

**SUMMER CODE CAMP 2024**

MINI PROJECT CUỐI KHÓA

TÊN DỰ ÁN: Quản lý chi tiêu hàng tháng cho sinh viên

**NHÓM 9**

**LINK GITHUB: https://github.com/Deadline-Student-Manage**

# **MÔ TẢ DỰ ÁN****:** Ứng dụng này giúp sinh viên quản lý và theo dõi chi tiêu hàng tháng của mình, đảm bảo họ không vượt quá ngân sách đã đặt ra.

# **THÀNH VIÊN TRONG NHÓM**

|  |  |  |
| --- | --- | --- |
|  | **Thành viên** | **Vai trò** |
| 1 | Đỗ Kiến Hưng | Winforms Version |
| 2 | Hồ Việt Anh | Web Version |
| 3 | Nguyễn Thị Ngọc Thùy | Kiểm thử, viết báo cáo, lên ý tưởng |
| 4 | Nguyễn Thị Hồng Nhung | Kiểm thử, viết báo cáo, lên ý tưởng |

# 

# **1. ĐẶT VẤN ĐỀ, MỤC TIÊU CẦN ĐẠT ĐƯỢC**

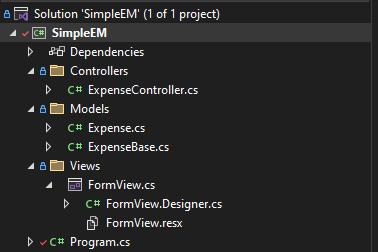
Vấn đề: Sinh viên thường xuyên gặp khó khăn trong việc quản lý chi tiêu, dẫn đến tình trạng "vung tay quá trán" và thiếu hụt tài chính vào cuối tháng. Điều này xuất phát từ việc không theo dõi chi tiết các khoản chi tiêu hàng ngày, cũng như không có kế hoạch ngân sách hợp lý cho từng danh mục chi tiêu.

Mục tiêu: Xây dựng một ứng dụng có thể giúp sinh viên dễ dàng theo dõi các khoản chi tiêu hàng ngày, đồng thời hỗ trợ họ thiết lập và tuân thủ ngân sách chi tiêu. Ứng dụng này cần phải đơn giản, dễ sử dụng, và cung cấp thông tin chi tiêu một cách trực quan, giúp người dùng nắm bắt được tình hình tài chính cá nhân của mình một cách nhanh chóng, đảm bảo được những tính năng và yêu cầu.

# **2. CÁCH GIẢI QUYẾT**

**2.1. Mô hình giải quyết**

Để giải quyết vấn đề quản lý chi tiêu cho sinh viên, ứng dụng sử dụng mô hình MVC (Model-View-Controller) để tổ chức code một cách rõ ràng và dễ bảo trì. Các tính năng chính được xây dựng dựa trên nguyên tắc lập trình hướng đối tượng (OOP), đảm bảo mã nguồn dễ dàng mở rộng và tái sử dụng.

  
*Solution Explorer of project*

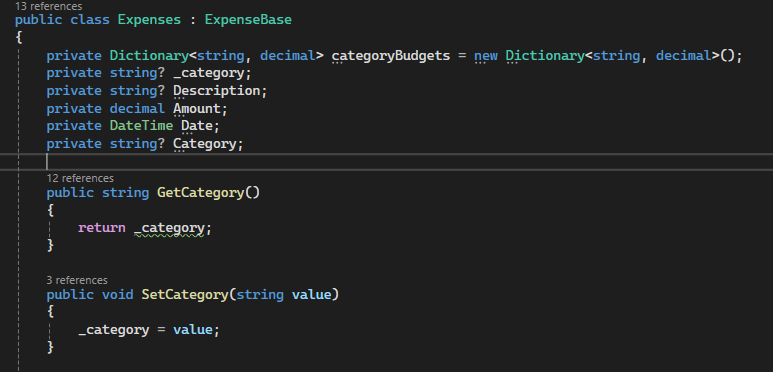
+ Model: Đại diện cho dữ liệu ứng dụng, bao gồm các lớp Expenses để quản lý thông tin về các giao dịch và CategoryBudgets để quản lý ngân sách cho từng danh mục.

