

KHOA KỸ THUẬT VÀ CÔNG NGHỆ
BỘ MÔN CÔNG NGHỆ THÔNG TIN



THỰC TẬP ĐỒ ÁN CHUYÊN NGÀNH
HỌC KỲ I, NĂM HỌC 2024-2025

XÂY DỰNG WEBSITE QUẢN LÝ CHI TIÊU CÁ NHÂN

Giảng viên hướng dẫn:
ThS. Lê Minh Tự

Sinh viên thực hiện:
Họ tên: Phạm Phước Vinh
MSSV: 110121144
Lớp: DA21TTC

Trà Vinh, tháng 01 năm 2025

KHOA KỸ THUẬT VÀ CÔNG NGHỆ
BỘ MÔN CÔNG NGHỆ THÔNG TIN



THỰC TẬP ĐỒ ÁN CHUYÊN NGÀNH
HỌC KỲ I, NĂM HỌC 2024-2025

XÂY DỰNG WEBSITE QUẢN LÝ CHI TIÊU CÁ NHÂN

Giảng viên hướng dẫn:
ThS. Lê Minh Tự

Sinh viên thực hiện:
Họ tên: Phạm Phước Vinh
MSSV: 110121144
Lớp: DA21TTC

Trà Vinh, tháng 01 năm 2025

Trà Vinh, ngày tháng năm
Giáo viên hướng dẫn
(Ký tên và ghi rõ họ tên)

[illegible]

Trà Vinh, ngày tháng năm
Thành viên hội đồng
(Ký tên và ghi rõ họ tên)

LỜI CẢM ƠN

Đề tài “Xây dựng website quản lý chi tiêu cá nhân” là nội dung em chọn để nghiên cứu và thực hiện đồ án chuyên ngành của bản thân sau ba năm theo học tại Trường Đại học Trà Vinh.

Để hoàn thành quá trình nghiên cứu và hoàn thiện đề tài này, lời đầu tiên em xin chân thành cảm ơn sâu sắc đến Thầy Lê Minh Tự thuộc Khoa Kỹ thuật & Công nghệ – Trường Đại học Trà Vinh Thầy đã trực tiếp chỉ bảo và hướng dẫn em trong suốt quá trình nghiên cứu để em hoàn thiện đề tài này. Ngoài ra em xin chân thành cảm ơn các Thầy, Cô trong Khoa Kỹ thuật & Công nghệ đã đóng góp những ý kiến quý báu cho đề tài.

Cuối cùng, em xin cảm ơn những người thân, bạn bè đã luôn bên em, động viên em trong quá trình thực hiện đề tài.

Trân trọng cảm ơn!

MỤC LỤC

LỜI CẢM ƠN.....	iii
MỤC LỤC	iv
DANH MỤC HÌNH ẢNH.....	vii
DANH MỤC BẢNG BIỂU	viii
TÓM TẮT ĐỒ ÁN CHUYÊN NGÀNH.....	ix
MỞ ĐẦU	1
CHƯƠNG 1: TỔNG QUAN.....	2
1.1 Đặt vấn đề	2
1.2 Mục đích nghiên cứu	2
CHƯƠNG 2: NGHIÊN CỨU LÝ THUYẾT.....	3
2.1 API và Restful API	3
2.1.1 Khái niệm	3
2.1.2 Lợi ích của API RESTful	3
2.1.3 Cách hoạt động.....	4
2.1 Tổng quan về Node.js	5
2.1.1 Giới thiệu Node.js	5
2.1.2 Cách hoạt động của Node.js.....	5
2.1.3 Ưu nhược điểm của Node.js.....	6
2.1.4 Ứng dụng của Node.js	7
2.2 Tổng quan về NestJs.....	8
2.2.1 Giới thiệu NestJs	8
2.2.2 Cấu trúc của NestJs	9
2.2.3 Ưu điểm của NestJs.....	9
2.2.4 Nhược điểm của NestJs.....	10
2.2.5 Cách cài đặt và tạo dự án NestJs	10

2.3	Tổng quan về React.js.....	12
2.3.1	Giới thiệu React.js	12
2.3.2	Ưu điểm của React.js.....	12
2.3.3	Các tính năng nổi bật của React.js	13
2.3.4	Cách sử dụng React.js trong dự án.....	14
2.4	Tổng quan về MongoDB	14
2.4.1	Sơ lược về NoSQL	14
2.4.2	Giới thiệu MongoDB.....	15
2.4.3	Một số khái niệm trong MongoDB	15
2.4.4	Ưu điểm của MongoDB	16
2.4.5	Nhược điểm của MongoDB	17
CHƯƠNG 3: HIỆN THỰC HÓA NGHIÊN CỨU		18
3.1	Mô tả bài toán	18
3.2	Yêu cầu chức năng.....	18
3.3	Yêu cầu phi chức năng.....	18
3.4	Cơ sở dữ liệu.....	19
3.4.1	Thiết kế cơ sở dữ liệu	19
3.4.2	Lược đồ use case	22
CHƯƠNG 4: KẾT QUẢ NGHIÊN CỨU		24
4.1	Giao diện trang chủ.....	24
4.2	Giao diện trang đăng nhập	24
4.3	Giao diện trang đăng ký.....	25
4.4	Trang tổng quan	25
4.5	Trang quản lí danh mục	26
4.6	Trang quản lí chi tiêu	26
4.7	Trang quản lí nguồn thu nhập.....	27

4.8	Trang quản lí thu nhập	27
4.9	Trang thông tin người dùng	28
CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN.....		29
5.1	Kết luận:.....	29
5.2	Hướng phát triển:	29
DANH MỤC TÀI LIỆU THAM KHẢO		30
PHỤ LỤC		31

DANH MỤC HÌNH ẢNH

Hình 2.1 Cách hoạt động của REST API	4
Hình 2.2 Cách hoạt động của Node.js	6
Hình 2.3 So sánh NoSql với SQL.....	15
Hình 2.4 Ví dụ về Collection trong MongoDB	16
Hình 2.5 Ví dụ minh họa cho Document trong MongoDB	16
Hình 3.1 UseCase tổng quát	22
Hình 3.2 UseCase đăng ký và đăng nhập	22
Hình 3.3 UseCase thống kê tài chính	22
Hình 3.4 UseCase quản lý chi tiêu và thu nhập	23
Hình 3.5 UseCase mục tiêu tiết kiệm	23
Hình 4.1 Hình ảnh trang chủ	24
Hình 4.2 Trang đăng nhập	24
Hình 4.3 Trang đăng ký	25
Hình 4.4 Trang tổng quan.....	25
Hình 4.5 Trang danh mục	26
Hình 4.6 Trang chi tiêu.....	26
Hình 4.7 Trang nguồn thu nhập.....	27
Hình 4.8 Trang thu nhập.....	27
Hình 4.9 Trang thông tin người dùng	28

DANH MỤC BẢNG BIỂU

TÓM TẮT ĐỒ ÁN CHUYÊN NGÀNH

Đề tài tập trung tìm hiểu và xây dựng website quản lý chi tiêu cá nhân với các chức năng quản lý doanh thu, chi tiêu, đưa ra các biểu đồ thống kê giúp người dùng có thể dễ dàng quản lý chi tiêu bản thân khi sử dụng website.

Phương pháp nghiên cứu:

- Nghiên cứu lý thuyết và tài liệu liên quan đến quản lý tài chính và công nghệ phát triển web.
- Sử dụng MongoDB để quản lý cơ sở dữ liệu.
- Dùng framework NestJS để xây dựng backend cho website.
- Sử dụng framework React.js để xây dựng frontend cho website.
- Hoàn thành và kiểm thử.

Kết quả đạt được: Hoàn thành website quản lý chi tiêu cá nhân với các tính năng hỗ trợ quản lý tài chính hiệu quả.

MỞ ĐẦU

1. Lí do chọn đề tài

Trong thời đại ngày nay, khi mọi người phải đối mặt với áp lực kinh tế từ nhiều phía như công việc, gia đình và các khoản chi tiêu hằng ngày, việc quản lý tài chính cá nhân trở thành một yếu tố quan trọng để duy trì sự ổn định về tài chính. Tuy nhiên, hầu hết mọi người không có quá nhiều thời gian hoặc các phương pháp hiệu quả để quản lý và theo dõi chi tiêu bản thân. Vì vậy, website quản lý tài chính cá nhân là thật sự cần thiết nó cung cấp cho người dùng nhiều công cụ để quản lý tài chính các nhân một cách đơn giản và hiệu quả mà ai cũng có thể sử dụng được.

2. Mục tiêu nghiên cứu

Đề tài được phát triển với mục đích cung cấp cho người dùng website để quản lý tài chính cá nhân bản thân một cách hiệu quả.

3. Đối tượng và phạm vi nghiên cứu

Đối tượng nghiên cứu: của đề tài là những người có nhu cầu về quản lý tài chính các nhân như học sinh, sinh viên, nhân viên văn phòng.

Phạm vi nghiên cứu:

- Nguyên cứu xây dựng các tính năng đăng nhập, ghi nhận giao dịch, thiết lập mục tiêu, xem báo cáo và biểu đồ.
- Đề tài sử dụng HTML, CSS, React để xây dựng giao diện người dùng, sử dụng NestJs xây dựng phần xử lí logic phí máy chủ và dùng MongoDB để quản lý cơ sở dữ liệu.

CHƯƠNG 1: TỔNG QUAN

1.1 Đặt vấn đề

Trong xã hội hiện đại, nhu cầu quản lý tài chính cá nhân ngày càng quan trọng. Việc quản lý và kiểm soát các khoản thu chi không chỉ giúp nắm rõ tình hình tài chính của bản thân mà còn hỗ trợ trong việc lập kế hoạch chi tiêu hợp lý. Tuy nhiên, không phải ai cũng có khả năng và thời gian làm việc đó một cách hiệu quả. Từ đó, việc xây dựng một website hỗ trợ quản lý chi tiêu cá nhân sẽ giúp người dùng dễ dàng ghi nhận, theo dõi, và phân tích thu chi của mình thông qua các biểu đồ và báo cáo trực quan, giúp cải thiện thói quen quản lý tài chính một cách khoa học hơn.

