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](https://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](https://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  
  
}

## 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

## 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.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.smartblooddonationandroid"  
 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  
 }  
}  
  
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)  
  
 // Hilt  
 implementation(libs.hilt.android)  
 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)  
  
}

## 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/smartblooddonationandroid/ExampleInstrumentedTest.kt

package com.example.smartblooddonationandroid  
  
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: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=".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>  
 <meta-data  
 android:name="com.google.android.geo.API\_KEY"  
 android:value="${MAPS\_API\_KEY}" />  
 </application>  
  
</manifest>

## app/src/main/java/com/example/smartblooddonationandroid/MainActivity.kt

//D:\SmartBloodDonationAndroid\app\src\main\java\com\example\smartblooddonationandroid\MainActivity.kt  
package com.example.smartblooddonationandroid  
  
import android.os.Bundle  
import androidx.activity.ComponentActivity  
import androidx.activity.compose.setContent  
import androidx.activity.enableEdgeToEdge  
import androidx.compose.foundation.layout.fillMaxSize  
import androidx.compose.foundation.layout.padding  
import androidx.compose.material3.MaterialTheme  
import androidx.compose.material3.Scaffold  
import androidx.compose.material3.Surface  
import androidx.compose.material3.Text  
import androidx.compose.runtime.Composable  
import androidx.compose.ui.Modifier  
import androidx.compose.ui.tooling.preview.Preview  
import com.example.smartblooddonationandroid.ui.theme.SmartBloodDonationAndroidTheme  
import com.smartblood.donation.navigation.AppNavHost  
import dagger.hilt.android.AndroidEntryPoint  
import com.smartblood.core.ui.theme.SmartBloodTheme  
  
@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  
 ) {  
 AppNavHost()  
 }  
 }  
 }  
 }  
}  
  
@Composable  
fun Greeting(name: String, modifier: Modifier = Modifier) {  
 Text(  
 text = "Hello $name!",  
 modifier = modifier  
 )  
}  
  
@Preview(showBackground = true)  
@Composable  
fun GreetingPreview() {  
 SmartBloodDonationAndroidTheme {  
 Greeting("Android")  
 }  
}

## app/src/main/java/com/example/smartblooddonationandroid/MainApplication.kt

// D:\SmartBloodDonationAndroid\app\src\main\java\com\example\smartblooddonationandroid\MainApplication.kt  
  
package com.smartblood.donation // Hoặc package của bạn  
  
import android.app.Application  
import dagger.hilt.android.HiltAndroidApp  
  
@HiltAndroidApp  
class MainApplication : Application()

## app/src/main/java/com/example/smartblooddonationandroid/features/dashboard/DashboardScreen.kt

//D:\SmartBloodDonationAndroid\app\src\main\java\com\example\smartblooddonationandroid\features\dashboard\DashboardScreen.kt  
package com.smartblood.donation.features.dashboard  
  
import androidx.compose.foundation.layout.\*  
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.unit.dp  
import androidx.hilt.navigation.compose.hiltViewModel  
  
@Composable  
fun DashboardScreen(  
 viewModel: DashboardViewModel = hiltViewModel()  
) {  
 val state by viewModel.state.collectAsState()  
  
 if (state.isLoading) {  
 Box(modifier = Modifier.fillMaxSize(), contentAlignment = Alignment.Center) {  
 CircularProgressIndicator()  
 }  
 } else {  
 Column(  
 modifier = Modifier  
 .fillMaxSize()  
 .padding(16.dp),  
 verticalArrangement = Arrangement.spacedBy(16.dp)  
 ) {  
 // Thẻ thông tin cá nhân  
 Card(modifier = Modifier.fillMaxWidth()) {  
 Column(Modifier.padding(16.dp)) {  
 Text(text = "Chào mừng, ${state.userName}", style = MaterialTheme.typography.titleLarge)  
 Spacer(Modifier.height(8.dp))  
 Text(text = "Nhóm máu: ${state.bloodType}")  
 Text(text = state.nextDonationMessage, style = MaterialTheme.typography.bodyMedium)  
 }  
 }  
  
 // Các nút Call-to-Action  
 Button(onClick = { /\* TODO \*/ }, modifier = Modifier.fillMaxWidth()) {  
 Text("Tìm điểm hiến máu gần đây")  
 }  
 OutlinedButton(onClick = { /\* TODO \*/ }, modifier = Modifier.fillMaxWidth()) {  
 Text("Xem yêu cầu khẩn cấp")  
 }  
  
 // Danh sách yêu cầu khẩn cấp (tạm thời)  
 Text("Yêu cầu khẩn cấp gần đây:", style = MaterialTheme.typography.titleMedium)  
 Box(modifier = Modifier.fillMaxWidth().weight(1f), contentAlignment = Alignment.Center){  
 Text("Chưa có yêu cầu nào.")  
 }  
 }  
 }  
}

## app/src/main/java/com/example/smartblooddonationandroid/features/dashboard/DashboardViewModel.kt

//D:\SmartBloodDonationAndroid\app\src\main\java\com\example\smartblooddonationandroid\features\dashboard\DashboardViewModel.kt  
package com.smartblood.donation.features.dashboard  
  
import androidx.lifecycle.ViewModel  
import androidx.lifecycle.viewModelScope  
import com.smartblood.profile.domain.usecase.CalculateNextDonationDateUseCase  
import com.smartblood.profile.domain.usecase.GetUserProfileUseCase  
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  
  
// Tạm thời định nghĩa State ở đây cho gọn  
data class DashboardState(  
 val isLoading: Boolean = true,  
 val userName: String = "",  
 val bloodType: String = "N/A",  
 val nextDonationMessage: String = ""  
)  
  
@HiltViewModel  
class DashboardViewModel @Inject constructor(  
 private val getUserProfileUseCase: GetUserProfileUseCase,  
 private val calculateNextDonationDateUseCase: CalculateNextDonationDateUseCase  
) : ViewModel() {  
  
 private val \_state = MutableStateFlow(DashboardState())  
 val state = \_state.asStateFlow()  
  
 init {  
 loadDashboardData()  
 }  
  
 private fun loadDashboardData() {  
 viewModelScope.launch {  
 \_state.update { it.copy(isLoading = true) }  
 getUserProfileUseCase().onSuccess { userProfile ->  
 \_state.update {  
 it.copy(  
 isLoading = false,  
 userName = userProfile.fullName,  
 bloodType = userProfile.bloodType ?: "N/A",  
 nextDonationMessage = calculateNextDonationDateUseCase(userProfile)  
 )  
 }  
 }.onFailure {  
 \_state.update { it.copy(isLoading = false, userName = "Không tải được dữ liệu") }  
 }  
 }  
 }  
}

## app/src/main/java/com/example/smartblooddonationandroid/navigation/AppNavHost.kt

//app/src/main/java/com/smartblood/donation/navigation/AppNavHost.kt  
package com.smartblood.donation.navigation  
  
import androidx.compose.foundation.layout.Box  
import androidx.compose.foundation.layout.fillMaxSize  
import androidx.compose.material3.Text  
import com.smartblood.auth.ui.register.RegisterScreen  
import androidx.compose.runtime.Composable  
import androidx.compose.ui.Alignment  
import androidx.compose.ui.Modifier  
import androidx.navigation.NavGraph.Companion.findStartDestination  
import androidx.navigation.compose.NavHost  
import androidx.navigation.compose.composable  
import androidx.navigation.compose.navigation  
import androidx.navigation.compose.rememberNavController  
import com.smartblood.auth.navigation.authGraph  
import com.smartblood.auth.ui.login.LoginScreen  
import com.smartblood.auth.ui.splash.SplashScreen  
import com.smartblood.donation.ui.MainScreen  
  
