

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC NGUYỄN TẤT THÀNH
KHOA CÔNG NGHỆ THÔNG TIN



KHÓA LUẬN TỐT NGHIỆP

**PHÁT TRIỂN HỆ THỐNG WEBSITE QUẢN LÝ
TIÊU DÙNG CÁ NHÂN BẰNG REACTJS VÀ
POSTGRESQL**

Giảng viên hướng dẫn : ThS. PHẠM VĂN ĐĂNG

Sinh viên thực hiện : HÀ VĂN THỌ

MSSV : 2200006616

Khoa : 2022

Ngành : KỸ THUẬT PHẦN MỀM

Chuyên ngành : CÔNG NGHỆ KỸ THUẬT PHẦN MỀM

Tp.HCM, tháng 12 năm 2025

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC NGUYỄN TẤT THÀNH
KHOA CÔNG NGHỆ THÔNG TIN



KHÓA LUẬN TỐT NGHIỆP

**PHÁT TRIỂN HỆ THỐNG WEBSITE QUẢN LÝ
TIÊU DÙNG CÁ NHÂN BẰNG REACTJS VÀ
POSTGRESQL**

Giảng viên hướng dẫn : ThS. PHẠM VĂN ĐĂNG

Sinh viên thực hiện : HÀ VĂN THỌ

MSSV : 2200006616

Khoa : 2022

Ngành : KỸ THUẬT PHẦN MỀM

Chuyên ngành : CÔNG NGHỆ KỸ THUẬT PHẦN MỀM

Tp.HCM, tháng 12 năm 2025

LỜI MỞ ĐẦU

Trong vài năm gần đây, cụm “Chuyển đổi số” xuất hiện nhiều lần trong đời sống. Công nghệ tiên tiến ở khắp nơi và ảnh hưởng rất nhiều hoạt động thường ngày từ công việc và học tập. Một trong số rất nhiều lĩnh vực chịu ảnh hưởng mạnh mẽ về tài chính cá nhân từ xu hướng này – tưởng chừng đơn giản và ai cũng tự làm được, lại đang trở thành vấn đề không hề dễ dàng như mọi người nghĩ.

Thực tế thì khi ghi chép thu chi không phải ai cũng có thói quen để có thể duy trì được, và ngay cả khi cách làm truyền thống như sổ hay bảng tính đôi lúc khá là phức tạp. Dữ liệu được lưu lại nhiều hơn, nhưng nó vẫn nằm yên khiến cho người dùng khó có thể rút ra được nhận định hay hướng điều chỉnh phù hợp nhất.

Trong khi đó, trên nền tảng công nghệ phát triển có nhiều các ứng dụng quản lý tài chính đang có trên thị trường, dù rất đa dạng, nhưng không phải ứng dụng nào cũng đáp ứng được nhu cầu thiết yếu của người. Một số ứng dụng quá nhiều thao tác, gây khó khăn cho người mới và thiếu các tính năng thông minh giúp người dùng hiểu hành vi chi tiêu của bản thân. Điều này có thể tạo ra một khoảng trống: người dùng cần một công cụ đơn giản, rõ ràng và trực quan nhưng vẫn đủ thông minh để mang lại giá trị trong quá trình quản lý tài chính.

Từ những trải nghiệm và quan sát, em bắt đầu có ý tưởng xây dựng một hệ thống quản lý tài chính cá nhân theo hướng thân thiện hơn với người dùng và có phần “lắng nghe” người dùng, hộ trợ và thậm chí đưa ra gợi ý khi cần thiết. Từ đó, đề tài “**Phát triển hệ thống Website quản lý tiêu dùng cá nhân bằng ReactJS và PostgreSQL**”.

Điểm khác biệt trong đề tài mà em muốn hướng tới không chỉ dừng lại tại việc ghi chép thu – chi đơn thuần, mà còn kết hợp thêm một chatbot sử dụng AI để hỗ trợ người dùng nhập liệu nhanh hơn, giải thích các báo cáo và thậm chí đưa ra vài gợi ý khi cần thiết.

LỜI CẢM ƠN

Để hoàn thành tốt khóa luận tốt nghiệp này, trước hết em xin gửi lời cảm ơn sâu sắc đến Thầy ThS.Phạm Văn Đăng, giảng viên đã tận tình hướng dẫn và đồng hành cùng em trong suốt quá trình thực hiện đề tài. Sự hỗ trợ, giải đáp tận tình những thắc mắc và những góp ý quý báu từ thầy đã giúp em khắc phục được những hạn chế, hoàn thiện khóa luận đúng tiêu bộ mà Khoa đã đề ra.

Em cũng xin gửi lời cảm ơn chân thành đến quý thầy cô trong Khoa Công nghệ Thông tin và Trường Đại học Nguyễn Tất Thành vì đã tạo điều kiện thuận lợi và môi trường học tập lý tưởng để em có thể phát huy tối đa năng lực bản thân trong quá trình làm đồ án. Em cảm thấy vô cùng may mắn và tự hào khi được học tập tại một môi trường giáo dục hiện đại, năng động, giàu tính thực tiễn, nơi luôn khuyến khích sinh viên chủ động sáng tạo và phát triển toàn diện cả về chuyên môn lẫn kỹ năng mềm.

Bên cạnh đó, em xin gửi lời cảm ơn chân thành đến gia đình và bạn bè – những người luôn bên cạnh, ủng hộ và động viên em trong suốt quá trình học tập và thực hiện đồ án. Chính sự tin tưởng và động viên kịp thời của mọi người là nguồn động lực to lớn giúp em vượt qua những thử thách, giữ vững tinh thần và nỗ lực hoàn thành tốt công việc được giao.

Một lần nữa, em xin trân trọng cảm ơn tất cả những ai đã đồng hành và hỗ trợ em trong hành trình học tập này. Em sẽ luôn ghi nhớ và trân trọng những giá trị đã nhận được để tiếp tục phấn đấu và đóng góp tích cực cho cộng đồng trong tương lai.

Tp.HCM, Ngày tháng năm 2025

Em xin chân thành cảm ơn!

Sinh viên thực hiện

Hà Văn Thọ

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

Tp.HCM, Ngày tháng năm 2025

Giáo viên hướng dẫn

(Ký tên, ghi rõ họ tên)

ThS.PHẠM VĂN ĐĂNG

NHẬN XÉT CỦA GIẢNG VIÊN PHẢN BIỆN

Tp.HCM, Ngày tháng năm 2025

Giáo viên phản biện

(Ký tên, ghi rõ họ tên)

THS. LUONG TRUONG AN

MỤC LỤC

LỜI MỞ ĐẦU	i
LỜI CẢM ƠN	ii
CHƯƠNG 1 : TỔNG QUAN VỀ ĐỀ TÀI	1
1.1 Khảo sát hiện trạng và đặt vấn đề	1
1.2 Mục tiêu của đề tài	2
1.3 Đối tượng nghiên cứu.....	2
1.4 Phạm vi nghiên cứu	4
1.5 Ý nghĩ thực tiễn của đề tài.....	6
1.6 Bố cục khóa luận	6
CHƯƠNG 2: CƠ SỞ LÝ THUYẾT	8
2.1 Về Phía Frontend: ReactJS và Hệ sinh thái	8
2.2 Về Phía Backend: Python FastAPI	12
2.3 Cơ sở dữ liệu: PostgreSQL	17
2.4 Trí tuệ nhân tạo (AI Intergration)	20
2.5 Xác Thực và Bảo mật với Firebase	22
CHƯƠNG 3: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG.....	26
3.1 Phân tích và thiết kế hệ thống.....	26
3.2 Sơ đồ chức năng Mind Map	32
3.3 Sơ đồ Use – Case	34
3.4 Đặc tả chi tiết Use – Case	37
3.5 Thiết kế cơ sở dữ liệu	46
CHƯƠNG 4: THỰC NGHIỆM VÀ TRIỂN KHAI HỆ THỐNG	57
4.1 Môi trường cài đặt và công nghệ sử dụng	57
4.2 Tổ chức mã nguồn	59
4.3 Thiết kế giao diện chương trình.....	61
KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	87

DANH MỤC BẢNG

Bảng 3. 1 Bảng phân rã chức năng cho Người Dùng (User)	29
Bảng 3. 2 Bảng Phân rã chức năng cho Quản trị viên (Admin)	31
Bảng 3. 3 Đặc tả Use – Case Đăng nhập (Login)	37
Bảng 3. 4 Đặc tả Use – Case Thêm giao dịch.....	39
Bảng 3. 5 Đặc tả Use-Case xem báo cáo phân tích.....	40
Bảng 3. 6 Đặc tả Use – Case Tương tác với trợ lý Áo Finbot	41
Bảng 3. 7 Đặc tả Use – Case Quản lý người dùng (Admin)	43
Bảng 3. 8 Đặc tả Use-Case Quản lý danh mục hệ thống (Admin)	44
Bảng 3. 9 Đặc tả Use-Case xem nhật ký hoạt động (Audit Logs)	45
Bảng 3. 10 Danh sách các bảng trong CSDL.....	47
Bảng 3. 11 Thiết kế chi tiết bảng Người Dùng (Users)	50
Bảng 3. 12 Thiết kế chi tiết bảng Danh Mục (categories)	51
Bảng 3. 13 Thiết kế chi tiết bảng Thu Nhập (Incomes)	52
Bảng 3. 14 Thiết kế chi tiết bảng Chi Tiêu (expenses)	53
Bảng 3. 15 Thiết kế chi tiết bảng Giao Dịch Tổng hợp (transactions)	54
Bảng 3. 16 Thiết kế chi tiết bảng Nhật Ký Hệ Thống (Audit Logs).....	55
Bảng 3. 17 Thiết kế chi tiết bảng Cấu Hình Hệ Thống (System_settings)	56
Bảng 4. 1 Đặc tả các thành phần Giao diện Đăng Nhập	62
Bảng 4. 2 Đặc tả các thành phần giao diện Đăng Ký.....	64
Bảng 4. 3 Đặc tả các thành phần Sidebar.....	68
Bảng 4. 4 Đặc tả các thành phần trên Dashboard	69
Bảng 4. 5 Đặc tả các thành phần trên trang Analytics	71
Bảng 4. 6 Đặc tả các thành phần trên trang Danh Mục (Category)	74
Bảng 4. 7 Đặc tả các thành phần trên trang Bảo Mật (Security)	76
Bảng 4. 8 Đặc tả các thành phần trên trang Hồ sơ (Profile)	78
Bảng 4. 9 Đặc tả các thành phần trên trang Admin Dashboard	80

Bảng 4. 10 Đặc tả các thành phần trên trang Quản lý người dùng (Admin)	82
Bảng 4. 11 Đặc tả các thành phần trên trang Danh mục hệ thống (Admin)	83
Bảng 4. 12 Đặc tả các thành phần trên trang Nhật Ký Hoạt Động (Admin)	84
Bảng 4. 13 Đặc tả các thành phần trên trang Cấu Hình Hệ Thông (Admin)	86

DANH MỤC HÌNH

Hình 2. 1 React (ReactJS)	8
Hình 2. 2 DOM vs Virtua DOM	9
Hình 2. 3 Tailwind	11
Hình 2. 4 Python	13
Hình 2. 5 FastAPI.....	14
Hình 2. 6 SQLAlchemy.....	15
Hình 2. 7 Pydantic.....	15
Hình 2. 8 PostgreSQL	18
Hình 2. 9 Google Gemini	20
Hình 2. 10 LangChain Framework.....	22
Hình 2. 11 Firebase	24
Hình 3. 1 Sơ đồ chức năng.....	32
Hình 3. 2 Sơ đồ kiến trúc hệ thống	33
Hình 3. 3 Sơ đồ Use-Case Tổng quát.....	35
Hình 3. 4 Sơ đồ Use – Case người dùng	36
Hình 3. 5 Phân rã Use – Case Quản trị viên	37
Hình 3. 6 Lược đồ CSDL quan hệ - ER Diagram	46
Hình 3. 7 Sơ đồ tổng quát ERD ở mức khái niệm	48
Hình 4. 1 Sơ đồ thư mục Backend	59
Hình 4. 2 Sơ đồ cây thư mục Frontend	60
Hình 4. 3 Giao diện trang Đăng nhập	62
Hình 4. 4 Giao diện trang Đăng Ký	64
Hình 4. 5 Thanh Sidebar Admin	66
Hình 4. 6 Thanh Sidebar Users	67
Hình 4. 7 Giao diện trang Dashboard (Users).....	69
Hình 4. 8 Giao diện trang Analytics	71

Hình 4. 9 Giao diện trang Category	73
Hình 4. 10 Giao diện Thêm Danh Mục Mới	73
Hình 4. 11 Giao diện Sửa Danh Mục	74
Hình 4. 12 Giao diện trang Security	76
Hình 4. 13 Giao diện trang cá nhân (Profile)	78
Hình 4. 14 Giao diện trang Admin Dashboard	80
Hình 4. 15 Giao diện trang Quản lý người dùng (Admin)	81
Hình 4. 16 Giao diện trang Danh mục hệ thống (Admin)	83
Hình 4. 17 Giao diện trang Nhật Ký Hoạt Động (Admin)	84
Hình 4. 18 Giao diện trang Cấu hình hệ thống	86

KÍ HIỆU CÁC CỤM TỪ VIẾT TẮT

Chữ viết tắt	Ý nghĩa
SPA	Single Page Application
LLM	Large Language Model (Mô hình ngôn ngữ lớn)
DB	Database
DOM	Document Object Model
CRUD	(Create, Read, Update, Delete)

CHƯƠNG 1 : TÔNG QUAN VỀ ĐỀ TÀI

1.1 Khảo sát hiện trạng và đặt vấn đề

Trong bối cảnh chuyển đổi số ngày càng mạnh mẽ, chuyện tiền bạc luôn vần đè khiến cho nhiều người không thoái mái cho nên việc quản lý tài chính các nhân vô cùng cấp thiết đối với nhiều người. Không chỉ là chuyện kiểm thêm thu nhập bao nhiêu mà quan trọng với việc giữ bao nhiêu, đặc biệt là các bạn sinh viên hay những người trẻ mới đi làm. Việc chi tiêu thu nhập bao nhiêu thường không được chú trọng dẫn đến tình trạng rơi vào cảnh cuối tháng mì tôm hoặc thiếu hụt và không nắm rõ tiền đã “đi” vào những khoản nào.

Trước đây, việc ghi chép thu chi bằng sổ tay và dần dần chuyển sang qua các công cụ như Excel. Tuy nhiên với thời đại công nghệ tiên tiến phát triển, người dùng sử dụng các ứng dụng trên điện thoại, mặc dù xuất hiện nhiều và đa dạng, nhưng không phải ứng dụng nào cũng có thể phù hợp với nhu cầu thực tế. Một vài ứng dụng có những giao diện phức tạp, nhiều tính năng lại không cần thiết khiến cho người dùng cảm thấy khó tiếp cận, các ứng dụng khác lại yêu cầu nâng cấp trả phí để sử dụng đầy đủ các chức năng hoặc hiển thị quảng cáo gây nhiễu hoặc gián đoạn khi trải nghiệm ứng dụng. Vấn đề mà người dùng hay gặp là phải nạp tiền để mở khóa chức năng (Premium), hoặc chặn các quảng cáo hiện liên tục gây khó chịu khi trải nghiệm.

Hơn nữa là vấn đề lớn nhất của việc ghi chép thu chi là sự nhảm chán lặp lại quá nhiều. Từ đó nhu cầu về một hệ thống quản lý tài chính vừa đơn giản, dễ sử dụng và vừa có sự thông minh để giảm bớt các thao tác thủ công cho nên em đã nảy ra ý tưởng và quyết định thực hiện đề tài “**Phát triển hệ thống website quản lý tiêu dùng cá nhân bằng ReactJS và PostgreSQL**” tích hợp các công nghệ AI và xử lý ngôn ngữ tự nhiên có thể giúp cho người dùng không chỉ giải quyết bài toán ghi chép cơ bản mà còn là một công cụ hỗ trợ quản lý tài chính trở nên tiện lợi, linh hoạt và dễ thở hơn.

1.2 Mục tiêu của đề tài

Khi bắt đầu vào làm đồ án với đề tài là phát triển một hệ thống quản lý tài chính cá nhân không chỉ muốn tạo ra một trang web CRUD (Thêm, Sửa, Xóa) đơn thuần mà còn hướng tới nền tảng web thân thiện để người dùng có thể sử dụng hàng ngày và thông minh để hỗ trợ họ đưa ra các quyết định phù hợp. Cụ thể đề tài hướng tới đến các mục tiêu như sau:

- **Về giao diện người dùng (Frontend):**

Xây dựng được một website có giao diện trực quan, hiện đại, bắt mắt và có khả năng hiển thị thông tin rõ ràng. Quan trọng là nó phải thuận tiện, dễ dùng và phù hợp với người trẻ hiện nay bô cục rõ ràng để người dùng nhìn vào là hiểu ngay mình còn bao nhiêu và đã tiêu những gì nhiều.

- **Về hệ thống xử lý (Backend):**

Thiết kế và xây dựng một cơ sở dữ liệu vững chắc và ổn định bằng PostgreSQL để đảm bảo an toàn dữ liệu và tính nhất quán. Kết hợp với Python sử dụng FastAPI để phát triển các API, xử lý logic nhanh chóng và chuyên nghiệp.

- **Về tích hợp trợ lý ảo (AI):**

Điểm quan trọng nhất tính năng hỗ trợ thông minh đó là một chatbot AI (FinBot) có khả năng nhận thông tin đầu vào qua câu lệnh tự nhiên với các câu đơn giản như là “Vừa đổ xăng 50k” thì hệ thống sẽ tự động hiểu và ghi nhận vào danh mục phù hợp. Ngoài ra, chatbot AI vẫn có thể vẽ biểu đồ và phân tích cho người dùng xem đang tiêu hao bao nhiêu tiền và đưa ra lời khuyên tổng quan.

- **Về quản trị hệ thống:**

Cuối cùng là xây dựng hệ thống quản trị cho phép quản lý người dùng, danh mục mặc định, theo dõi hoạt động và các thông số quan trọng của hệ thống.

1.3 Đối tượng nghiên cứu

- **Đối tượng sử dụng:** Người dùng có nhu cầu theo dõi việc thu chi, muốn theo dõi để hình thành thói quen quản lý tài chính hàng ngày một cách đơn giản và trực quan. Bên cạnh đó, quản trị viên sẽ là người vận hành hệ thống và kiểm soát dữ liệu.
- **Đối tượng công nghệ**

- **ReactJS:** Thư viện xây dựng giao diện người dùng
- **FastAPI Python:** Framework để tối ưu cho việc viết API xử lý logic và xây dựng API backend tốc độ cao.
- **Firebase:** Dùng để hỗ trợ xác thực người dùng và bảo mật tài khoản.
- **LnagChain và Google Gemini:** Công nghệ lõi để phát triển chatbot AI thông minh nhập liệu và phân tích.

1.3.1 Hệ thống quản lý tài chính trực tuyến

Đề tài nghiên cứu tổng quan mô hình hoạt động của một ứng dụng quản lý tài chính cá nhân (PFM - Personal Financial Management) trong môi trường số, bao gồm:

- **Quy trình vận hành:** Từ việc nhập liệu giao dịch, phân loại danh mục tự động đến việc tổng hợp báo cáo tài chính.
- **Cơ chế tương tác thông minh:** Cách thức kết nối giữa Người dùng và Hệ thống thông qua giao diện web trực quan và Trợ lý ảo AI (FinBot) để đơn giản hóa việc nhập liệu.
- **Các mô-đun hỗ trợ:** Phân tích dữ liệu (Analytics), gợi ý chi tiêu hợp lý và cảnh báo ngân sách.
- **Yêu cầu kỹ thuật:** Đảm bảo tính bảo mật dữ liệu tài chính, độ trễ thấp trong phản hồi và khả năng hoạt động ổn định trên nền tảng web.

Thông qua đó, đề tài phân tích các thành phần cốt lõi để xây dựng một hệ sinh thái quản lý tài chính hiện đại, giải quyết bài toán “lười ghi chép” của người dùng.

1.3.2 Nhóm người dùng và vai trò

Hệ thống được thiết kế phục vụ hai nhóm đối tượng chính với các vai trò và quyền hạn khác biệt:

- Người dùng cá nhân (User): Là đối tượng phục vụ chính của hệ thống, bao gồm các cá nhân có nhu cầu quản lý tài chính. Các hoạt động chính gồm:
 - Đăng ký/Đăng nhập và bảo mật tài khoản (2FA).
 - Ghi chép thu nhập, chi tiêu hàng ngày.
 - Tương tác với Chatbot AI để nhập liệu nhanh và hỏi đáp số dư.

- Theo dõi báo cáo thống kê, biểu đồ xu hướng.
 - Tùy chỉnh danh mục chi tiêu cá nhân.
- Quản trị viên (Admin): Là người vận hành và giám sát hệ thống:
 - Quản lý danh sách người dùng (User Management).
 - Thiết lập các cấu hình hệ thống chung (System Settings).
 - Quản lý danh mục mặc định (Default Categories).
 - Giám sát nhật ký hoạt động (Audit Logs) để đảm bảo an ninh.

1.3.3 Quy trình nghiệp vụ

Đề tài tập trung phân tích và thiết kế các thực thể dữ liệu cốt lõi tạo nên “xương sống” của hệ thống:

- User: Thông tin định danh, cấu hình tiền tệ, trạng thái bảo mật.
 - Transaction (Income/Expense): Dữ liệu chi tiết về dòng tiền (số tiền, ngày tháng, ghi chú, loại giao dịch).
 - Category: Hệ thống phân loại thu chi (Icon, Màu sắc, Loại).
 - Audit Log: Lịch sử hoạt động hệ thống.
 - System Setting: Các tham số cấu hình toàn cục.
- Song song đó là các quy trình nghiệp vụ chính:
- Quy trình xác thực và bảo mật đa lớp (Firebase + JWT).
 - Quy trình xử lý ngôn ngữ tự nhiên (NLP) để chuyển đổi câu chat thành giao dịch.
 - Quy trình tổng hợp và trực quan hóa dữ liệu báo cáo.

1.4 Phạm vi nghiên cứu

1.4.1 Phạm vi chức năng

Đề tài tập trung giải quyết các vấn đề trong phạm vi quản lý tài chính, không mở rộng sang kế toán doanh nghiệp hay các hệ thống quản trị tài chính quy mô lớn.

Phạm vi chức năng

- Quản lý thu nhập và chi tiêu

- Quản lý danh mục
- Báo cáo trực quan theo dạng biểu đồ
- Xuất dữ liệu ra Excel
- Chatbot AI hỗ trợ nhập liệu
- Bảo mật như xác thực 2 yếu tố (2FA)

Phạm vi triển khai

Hệ thống được triển khai dưới dạng ứng dụng web và hỗ trợ giao diện đáp ứng xem tốt cả máy tính và điện thoại.

1.4.2 Phạm vi công nghệ

Đề tài áp dụng bộ công nghệ (Tech Stack) hiện đại và phù hợp xu thế:

- Frontend: ReactJS (Vite) xây dựng giao diện SPA; Tailwind CSS cho thiết kế; Recharts cho biểu đồ.
- Backend: Python FastAPI xây dựng RESTful API hiệu năng cao; SQLAlchemy (ORM) quản lý dữ liệu.
- Cơ sở dữ liệu: PostgreSQL lưu trữ dữ liệu quan hệ đảm bảo tính toàn vẹn (ACID).
- AI & NLP: LangChain kết hợp Google Gemini (Flash Model) để xử lý trí tuệ nhân tạo.
- Bảo mật & Hạ tầng: Firebase Authentication quản lý định danh; Docker (tùy chọn) để triển khai.

1.4.3 Phạm vi người dùng

Hệ thống hướng tới phục vụ:

- Người dùng phổ thông: Sinh viên, nhân viên văn phòng, người làm tự do (Freelancer) cần công cụ quản lý tài chính đơn giản, hiệu quả.
- Quản trị viên: Đội ngũ vận hành kỹ thuật (Dev/Ops) cần công cụ giám sát hệ thống.

1.4.4 Phạm vi giới hạn

Do giới hạn về thời gian và nguồn lực nghiên cứu, đề tài tạm thời chưa triển khai các tính năng sau (được đề xuất cho hướng phát triển tương lai):

- **Ứng dụng di động (Mobile App Native):** Hiện tại chỉ chạy trên trình duyệt web (Web App), dù có hỗ trợ giao diện Responsive.
- **Liên kết ngân hàng (Open Banking):** Chưa hỗ trợ tự động đồng bộ giao dịch từ tài khoản ngân hàng do rào cản về pháp lý và API bên thứ 3.
- **Quét hóa đơn (OCR):** Chưa tích hợp tính năng chụp ảnh hóa đơn để tự động nhập liệu.
- **Lập kế hoạch tài chính dài hạn:** Chưa có các công cụ dự báo dòng tiền hoặc tư vấn đầu tư chuyên sâu.

1.5 Ý nghĩ thực tiễn của đề tài

Khi được triển khai hệ thống này được hoàn thiện vào thực tế, nó sẽ giúp ích hỗ trợ người dùng hình thành thói quen theo dõi quản lý tài chính. Việc tích hợp AI sẽ giúp bớt đi các thao tác thủ công, đồng thời tạo ra việc ghi chép trở nên thú vị hơn như đang trò chuyện.

Đối với người thực hiện đề tài này thì quá trình thực hiện xây dựng hệ thống là cơ hội quan trọng tốt để tổng hợp lại toàn bộ kiến thức đã học: từ thiết kế cơ sở dữ liệu, phát triển Backend – Frontend, viết API và đặc biệt là tiếp cận các công nghệ mới như AI/LLM.

1.6 Bố cục khóa luận

Đề trình bày nội dung khóa luận rõ ràng với quy trình nghiên cứu và thực hiện, báo cáo này được chia thành các chương như sau:

Chương 1: Tổng quan về đề tài

Trình bày bối cảnh, mục tiêu, đối tượng, công nghệ và phạm vi nghiên cứu.

Chương 2: Cơ sở lý thuyết

Giới thiệu các kiến thức nền tảng mà em sử dụng như ReactJS, PostgreSQL, kiến trúc Client-Server và cơ chế hoạt động của các mô hình ngôn ngữ lớn LLM trong Chatbot.

Chương 3: Phân tích và thiết kế hệ thống

Trình bày về các yêu cầu tính năng, sơ đồ Use – Case, thiết kế cơ sở dữ liệu và luồng dữ liệu trong hệ thống.

