BÀI TẬP LỚN

LẬP TRÌNH THIẾT BỊ DI ĐỘNG

Xây dựng ứng dụng chuyển đổi đơn Vị và tiền Tệ

Sinh viên thực hiện: Ba Si Co

MSSV: 62132949

Lớp: 64.TTMMT

Cán bộ giảng dạy: Mai Cường Thọ

1. Giới thiệu

1.1 Mục tiêu của bài tập:

Dự án bài tập "Ứng dụng Chuyển đổi Đơn vị và Tiền tệ"một công cụ đơn giản trên điện thoại để người dùng có thể chuyển đổi qua lại giữa các đơn vị đo lường và một số loại tiền tệ thường dùng.

Chuyển đổi giữa các đơn vị trong các nhóm: Độ dài (m, cm, km...), Khối lượng (kg, g, pound...), Nhiệt độ (°C, °F), Thời gian (giây, phút, giờ...), và Diện tích (m^2, ha...). Cung cấp tính năng đổi tiền tệ với tỷ giá tham khảo.

Chuyển đổi giữa một số loại tiền tệ chính (VND, USD, EUR...). Ứng dụng sẽ cố gắng lấy tỷ giá từ một API trực tuyến để người dùng có thông tin cập nhật.

Tích hợp một máy tính đơn giản: Thêm một màn hình máy tính để thực hiện các phép cộng, trừ, nhân, chia cơ bản, có thể kèm theo xử lý ngoặc và phần trăm ở mức độ đơn giản.

Lưu lại lịch sử các thao tác: Cho phép lưu lại kết quả các lần chuyển đổi và tính toán để người dùng có thể xem lại khi cần.

Giao diện dễ nhìn và dễ thao tác.

Thực hành việc tạo giao diện bằng XML, viết code Java xử lý sự kiện, làm việc với Fragment, lưu trữ dữ liệu đơn giản bằng SQLite và gọi API.

1.2 Tầm quan trọng của ứng dụng

Giải quyết nhu cầu chuyển đổi đa dạng: Ứng dụng gom nhiều loại chuyển đổi thông dụng (độ dài, khối lượng, nhiệt độ, thời gian, diện tích, tiền tệ) vào một nơi duy nhất, giúp người dùng tiết kiệm thời gian và công sức tìm kiếm.

Tiện lợi và nhanh chóng: Với giao diện được thiết kế để dễ thao tác, người dùng có thể nhanh chóng nhập liệu và nhận kết quả ngay lập tức. Việc tích hợp máy tính cơ bản cũng giúp thực hiện các phép tính liên quan một cách liền mạch.

Cung cấp thông tin tiền tệ cập nhật: Khả năng kết nối API để lấy tỷ giá mới nhất giúp đảm bảo tính chính xác tương đối cho các giao dịch hoặc ước tính tài chính liên quan đến ngoại tệ. Hỗ trợ tra cứu và quản lý thông tin:

Chức năng lưu lịch sử giúp người dùng dễ dàng xem lại các kết quả chuyển đổi và tính toán quan trọng mà không cần thực hiện lại.

Công cụ học tập và làm việc hiệu quả: Đối với học sinh, sinh viên và những người làm việc khi cần thiết.

1.3 Đặc điểm chính của ứng dụng

- Chuyển đổi Đơn vị Đo lường:

cho phép người dùng chuyển đổi giá trị giữa các đơn vị khác nhau trong cùng một loại đo lường.

Chi tiết:

+ Hỗ trợ các loại đơn vị: Độ dài (m, cm, km, inch, feet, mm), Khối lượng (kg, g, pound, oz, mg), Nhiệt độ (°C, °F, K), Thời gian (giây, phút, giờ, ngày), và Diện tích (m^2, km^2, ha, ft^2, acre).

+ Giao diện chọn loại đơn vị và chọn đơn vị nguồn/đích một cách trực quan.

+ Kết quả được cập nhật tự động ngay khi người dùng nhập giá trị hoặc thay đổi đơn vị.

+ Có nút "Hoán đổi" để nhanh chóng đảo chiều chuyển đổi giữa đơn vị nguồn và đích.

- Chuyển đổi Tiền tệ:

Cho phép người dùng quy đổi giá trị giữa các loại tiền tệ phổ biến.

Chi tiết:

+ Hỗ trợ các loại tiền tệ chính như VND, USD, EUR, JPY, KRW, CNY và các loại khác tùy thuộc vào dữ liệu từ API.

+ Tự động cập nhật tỷ giá hối đoái từ một nguồn API trực tuyến (ví dụ: ExchangeRate-API) để đảm bảo thông tin tương đối chính xác và mới nhất.

+ Hiển thị thời gian lần cuối cập nhật tỷ giá.

+ Cho phép người dùng làm mới tỷ giá thủ công.

+ Giao diện chọn tiền tệ nguồn và đích (sử dụng Spinner hoặc RecyclerView như thiết kế mới).

- Máy tính Cơ bản:

Tiện ích tính toán tích hợp sẵn trong ứng dụng.

Chi tiết:

+ Hỗ trợ các phép toán số học cơ bản: cộng (+), trừ (−), nhân (×), chia (÷).

+ Cho phép nhập liệu biểu thức có sử dụng dấu ngoặc đơn ( ).

+ Hỗ trợ tính toán với phần trăm (%).

+ Giao diện bàn phím số và các nút chức năng (C - Xóa hết, =, các phép toán) được thiết kế dễ sử dụng.

+ Hiển thị cả biểu thức đang nhập và kết quả tính toán.

- Lưu trữ và Quản lý Lịch sử:

Người dùng lưu lại các kết quả chuyển đổi đơn vị, tiền tệ và các phép tính đã thực hiện.

Chi tiết:

+ Lưu trữ thông tin bao gồm loại chuyển đổi/phép tính, giá trị/biểu thức nguồn, đơn vị nguồn (nếu có), kết quả, đơn vị đích (nếu có), và thời gian thực hiện.

+ Hiển thị danh sách lịch sử một cách rõ ràng, sắp xếp theo thời gian mới nhất trước.

+ Cung cấp chức năng tìm kiếm/lọc trong lịch sử.

+ Cho phép người dùng xóa từng mục lịch sử hoặc xóa toàn bộ lịch sử.

- Cài đặt Giao diện Người dùng :

Hiện tại, ứng dụng có mục "Giới thiệu ứng dụng" để cung cấp thông tin về ứng dụng.

- Giao diện Người dùng Trực quan và Dễ sử dụng:

Thiết kế giao diện người dùng và trải nghiệm người, dễ hiểu và dễ thao tác, sử dụng các thành phần của Material Design.

Chi tiết:

+ Điều hướng chính thông qua BottomNavigationView, các màn hình chức năng được tổ chức dưới dạng Fragment, thông báo và phản hồi cho người dùng được hiển thị qua Toast hoặc các TextView trạng thái.

2. Phân tích yêu cầu

2.1 Yêu cầu chức năng:

Ứng dụng "Chuyển đổi Đơn vị và Tiền tệ" được thiết kế để thực hiện các chức năng chính sau:

- Chuyển đổi Đơn vị Đo lường:

- Cho phép người dùng lựa chọn một loại đơn vị đo lường từ danh sách (ví dụ: Độ dài, Khối lượng, Nhiệt độ, Thời gian, Diện tích).
- Sau khi chọn loại, người dùng có thể xác định đơn vị nguồn và đơn vị đích từ các đơn vị con tương ứng.
- Người dùng nhập giá trị số cần chuyển đổi vào ô của đơn vị nguồn.
- Ứng dụng tự động tính toán và hiển thị kết quả ở ô đơn vị đích khi có thay đổi giá trị hoặc đơn vị.
- Cung cấp nút để người dùng hoán đổi nhanh giữa đơn vị nguồn và đích.

