MỤC LỤC

[1. Yêu cầu(Requirements) 4](#_Toc193052333)

[1.1 Đặt vấn đề 4](#_Toc193052334)

[1.2 Thuật ngữ 5](#_Toc193052335)

[1.3. Thông số kỹ thuật bổ sung 5](#_Toc193052336)

[1.3.1. Hiệu năng 5](#_Toc193052337)

[1.3.2 Tốc độ 5](#_Toc193052338)

[1.3.3 Độ tin cậy 5](#_Toc193052339)

[1.3.4 Tính an toàn và bảo mật 6](#_Toc193052340)

[1.3.5 Khả năng bảo trì 6](#_Toc193052341)

[1.3.6 Khả năng sử dụng 6](#_Toc193052342)

[1.3.7 Khả năng mở rộng 6](#_Toc193052343)

[1.3.8 Tính khả dụng 6](#_Toc193052344)

[1.4 Mô hình hóa chức năng 7](#_Toc193052345)

[1.4.1 Các yêu cầu chức năng 7](#_Toc193052346)

[1.4.2 Sơ đồ Use case 9](#_Toc193052347)

[1.4.3 Bảng ánh xạ yêu cầu chức năng với các UC 11](#_Toc193052348)

[1.5 Đặc tả Use-case 12](#_Toc193052349)

[1.5.1 UC1.1: Quản lý giao dịch 12](#_Toc193052350)

[1.5.2 UC1.2: Quản lý danh mục 14](#_Toc193052351)

[1.5.3 UC2: Xem và phân tích báo cáo tài chính 15](#_Toc193052352)

[1.5.4 UC3.1: Thiết lập ngân sách 16](#_Toc193052353)

[1.5.5 UC3.2: Cảnh báo khi vượt ngân sách 17](#_Toc193052354)

[1.5.6 UC4.1: Đồng bộ dữ liệu 18](#_Toc193052355)

[1.5.7 UC4.2: Bảo mật dữ liệu 19](#_Toc193052356)

[1.6 Giao diện minh họa 21](#_Toc193052357)

[2.Phân tích trường hợp sử dụng (Use-case analysis) 23](#_Toc193052358)

[2.1 Phân tích kiến trúc hệ thống 23](#_Toc193052359)

[2.1.1 Kiến trúc mức cao của hệ thống 23](#_Toc193052360)

[2.1.2 Các đối tượng trừu tượng chính của hệ thống 24](#_Toc193052361)

[2.2 Thực thi trường hợp sử dụng(Use-case relizations) 24](#_Toc193052362)

[2.2.1 Các biểu đồ tuần tự (Sequence diagrams) 24](#_Toc193052363)

[2.2.2. Góc nhìn của các lớp trong hệ thống (Views of participating classes) 31](#_Toc193052364)

[3. Thiết kế (Use-case design) 33](#_Toc193052365)

[3.1. Xác định các thành phần thiết kế (Identify design elements) 33](#_Toc193052366)

[3.1.1 Xác định các lớp (Identify classes) 33](#_Toc193052367)

[3.1.2. Xác định các hệ thống con và giao diện (Identify subsystems and interfaces) 37](#_Toc193052368)

[3.1.3. Xác định các gói (Identify packages) 38](#_Toc193052369)

[3.2. Thiết kế trường hợp sử dụng (Use-case design) 39](#_Toc193052370)

[3.2.1. Thiết kế biểu đồ lớp (Class diagrams) 39](#_Toc193052371)

[3.2.2. Thiết kế các biểu đồ hoạt động (Activity diagrams) 39](#_Toc193052372)

[3.3. Thiết kế cơ sở dữ liệu (Database Design) 39](#_Toc193052373)

[3.3.1. Lược đồ cơ sở dữ liệu 39](#_Toc193052374)

[3.3.2. Chi tiết các bảng 40](#_Toc193052375)

[4.Cài đặt 45](#_Toc193052376)

[4.1. Công nghệ sử dụng 45](#_Toc193052377)

[4.2. Cấu trúc mã nguồn 46](#_Toc193052378)

[5. Kết luận 48](#_Toc193052379)

[TÀI LIỆU THAM KHẢO 49](#_Toc193052380)

# **1. Yêu cầu(Requirements)**

## 1.1 Đặt vấn đề

- Trong bối cảnh kinh tế hiện đại, việc quản lý tài chính cá nhân đã trở thành một yếu tố không thể thiếu đối với mỗi cá nhân. Sự phát triển nhanh chóng của công nghệ cùng với sự đa dạng của các phương thức giao dịch, từ thanh toán trực tuyến đến giao dịch qua điện thoại, đã tạo ra một khối lượng dữ liệu tài chính tương đối lớn hàng ngày. Những phương pháp ghi chép thủ công truyền thống không những tiêu tốn thời gian mà còn dễ mắc sai sót, khiến người dùng khó có thể theo dõi và kiểm soát được tình hình tài chính của mình một cách chính xác và kịp thời.

- Thực tế cho thấy khi không sử dụng một hệ thống quản lý tài chính, nhiều cá nhân dễ rơi vào tình trạng chi tiêu vượt quá thu nhập. Điều này dẫn đến việc mắc nợ, mất cân đối tài chính và thậm chí là khó khăn trong việc đáp ứng các nhu cầu sinh hoạt cơ bản. Việc thiếu một kế hoạch tài chính cụ thể không chỉ gây ra áp lực về mặt kinh tế mà còn ảnh hưởng tiêu cực đến tâm lý và chất lượng cuộc sống của người dùng.

- Để khắc phục những vấn đề nêu trên, phần mềm "Quản lý chi tiêu cá nhân" được xây dựng nhằm cung cấp một giải pháp công nghệ toàn diện giúp người dùng theo dõi, phân loại và phân tích các giao dịch tài chính hàng ngày một cách tự động và chính xác. Hệ thống được thiết kế để ghi nhận các khoản thu nhập và chi tiêu theo từng danh mục cụ thể, từ đó tạo điều kiện cho người dùng nắm bắt được xu hướng chi tiêu của bản thân. Qua việc phân tích chi tiết các giao dịch, người dùng có thể nhận diện các khoản chi không cần thiết và điều chỉnh kế hoạch tài chính phù hợp.

- Một trong những ưu điểm nổi bật của hệ thống là khả năng chuyển đổi dữ liệu giao dịch thành các báo cáo trực quan thông qua biểu đồ và bảng thống kê rõ ràng. Nhờ đó, người dùng có thể dễ dàng theo dõi dòng tiền, so sánh các khoản chi tiêu theo thời gian và đưa ra các quyết định tài chính hợp lý. Hệ thống cũng hỗ trợ tính năng thiết lập ngân sách hàng tháng, đặt mục tiêu tiết kiệm và gửi thông báo nhắc nhở khi chi tiêu vượt quá giới hạn đã đặt ra, giúp duy trì sự kiểm soát và cân đối tài chính.

- Nhìn chung, phần mềm "Quản lý chi tiêu cá nhân" hướng đến việc tạo ra một công cụ hỗ trợ đắc lực cho những ai mong muốn có cái nhìn tổng quan và chi tiết về tình hình tài chính cá nhân. Qua đó, phần mềm góp phần xây dựng lối sống tiết kiệm và thúc đẩy thói quen chi tiêu có kế hoạch, giúp người dùng đạt được sự ổn định tài chính và hướng tới một tương lai kinh tế lành mạnh.

## 1.2 Thuật ngữ

- Giao dịch – Một lần nhập thu nhập hoặc chi tiêu.

- Danh mục chi tiêu – Nhóm các khoản chi như ăn uống, đi lại , giải trí.

- Ngân sách – Số tiền tối đa cho từng danh mục trong một khoảng thời gian.

- Biểu đồ tài chính – Cách hiển thị dữ liệu chi tiêu theo thời gian.

- Báo cáo tài chính - Tổng hợp thu nhập, chi tiêu theo thời gian.

- Người dùng – Cá nhân sử dụng ứng dụng

- Firebase – Nền tảng lưu trữ dữ liệu đám mây

- Đồng bộ dữ liệu – Chia sẻ dữ liệu trên nhiều thiết bị

## 1.3. Thông số kỹ thuật bổ sung

### 1.3.1. Hiệu năng

- Ứng dụng cần có khả năng xử lý các giao dịch với tốc độ tối ưu

- Mỗi thao tác( thêm, sửa, xóa giao dịch) cần được hoàn thành dưới 1 giây để đảm bảo trải nghiệm mượt mà.

- Cấu trúc hệ thống được tối ưu hóa về thuật toán và cấu trúc dữ liệu nhằm đảm bảo hiệu năng ngay cả khi khối lượng giao dịch lớn.

### 1.3.2 Tốc độ

- Thời gian phản hồi trung bình của hệ thống không vượt quá 1 giây đối với các thao tác chính

- Thời gian tải dữ liệu và hiể thị báo cáo không quá 2 giây, kể cả khi số lượng giao dịch lớn

- Áp dụng các biện pháp tối ưu, tối ưu hóa truy vấn cơ sử dữ liệu và giảm thiểu các lệnh tính phức tạp.

### 1.3.3 Độ tin cậy

- Hệ thống đảm bảo tính chính xác của dữ liệu mỗi giao dịch, tránh mất mát hoặc sai lệnh thông tin.

- Đảm bảo tính nhất quán của dữ liệu trong quá trình đồng bộ hóa hóa giữa các thiết bị

### 1.3.4 Tính an toàn và bảo mật

- Dữ liệu người dùng được mã hóa khi truyền tải và lưu trữ nhằm bảo vệ thông tin cá nhân.

- Áp dụng xác thực đa yếu tố và các biện pháp kiểm soát truy cập nghiêm ngặt.

### 1.3.5 Khả năng bảo trì

- Hệ thông được thiết kế giúp dễ dàng cập nhật và bảo trì mà không ảnh hưởng đến toàn bộ hệ thống.

- Có tài liệu hướng dẫn cho việc bảo trì và nâng cấp, hỗ trợ xử lý sự cố nhanh chóng.

- Các phiên bản cập nhật được triển khai có kế hoạch rõ rang nhằm giảm thiểu gián đoạn hoạt động của hệ thống.

### 1.3.6 Khả năng sử dụng

- Giao điện người dùng được thiết kế trực quan, thân thiện, và dễ sử dụng, giúp người dùng thao tác nhanh chóng và hiệu quả.

