

ĐẠI HỌC QUỐC GIA TP. HCM
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA ĐIỆN TỬ - VIỄN THÔNG



ĐỒ ÁN MÔN HỌC - KỸ THUẬT LẬP TRÌNH NÂNG CAO

XÂY DỰNG CHƯƠNG TRÌNH QUẢN LÝ KHÁCH SẠN
SỬ DỤNG NGÔN NGỮ LẬP TRÌNH C#

GIẢNG VIÊN HƯỚNG DẪN

Th.s Huỳnh Quốc Thịnh

NHÓM SINH VIÊN THỰC HIỆN

STT	Họ và tên	Mã số sinh viên
1	Nguyễn Bảo Duy	22200041
2	Bùi Hồng Hà	22200050
3	Đặng Đình Khôi	22200084

TP. Hồ Chí Minh – Năm 2025

ĐẠI HỌC QUỐC GIA TP. HCM
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA ĐIỆN TỬ - VIỄN THÔNG



ĐỒ ÁN MÔN HỌC - KỸ THUẬT LẬP TRÌNH NÂNG CAO

XÂY DỰNG CHƯƠNG TRÌNH QUẢN LÝ KHÁCH SẠN
SỬ DỤNG NGÔN NGỮ LẬP TRÌNH C#

GIẢNG VIÊN HƯỚNG DẪN

Th.s Huỳnh Quốc Thịnh

NHÓM SINH VIÊN THỰC HIỆN

STT	Họ và tên	Mã số sinh viên
1	Nguyễn Bảo Duy	22200041
2	Bùi Hồng Hà	22200050
3	Đặng Đình Khôi	22200084

TP. Hồ Chí Minh – Năm 2025

LỜI NÓI ĐẦU

Trong thời đại công nghệ thông tin phát triển mạnh mẽ, việc áp dụng các giải pháp phần mềm vào quản lý hoạt động kinh doanh là xu hướng tất yếu nhằm tăng cường hiệu suất làm việc, tính chính xác và tính chuyên nghiệp. Đặc biệt trong lĩnh vực lưu trú – khách sạn, nhu cầu quản lý hệ thống phòng, khách hàng, nhân viên và thanh toán một cách đồng bộ, hiệu quả và trực quan ngày càng trở nên quan trọng. Tuy nhiên, không phải cơ sở nào cũng có sẵn phần mềm phù hợp, đặc biệt là các khách sạn vừa và nhỏ còn gặp nhiều khó khăn trong việc tiếp cận các công cụ hiện đại.

Chính vì vậy, nhóm em đã quyết định lựa chọn đề tài **“Chương trình quản lý khách sạn sử dụng ngôn ngữ lập trình C#”** để xây dựng một phần mềm ứng dụng có thể hỗ trợ tối ưu quy trình quản lý tại các cơ sở lưu trú. Việc lựa chọn C# kết hợp với Windows Forms (WinForms) trong đề tài không chỉ phù hợp với nền tảng desktop mà còn giúp nhóm em khai thác tối đa ngôn ngữ C# qua những ưu điểm:

- Giao diện thân thiện, dễ sử dụng và dễ tùy chỉnh theo nhu cầu thực tế.
- Tích hợp tốt với cơ sở dữ liệu như SQL Server, hỗ trợ thao tác dữ liệu nhanh chóng và chính xác.
- Hệ sinh thái .NET mạnh mẽ, có tính ổn định cao, phù hợp với các ứng dụng quản lý nội bộ.
- Hỗ trợ phát triển các module rõ ràng theo hướng lập trình hướng đối tượng, dễ bảo trì và mở rộng.

Thông qua đề tài này, nhóm em không chỉ rèn luyện các kỹ năng lập trình, xây dựng cơ sở dữ liệu, tư duy logic mà còn hiểu rõ hơn về quy trình phát triển phần mềm thực tế từ khâu phân tích yêu cầu, thiết kế giao diện, viết mã nguồn, kiểm thử cho đến hoàn thiện sản phẩm. Đặc biệt, đề tài còn giúp nhóm tiếp cận với những kỹ năng chuyên môn như:

- Tạo các form với nhiều chức năng khác nhau: đăng nhập, thêm phòng, đăng ký khách hàng, thanh toán, quản lý nhân viên,...
- Kết nối cơ sở dữ liệu bằng ngôn ngữ C#, xử lý truy vấn SQL động.
- Quản lý lỗi, kiểm tra tính đầy đủ dữ liệu, xử lý các tương tác của người dùng với phần mềm.

Hơn nữa, đề tài này có tính ứng dụng thực tiễn rất cao. Sau khi hoàn thiện, phần mềm có thể được triển khai tại các khách sạn nhỏ hoặc sử dụng làm nền tảng phát triển cho các dự án lớn hơn trong tương lai, tích hợp thêm các chức năng như đặt phòng online, báo cáo tài chính, quản lý chấm công nhân viên,...

Tóm lại, việc lựa chọn đề tài **“Chương trình quản lý khách sạn sử dụng ngôn ngữ lập trình C#”** không chỉ xuất phát từ nhu cầu thực tiễn mà còn là cơ hội để nhóm em khẳng định năng lực bản thân trong việc xây dựng ứng dụng quản lý chuyên nghiệp, hiệu quả và hiện đại – một bước chuẩn bị quan trọng cho con đường nghề nghiệp trong lĩnh vực phát triển phần mềm sau này.

MỤC LỤC

CHƯƠNG 1: TỔNG QUAN VỀ ĐỒ ÁN.....	1
I. Lí do lựa chọn đề tài.....	1
II. Mô tả đồ án.....	1
1. Mô tả lưu đồ thuật toán	1
2. Mô tả cấu trúc chương trình.....	2
III. Liên kết đến đồ án trên GitHub.....	7
CHƯƠNG 2: CƠ SỞ LÝ THUYẾT.....	8
I. Ngôn ngữ lập trình C# và lập trình hướng đối tượng.....	8
1. Ngôn ngữ lập trình C#.....	8
2. Lập trình hướng đối tượng	8
3. .NET Framework.....	9
4. Giới thiệu về Windows Forms	10
a. Mô hình lập trình Windows Forms	10
b. Các thành phần cơ bản của Windows Forms	10
c. Quy trình xây dựng ứng dụng Windows Forms.....	11
d. Ưu điểm của Windows Forms.....	11
e. Hạn chế của Windows Forms.....	11
II. Xây dựng chương trình C# cơ bản.....	12
1. Cấu trúc chương trình trong C#	12
a. Cấu trúc chương trình.....	12
b. Các hàm console I/O cơ bản	12
c. Các loại dữ liệu cơ bản và chuyển đổi các loại dữ liệu	14
d. Cấu trúc điều kiện và vòng lặp.....	16
e. Truyền tham trị và truyền tham chiếu	18
2. Cấu trúc lớp (class)	20
3. Cấu trúc phương thức (method)	21

4. Cấu trúc hàm khởi tạo (constructor)	22
5. Cấu trúc thuộc tính (properties)	23
III. Các tính chất của lập trình hướng đối tượng	25
1. Tính đóng gói (Encapsulation).....	25
a. Định nghĩa	25
b. Nhiệm vụ của tính đóng gói.....	25
2. Tính kế thừa (Inheritance).....	26
a. Định nghĩa	26
b. Nhiệm vụ của tính kế thừa	26
c. Constructor trong class con	28
3. Tính đa hình (Polymorphic).....	29
a. Định nghĩa	29
b. Nhiệm vụ của tính đa hình	29
c. Chuyển đổi kiểu dữ liệu giữa class.....	30
d. Từ khoá virtual & override	31
4. Tính trừu tượng (Abstraction).....	32
IV. Các quy tắc Clean Code.....	33
1. Quy tắc đặt tên (name)	33
2. Quy tắc viết hàm (method) và lớp (class)	33
3. Quy tắc chú thích (comment).....	34
CHƯƠNG 3: QUÁ TRÌNH THỰC HIỆN ĐỒ ÁN.....	36
I. Quá trình thực hiện đồ án	36
1. Thiết kế Form Login và BackEnd Form Login.....	36
a. Thiết kế Form Login	36
b. Thiết kế BackEnd Login	36
2. Thiết kế Form DashBoard và BackEnd Form DashBoard.....	38
a. Thiết kế Form DashBoard	38

b. Thiết kế BackEnd Form DashBoard	40
3. Thiết kế Form Add Room và BackEnd Form Add Room	43
a. Thiết kế Form Add Room	43
b. Thiết kế BackEnd Form Add Room.....	43
4. Thiết kế Form đăng kí khách hàng và BackEnd Form đăng kí khách hàng	46
a. Thiết kế Form đăng kí khách hàng	46
b. Thiết kế BackEnd Form đăng kí khách hàng	46
5. Thiết kế Form thanh toán phòng và BackEnd Form thanh toán phòng	51
a. Thiết kế Form thanh toán phòng	51
b. Thiết kế BackEnd Form thanh toán phòng	51
6. Thiết kế Form thông tin chi tiết khách hàng và BackEnd Form thông tin chi tiết khách hàng ..	55
a. Thiết kế Form thông tin chi tiết khách hàng	55
b. Thiết kế BackEnd Form thông tin chi tiết khách hàng.....	55
7. Thiết kế Form quản lý nhân viên và BackEnd Form quản lý nhân viên.....	57
a. Thiết kế Form quản lý nhân viên	57
b. Thiết kế BackEnd Form quản lý nhân viên.....	58
8. Chương trình trong file Program.cs	62
9. Chương trình trong file function.cs.....	62
II. Kết quả đồ án	64
CHƯƠNG 4: TỔNG KẾT ĐỒ ÁN	74
I. Các kiến thức quan trọng đạt được từ đồ án.....	74
II. Hướng phát triển của đồ án.....	75
1. Tích hợp cơ sở dữ liệu từ xa (Online/Cloud)	75
2. Hệ thống quản lý khách sạn và thống kê nâng cao	75
3. Tích hợp thanh toán điện tử	75
4. Tích hợp công nghệ AI cho đề xuất phòng	75
5. Tăng cường bảo mật hệ thống.....	75

III. Phân công nhiệm vụ và tự đánh giá	76
1. Phân công nhiệm vụ	76
2. Tự đánh giá	77
3. Đánh giá tổng quát	78
LỜI CẢM ƠN	80
TÀI LIỆU THAM KHẢO.....	81

DANH MỤC CÁC HÌNH ẢNH

Hình 1: Lưu đồ thuật toán xây dựng chương trình quản lý khách sạn	1
Hình 2: Giao diện đăng nhập (login) để truy cập vào hệ thống quản lý khách sạn.....	2
Hình 3: Giao diện “Thêm phòng” của hệ thống quản lý khách sạn	3
Hình 4: Giao diện “Đăng kí khách hàng” của hệ thống quản lý khách sạn.....	4
Hình 5: Giao diện “Thanh toán” của hệ thống quản lý khách sạn.....	4
Hình 6: Giao diện “Thông tin chi tiết khách hàng” của hệ thống quản lý khách sạn.....	5
Hình 7: Giao diện nhỏ “Đăng ký nhân viên” trong giao diện “Nhân viên” của hệ thống quản lý khách sạn	6
Hình 8: Giao diện nhỏ “Thông tin nhân viên” trong giao diện “Nhân viên” của hệ thống quản lý khách sạn	6
Hình 9: Giao diện nhỏ “Xóa nhân viên” trong giao diện “Nhân viên” của hệ thống quản lý khách sạn	7
Hình 10: Ngôn ngữ lập trình C# và .NET Framework.....	8
Hình 11: Thiết kế giao diện Đăng nhập (Form Login).....	36
Hình 12: Database cho thông tin về phòng (dbo.rooms).....	38
Hình 13: Database cho thông tin về khách hàng (dbo.customer).....	39
Hình 14: Database cho thông tin về nhân viên (dbo.employee).....	39
Hình 15: Thiết kế giao diện DashBoard (Form DashBoard).....	39
Hình 16: Thiết kế giao diện “Thêm Phòng” (Form Add Room).....	43
Hình 17: Thiết kế giao diện “Đăng kí khách hàng”	46
Hình 18: Thiết kế giao diện “Thanh Toán” (Thanh Toán Tiền Phòng)	51
Hình 19: Thiết kế giao diện “Thông Tin Chi Tiết Khách Hàng”	55
Hình 20: Thiết kế giao diện tab “Đăng Ký Nhân Viên” trong giao diện “Nhân Viên”	57
Hình 21: Thiết kế giao diện tab “Thông tin nhân viên” trong giao diện “Nhân Viên”	57
Hình 22: Thiết kế giao diện tab “Xóa nhân viên” trong giao diện “Nhân Viên”	58
Hình 23: Dòng chữ thông báo xuất hiện ở giao diện “Đăng nhập”.....	64

Hình 24: Giao diện DashBoard của hệ thống quản lý khách sạn khi đăng nhập thành công	64
Hình 25: Cập nhật các thông tin về Số Phòng, Loại Phòng, Loại Giường và Giá Tiền trong giao diện “Thêm phòng”	65
Hình 26: Kết quả khi nhân viên cập nhật các thông tin về phòng ở lên hệ thống thành công	65
Hình 27: Điền các thông tin của khách thuê phòng trong giao diện “Đăng ký khách hàng”	66
Hình 28: Kết quả cập nhật thông tin của khách hàng lên hệ thống sau khi đăng ký thành công	66
Hình 29: Nhập vào thông tin Tên và Số Phòng của người thuê phòng để thanh toán	67
Hình 30: Hộp thoại thông báo xuất hiện khi thanh toán phòng thành công	67
Hình 31: Hộp thoại thông báo xuất hiện khi nhập một tên khách hàng không có trong hệ thống	68
Hình 32: Thông tin chi tiết về tất cả các khách hàng	68
Hình 33: Thông tin chi tiết về những khách hàng đang thuê phòng	69
Hình 34: Thông tin chi tiết về những khách hàng đã hoàn tất thanh toán tiền phòng và checkout.....	69
Hình 35: Điền các thông tin để đăng kí trở thành nhân viên của khách sạn.....	70
Hình 36: Kết quả sau khi đăng kí nhân viên thành công.....	70
Hình 37: Xem thông tin chi tiết về tất cả nhân viên đang làm việc tại khách sạn.....	71
Hình 38: Nhập ID của nhân viên cần xoá để hệ thống xoá khỏi danh sách nhân viên.....	71
Hình 39: Kết quả khi xoá nhân viên thành công	72
Hình 40: Biểu tượng khởi tạo của quản lí khách sạn.....	73
Hình 41: Ứng dụng WinForms quản lí khách sạn trên màn hình Window	73

DANH MỤC CÁC BẢNG

Bảng 1: So sánh lập trình tuần tự và lập trình hướng đối tượng.....	9
Bảng 2: Các loại dữ liệu cơ bản trong ngôn ngữ C#.....	15
Bảng 3: Phân công công việc các thành viên trong nhóm	76
Bảng 4: Kết quả tự đánh giá mức độ hoàn thành công việc của các thành viên trong nhóm	77
Bảng 5: Kết quả tự đánh giá mức độ hoàn thành công việc của nhóm dựa trên tiêu chí của đề án	79

CHƯƠNG 1: TỔNG QUAN VỀ ĐỒ ÁN

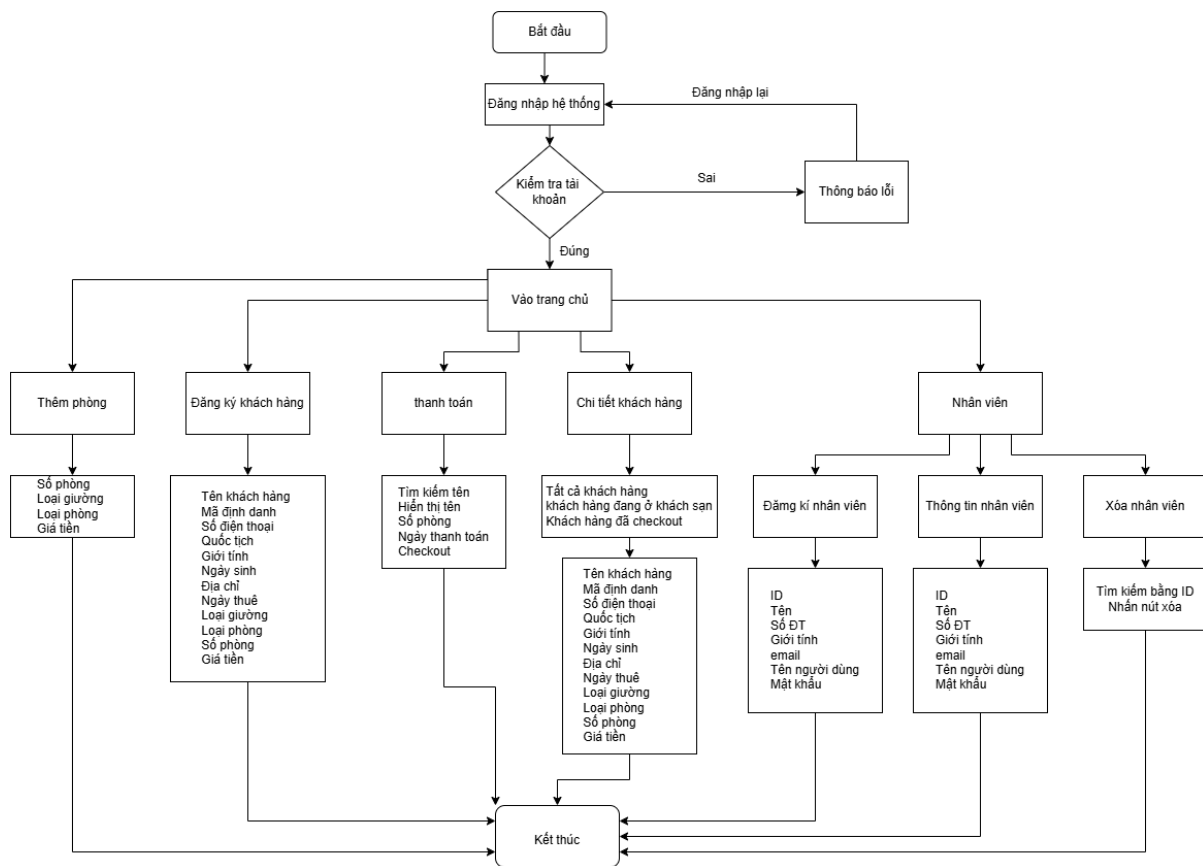
I. Lí do lựa chọn đề tài

Nhóm em quyết định chọn đề tài **“Chương trình quản lý khách sạn sử dụng ngôn ngữ lập trình C#”** cho đồ án môn học **Kỹ thuật lập trình nâng cao** vì đây là một ứng dụng thực tiễn, phù hợp với nhu cầu số hóa trong lĩnh vực du lịch – lưu trú. Đề tài giúp các thành viên trong nhóm rèn luyện được thêm nhiều kỹ năng và biết cách vận dụng kiến thức đã học trên lớp về lập trình C#, thiết kế giao diện WinForms, xử lý cơ sở dữ liệu SQL để xây dựng một hệ thống quản lý khoa học, dễ sử dụng, hỗ trợ hiệu quả cho việc quản lý phòng, khách hàng, nhân viên và thanh toán.

II. Mô tả đồ án

1. Mô tả lưu đồ thuật toán

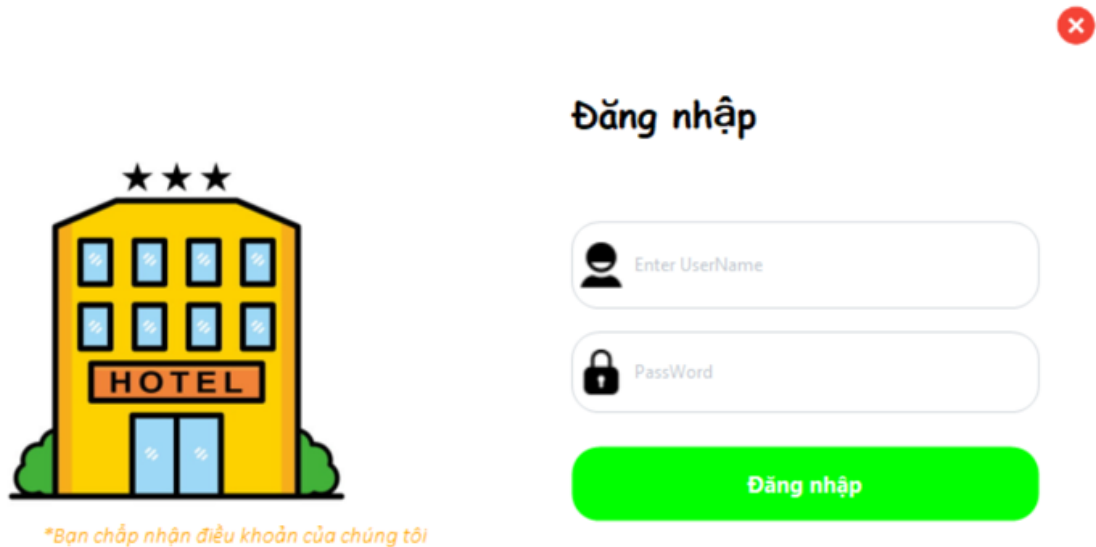
Để minh họa luồng hoạt động chính của chương trình quản lý khách sạn, nhóm em đã xây dựng lưu đồ thuật toán (Flowchart) chi tiết như hình bên dưới, thể hiện các module chính của hệ thống, bao gồm chức năng Đăng nhập, Thêm phòng, Đăng ký khách hàng, Thanh toán, Chi tiết khách hàng và Quản lý nhân viên, cung cấp cái nhìn tổng quan về luồng dữ liệu và tương tác giữa các thành phần.




Hình 1: Lưu đồ thuật toán xây dựng chương trình quản lý khách sạn

2. Mô tả cấu trúc chương trình

- Giao diện đầu tiên của hệ thống là form đăng nhập (login), nơi nhân viên nhập tên đăng nhập (username) và mật khẩu (password) để truy cập hệ thống.
 - Khi nhấn vào nút “Đăng nhập” màu xanh lá, nếu thông tin đăng nhập chính xác, hệ thống sẽ đưa người dùng đến giao diện kế tiếp.
 - Ngược lại, nếu thông tin đăng nhập không đúng, hệ thống sẽ hiển thị cảnh báo bằng dòng chữ màu đỏ: **“Tên đăng nhập hoặc mật khẩu sai”**.