- Chuyển đổi Tiền tệ:

- Cho phép người dùng lựa chọn tiền tệ nguồn và tiền tệ đích từ danh sách các loại tiền được hỗ trợ.

- Người dùng nhập số tiền cần chuyển đổi.
- Ứng dụng kết nối Internet để lấy và cập nhật tỷ giá hối đoái mới nhất từ API.
- Kết quả chuyển đổi tiền tệ được tự động tính toán và hiển thị dựa trên tỷ giá cập nhật.
- Hiển thị thông tin về thời điểm tỷ giá được cập nhật lần cuối.
- Cho phép chọn một hàng tiền tệ làm cơ sở để nhập giá trị, các hàng khác tự động cập nhật.

- Máy tính Cơ bản:

- Hỗ trợ nhập liệu các chữ số, dấu thập phân, và các phép toán cơ bản (+, −, ×, ÷).
- Cho phép sử dụng dấu ngoặc đơn ( ) trong biểu thức.
- Hỗ trợ tính toán với ký hiệu phần trăm %.
- Hiển thị đồng thời biểu thức đang nhập và kết quả tính toán.
- Cung cấp nút "C" để xóa toàn bộ phép tính và kết quả.

- Quản lý Lịch sử:

- Cho phép người dùng lưu lại kết quả của các phép chuyển đổi đơn vị, tiền tệ và các phép tính.
- Thông tin lưu trữ bao gồm chi tiết về thao tác (loại, giá trị/biểu thức nguồn, đơn vị nguồn, kết quả, đơn vị đích) và thời gian thực hiện.
- Hiển thị danh sách lịch sử đã lưu, sắp xếp theo thứ tự thời gian giảm dần.
- Cung cấp khả năng tìm kiếm hoặc lọc các mục trong lịch sử.
- Cho phép xóa từng mục lịch sử hoặc xóa toàn bộ lịch sử.

- Cài đặt Ứng dụng:

- Cung cấp mục "Giới thiệu ứng dụng" để người dùng xem thông tin về sản phẩm.

2.2 Kế hoạch phát triển

Quá trình phát triển ứng dụng "Chuyển đổi Đơn vị và Tiền tệ" được chia thành các giai đoạn chính như sau

Giai đoạn 1: Khởi tạo Dự án và Xây dựng Khung Giao diện Cơ bản

Mục tiêu:

- Thiết lập môi trường phát triển Android Studio.

- Tạo cấu trúc dự án ban đầu.

- Xây dựng các màn hình điều hướng chính và luồng hoạt động cơ bản của ứng dụng.

Công việc thực hiện:

- Tạo project Android mới.

- Thiết kế và triển khai **SplashActivity** (màn hình chờ).

- Thiết kế và triển khai **WelcomeActivity** (màn hình chào mừng).

- Thiết kế và triển khai **MainActivity** với **BottomNavigationView** để điều hướng giữa các chức năng chính.

- Tạo các Fragment rỗng ban đầu cho các mục: Đơn vị, Tiền tệ, Lịch sử, Cài đặt, và Máy tính.

- Thiết lập luồng chuyển màn hình cơ bản từ Splash -> Welcome -> Main.

Giai đoạn 2: Triển khai Chức năng Chuyển đổi Đơn vị Cốt lõi

Mục tiêu:

- Hoàn thiện chức năng chuyển đổi giữa các đơn vị đo lường phổ biến.

- Xây dựng giao diện người dùng cho việc chọn loại đơn vị, đơn vị nguồn/đích và nhập liệu.

Công việc thực hiện:

- Thiết kế layout **fragment_unit_converter.xml**.

- Triển khai lớp **UnitConverterUtil.java** chứa logic tính toán cho các loại đơn vị: Độ dài, Khối lượng, Nhiệt độ, Thời gian, Diện tích.

- Code trong **UnitConverterFragment.java** để: Hiển thị danh sách loại đơn vị và các đơn vị con tương ứng trong Spinner, xử lý nhập liệu từ người dùng, gọi UnitConverterUtil để thực hiện chuyển đổi, hiển thị kết quả.

Giai đoạn 3: Triển khai chức năng chuyển đổi tiền tệ và tích hợp api

Mục tiêu:

- Xây dựng chức năng chuyển đổi tiền tệ.

- Tích hợp API để lấy tỷ giá hối đoái tự động.

Công việc thực hiện:

- Thiết kế layout **fragment_currency_converter.xml**.

- Thêm quyền INTERNET vào AndroidManifest.xml.

- Tích hợp thư viện Volley để thực hiện các yêu cầu mạng.

- Code trong CurrencyConverterFragment.java: Gọi API lấy tỷ giá (ví dụ: từ ExchangeRate-API), hiển thị danh sách các loại tiền tệ và cho phép người dùng chọn tiền tệ nguồn/đích (hoặc chọn tiền tệ cơ sở), thực hiện tính toán chuyển đổi dựa trên tỷ giá, Hiển thị thời gian cập nhật tỷ giá và cho phép làm mới, xử lý trạng thái tải và lỗi.

Giai đoạn 4: Triển khai chức năng máy tính cơ bản, lưu trữ và hiển thị lịch sử

Mục tiêu:

- Tạo một máy tính đơn giản tích hợp trong ứng dụng.

- Cho phép người dùng lưu và xem lại lịch sử các phép chuyển đổi và tính toán.

Công việc thực hiện:

- Thiết kế layout fragment_calculator.xml với màn hình hiển thị và các nút bấm số, phép toán, chức năng.

- Code trong CalculatorFragment.java : Xử lý sự kiện nhấn nút, xây dựng chuỗi biểu thức, để đánh giá biểu thức và hiển thị kết quả.

- Thiết kế cơ sở dữ liệu SQLite: Tạo **HistoryContract.java** và **HistoryDbHelper.java**.

- Tạo lớp model HistoryItem.java.

- Cập nhật **UnitConverterFragment, CurrencyConverterFragment**, và **CalculatorFragment** để thêm chức năng lưu kết quả vào CSDL khi người dùng nhấn nút "Lưu".

- Thiết kế layout **fragment_history.xml** với **RecyclerView** và **SearchView.**

- Tạo **HistoryAdapter.java** để hiển thị dữ liệu lịch sử lên **RecyclerView**.

- Viết code trong **HistoryFragment.java** để đọc dữ liệu từ CSDL, hiển thị lên danh sách, xử lý tìm kiếm/lọc, và cho phép xóa từng mục hoặc toàn bộ lịch sử.

Giai đoạn 5: Hoàn thiện Giao diện, Cài đặt và Tối ưu

Mục tiêu:

- Đồng bộ màu sắc và phong cách giao diện toàn ứng dụng.

Công việc thực hiện:

- Định nghĩa bảng màu chung trong colors.xml và áp dụng vào themes.xml.

- Cập nhật màu sắc cho tất cả các Activity và Fragment.

- Điều chỉnh layout, style, font chữ, icon cho các màn hình để đạt được giao diện hiện đại và dễ sử dụng hơn.

