HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
BỘ MÔN CÔNG NGHỆ PHẦN MỀM

BÁO CÁO BÀI TẬP LỚN
HỌC PHẦN: THỰC TẬP CƠ SỞ
MÃ HỌC PHẦN: INT13147

Sinh viên thực hiện:

Họ và tên: Tạ Ngọc Tài

Số điện thoại: 0969259650

Mã sinh viên: B22DVCN281

Tên nhóm: VH03

Tên lớp: D22VHCN01-B

Giảng viên hướng dẫn: ThS. Nguyễn Văn Tiến

HÀ NỘI 2025

MỤC LỤC

DANH MỤC CÁC HÌNH VẼ

DANH MỤC CÁC BẢNG BIỂU

DANH MỤC CÁC TỪ VIẾT TẮT

Từ viết tắt	Thuật ngữ tiếng Anh/Giải thích	Thuật ngữ tiếng Việt/Giải thích
CNTT		Công nghệ thông tin
CSDL		Cơ sở dữ liệu
ERD	Entity Relationship Diagram	Sơ đồ thực thể kết hợp
HQTCSDL		Hệ quản trị cơ sở dữ liệu
IDE	Intergrated Development Environment	Môi trường phát triển tích hợp
iOS	iPhone Operating System	Hệ điều hành cho thiết bị di động của Apple
MVVM	Model - View - ViewModel	Mô hình kiến trúc phần mềm
UI	User Interface	Giao diện người dùng
UX	User Experience	Trải nghiệm người dùng
UML	Unified Modeling Language	Ngôn ngữ mô hình hóa thống nhất
UUID	Univerally Unique Identifier	Mã định danh duy nhất toàn cầu
CSV	Comma-Separated Values	Định dang tập tin văn bản lưu trữ dữ liệu dạng bảng

MỞ ĐẦU

Ngày nay, việc quản lý tài chính cá nhân là một khía cạnh quan trọng trong cuộc sống hiện đại. Sự thiếu hụt một công cụ hiểu quả để theo dõi chi tiêu và thu nhập có thể dẫn đến nhiều khó khăn trong việc kiểm soát tài chính. Xuất phát từ nhu cầu thực tiễn đó, đề tài xây dựng ứng dụng di động "App Quản lý Chi tiêu: ChiTiêu+" đã được lựa chọn.

Dự án này tập trung vào việc phát triển một giải pháp phần mêm trên nền tảng IOS chạy cục bộ, sử dụng ngôn ngữ lập trình Swift và hệ thống quản lý dữ liệu Core Data để lưu trữ. Mục tiêu chính của ứng dụng là cung cấp một công cụ đơn giản, trực quan, giúp người dùng có thể dễ dàng ghi chép, phân loại, phân tích, đánh giá được các khoản chi tiêu hàng ngày, tháng, năm ngay trên chính thiết bị của mình mà không cần kết nối internet.

Nội dung báo cáo này trình bày toàn bộ quá trình thực hiện dự án, từ giai đoạn phân tích yêu cầu, thiết kế hệ thống, cho đến phát triển và kiểm thử. Báo cáo được cấu trúc thành phần chính: Tổng quan dự án, Phân tích & Thiết kế hệ thống, Quá trình phát triển và Kết quả đạt được, nhằm cung cấp một cái nhìn toàn diện và có hệ thống về sản phẩm cuối cùng.

CHƯƠNG 1. GIỚI THIỆU CHUNG

1.1 Lý do chọn đề tài

Trong bối cảnh nền kinh tế số đang phát triển mạnh mẽ, thói quen chi tiêu của con người đã có những sự chuyển dịch đáng kể. Sự phổ biến của các hình thức thanh toán không dùng tiền mặt như ví điện tử, chuyển khoản ngân hàng hay thẻ tín dụng mạng lại sự tiện dụng to lớn, nhưng đồng thời cũng tạo ra thách thức không nhỏ trong việc kiểm soát dòng tiền cá nhân, Nhiều người đặc biệt là các bạn Gen Z mới đi học và đi làm, thường rơi vào tình trạng "rỗng túi" vào cuối tháng mà không biết tiền bay đi đâu, hoặc gặp khó khăn trong việc thiết lập các mục tiêu, ngân sách chi tiêu để dự phòng cho tương lai. Việc ghi chép thử công qua sổ sách hay Excel thường tốn thời gian và thiếu tính linh hoạt, dẫn đến việc người dùng dễ dàng từ bỏ thói quen này chỉ sau một thời gian ngắn.

Nhận thấy những bất cập đó, nhu cầu về một công cụ hỗ trợ quản lý tài chính thông minh, trực quan và luôn sẵn sàng trên thiết bị di động iOS là một trong những hệ điều hành điện thoại mà mọi người sử dụng rất phổ biến là vô cùng cần thiết. Một ứng dụng không chỉ giúp người dùng ghi lại các con số khô khan mà còn đóng vai trò như một trợ lý tài chính, cung cấp các báo cáo trực quan và cảnh báo kịp thời để điều chỉnh hành vi tiêu dùng.

Về mặt kỹ thuật, hệ sinh thái iOS của Apple luôn nổi tiếng với sự ổn định, bảo mật và trải nghiệm mượt mà. Đặc biệt, sự ra đời của SwiftUI và mô hình MVVM đã mở ra một kỷ nguyên mới cho việc phát triển ứng dụng, có phép tạo ra các giao diện phức tạp với mã nguồn ngắn gọn và dễ bảo trì hơn. Chính vì vậy, tôi quyết định lựa chọn đề tài: "Xây dựng ứng dụng quản lý chi tiêu cá nhân - Chi Tiêu+ (Expense+)". Đề tài này là sự kết hợp giữa việc giải quyết bài toán thực tế về tài chính cá nhân và tìm hiểu về một trong những công nghệ, ngôn ngữ lập trình di động khá mới và tiên tiến nhất hiện nay.

1.2 Giới thiệu về ngôn ngữ lập trình Swift, SwiftUI và CoreData

1.2.1 Giới thiệu về ngôn ngữ lập trình Swift

1.2.1.1 Khái niệm

Swift là một ngôn ngữ lập trình mã nguồn mở, đa năng, được Apple giới thiệu lần đầu tiên tại hội nghị WWDC năm 2014. Được thiết kế để thay thế Objective-C, Swift hướng tới việc phát triển các ứng dụng cho hệ sinh thái của Apple bao gồm iOS, macOS, watchOS và tvOS. Swift kết hợp hiệu năng của các ngôn ngữ biên dịch với sự đơn giản, dễ đọc của các ngôn ngữ kịch bản.

1.2.1.2 Ưu điểm của Swift

- An toàn (Safety): Swift loại bỏ các lớp lỗi lập trình không an toàn phổ biến. Các biến luôn được khởi tạo trước khi sử dụng, mảng và số nguyên được kiểm tra tràn bộ nhớ, và quản lý bộ nhớ được thực hiện tự động.
- Tốc độ (Speed): Swift được xây dựng dựa trên trình biên dịch LLVM hiệu năng cao, giúp mã nguồn được tối ưu hóa tối đa cho phần cứng của Apple.

- Cú pháp hiện đại (Modern Syntax): Cú pháp của Swift ngắn gọn, dễ đọc và loại bỏ nhiều sự rườm rà của Objective-C (như không cần dấu chấm phẩy kết thúc câu lệnh).
- Mã nguồn mở (Open Source): Cộng đồng phát triển lớn mạnh, liên tục đóng góp và cải tiến ngôn ngữ.

1.2.1.3 Nhược điểm của Swift

- Tuổi đời còn trẻ: So với các ngon ngữ lâu đời như C++ hay Java, Swift vẫn đang trong quá trình phát triển và thay đổi, đôi khi gây khó khăn trong việc tương thích ngược giữa các phiên bản.
- Hỗ trợ đa nền tảng rất hạn chế: Mặc dù đã có thể chạy trên Linux và Windows, nhưng Swift vẫn chủ yếu mạnh mẽ nhất ở hệ sinh thái của Apple.

1.2.2 Giới thiệu về SwiftUI

1.2.2.1 Khái niệm

SwiftUI là một framework giao diện người dùng (UI) được Apple ra mắt năm 2019, cho phép các nhà phát triển xây dựng giao diện người dùng cho tất cả các nền tảng của Apple bằng cách sử dụng ngôn ngữ Swift. SwiftUI sử dụng cú pháp khai báo (declarative syntax), nghĩa là lập trình viên chỉ cần mô tả giao diện nên trông như thế nào và hoạt động ra sao, thay vì phải viết mã lệnh chi tiết để tạo và quản lý từng thành phần.

1.2.2.2 Ưu điểm của SwiftUI

- Cú pháp khai báo (Declarative): Giúp mã nguồn ngắn gọn, dễ hiểu và giảm thiểu lỗi logic khi quản lý trạng thái UI.
- Live Preview: Tính năng xem trước trực tiếp trong Xcode giúp lập trình viên thấy ngay kết quả thay đổi mã nguồn mà không cần chạy lại ứng dụng.
- Tương thích đa nền tảng: Một mã nguồn SwiftUI có thể dễ dàng điều chỉnh để chạy trên iOS, macOS, watchOS và tvOS.
- Tích hợp chặt chẽ với Swift: Tận dụng tối đa sức mạnh và tính an toàn của ngôn ngữ Swift.

1.2.2.3 Nhược điểm của SwiftUI

- Yêu cầu hệ điều hành mới: SwiftUI chỉ hỗ trợ từ iOS 13 trở lên, gây khó khăn nếu muốn hỗ trợ các thiết bị cũ.
- Cộng đồng và tài liệu: Mặc dù đang phát triển nhanh, nhưng tài liệu và các thư viện hỗ trợ vẫn chưa phong phú bằng UIKit (framework cũ).

1.2.3 Giới thiệu về CoreData

1.2.3.1 Khái niệm

Core Data là một framework của Apple dùng để quản lý lớp mô hình đối tượng (model layer objects) trong ứng dụng. Nó không chỉ là một cơ sở dữ liệu, mà là một khung làm việc

mạnh mẽ để quản lý vòng đời của đối tượng, quản lý đồ thị đối tượng (object graph), và lưu trữ dữ liệu (persistence). Core Data thường sử dụng SQLite làm kho lưu trữ bên dưới nhưng cung cấp lớp trừu tượng hóa cao hơn.

1.2.3.2 Ưu điểm của CoreData

- Hiệu năng cao: Được tối ưu hóa sâu cho các thiết bị Apple, xử lý tốt lượng dữ liệu lớn.
- Quản lý quan hệ (Relationships): Dễ dàng định nghĩa và quản lý các mối quan hệ 1-1, 1-nhiều, nhiều-nhiều giữa các thực thể.
- Tích hợp iCloud: Hỗ trợ đồng bộ dữ liệu qua CloudKit một cách tự nhiên.
- Undo/Redo: Hỗ trợ sẵn tính năng hoàn tác và làm lại các thay đổi dữ liệu.

1.2.3.3 Nhược điểm của CoreData

- Độ phức tạp (Learning Curve): Core Data có kiến trúc phức tạp và khó tiếp cận đối với người mới bắt đầu.
- Khó debug: Việc gỡ lỗi các vấn đề liên quan đến Core Data, đặc biệt là đa luồng (multi-threading), thường gặp nhiều khó khăn.

1.2.4 Sự kết hợp giữa Swift, SwiftUI và CoreData

Trong dự án "Chi Tiêu+", sự kết hợp của ba công nghệ này tạo nên một nền tảng vững chắc:

- Swift đóng vai trò là ngôn ngữ nền tảng, xử lý logic nghiệp vụ và tính toán.
- SwiftUI đảm nahatj việc hiển thị giao diện người dùng hiện đại, phản hồi nhanh chóng với các thay đổi dữ liệu nhờ cơ chế Blinding và State.
- CoreData đóng vai trò là bộ nhớ, lưu trữ an toàn các giao dịch, danh mục, ngân sách và mục tiêu của người dùng ngay trên thiết bị.

1.3 Mục tiêu của đề tài

Mục tiêu chính của đề tài là xây dựng một sản phẩm phần mềm hoàn chỉnh, có tính ứng dụng cao và giao diện thân thiện với người dùng Việt Nam. Để đạt được điều đó, đề tài tập trung giải quyết hai nhóm mục tiêu chính:

- Mục tiêu sản phẩm: Xây dựng ứng dụng phản lý chi tiêu với đầy đủ các tính năng như: Quản lý giao dịch, Thống kê báo cáo, Ngân sách, Mục tiêu tiết kiệm và Bảo mật ứng dụng.
- Mục tiêu kỹ thuật: Hiểu biết nắm rõ về ngôn ngữ Swift, framework SwiftUI và cỡ sở dữ liệu CoreData.

1.4 Phạm vi nghiên cứu

- Không gian: Hệ điều hành iOS - iPhone.
- Công nghệ: Swift 5.9+, SwiftUI, CoreData, Xcode 26.

- Nghiệp vụ: Quản lý tài chính các nhân (Thu/Chi, Ngân sách, Mục tiêu, Báo cáo).

