**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**KHOA ĐIỆN TỬ - VIỄN THÔNG**

...................\*\*\*...................



**BÁO CÁO ĐỒ ÁN**

**BỘ MÔN: LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG**

**ĐỀ TÀI: CHƯƠNG TRÌNH QUẢN LÝ KHÁCH SẠN**

**Nhóm : 15**

**Lớp : 21Nhung**

**Giảng viên hướng dẫn : Ths. Lê Đức Trị**

*TP. Hồ Chí Minh, tháng 4, năm 2024*

**DANH SÁCH THÀNH VIÊN**

|  |  |  |
| --- | --- | --- |
| STT | Họ và tên | MSSV |
| 1 | Nguyễn Thừa Vũ Hiệp | 21200288 |
| 2 | Hồ Công Hiếu | 21200289 |
| 3 | Nguyễn Nhất Huy | 21200294 |

**NHẬN XÉT CỦA GIẢNG VIÊN**

**MỤC LỤC**

[LỜI MỞ ĐẦU 1](#_Toc163809539)

[KHAI BÁO CÁC LỚP VÀ CÁC HẰNG SỐ LIÊN QUAN 2](#_Toc163809540)

[I. Sơ đồ phân lớp 2](#_Toc163809541)

[II. Giải thích các lớp 2](#_Toc163809542)

[1. Lớp DataProcessor 2](#_Toc163809543)

[2. Lớp Display 3](#_Toc163809544)

[3. Lớp Room 3](#_Toc163809545)

*[4. Lớp Account 3](#_Toc163809546)*

[5. Lớp Guest 3](#_Toc163809547)

[6. Lớp Admin 4](#_Toc163809548)

[III. Các hằng số liên quan 4](#_Toc163809549)

[ĐỊNH NGHĨA CÁC PHƯƠNG THỨC CỦA CÁC LỚP 5](#_Toc163809550)

[I. Lớp DataProcessor 5](#_Toc163809551)

[II. Lớp Display 6](#_Toc163809552)

[III. Lớp Room 8](#_Toc163809553)

[IV. Lớp Guest 10](#_Toc163809554)

[V. Lớp Admin 12](#_Toc163809555)

[1. Phương thức quản trị phòng 13](#_Toc163809556)

[2. Phương thức quản trị khách hàng 19](#_Toc163809557)

[THỰC HIỆN CHƯƠNG TRÌNH 28](#_Toc163809558)

[BẢNG KẾ HOẠCH PHÂN CÔNG NHIỆM VỤ 33](#_Toc163809559)

[TÀI LIỆU THAM KHẢO 34](#_Toc163809560)

# LỜI MỞ ĐẦU

Trong thời đại hiện đại, ngành du lịch và khách sạn không chỉ là một phần quan trọng của nền kinh tế toàn cầu mà còn đóng vai trò không thể phủ nhận trong việc tạo ra cơ hội việc làm và nâng cao chất lượng cuộc sống của con người. Trong bối cảnh mà du lịch và nhu cầu một nơi nghỉ ngơi trở thành một ngành công nghiệp đầy tiềm năng và cạnh tranh, việc quản lý khách sạn hiệu quả đóng vai trò quyết định đối với sự thành công của các doanh nghiệp trong lĩnh vực này.

Điều này không chỉ đòi hỏi sự chuyên môn cao về quản lý vận hành và tiêu chuẩn dịch vụ mà còn đòi hỏi sự linh hoạt trong việc đáp ứng nhanh chóng với sự biến đổi của nhu cầu và mong muốn của khách hàng. Vì vậy chúng em quyết định chọn đề tài “**Chương trình quản lý khách sạn”.**

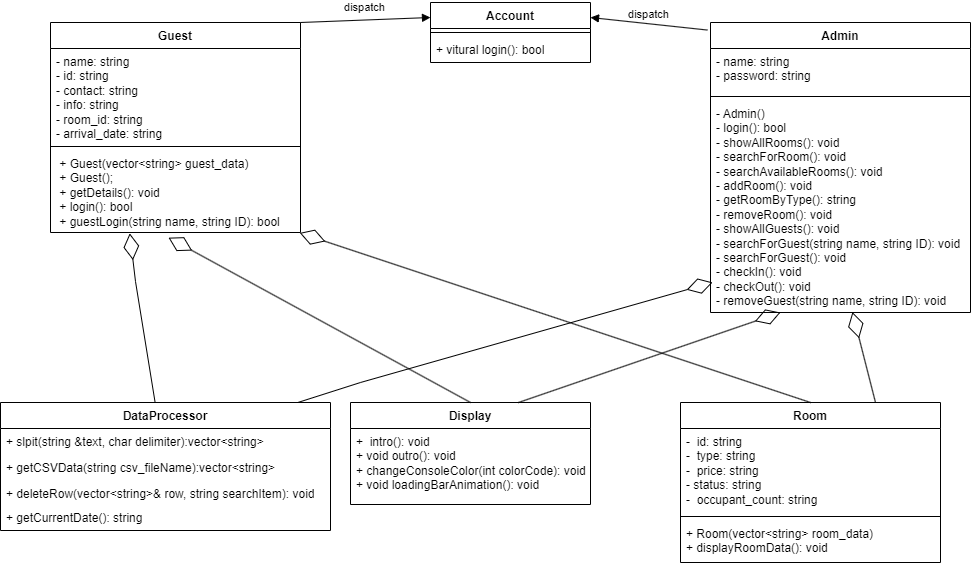
Chương trình quản lý khách sạn bằng C++ là một công cụ quan trọng để tự động hóa và tối ưu hóa các quy trình quản lý trong ngành khách sạn. Việc sử dụng ngôn ngữ lập trình C++ giúp chúng ta xây dựng một hệ thống linh hoạt, hiệu quả và dễ bảo trì. Tích hợp các tính năng như đặt phòng, quản lý thông tin khách hàng, thanh toán và báo cáo, chương trình này không chỉ giúp tăng cường trải nghiệm của khách hàng mà còn giúp tối ưu hóa hoạt động và quản lý tài nguyên của khách sạn. Trong chương trình này, chúng ta sẽ khám phá cách thiết kế và triển khai các tính năng quản lý khách sạn bằng C++, từ việc xây dựng cấu trúc dữ liệu đến việc lập trình giao diện người dùng thân thiện.

Trong quá trình tìm hiểu và thực hiện viết chương trình, nhóm chúng em đã vận dụng kiến thức môn “**Lập trình hướng đối tượng”** kết hợp tìm hiểu các thông tin để tìm cách vận hành chương trình quản lý khách sạn một cách phù hợp nhất. Tuy nhiên, do hạn chế về thời gian, kiến thức cũng như kỹ năng tích và sự hiểu biết về quản lý khách sạn, chúng em vẫn chưa thể hoàn thiện đầy đủ, chi tiết và chính xác mọi nội dung cho chương trình. Vì vậy, chúng em rất mong có thể những nhận xét và góp ý từ thầy để có thể hoàn chỉnh đề tài hơn.

Chúng em xin chân thành cảm ơn!