1.2 Mục đích nghiên cứu

Đề tài hướng đến mục đích là xây dựng một website quản lý chi tiêu cá nhân, cung cấp công cụ hỗ trợ người dùng trong việc quản lý tài chính cá nhân một cách thuận tiện và hiệu quả. Cụ thể là:

- Giúp người dùng dễ dàng ghi nhận các giao dịch thu nhập và chi tiêu hàng ngày.
- Cung cấp các biểu đồ phân tích và báo cáo tài chính trực quan, giúp người dùng có cái nhìn tổng quan về tình hình tài chính của mình.
- Hỗ trợ người dùng đặt ra mục tiêu tiết kiệm và theo dõi tiến độ hoàn thành mục tiêu tiết kiệm.

CHƯƠNG 2: NGHIÊN CỨU LÝ THUYẾT

2.1 API và Restful API

2.1.1 Khái niệm

Giao diện lập trình ứng dụng (API) xác định các quy tắc phải tuân theo để giao tiếp với các hệ thống phần mềm khác. Các nhà phát triển đưa ra hoặc tạo API để các ứng dụng khác có thể giao tiếp với ứng dụng của họ theo cách lập trình. Ví dụ: ứng dụng bảng chấm công đưa ra một API yêu cầu tên đầy đủ của nhân viên và phạm vi ngày. Khi nhận được thông tin này, bảng chấm công của nhân viên sẽ được xử lý nội bộ và trả về số giờ làm việc trong phạm vi ngày đó. Có thể coi API web như một cổng giữa client và tài nguyên trên web [1].

Chuyển trạng thái đại diện (REST) là một kiến trúc phần mềm quy định các điều kiện về cách thức hoạt động của API. REST ban đầu được tạo ra như một hướng dẫn để quản lý giao tiếp trên một mạng phức tạp như Internet. Có thể sử dụng kiến trúc dựa trên REST để hỗ trợ giao tiếp hiệu suất cao và đáng tin cậy trên quy mô lớn. Có thể dễ dàng triển khai và sửa đổi REST, mang lại khả năng hiển thị và tính di động đa nền tảng cho bất kỳ hệ thống API nào.

Các nhà phát triển API có thể thiết kế các API bằng cách sử dụng nhiều kiến trúc khác nhau. Các API tuân theo kiểu kiến trúc REST được gọi là API REST. Các dịch vụ web triển khai kiến trúc REST được gọi là dịch vụ web RESTful. Thuật ngữ API RESTful thường là chỉ các API web RESTful. Tuy nhiên, có thể sử dụng các thuật ngữ API REST và API RESTful thay thế cho nhau.

2.1.2 Lợi ích của API RESTful

API RESTful có những lợi ích sau:

Khả năng thay đổi quy mô: Các hệ thống triển khai API REST có thể thay đổi quy mô một cách hiệu quả vì REST tối ưu hóa các tương tác giữa client và máy chủ. Tình trạng phi trạng thái loại bỏ tải của máy chủ vì máy chủ không phải giữ lại thông tin yêu cầu của client trong quá khứ. Việc lưu bộ nhớ đệm được quản lý tốt sẽ loại bỏ một phần hoặc hoàn toàn một số tương tác giữa client và máy chủ. Tất cả các tính năng này hỗ trợ khả năng thay đổi quy mô mà không gây ra tắc nghẽn giao tiếp làm giảm hiệu suất.

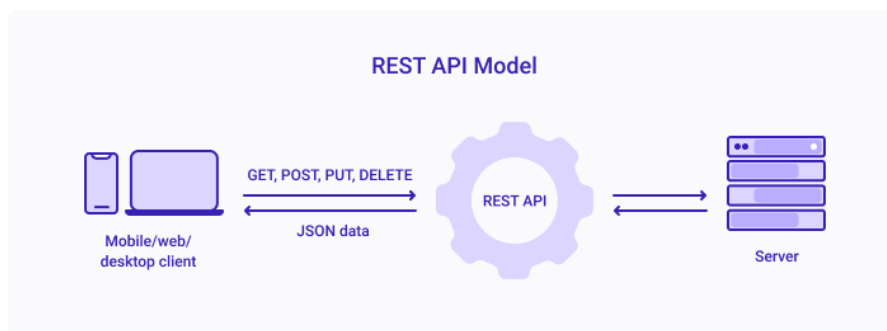
Sự linh hoạt: Các dịch vụ web RESTful hỗ trợ phân tách hoàn toàn giữa client và máy chủ. Các dịch vụ này đơn giản hóa và tách riêng các thành phần máy chủ khác nhau để mỗi phần có thể phát triển độc lập. Các thay đổi ở nền tảng hoặc công nghệ tại ứng dụng máy chủ không ảnh hưởng đến ứng dụng client. Khả năng phân lớp các chức năng ứng dụng làm tăng tính linh hoạt hơn nữa. Ví dụ: các nhà phát triển có thể thực hiện các thay đổi đối với lớp cơ sở dữ liệu mà không cần viết lại logic ứng dụng.

Sự độc lập: Các API REST không phụ thuộc vào công nghệ được sử dụng. Có thể viết cả ứng dụng client và máy chủ bằng nhiều ngôn ngữ lập trình khác nhau mà không ảnh hưởng đến thiết kế API. Đồng thời cũng có thể thay đổi công nghệ cơ sở ở hai phía mà không ảnh hưởng đến giao tiếp.

2.1.3 Cách hoạt động

Chức năng cơ bản của API RESTful cũng giống như việc duyệt Internet. Client liên hệ với máy chủ bằng cách sử dụng API khi yêu cầu tài nguyên. Các nhà phát triển API giải thích cách client nên sử dụng API REST trong tài liệu về API ứng dụng máy chủ. Đây là các bước chung cho bất kỳ lệnh gọi API REST nào:

- Client gửi một yêu cầu đến máy chủ. Client làm theo tài liệu API để định dạng yêu cầu theo cách mà máy chủ hiểu được.
- Máy chủ xác thực và xác nhận máy khách có quyền đưa ra yêu cầu đó.
- Máy chủ nhận yêu cầu và xử lý trong nội bộ.
- Máy chủ trả về một phản hồi đến client. Phản hồi chứa thông tin cho client biết liệu yêu cầu có thành công hay không. Phản hồi cũng bao gồm bất kỳ thông tin nào mà client yêu cầu.



Hình 2.1 Cách hoạt động của REST API

2.1 Tổng quan về Node.js

2.1.1 Giới thiệu Node.js

Node.js là một môi trường chạy (runtime environment) cho JavaScript với mã nguồn mở và đa nền tảng [2].

Node.js là mã nguồn mở: Mã nguồn của Node.js là công khai. Và nó được duy trì bởi những người đóng góp từ khắp nơi trên thế giới.

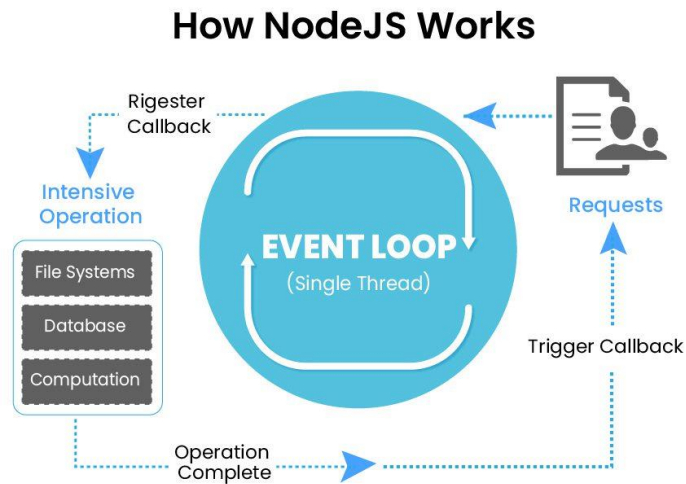
Node.js là môi trường thời gian chạy JavaScript: Nghĩa là khi viết mã JavaScript trong trình soạn thảo văn bản của mình, mã đó sẽ thực hiện tác vụ nào trừ khi thực thi nó. Và để chạy mã cần phải có một môi trường biên dịch.

Các trình duyệt như Chrome và Firefox có môi trường thời gian chạy. Đó là lý do tại sao các trình duyệt có thể chạy mã JavaScript. Trước khi Node.js được tạo ra thì JavaScript chỉ có thể được chạy trên trình duyệt. Và chỉ được sử dụng để xây dựng các ứng dụng front-end.

Node.js cung cấp một môi trường chạy JavaScript ngoài trình duyệt. Nó được xây dựng trên công cụ JavaScript Chrome V8. Điều này cho phép xây dựng các ứng dụng back-end bằng cách sử dụng cùng một ngôn ngữ lập trình JavaScript.

2.1.2 Cách hoạt động của Node.js

Node.js hoạt động dựa trên một số nguyên tắc cơ bản giúp nó hiệu quả trong việc xử lý các ứng dụng có nhiều hoạt động nhập/xuất (I/O) mà không bị chặn, đồng thời giảm đáng kể sự phức tạp trong quản lý các luồng thực thi [3].



Hình 2.2 Cách hoạt động của Node.js

2.1.3 Ưu nhược điểm của Node.js

Ưu điểm của Node.js:

- *Hiệu suất cao*: Node.js được xây dựng trên động cơ JavaScript V8 của Google Chrome, cho phép biên dịch mã JavaScript thành mã máy nhanh chóng. Nhờ đó, thời gian thực thi của Node.js rất nhanh, làm tăng hiệu suất của các ứng dụng.

- *Hệ sinh thái phong phú*: Với hơn 50,000 gói có sẵn trong Node Package Manager (NPM), các nhà phát triển có thể dễ dàng tìm và sử dụng các thư viện theo nhu cầu của họ mà không cần phải viết lại từ đầu, tiết kiệm đáng kể thời gian và công sức.

- *Xử lý bất đồng bộ và không chặn (Asynchronous and Non-blocking)*: Node.js hoạt động một cách bất đồng bộ và không chặn các hoạt động I/O, nghĩa là nó không cần chờ đợi API trả về dữ liệu trước khi tiếp tục xử lý yêu cầu tiếp theo. Điều này làm cho Node.js trở nên lý tưởng cho việc xây dựng các ứng dụng web thời gian thực và xử lý dữ liệu lớn.

- *Tính nhất quán trong mã nguồn*: Node.js cho phép sử dụng cùng một ngôn ngữ lập trình (JavaScript) cho cả phía máy chủ và máy khách. Điều này không chỉ giúp giảm thiểu sự không đồng bộ giữa client và server mà còn làm cho việc bảo trì và quản lý mã nguồn trở nên dễ dàng hơn.

