospitalDetailsUseCase: GetHospitalDetailsUseCase,
    savedStateHandle: SavedStateHandle
) : ViewModel() {

    private val _state = MutableStateFlow(LocationDetailState())
    val state = _state.asStateFlow()

    // Lấy ID được truyền từ màn hình Map sang
    private val hospitalId: String = checkNotNull(savedStateHandle["hospitalId"])

    init {
        fetchHospitalDetails()
    }

    private fun fetchHospitalDetails() {
        viewModelScope.launch {
            _state.update { it.copy(isLoading = true, error = null) }
        // --- SỬA ĐỔI QUAN TRỌNG: Gọi UseCase lấy dữ liệu thật ---
        getHospitalDetailsUseCase(hospitalId)
          .onSuccess { hospital ->
            _state.update {
              it.copy(
                isLoading = false,
                hospital = hospital,
                error = null
              )
            }
          }
          .onFailure { error ->
            _state.update {
              it.copy(
                isLoading = false,
                error = "Lỗi: Không tìm thấy bệnh viện với ID: $hospitalId"
              )
            }
          }
    }
  }
}
```

feature_map_booking/src/main/java/com/example/feature_map_booking/domain/ui/map/MapContract.kt

```
package com.example.feature_map_booking.domain.ui.map

//
feature_map_booking/src/main/java/com/smartblood/mapbooking/ui/map/MapContract
.kt

import com.trackasia.android.geometry.LatLng
import com.smartblood.core.domain.model.Hospital

data class MapState(
  val isLoading: Boolean = true,
  // SỬA KIỂU DỮ LIỆU Ở ĐÂY
  val lastKnownLocation: LatLng? = null,
  val hospitals: List<Hospital> = emptyList(),
  val error: String? = null
)
```

```
sealed class MapEvent {
    object OnMapLoaded : MapEvent()
    data class OnMapReady(val location: LatLng) : MapEvent()
    data class OnMarkerClick(val hospitalId: String) : MapEvent()
}
```

feature_map_booking/src/main/java/com/example/feature_map_booking/domain/ui/map/MapScreen.kt

```
//
feature_map_booking/src/main/java/com/example/feature_map_booking/domain/ui/map
/MapScreen.kt

package com.example.feature_map_booking.domain.ui.map

import android.util.Log
import androidx.compose.foundation.layout.Box
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.material3.CircularProgressIndicator
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.hilt.navigation.compose.hiltViewModel

@Composable
fun MapScreen(
    viewModel: MapViewModel = hiltViewModel(),
    onNavigateToLocationDetail: (String) -> Unit
) {
    val state by viewModel.state.collectAsState()
    Log.d("MapDebug", "MapScreen recomposed. Hospitals count: ${state.hospitals.size}")

    Box(Modifier.fillMaxSize()) {
        // Gọi Composable bản đồ và truyền dữ liệu vào
        TrackAsiaMapComposable(
            hospitals = state.hospitals,
            onMarkerClick = onNavigateToLocationDetail,
//          onMapReady = { viewModel.onEvent(MapEvent.OnMapLoaded) }
        )

        // Vẫn hiển thị loading indicator từ ViewModel
        if (state.isLoading) {
            CircularProgressIndicator(modifier = Modifier.align(Alignment.Center))
        }
    }

    // Trigger ViewModel tải dữ liệu bệnh viện (dữ liệu giả)
    LaunchedEffect(Unit) {
        viewModel.onEvent(MapEvent.OnMapLoaded)
    }
}
```

feature_map_booking/src/main/java/com/example/feature_map_booking/domain/ui/map/MapViewModel.kt

```
package com.example.feature_map_booking.domain.ui.map

//
feature_map_booking/src/main/java/com/smartblood/mapbooking/ui/map/MapViewModel.kt

import android.util.Log
import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import com.example.feature_map_booking.domain.usecase.GetNearbyHospitalsUseCase
import dagger.hilt.android.lifecycle.HiltViewModel
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.asStateFlow
import kotlinx.coroutines.flow.update
import kotlinx.coroutines.launch
import javax.inject.Inject

@HiltViewModel
class MapViewModel @Inject constructor(
    // Inject UseCase thay vì Repository trực tiếp (Clean Architecture)
    private val getNearbyHospitalsUseCase: GetNearbyHospitalsUseCase
) : ViewModel() {

    private val _state = MutableStateFlow(MapState())
    val state = _state.asStateFlow()

    fun onEvent(event: MapEvent) {
        when (event) {
            MapEvent.OnMapLoaded -> {
                fetchHospitals()
            }
            // Các event khác giữ nguyên hoặc TODO
            is MapEvent.OnMapReady -> {}
            is MapEvent.OnMarkerClick -> {}
        }
    }

    private fun fetchHospitals() {
        viewModelScope.launch {
            _state.update { it.copy(isLoading = true) }

            // Gọi UseCase để lấy dữ liệu thật
            // Lat/Lng 0.0, 0.0 là demo, thực tế sẽ lấy từ location của user
            val result = getNearbyHospitalsUseCase(10.7, 106.6, 10.0)

            result.onSuccess { hospitals ->
                Log.d("MapViewModel", "Lấy thành công: ${hospitals.size} bệnh viện")
                _state.update {
                    it.copy(
                        isLoading = false,
                        hospitals = hospitals,
                        error = null
                    )
                }
            }.onFailure { error ->
                Log.e("MapViewModel", "Lỗi lấy bệnh viện: ${error.message}")
                _state.update {
                    it.copy(
                        isLoading = false,
                        error = error.message
                    )
                }
            }
        }
    }
}
```

feature_map_booking/src/main/java/com/example/feature_map_booking/domain/ui/map/TrackAsiaMapComposable.kt

```
//
feature_map_booking/src/main/java/com/example/feature_map_booking/domain/ui/map
/TrackAsiaMapComposable.kt

package com.example.feature_map_booking.domain.ui.map

import com.smartblood.core.R
import android.Manifest
import android.annotation.SuppressLint
import android.graphics.BitmapFactory
import android.location.Location
import android.util.Log
import androidx.activity.compose.rememberLauncherForActivityResult
import androidx.activity.result.contract.ActivityResultContracts
import androidx.compose.foundation.layout.Box
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.shape.CircleShape
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.filled.MyLocation
import androidx.compose.material3.FloatingActionButton
import androidx.compose.material3.Icon
import androidx.compose.material3.MaterialTheme
import androidx.compose.runtime.Composable
import androidx.compose.runtime.DisposableEffect
import androidx.compose.runtime.LaunchedEffect
import androidx.compose.runtime.getValue
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.runtime.setValue
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.platform.LocalLifecycleOwner
import androidx.compose.ui.unit.dp
import androidx.compose.ui.viewinterop.AndroidView
import androidx.lifecycle.Lifecycle
import androidx.lifecycle.LifecycleEventObserver
import com.smartblood.core.domain.model.Hospital
import com.google.android.gms.location.LocationServices
import com.google.gson.JsonParser
import com.trackasia.android.camera.CameraUpdateFactory
import com.trackasia.android.geometry.LatLng
import com.trackasia.android.location.LocationComponent
import com.trackasia.android.location.LocationComponentActivationOptions
import com.trackasia.android.location.modes.CameraMode
import com.trackasia.android.location.modes.RenderMode
import com.trackasia.android.maps.MapView
import com.trackasia.android.maps.Style
import com.trackasia.android.plugins.annotation.SymbolManager
import com.trackasia.android.plugins.annotation.SymbolOptions

@SuppressLint("MissingPermission")
@Composable
fun TrackAsiaMapComposable(
    hospitals: List<Hospital>,
    onMarkerClick: (String) -> Unit
) {
    val context = LocalContext.current
    val mapView = rememberMapViewWithLifecycle()

    var trackasiaMap by remember {
mutableStateOf<com.trackasia.android.maps.TrackAsiaMap?>(null) }
    var symbolManager by remember { mutableStateOf<SymbolManager?>(null) }
    var locationComponent by remember { mutableStateOf<LocationComponent?>(null) }
    var hasLocationPermission by remember { mutableStateOf(false) }

    val locationPermissionLauncher = rememberLauncherForActivityResult(
        contract = ActivityResultContracts.RequestPermission(),
        onResult = { isGranted ->
            hasLocationPermission = isGranted
        }
    )

    LaunchedEffect(Unit) {
        locationPermissionLauncher.launch(Manifest.permission.ACCESS_FINE_LOCATION)
    }

    Box(Modifier.fillMaxSize()) {
        AndroidView(
            factory = {
                mapView.apply {
                    onCreate(null)
                    getMapAsync { map ->
                        trackasiaMap = map
                        // LƯU Ý: Thay "YOUR_API_KEY" bằng API key của bạn
                        val styleUrl = "https://maps.track-
asia.com/styles/v1/streets.json?key=52fedb6b306931761836057e5580a05be7"
                        map.setStyle(Style.Builder().fromUri(styleUrl))
                    }
                }
            },
            modifier = Modifier.fillMaxSize()
        )

        // EFFECT 1: Xử lý vị trí ban đầu và zoom
        LaunchedEffect(trackasiaMap, hasLocationPermission) {
            val map = trackasiaMap ?: return@LaunchedEffect
            if (!hasLocationPermission) return@LaunchedEffect

            map.getStyle { style ->
                if (!style.isFullyLoaded) return@getStyle

                val component = map.locationComponent
                locationComponent = component
                if (!component.isLocationComponentActivated) {
                    component.activateLocationComponent(
                        LocationComponentActivationOptions.builder(context, style).build()
                    )
                }
                component.isLocationComponentEnabled = true
                component.renderMode = RenderMode.COMPASS
                component.cameraMode = CameraMode.NONE

                val fusedLocationClient =
LocationServices.getFusedLocationProviderClient(context)
                fusedLocationClient.lastLocation.addOnSuccessListener { location: Location? ->
                    location?.let {
                        map.animateCamera(
                            CameraUpdateFactory.newLatLngZoom(LatLng(it.latitude, it.longitude),
14.0),
                            2000
                        )
                    }
                }.addOnFailureListener {
                    Log.d("MapDebug", "Could not get last known location.")
                }
            }
        }

        // EFFECT 2: Xử lý vẽ marker
        LaunchedEffect(trackasiaMap, hospitals) {
            val map = trackasiaMap ?: return@LaunchedEffect

            val CUSTOM_MARKER_ID = "my-marker-icon"
            val bitmap = BitmapFactory.decodeResource(context.resources,
R.drawable.ic_map_marker)

      map.getStyle { style ->
            if (!style.isFullyLoaded) return@getStyle

            style.addImage(CUSTOM_MARKER_ID, bitmap)

            if (symbolManager == null) {
              symbolManager = SymbolManager(mapView, map, style).apply {
                addClickListener { symbol ->
                  symbol.data?.asJsonObject?.get("hospital_id")?.asString?.let(onMarkerClick)
                  true
                }
              }
              symbolManager?.iconAllowOverlap = true
              symbolManager?.iconIgnorePlacement = true

            }

            symbolManager?.deleteAll()

            if (hospitals.isNotEmpty()) {
              val options: List<SymbolOptions> = hospitals.mapNotNull { hospital ->
                hospital.location?.let { geoPoint ->
                  SymbolOptions()
                    .withLatLng(LatLng(geoPoint.latitude, geoPoint.longitude))
                    .withIconImage(CUSTOM_MARKER_ID)
                    .withIconSize(0.1f)
                    .withIconAnchor("bottom")
                    .withData(JsonParser.parseString("""{"hospital_id": "${hospital.id}"}"""))
                }
              }
              // Dòng lệnh create bây giờ sẽ hoạt động chính xác
              symbolManager?.create(options)
            }
      }
        }

        // Nút "Vị trí của tôi"
        FloatingActionButton(
      onClick = {
            locationComponent?.lastKnownLocation?.let {
              trackasiaMap?.animateCamera(
                CameraUpdateFactory.newLatLngZoom(LatLng(it.latitude, it.longitude), 14.0),
                1000
              )
            }
          },
          modifier = Modifier
            .align(Alignment.BottomEnd)
            .padding(16.dp),
          shape = CircleShape,
          containerColor = MaterialTheme.colorScheme.surface,
          contentColor = MaterialTheme.colorScheme.primary
        ) {
          Icon(imageVector = Icons.Default.MyLocation, contentDescription = "Vị trí của tôi")
        }
    }
}