- Cung cấp các hướng dẫn sử dụng và trợ giúp tích hợp ứng dựng, hỗ trợ người dùng mới nhanh chóng làm quen với hệ thống.

### 1.3.7 Khả năng mở rộng

- Hệ thống được xây dựng với kiến trúc phân tán, dễ dàng mở rộng để đáp ứng số lượng người dùng và giao dịch ngày càng tăng.

- Thiết kế cho phép tích hợp thêm các dịch vụ và chức năng mới mà không ảnh hưởng đến hoạt động hiện tại của hệ thống.

### 1.3.8 Tính khả dụng

- Úng dụng đảm bảo hoạt động ổn định với thời gian uptime trên 99% trong năm.

- Có cơ chế dự phòng và khôi phục nhanh chóng trong trường hợp xảy ra sự cố, đảm bảo người dùng luôn có thể truy cập hệ thống.

- Hệ thống được kiểm tra và đánh giá định kỳ về tính khả dụng để đảm bảo cung cấp trải nghiệm người dùng tốt nhất.

## 1.4 Mô hình hóa chức năng

### 1.4.1 Các yêu cầu chức năng

R1: Các yêu cầu về quản lý giao dịch và danh mục

R1.1. Hệ thống cho phép quản lý giao dịch

* *Thêm giao dịch:*  
  Cho phép người dùng nhập thông tin giao dịch bao gồm số tiền, danh mục, ngày tháng và mô tả chi tiết.
* *Sửa giao dịch:*  
  Hỗ trợ chỉnh sửa thông tin của giao dịch đã lưu nếu có sai sót hoặc khi cần cập nhật thông tin.
* *Xóa giao dịch:*  
  Cho phép loại bỏ các giao dịch không cần thiết khỏi hệ thống.
* *Bộ lọc và tìm kiếm giao dịch:*  
  Hệ thống hỗ trợ tìm kiếm và lọc giao dịch theo các tiêu chí như thời gian, danh mục hoặc số tiền để người dùng dễ dàng truy xuất thông tin.

R1.2. Hệ thống cho phép quản lý danh mục

* *Danh mục mặc định:*  
  Cung cấp các danh mục tiêu chuẩn (ví dụ: Ăn uống, Đi lại, Giải trí, Hóa đơn, …) để phân loại giao dịch.
* *Tạo danh mục mới:*  
  Cho phép người dùng tự tạo danh mục tùy chỉnh theo nhu cầu cá nhân khi các danh mục mặc định không đáp ứng đủ yêu cầu.
* *Gợi ý danh mục:*  
  Khi nhập giao dịch, hệ thống tự động gợi ý danh mục dựa trên lịch sử giao dịch nhằm tối ưu quá trình nhập liệu.

R2: Các yêu cầu về báo cáo và phân tích tài chính

R2.1. Hệ thống cho phép xem và phân tích báo cáo tài chính

* *Lựa chọn khoảng thời gian:*  
  Người dùng có thể chọn khoảng thời gian (ngày, tuần, tháng, năm) để lọc và hiển thị báo cáo tài chính.
* *Hiển thị biểu đồ:*  
  Báo cáo được trình bày dưới dạng biểu đồ cột hoặc biểu đồ tròn, thể hiện tỷ lệ chi tiêu theo từng danh mục.
* *Xuất báo cáo:*  
  Hệ thống hỗ trợ xuất báo cáo dưới dạng file CSV hoặc hình ảnh, giúp người dùng lưu trữ, chia sẻ và phân tích dữ liệu bên ngoài hệ thống.

R3: Các yêu cầu về quản lý ngân sách và cảnh báo

R3.1. Hệ thống cho phép thiết lập ngân sách

* *Đặt ngân sách:*  
  Cho phép người dùng thiết lập ngân sách cho mỗi danh mục (ví dụ: 2 triệu/tháng cho chi tiêu ăn uống).
* *Cập nhật ngân sách:*  
  Hỗ trợ việc chỉnh sửa ngân sách khi nhu cầu hoặc điều kiện chi tiêu thay đổi theo thời gian.

R3.2. Hệ thống cho phép cảnh báo khi vượt ngân sách

* *Giám sát chi tiêu:*  
  Hệ thống theo dõi liên tục các giao dịch và so sánh với mức ngân sách đã đặt cho từng danh mục.
* *Thông báo cảnh báo:*  
  Khi tổng chi tiêu đạt đến hoặc vượt qua ngưỡng ngân sách, hệ thống tự động gửi cảnh báo qua thông báo trong ứng dụng hoặc qua email.
* *Báo cáo vượt ngân sách:*  
  Cung cấp báo cáo chi tiết về các giao dịch đã gây ra việc vượt ngân sách để người dùng có thể điều chỉnh kế hoạch chi tiêu.

R4: Các yêu cầu về đồng bộ hóa và bảo mật dữ liệu

R4.1. Hệ thống cho phép đồng bộ dữ liệu

* *Đăng nhập và xác thực:*  
  Sử dụng Firebase Authentication để xác thực người dùng khi đăng nhập, đảm bảo chỉ người dùng hợp lệ truy cập vào hệ thống.
* *Đồng bộ tự động:*  
  Dữ liệu giao dịch, danh mục và ngân sách được tự động đồng bộ hóa qua Firebase giữa các thiết bị, đảm bảo tính liền mạch cho người dùng.
* *Khôi phục dữ liệu:*  
  Hỗ trợ khôi phục dữ liệu từ bản sao lưu trên đám mây trong trường hợp mất dữ liệu cục bộ hoặc khi chuyển đổi thiết bị.

R4.2. Hệ thống đảm bảo bảo mật dữ liệu

* *Mã hóa dữ liệu:*  
  Áp dụng các phương pháp mã hóa khi truyền tải và lưu trữ dữ liệu nhằm bảo vệ thông tin cá nhân của người dùng.
* *Xác thực đa yếu tố:*  
  Tăng cường bảo mật thông qua việc sử dụng xác thực đa yếu tố (ví dụ: 2FA) để đảm bảo an toàn đăng nhập.
* *Phòng chống tấn công:*  
  Triển khai các biện pháp bảo vệ như chống SQL Injection, XSS và các lỗ hổng bảo mật khác nhằm ngăn chặn truy cập trái phép.
* *Kiểm tra định kỳ:*  
  Thực hiện đánh giá và cập nhật các giải pháp bảo mật định kỳ để đảm bảo hệ thống luôn được bảo vệ tốt nhất.

### 1.4.2 Sơ đồ Use case

**A diagram of a person with different colored circles

AI-generated content may be incorrect.**

*Sơ đồ Use – case*

### 1.4.3 Bảng ánh xạ yêu cầu chức năng với các UC

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | Mã Yêu Cầu | Yêu Cầu Chức Năng | Use Case (UC) Tương Ứng | Mô tả UC |
| 1 | R1.1 | Quản lý giao dịch: Thêm, sửa, xóa, tìm kiếm giao dịch | UC1.1: Quản lý giao dịch | Người dùng thực hiện các thao tác cơ bản với giao dịch: Nhập thông tin giao dịch, chỉnh sửa, xóa, và tìm kiếm theo tiêu chí(thời gian, danh mục,số tiền). |
| 2 | R1.2 | Quản lý danh mục: Danh mục mặc định, tạo mới, gợi ý danh mục | UC1.2: Quản lý danh mục | Người xem, tạo mới danh mục và nhận gợi ý danh mục khi nhập giao dịch, giúp phân loại giao dịch một cách hiệu quả. |
| 3 | R2.1 | Xem và phân tích báo cáo tài chính: Lựa chọn khoảng thời gian, biểu đò, xuất báo cáo | UC2: Xem và phân tích báo cáo tài chính | Người dùng lựa chọn khoảng thời gian, hệ thống hiển thị báo cáo dưới dạng biểu đồ( cột,tròn) và hỗ trợ xuất báo cáo dưới dạng file CSV hoặc hình ảnh |
| 4 | R3.1 | Thiết lập ngân sách: Đặt và cập nhật ngân sách cho từng danh mục | UC3.1: Thiết lập ngân sách | Người dùng thiết lập ngân sách cho mỗi danh mục chi tiêu( ví dụ: 2 triệu/tháng cho ăn uống) và có thể cập nhật ngân sách khi nhu cầu thay đổi. |
| 5 | R3.2 | Cảnh báo khi vượt ngân sách: giám sát chi tiêu, gửi cảnh báo, báo cáo vượt ngân sách | UC3.2:Cảnh báo khi vượt ngân sách | Hệ thống theo dõi giao dịch so sánh với ngân sách, khi chi tiêu đạt hoặc vượt ngưỡng thì gửi cảnh báo qua thông báo trong ứng dụng hoặc email, đồng thời cung cấp báo cáo chi tiết. |
| 6 | R4.1 | Đồng bộ dữ liệu: Đăng nhập, xác thực, đồng bộ tự động, khôi phục dữ liệu | UC4.1: Đồng bộ dữ liệu | Người dùng đăng nhập, dữ liệu giao dịch, danh mục và ngân sách được đồng bộ tự động giữa các thiết bị và hỗ trợ khôi phục dữ liệu khi cần. |
| 7 | R4.2 | Bảo mật dữ liệu: Mã hóa, xác thực đa yếu tố, phòng chống tấn công, kiểm tra bảo mật định kỳ | UC4.2: Bảo mật dữ liệu | Hệ thống bảo vệ thông tin người dùng thông qua các cơ chế mã hóa dữ liệu, xác thực yếu tố, triển khai phồn chống tấn công và kiểm tra, cập nhật bảo mật định kỳ |

## 1.5 Đặc tả Use-case

### 1.5.1 UC1.1: Quản lý giao dịch

Tên Use Case: Quản lý giao dịch  
Tác nhân: Người dùng  
Mục tiêu: Cho phép người dùng thực hiện các thao tác cơ bản với giao dịch tài chính: thêm, sửa, xóa và tìm kiếm giao dịch.  
Điều kiện tiền đề:

* Người dùng đã đăng nhập thành công vào hệ thống.
* Giao diện “Quản lý giao dịch” được hiển thị với danh sách giao dịch hiện có.

