HOSTEL MANAGEMENT SYSTEM

Lecturer: Dr. Noureen Talpur

GROUP MEMBERS

NAME	ID
YAP CLEMENT	22008723
ADIB THAQIFY BIN SUHAIMI	22011594
ZENG KAI MING	24001855
AIMAN NASRULLAH BIN MAS NASRULLAH	22009315

Project Idea

The goal of the Hostel Management System (HMS) is to manage hostel lodgings more efficiently by streamlining the administrative procedures. Hostel administrators may more easily oversee everyday operations with greater accuracy and efficiency thanks to this software system, which automates billing, cleaning service requests, and room distribution.

Objectives:

- To automate the process of room allocation and changes.
- To provide an easy-to-use interface for requesting cleaning services.
- To calculate and manage billing based on room type and additional services.
- To maintain data integrity by saving and loading room details from files.

Introduction & Background

The hostel management system is a comprehensive software solution designed to streamline various aspects of hostel accommodation. The program allows users to choose their type of room village, floor, room and cleaning service. This program also provides transparency, accountability and flexibility for student to select or change room and calculate the accommodation bill.

Room Management

Functionality: The system allows users to change their rooms by specifying details such as room type (shared/single), village (V1 to V6), block, floor, unit, and room number.

Implementation: The *changeRoom* function, along with *promptRoomDetails*, collects and updates room details based on user input. The room data is stored in a Room structure that includes attributes like type, village, block, floor, unit, room number, air conditioning status, cleaning service status, and price.

```
void changeRoom() {
    char type[10];
    char village[3];
    char block[4];
    int floor;
    int unit;
    int room;

    promptRoomDetails(type, village, block, floor, unit, room);
    cout << "Room change logic here\n";
    cout << "Changed to room: " << type << " in " << village << " " << block
         << " on floor " << floor << ", unit " << unit << ", room " << room
         << endl;
}
```

Sample output

```
Hostel Management System
1. Change Room
2. Cleaning Service
3. Calculate Bill
4. Save Room Details to File
5. Load Room Details from File
0. Exit
Enter your choice: 1
Enter room type (Shared/Single): shared
Enter village (V1/V2/V3/V4/V5/V6): v2
Enter block (e.g., V1a, V1b): v2b
Enter floor number: 1
Enter unit number: 4
Enter room number: 2
Room change logic here
Changed to room: shared in v2 v2b on floor 1, unit 4, room 2
```

Cleaning Service Management

Functionality: Users can request cleaning services for their rooms. The system tracks the status of these services and includes the associated costs in the final bill.

Implementation: The *`selectCleaningService`* function updates the cleaning service status for the specified room. This information is integrated into the overall room data managed by the system.

```
void selectCleaningService() {
    char type[10];
    char village[3];
    char block[4];
    int floor;
    int unit;
    int room;

    promptRoomDetails(type, village, block, floor, unit, room);
    cout << "Do you want cleaning service? (1 for Yes, 0 for No): ";
    int cleaning;
    cin >> cleaning;
    bool cleaningService = (cleaning == 1);
    cout << "Cleaning service: " << (cleaningService ? "Yes" : "No") << endl;
}
```

Sample output

```
Hostel Management System
1. Change Room
2. Cleaning Service
3. Calculate Bill
4. Save Room Details to File
5. Load Room Details from File
0. Exit
Enter your choice: 2
Enter room type (Shared/Single): 2
Enter village (V1/V2/V3/V4/V5/V6): v6
Enter block (e.g., V1a, V1b): v6b
Enter floor number: 2
Enter unit number: 4
Enter room number: 3
Do you want cleaning service? (1 for Yes, 0 for No): 1
Cleaning service: Yes
```

Billing System

Functionality: The system calculates the total bill for the user, considering factors such as room type, air conditioning, and cleaning services. It provides a detailed breakdown of the charges.

Implementation: The *calculateBill* function computes the bill by iterating through the room data and summing up the costs based on the selected services and room type.

```
void calculateBill() {
  totalBill = 0;
  for (int i = 0; i < roomCount; ++i) {
    totalBill += rooms[i].price;
  }
  cout << "Total bill: RM " << totalBill << endl;
}
```

Sample output

```
Hostel Management System
1. Change Room
2. Cleaning Service
3. Calculate Bill
4. Save Room Details to File
5. Load Room Details from File
0. Exit
Enter your choice: 3
Enter room type (Shared/Single): shared
Does the room have air conditioning? (1 for Yes, 0 for No): 1
Total bill for the semester: RM 5400
```

File Handling

Functionality: The system can save room details to a file and load them back, ensuring data persistence and easy recovery.

