HOSTEL MANAGEMENT SYSTEM(RESERVATION)

OBJECT ORIENTED PROGRAMMING (CSC2002)

– PROJECT COMPONENT - REVIEW REPORT

Submitted by

M.SHARAN(22BCS0016)
S.SUSEENDHARRAM(22BCS0185)
P.DHANUSH(22BCS0167)
S.JEEVA(22BCS0142)

Submitted To

R.S.Rampriya

Assistant Professor

SCHOOL OF INFORMATION TECHNOLOGY AND ENGINEERING

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

June, 2023

Table of Contents

1. **Abstract** 1
2. **Introduction** 1
3. **Project SCOPE** 1
4. **Key Contacts and Stakeholders** 2
5. **Project Resource Requirements** 2
   5.1 Software Resource Requirements 2
6. **Functions Used (Description)** 2
7. **Object Oriented Features Used (with code snippet – screenshot)** 3
8. **C++ Code (Entire code)** 4
9. **Output Screenshot** 18
10. **Conclusion & Future Work** 21
11. **Functions break down** 22
12. **Review evaluation** 22

1. ABSTRACT

*The Hostel Management System is a project that provide a platform for managing and organizing student accommodation in a hostel.

*The system allows an administrator to perform various functions such as adding and deleting students, checking room availability, modifying student details, and managing visitors.

* The project utilizes classes to represent students and rooms, with functionalities to add and remove occupants, keep track of visitor information, and display relevant information.

* It provides an interactive menu-based interface for the administrator to perform these operations.

*The Hostel Management System aims to streamline the management of hostel facilities and simplify the administrative tasks involved in student accommodation.

2. INTRODUCTION

The Hostel Management System can be used by

- Educational institutes
- Colleges
- Schools

.To maintain the records of students easily. It also provides a less time consuming process for viewing, adding, editing and deleting the info of the students.

• By this system we can store all details of the students,we can also see vacansies of the beds of particular room .And we can also see and add the visitors details.

3. PROJECT SCOPE [LIST ALL THE FUNCTIONALITIES HERE]

*Hostel Management System aims at automation of the following processes

1. Add Student: Allows the administrator to add a new student to a specific room by providing details. The student is assigned to a room based on the room number entered.

2. Delete Student: Allows the administrator to delete a student from the system by providing the student's ID.

3. Show Room Availability: Displays the availability of each room in the hostel.

4. Show Occupied Rooms: Displays the list of rooms that have occupied by the student..

5.Modify Student Details: Allows the administrator to modify the details of a student by providing the student's ID.

6. **Add Visitor:** Allows the administrator to add a visitor for a specific student.

7. Delete Visitor: Allows the administrator to delete the visitor for a specific student.

8. Exit: Exits the program and ends the execution.

4. KEY CONTACTS AND STAKEHOLDERS

NAME	REGISTER NUMBER	PHONE NUMBER
SHARAN.M	22BCS0016	93453 53240
SUSEENDHARRAM.S	22BCS0185	81481 79558
DHANUSH.P	22BCS0167	93446 16587
JEEVA.S	22BCS0142	63797 75973

STAKEHOLDERS	DETAILS
ADMIN	HOSTEL

5. PROJECT RESOURCE REQUIREMENTS

5.1 Software Resource Requirements

.Code Blocks

6. FUNCTION USED [WITH DESCRIPTION & DIAGRAM (IF POSSIBLE)]

*Let's go through the functions used in the code .

1)bool authenticateAdmin()

* This function is used to authenticate the administrator by comparing the provided admin ID and password with the hardcoded values. It returns true if the authentication is successful.

2)addStudent()

* This function allows the administrator to add a new student to a specific room.

3) deleteStudent()

*This function allows the administrator to delete a student from any room by providing the student ID.

4)showAvailability()

* This function displays the availability of each room by showing the number of occupied and available beds.

5) showOccupiedRooms()

*This function displays a list of occupied rooms and the students occupying each room.

6)modifyStudent()

* This function allows the administrator to modify the details of a student by providing the student ID.

7)addVisitor()

* This function adds a visitor's name to the “visitors” vector for a student.

8) deleteVisitor()

*This function removes the visitor from the “visitors” vector for a student.

