BỘ CÔNG THƯƠNG

TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI

ĐỒ ÁN TỐT NGHIỆP

NGÀNH CÔNG NGHỆ THÔNG TIN

TÊN ĐỀ TÀI: XÂY DỰNG ỨNG DỤNG DI ĐỘNG QUẢN LÝ CHI TIÊU SỬ DỤNG FLUTTER

Giảng viên hướng dẫn: **TS. Lương Thị Hồng Lan**

Sinh viên: **Nguyễn Thu Hoài**

Mã sinh viên: **2021601012**

Hà Nội - Năm 2025

BỘ CÔNG THƯƠNG

TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI

ĐỒ ÁN TỐT NGHIỆP

NGÀNH CÔNG NGHỆ THÔNG TIN

TÊN ĐỀ TÀI: XÂY DỰNG ỨNG DỤNG DI ĐỘNG QUẢN LÝ CHI TIÊU SỬ DỤNG FLUTTER

Giảng viên hướng dẫn: **TS. Lương Thị Hồng Lan**

Sinh viên: **Nguyễn Thu Hoài**

Mã sinh viên: **2021601012**

Hà Nội - Năm 2025

MỤC LỤC

DANH MỤC CÁC KÝ HIỆU, CÁC CHỮ VIẾT TẮT

STT	Viết tắt	Dịch nghĩa
1	HTML/CSS	HyperText Markup Language/Cascading Style Sheets
2	SQL	Structured Query Language
3	XML	Extensible Markup Language
4	RDBMS	Hệ quản trị cơ sở dữ liệu quan hệ
5	MVC	Model-View-Controller
6	CSDL	Cơ sở dữ liệu
7	SDK	Software Development Kit

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG BIỂU

LỜI CẢM ƠN

Trong suốt thời gian học tập và thực hiện đồ án tốt nghiệp em luôn nhận được sự hướng dẫn nhiệt tình từ phía nhà trường và các thầy cô giáo. Vì thế, lời đầu tiên em xin chân thành cảm ơn trường Đại học Công nghiệp Hà Nội, Trường Công nghệ thông tin & Truyền thông, thầy cô giáo khoa Công nghệ thông tin đã tạo điều kiện thuận lợi và truyền dạy kiến thức cho em trong thời gian qua để em hoàn thành đề tài một cách tốt nhất. Và hơn hết em xin bày tỏ lòng biết ơn sâu sắc nhất tới cô giáo hướng dẫn - **TS. Lương Thị Hồng Lan** đã tận tình giúp đỡ, định hướng em rất nhiều trong suốt quá trình tìm hiểu nghiên cứu và hoàn thành đồ án tốt nghiệp. Những góp ý chuyên môn, những chia sẻ kinh nghiệm và sự khích lệ từ cô chính là động lực lớn lao để em không ngừng cố gắng, vượt qua những khó khăn trong quá trình nghiên cứu và hoàn thiện đề tài.

Em rất mong nhận được sự đóng góp của tất cả các thầy cô giáo để đồ án của em được đầy đủ và hoàn chỉnh hơn, đồng thời rút ra được những bài học bổ ích cho con đường học tập và làm việc sau này.

Em xin chân thành cảm ơn!

MỞ ĐẦU

1. Lý do chọn đề tài

Trong cuộc sống hiện đại ngày nay, việc quản lý chi tiêu đóng vai trò vô cùng quan trọng đối với mỗi cá nhân, đặc biệt là trong bối cảnh nền kinh tế ngày càng phát triển, nhu cầu tiêu dùng tăng cao và sự đa dạng của các hình thức thanh toán không tiền mặt. Quản lý chi tiêu hiệu quả không chỉ giúp cá nhân kiểm soát tài chính, tiết kiệm hợp lý mà còn góp phần xây dựng một lối sống khoa học, ổn định và có kế hoạch. Tuy nhiên, trên thực tế, không phải ai cũng có khả năng quản lý tài chính cá nhân một cách tốt, nhất là đối với những người mới bắt đầu sống tự lập, sinh viên hoặc người có thu nhập trung bình – thấp. Việc ghi chép thủ công bằng sổ tay hoặc bảng tính truyền thống thường kém hiệu quả, dễ bỏ sót và thiếu sự trực quan, từ đó ảnh hưởng đến việc theo dõi và đánh giá các khoản thu – chi.

Trong bối cảnh công nghệ thông tin không ngừng phát triển, các ứng dụng di động hỗ trợ quản lý chi tiêu đã trở thành những công cụ thiết yếu và tiện lợi, giúp người dùng dễ dàng ghi nhận, phân tích và kiểm soát dòng tiền cá nhân một cách nhanh chóng, chính xác và hiệu quả. Với nhu cầu thực tiễn đó, việc xây dựng một ứng dụng quản lý chi tiêu là hoàn toàn phù hợp và mang ý nghĩa thiết thực.

Flutter – một bộ công cụ phát triển giao diện người dùng mã nguồn mở do Google phát triển – đã và đang nhận được sự quan tâm rộng rãi trong cộng đồng lập trình. Flutter cho phép lập trình viên phát triển ứng dụng đa nền tảng (Android và iOS) chỉ với một lần viết mã, nhờ vào khả năng biên dịch hiệu suất cao, giao diện đẹp mắt và dễ tùy chỉnh. Đây là một lựa chọn lý tưởng cho các cá nhân và nhóm phát triển có nhu cầu xây dựng sản phẩm nhanh chóng, tiết kiệm tài nguyên và đồng thời hỗ trợ nhiều nền tảng.

Xuất phát từ nhu cầu thực tế cũng như mong muốn nâng cao kỹ năng lập trình di động đa nền tảng, em đã quyết định chọn đề tài **"Ứng dụng quản lý chi tiêu bằng Flutter"** làm hướng nghiên cứu cho đồ án tốt nghiệp của mình. Đề tài không chỉ giúp em hiểu sâu hơn về cách xây dựng và phát triển ứng dụng trên nền tảng Flutter mà còn là cơ hội để em vận dụng kiến thức đã học vào giải quyết một vấn đề gần gũi, thiết thực trong cuộc sống hàng ngày.

Ngoài ra, việc phát triển một sản phẩm công nghệ thực tế còn giúp em rèn luyện các kỹ năng về thiết kế giao diện người dùng, xử lý dữ liệu, tư duy hệ thống cũng như khả năng phân tích và giải quyết vấn đề – những kỹ năng quan trọng đối với một lập trình viên trong tương lai.

2. Đối tượng, mục tiêu và phạm vi nghiên cứu

a. Mục tiêu nghiên cứu

Mục tiêu nghiên cứu của đề tài "Ứng dụng quản lý chi tiêu bằng Flutter" bao gồm:

- Nghiên cứu và phân tích các nhu cầu và yêu cầu của người dùng về ứng dụng quản lý chi tiêu.
- Tìm hiểu và áp dụng công nghệ xây dựng ứng dụng di động đa nền tảng Flutter để phát triển một ứng dụng quản lý chi tiêu đáp ứng các yêu cầu đặt ra.
- Xây dựng và thiết kế giao diện cho ứng dụng quản lý chi tiêu bằng Flutter với mục đích đảm bảo tính cơ động và dễ sử dụng cho người dùng.
- Đảm bảo tính hiệu quả của ứng dụng thông qua việc kiểm thử và đánh giá, đồng thời đảm bảo tính ổn định và bảo mật thông tin cá nhân của người dùng trong quá trình sử dụng ứng dụng.

- Tìm hiểu và áp dụng các công nghệ mới và tiên tiến trong quá trình phát triển ứng dụng để nâng cao tính năng và trải nghiệm của người dùng.

b. Đối tượng nghiên cứu

- Ngôn ngữ lập trình Dart để phát triển ứng dụng trên Flutter, cùng với các công cụ hỗ trợ như Visual Studio Code
- Mô hình MVC (Model-View-Controller) được sử dụng để phát triển ứng dụng, kèm theo đó là sử dụng các cơ sở dữ liệu như Firebase và SQLite để lưu trữ thông tin và dữ liệu cho ứng dụng.
- Quy trình phát triển phần mềm
- Các vấn đề liên quan đến quản lý chi tiêu

c. Phạm vi nghiên cứu

Phạm vi nghiên cứu của đề tài "Ứng dụng quản lý chi tiêu bằng Flutter" bao gồm:

- Phát triển một ứng dụng di động đa nền tảng bằng Flutter để hỗ trợ người dùng quản lý chi tiêu hàng ngày.
- Các tính năng của ứng dụng bao gồm quản lý chi tiêu, quản lý thu nhập, thống kê chi tiêu và thu nhập theo thời gian và loại, quản lý tài khoản ngân hàng, tạo và quản lý các khoản vay, tạo và quản lý các mục tiêu tài chính, báo cáo tổng quan về tình hình tài chính.
- Phát triển giao diện người dùng thân thiện và dễ sử dụng.
- Tối ưu hiệu suất ứng dụng và tính đa nền tảng bằng Flutter.

3. Phương pháp nghiên cứu

- Phương pháp nghiên cứu về mặt lý thuyết:
  - Tiến hành nghiên cứu và thu thập các tài liệu liên quan đến đề tài, bao gồm các tài liệu về quản lý chi tiêu, ứng dụng di động, Flutter, và các công nghệ liên quan.
  - Tổng hợp và phân tích các tài liệu thu thập được.

- Chọn lọc và sử dụng các tài liệu nghiên cứu để báo cáo đồ án tốt nghiệp.

- Phương pháp nghiên cứu trong thực nghiệm:
  - Tìm hiểu các tiêu chí kiểm thử chức năng và cách thức thực hiện.
  - Phân tích các ứng dụng có chức năng tương tự để tìm hiểu các tiêu chuẩn thiết kế và triển khai ứng dụng.
  - Tiến hành xây dựng ứng dụng theo phân tích và yêu cầu thực tế.
  - Tìm hiểu và sử dụng các công nghệ liên quan đến Flutter như Firebase để xây dựng tính năng đăng nhập và đồng bộ dữ liệu.
  - Tổng hợp kiến thức đã học và hoàn thành báo cáo đồ án tốt nghiệp.

CHƯƠNG 1. TỔNG QUAN VỀ NỘI DUNG NGHIÊN CỨU

1.1. Giới thiệu chung

Trong bối cảnh xã hội hiện đại, khi nhu cầu tiêu dùng ngày càng tăng và các hình thức thanh toán không dùng tiền mặt ngày càng phổ biến, việc quản lý tài chính cá nhân trở thành một yếu tố thiết yếu trong cuộc sống. Quản lý chi tiêu không chỉ giúp mỗi người kiểm soát thu nhập và các khoản chi hằng ngày, mà còn hỗ trợ họ lên kế hoạch tiết kiệm, đầu tư và sử dụng nguồn tài chính một cách hợp lý và khoa học. Tuy nhiên, không phải ai cũng có khả năng theo dõi và kiểm soát các giao dịch tài chính cá nhân một cách hiệu quả, đặc biệt là trong nhịp sống bận rộn hiện nay. Do đó, các ứng dụng hỗ trợ quản lý chi tiêu đã ra đời như một giải pháp thiết thực, đóng vai trò như một trợ lý tài chính cá nhân thông minh và tiện lợi.

Việc sử dụng Flutter – một nền tảng phát triển ứng dụng do Google phát triển – đang dần trở thành một công cụ phổ biến trong cộng đồng lập trình nhờ khả năng phát triển ứng dụng đa nền tảng với chỉ một lần viết mã. Flutter không chỉ giúp rút ngắn thời gian phát triển ứng dụng, tiết kiệm chi phí mà còn sở hữu hệ sinh thái các widget phong phú, hỗ trợ xây dựng giao diện người dùng đẹp mắt, linh hoạt và dễ sử dụng. Chính vì vậy, việc lựa chọn Flutter để phát triển một ứng dụng quản lý chi tiêu là một hướng đi hợp lý, đáp ứng cả về mặt công nghệ lẫn nhu cầu thực tiễn.

Ứng dụng quản lý chi tiêu được xây dựng bằng Flutter sẽ giúp người dùng dễ dàng ghi lại các khoản chi tiêu và thu nhập hằng ngày, phân loại chúng theo từng danh mục cụ thể như ăn uống, đi lại, học tập hay giải trí. Người dùng cũng có thể theo dõi lịch sử thu – chi thông qua các biểu đồ trực quan, từ đó có cái nhìn tổng thể về tình hình tài chính cá nhân. Ứng dụng còn cho phép lập kế hoạch chi tiêu theo tháng hoặc theo từng mục tiêu cụ thể, nhắc nhở người dùng khi đến hạn thanh toán hóa đơn, đồng thời hỗ trợ quản lý tài khoản ngân hàng và theo dõi số dư. Bên cạnh đó, việc cung cấp các báo cáo tài chính định kỳ

giúp người dùng đánh giá được hiệu quả trong cách quản lý tiền bạc của bản thân.

Tổng kết lại, việc phát triển một ứng dụng quản lý chi tiêu bằng Flutter không chỉ mang lại lợi ích về mặt kỹ thuật cho người xây dựng ứng dụng mà còn mang lại giá trị sử dụng thực tiễn cao cho người dùng cuối. Đây là một giải pháp công nghệ hiệu quả và phù hợp với xu hướng phát triển của thời đại số, góp phần thúc đẩy việc xây dựng lối sống tài chính chủ động, minh bạch và bền vững cho mỗi cá nhân, giúp người dùng quản lý tài chính cá nhân một cách dễ dàng và hiệu quả hơn.

1.2. Khảo sát các phần mềm liên quan

Mục tiêu của việc khảo sát các phần mềm liên quan là nhằm tìm hiểu tổng quan thị trường ứng dụng quản lý chi tiêu hiện nay, nhận diện điểm mạnh, điểm yếu và nhu cầu thực tế của người dùng, từ đó làm cơ sở đề xuất các tính năng phù hợp và định hướng phát triển ứng dụng quản lý chi tiêu tối ưu hơn cho người Việt trên nền tảng Flutter.

1.2.1. Money Lover

Money Lover là một trong những ứng dụng quản lý chi tiêu cá nhân phổ biến tại Việt Nam, được phát triển bởi công ty Finsify. Ứng dụng hỗ trợ đa nền tảng, bao gồm Android, iOS và phiên bản web, giúp người dùng dễ dàng ghi chép và theo dõi thu chi mọi lúc, mọi nơi.

Money Lover cung cấp đầy đủ các chức năng cơ bản cần thiết cho việc quản lý tài chính cá nhân, bao gồm:

- Ghi chép các khoản thu chi hàng ngày.
- Phân loại các khoản chi theo danh mục (ăn uống, di chuyển, mua sắm, v.v.).
- Tạo và quản lý nhiều ví tiền khác nhau (ví cá nhân, ví gia đình, ví tiết kiệm, v.v.).
- Thiết lập ngân sách chi tiêu theo tháng hoặc theo danh mục cụ thể.

- Hiển thị báo cáo chi tiêu dưới dạng biểu đồ trực quan.
- Nhắc nhở thanh toán hóa đơn và quản lý các khoản nợ.

Một trong những điểm mạnh lớn nhất của Money Lover là giao diện thân thiện, hỗ trợ hoàn toàn bằng tiếng Việt, phù hợp với đa số người dùng trong nước. Ứng dụng có thiết kế đơn giản, dễ sử dụng ngay cả với người dùng mới, đồng thời có khả năng hoạt động ổn định trên nhiều thiết bị. Việc phân loại thu chi rõ ràng, kết hợp với báo cáo tài chính minh bạch, giúp người dùng dễ dàng theo dõi tình hình tài chính của bản thân. Tuy nhiên, Money Lover vẫn tồn tại một số hạn chế nhất định. Các tính năng nâng cao như xuất dữ liệu ra Excel, đồng bộ hóa đa thiết bị theo thời gian thực hoặc phân tích tự động thường chỉ có trong phiên bản trả phí. Giao diện mặc dù thân thiện nhưng vẫn mang tính cổ điển, chưa thực sự hiện đại theo xu hướng thiết kế mới. Ngoài ra, tốc độ phản hồi đôi khi bị chậm trên các thiết bị cấu hình thấp.

Tóm lại, Money Lover đáp ứng tốt nhu cầu cơ bản của người dùng Việt trong quản lý chi tiêu, nhưng vẫn còn hạn chế về tính năng nâng cao và giao diện người dùng, là cơ sở để đề xuất những cải tiến cho ứng dụng mới.

1.2.2. Misa Money Keeper (Sổ thu chi Misa)

Misa Money Keeper, hay còn gọi là Sổ thu chi MISA, là một ứng dụng quản lý chi tiêu cá nhân được phát triển bởi Công ty Cổ phần MISA – một doanh nghiệp công nghệ nổi tiếng tại Việt Nam trong lĩnh vực phần mềm kế toán và quản trị. Ứng dụng được thiết kế nhằm hỗ trợ người dùng Việt ghi chép và theo dõi thu chi một cách dễ dàng, khoa học. Misa Money Keeper hiện hỗ trợ hai nền tảng chính là Android và iOS.

Ứng dụng cung cấp các chức năng cơ bản như ghi chép thu nhập và chi tiêu theo ngày, quản lý nhiều ví tiền, phân loại các khoản thu chi theo danh mục, thiết lập hạn mức chi tiêu và tạo mục tiêu tiết kiệm. Ngoài ra, MISA cũng cho phép người dùng theo dõi nợ vay, nhắc nhở thanh toán định kỳ và xuất báo cáo tài chính. Dữ liệu được đồng bộ hóa và sao lưu thông qua tài khoản người dùng, đảm bảo tính an toàn và liên tục trong quá trình sử dụng.

Misa Money Keeper có giao diện đơn giản, dễ sử dụng và phù hợp với người Việt. Ứng dụng miễn phí hầu hết các tính năng cơ bản, không có quá

nhiều giới hạn so với bản trả phí. Việc sử dụng ngôn ngữ tiếng Việt và cách trình bày quen thuộc giúp người dùng dễ dàng tiếp cận và thao tác. Giao diện của ứng dụng có phần truyền thống, chưa thực sự hiện đại và thân thiện với người dùng trẻ. Một số thao tác còn phức tạp và luồng sử dụng chưa tối ưu. Ngoài ra, ứng dụng chưa hỗ trợ nền tảng web hoặc đồng bộ hóa dữ liệu giữa nhiều thiết bị một cách linh hoạt.

Tóm lại, Misa Money Keeper là một ứng dụng quản lý chi tiêu hiệu quả và dễ tiếp cận đối với người dùng Việt Nam, đặc biệt là những người ưa thích sự đơn giản. Tuy nhiên, vẫn còn hạn chế ở mặt thiết kế giao diện và trải nghiệm người dùng hiện đại, là những điểm cần xem xét khi phát triển ứng dụng mới.

1.2.3. Spendee

Spendee là một ứng dụng quản lý chi tiêu cá nhân được phát triển bởi công ty Spendee a.s. (Cộng hòa Séc). Ứng dụng này hướng đến người dùng toàn cầu, hỗ trợ nhiều ngôn ngữ, trong đó có tiếng Việt. Spendee nổi bật nhờ thiết kế hiện đại, giao diện trực quan và khả năng đồng bộ hóa với tài khoản ngân hàng ở một số quốc gia.

Spendee cho phép người dùng ghi chép thu chi hằng ngày, phân loại các khoản chi tiêu theo danh mục, thiết lập ngân sách cho từng loại chi và theo dõi biến động tài chính thông qua các biểu đồ màu sắc sinh động. Ngoài ra, ứng dụng hỗ trợ tạo nhiều ví tiền (wallets), chia sẻ ví với người dùng khác, và tích hợp dữ liệu ngân hàng (ở các quốc gia được hỗ trợ). Dữ liệu người dùng được đồng bộ hóa thông qua tài khoản đăng nhập và có thể truy cập trên cả Android, iOS và trình duyệt web.

Spendee có giao diện hiện đại, thiết kế sinh động và dễ sử dụng, đặc biệt phù hợp với người dùng trẻ. Biểu đồ và báo cáo trực quan giúp việc theo dõi chi tiêu trở nên sinh động và dễ hiểu. Việc có thể chia sẻ ví với người khác cũng là điểm cộng trong các trường hợp quản lý tài chính gia đình hoặc nhóm. Một số tính năng nâng cao của Spendee chỉ khả dụng trong phiên bản trả phí, bao gồm đồng bộ với tài khoản ngân hàng hoặc chia sẻ ví. Ngoài ra, vì không phát triển dành riêng cho thị trường Việt Nam nên phần danh mục chi tiêu và ngân hàng tích hợp chưa thực sự phù hợp với thói quen tài chính của người Việt.

Tóm lại, Spendee là một ứng dụng quản lý tài chính cá nhân mạnh mẽ với thiết kế hiện đại và khả năng phân tích tài chính trực quan. Tuy nhiên, mức độ phù hợp với người dùng Việt vẫn còn hạn chế ở mặt nội dung và tính năng, là cơ sở để các ứng dụng nội địa có thể phát triển theo hướng thân thiện hơn với thói quen và nhu cầu của người Việt.

1.2.4. PocketGuard

PocketGuard là ứng dụng quản lý chi tiêu cá nhân được phát triển bởi công ty PocketGuard, Inc., có trụ sở tại Hoa Kỳ. Ứng dụng này tập trung vào việc giúp người dùng kiểm soát tài chính cá nhân một cách thông minh và tự động hóa việc quản lý ngân sách hàng tháng. PocketGuard hỗ trợ trên cả hai nền tảng phổ biến là Android và iOS.

