HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA CÔNG NGHỆ THÔNG TIN 1

----- 🕮 -----

ĐỒ ÁN TỐT NGHIỆP

ĐỀ TÀI:
PHÁT TRIỂN ỨNG DỤNG QUẢN LÝ TÀI CHÍNH CÁ NHÂN

Giảng viên hướng dẫn	**TS. Dương Trần Đức**
Lớp	**D20CNPM03**
Họ và tên	**Nguyễn Trác Năng**
Khoá	**2020 – 2025**
Hệ	**Đại học chính quy**

Hà Nội, tháng 12/2024

NHẬN XÉT, ĐÁNH GIÁ, CHO ĐIỂM
(CỦA GIẢNG VIÊN PHẢN BIỆN)

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Điểm:.....(bằng chữ.....)

Đồng ý/Không đồng ý cho sinh viên bảo vệ trước hội đồng chấm đồ án tốt nghiệp?

Hà Nội, ngày....tháng....năm 2023

Giảng viên phản biện

(Ký và ghi rõ họ tên)

NHẬN XÉT, ĐÁNH GIÁ, CHO ĐIỂM
(CỦA GIẢNG VIÊN HƯỚNG DẪN)

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Điểm:.....(bằng chữ.....)

Đồng ý/Không đồng ý cho sinh viên bảo vệ trước hội đồng chấm đồ án tốt nghiệp?

Hà Nội, ngày....tháng....năm 2023

Giảng viên hướng dẫn

(Ký và ghi rõ họ tên)

LỜI CẢM ƠN

Với lòng kính trọng và biết ơn sâu sắc, lời đầu tiên chúng em xin gửi lời cảm ơn chân thành đến các thầy cô của Học Viện Công Nghệ Bưu Chính Viễn Thông, đặc biệt là các thầy cô Khoa Công Nghệ Thông Tin 1, những người đã chia sẻ kiến thức, kinh nghiệm và hỗ trợ không ngừng nghỉ, giúp đỡ em trong suốt quá trình học tập vừa qua.

Tiếp theo, em xin gửi lời cảm ơn chân thành và biết ơn sâu sắc đến thầy TS. Dương Trần Đức đã dành thời gian để hướng dẫn, chỉ bảo và đề xuất hướng giải quyết vấn đề khi em gặp khó khăn trong suốt quá trình thực hiện đồ án này.

Cuối cùng, không thể thiếu lòng biết ơn tới gia đình, bạn bè và những người đã luôn động viên, hỗ trợ và cho em niềm tin để hoàn thành đồ án một cách tốt nhất.

Dù đã cố gắng hết sức, em nhận thấy rằng đồ án vẫn còn một số hạn chế và điều chưa hoàn hảo. Quãng thời gian này không đơn giản chỉ là việc nghiên cứu và thực hiện mà còn là những bài học quý báu về sự kiên trì, sự sáng tạo và khả năng vượt qua khó khăn. Sự hỗ trợ, định hướng và ý kiến đóng góp từ quý thầy cô, các bạn, và mọi người sẽ giúp em tiếp tục phát triển và hoàn thiện hơn nữa trong tương lai.

Với tất cả sự kính trọng, em xin chân thành cảm ơn!

Hà Nội, tháng 12 năm 2024

Nhóm sinh viên thực hiện

Nguyễn Trác Năng
Phạm Huy Hoàng

Mục lục

DANH SÁCH ẢNH

DANH SÁCH BẢNG

PHẦN MỞ ĐẦU

1. Phát biểu bài toán

Trong cuộc sống hiện đại, quản lý tài chính cá nhân là một kỹ năng thiết yếu, ảnh hưởng trực tiếp đến chất lượng cuộc sống của mỗi người. Tuy nhiên, trên thực tế, đây lại là một vấn đề gây nhiều khó khăn, đặc biệt đối với các bạn sinh viên, người mới đi làm, hoặc ngay cả những người đã có gia đình.

a. Khó khăn trong quản lý tài chính cá nhân:

- Thiếu kiến thức và thói quen quản lý tài chính: Nhiều người, đặc biệt là giới trẻ, chưa được trang bị đầy đủ kiến thức về quản lý tài chính. Họ thường không có khái niệm rõ ràng về lập ngân sách, theo dõi chi tiêu, hay đầu tư, dẫn đến việc chi tiêu thiếu kiểm soát, không có khả năng tiết kiệm, và dễ rơi vào tình trạng nợ nần, đặc biệt là khi sử dụng các hình thức vay tiêu dùng như thẻ tín dụng.
- Quản lý thủ công không hiệu quả: Đối với những người đã có gia đình, việc quản lý tài chính thường được thực hiện thủ công bằng sổ sách. Phương pháp này tiềm ẩn nhiều hạn chế:
    - Sai sót do ghi chép không đầy đủ: Việc quên ghi chép các khoản chi tiêu nhỏ, mất hóa đơn, hoặc ghi chép không chính xác là rất phổ biến, dẫn đến sai lệch trong thống kê và khó khăn trong việc kiểm soát dòng tiền.
    - Khó khăn trong việc phân tích và đánh giá: Sổ sách thủ công khó có thể cung cấp cái nhìn tổng quan và chi tiết về tình hình tài chính. Việc phân tích xu hướng chi tiêu, đánh giá hiệu quả tiết kiệm, hay dự đoán tình hình tài chính trong tương lai trở nên khó khăn.
    - Tốn thời gian và công sức: Việc ghi chép, tính toán và tổng hợp dữ liệu thủ công tốn rất nhiều thời gian và công sức.
- Khó khăn trong quản lý các khoản vay và tiết kiệm: Việc theo dõi các khoản vay, đặc biệt là các khoản vay dài hạn như vay mua nhà, vay trả góp, trở nên phức tạp nếu chỉ dựa vào sổ sách. Tương tự, việc quản lý các khoản tiết kiệm, theo dõi lãi suất, và đánh giá hiệu quả đầu tư cũng gặp nhiều khó khăn.

b. Tác động của việc thiếu công cụ quản lý tài chính:

Việc thiếu một công cụ hỗ trợ quản lý tài chính hiệu quả không chỉ ảnh hưởng đến cá nhân mà còn tác động đến cả gia đình. Ví dụ, trong các trường hợp vay mượn lớn như xây nhà, việc ghi chép thủ công dễ dẫn đến mất mát thông tin, quên cập nhật, gây khó khăn trong việc kiểm soát nợ nần và lập kế hoạch trả nợ. Điều này có thể gây ra áp lực tài chính lớn và ảnh hưởng đến chất lượng cuộc sống.

c. Đề xuất giải pháp và mục tiêu của đồ án:

Xuất phát từ những nhu cầu thực tế này, đồ án tập trung vào việc phát triển một ứng dụng quản lý tài chính cá nhân trên thiết bị di động. Ứng dụng này hướng đến

việc cung cấp một giải pháp toàn diện, dễ sử dụng và phù hợp với đa số người dùng, đặc biệt tập trung vào khả năng thống kê mạnh mẽ và linh hoạt.

d. Các chức năng thống kê dự kiến:

- Thống kê chi tiêu theo thời gian: Cho phép người dùng xem thống kê chi tiêu theo ngày, tuần, tháng, năm, hoặc một khoảng thời gian tùy chọn. Hiển thị thông tin dưới dạng biểu đồ dễ dàng so sánh và phân tích xu hướng chi tiêu.
- Thống kê chi tiêu theo danh mục: Phân loại chi tiêu theo các danh mục (ví dụ: ăn uống, đi lại, mua sắm, giải trí) và hiển thị tỷ lệ chi tiêu cho từng danh mục. Giúp người dùng nhận biết được các khoản chi tiêu chính và điều chỉnh ngân sách cho phù hợp.
- Thống kê thu nhập: Theo dõi và thống kê các nguồn thu nhập khác nhau, giúp người dùng có cái nhìn tổng quan về tình hình tài chính.
- Thống kê vay và nợ: Quản lý các khoản vay và nợ, bao gồm thông tin về số tiền vay, và lịch sử trả nợ. Cung cấp các báo cáo thống kê về tình hình nợ nần, giúp người dùng kiểm soát và lập kế hoạch trả nợ hiệu quả.
- Thống kê tiết kiệm: Theo dõi các khoản tiết kiệm, bao gồm số tiền tiết kiệm,và thời gian tiết kiệm. Giúp người dùng đánh giá hiệu quả tiết kiệm và lập kế hoạch tài chính cho tương lai.
- Tùy chỉnh báo cáo: Cho phép người dùng tùy chỉnh các báo cáo thống kê theo nhu cầu, ví dụ: lựa chọn khoảng thời gian.

e. Mục tiêu của đồ án:

Mục tiêu của đồ án không chỉ là phát triển một ứng dụng xử lý và thống kê tài chính cá nhân mà còn tập trung vào việc tạo ra một sản phẩm thân thiện, dễ sử dụng và phù hợp với đa số người dùng. Ứng dụng hướng đến việc cung cấp một giải pháp toàn diện nhằm cải thiện trải nghiệm quản lý tài chính và mang lại giá trị thiết thực trong đời sống hàng ngày, đặc biệt nhấn mạnh vào khả năng thống kê chi tiết, trực quan và dễ hiểu, giúp người dùng dễ dàng nắm bắt và kiểm soát tình hình tài chính của mình.

2. Lý do chọn đề tài

- **Nhu cầu thực tiễn cao:** Trong thời đại hiện nay, việc quản lý tài chính cá nhân đã trở thành một kỹ năng cần thiết đối với mỗi cá nhân. Tuy nhiên, nhiều người gặp khó khăn trong việc theo dõi thu nhập, chi tiêu, tiết kiệm và nợ. Một ứng dụng quản lý tài chính hiệu quả sẽ giúp người dùng giải quyết vấn đề này một cách khoa học và tiện lợi. Việc nắm dõ tình hình tài chính giúp mọi người sử dụng nguồn lực của mình một cách thông minh, tránh lãng phí, đồng thời đạt được các mục tiêu như tiết kiệm, đầu tư, hoặc giải quyết các khoản nợ.
- **Tiềm năng phát triển và ứng dụng rộng rãi:** Ứng dụng quản lý tài chính không chỉ phù hợp với nhu cầu cá nhân mà còn có thể mở rộng để hỗ trợ, gia đình, hoặc các tổ chức phi lợi nhuận. Đề tài này có tính ứng dụng cao, đáp ứng nhu cầu thực tế của nhiều đối tượng. Đề tài này không chỉ tập trung vào quản lý tài chính mà

còn tạo ra cơ hội để tích hợp AI vào phân tích dữ liệu để đưa ra những gợi ý hay dự đoán tài chính cá nhân

- **Tối ưu hóa thời gian và công sức:** Việc ghi chép và tính toán tài chính bằng các phương pháp truyền thống (ghi sổ tay) tốn nhiều thời gian, dễ sai sót và không thân thiện với mọi người. Ứng dụng sẽ giúp việc ghi chép dễ dàng hơn, tiếp kiệm thời gian và công sức. Các vấn đề về khoản vay, các khoản tiếp kiệm sẽ dễ dàng có thể xử lí hơn. Việc thống kê sẽ không còn phải tính toán một cách thủ công mà sẽ chỉ cần bấm một vài thao tác sẽ có ngay một bản thống kê chi tiêu hàng tháng hàng năm hay hàng ngày. Việc này sẽ giúp trách sai sót khi tính toán bằng cách truyền thống.

3. Mục tiêu đề tài

- **Phát triển ứng dụng hoàn chỉnh và có thể ứng dụng thực tế:** Xây dựng một ứng để quản lý tài chính cá nhân, tích hợp đầy đủ các tính năng cần thiết và có thể sử dụng thực tế trên các thiết bị di động. Ứng dụng cần đáp ứng các nhu cầu về tính năng, hiệu suất và trải nghiệm người dùng.
- **Thiết kế giao diện người dùng thân thiện và hiệu quả:** Xây dựng giao diện người dùng trực quan, dễ sử dụng, cho phép người dùng dễ dàng sử dụng để có thể thống kê các giao dịch, quản lí các khoản tiết kiệm, các khoản vay. Giao diện cần tối ưu để có thể hiện thị trên các màn hình khách nhau.
- **Hiểu rõ hơn về cách phát triển một hệ thống quản lí tài chính cá nhân:** Bằng cách nghiên cứu sâu về các phương pháp phát triển và thiết kê phần mềm. Và các kiến thức về tài chính liên quan để phát triển một ứng dụng quản lí tài chính các nhân có tính đúng đắn cao.

4. Phạm vi đồ án

- **Phạm vi nghiên cứu về lý thuyết:** Tìm hiểu các khái niệm cơ bản như thu nhập, chi tiêu, tiết kiệm, nợ . Bằng cách này chúng ta có thể xây dựng nên tảng lý thuyết vững chắc để hiểu dõ hơn cách mọi người chi tiêu.
- **Phạm vi về ứng dụng thực tế:**
  + **Nền tảng phát triển:** Ứng dụng sẽ được phát triển trên nền tảng Android, sử dụng ngôn ngữ lập trình Kotlin.
  + **Chức năng chính:**
    - Tạo tài khoản
    - Tạo ví: Người dùng sẽ tạo ví gồm (thẻ tín dụng và ví)
    - Quản lí debt transaction: Chức năng này sẽ quản lí mỗi giao dịch của các khoản vay
    - Quản lí Debt: Chức năng này sẽ quản lí các quản vay của người dùng gồm payment và receivable.
    - Quản lí Goal Transaction: Chức năng này sẽ quản lí các khoản tiếp kiệm có thể rút ra hoặc gửi thêm vào.
    - Tạo password: tăng cường bảo mật cho ứng dụng

- Quản lí Goal: Chức năng này dùng để quản lý mục tiêu đặt ra
- Quản lí Transfer: Quản lý các giao dịch
- Quản lí Budget: Quản lý các kinh phí đặt ra
- Xem wallet statistic: Xem thống kê của ví
- Chọn ngôn ngữ: Chọn ngôn ngữ phù hợp theo ý muốn của người dùng

5. Cấu trúc đồ án

Nội dung của đồ án được xây dựng thành các chương như sau:

Chương 1. Cơ sở lý thuyết về quản lí tài chính cá nhân

Chương 2. Phân tích thiết kế hệ thống của ứng dụng quản lí tài chính cá nhân

Chương 3. Phát triển ứng dụng quản lí tài chính cá nhân

Chương 4. Kết luận

CHƯƠNG 1: CỞ SỞ LÝ THUYẾT

Chương này trình bày về những kiến thức cơ bản và nên tảng liên quan đến những khái niệm về thu nhập ,chi tiêu ,hay khoản vay và những công nghệ, thư viện liên quan trong đồ án.

1.1. Giới thiệu về ứng dụng quản lý tài chính

1.1.1. Định nghĩa và tính chất cơ bản của ứng dụng quản lý tài chính

- **Ứng dụng quản lý tài chính** là một loại ứng dụng di động giúp người dùng theo dõi, quản lý thu nhập và chi tiêu một cách hiệu quả. Ứng dụng cung cấp các công cụ để lập kế hoạch tài chính, phân tích dòng tiền, và đưa ra các gợi ý nhằm tối ưu hóa việc sử dụng tài chính cá nhân.

1.1.2. Công nghệ sử dụng

- Ứng dụng sẽ được triển khai trên nền tảng **Android**, sử dụng kiến trúc **MVVM (Model-View-ViewModel)** để đảm bảo mã nguồn được tổ chức rõ ràng, dễ bảo trì và mở rộng.
- **Room Database** sẽ được sử dụng để quản lý dữ liệu cục bộ
- Trong phạm vi đồ án, ứng dụng sẽ tích hợp các chức năng chính sau:
  + **Quản lý giao dịch**: Cung cấp tính năng ghi chép các khoản thu nhập và chi tiêu, phân loại giao dịch theo danh mục (ví dụ: ăn uống, giải trí, đầu tư).
  + **Thống kê và báo cáo**: Hiển thị biểu đồ và báo cáo chi tiết về tình hình tài chính theo thời gian, giúp người dùng có cái nhìn tổng quan về các khoản chi tiêu lớn.
  + **Lập kế hoạch tài chính**: Hỗ trợ người dùng đặt mục tiêu tiết kiệm và theo dõi tiến độ thực hiện.
  + **Thông báo nhắc nhở**: Gửi thông báo nhắc nhở người dùng về các hóa đơn hoặc mục tiêu tài chính sắp đến hạn.