@Composable  
fun AppNavHost() {  
 val navController = rememberNavController()  
 NavHost(  
 navController = navController,  
 startDestination = Screen.SPLASH  
 ) {  
 composable(Screen.SPLASH) {  
 SplashScreen(  
 navigateToLogin = {  
 // Điều hướng đến đồ thị xác thực  
 navController.navigate(Graph.AUTHENTICATION) {  
 // Xóa SplashScreen khỏi back stack  
 popUpTo(Screen.SPLASH) { inclusive = true }  
 }  
 },  
 navigateToDashboard = {  
 // Điều hướng đến đồ thị chính  
 navController.navigate(Graph.MAIN) {  
 // Xóa SplashScreen khỏi back stack  
 popUpTo(Screen.SPLASH) { inclusive = true }  
 }  
 }  
 )  
 }  
  
  
 // --- ĐÂY LÀ NƠI KẾT NỐI ---  
 authGraph(navController)  
  
 composable(Graph.MAIN) {  
 MainScreen()  
 }  
 }  
}

## app/src/main/java/com/example/smartblooddonationandroid/navigation/BottomNavItem.kt

//D:\SmartBloodDonationAndroid\app\src\main\java\com\example\smartblooddonationandroid\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/example/smartblooddonationandroid/navigation/Screen.kt

//app/src/main/java/com/smartblood/donation/navigation/Screen.kt  
package com.smartblood.donation.navigation  
  
import com.smartblood.auth.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/example/smartblooddonationandroid/ui/theme/Color.kt

package com.example.smartblooddonationandroid.ui.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/example/smartblooddonationandroid/ui/theme/MainScreen.kt

//D:\SmartBloodDonationAndroid\app\src\main\java\com\example\smartblooddonationandroid\ui\theme\MainScreen.kt  
package com.smartblood.donation.ui  
  
import androidx.compose.foundation.layout.padding  
import androidx.compose.material3.\*  
import androidx.compose.runtime.Composable  
import androidx.compose.runtime.getValue  
import androidx.compose.ui.Modifier  
import androidx.navigation.NavDestination.Companion.hierarchy  
import androidx.navigation.NavGraph.Companion.findStartDestination  
import androidx.navigation.NavType  
import androidx.navigation.compose.NavHost  
import androidx.navigation.compose.composable  
import androidx.navigation.compose.currentBackStackEntryAsState  
import androidx.navigation.compose.rememberNavController  
import androidx.navigation.navArgument  
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.smartblood.donation.features.dashboard.DashboardScreen  
import com.smartblood.donation.navigation.BottomNavItem  
import com.smartblood.donation.navigation.Screen  
import com.smartblood.profile.ui.ProfileScreen  
  
@Composable  
fun MainScreen() {  
 // NavController này chỉ quản lý việc điều hướng BÊN TRONG MainScreen  
 // (giữa Dashboard, Map, Profile,...)  
 val navController = rememberNavController()  
  
 // Danh sách các mục sẽ hiển thị trên thanh điều hướng  
 val navItems = listOf(  
 BottomNavItem.Dashboard,  
 BottomNavItem.Map,  
 BottomNavItem.Profile,  
 )  
  
 Scaffold(  
 bottomBar = {  
 NavigationBar {  
 val navBackStackEntry by navController.currentBackStackEntryAsState()  
 val currentDestination = navBackStackEntry?.destination  
  
 // Lặp qua danh sách các mục để tạo các NavigationBarItem  
 navItems.forEach { screen ->  
 NavigationBarItem(  
 icon = { Icon(screen.icon, contentDescription = screen.title) },  
 label = { Text(screen.title) },  
 // Kiểm tra xem route hiện tại có khớp với mục này không để highlight nó  
 selected = currentDestination?.hierarchy?.any { it.route == screen.route } == true,  
 onClick = {  
 navController.navigate(screen.route) {  
 // Pop up về màn hình bắt đầu của graph để tránh chồng chất back stack  
 popUpTo(navController.graph.findStartDestination().id) {  
 saveState = true  
 }  
 // Tránh tạo nhiều bản sao của cùng một màn hình  
 launchSingleTop = true  
 // Khôi phục lại trạng thái khi chọn lại một mục đã chọn trước đó  
 restoreState = true  
 }  
 }  
 )  
 }  
 }  
 }  
 ) { innerPadding ->  
 // NavHost này sẽ là nơi hiển thị nội dung của màn hình được chọn  
 NavHost(  
 navController = navController,  
 startDestination = BottomNavItem.Dashboard.route, // Màn hình bắt đầu là Trang chủ  
 modifier = Modifier.padding(innerPadding)  
 ) {  
 // Định nghĩa Composable cho từng route  
 composable(BottomNavItem.Dashboard.route) {  
 DashboardScreen()  
 }  
 composable(BottomNavItem.Map.route) {  
 MapScreen(  
 onNavigateToLocationDetail = { hospitalId ->  
 navController.navigate("location\_detail/$hospitalId")  
 }  
 )  
 }  
 composable(BottomNavItem.Profile.route) {  
 // Sử dụng ProfileScreen đã tạo từ feature\_profile  
 ProfileScreen(  
 onNavigateToEditProfile = {  
  
 },  
 onNavigateToDonationHistory = {  
  
 }  
 )  
 }  
 composable(  
 route = "location\_detail/{hospitalId}",  
 arguments = listOf(navArgument("hospitalId") { type = NavType.StringType })  
 ) { backStackEntry ->  
 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 = {  
 // Quay về màn hình bản đồ sau khi đặt lịch thành công  
 navController.popBackStack(BottomNavItem.Map.route, inclusive = false)  
 },  
 onNavigateBack = { navController.popBackStack() }  
 )  
 }  
 composable(route = Screen.EDIT\_PROFILE) {  
 EditProfileScreen(  
 onNavigateBack = { navController.popBackStack() }  
 )  
 }  
 composable(route = Screen.DONATION\_HISTORY) {  
 DonationHistoryScreen(  
 onNavigateBack = { navController.popBackStack() }  
 )  
 }  
 }  
 }  
}

## app/src/main/java/com/example/smartblooddonationandroid/ui/theme/Theme.kt

package com.example.smartblooddonationandroid.ui.theme  
  
import android.app.Activity  
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/example/smartblooddonationandroid/ui/theme/Type.kt

package com.example.smartblooddonationandroid.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  
  