Triển khai **SettingsFragment** với chức năng "Giới thiệu ứng dụng".

3. Thiết kế ứng dụng

3.1 Kiến trúc tổng thể:

Ứng dụng "Chuyển đổi Đơn vị và Tiền tệ" được phát triển dưới dạng một ứng dụng di động Android độc lập. Kiến trúc của ứng dụng tập trung vào

việc xử lý logic và hiển thị dữ liệu ngay trên thiết bị của người dùng, với một thành phần nhỏ tương tác với dịch vụ bên ngoài để cập nhật dữ liệu tỷ giá.

1. **Phía Client :**

Toàn bộ giao diện người dùng (UI), logic xử lý chính của các phép chuyển đổi đơn vị, logic máy tính, và quản lý dữ liệu lịch sử đều được thực hiện trực tiếp trên thiết bị Android của người dùng.

Ứng dụng được xây dựng dựa trên các thành phần cơ bản của Android SDK:

- Activities: Quản lý các màn hình chính và luồng chính của ứng dụng (SplashActivity, WelcomeActivity, MainActivity).

- Fragments: Chia nhỏ giao diện và logic cho từng chức năng cụ thể UnitConverterFragment, CurrencyConverterFragment, CalculatorFragment, HistoryFragment, SettingsFragment, giúp quản lý code dễ dàng hơn và tối ưu hóa cho các kích thước màn hình khác nhau.

- Layouts (XML): Định nghĩa cấu trúc và giao diện của từng màn hình và thành phần.

- SQLite Database: Lưu trữ dữ liệu lịch sử chuyển đổi và tính toán một cách cục bộ trên thiếtGES thiết bị.

2. **Tương tác với Dịch vụ Bên ngoài (External API Integration):**

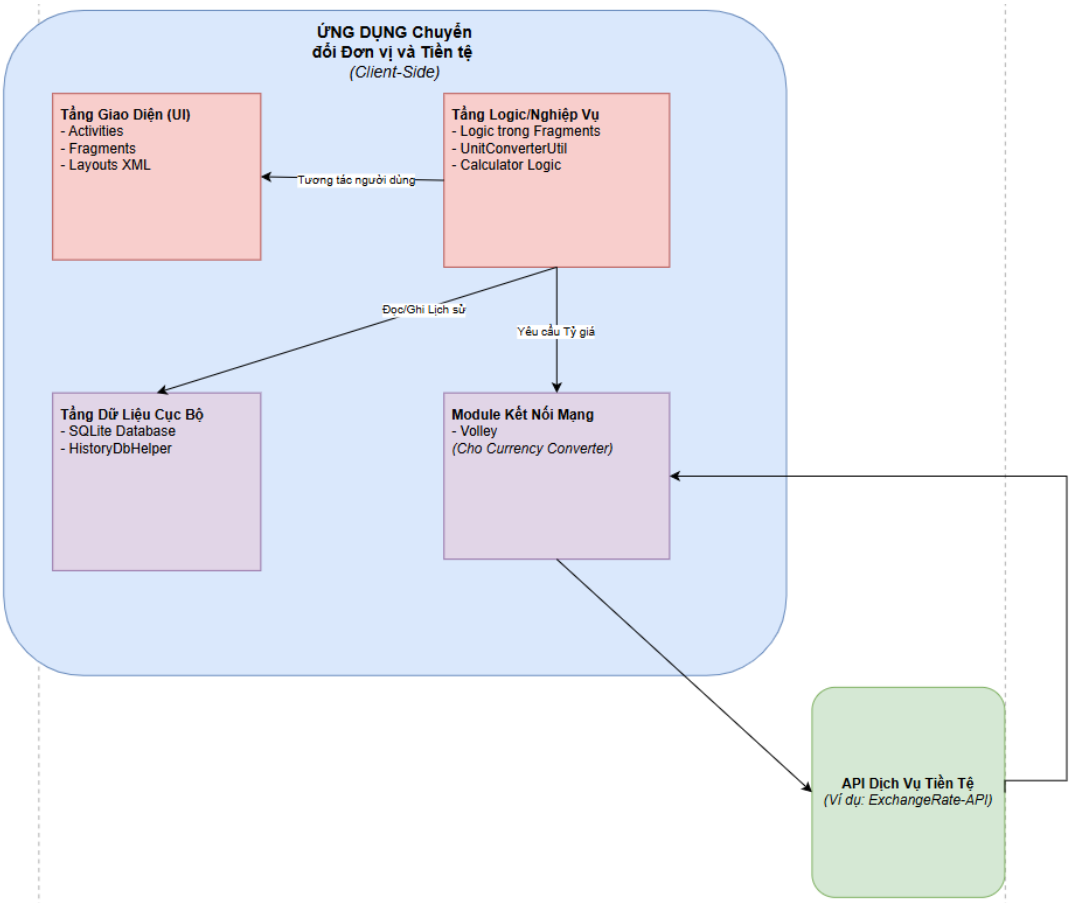
Chức năng Chuyển đổi Tiền tệ: Để đảm bảo tính cập nhật của tỷ giá hối đoái, ứng dụng có một thành phần nhỏ tương tác với một API dịch vụ tiền tệ công cộng bên ngoài (ví dụ: ExchangeRate-API).

Luồng tương tác:

- **CurrencyConverterFragment** sử dụng thư viện Volley để gửi yêu cầu HTTP GET đến endpoint của API.

- API trả về dữ liệu tỷ giá dưới dạng JSON.

Sơ đồ kiến trúc tổng thể đơn giản có thể hình dung như sau:



3.2 Thiết kế giao diện người dùng (UI Design):

- Giao diện người dùng (UI) của ứng dụng "Chuyển đổi Đơn vị và Tiền tệ" được thiết kế với mục tiêu đơn giản, trực quan, và dễ sử dụng, tuân thủ các nguyên tắc của Material Design để mang lại trải nghiệm người dùng tốt trên nền tảng Android. Màu sắc chủ đạo của ứng dụng là nền trắng, kết hợp với màu xanh biển sáng làm điểm nhấn và các sắc thái xám cho văn bản, tạo cảm giác sạch sẽ và dễ đọc.

3.2.1. Luồng Màn hình Chính (Main Flow)

Luồng hoạt động cơ bản của người dùng khi khởi động ứng dụng :

- Màn hình Chờ *(SplashActivity)*: hiển thị logo và tên ứng dụng trong một khoảng thời gian ngắn, nền màu xanh biển sáng (app_primary) chữ trắng, tự động chuyển sang Màn hình Chào mừng.

- Màn hình Chào mừng *(WelcomeActivity)*:

- Hiển thị lời chào và mô tả ngắn gọn về ứng dụng.

- Có nút "Bắt đầu ngay" nổi bật (nền xanh dương đậm, chữ trắng) để người dùng vào màn hình chính.
- Nền trắng, tiêu đề màu xanh biển sáng, mô tả màu xám nhạt.

- Màn hình Chính (MainActivity):

- Sử dụng *BottomNavigationView* ở cuối màn hình với các tab: Lịch sử, Đơn vị, Máy tính, Tiền tệ, Cài đặt.
- Mỗi tab sẽ hiển thị một *Fragment* tương ứng trong một FrameLayout chiếm phần lớn màn hình.
- Nền trắng, *BottomNavigationView* có nền trắng, item được chọn có icon và text màu xanh biển sáng, item không được chọn có màu xám nhạt.

