ICS1112 – PROBLEM SOLVING AND PROGRAMMING IN C LAB

APPLICATION DEVELOPMENT -MINI PROJECT

TITLE: HOTEL MANAGEMENT SYSTEM

TEAM DETAILS:

|  |  |  |
| --- | --- | --- |
| NAME | DIGITAL ID | REGISTER NUMBER: |
| VIJAYARAAGHAVAN K S | 2470005 | 3122247001066 |
| GOKUL N V | 2470019 | 3122247001017 |
| RAJGURU SUNDARESWARAN | 2470064 | 3122247001048 |

**Problem Statement**

Developing a **Hotel Management System** to streamline hotel operations and improve customer satisfaction. The system should provide the following functionalities:

1. **Customer Interface:**
   * Enable customers to:
     + Book rooms by selecting room types and specifying the duration of stay.
     + Order food for breakfast, lunch, and dinner with multiple options for each meal.
     + Access premium hotel services such as spa, gym, swimming pool, massage, and guided tours.
   * Calculate and display the total expenses for the customer, including charges for rooms, food, and services.
2. **Owner Interface:**
   * Allow hotel owners to:
     + View total earnings from all customers.
     + Retrieve details of all customers, including their name, room, food, and service costs.
   * Protect owner operations using password-based authentication to ensure secure access.
3. **Billing System:**
   * Generate a detailed bill for any customer, including:
     + A breakdown of room charges, food costs, and service fees.
     + Tax calculations (e.g., CGST and SGST).
     + A final total amount including all applicable taxes.
4. **Dynamic Data Management:**
   * Use dynamic memory allocation to:
     + Store customer details such as name, stay duration, and expenses.
     + Scale the system to handle multiple customers simultaneously.
   * Ensure efficient memory usage to avoid leaks or corruption.
5. **User-Friendly Interface:**
   * Provide a menu-driven interface that is intuitive and easy to navigate.
   * Separate functionalities clearly for customers and owners.
   * Display detailed prompts and options to guide users through various operations.

### Algorithm for the Main Program

1. **Start** the program.
2. **Initialize** global variables: customers (pointer to dynamically allocated customer array), customerCount (0), and totalEarnings (0).
3. Display a welcome message: "Welcome to Our GOAT Hotel Management System!"
4. Repeat until the user chooses to exit:
   1. Display the main menu:
      * 1. Customer Interface
      * 2. Owner Interface
      * 3. Exit
   2. Accept the user's choice.
   3. Based on the user's choice:
      * **Case 1:** Call the customerInterface() function.
      * **Case 2:** Call the ownerInterface() function.
      * **Case 3:** Exit the loop.
      * **Default:** Display an "Invalid choice" message.
5. **Free allocated memory** for all customers in the customers array.
6. **End** the program.

**Algorithm for customerInterface() Function**

1. **Start** by dynamically allocating memory for a new Customer object.
   * If memory allocation fails, print an error message and return.
2. **Initialize** the customer's foodCost, servicesCost, and totalCost to 0.
3. Dynamically allocate memory for the customer's name:
   * If memory allocation fails, print an error message, free already allocated memory, and return.
4. **Prompt the user** for their name and store it in the Customer object.
5. Call the bookRoom(Customer \*cust) function to handle room booking.
6. Call the orderFood(Customer \*cust) function to handle food ordering.
7. Call the useHotelServices(Customer \*cust) function to handle hotel service usage.
8. **Calculate the total cost** (sum of room, food, and service costs).
9. **Add the customer** to the global customers array.
   * Use realloc() to expand the array dynamically.
   * If memory allocation fails, print an error message, free customer memory, and return.
10. Update the totalEarnings by adding the customer's total cost.
11. Display a success message.
12. **End**.

**Algorithm for bookRoom(Customer \*cust) Function**

1. **Display room options** with costs.
2. **Prompt the user** to choose a room type.
3. **Prompt the user** to specify the number of days they wish to stay.
4. **Calculate room cost** based on the chosen room type:
   * Suite: days \* 2500
   * 2-Person Room: days \* 1500
   * Single Room: days \* 700
   * If an invalid choice is made, default to a Single Room.
5. Update the Customer object's days and roomCost.
6. Display a confirmation message.
7. **End**.