1.5 Đối tượng sử dụng và ý nghĩa thực tiễn

1.5.1 Đối tượng sử dụng

Ứng dụng "Chi Tiêu+" được thiết kế hướng tới nhóm đối tượng người dùng rộng rãi, không giới hạn độ tuổi hay nghề nghiệp, nhưng tập trung chủ yếu vào:

- Sinh viên và người mới đi làm: Nhóm đối tượng thường có nguồn thu nhập hạn chế hoặc chưa ổn định, cần một công cụ để kiểm soát chặt chẽ từng khoản chi tiêu nhỏ nhất nhằm đảm bảo cân đối sinh hoạt phí.
- Người làm văn phòng và hộ gia đình trẻ: Những người có thu nhập ổn định nhưng có nhiều khoản chi phức tạp (điện, nước, con cái, mua sắm...), cần công cụ để hoạch định ngân sách và theo dõi các mục tiêu tiết kiệm dài hạn.
- Những người yêu thích công nghệ: Nhóm người dùng mong muốn trải nghiệm các ứng dụng hiện đại, giao diện đẹp mắt và tận dụng các tính năng bảo mật sinh trắc học trên thiết bị iPhone.

1.5.2 Ý nghĩa thực tiễn

Việc nghiên cứu và phát triển ứng dụng mang lại những giá trị thiết thực trên nhiều phương diện:

- Đối với người sử dụng: Ứng dụng đóng vai trò như một "trợ lý tài chính ảo", giúp người dùng hình thành thói quen ghi chép tài chính kỷ luật. Thông qua các báo cáo trực quan, người dùng có thể nhìn nhận lại hành vi tiêu dùng của mình, từ đó cắt giảm các khoản chi lãng phí và tập trung nguồn lực cho các mục tiêu quan trọng hơn như mua nhà, đầu tư hay du lịch.
- Đối với xã hội: Góp phần thúc đẩy xu hướng "tài chính cá nhân thông minh" (Smart Personal Finance) trong cộng đồng, nâng cao nhận thức về quản lý tài chính, đặc biệt là trong giới trẻ
- Đối với bản thân sinh viên thực hiện: Đề tài là cơ hội quý báu để tìm hiểu và sử dụng một ngôn ngữ lập trình mới từ đó áp dụng vào thực tế phát triển một phần mềm di động hoạt động tốt.

CHƯƠNG 2. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

2.1 Phân tích yêu cầu hệ thống

2.1.1 Yêu cầu chức năng

Dựa trên khải sát như cầu người dùng và mục tiêu đề tài, hệ thống “Chi Tiêu+” cần đáp ứng các nhóm chức năng chính sau đây để đảm bảo khả năng quản lý tài chính cá nhân hiệu quả cho người dùng.

- Quản lý giao dịch (Transaction Management):

    - Thêm mới giao dịch với đầy đủ thông tin: Tiêu đề, số tiền, danh mục, ngày tháng, ghi chú.
    - Xem danh sách giao dịch theo ngày, tháng, năm.
    - Chỉnh sửa hoặc xóa các giao dịch đã nhập.
    - Tìm kiếm giao dịch theo từ khóa hoặc lọc theo danh mục.
- Quản lý danh mục (Category Management):
    - Hệ thống cung cấp sẵn các danh mục mặc định (Ăn uống, Di chuyển, Lương, Hóa đơn, v.v).
    - Người dùng có thể thêm danh mục mới với tên, màu sắc và biểu tượng tùy chính theo ý thích.
    - Phân loại danh mục thành hai nhóm chính: Thu nhập (Income) và Chi tiêu (Expense).
- Quản lý ngân sách (Budget Management):
    - Thiết lập ngân sách chi tiêu cho từng danh mục hoặc tổng thể.
    - Hỗ trợ các chu kỳ ngân sách: Hàng tháng, quý, năm.
    - Tính năng "Rollover" - Chuyển số dư ngân sách còn dư đến kỳ tiếp theo.
    - Cảnh báo chi tiêu vượt quá ngưỡng cho phép.
- Mục tiêu tiết kiệm (Savings Goals):
    - Tạo các mục tiết kiệm với số tiền mục tiêu và hạn chót.
    - Theo dõi tiến độ tiết kiệm thông qua thanh tiến trình trực quan.
    - Ghi nhận các khoản tiền nạp vào hoặc rút ra từ mục tiêu.
- Giao dịch định kỳ (Recurring Transactions):
    - Tự động tạo các giao dịch lặp lại.
    - Hỗ trợ các tần suất: Hàng này, tuần, tháng, năm.
    - Nhắc nhở khi đến hạn thanh toán.
- Báo cáo thống kê (Dashboard & Reporting):
    - Biểu đồ tròn thể hiện cơ cấu chi tiêu theo danh mục.
    - Biểu đồ cột so sánh thu chi theo thời gian.
    - Tổng quan tình hình tài chính: Tổng thu, chi, số dư liện tại.

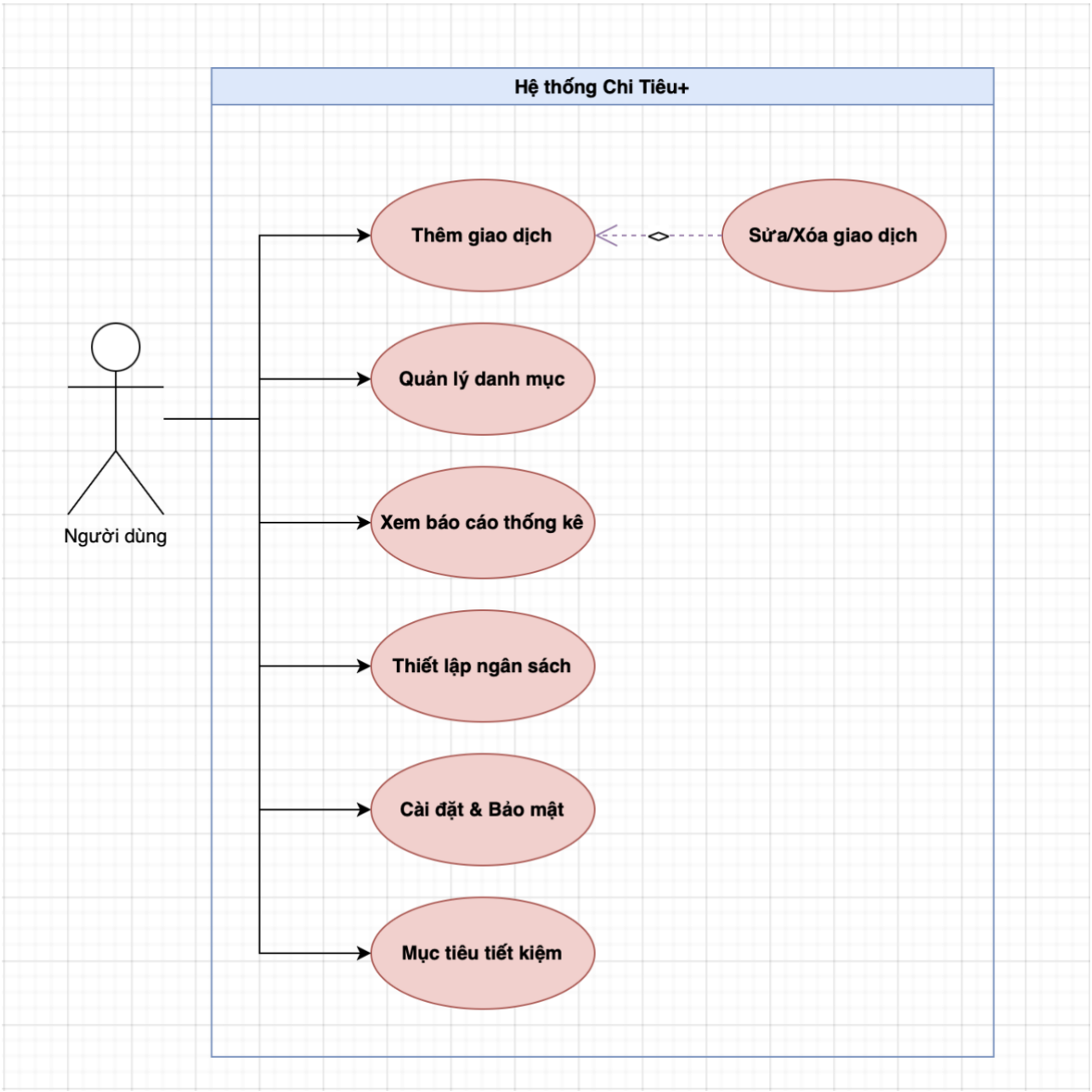
2.1.2 Yêu cầu phi chức năng

- Hiệu năng: Ứng dụng cần khởi động nhanh, phản hồi tức thì với các thao tác người dung, Sử dụng Core Data để lưu trữ cục bộ giúp truy xuất dữ liệu nhanh chóng ngay cả khi thiết bị không có mạng.

- Bảo mật: Hỗ trợ xác thực sinh trắc học (FaceID, TouchID) để bảo vệ các dữ liệu riêng tư trong app nếu người dùng cần sự riêng tư. Dữ liệu chỉ được lưu trên thiết bị người dùng không gửi về máy chủ trung gian.
- Giao diện (UX/UI): Thiết kế hiện đại, trực quan, dễ sử dụng. Hỗ trợ chế độ Dark Mode để bảo vệ mắt và tiết kiệm pin hơn.
- Độ tin cậy: Hoạt động ổn định, không bị crash đột ngột. Có cơ chế sao lưu và nhập dữ liệu.

2.2 Phân tích Use Case

Sơ đồ Use Case tổng quát mô tả các tương tác chính giữa người dùng và hệ thống, bao gồm các nhóm chức năng quản lý giao dịch, ngân sách và báo cáo.