Hình 2: Giao diện đăng nhập (login) để truy cập vào hệ thống quản lý khách sạn

- Khi đăng nhập thành công, ta sẽ thấy xuất hiện thanh "panel" màu xanh ngọc sẽ trượt mượt mà để di chuyển qua các giao diện chính của chương trình bao gồm: Thêm phòng, Đăng ký khách hàng, Thanh toán, Chi tiết khách hàng và Nhân viên.
 - Ở mỗi giao diện, nhân viên bắt buộc phải nhập đầy đủ tất cả thông tin yêu cầu thì mới có thể thực hiện thao tác khởi tạo dữ liệu. Nếu thông tin chưa đầy đủ, giao diện sẽ hiển thị cảnh báo: "Xin vui lòng điền đầy đủ thông tin" nhằm đảm bảo tính chính xác và đầy đủ của dữ liệu.
- Dấu  màu đỏ, nằm ở góc trên bên phải màn hình, được sử dụng để thoát khỏi giao diện hiện tại hoặc đăng xuất khỏi phần mềm khi cần.
- Giao diện thứ hai là giao diện “Thêm Phòng”, cho phép nhân viên tạo các phòng trống mới trong khách sạn.
 - Các thông tin cần nhập bao gồm: Số phòng, Loại phòng (gồm AC và Non-AC), Loại giường (gồm Single, Double, Triple) và Giá tiền.

- Sau khi nhập đầy đủ và nhấn vào nút “Thêm phòng” màu xanh dương, giao diện sẽ hiển thị thông báo “Đã thêm phòng”.
- Phòng mới tạo sẽ được hiển thị ở khung bên trái và nếu chưa được thuê, phòng đó sẽ có trạng thái “NO” hiển thị ở cuối dòng thông tin.

ID	Số Phòng	Loại Phòng	Loại Giường	Giá Tiền	Trạng Thái
1	123	Ac	Double	23	NO
2	5	Non_Ac	Single	1234	YES
3	10	Ac	Single	120000	YES
4	620	Non_Ac	Triple	300000	YES
5	6	Ac	Single	330000	YES

Hình 3: Giao diện “Thêm phòng” của hệ thống quản lý khách sạn

- Giao diện thứ ba là giao diện “Đăng ký khách hàng”, nơi nhân viên tiến hành nhập thông tin đăng ký cho khách hàng.
 - Các trường thông tin bao gồm: Họ tên, Số điện thoại, Quốc tịch, Giới tính (Male, Female, Other), Ngày sinh, Mã định danh, Địa chỉ, Ngày đăng ký, Loại giường, Loại phòng, Số phòng và Giá tiền.
 - Sau khi điền đầy đủ các trường thông tin trên và nhấn vào nút “Thêm khách hàng” màu xanh dương, giao diện sẽ hiển thị thông báo: “Số phòng ... đã đăng ký thành công cho khách hàng”.
 - Đặc biệt, khi nhân viên nhập xong các thông tin liên quan đến loại giường và loại phòng, sau đó nhấn vào mục số phòng, hệ thống sẽ tự động hiển thị danh sách các số phòng còn trống (được tạo ở giao diện “Thêm phòng”) để nhân viên lựa chọn.
 - Mục giá tiền cũng sẽ tự động cập nhật tương ứng với loại giường và loại phòng đã chọn.

Hình 4: Giao diện “Đăng kí khách hàng” của hệ thống quản lý khách sạn

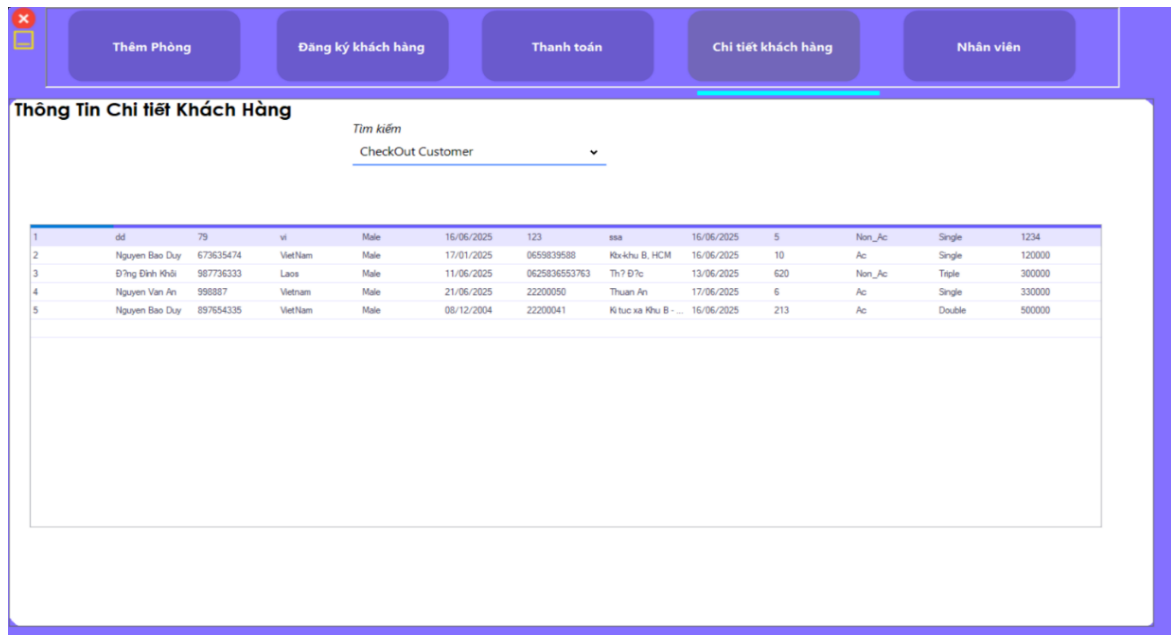
- Giao diện thứ tư là giao diện “Thanh toán”, nơi nhân viên thực hiện thủ tục thanh toán khi khách hàng trả phòng.
 - Nhân viên sẽ nhập tên khách hàng vào ô tìm kiếm, khung bên dưới sẽ hiển thị thông tin chi tiết về phòng khách hàng đã thuê.
 - Sau khi xác nhận thông tin chính xác, nhân viên nhấn chọn mục đó để cập nhật tên khách hàng và số phòng vào các trường bên dưới. Sau đó, nhập ngày thanh toán và nhấn vào nút “Thanh toán” màu xanh dương.
 - Khi hoàn tất, hệ thống sẽ hiển thị thông báo: “Thanh toán thành công” để xác nhận quá trình đã được xử lý.

Hình 5: Giao diện “Thanh toán” của hệ thống quản lý khách sạn

- Giao diện thứ năm là giao diện “Thông tin chi tiết khách hàng”, giúp nhân viên theo dõi thông tin của cả những khách đang lưu trú lẫn những khách đã trả phòng.

Tại ô tìm kiếm, hệ thống cung cấp ba lựa chọn:

- All Customer Details (Tất cả khách hàng).
- In Hotel Customer (Khách đang ở trong khách sạn).
- Checkout Customer (Khách đã trả phòng).
- Dựa trên lựa chọn của nhân viên, khung hiển thị bên dưới sẽ tự động cập nhật danh sách khách hàng tương ứng.



Hình 6: Giao diện “Thông tin chi tiết khách hàng” của hệ thống quản lý khách sạn

- Giao diện cuối cùng là giao diện “Nhân Viên”, bao gồm ba giao diện nhỏ: Đăng ký nhân viên, Thông tin chi tiết nhân viên và Xóa nhân viên.
 - Trong phần “Đăng ký nhân viên”, mỗi nhân viên sẽ có một ID riêng được hiển thị ở góc bên trái. Quản lý sẽ nhập các thông tin bao gồm: Họ tên, Số điện thoại, Giới tính (Male, Female, Other), Email, Tên người dùng (username), và Mật khẩu. Sau khi nhấn nút “Đăng ký” màu xanh dương, hệ thống sẽ hiển thị thông báo: “Đăng ký nhân viên thành công”, đồng thời tạo tài khoản đăng nhập để nhân viên có thể sử dụng tại form login ban đầu.
 - Trong phần “Thông tin chi tiết nhân viên”, giao diện sẽ hiển thị bảng dữ liệu chứa đầy đủ thông tin các nhân viên đã đăng ký, bao gồm: ID, Họ tên, Số điện thoại, Giới tính, Email, Tên người dùng và Mật khẩu.
 - Trong phần “Xóa nhân viên”, quản lý sẽ nhập ID của nhân viên cần xóa (ID được cấp ở phần đăng ký). Khi nhập đúng ID, khung bên dưới sẽ hiển thị đầy đủ thông tin tương ứng. Khi nhấn nút “Xóa”, hệ thống sẽ hiển thị cảnh báo: “Bạn có chắc chắn xóa không?”.

Nếu người dùng xác nhận bằng cách nhấn “Yes”, giao diện sẽ hiển thị thông báo: “Thông tin nhân viên được xóa”, xác nhận hành động đã hoàn tất.

Nhân Viên

Đăng Ký Nhân Viên | Thông tin nhân viên | Xóa nhân viên

ID - 6

Tên:

Tên Người Dùng:

Số Điện Thoại:

Mật Khẩu:

Giới tính:

Email:

Đăng Ký

Hình 7: Giao diện nhỏ “Đăng ký nhân viên” trong giao diện “Nhân viên” của hệ thống quản lý khách sạn

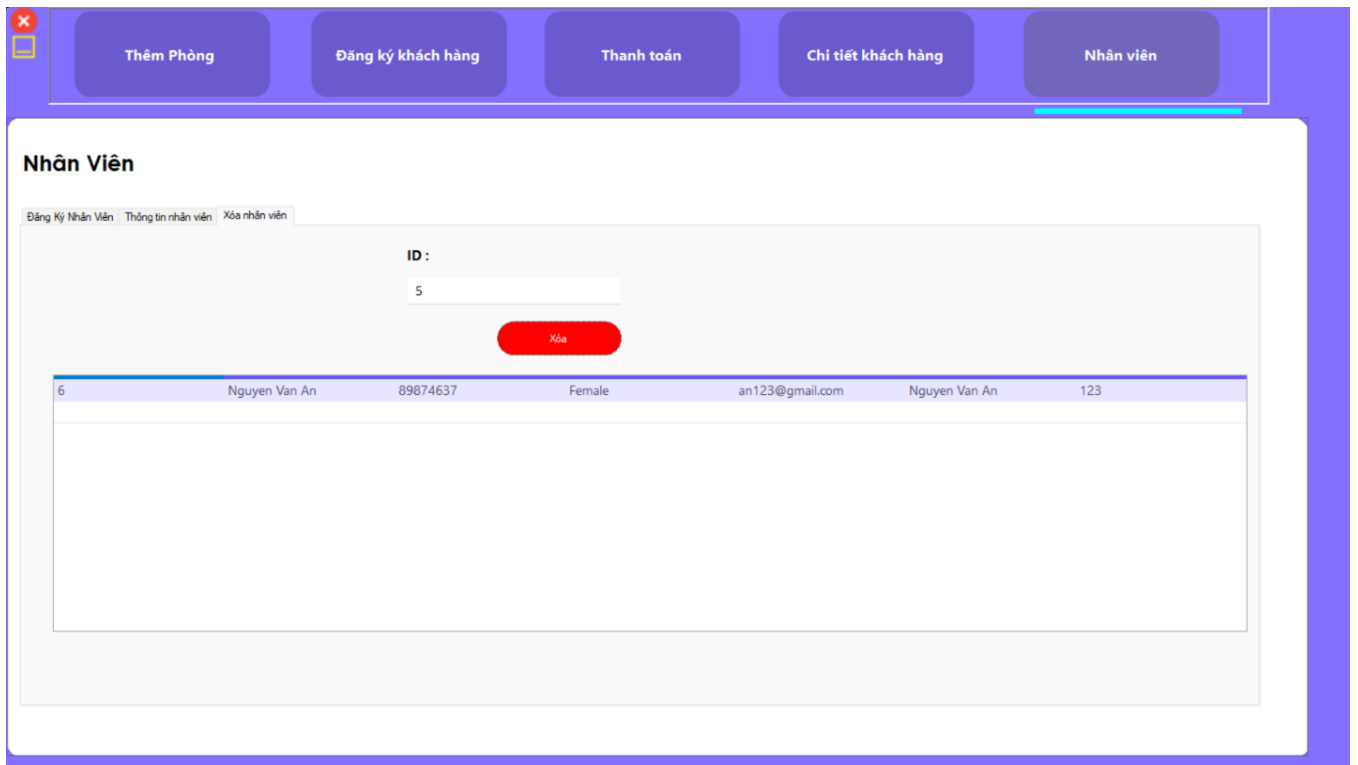
Nhân Viên

Đăng Ký Nhân Viên | Thông tin nhân viên | Xóa nhân viên

Thông tin nhân viên

5	Bùi Hồng Hà	978383773	Male	ha@gmail.com	Bùi Hồng Hà	123
6	Nguyễn Văn An	89874637	Female	an123@gmail.com	Nguyễn Văn An	123

Hình 8: Giao diện nhỏ “Thông tin nhân viên” trong giao diện “Nhân viên” của hệ thống quản lý khách sạn



Hình 9: Giao diện nhỏ “Xóa nhân viên” trong giao diện “Nhân viên” của hệ thống quản lý khách sạn

III. Liên kết đến đồ án trên GitHub

- Nhóm em đã cập nhật toàn bộ chương trình thực hiện trên Visual Studio 2022 lên GitHub, kèm theo video mô tả chi tiết quá trình hoạt động của chương trình quản lý khách sạn.
- Chúng ta có thể truy cập trực tiếp theo đường dẫn bên dưới hoặc quét mã QR để xem được toàn bộ nội dung của đồ án:

<https://github.com/dangdinhkhoi2010/DO-AN-KI-THUAT-LAP-TRINH-NANG-CAO>



CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

I. Ngôn ngữ lập trình C# và lập trình hướng đối tượng

1. Ngôn ngữ lập trình C#

C# (C-Sharp) là một ngôn ngữ lập trình hướng đối tượng (OOP) đa năng, được tập đoàn Microsoft phát triển vào năm 2000. C# chạy trên nền tảng .NET Framework, kết hợp sức mạnh của C++ với sự đơn giản của Visual Basic, hỗ trợ phát triển ứng dụng desktop, web, mobile, game (Unity) và Internet of Things (IoT).

Các đặc trưng của ngôn ngữ C#:

- Hướng đối tượng (OOP): mọi thành phần trong C# đều là đối tượng (class, struct, interface). C# hỗ trợ 4 tính chất quan trọng của lập trình hướng đối tượng là tính đóng gói, tính kế thừa, tính đa hình và tính trừu tượng.
- An toàn kiểu dữ liệu (Type-safe): C# hỗ trợ kiểm tra kiểu dữ liệu tại thời điểm biên dịch, giúp ngăn lỗi truy cập bộ nhớ không hợp lệ.
- Quản lý bộ nhớ tự động (Garbage Collection): C# tự động giải phóng bộ nhớ không dùng, giúp giảm rủi ro rò rỉ bộ nhớ.
- Đa nền tảng: C# có thể chạy trên hệ điều hành Windows, Linux, macOS qua .NET Core hoặc .NET 5+.
- Hỗ trợ LINQ (Language Integrated Query): C# hỗ trợ truy vấn dữ liệu trực tiếp trong code (SQL-like).



Hình 10: Ngôn ngữ lập trình C# và .NET Framework

2. Lập trình hướng đối tượng

OOP là mô hình lập trình dựa trên khái niệm đối tượng (object), mỗi đối tượng chứa dữ liệu (thuộc tính) và hành vi (phương thức). OOP giúp quản lý mã nguồn phức tạp bằng cách mô phỏng thế giới thực, có nhiều ưu điểm vượt trội hơn so với lập trình tuần tự (procedural).

So sánh lập trình tuần tự và hướng đối tượng:

Tiêu chí	Lập trình tuần tự	Lập trình hướng đối tượng
Tổ chức code	Tập trung vào hàm (function)	Tập trung vào đối tượng (object)
Dữ liệu & Code	Dữ liệu và hàm tách rời	Dữ liệu và hàm gói trong đối tượng
Khả năng tái sử dụng	Khó, phụ thuộc vào hàm	Dễ, thông qua tính kế thừa, đa hình
Tính bảo mật	Dữ liệu có thể bị truy cập tự do	Đóng gói ẩn dữ liệu nội bộ
Ngôn ngữ phổ biến	C, Pascal	C#, Java, Python

Bảng 1: So sánh lập trình tuần tự và lập trình hướng đối tượng

3. .NET Framework

.NET Framework là nền tảng phát triển ứng dụng của Microsoft, cung cấp thư viện lớp (FCL) và môi trường thực thi mã (CLR). .NET Framework hỗ trợ xây dựng ứng dụng desktop (WinForms, WPF), web (ASP.NET) và các dịch vụ (WCF).

Kiến trúc .NET Framework gồm hai thành phần chính là CLR và FCL.

- CLR (Common Language Runtime): giúp quản lý bộ nhớ (Garbage Collection), biên dịch mã (JIT Compiler) cũng như xử lý ngoại lệ (exception). Mã C# biên dịch thành MSIL (Microsoft Intermediate Language) và CLR giúp biên dịch MSIL thành mã máy.
- FCL (Framework Class Library): có các thư viện khổng lồ chứa các lớp để xử lý XML, kết nối DB, bảo mật, đa luồng,... ví dụ như System.IO thực hiện đọc hoặc ghi file, System.Data truy vấn database,...

Một trong các phiên bản .NET Framework mới nhất trong Visual Studio 2022 là 4.7.2, phiên bản này có các ưu điểm:

- Tính ổn định và tương thích cao: đây là phiên bản Long-Term Support (LTS), hỗ trợ đến năm 2024.
- Cải thiện hiệu năng: cải tiến GC, JIT, async.
- Cải thiện tính bảo mật: hỗ trợ TLS 1.2, SHA-256.
- Có tính tương thích ngược: có thể chạy ứng dụng viết trên các phiên bản .NET 4.x cũ.

4. Giới thiệu về Windows Forms

Windows Forms (WinForms) là một công nghệ giao diện đồ họa (GUI - Graphical User Interface) của nền tảng .NET, được Microsoft phát triển nhằm tạo ra các ứng dụng Windows có giao diện trực quan, dễ sử dụng. WinForms cung cấp các thành phần (controls) cơ bản như nút bấm (Button), hộp văn bản (TextBox), nhãn (Label), menu (MenuStrip) và nhiều điều khiển khác để xây dựng giao diện người dùng.

a. Mô hình lập trình Windows Forms: một ứng dụng Windows Forms được tổ chức theo mô hình lập trình hướng đối tượng, trong đó:

- Mỗi cửa sổ (Form) là một lớp kế thừa từ lớp cơ sở System.Windows.Forms.Form.
- Các thành phần điều khiển (controls) là các đối tượng thuộc các lớp có sẵn trong thư viện System.Windows.Forms.
- Các sự kiện (events) được xử lý thông qua các phương thức (methods) do lập trình viên định nghĩa.

b. Các thành phần cơ bản của Windows Forms

- Form: là cửa sổ chính hoặc các hộp thoại con của ứng dụng. Lớp Form đóng vai trò làm vùng chứa các điều khiển khác.
- Controls: là các thành phần giao diện như Button, Label, TextBox, ComboBox, ListBox, RadioButton, CheckBox, DataGridView,... Chúng được thêm vào Form để phục vụ nhập xuất dữ liệu và tương tác với người dùng.
- Event: mỗi điều khiển có thể phát sinh các sự kiện như Click, TextChanged, CheckedChanged,... Lập trình viên sẽ viết các trình xử lý (event handler) để định nghĩa hành vi của chương trình khi sự kiện xảy ra.

c. Quy trình xây dựng ứng dụng Windows Forms: một ứng dụng Windows Forms thường được phát triển theo các bước:

- Thiết kế giao diện: sử dụng Visual Studio để kéo thả các điều khiển từ Toolbox lên Form, sắp xếp, thiết lập thuộc tính.
- Viết mã xử lý: lập trình xử lý các sự kiện, khai báo biến, phương thức, truy xuất dữ liệu.
- Kiểm thử và chạy ứng dụng: chạy thử, kiểm tra các chức năng và sửa lỗi.
- Đóng gói và triển khai: tạo file cài đặt hoặc file thực thi (.exe) để triển khai cho người dùng cuối.

d. Ưu điểm của Windows Forms

- Dễ học, dễ sử dụng, phù hợp với người mới bắt đầu lập trình GUI.
- Tích hợp sẵn trong Visual Studio, hỗ trợ thiết kế kéo thả trực quan.
- Tích hợp tốt với các thành phần khác của nền tảng .NET.
- Thích hợp cho các ứng dụng doanh nghiệp, quản lý dữ liệu, công cụ nội bộ.

e. Hạn chế của Windows Forms

- Không phù hợp với các ứng dụng hiện đại yêu cầu giao diện đẹp, linh hoạt trên nhiều nền tảng.
- Chỉ chạy trên hệ điều hành Windows.
- Giao diện thiết kế theo phong cách cổ điển, khó tùy biến so với các công nghệ mới như WPF hay UWP.

II. Xây dựng chương trình C# cơ bản

1. Cấu trúc chương trình trong C#

a. Cấu trúc chương trình

Một chương trình C# cơ bản thường có cấu trúc sau:

- Các câu lệnh **using**: khai báo các thư viện cần dùng (System, Collections,...).
- **namespace**: nhóm các lớp liên quan.
- **class** (hoặc **struct**): chứa các thành phần của chương trình.
- Phương thức **Main**: điểm bắt đầu (entry point) của chương trình.
- Các phần hỗ trợ: biến toàn cục, hằng số,...

Chương trình minh họa:

```
using System;           // 1. Khai báo thư viện cần dùng
namespace DemoApp       // 2. Namespace gom nhóm lớp
{
    class Program        // 3. Class chính
    {
        // 4. Main: entry point
        static void Main(string[] args)
        {
            Console.WriteLine("Hello, C#!");
        }
    }
}
```

- o **using System;** cho phép dùng lớp **Console** mà không phải viết **System.Console**.
- o **namespace DemoApp {...}** thực hiện gom các lớp hoặc struct có liên quan vào chung một không gian tên.
- o **class Program** là nơi chứa phương thức **Main**.
- o **static void Main(string[] args)** là nơi .NET runtime gọi đầu tiên.
 - **static**: không cần khởi tạo đối tượng mới vẫn gọi được.
 - **void**: không trả về giá trị.
 - **string[] args**: mảng chuỗi nhận tham số dòng lệnh.

b. Các hàm console I/O cơ bản

- **Console.Write(string value)**

- o Chức năng: in (xuất) chuỗi **value** ra console mà không tự động xuống dòng sau khi in.
- o Có nhiều overload, cho phép in số, ký tự, hoặc định dạng.

Chương trình minh hoạ:

```
Console.Write("Điểm trung bình: ");  
Console.Write(9.5);
```

Kết quả chương trình: Điểm trung bình: 9.5

- `Console.WriteLine(string value)`

- Chức năng: in chuỗi `value` ra console và tự động thêm ký tự xuống dòng `\n` cuối cùng.
- Câu lệnh này hữu dụng khi chúng ta muốn mỗi lần ghi một dòng mới.

Chương trình minh hoạ:

```
Console.WriteLine("Chào mừng bạn đến với C#!");  
Console.WriteLine("Tổng = {0}", 5);
```

Kết quả chương trình:

Chào mừng bạn đến với C#!

Tổng = 5

- `Console.Read()`

- Chức năng: đọc một ký tự (dưới dạng `int`) từ bàn phím, nhưng không hiển thị ký tự đó lên console.
- Trả về giá trị Unicode của ký tự (hoặc `-1` nếu không còn dữ liệu).