# KHAI BÁO CÁC LỚP VÀ CÁC HẰNG SỐ LIÊN QUAN

1. **Sơ đồ phân lớp**

****

1. **Giải thích các lớp**
   * + 1. **Lớp DataProcessor**

class DataProcessor {

public:

vector<string> split(const string& text, char delimiter);

vector<string> getCSVData(string csv\_filename);

void deleteRow(vector<string>& row, string searchItem);

string getCurrentDate();

};

**Công dụng:** Xử lý dữ liệu.

* + - 1. **Lớp Display**

class Display{

public:

void intro();

void outro();

void changeConsoleColor(int colorCode);

void loadingBarAnimation();

};

**Công dụng:** Hiển thị thông tin và tương tác với người dùng.

* + - 1. **Lớp Room**

class Room {

private:

string id, type, price, status, occupant\_count;

public:

Room(vector<string> room\_data);

void displayRoomData();

};

**Công dụng:** Quản lý thông tin phòng.

* + - 1. **Lớp Account**

class Account{

public:

virtual bool login() = 0;

};

**Công dụng:** Cung cấp chức năng đăng nhập ảo.

* + - 1. **Lớp Guest**

class Guest: public Account {

private:

string name, id, contact\_info, room\_id, arrival\_date;

public:

Guest(vector<string> guest\_data);

Guest();

void getDetails();

bool login();

bool guestLogin(string name, string ID);

};

**Công dụng:** Quản lý thông tin khách hàng.

* + - 1. **Lớp Admin**

class Admin: public Account {

private:

string name, password;

public:

Admin();

bool login();

void showAllRooms();

void searchForRoom();

void searchAvailableRooms();

void addRoom();

string getRoomByType();

void removeRoom();

void showAllGuests();

void searchForGuest(string name, string ID);

void searchForGuest();

void checkIn();

void removeGuest(string name, string ID);

void checkOut();

};

**Công dụng:** Quản lý các chức năng của nhân viên khách sạn.

1. **Các hằng số liên quan**

const int a = 97;//admin

const int g = 103;//guest

const int e = 101;//exit

**Công dụng:** Mã ASCII cho các chữ cái được sử dụng trong chương trình.

enum ColorCode {

BLACK = 0,

BLUE = 1,

GREEN = 2,

CYAN = 3,

RED = 4,

MAGENTA = 5,

YELLOW = 6,

WHITE = 7,

GRAY = 8,

LIGHT\_BLUE = 9,

LIGHT\_GREEN = 10,

LIGHT\_CYAN = 11,

LIGHT\_RED = 12,

LIGHT\_MAGENTA = 13,

LIGHT\_YELLOW = 14,

BRIGHT\_WHITE = 15

};

**Công dụng:** Xác định mã màu cho văn bản bảng điều khiển.

# ĐỊNH NGHĨA CÁC PHƯƠNG THỨC CỦA CÁC LỚP

1. **Lớp DataProcessor**

Hàm để tách dữ liệu từ mỗi dòng của tệp CSV và trả về một vector chứa các giá trị vừa tách.

VD: split("1,Ho Hieu,1234567890,101,12-12-2021", ',')

=> ["1", "John Doe", "1234567890", "101", "12-12-2021"]

vector<string> DataProcessor::split(const string& text, char delimiter){

vector<string> tokens; // Vector để lưu trữ các dữ liệu vừa tách

string token; // Chuỗi để lưu trữ dữ liệu vừa tách

int i = 0;

while (i < text.length()) {

if(text[i] == delimiter) {

// Thêm token vào vector tokens khi tìm thấy dấu phân tách

tokens.push\_back(token);

token = ""; // Đặt lại token thành chuỗi rỗng

}

else token += text[i]; // Nếu không tìm thấy dấu phân tách, thêm ký tự vào token

i++;

}

// Khi đến cuối dòng, thêm giá trị cuối cùng vào vector

if(token.length() > 0) tokens.push\_back(token);

return tokens;

}

Hàm để đọc dữ liệu từ tệp CSV và trả về tất cả các hàng của tệp dưới dạng mảng vector

VD: getCSVData("data/Room\_Details.csv")

=> ["1,Single,1000,0,Available\n", "2,Double,2000,0,Available\n", ...]

vector<string> DataProcessor::getCSVData(string csv\_filename) {

ifstream file; // Đối tượng file để đọc từ tệp

string data; // Dữ liệu từng dòng được lưu trữ trong biến này

vector<string> row; // Dữ liệu của tất cả các dòng được lưu trữ trong vector này

file.open(csv\_filename);

while(getline(file, data)) { // Đọc đến cuối tệp

row.push\_back(data); // Thêm 1 dòng vào vector

}

file.close();

return row;

}

Hàm để xóa một dòng dữ liệu khỏi cơ sở dữ liệu

VD: deleteRow(["1,Single,1000,0,Available\n", "2,Double,2000,0,Available\n", ...], "1")

=> ["2,Double,2000,0,Available", ...]

void DataProcessor::deleteRow(vector<string> &row, string searchItem) {

for (int i = 0; i < row.size(); i++) {

if(row[i].find(searchItem) != string::npos) {

row.erase(row.begin() + i);

i--;

}

}

}

Hàm để lấy ngày hiện tại và định dạng nó thành chuỗi

VD: getCurrentDate() => "12-04-2024"

string DataProcessor::getCurrentDate() {

// Lấy thời gian hiện tại

time\_t now = time(nullptr);

// Tạo một bộ đệm để lưu trữ ngày được định dạng

char buffer[80];

// Định dạng thời gian hiện tại thành chuỗi với định dạng mong muốn

strftime(buffer, sizeof(buffer), "%d-%m-%Y", localtime(&now));

// Chuyển đổi bộ đệm thành chuỗi và trả về nó

return string(buffer);

}

1. **Lớp Display**

Phương thức ‘changeConsoleColor’ trong lớp ‘Display’ nhận vào một mã màu và thay đổi màu của văn bản trên console tương ứng.

// Hàm để thay đổi màu của console

void Display::changeConsoleColor(int colorCode) {

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hConsole, colorCode);

}

Hiệu ứng thanh tải

VD: loadingBarAnimation()

=> [ ] 0%

=> [=====================> ] 50%

=> [===============================> ] 75%

=> [=========================================>] 100%

void Display::loadingBarAnimation() {

const int totalProgress = 100; // Giá trị tổng tiến trình (ví dụ: 100%)

const int barWidth = 50; // Độ rộng của thanh tải trong ký tự

for (int progress = 0; progress <= totalProgress; ++progress) {

int completedWidth = barWidth \* progress / totalProgress;

cout << "\r[";

for (int i = 0; i < barWidth; ++i) {

if (i < completedWidth) {

cout << "=";

} else {

cout << " ";

}

}

cout << "] " << progress << "%";

cout.flush();

// Thêm một độ trễ nhỏ để điều khiển tốc độ của hiệu ứng

Sleep(5);

}

cout << endl;

}

Phương thức ‘intro’ trong lớp ‘Display’ hiển thị màn hình giới thiệu khi bắt đầu chương trình.



Phương thức ‘outro’ trong lớp ‘Display’ hiển thị màn hình kết thúc khi chương trình kết thúc.



1. **Lớp Room**

Hàm tạo để tạo đối tượng ‘Room’ mà ‘Admin’ có thể sử dụng.

Đây là constructor của lớp ‘Room’. Nó nhận vào một vector của chuỗi ‘room\_data’ và khởi tạo các thuộc tính của đối tượng ‘Room’.

