

B Tech Computer Science and Engineering (Data Science)

-311 Program Semester – II

Object Oriented Programming and Design

Mini Project

Hostel Management System

Developed By

L002 Niharika Amritkar

L005 Mishree Bagdai

L009 Vishwaja Bute

L015 Rishika Gade

L019 Kaunchi Jain

Faculty In charge

Dr. Dhirendra Mishra

SVKM's NMIMS

Mukesh Patel School of Technology Management and Engineering

Vile Parle West Mumbai-56

Contents

Chapter No.	Topics	Page Numbers
1	Project Background	3
2	Societal need of this project	4
3	Overall Block diagram of functionalities	5
4	Problem Statement	6
5	List of classes and its Members planned	7
6	Use Case Diagram	8
7	Class Diagram	9
8	Activity Diagram	10
9	Sequence Diagram	15
10	Object Oriented concepts[Encapsulation, Inheritance, Polymorphism, Friend class/function etc.] to be implemented in the code with justification	17
11	Algorithms	25
12	Flowcharts	30
13	Code written in C++	37
14	Output screenshots of all functions	59
15	List of Errors obtained and its resolution.	69
16	Scope of improvements [plans to improve it]	70
17	References/List of research papers, books, white papers referred. (in IEEE format)	71

1. Project Background

With fair changes in modern management techniques, there has been a major need of simplicity and sustainability in large systems with complicated and a large amount of data. This idea is what we've focused on while deciding on this project topic. The objective here is to implement the newly learnt object-oriented programming concepts.

A few necessary pre requisites of the project in terms of knowledge include:

- Knowing syntax and implementations of basic concepts like algorithms, flowcharts, loops, if else statements, functions, operators, arrays and pointers.
- Understanding object-oriented concepts like class, polymorphism, inheritance.
- Being aware about the indentations, neatness, accuracy and precision of the combined code and uml diagrams.
- Knowing exactly what result is gained by what logic.

Now, the storyline of the project involves: A student can login or register. Once the student is registered, they can access the portal and various functionalities of the same. On the output screen, the basic available functionalities will be displayed. Hostel guests will be able to enter entry/exit data, with the results being forwarded to administration or the child's parents with the help of external agents . Students and administrators will be able to see information on meals and laundry, as well as submit laundry items and view the meal menu. There are also display functions for terms and conditions and refund/ cancellation. Open ends in the code are tied up properly using error keywords to avoid user confusion or misdirection.

2. Societal Need of this Project

A management system is designed to identify and manage risks—safety, environmental, quality, business continuity, food safety (and many others)—through an organized set of policies, procedures, practices, and resources that guide the enterprise and its activities to maximize business value. To handle big data in a proper way we use management systems for feasible procedure completion.

Most of the common advantages of such management systems are include factors like greater efficiency and less waste, better and consistent control of major business processes, a better understanding of customer needs, regulation of successful working practices, improved risk management, increased customer satisfaction, improved participation of employees.

In this project we have focused on a hostel’s management system which helps the admin determine its workings and students understand the rules and facilities they’re provided with.

This project aims to use resources efficiently and improve financial performance on a small startup scale and also induce measured risk management. It’ll help the environment in general to functions in a ‘known’ manner rather than chaos in case of disagreements or indecisions. This being a mini project hinders and limits the amount of “complicated” that can be added to refine the code to make a better system but still manages to qualify the outline of the idea properly.

3. Overall Block Diagram of Functionalities

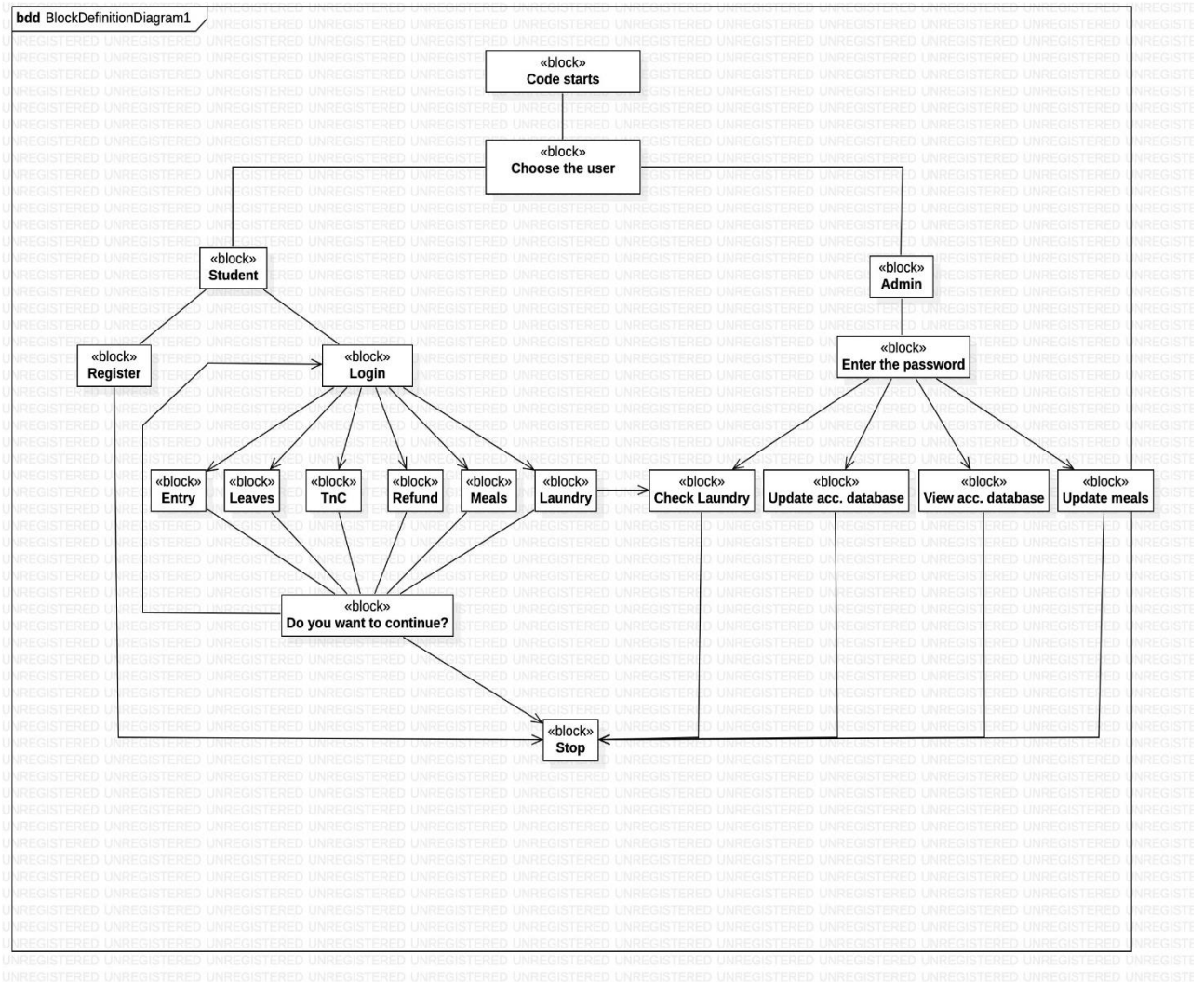


Figure 3.1: Block Diagram of Hostel Management System

A block diagram is a diagram of a system in which the principal parts or functions are represented by blocks connected by lines that show the relationships of the blocks. We have used this to simplify the complex structure of code into blocks of proper flow of the code.

4. Problem Statement

“To create a hostel management system on a small scale with its various aspects of entry/exit terms, laundry, meal information, student data, room data, permissions while keeping in mind object oriented programming concepts such as polymorphism, encapsulation, inheritance, abstraction and other basic concepts of if else, switch case, functions.”

5. List of Classes and its Members Planned

The following table represents the names of all the classes and data members as well as member functions giving a proper view of the layout of components.

Figure 5.1: Classes and Members used in the code

Class	Data Members	Member Functions
Login	(string) login_register, identity	login login_or_reg login_con
Admin	(char) Id, status, contact (int) room_num (string) fname, lname, hometown, type, lunch, dinner, day	view_accomodation_database update_accomodation_database TNC meals laundry_cleaning
Student	(char) date_registration, date_ admission, contact_email, payment_status (int) room, age, leaves (double) contact_num (string) first_name, last_name	login_ reg refund laundry_cleaning meals entry

6. Use Case Diagram

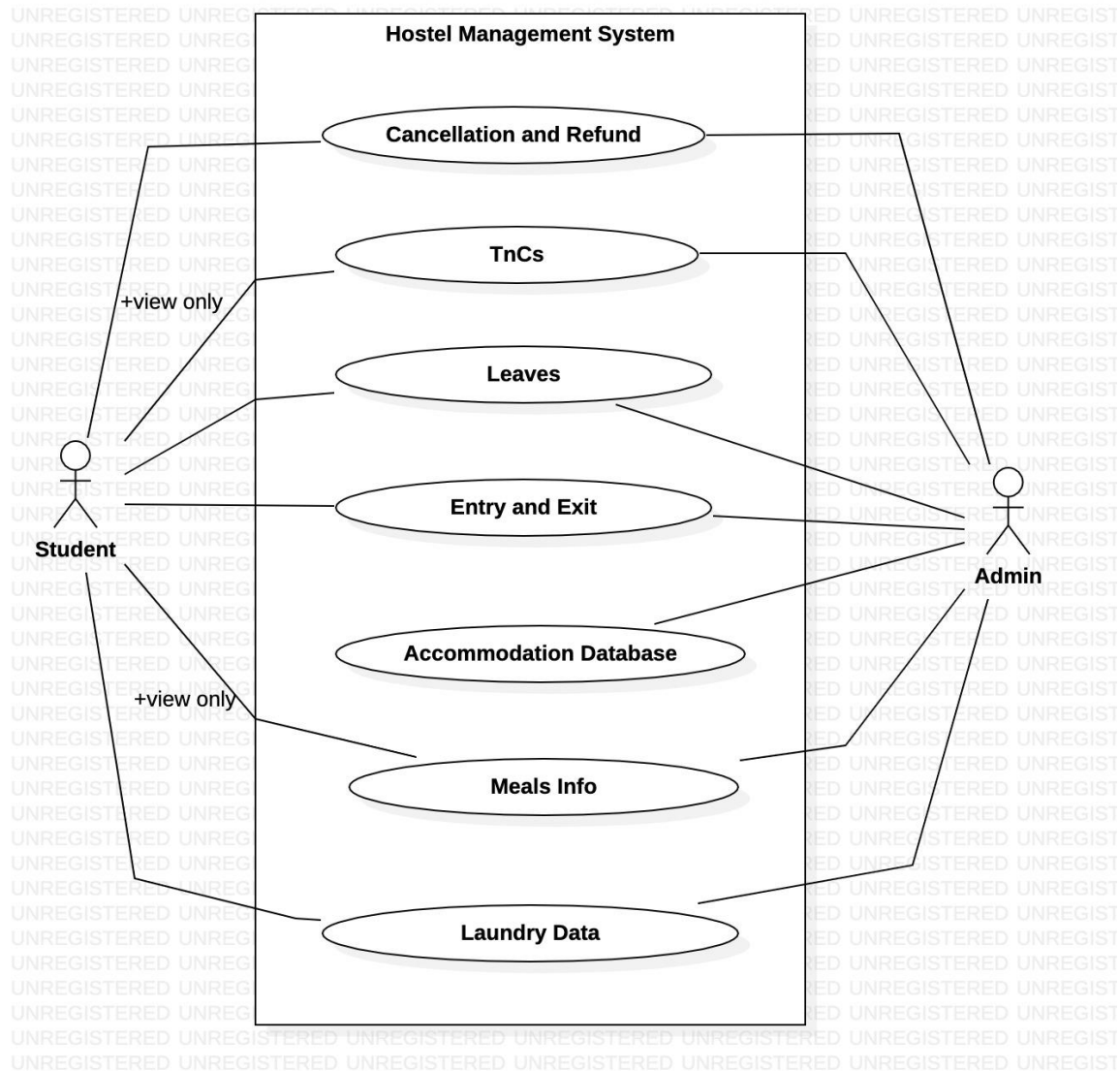


Figure 6.1: Use Case Diagram following the system code

Use-case diagrams describe the high-level functions and scope of a system. These diagrams also identify the interactions between the system and its actors which are the student and admin here.

