**TRƯỜNG ĐẠI HỌC THỦ DẦU MỘT**

**KHOA KỸ THUẬT – CÔNG NGHỆ**



**ĐỒ ÁN MÔN HỌC  
LẬP TRÌNH ỨNG DỤNG TRÊN WINDOWS**

**QUẢN LÝ HOẠT ĐỘNG DẠY HỌC**

**GVHD: ThS. Nguyễn Kim Duy**

**Nhóm 1:**

**Nguyễn Hoàng Anh – 2124802010025**

**Nguyễn Minh Hiếu – 2124802010706**

**Lê Hồng Lộc – 2124802010747**

**BÌNH DƯƠNG – 7/2023**

**TRƯỜNG ĐẠI HỌC THỦ DẦU MỘT**

**KHOA KỸ THUẬT – CÔNG NGHỆ**



**ĐỒ ÁN MÔN HỌC  
LẬP TRÌNH ỨNG DỤNG TRÊN WINDOWS**

**QUẢN LÝ HOẠT ĐỘNG DẠY HỌC**

**GVHD: ThS. Nguyễn Kim Duy**

**Nhóm 1:**

**Nguyễn Hoàng Anh – 2124802010025**

**Nguyễn Minh Hiếu – 2124802010706**

**Lê Hồng Lộc – 2124802010747**

**BÌNH DƯƠNG – 7/2023**

**TRƯỜNG ĐẠI HỌC THỦ DẦU MỘT**

**KHOA KỸ THUẬT – CÔNG NGHỆ**

**NHẬN XÉT VÀ CHẤM ĐIỂM CỦA GIẢNG VIÊN**

Họ và tên giảng viên: **Ths. Nguyễn Kim Duy**

Tên đề tài: **Quản lý hoạt động dạy học**

Nội dung nhận xét:

**Điểm:**

Bằng số:

Bằng chữ:

|  |  |
| --- | --- |
|  | **GIẢNG VIÊN CHẤM**  *(Ký, ghi rõ họ tên)* |

**ThS. Nguyễn Kim Duy**

# MỞ ĐẦU

## **Lý do chọn đề tài:**

Việc quản lý hoạt động dạy học C# WinForms là một chủ đề quan trọng và có giá trị trong việc đảm bảo chất lượng giáo dục và đào tạo cho sinh viên học ngành công nghệ thông tin.

Lý do chính để chọn đề tài này có thể là để cải thiện chất lượng giảng dạy và học tập của sinh viên, đặc biệt là trong lĩnh vực lập trình desktop. Quản lý hoạt động dạy học C# WinForms giúp cho giảng viên có thể hiểu rõ hơn về các khía cạnh của ngôn ngữ lập trình này và có thể truyền đạt kiến thức một cách rõ ràng, dễ hiểu hơn cho sinh viên.

Ngoài ra, đề tài này cũng giúp cho nhà trường có thể quản lý tốt hơn hoạt động dạy và học tại trường và đảm bảo việc truyền đạt kiến thức được diễn ra một cách hiệu quả, đồng thời giúp cho sinh viên có thể học tập và thực hành C# WinForms một cách nhanh chóng và chính xác.

Tóm lại, quản lý hoạt động dạy học C# WinForms là một đề tài quan trọng và có giá trị để cải thiện chất lượng giáo dục và đào tạo, giúp cho giảng viên và sinh viên có thể hiểu và thực hành C# WinForms một cách hiệu quả.Mục đích của đề tài:

## **Mục đích của đề tài:**

Mục đích của việc chọn đề tài quản lý hoạt động dạy học C# WinForms là để cải thiện chất lượng giáo dục và đào tạo, giúp cho giảng viên và sinh viên có thể hiểu và thực hành C# WinForms một cách hiệu quả.

Cụ thể, việc quản lý hoạt động dạy học C# WinForms nhằm giúp cho giảng viên có thể truyền đạt kiến thức một cách rõ ràng và dễ hiểu hơn cho sinh viên, đồng thời đảm bảo việc truyền đạt kiến thức được diễn ra một cách hiệu quả. Ngoài ra, việc quản lý hoạt động dạy học còn giúp cho nhà trường có thể quản lý tốt hơn hoạt động dạy và học tại trường và đảm bảo việc học tập và thực hành C# WinForms của sinh viên được diễn ra một cách nhanh chóng và chính xác.

Tóm lại, mục đích của việc chọn đề tài quản lý hoạt động dạy học C# WinForms là để cải thiện chất lượng giáo dục và đào tạo, giúp cho giảng viên và sinh viên có thể học tập và thực hành C# WinForms một cách hiệu quả.

## **Phạm vi của đề tài:**

Phạm vi đối tượng: Đề tài có thể tập trung vào nghiên cứu về quản lý và phát triển bền vững của một trường đại học cụ thể, hoặc so sánh giữa các trường đại học khác nhau.

Phạm vi nội dung: Đề tài tập trung vào nghiên cứu về các vấn đề liên quan đến quản lý hoạt động dạy học trong trường đại học, đối tượng chủ đạo là Giảng viên và Sinh viên. Quản trị thực hiện quản lý tài khoản và phân công Giảng viên thực hiện các công việc dạy học, phân công phòng học để lên lịch giảng dạy, phân công Sinh viên vào các lớp. Giảng viên thực hiện quản lý các lớp đang giảng dạy và sinh viên của mình. Sinh viên sỡ hữu các quyền lợi học tập.

|  |  |
| --- | --- |
| TRƯỜNG ĐH THỦ DẦU MỘT  **VIỆN KỸ THUẬT - CÔNG NGHỆ** | **CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**  **Độc lập – Tự do – Hạnh phúc** |

**PHIẾU ĐÁNH GIÁ PROJECT**

*(Đánh giá quá trình thực hiện, báo cáo văn bản và sản phẩm)*

**Học phần:** LẬP TRÌNH TRÊN WINDOWS (3+0) **Lớp:** KTCN.CQ.04

**Tên đề tài:** QUẢN LÝ HOẠT ĐỘNG DẠY HỌC

**Họ và tên sinh viên** (MSSV):

|  |  |
| --- | --- |
| **Họ tên** | **MSSV** |
| Nguyễn Hoàng Anh | 2124802010025 |
| Nguyễn Minh Hiếu | 2124802010706 |
| Lê Hồng Lộc | 2124802010747 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ***Tiêu chí*** | | ***Trọng số %*** | ***Tốt***  ***100%*** | | ***Khá***  ***75%*** | ***Trung bình***  ***50%*** | ***Kém***  ***0%*** |
| Thái độ tham gia tích cực | Nêu ý tưởng | 05 | Tích cực tìm kiếm và chủ động đưa ra ý tưởng mang tính mới | | Tìm kiếm và đưa ra được ý tưởng khá tốt | Chọn ý tưởng trong số được đề nghị | Không quan tâm lựa chọn ý tưởng |
| Lập kế hoạch thực hiện | 05 | Hoàn toàn hợp lý, không cần điều chỉnh | | Khá hợp lý, điều chỉnh chút ít theo góp ý | Chưa hợp lý, có điều chỉnh theo góp ý | Không hợp lý và không điều chỉnh theo góp ý |
| Quá trình thực hiện project nghiêm túc | Giai đoạn chuẩn bị | 10 | Chuẩn bị tốt mọi điều kiện cho việc thực hiện project, có thể khởi động ngay | | Chuẩn bị được đa số điều kiện cho việc thực hiện, có thể khởi động và bổ sung sau | Chuẩn bị được một số điều kiện cho việc thực hiện nhưng cần bổ sung thêm mới có thể khởi động | Không chuẩn bị được điều kiện nào |
| Giai đoạn thực hiện | 10 | Thực hiện hoàn toàn đúng phương pháp | | Thực hiện khá đúng phương pháp, sai sót nhỏ và có sửa chữa | Thực hiện tương đối đúng phương pháp, sai sót quan trọng và có sửa chữa | Thực hiện không đúng phương pháp, sai sót không sửa chữa |
| 10 | Triển khai đúng kế hoạch | | Triển khai khá đúng kế hoạch, có chậm trễ nhưng không gây ảnh hưởng | Triển khai tương đối đúng kế hoạch, có chậm trễ gây ảnh hưởng nhưng khắc phục được | Triển khai chậm trễ, gây ảnh hưởng không khắc phục được |
| Mức độ đạt được mục tiêu thành phần | 20 | * Giai đoạn 1: Nhận đề tài và xây dựng kịch bản * Giai đoạn 2: Tìm hiểu và thử nghiệm các phương pháp giải quyết kịch bản đề tài. Các thành viên trình bày các phương pháp hoặc ý tưởng và tiếp nhận góp ý từ giảng viên * Giai đoạn 3: Chọn phương pháp tối ưu, xây dựng môi trường thực nghiệm, kiểm tra các tài nguyên và kết nối theo kịch bản của đề tài * Giai đoạn 4: Thực nghiệm và xây dựng báo cáo * Giai đoạn 5: Hoàn thiện báo cáo và nộp sản phẩm | | | | | |
| Báo cáo kết quả bằng văn bản rõ ràng | Nội dung báo cáo | 10 | * Báo cáo tiến trình thực hiện * Thuyết minh sản phẩm * Trả lời câu hỏi/Giải thích đề tài * Kết luận và kiến nghị * Bài học rút ra | | | | | |
| Trình bày báo cáo | 10 | * Format nhất quán * Văn phong phù hợp * Demo dễ hiểu | | | | | |
| Chất lượng sản phẩm đáp ứng các tiêu chí đề ra |  | 20 | * Hoàn thành các tiêu chí đề ra của đề tài | | | | | |
| **ĐIỂM PROJECT NHÓM** | ĐIỂM CỦA NHÓM:  GÓP Ý LĨNH VỰC CẦN CẢI TIẾN:  - Về phương pháp học tập, suy nghĩ, xử lý vấn đề:  - Về kiến thức, kỹ năng, thái độ:  - Khác:  - Phương pháp cải tiến:  - Nơi có thể cung cấp sự hỗ trợ:  TÊN HỌ VÀ CHỮ KÝ XÁC NHẬN CỦA GV:  Hồ Ngọc Trung Kiên Nguyễn Kim Duy | | | | | | | |
| **Phần 2: Đánh giá cá nhân làm việc nhóm** | | | | | | | | | |
| **Làm việc nhóm**  *Hướng dẫn: Nhóm họp thảo luận đánh giá lẫn nhau. Kết quả này sẽ được sử dụng để qui đổi từ điểm của nhóm thành điểm của mỗi cá nhân, tuỳ theo % mỗi cá nhân đạt được*   1. **Thành viên có tham gia họp đầy đủ: 40%**   Đầy đủ: 40%  Vắng họp dưới 2 lần: 20%  Vắng họp hơn 2 lần: 0%  **2) Thành viên nộp sản phẩm được giao đúng hạn: 40%**  Đúng hạn: 40%  Trễ dưới 2 ngày: 20%  Trễ trên 2 ngày: 0%  **3) Thành viên có tham gia giải quyết vấn đề, đóng góp ý kiến cải tiến: 20%**  Đóng góp đạt hiệu quả: 20%  Có quan tâm đóng góp: 10%  Không quan tâm: 0% | | | | | | | | | |
| **ĐIỂM THÀNH VIÊN NHÓM**  **(do nhóm ghi)** | | | | ***Danh sách thành viên của Nhóm:***   1. *(Tên họ): Nguyễn Hoàng Anh*   *Điểm làm việc nhóm (%): 100*   1. *(Tên họ): Nguyễn Minh Hiếu*   *Điểm làm việc nhóm (%): 100*   1. *(Tên họ):* Lê Hồng Lộc   *Điểm làm việc nhóm (%): 100*  *TÊN HỌ VÀ CHỮ KÝ XÁC NHẬN CỦA CÁC THÀNH VIÊN TRONG NHÓM:*  *Nguyễn Hoàng Anh*  *Nguyễn Minh Hiếu*  *Lê Hồng Lộc* | | | | | |
| **ĐIỂM CỦA CÁ NHÂN**  **(do GV ghi)** | | | | ***Danh sách thành viên của Nhóm:***   1. *(Tên họ): Nguyễn Hoàng Anh*   *Điểm cá nhân:* …   1. *(Tên họ): Nguyễn Minh Hiếu*   *Điểm cá nhân:* …   1. *(Tên họ): Lê Hồng Lộc*   *Điểm cá nhân:* …  *TÊN HỌ VÀ CHỮ KÝ XÁC NHẬN CỦA GV:*  Hồ Ngọc Trung Kiên Nguyễn Danh Minh Trí | | | | | |
| **Ghi nhận của GV**  *(phần này do GV ghi và lưu riêng để theo dõi, tư vấn cho cá nhân SV, không phổ biến cho lớp, nhóm)* | | | | **GÓP Ý CẢI TIẾN CHO CÁ NHÂN:**   1. **Tên SV: Nguyễn Hoàng Anh**   GÓP Ý LĨNH VỰC CẦN CẢI TIẾN:  - Về phương pháp học tập, suy nghĩ, xử lý vấn đề:  - Về kiến thức, kỹ năng, thái độ:  - Khác:  - Phương pháp cải tiến:  - Nơi có thể cung cấp sự hỗ trợ:   1. **Tên SV: Nguyễn Minh Hiếu**   GÓP Ý LĨNH VỰC CẦN CẢI TIẾN:  - Về phương pháp học tập, suy nghĩ, xử lý vấn đề:  - Về kiến thức, kỹ năng, thái độ:  - Khác:  - Phương pháp cải tiến:  - Nơi có thể cung cấp sự hỗ trợ:   1. **Tên SV: Lê Hồng Lộc**   GÓP Ý LĨNH VỰC CẦN CẢI TIẾN:  - Về phương pháp học tập, suy nghĩ, xử lý vấn đề:  - Về kiến thức, kỹ năng, thái độ:  - Khác:  - Phương pháp cải tiến:  - Nơi có thể cung cấp sự hỗ trợ: | | | | | |

# MỤC LỤC

[MỞ ĐẦU ii](#_Toc141272109)

[**1.** **Lý do chọn đề tài:** ii](#_Toc141272110)

[**2.** **Mục đích của đề tài:** ii](#_Toc141272111)

[**3.** **Phạm vi của đề tài:** iii](#_Toc141272112)

[MỤC LỤC viii](#_Toc141272131)

[DANH MỤC HÌNH x](#_Toc141272132)

[CHƯƠNG 1. MÔ TẢ BÀI TOÁN VÀ SƠ ĐỒ CHỨC NĂNG 1](#_Toc141272134)

[**1.1. Mô tả bài toán:** 1](#_Toc141272135)

[**1.1.1.** **Hiện trạng** 1](#_Toc141272136)

[**1.1.2. Chức năng và phạm vi dự án:** 1](#_Toc141272137)

[**1.2. Sơ đồ chức năng (Business and UseCase Diagram):** 4](#_Toc141272138)

[**1.3. Lược đồ CSDL:** 4](#_Toc141272139)

[**1.4. Vai trò các thành viên:** 7](#_Toc141272140)

[CHƯƠNG 2. TỔNG QUAN VỀ LẬP TRÌNH WINDOWS BẰNG C#.NET 10](#_Toc141272141)

[**2.1. Giới thiệu C#.NET:** 10](#_Toc141272142)

[**2.2. Một số Control chính sử dụng trong đề tài:** 10](#_Toc141272143)

[**2.2.1. Button:** 10](#_Toc141272144)

[**2.2.2. ListView** 12](#_Toc141272145)

[**2.2.3. TreeView** 13](#_Toc141272146)

[**2.4. Phương thức kết nối dữ liệu:** 16](#_Toc141272147)

[CHƯƠNG 3. HIỆN THỰC CHỨC NĂNG VÀ KIỂM THỬ 17](#_Toc141272148)

[**3.1. Hiện thực chức năng:** 17](#_Toc141272149)

[**3.1.1. Màn hình (Form) Đăng nhập** 17](#_Toc141272150)

[**3.1.2. Màn hình (Form) Đăng ký:** 19](#_Toc141272151)

[**3.1.3. Màn hình (Form) Đổi mật khẩu** 22](#_Toc141272152)

[**3.1.5. Màn hình (Form) Chính:** 26](#_Toc141272153)

[**3.1.6. Màn hình (Form) Quản lý tài khoản Giảng viên và Sinh viên:** 39](#_Toc141272154)

[**3.1.7. Màn hình (Form) Khoa, Ngành, Lớp:** 58](#_Toc141272155)

[**3.1.8. Màn hình (Form) Quản lý môn học:** 72](#_Toc141272156)

[**3.1.9. Màn hình (Form) Phân công giảng dạy môn học:** 89](#_Toc141272157)

[**3.1.10. Màn hình (Form) Quản lý phòng học:** 97](#_Toc141272158)

[**3.1.11. Màn hình (Form) Sắp lớp cho Sinh viên:** 118](#_Toc141272159)

[**3.1.12. Màn hình (Form) Quản lý khảo sát** 130](#_Toc141272160)

[**3.1.13. Màn hình (Form) Xem lịch dạy / Xem thời khoá biểu:** 140](#_Toc141272161)

[**3.1.14. Màn hình (Form) Quản lý / Chấm điểm sinh viên:** 154](#_Toc141272162)

[**3.1.15. Màn hình (Form) Xem môn học:** 168](#_Toc141272163)

[**3.1.16. Màn hình (Form) Xem khảo sát:** 173](#_Toc141272164)

[**3.1.17. Màn hình (Form) Đăng ký môn:** 181](#_Toc141272165)

[**3.1.18. Màn hình (Form) Khảo sát Giảng viên:** 196](#_Toc141272166)

[**3.2.** **Kiểm thử ứng dụng:** 206](#_Toc141272167)

[CHƯƠNG 4. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 217](#_Toc141272168)

[**4.1. Kết luận:** 217](#_Toc141272169)

[**4.1.1. Các chức năng ứng dụng đã làm được:** 217](#_Toc141272170)

[**4.1.2. Hạn chế:** 217](#_Toc141272171)

[**4.2. Hướng phát triển:** 217](#_Toc141272172)

[TÀI LIỆU THAM KHẢO 218](#_Toc141272173)

# DANH MỤC HÌNH

[Hình ảnh 1.2.1. Business Diagram Quản lý hoạt động dạy học 4](#_Toc141274822)

[Hình ảnh 1.3.1. Lược đồ EDR CSDL 7](#_Toc141274824)

[Hình ảnh 3.2.1. Form Đăng nhập 206](#_Toc141274853)

[Hình ảnh 3.2.2. Form Đăng ký 206](#_Toc141274854)

[Hình ảnh 3.2.3. Form Đổi mật khẩu 207](#_Toc141274855)

[Hình ảnh 3.2.4. Form Quên mật khẩu 207](#_Toc141274856)

[Hình ảnh 3.2.5. Form chính cho Quản trị 208](#_Toc141274857)

[Hình ảnh 3.2.6. Form chính cho Giảng viên 208](#_Toc141274858)

[Hình ảnh 3.2.7. Form chính dành cho Sinh viên 209](#_Toc141274859)

[Hình ảnh 3.2.8. Form Quản lý tài khoản 209](#_Toc141274860)

[Hình ảnh 3.2.9. Form Quản lý khoa, ngành, thêm lớp 210](#_Toc141274861)

[Hình ảnh 3.2.10. Form Quản lý môn học 210](#_Toc141274862)

[Hình ảnh 3.2.11. Form Phân công giảng dạy cho Giảng viên 211](#_Toc141274863)

[Hình ảnh 3.2.12. Form Quản lý và đăng ký phòng 211](#_Toc141274864)

[Hình ảnh 3.2.13. Form Sắp lớp cho Sinh viên 212](#_Toc141274865)

[Hình ảnh 3.2.14. Form Quản lý khảo sát 212](#_Toc141274866)

[Hình ảnh 3.2.15. Form Xem lịch dạy học 213](#_Toc141274867)

[Hình ảnh 3.2.16. Form Quản lý / Chấm điểm Sinh viên 213](#_Toc141274868)

[Hình ảnh 3.2.17. Form Xem môn học theo Khoa dành cho Giảng viên 214](#_Toc141274869)

[Hình ảnh 3.2.18. Form Xem khảo sát từ Sinh viên 214](#_Toc141274870)

[Hình ảnh 3.2.19. Form Đăng ký môn học 215](#_Toc141274871)

[Hình ảnh 3.2.20. Form Xem thời khoá biểu 215](#_Toc141274872)

[Hình ảnh 3.2.21. Form Xem các môn học theo Ngành dành cho Sinh 216](#_Toc141274873)

[Hình ảnh 3.2.22. Form Khảo sát Giảng viên 216](#_Toc141274874)

# CHƯƠNG 1. MÔ TẢ BÀI TOÁN VÀ SƠ ĐỒ CHỨC NĂNG

## **Mô tả bài toán:**

1. Hiện trạng:

* Hệ thống quản lý hoạt động dạy học dành cho trường đại học nhằm phục vụ nhu cầu đơn giản hoá các thao tác quản lý trong trường học. Vì thế mà chúng tôi đã triển khai và hoàn thiện dự án này nhằm giải quyết những vấn đề được đặt ra.

### ***1.1.2. Chức năng và phạm vi dự án:***

* Đối với người dùng quản trị sẽ sở hữu quyền lực lớn nhất trong hệ thống:

+ Quản lý tài khoản của giảng viên và sinh viên trong trường: có thể cấp phát, chỉnh sửa hay xoá tài khoản giảng viên và sinh viên một cách dễ dàng, đồng thời cung cấp chức năng tìm kiếm, sắp xếp hay lọc danh sách để tăng tích tiện nghi cho thao tác quản lý.

+ Quản lý khoa, ngành, lớp: có thể thêm mới khoa, ngành hay lớp, có thể cập nhật hoặc xoá chúng khi cần thiết. Ngoài ra hệ thống còn cho phép tạo nhanh cả khoa, ngành và lớp nhanh giúp rút ngắn thời gian thao tác.

+ Quản lý môn học: có thể thêm môn học vào ngành học bất kì, nhập dữ liệu môn, cập nhật hoặc xoá đi môn học khi cần thiết. Hệ thống cung cấp chức năng tìm kiếm, bộ lọc giúp thuận tiện hơn trong khâu quản lý môn học

+ Phân công giảng viên: cho phép phân công các giảng viên tham gia giảng dạy môn học, mở lớp mới cho giảng viên. Hệ thống cung cấp bộ lọc cho phép lọc các môn học theo ngành, khoa để nhanh chống hơn cho phân công, khi lọc cũng sẽ chỉ hiện thị những giảng viên thuộc cùng khoa đó. Ngoài ra hệ thống cũng cho phép huỷ phân công khi cần.

+ Quản lý phòng học: có thể thêm mới phòng học, với mỗi phòng học cho phép đăng ký các lớp vào phòng đó. Khi đăng ký các lớp vào phòng học, hệ thống cung cấp bộ tìm kiếm phòng theo tên hay theo giờ phòng trống để tiện lợi cho việc chọn phòng. Trường hợp đăng ký phòng bị trùng giờ hệ thống sẽ cảnh báo và không cho phép đăng ký phòng đó (kể cả khi lớp đó đăng ký trùng giờ giấc tại phòng khác cũng sẽ cảnh báo vì một lớp không thể dạy cùng lúc hai phòng). Ngoài ra ta cũng có thể chỉnh sửa phòng, chỉnh sửa thông tin đăng ký phòng hay xoá phòng, xoá đăng ký phòng dễ dàng.

+ Sắp lớp cho sinh viên: sắp xếp cho sinh viên tham gia vào các lớp học hiện có, hệ thống sẽ hiển thị sinh viên phân theo ngành đồng bộ với lớp để giúp giảm tối đa sự khó khăn trong việc sắp lớp cho quản trị. Đồng thời cung cấp chức năng tìm kiếm theo tên lớp để tìm lớp nhanh chống và bộ lọc lớp theo khoa hoặc ngành. Ngoài ra có thể sắp xếp nhanh sinh viên vào lớp bằng lựa chọn phân bổ tự động, hệ thống sẽ tự sắp tất cả sinh viên cùng ngành vào từng lớp, đầy lớp này sang lớp khác đến khi hết sinh viên hoặc không còn lớp để thêm vào nữa và chức năng loại bỏ hay loại bỏ cả lớp hay loại bỏ toàn bộ sinh viên khỏi các lớp khi cần.

+ Quản lý khảo sát môn học từ sinh viên: Hệ thống hiển thị các bài khảo sát của sinh viên cho từng môn học của giảng viên dạy, cung cấp tính năng lọc giảng viên. Thông tin khảo sát được hiển thị rõ ràng, có thể xoá bài khảo sát ấy hoặc xoá toàn bộ bài khảo sát liên quan tới giảng viên nào đó hay xoá toàn bộ khảo sát trên hệ thống.

* Đối với giảng viên sẽ sở hữu các chức năng sau:

+ Xem lịch dạy học: Hệ thống sẽ lấy dữ liệu từ đăng ký phòng để lên lịch dạy cho giảng viên, lịch dạy chia qua từng tuần, có thể xem từng tuần, giảng viên dạy những tiết nào chi tiết.

+ Quản lý sinh viên và chấm điểm: Hiển thị danh sách sinh viên thuộc lớp mà mình đang làm cố vấn, có thể loại sinh viên khỏi lớp. Đồng thời hiển thị các lớp học giảng viên đang giảng dạy, với mỗi lớp học sẽ hiển thị danh sách sinh viên thuộc lớp đó cho phép chấm điểm hay gỡ bỏ điểm đã chấm cho sinh viên

+ Xem các môn học thuộc khoa của mình: hiển thị tên các môn học theo từng năm thuộc khoa đang dạy, có thể

+ Xem khảo sát: Hiển thị các bài khảo sát mà sinh viên gửi cho giảng viên sau khi kết thúc một môn học, cho phép lọc theo mã lớp

* Đối với sinh viên sẽ sở hữu các chức năng:

+ Đăng ký môn: hiển thị các môn học đang mở lớp dạy với thông tin chi tiết, bao gồm cả lịch học. Hệ thống hiển thị môn học nợ môn được xác định bằng việc sinh viên đó đã bỏ qua môn phải học của học kì / năm trước hay đã học nhưng không đủ điểm qua môn. Ngoài ra hệ thống còn cho phép in phiếu các môn đã đăng ký dưới dạng file .txt.

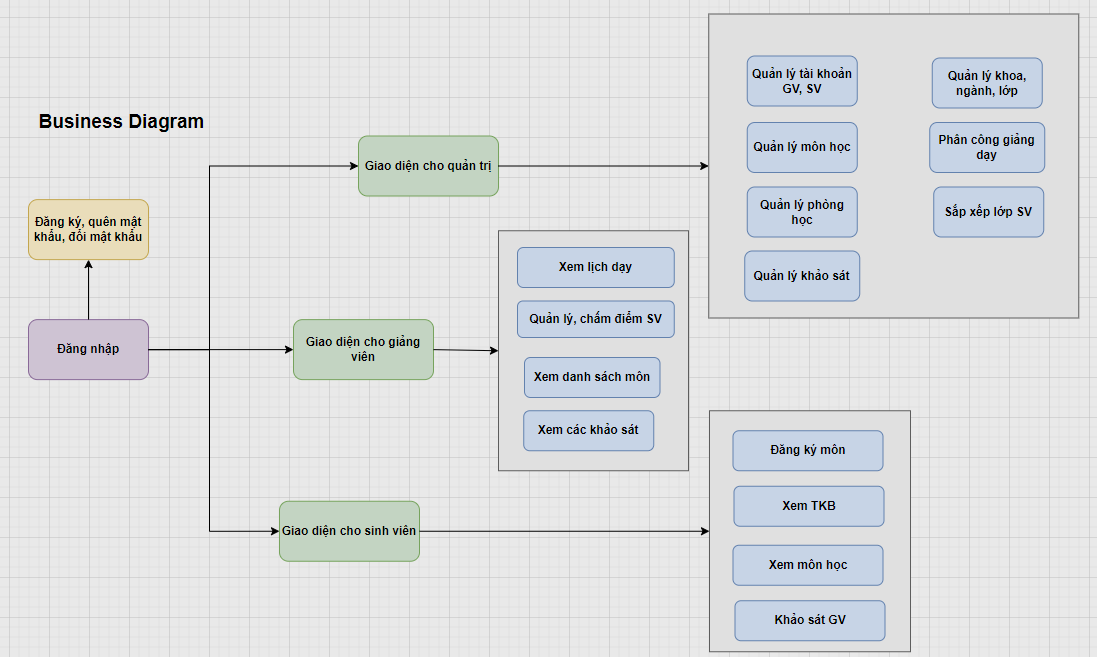
+ Xem thời khoá biểu: xem lịch học qua từng tuần của sinh viên, được lấy từ lịch đăng ký phòng của các môn mà sinh viên đã đăng ký. Hệ thống cho phép xem lại cả lịch học của các học kì / năm đã qua.

+ Xem môn học: xem các môn học thuộc ngành của sinh viên đang học, các môn hiển thị theo từng năm

+ Khảo sát giảng viên: hiển thị các lớp học sinh viên đang học, với các lớp học mà sinh viên đã được chấm điểm sẽ cho phép gửi bài khảo sát cho lớp đó. Sinh viên có thể chỉnh sửa bài khảo sát của mình.

## **1.2. Sơ đồ chức năng (Business and UseCase Diagram):**

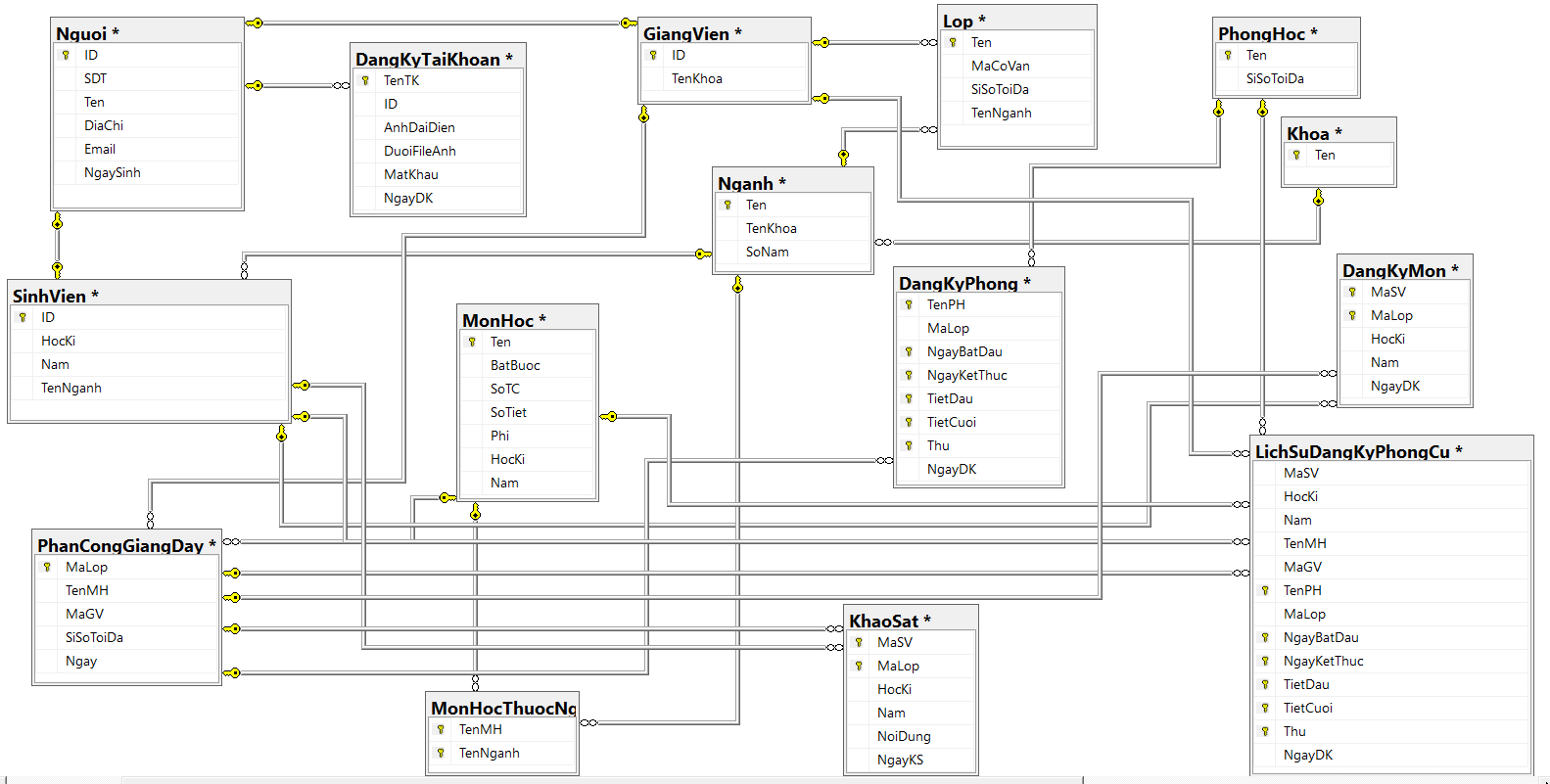
* **Business Diagram Quản lý hoạt động dạy học:**



#### Hình ảnh 1.2.1. Business Diagram Quản lý hoạt động dạy học

## **1.3.** **Lược đồ CSDL:**

* **Bảng Khoa:**
* Ten (NVARCHAR(100)): Tên khoa, Khoá chính
* **Bảng Ngành:**
* ID (INT) IDENTITY: Khoá chính
* Ten (NVARCHAR(100)) UNIQUE: Tên ngành
* TenKhoa (NVARCHAR(100)) Khoá ngoại tham chiếu tới Ten Khoa
* SoNam (INT): Số năm học yêu cầu cho ngành
* **Bảng Phòng học:**
* Ten (NVARCHAR(100)): Tên phòng, Khoá chính
* SiSoToiDa (INT): Sức chứa tối đa
* **Bảng Người:**
* ID (INT) IDENTITY: Khoá chính
* SDT (NVARCHAR(100)): Số điện thoại
* Ten (NVARCHAR(100)): Tên người dùng
* DiaChi (NVARCHAR(100)): Địa chỉ
* Email (NVARCHAR(100)): Địa chỉ Email
* NgaySinh (DATETIME): Ngày sinh
* Bảng Giảng viên:
* ID (INT): Khoá chính và khoá ngoại tham chiếu tới ID Người
* TenKhoa (NVARCHAR(100)): Khoá ngoại tham chiếu tới Ten Khoa
* **Bảng Lớp:**
* Ten (NVARCHAR(100)): Tên lớp, Khoá chính
* MaCoVan (NVARCHAR(100)): Mã cố vấn của lớp, khoá ngoại tham chiếu tới ID Giảng viên
* SiSoToiDa (INT): Sỉ số tối đa của lớp
* TenNganh (NVARCHAR(100)): Khoá ngoại tham chiếu tới Ten Ngành
* **Bảng Sinh viên:**
* ID (INT): Khoá chính, Khoá ngoại tham chiếu tới ID Sinh viên
* HocKi (INT): Học kì hiện tại
* Nam (INT): Năm học hiện tại
* TenNganh (NVARCHAR(100)): Khoá ngoại tham chiếu tới Ten Ngành
* **Bảng Môn học:**
* Ten (NVARCHAR(100)): Tên môn học, Khoá chính
* BatBuoc (BIT): Xác định môn học bắt buộc hay không, 1: bắt buộc, 0: ngược lại
* SoTC (INT): Số tín chỉ của môn
* SoTiet (INT): Số tiết phải học
* Phi (FLOAT): Học phí của môn
* HocKi (INT): Học kì của môn
* Nam (INT): Năm học của môn
* **Bảng Môn học thuộc ngành:**
* TenMH (NVARCHAR(100)): Khoá ngoại tham chiếu tới Ten Môn học
* TenNganh (NVARCHAR(100)): Khoá ngoại tham chiếu tới Ten Ngành
* TenMH và TenNganh là cặp khoá chính
* **Bảng Phân công giảng dạy:**
* ID (INT) IDENTITY: Khoá chính
* TenMH (NVARCHAR(100)): Khoá ngoại tham chiếu tới Ten Môn học
* MaGV (INT): Khoá ngoại tham chiếu tới ID Giảng viên
* SiSoToiDa (INT): Sỉ số tối đa của lớp
* Ngay (DATETIME): Ngày phân công
* **Bảng Đăng ký phòng:**
* TenPH (INT): Khoá ngoại tham chiếu tới Ten Phòng học
* MaLop (INT): Khoá ngoại tham chiếu tới ID Phân công giảng dạy
* NgayBatDau (DATETIME): Ngày bắt đầu đăng ký
* NgayKetThuc (DATETIME): Ngày kết thúc đăng ký
* TietDau (INT): Tiết bắt đầu
* TietCuoi (INT): Tiết kế thúc
* Thu (NVARCHAR(100)): Thứ đăng ký
* NgayDK (DATETIME): Ngày tạo đăng ký
* TenPH, NgayBatDau, NgayKetThuc, TietDau, TietCuoi và Thu là bộ khoá chính
* **Bảng Lịch sử Đăng ký phòng:**
* MaSV (INT): Khoá ngoại tham chiếu tới ID Sinh viên
* HocKi (INT): Học kì đăng ký
* Năm (INT): Năm học đăng ký
* TenMH (NVARCHAR(100)): Khoá ngoại tham chiếu tới Ten Môn học
* MaGV (INT): Khoá ngoại tham chiếu tới ID Giảng viên
* TenPH (INT): Khoá ngoại tham chiếu tới Ten Phòng học
* MaLop (INT): Khoá ngoại tham chiếu tới ID Phân công giảng dạy
* NgayBatDau (DATETIME): Ngày bắt đầu đăng ký
* NgayKetThuc (DATETIME): Ngày kết thúc đăng ký
* TietDau (INT): Tiết bắt đầu
* TietCuoi (INT): Tiết kế thúc
* Thu (NVARCHAR(100)): Thứ đăng ký
* NgayDK (DATETIME): Ngày tạo đăng ký
* TenPH, NgayBatDau, NgayKetThuc, TietDau, TietCuoi và Thu là bộ khoá chính
* **Bảng Đăng ký môn:**
* MaSV (INT): Khoá ngoại tham chiếu tới ID Sinh viên
* MaLop (INT): Khoá ngoại tham chiếu tới ID Phân công giảng dạy
* HocKi (INT): Học kì đăng ký
* Nam (INT): Năm học đăng ký
* NgayDK (DATETIME): Ngày đăng ký
* MaSV và MaLop là cặp khoá chính
* **Bảng Phân bổ Sinh viên:**
* ID (INT) IDENTITY: Khoá chính
* MaSV (INT): Khoá ngoại tham chiếu tới ID Sinh viên
* TenLop (NVARCHAR(100)): Khoá ngoại tham chiếu tới Ten Lớp
* Ngay (DATETIME): Ngày phân bổ
* **Bảng Ghi điểm:**
* MaSV (INT): Khoá ngoại tham chiếu tới ID Sinh viên
* MaLop (INT): Khoá ngoại tham chiếu tới ID Phân công giảng dạy
* KTDK (FLOAT): Điểm kiểm tra định kì
* KTHP (FLOAT): Điểm kiểm tra học phần
* TiLeKTDK (INT): Tỉ lệ % điểm định kì
* HocKi (INT): Điểm cho học kì
* Nam (INT): Điểm cho năm học
* **Bảng Đăng ký tài khoản:**
* TenTK (NVARCHAR(100): Tên tài khoản, Khoá chính
* ID\_NguoiDung (INT): Khoá ngoại tham chiếu tới ID Người
* AnhDaiDien (IMAGE): Ảnh đại diện
* DuoiFileAnh (NVARCHAR(100)): Đuôi file ảnh đại diện
* MatKhau (NVARCHAR(100)): Mật khẩu
* NgayDK (DATETIME): Ngày đăng ký
* **Bảng Khảo sát:**
* MaSV (INT): Khoá ngoại tham chiếu tới ID Sinh viên
* MaLop (INT): Khoá ngoại tham chiếu tới Phân công giảng dạy
* HocKi (INT): Học kì khảo sát
* Nam (INT): Năm học khảo sát
* NoiDung (NVARCHAR(100)): Nội dung khảo sát
* NgayKS (DATETIME): Ngày gửi khảo sát
* MaSV, MaLop là cặp khoá ngoại