VD: Room room1({"1", "Single", "1000", "0", "Available"})

Room::Room(vector<string> room\_data) {

this->id = room\_data[0]; // id phòng

this->type = room\_data[1]; // loại phòng

this->price = room\_data[2] + "/night"; // giá / 1 phòng

this->occupant\_count = room\_data[3]; // số người ở

this->status = room\_data[4]; // trạng thái phòng

}

Phương thức ‘displayRoomData’ trong lớp ‘Room’ hiển thị thông tin chi tiết về một phòng: id, loại phòng, giá, số người ở, trạng thái.

void Room::displayRoomData() {

string border = "+--------------+-----------------------------------+";

Sleep(900);

cout << left; // Thiết lập căn lề trái cho dữ liệu

cout << border << endl;

cout << "| Room Id | " << setw(34) << id << "|" << endl;

// Đặt màu chữ trên console dựa trên loại phòng

if (type == "Single") {

dis.changeConsoleColor(GRAY);

} else if (type == "Double") {

dis.changeConsoleColor(CYAN);

}else if (type == "Triple") {

dis.changeConsoleColor(LIGHT\_MAGENTA);

}else {

dis.changeConsoleColor(YELLOW);

}

cout << border << endl;

cout << "| Room Type | " << setw(34) << type << "|" << endl;

cout << border << endl;

// Đặt màu chữ trên console về mặc định (trắng)

dis.changeConsoleColor(BRIGHT\_WHITE);

cout << "| Price | " << setw(34) << price << "|" << endl;

cout << border << endl;

cout << "| Occupant No. | " << setw(34) << occupant\_count << "|" << endl;

cout << border << endl;

// Đặt màu chữ trên console dựa trên trạng thái

if (status == "Unavailable") {

dis.changeConsoleColor(LIGHT\_RED);

} else if (status == "Available") {

dis.changeConsoleColor(LIGHT\_GREEN);

}

cout << border << endl;

cout << "| Status | " << setw(34) << status << "|" << endl;

cout << border << endl << endl;

// Đặt màu chữ trên console về mặc định (trắng)

dis.changeConsoleColor(BRIGHT\_WHITE);

}

1. **Lớp Guest**

Hàm tạo để tạo đối tượng ‘Guest’ mà ‘Admin’ có thể sử dụng, hiển thị thông tin chi tiết về khách.

Đây là constructor của lớp ‘Guest’. Nó nhận vào một vector của chuỗi ‘guest\_data’ và khởi tạo các thuộc tính của đối tượng ‘Guest’.

VD: guest1({"Y678", "Ho Cong Hieu", "1234567890", "101", "10-04-2024"})

Guest::Guest(vector<string> guest\_data) {

this->id = guest\_data[0]; // id khách

this->name = guest\_data[1]; // tên khách

this->contact\_info = guest\_data[2]; // Số điện thoại

this->room\_id = guest\_data[3]; // id phòng

this->arrival\_date = guest\_data[4]; // ngày đến

};

Constructor mặc định

Guest::Guest() {};

Phương thức ‘getDetails’ trong lớp ‘Guest’ hiển thị thông tin chi tiết về một khách hàng. Hiển thị thông tin khách: id, tên, thông tin liên hệ, id phòng, ngày đến.

void Guest::getDetails() {

Sleep(900);

cout << left; // Thiết lập căn lề trái cho dữ liệu

cout << "+---------------+-----------------------------------+" << endl;

cout << "| Guest Id | " << setw(34) << id << "|" << endl;

dis.changeConsoleColor(LIGHT\_YELLOW);

cout << "+---------------+-----------------------------------+" << endl;

cout << "| Guest Name | " << setw(34) << name << "|" << endl;

cout << "+---------------+-----------------------------------+" << endl;

dis.changeConsoleColor(BRIGHT\_WHITE);

cout << "| Contact Info | " << setw(34) << contact\_info << "|" << endl;

cout << "+---------------+-----------------------------------+" << endl;

cout << "| Room Id | " << setw(34) << room\_id << "|" << endl;

cout << "+---------------+-----------------------------------+" << endl;

cout << "| Arrival Date | " << setw(34) << arrival\_date << "|" << endl;

cout << "+---------------+-----------------------------------+" << endl << endl;

}

Các đường dẫn tới tập tin dữ liệu.

//ĐƯỜNG DẪN TỚI TẬP TIN DỮ LIỆU PHÒNG CSV

string ROOM\_DATA = "data/Room\_Details.csv";

//ĐƯỜNG DẪN TỚI TẬP TIN DỮ LIỆU KHÁCH CSV

string GUEST\_DATA = "data/Guest\_Data.csv";

//ĐƯỜNG DẪN TỚI TẬP TIN DỮ LIỆU ADMIN CSV

string ADMIN\_DATA = "data/Admin\_data.csv";

Đoạn mã thực hiện chức năng đăng nhập cho khách hàng trong hệ thống khách sạn.

bool Guest::login() {

bool login\_status = false;

string guest\_name,f\_name, l\_name, guest\_id;

// Lấy dữ liệu từ tệp Guest\_Data.csv

vector<string> guest\_database = dp.getCSVData(GUEST\_DATA);

do {

// Nhập thông tin khách để đăng nhập

cout << "\nEnter First name (For testing, Enter \"Hieu\"): ";

cin >> f\_name;

cout << "\nEnter Last name(For testing, Enter \"Ho\"): ";

cin >> l\_name;

cout << "\nEnter ID(For testing, Enter \"01\"): ";

cin >> guest\_id;

guest\_name = l\_name + " " + f\_name;

// Duyệt qua tất cả các dòng trong tệp

for(int i = 0; i < guest\_database.size(); i++) {

// Tách dữ liệu từng dòng

vector<string> guest\_data = dp.split(guest\_database[i], ',');

// So sánh id và tên khách

if (guest\_data[0] == guest\_id && guest\_data[1] == guest\_name) {

login\_status = true;

break;

}

}

// Thông báo kết quả đăng nhập

if(!login\_status) {

dis.changeConsoleColor(LIGHT\_RED);

cout << "Invalid Login! Please try again\n";

}

else {

dis.changeConsoleColor(LIGHT\_GREEN);

cout << "Welcome " << guest\_name;

}

// Đặt màu chữ trên console về mặc định (trắng)

dis.changeConsoleColor(BRIGHT\_WHITE);

} while (!login\_status);

return login\_status;

}

Phương thức ‘guestLogin’ trong lớp ‘Guest’ kiểm tra xem tên và ID của khách hàng có khớp với dữ liệu trong cơ sở dữ liệu hay không.

bool Guest::guestLogin(string name, string ID) {

bool login\_status = false;

vector<string> guest\_database = dp.getCSVData(GUEST\_DATA);

//do {

for(int i = 0; i < guest\_database.size(); i++) {

vector<string> guest\_data = dp.split(guest\_database[i], ',');

if (guest\_data[0] == ID && guest\_data[1] == name) {

dis.changeConsoleColor(LIGHT\_GREEN);

cout << "Welcome " << name;

login\_status = true;

break;

}

}

if(!login\_status) {

dis.changeConsoleColor(LIGHT\_RED);

cout << "Invalid Login! Please try again\n";

}

else {

dis.changeConsoleColor(LIGHT\_GREEN);

cout << "Welcome " << name;

}

// Đặt màu chữ trên console về mặc định (trắng)

dis.changeConsoleColor(WHITE);

// } while (!login\_status);

return login\_status;

}

1. **Lớp Admin**

Hiển thị tất cả các phòng và khách hàng, tìm kiếm phòng và khách hàng, thêm và xóa phòng và khách hàng.

Phương thức ‘login’ trong lớp ‘Admin’ kiểm tra xem ID và mật khẩu của quản trị viên có khớp với dữ liệu trong cơ sở dữ liệu hay không.