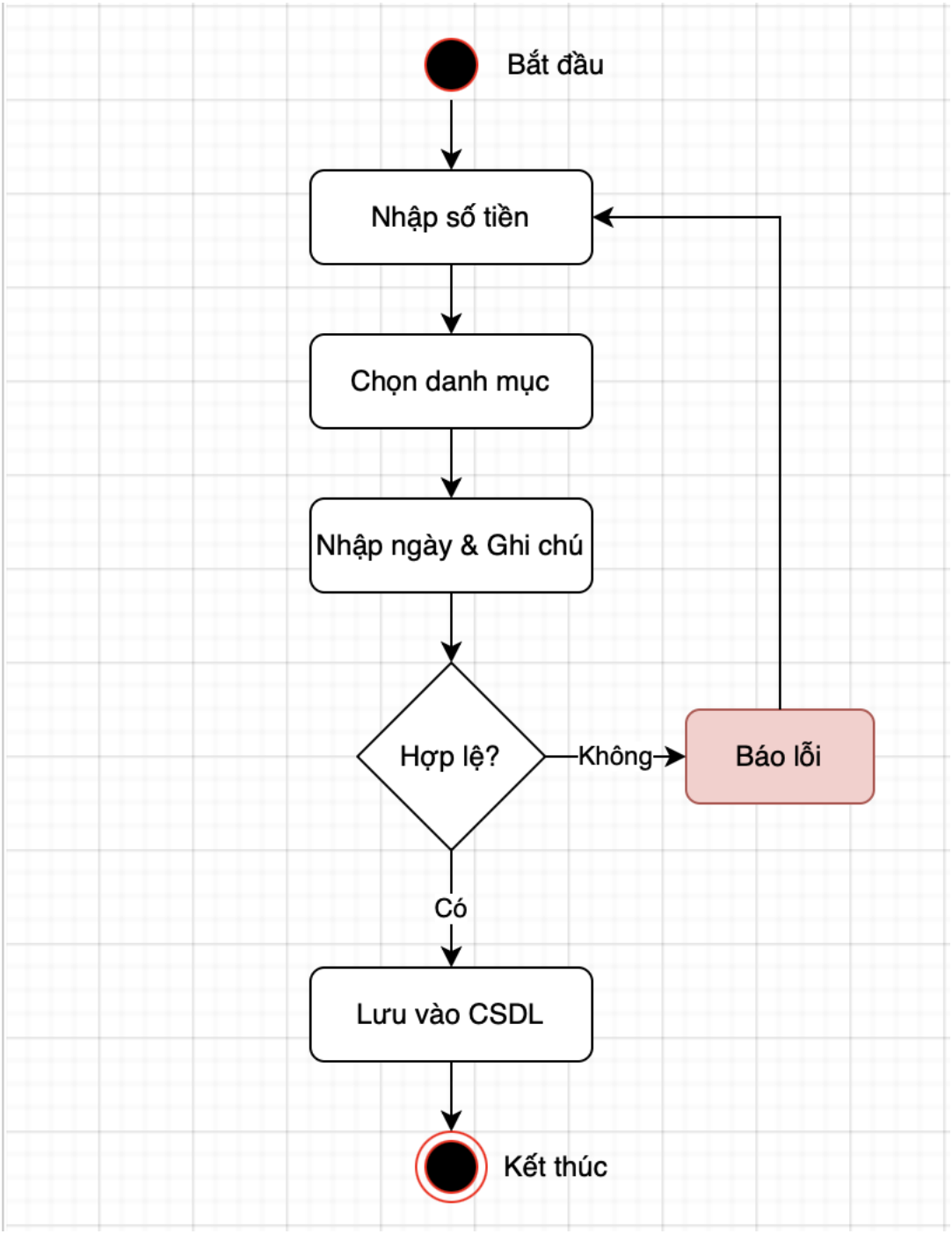


Hình 1 - Sơ đồ Use Case tổng quát của hệ thống

2.2.1 Đặc tả các Use Case chính

2.2.1.1 Use Case: Thêm giao dịch mới

- Tác nhân: Người dùng.
- Mục đích: Ghi lại một khoản thu hoặc chi mới.
- Tiền điều kiện: Người dùng đã mở ứng dụng và ở màn hình chính hoặc màn hình danh sách giao dịch
- Luồng sự kiện chính:
    - Người dùng nhấn nút "Thêm" (+) trên giao diện tabbar.
    - Hệ thống hiển thị màn hình "Thêm giao dịch".
    - Người dùng nhập số tiền giao dịch.
    - Người dùng chọn danh mục (Category) tương ứng (ví dụ: Ăn uống, Lương).
    - Người dùng chọn ngày thực hiện (mặc định là ngày hiện tại) và nhập ghi chú (nếu có).
    - Người dùng nhấn nút "Lưu".
    - Hệ thống kiểm tra tính hợp lệ của dữ liệu (số tiền > 0, danh mục đã chọn).
    - Hệ thống lưu giao dịch vào cơ sở dữ liệu Core Data.
    - Hệ thống cập nhật lại số dư tổng và biểu đồ thống kê.
    - Hệ thống thông báo thành công và quay lại màn hình trước đó.
- Luồng rẽ nhánh:
    - *Nếu người dùng không nhập số tiền hoặc nhập số âm:* Hệ thống hiển thị thông báo lỗi "Vui lòng nhập số tiền hợp lệ" và yêu cầu nhập lại.
    - *Nếu người dùng hủy bỏ:* Nhấn nút "Hủy", hệ thống không lưu dữ liệu và quay lại màn hình chính.
- Hậu điều kiện (Post-condition): Giao dịch mới được tạo, số dư tài khoản thay đổi tương ứng.

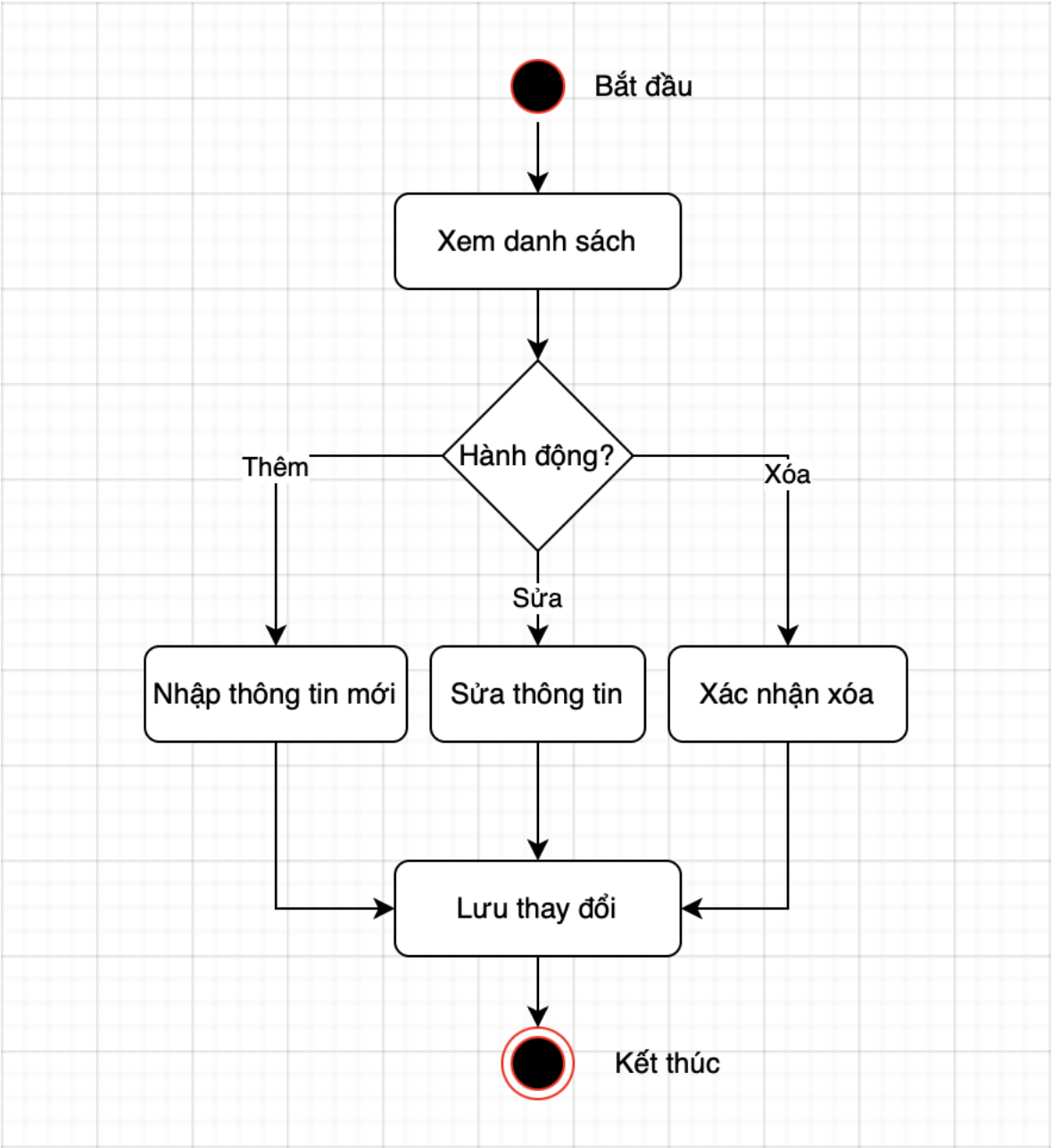


Hình 2 - Use Case thêm giao dịch mới

2.2.1.2 Use Case: Sửa giao dịch

- Tác nhân: Người dùng.
- Mục đích: Chỉnh sửa giao dịch khi đã tạo.
- Tiền điều kiện: Người dùng đã thêm giao dịch để có giao dịch có trong danh sách.
- Luồng hoạt động chính:
    - Người dùng chọn một giao dịch từ danh sách.

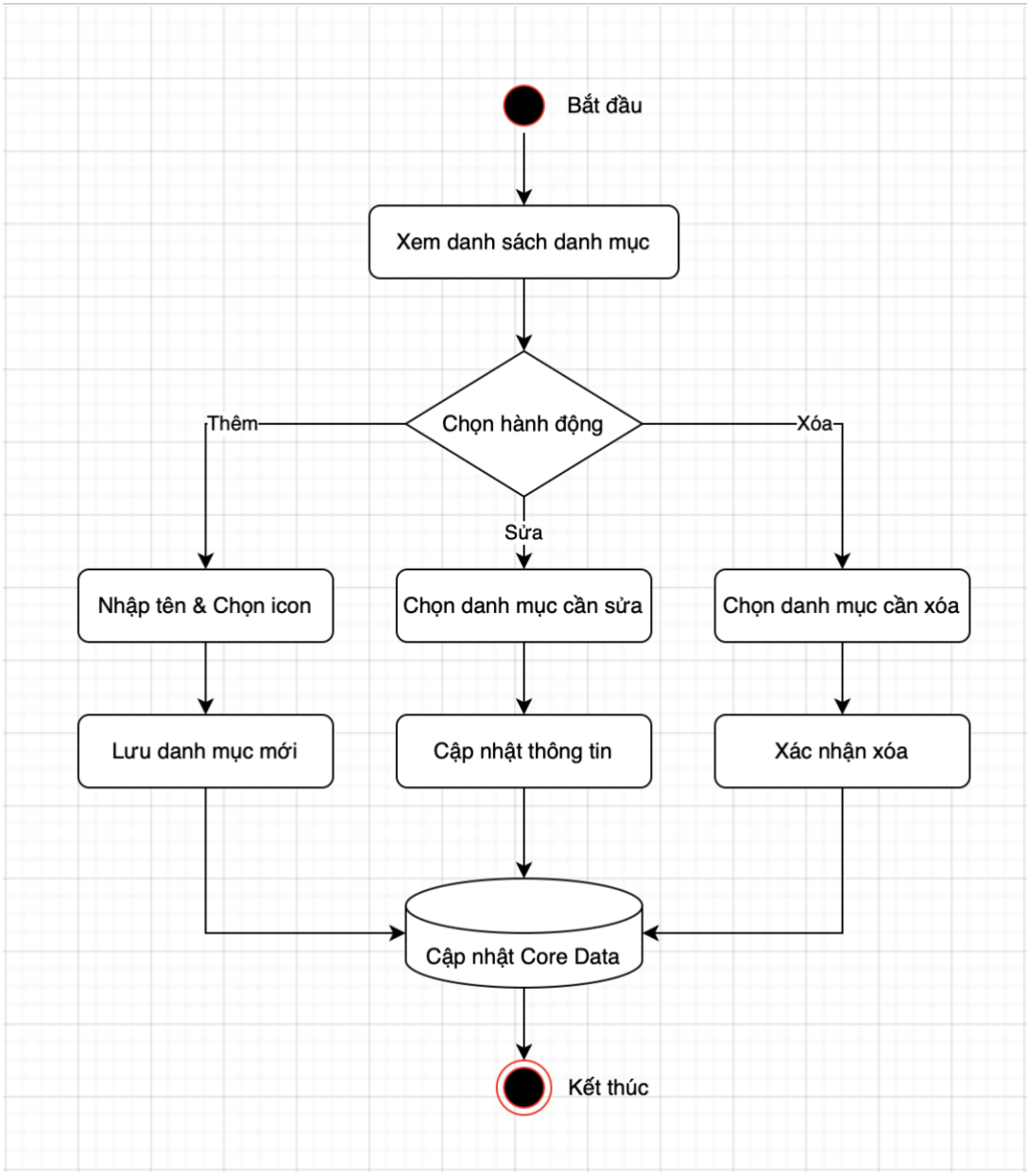
- Hệ thống hiển thị chi tiết giao dịch, người dùng có thể chọn chức năng thêm, sửa, xóa thông tin.
- Người dùng ấn lưu hoặc xóa, hệ thống cập nhật thông tin mới vào CSDL.



Hình 3 - Use Case sửa giao dịch

2.2.1.3 Use Case: Danh mục

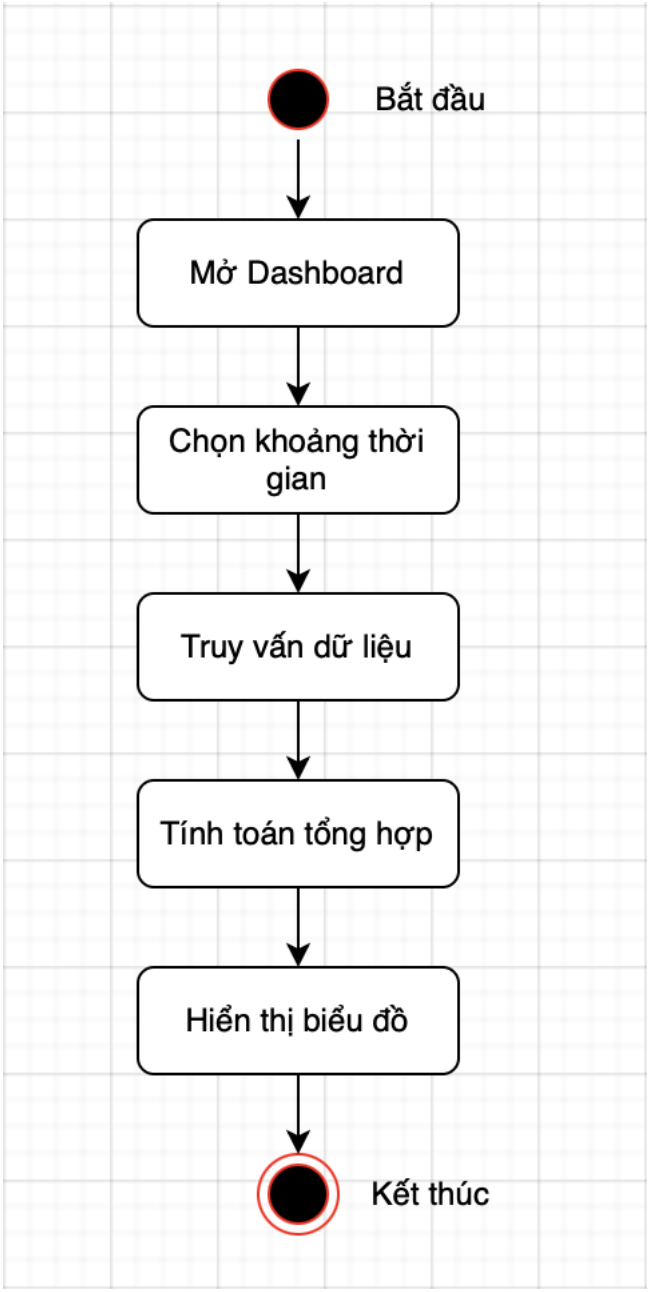
- Tác nhân: Người dùng.
- Mục đích: Thêm sửa xóa danh mục.
- Tiền điều kiện: Người dùng ở trang xem danh sách danh mục.



