ĐẠI HỌC QUỐC GIA
THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

CSC10003 – Phương Pháp Lập Trình Hướng Đối Tượng

BÁO CÁO ĐỒ ÁN

Ứng dụng Design Patterns trong thiết kế Hệ thống quản lý nhiệm vụ

Họ tên	MSSV
Tôn Thất Minh Đăng	23127037
Bùi Minh Duy	23127040
Nguyễn Thị Khánh Linh	23127082
Nguyễn Lê Hồ Anh Khoa	23127211

Giảng viên hướng dẫn

Bùi Tiến Lên

Thành phố Hồ Chí Minh, 30/12/2024

Mục lục

1 Thông tin giới thiệu

- Tên học phần: Phương pháp lập trình hướng đối tượng
- Giảng viên hướng dẫn: Bùi Tiến Lên
- Đồ án thực hiện: Ứng dụng Design Patterns trong thiết kế Hệ thống quản lý nhiệm vụ.
- Danh sách thành viên:

STT	MSSV	HỌ VÀ TÊN	EMAIL	VAI TRÒ
01	23127037	Tôn Thất Minh Đăng	ttmdang23@clc.fitus.edu.vn	Thành viên
02	23127040	Bùi Minh Duy	bmduy23@clc.fitus.edu.vn	Thành viên
03	23127082	Nguyễn Thị Khánh Linh	ntklinh23@clc.fitus.edu.vn	Nhóm trưởng
04	23127211	Nguyễn Lê Hồ Anh Khoa	nlhakhoa23@clc.fitus.edu.vn	Thành viên

Bảng 1: Danh sách thành viên

2 Thư mục nộp bài

1. Bài báo cáo: `3-23127037-23127040-23127082-23127211.pdf`
2. Slide: `S3-23127037-23127040-23127082-23127211.pdf` (vì không thể lưu trùng tên tệp với Bài báo cáo, nên thêm chữ 'S' ở phía trước)
3. Video: `3-23127037-23127040-23127082-23127211.mp4`
4. Hình ảnh `UML.jpg` chứa sơ đồ UML của hệ thống.
5. Mã nguồn chạy hệ thống: Thư mục `Task-Management-System`

3 Ý tưởng hình thành sản phẩm

Chúng tôi nhận thấy trong quá trình làm việc nhóm, việc quản lý nhiệm vụ thường gặp khó khăn, dẫn đến chồng chéo công việc và giảm hiệu suất. Các công cụ quản lý hiện có thường quá phức tạp hoặc thiếu tính năng phù hợp.

Vì vậy, chúng tôi đã phát triển **Task Management System (Hệ Thống Quản Lý Nhiệm Vụ)**, một ứng dụng đơn giản nhưng hiệu quả, tập trung vào quản lý nhiệm vụ, phân nhóm theo cấu trúc cây, theo dõi tiến độ trực quan và hỗ trợ chỉnh sửa linh hoạt. Mục tiêu của chúng tôi là tạo ra một công cụ dễ sử dụng, đáp ứng tốt nhu cầu của các nhóm nhỏ.

4 Vấn đề đặt ra và giải quyết bằng Design Pattern

4.1 Quản lý duy nhất và đồng bộ dữ liệu

- **Vấn đề:** Làm sao để đảm bảo chỉ có một đối tượng quản lý toàn bộ nhiệm vụ và dữ liệu?
- **Giải pháp:** Sử dụng **Singleton Pattern** để tạo ra một lớp quản lý nhiệm vụ duy nhất, đảm bảo tính đồng bộ và nhất quán của dữ liệu trong hệ thống.

4.2 Tạo nhiệm vụ phức tạp với nhiều thuộc tính

- **Vấn đề:** Làm sao để xây dựng nhiệm vụ với nhiều thông tin đầu vào mà không làm mã nguồn trở nên khó bảo trì?
- **Giải pháp:** Sử dụng **Builder Pattern**, giúp tạo nhiệm vụ linh hoạt và đảm bảo tính mở rộng, dễ dàng chỉnh sửa hoặc thêm thuộc tính mới.

