PROJECT REPORT

HOSTEL ROOM ALLOCATION SYSTEM

Submitted By:

Name: Pihu Agrawal

Sap Id: 590026330

Submitted To: Mohsin Dar

Subject: Programming in C

Date: 2 December , 2025

Abstract

The Hostel Room Allocation System is a C-based console application designed to manage the allocation of hostel rooms to students efficiently. The system allows adding new students, viewing student details, searching by room number, updating records, and removing entries when a student vacates the hostel. It also saves records in a text file for persistent storage.
This project demonstrates key programming concepts in C such as file handling, structures, arrays, modular programming, user-defined functions, and menu-driven interfaces. The system aims to reduce manual errors, save time, and simplify hostel management for administrators.

Problem Definition

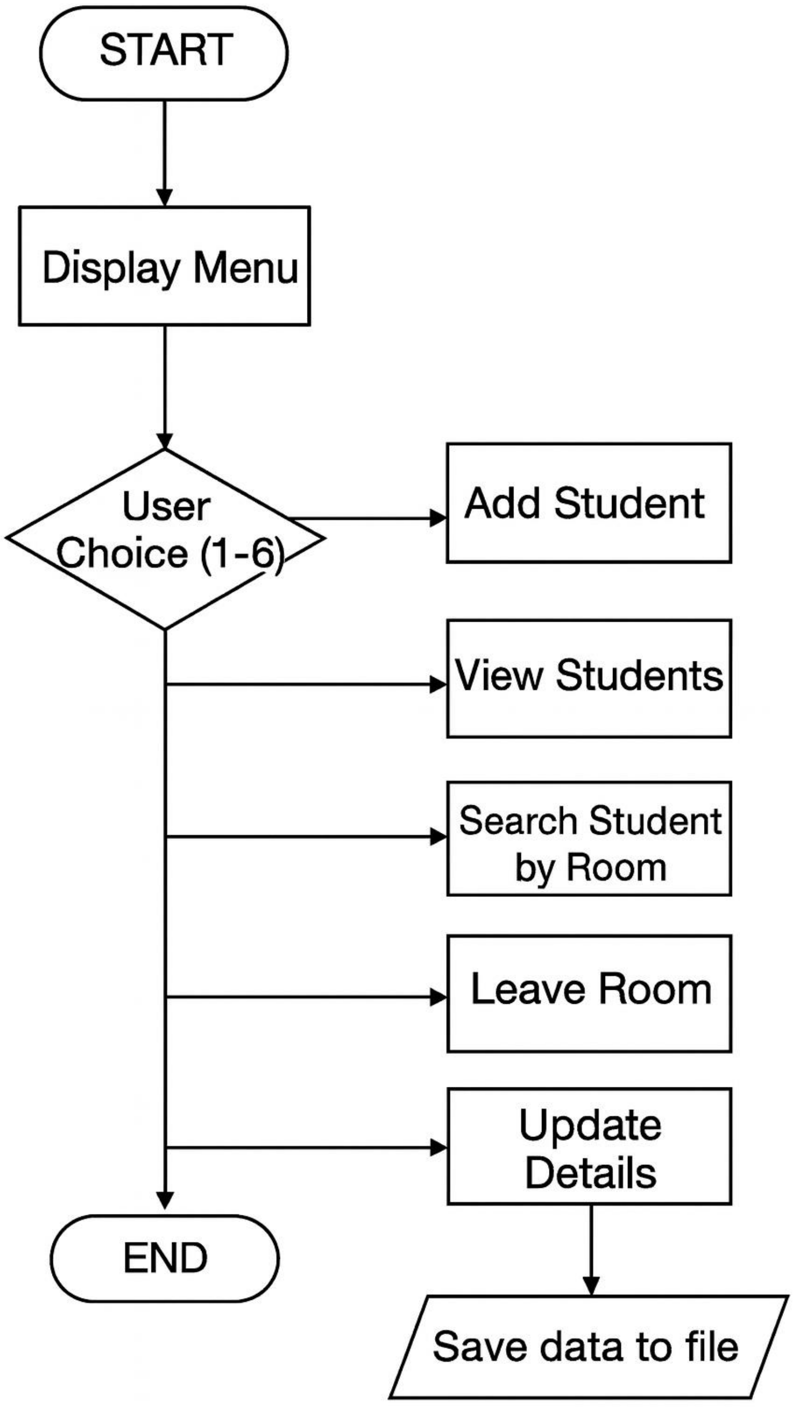
Management of hostel room allotment is typically done manually in registers, which leads to several problems:

- Manual entries are **time-consuming** and **error-prone**.
- Difficult to track **which rooms are vacant or occupied**.
- Updating student details or room changes becomes messy.
- Searching for student information requires scanning pages manually.
- Removing students after vacancy may leave inconsistent or outdated data.

Objective:
To develop a **simple, fast, and user-friendly** C program that stores student records digitally and provides essential hostel management functionalities.