1.2. Giới thiệu về công nghệ, kiến trúc sử dụng trong đồ án

1.2.1. Giới thiệu về Android

Trong thời đại công nghệ hiện nay, các thiết bị di động đóng vai trò quan trọng trong đời sống hằng ngày của con người. Android, với vai trò là hệ điều hành di động phổ biến nhất thế giới, đã trở thành nền tảng chính cho việc phát triển ứng dụng di động, phục vụ hàng tỷ người dùng trên toàn cầu. Để phát triển ứng dụng thay đổi giọng nói trong đồ án này, Android được chọn làm nền tảng chính bởi những đặc điểm vượt trội và tính ứng dụng rộng rãi của nó.

a. Đặc điểm của Android

- Tính mở và miễn phí: Android là một nền tảng mã nguồn mở, cho phép các nhà phát triển tùy chỉnh và mở rộng hệ điều hành theo nhu cầu.

- Khả năng tùy biến cao: Các nhà phát triển có thể xây dựng ứng dụng và giao diện phù hợp với từng đối tượng người dùng.
- Được hỗ trợ bởi cộng đồng lớn: Hệ sinh thái Android rất phong phú, với cộng đồng nhà phát triển lớn và tài nguyên đa dạng như tài liệu, thư viện mã nguồn mở.
- Phổ biến rộng rãi: Android chiếm hơn 70% thị phần hệ điều hành di động, đồng nghĩa với việc ứng dụng phát triển trên Android có cơ hội tiếp cận lượng lớn người dùng.

b. Tại sao sử dụng Android trong đồ án?

Android được lựa chọn làm nền tảng phát triển ứng dụng trong đồ án này vì những lý do sau:

- Tiếp cận rộng rãi: Android là nền tảng phổ biến, ứng dụng phát triển trên Android có khả năng tiếp cận với số lượng lớn người dùng phổ thông.
- Hỗ trợ đa dạng thiết bị: Từ điện thoại thông minh, máy tính bảng đến các thiết bị IoT, Android hỗ trợ nhiều loại thiết bị, mở rộng phạm vi ứng dụng của đồ án.
- Hỗ trợ ngôn ngữ lập trình hiện đại: Với Kotlin và Java, Android cho phép phát triển ứng dụng nhanh chóng, hiệu quả, và dễ bảo trì.

1.2.2. Giới thiệu về Kotlin

Kotlin là một ngôn ngữ lập trình hiện đại, mã nguồn mở, được phát triển bởi JetBrains và chính thức được Google công nhận là ngôn ngữ chính cho lập trình Android. Với cú pháp ngắn gọn, dễ đọc, hỗ trợ kiểm tra null an toàn và tích hợp mạnh mẽ với các công cụ như Android Studio, Kotlin giúp tăng năng suất và giảm thiểu lỗi lập trình. Bên cạnh đó, Kotlin hỗ trợ lập trình bất đồng bộ với coroutines, phù hợp với các ứng dụng yêu cầu hiệu suất cao. Với xu hướng hiện nay, Google và cộng đồng đang ưu tiên phát triển các thư viện và tài liệu hỗ trợ Kotlin, biến nó trở thành lựa chọn hàng đầu cho phát triển ứng dụng Android hiện đại, đặc biệt trong các đồ án cần hiệu quả và chất lượng cao.

a. Mục tiêu thiết kế của kotlin

Kotlin được thiết kế với các mục tiêu chính sau :

- Ngắn gọn và dễ đọc: Kotlin tập trung vào việc giảm bớt những đoạn mã dư thừa so với Java. Điều này giúp các lập trình viên viết mã nhanh hơn, dễ bảo trì hơn mà vẫn giữ được tính rõ ràng.
- An toàn: Một trong những vấn đề lớn nhất của Java là lỗi NullPointerException (NPE). Kotlin giải quyết vấn đề này bằng cách tích hợp cơ chế **null safety**, cho phép phân biệt rõ ràng giữa biến có thể null (nullable) và không thể null (non-nullable).
- Tương thích với java: Kotlin được xây dựng để có thể sử dụng song song với Java trong cùng một dự án. Điều này giúp các công ty hoặc nhà phát triển có thể dần chuyển đổi từ Java sang Kotlin mà không cần thay đổi toàn bộ mã nguồn.

- Hiệu năng cao: Kotlin chạy trên JVM, có khả năng tương thích ngược và hiệu năng tương tự Java. Ngoài ra, Kotlin còn hỗ trợ biên dịch xuống mã máy (native) hoặc mã JavaScript, giúp mở rộng phạm vi ứng dụng của nó.
- Đa nền tảng: Kotlin hỗ trợ phát triển đa nền tảng thông qua **Kotlin Multiplatform**, cho phép chia sẻ mã nguồn logic giữa các nền tảng như Android, iOS, web, và desktop.

b. Ưu điểm của kotlin

Kotlin có các ưu điểm sau :

- Cú pháp hiện địa và ngắn gọn: Kotlin giúp giảm bớt các đoạn mã dư thừa bằng cách loại bỏ những phần không cần thiết như findViewById() trong Android hoặc các khai báo lặp lại. Điều này giúp mã nguồn dễ đọc và dễ bảo trì hơn.
- Null Safety (An toàn với Null): Kotlin loại bỏ phần lớn các lỗi liên quan đến giá trị null nhờ cơ chế null safety. Các biến phải được khai báo rõ ràng là có thể null hay không, và các thao tác trên biến nullable yêu cầu kiểm tra trước.
- Hỗ trợ lập trình hàm: Kotlin cung cấp các tính năng lập trình hàm mạnh mẽ như lambda, higher-order functions, và collection operations (map, filter, reduce), giúp xử lý dữ liệu hiệu quả hơn.
- Tính tương thích cao: Kotlin hoàn toàn tương thích với Java, cho phép các lập trình viên sử dụng mã Java cũ trong các dự án Kotlin và ngược lại.
- Đa nền tảng: Kotlin Multiplatform cho phép chia sẻ mã giữa các nền tảng, giúp tiết kiệm thời gian và công sức trong việc phát triển ứng dụng di động và web.
- Hộ trợ mạnh mẽ từ Google và JetBrains: Với sự công nhận từ Google và sự hỗ trợ từ JetBrains, Kotlin ngày càng trở thành lựa chọn đáng tin cậy cho các nhà phát triển Android.

c. Nhược điểm của kotlin

Kotlin có các nhược điểm sau :

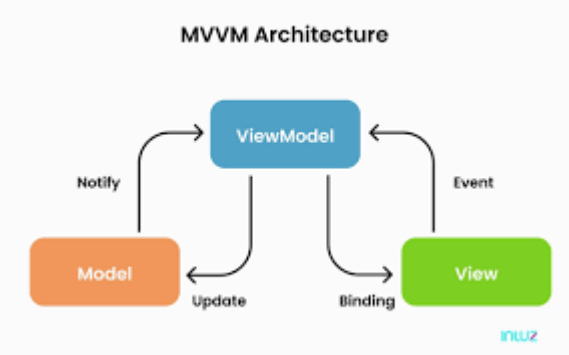
- Thời gian biên dịch : Mặc dù đã được tối ưu hóa , nhưng thời gian biên dịch của kotlin đôi khi chậm hơn java, đặc biệt với các dự án lớn.
- Tài liệu và cộng đồng: Dù cộng động kotlin đang phát triển nhanh chóng, nhưng nó vẫn nhỏ hơn so với java. Một số vấn đề có thể khó tìm được giải pháp so với java
- Đường cong học tập: Đối với người mới học của kotlin có thể hơi khó hiểu so với java, đặc biệt làm khi làm việc với các tính năng nâng cao như lập trình hàm hoặc Kotlin Multiplatform.

d. Tại sao lại sử dụng kotlin trong đồ án

- Kotlin là ngôn ngữ chính thức cho phát triển Android, điều này đảm bảo rằng mọi công cụ, thư viện, và tài liệu liên quan đến Android đều hỗ trợ Kotlin đầy đủ.
- Hỗ trợ lập trình bất đồng bộ với Coroutines: Kotlin hỗ trợ **Coroutines**, một tính năng mạnh mẽ giúp xử lý tác vụ bất đồng bộ

1.2.3. Giới thiệu về kiến trúc MVVM

Trong đồ án sẽ sử dụng kiến trúc MVVM (Model-View-ViewModel) để tổ chức code theo cách rõ ràng, linh hoạt và dễ bảo trì. Hình ảnh minh họa trên thể hiện cách các thành phần trong kiến trúc MVVM tương tác với nhau, giúp phân tách rõ ràng giữa giao diện (View), logic ứng dụng (ViewModel), và dữ liệu/logic nghiệp vụ (Model). Điều này không chỉ cải thiện sự độc lập giữa các thành phần mà còn giúp việc kiểm thử và mở rộng ứng dụng trở nên dễ dàng hơn.



Hình 1.1: Sơ đồ kiến trúc MVVM

a. Cấu trúc

- Kiến trúc MVVM (Model-View-ViewModel) là một mẫu thiết kế phần mềm phổ biến được sử dụng trong phát triển ứng dụng, đặc biệt là trong các nền tảng như Android, iOS và WPF (Windows Presentation Foundation). MVVM tách biệt các phần của ứng dụng thành ba thành phần chính:
  + Model**:** Chứa dữ liệu, logic kinh doanh và các thao tác xử lý dữ liệu.
  + View**:** Đại diện cho giao diện người dùng (UI) và chịu trách nhiệm hiển thị dữ liệu.
  + ViewModel**:** Là cầu nối giữa Model và View, xử lý các logic liên quan đến giao diện người dùng và phản hồi từ người dùng.

b. Đặc điểm

- Tách biệt trách nhiệm:
  + Model chỉ tập trung vào dữ liệu và xử lý nghiệp vụ.
  + View chỉ hiển thị dữ liệu và không chứa logic kinh doanh.
  + ViewModel trung gian xử lý, tương tác giữa Model và View.
- Data Binding:
  + ViewModel cung cấp dữ liệu cho View thông qua các cơ chế như LiveData hoặc Observable.
  + View tự động cập nhật khi dữ liệu trong ViewModel thay đổi.
- Không phụ thuộc giữa View và Model:
  + View không trực tiếp truy cập vào Model, mà thông qua ViewModel.
  + Điều này giúp giảm sự phụ thuộc và cải thiện khả năng tái sử dụng.
- Hỗ trợ kiểm thử:

+ Tách biệt logic ra khỏi View giúp kiểm thử dễ dàng hơn, đặc biệt là kiểm thử ViewModel mà không cần phụ thuộc vào UI.

c. Tại sao sử dụng MVVM trong đồ án?

- Tăng khả năng bảo trì: Mã nguồn được tổ chức rõ ràng và tách biệt trách nhiệm, giúp dễ dàng bảo trì và mở rộng.
- Tái sử dụng: Các ViewModel có thể được tái sử dụng trong nhiều View khác nhau.
- Dễ dàng kiểm thử: Logic được tách biệt trong ViewModel có thể kiểm thử độc lập mà không cần tương tác với giao diện người dùng.
- Hỗ trợ cập nhật dữ liệu thời gian thực: Với Data Binding hoặc LiveData, View có thể tự động cập nhật mà không cần viết thêm mã để xử lý.
- Giảm lỗi khi xử lý dữ liệu: Sự tách biệt rõ ràng giữa UI và logic xử lý giúp giảm nguy cơ lỗi do tương tác trực tiếp giữa giao diện và dữ liệu.

1.2.4. Giới thiệu về thư viện MPAndroidChart

MPAndroidChart là một thư viện mã nguồn mở mạnh mẽ, được thiết kế để hiển thị biểu đồ trong các ứng dụng Android. Thư viện hỗ trợ đa dạng các loại biểu đồ như biểu đồ đường, cột, tròn và biểu đồ kết hợp, với khả năng tùy chỉnh cao và hiệu suất mượt mà. MPAndroidChart tích hợp dễ dàng với Kotlin hoặc Java, đồng thời cung cấp nhiều tính năng như zoom, kéo thả, và chú thích, giúp nâng cao trải nghiệm người dùng. Trong xu hướng hiện nay, MPAndroidChart được ưu tiên lựa chọn nhờ cộng đồng hỗ trợ lớn, tài liệu đầy đủ và khả năng hiển thị dữ liệu trực quan, là giải pháp tối ưu cho đồ án cần trực quan hóa dữ liệu một cách hiệu quả.

Hình 1.2: Biểu đồ hình tròn vẽ bằng MPAndroidChart

a. Một số điểm nổi bật của MPAndroidChart :

- Đa dạng các loại biểu đồ:
  + Biểu đồ đường
  + Biểu đồ thanh
  + Biểu đồ tròn
  + Biểu đồ radar
  + Biểu đồ nến

+ Biểu đồ kết hợp

- Tính năng tương tác :
  + Hỗ trợ zoom, kéo, và xoay biểu đồ.
  + Hiển thị thông tin chi tiết khi người dùng chạm vào các điểm dữ liệu.
- Tùy chỉnh dễ dàng:
  + Cung cấp nhiều tùy chọn để thay đổi màu sắc, kiểu đường, kích thước văn bản, định dạng nhãn, v.v.
  + Dễ dàng tích hợp và tinh chỉnh theo yêu cầu của dự án.

b. Tại sao lại phải dụng trọng đồ án:

Công cụ này giúp vẽ biểu đồ hình tròn trong đồ án, và các hiệu ứng xung quang.

1.2.5. Giới thiệu về thư viện exp4j

Exp4j là một thư viện Java gọn nhẹ dùng để tính toán các biểu thức toán học dưới dạng chuỗi ký tự một cách nhanh chóng và hiệu quả. Với khả năng hỗ trợ các toán tử cơ bản, hàm toán học phổ biến, và tùy chỉnh biến hoặc hàm riêng, exp4j phù hợp cho các ứng dụng yêu cầu xử lý các phép tính động. Thư viện này có hiệu suất cao, dễ tích hợp vào các dự án và không phụ thuộc vào các thư viện khác, giúp giảm tải cho hệ thống. Trong xu hướng hiện nay, exp4j được sử dụng rộng rãi nhờ sự đơn giản, ổn định, và khả năng đáp ứng tốt các bài toán phức tạp, là lựa chọn lý tưởng cho đồ án cần xử lý biểu thức toán học linh hoạt và hiệu quả.

a. Tính năng chính của exp4j

- Hộ trợ đa dạng các phép tính toán :
  + Các phép toán cơ bản: cộng(+), trừ(-), nhân(*), chia(/) và mũ(^)
  + Các hàm toán học nâng cao: sin, cos, tan, log, sqrt, abs, exp, và nhiều hàm khác.
  + Hỗ trợ các hằng số như π (pi) và e (hằng số Euler).
- Gọn nhẹ và dễ sử dụng: thư viện chỉ khoản vài trăm KB và không có các phụ thuộc thức tạp khác.

b. Tại sao phải sử dụng trong đồ án:

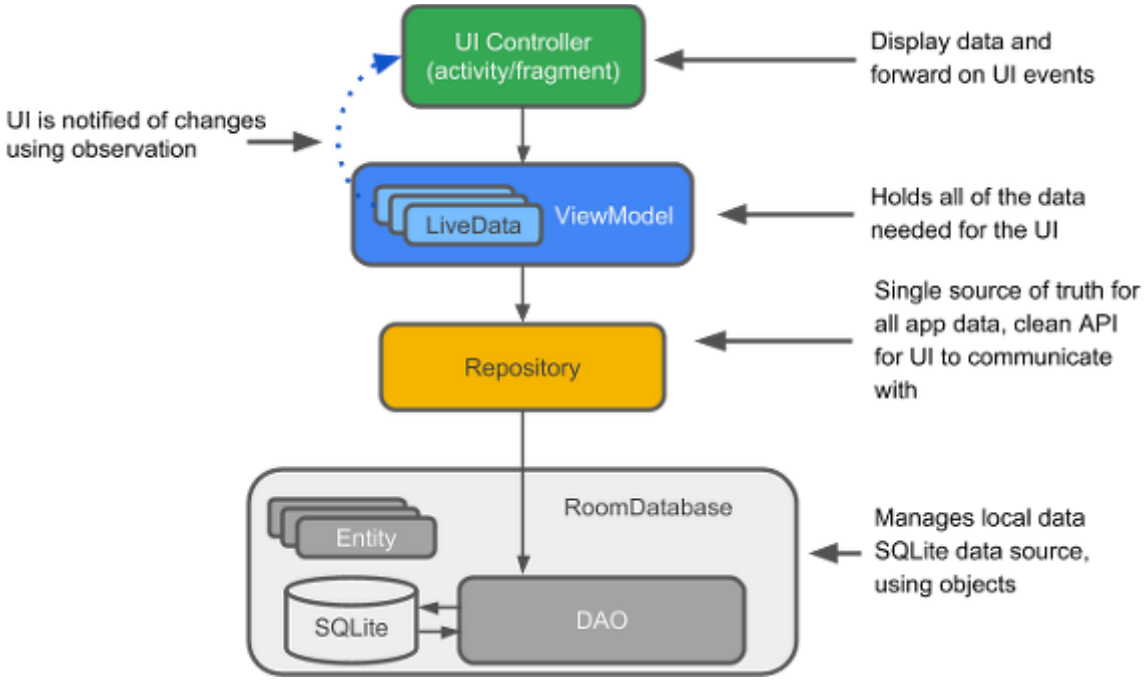
Trong đồ án sẽ có những phần yên cầu người dùng vào số tiền, người dùng có thể nhập các biểu thức.