Chương 4: Thực nghiệm và triển khai

Trình bày các quy trình thực nghiệm, các hình ảnh giao diện thực tế của đồ án và kết quả đạt được.

Kết luận

Đánh giá tổng quan, những gì đã làm được và những hạn chế còn tồn tại và sẽ định hướng nâng cấp trong tương lai.

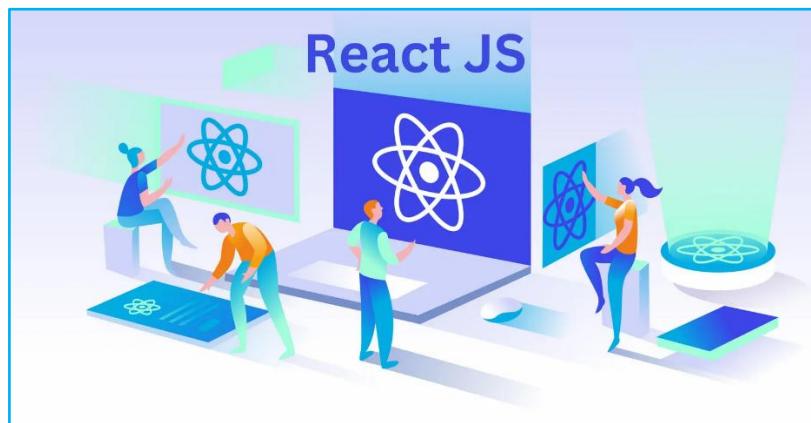
CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1 Về Phía Frontend: ReactJS và Hệ sinh thái

2.1.1 Tổng quan về ReactJS

ReactJS là một thư viện JavaScript mã nguồn mở do Facebook (nay là Meta) phát triển. Khác so với cách viết HTML/JS truyền thống, React tiếp cận để theo hướng **Component-based** (dựa trên thành phần) và đủ mạnh để tạo nên các ứng dụng SPA (Single Page Application) hiện đại [1].

Một trong những lý do em chọn React là kiến trúc *Component-based* thay vì viết một trang web dài ngoằng thì React giúp em chia phần nhỏ giao diện thành các khối nhỏ (Navbar, DashboardCard, ExpenseItem, ChartSection,...) sau đó ghép chúng lại với nhau. Điều này cực kỳ tiện lợi và giống với cách tư duy trong các bài mẫu Flutter khi mình muốn tái sử dụng code [2].



Hình 2. 1 React (ReactJS)

2.1.2 Nguyên lý hoạt động

Cơ chế vận hành của React dựa trên hai khái niệm nền tảng

- **Virtual DOM (Document Object Model ảo):** đây là bản sao nhẹ của DOM thật. Khi trạng thái (state) của một đối tượng thay đổi, React sẽ cập nhật Virtual DOM trước. Sau đó, nó sử dụng thuật toán “Diffing” để so sánh sự khác biệt giữa phiên bản Virtual DOM mới và cũ. Cuối cùng, quá trình “Reconciliation”

sẽ chỉ cập nhật những phần tử thực sự thay đổi lên DOM thật của trình duyệt. Cơ chế này giúp giảm thiểu các thao tác trực tiếp lên DOM thật, vốn là nguyên nhân chính gây chậm trễ trong các ứng dụng web truyền thống [3].

- **Luồng dữ liệu một chiều (One way Data Flow):** Trong React, dữ liệu được truyền từ component cha xuống con thông qua các thuộc tính gọi là props. Component con không thể trực tiếp thay đổi dữ liệu của cha mà phải thông qua các hàm callback. Nguyên lý này giúp luồng dữ liệu trở nên minh bạch, dễ dự đoán và dễ debug (gỡ lỗi) hơn [4].

Đây là Vũ khí bí mật giúp cho React có thể đạt hiệu năng cao. React không có thao tác trực tiếp lên DOM thật mà thực tế vốn rất chậm và tạo ra một **Virtual DOM** trong bộ nhớ.



Hình 2. 2 DOM vs Virtua DOM

Quy trình cập nhật gồm

1. Khi state thay đổi, React sẽ tạo ra một cây Virtual DOM mới
2. React so sánh Virtual DOM cũ và mới khi sử dụng thuật toán **Diffing**.
3. Sau đó React tìm ra những điểm khác biệt theo cấp độ node.
4. Chỉ những phần thay đổi trên React sẽ cập nhật cục bộ lên DOM thật.

Nhờ cơ chế này nên ứng dụng Expense Tracker dù có nhiều biểu đồ và danh sách giao dịch có phức tạp bao nhiêu thì vẫn hoạt động một cách mượt mà không cần load lại toàn bộ trang.

2.1.3 Component-Based và Hooks

Trong dự án của em, React tổ chức ứng dụng thành các **Component** độc lập. Mỗi phần giao diện đều được chia thành các component quản lý và giao diện riêng của nó. Điều này sẽ giúp cho code dễ bảo trì và tái sử dụng.

Đặc biệt khi phiên bản 16.8 của React giới thiệu **Hooks** cho phép sử dụng State và các tính năng của React trong Functional Component mà không cần sử dụng Class.

- **useState:** Quản lý trạng thái dữ liệu.
- **useEffect:** Xử lý các tác phụ như gọi API và lắng nghe.
- **useContext:** Quản lý state toàn cục mà không cần truyền props qua nhiều cấp.

React Hooks sẽ giúp code gọn gàng hơn rất nhiều so với Class Component ngày xưa.

2.1.4 Thiết kế bằng Tailwind CSS

Ngày xưa làm web thì hay dùng Bootstrap, nhưng với dự án này em quyết định dùng **Tailwind CSS**. Khác so với các framework cung cấp sẵn các component (nôm na là các nút bấm đã được style sẵn) thì Tailwind lại cung cấp các lớp tiện ích ở cấp thấp.

Đây là một framwork theo hướng có nghĩ là không cần mở file CSS để thiết kế các class dài dòng. Chỉ cần viết trực tiếp ngay trong file HTML (JSX) mà không cần nhảy qua lại giữa các file CSS với nhau. Điều này giúp cho web dễ dàng tạo ra giao diện **Glassmorphism** (Hiệu ứng kính mờ) cho các thẻ KPI hay Sidbar trôi nổi trong phần demo. Đặc biệt, Tailwind hợp cho giao diện Dashboard là thứ vốn cần sự gọn gàng và ít rườm rà, và cả hỗ trợ **Dark Mode** và **Responsive** (giao diện thích ứng mobile) rất tốt, điều này cực kỳ giúp cho người dùng có thể xem báo cáo tài chính trên điện thoại một cách dễ dàng hơn.



Hình 2. 3 Tailwind

2.1.5 Ưu điểm và nhược điểm

Ưu điểm

- **Hiệu năng cao:** Nhờ cơ chế Virtual DOM, React tối ưu hóa việc render lại giao diện, giúp ứng dụng hoạt động mượt mà ngay cả khi xử lý lượng dữ liệu lớn [3].
- **Tái sử dụng mã nguồn:** Cấu trúc component cho phép lập trình viên viết code một lần và sử dụng lại ở nhiều nơi, giúp tiết kiệm thời gian và giảm thiểu lỗi [2].
- **Hệ sinh thái phong phú:** Cộng đồng hỗ trợ lớn với hàng ngàn thư viện bổ trợ (như Redux, React Router) giúp giải quyết hầu hết các vấn đề phát sinh [1].

Nhược điểm

- **Chỉ là thư viện View:** React không phải là một Framework đầy đủ (Full – stack framework). Để xây dựng một ứng dụng hoàn chỉnh, lập trình viên phải tự tin hiểu và tích hợp thêm các thư viện định tuyến (routing) và quản lý trạng thái (state management), điều này có thể gây khó khăn cho người mới bắt đầu [4].
- **JSX (JavaScript XML):** Cú pháp JSX pha trộn giữa HTML và JavaScript có thể gây bỡ ngỡ ban đầu và vi phạm nguyên tắc “Separation of concerns” (tách biệt mỗi quan tâm) truyền thống [1].

2.1.6 Ứng dụng trong đề tài

Trong khuôn khổ đề tài “Hệ thống quản lý tiêu dùng cá nhân”, ReactJS đóng vai trò là công nghệ cốt lõi để xây dựng giao diện phía Client:

- Xây dựng cấu trúc ứng dụng đơn trang (SPA) giúp trải nghiệm người dùng liền mạch, không bị gián đoạn bởi việc tải lại trang.
- Sử dụng React Hooks (useState, useEffect, useContext) để quản lý trạng thái phiên đăng nhập, dữ liệu thu chi và giao diện Dark/ Light mode một cách hiệu quả.
- Tận dụng tính năng Component để xây dựng các thành phần tái sử dụng như TransactionCard, Modal, Sidebar, giúp mã nguồn gọn gàng và dễ bảo trì.
- Tích hợp với các thư viện trong hệ sinh thái React như Recharts để vẽ biểu đồ thống kê và Framer Motion để tạo hiệu ứng chuyển động mượt mà.

2.2 Về Phía Backend: Python FastAPI

Trong kiến trúc hệ thống Client-Server, nếu Frontend là “lớp vỏ” giao tiếp với người dùng thì Backend chính là “bộ não” xử lý logic nghiệp vụ và dữ liệu. Đối với đề tài này, yêu cầu đặt ra là phải xử lý nhanh các tác vụ tài chính (CRUD), đồng thời tích hợp mượt mà với các mô hình trí tuệ nhân tạo (AI). Do đó, sự kết hợp giữa ngôn ngữ Python và framework FastAPI là lựa chọn tối ưu nhất.

2.2.1 Ngôn ngữ Python

Python là một ngôn ngữ lập trình bậc cao, thông dịch, hướng đối tượng và đa năng, được Guido van Rossum ra mắt lần đầu vào năm 1991. Triết lý thiết kế của Python nhấn mạnh vào khả năng đọc mã (readability) với cú pháp rõ ràng, súc tích [5].

Trong bối cảnh công nghệ hiện nay, Python đã trở thành ngôn ngữ thống trị trong các lĩnh vực Khoa học dữ liệu (Data Science) và trí tuệ nhân tạo (Artificial Intelligence). Việc sử dụng Python cho Backend giúp hệ thống dễ dàng tiếp cận với hệ sinh thái thư viện AI khổng lồ như *LangChain* hay các SDL của Google Gemini mà không cần thông qua các cầu nối phức tạp [6].



Hình 2. 4 Python

2.2.2 Framework FastAPI

Tổng quan: FastAPI là một web framework hiện đại dùng để xây dựng các API bằng Python 3.8+, dựa trên các tiêu chuẩn Python mới nhất (đặc biệt là Type Hints). Được tạo ra bởi Sebastián Ramírez vào năm 2018, FastAPI được thiết kế để tối ưu hóa hiệu năng và tốc độ phát triển của lập trình viên [7].

Khác với các framework truyền thống như Django (theo mô hình “pin đi kèm” – batteries included) hay Flask (tối giản – microframework), FastAPI tập trung vào việc xây dựng API hiệu năng cao, tận dụng tối đa sức mạnh của lập trình bất đồng bộ.

Nguyên lý hoạt động: FastAPI vận hành dựa trên hai thư viện nền tảng mạnh mẽ:

- **Starlette :** Phụ trách phần web (routing, request/ response). Starlette giúp FastAPI hoạt động như một ứng dụng **ASGI** (Asynchronous Server Gateway Interface), cho phép xử lý hàng ngàn kết nối đồng thời một cách bất đồng bộ mà không bị tắc nghẽn (non – blocking I/O) [8].

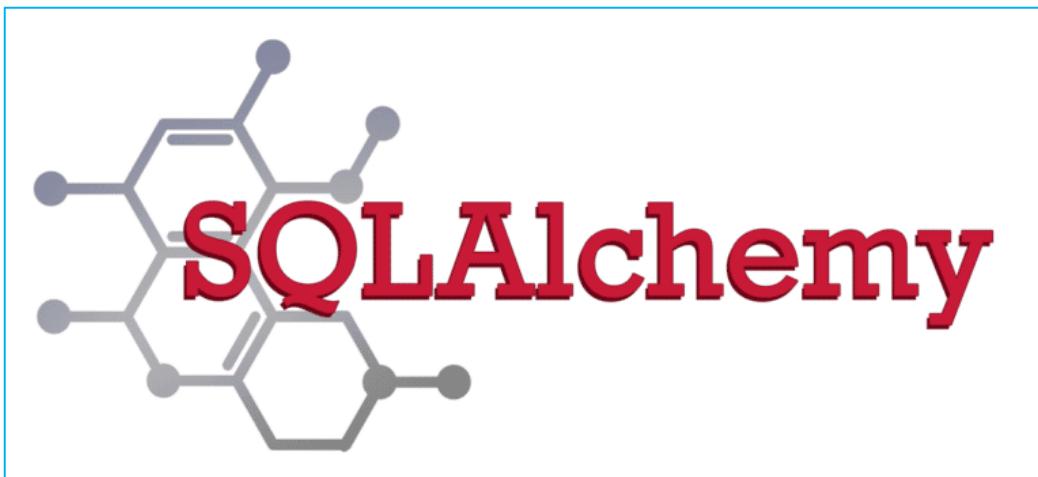
Cơ chế Dependency Injection (Tiêm phụ thuộc): Đây là một tính năng cốt lõi giúp FastAPI trở nên linh hoạt. Nó cho phép lập trình viên khai báo các thành phần cần thiết (như kết nối Database, thông tin User hiện tại) ngay trong tham số của hàm xử lý (Path Operation Function). Framework sẽ tự động giải quyết và cung cấp các thành phần này khi chạy [7].



Hình 2. 5 FastAPI

2.2.3 Quản lý dữ liệu với SQLAlchemy và Pydantic

- **SQLAlchemy (ORM):** Thay viết các câu lệnh SQL thuần dẽ sai sót thì mình dùng SQLAlchemy để mô hình hóa dữ liệu với cơ sở dữ liệu thông qua các đối tượng Python (Object Relational Mapping).



Hình 2. 6 SQLAlchemy

- **Pydantic:** Cái này dùng để gác cổng. Khi mà dữ liệu từ Frontend gửi tới thì Pydantic sẽ kiểm tra dữ liệu xem có đi đúng định dạng không. Nếu có sai thì sẽ bị chặn ngay và giúp cho hệ thống an toàn hơn và giảm rủi ro hỏng dữ liệu.



Hình 2. 7 Pydantic

2.2.4 Ưu điểm và Nhược điểm

Ưu điểm

- **Hiệu năng vượt trội:** Nhờ cơ chế bất đồng bộ (Async/Await) và Starlette, tốc độ của FastAPI ngang ngửa với NodeJS và Go, nhanh hơn nhiều so với các framework Python truyền thống [8].
- **Phát triển nhanh & Ít lỗi:** Việc sử dụng Type Hints và Pydantic giúp IDE hỗ trợ gợi ý code (autocomplete) tốt hơn và bắt lỗi ngay từ lúc viết code [9].
- **Tài liệu tự động:** FastAPI tự động sinh ra trang tài liệu API tương tác (Swagger UI và ReDoc) ngay khi khởi chạy server, giúp việc kiểm thử và tích hợp Frontend - Backend trở nên cực kỳ dễ dàng [7].

Nhược điểm

- **Cộng đồng nhỏ hơn Django/Flask:** Do là framework mới, số lượng tài liệu hướng dẫn và thư viện bên thứ 3 (third-party plugins) chưa phong phú bằng các “đàn anh” lâu đời.
- **Đòi hỏi kiến thức Async:** Lập trình viên cần hiểu rõ về cơ chế bất đồng bộ (async/await) trong Python để tránh các lỗi blocking làm giảm hiệu năng server.

2.2.5 Ứng dụng trong đề tài

Trong hệ thống **Expense Tracker**, FastAPI đóng vai trò là xương sống của Backend với các ứng dụng cụ thể:

- **Xây dựng RESTful API:** Định nghĩa các điểm cuối (Endpoints) để Frontend gọi xuống, ví dụ: GET /incomes, POST /transactions.
- **Kiểm soát dữ liệu (Data Validation):** Sử dụng các Pydantic Models (như TransactionCreate, UserUpdate) để đảm bảo dữ liệu tài chính (số tiền, ngày tháng) luôn đúng định dạng trước khi lưu vào Database.
- **Xử lý Bất đồng bộ (Async Handling):** Các tác vụ nặng như gọi API của Google Gemini (Chatbot) hay gửi request sang Firebase được xử lý bất đồng bộ (async def), đảm bảo server không bị “treo” khi chờ phản hồi từ bên thứ 3.

- **Quản lý Phụ thuộc (Dependency Injection):** Sử dụng Depends để tự động lấy phiên làm việc của Database (get_db) và xác thực người dùng (get_current_user_db) trong mỗi request, giúp code gọn gàng và bảo mật hơn.

2.3 Cơ sở dữ liệu: PostgreSQL

2.3.1 Tổng quan

PostgreSQL (thường được gọi là Postgres) là một hệ quản trị cơ sở dữ liệu quan hệ đối tượng (Object-Relational Database Management System - ORDBMS) mã nguồn mở mạnh mẽ. Nó được phát triển từ dự án POSTGRES tại Đại học California, Berkeley vào năm 1986 [10].

Khác với các hệ quản trị CSDL quan hệ truyền thống chỉ tập trung vào bảng và hàng, bản chất “hướng đối tượng” của PostgreSQL cho phép nó hỗ trợ các tính năng nâng cao như kế thừa bảng (table inheritance) và các kiểu dữ liệu phức tạp (Array, JSONB), giúp thu hẹp khoảng cách giữa lập trình hướng đối tượng và cơ sở dữ liệu quan hệ [11].

2.3.2 Nguyên lý hoạt động

Sức mạnh của PostgreSQL nằm ở kiến trúc tuân thủ nghiêm ngặt các chuẩn công nghiệp và các cơ chế quản lý dữ liệu tiên tiến:

- **Tuân thủ chuẩn ACID:** Đây là nguyên tắc vàng trong thiết kế CSDL tài chính. ACID là viết tắt của Atomicity (Tính nguyên tử), Consistency (Tính nhất quán), Isolation (Tính cô lập) và Durability (Tính bền vững). PostgreSQL đảm bảo rằng một giao dịch (transaction) – ví dụ như chuyển tiền từ ví này sang ví khác – hoặc là thành công hoàn toàn, hoặc là thất bại hoàn toàn, không bao giờ có chuyện tiền bị trừ ở ví A mà chưa cộng vào ví B [12].
- **MVCC (Multi-Version Concurrency Control):** Đây là cơ chế kiểm soát đồng thời đa phiên bản. Thay vì khóa (lock) toàn bộ dòng dữ liệu khi có người đang đọc hoặc ghi – điều thường gây tắc nghẽn trong các hệ thống cũ, PostgreSQL tạo ra các bản chụp (snapshot) dữ liệu cho từng giao dịch. Nhờ đó, “người đọc không chặn người ghi, và người ghi không chặn người đọc”, giúp

hệ thống duy trì hiệu năng cao ngay cả khi có nhiều người dùng thao tác cùng lúc [13].

Thứ cần sự chính xác và toàn vẹn cao nhất là dữ liệu tài chính của người dùng. Cho nên chúng ta không sử dụng NoSQL (như MongoDB) mà chúng ta chọn RDBMS (Hệ quản trị CSDL quan hệ) cụ thể đó là **PostgreSQL** với vai trò quan hệ mạnh và phù hợp cho tài chính vì

- Hỗ trợ ACID đảm bảo tính toàn vẹn dữ liệu
- Có các cơ chế khóa (Foreign, Unique)
- Tối ưu cho các câu truy vấn phức tạp
- Mạnh hơn MySQL trong các tính năng phân tích dữ liệu



Hình 2. 8 PostgreSQL

2.3.3 Ưu điểm và Nhược điểm

Ưu điểm

- **Độ tin cậy cao:** PostgreSQL nổi tiếng với khả năng bảo toàn dữ liệu (Data Integrity) và hoạt động ổn định trong thời gian dài mà không cần bảo trì quá nhiều [10].
- **Hỗ trợ kiểu dữ liệu phong phú:** Ngoài các kiểu chuẩn (Integer, Varchar), nó hỗ trợ cực tốt kiểu JSON/JSONB, cho phép lưu trữ dữ liệu phi cấu trúc ngay trong bảng quan hệ (NoSQL in SQL), giúp linh hoạt trong việc lưu các cấu hình tùy biến của người dùng [11].
- **Khả năng mở rộng (Extensibility):** Cộng đồng phát triển mạnh, cung cấp nhiều extension hữu ích như PostGIS (địa lý) hay pgvector (cho AI).

Nhược điểm

- **Tốc độ đọc:** Trong các tác vụ đọc đơn giản (simple read-heavy), PostgreSQL có thể chậm hơn một chút so với MySQL do kiến trúc phức tạp hơn của nó [12].
- **Tiêu tốn bộ nhớ:** Mỗi kết nối đến PostgreSQL thường tốn nhiều RAM hơn so với các CSDL khác, đòi hỏi cấu hình server phải tương đối tốt nếu lượng người dùng lớn [13].

2.3.4 Ứng dụng trong đề tài

Trong đồ án này, PostgreSQL đóng vai trò là “trái tim” lưu trữ toàn bộ dữ liệu nghiệp vụ:

- **Thiết kế Schema chuẩn hóa:** Dữ liệu được tổ chức thành các bảng có quan hệ chặt chẽ như Users, Incomes, Expenses, Categories. Các ràng buộc khóa ngoại (Foreign Key) được thiết lập để đảm bảo tính nhất quán (Ví dụ: Không thể tạo một khoản chi tiêu cho một danh mục không tồn tại).
- **Lưu trữ lịch sử (Audit Logs):** Tận dụng khả năng xử lý văn bản mạnh mẽ để lưu trữ bảng audit_logs, ghi lại mọi hành động quan trọng của Admin phục vụ cho việc tra soát.

- **Tương tác qua ORM:** Sử dụng thư viện **SQLAlchemy** trong Python để tương tác với PostgreSQL, giúp viết code truy vấn an toàn, tránh lỗi SQL Injection và dễ dàng bảo trì.

2.4 Trí tuệ nhân tạo (AI Intergration)

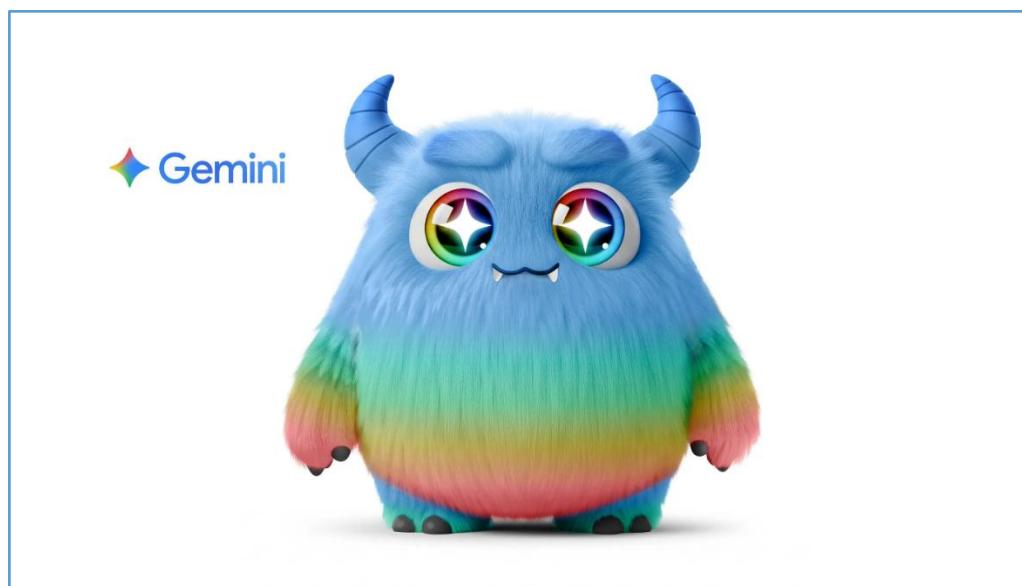
Đây là phần tạo nên sự khác biệt cho đề tài này. Để có thể xây dựng tính năng cho “Trợ lý tài chính ảo” (FinBot), em đã kết hợp với **Google Gemini** và **LangChain**.

2.4.1. Tổng quan về Generative AI và LLM

Generative AI (AI tạo sinh) là một nhánh của trí tuệ nhân tạo tập trung vào việc tạo ra nội dung mới (văn bản, hình ảnh, âm thanh) dựa trên dữ liệu đã được huấn luyện. Cốt lõi của các chatbot hiện đại là Large Language Models (LLM - Mô hình ngôn ngữ lớn).

LLM là các mạng nơ-ron nhân tạo (thường dựa trên kiến trúc Transformer) được huấn luyện trên một lượng dữ liệu văn bản khổng lồ từ internet. Chúng có khả năng hiểu ngữ cảnh, tóm tắt văn bản, dịch thuật và sinh ra văn bản giống như con người [14].

Trong đề tài này, mình sử dụng mô hình Gemini 2.0 Flash của Google. Đây là dòng mô hình đa phương thức (Multimodal), được tối ưu hóa cho tốc độ phản hồi cực nhanh (low latency) và chi phí thấp, rất phù hợp cho các tác vụ trợ lý ảo thời gian thực [15].



Hình 2. 9 Google Gemini

2.4.2. Nguyên lý hoạt động: Kiến trúc Transformer và Token

Khác với các chatbot thế hệ cũ dựa trên quy tắc (Rule-based) cứng nhắc (nếu A thì B), LLM hoạt động dựa trên xác suất thống kê.

- **Tokenization:** Khi người dùng nhập câu “Ăn sáng 50k”, LLM không hiểu đó là chữ cái, mà nó chia câu đó thành các đơn vị nhỏ gọi là “Token” (có thể là từ, hoặc một phần của từ).
- **Cơ chế Attention (Sự chú ý):** Kiến trúc Transformer cho phép mô hình tập trung vào các từ khóa quan trọng trong câu (ví dụ: “50k”, “Ăn sáng”) và bỏ qua các từ đệm không quan trọng, từ đó hiểu được ý định (Intent) của người dùng là muốn ghi chép chi tiêu [16].
- **Next-Token Prediction:** Dựa trên ngữ cảnh đã có, mô hình tính toán xác suất để dự đoán từ tiếp theo hợp lý nhất để hoàn thành câu trả lời.

2.4.3. Framework LangChain và Cơ chế Agent

Việc kết nối trực tiếp code Python với API của Google Gemini chỉ giúp chúng ta có một con bot biết “chat”. Để con bot biết “làm việc” (tương tác với Database), mình cần một framework điều phối, đó là LangChain.

