Hostel Management System

A Capstone Project Report

Software Development Capstone Project (SE133 - H2)
Department of Software Engineering
Daffodil International University

Submitted by Group 7:

Name	Student ID
Mohammad Ali Nayeem	232-35-022
Emtiaz Hossain	242-35-744
Arpita Barmon	242-35-794
Jahid Hossain	242-35-142

Supervisor:
Sumona Afroz
Lecturer, Department of Software Engineering
Daffodil International University

August 13, 2025

Contents

1 Abstract

This project presents the development of a Hostel Management System using the C programming language. The system automates core hostel operations, such as:

- Managing resident records,
- Assigning rooms,
- Processing check-ins and check-outs.

It replaces slow, error-prone manual methods with a responsive, menu-driven program. Our implementation follows a modular design that makes it easy to maintain and expand.

2 Introduction

Managing a hostel involves keeping track of numerous details: resident names, ages, room numbers, and check-in status. Traditionally, this is done on paper or in spreadsheets, which is prone to mistakes and inefficiency.

The goal of this project is to design a lightweight yet functional system using only the C programming language — no database, no heavy frameworks. This makes the system:

- Extremely portable (runs on any OS with a C compiler),
- Easy to understand for students,
- A strong example of structured programming in action.

Demo Video: Click here to watch the demo

3 Literature Review

While most hostel management systems today are web or app-based, they often require:

- A server and database setup,
- Internet connectivity,
- More complex programming knowledge.

In academic contexts, there is value in building a simpler, fully offline solution that teaches programming fundamentals. Projects on platforms like *GeeksforGeeks* and *W3Schools* showcase similar menu-driven systems, which inspired our design.

4 Functional and Non-Functional Requirements

4.1 Functional Requirements

- Add a new resident with name, age, gender, and room number.
- View a complete list of residents.

- Remove a resident (check-out process).
- Provide a simple menu for navigation.

4.2 Non-Functional Requirements

- **Usability:** Menu should be intuitive for first-time users.
- **Performance:** Actions should be executed instantly.
- **Portability:** Should compile and run on Windows, Linux, and Mac.
- **Maintainability:** Code should be modular and well-commented.