// Hàm hỗ trợ để quản lý vòng đời của MapView
@Composable
fun rememberMapViewWithLifecycle(): MapView {
    val context = LocalContext.current
    val mapView = remember { MapView(context) }

    val lifecycle = LocalLifecycleOwner.current.lifecycle
    DisposableEffect(lifecycle, mapView) {
        val lifecycleObserver = LifecycleEventObserver { _, event ->
            when (event) {
                Lifecycle.Event.ON_CREATE -> mapView.onCreate(null)
                Lifecycle.Event.ON_START -> mapView.onStart()
                Lifecycle.Event.ON_RESUME -> mapView.onResume()
                Lifecycle.Event.ON_PAUSE -> mapView.onPause()
                Lifecycle.Event.ON_STOP -> mapView.onStop()
                Lifecycle.Event.ON_DESTROY -> mapView.onDestroy()
                else -> {}
            }
        }
        lifecycle.addObserver(lifecycleObserver)
        onDispose {
            lifecycle.removeObserver(lifecycleObserver)
        }
    }
    return mapView
}
```

feature_map_booking/src/main/java/com/example/feature_map_booking/domain/usecase/BookAppointmentUseCase.kt

```
package com.example.feature_map_booking.domain.usecase
//
feature_map_booking/src/main/java/com/smartblood/mapbooking/domain/usecase/BookAppointmentUseCase.kt

import com.example.feature_map_booking.domain.repository.MapBookingRepository
import java.util.Date
import javax.inject.Inject
import kotlin.Result

class BookAppointmentUseCase @Inject constructor(
   private val repository: MapBookingRepository
) {
   suspend operator fun invoke(hospitalId: String, dateTime: Date, volume: String): Result<Unit> {
      if (hospitalId.isBlank()) {
         return Result.failure(IllegalArgumentException("Hospital ID is required."))
      }
      if (dateTime.before(Date())) {
         return Result.failure(IllegalArgumentException("Cannot book an appointment in the past."))
      }
      return repository.bookAppointment(hospitalId, dateTime, volume)
   }
}
```

feature_map_booking/src/main/java/com/example/feature_map_booking/domain/usecase/GetAvailableSlotsUseCase.kt

```
package com.example.feature_map_booking.domain.usecase

//
feature_map_booking/src/main/java/com/smartblood/mapbooking/domain/usecase/GetAvailableSlotsUseCase.kt

import com.smartblood.core.domain.model.TimeSlot
import com.example.feature_map_booking.domain.repository.MapBookingRepository
import java.util.Date
import javax.inject.Inject
import kotlin.Result

class GetAvailableSlotsUseCase @Inject constructor(
    private val repository: MapBookingRepository
) {
    suspend operator fun invoke(hospitalId: String, date: Date): Result<List<TimeSlot>> {
        return repository.getAvailableSlots(hospitalId, date)
    }
}
```

feature_map_booking/src/main/java/com/example/feature_map_booking/domain/usecase/GetHospitalDetailsUseCase.kt

```
package com.example.feature_map_booking.domain.usecase

//
feature_map_booking/src/main/java/com/smartblood/mapbooking/domain/usecase/GetHospitalDetailsUseCase.kt

import com.smartblood.core.domain.model.Hospital
import com.example.feature_map_booking.domain.repository.MapBookingRepository
import javax.inject.Inject
import kotlin.Result

class GetHospitalDetailsUseCase @Inject constructor(
    private val repository: MapBookingRepository
) {
    suspend operator fun invoke(hospitalId: String): Result<Hospital> {
        if (hospitalId.isBlank()) {
            return Result.failure(IllegalArgumentException("Hospital ID cannot be empty."))
        }
        return repository.getHospitalDetails(hospitalId)
    }
}
```

feature_map_booking/src/main/java/com/example/feature_map_booking/domain/usecase/GetMyAppointmentsUseCase.kt

```
package com.example.feature_map_booking.domain.usecase

import com.smartblood.core.domain.model.Appointment
import com.example.feature_map_booking.domain.repository.MapBookingRepository
import javax.inject.Inject
import kotlin.Result

class GetMyAppointmentsUseCase @Inject constructor(
    private val repository: MapBookingRepository
) {
    operator fun invoke() = repository.getMyAppointments()

}
```

feature_map_booking/src/main/java/com/example/feature_map_booking/domain/usecase/GetNearbyHospitalsUseCase.kt

```
package com.example.feature_map_booking.domain.usecase

import com.smartblood.core.domain.model.Hospital
import com.example.feature_map_booking.domain.repository.MapBookingRepository
import javax.inject.Inject
import kotlin.Result

class GetNearbyHospitalsUseCase @Inject constructor(
    private val repository: MapBookingRepository
) {
    suspend operator fun invoke(lat: Double, lng: Double, radiusKm: Double = 10.0):
Result<List<Hospital>> {
        return repository.getNearbyHospitals(lat, lng, radiusKm)
    }
}
```

feature_map_booking/src/test/java/com/example/feature_map_booking/ExampleUnitTest.kt

```
package com.example.feature_map_booking

import org.junit.Test

import org.junit.Assert.*

/**
 * Example local unit test, which will execute on the development machine (host).
 *
 * See [testing documentation](http://d.android.com/tools/testing).
 */
class ExampleUnitTest {
    @Test
    fun addition_isCorrect() {
        assertEquals(4, 2 + 2)
    }
}
```

feature_profile/.gitignore

```
/build
```

feature_profile/build.gradle.kts

```
plugins {
    alias(libs.plugins.android.library)
    alias(libs.plugins.kotlin.android)
    alias(libs.plugins.kotlin.compose.compiler)
    alias(libs.plugins.hilt)
    alias(libs.plugins.ksp)
}

android {
    namespace = "com.example.feature_profile"
    compileSdk = 34

    defaultConfig {
        minSdk = 24

        testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
        consumerProguardFiles("consumer-rules.pro")
    }

    buildTypes {
        release {
            isMinifyEnabled = false
            proguardFiles(
                getDefaultProguardFile("proguard-android-optimize.txt"),
                "proguard-rules.pro"
            )
        }
    }
    compileOptions {
        sourceCompatibility = JavaVersion.VERSION_11
        targetCompatibility = JavaVersion.VERSION_11
    }
    kotlinOptions {
        jvmTarget = "11"
    }
    buildFeatures {
        compose = true
    }
}
```

```
dependencies {
    implementation(project(":core"))
    implementation(platform(libs.androidx.compose.bom))

    implementation(libs.androidx.core.ktx)

    // THÊM VÀO: Jetpack Compose cho UI
    implementation(platform(libs.androidx.compose.bom)) // Bill of Materials
    implementation(libs.androidx.compose.ui)
    implementation(libs.androidx.compose.material3)
    implementation(libs.androidx.compose.ui.tooling.preview)

    // --- QUAN TRỌNG: Thêm thư viện này để sửa lỗi thiếu icon History, PhotoCamera ---
    implementation("androidx.compose.material:material-icons-extended:1.6.1")
    // ---------------------------------------------------------------------------

    implementation(project(":feature_map_booking"))
    implementation(project(":feature_emergency"))
    debugImplementation(libs.androidx.compose.ui.tooling)

    // THÊM VÀO: Hilt - Dependency Injection
    implementation(libs.hilt.android)
    ksp(libs.hilt.compiler)
    implementation(libs.androidx.hilt.navigation.compose) // Để dùng hiltViewModel() trong
Composable

    // THÊM VÀO: Firebase
    implementation(libs.firebase.auth.ktx)
    implementation(libs.firebase.firestore.ktx)

    // THÊM VÀO: Lifecycle cho ViewModel và Coroutine Scope
    implementation(libs.androidx.lifecycle.runtime.ktx)
    implementation(libs.androidx.lifecycle.viewmodel.compose) // Cho ViewModel

    // THÊM VÀO: Coil để tải ảnh (dùng cho AsyncImage)
    implementation(libs.coil.compose)

    // Testing
    testImplementation(libs.junit)
    androidTestImplementation(libs.androidx.junit)
    androidTestImplementation(libs.androidx.espresso.core)
    // THÊM VÀO: Testing cho Compose
    androidTestImplementation(platform(libs.androidx.compose.bom))
    androidTestImplementation(libs.androidx.compose.ui.test.junit4)

    implementation(libs.trackasia.sdk)
    implementation(libs.trackasia.annotation.plugin)
}
```

feature_profile/consumer-rules.pro

[File rỗng]

feature_profile/proguard-rules.pro

```
# Add project specific ProGuard rules here.
# You can control the set of applied configuration files using the
# proguardFiles setting in build.gradle.
#
# For more details, see
#   http://developer.android.com/guide/developing/tools/proguard.html

# If your project uses WebView with JS, uncomment the following
# and specify the fully qualified class name to the JavaScript interface
# class:
#-keepclassmembers class fqcn.of.javascript.interface.for.webview {
#   public *;
#}

# Uncomment this to preserve the line number information for
# debugging stack traces.
#-keepattributes SourceFile,LineNumberTable

# If you keep the line number information, uncomment this to
# hide the original source file name.
#-renamesourcefileattribute SourceFile
```

feature_profile/src/androidTest/java/com/example/feature_profile/ExampleInstrumentedTest.kt

```
package com.example.feature_profile

import androidx.test.platform.app.InstrumentationRegistry
import androidx.test.ext.junit.runners.AndroidJUnit4

import org.junit.Test
import org.junit.runner.RunWith
import org.junit.Assert.*

/**
 * Instrumented test, which will execute on an Android device.
 *
 * See [testing documentation](http://d.android.com/tools/testing).
 */
@RunWith(AndroidJUnit4::class)
class ExampleInstrumentedTest {
    @Test
    fun useAppContext() {
        // Context of the app under test.
        val appContext = InstrumentationRegistry.getInstrumentation().targetContext
        assertEquals("com.example.feature_profile.test", appContext.packageName)
    }
}
```

feature_profile/src/main/AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android">

</manifest>
```

feature_profile/src/main/java/com/example/feature_profile/data/repository/ProfileRepositoryImpl.kt

```
package com.example.feature_profile.data.repository

import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.firestore.FirebaseFirestore
import com.smartblood.profile.domain.model.DonationRecord
import com.smartblood.core.domain.model.UserProfile
import com.example.feature_profile.domain.repository.ProfileRepository
import kotlinx.coroutines.tasks.await
import java.util.Date
import javax.inject.Inject
import kotlin.Result

class ProfileRepositoryImpl @Inject constructor(
    private val firestore: FirebaseFirestore,
    private val auth: FirebaseAuth
) : ProfileRepository {

    override suspend fun getUserProfile(): Result<UserProfile> {
        return try {
            val userId = auth.currentUser?.uid ?: return Result.failure(Exception("User not logged
in"))
            val document = firestore.collection("users").document(userId).get().await()
            val userProfile = document.toObject(UserProfile::class.java)
                ?: return Result.failure(Exception("User profile not found"))
            Result.success(userProfile)
        } catch (e: Exception) {
            Result.failure(e)
        }
    }

    override suspend fun updateUserProfile(userProfile: UserProfile): Result<Unit> {
        return try {
            val userId = auth.currentUser?.uid ?: return Result.failure(Exception("User not logged
in"))
            firestore.collection("users").document(userId).set(userProfile).await()
            Result.success(Unit)
        } catch (e: Exception) {
            Result.failure(e)
        }
    }

    override suspend fun getDonationHistory(): Result<List<DonationRecord>> {
        return try {
            val userId = auth.currentUser?.uid
                ?: return Result.failure(Exception("User not logged in"))

            // Query vào collection chung "appointments"
            val querySnapshot = firestore.collection("appointments")
                .whereEqualTo("userId", userId)
                // .whereEqualTo("status", "COMPLETED") // Tạm bỏ để test xem có hiện lịch sử
không
                .orderBy("dateTime", com.google.firebase.firestore.Query.Direction.DESCENDING)
                .get()
                .await()

            val history = querySnapshot.documents.map { doc ->
                // --- XỬ LÝ AN TOÀN CHO LAB RESULT ---
                // Lấy field labResult dạng Map để tránh lỗi Deserialization
                val labResultMap = doc.get("labResult") as? Map<String, Any>
                val labResult = if (labResultMap != null) {
                    com.smartblood.core.domain.model.LabResult(
                        documentUrl = labResultMap["documentUrl"] as? String,
                        conclusion = labResultMap["conclusion"] as? String,
                        // Xử lý an toàn: Nếu là Timestamp thì convert, nếu null thì thôi
                        recordedAt = (labResultMap["recordedAt"] as?
com.google.firebase.Timestamp)?.toDate()
                    )
                } else {
                    null
                }

                // Map dữ liệu thủ công
                DonationRecord(
                    id = doc.id,
                    hospitalName = doc.getString("hospitalName") ?: "Bệnh viện",
                    hospitalAddress = doc.getString("hospitalAddress") ?: "",
                    // Xử lý ngày tháng an toàn
                    date = doc.getDate("dateTime") ?: Date(),
                    status = doc.getString("status") ?: "COMPLETED",
                    certificateUrl = doc.getString("certificateUrl"),
                    actualVolume = doc.getString("actualVolume"), // Lấy dung tích thực tế
                    labResult = labResult // Gán kết quả xét nghiệm
                )
            }
            Result.success(history)
        } catch (e: Exception) {
            e.printStackTrace()
            // Log lỗi ra để debug nếu cần
            Result.failure(e)
        }
    }
    override fun signOut() {
        auth.signOut()
    }
}
```