+ View: Cung cấp giao diện người dùng để nhập liệu và hiển thị thông tin chi tiêu. Các thành phần giao diện chính bao gồm các TextBox, ComboBox, ListView và Labels.

+ Controller: Điều khiển logic ứng dụng, tương tác giữa Model và View. Lớp ExpenseController chịu trách nhiệm quản lý các giao dịch và ngân sách của người dùng, bao gồm thêm, xóa, cập nhật và lưu trữ dữ liệu.

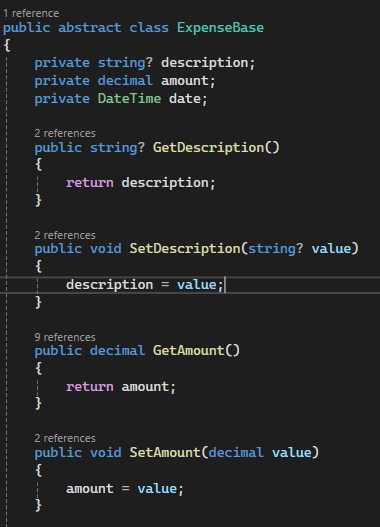
**2.2. Tính chất OOP của ứng dụng**

**2.2.1. Encapsulation (Đóng gói)**

  
*Đối tượng và phương thức thuộc lớp Expenses được kế thừa từ ExpenseBase   
(Nguồn: Models/Expense.cs)*

Các thuộc tính như Description, Amount, Date và Category được khai báo là private, chỉ có thể được truy cập thông qua các phương thức công khai như GetCategory và SetCategory. Điều này đảm bảo dữ liệu bên trong Expenses được bảo vệ khỏi sự thay đổi không mong muốn từ bên ngoài.

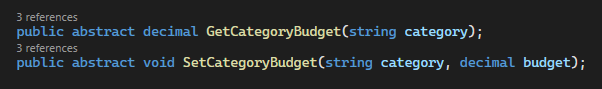
**2.2.2. Inheritance (Kế thừa)**

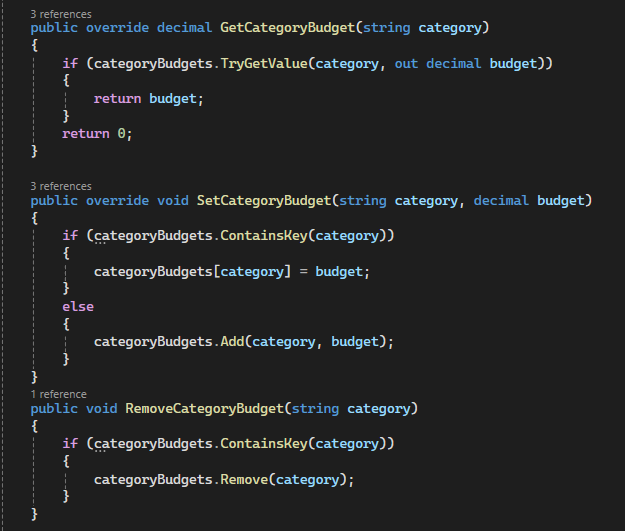


*Đối tượng và phương thức thuộc lớp ExpenseBase (Nguồn: Models/ExpenseBase.cs)*

Expenses kế thừa từ class ExpenseBase, hưởng các phương thức như GetDescription, SetDescription, GetAmount, SetAmount, và GetDate, giúp tái sử dụng mã nguồn.