Hình 4 - Use Case Danh mục

2.2.1.4 Use Case: Dashboard

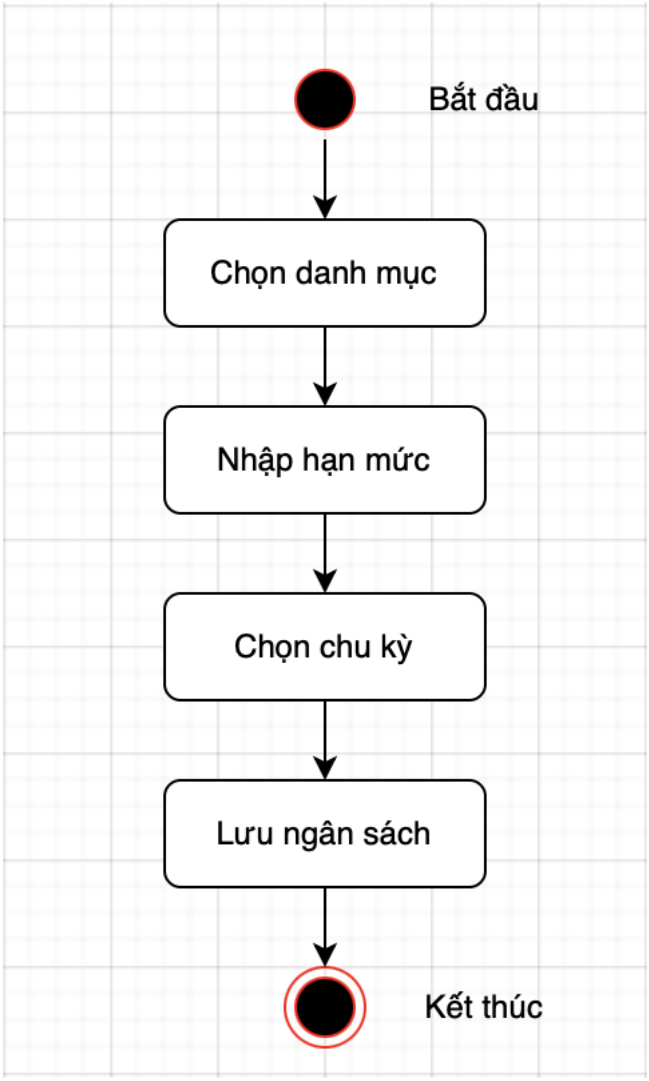
- Tác nhân: Người dùng.
- Mục đích: Xem tổng thu, chi, cơ cấu chi tiết chi tiêu theo danh mục, biểu đồ.



Hình 5 - Use Case Dashboard

2.2.1.5 Use Case: Ngân sách

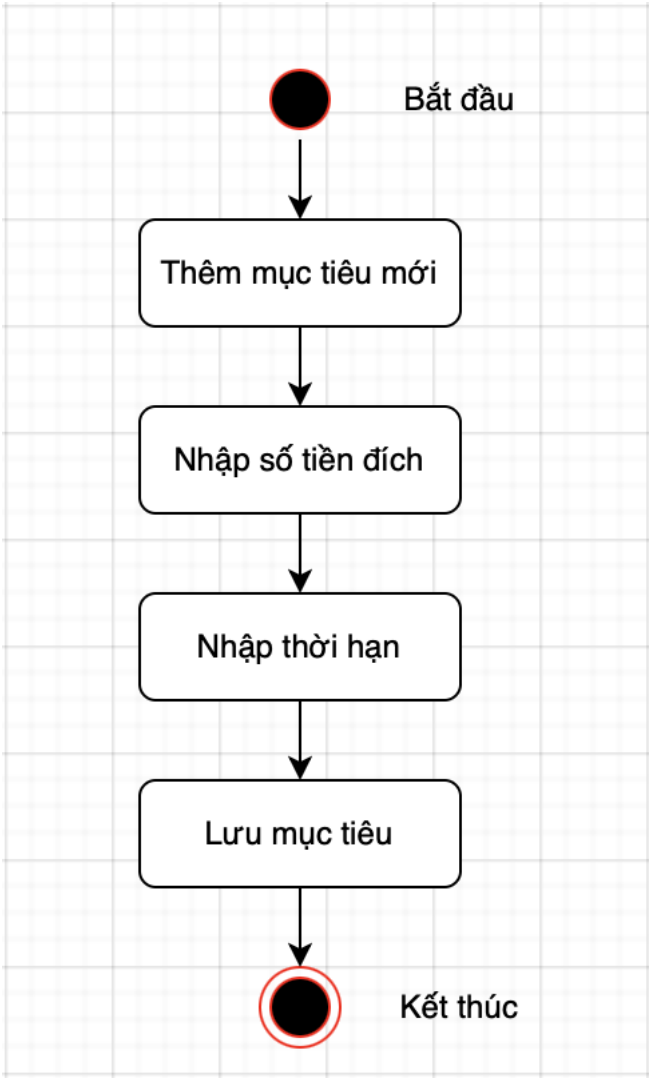
- Tác nhân: Người dùng.
- Mục đích: Đặt giới hạn chi tiêu cho một danh mục cụ thể trong một khoảng thời gian.



Hình 6 - Use Case Ngân sách

2.2.1.6 Use Case: Mục tiêu

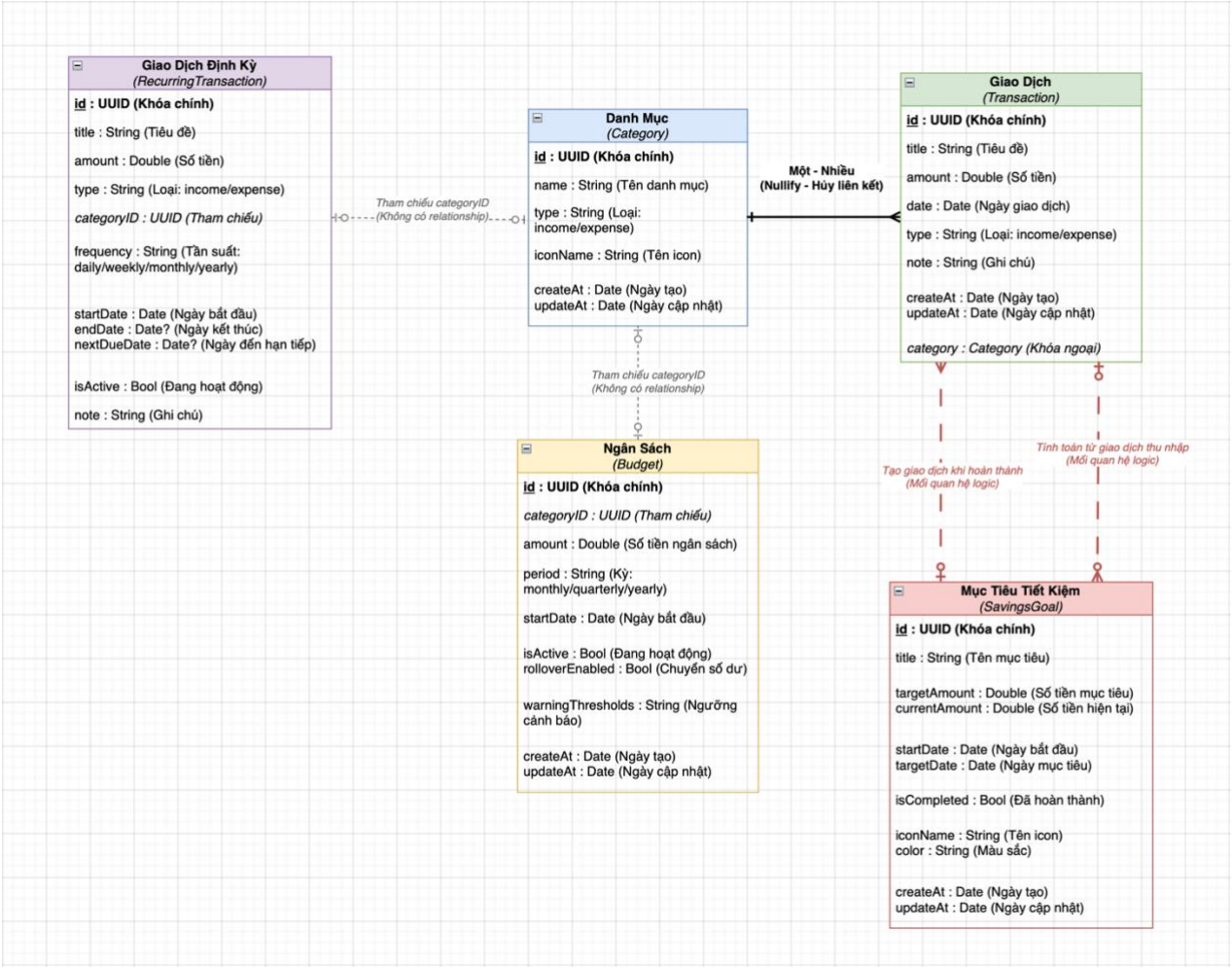
- Tác nhân: Người dùng.
- Mục tiêu: Người dùng tạo kế hoạch tiết kiệm cho một mục đích cụ thể.



Hình 7 - Use Case Mục tiêu

2.3 Thiết kế cơ sở dữ liệu

Hệ thống sử dụng Core Data làm nền tảng lưu trữ dữ liệu. Dưới đây là chi tiết các thực thể (Entities) trong mô hình dữ liệu.



Hình 8 - Sơ đồ thực thể kết hợp ERD của hệ thống

Mô hình dữ liệu bao gồm 5 thực thể chính:

- Transaction (Giao dịch): Lưu trữ các khoản thu chi, Có quan hệ N-1 với Category.
- Category (Danh mục): Quản lý loại hình thu chi, là trung tâm kết nối với Transaction, Budget và Recurring Transaction.
- Budget (Ngân sách): Thiết lập giới hạn chi tiêu cho từng danh mục (liên kết lỏng qua categoryID).
- Recurring Transaction (Giao dịch định kỳ): Quản lý các giao dịch lặp lại, tham chiếu đến danh mục qua categoryID.
- SavingsGoal (Mục tiêu tiết kiệm): Hoạt động độc lập để theo dõi tiến độ tích lũy tài chính.

2.3.1 Chi tiết các bảng dữ liệu

2.3.1.1 Bảng Category (Danh mục)

Lưu trữ thông tin về các nhóm thu chi.

Thuộc tính	Kiểu dữ liệu	Mô tả
id	UUID	Khóa chính, định danh duy nhất.
name	String	Tên danh mục (VD: Ăn uống, Lương,...).
type	String	Loại danh mục (“income” hoặc “expense”).
color	String	Mã màu hiển thị (Hex code).
icon	String	Tên icon (SF Symbols).
createAd	Date	Ngày tạo.

Bảng 1. Bảng Category

2.3.1.2 Bảng Transaction (Giao dịch)

Lưu trữ các khoản thu chi phát sinh.

Thuộc tính	Kiểu dữ liệu	Mô tả
id	UUID	Khóa chính, định danh duy nhất.
amount	Double	Số tiền giao dịch.
date	Date	Ngày thực hiện giao dịch.
note	String	Ghi chú chi tiết.
type	String	Loại giao dịch.
categoryID	UUID	Khóa ngoại, liên kết với bảng Category.

Bảng 2. Bảng Transaction

2.3.1.3 Bảng Budget (Ngân sách)

Quản lý hạn mức chi tiêu.

Thuộc tính	Kiểu dữ liệu	Mô tả
id	UUID	Khóa chính, định danh duy nhất.
amount	Double	Số tiền ngân sách giới hạn.
period	String	Chu kỳ (monthly, quarterly, yearly).
startDate	Date	Ngày bắt đầu áp dụng.
warningThreshholds	String	Các mốc cảnh báo (VD: "[80,90,100]").
rolloverEnable	Bool	Cho phép chuyển số dư sang kỳ sau không?.
categoryID	UUID	Khóa ngoại, liên kết với bảng Category.

Bảng 3. Bảng Budget

2.3.1.4 Bảng SavingsGoal (Mục tiêu tiết kiệm)

Quản lý các kế hoạch tích lũy tài chính.

Thuộc tính	Kiểu dữ liệu	Mô tả
id	UUID	Khóa chính, định danh duy nhất.
title	String	Tên mục tiêu.
targetAmount	Double	Số tiền cần tiết kiệm.
currentAmount	Double	Số tiền hiện có.
targetDate	Date	Ngày dự kiến hoàn thành.
isCompleted	Bool	Trạng thái hoàn thành.