#### Hình ảnh 1.3.1. Lược đồ EDR CSDL

## **1.4. Vai trò các thành viên:**

**Tên nhóm : Nhóm 1**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **MSSV** | **Họ tên** | **Vai trò** | **Nội dung thực hiện** |
| 1 | **212482010025** | **Nguyễn Hoàng Anh** | **Nhóm trưởng, chỉ đạo, code chính** | **Cố vấn / Hướng dẫn cho toàn bộ Forms và quá trình làm bài. Làm các Forms Quản lý Khoa, ngành, thêm lớp, Quản lý môn học, Quản lý khảo sát, Xem thời khoá biểu. Vẽ các Diagrams. Thiết kế SQL.** |
| 2 | **2124802010706** | **Nguyễn Minh Hiếu** | **Hỗ trợ code, đóng góp ý tưởng, làm báo cáo** | **làm Word, làm các Forms Đăng nhập, Đăng ký, Quên mật khẩu, Đổi mật khẩu, Xem lịch dạy học. Tham gia 1 phần vào các Forms Giao diện chính, Quản lý tài khoản, Phân công giảng dạy, Quản lý phòng học, Sắp lớp Sinh viên, Quản lý / Chấm điểm sinh viên, Xem các môn học và Nạp dữ liệu vào SQL** |
| 3 | **2124802010747** | **Lê Hồng Lộc** | **Hỗ trợ code, đóng góp ý tưởng, làm báo cáo** | **làm Slides, làm các Forms Xem các khảo sát, Khảo sát Giảng viên. Đóng góp 1 phần vào các Forms Giao diện chính, Quản lý tài khoản, Quản lý phòng học, Sắp lớp Sinh viên, Xem môn học, Đăng ký môn, Nạp dữ liệu vào SQL** |



# CHƯƠNG 2. TỔNG QUAN VỀ LẬP TRÌNH WINDOWS BẰNG C#.NET

## **2.1. Giới thiệu C#.NET:**

C#.NET là một ngôn ngữ lập trình hiện đại, đa năng và được phát triển bởi Microsoft. Nó được thiết kế để hỗ trợ việc phát triển các ứng dụng trên nền tảng Windows, nhưng cũng có thể được sử dụng để phát triển các ứng dụng trên nhiều nền tảng khác nhau, bao gồm cả macOS và Linux.

C#.NET được tạo ra với mục đích phát triển các ứng dụng đa nền tảng, có khả năng kết hợp với các công nghệ khác của Microsoft như .NET Framework hoặc .NET Core. Các ứng dụng được viết bằng C#.NET có thể chạy trên nhiều nền tảng và hệ điều hành khác nhau, bao gồm desktop, web, di động và các thiết bị IoT.

C#.NET có cú pháp tương tự như ngôn ngữ lập trình Java và C++, với cú pháp đơn giản và hướng đối tượng. Các ứng dụng được viết bằng C#.NET có thể được biên dịch thành mã máy hoặc chạy thông qua trình thông dịch.

Các tính năng của C#.NET bao gồm tính năng quản lý bộ nhớ tự động, tính năng quản lý ràng buộc kiểu dữ liệu, hỗ trợ đa luồng, hỗ trợ lập trình hướng sự kiện, và một số tính năng khác. C#.NET cũng hỗ trợ nhiều thư viện và framework phong phú, giúp cho việc phát triển ứng dụng trở nên dễ dàng và nhanh chóng hơn.

## **2.2. Một số Control chính sử dụng trong đề tài:**

### ***2.2.1. Button:***

* **Phương thức sự kiện:**

private void btnSignIn\_Click(object sender, EventArgs e)

{

foreach (Control control in gbSignIn.Controls)

{

if (control is TextBox)

{

if (string.IsNullOrEmpty(control.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Bạn chưa nhập đầy đủ thông tin!!!");

control.Focus();

return;

}

}

}

if (!Account.isExistsName(txtNameAcct.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Tên tài khoản không tồn tại!");

txtNameAcct.Focus();

return;

}

if (!Account.isRightPassword(txtNameAcct.Text, txtPassword.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Mật khẩu không đúng!");

txtNameAcct.Focus();

return;

}

Hide();

Person info = Account.getInfoByName(txtNameAcct.Text);

string typeAccount;

if (info is Student)

{

typeAccount = Account.typeSV;

}

else if (info is Teacher)

{

typeAccount = Account.typeGV;

}

else

{

typeAccount = Account.typeQT;

}

frmManageSchool frm = new frmManageSchool(typeAccount, txtNameAcct.Text );

frm.ShowDialog();

Close();

}

* **Gắn sự kiện:**

this.btnSignUp.Click += new System.EventHandler(this.btnSignUp\_Click);

### ***2.2.2. ListView***

* **Phương thức sự kiện:**

private void lvInfo\_SelectedIndexChanged(object sender, EventArgs e)

{

if (lvInfo.SelectedIndices.Count == 0)

{

currentPerson = null;

pnInfo.Enabled = false;

defaultGBInfo();

return;

}

int index = lvInfo.SelectedIndices[0];

ListViewItem lv = lvInfo.Items[index];

string ID = lv.SubItems[0].Text;

currentPerson = people[index];

Account account = Account.getInfoByID(ID);

cbbTypeAcct.SelectedIndex = -1;

if (lv.SubItems[2].Text == "Sinh viên")

{

cbbTypeAcct.SelectedIndex = 1;

}

else

{

cbbTypeAcct.SelectedIndex = 0;

}

txtName.Text = currentPerson.getFullName().getFullName();

txtPhone.Text = currentPerson.getPhone();

txtEmail.Text = currentPerson.getEmail();

txtAddress.Text = currentPerson.getAddress();

txtID.Text = currentPerson.getID();

txtNameAccount.Text = account.getName();

txtPassword.Text = account.getPassword();

dtpBirth.Value = currentPerson.getBirth().getDate();

cbbTypeAcct.Enabled = false;

pnInfo.Enabled = true;

}

* **Gắn sự kiện:**

this.lvInfo.SelectedIndexChanged += new System.EventHandler(this.lvInfo\_SelectedIndexChanged);

### ***2.2.3. TreeView***

* **Phương thức sự kiện:**

private void tvKhoa\_AfterSelect(object sender, TreeViewEventArgs e)

{

currentNode = e.Node;

string tag = (string)currentNode.Tag;

clearText();

gbInfo.Enabled = true;

btnAccept.Enabled = true;

if (tag == "0")

{

btnDelete.Enabled = false;

pnClass.Enabled = false;

return;

}

btnDelete.Enabled = true;

pnClass.Enabled = true;

if (tag == "1")

{

txtNameKhoa.Text = currentNode.Text;

}

else if (tag == "2")

{

txtNameKhoa.Text = currentNode.Parent.Text;

txtNameMajor.Text = currentNode.Text;

Major major = Major.getInfo(txtNameMajor.Text);

txtCountYear.Text = major.getCountYear().ToString();

}

else

{

txtNameKhoa.Text = currentNode.Parent.Parent.Text;

txtNameMajor.Text = currentNode.Parent.Text;

txtNameClass.Text = currentNode.Text;

Khoa khoa = new Khoa(txtNameKhoa.Text);

Major major = Major.getInfo(txtNameMajor.Text);

Class classs = Class.getInfo(txtNameClass.Text);

txtCountYear.Text = major.getCountYear().ToString();

txtCountMax.Text = classs.getMaxCount().ToString();

List<Class> lstClass = Class.getAllClass();

List<Teacher> teachers = khoa.getTeachers();

cbbMaGV.Items.Add(classs.getMaGV());

for (int i = 0; i < teachers.Count; i++)

{

bool isOk = true;

for (int j = 0; j < lstClass.Count; j++)

{

if (teachers[i].getID() == lstClass[j].getMaGV())

{

isOk = false;

break;

}

}

if (isOk)

{

cbbMaGV.Items.Add(teachers[i].getID());

break;

}

}

cbbMaGV.SelectedIndex = 0;

}

}

* **Gắn sự kiện:**

this.tvKhoa.AfterSelect += new System.Windows.Forms.TreeViewEventHandler(this.tvKhoa\_AfterSelect);

## **2.4. Phương thức kết nối dữ liệu :**

## ADO.NET: ADO.NET là một thư viện trong .NET Framework cho phép kết nối và truy xuất dữ liệu từ các nguồn dữ liệu khác nhau như SQL Server, Oracle, MySQL, PostgreSQL, v.v. Thư viện này cung cấp các đối tượng như SqlConnection, SqlCommand, SqlDataReader, SqlDataAdapter để thực hiện các thao tác trên cơ sở dữ liệu.

* LINQ to SQL: LINQ to SQL là một công nghệ ORM của Microsoft, cho phép truy xuất cơ sở dữ liệu bằng cách sử dụng các truy vấn LINQ. LINQ to SQL hỗ trợ truy xuất cơ sở dữ liệu từ SQL Server.

# CHƯƠNG 3. HIỆN THỰC CHỨC NĂNG VÀ KIỂM THỬ

## **3.1. Hiện thực chức năng:**

### ***3.1.1. Màn hình (Form) Đăng nhập***

* Mô tả màn hình: Form dành cho việc đăng nhập tài khoản bằng tên đăng nhập và mật khẩu, trên màn hình có thêm các nút cho phép xử lý quên mật khẩu hay đổi mật khẩu
* Hiện thực các phương thức xử lý sự kiện (Event Handlers):
* Click nút Đăng nhập:

private void btnSignIn\_Click(object sender, EventArgs e)

{

foreach (Control control in gbSignIn.Controls)

{

if (control is TextBox)

{

if (string.IsNullOrEmpty(control.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Bạn chưa nhập đầy đủ thông tin!!!");

control.Focus();

return;

}

}

}

if (!Account.isExistsName(txtNameAcct.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Tên tài khoản không tồn tại!");

txtNameAcct.Focus();

return;

}

if (!Account.isRightPassword(txtNameAcct.Text, txtPassword.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Mật khẩu không đúng!");

txtNameAcct.Focus();

return;

}

Hide();

Person info = Account.getInfoByName(txtNameAcct.Text);

string typeAccount;

if (info is Student)

{

typeAccount = Account.typeSV;

}

else if (info is Teacher)

{

typeAccount = Account.typeGV;

}

else

{

typeAccount = Account.typeQT;

}

frmManageSchool frm = new frmManageSchool(typeAccount, txtNameAcct.Text );

frm.ShowDialog();

Close();

}

* Click nút Đăng ký:

private void btnSignUp\_Click(object sender, EventArgs e)

{

Hide();

frmSignUp frm = new frmSignUp();

frm.ShowDialog();

Show();

}

* Click nút Đổi mật khẩu:

private void btnChangePassword\_Click(object sender, EventArgs e)

{

Hide();

frmChangePassword frm = new frmChangePassword();

frm.ShowDialog();

Show();

}

* Click chọn Quên mật khẩu:

private void lblForgotPassword\_Click(object sender, EventArgs e)

{

Hide();

frmForgotPassword frm = new frmForgotPassword();

frm.ShowDialog();

Show();

}

* Click Tắt / Bật hiển thị mật khẩu:

private void ckbShowPassword\_CheckedChanged(object sender, EventArgs e)

{

txtPassword.UseSystemPasswordChar = !ckbShowPassword.Checked;

}

* Hiện thực các phương thức khác:

### ***3.1.2. Màn hình (Form) Đăng ký:***

* Mô tả màn hình: Form cho phép đăng ký tài khoản cho Quản trị (tài khoản Giảng viên và Sinh viên do Quản trị cấp nên không đăng ký tại đây được). Thông tin đăng ký bao gồm tên tài khoản, mật khẩu, họ tên, ngày sinh, số điện thoại, email, địa chỉ, mã ID.
* Hiện thực các phương thức xử lý sự kiện (Event Handlers):
* Click nút Đăng ký:

private void btnSignUp\_Click(object sender, EventArgs e)

{

foreach (Control control in gbInfoAccount.Controls)

{

if (control is TextBox)

{

if (string.IsNullOrEmpty(control.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Bạn chưa nhập đầy đủ thông tin tài khoản!");

control.Focus();

return;

}

}

}

foreach (Control control in gbInfoPerson.Controls)

{

if (control is TextBox)

{

if (string.IsNullOrEmpty(control.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Bạn chưa nhập đầy đủ thông tin cá nhân!");

control.Focus();

return;

}

}

}

if (txtPassword.Text != txtRePassword.Text)

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Mật khẩu nhập lại không khớp!");

txtRePassword.Focus();

return;

}

if (!Person.isPhoneValid(txtPhone.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Số điện thoại không khả dụng!");

txtRePassword.Focus();

return;

}

if (!Person.isEmailValid(txtEmail.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Email không khả dụng!");

txtRePassword.Focus();

return;

}

Person info = new Person(txtID.Text, txtPhone.Text, txtName.Text, txtAddress.Text, txtEmail.Text, new Date(dtpBirth.Value));

if (Account.signUp(txtNameAcct.Text, txtPassword.Text, info))

{

Close();

}

}

* Click Bật / Tắt hiển thị mật khẩu:

private void ckbShowPassword\_CheckedChanged(object sender, EventArgs e)

{

txtPassword.UseSystemPasswordChar = !ckbShowPassword.Checked;

txtRePassword.UseSystemPasswordChar = !ckbShowPassword.Checked;

}

### ***3.1.3. Màn hình (Form) Đổi mật khẩu***

* Mô tả màn hình: Form cho phép người dùng đổi mật khẩu, thông tin cần điền bao gồm tên tài khoản, mật khẩu cũ và mật khẩu mới.
* Hiện thực các phương thức xử lý sự kiện (Event Handlers):
* Click nút Xác nhận:

private void btnAccept\_Click(object sender, EventArgs e)

{

foreach (Control control in gbInfo.Controls)

{

if (control is TextBox && string.IsNullOrEmpty(control.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Bạn chưa nhập đủ thông tin!");

control.Focus();

return;

}

}

if (!Account.isPasswordValid(txtNewPassword.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Mật khẩu mới không hợp lệ!");

txtNewPassword.Focus();

return;

}

if (txtNewPassword.Text != txtReNewPassword.Text)

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Mật khẩu nhập lại không khớp!");

txtReNewPassword.Focus();

return;

}

if (!Account.isExistsName(txtNameAcct.Text))

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Tên tài khoản không tồn tại!");

txtNameAcct.Focus();

return;

}

Account acct = new Account(txtNameAcct.Text);

if (acct.getPassword() != txtOldPassword.Text)

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Mật khẩu không chính xác!");

txtOldPassword.Focus();

return;

}

if (Account.updateField(acct.getID(), "TenTK", txtNewPassword.Text))

{

MessageInfo.makeMessage("Information", "Chúc mừng", "Đổi mật khẩu thành công!");

Close();

}

else

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Đổi mật khẩu thất bại!");

}

}

* KeyPress cho nhập tên đăng nhập:

private void txt\_keyPress(object sender, KeyPressEventArgs e)

{

if (e.KeyChar != (char)Keys.Enter)

{

if (e.KeyChar == ' ' || char.IsSymbol(e.KeyChar) || char.IsSymbol(e.KeyChar) || (!char.IsLetterOrDigit(e.KeyChar) && e.KeyChar > 30))

{

e.Handled = true;

}

}

}

* Click Bật / Tắt Hiển thị mật khẩu:

private void ckbShowPassword\_CheckedChanged(object sender, EventArgs e)

{

txtOldPassword.UseSystemPasswordChar = !ckbShowPassword.Checked;

txtNewPassword.UseSystemPasswordChar = !ckbShowPassword.Checked;

txtReNewPassword.UseSystemPasswordChar = !ckbShowPassword.Checked;

}

3.1.4. Màn hình (Form) Quên mật khẩu:

* Mô tả màn hình: Form dành cho xử lý việc quên mật khẩu của người dùng, người dùng cần dùng SĐT đăng ký tài khoản để đổi mật khẩu mới khi quên mật khẩu.
* Hiện thực các phương thức xử lý sự kiện (Event Handlers):
* Click nút xác nhận:

private void selectAccept(object sender, EventArgs e)

{

foreach (Control control in gbInfo.Controls)

{

if (control is TextBox && string.IsNullOrEmpty(control.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Bạn chưa nhập đủ thông tin!");

control.Focus();

return;

}

}

if (!Account.isPasswordValid(txtNewPassword.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Mật khẩu phải tối thiểu 8 kí tự, bao gồm chữ cái in hoa, in thường, số và kí tự đặc biệt!!!");

return;

}

if (txtRePassword.Text != txtNewPassword.Text)

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Mật khẩu nhập lại không trùng khớp!");

}

try

{

string id = (string)Person.getFieldPerson("ID", "SDT", txtPhone.Text);

Account acct = Account.getInfoByID(id);

Account.updateField(acct.getID(), "MatKhau", txtNewPassword.Text);

MessageInfo.makeMessage("Information", "Chúc mừng", "Đã đổi mật khẩu thành công!!");

Close();

}

catch

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Không tồn tại tài khoản có số điện thoại đã nhập!!");

}

}

* KeyPress cho nhập số điện thoại:

private void txtPhone\_KeyPress(object sender, KeyPressEventArgs e)

{

if (char.IsLetter(e.KeyChar) || char.IsSymbol(e.KeyChar))

{

e.Handled = true;

}

}

* Click Bật / Tắt hiển thị mật khẩu:

private void ckbShowPassword\_CheckedChanged(object sender, EventArgs e)

{

txtNewPassword.UseSystemPasswordChar = !ckbShowPassword.Checked;

txtRePassword.UseSystemPasswordChar = !ckbShowPassword.Checked;

}

* Hiện thực các phương thức khác:

### ***3.1.5. Màn hình (Form) Chính:***

* Mô tả màn hình: Form giao diện chính chứa các chức năng có thể dùng của Quản trị hay Giảng viên hay Sinh viên (Mỗi loại tài khoản sẽ có các chức năng phù hợp với quyền mỗi người).
* Hiện thực các phương thức xử lý sự kiện (Event Handlers):
* Click nút Đổi ảnh đại diện:

private void selectChangeAvt(object sender, MouseEventArgs e)

{

OpenFileDialog openDialog = new OpenFileDialog()

{

Multiselect = false,

Filter = "Tệp ảnh (\*.png, \*.jpg, \*jpeg)|\*.png;\*.jpg;\*.jpeg",

Title = "Chọn ảnh đại diện của bạn (tốt nhất là ảnh có chiều dài bằng chiều rộng)",

};

if (openDialog.ShowDialog() == DialogResult.OK)

{

newPathAvatar = openDialog.FileName;

ptrAvt.Image = Image.FromFile(newPathAvatar);

ptrAvt.SizeMode = PictureBoxSizeMode.StretchImage;

}

}

* Click nút Lưu:

private void selectSaveInfo(object sender, EventArgs e)

{

Account account = new Account(nameAct);

info.updateFullName(txtFullName.Text);

info.updateBirth(new Date(dtpBirth.Value));

info.updateAddress(txtAddress.Text);

if (!Person.isPhoneValid(txtPhone.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Số điện thoại không hợp lệ!!");

txtPhone.Text = info.getPhone();

txtPhone.Focus();

}

else

{

info.updatePhone(txtPhone.Text);

if (!Person.isEmailValid(txtEmail.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Email không hợp lệ!!");

txtEmail.Text = info.getEmail();

txtEmail.Focus();

}

else

{

info.updateEmail(txtEmail.Text);

}

}

info.updatePerson();

if (!string.IsNullOrEmpty(newPathAvatar))

{

account.updateAvatar(newPathAvatar);

}

}

* Click Mở form Quản lý tài khoản:

private void selectManageAcct(object sender, MouseEventArgs e)

{

if (e.Button != MouseButtons.Left)

{

return;

}

Hide();

frmManageAccount frmAcct = new frmManageAccount(info.getID());

frmAcct.ShowDialog();

Show();

}

* Click Mở form Quản lý Khoa / Ngành / Lớp:

private void selectManageKhoa(object sender, MouseEventArgs e)

{

if (e.Button != MouseButtons.Left)

{

return;

}

Hide();

frmManageKhoa frm = new frmManageKhoa();

frm.ShowDialog();

Show();

}

* Click Mở form Quản lý môn học:

private void selectManageSubject(object sender, MouseEventArgs e)

{

if (e.Button != MouseButtons.Left)

{

return;

}

Hide();

frmManageSubject frm = new frmManageSubject();

frm.ShowDialog();

Show();

}

* Click Mở form Phân công giảng dạy:

private void selectAssignmentTeacher(object sender, MouseEventArgs e)

{

if (e.Button != MouseButtons.Left)

{

return;

}

Hide();

frmAssignmentTeacher frm = new frmAssignmentTeacher();

frm.ShowDialog();

Show();

}

* Click Mở form Quản lý / Đăng ký phòng học:

private void selectManageClassroom(object sender, MouseEventArgs e)

{

if (e.Button != MouseButtons.Left)

{

return;

}

Hide();

frmManageClassroom frm = new frmManageClassroom();

frm.ShowDialog();

Show();

}

* Click Mở form Sắp lớp cho sinh viên:

private void selectAssignmentStudent(object sender, MouseEventArgs e)

{

if (e.Button != MouseButtons.Left)

{

return;

}

Hide();

frmAssignmentStudent frm = new frmAssignmentStudent();

frm.ShowDialog();

Show();

}

* Click Mở form Quản lý khảo sát:

private void selectManageSurvey(object sender, MouseEventArgs e)

{

if (e.Button != MouseButtons.Left)

{

return;

}

Hide();

frmManageSurvey frm = new frmManageSurvey();

frm.ShowDialog();

Show();

}

* Click Mở form Quản lý / Chấm điểm sinh viên:

private void selectManageStudent(object sender, MouseEventArgs e)

{

if (e.Button != MouseButtons.Left)

{

return;

}

Hide();

frmManageStudent frm = new frmManageStudent(nameAct);

frm.ShowDialog();

Show();

}

* Click Mở form Xem khảo sát:

private void selectWatchSurvey(object sender, MouseEventArgs e)

{

if (e.Button != MouseButtons.Left)

{

return;

}

Hide();

frmWatchSurvey frm;

if (typeAcct == Account.typeGV)

{

frm = new frmWatchSurvey(nameAct);

}

else

{

frm = new frmWatchSurvey();

}

frm.ShowDialog();

Show();

}

* Click Mở form Xem lịch dạy / Xem thời khoá biểu:

private void selectWatchCalendar(object sender, MouseEventArgs e)

{

if (e.Button != MouseButtons.Left)

{

return;

}

Hide();

frmWatchCalendar frm = new frmWatchCalendar(nameAct);

frm.ShowDialog();

Show();

}

* Click Mở form Xem môn học:

private void selectWatchSubject(object sender, MouseEventArgs e)

{

if (e.Button != MouseButtons.Left)

{

return;

}

Hide();

frmWatchSubject frm = new frmWatchSubject(nameAct);

frm.ShowDialog();

Show();

}

* Click Mở form Đăng ký môn:

private void selectEnrollSubject(object sender, MouseEventArgs e)

{

if (e.Button != MouseButtons.Left)

{

return;

}

Hide();

frmEnrollSubject frm = new frmEnrollSubject(nameAct);

frm.ShowDialog();

Show();

}

* Click Mở form Khảo sát giảng viên:

private void selectSurvey(object sender, MouseEventArgs e)

{

if (e.Button != MouseButtons.Left)

{

return;

}

Hide();

frmSurvey frm = new frmSurvey(nameAct);

frm.ShowDialog();

Show();

}

* Click nút Đăng xuất:

private void btnLogOut\_Click(object sender, EventArgs e)

{

if (MessageInfo.makeMessage("Question", "Hỏi lại cho chắc", "Bạn có chắc chắn muốn đăng xuất không?") != DialogResult.Yes)

{

return;

}

quickClose = true;

Hide();

frmSignIn frm = new frmSignIn();

frm.ShowDialog();

Close();

}

* Form Closing:

private void btnLogOut\_Click(object sender, EventArgs e)

{

if (MessageInfo.makeMessage("Question", "Hỏi lại cho chắc", "Bạn có chắc chắn muốn đăng xuất không?") != DialogResult.Yes)

{

return;

}

quickClose = true;

Hide();

frmSignIn frm = new frmSignIn();

frm.ShowDialog();

Close();

}

* Hiện thực các phương thức khác:
* Tạo 1 thanh thông tin:

Label createItem(string name, string text)

{

return new Label()

{

Name = name,

Text = text,

Size = new Size(375, 30),

Font = new Font("Times New Roman", 12, FontStyle.Bold),

ForeColor = Color.Navy,

};

}

* Hiển thị thông tin cá nhân:

void createInfo(string nameAct)

{

Image image = null;

object tmp = Account.getField("AnhDaiDien", "TenTK", nameAct);

if (tmp != null)

{

image = Avatar.ConvertBytesToImage((byte[])tmp);

}

ptrAvt.Image = image;

ptrAvt.SizeMode = PictureBoxSizeMode.StretchImage;

dtpBirth.Value = info.getBirth().getDate();

txtFullName.Text = info.getFullName().getFullName();

txtPhone.Text = info.getPhone();

txtEmail.Text = info.getEmail();

txtAddress.Text = info.getAddress();

fpnMoreInfo.Controls.Add(createItem("lblID", "Mã ID: " + info.getID()));

fpnMoreInfo.Controls.Add(createItem("lblNameAcct", "Tên tài khoản: " + account.getName()));

fpnMoreInfo.Controls.Add(createItem("lblNgayDK", "Ngày đăng ký tài khoản: " + account.getDate().getStrDate('-')));

if (info is Teacher)

{

Teacher teacher = (Teacher)info;

fpnMoreInfo.Controls.Add(createItem("lblTenKhoa", "Tên khoa: " + teacher.getTenKhoa()));

}

else if (info is Student)

{

Student student = (Student)info;

fpnMoreInfo.Controls.Add(createItem("lblNameMajor", "Tên ngành: " + student.getNameMajor()));

fpnMoreInfo.Controls.Add(createItem("lblHocKi", "Học kì: " + student.getHocKi()));

fpnMoreInfo.Controls.Add(createItem("lblYear", "Năm: " + student.getYear()));

fpnMoreInfo.Controls.Add(createItem("lblTongTC", "Số tín chỉ đã đạt: " + student.getTongSoTCDat() + " tín chỉ"));

fpnMoreInfo.Controls.Add(createItem("lblStatus", "Trạng thái học vấn: " + student.getStatus()));

}

}

* Hiển thị form cho Quản trị:

void frmAdmin()

{

BackColor = Color.CadetBlue;

Text = "Quản lý hoạt động giảng dạy - Quản trị - " + info.getID();

createInfo(nameAct);

int xStart = 50, yStart = 20;

int far = 20;

Button btnManageAcct = BoxButton.getButton("btnManageAcct", "Account", "Quản lý tài khoản", 220, 70, xStart, yStart);

pnOption.Controls.Add(btnManageAcct);

btnManageAcct.MouseClick += selectManageAcct;

xStart += btnManageAcct.Width + far;

Button btnManageKhoa = BoxButton.getButton("btnManageKhoa", "Khoa", "Quản lý khoa, ngành", 220, 70, xStart, yStart);

pnOption.Controls.Add(btnManageKhoa);

btnManageKhoa.MouseClick += selectManageKhoa;

xStart += btnManageKhoa.Width + far;

Button btnManageSubject = BoxButton.getButton("btnManageSubject", "Subject", "Quản lý môn học", 220, 70, xStart, yStart);

pnOption.Controls.Add(btnManageSubject);

btnManageSubject.MouseClick += selectManageSubject;

xStart = 50;

yStart += btnManageAcct.Height + far;

Button btnAssignmentTeacher = BoxButton.getButton("btnAssignmentTeacher", "Assignment Teacher", "Phân công giảng viên", 220, 70, xStart, yStart);

pnOption.Controls.Add(btnAssignmentTeacher);

btnAssignmentTeacher.MouseClick += selectAssignmentTeacher;

xStart += btnAssignmentTeacher.Width + far;

Button btnManageClassroom = BoxButton.getButton("btnManageClassroom", "Classroom", "Quản lý phòng học", 220, 70, xStart, yStart);

pnOption.Controls.Add(btnManageClassroom);

btnManageClassroom.MouseClick += selectManageClassroom;

xStart += btnManageClassroom.Width + far;

Button btnAssignmentStudent = BoxButton.getButton("btnAssignmentStudent", "Student", "Sắp xếp lớp SV", 220, 70, xStart, yStart);

pnOption.Controls.Add(btnAssignmentStudent);

btnAssignmentStudent.MouseClick += selectAssignmentStudent;

xStart = 50;

yStart += btnAssignmentTeacher.Height + far;

Button btnManageSurvey = BoxButton.getButton("btnManageSurvey", "Survey", "Quản lý khảo sát", 220, 70, xStart, yStart);

pnOption.Controls.Add(btnManageSurvey);

btnManageSurvey.MouseClick += selectManageSurvey;

}

* Hiển thị form dành cho Giảng viên:

void frmTeacher()

{

BackColor = Color.CadetBlue;

Text = "Quản lý giảng dạy - Giảng viên - " + info.getID();

createInfo(nameAct);

int xStart = 50, yStart = 20;

int far = 20;

Button btnWatchCalendar = BoxButton.getButton("btnWatchCalendar", "Calendar", "Xem lịch dạy", 220, 70, xStart, yStart);

pnOption.Controls.Add(btnWatchCalendar);

btnWatchCalendar.MouseClick += selectWatchCalendar;

xStart += btnWatchCalendar.Width + far;

Button btnMangeStudent = BoxButton.getButton("btnMangeStudent", "Student", "Quản lý, chấm điểm SV", 230, 70, xStart, yStart);

pnOption.Controls.Add(btnMangeStudent);

btnMangeStudent.MouseClick += selectManageStudent;

xStart += btnMangeStudent.Width + far;

Button btnManageKhoa = BoxButton.getButton("btnWatchSubject", "Subject", "Xem danh sách môn", 220, 70, xStart, yStart);

pnOption.Controls.Add(btnManageKhoa);

btnManageKhoa.MouseClick += selectWatchSubject;

xStart = 50;

yStart += btnManageKhoa.Height + far;

Button btnWatchSurvey = BoxButton.getButton("btnWatchSurvey", "Survey", "Xem các khảo sát", 220, 70, xStart, yStart);

pnOption.Controls.Add(btnWatchSurvey);

btnWatchSurvey.MouseClick += selectWatchSurvey;

}

* Hiển thị form dành cho Sinh viên:

void frmStudent()

{

BackColor = Color.CadetBlue;

Text = "Quản lý học tập - Sinh viên - " + info.getID();

createInfo(nameAct);

int xStart = 50, yStart = 20;

int far = 20;

Button btnEncrollSubject = BoxButton.getButton("btnEncrollSubject", "Enroll Subject", "Đăng ký môn học", 200, 70, xStart, yStart);

pnOption.Controls.Add(btnEncrollSubject);

btnEncrollSubject.MouseClick += selectEnrollSubject;

xStart += btnEncrollSubject.Width + far;

Button btnWatchCalendar = BoxButton.getButton("btnWatchCalendar", "Calendar", "Xem TKB", 200, 70, xStart, yStart);

pnOption.Controls.Add(btnWatchCalendar);

btnWatchCalendar.MouseClick += selectWatchCalendar;

xStart += btnWatchCalendar.Width + far;

Button btnWatchSubject = BoxButton.getButton("btnWatchSubject", "Subject", "Xem môn học", 200, 70, xStart, yStart);

pnOption.Controls.Add(btnWatchSubject);

btnWatchSubject.MouseClick += selectWatchSubject;

xStart = 50;

yStart += btnWatchSubject.Height + far;

Button btnWatchSurvey = BoxButton.getButton("btnSurvey", "Survey", "Khảo sát GV", 200, 70, xStart, yStart);

pnOption.Controls.Add(btnWatchSurvey);

btnWatchSurvey.MouseClick += selectSurvey;

}

### ***3.1.6. Màn hình (Form) Quản lý tài khoản Giảng viên và Sinh viên:***

* Mô tả màn hình: Form quản lý các tài khoản của Giảng viên và Sinh viên, danh sách tài khoản được hiển thị trên ListView có hỗ trợ check vào từng tài khoản để xoá tài khoản một lượt, có ComboBox lựa chọn kiểu sắp xếp và CheckBox để chọn sắp xếp tăng dần hay giảm dần. Cho phép lựa chọn tài khoản và chỉnh sửa thông tin cho nó hay chọn nút thêm tài khoản để tạo tài khoản mới (không thể tạo tài khoản Quản trị tại đây).
* Hiện thực các phương thức xử lý sự kiện (Event Handlers):
* TextChange tìm kiếm:

private void txtSearch\_TextChanged(object sender, EventArgs e)

{

reloadList();

}

* Click nút Xác nhận:

private void btnAccept\_Click(object sender, EventArgs e)

{

if (string.IsNullOrEmpty(txtNameAccount.Text) || string.IsNullOrEmpty(txtPassword.Text) || string.IsNullOrEmpty(txtName.Text) || string.IsNullOrEmpty(txtPhone.Text) || string.IsNullOrEmpty(txtEmail.Text) || string.IsNullOrEmpty(txtAddress.Text) || txtID.Text[txtID.Text.Length - 1] == '-')

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Bạn chưa nhập đủ thông tin!");

return;

}

if (string.IsNullOrEmpty(cbbTypeAcct.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Bạn chưa chọn kiểu tài khoản!");

return;