9)displayMenu()

*This function displays the menu options for the hostel management system.

10)run()

* This function is the entry point of the program. It initializes the rooms, prompts the administrator for login credentials, and presents the menu options. It calls the appropriate functions based on the administrator's choice until the program is exited.

11)main()

* The main function creates an instance of the HostelManagementSystem class and calls the run() function to start the hostel management system.

7. VARIOUS OBJECT ORIENTED FEATURES USED

*This code demonstrates ,

- Classes and Object.
- Constructors.
- Encapsulation.
- Member Functions.
- Inheritance.
- Control Structures.

- Vectors.

8. C++ CODE

```
#include <iostream>
#include <string>
#include <vector>

using namespace std;

class Student {
private:
    string name;
    int id;
    string gender;
    string course;
    string fatherName;
    string motherName;
    string address;
    string phoneNumber;
    vector<string> visitors;

public:
    Student(string name, int id, string gender, string course, string fatherName, string motherName, string
address, string phoneNumber)
        : name(name), id(id), gender(gender), course(course), fatherName(fatherName),
motherName(motherName), address(address), phoneNumber(phoneNumber) {}

    string getName() const {
        return name;
    }

    int getId() const {
        return id;
    }

    string getGender() const {
        return gender;
    }

    string getCourse() const {
        return course;
    }

    string getFatherName() const {
        return fatherName;
    }

    string getMotherName() const {
```

```
        return motherName;
    }

    string getAddress() const {
        return address;
    }

    string getPhoneNumber() const {
        return phoneNumber;
    }

    vector<string> getVisitors() const {
        return visitors;
    }

    void addVisitor(const string& visitorName) {
        visitors.push_back(visitorName);
    }

    void deleteVisitor() {
        if (!visitors.empty()) {
            visitors.pop_back();
        }
    }
};

class Room {
private:
    int roomNumber;
    int floor;
    int capacity;
    vector<Student> occupants;

public:
    Room(int roomNumber, int floor, int capacity)
        : roomNumber(roomNumber), floor(floor), capacity(capacity) {}

    int getRoomNumber() const {
        return roomNumber;
    }

    int getFloor() const {
        return floor;
    }

    int getCapacity() const {
        return capacity;
    }

    vector<Student> getOccupants() const {
        return occupants;
```

```
    }

    void addOccupant(const Student& student) {
        occupants.push_back(student);
    }

    void removeOccupant(int studentId) {
        for (size_t i = 0; i < occupants.size(); i++) {
            if (occupants[i].getId() == studentId) {
                occupants.erase(occupants.begin() + i);
                break;
            }
        }
    }
};

class HostelManagementSystem {
private:
    vector<Room> rooms;

    bool authenticateAdmin(int adminId, string password) {
        return (adminId == 123 && password == "password");
    }

    void addStudent() {
        cout << "Enter student details:" << endl;
        string name, gender, course, fatherName, motherName, address, phoneNumber;
        int id;

        cout << "Name: ";
        getline(cin >> ws, name);
        cout << "ID: ";
        cin >> id;
        cout << "Gender: ";
        getline(cin >> ws, gender);
        cout << "Course: ";
        getline(cin >> ws, course);
        cout << "Father's Name: ";
        getline(cin >> ws, fatherName);
        cout << "Mother's Name: ";
        getline(cin >> ws, motherName);
        cout << "Address: ";
        getline(cin >> ws, address);
        cout << "Phone Number: ";
        getline(cin >> ws, phoneNumber);

        cout << "Enter room number: ";
        int roomNumber;
        cin >> roomNumber;

        bool roomFound = false;
```

```
        for (Room& room : rooms) {
            if (room.getRoomNumber() == roomNumber) {
                if (room.getOccupants().size() < room.getCapacity()) {
                    Student student(name, id, gender, course, fatherName, motherName, address,
phoneNumber);
                    room.addOccupant(student);
                    cout << "Student added successfully to room " << roomNumber << endl;
                } else {
                    cout << "Room " << roomNumber << " is already full" << endl;
                }
                roomFound = true;
                break;
            }
        }

        if (!roomFound) {
            cout << "Room " << roomNumber << " not found" << endl;
        }
    }

    void deleteStudent() {
        cout << "Enter student ID to delete: ";
        int id;
        cin >> id;

        for (Room& room : rooms) {
            room.removeOccupant(id);
        }

        cout << "Student with ID " << id << " deleted successfully" << endl;
    }

    void showAvailability() {
        cout << "Room Availability:" << endl;
        cout << "------------------" << endl;

        for (const Room& room : rooms) {
            cout << "Room " << room.getRoomNumber() << ", Floor " << room.getFloor() << ": ";
            cout << room.getCapacity() - room.getOccupants().size() << "/" << room.getCapacity() << " beds
available" << endl;
        }
    }

    void showOccupiedRooms() {
        cout << "Occupied Rooms:" << endl;
        cout << "---------------" << endl;