Luồng cơ bản (Main Flow):

1. Người dùng chọn mục “Quản lý giao dịch” từ menu chính.
2. Hệ thống hiển thị danh sách các giao dịch hiện có.
3. Người dùng nhấn nút “Thêm giao dịch”.
4. Hệ thống hiển thị form nhập thông tin giao dịch.
5. Người dùng nhập đầy đủ thông tin: số tiền, danh mục, ngày giao dịch và mô tả chi tiết.
6. Người dùng nhấn nút “Lưu”.
7. Hệ thống xác thực thông tin, lưu giao dịch vào cơ sở dữ liệu và cập nhật danh sách giao dịch.
8. Hệ thống hiển thị thông báo “Thêm giao dịch thành công”.

Luồng phụ (Alternate Flows):

* Luồng phụ A – Chỉnh sửa giao dịch:  
  1. Người dùng chọn một giao dịch từ danh sách để chỉnh sửa.  
  2. Hệ thống hiển thị form chỉnh sửa với dữ liệu giao dịch đã lưu.  
  3. Người dùng thay đổi thông tin cần thiết và nhấn “Cập nhật”.  
  4. Hệ thống xác thực, cập nhật dữ liệu và làm mới danh sách giao dịch.  
  5. Hệ thống hiển thị thông báo “Cập nhật giao dịch thành công”.
* Luồng phụ B – Xóa giao dịch:  
  1. Người dùng chọn giao dịch cần xóa.  
  2. Hệ thống hiển thị thông báo xác nhận “Bạn có chắc chắn muốn xóa giao dịch này?”.  
  3. Người dùng xác nhận xóa.  
  4. Hệ thống xóa giao dịch khỏi cơ sở dữ liệu và cập nhật danh sách.  
  5. Hệ thống hiển thị thông báo “Xóa giao dịch thành công”.

Luồng thay thế (Exception/Alternative Flows):

* Luồng thay thế A – Lỗi nhập liệu:  
  1. Nếu thông tin nhập vào không hợp lệ (ví dụ: số tiền không phải số, ngày không đúng định dạng), hệ thống hiển thị thông báo lỗi và đánh dấu các trường cần sửa.  
  2. Người dùng điều chỉnh lại thông tin và nhấn “Lưu” lại.
* Luồng thay thế B – Lỗi kết nối:  
  1. Nếu hệ thống gặp lỗi kết nối khi lưu giao dịch, hiển thị thông báo “Lỗi kết nối. Vui lòng thử lại sau.”  
  2. Người dùng có thể chọn “Thử lại” hoặc “Hủy” thao tác.  
  3. Nếu chọn “Thử lại”, hệ thống thực hiện lại bước lưu giao dịch; nếu “Hủy”, hệ thống quay trở lại danh sách giao dịch.

### 1.5.2 UC1.2: Quản lý danh mục

Tên Use Case: Quản lý danh mục  
Tác nhân: Người dùng  
Mục tiêu: Cho phép người dùng quản lý danh mục giao dịch: xem danh mục, thêm mới, chỉnh sửa và xóa danh mục, đồng thời nhận gợi ý danh mục khi nhập giao dịch.  
Điều kiện tiền đề:

* Người dùng đã đăng nhập và đang ở giao diện “Quản lý danh mục”.

Luồng cơ bản (Main Flow):

1. Người dùng chọn mục “Quản lý danh mục” từ menu.
2. Hệ thống hiển thị danh sách các danh mục hiện có.
3. Người dùng nhấn nút “Thêm danh mục”.
4. Hệ thống hiển thị form nhập thông tin danh mục mới.
5. Người dùng nhập tên danh mục (và mô tả nếu cần).
6. Người dùng nhấn “Lưu”.
7. Hệ thống lưu thông tin danh mục mới và cập nhật danh sách.
8. Hệ thống hiển thị thông báo “Thêm danh mục thành công”.

Luồng phụ (Alternate Flows):

* Luồng phụ A – Chỉnh sửa danh mục:  
  1. Người dùng chọn một danh mục từ danh sách để chỉnh sửa.  
  2. Hệ thống hiển thị form chỉnh sửa với dữ liệu danh mục hiện có.  
  3. Người dùng thực hiện thay đổi và nhấn “Cập nhật”.  
  4. Hệ thống lưu thông tin mới và cập nhật danh sách.  
  5. Hệ thống hiển thị thông báo “Cập nhật danh mục thành công”.
* Luồng phụ B – Xóa danh mục:  
  1. Người dùng chọn danh mục cần xóa.  
  2. Hệ thống hiển thị thông báo xác nhận “Bạn có chắc chắn muốn xóa danh mục này?”.  
  3. Người dùng xác nhận xóa.  
  4. Hệ thống xóa danh mục khỏi cơ sở dữ liệu và cập nhật danh sách.  
  5. Hệ thống hiển thị thông báo “Xóa danh mục thành công”.

Luồng thay thế (Exception/Alternative Flows):

* Luồng thay thế A – Lỗi nhập liệu:  
  1. Nếu tên danh mục bị trống hoặc không hợp lệ, hệ thống hiển thị thông báo lỗi.  
  2. Người dùng điều chỉnh và nhấn “Lưu” hoặc “Cập nhật” lại.
* Luồng thay thế B – Hủy thao tác:  
  1. Người dùng nhấn nút “Hủy” trong form nhập/chỉnh sửa danh mục.  
  2. Hệ thống trở lại danh sách danh mục mà không lưu thay đổi.

### 1.5.3 UC2: Xem và phân tích báo cáo tài chính

Tên Use Case: Xem và phân tích báo cáo tài chính  
Tác nhân: Người dùng  
Mục tiêu: Cho phép người dùng xem báo cáo tài chính trực quan với biểu đồ, lọc theo khoảng thời gian và xuất báo cáo ra file.  
Điều kiện tiền đề:

* Người dùng đã đăng nhập và có giao dịch được lưu trữ trong hệ thống.

Luồng cơ bản (Main Flow):

1. Người dùng chọn mục “Báo cáo tài chính” từ menu.
2. Hệ thống hiển thị giao diện báo cáo với các tùy chọn lọc theo khoảng thời gian (ngày, tuần, tháng, năm).
3. Người dùng chọn khoảng thời gian mong muốn và nhấn “Xem báo cáo”.
4. Hệ thống tính toán và hiển thị báo cáo dưới dạng biểu đồ (cột hoặc tròn), thể hiện tỷ lệ chi tiêu theo danh mục.
5. Người dùng có tùy chọn nhấn “Xuất báo cáo” để lưu báo cáo dưới dạng file CSV hoặc hình ảnh.
6. Hệ thống hoàn tất thao tác và hiển thị thông báo “Xuất báo cáo thành công”.

Luồng phụ (Alternate Flows):

* Luồng phụ A – Thay đổi kiểu hiển thị báo cáo:  
  1. Người dùng có thể chuyển đổi giữa các loại biểu đồ (ví dụ: từ biểu đồ cột sang biểu đồ tròn).  
  2. Hệ thống làm mới giao diện báo cáo theo kiểu được chọn.

Luồng thay thế (Exception/Alternative Flows):

* Luồng thay thế A – Không có dữ liệu cho khoảng thời gian chọn:  
  1. Nếu không có giao dịch nào trong khoảng thời gian được chọn, hệ thống hiển thị thông báo “Không có dữ liệu cho khoảng thời gian này”.  
  2. Người dùng có thể chọn khoảng thời gian khác hoặc nhấn “Hủy” để quay lại giao diện chính.
* Luồng thay thế B – Lỗi khi xuất báo cáo:  
  1. Nếu quá trình xuất báo cáo gặp lỗi (ví dụ: lỗi hệ thống, lỗi định dạng file), hệ thống hiển thị thông báo “Lỗi khi xuất báo cáo. Vui lòng thử lại sau.”

### 1.5.4 UC3.1: Thiết lập ngân sách

Tên Use Case: Thiết lập ngân sách  
Tác nhân: Người dùng  
Mục tiêu: Cho phép người dùng đặt và cập nhật ngân sách cho từng danh mục chi tiêu nhằm kiểm soát chi tiêu cá nhân.  
Điều kiện tiền đề:

* Người dùng đã đăng nhập và có quyền truy cập chức năng quản lý ngân sách.
* Hệ thống có sẵn dữ liệu về danh mục giao dịch.

Luồng cơ bản (Main Flow):

1. Người dùng chọn mục “Ngân sách” từ menu.
2. Hệ thống hiển thị danh sách các danh mục với ngân sách hiện tại (nếu có).
3. Người dùng chọn “Thiết lập ngân sách” cho một danh mục cụ thể.
4. Hệ thống hiển thị form nhập số tiền ngân sách cho danh mục đó.
5. Người dùng nhập số tiền ngân sách và nhấn “Lưu”.
6. Hệ thống xác thực và lưu thông tin ngân sách vào cơ sở dữ liệu.
7. Hệ thống hiển thị thông báo “Thiết lập ngân sách thành công” và cập nhật giao diện.

Luồng phụ (Alternate Flows):

* Luồng phụ A – Cập nhật ngân sách:  
  1. Người dùng chọn danh mục có ngân sách đã được thiết lập.  
  2. Hệ thống hiển thị thông tin ngân sách hiện tại kèm tùy chọn “Chỉnh sửa”.  
  3. Người dùng thực hiện chỉnh sửa và nhấn “Cập nhật”.  
  4. Hệ thống lưu thông tin mới và cập nhật giao diện, hiển thị thông báo “Cập nhật ngân sách thành công”.

Luồng thay thế (Exception/Alternative Flows):

* Luồng thay thế A – Lỗi nhập liệu:  
  1. Nếu số tiền nhập không hợp lệ (ví dụ: để trống, không phải số), hệ thống hiển thị thông báo lỗi và đánh dấu trường cần sửa.  
  2. Người dùng điều chỉnh lại và nhấn “Lưu” lại.
* Luồng thay thế B – Hủy thao tác:  
  1. Người dùng nhấn “Hủy” trong form thiết lập ngân sách.  
  2. Hệ thống thoát khỏi form và quay lại giao diện danh mục ngân sách mà không lưu thay đổi.

### 1.5.5 UC3.2: Cảnh báo khi vượt ngân sách

Tên Use Case: Cảnh báo khi vượt ngân sách  
Tác nhân: Hệ thống (và người dùng nhận thông báo)  
Mục tiêu: Tự động giám sát và gửi cảnh báo cho người dùng khi tổng chi tiêu của một danh mục vượt quá ngân sách đã được thiết lập.  
Điều kiện tiền đề:

* Người dùng đã thiết lập ngân sách cho các danh mục.
* Giao dịch được nhập đầy đủ và hệ thống đang theo dõi tổng chi tiêu.