}

if (!Person.isPhoneValid(txtPhone.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Số điện thoại không hợp lệ!");

return;

}

if (!Person.isEmailValid(txtEmail.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Email không hợp lệ!");

return;

}

if (cbbTypeAcct.SelectedIndex == 0) //giảng viên

{

string tenKhoa = null;

foreach (Control control in gbInfo.Controls)

{

if (control.Name == "cbbTenKhoa")

{

tenKhoa = control.Text;

}

}

if (currentPerson == null)

{

Teacher newTeacher = new Teacher(txtID.Text, txtPhone.Text, txtName.Text, txtAddress.Text, txtEmail.Text, new Date(dtpBirth.Value), tenKhoa);

if (!Account.signUp(txtNameAccount.Text, txtPassword.Text, newTeacher))

{

return;

}

}

else

{

Account acct = new Account(txtNameAccount.Text, txtPassword.Text);

Teacher newTeacher = new Teacher(currentPerson.getID(), txtPhone.Text, txtName.Text, txtAddress.Text, txtEmail.Text, new Date(dtpBirth.Value), tenKhoa);

if (!acct.updateAccount(newTeacher, txtID.Text))

{

return;

}

}

}

else

{

int year = 0;

string nameMajor = null;

int hocKi = 1;

foreach (Control control in gbInfo.Controls)

{

if (control.Name == "cbbNameMajor")

{

if (string.IsNullOrEmpty(control.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Bạn chưa chọn tên ngành học!");

return;

}

nameMajor = control.Text;

}

else if (control.Name == "cbbYear")

{

if (string.IsNullOrEmpty(control.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Bạn chưa chọn năm của sinh viên!");

return;

}

year = int.Parse(control.Text);

}

else if (control.Name == "cbbHocKi")

{

if (string.IsNullOrEmpty(control.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Bạn chưa chọn học kì của sinh viên!");

return;

}

hocKi = int.Parse(control.Text);

}

}

if (currentPerson == null)

{

Student newStudent = new Student(txtID.Text, txtPhone.Text, txtName.Text, txtAddress.Text, txtEmail.Text, new Date(dtpBirth.Value), hocKi, year, nameMajor);

if (!Account.signUp(txtNameAccount.Text, txtPassword.Text, newStudent))

{

return;

}

}

else

{

Account acct = new Account(txtNameAccount.Text, txtPassword.Text);

Student newStudent = new Student(currentPerson.getID(), txtPhone.Text, txtName.Text, txtAddress.Text, txtEmail.Text, new Date(dtpBirth.Value), hocKi, year, nameMajor);

if (!acct.updateAccount(newStudent, txtID.Text))

{

return;

}

}

}

int index = -1;

if (lvInfo.SelectedIndices.Count != 0)

{

index = lvInfo.SelectedIndices[0];

}

reloadList();

if (index != -1)

{

lvInfo.SelectedIndices.Add(index);

}

}

* Click chọn tài khoản trên danh sách tài khoản:

private void lvInfo\_SelectedIndexChanged(object sender, EventArgs e)

{

if (lvInfo.SelectedIndices.Count == 0)

{

currentPerson = null;

pnInfo.Enabled = false;

defaultGBInfo();

return;

}

int index = lvInfo.SelectedIndices[0];

ListViewItem lv = lvInfo.Items[index];

string ID = lv.SubItems[0].Text;

currentPerson = people[index];

Account account = Account.getInfoByID(ID);

cbbTypeAcct.SelectedIndex = -1;

if (lv.SubItems[2].Text == "Sinh viên")

{

cbbTypeAcct.SelectedIndex = 1;

}

else

{

cbbTypeAcct.SelectedIndex = 0;

}

txtName.Text = currentPerson.getFullName().getFullName();

txtPhone.Text = currentPerson.getPhone();

txtEmail.Text = currentPerson.getEmail();

txtAddress.Text = currentPerson.getAddress();

txtID.Text = currentPerson.getID();

txtNameAccount.Text = account.getName();

txtPassword.Text = account.getPassword();

dtpBirth.Value = currentPerson.getBirth().getDate();

cbbTypeAcct.Enabled = false;

pnInfo.Enabled = true;

}

* Lựa chọn kiểu tài khoản Giảng viên hay Sinh viên:

private void cbbTypeAcct\_SelectedIndexChanged(object sender, EventArgs e)

{

clearTextInfo();

if (cbbTypeAcct.SelectedIndex == -1)

{

return;

}

if (cbbTypeAcct.SelectedIndex == 0)

{

showMoreInfoTeacher();

}

else

{

showMoreInfoStudent();

}

}

* Lựa chọn ngành học:

private void cbbNameMajor\_SelectedIndexChanged(object sender, EventArgs e)

{

ComboBox cbbNameMajor = (ComboBox)sender;

if (cbbNameMajor.SelectedIndex == -1)

{

return;

}

string nameMajor = cbbNameMajor.Text;

int years = (int)Major.getField("SoNam", "Ten", nameMajor);

List<int> datas = new List<int>();

for (int i = 1; i <= years; i++)

{

datas.Add(i);

}

foreach (Control control in gbInfo.Controls)

{

if (control.Name == "cbbYear")

{

ComboBox cbb = (ComboBox)control;

cbb.DataSource = datas;

return;

}

}

}

* Lựa chọn lọc danh sách:

private void rdoFilter\_CheckedChanged(object sender, EventArgs e)

{

reloadList();

}

* Click nút thêm tài khoản:

private void btnAdd\_Click(object sender, EventArgs e)

{

lvInfo.SelectedIndices.Clear();

defaultGBInfo();

gbInfo.Height = 397;

cbbTypeAcct.SelectedIndex = -1;

cbbTypeAcct.Enabled = true;

currentPerson = null;

pnInfo.Enabled = true;

}

* KeyPress cho nhập ID:

private void txtID\_KeyPress(object sender, KeyPressEventArgs e)

{

if (e.KeyChar == (char)Keys.Back)

{

if (!string.IsNullOrEmpty(txtID.Text) && txtID.Text[txtID.Text.Length - 1] == '-')

{

e.Handled = true;

}

}

else if (char.IsLetter(e.KeyChar) || char.IsSymbol(e.KeyChar) || (!char.IsLetterOrDigit(e.KeyChar) && e.KeyChar > 30))

{

e.Handled = true;

}

}

* Click nút Xoá tài khoản:

private void btnDelete\_Click(object sender, EventArgs e)

{

if (MessageInfo.makeMessage("Question", "Hỏi lại cho chắc", "Bạn có chắn chắn muốn xoá các tài khoản đã check?") != DialogResult.Yes)

{

return;

}

for (int i = 0; i < lvInfo.CheckedIndices.Count; i++)

{

string ID = lvInfo.Items[lvInfo.CheckedIndices[i]].SubItems[0].Text;

if (Student.isExistStudentID(ID))

{

Student.deleteStudent(ID);

}

else

{

Teacher.deleteTeacher(ID);

}

}

reloadList();

btnDelete.Enabled = false;

currentPerson = null;

}

* Lựa chọn sắp xếp tăng dần hay giảm dần:

private void ckbIncrease\_CheckedChanged(object sender, EventArgs e)

{

reloadList();

}

* Lựa chọn kiểu sắp xếp danh sách:

private void cbbSortBy\_SelectedIndexChanged(object sender, EventArgs e)

{

reloadList();

}

* Click Bật / Tắt hiển thị mật khẩu:

private void ckbShowPassword\_CheckedChanged(object sender, EventArgs e)

{

txtPassword.UseSystemPasswordChar = !ckbShowPassword.Checked;

}

* Check và Uncheck vào tài khoản trên danh sách:

private void lvInfo\_ItemChecked(object sender, ItemCheckedEventArgs e)

{

if (lvInfo.CheckedIndices.Count != 0)

{

btnDelete.Enabled = true;

}

else

{

btnDelete.Enabled = false;

}

}

* Hiện thực các phương thức khác:
* Khởi tạo form Quản lý tài khoản:

public frmManageAccount(string ID)

{

InitializeComponent();

this.ID = ID;

Text = "Quản lý đại học - Quản lý tài khoản giảng viên và sinh viên - " + ID;

cbbTypeSearch.SelectedIndex = 0;

cbbSortBy.SelectedIndex = 0;

currentPerson = null;

reloadList();

}

* Cập nhật số lượng tài khoản:

void reloadCountAcct()

{

lblCountAccount.Text = "Số lượng: " + lvInfo.Items.Count.ToString();

}

* Load danh sách tài khoản:

void loadList()

{

lvInfo.Items.Clear();

for (int i = 0; i < people.Count; i++)

{

string type;

if (Student.isExistStudentID(people[i].getID()))

{

type = "Sinh viên";

}

else

{

type = "Giảng viên";

}

ListViewItem lvi = new ListViewItem()

{

Text = people[i].getID(),

};

lvi.SubItems.Add(new ListViewItem.ListViewSubItem() { Text = people[i].getFullName().getFullName() });

lvi.SubItems.Add(new ListViewItem.ListViewSubItem() { Text = type });

lvInfo.Items.Add(lvi);

}

}

* Loại bỏ tài khoản khỏi danh sách bằng kí tự:

void removeListByCharName(char c)

{

c = char.ToLower(c);

for (int i = 0; i < people.Count; i++)

{

bool remove = !people[i].getFullName().getFullName().ToLower().Contains(c);

if (remove)

{

people.RemoveAt(i);

i--;

}

}

}

* Loại bỏ tài khoản khỏi danh sách theo ID:

void removeListByCharID(char c)

{

c = char.ToLower(c);

for (int i = 0; i < people.Count; i++)

{

bool remove = !people[i].getID().ToLower().Contains(c);

if (remove)

{

people.RemoveAt(i);

i--;

}

}

}

* Tìm kiếm:

void Search()

{

if (cbbTypeSearch.Text == cbbTypeSearch.Items[0].ToString()) //sắp xếp theo tên

{

if (!string.IsNullOrEmpty(txtSearch.Text))

{

for (int i = 0; i < txtSearch.Text.Length; i++)

{

removeListByCharName(txtSearch.Text[i]);

}

}

}

else

{

if (!string.IsNullOrEmpty(txtSearch.Text))

{

for (int i = 0; i < txtSearch.Text.Length; i++)

{

removeListByCharID(txtSearch.Text[i]);

}

}

}

}

* Load lại danh sách tài khoản:

void reloadList()

{

findAndSort();

findAndFilter();

Search();

loadList();

reloadCountAcct();

}

* Sắp xếp danh sách:

void findAndSort()

{

if (cbbSortBy.SelectedIndex == 0)

{

ckbIncrease.Enabled = false;

people = Person.getPeopleExceptAdmin();

return;

}

ckbIncrease.Enabled = true;

if (cbbSortBy.SelectedIndex == 1)

{

people = Person.getPersonSorted("Ten", ckbIncrease.Checked);

}

else

{

people = Person.getPersonSorted("ID", ckbIncrease.Checked);

}

}

* Tạo ComboBox:

ComboBox getCombobox(string name, int X, int Y)

{

ComboBox cbb = new ComboBox()

{

Font= new Font("Times New Roman", 10, FontStyle.Bold),

ForeColor = Color.RoyalBlue,

Name = name,

Width = 301,

DropDownStyle = ComboBoxStyle.DropDownList,

AutoCompleteSource = AutoCompleteSource.ListItems,

AutoCompleteMode = AutoCompleteMode.SuggestAppend,

Location = new Point(X, Y),

Cursor = Cursors.Hand,

};

return cbb;

}

* Hiển thị thông tin mở rộng cho tài khoản Giảng viên:

void showMoreInfoTeacher()

{

gbInfo.Height = 440;

Label lblTenKhoa = getLabel("lblTenKhoa", "Tên khoa", lblAddress.Left, lblAddress.Top + 50);

gbInfo.Controls.Add(lblTenKhoa);

ComboBox cbbTenKhoa = getCombobox("cbbTenKhoa", txtAddress.Left, lblTenKhoa.Top - 3);

List<string> nameKhoa = new List<string>();

List<Khoa> khoas = Khoa.getAllKhoa();

for (int i = 0; i < khoas.Count; i++)

{

nameKhoa.Add(khoas[i].getName());

}

cbbTenKhoa.DataSource = nameKhoa;

gbInfo.Controls.Add(cbbTenKhoa);

if (string.IsNullOrEmpty(txtID.Text))

{

txtID.Text = "GV-";

}

else

{

txtID.Text = "GV-" + txtID.Text.Substring(3);

}

if (currentPerson != null)

{

Teacher teacher = (Teacher)Teacher.getInfo("ID", currentPerson.getID());

for (int i =0; i < cbbTenKhoa.Items.Count; i++)

{

if (teacher.getTenKhoa() == cbbTenKhoa.Items[i].ToString())

{

cbbTenKhoa.SelectedIndex = i;

break;

}

}

}

}

* Hiển thị thông tin mở rộng dành cho Sinh viên:

void showMoreInfoStudent()

{

gbInfo.Height = 550;

Label lblNameMajor = getLabel("lblNameMajor", "Tên ngành", lblAddress.Left, lblAddress.Top + 50);

gbInfo.Controls.Add(lblNameMajor);

ComboBox cbbNameMajor = getCombobox("cbbNameMajor", txtAddress.Left, lblNameMajor.Top - 3);

List<string> nameMajor = new List<string>();

List<Major> majors = Major.getAllMajor();

cbbNameMajor.Items.Clear();

for (int i = 0; i < majors.Count; i++)

{

cbbNameMajor.Items.Add(majors[i].getName());

}

cbbNameMajor.SelectedIndexChanged += cbbNameMajor\_SelectedIndexChanged;

gbInfo.Controls.Add(cbbNameMajor);

Label lblHocKi = getLabel("lblHocKi", "Học kì", lblNameMajor.Left, lblNameMajor.Top + 50);

gbInfo.Controls.Add(lblHocKi);

ComboBox cbbHocKi = getCombobox("cbbHocKi", txtAddress.Left, lblHocKi.Top - 3);

for (int i = 1; i <= 2; i++)

{

cbbHocKi.Items.Add(i);

}

gbInfo.Controls.Add(cbbHocKi);

Label lblYear = getLabel("lbllYear", "Năm", lblHocKi.Left, lblHocKi.Top + 50);

gbInfo.Controls.Add(lblYear);

ComboBox cbbYear = getCombobox("cbbYear", txtAddress.Left, lblYear.Top - 3);

cbbHocKi.DropDownStyle = ComboBoxStyle.DropDownList;

gbInfo.Controls.Add(cbbYear);

if (string.IsNullOrEmpty(txtID.Text))

{

txtID.Text = "SV-";

}

else

{

txtID.Text = "SV-" + txtID.Text.Substring(3);

}

if (currentPerson != null)

{

Student student = (Student)Student.getInfo("ID", currentPerson.getID());

for (int i = 0; i < cbbNameMajor.Items.Count; i++)

{

if (student.getNameMajor() == cbbNameMajor.Items[i].ToString())

{

cbbNameMajor.SelectedIndex = i;

break;

}

}

for (int i = 0; i < cbbHocKi.Items.Count; i++)

{

if (student.getHocKi().ToString() == cbbHocKi.Items[i].ToString())

{

cbbHocKi.SelectedIndex = i;

break;

}

}

for (int i = 0; i < cbbYear.Items.Count; i++)

{

if (student.getYear().ToString() == cbbYear.Items[i].ToString())

{

cbbYear.SelectedIndex = i;

break;

}

}

}

else

{

cbbNameMajor.SelectedIndex = -1;

}

}

* Lọc danh sách tài khoản:

void findAndFilter()

{

if (rdoTeacher.Checked)

{

for (int i = 0; i < people.Count; i++)

{

if (Student.isExistStudentID(people[i].getID()))

{

people.RemoveAt(i);

i--;

}

}

}

else if (rdoStudent.Checked)

{

for (int i = 0; i < people.Count; i++)

{

if (Teacher.isExistTeacherID(people[i].getID()))

{

people.RemoveAt(i);

i--;

}

}

}

}

* Tắt thông tin mở rộng:

void defaultGBInfo()

{

for (int i = 0; i < gbInfo.Controls.Count; i++)

{

if (gbInfo.Controls[i].Name == "lblTenKhoa" || gbInfo.Controls[i].Name == "cbbTenKhoa" || gbInfo.Controls[i].Name == "lblNameMajor" || gbInfo.Controls[i].Name == "cbbNameMajor" || gbInfo.Controls[i].Name == "lblYear" || gbInfo.Controls[i].Name == "cbbYear" || gbInfo.Controls[i].Name == "lblHocKi" || gbInfo.Controls[i].Name == "cbbHocKi")

{

gbInfo.Controls.RemoveAt(i);

i--;

}

}

clearTextAccount();

clearTextInfo();

}

* Dọn dẹp TextBox của tài khoản:

void clearTextAccount()

{

foreach (Control control in gbAccount.Controls)

{

if (control is TextBox)

{

control.Text = "";

}

}

}

* Dọn dẹp TextBox của thông tin:

void clearTextInfo()

{

foreach (Control control in gbInfo.Controls)

{

if (control is TextBox)

{

control.Text = "";

}

}

}

### ***3.1.7. Màn hình (Form) Khoa, Ngành, Lớp:***

* Mô tả màn hình: Form quản lý các khoa, ngành và thêm mới lớp học. Thêm ngành cần xác định thêm vào khoa nào, thêm lớp cần xác định thêm vào ngành nào. Danh sách khoa, ngành, lớp hiển thị trên TreeView cho phép chọn vào 1 node bất kì và xoá hay chỉnh sửa thông tin.
* Hiện thực các phương thức xử lý sự kiện (Event Handlers):
* Lựa chọn Node trên danh sách:

private void tvKhoa\_AfterSelect(object sender, TreeViewEventArgs e)

{

currentNode = e.Node;

string tag = (string)currentNode.Tag;

clearText();

gbInfo.Enabled = true;

btnAccept.Enabled = true;

if (tag == "0")

{

btnDelete.Enabled = false;

pnClass.Enabled = false;

return;

}

btnDelete.Enabled = true;

pnClass.Enabled = true;

if (tag == "1")

{

txtNameKhoa.Text = currentNode.Text;

}

else if (tag == "2")

{

txtNameKhoa.Text = currentNode.Parent.Text;

txtNameMajor.Text = currentNode.Text;

Major major = Major.getInfo(txtNameMajor.Text);

txtCountYear.Text = major.getCountYear().ToString();

}

else

{

txtNameKhoa.Text = currentNode.Parent.Parent.Text;

txtNameMajor.Text = currentNode.Parent.Text;

txtNameClass.Text = currentNode.Text;

Khoa khoa = new Khoa(txtNameKhoa.Text);

Major major = Major.getInfo(txtNameMajor.Text);

Class classs = Class.getInfo(txtNameClass.Text);

txtCountYear.Text = major.getCountYear().ToString();

txtCountMax.Text = classs.getMaxCount().ToString();

List<Class> lstClass = Class.getAllClass();

List<Teacher> teachers = khoa.getTeachers();

cbbMaGV.Items.Add(classs.getMaGV());

for (int i = 0; i < teachers.Count; i++)

{

bool isOk = true;

for (int j = 0; j < lstClass.Count; j++)

{

if (teachers[i].getID() == lstClass[j].getMaGV())

{

isOk = false;

break;

}

}

if (isOk)

{

cbbMaGV.Items.Add(teachers[i].getID());

break;

}

}

cbbMaGV.SelectedIndex = 0;

}

}

* Click nút xoá Node:

private void btnDelete\_Click(object sender, EventArgs e)

{

if (MessageInfo.makeMessage("Question", "Hỏi lại cho chắc", "Bạn có chắn chắn muốn xoá không?") != DialogResult.Yes)

{

return;

}

string tag = (string)currentNode.Tag;

if (tag == "1")

{

Khoa.deleteKhoa(currentNode.Text);

}

else if (tag == "2")

{

Major.deleteNganh(currentNode.Text);

}

else if (tag == "3")

{

Class.deleteClass(currentNode.Text);

}

tvKhoa.Nodes.Remove(currentNode);

NoSelected();

}

* TextChanged tên khoa:

private void txtNameKhoa\_TextChanged(object sender, EventArgs e)

{

cbbMaGV.Items.Clear();

if (string.IsNullOrEmpty(txtNameKhoa.Text))

{

return;

}

string tag = (string)currentNode.Tag;

if (tag == "0" || !Khoa.isExistKhoa(txtNameKhoa.Text))

{

return;

}

Khoa khoa = new Khoa(txtNameKhoa.Text);

List<Teacher> teachers = khoa.getTeachers();

List<Class> classs = Class.getAllClass();

if (tag == "3")

{

return;

}

for (int j = 0; j < teachers.Count; j++)

{

bool isOK = true;

for (int k = 0; k < classs.Count; k++)

{

if (classs[k].getMaGV() == teachers[j].getID())

{

isOK = false;

break;

}

}

if (isOK)

{

for (int k = 0; k < cbbMaGV.Items.Count; k++)

{

if (cbbMaGV.Items[k].ToString() == teachers[j].getID())

{

isOK = false;

break;

}

}

if (isOK)

{

cbbMaGV.Items.Add(teachers[j].getID());

}

}

}

}

* Click nút Xác nhận:

private void btnAccept\_Click(object sender, EventArgs e)

{

if (currentNode == null)

{

return;

}

string tag = (string)currentNode.Tag;

if (tag == "0")

{

if (string.IsNullOrEmpty(txtNameKhoa.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Bạn chưa nhập tên khoa!");

txtNameKhoa.Focus();

return;

}

if (Khoa.isExistKhoa(txtNameKhoa.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Tên khoa đã tồn tại!");

txtNameKhoa.Focus();

return;

}

if (!Khoa.addKhoa(txtNameKhoa.Text))

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Có lỗi xảy ra!");

}

else

{

if (!string.IsNullOrEmpty(txtNameMajor.Text) && !string.IsNullOrEmpty(txtCountYear.Text))

{

if (Major.isExistsMajor(txtNameMajor.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Tên ngành đã tồn tại!");

txtNameMajor.Focus();

return;

}

if (!Major.addMajor(txtNameMajor.Text, txtNameKhoa.Text, int.Parse(txtCountYear.Text)))

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Có lỗi xảy ra!");

}

else

{

if (!string.IsNullOrEmpty(txtNameClass.Text) && cbbMaGV.SelectedIndex != -1 && !string.IsNullOrEmpty(txtCountMax.Text))

{

if (Class.isExistsClass(txtNameClass.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Tên lớp đã tồn tại!");

txtNameClass.Focus();

return;

}

if (!Class.addClass(txtNameClass.Text, cbbMaGV.Text, int.Parse(txtCountMax.Text), txtNameMajor.Text))

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Có lỗi xảy ra!");

}

}

}

}

}

}

else if (tag == "1")

{

if (string.IsNullOrEmpty(txtNameKhoa.Text)) //sửa khoa

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Bạn chưa nhập tên khoa!");

txtNameKhoa.Text = currentNode.Text;

txtNameKhoa.Focus();

return;

}

if (txtNameKhoa.Text != currentNode.Text && Khoa.isExistKhoa(txtNameKhoa.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Tên khoa đã tồn tại!");

txtNameKhoa.Focus();

return;

}

Khoa khoa = new Khoa(currentNode.Text);

if (!khoa.updateKhoa(txtNameKhoa.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Cập nhật khoa thất bại!");

txtNameKhoa.Focus();

return;

}

if (!string.IsNullOrEmpty(txtNameMajor.Text) && !(string.IsNullOrEmpty(txtCountYear.Text)))

{

if (Major.isExistsMajor(txtNameMajor.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Tên ngành đã tồn tại!");

txtNameMajor.Focus();

return;

}

if (!Major.addMajor(txtNameMajor.Text, txtNameKhoa.Text, int.Parse(txtCountYear.Text)))

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Có lỗi xảy ra!");

}

else

{

if (!string.IsNullOrEmpty(txtNameClass.Text) && cbbMaGV.SelectedIndex != -1 && !string.IsNullOrEmpty(txtCountMax.Text))

{

if (Class.isExistsClass(txtNameClass.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Tên lớp đã tồn tại!");

txtNameClass.Focus();

return;

}

if (!Class.addClass(txtNameClass.Text, cbbMaGV.Text, int.Parse(txtCountMax.Text), txtNameMajor.Text))

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Có lỗi xảy ra!");

}

}

else

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Không thể thêm lớp vì không đủ thông tin!");

}

}

}

else

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Không thể thêm ngành, lớp vì không đủ thông tin!");

}

}

else if (tag == "2")

{

if (string.IsNullOrEmpty(txtNameKhoa.Text)) //sửa khoa

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Bạn chưa nhập tên khoa để sửa!");

txtNameKhoa.Text = currentNode.Parent.Text;

txtNameKhoa.Focus();

return;

}

if (txtNameKhoa.Text != currentNode.Parent.Text && Khoa.isExistKhoa(txtNameKhoa.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Tên khoa đã tồn tại!");

txtNameKhoa.Focus();

return;

}

Khoa khoa = new Khoa(currentNode.Parent.Text);

if (!khoa.updateKhoa(txtNameKhoa.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Cập nhật khoa thất bại!");

txtNameKhoa.Focus();

return;

}

if (!string.IsNullOrEmpty(txtNameMajor.Text) && !string.IsNullOrEmpty(txtCountYear.Text))

{

Major major = Major.getInfo(currentNode.Text);

if (txtNameMajor.Text != currentNode.Text && Major.isExistsMajor(txtNameMajor.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Tên ngành đã tồn tại!");

txtNameMajor.Focus();

return;

}

if (!major.updateMajor(txtNameMajor.Text, int.Parse(txtCountYear.Text), txtNameKhoa.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Cập nhật ngành thất bại!");

txtNameMajor.Focus();

return;

}

if (!string.IsNullOrEmpty(txtNameClass.Text) && cbbMaGV.SelectedIndex != -1 && !string.IsNullOrEmpty(txtCountMax.Text))

{

if (Class.isExistsClass(txtNameClass.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Tên lớp đã tồn tại!");

txtNameClass.Focus();

return;

}

if (!Class.addClass(txtNameClass.Text, cbbMaGV.Text, int.Parse(txtCountMax.Text), txtNameMajor.Text))

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Có lỗi xảy ra!");

}

}

else

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Không thể thêm lớp vì không đủ thông tin!");

}

}

}

else

{

if (string.IsNullOrEmpty(txtNameKhoa.Text)) //sửa khoa

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Bạn chưa nhập tên khoa để sửa!");

txtNameKhoa.Text = currentNode.Parent.Parent.Text;

txtNameKhoa.Focus();

return;

}

if (txtNameKhoa.Text != currentNode.Parent.Parent.Text && Khoa.isExistKhoa(txtNameKhoa.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Tên khoa đã tồn tại!");

txtNameKhoa.Focus();

return;

}

Khoa khoa = new Khoa(currentNode.Parent.Parent.Text);

if (!khoa.updateKhoa(txtNameKhoa.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Cập nhật khoa thất bại!");

txtNameKhoa.Focus();

return;

}

if (!string.IsNullOrEmpty(txtNameMajor.Text) && !string.IsNullOrEmpty(txtCountYear.Text))

{

Major major = Major.getInfo(currentNode.Parent.Text);

if (txtNameMajor.Text != currentNode.Parent.Text && Major.isExistsMajor(txtNameMajor.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Tên ngành đã tồn tại!");

txtNameMajor.Focus();

return;

}

if (!major.updateMajor(txtNameMajor.Text, int.Parse(txtCountYear.Text), txtNameKhoa.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Cập nhật ngành thất bại!");

txtNameMajor.Focus();

return;

}

if (!string.IsNullOrEmpty(txtNameClass.Text) && cbbMaGV.SelectedIndex != -1 && !string.IsNullOrEmpty(txtCountMax.Text))

{

if (txtNameClass.Text != currentNode.Text && Class.isExistsClass(txtNameClass.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Tên lớp đã tồn tại!");

txtNameClass.Focus();

return;

}

Class classs = Class.getInfo(currentNode.Text);

if (!classs.updateClass(txtNameClass.Text, cbbMaGV.Text, int.Parse(txtCountMax.Text), txtNameMajor.Text))

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Cập nhật lớp thất bại!");

}

}

}

}

uploadInfo();

}

* Hiện thực các phương thức khác:
* Khởi tạo form Khoa, Ngành, Lớp:

public frmManageKhoa()

{

InitializeComponent();

ImageList img = new ImageList();

img.Images.Add(Image.FromFile("Assets/Imgs/Select.png"));

img.Images.Add(Image.FromFile("Assets/Imgs/Khoa.png"));

img.Images.Add(Image.FromFile("Assets/Imgs/Major.png"));

img.Images.Add(Image.FromFile("Assets/Imgs/Class.png"));

txtCountYear.KeyPress += General.number\_KeyPress;

txtCountMax.KeyPress += General.number\_KeyPress;

tvKhoa.ImageList = img;

uploadInfo();

if (currentNode == null)

{

NoSelected();

}

}

* Không chọn Node nào:

void NoSelected()

{

gbInfo.Enabled = false;

btnAccept.Enabled = false;

btnDelete.Enabled = false;

clearText();

tvKhoa.SelectedNode = null;

currentNode = null;

}

* Load danh sách khoa, ngành, lớp:

void uploadInfo(string currenTextNode = null)

{

tvKhoa.Nodes.Clear();

TreeNode root = new TreeNode()

{

Text = "Khoa / Viện",

Tag = "0",

ImageIndex = 0,

};

DataTable dataKhoa = SQL.Excute\_Values("Select \* from Khoa", null, null);

if (dataKhoa == null)

{

return;

}

for (int i = 0; i < dataKhoa.Rows.Count;i++)

{

TreeNode nodeKhoa = new TreeNode()

{

Text = (string)dataKhoa.Rows[i][0],

Tag = "1",

ImageIndex = 1,

};

DataTable dataMajor = SQL.Excute\_Values("Select \* from Nganh where TenKhoa = @TenKhoa", new List<string>() { "TenKhoa" }, new List<object>() { nodeKhoa.Text });

if (dataMajor != null)

{

for (int j = 0; j < dataMajor.Rows.Count; j++)

{

TreeNode nodeMajor = new TreeNode()

{

Text = (string)dataMajor.Rows[j][0],

Tag = "2",

ImageIndex = 2,

}; ;

DataTable dataSubject = SQL.Excute\_Values("Select \* from Lop where TenNganh = @TenNganh", new List<string>() { "TenNganh" }, new List<object>() { nodeMajor.Text });

if (dataSubject != null)

{

for (int k = 0; k < dataSubject.Rows.Count; k ++)

{

TreeNode nodeSubject = new TreeNode()

{

Text = (string)dataSubject.Rows[k][0],

Tag = "3",

ImageIndex = 3,

};

if (nodeSubject.Text == currenTextNode)

{

currentNode = nodeSubject;

}

nodeMajor.Nodes.Add(nodeSubject);

}

}

if (nodeMajor.Text == currenTextNode)

{

currentNode = nodeMajor;

}

nodeKhoa.Nodes.Add(nodeMajor);

}

}

if (nodeKhoa.Text == currenTextNode)

{

currentNode = nodeKhoa;

}

root.Nodes.Add(nodeKhoa);

}

tvKhoa.Nodes.Add(root);

tvKhoa.ExpandAll();

NoSelected();

}

* Dọn dẹp TextBox:

void clearText()

{

txtNameKhoa.Clear();

txtNameMajor.Clear();

txtCountYear.Clear();

txtNameClass.Clear();

txtCountMax.Clear();

cbbMaGV.Items.Clear();

cbbMaGV.Text = "";

}

### ***3.1.8. Màn hình (Form) Quản lý môn học:***

* Mô tả màn hình: Form quản lý các môn học trong từng khoa, ngành. Danh sách khoa, ngành để xem các môn nằm trong đó, có thể sắp xếp môn học theo nhiều cách hay tăng dần, giảm dần bằng RadioButton. Lọc danh sách môn học bằng RadioButton. Môn học hiển thị trên ListView cho phép check vào để xoá đồng thời nhiều môn. Hỗ trợ chọn vào môn học để chỉnh sửa hay chọn nút thêm môn để thêm môn mới vào khoa, ngành hiện tại.
* Hiện thực các phương thức xử lý sự kiện (Event Handlers):
* Lựa chọn Node trên danh sách Khoa, ngành:

private void tvKhoa\_AfterSelect(object sender, TreeViewEventArgs e)

{

string tag = (string)e.Node.Tag;

if (tag == "2")

{

currentKhoa = new Khoa(e.Node.Parent.Text);

currentMajor = Major.getInfo(e.Node.Text);

loadSubject();

pnSubject.Enabled = true;

}

else

{

currentMajor = null;

if (tag == "1")

{

currentKhoa = new Khoa(e.Node.Parent.Text);

pnSubject.Enabled = true;

loadSubject();

}

else

{

currentKhoa = null;

lvSubject.Items.Clear();

pnSubject.Enabled = false;

}

}

}

* Click nút Thêm môn:

private void btnAdd\_Click(object sender, EventArgs e)

{

lvSubject.SelectedIndices.Clear();

gbInfo.Text = "Thêm môn học";

resetDefault();

gbInfo.Enabled = true;

}

* KeyPress dành cho nhập phí môn học:

private void txtMoney\_KeyPress(object sender, KeyPressEventArgs e)

{

if (e.KeyChar != ',' && (char.IsLetter(e.KeyChar) || char.IsSymbol(e.KeyChar) || (!char.IsLetterOrDigit(e.KeyChar) && e.KeyChar > 30)))

{

e.Handled = true;

}

}

* TextChanged thanh tìm kiếm:

private void txtSearch\_TextChanged(object sender, EventArgs e)

{

findAndFilter();

search();

findAndSort();

uploadList();

}

* Lựa chọn kiểu sắp xếp:

private void radio\_CheckedTypeSort(object sender, EventArgs e)

{

if (rdoInscrease.Checked)

{

currentInscrease = true;

}

else

{

currentInscrease = false;

}

findAndFilter();

search();

findAndSort();

uploadList();

}

* Click Xoá môn học:

private void btnDelete\_Click(object sender, EventArgs e)

{

if (MessageInfo.makeMessage("Question", "Hỏi lại cho chắc", "Bạn có chắc muốn xoá các môn đã check không?") != DialogResult.Yes)

{

return;

}

for (int i = 0; i < lvSubject.CheckedIndices.Count; i++)

{

subjects[lvSubject.CheckedIndices[i]].deleteSubject();

}

loadSubject();

resetDefault();

}

* Lựa chọn môn học trên danh sách môn:

private void lvSubject\_SelectedIndexChanged(object sender, EventArgs e)

{

if (lvSubject.SelectedIndices.Count == 0)

{

gbInfo.Enabled = false;

return;

}

gbInfo.Enabled = true;

int index = lvSubject.SelectedIndices[0];

currentSubject = subjects[index];

txtName.Text = subjects[index].getName();

txtSoTC.Text = subjects[index].getSoTC().ToString();

txtSoTiet.Text = subjects[index].getSoTiet().ToString();

txtMoney.Text = subjects[index].getMoney().ToString();

ckMust.Checked = subjects[index].getMust();

cbbYear.Text = subjects[index].getYear().ToString();

cbbHocKi.Text = subjects[index].getHocKi().ToString();

}

* Check / Uncheck môn học trên danh sách:

private void lvSubject\_ItemChecked(object sender, ItemCheckedEventArgs e)

{

if (lvSubject.CheckedIndices.Count != 0)

{

btnDelete.Enabled = true;

}

else

{

btnDelete.Enabled = false;

}

}

* Click nút Xác nhận:

private void btnAccept\_Click(object sender, EventArgs e)

{

if (string.IsNullOrEmpty(txtName.Text) || string.IsNullOrEmpty(txtSoTC.Text) || string.IsNullOrEmpty(txtMoney.Text) || cbbYear.SelectedIndex == -1)

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Bạn chưa nhập đủ thông tin!");

return;

}

if (cbbHocKi.SelectedIndex == -1)

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Bạn chưa chọn học kì cho môn học!");

return;

}

Subject subject = new Subject(txtName.Text, ckMust.Checked, int.Parse(txtSoTC.Text), int.Parse(txtSoTiet.Text), double.Parse(txtMoney.Text), int.Parse(cbbHocKi.SelectedItem.ToString()), int.Parse(cbbYear.Text));

List<Major> majors;

if (currentMajor == null)

{

majors = currentKhoa.getMyMajor();

}

else

{

majors = new List<Major>() { currentMajor };

}

if (currentSubject == null)

{

if (subject.addSubject(majors))

{

MessageInfo.makeMessage("Information", "Chúc mừng", "Thêm môn thành công!");

loadSubject();

resetDefault();

gbInfo.Enabled = true;

txtName.Focus();

}

else

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Thêm môn thất bại!");

}

}

else

{

if (currentSubject.updateSubject(subject, majors))