- *Khả năng mở rộng*: Node.js hỗ trợ xây dựng các ứng dụng có khả năng mở rộng cao thông qua mô hình sự kiện và bất đồng bộ của mình. Điều này cho phép xử lý hàng ngàn kết nối đồng thời mà không làm giảm hiệu suất.

- *Ngôn ngữ quen thuộc*: Vì Node.js là một khung làm việc JavaScript, nó trở thành lựa chọn lý tưởng cho những nhà phát triển đã quen thuộc với JavaScript. Điều này làm cho quá trình học tập và phát triển dự án với Node.js trở nên dễ dàng hơn nhiều.

Nhược điểm:

- Đầu tiên phải kể đến chính là nó không có khả năng mở rộng. Do đó mà nó không thể tận dụng được lợi thế mô hình đa lõi ở các phần cứng cấp server trên thị trường hiện nay.

- Khó thao tác được với cơ sở dữ liệu quan hệ.
- Mỗi callback của nó sẽ đi kèm với nhiều callback lồng nhau khác.
- Cần trang bị kiến thức tốt về JavaScript.
- Không phù hợp với những tác vụ đòi hỏi nhiều CPU.

2.1.4 Ứng dụng của Node.js

Node.js được sử dụng rộng rãi trong nhiều loại ứng dụng web và server do khả năng xử lý bất đồng bộ, hiệu suất cao và hệ sinh thái phong phú của nó. Dưới đây là một số ứng dụng phổ biến của Node.js:

Ứng dụng Web Thời Gian Thực (Real-time Web Applications): Node.js là lựa chọn lý tưởng cho các ứng dụng web thời gian thực như trò chuyện trực tuyến và trò chơi trực tuyến do khả năng xử lý các sự kiện I/O một cách nhanh chóng và hiệu quả.

APIs Server-side: Node.js thường được sử dụng để xây dựng RESTful APIs do khả năng xử lý đồng thời lớn và tốc độ phản hồi nhanh, làm cơ sở cho các ứng dụng di động và web.

Streaming Data: Node.js hỗ trợ xử lý dữ liệu dạng stream, cho phép ứng dụng xử lý các tệp video, âm thanh hoặc các dữ liệu khác trong khi chúng vẫn đang được truyền, thay vì phải chờ cho đến khi toàn bộ tệp được tải về.

Ứng dụng Một Trang (Single Page Applications): Node.js phù hợp với việc phát triển các ứng dụng một trang (SPA) như Gmail, Google Maps, hay Facebook, nơi mà nhiều tương tác xảy ra trên một trang duy nhất mà không cần tải lại trang.

Công cụ và Tự Động Hóa: Node.js cũng được sử dụng để phát triển các công cụ dòng lệnh và các script tự động hóa quy trình làm việc, nhờ vào các gói NPM hỗ trợ đa dạng và khả năng tích hợp dễ dàng với các công nghệ khác.

Microservices Architecture: Node.js là một lựa chọn phổ biến cho kiến trúc microservices, nơi các ứng dụng lớn được chia thành các dịch vụ nhỏ, độc lập và dễ quản lý hơn.

Ứng dụng IoT (Internet of Things): Node.js thường được sử dụng trong các ứng dụng IoT, nơi cần xử lý một lượng lớn các kết nối đồng thời và các sự kiện từ các thiết bị IoT.

Dashboard và Monitoring: Node.js được sử dụng để xây dựng các dashboard hiển thị dữ liệu thời gian thực và các công cụ giám sát, giúp doanh nghiệp dễ dàng theo dõi hiệu suất và tình trạng của các hệ thống.

2.2 Tổng quan về NestJs

2.2.1 Giới thiệu NestJs

Nest (NestJS) là một framework để xây dựng các ứng dụng phía máy chủ hiệu quả, chạy trên Node.js. Được xây dựng và hỗ trợ TypeScript (nhưng vẫn cho phép các nhà phát triển viết mã bằng JavaScript thuần túy) và kết hợp các yếu tố của OOP (Lập trình hướng đối tượng), FP (Lập trình chức năng) và FRP (Lập trình phản ứng chức năng).

Nest sử dụng các khung HTTP Server mạnh mẽ như Express (mặc định) và tùy chọn cũng có thể được định cấu hình để sử dụng Fastify.

Nest cung cấp một mức độ trừu tượng trên các khung Node.js phổ biến này (Express / Fastify), nhưng cũng hiển thị API của chúng trực tiếp cho nhà phát triển. Điều này cho phép các nhà phát triển tự do sử dụng vô số mô-đun của bên thứ ba có sẵn cho nền tảng cơ bản [4].

2.2.2 Cấu trúc của NestJs

NestJS được xây dựng từ nhiều yếu tố khác nhau, tuy nhiên quan trọng nhất là ba thành phần chính sau:

Modules: được sử dụng để tổ chức code và chia các tính năng thành các đơn vị có thể tái sử dụng hợp lý. Các tệp TypeScript được nhóm bằng Decorator “@Module”. Decorator “@Module” cung cấp siêu dữ liệu cho NestJS giúp nó xác định các thành phần, bộ điều khiển hay những tài nguyên khác để sắp xếp cấu trúc ứng dụng khoa học, hiệu quả.

Providers: Một thành phần không thể thiếu ở NestJS đó là Providers. Nó tương tự như một dịch vụ giúp xử lý những tác vụ mang tính phức tạp, logic mà các trình xử lý Controller không thể làm được. Providers có thể được tạo và đưa vào Controllers hoặc Providers khác.

Controllers: Chức năng chính của Controllers là xử lý các yêu cầu gửi đến và đáp trả lại cho client-side. Sau khi nhận được yêu cầu HTTP nó sẽ soạn thảo câu trả lời chính xác, phù hợp nhất để gửi đi. Mỗi Controllers sẽ có bộ lộ trình riêng để giúp nó thực hiện tốt các tác vụ khác nhau.

2.2.3 Ưu điểm của NestJs

Đối với một lập trình viên Nodejs, việc lựa chọn đúng framework cho project rất quan trọng. Trong khi Expressjs được sử dụng để xây dựng ứng dụng web trong nhiều năm thì NestJS đã xuất hiện và mang lại các lợi ích và tính năng nổi trội hơn:

Ngôn ngữ mạnh về kiểu dữ liệu: NestJS được xây dựng trên nền TypeScript, một phiên bản hỗ trợ kiểu dữ liệu của JavaScript. Sử dụng Typescript khi viết code giúp code sạch hơn và dễ xử lý khi có lỗi.

Kiến trúc modular: NestJS được xây dựng trên kiến trúc modular giúp dễ tổ chức và phát triển mã nguồn trên quy mô lớn hơn. Kiến trúc này bao gồm các khối tách biệt nhau như controller, provider, và module, giúp cho việc quản lý dễ dàng hơn. Mình sẽ nói kĩ hơn về các thành phần này ở phần sau của bài viết nhé.

Dependency Injection: NestJS sử dụng Dependency Injection (DI) để quản lý luồng của các phụ thuộc giữa các module và component. Nó cho phép ứng dụng dễ

kiểm thử và bảo trì hơn, do sự độc lập giữa các component với nhau. Cụ thể độc lập như thế nào thì cùng đợi đến phần sau nhé.

Built-In Validation: NestJS được hỗ trợ sẵn tính năng validate dữ liệu đầu vào - thứ đặc biệt quan trọng khi xây dựng các API. Framework này giúp định nghĩa và thực thi các validation rule một cách dễ dàng hơn, giảm đi các sai sót và lỗi không đáng có.

Hỗ trợ các tính năng nâng cao: NestJS hỗ trợ hàng loạt các tính năng được tích hợp sẵn, ví dụ như WebSocket, GraphQL, và Microservices. Điều này giúp đơn giản hóa việc xây dựng các ứng dụng yêu cầu tính năng realtime hay kiến trúc microservices.

Cộng đồng hỗ trợ mạnh mẽ: NestJS có một cộng đồng các lập trình viên đóng góp cho framework thường xuyên. Điều này sẽ giúp NestJS luôn được cập nhật và sửa lỗi, trở thành 1 framework đáng tin cậy.

2.2.4 Nhược điểm của NestJs

Việc hỗ trợ nhiều tính năng mạnh mẽ không có nghĩa là NestJS không có bất kỳ điểm yếu nào.

Sự phụ thuộc vòng lặp (circular dependencies): Đây là một vấn đề phổ biến mà hầu hết các dự án NestJS sẽ gặp phải. Vấn đề này rất rắc rối và nó có thể làm chậm đi quá trình phát triển của phần mềm. May mắn là, một bài báo gần đây của Trilon đã đưa ra một công cụ tên là Madge, giúp xác định sớm circular dependencies.

Logs bị ẩn khi ứng dụng khởi động: Đây cũng là một vấn đề khá phổ biến. Điều này có thể khiến các lập trình viên khó debug khi có lỗi xảy ra. Để giải quyết thì họ thường vô hiệu hóa việc dừng ứng dụng khi gặp lỗi và gửi lại thông báo lỗi.

Khó khăn trong kiểm thử đơn vị (unit test): Unit test được tích hợp sâu vào framework. Việc kiểm thử ở mức đơn vị nhỏ thường gây ra nhiều boilerplate code. Hơn nữa, việc viết test có thể gây khó khăn với một số lập trình viên, vì họ phải hiểu được cơ chế hoạt động của dependency injection tree trong NestJS.

2.2.5 Cách cài đặt và tạo dự án NestJs

Để cài đặt NestJS, cần phải cài đặt Node.js và npm (Node Package Manager) trước tiên. Sau đó, có thể sử dụng npm để cài đặt NestJS CLI (Command Line

Interface) và tạo một dự án NestJS mới. Dưới đây là hướng dẫn cài đặt NestJS trên hệ điều hành Windows, macOS và Linux:

Bước 1: Cài đặt Node.js và npm

Trước tiên, hãy truy cập trang chủ của Node.js (<https://nodejs.org/>) để tải về phiên bản mới nhất và cài đặt Node.js và npm theo hướng dẫn trên trang web.

