## TRƯỜNG ĐẠI HỌC GIAO THÔNG VÂN TẢI

TP. HỒ CHÍ MINH

🙢✰🙠

A close up of a logo

Description automatically generated

BÀI BÁO CÁO ỨNG DỤNG MÁY TÍNH CẦM TAY

Giảng Viên Hướng Dẫn : Trương Quang Tuấn

Môn : LẬP TRÌNH THIẾT BỊ DI ĐỘNG

Thành Viên Thực Hiện:

Phạm Quang Long – 083205005448

Nguyễn Thị Yến Linh –089305001220

Mai Đình Huy-079205018385

Phạm Đình Thực-068205002809

*TP.HCM, ngày 7 tháng 7 năm 2025*

Phân công :

Phạm Đình Thực: Tester, Canva

Phạm Quang Long : Tester, Báo cáo

Nguyễn Thị Yến Linh : Developer

Mai Đình Huy : Figma, UI

Link Figma nhóm : https://www.figma.com/design/Ur4uAGPPOXFlHdTmfcVFex/calculator?m=dev&t=ba880KFuY86MEkfa-1

Link GitHub nhóm :

<https://github.com/yenlinh17>

Link GitHub cá nhân :

Phạm Đình Thực : https://github.com/Thuc2606

Mai Đình Huy : https://github.com/MaiDinhHuy1303

Nguyễn Thị Yến Linh: <https://github.com/yenlinh17>

Phạm Quang Long: <https://github.com/quanglong1>

Mục lục

[LỜI MỞ ĐẦU 1](#_Toc202917335)

[1.1. Bối cảnh thực tiễn 1](#_Toc202917336)

[1.2. Lý do chọn đề tài 1](#_Toc202917337)

[1.3. Mục tiêu của đề tài 2](#_Toc202917338)

[1.4. Phạm vi 2](#_Toc202917339)

[II. PHÂN TÍCH YÊU CẦU HỆ THỐNG 2](#_Toc202917340)

[2.1. Đối tượng sử dụng 2](#_Toc202917341)

[2.2. Tính năng chính 3](#_Toc202917342)

[2.3. sơ đồ Use Case 5](#_Toc202917343)

[III. THIẾT KẾ HỆ THỐNG 6](#_Toc202917344)

[3.1. Giao diện người dùng 6](#_Toc202917345)

[3.2. Cấu trúc mã nguồn 8](#_Toc202917347)

[3.3. Sơ đồ hoạt động 10](#_Toc202917348)

[IV. TRIỂN KHAI VÀ MÃ NGUỒN 11](#_Toc202917349)

[4.1 Hàm CalculatorScreen.kt 11](#_Toc202917350)

[4.1.1 Sơ đồ use case : 12](#_Toc202917351)

[4.2 Hàm CalculatorButton.kt (dùng để tạo nút bấm ): 13](#_Toc202917352)

[4.2.1. Cách hoạt động trong ứng dụng 13](#_Toc202917353)

[4.2.2. Chức năng : 14](#_Toc202917354)

[4.2.3 Sơ đồ use case: 15](#_Toc202917355)

[4.3 Hàm CalculatorLogic.kt : 15](#_Toc202917357)

[4.3.1 Chức năng: 15](#_Toc202917358)

[4.3.2 Sơ đồ use case : 16](#_Toc202917359)

[4.4 Hàm DataStoreHelp.kt : 17](#_Toc202917360)

[4.4.1 Chức năng : 17](#_Toc202917361)

[4.4.2 Sơ đò use case : 19](#_Toc202917362)

[4.5 Hàm Main ( đây là phần khởi tạo ): 20](#_Toc202917364)

[V. KẾT QUẢ THỰC HIỆN VÀ KIỂM THỬ 22](#_Toc202917365)

[5.1. Kiểm thử chức năng: 22](#_Toc202917366)

[5.2. Giao diện thử nghiệm 23](#_Toc202917367)

[VI. KẾT LUẬN 24](#_Toc202917369)

[6.1. Ưu điểm 24](#_Toc202917370)

[6.2. Hạn chế 25](#_Toc202917371)

[VII. TÀI LIỆU THAM KHẢO 26](#_Toc202917372)

# LỜI MỞ ĐẦU

## 1.1. Bối cảnh thực tiễn

Trong thời đại công nghệ 4.0, smartphone trở thành thiết bị cá nhân không thể thiếu. Người dùng hiện đại có xu hướng tích hợp mọi tiện ích cần thiết vào một thiết bị duy nhất – từ liên lạc, giải trí, học tập đến công cụ làm việc. Trong đó, **ứng dụng máy tính cầm tay** là một công cụ phổ thông, xuất hiện hầu hết trên mọi thiết bị Android.

Thay vì sử dụng máy tính cầm tay vật lý hoặc ứng dụng hệ thống mặc định, người dùng hoàn toàn có thể cài đặt ứng dụng máy tính do chính mình hoặc cộng đồng phát triển. Việc xây dựng một ứng dụng như vậy sẽ giúp sinh viên hiểu sâu hơn về kỹ thuật lập trình Android, quản lý trạng thái, xử lý UI bằng Jetpack Compose – một công nghệ hiện đại đang dần thay thế XML.

## 1.2. Lý do chọn đề tài

Việc lựa chọn xây dựng ****ứng dụng máy tính cầm tay**** nhằm:

- Tạo điều kiện thực hành kỹ năng lập trình Kotlin.

- Tiếp cận phương pháp xây dựng giao diện bằng Jetpack Compose.

- Vận dụng tư duy lập trình để xử lý biểu thức toán học.

- Làm quen với quy trình phát triển phần mềm từ thiết kế → triển khai → kiểm thử.

Mặc dù là một ứng dụng đơn giản, nhưng đề tài này thể hiện đầy đủ các yếu tố của một ứng dụng Android: UI, logic xử lý, trạng thái, sự kiện và trải nghiệm người dùng.

## 1.3. Mục tiêu của đề tài

* Xây dựng một ứng dụng máy tính cơ bản, thân thiện với người dùng và dễ thao tác.
* Thiết kế toàn bộ giao diện người dùng bằng thư viện **Jetpack Compose**, đảm bảo tính hiện đại, linh hoạt và dễ bảo trì.
* Hỗ trợ đầy đủ các phép toán số học thông dụng như **cộng (+)**, **trừ (−)**, **nhân (×)** và **chia (÷)**.
* Cho phép người dùng nhập biểu thức trực tiếp thông qua bàn phím ảo và thực hiện xử lý để **trả về kết quả chính xác**.
* Tối ưu hóa **trải nghiệm người dùng (UX)** thông qua bố cục rõ ràng, dễ nhìn, thao tác mượt mà và phản hồi nhanh chóng.
* Mã nguồn được tổ chức có hệ thống, tách biệt rõ giữa giao diện và logic xử lý, giúp dễ dàng **bảo trì và mở rộng trong tương lai**.