7. Class Diagram

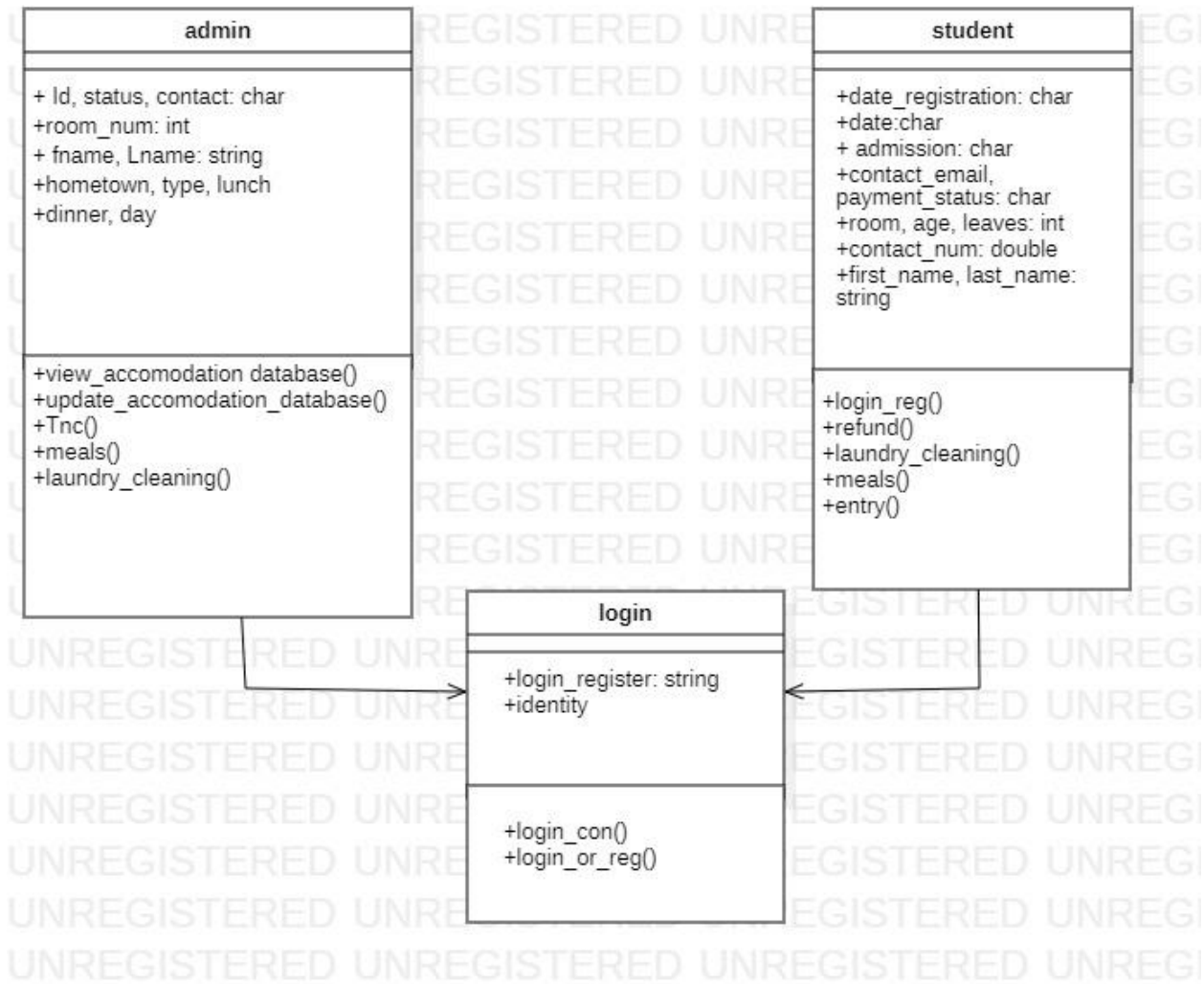


Figure 7.1: Class Diagram with respect to classes and members of the code

Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. We have used it to help in the modeling of object oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages.