Chương trình minh hoạ:

```
int code = Console.Read();  
char c = (char)code;  
  
Console.WriteLine("Bạn vừa nhấn: " + c);
```

Kết quả chương trình sau khi nhấn kí tự in hoa A:

Bạn vừa nhấn: A

- `Console.ReadLine()`

- Chức năng: đọc một dòng (từ vị trí con trỏ hiện tại tới khi gặp Enter) và trả về dưới dạng `string`.
- Là cách phổ biến nhất để nhập chuỗi từ người dùng.

Chương trình minh họa:

```
Console.Write("Nhập tên bạn: ");  
string name = Console.ReadLine();  
  
Console.WriteLine("Xin chào, " + name + "!");
```

Kết quả chương trình sau khi nhập vào tên Khoi:

Nhập tên bạn: Khoi

Xin chào, Khoi!

- `Console.ReadKey(bool intercept = false)`

- Chức năng: đọc một phím (khi người dùng nhấn bất kỳ phím nào), trả về một `ConsoleKeyInfo` chứa thông tin phím và trạng thái phím modifier (Shift, Alt, Ctrl).
- Nếu `intercept = true`, ký tự vừa nhấn sẽ không hiển thị lên console. Ngược lại (mặc định `false`), ký tự sẽ hiển thị.

Chương trình minh họa:

```
Console.WriteLine("Nhấn một phím bất kỳ để tiếp tục...");  
Console.ReadKey(true);
```

Kết quả chương trình: Nhấn một phím bất kỳ để tiếp tục...

Chương trình sẽ đợi ta ấn một phím bất kì, kí tự nhấn không được hiển thị lên console.

c. Các loại dữ liệu cơ bản và chuyển đổi các loại dữ liệu

Trong C# có các loại dữ liệu cơ bản là:

- Kiểu nguyên: `int` (số nguyên 32 bits), `long` (số nguyên 64 bits).
- Kiểu thực: `float` (số thực đơn), `double` (số thực kép, có độ chính xác cao hơn số thực đơn), `decimal` (số thực kép có độ chính xác cao, thường dùng trong tài chính).
- Kiểu kí tự: `char` (một kí tự Unicode 16 bits), `string` (một dãy kí tự Unicode, có thể rỗng hoặc rất dài).
- Kiểu logic: `bool` (giá trị logic, chỉ có `true` hoặc `false`).

Data Types	Memory Size	Range
char	1 byte	-128 to 127
signed char	1 byte	-128 to 127
unsigned char	1 byte	0 to 255
short	2 byte	-32,768 to 32,767
signed Short	2 byte	-32,768 to 32,767
unsigned Short	2 byte	0 to 65,535
int	4 byte	-2,147,483,648 to 2,147,483,647
signed int	4 byte	-2,147,483,648 to 2,147,483,647
unsigned int	4 byte	0 to 4,294,967,295
long	8 byte	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
signed long	8 byte	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
unsigned long	8 byte	0 to 18,446,744,073,709,551,615
float	4 byte	$1.5 * 10^{-45}$ to $3.4 * 10^{38}$ (7 Digit)
double	8 byte	$5 * 10^{-324}$ to $1.7 * 10^{308}$
decimal	16 byte	$-7.9 * 10^{28}$ to $7.9 * 10^{28}$

Bảng 2: Các loại dữ liệu cơ bản trong ngôn ngữ C#

Chuyển đổi giữa các loại dữ liệu trong C#:

- Chuyển đổi ngầm định (implicit): chuyển từ kiểu dữ liệu nhỏ hơn sang lớn hơn mà không làm mất dữ liệu.
- Chuyển đổi tường minh (explicit): chuyển từ kiểu dữ liệu lớn hơn sang nhỏ hơn, có thể mất dữ liệu, phải dùng ép kiểu (cast).
- Chuyển đổi dùng `Convert.Toxxx(...)`: cách dùng phổ biến, an toàn với `null` (trả về 0 cho số), tuy nhiên có thể gây lỗi với chuỗi không đúng định dạng.
- Chuyển đổi dùng `xxx.Parse(...)`: chỉ dành cho chuỗi đúng định dạng, nếu sai định dạng sẽ ném `FormatException`.
- Chuyển đổi dùng `xxx.TryParse(...)`: an toàn vì trả về `bool` cho biết có parse được hay không, tránh exception.

Chương trình minh họa:

```
using System;
class Program
{
    static void Main()
    {
        // 1. Implicit
        int a = 100;
        double d = a;
```

```

    Console.WriteLine(d);
    // 2. Explicit
    double pi = 3.14159;
    int ipi = (int)pi;
    Console.WriteLine(ipi);
    // 3. Convert
    string s1 = "200";
    int n1 = Convert.ToInt32(s1);
    Console.WriteLine(n1 + 5);
    // 4. Parse
    string s2 = "2.5";
    float f2 = float.Parse(s2);
    Console.WriteLine(f2 * 4);
    // 5. TryParse
    string bad = "xyz";
    bool ok = int.TryParse(bad, out int r);
    Console.WriteLine(ok);
    Console.WriteLine(r);
}
}

```

Kết quả chương trình:

```

100    // từ implicit int → double
3      // từ explicit double → int
205    // Convert: 200 + 5 = 205
10     // Parse: 2.5 * 4 = 10
False  // TryParse thất bại
0      // giá trị mặc định của r

```

d. Các cấu trúc điều kiện và vòng lặp

- Câu lệnh `if`

- Cú pháp:

```

if (điều kiện)
{
    // khối lệnh thực thi khi điều kiện đúng
}

```

- Chức năng: nếu điều kiện là `true` thì chạy khối lệnh, ngược lại bỏ qua.

- Câu lệnh `if...else`

- Cú pháp:

```

if (điều kiện)
{
    // khối lệnh thực thi khi điều kiện đúng
}
else
{
    // khối lệnh thực thi khi điều kiện sai
}

```

- Chức năng: nếu điều kiện là `true` thì thực hiện khối lệnh bên trong `if`, còn nếu điều kiện là `false` thì thực hiện khối lệnh bên trong `else`.

- Câu lệnh `if...else if...else`

- Cú pháp:

```
if (điều kiện 1)
{
    // nhánh 1
}
else if (điều kiện 2)
{
    // nhánh 2
}
else
{
    // mặc định
}
```

- Chức năng: tương tự như câu lệnh `if...else`, cho phép kiểm tra nhiều nhánh điều kiện.

- Vòng lặp `for`

- Cú pháp:

```
for (khởi_tạo; điều_kiện_dừng; bước_lặp)
{
    // khối lệnh thực thi mỗi lần lặp
}
```

- Chức năng: dùng khi biết trước số lần lặp (điều kiện khởi tạo, điều kiện dừng và bước lặp nằm ngay phần khai báo). Ví dụ: lặp qua chỉ số từ 0 đến $n - 1$.

- Vòng lặp `while`

- Cú pháp:

```
while (điều_kiện)
{
    // khối lệnh thực thi mỗi lần lặp
}
```

- Chức năng: lặp trong khi điều_kiện còn đúng, kiểm tra điều_kiện trước mỗi lần vào khối lệnh. Dừng khi không biết trước số lần lặp.

Chương trình minh họa các cấu trúc điều kiện và vòng lặp:

```
using System;
class Program
{
    static void Main()
    {
        // 1. if...else
        int x = -2;
        if (x > 0)
            Console.WriteLine("x dương");
        else if (x == 0)
            Console.WriteLine("x bằng 0");
        else
```

```

        Console.WriteLine("x âm");
// 2. switch
char grade = 'B';
switch (grade)
{
    case 'A': Console.WriteLine("Xuất sắc"); break;
    case 'B': Console.WriteLine("Khá"); break;
    default: Console.WriteLine("Khác"); break;
}
// 3. ternary
string parity = (x % 2 == 0) ? "Chẵn" : "Lẻ";
Console.WriteLine($"x là số {parity}");
// 4. for
Console.Write("for: ");
for (int i = 1; i <= 3; i++)
    Console.WriteLine(i);
// 5. foreach
Console.Write("foreach: ");
int[] arr = { 5, 10 };
foreach (int v in arr)
    Console.WriteLine(v);
// 6. while
Console.Write("while: ");
int j = 1;
while (j <= 2)
{
    Console.WriteLine(j);
    j++;
}
Console.WriteLine();
// 7. do..while
Console.Write("do_while: ");
int k = 1;
do
{
    Console.WriteLine(k);
    k++;
} while (k <= 1);
Console.WriteLine();
}
}

```

Kết quả chương trình:

```

x âm
Khá
x là số Chẵn
for: 123
foreach: 5 10
while: 12
do_while: 1

```

e. Truyền tham trị và truyền tham chiếu

- Truyền tham trị (Pass by Value)

- Khi truyền tham trị, C# truyền sao chép giá trị của biến vào tham số hàm.
- Các thay đổi trong hàm không làm ảnh hưởng đến biến gốc.

Chương trình minh họa:

```
using System;
class Program
{
    static void Increment(int x)
    {
        x = x + 1;
        Console.WriteLine("Trong hàm Increment: x = " + x);
    }
    static void Main()
    {
        int a = 5;
        Console.WriteLine("Trước khi gọi: a = " + a);
        Increment(a);
        Console.WriteLine("Sau khi gọi: a = " + a);
    }
}
```

- Biến a = 5 được sao chép vào x.
- Trong hàm Increment, x tăng thành 6 nhưng chỉ là bản sao.
- Ra khỏi hàm Main, a vẫn là 5.

Kết quả chương trình:

```
Trước khi gọi: a = 5
Trong hàm Increment: x = 6
Sau khi gọi: a = 5
```

- Truyền tham chiếu (Pass by Reference) với **ref**

- Trong C#, ta sử dụng từ khóa **ref** để truyền địa chỉ của biến.
- Các thay đổi trong hàm không làm ảnh hưởng biến gốc.

Chương trình minh họa:

```
using System;
class Program
{
    static void Swap(ref int p, ref int q)
    {
        int tmp = p;
        p = q;
        q = tmp;
    }
    static void Main()
    {
        int x = 3, y = 7;
        Console.WriteLine($"Trước Swap: x = {x}, y = {y}");
        Swap(ref x, ref y);
        Console.WriteLine($"Sau Swap: x = {x}, y = {y}");
    }
}
```

- Gọi Swap(**ref** x, **ref** y), p và q trở thành x, y.
- Hoán đổi qua p/q → trực tiếp thay đổi x, y.

Kết quả chương trình:

Trước Swap: x = 3, y = 7
Sau Swap: x = 7, y = 3

2. Cấu trúc lớp (class)

Class là khuôn mẫu (blueprint) để tạo đối tượng (object). Class bao gồm các thành phần:

- Trường (fields): lưu dữ liệu nội bộ.
- Thuộc tính (properties): đóng gói trường, dùng để truy xuất an toàn.
- Phương thức (methods): hành vi của đối tượng.
- Constructor: hàm khởi tạo.
- Phạm vi truy cập: `public`, `private`, `protected`, `internal`,...

Chương trình minh họa:

```
using System;
namespace DemoApp
{
    // Định nghĩa lớp Person
    public class Person
    {
        // Trường (private)
        private string name;
        private int age;

        // Constructor
        public Person(string name, int age)
        {
            this.name = name;
            this.age = age;
        }

        // Property cho phép đọc/ghi name
        public string Name
        {
            get { return name; }
            set { name = value; }
        }

        // Phương thức in thông tin
        public void Introduce()
        {
            Console.WriteLine($"Tôi tên là {Name}, {age} tuổi.");
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            // Tạo đối tượng Person
            Person p = new Person("Khôi", 22);
            // Gọi phương thức introduce
            p.Introduce();
        }
    }
}
```

- `public class Person`: lớp có thể truy cập từ bên ngoài `namespace`.
- Trường `name`, `age` để lưu dữ liệu; đặt `private` để bảo vệ tính đóng gói.
- Constructor `Person(string name, int age)`: gán giá trị cho trường khi khởi tạo.
- Property `Name` để cho phép đọc/ghi an toàn, có thể kiểm tra logic trong `get/set`.
- Method `Introduce()`: đóng gói hành vi in ra console.

Kết quả chương trình: Tôi tên là Khôi, 22 tuổi.

3. Cấu trúc phương thức (method)

Một phương thức trong C# có cấu trúc chung:

```
[<access_modifier>][static] <return_type> <MethodName> ([parameters])
{
    // phần thân phương thức
    [return < value >;]
}
```

- `access_modifier`: `public`, `private`, ...
- `static`: gọi trực tiếp qua tên lớp, không cần instance.
- `return_type`: kiểu dữ liệu trả về (`void` nếu không trả về kiểu dữ liệu nào).
- `MethodName`: tên phương thức, thường dùng kiểu PascalCase.
- `parameters`: danh sách các tham số.

Chương trình minh họa:

```
using System;
namespace DemoApp
{
    class Calculator
    {
        // Phương thức cộng hai số, trả về kết quả
        public int Add(int a, int b)
        {
            return a + b;
        }
        // Phương thức in kết quả; static nên gọi trực tiếp
        public static void PrintResult(int result)
        {
            Console.WriteLine("Kết quả: " + result);
        }
    }
    class Program
    {
        static void Main()
        {
            Calculator calc = new Calculator(); // tạo object
            int sum = calc.Add(3, 5);           // gọi method instance
            Calculator.PrintResult(sum);         // gọi method static
        }
    }
}
```

- `public int Add(int a, int b):`
 - `public`: ai cũng gọi được.
 - `int`: trả về số nguyên.
 - `(int a, int b)`: hai tham số kiểu `int`.
 - `return a + b;` trả kết quả tổng.
- `public static void PrintResult(int result):`
 - `static`: có thể gọi `Calculator.PrintResult(...)` mà không cần `new`.
 - `void`: không trả về gì, chỉ thực hiện in.
- Trong hàm `Main()`:
 - `new Calculator()` khởi tạo đối tượng để gọi phương thức không `static`.
 - `Calculator.PrintResult(sum);` gọi thẳng phương thức `static`.

Kết quả chương trình: Kết quả: 8

4. Cấu trúc hàm khởi tạo (constructor)

Constructor là phương thức đặc biệt của lớp, dùng để khởi tạo giá trị ban đầu cho các trường (fields) khi tạo đối tượng. Constructor có các đặc điểm:

- Tên constructor trùng với tên lớp, không có kiểu trả về.
- Có thể nạp chồng (overload) nhiều constructor với các tham số khác nhau.
- Nếu không định nghĩa constructor nào thì C# sẽ cung cấp mặc định một constructor không tham số (default constructor).

Chương trình minh họa:

```
using System;

namespace DemoApp
{
    public class Rectangle
    {
        // Fields
        private double width;
        private double height;

        // Constructor không tham số: khởi tạo kích thước mặc định
        public Rectangle()
        {
            width = 1.0;
            height = 1.0;
        }
    }
}
```



```

// Constructor có tham số: khởi tạo theo giá trị truyền vào
public Rectangle(double w, double h)
{
    width = w;
    height = h;
}
// Method tính diện tích
public double Area()
{
    return width * height;
}
}
class Program
{
    static void Main()
    {
        // Dùng constructor mặc định
        Rectangle r1 = new Rectangle();
        Console.WriteLine($"r1: {r1.Area()}");
        // Dùng constructor có tham số
        Rectangle r2 = new Rectangle(3.5, 2.0);
        Console.WriteLine($"r2: {r2.Area()}");
    }
}
}

```

- `public Rectangle() {...}`
 - Constructor không tham số, khởi tạo `width` và `height` đều bằng 1.
- `public Rectangle(double w, double h) {...}`
 - Constructor có hai tham số, cho phép tạo hình chữ nhật với kích thước do người dùng cung cấp.
- Trong hàm `Main()`, khi gọi `new Rectangle()` không truyền gì thì C# dùng constructor không tham số.
- Khi gọi `new Rectangle(3.5, 2.0)`, C# chọn constructor phù hợp về số và kiểu tham số, gán `width = 3.5`, `height = 2.0`.

Kết quả chương trình:

```

r1: 1
r2: 7

```

5. Cấu trúc thuộc tính (properties)

Property là thành phần của lớp dùng để đóng gói các trường, cho phép kiểm soát việc đọc hoặc ghi dữ liệu. Cú pháp của property gồm getter (thuộc tính chỉ đọc) và setter (thuộc tính ghi).

Ta có thể thực hiện:

- Định nghĩa full property với logic trong `get/set`.
- Dùng auto-property (`public int x { get; set; }`) để C# tự sinh trường ẩn và getter/setter đơn giản.
- Đặt giới hạn chỉ cho phép `get` (chỉ đọc) hoặc `private set` (chỉ lớp nội bộ mới gán).

Chương trình minh họa:

```
using System;
namespace DemoApp
{
    public class Student
    {
        // Auto-property: C# tự sinh private field ẩn
        public string Name { get; set; }
        // Full property: kiểm tra logic trong setter
        private int age;
        public int Age
        {
            get
            {
                return age;
            }
            set
            {
                // Chỉ cho phép gán độ tuổi >= 0
                if (value < 0)
                    throw new ArgumentException("Tuổi không được là số âm!");
                age = value;
            }
        }
        // Read-only property: chỉ getter
        public bool IsAdult
        {
            get
            {
                return age >= 18;
            }
        }
    }
}

class Program
{
    static void Main()
    {
        Student s = new Student();
        // Dùng auto-property
        s.Name = "Khoi";
        Console.WriteLine($"Tên: {s.Name}");
        // Dùng full property với kiểm tra
        s.Age = 22;
        Console.WriteLine($"Tuổi: {s.Age}");
        // Dùng read-only property
        Console.WriteLine($"Is adult? {s.IsAdult}");
        // Nếu gán tuổi âm sẽ ném exception:
        // Ví dụ: s.Age = -5;
    }
}
```

- `public string Name { get; set; }`
 - Auto-property: C# tự sinh trường ẩn, thành phần getter/setter mặc định là **public**.
- `public int Age { get { ... } set { ... } }`
 - Property đầy đủ, trong **set** kiểm tra value không âm, nếu vi phạm thì sẽ ném một **ArgumentException**.

- `public bool IsAdult { get { return age >= 18; } }`
 - Read-only property: chỉ có `get`, tính toán dựa trên `age`.
- Trong hàm `Main()` {...}
 - `s.Name = "Khoi"`; và `s.Age = 22`; lần lượt gọi setter.
 - `s.IsAdult` gọi getter để kiểm tra.
 - Nếu cố gắng đưa giá trị `s.Age = -5`; sẽ gặp exception, đảm bảo dữ liệu luôn hợp lệ.

Kết quả chương trình:

```
Tên: Khoi
Tuổi: 22
Is adult? True
```

III. Các tính chất của lập trình hướng đối tượng

1. Tính đóng gói (Encapsulation)

a. Định nghĩa

Đóng gói là việc giấu thông tin chi tiết về cách một đối tượng hoạt động, chỉ cung cấp các giao diện công khai (public interface) để tương tác. Điều này giúp:

- Đơn giản hóa việc sử dụng class.
- Bảo vệ tính toàn vẹn của dữ liệu nội bộ.
- Hạn chế sự phụ thuộc và sửa lỗi ngoài ý muốn.

b. Nhiệm vụ của tính đóng gói

- `Console.WriteLine()`, `Console.ReadLine()` là hai ví dụ cơ bản trong nguyên lý đóng gói. Người sử dụng không cần quan tâm tới việc các hàm này được viết như thế nào. Họ chỉ cần quan tâm tới việc sử dụng. Ngoài ra, encapsulation cũng hỗ trợ cho việc bảo vệ dữ liệu/ trạng thái của đối tượng khỏi sự thay đổi của người dùng.
- Các từ khoá kiểm soát quyền truy cập như `private`, `protected`,... bảo đảm tính toàn vẹn của dữ liệu cũng như giấu chi tiết của class khỏi người dùng.
- Property bảo đảm việc đọc/ghi tới trường được kiểm soát nhờ đó đảm bảo được tính toàn vẹn của dữ liệu.
- Ngoài ra, C# còn các lệnh khác như `constant`, `readonly`,...

Chương trình minh họa:

```
using System;
class Student
{
    private string name; // Dữ liệu được đóng gói
    public string Name    // Property kiểm soát truy cập
    {
        get { return name; }
        set
        {
            if (value.Length > 20)
                Console.WriteLine("Tên không được dài quá 20 ký tự.");
            else
                name = value;
        }
    }
}
```

- `private string name;`
 - Biến `name` là biến nội bộ (`private`), không thể truy cập trực tiếp từ bên ngoài `class Student`. Đây chính là đóng gói dữ liệu, ẩn thông tin bên trong đối tượng.
- `public string Name {...}` (property)
 - Đây là giao diện công khai để đọc (`get`) hoặc ghi (`set`) giá trị cho `name`. Trong phần `set`, có điều kiện kiểm tra: nếu chuỗi vượt quá 20 ký tự thì không cho gán giá trị và thông báo lỗi. Điều này giúp kiểm soát và bảo vệ dữ liệu, đảm bảo `name` luôn hợp lệ.

2. Tính kế thừa (Inheritance)

a. Định nghĩa

Tính kế thừa là cơ chế cho phép class con (derived class) kế thừa các thành phần (fields, methods, properties,...) từ class cha (base class).

- Giúp tái sử dụng mã nguồn, giảm trùng lặp.
- Class con có thể mở rộng hoặc ghi đè các phương thức từ class cha.
- Trong C#, tính kế thừa chỉ hỗ trợ kế thừa đơn (mỗi class chỉ kế thừa từ một class khác).

b. Nhiệm vụ của tính kế thừa

- Trong C#, một class chỉ có thể kế thừa từ một class duy nhất.
- Ta có thể dùng từ khóa `sealed` trong khai báo class để cấm việc kế thừa ở các class khác.
- Class con kế thừa hầu hết các phần tử của base class (properties, methods,...) ngoại trừ constructor.
- Quyền truy cập trong kế thừa:
 - `public`: thành viên khai báo là `public` có thể truy cập từ bất kì đâu, cả trong class gốc, class con và bên ngoài.

- **private**: thành viên khai báo là **private** chỉ truy cập được bên trong chính class đó. Class con không thể truy cập trực tiếp tới phần tử **private** của class cha.
- **protected**: thành viên khai báo là **protected** có thể truy cập trong chính class khai báo và tất cả các class kế thừa (class con), nhưng không truy cập được từ bên ngoài.