Implementation: The *`saveRoomDetailsToFile`* function writes the room details to a specified file, while the *`loadRoomDetailsFromFile`* function reads room details from the file.

```cpp
void saveRoomDetailsToFile(const char *filename) {
  ofstream outFile(filename);
  if (outFile.is_open()) {
    outFile << roomCount << endl;
    for (int i = 0; i < roomCount; ++i) {
      outFile << rooms[i].type << " " << rooms[i].village << " "
              << rooms[i].block << " " << rooms[i].floor << " " << rooms[i].unit
              << " " << rooms[i].room << " " << rooms[i].airCond << " "
              << rooms[i].cleaningService << " " << rooms[i].price << endl;
    }
    outFile.close();
    cout << "Room details saved to " << filename << endl;
  } else {
    cout << "Unable to open file for writing." << endl;
  }
}

void loadRoomDetailsFromFile(const char *filename) {
  ifstream inFile(filename);
  if (inFile.is_open()) {
    inFile >> roomCount;
    for (int i = 0; i < roomCount; ++i) {
      inFile >> rooms[i].type >> rooms[i].village >> rooms[i].block >>
          rooms[i].floor >> rooms[i].unit >> rooms[i].room >>
          rooms[i].airCond >> rooms[i].cleaningService >> rooms[i].price;
    }
    inFile.close();
    cout << "Room details loaded from " << filename << endl;
  } else {
    cout << "Unable to open file for reading." << endl;
  }
}
```

Sample output

```
Hostel Management System
1. Change Room
2. Cleaning Service
3. Calculate Bill
4. Save Room Details to File
5. Load Room Details from File
0. Exit
Enter your choice: 4
Room details saved to rooms.txt
```

hostel_management.cpp | rooms.txt

Hostel_management > rooms.txt

```
1000
Shared V1 V1a 0 1 1 0 1 998
Single V1 V1a 0 1 2 0 1 1100
Shared V1 V1a 0 1 3 0 1 998
Single V1 V1a 0 1 4 0 1 1100
Shared V1 V1a 0 1 5 0 1 998
Single V1 V1a 0 1 6 0 1 1100
Shared V1 V1a 0 2 1 0 1 998
Single V1 V1a 0 2 2 0 1 1100
Shared V1 V1a 0 2 3 0 1 998
Single V1 V1a 0 2 4 0 1 1100
Shared V1 V1a 0 2 5 0 1 998
Single V1 V1a 0 2 6 0 1 1100
Shared V1 V1a 0 3 1 0 1 998
Single V1 V1a 0 3 2 0 1 1100
Shared V1 V1a 0 3 3 0 1 998
Single V1 V1a 0 3 4 0 1 1100
Shared V1 V1a 0 3 5 0 1 998
Single V1 V1a 0 3 6 0 1 1100
Shared V1 V1a 0 4 1 0 1 998
Single V1 V1a 0 4 2 0 1 1100
Shared V1 V1a 0 4 3 0 1 998
Single V1 V1a 0 4 4 0 1 1100
Shared V1 V1a 0 4 5 0 1 998
Single V1 V1a 0 4 6 0 1 1100
Shared V1 V1a 1 1 1 0 1 998
Single V1 V1a 1 1 2 0 1 1100
Shared V1 V1a 1 1 3 0 1 998
Single V1 V1a 1 1 4 0 1 1100
Shared V1 V1a 1 1 5 0 1 998
Single V1 V1a 1 1 6 0 1 1100
Shared V1 V1a 1 2 1 0 1 998
Single V1 V1a 1 2 2 0 1 1100
Shared V1 V1a 1 2 3 0 1 998
Single V1 V1a 1 2 4 0 1 1100
Shared V1 V1a 1 2 5 0 1 998
Single V1 V1a 1 2 6 0 1 1100
Shared V1 V1a 1 3 1 0 1 998
Single V1 V1a 1 3 2 0 1 1100
```

PROBLEMS OUTPUT TERMINAL

User Interaction & Menu System

```cpp
void showInitialMenu() {
    cout << "\nHostel Management System\n";
    cout << "1. Change Room\n";
    cout << "2. Cleaning Service\n";
    cout << "3. Calculate Bill\n";
    cout << "4. Save Room Details to File\n";
    cout << "5. Load Room Details from File\n";
    cout << "0. Exit\n";
    cout << "Enter your choice: ";
}
```

```cpp
int main() {
    initializeRooms();

    bool exit = false;
    while (!exit) {
        showInitialMenu();
        int choice;
        cin >> choice;
        switch (choice) {
        case 1:
            changeRoom();
            break;
        case 2:
            selectCleaningService();
            break;
        case 3:
            calculateBill();
            break;
        case 4:
            saveRoomDetailsToFile("rooms.txt");
            break;
        case 5:
            loadRoomDetailsFromFile("rooms.txt");
            break;
        case 0:
            exit = true;
            break;
        default:
            cout << "Invalid choice. Try again." << endl;
        }
    }

    return 0;
}
```

Functionality: The system provides a user-friendly menu that guides users through various tasks, such as changing rooms, selecting cleaning services, calculating bills, and saving/loading room details to/from a file.

Implementation: The *`showInitialMenu`* function displays the menu options, and the `main` function handles user input, directing the flow based on the user's choices.

Sample output

```
Hostel Management System
1. Change Room
2. Cleaning Service
3. Calculate Bill
4. Save Room Details to File
5. Load Room Details from File
0. Exit
Enter your choice:
```

Source code

https://github.com/Canvas-Learning-Hub/Hostel_Management/tree/main

Thank you