8. Activity Diagram

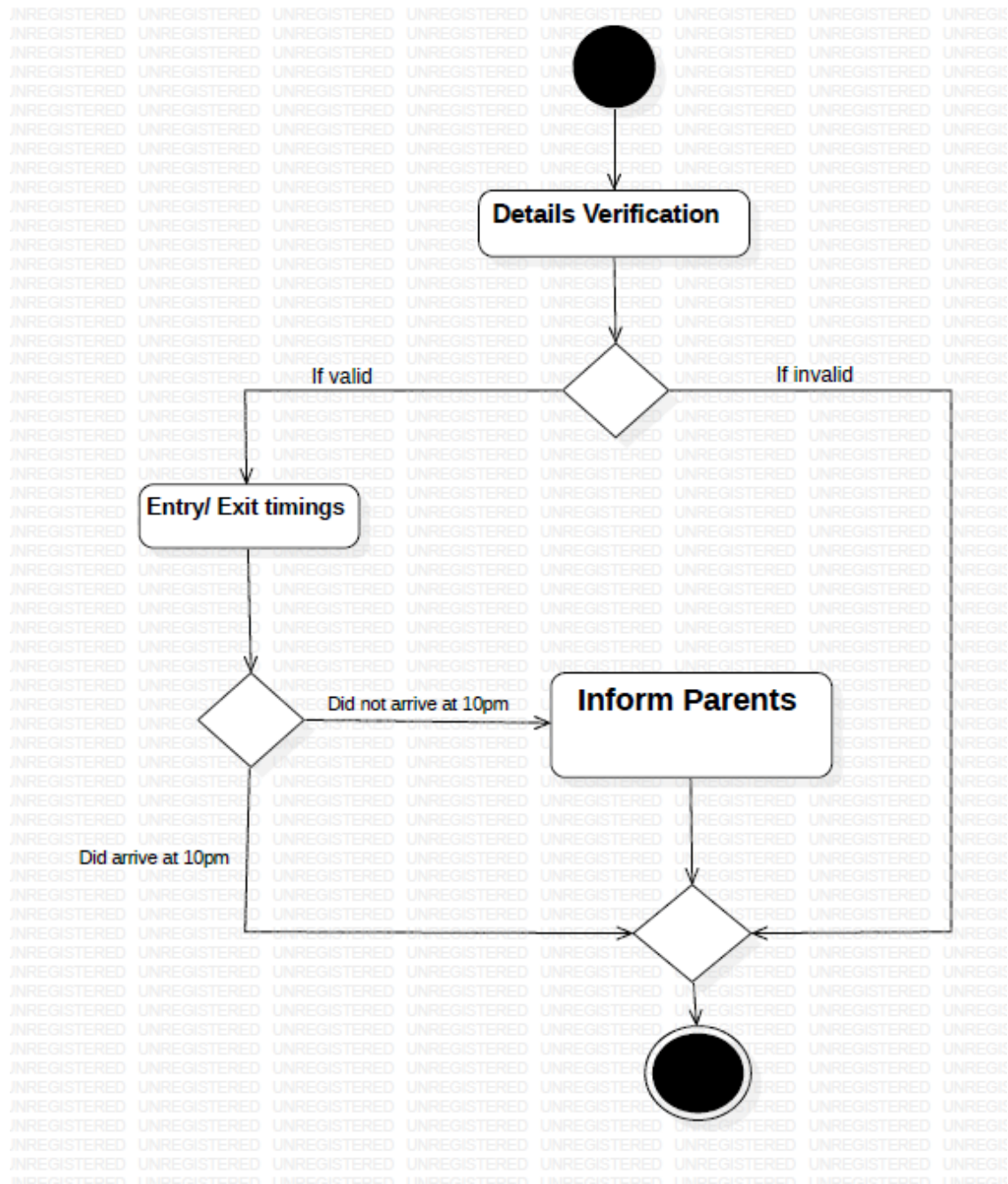


Figure 8.1: Entry/Exit Module

Activity Diagram

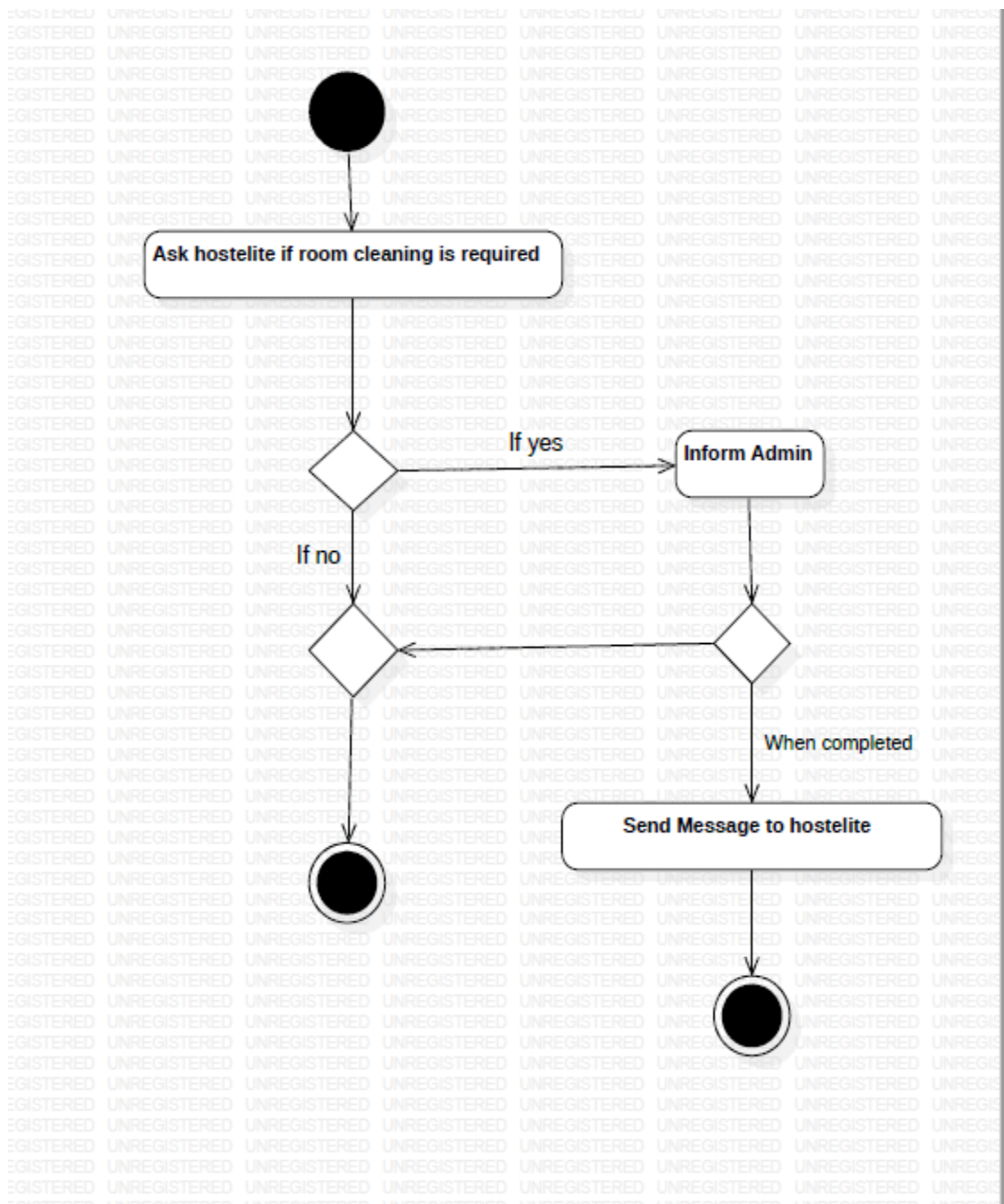


Figure 8.2: Room Cleaning Module

Activity Diagram

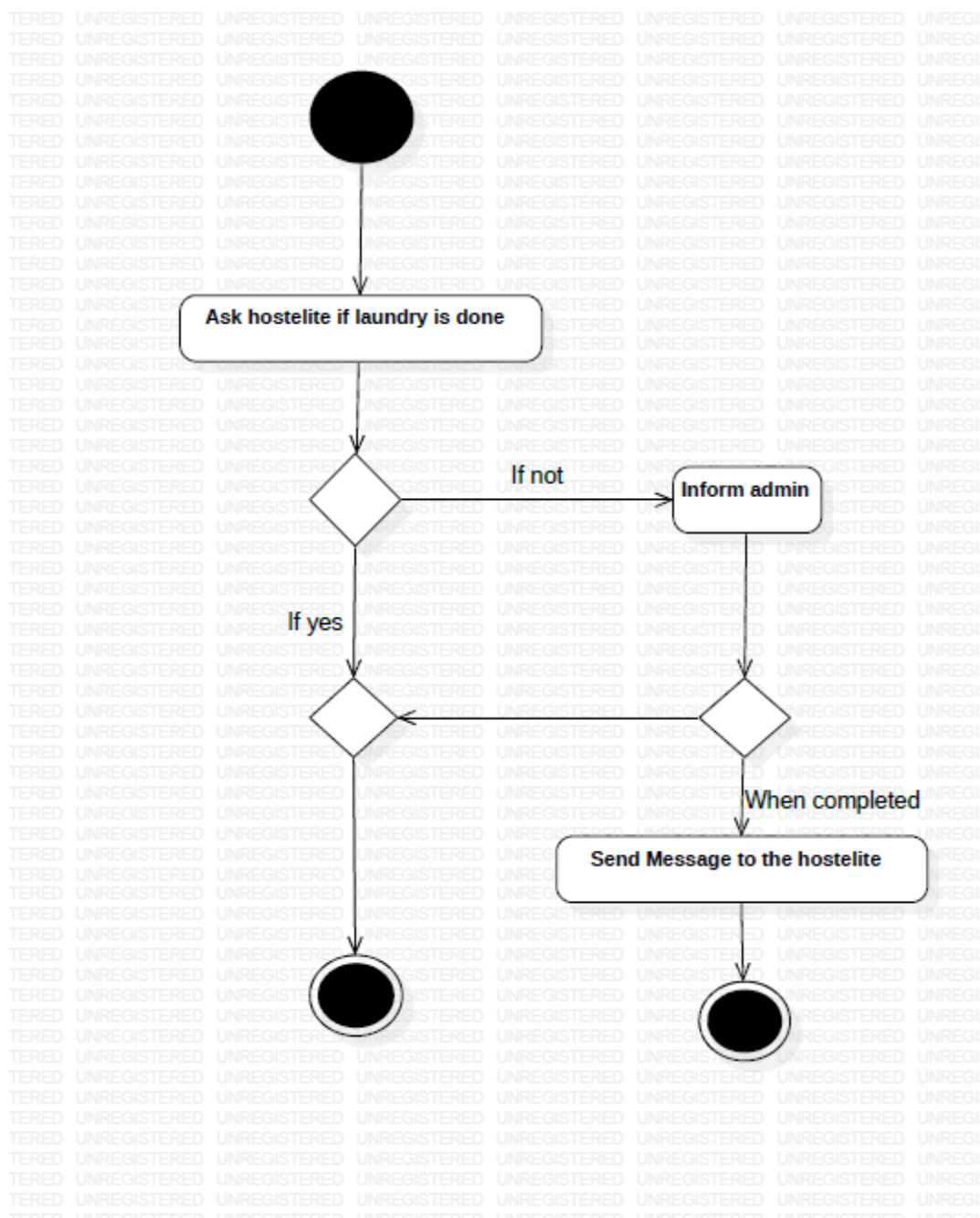


Figure 8.3: Laundry Module

Activity Diagram

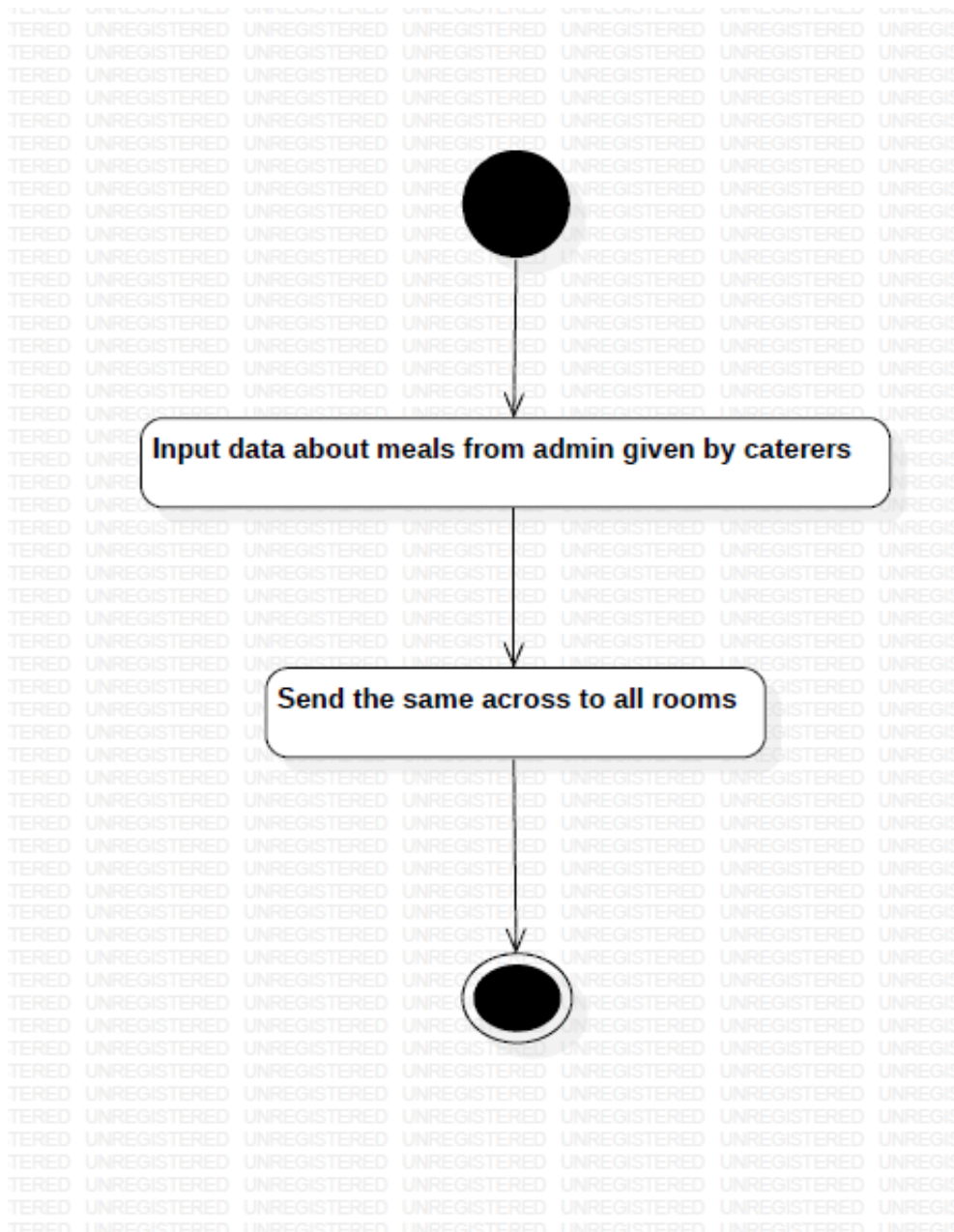


Figure 8.4: Meals Information Module

Activity Diagram

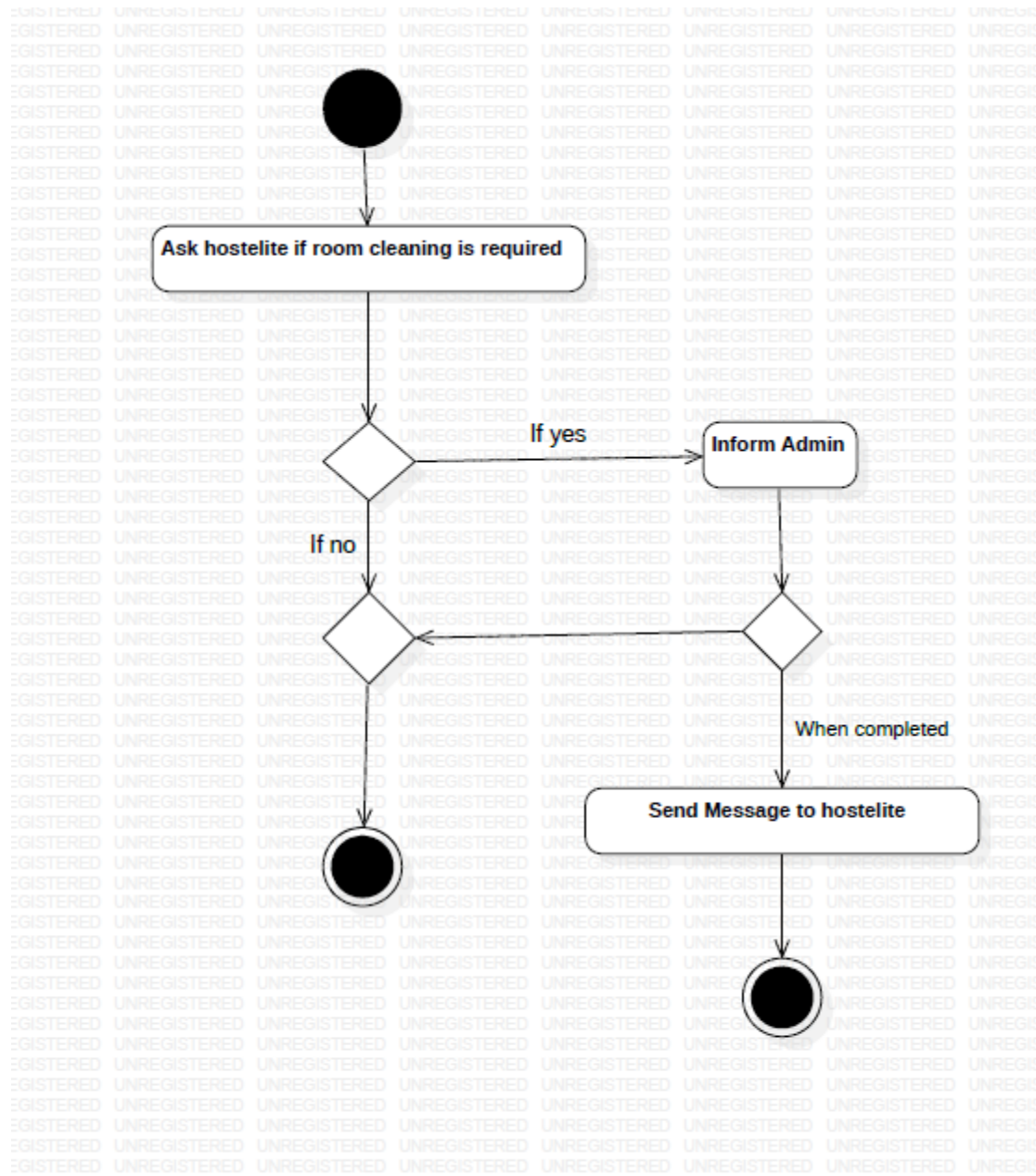


Figure 8.5: Entry/Exit Module

9. Sequence Diagram

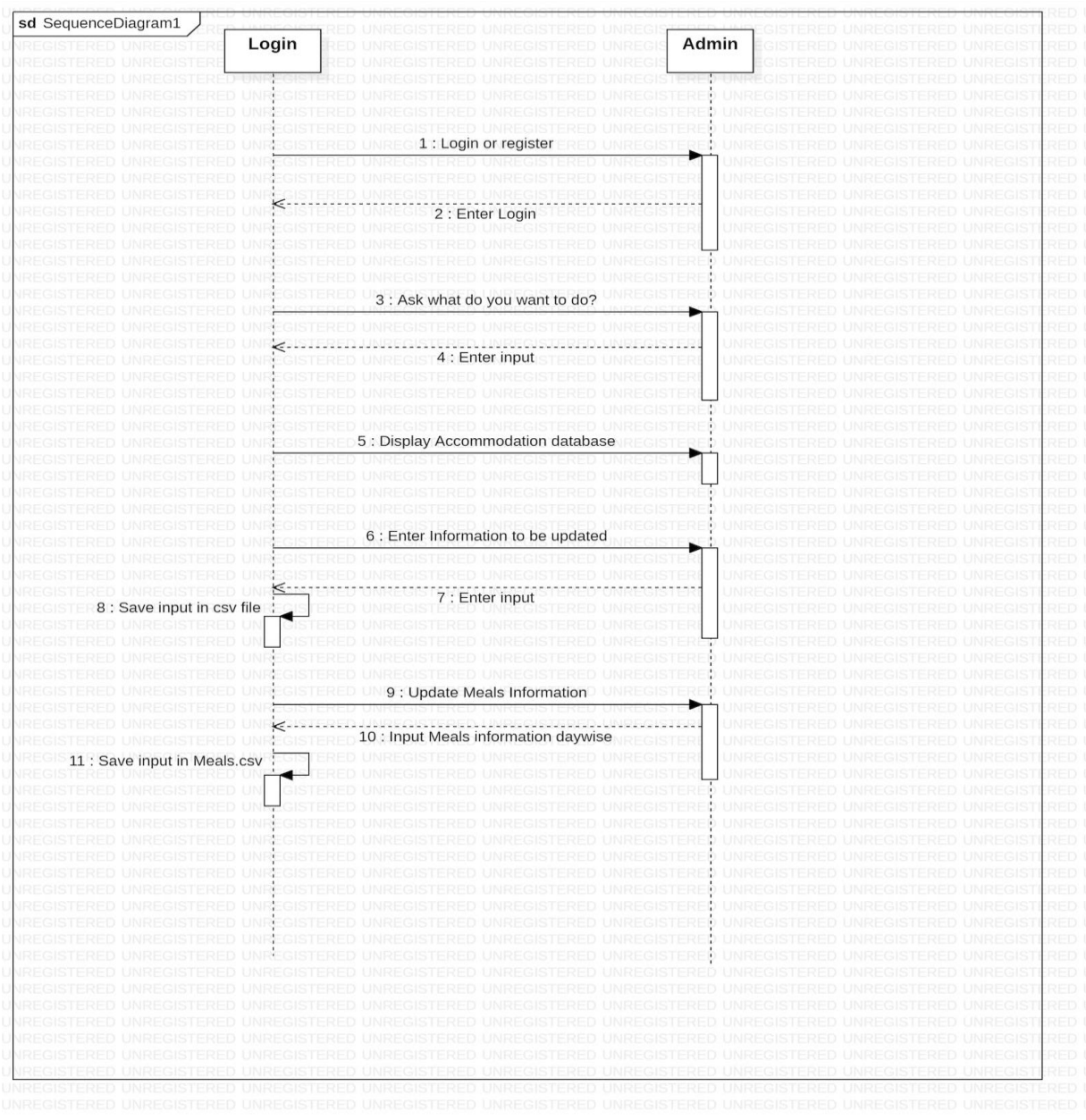


Figure 8.4: Sequence diagram depicting the login and admin sequence with fair highlight to the constituents of the code

9. Sequence Diagram

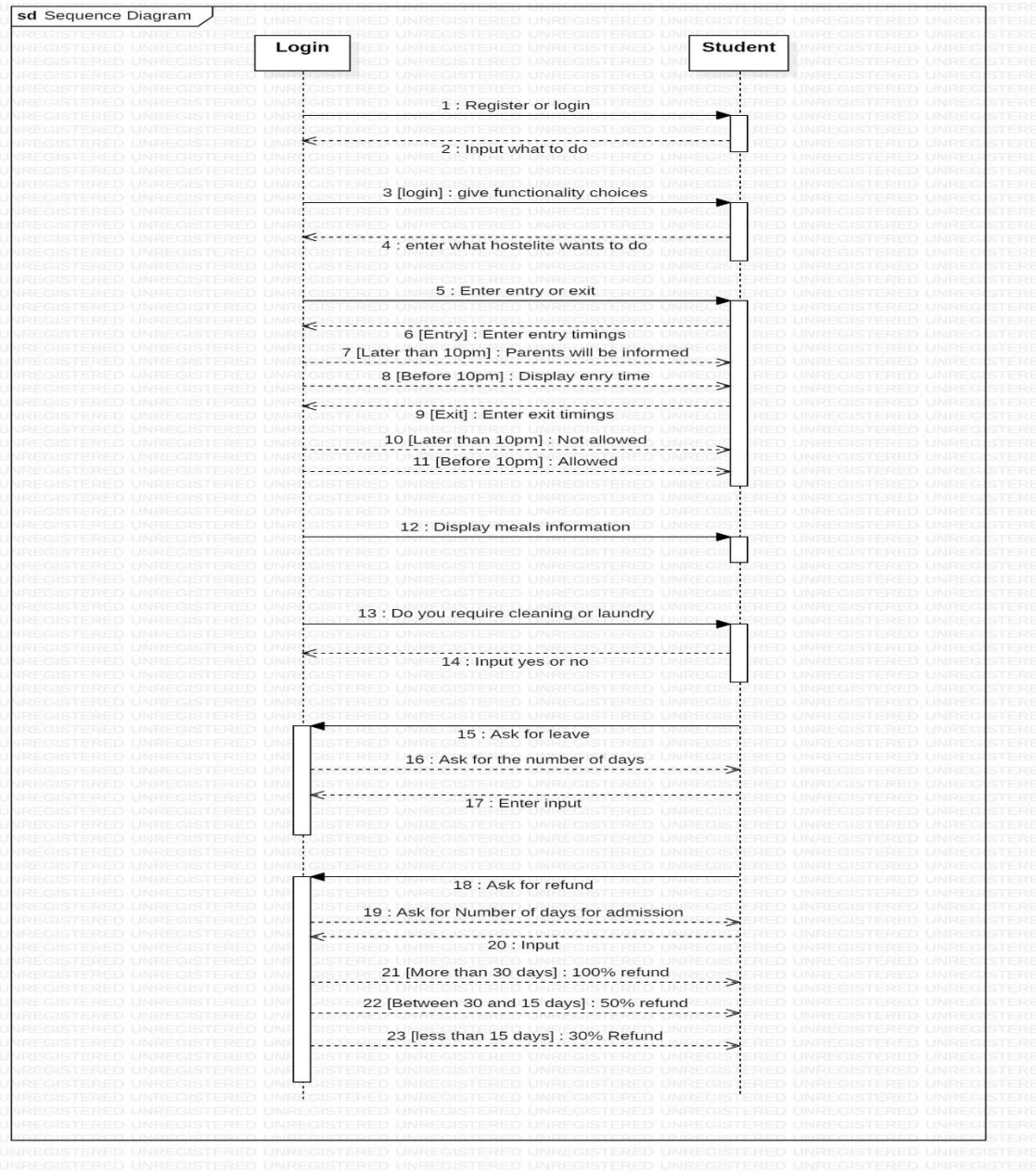


Figure 8.4: Sequence diagram depicting the login and student sequence with fair highlight to the constituents of the code

10. Object Oriented Concepts to be implemented in the Code

Concepts used:

10.1 Encapsulation

10.2 Class

10.3 Object

10.4 Abstraction: Private and Public access modifiers

10.5 Polymorphism: Function Overloading

10.6 Inheritance: Multiple

10.7 File Handling: .txt and .csv

10.8 Console I/O: unformatted

10.9 Scope Resolution

10.10 Constructors and Destructors

10.11 Passing of objects as input arguments

10.1 Encapsulation

Encapsulation is one of the key features of object-oriented programming. It involves the bundling of data members and functions inside a single class.

Bundling similar data members and functions inside a class together also helps in data hiding.

In C++, encapsulation helps us keep related data and functions together, which makes our code cleaner and easy to read.

It helps to control the modification of our data members.

It helps to decouple components of a system. For example, we can encapsulate code into multiple bundles. These decoupled components (bundles) can be developed, tested, and debugged independently and concurrently. And any changes in a particular component do not have any effect on other components.

Data hiding is a way of restricting the access of our data members by hiding the implementation details. Encapsulation also provides a way for data hiding. We can use access modifiers to achieve data hiding in C++. Making the variables private allowed us to restrict unauthorized access from outside the class. This is data hiding.

10.2 Class

A class is used to specify the form of an object and it combines data representation and methods for manipulating that data into one neat package. The data and functions within a class are called members of the class.

When you define a class, you define a blueprint for a data type. This doesn't actually define any data, but it does define what the class name means, that is, what an object of the class will consist of and what operations can be performed on such an object.

A class definition starts with the keyword class followed by the class name; and the class body, enclosed by a pair of curly braces. A class definition must be followed either by a semicolon or a list of declarations.

10.3 Object

An Object is an instance of a Class. When a class is defined, no memory is allocated but when it is instantiated (i.e. an object is created) memory is allocated.

When a class is defined, only the specification for the object is defined; no memory or storage is allocated. To use the data and access functions defined in the class, you need to create objects.

10.4 Abstraction

Data abstraction is one of the most essential and important feature of object oriented programming in C++. Abstraction means displaying only essential information and hiding the details. Data abstraction refers to providing only essential information about the data to the outside world, hiding the background details or implementation.