## 1.4. Phạm vi

Đề tài tập trung xây dựng một ứng dụng máy tính cơ bản, với các giới hạn và phạm vi cụ thể như sau:

* **Phép toán được hỗ trợ:**Gồm bốn phép toán số học cơ bản là**cộng (+), trừ (−), nhân (×)**và**chia (÷).**
* **Biểu thức hợp lệ:**Chỉ hỗ trợ các biểu thức gồm**số và phép toán**liên tiếp nhau,**không bao gồm dấu ngoặc hoặc các phép toán nâng cao.**
* **Nền tảng phát triển:**Ứng dụng được xây dựng trên**Android Studio**, sử dụng**ngôn ngữ lập trình Kotlin** và **thư viện giao diện Jetpack Compose**– công nghệ hiện đại thay thế cho XML truyền thống trong thiết kế giao diện Android.

# II. PHÂN TÍCH YÊU CẦU HỆ THỐNG

## 2.1. Đối tượng sử dụng

Ứng dụng được thiết kế hướng đến các nhóm người dùng sau:

- Sinh viên, học sinh:  
Những người cần một công cụ tính toán đơn giản, nhanh chóng để phục vụ học tập hoặc thực hiện các phép tính cơ bản hằng ngày.

- Người dùng điện thoại Android phổ thông:  
Người dùng đại chúng có nhu cầu sử dụng một ứng dụng máy tính nhẹ, dễ dùng, không yêu cầu cấu hình cao hoặc kết nối Internet.

- Người học lập trình Android:  
Những người mới học lập trình có thể tham khảo ứng dụng như một ví dụ thực tế về cách xây dựng giao diện bằng Jetpack Compose và xử lý logic trong Kotlin.

## 2.2. Tính năng chính

Ứng dụng Máy Tính Cầm Tay

Ứng dụng máy tính cầm tay được xây dựng với các chức năng cơ bản và mở rộng, đáp ứng nhu cầu tính toán đơn giản đến nâng cao của người dùng. Cụ thể như sau:

Hiển thị giao diện:

-Một cửa sổ chính hiển thị trên màn hình với giao diện trực quan và dễ sử dụng.

-Ô nhập liệu ở phía trên cho phép người dùng nhập biểu thức toán học.

-Lịch sử tính toán hiển thị phía dưới, giúp theo dõi các phép tính đã thực hiện.

-Nút menu để mở tìm kiếm và chuyển đổi giữa các chế độ tính toán.

Nút số và Phép Toán:

* Các nút số: 0 đến 9.
* Các nút phép toán cơ bản: +, -, ×, ÷.
* Các nút chức năng:
* ‘=’: để thực hiện tính toán và hiển thị kết quả.
* ‘C’: để xóa toàn bộ nội dung đã nhập.
* ‘⌫’ để xóa từng ký tự.
* ‘±’: để đổi dấu số.
* ‘.’: để nhập số thập phân.
* Các nút toán học nâng cao: sin, cos, tan, log, ln, x², √, %, π, e, !, (, ).

Xử lý biểu thức và tính toán:

* Khi người dùng nhấn các nút số hoặc phép toán, giá trị tương ứng sẽ được thêm vào chuỗi biểu thức trong ô nhập liệu.
* Khi nhấn =, ứng dụng sẽ phân tích biểu thức và thực hiện tính toán theo đúng quy tắc toán học, sau đó hiển thị kết quả.
* Nếu biểu thức sai cú pháp hoặc có lỗi (như chia cho 0), ứng dụng sẽ thông báo lỗi rõ ràng.
* Cho phép người dùng chỉnh sửa biểu thức dễ dàng với nút ⌫ (xóa từng ký tự) hoặc C (xóa toàn bộ).

Lịch sử tính toán:

-Ứng dụng lưu lại toàn bộ các phép tính đã thực hiện, bao gồm biểu thức, kết quả và thời gian.

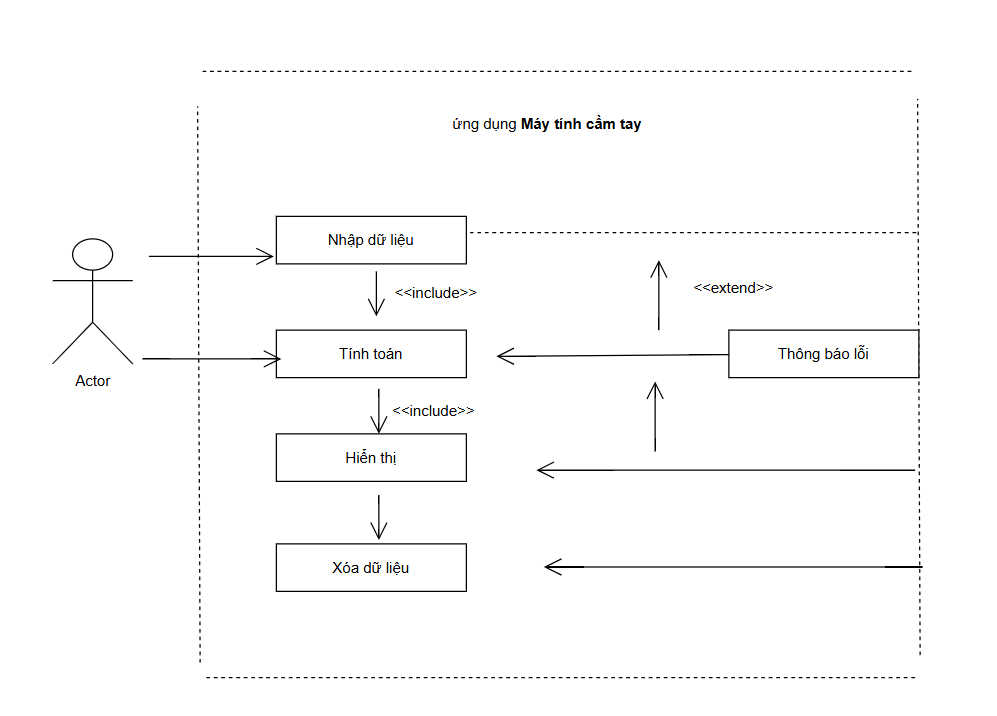
-Người dùng có thể:

* Xem lại lịch sử theo thứ tự thời gian.
* Tìm kiếm biểu thức hoặc kết quả theo từ khóa.
* Xoá từng phép tính hoặc toàn bộ lịch sử nếu cần.
* Nhấn vào mục trong lịch sử để đưa lại biểu thức vào ô nhập và tính tiếp.

Chế độ nâng cao & tuỳ chọn:

* Cho phép bật/tắt bàn phím máy tính.
* Hiển thị thêm các phím chức năng toán học nâng cao khi cần.
* Có thể tuỳ chỉnh giao diện để tập trung vào tính năng người dùng đang quan tâm.