3.2.2. Mô tả Giao diện các Màn hình Chức năng (Fragments)

- Màn hình Chuyển đổi Đơn vị *(UnitConverterFragment):*

- Thanh chọn loại đơn vị: Một *HorizontalScrollView* chứa các nút (icon + text) cho các loại đơn vị (Độ dài, Khối lượng, v.v.). Nút được chọn sẽ có giao diện nổi bật .
- Khu vực nhập liệu và kết quả: Được chia thành hai CardView (cho đơn vị nguồn và đơn vị đích).
- Mỗi CardView có một TextInputEditText lớn để hiển thị/nhập giá trị và một Spinner để chọn đơn vị con.
- Nút "Lưu lịch sử": Nằm ở cuối màn hình, sử dụng style nút chung của ứng dụng.
- Màu sắc: Nền trắng, CardView trắng có viền nhẹ, chữ nhập liệu/kết quả chính màu xám đậm, kết quả có thể được làm nổi bật bằng màu xanh biển sáng.

- Màn hình Chuyển đổi Tiền tệ *(CurrencyConverterFragment - Giao diện "Converter Neo"):*

- Nền tối: Toàn bộ *Fragment* có nền đen hoặc xám rất đậm.
- Tiêu đề: Hiển thị tên ứng dụng/chức năng và thời gian cập nhật tỷ giá, nút Refresh, nút Lưu lịch sử. Chữ và icon màu trắng/sáng.
- Danh sách tiền tệ: Sử dụng *RecyclerView* để hiển thị mỗi loại tiền tệ trên một hàng.

- Bàn phím tùy chỉnh: Nằm ở cuối màn hình, gồm các nút số , nút chức năng (C, ←, %, nền xám nhạt hơn, chữ đen), và nút phép toán/bằng .

- Màn hình Máy tính *(CalculatorFragment):*

- Nền sáng: Toàn bộ Fragment có nền trắng.
- Tiêu đề: Chữ "Máy tính" màu xanh biển sáng, có icon Lưu lịch sử.
- Màn hình hiển thị: Hai TextView lớn, một cho biểu thức đang nhập (chữ xám nhạt, kích thước vừa), một cho kết quả (chữ xám đậm, kích thước lớn, in đậm).
- Lưới nút bấm (GridLayout):
    - Các nút số (0-9, .): Nền xám rất nhạt hoặc trắng có viền, chữ xám đậm.
    - Các nút chức năng (C, (), %): Nền xám nhạt hơn, chữ xám đậm.
    - Các nút phép toán (+, −, ×, ÷): Nền xanh biển sáng, chữ trắng.
    - Nút bằng (=): Nền xanh dương đậm, chữ trắng.

- Màn hình Lịch sử (HistoryFragment):

- Nền trắng.
- Thanh tìm kiếm (SearchView): Nằm ở trên cùng, nền xám rất nhạt, có viền.
- Danh sách lịch sử (RecyclerView): Mỗi mục lịch sử là một CardView nền trắng, có viền nhẹ.
- Nút "Xóa tất cả lịch sử": Nằm ở cuối, dạng TextButton, chữ và icon màu xanh biển sáng.

- Màn hình Cài đặt (SettingsFragment):

- Các mục cài đặt được nhóm theo tiêu đề (ví dụ: "Thông tin" - chữ màu xanh biển sáng).
- Mục "Giới thiệu ứng dụng": Dạng TextButton, chữ xám đậm, có icon.

3.3 Thiết kế cơ sở dữ liệu

- Ứng dụng sử dụng một bảng duy nhất để lưu trữ tất cả các loại lịch sử, được đặt tên là **HistoryDbHelper.**

- Cấu trúc của bảng này được định nghĩa trong lớp **HistoryContract** và bao gồm các cột sau:

Tên cột	Kiểu dữ liệu SQL	Mô tả	Ràng buộc
_ID (Từ BaseColumns)	INTEGER	Khóa chính của bảng, tự động tăng giá trị để định danh duy nhất mỗi mục lịch sử.	PRIMARY KEY
COLUMN_NAME_CONVERSION_TYPE	TEXT	Loại thao tác được lưu, ví dụ: "Độ dài", "Khối lượng", "Tiền tệ", "Phép tính".	
COLUMN_NAME_SOURCE_VALUE	TEXT	Giá trị nguồn người dùng nhập (cho chuyển đổi) hoặc toàn bộ biểu thức (cho máy tính). Lưu dạng TEXT để giữ định dạng.	
COLUMN_NAME_SOURCE_UNIT	TEXT	Đơn vị của giá trị nguồn (ví dụ: "m","kg", "USD"). Đối với phép tính, có thể là dấu "=" hoặc để trống.	
COLUMN_NAME_RESULT_VALUE	TEXT	Kết quả của phép chuyển đổi hoặc tính toán. Lưu dạng TEXT để giữ định dạng.	
COLUMN_NAME_RESULT_UNIT	TEXT	Đơn vị của kết quả (ví dụ: "cm", "pound", "VND"). Đối với phép tính, có thể để trống.	
COLUMN_NAME_TIMESTAMP	INTEGER	Thời gian mục lịch sử được tạo, lưu dưới dạng Unix timestamp (số mili giây từ Epoch).	

- **ID làm khóa chính:** Đảm bảo mỗi bản ghi lịch sử là duy nhất và dễ dàng tham chiếu.
- **COLUMN_NAME_CONVERSION_TYPE:** Giúp phân loại các mục lịch sử, hữu ích cho việc hiển thị hoặc lọc sau này nếu cần.
- **Lưu giá trị và kết quả dạng TEXT:** Mặc dù giá trị có thể là số, việc lưu dưới dạng TEXT giúp giữ nguyên định dạng mà người dùng nhìn thấy trên giao diện (bao gồm cả dấu phẩy/chấm phân cách hàng nghìn, số chữ số thập phân nhất định). Khi cần tính toán lại hoặc so sánh, có thể chuyển đổi sang kiểu số.

- **COLUMN_NAME_SOURCE_UNIT và COLUMN_NAME_RESULT_UNIT:** Quan trọng cho việc hiển thị lại lịch sử chuyển đổi đơn vị và tiền tệ. Đối với lịch sử máy tính, COLUMN_NAME_SOURCE_UNIT được dùng để lưu dấu "=", và COLUMN_NAME_RESULT_UNIT có thể để trống, giúp HistoryAdapter phân biệt và hiển thị cho phù hợp.
- **COLUMN_NAME_TIMESTAMP:** Cho phép sắp xếp lịch sử theo thứ tự thời gian (mới nhất trước) và hiển thị thời gian người dùng thực hiện thao tác.

3.4 Phân tích về bảo mật

- Ứng dụng "Chuyển đổi Đơn vị và Tiền tệ" chủ yếu xử lý các dữ liệu nhập liệu từ người dùng cho mục đích tính toán và chuyển đổi, cùng với việc lưu trữ lịch sử các thao tác này một cách cục bộ.

Các xem xét chính về bảo mật bao gồm:

- **Dữ liệu cục bộ:** Lịch sử chuyển đổi và tính toán được lưu trữ trong cơ sở dữ liệu SQLite trên thiết bị người dùng, nằm trong vùng lưu trữ riêng của ứng dụng (app sandbox). Điều này hạn chế quyền truy cập trực tiếp từ các ứng dụng khác, trừ khi thiết bị đã bị can thiệp (rooted). Hiện tại, dữ liệu này không được mã hóa do tính chất không quá nhạy cảm của thông tin.