feature_profile/src/main/java/com/example/feature_profile/di/ProfileModule.kt

```
package com.example.feature_profile.di

import com.example.feature_profile.data.repository.ProfileRepositoryImpl
import com.example.feature_profile.domain.repository.ProfileRepository
import dagger.Binds
import dagger.Module
import dagger.hilt.InstallIn
import dagger.hilt.components.SingletonComponent
import javax.inject.Singleton

@Module
@InstallIn(SingletonComponent::class)
abstract class ProfileModule {

    @Binds
    @Singleton
    abstract fun bindProfileRepository(
        profileRepositoryImpl: ProfileRepositoryImpl
    ): ProfileRepository
}
```

feature_profile/src/main/java/com/example/feature_profile/domain/model/DonationRecord.kt

```
//D:\SmartBloodDonationAndroid\feature_profile\src\main\java\com\example\feature_profile\domain\model\DonationRecord.kt
package com.smartblood.profile.domain.model

import com.smartblood.core.domain.model.LabResult
import java.util.Date

data class DonationRecord(
    val id: String = "",
    val hospitalName: String = "",
    val hospitalAddress: String = "",
    val date: Date = Date(),
    val status: String = "",
    val certificateUrl: String? = null, // Link chứng nhận

    // --- CÁC TRƯỜNG MỚI ---
    val actualVolume: String? = null, // Hiển thị lượng máu đã hiến
    val labResult: LabResult? = null  // Chứa link PDF kết quả và lời dặn
)
```

feature_profile/src/main/java/com/example/feature_profile/domain/model/UserProfile.kt

```
package com.example.feature_profile.domain.model

import java.util.Date
```

```
data class UserProfile(
    val uid: String = "",
    val email: String = "",
    val fullName: String = "",
    val phoneNumber: String? = null,
    val bloodType: String? = null,
    val avatarUrl: String? = null,
    val dateOfBirth: Date? = null,
    val gender: String? = null,
    val lastDonationDate: Date? = null
)
```

feature_profile/src/main/java/com/example/feature_profile/domain/repository/ProfileRepository.kt

```
package com.example.feature_profile.domain.repository

import com.smartblood.profile.domain.model.DonationRecord
import com.smartblood.core.domain.model.UserProfile
import kotlin.Result

interface ProfileRepository {
    suspend fun getUserProfile(): Result<UserProfile>
    suspend fun updateUserProfile(userProfile: UserProfile): Result<Unit>
    suspend fun getDonationHistory(): Result<List<DonationRecord>>

    // Hàm đăng xuất
    fun signOut()
}
```

feature_profile/src/main/java/com/example/feature_profile/domain/usecase/CalculateNextDonationDateUseCase.kt

```
package com.smartblood.profile.domain.usecase

import com.smartblood.core.domain.model.UserProfile
import java.text.SimpleDateFormat
import java.util.Calendar
import java.util.Date
import java.util.Locale
import java.util.concurrent.TimeUnit
import javax.inject.Inject


class CalculateNextDonationDateUseCase @Inject constructor() {
    private val DAYS_WHOLE_BLOOD = 84 // 12 tuần
    private val DAYS_PLATELETS_PLASMA = 14 // 2 tuần

    operator fun invoke(userProfile: UserProfile?): String {
        val dateString = userProfile?.lastDonationDate
        if (dateString.isNullOrBlank()) {
            return "Bạn có thể hiến máu ngay!"
        }

        // 1. Chuyển đổi String "14/12/2025" sang Date
        val dateFormat = SimpleDateFormat("dd/MM/yyyy", Locale.getDefault())
        val lastDonationDate: Date = try {
            dateFormat.parse(dateString) ?: return "Dữ liệu ngày không hợp lệ"
        } catch (e: Exception) {
            return "Bạn có thể hiến máu ngay!"
        }

        // 2. Kiểm tra loại hiến (Map từ Tiếng Việt trên Firestore sang logic)
        // Trên Firestore ghi: "Máu toàn phần"
        val lastType = userProfile.lastDonationType?.lowercase(Locale.getDefault()) ?: ""

        val waitingDays = when {
            lastType.contains("tiểu cầu") || lastType.contains("huyết tương") || lastType ==
"platelets" -> DAYS_PLATELETS_PLASMA
            else -> DAYS_WHOLE_BLOOD // Mặc định (Máu toàn phần)
        }

        // 3. Tính toán ngày
        val calendar = Calendar.getInstance()
        calendar.time = lastDonationDate
        calendar.add(Calendar.DAY_OF_YEAR, waitingDays)

        val nextAvailableDate = calendar.time
        val today = Date()

        // Reset giờ phút giây về 0 để so sánh ngày chính xác
        val todayCal = Calendar.getInstance().apply {
            time = today
            set(Calendar.HOUR_OF_DAY, 0); set(Calendar.MINUTE, 0); set(Calendar.SECOND, 0);
set(Calendar.MILLISECOND, 0)
        }
        val nextCal = Calendar.getInstance().apply {
            time = nextAvailableDate
            set(Calendar.HOUR_OF_DAY, 0); set(Calendar.MINUTE, 0); set(Calendar.SECOND, 0);
set(Calendar.MILLISECOND, 0)
        }

        if (nextCal.before(todayCal) || nextCal == todayCal) {
            return "Bạn có thể hiến máu ngay!"
        }

        val diffInMillis = nextCal.timeInMillis - todayCal.timeInMillis
        val daysRemaining = TimeUnit.MILLISECONDS.toDays(diffInMillis)

        val typeText = when {
            lastType.contains("tiểu cầu") -> "tiểu cầu"
            lastType.contains("huyết tương") -> "huyết tương"
            else -> "máu toàn phần"
        }

        return "Bạn có thể hiến $typeText sau $daysRemaining ngày nữa
(${dateFormat.format(nextAvailableDate)})"
    }
}
```

feature_profile/src/main/java/com/example/feature_profile/domain/usecase/GetDonationHistoryUseCase.kt

```
//D:\SmartBloodDonationAndroid\feature_profile\src\main\java\com\example\feature_p
rofile\domain\usecase\GetDonationHistoryUseCase.kt
package com.smartblood.profile.domain.usecase

import com.example.feature_profile.domain.repository.ProfileRepository
import javax.inject.Inject

class GetDonationHistoryUseCase @Inject constructor(
    private val repository: ProfileRepository
) {
    suspend operator fun invoke() = repository.getDonationHistory()
}
```

feature_profile/src/main/java/com/example/feature_profile/domain/usecase/GetUserProfileUseCase.kt

```
package com.example.feature_profile.domain.usecase

import com.example.feature_profile.domain.repository.ProfileRepository
import javax.inject.Inject

class GetUserProfileUseCase @Inject constructor(
    private val repository: ProfileRepository
) {
    suspend operator fun invoke() = repository.getUserProfile()
}
```

feature_profile/src/main/java/com/example/feature_profile/domain/usecase/SignOutUseCase.kt

```
package com.example.feature_profile.domain.usecase

import com.example.feature_profile.domain.repository.ProfileRepository
import javax.inject.Inject

class SignOutUseCase @Inject constructor(
    private val repository: ProfileRepository
) {
    operator fun invoke() {
        repository.signOut()
    }
}
```

feature_profile/src/main/java/com/example/feature_profile/domain/usecase/UpdateUserProfileUseCase.kt

```
//D:\SmartBloodDonationAndroid\feature_profile\src\main\java\com\example\feature_p
rofile\domain\usecase\UpdateUserProfileUseCase.kt
package com.example.feature_profile.domain.usecase

import com.smartblood.core.domain.model.UserProfile
import com.example.feature_profile.domain.repository.ProfileRepository
import javax.inject.Inject

class UpdateUserProfileUseCase @Inject constructor(
    private val repository: ProfileRepository
) {
    suspend operator fun invoke(userProfile: UserProfile) =
repository.updateUserProfile(userProfile)
}
```