## 2.3. sơ đồ Use Case



**Tóm tắt hoạt động của sơ đồ:**

1. **Người dùng** bắt đầu bằng cách **Nhập dữ liệu** (số và phép toán) vào máy tính.
2. Sau khi dữ liệu được nhập (và có thể là nhấn nút "="), quá trình **Tính toán** được kích hoạt.
3. Quá trình **Tính toán** **luôn luôn** dẫn đến việc **Hiển thị** kết quả.
4. Nếu trong quá trình **Tính toán** hoặc (có thể là) **Nhập dữ liệu** phát sinh lỗi, thì chức năng **Thông báo lỗi** sẽ được kích hoạt để hiển thị thông báo tương ứng.
5. **Người dùng** có thể **Xóa dữ liệu** bất cứ lúc nào.

# 

# III. THIẾT KẾ HỆ THỐNG

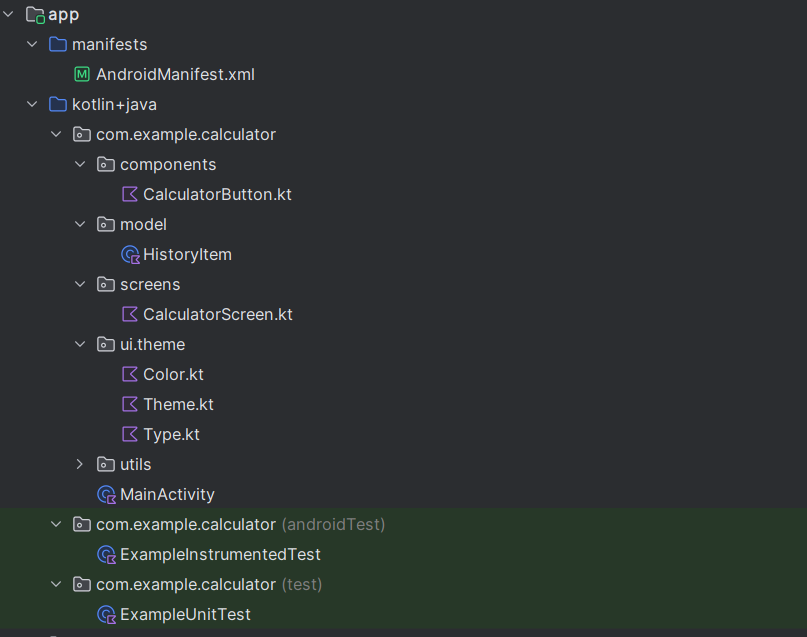
## 3.1. Giao diện người dùng

## Bố cục giao diện được chia theo lưới (grid layout), với các phím bấm được sắp xếp khoa học và hợp lý. Việc phân chia các nhóm chức năng rõ ràng giúp người dùng dễ thao tác và giảm thiểu sai sót khi nhập biểu thức.

* Ở phần trên cùng, ứng dụng có thanh trạng thái hệ thống Android hiển thị thông tin như thời gian, kết nối mạng, pin,... Phía bên trái có biểu tượng ba gạch ngang (menu hamburger), có thể dẫn đến các mục tùy chọn như lịch sử tính toán, đổi chủ đề,... Phía bên phải là biểu tượng bánh răng (cài đặt), giúp người dùng thay đổi các tùy chỉnh liên quan đến giao diện, âm thanh hoặc độ chính xác.
* Ngay bên dưới là khu vực các nút chức năng chính của máy tính. Nhóm đầu tiên bao gồm các phím điều khiển như “C” để xóa toàn bộ biểu thức, “⌫” để xóa một ký tự, “±” để đổi dấu âm/dương, và dấu chia “/”. Đây là những chức năng cơ bản giúp người dùng xử lý biểu thức đầu vào một cách linh hoạt.
* Tiếp theo là nhóm các nút số từ 0 đến 9, cùng với dấu chấm (.) để nhập số thập phân, và dấu ngoặc đơn để xử lý các biểu thức toán học phức tạp. Những nút này được đặt ở vị trí trung tâm của giao diện và có màu xám đậm, giúp dễ phân biệt với các phép toán.
* Nhóm tiếp theo là các phép toán cơ bản như cộng “+”, trừ “-”, nhân “\*”, và dấu bằng “=” dùng để cho ra kết quả. Các nút phép toán này được tô màu cam nổi bật, giúp người dùng nhanh chóng xác định và thao tác chính xác. Riêng dấu bằng “=” còn có vai trò quyết định việc thực hiện tính toán và chuyển đổi trạng thái biểu thức sang kết quả.
* Điểm đặc biệt của giao diện là phần cuối cùng chứa các hàm toán học nâng cao như căn bậc hai, bình phương, phần trăm, cùng với các hàm lượng giác sin, cos, tan, các hàm logarit như log, ln, và các hằng số toán học như π (pi) và e. Ngoài ra còn có phép toán giai thừa (!). Những nút này giúp mở rộng phạm vi tính toán của ứng dụng, không chỉ dừng lại ở các phép cộng trừ nhân chia mà còn phục vụ cho các yêu cầu toán học nâng cao như giải bài tập phổ thông hoặc kiểm tra nhanh biểu thức trong các môn khoa học tự nhiên.
* Màu sắc trong giao diện được sử dụng hợp lý: màu cam dành cho các thao tác quan trọng hoặc phép toán chính, màu xám cho các chức năng phụ trợ và số học cơ bản. Kích thước các nút được thiết kế vừa đủ lớn để thao tác bằng tay trên màn hình cảm ứng, giúp tránh nhấn nhầm.

## 

## 3.2. Cấu trúc mã nguồn



**Các thành phần chính của dự án:**

App: Là mô-đun ứng dụng chính của project. Thư mục này chứa toàn bộ mã nguồn, tài nguyên và logic xử lý của ứng dụng máy tính cầm tay.

manifests/AndroidManifest.xml :  
Khai báo các thành phần chính của ứng dụng (như MainActivity) và các quyền mà ứng dụng cần sử dụng.

java (hoặc kotlin)/com.example.calculator:  
Thư mục chứa mã nguồn chính viết bằng Kotlin. Được chia thành nhiều package nhỏ, mỗi package đảm nhiệm một vai trò cụ thể:

components:   
Chứa các thành phần giao diện người dùng có thể tái sử dụng. Trong dự án này có file CalculatorButton.kt, định nghĩa cấu trúc nút máy tính (ví dụ: số, phép toán, hàm toán học), được dùng nhiều lần trong toàn bộ giao diện.

Model:   
Chứa các lớp dữ liệu (data model). Ở đây có HistoryItem.kt, định nghĩa cấu trúc của một biểu thức đã tính, bao gồm biểu thức gốc và kết quả. Thường dùng để hiển thị lịch sử tính toán.

Screens:  
Chứa các màn hình giao diện chính của ứng dụng. File CalculatorScreen.kt là nơi xây dựng toàn bộ bố cục UI chính của máy tính bằng Jetpack Compose, bao gồm màn hình hiển thị kết quả, bàn phím các phép toán, số và hàm toán học.

ui.theme:   
Cấu hình chủ đề giao diện người dùng. Gồm:

Color.kt: định nghĩa mã màu được sử dụng trong ứng dụng.