Bảng 4. Bảng Savings Goal

2.3.1.5 Bảng RecurringTransaction (Giao dịch định kỳ)

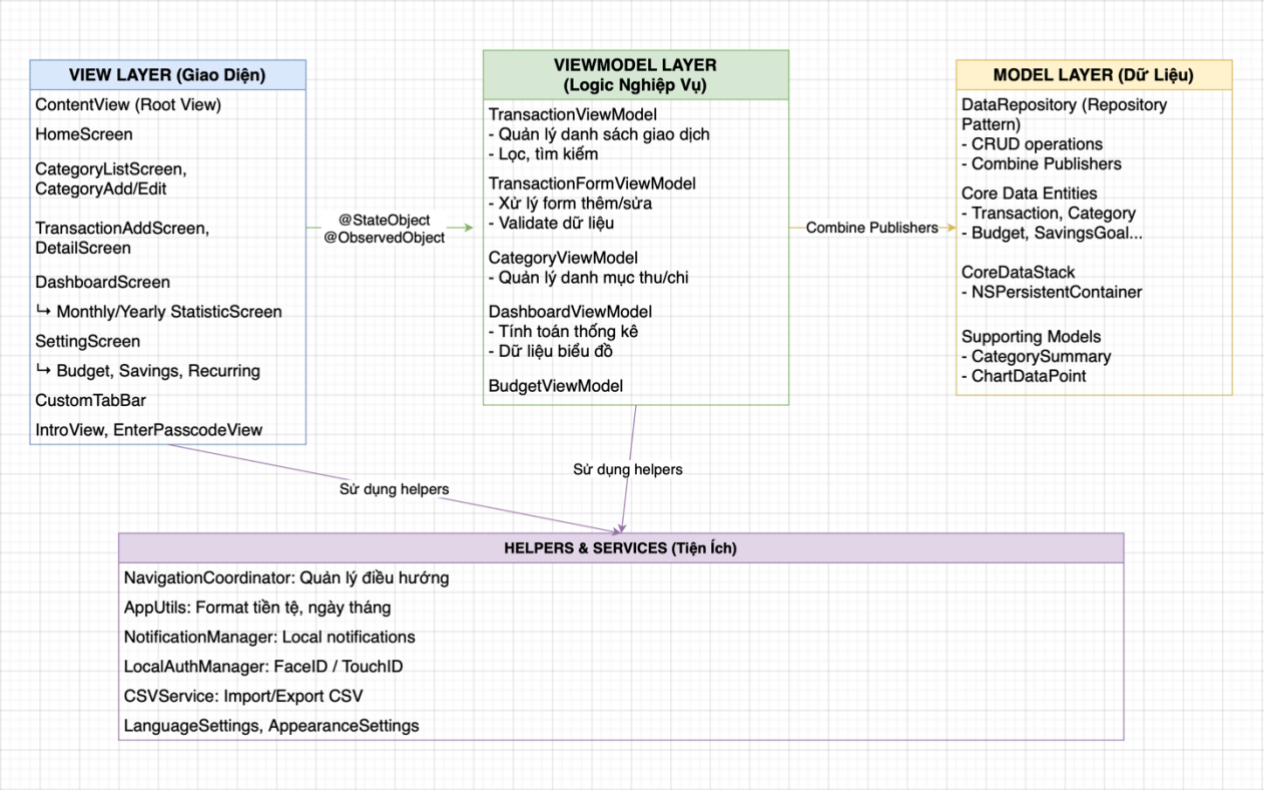
Quản lý các giao dịch lặp lại tự động.

Thuộc tính	Kiểu dữ liệu	Mô tả
id	UUID	Khóa chính, định danh duy nhất.
title	String	Tên giao dịch định kỳ.
amount	Double	Số tiền.
frequency	String	Tần suất (daily, weekly, monthly, yearly).
nextDueDate	Date	Ngày đến hạn tiếp theo.
isActive	Bool	Trạng thái hoạt động.

Bảng 5. Bảng Recurring Transaction

2.4 Thiết kế kiến trúc hệ thống

Ứng dụng "Chi Tiêu+" được xây dựng dựa trên kiến trúc MVVM (Model - View - ViewModel), đây là kiến trúc tiêu chuẩn được Apple áp dụng và khuyến nghị khi phát triển ứng dụng với SwiftUI.



Hình 9 - Mô hình kiến trúc MVVM của ứng dụng

Các thành phần chính trong kiến trúc như sau:

- View Layer (Giao diện): Bao gồm các màn hình SwiftUI như ContentView, DashboardScreen, TransactionAddScreen,... Các View này chỉ chịu trách nhiệm hiển thị và chuêynr tiếp hành động của người dùng.

- ViewModel Layer (Logic nghiệp vụ): Chứa các lớp quản lý trang thái như TransactionViewModel, CategoryViewModel, DashboardViewModel,... Chúng xử lý logic nghiệp vụ, validate dữ liệu và giáo tiếp với Model.
- Model Layer (Dữ liệu): Bao gồm DataRespository (Repository Pattern) để trừu tượng hóa việc truy xuất dữ liệu và các thực thể Core Data.
- Helper & Services (Tiện ích): Các lớp hỗ trợ như NavigationCoordinator (Điều hướng), LocalAuthManager (Bảo mật FaceID),...

Cơ chế hoạt động và kết nối:

- Quản lý trạng thái (State Management): Sử dụng các Property Wrappers của SwiftUI
    - @StateObject: Khởi tạo ViewModel trong View cha.
    - @ObservedObject: Truyền ViewModel xuống các View con để lắng nghe thay đổi.
    - @Published: Đánh dấu các biến trong ViewModel để tự động thông báo cho View cập nhật lại giao diện khi dữ liệu thay đổi.
- Lập trình phản ứng (Rẻtive Programming): Framework Combine được sử dụng trong lớp Model và ViewModel để xử lý các luồng dữ liệu bất đồng bộ. Ví dụ: Khi DataRespository tải xong dữ liệu từ Core Data, nó sẽ phát tín hiệu qua Publisher để ViewModel nhận và cập nhật UI.
- Tiện ích & dịch vụ (Helpers & Services): Các class Helper được thiết kế theo mô hình Singleton hoặc Static để dễ dàng gọi từ bất kì đâu (View hoặc ViewModel) mà không phát vỡ cấu trúc MVVM. Ví dụ: AppUtils.fornatCurrency (amount) được dùng trực tiếp trong View để hiển thị số tiền.

CHƯƠNG 3. CÀI ĐẶT VÀ KẾT QUẢ CÀI ĐẶT

3.1 Môi trường phát triển được cài dặt sẵn

Ứng dụng được phát triển và kiểm thử trên môi trường sau:

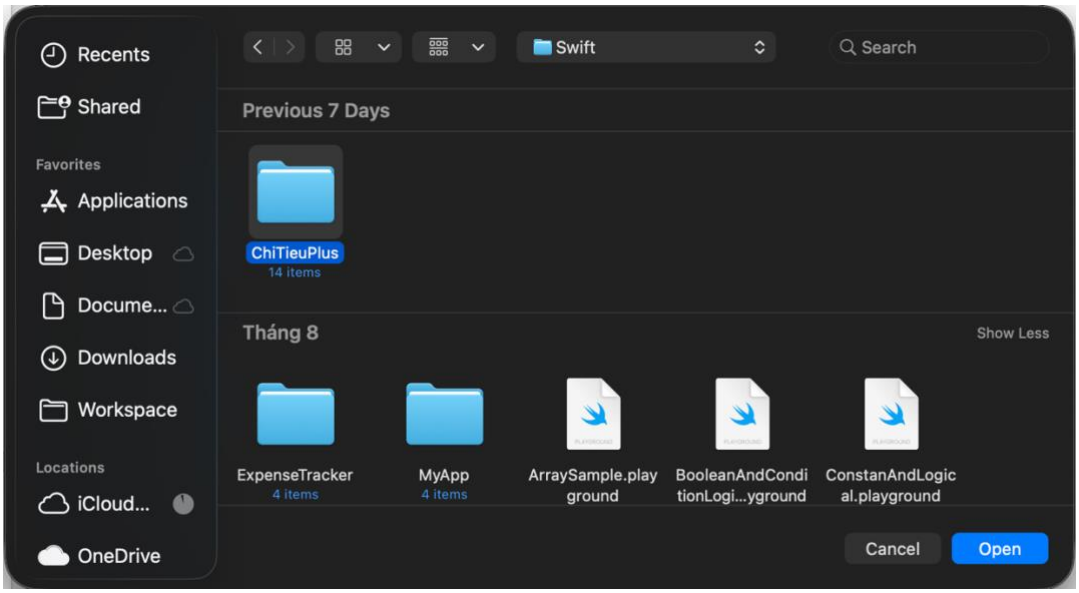
- IDE: Xcode 15.0 trở lên.
- Ngôn ngữ lập trình: Swift 5.9.
- Framework UI: SwiftUI.
- Cơ sở dữ liệu: Core Data.
- Hệ điều hành mục tiêu: íO 16.0 trở lên.
- Thiết bị kiểm thử và máy ảo: iPhone 13 Pro, iPhone 11 (Thiết bị thật).

3.2 Cài đặt và triển khai

Để cài đặt và chạy ứng dụng trên môi trường phát triển, cần thực hiện các bước sau:

Bước 1: Tải mã nguồn.

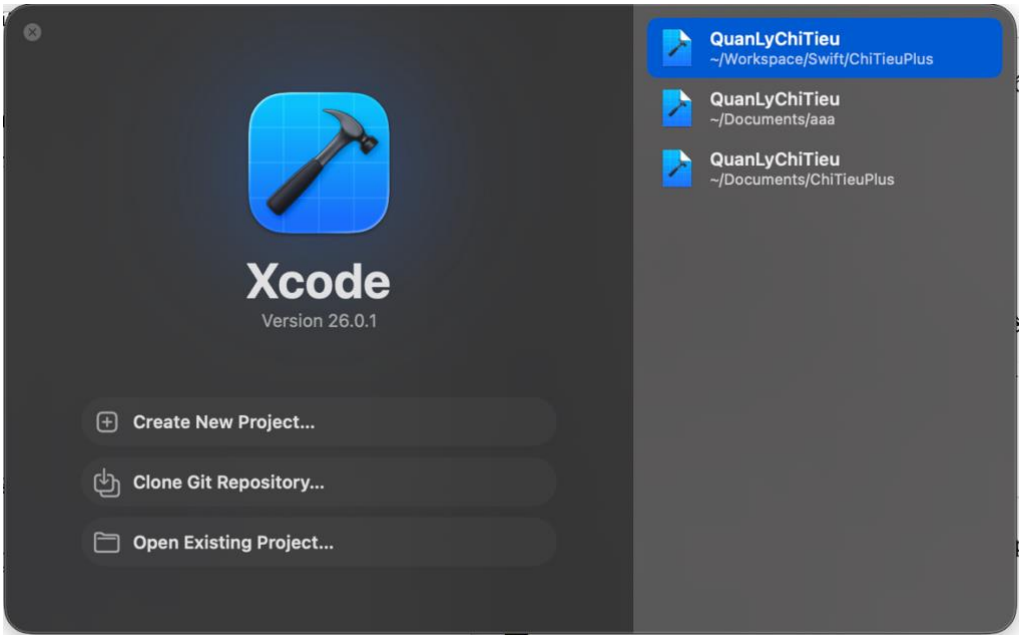
- Clone repo https://github.com/tangoctai2004/ChiTieuPlus.git về máy.



Hình 10 - Clone repo về máy

Bước 2: Mở project

- Mở Xcode và open thư mục vừa clone về.

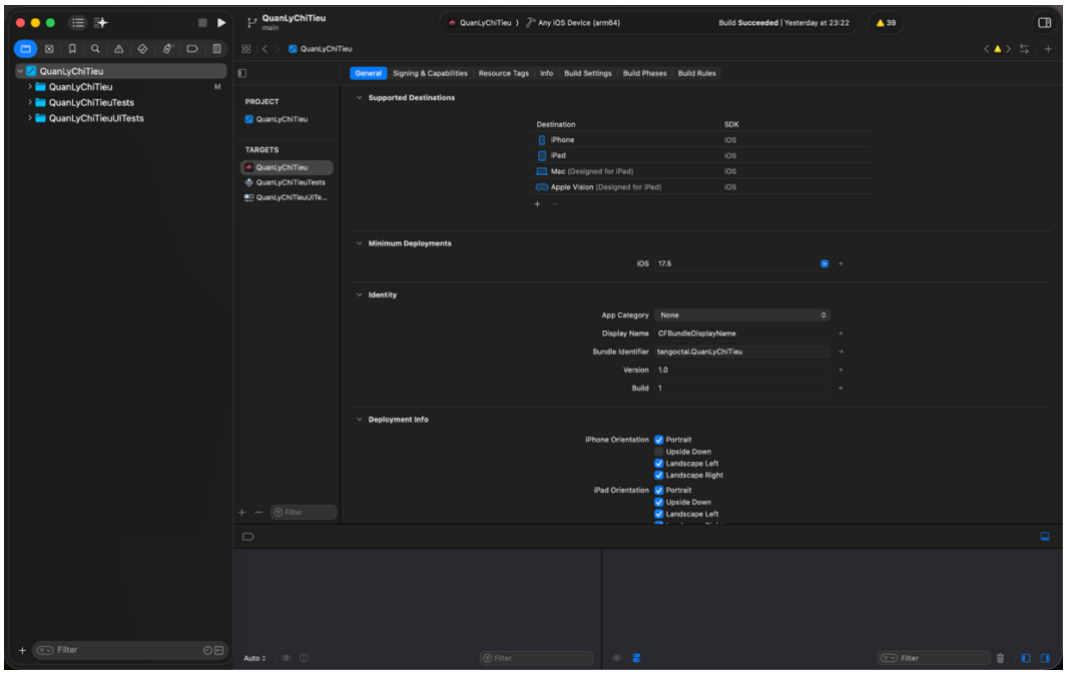


Hình 11 - Open Exiting Project ChiTieuPlus

Bước 3: Đợi project Indexing | Processing files.