// 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"  
 xmlns:aapt="http://schemas.android.com/aapt"  
 android:width="108dp"  
 android:height="108dp"  
 android:viewportWidth="108"  
 android:viewportHeight="108">  
 <path android:pathData="M31,63.928c0,0 6.4,-11 12.1,-13.1c7.2,-2.6 26,-1.4 26,-1.4l38.1,38.1L107,108.928l-32,-1L31,63.928z">  
 <aapt:attr name="android:fillColor">  
 <gradient  
 android:endX="85.84757"  
 android:endY="92.4963"  
 android:startX="42.9492"  
 android:startY="49.59793"  
 android:type="linear">  
 <item  
 android:color="#44000000"  
 android:offset="0.0" />  
 <item  
 android:color="#00000000"  
 android:offset="1.0" />  
 </gradient>  
 </aapt:attr>  
 </path>  
 <path  
 android:fillColor="#FFFFFF"  
 android:fillType="nonZero"  
 android:pathData="M65.3,45.828l3.8,-6.6c0.2,-0.4 0.1,-0.9 -0.3,-1.1c-0.4,-0.2 -0.9,-0.1 -1.1,0.3l-3.9,6.7c-6.3,-2.8 -13.4,-2.8 -19.7,0l-3.9,-6.7c-0.2,-0.4 -0.7,-0.5 -1.1,-0.3C38.8,38.328 38.7,38.828 38.9,39.228l3.8,6.6C36.2,49.428 31.7,56.028 31,63.928h46C76.3,56.028 71.8,49.428 65.3,45.828zM43.4,57.328c-0.8,0 -1.5,-0.5 -1.8,-1.2c-0.3,-0.7 -0.1,-1.5 0.4,-2.1c0.5,-0.5 1.4,-0.7 2.1,-0.4c0.7,0.3 1.2,1 1.2,1.8C45.3,56.528 44.5,57.328 43.4,57.328L43.4,57.328zM64.6,57.328c-0.8,0 -1.5,-0.5 -1.8,-1.2s-0.1,-1.5 0.4,-2.1c0.5,-0.5 1.4,-0.7 2.1,-0.4c0.7,0.3 1.2,1 1.2,1.8C66.5,56.528 65.6,57.328 64.6,57.328L64.6,57.328z"  
 android:strokeWidth="1"  
 android:strokeColor="#00000000" />  
</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="@drawable/ic\_launcher\_background" />  
 <foreground android:drawable="@drawable/ic\_launcher\_foreground" />  
 <monochrome 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="@drawable/ic\_launcher\_background" />  
 <foreground android:drawable="@drawable/ic\_launcher\_foreground" />  
 <monochrome 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/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/smartblooddonationandroid/ExampleUnitTest.kt

package com.example.smartblooddonationandroid  
  
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)  
  
}  
  
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  
 }  
  
}  
  
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)  
}

## 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/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)  
  
 // 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.smartblood.auth.data.repository  
  
import com.google.firebase.auth.FirebaseAuth  
import com.google.firebase.firestore.FirebaseFirestore  
import com.smartblood.auth.domain.model.User  
import com.smartblood.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.smartblood.auth.data.repository.AuthRepositoryImpl  
import com.smartblood.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.smartblood.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.smartblood.auth.domain.repository  
  
// Sử dụng Result của Kotlin để đóng gói thành công hoặc lỗi một cách an toàn  
import com.smartblood.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.smartblood.auth.domain.usecase  
  
import com.smartblood.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.smartblood.auth.domain.usecase  
  
import com.smartblood.auth.domain.model.User  
import com.smartblood.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.smartblood.auth.domain.usecase  
  
import com.smartblood.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.smartblood.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

// feature\_auth/src/main/java/com/smartblood/auth/ui/login/LoginScreen.kt  
  
package com.smartblood.auth.ui.login  
  
import androidx.compose.foundation.layout.\*  
import androidx.compose.material3.\*  
import androidx.compose.runtime.\*  
import androidx.compose.ui.Alignment  
import androidx.compose.ui.Modifier  
import androidx.compose.ui.text.input.PasswordVisualTransformation  
import androidx.compose.ui.unit.dp  
import androidx.hilt.navigation.compose.hiltViewModel  
import com.smartblood.core.ui.components.PrimaryButton  
  
@Composable  
fun LoginScreen(  
 viewModel: LoginViewModel = hiltViewModel(),  
 navigateToDashboard: () -> Unit,  
 navigateToRegister: () -> Unit,  
) {  
 val state by viewModel.state.collectAsState()  
  
 // Điều hướng khi đăng nhập thành công  
 LaunchedEffect(state.loginSuccess) {  
 if (state.loginSuccess) {  
 navigateToDashboard()  
 }  
 }  
  
 // Hiển thị thông báo lỗi  
 if (state.error != null) {  
 AlertDialog(  
 onDismissRequest = { viewModel.onEvent(LoginEvent.OnErrorDismissed) },  
 title = { Text("Login Failed") },  
 text = { Text(state.error!!) },  
 confirmButton = {  
 TextButton(onClick = { viewModel.onEvent(LoginEvent.OnErrorDismissed) }) {  
 Text("OK")  
 }  
 }  
 )  
 }  
  
 Box(modifier = Modifier.fillMaxSize(), contentAlignment = Alignment.Center) {  
 if (state.isLoading) {  
 CircularProgressIndicator()  
 } else {  
 Column(  
 modifier = Modifier  
 .fillMaxWidth()  
 .padding(horizontal = 32.dp),  
 horizontalAlignment = Alignment.CenterHorizontally,  
 verticalArrangement = Arrangement.spacedBy(16.dp)  
 ) {  
 Text("Welcome Back!", style = MaterialTheme.typography.headlineMedium)  
  
 OutlinedTextField(  
 value = state.email,  
 onValueChange = { viewModel.onEvent(LoginEvent.OnEmailChanged(it)) },  
 label = { Text("Email") },  
 modifier = Modifier.fillMaxWidth()  
 )  
  
 OutlinedTextField(  
 value = state.password,  
 onValueChange = { viewModel.onEvent(LoginEvent.OnPasswordChanged(it)) },  
 label = { Text("Password") },  
 visualTransformation = PasswordVisualTransformation(),  
 modifier = Modifier.fillMaxWidth()  
 )  
  
 PrimaryButton(  
 text = "Login",  
 onClick = { viewModel.onEvent(LoginEvent.OnLoginClicked) }  
 )  
  
 TextButton(onClick = navigateToRegister) {  
 Text("Don't have an account? Sign Up")  
 }  
 }  
 }  
 }  
}

## feature\_auth/src/main/java/com/example/feature\_auth/ui/login/LoginViewModel.kt

// feature\_auth/src/main/java/com/smartblood/auth/ui/login/LoginViewModel.kt  
  
package com.smartblood.auth.ui.login  
  
import androidx.lifecycle.ViewModel  
import androidx.lifecycle.viewModelScope  
import com.smartblood.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) }  
 val result = loginUseCase(state.value.email, state.value.password)  
 result.onSuccess { user ->  
 \_state.update { it.copy(isLoading = false, loginSuccess = true) }  
 }.onFailure { exception ->  
 \_state.update {  
 it.copy(  
 isLoading = false,  
 error = exception.message ?: "Đã xảy ra lỗi không xác định."  
 )  
 }  
 }  
 }  
 }  
}

## feature\_auth/src/main/java/com/example/feature\_auth/ui/navigation/AuthNavigation.kt

//D:\SmartBloodDonationAndroid\feature\_auth\src\main\java\com\example\feature\_auth\ui\navigation\AuthNavigation.kt  
package com.smartblood.auth.navigation  
  
import androidx.navigation.NavGraphBuilder  
import androidx.navigation.NavHostController  
import androidx.navigation.compose.composable  
import androidx.navigation.navigation  
import com.smartblood.auth.ui.login.LoginScreen  
import com.smartblood.auth.ui.register.RegisterScreen  
  
// Định nghĩa một route duy nhất cho cả đồ thị này  
// Module :app sẽ dùng route này để gọi vào  
const val AUTH\_GRAPH\_ROUTE = "auth\_graph"  
  