- **Kết nối mạng:** Chức năng cập nhật tỷ giá tiền tệ sử dụng kết nối Internet. Ứng dụng đảm bảo việc sử dụng giao thức HTTPS khi giao tiếp với API dịch vụ tiền tệ bên ngoài để bảo vệ dữ liệu tỷ giá trên đường truyền khỏi bị nghe lén.

- **Quyền ứng dụng:** Ứng dụng chỉ yêu cầu quyền tối thiểu cần thiết là

android.permission.INTERNET cho việc cập nhật tỷ giá, không yêu cầu các quyền truy cập không liên quan khác.

- **Xử lý đầu vào:** Ứng dụng có các bước kiểm tra và xử lý cơ bản đối với dữ liệu nhập từ người dùng để tránh các lỗi không mong muốn do đầu vào không hợp lệ gây ra trong quá trình tính toán.

4. Cài đặt các xử lý, các chức năng chính

4.1. *Quản lý Dữ liệu Lịch sử với SQLite*

Để lưu trữ lịch sử các phép chuyển đổi và tính toán của người dùng, ứng dụng sử dụng cơ sở dữ liệu SQLite. Việc khởi tạo và quản lý cơ sở dữ liệu được thực hiện thông qua lớp HistoryDbHelper, kế thừa từ SQLiteOpenHelper.

Lớp **HistoryContract** chứa một lớp tĩnh bên trong là **HistoryEntry** để định nghĩa các hằng số cho tên bảng (TABLE_NAME) và tên các cột. Việc **implements BaseColumns** giúp tự động có một cột _ID là khóa chính, một quy ước phổ biến trong Android khi làm việc với SQLite. Các cột được chọn để lưu trữ đầy đủ thông tin cần thiết cho việc hiển thị lại lịch sử, bao gồm loại thao tác, các giá trị và đơn vị liên quan, cùng với dấu thời gian để sắp xếp.

```
public final class HistoryContract {
    private HistoryContract() {}

    /* Định nghĩa nội dung bảng lịch sử */
    public static class HistoryEntry implements BaseColumns {
        public static final String TABLE_NAME = "history_entries";

        // Cột _ID được tự động cung cấp bởi BaseColumns, dùng làm khóa chính

        public static final String COLUMN_NAME_CONVERSION_TYPE = "conversion_type"; // Loại: "Độ dài", "Tiền tệ", "Phép tính"
        public static final String COLUMN_NAME_SOURCE_VALUE = "source_value";   // Giá trị/Biểu thức nguồn
        public static final String COLUMN_NAME_SOURCE_UNIT = "source_unit";     // Đơn vị nguồn / Dấu "=" cho phép tính
        public static final String COLUMN_NAME_RESULT_VALUE = "result_value";      // Kết quả
        public static final String COLUMN_NAME_RESULT_UNIT = "result_unit";     // Đơn vị đích
        public static final String COLUMN_NAME_TIMESTAMP = "timestamp";   // Thời gian lưu
    }
}
```

4.1.1. Khởi tạo Cơ sở dữ liệu và Định nghĩa Bảng

Khi ứng dụng khởi chạy lần đầu và cơ sở dữ liệu chưa tồn tại, lớp **HistoryDbHelper** (kế thừa **SQLiteOpenHelper**) sẽ thực thi việc tạo bảng thông qua phương thức **onCreate().** Đoạn mã sau minh họa câu lệnh SQL cốt lõi để tạo bảng này:

```
private static final String SQL_CREATE_ENTRIES =
    "CREATE TABLE " + HistoryContract.HistoryEntry.TABLE_NAME + " (" +
        HistoryContract.HistoryEntry._ID + " INTEGER PRIMARY KEY," +
        HistoryContract.HistoryEntry.COLUMN_NAME_CONVERSION_TYPE + " TEXT," +
        HistoryContract.HistoryEntry.COLUMN_NAME_SOURCE_VALUE + " TEXT," +
        HistoryContract.HistoryEntry.COLUMN_NAME_SOURCE_UNIT + " TEXT," +
        HistoryContract.HistoryEntry.COLUMN_NAME_RESULT_VALUE + " TEXT," +
        HistoryContract.HistoryEntry.COLUMN_NAME_RESULT_UNIT + " TEXT," +
        HistoryContract.HistoryEntry.COLUMN_NAME_TIMESTAMP + " INTEGER)";

// Câu lệnh xóa bảng SQL
private static final String SQL_DELETE_ENTRIES =
    "DROP TABLE IF EXISTS " + HistoryContract.HistoryEntry.TABLE_NAME;

public HistoryDbHelper(Context context) {
  super(context, DATABASE_NAME, null, DATABASE_VERSION);
}

@Override
public void onCreate(SQLiteDatabase db) {
  Log.d(TAG, "onCreate: Creating database table: " + SQL_CREATE_ENTRIES);
  db.execSQL(SQL_CREATE_ENTRIES);
}
```

4.1.2. Thao tác Lưu và Đọc Lịch sử

Mục lịch sử mới được thực hiện qua phương thức **insertHistory()** trong **HistoryDbHelper**. Phương thức này sử dụng ContentValues để đóng gói dữ liệu và gọi **db.insert()** để chèn vào bảng.

```
public boolean insertHistory(String type, String sourceValue, String sourceUnit, String
resultValue, String resultUnit) {
  SQLiteDatabase db = this.getWritableDatabase();
  ContentValues values = new ContentValues();
  values.put(HistoryContract.HistoryEntry.COLUMN_NAME_CONVERSION_TYPE, type);
  values.put(HistoryContract.HistoryEntry.COLUMN_NAME_SOURCE_VALUE, sourceValue);
  values.put(HistoryContract.HistoryEntry.COLUMN_NAME_SOURCE_UNIT, sourceUnit);
  values.put(HistoryContract.HistoryEntry.COLUMN_NAME_RESULT_VALUE, resultValue);
  values.put(HistoryContract.HistoryEntry.COLUMN_NAME_RESULT_UNIT, resultUnit);
  values.put(HistoryContract.HistoryEntry.COLUMN_NAME_TIMESTAMP,
System.currentTimeMillis()); // Thời gian hiện tại

  long newRowId = db.insert(HistoryContract.HistoryEntry.TABLE_NAME, null, values);
  // db.close(); // Không nên close db ở đây nếu dùng helper nhiều lần
  Log.d(TAG, "insertHistory: Inserted row with ID: " + newRowId);
  return newRowId != -1; // Trả về true nếu insert thành công (ID khác -1)
}
```