{

loadSubject();

resetDefault();

}

else

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Cập nhật thất bại");

}

}

}

* Lựa chọn lọc môn học:

private void rdo\_CheckedChangeFilter(object sender, EventArgs e)

{

findAndFilter();

search();

findAndSort();

uploadList();

}

* Lựa chọn kiểu sắp xếp:

private void rdoSortBy\_CheckedChanged(object sender, EventArgs e)

{

findAndFilter();

search();

findAndSort();

uploadList();

}

* Hiện thực các phương thức khác:
* Khởi tạo form Quản lý môn:

public frmManageSubject()

{

InitializeComponent();

ImageList img = new ImageList();

img.Images.Add(Image.FromFile("Assets/Imgs/Select.png"));

img.Images.Add(Image.FromFile("Assets/Imgs/Khoa.png"));

img.Images.Add(Image.FromFile("Assets/Imgs/Major.png"));

img.Images.Add(Image.FromFile("Assets/Imgs/Class.png"));

tvKhoa.ImageList = img;

txtSoTC.KeyPress += General.number\_KeyPress;

txtSoTiet.KeyPress += General.number\_KeyPress;

loadKhoa();

cbbHocKi.SelectedIndex = 0;

cbbYear.SelectedIndex = 0;

}

* Load danh sách khoa, ngành:

void loadKhoaNganh()

{

tvKhoa.Nodes.Clear();

TreeNode root = new TreeNode()

{

Text = "Khoa / Viện",

Tag = "0",

ImageIndex = 0,

};

List<Khoa> khoas = Khoa.getAllKhoa();

for (int i = 0; i < khoas.Count; i++)

{

TreeNode nodeKhoa = new TreeNode()

{

Text = khoas[i].getName(),

Tag = "1",

ImageIndex = 1,

};

List<Major> majors = khoas[i].getMyMajor();

for (int j = 0; j < majors.Count; j++)

{

TreeNode nodeMajor = new TreeNode()

{

Text = majors[j].getName(),

Tag = "2",

ImageIndex = 2,

};

nodeKhoa.Nodes.Add(nodeMajor);

}

root.Nodes.Add(nodeKhoa);

}

tvKhoa.Nodes.Add(root);

tvKhoa.ExpandAll();

}

* Reset giá trị:

void resetDefault()

{

currentSubject = null;

btnDelete.Enabled = false;

gbInfo.Enabled = false;

txtName.Text = "";

}

* Load danh sách môn học:

void loadSubject()

{

if (currentMajor != null)

{

subjects = currentMajor.getMySubjects();

}

else

{

subjects = currentKhoa.getMySubjects();

}

findAndSort();

uploadList();

}

* Hiển thị danh sách môn học:

void uploadList()

{

if (subjects == null)

{

return;

}

lvSubject.Items.Clear();

currentSubject = null;

gbInfo.Enabled = false;

btnDelete.Enabled = false;

for (int i = 0; i < subjects.Count; i++)

{

ListViewItem lvi = new ListViewItem()

{

Text = subjects[i].getName(),

};

lvi.SubItems.Add(new ListViewItem.ListViewSubItem() { Text = subjects[i].getMust() == true ? "X" : "" });

lvi.SubItems.Add(new ListViewItem.ListViewSubItem() { Text = subjects[i].getSoTC().ToString() });

lvi.SubItems.Add(new ListViewItem.ListViewSubItem() { Text = subjects[i].getSoTiet().ToString() });

lvi.SubItems.Add(new ListViewItem.ListViewSubItem() { Text = subjects[i].getMoney().ToString() });

lvi.SubItems.Add(new ListViewItem.ListViewSubItem() { Text = subjects[i].getHocKi().ToString() });

lvi.SubItems.Add(new ListViewItem.ListViewSubItem() { Text = subjects[i].getYear().ToString() });

lvSubject.Items.Add(lvi);

}

}

* Loại bỏ môn học khỏi danh sách môn theo kí tự:

void removeListByCharName(char c)

{

c = char.ToLower(c);

for (int i = 0; i < subjects.Count; i++)

{

bool remove = !subjects[i].getName().ToLower().Contains(c);

if (remove)

{

subjects.RemoveAt(i);

i--;

}

}

}

* Tìm kiếm:

void search()

{

if (!string.IsNullOrEmpty(txtSearch.Text))

{

for (int i = 0; i < txtSearch.Text.Length; i++)

{

removeListByCharName(txtSearch.Text[i]);

}

}

}

* Sắp xếp danh sách theo từng kì, từng năm:

void sortByHocKiAndYear(bool isIncrease = true)

{

bool[] check = new bool[10];

for (int i =0; i < check.Length; i++)

{

check[i] = false;

}

List<List<Subject>> lst = new List<List<Subject>>();

for (int i = 0; i < subjects.Count; i++)

{

if (!check[subjects[i].getYear()])

{

check[subjects[i].getYear()] = true;

List<Subject> tmp = new List<Subject>();

tmp.Add(subjects[i]);

for (int j = i + 1; j < subjects.Count; j++)

{

if (subjects[i].getYear() == subjects[j].getYear())

{

tmp.Add(subjects[j]);

}

}

lst.Add(tmp);

}

}

for (int i = 0; i < lst.Count - 1; i++)

{

for (int j = i + 1; j < lst.Count; j++)

{

if (isIncrease)

{

if (lst[i][0].getYear() > lst[j][0].getYear())

{

List<Subject> tmp = lst[i];

lst[i] = lst[j];

lst[j] = tmp;

}

}

else

{

if (lst[i][0].getYear() < lst[j][0].getYear())

{

List<Subject> tmp = lst[i];

lst[i] = lst[j];

lst[j] = tmp;

}

}

}

}

for (int i = 0; i < lst.Count; i++)

{

for (int j = 0; j < lst[i].Count - 1; j++)

{

for (int k = j + 1; k < lst[i].Count; k++)

{

if (lst[i][j].getHocKi() > lst[i][k].getHocKi())

{

Subject sub = lst[i][j].getCoppy();

lst[i][j] = lst[i][k].getCoppy();

lst[i][k] = sub.getCoppy();

}

}

}

}

subjects = new List<Subject>();

for (int i = 0; i < lst.Count; i++)

{

for (int j = 0; j < lst[i].Count; j++)

{

subjects.Add(lst[i][j]);

}

}

}

* Sắp xếp danh sách theo tên:

void sortByName(bool isIncrease = true)

{

for (int i = 0; i < subjects.Count - 1; i++)

{

for (int j = i + 1; j < subjects.Count; j++)

{

int lengthA = subjects[i].getName().Length;

int lengthB = subjects[j].getName().Length;

int length = lengthA > lengthB ? lengthB : lengthA;

for (int k = 0; k < length; k++)

{

if (isIncrease)

{

if (subjects[i].getName()[k] > subjects[j].getName()[k])

{

Subject subject = subjects[i].getCoppy();

subjects[i] = subjects[j].getCoppy();

subjects[j] = subject.getCoppy();

break;

}

else if (subjects[i].getName()[k] < subjects[j].getName()[k])

{

break;

}

}

else

{

if (subjects[i].getName()[k] < subjects[j].getName()[k])

{

Subject subject = subjects[i].getCoppy();

subjects[i] = subjects[j].getCoppy();

subjects[j] = subject.getCoppy();

break;

}

else if (subjects[i].getName()[k] > subjects[j].getName()[k])

{

break;

}

}

}

}

}

}

* Sắp xếp danh sách môn theo phí:

void sortByMoney(bool isIncrease = true)

{

for (int i = 0; i < subjects.Count - 1; i++)

{

for (int j = i + 1; j < subjects.Count; j++)

{

if (isIncrease)

{

if (subjects[i].getMoney() > subjects[j].getMoney())

{

Subject subject = subjects[i].getCoppy();

subjects[i] = subjects[j].getCoppy();

subjects[j] = subject.getCoppy();

break;

}

else if (subjects[i].getMoney() < subjects[j].getMoney())

{

break;

}

}

else

{

if (subjects[i].getMoney() < subjects[j].getMoney())

{

Subject subject = subjects[i].getCoppy();

subjects[i] = subjects[j].getCoppy();

subjects[j] = subject.getCoppy();

break;

}

else if (subjects[i].getMoney() > subjects[j].getMoney())

{

break;

}

}

}

}

}

* Tìm và sắp xếp danh sách môn:

void findAndSort(RadioButton rdo = null)

{

if (rdo != null)

{

if (rdo == rdoSortHocKiAndYear)

{

sortByHocKiAndYear(currentInscrease);

}

else if (rdo == rdoSortName)

{

sortByName(currentInscrease);

}

else

{

sortByMoney(currentInscrease);

}

}

else

{

if (rdoSortHocKiAndYear.Checked)

{

sortByHocKiAndYear(currentInscrease);

}

else if (rdoSortName.Checked)

{

sortByName(currentInscrease);

}

else

{

sortByMoney(currentInscrease);

}

}

}

* Tìm và lọc danh sách môn:

void findAndFilter()

{

if (currentMajor != null)

{

subjects = currentMajor.getMySubjects();

}

else

{

subjects = currentKhoa.getMySubjects();

}

if (rdoFilterMust.Checked)

{

for (int i = 0; i < subjects.Count; i++)

{

if (!subjects[i].getMust())

{

subjects.RemoveAt(i);

i--;

}

}

}

else if (rdoFilterNotMust.Checked)

{

for (int i = 0; i < subjects.Count; i++)

{

if (subjects[i].getMust())

{

subjects.RemoveAt(i);

i--;

}

}

}

}

### ***3.1.9. Màn hình (Form) Phân công giảng dạy môn học:***

* Mô tả màn hình: Form phân công giảng viên giảng dạy các môn học, mở lớp dạy học cho Sinh viên. Combobox khoa và ngành để lọc, Combobox môn học để chọn môn cho giảng viên dạy, danh sách môn thay đổi theo bộ lọc. Sau khi chọn môn học, danh sách sinh viên được hiển thị trên ListView cho phép check vào để mở lớp đồng thời nhiều lớp cho nhiều giảng viên cho cùng 1 môn học. Các lớp đã mở sẽ được hiển thị trên danh sách các GroupBox
* Hiện thực các phương thức xử lý sự kiện (Event Handlers):
* Click chọn lớp học đã sắp:

private void selectAssignSubject(object sender, EventArgs e)

{

currentInfoAssign = (InfoAssignTeacher)((Control)sender).Tag;

btnCancel.Enabled = true;

}

* Check / Uncheck Giảng viên trên danh sách Giảng viên:

private void lvTeachers\_ItemChecked(object sender, ItemCheckedEventArgs e)

{

if (lvTeachers.CheckedIndices.Count != 0)

{

btnTeach.Enabled = true;

}

else

{

btnTeach.Enabled = false;

}

}

* Click nút Phân công:

private void btnTeach\_Click(object sender, EventArgs e)

{

if (string.IsNullOrEmpty(txtMaxCount.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Bạn chưa nhập sỉ số lớp tối đa!");

txtMaxCount.Focus();

return;

}

for (int i = 0; i < lvTeachers.CheckedIndices.Count; i++)

{

ListViewItem lv = lvTeachers.Items[lvTeachers.CheckedIndices[i]];

InfoAssignTeacher info = new InfoAssignTeacher(currentSubject.getName(), lv.SubItems[0].Text, int.Parse(txtMaxCount.Text));

info.addAssign();

}

MessageInfo.makeMessage("Information", "Chúc mừng", "Phân công thành công!");

loadListTeachers();

loadAssignSubjects();

}

* Click nút Huỷ phân công:

private void btnCancel\_Click(object sender, EventArgs e)

{

if (currentInfoAssign == null)

{

return;

}

if (currentInfoAssign.deleteAssign())

{

MessageInfo.makeMessage("Information", "Chúc mừng", "Xoá phân công thành công!");

loadListTeachers();

loadAssignSubjects();

}

else

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Xoá phân công thất bại!");

}

}

* Lựa chọn môn học:

private void cbbSubject\_SelectedIndexChanged(object sender, EventArgs e)

{

currentSubject = Subject.getInfo("Ten", cbbSubject.Text);

loadListTeachers();

gbTeacher.Enabled = true;

}

* Lựa chọn lọc theo Khoa:

private void cbbKhoa\_SelectedIndexChanged(object sender, EventArgs e)

{

List<Major> majors;

if (cbbKhoa.SelectedIndex == 0)

{

cbbMajor.Enabled = false;

majors = Major.getAllMajor();

}

else

{

cbbMajor.Enabled = true;

Khoa khoa = new Khoa(cbbKhoa.Items[cbbKhoa.SelectedIndex].ToString());

majors = khoa.getMyMajor();

}

cbbMajor.Items.Clear();

cbbMajor.Items.Add("Hiển thị toàn bộ");

for (int i = 0; i< majors.Count; i++)

{

cbbMajor.Items.Add(majors[i].getName());

}

cbbMajor.SelectedIndex = 0;

}

* Lựa chọn lọc môn học theo Ngành:

private void cbbMajor\_SelectedIndexChanged(object sender, EventArgs e)

{

loadAssignSubjects();

loadSubjects();

}

* Hiện thực các phương thức khác:
* Khởi tạo Form:

public frmAssignmentTeacher()

{

InitializeComponent();

txtMaxCount.KeyPress += General.number\_KeyPress;

cbbKhoa.Items.Clear();

List<Khoa> khoas = Khoa.getAllKhoa();

cbbKhoa.Items.Add("Hiển thị toàn bộ");

for (int i = 0; i < khoas.Count; i++)

{

cbbKhoa.Items.Add(khoas[i].getName());

}

cbbKhoa.SelectedIndex = 0;

}

* Tạo khối lớp:

GroupBox getGroupAssignSubject(InfoAssignTeacher infoAssign)

{

GroupBox gb = new GroupBox()

{

Text = "Mã lớp: " + infoAssign.getMaLop(),

Size = new Size(480, 100),

Cursor = Cursors.Hand,

Tag = infoAssign,

BackColor = Color.Teal,

Font = new Font("Times New Roman", 9, FontStyle.Bold),

ForeColor = Color.WhiteSmoke,

};

int xstart = 10, yStart = 30;

Label lblNameSubject = new Label()

{

Text = "Tên môn: " + infoAssign.getNameSubject(),

ForeColor = Color.White,

Location = new Point(xstart, yStart),

AutoSize = true,

};

gb.Controls.Add(lblNameSubject);

yStart += 27;

Label lblMaGV = new Label()

{

Text = "Mã GV: " + infoAssign.getMaGV(),

ForeColor = Color.White,

Location = new Point(xstart, yStart),

AutoSize = true,

};

gb.Controls.Add(lblMaGV);

yStart += 27;

Label lblMaxCount = new Label()

{

Text = "Sỉ số tối đa: " + infoAssign.getMaxCount(),

ForeColor = Color.White,

Location = new Point(xstart, yStart),

AutoSize = true,

};

gb.Controls.Add(lblMaGV);

gb.Click += selectAssignSubject;

gb.Click += selectAssignSubject;

return gb;

}

* Load danh sách môn học:

void loadSubjects()

{

cbbSubject.Items.Clear();

List<Subject> subs;

if (cbbMajor.SelectedIndex == 0)

{

if (cbbKhoa.SelectedIndex == 0)

{

subs = Subject.getAllSubject();

}

else

{

Khoa khoa = new Khoa(cbbKhoa.Items[cbbKhoa.SelectedIndex].ToString());

subs = khoa.getMySubjects();

}

}

else

{

subs = Subject.getSubjectInMajor(cbbMajor.Items[cbbMajor.SelectedIndex].ToString());

}

for (int i = 0; i< subs.Count; i++)

{

cbbSubject.Items.Add(subs[i].getName());

}

}

* Load danh sách lớp học đã sắp:

void loadAssignSubjects()

{

fpnAssignedSubject.Controls.Clear();

List<InfoAssignTeacher> infoAssigns;

if (cbbMajor.SelectedIndex == 0)

{

if (cbbKhoa.SelectedIndex == 0)

{

infoAssigns = InfoAssignTeacher.getAllAssign();

}

else

{

Khoa khoa = new Khoa(cbbKhoa.Items[cbbKhoa.SelectedIndex].ToString());

infoAssigns = InfoAssignTeacher.getInfoAssignByMajor(khoa.getMyMajor());

}

}

else

{

List<Major> majors = new List<Major>() { Major.getInfo(cbbMajor.Items[cbbMajor.SelectedIndex].ToString()) };

infoAssigns = InfoAssignTeacher.getInfoAssignByMajor(majors);

}

for (int i = 0; i < infoAssigns.Count; i++)

{

fpnAssignedSubject.Controls.Add(getGroupAssignSubject(infoAssigns[i]));

}

}

* Load danh sách Giảng viên:

void loadListTeachers()

{

lvTeachers.Items.Clear();

if (currentSubject == null)

{

return;

}

List<Major> majors = currentSubject.getMajor();

List<Teacher> teachers = new List<Teacher>();

for (int i = 0; i < majors.Count; i++)

{

Khoa khoa = new Khoa(majors[i].getTenKhoa());

List<Teacher> tmpTeachers = khoa.getTeachers();

for (int j = 0; j < tmpTeachers.Count; j++)

{

bool ok = true;

for (int k = 0; k < teachers.Count; k++)

{

if (tmpTeachers[j].getID() == teachers[k].getID())

{

ok = false;

break;

}

}

if (ok)

{

teachers.Add(tmpTeachers[j]);

ListViewItem li = new ListViewItem()

{

Text = tmpTeachers[j].getID(),

};

li.SubItems.Add(new ListViewItem.ListViewSubItem() { Text = tmpTeachers[j].getFullName().getFullName() });

lvTeachers.Items.Add(li);

}

}

}

}

### ***3.1.10. Màn hình (Form) Quản lý phòng học:***

* Mô tả màn hình: Form quản lý phòng học và hỗ trợ cho các lớp đã mở đăng ký phòng. Danh sách phòng hiện có sẽ được hiển thị trên danh sách các GroupBox với thông số gồm tên phòng và sức chứa, cho phép click chọn để chỉnh sửa thông tin phòng hay xoá phòng hay chọn nút thêm để thêm mới phòng. Khi chọn vào từng phòng sẽ hiển thị thông tin phòng và các lớp đã đăng ký phòng này trên danh sách GroupBox, cho phép đăng ký thêm cho phòng với danh sách lớp học khả dụng ở trên ListView, để đăng ký Quản trị sẽ nhập đủ thông tin về thời gian dùng phòng và check chọn lớp đăng ký phòng đó. Khi click chọn vào các lớp đã đăng ký cho phép chỉnh sửa thông tin hay xoá đăng ký. Ngoài ra còn cho phép nhập tên phòng vào để tìm kiếm, hay nhập thời gian đăng ký vào và tìm kiếm để hệ thống lọc danh sách phòng đang trống giờ đó.
* Hiện thực các phương thức xử lý sự kiện (Event Handlers):
* Click chọn lớp học:

private void selectSubjectAssign(object sender, EventArgs e)

{

GroupBox gb = (GroupBox)sender;

InfoAssignRoom infoAssignRoom = (InfoAssignRoom)gb.Tag;

InfoAssignTeacher infoAssignTeacher = InfoAssignTeacher.getInfo(infoAssignRoom.getMaLop());

Subject sub = Subject.getInfo("Ten", infoAssignTeacher.getNameSubject());

loadSubject();

showInfoSubject(sub);

dtpDateStart.Value = infoAssignRoom.getDateStart().getDate();

dtpDateEnd.Value = infoAssignRoom.getDateEnd().getDate();

txtTietStart.Text = infoAssignRoom.getTietStart().ToString();

txtTietEnd.Text = infoAssignRoom.getTietEnd().ToString();

checkDayOfWeek(infoAssignRoom.getStrDayOfWeek());

currentAssign = infoAssignRoom;

gbSubject.Enabled = true;

lvClassSubject.Enabled = true;

for (int i = 0; i < lvClassSubject.Items.Count; i++)

{

if (sub.getName() == lvClassSubject.Items[i].SubItems[2].Text && infoAssignRoom.getMaLop() == int.Parse(lvClassSubject.Items[i].SubItems[0].Text) && infoAssignTeacher.getMaGV() == lvClassSubject.Items[i].SubItems[1].Text)

{

lvClassSubject.Items[i].Checked = true;

break;

}

}

btnAssignSubject.Enabled = false;

btnCancelSubject.Enabled = true;

btnSave.Enabled = true;

gbSubject.Text = "Chỉnh sửa thông tin đăng ký";

}

* Lựa chọn lọc danh sách lớp đăng ký phòng:

private void cbbAssignRoom\_SelectedIndexChanged(object sender, EventArgs e)

{

if (cbbAssignRoom.SelectedIndex == -1)

{

cbbAssignRoom.SelectedIndex = 0;

return;

}

if (cbbAssignRoom.SelectedIndex == 0)

{

loadSubjectAssignRoom();

}

else

{

gbSubject.Text = "Đăng ký phòng " + currentRoom.getName();

currentAssign = null;

fpnClassAssign.Controls.Clear();

btnSearch.Enabled = true;

btnAssignSubject.Enabled = false;

btnSave.Enabled = false;

btnCancelSubject.Enabled = false;

string item = cbbAssignRoom.Items[cbbAssignRoom.SelectedIndex].ToString();

int maLop = int.Parse(item.Substring(item.LastIndexOf(' ')));

List<InfoAssignRoom> assignRooms = Classroom.getAssignRoom(currentRoom.getName());

for (int i = 0; i < assignRooms.Count; i++)

{

if (assignRooms[i].getMaLop() == maLop)

{

fpnClassAssign.Controls.Add(getGroupSubjectAssign(assignRooms[i]));

}

}

}

}

* Click chọn phòng học:

private void selectRoom(object sender, EventArgs e)

{

cbbAssignRoom.Enabled = true;

currentRoom = (Classroom)((Control)sender).Tag;

txtName.Text = currentRoom.getName();

txtCountMax.Text = currentRoom.getMaxCount().ToString();

gbInfo.Enabled = true;

lvClassSubject.Enabled = true;

gbSubject.Enabled = true;

btnDelete.Enabled = true;

gbInfo.Text = "Chỉnh sửa thông tin phòng " + currentRoom.getName();

cbbAssignRoom.Items.Clear();

cbbAssignRoom.Items.Add("Toàn bộ đăng ký");

List<InfoAssignRoom> assignRooms = Classroom.getAssignRoom(currentRoom.getName());

for (int i = 0; i < assignRooms.Count; i++)

{

bool isOk = true;

for (int j = 0; j < cbbAssignRoom.Items.Count; j++)

{

if (cbbAssignRoom.Items[j].ToString() == "Mã lớp " + assignRooms[i].getMaLop())

{

isOk = false;

break;

}

}

if (isOk)

{

cbbAssignRoom.Items.Add("Mã lớp " + assignRooms[i].getMaLop());

}

}

cbbAssignRoom.SelectedIndex = 0;

loadSubject();

if (string.IsNullOrEmpty(txtTietStart.Text) || string.IsNullOrEmpty(txtTietEnd.Text))

{

btnAssignSubject.Enabled = false;

btnSave.Enabled = false;

btnCancelSubject.Enabled = false;

}

else

{

btnAssignSubject.Enabled = true;

btnSave.Enabled = false;

btnCancelSubject.Enabled = false;

}

}

* Click nút Thêm phòng:

private void btnAdd\_Click(object sender, EventArgs e)

{

txtName.Clear();

txtCountMax.Clear();

gbInfo.Enabled = true;

currentAssign = null;

currentRoom = null;

gbInfo.Text = "Thêm phòng học";

btnDelete.Enabled = false;

cbbAssignRoom.Enabled = false;

loadRooms();

}

* Click nút Xác nhận:

private void btnAccept\_Click(object sender, EventArgs e)

{

foreach (Control c in gbInfo.Controls)

{

if (c is TextBox)

{

if (string.IsNullOrEmpty(c.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Bạn chưa nhập đủ thông tin phòng!");

c.Focus();

return;

}

}

}

if (currentRoom == null)

{

Classroom room = new Classroom(txtName.Text, int.Parse(txtCountMax.Text));

if (room.addNewRoom())

{

MessageInfo.makeMessage("Information", "Chúc mừng", "Thêm phòng thành công!");

FilterRooms();

}

}

else

{

List<Subject> subs = currentRoom.getSubject();

for (int i = 0; i < subs.Count; i++)

{

List<Teacher> teachers = subs[i].getTeacherTeach();

for (int j = 0; j < teachers.Count; j++)

{

List<InfoAssignTeacher> infoAssigns = InfoAssignTeacher.getAllAssign();

for (int k = 0; k < infoAssigns.Count; k++)

{

List<Student> students = InfoAssignSubject.getStudentsAssignSubject(infoAssigns[k].getMaLop());

if (students.Count > int.Parse(txtCountMax.Text))

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Sửa sỉ số tối đa của phòng thất bại!");

currentRoom.updateRoom(txtName.Text, currentRoom.getMaxCount());

txtCountMax.Text = currentRoom.getMaxCount().ToString();

return;

}

}

}

}

if (currentRoom.updateRoom(txtName.Text, int.Parse(txtCountMax.Text)))

{

MessageInfo.makeMessage("Information", "Chúc mừng", "Đã sửa thành công!");

}

else

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Thêm phòng thất bại!");

}

FilterRooms();

}

}

* Click nút Huỷ đăng ký phòng:

private void btnCancelSubject\_Click(object sender, EventArgs e)

{

if (MessageInfo.makeMessage("Question", "Hỏi lại cho chắc", "Bạn có chắc chắn muốn huỷ đăng ký phòng?") != DialogResult.Yes)

{

return;

}

try

{

if (currentAssign.deleteAssignRoom())

{

MessageInfo.makeMessage("Information", "Chúc mừng", "Huỷ đăng ký phòng thành công!");

loadSubject();

loadSubjectAssignRoom();

btnAssignSubject.Enabled = true;

btnCancelSubject.Enabled = false;

btnSave.Enabled = false;

}

else

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Huỷ đăng ký phòng thất bại!");

}

}

catch

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Huỷ đăng ký phòng thất bại!");

}

}

* Click nút Đăng ký phòng:

private void btnAssignSubject\_Click(object sender, EventArgs e)

{

if (string.IsNullOrEmpty(txtTietStart.Text) || string.IsNullOrEmpty(txtTietEnd.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Bạn chưa nhập đủ thông tin!");

return;

}

if (lvClassSubject.CheckedIndices.Count == 0)

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Bạn chưa chọn lớp để đăng ký phòng!");

return;

}

try

{

Date dateStart = new Date(dtpDateStart.Value);

Date dateEnd = new Date(dtpDateEnd.Value);

if (Date.compareDate(dateStart, dateEnd) == 1)

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Ngày bắt đầu không được lớn hơn ngày kết thúc!");

return;

}

int tietStart = int.Parse(txtTietStart.Text);

int tietEnd = int.Parse(txtTietEnd.Text);

if (tietStart > tietEnd)

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Tiết bắt đầu không được lớn nhớ tiết kết thúc!");

return;

}

string dayOfWeek = findDayOfWeek();

ListViewItem li = lvClassSubject.Items[lvClassSubject.CheckedIndices[0]];

int maLop = int.Parse(li.SubItems[0].Text);

List<InfoAssignRoom> infoRoomsByMaLop = Classroom.getAssignRoom(maLop);

for (int i = 0; i < infoRoomsByMaLop.Count; i++)

{

if (General.isDateAInB(dateStart, dateEnd, tietStart, tietEnd, dayOfWeek, infoRoomsByMaLop[i].getDateStart(), infoRoomsByMaLop[i].getDateEnd(), infoRoomsByMaLop[i].getTietStart(), infoRoomsByMaLop[i].getTietEnd(), infoRoomsByMaLop[i].getStrDayOfWeek()))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Mã lớp " + maLop + " đã đăng ký: " + infoRoomsByMaLop[i].getCalender() + "!");

return;

}

}

List<InfoAssignRoom> infoRoomsByNameRoom = Classroom.getAssignRoom(currentRoom.getName());

for (int i = 0; i < infoRoomsByNameRoom.Count; i++)

{

if (General.isDateAInB(dateStart, dateEnd, tietStart, tietEnd, dayOfWeek, infoRoomsByNameRoom[i].getDateStart(), infoRoomsByNameRoom[i].getDateEnd(), infoRoomsByNameRoom[i].getTietStart(), infoRoomsByNameRoom[i].getTietEnd(), infoRoomsByNameRoom[i].getStrDayOfWeek()))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Phòng đã được đăng ký: " + infoRoomsByNameRoom[i].getCalender() + "!");

return;

}

}

int STT = int.Parse(li.SubItems[0].Text);

InfoAssignRoom info = new InfoAssignRoom(currentRoom.getName(), STT, dateStart, dateEnd, tietStart, tietEnd, dayOfWeek);

if (info.addAssignRoom())

{

MessageInfo.makeMessage("Information", "Chúc mừng", "Đăng ký phòng thành công!");

loadSubject();

loadSubjectAssignRoom();

}

else

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Đăng ký phòng thất bại!");

}

}

catch (Exception ex)

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Đăng ký phòng thất bại! " + ex.Message);

}

}

* Click nút Xoá môn học:

private void btnDelete\_Click(object sender, EventArgs e)

{

if (currentRoom != null)

{

if (MessageInfo.makeMessage("Question", "Hỏi lại cho chắc", "Bạn có chắc chắn muốn xoá phòng " + currentRoom.getName() +"?") != DialogResult.Yes)

{

return;

}

currentRoom.deleteRoom();

currentRoom = null;

btnDelete.Enabled = false;

cbbAssignRoom.Enabled = false;

FilterRooms();

}

}

* Click nút Lưu:

private void btnSave\_Click(object sender, EventArgs e)

{

if (lvClassSubject.CheckedIndices.Count == 0)

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Bạn chưa chọn môn để chỉnh sửa!");

return;

}

ListViewItem li = lvClassSubject.Items[lvClassSubject.CheckedIndices[0]];

int STT = int.Parse(li.SubItems[0].Text);

string dayOfWeek = findDayOfWeek();

InfoAssignRoom info = new InfoAssignRoom(currentRoom.getName(), STT, new Date(dtpDateStart.Value), new Date(dtpDateEnd.Value), int.Parse(txtTietStart.Text), int.Parse(txtTietEnd.Text), dayOfWeek);

if (currentAssign.updateAssignRoom(info))

{

loadSubjectAssignRoom();

currentAssign = info;

}

else

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Chỉnh sửa thông tin lớp thất bại!");

}

}

* Check / Uncheck lớp học trên danh sách lớp học đề xuất:

private void lvSubjects\_ItemChecked(object sender, ItemCheckedEventArgs e)

{

if (lvClassSubject.CheckedIndices.Count == 0 && currentAssign == null)

{

return;

}

if (e.Item.Checked == false)

{

return;

}

if (lvClassSubject.CheckedIndices.Count >= 1)

{

for (int i = 0; i < lvClassSubject.CheckedIndices.Count; i++)

{

if (lvClassSubject.Items[lvClassSubject.CheckedIndices[i]] != e.Item)

{

lvClassSubject.Items[lvClassSubject.CheckedIndices[i]].Checked = false;

}

}

}

gbSubject.Enabled = true;

if (currentAssign == null)

{

btnAssignSubject.Enabled = true;

btnSave.Enabled = false;

btnCancelSubject.Enabled = false;

gbSubject.Text = "Đăng ký phòng " + currentRoom.getName();

}

else

{

btnCancelSubject.Enabled = true;

btnSave.Enabled = true;

btnAssignSubject.Enabled = false;

gbSubject.Text = "Chỉnh sửa thông tin đăng ký phòng " + currentRoom.getName();

}

}

* Click nút tìm kiếm phòng:

private void btnSearch\_Click(object sender, EventArgs e)

{

FilterRooms();

}

* ***Click nút Reset giá trị:***

private void btnReset\_Click(object sender, EventArgs e)

{

resetInputSubject();

btnAssignSubject.Enabled = false;

btnSave.Enabled = false;

btnCancelSubject.Enabled = false;

btnDelete.Enabled = false;

txtName.Clear();

txtCountMax.Clear();

createRooms();

currentRoom = null;

fpnClassAssign.Controls.Clear();

lvClassSubject.Items.Clear();

cbbAssignRoom.Enabled = false;

}

* Hiện thực các phương thức khác:
* ***Tạo khối lớp học:***

GroupBox getGroupSubjectAssign(InfoAssignRoom infoAssignRoom)

{

GroupBox gb = new GroupBox()

{

Text = "Tên phòng: " + infoAssignRoom.getNameRoom(),

Size = new Size(fpnClassAssign.Width - 5, 160),

Cursor = Cursors.Hand,

Tag = infoAssignRoom,

BackColor = Color.LavenderBlush,

Font = new Font("Times New Roman",9,FontStyle.Bold),

};

int xstart = 10, yStart = 30;

Label lblSTT = new Label()

{

Text = "Mã lớp: " + infoAssignRoom.getMaLop(),

ForeColor = Color.Teal,

Location = new Point(xstart, yStart),

AutoSize = true,

};

gb.Controls.Add(lblSTT);

yStart += 27;

InfoAssignTeacher infoAssignTeacher = InfoAssignTeacher.getInfo(infoAssignRoom.getMaLop());

Subject subject = Subject.getInfo("Ten", infoAssignTeacher.getNameSubject());

Label lblNameSubject = new Label()

{

Text = "Tên môn: " + subject.getName().ToString(),

ForeColor = Color.Teal,

Location = new Point(xstart, yStart),

AutoSize = true,

};

gb.Controls.Add(lblNameSubject);

yStart += 27;

Label lblMaGV = new Label()

{

Text = "Mã GV: " + infoAssignTeacher.getMaGV(),

ForeColor = Color.Teal,

Location = new Point(xstart, yStart),

AutoSize = true,

};

gb.Controls.Add(lblMaGV);

yStart += 27;

Label lblDateStart = new Label()

{

Text = infoAssignRoom.getStrDayOfWeek() + ", Từ ngày: " + infoAssignRoom.getDateStart().getDateNotDayOfWeek('/') + " - " + infoAssignRoom.getDateEnd().getDateNotDayOfWeek('/'),

ForeColor = Color.Teal,

Location = new Point(xstart, yStart),

AutoSize = true,

};

gb.Controls.Add(lblDateStart);

yStart += 27;

Label lblTiet = new Label()

{

Text = "Tiết bắt đầu từ: " + infoAssignRoom.getTietStart() + " - " + infoAssignRoom.getTietEnd(),

ForeColor = Color.Teal,

Location = new Point(xstart, yStart),

AutoSize = true,

};

gb.Controls.Add(lblTiet);

gb.Click += selectSubjectAssign;

return gb;

}

* Tạo phòng học:

void createRooms()

{

rooms = Classroom.getAllRooms();

loadRooms();

}

* Reset giá trị:

void resetInputSubject()

{

dtpDateStart.Value = DateTime.Now;

dtpDateEnd.Value = DateTime.Now;

txtTietStart.Clear();

txtTietEnd.Clear();

}

* Load phòng:

void loadRooms()

{

cbbAssignRoom.Items.Clear();

btnAssignSubject.Enabled = false;

btnDelete.Enabled = false;

btnSave.Enabled = false;

gbInfo.Text = "Thông tin phòng";

fpnClassAssign.Controls.Clear();

int xStart = 30, yStart = 20;

int widthRoom = 150;

int heightRoom = 80;

int far = 20;

pnClassRoom.Controls.Clear();

for (int i = 0; i < rooms.Count; i++)

{

GroupBox gb = new GroupBox()

{

Text = rooms[i].getName(),

ForeColor = Color.Purple,

Size = new Size(widthRoom, heightRoom),

Location = new Point(xStart, yStart),

Tag = new Classroom(rooms[i].getName(), rooms[i].getMaxCount()),

Cursor = Cursors.Hand,

Font = new Font("Times New Roman", 10, FontStyle.Bold),

BackColor = Color.OldLace,

};

int x = 10, y = 20;

Label lblMaxCount = new Label()

{

Name = "lblMaxCount",

Location = new Point(x, y),

AutoSize = true,

};

lblMaxCount.Text = "Sức chứa: " + rooms[i].getMaxCount();

gb.Controls.Add(lblMaxCount);

xStart += widthRoom + far;

if (xStart + widthRoom + far >= pnClassRoom.Width)

{

xStart = 30;

yStart += heightRoom + far;

}

gb.Click += selectRoom;

pnClassRoom.Controls.Add(gb);

}

}

* Load danh sách lớp học đăng ký phòng:

void loadSubjectAssignRoom()

{

currentAssign = null;

fpnClassAssign.Controls.Clear();

List<InfoAssignRoom> assignRooms = Classroom.getAssignRoom(currentRoom.getName());

for (int i = 0; i < assignRooms.Count; i++)

{

fpnClassAssign.Controls.Add(getGroupSubjectAssign(assignRooms[i]));