// Constructor mặc định

Admin::Admin() {}

bool Admin::login() {

bool login\_status = false;

string admin\_id, admin\_password;

// Lấy dữ liệu từ tệp Admin\_data.csv

vector<string> admin\_database = dp.getCSVData(ADMIN\_DATA); // admin\_database = ["admin,admin", "admin1,admin1", ...]

do {

// Nhập thông tin admin để đăng nhập

cout << "\nEnter Admin ID (For testing, Enter admin): ";

cin >> admin\_id;

cout << "\nEnter password (For testing, Enter admin): ";

cin >> admin\_password;

// Duyệt qua tất cả các dòng trong tệp

for(int i = 0; i < admin\_database.size(); i++) {

// Tách dữ liệu từng dòng

vector<string> admin\_data = dp.split(admin\_database[i], ','); //admin\_data = {id, password}

// So sánh id và password

if (admin\_data[0] == admin\_id && admin\_data[1] == admin\_password) {

login\_status = true;

break;

}

}

// Thông báo kết quả đăng nhập

if(!login\_status) {

dis.changeConsoleColor(LIGHT\_RED);

cout << "Invalid Login! Please try again\n";

}

else {

dis.changeConsoleColor(LIGHT\_GREEN);

cout << "Welcome " << admin\_id;

}

// Đặt màu chữ trên console về mặc định (trắng)

dis.changeConsoleColor(WHITE);

} while (!login\_status); // Lặp lại quá trình đăng nhập nếu không thành công

return login\_status;

}

**Phương thức quản trị phòng**

Phương thức ‘showAllRooms’ trong lớp ‘Admin’ hiển thị thông tin về tất cả các phòng trong cơ sở dữ liệu. Lấy dữ liệu từ tệp Room\_Details.csv.

void Admin::showAllRooms() {

// Lấy dữ liệu từ tệp Room\_Details.csv

vector<string> rows = dp.getCSVData(ROOM\_DATA); // rows = ["1,Single,1000,0,Available", "2,Double,2000,0,Available", ...]

cout << "\nDisplaying all Rooms...\n";

dis.loadingBarAnimation();

for (int i = 0; i < rows.size(); i++) { // Duyệt qua tất cả các dòng trong tệp

// Tách dữ liệu từng dòng

vector<string> data = dp.split(rows[i], ',');// data = {id, type, price, occupant\_count, status}

Room room(data); // Tạo một đối tượng phòng từ dữ liệu dòng hiện tại

room.displayRoomData(); // Hiển thị thông tin phòng

}

}

Phương thức ‘searchForRoom’ trong lớp ‘Admin’ cho phép tìm kiếm một phòng dựa trên ID phòng.

void Admin::searchForRoom() {

bool isRoomFound = false;

// Lấy dữ liệu từ tệp Room\_Details.csv

vector<string> rows = dp.getCSVData(ROOM\_DATA); // rows = ["1,Single,1000,0,Available", "2,Double,2000,0,Available", ...]

string room\_id;

cout << "\nEnter Room ID: ";

cin >> room\_id;

cout << "\nSearching for Room " << room\_id << " ...\n";

dis.loadingBarAnimation();

// Duyệt qua tất cả các phòng

for (int i = 0; i < rows.size(); i++) {

// Nếu id phòng trùng với id nhập vào

if (dp.split(rows[i], ',')[0] == room\_id) {

isRoomFound = true;

// Tạo một đối tượng phòng từ dữ liệu phòng tìm thấy

vector<string> data = dp.split(rows[i], ','); // data = {id, type, price, occupant\_count, status}

Room room(data);

// Hiển thị thông tin phòng

room.displayRoomData();

break;

}

}

if(!isRoomFound) {

dis.changeConsoleColor(LIGHT\_RED);

cout << "\nRoom not found. Please try again\n";

dis.changeConsoleColor(WHITE);

}

}

Phương thức ‘searchAvailableRooms’ trong lớp ‘Admin’ tìm kiếm và hiển thị tất cả các phòng có sẵn.

void Admin::searchAvailableRooms() {

vector<string> rows = dp.getCSVData(ROOM\_DATA); //Lấy dữ liệu từ tệp Room\_Details.csv

bool areRoomsFound = false;

dis.loadingBarAnimation();

// Duyệt qua tất cả các phòng

for (int i = 0; i < rows.size(); i++) {

// Tách dữ liệu từng dòng

vector<string> data = dp.split(rows[i], ','); // data = {id, type, price, occupant\_count, status}

// Nếu phòng chưa được sử dụng hoặc chưa đạt giới hạn

if (data[4] == "Available") {

areRoomsFound = true;

Room room(data);

room.displayRoomData();

}

}

if(!areRoomsFound)

{

dis.changeConsoleColor(LIGHT\_RED);

cout << "\nNo available Rooms. Check again later\n";

dis.changeConsoleColor(WHITE);

}

}

Phương thức ‘addRoom’ trong lớp ‘Admin’ cho phép thêm một phòng mới vào cơ sở dữ liệu.

void Admin::addRoom() {

bool success, error;

string id, type, status, price, occupant\_count;

// Lấy dữ liệu từ tệp Room\_Details.csv

vector <string> rows = dp.getCSVData(ROOM\_DATA); // rows = ["1,Single,1000,0,Available", "2,Double,2000,0,Available", ...]

do {

success = true;

cout << "\nEnter Room ID: ";

cin >> id;

bool doesRoomExist = false;

// Kiểm tra xem phòng đã tồn tại chưa

for (int i = 0; i < rows.size(); i++) {

vector<string> data = dp.split(rows[i], ','); // data = {id, type, price, occupant\_count, status}

if(data[0] == id) { // So sánh id phòng vừa nhập

doesRoomExist = true;

break;

}

}

if(doesRoomExist) { // Nếu phòng đã tồn tại

dis.changeConsoleColor(LIGHT\_RED);

cout << "\nERROR!!\nRoom already exists. Enter another ID\n";

dis.changeConsoleColor(WHITE);

success = false;

}

else { // Nếu phòng chưa tồn tại

int choice;

do {

error = false;

cout << "\nSelect Room Type: \n1 - Single\n2 - Double\n3 - Triple\n4 - Quadruple\nYour Choice: ";

cin >> choice;

switch (choice) {

case 1:

type = "Single";

price = "1000";

break;

case 2:

type = "Double";

price = "2000";

break;

case 3:

type = "Triple";

price = "3000";

break;

case 4:

type = "Quadruple";

price = "4000";

break;

default:

cout << "Invalid Option";

error = true;

break;

}

} while(error);

occupant\_count = "0", status = "Available"; // Mặc định số người ở là 0 và trạng thái là "Available"

string new\_room\_data = id + "," + type+ "," + price + "," + occupant\_count + "," + status;

// Lấy dữ liệu từ tệp Room\_Details.csv

vector<string> room\_details = dp.getCSVData(ROOM\_DATA); // room\_details = ["1,Single,1000,0,Available", "2,Double,2000,0,Available", ...]

room\_details.push\_back(new\_room\_data); // Thêm phòng mới vào cuối danh sách

ofstream outStream(ROOM\_DATA);

// Thêm tất cả các phòng vào tệp Room\_Details.csv

for (int i = 0; i < room\_details.size(); i++)

outStream << room\_details[i] << endl;

outStream.close();

// Thông báo phòng đã được thêm thành công

dis.changeConsoleColor(LIGHT\_GREEN);

cout << "\nRoom successfully Added\n";

success = true;

dis.changeConsoleColor(WHITE);

}

} while (!success);