PocketGuard tự động liên kết với tài khoản ngân hàng và thẻ tín dụng của người dùng để tổng hợp các giao dịch chi tiêu và thu nhập. Ứng dụng phân tích các khoản chi tiêu, đề xuất ngân sách phù hợp và cảnh báo khi người dùng sắp vượt ngân sách. Ngoài ra, PocketGuard còn cung cấp tính năng theo dõi hóa đơn, thiết lập mục tiêu tiết kiệm và báo cáo chi tiêu dưới dạng biểu đồ trực quan. Dữ liệu được đồng bộ hóa đám mây và bảo mật cao.

PocketGuard nổi bật với khả năng tự động kết nối và đồng bộ dữ liệu từ nhiều nguồn tài chính, giúp người dùng tiết kiệm thời gian nhập liệu thủ công. Giao diện thân thiện, hiện đại cùng các cảnh báo thông minh giúp kiểm soát tài chính hiệu quả. Ứng dụng rất phù hợp với những người có nhu cầu quản lý tài chính kỹ lưỡng và theo dõi chi tiêu thực tế hàng ngày. Ngoài ra, một số tính năng nâng cao chỉ có trong phiên bản trả phí, làm hạn chế trải nghiệm đầy đủ cho người dùng miễn phí.

Tóm lại, PocketGuard là giải pháp quản lý chi tiêu hiện đại, thích hợp với người dùng cần tự động hóa và phân tích chi tiết tài chính cá nhân. Tuy nhiên, do hạn chế về hỗ trợ ngân hàng tại Việt Nam, ứng dụng này chưa thực sự phù hợp với người dùng trong nước và cần cải tiến để đáp ứng nhu cầu của thị trường Việt Nam.

1.3. Bài toán và khảo sát người dùng

1.3.1. Bài toán đặt ra

Trong cuộc sống hiện đại, việc quản lý tài chính cá nhân là một nhu cầu thiết yếu đối với hầu hết mọi người. Tuy nhiên, nhiều người vẫn gặp khó khăn trong việc theo dõi thu chi, lập kế hoạch chi tiêu hoặc đánh giá tình hình tài chính của bản thân theo từng thời điểm. Các lý do phổ biến dẫn đến việc quản lý tài chính không hiệu quả bao gồm:

- Không có công cụ tiện lợi để ghi lại và phân loại các khoản chi tiêu hằng ngày.
- Thiếu báo cáo trực quan để đánh giá mức độ chi tiêu theo thời gian.
- Khó khăn trong việc lập ngân sách và theo dõi các mục tiêu tiết kiệm.
- Sử dụng giấy tờ hoặc bảng tính Excel phức tạp, thiếu tính linh hoạt.

Chính vì vậy, việc xây dựng một ứng dụng quản lý chi tiêu cá nhân đơn giản, dễ sử dụng, đa nền tảng và hiệu quả là vô cùng cần thiết. Ứng dụng này sẽ giúp người dùng nhanh chóng ghi chú các khoản thu chi hàng ngày, phân loại chi tiêu theo các danh mục như ăn uống, đi lại, mua sắm, hóa đơn và nhiều loại khác. Bên cạnh đó, người dùng có thể dễ dàng xem báo cáo tổng hợp chi tiêu theo ngày, tuần hoặc tháng, đồng thời thiết lập và theo dõi ngân sách cá nhân một cách hiệu quả. Việc hỗ trợ đa nền tảng trên cả Android và iOS thông qua Flutter cũng giúp ứng dụng tiếp cận rộng rãi và thuận tiện hơn trong quá trình sử dụng.

1.3.2. Khảo sát người dùng

Để đánh giá nhu cầu thực tế và xác định các chức năng cần thiết cho ứng dụng, nhóm thực hiện đã tiến hành khảo sát với 100 người dùng tiềm năng, chủ yếu là sinh viên, nhân viên văn phòng và người đi làm trẻ tuổi, độ tuổi từ 18–35.

Mục tiêu khảo sát

- Tìm hiểu thói quen quản lý chi tiêu của người dùng hiện nay.
- Xác định các vấn đề người dùng gặp phải khi theo dõi thu chi.

- Khám phá kỳ vọng của người dùng về một ứng dụng quản lý tài chính cá nhân.

Câu hỏi khảo sát

- Câu 1: Bạn gặp khó khăn hay vấn đề gì trong việc quản lý chi tiêu?

  Trả lời:

  - Bạn đang lập ngân sách mà không có mục tiêu rõ ràng
  - Bạn quên lập ngân sách cho các khoản chi tiêu không cố định
  - Thiếu kiểm soát trong chi tiêu thường nhật
- Câu 2: Bạn có thường dễ quên và bỏ lỡ việc ghi lại chi tiêu những thứ nhỏ không?

  Trả lời:

  - Có
  - Không
- Câu 3: Bạn thường dùng cách nào để quản lý chi tiêu khoa học?

  Trả lời:

  - Sổ ghi chép, app hoặc tự nhớ
  - Dùng google sheet
  - Ghi lại trên ứng dụng Note
- Câu 4: Bạn đã dùng ứng dụng nhắc nhở nào?

  Trả lời:

  - Google sheet
  - Chưa dùng app nào
  - Dùng các ứng dụng sẵn của điện thoại
- Câu 5: Bạn hay gặp khó khăn gì khi sử dụng ứng dụng quản lý chi tiêu?

  Trả lời:

  - Giao diện hơi khó dùng
  - Chưa phù hợp với bản thân nên khó trong việc sử dụng
- Câu 6: Bạn thích nhất chức năng gì trong một ứng dụng quản lý chi tiêu?

  Trả lời:

- Thống kê tổng chi tiêu trong tháng
- Có thể thay đổi linh hoạt, nhiều chức năng để tích hợp với từng hoàn cảnh, người dùng khác nhau
- Thông báo trước một khoảng thời gian trước khi sự kiện đã lập ngân sách diễn ra
- Giao diện đẹp, trực quan.

- Câu 7: Bạn thường ghi lại chi tiêu với tần suất như thế nào?

  Trả lời:
  - Ghi hằng ngày
  - Ghi theo tuần
  - Ghi khi nhớ ra
  - Rất ít ghi lại
- Câu 8: Bạn có đặt mục tiêu tiết kiệm hàng tháng không?

  Trả lời:
  - Có
  - Không
  - Thỉnh thoảng
- Câu 9: Bạn mong muốn ứng dụng hỗ trợ quản lý bao nhiêu ví/nguồn tiền (ngân hàng, tiền mặt, ví điện tử...)?

  Trả lời:
  - Một ví là đủ
  - Từ 2-3 ví
  - Càng nhiều càng tốt miễn dễ quản lý
- Câu 10: Bạn có nhu cầu đồng bộ dữ liệu giữa các thiết bị (điện thoại, máy tính bảng, máy tính)?

  Trả lời:
  - Có
  - Không
  - Không quan trọng

- Câu 11: Bạn có quan tâm đến bảo mật thông tin tài chính cá nhân trên ứng dụng không?

  Trả lời:

  - Rất quan tâm
  - Có quan tâm
  - Không quan tâm lắm
- Câu 12: Bạn có muốn ứng dụng đưa ra đề xuất tiết kiệm hoặc kế hoạch tài chính theo thói quen chi tiêu không?

  Trả lời:

  - Có, rất cần
  - Có thể có
  - Không cần thiết

Kết quả khảo sát

- **65%** người được hỏi không có thói quen ghi chép chi tiêu hàng ngày.
- **80%** từng cảm thấy khó kiểm soát chi tiêu trong tháng.
- **70%** cho rằng việc quản lý chi tiêu giúp họ tiết kiệm và sử dụng tiền hiệu quả hơn.
- **85%** mong muốn có một ứng dụng đơn giản, dễ sử dụng để ghi lại các khoản thu chi và xem báo cáo tổng quan tài chính.
- **60%** từng thử sử dụng một ứng dụng tài chính nhưng từ bỏ vì giao diện phức tạp hoặc không hỗ trợ tiếng Việt.

Kết luận:

Từ khảo sát, có thể thấy rằng người dùng hiện nay có nhu cầu cao về một công cụ hỗ trợ quản lý chi tiêu cá nhân. Tuy nhiên, để được sử dụng lâu dài, ứng dụng cần có: giao diện thân thiện, dễ thao tác; tính năng cơ bản nhưng đầy đủ: ghi chép, phân loại, báo cáo; hỗ trợ ngôn ngữ tiếng Việt, có thể tùy chỉnh danh mục chi tiêu.

1.4. Một số công cụ xây dựng ứng dụng di động

1.4.1. Tổng quan về lập trình di động

Hiện nay, trong lĩnh vực phát triển ứng dụng di động, có ba hướng tiếp cận chính được sử dụng phổ biến là **Native App**, **Web App** và **Hybrid App**. Mỗi hướng đều có đặc điểm riêng, phù hợp với các mục tiêu, nhu cầu và nguồn lực khác nhau trong quá trình phát triển sản phẩm.

1.4.1.1. Hybrid App

Hybrid App [1] (ứng dụng lai) là một mô hình ứng dụng di động kết hợp giữa hai hướng phát triển: Mobile Web App và Native App. Cốt lõi của Hybrid App được xây dựng bằng các công nghệ web phổ biến như HTML, CSS và JavaScript, sau đó được "gói" lại bên trong một native container (trình bao bọc gốc). Nhờ đó, ứng dụng có thể được triển khai và phân phối thông qua các nền tảng cửa hàng ứng dụng như Google Play Store hoặc Apple App Store, giống như các ứng dụng gốc thông thường.

Một trong những ưu điểm lớn nhất của Hybrid App là khả năng phát triển nhanh chóng và hiệu quả. Lập trình viên chỉ cần viết mã một lần và có thể sử dụng cho nhiều nền tảng khác nhau, giúp tiết kiệm đáng kể thời gian và chi phí so với việc phát triển riêng biệt cho từng hệ điều hành. Việc sử dụng các công nghệ web quen thuộc như HTML, CSS và JavaScript cũng giúp giảm bớt rào cản về mặt kỹ năng cho các nhà phát triển. Bên cạnh đó, Hybrid App vẫn có khả năng tận dụng một phần tài nguyên của hệ điều hành, như truy cập camera, GPS, bộ nhớ... thông qua các plugin hỗ trợ. Một số nền tảng Hybrid hiện đại còn hỗ trợ chế độ hoạt động offline, giúp tăng tính linh hoạt và cải thiện trải nghiệm người dùng.

Tuy nhiên, Hybrid App cũng tồn tại một số hạn chế nhất định. Do mã nguồn web phải được biên dịch hoặc thông dịch qua một lớp trung gian để tương tác với hệ thống, nên hiệu năng thường không thể sánh được với ứng

dụng native. Đặc biệt là đối với những ứng dụng yêu cầu xử lý đồ họa cao, tốc độ phản hồi nhanh hoặc tương tác phức tạp. Ngoài ra, việc debug và tối ưu hóa trong quá trình phát triển Hybrid App cũng gặp khó khăn hơn. Các framework Hybrid thường dịch mã JavaScript sang mã native ở dạng trung gian, khiến lập trình viên khó theo dõi quá trình chuyển đổi, từ đó việc phát hiện và sửa lỗi trở nên phức tạp và mất thời gian hơn.

Mặc dù còn một số mặt hạn chế, nhưng với ưu điểm về tốc độ phát triển, khả năng đa nền tảng và chi phí thấp, Hybrid App vẫn là một lựa chọn phù hợp trong nhiều trường hợp, đặc biệt với các ứng dụng có quy mô vừa và nhỏ, hoặc ứng dụng phục vụ nhu cầu cơ bản và phổ biến.

1.4.1.2. Native App

Native App [1] là loại ứng dụng được phát triển riêng biệt cho từng nền tảng di động cụ thể như Android, iOS hoặc Windows Phone, sử dụng các ngôn ngữ lập trình và công cụ tương thích với từng hệ điều hành. Ví dụ, các ứng dụng Android thường được xây dựng bằng Java hoặc Kotlin, trong khi ứng dụng iOS được viết bằng Objective-C hoặc Swift và phát triển thông qua môi trường Xcode trên hệ điều hành macOS. Mỗi Native App chỉ có thể chạy trên nền tảng mà nó được thiết kế cho, và không thể sử dụng trực tiếp trên các nền tảng khác nếu không được viết lại hoàn toàn.

Ưu điểm lớn nhất của Native App chính là khả năng tận dụng tối đa các tính năng phần cứng và phần mềm của thiết bị như GPS, camera, cảm biến, microphone, bộ nhớ trong... nhờ vào quyền truy cập sâu vào hệ thống thông qua các API gốc. Do được biên dịch trực tiếp thành mã máy và chạy ngay trên nền tảng mục tiêu, Native App thường có hiệu năng cao, tốc độ xử lý nhanh, độ ổn định tốt và trải nghiệm người dùng mượt mà. Bên cạnh đó, Native App vẫn có thể hoạt động ở chế độ offline, không phụ thuộc hoàn toàn vào kết nối Internet, điều này rất hữu ích trong các tình huống người dùng không có truy cập mạng.

Tuy nhiên, Native App cũng tồn tại một số hạn chế đáng kể. Do mỗi ứng dụng chỉ tương thích với một hệ điều hành duy nhất, nên nếu muốn triển khai trên nhiều nền tảng, lập trình viên phải phát triển riêng biệt từng phiên bản ứng dụng cho từng nền tảng, dẫn đến việc tăng chi phí phát triển, tốn thời gian bảo trì và gây khó khăn trong việc đồng bộ hóa tính năng giữa các nền tảng. Ngoài ra, quá trình phát triển Native App đòi hỏi phải sử dụng các công cụ chuyên biệt như Android Studio cho Android và Xcode cho iOS, điều này yêu cầu lập trình viên có kỹ năng đa dạng và đôi khi cần sử dụng các hệ điều hành cụ thể (ví dụ: phải có máy Mac để lập trình iOS).

Tóm lại, Native App là lựa chọn lý tưởng cho các ứng dụng yêu cầu hiệu suất cao, tương tác phức tạp hoặc trải nghiệm người dùng tối ưu. Tuy nhiên, nó cũng đi kèm với chi phí và công sức phát triển lớn hơn, đặc biệt khi cần hỗ trợ đa nền tảng.

1.4.1.3. Web App

Web app [1] (ứng dụng web) là một dạng ứng dụng được phát triển và vận hành thông qua trình duyệt web của thiết bị di động, sử dụng các công nghệ phổ biến như HTML, CSS, JavaScript và các thư viện hỗ trợ như Bootstrap, jQuery hoặc các framework hiện đại như Angular, React, Vue.js. Khác với các ứng dụng di động thông thường, Web App không cần cài đặt trực tiếp lên thiết bị mà người dùng có thể truy cập và sử dụng thông qua địa chỉ URL như một trang web thông thường. Thông thường, Web App được phát triển dựa trên nền tảng một website đã có sẵn, sau đó được tối ưu hóa giao diện và chức năng để phù hợp hơn với thiết bị di động.

Một trong những ưu điểm nổi bật của Web App là khả năng tương thích cao, có thể chạy trên hầu hết các trình duyệt hiện đại của thiết bị di động, miễn là thiết bị đó hỗ trợ HTML và JavaScript. Điều này giúp giảm đáng kể chi phí và thời gian phát triển, do chỉ cần duy trì một mã nguồn duy nhất cho mọi thiết bị, không cần phân tách riêng biệt như Native App. Ngoài ra, Web App không

yêu cầu cài đặt, người dùng chỉ cần truy cập vào trang web là có thể sử dụng ngay. Việc cập nhật và bảo trì cũng trở nên đơn giản hơn vì các thay đổi được triển khai trực tiếp lên máy chủ, người dùng không cần thực hiện thao tác cập nhật từ các cửa hàng ứng dụng.

Tuy nhiên, Web App cũng tồn tại nhiều hạn chế, đặc biệt là về hiệu năng và khả năng tích hợp hệ thống. Do hoạt động thông qua trình duyệt, Web App thường không đạt được hiệu năng cao như Native App, nhất là trong các tác vụ xử lý phức tạp hoặc yêu cầu độ phản hồi nhanh. Một điểm yếu nữa là Web App phụ thuộc hoàn toàn vào kết nối Internet, và hầu như không thể sử dụng khi thiết bị offline. Ngoài ra, Web App không có quyền truy cập vào nhiều tính năng phần cứng của thiết bị như thông báo đẩy (push notification), camera, cảm biến chuyển động, định vị GPS, v.v. Điều này giới hạn đáng kể trải nghiệm người dùng. Với các thiết bị đời cũ hoặc trình duyệt không được cập nhật, Web App cũng có nguy cơ gặp lỗi giao diện, lỗi tương thích hoặc các đoạn mã JavaScript không thể thực thi đúng cách.

Tổng kết lại, Web App là lựa chọn phù hợp cho các ứng dụng có tính chất đơn giản, không yêu cầu tích hợp sâu vào phần cứng, và ưu tiên khả năng truy cập nhanh, chi phí thấp. Tuy nhiên, với các yêu cầu cao về hiệu suất và trải nghiệm người dùng, Web App khó có thể thay thế được hoàn toàn các ứng dụng di động truyền thống như Native App hoặc Hybrid App.

1.4.2. Giới thiệu công cụ lập trình Visual Studio Code

Visual Studio Code [2] (hay gọi tắt là VSCode) là một trình biên tập mã nguồn mở, đa nền tảng và miễn phí, được phát triển bởi Microsoft. Với thiết kế nhẹ, giao diện trực quan và khả năng mở rộng mạnh mẽ, VSCode đã nhanh chóng trở thành một trong những công cụ lập trình phổ biến nhất hiện nay, được sử dụng rộng rãi trong cộng đồng phát triển phần mềm trên toàn thế giới.

Dưới đây là một số tính năng và công cụ hỗ trợ của VSCode:

- Đa nền tảng: VSCode có sẵn cho Windows, macOS và Linux, giúp cho các nhà phát triển có thể sử dụng trình biên tập này trên nhiều hệ điều hành khác nhau.

- Hỗ trợ nhiều ngôn ngữ lập trình: VSCode hỗ trợ nhiều ngôn ngữ lập trình như C#, C++, Java, Python, JavaScript, TypeScript đến các ngôn ngữ chuyên biệt hơn, nhờ vào hệ thống mở rộng (extensions) phong phú.

- Công cụ IntelliSense: VSCode có tính năng IntelliSense – công cụ gợi ý mã thông minh, cung cấp khả năng tự động hoàn thành cú pháp, hiển thị tài liệu API, cũng như phát hiện và đề xuất sửa lỗi trong quá trình viết mã. Điều này giúp giảm thiểu lỗi cú pháp và tiết kiệm thời gian cho lập trình viên.

- Hỗ trợ Git: VSCode tích hợp sẵn với Git, cho phép lập trình viên theo dõi lịch sử mã nguồn, thực hiện commit, push, pull hoặc giải quyết xung đột mã ngay trong giao diện biên tập mà không cần sử dụng công cụ dòng lệnh bên ngoài. Tính năng này đặc biệt hữu ích trong làm việc nhóm và các dự án lớn có tích hợp kiểm soát mã nguồn.

- Thư viện mở rộng (Extensions): VSCode có thể được mở rộng bằng cách cài đặt các extension, giúp cho việc phát triển phần mềm trở nên linh hoạt hơn. Người dùng có thể dễ dàng cài đặt các extension để hỗ trợ thêm nhiều ngôn ngữ lập trình, công cụ linting, kiểm thử, định dạng mã, quản lý cơ sở dữ liệu, Docker, DevOps, và nhiều công cụ tích hợp khác. Điều này giúp VSCode dễ dàng được tùy chỉnh để phù hợp với nhu cầu và thói quen làm việc của từng cá nhân hoặc nhóm phát triển.

- Tích hợp bộ nhớ đệm (Workspace): VSCode cho phép lưu trữ toàn bộ thông tin dự án, cấu hình, cài đặt môi trường, giúp người dùng quản lý và truy cập lại công việc một cách nhanh chóng và hiệu quả.

Tóm lại, VSCode là một công cụ biên tập mã hiện đại, mạnh mẽ và linh hoạt, thích hợp với nhiều đối tượng từ người mới học lập trình đến các lập trình

viên chuyên nghiệp. Với khả năng mở rộng cao và cộng đồng phát triển lớn mạnh, VSCode ngày càng khẳng định vai trò quan trọng trong hệ sinh thái công cụ phát triển phần mềm hiện đại.

1.4.3. Ngôn ngữ lập trình Dart

Dart [3] là một ngôn ngữ lập trình được phát triển bởi Google vào năm 2011, đặc biệt được sử dụng để phát triển ứng dụng di động và web. Dart được thiết kế để cải thiện các vấn đề của JavaScript và cung cấp nhiều tính năng hiện đại hơn. Dart hỗ trợ kiểu tĩnh và kiểu động, cú pháp đơn giản và dễ đọc, hỗ trợ lập trình hướng đối tượng, hàm bậc cao, và bộ thu gom rác tự động.

Dart cũng hỗ trợ viết code đa nền tảng, cho phép phát triển ứng dụng di động cho cả Android và iOS, cũng như ứng dụng web. Flutter, một framework phát triển ứng dụng di động đa nền tảng, cũng được phát triển bởi Google và sử dụng Dart làm ngôn ngữ lập trình chính. Flutter hỗ trợ phát triển ứng dụng di động với tốc độ và hiệu suất cao, và được đánh giá là một trong những framework phát triển ứng dụng di động tốt nhất hiện nay.