There are two types of abstraction in the C++ language namely: Control abstraction - The details of abstraction implementation will always be hidden under control and will not be visible. Data abstraction - The details about the data in the program is always concealed in data abstraction.

10.5 Polymorphism

Polymorphism is an important concept of object-oriented programming. It simply means more than one form. That is, the same entity (function or operator) behaves differently in different scenarios.

We can define polymorphism as the ability of a message to be displayed in more than one form. A real-life example of polymorphism, a person at the same time can have different characteristics. Like a man at the same time is a father, a husband, an employee.

Typically, polymorphism occurs when there is a hierarchy of classes and they are related by inheritance. C++ polymorphism means that a call to a member function will cause a different function to be executed depending on the type of object that invokes the function.

We can implement polymorphism in C++ using the following ways:

1. Function overloading
2. Operator overloading
3. Function overriding
4. Virtual functions

10.6 Inheritance

It is a mechanism of reusing and extending existing classes without modifying them, thus producing hierarchical relationships between them.

Inheritance is almost like embedding an object into a class. Suppose that you declare an object x of class A in the class definition of B. As a result, class B will have access to all the public data members and member functions of class A. However, in class B, you have to access the data members and member functions of class A through object x.

Inheritance lets you include the names and definitions of another class's members as part of a new class. The class whose members you want to include in your new class is called a base class. Your new class is derived from the base class. The new class contains a subobject of the type of the base class.

Types of Inheritance

- 1. Single Inheritance**
- 2. Multiple Inheritance**
- 3. Multilevel Inheritance**
- 4. Hierarchical Inheritance**
- 5. Hybrid Inheritance**

10.7 File Handling

Files are used to store data in a storage device permanently. File handling provides a mechanism to store the output of a program in a file and to perform various operations on it.

A stream is an abstraction that represents a device on which operations of input and output are performed. A stream can be represented as a source or destination of characters of indefinite length depending on its usage.

In C++ we have a set of file handling methods. These include ifstream, ofstream, and fstream. These classes are derived from fstreambase and from the corresponding istream class. These classes, designed to manage the disk files, are declared in fstream and therefore we must include fstream and therefore we must include this file in any program that uses files.

In C++, files are mainly dealt by using three classes fstream, ifstream, ofstream.

- ofstream: This Stream class signifies the output file stream and is applied to create files for writing information to files**
- ifstream: This Stream class signifies the input file stream and is applied for reading information from files**
- fstream: This Stream class can be used for both read and write from/to files.**

All the above three classes are derived from `fstreambase` and from the corresponding `iostream` class and they are designed specifically to manage disk files.

C++ provides us with the following operations in File Handling:

- Creating a file: `open()`
- Reading data: `read()`
- Writing new data: `write()`
- Closing a file: `close()`

10.8 Console I/O

The I/O system in C++ is designed to work with a wide variety of devices including terminals, disks, and tape drives. Although each device is very different, the I/O system supplies an interface to the programmer that is independent of the actual device being accessed. This interface is known as the stream.