}

Phương thức ‘getRoomByType’ trong lớp ‘Admin’ cho phép tìm kiếm và hiển thị tất cả các phòng có sẵn theo loại phòng. Dùng kết hợp với phương thức ‘checkIn()’ để hiển thị thông tin phòng trống cho khách hàng khi họ đến đặt phòng

string Admin::getRoomByType() {

int choice;

bool isValidChoice;

string room\_type;

// Lấy dữ liệu từ tệp Room\_Details.csv

vector<string> rows = dp.getCSVData(ROOM\_DATA); // rows = ["1,Single,1000,0,Available", "2,Double,2000,0,Available", ...]

do {

isValidChoice = true;

cout << "\nSelect preferred Room Type: \n1 - Single\n2 - Double\n3 - Triple\n4 - Quadruple\nYour Choice: ";

cin >> choice;

switch (choice) {

case 1:

room\_type = "Single";

break;

case 2:

room\_type = "Double";

break;

case 3:

room\_type = "Triple";

break;

case 4:

room\_type = "Quadruple";

break;

default:

dis.changeConsoleColor(LIGHT\_RED);

cout << "\nInvalid Option\n";

isValidChoice = false;

dis.changeConsoleColor(WHITE);

break;

}

bool areRoomsFound = false;

cout << "\nSearching for Available Rooms...\n";

dis.loadingBarAnimation();

for (int i = 0; i < rows.size(); i++) {

vector<string> data = dp.split(rows[i], ',');

//Nếu phòng có loại phòng được chọn và trạng thái "Available"

if (data[1] == room\_type && data[4] == "Available") {

areRoomsFound = true;

Room room(data);

room.displayRoomData();

}

}

if(!areRoomsFound) { // Nếu không có phòng nào trống

dis.changeConsoleColor(LIGHT\_RED);

cout << "\nNo available rooms of this type. Check again later\n";

dis.changeConsoleColor(WHITE);

isValidChoice = false;

}

} while(!isValidChoice);

return room\_type;

}

Phương thức ‘removeRoom’ trong lớp ‘Admin’ cho phép xóa một phòng khỏi cơ sở dữ liệu.

void Admin::removeRoom() {

string room\_id;

cout << "\nEnter room ID: ";

cin >> room\_id;

// Lấy dữ liệu từ tệp Room\_Details.csv

vector<string> rows = dp.getCSVData(ROOM\_DATA); // rows = ["1,Single,1000,0,Available", "2,Double,2000,0,Available", ...]

bool isRoomFound = false, is\_occupied = false;

// Duyệt qua tất cả các phòng

for(int i = 0; i < rows.size(); i++) {

// Tách dữ liệu từng dòng

vector<string> data = dp.split(rows[i], ','); // data = {id, type, price, occupant\_count, status}

if(data[0] == room\_id) {

if(data[4] != "Available") { // Kiểm tra phòng đang được sử dụng hay không

is\_occupied = true;

break;

}

else {

isRoomFound = true;

dp.deleteRow(rows, room\_id); // Xóa phòng khỏi danh sách

break;

}

}

}

if (is\_occupied) { // Nếu phòng đang được sử dụng

dis.changeConsoleColor(LIGHT\_RED);

cout << "\nRoom is currently occupied. Cannot be removed\n";

dis.changeConsoleColor(WHITE);

}

else if(!isRoomFound) { // Nếu phòng không tồn tại

dis.changeConsoleColor(LIGHT\_RED);

cout << "\nERROR!!\nRoom not found. Kindly make sure you entered the right ID\n";

dis.changeConsoleColor(WHITE);

}

else { // Nếu phòng được tìm thấy và xóa thành công

// Ghi lại dữ liệu sau khi xóa

ofstream outStream(ROOM\_DATA);

for(int i = 0; i < rows.size(); i++)

outStream << rows[i] << endl;

outStream.close();

dis.changeConsoleColor(LIGHT\_GREEN);

cout << "\nRoom removed successfully";

dis.changeConsoleColor(WHITE);

}

}

**Phương thức quản trị khách hàng**

Phương thức ‘showAllGuests’ trong lớp ‘Admin’ được sử dụng để lấy và hiển thị thông tin về tất cả khách hàng. Lấy dữ liệu từ tệp Guest\_Data.csv

void Admin::showAllGuests() {

// Lấy dữ liệu từ tệp Guest\_Data.csv

vector<string> rows = dp.getCSVData(GUEST\_DATA);

// rows = ["Y678,HC Hieu,12345,101,10-04-2024", "Y679,NTV Hiep,67890,102,10-04-2024", ...]

cout << "Retrieving Guest Data...\n\n";

dis.loadingBarAnimation();

for (int i = 0; i < rows.size(); i++) { // Duyệt qua tất cả các dòng trong tệp

vector<string> data = dp.split(rows[i], ','); // data = {id, name, contact\_info, room\_id, arrival\_date}

Guest guest(data); // Tạo một đối tượng khách từ data

guest.getDetails(); // Hiển thị thông tin khách

}

}

Phương thức ‘searchForGuest’ trong lớp ‘Admin’ cho phép người dùng tìm kiếm một khách hàng bằng tên và ID của họ.

void Admin::searchForGuest(string name, string ID) {

// Lấy dữ liệu từ tệp Guest\_Data.csv

vector<string> rows = dp.getCSVData(GUEST\_DATA);

// rows = ["Y678,HC Hieu,12345,101,10-04-2024", "Y679,NTV Hiep,67890,102,10-04-2024", ...]

bool isGuestFound = false;

for (int i = 0; i < rows.size(); i++) {

vector<string> data = dp.split(rows[i], ','); // data = {id, name, contact\_info, room\_id, arrival\_date}

if (data[0] == ID && data[1] == name) { // So sánh id và tên khách

isGuestFound = true;

dis.loadingBarAnimation();

Guest guest(data); // Tạo một đối tượng khách từ data

guest.getDetails(); // Hiển thị thông tin khách

break;

}

}

if (!isGuestFound) { // Nếu không tìm thấy khách

dis.changeConsoleColor(LIGHT\_RED);

cout << "\nGuest not found. Please try again\n";

dis.changeConsoleColor(WHITE);

}

}

Nạp chồng phương thức ‘searchForGuest()’ để tìm kiếm khách hàng bằng cách nhập tên và ID.

void Admin::searchForGuest() {

string name,f\_name, l\_name, id;

cout << "\nEnter Guest's first name: ";

cin >> f\_name;

cout << "Enter Guest's last name: ";

cin>> l\_name;

name = l\_name + " " + f\_name;

cout << "Enter Guest ID: ";

cin >> id;

cout << "\nSearching for Guest...";

// Lấy dữ liệu từ tệp Guest\_Data.csv

vector<string> rows = dp.getCSVData(GUEST\_DATA);

// rows = ["Y678,HC Hieu,12345,101,10-04-2024", "Y679,NTV Hiep,67890,102,10-04-2024", ...]

bool isGuestFound = false;

for (int i = 0; i < rows.size(); i++) { // Duyệt qua tất cả các dòng trong tệp

vector<string> data = dp.split(rows[i], ','); // data = {id, name, contact\_info, room\_id, arrival\_date}

if (data[0] == id && data[1] == name) { // So sánh id và tên khách

isGuestFound = true;

dis.loadingBarAnimation();

Guest guest(data); // Tạo một đối tượng khách từ data

guest.getDetails(); // Hiển thị thông tin khách

break;

}

}

if (!isGuestFound) { // Nếu không tìm thấy khách

dis.changeConsoleColor(LIGHT\_RED);

cout << "\nGuest not found. Please try again\n";

dis.changeConsoleColor(WHITE);