➢ Ưu điểm:

- Tính năng hiện đại: Dart được thiết kế để cải thiện các vấn đề của JavaScript và cung cấp nhiều tính năng hiện đại hơn, bao gồm hỗ trợ kiểu tĩnh và kiểu động, hàm bậc cao, và bộ thu gom rác tự động.

- Đơn giản và dễ đọc: Cú pháp của Dart đơn giản và dễ đọc, giúp cho các nhà phát triển dễ dàng hiểu và sử dụng ngôn ngữ này.

- Lập trình hướng đối tượng: Dart hỗ trợ lập trình hướng đối tượng, cho phép phát triển ứng dụng với cấu trúc logic rõ ràng và dễ bảo trì.

- Viết code đa nền tảng: Dart hỗ trợ viết code đa nền tảng, cho phép phát triển ứng dụng di động cho cả Android và iOS, cũng như ứng dụng web. Điều này đặc biệt quan trọng đối với các dự án phát triển ứng dụng di động hiện nay,

vì nó giúp giảm chi phí và thời gian phát triển, đồng thời tăng tính linh hoạt trong việc mở rộng ứng dụng.

- Flutter: Flutter - một framework phát triển ứng dụng di động đa nền tảng của Google, được sử dụng Dart làm ngôn ngữ lập trình chính. Nhờ vào sự kết hợp này, Flutter cung cấp khả năng phát triển ứng dụng với hiệu suất cao, giao diện mượt mà và tốc độ xử lý nhanh, giúp các lập trình viên tạo ra những ứng dụng di động chất lượng, đồng thời tiết kiệm thời gian và công sức trong quá trình phát triển, và được đánh giá là một trong những framework phát triển ứng dụng di động tốt nhất hiện nay.

Với những ưu điểm trên, Dart đang trở thành một ngôn ngữ lập trình ngày càng phổ biến và được nhiều nhà phát triển tin tưởng sử dụng cho các dự án phát triển ứng dụng di động và web. Với các tính năng vượt trội như cú pháp đơn giản, hỗ trợ lập trình hướng đối tượng, và khả năng phát triển đa nền tảng, Dart đang ngày càng được ưa chuộng và trở thành một lựa chọn phổ biến trong cộng đồng lập trình, đặc biệt là khi kết hợp với framework Flutter.

1.4.4. Một số framework trong lập trình Flutter

1.4.4.1. Flutter SDK

Flutter SDK [4] là một bộ công cụ phát triển phần mềm (Software Development Kit) được phát triển bởi Google, dùng để xây dựng các ứng dụng di động đa nền tảng với hiệu suất cao và giao diện đẹp mắt. Flutter SDK sử dụng ngôn ngữ lập trình Dart, cho phép các nhà phát triển viết một lần và triển khai ứng dụng trên cả Android và iOS, giúp tiết kiệm thời gian và chi phí phát triển.

Flutter SDK cung cấp một hệ thống widget phong phú được thiết kế theo hai phong cách chính là Material Design (theo ngôn ngữ thiết kế của Google) và Cupertino (mô phỏng giao diện của iOS). Các widget này được thiết kế để

tùy biến cao, giúp lập trình viên dễ dàng tạo ra các giao diện người dùng hiện đại, mượt mà và nhất quán trên nhiều thiết bị khác nhau.

Ngoài widget, Flutter SDK còn tích hợp nhiều tính năng mạnh mẽ hỗ trợ quá trình phát triển ứng dụng như:

- Animation: Hỗ trợ xây dựng các hiệu ứng chuyển động và tương tác mượt mà, góp phần nâng cao trải nghiệm người dùng.
- Routing & Navigation: Cung cấp hệ thống định tuyến linh hoạt, cho phép quản lý và điều hướng giữa các màn hình trong ứng dụng một cách hiệu quả.
- Hot Reload: Tính năng nổi bật giúp lập trình viên có thể xem kết quả thay đổi ngay lập tức mà không cần khởi động lại toàn bộ ứng dụng.
- Cross-platform Compilation: Flutter SDK biên dịch mã nguồn Dart thành mã máy gốc (native code), giúp tối ưu hóa hiệu suất và giảm độ trễ khi chạy ứng dụng.

Flutter không chỉ hỗ trợ phát triển ứng dụng di động mà còn đang mở rộng sang các nền tảng khác như web, desktop (Windows, macOS, Linux) và thậm chí cả nhúng (embedded devices), giúp xây dựng hệ sinh thái ứng dụng đa nền tảng đồng bộ và linh hoạt.

Tóm lại, Flutter SDK là một giải pháp toàn diện dành cho việc phát triển ứng dụng đa nền tảng, với đầy đủ các công cụ và thư viện cần thiết để tạo ra các sản phẩm chất lượng cao, dễ bảo trì và tối ưu hóa trải nghiệm người dùng. Với cộng đồng phát triển ngày càng lớn mạnh và sự hỗ trợ từ Google, Flutter đang ngày càng khẳng định vị thế của mình trong lĩnh vực lập trình ứng dụng hiện đại.

1.4.4.2. AngularDart

AngularDart [5] là một framework phát triển ứng dụng web do Google phát triển, sử dụng Dart làm ngôn ngữ lập trình chính. Đây là một phiên bản của Angular – vốn là framework nổi tiếng cho phát triển frontend – nhưng được viết và tối ưu hóa hoàn toàn cho ngôn ngữ Dart. AngularDart cung cấp đầy đủ các tính năng cần thiết cho việc xây dựng các ứng dụng web hiện đại, có cấu trúc rõ ràng, dễ bảo trì và mở rộng.

Framework này hỗ trợ mô hình MVC (Model-View-Controller), cho phép tách biệt rõ ràng giữa dữ liệu, giao diện người dùng và logic điều khiển, giúp quản lý và phát triển ứng dụng hiệu quả hơn. Một tính năng nổi bật khác của AngularDart là dependency injection (tiêm phụ thuộc), cho phép dễ dàng quản lý các phụ thuộc trong ứng dụng, từ đó nâng cao khả năng tái sử dụng và kiểm thử mã nguồn.

AngularDart cũng hỗ trợ routing – điều hướng giữa các trang trong ứng dụng web – một cách linh hoạt và dễ cấu hình. Bên cạnh đó, framework này có thể tích hợp dễ dàng với các thư viện và công cụ Dart khác, cho phép mở rộng chức năng và xây dựng các ứng dụng web phong phú, phản hồi nhanh và tối ưu hóa hiệu suất.

Mặc dù AngularDart không phổ biến rộng rãi như các framework JavaScript như React hay Angular (TypeScript), nhưng nhờ vào sự hỗ trợ từ Google và khả năng tích hợp sâu với Dart, nó vẫn là một lựa chọn đáng cân nhắc đối với những dự án web cần hiệu suất cao, quy mô lớn, và mong muốn đồng bộ hệ sinh thái lập trình với Dart, đặc biệt là khi kết hợp với Flutter trong các giải pháp đa nền tảng.

1.4.4.3. StageXL

StageXL [6] là một framework phát triển trò chơi web sử dụng Dart làm ngôn ngữ lập trình chính. Được thiết kế dành riêng cho việc xây dựng các trò

chơi và ứng dụng đồ họa tương tác trên nền tảng web, StageXL cung cấp một bộ công cụ mạnh mẽ và linh hoạt giúp các lập trình viên dễ dàng tạo ra các sản phẩm có hiệu suất cao và giao diện sống động.

StageXL hỗ trợ nhiều tính năng quan trọng trong phát triển game như sprite (đối tượng đồ họa động), animation (hiệu ứng chuyển động), sound (âm thanh), và các hiệu ứng đặc biệt khác. Framework này được xây dựng dựa trên HTML5 Canvas, giúp tối ưu hóa hiệu suất khi chạy trên các trình duyệt hiện đại và đảm bảo trải nghiệm mượt mà cho người chơi.

Một trong những điểm mạnh của StageXL là khả năng phát triển trò chơi đa nền tảng web. Với việc sử dụng Dart – một ngôn ngữ mạnh mẽ, hướng đối tượng và dễ học – các nhà phát triển có thể viết mã một lần và triển khai dễ dàng trên nhiều trình duyệt khác nhau, mà không cần lo lắng nhiều về khả năng tương thích hay hiệu suất xử lý.

StageXL còn đi kèm với các tính năng hỗ trợ như quản lý khung hình, xử lý tương tác người dùng (mouse, touch), hệ thống hiển thị phân cấp (display tree), và tải tài nguyên (resource loading), giúp việc phát triển game trở nên đơn giản, có tổ chức và dễ bảo trì.

Tóm lại, StageXL là một framework hữu ích và mạnh mẽ dành cho các lập trình viên Dart đang muốn phát triển trò chơi hoặc ứng dụng đồ họa động trên web. Với khả năng hỗ trợ đa nền tảng, hiệu năng cao và tích hợp đầy đủ các công cụ cần thiết, StageXL là một lựa chọn lý tưởng cho việc xây dựng các trò chơi trình duyệt hấp dẫn và chuyên nghiệp.

1.4.5. SQLite Database

SQLite [7] là một hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) nhỏ gọn, được thiết kế để tích hợp vào các ứng dụng. Nó là một thư viện mã nguồn mở, được viết bằng ngôn ngữ C, và cung cấp các tính năng quản lý cơ sở dữ liệu như các hệ quản trị cơ sở dữ liệu lớn hơn như MySQL, PostgreSQL hay Oracle.

SQLite được phát triển bởi D. Richard Hipp vào năm 2000 và đã trở thành một trong những hệ quản trị cơ sở dữ liệu phổ biến nhất trên thế giới. SQLite được tích hợp sẵn trên hầu hết các hệ điều hành phổ biến như Windows, macOS và Linux, và cũng được hỗ trợ trên các nền tảng di động như Android và iOS.

- Cấu trúc của SQLite

SQLite là một thư viện mã nguồn mở và được viết bằng ngôn ngữ C. Nó bao gồm một bộ máy truy vấn SQL, một bộ máy quản lý tệp và các thành phần bổ sung khác. Các đối tượng trong SQLite bao gồm bảng, cột, khóa chính và các ràng buộc.

- Bảng: Là một đối tượng trong SQLite để lưu trữ dữ liệu. Bảng bao gồm các cột và hàng, trong đó mỗi cột đại diện cho một kiểu dữ liệu và mỗi hàng đại diện cho một bản ghi.
- Cột: Là một đối tượng trong SQLite để đại diện cho một kiểu dữ liệu trong bảng. Các kiểu dữ liệu có thể là INTEGER, REAL, TEXT hoặc BLOB.
- Khóa chính: Là một cột hoặc tập hợp các cột trong bảng, được sử dụng để định danh duy nhất cho mỗi hàng trong bảng.
- Ràng buộc: Là các quy tắc được áp dụng trên bảng để đảm bảo tính toàn vẹn của dữ liệu. Ví dụ, ràng buộc NOT NULL sẽ đảm bảo rằng giá trị của cột không được phép là null.

- Cách thức hoạt động của SQLite

SQLite hoạt động như một thư viện được tích hợp vào các ứng dụng. Khi một ứng dụng sử dụng SQLite, nó sẽ tạo ra một tệp tin cơ sở dữ liệu, trong đó sẽ lưu trữ các bảng, cột và các bản ghi. Ứng dụng có thể truy vấn và thay đổi dữ liệu trong cơ sở dữ liệu này bằng cách sử dụng các lệnh SQL.

SQLite hỗ trợ các lệnh SQL như SELECT, INSERT, UPDATE và DELETE để truy vấn và thay đổi dữ liệu trong cơ sở dữ liệu. Nó cũng hỗ trợ

các câu lệnh để tạo bảng, chỉnh sửa bảng và xóa bảng. SQLite cũng có thể thực hiện các hoạt động như tạo chỉ mục, ràng buộc và trigger.

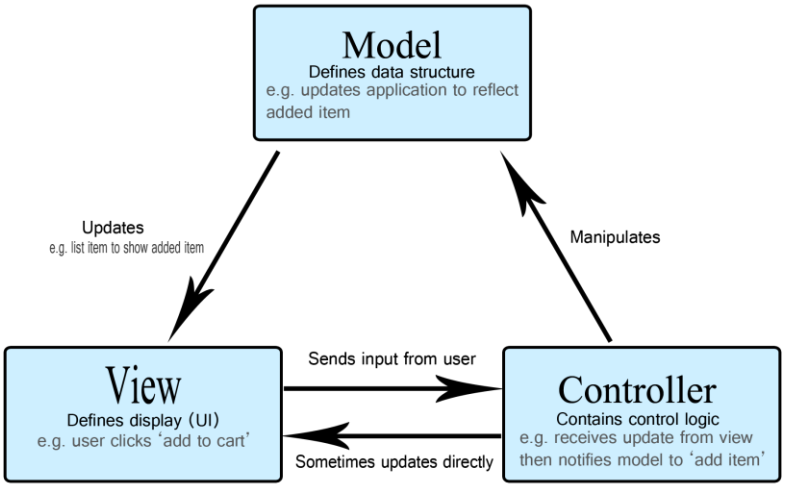
Trong ứng dụng của chúng ta, chúng ta sẽ sử dụng các package hỗ trợ của Flutter như sqflite và path_provider để tương tác với cơ sở dữ liệu SQLite. Sqflite là một package hỗ trợ cho phép tương tác với cơ sở dữ liệu SQLite trong Flutter, trong khi path_provider là một package cho phép truy cập vào các thư mục và tệp tin trong hệ thống tệp của thiết bị.

Với việc sử dụng cơ sở dữ liệu SQLite, chúng ta có thể lưu trữ thông tin và dữ liệu của người dùng một cách hiệu quả và tiện lợi. Ví dụ, chúng ta có thể lưu trữ thông tin về các khoản chi tiêu của người dùng và tạo báo cáo thống kê chi tiêu theo tháng, quý, năm, v.v. từ các dữ liệu được lưu trữ trong cơ sở dữ liệu SQLite.

- Ưu điểm của SQLite Database:
  - Kích thước nhỏ gọn: SQLite có dung lượng nhỏ gọn và khả năng tích hợp cao, giúp dễ dàng tích hợp vào các ứng dụng.
  - Không cần server: SQLite không yêu cầu một server đặc biệt để chạy, vì vậy nó rất tiện lợi cho các ứng dụng đơn giản.
  - Tốc độ truy vấn nhanh: SQLite có tốc độ truy vấn nhanh và khả năng xử lý dữ liệu tốt.
  - Hỗ trợ nhiều ngôn ngữ lập trình: SQLite được hỗ trợ bởi nhiều ngôn ngữ lập trình như C/C++, Python, Java, PHP, v.v.

1.4.6. Tổng quan về mô hình MVC

Mô hình MVC [8] (Model-View-Controller) là một mô hình thiết kế phần mềm được sử dụng để phát triển các ứng dụng có giao diện người dùng. Mô hình này bao gồm ba thành phần chính: Model, View và Controller.



Hình 1.1: Mô hình MVC

- *Model*

Model là thành phần chịu trách nhiệm xử lý dữ liệu. Nó đại diện cho các đối tượng dữ liệu và quy trình xử lý dữ liệu. Model không biết gì về giao diện người dùng, nó chỉ tập trung vào việc xử lý dữ liệu.

- *View*

View là thành phần chịu trách nhiệm hiển thị dữ liệu và tương tác với người dùng. View đại diện cho giao diện người dùng, bao gồm các thành phần như các nút bấm, trường nhập liệu và các thành phần khác. View không biết gì về dữ liệu và quy trình xử lý, nó chỉ tập trung vào việc hiển thị và tương tác với người dùng.

- *Controller*

Controller là thành phần chịu trách nhiệm điều khiển quá trình tương tác giữa Model và View. Controller đóng vai trò là trung gian giữa Model và View, nó nhận lệnh từ View, xử lý dữ liệu từ Model và cập nhật lại View để hiển thị

kết quả cho người dùng. Controller là thành phần duy nhất có thể tương tác với cả Model và View.

Mô hình MVC giúp tách biệt logic xử lý dữ liệu và giao diện người dùng, giúp cho việc phát triển và bảo trì ứng dụng trở nên dễ dàng hơn. Nó cũng giúp cho việc thay đổi giao diện người dùng hoặc cơ sở dữ liệu không ảnh hưởng đến logic xử lý dữ liệu và ngược lại.

- *Một ứng dụng thực tế có thể được thiết kế theo mô hình MVC*

- Model: Chứa các đối tượng dữ liệu như thông tin người dùng, thông tin sản phẩm, danh sách đơn hàng, v.v. Nó cũng chứa các phương thức để truy xuất và xử lý dữ liệu.

- View: Chứa các màn hình giao diện người dùng để hiển thị thông tin và tương tác với người dùng, ví dụ như màn hình đăng nhập, màn hình danh sách sản phẩm, màn hình đặt hàng, v.v.

- Controller: Chứa các phương thức để điều khiển quá trình tương tác giữa Model và View, ví dụ như phương thức để xử lý đăng nhập, phương thức để lấy danh sách sản phẩm, phương thức để đặt hàng, v.v.

Với việc sử dụng mô hình MVC, việc phát triển và bảo trì ứng dụng sẽ trở nên dễ dàng hơn và có thể được thực hiện bởi các nhóm phát triển khác nhau độc lập với nhau.

**** Ưu điểm***

+ Thực hiện Unit testing bây giờ sẽ rất dễ dàng, vì bạn thực sự không phụ thuộc vào view

+ MVC sẽ tạo sự tương tác hiệu quả giữa designer và developer

+ Tăng khả năng sử dụng lại các thành phần hay việc thay đổi giao diện chương trình mà không cần phải viết lại code quá nhiều

+ Phát triển ứng dụng nhanh, đơn giản, dễ nâng cấp, bảo trì…

**** Nhược điểm***

+ Khả năng duy trì khi view có thể gán cả biến và biểu thức, các logic không liên quan sẽ tăng dần theo thời gian, ảnh hưởng đến việc thêm code vào XML

+ Đối với dự án nhỏ việc áp dụng mô hình MVVM gây cồng kềnh, tốn thời gian trong quá trình phát triển. Tốn thời gian trung chuyển dữ liệu của các thành phần

+ Đối với dự án lớn hơn, nó gây khó khăn và mất thời gian để thiết kế các ViewModel

+ Việc liên kết dữ liệu cho tất cả các thành phần gây khó khăn trong việc debug khi cơ sở dữ liệu phức tạp

CHƯƠNG 2. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

2.1. Giới thiệu về hệ thống

Đồ án tốt nghiệp với đề tài "Ứng dụng quản lý chi tiêu bằng Flutter" hướng đến việc xây dựng một hệ thống ứng dụng di động đa nền tảng, hỗ trợ người dùng trong việc quản lý tài chính cá nhân một cách khoa học, trực quan và hiệu quả. Ứng dụng được phát triển trên nền tảng Flutter, một framework hiện đại và mạnh mẽ do Google phát triển, cho phép xây dựng ứng dụng hoạt động mượt mà trên cả Android và iOS chỉ với một bộ mã nguồn duy nhất.

Hệ thống sẽ cho phép người dùng ghi lại các khoản chi tiêu hàng ngày, phân loại theo từng danh mục cụ thể như: thực phẩm, đi lại, hóa đơn điện thoại, mua sắm, giải trí, giáo dục, v.v. Người dùng có thể dễ dàng theo dõi và thống kê chi tiêu của mình theo các tiêu chí như khoảng thời gian (ngày, tuần, tháng), loại danh mục, hoặc mức độ sử dụng tài chính (số tiền đã chi, còn lại, vượt ngân sách,...). Dữ liệu sẽ được hiển thị qua các biểu đồ trực quan, giúp người dùng dễ dàng đánh giá thói quen chi tiêu và đưa ra các quyết định tài chính hợp lý hơn.

Bên cạnh đó, ứng dụng còn tích hợp tính năng đặt mục tiêu tiết kiệm, cho phép người dùng định nghĩa khoản tiền mong muốn tiết kiệm trong một mốc thời gian cụ thể, đồng thời theo dõi tiến độ đạt mục tiêu qua từng giai đoạn. Điều này không chỉ hỗ trợ kiểm soát chi tiêu mà còn khuyến khích người dùng hình thành thói quen tiết kiệm tài chính bền vững.

Ngoài ra, hệ thống cũng cung cấp tính năng nhắc nhở cho người dùng về các khoản chi tiêu cần phải thanh toán trong thời gian sắp tới, giúp họ không quên các khoản chi tiêu quan trọng. Người dùng có thể đặt lịch cho các khoản chi định kỳ như hóa đơn tiền điện, nước, học phí, nợ,… và hệ thống sẽ gửi thông báo đến người dùng để đảm bảo không bị quên hoặc trễ hạn thanh toán.

Hệ thống sẽ được xây dựng trên nền tảng Flutter, một framework phổ biến cho việc phát triển ứng dụng di động đa nền tảng. Hệ thống sẽ được thiết

kế với giao diện đơn giản, dễ sử dụng và thân thiện với người dùng, đơn giản nhưng hiện đại, tối ưu trải nghiệm trên cả điện thoại và máy tính bảng. Hệ thống đảm bảo hoạt động ổn định, hiệu suất cao, dễ dàng mở rộng và bảo trì trong tương lai.

Kết luận, đồ án không chỉ thể hiện khả năng áp dụng kiến thức lập trình di động đa nền tảng, mà còn mang tính thực tiễn cao, giải quyết một nhu cầu thiết yếu trong đời sống cá nhân, góp phần nâng cao nhận thức và kỹ năng quản lý tài chính cho người sử dụng.