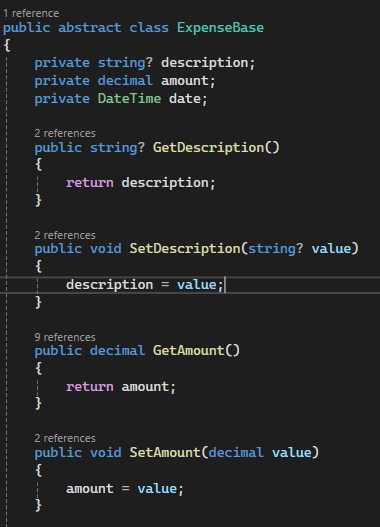
**2.2.3. Polymorphism (Đa hình)**



  
*Các phương thức OOP (Nguồn: Models)*

Các phương thức GetCategoryBudget và SetCategoryBudget trong ExpenseBase là abstract, buộc các lớp con như Expenses phải override lại và thực hiện theo cách riêng. Điều này cho phép các lớp con thực thi các phương thức theo các cách khác nhau.

**2.2.4. Abstraction (Trừu tượng hóa)**

  
*Lớp ExpenseBase đảm bảo tính trừu tượng hóa (Nguồn: Models/ExpenseBase.cs)*

ExpenseBase là một class abstract, cung cấp một cấu trúc cơ bản cho các lớp con như Expenses. Nó ẩn đi các chi tiết cụ thể của các phương thức GetCategoryBudget và SetCategoryBudget, buộc các lớp con phải triển khai cụ thể chúng.