4.2. Triển khai Logic Chuyển đổi Đơn vị

4.2.1. Cấu trúc và Nguyên tắc Hoạt động của UnitConverterUtil.java

Lớp **UnitConverterUtil** được thiết kế để chứa các phương thức tĩnh, giúp việc gọi hàm chuyển đổi từ các phần khác của ứng dụng (cụ thể là UnitConverterFragment) trở nên đơn giản và không cần khởi tạo đối tượng.

```
public static double convert(double value, String fromUnit, String toUnit, String unitType) {
   Log.i(TAG_UTIL, "Attempting to convert: value=" + value + ", fromUnit='" + fromUnit + "',
toUnit='" + toUnit + "', unitType='" + unitType + "'");

   if (unitType == null || fromUnit == null || toUnit == null) {
      Log.e(TAG_UTIL, "Conversion failed: Null input parameter(s).");
      return Double.NaN;
   }

   if (Objects.equals(unitType, TYPE_LENGTH)) {
      return convertLength(value, fromUnit, toUnit);
   } else if (Objects.equals(unitType, TYPE_WEIGHT)) {
      return convertWeight(value, fromUnit, toUnit);
   } else if (Objects.equals(unitType, TYPE_TEMPERATURE)) {
      return convertTemperature(value, fromUnit, toUnit);
   } else if (Objects.equals(unitType, TYPE_TIME)) {
      return convertTime(value, fromUnit, toUnit);
   } else if (Objects.equals(unitType, TYPE_AREA)) {
      return convertArea(value, fromUnit, toUnit);
   } else {
      Log.e(TAG_UTIL, "Conversion failed: Unknown unitType: '" + unitType + "'");
      return Double.NaN;
   }
}
```

4.3. Triển khai Chức năng Chuyển đổi Tiền tệ và Tương tác API

Chức năng chuyển đổi tiền tệ của ứng dụng yêu cầu lấy dữ liệu tỷ giá hối đoái cập nhật từ một dịch vụ API trực tuyến. Thư viện Volley được sử dụng để thực hiện các yêu cầu mạng và xử lý phản hồi.

4.3.1. Lấy Dữ liệu Tỷ giá từ API

```
private void fetchExchangeRates() {
   progressBarLoading.setVisibility(View.VISIBLE);
```

```
    textViewErrorMessage.setVisibility(View.GONE);
    recyclerViewCurrencies.setVisibility(View.INVISIBLE); // Ẩn đi khi đang tải

    JsonObjectRequest jsonObjectRequest = new JsonObjectRequest(Request.Method.GET,
API_URL, null,
            response -> {
                progressBarLoading.setVisibility(View.GONE);
                recyclerViewCurrencies.setVisibility(View.VISIBLE);
                try {
                    if ("success".equals(response.getString("result"))) {
                        updateTimestamp(response.getLong("time_last_update_unix"));
                        JSONObject rates = response.getJSONObject("rates");
                        exchangeRates.clear();
                        List<String> tempCurrencyCodes = new ArrayList<>(); // Danh sách mã tiền tệ
tạm thời

                        Iterator<String> keys = rates.keys();
                        while (keys.hasNext()) {
                            String key = keys.next();
                            exchangeRates.put(key, rates.getDouble(key));
                            tempCurrencyCodes.add(key);
                        }
                        setupDisplayItems(filterAndSortCurrencies(tempCurrencyCodes));

                        setBaseCurrencyByCode(currentBaseCurrencyCode);
                        updateConversions(); // Tính toán lại với tỷ giá mới
                    } else {
                        handleApiError("API error: " + response.optString("error-type"));
                    }
                } catch (Exception e) {
                    handleApiError("Error parsing API response: " + e.getMessage());
                }
            },
            error -> {
                progressBarLoading.setVisibility(View.GONE);
                recyclerViewCurrencies.setVisibility(View.VISIBLE);
                handleApiError("Network error: " + error.getMessage());
            });
    requestQueue.add(jsonObjectRequest);
}
```

4.3.2. Logic Tính toán Chuyển đổi Tiền tệ

Hàm calculateConversion (ví dụ minh họa, logic thực tế nằm trong updateConversions của Fragment) lấy số tiền cần đổi, mã tiền tệ nguồn và đích, cùng với bản đồ tỷ giá.

Nó hoạt động bằng cách: Quy đổi số tiền từ đơn vị nguồn sang đơn vị tham chiếu của API (thường là USD) bằng cách chia cho tỷ giá sourceCurrencyCode/USD. Sau đó, quy đổi giá trị USD đó sang đơn vị đích bằng cách nhân với tỷ giá targetCurrencyCode/USD.

Trong CurrencyConverterFragment với giao diện "Converter Neo", updateConversions() thực hiện logic tương tự:lấy giá trị nhập ở đồng tiền cơ sở, quy đổi nó về đồng tiền tham chiếu của API (USD), sau đó từ USD tính ra giá trị cho tất cả các đồng tiền khác trong danh sách và cập nhật RecyclerView.

```
public double calculateConversion(double amount, String sourceCurrencyCode, String
targetCurrencyCode, Map<String, Double> rates) {
   if (!rates.containsKey(sourceCurrencyCode) || !rates.containsKey(targetCurrencyCode)) {
      Log.e(TAG_UTIL, "Rate not found for " + sourceCurrencyCode + " or " +
targetCurrencyCode);
      return Double.NaN; // Không tìm thấy tỷ giá
   }

   double rateSourceToUSD = rates.get(sourceCurrencyCode); // Tỷ giá của đồng nguồn so với
USD
   double rateTargetToUSD = rates.get(targetCurrencyCode); // Tỷ giá của đồng đích so với
USD

   if (rateSourceToUSD == 0) return Double.NaN; // Tránh chia cho 0

   // Bước 1: Chuyển số tiền nguồn sang USD
   double amountInUSD = amount / rateSourceToUSD;

   // Bước 2: Chuyển từ USD sang số tiền đích
   double resultAmount = amountInUSD * rateTargetToUSD;

   return resultAmount;
}
```

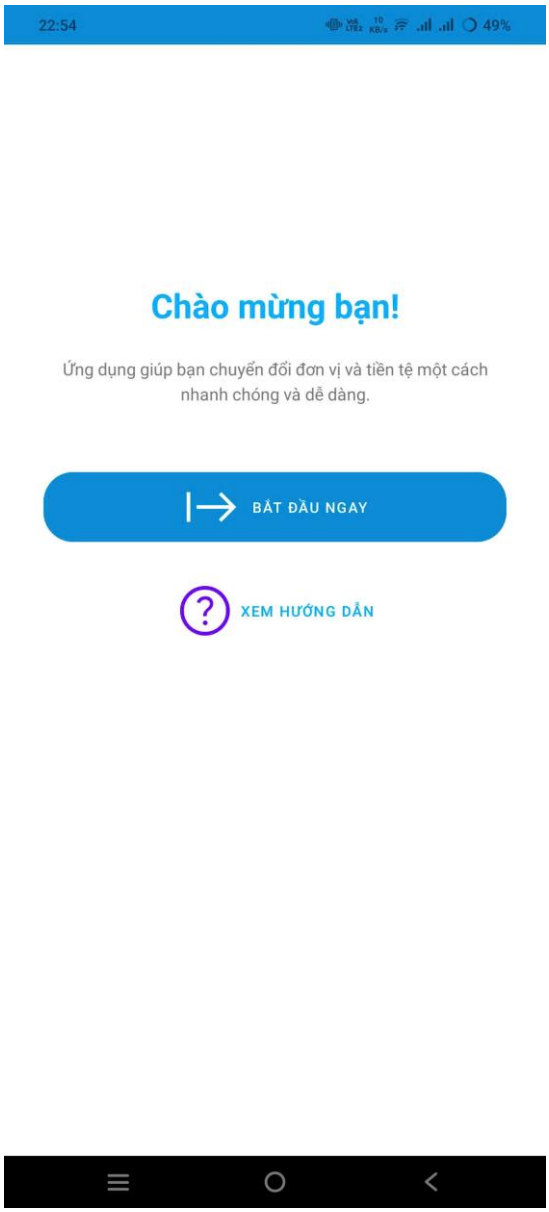
5. Kết quả

5.1 Màn hình Chờ (Splash Screen):

"Hình 5.1: Giao diện màn hình chờ của ứng dụng".