2.2. Phân tích yêu cầu

2.2.1. Về hệ thống

Ứng dụng Quản lý chi tiêu hướng đến việc xây dựng một hệ thống hiện đại, tiện lợi và thân thiện với người dùng, trong đó yếu tố trải nghiệm người dùng (UX) được đặt lên hàng đầu. Giao diện ứng dụng được thiết kế theo phong cách tối giản, trực quan và dễ thao tác, giúp người dùng ở mọi lứa tuổi có thể sử dụng một cách dễ dàng mà không cần kiến thức công nghệ phức tạp.

Ngoài việc tối ưu về mặt giao diện, ứng dụng cũng tập trung vào việc tăng cường hiệu suất hoạt động, đảm bảo các thao tác như ghi chép chi tiêu, xem báo cáo, đặt mục tiêu hay nhận nhắc nhở đều được thực hiện nhanh chóng và mượt mà. Điều này giúp người dùng tiết kiệm thời gian trong việc theo dõi và kiểm soát tài chính cá nhân, từ đó hình thành thói quen quản lý chi tiêu hiệu quả hơn.

Với những yếu tố trên, ứng dụng không chỉ đơn thuần là một công cụ hỗ trợ ghi chú tài chính, mà còn là một trợ lý tài chính cá nhân tin cậy, đồng hành cùng người dùng trong quá trình xây dựng và duy trì cuộc sống tài chính ổn định, thông minh và bền vững.

2.2.2. Về người sử dụng

Ứng dụng cho phép người dùng dễ dàng nắm bắt và quản lý các giao dịch chi tiêu hằng ngày một cách rõ ràng và có hệ thống. Mỗi khoản chi tiêu được ghi lại chi tiết theo từng danh mục cụ thể, kèm theo các thông tin như ngày giờ giao dịch, số tiền, ví sử dụng và ghi chú nếu cần. Điều này giúp người dùng có cái nhìn tổng quan về tình hình tài chính cá nhân theo từng ngày, tuần hoặc tháng.

Bên cạnh đó, người dùng còn có thể quản lý danh sách các ví tiền như ví tiền mặt, tài khoản ngân hàng, ví điện tử... để theo dõi số dư và các biến động tài chính một cách chính xác. Ứng dụng cũng hỗ trợ theo dõi các sự kiện liên quan đến việc sử dụng tiền, chẳng hạn như chi tiêu cho du lịch, lễ cưới, sinh nhật hay các kế hoạch cá nhân khác.

Đặc biệt, hệ thống còn cung cấp tính năng lọc và tìm kiếm thông tin nâng cao, cho phép người dùng dễ dàng truy xuất dữ liệu theo các tiêu chí như khoảng thời gian, danh mục chi tiêu, ví tiền, số tiền... giúp việc phân tích và thống kê chi tiêu trở nên nhanh chóng và hiệu quả hơn.

2.2.3. Yêu cầu về chức năng

- Các chức năng chính của hệ thống:
  - Xem thông tin tổng quan
  - Quản lý giao dịch
  - Quản lý sự kiện
  - Quản lý nguồn tiền
  - Xem thông báo
- Yêu cầu phi chức năng:
  - Giao diện đồ họa dễ hình, dễ thao tác
  - Hoạt động của hệ thống ổn định
  - Độ tin cậy của ứng dụng

- Hiệu năng của ứng dụng

2.3. Thiết kế hệ thống

2.3.1. Mô hình hóa Usecase

2.3.1.1. Xác định các tác nhân

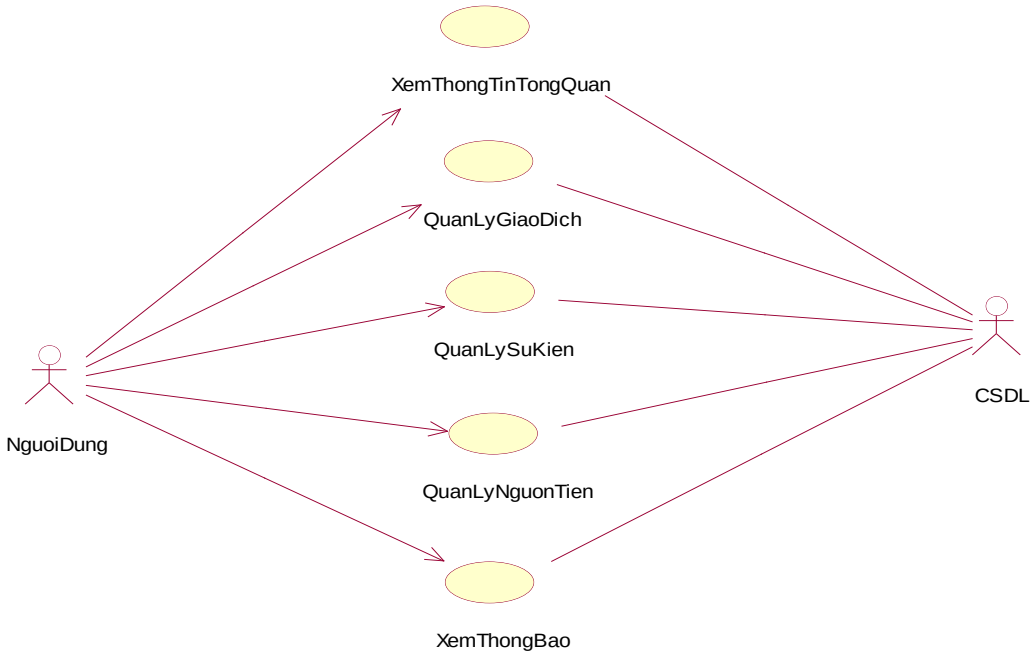
- Người dùng

2.3.1.2. Xác định các Usecase

- Xem thông tin tổng quan
- Quản lý giao dịch
- Quản lý sự kiện
- Quản lý nguồn tiền
- Xem thông báo

2.3.1.3. Biểu đồ Usecase

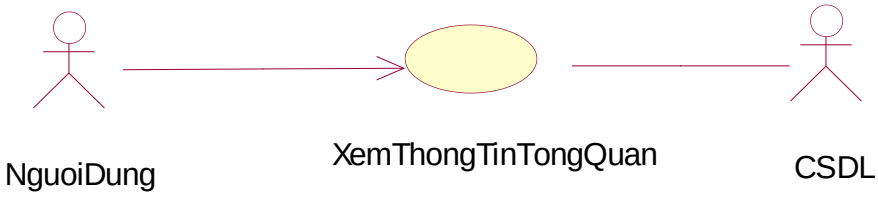
1. ***Biểu đồ Usecase chính***



Hình 2.1: Biểu đồ Usecase chính

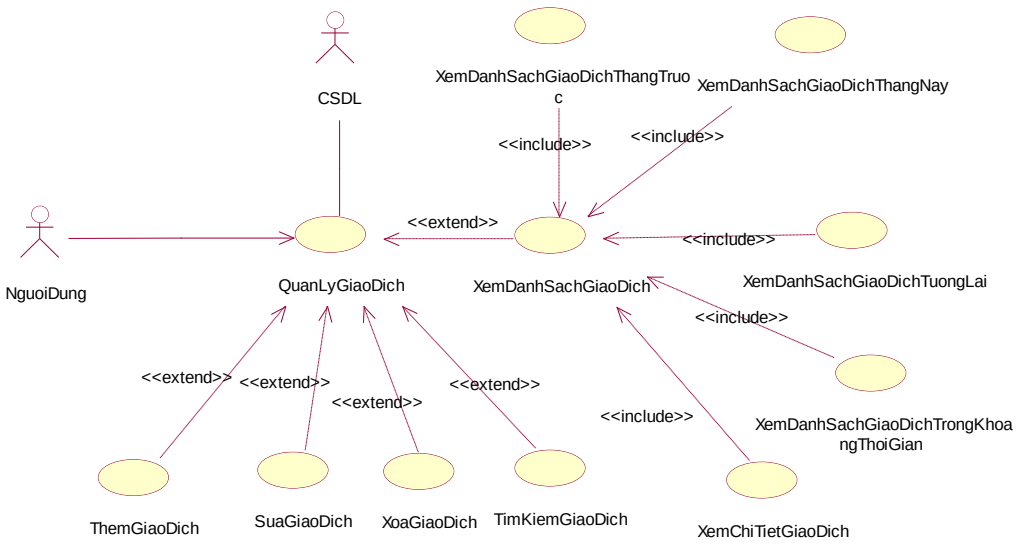
2. Biểu đồ Usecase thứ cấp

a. Phân rã Usecase <Xem Thông tin tổng quan>



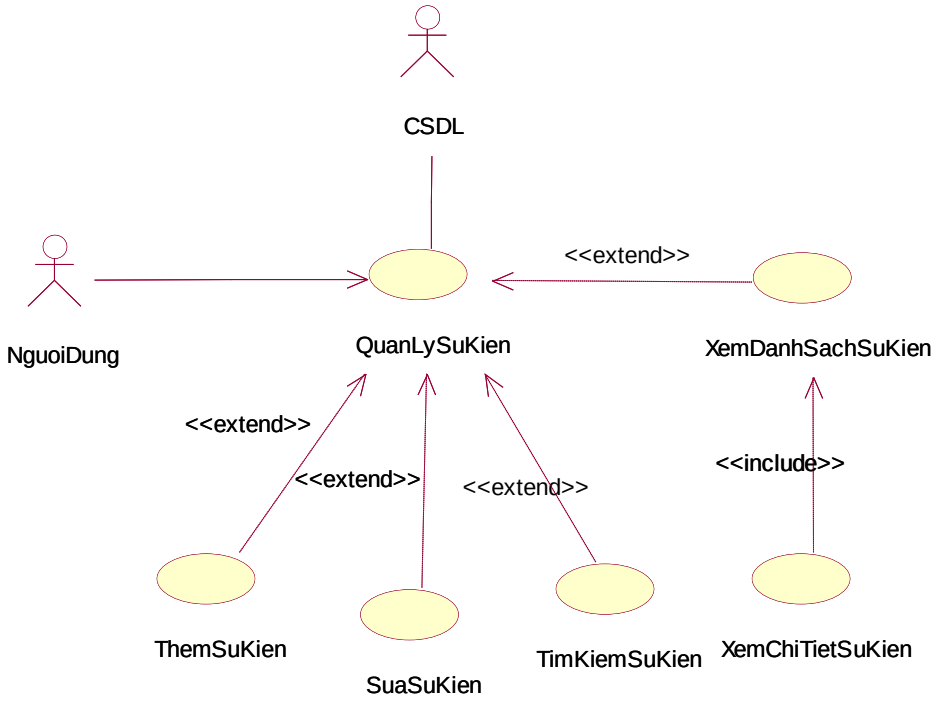
Hình 2.2: Phân rã Usecase Xem Thông tin tổng quan

b. Phân rã Usecase <Quản lý Giao Dịch>



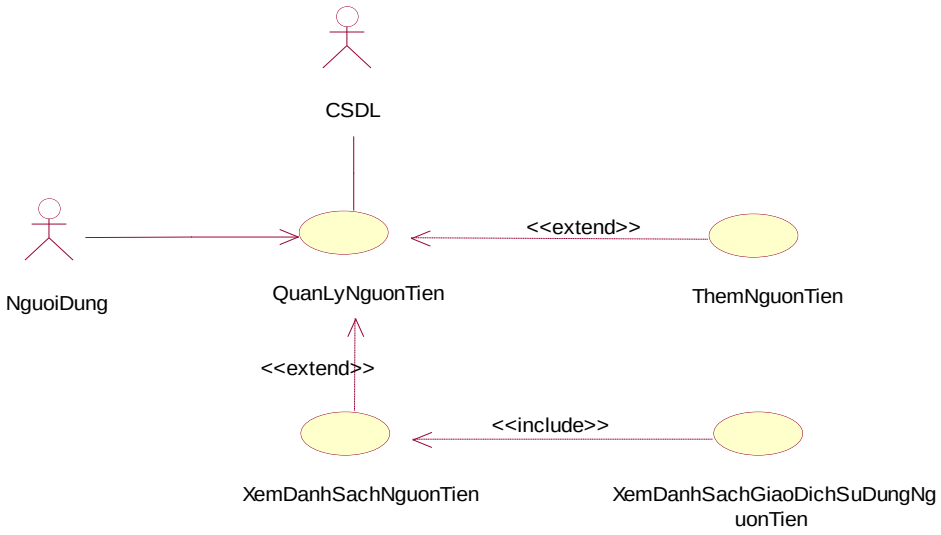
Hình 2.3: Phân rã Usecase Quản lý giao dịch

c. Phân rã Usecase <Quản Lý Sự kiện>



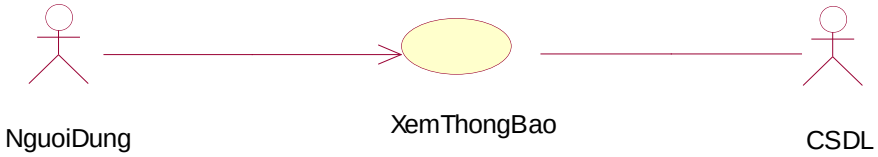
Hình 2.4: Phân rã Usecase Quản lý sự kiện

d. Phân rã Usecase <Quản lý Nguồn Tiền>



Hình 2.5: Phân rã Usecase Quản lý nguồn tiền

e. Phân rã Usecase <Xem Thông Báo>



Hình 2.6: Phân rã Usecase Xem thông báo

2.3.2. Mô tả chi tiết các Usecase

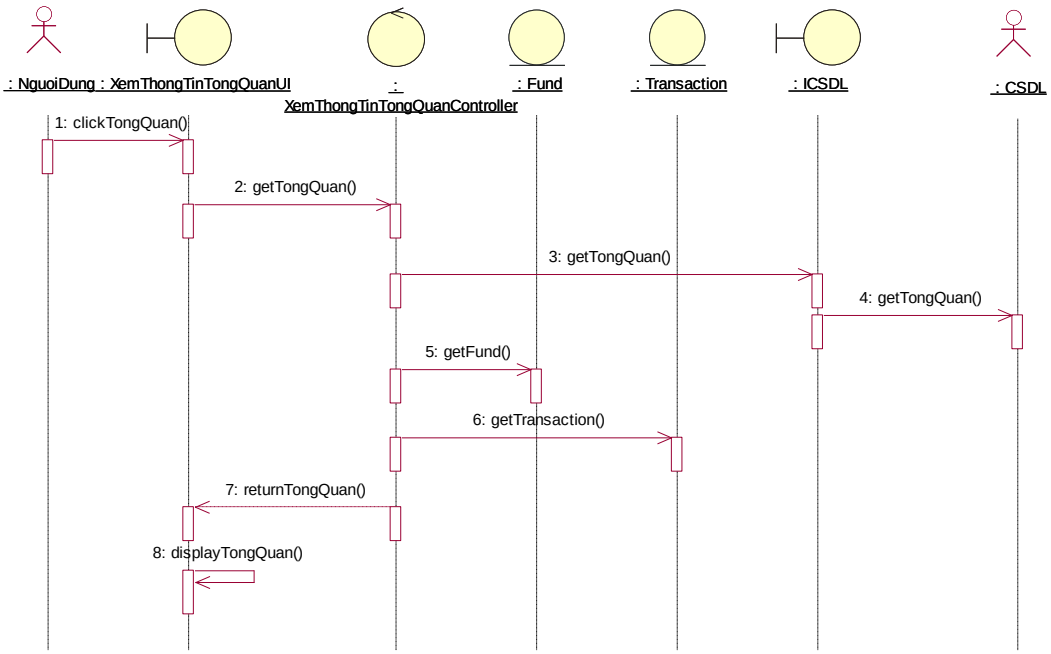
2.3.2.1. Mô tả Usecase <Xem thông tin tổng quan>

Bảng 2.1: Mô tả Usecase Xem thông tin tổng quan

Mô tả	Usecase này cho phép người dùng xem thông tin tổng quan về Nguồn tiền, báo cáo chi tiêu trong năm, các giao dịch gần đây.

Tác nhân		Người dùng
Tiền điều kiện		Người dùng sử dụng ứng dụng
Luồng sự kiện	**Luồng sự kiện chính**	Usecase bắt đầu khi người dùng mở ứng dụng hoặc bấm vào nút "Tổng quan" trên màn hình. - Khi người dùng mở ứng dụng hoặc bấm vào nút "Tổng quan" trên màn hình, hệ thống hiển thị các thông tin tổng quan về Nguồn tiền, báo cáo chi tiêu và các giao dịch gần đây. • Ca sử dụng kết thúc khi người dùng chọn chức năng khác hoặc thoát khỏi hệ thống.
	Luồng rẽ nhánh	Không có
Hậu điều kiện		Không có

- Biểu đồ trình tự:



Hình 2.7: Biểu đồ trình tự Xem thông tin tổng quan

2.3.2.2. Mô tả Usecase <Quản lý giao dịch>

Bảng 2.2: Mô tả Usecase Quản lý giao dịch

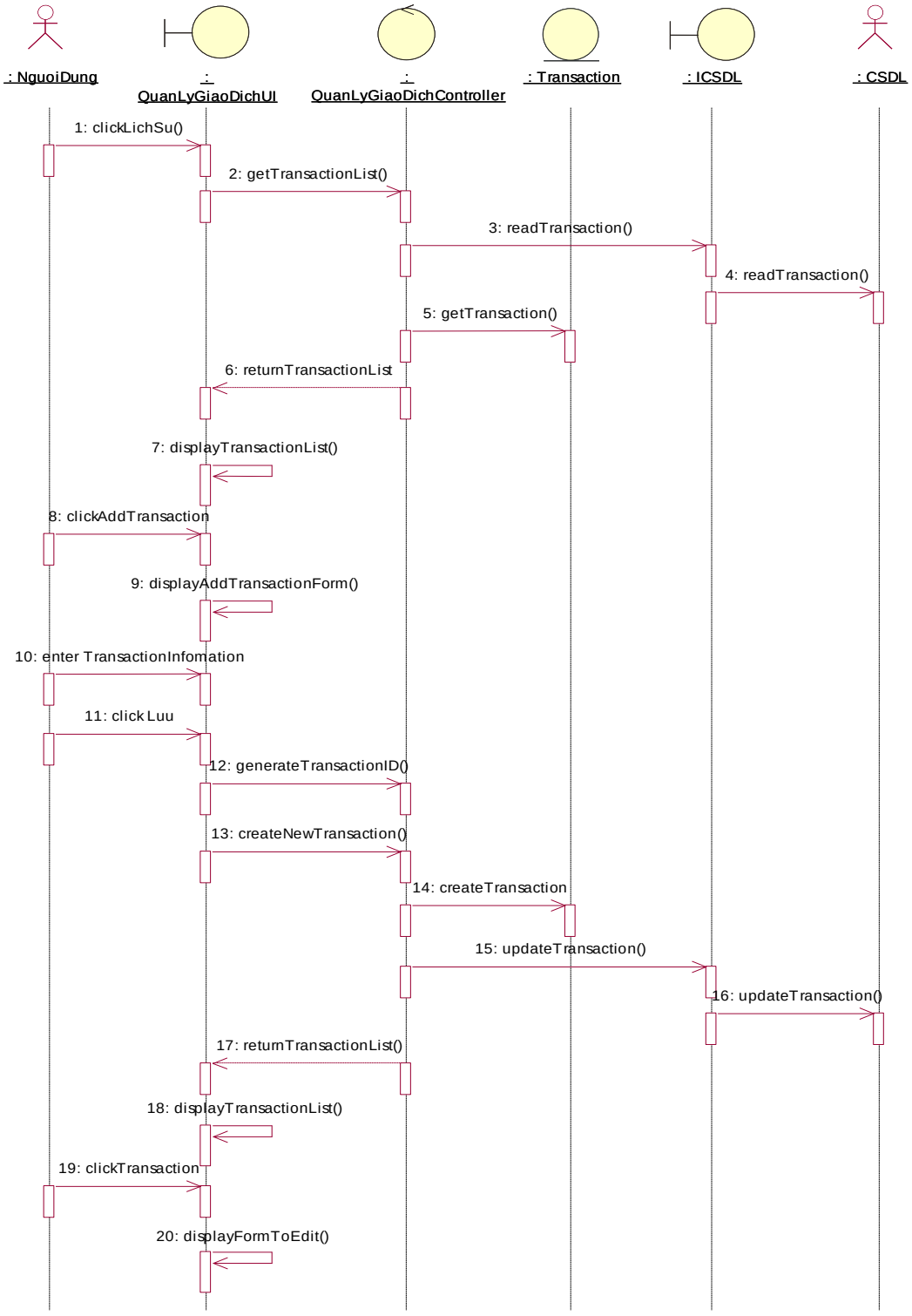
Mô tả		Usecase này cho phép người dùng quản lý và xem thông tin các giao dịch.
Tác nhân		Người dùng
Tiền điều kiện		Người dùng sử dụng ứng dụng
Luồng sự kiện	**Luồng sự kiện chính**	Usecase bắt đầu khi người dùng bấm vào "Lịch sử". Khi đó trên màn hình chính của ứng dụng hiển thị lên màn hình quản lý giao dịch 1. Xem danh sách giao dịch - Khi người dùng bấm vào "Lịch sử" trên màn hình, hệ thống hiển thị danh sách giao dịch trong tháng này.