/\*\*  
 \* Extension function để đóng gói toàn bộ luồng navigation của feature\_auth.  
 \* @param onNavigateToMainGraph Callback được gọi khi xác thực thành công để điều hướng  
 \* ra khỏi luồng này và vào luồng chính của app.  
 \*/  
fun NavGraphBuilder.authGraph(navController: NavHostController) {  
 // Sử dụng hàm navigation() để tạo một đồ thị con (nested graph)  
 navigation(  
 // Màn hình bắt đầu của luồng này  
 startDestination = AuthScreen.Login.route,  
 // Route của cả đồ thị con này  
 route = AUTH\_GRAPH\_ROUTE  
 ) {  
 // Định nghĩa màn hình Login  
 composable(route = AuthScreen.Login.route) {  
 LoginScreen(  
 navigateToDashboard = {  
 // Khi đăng nhập thành công, điều hướng ra khỏi luồng auth  
 // và xóa luồng auth khỏi back stack  
 navController.navigate("main\_graph\_route") { // Route này sẽ được định nghĩa ở :app  
 popUpTo(AUTH\_GRAPH\_ROUTE) {  
 inclusive = true // Xóa cả auth\_graph  
 }  
 }  
 },  
 navigateToRegister = {  
 navController.navigate(AuthScreen.Register.route)  
 }  
 )  
 }  
  
 // Định nghĩa màn hình Register  
 composable(route = AuthScreen.Register.route) {  
 RegisterScreen(  
 navigateToDashboard = {  
 // Tương tự, khi đăng ký thành công, điều hướng ra khỏi luồng auth  
 navController.navigate("main\_graph\_route") {  
 popUpTo(AUTH\_GRAPH\_ROUTE) {  
 inclusive = true  
 }  
 }  
 },  
 navigateBack = {  
 navController.popBackStack() // Quay lại màn hình trước đó (LoginScreen)  
 }  
 )  
 }  
  
 // Thêm các composable cho các màn hình khác như FaceAuth... tại đây  
 }  
}

## 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.smartblood.auth.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.smartblood.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

// feature\_auth/src/main/java/com/smartblood/auth/ui/register/RegisterScreen.kt  
  
package com.smartblood.auth.ui.register  
  
import androidx.compose.foundation.layout.\*  
import androidx.compose.material3.\*  
import androidx.compose.runtime.\*  
import androidx.compose.ui.Alignment  
import androidx.compose.ui.Modifier  
import androidx.compose.ui.text.input.PasswordVisualTransformation  
import androidx.compose.ui.unit.dp  
import androidx.hilt.navigation.compose.hiltViewModel  
import com.smartblood.core.ui.components.PrimaryButton  
  
@Composable  
fun RegisterScreen(  
 viewModel: RegisterViewModel = hiltViewModel(),  
 navigateToDashboard: () -> Unit,  
 navigateBack: () -> Unit,  
) {  
 val state by viewModel.state.collectAsState()  
  
 // Điều hướng khi đăng ký thành công  
 LaunchedEffect(state.registrationSuccess) {  
 if (state.registrationSuccess) {  
 navigateToDashboard()  
 }  
 }  
  
 // Hiển thị thông báo lỗi  
 if (state.error != null) {  
 AlertDialog(  
 onDismissRequest = { viewModel.onEvent(RegisterEvent.OnErrorDismissed) },  
 title = { Text("Registration Failed") },  
 text = { Text(state.error!!) },  
 confirmButton = {  
 TextButton(onClick = { viewModel.onEvent(RegisterEvent.OnErrorDismissed) }) {  
 Text("OK")  
 }  
 }  
 )  
 }  
  
 Box(modifier = Modifier.fillMaxSize(), contentAlignment = Alignment.Center) {  
 if (state.isLoading) {  
 CircularProgressIndicator()  
 } else {  
 Column(  
 modifier = Modifier  
 .fillMaxWidth()  
 .padding(horizontal = 32.dp),  
 horizontalAlignment = Alignment.CenterHorizontally,  
 verticalArrangement = Arrangement.spacedBy(16.dp)  
 ) {  
 Text("Create an Account", style = MaterialTheme.typography.headlineMedium)  
  
 OutlinedTextField(  
 value = state.fullName,  
 onValueChange = { viewModel.onEvent(RegisterEvent.OnFullNameChanged(it)) },  
 label = { Text("Full Name") },  
 modifier = Modifier.fillMaxWidth()  
 )  
  
 OutlinedTextField(  
 value = state.email,  
 onValueChange = { viewModel.onEvent(RegisterEvent.OnEmailChanged(it)) },  
 label = { Text("Email") },  
 modifier = Modifier.fillMaxWidth()  
 )  
  
 OutlinedTextField(  
 value = state.password,  
 onValueChange = { viewModel.onEvent(RegisterEvent.OnPasswordChanged(it)) },  
 label = { Text("Password") },  
 visualTransformation = PasswordVisualTransformation(),  
 modifier = Modifier.fillMaxWidth()  
 )  
  
 PrimaryButton(  
 text = "Sign Up",  
 onClick = { viewModel.onEvent(RegisterEvent.OnRegisterClicked) }  
 )  
  
 TextButton(onClick = navigateBack) {  
 Text("Already have an account? Log In")  
 }  
 }  
 }  
 }  
}

## feature\_auth/src/main/java/com/example/feature\_auth/ui/register/RegisterViewModel.kt

// feature\_auth/src/main/java/com/smartblood/auth/ui/register/RegisterViewModel.kt  
  
package com.smartblood.auth.ui.register  
  
import androidx.lifecycle.ViewModel  
import androidx.lifecycle.viewModelScope  
import com.smartblood.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) }  
 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 ->  
 \_state.update {  
 it.copy(  
 isLoading = false,  
 error = exception.message ?: "An unknown error occurred."  
 )  
 }  
 }  
 }  
 }  
}

## feature\_auth/src/main/java/com/example/feature\_auth/ui/splash/SplashScreen.kt

// feature\_auth/src/main/java/com/smartblood/auth/ui/splash/SplashScreen.kt  
  
package com.smartblood.auth.ui.splash  
  
import androidx.compose.foundation.background  
import androidx.compose.foundation.layout.Box  
import androidx.compose.foundation.layout.fillMaxSize  
import androidx.compose.material3.MaterialTheme  
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.ui.Alignment  
import androidx.compose.ui.Modifier  
import androidx.hilt.navigation.compose.hiltViewModel  
  
@Composable  
fun SplashScreen(  
 viewModel: SplashViewModel = hiltViewModel(),  
 navigateToLogin: () -> Unit,  
 navigateToDashboard: () -> Unit  
) {  
 val isAuthenticated by viewModel.isAuthenticated.collectAsState()  
  
 // LaunchedEffect sẽ được kích hoạt khi `isAuthenticated` thay đổi giá trị từ null  
 LaunchedEffect(isAuthenticated) {  
 when (isAuthenticated) {  
 true -> navigateToDashboard()  
 false -> navigateToLogin()  
 null -> { /\* Do nothing, wait for the check to complete \*/ }  
 }  
 }  
  
 // Giao diện đơn giản của Splash Screen  
 Box(  
 modifier = Modifier  
 .fillMaxSize()  
 .background(MaterialTheme.colorScheme.primary),  
 contentAlignment = Alignment.Center  
 ) {  
 Text(  
 text = "Smart Blood Donation",  
 style = MaterialTheme.typography.displayLarge,  
 color = MaterialTheme.colorScheme.onPrimary  
 )  
 }  
}