A stream is a sequence of bytes.

- The source stream that provides the data to the program is called the input stream.
- The destination stream that receives output from the program is called the output stream.
- The data in the input stream can come from the keyboard or any other input device.
- The data in the output stream can go to the screen or any other output device.

C++ contains several pre-defined streams that are automatically opened when a program begins its execution. These include `cin` and `cout`. It is known that `cin` represents the input stream connected to the standard input device (usually the

keyboard) and cout represents the output stream connected to the standard output device (usually the screen).

The C++ I/O system contains a hierarchy of classes that are used to define various streams to deal with both the console and disk files. These classes are called stream classes. The hierarchy of stream classes used for input and output operations is with the console unit. These classes are declared in the header file iostream. This file should be included in all the programs that communicate with the console unit.

10.9 Scope Resolution

The scope resolution operator is used to reference the global variable or member function that is out of scope. Therefore, we use the scope resolution operator to access the hidden variable or function of a program. The operator is represented as the double colon (::) symbol.

- It is used to access the hidden variables or member functions of a program.
- It defines the member function outside of the class using the scope resolution.
- It is used to access the static variable and static function of a class.
- The scope resolution operator is used to override function in the Inheritance.

10.10 Constructors and Destructors

Constructor in C++ is a special member function of a class whose task is to initialize the object of the class. A destructor is also a member function of a class that is instantaneously called whenever an object is destroyed.

Constructor in C++ is a special member function of a class whose task is to initialize the object of the class, it's special because it has the same name as that of the class. It is called a constructor because it constructs the value of data members at the time of

object initialization. The compiler invokes the constructor whenever an object is created. Since a constructor defines the value to a data member, it has no return type.

10.11 Passing of objects as input arguments

To pass an object as an argument we write the object name as the argument while calling the function the same way we do it for other variables. Syntax:

function_name(object_name); Example: In this Example there is a class which has an integer variable 'a' and a function 'add' which takes an object as argument.

11. Algorithms

1. Login

Step 1: Start the program

Step 2: Select login or register

Step 3: If login, select hostelite or admin

Step 4: If hostelite, select any of the 6 options to continue. Step 5:- if admin, enter the password

Step 6: Select any of the 4 options to continue.

Step 7: End program

2. Entry/Exit

Step 1: Start program

Step 2: Select login as the command

Step 3: Select hostelite, when asked are you hostelite or admin

Step 4: On selecting the 1st functionality, student will be displayed the entry/exit page

Step 5: If the student is entering the hostel, using if-else statements the time will be checked to see whether the student can be permitted entry or not

Step 6: If the time exceeds beyond entry time, call will be made to parents

Step 7: If the student is leaving the hostel, exit time will be asked

Step 8: If the exit time is beyond leaving hours of the hostel student will be denied

Step 9: Display message “you are not allowed to leave the hostel as per the rules.”

Step 10: End program

3. Cancellation/refund

Step 1: Start program

Step 2: Select login as the command

Step 3: Select hostelite, when asked are you a hostelite or admin

Step 4: On selecting the 2nd functionality the user will be displayed the cancellation/refund page

Step 5: The user is asked their payment status

Step 6: If the payment is completed, the user presses 1 and if not, 0

Step 7: Upon selecting 1, the user is asked the no. of days left before joining the hostel

Step 8: Using if-else statements, if more than 30 days left, 100% refund granted, in case of 15-30 days left 80% refund granted, in case of less than 15 days 50% refund granted.

Step 9: End program

4. TnC's/ Rules and Regulations

Step 1: Start program

Step 2: Select login as the command

Step 3: Select hostelite, when asked are you a hostelite or admin

Step 4: On selecting the 3rd functionality the user will be displayed the Terms and

Conditions page of the hostel

Step 5: Terms and conditions in detail as directed by the admin.

Step 6: End program

5. Leaves

Step 1: Start program

Step 2: Select login as the command

Step 3: Select hostelite, when asked are you a hostelite or admin

Step 4: On selecting the 4th functionality the user will be displayed the leaves page

Step 5: The user is asked the no. of days for which a leave is required

Step 6: On inputting the days the message will be sent to the admin class from where it will be verified by the parents and displayed back to students withing 24 hours

Step 7: End program

6. Meals info

Step 1: Start program

Step 2: Select 5th functionality

Step 3: The meals menu will be displayed for the entire week

Step 4: End program

7. Laundry

Step 1: Start program

Step 2: Select 6th functionality

Step 3: A csv file is created and user is asked two questions

Step 4: System asks the user if they require laundry & room cleaning

Step 5: The user input is stored in a csv file which can be accessed by the admin

Step 6: End program

8. View accommodation database

Step 1: Start program

Step 2: When 1 is selected under admin, the admin id is asked

Step 3: If the admin id is correct the accommodation database is displayed to the admin

Step 4: End program

Update accommodation database

Step 1: Start program

Step 2: When 2 is selected under admin, a csv file is created and information is appended in it

Step 3: If the admin id is correct, further information can be stored

Step 4: The following information is taken, the room number, first name, last name, room type, place of origin, payment status, and contact number.

Step 5: These data is appended in the csv file and the database is updated

Step 6: End program

9. Check students requirements

Step 1: Start program

Step 2: When 3 is selected under admin, the admin id will be asked

Step 3: A previously created csv file is displayed stating the rooms in which cleaning and laundry is required

