**xxxxxx**

**xxxxxxxxxxxxxx**

**xxxxxxxxxxxxxxxxxxxxxxxxxxx**

-----🙞🙜🕮🙞🙜-----

**KHOÁ LUẬN**

**TỐT NGHIỆP ĐẠI HỌC**

**Đề tài:**

**TÌM HIỂU WINFORM  
XÂY DỰNG ỨNG DỤNG QUẢN LÝ   
TÀI CHÍNH DANH CHO SINH VIÊN**

Sinh viên thực hiện : xxxxx

Khóa : xxxxxxxxxxx

**Hồ Chí Minh, 12 – 2023**

**XXXXXX**

**XXXXXXXXXXX**

**XXXXXXXXX**

-----🙞🙜🕮🙞🙜-----

**KHOÁ LUẬN**

**TỐT NGHIỆP ĐẠI HỌC**

**Đề tài:**

**TÌM HIỂU WINFORM  
XÂY DỰNG ỨNG DỤNG QUẢN LÝ   
TÀI CHÍNH DANH CHO SINH VIÊN**

Sinh viên thực hiện : xxxxx

Khóa : xxx – Hệ chính quy

Giáo viên hướng dẫn : xxxxxxxxx

**Hồ Chí Minh, 12 – 20xx**

**LỜI CẢM ƠN**

Kính thưa Quý thầy cô giáo,

Trước hết, chúng em xin gửi lời cảm ơn chân thành nhất đến Quý thầy cô giáo trong trường Đại học [Tên trường], Bộ môn [Chuyên ngành] đã truyền đạt cho chúng em những kiến thức bổ ích trong suốt quá trình học tập. Nhờ sự giảng dạy tận tình, tâm huyết của Quý thầy cô, chúng em đã có được nền tảng kiến thức vững chắc về các môn đại cương cũng như các môn chuyên ngành, từ đó có thể tự tin thực hiện đề tài khóa luận tốt nghiệp này.

Đặc biệt, chúng em xin chân thành cảm ơn [tên thầy] đã dành nhiều thời gian, tâm huyết hướng dẫn, định hướng, chỉ bảo chúng em trong suốt quá trình làm khóa luận. Nhờ sự giúp đỡ của thầy, chúng em đã có thể vượt qua những khó khăn, vướng mắc trong quá trình nghiên cứu và hoàn thành đề tài một cách tốt nhất.

Chúng em cũng xin gửi lời cảm ơn chân thành đến bạn bè, người thân đã luôn tạo điều kiện, quan tâm, giúp đỡ, động viên chúng em trong suốt quá trình học tập và hoàn thành khóa luận tốt nghiệp.

Tuy nhiên, do vốn kiến thức chuyên ngành và kinh nghiệm cá nhân còn hạn hẹp nên việc xảy ra nhiều thiếu sót trong nội dung báo cáo là điều không thể tránh khỏi. Chúng em rất mong nhận được sự góp ý, chỉ bảo của Quý thầy cô để hoàn thiện bài báo cáo này tốt hơn.

Một lần nữa, chúng em xin chân thành cảm ơn Quý thầy cô giáo, bạn bè và người thân đã luôn đồng hành và giúp đỡ chúng em trong suốt thời gian qua.

Trân trọng!

Hồ Chí Minh, x tháng x năm 20xx

Sinh viên thực hiện

TÂN + ĐẠT

# MỤC LỤC

[MỤC LỤC ii](#_Toc152282340)

[DANH MỤC HÌNH ẢNH iv](#_Toc152282341)

[CHƯƠNG 1: XÂY DỰNG PHẦN MỀM QUẢN LÝ TÀI CHÍNH 1](#_Toc152282342)

[1.1 Phân tích thực trạng 1](#_Toc152282343)

[1.2 Yêu cầu bài toán 3](#_Toc152282344)

[1.3 Mục tiêu hệ thống 4](#_Toc152282345)

[CHƯƠNG 2: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG 6](#_Toc152282346)

[2.1 Mô hình hóa yêu cầu nghiệp vụ 6](#_Toc152282347)

[2.3.1 Xác định Actor 6](#_Toc152282348)

[2.3.2 Xác định Use case 6](#_Toc152282349)

[2.3.3 Sơ đồ Use case 8](#_Toc152282350)

[2.3.4 Sơ đồ hoạt động (Activity Diagram) 15](#_Toc152282351)

[2.3.5 Sơ đồ tuần tự (Sequence Diagram) 16](#_Toc152282352)

[2.2 Mô hình hóa quy trình nghiệp vụ 17](#_Toc152282353)

[2.3 Mô hình hóa cơ sở dữ liệu 17](#_Toc152282354)

[2.3.1 Thiết kế cơ sở dữ liệu 17](#_Toc152282355)

[2.3.2 Ý nghĩa các bảng 18](#_Toc152282356)

[2.4 Thiết kế giao diện hệ thống 24](#_Toc152282357)

[2.5.1 Giao diện dành cho sinh viên 24](#_Toc152282358)

[2.5.2 Giao diện dành cho admin 24](#_Toc152282359)

[2.5 Cài đặt phần mềm lập trình và công nghệ sử dụng 24](#_Toc152282360)

[2.5.1 Công cụ lập trình 24](#_Toc152282361)

[2.5.1 Công nghệ sử dụng 25](#_Toc152282362)

[CHƯƠNG 3: TỔNG QUAN VỀ WINFORM 25](#_Toc152282363)

[3.1 Giới thiệu về .NET Framework 25](#_Toc152282364)

[3.1.1 Định nghĩa về .NET Framework 25](#_Toc152282365)

[3.1.2 Lịch sử của .NET Framework 26](#_Toc152282366)

[3.1.3 Thành phần cơ bản của .NET Framework 27](#_Toc152282367)

[3.1.4 Đặc điểm .NET Framework 35](#_Toc152282368)

[3.1.5 Ưu điểm và nhược điểm của .NET Framework 36](#_Toc152282369)

[3.1.6 Ứng dụng của .NET Framework 38](#_Toc152282370)

[3.2. Giới thiệu về Winform 40](#_Toc152282371)

[3.2.1 Tổng quan về Winform 40](#_Toc152282372)

[3.2.2 Kiến trúc của Winform 41](#_Toc152282373)

[3.2.3 Cách thức hoạt động của Winform 43](#_Toc152282374)

[3.2.4 Ưu điểm và nhược điểm của Winform 45](#_Toc152282375)

[3.2.4 Ứng dụng của Winform 46](#_Toc152282376)

[CHƯƠNG 4: XÂY DỰNG PHẦN MỀM QUẢN LÝ TÀI CHO SINH VIÊN 48](#_Toc152282377)

[4.1 Thiết kế giao phần mềm 48](#_Toc152282378)

[4.1.1 Giao diện đăng nhập 48](#_Toc152282379)

[4.1.2 Giao diện đăng ký cho sinh viên 48](#_Toc152282380)

[4.1.3 Giao diện trang chủ sinh viên 48](#_Toc152282381)

[4.1.4 Giao diện quản lý vay mượn cho sinh viên 48](#_Toc152282382)

[4.1.5 Giao diện quản lý nguồn thu cho sinh viên 48](#_Toc152282383)

[4.1.6 Giao diện quản lý chi tiêu cho sinh viên 48](#_Toc152282384)

[4.1.7 Giao diện quản lý kế hoạch tài chính cho sinh viên 48](#_Toc152282385)

[4.1.8 Giao diện quản lý giao dịch cho sinh viên 48](#_Toc152282386)

[4.1.9 Giao diện cập nhật thông tin tài khoản sinh viên 48](#_Toc152282387)

[4.1.10 Giao diện thống kê cho admin 48](#_Toc152282388)

[4.1.11 Giao diện quản lý địa điểm cho admin 48](#_Toc152282389)

[4.1.12 Giao diện quản lý tài khoản cho admin 48](#_Toc152282390)

[4.1.13 Giao diện quản lý giao dịch cho admin 48](#_Toc152282391)

[4.1.14 Giao diện quản lý nguồn thu cho admin 48](#_Toc152282392)

[4.1.15 Giao diện quản lý chi tiêu cho admin 48](#_Toc152282393)

[4.1.16 Giao diện quản lý vay mượn cho admin 48](#_Toc152282394)

[4.1.17 Giao diện quản lý kế hoạch tài chính cho admin 48](#_Toc152282395)

[4.2 Cài đặt phần mềm 48](#_Toc152282396)

[KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN TƯƠNG LAI 49](#_Toc152282397)

[1.1 Kết luận 49](#_Toc152282398)

[1.2 Hướng phát triển tương lai 50](#_Toc152282399)

[TÀI LIỆU THAM KHẢO 52](#_Toc152282400)

# DANH MỤC HÌNH ẢNH

# CHƯƠNG 1: XÂY DỰNG PHẦN MỀM QUẢN LÝ TÀI CHÍNH

### 1.1 Phân tích thực trạng

Sinh viên là nhóm đối tượng có thu nhập hạn chế và chi tiêu thường xuyên. Do đó, việc quản lý tài chính cá nhân là vô cùng quan trọng đối với sinh viên. Tuy nhiên, thực tế cho thấy, nhiều sinh viên chưa có kỹ năng quản lý tài chính hiệu quả, dẫn đến tình trạng chi tiêu vượt quá khả năng, thiếu hụt tiền bạc, thậm chí là nợ nần.

* Một số nguyên nhân dẫn đến thực trạng này là:
* Thiếu kiến thức và kỹ năng quản lý tài chính: Nhiều sinh viên chưa được trang bị kiến thức và kỹ năng quản lý tài chính cần thiết. Họ thường chỉ dựa vào cảm tính và kinh nghiệm cá nhân trong việc chi tiêu, dẫn đến tình trạng chi tiêu không hợp lý.
* Tác động của xã hội: Xã hội hiện đại với sự phát triển của công nghệ, mạng xã hội, các hoạt động giải trí, mua sắm ngày càng đa dạng, phong phú. Điều này khiến sinh viên dễ bị chi phối bởi các nhu cầu tiêu dùng, dẫn đến chi tiêu quá mức.
* Thiếu ý thức tiết kiệm: Nhiều sinh viên chưa có ý thức tiết kiệm, thường chi tiêu tất cả số tiền có được, không có kế hoạch tích lũy cho tương lai.
* Để giúp sinh viên quản lý tài chính hiệu quả, cần có các giải pháp sau:
* Tăng cường giáo dục tài chính cho sinh viên: Các trường đại học, cao đẳng cần đưa nội dung giáo dục tài chính vào chương trình đào tạo, giúp sinh viên trang bị kiến thức và kỹ năng quản lý tài chính cần thiết.
* Tạo môi trường sống lành mạnh: Các tổ chức đoàn thể, hội sinh viên cần tổ chức các hoạt động tuyên truyền, giáo dục về ý thức tiết kiệm, chi tiêu hợp lý cho sinh viên.
* Hỗ trợ sinh viên tiếp cận các nguồn vốn vay ưu đãi: Nhà nước cần có các chính sách hỗ trợ sinh viên tiếp cận các nguồn vốn vay ưu đãi để đáp ứng nhu cầu học tập, sinh hoạt.
* Giải pháp cụ thể cho hệ thống quản lý tài chính cá nhân cho sinh viên

Chức năng của hệ thống quản lý tài chính cá nhân cho sinh viên được phân tích ở trên là phù hợp với nhu cầu thực tế của sinh viên. Tuy nhiên, để hệ thống phát huy hiệu quả, cần có những giải pháp cụ thể sau:

* Đơn giản, dễ sử dụng: Hệ thống cần được thiết kế đơn giản, dễ sử dụng, phù hợp với khả năng tiếp cận của sinh viên.
* Tính linh hoạt: Hệ thống cần có tính linh hoạt cao, đáp ứng được nhu cầu quản lý tài chính đa dạng của sinh viên.
* Tính bảo mật: Hệ thống cần có tính bảo mật cao, đảm bảo an toàn cho thông tin tài chính của sinh viên.

Với những giải pháp cụ thể trên, hệ thống quản lý tài chính cá nhân cho sinh viên sẽ là công cụ hữu ích giúp sinh viên quản lý tài chính hiệu quả, tiết kiệm chi tiêu, xây dựng kế hoạch tài chính cá nhân vững chắc.