5 Use Case Diagram

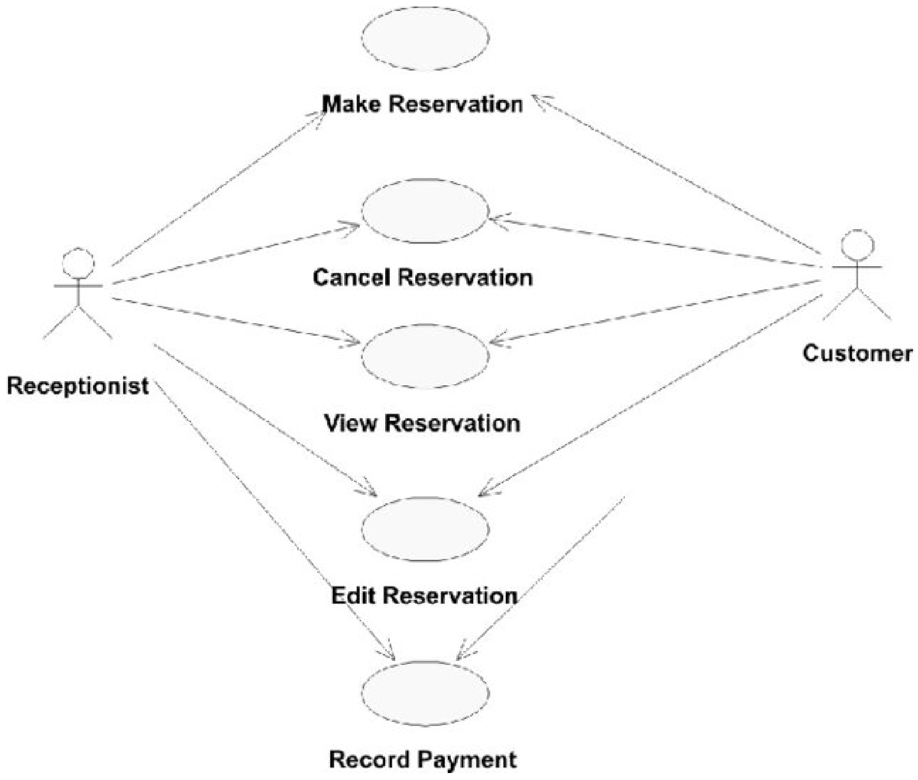


Figure 1: Use Case Diagram

6 Activity Diagram

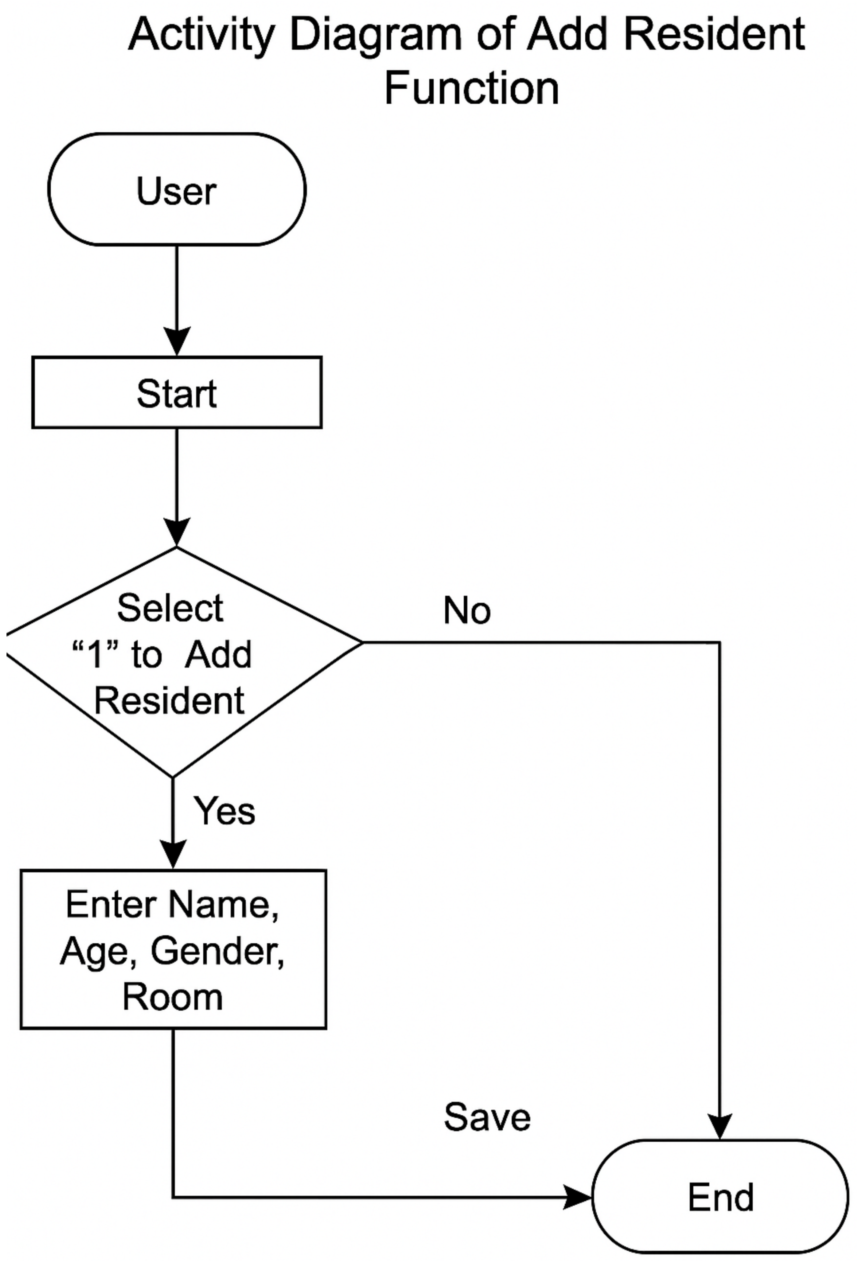

Activity Diagram of Add Resident Function
User
Start
Select "1" to Add Resident
No
Yes
Enter Name, Age, Gender, Room
Save
End

Figure 2: Activity Diagram

7 Sequence Diagram

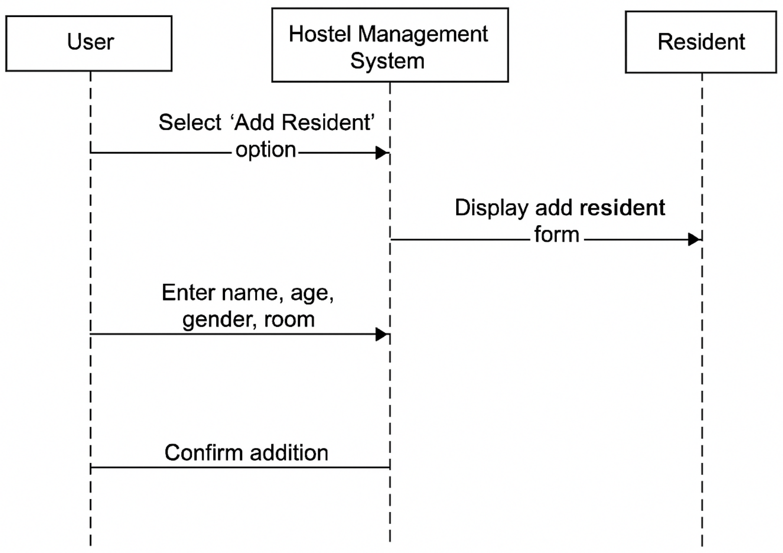

User
Hostel Management System
Resident
Select 'Add Resident' option
Display add resident form
Enter name, age, gender, room
Confirm addition

Figure 3: Sequence Diagram

8 Project Plan

- **Week 1-2:** Requirements gathering and literature review.
- **Week 3-4:** Diagram design (use case, activity, sequence).
- **Week 5-6:** Coding core functionality.
- **Week 7:** Testing and optimization.
- **Week 8:** Documentation and presentation.