        for (const Room& room : rooms) {
            if (!room.getOccupants().empty()) {
                cout << "Room " << room.getRoomNumber() << ", Floor " << room.getFloor() << ":" << endl;
                for (const Student& student : room.getOccupants()) {
```

```
                cout << "  - " << student.getName() << " (ID: " << student.getId() << ")" << endl;
            }
        }
    }
}

void modifyStudent() {
    cout << "Enter student ID to modify: ";
    int id;
    cin >> id;

    for (Room& room : rooms) {
        for (Student& student : room.getOccupants()) {
            if (student.getId() == id) {
                cout << "Enter new details for student with ID " << id << ":" << endl;

                string name, gender, course, fatherName, motherName, address, phoneNumber;

                cout << "Name: ";
                getline(cin >> ws, name);
                cout << "Gender: ";
                getline(cin >> ws, gender);
                cout << "Course: ";
                getline(cin >> ws, course);
                cout << "Father's Name: ";
                getline(cin >> ws, fatherName);
                cout << "Mother's Name: ";
                getline(cin >> ws, motherName);
                cout << "Address: ";
                getline(cin >> ws, address);
                cout << "Phone Number: ";
                getline(cin >> ws, phoneNumber);

                student = Student(name, id, gender, course, fatherName, motherName, address,
phoneNumber);

                cout << "Student with ID " << id << " modified successfully" << endl;
                return;
            }
        }
    }

    cout << "Student with ID " << id << " not found" << endl;
}

void addVisitor() {
    cout << "Enter visitor details:" << endl;
    string visitorName;
    cout << "Name: ";
    getline(cin >> ws, visitorName);
```

```
    cout << "Enter student ID to visit: ";
    int studentId;
    cin >> studentId;

    bool studentFound = false;
    for (Room& room : rooms) {
        for (Student& student : room.getOccupants()) {
            if (student.getId() == studentId) {
                if (student.getVisitors().size() < room.getCapacity()) {
                    student.addVisitor(visitorName);
                    cout << "Visitor added successfully for student with ID " << studentId << endl;
                } else {
                    cout << "Room is already full" << endl;
                }
                studentFound = true;
                break;
            }
        }
        if (studentFound) {
            break;
        }
    }

    if (!studentFound) {
        cout << "Student with ID " << studentId << " not found" << endl;
    }
}

void deleteVisitor() {
    cout << "Enter student ID to delete visitor: ";
    int studentId;
    cin >> studentId;

    bool studentFound = false;
    for (Room& room : rooms) {
        for (Student& student : room.getOccupants()) {
            if (student.getId() == studentId) {
                if (!student.getVisitors().empty()) {
                    student.deleteVisitor();
                    cout << "Visitor deleted successfully" << endl;
                } else {
                    cout << "No visitors found for student with ID " << studentId << endl;
                }
                studentFound = true;
                break;
            }
        }
        if (studentFound) {
            break;
        }
    }
```

```
        if (!studentFound) {
            cout << "Student with ID " << studentId << " not found" << endl;
        }
    }

    void displayMenu() {
        cout << "Hostel Management System" << endl;
        cout << "------------------------" << endl;
        cout << "1. Add Student" << endl;
        cout << "2. Delete Student" << endl;
        cout << "3. Show Room Availability" << endl;
        cout << "4. Show Occupied Rooms" << endl;
        cout << "5. Modify Student Details" << endl;
        cout << "6. Add Visitor" << endl;
        cout << "7. Delete Visitor" << endl;
        cout << "8. Exit" << endl;
        cout << "Enter your choice: ";
    }

public:
    void run() {
        rooms = {
            Room(101, 1, 2),
            Room(102, 1, 2),
            Room(201, 2, 3),
            Room(202, 2, 3),
            Room(301, 3, 4),
            Room(302, 3, 4)
        };

        int adminId;
        string password;

        cout << "Admin Login" << endl;
        cout << "-----------" << endl;
        cout << "Enter Admin ID: ";
        cin >> adminId;
        cout << "Enter Password: ";
        cin >> password;

        if (authenticateAdmin(adminId, password)) {
            int choice;
            do {
                displayMenu();
                cin >> choice;

                switch (choice) {
                    case 1:
                        addStudent();
                        break;
```

```
                case 2:
                    deleteStudent();
                    break;
                case 3:
                    showAvailability();
                    break;
                case 4:
                    showOccupiedRooms();
                    break;
                case 5:
                    modifyStudent();
                    break;
                case 6:
                    addVisitor();
                    break;
                case 7:
                    deleteVisitor();
                    break;
                case 8:
                    cout << "Exited successfully" << endl;