Theme.kt: cài đặt theme chính bằng Material3 hoặc MaterialTheme.

Type.kt: quản lý kiểu chữ và kích thước văn bản.

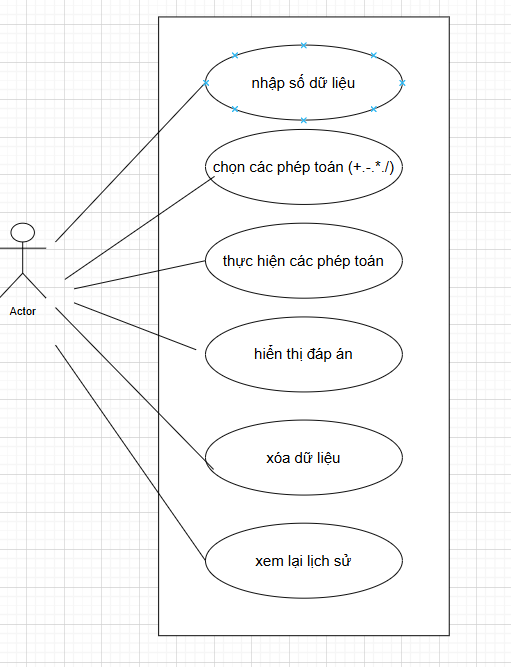
Utils: Chứa các lớp tiện ích hoặc các phần logic riêng biệt không thuộc về UI. Trong ảnh chỉ có MainActivity.kt, nhưng bạn có thể thêm các file như CalculatorLogic.kt, DataStoreHelper.kt vào đây nếu có xử lý logic hoặc lưu trữ.

MainActivity.kt  
Là điểm khởi đầu của ứng dụng. File này gọi setContent {} để hiển thị CalculatorScreen, đồng thời khởi tạo các thành phần cần thiết khi app chạy.

com.example.calculator (androidTest)  
Chứa các bài kiểm thử tích hợp (instrumented tests) chạy trên thiết bị thật hoặc giả lập. File ExampleInstrumentedTest.kt có thể kiểm tra xem UI hoặc logic app hoạt động đúng trên Android không.

com.example.calculator (test)  
Chứa các bài kiểm thử đơn vị (unit test) chạy trên JVM. File ExampleUnitTest.kt dùng để test các hàm logic thuần túy như tính biểu thức, xử lý phép toán,...

## 3.3. Sơ đồ hoạt động



- Sơ đồ trên mô tả luồng hoạt động chính của ứng dụng máy tính cầm tay từ khi người dùng mở ứng dụng cho đến khi nhập biểu thức, nhận kết quả và lưu lịch sử. Sau khi người dùng thực hiện thao tác tính toán, hệ thống sẽ xử lý biểu thức đầu vào bằng thư viện toán học và hiển thị kết quả. Nếu phép tính có lỗi (như cú pháp sai, chia cho 0), hệ thống sẽ phản hồi bằng thông báo lỗi. Kết quả hợp lệ được lưu trữ vào bộ nhớ cục bộ (DataStore), giúp người dùng có thể xem lại lịch sử các phép tính đã thực hiện trước đó. Sơ đồ thể hiện rõ ràng quy trình hoạt động và khả năng phản hồi của hệ thống trong từng tình huống.

# IV. TRIỂN KHAI VÀ MÃ NGUỒN

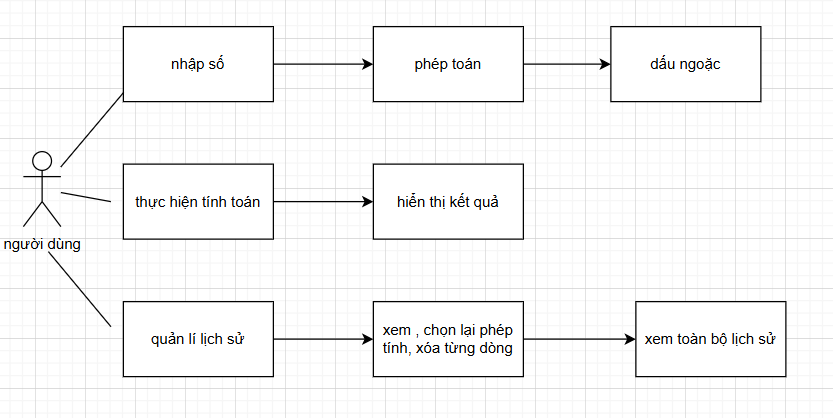
## 4.1 Hàm CalculatorScreen.kt (đây là hàm giao diện chính trong ứng dụng):

CalculatorScreen.kt là thành phần trung tâm xây dựng toàn bộ giao diện và chức năng của màn hình máy tính trong ứng dụng. Màn hình này được tạo bằng Jetpack Compose, đảm bảo tính hiện đại, dễ mở rộng và thân thiện với người dùng.

*Các chức năng chính:*

* Hiển thị biểu thức đang nhập: Ở phần đầu màn hình, một dòng Text lớn hiển thị biểu thức toán học mà người dùng đang nhập. Biểu thức này sẽ được cập nhật liên tục khi người dùng nhấn nút.
* Hiển thị lịch sử phép tính: Sử dụng LazyColumn, ứng dụng hiển thị các phép tính đã thực hiện, bao gồm biểu thức và kết quả, cùng thời gian thực hiện. Các kết quả này được tải từ bộ nhớ cục bộ bằng DataStoreHelper.
* Cho phép người dùng thao tác với lịch sử
* Người dùng có thể chạm vào một dòng để sử dụng lại biểu thức đó.
* Có thể xóa từng dòng lịch sử bằng biểu tượng ❌.
* Có thể xóa toàn bộ lịch sử bằng biểu tượng 🗑️ ở cuối danh sách.  
  Bàn phím được chia thành 2 phần:
* Bàn phím cơ bản: gồm các số (0-9), phép tính (+ - \* /), căn bậc hai √, bình phương x², phần trăm %, dấu ngoặc, dấu ±, C, ⌫ và =.
* Bàn phím nâng cao (ẩn/hiện bằng nút cài đặt): gồm các hàm lượng giác (sin, cos, tan), logarit (log, ln), hằng số (π, e) và giai thừa !.
* Xử lý tính toán và lưu lịch sử  
  Khi nhấn dấu =, hệ thống kiểm tra biểu thức:
* Nếu sai cú pháp → hiển thị "Lỗi cú pháp".
* Nếu hợp lệ → tính kết quả bằng evaluateExpression, sau đó:
* Hiển thị kết quả.
* Tạo HistoryItem và lưu vào danh sách history.
* Lưu vào bộ nhớ qua DataStoreHelper.saveHistory(...).