Một số ví dụ cụ thể về cách sử dụng hệ thống quản lý tài chính cá nhân cho sinh viên

* Dùng hệ thống để theo dõi thu nhập: Sinh viên có thể sử dụng hệ thống để theo dõi thu nhập từ các nguồn khác nhau như: tiền làm thêm, học bổng, tiền phụ huynh,... Điều này giúp sinh viên nắm được tình hình thu nhập của mình, từ đó có kế hoạch chi tiêu hợp lý.
* Dùng hệ thống để phân loại chi tiêu: Sinh viên có thể sử dụng hệ thống để phân loại chi tiêu theo các mục đích như: ăn uống, học tập, giải trí,... Điều này giúp sinh viên hiểu rõ hơn về thói quen chi tiêu của mình, từ đó có thể điều chỉnh cho phù hợp.
* Dùng hệ thống để lập kế hoạch tài chính: Sinh viên có thể sử dụng hệ thống để lập kế hoạch tài chính cá nhân, bao gồm các mục tiêu tiết kiệm, chi tiêu trong ngắn hạn và dài hạn. Điều này giúp sinh viên có định hướng rõ ràng trong việc sử dụng tiền bạc, tránh chi tiêu lãng phí.

Hệ thống quản lý tài chính cá nhân cho sinh viên là một công cụ hữu ích, có thể giúp sinh viên quản lý tài chính hiệu quả, xây dựng nền tảng tài chính vững chắc cho tương lai.

### 1.2 Yêu cầu bài toán

Mục tiêu:

Xây dựng phần mềm quản lý tài chính cho sinh viên, giúp sinh viên dễ dàng quản lý các hoạt động tài chính của bản thân, từ đó giúp nâng cao hiệu quả quản lý tài chính và đạt được các mục tiêu tài chính cá nhân.

Nội dung:

* Sinh viên đăng nhập vào hệ thống có thực hiện các chức năng:
* Quản lý thông tin cá nhân: Họ tên, ngày sinh, giới tính, địa chỉ, số điện thoại, email,...
* Quản lý nguồn thu: Nguồn thu nhập từ học bổng, làm thêm, tiền phụ huynh,...
* Quản lý chi tiêu: Nguồn chi tiêu cho học tập, sinh hoạt, giải trí,...
* Quản lý tài chính: Vay mượn, kế hoạch tài chính,...
* Quản lý giao dịch: Lịch sử giao dịch chi tiêu, nguồn thu, vay mượn,…
* Quản lý thông báo: Nhắc nhở chi tiêu, công việc, trả nợ,….
* Quản lý thông kế: Doanh thu, hoạt động chi tiêu, công cần làm,…
* Quản trị viên:
* Quản lý thông tin người dùng: bao gồm thông tin cá nhân, thông tin tài khoản,...
* Quản lý lịch sử giao dịch: bao gồm các giao dịch mua hàng, giao dịch thanh toán,...
* Quản lý báo cáo tài chính: bao gồm thu nhập, chi tiêu, vay mượn,...
* Quản lý địa chỉ: bao gồm tỉnh thành, quận huyện, phường xã,…
* Quản lý thông kê: bao gồm báo cáo, lịch sử giao dịch, tài chính….

### 1.3 Mục tiêu hệ thống

Mục tiêu của hệ thống quản lý tài chính cho sinh viên được xây dựng nhằm mục đích giúp sinh viên:

* Hệ thống quản lý tài chính cá nhân cho sinh viên cung cấp các chức năng giúp sinh viên theo dõi thu nhập, chi tiêu, lập kế hoạch tài chính,... Từ đó, sinh viên có thể nắm rõ tình hình tài chính của mình, từ đó có kế hoạch chi tiêu hợp lý, tiết kiệm tiền bạc, đạt được các mục tiêu tài chính cá nhân.
* Hệ thống giúp sinh viên hiểu rõ hơn về thói quen chi tiêu của mình, từ đó có thể điều chỉnh cho phù hợp. Ngoài ra, hệ thống cũng cung cấp các tính năng giúp sinh viên tiết kiệm tiền bạc, như nhắc nhở chi tiêu, phân loại chi tiêu,... Từ đó, sinh viên có thể xây dựng nền tảng tài chính vững chắc cho tương lai.
* Giúp sinh viên theo dõi thu nhập: Hệ thống cung cấp các chức năng giúp sinh viên theo dõi thu nhập từ các nguồn khác nhau như tiền làm thêm, học bổng, tiền phụ huynh,... Từ đó, sinh viên nắm được tình hình thu nhập của mình, từ đó có kế hoạch chi tiêu hợp lý.
* Giúp sinh viên phân loại chi tiêu: Hệ thống cung cấp các chức năng giúp sinh viên phân loại chi tiêu theo các mục đích như ăn uống, học tập, giải trí,... Từ đó, sinh viên hiểu rõ hơn về thói quen chi tiêu của mình, từ đó có thể điều chỉnh cho phù hợp.
* Giúp sinh viên lập kế hoạch tài chính: Hệ thống cung cấp các chức năng giúp sinh viên lập kế hoạch tài chính cá nhân, bao gồm các mục tiêu tiết kiệm, chi tiêu trong ngắn hạn và dài hạn. Từ đó, sinh viên có định hướng rõ ràng trong việc sử dụng tiền bạc, tránh chi tiêu lãng phí.
* Giúp sinh viên tiết kiệm tiền bạc: Hệ thống cung cấp các tính năng giúp sinh viên tiết kiệm tiền bạc, như nhắc nhở chi tiêu, phân loại chi tiêu,... Từ đó, sinh viên có thể xây dựng nền tảng tài chính vững chắc cho tương lai.
* Hệ thống quản lý tài chính cá nhân cho sinh viên là một công cụ hữu ích, có thể giúp sinh viên quản lý tài chính hiệu quả, xây dựng nền tảng tài chính vững chắc cho tương lai. Để hệ thống phát huy hiệu quả, cần có sự phối hợp giữa sinh viên và nhà phát triển hệ thống. Sinh viên cần chủ động sử dụng hệ thống để quản lý tài chính của mình, còn nhà phát triển hệ thống cần thường xuyên cập nhật hệ thống để đáp ứng nhu cầu của sinh viên.

Ngoài ra hệ thống quản lý tài chính cho sinh viên phải đáp ứng các yêu cầu sau:

* Đơn giản, dễ sử dụng: Hệ thống cần được thiết kế đơn giản, dễ sử dụng, phù hợp với khả năng tiếp cận của sinh viên.
* Tính linh hoạt: Hệ thống cần có tính linh hoạt cao, đáp ứng được nhu cầu quản lý tài chính đa dạng của sinh viên.
* Tính bảo mật: Hệ thống cần có tính bảo mật cao, đảm bảo an toàn cho thông tin tài chính của sinh viên.

# CHƯƠNG 2: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

## 2.1 Mô hình hóa yêu cầu nghiệp vụ

### 2.3.1 Xác định Actor

Người sử dụng hệ thống:

* Người dùng (sinh viên)
* Admin (người quản trị hệ thống)

### 2.3.2 Xác định Use case

+ Use Cases cho Sinh Viên:

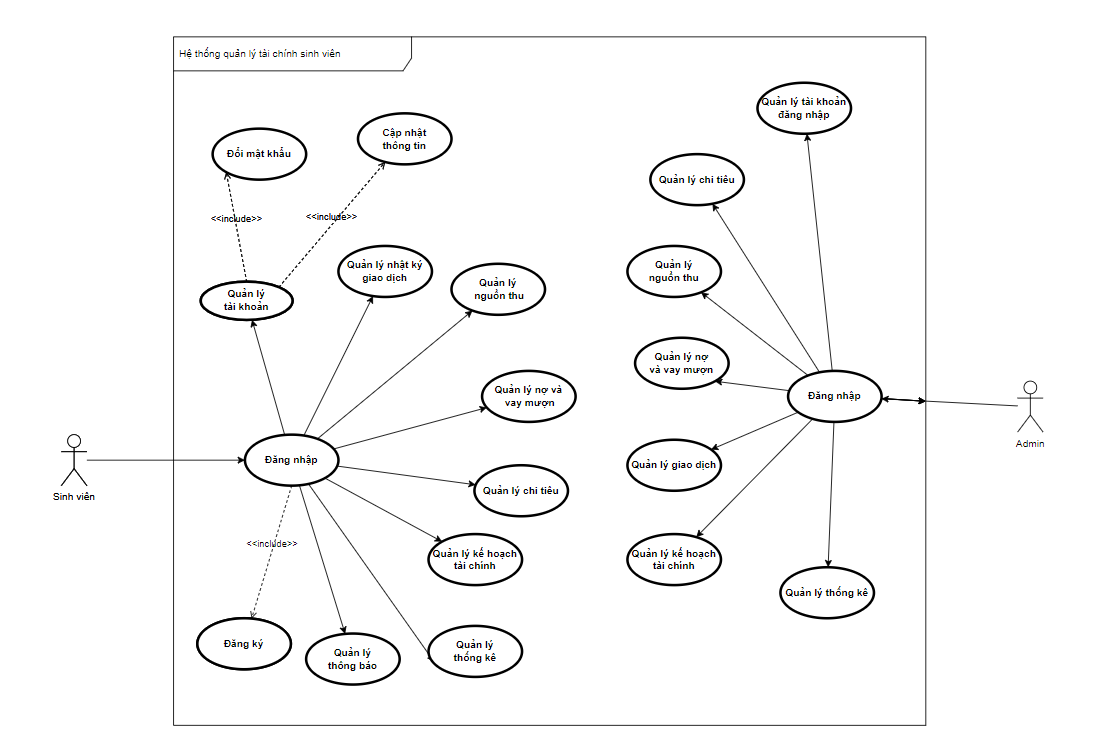
* Đăng nhập: Sinh viên có thể đăng nhập vào hệ thống bằng tài khoản của mình.
* Đăng ký: Sinh viên có thể đăng ký tài khoản để đăng nhập vào hệ thống
* Quản lý thông tin cá nhân: Sinh viên có thể cập nhật và quản lý thông tin cá nhân của mình.
* Quản lý nguồn thu: Sinh viên có thể ghi nhận và quản lý các nguồn thu nhập của mình. Các chức năng: Thêm, sửa, xóa nguồn thu nhập từ học bổng, làm thêm, tiền phụ huynh,...
* Quản lý chi tiêu: Sinh viên có thể ghi nhận và quản lý các chi tiêu của mình. Các chức năng: Thêm, sửa, xóa chi tiêu cho học tập, sinh hoạt, giải trí,...
* Quản lý vay mượn: Sinh viên có thể ghi nhận và quản lý các nguồn vay mượn của mình. Các chức năng: Thêm, sửa, xóa vay mượn từ bạn bè, người thân cho mục đích cho học tập, sinh hoạt, giải trí,...
* Quản lý kế hoạch tài chính: Sinh viên có thể xem và cập nhật thông tin về kế hoạch tài chính phục vụ mục đích cá nhân.
* Quản lý giao dịch: Sinh viên có thể xem lịch sử giao dịch chi tiêu, nguồn thu, vay mượn,...
* Quản lý thông báo: Sinh viên có thể đặt và nhận các thông báo nhắc nhở về chi tiêu, công việc, trả nợ,...
* Quản lý thống lê: Sinh viên có thể xem thông kế doanh thu, hoạt động chi tiêu, công việc cần làm,...

+ Use Cases cho Quản Trị Viên (Admin):

* Quản lý thông tin người dùng: Quản trị viên có thể xem và quản lý thông tin cá nhân của tất cả sinh viên. Các chức năng: Xem thông tin cá nhân, thông tin tài khoản của sinh viên, sửa, xóa thông tin người dùng.
* Quản lý giao dịch: Quản trị viên có thể xem và quản lý lịch sử các giao dịch mua hàng, thanh toán của sinh viên.
* Quản lý địa chỉ: Quản trị viên có thể thêm, sửa, xóa thông tin địa chỉ như tỉnh thành, quận huyện, phường xã,...
* Quản lý vay mượn: Quản trị viên có thể xem và tạo báo cáo về vay mượn của toàn bộ sinh viên.
* Quản lý chi tiêu : Quản trị viên có thể xem và tạo báo cáo về nguồn chi tiêu của toàn bộ sinh viên.
* Quản lý nguồn thu: Quản trị viên có thể xem và tạo báo cáo về thu nhập, của toàn bộ sinh viên.
* Quản lý kế hoạch tài chính : Quản trị viên có thể xem và tạo báo cáo về kế hoạch tài chính của toàn bộ sinh viên.
* Quản lý thông kê: Quản trị viên có thể xem báo cáo, lịch sử giao dịch, tình hình tài chính của toàn bộ sinh viên.