                    break;
                default:
                    cout << "Invalid option" << endl;
            }

            cout << endl;
        } while (choice != 9);
    } else {
        cout << "Authentication failed. Exiting program." << endl;
    }
  }
};

int main() {
  HostelManagementSystem system;
  system.run();

  return 0;
}
```

```cpp
#include <iostream>
#include <string>
#include <vector>

using namespace std;

class Student {
private:
    string name;
    int id;
    string gender;
    string course;
    string fatherName;
    string motherName;
    string address;
    string phoneNumber;
    vector<string> visitors;

public:
    Student(string name, int id, string gender, string course, string fatherName, string motherName, string address, string phoneNumber)
        : name(name), id(id), gender(gender), course(course), fatherName(fatherName), motherName(motherName), address(address), phoneNumber(phoneNumber) {}

    string getName() const {
        return name;
    }

    int getId() const {
        return id;
    }

    string getGender() const {
        return gender;
    }

    string getCourse() const {
        return course;
    }

    string getFatherName() const {
        return fatherName;
    }

    string getMotherName() const {
        return motherName;
    }

    string getAddress() const {
        return address;
    }

    string getPhoneNumber() const {
        return phoneNumber;
    }

    vector<string> getVisitors() const {
        return visitors;
    }

    void addVisitor(const string& visitorName) {
        visitors.push_back(visitorName);
    }

    void deleteVisitor() {
        if (!visitors.empty()) {
            visitors.pop_back();
        }
    }
};

class Room {
private:
    int roomNumber;
    int floor;
    int capacity;
    vector<Student> occupants;
```

```
public:
    Room(int roomNumber, int floor, int capacity)
        : roomNumber(roomNumber), floor(floor), capacity(capacity) {}

    int getRoomNumber() const {
        return roomNumber;
    }

    int getFloor() const {
        return floor;
    }

    int getCapacity() const {
        return capacity;
    }

    vector<Student> getOccupants() const {
        return occupants;
    }

    void addOccupant(const Student& student) {
        occupants.push_back(student);
    }

    void removeOccupant(int studentId) {
        for (size_t i = 0; i < occupants.size(); i++) {
            if (occupants[i].getId() == studentId) {
                occupants.erase(occupants.begin() + i);
                break;
            }
        }
    }
};
```

```
class HostelManagementSystem {
private:
    vector<Room> rooms;

    bool authenticateAdmin(int adminId, string password) {
        return (adminId == 123 && password == "password");
    }

    void addStudent() {
        cout << "Enter student details:" << endl;
        string name, gender, course, fatherName, motherName, address, phoneNumber;
        int id;

        cout << "Name: ";
        getline(cin >> ws, name);
        cout << "ID: ";
        cin >> id;
        cout << "Gender: ";
        getline(cin >> ws, gender);
        cout << "Course: ";
        getline(cin >> ws, course);
        cout << "Father's Name: ";
        getline(cin >> ws, fatherName);
        cout << "Mother's Name: ";
        getline(cin >> ws, motherName);
        cout << "Address: ";
        getline(cin >> ws, address);
        cout << "Phone Number: ";
        getline(cin >> ws, phoneNumber);

        cout << "Enter room number: ";
        int roomNumber;
        cin >> roomNumber;
```

```
    bool roomFound = false;
    for (Room& room : rooms) {
        if (room.getRoomNumber() == roomNumber) {
            if (room.getOccupants().size() < room.getCapacity()) {
                Student student(name, id, gender, course, fatherName, motherName, address, phoneNumber);
                room.addOccupant(student);
                cout << "Student added successfully to room " << roomNumber << endl;
            } else {
                cout << "Room " << roomNumber << " is already full" << endl;
            }
            roomFound = true;
            break;
        }
    }

    if (!roomFound) {
        cout << "Room " << roomNumber << " not found" << endl;
    }
}

void deleteStudent() {
    cout << "Enter student ID to delete: ";
    int id;
    cin >> id;

    for (Room& room : rooms) {
        room.removeOccupant(id);
    }

    cout << "Student with ID " << id << " deleted successfully" << endl;
}

void showAvailability() {
    cout << "Room Availability:" << endl;
    cout << "------------------" << endl;

    for (const Room& room : rooms) {
        cout << "Room " << room.getRoomNumber() << ", Floor " << room.getFloor() << ": ";
        cout << room.getCapacity() - room.getOccupants().size() << "/" << room.getCapacity() << " beds available" << endl;
    }
}

void showOccupiedRooms() {
    cout << "Occupied Rooms:" << endl;
    cout << "---------------" << endl;

    for (const Room& room : rooms) {
        if (!room.getOccupants().empty()) {
            cout << "Room " << room.getRoomNumber() << ", Floor " << room.getFloor() << ":" << endl;
            for (const Student& student : room.getOccupants()) {
                cout << "  - " << student.getName() << " (ID: " << student.getId() << ")" << endl;
            }
        }
    }
}
```

```
    void modifyStudent() {
        cout << "Enter student ID to modify: ";
        int id;
        cin >> id;