## 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.smartblood.auth.ui.splash  
  
import androidx.lifecycle.ViewModel  
import androidx.lifecycle.viewModelScope  
import com.smartblood.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)  
  
}  
  
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 {  
 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\_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/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)  
  
 // 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).  
 \*/  
@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 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/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.model.Appointment  
import com.example.feature\_map\_booking.domain.model.Hospital  
import com.example.feature\_map\_booking.domain.model.TimeSlot  
import com.example.feature\_map\_booking.domain.repository.MapBookingRepository  
import com.google.firebase.auth.FirebaseAuth  
import com.google.firebase.firestore.FirebaseFirestore  
import com.google.firebase.firestore.ktx.toObject  
  
import kotlinx.coroutines.tasks.await  
import java.util.Calendar  
import java.util.Date  
import javax.inject.Inject  
import kotlin.Result  
  
class MapBookingRepositoryImpl @Inject constructor(  
 private val firestore: FirebaseFirestore,  
 private val auth: FirebaseAuth  
) : MapBookingRepository {  
  
 // Lưu ý: Việc tìm kiếm theo vị trí cần cấu hình Geo-query phức tạp hơn  
 // hoặc sử dụng thư viện như GeoFirestore.  
 // Để đơn giản, ở đây chúng ta sẽ lấy tất cả và lọc ở client-side (không hiệu quả cho quy mô lớn).  
 override suspend fun getNearbyHospitals(lat: Double, lng: Double, radiusKm: Double): Result<List<Hospital>> {  
 return try {  
 val result = firestore.collection("hospitals").get().await()  
 val hospitals = result.documents.mapNotNull { it.toObject<Hospital>()?.copy(id = it.id) }  
 // TODO: Triển khai logic lọc theo khoảng cách nếu cần  
 Result.success(hospitals)  
 } catch (e: Exception) {  
 Result.failure(e)  
 }  
 }  
  
 override suspend fun getHospitalDetails(hospitalId: String): Result<Hospital> {  
 return try {  
 val document = firestore.collection("hospitals").document(hospitalId).get().await()  
 val hospital = document.toObject<Hospital>()?.copy(id = document.id)  
 if (hospital != null) {  
 Result.success(hospital)  
 } else {  
 Result.failure(Exception("Hospital not found."))  
 }  
 } catch (e: Exception) {  
 Result.failure(e)  
 }  
 }  
  
 // Giả lập logic lấy khung giờ. Thực tế sẽ truy vấn collection appointments.  
 override suspend fun getAvailableSlots(hospitalId: String, date: Date): Result<List<TimeSlot>> {  
 return try {  
 val calendar = Calendar.getInstance()  
 calendar.time = date  
 calendar.set(Calendar.HOUR\_OF\_DAY, 0)  
 calendar.set(Calendar.MINUTE, 0)  
 val startOfDay = calendar.time  
 calendar.set(Calendar.HOUR\_OF\_DAY, 23)  
 calendar.set(Calendar.MINUTE, 59)  
 val endOfDay = calendar.time  
  
 // Lấy các lịch hẹn đã có trong ngày đó  
 val appointmentsSnapshot = firestore.collection("appointments")  
 .whereEqualTo("hospitalId", hospitalId)  
 .whereGreaterThanOrEqualTo("dateTime", startOfDay)  
 .whereLessThanOrEqualTo("dateTime", endOfDay)  
 .get()  
 .await()  
  
 val bookedTimes = appointmentsSnapshot.documents.mapNotNull {  
 val appointment = it.toObject<Appointment>()  
 appointment?.dateTime  
 }  
  
 // Tạo danh sách các khung giờ mặc định (ví dụ từ 8h -> 17h)  
 val allSlots = mutableListOf<TimeSlot>()  
 val slotCalendar = Calendar.getInstance().apply { time = date }  
 for (hour in 8..16) { // 8 AM to 4 PM  
 slotCalendar.set(Calendar.HOUR\_OF\_DAY, hour)  
 slotCalendar.set(Calendar.MINUTE, 0)  
 val timeString = String.format("%02d:00", hour)  
  
 val isBooked = bookedTimes.any { bookedDate ->  
 val bookedCalendar = Calendar.getInstance().apply { time = bookedDate }  
 bookedCalendar.get(Calendar.HOUR\_OF\_DAY) == 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): Result<Unit> {  
 val currentUser = auth.currentUser  
 if (currentUser == null) {  
 return Result.failure(Exception("User not authenticated."))  
 }  
 return try {  
 // Lấy thông tin bệnh viện để lưu vào lịch hẹn  
 val hospital = getHospitalDetails(hospitalId).getOrThrow()  
  
 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,  
 status = "CONFIRMED"  
 )  
 // TODO: Triển khai transaction để đảm bảo không có 2 người đặt cùng lúc  
 firestore.collection("appointments").document(appointmentId).set(appointment).await()  
 Result.success(Unit)  
 } catch (e: Exception) {  
 Result.failure(e)  
 }  
 }  
}

## 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/model/Appointment.kt

package com.example.feature\_map\_booking.domain.model  
  
// feature\_map\_booking/src/main/java/com/smartblood/mapbooking/domain/model/Appointment.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  
)

## feature\_map\_booking/src/main/java/com/example/feature\_map\_booking/domain/model/Hospital.kt

// feature\_map\_booking/src/main/java/com/smartblood/mapbooking/domain/model/Hospital.kt  
package com.example.feature\_map\_booking.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()  
)

## feature\_map\_booking/src/main/java/com/example/feature\_map\_booking/domain/model/TimeSlot.kt

// feature\_map\_booking/src/main/java/com/smartblood/mapbooking/domain/model/TimeSlot.kt  
package com.example.feature\_map\_booking.domain.model  
data class TimeSlot(  
 val time: String = "", // e.g., "09:00"  
 val isAvailable: Boolean = true  
)

## 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.example.feature\_map\_booking.domain.model.Hospital  
import com.example.feature\_map\_booking.domain.model.TimeSlot  
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  
 ): Result<Unit>  
}

## 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.example.feature\_map\_booking.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 timeSlots: List<TimeSlot> = emptyList(),  
 val error: String? = null,  
 val bookingSuccess: Boolean = false  
)  
  
sealed class BookingEvent {  
 data class OnDateSelected(val date: Date) : 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  
  
  
// feature\_map\_booking/src/main/java/com/smartblood/mapbooking/ui/booking/BookingScreen.kt  
  
  
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.Alignment  
import androidx.compose.ui.Modifier  
import androidx.compose.ui.text.font.FontWeight  
import androidx.compose.ui.unit.dp  
import androidx.hilt.navigation.compose.hiltViewModel  
import com.example.feature\_map\_booking.domain.model.TimeSlot  
import java.text.SimpleDateFormat  
import java.util.Calendar  
import java.util.Date  
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() }  
  
 LaunchedEffect(state.bookingSuccess) {  
 if (state.bookingSuccess) {  
 onBookingSuccess()  
 }  
 }  
  
 LaunchedEffect(state.error) {  
 state.error?.let {  
 snackbarHostState.showSnackbar(it)  
 }  
 }  
  
 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)  
 ) {  
 Text(  
 text = state.hospitalName,  
 style = MaterialTheme.typography.titleLarge,  
 fontWeight = FontWeight.Bold  
 )  
 Spacer(modifier = Modifier.height(16.dp))  
  