System Design

Flowchart



Algorithms

Algorithm: Add Student

1. Check if maximum capacity reached.
2. Ask for room number.
3. Check if room is already taken.
4. Take student ID and name.
5. Store record in the array.
6. Save data to file.
7. Display success message.

Algorithm: View students

```
1. Prompt the user to enter the room number

2.Initialize a variable found=0.

3.For each room record in the room list, do:
a. If the room number matches the entered room number:
    i. Display the student’s name
    ii. Display the student’s ID
    iii. Display the room number
    iv. Set found=1
     v. Exit the loop

4.If found=0, display "Room not found".

 5.Stop
```

Algorithm: Search student by Room

1. Take room number from user.
2. Loop through student array.

3. If match found, display details.
4. Otherwise report “not found”.

Algorithm: leave room

```
1.Start
2.Prompt the user to enter the room number to be vacated.
3.Initialize a variable found = 0.
4.For each room record in the room list, do:
a. If the room number matches:
i. Shift all records after this position one step backward
ii. Reduce the total room count by 1
iii. Set found = 1
iv. Exit the loop
5. If found = 0, display "Room not present".
```

Algorithm: Update Student Details

1. Input Student ID.
2. Locate student in array.
3. Show existing details.
4. Ask update choice: Name, Room, or Both.
5. Validate room availability if changed.
6. Save updated data to file.

Algorithm: Save to File

```
1.Start
2.Open students.txt in write mode
3. For each student: write ID, name, room to file
4. close file
```

Implementation Details

Programming Language

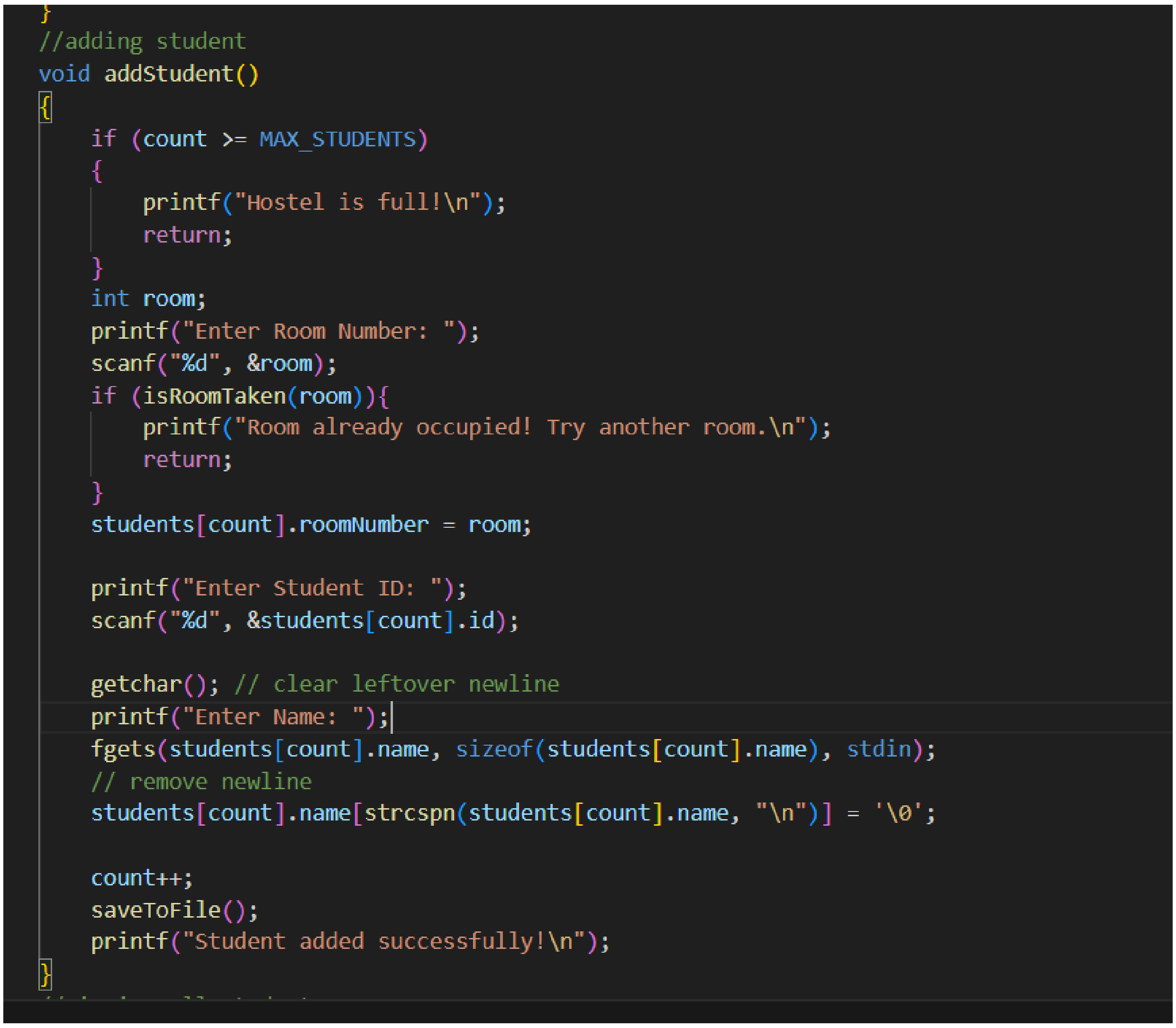
- C (ANSI Standard)