### 4.1.1 Sơ đồ use case :

****

#### a. **Nhập số → Phép toán → Dấu ngoặc**

* Người dùng bắt đầu quá trình nhập liệu bằng cách chọn các **chữ số (0–9).**
* Sau đó, có thể chọn **các phép toán** như cộng, trừ, nhân, chia,...
* Cuối cùng, có thể sử dụng **dấu ngoặc ((), …)** để tạo biểu thức phức tạp.
* Đây là **giai đoạn xây dựng biểu thức toán học.**

#### b. **Thực hiện tính toán → Hiển thị kết quả**

* Sau khi nhập xong biểu thức, người dùng nhấn nút =.
* Ứng dụng sẽ **xử lý biểu thức**, kiểm tra cú pháp, thực hiện phép toán.
* Kết quả được **hiển thị trực tiếp trên màn hình**.
* Nếu biểu thức lỗi (ví dụ chia cho 0), hệ thống sẽ báo lỗi rõ ràng.

#### c. **Quản lý lịch sử → Xem, chọn lại, xóa từng dòng → Xem toàn bộ lịch sử**

* Sau mỗi phép tính, biểu thức và kết quả sẽ được **lưu lại trong hệ thống.**
* Người dùng có thể **xem lại từng dòng, chọn lại** để tái sử dụng, hoặc**xóa một dòng riêng lẻ** nếu cần.
* Ngoài ra, người dùng có thể mở **toàn bộ lịch sử phép tính** trong một màn hình riêng.
* Tính năng này sử dụng **DataStore để lưu trữ cục bộ**, đảm bảo dữ liệu vẫn được giữ lại ngay cả khi đóng ứng dụng.

## 4.2 Hàm CalculatorButton.kt (dùng để tạo nút bấm ):

### ****4.2.1.**** Cách hoạt động trong ứng dụng

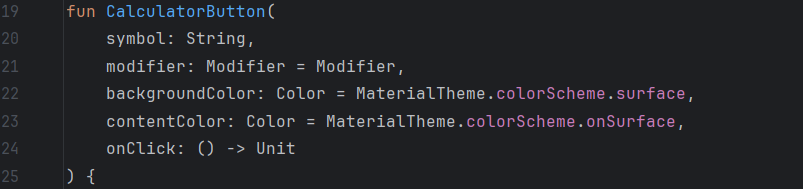
Hiển thị: Các nút được hiển thị theo lưới hoặc hàng trong giao diện bàn phím.

Tương tác người dùng: Khi người dùng nhấn vào nút (ví dụ "+"), sự kiện click sẽ được gửi tới logic xử lý.

Xử lý sự kiện: Logic xử lý (thường nằm trong ViewModel hoặc CalculatorScreen.kt) sẽ cập nhật biểu thức hoặc tính toán.

Cập nhật UI: Kết quả được hiển thị lại trên màn hình máy tính.

### 4.2.2. Chức năng :

****

Khai báo một hàm giao diện (Composable) để tạo nút bấm máy tính, có thể tùy chỉnh ký hiệu hiển thị, màu nền, màu chữ, và xử lý khi người dùng nhấn nút.

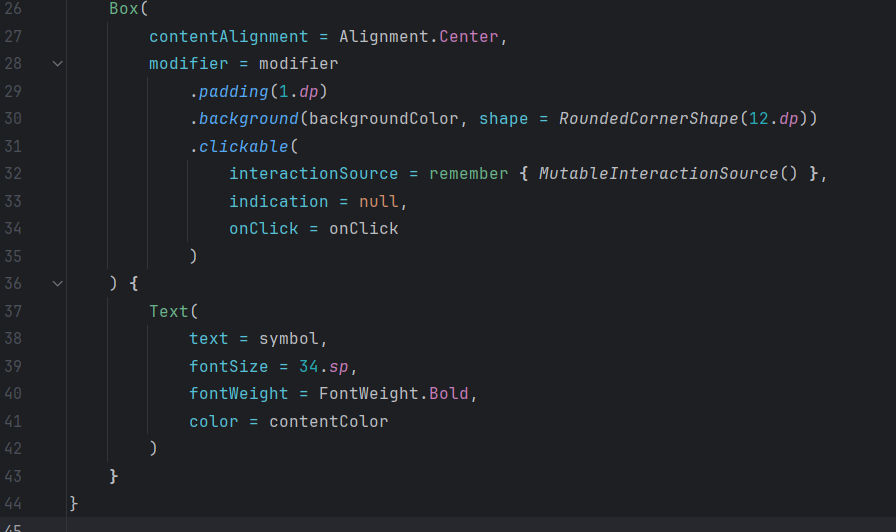
- symbol: nội dung hiển thị trên nút.

- modifier: cho phép thay đổi bố cục từ bên ngoài.

- backgroundColor: chỉ định màu nền nút.

- contentColor: chỉ định màu chữ trên nút.

- onClick: hàm xử lý khi nút được nhấn.

****

- Tạo giao diện nút bấm với:

+ Căn giữa nội dung (Alignment.Center)

+ Bo góc, màu nền, và padding cho nút

+ Khả năng nhấn để thực thi hành động (onClick)

+ Hiển thị ký hiệu ở giữa nút với cỡ chữ lớn, màu chữ, và in đậm

### 4.2.3 Sơ đồ use case:

# 

## 4.3 Hàm CalculatorLogic.kt :

### 4.3.1 Chức năng:

a. Hàm handleInput(expression: String, input: String): String

- Mục đích: Hàm này đóng vai trò bộ điều khiển chính, tiếp nhận và xử lý tương tác từ người dùng với các nút bấm.

- Cơ chế hoạt động: Sử dụng cấu trúc when để xử lý linh hoạt từng loại đầu vào:

* Tính toán (=): Gọ evaluateExpression, dùng try-catch để bắt lỗi như chia cho 0 hoặc cú pháp sai. Trả về "Lỗi" nếu có lỗi.
* Xóa (C, ⌫): "C" xóa toàn bộ, "⌫" xóa 1 ký tự cuối.
* Đổi dấu (±): Tìm số cuối dùng regex, đổi dấu (+ -> -, hoặc ngược lại).
* Hàm toán học (√, x², %): Biến đổi chuỗi theo định dạng có thể tính bởi thư viện (vd: sqrt(), ^2).
* Ký tự bình thường: Gắn thêm vào cuối chuỗi biểu thức.