Chương trình minh họa:

```
using System;
class Car
{
    public readonly int maxSpeed;    // Tốc độ tối đa (read-only)
    private int currSpeed;          // Tốc độ hiện tại (chỉ dùng bên trong class)
    // Constructor có tham số
    public Car(int max)
    {
        maxSpeed = max;
    }
    // Constructor mặc định
    public Car()
    {
        maxSpeed = 55;
    }
    // Property để truy cập tốc độ hiện tại
    public int Speed
    {
        get { return currSpeed; }
        set
        {
            currSpeed = value;
            if (currSpeed > maxSpeed)
                currSpeed = maxSpeed;
        }
    }
}
// Class con kế thừa từ Car
class MiniVan : Car
{
    // MiniVan kế thừa tất cả các thành phần public từ Car
}
```

```
static void Main(string[] args)
{
    MiniVan myVan = new MiniVan();    // Tạo đối tượng từ class con
    myVan.Speed = 10;                 // Truy cập property Speed từ class cha
    Console.WriteLine("My van is going {0} MPH", myVan.Speed);

    myVan.Speed = 100;                // Vượt quá maxSpeed
    Console.WriteLine("My van is going {0} MPH", myVan.Speed);

    Console.ReadLine();
}
```

- **class Car** là class cha (base class):
 - Định nghĩa các thuộc tính chung như maxSpeed, currSpeed và property Speed.

- Có hai constructor: một mặc định và một có tham số.
- `class MiniVan` kế thừa từ `Car`:
 - Sử dụng cú pháp `MiniVan : Car`.
 - Không cần viết lại các thuộc tính và phương thức của `Car`, vì chúng đã được kế thừa.
 - Có thể mở rộng thêm phương thức hoặc thuộc tính riêng nếu muốn.
- Trong hàm `Main()`:
 - Tạo đối tượng `MiniVan`.
 - Gán giá trị cho `Speed` thông qua property kế thừa từ `Car`.
 - Nếu gán `Speed` vượt quá `maxSpeed`, nó sẽ tự động bị giới hạn.

c. Constructor trong class con

Class con không thừa hưởng constructor. Khi khởi tạo object, các phần tử được khai báo ở base class sẽ được khởi tạo bằng constructor của base class.

Chương trình minh họa:

```
// Class cha (base class)
using System;
class Employee
{
    public string Name { get; set; }
    public int ID { get; set; }
    public float Pay { get; set; }
    // Constructor của class cha
    public Employee(string name, int id, float pay)
    {
        Name = name;
        ID = id;
        Pay = pay;
    }
    public void DisplayInfo()
    {
        Console.WriteLine("Name: {0}, ID: {1}, Pay: {2}", Name, ID, Pay);
    }
}

// Class con kế thừa từ Employee
class Manager : Employee
{
    public int StockOptions { get; set; }
    // Constructor của class con, dùng từ khóa : base để gọi constructor của class cha
    public Manager(string name, int id, float pay, int stockOptions)
        : base(name, id, pay) // Gọi constructor của class cha
    {
        StockOptions = stockOptions; // Khởi tạo thêm thuộc tính riêng của class con
    }
    public void DisplayManagerInfo()
    {
        DisplayInfo(); // Gọi hàm từ class cha
        Console.WriteLine("Stock Options: {0}", StockOptions);
    }
}
```

- Class cha **Employee**
 - Có 3 thuộc tính: Name, ID, Pay.
 - Có một constructor có tham số để khởi tạo các thuộc tính này.
 - Có thêm hàm `DisplayInfo()` để in ra thông tin nhân viên.
- Class con **Manager** kế thừa từ **Employee**
 - Kế thừa các thuộc tính và phương thức từ **Employee**.
 - Có thêm một thuộc tính riêng là `StockOptions`.
 - Constructor của **Manager** sử dụng từ khóa : `base(...)` để gọi constructor của **Employee**, giúp khởi tạo các thuộc tính được thừa hưởng (Name, ID, Pay).

3. Tính đa hình (Polymorphic)

a. Định nghĩa

Tính đa hình là cho phép nhiều đối tượng từ các class con cùng kế thừa từ một class cha, nhưng hành xử khác nhau khi gọi cùng một phương thức. Tính đa hình giúp dễ dàng mở rộng hệ thống mà không cần sửa code hiện tại, được thực hiện trong C# qua **virtual** (ở class cha), **override** (ở class con) và thông qua tham chiếu kiểu class cha (base class).

b. Nhiệm vụ của tính đa hình

Chương trình minh họa:

```
// Class cha
using System;
class Animal
{
    public virtual void Speak()
    {
        Console.WriteLine("Animal makes a sound");
    }
}

// Class con 1
class Dog : Animal
{
    public override void Speak()
    {
        Console.WriteLine("Dog barks");
    }
}

// Class con 2
class Cat : Animal
{
    public override void Speak()
    {
        Console.WriteLine("Cat meows");
    }
}
```

```

static void Main(string[] args)
{
    // Mảng các đối tượng kiểu Animal
    Animal[] animals = { new Dog(), new Cat(), new Animal() };

    // Gọi phương thức Speak() theo cách đa hình
    foreach (Animal a in animals)
    {
        a.Speak();
    }

    Console.ReadLine();
}

```

- **Animal** là class cha, định nghĩa phương thức **Speak()** với từ khóa **virtual**. Điều này cho phép các class con ghi đè phương thức này bằng hành vi riêng.
- **Dog** và **Cat** là class con, kế thừa từ **Animal** và dùng từ khóa **override** để định nghĩa lại hành vi riêng của **Speak()**:
 - **Dog.Speak()** in ra **"Dog barks"**
 - **Cat.Speak()** in ra **"Cat meows"**
- Trong hàm **Main()**
 - Tạo một mảng chứa các đối tượng **Animal**, nhưng thực tế mỗi phần tử là **Dog**, **Cat**, và **Animal**.
 - Khi gọi **a.Speak()** trong vòng lặp **foreach**, chương trình sẽ tự động xác định đúng phiên bản phương thức cần gọi dựa vào kiểu thực tế của đối tượng.

c. Chuyển đổi kiểu dữ liệu giữa class

- Khi chuyển đổi kiểu dữ liệu giữa của object, ta sử dụng từ khóa **as**.
- Nếu chỉ cần kiểm tra kiểu dữ liệu của object, ta có thể dùng từ khóa **is**.

Chương trình minh họa:

```

using System;
class Shape
{
    public string Name { get; set; }
    public Shape(string name = "NoName")
    {
        Name = name;
    }
    public virtual void Draw()
    {
        Console.WriteLine("Drawing a shape");
    }
}
class Circle : Shape
{
    public Circle(string name) : base(name) { }
}

```



```

public override void Draw()
{
    Console.WriteLine("Drawing a circle: " + Name);
}
public void RadiusInfo()
{
    Console.WriteLine("Radius is 10 units.");
}
}

```

```

static void Main(string[] args)
{
    Shape s = new Circle("Circle A"); // Upcasting: Circle → Shape
    s.Draw();                          // Gọi hàm override trong Circle

    // Downcasting: Ép kiểu ngược lại
    Circle c = s as Circle;
    if (c != null)
    {
        c.RadiusInfo(); // Gọi hàm riêng của Circle
    }

    // Kiểm tra kiểu thực tế
    if (s is Circle)
    {
        Console.WriteLine("s is actually a Circle object");
    }

    Console.ReadLine();
}

```

d. Từ khóa virtual & override

- Từ khóa **virtual** dùng trong class cha để cho phép class con ghi đè (**override**) phương thức đó.
- Từ khóa **override** dùng trong class con để ghi đè lại phương thức **virtual** từ class cha.

Chương trình minh họa:

```

using System;
class Animal
{
    public virtual void Speak()
    {
        Console.WriteLine("Animal speaks...");
    }
}

class Dog : Animal
{
    public override void Speak()
    {
        Console.WriteLine("Dog barks");
    }
}

```

```

class Cat : Animal
{
    public override void Speak()
    {
        Console.WriteLine("Cat meows");
    }
}

```

4. Tính trừu tượng (Abstraction)

- Nếu ta không muốn người dùng tạo object từ base classes, ta có thể chuyển các class này thành **abstract class**.
- Để bảo đảm tất cả các class con đều phải viết lại một hoặc nhiều method của base class, ta có thể dùng **abstract method**.
- **abstract method** là các hàm chỉ được khai báo trong base class, ngoài ra không có nội dung.

Chương trình minh họa:

```

// Lớp trừu tượng
using System;
abstract class Shape
{
    public string Name { get; set; }
    public Shape(string name)
    {
        Name = name;
    }

    // Phương thức trừu tượng - không có phần thân
    public abstract void Draw();
}

// Lớp con kế thừa và định nghĩa lại phương thức Draw()
class Circle : Shape
{
    public Circle(string name) : base(name) { }
    public override void Draw()
    {
        Console.WriteLine("Drawing a circle: " + Name);
    }
}

// Lớp con khác
class Rectangle : Shape
{
    public Rectangle(string name) : base(name) { }
    public override void Draw()
    {
        Console.WriteLine("Drawing a rectangle: " + Name);
    }
}

```

- **Shape** là một class trừu tượng (**abstract class**):
 - Không thể tạo object trực tiếp từ **Shape**.
 - Khai báo một phương thức trừu tượng **Draw()**, chưa có phần thân nên yêu cầu tất cả các lớp con phải định nghĩa lại phương thức này.

IV. Các quy tắc Clean Code

1. Quy tắc đặt tên (name)

- Rõ ràng và dễ hiểu: tên biến, tên phương thức, tên lớp cần phải diễn tả chính xác mục đích.
- Đảm bảo tính nhất quán theo chuẩn của ngôn ngữ, ví dụ: PascalCase cho class, camelCase cho biến và tham số, UPPER_CASE cho hằng.
- Tránh viết tắt, trừ những thuật ngữ quen thuộc như “ID”, “XML”,...

Chương trình minh họa:

```
// Tên lớp: PascalCase
public class OrderProcessor
{
    // Tên biến/field private: _camelCase
    private decimal _discountRate;

    // Tên hằng: UPPER_CASE
    private const int MAX_RETRY = 3;

    // Tên phương thức: PascalCase, mô tả hành động
    public void ProcessOrder(Order order)
    {
        // Tên tham số: camelCase
        decimal finalPrice = CalculateFinalPrice(order.Amount);
        // ...
    }
    // Tính toán giá sau chiết khấu
    private decimal CalculateFinalPrice(decimal amount)
    {
        return amount * (1 - _discountRate);
    }
}
```

2. Quy tắc viết hàm (method) và lớp (class)

- Đảm bảo mỗi class hay method chỉ đảm nhận một nhiệm vụ duy nhất (single responsibility principle).
- Viết method ngắn và rõ ràng: phần code bên trong một method nên có độ dài từ 20 đến 30 dòng để dễ đọc nhanh. Nếu chương trình phức tạp thì có thể tách ra thành các helper method nhỏ hơn.
- Viết class gọn và chuyên biệt: các method trong cùng một class phải liên quan chặt chẽ với một mục đích chung, tránh đưa vào một class những method không cùng chủ đề.

Chương trình minh họa:

```
// Thiếu clean code: phương thức này vừa gửi email, vừa ghi log, vừa xác thực dữ liệu
using System;
public class BadNotificationService
{
    public void Notify(User user, string message)
```

```

{
    // xác thực
    if (string.IsNullOrEmpty(user.Email)) throw new ArgumentException();

    // gửi email
    EmailSender.Send(user.Email, message);

    // ghi log
    File.AppendAllText("log.txt", $"Sent to {user.Email}\n");
}
}

```

```

// Clean code: tách thành các class/method riêng
public class UserValidator
{
    public void ValidateHasEmail(User user)
    {
        if (string.IsNullOrEmpty(user.Email))
            throw new ArgumentException("User must have an email");
    }
}

public class EmailNotifier
{
    public void SendEmail(string email, string message)
    {
        EmailSender.Send(email, message);
    }
}

public class Logger
{
    public void Log(string text)
    {
        File.AppendAllText("log.txt", text + Environment.NewLine);
    }
}

public class NotificationService
{
    private readonly UserValidator _validator = new();
    private readonly EmailNotifier _notifier = new();
    private readonly Logger _logger = new();
    public void Notify(User user, string message)
    {
        _validator.ValidateHasEmail(user);           // 1 nhiệm vụ: xác thực
        _notifier.SendEmail(user.Email, message);    // 1 nhiệm vụ: gửi
        _logger.Log($"Sent to {user.Email}");        // 1 nhiệm vụ: log
    }
}

```

3. Quy tắc chú thích (comment)

- Chỉ comment khi thật cần thiết: giải thích “tại sao” (why), không comment “cái gì” (what) – phần này đã thể hiện rõ ý nghĩa trong code.
- Cập nhật các comment khi code thay đổi, tránh các comment cũ có thể gây hiểu nhầm trong chương trình.
- Dùng XML comments (///) cho public API để IDE tự hiển thị.

```
public class Calculator
{
    // BAD: Comment “what” – không cần thiết
    // Tính bình phương của số
    public int Square_Bad(int x)
    {
        return x * x;
    }

    // GOOD: Dùng XML comment cho public API, mô tả “why” nếu cần
    /// <summary>
    /// Tính bình phương của số nguyên.
    /// </summary>
    /// <param name="x">Số cần bình phương</param>
    /// <returns>Giá trị x x x</returns>
    public int Square(int x)
    {
        // Không cần comment "return x * x" vì đơn giản và rõ ràng
        return x * x;
    }

    // GOOD: Comment “why” nếu logic không hiển nhiên
    /// <summary>
    /// Tính tổng các số trong mảng, bỏ qua phần tử null.
    /// </summary>
    public int Sum(int?[] values)
    {
        int total = 0;
        foreach (var v in values)
        {
            // Nếu gặp null, bỏ qua để tránh lỗi NullReference
            if (v.HasValue)
                total += v.Value;
        }
        return total;
    }
}
```

CHƯƠNG 3: QUÁ TRÌNH THỰC HIỆN ĐỒ ÁN

I. Quá trình thực hiện đồ án

1. Thiết kế Form Login và BackEnd Form Login

a. Thiết kế Form Login



Hình 11: Thiết kế giao diện Đăng nhập (Form Login)

b. Thiết kế BackEnd Form Login

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using static System.Net.Mime.MediaTypeNames;

namespace ChuongTrinhquanlykhachsan
{
    public partial class Form1 : Form
    {
        function function = new function();
        string query;
        public Form1()
        {
            InitializeComponent();

            private void guna2Panel1_Paint(object sender, PaintEventArgs e)
            {

            }

            private void label2_Click(object sender, EventArgs e)
            {

            }

            private void btnExit_Click(object sender, EventArgs e)
```

```

        {
            Application.Exit();
        }
        private void btnLogin_Click(object sender, EventArgs e)
        {
            query = "select username,pass from employee where username= '" +
txtUsername.Text + "' and pass = '" + txtPassword.Text + "'";
            DataSet ds = function.getData(query);
            if (ds.Tables[0].Rows.Count != 0)
            {
                labelError.Visible = false;
                Dashboard dash = new Dashboard();
                this.Hide();
                dash.Show();
            }
            else
            {
                labelError.Visible = true;
                txtPassword.Clear();
            }
        }


        private void txtUsername_TextChanged(object sender, EventArgs e)
        {
        }
    }
}

```

- Chức năng :

- Cho phép người dùng nhập Username và Password.
- Kiểm tra thông tin đăng nhập.
- Nếu đúng: ẩn Form Login, mở Form Dashboard.
- Nếu sai: hiện thông báo lỗi.

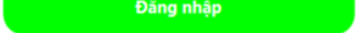
- Giải thích code:

- Thiết lập nút Exit  :

```

private void btnExit_Click(object sender, EventArgs e)
{
    Application.Exit(); //Thoát chương trình khi bấm nút Exit
}

```

- Thiết lập nút “Đăng nhập”  :

```

        private void btnLogin_Click(object sender, EventArgs e)
        {
            // Tạo câu lệnh SQL kiểm tra tài khoản
            query = "select username,pass from employee where username= '" +
txtUsername.Text + "' and pass = '" + txtPassword.Text + "'";
            DataSet ds = function.getData(query);

```

```

if (ds.Tables[0].Rows.Count != 0) // Nếu có dữ liệu => đúng tài khoản
{
    labelError.Visible = false; // Ẩn thông báo lỗi
    Dashboard dash = new Dashboard(); // Tạo Form Dashboard
    this.Hide(); // Ẩn Form Login
    dash.Show(); // Hiện Dashboard
}
Else // Sai tài khoản
{
    labelError.Visible = true; // Hiện thông báo lỗi
    txtPassword.Clear(); // Xóa Password nhập sai
}
}

```

- *Liên hệ kiến thức:*

- Form1 là class kế thừa Form (Inheritance).
- Dùng constructor public Form1() để khởi tạo.
- Có method btnLogin_Click, btnExit_Click xử lý sự kiện (Event handler).
- Biến function function = new function(); là field.
- Gọi function.getData(query) → áp dụng Encapsulation: function che giấu logic CSDL.
- Truyền dữ liệu qua biến query (Pass by Value)
- Xử lý điều kiện if-else để điều khiển giao diện → Logic điều kiện.

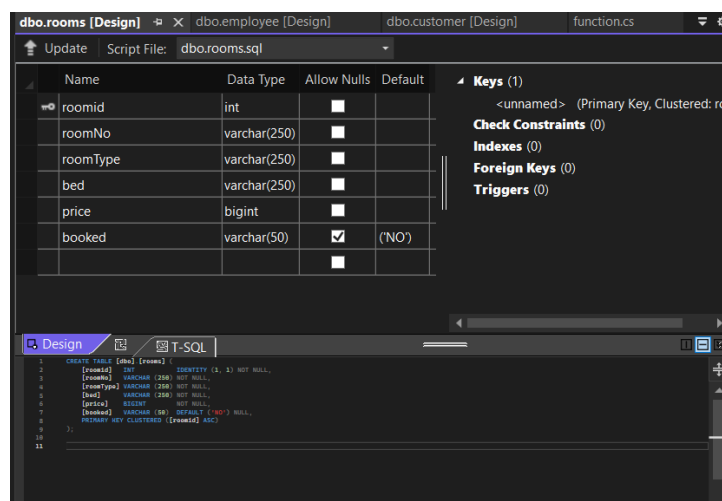
- *Kiến thức áp dụng:*

- Nền tảng WinForm.
- Class, Field, Constructor, Method.
- Inheritance (Form1 kế thừa Form).
- Encapsulation (class function).
- Pass by Value.

2. Thiết kế Form DashBoard và BackEnd Form DashBoard

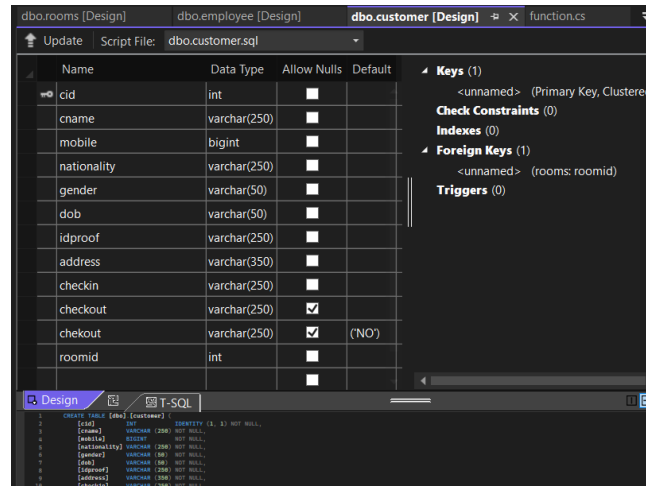
a. Thiết kế Form DashBoard

- Database cho thông tin về phòng:



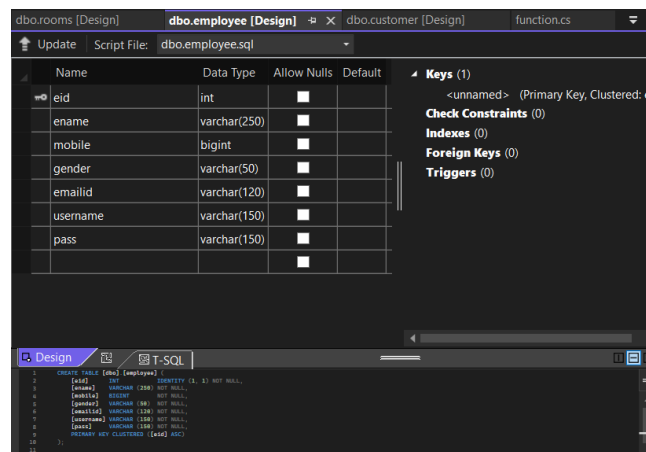
Hình 12: Database cho thông tin về phòng (dbo.rooms)

- Database cho thông tin về khách hàng:



Hình 13: Database cho thông tin về khách hàng (dbo.customer)

- Database cho thông tin về Nhân viên:



Hình 14: Database cho thông tin về nhân viên (dbo.employee)

Thêm Phòng

Đăng ký khách hàng

Thanh toán

Chi tiết khách hàng

Nhân viên

Thêm Phòng

	Roomid	Roomname	Roomtype	Roomrate	Roomstatus
1	123	Ac	Double	23	NO
2	5	Non_Ac	Single	1234	YES
3	10	Ac	Single	120000	YES
4	620	Non_Ac	Triple	300000	YES
5	6	Ac	Single	330000	YES

Số Phòng

Loại Phòng

Loại Giường

Giá Tiền

Thêm Phòng

b. Thiết kế BackEnd Form DashBoard

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using static System.Net.Mime.MediaTypeNames;

namespace ChuongTrinhquanlykhachsan
{
    public partial class Dashboard : Form
    {
        public Dashboard()
        {
            InitializeComponent();
        }
        private void panel2_Paint(object sender, PaintEventArgs e)
        {
        }
        private void guna2Panel1_Paint(object sender, PaintEventArgs e)
        {
        }
        private void guna2Button3_Click(object sender, EventArgs e)
        {
        }
        private void btnExit_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }
        private void Dashboard_Load(object sender, EventArgs e)
        {
            uC_AddRoom1.Visible = false;
            uC_CustomerRes1.Visible = false;
            uC_CheckOut1.Visible = false;
            uC_CustomerDetails1.Visible = false;
            uC_Employee1.Visible = false;
            btnAddRoom.PerformClick();
        }
        private void uC_AddRoom1_Load(object sender, EventArgs e)
        {
        }
        private void btnCustomerRes_Click(object sender, EventArgs e)
        {
            PanelMoving.Left = btnCustomerRes.Left + 60;
            uC_CustomerRes1.Visible = true;
            uC_CustomerRes1.BringToFront();
        }
        private void btnAddRoom_Click(object sender, EventArgs e)
        {
            PanelMoving.Left = btnAddRoom.Left + 50;
            uC_AddRoom1.Visible = true;
            uC_AddRoom1.BringToFront();
        }
    }
}
```

```


private void btnCheckOut_Click(object sender, EventArgs e)
{
    PanelMoving.Left = btnCheckOut.Left + 60;
    uC_CheckOut1.Visible = true;
    uC_CheckOut1.BringToFront();
}
private void btnCustomerDetail_Click(object sender, EventArgs e)
{
    PanelMoving.Left = btnCustomerDetail.Left + 60;
    uC_CustomerDetails1.Visible = true;
    uC_CustomerDetails1.BringToFront();
}
private void btnEmployee_Click(object sender, EventArgs e)
{
    PanelMoving.Left = btnEmployee.Left + 60;
    uC_Employee1.Visible = true;
    uC_Employee1.BringToFront();
}
}
}

```

- Chức năng :

- DashBoard là giao diện chính sau khi đăng nhập thành công.
- Gồm các nút quản lý: Add Room (Thêm phòng), Customer Reservation (Đăng ký khách hàng), Check Out (Thanh toán) , Customer Details (Chi tiết khách hàng), Employee (Nhân viên).
- Hiện thị UserControl tương ứng.