Hình 12 - Indexing

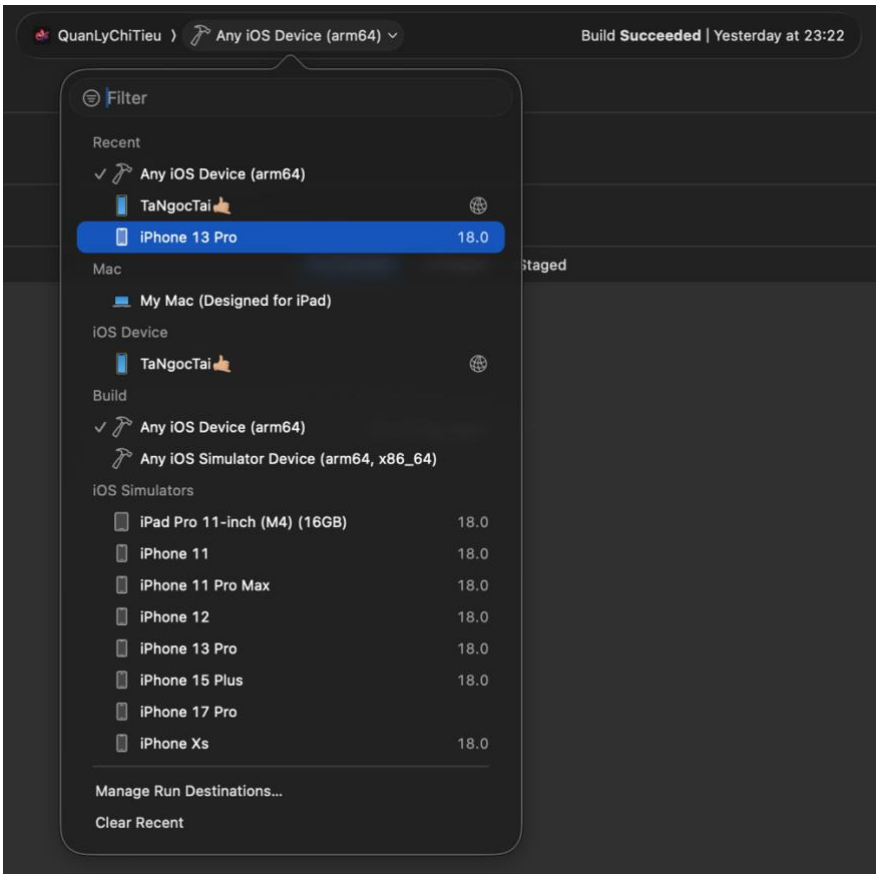
- Sau khi index xong ta mở được project hoàn chỉnh.



Hình 13 - Giao diện Xcode với project được cài đặt hoàn chỉnh.

Bước 4: Chọn thiết bị.

- Chọn thiết bị giả lập (Simulator) mong muốn (ví dụ: iPhone 13 Pro) hoặc kết nối thiết bị thật.



Hình 14 - Chọn thiết bị để chạy code

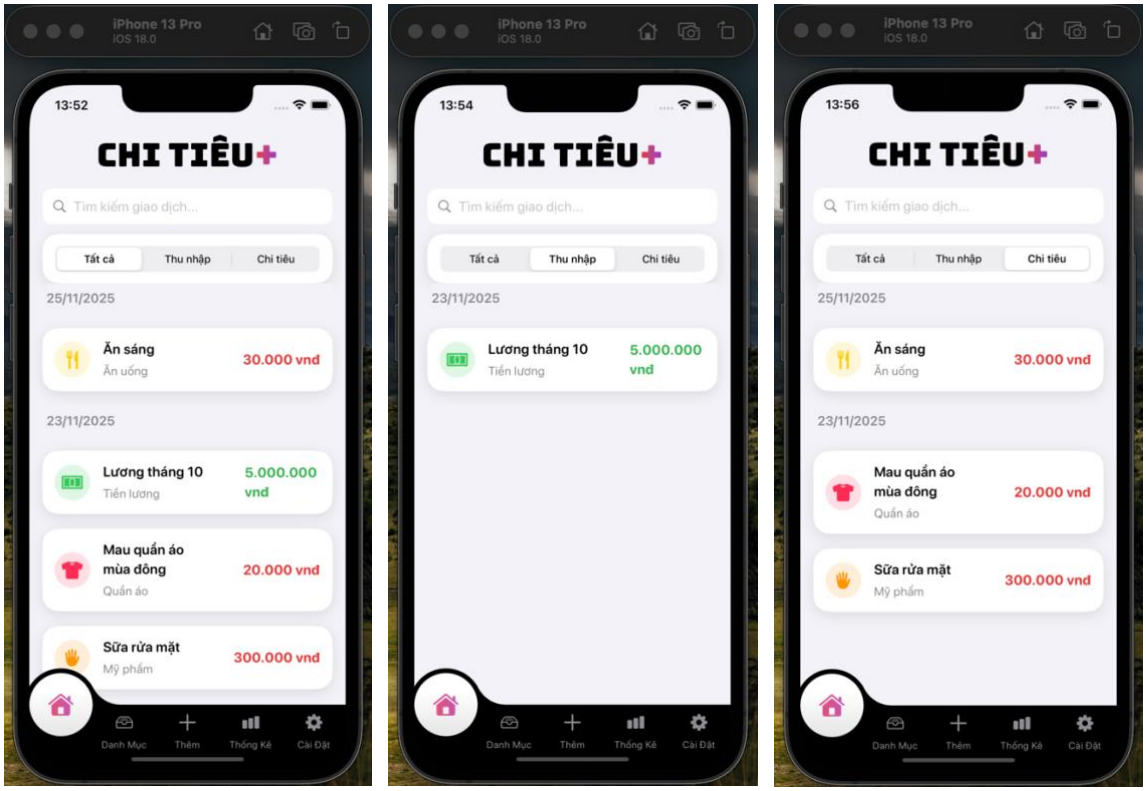
Bước 5: Chạy ứng dụng.

- Ấn tổ hợp phím Cmd + R hoặc ấn nút Run (hình tam giác) trên thanh công cụ để biên dịch và chạy ứng dụng.
- Sau khi Build Success thì Simulator hoặc điện thoại được chọn sẽ có giao diện như ảnh dưới.