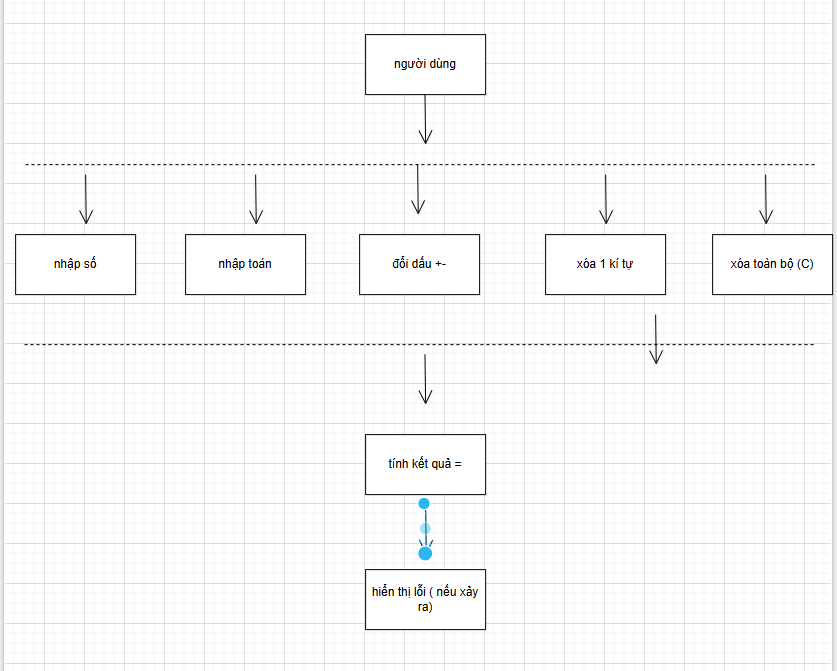
b. Hàm evaluateExpression(expr: String): Double

* Mục đích: Tính toán giá trị của chuỗi biểu thức.
* Cơ chế hoạt động: Sử dụng thư viện net.objecthunter.exp4j để build biểu thức và tính toán bằng .evaluate().

c. Hàm isValidExpression(expression: String): Boolean

* Mục đích: Kiểm tra biểu thức có hợp lệ hay không.
* Cơ chế : Dùng ExpressionBuilder(...).build().evaluate() trong khối try-catch để bắt lỗi. Trả về true nếu hợp lệ.

### 4.3.2 Sơ đồ use case :

****

Giải thích sơ đồ Use Case:

Sơ đồ Use Case thể hiện các hành vi chính mà người dùng có thể thực hiện trong quá trình tương tác với ứng dụng máy tính. Cụ thể, người dùng có thể:

* Nhập số (từ 0 đến 9) để hình thành biểu thức cần tính.
* Nhập các phép toán như cộng, trừ, nhân, chia, phần trăm, căn bậc hai hoặc bình phương – giúp biểu thức trở nên đa dạng và linh hoạt hơn.
* Đổi dấu số (±), cho phép thay đổi số dương thành âm và ngược lại, giúp xử lý các phép toán liên quan đến số âm.
* Xóa 1 ký tự (⌫) khi nhập sai từng phần của biểu thức.
* Xóa toàn bộ (C) để nhập lại từ đầu trong trường hợp cần làm mới toàn bộ biểu thức.
* Tính kết quả (=) sau khi đã hoàn thiện biểu thức. Hệ thống sẽ xử lý chuỗi toán học thông qua thư viện ExpressionBuilder và trả kết quả.
* Nếu biểu thức có lỗi như cú pháp sai, chia cho 0 hoặc thiếu toán hạng, hệ thống sẽ hiển thị thông báo lỗi (thông qua chuỗi “lỗi”) nhằm đảm bảo phản hồi rõ ràng cho người dùng.

Sơ đồ không chỉ thể hiện các chức năng chính của ứng dụng, mà còn cho thấy sự liên kết giữa người dùng và hệ thống, minh họa cách mà hệ thống phản hồi lại các thao tác. Đây là một phần quan trọng trong quá trình phân tích và thiết kế hệ thống, đảm bảo ứng dụng thân thiện và dễ sử dụng đối với người dùng cuối.

## 4.4 Hàm DataStoreHelp.kt :

### 4.4.1 Chức năng :

Tệp DataStoreHelper.kt trong dự án đảm nhiệm vai trò quản lý lưu trữ và truy xuất dữ liệu lịch sử phép tính mà người dùng đã thực hiện. Đây là một thành phần quan trọng giúp mở rộng chức năng của ứng dụng, cho phép người dùng xem lại các phép tính trước đó, từ đó nâng cao trải nghiệm sử dụng và đáp ứng nhu cầu tra cứu lại thông tin một cách thuận tiện.

Trong tệp này, nhóm lập trình đã sử dụng Jetpack DataStore – một thư viện mới do Google phát triển, thay thế cho SharedPreferences truyền thống. DataStore hỗ trợ lưu trữ dữ liệu dạng khóa-giá trị một cách an toàn, bất đồng bộ, không bị rò rỉ bộ nhớ và hoạt động tốt trong các luồng coroutine.

Cụ thể, đoạn mã khai báo một biến mở rộng Context.dataStore sử dụng preferencesDataStore(name = "calculator\_history"). Đây là nơi định nghĩa một kho lưu trữ dữ liệu có tên là "calculator\_history", nơi toàn bộ lịch sử phép tính sẽ được ghi vào.

Bên trong object DataStoreHelper, có hai hàm chính:

*1. saveHistory(context: Context, historyList: List<HistoryItem>)*

* Đây là hàm chịu trách nhiệm lưu danh sách các phép tính (được biểu diễn dưới dạng danh sách các đối tượng HistoryItem) vào bộ nhớ cục bộ thông qua DataStore. Cụ thể:
* Hàm sẽ sử dụng thư viện Gson để chuyển đổi danh sách historyList thành một chuỗi JSON.
* Sau đó, chuỗi JSON này sẽ được lưu vào DataStore tại khóa "history".
* Việc lưu trữ diễn ra bất đồng bộ thông qua coroutine, đảm bảo không chặn luồng chính, từ đó không làm gián đoạn trải nghiệm người dùng.
* Hàm này thường được gọi sau khi người dùng nhấn nút = để tính toán, nhằm lưu lại biểu thức và kết quả vừa thực hiện.

*2. loadHistory(context: Context): List<HistoryItem>*

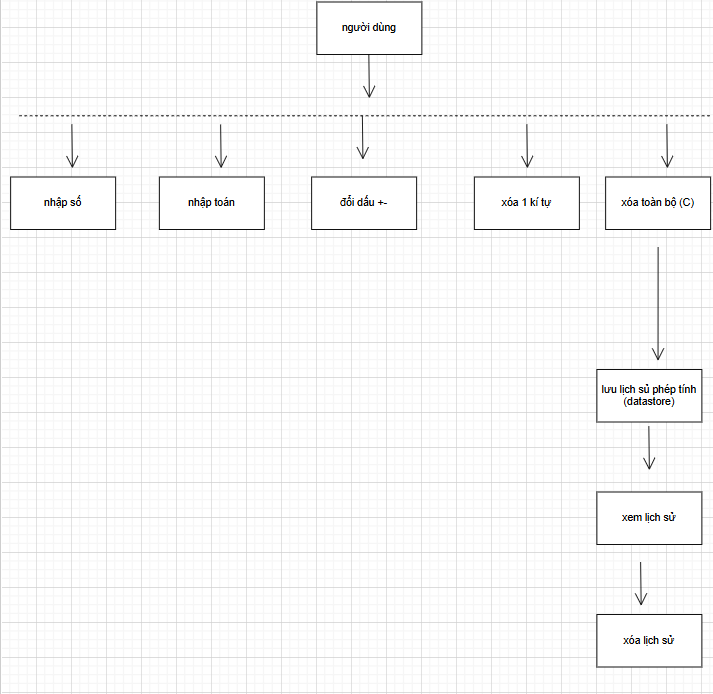
* Hàm này được sử dụng để truy xuất danh sách lịch sử đã được lưu trước đó từ DataStore. Quy trình hoạt động như sau:
* Truy xuất dữ liệu dưới khóa "history" từ DataStore.
* Nếu dữ liệu rỗng hoặc không tồn tại, hàm trả về danh sách rỗng.
* Nếu có dữ liệu, hàm sẽ dùng Gson để chuyển đổi chuỗi JSON đó trở lại thành một danh sách các đối tượng HistoryItem.
* Kết quả được trả về dưới dạng List<HistoryItem> để hiển thị trên giao diện lịch sử.
* Hàm này thường được gọi khi người dùng chuyển sang màn hình xem lịch sử phép tính, giúp ứng dụng hiển thị đầy đủ các phép tính đã thực hiện trước đó, ngay cả khi người dùng đã thoát ứng dụng trước đó.