Luồng cơ bản (Main Flow):

1. Hệ thống liên tục theo dõi và tổng hợp các giao dịch của từng danh mục theo thời gian thực.
2. Khi tổng chi tiêu của một danh mục đạt đến ngưỡng ngân sách đã thiết lập, hệ thống tự động kích hoạt quá trình cảnh báo.
3. Hệ thống gửi thông báo cảnh báo (trong ứng dụng và/hoặc qua email) cho người dùng với nội dung “Chi tiêu của danh mục [Tên danh mục] đã đạt ngưỡng ngân sách”.
4. Hệ thống ghi nhận thông báo và cập nhật báo cáo vượt ngân sách.

Luồng phụ (Alternate Flows):

* Luồng phụ A – Xem báo cáo vượt ngân sách:  
  1. Người dùng có thể nhấn vào thông báo cảnh báo để chuyển sang giao diện báo cáo chi tiết.  
  2. Hệ thống hiển thị báo cáo các giao dịch đã góp phần vượt ngân sách, cho phép người dùng phân tích và điều chỉnh kế hoạch chi tiêu.

Luồng thay thế (Exception/Alternative Flows):

* Luồng thay thế A – Sai lệch dữ liệu:  
  1. Nếu dữ liệu giao dịch không được cập nhật chính xác, hệ thống hiển thị thông báo “Dữ liệu chưa được cập nhật đầy đủ. Vui lòng kiểm tra lại”.  
  2. Người dùng có thể chọn “Thử lại” sau vài phút để hệ thống tự động cập nhật lại dữ liệu.

### 1.5.6 UC4.1: Đồng bộ dữ liệu

Tên Use Case: Đồng bộ dữ liệu  
Tác nhân: Người dùng  
Mục tiêu: Đảm bảo dữ liệu giao dịch, danh mục và ngân sách được lưu trữ và cập nhật đồng bộ trên nhiều thiết bị thông qua nền tảng đám mây (ví dụ: Firebase).  
Điều kiện tiền đề:

* Người dùng đã đăng nhập thành công vào hệ thống.
* Thiết bị có kết nối Internet ổn định.

Luồng cơ bản (Main Flow):

1. Người dùng đăng nhập vào hệ thống bằng tài khoản được xác thực (sử dụng Firebase Authentication).
2. Hệ thống tự động tải dữ liệu giao dịch, danh mục và ngân sách từ đám mây về thiết bị hiện tại.
3. Trong quá trình sử dụng, mọi thay đổi (thêm, sửa, xóa giao dịch/danh mục, cập nhật ngân sách) được tự động đồng bộ lên đám mây.
4. Khi người dùng chuyển sang một thiết bị khác, hệ thống tự động đồng bộ và hiển thị dữ liệu cập nhật nhất.

Luồng phụ (Alternate Flows):

* Luồng phụ A – Khôi phục dữ liệu:  
  1. Trong trường hợp dữ liệu cục bộ bị mất, người dùng có thể kích hoạt chức năng “Khôi phục dữ liệu”.  
  2. Hệ thống lấy lại dữ liệu từ bản sao lưu trên đám mây và khôi phục vào thiết bị.

Luồng thay thế (Exception/Alternative Flows):

* Luồng thay thế A – Lỗi đồng bộ:  
  1. Nếu quá trình đồng bộ bị gián đoạn (do lỗi mạng, lỗi máy chủ), hệ thống hiển thị thông báo “Lỗi đồng bộ dữ liệu. Vui lòng kiểm tra kết nối Internet và thử lại.”  
  2. Hệ thống tự động thử lại đồng bộ sau 1 khoảng thời gian nhất định hoặc khi người dùng nhấn nút “Thử lại”.

### 1.5.7 UC4.2: Bảo mật dữ liệu

Tên Use Case: Bảo mật dữ liệu  
Tác nhân: Hệ thống (và người dùng nhận sự bảo vệ)  
Mục tiêu: Đảm bảo an toàn và bảo mật cho toàn bộ thông tin người dùng thông qua các biện pháp mã hóa, xác thực đa yếu tố và phòng chống tấn công.  
Điều kiện tiền đề:

* Người dùng đã đăng nhập vào hệ thống.
* Hệ thống đã được cấu hình các biện pháp bảo mật cần thiết.

Luồng cơ bản (Main Flow):

1. Trong quá trình đăng nhập, hệ thống sử dụng cơ chế xác thực (ví dụ: email, mật khẩu và 2FA nếu được kích hoạt) để đảm bảo chỉ người dùng hợp lệ truy cập.
2. Khi truyền tải dữ liệu giữa thiết bị và máy chủ, hệ thống áp dụng mã hóa để bảo vệ dữ liệu.
3. Trong quá trình hoạt động, hệ thống giám sát các yêu cầu truy cập và các hành vi bất thường nhằm phát hiện sớm các cuộc tấn công (SQL Injection, XSS, …).
4. Hệ thống thực hiện kiểm tra bảo mật định kỳ và cập nhật các biện pháp phòng chống mới nếu cần.

Luồng phụ (Alternate Flows):

* Luồng phụ A – Xác thực bổ sung:  
  1. Trong trường hợp nghi ngờ truy cập trái phép, hệ thống yêu cầu xác thực bổ sung (ví dụ: nhập mã OTP gửi qua SMS hoặc email).  
  2. Người dùng nhập mã xác thực và hệ thống kiểm tra, nếu hợp lệ thì cho phép tiếp tục truy cập.

Luồng thay thế (Exception/Alternative Flows):

* Luồng thay thế A – Lỗi xác thực:  
  1. Nếu thông tin đăng nhập không hợp lệ hoặc mã OTP sai, hệ thống hiển thị thông báo “Xác thực không thành công. Vui lòng thử lại.”  
  2. Sau một số lần thất bại, hệ thống có thể khóa tạm thời tài khoản và yêu cầu liên hệ với bộ phận hỗ trợ.
* Luồng thay thế B – Lỗi mã hóa dữ liệu:  
  1. Nếu phát hiện lỗi trong quá trình mã hóa hoặc truyền tải dữ liệu, hệ thống ghi nhận sự cố và hiển thị thông báo “Lỗi bảo mật. Vui lòng thử lại sau.”  
  2. Hệ thống gửi báo cáo lỗi đến bộ phận quản trị để xử lý kịp thời.

## 1.6 Giao diện minh họa

**Screens screenshot of a phone

AI-generated content may be incorrect.**

*Hình 1.1 Giao diện bắt đầu*

***Screens screenshots of a mobile phone

AI-generated content may be incorrect.***

*Hình 1.2 Giao diện trang chủ*

***Screens screenshot of a phone

AI-generated content may be incorrect.***

*Hình 1.3 Giao diện báo cáo tài chính*

# **2.Phân tích trường hợp sử dụng (Use-case analysis)**

## 2.1 Phân tích kiến trúc hệ thống

Kiến trúc hệ thống là yếu tố quan trọng đảm bảo sự hoạt động ổn định, hiệu suất cao và khả năng mở rộng của hệ thống quản lý chi tiêu cá nhân. Một hệ thống tốt cần có sự phân tách rõ rang giữa các thành phần, đảm bảo tính bảo mật và dễ dàng bảo trì. Hệ thống được thiết kế dựa trên mô hình Client-Server, giao diện người dùng sẽ gửi yêu cầu đến máy chủ, nơi đảm nhiệm xử lý dữ liệu và trả về kết quả tương ứng.

### 2.1.1 Kiến trúc mức cao của hệ thống

Hệ thống "Quản lý chi tiêu cá nhân" được phát triển bằng Flutter và sử dụng Firebase làm nền tảng backend để quản lý dữ liệu. Hệ thống có kiến trúc tổng thể như sau:

* Giao diện người dùng (Frontend):
  + Được phát triển bằng Flutter, hỗ trợ đa nền tảng (Android, iOS, Web).
  + Giao diện trực quan, thân thiện với người dùng, tối ưu trải nghiệm.
  + Sử dụng các widget để xây dựng các thành phần UI linh hoạt.
* Máy chủ và Backend:
  + Firebase Authentication để xác thực người dùng.
  + Firebase Firestore để lưu trữ và quản lý dữ liệu giao dịch, danh mục, ngân sách.
  + Firebase Cloud Functions để xử lý các tác vụ nền như thông báo cảnh báo vượt ngân sách.
* Hệ thống bảo mật:
  + Dữ liệu người dùng được mã hóa khi lưu trữ và truyền tải.
  + Áp dụng xác thực đa yếu tố để bảo vệ tài khoản.
  + Firebase Security Rules đảm bảo quyền truy cập phù hợp với từng người dùng.
* API và Giao Tiếp Dữ Liệu:
  + Flutter sử dụng thư viện http hoặc dio để gọi API khi cần thiết.
  + Firestore hỗ trợ truy vấn theo thời gian thực, giúp dữ liệu được đồng bộ ngay lập tức trên nhiều thiết bị.

### 2.1.2 Các đối tượng trừu tượng chính của hệ thống

Dưới đây là bảng mô tả các đối tượng trừu tượng chính của hệ thống:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | Tên Đối Tượng | Vai Trò | Thuộc Tính | Phương Thức |
| 1 | Người dùng | Quản lý tài khoản và dữ liệu cá nhân | ID, Tên, Email, Mật khẩu, Ngân sách cá nhân | Đăng ký, Đăng nhập, Cập nhật thông tin, Xóa tài khoản |
| 2 | Giao dịch | Lưu trữ các khoản thu và chi tiêu | ID, Số tiền, Danh mục, Ngày, Mô tả | Thêm, Sửa, Xóa, Tìm kiếm giao dịch |
| 3 | Danh mục | Nhóm các khoản chi tiêu | ID, Tên danh mục, Màu sắc | Thêm, Sửa, Xóa, Lấy danh sách danh mục |
| 4 | Ngân sách | Quản lý giới hạn chi tiêu | ID, Số tiền tối đa, Danh mục, Thời gian hiệu lực | Đặt ngân sách, Cập nhật ngân sách, Xóa ngân sách |
| 5 | Báo cáo tài chính | Tổng hợp dữ liệu thu chi | ID, Dữ liệu giao dịch, Tổng thu, Tổng chi | Tạo báo cáo, Hiển thị biểu đồ, Xuất file CSV |
| 6 | Thông báo | Cảnh báo khi chi tiêu vượt hạn mức | ID, Nội dung, Loại thông báo, Trạng thái | Gửi thông báo, Xác nhận đã đọc |

## 2.2 Thực thi trường hợp sử dụng(Use-case relizations)

### 2.2.1 Các biểu đồ tuần tự (Sequence diagrams)

2.2.1.1 Cấu trúc chung

**\*** Sơ đồ lớp phân tích

A diagram of a diagram

AI-generated content may be incorrect.