feature_profile/src/main/java/com/example/feature_profile/ui/DonationHistoryScreen.kt

```
package com.example.feature_profile.ui

import androidx.compose.foundation.BorderStroke
import androidx.compose.foundation.background
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.items
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.automirrored.filled.ArrowBack
import androidx.compose.material.icons.filled.CheckCircle
import androidx.compose.material.icons.filled.Description
import androidx.compose.material.icons.filled.Star
import androidx.compose.material.icons.filled.VerifiedUser
import androidx.compose.material.icons.outlined.StarBorder
import androidx.compose.material3.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.platform.LocalUriHandler
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.dp
import androidx.hilt.navigation.compose.hiltViewModel
// Import ViewModel từ module emergency
import com.example.feature_emergency.ui.history.EmergencyHistoryViewModel
import com.example.feature_emergency.domain.model.EmergencyDonationRecord
import com.smartblood.core.ui.theme.PrimaryRed
import com.smartblood.profile.domain.model.DonationRecord
import java.text.SimpleDateFormat
import java.util.Locale

@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun DonationHistoryScreen(
   // ViewModel cho lịch hẹn thường
   viewModel: DonationHistoryViewModel = hiltViewModel(),
   // ViewModel cho lịch sử khẩn cấp (Inject thêm vào đây)
   emergencyViewModel: EmergencyHistoryViewModel = hiltViewModel(),
   onNavigateBack: () -> Unit
) {
```

```
val state by viewModel.state.collectAsState()
val emergencyState by emergencyViewModel.state.collectAsState()

// Quản lý trạng thái Tab (0: Đặt lịch, 1: Khẩn cấp)
var selectedTabIndex by remember { mutableIntStateOf(0) }
val tabs = listOf("Theo lịch hẹn", "Khẩn cấp")

Scaffold(
  topBar = {
    TopAppBar(
      title = { Text("Lịch sử hiến máu") },
      navigationIcon = {
        IconButton(onClick = onNavigateBack) {
          Icon(Icons.AutoMirrored.Filled.ArrowBack, "Back")
        }
      },
      colors = TopAppBarDefaults.topAppBarColors(containerColor = Color.White)
    )
  },
  containerColor = Color(0xFFF5F5F5) // Màu nền xám nhạt
) { paddingValues ->
  Column(
    modifier = Modifier
      .fillMaxSize()
      .padding(paddingValues)
  ) {
    // --- TAB ROW ---
    TabRow(
      selectedTabIndex = selectedTabIndex,
      containerColor = Color.White,
      contentColor = MaterialTheme.colorScheme.primary
    ) {
      tabs.forEachIndexed { index, title ->
        Tab(
          selected = selectedTabIndex == index,
          onClick = { selectedTabIndex = index },
          text = { Text(title, fontWeight = FontWeight.SemiBold) }
        )
      }
    }

    Spacer(modifier = Modifier.height(8.dp))
```

```
// --- NỘI DUNG THEO TAB ---
when (selectedTabIndex) {
  0 -> {
    // TAB 1: Lịch hẹn thông thường (Code cũ)
    if (state.isLoading) {
      Box(Modifier.fillMaxSize(), contentAlignment = Alignment.Center) {
        CircularProgressIndicator()
      }
    } else if (state.history.isEmpty()) {
      EmptyHistoryView("Bạn chưa có lịch sử đặt hẹn.")
    } else {
      LazyColumn(
        modifier = Modifier.fillMaxSize(),
        contentPadding = PaddingValues(16.dp),
        verticalArrangement = Arrangement.spacedBy(12.dp)
      ) {
        items(state.history) { record ->
          DonationHistoryItem(record = record)
        }
      }
    }
  }
  1 -> {
    // TAB 2: Lịch sử khẩn cấp (Code mới)
    if (emergencyState.isLoading) {
      Box(Modifier.fillMaxSize(), contentAlignment = Alignment.Center) {
        CircularProgressIndicator()
      }
    } else if (emergencyState.history.isEmpty()) {
      EmptyHistoryView("Bạn chưa có lịch sử hiến khẩn cấp.")
    } else {
      LazyColumn(
        modifier = Modifier.fillMaxSize(),
        contentPadding = PaddingValues(16.dp),
        verticalArrangement = Arrangement.spacedBy(12.dp)
      ) {
        items(emergencyState.history) { record ->
          EmergencyHistoryItemCard(record = record)
        }
      }
    }
  }
}
```

```
        }
    }
}

// --- UI COMPONENTS ---

@Composable
fun EmptyHistoryView(message: String) {
    Box(Modifier.fillMaxSize(), contentAlignment = Alignment.Center) {
        Text(text = message, color = Color.Gray)
    }
}

// Item cho Lịch hẹn thường (Giữ nguyên logic cũ nhưng cập nhật UI đẹp hơn)
@Composable
fun DonationHistoryItem(record: DonationRecord) {
    val dateFormat = remember { SimpleDateFormat("dd/MM/yyyy", Locale.getDefault()) }
    val uriHandler = LocalUriHandler.current

    Card(
        modifier = Modifier.fillMaxWidth(),
        elevation = CardDefaults.cardElevation(defaultElevation = 2.dp),
        colors = CardDefaults.cardColors(containerColor = Color.White),
        shape = RoundedCornerShape(12.dp)
    ) {
        Column(modifier = Modifier.padding(16.dp)) {
            // Header: Bệnh viện & Trạng thái
            Row(verticalAlignment = Alignment.CenterVertically) {
                Column(modifier = Modifier.weight(1f)) {
                    Text(
                        text = record.hospitalName,
                        style = MaterialTheme.typography.titleMedium,
                        fontWeight = FontWeight.Bold
                    )
                    Text(
                        text = record.hospitalAddress,
                        style = MaterialTheme.typography.bodySmall,
                        color = Color.Gray
                    )
                }
                Icon(
                    imageVector = Icons.Default.CheckCircle,
                    contentDescription = null,
                    tint = Color(0xFF388E3C) // Xanh lá
                )
            }

            HorizontalDivider(
                modifier = Modifier.padding(vertical = 12.dp),
                color = Color.LightGray.copy(alpha = 0.3f)
            )

            // Thông tin chi tiết: Ngày & Lượng máu
            Row(
                horizontalArrangement = Arrangement.SpaceBetween,
                modifier = Modifier.fillMaxWidth()
            ) {
                Text("Ngày hiến máu:", color = Color.Gray, style =
MaterialTheme.typography.bodyMedium)
                Text(dateFormat.format(record.date), fontWeight = FontWeight.SemiBold)
            }

            // --- CẬP NHẬT MỚI: Actual Volume ---
            if (!record.actualVolume.isNullOrEmpty()) {
                Spacer(modifier = Modifier.height(4.dp))
                Row(
                    horizontalArrangement = Arrangement.SpaceBetween,
                    modifier = Modifier.fillMaxWidth()
                ) {
                    Text("Dung tích:", color = Color.Gray, style =
MaterialTheme.typography.bodyMedium)
                    Text(record.actualVolume, fontWeight = FontWeight.Bold, color = PrimaryRed)
                }
            }

            // --- CẬP NHẬT MỚI: KẾT QUẢ XÉT NGHIỆM (LAB RESULT) ---
            if (record.labResult != null) {
                Spacer(modifier = Modifier.height(12.dp))
                Column(
                    modifier = Modifier
                        .fillMaxWidth()
                        .background(Color(0xFFE3F2FD).copy(alpha = 0.5f),
RoundedCornerShape(8.dp))
                        .padding(12.dp)
                ) {
                    Row(verticalAlignment = Alignment.CenterVertically) {
```

```
            Icon(Icons.Default.Description, contentDescription = null, tint =
Color(0xFF1565C0), modifier = Modifier.size(16.dp))
            Spacer(modifier = Modifier.width(4.dp))
            Text(
              text = "Kết quả xét nghiệm",
              style = MaterialTheme.typography.labelLarge,
              fontWeight = FontWeight.Bold,
              color = Color(0xFF1565C0)
            )
          }

          // Hiển thị lời dặn/kết luận
          if (!record.labResult.conclusion.isNullOrEmpty()) {
            Spacer(modifier = Modifier.height(4.dp))
            Text(
              text = "Bác sĩ: ${record.labResult.conclusion}",
              style = MaterialTheme.typography.bodyMedium,
              color = Color.Black.copy(alpha = 0.8f)
            )
          }

          // Nút mở file PDF
          val docUrl = record.labResult.documentUrl
          if (!docUrl.isNullOrEmpty()) {
            Spacer(modifier = Modifier.height(8.dp))
            Button(
              onClick = {
                try {
                  uriHandler.openUri(docUrl)
                } catch (e: Exception) {
                  // Bắt lỗi nếu link hỏng hoặc không có trình duyệt
                }
              },
              modifier = Modifier.fillMaxWidth().height(36.dp),
              colors = ButtonDefaults.buttonColors(
                containerColor = Color(0xFF2196F3),
                contentColor = Color.White
              ),
              contentPadding = PaddingValues(0.dp),
              shape = RoundedCornerShape(6.dp)
            ) {
              Text("Xem file chi tiết (PDF)", style =
MaterialTheme.typography.labelMedium)
```

```
                }
            }
        }
    }
    // Nút xem chứng nhận (Cũ - Giữ lại)
    if (!record.certificateUrl.isNullOrEmpty()) {
        Spacer(modifier = Modifier.height(12.dp))
        OutlinedButton(
            onClick = { uriHandler.openUri(record.certificateUrl!!) },
            modifier = Modifier.fillMaxWidth().height(40.dp),
            colors = ButtonDefaults.outlinedButtonColors(
                contentColor = Color(0xFF388E3C)
            ),
            border = BorderStroke(1.dp, Color(0xFF388E3C).copy(alpha = 0.5f)),
            shape = RoundedCornerShape(8.dp)
        ) {
            Icon(Icons.Default.VerifiedUser, null, modifier = Modifier.size(16.dp))
            Spacer(modifier = Modifier.width(8.dp))
            Text("Chứng Nhận Hiến Máu", style = MaterialTheme.typography.labelLarge)
        }
    }
  }
 }
}

// Item cho Lịch sử Khẩn cấp (MỚI)
@Composable
fun EmergencyHistoryItemCard(record: EmergencyDonationRecord) {
  val dateFormat = remember { SimpleDateFormat("dd/MM/yyyy", Locale.getDefault()) }
  val uriHandler = LocalUriHandler.current

  val (statusColor, statusText) = when(record.status) {
    "Completed" -> Color(0xFF4CAF50) to "Đã hiến"
    "Pending" -> Color(0xFFFF9800) to "Đang chờ"
    "Cancelled" -> Color(0xFFF44336) to "Đã hủy"
    else -> Color.Gray to record.status
  }

  Card(
    modifier = Modifier.fillMaxWidth(),
    elevation = CardDefaults.cardElevation(defaultElevation = 2.dp),
    colors = CardDefaults.cardColors(containerColor = Color.White),
    shape = RoundedCornerShape(12.dp)
  ) {
    Column(modifier = Modifier.padding(16.dp)) {
      // Header
      Row(verticalAlignment = Alignment.Top) {
        Column(modifier = Modifier.weight(1f)) {
          Text(
            text = record.hospitalName,
            style = MaterialTheme.typography.titleMedium,
            fontWeight = FontWeight.Bold
          )
          Text(
            text = "Yêu cầu khẩn cấp",
            style = MaterialTheme.typography.bodySmall,
            color = Color.Red
          )
        }
        Surface(
          color = statusColor.copy(alpha = 0.1f),
          shape = RoundedCornerShape(16.dp)
        ) {
          Text(
            text = statusText,
            color = statusColor,
            style = MaterialTheme.typography.labelSmall,
            modifier = Modifier.padding(horizontal = 8.dp, vertical = 4.dp),
            fontWeight = FontWeight.Bold
          )
        }
      }

      HorizontalDivider(modifier = Modifier.padding(vertical = 12.dp), color =
Color.LightGray.copy(alpha = 0.3f))

      // Thông tin chi tiết
      Row(horizontalArrangement = Arrangement.SpaceBetween, modifier =
Modifier.fillMaxWidth()) {
        Text("Ngày tiếp nhận:", color = Color.Gray, style =
MaterialTheme.typography.bodyMedium)
        Text(dateFormat.format(record.pledgedAt), fontWeight = FontWeight.SemiBold,
style = MaterialTheme.typography.bodyMedium)
      }
      Spacer(modifier = Modifier.height(4.dp))
      Row(horizontalArrangement = Arrangement.SpaceBetween, modifier =
Modifier.fillMaxWidth()) {
            Text("Nhóm máu hiến:", color = Color.Gray, style =
MaterialTheme.typography.bodyMedium)
            Text(record.userBloodType, fontWeight = FontWeight.Bold, color = Color.Red, style
= MaterialTheme.typography.bodyMedium)
        }

        // Phần đánh giá từ bệnh viện (Chỉ hiện khi đã hoàn thành)
        if (record.status == "Completed") {
            Spacer(modifier = Modifier.height(12.dp))
            Row(verticalAlignment = Alignment.CenterVertically, modifier =
Modifier.background(Color(0xFFFFF8E1),
RoundedCornerShape(8.dp)).padding(8.dp).fillMaxWidth()) {
                Column {
                    Row(verticalAlignment = Alignment.CenterVertically) {
                        Text("Đánh giá:", style = MaterialTheme.typography.labelMedium,
fontWeight = FontWeight.Bold)
                        Spacer(modifier = Modifier.width(8.dp))
                        repeat(5) { index ->
                            Icon(
                                imageVector = if (index < record.rating) Icons.Filled.Star else
Icons.Outlined.StarBorder,
                                contentDescription = null,
                                tint = Color(0xFFFFC107),
                                modifier = Modifier.size(14.dp)
                            )
                        }
                    }
                    if (!record.review.isNullOrBlank()) {
                        Spacer(modifier = Modifier.height(2.dp))
                        Text(
                            text = "\"${record.review}\"",
                            style = MaterialTheme.typography.bodySmall,
                            fontStyle = androidx.compose.ui.text.font.FontStyle.Italic,
                            color = Color.DarkGray
                        )
                    }
                }
            }
        }

        // Kiểm tra nếu có link chứng nhận
        record.certificateUrl?.let { url ->
```

```
            if (url.isNotBlank()) {
                Spacer(modifier = Modifier.height(12.dp))
                Button(
                    onClick = { uriHandler.openUri(url) }, // Dùng biến 'url' đã được smart cast an
toàn
                    modifier = Modifier.fillMaxWidth().height(40.dp),
                    colors = ButtonDefaults.buttonColors(
                        containerColor = Color(0xFFE8F5E9),
                        contentColor = Color(0xFF2E7D32)
                    ),
                    shape = RoundedCornerShape(8.dp),
                    contentPadding = PaddingValues(0.dp)
                ) {
                    Icon(Icons.Default.Description, null, modifier = Modifier.size(16.dp))
                    Spacer(modifier = Modifier.width(8.dp))
                    Text("Xem Chứng Nhận Hiến Máu", style =
MaterialTheme.typography.labelLarge)
                }
            }
        }
    }
  }
}
```

feature_profile/src/main/java/com/example/feature_profile/ui/DonationHistoryViewModel.kt

```
package com.example.feature_profile.ui

//
feature_profile/src/main/java/com/smartblood/profile/ui/history/DonationHistoryView
Model.kt


import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import com.smartblood.profile.domain.model.DonationRecord
import com.smartblood.profile.domain.usecase.GetDonationHistoryUseCase
import dagger.hilt.android.lifecycle.HiltViewModel
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.asStateFlow
import kotlinx.coroutines.flow.update
import kotlinx.coroutines.launch
import javax.inject.Inject
```

```
data class DonationHistoryState(
    val isLoading: Boolean = true,
    val history: List<DonationRecord> = emptyList(),
    val error: String? = null
)

@HiltViewModel
class DonationHistoryViewModel @Inject constructor(
    private val getDonationHistoryUseCase: GetDonationHistoryUseCase
) : ViewModel() {

    private val _state = MutableStateFlow(DonationHistoryState())
    val state = _state.asStateFlow()

    init {
        loadHistory()
    }

    private fun loadHistory() {
        viewModelScope.launch {
            _state.update { it.copy(isLoading = true) }
            getDonationHistoryUseCase()
                .onSuccess { historyList ->
                    _state.update { it.copy(isLoading = false, history = historyList) }
                }
                .onFailure { error ->
                    _state.update { it.copy(isLoading = false, error = error.message) }
                }
        }
    }
}
```