***Ý nghĩa và vai trò trong ứng dụng :***

* Việc sử dụng DataStoreHelper.kt giúp ứng dụng máy tính mở rộng chức năng vượt ra khỏi giới hạn của một máy tính cơ bản. Thay vì chỉ xử lý biểu thức và trả về kết quả tức thì, ứng dụng còn ghi nhớ lịch sử tính toán, tạo điều kiện cho người dùng có thể:
* Xem lại kết quả cũ mà không cần tính lại.
* Đối chiếu, kiểm tra các phép toán đã thực hiện.
* Tránh lặp lại thao tác nếu chẳng may đóng ứng dụng.
* Tính năng này đặc biệt hữu ích trong môi trường học tập, nghiên cứu hoặc công việc văn phòng, nơi người dùng thường xuyên phải thực hiện và lưu lại nhiều phép tính khác nhau.

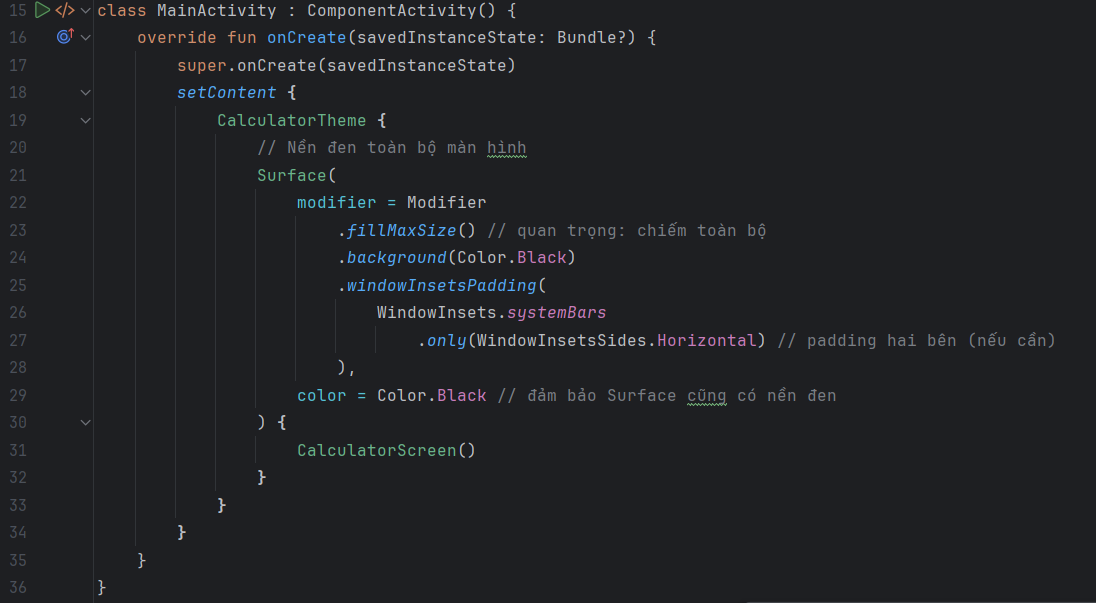
Ngoài ra, việc tổ chức logic lưu – đọc dữ liệu tách biệt rõ ràng trong DataStoreHelper giúp tăng tính tái sử dụng và dễ bảo trì cho mã nguồn, tuân theo nguyên tắc phân tách trách nhiệm

### 4.4.2 Sơ đò use case :



## Sơ đồ Use Case dưới đây mô tả các chức năng liên quan đến việc **quản lý lịch sử phép tính** trong ứng dụng. Khi người dùng thực hiện phép tính, hệ thống sẽ **tự động lưu** biểu thức và kết quả vào bộ nhớ cục bộ bằng cơ chế DataStore. Sau đó, khi người dùng muốn xem lại, hệ thống sẽ **đọc lại dữ liệu đã lưu**, chuyển đổi từ định dạng JSON thành danh sách các phép tính và **hiển thị lên giao diện** dưới dạng danh sách lịch sử. Toàn bộ quá trình diễn ra tự động, giúp người dùng dễ dàng tra cứu lại các phép tính trước đó. Ngoài ra, ứng dụng có thể cung cấp thêm chức năng ****x**óa toàn bộ lịch sử** nếu cần.

## 4.5 Hàm Main ( đây là phần khởi tạo ):



Tệp MainActivity.kt là điểm khởi đầu của ứng dụng máy tính, nơi khởi tạo giao diện người dùng bằng Jetpack Compose. Lớp MainActivity kế thừa từ ComponentActivity và ghi đè phương thức onCreate() – nơi định nghĩa giao diện chính được hiển thị khi người dùng mở ứng dụng.

Bên trong onCreate(), hàm setContent { ... } được sử dụng để hiển thị toàn bộ giao diện Compose. Toàn bộ giao diện được bọc trong một CalculatorTheme, đảm bảo ứng dụng sử dụng nhất quán các màu sắc, phông chữ, và phong cách chủ đề đã định nghĩa.

*Thiết lập nền đen toàn màn hình*

* Thành phần chính trong giao diện là một Surface – container đóng vai trò làm lớp nền chính cho toàn bộ màn hình. Surface này có các thuộc tính được thiết lập như sau:
* modifier = Modifier.fillMaxSize() – đảm bảo Surface chiếm toàn bộ kích thước màn hình.
* .background(Color.Black) – thiết lập nền đen cho toàn bộ ứng dụng.
* .windowInsetsPadding(...) – thêm phần đệm hai bên theo các thanh điều hướng hệ thống (status bar, navigation bar), để tránh bị che khuất nội dung trên thiết bị có notch hoặc mép cong.
* Màu nền (color) của Surface cũng được thiết lập là Color.Black để đồng bộ với modifier background, đảm bảo giao diện có màu tối đồng nhất.