Hình 15 - Giao diện App Chi Tiêu+ sau khi chạy thành công

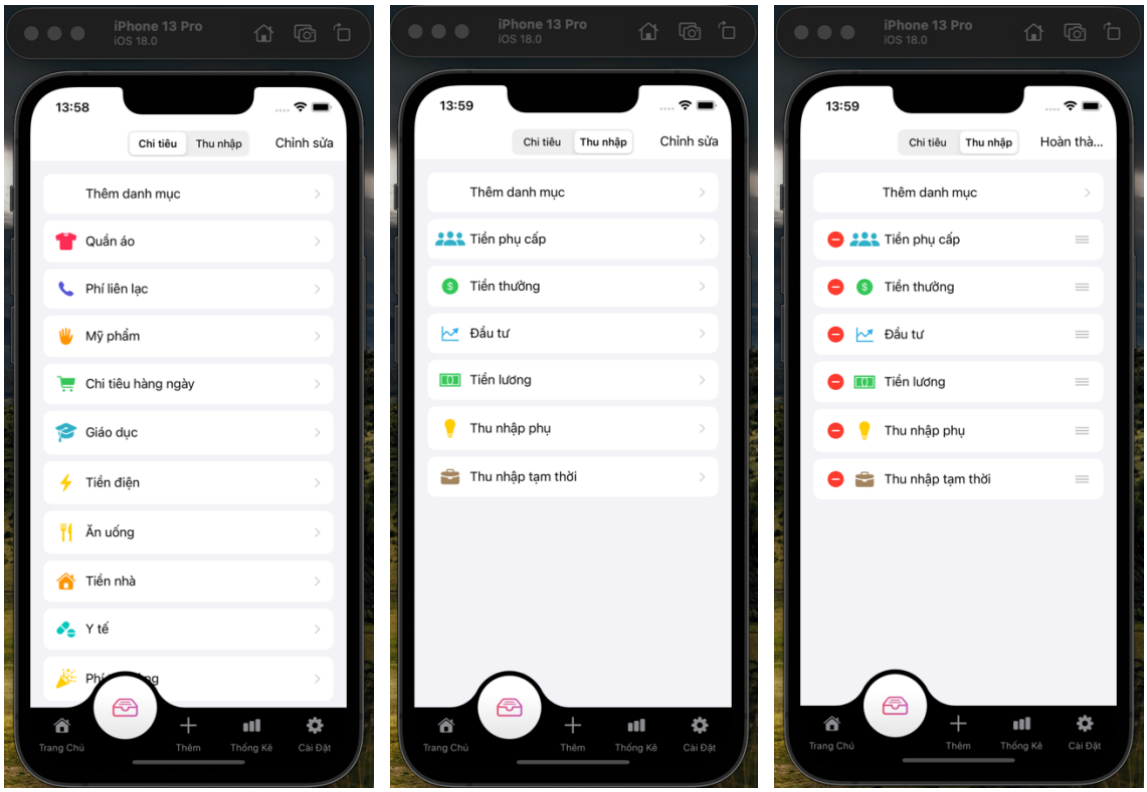
3.3 Kết quả giao diện

3.3.1 Màn hình HomeScreen

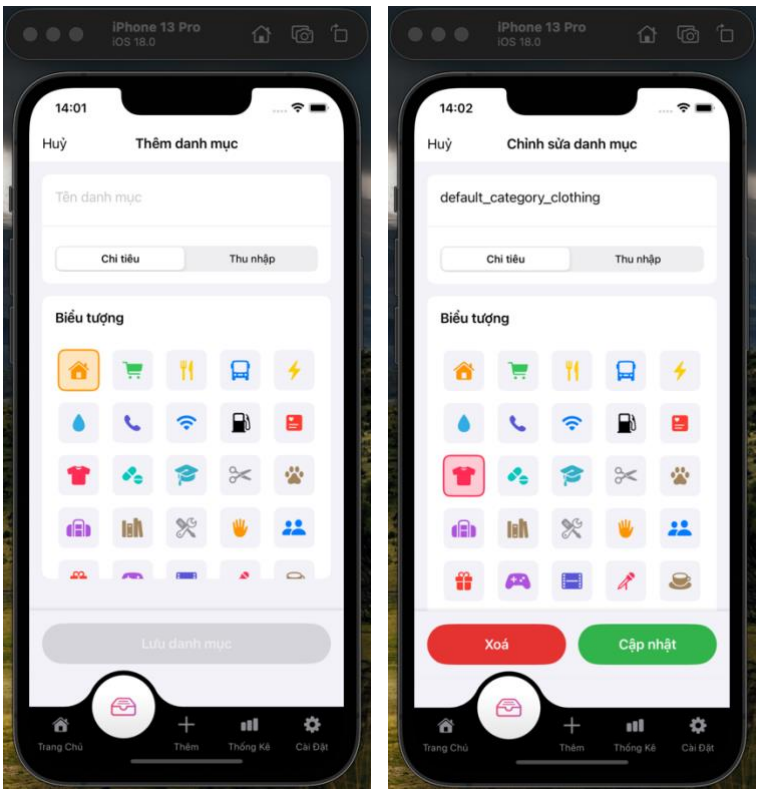


Hình 16 - Giao diện Home Screen

3.3.2 Màn hình Category

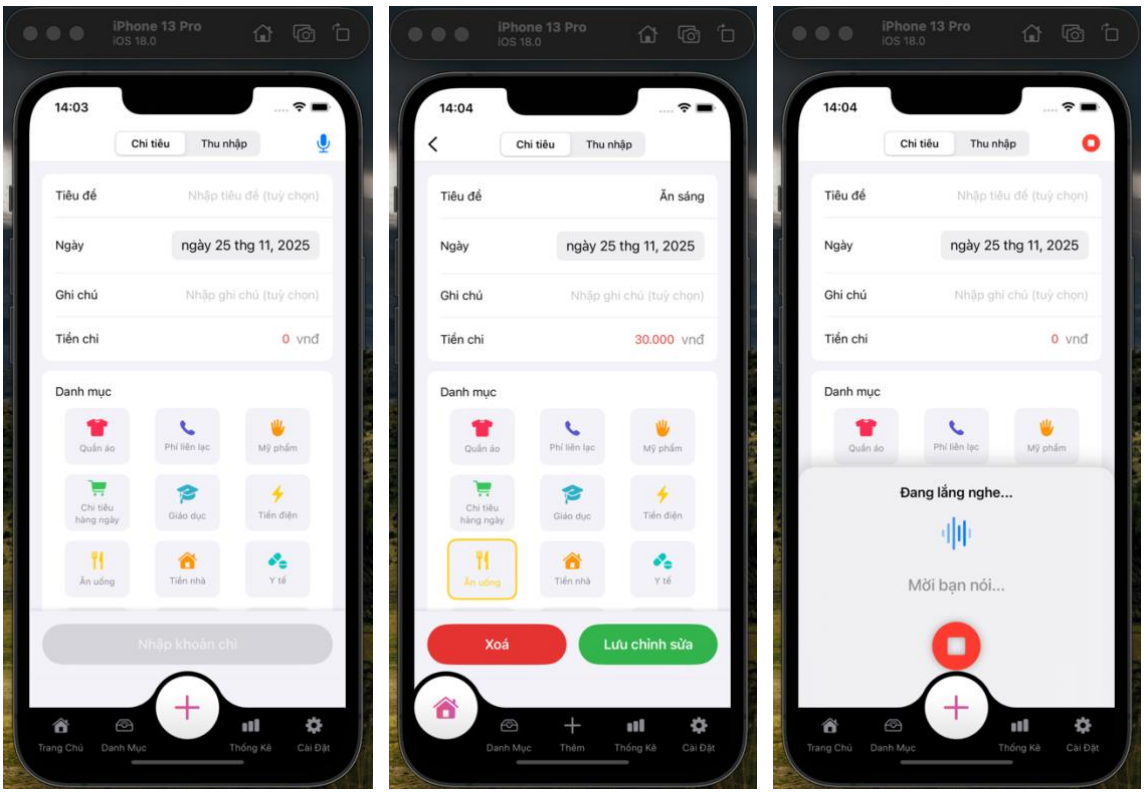


Hình 17 - Giao diện Category



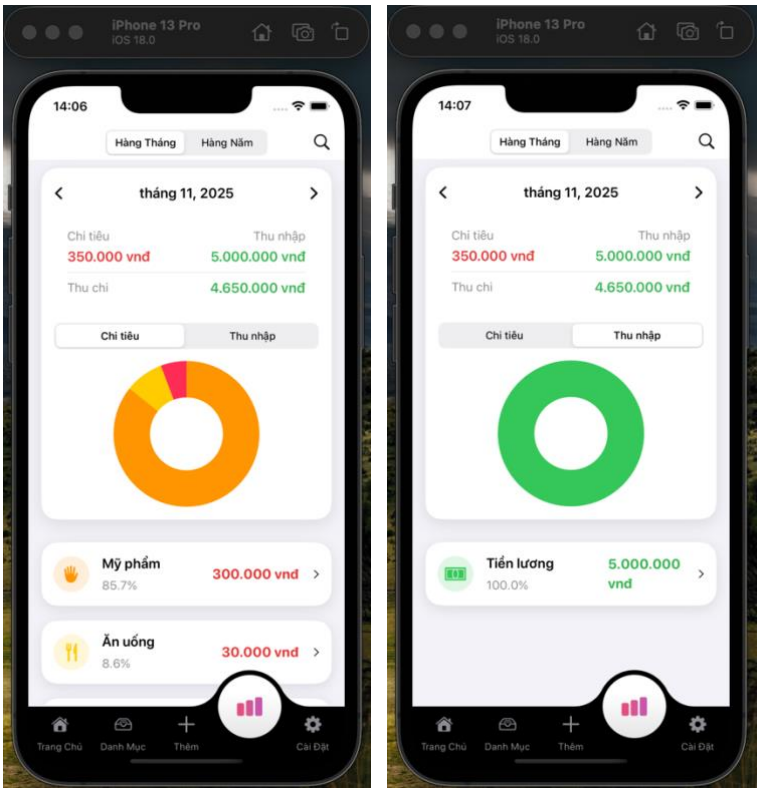
Hình 18 - Giao diện thêm, sửa, xóa Category

3.3.3 Màn hình Transaction

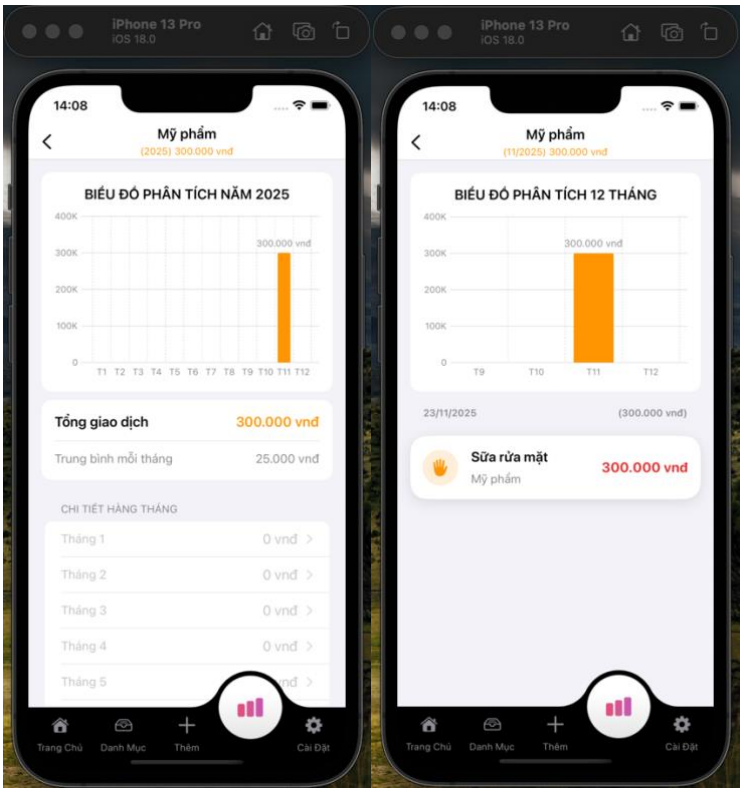


Hình 19 - Giao diện thêm, sửa, xóa Transaction

3.3.4 Màn hình Dashboard

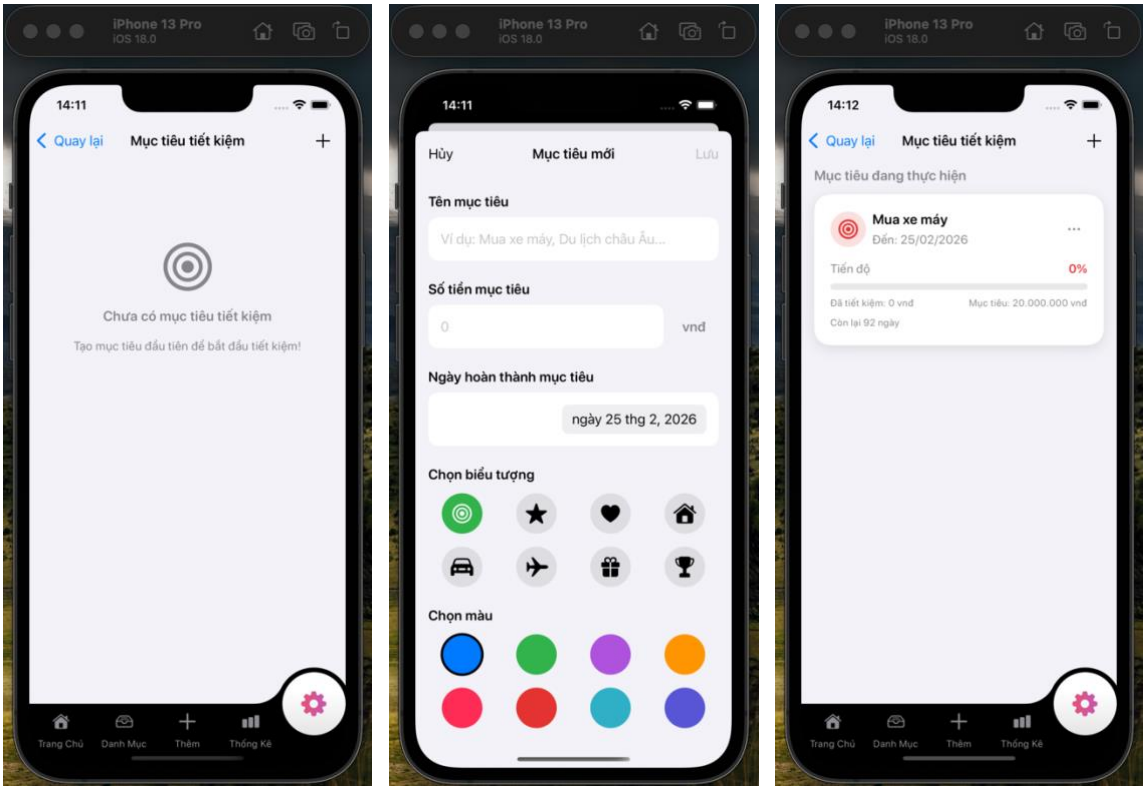


Hình 20 - Giao diện thu/chi theo từng tháng



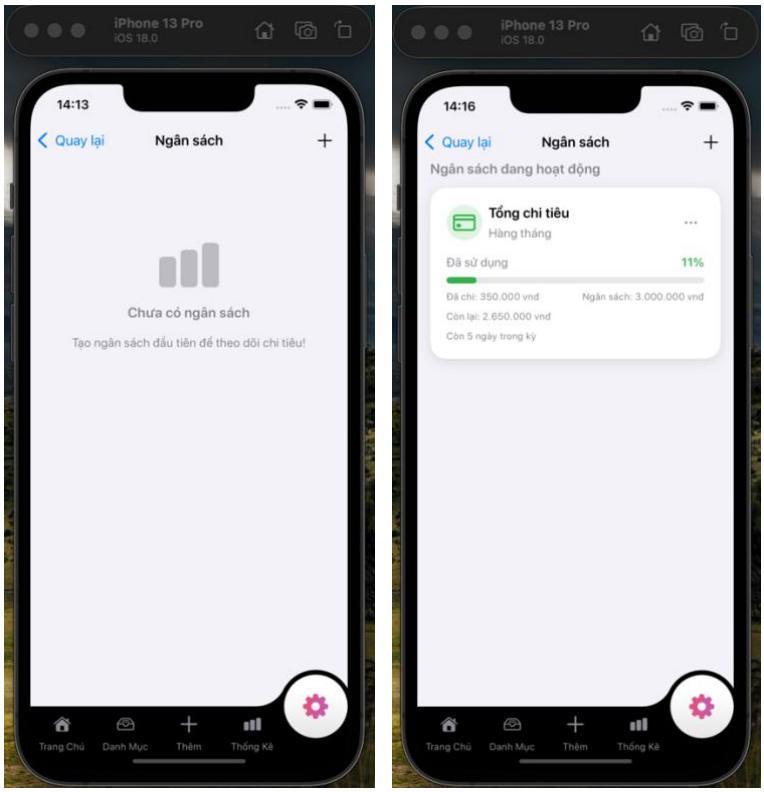
Hình 21 - Giao diện biểu đồ phân tích Category theo tháng và năm