### 2.3.3 Sơ đồ Use case

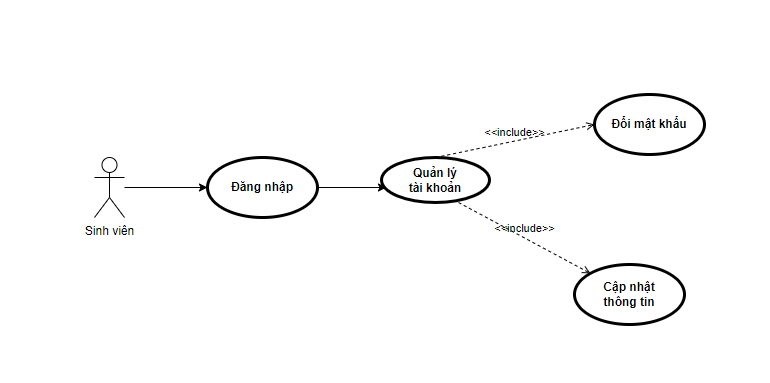
#### 2.3.3.1 Use case tổng quát



*Hình 2.1 Sơ đồ Use case tổng quát*

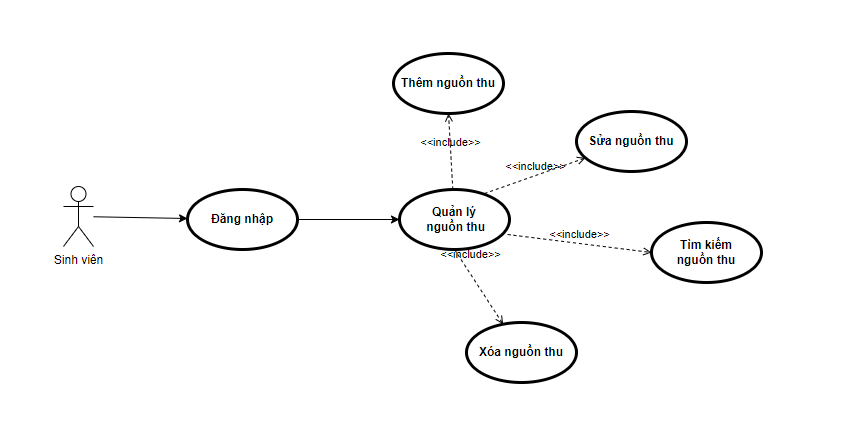
#### 2.3.3.2 Use case sinh viên

* Quản lý thông tin tài khoản



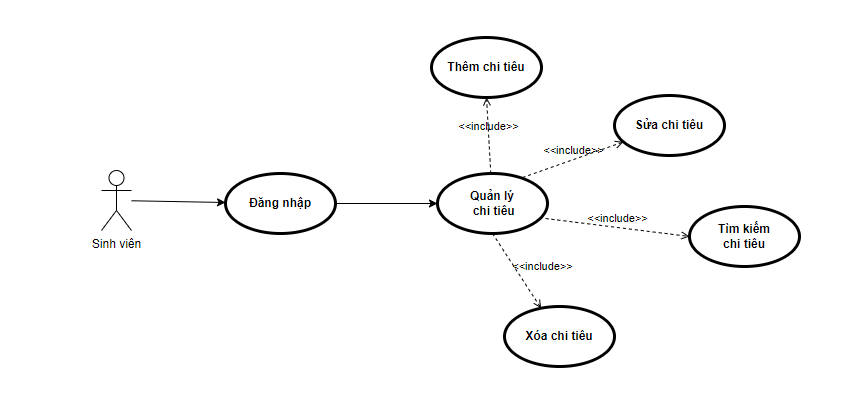
*Hình 2.2 Use case quản lý thông tin tài khoản*

* Quản lý nguồn thu



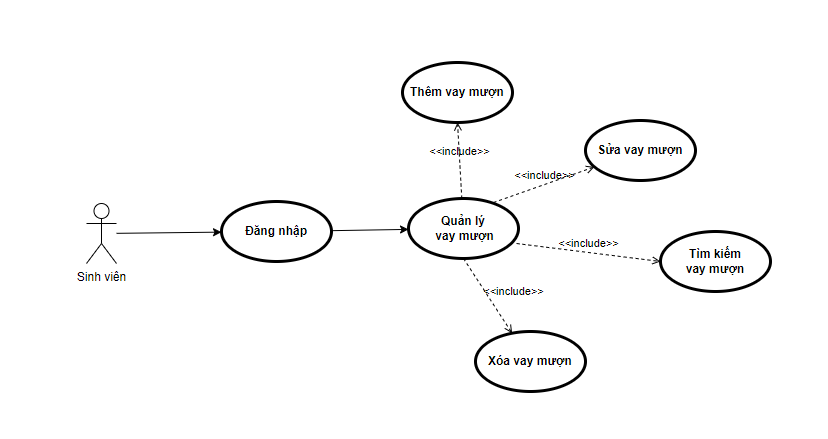
*Hình 2.3* *Use case quản lý nguồn thu*

* Quản lý chi tiêu



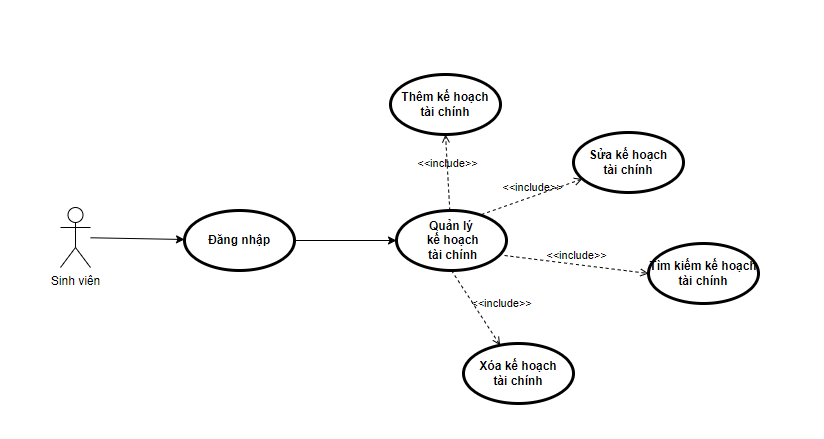
*Hình 2.4* *Use case quản chi tiêu*

* Quản lý vay mượn



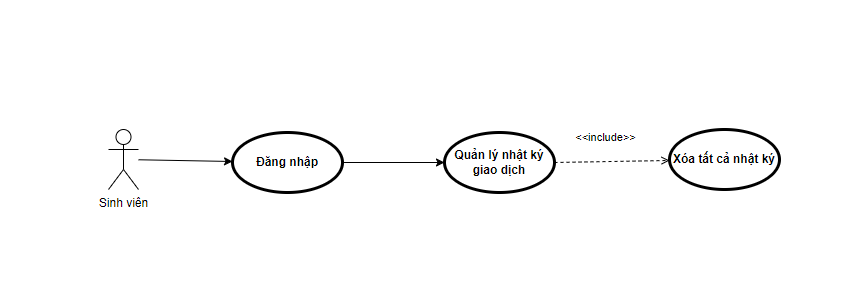
*Hình 2.5* *Use case quản lý vay mượn*

* Quản lý kế hoạch tài chính



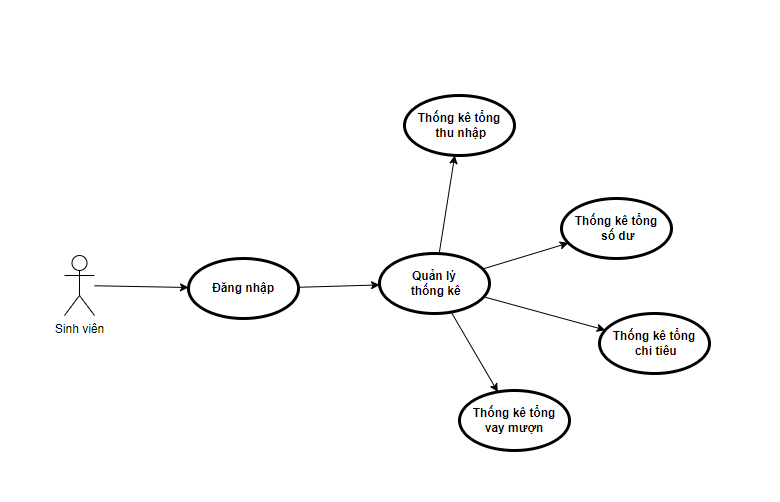
*Hình 2.6* *Use case Quản lý kế hoạch tài chính*