feature_profile/src/main/java/com/example/feature_profile/ui/EditProfileScreen.kt

```
package com.example.feature_profile.ui

// feature_profile/src/main/java/com/smartblood/profile/ui/edit/EditProfileScreen.kt


import androidx.compose.foundation.layout.*
import androidx.compose.foundation.rememberScrollState
import androidx.compose.foundation.verticalScroll
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.automirrored.filled.ArrowBack
import androidx.compose.material3.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.unit.dp
import androidx.hilt.navigation.compose.hiltViewModel

@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun EditProfileScreen(
  viewModel: EditProfileViewModel = hiltViewModel(),
  onNavigateBack: () -> Unit
) {
  val state by viewModel.state.collectAsState()

  // Tự động quay lại khi lưu thành công
  LaunchedEffect(state.saveSuccess) {
    if (state.saveSuccess) {
      onNavigateBack()
    }
  }

  Scaffold(
    topBar = {
      TopAppBar(
        title = { Text("Chỉnh sửa hồ sơ") },
        navigationIcon = {
          IconButton(onClick = onNavigateBack) {
            Icon(Icons.AutoMirrored.Filled.ArrowBack, "Back")
          }
        }
      )
    }
  ) { paddingValues ->
    Box(
      modifier = Modifier
        .fillMaxSize()
        .padding(paddingValues)
    ) {
      if (state.isLoading) {
        CircularProgressIndicator(modifier = Modifier.align(Alignment.Center))
        } else if (state.error != null) {
            Text(text = "Lỗi: ${state.error}", modifier = Modifier.align(Alignment.Center))
        } else {
            Column(
                modifier = Modifier
                    .fillMaxSize()
                    .padding(16.dp)
                    .verticalScroll(rememberScrollState()),
                verticalArrangement = Arrangement.spacedBy(16.dp)
            ) {
                OutlinedTextField(
                    value = state.fullName,
                    onValueChange = {
viewModel.onEvent(EditProfileEvent.OnFullNameChanged(it)) },
                    label = { Text("Họ và tên") },
                    modifier = Modifier.fillMaxWidth()
                )

                OutlinedTextField(
                    value = state.phoneNumber,
                    onValueChange = {
viewModel.onEvent(EditProfileEvent.OnPhoneNumberChanged(it)) },
                    label = { Text("Số điện thoại") },
                    modifier = Modifier.fillMaxWidth()
                )

                OutlinedTextField(
                    value = state.bloodType,
                    onValueChange = {
viewModel.onEvent(EditProfileEvent.OnBloodTypeChanged(it)) },
                    label = { Text("Nhóm máu (ví dụ: A+, O-)") },
                    modifier = Modifier.fillMaxWidth()
                )

                Spacer(modifier = Modifier.weight(1f))

                Button(
                    onClick = { viewModel.onEvent(EditProfileEvent.OnSaveClicked) },
                    enabled = !state.isSaving,
                    modifier = Modifier
                        .fillMaxWidth()
                        .height(50.dp)
                ) {
```

```
                if (state.isSaving) {
                    CircularProgressIndicator(
                        modifier = Modifier.size(24.dp),
                        color = MaterialTheme.colorScheme.onPrimary,
                        strokeWidth = 2.dp
                    )
                } else {
                    Text("Lưu thay đổi")
                }
            }
        }
    }
}
}
}
```

feature_profile/src/main/java/com/example/feature_profile/ui/EditProfileViewModel.kt

```
package com.example.feature_profile.ui

import android.net.Uri
import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import com.example.feature_profile.domain.usecase.GetUserProfileUseCase
import com.example.feature_profile.domain.usecase.UpdateUserProfileUseCase
import com.smartblood.core.storage.domain.usecase.UploadImageUseCase
import com.smartblood.core.domain.model.UserProfile
import dagger.hilt.android.lifecycle.HiltViewModel
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.asStateFlow
import kotlinx.coroutines.flow.update
import kotlinx.coroutines.launch
import javax.inject.Inject

data class EditProfileState(
    val isLoading: Boolean = true,
    val isSaving: Boolean = false,
    val saveSuccess: Boolean = false,
    val error: String? = null,
    val fullName: String = "",
    val bloodType: String = "",
    val phoneNumber: String = "",
    val avatarUri: Uri? = null,
    val isUploading: Boolean = false,
    val originalProfile: UserProfile? = null
) {
    fun toUserProfile(): UserProfile? {
        return originalProfile?.copy(
            fullName = fullName,
            bloodType = bloodType.ifBlank { null },
            phoneNumber = phoneNumber.ifBlank { null }
        )
    }
}

sealed class EditProfileEvent {
    data class OnFullNameChanged(val value: String) : EditProfileEvent()
    data class OnBloodTypeChanged(val value: String) : EditProfileEvent()
    data class OnPhoneNumberChanged(val value: String) : EditProfileEvent()
    data class OnAvatarChanged(val uri: Uri?) : EditProfileEvent()
    object OnSaveClicked : EditProfileEvent()
}

@HiltViewModel
class EditProfileViewModel @Inject constructor(
    private val getUserProfileUseCase: GetUserProfileUseCase,
    private val updateUserProfileUseCase: UpdateUserProfileUseCase,
    private val uploadImageUseCase: UploadImageUseCase
) : ViewModel() {

    private val _state = MutableStateFlow(EditProfileState())
    val state = _state.asStateFlow()

    init {
        loadInitialProfile()
    }

    fun onEvent(event: EditProfileEvent) {
        when(event) {
            is EditProfileEvent.OnFullNameChanged -> _state.update { it.copy(fullName =
event.value) }
            is EditProfileEvent.OnBloodTypeChanged -> _state.update { it.copy(bloodType =
event.value) }
            is EditProfileEvent.OnPhoneNumberChanged -> _state.update { it.copy(phoneNumber
= event.value) }
            // ĐÃ SỬA: Chỉ giữ lại một nhánh OnSaveClicked
            EditProfileEvent.OnSaveClicked -> saveProfile()
            is EditProfileEvent.OnAvatarChanged -> {
                _state.update { it.copy(avatarUri = event.uri, error = null) }
            }
        }
    }

    private fun loadInitialProfile() {
        viewModelScope.launch {
            _state.update { it.copy(isLoading = true) }
            getUserProfileUseCase()
                .onSuccess { profile ->
                    _state.update {
                        it.copy(
                            isLoading = false,
                            originalProfile = profile,
                            fullName = profile.fullName,
                            bloodType = profile.bloodType ?: "",
                            phoneNumber = profile.phoneNumber ?: "",
                            avatarUri = profile.avatarUrl?.let { url -> Uri.parse(url) }
                        )
                    }
                }
                .onFailure { error ->
                    _state.update { it.copy(isLoading = false, error = error.message) }
                }
        }
    }

    private fun saveProfile() {
        viewModelScope.launch {
            _state.update { it.copy(isSaving = true, error = null) }

            val currentAvatarUri = _state.value.avatarUri
            var newAvatarUrl: String? = null

            if (currentAvatarUri != null && currentAvatarUri !=
_state.value.originalProfile?.avatarUrl?.let { Uri.parse(it) }) {
                _state.update { it.copy(isUploading = true) }
                // Lưu ý: UploadImageUseCase trả về Result<String>
                uploadImageUseCase(currentAvatarUri,
"avatars/${_state.value.originalProfile?.uid}")
                    .onSuccess { url ->
                newAvatarUrl = url
                _state.update { it.copy(isUploading = false) }
            }
            .onFailure { error ->
                _state.update { it.copy(isSaving = false, isUploading = false, error = "Lỗi tải ảnh:
${error.message}") }
                return@launch
            }
        }

        val updatedProfile = _state.value.toUserProfile()?.copy(
            avatarUrl = newAvatarUrl ?: _state.value.originalProfile?.avatarUrl
        )

        if (updatedProfile == null) {
            _state.update { it.copy(isSaving = false, error = "Không thể cập nhật hồ sơ.") }
            return@launch
        }

        updateUserProfileUseCase(updatedProfile)
            .onSuccess {
                _state.update { it.copy(isSaving = false, saveSuccess = true) }
            }
            .onFailure { error ->
                _state.update { it.copy(isSaving = false, error = "Lỗi cập nhật hồ sơ:
${error.message}") }
            }
        }
    }
}
```

feature_profile/src/main/java/com/example/feature_profile/ui/ProfileContract.kt

```
package com.example.feature_profile.ui

import com.smartblood.core.domain.model.Appointment
import com.smartblood.core.domain.model.BloodRequest
import com.smartblood.core.domain.model.UserProfile

// Cập nhật State
data class ProfileState(
    val isLoading: Boolean = true,
    val isUploading: Boolean = false,
    val userProfile: UserProfile? = null,
    val upcomingAppointments: List<Appointment> = emptyList(),
    val todayAppointments: List<Appointment> = emptyList(),
    val pastAppointments: List<Appointment> = emptyList(),
    val pledgedRequests: List<BloodRequest> = emptyList(),
    val error: String? = null,
    val isSignedOut: Boolean = false
)

// Cập nhật Event
sealed class ProfileEvent {
    object OnEditProfileClicked : ProfileEvent()
    object OnViewDonationHistoryClicked : ProfileEvent()
    object OnSignOutClicked : ProfileEvent()
}
```