3.3.5 Màn hình quản lý mục tiêu

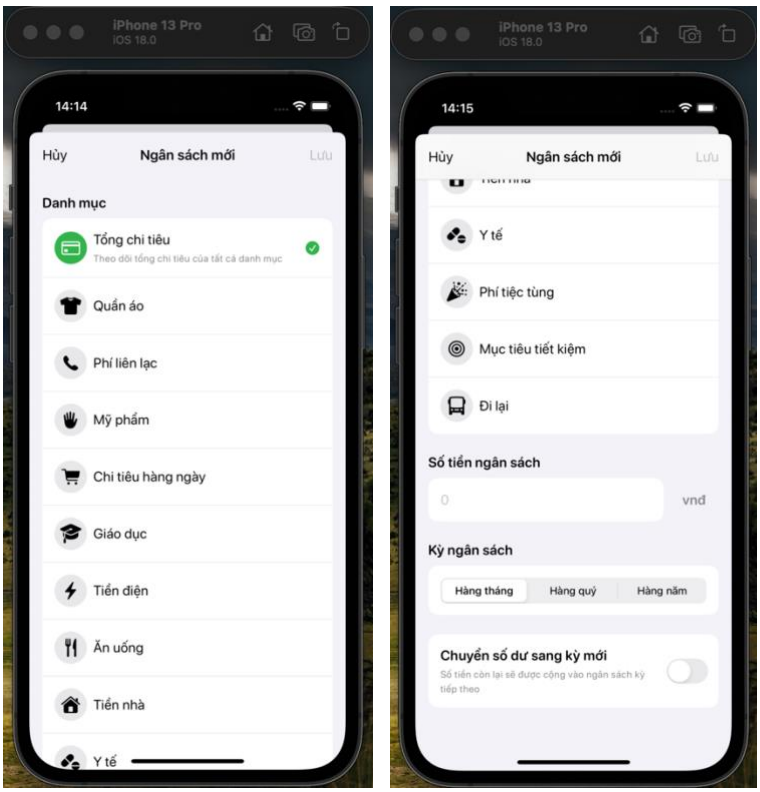


Hình 22 - Giao diện quản lý mục tiêu

3.3.6 Màn hình quản lý ngân sách

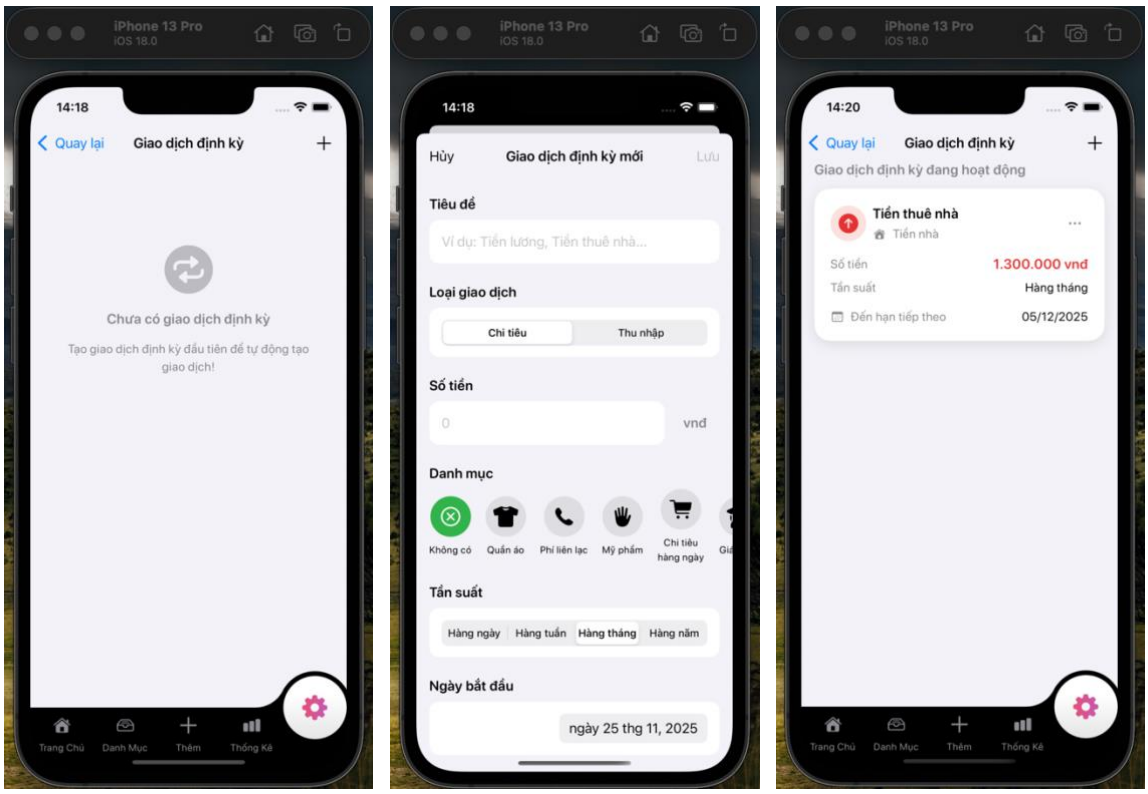


Hình 23 - Giao diện quản lý ngân sách



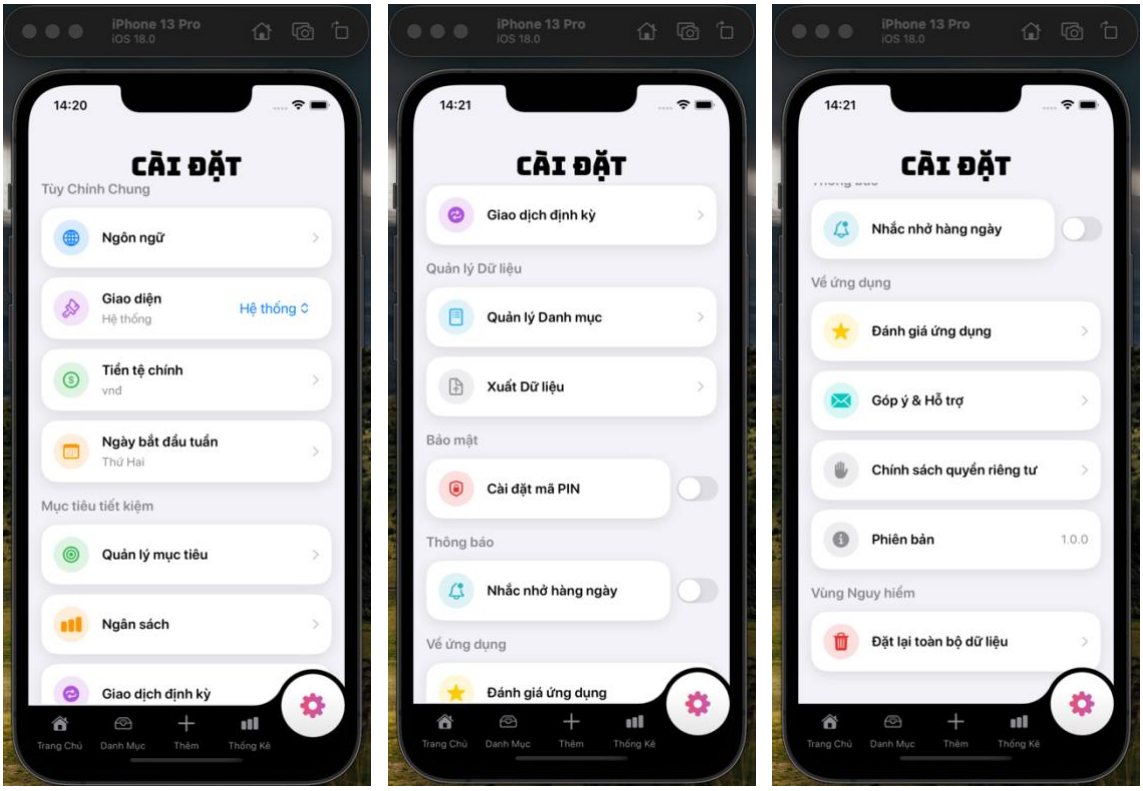
Hình 24 - Giao diện thêm ngân sách

3.3.7 Màn hình giao dịch định kỳ



Hình 25 - Giao diện giao dịch định kỳ

3.3.8 Màn hình cài đặt



Hình 26 - Giao diện cài đặt

3.4 Các chức năng chính

Sau quá trình hiện thực, ứng dụng đã hoàn thiện các chức năng cốt lõi sau:

- Quản lý thu chi: Cho phép thêm, sửa, xóa giao dịch nhanh chóng. Hỗ trợ tìm kiếm và lọc giao dịch theo nhiều tiêu chí.
- Ngân sách thông minh: Tự động cảnh báo khi chi tiêu vượt quá ngưỡng thiết lập. Hỗ trợ chuyển số dư ngân sách còn lại sang tháng sau (Rollover).
- Giao dịch định kỳ: Tự động tạo các giao dịch lặp lại (tiền nhà, tiền điện, lương...) giúp tiết kiệm thời gian nhập liệu.
- Báo cáo trực quan: Biểu đồ tròn và biểu đồ cột giúp người dùng dễ dàng nhận diện xu hướng chi tiêu.
- Bảo mật & Tiện ích: Hỗ trợ khóa ứng dụng bằng FaceID và xuất dữ liệu ra file CSV để lưu trữ cá nhân.

CHƯƠNG 4. KẾT LUẬN

4.1 Kết quả đạt được

"Xây dựng ứng dụng Quản lý chi tiêu cá nhân trên iOS" xuất phát từ thực tế ngày nay nhằm tạo một công cụ hỗ trợ đắc lực cho mọi người trong việc theo dõi và kiểm soát tài chính cá nhân một cách hiệu quả và tiện lợi ngay trên thiết bị di động.

Những kết quả đạt được:

- Về công nghệ:
    - Tìm hiểu sâu về công nghệ lập trình di động trên nền tảng iOS: Ngôn ngữ lập trình Swift, framework giao diện SwiftUI và hệ quản trị cơ sở dữ liệu Core Data.
    - - Tìm hiểu và nắm bắt được các bước thực hiện, xây dựng một ứng dụng hoàn chỉnh dựa trên kiến trúc MVVM (Model-View-ViewModel).
    - - Biết được cách thiết kế giao diện hiện đại (Modern UI) cũng như cách tổ chức cơ sở dữ liệu cục bộ hiệu quả.
- Về cài đặt chương trình:
    - Cho phép người dùng ghi chép các khoản thu chi hàng ngày một cách nhanh chóng.
    - Quản lý danh mục chi tiêu, thiết lập ngân sách và theo dõi các mục tiêu tiết kiệm.
    - Tự động hóa việc tạo các giao dịch định kỳ (lặp lại).
    - Cập nhật thông tin giao dịch, số dư ví, tiến độ ngân sách,...
    - Thống kê, báo cáo tình hình tài chính qua các biểu đồ trực quan.
- Tính năng của chương trình
    - Thông tin về giao dịch, ngân sách, báo cáo… được cập nhật kịp thời, chính xác.
    - Giao diện thân thiện với người dùng, hỗ trợ chế độ tối (Dark Mode).

4.2 Lợi ích đạt được

- Đối với người dùng:
    - Có thể tra cứu lịch sử chi tiêu, cập nhật thông tin giao dịch mọi lúc mọi nơi;
    - Quản lý tài chính dễ dàng, nhanh chóng, tiết kiệm thời gian ghi chép sổ sách thủ công.
    - Có cái nhìn tổng quan về tình hình tài chính để điều chỉnh thói quen chi tiêu hợp lý.
- Đối với nhà phát triển (Sinh viên:

- Tạo một ứng dụng đơn giản, nhanh chóng và hiệu quả trong việc quản lý tài chính cá nhân.
- Hoàn thành bài tập môn học, đồng thời qua đó nâng cao trình độ lập trình Swift và tư duy thiết kế hệ thống của mình.
- Tích lũy kinh nghiệm thực tế trong quy trình phát triển phần mềm từ khâu ý tưởng đến sản phẩm hoàn thiện.

4.3 Hướng phát triển

- Phát triển thêm các chức năng của ứng dụng: đồng bộ dữ liệu qua iCloud (CloudKit) để sử dụng trên nhiều thiết bị, tích hợp AI để gợi ý chi tiêu thông minh, xuất báo cáo ra file Excel/PDF chuyên sâu,...
- Tìm hiểu sâu hơn về các Framework nâng cao của Apple như Combine, WidgetKit để đáp ứng nhiều hơn nữa nhu cầu của người sử dụng, phát triển và tối ưu hóa ứng dụng.
- Tìm hiểu thêm về thiết kế UI/UX để nâng cao giao diện đồ họa đẹp mắt, sinh động hơn.
- Xây dựng ứng dụng quy mô lớn hơn với nhiều tính năng mở rộng...

Với thời gian và năng lực có hạn, trong một thời gian tôi đã nghiên cứu và ứng dụng ngôn ngữ Swift cùng framework SwiftUI để xây dựng ứng dụng "Chi Tiêu+". Trong quá trình thực hiện báo cáo này không thể tránh khỏi những thiếu sót. Kính mong sự thông cảm, góp ý và bổ sung của các thầy để ứng dụng ngày càng hoàn thiện hơn.

Em xin chân thành cảm ơn!

TÀI LIỆU THAM KHẢO

[1] **Apple Inc.** (2023). *Swift Programming Language (Swift 5.9)*. Truy cập từ https://docs.swift.org/swift-book/

[2] **Apple Developer Documentation.** (2023). *SwiftUI Framework*. Truy cập từ https://developer.apple.com/documentation/swiftui/

[3] **Apple Developer Documentation.** (2023). *Core Data Framework*. Truy cập từ https://developer.apple.com/documentation/coredata/

[4] **Apple Human Interface Guidelines.** (2023). *Designing for iOS*. Truy cập từ https://developer.apple.com/design/human-interface-guidelines/ios

[5] **Paul Hudson.** (2022). *Hacking with Swift: SwiftUI Edition*. Truy cập từ https://www.hackingwithswift.com/

[6] **Ray Wenderlich Team.** (2022). *Real-World iOS by Tutorials*. Razeware LLC.

[7] **Martin Fowler.** (2004). *Presentation Model (MVVM pattern origin)*. Truy cập từ https://martinfowler.com/eaaDev/PresentationModel.html

[8] **Microsoft.** (2022). *The MVVM Pattern*. Truy cập từ https://learn.microsoft.com/en-us/dotnet/architecture/maui/mvvm

[9] Các AI hỗ trợ: ChatGPT, Grok, Gemni, Cusor.