* Quản lý giao dịch



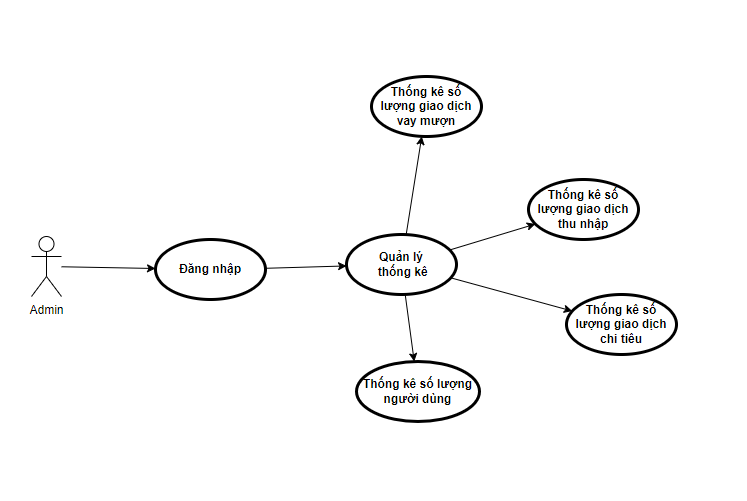
*Hình 2.7* *Use case Quản lý giao dịch*

* Quản lý thống kê

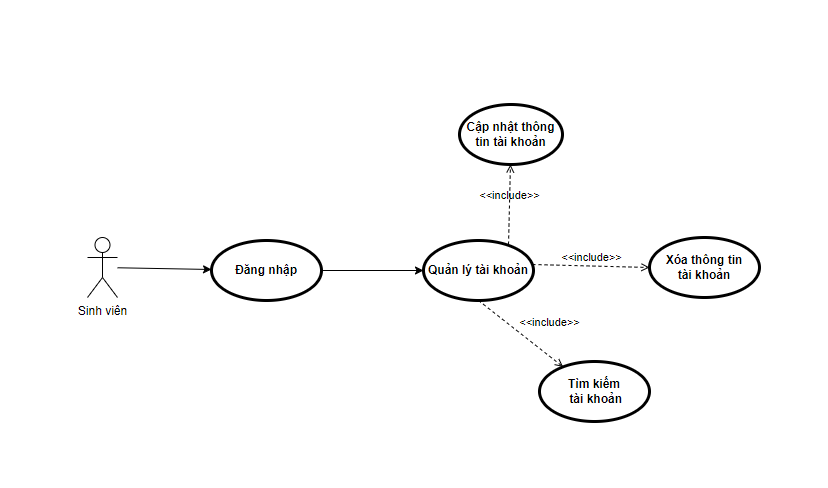


*Hình 2.8* *Use case Quản lý thống kê*

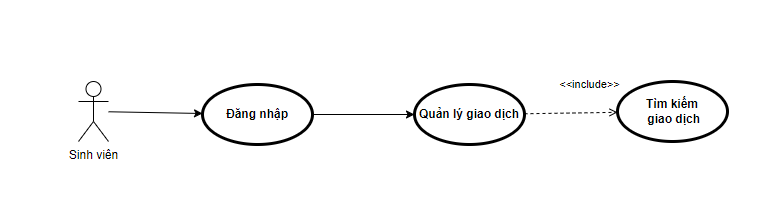
#### 2.3.3.3 Use case của Admin

* Quản lý thống kê  
  

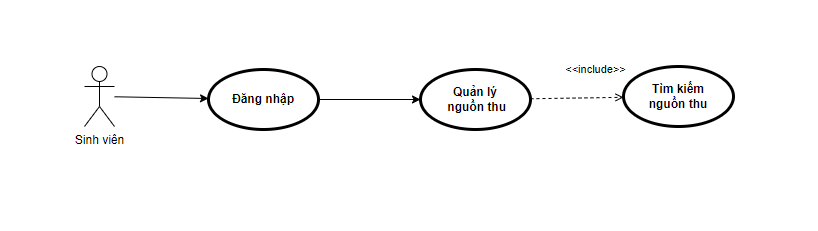
*Hình 2.9 Use case quản lý thống kê*

* Quản lý tài khoản  
  

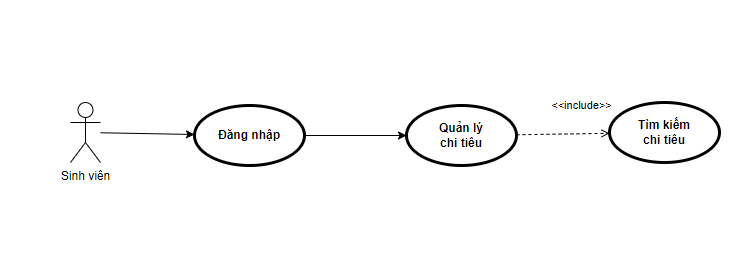
*Hình 2.10 Use case quản lý tài khoản*

* Quản lý giao dịch  
  

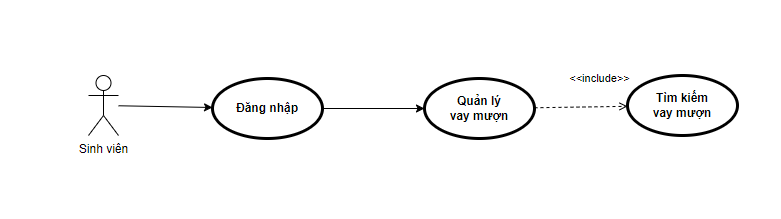
*Hình 2.11 Use case quản lý giao dịch*

* Quản lý nguồn thu  
  

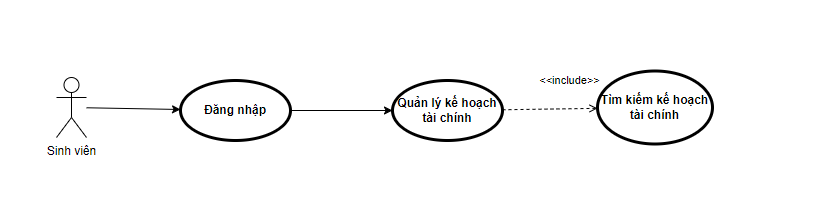
*Hình 2.12 Use case quản lý nguồn thu*

* Quản lý chi tiêu  
  

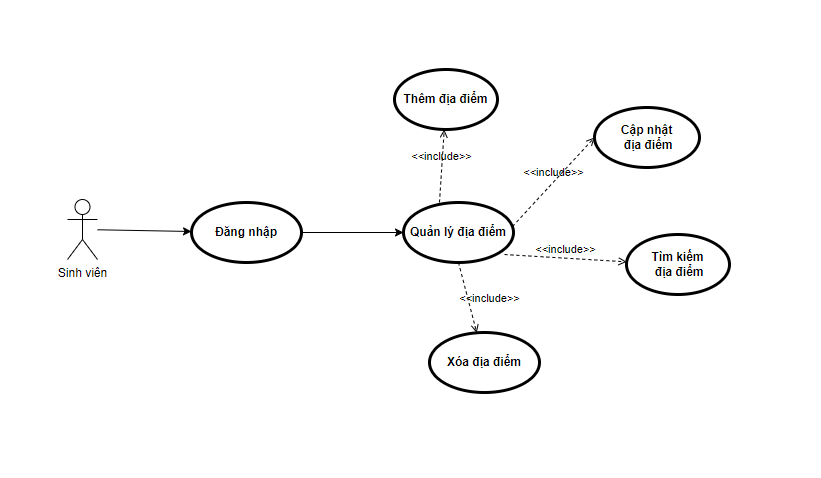
*Hình 2.13 Use case quản lý nhà cung cấp*

* Quản lý tài vay mượn  
  

*Hình 2.14 Use case quản lý vay mượn*

* Quản lý kế hoạch tài chính  
  

*Hình 2.15 Use case quản lý kế hoạch tài chính*

* Quản lý địa điểm  
  

*Hình 2.16 Use case quản lý địa điểm*

### 2.3.4 Sơ đồ hoạt động (Activity Diagram)

#### 2.3.4.1 Đăng nhập cho sinh viên

#### 2.3.4.2 Đăng ký cho sinh viên

#### 2.3.4.3 Thêm nguồn thu cho sinh viên

#### 2.3.4.4 Sửa nguồn thu cho sinh viên

#### 2.3.4.5 Xóa nguồn thu cho sinh viên

#### 2.3.4.6 Tìm kiếm nguồn thu cho sinh viên

#### 2.3.4.7 Thêm chi tiêucho sinh viên

#### 2.3.4.8 Sửa chi tiêu cho sinh viên

#### 2.3.4.9 Xóa chi tiêu cho sinh viên

#### 2.3.4.10 Tìm kiếm chi tiêu cho sinh viên

#### 2.3.4.11 Thêm vay mượn cho sinh viên

#### 2.3.4.12 Sửa vay mượn cho sinh viên

#### 2.3.4.13 Xóa vay mượn cho sinh viên

#### 2.3.4.14 Tìm kiếm vay mượn cho sinh viên 2.3.4.15 Thêm kế hoạch tài chính cho sinh viên

#### 2.3.4.16 Sửa kế hoạch tài chính cho sinh viên

#### 2.3.4.17 Xóa kế hoạch tài chính cho sinh viên

#### 2.3.4.18 Tìm kiếm vay kế hoạch tài chính cho sinh viên

#### 2.3.4.19 Xóa giao dịch cho sinh viên

#### 2.3.4.20 Cập nhật thông tin cá nhân cho sinh viên

#### 2.3.4.21 Thêm địa điểm cho admin

#### 2.3.4.22 Sửa địa điểm cho admin

#### 2.3.4.23 Xóa địa điểm cho admin

#### 2.3.4.24 Tìm kiếm địa điểm cho admin 2.3.4.25 Thêm tài khoản cho admin

#### 2.3.4.26 Sửa tài khoản cho admin

#### 2.3.4.27 Xóa tài khoản cho admin

#### 2.3.4.28 Tìm kiếm tài khoản cho admin

#### 2.3.4.29 Tìm kiếm nguồn thu cho admin

#### 2.3.4.30 Tìm kiếm chi tiêu cho admin

#### 2.3.4.31 Tìm kiếm vay mượn cho admin

#### 2.3.4.32 Tìm kiếm giao dịch cho admin

### 2.3.5 Sơ đồ tuần tự (Sequence Diagram)

#### 2.3.5.1 Đăng nhập cho sinh viên

#### 2.3.5.2 Đăng ký cho sinh viên

#### 2.3.5.3 Thêm nguồn thu cho sinh viên

#### 2.3.5.5 Sửa nguồn thu cho sinh viên

#### 2.3.5.5 Xóa nguồn thu cho sinh viên

#### 2.3.5.6 Tìm kiếm nguồn thu cho sinh viên

#### 2.3.5.7 Thêm chi tiêucho sinh viên

#### 2.3.5.8 Sửa chi tiêu cho sinh viên

#### 2.3.5.9 Xóa chi tiêu cho sinh viên

#### 2.3.5.10 Tìm kiếm chi tiêu cho sinh viên

#### 2.3.5.11 Thêm vay mượn cho sinh viên

#### 2.3.5.12 Sửa vay mượn cho sinh viên

#### 2.3.5.13 Xóa vay mượn cho sinh viên

#### 2.3.5.15 Tìm kiếm vay mượn cho sinh viên 2.3.5.15 Thêm kế hoạch tài chính cho sinh viên

#### 2.3.5.16 Sửa kế hoạch tài chính cho sinh viên

#### 2.3.5.17 Xóa kế hoạch tài chính cho sinh viên

#### 2.3.5.18 Tìm kiếm vay kế hoạch tài chính cho sinh viên

#### 2.3.5.19 Xóa giao dịch cho sinh viên

#### 2.3.5.20 Cập nhật thông tin cá nhân cho sinh viên

#### 2.3.5.21 Thêm địa điểm cho admin

#### 2.3.5.22 Sửa địa điểm cho admin

#### 2.3.5.23 Xóa địa điểm cho admin

#### 2.3.5.25 Tìm kiếm địa điểm cho admin 2.3.5.25 Thêm tài khoản cho admin

#### 2.3.5.26 Sửa tài khoản cho admin

#### 2.3.5.27 Xóa tài khoản cho admin

#### 2.3.5.28 Tìm kiếm tài khoản cho admin

#### 2.3.5.29 Tìm kiếm nguồn thu cho admin

#### 2.3.5.30 Tìm kiếm chi tiêu cho admin

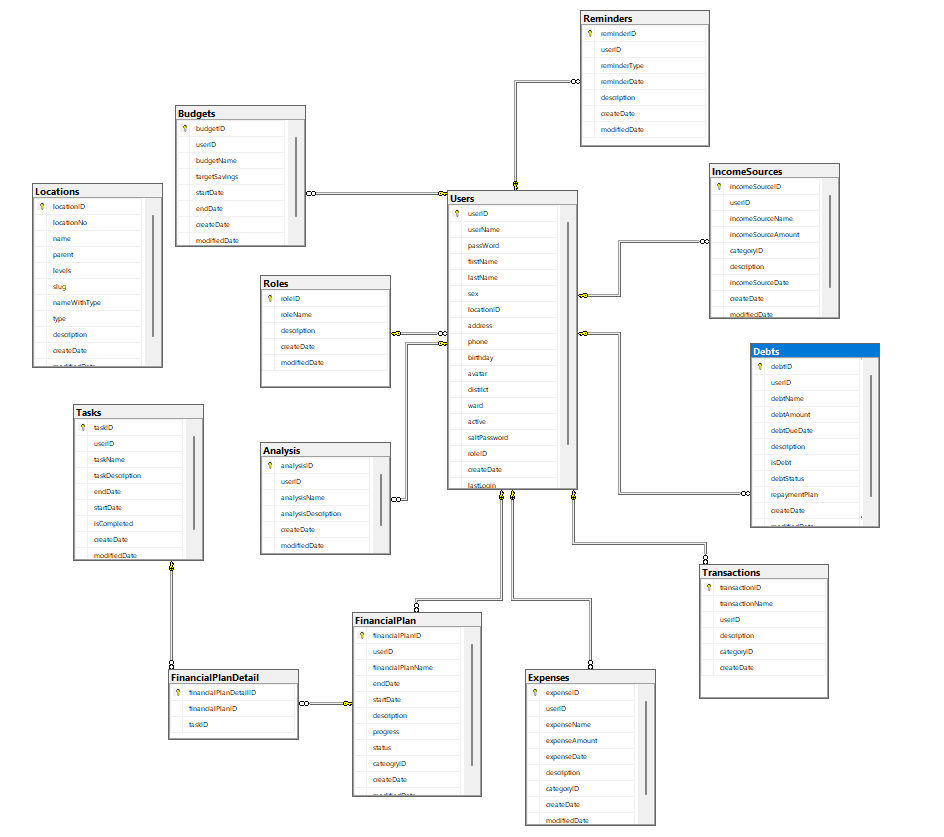
#### 2.3.5.31 Tìm kiếm vay mượn cho admin

#### 2.3.5.32 Tìm kiếm giao dịch cho admin

## 2.2 Mô hình hóa quy trình nghiệp vụ

## 2.3 Mô hình hóa cơ sở dữ liệu

### 2.3.1 Thiết kế cơ sở dữ liệu



*Hình 2.22 Cơ sở dữ liệu hệ thống*

### 2.3.2 Ý nghĩa các bảng

#### 2.3.2.1 Bảng Debts

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên trường** | **Ý nghĩa** | **Kiểu dữ liệu** | **Khóa** |
| 1 | debtID | ID nợ | int | Khóa chính |
| 2 | userID | ID người dùng | int | Khóa ngoại |
| 3 | debtName | Tên nợ | nvarchar(100) |  |
| 4 | debtAmount | Số tiền vay mượn | decimal(18, 2) |  |
| 5 | debtDueDate | Ngày phát sinh | date |  |
| 6 | description | Mô tả vay mượn | nvarchar(max) |  |
| 7 | isDebt | Là vay hay mượn | bit |  |
| 8 | debtStatus | Trạng thái vay mượn | int |  |
| 10 | createDate | Ngày tạo | datetime |  |
| 11 | modifiedDate | Ngày sửa đổi | datetime |  |

#### 2.3.2.2 Bảng Expenses

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên trường** | **Ý nghĩa** | **Kiểu dữ liệu** | **Khóa** |
| 1 | expenseID | ID chi tiêu | int | Primary Key |
| 2 | userID | ID người dùng | int |  |
| 3 | expenseName | Tên chi tiêu | nvarchar(100) |  |
| 4 | expenseAmount | Số tiền chi tiêu | decimal(18, 2) |  |
| 5 | expenseDate | Ngày chi tiêu | date |  |
| 6 | description | Mô tả chi tiêu | nvarchar(max) |  |
| 7 | categoryID | ID danh mục | int |  |
| 8 | createDate | Ngày tạo | datetime |  |
| 9 | modifiedDate | Ngày sửa đổi | datetime |  |

