BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC ĐẠI NAM

ĐỒ ÁN TỐT NGHIỆP

ĐỀ TÀI: SỬ DỤNG THƯ VIỆN MPANDROIDCHART XÂY DỰNG ỨNG DỤNG MOBILE HỖ TRỢ VẼ ĐỒ THỊ HÀM SỐ

SINH VIÊN THỰC HIỆN : NGUYỄN ĐỨC QUÂN
MÃ SINH VIÊN : 1451020186
KHOA : CÔNG NGHỆ THÔNG TIN

Hà Nội, năm 2024

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC ĐẠI NAM

ĐỒ ÁN TỐT NGHIỆP

ĐỀ TÀI: SỬ DỤNG THƯ VIỆN MPANDROIDCHART XÂY DỰNG ỨNG DỤNG MOBILE HỖ TRỢ VẼ ĐỒ THỊ HÀM SỐ

CHUYÊN NGÀNH : CÔNG NGHỆ THÔNG TIN

MÃ SỐ : 74.80.201

GIẢNG VIÊN HƯỚNG DẪN: TS. NGUYỄN NGỌC TÂN

Hà Nội, năm 2024

NHẬN XÉT

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

LỜI CAM ĐOAN

Em xin cam đoan rằng toàn bộ nội dung trong bài báo cáo này là công sức làm việc và nghiên cứu của em, không sao chép từ bất kỳ nguồn nào khác mà không được ghi rõ. Mọi thông tin, số liệu và ý kiến trong báo cáo đều được thể hiện một cách trung thực và minh bạch, dựa trên kiến thức và kinh nghiệm của em.

Em cam đoan rằng bài báo cáo này không vi phạm bất kỳ quy định hay nguyên tắc nào của trường, tổ chức hoặc cộng đồng nghiên cứu. Mọi thông tin được trích dẫn từ nguồn khác đều được ghi rõ nguồn gốc, và các ý kiến cá nhân của em được biểu đạt một cách chính xác và không gây hiểu nhầm.

Em sẵn sàng chịu trách nhiệm nếu có bất kỳ vấn đề nào phát sinh từ việc sử dụng thông tin trong báo cáo.

Họ và tên sinh viên

LỜI CẢM ƠN

Chúng em xin chân thành cảm ơn giảng viên Trường Đại Học Đại Nam, cũng như thầy Nguyễn Ngọc Tân – giảng viên hướng dẫn đã dành thời gian giảng dạy những kiến thức môn học này trong những buổi học vừa qua, giúp em nắm vững lý thuyết, tạo điều kiện cho chúng em thực hiện để tài này.

Các buổi học lý thuyết đan xen với thực hành là cơ sở để chúng em có những kiến thức cần thiết để hoàn thiện bài tập lớn này. Điều này là nhờ phương pháp giảng dạy tận tình của các thầy cô trong khoa.

Tuy vậy trong quá trình học tập cũng như làm bài tập lớn, chúng em cũng không thể tránh khỏi những thiếu sót, chúng em rất mong được những sự góp ý, đánh giá của tất cả các thầy cô để kết quả bài tập được tốt nhất.

Một lần nữa, em xin chân thành cảm ơn!

LỜI NÓI ĐẦU

Trong thời đại công nghệ phát triển mạnh mẽ, ngành lập trình Mobile được coi là một trong những nghề hot nhất hiện nay. Với sự phổ biến của các thiết bị di động và ứng dụng mobile, nhu cầu tìm kiếm các chuyên gia trong lĩnh vực này ngày càng cao.

Để có thể tìm hiểu về quá trình làm một ứng dụng Mobile, em đã chọn "sử dụng thư viện MPAndroidChard xây dựng ứng dụng mobile hỗ trợ vẽ đồ thị hàm số" làm đề tài của mình. Đầu tiên, ứng dụng này hữu ích trong thực tế, giúp người dùng dễ dàng học và áp dụng kiến thức toán học. Nó cũng thúc đẩy sáng tạo bằng cách cho phép người dùng tạo và tương tác với các đồ thị hàm số. Thứ hai, phát triển ứng dụng này là cơ hội để nâng cao kỹ năng lập trình mobile và tích hợp tính năng di động, làm cho nó trở thành một dự án thú vị và giúp người phát triển phát triển tốt hơn.

MỤC LỤC

MỤC LỤC HÌNH ẢNH

MỞ ĐẦU

1. Lý do chọn đề tài

Đồ thị hàm số là một phần quan trọng trong chương trình Toán học phổ thông. Việc hiểu và vẽ được đồ thị hàm số giúp học sinh, sinh viên có cái nhìn trực quan hơn về các khái niệm toán học phức tạp, từ đó nâng cao khả năng phân tích và giải quyết vấn đề. Một ứng dụng vẽ đồ thị hàm số hỗ trợ tốt cho quá trình học tập, giúp học sinh sinh viên tự học và làm bài tập một cách hiệu quả hơn.

Trong thời đại công nghệ số, việc sử dụng các ứng dụng hỗ trợ học tập ngày càng phổ biến. Tuy nhiên, hiện nay, các ứng dụng vẽ đồ thị hàm số đa phần còn hạn chế về tính năng và chưa đáp ứng được đầy đủ nhu cầu của người dùng. Một ứng dụng vẽ đồ thị hàm số với giao diện thân thiện, tính năng đa dạng và chính xác sẽ đáp ứng được nhu cầu này, đồng thời tạo ra giá trị thực tế cho người sử dụng.

Ngoài giáo dục, việc vẽ và phân tích đồ thị hàm số còn ứng dụng trong nhiều lĩnh vực khác như kinh tế, kỹ thuật, khoa học dữ liệu, và nhiều lĩnh vực nghiên cứu khác. Một ứng dụng có khả năng vẽ đồ thị chính xác và tính toán các thông số quan trọng sẽ là công cụ hữu ích cho các nhà nghiên cứu và chuyên gia trong các lĩnh vực này.

Đề tài vẽ đồ thị hàm số đòi hỏi sự kết hợp của nhiều kỹ thuật lập trình khác nhau như tính toán toán học, xử lý đồ họa, thiết kế giao diện người dùng (UI/UX) và tối ưu hóa hiệu suất. Việc thực hiện đề tài này mang lại cơ hội để học hỏi và rèn luyện các kỹ năng lập trình, giải quyết vấn đề, và quản lý dự án. Đồng thời, việc giải quyết những thách thức kỹ thuật sẽ giúp tích lũy kinh nghiệm quý báu cho các dự án trong tương lai.

2. Đối tượng và phạm vi nghiên cứu

- Thư viện MPAndroidChart

Thư viện MPAndroidChart là một công cụ mạnh mẽ trên nền tảng Android, cung cấp các chức năng vẽ đồ thị đa dạng và hiệu quả. Thư viện này hỗ trợ nhiều loại biểu đồ như biểu đồ đường (LineChart), biểu đồ thanh (BarChart), biểu đồ tròn (PieChart), biểu đồ nến (CandleStickChart), và nhiều loại biểu đồ khác. MPAndroidChart cho phép tùy chỉnh cao và có hiệu suất tốt, là lựa chọn lý tưởng cho việc hiển thị dữ liệu một cách trực quan và dễ hiểu trong ứng dụng vẽ đồ thị hàm số.

- Giao diện người dùng (UI)

Thiết kế giao diện người dùng là một phần quan trọng của ứng dụng. Sử dụng các Fragment để chia ứng dụng thành các trang khác nhau giúp quản lý và hiển thị thông tin một cách rõ ràng và dễ dàng. Các thành phần giao diện như nút bấm, ô nhập liệu và biểu đồ được thiết kế để tối ưu hóa trải nghiệm người dùng, đảm bảo tính thân thiện và dễ sử dụng. Giao diện đẹp mắt và tương tác tốt là yếu tố then chốt để ứng dụng trở nên hấp dẫn và hiệu quả.

- Công cụ và Môi trường phát triển

Android Studio là môi trường phát triển tích hợp (IDE) chính được sử dụng trong quá trình phát triển ứng dụng. Đây là công cụ mạnh mẽ hỗ trợ việc viết mã, gỡ lỗi và kiểm thử, giúp quá trình phát triển trở nên hiệu quả và thuận lợi hơn. Ngôn ngữ lập trình Java/Kotlin được sử dụng kết hợp với thư viện MPAndroidChart để tạo ra các chức năng vẽ đồ thị, mang lại tính linh hoạt và hiệu quả cao trong quá trình phát triển ứng dụng.

3. Mục tiêu và nhiệm vụ nghiên cứu

Mục đích chính của luận án "Sử dụng MPAndroidChart xây dựng ứng dụng vẽ đồ thị hàm số" là phát triển một ứng dụng di động hỗ trợ học tập và giảng dạy Toán học, tập trung vào việc vẽ đồ thị hàm số. Cụ thể, mục đích của nghiên cứu bao gồm:

- Cung cấp **công cụ hỗ trợ học tập hiệu quả**: Tạo ra một ứng dụng giúp học sinh và sinh viên hiểu và vẽ đồ thị hàm số một cách trực quan và chính xác. Ứng dụng này sẽ hỗ trợ học tập, làm bài tập và ôn luyện các kiến thức Toán học.
- Nâng **cao chất lượng giảng dạy**: Hỗ trợ giáo viên và giảng viên trong việc minh họa các khái niệm toán học, giúp giảng dạy hiệu quả hơn thông qua các công cụ trực quan và tương tác.
- Ứng dụng **công nghệ hiện đại trong giáo dục**: Khai thác và ứng dụng các công nghệ mới, đặc biệt là thư viện MPAndroidChart, để phát triển các công cụ giáo dục tiên tiến, góp phần nâng cao chất lượng giáo dục.

Để đạt được các mục đích nêu trên, nghiên cứu cần thực hiện các nhiệm vụ sau:

- Nghiên cứu và Phân tích Thư viện MPAndroidChart:
    - Tìm hiểu các tính năng và khả năng của MPAndroidChart.
    - Đánh giá khả năng áp dụng thư viện này trong việc vẽ đồ thị hàm số.
- Thiết Kế Giao Diện Người Dùng:
    - Sử dụng các Fragment để chia ứng dụng thành các trang khác nhau, giúp quản lý và hiển thị thông tin một cách rõ ràng và dễ dàng.
    - Thiết kế các thành phần giao diện như nút bấm, ô nhập liệu và biểu đồ để tối ưu hóa trải nghiệm người dùng.

Mục đích và nhiệm vụ nghiên cứu trong luận án "Sử dụng MPAndroidChart xây dựng ứng dụng vẽ đồ thị hàm số" tập trung vào việc phát triển một ứng dụng hỗ trợ học tập và giảng dạy Toán học hiệu quả, sử dụng công nghệ hiện đại để cung cấp các công cụ trực quan và mạnh mẽ. Việc nghiên cứu thư viện MPAndroidChart, thiết kế giao diện người dùng và tối ưu hóa hiệu suất là những nhiệm vụ then chốt để đạt được mục đích này. Thông qua quá trình nghiên cứu và phát triển, ứng dụng sẽ mang lại giá trị thực tiễn cho giáo dục và nâng cao kỹ năng chuyên môn của người nghiên cứu.

4. Phương pháp nghiên cứu

- Phương pháp nghiên cứu tài liệu

Mục tiêu của phương pháp này là tìm hiểu và thu thập kiến thức lý thuyết liên quan đến các khái niệm về đồ thị hàm số, các phép toán toán học, cũng như các công nghệ và thư viện hỗ trợ lập trình Android. Cụ thể, việc thực hiện bao gồm:

  - Đọc các sách giáo khoa, tài liệu chuyên ngành về Toán học, đặc biệt là về đồ thị hàm số.
  - Tham khảo các tài liệu hướng dẫn sử dụng thư viện MPAndroidChart và các tài liệu lập trình Android.
  - Nghiên cứu các bài báo khoa học, luận án và các công trình nghiên cứu liên quan.

- Phương pháp lập trình và phát triển
  - Mục tiêu của phương pháp này là triển khai các chức năng của ứng dụng dựa trên thiết kế đã đề ra, sử dụng ngôn ngữ lập trình và công cụ phù hợp. Cụ thể, việc thực hiện bao gồm:
  - Sử dụng Android Studio làm môi trường phát triển chính.
  - Lập trình các thành phần giao diện người dùng bằng XML và Java/Kotlin.
  - Sử dụng thư viện MPAndroidChart để vẽ đồ thị hàm số.

CHƯƠNG 1. CƠ SỞ LÝ THUYẾT

1.1. Khái niệm hệ điều hành Android:

Android là một hệ điều hành di động phổ biến được phát triển bởi Google. Nó được thiết kế để hoạt động trên các thiết bị di động như điện thoại thông minh, máy tính bảng, và các thiết bị khác có màn hình cảm ứng. Android được xây dựng trên nền tảng mã nguồn mở, điều này có nghĩa là mã nguồn của nó là công khai và có thể được sửa đổi và phát triển bởi cộng đồng lập trình viên.

Hình 1. Hệ điều hành Android

Android cung cấp một môi trường phát triển mạnh mẽ cho việc xây dựng ứng dụng di động sử dụng ngôn ngữ lập trình Java hoặc Kotlin. Hệ thống ứng dụng Android được viết dựa trên các thành phần cơ bản như Activity (màn hình), Service (dịch vụ), Broadcast Receiver (nhận tín hiệu phát sóng), và Content Provider (cung cấp dữ liệu). Nó cũng hỗ trợ nhiều tính năng quan trọng bao gồm cơ sở dữ liệu SQLite, tương tác với cảm biến, quản lý dữ liệu, và tích hợp với các tính năng di động như máy ảnh, GPS và nhiều hơn nữa.

Android là nền tảng phát triển ứng dụng di động phổ biến, với một cộng đồng lập trình viên lớn và nhiều ứng dụng có sẵn trên Google Play Store. Điều này làm cho Android trở thành một môi trường quan trọng cho phát triển ứng dụng di động trên nhiều loại thiết bị và kích thước màn hình.

1.2. Ngôn ngữ lập trình Android:

1.2.1. Ngôn ngữ Java

1.2.1.1. Khái niệm

[1]Java là một trong những ngôn ngữ lập trình hướng đối tượng. Nó được sử dụng trong phát triển phần mềm, trang web, game hay ứng dụng trên các thiết bị di động.

Hình 2. Ngôn ngữ lập trình Java

Java được khởi đầu bởi James Gosling và bạn đồng nghiệp ở Sun MicroSystem năm 1991. Ban đầu Java được tạo ra nhằm mục đích viết phần mềm cho các sản phẩm gia dụng, và có tên là Oak.

Java được phát hành năm 1994, đến năm 2010 được Oracle mua lại từ Sun MicroSystem.

Java được tạo ra với tiêu chí "*Viết (code) một lần, thực thi khắp nơi*" (Write Once, Run Anywhere – WORA). Chương trình phần mềm viết bằng Java có thể chạy trên mọi nền tảng (platform) khác nhau thông qua một môi trường thực thi với điều kiện có môi trường thực thi thích hợp hỗ trợ nền tảng đó.

1.2.1.2. Đặc điểm

a. Tương tự C++, hướng đối tượng hoàn toàn

Trong quá trình tạo ra một ngôn ngữ mới phục vụ cho mục đích chạy được trên nhiều nền tảng, các kỹ sư của Sun MicroSystem muốn tạo ra một ngôn ngữ dễ học và

quen thuộc với đa số người lập trình. Vì vậy họ đã sử dụng lại các cú pháp của C và C++.

Tuy nhiên, trong Java thao tác với con trỏ bị lược bỏ nhằm đảo bảo tính an toàn và dễ sử dụng hơn. Các thao tác overload, goto hay các cấu trúc như struct và union cũng được loại bỏ khỏi Java.

b. Độc lập phần cứng và hệ điều hành

Một chương trình viết bằng ngôn ngữ Java có thể chạy tốt ở nhiều môi trường khác nhau. Gọi là khả năng "cross-platform". Khả năng độc lập phần cứng và hệ điều hành được thể hiện ở 2 cấp độ là cấp độ mã nguồn và cấp độ nhị phân.

Ở cấp độ mã nguồn: Kiểu dữ liệu trong Java nhất quán cho tất cả các hệ điều hành và phần cứng khác nhau. Java có riêng một bộ thư viện để hỗ trợ vấn đề này. Chương trình viết bằng ngôn ngữ Java có thể biên dịch trên nhiều loại máy khác nhau mà không gặp lỗi.

Ở cấp độ nhị phân: Một mã biên dịch có thể chạy trên nhiều nền tảng khác nhau mà không cần dịch lại mã nguồn. Tuy nhiên cần có Java Virtual Machine để thông dịch đoạn mã này.

c. Ngôn ngữ thông dịch

Ngôn ngữ lập trình thường được chia ra làm 2 loại (tùy theo các hiện thực hóa ngôn ngữ đó) là ngôn ngữ thông dịch và ngôn ngữ biên dịch.

- Thông dịch (Interpreter) : Nó dịch từng lệnh rồi chạy từng lệnh, lần sau muốn chạy lại thì phải dịch lại.
- Biên dịch (Compiler): Code sau khi được biên dịch sẽ tạo ra 1 file thường là .exe, và file .exe này có thể đem sử dụng lại không cần biên dịch nữa.