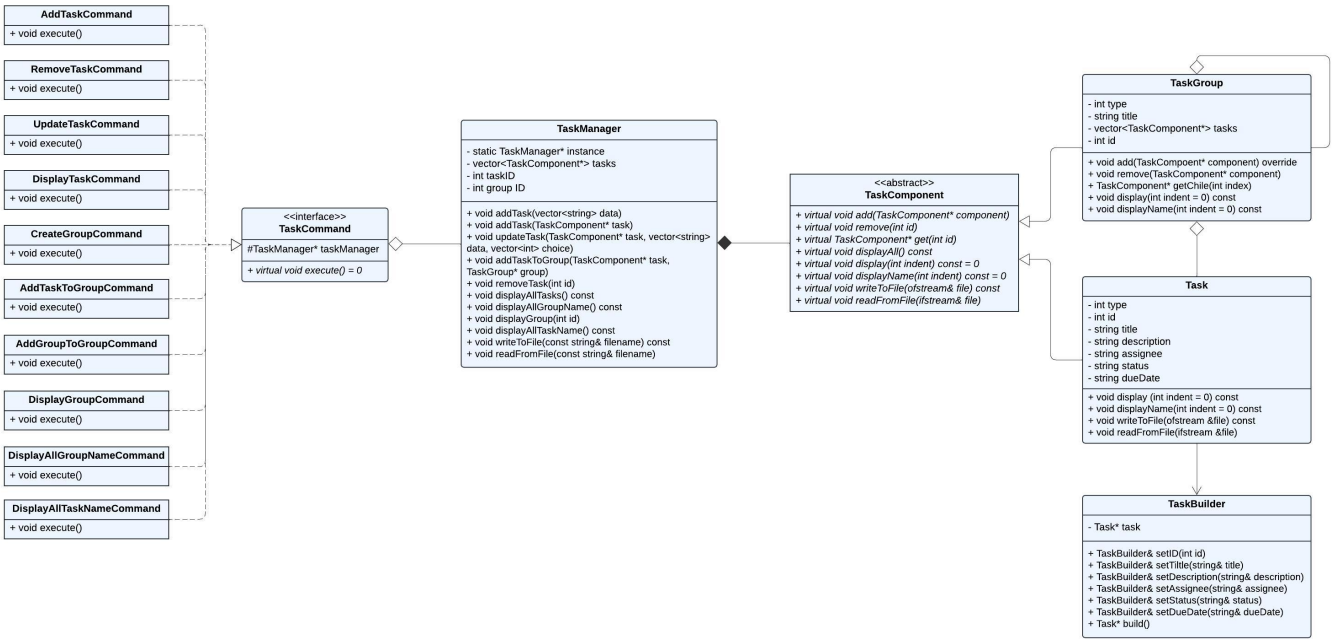
4.3 Tổ chức nhiệm vụ phân cấp

- **Vấn đề:** Làm sao để quản lý các nhiệm vụ theo nhóm, phân cấp nhóm và hiển thị cấu trúc một cách trực quan?
- **Giải pháp:** Sử dụng **Composite Pattern**, cho phép tổ chức nhiệm vụ thành các nhóm lớn nhỏ theo cấu trúc cây, từ đó dễ dàng quản lý và theo dõi.

4.4 Tách biệt logic xử lý thao tác với giao diện

- **Vấn đề:** Làm sao để đảm bảo các thao tác như thêm, sửa, xóa nhiệm vụ có thể được quản lý độc lập với giao diện người dùng?
- **Giải pháp:** Sử dụng **Command Pattern**, giúp đóng gói các thao tác thành các lệnh riêng biệt, đảm bảo tách biệt logic xử lý khỏi giao diện, từ đó dễ dàng kiểm tra và bảo trì hệ thống.