*Giao diện chính: CalculatorScreen()*

* Bên trong Surface, hàm CalculatorScreen() được gọi để hiển thị toàn bộ giao diện chính của máy tính, bao gồm:
* Khu vực hiển thị biểu thức và kết quả.
* Bàn phím số và các nút chức năng như: cộng, trừ, căn bậc hai, phần trăm, xóa,...
* Các tính năng nâng cao như đổi dấu, lịch sử, bình phương,...

Việc tách CalculatorScreen() thành một composable riêng giúp mã nguồn dễ tổ chức, thuận tiện cho việc bảo trì và mở rộng về sau.

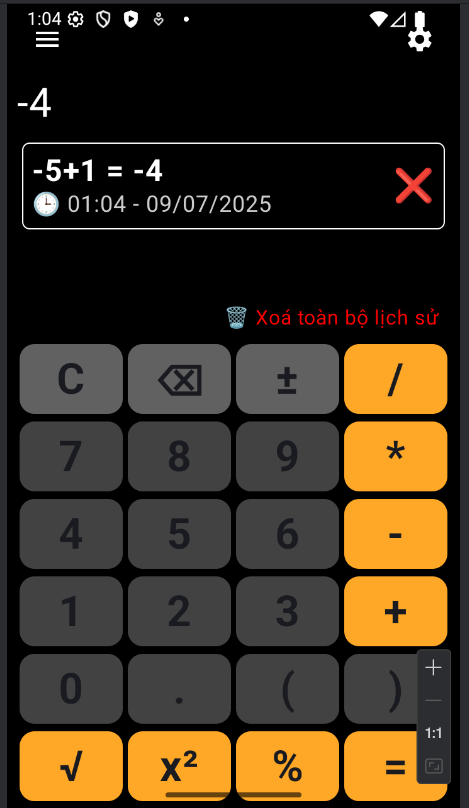
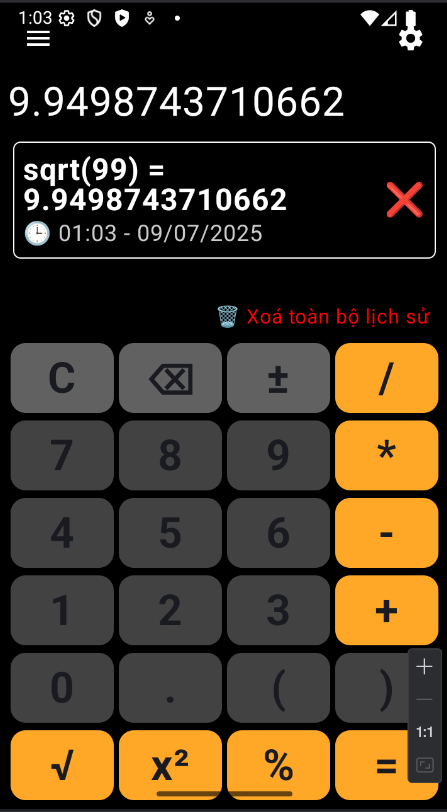
# V. KẾT QUẢ THỰC HIỆN VÀ KIỂM THỬ

## 5.1. Kiểm thử chức năng:

|  |  |  |
| --- | --- | --- |
| Nhập biểu thức | Kết quả | Nhận xét |
| 6+2 | 8 | Đúng |
| 4\*3+1 | 13 | Đúng |
| 3log | Lỗi cú pháp | Đúng |
| Sin(30) | -0.9880316241 | Đúng |
|  | 6.283185307 | Đúng |
| 3-6 | -3 | Đúng |
| Xem lại lịch sử | Lịch sử | Hiện bảng lịch sử tính toán |
| C ( nhập xóa ) | Xóa | Biểu thức xóa thành công |
| Xóa lịch sử | Xóa | Lịch sử tính toán được xóa thành công |

## 

## 5.2. Giao diện thử nghiệm



# 

# VI. KẾT LUẬN

## 6.1. Ưu điểm

1. Giao diện trực quan, dễ thao tác  
   Ứng dụng được thiết kế với bố cục đơn giản, các nút số và phép toán được sắp xếp khoa học, giúp người dùng dễ dàng thao tác, đặc biệt phù hợp với cả người mới sử dụng hoặc người lớn tuổi.
2. Mã nguồn tách biệt rõ giữa UI và logic  
   Cấu trúc mã nguồn được tổ chức hợp lý: phần giao diện (XML hoặc Jetpack Compose) tách biệt với phần xử lý logic (Kotlin), giúp dễ bảo trì, sửa lỗi và mở rộng tính năng trong tương lai.
3. Ứng dụng tương thích với nhiều thiết bị  
   Ứng dụng được xây dựng trên nền tảng Android Studio với cấu hình tương thích từ Android 5.0 (API 21) trở lên, nên có thể chạy tốt trên nhiều loại thiết bị như điện thoại, máy tính bảng với các kích thước màn hình khác nhau.
4. Dễ dàng mở rộng và nâng cấp  
   Với thiết kế linh hoạt, ứng dụng có thể dễ dàng nâng cấp các chức năng mới như: lưu lịch sử tính toán, hỗ trợ phép tính nâng cao (sin, cos, log…), thay đổi chủ đề giao diện (dark/light mode), v.v.

## 6.2. Hạn chế

Mặc dù ứng dụng máy tính nâng cao đã tích hợp nhiều chức năng mở rộng và hỗ trợ người dùng thực hiện các phép tính phức tạp, nhưng vẫn tồn tại một số hạn chế nhất định như sau:

1. Chưa hỗ trợ nhập biểu thức theo định dạng tự nhiên (infix đầy đủ)  
   Ứng dụng hiện vẫn yêu cầu người dùng nhập biểu thức theo thứ tự từ trái sang phải, chưa có tính năng phát hiện hoặc gợi ý sai cú pháp, thiếu dấu ngoặc, hoặc gợi ý kết thúc biểu thức chưa hợp lệ.
2. Chưa hỗ trợ biểu thức nhiều dòng hoặc đa phép toán trên từng dòng  
   Giao diện hiện tại chỉ cho phép người dùng nhập và xử lý một biểu thức duy nhất tại một thời điểm, chưa hỗ trợ tính năng nhập biểu thức nhiều dòng như trong các phần mềm toán học chuyên dụng.
3. Không có chức năng đồ họa hoặc biểu đồ  
   Máy tính nâng cao trong ứng dụng chưa có khả năng vẽ biểu đồ hàm số, đồ thị tọa độ, hoặc các biểu diễn hình học – những tính năng hữu ích trong các ứng dụng toán học hiện đại.
4. Không đồng bộ dữ liệu  
   Dữ liệu lịch sử được lưu cục bộ bằng DataStore, do đó người dùng sẽ bị mất toàn bộ dữ liệu nếu xóa ứng dụng hoặc đổi thiết bị. Hệ thống chưa hỗ trợ đăng nhập, sao lưu đám mây hoặc đồng bộ đa thiết bị.

# VII. TÀI LIỆU THAM KHẢO