Ngôn ngữ lập trình Java thuộc loại ngôn ngữ thông dịch. Chính xác hơn, Java là loại ngôn ngữ vừa biên dịch vừa thông dịch. Cụ thể như sau

Khi viết mã, hệ thống tạo ra một tệp .java. Khi biên dịch mã nguồn của chương trình sẽ được biên dịch ra mã byte code. Máy ảo Java (Java Virtual Machine) sẽ thông dịch mã byte code này thành machine code (hay native code) khi nhận được yêu cầu chạy chương trình.

- Ưu điểm : Phương pháp này giúp các đoạn mã viết bằng Java có thể chạy được trên nhiều nền tảng khác nhau. Với điều kiện là JVM có hỗ trợ chạy trên nền tảng này.
- Nhược điểm : Cũng như các ngôn ngữ thông dịch khác, quá trình chạy các đoạn mã Java là chậm hơn các ngôn ngữ biên dịch khác (tuy nhiên vẫn ở trong một mức chấp nhận được).

d. Ngôn ngữ thông dịch

Khi tạo ra các đối tượng trong Java, JRE sẽ tự động cấp phát không gian bộ nhớ cho các đối tượng ở trên heap.

Với ngôn ngữ như C C++, bạn sẽ phải yêu cầu hủy vùng nhớ mà bạn đã cấp phát, để tránh việc thất thoát vùng nhớ. Tuy nhiên vì một lý do nào đó, bạn không hủy một vài vùng nhớ, dẫn đến việc thất thoát và làm giảm hiệu năng chương trình.

Ngôn ngữ lập trình Java hỗ trợ cho bạn điều đó, nghĩa là bạn không phải tự gọi hủy các vùng nhớ. Bộ thu dọn rác của Java sẽ theo vết các tài nguyên đã được cấp. Khi không có tham chiếu nào đến vùng nhớ, bộ thu dọn rác sẽ tiến hành thu hồi vùng nhớ đã được cấp phát.

e. Đa luồng

Java hỗ trợ lập trình đa tiến trình (multithread) để thực thi các công việc đồng thời. Đồng thời cũng cung cấp giải pháp đồng bộ giữa các tiến trình (giải pháp sử dụng priority…).

f. Tính an toàn và bảo mật

- Tính an toàn

Ngôn ngữ lập trình Java yêu cầu chặt chẽ về kiểu dữ liệu.

Dữ liệu phải được khai báo tường minh.

Không sử dụng con trỏ và các phép toán với con trỏ.

Java kiểm soát chặt chẽ việc truy nhập đến mảng, chuỗi. Không cho phép sử dụng các kỹ thuật tràn. Do đó các truy nhập sẽ không vượt quá kích thước của mảng hoặc chuỗi.

Quá trình cấp phát và giải phóng bộ nhớ được thực hiện tự động.

Cơ chế xử lý lỗi giúp việc xử lý và phục hồi lỗi dễ dàng hơn.

- Tính bảo mật

Java cung cấp một môi trường quản lý chương trình với nhiều mức khác nhau.

Mức 1 : Chỉ có thể truy xuất dữ liệu cũng như phương phức thông qua giao diện mà lớp cung cấp.

Mức 2 : Trình biên dịch kiểm soát các đoạn mã sao cho tuân thủ các quy tắc của ngôn ngữ lập trình Java trước khi thông dịch.

Mức 3 : Trình thông dịch sẽ kiểm tra mã byte code xem các đoạn mã này có đảm bảo được các quy định, quy tắc trước khi thực thi.

Mức 4: Java kiểm soát việc nạp các lớp vào bộ nhớ để giám sát việc vi phạm giới hạn truy xuất trước khi nạp vào hệ thống.

1.2.2. Ngôn ngữ Kotlin

1.2.2.1. Khái niệm

[2]Kotlin là một ngôn ngữ lập trình kiểu tĩnh chạy trên máy ảo Java (JVM) và có thể được biên dịch sang mã nguồn Java hay sử dụng cơ sở hạ tầng trình biên dịch LLVM. Nó được tài trợ và phát triển bởi bởi JetBrains. Mặc dù cú pháp không tương thích với Java, nhưng bản thực hiện JVM của thư viện chuẩn Kotlin được thiết kế để tương tác với mã Java và dựa vào mã Java từ Java Class Library có sẵn, ví dụ như collections framework. Kotlin sử dụng suy luận kiểu một cách tích cực để xác định kiểu của giá trị và biểu thức vốn không được nêu rõ. Điều này giúp giảm tính dài dòng của ngôn ngữ so với Java, vốn thường đòi hỏi toàn bộ đặc kiểu một cách dư thừa mãi đến phiên bản 10. Mã Kotlin có thể chạy trên JVM đến phiên bản Java 11 mới nhất.

Hình 3. Ngôn ngữ lập trình Kotlin

1.2.2.2. Ưu điểm nổi bật của ngôn ngữ lập trình Kotlin[3]

a. Khắc phục nhược điểm của Java:

Không thể phủ nhận vai trò của Java đối với Android nói riêng và ngành công nghệ thông tin nói chung. Hiện nay, Java là một trong những ngôn ngữ lập trình được ưu tiên sử dụng khi các lập trình viên thực hiện các dự án Android nhưng nó vẫn tồn tại những hạn chế nhất định. Một trong những vấn đề lớn nhất còn tồn đọng ở Java là thiếu khả năng mở rộng cũng như không thể hỗ trợ tính năng cho các lập trình hàm. Mặc dù Java giới thiệu rằng sẽ mang lại những tính năng cho nhà phát triển như: biểu thức lambda, interface methods và những yếu tố lập trình hàm. Tuy nhiên, tại Android thì Java mới chỉ có thể hỗ trợ một phần của các tính năng mà Java 8 cung cấp. Chính vì vậy, sự ra đời của Kotlin được các chuyên gia đánh giá là có thể khắc phục hoàn toàn mọi hạn chế mà Java không thực hiện được. Với các đặc tính đều được thừa hưởng từ Java nên bạn có thể sử dụng Kotlin cũng như khai thác được mọi nền tảng từ Java class Library hiện đang có.

b. Code ngắn gọn dễ hiểu

Ngôn ngữ lập trình Kotlin được xây dựng bằng hệ thống code ít giúp lập trình viên dễ đọc, dễ viết và dễ làm việc cùng. Những người mới bắt đầu đều có thể tiếp thu dễ dàng các kiến thức đặc thù của loại ngôn ngữ này. Việc tối giản được số lượng code

đã giúp cho Kotlin mang lại những trải nghiệm thú vị hơn cho người dùng so với các loại ngôn ngữ khác như Java.

c. Kotlin không bị lỗi NullPointerException

Với những lập trình viên thì bạn có thể thấy rõ NullPointerException là 1 trong những lỗi thường xuyên xuất hiện trong các dự án được viết bằng ngôn ngữ lập trình Java. Lỗi NullPointerException sẽ xuất hiện ngay khi bạn gán giá trị null đến với một đối tượng nào đó, tuy nhiên khi truy xuất thì đối tượng này lại bị xuất hiện lỗi. Khi chạy trên Android, nếu như bạn quên cập nhật đối tượng cho Java thì bạn sẽ nhận được log crash (dừng đột ngột) đã được báo cáo về hệ thống. Những lỗi NullPointerException.Kotlin đều được thiết kế để có thể giảm thiểu cũng như loại bỏ được hầu hết các nguồn tham chiếu Null dựa vào cơ chế null-safety. Chính vì vậy, theo các chuyên gia thì việc sử dụng ngôn ngữ lập trình Kotlin sẽ trở nên an toàn hơn Java rất nhiều.

Hầu hết các lập trình viên đều sẽ rất chú trọng đến những lỗi xảy ra ở code, vậy nên những code các ngắn thì càng ít lỗi hơn. Chính vì thế, nên code trong Kotlin đều được thiết kế ngắn gọn hơn so với code được viết bằng Java mà kết quả kiểm tra đều cho ra giống nhau. Để dễ hiểu hơn thì bạn có thể hình dung như sau: Nếu bạn định nghĩa một class trong Java cần phải sử dụng 7 đến 8 dòng thì khi dùng Kotlin dòng code đó sẽ được giảm xuống còn 2 đến 3 dòng hoặc thậm chí là 1 dòng mà kết quả sau cùng cho ra vẫn tương tự nhau.

d. Kotlin có khả năng tương tác cao

Kotlin được xem là một trong những ngôn ngữ lập trình có thể chạy trên máy ảo tương tự như Java. Tuy nhiên, khả năng tương tác cao giúp cho Kotlin có thể tương thích 100% với Java nên 1 dự án có thể sử dụng cả Java và Kotlin. Hiện nay, các developer Android đều có khả năng sử dụng Java class library ngay khi dùng Kotlin để có thể thực hiện viết code và ngược lại. Nhờ vậy, bạn có thể sử dụng ngôn ngữ này để có thể phát triển cũng như mở rộng cho việc phát triển các dự án Java bằng cũ mà không cần phải bắt lại. có thể sử dụng Java class library khi dùng Kotlin để viết code và ngược lại. Điều này có nghĩa , bạn có thể sử dụng Kotlin để bạn mở rộng và phát triển các Với những lập trình viên đã quen làm việc với Java khi chuyển sang hợp tác cùng Kotlin thì sẽ không còn cảm thấy bỡ ngỡ hay xa lạ bởi cú pháp của nó bây giờ đều rất quen thuộc.

Nhờ vậy,k các kỹ năng bạn đã có trong việc code Java đều có thể áp dụng được với Kotlin.

e. Kotlin được ưu tiên hỗ trợ trong Android Studio và IDE

Hiện nay, các developer Android đều có thể dễ dàng tận dụng các IDE được tích hợp từ Android Studio 3.0. Những phiên bản Android Studio thấp hơn cần phải thực hiện cài thêm plugin và nó khiến cho cấu hình Kotlin trong dự án trở nên đơn giản hơn. Hiện tại, các IDE hỗ trợ cho Java đều sẽ có thể hỗ trợ cho cả Kotlin. Chính vì thế, hầu hết các Developer đều có thể tận dụng IDE trong số đó cả cả Android Studio. Kotlin cùng với tool làm việc thân thiện sẽ hỗ trợ cho bạn lựa chọn Java IDE làm việc hoặc thực hiện làm việc cùng với command line.

1.2.2.3. Hạn chế của ngôn ngữ lập trình Kotlin

Song song với các ưu điểm trên thì ngôn ngữ lập trình Kotlin còn tồn tại những mặt hạn chế như sau: Chưa có kiểu Aliases. Vì chưa có kiểu Aliases nên kiểu hàm vẫn còn phải viết thủ công nên phần mã nguồn sẽ bị thừa thãi. Các mặc định class trong Kotlin là final. Bạn cần phải thêm từ khóa Open nếu như muốn class final trở thành class thông thường như trong Java. Đây chính là hạn chế có thể khiến cho các dự án có mã nguồn kết hợp giữa Kotlin và Java. Bởi vì một số Java Framework thường tự động bỏ qua từ khóa Final trong mã của Kotlin. Điều này sẽ khiến cho Kotlin có thể chạy không đúng với ý đồ của lập trình viên. Cộng đồng hỗ trợ hạn chế Mặc dù là ngôn ngữ có thể sử dụng được toàn bộ cũng như thư viện của Java nhưng theo nhiều lập trình viên thì phiên bản chính chủ vẫn tốt hơn. Kotlin không tự ép kiểu dữ liệu Kotlin sẽ không thực hiện tự động ép kiểu với những dữ liệu thuộc dạng nguyên thủy.

1.3. Công cụ phát triển – Android Studio

Android Studio là IDE chính thức được sử dụng trong phát triển ứng dụng Android dựa trên IntelliJ IDEA. Chức năng chính của Android Studio là cung cấp các giao diện giúp người dùng có thể tạo các ứng dụng và xử lý các công cụ file phức tạp sau hậu trường. Ngôn ngữ lập trình được sử dụng trong Android Studio là Java và nó sẽ được cài đặt sẵn trên thiết bị của bạn. Khi sử dụng Android Studio thì bạn chỉ cần viết, chỉnh sửa và lưu trữ chúng trên các dự án của mình và các file nằm trong dự án đó. Đồng thời, Android Studio còn cung cấp quyền truy cập vào Android SDK.

Hình 4. Android Studio

1.4. Thư viện MPAndroidChart

MPAndroidChart là thư viện mạnh mẽ và dễ sử dụng để phát triển cho hệ điều hành Android, chạy trên API cấp 8 và các phiên bản mới hơn. Được phát triển bởi Philipp Jahoda, MPAndroidChart cung cấp một loạt các loại biểu đồ như biểu đồ đường (Line Chart), biểu đồ thanh (Bar Chart), biểu đồ nến (Candlestick Chart), biểu đồ bong bóng (Bubble Chart), và nhiều loại khác.

Một trong những điểm mạnh của MPAndroidChart là tính dễ sử dụng và tính linh hoạt cao. Người dùng có thể dễ dàng tích hợp thư viện này vào dự án Android của mình thông qua việc thêm một dòng vào file build.gradle. Sau khi tích hợp, việc tạo biểu đồ trở nên rất trực quan, chỉ cần chuẩn bị dữ liệu và áp dụng các thiết lập cần thiết.

Thư viện này cũng hỗ trợ nhiều tính năng tùy chỉnh như thay đổi màu sắc, kiểu dáng, thêm các chú thích, đường chỉ báo, và khả năng tương tác với biểu đồ. Người dùng có thể phóng to, thu nhỏ, kéo thả, và chọn các điểm dữ liệu trên biểu đồ một cách dễ dàng. MPAndroidChart cũng cung cấp các công cụ để xử lý và hiển thị dữ liệu lớn mà không ảnh hưởng đến hiệu suất của ứng dụng.

Bên cạnh đó, thư viện này cũng được hỗ trợ tốt và liên tục cập nhật với các phiên bản mới, giúp đảm bảo tính tương thích và hiệu năng tốt nhất trên các thiết bị Android hiện đại. Cộng đồng người dùng MPAndroidChart rất đông đảo, cung cấp nhiều tài liệu, ví dụ, và hỗ trợ thông qua các diễn đàn, giúp người mới bắt đầu có thể nhanh chóng nắm bắt và sử dụng hiệu quả.

MPAndroidChart là một công cụ không thể thiếu cho các nhà phát triển Android muốn tích hợp các biểu đồ phức tạp và trực quan vào ứng dụng của mình. Với tính linh hoạt, dễ sử dụng và hiệu suất cao, MPAndroidChart mang lại tiềm năng tạo ra những ứng dụng chuyên nghiệp và hấp dẫn.

Để cài đặt sử dụng thư viện này, ta tiến hành add dependency vào file gradle bằng phiên bản MPAndroid Chart thích hợp:[4]

```
repositories {
    maven { url 'https://jitpack.io' }
}
dependencies {
    implementation 'com.github.PhilJay:MPAndroidChart:v3.1.0'
}
```

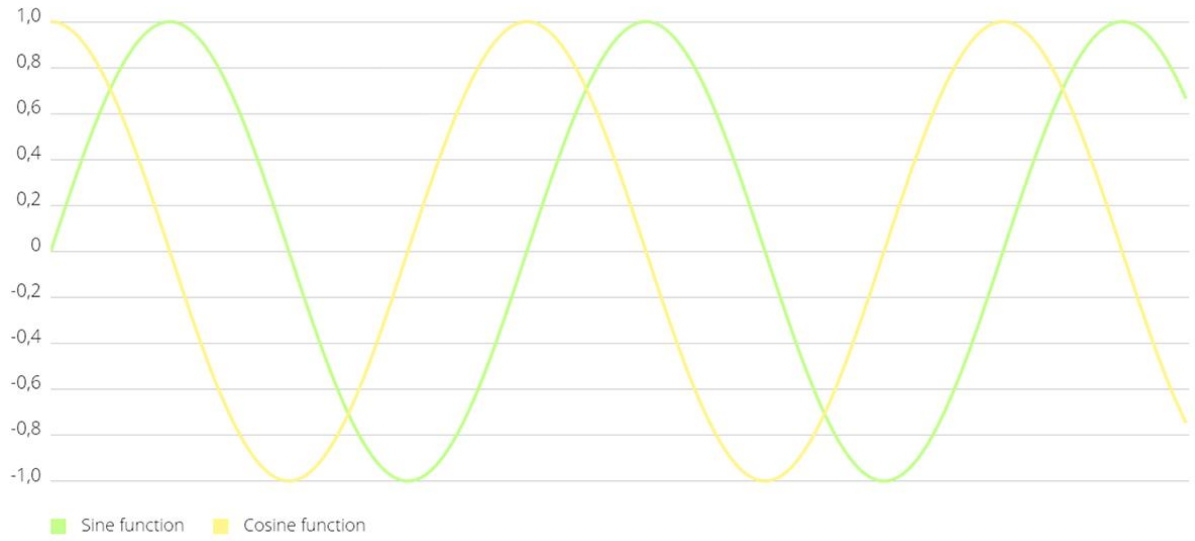
MPAndroidChart có nhiều biểu đồ khác nhau để thể hiện dữ liệu trên di động như sau:

1.4.1. Biểu đồ dạng đường

Code để tạo View:

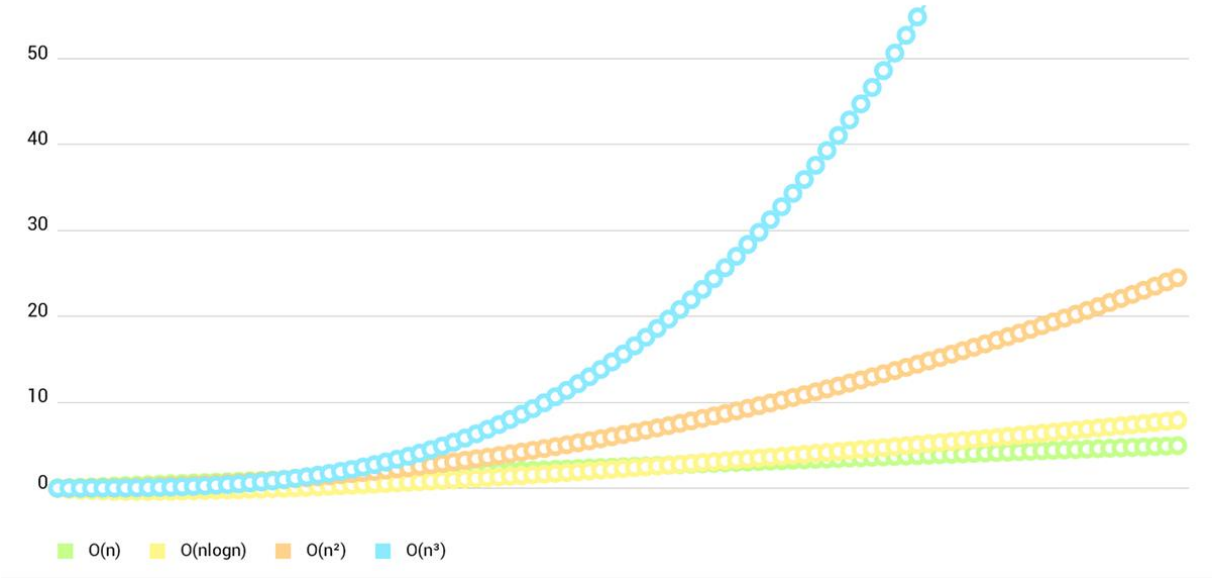
```
<com.github.mikephil.charting.charts.LineChart
    android:id="@+id/chart"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

- Biểu đồ dạng đường có thiết kế đơn giản với chú thích:



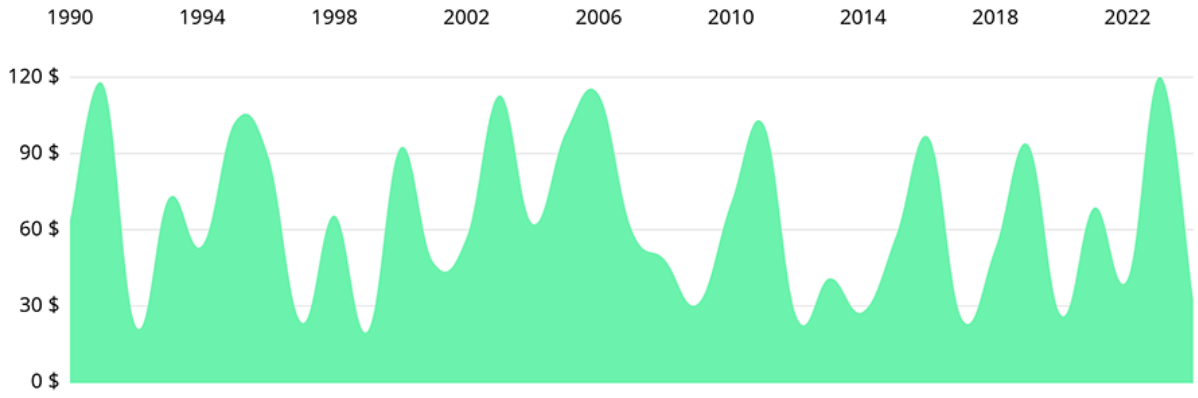
Hình 5. Ví dụ về biểu đồ dạng đường

- Biểu đồ dạng đường, với các điểm giá trị được làm nổi bật, có chú thích:



Hình 6. Ví dụ về biểu đồ dạng đường (2)

- Đồ thị dạng đường với khối được tô màu:



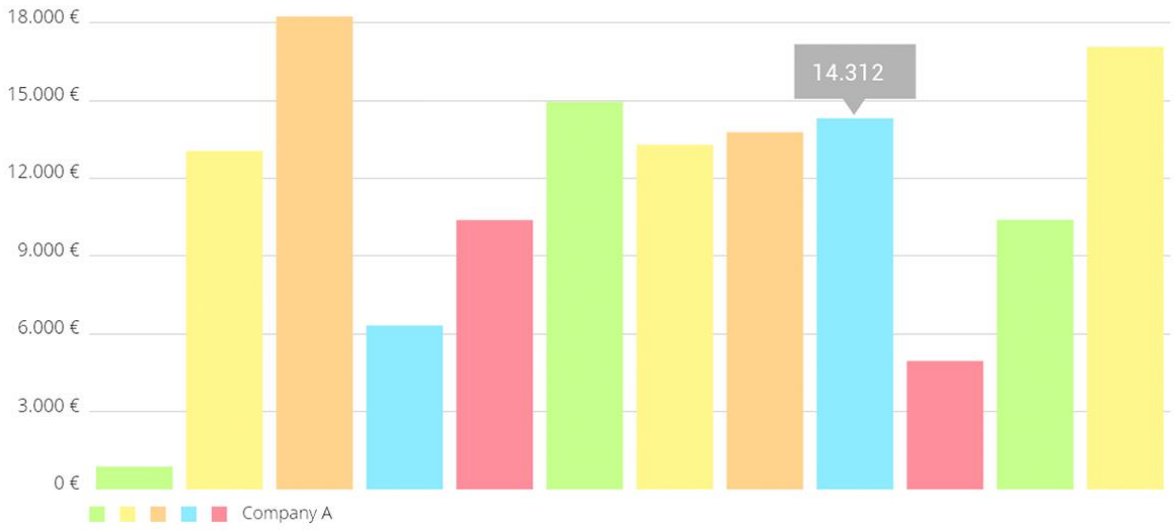
Hình 7. Ví dụ về các biểu đồ dạng đường (3)

1.4.2. Biểu đồ dạng cột

Code để tạo View:

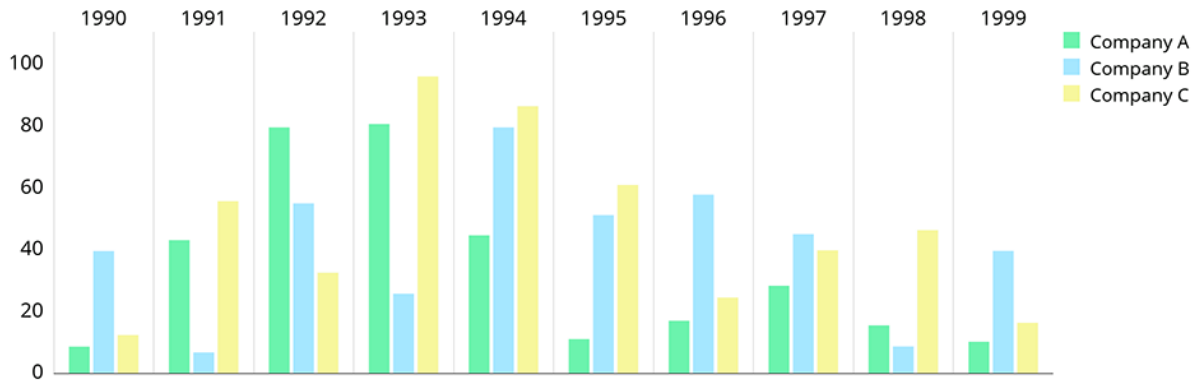
```
<com.github.mikephil.charting.charts.BarChart
    android:id="@+id/chart"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

- Biểu đồ dạng cột đơn giản, có chú thích:



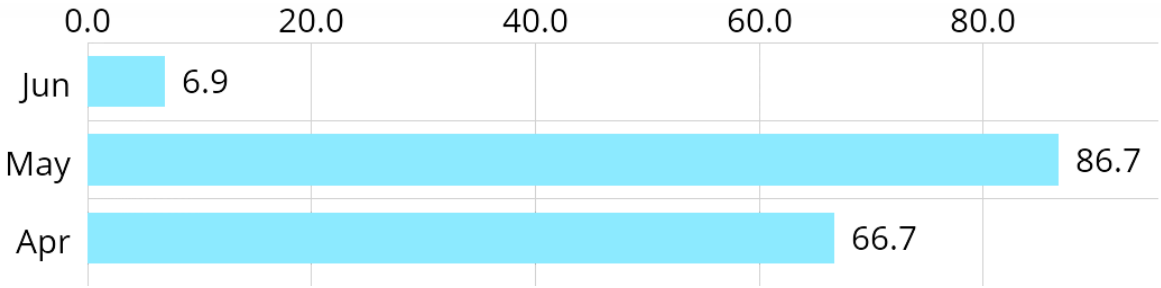
Hình 8. Ví dụ về Biểu đồ dạng cột

- Biểu đồ dạng cột với các nhóm dữ liệu:



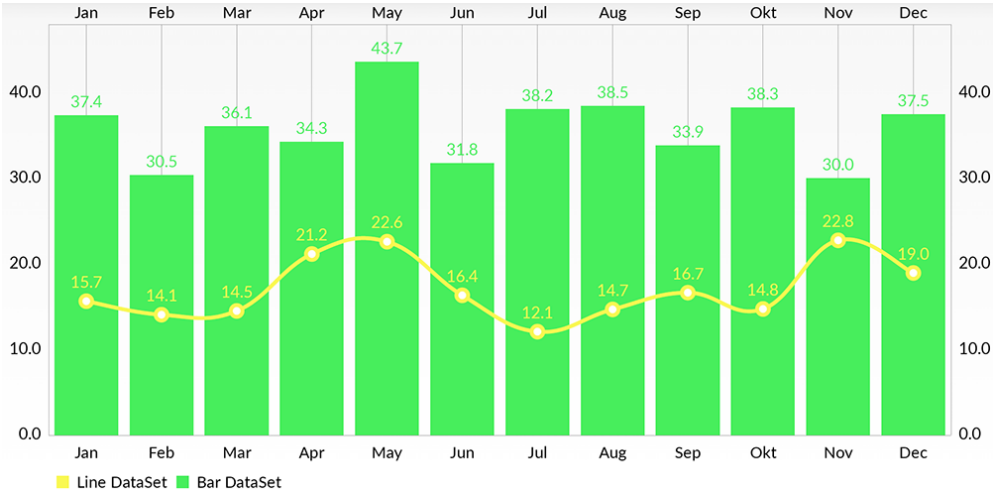
Hình 9. Ví dụ về Biểu đồ dạng cột (2)

- Biểu đồ dạng cột nằm ngang:



Hình 10. Ví dụ về biểu đồ dạng cột (3)

- Biểu đồ dạng cột kết hợp với đường:



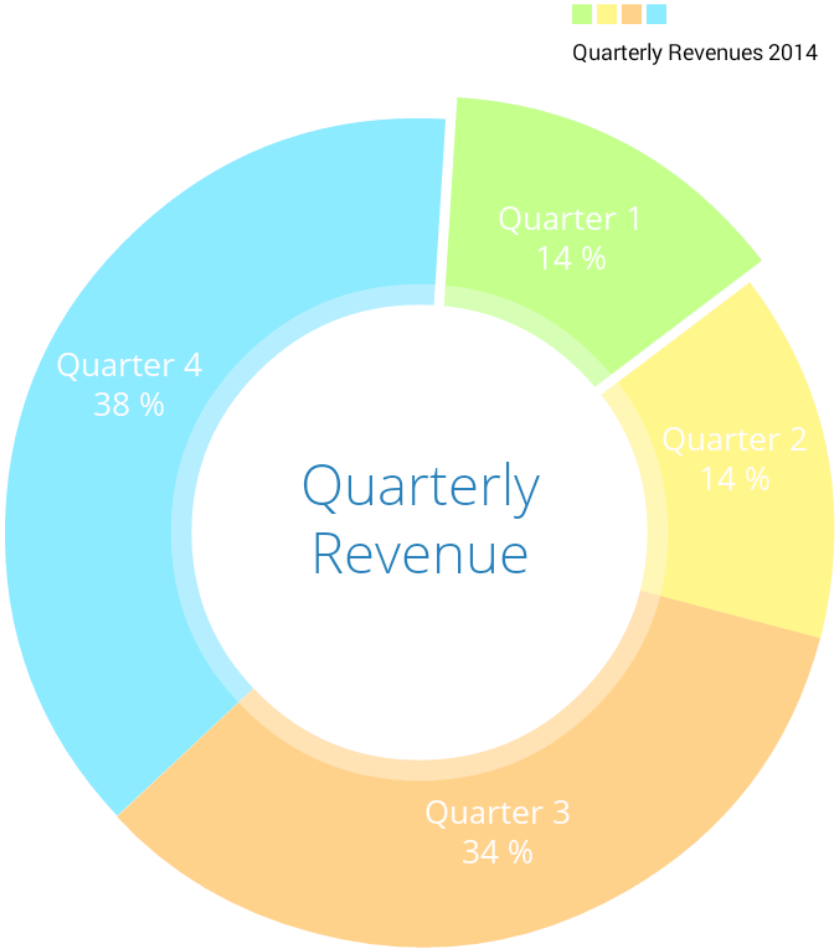
Hình 11. Biểu đồ đường kết hợp với cột

1.4.3. Biểu đồ hình tròn

Code để tạo view:

```
<com.github.mikephil.charting.charts.PieChart
    android:id="@+id/chart"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

Ví dụ về một biểu đồ tròn:



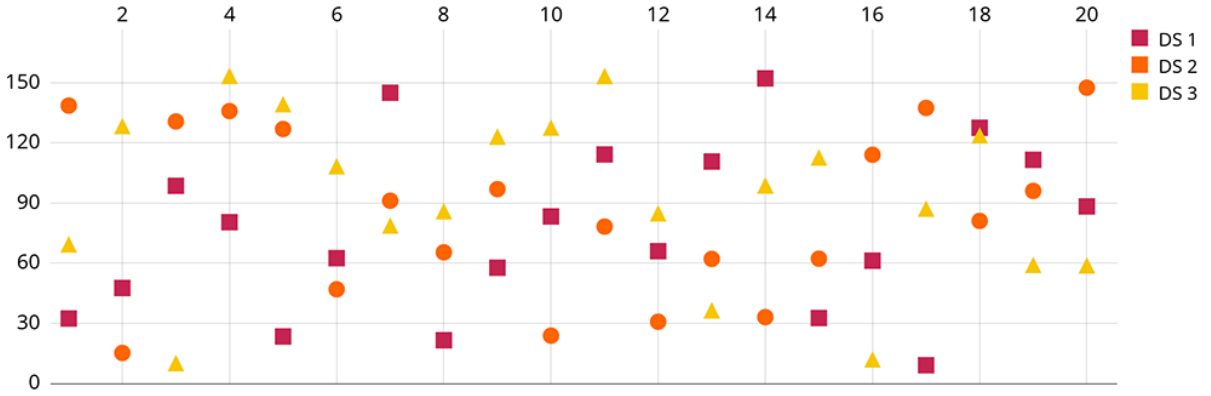
Hình 13. Ví dụ về biểu đồ tròn

1.4.4. Biểu đồ phân tán:

Code tạo View:

```
<com.github.mikephil.charting.charts.ScatterChart
    android:id="@+id/chart"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>
```

Ví dụ:



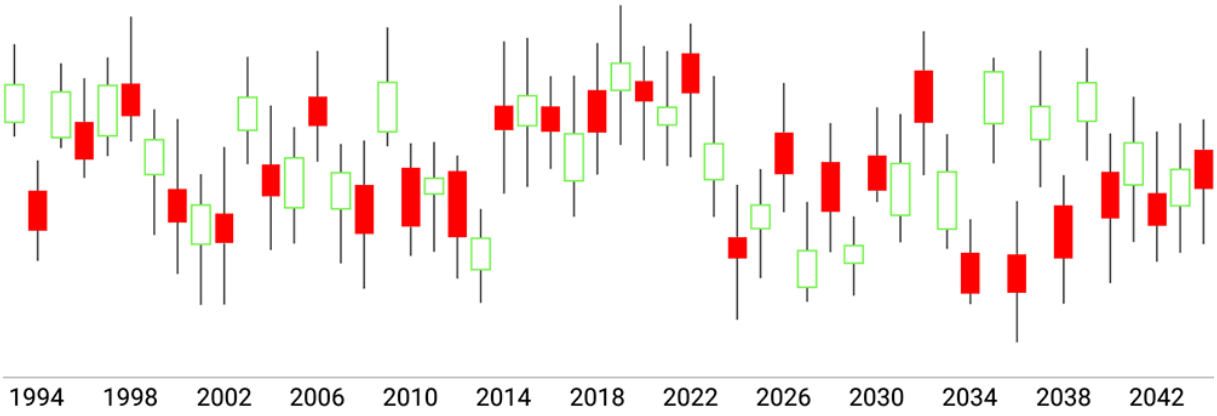
Hình 14. Ví dụ về biểu đồ phân tán

1.4.5. Biểu đồ hình nến

Code tạo View:

```
<com.github.mikephil.charting.charts.CandleStickChart
    android:id="@+id/chart"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>
```

Ví dụ:



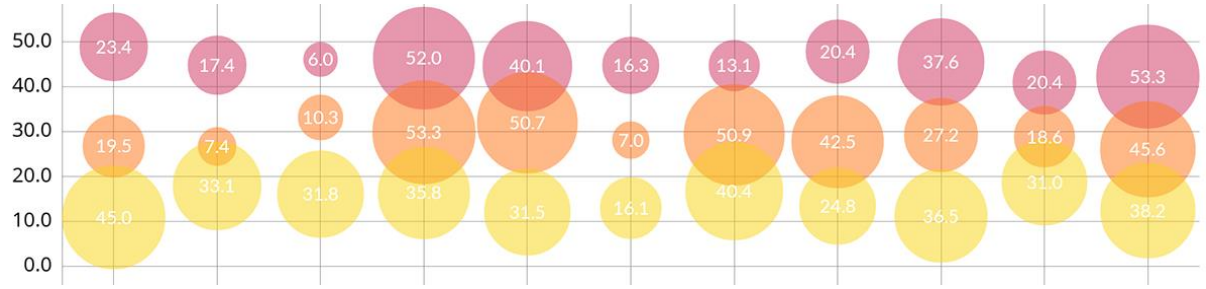
Hình 15. Ví dụ về biểu đồ hình nến

1.4.6. Biểu đồ bong bóng

```
<com.github.mikephil.charting.charts.BubbleChart
    android:id="@+id/chart"
```

```
    android:layout_width="match_parent"
android:layout_height="match_parent"/>
```

Ví dụ:

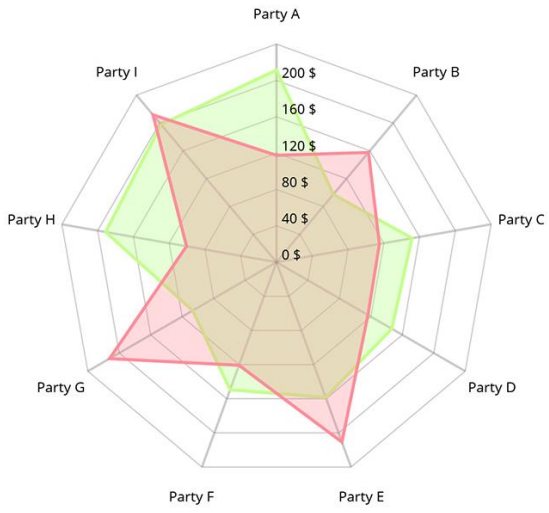


Hình 16. Ví dụ về biểu đồ bong bóng

1.4.7. Biểu đồ Radar