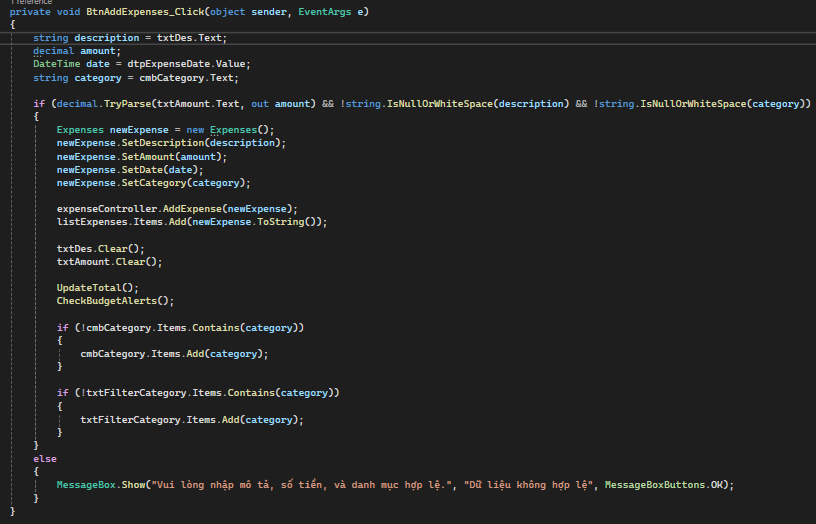
# **3. TÍNH NĂNG ĐÃ XÂY DỰNG**

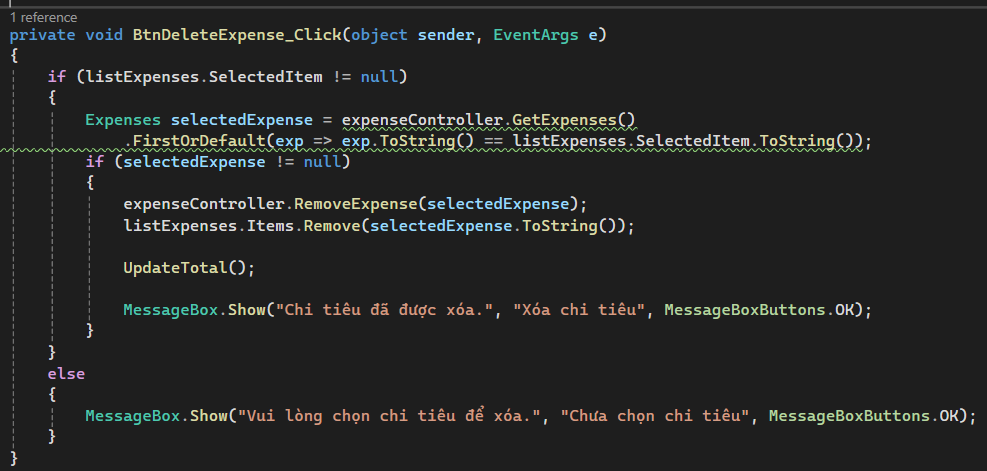


*Giao diện người dùng (Nguồn: Views/FormView.Designer.cs)*

**3.1. Thêm, xóa, và cập nhật giao dịch**

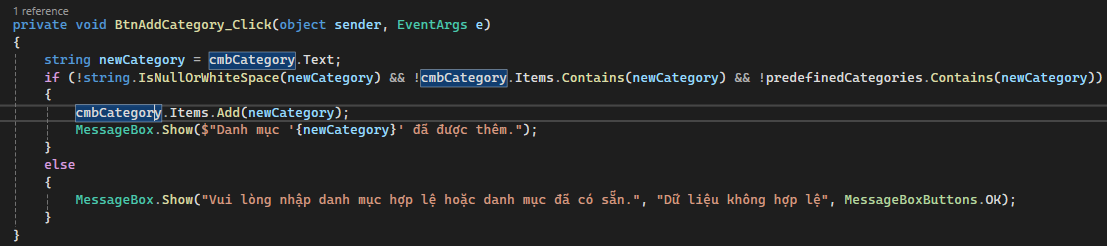
Người dùng có thể thêm giao dịch mới vào các danh mục chi tiêu, xóa các giao dịch không cần thiết, hoặc cập nhật các giao dịch đã có. Ngoài những danh mục có sẵn ra, người dùng cũng có thể bổ sung thêm và xóa danh mục tùy ý thích.



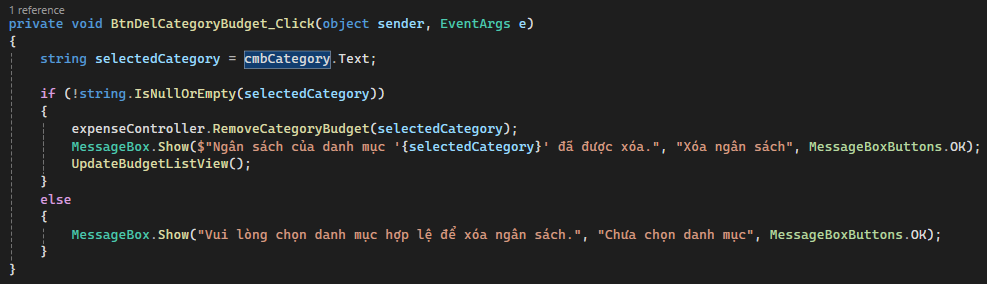
*Phương thức thêm giao dịch mới (Nguồn: Views/FormView.cs)* *Phương thức xóa giao dịch mới (Nguồn: Views/FormView.cs)*

**3.2. Quản lý ngân sách cho từng danh mục**

Người dùng có thể đặt hoặc xóa ngân sách cho từng danh mục



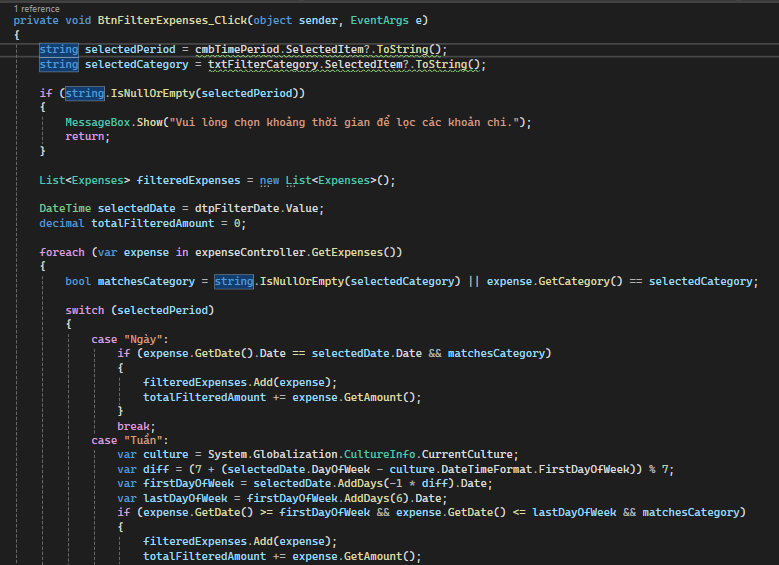
*Phương thức thêm danh mục (Nguồn: Views/FormView.cs)*