LangChain giải quyết bài toán kết nối LLM với dữ liệu bên ngoài thông qua khái niệm Agent (Tác nhân) và Tools (Công cụ) [17].

- **Tools:** Là các hàm Python được lập trình viên viết sẵn (ví dụ: `create_transaction`, `get_balance`). Mỗi tool được gắn kèm một bản mô tả (Description) bằng tiếng Anh để AI hiểu công dụng của nó.
- **Agent:** Là bộ não trung gian. Khi nhận câu lệnh từ người dùng, Agent sẽ sử dụng LLM để suy luận: “*Với câu hỏi này, mình có cần dùng Tool nào không?*”.
 - Nếu cần, Agent sẽ trích xuất tham số từ câu nói, gọi Tool, lấy kết quả từ Database, và cuối cùng dùng LLM để diễn đạt lại kết quả đó thành câu trả lời tự nhiên cho người dùng.



Hình 2. 10 LangChain Framework

2.4.4. Ứng dụng trong đề tài

Hệ thống FinBot trong đồ án được xây dựng dựa trên sự kết hợp giữa Google Gemini và LangChain:

- Input: Người dùng nhập “Đỗ xăng 100k”.
- Processing: LangChain gửi prompt đến Gemini. Gemini phân tích và nhận diện đây là hành động ghi chép (Type: Expense, Amount: 100000, Category: Transport).
- Action: LangChain kích hoạt tool create_transaction.
- Output: Hàm Python thực thi lệnh SQL INSERT vào bảng expenses trong PostgreSQL và trả về thông báo thành công.

2.5 Xác Thực và Bảo mật với Firebase

Vấn đề bảo mật rất đặc biệt là quản lý mật khẩu cho người dùng luôn luôn là một bài toán đau đầu. Cho nên để tránh rủi ro lộ và lọt các thông tin và giám sát việc phải tự

xây dựng hệ thống Auth phức tạp, chúng ta sử dụng **Firebase Authentication** của Google.

Bảo mật danh tính người dùng là lớp bảo vệ đầu tiên và quan trọng nhất của bất kỳ hệ thống tài chính nào. Thay vì tự xây dựng hệ thống xác thực từ đầu (vốn tiềm ẩn nhiều rủi ro lỗ hổng bảo mật như SQL Injection hay lưu trữ mật khẩu yếu), mình quyết định sử dụng **Firebase Authentication**.

2.5.1. Tổng quan

Firebase Authentication là một dịch vụ Backend-as-a-Service (BaaS) của Google, cung cấp giải pháp xác thực người dùng toàn diện, an toàn và dễ dàng tích hợp. Nó hỗ trợ đa dạng các phương thức đăng nhập như Email/Password, Google, Facebook, GitHub [18].

2.5.2. Nguyên lý hoạt động (Cơ chế Token-based)

Hệ thống sử dụng chuẩn **OAuth 2.0** và **OpenID Connect** để xác thực:

1. **Client-side (ReactJS):** Người dùng nhập thông tin đăng nhập. Firebase SDK trên trình duyệt sẽ gửi thông tin này trực tiếp đến máy chủ Google.
2. **Token Issuance:** Nếu thông tin đúng, Google trả về một **ID Token** (định dạng JWT - JSON Web Token). Token này chứa thông tin định danh của người dùng (UID, Email) và có thời hạn ngắn (thường là 1 giờ).
3. **Server-side (FastAPI):** Khi người dùng gọi API (ví dụ: xem danh sách thu chi), Client gửi kèm ID Token này trong Header (Authorization: Bearer <token>).
4. **Verification:** Backend sử dụng Firebase Admin SDK để giải mã và kiểm tra chữ ký số của Token. Nếu hợp lệ, Backend mới tin tưởng request đó và cho phép truy cập dữ liệu trong PostgreSQL [19].



Hình 2. 11 Firebase

Firebase Auth cung cấp giải pháp toàn diện cho việc quản lý định danh:

- **Token-based Authentication:** Sử dụng JWT (JSON Web Token) để xác thực các phiên bản làm việc. Khi người dùng đăng ký, Firebase cấp một ID Token. Token này được gửi kèm mỗi Request lên FastAPI Server để có thể xác minh danh tính người dùng là ai.
- **Đa nền tảng:** Hỗ trợ đăng nhập bằng Email/ Password, Google, Facebook, ... dễ dàng.
- **Bảo mật cao:** Google sẽ chịu trách nhiệm về mã hóa mật khẩu để có thể bảo vệ và chống lại các cuộc tấn công Brute-force giúp giảm tải gánh nặng bảo mật cho server của chúng ta.

2.5.3. Ưu điểm và Nhược điểm

- **Ưu điểm:** Độ bảo mật cực cao (do Google quản lý); Không cần lo lắng về việc mã hóa hay lưu trữ mật khẩu trong Database riêng; Tích hợp sẵn các tính năng nâng cao như gửi email xác thực, quên mật khẩu.
- **Nhược điểm:** Phụ thuộc vào dịch vụ bên thứ 3; Cần xử lý đồng bộ dữ liệu giữa Firebase (lưu user auth) và PostgreSQL (lưu user profile/data).

2.5.4. Ứng dụng trong đề tài

Trong Expense Tracker, Firebase đóng vai trò “người gác cổng”:

- Quản lý đăng ký, đăng nhập, đăng xuất.
- Cung cấp UID duy nhất để liên kết dữ liệu người dùng trong PostgreSQL.
- Hỗ trợ bảo mật 2 lớp (khi kết hợp với logic 2FA tự xây dựng ở Backend).

Tại sao sử dụng Firebase?

- Bảo mật tốt hơn tự làm tay
- Có sẵn đăng nhập qua Email/ Password
- Tạo và xác thực token chuẩn
- Tích hợp dễ vào React lẫn FastAPI

Cho nên Backend chỉ việc nhận token từ Client và gửi về Firebase kiểm tra. Nếu hợp lệ sẽ cho phép truy cập API.

CHƯƠNG 3: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

3.1 Phân tích và thiết kế hệ thống

3.1.1. Yêu cầu hệ thống (Functional Requirements)

Hệ thống được thiết kế để xử lý bài toán quản lý tài chính cá nhân một cách toàn diện, với việc ghi chép thủ công cho đến việc nhờ AI hỗ trợ. Cụ thể là hệ thống sẽ phải đáp ứng được các nhóm chức năng chính như sau:

- **Tài khoản và xác thực:** Hệ thống phải cho phép người dùng đăng ký, đăng nhập an toàn. Ở đây mình ưu tiên sử dụng Email/ Password và hỗ trợ các cơ chế bảo mật như đổi mật khẩu, quên mật khẩu. Đặc biệt là phải có tính năng xác thực 2 yếu tố (2FA) để bảo vệ dữ liệu tài chính nhạy cảm.
- **Giao dịch:** Người dùng có thể thêm, sửa, xóa được các khoản thu nhập và chi tiêu. Mỗi giao dịch cần có đầy đủ thông tin như số tiền, ngày tháng, danh mục, ghi chú chi tiết.
- **Danh mục:** Không ai tiêu tiền giống ai cả cho nên hệ thống cần có bộ danh mục linh hoạt. Ngoài các danh mục mặc định, người dùng thoải mái tự tạo được danh mục riêng với màu sắc và icon tùy ý thích.
- **Chatbot AI:** Đây là tính năng nâng cao. Hệ thống phải có một con chatbot(FinBot) hiểu được ngôn ngữ tự nhiên tiếng việt. Người dùng nhắn “ăn sáng 50k” là nó phải tự hiểu để lưu vào database. Nó cũng phải biết về biểu đồ và báo cáo số dư khi được hỏi.
- **Phân tích và báo cáo:** Số liệu nhập vào phải được trực quan hóa. Hệ thống cần có Dashboard tổng quan, các biểu đồ xu hướng (Trend) và biểu đồ cơ cấu (Breakdown) để người dùng nhìn vào là biết mình đang tiêu quá trớn ở đâu.
- **Xuất dữ liệu:** Xuất Excel theo khoảng thời gian hoặc toàn bộ, lựa chọn loại xuất.
- **Quản trị hệ thống:** Cần một khu vực riêng cho Admin để giám sát lượng người dùng, quản lý các danh mục hệ thống và xem nhật ký hoạt động (Audit Logs) để xử lý sự cố.

3.1.2. Yêu cầu phi chức năng

Ngoài việc làm được việc thì hệ thống còn phải đảm bảo chạy tốt. Dưới đây là tiêu chuẩn cho phi chức năng em đặt ra:

- **Hiệu năng (Performance):**

- Hệ thống phải phản hồi nhanh cho thao tác CRUD giao dịch gần như phải tức thì.
- Riêng với chatbotAI, độ trễ phải ở mức chấp nhận được để không làm gãy mạch hội thoại của người dùng.
- Việc tải các biểu đồ thống kê không được làm đơ trình duyệt, kể cả khi dữ liệu lớn.

- **Bảo mật (Security):**

- Mật khẩu người dùng không được lưu trực tiếp mà phải thông qua Provider uy tín (Firebase).
- Các API nhạy cảm phải được bảo vệ bằng Token (JWT).
- Dữ liệu của người dùng này tuyệt đối không được để lộ sang người dùng khác (Data Isolation)

- **Tính khả dụng và giao diện (Usability):**

- Giao diện phải thân thiện, dễ nhìn
- Hệ thống phải Responsive, tức là hiển thị tốt trên cả máy tính và điện thoại di động, vì người dùng thường ghi chép chi tiêu ngày trên điện thoại.

- **Khả năng mở rộng (Scalability):**

- Cơ sở dữ liệu PostgreSQL được thiết kế chuẩn hóa để có thể chứa lượng lớn giao dịch mà không bị vỡ cấu trúc.
- Backend FastAPI viết theo mô hình module(Router – Controller – Service) để dễ dàng thêm tính năng mới sau này.

3.1.3 Xác định các tác nhân của hệ thống.

Để hệ thống hoạt động trơn tru và bảo mật, việc đầu tiên mình cần làm là xác định rõ: “Ai là người sử dụng?” và “Họ được phép làm những gì?”. Dựa trên mô hình phân quyền đã xây dựng ở Backend, mình xác định hệ thống có 3 nhóm tác nhân (Actors) chính tương tác trực tiếp.

1. Khách (Guest) Đây là nhóm đối tượng chưa được định danh trong hệ thống. Họ chỉ mới tiếp cận ở “cửa ngõ” và chưa có quyền truy cập vào dữ liệu tài chính.

- **Hoạt động chính:** Truy cập trang chủ (Landing page), thực hiện Đăng ký tài khoản mới hoặc Đăng nhập nếu đã có tài khoản. Nếu quên mật khẩu, họ cũng thực hiện quy trình khôi phục tại đây.

2. Người dùng (User) Đây là nhân vật chính, đối tượng phục vụ cốt lõi của ứng dụng **Expense Tracker**. Họ là những cá nhân có nhu cầu ghi chép và kiểm soát tài chính của riêng mình.

- **Quyền hạn:** Họ có toàn quyền “sinh sát” đối với dữ liệu cá nhân của họ (Thu nhập, Chi tiêu, Danh mục).
- **Hoạt động chính:**
 - Nhập liệu các khoản thu/chi hàng ngày.
 - Tương tác với **Chatbot AI (FinBot)** để nhập liệu nhanh hoặc hỏi số dư.
 - Xem các báo cáo, biểu đồ phân tích (Analytics) để điều chỉnh hành vi tiêu dùng.
 - Tùy biến hệ thống (tạo danh mục màu sắc riêng, cài đặt giao diện Sáng/Tối).
 - Xuất dữ liệu ra file Excel để lưu trữ cá nhân .

3. Quản trị viên (Admin) Nhóm này đóng vai trò là người vận hành và giám sát sự ổn định của hệ thống. Họ không can thiệp vào ví tiền của người dùng (vì lý do riêng tư), nhưng họ nắm quyền kiểm soát hạ tầng dữ liệu.

- **Quyền hạn:** Truy cập vào trang **Admin Dashboard** riêng biệt.
- **Hoạt động chính:**
 - Giám sát các chỉ số vĩ mô (Tổng user, Dòng tiền toàn hệ thống).

- Quản lý vòng đời của tài khoản người dùng (Xem, Cấp quyền, Xóa/Ban tài khoản vi phạm).
- Thiết lập các cấu hình nền tảng như: Danh mục mặc định (Default Categories) hay gửi thông báo toàn hệ thống (Broadcast) .

3.1.4. Xác định chức năng cho từng nhóm đối tượng

Hệ thống phân chia rõ ràng cho hai nhóm đối tượng sử dụng: **Người dùng (User)** và **Quản trị viên (Admin)**. Dưới đây là bảng phân rã chức năng chi tiết cho từng nhóm.

▪ Chức năng dành cho Người dùng (User)

Đây là danh sách các tính năng mà một người dùng bình thường sẽ sử dụng hàng ngày để quản lý ví tiền của họ.

Bảng 3. 1 Bảng phân rã chức năng cho Người Dùng (User)

STT	Nhóm chức năng	Chức năng chính	Mô tả chi tiết
1	Xác thực và Tài khoản	Đăng nhập / Đăng xuất	Truy cập vào hệ thống bảo mật thông qua Email/Password hoặc Google.
2		Đăng ký tài khoản	Tạo tài khoản mới, hệ thống tự động khởi tạo dữ liệu mẫu.
3		Quên mật khẩu	Quy trình khôi phục mật khẩu qua email xác thực.
4		Quản lý Hồ sơ (Profile)	Cập nhật thông tin cá nhân (Avatar, Tên, Ngày sinh, Tiền tệ mặc định).
5		Bảo mật và Session	Đổi mật khẩu, Bật/Tắt xác thực 2 lớp (2FA), Quản lý phiên đăng nhập (Xem IP, thiết bị).
6	Dashboard (Tổng quan)	Xem KPI Tài chính	Hiển thị nhanh: Tổng thu, Tổng chi, Số dư hiện tại ngay khi vào app.

7		Biểu đồ xu hướng	Xem nhanh biểu đồ thu chi trong 30 ngày gần nhất.
8		Nhật ký hoạt động	Xem danh sách các giao dịch vừa mới thực hiện.
9	Quản lý Giao dịch	Thêm Thu nhập / Chi tiêu	Nhập liệu giao dịch mới với các trường: Số tiền, Ngày, Danh mục, Ghi chú.
10		Chỉnh sửa / Xóa	Sửa lại thông tin nếu nhập sai hoặc xóa bỏ giao dịch cũ.
11		Lọc và Tìm kiếm	Tìm lại giao dịch cũ theo khoảng thời gian hoặc loại danh mục.
12	Quản lý Danh mục	Tùy biến Danh mục	Tạo mới các danh mục thu/chi riêng (Ví dụ: “Quỹ đen”, “Tiền thưởng”).
13		Cá nhân hóa	Chọn màu sắc (Color Picker) và biểu tượng (Emoji) cho danh mục để dễ nhìn hơn.
14	Trợ lý ảo AI (FinBot)	Chat nhập liệu	Nhập liệu bằng ngôn ngữ tự nhiên (VD: “Ăn sáng 30k” -> Tự động lưu).
15		Hỏi đáp tài chính	Hỏi AI về số dư, thống kê chi tiêu mà không cần vào trang báo cáo.
16		Vẽ biểu đồ tức thì	Yêu cầu AI phân tích và vẽ biểu đồ ngay trong khung chat.
17	Báo cáo và Tiện ích	Phân tích (Analytics)	Xem biểu đồ tròn (Cơ cấu chi tiêu) và biểu đồ cột (So sánh Thu/Chi) chuyên sâu.
18		Xuất dữ liệu (Export)	Tải toàn bộ lịch sử giao dịch về máy dưới dạng file Excel (.xlsx) để lưu trữ.

- **Chức năng dành cho Quản trị viên (Admin)**

Admin là người vận hành, có quyền “sinh sát” trong hệ thống nhưng không can thiệp vào dữ liệu tài chính cá nhân của người dùng.

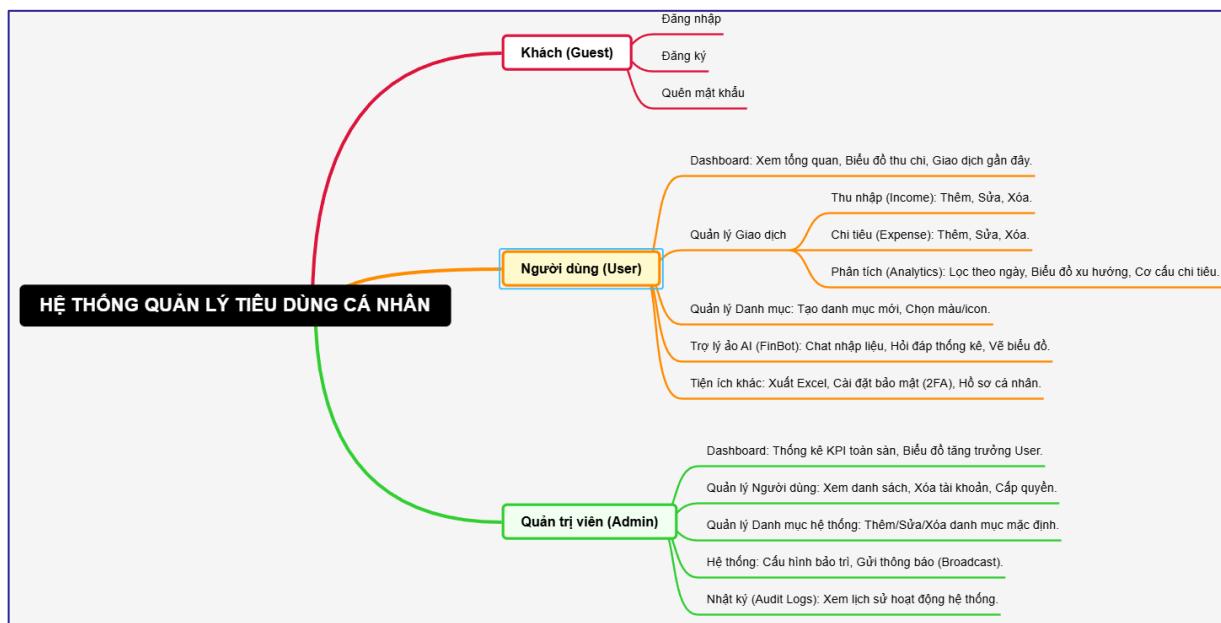
Bảng 3. 2 Bảng Phân rã chức năng cho Quản trị viên (Admin)

STT	Nhóm chức năng	Chức năng chính	Mô tả chi tiết
1	Dashboard Quản trị	Thông kê toàn sàn	Xem tổng số User, tổng dòng tiền luân chuyển trong hệ thống.
2		Biểu đồ tăng trưởng	Theo dõi số lượng người đăng ký mới theo thời gian thực (User Growth).
3	Quản lý Người dùng	Danh sách người dùng	Xem danh sách toàn bộ tài khoản, trạng thái hoạt động (Active/Banned).
4		Xóa / Khóa tài khoản	Xóa vĩnh viễn các tài khoản vi phạm hoặc spam khỏi Database và Firebase.
5		Phân quyền	Cấp quyền Admin cho người dùng khác hoặc thu hồi quyền.
6	Cấu hình Hệ thống	Danh mục Mặc định	Thêm, Sửa, Xóa các danh mục hệ thống (System Categories) dùng chung cho mọi user.
7		Thông báo (Broadcast)	Gửi tin nhắn thông báo bảo trì hoặc cập nhật tính năng đến màn hình Dashboard của tất cả user.
8		Cài đặt vận hành	Bật/Tắt chế độ bảo trì (Maintenance Mode), Cho phép/Chặn đăng ký mới.
9	Giám sát và Log	Nhật ký hệ thống (Audit Logs)	Xem lịch sử các hành động nhạy cảm (như ai vừa xóa user nào, vào lúc nào) để truy vết lỗi.

3.2 Sơ đồ chức năng Mind Map

3.2.1 Sơ đồ chức năng (Functional Diagram)

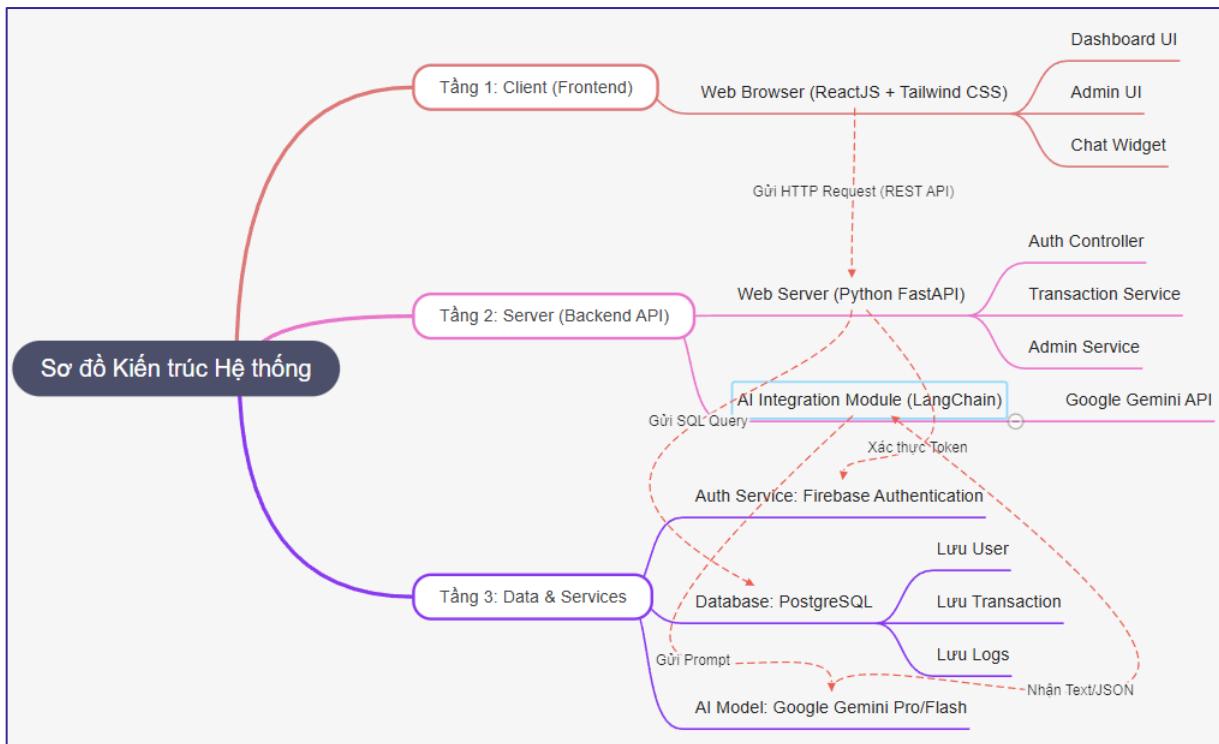
Hệ thống Website Quản lý tiêu dùng cá nhân được phân chia thành 3 nhánh đối tượng chính: Khách (Guest), Người dùng (User) và Quản trị viên (Admin). Mỗi đối tượng sẽ có quyền truy cập vào những vùng chức năng riêng biệt, đảm bảo tính bảo mật và sự tập trung trong trải nghiệm.



Hình 3. 1 Sơ đồ chức năng

3.2.2 Sơ đồ kiến trúc hệ thống (Architecture Diagram)

Để hiện thực hóa các chức năng trên, mình không thể nhồi nhét tất cả vào một chỗ. Hệ thống được thiết kế theo mô hình Client-Server đa tầng, có sự tách biệt rõ ràng giữa giao diện và xử lý dữ liệu.



Hình 3. 2 Sơ đồ kiến trúc hệ thống

3.3 Sơ đồ Use – Case

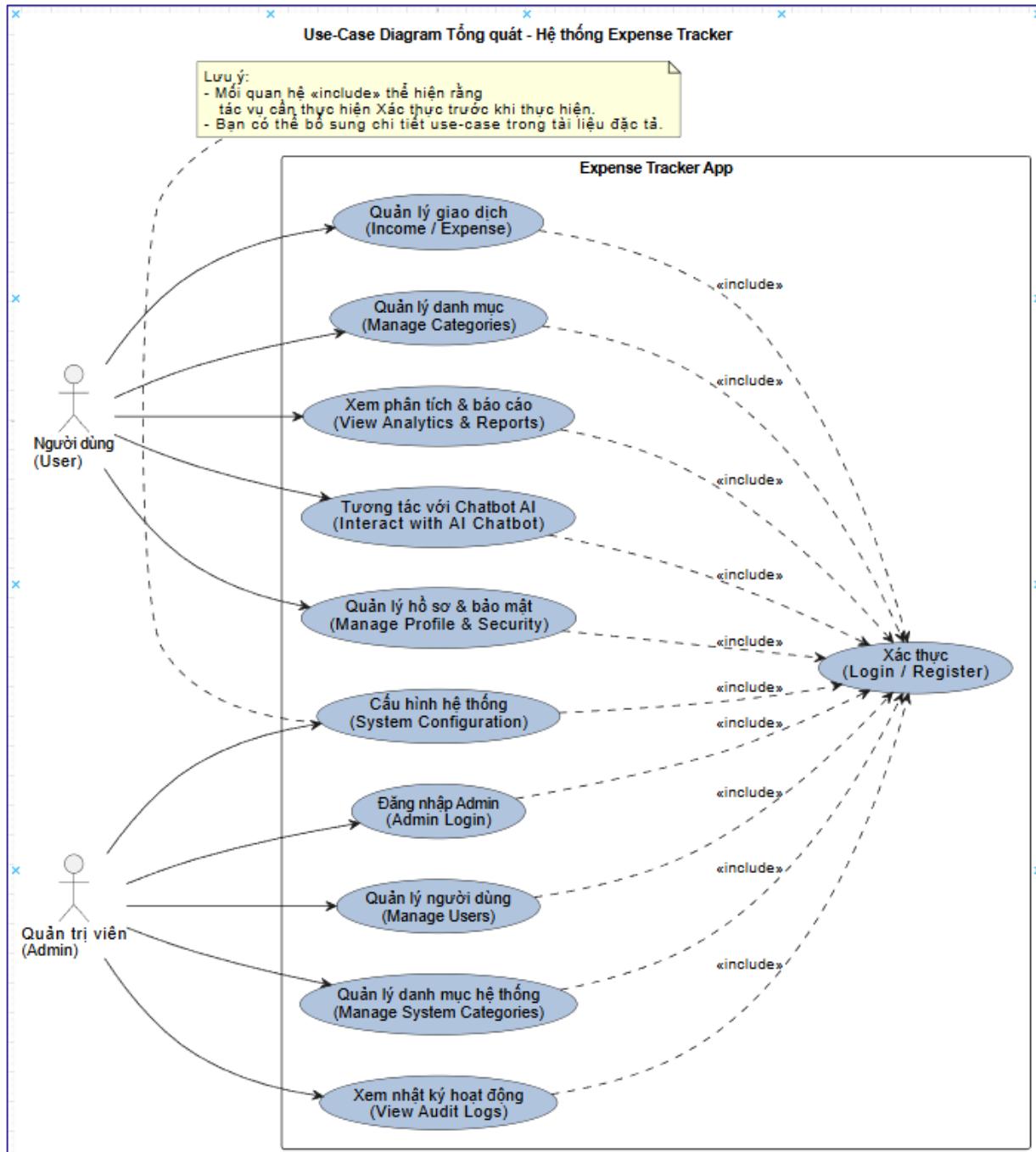
3.3.1 Xác định các tác nhân (Actors)

Dựa trên khảo sát và thiết kế phân quyền trong mã nguồn (file user_model.py và auth_token_db.py), hệ thống của mình có các tác nhân chính sau:

1. **Khách (Guest):** Là người dùng vãng lai, chưa có tài khoản hoặc chưa đăng nhập. Phạm vi quyền hạn của họ rất hạn chế, chỉ dừng lại ở việc xem trang giới thiệu hoặc thực hiện đăng ký/đăng nhập.
2. **Người dùng (User):** Đây là tác nhân chính của hệ thống. Họ là những người đã có tài khoản và đăng nhập thành công. Họ có toàn quyền thao tác với dữ liệu tài chính cá nhân của mình như thêm thu chi, xem báo cáo, và chat với trợ lý ảo.
3. **Quản trị viên (Admin):** Là người chịu trách nhiệm vận hành hệ thống. Admin có quyền truy cập vào trang Dashboard riêng để giám sát số liệu, quản lý người dùng và can thiệp vào các danh mục mặc định của hệ thống.
4. **Hệ thống AI (System Actor):** Trong đề tài này, con Chatbot (FinBot) đóng vai trò như một tác nhân hệ thống đặc biệt. Nó không phải con người, nhưng nó tham gia trực tiếp vào quy trình xử lý giao dịch (tự động tạo giao dịch từ tin nhắn).