```
<com.github.mikephil.charting.charts.RadarChart
    android:id="@+id/chart"
    android:layout_width="match_parent"
android:layout_height="match_parent"/>
```

Ví dụ:



Hình 17. Ví dụ về biểu đồ Radar

1.5. Lý thuyết đồ thị

1.5.1. Đồ thị đa thức

Biểu thức đa thức là biểu thức được xây từ các hằng số và các ký hiệu chữ số được gọi là biến và được nối với nhau bằng các phép cộng, phép nhân. Các biến trong đa thức có thể được mũ lên số nguyên không âm. Hằng số thường là các con số nói chung, nhưng cũng có thể biểu diễn các đối tượng toán học khác cũng có thể nhân và cộng với biến và các hằng số còn lại. Hai biểu thức đa thức được gọi là biểu diễn chung một đa thức nếu một trong hai cái có thể biến đổi về cái còn lại qua việc sử dụng các tính chất của giao hoán, kết hợp và phân phối của phép cộng và phép nhân.[4]

Đa thức trong một biến x luôn có thể viết hoặc viết lại dưới dạng sau:

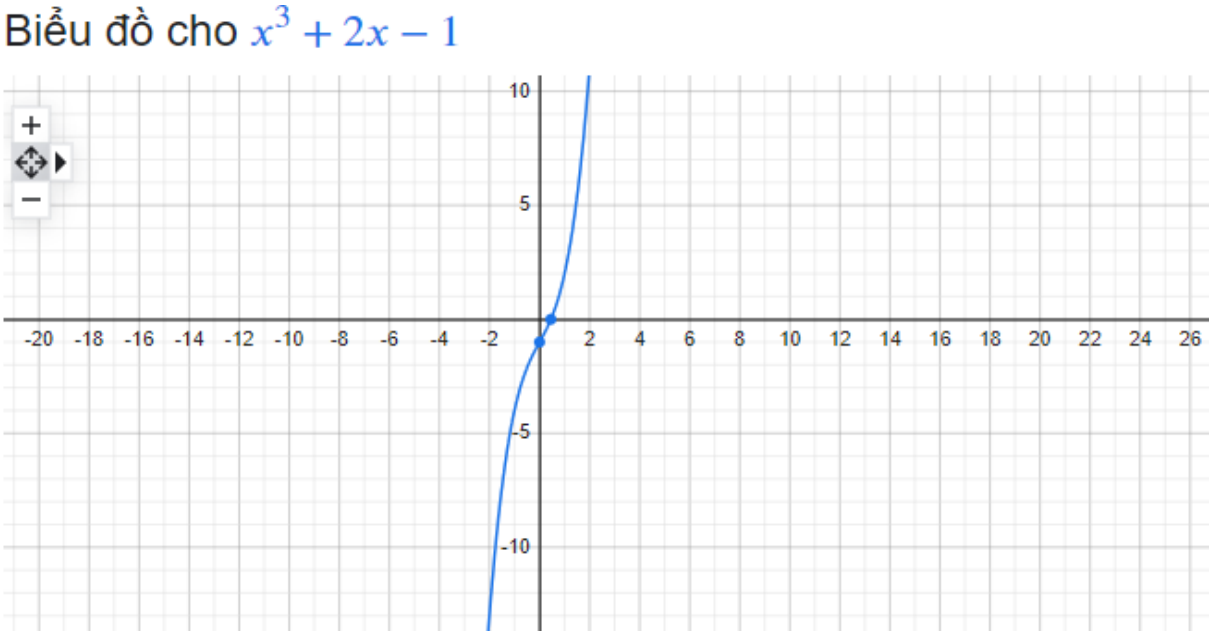
$$a_n x^n + a_{n-1} x^{n-1} + \ldots + a_2 x^2 + a_1 x + a_0,$$

trong đó $a_0,\ldots,a_n$ là các hằng số được gọi là *hệ số* của đa thức, còn x được gọi là biến số (hay nói ngắn gọn là *biến*).

Đồ thị đa thức là một biểu diễn đồ thị của một hàm đa thức trên hệ tọa độ Descartes. Một hàm đa thức có dạng tổng quát như sau:

$$P = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_2 x^2 + a_1 x + a_0$$

Ví dụ về đồ thị của một đa thức bậc 3



Hình 18. Đồ thị $y=x^3+2x-1$

1.5.2. Đồ thị hyperbol

Đường hyperbol là một trong các loại đường conic (đường cong bậc 2), được xác định bởi một tập hợp các điểm trên mặt phẳng mà hiệu tuyệt đối của khoảng cách từ mỗi điểm đến hai điểm cố định (gọi là tiêu điểm) là một hằng số. Hyperbol có hai nhánh đối xứng qua trục đối xứng của nó.

Trong chương trình toán học phổ thông có một số dạng hàm số mà đồ thị của chúng có thể tạo thành đường hyperbol như:

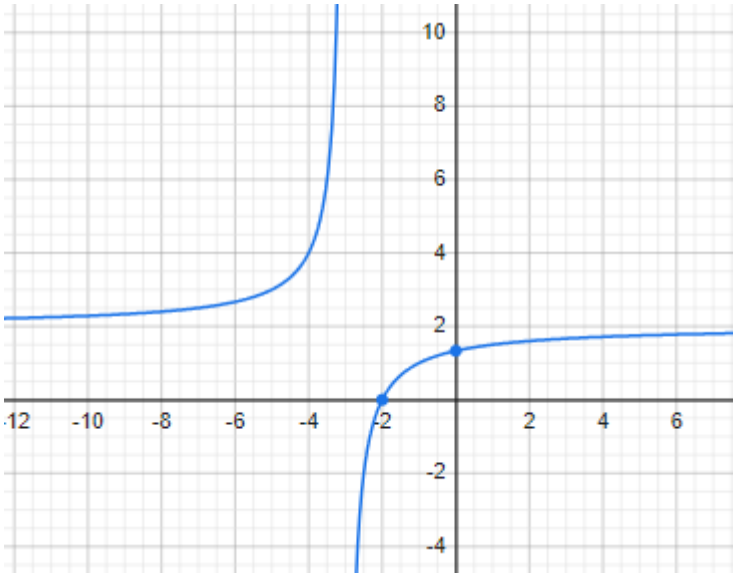
- Hàm số nghịch đảo: $y = \frac{k}{x}$ (với k là hằng số)

VD: y = 4/x

Hình 19. Đồ thị y = 4/x

- Hàm phân thức bậc nhất: $y = \frac{ax+b}{cx+b}$ $(a, b$ *là hằng số*$)$

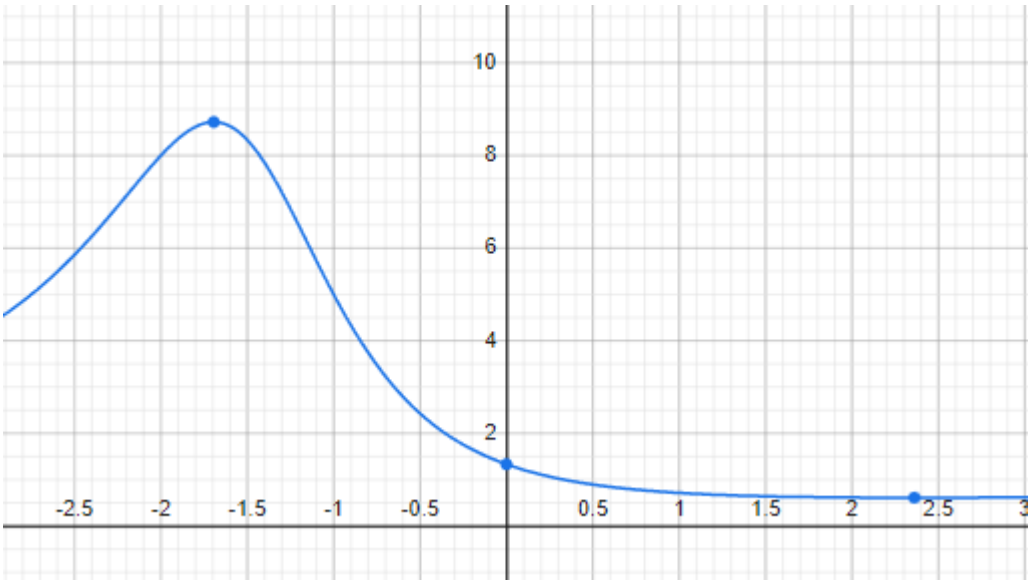
VD: Đồ thị hàm số $y = \frac{2x+4}{x+3}$



Hình 20. Đồ thị hàm số $y = \frac{2x+4}{x+3}$

- Hàm phân thức bậc hai: y = $\frac{ax^2+bx+c}{dx^2+ex+f}$

VD: Đồ thị hàm số y = $\frac{x^2+4}{x^2+3x+3}$



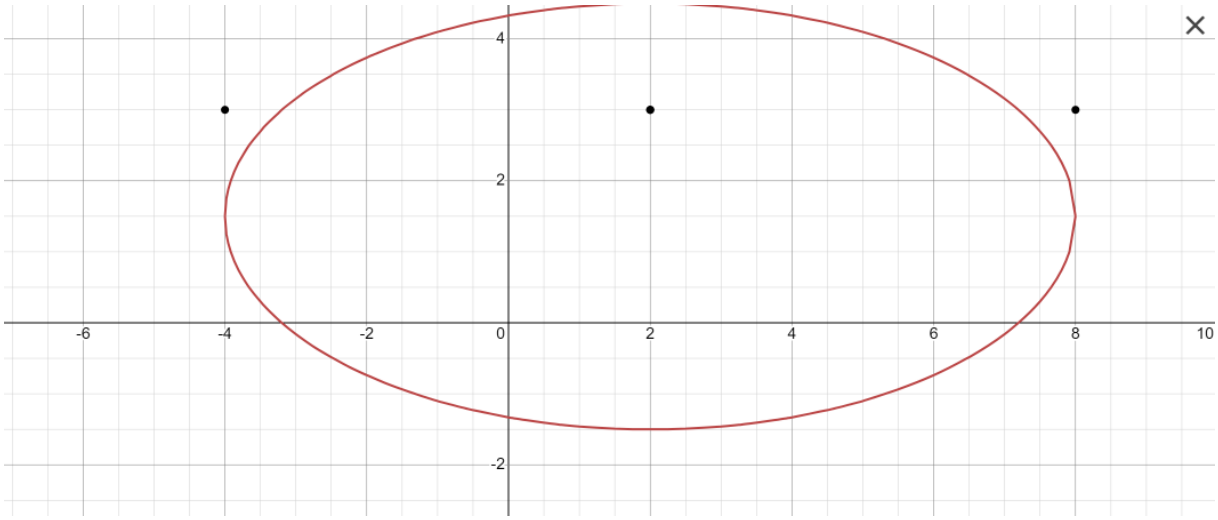
Hình 21. Đồ thị hàm số $y = \frac{x^2+4}{x^2+3x+3}$

1.5.3. *Đồ thị elip*

Đồ thị elip là dạng đồ thị tổng quát hơn của đồ thị đường tròn. Phương trình dạng:

$$(ax - b)^2 + (cy - d)^2 = r^2$$

Ví dụ đồ thị $(x-2)^2+(2y-3)^2=36$

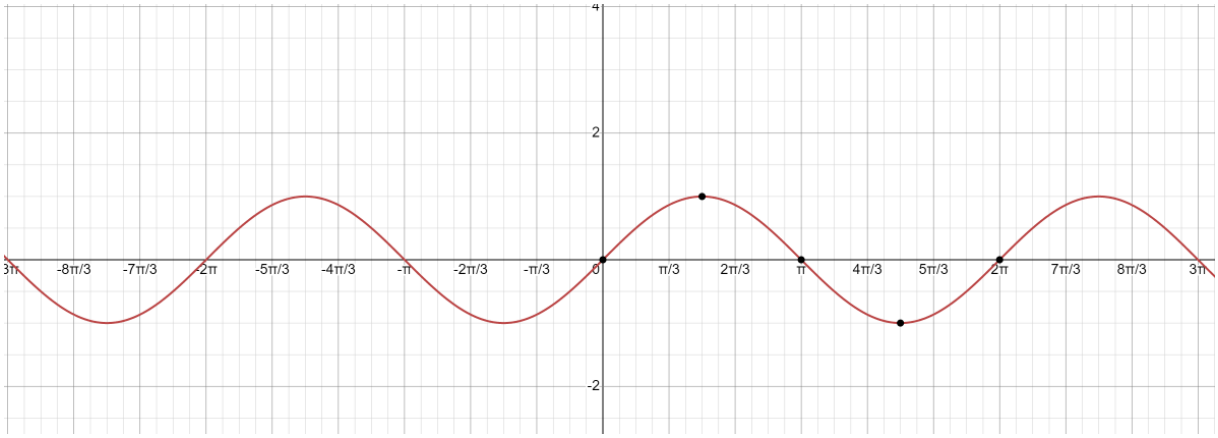


Hình 22. Đồ thị $(x-2)^2 + (2y-3)^2=36$

1.5.4. *Đồ thị lượng giác*

Đồ thị lượng giác thường liên quan đến các hàm số lượng giác như sin(x), cos(x), tan(x), v.v. Các hàm số này thường được biểu diễn trên trục hoành là góc x và trục tung là giá trị của hàm số.

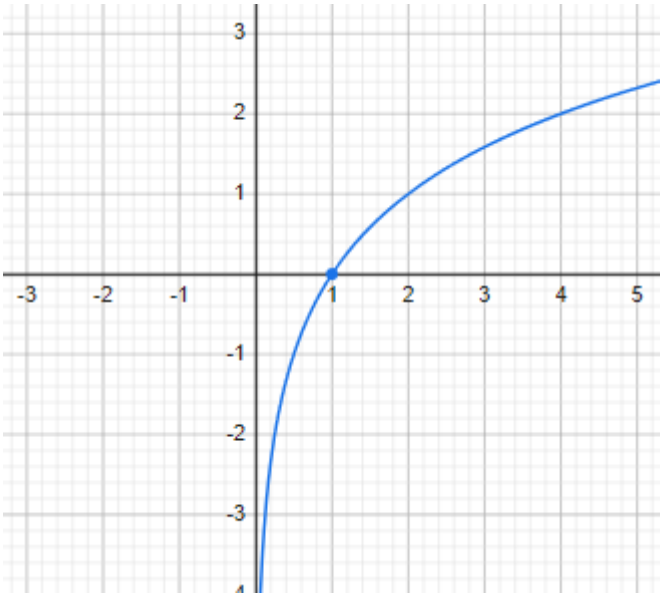
VD: y = sin x



Hình 23. Đồ thị y = sin x

1.5.5. Đồ thị logarit

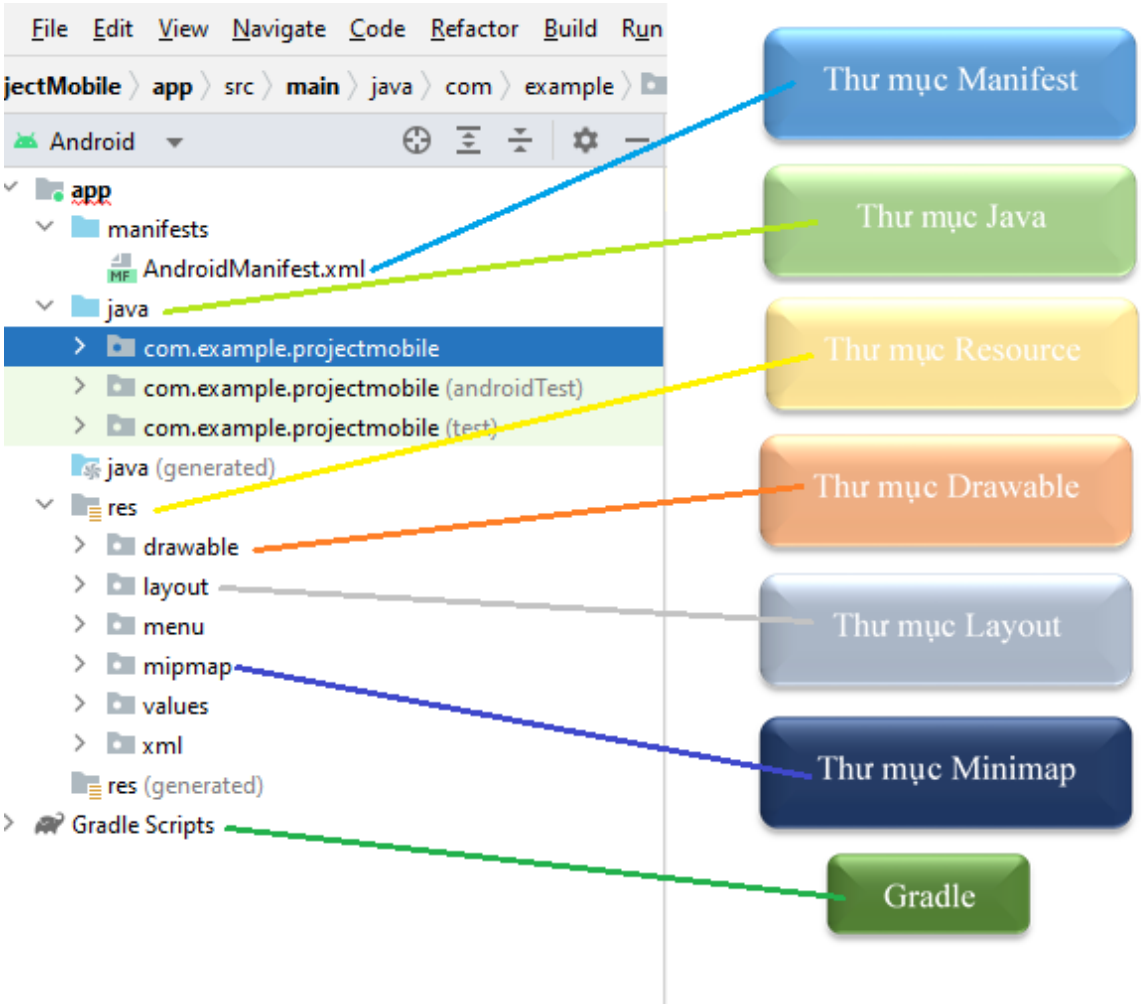
Đồ thị logarit là đồ thị biểu diễn hàm số logarith ví dụ như $y = \log_2(x)$



Hình 24. Đồ thị $y = log_2(x)$

1.6. Cấu trúc thư mục Project Android

Khi tạo một thư mục bằng Android Studio thì chúng ta sẽ nhận được cấu trúc thư mục project sẽ giống như hình dưới đây[5,6]



Hình 25. Cấu trúc một project trong Android Studio

1.6.1. Thư mục Manifest

Thư mục này sẽ chứa một file Manifest (AndroidManifest.xml) cho ứng dụng Android. Đây là file bắt buộc phải có trong project Android, nó quản lý mọi thành phần trong project bao gồm:[6]

- Các thông số của app như icon, theme,…
- App components

<activity> Activity

<service> Service: thực hiện các hoạt động chạy nền như giao dịch mạng, chơi nhạc…

<receiver> BroadcastReceiver: dùng để lắng nghe các sự kiện của thiết bị và của app

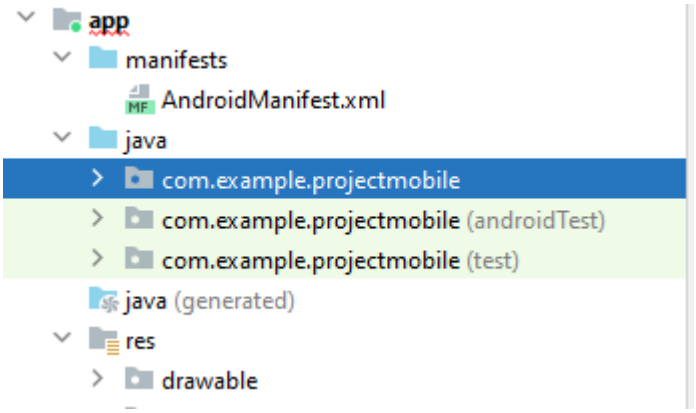
<provider> ContentProvider: được dùng để chia sẽ dữ liệu giữa các app

- Các quyền truy cập của app như quyền truy cập danh bạ, quyền lấy thông tin vị trí…