}

gbSubject.Text = "Đăng ký phòng " + currentRoom.getName();

}

* Load danh sách môn học:

void loadSubject()

{

lvClassSubject.Items.Clear();

List<InfoAssignTeacher> infos = InfoAssignTeacher.getAllAssign();

for (int i = 0; i < infos.Count; i++)

{

ListViewItem li = new ListViewItem()

{

Text = infos[i].getMaLop().ToString(),

};

li.SubItems.Add(new ListViewItem.ListViewSubItem() { Text = infos[i].getMaGV() });

li.SubItems.Add(new ListViewItem.ListViewSubItem() { Text = infos[i].getNameSubject() });

lvClassSubject.Items.Add(li);

}

}

* Xác định thứ mấy:

string findDayOfWeek()

{

foreach (Control control in pnDayOfWeek.Controls)

{

if (control is RadioButton)

{

if (((RadioButton)control).Checked)

{

return control.Text;

}

}

}

return null;

}

* Check vào Thứ:

void checkDayOfWeek(string dayOfWeek)

{

foreach (Control control in pnDayOfWeek.Controls)

{

if (control is RadioButton)

{

if (control.Text == dayOfWeek)

{

((RadioButton)control).Checked = true;

}

}

}

}

* Lọc danh sách phòng học:

void FilterRooms()

{

rooms = Classroom.getAllRooms();

Date dateStart = new Date(dtpDateStart.Value);

Date dateEnd = new Date(dtpDateEnd.Value);

if (string.IsNullOrEmpty(txtTietStart.Text) || string.IsNullOrEmpty(txtTietEnd.Text))

{

for (int i = 0; i < rooms.Count; i++)

{

if (!string.IsNullOrEmpty(txtSearch.Text))

{

if (rooms[i].getName() != txtSearch.Text)

{

rooms.RemoveAt(i);

i--;

}

}

}

loadRooms();

return;

}

string dayOfWeek = findDayOfWeek();

int tietStart = int.Parse(txtTietStart.Text);

int tietEnd = int.Parse(txtTietEnd.Text);

for (int i = 0; i < rooms.Count; i++)

{

if (!string.IsNullOrEmpty(txtSearch.Text))

{

if (rooms[i].getName() != txtSearch.Text)

{

rooms.RemoveAt(i);

i--;

continue;

}

}

List<InfoAssignRoom> infoRooms = Classroom.getAssignRoom(rooms[i].getName());

for (int j = 0; j < infoRooms.Count; j++)

{

if (General.isDateAInB(infoRooms[j].getDateStart(), infoRooms[j].getDateEnd(), infoRooms[j].getTietStart(), infoRooms[j].getTietEnd(), infoRooms[j].getStrDayOfWeek() , dateStart, dateEnd, tietStart, tietEnd, dayOfWeek))

{

rooms.RemoveAt(i);

i--;

break;

}

}

}

loadRooms();

}

### ***3.1.11. Màn hình (Form) Sắp lớp cho Sinh viên:***

* **Mô tả màn hình:**Form sắp xếp sinh viên vào các lớp học của các cố vấn. Hỗ trợ tìm kiếm tên lớp để thêm sinh viên, bộ lọc bằng ComboBox để lọc theo khoa và ngành để tìm nhanh lớp. Danh sách lớp học hiện có hiển thị trên GroupBox. Khi click vào lớp sẽ xem được danh sách sinh viên hiện tại trong lớp và danh sách sinh viên có cùng ngành nhưng chưa có lớp để cân nhắc check vào để thêm vào lớp, hoặc check vào sinh viên trong lớp để loại khỏi lớp. Ngoài ra còn sỡ hửu nút loại bỏ toàn bộ để loại hết Sinh viên ở các lớp ra ngoài hay nút loại bỏ cả lớp để chỉ loại toàn bộ Sinh viên trong lớp đó hay nút phân bổ tự động để thêm tự động Sinh viên vào các lớp theo cơ chế lắp đầy 1 lớp rồi tới lớp khác.
* ***Hiện thực các phương thức xử lý sự kiện (Event Handlers):***
* ***Lựa chọn lọc danh sách lớp theo Khoa:***

private void cbbKhoa\_SelectedIndexChanged(object sender, EventArgs e)

{

if (cbbKhoa.SelectedIndex == -1)

{

fpnClasss.Controls.Clear();

gbStudentInClass.Enabled = false;

gbStudents.Enabled = false;

cbbMajor.Text = "";

cbbMajor.Items.Clear();

cbbMajor.Enabled = false;

return;

}

else

{

if (cbbKhoa.SelectedIndex == 0)

{

cbbMajor.Text = "";

cbbMajor.Items.Clear();

cbbMajor.Enabled = false;

loadClass();

}

else

{

cbbMajor.Enabled = true;

loadMajor();

}

}

}

* ***Lựa chọn lọc danh sách lớp theo Ngành:***

private void cbbMajor\_SelectedIndexChanged(object sender, EventArgs e)

{

loadClass();

}

* ***Click chọn lớp học:***

private void selectClass(object sender, EventArgs e)

{

currentClass = (Class)((Control)sender).Tag;

loadListStudentInClass();

loadListStudents();

gbStudentInClass.Enabled = true;

gbStudents.Enabled = true;

gbStudentInClass.Text = "Sinh viên lớp " + currentClass.getName();

}

* ***Check / Uncheck Sinh viên trên danh sách sinh viên chưa có lớp:***

private void lvStudents\_ItemChecked(object sender, ItemCheckedEventArgs e)

{

if (lvStudents.CheckedIndices.Count != 0)

{

btnAdd.Enabled = true;

}

else

{

btnAdd.Enabled = false;

}

}

* ***Check / Uncheck Sinh viên trên danh sách sinh viên thuộc lớp:***

private void lvStudentInClass\_ItemChecked(object sender, ItemCheckedEventArgs e)

{

if (lvStudentInClass.CheckedIndices.Count != 0)

{

btnKick.Enabled = true;

}

else

{

btnKick.Enabled = false;

}

}

* ***Click nút Loại sinh viên khỏi lớp:***

private void btnKick\_Click(object sender, EventArgs e)

{

if (MessageInfo.makeMessage("Question", "Hỏi lại cho chắc", "Bạn có chắc muốn các loại sinh viên khỏi lớp " + currentClass.getName() +"?") != DialogResult.Yes)

{

return;

}

for (int i = 0; i < lvStudentInClass.CheckedIndices.Count; i++)

{

ListViewItem li = lvStudentInClass.Items[lvStudentInClass.CheckedIndices[i]];

InfoAssignStudent.kickStudent(li.Text);

}

loadClass();

loadListStudentInClass();

loadListStudents();

}

* ***Click nút Thêm Sinh viên vào lớp:***

private void btnAdd\_Click(object sender, EventArgs e)

{

for (int i = 0; i < lvStudents.CheckedIndices.Count; i++)

{

ListViewItem li = lvStudents.Items[lvStudents.CheckedIndices[i]];

InfoAssignStudent.addAssignStudent(li.Text, currentClass.getName());

}

loadClass();

loadListStudentInClass();

loadListStudents();

}

* ***TextChanged Tìm kiếm:***

private void txtSearch\_TextChanged(object sender, EventArgs e)

{

loadClass();

}

* ***Click nút Sắp lớp tự động:***

private void btnAssignAll\_Click(object sender, EventArgs e)

{

List<Class> classs = Class.getAllClass();

for (int i = 0; i < classs.Count; i++)

{

Major major = Major.getInfo(classs[i].getNameMajor());

List<Student> studentInMajor = major.getStudents();

for (int j = 0; j < studentInMajor.Count; j++)

{

if (Student.checkHadClass(studentInMajor[j]) || InfoAssignStudent.isExistsAssignStudent(studentInMajor[j].getID()))

{

studentInMajor.RemoveAt(j);

j--;

}

}

while (studentInMajor.Count != 0 && classs[i].getStudents().Count() < classs[i].getMaxCount())

{

if (!InfoAssignStudent.addAssignStudent(studentInMajor[0].getID(), classs[i].getName()))

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Có lỗi trong quá trình phân bổ sinh viên!");

break;

}

studentInMajor.RemoveAt(0);

}

}

loadClass();

loadListStudentInClass();

loadListStudents();

}

* ***Click nút loại bỏ tất cả sinh viên khỏi mọi lớp:***

private void btnKickAll\_Click(object sender, EventArgs e)

{

if (MessageInfo.makeMessage("Question", "Hỏi lại cho chắc", "Bạn có chắc muốn các loại toàn bộ sinh viên khỏi mọi lớp?") != DialogResult.Yes)

{

return;

}

List<Class> classs = Class.getAllClass();

for (int i = 0; i < classs.Count; i++)

{

List<Student> studentInMajor = classs[i].getStudents();

for (int j = 0; j < studentInMajor.Count; j++)

{

if (!InfoAssignStudent.kickStudent(studentInMajor[j].getID()))

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Có lỗi trong quá trình loại bỏ sinh viên!");

break;

}

}

}

loadClass();

loadListStudentInClass();

loadListStudents();

}

* ***Click nút Loại bỏ tất cả sinh viên khỏi lớp:***

private void btnKickAllClass\_Click(object sender, EventArgs e)

{

if (MessageInfo.makeMessage("Question", "Hỏi lại cho chắc", "Bạn có chắc muốn các loại toàn bộ sinh viên khỏi lớp " + currentClass.getName() + "?") != DialogResult.Yes)

{

return;

}

List<Student> studentInMajor = currentClass.getStudents();

for (int i = 0; i < studentInMajor.Count; i++)

{

if (!InfoAssignStudent.kickStudent(studentInMajor[i].getID()))

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Có lỗi trong quá trình loại bỏ sinh viên!");

break;

}

}

loadClass();

loadListStudentInClass();

loadListStudents();

}

* ***Hiện thực các phương thức khác:***
* ***Khởi tạo Form:***

public frmAssignmentStudent()

{

InitializeComponent();

loadKhoa();

}

* ***Load Khoa:***

void loadKhoa()

{

List<Khoa> khoas = Khoa.getAllKhoa();

cbbKhoa.Items.Clear();

cbbKhoa.Items.Add("Toàn bộ khoa");

for (int i =0; i <khoas.Count; i++)

{

cbbKhoa.Items.Add(khoas[i].getName());

}

cbbKhoa.SelectedIndex = 0;

}

* ***Load Ngành:***

void loadMajor()

{

string tenKhoa = cbbKhoa.Text;

Khoa khoa = new Khoa(tenKhoa);

List<Major> majors = khoa.getMyMajor();

cbbMajor.Items.Clear();

cbbMajor.Items.Add("Hiển thị toàn bộ");

for (int i = 0; i < majors.Count; i++)

{

cbbMajor.Items.Add(majors[i].getName());

}

cbbMajor.SelectedIndex = 0;

}

* ***Tạo khối lớp học:***

GroupBox getGroupClass(Class aClass)

{

GroupBox gb = new GroupBox()

{

Text = "Tên lớp: " + aClass.getName(),

Size = new Size(390, 150),

Cursor = Cursors.Hand,

Tag = aClass,

BackColor = Color.LavenderBlush,

};

int xstart = 5, yStart = 30;

Label lblMaCoVan = new Label()

{

Text = "Mã cố vấn: " + aClass.getMaGV(),

ForeColor = Color.Teal,

Location = new Point(xstart, yStart),

AutoSize = true,

};

gb.Controls.Add(lblMaCoVan);

yStart += 27;

Label lblCount = new Label()

{

Text = "Sỉ số: " + aClass.getStudents().Count() + "/" + aClass.getMaxCount(),

ForeColor = Color.Teal,

Location = new Point(xstart, yStart),

AutoSize = true,

};

gb.Controls.Add(lblCount);

yStart += 27;

Label lblNameMajor = new Label()

{

Text = "Tên ngành: " + aClass.getNameMajor(),

ForeColor = Color.Teal,

Location = new Point(xstart, yStart),

AutoSize = true,

};

gb.Controls.Add(lblNameMajor);

gb.Click += selectClass;

return gb;

}

* ***Load danh sách sinh viên trong lớp:***

void loadListStudentInClass()

{

btnKick.Enabled = false;

lvStudentInClass.Items.Clear();

if (currentClass == null)

{

return;

}

studentsInClass = currentClass.getStudents();

for (int i = 0; i < studentsInClass.Count; i++)

{

ListViewItem li = new ListViewItem()

{

Text = studentsInClass[i].getID(),

};

li.SubItems.Add(new ListViewItem.ListViewSubItem() { Text = studentsInClass[i].getFullName().getFullName() });

lvStudentInClass.Items.Add(li);

}

}

* ***Load danh sách sinh viên chưa có lớp:***

void loadListStudents()

{

if (currentClass == null)

{

return;

}

students = Student.getAllStudents();

lvStudents.Items.Clear();

for (int i = 0; i < students.Count; i++)

{

if (students[i].getNameMajor() != currentClass.getNameMajor() || Student.checkHadClass(students[i]))

{

continue;

}

ListViewItem li = new ListViewItem()

{

Text = students[i].getID(),

};

li.SubItems.Add(new ListViewItem.ListViewSubItem() { Text = students[i].getFullName().getFullName() });

lvStudents.Items.Add(li);

}

btnAdd.Enabled = false;

}

* ***Load danh sách lớp:***

void loadClass()

{

classs = Class.getAllClass();

for (int i = 0; i < txtSearch.Text.Length; i++)

{

removeListByCharName(txtSearch.Text[i]);

}

Khoa khoa = null;

if (cbbKhoa.SelectedIndex != 0)

{

khoa = new Khoa(cbbKhoa.Items[cbbKhoa.SelectedIndex].ToString());

}

fpnClasss.Controls.Clear();

for (int i = 0; i < classs.Count; i++)

{

if (cbbMajor.SelectedIndex == 0 && cbbKhoa.SelectedIndex != 0)

{

if (!khoa.majorInKhoa(classs[i].getNameMajor()))

{

classs.RemoveAt(i);

i--;

continue;

}

}

else if (cbbMajor.SelectedIndex > 0 && cbbMajor.Items[cbbMajor.SelectedIndex].ToString() != classs[i].getNameMajor())

{

classs.RemoveAt(i);

i--;

continue;

}

fpnClasss.Controls.Add(getGroupClass(classs[i]));

}

}

* ***Loại lớp khỏi danh sách theo kí tự:***

void removeListByCharName(char c)

{

c = char.ToLower(c);

for (int i = 0; i < classs.Count; i++)

{

bool remove = !classs[i].getName().ToLower().Contains(c);

if (remove)

{

classs.RemoveAt(i);

i--;

}

}

}

### ***3.1.12. Màn hình (Form) Quản lý khảo sát***

* **Mô tả màn hình:** Form quản lý các bài khảo sát của các lớp học từ Sinh viên gửi lên. ComboBox lọc danh sách lớp theo Giảng viên, khi click vào lớp học sẽ hiển thị danh sách bài khảo sát trên ListView, khi chọn vào bài khảo sát sẽ xem được nội dung khảo sát và có thể xoá bài khảo sát đó hay xoá toàn bộ khảo sát của Giảng viên đó hay của toàn bộ Giảng viên.
* ***Hiện thực các phương thức xử lý sự kiện (Event Handlers):***
* ***Click chọn lớp học để xem khảo sát:***

private void selectInfoAssign(object sender, EventArgs e)

{

currentAssign = (InfoAssignTeacher)((Control)sender).Tag;

cbbSurvey.Enabled = true;

cbbSurvey.SelectedIndex = -1;

cbbSurvey.SelectedIndex = 0;

btnDeleteAllSurveyGV.Enabled = true;

}

* ***Lựa chọn xem khảo sát của học kì, năm nào:***

private void cbbSurvey\_SelectedIndexChanged(object sender, EventArgs e)

{

if (cbbSurvey.SelectedIndex == -1)

{

return;

}

lvStudent.Items.Clear();

List<Survey> survey = currentAssign.getAllSurveys();

for (int i = 0; i < survey.Count; i++)

{

Student student = (Student)Person.getInfo("ID", survey[i].getMaSV());

if (cbbSurvey.SelectedIndex == 0)

{

if (survey[i].getHocKi() != student.getHocKi() || survey[i].getYear() != student.getYear())

{

continue;

}

}

else

{

if (survey[i].getHocKi() == student.getHocKi() && survey[i].getYear() == student.getYear())

{

continue;

}

}

ListViewItem li = new ListViewItem()

{

Text = survey[i].getMaSV(),

};

li.SubItems.Add(new ListViewItem.ListViewSubItem() { Text = (string)Student.getFieldStudentByID("Ten", survey[i].getMaSV()) });

li.SubItems.Add(new ListViewItem.ListViewSubItem() { Text = survey[i].getHocKi().ToString() });

li.SubItems.Add(new ListViewItem.ListViewSubItem() { Text = survey[i].getYear().ToString() });

lvStudent.Items.Add(li);

}

clearSurvey();

}

* ***Click nút Xoá khảo sát:***

private void btnDelete\_Click(object sender, EventArgs e)

{

ListViewItem li = lvStudent.Items[lvStudent.SelectedIndices[0]];

if (MessageInfo.makeMessage("Question", "Hỏi lại cho chắc", "Bạn có chắc muốn xoá khảo sát: " + li.SubItems[0].Text + "-" + currentAssign.getMaLop() + "-" + li.SubItems[2].Text + "-" + li.SubItems[3].Text + "?") != DialogResult.Yes)

{

return;

}

if (!Survey.deleteSurvey(li.SubItems[0].Text, currentAssign.getMaLop(), int.Parse(li.SubItems[2].Text), int.Parse(li.SubItems[3].Text)))

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Xoá khảo sát thất bại!");

return;

}

loadSubjectClass();

}

* ***Chọn Sinh viên tham gia khảo sát để xem nội dung khảo sát:***

private void lvStudent\_SelectedIndexChanged(object sender, EventArgs e)

{

if (lvStudent.SelectedIndices.Count != 0)

{

btnDelete.Enabled = true;

gbSurvey.Enabled = true;

ListViewItem li = lvStudent.Items[lvStudent.SelectedIndices[0]];

Survey survey = Survey.getSurvey(li.SubItems[0].Text, currentAssign.getMaLop(), int.Parse(li.SubItems[2].Text), int.Parse(li.SubItems[3].Text));

if (survey == null)

{

return;

}

Student student = (Student)Person.getInfo("ID", survey.getMaSV());

lblNameStudent.Text = student.getFullName().getFullName();

object img = Account.getField("AnhDaiDien", "ID", student.getID());

if (img != null)

{

ptrAvt.Image = Avatar.ConvertBytesToImage((byte[])img);

}

else

{

ptrAvt.Image = Image.FromFile("Assets/Imgs/No Image.png");

}

int[] content = survey.getContent();

foreach (Control pn in gbSurvey.Controls)

{

if (pn is Panel)

{

int index = int.Parse(pn.Tag.ToString()) - 1;

foreach (Control rdo in pn.Controls)

{

if (rdo is RadioButton)

{

int value = int.Parse(rdo.Tag.ToString());

if (value == content[index])

{

RadioButton radio = (RadioButton)rdo;

radio.Checked = true;

break;

}

}

}

}

}

}

else

{

btnDelete.Enabled = false;

clearSurvey();

}

}

* ***Lọc danh sách lớp theo mã Giảng viên:***

private void cbbMaGV\_SelectedIndexChanged(object sender, EventArgs e)

{

cbbSurvey.Enabled = false;

btnDeleteAllSurveyGV.Enabled = true;

loadSubjectClass();

}

* ***Click nút Xoá toàn bộ bài khảo sát:***

private void btnDeleteAll\_Click(object sender, EventArgs e)

{

if (MessageInfo.makeMessage("Question", "Hỏi lại cho chắc", "Bạn có chắc muốn muốn xoá toàn bộ khảo sát!") != DialogResult.Yes)

{

return;

}

if (!Survey.deleteAllSurvey())

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Xoá toàn bộ khảo sát thất bại!");

}

else

{

loadSubjectClass();

}

}

* ***Click nút Xoá tất cả bài khảo sát của Giảng viên cụ thể:***

private void btnDeleteAllSurveyGV\_Click(object sender, EventArgs e)

{

if (MessageInfo.makeMessage("Question", "Hỏi lại cho chắc", "Bạn có chắc muốn muốn xoá toàn bộ khảo sát của giảng viên + " + cbbMaGV.Text + "!") != DialogResult.Yes)

{

return;

}

if (cbbMaGV.SelectedIndex == 0)

{

if (!Survey.deleteAllSurvey(currentAssign.getMaGV()))

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Xoá toàn bộ khảo sát của giảng viên " + currentAssign.getMaGV() + " thất bại!");

}

else

{

loadSubjectClass();

}

}

else

{

if (!Survey.deleteAllSurvey(cbbMaGV.Text))

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Xoá toàn bộ khảo sát của giảng viên " + cbbMaGV.Text + " thất bại!");

}

else

{

loadSubjectClass();

}

}

}

* ***Hiện thực các phương thức khác:***
* ***Khởi tạo Form:***

public frmManageSurvey()

{

InitializeComponent();

cbbMaGV.Items.Clear();

cbbMaGV.Items.Add("Hiển thị toàn bộ");

teaches = Teacher.getAllTeacher();

for (int i = 0; i < teaches.Count; i++)

{

cbbMaGV.Items.Add(teaches[i].getID());

}

cbbMaGV.SelectedIndex = 0;

}

* ***Tạo khối lớp để xem khảo sát:***

GroupBox getGroupSurvey(InfoAssignTeacher infoAssign)

{

GroupBox gb = new GroupBox()

{

Size = new Size(490, 150),

Font = new Font("Times New Roman", 10, FontStyle.Bold),

Cursor = Cursors.Hand,

Tag = infoAssign,

BackColor = Color.LavenderBlush,

};

int xstart = 5, yStart = 10;

Label lblMaLop = new Label()

{

Text = "Mã lớp: " + infoAssign.getMaLop(),

ForeColor = Color.Teal,

Location = new Point(xstart, yStart),

AutoSize = true,

};

gb.Controls.Add(lblMaLop);

yStart += 27;

Label lblNameSubject = new Label()

{

Text = "Tên môn: " + infoAssign.getNameSubject(),

ForeColor = Color.Teal,

Location = new Point(xstart, yStart),

AutoSize = true,

};

gb.Controls.Add(lblNameSubject);

yStart += 27;

Label lblMaGV = new Label()

{

Text = "Mã GV: " + infoAssign.getMaGV(),

ForeColor = Color.Teal,

Location = new Point(xstart, yStart),

AutoSize = true,

};

gb.Controls.Add(lblMaGV);

yStart += 27;

Label lblCount = new Label()

{

Text = "Sỉ số: " + infoAssign.getStudents().Count() + "/" + infoAssign.getMaxCount(),

ForeColor = Color.Teal,

Location = new Point(xstart, yStart),

AutoSize = true,

};

gb.Controls.Add(lblCount);

yStart += 27;

Label lblCountSurvey = new Label()

{

Text = "Số bài khảo sát: " + InfoAssignSubject.getSurveysAssignSubject(infoAssign.getMaLop()).Count() + "/" + infoAssign.getStudents().Count(),

ForeColor = Color.Teal,

Location = new Point(xstart, yStart),

AutoSize = true,

};

gb.Controls.Add(lblCountSurvey);

gb.Click += selectInfoAssign;

return gb;

}

* ***Dọn dẹp khảo sát:***

void clearSurvey()

{

gbSurvey.Enabled = false;

ptrAvt.Image = Image.FromFile("Assets/Imgs/No Image.png");

lblNameStudent.Text = "Họ và tên";

foreach (Control pn in gbSurvey.Controls)

{

if (pn is Panel)

{

foreach (Control rdo in pn.Controls)

{

if (rdo is RadioButton)

{

int value = int.Parse(rdo.Tag.ToString());

if (value == 1)

{

RadioButton radio = (RadioButton)rdo;

radio.Checked = true;

break;

}

}

}

}

}

}

* ***Load danh sách các lớp học:***

void loadSubjectClass()

{

lvStudent.Items.Clear();

List<InfoAssignTeacher> infoAssignTeachers;

if (cbbMaGV.SelectedIndex == 0)

{

infoAssignTeachers = InfoAssignTeacher.getAllAssign();

btnDeleteAllSurveyGV.Enabled = false;

}

else

{

btnDeleteAllSurveyGV.Enabled = true;

infoAssignTeachers = teaches[cbbMaGV.SelectedIndex - 1].getInfoAssign();

}

fpnSubjectClass.Controls.Clear();

for (int i = 0; i < infoAssignTeachers.Count; i++)

{

fpnSubjectClass.Controls.Add(getGroupSurvey(infoAssignTeachers[i]));

}

clearSurvey();

}

### ***3.1.13. Màn hình (Form) Xem lịch dạy / Xem thời khoá biểu:***

* **Mô tả màn hình:** Form xem lịch dạy đối với Giảng viên hay xem lịch học đối với Sinh viên. Lịch học hiển thị bằng các label, mỗi label là 1 ô trong ma trận với dòng tiết và cột thứ. Những buổi học hiển thị bằng GroupBox đè lên các ô trên lịch để biết học tiết mấy tới tiết mấy vào thứ mấy. ComboBox để chọn tuần xem lịch, ComboBox chọn năm và học kì của sinh viên để xem lịch học theo học kì và năm nào.
* ***Hiện thực các phương thức xử lý sự kiện (Event Handlers):***
* ***Chọn năm để xem lịch:***

private void cbbYear\_SelectedIndexChanged(object sender, EventArgs e)

{

cbbHocKi.Items.Clear();

if (cbbYear.SelectedIndex + 1 == student.getYear())

{

for (int i = 1; i <= student.getHocKi(); i++)

{

cbbHocKi.Items.Add("Học kì " + i);

}

cbbHocKi.SelectedIndex = student.getHocKi() - 1;

}

else

{

for (int i = 1; i <= 2; i++)

{

cbbHocKi.Items.Add("Học kì " + i);

}

cbbHocKi.SelectedIndex = 0;

}

}

* ***Lựa chọn học kì để xem lịch:***

private void cbbHocKi\_SelectedIndexChanged(object sender, EventArgs e)

{

loadDate();

}

* ***Click nút xem lịch tuần trước:***

private void btnPrev\_Click(object sender, EventArgs e)

{

cbbDate.SelectedIndex--;

}

* ***Click nút xem lịch tuần sau:***

private void btnNext\_Click(object sender, EventArgs e)

{

cbbDate.SelectedIndex++;

}

* ***Lựa chọn ngày trong tuần để xem lịch:***

private void cbbDate\_SelectedIndexChanged(object sender, EventArgs e)

{

if (cbbDate.SelectedIndex == -1)

{

return;

}

enableButton();

loadCellCalendar();

string strDateStart = cbbDate.Text.Substring(0, cbbDate.Text.IndexOf(' '));

Date dateStart = new Date(strDateStart, '/');

if (student != null)

{

int l = 0;

while (dateStart.getStrDayOfWeek() != "Chủ Nhật")

{

if (cbbYear.SelectedIndex + 1 == student.getYear() && cbbHocKi.SelectedIndex + 1 == student.getHocKi())

{

for (int j = 0; j < infoAssignRooms.Count; j++)

{

if (dateStart.getStrDayOfWeek() == infoAssignRooms[j].getStrDayOfWeek())

{

if (Date.compareDate(infoAssignRooms[j].getDateStart(), dateStart) <= 0 && Date.compareDate(infoAssignRooms[j].getDateEnd(), dateStart) >= 0)

{

int width = widthCell;

int height = 0;

for (int k = infoAssignRooms[j].getTietStart(); k <= infoAssignRooms[j].getTietEnd(); k++)

{

height += heightCell;

pnCalendar.Controls.Remove(cells[k - 1, l]);

}

string[] split = cells[infoAssignRooms[j].getTietStart() - 1, l].Tag.ToString().Split(' ');

int yStart = int.Parse(split[1]);

int xStart = int.Parse(split[0]);

FlowLayoutPanel fpn = getBox(width, height, xStart, yStart);

InfoAssignTeacher infoAssignTeacher = InfoAssignTeacher.getInfo(infoAssignRooms[j].getMaLop());

fpn.Controls.Add(new Label() { Text = infoAssignTeacher.getNameSubject(), Size = new Size(widthCell - 10, 50), Font = new Font("Times New Roman", 10, FontStyle.Bold), ForeColor = Color.Navy });

fpn.Controls.Add(getLabel("Mã lớp: " + infoAssignRooms[j].getMaLop()));

fpn.Controls.Add(getLabel(infoAssignRooms[j].getNameRoom()));

fpn.Controls.Add(getLabel("Mã GV: " + infoAssignTeacher.getMaGV()));

pnCalendar.Controls.Add(fpn);

}

}

}

}

else

{

for (int j = 0; j < historyInfoAssignRooms.Count; j++)

{

if (dateStart.getStrDayOfWeek() == historyInfoAssignRooms[j].getInfoAssignRoom().getStrDayOfWeek())

{

if (Date.compareDate(historyInfoAssignRooms[j].getInfoAssignRoom().getDateStart(), dateStart) <= 0 && Date.compareDate(historyInfoAssignRooms[j].getInfoAssignRoom().getDateEnd(), dateStart) >= 0)

{

int width = widthCell;

int height = 0;

for (int k = historyInfoAssignRooms[j].getInfoAssignRoom().getTietStart(); k <= historyInfoAssignRooms[j].getInfoAssignRoom().getTietEnd(); k++)

{

height += heightCell;

pnCalendar.Controls.Remove(cells[k - 1, l]);

}

string[] split = cells[historyInfoAssignRooms[j].getInfoAssignRoom().getTietStart() - 1, l].Tag.ToString().Split(' ');

int yStart = int.Parse(split[1]);

int xStart = int.Parse(split[0]);

FlowLayoutPanel fpn = getBox(width, height, xStart, yStart);

fpn.Controls.Add(new Label() { Text = historyInfoAssignRooms[j].getNameSubject(), Size = new Size(widthCell - 10, 40), Font = new Font("Times New Roman", 10, FontStyle.Bold), ForeColor = Color.Navy });

fpn.Controls.Add(getLabel("Mã lớp: " + historyInfoAssignRooms[j].getInfoAssignRoom().getMaLop()));

fpn.Controls.Add(getLabel(historyInfoAssignRooms[j].getInfoAssignRoom().getNameRoom()));

fpn.Controls.Add(getLabel("Mã GV: " + historyInfoAssignRooms[j].getMaGV()));

pnCalendar.Controls.Add(fpn);

}

}

}

}

dateStart = Date.getNextDate(dateStart);

l++;

}

}

else

{

int l = 0;

while (dateStart.getStrDayOfWeek() != "Chủ Nhật")

{

for (int j = 0; j < infoAssignRooms.Count; j++)

{

if (dateStart.getStrDayOfWeek() == infoAssignRooms[j].getStrDayOfWeek())

{

if (Date.compareDate(infoAssignRooms[j].getDateStart(), dateStart) <= 0 && Date.compareDate(infoAssignRooms[j].getDateEnd(), dateStart) >= 0)

{

int width = widthCell;

int height = 0;

for (int k = infoAssignRooms[j].getTietStart(); k <= infoAssignRooms[j].getTietEnd(); k++)

{

height += heightCell;

pnCalendar.Controls.Remove(cells[k - 1, l]);

}

string[] split = cells[infoAssignRooms[j].getTietStart() - 1, l].Tag.ToString().Split(' ');

int yStart = int.Parse(split[1]);

int xStart = int.Parse(split[0]);

FlowLayoutPanel fpn = getBox(width, height, xStart, yStart);

InfoAssignTeacher info = InfoAssignTeacher.getInfo(infoAssignRooms[j].getMaLop());

fpn.Controls.Add(new Label() { Text = info.getNameSubject(), Size = new Size(widthCell - 10, 50), Font = new Font("Times New Roman", 10, FontStyle.Bold), ForeColor = Color.Navy });

fpn.Controls.Add(getLabel("Mã lớp: " + infoAssignRooms[j].getMaLop()));

fpn.Controls.Add(getLabel(infoAssignRooms[j].getNameRoom()));

fpn.Controls.Add(getLabel("Mã GV: " + info.getMaGV()));

pnCalendar.Controls.Add(fpn);

}

}

}

dateStart = Date.getNextDate(dateStart);

l++;

}

}

}

* ***Hiện thực các phương thức khác:***
* ***Khởi tạo Form:***

public frmWatchCalendar(string nameAcct)

{

InitializeComponent();

string ID = (string) Account.getField("ID", "TenTK", nameAcct);

loadCellCalendar();

if (Student.isExistStudentID(ID))

{

student = (Student)Person.getInfo("ID", ID);

cbbYear.Items.Clear();

for (int i = 1; i <= student.getYear();i++)

{

cbbYear.Items.Add("Năm " + i);

}

cbbYear.SelectedIndex = student.getYear() - 1;

}

else

{

Text = "Xem lịch dạy học";

lblTitle.Text = "XEM LỊCH DẠY HỌC";

lblTitle.Left = cbbDate.Left + cbbDate.Width / 2 - lblTitle.Width / 2;

teacher = (Teacher)Person.getInfo("ID", ID);

loadDate();

Controls.Remove(cbbHocKi);

Controls.Remove(cbbYear);

}

}

* ***Load các tuần:***

void loadDate()

{

startDate = endDate = null;

infoAssignRooms = new List<InfoAssignRoom>();

if (student != null)

{

List<InfoAssignSubject> infoAssignSubjects = student.getInfoAssignSubject();

for (int i = 0; i < infoAssignSubjects.Count; i++)

{

if (infoAssignSubjects[i].getHocKi() != cbbHocKi.SelectedIndex + 1 || infoAssignSubjects[i].getYear() != cbbYear.SelectedIndex + 1)

{

infoAssignSubjects.RemoveAt(i);

i--;

continue;

}

if (cbbYear.SelectedIndex + 1 == student.getYear() && cbbHocKi.SelectedIndex + 1 == student.getHocKi())

{

List<InfoAssignRoom> tmpInfoAssignRooms = InfoAssignRoom.getInfo(infoAssignSubjects[i].getMaLop());

for (int j = 0; j < tmpInfoAssignRooms.Count; j++)

{

if (startDate == null)

{

startDate = tmpInfoAssignRooms[j].getDateStart();

}

else if (Date.compareDate(tmpInfoAssignRooms[j].getDateStart(), startDate) == -1)

{

startDate = tmpInfoAssignRooms[j].getDateStart();

}

if (endDate == null)

{

endDate = tmpInfoAssignRooms[j].getDateEnd();

}

else if (Date.compareDate(tmpInfoAssignRooms[j].getDateEnd(), endDate) == 1)

{

endDate = tmpInfoAssignRooms[j].getDateEnd();

}

infoAssignRooms.Add(tmpInfoAssignRooms[j]);

}

}

else

{

historyInfoAssignRooms = HistoryAssignRoom.getInfo(student.getID(), cbbHocKi.SelectedIndex + 1, cbbYear.SelectedIndex + 1);

for (int j = 0; j < historyInfoAssignRooms.Count; j++)

{

if (historyInfoAssignRooms[j].getInfoAssignRoom().getMaLop() != infoAssignSubjects[i].getMaLop())

{

historyInfoAssignRooms.RemoveAt(j);

j--;

continue;

}

Date dateStart = historyInfoAssignRooms[j].getInfoAssignRoom().getDateStart();

Date dateEnd = historyInfoAssignRooms[j].getInfoAssignRoom().getDateStart();

if (startDate == null)

{

startDate = dateStart;

}

else if (Date.compareDate(dateStart, startDate) == -1)

{

startDate = dateStart;

}

if (endDate == null)

{

endDate = dateEnd;

}

else if (Date.compareDate(dateEnd, endDate) == 1)

{

endDate = dateEnd;

}

}

}

}

}

else

{

List<InfoAssignTeacher> infoAssignTeachers = teacher.getInfoAssign();

for (int i = 0; i < infoAssignTeachers.Count; i++)

{

List<InfoAssignRoom> tmpInfoAssignRooms = InfoAssignRoom.getInfo(infoAssignTeachers[i].getMaLop());

for (int j = 0; j < tmpInfoAssignRooms.Count; j++)

{

if (startDate == null)

{

startDate = tmpInfoAssignRooms[j].getDateStart();

}

else if (Date.compareDate(tmpInfoAssignRooms[j].getDateStart(), startDate) == -1)

{

startDate = tmpInfoAssignRooms[j].getDateStart();

}

if (endDate == null)

{

endDate = tmpInfoAssignRooms[j].getDateEnd();

}

else if (Date.compareDate(tmpInfoAssignRooms[j].getDateEnd(), endDate) == 1)

{

endDate = tmpInfoAssignRooms[j].getDateEnd();

}

infoAssignRooms.Add(tmpInfoAssignRooms[j]);

}

}

}

cbbDate.Items.Clear();

if (startDate != null)