1.2.6. Giới thiệu về Roomdatabase

Room Database là một thư viện trong Jetpack, cung cấp lớp trừu tượng để quản lý cơ sở dữ liệu SQLite một cách dễ dàng và hiệu quả. Với khả năng ánh xạ bảng dữ liệu thành các lớp Kotlin/Java thông qua Entity, kiểm tra cú pháp SQL khi biên dịch và hỗ trợ LiveData hoặc Flow để cập nhật dữ liệu theo thời gian thực, Room giúp giảm thiểu lỗi và tăng tốc phát triển. Thư viện này cũng tích hợp tốt với các công nghệ hiện đại như Hilt và WorkManager. Trong xu hướng hiện nay, Room được ưu tiên sử dụng trong các dự án Android nhờ tính dễ mở rộng, hiệu suất cao, và khả

năng đáp ứng các yêu cầu phức tạp, là lựa chọn lý tưởng cho đồ án cần cơ sở dữ liệu ổn định và dễ quản lý.

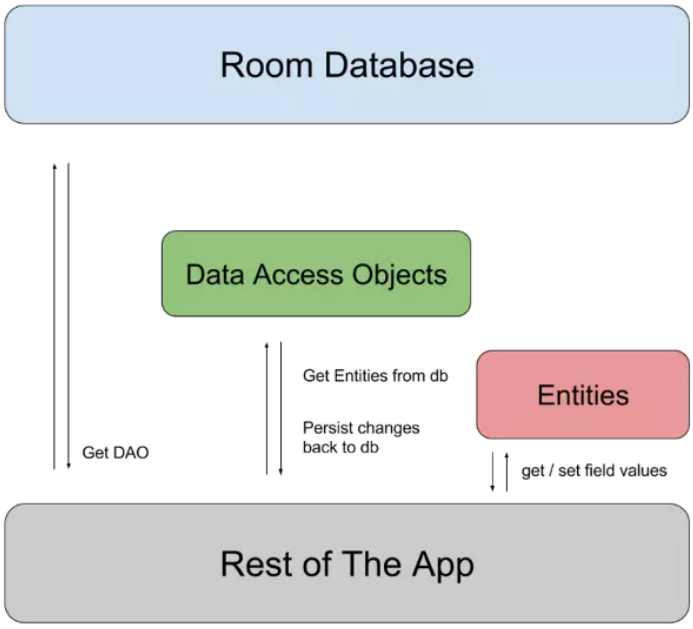
a. Đặc điểm của Room Database



Hình 1.3: Room được sử dụng trong kiến trúc tổng thể.

Room database gồm những thành phần chính như sau:

- Database class: là nơi chứa database và là điểm truy cập chính cho kết nối dữ liệu liên quan đến ứng dụng.
- Data entities: là đại diện cho các bảng trong databse của ứng dụng
- Databse access object (DAOs): cung cấp các phương thức mà ứng dụng có thể sử dụng để truy vấn , cập nhật , chèn và xóa dữ liệu trong databse.



Hình 1.4: Các thành phần tính của room

Ưu điểm của Room Database:

- ORM (Object Relational Mapping): Room giúp ánh xạ các bảng trong SQLite với các lớp Entity trong Kotlin hoặc Java, dễ dàng quản lý dữ liệu dưới dạng đối tượng.
- Kiểm tra biên dịch: Room thực hiện kiểm tra cú pháp SQL khi biên dịch, giúp giảm thiểu lỗi.
- Hỗ trợ Flow và LiveData: Dễ dàng sử dụng với Flow hoặc LiveData để cập nhật dữ liệu theo thời gian thực trong giao diện người dùng.
- Dễ dàng mở rộng: Hỗ trợ các truy vấn phức tạp và tích hợp với các công nghệ khác trong Jetpack như WorkManager, Hilt, ViewModel.
- Tích hợp RxJava: Room hỗ trợ sử dụng với RxJava để xử lý dữ liệu không đồng bộ.

b. Tại sao sử dụng Room database cho đồ án?

- Room Database là một lựa chọn tuyệt vời để xây dựng các ứng dụng Android hiện đại, giúp tăng tốc quá trình phát triển và đảm bảo chất lượng đồ án.
- Quản lý dữ liệu hiệu quả: Room giúp tổ chức dữ liệu dưới dạng bảng, dễ dàng truy vấn, thêm, sửa, xóa mà không cần thao tác trực tiếp với SQLite.
- Đảm bảo độ ổn định: Việc kiểm tra biên dịch giúp tránh lỗi logic SQL, nâng cao chất lượng ứng dụng.
- Tích hợp linh hoạt: Hỗ trợ LiveData hoặc Flow, giúp giao diện người dùng tự động cập nhật khi dữ liệu thay đổi, phù hợp với kiến trúc MVVM.
- Hiệu suất cao: Các truy vấn được tối ưu hóa và thực thi không đồng bộ giúp ứng dụng mượt mà hơn, không bị treo khi xử lý dữ liệu lớn.
- Dễ duy trì và mở rộng: Code dễ đọc, dễ bảo trì, thuận lợi khi nâng cấp hoặc thêm chức năng mới trong đồ án.

1.3. Kết luận

Chương 1 đã cung cấp các cơ sở lý thuyết cần thiết để hiểu rõ hơn các công nghệ và phương pháp sử dụng trong đồ án. Những kiến thức này là nền tảng để phân tích và thiết kế hệ thống trong chương tiếp theo .

CHƯƠNG 2: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

Chương này phân tích yêu cầu của ứng dụng, đề xuất các mô hình hoạt động, và thiết kế hệ thống dựa trên kiến trúc MVVM. Phần thiết kế bao gồm việc mô tả các luồng xử lý chính và các thành phần giao diện trên ứng dụng.

2.1. Pha lấy yêu cầu

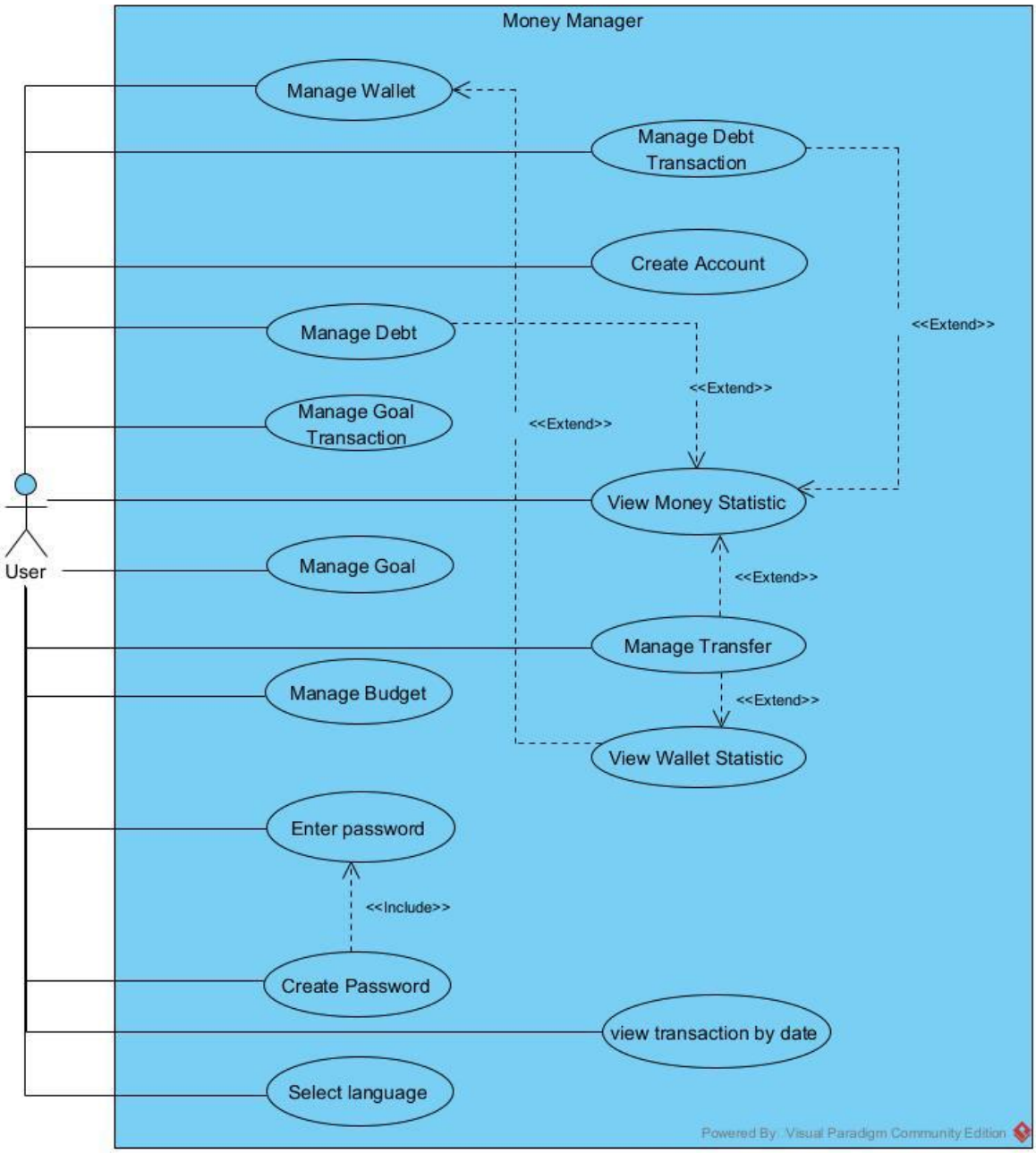
2.1.1. Biểu đồ usecase tổng quát

- Các tác nhân tham gia hệ thống và mô tả họ được thể hiện trong bảng 1 dưới đây :

Bảng 1: Mô tả các tác nhân của hệ thống

STT	Tác nhân	Mô tả	Nghiệp vụ
1	User	Người sử dụng ứng dụng để quản lý tài chính cá nhân	1. Tạo tài khoản 2. Tạo ví 3. Quản lí debt transaction 4. Quản lí Debt 5. Quản lí Manage Goal Transaction 6. Quản lí Password 7. Quản lí Goal 8. Quản lí Transfer 9. Quản lí Budget 10. Xem wallet statistic 11. Tạo password 12. Select language

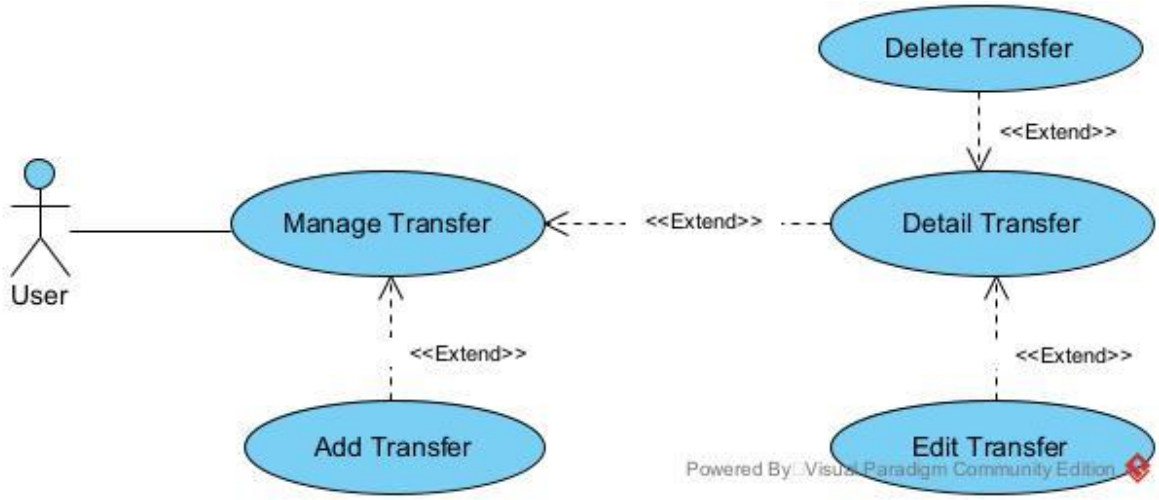
Sau khi đã có được các tác nhân và mô tả, nhiệm vụ ta vẽ được usecase tổng quát như sau :



Hình 2.1: Biểu đồ usecase tổng quát

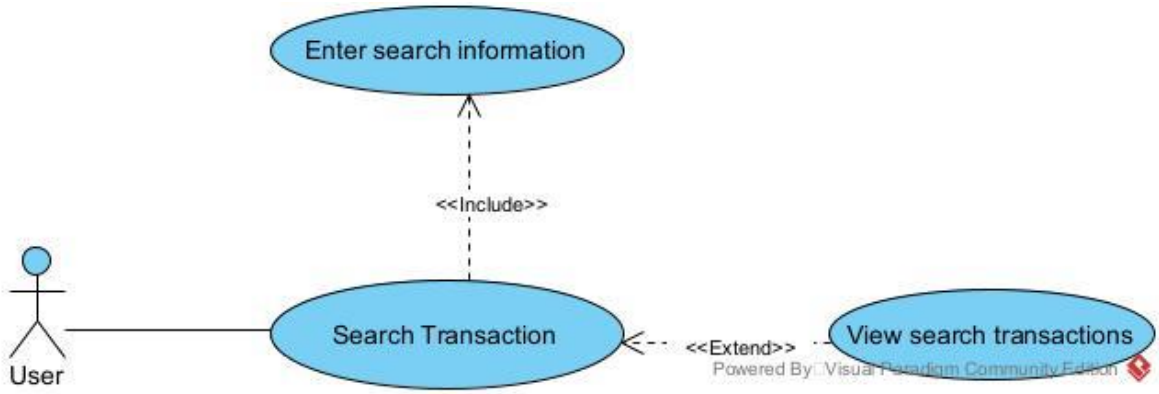
2.1.2. Biểu đồ usecase chi tiết

a. Quản lí giao dịch



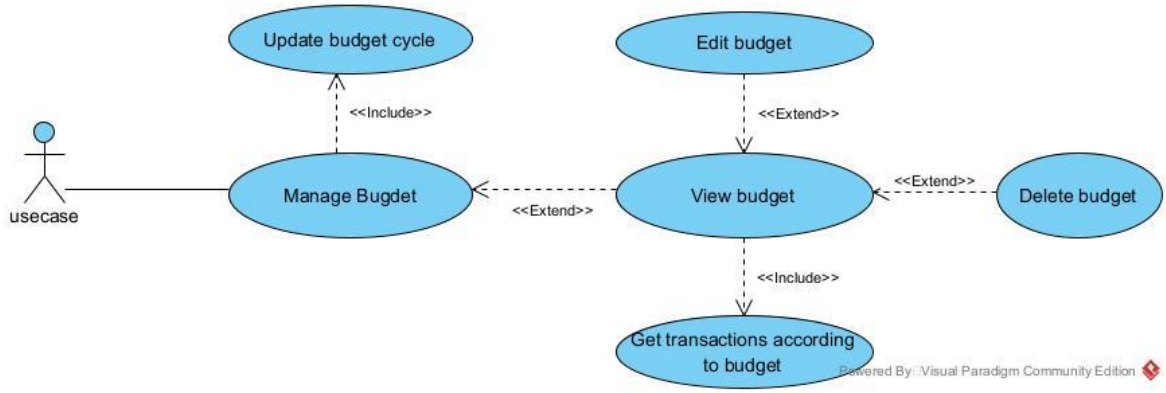
Hình 2.2: Biểu đồ usecase chức năng quản lí giao dịch