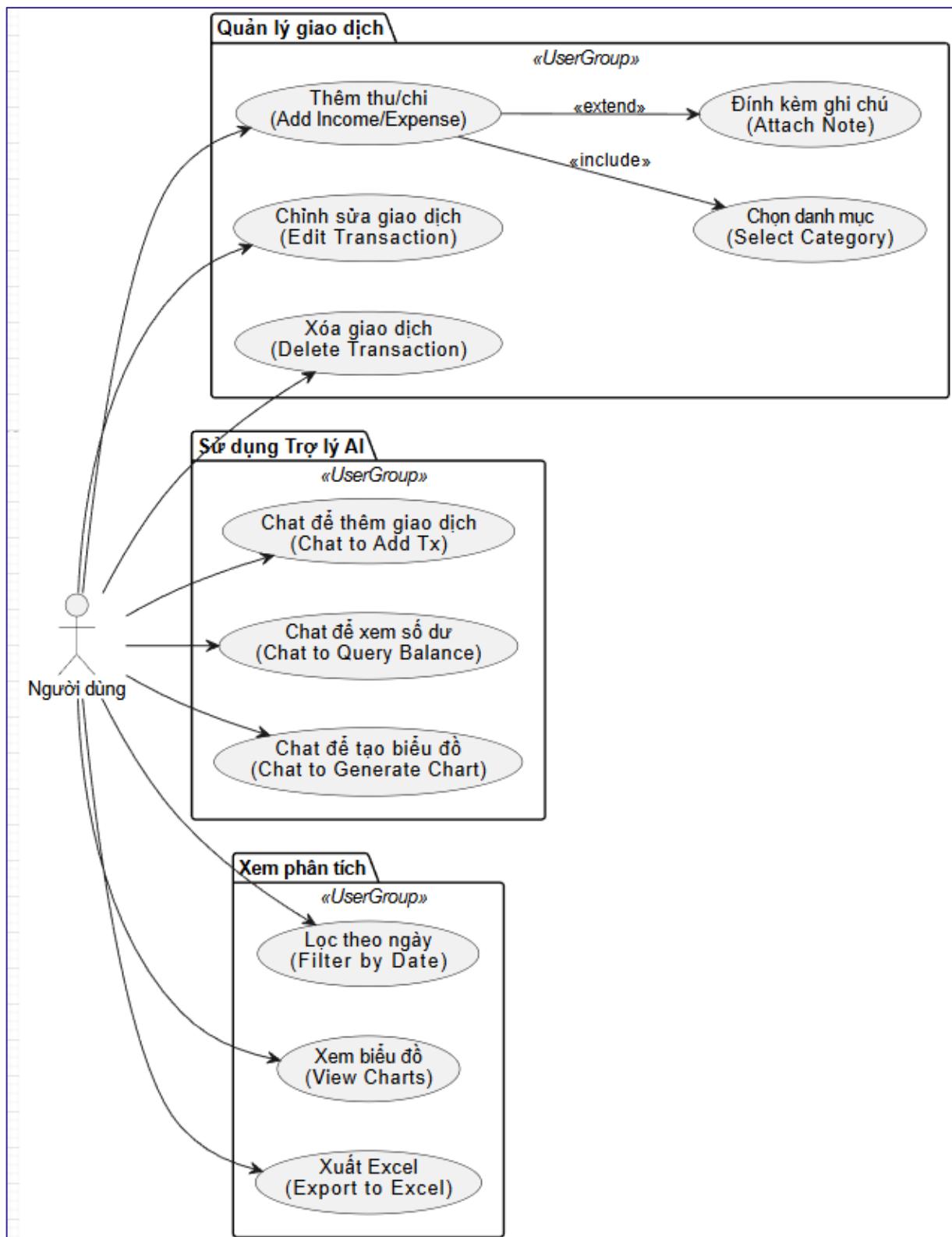
3.3.2 Sơ đồ Use – Case Tổng quát

Sơ đồ này sẽ cho chúng ta cái nhìn bao quát nhất về toàn bộ hệ thống, xem các tác nhân kết nối với những nhóm chức năng lớn nào.



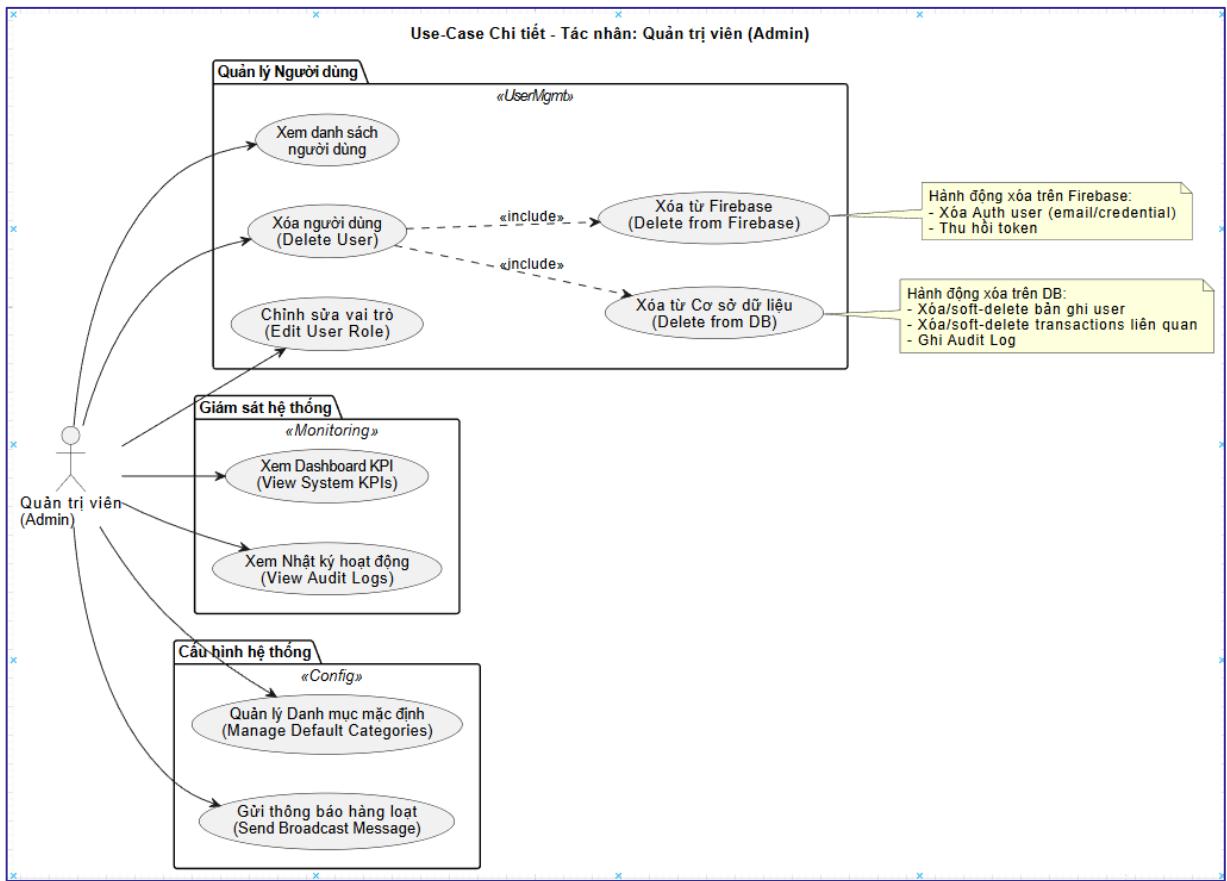
Hình 3. 3 Sơ đồ Use-Case Tổng quát

3.3.3 Phân rã Use – Case cho người dùng (User)



Hình 3. 4 Sơ đồ Use – Case người dùng

3.3.4 Phân rã Use – Case cho Quản trị viên (Admin)



Hình 3. 5 Phân rã Use – Case Quản trị viên

3.4 Đặc tả chi tiết Use – Case

3.4.1 Nhóm chức năng Xác thực và Tài khoản

Đây là “cánh cửa” đầu tiên của hệ thống. Do mình sử dụng Firebase Authentication kết hợp với Backend riêng, nên luồng xử lý sẽ có chút đặc biệt so với các web thông thường.

Bảng 3. 3 Đặc tả Use – Case Đăng nhập (Login)

Tiêu đề	Nội dung
Tên Use-Case	Đăng nhập hệ thống

Mô tả	Cho phép Người dùng hoặc Quản trị viên truy cập vào hệ thống bằng tài khoản đã đăng ký. Hệ thống sẽ xác thực qua Firebase và đồng bộ dữ liệu từ Backend.
Sự kiện kích hoạt	Người dùng truy cập vào trang chủ và nhấn nút “Login”.
Tác nhân (Actor)	User, Admin.
Tiền đề kiện	<ul style="list-style-type: none"> - Người dùng đã có tài khoản hợp lệ. - Kết nối mạng ổn định.
Hậu điều kiện	<ul style="list-style-type: none"> - Người dùng được chuyển hướng vào trang Dashboard tương ứng với quyền hạn (User hoặc Admin Dashboard). - Token đăng nhập (JWT) được lưu vào trình duyệt.
Luồng sự kiện chính	<ol style="list-style-type: none"> 1. Người dùng nhập Email và Mật khẩu tại form Đăng nhập. 2. Người dùng nhấn nút “Login”. 3. Hệ thống gửi thông tin lên Firebase Authentication để xác thực. 4. Firebase trả về idToken nếu thông tin đúng. 5. Frontend gửi idToken này xuống Backend (API /auth/sync) để lấy thông tin chi tiết từ PostgreSQL. 6. Backend kiểm tra quyền (User hay Admin) và trả về Profile đầy đủ. 7. Hệ thống lưu thông tin vào LocalStorage và chuyển hướng người dùng vào trang chủ.
Luồng rẽ nhánh	<p>A1: Sai thông tin đăng nhập</p> <ul style="list-style-type: none"> - Tại bước 4, nếu Firebase báo lỗi, hệ thống hiển thị thông báo “Invalid Email or Password” màu đỏ và yêu cầu nhập lại. <p>A2: Tài khoản bị khóa</p> <ul style="list-style-type: none"> - Tại bước 6, nếu Backend phát hiện user bị Admin khóa (is_active = false), hệ thống từ chối truy cập và thông báo lý do.

3.4.2 Nhóm chức năng Quản lý tài chính (Cốt lõi)

Đây là “trái tim” của ứng dụng, nơi người dùng dành nhiều thời gian nhất để ghi chép.

Bảng 3. 4 Đặc tả Use – Case Thêm giao dịch

Tiêu đề	Nội dung
Tên Use-Case	Thêm mới Thu nhập / Chi tiêu
Mô tả	Người dùng ghi chép một khoản tiền vào sổ cái. Hệ thống hỗ trợ nhập số tiền, chọn ngày, chọn danh mục và thêm ghi chú chi tiết.
Sự kiện kích hoạt	Người dùng nhấn nút “Add New” trên thanh Sidebar hoặc nút “+” nhanh trên Dashboard.
Tác nhân (Actor)	User.
Tiền điều kiện	Người dùng đã đăng nhập.
Hậu điều kiện	- Giao dịch mới được lưu vào CSDL PostgreSQL. - Số dư tổng (Balance) và các biểu đồ được cập nhật ngay lập tức.
Luồng sự kiện chính	<ol style="list-style-type: none"> 1. Hệ thống hiển thị Modal (cửa sổ bật lên) nhập liệu. 2. Người dùng chọn loại giao dịch (Income/Expense). 3. Người dùng nhập Số tiền (Amount) và chọn Ngày giao dịch (Date). 4. Người dùng chọn Danh mục (Category) từ danh sách sổ xuống (Dropdown). <i>Lưu ý: Danh sách này bao gồm cả danh mục mặc định và danh mục riêng của user.</i> 5. Người dùng nhập Ghi chú (Note) để diễn giải thêm (VD: “Đi ăn với bạn”). 6. Người dùng nhấn “Save”. 7. Hệ thống kiểm tra dữ liệu hợp lệ (Validate). 8. Hệ thống gọi API lưu vào Database và đóng Modal.

	9. Giao diện tự động tải lại danh sách để hiển thị giao dịch vừa thêm.
Luồng rẽ nhánh	<p>A1: Bỏ trống thông tin bắt buộc</p> <ul style="list-style-type: none"> - Nếu người dùng không nhập số tiền hoặc ngày, hệ thống sẽ hiện cảnh báo lỗi ngay tại ô nhập liệu và không cho lưu. <p>A2: Nhập sai định dạng tiền</p> <ul style="list-style-type: none"> - Nếu nhập chữ cái vào ô số tiền, hệ thống sẽ tự động chặn hoặc báo lỗi.

3.4.3 Đặt tả Use – Case xem báo cáo phân tích

Bảng 3. 5 Đặc tả Use-Case xem báo cáo phân tích

Tiêu đề	Nội dung
Tên Use-Case	Xem Báo cáo Phân tích Tài chính
Mô tả	Người dùng xem các biểu đồ trực quan để hiểu rõ thói quen chi tiêu của mình trong một khoảng thời gian cụ thể.
Sự kiện kích hoạt	Người dùng chọn menu “Analytics” trên thanh Sidebar.
Tác nhân (Actor)	User.
Tiền điều kiện	Người dùng đã có dữ liệu giao dịch trong hệ thống.
Hậu điều kiện	Hiển thị đầy đủ biểu đồ cột, biểu đồ tròn và danh sách chi tiết.
Luồng sự kiện chính	<ol style="list-style-type: none"> 1. Người dùng vào trang Analytics. 2. Hệ thống mặc định tải dữ liệu của toàn bộ thời gian. 3. Hệ thống hiển thị:

	<ul style="list-style-type: none"> - Biểu đồ cột (Bar Chart): So sánh tổng Thu vs tổng Chi. - Biểu đồ tròn (Donut Chart): Tỷ lệ phần trăm các khoản chi theo danh mục (VD: Ăn uống 40%, Di chuyển 10%). <p>4. Người dùng sử dụng bộ lọc (Filter) để chọn khoảng thời gian (VD: Tháng này, Tuần trước).</p> <p>5. Hệ thống tính toán lại và cập nhật biểu đồ theo thời gian đã chọn.</p>
Luồng rẽ nhánh	<p>A1: Không có dữ liệu</p> <ul style="list-style-type: none"> - Nếu trong khoảng thời gian chọn không có giao dịch nào, biểu đồ sẽ hiển thị trạng thái rỗng (Empty State) với thông báo “No Data Available” thay vì bị lỗi trắng trang.

3.4.4 Nhóm chức năng Thông Minh (AI Chatbot)

Đây là tính năng nổi bật nhất của đồ án, giúp phân biệt sản phẩm của mình với các ứng dụng truyền thống.

Bảng 3. 6 Đặc tả Use – Case Tương tác với trợ lý Áo Finbot

Tiêu đề	Nội dung
Tên Use-Case	Nhập liệu và Tra cứu qua Chatbot AI
Mô tả	Người dùng giao tiếp bằng ngôn ngữ tự nhiên (Tiếng Việt) để yêu cầu hệ thống thực hiện các tác vụ như ghi chép hoặc báo cáo mà không cần thao tác trên giao diện.
Sự kiện kích hoạt	Người dùng mở Widget Chat ở góc màn hình.
Tác nhân (Actor)	User, AI System (Google Gemini).
Tiền điều kiện	<ul style="list-style-type: none"> - API Key của Gemini hoạt động tốt. - Kết nối mạng ổn định.

Hậu điều kiện	<ul style="list-style-type: none"> - Yêu cầu của người dùng được thực hiện (Lưu DB hoặc Trả lời thông tin). - Giao diện Dashboard tự động cập nhật nếu có thay đổi dữ liệu.
Luồng sự kiện chính	<ol style="list-style-type: none"> 1. Người dùng nhập câu lệnh: “Sáng nay đổ xăng 50k”. 2. Hệ thống gửi câu lệnh kèm ngữ cảnh (ngày giờ, danh sách category hiện có) lên Server. 3. Server chuyển tiếp cho mô hình AI (LangChain Agent). 4. AI phân tích và nhận diện ý định: Tạo Giao dịch. 5. AI trích xuất thông tin: Type=Expense, Amount=50000, Category=Transport. 6. AI gọi hàm nội bộ (Tool) để lưu vào CSDL. 7. Hệ thống phản hồi lại người dùng: “Đã thêm 50k vào mục Di chuyển <input checked="" type="checkbox"/>” và gửi tín hiệu [REFRESH] để làm mới giao diện.
Luồng rẽ nhánh	<p>A1: AI không hiểu ý định</p> <ul style="list-style-type: none"> - Nếu câu lệnh quá mơ hồ (VD: “Tiền nong thế nào”), AI sẽ hỏi lại để làm rõ: “Bạn muốn xem báo cáo hay ghi chép khoản nào?”. <p>A2: Yêu cầu vẽ biểu đồ</p> <ul style="list-style-type: none"> - Nếu người dùng bảo “Vẽ biểu đồ tháng này”, AI sẽ trả về dữ liệu dạng JSON đặc biệt để Frontend tự động render thành biểu đồ ngay trong khung chat.

3.4.5 Nhóm chức năng Quản trị hệ thống (Admin)

Khác với người dùng thường, các thao tác của Admin mang tính chất quản lý và giám sát. Do đó, các luồng xử lý ở đây thường đi kèm với việc ghi nhật ký (logging) để đảm bảo tính minh bạch.

Bảng 3. 7 Đặc tả Use – Case Quản lý người dùng (Admin)

Tiêu đề	Nội dung
Tên Use-Case	Quản lý và Xóa tài khoản Người dùng
Mô tả	Admin xem danh sách toàn bộ người dùng trong hệ thống, tìm kiếm tài khoản cụ thể và thực hiện xóa tài khoản nếu phát hiện vi phạm hoặc spam.
Sự kiện kích hoạt	Admin chọn menu “User Management” trên thanh Sidebar.
Tác nhân (Actor)	Admin.
Tiền điều kiện	Tài khoản Admin đã đăng nhập và có quyền hạn is_admin = true.
Hậu điều kiện	<ul style="list-style-type: none"> - Tài khoản người dùng bị xóa khỏi cả Database và Firebase Authentication. - Một bản ghi được tạo trong bảng audit_logs.
Luồng sự kiện chính	<ol style="list-style-type: none"> 1. Admin truy cập trang quản lý User. 2. Hệ thống tải danh sách user (phân trang) hiển thị Avatar, Email, Ngày tham gia. 3. Admin sử dụng thanh tìm kiếm để lọc user theo email. 4. Admin nhấn nút “Delete” (biểu tượng thùng rác) trên dòng user cần xóa. 5. Hệ thống hiển thị Popup cảnh báo đỏ: “Hành động này không thể hoàn tác”. 6. Admin xác nhận xóa. 7. Backend thực hiện xóa song song: Xóa trong PostgreSQL và gọi Firebase Admin SDK để xóa Auth.

	<p>8. Backend ghi lại hành động vào bảng Audit Log (Ai xóa, Xóa ai, Thời gian).</p> <p>9. Hệ thống thông báo thành công và cập nhật lại danh sách.</p>
Luồng rẽ nhánh	<p>A1: Lỗi kết nối Firebase</p> <p>- Nếu không xóa được trên Firebase (lỗi mạng), hệ thống vẫn tiến hành xóa dữ liệu trong Database để đảm bảo sạch rác, đồng thời ghi log cảnh báo lỗi.</p> <p>A2: Admin tự xóa chính mình</p> <p>- Hệ thống chặn hành động này để tránh việc Admin tự tay “khóa cửa” chính mình.</p>

3.4.6 Đặc tả Use-Case Quản Lý Danh Mục Mặc Định

Bảng 3. 8 Đặc tả Use-Case Quản lý danh mục hệ thống (Admin)

Tiêu đề	Nội dung
Tên Use-Case	Quản lý Danh mục Hệ thống
Mô tả	Admin tạo ra bộ danh mục chuẩn (VD: Lương, Ăn uống, Di chuyển) để người dùng mới đăng ký có sẵn cái để dùng ngay mà không cần tạo từ đầu.
Sự kiện kích hoạt	Admin chọn menu “Default Categories”.
Tác nhân (Actor)	Admin.
Hậu điều kiện	Danh mục mới được tạo với user_id = NULL và hiển thị cho tất cả người dùng.
Luồng sự kiện chính	<ol style="list-style-type: none"> 1. Admin vào trang quản lý danh mục. 2. Admin nhấn “Add New”.

	<p>3. Admin nhập tên danh mục, chọn loại (Thu/Chi), chọn màu sắc và biểu tượng (Icon).</p> <p>4. Admin nhấn “Save”.</p> <p>5. Hệ thống lưu vào bảng categories với cờ is_default (hoặc user_id là null).</p> <p>6. Hệ thống ghi log “CREATE_CATEGORY” vào bảng nhật ký.</p>
--	---

3.4.7 Đặc tả Use-case Xem Nhật Ký Hoạt động (View Audit Logs)

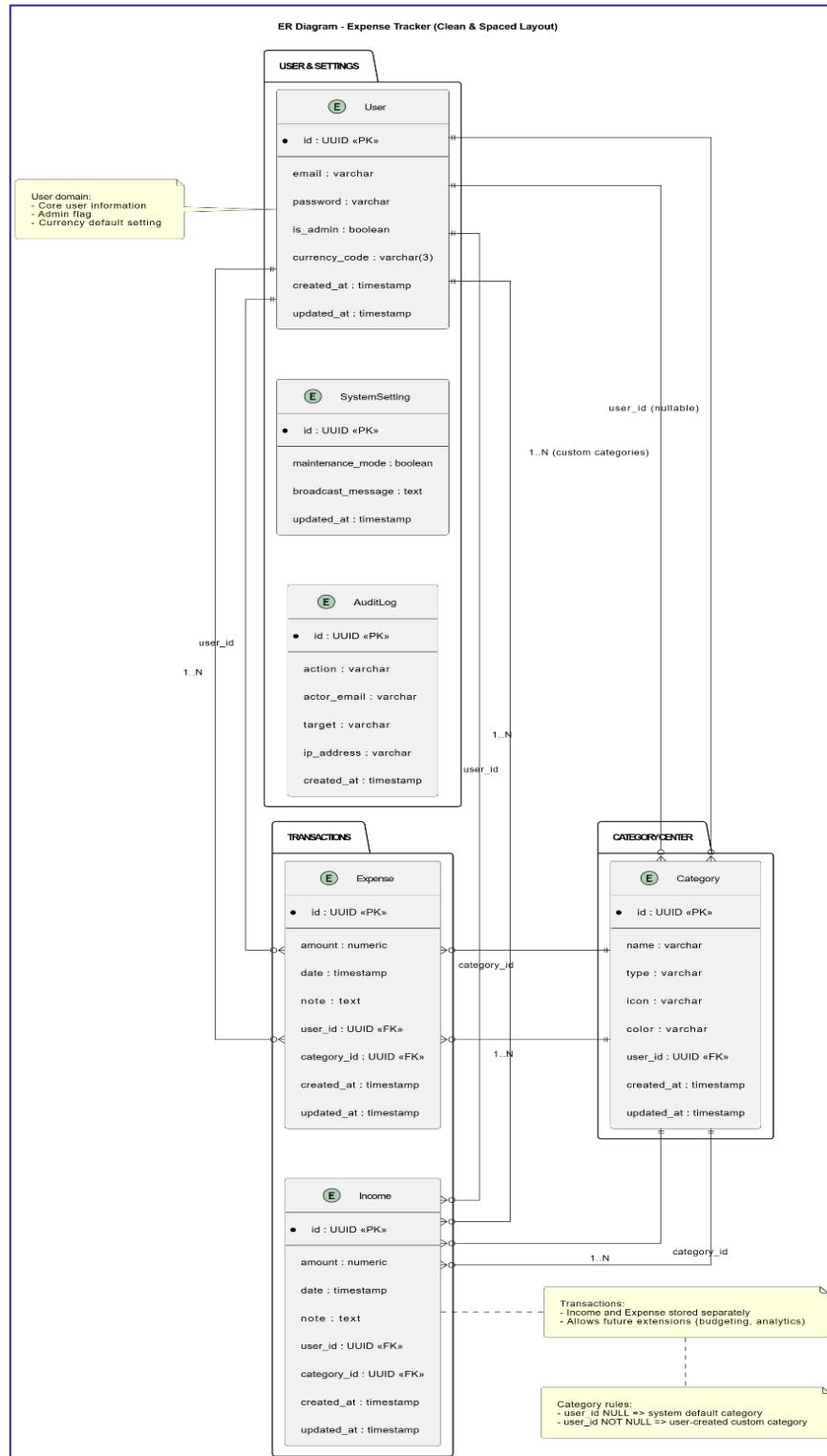
Đây là chức năng giúp hệ thống đạt tiêu chuẩn bảo mật cao hơn.