        for (Room& room : rooms) {
            for (Student& student : room.getOccupants()) {
                if (student.getId() == id) {
                    cout << "Enter new details for student with ID " << id << ":" << endl;

                    string name, gender, course, fatherName, motherName, address, phoneNumber;

                    cout << "Name: ";
                    getline(cin >> ws, name);
                    cout << "Gender: ";
                    getline(cin >> ws, gender);
                    cout << "Course: ";
                    getline(cin >> ws, course);
                    cout << "Father's Name: ";
                    getline(cin >> ws, fatherName);
                    cout << "Mother's Name: ";
                    getline(cin >> ws, motherName);
                    cout << "Address: ";
                    getline(cin >> ws, address);
                    cout << "Phone Number: ";
                    getline(cin >> ws, phoneNumber);

                    student = Student(name, id, gender, course, fatherName, motherName, address, phoneNumber);

                    cout << "Student with ID " << id << " modified successfully" << endl;
                    return;
                }
            }
        }

        cout << "Student with ID " << id << " not found" << endl;
    }

void addVisitor() {
    cout << "Enter visitor details:" << endl;
    string visitorName;
    cout << "Name: ";
    getline(cin >> ws, visitorName);

    cout << "Enter student ID to visit: ";
    int studentId;
    cin >> studentId;

    bool studentFound = false;
    for (Room& room : rooms) {
        for (Student& student : room.getOccupants()) {
            if (student.getId() == studentId) {
                if (student.getVisitors().size() < room.getCapacity()) {
                    student.addVisitor(visitorName);
                    cout << "Visitor added successfully for student with ID " << studentId << endl;
                } else {
                    cout << "Room is already full" << endl;
                }
                studentFound = true;
                break;
            }
        }
        if (studentFound) {
            break;
        }
    }

    if (!studentFound) {
        cout << "Student with ID " << studentId << " not found" << endl;
    }
}
```

```
void deleteVisitor() {
    cout << "Enter student ID to delete visitor: ";
    int studentId;
    cin >> studentId;

    bool studentFound = false;
    for (Room& room : rooms) {
        for (Student& student : room.getOccupants()) {
            if (student.getId() == studentId) {
                if (!student.getVisitors().empty()) {
                    student.deleteVisitor();
                    cout << "Visitor deleted successfully" << endl;
                } else {
                    cout << "No visitors found for student with ID " << studentId << endl;
                }
                studentFound = true;
                break;
            }
        }
        if (studentFound) {
            break;
        }
    }

    if (!studentFound) {
        cout << "Student with ID " << studentId << " not found" << endl;
    }
}

void displayMenu() {
    cout << "Hostel Management System" << endl;
    cout << "------------------------" << endl;
    cout << "1. Add Student" << endl;
    cout << "2. Delete Student" << endl;
    cout << "3. Show Room Availability" << endl;
    cout << "4. Show Occupied Rooms" << endl;
    cout << "5. Modify Student Details" << endl;
    cout << "6. Add Visitor" << endl;
    cout << "7. Delete Visitor" << endl;
    cout << "8. Exit" << endl;
    cout << "Enter your choice: ";
}
```

```
public:
    void run() {
        rooms = {
            Room(101, 1, 2),
            Room(102, 1, 2),
            Room(201, 2, 3),
            Room(202, 2, 3),
            Room(301, 3, 4),
            Room(302, 3, 4)
        };

        int adminId;
        string password;

        cout << "Admin Login" << endl;
        cout << "-----------" << endl;
        cout << "Enter Admin ID: ";
        cin >> adminId;
        cout << "Enter Password: ";
        cin >> password;

        if (authenticateAdmin(adminId, password)) {
            int choice;
            do {
                displayMenu();
                cin >> choice;

                switch (choice) {
                    case 1:
                        addStudent();
                        break;
                    case 2:
                        deleteStudent();
                        break;
                    case 3:
                        showAvailability();
                        break;
                    case 4:
                        showOccupiedRooms();
                        break;
                    case 5:
                        modifyStudent();
                        break;
                    case 6:
                        addVisitor();
                        break;
                   case 7:
                       deleteVisitor();
                       break;
                   case 8:
                       cout << "Exited successfully" << endl;
                       break;
                   default:
                       cout << "Invalid option" << endl;
                }

                cout << endl;
            } while (choice != 9);
        } else {
            cout << "Authentication failed. Exiting program." << endl;
        }
    }
};

int main() {
    HostelManagementSystem system;
    system.run();

    return 0;
}
```