b. Tìm kiếm các giao dịch



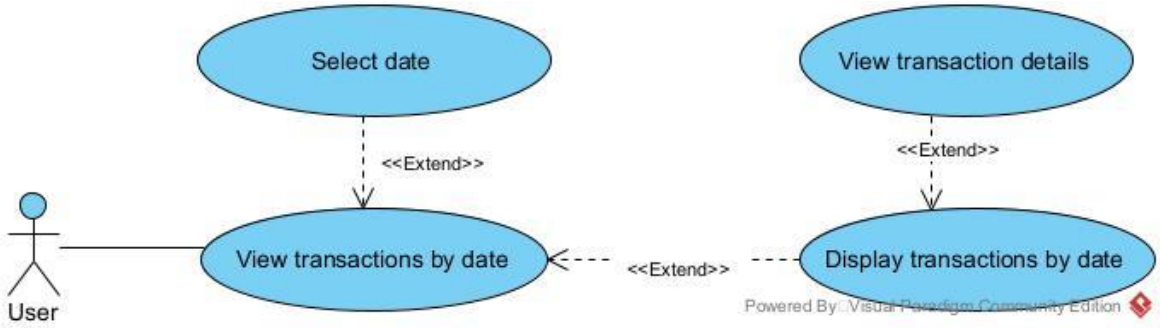
Hình 2.3: Biểu đồ usecase chức năng tìm kiếm

c. Chức năng quản lí ngân sách



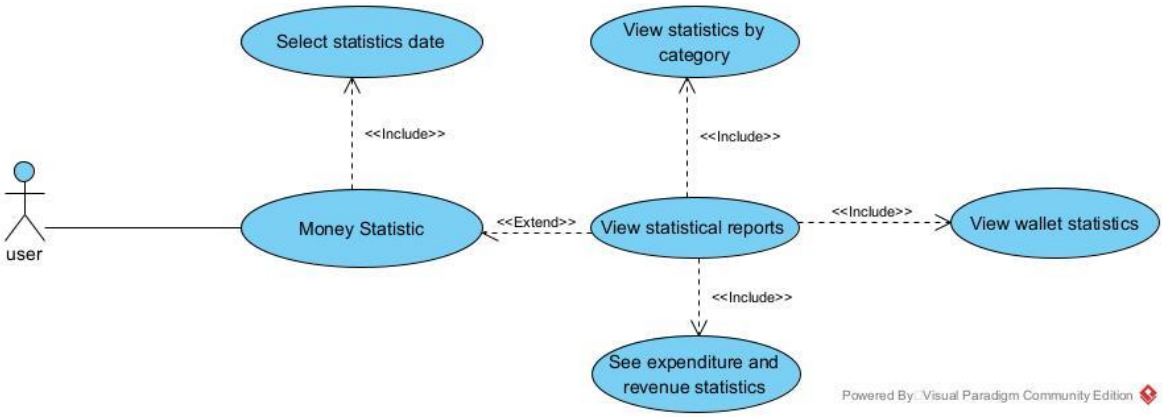
Hình 2.4: Usecase quản lí ngân sách

d. Chức năng xem danh sách chi tiết giao dịch theo ngày



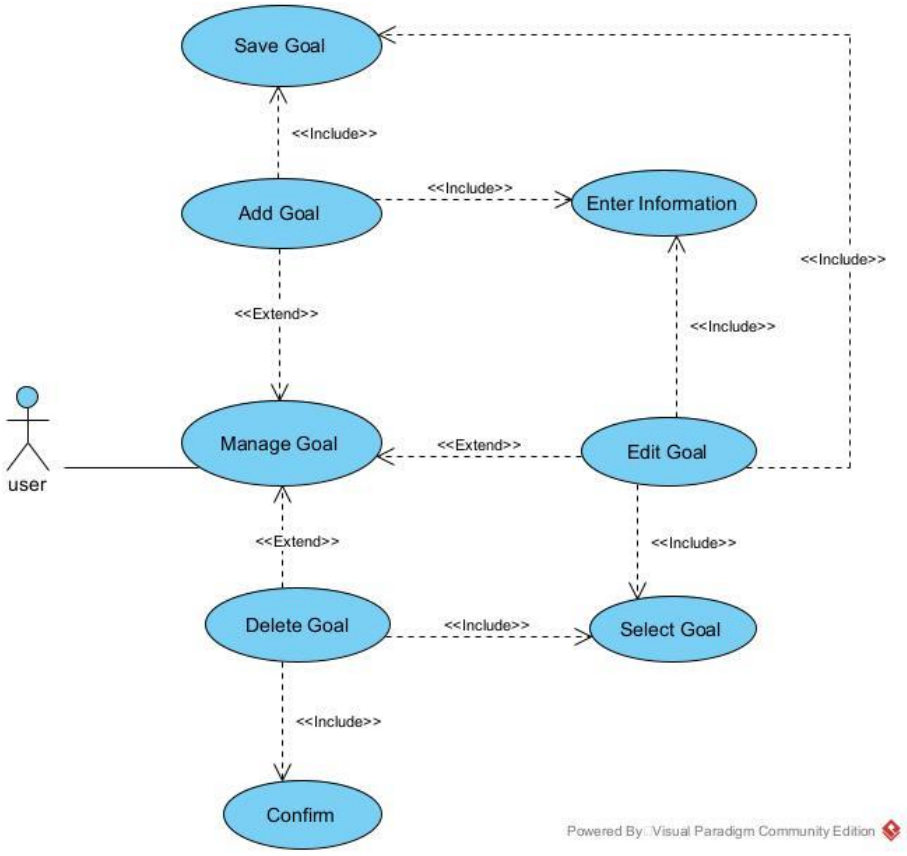
Hình 2.5: Use case xem danh sách chi tiết giao dịch theo ngày

e. Chức năng thông kê giao dịch theo ngày tuần ,tháng , năm



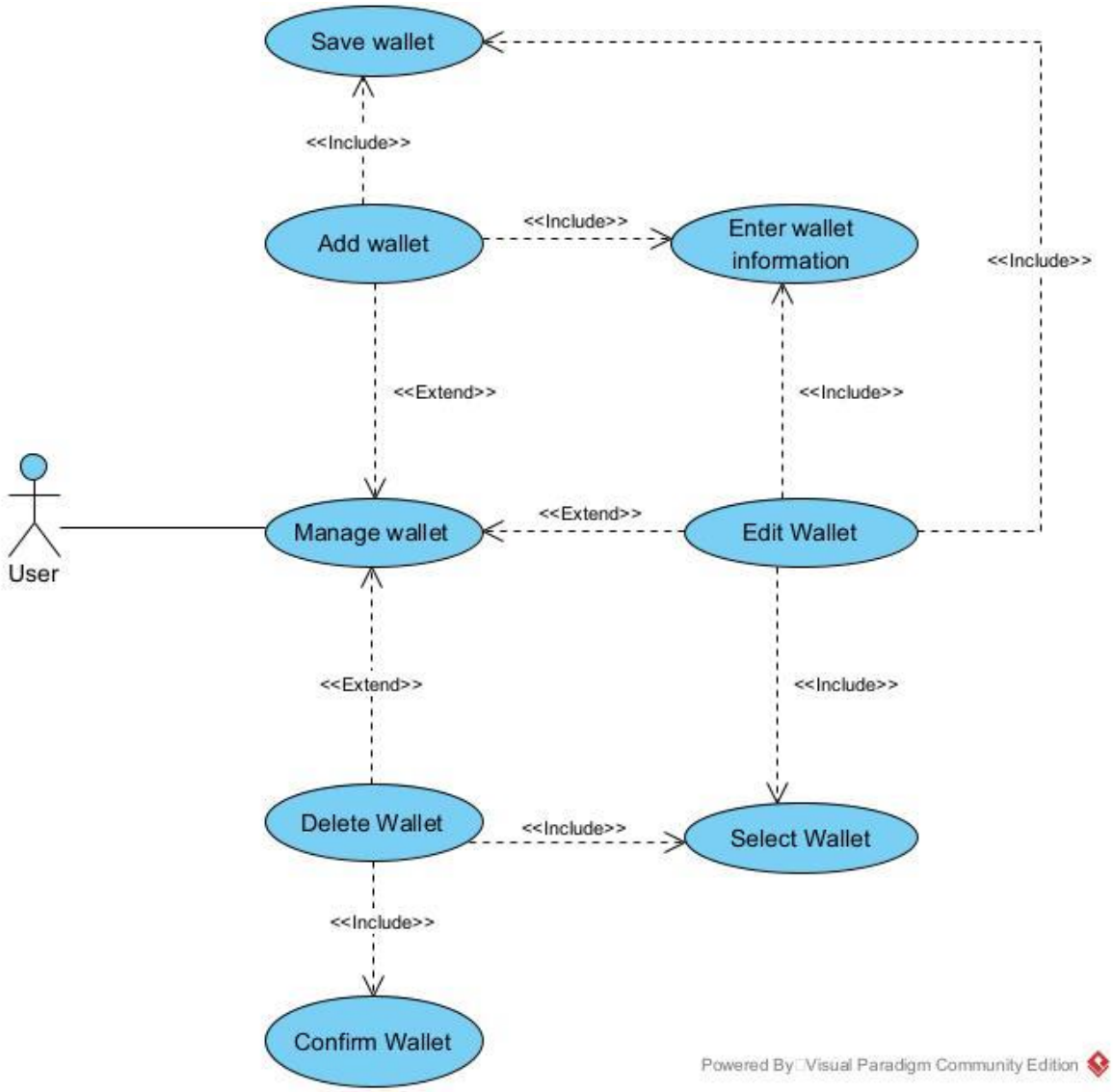
Hình 2.6: Xem danh sách thống kê theo ngày , tuần, tháng, năm

f. Chức năng quản lí goal



Hình 2.7: Usecase quản lí goal

g. Chức năng quản lí ví



Hình 2.8: Usecase quản lí wallet

2.2. Pha phân tích

2.2.1. Kịch bản chuẩn và ngoai lệ

a. Chức năng sửa giao dịch

Bảng 2: Kịch bản chức năng sửa giao dịch

Usecase	Sửa giao dịch
Actor	User
Pre-condition	Người dùng cần nhập đăng nhập vào hệ thống
Post-condition	Người dùng sửa thành công giao dịch
Main event	1. Người dùng đăng nhập vào hệ thống 2. Giao diện trang chủ hiện lên 3. Người dùng chọn vào giao dịch 4. Trang chi tiết giao dịch hiện lên gồm :

<table>
<tr><td></td><td>- Một nút chỉnh sửa
- Một nút xóa
- Thông tin chi tiết giao dịch
- Một nút để quay trở lại
5. Người dùng nhấn vào nút chỉnh sửa
6. Giao dịch chỉnh sửa hiện lên
7. Người dùng nhập các thông tin giao dịch gồm :
- Nhập thông tin ngày thán
- Khi người dùng bấm vào mục nhập số tiền lúc này hệ thống sẽ chuyển sang giao diện nhập số tiền
- Khi người dùng bấm vào chọn danh mục lúc này giao diện chọn danh mục sẽ hiện nên
<table><tr><td>All Category</td></tr><tr><td>Bill</td></tr><tr><td>…</td></tr></table>
- Khi người dùng bấm vào chọn ví thì giao diện chọn ví sẽ hiện lên có 2 loại một.
- Nhập ảnh
- Nhập mô tả về gao dịch.
8. Người dùng nhấn save và trở lại giao diện trang chủ</td></tr>
<tr><td>Exception</td><td>7. Người dùng sửa thông tin
7.1 Người dùng nhập thiếu thông tin số tiền, ví, danh mục
7.2 Hệ thống yêu cầu người dùng nhập lại</td></tr>
</table>

b. Chức năng tìm kiếm giao dịch

Bảng 3: Kịch bản tìm kiếm các giao dịch

<table>
<tr><td>Usecase</td><td>Tìm kiếm giao dịch</td></tr>
<tr><td>Actor</td><td>User</td></tr>
<tr><td>Pre-condition</td><td>Người dùng cần nhập đăng nhập vào hệ thống</td></tr>
<tr><td>Post-condition</td><td>Người dùng tìm kiếm thành công</td></tr>
<tr><td>Main event</td><td>1. Người dùng đăng nhập vào hệ thống
2. Giao diện trang chủ hiện lên
3. Người dùng chọn chức năng tìm kiếm
4. Giao diện tìm kiếm hiện lên
<table><tr><td>Ngày bắt đầu :</td><td>Ngày kết thúc :</td></tr><tr><td colspan="2">Chọn ví :</td></tr><tr><td colspan="2">Chọn danh mục :</td></tr><tr><td colspan="2">Nhập mô tả :</td></tr></table></td></tr>
</table>

	<table><tr><td>Số tiền min :</td><td>Số tiền max :</td></tr></table> 5. Người dùng chọn nút lọc các giao dịch 6. Người dùng nhập các thông tin để tìm kiếm 7. Người dùng bấm vào nút áp dụng 8. Hệ thống hiện lên các giao dịch 9. Người dùng chọn một giao dịch tìm kiếm 10. Thông tin chi tiết giao dịch hiện lên gồm : - Tên giao dịch - Loại giao dịch - Mô tả - Số tiền - Ảnh ….
Exception	6. Người dùng không nhập thông tin và ấn tìm kiếm 6.1 Tất cả giao dịch sẽ hiện lên 6.2 Nếu số tiền max bé hơn số tiền min sex báo lỗi

c. Chức năng sửa ngân sách

Bảng 4: Kịch bản sửa ngân sách

Usecase	Sửa ngân sách
Actor	User
Pre-condition	Người dùng cần nhập đăng nhập vào hệ thống
Post-condition	Người dùng tìm kiếm thành công
Main event	1. Người dùng đăng nhập vào hệ thống 2. Người dùng chọn ngân sách 3. Thông tin chi tiết của ngân sách hiện lên gồm : - Danh sách các giao dịch theo danh mục - Số tiền đã chi tiêu - Ngân sách - Ngày sắp đến hạn của ngân sách … 4. Người dùng chọn vào sửa ngân sách 5. Hệ thống hiện thông tin của ngân sách gồm : - Tên ngày sách - Loại danh mục đã chọn - Chu kì - Màu sắc - Một nút save và một nút quay trở lại 6. Người dùng sửa các thông tin của ngân sách 7. Người dùng bấm lưu ngân sách

Exception	6. Người dùng không nhập thông tin và sửa ngân sách 6.1 Người dùng nhập sai định dạng số tiền 6.2 Hệ thống thông báo lỗi nhập sai định dạng số tiền.

d. Chức năng xem các giao dịch theo ngày

Bảng 5: Kịch bản xem danh sách chi tiết giao dịch theo ngày

Usecase	Sửa ngân sách
Actor	User
Pre-condition	Người dùng cần nhập đăng nhập vào hệ thống
Post-condition	Người dùng tìm kiếm thành công
Main event	1. Người dùng đăng nhập vào hệ thống 2. Người dùng chọn chức năng lịch 3. Hệ thống sẽ hiện lịch bên trong một ô lịch là các khoản chi tiêu: - Một thanh chọn ngày tháng ở trên cùng - 2 nút sang trái sang phải để chuyển tháng - Một bảng lịch bên dưới , bên trong là tổng thu nhập và chi tiêu 4. Người dùng chọn vào một ô trong đó 5. Hệ thống hiện sẽ hiện thị các giao dịch trong ngày hôm đó 6. Người dùng bấm vào một chi tiết giao dịch 7. Hệ thống sẽ hiện thị lên chi tiết một giao dịch gồm - Tên giao dịch - Số tiền giao dịch - Ảnh - …
Exception	3.1 Nếu ngày đó không có giao dịch thì ô đấy sẽ trống

e. Kịch bản thêm ví

Bảng 6: Kịch bản thêm ví

Usecase	Thêm ví
Actor	User
Pre-condition	Người dùng cần nhập đăng nhập vào hệ thống
Post-condition	Người dùng thêm ví thành công
Main event	1. Người dùng đăng nhập vào hệ thống 2. Hệ thống hiện giao diện trang chủ 3. Người dùng chọn chức năng quản lý ví 4. Hệ thống hiện giao diện quản lý ví:

	<table><tr><td>Ví hiện tại</td><td>+ Thêm ví</td></tr></table> 5. Người dùng chọn chức năng thêm ví 6. Hệ thống hiện giao diện thêm ví 7. Người dùng nhập thông tin của ví muốn thêm và ấn lưu 8. Hệ thống lưu ví và quay trở về giao diện quản lý ví
Exception	8. Người dùng nhập thông tin không hợp lệ hoặc bỏ trống 1 số trường bắt buộc 8.1 Hệ thống thông báo không thể thêm ví vì 1 số trường không hợp lệ

f. Kịch bản xem danh sách thống kê theo ngày , tuần, tháng, năm

Bảng 7: Kịch bản xem danh sách thống kê theo ngày , tuần, tháng, năm

Usecase	Thống kê
Actor	User
Pre-condition	Người dùng cần nhập đăng nhập vào hệ thống
Post-condition	Người dùng thống kê thành công
Main event	1. Người dùng đăng nhập thành công vào hệ thống. 2. Người dùng chọn tính năng thông kế 3. Giao diện thống kê hiện lên gồm : - Một button chọn ngày - Một mục hiện số tiền ban đầu của ví, và số tiền ví hiện tại - Một mục hiện chi tiêu và thu nhập - Một mục hiện số tiền thống kê theo danh mục chi tiêu 4. Người dùng chọn ngày thống kê 5. Hệ thống sẽ hiện thị chọn ngày thống kê theo : <table><tr><td>Thống kê theo</td></tr><tr><td>Ngày</td></tr><tr><td>Tuần</td></tr><tr><td>Tháng</td></tr><tr><td>Năm</td></tr><tr><td>Tùy chỉnh</td></tr></table> 6. Người dùng chọn ngày thống kê 7. Số liệu thống kê sẽ hiện lên
Exception	6. Người dùng chọn ngày thống kê 6.1 Nếu ngày hôm đó không có số liệu thì kết quả sẽ bằng 0

2.2.2. Biểu đồ lớp pha phân tích

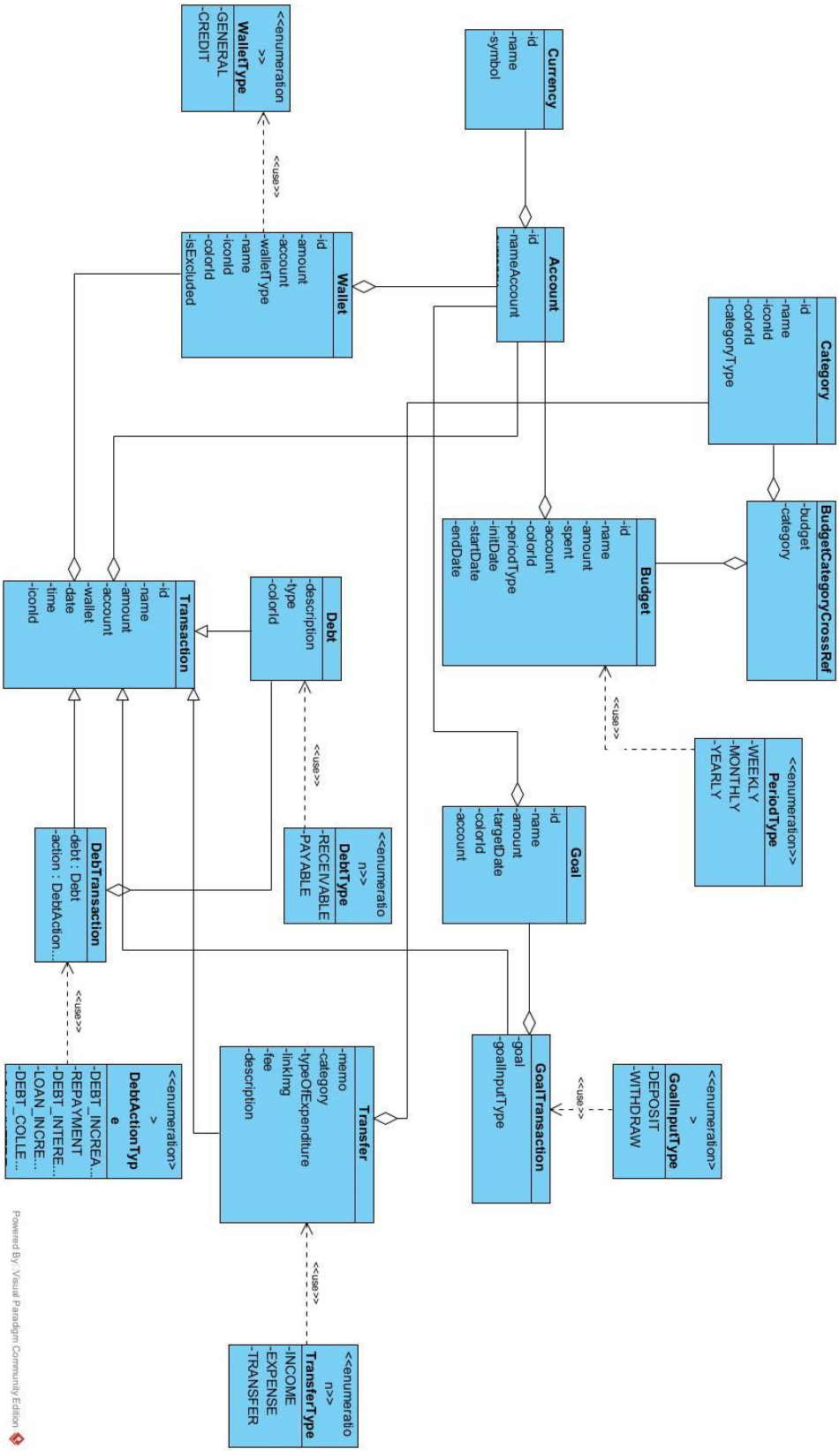
Từ kịch bản chuẩn bên trên ta trích xuất các danh từ để đề xuất các lớp.
Các đối tượng thực thể cần xử lý:

- Người dùng -> Lớp thực thể **Account**
- Ví -> Lớp thực thể **Wallet**
- Ngân sách -> Lớp thực thể **Budget**
- Danh mục -> Lớp thực thể **Category**
- Khoản vay -> Lớp thực thể **Debt**
- Khoản tiết kiệm -> Lớp thực thể **Goal**
- Giao dịch -> Lớp thực thể **Transfer**
- Các giao dịch của khoản vay -> Lớp thực thể **DebtTransaction**
- Các giao dịch của tiếp kiệp -> Lớp thực thể **GoalTransaction**

Quan hệ số lượng giữa các lớp thực thể :

- Một danh mục thì sẽ có nhiều ngân sách và một ngân sách thì nó sẽ có nhiều danh mục: quan hệ của hai lớp này là n-n do vậy ta đề xuất một lớp BudgetCategoryCrossRef

Từ những phân tích trên ta có biểu đồ sau :



Hình 2.9: Biểu đồ lớp của pha phân tích

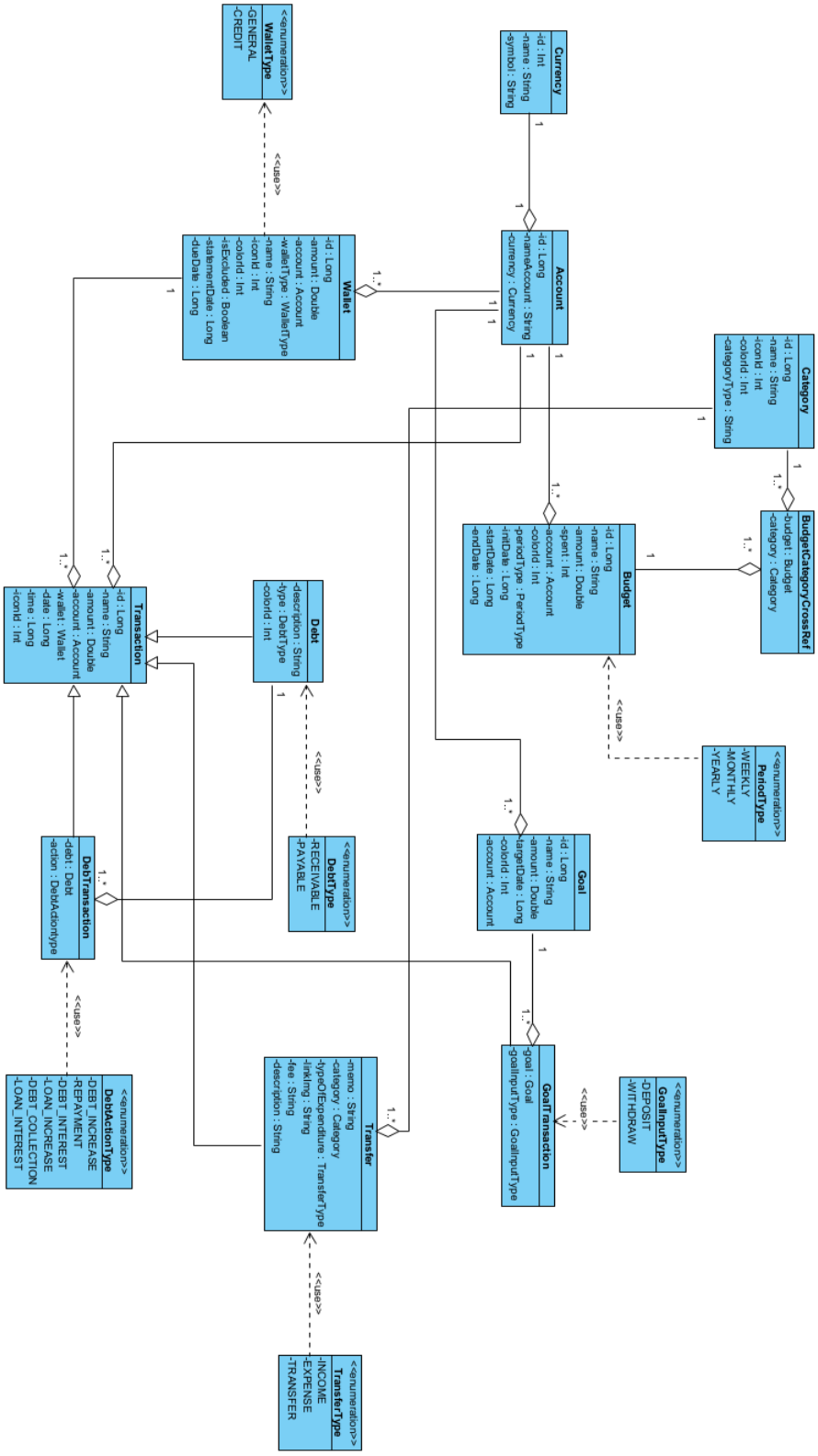
2.3. Pha thiết kê

2.3.1. Biểu đồ lớp của pha thiết kế.

Xác định quan hệ giữa các lớp :

- Account với Currency có quan hệ là 1-1, với mỗi tài khoản chỉ có 1 loại tiền.
- Account với Wallte có quan hệ là 1-n, với mỗi tài khoản sẽ có nhiều ví.
- Account với Budget có quan hệ với 1-n, với mỗi tài khoản sẽ có nhiều Budget.
- Account với Goal có quan hệ với 1-n, với mỗi tài khoản sẽ có nhiều Goal
- Account với Transaction có quan hệ 1-n, với mỗi tài khoản thì sẽ có nhiều Transaction.
- Wallet với Transaction có quan hệ 1-n, với mỗi tài khoản thì sẽ có nhiều Transaction.
- Budget với BudgetCategoryCrossRef có quan hệ 1-n.
- Category với BudgetCategoryCrossRef có quan hệ một n.
- Goal với GoalTransaction có quan hệ là 1-n
- Debt, transfer, DebtTrasaction, GoalTransaction kế thừa lớp Transaction

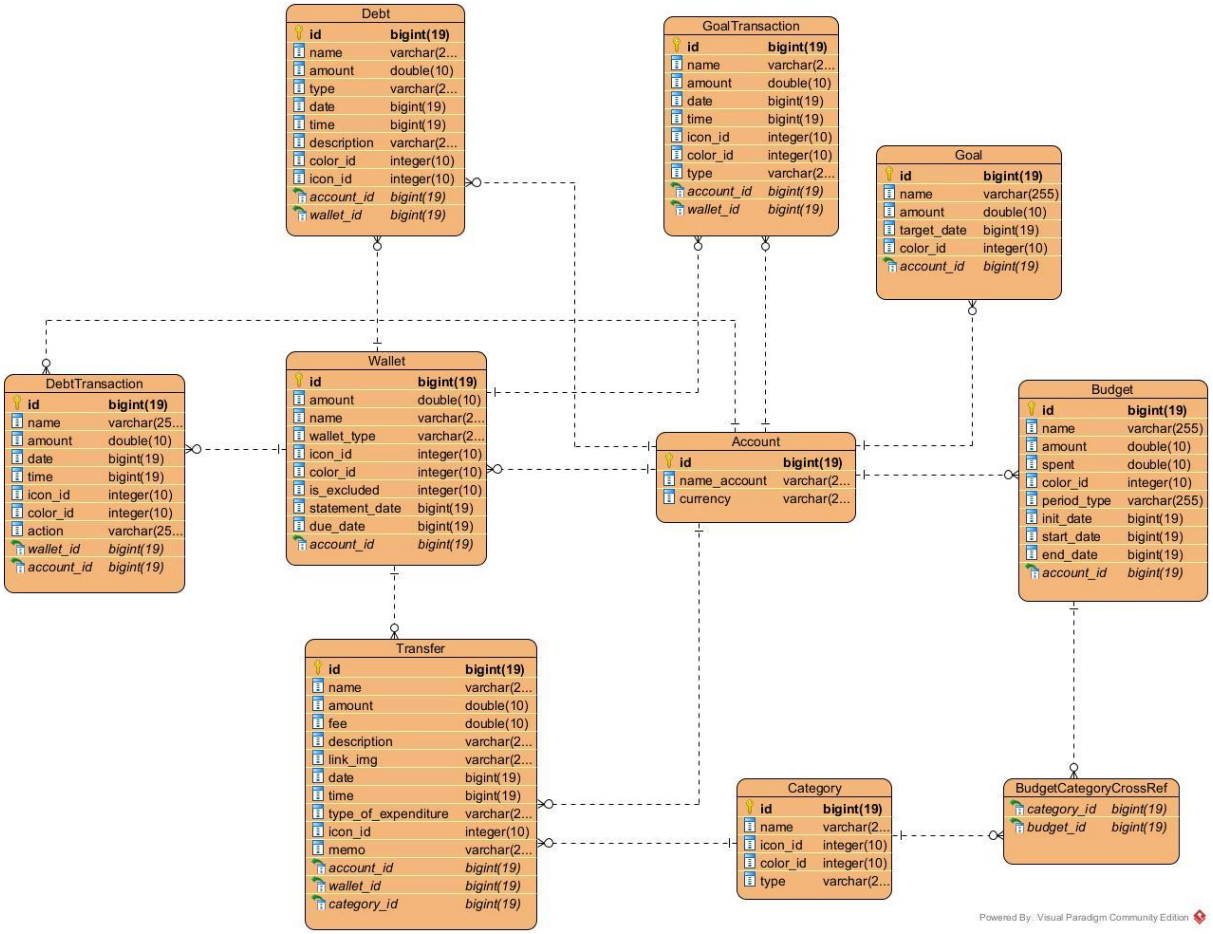
Từ những phân tích trên ta được biểu đồ như sau :



Hình 2.10: Biểu đồ lớp của pha thiết kế

2.3.2. Biểu đồ cơ sở dữ liệu

Từ các phân tích ở biểu đồ pha thiết kế ta suy ra được biểu đồ cơ sở dữ liệu như sau:

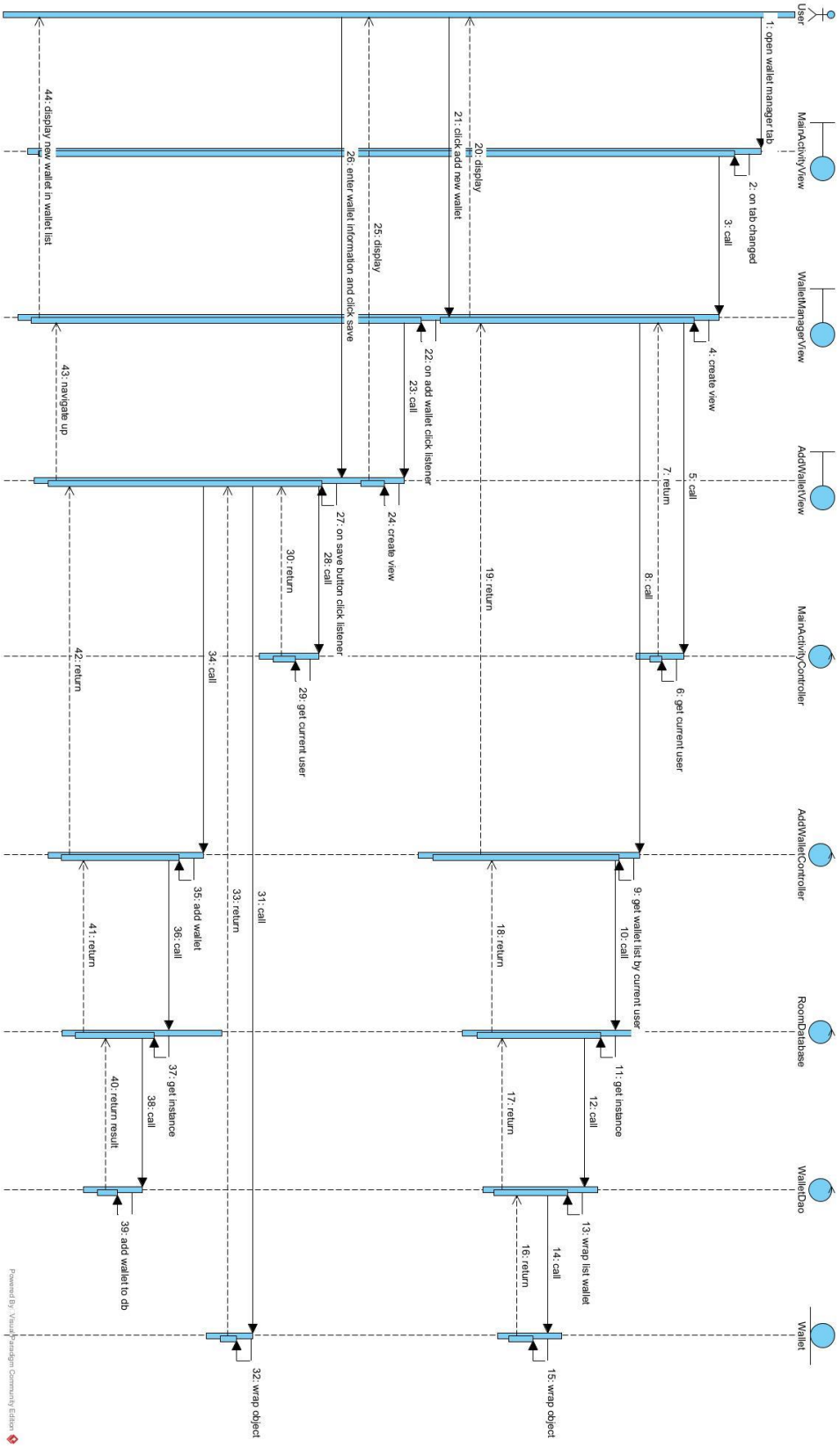


Hình 2.11: Biểu đồ cơ sở dữ liệu

2.3.3. Biểu đồ tuần tự

a. Chức năng thêm ví

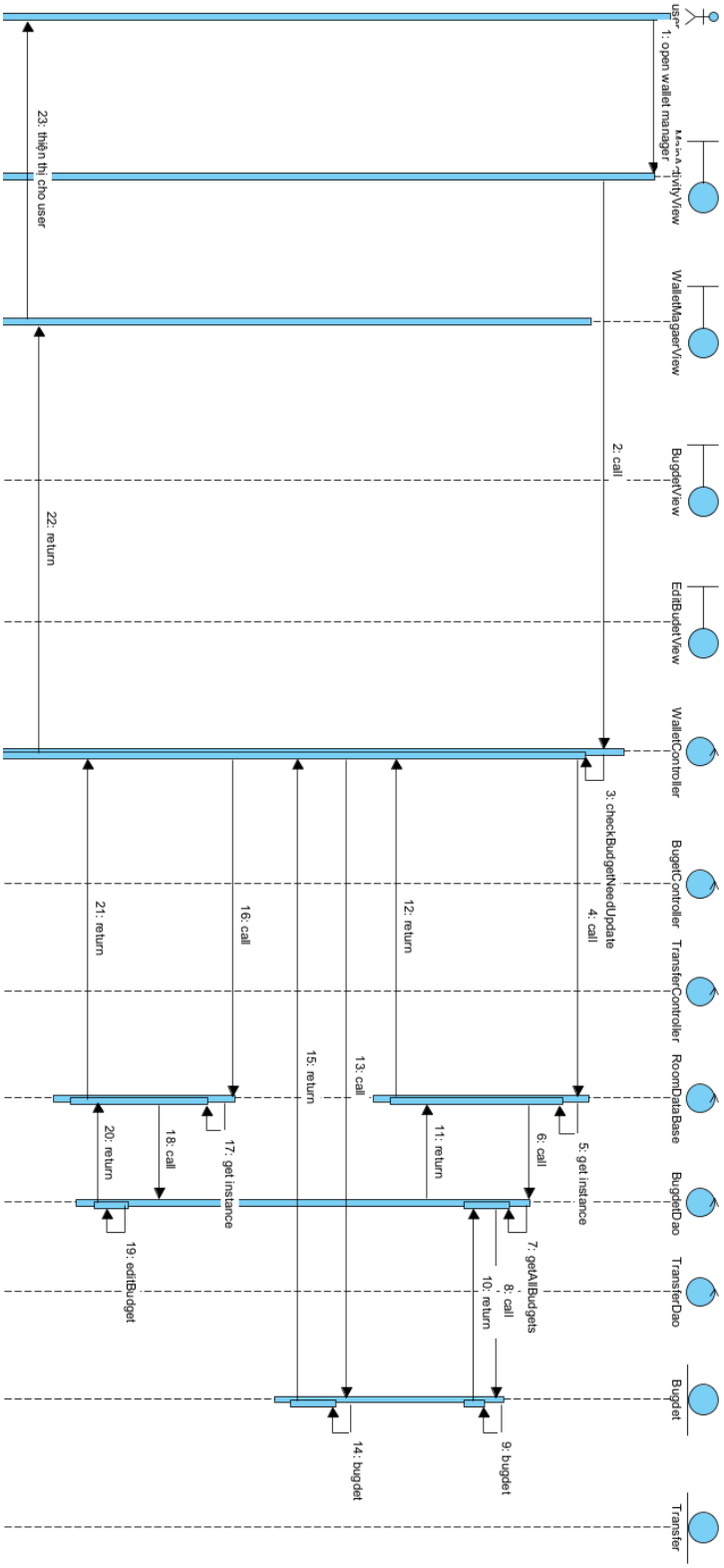
Từ kịch bản chuẩn chức năng thêm ví pha phân tích (2.2.1.e), ta vẽ được biểu đồ tuần tự như hình 2.1.2



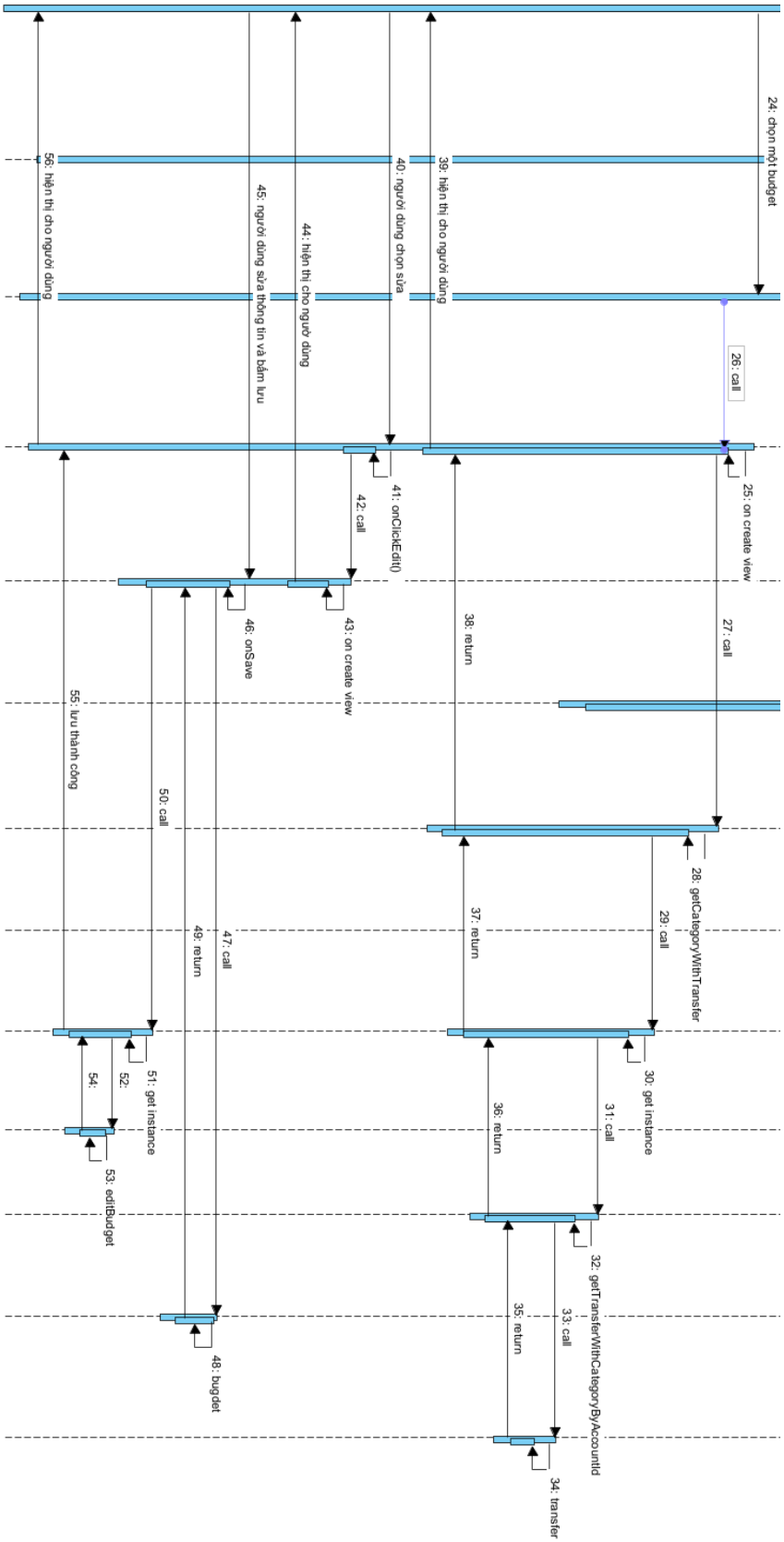
Hình 2.12: Biểu đồ tuần tự của chức năng thêm ví

b. Chức năng sửa Budget

Từ kịch bản chuẩn chức năng thêm ví pha phân tích (2.2.1.c), ta vẽ được biểu đồ tuần tự như hình 2.1.3



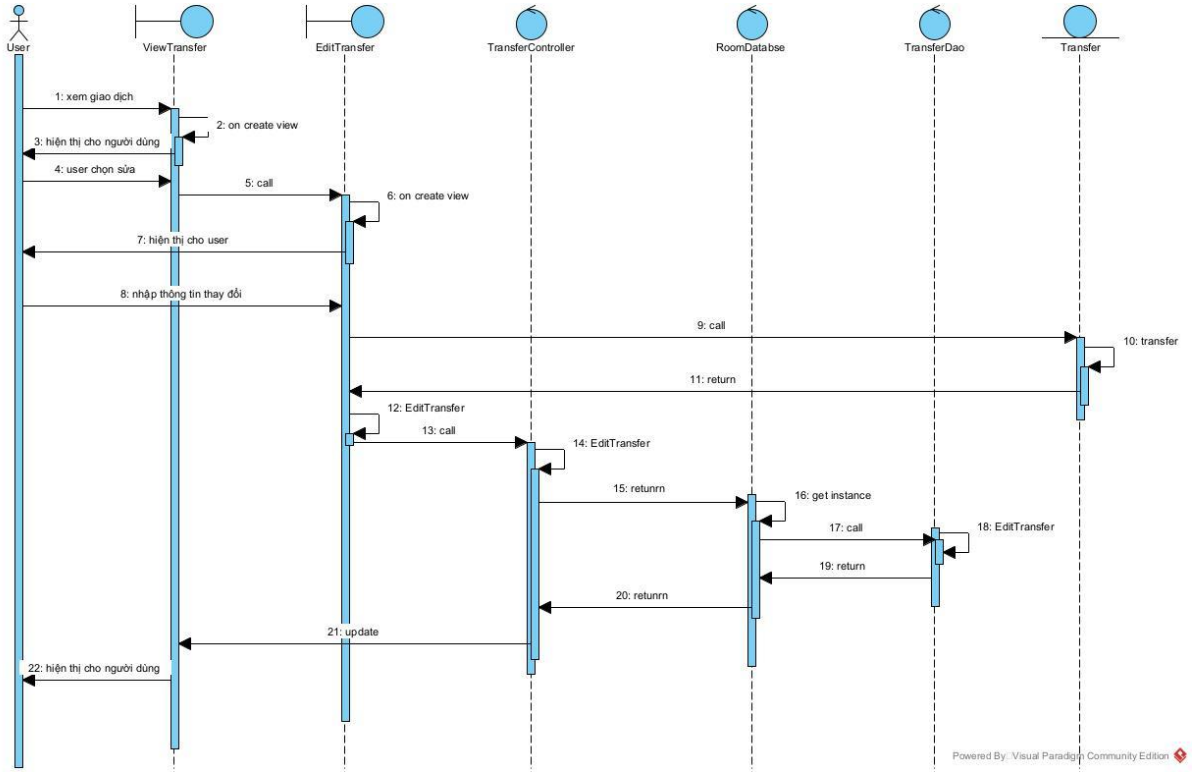
Hình 2.13: Biểu đồ tuần tự của chức năng sửa Budget