Bước 2: Cài đặt NestJS CLI

Sau khi cài đặt Node.js và npm thành công, mở cửa sổ dòng lệnh hoặc terminal trên máy tính của bạn và chạy lệnh sau để cài đặt NestJS CLI một cách toàn cầu (global):

```
``` npm install -g @nestjs/cli ```
```

### **Bước 3: Tạo một dự án NestJS mới**

Sau khi cài đặt NestJS CLI, có thể sử dụng nó để tạo một dự án NestJS mới. Để tạo một dự án mới, hãy chạy lệnh sau trong cửa sổ dòng lệnh hoặc terminal:

```
``` nest new project-name ```
```

Trong đó, 'project-name' là tên của dự án muốn tạo. NestJS CLI sẽ tạo ra một cấu trúc dự án NestJS cơ bản với các tập tin và thư mục cần thiết.

Bước 4: Chạy ứng dụng NestJS

Sau khi dự án NestJS đã được tạo thành công, có thể di chuyển vào thư mục dự án bằng cách chạy lệnh:

```
``` cd project-name ```
```

Sau đó, có thể chạy ứng dụng NestJS bằng lệnh sau:

```
``` npm run start ```
```

Hoặc nếu muốn tự động khởi động lại ứng dụng khi có thay đổi trong mã nguồn, có thể sử dụng lệnh:

```
``` npm run start:dev ```
```

Sau khi ứng dụng đã chạy thành công, có thể truy cập vào <http://localhost:3000> (hoặc cổng khác nếu đã thay đổi cài đặt) để xem ứng dụng NestJS.

## 2.3 Tổng quan về React.js

### 2.3.1 Giới thiệu React.js

React.js được hiểu là một thư viện, mà trong đó sẽ chứa nhiều JavaScript mã nguồn mở và cha đẻ của React.js đó chính là Facebook. React.js được sử dụng làm cơ sở để phát triển các ứng dụng SPA (Single page), ứng dụng, thiết bị di động.

Bên cạnh đó, React.js hướng tới việc quản lý và hiển thị trạng thái do DOM. Vì vậy, việc xây dựng ứng dụng bằng React.js thường được yêu cầu dùng thêm các thư viện bổ sung để thực hiện định tuyến trang [5].

### 2.3.2 Ưu điểm của React.js

*Sử dụng dễ dàng:* React.js là một thư viện GUI nguồn mở JavaScript đặc biệt hữu ích trong việc xây dựng giao diện người dùng (UI). Nằm trong phần “View” của mô hình MVC (Model-View-Controller), React.js làm cho việc phát triển UI trở nên hiệu quả và trực quan.

*Hỗ trợ Component tái sử dụng trong Java:* React.js cung cấp khả năng tái sử dụng các components đã phát triển, đây là một tính năng quan trọng giúp tiết kiệm thời gian và công sức cho lập trình viên. Các components này có thể dễ dàng được sử dụng lại trong các ứng dụng khác có cùng chức năng, đồng thời mang lại lợi ích đáng kể trong quá trình phát triển phần mềm.

*Việc viết Component trở nên dễ dàng hơn:* React.js thường sử dụng JSX – một phần mở rộng cú pháp cho JavaScript, giúp việc viết component trở nên dễ dàng hơn. JSX kết hợp giữa JavaScript và HTML tạo ra một cấu trúc trực quan và dễ hiểu. Điều này không chỉ làm cho quá trình viết code trở nên rõ ràng hơn mà còn giúp dễ dàng render các components hơn.

Mặc dù đây không phải là phần mở rộng cú pháp phổ biến nhất, nhưng JSX vẫn được đánh giá cao trong việc phát triển các components phức tạp và ứng dụng trong phạm vi quy mô lớn.

*Hiệu suất cao:* React.js cải thiện đáng kể quá trình làm việc với DOM (Document Object Model). Trong phát triển web, quản lý DOM thường gặp nhiều khó khăn và gây thất vọng. React.js thường sử dụng virtual DOMs để tránh được những vấn đề này. Sự thay đổi trong DOM thực tế sẽ được phản ánh nhanh chóng

trong virtual DOM, đồng thời giúp tăng tốc độ và hiệu suất ứng dụng mà không làm gián đoạn quá trình cập nhật.

*Thân thiện với SEO:* Một ưu điểm lớn của React.js là khả năng tạo ra các giao diện người dùng tối ưu cho SEO. Điều này rất quan trọng vì không phải tất cả các framework JavaScript đều thân thiện với SEO. Bên cạnh đó, React.js cũng hỗ trợ tăng tốc độ xử lý của ứng dụng, đây cũng là một yếu tố quan trọng trong việc cải thiện kết quả SEO.

Tuy nhiên, cần lưu ý rằng React.js vẫn chỉ là một thư viện JavaScript và có thể cần sử dụng các thư viện bổ sung cho các nhiệm vụ như quản lý, định tuyến và tương tác.

### **2.3.3 Các tính năng nổi bật của React.js**

React.js có nhiều tính năng hữu ích cho việc phát triển ứng dụng web, bao gồm:

**Components:** React.js cho phép phát triển ứng dụng web theo mô hình component. Các component là các phần tử UI độc lập có thể được tái sử dụng trong nhiều phần khác nhau của ứng dụng.

**Virtual DOM:** React.js sử dụng Virtual DOM để tối ưu hóa hiệu suất của ứng dụng. Virtual DOM là một bản sao của DOM được lưu trữ trong bộ nhớ và được cập nhật một cách nhanh chóng khi có thay đổi, giúp tăng tốc độ và hiệu suất của ứng dụng.

**JSX:** JSX là một ngôn ngữ lập trình phân biệt được sử dụng trong React.js để mô tả các thành phần UI. JSX kết hợp HTML và JavaScript, giúp cho việc viết mã dễ hiểu và dễ bảo trì hơn.

**State và Props:** React.js cho phép quản lý trạng thái của các thành phần UI thông qua State và Props. State là trạng thái của một thành phần được quản lý bởi nó chính, trong khi Props là các giá trị được truyền vào từ bên ngoài để tùy chỉnh hoặc điều khiển hành vi của một thành phần.

**Hỗ trợ tốt cho SEO:** React.js hỗ trợ tốt cho việc tối ưu hóa SEO. Với các thư viện như React Helmet, các nhà phát triển có thể quản lý các phần tử meta và title cho từng trang web, giúp tăng khả năng tìm kiếm và tăng cường trải nghiệm người dùng.



Hỗ trợ đa nền tảng: React.js không chỉ được sử dụng để phát triển ứng dụng web, mà còn được sử dụng để phát triển ứng dụng di động với React Native. Sử dụng React Native, các nhà phát triển có thể xây dựng ứng dụng di động cho cả iOS và Android sử dụng cùng một mã nguồn.

Redux: Redux là một thư viện quản lý trạng thái cho các ứng dụng React.js. Nó giúp quản lý trạng thái của ứng dụng một cách chính xác và dễ dàng, đồng thời giúp tăng tính linh hoạt và khả năng mở rộng của ứng dụng.

### **2.3.4 Cách sử dụng React.js trong dự án**

*Bước 1. Cài đặt Node.js và npm:* React.js được xây dựng trên nền tảng Node.js, do đó cần cài đặt Node.js và npm để phát triển ứng dụng React.js.

*Bước 2. Tạo một ứng dụng React.js:* Có thể tạo một ứng dụng React.js bằng cách sử dụng lệnh "create-react-app" trong Command Prompt hoặc Terminal.

*Bước 3. Tạo các component:* Tạo các component để xây dựng giao diện người dùng cho ứng dụng. Có thể tạo component bằng cách sử dụng class hoặc hàm.

*Bước 4. Xây dựng giao diện người dùng:* Sử dụng JSX để xây dựng giao diện người dùng cho ứng dụng. JSX là một ngôn ngữ phân biệt được sử dụng trong React.js để mô tả các thành phần UI.

*Bước 5. Quản lý trạng thái:* Sử dụng State và Props để quản lý trạng thái của các thành phần UI. State là trạng thái của một thành phần được quản lý bởi nó chính, trong khi Props là các giá trị được truyền vào từ bên ngoài để tùy chỉnh hoặc điều khiển hành vi của một thành phần.

*Bước 6. Kết nối với API:* Sử dụng thư viện như Axios để kết nối với API và lấy dữ liệu từ server.

*Bước 7 - Build và triển khai ứng dụng:* Sử dụng lệnh "npm run build" để build ứng dụng và triển khai nó trên môi trường sản phẩm.

## **2.4 Tổng quan về MongoDB**

### **2.4.1 Sơ lược về NoSQL**

Cơ sở dữ liệu NoSQL chuyên dành cho các mô hình dữ liệu cụ thể và lưu trữ dữ liệu trong các sơ đồ linh hoạt dễ dàng điều chỉnh quy mô cho các ứng dụng hiện đại.

Cơ sở dữ liệu NoSQL được công nhận rộng rãi vì khả năng dễ phát triển, chức năng cũng như hiệu năng ở quy mô lớn [6].



Hình 2.3 So sánh NoSql với SQL

## 2.4.2 Giới thiệu MongoDB

MongoDB là phần mềm cơ sở dữ liệu mã nguồn mở NoSQL hỗ trợ đa nền tảng được thiết kế theo hướng đối tượng. Các bảng (trong MongoDB gọi là collection) có cấu trúc linh hoạt cho phép dữ liệu không cần tuân theo dạng cấu trúc nào.

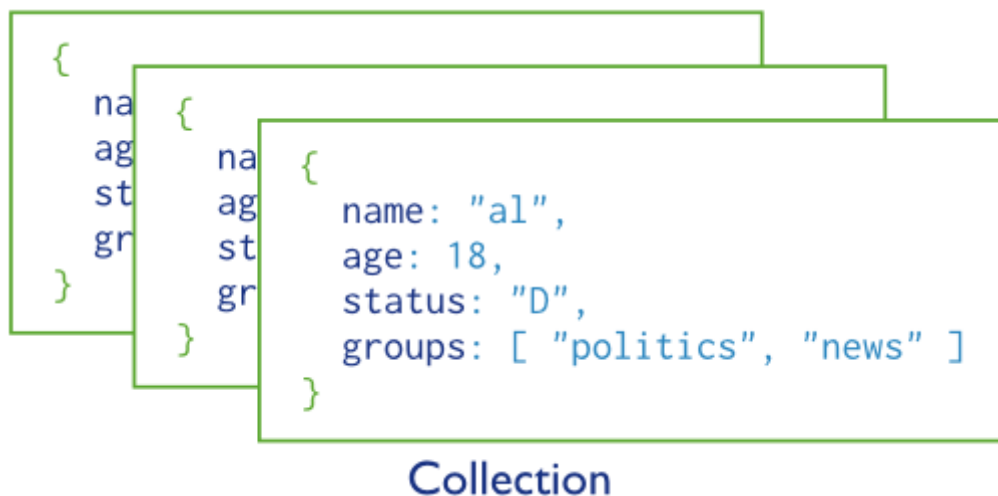
Vì thế, nó có thể dùng để lưu trữ dữ liệu có cấu trúc phức tạp và đa dạng. Dữ liệu được gọi là Big Data. Đặc biệt, chương trình này lưu trữ dữ liệu vào collection theo hướng tài liệu kiểu JSON thay vì bảng nên có hiệu suất cao và tính khả dụng cao [7].

## 2.4.3 Một số khái niệm trong MongoDB

*Database* là một container vật lý cho các collection. Mỗi DB được thiết lập cho riêng nó một danh sách các files hệ thống files. Một máy chủ MongoDB đơn thường có nhiều DB.

*Collection* là một nhóm các documents của MongoDB. Nó tương đương với một table trong RDBMS. Một Collection tồn tại trong một cơ sở dữ liệu duy nhất. Các collection ko tạo nên một schema. Documents trong collection có thể có các

fields khác nhau. Thông thường, tất cả các documents trong collections có mục đích khá giống nhau hoặc liên quan tới nhau.



Hình 2.4 Ví dụ về Collection trong MongoDB

*Document* là một tập hợp các cặp key-value. Documents có schema động. Schema động có nghĩa là documents trong cùng một collection không cần phải có cùng một nhóm các fields hay cấu trúc giống nhau, và các fields phổ biến trong các documents của collection có thể chứa các loại dữ liệu khác nhau.