- Phần cứng và phần mềm mà app yêu cầu như la bàn kế.

1.6.2. Thư mục Java

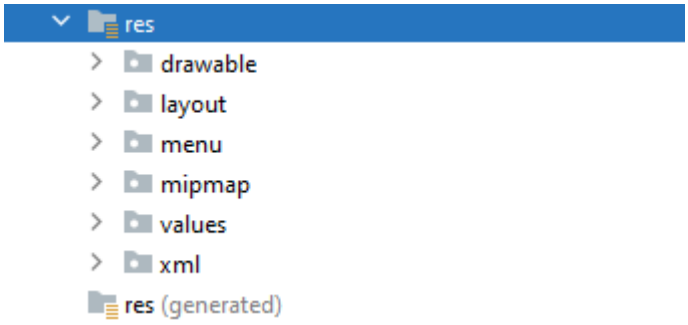
Thư mục này sẽ chứa tất cả các file mã nguồn java (.java) mà chúng ta sẽ tạo trong quá trình phát triển ứng dụng. Bất cứ khi nào tạo bất kỳ project / ứng dụng mới nào, file lớp MainActivity.java sẽ tự động tạo trong package, ví dụ com.example.projectmobile, giống như dưới đây:[6]



Hình 26. Thư mục Java

1.6.3. Thư mục res (Resources)

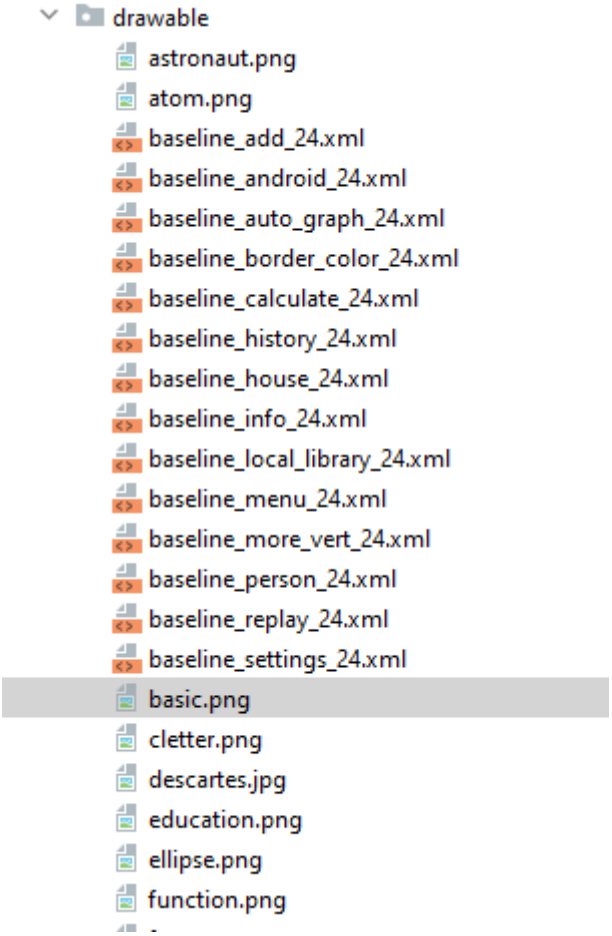
Đây là một thư mục quan trọng sẽ chứa tất cả các resource không phải code, chẳng hạn như ảnh bitmap, UI strings, XML layouts như hiển thị bên dưới.[6]



Hình 27. Thư mục res

1.6.3.1. Thư mục drawable:

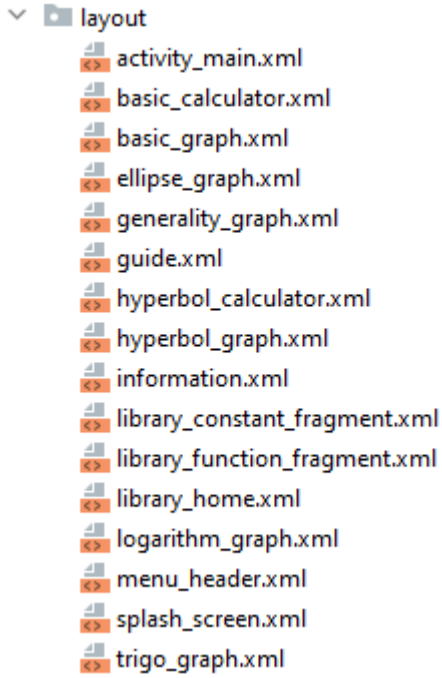
Thư mục drawable chứa tất cả file hình ảnh được sử dụng trong ứng dụng. Gồm file XML: chứa mô tả hoặc định nghĩa cho các hình ảnh vector, animation, hoặc các thuộc tính tùy chỉnh khác trong giao diện người dùng và các file có đuôi .png hoặc .jpg.[6]



Hình 28. Thư mục drawable

1.6.3.2. Thư mục Layout

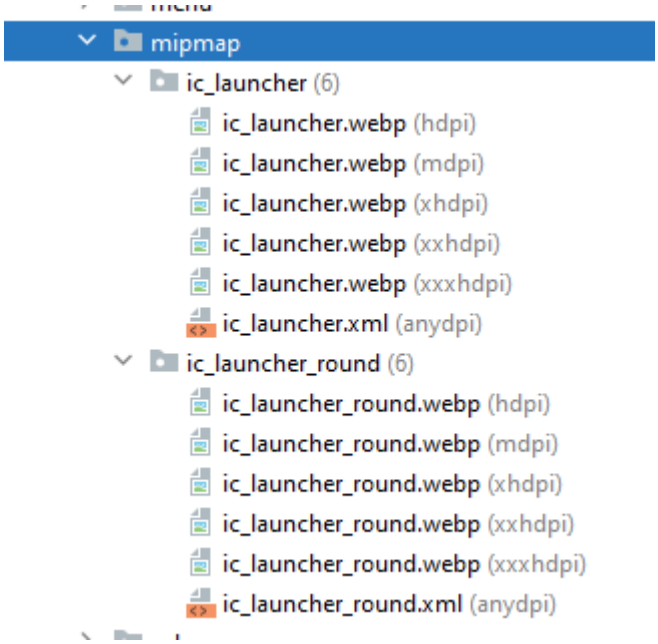
Thư mục này sẽ chứa tất cả các file XML layout đã sử dụng để xác định Giao diện người dùng của ứng dụng.[6]



Hình 29. Thư mục layout

1.6.3.3. Thư mục minimap

Thư mục này sẽ chứa các biểu tượng ứng dụng/laucher được sử dụng để hiển thị trên màn hình chính. Các loại biểu tượng sẽ có tỷ trọng khác nhau như hdpi, mdpi, xhdpi, xxhdpi, xxxhdpi, để sử dụng dựa trên kích thước của thiết bị.

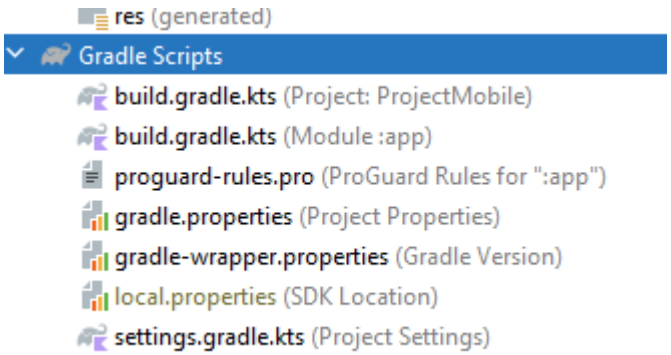


Hình 30. Thư mục minimap

1.6.4. Gradle Script

Trong Android, Gradle là công cụ build hệ thống và Gradle được tích hợp sẵn vào Android Studio, và được điều khiển một cách tự động thông qua Android Studio.

Trong gradle có build.gradle (Project) và build.gradle (Module) được sử dụng để build các cấu hình áp dụng cho tất cả các module ứng dụng hoặc dành riêng cho một mô-đun ứng dụng.



Hình 31. Thư mục Gradle Script

1.7. Các thành phần của một Activity

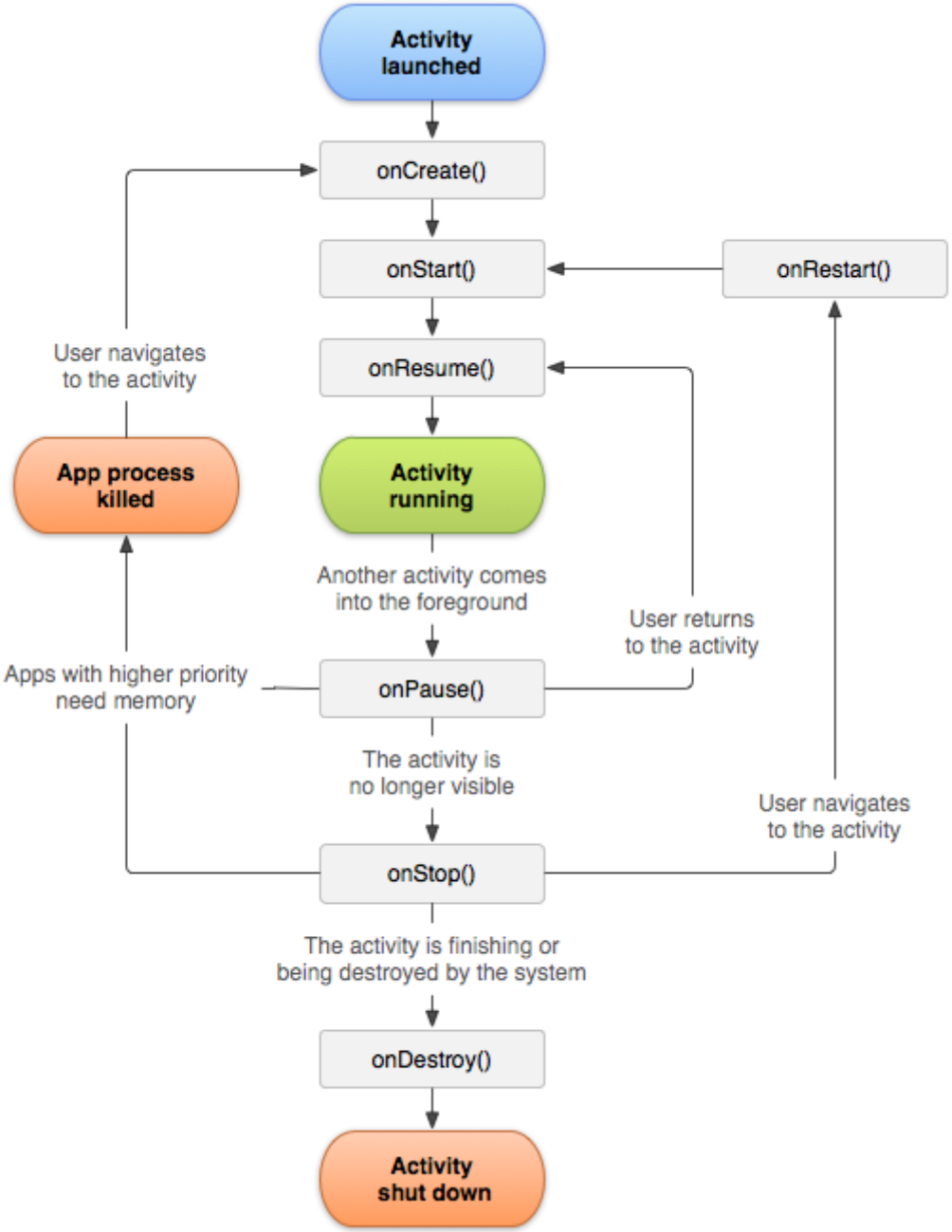
1.7.1. Activity

Một Activity trong Android là một màn hình giao diện người dùng của ứng dụng Android. Một ứng dụng Android có thể chứa một hoặc nhiều Activity, nghĩa là một hoặc nhiều màn hình giao diện. Ứng dụng Android bắt đầu bằng cách hiển thị Activity

chính (Main Activity) và có thể mở thêm nhiều màn hình Activity khác tuỳ theo chức năng của ứng dụng.

Vòng đời của một Activity:

Vòng đời Activity trong Android là chuỗi các trạng thái mà một Activity trải qua từ khi khi nó được tạo ra đến khi kết thúc.[7]



Hình 32. Vòng đời của một Activity

- **onCreate():** Được gọi khi Activity được tạo ra.
- **onStart():** Được gọi khi Activity được hiển thị trên màn hình.

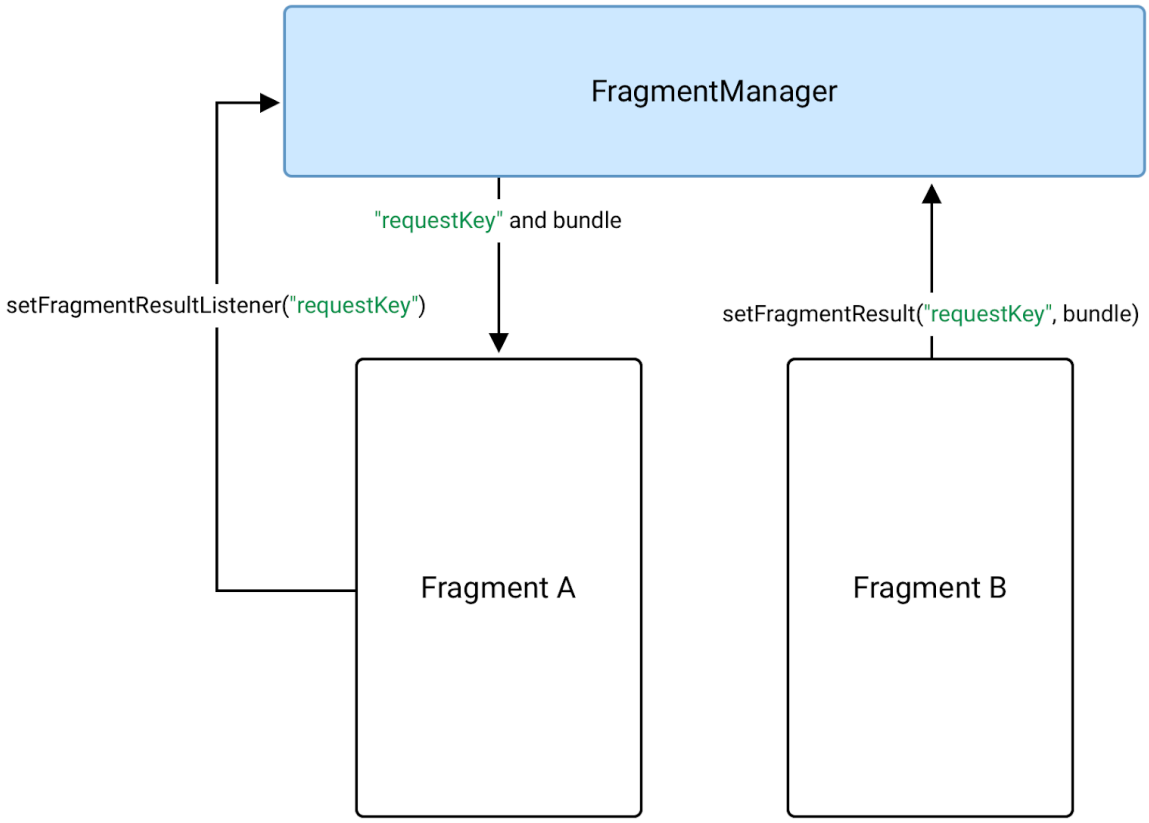
- **onResume():** Được gọi khi Activity được hiển thị trên màn hình và trở thành hoạt động chính.
- **onPause():** trạng thái Activity vẫn đang chạy, người dùng vẫn nhìn thấy, nhưng Activity bị che một phần.
- **onStop():** Được gọi khi Activity bị che khuất hoàn bởi một Activity khác hoặc bị dừng lại hoàn toàn.
- **onDestroy():** Được gọi khi Activity bị phá hủy.
- **onRestart():** Được gọi khi Activity đã bị dừng lại và được khởi động lại.

1.7.2. Fragment

Là một thành phần trong giao diện màn hình.[7]

Được xác định và quản lý bố cục riêng, có vòng đời riêng và có thể xử lý các sự kiện đầu vào riêng.

Không thể đứng riêng được mà cần nằm trong một Activity.

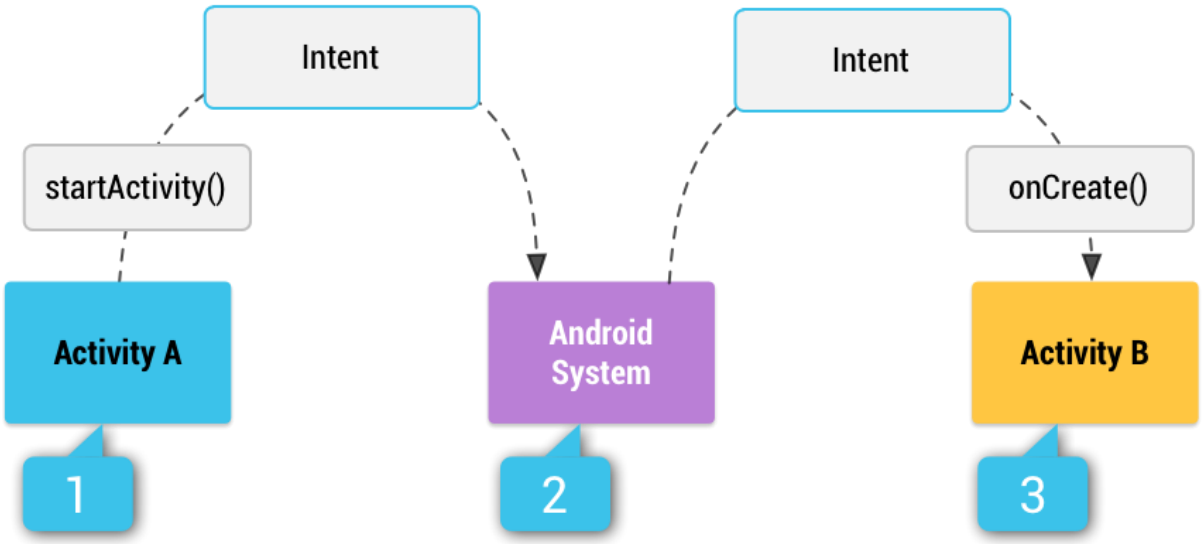


Hình 33. Nguyên lý hoạt động của Fragment

1.7.3. Intent

Là một đối tượng được sử dụng để chuyển tiếp dữ liệu giữa các thành phần khác nhau của ứng dụng Android, bao gồm các Activity, Service, Broadcast Receiver và Content Provider.

Intent có thể được sử dụng để thực hiện các nhiệm vụ như khởi động một Activity mới, gửi tin nhắn Broadcast, hoặc khởi chạy một dịch vụ.



Hình 34. Ví dụ đơn giản của Intent

Có hai loại:

- **Implicit Intent**: để khởi chạy một thành phần không cụ thể của ứng dụng, ví dụ như khởi chạy một trình duyệt web.

Ví dụ:

```
Uri webpage = Uri.parse("https://www.example.com");
Intent intent = new Intent(Intent.ACTION_VIEW, webpage);
if (intent.resolveActivity(getPackageManager()) != null) {
    startActivity(intent);
}
```

Hình 35. Ví dụ về Implicit Intent

- **Explicit Intent**: Đây là loại Intent được sử dụng để khởi chạy một thành phần cụ thể của ứng dụng, ví dụ như khởi động một Activity mới hoặc một Service.

Ví dụ:

```
Intent intent = new Intent(MainActivity.this, SecondActivity.class);
intent.putExtra("message", "Hello from MainActivity");
startActivity(intent);



String message = getIntent().getStringExtra("message");
```

Hình 36. Ví dụ về Explicit Intent

CHƯƠNG 2. THIẾT KẾ HỆ THỐNG

2.1. Yêu cầu người dùng

Khi phát triển ứng dụng "Vẽ đồ thị hàm số", việc hiểu rõ và đáp ứng các yêu cầu của người dùng là một yếu tố then chốt để đảm bảo ứng dụng mang lại giá trị thực sự và được sử dụng rộng rãi. Dưới đây là các yêu cầu chính từ người dùng mà ứng dụng cần đáp ứng:

2.1.1. Yêu cầu chức năng

- Vẽ đồ thị hàm số: Ứng dụng phải có khả năng vẽ đồ thị của các hàm số thường gặp trong chương trình Toán học phổ thông như hàm bậc nhất, bậc hai, bậc ba, hàm mũ, hàm logarit, và hàm lượng giác.
- Tính toán giao điểm: Ứng dụng cần cung cấp chức năng tính toán và hiển thị giao điểm của các hàm số một cách chính xác.
- Chỉnh sửa và tùy biến đồ thị: Người dùng cần khả năng chỉnh sửa các thông số của hàm số và tùy biến giao diện đồ thị, chẳng hạn như thay đổi màu sắc, kiểu đường, và đơn vị trên trục tọa độ.
- Lưu và chia sẻ đồ thị: Ứng dụng nên hỗ trợ tính năng lưu trữ đồ thị đã vẽ và cho phép người dùng chia sẻ chúng qua email hoặc các mạng xã hội.

2.1.2. Yêu cầu phi chức năng

- Giao diện thân thiện và dễ sử dụng: Ứng dụng phải có giao diện đơn giản, trực quan, giúp người dùng dễ dàng thao tác mà không cần hướng dẫn phức tạp.

- Hiệu suất cao: Ứng dụng cần hoạt động mượt mà, xử lý và hiển thị đồ thị nhanh chóng, không gặp phải hiện tượng giật lag, đặc biệt khi vẽ các đồ thị phức tạp.

- Tương thích với nhiều thiết bị: Ứng dụng phải tương thích với nhiều phiên bản hệ điều hành Android khác nhau và hoạt động tốt trên nhiều loại thiết bị từ điện thoại thông minh đến máy tính bảng.

- Bảo mật và an toàn dữ liệu: Ứng dụng phải đảm bảo an toàn dữ liệu người dùng, không để lộ thông tin cá nhân và bảo mật thông tin các đồ thị đã vẽ và lưu trữ.

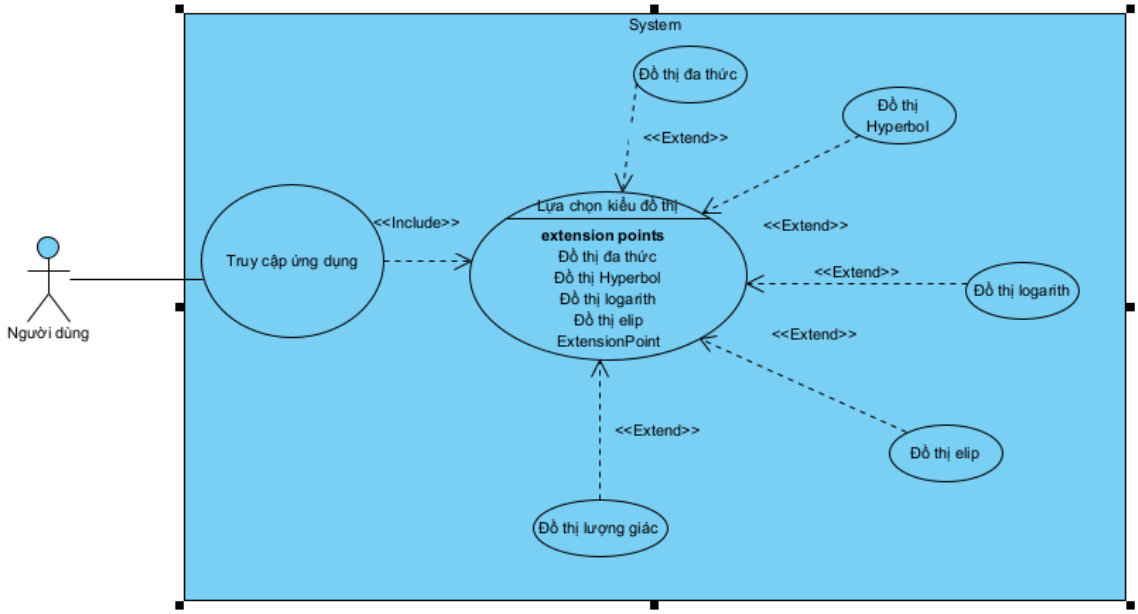
2.1.3. Yêu cầu về trải nghiệm người dùng (UX)

- Hướng dẫn sử dụng: Ứng dụng cần cung cấp các hướng dẫn rõ ràng, dễ hiểu để người dùng, đặc biệt là những người mới, có thể nhanh chóng làm quen và sử dụng hiệu quả.
- Phản hồi nhanh: Ứng dụng nên cung cấp phản hồi nhanh chóng và rõ ràng cho các thao tác của người dùng, chẳng hạn như khi nhập liệu, vẽ đồ thị hoặc tính toán giao điểm.
- Tính năng hỗ trợ và trợ giúp: Có sẵn mục trợ giúp hoặc hướng dẫn sử dụng trong ứng dụng để hỗ trợ người dùng khi gặp khó khăn hoặc cần giải đáp thắc mắc.

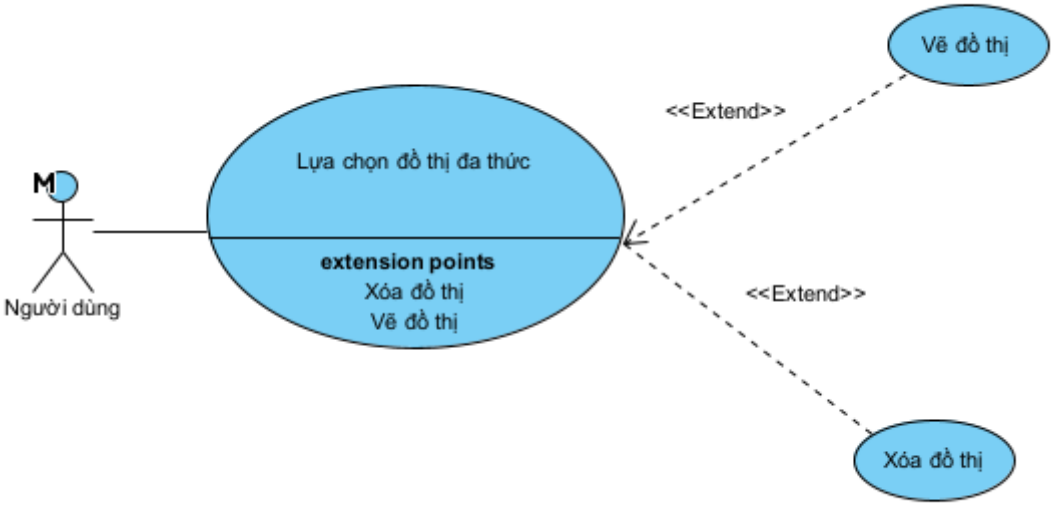
2.1.4. Yêu cầu về tính tương tác

- Tương tác trực tiếp với đồ thị: Người dùng cần có khả năng tương tác trực tiếp với đồ thị, chẳng hạn như phóng to, thu nhỏ, kéo, thả và xem giá trị cụ thể tại từng điểm trên đồ thị.
- Cập nhật theo thời gian thực: Khi người dùng thay đổi thông số của hàm số, đồ thị phải được cập nhật theo thời gian thực để hiển thị kết quả tức thì.

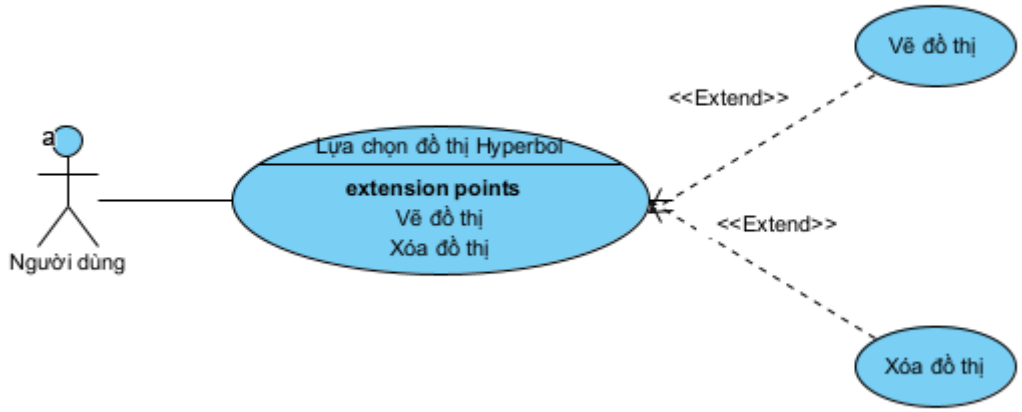
2.2. Phân tích thiết kế



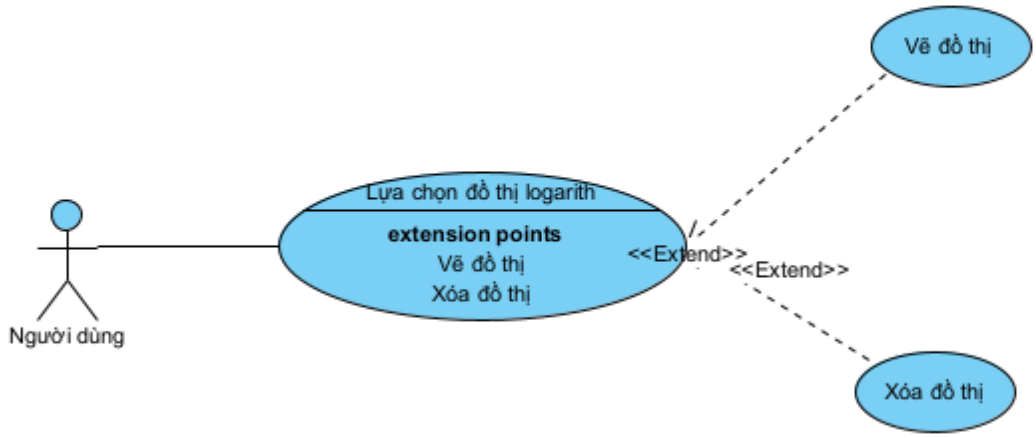
Hình 37. Biểu đồ Use Case ứng dụng



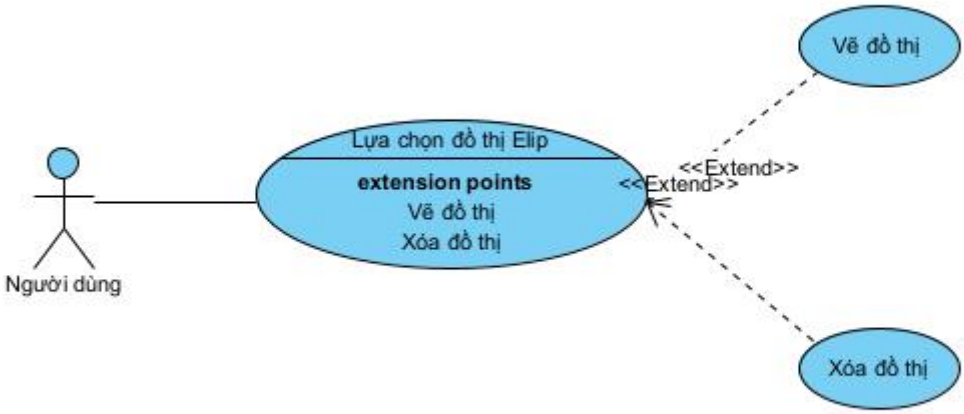
Hình 38. Usecase đồ thị đa thức



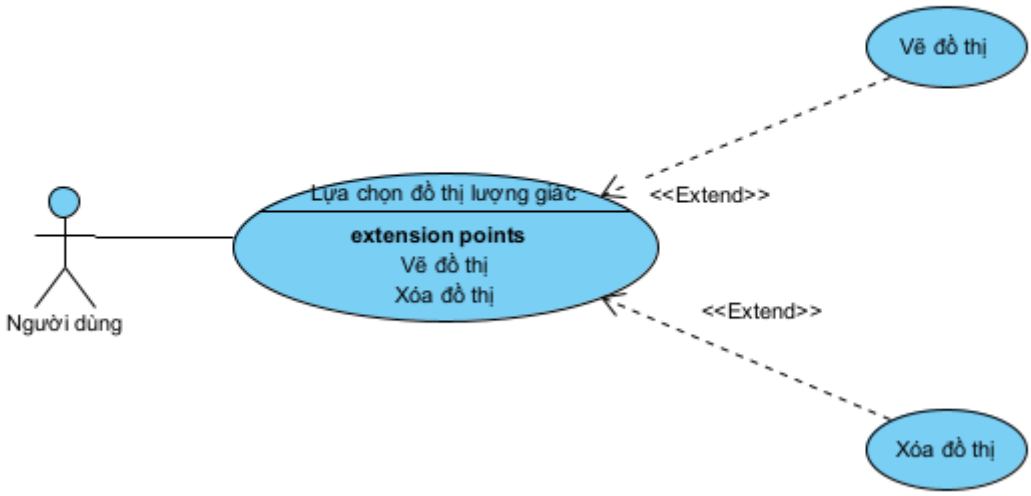
Hình 39. Usecase đồ thị Hyperbol



Hình 40. Usecase đồ thị logarith



Hình 41. Usecase đồ thị Elip



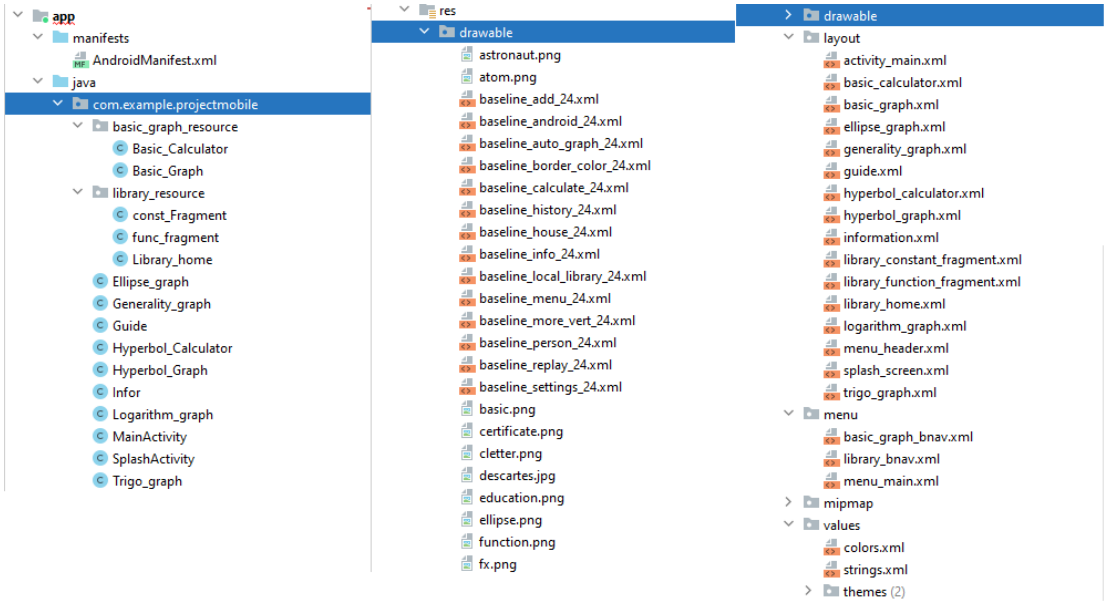
Hình 42. Usecase đồ thị lượng giác

2.3. Thiết kế chi tiết

2.2.1. Cấu trúc thư mục

Bảng 1. Cấu trúc ứng dụng Vẽ đồ thị hàm số

Layout			Chức năng tương đương (Backend)
activity_main			MainActivity
Fragment	basic_graph.xml		Basic_graph.java
	hyperbol_graph.xml		Hyperbol_Graph.java
	ellipse_graph.xml		Ellipse_graph.java
	logarithm_graph.xml		Logarithm_graph.java
	trigo_graph.xml		Trigo_graph.java
	generality_graph.xml		Generality_graph.java
	Library	Library_home.xml	Library_home.java
		Library_constrant_fragment.xml	Library_constrant_fragment.java
		Library_function_fragment.xml	Library_function_fragment.java

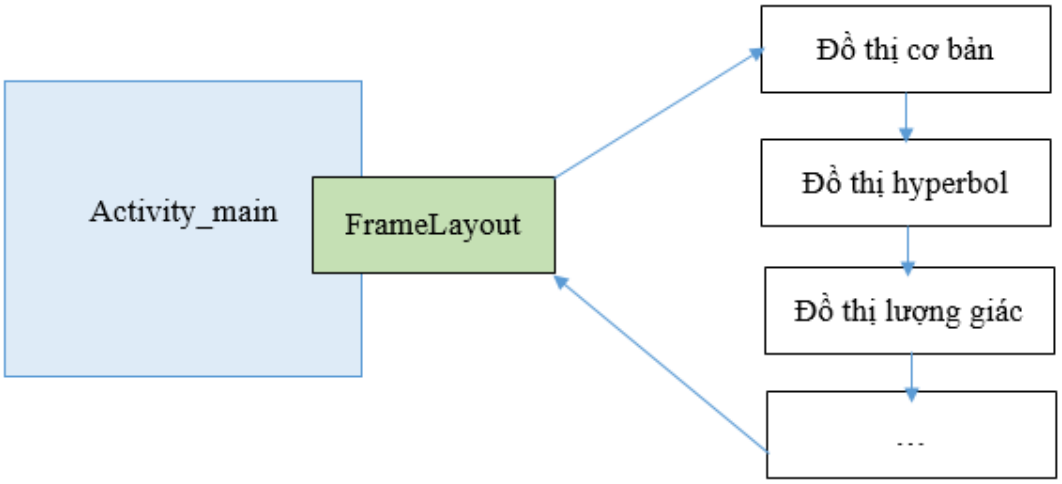


Hình 43. Thư viện java, drawable và layout của Project

2.2.2. Thiết kế giao diện

Để có thể trình bày được hết các loại đồ thị có thể vẽ, ta sử dụng component fragment để xây dựng giao diện linh hoạt và đầy đủ hơn.

Cụ thể, ta xây dựng một activity_main làm layout gốc, từ layout này, ta đặt một component FrameLayout nhằm hoán đổi component này với từng trang giao diện mới, với những loại đồ thị mới.



Hình 44. Fragment

2.2.3. Quy trình thực hiện

Vậy, từ thư viện MPAndroidChart với một khả năng cung cấp một loạt các biểu đồ đủ mọi hình dáng khác nhau, làm thế nào để có thể vẽ những đồ thị hàm số tùy ý? Câu trả lời đó là ta sẽ vẽ những biểu đồ đường được nối bởi hàng loạt các điểm thỏa mãn hàm số đó.

Các bước nhìn chung có thể trình bày như sau:

- Ta gọi các thành phần editText đóng vai trò làm biến số nhập vào dưới dạng một đối tượng. Cái editText có thể biến đổi liên tục tùy theo dữ liệu nhập vào.

  VD:

  - Khai báo các editText

```
private EditText editText1, editText2, editText3, editText4, editText5;
```

  - Đặt các đối tượng này kết nối với các editText ở giao diện.

```
editText1 = view.findViewById(R.id.parameter1);
editText2 = view.findViewById(R.id.parameter2);
editText3 = view.findViewById(R.id.parameter3);
editText4 = view.findViewById(R.id.parameter4);
editText5 = view.findViewById(R.id.parameter5);
```

- Tạo một mảng chứa hàng loại các cặp số x, y tương ứng với tọa độ của hàng loạt các điểm tạo nên đồ thị.

VD đối với đồ thị hàm số cơ bản:

```
//4.2. Tạo mảng chứa tập hợp các điểm hình thành nên đồ thị
List<Entry> entries = new ArrayList<>();
for (float x = -65; x <= 65; x += 0.01) {
    float y = p1 * x * x * x * x + p2 * x * x * x + p3 * x * x + p4 * x + p5;
    entries.add(new Entry(x, y));
}
```

- Tạo một dataset để chứa các mảng trên, từ đó setup thuộc tính của đồ thị để cho dataset này hiển thị dưới dạng hình vẽ.