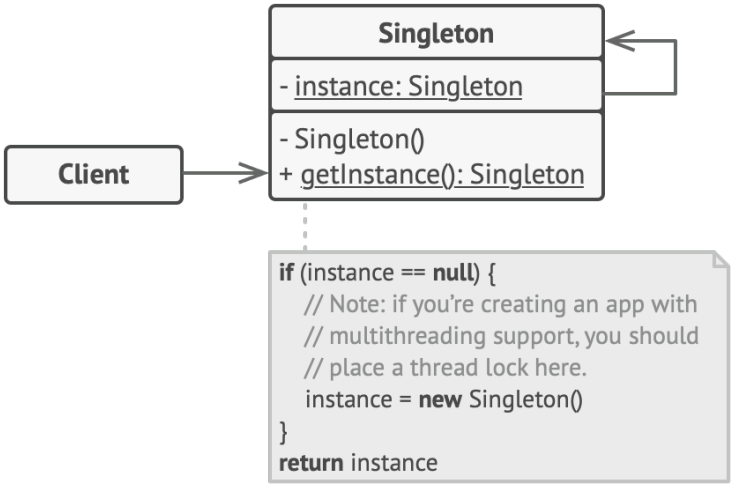
5 Bản vẽ UML



6 Chi tiết về các Design Pattern được sử dụng

6.1 Singleton Pattern

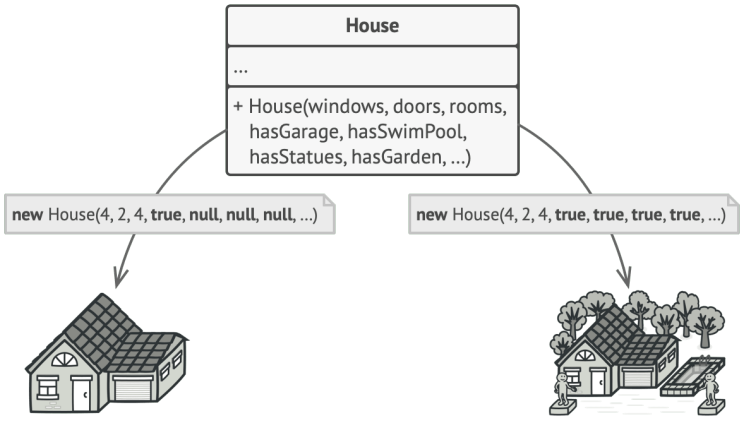
- **Mục đích**
  - Quản lý tập trung toàn bộ nhiệm vụ trong hệ thống, đảm bảo không có sự trùng lặp và thống nhất dữ liệu.
- **Đặt vấn đề**
  - Khi xây dựng hệ thống quản lý nhiệm vụ, chúng tôi cần một đối tượng duy nhất để lưu trữ và quản lý danh sách nhiệm vụ. Nếu có nhiều hơn một instance của đối tượng này, dữ liệu có thể bị xung đột hoặc không đồng nhất.
- **Giải pháp**
  - Singleton Pattern đảm bảo rằng chỉ có duy nhất một thể hiện của lớp `TaskManager` trong toàn bộ chương trình. Điều này được thực hiện bằng cách
    * Ẩn hàm khởi tạo của lớp.
    * Sử dụng một phương thức tĩnh để trả về thể hiện duy nhất.
    * Minh họa chung cấu trúc của Singleton Pattern



- **Áp dụng trong sản phẩm**
  - Trong sản phẩm của chúng tôi, lớp `TaskManager` được triển khai dưới dạng Singleton để quản lý danh sách các nhiệm vụ và nhóm nhiệm vụ. Bất kỳ phần nào của hệ thống muốn truy cập danh sách này đều thông qua instance duy nhất của `TaskManager`.

6.2 Builder Pattern

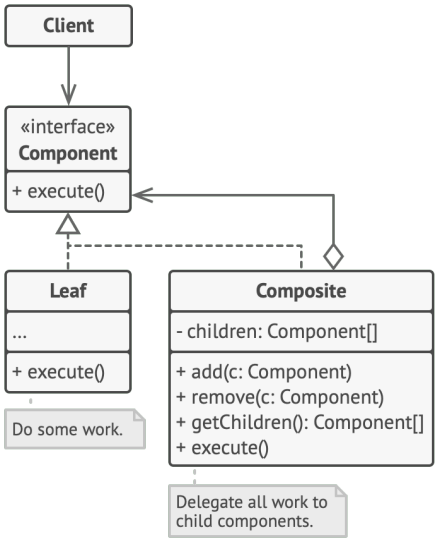
- **Mục đích**
  - Tạo các nhiệm vụ với nhiều thuộc tính một cách dễ dàng và linh hoạt hơn.
- **Đặt vấn đề**
  - Trong hệ thống, mỗi nhiệm vụ có thể có nhiều thuộc tính như tiêu đề, mô tả, thành viên thực hiện, ngày hoàn thành,... Việc khởi tạo các nhiệm vụ phức tạp bằng một constructor có quá nhiều tham số sẽ dẫn đến mã nguồn khó đọc và dễ lỗi.
- **Giải pháp**
  - Builder Pattern giúp tách biệt việc khởi tạo đối tượng khỏi logic nghiệp vụ. Chúng tôi triển khai một lớp TaskBuilder với các phương thức cho phép thiết lập từng thuộc tính của nhiệm vụ.
    - Minh họa chung cấu trúc của Builder Pattern