Tools Used

- VS Code
- GCC Compiler

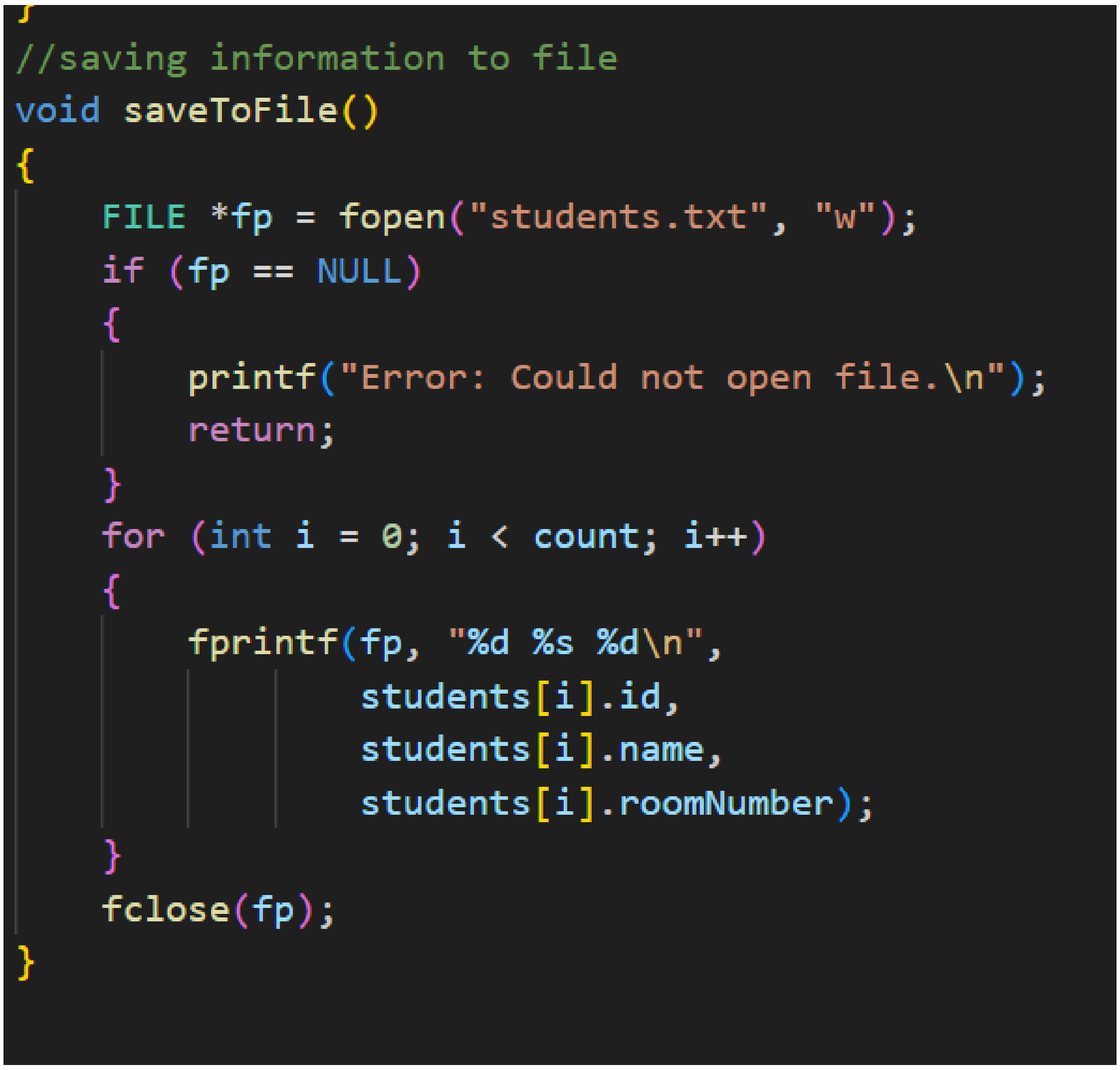
Code Snippets:

*Adding a student



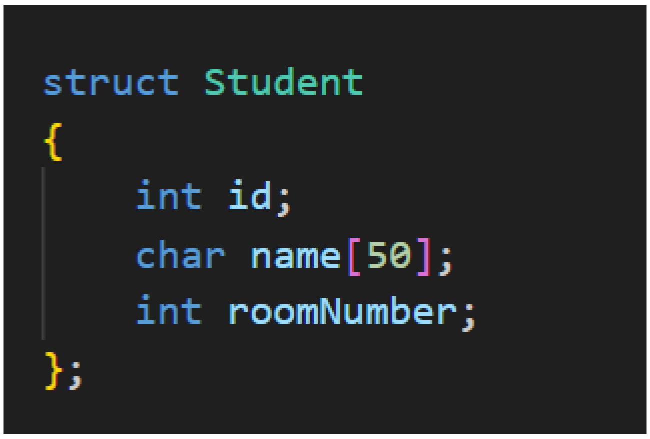
```c
}
//adding student
void addStudent()
{
    if (count >= MAX_STUDENTS)
    {
        printf("Hostel is full!\n");
        return;
    }
    int room;
    printf("Enter Room Number: ");
    scanf("%d", &room);
    if (isRoomTaken(room)){
        printf("Room already occupied! Try another room.\n");
        return;
    }
    students[count].roomNumber = room;

    printf("Enter Student ID: ");
    scanf("%d", &students[count].id);

    getchar(); // clear leftover newline
    printf("Enter Name: ");
    fgets(students[count].name, sizeof(students[count].name), stdin);
    // remove newline
    students[count].name[strcspn(students[count].name, "\n")] = '\0';

    count++;
    saveToFile();
    printf("Student added successfully!\n");
}
```

*Saving to file



```c
//saving information to file
void saveToFile()
{
    FILE *fp = fopen("students.txt", "w");
    if (fp == NULL)
    {
        printf("Error: Could not open file.\n");
        return;
    }
    for (int i = 0; i < count; i++)
    {
        fprintf(fp, "%d %s %d\n",
                students[i].id,
                students[i].name,
                students[i].roomNumber);
    }
    fclose(fp);
}
```