**Algorithm for orderFood(Customer \*cust) Function**

1. Retrieve the number of days from the Customer object.
2. For each day:
   1. **Prompt the user** for breakfast, lunch, and dinner choices.
   2. Update foodCost based on the user's selections:
      * Breakfast: 100
      * Lunch: 300
      * Dinner:
        + Buffet: 500
        + Room Service: 600
3. Display the updated food cost after each meal selection.
4. At the end of the loop, display the total food cost.
5. **End**.

**Algorithm for useHotelServices(Customer \*cust) Function**

1. Display the available services with their costs:
   * Spa: 500
   * Gym: 200
   * Swimming Pool: 300
   * Massage: 400
   * Guided Tour: 600
2. **Prompt the user** for each service (Yes/No).
3. Add the corresponding cost to the Customer object's servicesCost if the user opts for a service.
4. Display the total cost of services.
5. **End**.

**Algorithm for ownerInterface() Function**

1. **Start** by validating the owner's password using the validateOwnerPassword() function.
   * If the password is invalid, return.
2. Repeat until the user chooses to exit:
   * Display the owner menu:
     + 1. View Total Earnings
     + 2. View All Customers
     + 3. Exit
   * Accept the user's choice.
   * Based on the user's choice:
     + **Case 1:** Call the viewTotalEarnings() function.
     + **Case 2:** Call the viewAllCustomers() function.
     + **Case 3:** Exit the loop.
     + **Default:** Display an "Invalid choice" message.
3. **End**.

**Algorithm for generateBill() Function**

1. **Prompt the user** for the customer's name.
2. Search for the customer in the global customers array.
3. If the customer is found:
   1. Calculate the total cost (room, food, and services).
   2. Calculate CGST and SGST (5% each).
   3. Calculate the final total (including tax).
   4. Display a detailed bill with all costs and taxes.
4. If the customer is not found, display an error message.
5. **End**.

**Algorithm for validateOwnerPassword() Function**

1. Prompt the user to enter the owner's password.
2. Compare the entered password with the predefined OWNER\_PASSWORD.
3. If the passwords match:
   * Display a success message.
   * Return 1 (valid password).
4. If the passwords do not match:
   * Display an error message.
   * Return 0 (invalid password).
5. **End**.

**Algorithm for Memory Management Functions**

**freeCustomerMemory(Customer \*cust)**

1. If the Customer object is not NULL:
   1. Free the dynamically allocated name.
   2. Free the Customer object itself.
2. **End**.

**viewAllCustomers()**

1. If there are no customers, display an appropriate message.
2. For each customer in the customers array:
   * Display their details (name, room cost, food cost, services cost, and total cost).
3. **End**.

**viewTotalEarnings()**

1. Display the value of the totalEarnings variable.
2. **End**.

CODE:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

// Customer structure

typedef struct Customer\_Details

{

char \*name;

int days;

int roomCost;

int foodCost;

int servicesCost;

int totalCost;

} Customer;

// Global variables

Customer \*\*customers = NULL; // Pointer to dynamically allocated customer array

int customerCount = 0;

int totalEarnings = 0;

const char OWNER\_PASSWORD[] = "admin123"; // Pre-defined password for owner

// Function prototypes

void bookRoom(Customer \*cust);

void orderFood(Customer \*cust);

void useHotelServices(Customer \*cust);

void customerInterface();

void generateBill();

void ownerInterface();

int validateOwnerPassword();

void viewTotalEarnings();

void viewAllCustomers();

void freeCustomerMemory(Customer \*cust);

// Room Booking

void bookRoom(Customer \*cust)

{

int roomType, days;

printf("\nWelcome to Room Booking!\n");

printf("We offer a variety of luxurious rooms to make your stay unforgettable. Please choose one:\n");

printf("1. Suite - Rs. 2,500/day\n");

printf("2. 2-Person Room - Rs. 1,500/day\n");

printf("3. Single Room - Rs. 700/day\n");

printf("Enter your choice: ");

scanf("%d", &roomType);

printf("How many days would you like to stay with us? ");

scanf("%d", &days);

cust->days = days;

switch (roomType) {

case 1: cust->roomCost = days \* 2500; break;

case 2: cust->roomCost = days \* 1500; break;

case 3: cust->roomCost = days \* 700; break;

default:

printf("Invalid choice! But don't worry, we've assigned you a cozy Single Room.\n");

cust->roomCost = days \* 700;

}

printf("Room booked for %d days! Total Room Cost: Rs. %d\n", days, cust->roomCost);