- **Áp dụng trong sản phẩm**
  - Khi tạo một nhiệm vụ mới, hệ thống sử dụng `TaskBuilder` để xây dựng đối tượng `Task` với các thuộc tính cần thiết. Điều này giúp mã nguồn dễ hiểu, dễ mở rộng và giảm thiểu rủi ro lỗi.

6.3 Composite Pattern

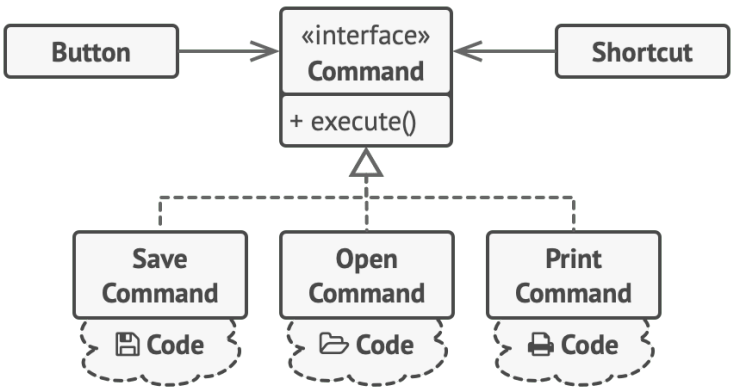
- **Mục đích**
  - Tổ chức các nhiệm vụ theo cấu trúc phân cấp thành từng nhóm.
- **Đặt vấn đề**
  - Trong một dự án, nhiệm vụ thường được chia thành các nhiệm vụ con hoặc nhóm nhiệm vụ. Cần có một cấu trúc cho phép xử lý cả nhiệm vụ riêng lẻ và nhóm nhiệm vụ một cách đồng nhất.
- **Giải pháp**
  - Composite Pattern cho phép tổ chức các đối tượng theo cấu trúc cây. Chúng tôi tạo một lớp cơ sở `TaskComponent`, từ đó các lớp Task và TaskGroup kế thừa. Một nhóm nhiệm vụ (`TaskGroup`) có thể chứa nhiều nhiệm vụ hoặc nhóm nhiệm vụ khác.ác phương thức cho phép thiết lập từng thuộc tính của nhiệm vụ.
    * Minh họa chung cấu trúc của Composite Pattern



- **Áp dụng trong sản phẩm**
  - Sản phẩm của chúng tôi sử dụng Composite Pattern để quản lý danh sách nhiệm vụ. Người dùng có thể tạo các nhóm nhiệm vụ và thêm các nhiệm vụ con vào đó. Điều này giúp việc quản lý nhiệm vụ trở nên trực quan và hiệu quả hơn.

6.4 Command Pattern

- **Mục đích**
  - Tách biệt logic thực thi hành động và khả năng lưu trữ các hành động này để sử dụng lại.
- **Đặt vấn đề**
  - Hệ thống cần một cách tiếp cận để thực hiện các thao tác như thêm, sửa hoặc xóa nhiệm vụ mà không làm rối mã nguồn chính. Đồng thời, cần lưu trữ các thao tác này để xử lý các yêu cầu khác như log dữ liệu.
- **Giải pháp**
  - Command Pattern giúp đóng gói mỗi thao tác vào một đối tượng riêng biệt. Mỗi lệnh sẽ chứa thông tin cần thiết để thực thi hành động. Điều này cho phép dễ dàng thêm các tính năng mới hoặc thay đổi logic thực thi mà không ảnh hưởng đến hệ thống chính.
    - Minh họa chung cấu trúc của Command Pattern



- **Áp dụng trong sản phẩm**
  - Chúng tôi sử dụng Command Pattern để quản lý các thao tác liên quan đến nhiệm vụ, như thêm hoặc xóa nhiệm vụ. Mỗi thao tác được gói gọn trong một lớp lệnh cụ thể, giúp mã nguồn dễ mở rộng và duy trì.