Hình 2.14: Biểu đồ tuần tự của chức năng sửa Budget

c. Chức năng sửa transfer

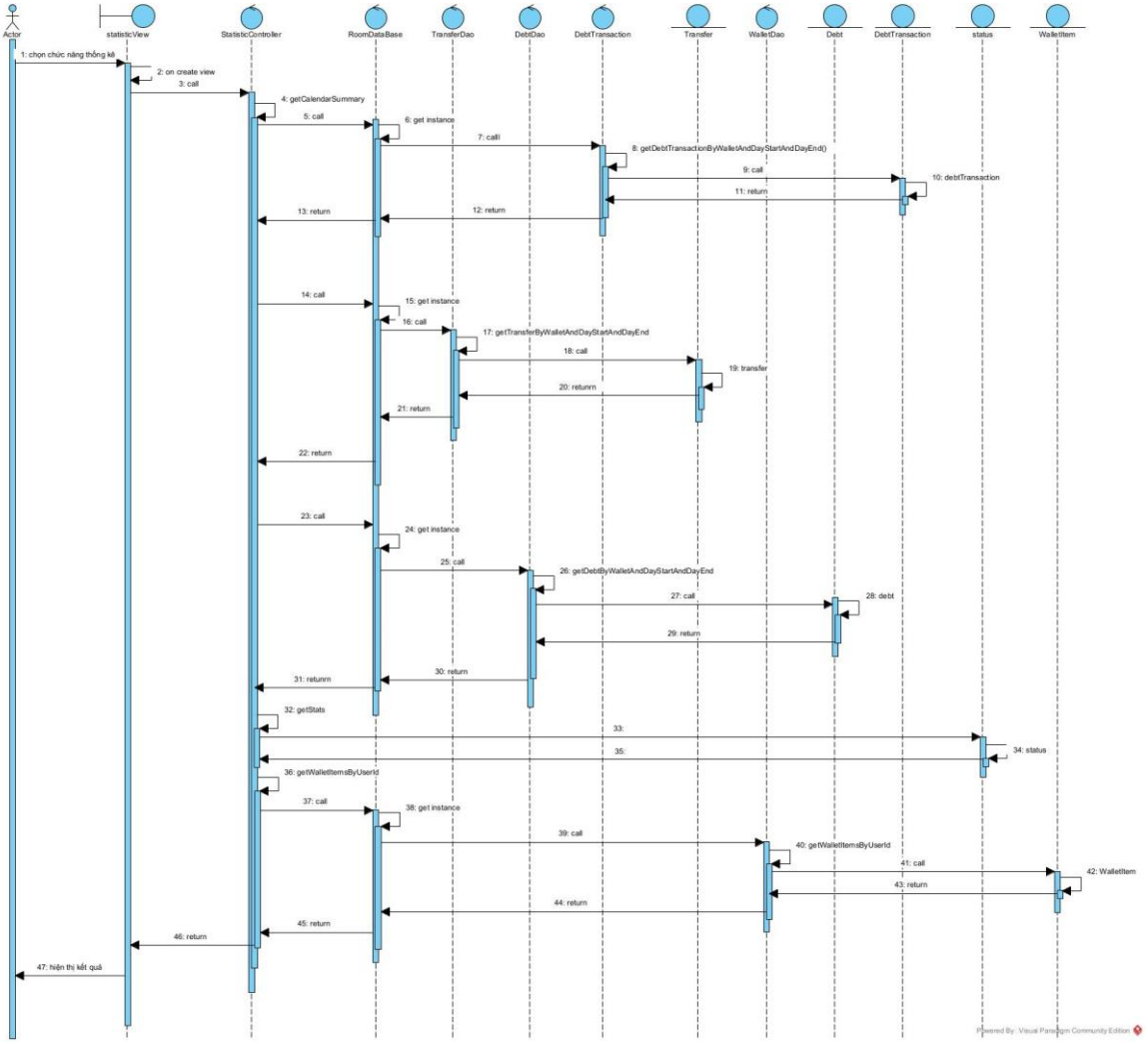
Từ kịch bản chuẩn chức năng thêm ví pha phân tích (2.2.1.a), ta vẽ được biểu đồ tuần tự như hình 2.1.4



Hình 2.15: Biểu đồ tuần tự của chức năng sửa transfer

d. Chức năng thống kê

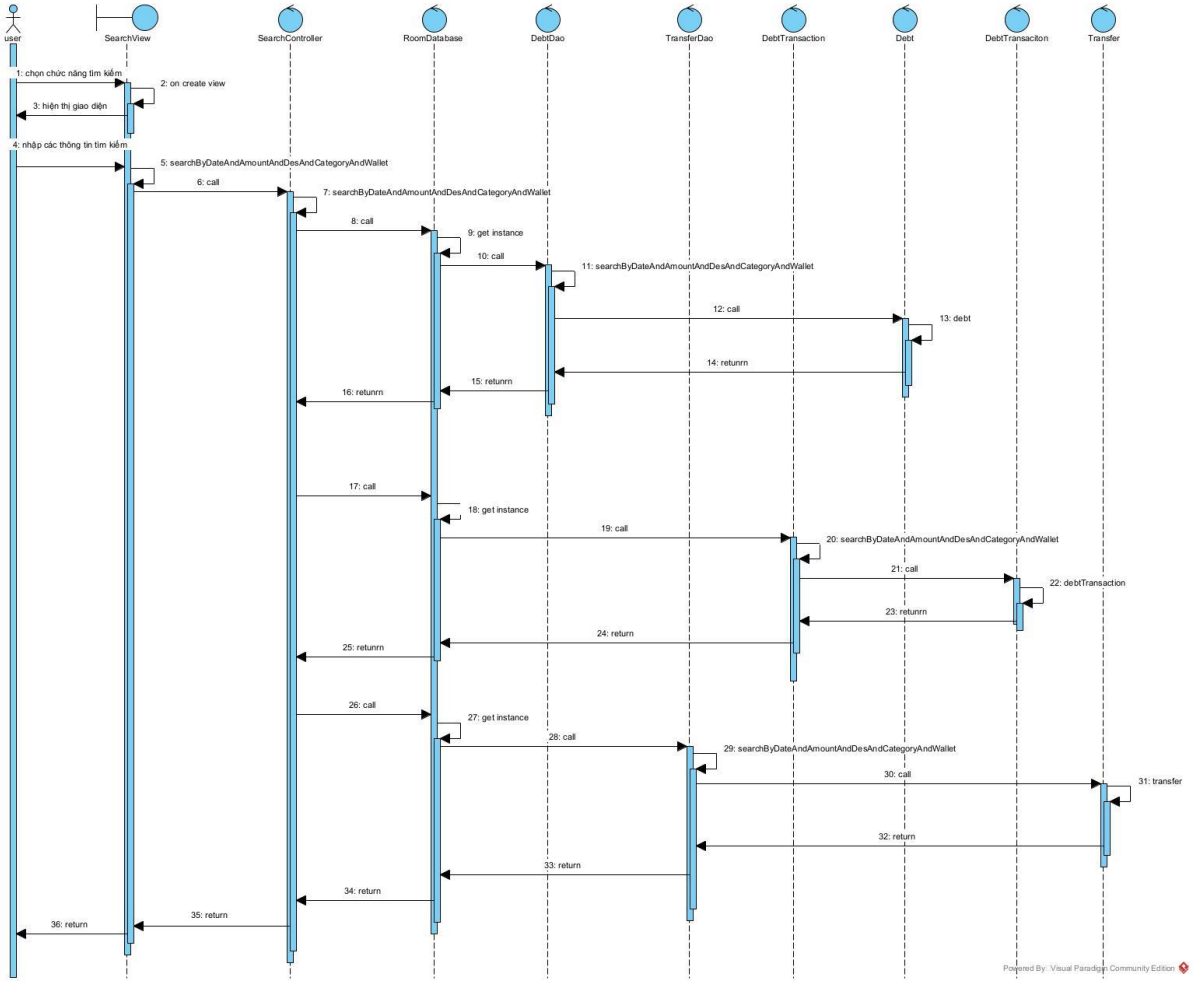
Từ kịch bản chuẩn chức năng thêm ví pha phân tích (2.2.1.e), ta vẽ được biểu đồ tuần tự như hình 2.1.5



Hình 2.16: Biểu đồ tuần tự của chức năng thống kê

e. Chức năng tìm kiếm transaction

Từ kịch bản chuẩn chức năng thêm ví pha phân tích (2.2.1.b), ta vẽ được biểu đồ tuần tự như hình 2.1.7



Hình 2.17: Biểu đồ tuần tự của chức năng tìm kiếm transaction

2.4. Kết luận

Chương 2 đã hoàn thành việc phân tích và thiết kế hệ thống, tạo tiền đề cho việc hiện thực hóa ứng dụng trong chương tiếp theo.

CHƯƠNG 3: PHÁT TRIỂN ỨNG DỤNG

Từ những bước phân tích và thiết kế hệ thống chi tiết như trên, chương cuối này tiến hành triển khai ứng dụng. Giai đoạn này sẽ lựa chọn môi trường triển khai ứng dụng, lập trình và chạy thử chương trình và cuối cùng là đưa ra kết quả đạt được, những ưu nhược điểm, định hướng phát triển trong tương lai của hệ thống.

3.1. Phát triển cơ sở dữ liệu cho ứng dụng và áp dụng các nguyên lý SOLID với Hilt

Để phát triển cơ sở dữ liệu cho ứng dụng quản lí tài chính cá nhân, Room Database là một lựa chọn phổ biến trên nền tảng Android. Phần này trình bày cách thiết lập Room Database vào dự án, cùng việc tích hợp Dependency Injection để quản lý các thành phần trong dự án.

3.1.1. Room Database

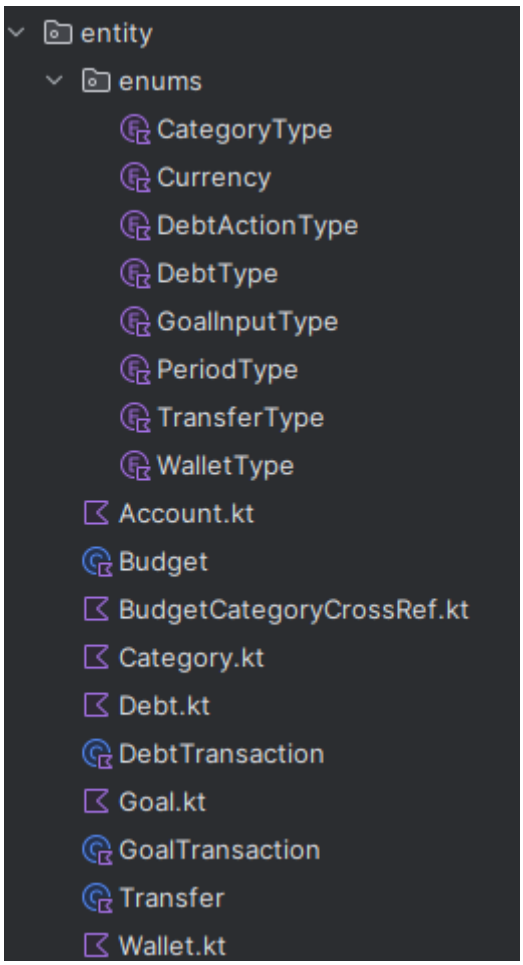
Bước 1: Thêm thư viện Room vào dự án

```
plugins {
    alias(libs.plugins.android.application)
    alias(libs.plugins.kotlin.android)
    id("kotlin-kapt")
    id("androidx.room")
    id("com.google.dagger.hilt.android")
    id("kotlin-parcelize")
}
```

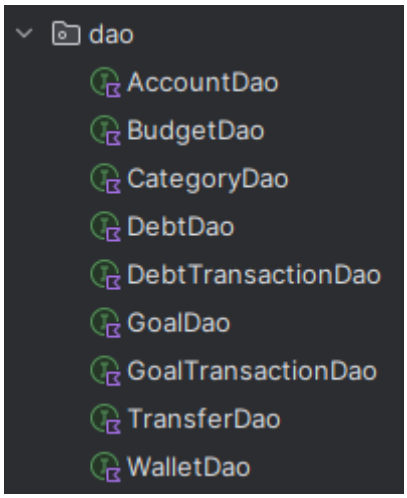
```
android {
        jvmTarget = "11"
    }
    room {
        schemaDirectory( path: "$projectDir/schemas")
    }
}
```

```
implementation(libs.androidx.room.runtime)
//noinspection KaptUsageInsteadOfKsp
kapt(libs.androidx.room.compiler)
implementation(libs.androidx.room.ktx)
```

Bước 2: Tạo các bảng



Bước 3: Tạo các Dao



Bước 4: Tạo Room Database

```
@Database(
    entities = [
        Account::class, Budget::class, Goal::class, Category::class, BudgetCategoryCrossRef::class,
        Transfer::class, Wallet::class, Debt::class, DebtTransaction::class, GoalTransaction::class
    ],
    version = 1,
    exportSchema = false
)
abstract class AppDatabase : RoomDatabase(){
    abstract fun accountDao(): AccountDao
    abstract fun budgetDao(): BudgetDao
    abstract fun debtDao(): DebtDao
    abstract fun debtTransactionDao(): DebtTransactionDao
    abstract fun categoryDao(): CategoryDao
    abstract fun goalDao(): GoalDao
    abstract fun goalTransactionDao(): GoalTransactionDao
    abstract fun transferDao(): TransferDao
    abstract fun walletDao(): WalletDao
}
```

3.1.2. Dagger Hilt

Bước 1: Thêm thư viện vào dự án

```
implementation(libs.hilt.android)
kapt(libs.hilt.android.compiler)
```

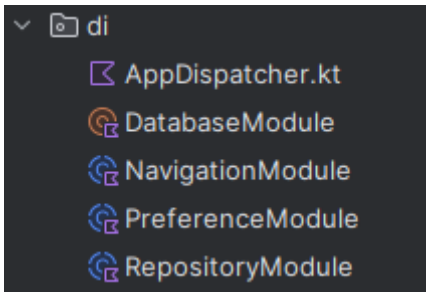
Bước 2: Khởi tạo Hilt trong ứng dụng

```
@HiltAndroidApp
class MoneyManagerApplication : BaseApplication()
```

```
<application
    android:name=".MoneyManagerApplication"
    android:allowBackup="true"
    android:dataExtractionRules="@xml/data_extraction_rules"
    android:fullBackupContent="@xml/backup_rules"
    android:icon="@mipmap/ic_launcher"
    android:label="Money_Manager_App"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/Theme.MoneyManager"
    tools:targetApi="31" >
    <activity
        android:name=".activity.MainActivity"
        android:exported="true" >
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

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

Bước 3: Tạo Module cung cấp các thành phần như Room Database giúp hỗ trợ Singleton design pattern



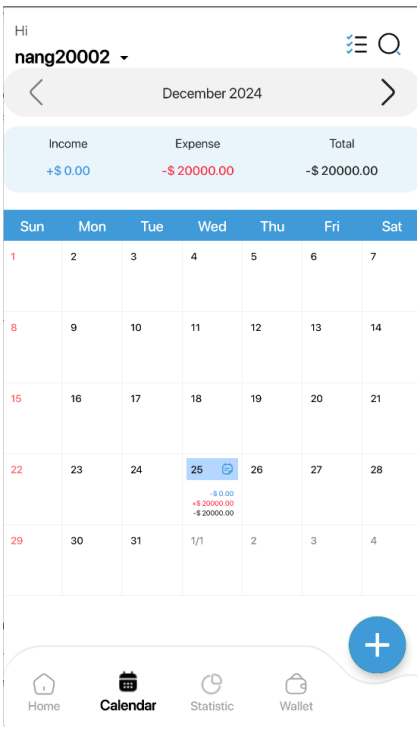

```kotlin
@Module
@InstallIn(SingletonComponent::class)
object DatabaseModule {
    @Provides
    @Dispatcher(AppDispatchers.IO)
    fun providesIODispatcher(): CoroutineDispatcher = Dispatchers.IO

    @Provides
    @Dispatcher(AppDispatchers.Default)
    fun providesDefaultDispatcher(): CoroutineDispatcher = Dispatchers.Default

    @Provides
    @Singleton
    fun provideAppDatabase(@ApplicationContext context: Context): AppDatabase {
        return Room.databaseBuilder(
            context, AppDatabase::class.java, name: "app_database"
        ).build()
    }
```

3.2. Chức năng xem giao dịch theo lịch

Sau khi người dùng thêm các giao dịch thì danh sách các giao dịch sẽ được hiện thì vào trang lịch này, để người dùn có thể dễ dàng theo dõi được chi tiêu hàng tháng



Hình 3.1: Giao diện xem tổng chi và thu theo tháng.

Bảng lịch dùng một adapter để hiện thị gồm 7 cột và 6 dòng, để custom được như vậy ta cần phải xét từng vị trí một.

Đầu tiên ta phải xác định được số phần tử:

```
override fun getItemCount(): Int {
    val dayOfMonth = CalendarHelper.getDayOfMonth(date)
    var dayOfWeek = CalendarHelper.getDayOfWeek(date) -
            SharePreferenceHelper.getFirstDayOfWeek(context)
    if (dayOfWeek < 0) dayOfWeek += 7
    val totalDays = dayOfMonth + dayOfWeek
    return (((totalDays / 7) + if (totalDays % 7 == 0) 0 else 1) * 7) + 7
}
```

Ta có biến dayOfMonth tính số ngày của tháng hiện tại, dayOfWeek tính số ngày trống trước đó, totalDays là tổng số ngày cần hiện thị. Sau khi tính được thì ta sẽ chia cho 7 để ra số tuần cần thiết, nếu do dư tức là ta cần phải cộng thêm một tuần nữa. Cuối cùng phải cộng thêm 7 vì có thêm một hàng tiêu đề.

Đầu tiên ta kiểm tra xem cái vị trí item. Nếu vị trí item bé hơn 7 nghĩa là nó là tiêu đề.