}

// Food Ordering

void orderFood(Customer \*cust) {

int days = cust->days;

int mealChoice, dinnerChoice;

int breakfastCost = 100, lunchCost = 300, buffetDinnerCost = 500, roomServiceDinnerCost = 600;

printf("\nLet's Talk About Food!\n");

printf("We provide delicious meals to make your stay delightful.\n");

for (int day = 1; day <= days; day++) {

printf("\nDay %d - Start Your Day Right!\n", day);

// Breakfast

printf("Would you like a hearty breakfast? (1 for Yes, 0 for No): ");

scanf("%d", &mealChoice);

if (mealChoice == 1) {

cust->foodCost += breakfastCost;

printf("Breakfast added for Rs. %d. Current Food Cost: Rs. %d\n", breakfastCost, cust->foodCost);

} else {

printf("Breakfast skipped. Maybe tomorrow!\n");

}

// Lunch

printf("How about a delightful lunch? (1 for Yes, 0 for No): ");

scanf("%d", &mealChoice);

if (mealChoice == 1) {

cust->foodCost += lunchCost;

printf("Lunch added for Rs. %d. Current Food Cost: Rs. %d\n", lunchCost, cust->foodCost);

} else {

printf("Lunch skipped. Don't miss our specials tomorrow!\n");

}

// Dinner

printf("End your day with a splendid dinner? (1 for Yes, 0 for No): ");

scanf("%d", &mealChoice);

if (mealChoice == 1) {

printf("Choose your dinner option (1 for Buffet - Rs. 500, 2 for Room Service - Rs. 600): ");

scanf("%d", &dinnerChoice);

if (dinnerChoice == 1) {

cust->foodCost += buffetDinnerCost;

printf("Buffet dinner added for Rs. %d. Current Food Cost: Rs. %d\n", buffetDinnerCost, cust->foodCost);

} else if (dinnerChoice == 2) {

cust->foodCost += roomServiceDinnerCost;

printf("Room service dinner added for Rs. %d. Current Food Cost: Rs. %d\n", roomServiceDinnerCost, cust->foodCost);

} else {

printf("Invalid choice! Skipping dinner for now.\n");

}

} else {

printf("Dinner skipped. Don't miss our chef's specials tomorrow!\n");

}

}

printf("\nTotal Food Cost: Rs. %d\n", cust->foodCost);

}

// Hotel Services

void useHotelServices(Customer \*cust) {

int choice;

int spaFee = 500, gymFee = 200, poolFee = 300, massageFee = 400, tourFee = 600;

int totalServiceCost = 0;

printf("\nExplore Our Premium Hotel Services\n");

printf("Choose from these amazing options:\n");

printf("1. Spa - Rs. %d\n2. Gym - Rs. %d\n3. Swimming Pool - Rs. %d\n4. Massage - Rs. %d\n5. Guided Tour - Rs. %d\n", spaFee, gymFee, poolFee, massageFee, tourFee);

printf("Would you like to indulge in a Spa? (1 for Yes, 0 for No): ");

scanf("%d", &choice);

if (choice == 1) totalServiceCost += spaFee;

printf("Want to sweat it out in our Gym? (1 for Yes, 0 for No): ");

scanf("%d", &choice);

if (choice == 1) totalServiceCost += gymFee;

printf("Take a refreshing dip in our Swimming Pool? (1 for Yes, 0 for No): ");

scanf("%d", &choice);

if (choice == 1) totalServiceCost += poolFee;

printf("Relax with a soothing Massage? (1 for Yes, 0 for No): ");

scanf("%d", &choice);

if (choice == 1) totalServiceCost += massageFee;

printf("Discover the city with a Guided Tour? (1 for Yes, 0 for No): ");

scanf("%d", &choice);

if (choice == 1) totalServiceCost += tourFee;

cust->servicesCost += totalServiceCost;

printf("Services Total Cost: Rs. %d\n", totalServiceCost);

}

// Customer Interface

void customerInterface() {

Customer \*cust = (Customer \*)malloc(sizeof(Customer));

if (!cust) {

printf("Memory allocation failed!\n");

return;

}

cust->foodCost = 0;

cust->servicesCost = 0;

cust->totalCost = 0;

printf("\nWelcome to Our Hotel!\n");

printf("We're thrilled to have you here. Let's get started with your details.\n");

cust->name = (char \*)malloc(50 \* sizeof(char));

if (!cust->name) {

printf("Memory allocation failed!\n");

free(cust);

return;