7 Quy trình hoạt động của hệ thống

7.1 Quy trình hoạt động

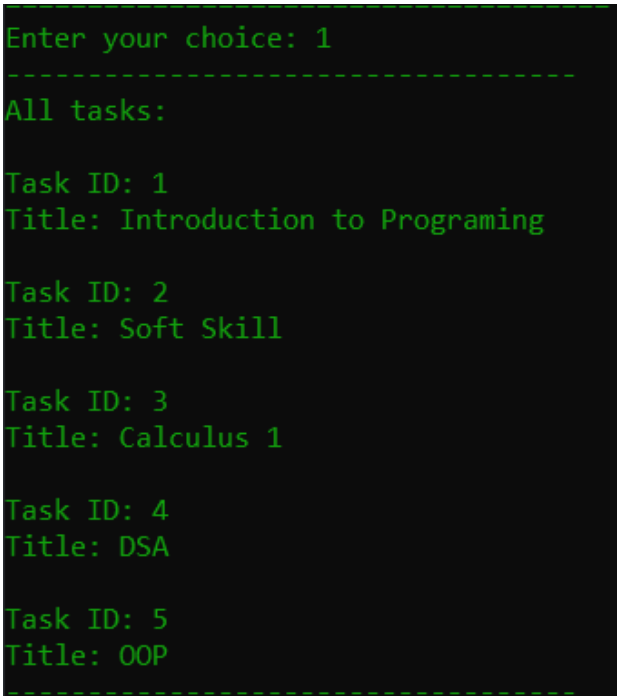
- Khi chạy chương trình, hệ thống đọc file text chứa thông tin lưu trữ các nhiệm vụ và nhóm nhiệm vụ từ lần sử dụng trước để tiếp tục cập nhật.
- Trong quá trình chạy hệ thống, người dùng có thể thực hiện các thao tác thêm, sửa, xóa các nhiệm vụ và các nhóm nhiệm vụ hoặc xem thông tin của nhiệm vụ và các nhóm nhiệm vụ.
- Khi chọn thoát chương trình, danh sách nhiệm vụ và nhóm nhiệm vụ sau khi cập nhật được lưu lại vào file text cho lần sử dụng tiếp theo.

7.2 Những chức năng chính

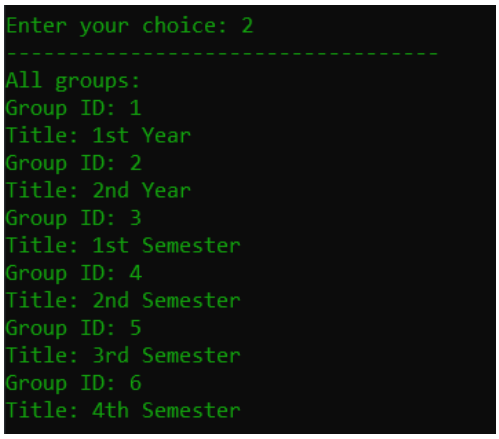
1. **Hiển thị danh sách nhiệm vụ:** Cho phép người dùng xem nhanh tất cả các nhiệm vụ (bao gồm ID và tên) đang có trong hệ thống.
2. **Hiển thị danh sách nhóm nhiệm vụ:** Liệt kê các nhóm nhiệm vụ đã tạo (bao gồm ID và tên) đang có trong hệ thống.
3. **Xem chi tiết nhóm nhiệm vụ:** Hiển thị thông tin cụ thể và danh sách nhiệm vụ thuộc về một nhóm nhất định.
4. **Xem chi tiết nhiệm vụ:** Cung cấp thông tin chi tiết về một nhiệm vụ cụ thể như tên, mô tả, ngày hoàn thành,...
5. **Thêm nhiệm vụ mới:** Tạo mới một nhiệm vụ với các thuộc tính như tiêu đề, mô tả và thời hạn,.. Người dùng có quyền bỏ trống bất kỳ thuộc tính nào.
6. **Xóa nhiệm vụ:** Loại bỏ một nhiệm vụ khỏi hệ thống khi không còn cần thiết.