{

while(startDate.getDayOfWeek() != DayOfWeek.Monday)

{

startDate = Date.getPrevDate(startDate);

}

while (endDate.getDayOfWeek() != DayOfWeek.Sunday)

{

endDate = Date.getNextDate(endDate);

}

Date date = new Date(startDate.getStrDayOfWeek(), startDate.getDay(), startDate.getMonth(), startDate.getYear());

while (Date.compareDate(date, endDate) == -1)

{

string itemDate = date.getDateNotDayOfWeek('/');

for (int i = 0; i < 6; i++)

{

date = Date.getNextDate(date);

}

itemDate += " đến " +date.getDateNotDayOfWeek('/');

cbbDate.Items.Add(itemDate);

date = Date.getNextDate(date);

}

cbbDate.SelectedIndex = 0;

}

if (cbbDate.Items.Count == 0)

{

btnNext.Enabled = false;

btnPrev.Enabled = false;

}

else

{

btnPrev.Enabled = false;

}

}

* ***Load các ô trên lịch:***

void loadCellCalendar()

{

pnCalendar.Controls.Clear();

int x = 0;

int y = 0;

cells = new Label[14, 7];

for (int i = 0; i < 14; i++)

{

for (int j = 0; j < 7; j++)

{

cells[i, j] = new Label()

{

AutoSize = false,

Size = new Size(widthCell, heightCell),

Location = new Point(x, y),

BorderStyle = BorderStyle.FixedSingle,

Tag = x + " " + y,

BackColor= Color.White,

};

pnCalendar.Controls.Add(cells[i, j]);

x += widthCell;

}

x = 0;

y += heightCell;

}

}

* ***Tạo 1 ô trên lịch:***

Label getLabel(string text)

{

return new Label()

{

Text = text,

Font = new Font("Times New Roman", 10, FontStyle.Bold),

ForeColor = Color.Indigo,

AutoSize = false,

Size = new Size(widthCell - 10, 30),

};

}

* ***Bất / Tắt nút tiến, lùi lịch:***

void enableButton()

{

if (cbbDate.SelectedIndex == 0)

{

btnPrev.Enabled = false;

}

else

{

btnPrev.Enabled = true;

}

if (cbbDate.SelectedIndex == cbbDate.Items.Count - 1)

{

btnNext.Enabled = false;

}

else

{

btnNext.Enabled = true;

}

}

* ***Tạo khối lớp học dạy trên lịch:***

FlowLayoutPanel getBox(int width, int height, int xStart, int yStart)

{

return new FlowLayoutPanel()

{

Size = new Size(width, height),

Location = new Point(xStart, yStart),

AutoScroll = true,

BackColor = Color.Pink,

BorderStyle = BorderStyle.FixedSingle,

};

}

### ***3.1.14. Màn hình (Form) Quản lý / Chấm điểm sinh viên:***

* **Mô tả màn hình:** Form quản lý danh sách sinh viên và có thể chấm điểm cho các sinh viên đang theo học. Danh sách sinh viên hiển thị trên ListView cho phép check vào để loại khỏi lớp 1 lượt. Khi chọn vào sinh viên sẽ xem được thông tin sinh viên, đối với sinh viên đang theo học sẽ chấm điểm hay xoá điểm đã chấm của sinh viên đó.
* ***Hiện thực các phương thức xử lý sự kiện (Event Handlers):***
* ***Lựa chọn phần trăm (%) điểm thành phần:***

private void changePercent\_ComboBox(object sender, EventArgs e)

{

if (cbbPercentKTDK.SelectedIndex == cbbPercentKTHP.Items.Count - cbbPercentKTHP.SelectedIndex - 1)

{

return;

}

ComboBox currentCbb = (ComboBox)sender;

if (currentCbb == cbbPercentKTDK)

{

cbbPercentKTHP.SelectedIndex = currentCbb.Items.Count - currentCbb.SelectedIndex - 1;

}

else

{

cbbPercentKTDK.SelectedIndex = currentCbb.Items.Count - currentCbb.SelectedIndex - 1;

}

}

* ***Click chọn lớp đang dạy:***

private void selectInfoAssign(object sender, EventArgs e)

{

currentAssignSubject = (InfoAssignTeacher)((Control)sender).Tag;

cbbStudent.SelectedIndex = -1;

cbbStudent.SelectedIndex = 0;

}

* ***Lựa chọn xem danh sách lớp hiện tại / cũ:***

private void cbbStudent\_SelectedIndexChanged(object sender, EventArgs e)

{

if (cbbStudent.SelectedIndex == -1)

{

return;

}

lvStudentLearn.Items.Clear();

List<Student> students = InfoAssignSubject.getStudentsAssignSubject(currentAssignSubject.getMaLop());

for (int j = 0; j < students.Count; j++)

{

List<InfoAssignSubject> infoAssigns = InfoAssignSubject.getAssign(students[j].getID(), currentAssignSubject.getMaLop());

for (int i = 0; i < infoAssigns.Count; i++)

{

if (cbbStudent.SelectedIndex == 0)

{

if (students[j].getHocKi() != infoAssigns[i].getHocKi() || students[j].getYear() != infoAssigns[i].getYear())

{

continue;

}

}

else

{

if (students[j].getHocKi() == infoAssigns[i].getHocKi() && students[j].getYear() == infoAssigns[i].getYear())

{

continue;

}

}

ListViewItem li = new ListViewItem()

{

Text = students[j].getID(),

};

li.SubItems.Add(new ListViewItem.ListViewSubItem() { Text = students[j].getFullName().getFullName() });

li.SubItems.Add(new ListViewItem.ListViewSubItem() { Text = infoAssigns[i].getHocKi().ToString() });

li.SubItems.Add(new ListViewItem.ListViewSubItem() { Text = infoAssigns[i].getYear().ToString() });

lvStudentLearn.Items.Add(li);

}

}

clearText();

}

* ***KeyPress thanh nhập điểm:***

private void keyPressInputValueMark(object sender, KeyPressEventArgs e)

{

if (e.KeyChar != ',' && (char.IsLetter(e.KeyChar) || char.IsSymbol(e.KeyChar) || (!char.IsLetterOrDigit(e.KeyChar) && e.KeyChar > 30)))

{

e.Handled = true;

}

}

* ***Click nút Chấm điểm:***

private void btnMark\_Click(object sender, EventArgs e)

{

if (string.IsNullOrEmpty(txtKTDK.Text) || string.IsNullOrEmpty(txtKTHP.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Điểm không thể để trống!");

return;

}

int percentKTDK = (cbbPercentKTDK.SelectedIndex + 1) \* 10;

List<MarkSubject> pointSubs = MarkSubject.getAllPoint();

for (int i = 0; i < pointSubs.Count; i++)

{

pointSubs[i].updateMarkSubject(pointSubs[i].getKTDK(), pointSubs[i].getKTHP(), percentKTDK);

}

Student student = (Student)Person.getInfo("ID", currentAcct.getID());

InfoAssignSubject infoAssign = InfoAssignSubject.getInfo(student.getID(), currentAssignSubject.getMaLop(), student.getHocKi(), student.getYear());

if (MarkSubject.isExistsMarkSubject(currentAcct.getID(), currentAssignSubject.getMaLop(), infoAssign.getHocKi(), infoAssign.getYear()))

{

MarkSubject mark = MarkSubject.getInfo(currentAssignSubject.getMaLop(), student.getID(), infoAssign.getHocKi(), infoAssign.getYear());

;if (!mark.updateMarkSubject(double.Parse(txtKTDK.Text), double.Parse(txtKTHP.Text), percentKTDK))

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Lưu điểm thất bại!");

}

else

{

clearText();

lvStudentLearn.SelectedItems.Clear();

}

}

else

{

MarkSubject mark = new MarkSubject(currentAcct.getID(), currentAssignSubject.getMaLop(), double.Parse(txtKTDK.Text), double.Parse(txtKTHP.Text), percentKTDK, infoAssign.getHocKi(), infoAssign.getYear());

if (mark.insertMark())

{

clearText();

lvStudentLearn.SelectedItems.Clear();

}

else

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Lưu điểm thất bại!");

}

}

}

* ***Lựa chọn Sinh viên thuộc danh sách sinh viên lớp cố vấn:***

private void lvMainClass\_SelectedIndexChanged(object sender, EventArgs e)

{

if (lvMainClass.SelectedIndices.Count != 0)

{

lvStudentLearn.SelectedIndices.Clear();

string ID = lvMainClass.Items[lvMainClass.SelectedIndices[0]].Text;

string nameAcct = (string) Account.getField("TenTK", "ID", ID);

currentAcct = new Account(nameAcct);

gbInfo.Enabled = true;

gbMark.Enabled = false;

Student student = (Student) Person.getInfo("ID", ID);

txtName.Text = student.getFullName().getFullName();

txtAddress.Text = student.getAddress();

txtEmail.Text = student.getEmail();

txtID.Text = ID;

txtPhone.Text = student.getPhone();

dtpBirth.Value = student.getBirth().getDate();

}

else

{

currentAcct = null;

gbInfo.Enabled = false;

clearText();

}

}

* ***Lựa chọn Sinh viên theo học các lớp đang dạy:***

private void lvStudentLearn\_SelectedIndexChanged(object sender, EventArgs e)

{

if (lvStudentLearn.SelectedIndices.Count != 0)

{

lvMainClass.SelectedIndices.Clear();

ListViewItem li = lvStudentLearn.SelectedItems[0];

string nameAcct = (string)Account.getField("TenTK", "ID", lvStudentLearn.Items[lvStudentLearn.SelectedIndices[0]].Text);

currentAcct = new Account(nameAcct);

Student student = (Student)Person.getInfo("ID", currentAcct.getID());

txtName.Text = student.getFullName().getFullName();

txtAddress.Text = student.getAddress();

txtEmail.Text = student.getEmail();

txtID.Text = student.getID();

txtPhone.Text = student.getPhone();

dtpBirth.Value = student.getBirth().getDate();

if (cbbStudent.SelectedIndex == 0)

{

gbInfo.Enabled = true;

}

else

{

gbInfo.Enabled = false;

}

gbMark.Enabled = true;

InfoAssignSubject infoAssign = InfoAssignSubject.getInfo(student.getID(), currentAssignSubject.getMaLop(), int.Parse(li.SubItems[2].Text), int.Parse(li.SubItems[3].Text));

MarkSubject mark = MarkSubject.getInfo(currentAssignSubject.getMaLop(), currentAcct.getID(), infoAssign.getHocKi(), infoAssign.getYear());

if (mark == null)

{

btnDeleteMark.Enabled = false;

txtKTDK.Clear();

txtKTHP.Clear();

}

else

{

if (infoAssign.getHocKi() != student.getHocKi() || infoAssign.getYear() != student.getYear())

{

btnDeleteMark.Enabled = false;

}

else

{

btnDeleteMark.Enabled = true;

}

int percent = mark.getPercentKTDK();

cbbPercentKTDK.SelectedIndex = percent / 10 - 1;

txtKTDK.Text = mark.getKTDK().ToString();

txtKTHP.Text = mark.getKTHP().ToString();

}

if (infoAssign.getHocKi() != student.getHocKi() || infoAssign.getYear() != student.getYear())

{

btnMark.Enabled = false;

}

else

{

btnMark.Enabled = true;

}

}

else

{

clearText();

btnDeleteMark.Enabled = false;

btnKickStudent.Enabled = false;

currentAcct = null;

gbInfo.Enabled = false;

gbMark.Enabled = false;

}

}

* ***Click loại sinh viên khỏi lớp:***

private void btnKickStudent\_Click(object sender, EventArgs e)

{

if (MessageInfo.makeMessage("Question", "Hỏi lại cho chắc", "Bạn có chắc chắn muốn loại sinh viên khỏi lớp?") != DialogResult.Yes)

{

return;

}

for (int i = 0; i < lvMainClass.CheckedIndices.Count; i++)

{

InfoAssignStudent.kickStudent(lvMainClass.Items[lvMainClass.CheckedIndices[i]].SubItems[0].Text);

}

for (int i = 0; i < lvStudentLearn.CheckedIndices.Count; i++)

{

ListViewItem li = lvStudentLearn.Items[lvStudentLearn.CheckedIndices[i]];

InfoAssignSubject infoAssignSubject = InfoAssignSubject.getInfo(li.SubItems[0].Text, currentAssignSubject.getMaLop(), int.Parse(li.SubItems[2].Text), int.Parse(li.SubItems[3].Text));

infoAssignSubject.deleteAssignSubject();

}

loadMainClassStudents();

loadClassTeach();

}

* ***Click Xoá điểm:***

private void btnDeleteMark\_Click(object sender, EventArgs e)

{

ListViewItem li = lvStudentLearn.SelectedItems[0];

if (MessageInfo.makeMessage("Question", "Hỏi lại cho chắc", "Bạn có chắc chắn muốn xoá điểm của sinh viên " + txtID.Text + " - HK" + li.SubItems[2].Text + " - Năm " + li.SubItems[3].Text + " - Mã lớp: " + currentAssignSubject.getMaLop() +"?") != DialogResult.Yes)

{

return;

}

InfoAssignSubject infoAssignSubject = InfoAssignSubject.getInfo(txtID.Text, currentAssignSubject.getMaLop(), int.Parse(li.SubItems[2].Text), int.Parse(li.SubItems[3].Text));

if (!MarkSubject.deleteMarkSubject(txtID.Text, currentAssignSubject.getMaLop(), infoAssignSubject.getHocKi(), infoAssignSubject.getYear()))

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Xoá điểm sinh viên thất bại!");

}

else

{

loadClassTeach();

}

}

* ***Check / Uncheck Sinh viên thuộc danh sách lớp đang cố vấn:***

private void lvMainClass\_ItemChecked(object sender, ItemCheckedEventArgs e)

{

if (lvMainClass.CheckedIndices.Count != 0)

{

btnKickStudent.Enabled = true;

}

else if (lvStudentLearn.CheckedIndices.Count != 0)

{

if (cbbStudent.SelectedIndex != 0)

{

btnKickStudent.Enabled = false;

}

else

{

btnKickStudent.Enabled = true;

}

}

else

{

btnKickStudent.Enabled = false;

}

}

* ***Check / Uncheck Sinh viên thuộc danh sách lớp đang dạy:***

private void lvStudentLearn\_ItemChecked(object sender, ItemCheckedEventArgs e)

{

if (lvMainClass.CheckedIndices.Count != 0)

{

btnKickStudent.Enabled = true;

}

else if (lvStudentLearn.CheckedIndices.Count != 0)

{

if (cbbStudent.SelectedIndex != 0)

{

btnKickStudent.Enabled = false;

}

else

{

btnKickStudent.Enabled = true;

}

}

else

{

btnKickStudent.Enabled = false;

}

}

* ***Hiện thực các phương thức khác:***
* ***Khởi tạo Form:***

public frmManageStudent(string nameAcct)

{

InitializeComponent();

int percent = 10;

while(percent != 100)

{

cbbPercentKTDK.Items.Add(percent + "%");

cbbPercentKTHP.Items.Add(percent + "%");

percent += 10;

}

cbbPercentKTDK.SelectedIndex = 4;

txtPhone.KeyPress += General.number\_KeyPress;

teacher = (Teacher) Account.getInfoByName(nameAcct);

loadMainClassStudents();

loadClassTeach();

}

* ***Load danh sách sinh viên lớp Giảng viên cố vấn:***

void loadMainClassStudents()

{

lvMainClass.Items.Clear();

List<Student> students = teacher.getMyMainClass();

if (students == null)

{

return;

}

for (int i =0; i<students.Count; i++)

{

ListViewItem li = new ListViewItem()

{

Text = students[i].getID(),

};

li.SubItems.Add(new ListViewItem.ListViewSubItem() { Text = students[i].getFullName().getFullName() });

li.SubItems.Add(new ListViewItem.ListViewSubItem() { Text = students[i].getBirth().getStrDate('-')});

lvMainClass.Items.Add(li);

}

}

* ***Tạo khối lớp đang dạy:***

GroupBox getGroupInfoAssign(InfoAssignTeacher infoAssign)

{

GroupBox gb = new GroupBox()

{

Size = new Size(500, 150),

Cursor = Cursors.Hand,

Tag = infoAssign,

BackColor = Color.LavenderBlush,

};

int xstart = 5, yStart = 10;

Label lblMaLop = new Label()

{

Text = "Mã lớp: " + infoAssign.getMaLop(),

ForeColor = Color.Teal,

Location = new Point(xstart, yStart),

AutoSize = true,

};

gb.Controls.Add(lblMaLop);

yStart += 27;

Label lblNameSubject = new Label()

{

Text = "Tên môn: " + infoAssign.getNameSubject(),

ForeColor = Color.Teal,

Location = new Point(xstart, yStart),

AutoSize = true,

};

gb.Controls.Add(lblNameSubject);

yStart += 27;

List<Student> students = InfoAssignSubject.getStudentsAssignSubject(infoAssign.getMaLop());

Label lblCount = new Label()

{

Text = "Số lượng sinh viên: " + students.Count,

ForeColor = Color.Teal,

Location = new Point(xstart, yStart),

AutoSize = true,

};

gb.Controls.Add(lblCount);

yStart += 27;

gb.Click += selectInfoAssign;

return gb;

}

* ***Load danh sách lớp đang dạy:***

void loadClassTeach()

{

fpnClassTeach.Controls.Clear();

List<InfoAssignTeacher> infoAssigns = teacher.getInfoAssign();

for (int i = 0; i < infoAssigns.Count; i++)

{

fpnClassTeach.Controls.Add(getGroupInfoAssign(infoAssigns[i]));

}

currentAssignSubject = null;

lvStudentLearn.Items.Clear();

clearText();

}

* ***Nhập điểm:***

private void inputValueMark(object sender, EventArgs e)

{

if (string.IsNullOrEmpty(txtKTDK.Text) || string.IsNullOrEmpty(txtKTHP.Text))

{

lblSum.Text = "Tổng điểm: 0";

return;

}

double ptA = double.Parse(txtKTDK.Text);

double ptB = double.Parse(txtKTHP.Text);

double result = ptA \* (cbbPercentKTDK.SelectedIndex + 1) / 10 + ptB \* (cbbPercentKTHP.SelectedIndex + 1) / 10;

lblSum.Text = "Tổng điểm: " + result;

}

* ***Dọn dẹp TextBox:***

void clearText()

{

txtID.Clear();

txtName.Clear();

dtpBirth.Value = DateTime.Now;

txtPhone.Clear();

txtEmail.Clear();

txtAddress.Clear();

txtKTDK.Clear();

txtKTHP.Clear();

cbbPercentKTDK.SelectedIndex = 4;

}

### ***3.1.15. Màn hình (Form) Xem môn học:***

* **Mô tả màn hình:** Form xem môn học thuộc Khoa đối với Giảng viên, thuộc Ngành đối với Sinh viên. Các môn học hiển thị trên nhiều GroupBox với mỗi GroupBox là 1 học kì và ComboBox chọn năm học để xem các năm khác.
* ***Hiện thực các phương thức xử lý sự kiện (Event Handlers):***
* ***Click chọn ngành để xem môn học:***

private void selectMajor(object sender, EventArgs e)

{

currentMajor = Major.getInfo(((Control)sender).Text);

currentYear = 1;

loadSubjects(1, true);

if (cbbYear.Items.Count != 0)

{

cbbYear.SelectedIndex = 0;

}

else

{

currentYear = 0;

}

}

* ***Lựa chọn năm học để xem môn:***

private void cbbYear\_SelectedIndexChanged(object sender, EventArgs e)

{

if (currentYear == cbbYear.SelectedIndex + 1)

{

return;

}

currentYear = cbbYear.SelectedIndex + 1;

loadSubjects(cbbYear.SelectedIndex + 1);

}

* ***Hiện thực các phương thức khác:***
* ***Khởi tạo Form:***

public frmWatchSubject(string nameAcct)

{

InitializeComponent();

string ID = (string)Account.getField("ID", "TenTK", nameAcct);

if (Teacher.isExistTeacherID(ID))

{

Teacher teacher = (Teacher)Person.getInfo("ID", ID);

Text = "Xem các môn học thuộc khoa " + teacher.getTenKhoa();

gbMajor.Text = "Danh sách ngành thuộc khoa " + teacher.getTenKhoa();

loadMajor(teacher);

}

else

{

Student student = (Student)Person.getInfo("ID", ID);

Text = "Xem các môn học thuộc ngành " + student.getNameMajor();

gbMajor.Text = "Ngành học của sinh viên";

currentMajor = Major.getInfo(student.getNameMajor());

Label lblNameMajor = new Label()

{

Text = currentMajor.getName(),

Font = new Font("Times New Roman", 10, FontStyle.Bold),

Size = new Size(375, 40),

ForeColor = Color.Navy,

BorderStyle = BorderStyle.FixedSingle,

BackColor = Color.DarkSeaGreen,

Cursor = Cursors.Hand,

TextAlign = ContentAlignment.MiddleLeft,

};

fpnMajor.Controls.Add(lblNameMajor);

currentYear = 1;

loadSubjects(1, true);

if (cbbYear.Items.Count != 0)

{

cbbYear.SelectedIndex = 0;

}

else

{

currentYear = 0;

}

}

}

* ***Load danh sách Ngành:***

void loadMajor(Teacher teacher)

{

Khoa khoa = new Khoa(teacher.getTenKhoa());

List<Major> majors = khoa.getMyMajor();

for (int i = 0; i < majors.Count; i++)

{

Label lblNameMajor = new Label()

{

Text = majors[i].getName(),

Font = new Font("Times New Roman", 10, FontStyle.Bold),

Size = new Size(375, 40),

ForeColor = Color.Navy,

BorderStyle = BorderStyle.FixedSingle,

Cursor = Cursors.Hand,

TextAlign = ContentAlignment.MiddleLeft,

BackColor =Color.DarkSeaGreen,

};

lblNameMajor.Click += selectMajor;

fpnMajor.Controls.Add(lblNameMajor);

}

}

* ***Load danh sách môn học:***

void loadSubjects(int year, bool createCbb = false)

{

fpnSubject.Controls.Clear();

subjects = currentMajor.getMySubjects();

if (createCbb)

{

int max = 0;

for (int i =0; i < subjects.Count; i++)

{

if (subjects[i].getYear() > max)

{

max = subjects[i].getYear();

}

}

cbbYear.Items.Clear();

for (int i = 1; i <= max; i++)

{

cbbYear.Items.Add("Năm " + i);

}

}

for (int i =0; i < subjects.Count; i++)

{

if (subjects[i].getYear() != year)

{

subjects.RemoveAt(i);

}

}

List<GroupBox> gbHocKi = new List<GroupBox>();

List<FlowLayoutPanel> fpnHocKi = new List<FlowLayoutPanel>();

for (int i = 0; i < subjects.Count; i++)

{

int index = subjects[i].getHocKi() - 1;

if (gbHocKi.Count <= index)

{

GroupBox gb = new GroupBox()

{

Text = "Học kì " + (index + 1),

Size = new Size(540, 200),

BackColor = Color.White,

Font= new Font("Times New Roman", 10, FontStyle.Bold),

ForeColor = Color.Crimson,

};

FlowLayoutPanel fpn = new FlowLayoutPanel()

{

Location = new Point(10, 30),

Size = new Size(520, 150),

AutoScroll = true,

};

gb.Controls.Add(fpn);

fpnSubject.Controls.Add(gb);

fpnHocKi.Add(fpn);

gbHocKi.Add(gb);

}

fpnHocKi[index].Controls.Add(new Label() { Text = subjects[i].getName(),BackColor =Color.Yellow, Font = new Font("Times New Roman", 12, FontStyle.Bold), ForeColor = Color.Navy, TextAlign = ContentAlignment.MiddleLeft, BorderStyle = BorderStyle.FixedSingle, Size = new Size(fpnHocKi[index].Width - 20, 40) });

}

}

### ***3.1.16. Màn hình (Form) Xem khảo sát:***

* **Mô tả màn hình:** Xem các bài khảo sát sinh viên gửi cho mình qua từng lớp học. ComboBox lọc theo tên môn học, lọc mã lớp, lựa chọn xem khảo sát hiện tại hay cũ hơn. Danh sách khảo sát hiển thị với mỗi bài là 1 GroupBox, click vào bài khảo sát sẽ xem được nội dung khảo sát
* ***Hiện thực các phương thức xử lý sự kiện (Event Handlers):***
* ***Lựa chọn tên môn học để lọc danh sách lớp:***

private void cbbNameSubject\_SelectedIndexChanged(object sender, EventArgs e)

{

cbbMaLop.Items.Clear();

cbbMaLop.Items.Add("Hiển thị toàn bộ");

for (int i = 0; i < infoAssignTeachers.Count; i++)

{

if (cbbNameSubject.SelectedIndex != 0 && infoAssignTeachers[i].getNameSubject() != cbbNameSubject.Text)

{

infoAssignTeachers.RemoveAt(i);

i--;

continue;

}

cbbMaLop.Items.Add(infoAssignTeachers[i].getMaLop());

}

cbbMaLop.SelectedIndex = 0;

}

* ***Lựa chọn lọc danh sách lớp theo mã lớp:***

private void cbbMaLop\_SelectedIndexChanged(object sender, EventArgs e)

{

cbbSurvey.SelectedIndex = -1;

cbbSurvey.SelectedIndex = 0;

}

* ***Lựa chọn xem khảo sát hiện tại / cũ:***

private void cbbSurvey\_SelectedIndexChanged(object sender, EventArgs e)

{

if (cbbSurvey.SelectedIndex == -1)

{

return;

}

fpnSurvey.Controls.Clear();

List<Survey> lstSurvey = Teacher.getAllSurvey(teacher.getID());

for (int i = 0; i < lstSurvey.Count; i++)

{

InfoAssignTeacher infoAssignTeacher = InfoAssignTeacher.getInfo(lstSurvey[i].getMaLop());

if (cbbNameSubject.SelectedIndex != 0 && infoAssignTeacher.getNameSubject() != cbbNameSubject.Text)

{

continue;

}

if (cbbMaLop.SelectedIndex != 0 && lstSurvey[i].getMaLop().ToString() != cbbMaLop.Text)

{

continue;

}

Student student = (Student)Person.getInfo("ID", lstSurvey[i].getMaSV());

if (cbbSurvey.SelectedIndex == 0)

{

if (lstSurvey[i].getHocKi() != student.getHocKi() || lstSurvey[i].getYear() != student.getYear())

{

continue;

}

}

else

{

if (lstSurvey[i].getHocKi() == student.getHocKi() && lstSurvey[i].getYear() == student.getYear())

{

continue;

}

}

fpnSurvey.Controls.Add(getGroupSurvey(lstSurvey[i]));

}

clearSurvey();

}

* ***Click chọn xem khảo sát:***

private void selectSurvey(object sender, EventArgs e)

{

Survey survey = (Survey)((Control)sender).Tag;

if (survey == null)

{

return;

}

Student student = (Student)Person.getInfo("ID", survey.getMaSV());

lblNameStudent.Text = student.getFullName().getFullName();

object img = Account.getField("AnhDaiDien", "ID", student.getID());

if (img != null)

{

ptrAvt.Image = Avatar.ConvertBytesToImage((byte[])img);

}

else

{

ptrAvt.Image = Image.FromFile("Assets/Imgs/No Image.png");

}

int[] content = survey.getContent();

foreach (Control pn in gbSurvey.Controls)

{

if (pn is Panel)

{

int index = int.Parse(pn.Tag.ToString()) - 1;

foreach (Control rdo in pn.Controls)

{

if (rdo is RadioButton)

{

int value = int.Parse(rdo.Tag.ToString());

if (value == content[index])

{

RadioButton radio = (RadioButton)rdo;

radio.Checked = true;

break;

}

}

}

}

}

}

* ***Hiện thực các phương thức khác:***
* ***Khởi tạo Form:***

public frmWatchSurvey(string nameAcct)

{

InitializeComponent();

string ID = (string)Account.getField("ID", "TenTK", nameAcct);

teacher = (Teacher)Person.getInfo("ID", ID);

infoAssignTeachers = teacher.getInfoAssign();

cbbNameSubject.Items.Clear();

cbbNameSubject.Items.Add("Hiển thị toàn bộ");

for (int i = 0; i < infoAssignTeachers.Count; i++)

{

bool isOk = true;

for (int j = 0; j < cbbNameSubject.Items.Count; j++)

{

if (cbbNameSubject.Items[j].ToString() == infoAssignTeachers[i].getNameSubject())

{

isOk = false;

}

}

if (isOk)

{

cbbNameSubject.Items.Add(infoAssignTeachers[i].getNameSubject());

}

}

cbbNameSubject.SelectedIndex = 0;

}

public frmWatchSurvey()

{

InitializeComponent();

}

* ***Dọn dẹp khảo sát:***

void clearSurvey()

{

ptrAvt.Image = Image.FromFile("Assets/Imgs/No Image.png");

lblNameStudent.Text = "Họ và tên";

foreach (Control pn in gbSurvey.Controls)

{

if (pn is Panel)

{

foreach (Control rdo in pn.Controls)

{

if (rdo is RadioButton)

{

int value = int.Parse(rdo.Tag.ToString());

if (value == 1)

{

RadioButton radio = (RadioButton)rdo;

radio.Checked = true;

break;

}

}

}

}

}

}

* ***Tạo khối khảo sát:***

GroupBox getGroupSurvey(Survey survey)

{

InfoAssignTeacher infoAssignTeacher = InfoAssignTeacher.getInfo(survey.getMaLop());

GroupBox gb = new GroupBox()

{

Size = new Size(500, 150),

Font = new Font("Times New Roman", 10, FontStyle.Bold),

Cursor = Cursors.Hand,

Tag = survey,

BackColor = Color.LavenderBlush,

};

int xstart = 5, yStart = 10;

Label lblMaLop = new Label()

{

Text = "Mã lớp: " + survey.getMaLop(),

ForeColor = Color.Teal,

Location = new Point(xstart, yStart),

AutoSize = true,

};

gb.Controls.Add(lblMaLop);

yStart += 27;

Label lblNameSubject = new Label()

{

Text = "Tên môn: " + infoAssignTeacher.getNameSubject(),

ForeColor = Color.Teal,

Location = new Point(xstart, yStart),

AutoSize = true,

};

gb.Controls.Add(lblNameSubject);

yStart += 27;

Label lblMaSV = new Label()

{

Text = "Mã SV: " + survey.getMaSV(),

ForeColor = Color.Teal,

Location = new Point(xstart, yStart),

AutoSize = true,

};

gb.Controls.Add(lblMaSV);

yStart += 27;

Label lblHocKi = new Label()

{

Text = "Học kì: " + survey.getHocKi(),

ForeColor = Color.Teal,

Location = new Point(xstart, yStart),

AutoSize = true,

};

gb.Controls.Add(lblHocKi);

yStart += 27;

Label lblYear = new Label()

{

Text = "Năm: " + survey.getYear(),

ForeColor = Color.Teal,

Location = new Point(xstart, yStart),

AutoSize = true,

};

gb.Controls.Add(lblYear);

gb.Click += selectSurvey;

return gb;

}

### ***3.1.17. Màn hình (Form) Đăng ký môn:***

* **Mô tả màn hình:** Form cho đăng ký môn học, hiển thị danh sách môn học theo tiến trình, danh sách môn học lại và danh sách môn đã đăng ký. ComboBox xem đăng ký môn cũ hơn hay hiện tại, nút in phiếu đăng ký để in phiếu dưới dạng .txt.
* ***Hiện thực các phương thức xử lý sự kiện (Event Handlers):***
* ***Lựa chọn năm học:***

private void cbbYear\_SelectedIndexChanged(object sender, EventArgs e)

{

cbbHocKi.Items.Clear();

int maxHocki;

if (cbbYear.SelectedIndex + 1 == student.getYear())

{

maxHocki = student.getHocKi();

}

else

{

maxHocki = 2;

}

for (int i = 1; i <= maxHocki; i++)

{

cbbHocKi.Items.Add("Học Kì " + i);

}

cbbHocKi.SelectedIndex = cbbHocKi.Items.Count - 1;

}

* ***Lựa chọn học kì:***

private void cbbHocKi\_SelectedIndexChanged(object sender, EventArgs e)

{

if (cbbYear.SelectedIndex == -1)

{

return;

}

clearInput();

loadSubjectAssigned();

}

* ***Click nút xem lịch học:***

private void selectDate(object sender, EventArgs e)

{

int maLop = (int)((Control)sender).Tag;

frmDateAssign frm = new frmDateAssign(maLop);

frm.ShowDialog();

}

* ***Click vào lớp học theo tiến trình / học lại:***

private void selectAssignTeacher(object sender, EventArgs e)

{

InfoAssignTeacher infoAssignTeacher = (InfoAssignTeacher)((Control)sender).Tag;

if (infoAssignTeacher.getMaxCount() == infoAssignTeacher.getStudents().Count())

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Sỉ số đã đạt tối đa!!");

return;

}

currentInfoAssignTeacher = infoAssignTeacher;

if (cbbHocKi.SelectedIndex + 1 != student.getHocKi() || cbbYear.SelectedIndex + 1 != student.getYear())

{

btnAssignSubject.Enabled = false;

}

else

{

btnAssignSubject.Enabled = true;

}

lblMaLop.Text = "Mã lớp: " + currentInfoAssignTeacher.getMaLop().ToString();

lblNameSubject.Text = "Tên môn: " + currentInfoAssignTeacher.getNameSubject();

lblMaGV.Text = "Mã GV: " + currentInfoAssignTeacher.getMaGV();

lblCurrentCount.Text = "Sỉ số hiện tại: " + InfoAssignSubject.getStudentsAssignSubject(currentInfoAssignTeacher.getMaLop()).Count() + "/" + currentInfoAssignTeacher.getMaxCount();

btnCancelAssignSubject.Enabled = false;

}

* ***Click vào lớp đã đăng ký:***

private void selectAssignedTeacher(object sender, EventArgs e)

{

InfoAssignTeacher infoAssignTeacher = (InfoAssignTeacher)((Control)sender).Tag;

InfoAssignSubject infoAssignSubject = InfoAssignSubject.getInfo(student.getID(), infoAssignTeacher.getMaLop(), cbbHocKi.SelectedIndex + 1, cbbYear.SelectedIndex + 1);

if (infoAssignSubject.getYear() != cbbYear.SelectedIndex + 1 || infoAssignSubject.getHocKi() != cbbHocKi.SelectedIndex + 1)

{

return;

}

currentInfoAssignTeacher = infoAssignTeacher;

if (cbbHocKi.SelectedIndex + 1 != student.getHocKi() || cbbYear.SelectedIndex + 1 != student.getYear())

{

btnCancelAssignSubject.Enabled = false;

}

else

{

btnCancelAssignSubject.Enabled = true;

}

lblMaLop.Text = "Mã lớp: " + currentInfoAssignTeacher.getMaLop().ToString();

lblNameSubject.Text = "Tên môn: " + currentInfoAssignTeacher.getNameSubject();

lblMaGV.Text = "Mã GV: " + currentInfoAssignTeacher.getMaGV();

lblCurrentCount.Text = "Sỉ số hiện tại: " + InfoAssignSubject.getStudentsAssignSubject(currentInfoAssignTeacher.getMaLop()).Count() + "/" + currentInfoAssignTeacher.getMaxCount();

btnAssignSubject.Enabled = false;

}

* ***Click Đăng ký môn:***

private void btnAssignSubject\_Click(object sender, EventArgs e)

{

if (currentInfoAssignTeacher == null)

{

return;

}

if (InfoAssignSubject.isSameDateAssignSubject(student.getID(), currentInfoAssignTeacher.getMaLop()))

{

MessageInfo.makeMessage("Question", "Hỏi lại cho chắc", "Lịch học bị trùng!");

return;

}

InfoAssignTeacher CurrentInfoAssignTeacher = InfoAssignTeacher.getInfo(currentInfoAssignTeacher.getMaLop());

List<InfoAssignSubject> infoAssignSubjects = InfoAssignSubject.getAllAssignSubject();

for (int i = 0; i < infoAssignSubjects.Count; i++)

{

if (!InfoAssignSubject.isExistsAssignSubject(student.getID(), infoAssignSubjects[i].getMaLop(), cbbHocKi.SelectedIndex + 1, cbbYear.SelectedIndex + 1))

{

continue;

}

InfoAssignTeacher infoAssignTeacher = InfoAssignTeacher.getInfo(infoAssignSubjects[i].getMaLop());

if (infoAssignTeacher.getNameSubject() == CurrentInfoAssignTeacher.getNameSubject())

{

if (MessageInfo.makeMessage("Question", "Hỏi lại cho chắc", "Bạn đã đăng ký môn " + currentInfoAssignTeacher.getNameSubject() + ", Bạn có chắc muốn thay thế lớp mới không?") != DialogResult.Yes)

{

return;

}

infoAssignSubjects[i].deleteAssignSubject();

break;