#### 2.3.2.3 Bảng IncomeSources

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên trường** | **Ý nghĩa** | **Kiểu dữ liệu** | **Khóa** |
| 1 | incomeSourceID | ID nguồn thu nhập | int | Primary Key |
| 2 | userID | ID người dùng | int |  |
| 3 | incomeSourceName | Tên nguồn thu nhập | nvarchar(100) |  |
| 4 | incomeSourceAmount | Số tiền thu nhập | decimal(18, 2) |  |
| 5 | categoryID | ID danh mục | int |  |
| 6 | description | Mô tả nguồn thu nhập | nvarchar(max) |  |
| 7 | incomeSourceDate | Ngày thu nhập | datetime |  |
| 8 | createDate | Ngày tạo | datetime |  |
| 9 | modifiedDate | Ngày sửa đổi | datetime |  |

#### 2.3.2.4 Bảng FinancialPlans

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên trường** | **Ý nghĩa** | **Kiểu dữ liệu** | **Khóa** |
| 1 | financialPlanID | ID kế hoạch tài chính | int | Primary Key |
| 2 | userID | ID người dùng | int |  |
| 3 | financialPlanName | Tên kế hoạch tài chính | nvarchar(100) |  |
| 4 | endDate | Ngày kết thúc | date |  |
| 5 | startDate | Ngày bắt đầu | datetime |  |
| 6 | description | Mô tả kế hoạch tài chính | nvarchar(max) |  |
| 7 | progress | Tiến độ | decimal(18, 2) |  |
| 8 | status | Trạng thái | int |  |
| 9 | cateogryID | ID danh mục | int |  |
| 10 | createDate | Ngày tạo | datetime |  |
| 11 | modifiedDate | Ngày sửa đổi | datetime |  |

#### 2.3.2.5 Bảng FinancialPlanDetail

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên trường** | **Ý nghĩa** | **Kiểu dữ liệu** | **Khóa** |
| 1 | financialPlanDetailID | ID chi tiết kế hoạch tài chính | int | Primary Key |
| 2 | financialPlanID | ID kế hoạch tài chính | int | Foreign Key |
| 3 | taskID | ID công việc | int | Foreign Key |

#### 2.3.2.6 Bảng Tasks

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên trường** | **Ý nghĩa** | **Kiểu dữ liệu** | **Khóa** |
| 1 | taskID | ID công việc | int | Primary Key |
| 2 | userID | ID người dùng | int |  |
| 3 | taskName | Tên công việc | nvarchar(255) |  |
| 4 | taskDescription | Mô tả công việc | nvarchar(max) |  |
| 5 | endDate | Ngày kết thúc công việc | datetime |  |
| 6 | startDate | Ngày bắt đầu công việc | datetime |  |
| 7 | isCompleted | Đã hoàn thành hay chưa | bit |  |
| 8 | createDate | Ngày tạo | datetime |  |
| 9 | modifiedDate | Ngày sửa đổi | datetime |  |

#### 2.3.2.7 Bảng Locations

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên trường** | **Ý nghĩa** | **Kiểu dữ liệu** | **Khóa** |
| 1 | locationID | ID địa điểm | int | Primary Key |
| 2 | locationNo | Số địa điểm | nchar(255) |  |
| 3 | name | Tên địa điểm | nvarchar(50) |  |
| 4 | parent | Địa điểm cha | nchar(255) |  |
| 5 | levels | Cấp độ | int |  |
| 6 | slug | Đường dẫn | nvarchar(100) |  |
| 7 | nameWithType | Tên với loại | nvarchar(100) |  |
| 8 | type | Loại địa điểm | int |  |
| 9 | description | Mô tả địa điểm | nvarchar(max) |  |
| 10 | createDate | Ngày tạo | datetime |  |
| 11 | modifiedDate | Ngày sửa đổi | datetime |  |

#### 2.3.2.8 Bảng Roles

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên trường** | **Ý nghĩa** | **Kiểu dữ liệu** | **Khóa** |
| 1 | roleID | ID vai trò | int | Primary Key |
| 2 | roleName | Tên vai trò | nvarchar(50) |  |
| 3 | description | Mô tả vai trò | nvarchar(max) |  |
| 4 | createDate | Ngày tạo | datetime |  |
| 5 | modifiedDate | Ngày sửa đổi | datetime |  |

#### 2.3.2.9 Bảng Transactions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên trường** | **Ý nghĩa** | **Kiểu dữ liệu** | **Khóa** |
| 1 | transactionID | ID giao dịch | int | Primary Key |
| 2 | transactionName | Tên giao dịch | nvarchar(255) |  |
| 3 | userID | ID người dùng | int |  |
| 4 | description | Mô tả giao dịch | nvarchar(max) |  |
| 5 | categoryID | ID danh mục | int |  |
| 6 | createDate | Ngày tạo | datetime |  |

#### 2.3.2.10 Bảng Users

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên trường** | **Ý nghĩa** | **Kiểu dữ liệu** | **Khóa** |
| 1 | userID | ID người dùng | int | Primary Key |
| 2 | userName | Tên đăng nhập | nvarchar(50) |  |
| 3 | passWord | Mật khẩu | nvarchar(255) |  |
| 4 | firstName | Tên | nvarchar(50) |  |
| 5 | lastName | Họ | nvarchar(50) |  |
| 6 | sex | Giới tính | nvarchar(10) |  |
| 7 | locationID | ID địa điểm | nchar(255) |  |
| 8 | address | Địa chỉ | nvarchar(255) |  |
| 9 | phone | Số điện thoại | nchar(15) |  |
| 10 | birthday | Ngày sinh | datetime |  |
| 11 | avatar | Avatar | nvarchar(max) |  |
| 12 | district | Quận/Huyện | nchar(255) |  |
| 13 | ward | Phường/Xã | nchar(255) |  |
| 14 | active | Trạng thái hoạt động | bit |  |
| 15 | saltPassword | Salt Mật khẩu | nchar(255) |  |
| 16 | roleID | ID vai trò | int |  |
| 17 | createDate | Ngày tạo | datetime |  |
| 18 | lastLogin | Lần đăng nhập cuối cùng | datetime |  |
| 19 | modifiedDate | Ngày sửa đổi | datetime |  |

#### 2.3.2.11 Bảng Reminders

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên trường** | **Ý nghĩa** | **Kiểu dữ liệu** | **Khóa** |
| 1 | reminderID | ID thông báo | int | Primary Key |
| 2 | userID | ID người dùng | int |  |
| 3 | reminderType | Loại thông báo | nvarchar(50) |  |
| 4 | reminderDate | Ngày thông báo | datetime |  |
| 5 | description | Mô tả thông báo | nvarchar(max) |  |
| 6 | createDate | Ngày tạo | datetime |  |
| 7 | modifiedDate | Ngày sửa đổi | datetime |  |

#### 2.3.2.12 Bảng Analysis

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên trường** | **Ý nghĩa** | **Kiểu dữ liệu** | **Khóa** |
| 1 | analysisID | ID phân tích | int | Primary Key |
| 2 | userID | ID người dùng | int |  |
| 3 | analysisName | Tên phân tích | nvarchar(100) |  |
| 4 | analysisDescription | Mô tả phân tích | nvarchar(max) |  |
| 5 | createDate | Ngày tạo | datetime |  |
| 6 | modifiedDate | Ngày sửa đổi | datetime |  |

#### 2.3.2.13 Bảng Budgets

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên trường** | **Ý nghĩa** | **Kiểu dữ liệu** | **Khóa** |
| 1 | budgetID | ID ngân sách | int | Primary Key |
| 2 | userID | ID người dùng | int |  |
| 3 | budgetName | Tên ngân sách | nvarchar(100) |  |
| 4 | targetSavings | Mục tiêu tiết kiệm | decimal(18, 2) |  |
| 5 | startDate | Ngày bắt đầu | datetime |  |
| 6 | endDate | Ngày kết thúc | datetime |  |
| 7 | createDate | Ngày tạo | datetime |  |
| 8 | modifiedDate | Ngày sửa đổi | datetime |  |

## 2.4 Thiết kế giao diện hệ thống

### 2.5.1 Giao diện dành cho sinh viên

### 2.5.2 Giao diện dành cho admin

## 2.5 Cài đặt phần mềm lập trình và công nghệ sử dụng

### 2.5.1 Công cụ lập trình

*+* Cài đặt phần mềm

* Visual Studio 2022 - Enterprise
* SQL Server 2022

+ Yêu cầu hệ thống

* Đối với Visual Studio 2022
  + Yêu cầu bộ xử lý 2,6 GHz hoặc nhanh hơn
  + Yêu cầu tối thiểu RAM 8GB trở lên
  + Yêu cầu 10GB dung lượng đĩa cứng trở lên
* Đối với Sql Server 2022
  + Yêu cầu bộ xử lý tối thiểu 1,4 GHz hoặc nhanh hơn
  + Hệ điều hành win 10 TH1 1507 Trở lên
  + Yêu cầu tối thiểu RAM 8GB trở lên
  + Yêu cầu 10GB dung lượng đĩa cứng trở lên

### 2.5.1 Công nghệ sử dụng

* Ngôn ngữ lập trình C# 10.0, Linq.
* Hệ quản trị cơ sở dữ liệu SQL Server 2022.
* Framework Entity Framework.

# CHƯƠNG 3: TỔNG QUAN VỀ WINFORM

## 3.1 Giới thiệu về .NET Framework

### 3.1.1 Định nghĩa về .NET Framework

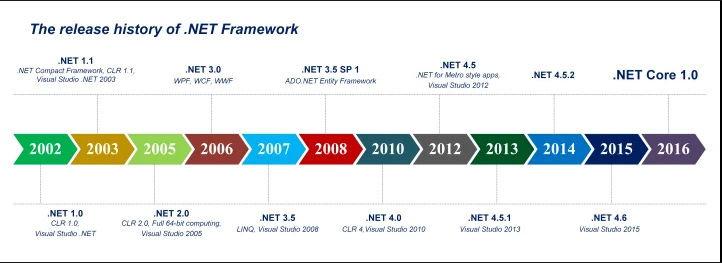
.NET Framework là một nền tảng phần mềm của Microsoft, được thiết kế để xây dựng và triển khai ứng dụng Windows đa dạng. Nó cung cấp một môi trường thực thi cho các ngôn ngữ lập trình khác nhau, cho phép chúng tương tác và sử dụng chung các thư viện và dịch vụ. Mục tiêu của .NET Framework là hỗ trợ phát triển ứng dụng linh hoạt, bảo mật, và có khả năng tương thích cao trên nền tảng Windows.

.NET Framework bao gồm hai thành phần chính: Common Language Runtime (CLR) và .NET Framework Class Library (FCL). CLR quản lý quá trình thực thi mã và cung cấp các tính năng như quản lý bộ nhớ, thu gom rác, và an ninh mã. FCL chứa một bộ thư viện các lớp, giao diện, và công cụ hỗ trợ để phát triển ứng dụng.

Mục tiêu của .NET Framework là tạo ra một môi trường phát triển mạnh mẽ và linh hoạt, giúp lập trình viên xây dựng ứng dụng Windows hiệu quả, bảo mật và dễ bảo trì. Sự tích hợp chặt chẽ giữa các thành phần như CLR và FCL mang lại sự thuận tiện trong quá trình phát triển và triển khai ứng dụng.

### 3.1.2 Lịch sử của .NET Framework

.NET Framework được giới thiệu lần đầu tiên vào năm 2002 với phiên bản 1.0 và nhanh chóng trở thành một trong những nền tảng phổ biến nhất cho phát triển ứng dụng Windows.

Lịch sử các phiên bản và kế hoạch phát triển .NET Framework:  


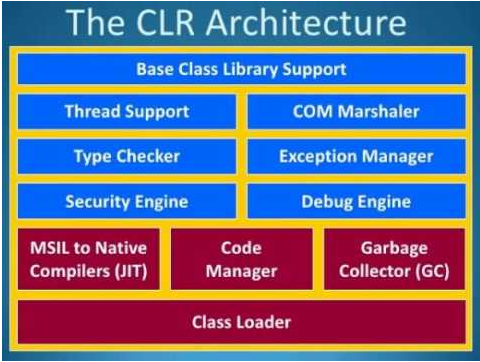
*Hình 3.1 Phiên bản phát hành của .NET Framework*

* .NET Framework 1.0 (2002): Ra mắt với mục tiêu hỗ trợ phát triển ứng dụng Windows Forms và ASP.NET.
* .NET Framework 2.0 (2005): Bổ sung nhiều cải tiến như hỗ trợ 64-bit, Windows Presentation Foundation (WPF), Windows Communication Foundation (WCF), và Windows Workflow Foundation (WWF).
* .NET Framework 3.0 (2006): Không thay đổi CLR nhưng bổ sung các thư viện mới như WPF, WCF, và WWF.
* .NET Framework 3.5 (2008): Bổ sung nhiều tính năng mới, bao gồm LINQ (Language Integrated Query) và ASP.NET AJAX.
* .NET Framework 4.0 (2010): Cung cấp nhiều cải tiến trong CLR và thêm hỗ trợ đặc tính động (dynamic).
* .NET Framework 4.5 (2012): Đưa ra nhiều cải tiến như Async/Await, thêm tích hợp với Windows 8.