7. **Cập nhật nhiệm vụ:** Chỉnh sửa thông tin của một nhiệm vụ đã có để phù hợp với thay đổi.
8. **Tạo nhóm nhiệm vụ mới:** Tạo một nhóm nhiệm vụ để tổ chức các nhiệm vụ liên quan thành một cụm.
9. **Thêm nhiệm vụ vào nhóm:** Gán một nhiệm vụ cụ thể vào một nhóm nhiệm vụ đã có.
10. **Thêm nhóm vào nhóm khác:** Xây dựng cấu trúc phân cấp bằng cách gộp các nhóm nhiệm vụ nhỏ hơn vào một nhóm lớn hơn.

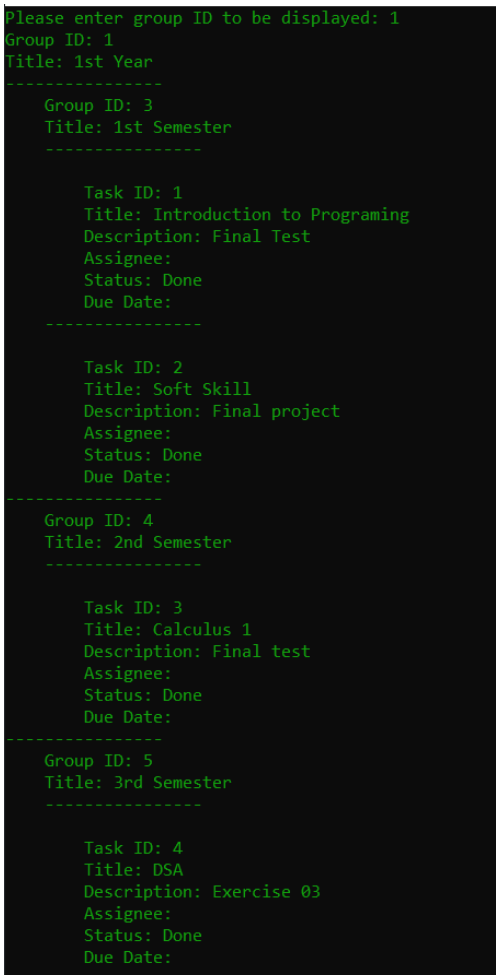
8 Minh họa sử dụng hệ thống



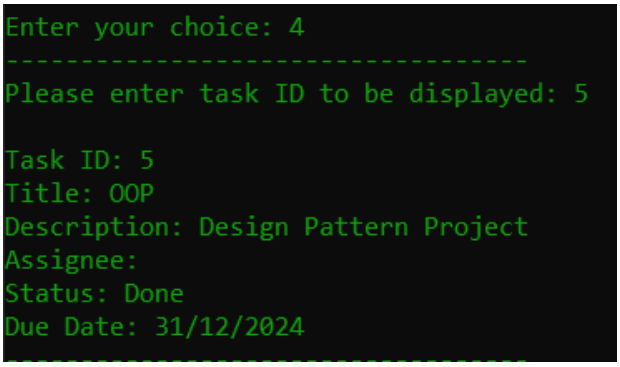
Hình 1: In ra tất cả nhiệm vụ đang có trong hệ thống