Bảng 3. 9 Đặc tả Use-Case xem nhật ký hoạt động (Audit Logs)

Tiêu đề	Nội dung
Tên Use-Case	Tra cứu Nhật ký Hệ thống
Mô tả	Admin xem lại lịch sử các tác vụ quan trọng đã diễn ra để truy vết sự cố hoặc kiểm tra bảo mật.
Sự kiện kích hoạt	Admin chọn menu “Audit Logs”.
Luồng sự kiện chính	<p>1. Hệ thống hiển thị bảng Log dạng lưới (Grid) giống Terminal.</p> <p>2. Admin xem các thông tin: Thời gian, Hành động (VD: DELETE_USER), Người thực hiện (Admin Email), Đối tượng (Target Email), Địa chỉ IP.</p> <p>3. Admin lọc log theo từ khóa hoặc trạng thái (Success/Error).</p> <p>4. Admin nhấn “Export” để tải toàn bộ log về file Excel phục vụ báo cáo.</p>

3.5 Thiết kế cơ sở dữ liệu

3.5.1 Lược đồ CSDL quan hệ - ER Diagram



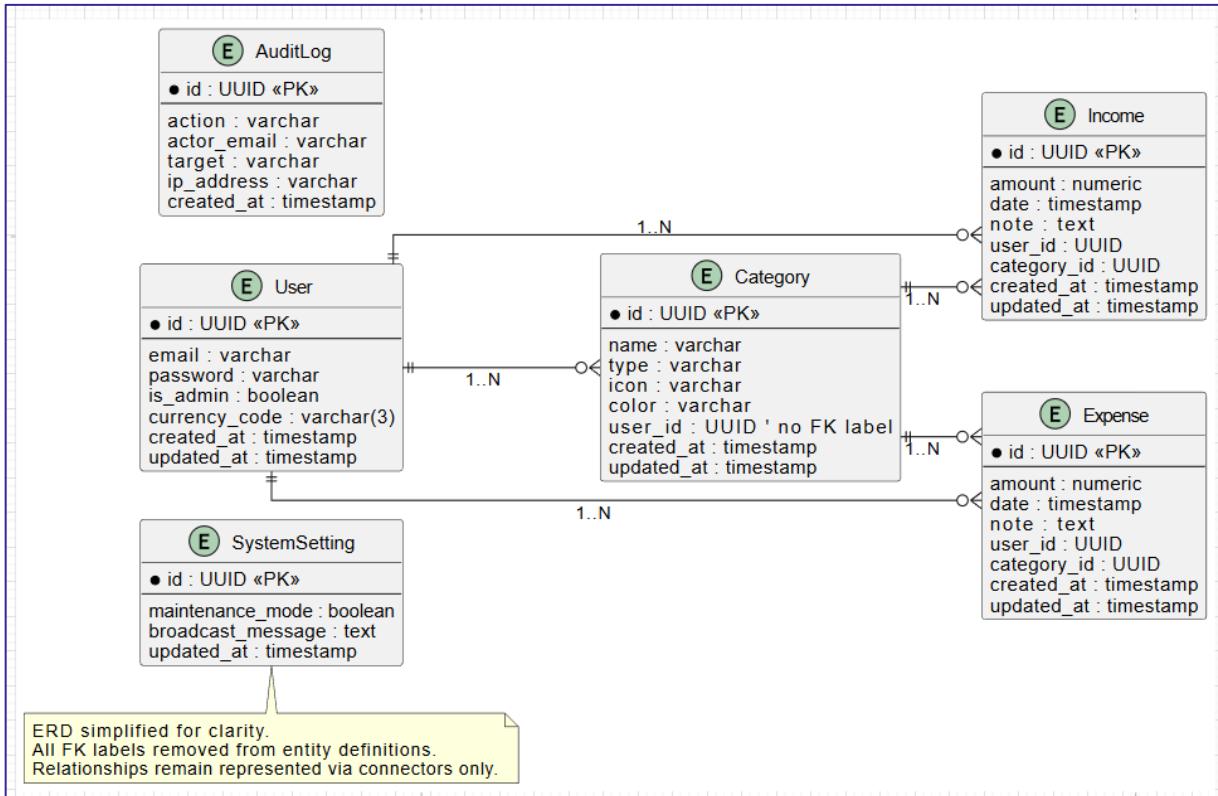
Hình 3. 6 Lược đồ CSDL quan hệ - ER Diagram

3.5.2 Mô tả liệt kê các bảng

Bảng 3. 10 Danh sách các bảng trong CSDL

STT	Tên bảng (Table Name)	Mục đích sử dụng
1	users	Lưu trữ toàn bộ thông tin người dùng trong hệ thống, bao gồm cả User thường và Admin. Bảng này chứa thông tin định danh (email, password), thông tin cá nhân (profile), các cài đặt bảo mật (2FA) và cấu hình tiền tệ.
2	categories	Quản lý danh mục thu chi. Điểm đặc biệt là bảng này chứa cả các Danh mục mặc định (do hệ thống tạo, dùng chung cho mọi người) và Danh mục cá nhân (do người dùng tự tạo).
3	incomes	Lưu trữ chi tiết các giao dịch Thu nhập của người dùng. Mỗi dòng dữ liệu đại diện cho một khoản thu (Lương, Thưởng...) gắn với một thời gian và danh mục cụ thể.
4	expenses	Lưu trữ chi tiết các giao dịch Chi tiêu . Đây là bảng có tốc độ tăng trưởng dữ liệu nhanh nhất, lưu lại mọi khoản chi (Ăn uống, Mua sắm...) để phục vụ cho việc báo cáo và vẽ biểu đồ.
5	audit_logs	Bảng nhật ký hệ thống dành cho Admin. Nó ghi lại lịch sử các hành động quan trọng (như Xóa User, Cập nhật Cấu hình) kèm theo thông tin người thực hiện và thời gian, giúp truy vết sự cố và đảm bảo an toàn.
6	system_settings	Bảng cấu hình toàn cục. Bảng này chỉ chứa duy nhất một dòng dữ liệu (Single Row) để lưu các trạng thái vận hành của hệ thống như: Chế độ bảo trì (Maintenance Mode), Cho phép đăng ký mới, Thông báo khẩn cấp (Broadcast).

3.5.3 Mô hình hóa CSDL ở mức khái niệm



Hình 3. 7 Sơ đồ tổng quát ERD ở mức khái niệm

❖ Giải thích chi tiết các quan hệ giữa các thực thể:

Hệ thống được xây dựng xoay quanh người dùng, vì vậy hầu hết các bảng đều có mối liên kết chặt chẽ với bảng Users. Dưới đây là phân tích chi tiết từng cặp quan hệ:

1. Quan hệ giữa Người dùng (Users) và Giao dịch (Incomes/Expenses)

- **Loại quan hệ:** Một - Nhiều (1 - N).
- **Diễn giải:** Một người dùng có thể tạo ra vô số khoản thu nhập hoặc chi tiêu trong suốt quá trình sử dụng ứng dụng. Ngược lại, mỗi giao dịch cụ thể (như “Ăn sáng 30k”) bắt buộc phải thuộc về duy nhất một người dùng.
- **Cài đặt kỹ thuật:** Trong bảng incomes và expenses, mình sử dụng trường user_id làm Khóa ngoại (Foreign Key) tham chiếu tới id của bảng users. Đặc biệt, mình thiết lập ràng buộc ON DELETE CASCADE. Điều này có nghĩa là nếu Admin xóa một tài khoản User, toàn bộ dữ liệu tài chính của người đó cũng sẽ tự động biến mất theo, giúp dọn dẹp sạch sẽ cơ sở dữ liệu.

2. Quan hệ giữa Người dùng (Users) và Danh mục (Categories)

- **Loại quan hệ:** Một - Nhiều (1 - N) có điều kiện.
- **Diễn giải:** Một người dùng có thể tự tạo ra nhiều danh mục cá nhân (Custom Categories).
- **Điểm đặc biệt:** Trường user_id trong bảng categories được thiết kế cho phép giá trị NULL.
 - Nếu user_id có dữ liệu: Đó là danh mục riêng của người dùng đó.
 - Nếu user_id là NULL: Đó là **Danh mục mặc định (System Default)** do hệ thống cung cấp, hiển thị chung cho tất cả mọi người.
- **Ý nghĩa:** Thiết kế này giúp hệ thống linh hoạt: vừa cung cấp nền tảng sẵn có cho người mới (dùng Default), vừa cho phép cá nhân hóa sâu (dùng Custom) mà không cần tạo ra hai bảng danh mục riêng biệt.

3. Quan hệ giữa Danh mục (Categories) và Giao dịch (Incomes/Expenses)

- **Loại quan hệ:** Một - Nhiều (1 - N).
- **Diễn giải:** Một danh mục (ví dụ: “Ăn uống”) có thể được gắn cho hàng ngàn giao dịch chi tiêu khác nhau. Tuy nhiên, mỗi giao dịch chỉ được phép thuộc về một danh mục duy nhất tại một thời điểm.
- **Cài đặt kỹ thuật:** Bảng incomes và expenses chứa khóa ngoại category_id. Mỗi quan hệ này là cốt lõi để phục vụ cho chức năng **Báo cáo và Thống kê (Analytics)**. Nhờ sự liên kết này, hệ thống mới có thể tính toán được: “Tháng này bạn đã tiêu bao nhiêu tiền cho việc Ăn uống?” bằng cách gom nhóm (Group By) theo category_id.

4. Các bảng độc lập (AuditLogs, SystemSettings)

- **Đặc điểm:** Không có khóa ngoại (Foreign Key) liên kết chặt chẽ với Users.
- **Lý do thiết kế:**
 - Với AuditLogs: Bảng này lưu lịch sử hành động của Admin. Mình quyết định không dùng khóa ngoại liên kết với User để đảm bảo tính toàn vẹn lịch sử. Ngay cả khi một User bị xóa khỏi hệ thống (mất ID trong bảng Users), dòng

- log ghi lại việc “Xóa User đó” vẫn phải tồn tại để đối soát sau này. Do đó, mình lưu actor_email và target dưới dạng văn bản thuần (String) thay vì ID.
- Với SystemSettings: Đây là bảng cấu hình toàn cục, chỉ chứa duy nhất một dòng dữ liệu để bật tắt các tính năng hệ thống (như Bảo trì), nên không cần quan hệ với bất kỳ thực thể nào khác.

3.5.4 Mô hình hóa cơ sở dữ liệu mức vật lý (Physical Data Models)

Dưới đây là từ điển dữ liệu (Data Dictionary) mô tả chi tiết từng bảng.

1. **Bảng Người dùng (users)** Đây là bảng trung tâm, lưu trữ thông tin định danh và các cài đặt cá nhân. Điểm đặc biệt của bảng này là sự tích hợp với Firebase (qua firebase_uid) và các trường cờ (flag) để phân quyền quản trị.

Bảng 3. 11 Thiết kế chi tiết bảng Người Dùng (Users)

STT	Tên trường (Field)	Kiểu dữ liệu (Type)	Ràng buộc (Constraint)	Mô tả
1	id	UUID	PK, Not Null	Khóa chính, định danh duy nhất cho user.
2	firebase_uid	Varchar(255)	Unique, Index	ID ánh xạ với hệ thống xác thực Firebase.
3	email	Varchar(255)	Unique, Not Null	Email đăng nhập.
4	name	Varchar(255)	Nullable	Tên hiển thị của người dùng.

5	profile_image	Text	Nullable	Đường dẫn hoặc chuỗi Base64 ảnh đại diện.
6	currency_code	Varchar(5)	Default 'USD'	Mã tiền tệ mặc định (VD: USD, VND).
7	currency_symbol	Varchar(5)	Default '\$'	Ký hiệu tiền tệ (VD: \$, ₫).
8	is_admin	Boolean	Default False	Có phân quyền: True là Admin, False là User thường.
9	is_2fa_enabled	Boolean	Default False	Trạng thái bật/tắt bảo mật 2 lớp.
10	otp_secret	Varchar	Nullable	Mã bí mật dùng để tạo mã TOTP cho 2FA.
11	created_at	Timestamp	Default Now()	Thời điểm tạo tài khoản.

2. **Bảng Danh mục (categories)** Bảng này quản lý việc phân loại thu chi. Thiết kế của nó cho phép linh hoạt giữa danh mục hệ thống và danh mục cá nhân thông qua trường user_id.

Bảng 3. 12 Thiết kế chi tiết bảng Danh Mục (categories)

STT	Tên trường	Kiểu dữ liệu	Ràng buộc	Mô tả
1	id	UUID	PK, Not Null	Khóa chính.

2	user_id	UUID	FK, Nullable	Tham chiếu đến users.id. Nếu NULL: Danh mục mặc định (Admin tạo). Nếu có giá trị: Danh mục riêng.
3	name	Varchar(100)	Not Null	Tên danh mục (VD: Ăn uống, Lương).
4	type	Varchar(10)	Not Null	Loại danh mục: 'income' hoặc 'expense'.
5	icon	Varchar(50)	Nullable	Lưu mã Emoji hoặc tên Icon hiển thị.
6	color	Varchar(10)	Nullable	Mã màu Hex để hiển thị trên biểu đồ.
7	created_at	Timestamp	Default Now()	Thời điểm tạo.

3. **Bảng Thu nhập (incomes)** Lưu trữ chi tiết các khoản tiền vào. Bảng này tách biệt với expenses để tối ưu hóa việc truy vấn và mở rộng các trường đặc thù sau này (như nguồn thu, thuế...).

Bảng 3. 13 Thiết kế chi tiết bảng Thu Nhập (Incomes)

STT	Tên trường	Kiểu dữ liệu	Ràng buộc	Mô tả
1	id	UUID	PK, Not Null	Khóa chính.
2	user_id	UUID	FK, Not Null	Tham chiếu đến users.id (người tạo).
3	category_id	UUID	FK, Not Null	Tham chiếu đến categories.id.
4	amount	Numeric(14, 2)	Not Null	Số tiền thu nhập (độ chính xác cao).

5	category_name	Varchar(255)	Nullable	Lưu tên danh mục tại thời điểm tạo (dùng làm cache hiển thị).
6	date	Date	Not Null	Ngày ghi nhận thu nhập.
7	note	Text	Nullable	Ghi chú chi tiết cho khoản thu.
8	emoji	Varchar(64)	Nullable	Icon tùy chỉnh cho giao dịch này.

4. **Bảng Chi tiêu (expenses)** Tương tự bảng Thu nhập nhưng dành cho các khoản chi. Đây thường là bảng có lượng dữ liệu lớn nhất trong hệ thống.

Bảng 3. 14 Thiết kế chi tiết bảng Chi Tiêu (expenses)

STT	Tên trường	Kiểu dữ liệu	Ràng buộc	Mô tả
1	id	UUID	PK, Not Null	Khóa chính.
2	user_id	UUID	FK, Not Null	Tham chiếu đến users.id.
3	category_id	UUID	FK, Not Null	Tham chiếu đến categories.id.
4	amount	Numeric(14, 2)	Not Null	Số tiền chi tiêu.
5	category_name	Varchar(255)	Nullable	Tên danh mục cache.
6	date	Date	Not Null	Ngày chi tiêu.
7	note	Text	Nullable	Ghi chú chi tiết (VD: “Ăn tối với bạn”).
8	emoji	Varchar(64)	Nullable	Icon tùy chỉnh.

5. Bảng Giao dịch Tổng hợp (transactions) Bảng này đóng vai trò như một “nhật ký chung” hoặc bộ nhớ đệm (cache) để hỗ trợ việc truy vấn lịch sử nhanh mà không cần phải UNION hai bảng incomes và expenses liên tục.

Bảng 3. 15 Thiết kế chi tiết bảng Giao Dịch Tổng hợp (transactions)

STT	Tên trường	Kiểu dữ liệu	Ràng buộc	Mô tả
1	id	UUID	PK, Not Null	Khóa chính.
2	user_id	UUID	FK, Not Null	Tham chiếu đến users.id.
3	type	Varchar(10)	Not Null	Phân loại: 'income' hay 'expense'.
4	amount	Numeric(14, 2)	Not Null	Số tiền.
5	category_id	UUID	FK, Not Null	Tham chiếu danh mục.
6	note	Text	Nullable	Ghi chú.
7	transaction_date	Date	Not Null	Ngày giao dịch thực tế.

6. Bảng Nhật ký Hệ thống (audit_logs) Đây là thành phần quan trọng của phân hệ Admin, dùng để lưu vết các hành động quản trị nhằm đảm bảo tính minh bạch và an toàn.

Bảng 3. 16 Thiết kế chi tiết bảng Nhật Ký Hệ Thống (Audit Logs)

STT	Tên trường	Kiểu dữ liệu	Ràng buộc	Mô tả
1	id	UUID	PK, Not Null	Khóa chính.
2	action	Varchar(50)	Not Null	Tên hành động (VD: DELETE_USER, UPDATE_SETTING).
3	actor_email	Varchar(255)	Not Null	Email của Admin thực hiện hành động.
4	target	Varchar(255)	Nullable	Đối tượng bị tác động (VD: Email user bị xóa).
5	status	Varchar(20)	Default 'SUCCESS'	Trạng thái thực hiện (Thành công/Thất bại).
6	details	Text	Nullable	Chi tiết lỗi hoặc thông tin bổ sung.
7	ip_address	Varchar(50)	Nullable	Địa chỉ IP của người thực hiện.
8	created_at	Timestamp	Default Now()	Thời gian ghi log.

7. Bảng Cấu hình Hệ thống (system_settings) Bảng này được thiết kế theo dạng *Singleton* (chỉ có 1 dòng duy nhất) để lưu các tham số cấu hình toàn cục của ứng dụng.

Bảng 3. 17 Thiết kế chi tiết bảng Cấu Hình Hệ Thống (System_settings)

STT	Tên trường	Kiểu dữ liệu	Ràng buộc	Mô tả
1	id	Integer	PK, Default 1	Luôn có giá trị là 1.
2	maintenance_mode	Boolean	Default False	Bật/Tắt chế độ bảo trì hệ thống.
3	allow_signup	Boolean	Default True	Cho phép hoặc chặn đăng ký người dùng mới.
4	broadcast_message	Text	Nullable	Nội dung thông báo toàn cục hiển thị cho mọi user.
5	updated_at	Timestamp	Nullable	Thời điểm cập nhật cấu hình gần nhất.

CHƯƠNG 4: THỰC NGHIỆM VÀ TRIỂN KHAI HỆ THỐNG

4.1 Môi trường cài đặt và công nghệ sử dụng

Để đảm bảo hệ thống vận hành ổn định, mượt mà và có khả năng mở rộng, mình đã lựa chọn xây dựng dựa trên các công nghệ mã nguồn mở hiện đại nhất hiện nay. Hệ thống được chia thành 3 khối chính: Frontend, Backend và Database, kết hợp với các dịch vụ bên thứ 3 (Third-party services).

4.1.1. Frontend (Giao diện người dùng)

Phía Client được xây dựng theo mô hình Single Page Application (SPA) để tối ưu trải nghiệm người dùng.

- **ReactJS (Vite):** Thư viện cốt lõi để xây dựng giao diện. Mình chọn Vite thay vì Create-react-app truyền thống để có tốc độ khởi động và build nhanh hơn hẳn.
- **Tailwind CSS:** Framework CSS chủ đạo. Thay vì viết file CSS rác, mình dùng Tailwind để style trực tiếp (Utility-first), giúp giao diện đồng bộ và dễ dàng tùy biến Dark/Light mode.
- **Recharts:** Thư viện vẽ biểu đồ mạnh mẽ, được dùng để hiển thị các biểu đồ xu hướng thu chi và biểu đồ tròn (Pie chart) trên Dashboard.
- **Framer Motion:** Thư viện xử lý chuyển động, giúp tạo ra các hiệu ứng mượt mà khi chuyển trang hoặc khi các bảng thông báo (Toast) xuất hiện.
- **Lucide React:** Bộ icon nhẹ, hiện đại, giúp giao diện trông sạch sẽ và chuyên nghiệp hơn.

4.1.2. Backend (Máy chủ ứng dụng)

Phía Server được viết bằng Python, tận dụng sức mạnh xử lý dữ liệu và khả năng tích hợp AI.

- **Python và FastAPI:** Đây là trái tim của hệ thống. FastAPI cho hiệu năng cực cao (nhờ Asynchronous), tự động sinh tài liệu API (Swagger UI) và rất dễ bắt lỗi dữ liệu đầu vào.

- **SQLAlchemy (ORM):** Giúp mình thao tác với cơ sở dữ liệu bằng các đối tượng Python (Class) thay vì phải viết câu lệnh SQL trần, giảm thiểu rủi ro SQL Injection.
- **Pydantic:** Dùng để định nghĩa Schema, đảm bảo dữ liệu gửi lên và trả về luôn đúng định dạng (Validate dữ liệu chặt chẽ).
- **LangChain:** Framework quan trọng nhất để kết nối logic của ứng dụng với trí tuệ nhân tạo, giúp tạo ra các “Agent” thông minh cho Chatbot.

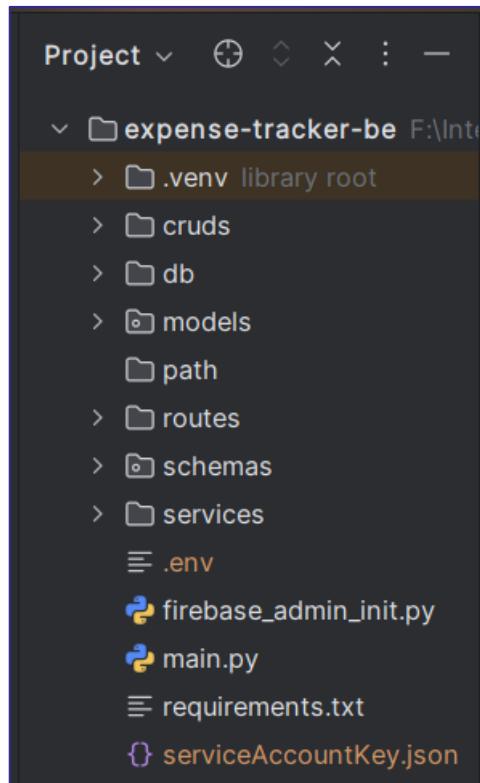
4.1.3. Database và Services (Lưu trữ và Tích hợp)

- **PostgreSQL:** Hệ quản trị cơ sở dữ liệu quan hệ, dùng để lưu trữ người dùng, giao dịch, danh mục và logs.
- **Firebase Authentication:** Dịch vụ của Google giúp xử lý việc đăng ký, đăng nhập, quản lý Token và bảo mật mật khẩu người dùng.
- **Google Gemini API (Model gemini-2.0-flash):** Mô hình ngôn ngữ lớn (LLM) được tích hợp để xử lý ngôn ngữ tự nhiên, giúp Chatbot hiểu và thực hiện lệnh của người dùng với tốc độ phản hồi cực nhanh.

4.2 Tô chức mã nguồn

Mã nguồn dự án được tổ chức tách biệt rõ ràng giữa Frontend và Backend để dễ dàng quản lý và bảo trì.

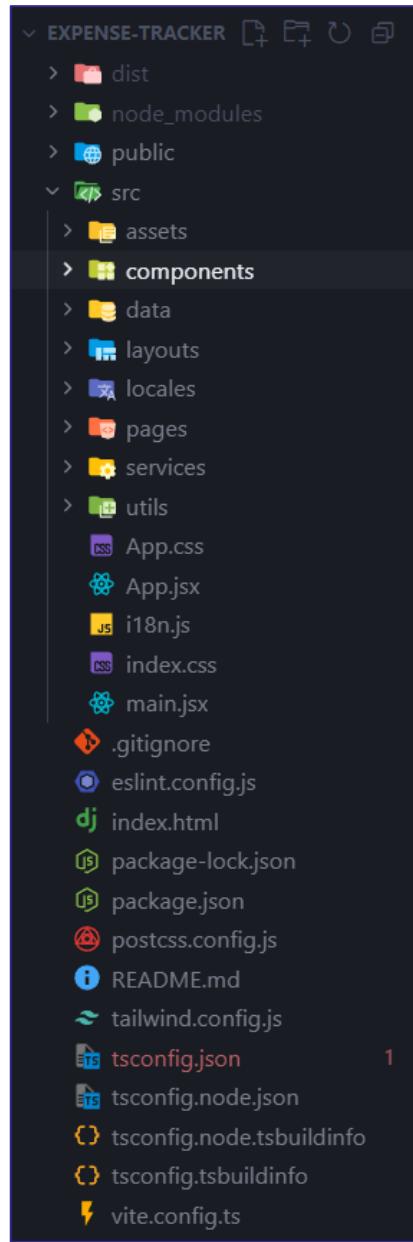
- **Phía Backend** (expense-tracker-be)



Hình 4. 1 Sơ đồ thư mục Backend

- /cruds: Chứa các hàm tương tác trực tiếp với Database (Create, Read, Update, Delete).
- /models: Định nghĩa cấu trúc bảng (Tables) trong PostgreSQL.
- /routes: Định nghĩa các API Endpoints (đường dẫn) để Frontend gọi vào.
- /schemas: Định nghĩa khuôn mẫu dữ liệu (Input/Output) cho API.
- /services: Chứa logic nghiệp vụ phức tạp như Chatbot AI, Auth Token.

- Phía Frontend (expense-tracker)



Hình 4. 2 Sơ đồ cây thư mục Frontend

- /src/components: Các thành phần giao diện nhỏ (Button, Sidebar, Modal).
- /src/pages: Các trang màn hình chính (Home, Income, Admin...).
- /src/services: Các hàm gọi API kết nối xuống Backend.
- /src/layouts: Cấu trúc khung sườn chung (DashboardLayout, AuthLayout).