Giao diện màn hình chờ, xuất hiện đầu tiên khi người dùng khởi chạy ứng dụng. Màn hình này hiển thị logo và tên ứng dụng trong một khoảng thời gian ngắn (khoảng 2 giây) trước khi tự động chuyển sang màn hình chào mừng. Màu nền xanh chủ đạo và chữ trắng tạo cảm giác thân thiện và dễ nhận diện thương hiệu."

5.2 Màn hình Chào mừng (Welcome Screen):



Hình 5.2: Giao diện màn hình chào mừng người dùng.

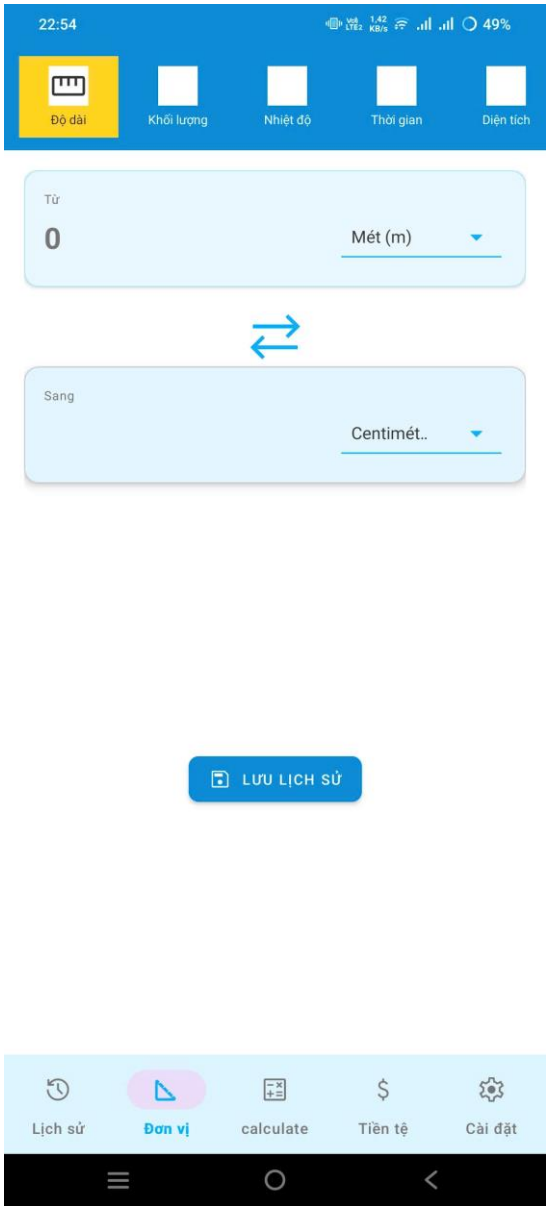
Thể hiện màn hình chào mừng với lời giới thiệu ngắn gọn về chức năng chính của ứng dụng. Nút 'Bắt đầu ngay' được thiết kế nổi bật, sử dụng màu xanh dương đậm, hướng người dùng vào trải nghiệm các tính năng của ứng dụng. Giao diện sử dụng nền trắng và các màu sắc đã được đồng bộ."

5.3 Màn hình Chính và Điều hướng (MainActivity với BottomNavigationView):

Hình 5.3: Giao diện màn hình chính với thanh điều hướng dưới.

Thanh điều hướng dưới cùng (BottomNavigationView) bao gồm các biểu tượng và nhãn cho các chức năng chính: Lịch sử, Đơn vị, Máy tính, Tiền tệ, và Cài đặt. Khi người dùng chọn một mục, nội dung tương ứng sẽ được hiển thị ở khu vực chính phía trên."

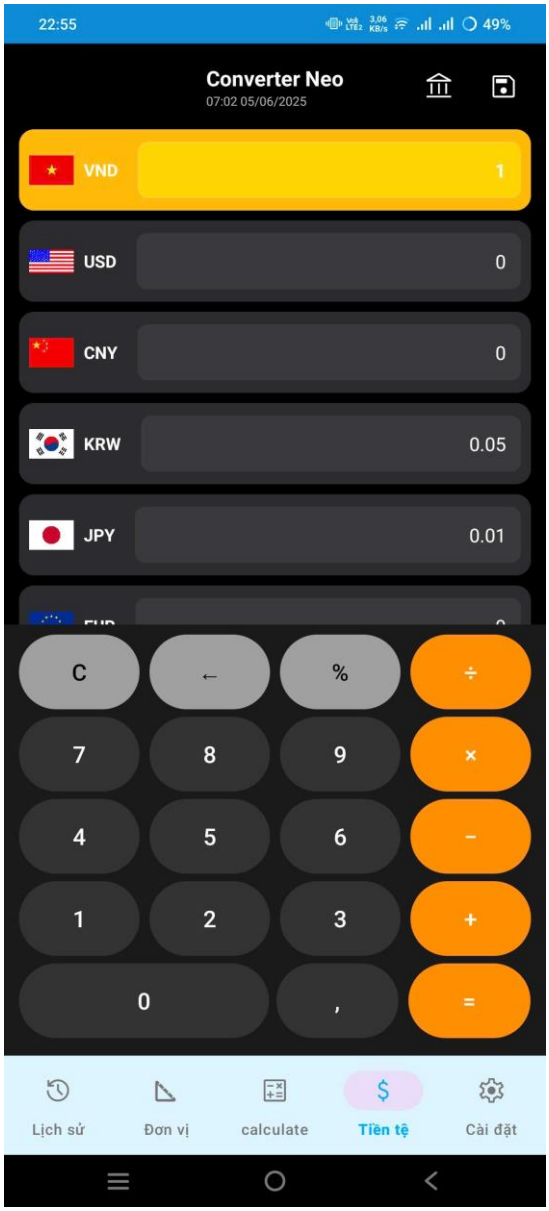
5.4 Màn hình Chuyển đổi Đơn vị (UnitConverterFragment):



Hình 5.4: Giao diện chức năng chuyển đổi đơn vị đo lường.

Người dùng có thể chọn loại đơn vị (Độ dài, Khối lượng, v.v.) từ thanh chọn ngang phía trên. Khu vực chính bao gồm hai CardView để nhập giá trị nguồn và hiển thị kết quả, cùng với Spinner để chọn đơn vị cụ thể. Nút hoán đổi đơn vị và nút lưu lịch sử cũng được bố trí tiện lợi."