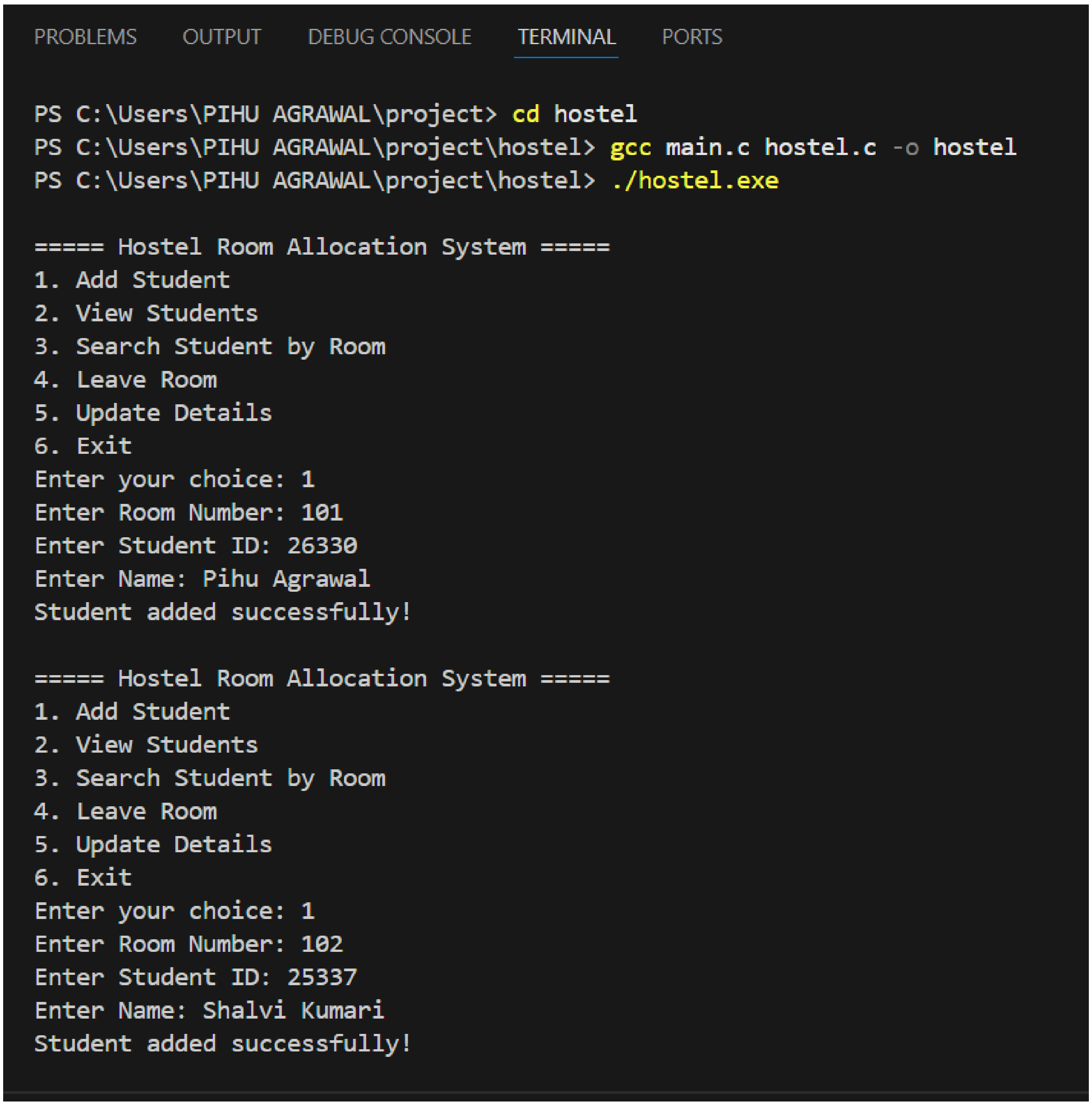
*Structure definition



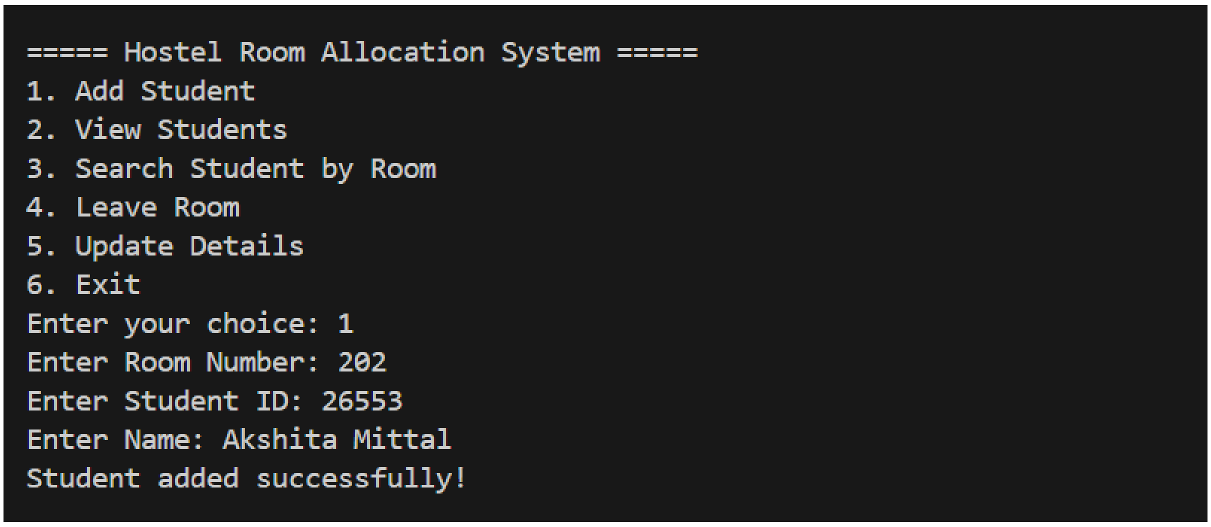
```c
struct Student
{
    int id;
    char name[50];
    int roomNumber;
};
```

Testing and result

1.Adding student:

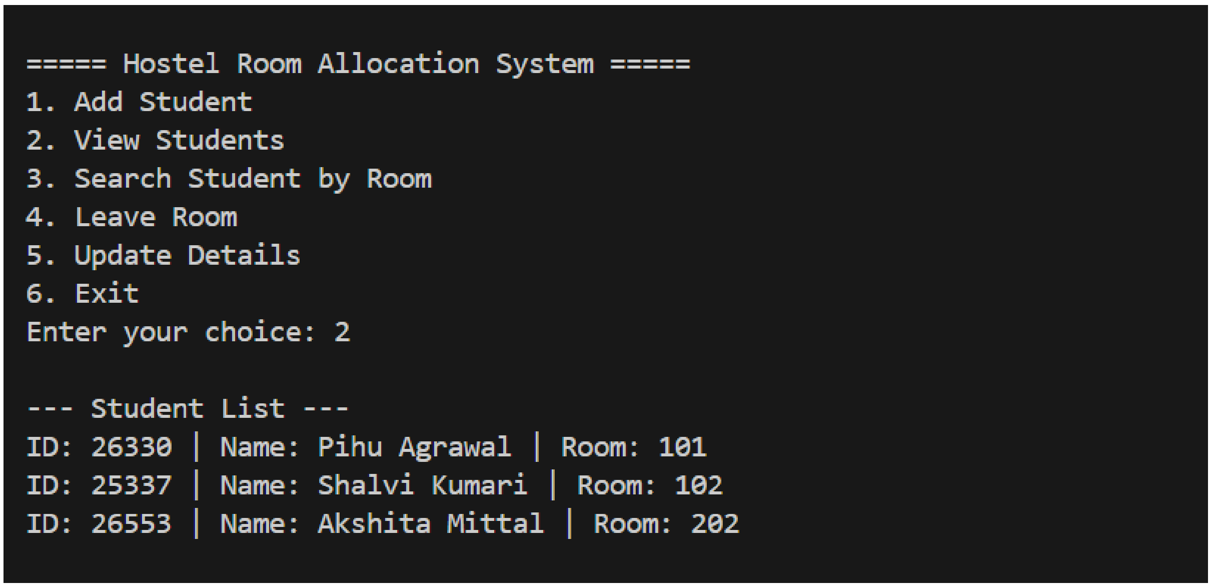


```
PS C:\Users\PIHU AGRAWAL\project> cd hostel
PS C:\Users\PIHU AGRAWAL\project\hostel> gcc main.c hostel.c -o hostel
PS C:\Users\PIHU AGRAWAL\project\hostel> ./hostel.exe

===== Hostel Room Allocation System =====
1. Add Student
2. View Students
3. Search Student by Room
4. Leave Room
5. Update Details
6. Exit
Enter your choice: 1
Enter Room Number: 101
Enter Student ID: 26330
Enter Name: Pihu Agrawal
Student added successfully!

===== Hostel Room Allocation System =====
1. Add Student
2. View Students
3. Search Student by Room
4. Leave Room
5. Update Details
6. Exit
Enter your choice: 1
Enter Room Number: 102
Enter Student ID: 25337
Enter Name: Shalvi Kumari
Student added successfully!
```