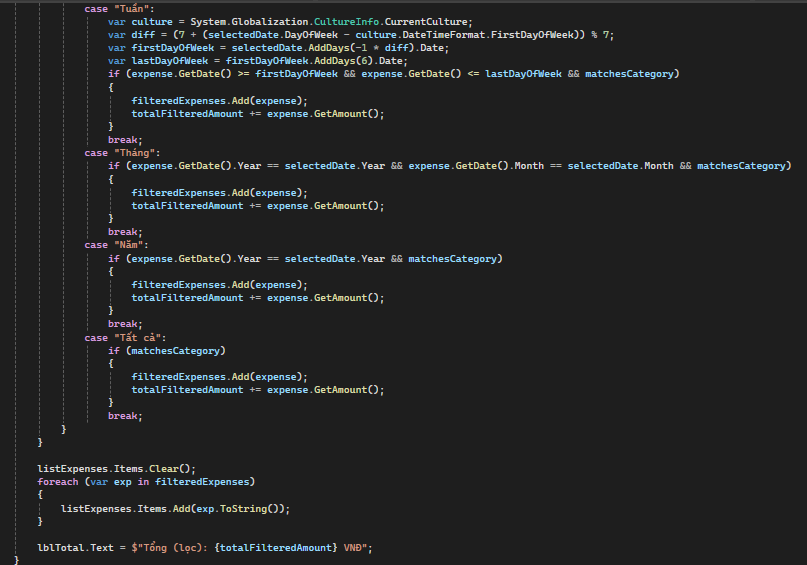


*Phương thức thêm danh mục (Nguồn: Views/FormView.cs)*

**3.3. Lọc và hiển thị giao dịch**

Giao diện người dùng hỗ trợ lọc giao dịch theo thời gian (ngày, tháng, năm, tuần, tất cả) và theo danh mục, giúp người dùng dễ dàng theo dõi chi tiêu.