}

}

InfoAssignSubject info = new InfoAssignSubject(student.getID(), currentInfoAssignTeacher.getMaLop(), cbbHocKi.SelectedIndex + 1, cbbYear.SelectedIndex + 1);

info.addAssignSubject();

loadSubject();

loadRelearn();

loadSubjectAssigned();

}

* ***Click Huỷ đăng ký môn:***

private void btnCancelAssignSubject\_Click(object sender, EventArgs e)

{

InfoAssignSubject info = InfoAssignSubject.getInfo(student.getID(), currentInfoAssignTeacher.getMaLop(), cbbHocKi.SelectedIndex + 1, cbbYear.SelectedIndex + 1);

if (!info.deleteAssignSubject())

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Xoá đăng ký thất bại!");

return;

}

loadSubject();

loadRelearn();

loadSubjectAssigned();

btnCancelAssignSubject.Enabled = false;

}

* ***Click nút In phiếu đăng ký:***

private void btnPrint\_Click(object sender, EventArgs e)

{

if (infoAssignedSubjects.Count == 0)

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Bạn chưa đăng ký môn nào để in phiếu!");

return;

}

List<string> lstContent = new List<string>();

string nameFile = "Phiếu đăng ký môn học HK" + (cbbHocKi.SelectedIndex + 1) + " - Năm " + (cbbYear.SelectedIndex + 1) + " - " + student.getFullName().getFullName() + " - " + student.getID();

lstContent.Add(getTextAlignCenter(nameFile, 76));

lstContent.Add("----------------------------------------------------------------------------");

if (File.Exists(nameFile + ".txt"))

{

int i = 1;

while (File.Exists(nameFile + " (" + i + ").txt"))

{

i++;

}

nameFile += " (" + i + ").txt";

}

else

{

nameFile += ".txt";

}

lstContent.Add("| Mã lớp | Tên môn | Mã GV |");

lstContent.Add("----------------------------------------------------------------------------");

for (int j = 0; j < infoAssignedSubjects.Count; j++)

{

InfoAssignTeacher infoAssignTeacher = InfoAssignTeacher.getInfo(infoAssignedSubjects[j].getMaLop());

string line = "|" + getTextAlignCenter(infoAssignTeacher.getMaLop().ToString(), 12) + "|";

line += getTextAlignCenter(infoAssignTeacher.getNameSubject(), 49) + "|";

line += getTextAlignCenter(infoAssignTeacher.getMaGV(), 11) + "|";

lstContent.Add(line);

lstContent.Add("----------------------------------------------------------------------------");

}

File.WriteAllLines(nameFile, lstContent);

MessageInfo.makeMessage("Information", "Chúc mừng", "In phiếu thành công!");

}

* ***Hiện thực các phương thức khác:***
* ***Khởi tạo Form:***

public frmEnrollSubject(string nameAcct)

{

InitializeComponent();

string ID = (string)Account.getField("ID", "TenTK", nameAcct);

student = (Student) Person.getInfo("ID", ID);

cbbYear.Items.Clear();

for (int i = 1; i <= student.getYear(); i++)

{

cbbYear.Items.Add("Năm " + i);

}

cbbYear.SelectedIndex = cbbYear.Items.Count - 1;

loadSubject();

loadRelearn();

loadSubjectAssigned();

}

* ***Load danh sách lớp học học lại:***

void loadRelearn()

{

subjectClassReLearn.Clear();

string nameMajor = student.getNameMajor();

Major major = Major.getInfo(nameMajor);

List<Subject> subjects = major.getMySubjects();

for (int i = 0; i < subjects.Count; i++)

{

List<InfoAssignTeacher> infoAssignTeachers = Subject.getAssignTeachers(subjects[i].getName());

if (subjects[i].getYear() == student.getYear())

{

if(subjects[i].getHocKi() < student.getHocKi())

{

foreach (InfoAssignTeacher infoAssign in infoAssignTeachers)

{

if (InfoAssignRoom.isExistsAssignRoom(infoAssign.getMaLop()))

{

bool isOk = true;

List<InfoAssignSubject> infoAssignSubjects = InfoAssignSubject.getAssignSameSubject(student.getID(), infoAssign.getNameSubject());

for (int j = 0; j < infoAssignSubjects.Count; j++)

{

if (MarkSubject.isExistsMarkSubject(student.getID(), infoAssign.getMaLop(), infoAssignSubjects[j].getHocKi(), infoAssignSubjects[j].getYear()))

{

if (MarkSubject.isFail(student.getID(), infoAssign.getMaLop(), infoAssignSubjects[j].getHocKi(), infoAssignSubjects[j].getYear()))

{

isOk = false;

break;

}

}

}

if (!isOk)

{

subjectClassReLearn.Add(infoAssign);

}

}

}

}

}

else if (subjects[i].getYear() < student.getYear())

{

foreach (InfoAssignTeacher infoAssign in infoAssignTeachers)

{

if (InfoAssignRoom.isExistsAssignRoom(infoAssign.getMaLop()))

{

bool isOk = false;

List<InfoAssignSubject> infoAssignSubjects = InfoAssignSubject.getAssignSameSubject(student.getID(), infoAssign.getNameSubject());

for (int j = 0; j < infoAssignSubjects.Count; j++)

{

if (MarkSubject.isExistsMarkSubject(student.getID(), infoAssign.getMaLop(), infoAssignSubjects[j].getHocKi(), infoAssignSubjects[j].getYear()))

{

if (!MarkSubject.isFail(student.getID(), infoAssign.getMaLop(), infoAssignSubjects[j].getHocKi(), infoAssignSubjects[j].getYear()))

{

isOk = true;

break;

}

}

}

if (!isOk)

{

subjectClassReLearn.Add(infoAssign);

}

}

}

}

}

fpnSubjectReLearn.Controls.Clear();

for (int i = 0; i < subjectClassReLearn.Count; i++)

{

GroupBox gb = getGroupClassSubject(subjectClassReLearn[i]);

gb.Click += selectAssignTeacher;

fpnSubjectReLearn.Controls.Add(gb);

}

}

* ***Load danh sách lớp đã đăng ký:***

void loadSubjectAssigned()

{

fpnSubjectAssigned.Controls.Clear();

infoAssignedSubjects = student.getInfoAssignSubject();

for (int i = 0; i < infoAssignedSubjects.Count; i++)

{

if (infoAssignedSubjects[i].getHocKi() != cbbHocKi.SelectedIndex + 1 || infoAssignedSubjects[i].getYear() != cbbYear.SelectedIndex + 1)

{

infoAssignedSubjects.RemoveAt(i);

i--;

continue;

}

GroupBox gb = getGroupClassSubject(InfoAssignTeacher.getInfo(infoAssignedSubjects[i].getMaLop()));

gb.Click += selectAssignedTeacher;

fpnSubjectAssigned.Controls.Add(gb);

}

}

* ***Load danh sách lớp theo tiến trình:***

void loadSubject()

{

clearInput();

btnAssignSubject.Enabled = false;

subjectClassMain.Clear();

string nameMajor = student.getNameMajor();

Major major = Major.getInfo(nameMajor);

List<Subject> subjects = major.getMySubjects();

for (int i = 0; i < subjects.Count; i++)

{

List<InfoAssignTeacher> infoAssignTeachers = Subject.getAssignTeachers(subjects[i].getName());

if (subjects[i].getHocKi() != student.getHocKi() || subjects[i].getYear() != student.getYear())

{

continue;

}

foreach (InfoAssignTeacher infoAssign in infoAssignTeachers)

{

if (InfoAssignRoom.isExistsAssignRoom(infoAssign.getMaLop()))

{

if (!InfoAssignSubject.isExistsAssignSubject(student.getID(), infoAssign.getMaLop(), student.getHocKi(), student.getYear()))

{

subjectClassMain.Add(infoAssign);

}

}

}

}

fpnSubject.Controls.Clear();

for (int i = 0; i < subjectClassMain.Count; i++)

{

GroupBox gb = getGroupClassSubject(subjectClassMain[i]);

gb.Click += selectAssignTeacher;

fpnSubject.Controls.Add(gb);

}

}

* ***Tạo khối lớp:***

GroupBox getGroupClassSubject(InfoAssignTeacher classSubject)

{

GroupBox gb = new GroupBox()

{

Text = "Mã lớp: " + classSubject.getMaLop(),

Size = new Size(483, 210),

Cursor = Cursors.Hand,

Tag = classSubject,

BackColor = Color.LavenderBlush,

};

int xstart = 5, yStart = 30;

Label lblNameSubject = new Label()

{

Text = "Tên môn: " + classSubject.getNameSubject(),

ForeColor = Color.Teal,

Location = new Point(xstart, yStart),

Size = new Size(gb.Width - 10, 20),

};

gb.Controls.Add(lblNameSubject);

yStart += 27;

Label lblMaGV = new Label()

{

Text = "Mã GV: " + classSubject.getMaGV(),

ForeColor = Color.Teal,

Location = new Point(xstart, yStart),

Size = new Size(gb.Width - 10, 20),

};

gb.Controls.Add(lblMaGV);

yStart += 27;

Label lblSiSoHienTai = new Label()

{

Text = "Sỉ số hiện tại: " + classSubject.getStudents().Count() + "/" + classSubject.getMaxCount(),

ForeColor = Color.Teal,

Location = new Point(xstart, yStart),

Size = new Size(gb.Width - 10, 20),

};

gb.Controls.Add(lblSiSoHienTai);

List< InfoAssignRoom> infoAssignRooms = InfoAssignRoom.getInfo(classSubject.getMaLop());

yStart += 27;

Button btnDate = new Button()

{

Text = "Xem lịch học",

Font = new Font("Times New Roman", 10),

Cursor = Cursors.Hand,

Tag = classSubject.getMaLop(),

Location = new Point(xstart, yStart),

Size = new Size(120, 40),

ForeColor = Color.Navy,

};

gb.Controls.Add(btnDate);

btnDate.Click += selectDate;

return gb;

}

* ***Dọn dẹp dữ liệu:***

void clearInput()

{

btnAssignSubject.Enabled = false;

lblMaLop.Text = "Mã Lớp";

lblNameSubject.Text = "Tên môn";

lblMaGV.Text = "Mã GV";

lblCurrentCount.Text = "Sỉ số hiện tại";

}

* ***Căn lề văn bản:***

string getTextAlignCenter(string text, int countMaxChar)

{

int odd = countMaxChar - text.Length;

if (odd <= 0)

{

return text;

}

for (int i = 0; i < odd / 2; i++)

{

text = " " + text;

}

if (odd % 2 == 0)

{

for (int i = 0; i < odd / 2; i++)

{

text = text + " ";

}

}

else

{

for (int i = 0; i < odd / 2 + 1; i++)

{

text = text + " ";

}

}

return text;

}

### ***3.1.18. Màn hình (Form) Khảo sát Giảng viên:***

* **Mô tả màn hình:**Gửi khảo sát cho Giảng viên đã dạy lớp học Sinh viên đã học. ComboBox năm học và học kì để lọc lớp học để khảo sát, năm học và học kì đã qua sẽ chỉ có thể xem lại chứ không thể sửa nó. Combobox lọc lớp theo mã Giảng viên. Danh sách lớp hiển thị với mỗi lớp bằng 1 GroupBox, click vào lớp để thực hiện gửi khảo sát.
* ***Hiện thực các phương thức xử lý sự kiện (Event Handlers):***
* ***Click chọn lớp để làm khảo sát:***

private void selectAssignSubject(object sender, EventArgs e)

{

currentInfoAssignSubject = (InfoAssignSubject)((Control)sender).Tag;

MarkSubject mark = MarkSubject.getInfo(currentInfoAssignSubject.getMaLop(), student.getID(), currentInfoAssignSubject.getHocKi(), currentInfoAssignSubject.getYear());

if (mark == null || cbbYear.SelectedIndex + 1 != student.getYear() || cbbHocKi.SelectedIndex + 1 != student.getHocKi())

{

gbSurvey.Enabled = false;

clearSurvey();

return;

}

clearSurvey();

gbSurvey.Enabled = true;

Survey survey = Survey.getSurvey(student.getID(), currentInfoAssignSubject.getMaLop(), cbbHocKi.SelectedIndex + 1, cbbYear.SelectedIndex + 1);

if (survey == null)

{

return;

}

int[] content = survey.getContent();

foreach (Control pn in gbSurvey.Controls)

{

if (pn is Panel)

{

int index = int.Parse(pn.Tag.ToString()) - 1;

foreach (Control rdo in pn.Controls)

{

if (rdo is RadioButton)

{

int value = int.Parse(rdo.Tag.ToString());

if (value == content[index])

{

RadioButton radio = (RadioButton)rdo;

radio.Checked = true;

break;

}

}

}

}

}

}

* ***Click nút Xác nhận gửi khảo sát:***

private void btnAccept\_Click(object sender, EventArgs e)

{

if (currentInfoAssignSubject == null)

{

return;

}

int[] content = new int[5];

foreach (Control pn in gbSurvey.Controls)

{

if (pn is Panel)

{

int index = int.Parse(pn.Tag.ToString()) - 1;

foreach (Control rdo in pn.Controls)

{

if (rdo is RadioButton)

{

RadioButton radio = (RadioButton)rdo;

if (radio.Checked)

{

content[index] = int.Parse(rdo.Tag.ToString());

break;

}

}

}

}

}

Survey survey = Survey.getSurvey(student.getID(), currentInfoAssignSubject.getMaLop(), cbbHocKi.SelectedIndex + 1, cbbYear.SelectedIndex + 1);

if (survey == null)

{

survey = new Survey(student.getID(), currentInfoAssignSubject.getMaLop(), cbbHocKi.SelectedIndex + 1, cbbYear.SelectedIndex + 1, content);

if (survey.insertSurvey())

{

MessageInfo.makeMessage("Information", "Chúc mừng", "Đã gửi khảo sát!");

}

else

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Gửi khảo sát thất bại!");

}

}

else

{

if (survey.updateSurvey(content))

{

MessageInfo.makeMessage("Information", "Chúc mừng", "Cập nhật khảo sát thành công!");

}

else

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Cập nhật khảo sát thất bại!");

}

}

}

* ***Lựa chọn năm để lọc khảo sát:***

private void cbbYear\_SelectedIndexChanged(object sender, EventArgs e)

{

cbbHocKi.Items.Clear();

int maxHocki;

if (cbbYear.SelectedIndex + 1 == student.getYear())

{

maxHocki = student.getHocKi();

}

else

{

maxHocki = 2;

}

for (int i = 1; i <= maxHocki; i++)

{

cbbHocKi.Items.Add("Học kì " + i);

}

cbbHocKi.SelectedIndex = cbbHocKi.Items.Count - 1;

gbSurvey.Enabled = false;

clearSurvey();

}

* ***Lựa chọn học kì để lọc khảo sát:***

private void cbbHocKi\_SelectedIndexChanged(object sender, EventArgs e)

{

if (cbbYear.SelectedIndex == -1)

{

return;

}

lstInfoAssigns = student.getInfoAssignSubject();

cbbMaGV.Items.Clear();

cbbMaGV.Items.Add("Hiển thị toàn bộ");

for (int i = 0; i < lstInfoAssigns.Count; i++)

{

if (cbbYear.SelectedIndex + 1 != lstInfoAssigns[i].getYear())

{

continue;

}

InfoAssignTeacher infoAssignTeacher = InfoAssignTeacher.getInfo(lstInfoAssigns[i].getMaLop());

cbbMaGV.Items.Add(infoAssignTeacher.getMaGV());

}

cbbMaGV.SelectedIndex = 0;

gbSurvey.Enabled = false;

clearSurvey();

}

* ***Lựa chọn mã Giảng viên để lọc khảo sát:***

private void cbbMaGV\_SelectedIndexChanged(object sender, EventArgs e)

{

lstInfoAssigns = student.getInfoAssignSubject();

for (int i = 0; i < lstInfoAssigns.Count; i++)

{

InfoAssignTeacher infoAssignTeacher = InfoAssignTeacher.getInfo(lstInfoAssigns[i].getMaLop());

if (cbbYear.SelectedIndex + 1 != lstInfoAssigns[i].getYear() || cbbMaGV.SelectedIndex != 0 && infoAssignTeacher.getMaGV() != cbbMaGV.Text)

{

lstInfoAssigns.RemoveAt(i);

i--;

}

}

gbSurvey.Enabled = false;

clearSurvey();

loadInfoAssignSubject();

}

* ***Hiện thực các phương thức khác:***
* Khởi tạo Form:

public frmSurvey(string nameAcct)

{

InitializeComponent();

string MaSV = (string)Account.getField("ID", "TenTK", nameAcct);

student = (Student) Person.getInfo("ID", MaSV);

cbbYear.Items.Clear();

cbbYear.Items.Clear();

for (int i = 1; i <= student.getYear(); i++)

{

cbbYear.Items.Add("Năm " + i);

}

cbbYear.SelectedIndex = cbbYear.Items.Count - 1;

}

* Tạo khối lớp đã đang học:

GroupBox getGroupSubject(InfoAssignSubject infoAssignSubject)

{

MarkSubject mark = MarkSubject.getInfo(infoAssignSubject.getMaLop(), student.getID(), infoAssignSubject.getHocKi(), infoAssignSubject.getYear());

GroupBox gb = new GroupBox()

{

Size = new Size(580, 200),

Cursor = Cursors.Hand,

Tag = infoAssignSubject,

BackColor = Color.LavenderBlush,

};

int xstart = 5, yStart = 10;

Label lblMaLop = new Label()

{

Text = "Mã lớp: " + infoAssignSubject.getMaLop(),

ForeColor = Color.Teal,

Location = new Point(xstart, yStart),

AutoSize = true,

};

gb.Controls.Add(lblMaLop);

yStart += 27;

InfoAssignTeacher infoAssignTeacher = InfoAssignTeacher.getInfo(infoAssignSubject.getMaLop());

Label lblNameSub = new Label()

{

Text = "Tên môn: " + infoAssignTeacher.getNameSubject(),

ForeColor = Color.Teal,

Location = new Point(xstart, yStart),

AutoSize = true,

};

gb.Controls.Add(lblNameSub);

yStart += 27;

Label lblMaGV = new Label()

{

Text = "Mã lớp: " + infoAssignTeacher.getMaGV(),

ForeColor = Color.Teal,

Location = new Point(xstart, yStart),

AutoSize = true,

};

gb.Controls.Add(lblMaGV);

yStart += 27;

Label lblHocKi = new Label()

{

Text = "Học kì: " + infoAssignSubject.getHocKi(),

ForeColor = Color.Teal,

Location = new Point(xstart, yStart),

AutoSize = true,

};

gb.Controls.Add(lblHocKi);

yStart += 27;

Label lblNam = new Label()

{

Text = "Năm: " + infoAssignSubject.getYear(),

ForeColor = Color.Teal,

Location = new Point(xstart, yStart),

AutoSize = true,

};

gb.Controls.Add(lblNam);

string diemTB = "Chưa có";

if (mark != null)

{

diemTB = mark.caculateDTB().ToString();

}

yStart += 27;

Label lblDiemTB = new Label()

{

Text = "Điểm TB: " + diemTB,

ForeColor = Color.Teal,

Location = new Point(xstart, yStart),

AutoSize = true,

};

gb.Controls.Add(lblDiemTB);

gb.Click += selectAssignSubject;

return gb;

}

* Dọn dẹp khảo sát:

void clearSurvey()

{

foreach (Control pn in gbSurvey.Controls)

{

if (pn is Panel)

{

foreach (Control rdo in pn.Controls)

{

if (rdo is RadioButton)

{

int value = int.Parse(rdo.Tag.ToString());

if (value == 1)

{

RadioButton radio = (RadioButton)rdo;

radio.Checked = true;

break;

}

}

}

}

}

}

* Load danh sách lớp:

void loadInfoAssignSubject()

{

fpnInfoAssignSubjects.Controls.Clear();

for (int i = 0; i < lstInfoAssigns.Count; i++)

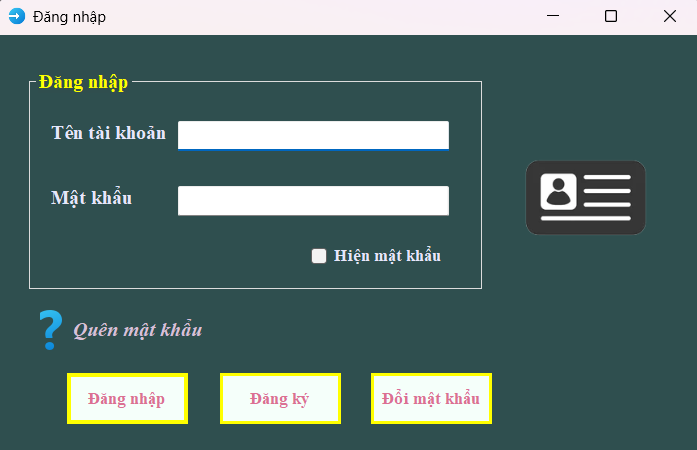
{

fpnInfoAssignSubjects.Controls.Add(getGroupSubject(lstInfoAssigns[i]));

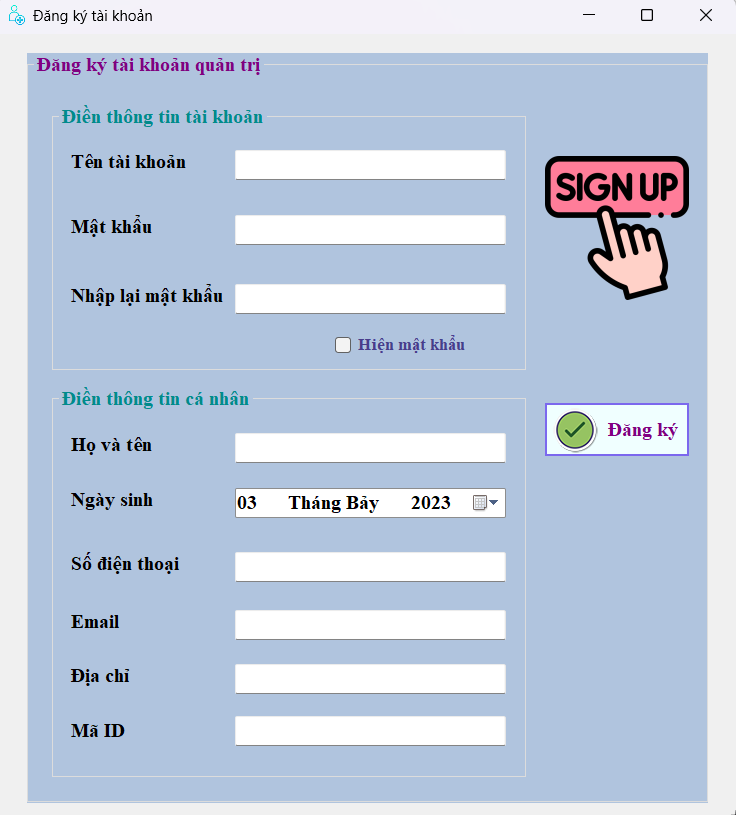
}

}

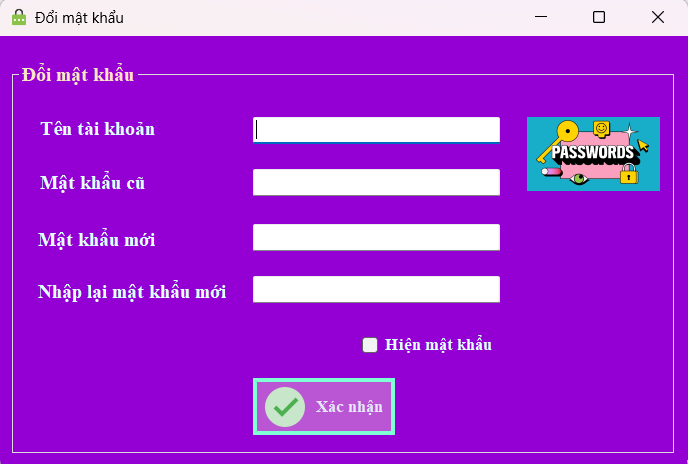
## **Kiểm thử ứng dụng:**



#### *Hình ảnh 3.2.1. Form Đăng nhập*



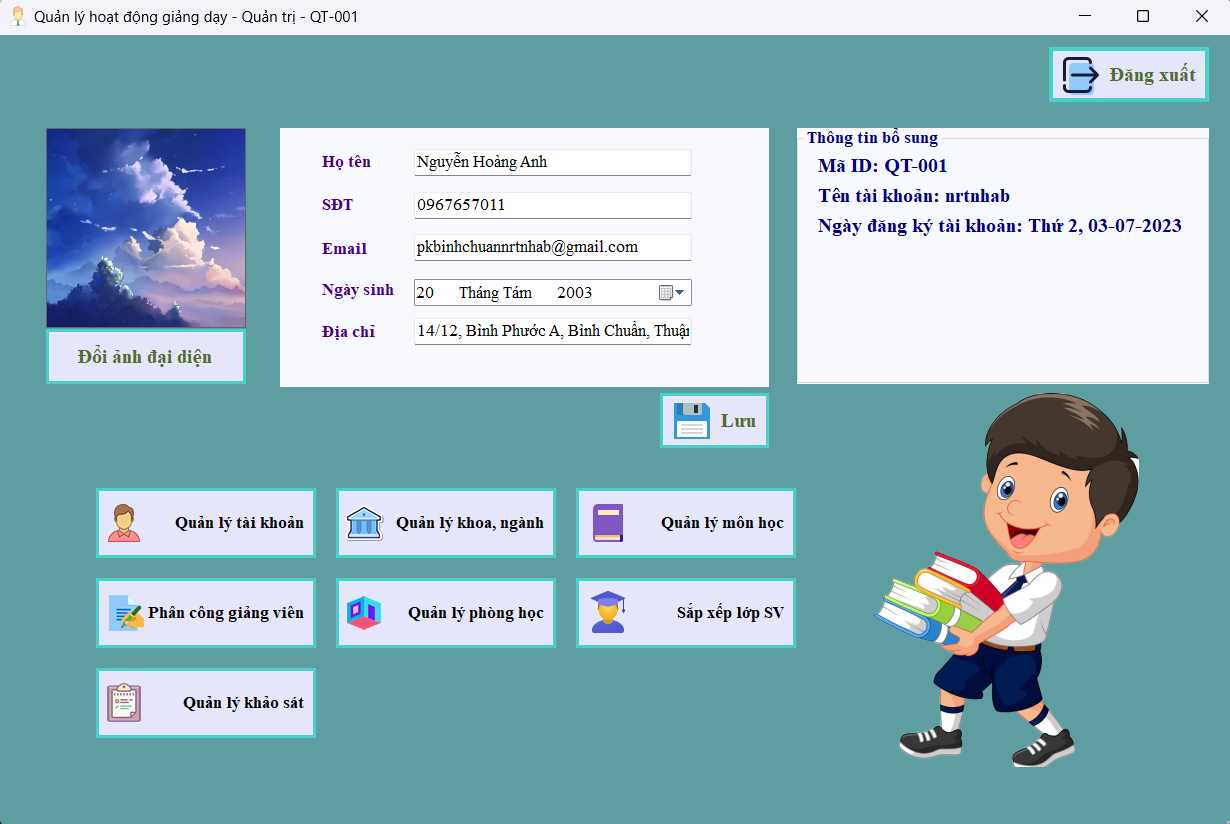
#### Hình ảnh 3.2.2. Form Đăng ký



#### Hình ảnh 3.2.3. Form Đổi mật khẩu



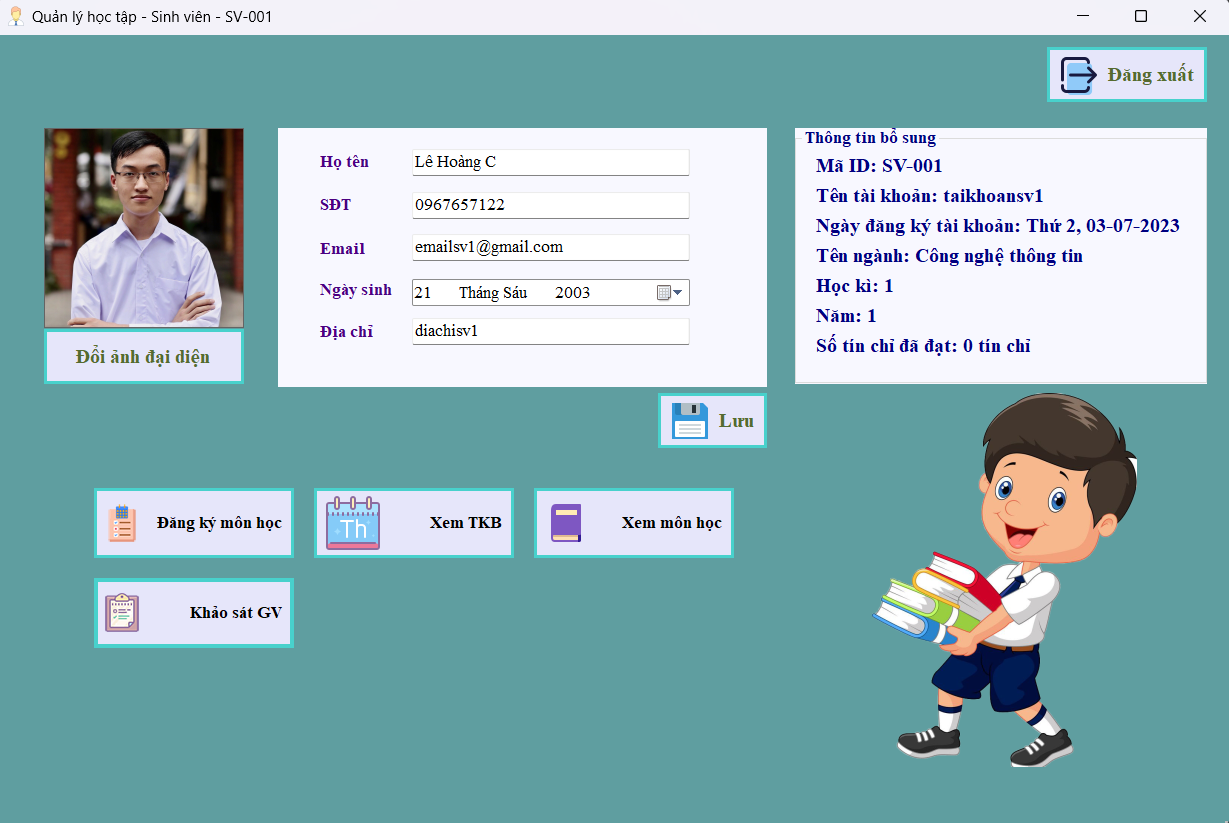
#### Hình ảnh 3.2.4. Form Quên mật khẩu



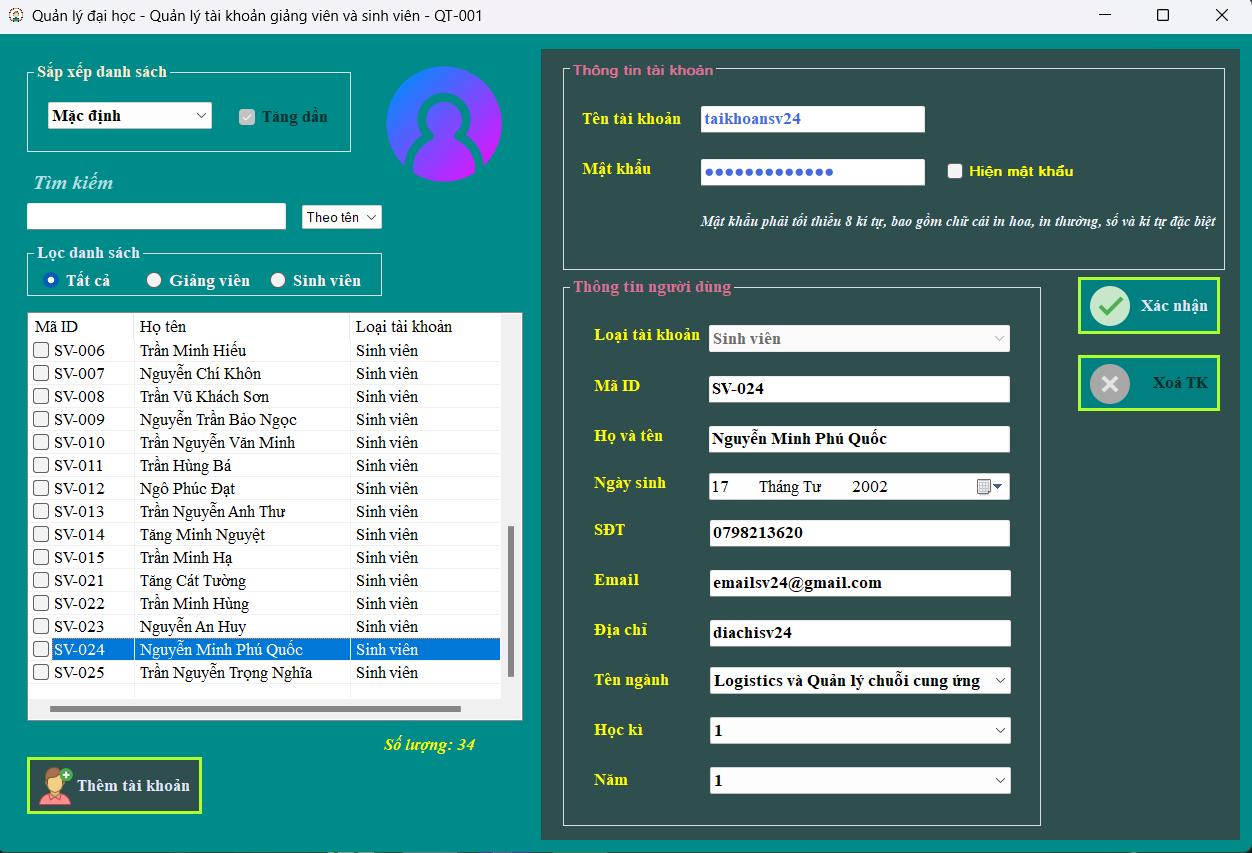
#### Hình ảnh 3.2.5. Form chính cho Quản trị



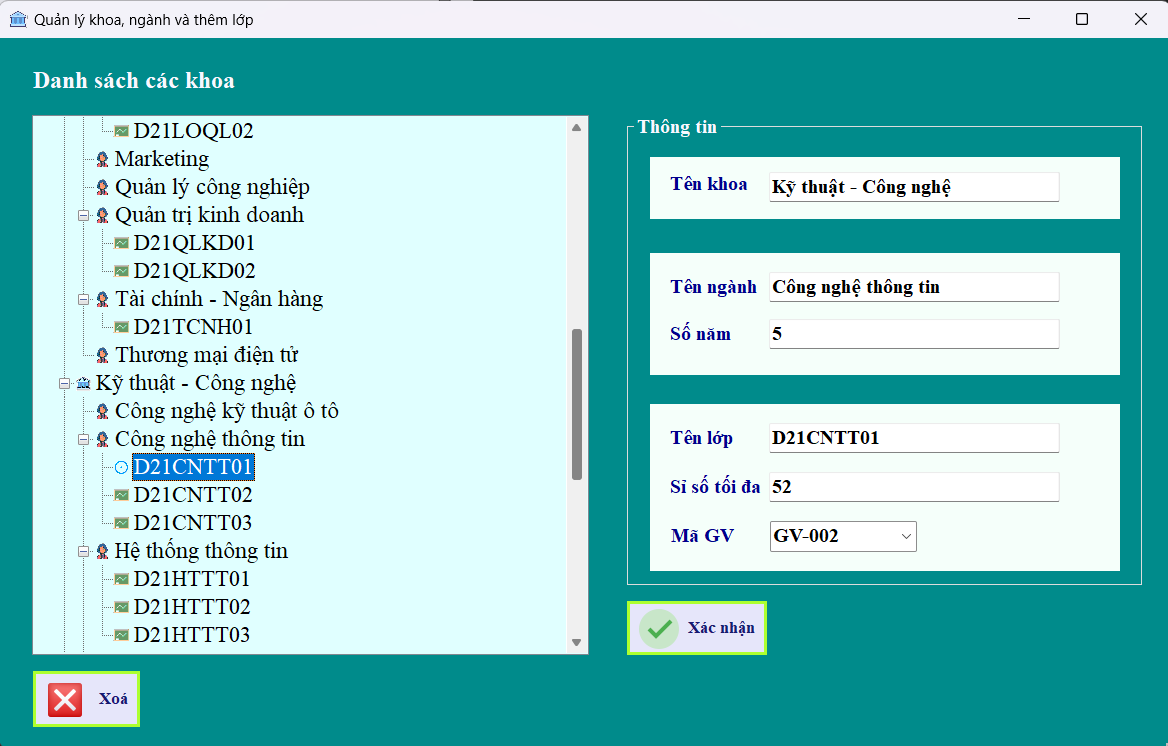
#### Hình ảnh 3.2.6. Form chính cho Giảng viên