Step 4: End program

10. Update meals information

Step 1: Start program

Step 2: Open file meals.txt

Step 3: Enter day, lunch, dinner details

Step 4: Close file

Step 5: End program

12. Flowchart

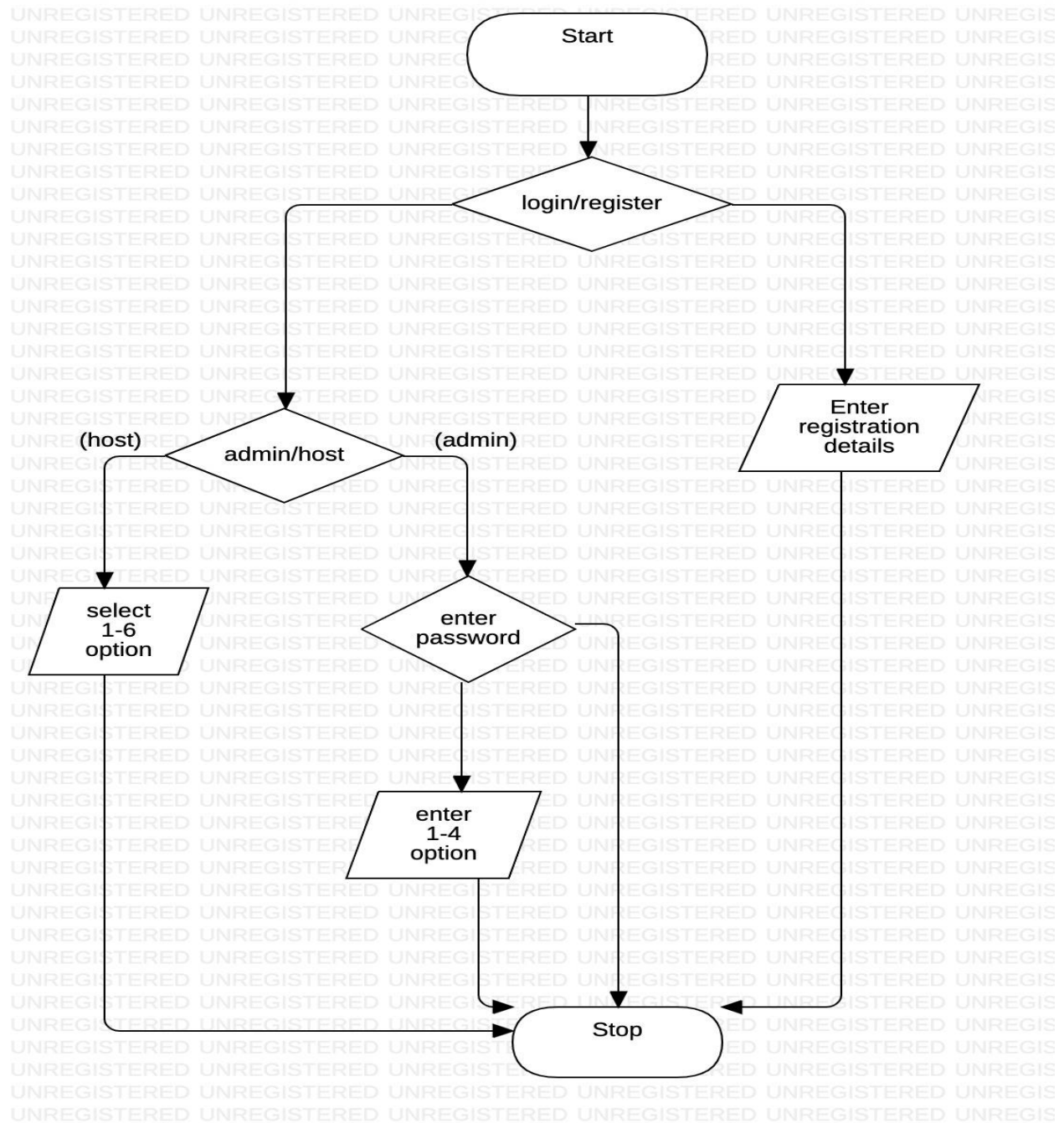


Figure 12.1: Login

Flowchart

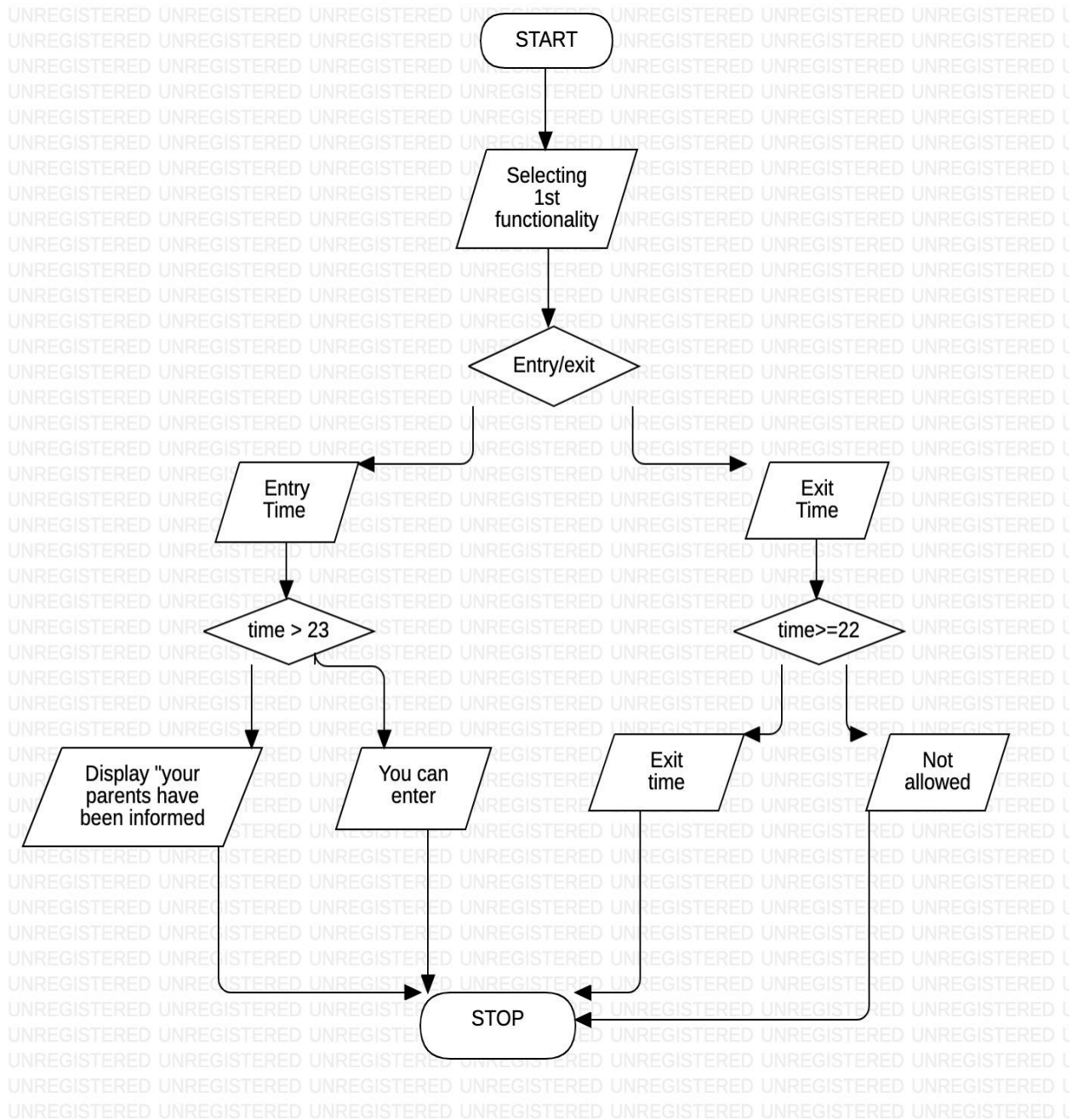


Figure 12.2: Entry/ Exit

Flowchart

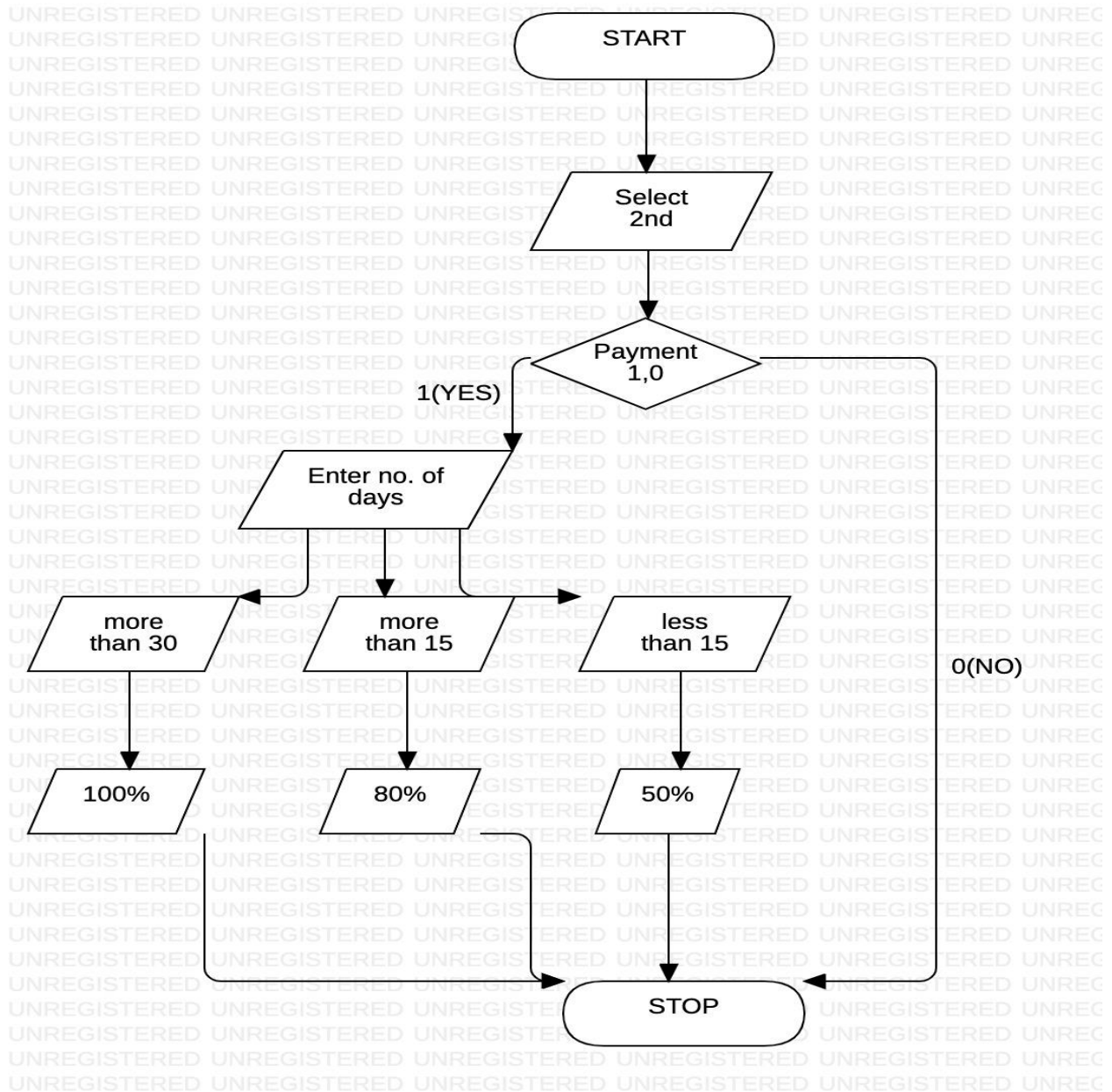


Figure 12.3: Refund/ Cancellation

Flowchart

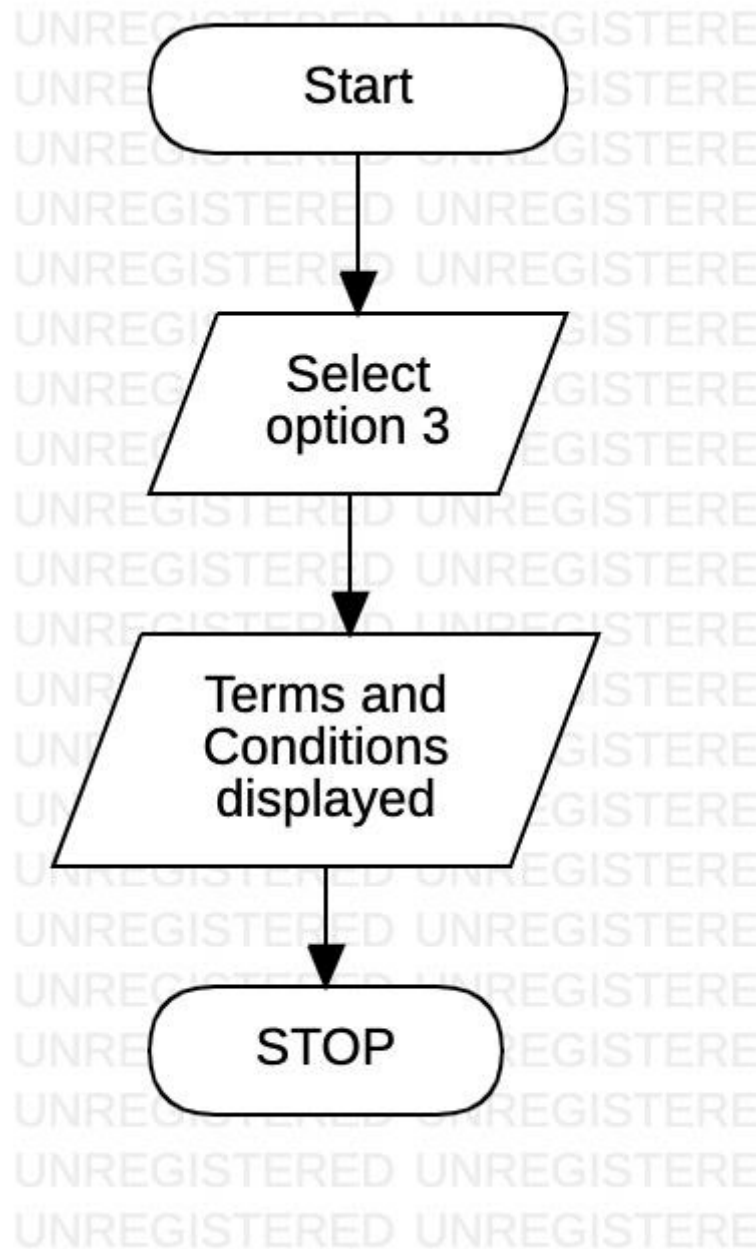


Figure 12.4: TnC's

Flowchart

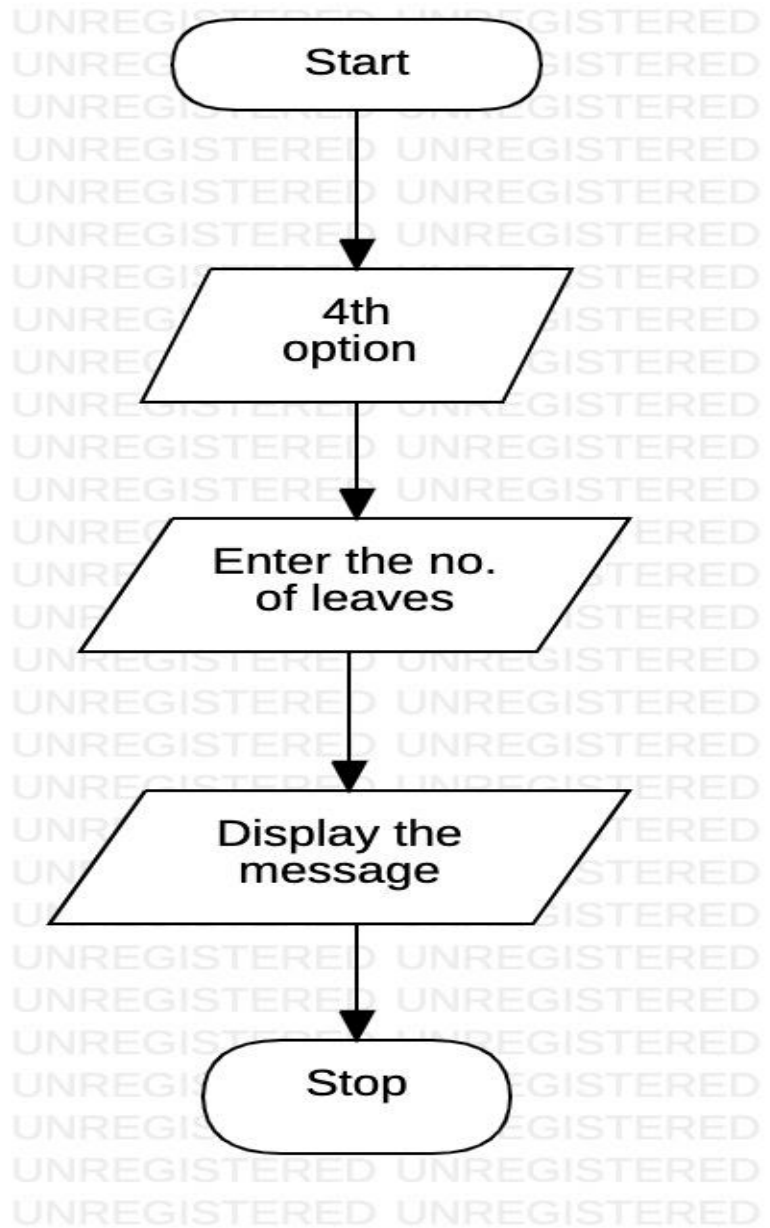


Figure 12.1: Leaves

Flowchart

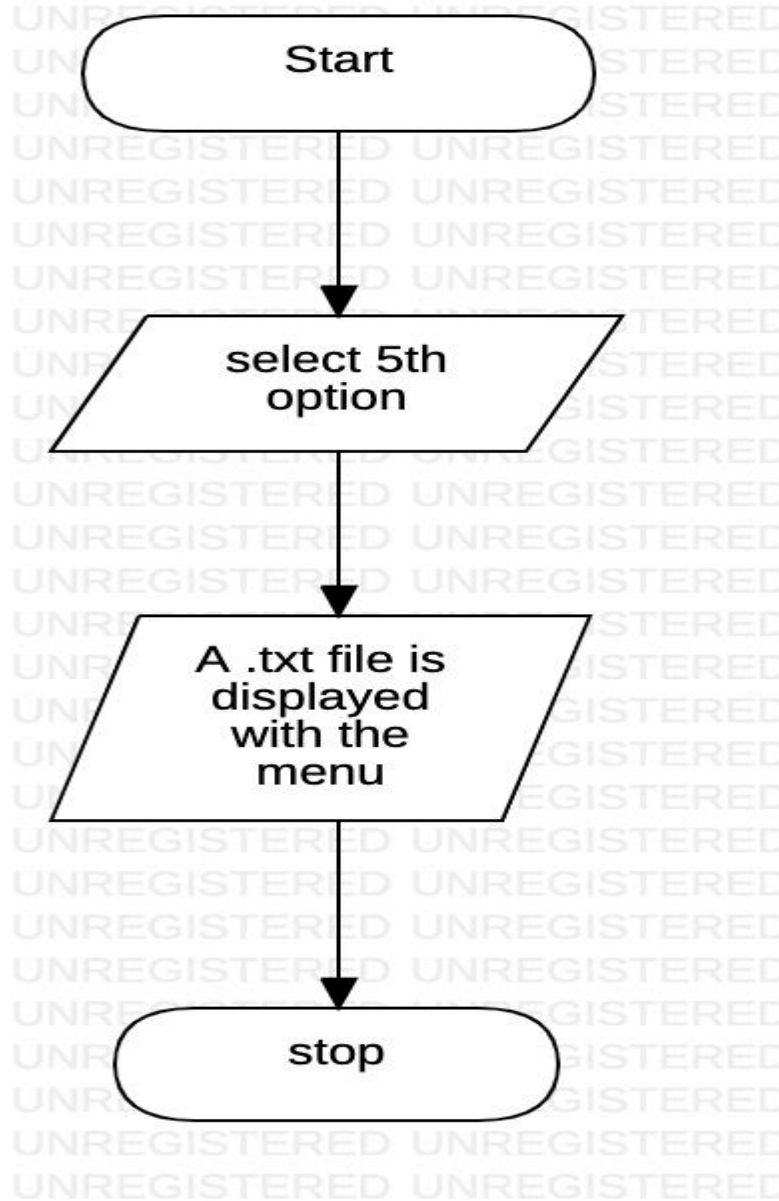


Figure 12.1: Meal Information

Flowchart

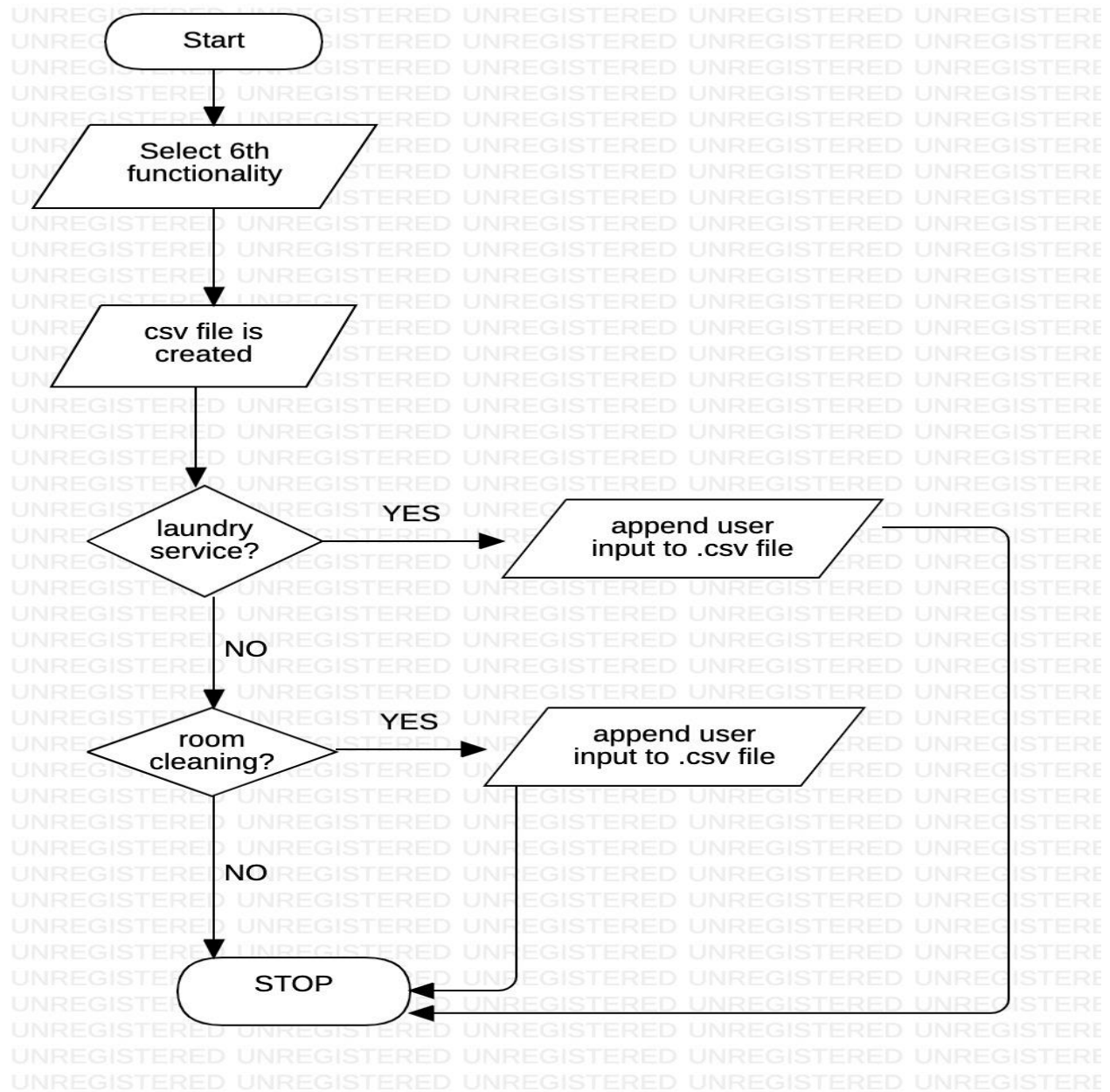


Figure 12.1: Laundry

13. Code written in c++

```
#include<iostream>

//library for input output stream

#include<fstream>

//file handling

#include<stdlib.h>

//for console clearance,to exit the main screen etc.

#include<string>

//to get an input of an array of characters


using namespace std;

class admin;

class student;

fstream file;

//declaring a file object of class fstream from fstream library

ifstream ifile;

//declaring a file object of class ifstream from fstream library


/*

*****

class admin

*****

*/
```

```

class admin
{
    char id;
    int room_num=0;
    string fname, lname, hometown, type;
    string status;
    string lunch, dinner;
    string day;
    char contact[10];
    //declaring data members for functionalities of admin

public:
    //declaring and defining function for viewing data in the existing file
    void view_accommodation_database(admin a)
    {
        cout<<"Enter Admin ID:";
        cin>>id;
        if(id=='z')
        {
            ifstream ifile;
            //opening an excel file named accomodation.csv
            ifile.open("Accommodation.csv");
            cout<<endl<<"Reading from the file"<<endl;
            if(ifile.fail())
            //if the file fails to open, an error message will be declared.
            {

```

```

        cout<<"Input file could not be opened.\n";
        exit(1);
    }
    else

        ifile>>room_num;
        ifile>>fname;
        ifile>>lname;
        ifile>>type;
        ifile>>hometown;
        ifile>>status;
        ifile>>contact;

    //extracting data from the excel file
    cout<<"Room No.|First name|Last Name|Room type|Hometown|Payment
status|Contact|"<<endl;

    cout<<room_num<<"|"<<fname<<"|"<<lname<<"|"<<type<<"|"<<hometown<<"|"<<
status<<"|"<<contact<<"|"<<endl;

    //displaying data in the csv file

    ifile.close();
    //closing the file
}
else
    cout<<"No access."<<endl;
}

```

```

void update_accommodation_database(admin a)
//function for updating accommodation database and taking an instance of admin as
an argument
{
    cout<<"Enter Admin ID:";
    cin>>id;
    if(id=='z')
    //verifying admin and giving access to private data
    {
        ofstream ofile("Accommodation.csv", ios::app);
        //opening accomodation file and appending data in it
        ofile<<" Room No. , First Name , Last Name , Room type , Hometown , Payment
status , Contact number\n";
        cout<<"Enter room number: "; cin>>room_num;
        cout<<"Enter first name: "; cin>>fname;
        cout<<"Enter last name: "; cin>>lname;
        cout<<"Enter room type: "; cin>>type;
        cout<<"Enter place of origin: "; cin>>hometown;
        cout<<"Enter payment status: "; cin>>status;
        cout<<"Enter contact number: "; cin>>contact;

        ofile<<room_num<<","<<fname<<","<<lname<<","<<type<<","<<hometown<<","<<st
atus<<","<<contact;

        //reading the data into the file
        ofile.close();}

```



```
//closing the file object
```

```
else
```

```
cout<<"No access."<<endl;
```

```
}
```

```
void TNC()
```

```
{
```

```
fstream newfile;
```

```
newfile.open("TNC.txt",ios::out);