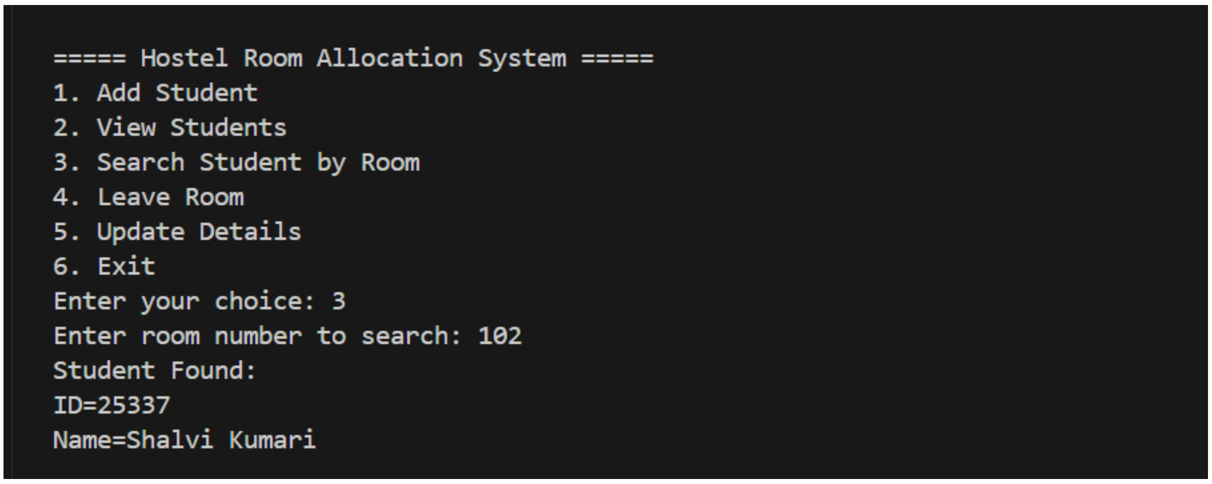
```
===== Hostel Room Allocation System =====
1. Add Student
2. View Students
3. Search Student by Room
4. Leave Room
5. Update Details
6. Exit
Enter your choice: 1
Enter Room Number: 202
Enter Student ID: 26553
Enter Name: Akshita Mittal
Student added successfully!
```

2.Viewing all added student:



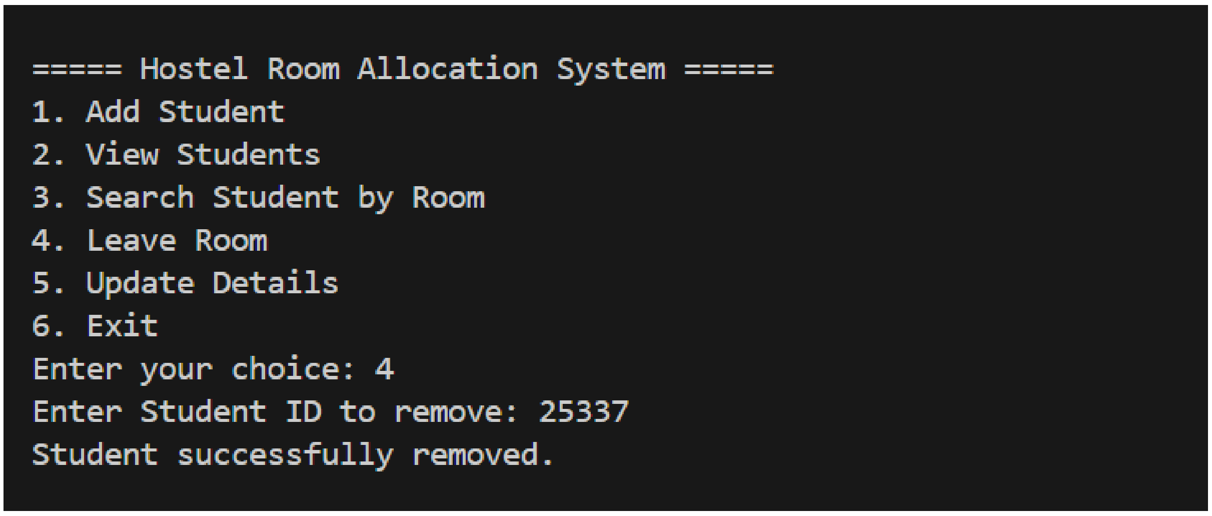
```
===== Hostel Room Allocation System =====
1. Add Student
2. View Students
3. Search Student by Room
4. Leave Room
5. Update Details
6. Exit
Enter your choice: 2

--- Student List ---
ID: 26330 | Name: Pihu Agrawal | Room: 101
ID: 25337 | Name: Shalvi Kumari | Room: 102
ID: 26553 | Name: Akshita Mittal | Room: 202
```

