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

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".**

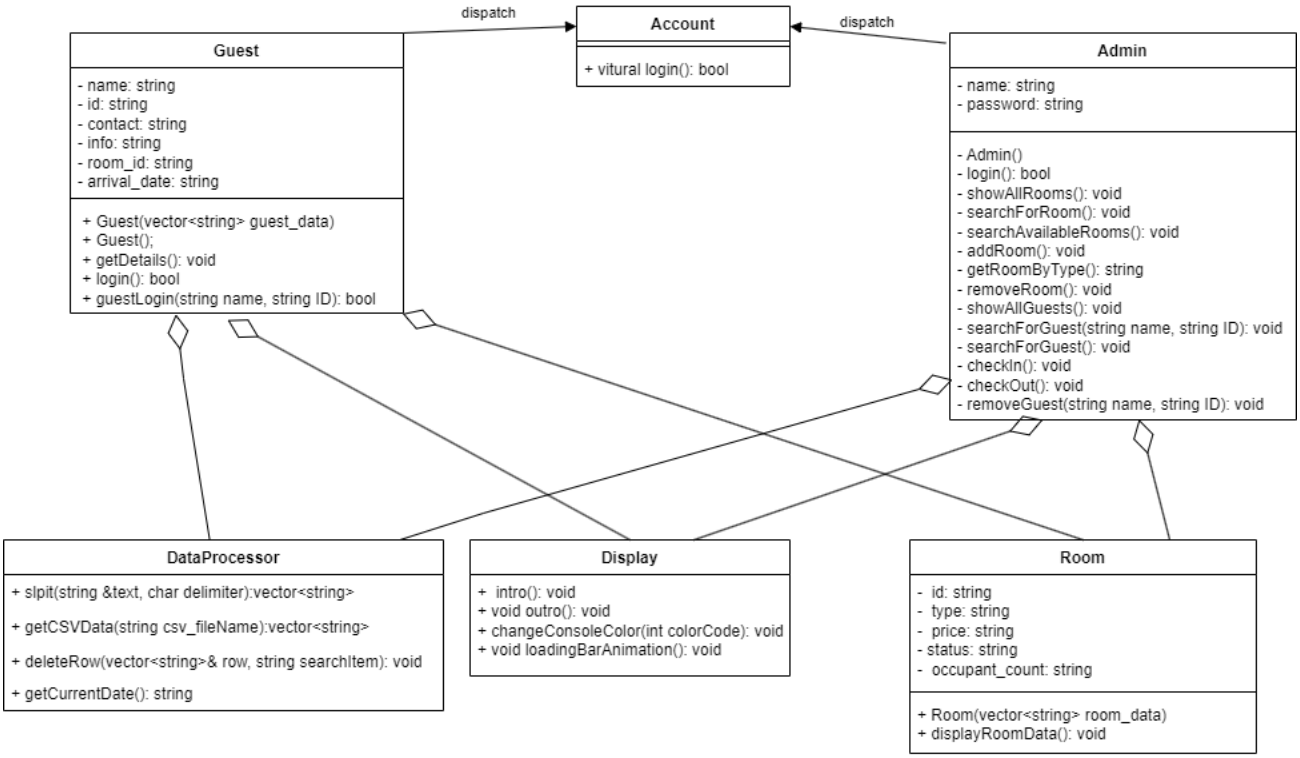
Chương trình quản lý khách sạn bằng C++ là một công cụ quan trọng để tự động hóa và tối ưu hóa các quy trình quản lý trong ngành khách sạn. Việc sử dụng ngôn ngữ lập trình C++ giúp chúng ta xây dựng một hệ thống linh hoạt, hiệu quả và dễ bảo trì. Tích hợp các tính năng như đặt phòng, quản lý thông tin 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

I. Sơ đồ phân lớp



II. 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.

2. 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.

3. 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.

4. Lớp Account

```
class Account{
public:
    virtual bool login() = 0;
};
```

Công dụng: Cung cấp chức năng đăng nhập ảo.

5. 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.

6. 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.

III. 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

I. 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);
}
```

II. 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.

```
void Display::intro() {
    changeConsoleColor(YELLOW);
    // ASCII Art generated from https://patorjk.com/software/taag/#p=display&f=Doh&t=Welcome

    cout << R"(

WWWWWWWW                           WWWWWWWW             lllllll
W::::::W                           W::::::W             l:::::l
W::::::W                           W::::::W             l:::::l
W::::::W                           W::::::W             l:::::l
 W:::::W           WWWWW           W:::::W eeeeeeeeeee    l::::l     ccccccccccccccc   ooooooooooo      mmmmmmm    mmmmmmm       eeeeeeeeeee
  W:::::W         W:::::W         W:::::Wee::::::::::::ee  l::::l   cc:::::::::::::::c oo:::::::::::oo  mm:::::::m  m:::::::mm   ee::::::::::::ee
   W:::::W       W:::::::W       W:::::We::::::eeeee:::::eel::::l  c:::::::::::::::::co:::::::::::::::om::::::::::mm::::::::::m e::::::eeeee:::::ee
    W:::::W     W:::::::::W     W:::::We::::::e     e:::::el::::l c:::::::ccccccc:::::co:::::ooooo:::::om:::::::::::::::::::::::me::::::e     e:::::e
     W:::::W   W:::::W:::::W   W:::::W e:::::::eeeee:::::el::::l c::::::c     ccccccco::::o     o::::om::::mmm::::::mmm::::me:::::::eeeee:::::e
      W:::::W W:::::W W:::::W W:::::W  e:::::::::::::::::e l::::l c:::::c             o::::o     o::::om::::m   m::::m   m::::me::::::::::::::::e
       W:::::W:::::W   W:::::W:::::W   e::::::eeeeeeeeeee  l::::l c:::::c             o::::o     o::::om::::m   m::::m   m::::me::::::eeeeeeeeee
        W:::::::::W     W:::::::::W    e:::::::e           l::::l c::::::c     ccccccco::::o     o::::om::::m   m::::m   m::::me:::::::e
         W:::::::W       W:::::::W     e::::::::e         l::::::lc:::::::cccccc:::::co:::::ooooo:::::om::::m   m::::m   m::::me::::::::e
          W:::::W         W:::::W       e::::::::eeeeeeee l::::::l c:::::::::::::::::co:::::::::::::::om::::m   m::::m   m::::m e::::::::eeeeeeee
           W:::W           W:::W         ee:::::::::::::e l::::::l  cc:::::::::::::::c oo:::::::::::oo m::::m   m::::m   m::::m  ee::::::::::::e
            WWW             WWW            eeeeeeeeeeeeee lllllllll    cccccccccccccccc   ooooooooooo   mmmmmm   mmmmmm   mmmmmm    eeeeeeeeeeeeee
    )" << endl;