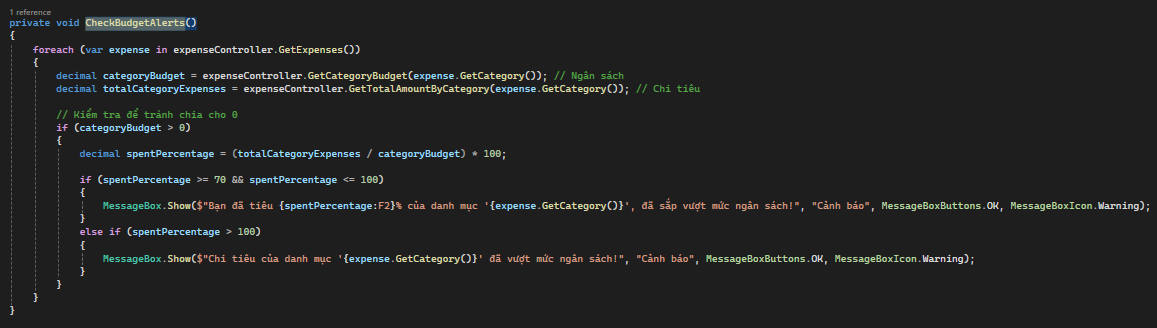




*Phương thức thực hiện việc lọc chi tiêu (Nguồn: Views/FormView.cs)*

**3.4. Cảnh báo chi tiêu vượt mức**

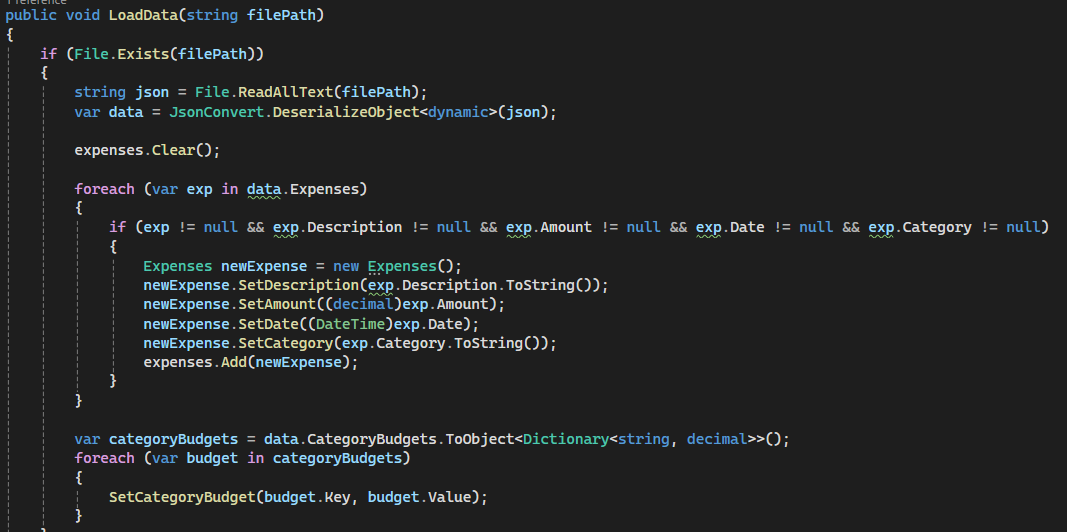
Khi chi tiêu của 1 danh mục vượt quá 70% ngân sách của danh mục đó sẽ có hộp tin nhắn hiện lên thông báo vượt mức chi tiêu cho phép



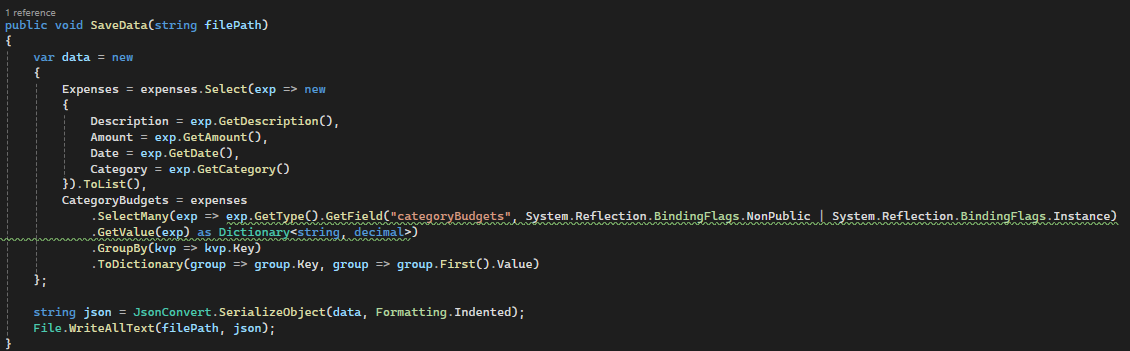
*Phương thức cảnh báo vượt chi tiêu (Nguồn: Views/FormView.cs)*

**3.5. Lưu trữ và tải lại dữ liệu**

Sử dụng file JSON cùng thư viện Newtonsoft.Json để lưu tất cả dữ liệu chi tiêu và ngân sách của người dùng khi mở lại ứng dụng.



*Phương thức tải dữ liệu (Nguồn: Controllers/ExpenseController.cs)*



*Phương thức lưu dữ liệu (Nguồn: Controllers/ExpenseController.cs)*

# **4. ĐỊNH HƯỚNG**

# Tính năng phân tích chi tiêu: Sử dụng thư viện WinForms.DataVisualization nhằm cung cấp biểu đồ và báo cáo chi tiêu, giúp người dùng hiểu rõ hơn về thói quen chi tiêu của mình.