9. OUTPUT SCREENSHOT

```
Admin Login
-----------
Enter Admin ID: 123
Enter Password: password
Hostel Management System
------------------------
1. Add Student
2. Delete Student
3. Show Room Availability
4. Show Occupied Rooms
5. Modify Student Details
6. Add Visitor
7. Delete Visitor
8. Exit
Enter your choice: 1
Enter student details:
Name: Ravi
ID: 227654
Gender: male
Course: Bsc
Father's Name: Ramesh
Mother's Name: kalpana
Address: 1\2main street,vellore
Phone Number: 9876543216
Enter room number: 101
Student added successfully to room 101

Hostel Management System
------------------------
1. Add Student
2. Delete Student
3. Show Room Availability
4. Show Occupied Rooms
5. Modify Student Details
6. Add Visitor
7. Delete Visitor
8. Exit
Enter your choice: 4
Occupied Rooms:
---------------
Room 101, Floor 1:
  - Ravi (ID: 227654)
```

```
Hostel Management System
-----------------------
1. Add Student
2. Delete Student
3. Show Room Availability
4. Show Occupied Rooms
5. Modify Student Details
6. Add Visitor
7. Delete Visitor
8. Exit
Enter your choice: 3
Room Availability:
-----------------
Room 101, Floor 1: 1/2 beds available
Room 102, Floor 1: 2/2 beds available
Room 201, Floor 2: 3/3 beds available
Room 202, Floor 2: 3/3 beds available
Room 301, Floor 3: 4/4 beds available
Room 302, Floor 3: 4/4 beds available
```

```
Hostel Management System
-----------------------
1. Add Student
2. Delete Student
3. Show Room Availability
4. Show Occupied Rooms
5. Modify Student Details
6. Add Visitor
7. Delete Visitor
8. Exit
Enter your choice: 6
Enter visitor details:
Name: Kannan
Enter student ID to visit: 227654
Visitor added successfully for student with ID 227654

Hostel Management System
-----------------------
1. Add Student
2. Delete Student
3. Show Room Availability
4. Show Occupied Rooms
5. Modify Student Details
6. Add Visitor
7. Delete Visitor
8. Exit
Enter your choice: 7
Enter student ID to delete visitor: 227654
No visitors found for student with ID 227654
```

```
Hostel Management System
------------------------
1. Add Student
2. Delete Student
3. Show Room Availability
4. Show Occupied Rooms
5. Modify Student Details
6. Add Visitor
7. Delete Visitor
8. Exit
Enter your choice: 5
Enter student ID to modify: 227654
Enter new details for student with ID 227654:
Name: Ravi
Gender: male
Course: BCA
Father's Name: Ramesh
Mother's Name: kalpana
Address: 1/2,kamaraja street,vellore
Phone Number: 9876453215
Student with ID 227654 modified successfully
```

```
Hostel Management System
------------------------
1. Add Student
2. Delete Student
3. Show Room Availability
4. Show Occupied Rooms
5. Modify Student Details
6. Add Visitor
7. Delete Visitor
8. Exit
Enter your choice: 2
Enter student ID to delete: 227654
Student with ID 227654 deleted successfully

Hostel Management System
------------------------
1. Add Student
2. Delete Student
3. Show Room Availability
4. Show Occupied Rooms
5. Modify Student Details
6. Add Visitor
7. Delete Visitor
8. Exit
Enter your choice: 8
Exited successfully
```