 // Calendar Picker (Basic implementation)  
 // For a real app, use a proper library or build a more complex one  
 val dateFormat = SimpleDateFormat("dd/MM/yyyy", Locale.getDefault())  
 Text("Chọn ngày: ${dateFormat.format(state.selectedDate)}")  
 Spacer(modifier = Modifier.height(8.dp))  
 Button(onClick = {  
 // TODO: Show a DatePickerDialog  
 // For now, we just fetch for the current date  
 }) {  
 Text("Đổi ngày")  
 }  
  
  
 Spacer(modifier = Modifier.height(24.dp))  
 Text("Chọn khung giờ:", style = MaterialTheme.typography.titleMedium)  
 Spacer(modifier = Modifier.height(8.dp))  
  
 if (state.isLoadingSlots) {  
 CircularProgressIndicator()  
 } else {  
 var selectedTime by remember { mutableStateOf<String?>(null) }  
  
 LazyVerticalGrid(  
 columns = GridCells.Adaptive(minSize = 100.dp),  
 verticalArrangement = Arrangement.spacedBy(8.dp),  
 horizontalArrangement = Arrangement.spacedBy(8.dp)  
 ) {  
 items(state.timeSlots) { slot ->  
 TimeSlotItem(  
 timeSlot = slot,  
 isSelected = slot.time == selectedTime,  
 onSelect = {  
 selectedTime = it  
 viewModel.onEvent(BookingEvent.OnSlotSelected(it))  
 }  
 )  
 }  
 }  
 }  
  
 Spacer(modifier = Modifier.weight(1f))  
  
 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")  
 }  
 }  
 }  
 }  
}  
  
@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/BookingViewModel.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 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.OnConfirmBooking -> {  
 confirmBooking()  
 }  
 }  
 }  
  
 private fun fetchSlotsForDate(date: Date) {  
 viewModelScope.launch {  
 \_state.update { it.copy(isLoadingSlots = true) }  
 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() {  
 var time = selectedTime ?: return // Cần thông báo lỗi cho người dùng  
 val calendar = Calendar.getInstance().apply {  
 time = \_state.value.selectedDate.toString()  
 val (hour, minute) = time.split(":").map { it.toInt() }  
 set(Calendar.HOUR\_OF\_DAY, hour)  
 set(Calendar.MINUTE, minute)  
 set(Calendar.SECOND, 0)  
 }  
 val finalDateTime = calendar.time  
  
 viewModelScope.launch {  
 \_state.update { it.copy(isBooking = true) }  
 bookAppointmentUseCase(hospitalId, finalDateTime)  
 .onSuccess {  
 \_state.update { it.copy(isBooking = false, bookingSuccess = true) }  
 }  
 .onFailure { error ->  
 \_state.update { it.copy(isBooking = false, error = error.message) }  
 }  
 }  
 }  
}

## 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.example.feature\_map\_booking.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.layout.\*  
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.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.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.example.feature\_map\_booking.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),  
 verticalArrangement = Arrangement.spacedBy(16.dp)  
 ) {  
 Text(  
 text = hospital.name,  
 style = MaterialTheme.typography.headlineMedium,  
 fontWeight = FontWeight.Bold  
 )  
  
 Divider()  
  
 InfoRow(icon = Icons.Filled.Business, text = hospital.address)  
 InfoRow(icon = Icons.Filled.Phone, text = hospital.phone)  
 InfoRow(icon = Icons.Filled.Schedule, text = hospital.workingHours)  
 InfoRow(  
 icon = Icons.Default.Bloodtype,  
 text = "Nhóm máu đang cần: ${hospital.availableBloodTypes.joinToString(", ")}"  
 )  
  
 Spacer(modifier = Modifier.weight(1f))  
  
 Button(  
 onClick = onBookAppointment,  
 modifier = Modifier  
 .fillMaxWidth()  
 .height(50.dp)  
 ) {  
 Text(text = "Đặt lịch hiến máu", fontSize = 16.sp)  
 }  
 }  
}  
  
@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()  
  
 private val hospitalId: String = checkNotNull(savedStateHandle["hospitalId"])  
  
 init {  
 fetchHospitalDetails()  
 }  
  
 private fun fetchHospitalDetails() {  
 viewModelScope.launch {  
 \_state.update { it.copy(isLoading = true) }  
 getHospitalDetailsUseCase(hospitalId)  
 .onSuccess { hospital ->  
 \_state.update { it.copy(isLoading = false, hospital = hospital) }  
 }  
 .onFailure { error ->  
 \_state.update { it.copy(isLoading = false, error = error.message) }  
 }  
 }  
 }  
}

## 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.example.feature\_map\_booking.domain.model.Hospital  
import com.google.android.gms.maps.model.LatLng  
  
data class MapState(  
 val isLoading: Boolean = true,  
 val lastKnownLocation: LatLng? = null,  
 val hospitals: List<Hospital> = emptyList(),  
 val error: String? = null  
)  
  
sealed class 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/smartblood/mapbooking/ui/map/MapScreen.kt  
  
package com.example.feature\_map\_booking.domain.ui.map  
  
import android.Manifest  
import android.annotation.SuppressLint  
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.material3.CircularProgressIndicator  
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.platform.LocalContext  
import androidx.hilt.navigation.compose.hiltViewModel  
import com.example.feature\_map\_booking.domain.ui.map.MapEvent  
import com.example.feature\_map\_booking.domain.ui.map.MapViewModel  
import com.google.android.gms.location.LocationServices  
import com.google.android.gms.maps.CameraUpdateFactory  
import com.google.android.gms.maps.model.CameraPosition  
import com.google.android.gms.maps.model.LatLng  
import com.google.maps.android.compose.GoogleMap  
import com.google.maps.android.compose.Marker  
import com.google.maps.android.compose.MarkerState  
import com.google.maps.android.compose.rememberCameraPositionState  
  
@SuppressLint("MissingPermission") // Chúng ta đã xử lý quyền bên dưới  
@Composable  
fun MapScreen(  
 viewModel: MapViewModel = hiltViewModel(),  
 onNavigateToLocationDetail: (String) -> Unit  
) {  
 val state by viewModel.state.collectAsState()  
 val context = LocalContext.current  
  
 var hasLocationPermission by remember { mutableStateOf(false) }  
 val locationPermissionLauncher = rememberLauncherForActivityResult(  
 contract = ActivityResultContracts.RequestPermission(),  
 onResult = { isGranted ->  
 hasLocationPermission = isGranted  
 }  
 )  
  
 val fusedLocationClient = remember { LocationServices.getFusedLocationProviderClient(context) }  
  
 LaunchedEffect(Unit) {  
 locationPermissionLauncher.launch(Manifest.permission.ACCESS\_FINE\_LOCATION)  
 }  
  
 LaunchedEffect(hasLocationPermission) {  
 if (hasLocationPermission) {  
 fusedLocationClient.lastLocation.addOnSuccessListener { location ->  
 location?.let {  
 viewModel.onEvent(MapEvent.OnMapReady(LatLng(it.latitude, it.longitude)))  
 }  
 }  
 }  
 }  
  
 val defaultLocation = LatLng(10.762622, 106.660172) // TP.HCM  
 val cameraPositionState = rememberCameraPositionState {  
 position = CameraPosition.fromLatLngZoom(state.lastKnownLocation ?: defaultLocation, 12f)  
 }  
  
 LaunchedEffect(state.lastKnownLocation) {  
 state.lastKnownLocation?.let {  
 cameraPositionState.animate(  
 update = CameraUpdateFactory.newLatLngZoom(it, 15f),  
 durationMs = 1000  
 )  
 }  
 }  
  