		- Xem danh sách giao dịch tháng trước: Khi người dùng bấm vào "Tháng trước" hệ thống hiển thị các giao dịch trong tháng trước. - Xem danh sách giao dịch tháng này: Khi người dùng bấm vào "Tháng này" hệ thống hiển thị các giao dịch trong tháng này. - Xem danh sách giao dịch tương lai: Khi người dùng bấm vào "Tương lai" hệ thống hiển thị các giao dịch trong tương lai. - Xem danh sách giao dịch trong khoảng thời gian: Khi người dùng bấm vào nút "Lọc", hệ thống hiển thị form, người dùng nhập vào ngày đầu và ngày cuối rồi nhấn áp dụng, hệ thống hiển thị tất cả giao dịch trong khoảng thời gian người dùng vừa nhập. - Xem chi tiết giao dịch: Khi người dùng bấm vào một giao dịch bất kỳ, hệ thống hiển thị thông tin chi tiết giao dịch đó. 2. Thêm giao dịch - Người dùng bấm vào nút thêm giao dịch (Dấu cộng màu xanh lam) trên màn hình, hệ thống hiển thị lên màn hình thêm giao dịch.

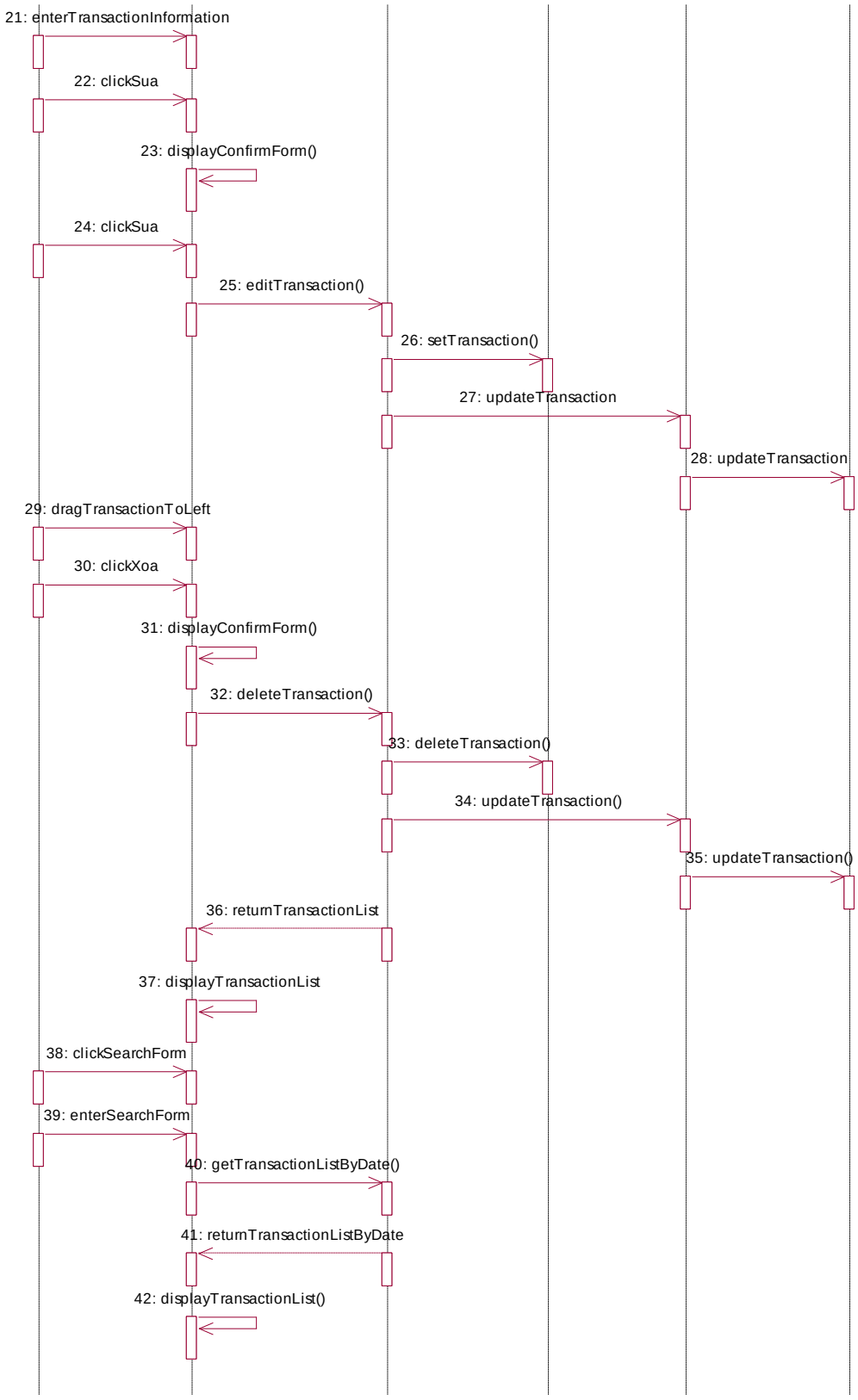
		- Người dùng nhập thông tin và chọn "Lưu" để hoàn thành. - Hệ thống kiểm tra thông tin và cập nhật thông tin vào cơ sở dữ liệu 3. Sửa giao dịch - Trên màn hình "Chi tiết giao dịch", người dùng nhập thông tin cần thay đổi và nhấn "Sửa". - Hệ thống cập nhật vào cơ sở dữ liệu. 4. Xóa giao dịch - Trong màn hình hiển thị các giao dịch, kéo giao dịch sang trái sẽ hiển thị nút "Xóa". - Bấm "Xóa" để thực hiện xóa giao dịch. - Hệ thống cập nhật thông tin vào cơ sở dữ liệu. 5. Tìm kiếm giao dịch - Trên màn hình quản lý giao dịch, người dùng nhấn vào thanh tìm kiếm và nhập tên loại giao dịch hoặc ghi chú của giao dịch cần tìm. - Hệ thống hiển thị màn hình danh sách chứa các giao dịch chứa các từ khóa tìm kiếm của người dùng đã nhập. • Ca sử dụng kết thúc khi người dùng chọn chức năng khác hoặc thoát khỏi hệ thống.

	Luồng rẽ nhánh	• Luồng 1.1: Trong Usecase xem danh sách giao dịch trong khoảng thời gian, khi đang điền form, nếu người dùng nhấn nút "Bỏ lọc", hệ thống quay về màn hình "Quản lý giao dịch" • Luồng 1.2: Trong Usecase xem danh sách giao dịch trong khoảng thời gian, khi đang điền form, nếu người dùng nhấn nút "Xóa", hệ thống xóa toàn bộ giá trị ngày bắt đầu và ngày kết thúc mà Người Dùng đã nhập. • Luồng 2.1: Người dùng nhập thiếu thông tin giao dịch: Hệ thống không cho phép lưu và hiển thị thông báo các trường bắt buộc. • Luồng 3.1: Người dùng nhập thiếu thông tin giao dịch để chỉnh sửa: Hệ thống không cho phép lưu và hiển thị thông báo các trường bắt buộc.
Hậu điều kiện		Không có

➢ Biểu đồ trình tự:



Hình 2.8: Mô tả Usecase Quản lý giao dịch



Hình 2.9: Biểu đồ trình tự Usecase Quản lý giao dịch

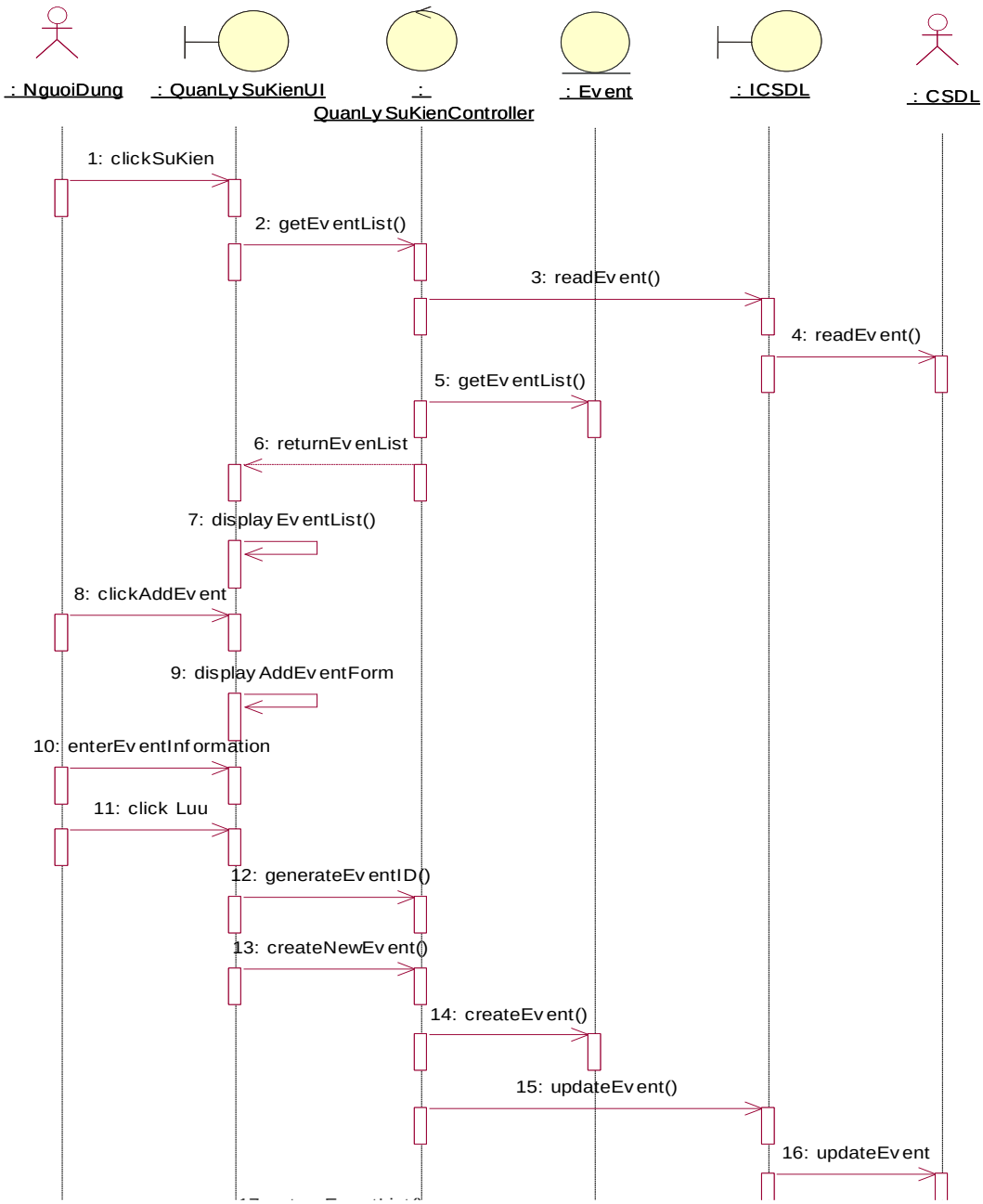
2.3.2.3. Mô tả Usecase <Quản lý sự kiện>

Bảng 2.3: Mô tả Usecase Quản lý sự kiện

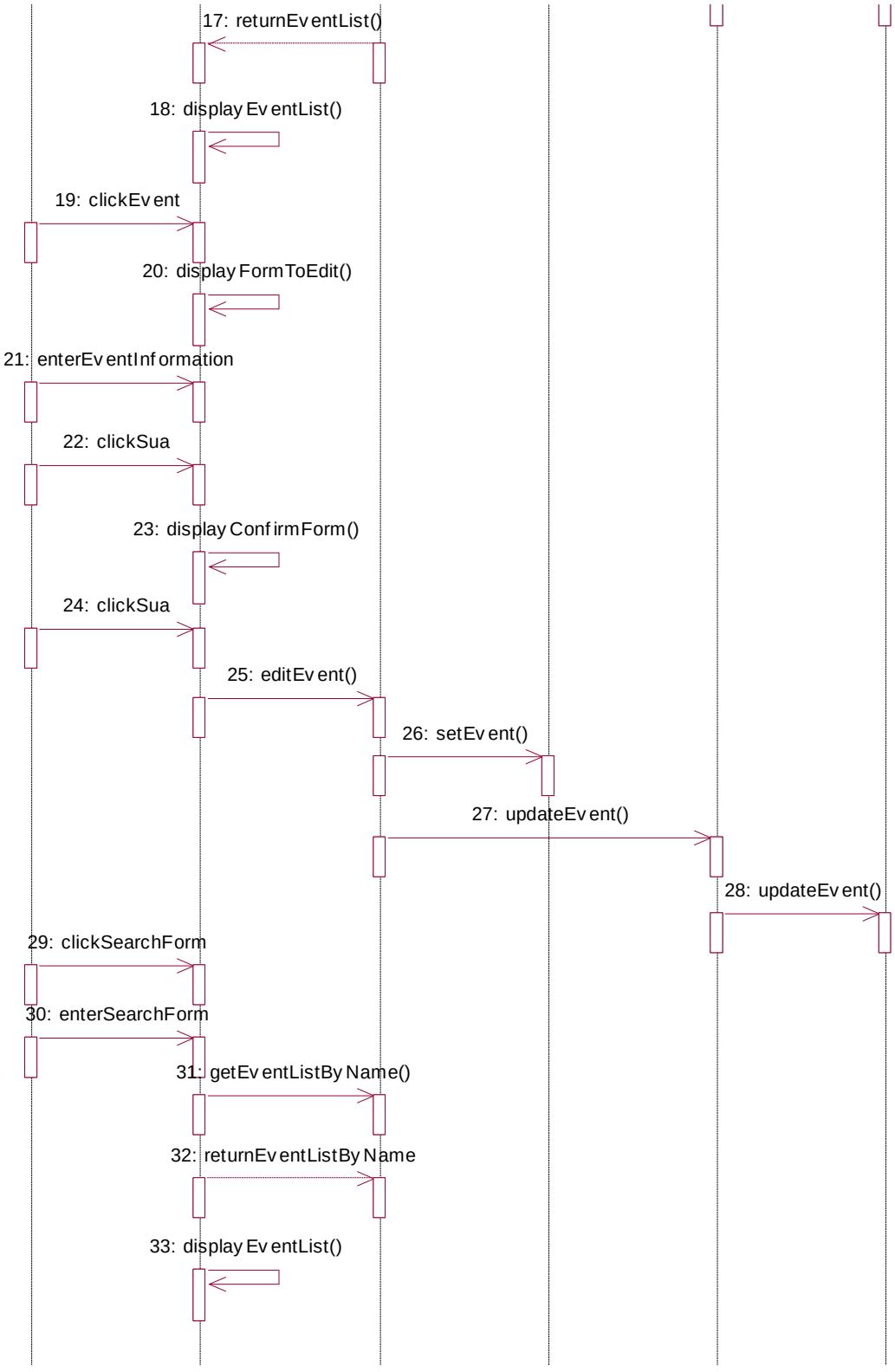
Mô tả		Usecase này cho phép người dùng quản lý và xem thông tin các sự kiện.
Tác nhân		Người dùng
Tiền điều kiện		Người dùng sử dụng ứng dụng
Luồng sự kiện	**Luồng sự kiện chính**	Usecase bắt đầu khi người dùng bấm vào nút "Sự kiện" trên màn hình 1. Thêm sự kiện - Người dùng bấm vào nút thêm sự kiện (Dấu cộng màu xanh lục) trên màn hình "Quản lý sự kiện", hệ thống hiển thị lên màn hình thêm sự kiện. - Người dùng nhập thông tin vào chọn "Lưu" để hoàn thành. - Hệ thống kiểm tra thông tin và cập nhật thông tin vào cơ sở dữ liệu 2. Xem danh sách sự kiện - Khi người dùng bấm vào nút "Sự kiện" trên màn hình, hệ thống hiển thị danh sách các sự kiện. 3. Xem chi tiết sự kiện - Ở màn hình danh sách sự kiện, khi người dùng bấm vào một sự kiện, hệ thống hiển thị màn hình "Chi tiết sự kiện" bao gồm tên sự kiện, số tiền cho phép của sự kiện, thời gian diễn ra, số

		tiền đã chi tiêu, danh sách các giao dịch trong sự kiện. 4. Sửa sự kiện - Trên màn hình "Chi tiết sự kiện", người dùng nhập thông tin cần thay đổi và nhấn "Sửa". - Hệ thống kiểm tra lại thông tin và cập nhật vào cơ sở dữ liệu. • Ca sử dụng kết thúc khi người dùng chọn chức năng khác hoặc thoát khỏi hệ thống.
	Luồng rẽ nhánh	• Luồng 1.1: Người dùng nhập thiếu thông tin sự kiện: Hệ thống không cho phép lưu và hiển thị thông báo các trường bắt buộc.
Hậu điều kiện		Không có

➢ Biểu đồ trình tự:



Hình 2.10: Biểu đồ trình tự Usecase Quản lý sự kiện



Hình 2.11: Biểu đồ trình tự Usecase Quản lý sự kiện

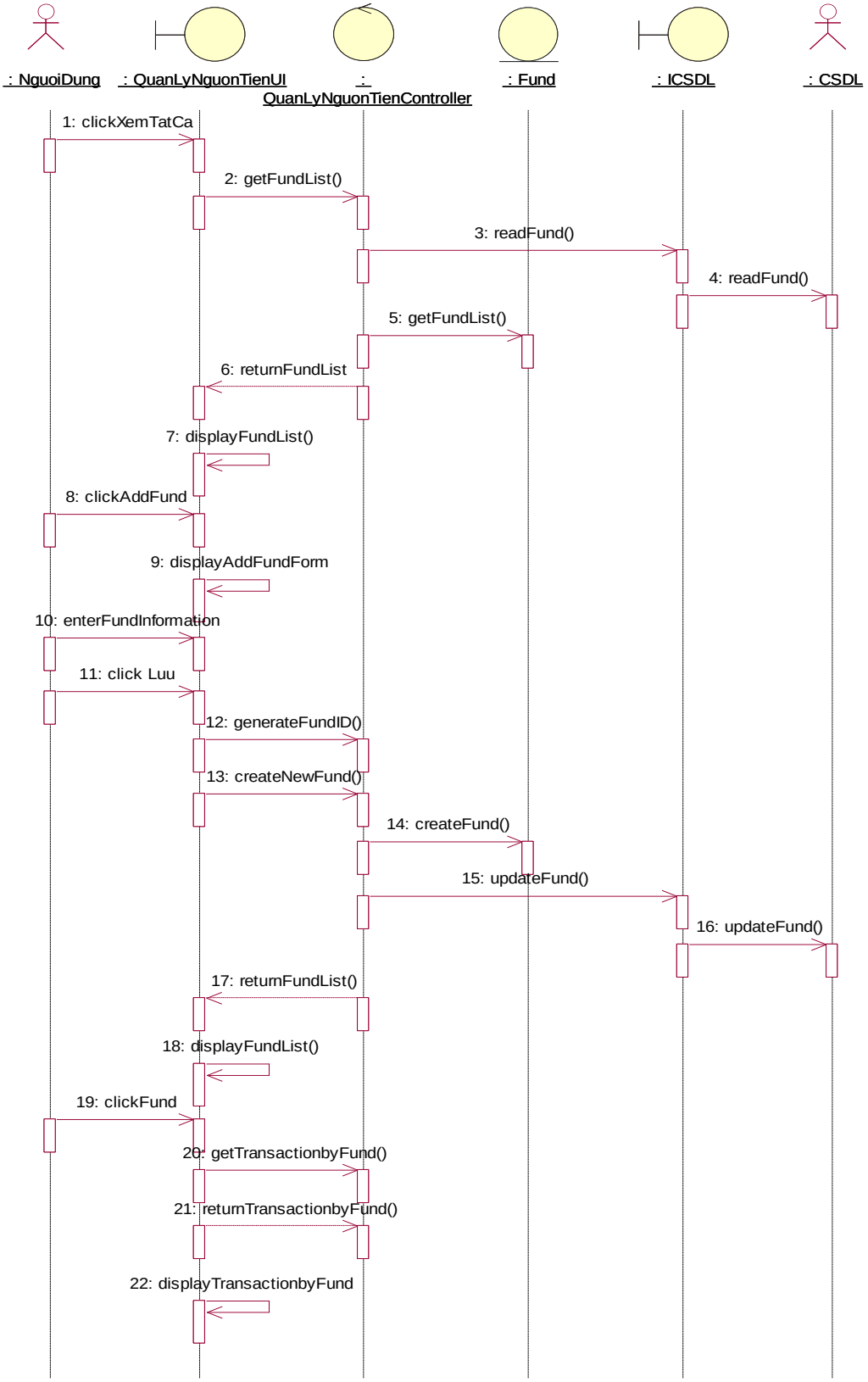
2.3.2.4. Mô tả Usecase <Quản lý nguồn tiền>

Bảng 2.4: Mô tả Usecase Quản lý nguồn tiền

Mô tả		Usecase này cho phép người dùng quản lý và xem thông tin các nguồn tiền.
Tác nhân		Người dùng
Tiền điều kiện		Người dùng sử dụng ứng dụng
Luồng sự kiện	**Luồng sự kiện chính**	Usecase bắt đầu khi người dùng bấm vào "Xem tất cả" trong mục "Ví của tôi" trên màn hình tổng quan. 1. Xem danh sách các nguồn tiền - Khi người dùng bấm vào "Xem tất cả" trong mục "Ví của tôi" trên màn hình tổng quan, hệ thống hiển thị danh sách các nguồn tiền. 2. Thêm nguồn tiền - Người dùng bấm vào nút thêm giao dịch (Dấu cộng) trên màn hình "Quản lý nguồn tiền", hệ thống hiển thị lên màn hình thêm nguồn tiền. - Người dùng nhập thông tin vào chọn "Lưu" để hoàn thành. - Hệ thống kiểm tra thông tin và cập nhật thông tin vào cơ sở dữ liệu 3. Xem danh sách giao dịch sử dụng nguồn tiền - Trên màn hình hiển thị danh sách các nguồn tiền, khi người dùng bấm vào

		một nguồn tiền, hệ thống hiển thị ra các giao dịch sử dụng nguồn tiền đó. • Ca sử dụng kết thúc khi người dùng chọn chức năng khác hoặc thoát khỏi hệ thống.
	Luồng rẽ nhánh	Không có
Hậu điều kiện		Không có