### 3.1.3 Thành phần cơ bản của .NET Framework

Về cơ bản .Net Framework được chia làm 2 phần là:

* *Common Language Runtime (CLR)*: Bộ thực thi ngôn ngữ chung.
* .Net Framework Class Library



*Hình 3.2 Kiến trúc của Common Language Runtime (CLR)*

* **Common Language Runtime (CLR)**

CLR là một bộ thực thi đa ngôn ngữ cho phép các ứng dụng được viết bằng nhiều ngôn ngữ lập trình khác nhau. CLR thực hiện các nhiệm vụ sau:

* Biên dịch mã nguồn thành mã máy gốc
* Quản lý bộ nhớ
* Quản lý an ninh
* Quản lý lỗi
* **NET Assemblies**

Assembly là đơn vị cơ bản của ứng dụng .NET. Nó là một tập hợp các mã đã biên dịch, tài nguyên và siêu dữ liệu. Các assembly thường được đóng gói dưới dạng các tệp có phần mở rộng .dll hoặc .exe.

Assemblies được CLR sử dụng để tải và thực thi mã được quản lý. CLR sử dụng siêu dữ liệu trong assembly để xác định các loại lớp và phương thức được định nghĩa trong assembly. CLR cũng sử dụng siêu dữ liệu để xác định các quyền bảo mật cần thiết để truy cập các lớp và phương thức trong assembly.

* **Common Type System (CTS)**

CTS là một tập hợp các quy tắc định nghĩa các loại dữ liệu có thể được sử dụng trong .NET Framework. CTS quy định các quy tắc cho các nội dung sau:

* Các kiểu dữ liệu
* Các toán tử
* Kế thừa
* Giao diện
* Thuộc tính

CTS rất quan trọng vì nó đảm bảo rằng tất cả mã được quản lý có thể tương tác với nhau. Điều này có nghĩa là các lớp và phương thức từ các assembly khác nhau có thể được sử dụng cùng nhau mà không gặp bất kỳ vấn đề gì.

* **Common Language Specification (CLS)**

CLS là một tập hợp các hướng dẫn định nghĩa các yêu cầu tối thiểu đối với một ngôn ngữ lập trình để tương thích với .NET Framework. CLS quy định các nội dung sau:

* Tập hợp các kiểu dữ liệu được hỗ trợ
* Tập hợp các toán tử được hỗ trợ
* Tập hợp các câu lệnh điều khiển được hỗ trợ
* Tập hợp các cơ chế xử lý ngoại lệ được hỗ trợ

CLS rất quan trọng vì nó cho phép các ngôn ngữ lập trình được sử dụng để phát triển ứng dụng .NET mà không cần phải lo lắng về việc ngôn ngữ có tương thích với .NET Framework hay không.

* **Microsoft Intermediate Language (MSIL)**

MSIL là một ngôn ngữ cấp thấp được sử dụng để biểu diễn mã .NET đã biên dịch. MSIL không thể được thực thi trực tiếp bởi CLR. Thay vào đó, CLR phải biên dịch MSIL thành mã máy có thể được thực thi bởi bộ xử lý.

MSIL rất quan trọng vì nó cung cấp một biểu diễn trung gian di động của mã .NET. Điều này có nghĩa là mã MSIL có thể được biên dịch thành mã máy cho bất kỳ nền tảng nào hỗ trợ CLR.

* **IL: Intermediate Language**

IL hoặc Intermediate Language là một thuật ngữ khác cho Microsoft Intermediate Language (MSIL).

* **CIL: Common Intermediate Language**

CIL hoặc Common Intermediate Language là một thuật ngữ khác cho Microsoft Intermediate Language (MSIL).

* **JIT – Just-In-Time**

Biên dịch Just-in-time (JIT) là một kỹ thuật được sử dụng để cải thiện hiệu suất của ứng dụng .NET. Khi một phương thức được gọi lần đầu tiên, CLR JIT biên dịch mã MSIL cho phương thức thành mã máy. Mã máy này sau đó được lưu trữ trong bộ đệm của CLR. Khi phương thức được gọi lại, CLR có thể thực thi mã máy từ bộ đệm thay vì phải biên dịch lại mã MSIL.

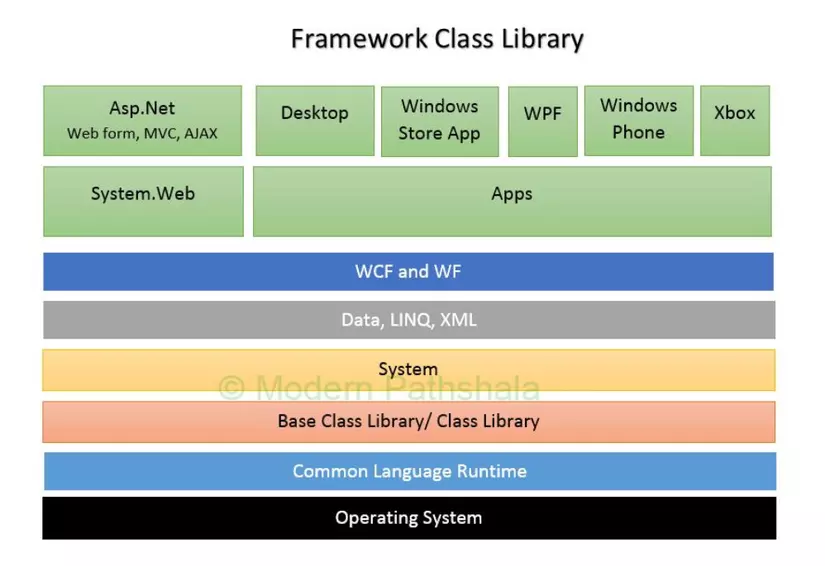
Biên dịch JIT có thể cải thiện đáng kể hiệu suất của ứng dụng .NET vì nó loại bỏ nhu cầu giải thích mã MSIL mỗi khi một phương thức được gọi.

* **Managed Code**

Mã được quản lý là mã được thực thi bởi CLR. Mã được quản lý chịu sự điều chỉnh của các quy tắc bảo mật và loại hệ thống của CLR. Điều này có nghĩa là mã được quản lý an toàn và đáng tin cậy hơn mã không được quản lý.

* **Garbage Collection (GC): Bộ phận thu gom rác**

Thu gom rác (GC) là một quá trình tự động thu hồi bộ



*Hình 3.3 Kiến trúc của Common Language Runtime (CLR)*

Kiến trúc của NET Framework Class Library (FCL) được chia thành hai thành phần chính:

* **Base Class Library (BCL):**

BCL là một thư viện lớp cơ sở cung cấp các lớp và phương thức chung cho các ứng dụng .NET. BCL bao gồm các lớp và phương thức cho các nhiệm vụ như:

* + Quản lý bộ nhớ
  + Quản lý chuỗi
  + Quản lý tệp
  + Quản lý thời gian
  + Quản lý lỗi
  + Quản lý mạng
  + Quản lý giao diện người dùng
  + Quản lý dữ liệu
  + Quản lý tài nguyên
  + Quản lý bảo mật
* **Kiến trúc của Base Class Library (BCL)**

BCL được tổ chức thành các namespace, mỗi namespace đại diện cho một tập hợp các lớp và phương thức liên quan. Các namespace phổ biến nhất trong BCL bao gồm:

* System: Namespace này cung cấp các lớp và phương thức cơ bản nhất cho các ứng dụng .NET, chẳng hạn như các lớp cho chuỗi, tệp, thời gian, lỗi, và bộ nhớ.
* System.IO: Namespace này cung cấp các lớp và phương thức cho việc truy cập và xử lý tệp.
* System.Collections: Namespace này cung cấp các lớp và phương thức cho việc quản lý danh sách, danh sách liên kết, và bộ.
* System.Text: Namespace này cung cấp các lớp và phương thức cho việc thao tác với chuỗi.
* System.Threading: Namespace này cung cấp các lớp và phương thức cho việc lập trình đồng thời.
* System.Data: Namespace này cung cấp các lớp và phương thức cho việc truy cập cơ sở dữ liệu.
* System.Xml: Namespace này cung cấp các lớp và phương thức cho việc thao tác với XML.
* **Framework Class Library (FCL)**

FCL là một thư viện lớp mở rộng cung cấp các lớp và phương thức bổ sung cho các ứng dụng .NET. FCL bao gồm các lớp và phương thức cho các nhiệm vụ như:

* + Giao tiếp dịch vụ web
  + Đồ họa
  + Mạng
  + Đa phương tiện
  + Cơ sở dữ liệu
  + Thương mại điện tử
* **Kiến trúc của Framework Class Library (FCL)**

FCL được tổ chức thành các namespace, mỗi namespace đại diện cho một tập hợp các lớp và phương thức liên quan. Các namespace phổ biến nhất trong FCL bao gồm:

* System.Web: Namespace này cung cấp các lớp và phương thức cho việc phát triển ứng dụng web.
* System.Net: Namespace này cung cấp các lớp và phương thức cho việc truy cập mạng.
* System.Drawing: Namespace này cung cấp các lớp và phương thức cho việc tạo đồ họa.
* System.Media: Namespace này cung cấp các lớp và phương thức cho việc phát đa phương tiện.
* System.Data.Entity: Namespace này cung cấp các lớp và phương thức cho việc truy cập cơ sở dữ liệu quan hệ.
* System.ServiceModel: Namespace này cung cấp các lớp và phương thức cho việc phát triển dịch vụ web.

Lợi ích của việc sử dụng .NET Framework Class Library (FCL)

* FCL cung cấp một tập hợp các lớp và phương thức phong phú, giúp lập trình viên tiết kiệm thời gian và công sức trong quá trình phát triển ứng dụng.
* FCL được thiết kế để dễ sử dụng và học hỏi, giúp lập trình viên có thể nhanh chóng bắt đầu sử dụng.
* FCL được hỗ trợ bởi Microsoft và cộng đồng .NET, đảm bảo tính ổn định và bảo mật.

### 3.1.4 Đặc điểm .NET Framework

.NET Framework là một nền tảng phát triển phần mềm và cũng là một nền tảng thực thi ứng dụng chủ yếu trên hệ điều hành Microsoft Windows. Nền tảng này bao gồm một bộ các thành phần và công cụ để lập trình viên có thể sử dụng để tạo ra các ứng dụng đa nền tảng, hiệu suất cao và an toàn.

Các đặc điểm chính của .NET Framework bao gồm:

* **Đa nền tảng:** .NET Framework hỗ trợ các ứng dụng được viết bằng nhiều ngôn ngữ lập trình khác nhau, bao gồm C#, Visual Basic .NET, F#, IronPython và IronRuby. Điều này cho phép các lập trình viên tạo ra các ứng dụng có thể chạy trên nhiều nền tảng khác nhau, bao gồm Windows, macOS, Linux và iOS.
* **Hiệu suất cao:** .NET Framework được thiết kế để cung cấp hiệu suất cao cho các ứng dụng. Nền tảng này sử dụng bộ thực thi đa ngôn ngữ (CLR) để biên dịch mã nguồn thành mã máy gốc. Điều này giúp cho các ứng dụng .NET chạy nhanh hơn các ứng dụng được biên dịch sang bytecode.
* **An toàn:** .NET Framework được tích hợp các tính năng bảo mật để giúp bảo vệ các ứng dụng khỏi các cuộc tấn công bảo mật. Nền tảng này sử dụng các vùng bảo mật (security zones) để kiểm soát quyền truy cập vào tài nguyên hệ thống. Ngoài ra, .NET Framework cũng cung cấp các tính năng như mã hóa và xác thực để giúp bảo vệ dữ liệu nhạy cảm.
* **Tính tương tác**: .NET Framework cung cấp một tập hợp các lớp và phương thức cho việc tương tác giữa các ứng dụng. Điều này cho phép các lập trình viên tạo ra các ứng dụng có thể giao tiếp với nhau một cách dễ dàng.
* **Tính bảo trì:** .NET Framework được thiết kế để dễ dàng bảo trì. Nền tảng này sử dụng bộ thu gom rác (garbage collection) để tự động thu hồi bộ nhớ không còn được sử dụng. Điều này giúp giảm thiểu khối lượng công việc bảo trì cho lập trình viên.