    cout << endl;
    cout << "\n\n\t\tBY: Ho Cong Hieu - 21200289";
    cout << "\t\tBY: Nguyen Thua Vu Hiep - 21200288";
    cout << "\t\tBY: Nguyen Nhat Huy - 21200294\n";
    changeConsoleColor(WHITE);
}
```

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

```
void Display::outro() {
    changeConsoleColor(YELLOW);
    // ASCII Art generated from https://patorjk.com/software/taag/#p=display&f=Doh&t=Goodbye
    cout << R"(

                                                                   ddddddddbbbbbbbb
        GGGGGGGGGGGGG                                              d::::::db::::::b
     GGG::::::::::::G                                              d::::::db::::::b
   GG:::::::::::::::G                                              d::::::db::::::b
  G:::::GGGGGGGG::::G                                              d:::::d  b:::::b
 G:::::G       GGGGGG   ooooooooooo      ooooooooooo       ddddddddd:::::d  b:::::bbbbbbbbb yyyyyyy           yyyyyyy eeeeeeeeeeee
G:::::G               oo:::::::::::oo  oo:::::::::::oo   dd::::::::::::::d  b::::::::::::::bby:::::y         y:::::yee::::::::::::ee
G:::::G              o:::::::::::::::oo:::::::::::::::o d::::::::::::::::d  b::::::::::::::::by:::::y       y:::::ye::::::eeeee:::::ee
G:::::G    GGGGGGGGGGo:::::ooooo:::::oo:::::ooooo:::::od:::::::ddddd:::::d  b:::::bbbbb:::::::by:::::y     y:::::ye::::::e     e:::::e
G:::::G    G::::::::Go::::o     o::::oo::::o     o::::od::::::d    d:::::d  b:::::b    b::::::b y:::::y   y:::::y e:::::::eeeee::::::e
G:::::G    GGGGG::::Go::::o     o::::oo::::o     o::::od:::::d     d:::::d  b:::::b     b:::::b  y:::::y y:::::y  e:::::::::::::::::e
G:::::G        G::::Go::::o     o::::oo::::o     o::::od:::::d     d:::::d  b:::::b     b:::::b   y:::::y:::::y   e::::::eeeeeeeeeee
 G:::::G       G::::Go::::o     o::::oo::::o     o::::od:::::d     d:::::d  b:::::b     b:::::b    y:::::::::y    e:::::::e
  G:::::GGGGGGGG::::Go:::::ooooo:::::oo:::::ooooo:::::od::::::ddddd::::::dd b:::::bbbbbb::::::b     y:::::::y     e::::::::e
   GG:::::::::::::::Go:::::::::::::::oo:::::::::::::::o d:::::::::::::::::d b::::::::::::::::b       y:::::y       e::::::::eeeeeeee
     GGG::::::GGG:::G oo:::::::::::oo  oo:::::::::::oo   d:::::::::ddd::::d b:::::::::::::::b       y:::::y         ee:::::::::::::e
        GGGGGG   GGGG   ooooooooooo      ooooooooooo      ddddddddd   ddddd bbbbbbbbbbbbbbbb       y:::::y            eeeeeeeeeeeeee
                                                                                                  y:::::y
                                                                                                 y:::::y
                                                                                                y:::::y
                                                                                               y:::::y
                                                                                              yyyyyyy

)" << endl;
    changeConsoleColor(WHITE);
}
```

III. 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);
}
```

IV. 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;
}
```

V. 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;
}
```

1. 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);

    }
}
```

2. 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':
```

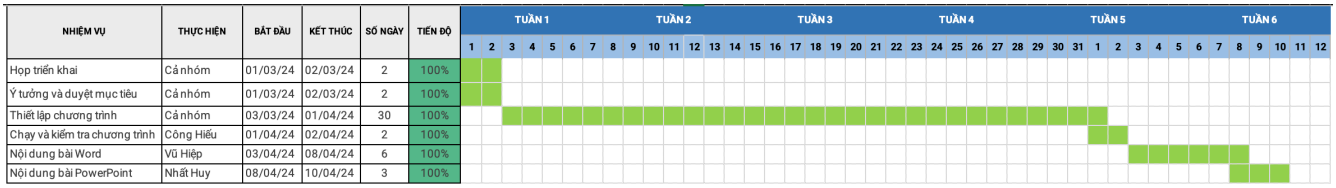
```
                admin1.searchForGuest();
                cout << "\nPerform another task? (y/n[Back to main menu])\n>> ";
                choice = getch();
                if(choice == 'n') isProgramRunning = true;
                else if (choice == 'y') adminMenu = true;
                break;

            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