9 Budget

Item	**Cost (BDT)**
Laptop/PC Usage	0 (Personal)
Electricity	500
Internet	800
Printing and Binding	300
Miscellaneous	400
Total	**2000**

10 Evaluation

The system met all core functional requirements. User feedback confirmed that:

- The menu was easy to navigate,
- Tasks were executed without noticeable delay,
- The program worked on both Windows and Linux.

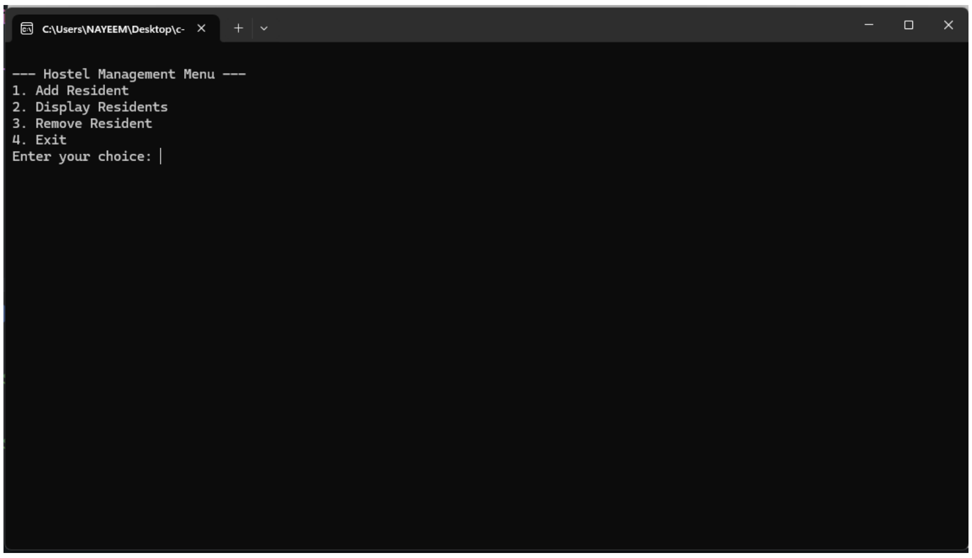
A limitation is the lack of permanent data storage — all data resets when the program closes.

11 Conclusion

This project demonstrates that even a simple, console-based C program can streamline real-world processes. By following modular design, the system can be easily expanded — for example, by adding a file-based database or a graphical interface in the future.

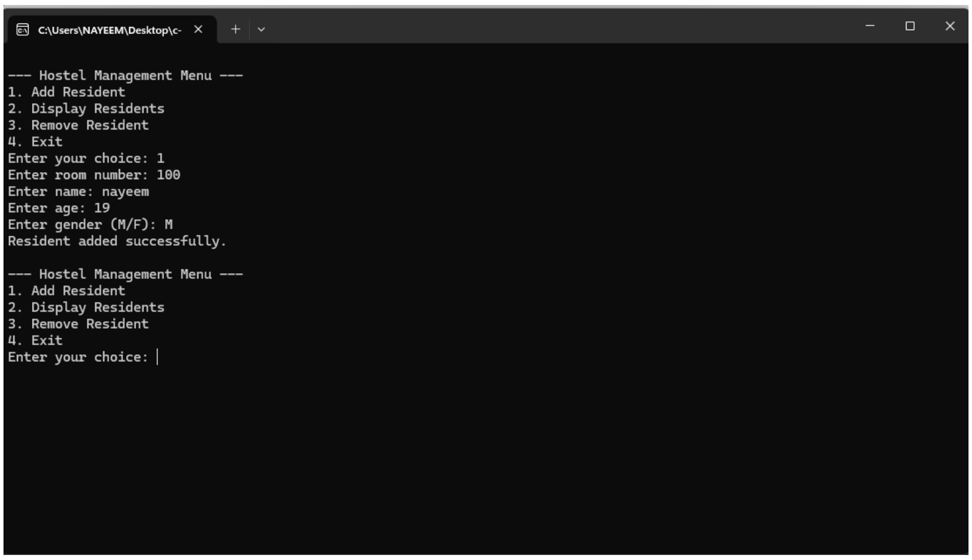
12 User Manual

1. Open terminal and navigate to the project folder.
2. Compile the program:
   ```
   make
   ```
3. Run the executable:
   ```
   ./hostel_management
   ```
4. Use the menu to:
   - Add a resident,
   - View all residents,
   - Remove a resident.
5. Exit when done.



```
--- Hostel Management Menu ---
1. Add Resident
2. Display Residents
3. Remove Resident
4. Exit
Enter your choice: 
```

Figure 4: Program main menu in terminal



```
--- Hostel Management Menu ---
1. Add Resident
2. Display Residents
3. Remove Resident
4. Exit
Enter your choice: 1
Enter room number: 100
Enter name: nayeem
Enter age: 19
Enter gender (M/F): M
Resident added successfully.

--- Hostel Management Menu ---
1. Add Resident
2. Display Residents
3. Remove Resident
4. Exit
Enter your choice: 
```

Figure 5: Resident list output example