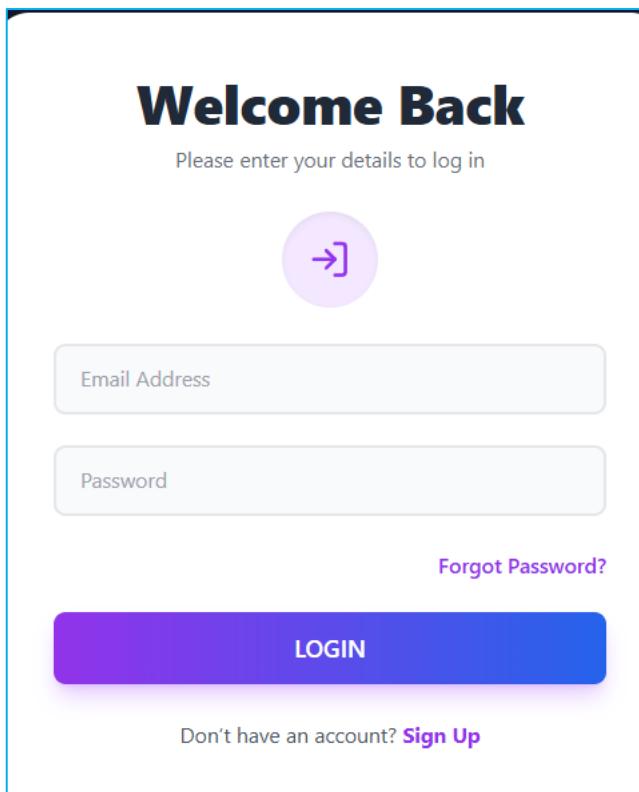
4.3 Thiết kế giao diện chương trình

Sau khi đã hoàn thiện phần xử lý logic phía sau (Backend), việc tiếp theo là xây dựng “bộ mặt” cho ứng dụng. Giao diện người dùng (UI) không chỉ cần đẹp mà quan trọng hơn là phải dễ hiểu, dẫn dắt người dùng đi đúng luồng nghiệp vụ mà mình đã thiết kế.

Dưới đây là các màn hình thực nghiệm của hệ thống, kèm theo mô tả chi tiết các thành phần chức năng.

4.3.1 Giao diện Đăng nhập (Login)

Đây là màn hình đầu tiên người dùng nhìn thấy khi truy cập vào hệ thống (nếu chưa có phiên làm việc). Thay vì làm một form đăng nhập đơn điệu giữa màn hình trắng, mình đã sử dụng bố cục chia đôi (Split Layout): một bên là Form nhập liệu, một bên là “**Hero Card**” chứa các biểu đồ minh họa. Mục đích là để ngay từ cái nhìn đầu tiên, người dùng đã cảm nhận được đây là một ứng dụng quản lý tài chính chuyên nghiệp.



Hình 4. 3 Giao diện trang Đăng nhập

Bảng 4. 1 Đặc tả các thành phần Giao diện Đăng Nhập

STT	Thành phần giao diện	Mô tả chức năng
1	Logo và Branding	Nằm ở góc trên cùng bên trái, hiển thị Logo và tên ứng dụng “Expense Tracker” để định vị thương hiệu ngay lập tức.
2	Hero Section (Phải)	Khu vực trang trí bên phải, hiển thị một biểu đồ tài chính minh họa và các con số thống kê giả lập. Phần này giúp giao diện bớt nhảm chán và gợi mở về tính năng chính của ứng dụng.
3	Tiêu đề “Welcome Back”	Lời chào mừng thân thiện, xác nhận người dùng đang ở trang Đăng nhập chứ không phải Đăng ký.

4	Ô nhập Email	Cho phép người dùng nhập địa chỉ email đã đăng ký. Hệ thống có validate định dạng email ngay khi nhập.
5	Ô nhập Mật khẩu	Nơi nhập mật khẩu. Các ký tự được ẩn đi (dạng dấu chấm) để đảm bảo riêng tư.
6	Nút “Forgot Password?”	Liên kết dẫn đến trang Khôi phục mật khẩu, hỗ trợ trường hợp người dùng quên thông tin đăng nhập.
7	Nút “LOGIN”	Nút hành động chính (Call-to-Action). Khi nhấn, hệ thống sẽ gửi thông tin lên Firebase để xác thực. Nếu thành công, chuyển hướng vào Dashboard; nếu sai, hiện thông báo lỗi (Toast).
8	Liên kết “Sign Up”	Dòng chữ “Don't have an account?” kèm liên kết để chuyển hướng nhanh sang trang Đăng ký nếu người dùng chưa có tài khoản.

4.3.2 Giao diện Đăng ký (Sign Up)

Giao diện này có cấu trúc tương đồng với trang Đăng nhập (do sử dụng chung AuthLayout), giúp duy trì trải nghiệm nhất quán. Tuy nhiên, form nhập liệu sẽ có thêm trường “Full Name” để hệ thống có thể lưu tên hiển thị của người dùng ngay từ đầu. Một điểm mình chú trọng ở đây là trải nghiệm người dùng: sau khi đăng ký thành công, hệ thống sẽ tự động chuyển hướng chứ không bắt người dùng phải thao tác nhiều.

Create Account

It's free and easy to set up.

FULL NAME

 John Doe

EMAIL

 name@example.com

PASSWORD

 Create a strong password

Must be at least 6 characters.

Get Started →

Already a member? [Sign In](#)

Hình 4. 4 Giao diện trang Đăng Ký

Bảng 4. 2 Đặc tả các thành phần giao diện Đăng Ký

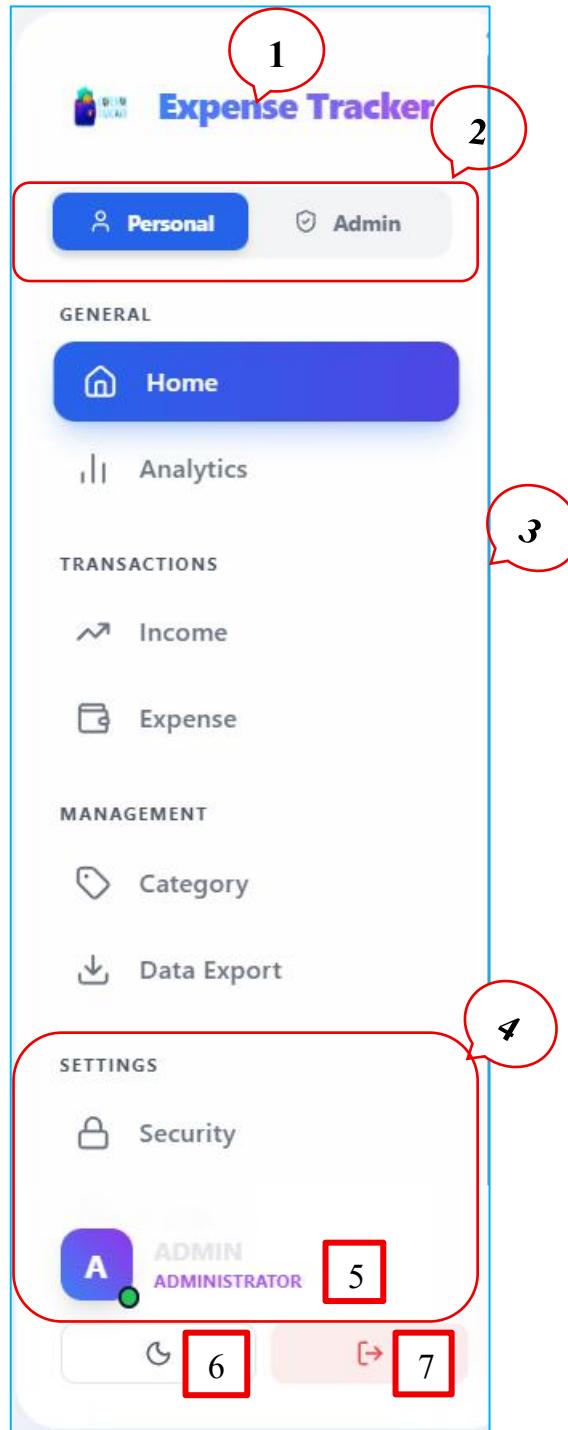
STT	Thành phần giao diện	Mô tả chức năng
1	Tiêu đề “Create an Account”	Xác định rõ đây là khu vực tạo tài khoản mới.

2	Ô nhập Full Name	Yêu cầu người dùng nhập tên đầy đủ. Thông tin này sẽ được dùng để hiển thị lời chào “Good Morning, [Name]” trên Dashboard sau này.
3	Ô nhập Email	Nhập email đăng ký. Hệ thống sẽ kiểm tra xem email này đã tồn tại trong CSDL chưa. Nếu trùng, sẽ thông báo lỗi ngay.
4	Ô nhập Mật khẩu	Nhập mật khẩu mới. Hệ thống có thể yêu cầu độ dài tối thiểu (VD: 6 ký tự) để đảm bảo an toàn.
5	Nút “SIGN UP”	Khi nhấn: <ul style="list-style-type: none"> - Tạo tài khoản trên Firebase Authentication. - Đồng thời tạo một bản ghi User mới trong PostgreSQL (Backend). - Tự động đăng nhập và chuyển vào trang chính.
6	Liên kết “Login”	Dòng chữ “Already have an account?” giúp người dùng quay lại trang Đăng nhập nếu lỡ bấm nhầm vào đây.
7	Thông báo (Toast)	Một popup nhỏ sẽ hiện ra ở góc màn hình để báo trạng thái: “Đăng ký thành công” (Màu xanh) hoặc “Email đã tồn tại” (Màu đỏ).

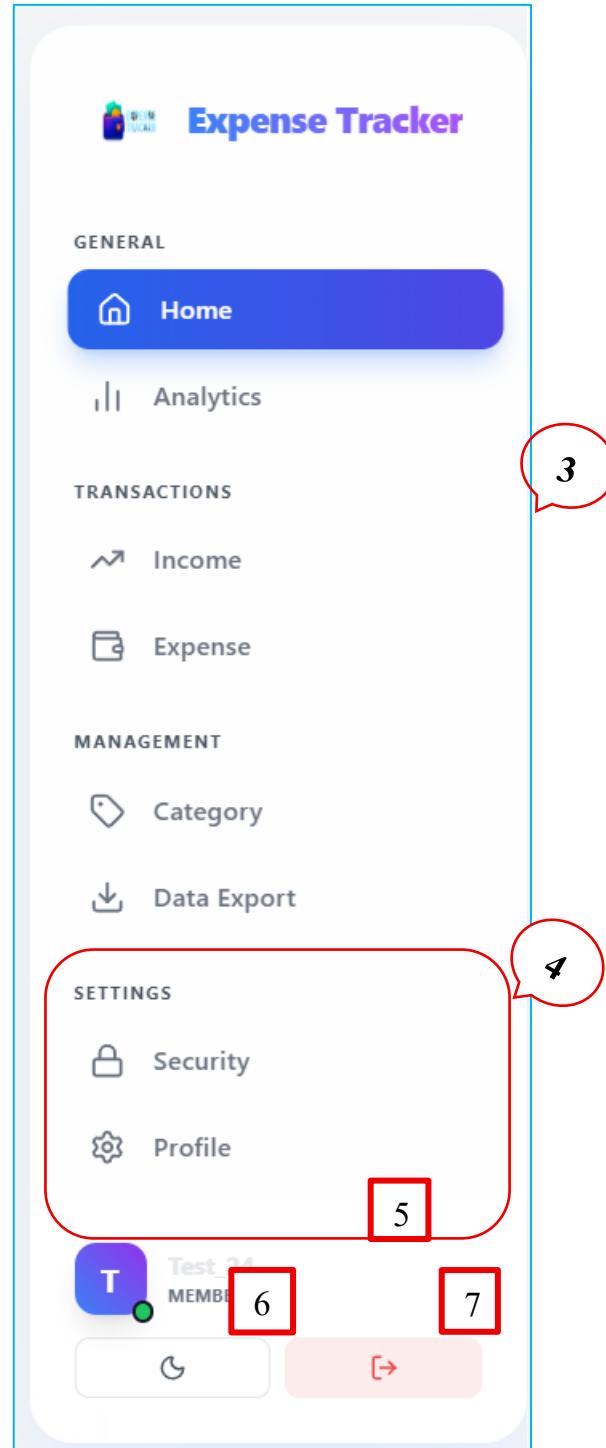
4.3.3 Giao diện Thanh điều hướng (Sidebar)

Khác với các website truyền thống sử dụng thanh menu ngang, hệ thống sử dụng thanh Sidebar dọc nằm cố định bên trái màn hình. Mình đã thiết kế Sidebar theo phong cách “Floating Glassmorphism” (Thủy tinh trôi nổi): nó không dính sát vào cạnh màn hình mà nằm tách biệt, bo tròn các góc và có hiệu ứng mờ nền, tạo cảm giác hiện đại và sang trọng.

Đặc biệt, Sidebar được tích hợp cơ chế “Workspace Switcher”, cho phép người dùng có quyền Admin chuyển đổi linh hoạt giữa không gian làm việc Cá nhân và Quản trị viên chỉ với một nút bấm.



Hình 4. 5 Thanh Sidebar Admin



Hình 4. 6 Thanh Sidebar Users

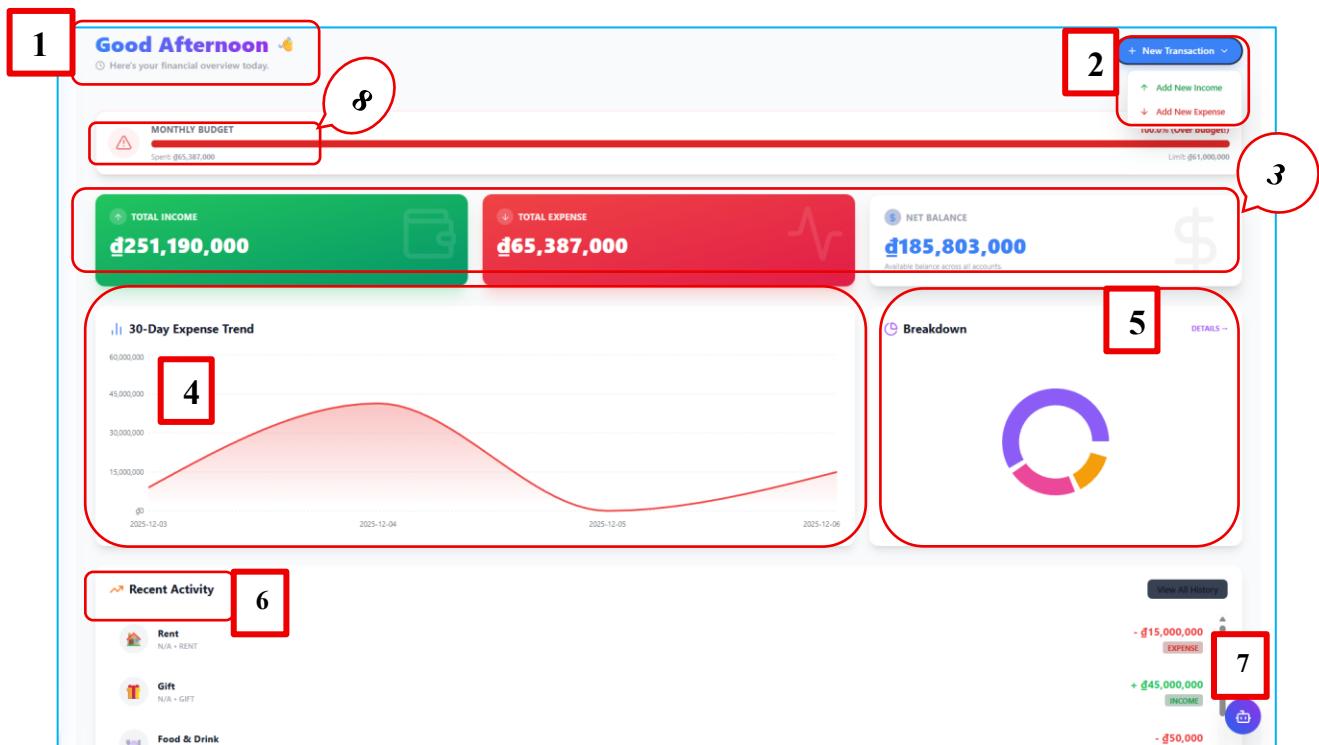
Bảng 4. 3 Đặc tả các thành phần Sidebar

STT	Thành phần giao diện	Mô tả chức năng
1	Logo và Tên ứng dụng	Nằm ở vị trí cao nhất, hiển thị Logo và tên “Expense Tracker” với hiệu ứng chữ Gradient (chuyển màu) để tạo điểm nhấn thương hiệu.
2	Workspace Switcher	(<i>Chỉ hiện với Admin</i>) Bộ chuyển đổi gồm 2 nút: “Personal” và “Admin”. Giúp người dùng chuyển đổi ngữ cảnh làm việc mà không cần đăng xuất/đăng nhập lại.
3	Menu Điều hướng chính	Chứa các liên kết đến các chức năng nghiệp vụ: Home (Tổng quan), Analytics (Phân tích), Income (Thu nhập), Expense (Chi tiêu), Category (Danh mục), Data Export (Xuất dữ liệu). Mục đang được chọn sẽ sáng lên (Glow effect).
4	Menu Cài đặt	Nhóm các chức năng cấu hình tài khoản: Security (Bảo mật 2 lớp) và Profile (Thông tin cá nhân).
5	Thẻ Hồ sơ (User Card)	Nằm dưới cùng, hiển thị Avatar (hoặc chữ cái đầu tên nếu chưa có ảnh), Tên người dùng và Vai trò (Member/Administrator).
6	Nút Giao diện (Theme)	Cho phép chuyển đổi qua lại giữa Dark Mode (Giao diện tối - mặc định) và Light Mode (Giao diện sáng) để bảo vệ mắt.
7	Nút Đăng xuất (Logout)	Cho phép người dùng thoát khỏi phiên làm việc hiện tại và xóa Token khỏi trình duyệt.

4.3.4 Giao diện Trang chủ (Dashboard)

Dashboard là trung tâm điều khiển của ứng dụng. Ngay khi đăng nhập, người dùng sẽ thấy ngay bức tranh toàn cảnh về tài chính của mình. Giao diện được chia thành

các khôi thông tin (Widgets) sắp xếp theo bố cục lưới thông minh (Smart Grid), ưu tiên các chỉ số quan trọng nhất lên đầu.



Hình 4. 7 Giao diện trang Dashboard (Users)

Bảng 4. 4 Đặc tả các thành phần trên Dashboard

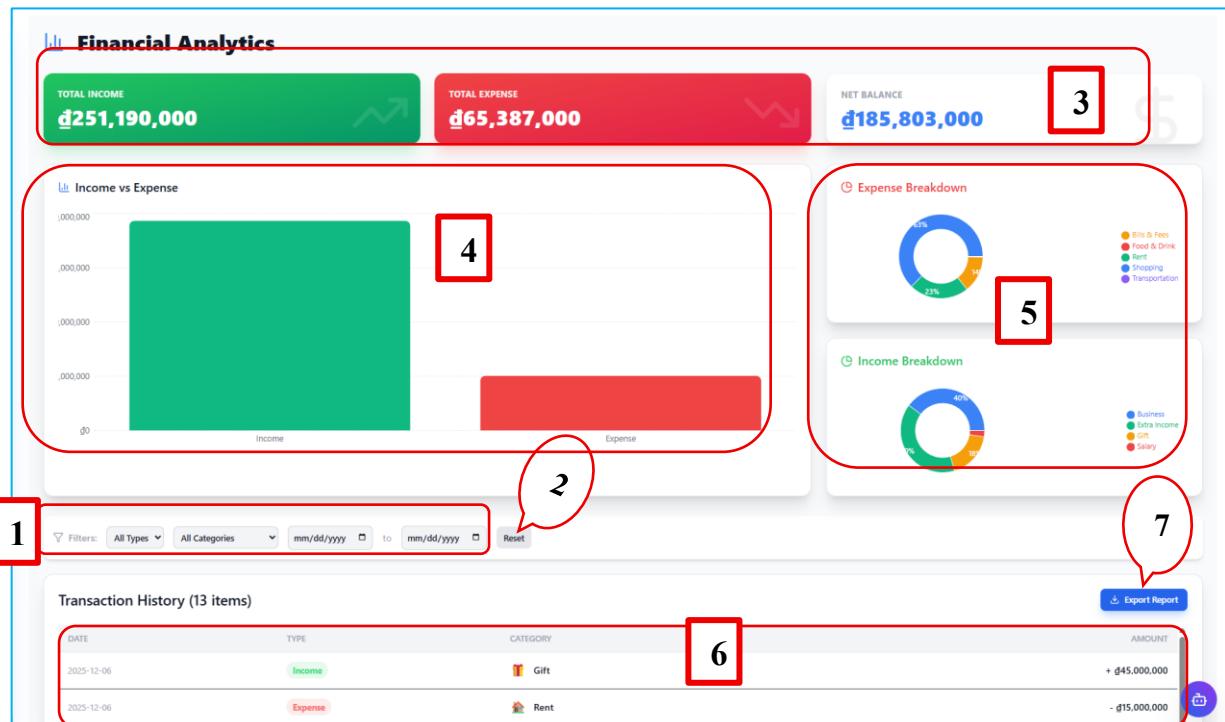
STT	Thành phần giao diện	Mô tả chức năng
1	Lời chào và Thông báo	- Hiển thị lời chào theo buổi (VD: “Good Morning”) kèm tên người dùng. - Hiển thị dòng thông báo chạy (Broadcast Message) từ Admin nếu có sự kiện quan trọng (VD: “Hệ thống bảo trì lúc 22h”).
2	Nút “New Transaction”	Nút hành động nhanh (Quick Action). Khi nhấn vào sẽ xổ xuống menu cho phép chọn thêm nhanh Thu nhập hoặc Chi tiêu mà không cần chuyển trang.

3	Thẻ KPI (KPI Cards)	3 thẻ hiển thị chỉ số tài chính cốt lõi: - Total Income: Tổng thu nhập (Màu xanh). - Total Expense: Tổng chi tiêu (Màu đỏ). - Net Balance: Số dư hiện tại (Màu xanh hoặc đỏ tùy vào số dư). Các thẻ này sử dụng hiệu ứng kính mờ giúp nổi bật trên nền tối.
4	Biểu đồ Xu hướng (Trend Chart)	Biểu đồ vùng (Area Chart) chiếm 2/3 màn hình, hiển thị biến động chi tiêu trong 30 ngày gần nhất. Giúp người dùng nhận diện ngay các ngày chi tiêu bất thường.
5	Biểu đồ Cơ cấu (Breakdown)	Biểu đồ tròn (Pie Chart) chiếm 1/3 màn hình, hiển thị tỷ lệ phần trăm chi tiêu theo từng danh mục (VD: Ăn uống chiếm bao nhiêu %).
6	Hoạt động gần đây (Recent Activity)	Danh sách liệt kê 5-10 giao dịch mới nhất. Mỗi dòng hiển thị Icon danh mục, Tên, Ngày và Số tiền (+ Xanh / - Đỏ).
7	Trợ lý ảo FinBot (Chat Widget)	Nút tròn nổi ở góc dưới bên phải. Khi nhấn vào sẽ mở cửa sổ chat với AI. Người dùng có thể gõ “Hôm nay tiêu gì?” hoặc “Vẽ biểu đồ” để tương tác trực tiếp.
8	Thanh biểu Hiện ngân sách chi tiêu	Thanh biểu hiện đề ra mỗi tháng đặt ngân sách của khách hàng, thanh sẽ thay đổi theo từng cấp độ màu khác và đến màu đó là báo hiệu đã vượt quá ngân sách.

4.3.5 Giao diện Phân tích tài chính (Analytics)

Nếu Dashboard chỉ cung cấp cái nhìn tổng quan, thì trang Analytics là nơi người dùng đào sâu vào dữ liệu. Mình đã thiết kế trang này theo bố cục Smart Grid bất đối xứng (2/3 cho biểu đồ chính và 1/3 cho biểu đồ phụ) để tối ưu không gian hiển thị. Điểm

nhấn của trang này là khả năng lọc dữ liệu linh hoạt, giúp người dùng trả lời được các câu hỏi như: “*Tháng trước mình tiêu bao nhiêu cho ăn uống?*”.



Hình 4. 8 Giao diện trang Analytics

Bảng 4. 5 Đặc tả các thành phần trên trang Analytics

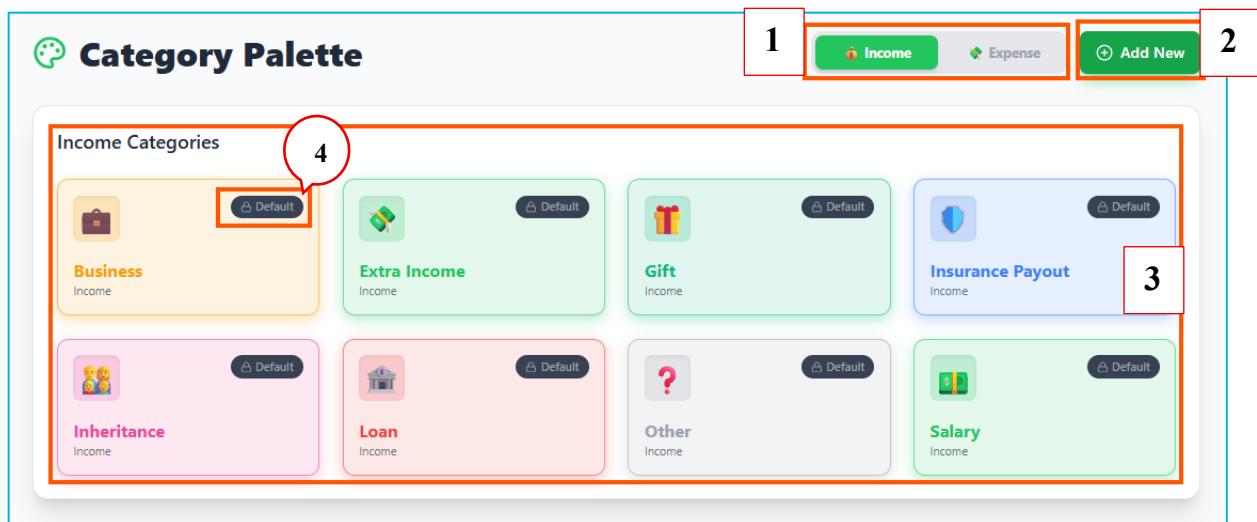
STT	Thành phần / Ký hiệu	Mô tả chức năng
1	Thanh Bộ lọc (Filter Bar)	Nằm ngay trên đầu, cho phép lọc dữ liệu theo 3 tiêu chí: - Type : Chọn xem Income, Expense hoặc Tất cả. - Category : Chọn xem một danh mục cụ thể (VD: chỉ xem “Lương”). - Date Range : Chọn ngày bắt đầu và kết thúc tùy ý.