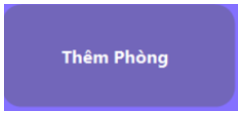
- Giải thích code:

- Thiết lập nút Exit  :

```

private void btnExit_Click(object sender, EventArgs e)
{
    Application.Exit(); //Thoát chương trình khi bấm nút Exit
}

```

- Thiết lập nút “Thêm phòng”  :

```

private void btnAddRoom_Click(object sender, EventArgs e)
{
    PanelMoving.Left = btnAddRoom.Left + 50; // Di chuyển Panel báo vị trí nút
    uC_AddRoom1.Visible = true; // Hiện UserControl AddRoom
    uC_AddRoom1.BringToFront(); // Đưa lên trên
}

```

Đăng ký khách hàng

- Thiết lập nút “Đăng ký khách hàng” :

```
nút private void btnCustomerRes_Click(object sender, EventArgs e)
{
    PanelMoving.Left = btnCustomerRes.Left + 60; // Di chuyển Panel báo vị trí
    uC_CustomerRes1.Visible = true; // Hiện UserControl CustomerRes
    uC_CustomerRes1.BringToFront(); // Đưa lên trên
}
```

Thanh toán

- Thiết lập nút “Thanh toán” :

```
private void btnCheckOut_Click(object sender, EventArgs e)
{
    PanelMoving.Left = btnCheckOut.Left + 60; // Di chuyển Panel báo vị trí nút
    uC_CheckOut1.Visible = true; // Hiện UserControl CheckOut
    uC_CheckOut1.BringToFront(); // Đưa lên trên
}
```

Chi tiết khách hàng

- Thiết lập nút “Chi tiết khách hàng” :

```
trí nút private void btnCustomerDetail_Click(object sender, EventArgs e)
{
    PanelMoving.Left = btnCustomerDetail.Left + 60; // Di chuyển Panel báo vị
    uC_CustomerDetails1.Visible = true; // Hiện UserControl CustomerDetail
    uC_CustomerDetails1.BringToFront(); // Đưa lên trên
}
```

Nhân viên

- Thiết lập nút “Nhân viên” :

```
private void btnEmployee_Click(object sender, EventArgs e)
{
    PanelMoving.Left = btnEmployee.Left + 60; // Di chuyển Panel báo vị trí nút
    uC_Employee1.Visible = true; // Hiện UserControl Employee
    uC_Employee1.BringToFront(); // Đưa lên trên
}
```

- Liên hệ kiến thức:

- Dashboard cũng là class, kế thừa Form (Inheritance).
- Dùng constructor public Dashboard().
- Các method như btnAddRoom_Click thể hiện Method, Event handler.

- Gọi BringToFront(), thay đổi giao diện động.
 - Không có xử lý ngoại lệ ở đây, chỉ điều khiển giao diện.
- *Kiến trúc áp dụng:*
- Nền tảng WinForm.
 - Class, Constructor, Method.
 - Inheritance (Dashboard kế thừa Form).

3. Thiết kế Form Add Room và BackEnd Form Add Room

a. Thiết kế Form Add Room

Hình 16: Thiết kế giao diện “Thêm Phòng” (Form Add Room)

b. Thiết kế BackEnd Form Add Room

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace ChuongTrinhquanlykhachsan.All_User
{
    public partial class UC_AddRoom : UserControl
    {
        function fn = new function();
        String query;
        public UC_AddRoom()
        {
            InitializeComponent();
        }
    }
}
```

```

    }
    private void guna2CustomGradientPanel1_Paint(object sender, PaintEventArgs e)
    {
    }
    private void UC_AddRoom_Load(object sender, EventArgs e)
    {
        query = "select * from rooms";
        DataSet ds = fn.getData(query);
        DataGridView2.DataSource = ds.Tables[0];
    }
    private void btnAddRoom_Click(object sender, EventArgs e)
    {
        if (txtRoomNo.Text != "" && txtRoomType.Text != "" && txtBed.Text != "" &&
txtPrice.Text != "")
        {
            String roomno = txtRoomNo.Text;
            String type = txtRoomType.Text;
            String bed = txtBed.Text;
            Int64 price = Int64.Parse(txtPrice.Text);
            query = "insert into rooms (roomNo,roomType,bed,price) values ('" +
roomno + "','" + type + "','" + bed + "','" + price + ")";
            fn.setData(query, "Đã Thêm Phòng");
            UC_AddRoom_Load(this, null);
            clearALL();
        }
        else
        {
            MessageBox.Show("Xin vui Lòng điền đầy đủ thông tin ", "Warning !",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
        }
    }
    public void clearALL()
    {
        txtRoomNo.Clear();
        txtRoomType.SelectedIndex = -1;
        txtBed.SelectedIndex = -1;
        txtPrice.Clear();
    }
    private void UC_AddRoom_Leave(object sender, EventArgs e)
    {
        clearALL();
    }
    private void UC_AddRoom_Enter(object sender, EventArgs e)
    {
        UC_AddRoom_Load(this, null);
    }
    private void txtRoomType_SelectedIndexChanged(object sender, EventArgs e)
    {
    }
    private void DataGridView2_CellContentClick(object sender,
DataGridViewCellEventArgs e)
    {
    }
}

```

- Chức năng :
 - Thêm phòng mới.
 - Hiện thị danh sách phòng đang có.
- Giải thích code:

- Thiết lập sự kiện Load dữ liệu thêm phòng:

```
private void UC_AddRoom_Load(object sender, EventArgs e)
{
    query = "select * from rooms"; // Tạo câu SQL: select * from rooms → lấy toàn bộ phòng
    DataSet ds = fn.GetData(query); //Gọi fn.GetData() để chạy truy vấn, trả về DataSet
    DataGridView2.DataSource = ds.Tables[0]; //Gán kết quả vào DataGridView2 để hiển thị
    danh sách phòng
}
```

Thêm Phòng

- Thiết lập nút “Thêm phòng” :

```
private void btnAddRoom_Click(object sender, EventArgs e)
{
    if (txtRoomNo.Text != "" && txtRoomType.Text != "" && txtBed.Text != "" &&
    txtPrice.Text != "") //Khi bấm nút kiểm tra xem các ô đã điền chưa
    {
        String roomno = txtRoomNo.Text; // Nếu đủ thì gán các dữ liệu từ ô
        String type = txtRoomType.Text;
        String bed = txtBed.Text;
        Int64 price = Int64.Parse(txtPrice.Text); // Chuyển giá trị Price sang
        query = "insert into rooms (roomNo,roomType,bed,price) values ('" +
        roomno + "','" + type + "','" + bed + "','" + price + ")"; // Tạo câu SQL INSERT để thêm
        fn.setData(query, "Đã Thêm Phòng"); //Gọi fn.setData() để chạy câu SQL
        UC_AddRoom_Load(this, null); // Load lại bảng phòng (UC_AddRoom_Load)
        clearALL(); // Gọi clearALL() để reset form
    }
    else
    {
        MessageBox.Show("Xin vui Lòng điền đầy đủ thông tin ", "Warning !",
        MessageBoxButtons.OK, MessageBoxIcon.Warning); // Nếu thiếu dữ liệu thì báo lỗi bằng
    }
}
```

- Thiết lập hàm clearAll():

```
public void clearALL() // Hàm tự động xóa dữ liệu nhập
{
    txtRoomNo.Clear();
    txtRoomType.SelectedIndex = -1;
    txtBed.SelectedIndex = -1;
    txtPrice.Clear();
}
```

- Thiết lập sự kiện Leave:

```
private void UC_AddRoom_Leave(object sender, EventArgs e)
{
    clearALL(); //Khi rời UserControl → Tự động xóa dữ liệu nhập
}
```

- Thiết lập sự kiện Enter:

```
private void UC_AddRoom_Enter(object sender, EventArgs e)
{
    UC_AddRoom_Load(this, null); // Khi quay lại UserControl → Tải lại danh sách phòng
}
```

- *Liên hệ kiến thức:*
 - UC_AddRoom kế thừa UserControl (Inheritance).
 - Có constructor, field, method (clearALL, btnAddRoom_Click).
 - Sử dụng Encapsulation: function xử lý CSDL.
 - Có xử lý if-else → Logic điều kiện.
 - Không có Exception try-catch rõ ràng nhưng có MessageBox.Show để thông báo.
- *Kiến thức áp dụng:*
 - Nền tảng WinForm + UserControl.
 - Class, Constructor, Method, Field.
 - Inheritance (UserControl).
 - Encapsulation.
 - Pass by Value.

4. Thiết kế Form đăng kí khách hàng và Backend Form đăng kí khách hàng

a. Thiết kế Form đăng kí khách hàng

Hình 17: Thiết kế giao diện “Đăng kí khách hàng”

b. Thiết kế Backend Form đăng kí khách hàng

- Code backend UC_CustomerRes

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
```



```

using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Xml.Linq;
namespace ChuongTrinhquanlykhachsan.All_User
{
    public partial class UC_CustomerRes : UserControl
    {
        function fn = new function();
        string query;
        public UC_CustomerRes()
        {
            InitializeComponent();
        }
        public void setComboBox(string query, ComboBox combo)
        {
            SqlDataReader sdr = fn.getForCombo(query);
            while (sdr.Read())
            {
                for (int i = 0; i < sdr.FieldCount; i++)
                {
                    combo.Items.Add(sdr.GetString(i));
                }
            }
            sdr.Close();
        }
        private void label6_Click(object sender, EventArgs e)
        {
        }
        private void UC_CustomerRes_Load(object sender, EventArgs e)
        {
        }
        private void txtBed_SelectedIndexChanged(object sender, EventArgs e)
        {
            txtRoom.SelectedIndex = -1;
            txtRoomNo.Items.Clear();
            txtPrice.Clear();
        }
        private void txtRoom_SelectedIndexChanged(object sender, EventArgs e)
        {
            txtRoomNo.Items.Clear();
            query = "select roomNo from rooms where Bed = '" + txtBed.Text + "' and
roomType = '" + txtRoom.Text + "' and booked = 'NO'";
            setComboBox(query, txtRoomNo);
        }
        int rid;
        private void txtRoomNo_SelectedIndexChanged(object sender, EventArgs e)
        {
            query = "select price , roomid from rooms where roomNo = '" + txtRoomNo.Text
+ "'";
            DataSet ds = fn.getData(query);
            txtPrice.Text = ds.Tables[0].Rows[0][0].ToString();
            rid = int.Parse(ds.Tables[0].Rows[0][1].ToString());
        }
        private void btnAllotCustomer_Click(object sender, EventArgs e)
        {
            if (txtName.Text != "" && txtContact.Text != "" && txtNationality.Text != ""
&& txtGender.Text != "" && txtDob.Text != "" && txtIDProof.Text != "" && txtAddress.Text
!= "" && txtCheckin.Text != "" && txtPrice.Text != "")
            {

```

```

        string name = txtName.Text;
        Int64 mobile = Int64.Parse(txtContact.Text);
        string national = txtNationality.Text;
        string gender = txtGender.Text;
        string dob = txtDob.Text;
        string idproof = txtIDProof.Text;
        string address = txtAddress.Text;
        string checkin = txtCheckin.Text;
        query = "insert into customer (cname, mobile, nationality, gender, dob,
idproof, address, checkin, roomid) values ('" + name + "', " + mobile + ", '" + national
+ "', '" + gender + "', '" + dob + "', '" + idproof + "', '" + address + "', '" +
checkin + "', " + rid + ")"; update rooms set booked = 'YES' where roomNo = '" +
txtRoomNo.Text + "'";
        fn.setData(query, "Số Phòng " + txtRoomNo.Text + " Đăng ký thành công
cho khách hàng.");
        clearALL();
    }
    else
    {
        MessageBox.Show("Vui lòng điền đầy đủ thông tin", "Thông tin",
        MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
}

public void clearALL()
{
    txtName.Clear();
    txtContact.Clear();
    txtNationality.Clear();
    txtGender.SelectedIndex = -1;
    txtDob.ResetText();
    txtIDProof.Clear();
    txtAddress.Clear();
    txtCheckin.ResetText();
    txtBed.SelectedIndex = -1;
    txtRoom.SelectedIndex = -1;
    txtRoomNo.Items.Clear();
    txtPrice.Clear();
}

private void UC_CustomerRes_Leave(object sender, EventArgs e)
{
    clearALL();
}

private void txtName_TextChanged(object sender, EventArgs e)
{
}
}
}

```

- Chức năng :
 - Đăng ký khách hàng.
 - Gán khách vào phòng trống.
- Giải thích code:
 - Thiết lập phương thức setComboBox():

```

public void setComboBox(string query, ComboBox combo) // Gửi query tới database thông qua
fn.getForCombo()
{
    SqlDataReader sdr = fn.getForCombo(query); //getForCombo trả về
    SqlDataReader → đọc tuần tự từng dòng dữ liệu
    while (sdr.Read()) //Với mỗi dòng, lặp qua tất cả các cột (FieldCount)
    {
        for (int i = 0; i < sdr.FieldCount; i++)

```

```

        {
            combo.Items.Add(sdr.GetString(i)); //Lấy giá trị dạng chuỗi và thêm
(Add) vào combo.Items.
        }
    }
    sdr.Close(); //Đọc hết → Đóng SqlDataReader (sdr.Close())
}

```

➤ Chức năng: thêm các giá trị vào ComboBox cụ thể là danh sách số phòng còn trống.

- Thiết lập xử lý chọn loại giường:

```

private void txtBed_SelectedIndexChanged(object sender, EventArgs e) // Khi đổi loại
giường, sẽ:
{
    txtRoom.SelectedIndex = -1; // Reset loại phòng (Room)
    txtRoomNo.Items.Clear(); // Xóa số phòng (RoomNo)
    txtPrice.Clear(); // Xóa giá phòng (Price)
}

```

- Thiết lập xử lý chọn loại phòng:

```

private void txtRoom_SelectedIndexChanged(object sender, EventArgs e) // Khi chọn loại
phòng:
{
    txtRoomNo.Items.Clear(); // Xóa danh sách số phòng cũ
    query = "select roomNo from rooms where Bed = '" + txtBed.Text + "' and
roomType = '" + txtRoom.Text + "' and booked = 'NO'"; // Tìm các số phòng chưa đặt, phù
hợp Bed và RoomType
    setComboBox(query, txtRoomNo); // Đổ vào ComboBox số phòng (RoomNo)
}


```

- Thiết lập xử lý chọn số phòng:

```

int rid;
private void txtRoomNo_SelectedIndexChanged(object sender, EventArgs e) // Khi chọn số
phòng
{
    query = "select price , roomid from rooms where roomNo = '" + txtRoomNo.Text
+ "'";
    DataSet ds = fn.getData(query); // Lấy giá và roomid từ Database
    txtPrice.Text = ds.Tables[0].Rows[0][0].ToString(); // Hiển thị giá
    rid = int.Parse(ds.Tables[0].Rows[0][1].ToString()); // Lưu rid để sau này
ghi vào bảng customer
}

```

- Thiết lập nút “Thêm khách hàng”  :

```

private void btnAllotCustomer_Click(object sender, EventArgs e)
{
    if (txtName.Text != "" && txtContact.Text != "" && txtNationality.Text != ""
&& txtGender.Text != "" && txtDob.Text != "" && txtIDProof.Text != "" && txtAddress.Text
!= "" && txtCheckin.Text != "" && txtPrice.Text != "") // Kiểm tra tất cả ô nhập đã điền chưa
    { // Nếu đủ thì lấy dữ liệu từ Form
        string name = txtName.Text;
        Int64 mobile = Int64.Parse(txtContact.Text);
        string national = txtNationality.Text;
        string gender = txtGender.Text;
        string dob = txtDob.Text;
        string idproof = txtIDProof.Text;
    }
}

```

```

        string address = txtAddress.Text;
        string checkin = txtCheckin.Text;
        // Tạo SQL INSERT thêm khách mới
        query = "insert into customer (cname, mobile, nationality, gender, dob, idproof, address, checkin, roomid) values ('" + name + "', '" + mobile + "', '" + nationality + "', '" + gender + "', '" + dob + "', '" + idproof + "', '" + address + "', '" + checkin + "', '" + rid + "')"; update rooms set booked = 'YES' where roomNo = '" + txtRoomNo.Text + "'"; //Đồng thời UPDATE phòng thành booked = 'YES'
        fn.setData(query, "Số Phòng " + txtRoomNo.Text + " Đăng ký thành công cho khách hàng."); //Chạy SQL qua fn.setData()
        // Hiện thông báo
        clearALL(); //Gọi clearALL() để reset form
    }
    else
    {
        MessageBox.Show("Vui lòng điền đầy đủ thông tin", "Thông tin", MessageBoxButtons.OK, MessageBoxIcon.Information); // Nếu thiếu: Báo lỗi
    }
}

```

- Thiết lập hàm clearAll():

```

public void clearALL() // Xóa sạch form về trạng thái ban đầu
{
    txtName.Clear();
    txtContact.Clear();
    txtNationality.Clear();
    txtGender.SelectedIndex = -1;
    txtDob.ResetText();
    txtIDProof.Clear();
    txtAddress.Clear();
    txtCheckin.ResetText();
    txtBed.SelectedIndex = -1;
    txtRoom.SelectedIndex = -1;
    txtRoomNo.Items.Clear();
    txtPrice.Clear();
}

```

- Thiết lập sự kiện Leave:

```

private void UC_CustomerRes_Leave(object sender, EventArgs e)
{
    clearALL();
}

```

- *Liên hệ kiến thức:*
 - UC_CustomerRes kế thừa UserControl (Inheritance).
 - Dùng constructor, method, field.
 - Dùng setComboBox → minh họa Method với tham số tham chiếu (truyền ComboBox tham chiếu → Pass by Reference).
 - Gọi fn.getForCombo → Encapsulation.
 - Xử lý query, if-else.
- *Kiến thức áp dụng:*
 - Nền tảng WinForm + UserControl.
 - Class, Constructor, Method, Field.
 - Inheritance.

- Encapsulation.
- Pass by Reference (ComboBox).

5. Thiết kế Form thanh toán phòng và BackEnd Form thanh toán phòng

a. Thiết kế Form thanh toán phòng

Hình 18: Thiết kế giao diện “Thanh Toán” (Thanh Toán Tiền Phòng)

b. Thiết kế BackEnd Form thanh toán phòng

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Xml.Linq;

namespace ChuongTrinhquanlykhachsan.All_User
{
    public partial class UC_CheckOut : UserControl
    {
        function fn = new function();
        string query;
        public UC_CheckOut()
        {
            InitializeComponent();
        }
        private void label2_Click(object sender, EventArgs e)
        {
        }
        private void UC_CheckOut_Load(object sender, EventArgs e)
        {
            query = "select customer.cid, customer.cname,customer.mobile,
customer.nationality,customer.gender,customer.dob,customer.idproof,customer.address,cust
```

```

omer.checkin,rooms.roomNo,rooms.roomType,rooms.bed,rooms.price from customer inner join
rooms on customer.roomid = rooms.roomid where chekout = 'NO';
    DataSet ds = fn.getData(query);
    guna2DataGridView1.DataSource = ds.Tables[0];
}
private void txtName_TextChanged(object sender, EventArgs e)
{
    query = "select customer.cid, customer.cname,customer.mobile,
customer.nationality,customer.gender,customer.dob,customer.idproof,customer.address,cust
omer.checkin,rooms.roomNo,rooms.roomType,rooms.bed,rooms.price from customer inner join
rooms on customer.roomid = rooms.roomid where cname like '" + txtName.Text + "%' and
chekout = 'NO'";
    DataSet ds = fn.getData(query);
    guna2DataGridView1.DataSource = ds.Tables[0];
}
int id;
private void guna2DataGridView1_CellContentClick(object sender,
DataGridViewCellEventArgs e)
{
    if (guna2DataGridView1.Rows[e.RowIndex].Cells[e.RowIndex].Value != null)
    {
        id =
int.Parse(guna2DataGridView1.Rows[e.RowIndex].Cells[0].Value.ToString());
        txtCName.Text =
guna2DataGridView1.Rows[e.RowIndex].Cells[1].Value.ToString();
        txtRoom.Text =
guna2DataGridView1.Rows[e.RowIndex].Cells[2].Value.ToString();
    }
}
private void btnCheckOut_Click(object sender, EventArgs e)
{
    if (txtCName.Text != "")
    {
        if (MessageBox.Show("Bạn có chắc chắn không ?", "Xác nhận ",
MessageBoxButtons.OKCancel, MessageBoxIcon.Warning) == DialogResult.OK)
        {
            string cdate = txtCheckOutDate.Text;
            query = "update customer set chekout = 'YES', checkout = '" + cdate
+ "' where cid = " + id + " update rooms set booked = 'NO' where roomNO = '" +
txtRoom.Text + "'";
            fn.setData(query, "Khách hàng thanh toán thành công !");
            UC_CheckOut_Load(this, null);
            clearALL();
        }
    }
    else
    {
        MessageBox.Show("Không có thông tin khách hàng !", "Thông báo",
MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
}
public void clearALL()
{
    txtCName.Clear();
    txtName.Clear();
    txtRoom.Clear();
    txtCheckOutDate.ResetText();
}
private void UC_CheckOut_Leave(object sender, EventArgs e)
{
    clearALL();
}
}
}

```

- Chức năng :
 - Hiện thị khách hàng đang ở.
 - Chọn khách hàng cần thanh toán.
 - Cập nhật checkout và trả phòng.
- Giải thích code:
 - Thiết lập sự kiện Load:


```
private void UC_CheckOut_Load(object sender, EventArgs e)
{
    query = "select customer.cid, customer.cname, customer.mobile,
customer.nationality, customer.gender, customer.dob, customer.idproof, customer.address, customer.checkin, rooms.roomNo, rooms.roomType, rooms.bed, rooms.price from customer inner join rooms on customer.roomid = rooms.roomid where checkout = 'NO'"; // Khi UserControl load, lấy danh sách khách chưa checkout (checkout = 'NO')
    DataSet ds = fn.GetData(query); // Ghép bảng customer và rooms
    guna2DataGridView1.DataSource = ds.Tables[0]; // Hiện thị trong DataGridView (guna2DataGridView1)
}
```

- Thiết lập tìm nhanh theo tên:

```
private void txtName_TextChanged(object sender, EventArgs e)
{
    query = "select customer.cid, customer.cname, customer.mobile,
customer.nationality, customer.gender, customer.dob, customer.idproof, customer.address, customer.checkin, rooms.roomNo, rooms.roomType, rooms.bed, rooms.price from customer inner join rooms on customer.roomid = rooms.roomid where cname like '" + txtName.Text + "%' and checkout = 'NO'"; // Khi gõ tên khách (txtName), tự lọc danh sách khách chưa checkout (LIKE)
    DataSet ds = fn.GetData(query);
    guna2DataGridView1.DataSource = ds.Tables[0]; // Tìm nhanh tên gần giống
}
```

- Thiết lập sự kiện click trong dòng bảng:

```
int id;
private void guna2DataGridView1_CellContentClick(object sender, DataGridViewCellEventArgs e) // Khi click 1 ô trong DataGridView
{
    if (guna2DataGridView1.Rows[e.RowIndex].Cells[e.RowIndex].Value != null)
    {
        id =
        int.Parse(guna2DataGridView1.Rows[e.RowIndex].Cells[0].Value.ToString()); // Lấy cid của khách (Rows[e.RowIndex].Cells[0])
        txtCName.Text =
        guna2DataGridView1.Rows[e.RowIndex].Cells[1].Value.ToString();
        txtRoom.Text =
        guna2DataGridView1.Rows[e.RowIndex].Cells[2].Value.ToString(); // Lấy cname và roomNo để hiển thị ra TextBox
        // Lưu id để khi bấm Check Out, sẽ chạy câu SQL UPDATE đúng khách
    }
}
```

- Thiết lập nút “Thanh toán” :

```
private void btnCheckOut_Click(object sender, EventArgs e)
{
}
```

```

        if (txtCName.Text != "") // Nếu đã chọn khách (txtCName không rỗng)
        {
            if (MessageBox.Show("Bạn có chắc chắn không ?", "Xác nhận ",
                MessageBoxButtons.OKCancel, MessageBoxIcon.Warning) == DialogResult.OK) //Hiện hộp thoại
                xác nhận, nếu ok:
            {
                string cdate = txtCheckOutDate.Text; // Lấy ngày trả phòng
                (txtCheckOutDate)
                query = "update customer set checkout = 'YES', checkout = '" + cdate
                + "' where cid = " + id + " update rooms set booked = 'NO' where roomNO = '" +
                txtRoom.Text + "'"; // Cập nhật customer: checkout = 'YES', lưu ngày, cập nhật rooms:
                booked = 'NO' → phòng trống lại
                fn.setData(query, "Khách hàng thanh toán thành công !"); // Gọi
                fn.setData để chạy SQL
                UC_CheckOut_Load(this, null); // Load lại danh sách khách
                clearALL(); // Gọi clearALL() để reset form
            }
        }
        else
        {
            MessageBox.Show("Không có thông tin khách hàng !", "Thông báo",
                MessageBoxButtons.OK, MessageBoxIcon.Information); // Nếu chưa chọn khách: Báo lỗi
        }
    }
}

```

- Thiết lập hàm clearAll():

```

public void clearALL() // Xóa dữ liệu form
{
    txtCName.Clear();
    txtName.Clear();
    txtRoom.Clear();
    txtCheckOutDate.ResetText();
}

```

- Thiết lập sự kiện Leave:

```

private void UC_CheckOut_Leave(object sender, EventArgs e)
{
    clearALL(); // Khi thoát UserControl: Tự động xóa form
}

```

- *Liên hệ kiến thức:*

- UC_CheckOut kế thừa UserControl (Inheritance).
- Dùng constructor, method, field.
- btnCheckOut_Click có MessageBox.Show + if-else.
- Gọi fn.setData → Encapsulation.
- Không có xử lý try-catch cụ thể → chưa khai thác Exception Handling.

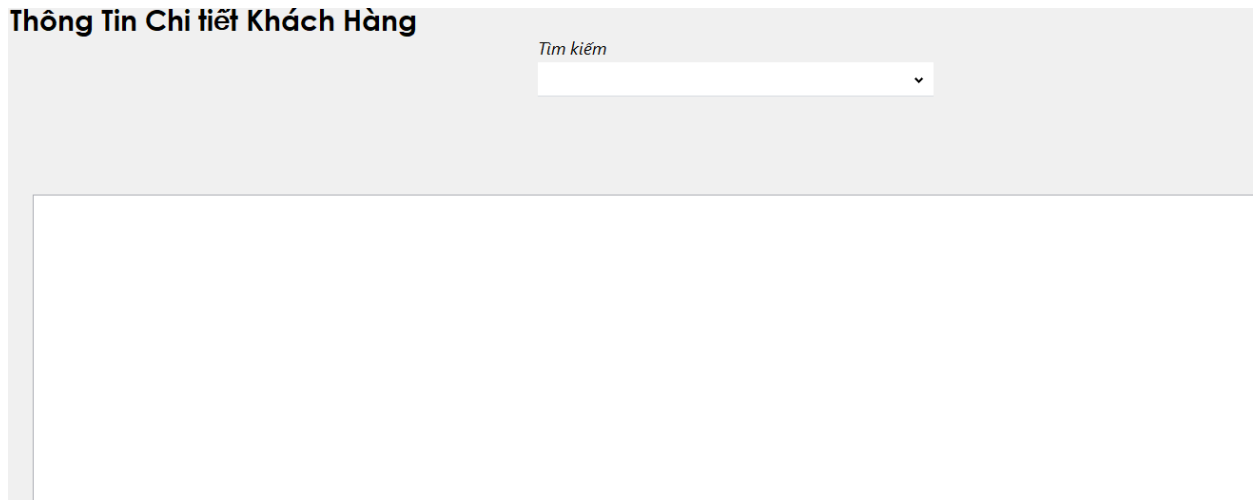
- *Kiến thức áp dụng:*

- Nền tảng WinForm + UserControl.
- Class, Constructor, Method, Field.
- Inheritance.
- Encapsulation.
- Pass by Value.

6. Thiết kế Form thông tin chi tiết khách hàng và BackEnd Form thông tin chi tiết khách hàng

a. Thiết kế Form thông tin chi tiết khách hàng

Thông Tin Chi tiết Khách Hàng



Hình 19: Thiết kế giao diện “Thông Tin Chi Tiết Khách Hàng”

b. Thiết kế BackEnd Form thông tin chi tiết khách hàng

- o Code backend `UC_CustomerDetails`

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Xml.Linq;

namespace ChuongTrinhquanlykhachsanh.All_User
{
    public partial class UC_CustomerDetails : UserControl
    {
        function fn = new function();
        string query;
        public UC_CustomerDetails()
        {
            InitializeComponent();
        }
        private void label2_Click(object sender, EventArgs e)
        {
        }
        private void txtSearchBy_SelectedIndexChanged(object sender, EventArgs e)
        {
            if (txtSearchBy.SelectedIndex == 0)
            {
                query = "select customer.cid, customer.cname,customer.mobile,
customer.nationality,customer.gender,customer.dob,customer.idproof,customer.address,cust
omer.checkin,rooms.roomNo,rooms.roomType,rooms.bed,rooms.price from customer inner join
rooms on customer.roomid = rooms.roomid";
                getRecord(query);
            }
            else if (txtSearchBy.SelectedIndex == 1)
            {
            }
        }
    }
}
```

```

        query = "select customer.cid, customer.cname,customer.mobile,
customer.nationality,customer.gender,customer.dob,customer.idproof,customer.address,cust
omer.checkin,rooms.roomNo,rooms.roomType,rooms.bed,rooms.price from customer inner join
rooms on customer.roomid = rooms.roomid where checkout is null";
        getRecord(query);
    }
    else if (txtSearchBy.SelectedIndex == 2)
    {
        query = "select customer.cid, customer.cname,customer.mobile,
customer.nationality,customer.gender,customer.dob,customer.idproof,customer.address,cust
omer.checkin,rooms.roomNo,rooms.roomType,rooms.bed,rooms.price from customer inner join
rooms on customer.roomid = rooms.roomid where checkout is not null";
        getRecord(query);
    }
}
private void getRecord(string query)
{
    DataSet ds = fn.getData(query);
    guna2DataGridView1.DataSource = ds.Tables[0];
}
private void guna2DataGridView1_CellContentClick(object sender,
DataGridViewCellEventArgs e)
{
}
}
}

```

- Chức năng :
 - Tra cứu danh sách khách hàng.
 - Lọc theo: tất cả các khách hàng, khách hàng chưa checkout và khách hàng đã checkout.
- Giải thích code:
 - Thiết lập sự kiện chọn điều kiện lọc:

```

private void txtSearchBy_SelectedIndexChanged(object sender, EventArgs e)
{
    if (txtSearchBy.SelectedIndex == 0) //Nếu chọn All Customer Detail
    {
        query = "select customer.cid, customer.cname,customer.mobile,
customer.nationality,customer.gender,customer.dob,customer.idproof,customer.address,cust
omer.checkin,rooms.roomNo,rooms.roomType,rooms.bed,rooms.price from customer inner join
rooms on customer.roomid = rooms.roomid"; // Lấy TẤT CẢ khách hàng
        getRecord(query);
    }
    else if (txtSearchBy.SelectedIndex == 1) //Nếu chọn Customer In Hotel
    {
        query = "select customer.cid, customer.cname,customer.mobile,
customer.nationality,customer.gender,customer.dob,customer.idproof,customer.address,cust
omer.checkin,rooms.roomNo,rooms.roomType,rooms.bed,rooms.price from customer inner join
rooms on customer.roomid = rooms.roomid where checkout is null"; // Khách CHƯA checkout
        getRecord(query);
    }
    else if (txtSearchBy.SelectedIndex == 2) // Nếu chọn CheckOut Customer
    {
        query = "select customer.cid, customer.cname,customer.mobile,
customer.nationality,customer.gender,customer.dob,customer.idproof,customer.address,cust
omer.checkin,rooms.roomNo,rooms.roomType,rooms.bed,rooms.price from customer inner join
rooms on customer.roomid = rooms.roomid where checkout is not null"; //Khách ĐÃ checkout
        getRecord(query);
    }
}
}

```

- Thiết lập hàm getRecord:

```
private void getRecord(string query)
{
    DataSet ds = fn.getData(query);    // Thực thi câu SQL (query) qua fn.getData,
    lấy DataSet trả về
    guna2DataGridView1.DataSource = ds.Tables[0];    //Hiển thị kết quả vào
    DataGridView
}
```

- Liên hệ kiến thức:
 - UC_CustomerDetails kế thừa UserControl.
 - Dùng constructor, method, field.
 - Có Method getRecord tái sử dụng query → tính mở rộng.
- Kiến thức áp dụng:
 - Nền tảng WinForm + UserControl.
 - Class, Constructor, Method, Field.
 - Inheritance.
 - Encapsulation.

7. Thiết kế Form quản lý nhân viên và BackEnd Form quản lý nhân viên

a. Thiết kế Form quản lý nhân viên

- Đăng kí nhân viên:

The screenshot shows a web form titled 'Nhân Viên' with three tabs: 'Đăng Ký Nhân Viên', 'Thông tin nhân viên', and 'Xóa nhân viên'. The 'Đăng Ký Nhân Viên' tab is active. It features several input fields: 'ID' (with a dropdown arrow), 'Tên' (Name) with 'Enter Name' placeholder, 'Số Điện Thoại' (Mobile Number) with 'Enter Mobile' placeholder, 'Email' with 'Enter Email' placeholder, 'Mật Khẩu' (Password) with 'Enter Password' placeholder, and 'Xác nhận mật khẩu' (Confirm Password) with 'Enter Password' placeholder. A 'Đăng Ký' button is positioned at the bottom right of the form.

Hình 20: Thiết kế giao diện tab “Đăng Ký Nhân Viên” trong giao diện “Nhân Viên”

- Thông tin nhân viên:

The screenshot shows the same 'Nhân Viên' form with the 'Thông tin nhân viên' tab selected. The main content area of the form displays the text 'Thông tin nhân viên' in a bold, black font, centered horizontally.

Hình 21: Thiết kế giao diện tab “Thông tin nhân viên” trong giao diện “Nhân Viên”

- Xoá nhân viên:

Hình 22: Thiết kế giao diện tab “Xóa nhân viên” trong giao diện “Nhân Viên”

b. Thiết kế BackEnd Form quản lý nhân viên

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Xml.Linq;

namespace ChuongTrinhquanlykhachsan.All_User
{
    public partial class UC_Employee : UserControl
    {
        function fn = new function();
        string query;
        public UC_Employee()
        {
            InitializeComponent();
        }
        private void UC_Employee_Load(object sender, EventArgs e)
        {
            getMathID();
        }
        // *****
        public void getMathID()
        {
            query = "select max(eid) from employee";
            DataSet ds = fn.getData(query);
            if (ds.Tables[0].Rows[0][0].ToString() != "")
            {
                Int64 num = Int64.Parse(ds.Tables[0].Rows[0][0].ToString());
                labelToSET.Text = (num + 1).ToString();
            }
        }
        private void btnRegistration_Click(object sender, EventArgs e)
        {
            if (txtName.Text != "" && txtMobile.Text != "" && txtGender.Text != "" &&
                txtEmail.Text != "" && txtUsername.Text != "" && txtPassword.Text != "")
            {
                string name = txtName.Text;
                int mobile = int.Parse(txtMobile.Text);
                string gender = txtGender.Text;
                string email = txtEmail.Text;
            }
        }
    }
}
```

```

        string username = txtUsername.Text;
        string pass = txtPassword.Text;
        query = " insert into employee
(ename,mobile,gender,emailid,username,pass) values ('" + name + "','" + mobile + "','" +
gender + "','" + email + "','" + username + "','" + pass + "')";
        fn.setData(query, "Đăng ký nhân viên thành công !");
        clearALL();
        getMathID();
    }
}
public void clearALL()
{
    txtName.Clear();
    txtMobile.Clear();
    txtGender.SelectedIndex = -1;
    txtUsername.Clear();
    txtPassword.Clear();
    txtEmail.Clear();
}
private void tabEmployee_SelectedIndexChanged(object sender, EventArgs e)
{
    if (tabEmployee.SelectedIndex == 1)
    {
        setEmployee(guna2DataGridView2);
    }
    else if (tabEmployee.SelectedIndex == 2)
    {
        setEmployee(guna2DataGridView3);
    }
}
public void setEmployee(DataGridView dgv)
{
    query = "select * from employee";
    DataSet ds = fn.getData(query);
    dgv.DataSource = ds.Tables[0];
}
private void btnDelete_Click(object sender, EventArgs e)
{
    if (txtID.Text != "")
    {
        if (MessageBox.Show("Bạn có chắc chắn muốn xóa Không ?", "Xác nhận",
MessageBoxButtons.YesNo, MessageBoxIcon.Warning) == DialogResult.Yes)
        {
            query = "delete from employee where eid = " + txtID.Text;
            fn.setData(query, "Xóa nhân viên thành công !");
            tabEmployee_SelectedIndexChanged(this, null);
        }
    }
}
private void UC_Employee_Leave(object sender, EventArgs e)
{
    clearALL();
}
private void txtName_TextChanged(object sender, EventArgs e)
{
}
}
}

```

- Chức năng :

- Thêm nhân viên, thông tin nhân viên và xóa nhân viên.

- Giải thích code:

- Thiết lập sự kiện Load:

```
private void UC_Employee_Load(object sender, EventArgs e) // Khi UserControl load
{
    getMathID();
}
```

- Thiết lập hàm getMathID():

```
public void getMathID()
{
    query = "select max(eid) from employee"; // Lấy ID nhân viên lớn nhất
    DataSet ds = fn.getData(query);
    if (ds.Tables[0].Rows[0][0].ToString() != "")
    {
        Int64 num = Int64.Parse(ds.Tables[0].Rows[0][0].ToString());
        labelToSET.Text = (num + 1).ToString(); // Tăng thêm 1 → hiển thị sẵn để
        cấp mã cho nhân viên mới
    }
}
```

- Thiết lập nút “Đăng ký” :

Đăng Ký

```
private void btnRegistration_Click(object sender, EventArgs e) // Khi bấm nút
{
    if (txtName.Text != "" && txtMobile.Text != "" && txtGender.Text != "" &&
    txtEmail.Text != "" && txtUsername.Text != "" && txtPassword.Text != "") // Kiểm tra ô
    nhập đã điền đủ chưa
    {
        string name = txtName.Text; // Lấy dữ liệu
        int mobile = int.Parse(txtMobile.Text);
        string gender = txtGender.Text;
        string email = txtEmail.Text;
        string username = txtUsername.Text;
        string pass = txtPassword.Text;
        query = " insert into employee
        (ename,mobile,gender,emailid,username,pass) values ('" + name + "','" + mobile + "','" +
        gender + "','" + email + "','" + username + "','" + pass + "')"; // Tạo câu SQL INSERT
        fn.setData(query, "Đăng ký nhân viên thành công !"); // Gọi fn.setData()
        để ghi DB
        clearALL(); // Gọi clearALL() để xóa form
        getMathID(); // Gọi lại getMathID() để chuẩn bị ID mới
    }
}
```

- Thiết lập hàm clearAll():

```
public void clearALL() // Reset form đăng ký
{
    txtName.Clear();
    txtMobile.Clear();
    txtGender.SelectedIndex = -1;
    txtUsername.Clear();
    txtPassword.Clear();
    txtEmail.Clear();
}
```

- Thiết lập chuyển tab (giao diện nhỏ) Employee_SelectedIndexChanged:

```
private void tabEmployee_SelectedIndexChanged(object sender, EventArgs e)
{ // Khi đổi tab
  if (tabEmployee.SelectedIndex == 1)
  {
    setEmployee(guna2DataGridView2); // Nếu tab thứ 2 → gọi setEmployee() để
    hiển thị danh sách nhân viên ở guna2DataGridView2
  }
  else if (tabEmployee.SelectedIndex == 2)
  {
    setEmployee(guna2DataGridView3); // Nếu tab thứ 3 → gọi setEmployee() ở
    guna2DataGridView3
  }
}
```

- Thiết lập hàm setEmployee:

```
public void setEmployee(DataGridView dgv)
{
  query = "select * from employee";
  DataSet ds = fn.getData(query); // Lấy toàn bộ nhân viên từ DB
  dgv.DataSource = ds.Tables[0]; // Đổ vào DataGridView (dgv)
}
```

- Thiết lập nút “Xóa” :

```
private void btnDelete_Click(object sender, EventArgs e)
{
  if (txtID.Text != "") // Nếu nhập ID
  {
    if (MessageBox.Show("Bạn có chắc chắn muốn xóa Không ?", "Xác nhận",
    MessageBoxButtons.YesNo, MessageBoxIcon.Warning) == DialogResult.Yes) // Hiện MessageBox
    xác nhận, nếu Yes
    {
      query = "delete from employee where eid = " + txtID.Text;
      fn.setData(query, "Xóa nhân viên thành công !"); // Xóa nhân viên
      (DELETE)
      tabEmployee_SelectedIndexChanged(this, null); // Gọi lại
      tabEmployee_SelectedIndexChanged để reload danh sách
    }
  }
}
```

- Thiết lập sự kiện Leave:

```
private void UC_Employee_Leave(object sender, EventArgs e)
{
  clearALL(); // Khi thoát UC_Employee → tự động xóa form đăng ký
}
```

- Liên hệ kiến thức:

- UC_Employee kế thừa UserControl.
- Có constructor, field, method.
- Có btnDelete_Click → dùng MessageBox để xác nhận.
- Không có try-catch nhưng có điều kiện, Encapsulation.
- Có tái sử dụng method setEmployee → tính mở rộng.

- *Kiến thức áp dụng:*
 - o Nền tảng WinForm + UserControl.
 - o Class, Constructor, Method, Field.
 - o Inheritance.
 - o Encapsulation.

8. Chương trình trong file Program.cs

```
using Microsoft.SqlServer.Server;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;
using static System.Net.Mime.MediaTypeNames;

namespace ChuongTrinhquanlykhachsan
{
    internal static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

- *Liên hệ kiến thức:*
 - o Entry point Main → chạy ứng dụng WinForm.
 - o [STAThread] → cần cho UI Thread của WinForm.
 - o Application.Run(new Form1()); → khởi chạy Form bằng Constructor.
- *Kiến thức áp dụng:*
 - o Nền tảng WinForm.
 - o Method Main (entry point).
 - o Constructor.