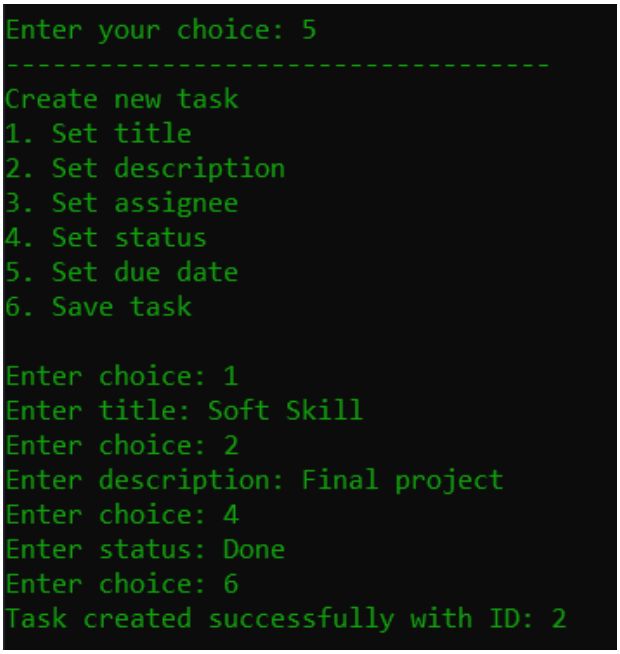
Hình 2: In ra tất cả nhóm nhiệm vụ đang có trong hệ thống



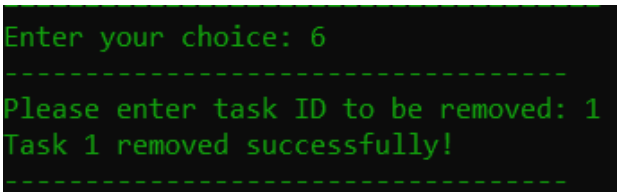
Hình 3: In ra tất cả thành phần trong nhóm chỉ định



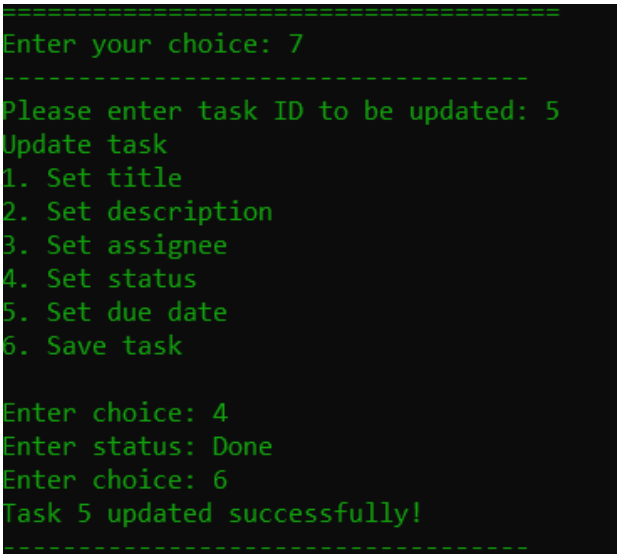
Hình 4: In ra thông tin chi tiết của nhiệm vụ chỉ định



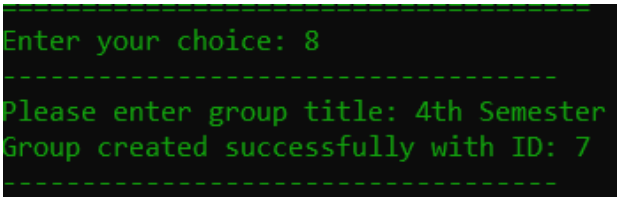
Hình 5: Thêm nhiệm vụ mới vào hệ thống



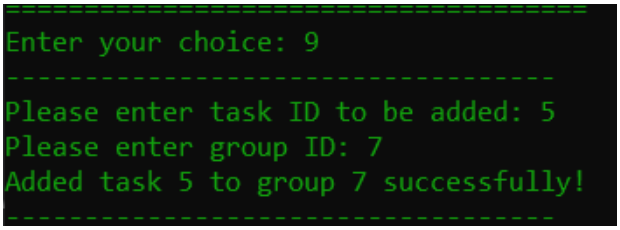
Hình 6: Xóa một nhiệm vụ chỉ định khỏi hệ thống