```
{
 name: "sue",
 age: 26,
 status: "A",
 groups: ["news", "sports"]
}
```

← field: value  
← field: value  
← field: value  
← field: value

Hình 2.5 Ví dụ minh họa cho Document trong MongoDB

#### 2.4.4 Ưu điểm của MongoDB

Đầu tiên có thể nhắc đến là tính linh hoạt lưu trữ dữ liệu theo các kích cỡ khác nhau, dữ liệu dưới dạng hướng tài liệu JSON nên có thể chèn vào thoải mái bất cứ thông tin mong muốn.

Khác với RDBMS, dữ liệu trong đây không có sự ràng buộc và không có yêu cầu tuân theo khuôn khổ nhất định, điều này giúp tiết kiệm thời gian cho việc kiểm tra

sự thỏa mãn về cấu trúc nếu muốn chèn, xóa, cập nhật hay thay đổi các dữ liệu trong bảng.

MongoDB dễ dàng mở rộng hệ thống bằng cách thêm node vào cluster – cụm các node chứa dữ liệu giao tiếp với nhau.

Ưu điểm thứ tư là tốc độ truy vấn nhanh hơn nhiều so với hệ quản trị cơ sở dữ liệu quan hệ RDBMS do dữ liệu truy vấn được cached lên bộ nhớ RAM để lượt truy vấn sau diễn ra nhanh hơn mà không cần đọc từ ổ cứng.

Cũng là một ưu điểm về hiệu suất truy vấn của MongoDB, trường dữ liệu “\_id” luôn được tự động đánh chỉ mục để đạt hiệu suất cao nhất.

#### **2.4.5 Nhược điểm của MongoDB**

Dữ liệu trong MongoDB không bị ràng buộc như RDBMS nhưng người sử dụng lưu ý cẩn thận mọi thao tác để không xảy ra các kết quả ngoài ý muốn gây ảnh hưởng đến dữ liệu.

Một nhược điểm mà “dân công nghệ” hay lo ngại là bộ nhớ của thiết bị. Chương trình này thường tốn bộ nhớ do dữ liệu được lưu dưới dạng key-value, trong khi các collection chỉ khác về value nên sẽ lặp lại key dẫn đến thừa dữ liệu.

Thông thường, dữ liệu thay đổi từ RAM xuống ổ cứng phải qua 60 giây thì chương trình mới thực hiện hoàn tất, đây là nguy cơ bị mất dữ liệu nếu bất ngờ xảy ra tình huống mất điện trong vòng 60 giây đó.

## CHƯƠNG 3: HIỆN THỰC HÓA NGHIÊN CỨU

### 3.1 Mô tả bài toán

Bài toán đặt ra là xây dựng một website quản lý chi tiêu cá nhân, cung cấp cho người dùng một nền tảng hỗ trợ người dùng theo dõi, phân tích và tối ưu hóa tình hình tài chính của mình hiệu quả hơn.

Hệ thống sẽ cho phép người dùng nhập các khoản thu nhập và chi tiêu hàng ngày, phân loại chúng theo danh mục như thực phẩm, giải trí, giáo dục, hay hóa đơn. Người dùng có thể dễ dàng xem lại các giao dịch đã ghi nhận, sửa đổi hoặc xóa nếu cần thiết. Người dùng có thể đặt ra mục tiêu tiết kiệm cho bản thân hàng tháng. Ngoài ra, hệ thống cần cung cấp các công cụ phân tích trực quan như biểu đồ cột để theo dõi dòng tiền hàng tháng và biểu đồ tròn để thấy rõ sự phân bổ chi tiêu theo danh mục.

### 3.2 Yêu cầu chức năng

Quản lý người dùng: Cung cấp cho người dùng các chức năng đăng ký, đăng nhập và đăng xuất khỏi hệ thống.

Ghi nhận và quản lý giao dịch: Cho phép người dùng thêm các khoản giao dịch. Hiện thị lịch sử giao dịch.

Quản lý danh mục: Có thể thêm sửa xóa các danh mục thu chi của bản thân.

Thống kê tài chính: Sử dụng dữ liệu thu chi mà người dùng thêm vào để thực hiện thống kê, vẽ biểu đồ, đưa ra cảnh báo cho người dùng nếu chi tiêu vượt ngưỡng.

Đặt mục tiêu tiết kiệm: Người dùng có thể đặt ra mục tiêu tiết kiệm hàng tháng của bản thân.

### 3.3 Yêu cầu phi chức năng

Hiệu năng: Hệ thống đảm bảo hiệu năng ổn định.

Bảo mật: Sử dụng JWT để xác thực người dùng an toàn và hiệu quả.

Bảo trì và nâng cấp:

- Backend: Áp dụng kiến trúc module hóa chia nhỏ các tính năng, giúp dễ bảo trì và mở rộng.

- Frontend: Component hóa giao diện, chia nhỏ các thành phần giao diện.

### 3.4 Cơ sở dữ liệu

#### 3.4.1 Thiết kế cơ sở dữ liệu

Cơ sở dữ liệu được chia làm

##### 1. Collection “users”: Lưu trữ thông tin người dùng

```
"users": {
 "_id": "ObjectId()",
 "name": "String",
 "username": "String",
 "password": "String",
 "balance": "Number",
 "role": "String",
 "createdAt": "ISODate",
 "updatedAt": "ISODate"
},
```

##### 2. Collection “categories”: Danh mục chi tiêu (ăn uống, đi lại, ...)

```
"categories": {
 "_id": "ObjectId()",
 "name": "String",
 "description": "String",
 "userId": "ObjectId()",
 "createdAt": "ISODate",
 "updatedAt": "ISODate"
},
```

##### 3. Collection “expenses”: Quản lý thông tin các khoản chi

```
"expenses": {
 "_id": "ObjectId()",
```

```
"userId": "ObjectId()",
"categoryId": "ObjectId()",
"amount": "Number",
"description": "String",
"date": "ISODate",
"createdAt": "ISODate",
"updatedAt": "ISODate"
},
```

**4. Collection “sources”:** Nguồn thu nhập (Lương, làm thêm ...)

```
"sources": {
 "_id": "ObjectId()",
 "name": "String",
 "description": "String",
 "userId": "ObjectId()",
 "createdAt": "ISODate",
 "updatedAt": "ISODate"
},
```

**5. Collection “income”:** Quản lý các khoản thu nhập

```
"income": {
 "_id": "ObjectId()",
 "userId": "ObjectId()",
 "sourceId": "ObjectId()",
 "amount": "Number",
 "description": "String",
 "date": "ISODate",
 "createdAt": "ISODate",
```