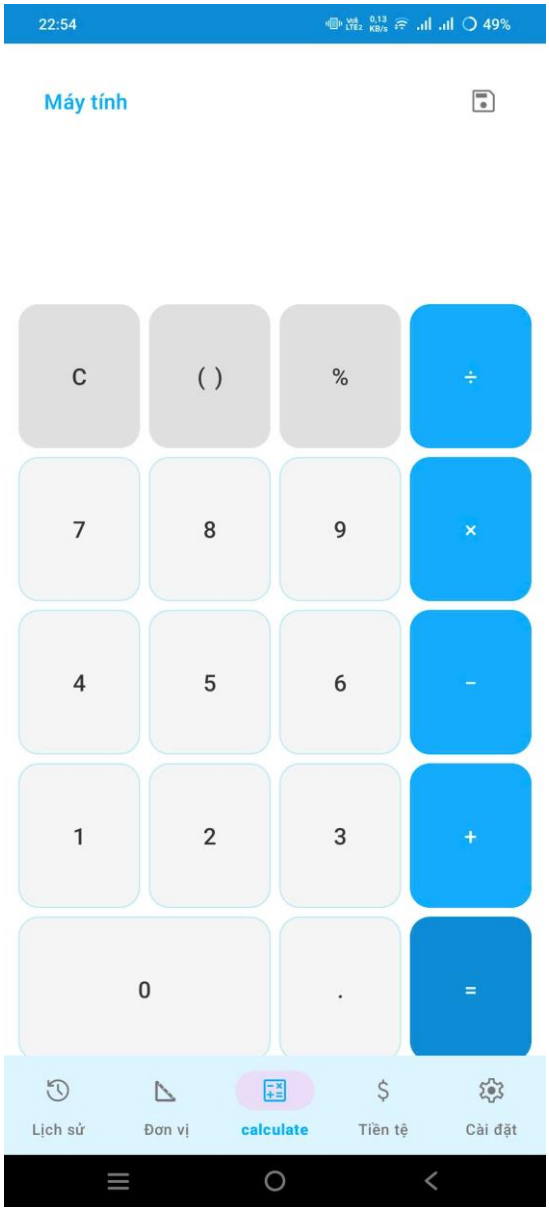
5.5 Màn hình Chuyển đổi Tiền tệ (CurrencyConverterFragment - Giao diện "Converter Neo"):



Hình 5.5: Giao diện chức năng chuyển đổi tiền tệ với tỷ giá cập nhật.

Danh sách các loại tiền tệ được hiển thị trong RecyclerView, mỗi hàng bao gồm cờ, mã tiền tệ và giá trị. Người dùng nhập liệu vào một đồng tiền cơ sở (ví dụ VND trong hình) thông qua bàn phím số tùy chỉnh ở dưới, và giá trị các đồng tiền khác sẽ tự động cập nhật theo tỷ giá lấy từ API. Thời gian cập nhật tỷ giá và nút làm mới, lưu lịch sử được bố trí ở phần tiêu đề."

5.6 Màn hình Máy tính (CalculatorFragment):



Hình 5.6: Giao diện chức năng máy tính cơ bản.

Phía trên là khu vực hiển thị biểu thức đang nhập và kết quả tính toán. Bên dưới là lưới các nút bấm số, phép toán cơ bản, và các nút chức năng như 'C' (Xóa), '%', '()'. Nút lưu lịch sử phép tính cũng được cung cấp.

5.7 Màn hình Lịch sử (HistoryFragment):

Caption: "Hình 5.7: Giao diện quản lý lịch sử chuyển đổi và tính toán."

Người dùng có thể xem lại các phép chuyển đổi và tính toán đã lưu. Mỗi mục lịch sử được hiển thị trên một CardView, bao gồm loại thao tác, chi tiết và thời gian. Thanh tìm kiếm ở trên cùng cho phép lọc nhanh các mục. Nút xóa tất cả lịch sử và icon xóa từng mục cũng được cung cấp."

5.8 Màn hình Cài đặt (SettingsFragment):

Caption: "Hình 5.8: Giao diện màn hình Cài đặt."

Hiện tại, màn hình này cung cấp tùy chọn 'Giới thiệu ứng dụng' để người dùng xem thông tin về sản phẩm. Giao diện được thiết kế đơn giản, dễ dàng mở rộng thêm các cài đặt khác trong tương lai."

6. Đánh giá và kết luận

6.1 Đánh giá kết quả:

Sau quá trình triển khai, dự án "Ứng dụng Chuyển đổi Đơn vị và Tiền tệ" đã cơ bản hoàn thành các mục tiêu và yêu cầu chính đã đề ra. Ứng dụng cung

cấp đầy đủ các chức năng cốt lõi bao gồm: chuyển đổi đa dạng các loại đơn vị đo lường (Độ dài, Khối lượng, Nhiệt độ, Thời gian, Diện tích); chuyển đổi các loại tiền tệ phổ biến với khả năng cập nhật tỷ giá tự động qua API; một máy tính cơ bản hỗ trợ các phép toán thông thường, dấu ngoặc và phần trăm; cùng với hệ thống lưu trữ và quản lý lịch sử các thao tác chuyển đổi và tính toán.

Giao diện người dùng đã được xây dựng theo hướng trực quan, dễ sử dụng, với việc áp dụng bảng màu đồng bộ và các cải tiến trong thiết kế của từng màn hình chức năng, đặc biệt là giao diện chuyển đổi tiền tệ và đơn vị. Các yêu cầu phi chức năng cơ bản về hiệu năng, độ tin cậy và tính dễ sử dụng cũng đã được chú trọng.

Nhìn chung, sản phẩm đã đáp ứng được vai trò là một công cụ tiện ích đa năng cho nhu cầu chuyển đổi và tính toán hàng ngày, đồng thời là kết quả của việc vận dụng các kiến thức và kỹ năng lập trình Android đã học. Mặc dù có thể có những hạn chế nhất định hoặc các tính năng nâng cao chưa được triển khai do giới hạn của một đồ án, nền tảng ứng dụng đã được xây dựng một cách tương đối hoàn chỉnh.

6.2 Kết luận:

Qua quá trình thực hiện dự án, ứng dụng "Chuyển đổi Đơn vị và Tiền tệ" đã được xây dựng thành công với các chức năng chính đáp ứng mục tiêu đề ra. Từ việc phân tích yêu cầu, thiết kế giao diện, cơ sở dữ liệu, đến triển khai code cho các module chuyển đổi đơn vị, tiền tệ (tích hợp API), máy tính và quản lý lịch sử, dự án đã giúp củng cố và áp dụng hiệu quả các kiến thức về lập trình ứng dụng Android. Sản phẩm cuối cùng là một công cụ di động tiện ích, có giao diện tương đối thân thiện và dễ sử dụng. Ứng dụng có thể được cải tiến và mở rộng thêm các tính năng như hỗ trợ thêm nhiều loại đơn vị và tiền tệ hơn, cho phép người dùng tùy chỉnh sâu hơn về giao diện (ví dụ như lựa chọn chủ đề, font chữ) cũng như thiết lập các đơn vị và tiền tệ ưu tiên thường dùng. Bên cạnh đó, chức năng máy tính có thể được nâng cấp để bao gồm các hàm tính toán khoa học phức tạp hơn. Một tính năng hữu ích khác có thể được xem xét là việc cung cấp biểu đồ hiển thị lịch sử biến động tỷ giá tiền tệ. Ngoài ra, việc hỗ trợ đa ngôn ngữ sẽ giúp ứng dụng tiếp cận được nhiều đối tượng người dùng hơn trên toàn cầu. Cuối cùng, việc liên tục tối ưu hóa hiệu năng và cải thiện trải

nghiệm người dùng dựa trên phản hồi thực tế cũng là một hướng đi quan trọng để ứng dụng ngày càng hoàn thiện.

7. Phụ lục

Mục Lục

Mã nguồn

Link github:

https://github.com/BaSiCo62132949/62132949-AndroidProgrammin/tree/main/BaiThiCK_62132949