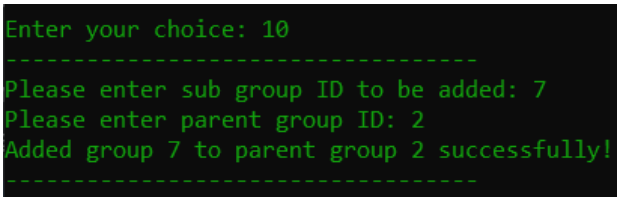
Hình 7: Cập nhật thông tin của nhiệm vụ được chỉ định



Hình 8: Tạo nhóm nhiệm vụ mới



Hình 9: Thêm nhiệm vụ vào nhóm nhiệm vụ



Hình 10: Thêm nhóm nhiệm vụ vào nhóm khác

9 Phân công công việc

STT	MSSV	HỌ VÀ TÊN	PHÂN CÔNG CÔNG VIỆC	HOÀN THÀNH
01	23127037	Tôn Thất Minh Đăng	• Cài đặt **Command Pattern** bao gồm các lớp trong file `Command.h`. • Thiết kế Slide cho đồ án. • Quay video. • Viết kịch bản video.	100%
02	23127040	Bùi Minh Duy	• Cài đặt **Build Pattern** bao gồm lớp `TaskBuild` và lớp `Task`. • Testing và sửa lỗi. • Thiết kế UML. • Quay video demo hệ thống.	100%
03	23127082	Nguyễn Thị Khánh Linh	• Cài đặt **Singleton Pattern** bao gồm lớp `TaskManager`. • Xử lí các hàm display, đọc ghi file, testing và sửa lỗi. • Viết báo cáo, kịch bản video. • Chỉnh sửa video.	100%
04	23127211	Nguyễn Lê Hồ Anh Khoa	• Cài đặt **Composite Pattern** bao gồm lớp `TaskGroup` và `TaskComponent`. • Testing và sửa lỗi. • Viết báo cáo. • Tạo lồng tiếng cho video.	100%

Bảng 2: Bảng phân công công việc

10 Đánh giá quá trình làm đồ án

10.1 Ưu điểm

- Phân công công việc rõ ràng và hợp lý: Các thành viên đã có sự phân chia nhiệm vụ cụ thể, mỗi người tập trung vào một khía cạnh mà mình mạnh nhất như thiết kế giao diện, xây dựng logic xử lý, và viết tài liệu.
- Sử dụng Design Pattern hiệu quả: Nhóm đã chọn các mẫu phù hợp với từng chức năng, đảm bảo tính linh hoạt và tối ưu hóa.
- Khai thác tối đa công cụ hỗ trợ: Nhóm đã sử dụng các công cụ như GitHub, Google

Sheet, Google Doc để quản lý mã nguồn và theo dõi tiến độ, đảm bảo mọi công việc đều được lưu trữ và kiểm soát tốt.

10.2 Nhược điểm

- Thời gian phân bổ chưa đồng đều: Một số giai đoạn trong quá trình làm đồ án bị chậm tiến độ, đặc biệt là khi xử lý lỗi phát sinh.
- Chưa tối ưu giao diện người dùng (UI): Chưa dựng được giao diện người dùng, làm việc với console vẫn còn đơn giản và chưa trực quan.
- Thiếu kiểm thử toàn diện: Hệ thống chỉ mới được kiểm thử thông qua các thành viên của nhóm, không thể tránh khỏi việc bỏ sót các trường hợp gây ra lỗi.

10.3 Cải thiện trong tương lai:

- Tăng cường lập kế hoạch: Xây dựng kế hoạch chi tiết hơn, phân chia rõ ràng thời gian cho từng giai đoạn và xác định mốc hoàn thành cụ thể để tránh tình trạng chậm tiến độ.
- Đầu tư vào giao diện: Thiết kế giao diện người dùng, sử dụng các framework UI hiện đại hơn để cải thiện trải nghiệm người dùng.
- Mở rộng kiểm thử: Thực hiện kiểm thử nhiều trường hợp hơn, đặc biệt là các tình huống biên hoặc dữ liệu lớn, nhằm đảm bảo hệ thống hoạt động ổn định trong mọi tình huống.

11 Nguồn tài liệu và công cụ tham khảo

- **Refactoring and Design Patterns**
- **ChatGPT**
- **GitHub**