```
override fun getItemViewType(position: Int): Int =
 if (position >= 7) 1 else 0
```

Tiếp theo ta xem đoạn code này, Biến firstDayOfWeek vị trí trong tuần hiện tại. Ví dụ firstDayOfWeek = 6 thì weekdays[firstDayOfWeek] trả về là hôm chủ nhật.

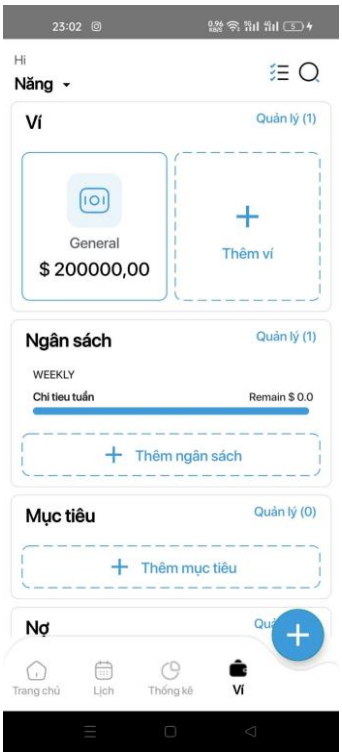
```
if (getItemViewType(position) == 0) {
    val headerViewHolder = holder as HeaderViewHolder
    var firstDayOfWeek = SharePreferenceHelper.getFirstDayOfWeek(context)
- 1 + position
    if (firstDayOfWeek >= 7) {
        firstDayOfWeek -= 7
    }
```

Ở đoạn code này ta sẽ thiết lập chiều cao của ô. Sau khi tiết lập xong chiều cao cho các ô thì đến phần gán giao trị cho các ô, ở phần lịch này sẽ có 3 phần. Phần 1 sẽ là các ngày trước tháng đấy, các ngày trong tháng, và các ngày sau tháng đó.

```
    headerViewHolder.weekLabel.setTextColor(
        context.getColor(R.color.white)
    )
    headerViewHolder.weekLabel.text = weekdays[firstDayOfWeek]
    return
}
    val itemViewHolder = holder as ItemViewHolder
    var height = (itemViewHolder.parent.height -
    Helper.convertDpToPixel(context, 30f)) / ((itemCount - 7) / 7f)
    val layoutParams = itemViewHolder.itemView.layoutParams as
    RecyclerView.LayoutParams
    if (height <= Helper.convertDpToPixel(context, 70f)) {
        height = Helper.convertDpToPix el(context, 70f)
    }
    layoutParams.height = height.toInt()
    itemViewHolder.itemView.layoutParams = layoutParams
```

3.3. Chức năng xem sửa budget

Giao diện ban đầu khi bấm vào quản lí wallet sẽ hiện lên tất cả các danh sách buget người dùng đã tạo, trước khi tạo ta cần phải cập nhật lại cho nó cái chu kì của budget đó



Hình 3.2: Giao diện ngân sách

```
    if (listBudget.isEmpty()) {
        for (item in listBudget) {
            if (item.endDate < todayDate) {
                val start_date = item.startDate
```

```
                val end_date = item.endDate
                val periodType = item.periodType
                val newPeriod = updatePeriodBudget(start_date, end_date,
    todayDate, periodType)
                val newStartDate = newPeriod.first
                val newEndDate = newPeriod.second
                budgetRepository.editBudget(item.copy(startDate =
    newStartDate, endDate = newEndDate))
            }
        }
    }
```

Đoạn code bên trên sẽ tiến hành cập nhật chu kì cho buget. Nó sẽ kiểm tra xem ngày hiện tại có lơn hơn ngày kết thúc của chu kì không, nếu mà lớn hơn sẽ bắt đầu cập nhật lại ngày bắt đầu và ngày kết thúc. Nó sẽ gọi đến hàm updatePeriodBudget để cập nhật lại ngày bắt đầu và ngày kết thúc.

```
    PeriodType.WEEKLY -> {
        val weeksPassed = diffInDays / 7
        var newStartDate = endDate
        val calendarNew = Calendar.getInstance()
        calendarNew.timeInMillis = newStartDate
        calendarNew.add(Calendar.WEEK_OF_YEAR, weeksPassed.toInt())
        newStartDate = calendarNew.timeInMillis
        calendarNew.add(Calendar.WEEK_OF_YEAR, 1)
        val newEndDate = calendarNew.timeInMillis
        return Pair(newStartDate, newEndDate)
    }
```

Trọng đoạn code này, kiểu tra xem chu kì của budget này là tháng , năm hay tuần. Nếu là tuần thì bắt đầu sẽ tính ngày chênh lệch rồi chia cho 7 để ra chu kì. Ngày kết thu cộng thêm số chi kì đó sẽ ra ngày bắt đầu hiện tại, còn ngày kết thúc của chu kì mới thì sẽ cộng thêm 1 bằng cách gọi hàm: calendarNew.add(Calendar.WEEK_OF_YEAR,1).

3.4. Giao diện xem chi tiết Budget

Đoạn code sau đây dùng để lấy ra danh sách các transaction của budget theo các category vì mỗi loại transaction đề có một category nhất định ta chỉ cần xem nó thuộc loại category nào để tính ra.

```
fun getCategoryWithTransfer(budgetWithCategory: BudgetWithCategory) {
    viewModelScope.launch(ioDispatcher) {
        var listcategoryWithTransfer =
mutableListOf<CategoryWithTransfer>()
        for(i in budgetWithCategory.categories){
            var listTransfer =
getCategoryWithTransfer(budgetWithCategory.budget.accountId,
                i.id, budgetWithCategory.budget.startDate,
                budgetWithCategory.budget.endDate)
            listcategoryWithTransfer.add(CategoryWithTransfer(i,
listTransfer))
        }
        _listCategoryWithTransfer.value = listcategoryWithTransfer
    }
}
```

Sau khi ta lấy được `listCategoryWithTransfer` ta sẽ tính toán xem tổng các chi tiêu của các transfer đó

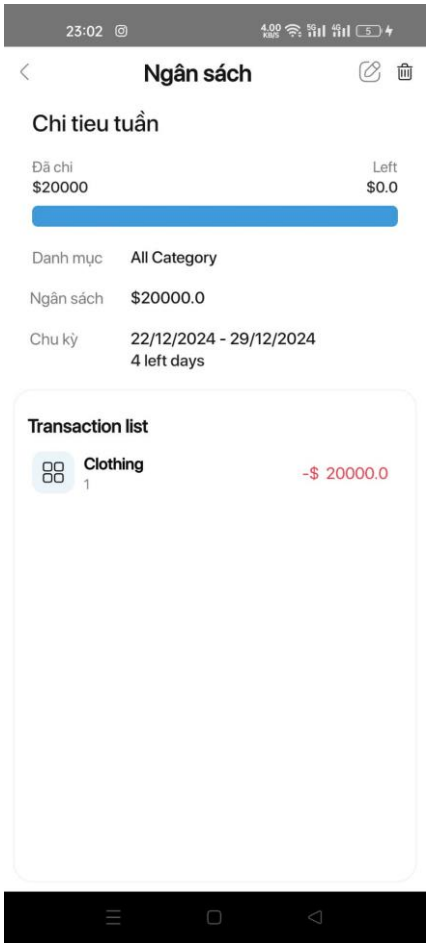
```
fun toCategoryTransactionDetail(listCategoryWithTransfer:
List<CategoryWithTransfer>): List<CategoryTransactionDetail> {
```

```
    var listCategoryTransactionDetail =
mutableListOf<CategoryTransactionDetail>()
    for(i in listCategoryWithTransfer){
        if(i.transfers.isNotEmpty()){
            var total = 0.0
            var name = i.category.name
            var totalCategoryTransaction = 0
            for(j in i.transfers){
                total += j.amount
                totalCategoryTransaction++
            }

listCategoryTransactionDetail.add(CategoryTransactionDetail(name = name,
totalAmount = total,
                totalCategoryTransaction = totalCategoryTransaction,
                listTransfer = i.transfers))
        }

    }
    return listCategoryTransactionDetail
}
```

Sau khi lấy được danh sách chi tiết các giao dịch sẽ được hiện lên trang chi tiết budget.



Hình 3.3: Giao diện chi tiết ngân sách

3.5. Giao diện thống kê

Đầu tiên ta sẽ lấy ra tất cả các danh sách transaction, rồi sau đố sẽ thống kê theo các danh mục , ở bước này sẽ truy vấn các cơ sở dữ liệu để lấy được ra các transaction theo ngày tháng truyền vào bên trong

```
fun getCalendarSummary(listWallet: List<Wallet>, idAccount: Long){
    viewModelScope.launch(ioDispatcher){
        var listTrasaction = mutableListOf<Transaction>()
        for(wallet in listWallet){
           listTrasaction.addAll(getDebtTransaction(wallet.id,idAccount))
            listTrasaction.addAll(getTransfer(wallet.id,idAccount))
            listTrasaction.addAll(getDebt(wallet.id,idAccount))

        }
        getStats(listTrasaction)
        setCalendarSummary(totalCalendarSummary(listTrasaction))

    }
}
```

Tiếp theo chúng ta sẽ bắt đầu thống kê như sau :

B1: tạo ra một danh sách gồm 2 mảng để lưu thu nhập và doanh thu , mục đích của việc tạo ra như vậy, như kiểu một map khi gặp các loại giao dịch trùng thì sẽ tự động tăng lên một transaction

```
var categoryList = categoryRepository.getAllCategory()
var listStatsIncome = mutableListOf<Stats>()
var listStatsExpense: MutableList<Stats> = mutableListOf()
```

và thực hiện kiểm tra

```
fun checkCategory(category: Category): Boolean{
    return category.type == CategoryType.INCOME ||
            category.type == CategoryType.PAYABLE ||
            category.type == CategoryType.DEBT_COLLECTION ||
            category.type == CategoryType.DEBT_INCREASE
}
```

nếu điều kiện này mà đúng thì sẽ thêm vào danh sách `listStatsIncome` và ngược lại sẽ thêm vào `listStatsExpense`

B2: Thực hiện lần lượt tính tổng số tiền và cập nhật số lượng giao dịch để ra được số tiền mà khách hàng đã chi tiêu ra cho mỗi danh mục mặt hàng.

```
for(transaction in listTransaction){
    when(transaction){
        is Transfer -> {
            if(transaction.typeOfExpenditure == TransferType.Income){
                for(stats in listStatsIncome){
                    if(stats.icon == transaction.categoryId){
                        stats.amount += transaction.amount
                        stats.trans++
                    }
                }
            }

      }

}
```

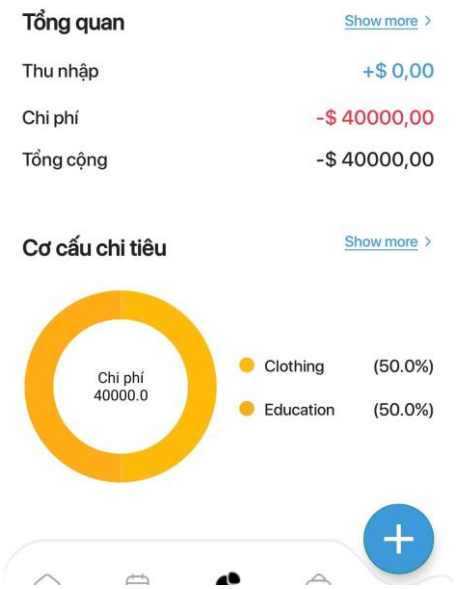
B3: Tính số phần trăm và sắp xếp lại giao dịch

```
if (listStatsExpense.size > 1) {
    for (i in 0 until listStatsExpense.size - 1) {
        val stat = listStatsExpense[i]
        val percent = (stat.amount / totalAmountExpense) * 100
        val bigDecimal = BigDecimal(percent).setScale(2,
RoundingMode.HALF_UP)
        val number = bigDecimal.toDouble()
        stat.percent = number
    }
    listStatsExpense[listStatsExpense.size - 1].percent = 100 -
listStatsExpense.sumOf { it.percent }
} else {
    if(listStatsExpense.size == 1) listStatsExpense[0].percent = 100.0
}
```

Sắp xếp lại các danh mục chi tiêu đã tính toán

```
listStatsExpense.sortByDescending { it.amount }
listStatsIncome.sortByDescending { it.amount }
```

Sau đó ta sẽ thu được một giao diện như sau :



Hình 3.4: Giao diện thống kê chi tiêu theo danh mục

Tổng Kết

1. Kết quả đạt được

- Hệ thống đã hoàn thành và đáp ứng các chức năng cơ bản của một ứng dụng quản lý tài chính cá nhân trên thiết bị di động. Điều này bao gồm khả năng theo dõi chi tiêu, quản lý thu nhập, và thiết lập các kế hoạch tài chính. Các tính năng được thiết kế nhằm hỗ trợ người dùng quản lý tài chính cá nhân một cách đơn giản và hiệu quả.
- Ứng dụng đã được triển khai thành công trên nền tảng Android, đảm bảo khả năng hoạt động ổn định. Trải nghiệm người dùng được đánh giá là mượt mà với giao diện thân thiện, phù hợp cho nhiều nhóm đối tượng sử dụng, từ người dùng cơ bản đến người có nhu cầu quản lý tài chính chuyên sâu.

2. Những hạn chế

- Ứng dụng hiện tại chưa tích hợp được công nghệ trí tuệ nhân tạo (AI) để hỗ trợ người dùng đánh giá khả năng tài chính, phân tích dữ liệu thu chi và đề xuất các kế hoạch tiết kiệm hoặc đầu tư. Đây là một điểm hạn chế so với các ứng dụng hiện đại cùng phân khúc
- Một số chức năng trong ứng dụng vẫn chưa được tối ưu hóa hoàn toàn, dẫn đến khó khăn khi thao tác. Người dùng có thể gặp khó khăn trong việc sử dụng một số tính năng nâng cao hoặc khi quản lý dữ liệu phức tạp.
- Giao diện của một số màn hình trong ứng dụng còn khá đơn giản và chưa thật sự thu hút về mặt thẩm mỹ. Điều này có thể làm giảm trải nghiệm người dùng, đặc biệt đối với những người có yêu cầu cao về thiết kế giao diện.

3. Hướng phát triển tiếp theo

- Cải thiện giao diện người dùng: Nâng cao tính thẩm mỹ của ứng dụng bằng cách thiết kế giao diện hiện đại, thân thiện hơn và dễ dàng thao tác. Chú trọng vào việc tạo trải nghiệm tốt nhất cho người dùng thông qua việc bố trí hợp lý các thành phần, màu sắc và phong cách đồ họa.
- Phát triển thêm tính năng mới: Bổ sung các tính năng nâng cao như quản lý tài khoản ngân hàng, nhắc nhở thanh toán hóa đơn, lập kế hoạch đầu tư, và tạo báo cáo tài chính chi tiết. Điều này sẽ giúp ứng dụng trở nên toàn diện hơn, đáp ứng nhiều nhu cầu đa dạng của người dùng.
- Đồng bộ hóa dữ liệu: Tích hợp thêm tính năng đồng bộ hóa dữ liệu với các dịch vụ đám mây như Google Drive, cho phép người dùng dễ dàng sao lưu và khôi phục dữ liệu khi cần. Ngoài ra, bổ sung tùy chọn đăng nhập bằng tài khoản của bên thứ ba (Google, Facebook, Apple ID) để nâng cao sự tiện lợi và bảo mật.

TÀI LIỆU THAM KHẢO

[1] Trần Đình Quế. Phân tích và thiết kế hệ thống thông tin. Học viện Công nghệ Bưu chính Viễn thông. 2018

[2] "Understanding MVVM: Model-View-ViewModel Architecture." [Online]. Available: https://learn.microsoft.com/en-us/xamarin/xamarin-forms/enterprise-application-patterns/mvvm.

[3] "Save data in a local database using Room." [Online]. Available: https://developer.android.com/training/data-storage/room

[4] "Dependency injection with Hilt. " [Online]. Available: https://developer.android.com/training/dependency-injection/hilt-android

[5] "SOLID: The First 5 Principles of Object Oriented Design." [Online]. Available: https://www.digitalocean.com/community/conceptual-articles/s-o-l-i-d-the-first-five-principles-of-object-oriented-design

[6] "Navigation Component. " [Online]. Available: https://developer.android.com/guide/navigation