.NET Framework là một nền tảng phát triển phần mềm mạnh mẽ và linh hoạt, cung cấp cho các lập trình viên một bộ các công cụ và thành phần để tạo ra các ứng dụng đa nền tảng, hiệu suất cao và an toàn.

### 3.1.5 Ưu điểm và nhược điểm của .NET Framework

|  |  |
| --- | --- |
| **Ưu Điểm .NET Framework** | **Nhược Điểm .NET Framework** |
| 1. Đa Nền Tảng (Cross-Platform): .NET 5 trở đi (bao gồm .NET Core và .NET 5+) hỗ trợ đa nền tảng, giúp phát triển ứng dụng trên nhiều hệ điều hành (Windows, Linux, macOS). | 1. Khả Năng Tương Thích Ngược (Backward Compatibility): Một số ứng dụng và thư viện có thể không tương thích ngược với các phiên bản mới, đặc biệt là khi chuyển từ .NET Framework cũ sang .NET 5+ hoặc .NET Core. |
| 2. Môi Trường Phát Triển Linh Hoạt: Hỗ trợ nhiều ngôn ngữ lập trình (C#, VB.NET, F#) và tích hợp với nhiều công cụ phát triển như Visual Studio. | 2. Kích Thước Lớn: Ứng dụng .NET Framework có thể có kích thước lớn do yêu cầu thêm các thành phần runtime và thư viện. |
| 3. Bảo Mật Cao: Cung cấp các tính năng bảo mật như kiểm soát quyền truy cập, quản lý danh tính, và mã hóa dữ liệu. | 3. Khả Năng Di Động (Mobility): .NET Framework truyền thống hạn chế trong việc phát triển ứng dụng di động so với các nền tảng như Android hoặc iOS. |
| 4. Tích Hợp Dễ Dàng với Công Nghệ Microsoft: Tích hợp tốt với các công nghệ khác của Microsoft như Azure, SQL Server, và Windows Server. | 4. Phiên Bản .NET Framework Được Hạn Chế trên Linux và macOS: Một số tính năng có sẵn trên Windows có thể bị hạn chế hoặc không hỗ trợ trên Linux và macOS. |
| 5. Hỗ Trợ Thư Viện Phong Phú (FCL): .NET Framework Class Library (FCL) cung cấp một loạt các thư viện để giảm độ phức tạp và tăng tính tương thích giữa các ứng dụng. | 5. Quản Lý Bộ Nhớ Tự Động: Mặc dù CLR có tự động quản lý bộ nhớ, nhưng điều này có thể dẫn đến hiệu suất kém trong một số trường hợp. |
| 6. Cộng Đồng Lớn và Hỗ Trợ Mạnh Mẽ: .NET có một cộng đồng phát triển lớn, cung cấp tài liệu, diễn đàn, và các nguồn tài nguyên học tốt. | 6. Khả Năng Mở Rộng (Scalability): Trong một số tình huống, .NET Framework có thể không linh hoạt và khả năng mở rộng như một số nền tảng khác. |
| 7. Sự Thống Nhất của Ngôn Ngữ và Thư Viện: Sự thống nhất giữa các ngôn ngữ và thư viện giúp làm tăng năng suất phát triển. | 7. Tính Linh Động trong Việc Lựa Chọn Công Nghệ: Trong một số trường hợp, lựa chọn .NET Framework có thể không phải là lựa chọn linh hoạt nhất cho mọi loại ứng dụng. |

### 3.1.6 Ứng dụng của .NET Framework

NET Framework đã được sử dụng rộng rãi trong nhiều lĩnh vực khác nhau và đã tạo ra nhiều ứng dụng thành công. Dưới đây là một số ứng dụng phổ biến của .NET Framework:

1. Ứng Dụng Windows Forms và WPF:

.NET Framework thường được sử dụng để phát triển ứng dụng desktop Windows bằng cách sử dụng Windows Forms hoặc Windows Presentation Foundation (WPF). Điều này bao gồm các ứng dụng văn phòng, ứng dụng quản lý, và các ứng dụng desktop khác.

2. ASP.NET và Ứng Dụng Web:

ASP.NET, một phần của .NET Framework, được sử dụng để phát triển ứng dụng web và trang web. Các trang web lớn như Microsoft Outlook và Stack Overflow đã sử dụng ASP.NET.

3. Ứng Dụng Doanh Nghiệp và Quản Lý:

.NET Framework thường được sử dụng để phát triển các ứng dụng doanh nghiệp và quản lý, bao gồm quản lý tài khoản, hệ thống quản lý khách hàng (CRM), và các ứng dụng kinh doanh tổng quát.

4. Ứng Dụng Di Động (Xamarin):

Sử dụng Xamarin, một framework tích hợp với .NET, để phát triển ứng dụng di động cho cả Android và iOS. Điều này cung cấp tính tương thích và chia sẻ mã nguồn giữa các nền tảng di động.

5. Dịch Vụ Web và API:

.NET Framework được sử dụng để xây dựng các dịch vụ web và API, giúp kết nối các ứng dụng và hệ thống khác nhau. WCF (Windows Communication Foundation) thường được sử dụng cho mục đích này.

6. Game và Giải Trí Số:

.NET Framework được sử dụng trong việc phát triển các trò chơi và ứng dụng giải trí số. Unity, một platform phổ biến để phát triển trò chơi, hỗ trợ sử dụng C# và .NET.

7. Dự Án IoT (Internet of Things):

.NET Framework được tích hợp trong dự án IoT, giúp phát triển các ứng dụng điều khiển và quản lý thiết bị thông minh.

8. Ứng Dụng Khoa Học và Nghiên Cứu:

Trong lĩnh vực nghiên cứu và khoa học, .NET Framework được sử dụng để xây dựng các ứng dụng tính toán khoa học và mô phỏng.

9. Công Cụ Phân Tích Dữ Liệu và Business Intelligence:

.NET Framework được tích hợp trong các công cụ và ứng dụng phân tích dữ liệu và Business Intelligence để xử lý, phân tích, và hiển thị thông tin kinh doanh.

10. Ứng Dụng Cloud (Azure):

.NET Framework được tích hợp tốt với Microsoft Azure, nền tảng đám mây của Microsoft, để phát triển và triển khai các ứng dụng đám mây.

Một số ví dụ cụ thể về các ứng dụng sử dụng .NET Framework:

* **Ứng dụng Web:** Microsoft Office 365, Amazon Web Services, eBay, Facebook ,Twitter, Ứng dụng di động, Microsoft Dynamics 365, Skype, LinkedIn,…
* **Ứng dụng máy tính để bàn:** Microsoft Visual Studio, Microsoft Office, Adobe Photoshop, Google Chrome, VLC Media Player,…
* **Ứng dụng nhúng:** Máy ATM, Máy bán hàng tự động, Tivi thông minh,Thiết bị IoT,..

.NET Framework là một nền tảng phát triển phần mềm phổ biến và mạnh mẽ, được sử dụng bởi nhiều công ty và tổ chức lớn trên thế giới.

## 3.2. Giới thiệu về Winform

### 3.2.1 Tổng quan về Winform

Winform là một công nghệ lập trình GUI (Graphical User Interface) được Microsoft phát triển cho các ứng dụng Windows. Winform dựa trên các thành phần đồ họa Windows gốc, chẳng hạn như nút, hộp kiểm, hộp văn bản, v.v., và cung cấp các lớp và phương thức để tạo và quản lý các thành phần này.

Windows Forms (WinForms) là một phần của .NET Framework, được sử dụng để phát triển ứng dụng desktop chạy trên hệ điều hành Windows. WinForms cung cấp một framework để tạo giao diện người dùng (GUI) cho các ứng dụng Windows bằng cách sử dụng các thành phần như forms (cửa sổ), controls (điều khiển), và sự kiện.

Điểm đặc biệt của WinForms là sự đơn giản và dễ sử dụng. Nó được giới thiệu từ .NET Framework 1.0 và trở thành một phương tiện phổ biến để phát triển các ứng dụng Windows truyền thống. WinForms cung cấp một giao diện lập trình ứng dụng (API) linh hoạt cho việc tạo và quản lý các controls trên các forms, xử lý sự kiện người dùng, và thực hiện logic ứng dụng.

Điểm chính về WinForms:

1. **Visual Studio:** Thường được phát triển trong môi trường phát triển Visual Studio, nơi lập trình viên có thể thiết kế giao diện người dùng bằng cách kéo và thả controls.
2. **Event-Driven Programming:** Sự kiện đóng vai trò quan trọng trong WinForms. Khi người dùng tương tác với controls (như nhấp chuột, nhập từ bàn phím), các sự kiện được kích hoạt và lập trình viên có thể xử lý chúng.
3. **Controls và Forms:** WinForms cung cấp một loạt các controls như TextBox, Button, ComboBox, và nhiều controls khác, cũng như forms để chứa chúng.
4. **Phong Cách Truyền Thống:** WinForms thường được sử dụng để phát triển các ứng dụng có giao diện người dùng truyền thống dựa trên cửa sổ và controls.
5. **Tích Hợp với .NET Framework:** WinForms tận dụng các tính năng khác của .NET Framework như quản lý bộ nhớ tự động, thư viện lớp và khả năng tương thích ngôn ngữ

### 3.2.2 Kiến trúc của Winform

Kiến trúc Winform được xây dựng trên kiến trúc thành phần, bao gồm các thành phần sau:

* Form: Là lớp cơ bản của Winform, đại diện cho một cửa sổ ứng dụng.
* Control: Là lớp đại diện cho một thành phần đồ họa, chẳng hạn như nút, hộp kiểm, hộp văn bản,....
* Event: Là sự kiện xảy ra khi một thành phần đồ họa được kích hoạt, chẳng hạn như khi người dùng nhấp vào một nút.
* **Form**

Form là lớp cơ bản của Winform, đại diện cho một cửa sổ ứng dụng. Một ứng dụng Winform có thể có nhiều Form, mỗi Form đại diện cho một cửa sổ riêng biệt.

Form cung cấp các thuộc tính và phương thức để quản lý cửa sổ ứng dụng, bao gồm:

* Title: Tiêu đề cửa sổ
* Size: Kích thước cửa sổ
* Location: Vị trí cửa sổ
* Show: Hiển thị cửa sổ
* Hide: Ẩn cửa sổ
* Close: Đóng cửa sổ
* **Control**

Control là lớp đại diện cho một thành phần đồ họa, chẳng hạn như nút, hộp kiểm, hộp văn bản, v.v. Một Form có thể chứa nhiều Control.

Control cung cấp các thuộc tính và phương thức để quản lý thành phần đồ họa, bao gồm:

* Text: Nội dung của thành phần
* Size: Kích thước của thành phần
* Location: Vị trí của thành phần
* Enabled: Trạng thái của thành phần
* Visible: Trạng thái hiển thị của thành phần
* OnClick: Sự kiện xảy ra khi người dùng nhấp vào thành phần
* **Event**

Event là sự kiện xảy ra khi một thành phần đồ họa được kích hoạt, chẳng hạn như khi người dùng nhấp vào một nút.

Các sự kiện của Winform được chia thành hai loại:

* System event: Là các sự kiện được kích hoạt bởi hệ thống, chẳng hạn như sự kiện OnClick của nút.
* User event: Là các sự kiện được kích hoạt bởi người dùng, chẳng hạn như sự kiện KeyPress của hộp văn bản.

### 3.2.3 Cách thức hoạt động của Winform

* **Cách thức hoạt động của WinForms**

WinForms là một công nghệ lập trình GUI (Graphical User Interface) được Microsoft phát triển cho các ứng dụng Windows. WinForms dựa trên các thành phần đồ họa Windows gốc, chẳng hạn như nút, hộp kiểm, hộp văn bản, v.v., và cung cấp các lớp và phương thức để tạo và quản lý các thành phần này.