```

```
// open a file to perform write operation using file object
```

```
if(newfile.is_open())
```

```
//checking whether the file is open
```

```
{
```

```
newfile<<"1.The fees will be collected for entire academic session  
(annually).\n2.Student residing in the hostel premises shall strictly observe all the Rules  
and Regulations in force from time to time.\nBreach of rules/regulations may invite  
rustication/fine.\n3.Smoking, consumption of alcoholic drinks, drugs, spitting are  
strictly prohibited. \nFRIENDS are strictly prohibited in the hostel premises after  
20:00hrs.\nStrict action will be taken against defaulter.\n4.Every student shall be in his  
room by 23:00hrs.\n5. Students will not enter rooms of other students without  
permission of the inmates.\n6. Every case of illness and accident must be reported  
immediately on the Application and to respective Property Manage of  
premises.\n7.Students suffering from any contagious disease will not be allowed to stay  
in the respective Property Manager.\n8. Every student shall keep the room allotted to  
him neat and clean.\nHe/She shall take proper care of the furniture and fixtures handed
```

over to him / her.\n**The Administrator has the right to enter and inspect the rooms at any time.\n8. All matters relating to differences among students and complaints about the servants shall be brought to the notice of the Property Manager, who will take action as may be necessary.\n9. Students must switch off the lights, fans AC and Bathroom Geyser in their rooms every time they go out and take precautions to economize electricity consumption.\nAir-condition can be used 24x7 but it must be switched off if there is no one in the room.\n10. Charges for any damages to the property as well as to the furniture and fixtures caused by student/students negligence will be recovered from the student staying in the said premises.\n11. Student should not drive nails, screws etc. into the wall or doors. No repair shall be done by the students themselves.\nThey should raise a ticket on hostel's Application or approach the respective Property Manager who will arrange for repairs.\n12. Visitors are not allowed to enter any room.\nThey have to sit at the common area.\n13. All the facilities including T.V., Magazines, Newspaper, Internet etc., if misused, shall be discontinued without given any notice and disciplinary action will be taken against the students involved.\n14. Before leaving the PG Accommodation, a student must pay all dues and hand over the charges of rooms and other material in satisfactory condition to the\nrespective Property Manager of the premises.\n15. If any student is found misbehaving and misconducting himself, he/she will be expelled from the Accommodation immediately and\nthe deposit paid by him / her will be forfeited.\n16. Permission letter for night outs from parents should be mailed at XYZhostel@gmail.com and approval of the same should be submitted to respective Property Manager before 20:30hrs.\n17. No music system is allowed in the premises.\n18. Any complaint (indecent behavior/noisy) from the neighbors/society will result in severe action.\n19. Ragging is strictly prohibited inside the premises."**; //inserting text

newfile.close();

```

    //close the file object
}
newfile.open("TNC.txt",ios::in);
//open a file to perform read operation using file object
if (newfile.is_open())
//checking whether the file is open
{
string tp;
while(getline(newfile, tp))
{
    //read data from file object and put it into string.
    cout << tp << "\n";
    //print the data of the string
}
newfile.close();
//close the file object.
}
}

void meals()
{
    ofstream newfile;
    newfile.open("Meals.txt",ios::app);
    // open a file to perform write operation using file object

    for(int i=0; i<6; i++)