```
"updatedAt": "ISODate"
```

```
},
```

## 6. Collection “savings”: Quản lý khoản tiết kiệm

```
"savings": {
```

```
 "_id": "ObjectId()",
```

```
 "userId": "ObjectId()",
```

```
 "targetAmount": "Number",
```

```
 "currentAmount": "Number",
```

```
 "monthlyTargetAmount": "Number",
```

```
 "startDate": "ISODate",
```

```
 "endDate": "ISODate",
```

```
 "createdAt": "ISODate",
```

```
 "updatedAt": "ISODate"
```

```
}
```

```
}
```

### Quan hệ giữa các collection:

users -> categories: Mỗi người dùng có thể có nhiều danh mục (1-n).

users -> incomes: Mỗi người dùng có thể có nhiều khoản thu nhập (1-n).

users -> expenses: Mỗi người dùng có thể có nhiều khoản chi tiêu (1-n).

users -> savings: Mỗi người dùng có thể có nhiều khoản tiết kiệm(1-n).

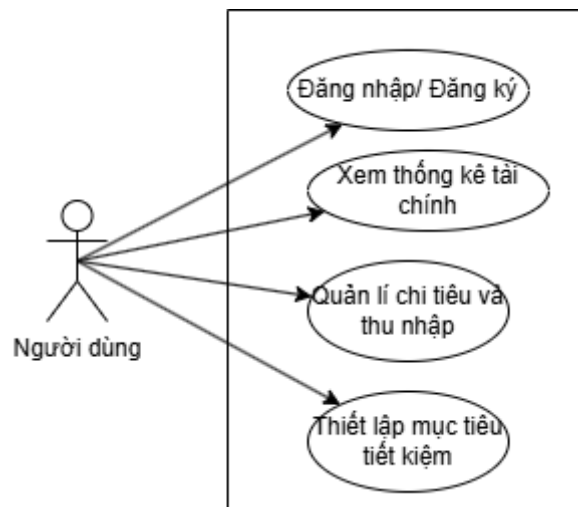
categories -> expenses: Một danh mục thì có thể nhiều khoản chi (1-n).

sources -> incomes: Mỗi nguồn thu nhập có thể có nhiều khoản thu nhập (1-n).



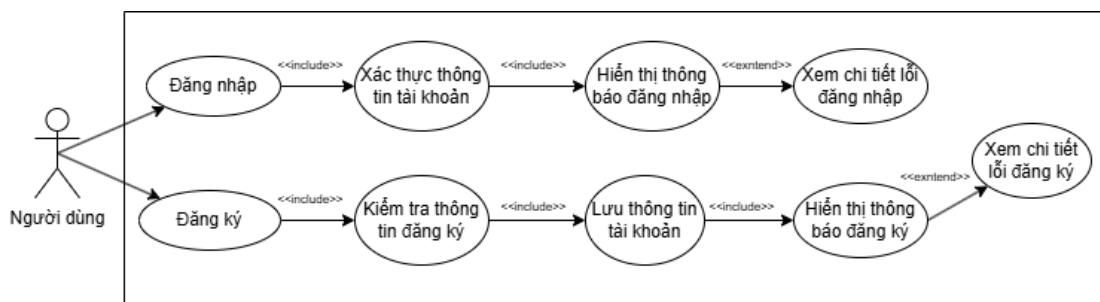
### 3.4.2 Lược đồ use case

UseCase tổng quát:



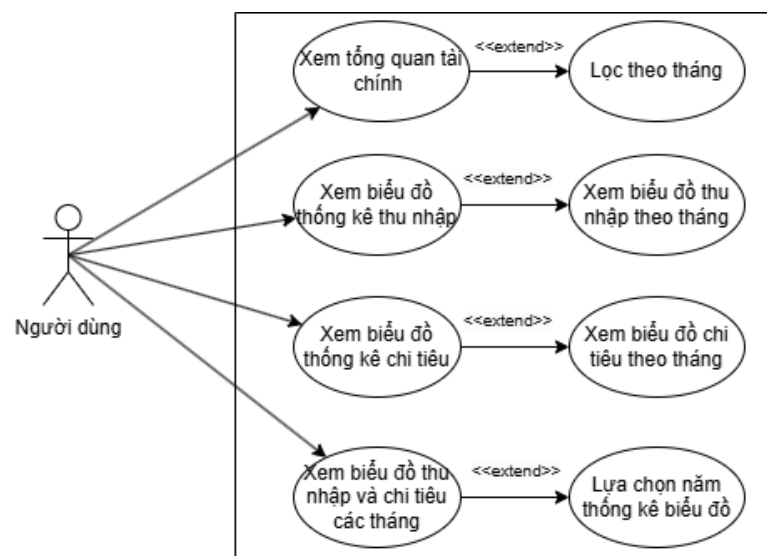
Hình 3.1 UseCase tổng quát

UseCase Đăng ký / Đăng nhập:



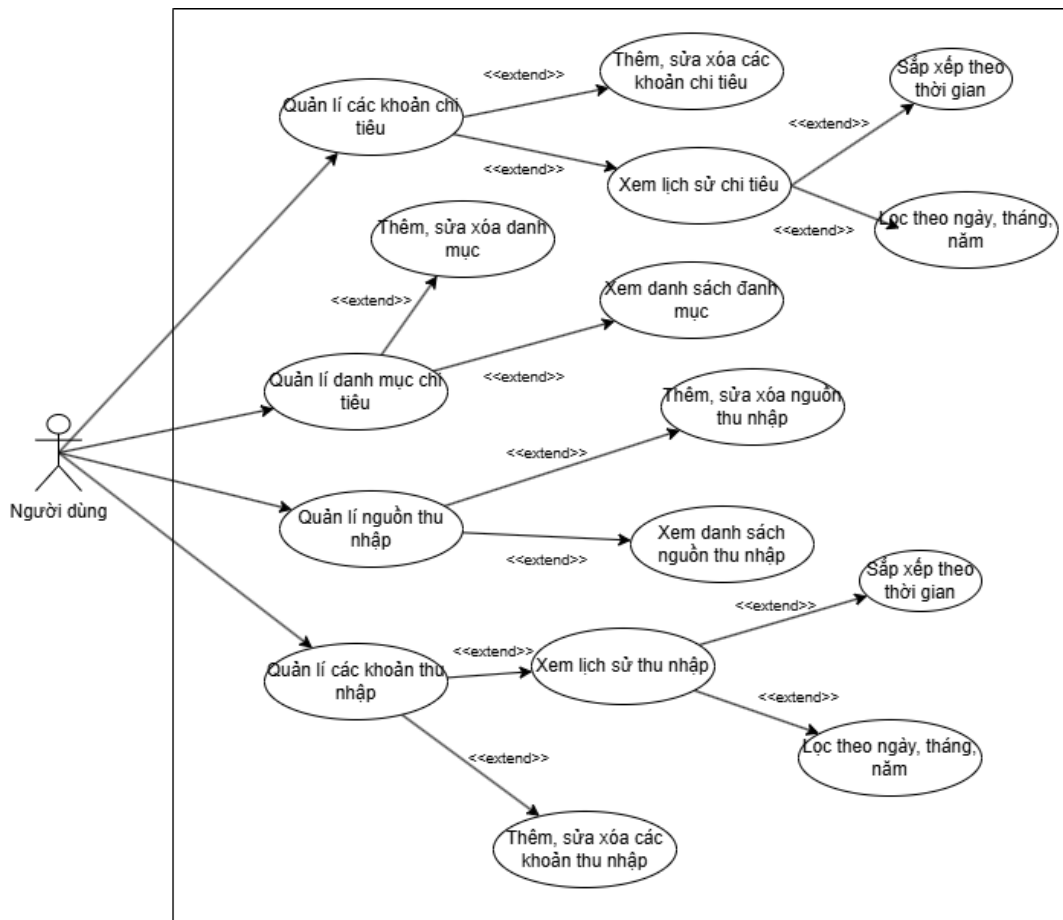
Hình 3.2 UseCase đăng ký và đăng nhập

UseCase thống kê tài chính:



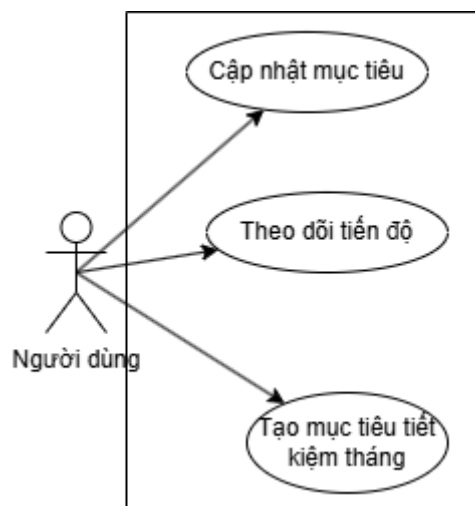
Hình 3.3 UseCase thống kê tài chính

UseCase quản lý chỉ tiêu và thu nhập:



Hình 3.4 UseCase quản lý chỉ tiêu và thu nhập

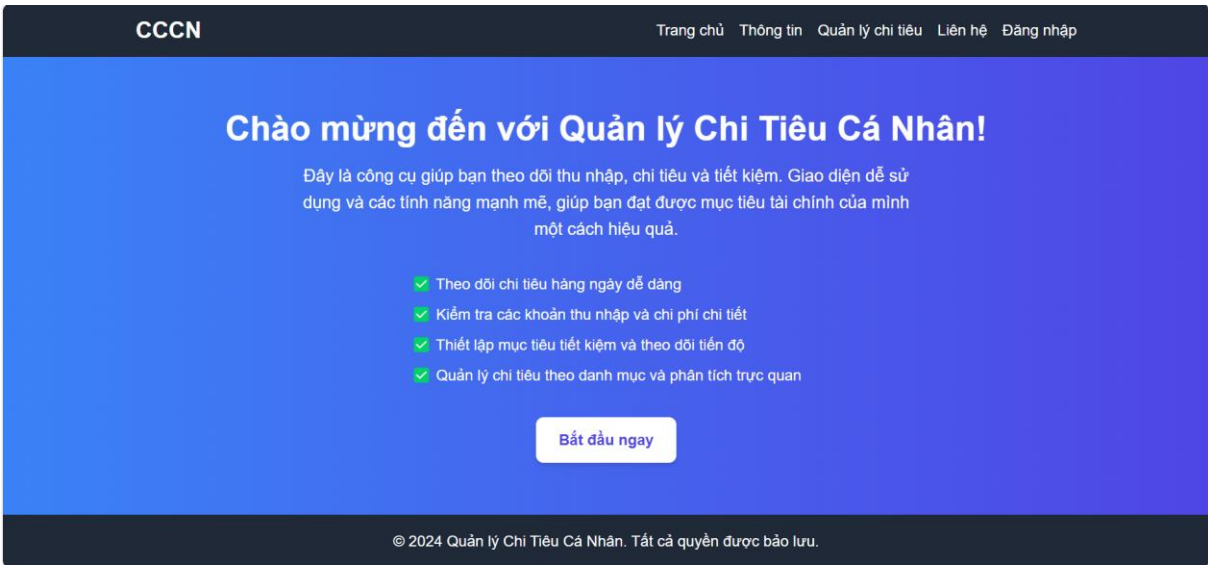
UseCase thiết lập mục tiêu tiết kiệm:



Hình 3.5 UseCase mục tiêu tiết kiệm

## CHƯƠNG 4: KẾT QUẢ NGHIÊN CỨU

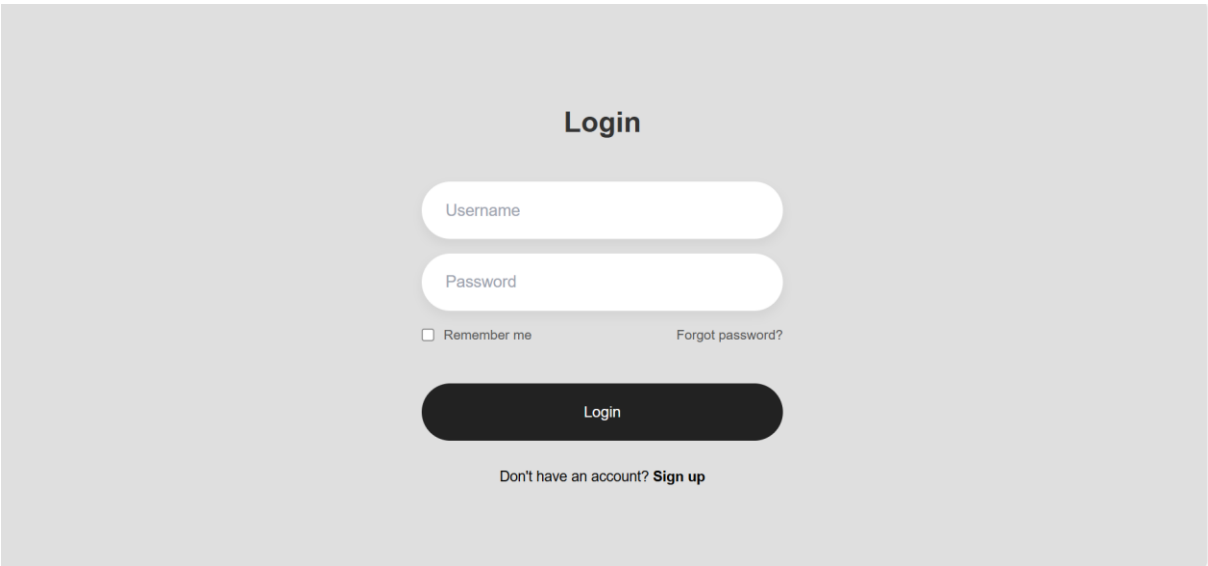
### 4.1 Giao diện trang chủ



Hình 4.1 Hình ảnh trang chủ

Đây là trang để chào mừng người dùng giới thiệu với người dùng về trang quản lý chi tiêu cá nhân

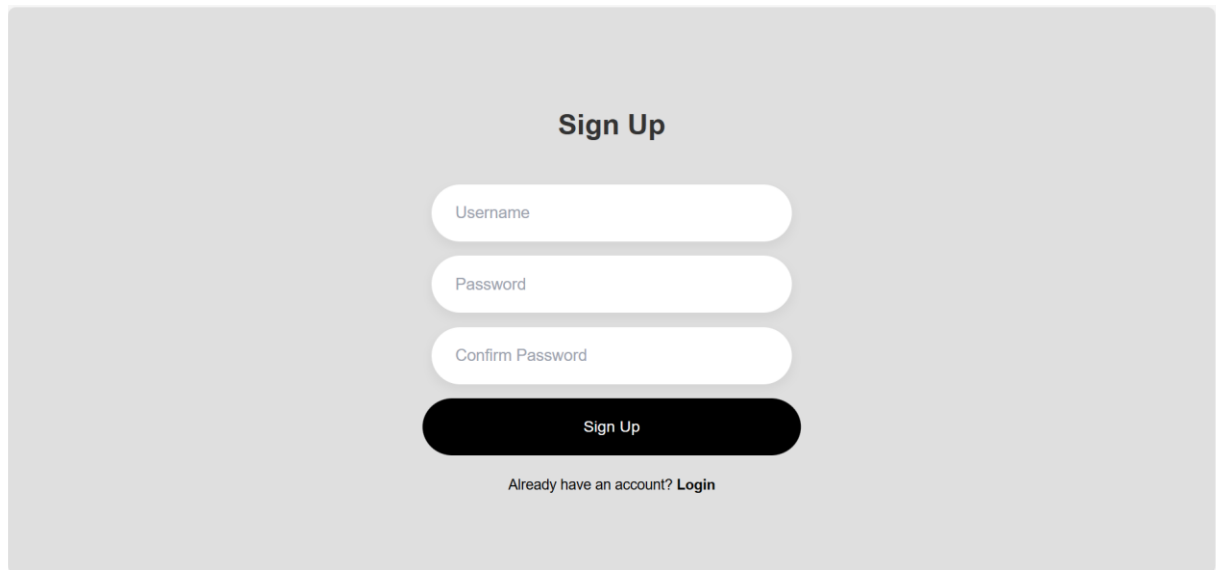
### 4.2 Giao diện trang đăng nhập



Hình 4.2 Trang đăng nhập

Trang để người dùng nhập thông tin để đăng nhập vào hệ thống.

### 4.3 Giao diện trang đăng ký

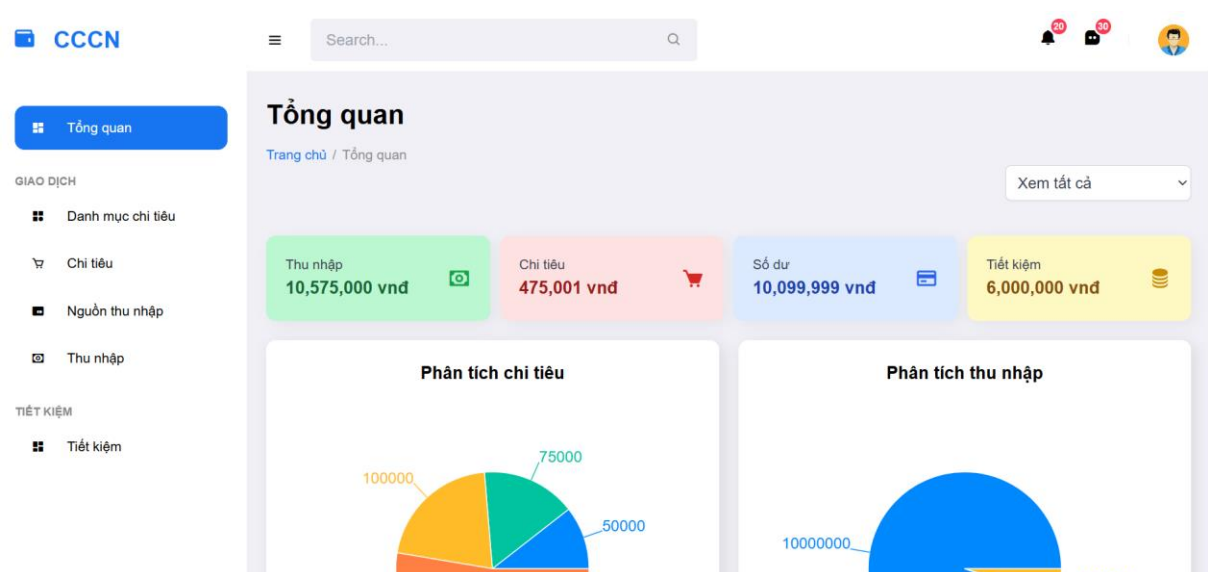