- Actor**:** Tác nhân tham gia sửa dụng hệ thống

- Form**:** Form là lớp trung gian thể hiện sự tương tác giữa hệ thống và những gì bên ngoài hệ thống. Ở đây là giao diện thực hiện chức năng chính của UC

- Control**:** Lớp điều khiển thực hiện chức năng chính của UC

- Entity**:** Lớp thực hiện chứa những thông tin tồn tại và được lưu trữ lâu dài trong hệ thống

- API**:** Lớp điều khiển thực hiện việc nhân/trả các API

- Database**:** Cơ sở dữ liệu

\* Sơ đồ tuần tự chung:

A diagram of a diagram

AI-generated content may be incorrect.

2.2.1.2 Các sơ đồ lớp và sơ đồ tuần tự của các chức năng:

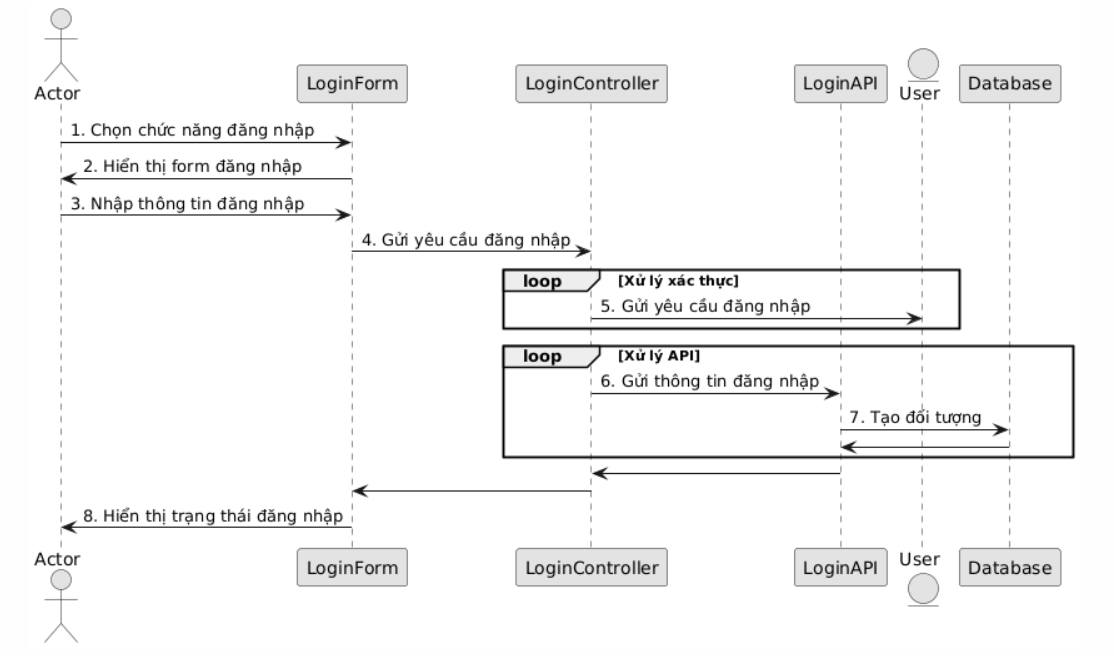
#### \* Sơ đồ tuần tự chức năng đăng nhập

* Sơ đồ lớp phân tích

A diagram of a diagram

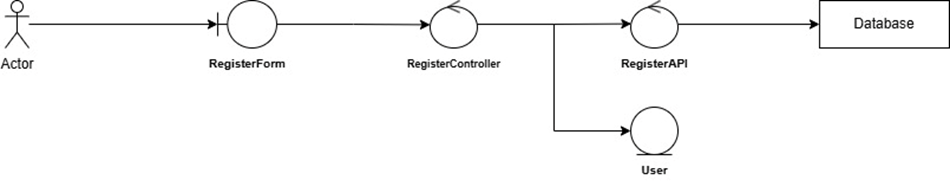
AI-generated content may be incorrect.

* Sơ đồ tuần tự



#### \* Sơ đồ tuần tự chức năng đăng ký

* Sơ đồ lớp phân tích



* Sơ đồ tuần tự

A diagram of a register

AI-generated content may be incorrect.

#### Sơ đồ tuần tự chức năng cập nhật thông tin cá nhân

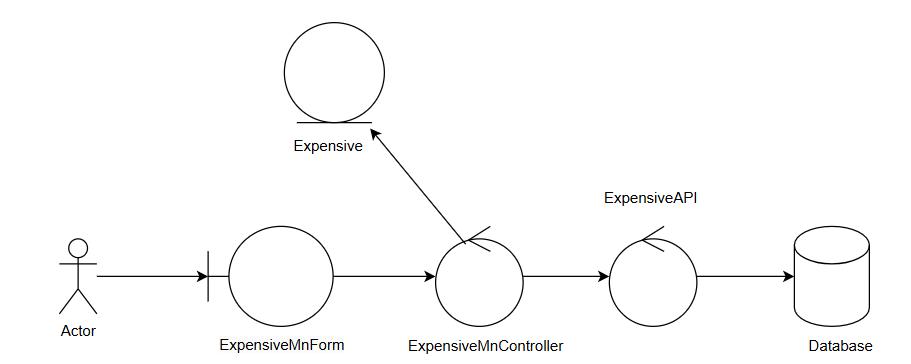
* Sơ đồ lớp phân tíchẢnh có chứa biểu đồ, bản phác thảo, vòng tròn, thiết kế

  Mô tả được tạo tự động
* Sơ đồ tuần tự

A screenshot of a computer

AI-generated content may be incorrect.

#### Sơ đồ tuần tự chức năng quản lý tài chính

* Sơ đồ lớp phân tích 
* Sơ đồ tuần tựA black and white document with text

  AI-generated content may be incorrect.

### 2.2.2. Góc nhìn của các lớp trong hệ thống (Views of participating classes)

|  |  |  |  |
| --- | --- | --- | --- |
| **Tên Lớp (Class)** | Loại Lớp (Class Type) | Mô Tả (Description) | Mối Quan Hệ (Relationships) |
| AuthWrapper | Boundary | Quyết định màn hình hiển thị (đăng nhập hay chính) dựa trên trạng thái xác thực. | - Sử dụng LoginScreen và MainScreen để điều hướng. |
| ForgotPassword | Boundary | Hiển thị giao diện khôi phục mật khẩu. | - Gọi AuthService để gửi yêu cầu khôi phục mật khẩu. |
| GoogleSignIn | Boundary | Xử lý đăng nhập bằng Google. | - Gọi AuthService để xác thực qua Google. |
| LoginScreen | Boundary | Hiển thị giao diện đăng nhập | - Gọi AuthService để xác thực người dùng. |
| Registration | Boundary | Hiển thị giao diện đăng ký. | - Gọi AuthService để tạo tài khoản mới. |
| ExpenseData | Entity | Lưu trữ dữ liệu chi tiêu (có thể liên quan đến phân tích giao dịch). | - Được sử dụng bởi MainScreen hoặc TransactionScreen để hiển thị dữ liệu phân tích. |
| Transaction | Entity | Lưu trữ thông tin giao dịch (số tiền, loại, ngày, v.v.). | - Được sử dụng bởi TransactionService và TransactionScreen. |
| AddTrans | Boundary | Hiển thị giao diện thêm giao dịch mới. | - Gọi TransactionService để lưu giao dịch mới. |
| AppTheme | Utility | Xác định chủ đề giao diện (theme) của ứng dụng. | - Được sử dụng bởi MainScreen và các màn hình khác để áp dụng giao diện. |
| BudgetScreen | Boundary | Hiển thị giao diện quản lý ngân sách. | - Gọi BudgetService để lấy và cập nhật ngân sách. |
| HomeScreen | Boundary | Hiển thị giao diện chính (tổng quan tài chính). | - Gọi BudgetService và TransactionService để lấy dữ liệu tổng quan. |
| MainScreen | Boundary | Giao diện chính điều hướng đến các màn hình khác (Home, Transaction, Budget, Profile). | - Sử dụng HomeScreen, TransactionScreen, BudgetScreen, ProfileScreen để điều hướng. |
| ProfileScreen | Boundary | Hiển thị và cập nhật thông tin cá nhân. | - Gọi AuthService để lấy và cập nhật thông tin người dùng. |
| TransactionScreen | Boundary | Hiển thị danh sách giao dịch. | - Gọi TransactionService để lấy danh sách giao dịch. |
| BudgetService | Service | Quản lý dữ liệu ngân sách (lấy, cập nhật). | - Sử dụng Transaction để tính toán chi tiêu liên quan đến ngân sách. |
| TransactionService | Service | Quản lý dữ liệu giao dịch (lấy, thêm, lọc). | - Sử dụng Transaction để truy xuất và lưu trữ dữ liệu giao dịch. |
| AnimatedPieChart | Widget | Widget hiển thị biểu đồ tròn động để trực quan hóa dữ liệu tài chính. | - Được sử dụng bởi MainScreen hoặc HomeScreen để hiển thị dữ liệu. |
| FirebaseOptions | Utility | Cấu hình kết nối với Firebase (dùng cho xác thực, lưu trữ). | - Được sử dụng bởi AuthService và TransactionService để kết nối Firebase. |