 Box(Modifier.fillMaxSize()) {  
 GoogleMap(  
 modifier = Modifier.fillMaxSize(),  
 cameraPositionState = cameraPositionState,  
 ) {  
 state.hospitals.forEach { hospital ->  
 hospital.location?.let { geoPoint ->  
 Marker(  
 state = MarkerState(position = LatLng(geoPoint.latitude, geoPoint.longitude)),  
 title = hospital.name,  
 snippet = hospital.address,  
 onInfoWindowClick = {  
 onNavigateToLocationDetail(hospital.id)  
 }  
 )  
 }  
 }  
 }  
  
 if (state.isLoading) {  
 CircularProgressIndicator(modifier = Modifier.align(Alignment.Center))  
 }  
  
 state.error?.let {  
 // TODO: Hiển thị thông báo lỗi thân thiện hơn, ví dụ như một SnackBar  
 }  
  
 // TODO: Thêm UI cho bộ lọc và tìm kiếm ở đây (ví dụ: một nút ở góc màn hình)  
 }  
}

## 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 androidx.lifecycle.ViewModel  
import androidx.lifecycle.viewModelScope  
import com.example.feature\_map\_booking.domain.usecase.GetNearbyHospitalsUseCase  
import com.google.android.gms.maps.model.LatLng  
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(  
 private val getNearbyHospitalsUseCase: GetNearbyHospitalsUseCase  
) : ViewModel() {  
  
 private val \_state = MutableStateFlow(MapState())  
 val state = \_state.asStateFlow()  
  
 fun onEvent(event: MapEvent) {  
 when (event) {  
 is MapEvent.OnMapReady -> {  
 \_state.update { it.copy(lastKnownLocation = event.location) }  
 fetchHospitals(event.location)  
 }  
 is MapEvent.OnMarkerClick -> {  
 // Navigation will be handled in the Composable  
 }  
 }  
 }  
  
 private fun fetchHospitals(location: LatLng) {  
 viewModelScope.launch {  
 \_state.update { it.copy(isLoading = true) }  
 getNearbyHospitalsUseCase(location.latitude, location.longitude)  
 .onSuccess { hospitals ->  
 \_state.update {  
 it.copy(isLoading = false, hospitals = hospitals)  
 }  
 }  
 .onFailure { error ->  
 \_state.update {  
 it.copy(isLoading = false, error = error.message)  
 }  
 }  
 }  
 }  
}

## 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): 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)  
 }  
}

## 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.example.feature\_map\_booking.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.example.feature\_map\_booking.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/GetNearbyHospitalsUseCase.kt

package com.example.feature\_map\_booking.domain.usecase  
  
// feature\_map\_booking/src/main/java/com/smartblood/mapbooking/domain/usecase/GetNearbyHospitalsUseCase.kt  
  
import com.example.feature\_map\_booking.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(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)  
 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)  
}

## 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

//D:\SmartBloodDonationAndroid\feature\_profile\src\main\java\com\example\feature\_profile\data\repository\ProfileRepositoryImpl.kt  
package com.smartblood.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.profile.domain.model.UserProfile  
import com.smartblood.profile.domain.repository.ProfileRepository  
import kotlinx.coroutines.tasks.await  
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"))  
 val querySnapshot = firestore.collection("users").document(userId)  
 .collection("donation\_history")  
 .orderBy("date", com.google.firebase.firestore.Query.Direction.DESCENDING)  
 .get().await()  
  
 val history = querySnapshot.toObjects(DonationRecord::class.java)  
 Result.success(history)  
 } catch (e: Exception) {  
 Result.failure(e)  
 }  
 }  
}

## feature\_profile/src/main/java/com/example/feature\_profile/di/ProfileModule.kt

//D:\SmartBloodDonationAndroid\feature\_profile\src\main\java\com\example\feature\_profile\di\ProfileModule.kt  
package com.smartblood.profile.di  
  
import com.smartblood.profile.data.repository.ProfileRepositoryImpl  
import com.smartblood.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 java.util.Date  
  
data class DonationRecord(  
 val id: String = "",  
 val hospitalName: String = "",  
 val date: Date = Date(),  
 val unitsDonated: Int = 1  
)

## feature\_profile/src/main/java/com/example/feature\_profile/domain/model/UserProfile.kt

//D:\SmartBloodDonationAndroid\feature\_profile\src\main\java\com\example\feature\_profile\domain\model\UserProfile.kt  
package com.smartblood.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, // Ví dụ: "A+", "O-", ...  
 val avatarUrl: String? = null,  
 val dateOfBirth: Date? = null,  
 val gender: String? = null, // "Male", "Female", "Other"  
 val lastDonationDate: Date? = null  
)

## feature\_profile/src/main/java/com/example/feature\_profile/domain/repository/ProfileRepository.kt

//D:\SmartBloodDonationAndroid\feature\_profile\src\main\java\com\example\feature\_profile\domain\repository\ProfileRepository.kt  
package com.smartblood.profile.domain.repository  
  
import com.smartblood.profile.domain.model.DonationRecord  
import com.smartblood.profile.domain.model.UserProfile  
import kotlin.Result  
  
interface ProfileRepository {  
 /\*\*  
 \* Lấy thông tin hồ sơ của người dùng hiện tại.  
 \*/  
 suspend fun getUserProfile(): Result<UserProfile>  
  
 /\*\*  
 \* Cập nhật thông tin hồ sơ của người dùng.  
 \*/  
 suspend fun updateUserProfile(userProfile: UserProfile): Result<Unit>  
  
 /\*\*  
 \* Lấy lịch sử hiến máu của người dùng hiện tại.  
 \*/  
 suspend fun getDonationHistory(): Result<List<DonationRecord>>  
}

## feature\_profile/src/main/java/com/example/feature\_profile/domain/usecase/CalculateNextDonationDateUseCase.kt

package com.smartblood.profile.domain.usecase  
  
import com.smartblood.profile.domain.model.UserProfile  
import java.util.Calendar  
import java.util.Date  
import java.util.concurrent.TimeUnit  
import javax.inject.Inject  
  
class CalculateNextDonationDateUseCase @Inject constructor() {  
  
 // Giả sử thời gian chờ giữa 2 lần hiến máu là 84 ngày  
 private val WAITING\_DAYS = 84  
  
 operator fun invoke(userProfile: UserProfile?): String {  
 val lastDonationDate = userProfile?.lastDonationDate ?: return "Bạn có thể hiến máu ngay!"  
  
 val calendar = Calendar.getInstance()  
 calendar.time = lastDonationDate  
 calendar.add(Calendar.DAY\_OF\_YEAR, WAITING\_DAYS)  
 val nextAvailableDate = calendar.time  
  
 val today = Date()  
  
 if (nextAvailableDate.before(today) || nextAvailableDate == today) {  
 return "Bạn có thể hiến máu ngay!"  
 }  
  
 val diffInMillis = nextAvailableDate.time - today.time  
 val daysRemaining = TimeUnit.MILLISECONDS.toDays(diffInMillis)  
  
 return if (daysRemaining > 1) {  
 "Bạn có thể hiến máu sau $daysRemaining ngày nữa"  
 } else {  
 "Bạn có thể hiến máu vào ngày mai"  
 }  
 }  
}