}

printf("Please enter your name: ");

scanf(" %49[^\n]", cust->name);

printf("\nThank you, %s! Let's proceed to plan your stay.\n", cust->name);

bookRoom(cust); // Book the room

orderFood(cust); // Order food

useHotelServices(cust); // Use hotel services

// Calculate total cost (room + food + services)

cust->totalCost = cust->roomCost + cust->foodCost + cust->servicesCost;

// Store the customer data

customers = (Customer \*\*)realloc(customers, (customerCount + 1) \* sizeof(Customer \*));

if (!customers) {

printf("Memory allocation failed!\n");

freeCustomerMemory(cust);

return;

}

customers[customerCount++] = cust;

totalEarnings += cust->totalCost; // Update total earnings

printf("\nAll set, %s! Your stay details have been saved.\n", cust->name);

printf("We hope you have a wonderful time at our hotel!\n");

}

// Generate Bill

void generateBill() {

char name[50];

int found = 0;

printf("\nBill Generation\n");

printf("Enter the customer's name to fetch their bill: ");

scanf(" %49[^\n]", name);

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

if (strcmp(customers[i]->name, name) == 0) {

Customer \*cust = customers[i];

cust->totalCost = cust->roomCost + cust->foodCost + cust->servicesCost;

printf("\n--- Bill for %s ---\n", cust->name);

printf("Room Cost: Rs. %d\n", cust->roomCost);

printf("Food Cost: Rs. %d\n", cust->foodCost);

printf("Services Cost: Rs. %d\n", cust->servicesCost);

// Tax Calculation

float cgst = cust->totalCost \* 0.05;

float sgst = cust->totalCost \* 0.05;

float totalTax = cgst + sgst;

// Final Total

float finalTotal = cust->totalCost + totalTax;

printf("CGST (5%%): Rs. %.2f\n", cgst);

printf("SGST (5%%): Rs. %.2f\n", sgst);

printf("Total Tax: Rs. %.2f\n", totalTax);

printf("Final Total (Including Tax): Rs. %.2f\n", finalTotal);

found = 1;

break;

}

}

if (!found) {

printf("Sorry, no records found for the name '%s'. Please check and try again.\n", name);

}

}

// Validate Owner Password

int validateOwnerPassword() {

char enteredPassword[50];

printf("Enter Owner Password: ");

scanf(" %49s", enteredPassword);

if (strcmp(enteredPassword, OWNER\_PASSWORD) == 0) {

printf("Access Granted! Welcome, esteemed owner!\n");

return 1;

} else {

printf("Invalid Password! Access Denied.\n");

return 0;

}

}

// View Total Earnings

void viewTotalEarnings() {

printf("\nTotal Earnings\n");

printf("Total earnings from all customers so far: Rs. %d\n", totalEarnings);

}

// View All Customers

void viewAllCustomers() {

printf("\nAll Customer Details\n");

if (customerCount == 0) {

printf("No customers have checked in yet. Business is just warming up!\n");

return;

}

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

Customer \*cust = customers[i];

printf("\nCustomer %d:\n", i + 1);

printf("Name: %s\n", cust->name);

printf("Room Cost: Rs. %d\n", cust->roomCost);

printf("Food Cost: Rs. %d\n", cust->foodCost);

printf("Services Cost: Rs. %d\n", cust->servicesCost);

printf("Total Cost (Excluding Tax): Rs. %d\n", cust->totalCost);

printf("\n");

}

}

// Free Customer Memory

void freeCustomerMemory(Customer \*cust) {

if (cust) {

free(cust->name);

free(cust);

}

}

// Owner Interface

void ownerInterface() {

if (!validateOwnerPassword()) return;

int choice;

do {

printf("\nOwner Interface\n");

printf("1. View Total Earnings\n");

printf("2. View All Customers\n");

printf("3. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

viewTotalEarnings();

break;

case 2:

viewAllCustomers();

break;

case 3:

printf("Exiting Owner Interface. Have a great day!\n");

break;

default:

printf("Invalid choice! Please try again.\n");

}

} while (choice != 3);