➢ Biểu đồ trình tự:



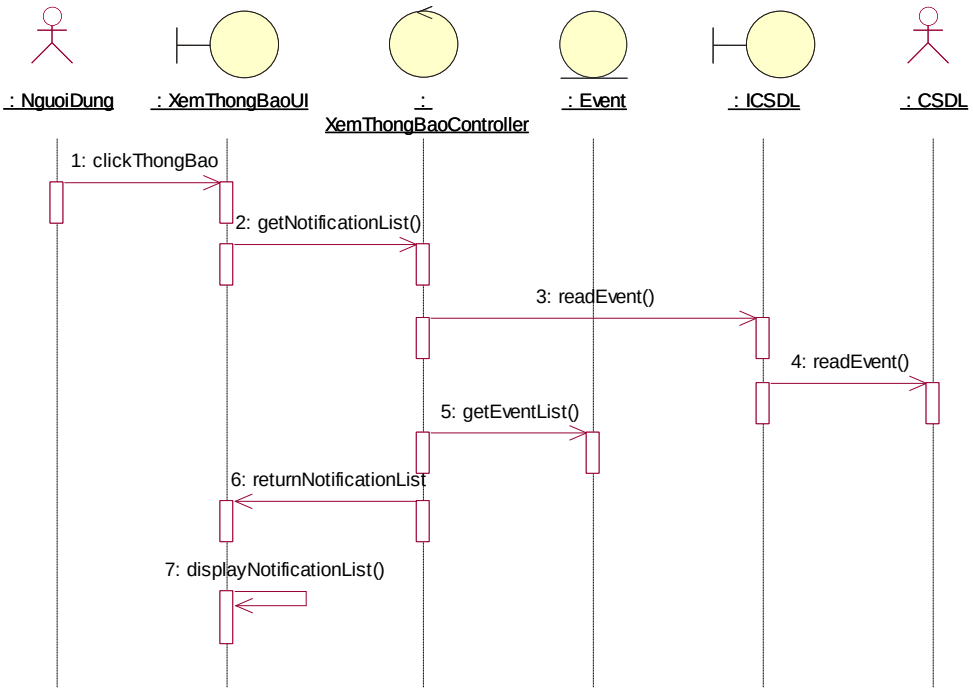
Hình 2.12: Biểu đồ trình tự Usecase Quản lý nguồn tiền

2.3.2.5. Mô tả Usecase <Xem thông báo>

Bảng 2.5: Mô tả Usecase Xem thông báo

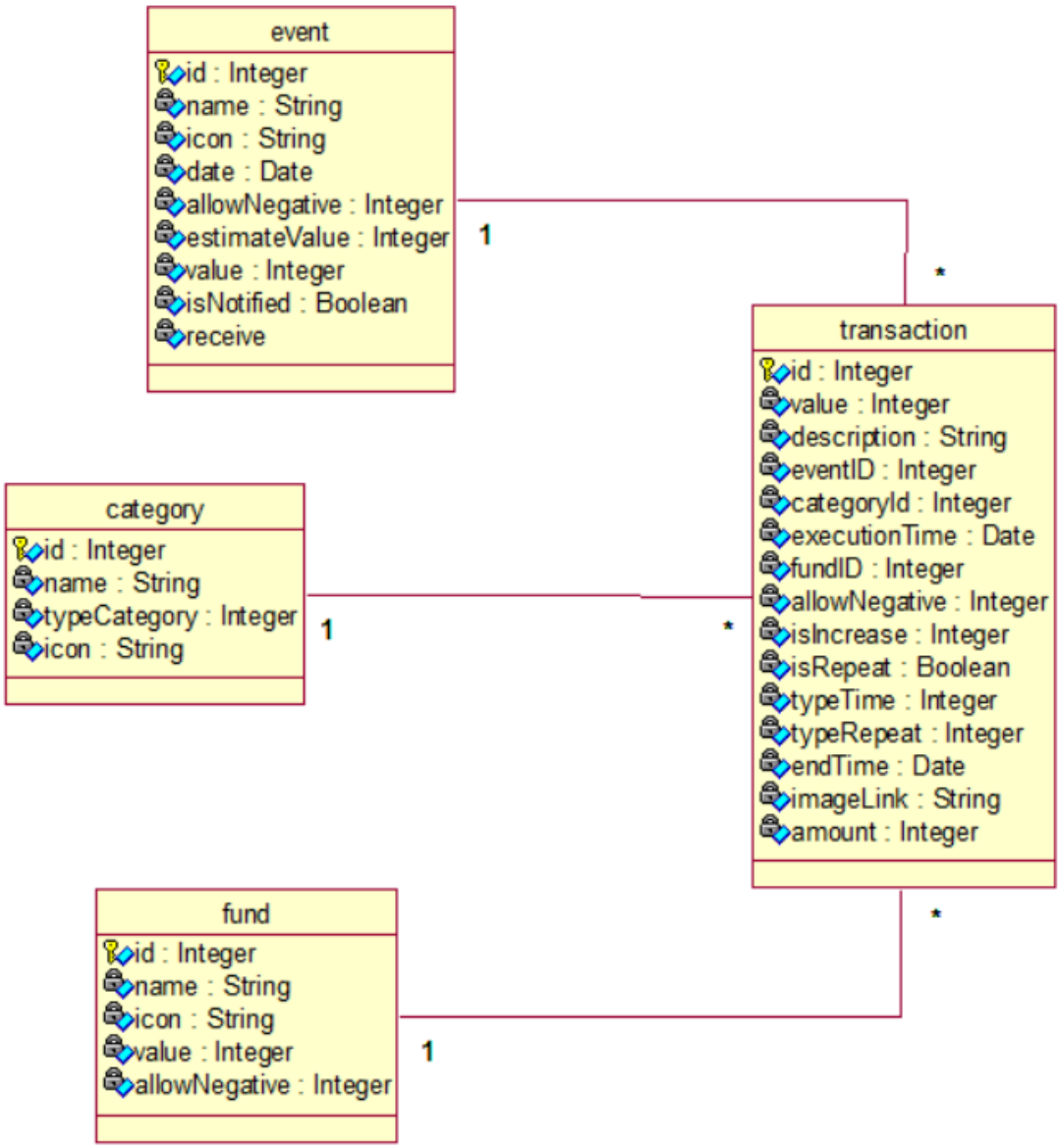
Mô tả		Usecase cho phép người dùng theo dõi các thông báo về các giao dịch.
Tác nhân		Người dùng
Tiền điều kiện		Người dùng sử dụng ứng dụng
Luồng sự kiện	**Luồng sự kiện chính**	Usecase bắt đầu khi người dùng bấm vào nút "Thông báo" trên màn hình - Sau khi người dùng bấm nút "Thông báo", hệ thống hiển thị danh sách thông báo.
	Luồng rẽ nhánh	Không có
Hậu điều kiện		Không có

- Biểu đồ trình tự:



Hình 2.13: Biểu đồ trình tự Xem thông báo

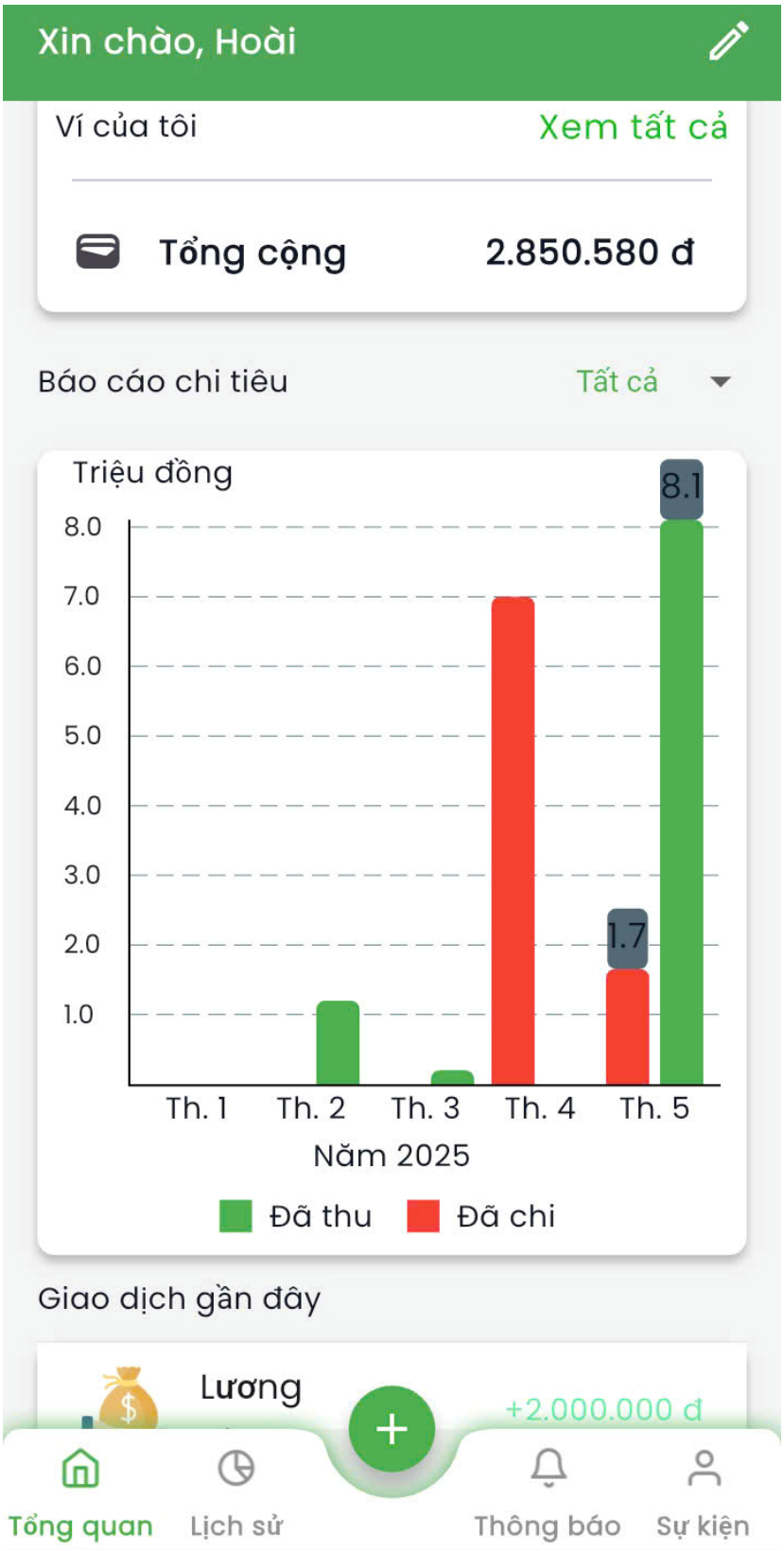
2.3.3. Biểu đồ lớp hệ thống



Hình 2.14: Biểu đồ lớp hệ thống

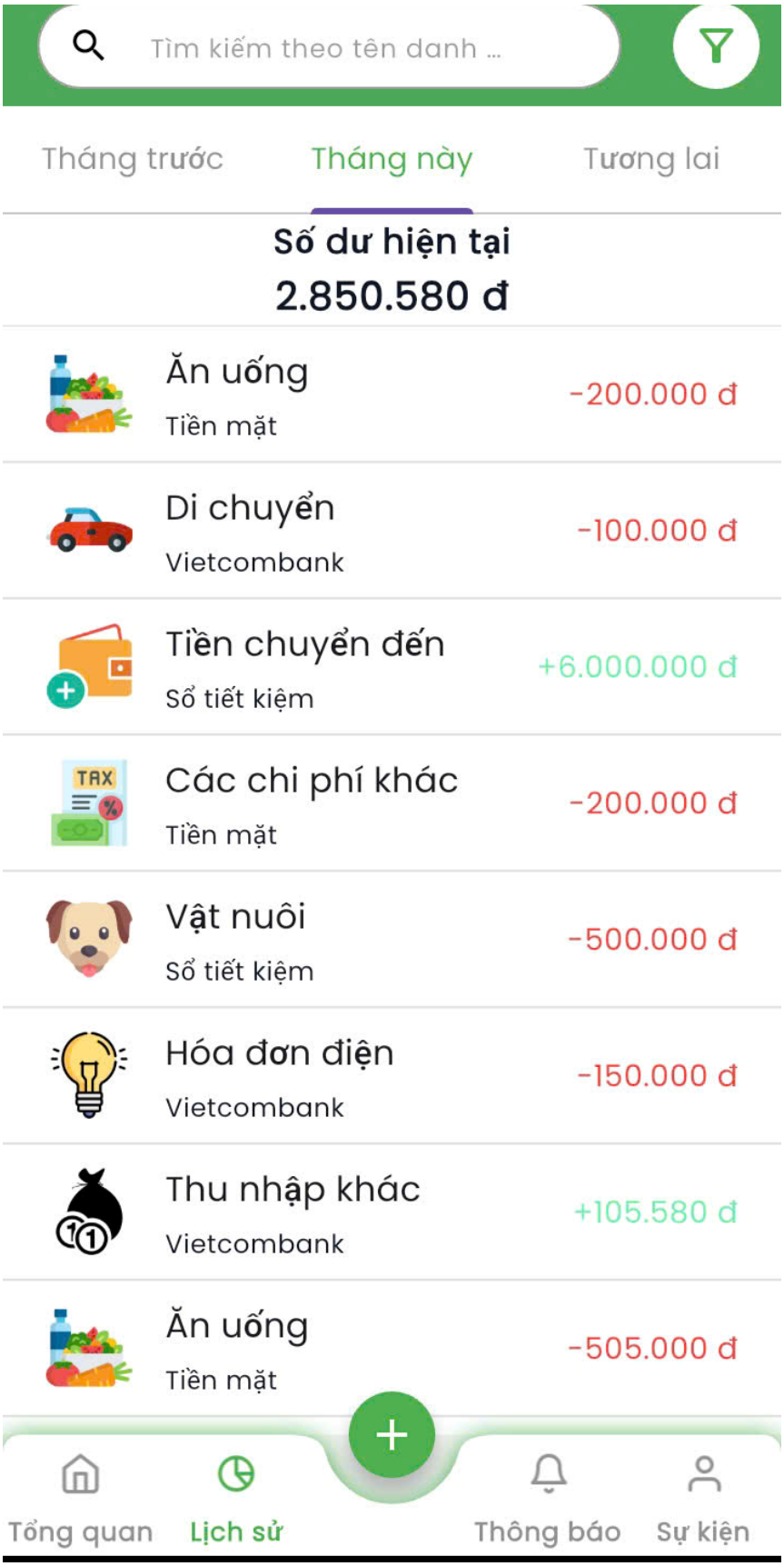
2.4. Thiết kế giao diện

2.4.1. Giao diện Tổng quan



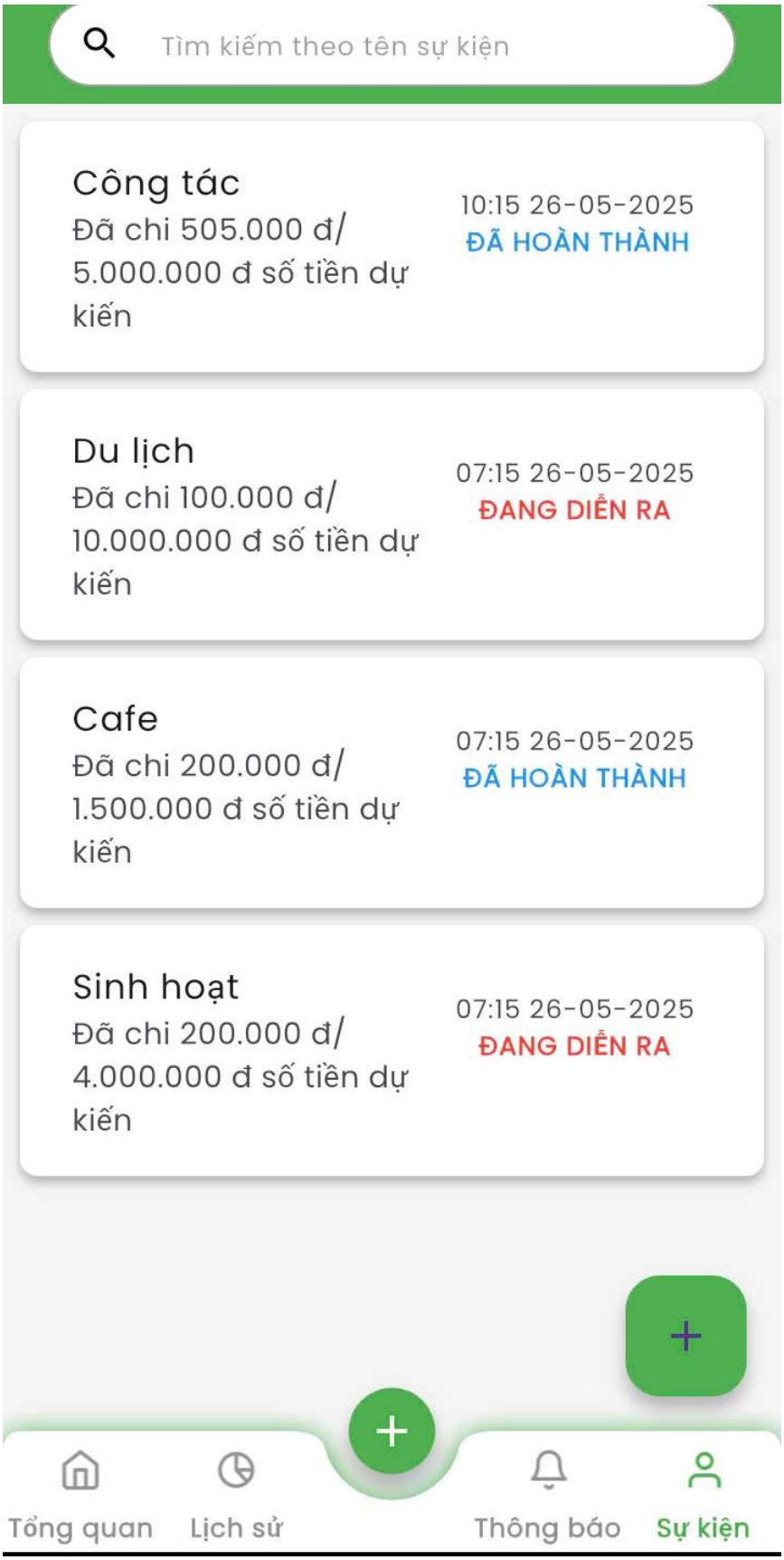
Hình 2.15: Giao diện tổng quan

2.4.2. Giao diện Quản lý giao dịch



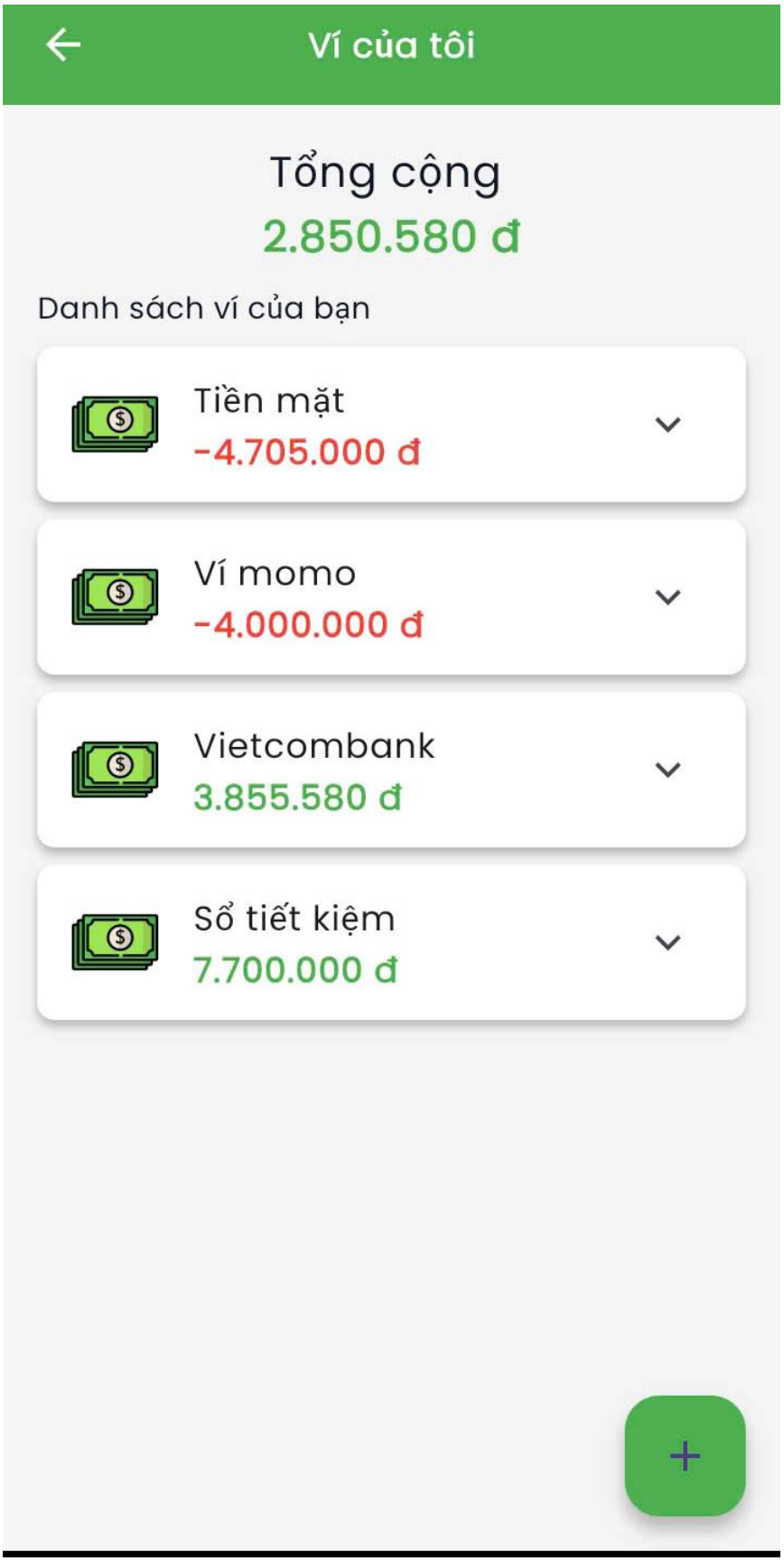
Hình 2.16: Giao diện Quản lý giao dịch

2.4.3. Giao diện Quản lý sự kiện



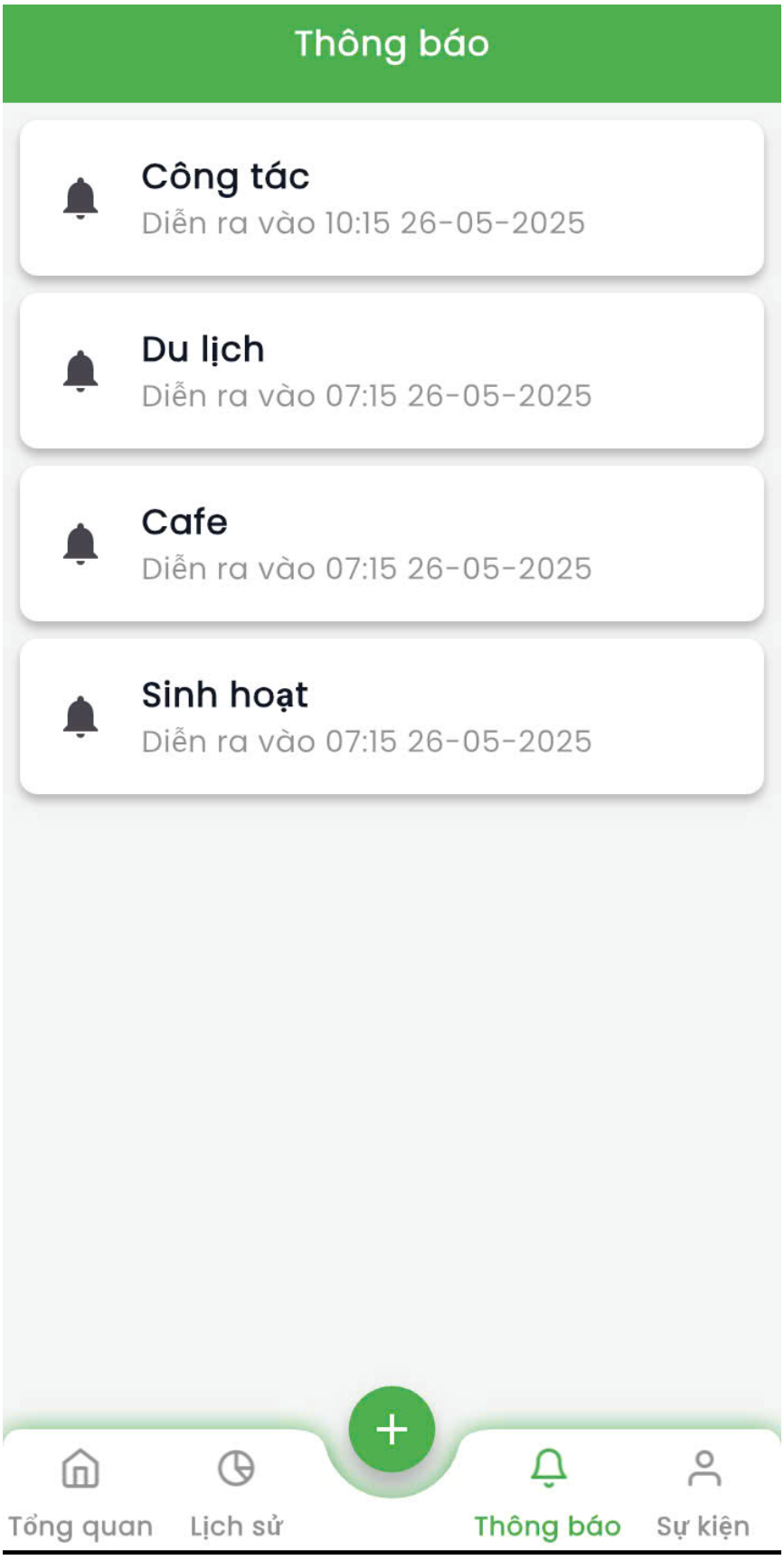
Hình 2.17: Giao dịch quản lý sự kiện

2.4.4. Giao diện Quản lý nguồn tiền



Hình 2.18: Giao diện quản lý nguồn tiền

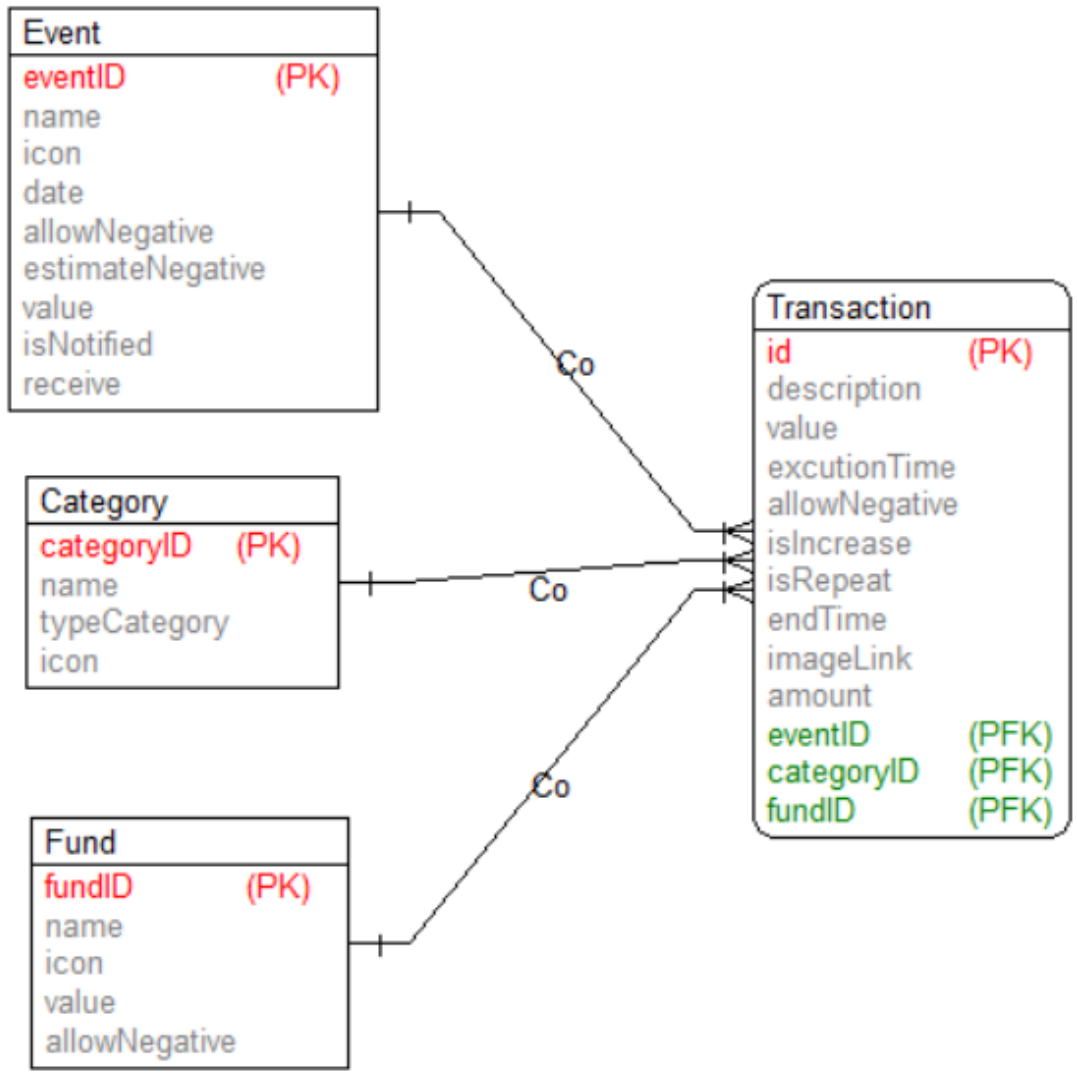
2.4.5. Giao diện Xem thông báo



Hình 2.19: Giao diện Xem thông báo

2.5. Thiết kế cơ sở dữ liệu

Bảng 2.6: Thiết kế cơ sở dữ liệu

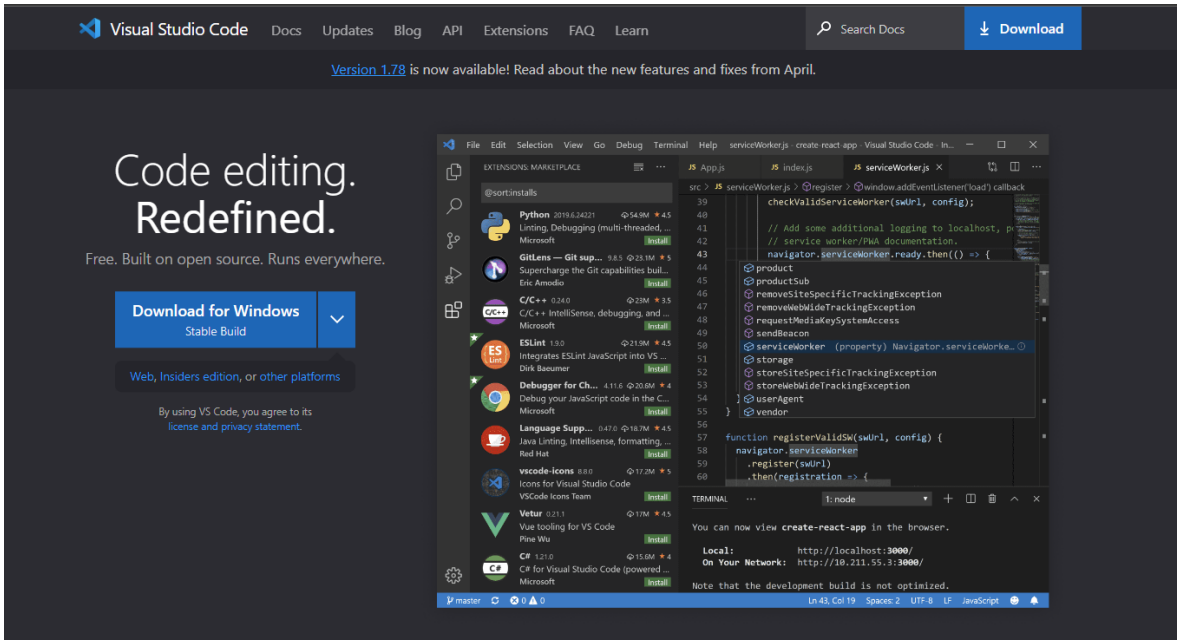


CHƯƠNG 3. KẾT QUẢ

3.1. Cài đặt môi trường

❖ Cài đặt VS Code

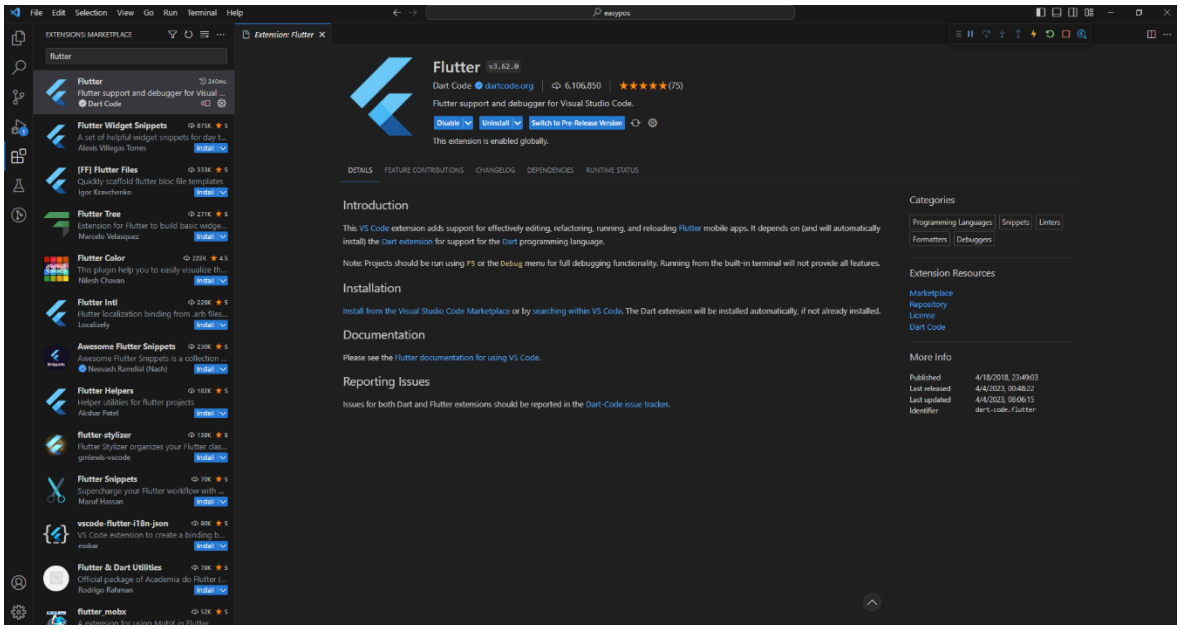
- Bước 1: Truy cập vào link https://code.visualstudio.com/ để tải xuống phiên bản VS Code
- Bước 2: Ấn nút download để thực hiện tải và cài đặt phiên bản mới nhất



Hình 3.1: Visual Studio Code

❖ Cài đặt Flutter và Dart plugin

- Mở VS Code.
- Chọn View > Command Palette….
- Nhấn “install”, sau đó chọn Extensions: Install Extensions.
- Tìm kiếm “flutter” trong khung tìm kiếm extensions, chọn Flutter trong danh sách, sau đó nhấn “install”.



Hình 3.2: Cài đặt Flutter

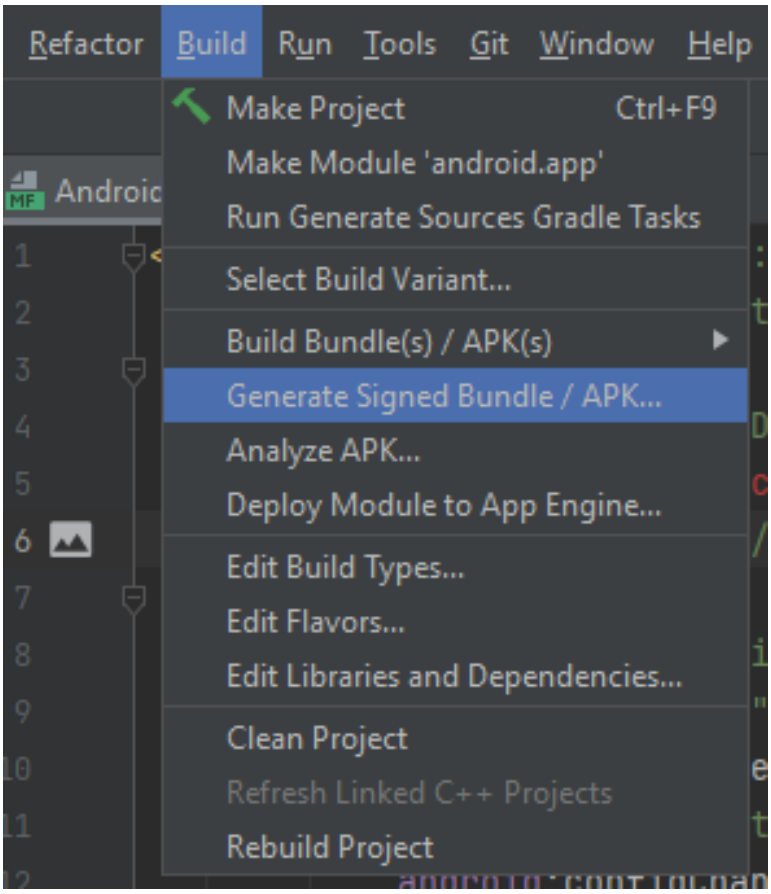
❖ Xác thực cài đặt bằng Flutter doctor

- Chọn View > Command Palette….
- Nhập “doctor” và chọn “Flutter: Run Flutter Doctor”
- Xem lại đầu ra trong khung OUTPUT để biết bất kỳ sự cố nào.

3.2. Cài đặt chương trình

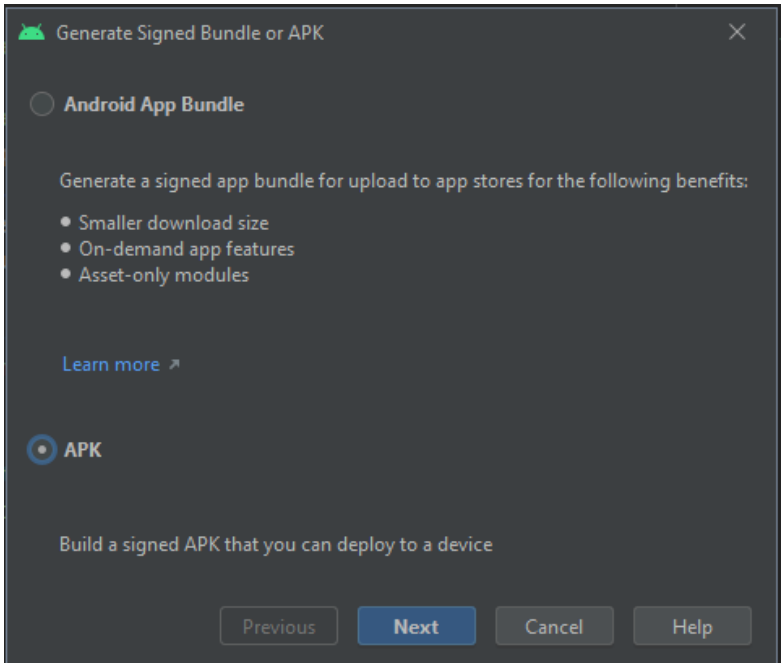
Bước 1: Tạo file keystore

- Tại menu Build / chọn Generate Signed APK…



Hình 3.3: Mô tả cách cài đặt chương trình

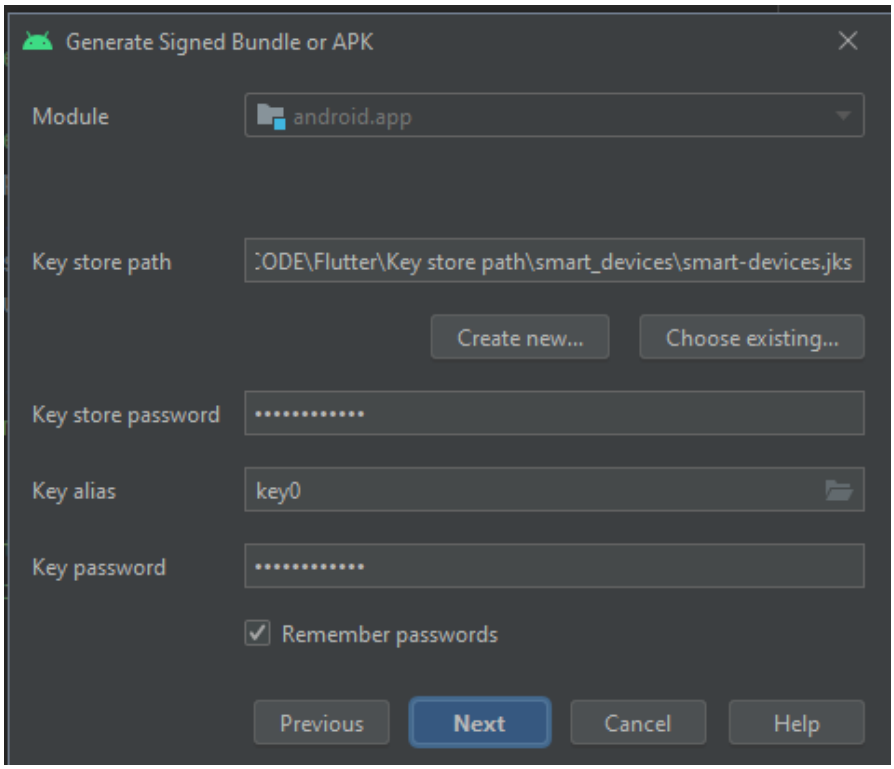
- Lựa chọn APK tại cửa sổ Generate Signed Bundle or APK và chọn Next