The image shows a 'Sign Up' form with a light gray background. At the top, the text 'Sign Up' is centered in a bold, dark font. Below it are three white input fields with rounded corners, each with a placeholder label: 'Username', 'Password', and 'Confirm Password'. These fields are stacked vertically. Below the fields is a black button with the text 'Sign Up' in white. At the bottom, there is a link that says 'Already have an account? Login'.

Hình 4.3 Trang đăng ký

Để có tài khoản để đăng nhập vào hệ thống người dùng cần đăng ký tài khoản, trang đăng ký giúp người dùng tạo tài khoản đăng nhập.

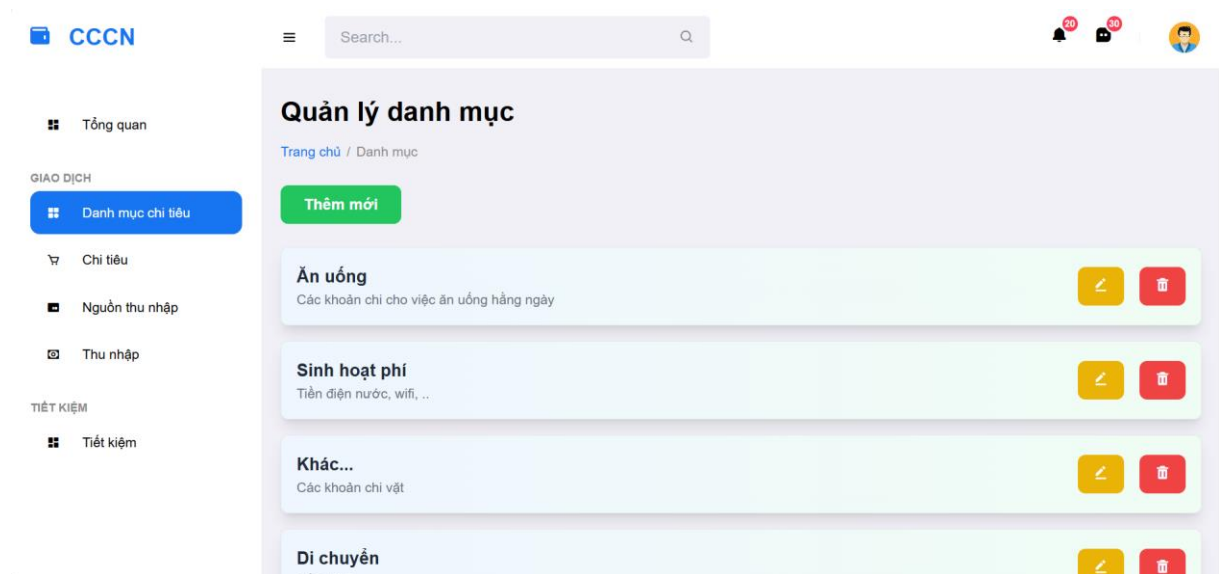
### 4.4 Trang tổng quan



Hình 4.4 Trang tổng quan

Là trang chứa các thông tin tổng quát về tổng thu nhập, chi tiêu, số dư của người dùng. Ngoài ra còn có các biểu đồ phân tích chi tiêu giúp người dùng nắm được tổng quan tình hình tài chính bản thân

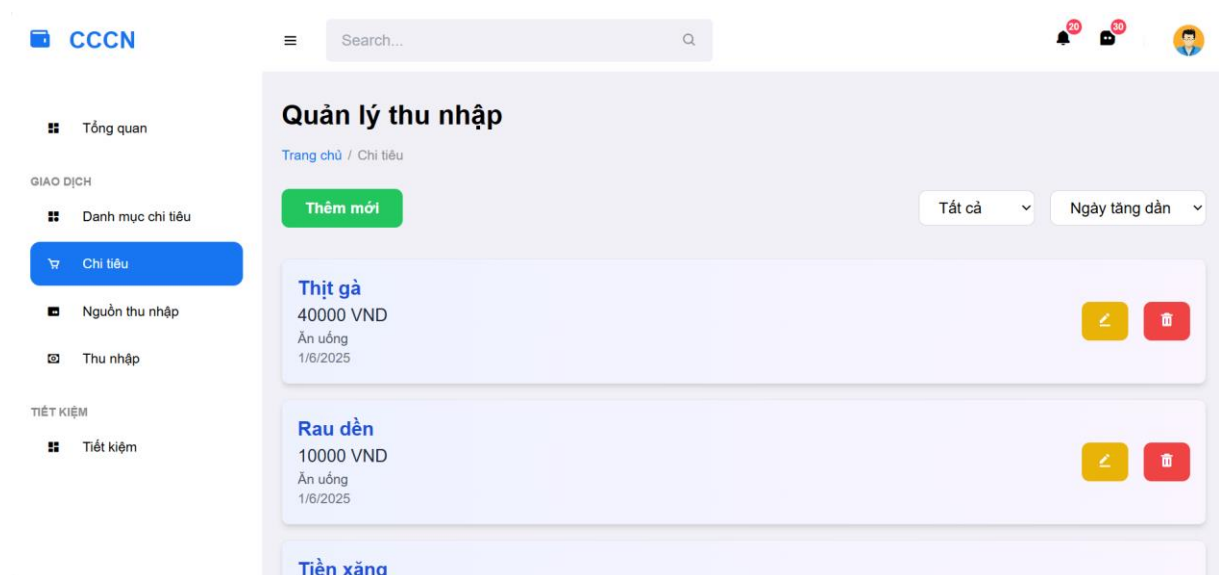
## 4.5 Trang quản lý danh mục



Hình 4.5 Trang danh mục

Là trang để người dùng có thể quản lý danh mục bản thân, người dùng có thể tạo mới, thêm sửa xóa danh mục.

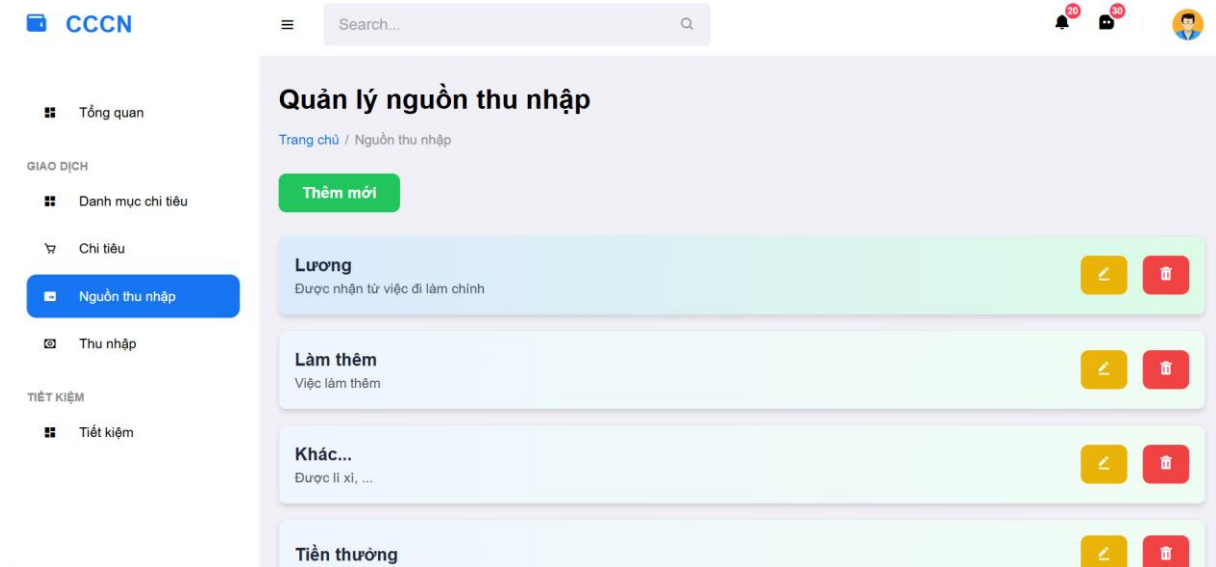
## 4.6 Trang quản lý chi tiêu



Hình 4.6 Trang chi tiêu

Là trang để người dùng có thể quản lý chi tiêu bản thân, người dùng có thể tạo mới, thêm sửa xóa chi tiêu. Xem lịch sử các khoản chi tiêu theo ngày, tháng năm,...

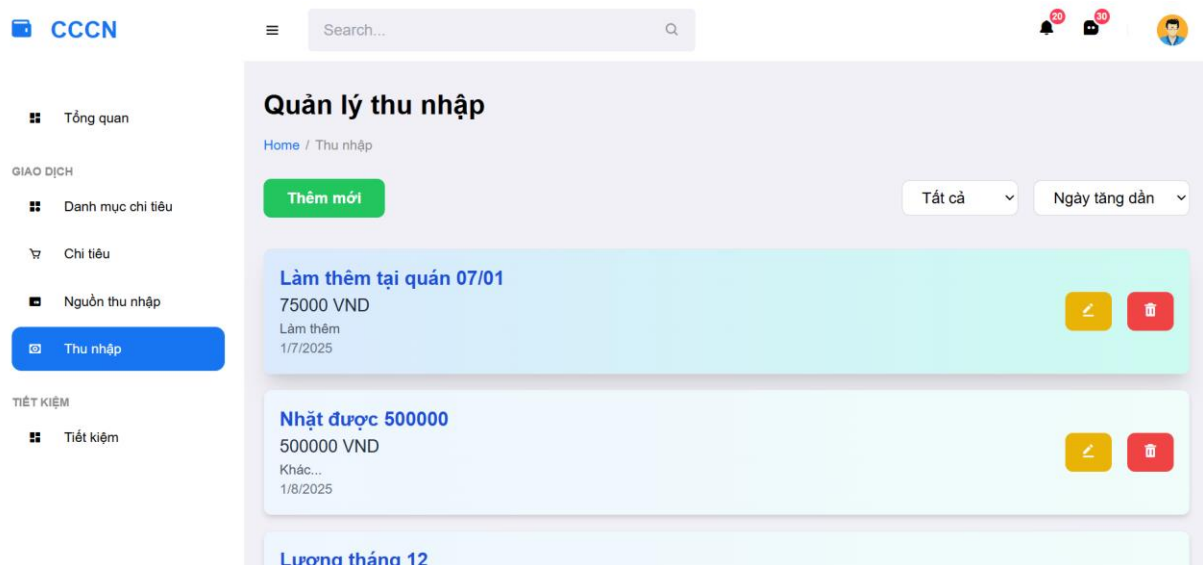
## 4.7 Trang quản lý nguồn thu nhập



Hình 4.7 Trang nguồn thu nhập

Là trang để người dùng có thể quản lý các nguồn thu nhập, người dùng có thể tạo mới, thêm sửa xóa nguồn thu nhập.

## 4.8 Trang quản lý thu nhập




Hình 4.8 Trang thu nhập

Là trang để người dùng có thể quản lý thu nhập bản thân, người dùng có thể tạo mới, thêm sửa xóa thu nhập. Hiện thị danh sách thu nhập.

## 4.9 Trang thông tin người dùng

### Thông tin người dùng



Choose File

download (3).jpg

Update

Tên:

Phước Vinh

Tên đăng nhập:

vinhne2605

Chỉnh sửa thông tin

Đổi mật khẩu

Hình 4.9 Trang thông tin người dùng

Trang để người dùng có thể xem thông tin của bản thân có thể sửa thông tin, cập nhật ảnh đại diện

## CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

### 5.1 Kết luận:

Về thực hiện, đã xây dựng được webstie quản lý chi tiêu hiệu quả, giúp người dùng dễ dàng theo dõi và kiểm soát các khoản chi tiêu hàng ngày. Hệ thống này không chỉ cung cấp các chức năng cơ bản như ghi chép chi tiêu, phân loại chi tiêu mà còn tích hợp các tính năng nâng cao như báo cáo thống kê.

Về kiến thức, qua quá trình nghiên cứu và phát triển, em đã học được nhiều kiến thức và kỹ năng mới, từ việc phân tích yêu cầu, thiết kế hệ thống đến triển khai và kiểm thử. Em đã nắm vững các công nghệ như NestJS, ReactJS, giúp nâng cao hiệu quả và chất lượng của sản phẩm. Những kinh nghiệm này sẽ là nền tảng vững chắc cho em trong các dự án tương lai.

### 5.2 Hướng phát triển:

Trong tương lai, website quản lý chi tiêu cá nhân có thể được phát triển thêm nhiều tính năng và cải tiến để đáp ứng tốt hơn nhu cầu của người dùng. Một số hướng phát triển tiềm năng bao gồm:

- Tối ưu lại mã nguồn website: Điều chỉnh mã nguồn, tối ưu hóa hiệu năng cho website.
- Phát triển hoàn thiện hơn các tính năng: Điều chỉnh, cải thiện các tính năng.
- Tích hợp với các dịch vụ tài chính: Kết nối với các ngân hàng ví điện tử, dịch vụ tài chính khác.



## DANH MỤC TÀI LIỆU THAM KHẢO

- [1] “What is a RESTful API?” AWS, truy cập vào 12/11/2024. <https://aws.amazon.com/what-is/restful-api/>
- [2] “What Exactly is Node.js? Explained for Beginners” freeCodeCamp, truy cập 12/11/2024. <https://www.freecodecamp.org/news/what-is-node-js/>
- [3] “Node.js là gì? Tổng hợp kiến thức NodeJS từ A-Z ” TopDev, truy cập 13/11/2024. <https://topdev.vn/blog/node-js-la-gi/>
- [4] “NestJs Introduction” NestJs, truy cập 14/11/2024. <https://docs.nestjs.com/>
- [5] “React Là Gì? Tất Tần Tật Thông Tin Cần Biết Về React.js” Mona.media, truy cập 14/11/2024. <https://monamedia.co/React.js-la-gi/>
- [6] “NoSQL là gì?” AWS, truy cập 14/11/2024, <https://aws.amazon.com/vi/nosql/>
- [7] “MongoDB là gì? Tính năng nổi bật từ MongoDB bạn cần biết” MatBao, truy cập 14/11/2024, <https://wiki.matbao.net/mongodb-la-gi-tinh-nang-noi-bat-tu-mongodb-ban-can-biet/>

## **PHỤ LỤC**