2	Nút “Reset”	Xóa toàn bộ bộ lọc đang chọn để quay về trạng thái xem mặc định (Toàn thời gian).
3	KPI Cards (Thẻ chỉ số)	Hiển thị 3 con số tổng kết dựa trên bộ lọc hiện tại: - Total Income: Tổng thu nhập trong khoảng đã chọn. - Total Expense: Tổng chi tiêu trong khoảng đã chọn. - Net Balance: Số dư còn lại (Thu - Chi).
4	Biểu đồ So sánh (Bar Chart)	Biểu đồ cột lớn, so sánh trực quan giữa Tổng Thu và Tổng Chi. Khi di chuột vào cột, Tooltip sẽ hiển thị số tiền cụ thể.
5	Biểu đồ Cơ cấu (Donut Chart)	Hai biểu đồ tròn xếp chồng, thể hiện tỷ trọng phần trăm của từng danh mục (VD: Miếng bánh “Ăn uống” chiếm 45%). Có chú thích (Legend) màu sắc rõ ràng bên cạnh.
6	Bảng Dữ liệu Chi tiết	Danh sách liệt kê từng giao dịch thỏa mãn điều kiện lọc. Các cột gồm: Ngày, Loại, Danh mục (kèm Icon), Số tiền.
7	Nút “Export Report”	Cho phép tải dữ liệu đang hiển thị trên bảng về máy dưới dạng file Excel (.xlsx) để lưu trữ.

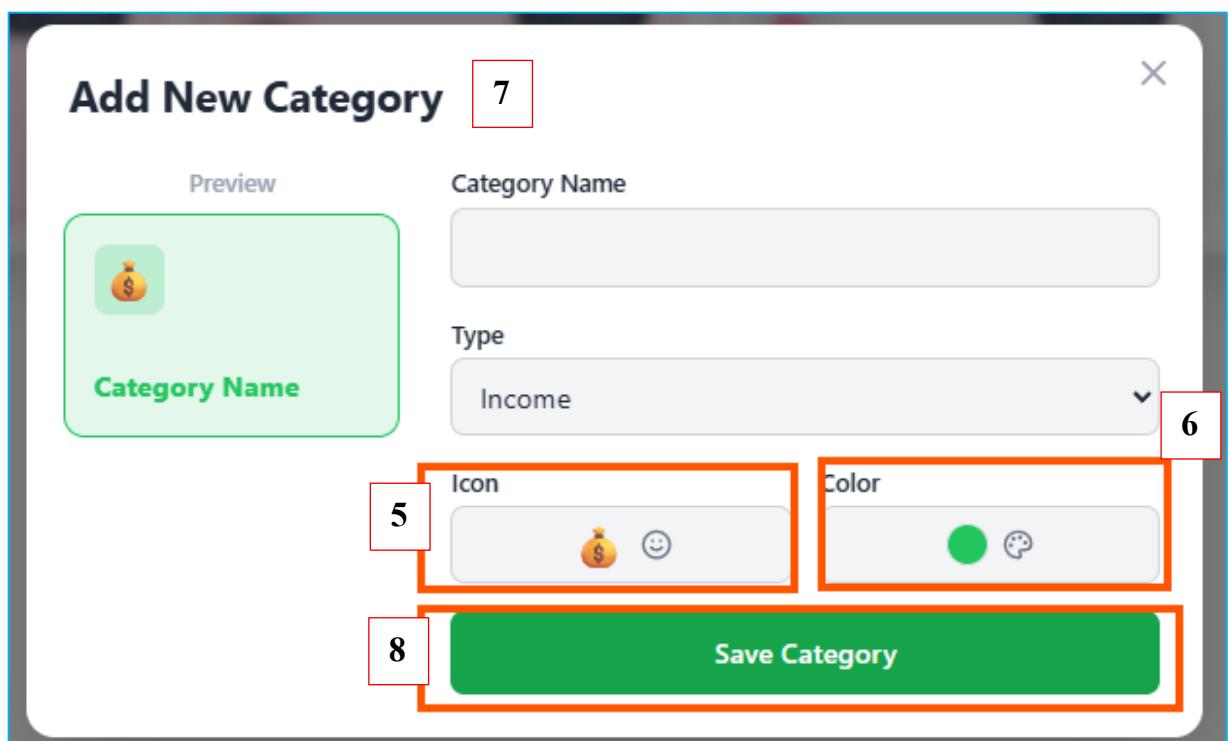
4.3.5 Giao diện Quản lý Danh mục (Category Palette)

Đây là trang giúp người dùng cá nhân hóa hệ thống. Mình đặt tên là Category Palette vì giao diện ở đây rực rỡ như một bảng màu. Thay vì danh sách dạng bảng nhảm

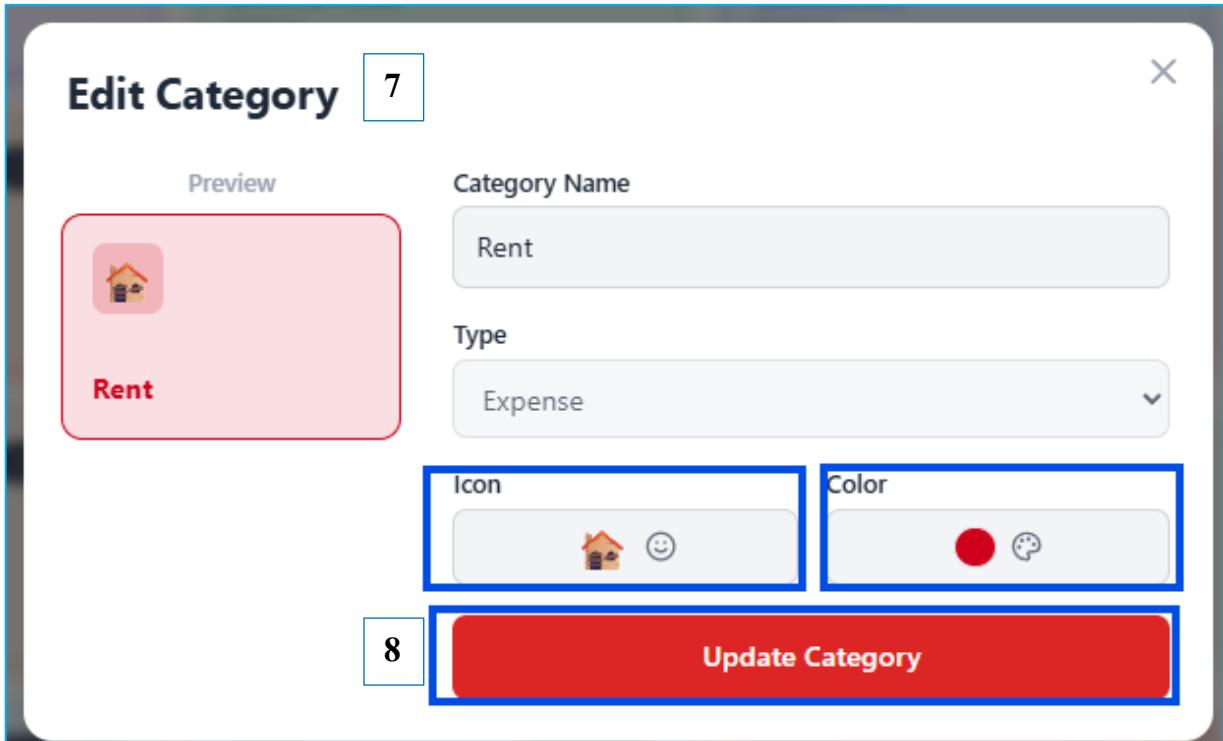
chán, mình sử dụng các thẻ bài (Cards) có hiệu ứng phát sáng (Glow) theo màu của danh mục, tạo cảm giác thú vị khi tương tác.



Hình 4. 9 Giao diện trang Category



Hình 4. 10 Giao diện Thêm Danh Mục Mới



Hình 4. 11 Giao diện Sửa Danh Mục

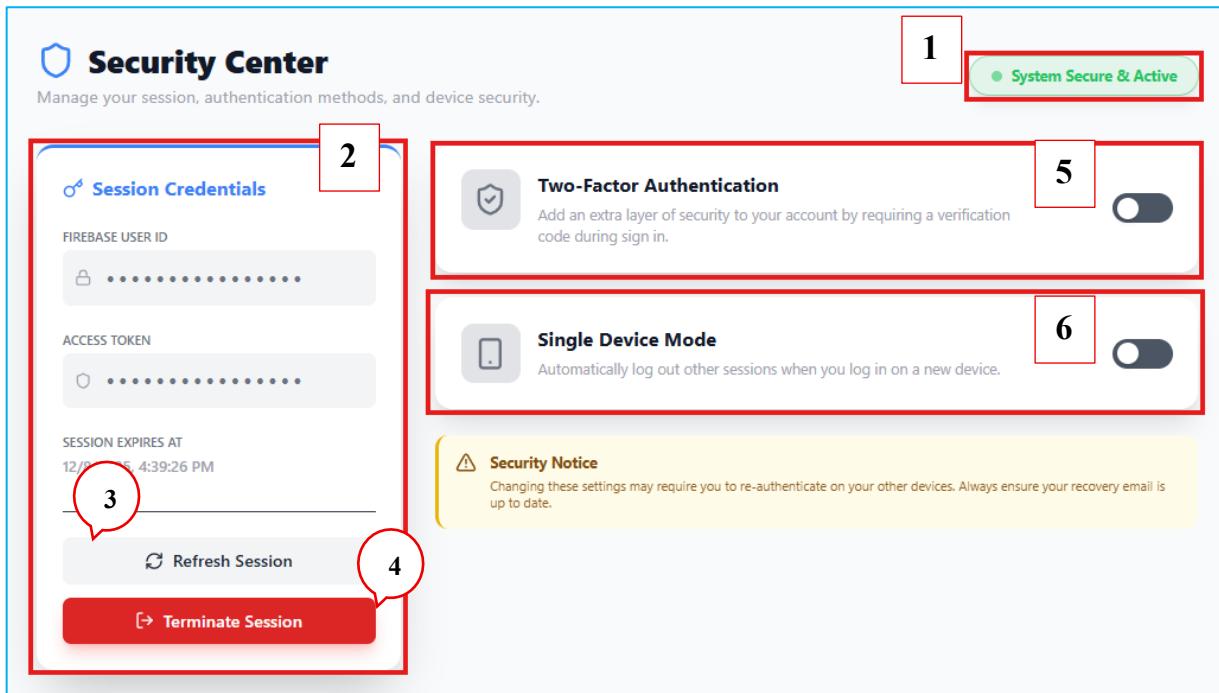
Bảng 4. 6 Đặc tả các thành phần trên trang Danh Mục (Category)

STT	Thành phần / Ký hiệu	Mô tả chức năng
1	Bộ chuyển đổi (Toggle)	Hai nút bấm lớn “Income” và “Expense” để chuyển qua lại giữa danh sách danh mục Thu nhập và Chi tiêu.
2	Nút “Add New”	Mở cửa sổ (Modal) để tạo danh mục mới.
3	Thẻ Danh mục (Category Card)	Mỗi thẻ đại diện cho một danh mục, bao gồm: - Icon: Biểu tượng Emoji ở giữa. - Tên: Tên danh mục (VD: Food và Drink). - Màu nền: Hiệu ứng màu mờ (Opacity) dựa trên màu chủ đạo của danh mục đó.

4	Biểu tượng Ô khóa (Lock)	(Chỉ hiện trên Danh mục Mặc định) Ký hiệu cho biết đây là danh mục của hệ thống, người dùng không thể Sửa hoặc Xóa.
5	Nút Sửa (Edit Icon)	(Chỉ hiện trên Danh mục Cá nhân) Mở Modal để đổi tên, đổi màu hoặc icon.
6	Nút Xóa (Trash Icon)	(Chỉ hiện trên Danh mục Cá nhân) Xóa danh mục khỏi hệ thống. Có hiển thị cảnh báo xác nhận trước khi xóa.
7	Modal Tạo/Sửa (Popup)	Cửa sổ nhập liệu gồm: <ul style="list-style-type: none"> - Name: Nhập tên danh mục. - Type: Chọn loại Thu/Chi. - Icon Picker: Bảng chọn Emoji sinh động. - Color Picker: Bảng chọn mã màu Hex.
8	Nút Save/ Update	Lưu và Cập nhật khi người dùng hoặc quản trị viên đã hoàn thành

4.1.1 Giao diện bảo mật (Security Center)

Bảo mật là yếu tố sống còn của ứng dụng tài chính. Trang này được thiết kế như một Trung tâm kiểm soát (Control Center) với các thông tin về phiên đăng nhập và các công tắc (Toggle Switches) để bật tắt tính năng bảo vệ.



Hình 4. 12 Giao diện trang Security

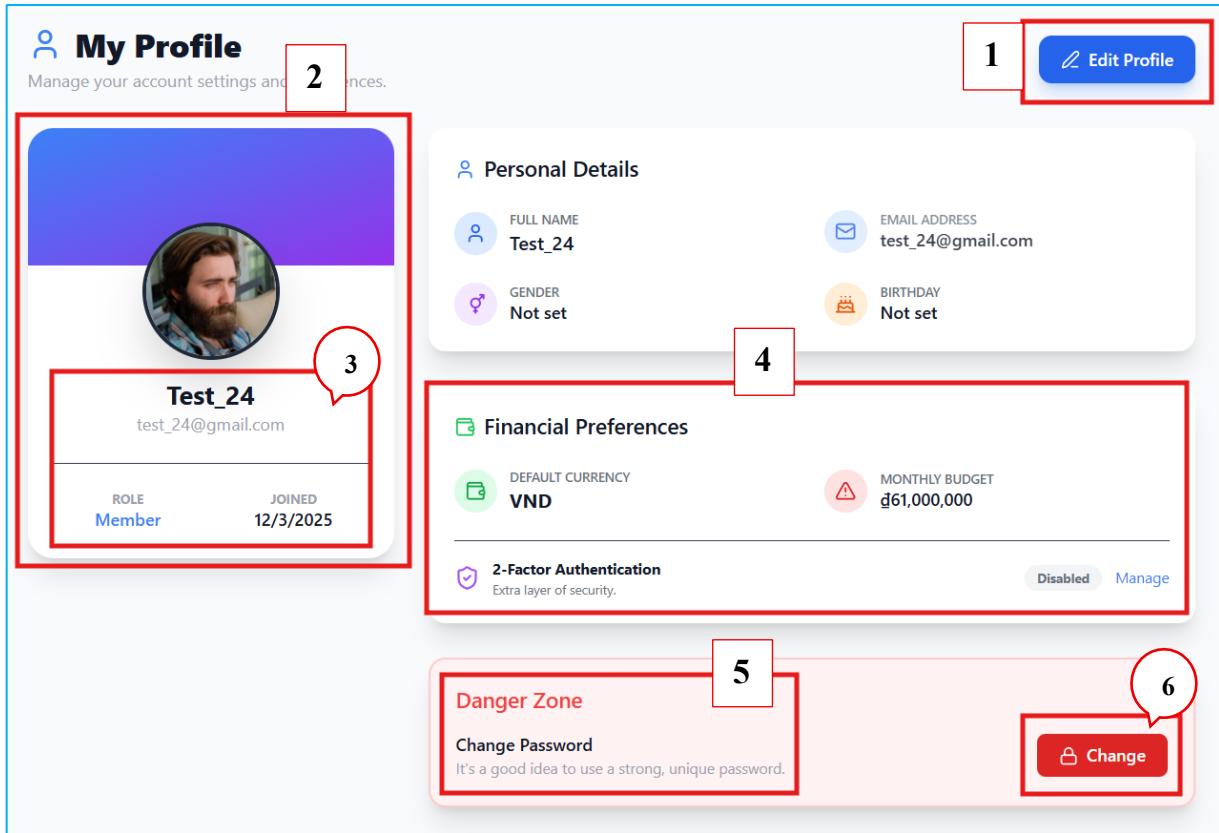
Bảng 4. 7 Đặc tả các thành phần trên trang Bảo Mật (Security)

STT	Thành phần / Ký hiệu	Mô tả chức năng
1	Trạng thái Phiên (Status Badge)	Một nhãn (Badge) hiển thị trạng thái kết nối thực: “Session Active” (Xanh) hoặc “Expired” (Đỏ). Giúp người dùng biết ngay lập tức liệu kết nối của họ có ổn định và an toàn hay không.
2	Thông tin Phiên (Session Info)	Hiển thị các thông số kỹ thuật của phiên làm việc hiện tại: - Token: Chuỗi mã hóa JWT (được che bót để bảo mật).

		<ul style="list-style-type: none"> - User ID: Định danh người dùng trên hệ thống Firebase. - Expires At: Thời gian cụ thể phiên đăng nhập sẽ hết hạn.
3	Nút “Refresh Token”	Cho phép người dùng làm mới token xác thực (gửi request lên Firebase) mà không cần đăng nhập lại. Tính năng này hữu ích khi phiên làm việc sắp hết hạn.
4	Nút “Force Logout”	Chức năng đăng xuất khẩn cấp. Khi kích hoạt, hệ thống sẽ xóa toàn bộ token trong LocalStorage và ngắt kết nối ngay lập tức.
5	Bật/Tắt 2FA (Toggle Switch)	<p>Công tắc kích hoạt Xác thực 2 yếu tố (Two-Factor Authentication).</p> <ul style="list-style-type: none"> - Khi bật: Hệ thống hiển thị mã QR để quét bằng ứng dụng Authenticator. - Tác dụng: Tăng cường bảo mật, yêu cầu mã OTP mỗi khi đăng nhập.
6	Giới hạn thiết bị (Device Restriction)	Tùy chọn cho phép tài khoản chỉ được đăng nhập trên một thiết bị duy nhất tại một thời điểm. Nếu đăng nhập nơi khác, phiên cũ sẽ bị đăng xuất.
7	Cảnh báo (Warning Alert)	Khu vực thông báo màu đỏ/vàng phía dưới, nhắc nhở người dùng về hệ quả của việc đăng xuất hoặc thay đổi cài đặt bảo mật.

4.1.2 Giao diện Hồ sơ người dùng (My Profile)

Trang Hồ sơ không chỉ là nơi hiển thị thông tin cá nhân mà còn đóng vai trò là trung tâm quản lý định danh (Identity Management). Giao diện được thiết kế theo bố cục Dashboard 2 cột: Cột trái hiển thị nhận diện (Avatar, Tên), cột phải hiển thị thông tin chi tiết và các thiết lập ứng dụng. Cách bố trí này giúp tận dụng tối đa không gian màn hình và tạo cảm giác chuyên nghiệp, gọn gàng.



Hình 4. 13 Giao diện trang cá nhân (Profile)

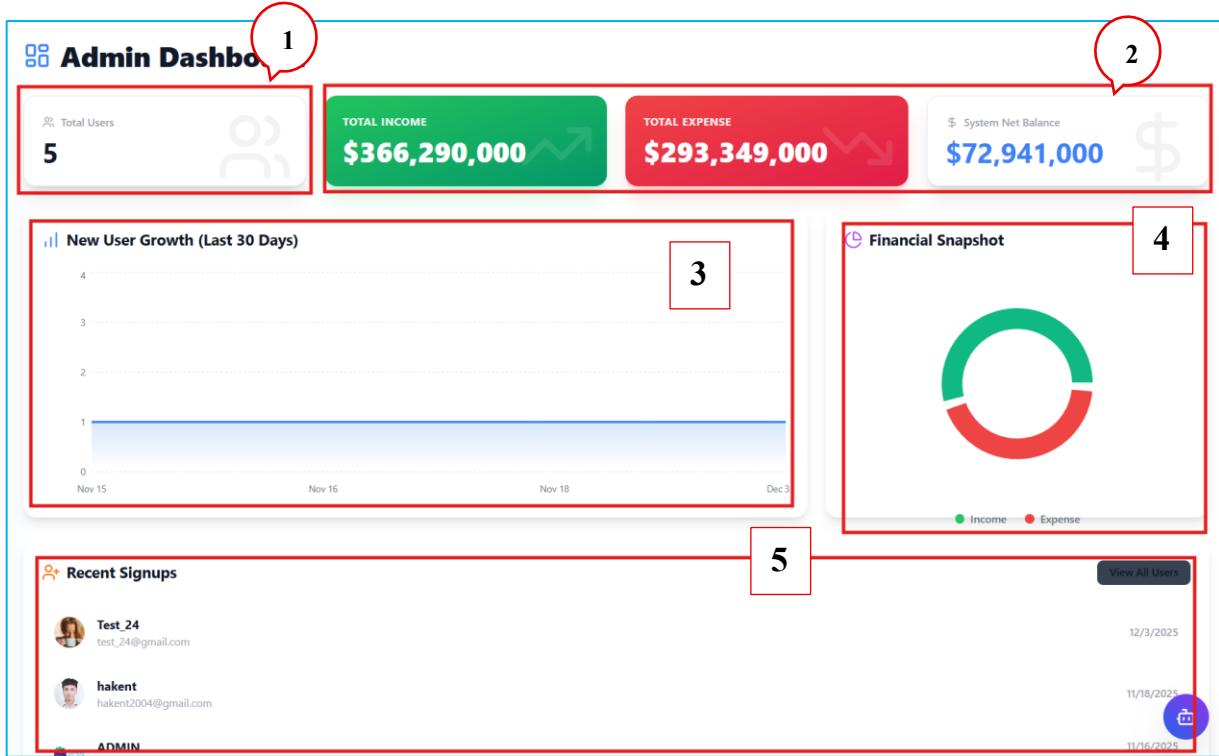
Bảng 4. 8 Đặc tả các thành phần trên trang Hồ sơ (Profile)

STT	Thành phần / Ký hiệu	Mô tả chức năng
1	Nút “Edit Profile”	Nút hành động chính nằm ở góc trên phải. Khi nhấp vào, giao diện sẽ chuyển sang chế độ Chính sửa (Edit Mode) , biến các dòng chữ thông tin thành các ô nhập liệu (Input) để người dùng thay đổi dữ liệu.
2	Thẻ Định danh (Identity Card)	Khối bên trái hiển thị: <ul style="list-style-type: none"> - Avatar: Ảnh đại diện của người dùng (bo tròn). - Tên hiển thị và Email: Thông tin cơ bản. - Ngày tham gia: Thời điểm tài khoản được tạo.

3	Thông tin Chi tiết (Personal Details)	Hiển thị các thông tin nhân khẩu học bổ sung: - Gender: Giới tính (Nam/Nữ/Khác). - Birthday: Ngày sinh nhật. Các thông tin này giúp cá nhân hóa trải nghiệm người dùng.
4	Cấu hình ứng dụng (Finance Preference)	Khu vực quản lý các thiết lập riêng của người dùng: - Default Currency: Đơn vị tiền tệ mặc định (USD/ VND) - 2FA Security: Trạng thái bảo mật (Active/Inactive) kèm liên kết nhanh đến trang Security để quản lý. - Monthly Budget: Ngân sách chi tiêu cho 1 tháng.
5	Khu vực Nguy hiểm (Danger Zone)	Được thiết kế tách biệt với viền đỏ để cảnh báo sự chú ý. Chứa chức năng “ Change Password ” (Đổi mật khẩu), giúp người dùng cập nhật mật khẩu mới một cách an toàn.
6	Modal Đổi mật khẩu	(<i>Xuất hiện khi nhấn Change Password</i>) Giao diện nhập mật khẩu cũ, mật khẩu mới và xác nhận. Có tích hợp thanh đo độ mạnh mật khẩu (Strength Meter) để khuyến khích đặt mật khẩu an toàn.

4.1.3 Giao diện Bảng điều khiển Quản Trị (Admin Dashboard)

Khác với Dashboard của người dùng tập trung vào chi tiêu cá nhân, Dashboard của Admin đóng vai trò là Tháp kiểm soát. Tại đây, quản trị viên có thể nắm bắt nhanh sức khỏe của toàn bộ hệ thống thông qua các chỉ số vĩ mô và biểu đồ tăng trưởng. Giao diện được thiết kế dạng lưới (Grid) với các thẻ KPI nổi bật phía trên và các biểu đồ phân tích chi tiết phía dưới.



Hình 4. 14 Giao diện trang Admin Dashboard

Bảng 4. 9 Đặc tả các thành phần trên trang Admin Dashboard

STT	Thành phần / Ký hiệu	Mô tả chức năng và Ý nghĩa nghiệp vụ
1	Thẻ KPI Tổng User	Hiển thị tổng số lượng tài khoản đã đăng ký vào hệ thống. Giúp Admin theo dõi quy mô người dùng.
2	Thẻ KPI Dòng tiền (Income/Expense)	Tổng hợp toàn bộ số tiền Thu nhập và Chi tiêu đã được ghi nhận trên toàn hệ thống. Chỉ số này phản ánh mức độ hoạt động (volume) của ứng dụng.
3	Biểu đồ Tăng trưởng (User Growth)	Biểu đồ vùng (Area Chart) thể hiện số lượng người dùng mới đăng ký trong 30 ngày gần nhất. Giúp đánh giá hiệu quả của các chiến dịch thu hút người dùng.
4	Biểu đồ Cơ cấu Hệ thống	Biểu đồ tròn (Donut Chart) so sánh tỷ lệ giữa dòng tiền vào và ra của toàn hệ thống.

5	Danh sách Đăng ký mới	Bảng liệt kê 5 tài khoản vừa đăng ký gần nhất (Avatar, Tên, Email). Giúp Admin phát hiện nhanh các tài khoản spam hoặc đáng ngờ.
---	------------------------------	--

4.1.4 Giao diện Quản lý người dùng (User Management)

Đây là chức năng quan trọng nhất để duy trì trật tự của hệ thống. Giao diện cung cấp công cụ để Admin tìm kiếm, xem xét và xử lý các tài khoản người dùng.