}

}

Phương thức ‘checkIn’ trong lớp ‘Admin’ cho phép thêm một khách hàng mới vào cơ sở dữ liệu.

void Admin::checkIn() {

string f\_name, l\_name, name, id, contact\_info, arrival\_date, room\_id, occupant\_no;

vector<string> guest\_info;

// Lấy dữ liệu từ tệp Room\_Details.csv

vector <string> room\_data = dp.getCSVData(ROOM\_DATA);

// room\_data = ["1,Single,1000,0,Available", "2,Double,2000,0,Available", ...]

bool success;

do {

int rm\_id;

cout << "\nFirst name: ";

cin >> f\_name;

fflush(stdin);

cout << "\nLast name: ";

cin >> l\_name;

fflush(stdin);

name = l\_name + " " + f\_name;

cout << "Enter Guest contact info: ";

cin >> contact\_info;

fflush(stdin);

cout << "Enter Guest ID: ";

fflush(stdin);

cin >> id;

string typeRoom = getRoomByType(); // Lấy loại phòng từ phương thức getRoomByType()

cout << "Enter preferred Room ID: ";

cin >> room\_id;

fflush(stdin);

cout << "Enter occupant number: ";

cin >> occupant\_no;

arrival\_date = dp.getCurrentDate(); // Lấy ngày hiện tại

// Tạo một đối tượng khách mới từ thông tin khách vừa nhập

guest\_info.push\_back(id);

guest\_info.push\_back(name);

guest\_info.push\_back(contact\_info);

guest\_info.push\_back(room\_id);

guest\_info.push\_back(arrival\_date);

Guest new\_guest(guest\_info); // Tạo một đối tượng khách từ guest\_info

string new\_guest\_data = id + "," + name + "," + contact\_info + "," + room\_id + "," + arrival\_date;

bool isRoomFound = false ,isRoomFull = false;

for (int i = 0; i < room\_data.size(); i++) { // Duyệt qua tất cả các phòng

vector<string> data = dp.split(room\_data[i], ','); // data = {id, type, price, occupant\_count, status}

if(data[0] == room\_id && data[1] == typeRoom) { // So sánh id và loại phòng

isRoomFound = true;

if (data[4] == "Unavailable") { // Kiểm tra phòng đã có người ở chưa

isRoomFull = true;

}

break;

}

}

if(!isRoomFound) { // Nếu phòng không tồn tại

dis.changeConsoleColor(LIGHT\_RED);

cout << "ERROR!!\nRoom not found. Kindly make sure you entered the right ID\n";

success = false;

dis.changeConsoleColor(WHITE);

}

else if(isRoomFull) { // Nếu phòng đã có người ở

dis.changeConsoleColor(YELLOW);

cout << "\nRoom is currently Full. Please try another room\n";

success = false;

dis.changeConsoleColor(WHITE);

}

else { // Nếu phòng tồn tại và chưa có người ở

// Update Guest database

// Lấy dữ liệu từ tệp Guest\_Data.csv

vector<string> guest\_details = dp.getCSVData(GUEST\_DATA);

// guest\_details = ["Y678,HC Hieu,12345,101,10-04-2024", "Y679,NTV Hiep,67890,102,10-04-2024", ...]

guest\_details.push\_back(new\_guest\_data); // Thêm khách mới vào danh sách khách

ofstream outStream(GUEST\_DATA);

// Thêm tất cả khách vào tệp Guest\_Data.csv

for (int i = 0; i < guest\_details.size(); i++) {

outStream << guest\_details[i] << endl;

}

outStream.close();

// Update Room database

room\_data = dp.getCSVData(ROOM\_DATA);

string updatedRoomInfo;

for(int i = 0; i < room\_data.size(); i++) { // Duyệt qua tất cả các phòng

vector<string> data = dp.split(room\_data[i], ','); // data = {id, type, price, occupant\_count, status}

if(data[0] == room\_id) { // So sánh id phòng

// Cập nhật thông tin phòng sau khi có khách

updatedRoomInfo = data[0] + "," + data[1] + "," + data[2] + "," + occupant\_no + ",Unavailable";

dp.deleteRow(room\_data,room\_id); // Xóa phòng cũ

room\_data.push\_back(updatedRoomInfo); // Thêm phòng mới

break;

}

}

//Update Room database

ofstream room\_file(ROOM\_DATA);

for(int i = 0; i < room\_data.size(); i++) // Ghi lại dữ liệu sau khi cập nhật

room\_file << room\_data[i] << endl; // Ghi dữ liệu vào tệp Room\_Details.csv

room\_file.close();

cout << "\nRegistering New Guest...\n";

dis.loadingBarAnimation();

new\_guest.getDetails(); // Hiển thị thông tin khách mới

dis.changeConsoleColor(LIGHT\_GREEN);

cout << "\nGuest successfully Added\n";

success = true;

dis.changeConsoleColor(WHITE);

}

} while (!success);

}

Phương thức ‘checkOut’ trong lớp ‘Admin’ cho phép xóa một khách hàng khỏi cơ sở dữ liệu.

void Admin::checkOut() {

string guest\_name, f\_name, l\_name, guest\_id;

cout << "Enter Guest's First name: ";

cin >> f\_name;

cout << "Enter Guest's Last name: ";

cin >> l\_name;

guest\_name = l\_name + " " + f\_name;

cout << "Enter Guest ID: ";

cin >> guest\_id;

// Lấy dữ liệu từ tệp Guest\_Data.csv

vector<string> guest\_data = dp.getCSVData(GUEST\_DATA);

// guest\_data = ["Y678,HC Hieu,12345,101,10-04-2024", "Y679,NTV Hiep,67890,102,10-04-2024", ...]

string roomID;

bool isGuestFound = false;

dis.loadingBarAnimation();

for(int i = 0; i < guest\_data.size(); i++) { // Duyệt qua tất cả các dòng trong tệp

vector<string> data = dp.split(guest\_data[i], ','); // data = {id, name, contact\_info, room\_id, arrival\_date}

if(data[0] == guest\_id && data[1] == guest\_name) { // So sánh id và tên khách

Guest guest(data); // Tạo một đối tượng khách từ data

isGuestFound = true;

roomID = data[3];

dis.changeConsoleColor(LIGHT\_GREEN);

cout << "Guest Identified" << endl;

dis.changeConsoleColor(WHITE);

guest.getDetails(); // Hiển thị thông tin khách trước khi xóa

// Tính tiền phòng

vector<string> room\_data = dp.getCSVData(ROOM\_DATA);

string room\_price;

for(int j = 0; j < room\_data.size(); j++) {

vector<string> room\_info = dp.split(room\_data[j], ',');

if(room\_info[0] == roomID) {

room\_price = room\_info[2];

break;

}

}

// Tính số ngày đã ở

int num\_of\_days = 0;

string current\_date = dp.getCurrentDate();

vector<string> guest\_info = dp.split(guest\_data[i], ',');

// guest\_info = {id, name, contact\_info, room\_id, arrival\_date}

string arrival\_date = guest\_info[4];

// Tách ngày, tháng, năm từ chuỗi ngày đến

int arrival\_day = stoi(dp.split(arrival\_date, '-')[0]);

int arrival\_month = stoi(dp.split(arrival\_date, '-')[1]);

// Tách ngày, tháng, năm từ chuỗi ngày hiện tại

int current\_day = stoi(dp.split(current\_date, '-')[0]);

int current\_month = stoi(dp.split(current\_date, '-')[1]);