9. Chương trình trong file function.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.SqlClient;
using System.Data;
using System.Windows.Forms;
namespace ChuongTrinhquanlykhachsan
```



```

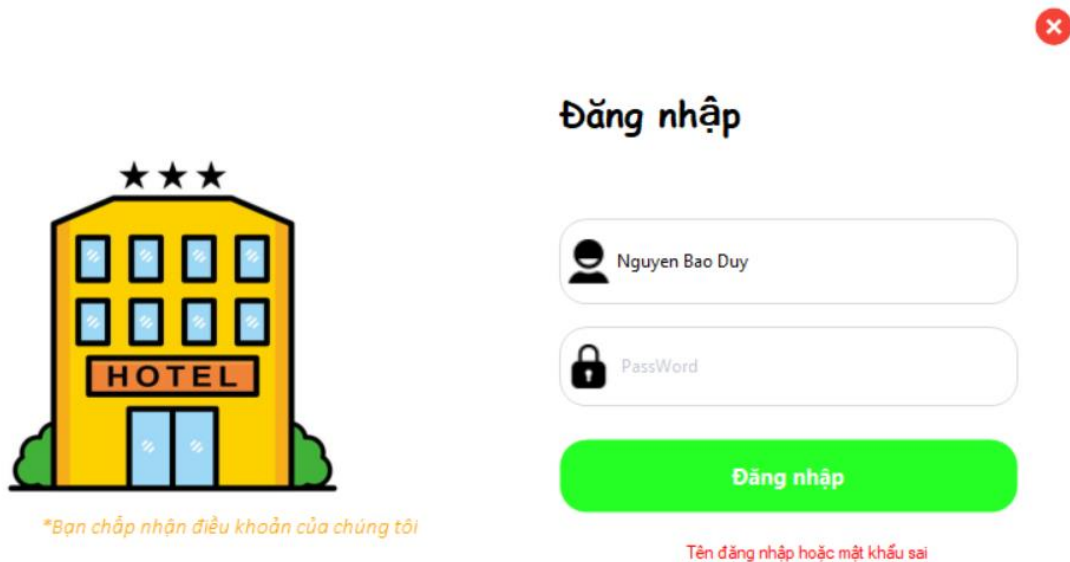
{
    class function
    {
        protected SqlConnection getConnection()
        {
            SqlConnection con = new SqlConnection();
            con.ConnectionString = "Data
Source=(LocalDB)\\MSSQLLocalDB;AttachDbFilename=C:\\Users\\duyba\\Documents\\dbMyHotel.m
df;Integrated Security=True;Connect Timeout=30";
            return con;
        }
        public DataSet getData(string query)
        {
            SqlConnection con = getConnection();
            SqlCommand cmd = new SqlCommand();
            cmd.Connection = con;
            cmd.CommandText = query;
            SqlDataAdapter da = new SqlDataAdapter(cmd);
            DataSet ds = new DataSet();
            da.Fill(ds);
            return ds;
        }
        public void setData(string query, string message)
        {
            SqlConnection con = getConnection();
            SqlCommand cmd = new SqlCommand();
            cmd.Connection = con;
            con.Open();
            cmd.CommandText = query;
            cmd.ExecuteNonQuery();
            con.Close();
            MessageBox.Show(message, "Success", MessageBoxButtons.OK,
            MessageBoxIcon.Information);
        }
        public SqlDataReader getForCombo(string query)
        {
            SqlConnection con = getConnection();
            SqlCommand cmd = new SqlCommand();
            cmd.Connection = con;
            con.Open();
            cmd = new SqlCommand(query, con);
            SqlDataReader sdr = cmd.ExecuteReader();
            return sdr;
        }
    }
}

```

- *Liên hệ kiến thức:*
 - function là class độc lập → chứa method getConnection, getData, setData, getForCombo.
 - SqlConnection bên trong → Encapsulation: chỉ có expose method, không để lộ chi tiết.
 - Có xử lý mở kết nối, đóng kết nối.
 - MessageBox.Show → xử lý logic người dùng.
- *Kiến thức áp dụng:*
 - Class.
 - Method.
 - Encapsulation.
 - Database Connection.

II. Kết quả đồ án

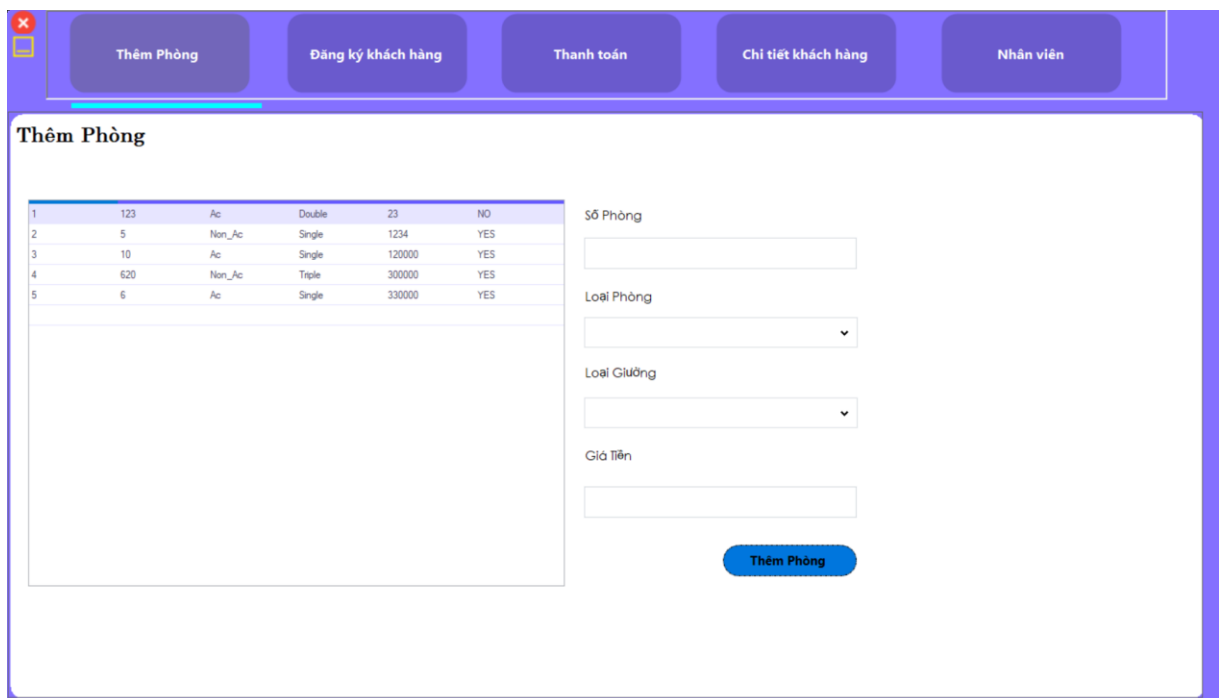
- Ở giao diện ban đầu chương trình, khi nhập sai mật khẩu, sai tên đăng nhập hoặc để trống không điền vào thông tin thì màn hình sẽ thông báo dòng chữ màu đỏ với nội dung: “Tên đăng nhập hoặc mật khẩu sai”.



The screenshot shows a login interface titled "Đăng nhập". On the left is a cartoon illustration of a yellow hotel building with three stars and the word "HOTEL" on its sign. Below the illustration is the text: "*Bạn chấp nhận điều khoản của chúng tôi". On the right, there are two input fields: the first contains the name "Nguyen Bao Duy" and the second is labeled "PassWord". Below these fields is a green button labeled "Đăng nhập". A red error message "Tên đăng nhập hoặc mật khẩu sai" is displayed at the bottom right. A red 'X' icon is in the top right corner.

Hình 23: Dòng chữ thông báo xuất hiện ở giao diện “Đăng nhập”

- Khi nhập đúng tên đăng nhập và mật khẩu thì ta sẽ truy cập vào được DashBoard của hệ thống quản lý khách sạn.



The screenshot shows a dashboard with a purple header containing five buttons: "Thêm Phòng", "Đăng ký khách hàng", "Thanh toán", "Chi tiết khách hàng", and "Nhân viên". The "Thêm Phòng" button is selected. Below the header, the page is titled "Thêm Phòng". On the left, there is a table with 6 columns: ID, Room Number, Room Type, Bed Type, Price, and Status. The table contains 5 rows of data. On the right, there are four input fields labeled "Số Phòng", "Loại Phòng", "Loại Giường", and "Giá Tiền". Below these fields is a blue button labeled "Thêm Phòng".

ID	Room Number	Room Type	Bed Type	Price	Status
1	123	Ac	Double	23	NO
2	5	Non_Ac	Single	1234	YES
3	10	Ac	Single	120000	YES
4	620	Non_Ac	Triple	300000	YES
5	6	Ac	Single	330000	YES

Hình 24: Giao diện DashBoard của hệ thống quản lý khách sạn khi đăng nhập thành công

- Ở giao diện “Thêm Phòng” khi nhân viên nhập *Số Phòng*, *Loại Phòng*, *Loại Giường*, *Giá Tiền* như hình bên dưới thì thông tin phòng được cập nhật lên hệ thống để khách hàng có thể thuê phòng.

Thêm Phòng

1	123	Ac	Double	23	NO
2	5	Non_Ac	Single	1234	YES
3	10	Ac	Single	120000	YES
4	620	Non_Ac	Triple	300000	YES
5	6	Ac	Single	330000	YES

Số Phòng:

Loại Phòng:

Loại Giường:

Giá Tiền:

Thêm Phòng

Hình 25: Cập nhật các thông tin về Số Phòng, Loại Phòng, Loại Giường và Giá Tiền trong giao diện “Thêm Phòng”

- Kết quả sau khi chọn *Thêm Phòng* thì thông tin phòng được cập nhật ngay lập tức lên hệ thống để sẵn sàng cho thuê.

Thêm Phòng

1	123	Ac	Double	23	NO
2	5	Non_Ac	Single	1234	YES
3	10	Ac	Single	120000	YES
4	620	Non_Ac	Triple	300000	YES
5	6	Ac	Single	330000	YES
6	213	Ac	Double	500000	NO

Số Phòng:

Loại Phòng:

Loại Giường:

Giá Tiền:

Thêm Phòng

Hình 26: Kết quả khi nhân viên cập nhật các thông tin về phòng ở lên hệ thống thành công

- Ở giao diện “Đăng kí khách hàng”, khi nhân viên nhập các thông tin cơ bản của người thuê như tên, số điện thoại, quốc tịch,... thì hệ thống sẽ cập nhật thông tin khách hàng lên hệ thống phòng đang cho thuê và xóa phòng đang được thuê đó ra khỏi phòng sẵn sàng cho thuê.

Hình 27: Điền các thông tin của khách thuê phòng trong giao diện “Đăng ký khách hàng”

- Kết quả thông tin của khách hàng vừa thuê phòng được cập nhật lên hệ thống đang cho thuê phòng. Ta sẽ quan sát được tất cả các khách hàng đang thuê phòng tại khách sạn tương ứng với các thông tin cá nhân cơ bản cũng như thông tin về loại phòng mà họ đang thuê.

Thêm Phòng

Đăng ký khách hàng

Thanh toán

Chi tiết khách hàng

Nhân viên

Thông Tin Chi tiết Khách Hàng

Tìm kiếm

In Hotel Customer

5	Nguyen Bao Duy	897654335	VietNam	Male	08/12/2004	22200041	Kí túc xá Khu B - ...	16/06/2025	213	Ac	Double	500000
6	Đặng Đình Khôi	3982847727	VietNam	Male	12/06/2004	22200084	Đi An, Bình Dương	18/06/2025	123	Ac	Double	23
7	Bùi Hồng Hà	7894736382	VietNam	Male	21/06/2004	22200050	Thuan An ,Bình D...	18/06/2025	305	Non_Ac	Triple	100000

Hình 28: Kết quả cập nhật thông tin của khách hàng lên hệ thống sau khi đăng ký thành công

- Ở giao diện “Thanh Toán”, khi nhân viên nhập tên của người thuê phòng thì hệ thống sẽ cập nhật thông tin khách hàng lên hệ thống và số phòng đang cho thuê.

Thanh Toán

Tìm Kiếm

Nguyễn Bao Duy

5	Nguyễn Bao Duy	897654335	Việt Nam	Male	08/12/2004	22200041	Khách sạn Khu B - ...	16/06/2025	213	Ac	Double	500000
---	----------------	-----------	----------	------	------------	----------	-----------------------	------------	-----	----	--------	--------

Tên: Nguyễn Bao Duy Số Phòng: 897654335 Ngày Thanh Toán: 16/06/2025 **Thanh Toán**

Hình 29: Nhập vào thông tin Tên và Số Phòng của người thuê phòng để thanh toán

- Sau khi hoàn tất thanh toán, hệ thống sẽ hiển thị ra một hộp thoại thông báo (Message Box) để thông báo rằng Khách hàng đã thanh toán thành công.

Thanh Toán

Tìm Kiếm

Nguyễn Bao Duy

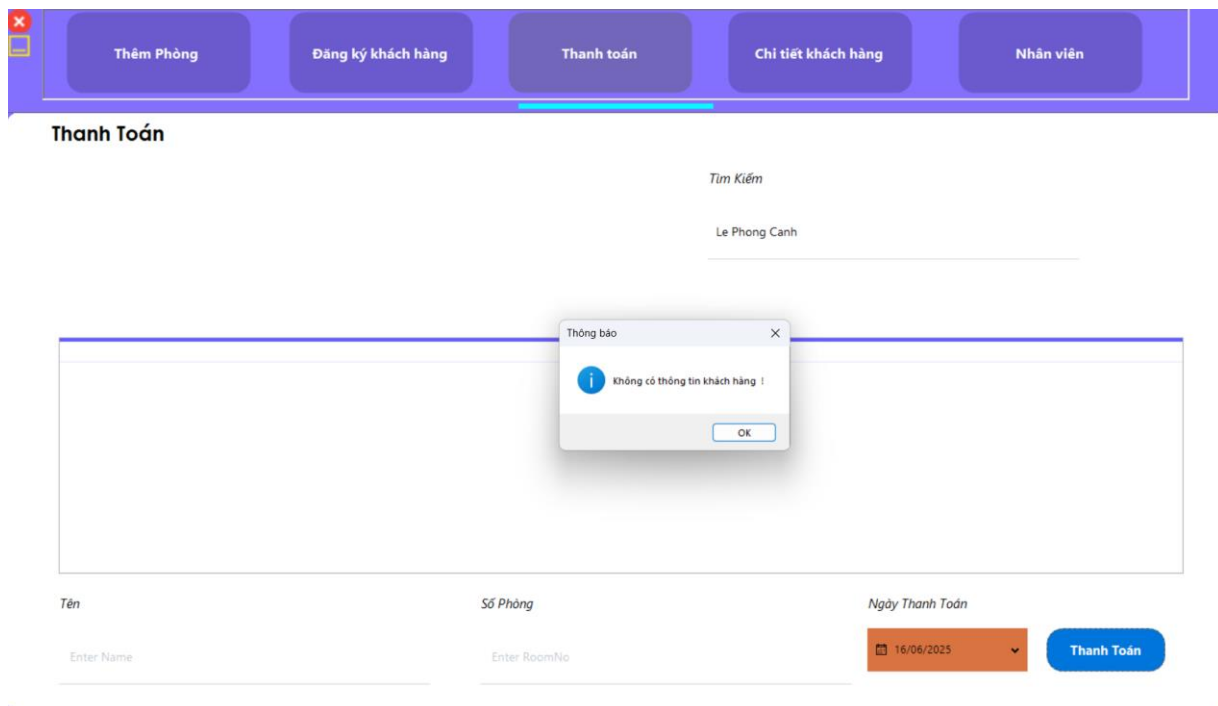
5	Nguyễn Bao Duy	897654335	Việt Nam	Male	08/12/2004	22200041	Khách sạn Khu B - ...	16/06/2025	213	Ac	Double	500000
---	----------------	-----------	----------	------	------------	----------	-----------------------	------------	-----	----	--------	--------

Tên: Nguyễn Bao Duy Số Phòng: 897654335 Ngày Thanh Toán: 16/06/2025 **Thanh Toán**

Success
Khách hàng thanh toán thành công!
OK

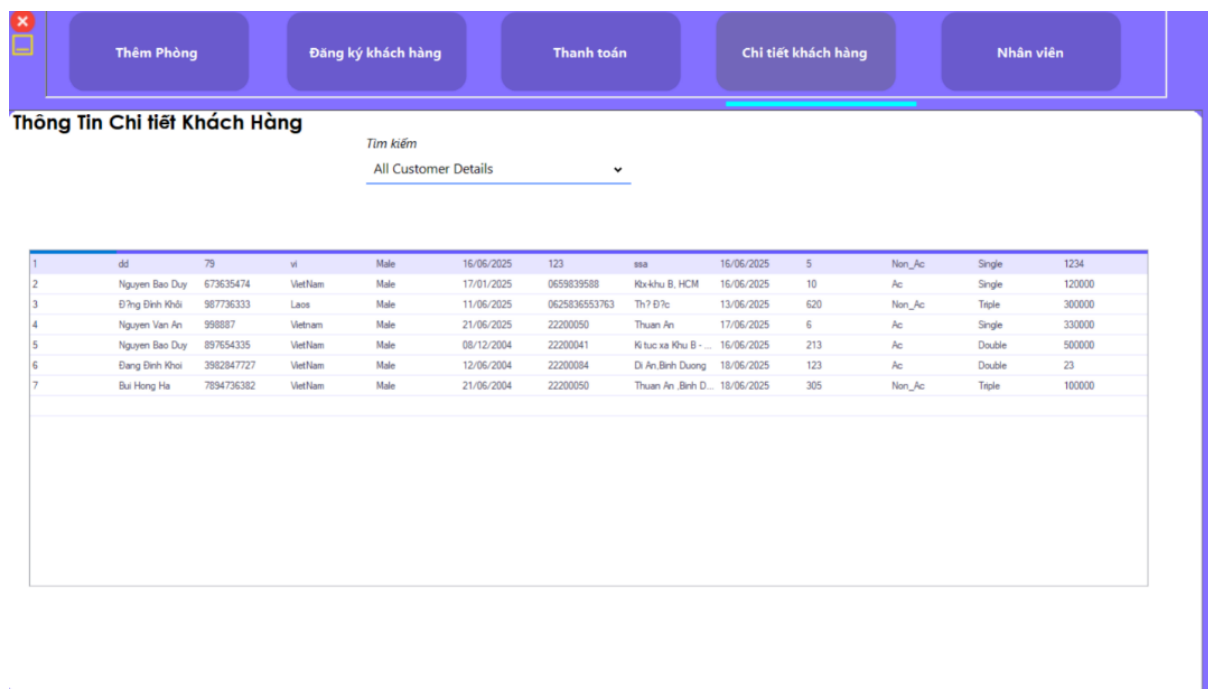
Hình 30: Hộp thoại thông báo xuất hiện khi thanh toán phòng thành công

- Trong giao diện “Thanh Toán”, nếu ta nhập vào tên của khách hàng không có trên hệ thống thì sẽ có thông báo lỗi với nội dung: “Không có thông tin khách hàng !”.



Hình 31: Hộp thoại thông báo xuất hiện khi nhập một tên khách hàng không có trong hệ thống

- Ở giao diện “Chi tiết khách hàng”, các nhân viên có thể xem thông tin chi tiết của tất cả những người thuê phòng (All Customer Details), những khách đang thuê khách sạn (In Hotel Customer) và những khách đã checkout (CheckOut Customer).



Hình 32: Thông tin chi tiết về tất cả các khách hàng

<div> <div>Thêm Phòng</div> <div>Đăng ký khách hàng</div> <div>Thanh toán</div> <div>Chi tiết khách hàng</div> <div>Nhân viên</div> </div>												
<div> <div>Thông Tin Chi tiết Khách Hàng</div> <div> <div>Tìm kiếm</div> <div>In Hotel Customer</div> </div> </div>												
6	Đang Đình Khôi	3982847727	VietNam	Male	12/06/2004	22200084	Di An,Binh Duong	18/06/2025	123	Ac	Double	23
7	Bùi Hồng Hà	7894736382	VietNam	Male	21/06/2004	22200050	Thuan An ,Binh D...	18/06/2025	305	Non_Ac	Triple	100000

Hình 33: Thông tin chi tiết về những khách hàng đang thuê phòng

<div> <div>Thêm Phòng</div> <div>Đăng ký khách hàng</div> <div>Thanh toán</div> <div>Chi tiết khách hàng</div> <div>Nhân viên</div> </div>												
<div> <div>Thông Tin Chi tiết Khách Hàng</div> <div> <div>Tìm kiếm</div> <div>CheckOut Customer</div> </div> </div>												
1	dd	79	vi	Male	16/06/2025	123	ssa	16/06/2025	5	Non_Ac	Single	1234
2	Nguyen Bao Duy	673635474	VietNam	Male	17/01/2025	0659839588	Kbxxkhu B, HCM	16/06/2025	10	Ac	Single	120000
3	Đ'ng Đình Khôi	987736333	Laos	Male	11/06/2025	0625836553763	Th? Đ'c	13/06/2025	620	Non_Ac	Triple	300000
4	Nguyen Van An	998887	Vietnam	Male	21/06/2025	22200050	Thuan An	17/06/2025	6	Ac	Single	330000
5	Nguyen Bao Duy	897654335	VietNam	Male	08/12/2004	22200041	Ki tuc xa Khu B - ...	16/06/2025	213	Ac	Double	500000

Hình 34: Thông tin chi tiết về những khách hàng đã hoàn tất thanh toán tiền phòng và checkout

- Ở giao diện “Nhân Viên”, ta có thể thực hiện ba tác vụ là đăng ký nhân viên, xem thông tin nhân viên và xóa nhân viên.

Nhân Viên

Đăng Ký Nhân Viên | Thông tin nhân viên | Xóa nhân viên

ID - 6

Tên: Nguyen Van An

Số Điện Thoại: 089874637

Giới tính: Female

Email: an123@gmail.com

Tên Người Dùng: Nguyen Van An

Mật Khẩu: 123 | PassWord

Đăng Ký

Hình 35: Điền các thông tin để đăng kí trở thành nhân viên của khách sạn

Nhân Viên

Đăng Ký Nhân Viên | Thông tin nhân viên | Xóa nhân viên

ID - 6

Tên: Nguyen Van An

Số Điện Thoại: 089874637

Giới tính: Female

Email: an123@gmail.com

Tên Người Dùng: Nguyen Van An

Mật Khẩu: 123 | PassWord

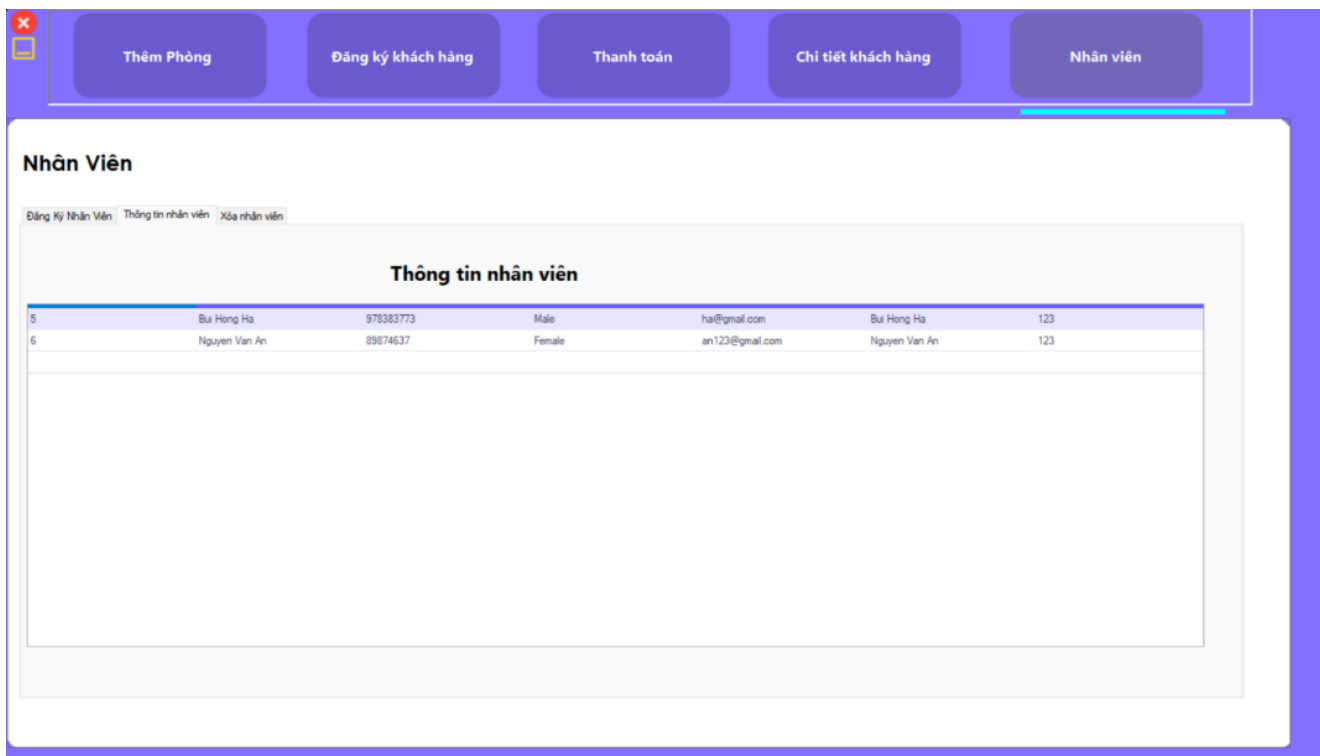
Đăng Ký

Success

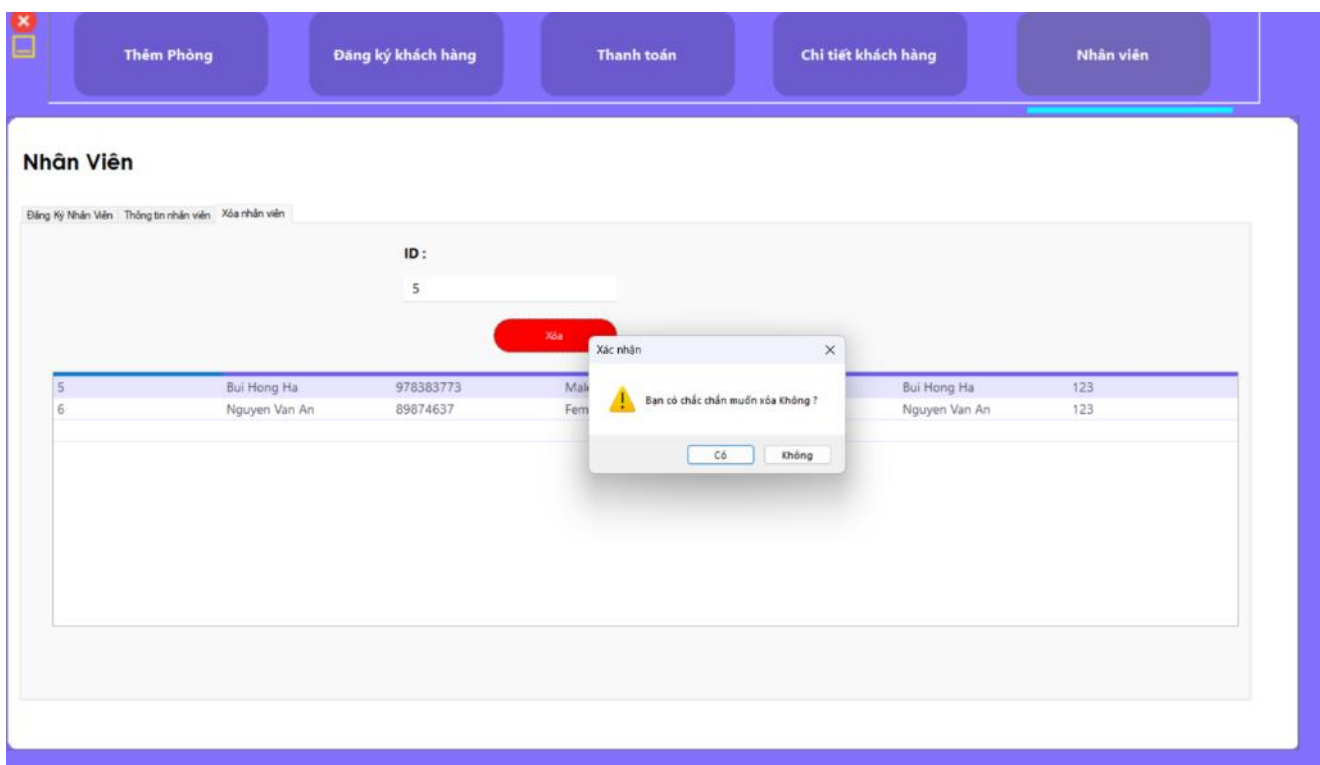