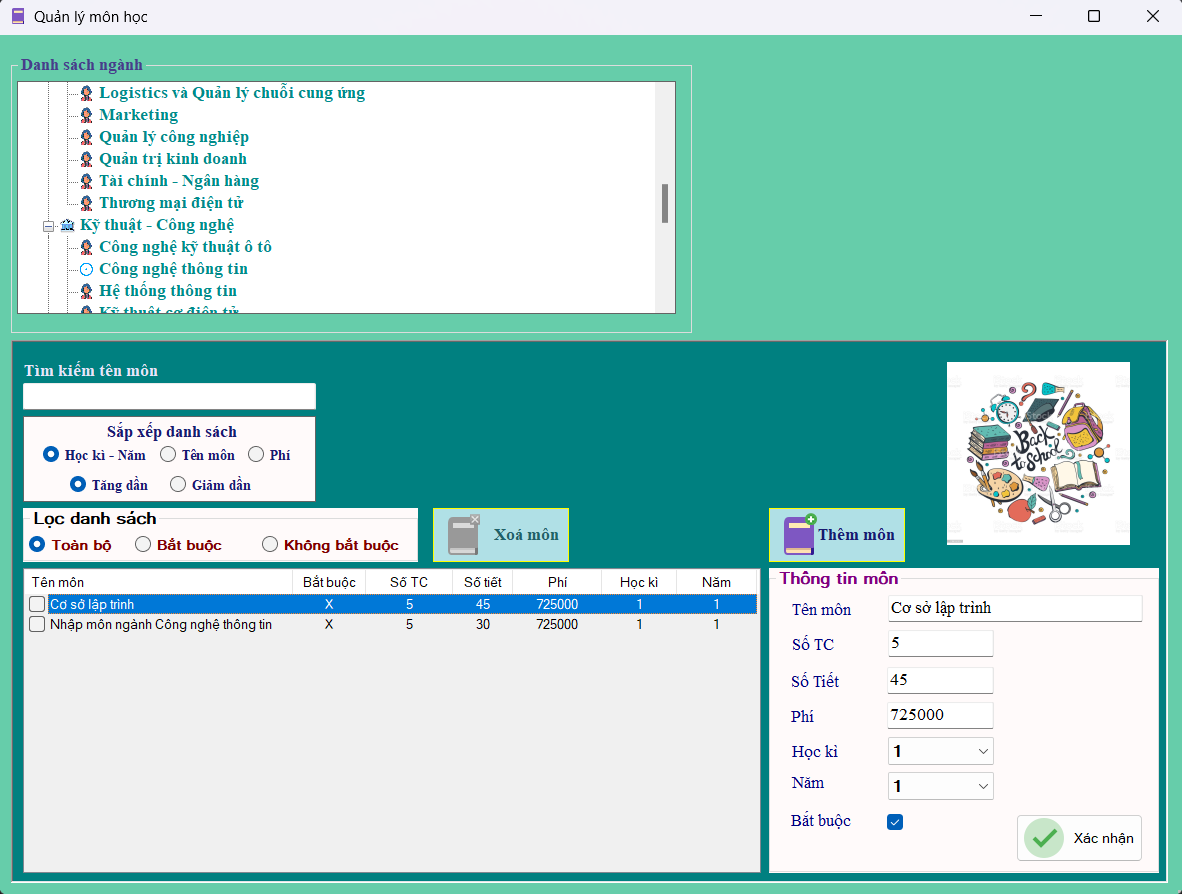
#### Hình ảnh 3.2.7. Form chính dành cho Sinh viên



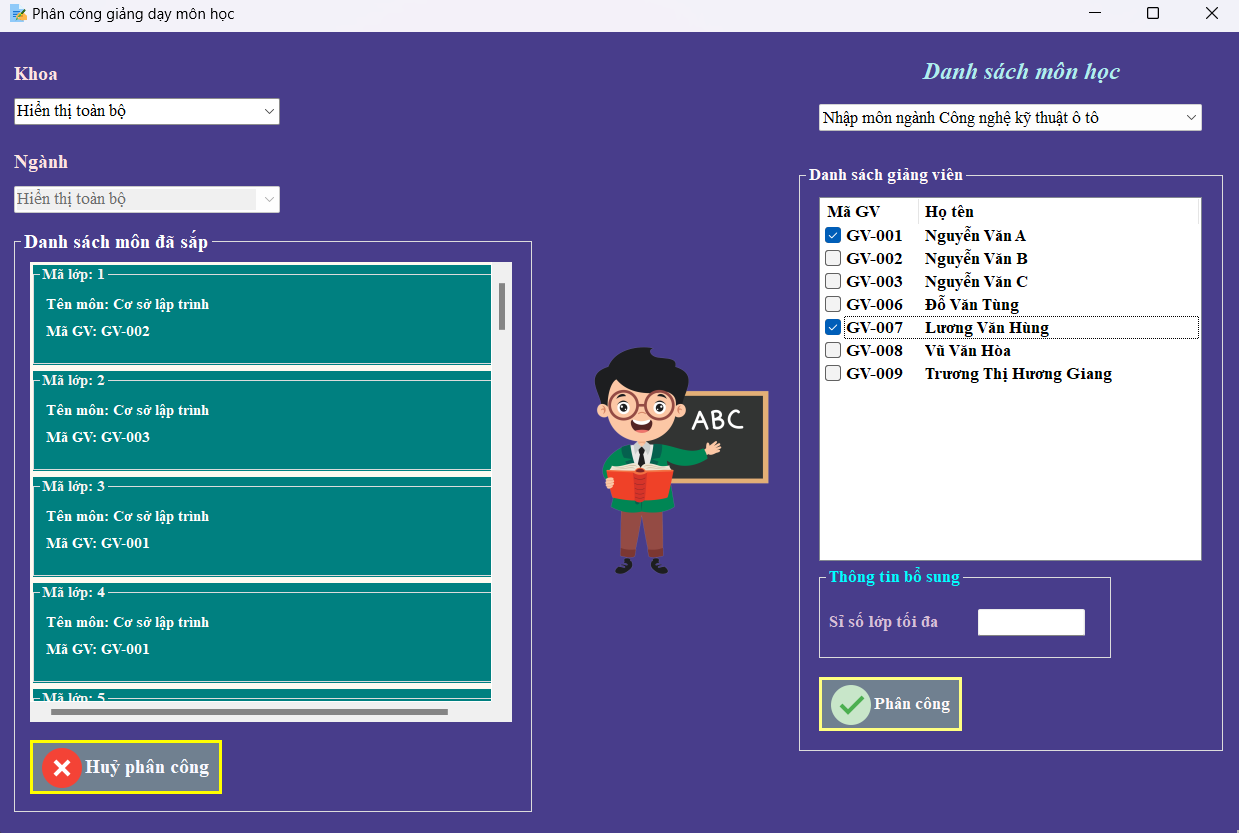
#### Hình ảnh 3.2.8. Form Quản lý tài khoản



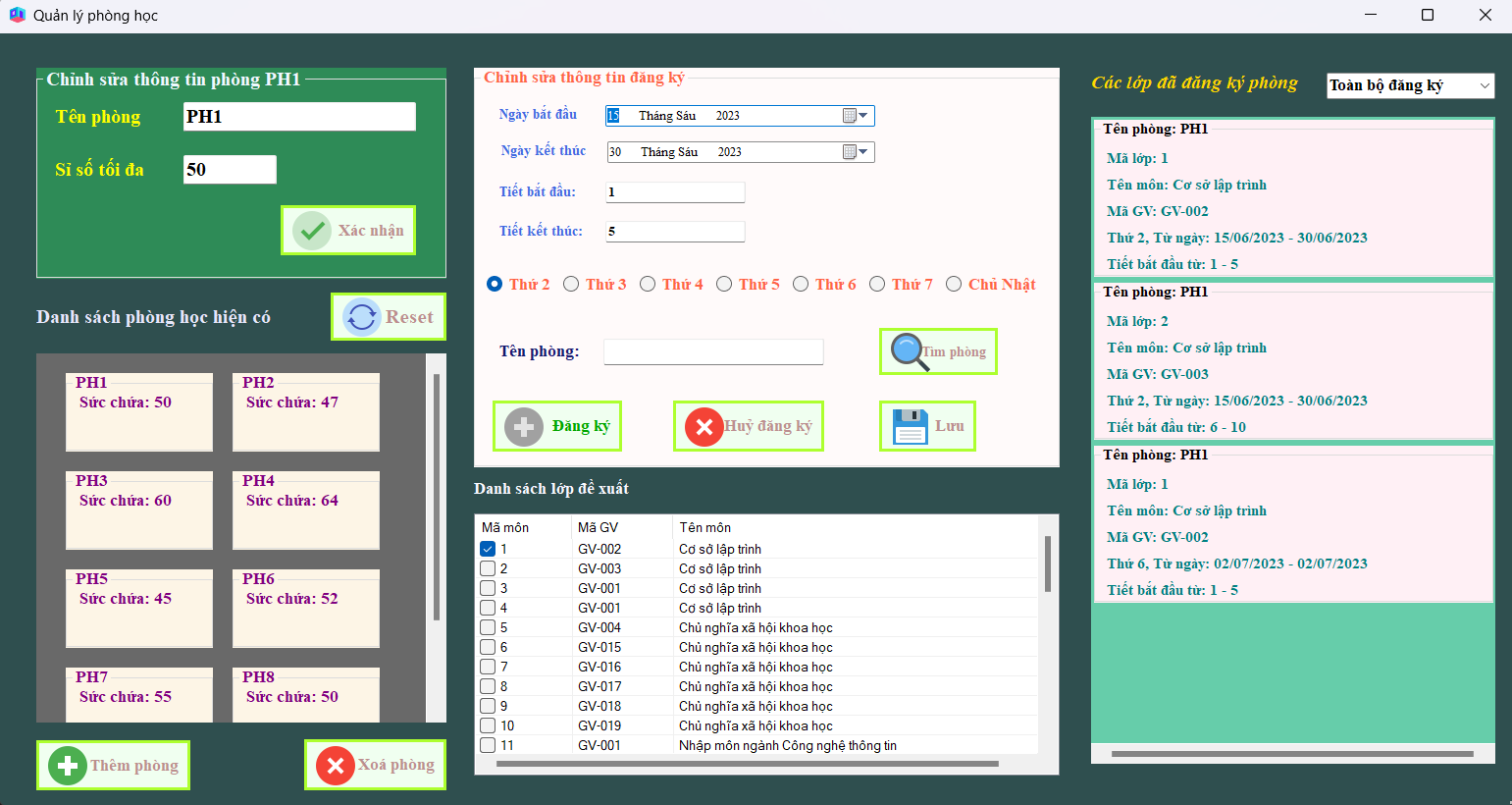
#### Hình ảnh 3.2.9. Form Quản lý khoa, ngành, thêm lớp



#### Hình ảnh 3.2.10. Form Quản lý môn học



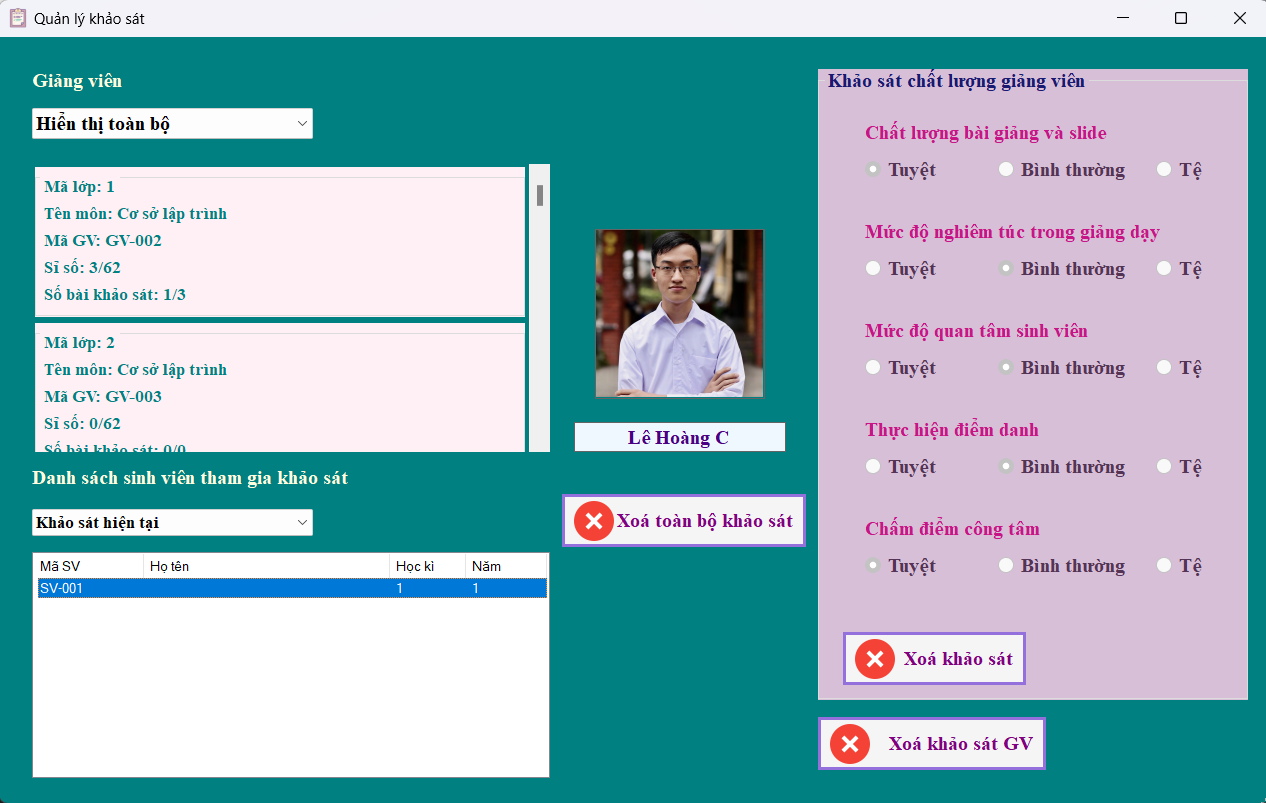
#### Hình ảnh 3.2.11. Form Phân công giảng dạy cho Giảng viên



#### Hình ảnh 3.2.12. Form Quản lý và đăng ký phòng



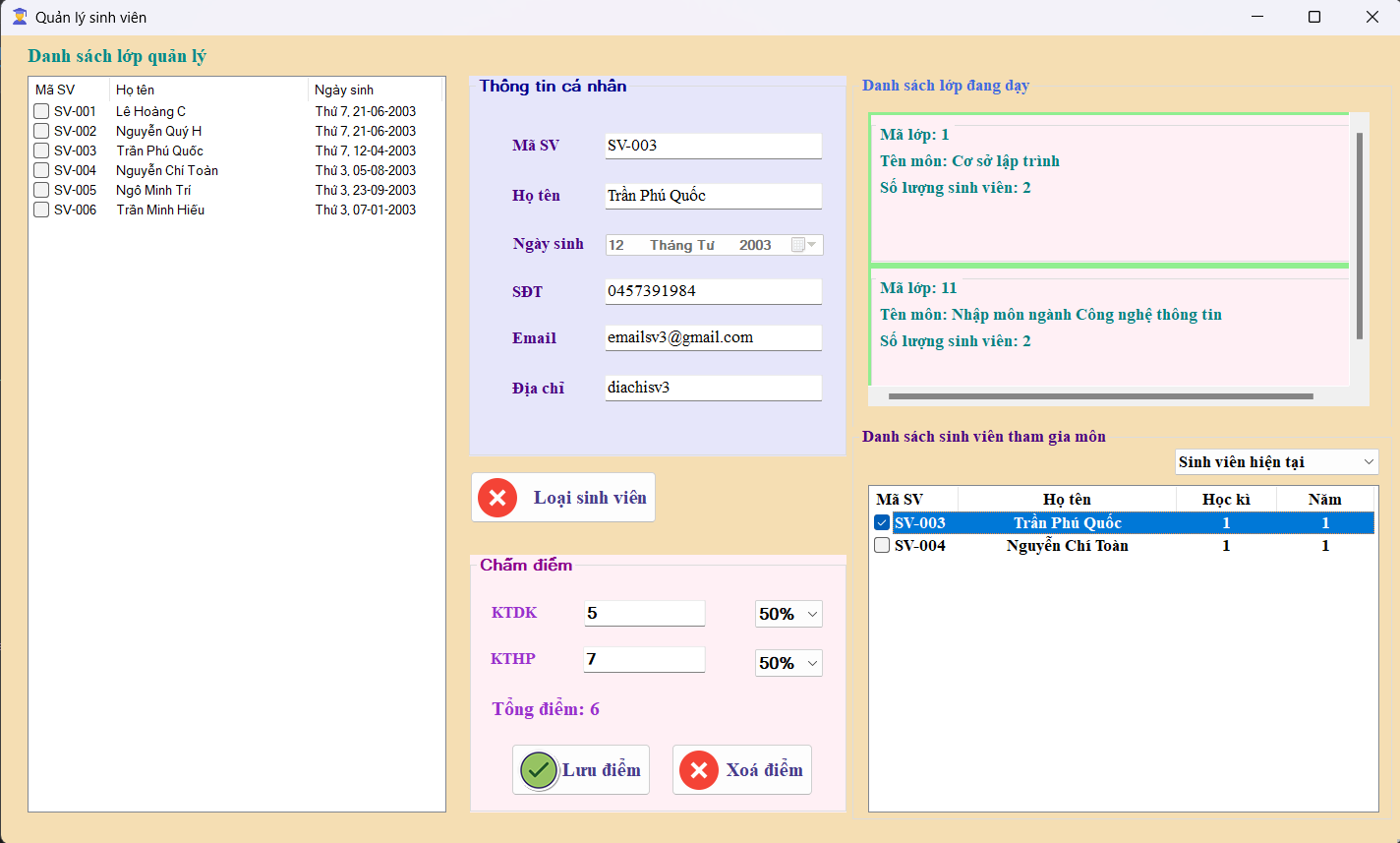
#### Hình ảnh 3.2.13. Form Sắp lớp cho Sinh viên



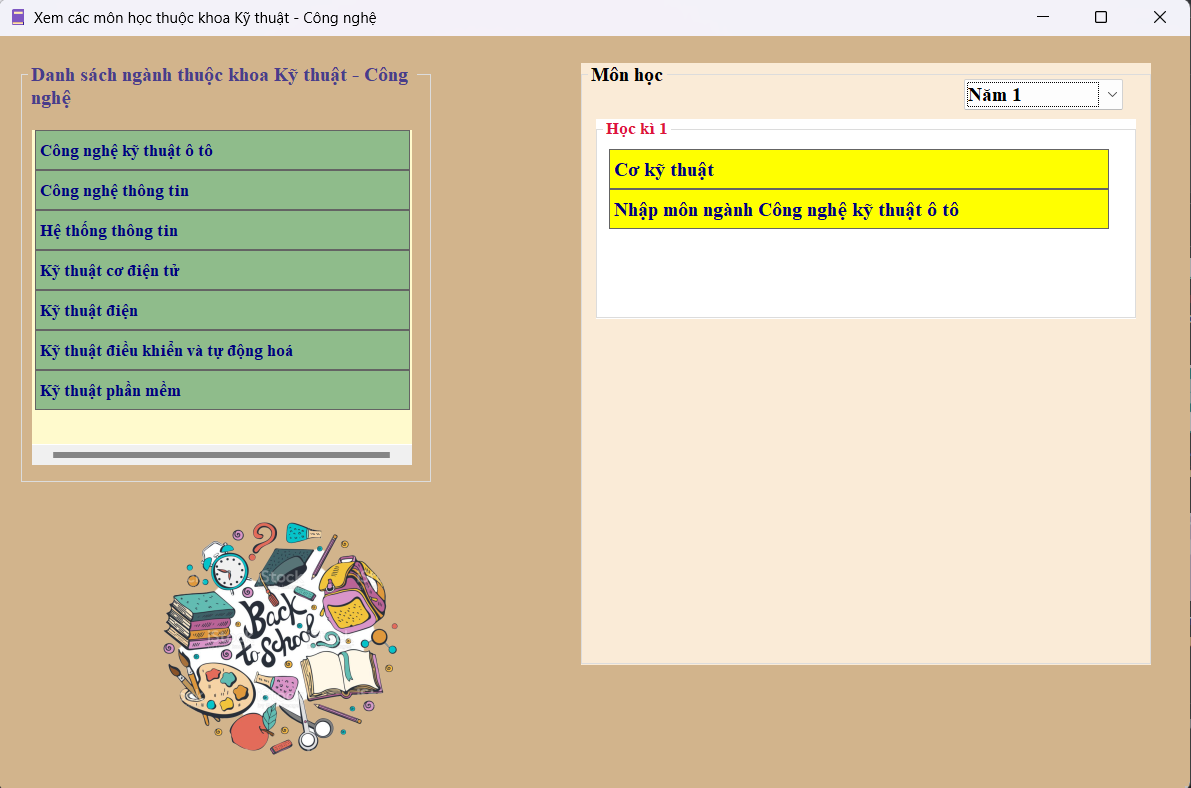
#### Hình ảnh 3.2.14. Form Quản lý khảo sát



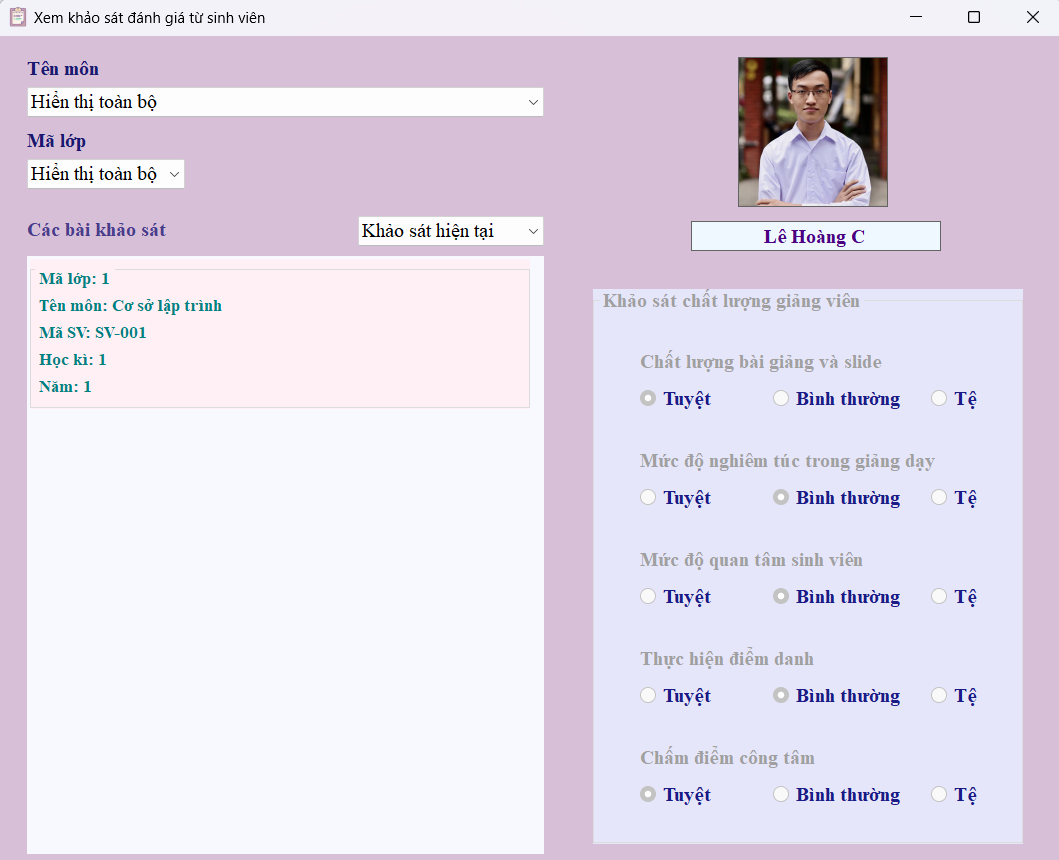
#### Hình ảnh 3.2.15. Form Xem lịch dạy học



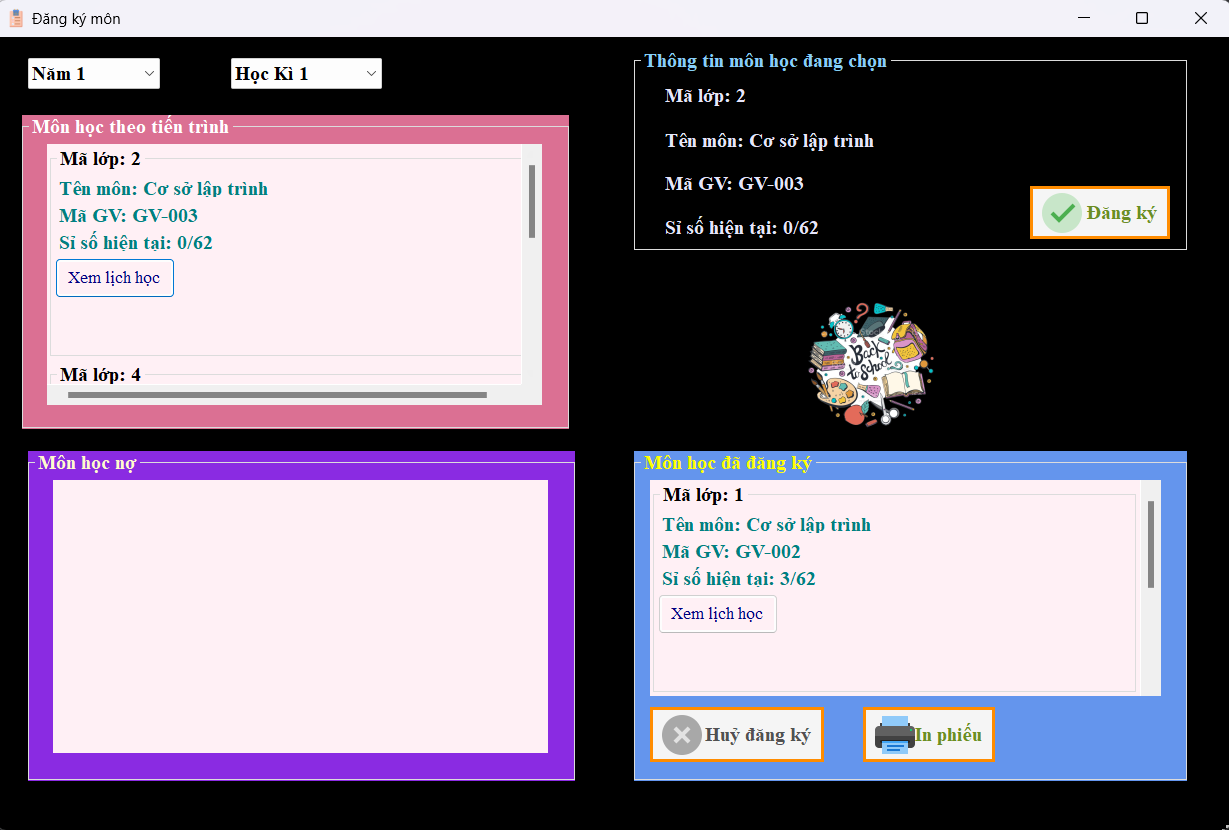
#### Hình ảnh 3.2.16. Form Quản lý / Chấm điểm Sinh viên



#### Hình ảnh 3.2.17. Form Xem môn học theo Khoa dành cho Giảng viên



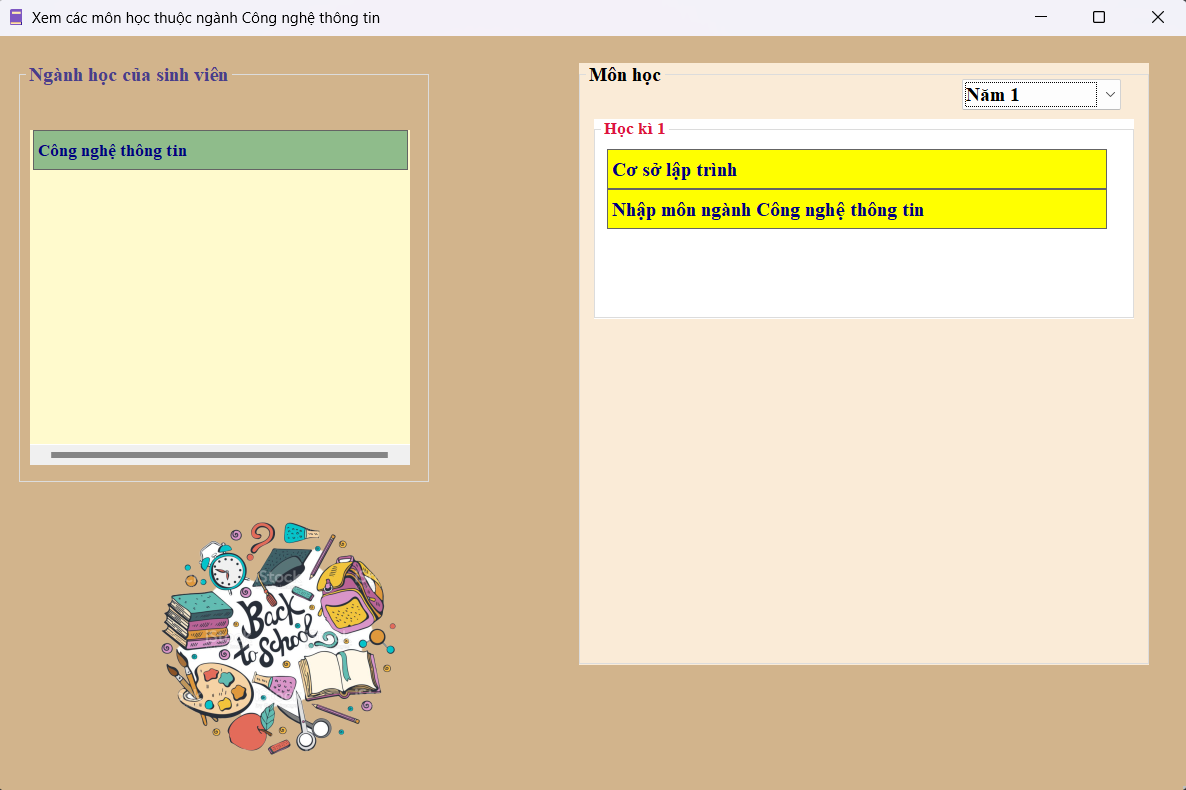
#### Hình ảnh 3.2.18. Form Xem khảo sát từ Sinh viên



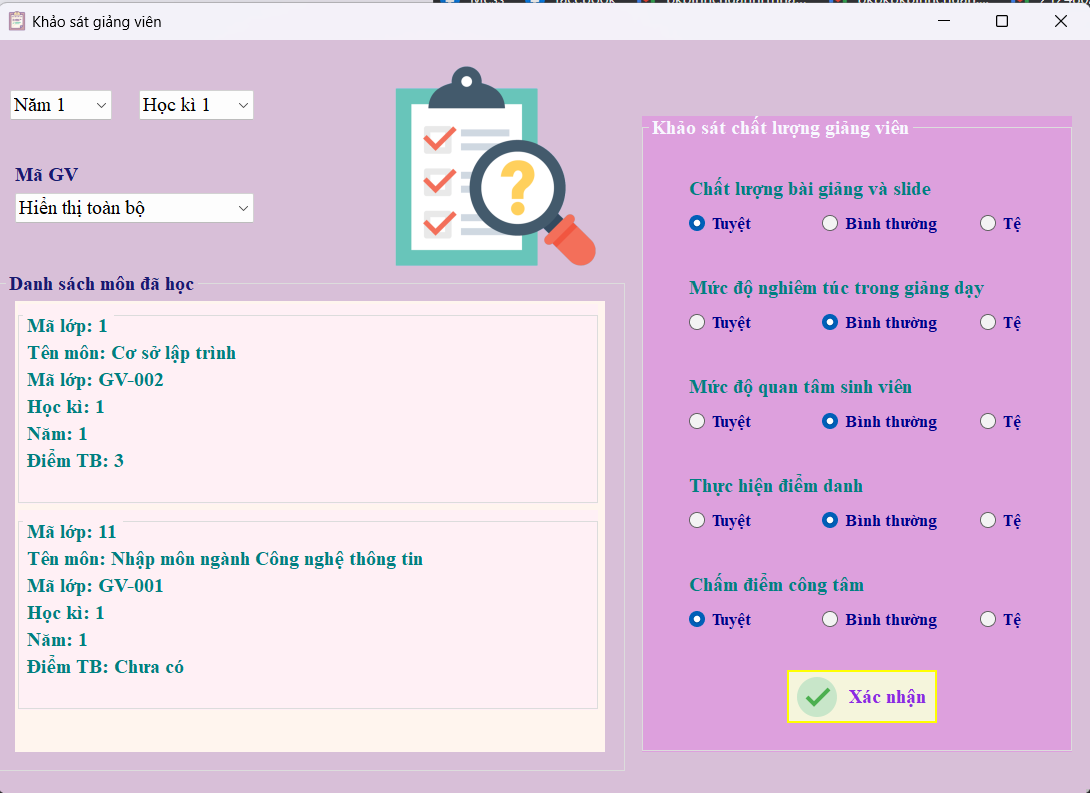
#### Hình ảnh 3.2.19. Form Đăng ký môn học



#### Hình ảnh 3.2.20. Form Xem thời khoá biểu



Hình ảnh 3.2.21. Form Xem các môn học theo Ngành dành cho Sinh viên



#### Hình ảnh 3.2.22. Form Khảo sát Giảng viên

# CHƯƠNG 4. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## **4.1. Kết luận:**

### ***4.1.1. Các chức năng ứng dụng đã làm được:***

* Chưa có sự xác định các môn học Giảng viên có thể dạy (hệ thống chỉ mới xác định Giảng viên thuộc Khoa dạy được toàn bộ môn của khoa đó.

### ***4.1.2. Hạn chế:***

* Một vài thao tác chưa được hoàn hảo, còn một số ít bất tiện trong thao tác do chưa được tối ưu tối đa.
* Thời gian dành cho dự án không quá nhiều vì bị chi phối bởi các môn học khác với khối lượng công việc nặng kha khá.

## **4.2. Hướng phát triển:**

* Nâng cấp các chi tiết thông tin xoay quanh hoạt động dạy học để cải thiện chất lượng sử dụng cho người dùng.
* Tối ưu một vài tính năng để cải thiện tốc độ xử lý và độ tiện lợi cho ứng dụng
* Thêm hệ thống liên lạc hỗ trợ từ Sinh viên gửi tới Giảng viên và từ Giảng viên đến Quản trị.
* Cân nhắc mở rộng phạm vi quản lý nếu cần.

# TÀI LIỆU THAM KHẢO

[1] O’Reilly, C# 7.0 in a Nutshell, 7th Edition, 2017

[2] Andrew Troelsen, Philip Japikse - Pro C# 7 With .NET and .NET Core, 2017

**Website**

**HưỚng dẪn hình thỨc trình bày báo cáo ĐỒ ÁN**

* **Hình thức chung**

- Font chữ trình bày : **Times New Roman**, size: **13**

- Mỗi trang format theo cỡ **top** 2 cm, **bottom** 2 cm, **Right** 2 cm, **Left** 3 cm

* **Hình thức báo cáo theo thứ tự như sau (có mẫu đính kèm):**

**1.** Trang **bìa**

**2.** Trang **bìa lót**

**3.** Trang **mục lục**

**4. Các chương báo cáo**

**6. Kết luận**

**7.** Trang **Tài liệu tham khảo**

[1] Tài liệu từ sách - giáo trình

[2] Tài liệu trên mạng.

**Lưu ý:** Nghiêm cấm các nhóm có ***hành vi sao chép*** nội dung báo cáo từ bất cứ nguồn nào. Nếu bị phát hiện ***hành vi sao chép*** thì sẽ bị xử lý theo quy định.

**Trang mục lục** (Đánh số trang theo kiểu La mã (i, ii, iii…)

* Tên các chương, mục và số thứ tự trang bắt đầu
* các chương mục đưa ra trình bày tối đa 3 cấp.
* Ví dụ:

|  |
| --- |
| Chương I: …….. |
| 1.1. ……………… |
| 1.1.1………………. |
| 1.1.1.1. …………………… |
| 1.2. ……………………. |

**Mục: Vai trò các thành viên**

**Tên nhóm : Nhóm X**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **MSSV** | **Họ tên** | **Vai trò** | **Nội dung thực hiện** |
| 1 |  |  |  |  |
| 2 |  |  |  |  |
| … |  |  |  |  |