feature_profile/src/main/java/com/example/feature_profile/ui/ProfileScreen.kt

```
package com.example.feature_profile.ui

import android.net.Uri
import androidx.activity.compose.rememberLauncherForActivityResult
import androidx.activity.result.contract.ActivityResultContracts
import androidx.compose.foundation.BorderStroke
import androidx.compose.foundation.background
import androidx.compose.foundation.border
import androidx.compose.foundation.clickable
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.grid.GridCells
import androidx.compose.foundation.lazy.grid.LazyVerticalGrid
import androidx.compose.foundation.lazy.grid.items
import androidx.compose.foundation.lazy.items
import androidx.compose.foundation.shape.CircleShape
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.automirrored.filled.ExitToApp
import androidx.compose.material.icons.filled.Close
import androidx.compose.material.icons.filled.Edit
import androidx.compose.material.icons.filled.History
import androidx.compose.material.icons.filled.Image
import androidx.compose.material.icons.filled.PhotoCamera
import androidx.compose.material.icons.filled.Settings
import androidx.compose.material.icons.filled.Star
import androidx.compose.material3.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.clip
import androidx.compose.ui.graphics.Brush
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.style.TextAlign
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.hilt.navigation.compose.hiltViewModel
import coil.compose.AsyncImage
import com.smartblood.core.domain.model.Appointment
import com.smartblood.core.domain.model.UserProfile
import com.smartblood.core.ui.theme.PrimaryRed
import com.smartblood.core.ui.theme.PrimaryRedDark
import java.text.SimpleDateFormat
import java.util.Date
import java.util.Locale

@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun ProfileScreen(
  viewModel: ProfileViewModel = hiltViewModel(),
  onNavigateToEditProfile: () -> Unit,
  onNavigateToDonationHistory: () -> Unit,
  onNavigateToLogin: () -> Unit
) {
  val state by viewModel.state.collectAsState()

  // Biến quản lý trạng thái hiển thị BottomSheet chọn ảnh
  var showAvatarBottomSheet by remember { mutableStateOf(false) }
  val bottomSheetState = rememberModalBottomSheetState()

  // Lắng nghe sự kiện đăng xuất
  LaunchedEffect(state.isSignedOut) {
    if (state.isSignedOut) {
      onNavigateToLogin()
    }
  }
```

```
// Launcher mở thư viện ảnh hệ thống
val imagePickerLauncher = rememberLauncherForActivityResult(
    contract = ActivityResultContracts.GetContent()
) { uri: Uri? ->
    uri?.let {
        viewModel.onAvatarChange(it)
        showAvatarBottomSheet = false // Đóng sheet khi chọn xong
    }
}

Scaffold(
    containerColor = Color(0xFFF8F9FA)
) { paddingValues ->
    if (state.isLoading) {
        Box(Modifier.fillMaxSize(), contentAlignment = Alignment.Center) {
            CircularProgressIndicator(color = PrimaryRed)
        }
    } else if (state.error != null && state.userProfile == null) {
        NotLoggedInView(error = state.error, onLoginClick = onNavigateToLogin)
    } else {
        LazyColumn(
            modifier = Modifier
                .fillMaxSize()
                .padding(bottom = paddingValues.calculateBottomPadding()),
            horizontalAlignment = Alignment.CenterHorizontally
        ) {
            // 1. HEADER PROFILE
            item {
                state.userProfile?.let { profile ->
                    ProfileHeaderSection(
                        profile = profile,
                        isUploading = state.isUploading,
                        onAvatarClick = { showAvatarBottomSheet = true }
                    )
                }
            }

            // 2. CÁC NÚT CHỨC NĂNG
            item {
                Spacer(modifier = Modifier.height(24.dp))
                Row(
                    modifier = Modifier
                .fillMaxWidth()
                .padding(horizontal = 16.dp),
            horizontalArrangement = Arrangement.spacedBy(12.dp)
        ) {
            ActionButton(
                text = "Chỉnh sửa",
                icon = Icons.Default.Edit,
                modifier = Modifier.weight(1f),
                onClick = onNavigateToEditProfile
            )
            ActionButton(
                text = "Lịch sử",
                icon = Icons.Default.History,
                modifier = Modifier.weight(1f),
                onClick = onNavigateToDonationHistory
            )
        }
    }

    // 3. DANH SÁCH YÊU CẦU ĐÃ CHẤP NHẬN
    if (state.pledgedRequests.isNotEmpty()) {
        item { SectionHeader("Yêu cầu đã chấp nhận") }
        items(state.pledgedRequests) { request ->

            // Định dạng ngày giờ
            val dateFormat = remember { SimpleDateFormat("HH:mm - dd/MM/yyyy",
Locale.getDefault()) }

            // Ưu tiên hiển thị ngày người dùng chấp nhận
            val displayDate = request.userPledgedDate ?: request.createdAt
            val dateLabel = if (request.userPledgedDate != null) "Đã nhận lúc:" else "Ngày
tạo:"

            GenericInfoCard(
                title = request.hospitalName,
                detailLine1 = "Cần nhóm máu: ${request.bloodType}
(${request.preferredVolume})",
                // --- CẬP NHẬT DÒNG NÀY ---
                detailLine2 = "$dateLabel ${dateFormat.format(displayDate)}",
                status = "ĐÃ CHẤP NHẬN",
                statusColor = Color(0xFF1976D2)
            )
            Spacer(modifier = Modifier.height(12.dp))
        }
    }

    // 4. DANH SÁCH LỊCH HẸN
    if (state.todayAppointments.isNotEmpty()) {
        item { SectionHeader("Lịch hẹn hôm nay") }
        items(state.todayAppointments) { appt ->
            AppointmentCard(appt, isPast = false)
            Spacer(modifier = Modifier.height(12.dp))
        }
    }

    if (state.upcomingAppointments.isNotEmpty()) {
        item { SectionHeader("Lịch hẹn sắp tới") }
        items(state.upcomingAppointments) { appt ->
            AppointmentCard(appt, isPast = false)
            Spacer(modifier = Modifier.height(12.dp))
        }
    }

    // 5. NÚT ĐĂNG XUẤT
    item {
        Spacer(modifier = Modifier.height(32.dp))
        Button(
            onClick = { viewModel.onEvent(ProfileEvent.OnSignOutClicked) },
            colors = ButtonDefaults.buttonColors(
                containerColor = Color.White,
                contentColor = Color.Red
            ),
            elevation = ButtonDefaults.buttonElevation(defaultElevation = 2.dp),
            modifier = Modifier
                .padding(horizontal = 16.dp)
                .fillMaxWidth()
                .height(50.dp),
            shape = RoundedCornerShape(12.dp)
        ) {
            Icon(Icons.AutoMirrored.Filled.ExitToApp, contentDescription = null)
            Spacer(modifier = Modifier.width(8.dp))
            Text("Đăng xuất")
        }
        Spacer(modifier = Modifier.height(32.dp))
    }
}
```

```
        }

        // --- BOTTOM SHEET CHỌN AVATAR ---
        if (showAvatarBottomSheet) {
            ModalBottomSheet(
                onDismissRequest = { showAvatarBottomSheet = false },
                sheetState = bottomSheetState,
                containerColor = Color.White,
                dragHandle = { BottomSheetDefaults.DragHandle() }
            ) {
                AvatarSelectionSheetContent(
                    onGalleryClick = {
                        imagePickerLauncher.launch("image/*")
                    },
                    onPresetSelected = { uri ->
                        viewModel.onAvatarChange(uri)
                        showAvatarBottomSheet = false
                    },
                    onClose = { showAvatarBottomSheet = false }
                )
            }
        }
    }
}

// --- NỘI DUNG BOTTOM SHEET (CẬP NHẬT: Nút đỏ & 24 Avatar) ---
@Composable
fun AvatarSelectionSheetContent(
    onGalleryClick: () -> Unit,
    onPresetSelected: (Uri) -> Unit,
    onClose: () -> Unit
) {
    // Danh sách 24 Avatar phong phú
    val presets = remember {
        listOf(
            // Phong cách 1: Thám hiểm
            "https://api.dicebear.com/9.x/adventurer/png?seed=Felix",
            "https://api.dicebear.com/9.x/adventurer/png?seed=Aneka",
            "https://api.dicebear.com/9.x/adventurer/png?seed=Snow",
            "https://api.dicebear.com/9.x/adventurer/png?seed=Ginger",
            "https://api.dicebear.com/9.x/adventurer/png?seed=Abner",
            "https://api.dicebear.com/9.x/adventurer/png?seed=Coco",
            // Phong cách 2: Hiện đại
        "https://api.dicebear.com/9.x/avataaars/png?seed=Sophy",
        "https://api.dicebear.com/9.x/avataaars/png?seed=Alexander",
        "https://api.dicebear.com/9.x/avataaars/png?seed=Nolan",
        "https://api.dicebear.com/9.x/avataaars/png?seed=Zoe",
        "https://api.dicebear.com/9.x/avataaars/png?seed=Midnight",
        "https://api.dicebear.com/9.x/avataaars/png?seed=Luna",
        // Phong cách 3: Nghệ thuật
        "https://api.dicebear.com/9.x/lorelei/png?seed=Robert",
        "https://api.dicebear.com/9.x/lorelei/png?seed=Mitten",
        "https://api.dicebear.com/9.x/lorelei/png?seed=Gizmo",
        "https://api.dicebear.com/9.x/lorelei/png?seed=Cali",
        "https://api.dicebear.com/9.x/lorelei/png?seed=Oreo",
        "https://api.dicebear.com/9.x/lorelei/png?seed=Boots",
        // Phong cách 4: Tối giản
        "https://api.dicebear.com/9.x/micah/png?seed=Callie",
        "https://api.dicebear.com/9.x/micah/png?seed=Simba",
        "https://api.dicebear.com/9.x/micah/png?seed=Pepper",
        "https://api.dicebear.com/9.x/micah/png?seed=Bubba",
        "https://api.dicebear.com/9.x/micah/png?seed=Missy",
        "https://api.dicebear.com/9.x/micah/png?seed=Scooter"
    )
}

Column(
    modifier = Modifier
        .fillMaxWidth()
        .padding(horizontal = 24.dp)
        .padding(bottom = 48.dp)
) {
    // Header
    Box(
        modifier = Modifier.fillMaxWidth(),
        contentAlignment = Alignment.Center
    ) {
        Text(
            text = "Đổi ảnh đại diện",
            style = MaterialTheme.typography.titleLarge,
            fontWeight = FontWeight.Bold
        )
        IconButton(
            onClick = onClose,
            modifier = Modifier.align(Alignment.CenterEnd)
        ) {
```

```
        Icon(Icons.Default.Close, contentDescription = "Close")
    }
}

Spacer(modifier = Modifier.height(24.dp))

// === NÚT MÀU ĐỎ (PRIMARY RED) ===
Button(
    onClick = onGalleryClick,
    modifier = Modifier
        .fillMaxWidth()
        .height(56.dp),
    shape = RoundedCornerShape(12.dp),
    colors = ButtonDefaults.buttonColors(
        containerColor = PrimaryRed, // Đỏ
        contentColor = Color.White   // Trắng
    ),
    elevation = ButtonDefaults.buttonElevation(defaultElevation = 2.dp)
) {
    Icon(Icons.Default.Image, contentDescription = null)
    Spacer(modifier = Modifier.width(12.dp))
    Text("Tải ảnh từ thư viện máy", fontSize = 16.sp, fontWeight = FontWeight.SemiBold)
}
// ================================

Spacer(modifier = Modifier.height(24.dp))

Text(
    text = "Hoặc chọn Avatar có sẵn",
    style = MaterialTheme.typography.labelLarge,
    color = Color.Gray
)
Spacer(modifier = Modifier.height(12.dp))

// Danh sách Avatar (Lưới cao hơn để hiển thị nhiều hình)
LazyVerticalGrid(
    columns = GridCells.Fixed(4),
    verticalArrangement = Arrangement.spacedBy(12.dp),
    horizontalArrangement = Arrangement.spacedBy(12.dp),
    modifier = Modifier.height(300.dp) // Tăng chiều cao khung nhìn
) {
    items(presets) { url ->
        Surface(
                shape = CircleShape,
                modifier = Modifier
                    .size(70.dp)
                    .clickable { onPresetSelected(Uri.parse(url)) },
                color = Color(0xFFF0F0F0),
                border = androidx.compose.foundation.BorderStroke(1.dp, Color.LightGray)
            ) {
                AsyncImage(
                    model = url,
                    contentDescription = null,
                    modifier = Modifier.padding(4.dp).clip(CircleShape),
                    contentScale = ContentScale.Crop
                )
            }
        }
    }
  }
}

// --- CÁC COMPOSABLE KHÁC GIỮ NGUYÊN ---

@Composable
fun ProfileHeaderSection(
    profile: UserProfile,
    isUploading: Boolean,
    onAvatarClick: () -> Unit
) {
    Box(
        modifier = Modifier.fillMaxWidth(),
        contentAlignment = Alignment.TopCenter
    ) {
        // ... (Phần Background giữ nguyên) ...
        Box(
            modifier = Modifier
                .fillMaxWidth()
                .height(160.dp)
                .clip(RoundedCornerShape(bottomStart = 50.dp, bottomEnd = 50.dp))
                .background(
                    Brush.verticalGradient(colors = listOf(PrimaryRed, PrimaryRedDark))
                )
        )

        Column(
        horizontalAlignment = Alignment.CenterHorizontally,
        modifier = Modifier.padding(top = 100.dp)
    ) {
        // ... (Phần Avatar giữ nguyên) ...
        Box(contentAlignment = Alignment.Center) {
            Surface(
                shape = CircleShape,
                color = Color.White,
                modifier = Modifier.size(128.dp),
                shadowElevation = 4.dp
            ) {
                Box(
                    modifier = Modifier.fillMaxSize(),
                    contentAlignment = Alignment.Center
                ) {
                    AsyncImage(
                        model = profile.avatarUrl ?: "https://ui-
avatars.com/api/?name=${profile.fullName}&background=random&size=256",
                        contentDescription = "Avatar",
                        contentScale = ContentScale.Crop,
                        modifier = Modifier
                            .size(120.dp)
                            .clip(CircleShape)
                            .clickable { onAvatarClick() }
                    )
                }
            }
            // ... (Icon Camera nhỏ giữ nguyên) ...
            Surface(
                shape = CircleShape,
                color = Color(0xFFE0E0E0),
                modifier = Modifier
                    .align(Alignment.BottomEnd)
                    .offset(x = (-10).dp, y = (-10).dp)
                    .size(36.dp)
                    .clickable { onAvatarClick() }
            ) {
                Icon(
                    imageVector = Icons.Default.PhotoCamera,
                    contentDescription = "Edit Avatar",
                    modifier = Modifier.padding(8.dp),
                    tint = Color.Gray
                )
    }

    if (isUploading) {
        CircularProgressIndicator(modifier = Modifier.align(Alignment.Center))
    }
}

Spacer(modifier = Modifier.height(16.dp))

// Tên và Email
Text(
    text = profile.fullName,
    style = MaterialTheme.typography.headlineMedium,
    fontWeight = FontWeight.Bold,
    color = Color.Black
)
Text(
    text = profile.email,
    style = MaterialTheme.typography.bodyMedium,
    color = Color.Gray
)

Spacer(modifier = Modifier.height(12.dp))

// --- CẬP NHẬT MỚI: HUY HIỆU & SỐ LẦN HIẾN ---
Row(
    horizontalArrangement = Arrangement.spacedBy(8.dp),
    verticalAlignment = Alignment.CenterVertically
) {
    // Badge Nhóm máu
    Surface(
        color = PrimaryRed.copy(alpha = 0.1f),
        shape = RoundedCornerShape(20.dp),
        border = BorderStroke(1.dp, PrimaryRed.copy(alpha = 0.2f))
    ) {
        Text(
            text = "Nhóm: ${profile.bloodType ?: "N/A"}",
            modifier = Modifier.padding(horizontal = 12.dp, vertical = 6.dp),
            style = MaterialTheme.typography.labelLarge,
            color = PrimaryRedDark,
            fontWeight = FontWeight.Bold
        )
    }
```

```
        // Badge Số lần hiến
        Surface(
            color = Color(0xFFE3F2FD), // Xanh nhạt
            shape = RoundedCornerShape(20.dp),
            border = BorderStroke(1.dp, Color(0xFF2196F3).copy(alpha = 0.3f))
        ) {
            Text(
                text = "${profile.donationCount} lần hiến",
                modifier = Modifier.padding(horizontal = 12.dp, vertical = 6.dp),
                style = MaterialTheme.typography.labelLarge,
                color = Color(0xFF1565C0), // Xanh đậm
                fontWeight = FontWeight.Bold
            )
        }
    }

    // Badge Ưu tiên (Chỉ hiện nếu isPriority = true)
    if (profile.isPriority) {
        Spacer(modifier = Modifier.height(8.dp))
        Surface(
            color = Color(0xFFFFF8E1), // Vàng nhạt
            shape = RoundedCornerShape(20.dp),
            border = BorderStroke(1.dp, Color(0xFFFFC107))
        ) {
            Row(
                modifier = Modifier.padding(horizontal = 12.dp, vertical = 6.dp),
                verticalAlignment = Alignment.CenterVertically
            ) {
                Icon(
                    imageVector = Icons.Default.Star,
                    contentDescription = null,
                    tint = Color(0xFFFFC107), // Màu vàng
                    modifier = Modifier.size(16.dp)
                )
                Spacer(modifier = Modifier.width(4.dp))
                Text(
                    text = "Người hiến ưu tiên",
                    style = MaterialTheme.typography.labelLarge,
                    color = Color(0xFFFF8F00), // Cam đậm
                    fontWeight = FontWeight.Bold
                )
            }
```

```
                }
            }
        }
    }
}

@Composable
fun ActionButton(
    text: String,
    icon: androidx.compose.ui.graphics.vector.ImageVector,
    modifier: Modifier = Modifier,
    onClick: () -> Unit
) {
    Button(
        onClick = onClick,
        modifier = modifier.height(50.dp),
        shape = RoundedCornerShape(12.dp),
        colors = ButtonDefaults.buttonColors(containerColor = PrimaryRed),
        elevation = ButtonDefaults.buttonElevation(defaultElevation = 2.dp)
    ) {
        Icon(imageVector = icon, contentDescription = null, modifier = Modifier.size(20.dp))
        Spacer(modifier = Modifier.width(8.dp))
        Text(text = text, fontWeight = FontWeight.SemiBold)
    }
}

@Composable
fun SectionHeader(title: String) {
    Text(
        text = title,
        style = MaterialTheme.typography.titleMedium,
        fontWeight = FontWeight.Bold,
        modifier = Modifier
            .fillMaxWidth()
            .padding(start = 16.dp, end = 16.dp, top = 24.dp, bottom = 8.dp),
        color = Color.Black.copy(alpha = 0.8f)
    )
}

@Composable
fun GenericInfoCard(
    title: String,
    detailLine1: String,
    detailLine2: String,
    status: String,
    statusColor: Color
) {
    Card(
        modifier = Modifier
            .fillMaxWidth()
            .padding(horizontal = 16.dp),
        colors = CardDefaults.cardColors(containerColor = Color.White),
        elevation = CardDefaults.cardElevation(defaultElevation = 2.dp),
        shape = RoundedCornerShape(12.dp)
    ) {
        Column(
            modifier = Modifier.padding(16.dp),
            verticalArrangement = Arrangement.spacedBy(4.dp)
        ) {
            Text(text = title, style = MaterialTheme.typography.titleMedium, fontWeight =
FontWeight.Bold)
            HorizontalDivider(color = Color.LightGray.copy(alpha = 0.3f), modifier =
Modifier.padding(vertical = 8.dp))
            Text(text = detailLine1, style = MaterialTheme.typography.bodyMedium, color =
Color.Gray)
            Text(text = detailLine2, style = MaterialTheme.typography.bodyMedium, color =
Color.Gray)
            Spacer(modifier = Modifier.height(4.dp))
            Text(
                text = status,
                style = MaterialTheme.typography.labelLarge,
                color = statusColor,
                fontWeight = FontWeight.Bold
            )
        }
    }
}

@Composable
fun AppointmentCard(appointment: Appointment, isPast: Boolean) {
    // Xác định trạng thái và màu sắc dựa trên dữ liệu Firebase
    val (statusText, statusColor) = when {
        isPast -> "ĐÃ KẾT THÚC" to Color.Gray
        appointment.status == "CONFIRMED" -> "ĐÃ DUYỆT" to Color(0xFF388E3C) // Màu
xanh lá
        appointment.status == "PENDING" -> "CHỜ DUYỆT" to Color(0xFFFFA000) // Màu vàng
```

```
cam
        appointment.status == "CANCELLED" -> "ĐÃ HỦY" to Color.Red
        else -> appointment.status to Color.Gray
    }

    GenericInfoCard(
        title = appointment.hospitalName,
        detailLine1 = appointment.hospitalAddress,
        detailLine2 = "Thời gian: ${formatDateTime(appointment.dateTime)}",
        status = statusText,
        statusColor = statusColor
    )
}

@Composable
fun NotLoggedInView(error: String?, onLoginClick: () -> Unit) {
    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(24.dp),
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        Icon(
            imageVector = Icons.Default.Settings,
            contentDescription = null,
            modifier = Modifier.size(80.dp),
            tint = Color.Gray.copy(alpha = 0.5f)
        )
        Spacer(modifier = Modifier.height(16.dp))
        Text(
            text = "Bạn chưa đăng nhập",
            style = MaterialTheme.typography.headlineSmall,
            fontWeight = FontWeight.Bold
        )
        Text(
            text = "Vui lòng đăng nhập để quản lý hồ sơ và lịch hẹn.",
            style = MaterialTheme.typography.bodyMedium,
            color = Color.Gray,
            textAlign = TextAlign.Center
        )
        Spacer(modifier = Modifier.height(24.dp))
        Button(
            onClick = onLoginClick,
            colors = ButtonDefaults.buttonColors(containerColor = PrimaryRed),
            modifier = Modifier.fillMaxWidth()
        ) {
            Text("Đăng nhập ngay")
        }
        if (error != null) {
            Spacer(modifier = Modifier.height(8.dp))
            Text(text = "Lỗi: $error", color = Color.Red, style =
MaterialTheme.typography.labelSmall)
        }
    }
}

@Composable
private fun formatDateTime(date: Date): String {
    return SimpleDateFormat("dd/MM/yyyy 'lúc' HH:mm", Locale.getDefault()).format(date)
}

@Composable
private fun formatDate(date: Date): String {
    return SimpleDateFormat("dd/MM/yyyy", Locale.getDefault()).format(date)
}
```