// Tính số ngày đã ở

num\_of\_days = (current\_month - arrival\_month) \* 30

+ (current\_day - arrival\_day);

cout << "Number of days stayed: " << num\_of\_days << endl;

// Tính tổng tiền

int total\_price = stoi(room\_price) \* num\_of\_days;

dis.changeConsoleColor(LIGHT\_GREEN);

cout << "Total Price: " << total\_price << " VND" << endl;

dis.changeConsoleColor(WHITE);

break;

}

}

if(!isGuestFound){ // Nếu không tìm thấy khách

dis.changeConsoleColor(LIGHT\_RED);

cout << "\nERROR!!\nGuest not found. Kindly make sure you entered the right details\n";

dis.changeConsoleColor(WHITE);

}

else { // Nếu tìm thấy khách

cout << "\nPress any key to check out\n" << endl;

int pause = getch();

dis.loadingBarAnimation();

dp.deleteRow(guest\_data, guest\_id); // Xóa khách khỏi danh sách

string updatedRoomInfo; // Thông tin phòng sau khi cập nhật

ofstream guest\_data\_update(GUEST\_DATA); // Mở tệp Guest\_Data.csv để ghi lại dữ liệu

for(int i = 0; i < guest\_data.size(); i++) // Ghi lại dữ liệu sau khi xóa

{

guest\_data\_update << guest\_data[i] << endl;

}

// Lấy dữ liệu từ tệp Room\_Details.csv

vector<string> room\_data = dp.getCSVData(ROOM\_DATA);

// room\_data = ["1,Single,1000,0,Available", "2,Double,2000,0,Available", ...]

for(int i = 0; i < room\_data.size(); i++) { // Duyệt qua tất cả các phòng

vector<string> data = dp.split(room\_data[i], ','); // data = {id, type, price, occupant\_count, status}

if(data[0] == roomID) { // So sánh id phòng

// Cập nhật thông tin phòng

updatedRoomInfo = data[0] + "," + data[1] + "," + data[2] + "," + "0,Available";

dp.deleteRow(room\_data,roomID); // Xóa phòng cũ

room\_data.push\_back(updatedRoomInfo); // Thêm phòng mới

break;

}

}

//Update Room database

ofstream room\_file(ROOM\_DATA);

for(int i = 0; i < room\_data.size(); i++) // Ghi lại dữ liệu sau khi cập nhật

{

room\_file << room\_data[i] << endl; // Ghi dữ liệu vào tệp Room\_Details.csv

}

room\_file.close();

dis.changeConsoleColor(LIGHT\_GREEN);

cout << "\nGuest removed successfully";

dis.changeConsoleColor(WHITE);

}

}

Phương thức ‘removeGuest’ trong lớp ‘Admin’ cho phép xóa một khách hàng khỏi cơ sở dữ liệu từ tên và ID.

void Admin::removeGuest(string name, string ID) {

// Lấy dữ liệu từ tệp Guest\_Data.csv

vector<string> guest\_data = dp.getCSVData(GUEST\_DATA);

// guest\_data = ["Y678,HC Hieu,12345,101,10-04-2024", "Y679,NTV Hiep,67890,102,10-04-2024", ...]

string roomID;

bool isGuestFound = false;

dis.loadingBarAnimation();

for(int i = 0; i < guest\_data.size(); i++) { // Duyệt qua tất cả các dòng trong tệp

vector<string> data = dp.split(guest\_data[i], ',');

// data = {id, name, contact\_info, room\_id, arrival\_date}

if(data[0] == ID && data[1] == name) { // So sánh id và tên khách

Guest guest(data); // Tạo một đối tượng khách từ data

isGuestFound = true;

roomID = data[3];

dis.changeConsoleColor(LIGHT\_GREEN);

cout << "Guest Identified" << endl;

dis.changeConsoleColor(WHITE);

guest.getDetails(); // Hiển thị thông tin khách trước khi xóa

break;

}

}

if(!isGuestFound){ // Nếu không tìm thấy khách

dis.changeConsoleColor(LIGHT\_RED);

cout << "\nERROR!!\nGuest not found. Kindly make sure you entered the right ID\n";

dis.changeConsoleColor(WHITE);

}

else { // Nếu tìm thấy khách

cout << "\nPress any key to delete Guest\n" << endl;

int pause = getch();

dis.loadingBarAnimation();

// Xóa khách khỏi danh sách

dp.deleteRow(guest\_data, ID);

string updatedRoomInfo;

// Ghi lại dữ liệu sau khi xóa

ofstream guest\_data\_update(GUEST\_DATA);

for(int i = 0; i < guest\_data.size(); i++) // Ghi lại dữ liệu sau khi xóa

{

guest\_data\_update << guest\_data[i] << endl; // Ghi dữ liệu vào tệp Guest\_Data.csv

}

// Lấy dữ liệu từ tệp Room\_Details.csv

vector<string> room\_data = dp.getCSVData(ROOM\_DATA);

for(int i = 0; i < room\_data.size(); i++) { // Duyệt qua tất cả các phòng

vector<string> data = dp.split(room\_data[i], ','); // data = {id, type, price, occupant\_count, status}

if(data[0] == roomID) { // So sánh id phòng

updatedRoomInfo = data[0] + "," + data[1] + "," + data[2] + "," + "0,Available";

dp.deleteRow(room\_data,roomID); // Xóa phòng cũ

room\_data.push\_back(updatedRoomInfo); // Thêm phòng mới

break;

}

}

//Update Room database

ofstream room\_file(ROOM\_DATA);

for(int i = 0; i < room\_data.size(); i++) // Ghi lại dữ liệu sau khi cập nhật

{

room\_file << room\_data[i] << endl;

}

room\_file.close();

dis.changeConsoleColor(LIGHT\_GREEN);

cout << "\nGuest removed successfully";

dis.changeConsoleColor(WHITE);

}

}

# THỰC HIỆN CHƯƠNG TRÌNH

int main() {

Display display;

bool isProgramRunning, adminMenu, guestMenu;

display.intro();

do {

cout << "\n\t\tHOTEL MANAGEMENT SYSTEM"

<< "\n---------------------------------"

<< "\nADMIN LOGIN - Press a"

<< "\nGUEST LOGIN - Press g"

<< "\nEXIT PROGRAM - Press e"

<< "\n>> ";

char option; cin >> option;

Account \*acc;

Guest guest1;

Admin admin1;

if (option == a) {

acc = &admin1;

bool login\_success = acc->login();

if(login\_success) {

do {

adminMenu = false;

char choice;

cout << "\nPlease enter an option"

<< "\n---------------------------------"

<< "\n1. Show all rooms"

<< "\n2. Search for a particular room"

<< "\n3. Search for available rooms"

<< "\n4. Add room"

<< "\n5. Remove a room"

<< "\n6. Show all guests"

<< "\n7. Search for Guest"

<< "\n8. Book a room"

<< "\n9. Return the room"

<< "\nb. Back to main menu\n"