}

// Main Function

int main() {

int choice;

printf("Welcome to Our GOAT Hotel Management System!\n");

printf("We are delighted to serve you.\n");

do {

printf("\n--- Hotel Management System ---\n");

printf("1. Customer Interface\n");

printf("2. Owner Interface\n");

printf("3. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

customerInterface();

break;

case 2:

ownerInterface();

break;

case 3:

printf("Thank you for using the Hotel Management System. Goodbye!\n");

break;

default:

printf("Invalid choice! Please try again.\n");

}

} while (choice != 3);

// Free allocated memory for all customers

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

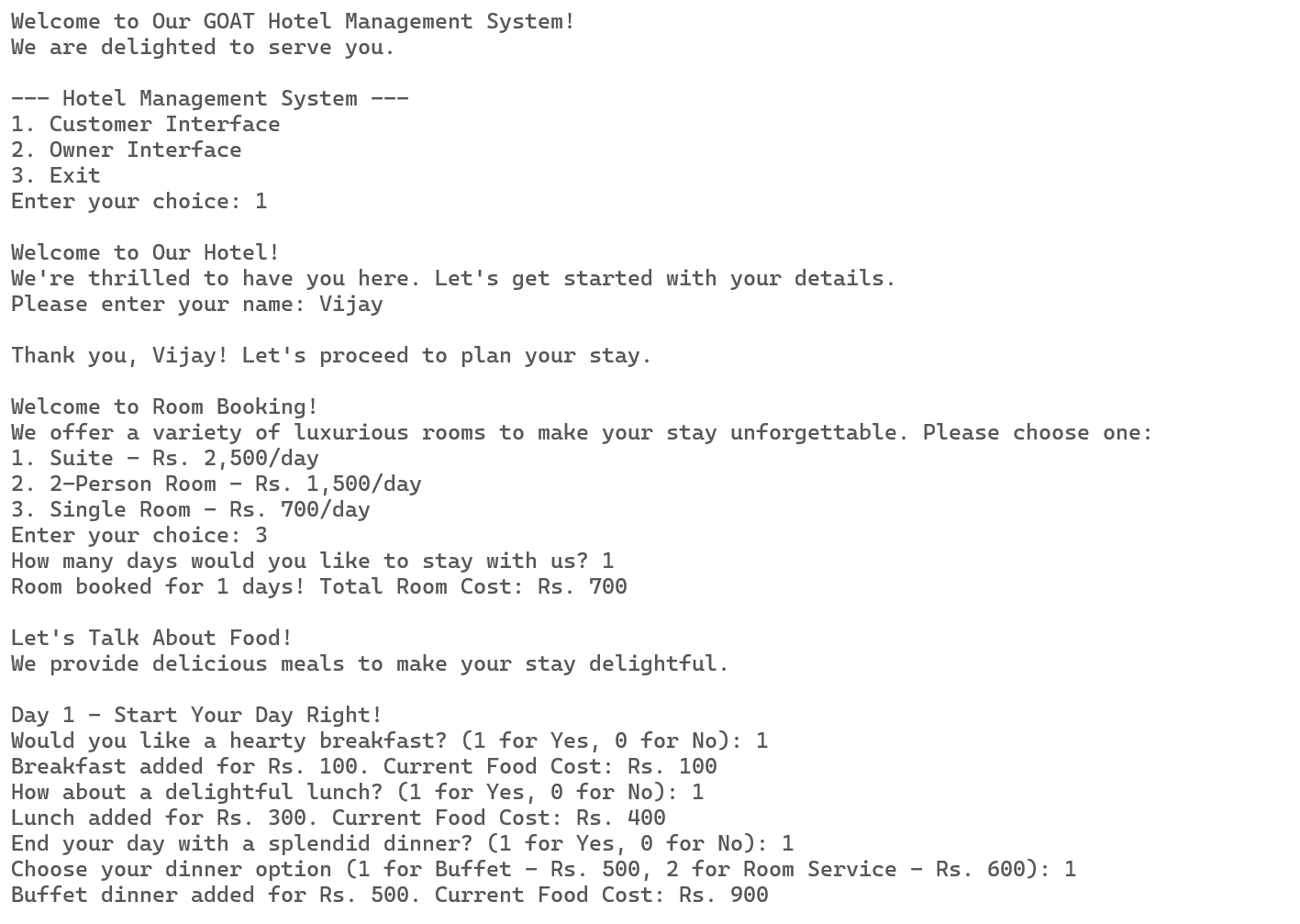
freeCustomerMemory(customers[i]);

}

free(customers);

return 0;

}

**OUTPUT**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**