# **3. Thiết kế (Use-case design)**

## 3.1. Xác định các thành phần thiết kế (Identify design elements)

### 3.1.1 Xác định các lớp (Identify classes)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tên Lớp (Class) | Thuộc về Subsystem | Trách nhiệm chính (Responsibilities) | Các Thuộc Tính Chính (Attributes) | Các Phương Thức Chính (Methods) |
| AuthWrapper | User Interface | Điều hướng giữa màn hình đăng nhập và chính dựa trên trạng thái xác thực. | bool isAuthenticated | - build(BuildContext): Xây dựng giao diện điều hướng. |
| ForgotPassword | User Interface | Hiển thị giao diện khôi phục mật khẩu. | TextEditingController emailController | build(BuildContext): Xây dựng giao diện.  - onRecoverPressed(): Gửi yêu cầu khôi phục mật khẩu. |
| GoogleSignIn | User Interface | Xử lý đăng nhập bằng Google. | String? googleToken | - build(BuildContext): Xây dựng giao diện.  - onGoogleSignIn(): Xử lý đăng nhập Google. |
| LoginScreen | User Interface | Hiển thị giao diện đăng nhập. | TextEditingController usernameController, passwordController | - build(BuildContext): Xây dựng giao diện.  - onLoginPressed(): Gửi thông tin đăng nhập. |
| Registration | User Interface | Hiển thị giao diện đăng ký. | TextEditingController usernameController, emailController, passwordController | - build(BuildContext): Xây dựng giao diện.  - onSignupPressed(): Gửi thông tin đăng ký. |
| ExpenseData | Finance Management | Lưu trữ dữ liệu phân tích chi tiêu. | double totalExpense, List<double> monthlyExpenses | - calculateTotal(): Tính tổng chi tiêu.  - toJson(): Chuyển thành JSON. |
| Transaction | Finance Management | Lưu trữ thông tin giao dịch. | String id, double amount, String type, DateTime date | - factory Transaction.fromJson(): Tạo Transaction từ JSON.  - toJson(): Chuyển thành JSON. |
| AddTrans | User Interface | Hiển thị giao diện thêm giao dịch. | TextEditingController amountController, String type | - build(BuildContext): Xây dựng giao diện.  - onSubmit(): Gửi dữ liệu giao dịch. |
| AppTheme | Theme Management | Xác định chủ đề giao diện của ứng dụng. | ThemeData lightTheme, ThemeData darkTheme | - getTheme(bool isDark): Trả về chủ đề dựa trên chế độ sáng/tối. |
| BudgetScreen | User Interface | Hiển thị giao diện quản lý ngân sách. | double budget, double remainingBudget | - build(BuildContext): Xây dựng giao diện.  - refresh(): Cập nhật dữ liệu ngân sách. |
| HomeScreen | User Interface | Hiển thị giao diện chính (tổng quan tài chính). | double balance, double totalIncome, double totalExpense | - build(BuildContext): Xây dựng giao diện.  - refreshData(): Cập nhật dữ liệu tổng quan. |
| MainScreen | User Interface | Giao diện chính điều hướng đến các màn hình khác. | int currentIndex | - build(BuildContext): Xây dựng giao diện.  - onTabChanged(int): Chuyển đổi giữa các tab. |
| ProfileScreen | User Interface | Hiển thị và cập nhật thông tin cá nhân. | TextEditingController usernameController, emailController | - build(BuildContext): Xây dựng giao diện.  - onUpdatePressed(): Gửi thông tin cập nhật. |
| TransactionScreen | User Interface | Hiển thị danh sách giao dịch. | List<Transaction> transactions | - build(BuildContext): Xây dựng giao diện.  - filterTransactions(String type): Lọc giao dịch theo loại. |
| BudgetService | Finance Management | Quản lý dữ liệu ngân sách. | FirebaseFirestore db | - Future<double> getBudgetStatus(String userId): Lấy trạng thái ngân sách.  - setBudget(Map): Cập nhật ngân sách. |
| TransactionService | Finance Management | Quản lý dữ liệu giao dịch. | FirebaseFirestore db | - Future<List<Transaction>> getTransactions(String userId): Lấy giao dịch.  - addTransaction(Map): Thêm giao dịch. |
| AnimatedPieChart | User Interface | Hiển thị biểu đồ tròn động để trực quan hóa dữ liệu tài chính. | List<double> data, List<Color> colors | - build(BuildContext): Xây dựng biểu đồ tròn. |
| FirebaseOptions | Configuration | Cấu hình kết nối với Firebase. | String apiKey, String appId | - getOptions(): Trả về cấu hình Firebase. |

### 3.1.2. Xác định các hệ thống con và giao diện (Identify subsystems and interfaces)

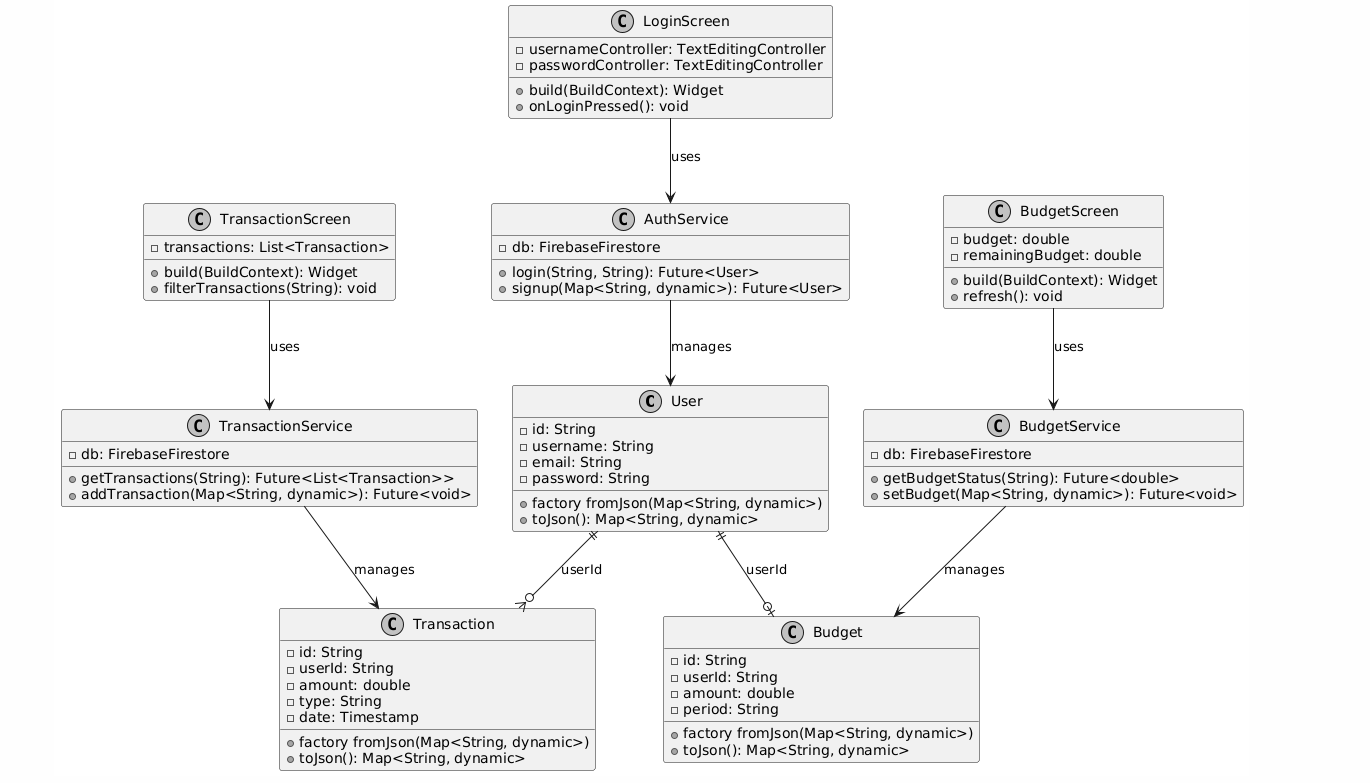
|  |  |  |  |
| --- | --- | --- | --- |
| Tên Hệ Thống Con (Subsystem) | Trách Nhiệm Chính (Responsibilities) | Giao Diện Cung Cấp (Provided Interfaces) | Giao Diện Yêu Cầu (Required Interfaces) |
| Authentication | Quản lý xác thực người dùng (đăng nhập, đăng ký, khôi phục mật khẩu, Google Sign-In). | - IAuthService: login(), signup(), recoverPassword(), googleSignIn(). | - IUserRepository: Truy cập dữ liệu người dùng (Firebase). |
| Finance Management | Quản lý tài chính (giao dịch, ngân sách, phân tích chi tiêu). | - ITransactionService: getTransactions(), addTransaction().<br>- IBudgetService: getBudgetStatus(), setBudget(). | - ITransactionRepository, IBudgetRepository: Truy cập dữ liệu giao dịch và ngân sách (Firebase). |
| User Interface | Hiển thị giao diện và thu thập dữ liệu từ người dùng. | - IWidgetBuilder: build() (cho các màn hình và widget). | - IAuthService, ITransactionService, IBudgetService: Lấy dữ liệu và logic từ các hệ thống con khác. |
| Theme Management | Quản lý chủ đề giao diện của ứng dụng. | - IThemeService: getTheme(bool isDark). | - Không yêu cầu giao diện cụ thể. |
| Configuration | Cung cấp cấu hình cho các dịch vụ bên ngoài (Firebase). | - IConfigService: getOptions(). | - Không yêu cầu giao diện cụ thể. |