10. CONCLUSION & FUTURE WORK

* In conclusion, the provided code implements a basic Hostel Management System using classes and vectors in C++. It allows an administrator to perform various operations such as adding students, deleting students, modifying student details, managing room occupancy, and handling visitor information.

*The current implementation provides the core functionality of the hostel management system, allowing for basic management of student records and room occupancy. However, there are some areas for improvement and future works:

1. **Data Persistence**: Currently, the data is stored only in memory, which means that once the program exits, all the data is lost. Implementing data persistence, such as saving the data to a file or a database, would be beneficial for long-term data storage and retrieval.

2. **Input Validation:** The code does not include input validation for user inputs, such as checking for valid IDs, room numbers, or ensuring that fields are not left empty. Adding input validation would improve the robustness of the system and prevent unexpected errors.

3. **Search and Sorting Functionality:** Adding functionality to search for students based on specific criteria (e.g., name, ID, course) or sorting the list of students by various parameters (e.g., name, ID) would enhance the usability of the system and make it easier to find and manage student records.

4. **Error Handling:** Currently, the code does not handle exceptional scenarios, such as when a room is not found, a student is not found, or an operation fails. Implementing appropriate error handling mechanisms, such as exception handling or error codes, would improve the reliability and user experience of the system.

5. **User Interface:** The current implementation uses a command-line interface. Enhancing the user interface by developing a graphical user interface (GUI) or a web-based interface would make the system more user-friendly and visually appealing.

6. **Additional Features**: Depending on the specific requirements of the hostel management system, additional features can be implemented. Some possible features include generating reports, tracking attendance, managing maintenance requests, or integrating with a payment system for fee collection.

*By addressing these areas for improvement and considering additional features, the Hostel Management System can be further enhanced to meet the specific needs of a hostel or similar accommodation facility.

11.WORK BREAK DOWN

Team Members Registration Numbers	NAMES	Components Worked
22BCS0016	**SHARAN.M**	**ROOMS MODULE,ADMIN LOGIN MODULE**
22BCS0185	**SUSEENDHARRAM.S**	**VISITORS MODULE**
22BCS0167	**DHANUSH.P**	**ROOMS MODULE**
22BCS0142	**JEEVA.S**	**STUDENT MODULE**

12.REVIEW EVALUATION

COMPONENT	MARKS	MEMBER 1	MEMBER 2	MEMBER 3	MEMBER 4	MEMBER 5
USED PRINCIPLES						
DEVELOPED LOGIC						
FUNCTIONALITIES						
CODE EXECUTION						
PRESENTATION						
REPORT						
TOTAL	100 MARKS					