feature_profile/src/main/java/com/example/feature_profile/ui/ProfileViewModel.kt

```
package com.example.feature_profile.ui

import android.net.Uri
import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import com.example.feature_emergency.domain.usecase.GetMyPledgedRequestsUseCase
import com.example.feature_map_booking.domain.usecase.GetMyAppointmentsUseCase
import com.example.feature_profile.domain.usecase.GetUserProfileUseCase
import com.example.feature_profile.domain.usecase.UpdateUserProfileUseCase
import com.example.feature_profile.domain.usecase.SignOutUseCase
import com.smartblood.core.domain.model.Appointment
import com.smartblood.core.domain.model.BloodRequest
import com.smartblood.core.storage.domain.usecase.UploadImageUseCase
import dagger.hilt.android.lifecycle.HiltViewModel
import kotlinx.coroutines.flow.*
import kotlinx.coroutines.launch
import java.util.Calendar
import javax.inject.Inject

@HiltViewModel
class ProfileViewModel @Inject constructor(
  private val getUserProfileUseCase: GetUserProfileUseCase,
  private val getMyAppointmentsUseCase: GetMyAppointmentsUseCase,
  private val getMyPledgedRequestsUseCase: GetMyPledgedRequestsUseCase,
  private val updateUserProfileUseCase: UpdateUserProfileUseCase,
  private val uploadImageUseCase: UploadImageUseCase,
  private val signOutUseCase: SignOutUseCase
) : ViewModel() {

  private val _state = MutableStateFlow(ProfileState())
  val state: StateFlow<ProfileState> = _state.asStateFlow()

  init {
    listenToProfileData()
  }

  fun onEvent(event: ProfileEvent) {
    when (event) {
      ProfileEvent.OnSignOutClicked -> signOut()
      is ProfileEvent.OnEditProfileClicked -> { /* Xử lý ở UI */ }
      is ProfileEvent.OnViewDonationHistoryClicked -> { /* Xử lý ở UI */ }
    }
  }

  private fun signOut() {
    viewModelScope.launch {
      signOutUseCase()
      _state.update { it.copy(isSignedOut = true) }
    }
  }

  private fun listenToProfileData() {
    viewModelScope.launch {
      _state.update { it.copy(isLoading = true, error = null) }
      getUserProfileUseCase().onSuccess { user ->
        _state.update { it.copy(userProfile = user) }
      }.onFailure { error ->
        _state.update { it.copy(isLoading = false, error = error.message) }
      }
    }
```

```
        viewModelScope.launch {
            val appointmentsFlow = getMyAppointmentsUseCase()
            val pledgedRequestsFlow = getMyPledgedRequestsUseCase()

            combine(appointmentsFlow, pledgedRequestsFlow) { appointmentsResult,
pledgedResult ->
                Pair(appointmentsResult, pledgedResult)
            }.collect { (appointmentsResult, pledgedResult) ->
                val errorMessage = appointmentsResult.exceptionOrNull()?.message
                    ?: pledgedResult.exceptionOrNull()?.message

                val allAppointments = appointmentsResult.getOrNull() ?: emptyList()
                val pledgedRequests = pledgedResult.getOrNull() ?: emptyList()

                val (upcoming, today, past) = classifyAppointments(allAppointments)

                _state.update {
                    it.copy(
                        isLoading = false,
                        upcomingAppointments = upcoming,
                        todayAppointments = today,
                        pastAppointments = past,
                        pledgedRequests = pledgedRequests,
                        error = errorMessage
                    )
                }
            }
        }
    }

    // --- SỬA CHÍNH Ở ĐÂY ---
    fun onAvatarChange(uri: Uri) {
        viewModelScope.launch {
            _state.update { it.copy(isUploading = true, error = null) }

            val currentProfile = _state.value.userProfile ?: run {
                _state.update { it.copy(isUploading = false, error = "Không tìm thấy thông tin người
dùng.") }
                return@launch
            }

            // Kiểm tra xem URI là file trong máy hay là link web (http/https)
        val isWebLink = uri.scheme?.startsWith("http") == true

        if (isWebLink) {
          // TRƯỜNG HỢP 1: Chọn Avatar có sẵn (Link web) -> Cập nhật trực tiếp, KHÔNG
upload
          val updatedProfile = currentProfile.copy(avatarUrl = uri.toString())
          updateUserProfileUseCase(updatedProfile).onSuccess {
            _state.update { it.copy(isUploading = false, userProfile = updatedProfile) }
          }.onFailure { error ->
            _state.update { it.copy(isUploading = false, error = "Lỗi cập nhật avatar:
${error.message}") }
          }
        } else {
          // TRƯỜNG HỢP 2: Chọn ảnh từ thư viện (File máy) -> Upload lên Cloudinary rồi
mới cập nhật
          uploadImageUseCase(uri, "avatars/${currentProfile.uid}").onSuccess {
downloadUrl ->
            val updatedProfile = currentProfile.copy(avatarUrl = downloadUrl)
            updateUserProfileUseCase(updatedProfile).onSuccess {
              _state.update { it.copy(isUploading = false, userProfile = updatedProfile) }
            }.onFailure { error ->
              _state.update { it.copy(isUploading = false, error = "Lỗi cập nhật profile:
${error.message}") }
            }
          }.onFailure { error ->
            _state.update { it.copy(isUploading = false, error = "Lỗi tải ảnh lên:
${error.message}") }
          }
        }
      }
    }

    private fun classifyAppointments(appointments: List<Appointment>):
Triple<List<Appointment>, List<Appointment>, List<Appointment>> {
      val upcoming = mutableListOf<Appointment>()
      val today = mutableListOf<Appointment>()
      val past = mutableListOf<Appointment>()

      val startOfToday = Calendar.getInstance().apply {
        set(Calendar.HOUR_OF_DAY, 0); set(Calendar.MINUTE, 0); set(Calendar.SECOND, 0);
set(Calendar.MILLISECOND, 0)
      }.time
      val endOfToday = Calendar.getInstance().apply {
         set(Calendar.HOUR_OF_DAY, 23); set(Calendar.MINUTE, 59); set(Calendar.SECOND,
59); set(Calendar.MILLISECOND, 999)
      }.time

      for (appointment in appointments) {
         when {
            appointment.dateTime.after(endOfToday) -> upcoming.add(appointment)
            appointment.dateTime.before(startOfToday) -> past.add(appointment)
            else -> today.add(appointment)
         }
      }
      upcoming.sortBy { it.dateTime }
      today.sortBy { it.dateTime }
      past.sortByDescending { it.dateTime }

      return Triple(upcoming, today, past)
   }
}
```