### 3.1.3. Xác định các gói (Identify packages)

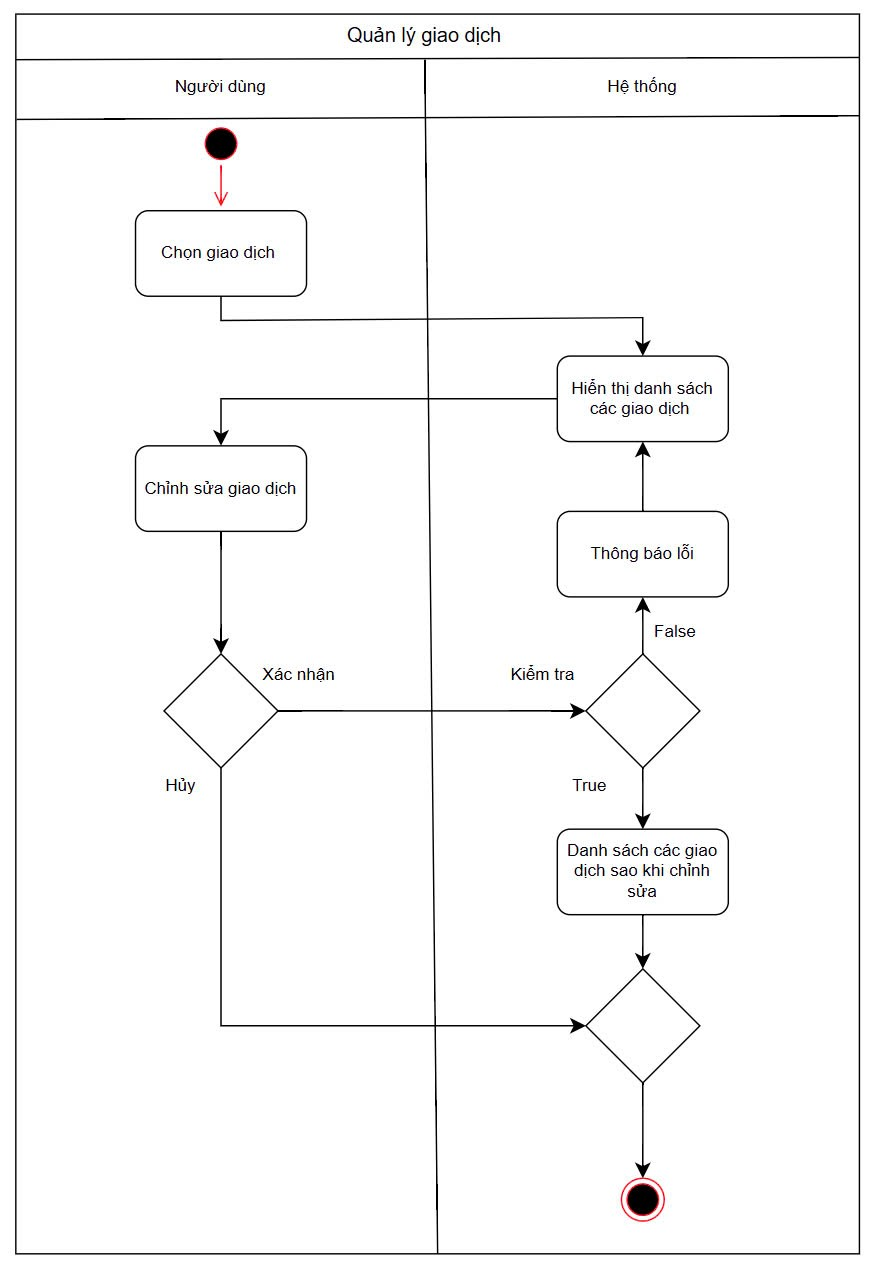
|  |  |  |  |
| --- | --- | --- | --- |
| Tên Gói (Package) | Mục Đích (Purpose) | Các Lớp Liên Quan (Related Classes) | Mối Quan Hệ Với Các Gói Khác (Relationships with Other Packages) |
| auth | Quản lý các màn hình và logic liên quan đến xác thực người dùng. | AuthWrapper, ForgotPassword, GoogleSignIn, LoginScreen, Registration | - Sử dụng services để gọi AuthService.  - Sử dụng models để định dạng dữ liệu người dùng. |
| models | Chứa các lớp mô hình dữ liệu (Entity). | ExpenseData, Transaction | - Được sử dụng bởi services, screens để truy cập dữ liệu.  - Không phụ thuộc vào gói nào khác. |
| screens | Chứa các lớp giao diện người dùng (Boundary). | AddTrans, BudgetScreen, HomeScreen, MainScreen, ProfileScreen, TransactionScreen | - Phụ thuộc vào services để lấy dữ liệu.  - Sử dụng models để hiển thị dữ liệu.  - Sử dụng widgets để tái sử dụng UI. |
| services | Chứa các lớp dịch vụ để giao tiếp với Firebase (hoặc API). | BudgetService, TransactionService | - Sử dụng models để định dạng dữ liệu.  - Sử dụng FirebaseOptions từ widgets để cấu hình Firebase. |
| theme | Quản lý chủ đề giao diện của ứng dụng. | AppTheme | - Được sử dụng bởi screens để áp dụng chủ đề cho giao diện.  - Không phụ thuộc vào gói nào khác. |
| widgets | Chứa các widget tái sử dụng và cấu hình Firebase. | AnimatedPieChart, FirebaseOptions | - AnimatedPieChart được sử dụng bởi screens để hiển thị biểu đồ.  - FirebaseOptions được sử dụng bởi services để cấu hình Firebase. |

## 3.2. Thiết kế trường hợp sử dụng (Use-case design)

### 3.2.1. Thiết kế biểu đồ lớp (Class diagrams)

****

### 3.2.2. Thiết kế các biểu đồ hoạt động (Activity diagrams)

****

*UC1.1: Quản lí giao dịch*

**A diagram of a flowchart

AI-generated content may be incorrect.**

*UC1.2: Quản lí danh mục*

A diagram of a flowchart

AI-generated content may be incorrect.

*UC2: Xem và phân tích báo cáo tài chính*

A diagram of a flowchart

AI-generated content may be incorrect.

*UC3.1: Thiết lập ngân sách*

A diagram of a diagram

AI-generated content may be incorrect.

*UC3.2: Cảnh báo khi vượt ngân sách*

## 3.3. Thiết kế cơ sở dữ liệu (Database Design)

### 3.3.1. Lược đồ cơ sở dữ liệu

**A diagram of a username

AI-generated content may be incorrect.**

### 3.3.2. Chi tiết các bảng

3.3.2.1. Users

- Mục đích: Lưu trữ thông tin cơ bản của người dùng trong ứng dụng quản lý tài chính. Collection này chứa dữ liệu cá nhân của người dùng như tên, email, và mật khẩu, được sử dụng để xác thực và quản lý hồ sơ.

- Vai trò trong hệ thống: Là thực thể trung tâm, liên kết với các giao dịch (transactions) và ngân sách (budgets) thông qua trường userId.

- Cấu trúc dữ liệu:

| Tên Trường (Field) | Kiểu Dữ Liệu | Mô tả | Bắt Buộc (Required) | Ghi Chú |
| --- | --- | --- | --- | --- |
| id | String | Định danh duy nhất của người dùng (khóa chính). | Có | Tự động tạo bởi Firestore hoặc từ UID của Firebase Authentication. |
| username | String | Tên người dùng. | Có | Được nhập khi đăng ký. |
| email | String | Địa chỉ email của người dùng. | Có | Sử dụng cho xác thực qua Firebase Authentication. |
| password | String | Mật khẩu người dùng (đã mã hóa). | Có | Lưu dưới dạng hash (nếu không dùng Firebase Authentication). |

- Mối quan hệ:

* Liên kết với transactions: Mối quan hệ 1-nhiều. Một người dùng có thể có nhiều giao dịch, được tham chiếu qua trường userId trong collection transactions.
* Liên kết với budgets: Mối quan hệ 1-1. Một người dùng có một ngân sách, được tham chiếu qua trường userId trong collection budgets

- Ví dụ dữ liệu:

{

"id": "user123",

"username": "john\_doe",

"email": "john@example.com",

"password": "hashed\_password"

}

- Lưu ý triển khai trong Flutter/Dart:

* Tích hợp Firebase Authentication: Sử dụng Firebase Authentication để quản lý email và password. id có thể được lấy từ firebaseAuth.currentUser?.uid.
* Truy vấn: Sử dụng FirebaseFirestore.instance.collection('users').doc(userId) để truy cập tài liệu người dùng cụ thể.
* Bảo mật: Sử dụng Firestore Security Rules để giới hạn truy cập chỉ cho người dùng đã xác thực (ví dụ: allow read, write: if request.auth != null && request.auth.uid == userId;).
* Cập nhật: Dữ liệu có thể được cập nhật qua phương thức set() hoặc update() trong Flutter (ví dụ: docRef.update({'username': 'new\_name'});).

3.3.2.2. Transactions

**-** Mục đích: Lưu trữ thông tin về các giao dịch tài chính (thu nhập hoặc chi tiêu) của người dùng. Collection này hỗ trợ tính năng theo dõi và phân tích tài chính trong ứng dụng.

- Vai trò trong hệ thống: Là thực thể phụ thuộc, liên kết với users để xác định giao dịch thuộc về người dùng nào.

- Cấu trúc dữ liệu

| Tên Trường (Field) | Kiểu Dữ Liệu | Mô tả | Bắt Buộc (Required) | Ghi Chú |
| --- | --- | --- | --- | --- |
| id | String | Định danh duy nhất của giao dịch (khóa chính). | Có | Tự động tạo bởi Firestore. |
| userId | String | Tham chiếu đến ID của người dùng trong users. | Có | Đảm bảo tính toàn vẹn dữ liệu. |
| amount | double | Số tiền của giao dịch. | Có | Giá trị dương hoặc âm (tùy type). |
| type | String | Loại giao dịch (thu nhập hoặc chi tiêu). | Có | Giá trị cố định: income, expense. |
| date | Timestamp | Thời gian thực hiện giao dịch. | Có | Sử dụng Timestamp.now() trong Flutter. |

**-** Mối quan hệ

* Liên kết với users: Mối quan hệ nhiều-1. Nhiều giao dịch thuộc về một người dùng, được tham chiếu qua userId.

- Ví dụ dữ liệu

{

"id": "trans123",

"userId": "user123",

"amount": 500.0,

"type": "expense",

"date": "2025-03-16T10:00:00Z"

}

**-** Lưu ý triển khai trong Flutter/Dart

* Thêm giao dịch: Sử dụng FirebaseFirestore.instance.collection('transactions').add({...}) để thêm giao dịch mới.
* Truy vấn: Lấy danh sách giao dịch của người dùng bằng collection('transactions').where('userId', isEqualTo: userId).get().
* Bảo mật: Áp dụng Security Rules để chỉ cho phép người dùng truy cập giao dịch của chính họ (ví dụ: allow read, write: if request.auth != null && request.auth.uid == resource.data.userId;).
* Tối ưu hóa: Tạo index cho userId và date nếu cần sắp xếp hoặc lọc giao dịch theo thời gian.
* Xử lý lỗi: Kiểm tra tính toàn vẹn của userId trước khi thêm giao dịch để tránh lỗi tham chiếu.

3.3.2.3. Budgets

**-** Mục đích: Lưu trữ thông tin về ngân sách tài chính của người dùng, bao gồm số tiền và chu kỳ (hàng tháng hoặc hàng năm). Collection này hỗ trợ tính năng quản lý ngân sách trong ứng dụng.