* **Quá trình tạo và hiển thị form**
* Khi ứng dụng WinForms được khởi động, form chính của ứng dụng sẽ được tạo và hiển thị. Form được tạo ra bằng cách tạo một đối tượng của lớp Form. Phương thức Main() của ứng dụng thường sẽ tạo và hiển thị form chính này.
* Form có thể được hiển thị bằng cách sử dụng phương thức Show() hoặc ShowDialog(). Phương thức Show() là phương thức không chặn, trong khi phương thức ShowDialog() là phương thức chặn, làm cho form hiển thị như một hộp thoại và chặn các hoạt động khác cho đến khi form được đóng.
* **Quản lý controls**
* Lập trình viên thêm controls vào form bằng cách sử dụng trình thiết kế hoặc thông qua mã nguồn. Các controls như nút, hộp kiểm, hộp văn bản, v.v., được thêm vào form dựa trên yêu cầu thiết kế giao diện.
* Form chủ yếu được xem như một container chứa các controls. Containers như Panel, GroupBox được sử dụng để nhóm và quản lý các controls.
* **Xử lý sự kiện**
* Khi người dùng tương tác với controls, các sự kiện sẽ được kích hoạt. Lập trình viên đăng ký các sự kiện cho các controls bằng cách liên kết một hoặc nhiều phương thức xử lý với các sự kiện như Click, TextChanged.
* Khi sự kiện xảy ra, các phương thức đã được đăng ký sẽ được gọi để xử lý sự kiện đó.
* WinForms sử dụng mô hình đẩy sự kiện, nghĩa là khi sự kiện xảy ra, nó sẽ đẩy thông báo đến các phương thức đăng ký để xử lý.
* **Hiển thị và vẽ giao diện người dùng**
* WinForms sử dụng GDI+ (Graphics Device Interface) để vẽ và hiển thị các yếu tố đồ họa như hình ảnh, văn bản, và controls. Điều này cho phép tùy chỉnh giao diện người dùng thông qua việc vẽ các controls và hình ảnh.
* Các controls có khả năng tự động refresh khi trạng thái thay đổi, hoặc có thể được buộc phải refresh thông qua phương thức Invalidate().
* **Điều khiển luồng**
* Giao diện người dùng (UI) chạy trên một luồng riêng biệt gọi là UI thread. UI thread chịu trách nhiệm cho việc vẽ và hiển thị giao diện người dùng.
* Các nhiệm vụ nền có thể chạy trên các background thread để tránh làm đơ giao diện khi thực hiện các tác vụ nặng.
* **Life cycle của form**
* Sự kiện Load được kích hoạt khi form được load. Sự kiện Closing được kích hoạt khi form đang được đóng.
* Khi form đóng, nó có thể được giải phóng bộ nhớ thông qua phương thức Dispose().
* **Interactions và data handling**
* WinForms hỗ trợ data binding để liên kết giữa controls và dữ liệu. Các controls có thể tự động cập nhật khi dữ liệu thay đổi.
* WinForms thường được sử dụng để tương tác với cơ sở dữ liệu, thực hiện các thao tác như đọc, ghi, và cập nhật dữ liệu.

### 3.2.4 Ưu điểm và nhược điểm của Winform

|  |  |
| --- | --- |
| **Ưu Điểm của WinForms** | **Nhược Điểm của WinForms** |
| 1. Dễ Học và Sử Dụng: WinForms có ngưỡng đầu vào thấp, rất dễ học và sử dụng cho người mới học lập trình. | 1. Giao Diện Người Dùng Truyền Thống: Giao diện người dùng của WinForms không cung cấp tính tương tác và đa dạng như các công nghệ giao diện người dùng hiện đại khác như WPF. |
| 2. Tương Thích Ngược (Backward Compatibility): WinForms ổn định và tương thích ngược, tức là các ứng dụng được xây dựng trên các phiên bản .NET Framework cũ vẫn chạy trên các phiên bản mới hơn. | 2. Khả Năng Mở Rộng Hạn Chế: Đối với các ứng dụng lớn và phức tạp, khả năng mở rộng của WinForms có thể bị hạn chế so với các công nghệ khác như WPF. |
| 3. Visual Studio Integration: Được tích hợp mạnh mẽ với Visual Studio, giúp lập trình viên dễ dàng thiết kế và phát triển giao diện người dùng. | 3. Giao Diện Người Dùng Ít Linh Hoạt: WinForms không cung cấp tính linh hoạt lớn trong việc thiết kế giao diện người dùng so với các công nghệ khác như WPF. |
| 4. Dung Lượng Nhỏ Gọn: Các ứng dụng WinForms thường có kích thước nhỏ gọn và tốn ít tài nguyên hệ thống. | 4. Không Hỗ Trợ Đa Nhiệm Tốt: WinForms có thể gặp khó khăn khi xử lý các tác vụ đa nhiệm và đồng thời. |
| 5. Phổ Biến và Stabil: WinForms đã được sử dụng rộng rãi trong thời gian dài, và nhiều dự án và ứng dụng vẫn sử dụng nó, chứng minh tính ổn định và phổ biến của nó. | 5. Giao Diện Người Dùng Lạc Hậu: Giao diện người dùng của WinForms không mang tính hiện đại và thường được coi là lạc hậu so với các công nghệ mới hơn. |
| 6. Đội Ngũ Phát Triển: Có nhiều lập trình viên có kinh nghiệm với WinForms, điều này có nghĩa là có nhiều nguồn lực và hỗ trợ trên cộng đồng. | 6. Không Đa Nền Tảng: WinForms chủ yếu hỗ trợ môi trường Windows, không đa nền tảng như các công nghệ gần đây như Xamarin. |

### 3.2.4 Ứng dụng của Winform

Windows Forms (WinForms) thường được sử dụng để phát triển các ứng dụng desktop chạy trên hệ điều hành Windows. Dưới đây là một số ứng dụng phổ biến mà WinForms thường được sử dụng:

1. **Ứng Dụng Quản Lý và Điều Khiển:**
   * **Mô Tả:** Các ứng dụng quản lý kho, quản lý hàng tồn, và các hệ thống điều khiển quy trình.
   * **Ví Dụ:** Hệ thống quản lý kho, ứng dụng theo dõi sản xuất.
2. **Ứng Dụng Văn Phòng và Doanh Nghiệp:**
   * **Mô Tả:** Các ứng dụng văn phòng như xử lý văn bản, bảng tính, và quản lý thông tin doanh nghiệp.
   * **Ví Dụ:** Microsoft Office có thể sử dụng WinForms trong một số ứng dụng của mình.
3. **Công Cụ Quản Lý Dự Án:**
   * **Mô Tả:** Ứng dụng giúp quản lý dự án, theo dõi tiến độ, và tương tác với nhóm làm việc.
   * **Ví Dụ:** Công cụ quản lý dự án nội bộ cho doanh nghiệp.
4. **Ứng Dụng Kiosk và Bán Lẻ:**
   * **Mô Tả:** Các ứng dụng dành cho các thiết bị tự dịch vụ, máy bán hàng, và kiosks.
   * **Ví Dụ:** Hệ thống thanh toán tự động trong cửa hàng.
5. **Ứng Dụng Desktop Nội Bộ:**
   * **Mô Tả:** Ứng dụng chuyên nghiệp phát triển nội bộ cho nhu cầu cụ thể của doanh nghiệp hoặc tổ chức.
   * **Ví Dụ:** Hệ thống quản lý thông tin nội bộ, ứng dụng nội bộ doanh nghiệp.
6. **Công Cụ Hỗ Trợ Quy Trình Kinh Doanh:**
   * **Mô Tả:** Các ứng dụng hỗ trợ quy trình kinh doanh như xử lý đơn hàng, quản lý khách hàng.
   * **Ví Dụ:** Hệ thống quản lý bán hàng và dịch vụ khách hàng.
7. **Ứng Dụng Dữ Liệu Lớn:**
   * **Mô Tả:** Các ứng dụng xử lý và hiển thị dữ liệu lớn mà không yêu cầu tính năng đồ họa và hiệu suất cao.
   * **Ví Dụ:** Công cụ phân tích dữ liệu nội bộ cho doanh nghiệp.
8. **Ứng Dụng Quản Lý Tài Nguyên:**
   * **Mô Tả:** Các ứng dụng giúp quản lý và theo dõi tài nguyên như nhân sự, vật tư.
   * **Ví Dụ:** Hệ thống quản lý nhân sự và tài sản.

# CHƯƠNG 4: XÂY DỰNG PHẦN MỀM QUẢN LÝ TÀI CHO SINH VIÊN

## 4.1 Thiết kế giao phần mềm

### 4.1.1 Giao diện đăng nhập

### 4.1.2 Giao diện đăng ký cho sinh viên

### 4.1.3 Giao diện trang chủ sinh viên

### 4.1.4 Giao diện quản lý vay mượn cho sinh viên

### 4.1.5 Giao diện quản lý nguồn thu cho sinh viên

### 4.1.6 Giao diện quản lý chi tiêu cho sinh viên

### 4.1.7 Giao diện quản lý kế hoạch tài chính cho sinh viên

### 4.1.8 Giao diện quản lý giao dịch cho sinh viên

### 4.1.9 Giao diện cập nhật thông tin tài khoản sinh viên

### 4.1.10 Giao diện thống kê cho admin

### 4.1.11 Giao diện quản lý địa điểm cho admin

### 4.1.12 Giao diện quản lý tài khoản cho admin

### 4.1.13 Giao diện quản lý giao dịch cho admin

### 4.1.14 Giao diện quản lý nguồn thu cho admin

### 4.1.15 Giao diện quản lý chi tiêu cho admin

### 4.1.16 Giao diện quản lý vay mượn cho admin

### 4.1.17 Giao diện quản lý kế hoạch tài chính cho admin

## 4.2 Cài đặt phần mềm

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN TƯƠNG LAI

## 1.1 Kết luận

Đề tài "Tìm hiểu WinForms - Xây dựng ứng dụng quản lý tài chính cho sinh viên" đã đạt được những kết quả đáng khích lệ, góp phần nâng cao kiến thức về WinForms và quản lý tài chính cho sinh viên.

Kết quả đạt được

* Về kiến thức WinForms:
  + Nắm vững kiến thức về WinForms, từ thiết kế giao diện đến xử lý sự kiện và quản lý dữ liệu.
  + Áp dụng kiến thức WinForms để xây dựng ứng dụng quản lý tài chính.
* Về quản lý tài chính:
  + Xây dựng ứng dụng có chức năng quản lý thu chi, theo dõi nguồn thu nhập, và tạo kế hoạch tài chính.
* Về ý nghĩa và ứng dụng:
  + Cung cấp công cụ quản lý tài chính hiệu quả giúp sinh viên kiểm soát tài chính cá nhân và phát triển thói quen tiết kiệm.
  + Mô phỏng một ứng dụng thực tế, giúp người học áp dụng kiến thức vào các dự án thực tế khác.

## 1.2 Hướng phát triển tương lai

Ứng dụng quản lý tài chính cho sinh viên có thể được phát triển và cải tiến thêm trong tương lai theo các hướng sau:

* Nâng cấp giao diện và trải nghiệm người dùng
  + Tối ưu hóa giao diện để đảm bảo trải nghiệm người dùng mượt mà và thuận lợi.
  + Thêm các tính năng tương tác như biểu đồ và thống kê tài chính.
* Kết nối cơ sở dữ liệu và bảo mật
  + Nâng cấp để hỗ trợ lưu trữ dữ liệu, có thể tích hợp với cơ sở dữ liệu để quản lý thông tin cá nhân của người dùng.
  + Cải thiện bảo mật với các tính năng xác thực và mã hóa dữ liệu.
* Tích hợp đa nền tảng và mở rộng tính năng
  + Xem xét khả năng tích hợp với các nền tảng di động để mở rộng sự tiện ích của ứng dụng.
  + Thêm các chức năng mới như quản lý nợ, lập kế hoạch chi tiêu hàng tháng, và nhắc nhở.
* Hỗ trợ và phản hồi
  + Tạo cộng đồng người dùng để họ có thể chia sẻ ý kiến, góp ý, và tạo ra một môi trường phát triển tích cực.
  + Cam kết duy trì và cung cấp hỗ trợ để đảm bảo ứng dụng luôn hoạt động mượt mà.
* Phát triển theo hướng Blockchain và Smart Contracts
  + Tích hợp công nghệ Blockchain để cải thiện tính an toàn và đáng tin cậy của việc quản lý tài chính cá nhân.
  + Sử dụng Smart Contracts để tự động hóa một số quy trình liên quan đến tài chính.
* Tối ưu hóa và đa dạng hóa dịch vụ
  + Tối ưu hóa ứng dụng để đảm bảo hiệu suất cao và phản hồi nhanh chóng.
  + Tích hợp với nhiều dịch vụ khác nhau như ngân hàng trực tuyến và ví điện tử để mở rộng tính năng thanh toán.
* Tích hợp tính năng giáo dục tài chính
  + Thêm các tính năng hướng dẫn và giáo dục về quản lý tài chính để người sử dụng có thể nâng cao kiến thức của mình.

Những ý kiến nêu trên là những hướng phát triển dự án này trong thời gian tới và chúng em rất mong những được những nhận xét và góp ý của các thầy, cô giáo về những ý kiến trên để có thể có được những nhìn nhận, kinh nghiệm về việc phát triển các dự án sau này.

# TÀI LIỆU THAM KHẢO