feature_profile/src/test/java/com/example/feature_profile/ExampleUnitTest.kt

```
package com.example.feature_profile

import org.junit.Test

import org.junit.Assert.*

/**
 * Example local unit test, which will execute on the development machine (host).
 *
 * See [testing documentation](http://d.android.com/tools/testing).
 */
class ExampleUnitTest {
   @Test
   fun addition_isCorrect() {
      assertEquals(4, 2 + 2)
   }
}
```

gradle/libs.versions.toml

```
# gradle/libs.versions.toml

[versions]
```

```
# Plugins
crashlyticsPlugin = "3.0.1"
androidGradlePlugin = "8.4.1"
kotlin = "2.0.0"
ksp = "2.0.0-1.0.21"
googleServices = "4.4.2"
mapsSecretsPlugin = "2.0.1" # <--- THÊM MỚI: Plugin cho Maps API Key
materialIconsExtended = "1.6.8"
trackasia = "2.0.2"
accompanist = "0.32.0"
secrets-gradle-plugin = "2.0.1"

# SDKs
compileSdk = "34"
minSdk = "24"
targetSdk = "34"

# THÊM CÁC THƯ VIỆN TRACKASIA
trackasia-sdk = { group = "com.track-asia", name = "trackasia-android-sdk", version.ref =
"trackasia" }
trackasia-annotation-plugin = { group = "com.track-asia", name = "trackasia-android-plugin-
annotation-v9", version.ref = "trackasia" }

# Libraries
coreKtx = "1.13.1"
activityCompose = "1.9.0"
composeBom = "2024.02.02"
composeCompiler = "1.5.14"
hilt = "2.51.1"
lifecycle = "2.7.0"
room = "2.6.1"
firebaseBom = "33.1.1"
playServicesAuth = "21.2.0"
playServicesCoroutines = "1.8.1"
coroutines = "1.8.0"
retrofit = "2.9.0"
okhttp = "4.12.0"
navigation = "2.7.7"
hiltNavigation = "1.2.0"
coil = "2.6.0"
mapsCompose = "4.3.3"        # <--- THÊM MỚI: Thư viện Maps cho Compose
playServicesMaps = "18.2.0"   # <--- THÊM MỚI: Thư viện Google Maps SDK
playServicesLocation = "21.3.0" # <--- THÊM MỚI: Thư viện Location Services
```

```
# Testing
junit = "4.13.2"
androidxJunit = "1.1.5"
espressoCore = "3.5.1"
firebaseAuth = "23.0.0"
viewbinding = "7.4.2"
transportBackendCct = "3.1.9"
firebaseFirestore = "25.0.0"


[libraries]
# AndroidX Core
androidx-core-ktx = { group = "androidx.core", name = "core-ktx", version.ref = "coreKtx" }
androidx-lifecycle-runtime-ktx = { group = "androidx.lifecycle", name = "lifecycle-runtime-
ktx", version.ref = "lifecycle" }
androidx-activity-compose = { group = "androidx.activity", name = "activity-compose",
version.ref = "activityCompose" }

accompanist-navigation-animation = { group = "com.google.accompanist", name =
"accompanist-navigation-animation", version.ref = "accompanist" }


# Jetpack Compose
androidx-compose-bom = { group = "androidx.compose", name = "compose-bom",
version.ref = "composeBom" }
androidx-compose-ui = { group = "androidx.compose.ui", name = "ui" }
androidx-compose-ui-graphics = { group = "androidx.compose.ui", name = "ui-graphics" }
androidx-compose-ui-tooling = { group = "androidx.compose.ui", name = "ui-tooling" }
androidx-compose-ui-tooling-preview = { group = "androidx.compose.ui", name = "ui-
tooling-preview" }
androidx-lifecycle-viewmodel-compose = { group = "androidx.lifecycle", name = "lifecycle-
viewmodel-compose", version.ref = "lifecycle" }
androidx-lifecycle-runtime-compose = { group = "androidx.lifecycle", name = "lifecycle-
runtime-compose", version.ref = "lifecycle" }
androidx-compose-material-icons-extended = { group = "androidx.compose.material", name
= "material-icons-extended", version.ref = "materialIconsExtended" }

# Image Loading - Coil
coil-compose = { module = "io.coil-kt:coil-compose", version.ref = "coil" }

# Navigation
androidx-navigation-compose = { group = "androidx.navigation", name = "navigation-
compose", version.ref = "navigation" }
androidx-hilt-navigation-compose = { group = "androidx.hilt", name = "hilt-navigation-
compose", version.ref = "hiltNavigation" }

# Dependency Injection - Hilt
hilt-android = { group = "com.google.dagger", name = "hilt-android", version.ref = "hilt" }
hilt-compiler = { group = "com.google.dagger", name = "hilt-compiler", version.ref = "hilt" }

# Local Database - Room
androidx-room-runtime = { group = "androidx.room", name = "room-runtime", version.ref =
"room" }
androidx-room-ktx = { group = "androidx.room", name = "room-ktx", version.ref = "room" }
androidx-room-compiler = { group = "androidx.room", name = "room-compiler", version.ref
= "room" }

# Remote - Firebase
firebase-bom = { group = "com.google.firebase", name = "firebase-bom", version.ref =
"firebaseBom" }
firebase-auth-ktx = { group = "com.google.firebase", name = "firebase-auth-ktx" }
firebase-firestore-ktx = { group = "com.google.firebase", name = "firebase-firestore-ktx" }
firebase-storage-ktx = { group = "com.google.firebase", name = "firebase-storage-ktx" }
firebase-messaging-ktx = { group = "com.google.firebase", name = "firebase-messaging-ktx"
}
firebase-crashlytics-ktx = { group = "com.google.firebase", name = "firebase-crashlytics-ktx"
}
play-services-auth = { group = "com.google.android.gms", name = "play-services-auth",
version.ref = "playServicesAuth" }

# THÊM MỚI: Google Maps & Location
maps-compose = { group = "com.google.maps.android", name = "maps-compose",
version.ref = "mapsCompose" }
play-services-maps = { group = "com.google.android.gms", name = "play-services-maps",
version.ref = "playServicesMaps" }
play-services-location = { group = "com.google.android.gms", name = "play-services-
location", version.ref = "playServicesLocation" }
trackasia-sdk = { group = "io.github.track-asia", name = "android-sdk", version.ref =
"trackasia" }
trackasia-annotation-plugin = { group = "io.github.track-asia", name = "android-plugin-
annotation-v9", version = "2.0.1" }

# Asynchronous - Coroutines
kotlinx-coroutines-core = { group = "org.jetbrains.kotlinx", name = "kotlinx-coroutines-
core", version.ref = "coroutines" }
```

```
kotlinx-coroutines-android = { group = "org.jetbrains.kotlinx", name = "kotlinx-coroutines-
android", version.ref = "coroutines" }
kotlinx-coroutines-play-services = { group = "org.jetbrains.kotlinx", name = "kotlinx-
coroutines-play-services", version.ref = "playServicesCoroutines" }

# Networking - Retrofit & OkHttp
retrofit = { group = "com.squareup.retrofit2", name = "retrofit", version.ref = "retrofit" }
converter-gson = { group = "com.squareup.retrofit2", name = "converter-gson", version.ref =
"retrofit" }
logging-interceptor = { group = "com.squareup.okhttp3", name = "logging-interceptor",
version.ref = "okhttp" }

# Testing
junit = { group = "junit", name = "junit", version.ref = "junit" }
androidx-junit = { group = "androidx.test.ext", name = "junit", version.ref = "androidxJunit" }
androidx-espresso-core = { group = "androidx.test.espresso", name = "espresso-core",
version.ref = "espressoCore" }
androidx-compose-ui-test-junit4 = { group = "androidx.compose.ui", name = "ui-test-junit4"
}
androidx-compose-ui-test-manifest = { group = "androidx.compose.ui", name = "ui-test-
manifest" }
firebase-auth = { group = "com.google.firebase", name = "firebase-auth", version.ref =
"firebaseAuth" }
androidx-viewbinding = { group = "androidx.databinding", name = "viewbinding",
version.ref = "viewbinding" }
transport-backend-cct = { group = "com.google.android.datatransport", name = "transport-
backend-cct", version.ref = "transportBackendCct" }
firebase-firestore = { group = "com.google.firebase", name = "firebase-firestore", version.ref
= "firebaseFirestore" }
androidx-compose-material3 = { group = "androidx.compose.material3", name =
"material3" }

[plugins]
# Khai báo các plugin của dự án
android-secrets-gradle-plugin = { id = "com.google.android.libraries.mapsplatform.secrets-
gradle-plugin", version.ref = "secrets-gradle-plugin" }
android-application = { id = "com.android.application", version.ref = "androidGradlePlugin"
}
android-library = { id = "com.android.library", version.ref = "androidGradlePlugin" }
kotlin-android = { id = "org.jetbrains.kotlin.android", version.ref = "kotlin" }
ksp = { id = "com.google.devtools.ksp", version.ref = "ksp" }
hilt = { id = "com.google.dagger.hilt.android", version.ref = "hilt" }
google-services = { id = "com.google.gms.google-services", version.ref = "googleServices" }
```

```
kotlin-compose-compiler = { id = "org.jetbrains.kotlin.plugin.compose", version.ref =
"kotlin" }
firebase-crashlytics = { id = "com.google.firebase.crashlytics", version.ref =
"crashlyticsPlugin" }
maps-secrets = { id = "com.google.android.libraries.mapsplatform.secrets-gradle-plugin",
version.ref = "mapsSecretsPlugin" } # <--- THÊM MỚI

[bundles]
# Nhóm các thư viện thường đi chung với nhau để gọi cho gọn
compose = ["androidx-compose-ui", "androidx-compose-ui-graphics", "androidx-compose-
ui-tooling-preview", "androidx-compose-material3"]
room = ["androidx-room-runtime", "androidx-room-ktx"]
coroutines = ["kotlinx-coroutines-core", "kotlinx-coroutines-android"]
```

gradle/wrapper/gradle-wrapper.properties

```
#Tue Oct 28 12:42:33 ICT 2025
distributionBase=GRADLE_USER_HOME
distributionPath=wrapper/dists
distributionUrl=https\://services.gradle.org/distributions/gradle-8.13-bin.zip
networkTimeout=10000
validateDistributionUrl=true
zipStoreBase=GRADLE_USER_HOME
zipStorePath=wrapper/dists
```