Hình 3.4: Mô tả cách cài đặt chương trình

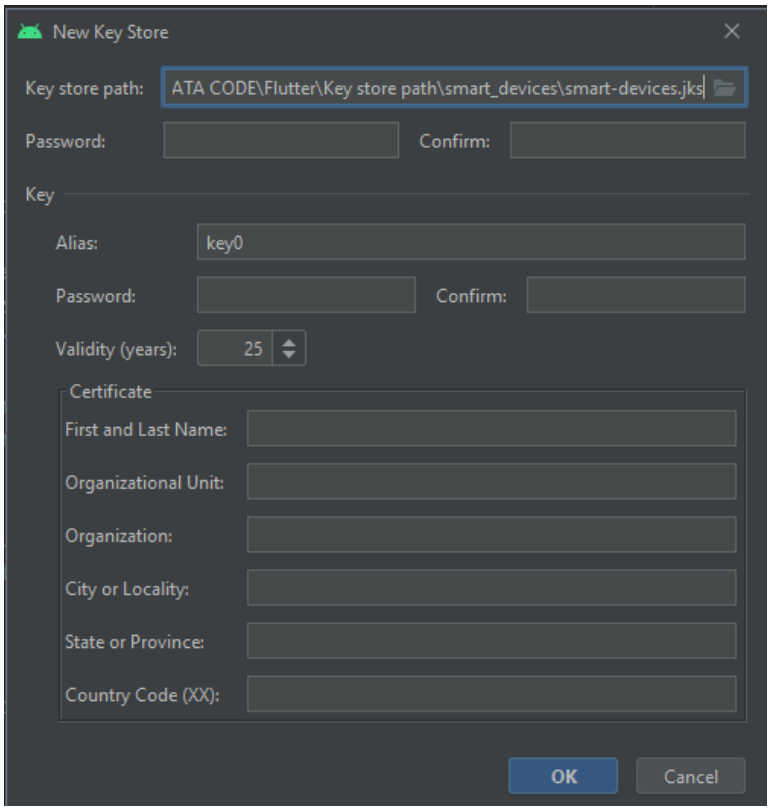
Bước 2: Nhúng keystore vào file apk

- Cửa sổ hiện lên chọn **Create new…** để tạo **Key Store**



Hình 3.5: Mô tả cách cài đặt chương trình

- Chọn đường dẫn tại mục Key store path và điền các thông tin cần thiết

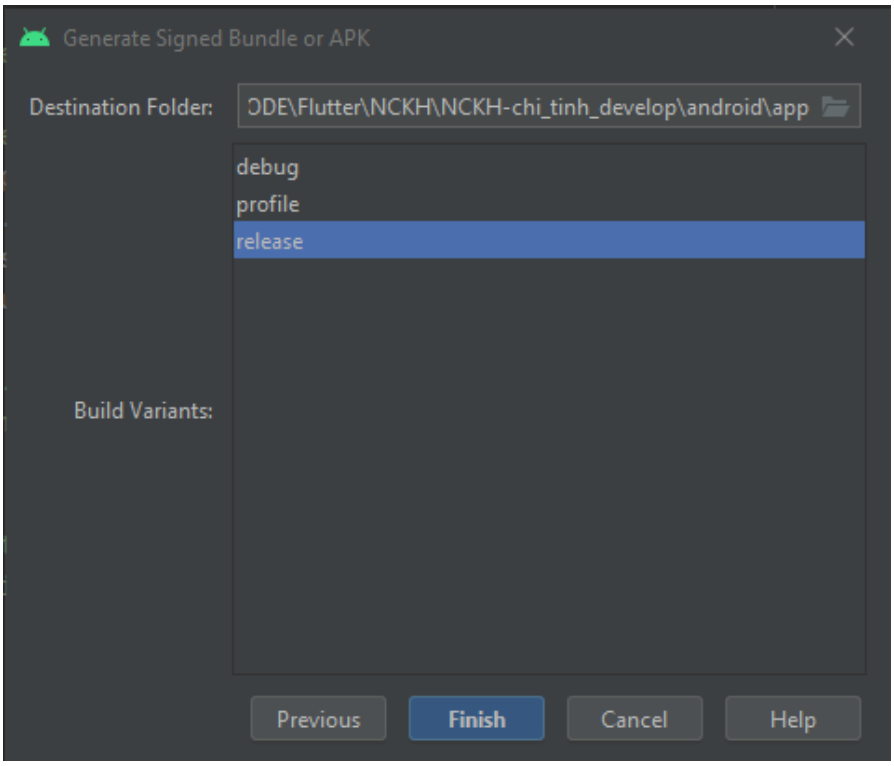


Hình 3.6: Mô tả cách cài đặt chương trình

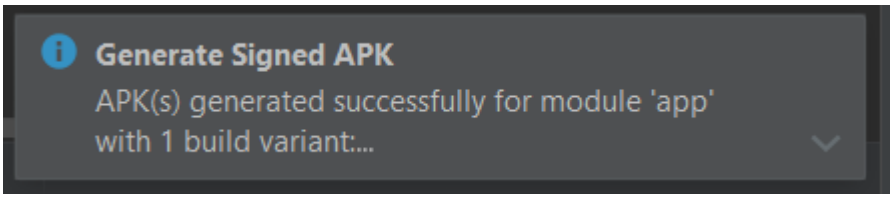
- Click "**OK**" để hoàn thành tạo Keystore, sau đó chọn "**Next**".

Bước 3: Chuyển file apk từ debug sang release

- Tiếp tục chọn **release** để và **Finish** để Generate file .apk của app



Hình 3.7: Mô tả cách cài đặt chương trình



Hình 3.8: Mô tả cách cài đặt chương trình

- Quá trình Generate hoàn thành và tạo được file cài đặt trên hệ điều hành Android.

3.3. Kết quả thu được

Hệ thống được xây dựng với những chức năng, đó là:

3.3.1. Quản lý loại giao dịch

- Phân loại có 2 loại giao dịch

  + Giao dịch lặp lại (Lặp theo ngày/tuần/tháng, sẽ tự sinh ra giao dịch theo từng loại lặp lại)

  + Giao dịch không lặp lại

- Thêm, sửa, xóa giao dịch:
  + Xác thực các trường dữ liệu như số tiền, danh mục, thời gian, nguồn tiền trước khi thêm/sửa
  + Xác nhận xóa/sửa
- Tìm kiếm theo tên, mô tả
- Lọc theo thời gian

3.3.2. Quản lý ví tiền

- Thêm mới ví tiền: Xác thực tên ví trước khi thêm

3.3.3. Thống kê

- Thống kê giao dịch, tổng thu, tổng chi theo từng ngày, theo ví tiền, theo sự kiện
- Thống kê chi tiêu theo tất cả các tháng, lọc theo từng ví
- Tổng hợp 5 giao dịch gần nhất

3.3.4. Quản lý sự kiện

- Thêm mới, sửa, đánh dấu hoàn thành hoặc chưa hoàn thành sự kiện
  + Xác thực tên, số tiền trước khi thêm/sửa
  + Xác nhận sửa, cập nhật trạng thái hoàn thành, chưa hoàn thành

3.3.5. Thông báo

- Thông báo trực tiếp những về thông tin sự kiện đã diễn ra trong khi dùng ứng dụng
- Thông báo từ hệ thống trước đúng một ngày diễn ra sự kiện đối với sự kiện diễn ra sau thời điểm thêm một ngày và thông báo ngay sau đó một phút với sự kiện diễn ra trước thời điểm sau thời điểm hiện tại một ngày

3.4. Kết quả kiểm thử

3.4.1. Kỹ thuật kiểm thử hộp đen (Black box testing)

Kiểm thử hộp đen (Black box testing) là một phương pháp kiểm thử phần mềm mà việc kiểm tra các chức năng của một ứng dụng không cần quan tâm vào cấu trúc nội bộ hoặc hoạt động của nó. Mục đích chính của kiểm thử hộp đen chỉ là để xem phần mềm có hoạt động như dự kiến trong tài liệu yêu cầu và liệu nó có đáp ứng được sự mong đợi của người dùng hay không.

- Các loại kiểm thử hộp đen

  - Kiểm thử chức năng (Functional Testing): Là loại kiểm thử có liên quan tới chức năng của toàn bộ hệ thống bên trong sản phẩm, phần mềm. Trong quá trình kiểm thử với Functional Testing chức năng sẽ được kiểm tra kỹ càng trong luồng dữ liệu đầu vào, sau đó đánh giá kết quả đầu ra mà không lo bị ảnh hưởng tới cấu trúc bên trong ứng dụng.

  - Kiểm thử phi năng (Non - Functional Testing): Loại kiểm thử này không tập trung vào chức năng cụ thể của hệ thống mà đánh giá các yếu tố phi chức năng như hiệu suất, độ ổn định, khả năng mở rộng, khả năng sử dụng, bảo mật, và khả năng tương thích. Mục đích là để kiểm tra xem phần mềm có đáp ứng các yêu cầu chất lượng đã đề ra hay không.

  - Kiểm thử hồi quy (Regression testing): Đây là phương pháp kiểm thử được thực hiện sau khi phần mềm đã được sửa lỗi, nâng cấp hoặc bảo trì. Kiểm thử hồi quy giúp đảm bảo rằng các thay đổi mới không gây ảnh hưởng tiêu cực đến các chức năng đã có từ trước. Điều này rất quan trọng để duy trì tính ổn định và độ tin cậy của phần mềm trong quá trình phát triển liên tục.

- Kiểm thử hộp đen có 8 kỹ thuật phổ biến và có nhiều ưu điểm nhất:
    - Phân vùng tương đương (equivalence partitioning)
    - Phân tích giá trị biên (boundary value analysis)
    - Bảng quyết định (decision tables)
    - Kiểm thử chuyển đổi trạng thái (state transition testing).
    - Kỹ thuật kiểm thử các bộ n thần kỳ (Pairwise)
    - Kỹ thuật phân tích vùng miền (Domain analysis)
    - Kỹ thuật dựa trên đặc tả Use case (Use case)
    - Kỹ thuật dùng lược đồ quan hệ nhân quả (Cause- Effect Diagram)

3.4.2. Kiểm thử chức năng

Phần mềm được thực hiện kiểm thử chức năng (Functional Testing). Kiểm thử chức năng là một loại kiểm thử hộp đen (Black box testing) và test case của nó được dựa trên đặc tả của ứng dụng phần mềm/thành phần đang test. Các chức năng được test bằng cách nhập vào các giá trị và kiểm tra kết quả đầu ra, ít quan tâm đến cấu trúc bên trong của ứng dụng.

Bảng 3.1: Bảng kiểm thử chức năng

ID	Chức năng	Trường hợp kiểm thử	Bước thực hiện	Dữ liệu kiểm thử	Đầu ra thực tế	Đầu ra mong muốn	Kết quả

TQ1	Xem thông tin tổng quan	Xem thông tin tổng quan	1. Bấm vào nút "Tổng quan" trên màn hình		Hệ thống hiển thị các thông tin tổng quan về Nguồn tiền, báo cáo chi tiêu và các giao dịch gần đây	Hệ thống hiển thị các thông tin tổng quan về Nguồn tiền, báo cáo chi tiêu và các giao dịch gần đây	Pass
SK1	Quản lý sự kiện	Thêm sự kiện	1. Bấm vào nút thêm sự kiện (Dấu cộng màu xanh lục) trên màn hình "Quản lý sự kiện" 2. Điền thông tin về tên sự kiện, số tiền, chọn ngày, chọn giờ 3. Nhấn vào "Lưu" bên	Tên sự kiện: Ăn uống Số tiền: 100.000đ Ghi chú: Tiền ăn Chọn ngày: 07/05/2025 Chọn thời gian: 07:15	Màn hình hiển thị "Thêm mới sự kiện thành công"	Màn hình hiển thị "Thêm mới sự kiện thành công"	Pass

			góc phải trên của màn hình				
SK2		Xem danh sách sự kiện	1. Bấm vào nút "Sự kiện" trên màn hình	Thông tin sự kiện vừa thêm ở trên	Màn hình hiển thị các sự kiện	Màn hình hiển thị các sự kiện	Pass
SK3		Xem chi tiết sự kiện	1. Ở màn hình danh sách sự kiện, khi người dùng bấm vào một sự kiện, hệ thống hiển thị màn hình "Chi tiết sự kiện"		Hiển thị thông tiên chi tiết của sự kiện	Hiển thị thông tiên chi tiết của sự kiện	Pass
SK4		Sửa sự kiện	1. Trên màn hình "Chi tiết sự kiện", người dùng nhập thông tin cần thay đổi và nhấn "Sửa"		Màn hình hiển thị "Sửa sự kiện thành công"	Màn hình hiển thị "Sửa sự kiện thành công"	Pass

GD1	Quản lý giao dịch	Thêm giao dịch	1. Bấm vào nút thêm giao dịch (Dấu cộng màu xanh lam) 2. Điền thông tin về số tiền, nhập ghi chú, chọn danh mục, thời gian, nguồn tiền, sự kiện, hình ảnh 3. Nhấn vào "Lưu" bên góc phải trên của màn hình	Số tiền: 100.000đ Ghi chú: Tiền ăn Chọn danh mục: Ăn uống Chọn thời gian: 07/05/2025 – 07:15 Chọn nguồn tiền: Tiền mặt Chọn sự kiện: Ăn uống	Màn hình hiển thị "Thêm giao dịch thành công"	Màn hình hiển thị "Thêm giao dịch thành công"	Pass
GD2		Xem giao dịch	1. Bấm vào "Lịch sử"	Xem được thông tin giao dịch vừa thêm ở trên	Màn hình hiển thị thông tin các cuộc giao dịch	Màn hình hiển thị thông tin các cuộc giao dịch	Pass

					trong tháng này	trong tháng này	
GD3		Sửa giao dịch	1. Trên màn hình "Chi tiết giao dịch", người dùng nhập thông tin cần thay đổi và nhấn "Sửa" 2. Sửa thông tin 3. Trên màn hình xác bấm nhấn vào nút "Sửa"	Sửa số tiền thành: 150.000đ	Màn hình hiển thị "Sửa giao dịch thành công"	Màn hình hiển thị "Sửa giao dịch thành công"	Pass
GD4		Xoá giao dịch	1. Trong màn hình hiển thị các giao dịch, kéo giao dịch sang trái sẽ hiển thị nút "Xóa"	Xoá giao dịch vừa thêm ở trên	Màn hình hiển thị "Xoá giao dịch thành công"	Màn hình hiển thị "Xoá giao dịch thành công"	Pass

			2. Nhấn vào “Xoá” 3. Tại màn hình xác nhận nhấn nút “Xoá”				
GD5		Tìm kiếm giao dịch	1. Trên màn hình quản lý giao dịch, người dùng nhấn vào thanh tìm kiếm và nhập tên loại giao dịch	Thông tin tìm kiếm: Ăn uống	Màn hình hiển thị thông tin giao dịch “Ăn uống”	Màn hình hiển thị thông tin giao dịch “Ăn uống”	Pass
NT1	Quản lý nguồn tiền	Xem danh sách các nguồn tiền	1. Bấm vào “Xem tất cả” trong mục “Ví của tôi” trên màn hình tổng quan		Màn hình hiển thị danh sách các nguồn tiền	Màn hình hiển thị danh sách các nguồn tiền	Pass
NT2		Thêm nguồn tiền	1. Bấm vào nút thêm giao dịch (Dấu cộng)	Tên ví: Momo	Màn hình hiển thị “Thêm mới Ví	Màn hình hiển thị “Thêm mới Ví	Pass

			trên màn hình “Quản lý nguồn tiền” 2. Nhập thông tin nguồn tiền 3. Bấm “Lưu”	Số tiền: 1.000.000 đ Chọn loại ví: Ví cơ bản	tiền thành công”	tiền thành công”	
NT3		Xem danh sách giao dịch sử dụng nguồn tiền	1. Trên màn hình hiển thị danh sách các nguồn tiền, khi người dùng bấm vào một nguồn tiền		Màn hình hiển thị ra các giao dịch sử dụng nguồn tiền đó	Màn hình hiển thị ra các giao dịch sử dụng nguồn tiền đó	Pass
TB1	Xem thông báo	Xem thông báo	1. Bấm vào nút “Thông báo” trên màn hình		Màn hình hiển thị danh sách thông báo	Màn hình hiển thị danh sách thông báo	Pass

KẾT LUẬN

❖ Đánh giá kết quả

Sau quá trình nghiên cứu và phát triển, đề tài "Ứng dụng quản lý chi tiêu bằng Flutter" đã đạt được nhiều kết quả tích cực, thể hiện qua việc áp dụng hiệu quả các kiến thức đã học và triển khai thành công một sản phẩm ứng dụng cơ bản có tính thực tiễn cao.

- Phần hoàn thành:

+ Nắm vững và áp dụng đúng quy trình phát triển một ứng dụng di động đa nền tảng, từ giai đoạn khảo sát, phân tích yêu cầu, thiết kế hệ thống cho đến triển khai và thử nghiệm ứng dụng.

+ Thành thạo trong việc sử dụng trình biên tập mã nguồn Visual Studio Code (VSCode) cùng các công cụ hỗ trợ phát triển, gỡ lỗi và quản lý mã nguồn trong suốt quá trình thực hiện đồ án.

+ Hoàn thiện bài khảo sát, phân tích và thiết kế hệ thống ứng dụng "Quản lý chi tiêu bằng Flutter", đảm bảo các chức năng cơ bản như ghi chép chi tiêu, quản lý ví tiền, theo dõi sự kiện và lọc dữ liệu.

+ Sử dụng thành thạo ngôn ngữ lập trình Dart và framework Flutter, thể hiện qua việc xây dựng được một ứng dụng hoàn chỉnh về mặt kỹ thuật, có khả năng hoạt động ổn định trên cả hai nền tảng Android và iOS

- Phần chưa hoàn thành:

+ Tuy nhiên, cần hoàn thiện thêm giao diện để khách hàng dễ sử dụng. Điều này rất quan trọng vì giao diện đóng vai trò quan trọng trong trải nghiệm người dùng và ảnh hưởng đến sự thành công của ứng dụng. Em có thể đưa thêm các tính năng và cải thiện trải nghiệm người dùng để tăng tính hữu dụng của ứng dụng.

Tóm lại, em đã hoàn thành một phần lớn của đề tài "Ứng dụng quản lý chi tiêu bằng Flutter", đồng thời đã có kiến thức và kinh nghiệm để phát triển các ứng dụng đa nền tảng. Tuy nhiên, để đạt được sự thành công về mặt thương mại, em cần phải hoàn thiện giao diện và trải nghiệm người dùng của ứng dụng.

❖ **Hướng phát triển**

- Thiết kế và lập trình mở rộng các chức năng: Đồng bộ tự động tích hợp hệ thống Back End, Tích hợp thêm quản lý nguồn tiền đến các thẻ ngân hàng và các loại ví điện tử
- Hỗ trợ thêm chế độ Dark mode.
- Hỗ trợ giao diện dạng widget cho ứng dụng trên màn hình chính của điện thoại.
- Hỗ trợ giao diện ứng dụng trên IOS, Window.
- Cải thiện sự mượt mà của ứng dụng. Nâng cao trải nghiệm người dùng.
- Em rất mong tiếp tục nhận được sự giúp đỡ và tạo điều kiện của thầy cô và nhà trường để em có cơ hội phát triển, hoàn thiện ứng dụng tốt hơn trong thời gian tới.

TÀI LIỆU THAM KHẢO

[1]. Nguyễn Văn Hiểu, Nguyễn Văn Biên, Nguyễn Văn Huy (2020), "Giáo trình lập trình ứng dụng di động", NXB Đại học Quốc gia Hà Nội.

[2]. Kameron Hussain, Frahaan Hussain (2024), "Mastering Visual Studio Code: Navigating the Future of Development", Sonar Publishing.

[3]. Gilad Bracha (2015), "The Dart Programming Language", Addison-Wesley Professional.

[4]. Google (2017), "Flutter SDK Overview", *Flutter Documentation*, Available: https://docs.flutter.dev/tools/sdk.

[5]. AngularDart Community (2023), "AngularDart Community Documentation," *angulardart.xyz*, Available: https://angulardart.xyz/guide/setup.

[6]. E. Freitas (2019), *AngularDart Succinctly*, Syncfusion Free Ebooks, Available: https://www.syncfusion.com/succinctly-free-ebooks/angulardart-succinctly/base-ui-component.

[7]. Michael Owens (2021), "The Definitive Guide to SQLite", Apress Publishing.

[8]. John R. Irons (2007), "Model-View-Controller Architecture", Addison-Wesley Professional.

[9]. Đỗ Ngọc Sơn, Phan Văn Viên, Nguyễn Phương Nga (2015), "Giáo trình hệ quản trị cơ sở dữ liệu SQL", NXB Khoa học và Kỹ thuật.

[10]. Nguyễn Thị Thanh Huyền, Ngô Thị Bích Thúy, Phạm Kim Phượng (2011), "Giáo trình phân tích thiết kế hệ thống", NXB Giáo dục Việt Nam.

[11]. Vũ Thị Dương, Phùng Đức Hòa, Nguyễn Thị Hương Lan (2015), "Giáo trình Phân tích thiết kế hướng đối tượng", NXB Khoa học và Kỹ thuật.