3.Search student by room number:



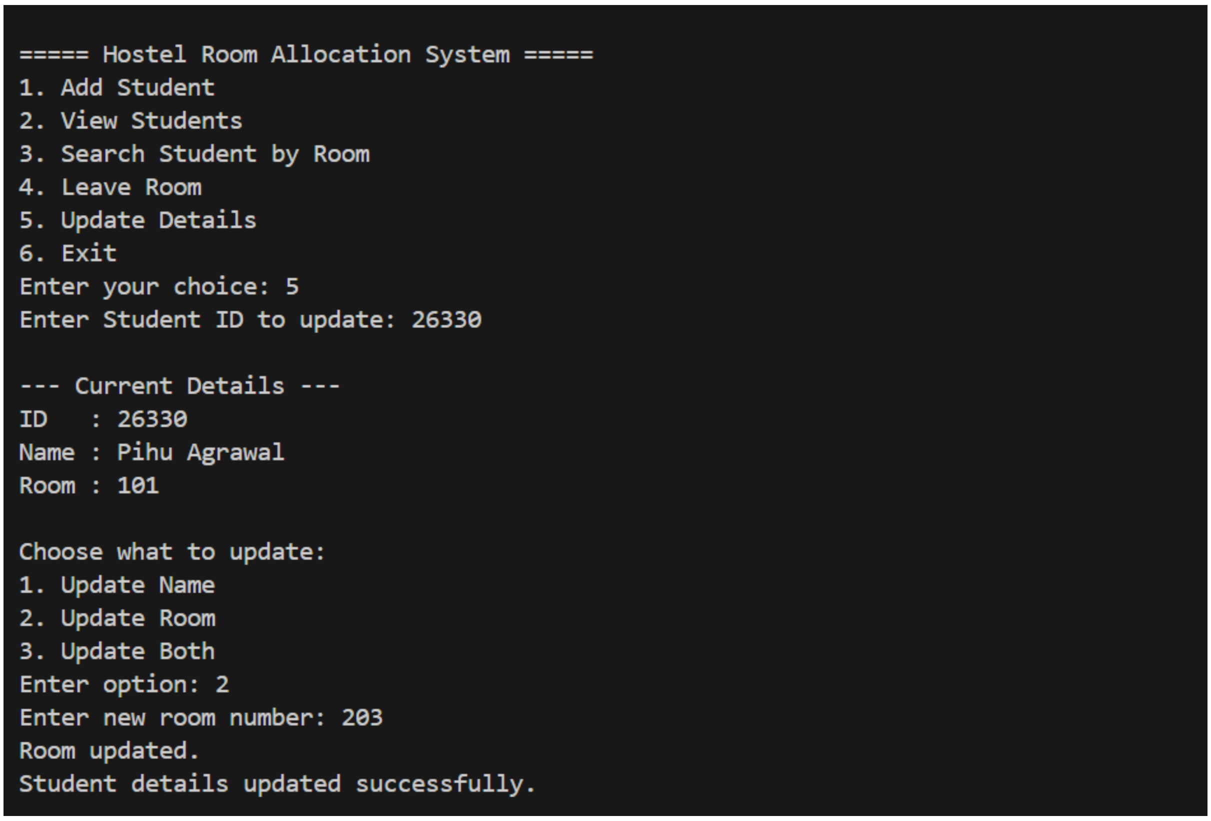
```
===== Hostel Room Allocation System =====
1. Add Student
2. View Students
3. Search Student by Room
4. Leave Room
5. Update Details
6. Exit
Enter your choice: 3
Enter room number to search: 102
Student Found:
ID=25337
Name=Shalvi Kumari
```

4.Leave room (remove student):