- Vai trò trong hệ thống: Là thực thể phụ thuộc, liên kết với users để xác định ngân sách thuộc về người dùng nào.

- Cấu trúc dữ liệu

| Tên Trường (Field) | Kiểu Dữ Liệu | Mô tả | Bắt Buộc (Required) | Ghi Chú |
| --- | --- | --- | --- | --- |
| id | String | Định danh duy nhất của ngân sách (khóa chính). | Có | Tự động tạo bởi Firestore. |
| userId | String | Tham chiếu đến ID của người dùng trong users. | Có | Đảm bảo tính toàn vẹn dữ liệu. |
| amount | double | Tổng số tiền ngân sách. | Có | Giá trị dương. |
| period | String | Chu kỳ ngân sách. | Có | Giá trị cố định: monthly, yearly. |

**-** Mối quan hệ

* Liên kết với users: Mối quan hệ 1-1. Một người dùng chỉ có một ngân sách, được tham chiếu qua userId.

- Ví dụ dữ liệu

{

"id": "budget123",

"userId": "user123",

"amount": 1000.0,

"period": "monthly"

}

- Lưu ý triển khai trong Flutter/Dart

* Cập nhật ngân sách: Sử dụng FirebaseFirestore.instance.collection('budgets').doc(budgetId).update({...}) để cập nhật số tiền hoặc chu kỳ.
* Truy vấn: Lấy ngân sách của người dùng bằng collection('budgets').where('userId', isEqualTo: userId).get(). Vì mối quan hệ 1-1, đảm bảo chỉ trả về một tài liệu.
* Bảo mật: Sử dụng Security Rules để giới hạn truy cập (ví dụ: allow read, write: if request.auth != null && request.auth.uid == resource.data.userId;).
* Tính toán: Ngân sách có thể được so sánh với tổng chi tiêu từ transactions để cảnh báo khi vượt ngân sách (xử lý trong BudgetService).
* Khởi tạo: Khi người dùng mới đăng ký, tự động tạo một tài liệu trong budgets với giá trị mặc định (ví dụ: amount: 0.0, period: "monthly").

# **4.Cài đặt**

## 4.1. Công nghệ sử dụng

- Flutter và các gói hỗ trợ

* Flutter: Framework mã nguồn mở của Google, cho phép phát triển ứng dụng đa nền tảng (iOS và Android) với một codebase duy nhất. Flutter được chọn vì hiệu suất cao và khả năng tùy chỉnh giao diện.
* Device Preview: Công cụ hỗ trợ kiểm tra giao diện trên nhiều kích thước màn hình khác nhau trực tiếp trong quá trình phát triển.
* FL Chart: Thư viện vẽ biểu đồ mạnh mẽ, được tích hợp để hiển thị dữ liệu giao dịch dưới dạng trực quan (biểu đồ cột, đường, tròn, v.v.).
* StreamBuilder: Widget của Flutter, kết hợp với Firebase để cập nhật giao diện theo thời gian thực khi dữ liệu thay đổi.
* Intl Package: Được sử dụng để định dạng và hiển thị ngày giờ theo chuẩn quốc tế, đảm bảo giao diện thân thiện với người dùng ở các khu vực khác nhau.
* Flutter Dev Tool: Công cụ tích hợp trong Visual Studio Code, hỗ trợ debug, kiểm tra hiệu suất và tối ưu hóa ứng dụng.

- Firebase

* Firebase Authentication (FirebaseAuth): Hệ thống xác thực người dùng, hỗ trợ đăng nhập qua email/mật khẩu, Google, và các phương thức khác. Giao diện Firebase Authentication UI Animation được tùy chỉnh để tăng trải nghiệm người dùng.
* Cloud Firestore: Cơ sở dữ liệu NoSQL thời gian thực, dùng để lưu trữ và quản lý thông tin giao dịch. Firestore cho phép đồng bộ dữ liệu giữa các thiết bị một cách nhanh chóng.
* Firestore Security Rules: Quy tắc bảo mật được thiết lập để đảm bảo chỉ người dùng đã xác thực mới có quyền truy cập và chỉnh sửa dữ liệu của họ, ngăn chặn truy cập trái phép.

- Công cụ phát triển

* Visual Studio Code: IDE chính để viết mã, tích hợp tốt với Flutter và Firebase thông qua các extension như Flutter Dev Tool.
* Google API: Được sử dụng để kết nối Firebase với các dịch vụ khác của Google (nếu cần mở rộng tính năng trong tương lai).

- Mô hình và dịch vụ

* TransactionModel: Lớp dữ liệu định nghĩa cấu trúc của một giao dịch, bao gồm các thuộc tính như số tiền, ngày giờ, loại giao dịch, v.v.
* TransactionService: Lớp dịch vụ trung gian giữa giao diện và Firestore, xử lý các thao tác CRUD (Create, Read, Update, Delete) trên dữ liệu giao dịch.

## 4.2. Cấu trúc mã nguồn

BTL\_PTTKPM/

├── android/ # Cấu hình Android

├── assets/ # Tài nguyên tĩnh (hình ảnh, font, v.v.)

├── ios/ # Cấu hình iOS

├── lib/ # Thư mục chứa mã nguồn chính

│ ├── auth/ # Quản lý xác thực người dùng

│ │ ├── auth\_wrapper.dart # Điều hướng dựa trên trạng thái xác thực

│ │ ├── forgot\_password.dart # Màn hình khôi phục mật khẩu

│ │ ├── google\_signin.dart # Đăng nhập bằng Google

│ │ ├── login\_screen.dart # Màn hình đăng nhập

│ │ ├── registration.dart # Màn hình đăng ký

│ ├── models/ # Các lớp mô hình dữ liệu

│ │ ├── expense\_data.dart # Dữ liệu phân tích chi tiêu

│ │ ├── transaction.dart # Lớp mô hình giao dịch

│ ├── screens/ # Các màn hình giao diện người dùng

│ │ ├── add\_trans.dart # Màn hình thêm giao dịch

│ │ ├── app\_theme.dart # Định nghĩa chủ đề giao diện (nên chuyển vào theme/)

│ │ ├── budget\_screen.dart # Màn hình quản lý ngân sách

│ │ ├── home\_screen.dart # Màn hình chính (tổng quan)

│ │ ├── main\_screen.dart # Màn hình chính điều hướng

│ │ ├── profile\_screen.dart # Màn hình hồ sơ cá nhân

│ │ ├── transactions\_screen.dart # Màn hình danh sách giao dịch

│ ├── services/ # Các dịch vụ giao tiếp với Firebase

│ │ ├── budget\_service.dart # Quản lý dữ liệu ngân sách

│ │ ├── transaction\_service.dart # Quản lý dữ liệu giao dịch

│ ├── theme/ # Quản lý chủ đề giao diện

│ │ ├── app\_theme.dart # Định nghĩa chủ đề (trùng lặp, nên hợp nhất)

│ ├── widgets/ # Các widget tái sử dụng

│ │ ├── animated\_pie\_chart.dart # Widget biểu đồ tròn động

│ │ ├── firebase\_options.dart # Cấu hình Firebase

│ ├── main.dart # Điểm khởi đầu của ứng dụng

├── linux/ # Cấu hình Linux

├── macos/ # Cấu hình macOS

├── node\_modules/ # Thư mục phụ thuộc Node.js (không cần trong Flutter)

├── test/ # Thư mục chứa các bài kiểm tra

├── web/ # Cấu hình Web

├── windows/ # Cấu hình Windows

├── .gitignore # File cấu hình bỏ qua Git

├── .metadata # Metadata của IDE (tự động tạo)

├── analysis\_options.yaml # Cấu hình phân tích mã nguồn

├── devtools\_options.yaml # Cấu hình DevTools

├── firebase.json # Cấu hình Firebase Hosting (nếu có)

├── package-lock.json # File khóa phụ thuộc Node.js (không cần trong Flutter)

├── package.json # File cấu hình Node.js (không cần trong Flutter)

├── pubspec.lock # File khóa phụ thuộc Flutter

├── pubspec.yaml # File cấu hình phụ thuộc Flutter

├── README.md # Tài liệu mô tả dự án

# **5. Kết luận**

**-** Phần mềm "Quản lý chi tiêu cá nhân" được thiết kế và phát triển nhằm cung cấp một công cụ hữu ích giúp người dùng kiểm soát tài chính cá nhân hiệu quả hơn. Thông qua việc tự động ghi nhận các giao dịch, phân loại chi tiêu, cung cấp biểu đồ thống kê và đưa ra cảnh báo khi vượt ngân sách, hệ thống giúp người dùng có cái nhìn tổng quan và chi tiết về tình hình tài chính của mình.

Quá trình phân tích và thiết kế phần mềm đã được thực hiện một cách hệ thống, từ đặc tả yêu cầu, xây dựng mô hình dữ liệu, thiết kế giao diện cho đến triển khai backend với Firebase. Kiến trúc hệ thống được thiết kế linh hoạt, có tính mở rộng cao, đảm bảo khả năng hoạt động ổn định và bảo mật dữ liệu người dùng.

Nhìn chung, phần mềm này không chỉ hỗ trợ cá nhân trong việc kiểm soát chi tiêu mà còn có tiềm năng phát triển thêm nhiều tính năng nâng cao như tích hợp AI để phân tích hành vi tiêu dùng, đồng bộ hóa với tài khoản ngân hàng, hoặc hỗ trợ lập kế hoạch tài chính dài hạn. Trong tương lai, việc mở rộng tính năng và tối ưu hóa hiệu suất sẽ giúp phần mềm tiếp cận được nhiều người dùng hơn, góp phần thúc đẩy thói quen quản lý tài chính thông minh.

# **TÀI LIỆU THAM KHẢO**

1. Sommerville, I. (2015). *Software Engineering* (10th Edition). Pearson Education.
2. Pressman, R. S. (2019). *Software Engineering: A Practitioner's Approach* (9th Edition). McGraw-Hill.
3. Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.
4. Firebase Documentation. (2024). *Cloud Firestore Guide*. Truy cập từ: https://firebase.google.com/docs/firestore
5. Google Developers. (2024). *Flutter Documentation*. Truy cập từ: https://docs.flutter.dev
6. IEEE Standard for Software Requirement Specifications (IEEE 830-1998).