Đăng ký nhân viên thành công!

OK

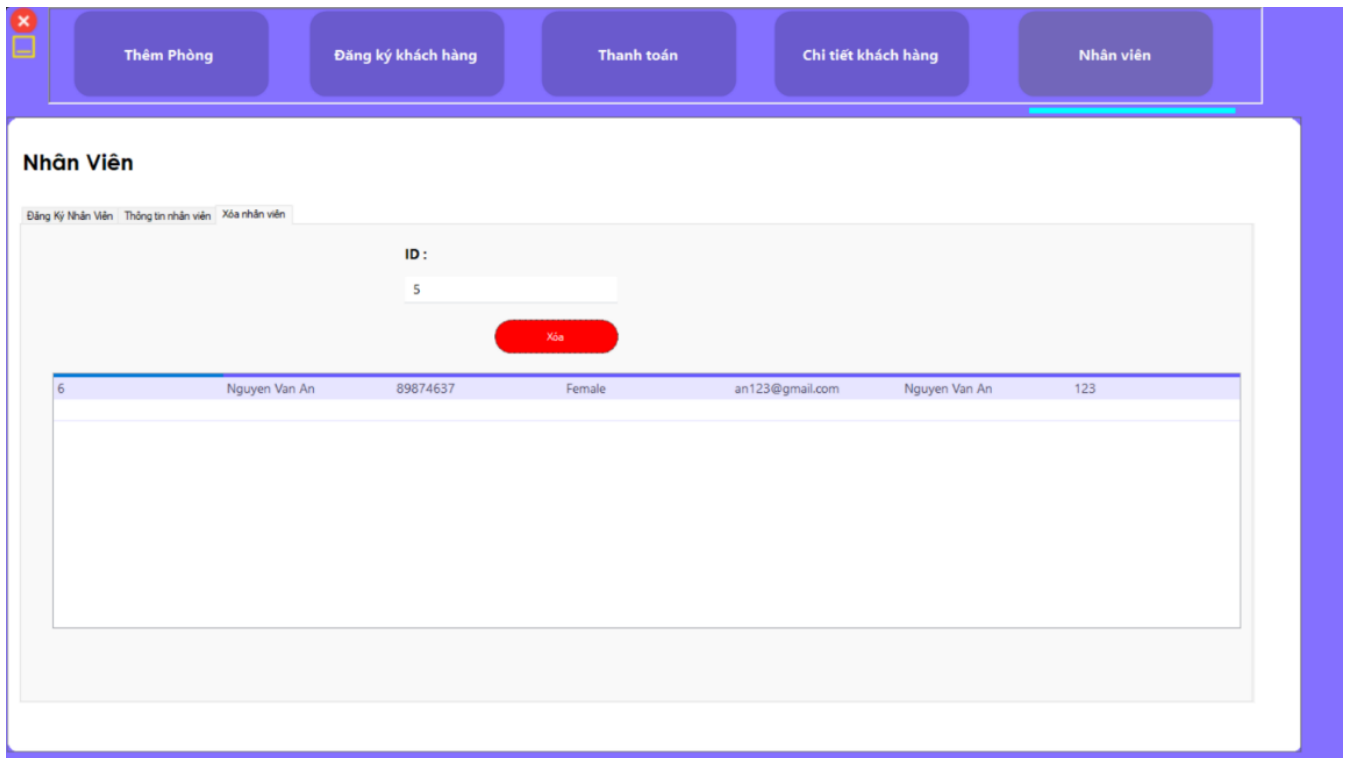
Hình 36: Kết quả sau khi đăng kí nhân viên thành công



Hình 37: Xem thông tin chi tiết về tất cả nhân viên đang làm việc tại khách sạn



Hình 38: Nhập ID của nhân viên cần xóa để hệ thống xóa khỏi danh sách nhân viên



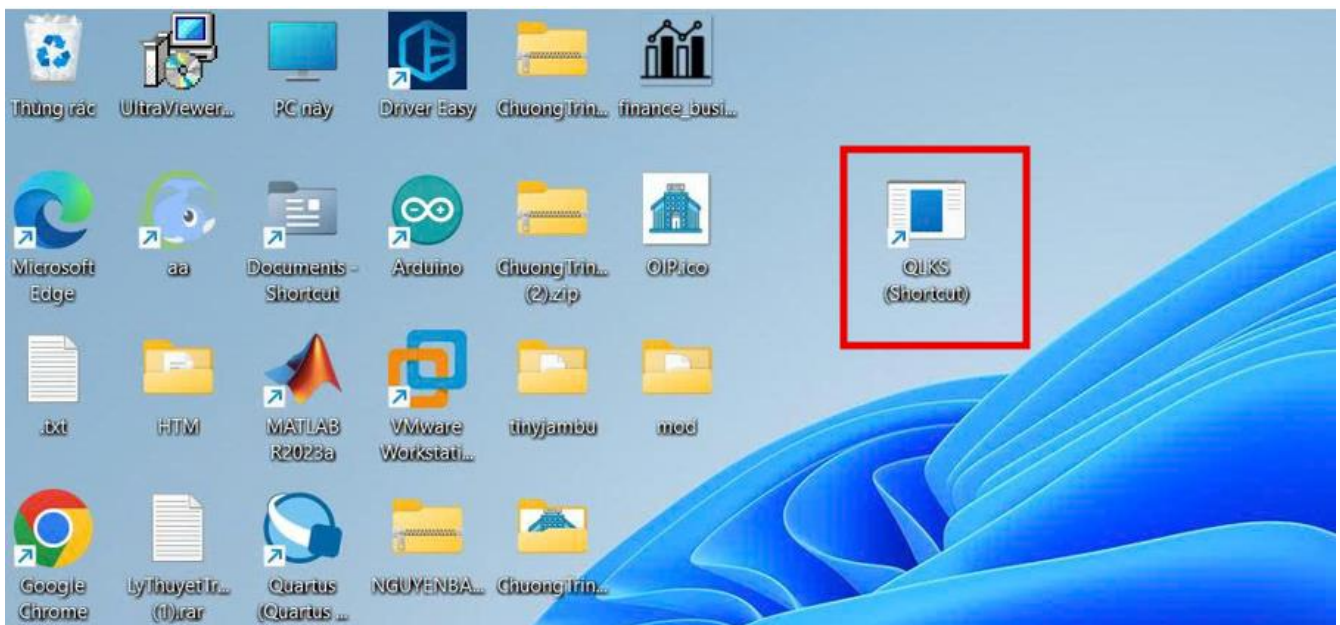
Hình 39: Kết quả khi xóa nhân viên thành công

- Để đóng gói ứng dụng WinForms, WPF thành file cài đặt (.msi hoặc .exe), ta thực hiện các bước:
 - Bước 1 – Cài đặt Extension
 - ❖ Mở Visual Studio.
 - ❖ Vào Extensions → Manage Extensions.
 - ❖ Tìm và cài đặt “Visual Studio Installer Projects” .
 - ❖ Sau khi cài đặt, khởi động lại Visual Studio để extension sẵn sàng.
 - Bước 2 – Tạo Project Installer
 - ❖ Mở giải pháp (solution) chứa ứng dụng WinForms hoặc WPF.
 - ❖ Add New Project vào solution, chọn loại “Setup Project” (Installer).
 - ❖ Đặt tên QLKS Setup.
 - Bước 3 – Kết nối với ứng dụng chính
 - ❖ Trong Setup Project, vào File System tab.
 - ❖ Chọn vào Application Folder → chọn “Add → Project Output...”.
 - ❖ Chọn ứng dụng WinForms/WPF → chọn Primary output (thường chứa exe và dll).
Điều này đảm bảo bộ cài đặt bao gồm file chính và các phụ thuộc.
 - Bước 4 – Giá trị bổ sung
 - ❖ Ta có thể tùy chọn thêm Content Files nếu có tài nguyên như ảnh, file cấu hình.
 - ❖ Thêm shortcut: chọn vào Primary output, chọn Create Shortcut, kéo vào User’s Desktop hoặc Programs Menu để tạo lối tắt khi cài đặt.



Hình 40: Biểu tượng khởi tạo của quản lý khách sạn

- Bước 5 – Thiết lập metadata
 - ❖ Trong Setup Project → Properties: điền thông tin như Product Name, Manufacturer, Version, Install Location.
- Bước 6 – Build installer
 - ❖ Chọn Setup Project trong Solution Explorer.
 - ❖ Chọn Build → Build Solution.
 - ❖ Sau khi hoàn tất, ta vào thư mục giải pháp → bin\Release hoặc bin\Debug của Setup Project, tra sẽ thấy .msi (Microsoft Installer) hoặc .exe bootstrap file.
- Bước 7 – Cài đặt thử
 - ❖ Chạy file .msi hoặc .exe trên máy không có Visual Studio.
 - ❖ Kiểm tra ứng dụng được cài đúng, shortcut hoạt động, tất cả resources hiển thị đúng.



Hình 41: Ứng dụng WinForms quản lý khách sạn trên màn hình Window

CHƯƠNG 4: TỔNG KẾT ĐỒ ÁN

I. Các kiến thức quan trọng đạt được từ đồ án

Thông qua quá trình thực hiện đồ án “**Xây dựng chương trình quản lý khách sạn sử dụng ngôn ngữ lập trình C#**” trong môn học **Kỹ thuật lập trình nâng cao**, nhóm chúng em đã tích lũy được nhiều kiến thức chuyên môn và kỹ năng thực tiễn, điều đó thể hiện qua các ưu điểm mà đồ án đã đạt được, tuy nhiên vẫn còn những nhược điểm và thiếu sót. Cụ thể:

- Ưu điểm:

- **Chức năng đầy đủ, sát với thực tế:** đồ án đã triển khai các chức năng thiết thực của một phần mềm quản lý khách sạn như: quản lý khách hàng, đặt phòng, trả phòng, tra cứu thông tin, lập hóa đơn,... đáp ứng đúng và tương đối đầy đủ các yêu cầu của một hệ thống quản lý vận hành khách sạn cơ bản.
- **Giao diện đơn giản, dễ sử dụng:** giao diện xây dựng bằng Windows Forms trực quan, sử dụng các thành phần quen thuộc như textbox, datagridview, combobox,... giúp người dùng dễ thao tác mà không cần đào tạo nhiều.
- **Cấu trúc chương trình rõ ràng, dễ mở rộng:** các chức năng được phân tách theo từng module cụ thể (khách hàng, phòng, dịch vụ, hóa đơn,...), thuận tiện cho việc nâng cấp, bảo trì hoặc tích hợp thêm tính năng mới.
- **Sử dụng ngôn ngữ lập trình phổ biến:** dựa trên ngôn ngữ C# kết hợp với SQL Server – đây là công nghệ được ứng dụng rộng rãi trong phát triển phần mềm quản lý, có tài liệu tham khảo phong phú, cộng đồng hỗ trợ lớn.
- **Khả năng ứng dụng thực tế cao:** phần mềm có thể được triển khai thực tế cho các khách sạn nhỏ, nhà nghỉ hoặc homestay để giúp tối ưu hóa công tác quản lý và tiết kiệm nhân lực.

- Hạn chế và nhược điểm:

- **Chưa có tính năng phân quyền người dùng:** hệ thống hiện tại chưa phân biệt quyền truy cập giữa quản trị viên và nhân viên, gây hạn chế về tính bảo mật và kiểm soát người dùng.
- **Thiếu tính năng báo cáo, thống kê:** chưa tích hợp chức năng thống kê doanh thu theo ngày, tháng, năm hoặc thống kê lượng khách – đây là các yêu cầu thiết yếu trong vận hành kinh doanh.

- **Kiểm tra dữ liệu đầu vào còn hạn chế:** việc kiểm tra hợp lệ dữ liệu nhập (như định dạng số điện thoại, ngày tháng...) còn đơn giản, dễ dẫn đến lỗi khi người dùng thao tác sai.
- **Chỉ hỗ trợ hoạt động cục bộ:** phần mềm chưa hỗ trợ lưu trữ dữ liệu online hay đồng bộ từ nhiều máy, nên chưa phù hợp với các mô hình khách sạn có nhiều chi nhánh hoặc nhiều quầy lễ tân.

II. Hướng phát triển của đề án

Dựa trên những kết quả đạt được, nhóm chúng em đề xuất 5 hướng phát triển chính sau:

1. Tích hợp cơ sở dữ liệu từ xa (Online/Cloud): hiện tại hệ thống đang sử dụng cơ sở dữ liệu cục bộ (.mdf) trên máy. Trong tương lai, nhóm sẽ tiếp tục tìm hiểu và phát triển để có thể chuyển sang dùng SQL Server online, Firebase hoặc MySQL trên Cloud, cho phép nhiều thiết bị truy cập và quản lý từ xa. Hệ thống sẽ tiếp tục được phát triển lên phiên bản Web (ASP.NET) hoặc Mobile (Xamarin, MAUI) để giúp khách hàng đặt phòng từ xa và nhân viên có thể quản lý dễ dàng trên smartphone, laptop,...

2. Hệ thống quản lý và thống kê nâng cao: trong phần chứng năng, hệ thống sẽ bổ sung thêm chức năng xuất báo cáo doanh thu theo tháng, tỷ lệ đặt phòng hoặc phân tích tình hình kinh doanh theo từng quý trong năm, cũng như có thể quản lý cùng lúc nhiều khách sạn/chi nhánh, áp dụng mô hình multi-branch.

3. Tích hợp thanh toán điện tử: tích hợp API của Momo, VNPAY, ZaloPay giúp khách hàng thanh toán dễ dàng qua ví điện tử. Đây là hình thức thanh toán phổ biến và nhanh chóng, hữu dụng trong thời đại công nghệ phát triển ngày nay.

4. Tích hợp công nghệ AI cho đề xuất phòng: sử dụng các thuật toán học máy (machine learning) để đề xuất phòng phù hợp với nhu cầu khách hàng dựa trên lịch sử đặt phòng, thời gian lưu trú, giá tiền,...

5. Tăng cường bảo mật hệ thống: thêm quản lý phân quyền người dùng (admin, lễ tân, nhân viên dọn phòng,...), mã hóa thông tin khách hàng, log các truy cập,...

III. Phân công nhiệm vụ và tự đánh giá

1. Phân công nhiệm vụ

STT	Nhiệm vụ	Thành viên thực hiện
THỰC HIỆN PHẦN MỀM		
1	Thiết kế và phát triển BackEnd Form Đăng nhập, Dashboard, Thêm Phòng, Đăng ký khách hàng, Thanh toán	Nguyễn Bảo Duy (nhóm trưởng)
2	Thiết kế giao diện và BackEnd Form Thông tin chi tiết khách hàng	Đặng Đình Khôi
3	Thiết kế giao diện và BackEnd Form Quản lý nhân viên	Bùi Hồng Hà
VIẾT BÁO CÁO ĐỒ ÁN		
4	Lời nói đầu, Mục lục, Lời cảm ơn và Tài liệu tham khảo Hoàn thiện báo cáo đồ án môn học	Bùi Hồng Hà
5	Chương 1: Tổng quan về đồ án	Đặng Đình Khôi
6	Chương 2: Cơ sở lý thuyết	Đặng Đình Khôi Bùi Hồng Hà
7	Chương 3: Quá trình thực hiện đồ án	Nguyễn Bảo Duy Bùi Hồng Hà
8	Chương 4: Tổng kết đồ án	Nguyễn Bảo Duy Bùi Hồng Hà Đặng Đình Khôi
<p>Mỗi thành viên trong nhóm em đều cố gắng làm tốt phần nhiệm vụ của mình và có sự phối hợp, hỗ trợ nhiệt tình với các bạn trong các phần nội dung khác.</p>		

Bảng 3: Phân công công việc các thành viên trong nhóm

2. Tự đánh giá

STT	Thành viên	Mức độ hoàn thành
1	Nguyễn Bảo Duy	100%
2	Bùi Hồng Hà	100%
3	Đặng Đình Khôi	100%

Bảng 4: Kết quả tự đánh giá mức độ hoàn thành công việc của các thành viên trong nhóm

3. Đánh giá tổng quát

STT	Nội dung	Điểm tối đa	Nhóm tự đánh giá
1	Dự án sử dụng C#/C++ để thiết kế ứng dụng hoặc game	0.5	0.5
2	Dự án sử dụng một trong các nền tảng: WPF, Win Form, UWP, Unity 3D,... để thiết kế giao diện và chức năng	0.5	0.5
3	Sử dụng các kỹ thuật lập trình: lớp, phương thức, field, properties	1	1
4	Có sử dụng các kỹ thuật kế thừa và đa hình	1	1
5	Có sử dụng interface hoặc abstract class	0.5	0.5
6	Có xử lý ngoại lệ (Exception)	0.5	0.5
7	Trình bày báo cáo trên file pdf (tối thiểu 12 trang A4)	0.5	0.5
7.1	Có trình bày mục tiêu, chức năng của sản phẩm	0.5	0.5
7.2	Có trình bày thuật toán rõ ràng	1	1
7.3	Thiết kế hoàn chỉnh giao diện	0.5	0.5
7.4	Có giải thích code	0.5	0.5
7.5	Minh họa hoạt động của ứng dụng (game) bằng hình ảnh	1	1
7.6	Có bảng phân công nhiệm vụ và đánh giá nhóm	0.5	0.5
8	Có sử dụng nền tảng cộng tác github hoặc tương đương hoặc sử dụng nền tảng đám mây để xây dựng web app hoặc app liên quan đến giao tiếp internet, IoT	0.5	0.5
9	Thể hiện sự hợp tác hiệu quả giữa các thành viên	0.5	0.5
10	Đóng gói ứng dụng WinForms, WPF thành file cài đặt	0.5	0.5

11	Sử dụng các kỹ thuật lập trình: Event	0.5	0.5
12	Code tự viết: <ul style="list-style-type: none"> ○ Trên 90% (1 điểm) ○ Trên 70% (0.75 điểm) ○ Trên 50% (0.5 điểm) ○ Trên 25% (0.25 điểm) 	1	0.5

Bảng 5: Kết quả tự đánh giá mức độ hoàn thành công việc của nhóm
dựa trên tiêu chí của đề án

LỜI CẢM ƠN

Nhóm em xin chân thành bày tỏ lòng biết ơn sâu sắc đến **Thầy Thịnh** – giảng viên đã tận tâm đồng hành, hướng dẫn và hỗ trợ nhóm trong suốt quá trình thực hiện đồ án. Những lời góp ý quý báu, chỉ bảo và nguồn động viên kịp thời của Thầy đã giúp nhóm vượt qua nhiều khó khăn, từ định hướng ban đầu cho đến hoàn thiện báo cáo. Nhóm em cũng xin gửi lời cảm ơn đặc biệt đến các bạn khoá 22 chuyên ngành Điện tử – những người bạn cùng học, cùng chia sẻ kiến thức và nhiệt tình hỗ trợ khi nhóm gặp khó khăn trong quá trình xử lý các lỗi gặp phải khi thực thi chương trình.

Bên cạnh đó, nhóm em không thể quên gửi lời tri ân đến gia đình và bạn bè – những người luôn ở bên động viên về tinh thần, giúp chúng em giữ vững tinh thần và quyết tâm hoàn thành đồ án. Sự quan tâm và niềm tin của mọi người chính là nguồn động lực lớn để nhóm cố gắng đến phút cuối cùng.

Cuối cùng, nhóm em rất mong nhận được những ý kiến đóng góp quý báu từ Thầy và mọi người để đồ án có thể hoàn thiện hơn, khắc phục những hạn chế và mở ra hướng phát triển mới trong tương lai. Nhóm em cũng hy vọng rằng kết quả của đồ án không chỉ dừng lại ở đây mà sẽ tiếp tục được cải tiến, ứng dụng thực tế để mang lại nhiều giá trị thiết thực.

Một lần nữa, nhóm em xin chân thành cảm ơn tất cả mọi người đã đồng hành và giúp đỡ trong hành trình đầy ý nghĩa này!

TÀI LIỆU THAM KHẢO

- [1] Programming Windows Forms in C# - Charles Petzold
- [2] Windows Forms Programming with C# - Chris Sells
- [3] Head First C# - Andrew Stellman & Jennifer Greene
- [4] Clean Code: A Handbook of Agile Software Craftsmanship - Robert C. Martin
- [5] C# Windows Forms Application Tutorial