The screenshot shows the 'User Management' page with the following elements:

- Statistics:**
 - Total Users: 5 (circled 1)
 - 2FA Enabled: 2 (circled 2)
 - New Users (24h): +0 (circled 3)
 - Search bar: Search by name or email... (circled 4)
- User List:**

USER	STATUS	2FA	JOINED DATE	ACTION
Test_24 test_24@gmail.com	Active	Off	Dec 3, 2025	<input checked="" type="checkbox"/> <input type="button" value="Edit"/>
hakent hakent2004@gmail.com	Active	On	Nov 18, 2025	<input type="checkbox"/> <input type="button" value="Edit"/>
ADMIN admin@gmail.com	Admin	Off	Nov 16, 2025	<input type="checkbox"/> <input type="button" value="Edit"/>
User_123 user123@gmail.com	Active	Off	Nov 15, 2025	<input checked="" type="checkbox"/> <input type="button" value="Edit"/>
Thø havantho2004@gmail.com	Active	On	Oct 5, 2025	<input type="checkbox"/> <input type="button" value="Edit"/>

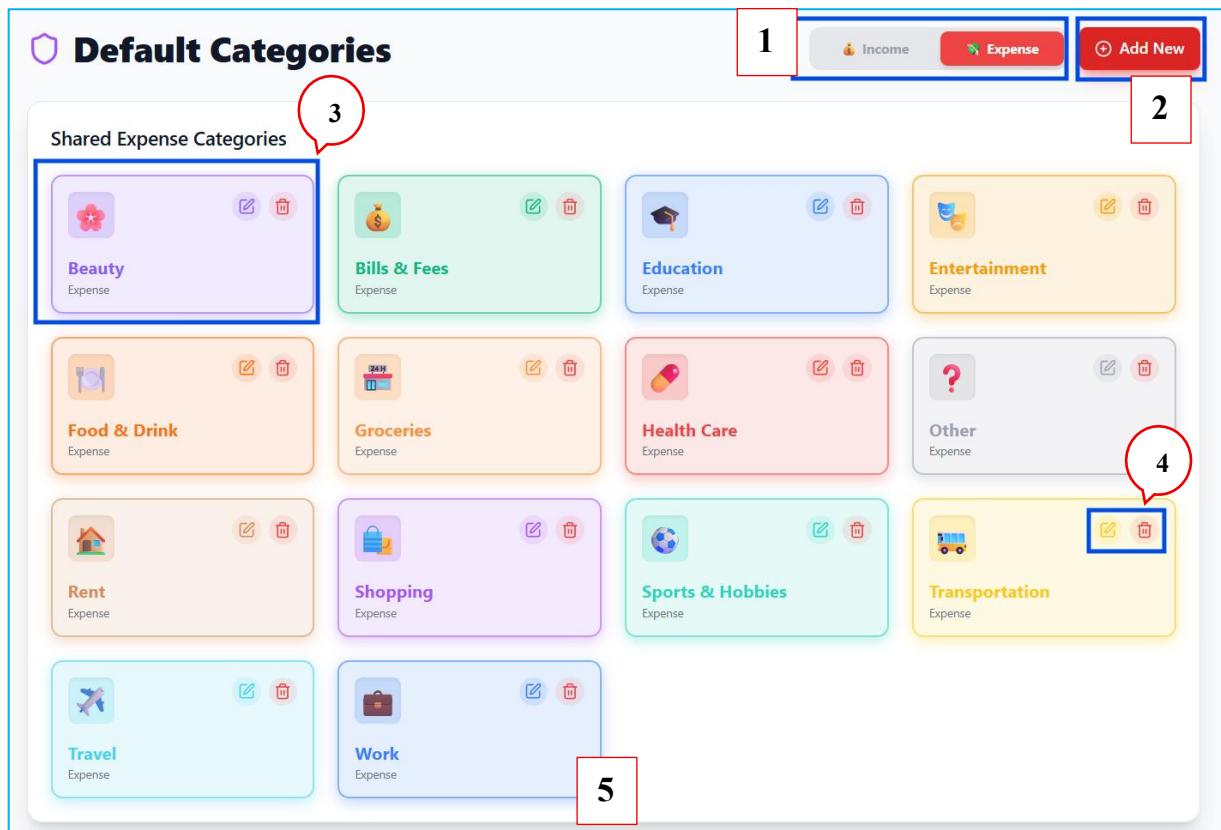
Hình 4. 15 Giao diện trang Quản lý người dùng (Admin)

Bảng 4. 10 Đặc tả các thành phần trên trang Quản lý người dùng (Admin)

STT	Thành phần / Ký hiệu	Mô tả chức năng và Ý nghĩa nghiệp vụ
1	Thanh Tìm kiếm (Search Bar)	Cho phép Admin nhập tên hoặc email để lọc nhanh danh sách người dùng, giúp tiết kiệm thời gian khi quản lý số lượng lớn tài khoản.
2	Thẻ Thống kê nhanh	Hiển thị 3 chỉ số ngay trên bảng dữ liệu: Tổng số User, Số lượng User đang bật bảo mật 2 lớp (2FA), và Số User mới trong 24h.
3	Cột Thông tin User	Hiển thị Avatar, Tên hiển thị và Email của người dùng. Giúp nhận diện danh tính chính xác.
4	Nhãn Trạng thái (Status Badge)	- Admin (Màu tím) : Tài khoản có quyền quản trị cao cấp. - Active (Màu xanh) : Tài khoản người dùng đang hoạt động bình thường.
5	Cột 2FA (Bảo mật)	Biểu tượng Check xanh (Đã bật) hoặc X xám (Chưa bật). Giúp Admin đánh giá mức độ an toàn của các tài khoản.
6	Cột Ngày tham gia	Hiển thị ngày tài khoản được khởi tạo, phục vụ việc tra cứu thâm niên sử dụng.
7	Nút Chỉnh sửa (Edit Icon)	Mở cửa sổ (Modal) để Admin cập nhật thông tin người dùng hoặc cấp quyền Quản trị (Is Admin) cho tài khoản đó.
8	Nút Xóa (Trash Icon)	Khi nhấp vào, hệ thống sẽ hiện cảnh báo xác nhận. Nếu đồng ý, tài khoản sẽ bị xóa vĩnh viễn khỏi cả Cơ sở dữ liệu và Firebase Authentication.

4.1.5 Giao diện Quản lý danh mục Mặc định (System Categories)

Để đảm bảo tính chuẩn hóa dữ liệu ngay từ đầu, Admin cung cấp một bộ danh mục mẫu (như Ăn uống, Lương, Di chuyển...). Trang này cho phép Admin thêm, sửa, xóa các danh mục dùng chung đó.



Hình 4. 16 Giao diện trang Danh mục hệ thống (Admin)

Bảng 4. 11 Đặc tả các thành phần trên trang Danh mục hệ thống (Admin)

STT	Thành phần / Ký hiệu	Mô tả chức năng và Ý nghĩa nghiệp vụ
1	Bộ lọc Loại (Toggle)	Chuyển đổi xem danh sách danh mục thuộc nhóm Income (Thu nhập) hay Expense (Chi tiêu).
2	Nút “Add New”	Mở Form để tạo mới một danh mục hệ thống. Danh mục này sau khi tạo sẽ xuất hiện cho tất cả người dùng mới đăng ký.

3	Thẻ Danh mục (Card)	Hiển thị trực quan: Icon (Emoji), Tên danh mục và Màu sắc đại diện.
4	Nút Sửa/Xóa	Admin có toàn quyền chỉnh sửa tên, icon, màu sắc hoặc xóa bỏ danh mục hệ thống nếu thấy không còn phù hợp.
5	Bảng chọn (Pickers)	Tích hợp công cụ chọn Emoji và Bảng mã màu (Color Picker) giúp Admin thiết kế danh mục sinh động, trực quan.

4.1.6 Giao diện Nhật ký hoạt động (Audit Logs)

Đây là tính năng nâng cao giúp đảm bảo tính minh bạch và khả năng truy vết (Traceability) của hệ thống. Mọi hành động nhạy cảm của Admin đều được ghi lại tại đây, hoạt động như một “Hộp đen”.

Status	Timestamp	Action	Actor (Admin)	Target	Details
OK	Dec 7, 10:21:26 PM	REVOKE_ADMIN	admin@gmail.com	test_24@gmail.com	[127.0.0.1] Changed Admin privileges to False
OK	Dec 7, 10:21:16 PM	GRANT_ADMIN	admin@gmail.com	test_24@gmail.com	[127.0.0.1] Changed Admin privileges to True
OK	Dec 6, 09:48:36 PM	REVOKE_ADMIN	admin@gmail.com	havantho2004@gmail.com	[127.0.0.1] Changed Admin privileges to False
OK	Dec 5, 11:07:48 PM	GRANT_ADMIN	admin@gmail.com	havantho2004@gmail.com	[127.0.0.1] Changed Admin privileges to True
OK	Dec 5, 11:06:57 PM	CREATE_CATEGORY	admin@gmail.com	Rent	[127.0.0.1] Created default category 'Rent' (expense)
OK	Nov 28, 12:33:56 PM	DELETE_USER	admin@gmail.com	auditlog@gmail.com	[127.0.0.1] User deleted successfully
ERR	Nov 28, 12:19:36 PM	DELETE_USER	admin@gmail.com	auditlog@gmail.com	[127.0.0.1] cannot unpack non-iterable bool object

Hình 4. 17 Giao diện trang Nhật Ký Hoạt Động (Admin)

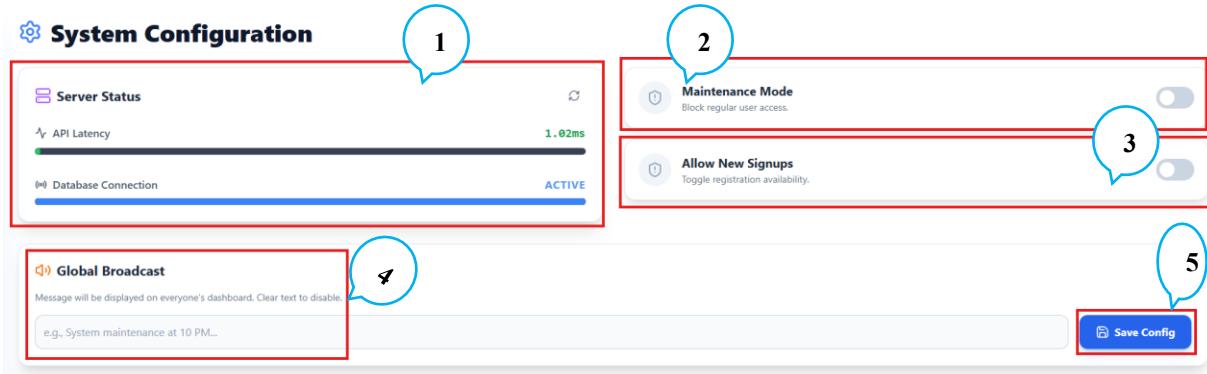
Bảng 4. 12 Đặc tả các thành phần trên trang Nhật Ký Hoạt Động (Admin)

STT	Thành phần / Ký hiệu	Mô tả chức năng và Ý nghĩa nghiệp vụ
1	Thanh Lọc (Filter Input)	Cho phép tìm kiếm log theo từ khóa: Email người thực hiện, Hành động (Action), hoặc Đối tượng bị tác động.

2	Nút “Refresh”	Tải lại dữ liệu log mới nhất từ Server mà không cần tải lại cả trang web.
3	Cột Trạng thái (Status)	- OK (Xanh) : Hành động thực hiện thành công. - ERR (Đỏ) : Hành động gặp lỗi (ví dụ: lỗi kết nối Firebase, lỗi Database).
4	Cột Thời gian (Timestamp)	Ghi nhận chính xác thời điểm hành động xảy ra (Ngày, Giờ, Phút).
5	Cột Hành động (Action)	Mã hành động nghiệp vụ, ví dụ: DELETE_USER (Xóa người dùng), UPDATE_SETTINGS (Sửa cài đặt).
6	Cột Người thực hiện (Actor)	Hiển thị Email của Admin đã thực hiện thao tác này. Đảm bảo tính trách nhiệm (Accountability).
7	Cột Đối tượng (Target)	Hiển thị đối tượng chịu tác động (Ví dụ: Email của user bị xóa).
8	Cột Chi tiết (Details)	Hiển thị thông tin kỹ thuật bổ sung, bao gồm cả địa chỉ IP của người thực hiện, giúp phát hiện các truy cập bất thường.

4.1.7 Giao diện Cấu hình hệ thống (System Settings)

Đây là trang quản trị cấp cao nhất, nơi Admin kiểm soát các tham số vận hành của toàn bộ website. Giao diện được thiết kế tối giản nhưng hiện đại, sử dụng các công tắc (Toggles) và ô nhập liệu trực quan.



Hình 4. 18 Giao diện trang Cấu hình hệ thống

Bảng 4. 13 Đặc tả các thành phần trên trang Cấu Hình Hệ Thống (Admin)

STT	Thành phần / Ký hiệu	Mô tả chức năng và Ý nghĩa nghiệp vụ
1	Server Status (Trạng thái Máy chủ)	Một bảng nhỏ hiển thị các chỉ số kỹ thuật giả lập như “API Latency” (Độ trễ) và “Database Connection” (Kết nối CSDL). Giúp Admin biết hệ thống có đang “khỏe” hay không.
2	Toggle “Maintenance Mode”	Chế độ bảo trì. Khi bật, hệ thống sẽ ngăn chặn các truy cập từ người dùng thường hoặc hiển thị trang “Đang bảo trì”. Dùng khi cần nâng cấp server hoặc sửa lỗi gấp.
3	Toggle “Allow New Signups”	Cho phép hoặc chặn đăng ký mới. Admin có thể tắt chức năng này nếu hệ thống quá tải hoặc đang trong giai đoạn đóng cửa thử nghiệm.
4	Global Broadcast (Thông báo toàn cục)	Ô nhập liệu cho phép Admin soạn thảo một tin nhắn. Khi nhấn “Save Config”, tin nhắn này sẽ chạy hoặc hiển thị nổi bật trên Dashboard của TẤT CẢ người dùng đang online. Dùng để thông báo sự kiện hoặc lịch bảo trì.
5	Nút “Save Config”	Lưu lại các thay đổi cấu hình vào bảng system_settings trong cơ sở dữ liệu.

KẾT LUẬN

Kết quả đạt được:

Sau quá trình nghiên cứu, phân tích, thiết kế và cài đặt, đề tài “**Phát triển hệ thống website quản lý tiêu dùng cá nhân bằng ReactJS và PostgreSQL**” đã hoàn thành và đáp ứng được các mục tiêu đề ra ban đầu.

Cụ thể, hệ thống đã đạt được những kết quả sau:

- **Xây dựng thành công quy trình nghiệp vụ**

Từ việc đăng ký/đăng nhập, quản lý hồ sơ, cho đến các chức năng cốt lõi là ghi chép thu chi và xem báo cáo thống kê.

- **Ứng dụng công nghệ**

- **Frontend:** Giao diện ReactJS mượt mà, chuẩn SPA, thiết kế Glassmorphism đẹp mắt và Responsive tốt trên thiết bị di động.
- **Backend:** API FastAPI xử lý tốc độ cao, cấu trúc code chuẩn mực (MVC/Layered Architecture), đảm bảo tính bảo trì và mở rộng.
- **Database:** Thiết kế CSDL PostgreSQL chuẩn hóa, đảm bảo toàn vẹn dữ liệu.

- **Tích hợp trí tuệ nhân tạo:**

Xây dựng thành công Chatbot **FinBot** sử dụng mô hình ngôn ngữ lớn (Google Gemini). Bot có khả năng hiểu tiếng Việt tự nhiên, tự động phân loại giao dịch, tra cứu số liệu và thậm chí vẽ biểu đồ phân tích ngay trong khung chat.

- **Hệ thống Quản trị & Bảo mật mạnh mẽ**

- Phân quyền chặt chẽ giữa User và Admin.
- Cơ chế xác thực 2 lớp (2FA) và quản lý phiên đăng nhập.
- Hệ thống Audit Logs giúp truy vết mọi hành động nhạy cảm.

- **Các chức năng của hệ thống website, gồm có:**

Hệ thống Expense Tracker được xây dựng hoàn thiện với đầy đủ các phân hệ chức năng phục vụ cho hai đối tượng chính là Người dùng và Quản trị viên, cụ thể như sau:

1. Nhóm chức năng dành cho Người dùng (User)

- Đăng ký và Đăng nhập (Authentication)
 - Cung cấp quy trình xác thực an toàn, cho phép người dùng tạo tài khoản mới và truy cập hệ thống thông qua Email/Mật khẩu hoặc liên kết nhanh với tài khoản Google.
 - Hỗ trợ tính năng quên mật khẩu và khôi phục tài khoản qua email.
- Quản lý Hồ sơ cá nhân (Profile Management)
 - Cho phép người dùng cập nhật thông tin định danh (avatar, tên, ngày sinh, giới tính).
 - Thiết lập các tùy chọn cá nhân hóa như đơn vị tiền tệ mặc định.
 - Cài đặt Ngân sách hàng tháng (Monthly Budget) để nhận cảnh báo khi chi tiêu vượt mức.
- Bảng điều khiển tổng quan (Dashboard)
 - Trung tâm hiển thị thông tin, cung cấp cái nhìn toàn cảnh về sức khỏe tài chính thông qua các thẻ chỉ số KPI (Tổng thu, Tổng chi, Số dư thực tế).
 - Hiển thị Thanh trạng thái ngân sách giúp người dùng biết mình đã tiêu bao nhiêu phần trăm định mức.
 - Biểu đồ xu hướng tài chính trong 30 ngày gần nhất và danh sách các giao dịch mới thực hiện.
- Quản lý Thu nhập và Chi tiêu (Transaction Management)
 - Chức năng cốt lõi cho phép người dùng ghi chép chi tiết các dòng tiền vào và ra.
 - Hỗ trợ nhập liệu đầy đủ các trường thông tin: số tiền, ngày tháng, danh mục, ghi chú chi tiết (Note) và biểu tượng cảm xúc (emoji).
 - Cho phép chỉnh sửa hoặc xóa các giao dịch đã nhập.
- Quản lý Danh mục (Category Management)
 - Tăng tính linh hoạt cho hệ thống bằng cách cho phép người dùng tự tạo, chỉnh sửa hoặc xóa các danh mục chi tiêu riêng (như “Quỹ đen”, “Du lịch”).
 - Tùy chỉnh màu sắc và biểu tượng (Icon) cho từng danh mục để dễ dàng nhận diện.

- Báo cáo và Phân tích (Analytics)
 - Cung cấp các công cụ trực quan hóa dữ liệu mạnh mẽ.
 - Xem biểu đồ cột so sánh tổng thu/chi, biểu đồ tròn thể hiện cơ cấu chi tiêu (tiêu vào đâu nhiều nhất).
 - Bộ lọc dữ liệu linh hoạt theo khoảng thời gian (tuần, tháng, năm) hoặc theo loại giao dịch.
- Trợ lý ảo thông minh (AI Chatbot - FinBot)
 - Tích hợp trí tuệ nhân tạo (Google Gemini) để hỗ trợ người dùng nhập liệu nhanh bằng ngôn ngữ tự nhiên (Ví dụ: “Ăn sáng 50k với bạn” -> Tự động lưu và trích xuất ghi chú).
 - Tra cứu số dư, thống kê chi tiêu tức thì qua hội thoại.
 - Vẽ biểu đồ phân tích trực tiếp ngay trong khung chat khi được yêu cầu.
- Xuất dữ liệu (Data Export)
 - Tính năng tiện ích cho phép người dùng trích xuất toàn bộ lịch sử giao dịch thu nhập và chi tiêu ra file Excel (.xlsx) kèm theo ghi chú chi tiết để lưu trữ hoặc xử lý nâng cao.
- Bảo mật nâng cao (Security)
 - Trang bị lớp bảo vệ tài khoản với tính năng Xác thực 2 yếu tố (2FA) sử dụng mã OTP thời gian thực (TOTP).
 - Quản lý phiên đăng nhập, cho phép đăng xuất từ xa hoặc giới hạn thiết bị truy cập.
- Hướng dẫn người dùng (Onboarding Tour)
 - Tự động hướng dẫn các chức năng chính cho người dùng mới đăng nhập lần đầu, giúp họ làm quen nhanh chóng với hệ thống.

2. Nhóm chức năng dành cho Quản trị viên (Admin)

- Dashboard Quản trị (Admin Dashboard)
 - Cung cấp cái nhìn vĩ mô về hoạt động của toàn bộ hệ thống.

- Thông kê các chỉ số quan trọng: Tổng số người dùng, Tổng dòng tiền giao dịch toàn hệ thống, Số lượng người dùng mới trong 24h và Số lượng người dùng kích hoạt bảo mật 2FA.
 - Biểu đồ tăng trưởng thành viên mới theo thời gian.
- Quản lý Người dùng (User Management)
 - Công cụ giúp quản trị viên giám sát danh sách tài khoản, tìm kiếm thông tin người dùng theo email hoặc tên.
 - Thực hiện cấp quyền quản trị (Admin) hoặc xóa bỏ vĩnh viễn các tài khoản vi phạm khỏi hệ thống.
- Quản lý Danh mục hệ thống (Default Categories)
 - Cho phép thiết lập và duy trì bộ danh mục chuẩn (như Ăn uống, Di chuyển, Lương...) dùng chung cho toàn bộ người dùng mới tham gia hệ thống.
- Cấu hình hệ thống (System Settings)
 - Các chức năng vận hành quan trọng như bật/tắt Chế độ bảo trì (Maintenance Mode) để tạm ngưng hệ thống khi cần nâng cấp.
 - Kiểm soát việc Cho phép đăng ký mới.
 - Gửi Thông báo toàn cục (Broadcast) đến màn hình Dashboard của tất cả người dùng để thông báo tin tức quan trọng.
 - Giám sát trạng thái sức khỏe máy chủ (Server Health Check) và độ trễ kết nối (Ping).
- Nhật ký hoạt động (Audit Logs)
 - Hệ thống tự động ghi lại lịch sử các tác vụ quan trọng của quản trị viên (như xóa người dùng, sửa cấu hình, thêm danh mục).
 - Hiển thị chi tiết: Ai làm gì, vào lúc nào, đối tượng bị tác động là ai, giúp phục vụ công tác tra soát lỗi và đảm bảo tính minh bạch.

Hướng phát triển:

Trong tương lai, để hệ thống hoàn thiện hơn và có thể triển khai thực tế rộng rãi, em đề xuất các hướng phát triển sau:

Bên cạnh những kết quả đạt được, hệ thống vẫn còn một số hạn chế do giới hạn về thời gian và nguồn lực:

- Chưa hỗ trợ tính năng quét hóa đơn (OCR) để nhập liệu tự động từ ảnh chụp.
 - Chưa có ứng dụng di động (Mobile App) gốc (Native), hiện tại chỉ chạy trên trình duyệt web.
 - Chưa tích hợp liên kết ngân hàng (Open Banking) để đồng bộ giao dịch tự động.
1. **Tích hợp OCR:** Sử dụng các thư viện AI Vision để quét hóa đơn siêu thị và tự động điền vào form chi tiêu.
 2. **Phát triển Mobile App:** Xây dựng phiên bản App (sử dụng React Native) để tận dụng các tính năng như thông báo đẩy (Push Notification) nhắc nhở nhập liệu hàng ngày.
 3. **Nâng cấp FinBot:** Huấn luyện model chuyên sâu hơn để Bot có thể đưa ra lời khuyên tài chính cá nhân hóa (Ví dụ: “Bạn đang tiêu quá nhiều cho trà sữa, hãy giảm bớt để đạt mục tiêu mua xe”).
 4. **Triển khai (Deployment):** Đưa hệ thống lên các nền tảng Cloud (như AWS, Vercel, Render) để người dùng thực tế có thể truy cập.

DANH MỤC TÀI LIỆU THAM KHẢO

- [1] React Documentation. “React: The Library for Web and Native User Interfaces”. *react.dev*. [Online]. Available: <https://react.dev/>
- [2] Mozilla Developer Network (MDN). “Getting started with React”. *developer.mozilla.org*. [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Clientside_JavaScript_frameworks/React_getting_started
- [3] Codecademy. “React: The Virtual DOM”. *codecademy.com*. [Online]. Available: <https://www.codecademy.com/article/react-virtual-dom>
- [4] W3Schools. “React Introduction”. *w3schools.com*. [Online]. Available: https://www.w3schools.com/react/react_intro.asp
- [5] Python Software Foundation. “General Python FAQ”. *python.org*. [Online]. Available: <https://docs.python.org/3/faq/general.html>
- [6] LangChain AI. “LangChain Python Documentation”. *python.langchain.com*. [Online]. Available: https://python.langchain.com/docs/get_started/introduction
- [7] Sebastián Ramírez. “FastAPI Documentation”. *fastapi.tiangolo.com*. [Online]. Available: <https://fastapi.tiangolo.com/>
- [8] Encode OSS. “Starlette: The Little ASGI Framework That Shines”. *starlette.io*. [Online]. Available: <https://www.starlette.io/>
- [9] Pydantic. “Data validation using Python type hints”. *docs.pydantic.dev*. [Online]. Available: <https://docs.pydantic.dev/>
- [10] The PostgreSQL Global Development Group. “PostgreSQL: The World's Most Advanced Open Source Relational Database”. *postgresql.org*. [Online]. Available: <https://www.postgresql.org/about/>
- [11] GeeksforGeeks. “Introduction to PostgreSQL”. *geeksforgeeks.org*. [Online]. Available: <https://www.geeksforgeeks.org/introduction-to-postgresql/>

- [12] IBM. “ACID Properties of Transactions”. *ibm.com*. [Online]. Available: <https://www.ibm.com/docs/en/cics-ts/5.4?topic=processing-acid-properties-transactions>
- [13] Suzuki, H. “The Internals of PostgreSQL: Concurrency Control”. *interdb.jp*. [Online]. Available: <http://www.interdb.jp/pg/pgsql05.html>
- [14] Google Cloud. “What is Generative AI?”. *cloud.google.com*. [Online]. Available: <https://cloud.google.com/use-cases/generative-ai>
- [15] Google DeepMind. “Gemini: A family of highly capable multimodal models”. *deepmind.google*. [Online]. Available: <https://deepmind.google/technologies/gemini/>
- [16] NVIDIA. “What Is a Transformer Model?”. *nvidia.com*. [Online]. Available: <https://blogs.nvidia.com/blog/what-is-a-transformer-model/>
- [17] LangChain. “LangChain Introduction”. *python.langchain.com*. [Online]. Available: https://python.langchain.com/docs/get_started/introduction
- [18] Google Firebase. “Firebase Authentication Documentation”. *Firebase.google.com*. [Online]. Available: <https://firebase.google.com/docs/auth>
- [19] Auth0. “JSON Web Token (JWT) Introduction”. *jwt.io*. [Online]. Available: <https://jwt.io/introduction>