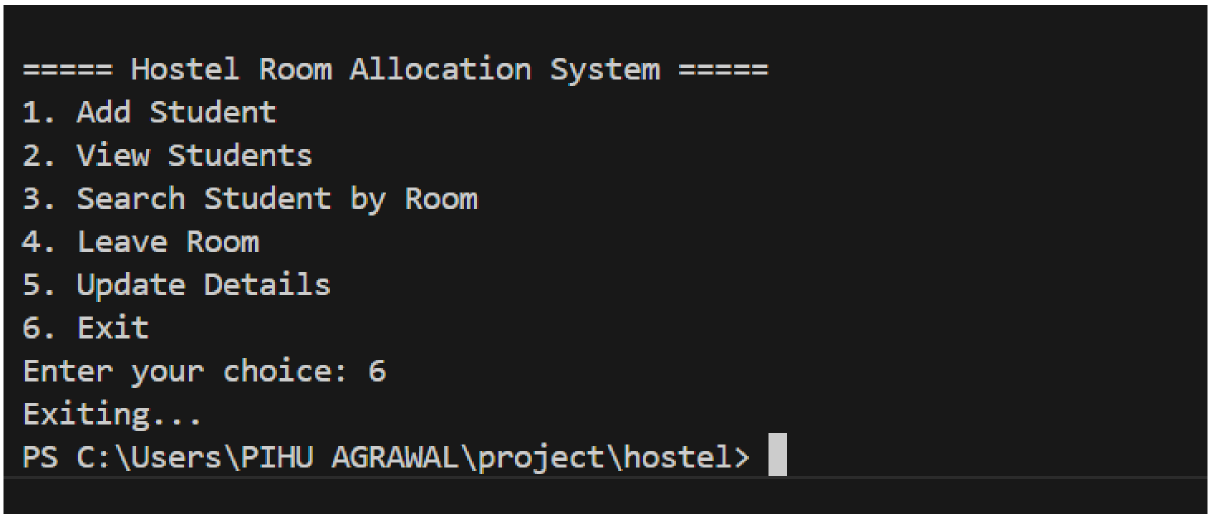
```
===== Hostel Room Allocation System =====
1. Add Student
2. View Students
3. Search Student by Room
4. Leave Room
5. Update Details
6. Exit
Enter your choice: 4
Enter Student ID to remove: 25337
Student successfully removed.
```

5.Update detail:



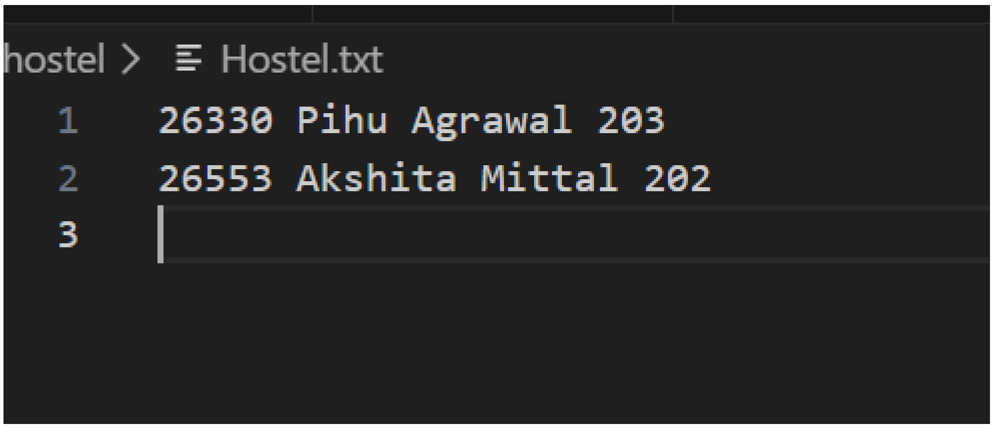
```
===== Hostel Room Allocation System =====
1. Add Student
2. View Students
3. Search Student by Room
4. Leave Room
5. Update Details
6. Exit
Enter your choice: 5
Enter Student ID to update: 26330

--- Current Details ---
ID   : 26330
Name : Pihu Agrawal
Room : 101

Choose what to update:
1. Update Name
2. Update Room
3. Update Both
Enter option: 2
Enter new room number: 203
Room updated.
Student details updated successfully.
```

6.Exiting:



```
===== Hostel Room Allocation System =====
1. Add Student
2. View Students
3. Search Student by Room
4. Leave Room
5. Update Details
6. Exit
Enter your choice: 6
Exiting...
PS C:\Users\PIHU AGRAWAL\project\hostel>
```

7.Text file containing details of student:



```
26330 Pihu Agrawal 203
26553 Akshita Mittal 202
```

Conclusion & Future Work

Conclusion

The Hostel Room Allocation System successfully automates basic hostel management tasks such as allocating rooms, updating student information, searching records, and saving data persistently.
It is simple, user-friendly, and demonstrates effective use of C programming fundamentals.

Future Enhancements

- Add a GUI interface instead of console.
- Store data in database (MySQL/SQLite) instead of text file.
- Add login system for security.
- Auto-generate room availability reports.
- Add import/export features.

References

- Let Us C — Yashavant Kanetkar
- Classroom lecture notes and pdf
- Online documentation for file handling in C

Appendix

Compilation and Executation:

```
gcc  main.c  hostel.c  -o  hostel

./hostel.c
```