```
lineData.addDataSet(dataSet);
lineChart.setData(lineData);
```

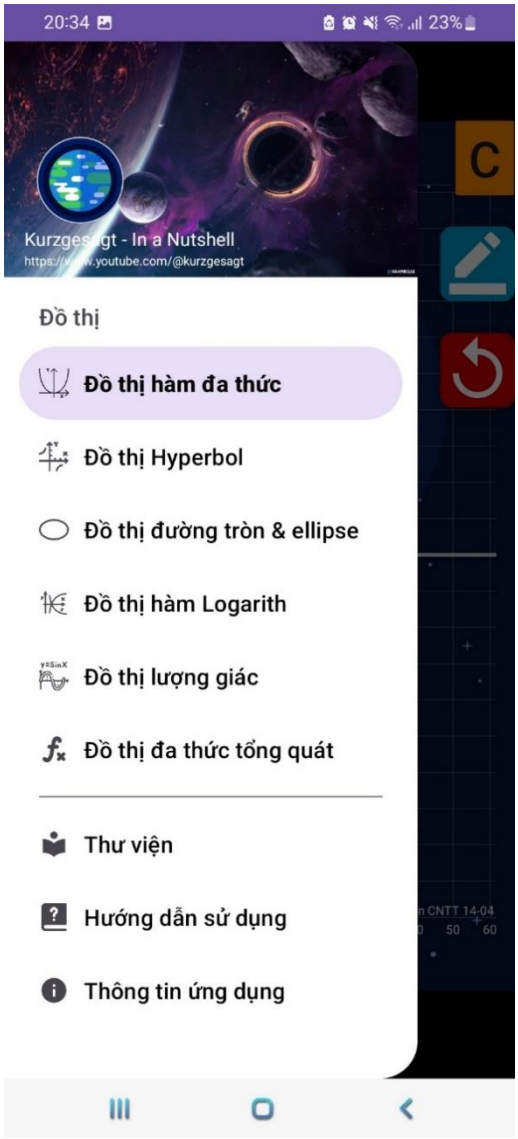
CHƯƠNG 3. KẾT QUẢ ĐẠT ĐƯỢC

3.1. Splash screen (màn hình khởi động ứng dụng)

Hình 45. Màn hình khởi chạy

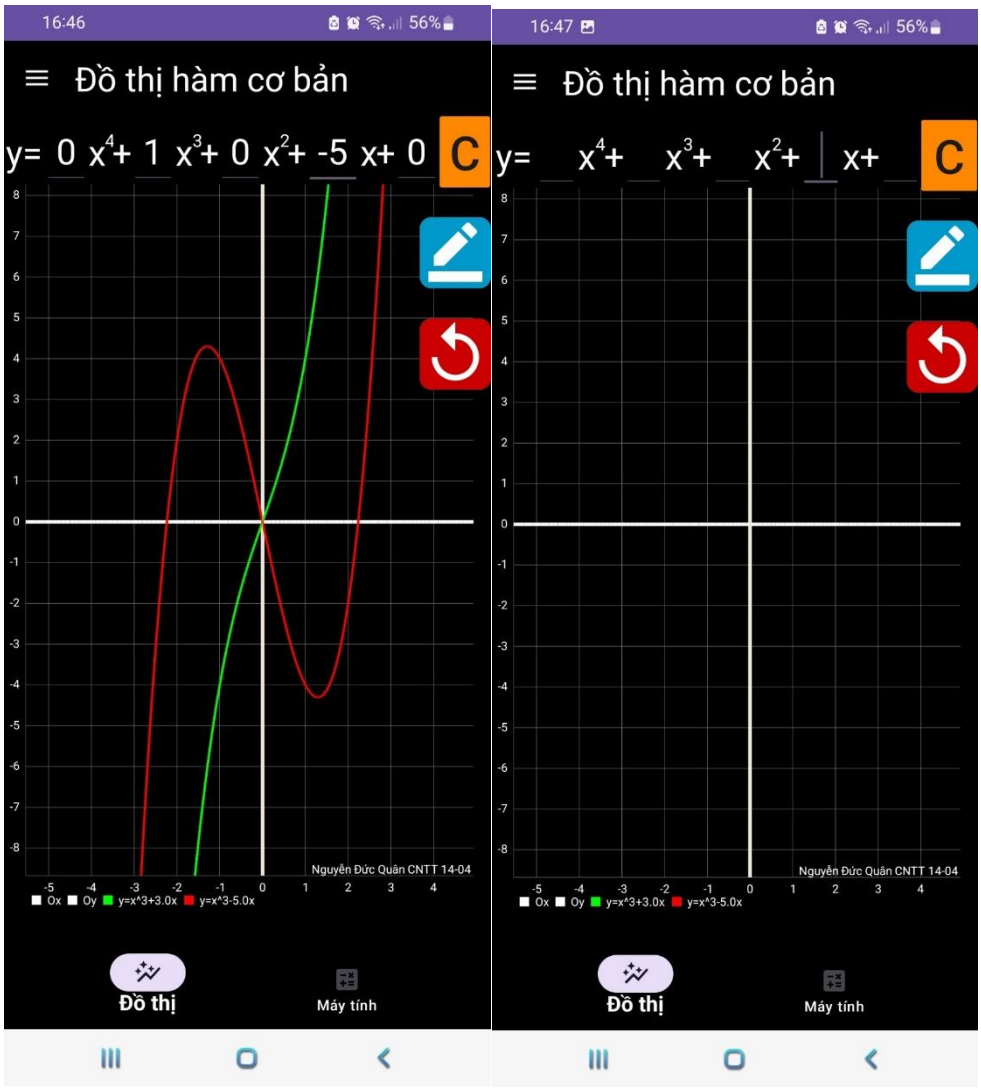
3.2. Giao diện vẽ đồ thị

3.2.1. Thanh điều hướng

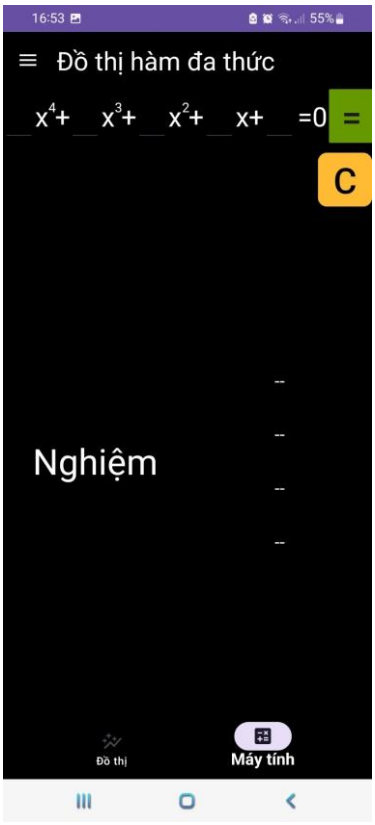


Hình 46. Thanh điều hướng

3.2.2. Đồ thị hàm đa thức

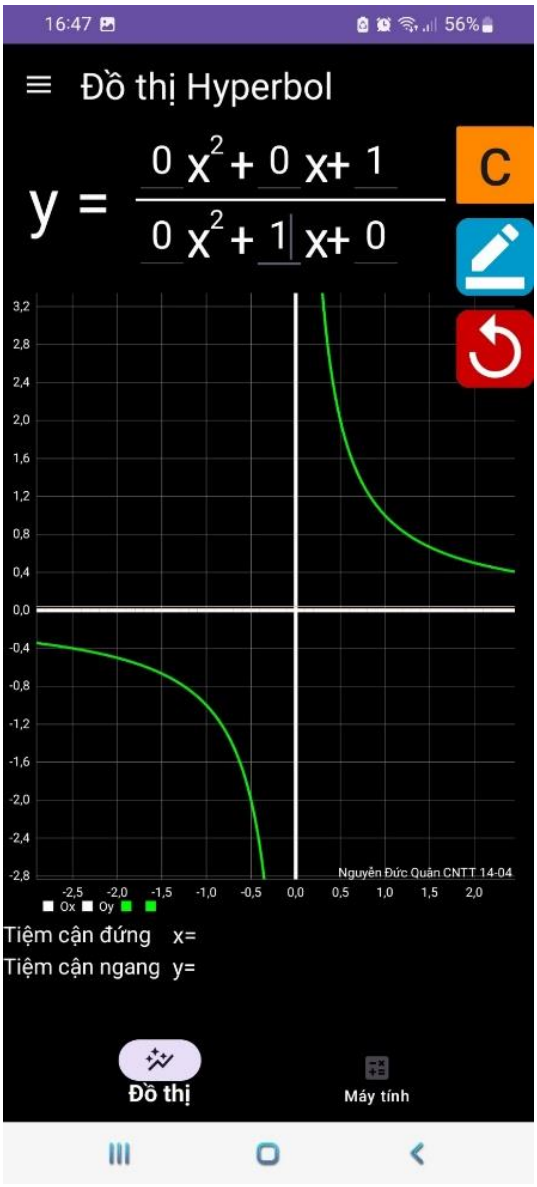


Hình 47. Đồ thị hàm đa thức



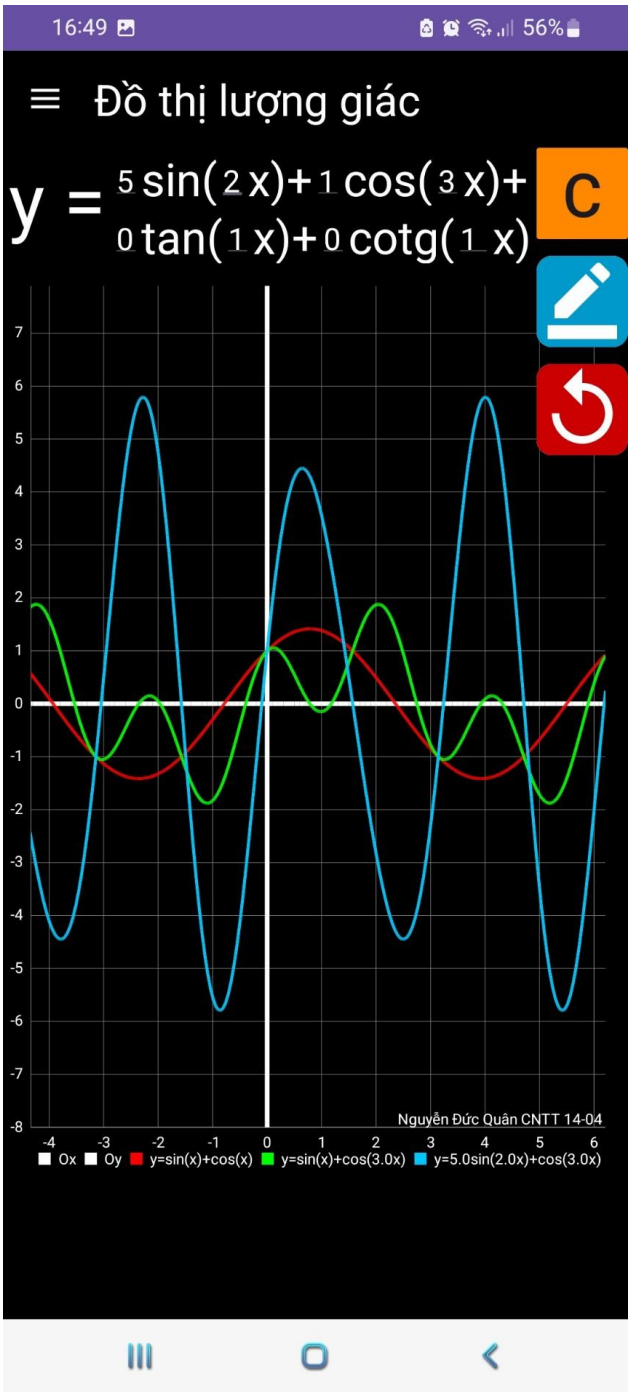
Hình 48. Máy tính

3.2.3. Đồ thị Hyperbole



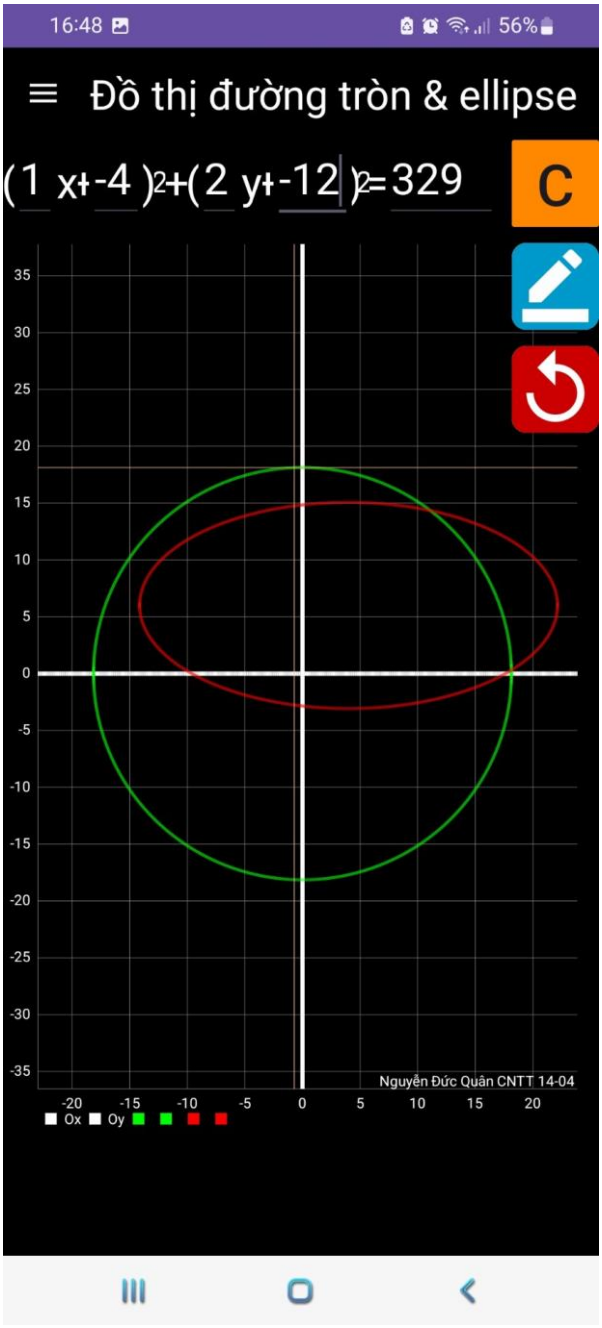
Hình 49. Đồ thị hyberbol

3.2.4. Đồ thị lượng giác



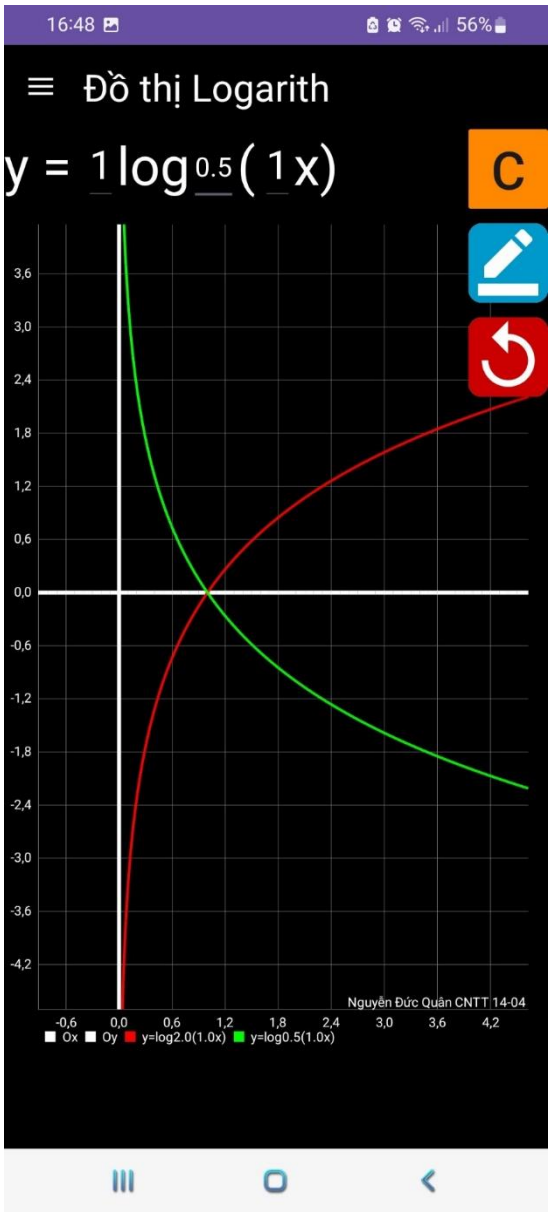
Hình 50. Đồ thị lượng giác

3.2.5. Đồ thị elip



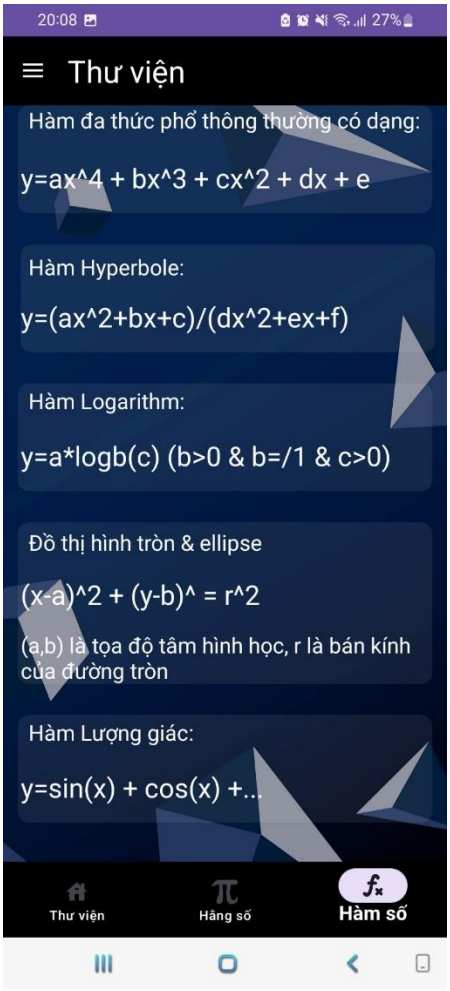
Hình 51. Đồ thị elip

3.2.6. Đồ thị logarith



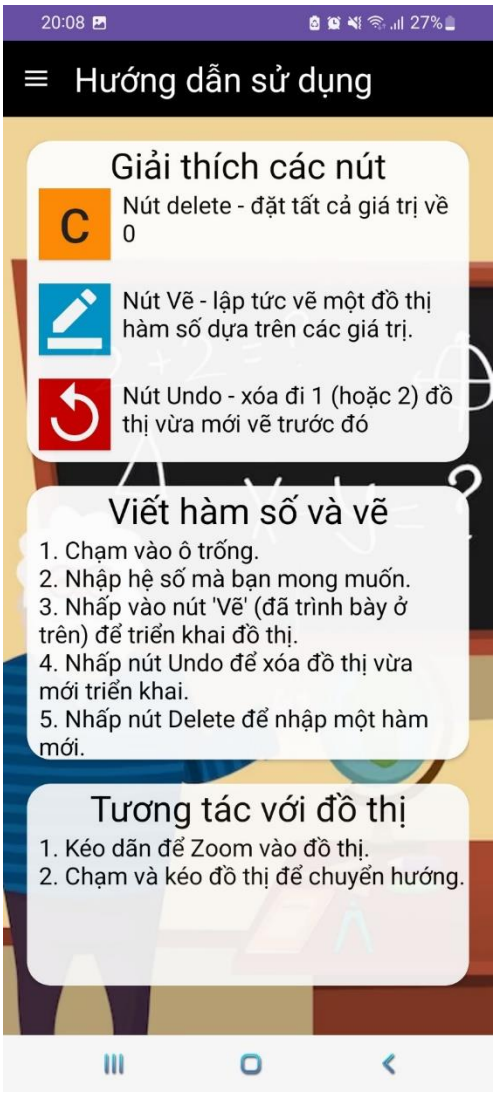
Hình 52. Đồ thị logarith

3.4.1. Thư viện



Hình 53. Giao diện thư viện

3.4.2. Hướng dẫn sử dụng



Hình 54. Hướng dẫn sử dụng

KẾT LUẬN

4.1. Kết quả thu được

Ứng dụng "Vẽ đồ thị hàm số" về cơ bản đã hoàn thành kỳ vọng ban đầu. Ứng dụng đã có thể làm được những việc sau:

- Vẽ được gần chính xác đồ thị hàm số của những hàm số thường xuất hiện trong chương trình Toán học phổ thông.
- Có chức năng tính toán giao điểm của hàm đa thức cơ bản
- Giao diện đẹp mắt, dễ sử dụng cho người mới.

4.2. Hạn chế

Ứng dụng vẫn còn một số hạn chế sau:

- Do hạn chế về mặt logic lập trình, một số đồ thị vẽ vẫn còn lỗi và không chính xác với dự kiến.
- Chưa có chức năng lưu lại những đồ thị đã vẽ.

TÀI LIỆU THAM KHẢO

[1]. GP Coder, "Java là gì? Tổng quan về ngôn ngữ lập trình java", *TOPDEV*, truy cập ngày 10/4/2023, từ: https://topdev.vn/blog/tong-quan-ve-ngon-ngu-lap-trinh-java

[2]. Nhiều tác giả, "Kotlin (ngôn ngữ lập trình)", *WIKIPEDIA*, truy cập ngày 10/4/2023, từ: https://vi.wikipedia.org/wiki/Kotlin_(ngôn_ngữ_lập_trình)

[3]. ITNavi, "Kotlin là gì? Ưu điểm nổi bật của ngôn ngữ lập trình Kotlin", *ITNavi*, truy cập ngày 10/4/2023, từ: https://itnavi.com.vn/blog/kotlin-la-gi-va-uu-diem-cua-ngon-ngu-lap-trinh-kotlin

[4]. Huy Nguyễn (2017), "Hướng dẫn tạo biểu đồ bằng thư viện MPAndroidChart", Viblo, truy cập ngày 11/4/2023, từ: https://viblo.asia/p/huong-dan-tao-bieu-do-bang-thu-vien-mpandroidchart-bWrZneWbKxw

[5]. Poly's Blog, "Tìm hiểu cấu trúc thư mục project Android", *Trường Cao đẳng FPT Polytechnic*, truy cập ngày 10/4/2023, từ: https://caodang.fpt.edu.vn/tin-tuc poly/blog/tim-hieu-cau-truc-thu-muc-project-android.html

[6]. Phạm Trung Dũng (2023), giáo trình *Cấu tạo của project Android*, Trường Đại học Phenikaa.

[7]. Phạm Trung Dũng (2023), giáo trình *Lập trình di động,* Trường Đại học Phenikaa.