<< "\ne. Exit\n";

cout << ">> ";

cin >> choice;

switch (choice) {

case '1':

admin1.showAllRooms();

cout << "\nPerform another task? (y/n[Back to main menu])\n>> ";

choice = getch();

if(choice == 'n') isProgramRunning = true;

else if (choice == 'y') adminMenu = true;

break;

case '2':

admin1.searchForRoom();

cout << "\nPerform another task? (y/n[Back to main menu])\n>> ";

choice = getch();

if(choice == 'n') isProgramRunning = true;

else if (choice == 'y') adminMenu = true;

break;

case '3':

admin1.searchAvailableRooms();

cout << "\nPerform another task? (y/n[Back to main menu])\n>> ";

choice = getch();

if(choice == 'n') isProgramRunning = true;

else if (choice == 'y') adminMenu = true;

break;

case '4':

admin1.addRoom();

cout << "\nPerform another task? (y/n[Back to main menu])\n>> ";

choice = getch();

if(choice == 'n') isProgramRunning = true;

else if (choice == 'y') adminMenu = true;

break;

case '5':

admin1.removeRoom();

cout << "\nPerform another task? (y/n[Back to main menu])\n>> ";

choice = getch();

if(choice == 'n') isProgramRunning = true;

else if (choice == 'y') adminMenu = true;

break;

case '6':

admin1.showAllGuests();

cout << "\nPerform another task? (y/n[Back to main menu])\n>> ";

choice = getch();

if(choice == 'n') isProgramRunning = true;

else if (choice == 'y') adminMenu = true;

break;

case '7':

admin1.searchForGuest();

cout << "\nPerform another task? (y/n[Back to main menu])\n>> ";

choice = getch();

if(choice == 'n') isProgramRunning = true;

else if (choice == 'y') adminMenu = true;

break;

case '8':

admin1.checkIn();

cout << "\nPerform another task? (y/n[Back to main menu])\n>> ";

choice = getch();

if(choice == 'n') isProgramRunning = true;

else if (choice == 'y') adminMenu = true;

break;

case '9':

admin1.checkOut();

cout << "\nPerform another task? (y/n[Back to main menu])\n>> ";

choice = getch();

if(choice == 'n') isProgramRunning = true;

else if (choice == 'y') adminMenu = true;

break;

case 'b':

isProgramRunning = true;

break;

case 'e':

isProgramRunning = false;

break;

default:

display.changeConsoleColor(LIGHT\_RED);

cout<< "\nYou entered an invalid option, please try again."<< endl;

adminMenu = true;

display.changeConsoleColor(WHITE);

break;

}

} while(adminMenu);

}

}

else if (option == g) {

acc = &guest1;

bool login\_success = acc->login();

if(login\_success) {

char choice;

do {

guestMenu = false;

cout << "\nPlease enter an option"

<< "\n----------------------------------"

<< "\n1. Search for available rooms"

<< "\n2. Check for personal details"

<< "\n3. Leave room"

<< "\n4. Back to Main Menu\n"

<< "\n5. Exit\n\n";

string name, f\_name, l\_name, ID;

bool login\_success;

cout << ">> ";

cin >> choice;

switch (choice)

{

case '1':

cout << "These are the rooms available:"

<< "\n---------------------------------------\n";

admin1.searchAvailableRooms();

cout << "\nPerform another action?(y/n [Back to main menu])\n >> ";

choice = getch();

if(choice == 'n') isProgramRunning = true;

else if (choice == 'y') guestMenu = true;

break;

case '2':

cout << "Enter your first name: ";

cin >> f\_name;

cout << "Enter your last name: ";

cin >> l\_name;

name = l\_name + " " + f\_name;

cout << "Enter you ID: ";

cin >> ID;

login\_success = guest1.guestLogin(name, ID);

if(login\_success){

admin1.searchForGuest(name, ID);

cout << "\nPerform another action?(y/n [Back to main menu])\n >> ";

choice = getch();

if(choice == 'n') isProgramRunning = true;

else if (choice == 'y') guestMenu = true;

}

else guestMenu = true;

break;

case '3':

cout << "Enter your first name: ";

cin >> f\_name;

cout << "Enter your last name: ";

cin >> l\_name;

name = l\_name + " " + f\_name;

cout << "Enter you ID: ";

cin >> ID;

login\_success = guest1.guestLogin(name, ID);

if(login\_success){

admin1.removeGuest(name, ID);

cout << "\nPerform another action?(y/n [Back to main menu])\n >> ";

choice = getch();

if(choice == 'n') isProgramRunning = true;

else if (choice == 'y') guestMenu = true;

} else guestMenu = true;

break;

case '4':

isProgramRunning = true;

break;

case '5':

isProgramRunning = false;

break;

case 'y':

guestMenu = true;

break;

case 'n' :

isProgramRunning = true;

break;

default:

display.changeConsoleColor(LIGHT\_RED);

cout<< "\nYou entered an invalid option, please try again.\n"<< endl;

guestMenu = true;

display.changeConsoleColor(WHITE);

break;

}

} while(guestMenu);

}

}

else if (option == e) {

isProgramRunning = false;

}

else {

display.changeConsoleColor(LIGHT\_RED);

cout<< "\nYou entered an invalid option, please try again.\n"<< endl;

//char temp = getch();

display.changeConsoleColor(WHITE);

isProgramRunning = true;

}

} while(isProgramRunning);

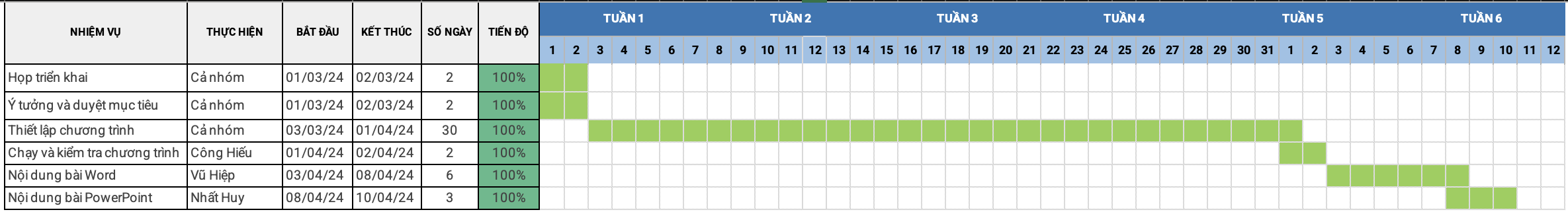
display.outro();

return 0;

}

# BẢNG KẾ HOẠCH PHÂN CÔNG NHIỆM VỤ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nhiệm vụ** | **Thực hiện** | **Bắt đầu** | **Kết thúc** | **Số ngày** | **Tiến độ** |
| **Họp triển khai** | **Cả nhóm** | **01/03/2024** | **02/03/2024** | **2** | **Hoàn thành** |
| **Ý tưởng và duyệt mục tiêu** | **Cả nhóm** | **01/03/2024** | **02/03/2024** | **2** | **Hoàn thành** |
| **Thiết lập chương trình** | **Cả nhóm** | **03/03/2024** | **01/04/2024** | **30** | **Hoàn thành** |
| **Chạy và kiểm tra chương trình** | **Công Hiếu** | **01/04/2024** | **02/04/2024** | **2** | **Hoàn thành** |
| **Nội dung bài Word** | **Vũ Hiệp** | **03/04/2024** | **08/04/2024** | **6** | **Hoàn thành** |
| **Nội dung bài PowerPoint** | **Nhất Huy** | **08/04/2024** | **10/04/2024** | **3** | **Hoàn thành** |

****

*Biểu đồ Gantt*

# TÀI LIỆU THAM KHẢO

1. *QUY TRÌNH LÀM VIỆC CỦA NHÂN VIÊN ĐẶT PHÒNG KHÁCH SẠN, hoteljob.vn.* Available at: https://www.hoteljob.vn/tin-tuc/quy-trinh-lam-viec-cua-nhan-vien-dat-phong-khach-san