## feature\_profile/src/main/java/com/example/feature\_profile/domain/usecase/GetDonationHistoryUseCase.kt

//D:\SmartBloodDonationAndroid\feature\_profile\src\main\java\com\example\feature\_profile\domain\usecase\GetDonationHistoryUseCase.kt  
package com.smartblood.profile.domain.usecase  
  
import com.smartblood.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

//D:\SmartBloodDonationAndroid\feature\_profile\src\main\java\com\example\feature\_profile\domain\usecase\GetUserProfileUseCase.kt  
package com.smartblood.profile.domain.usecase  
  
import com.smartblood.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/UpdateUserProfileUseCase.kt

//D:\SmartBloodDonationAndroid\feature\_profile\src\main\java\com\example\feature\_profile\domain\usecase\UpdateUserProfileUseCase.kt  
package com.smartblood.profile.domain.usecase  
  
import com.smartblood.profile.domain.model.UserProfile  
import com.smartblood.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  
  
// feature\_profile/src/main/java/com/smartblood/profile/ui/history/DonationHistoryScreen.kt  
  
  
import androidx.compose.foundation.layout.Box  
import androidx.compose.foundation.layout.fillMaxSize  
import androidx.compose.foundation.layout.padding  
import androidx.compose.material.icons.Icons  
import androidx.compose.material.icons.automirrored.filled.ArrowBack  
import androidx.compose.material3.\*  
import androidx.compose.runtime.Composable  
import androidx.compose.ui.Alignment  
import androidx.compose.ui.Modifier  
  
@OptIn(ExperimentalMaterial3Api::class)  
@Composable  
fun DonationHistoryScreen(  
 onNavigateBack: () -> Unit  
) {  
 Scaffold(  
 topBar = {  
 TopAppBar(  
 title = { Text("Lịch sử hiến máu") },  
 navigationIcon = {  
 IconButton(onClick = onNavigateBack) {  
 Icon(Icons.AutoMirrored.Filled.ArrowBack, "Back")  
 }  
 }  
 )  
 }  
 ) { paddingValues ->  
 Box(  
 modifier = Modifier  
 .fillMaxSize()  
 .padding(paddingValues),  
 contentAlignment = Alignment.Center  
 ) {  
 Text("Danh sách các lần hiến máu sẽ ở đây.")  
 }  
 }  
}

## 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.Box  
import androidx.compose.foundation.layout.fillMaxSize  
import androidx.compose.foundation.layout.padding  
import androidx.compose.material.icons.Icons  
import androidx.compose.material.icons.automirrored.filled.ArrowBack  
import androidx.compose.material3.\*  
import androidx.compose.runtime.Composable  
import androidx.compose.ui.Alignment  
import androidx.compose.ui.Modifier  
  
@OptIn(ExperimentalMaterial3Api::class)  
@Composable  
fun EditProfileScreen(  
 onNavigateBack: () -> Unit  
) {  
 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),  
 contentAlignment = Alignment.Center  
 ) {  
 Text("Giao diện chỉnh sửa hồ sơ sẽ ở đây.")  
 }  
 }  
}

## feature\_profile/src/main/java/com/example/feature\_profile/ui/ProfileContract.kt

//D:\SmartBloodDonationAndroid\feature\_profile\src\main\java\com\example\feature\_profile\ui\ProfileContract.kt  
package com.smartblood.profile.ui  
  
import com.smartblood.profile.domain.model.UserProfile  
  
data class ProfileState(  
 val isLoading: Boolean = false,  
 val userProfile: UserProfile? = null,  
 val error: String? = null  
)  
  
sealed class ProfileEvent {  
 object OnEditProfileClicked : ProfileEvent()  
 object OnViewDonationHistoryClicked : ProfileEvent()  
}

## feature\_profile/src/main/java/com/example/feature\_profile/ui/ProfileScreen.kt

//D:\SmartBloodDonationAndroid\feature\_profile\src\main\java\com\example\feature\_profile\ui\ProfileScreen.kt  
package com.smartblood.profile.ui  
  
import androidx.compose.foundation.layout.\*  
import androidx.compose.foundation.shape.CircleShape  
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.draw.clip  
import androidx.compose.ui.layout.ContentScale  
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 coil.compose.AsyncImage  
  
@OptIn(ExperimentalMaterial3Api::class)  
@Composable  
fun ProfileScreen(  
 viewModel: ProfileViewModel = hiltViewModel(),  
 onNavigateToEditProfile: () -> Unit,  
 onNavigateToDonationHistory: () -> Unit  
) {  
 val state by viewModel.state.collectAsState()  
  
 Scaffold(  
 topBar = {  
 TopAppBar(title = { Text("Hồ sơ của tôi") })  
 }  
 ) { paddingValues ->  
 Box(  
 modifier = Modifier  
 .fillMaxSize()  
 .padding(paddingValues)  
 .padding(16.dp),  
 contentAlignment = Alignment.Center  
 ) {  
 if (state.isLoading) {  
 CircularProgressIndicator()  
 } else if (state.error != null) {  
 Text(text = "Lỗi: ${state.error}", color = MaterialTheme.colorScheme.error)  
 } else if (state.userProfile != null) {  
 val profile = state.userProfile!!  
 Column(  
 horizontalAlignment = Alignment.CenterHorizontally,  
 verticalArrangement = Arrangement.spacedBy(16.dp)  
 ) {  
 AsyncImage(  
 model = profile.avatarUrl ?: "https://example.com/default\_avatar.png", // Thay bằng link ảnh mặc định  
 contentDescription = "Ảnh đại diện",  
 modifier = Modifier  
 .size(120.dp)  
 .clip(CircleShape),  
 contentScale = ContentScale.Crop  
 )  
 Text(  
 text = profile.fullName,  
 fontSize = 24.sp,  
 fontWeight = FontWeight.Bold  
 )  
 Text(text = "Email: ${profile.email}")  
 Text(text = "Nhóm máu: ${profile.bloodType ?: "Chưa cập nhật"}")  
  
 Spacer(modifier = Modifier.height(24.dp))  
  
 Button(onClick = onNavigateToEditProfile, modifier = Modifier.fillMaxWidth()) {  
  
 Text("Chỉnh sửa thông tin")  
 }  
 OutlinedButton(onClick = onNavigateToDonationHistory, modifier = Modifier.fillMaxWidth()) {  
 Text("Xem lịch sử hiến máu")  
 }  
 }  
 }  
 }  
 }  
}

## feature\_profile/src/main/java/com/example/feature\_profile/ui/ProfileViewModel.kt

//D:\SmartBloodDonationAndroid\feature\_profile\src\main\java\com\example\feature\_profile\ui\ProfileViewModel.kt  
package com.smartblood.profile.ui  
  
import androidx.lifecycle.ViewModel  
import androidx.lifecycle.viewModelScope  
import com.smartblood.profile.domain.usecase.GetUserProfileUseCase  
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 ProfileViewModel @Inject constructor(  
 private val getUserProfileUseCase: GetUserProfileUseCase  
) : ViewModel() {  
  
 private val \_state = MutableStateFlow(ProfileState())  
 val state = \_state.asStateFlow()  
  
 init {  
 loadUserProfile()  
 }  
  
 private fun loadUserProfile() {  
 viewModelScope.launch {  
 \_state.update { it.copy(isLoading = true) }  
 getUserProfileUseCase()  
 .onSuccess { userProfile ->  
 \_state.update { it.copy(isLoading = false, userProfile = userProfile, error = null) }  
 }  
 .onFailure { exception ->  
 \_state.update { it.copy(isLoading = false, error = exception.message) }  
 }  
 }  
 }  
}

## 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/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