```

```

{
    cout<<endl;
    cout<<"Enter day: ";
    cin>>day;
    cout<<"Enter Lunch menu: ";
    cin>>lunch;
    cout<<"Enter dinner Menu: ";
    cin>>dinner;
    newfile<<day<<" | "<<lunch<<" | "<<dinner<<endl;
}
newfile.close();
}

void laundry_cleaning()
{
    ifstream newfile;
    newfile.open("Laundry_cleaning.csv", ios::in);
    //open a file to perform read operation using file object
    if (newfile.is_open())
    //checking whether the file is open
    {
        string tp;
        while(getline(newfile, tp))
        {
            //read data from file object and put it into string.
            cout << tp << "\n"; //print the data of the string
        }
    }
}

```

```

    }

    newfile.close();

    //close the file object.

}}

};

/*
*****

class student
*****

*/

class student
//creating a class student, parent class of login
{
    int room;

    string first_name, last_name;

    int age=0;

    int leaves;

    double contact_num;

    char date_registration[10], date_admission[10], contact_email[10];

    char payment_status;

public:

    void login_()

    //defining and declaring a function to take an input of the login identity

```

```

{
    cout<<"\t\t\tEnter the room number:";
    //taking an input of the room number
    cin>>room;
}

```

```

void reg()
//defining and declaring a function to take input of the registration details
{
    file.open("hostel(2).csv" , ios::app);
    //opening file
    cout<<"Enter your details:\n";
    cout<<"First name: ";
    cin>>first_name;
    cout<<"Last name: ";
    cin>>last_name;
    cout<<"Age: ";
    cin>>age;
    cout<<"Date of registration: ";
    cin>>date_registration;
    cout<<"Date you want to get admitted: ";
    cin>>date_admission;
    cout<<"Contact Number: +91 ";
    cin>>contact_num;
    cout<<"Contact Email Address: ";
    cin>>contact_email;
}

```

```

file<<first_name<<","<<last_name<<","<<age<<","<<date_registration<<","<<date_ad
mission<<","<<contact_num<<","<<contact_email;

    //inputting data in the file
    file.close();
    //closing the file
    system("cls");
    //clearing the console
    cout<<"The request has been sent and you will get a response for the same soon.";
}

```

```

void refund()

```

```

/*defining and declaring a member function to take an input if student proceeds with
cancellation and proceed accordingly*/

```

```

{
    int n; int d;
    cout<<"Payment status: enter 1 if completed, or 0 if not:"<<endl; cin>>n;
    switch(n)
    {
        case 1:
        {
            cout<<"Enter days left before joining: "<<endl; cin>>d;
            if(d>=30)
            {
                cout<<"100% Refund."<<endl;
            }
        }
    }
}

```



```

else if (d>=15)
{
    cout<<"80% Refund."<<endl;
}
else if (d<15)
{
    cout<<"50% Refund"<<endl;
}
break;
}
case 2:
{
    cout<<"no refund."<<endl;
    break;
}
}
}

void entry();

//declaring a function, will define it outside the class

void leave()
//declaring and defining function leave to get permission of the leaves
{
    cout<<"Please enter number of leaves:"<<endl;
    cin>>leaves;

```

```

cout<<"Leave days: "<<leaves<<endl;
cout<<"You will receive your leave grant within a span of 48 hours."<<endl<<"Thank
you!"<<endl;
}

```

```

void laundry_cleaning()
//declaring and defining function to keep a record of laundry cleaning
{
    string l,c;
    ofstream laundry("Laundry_cleaning.csv", ios::app);
    //appending data
    laundry<<"Room no , Laundry , Cleaning\n";
    cout<<"Do you require laundry done?"<<endl;
    cin>>l;
    cout<<"Do you require Room cleaning done?"<<endl;
    cin>>c;
    laundry<<room<<" "<<l<<" "<<c;
    laundry.close();
    //closing the file
}

```

```

void meals()
{
    ifstream newfile;
    newfile.open("Meals.txt", ios::in);
    //open a file to perform read operation using file object
}

```

```

if (newfile.is_open())
//checking whether the file is open
{
string tp;
while(getline(newfile, tp))
{
//read data from file object and put it into string.
cout << tp << "\n";
//print the data of the string
}
newfile.close();
//close the file object.
}
};

```

```

void student::entry()
//defining function outside the class
//taking an input of entry and exit
{
int m, t, t1;
string enter;
cout<<"do you want to enter entry time or exit time?"<<endl;
cin>>enter;
if(enter=="entry" || enter=="Entry")
{

```

```

cout<<"enter entry time (24hrs format): "<<endl;
cin>>t>>t1;
if(t>22)
{
    cout<<"your parents have been informed."<<endl;
}
else
{
    cout<<"entry time: "<<t<<":"<<t1;
}

}

else if(enter=="exit" || enter=="Exit")
{
    cout<<"enter exit time (24hrs format): "<<endl;
    cin>>t>>t1;
    if (t>20)
    {
        cout<<"you are not allowed to leave the hostel as per the rules."<<endl;
    }
    else
    {
        cout<<"Exit time: "<<t<<":"<<t1;
    }
}
}

```

```
}
```

```
/*
```

```
*****
```

```
class login: child of student and admin
```

```
*****
```

```
*/
```

```
class login:public student, public admin
```

```
//secures private data from student and admin
```

```
{
```

```
    string login_register;
```

```
    string identity;
```

```
    student s;
```

```
    admin a;
```

```
    //private data members declared
```

```
public:
```

```
    login()
```

```
{
```

```
    cout<<endl<<"\n\n\n\n\n\n\n\n\n\n"<<endl;
```

```
    cout<<"\t\t\t\t\tWelcome to XYZ Hostel Management System."<<endl;
```

```
    cout<<"\t\t\t\t\tDo you want to login or register?"<<endl;
```

```
    cout<<"\t\t\t\t\t";
```

```
    cin>>login_register;
```

```
    system("cls");
```

```
    //clearing the console
```

```
}
```

```
void login_or_reg()
```

```
//login page and verification
```

```
{
```

```
if (login_register=="login" || login_register=="Login")
```

```
{
```

```
cout<<endl<<"\n\n\n\n\n\n\n\n\n"<<endl;
```

```
cout<<"\t\t\t\t\tAre you Admin or Hostelite?"<<endl;
```

```
cout<<"\t\t\t\t\t";
```

```
cin>>identity;
```

```
system("cls");
```

```
}
```

```
else if (login_register=="Register" || login_register=="register")
```

```
{
```

```
s.reg();
```

```
}
```

```
else
```

```
{
```

```
clog<<"Enter either login or register."<<endl;
```

```
}
```

```
//displaying error if undesirable input is received
```

```
}
```

```

void login_con()
//login condition (identity)
{
    if(identity=="hostelite" || identity=="Hostelite")
    {
        s.login_();
        int ch; char g;
        do
        {
            cout<<"\t\t\t\t\t";
            cout<<"1.Entry/Exit"<<endl;
            cout<<"\t\t\t\t\t";
            cout<<"2.Cancellation/ Refund"<<endl;
            cout<<"\t\t\t\t\t";
            cout<<"3.TnC's Rules and Regulations"<<endl;
            cout<<"\t\t\t\t\t";
            cout<<"4.Leaves"<<endl;
            cout<<"\t\t\t\t\t";
            cout<<"5.Meals Information"<<endl;
            cout<<"\t\t\t\t\t";
            cout<<"6.Laundry and Cleaning Details"<<endl;
            cout<<"\t\t\t\t\t";
            cout<<"Enter your choice"<<endl;
            cout<<"\t\t\t\t\t";
            cin>>ch;

```



```
system("cls");  
switch(ch)  
{  
    case 1:  
    {  
        s.entry();  
        break;  
    }  
    case 2:  
    {  
        s.refund();  
        break;  
    }  
    case 3:  
    {  
        a.TNC();  
        break;  
    }  
    case 4:  
    {  
        s.leave();  
        break;  
    }  
    case 5:  
    {  
        s.meals();
```

```

        break;
    }
    case 6:
    {
        s.laundry_cleaning();
        break;
    }
}

cout << "\nDo you want to continue(Y / N) ";
cin >>g;
system("cls");
}while (g == 'Y' | | g == 'y');
}

else if (identity=="admin" | | identity=="Admin")
{
    char g;
    do
    {
        int x;
        cout<<"\t\t\t\t\t";
        cout<<"Enter 1 to view accommodation data base."<<endl;
        cout<<"\t\t\t\t\t";
        cout<<"Enter 2 to update accommodation data base."<<endl;
        cout<<"\t\t\t\t\t";
        cout<<"Enter 3 to check students requirements (laundry/cleaning)."<<endl;

```

```

cout<<"\t\t\t\t\t";
cout<<"Enter 4 to update meals information."<<endl;
cout<<"\t\t\t\t\t";
cout<<"Enter your choice"<<endl;
cout<<"\t\t\t\t\t";
cin>>x;
system("cls");

switch(x)
{
case 1:
    a.view_accommodation_database(a);
    break;
case 2:
    a.update_accommodation_database(a);
    break;
case 3:
    a.laundry_cleaning();
    break;
case 4:
    a.meals();
    break;
}
cout << "\nDo you want to continue(Y / N) ";
cin >>g;
system("cls");

```

```

        }while (g =='Y' || g =='y');
    }

    else
    {
        clog<<"Enter your input again."<<endl;
    }
    //displaying error if undesirable input is received
}
};

```

```

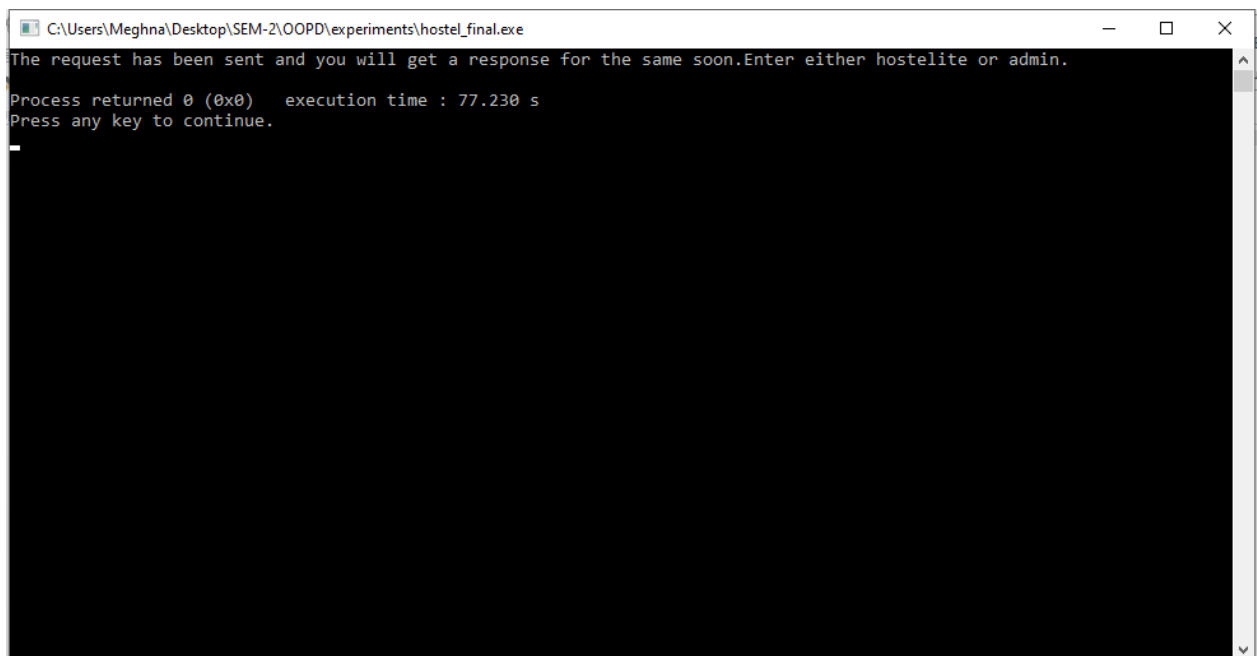
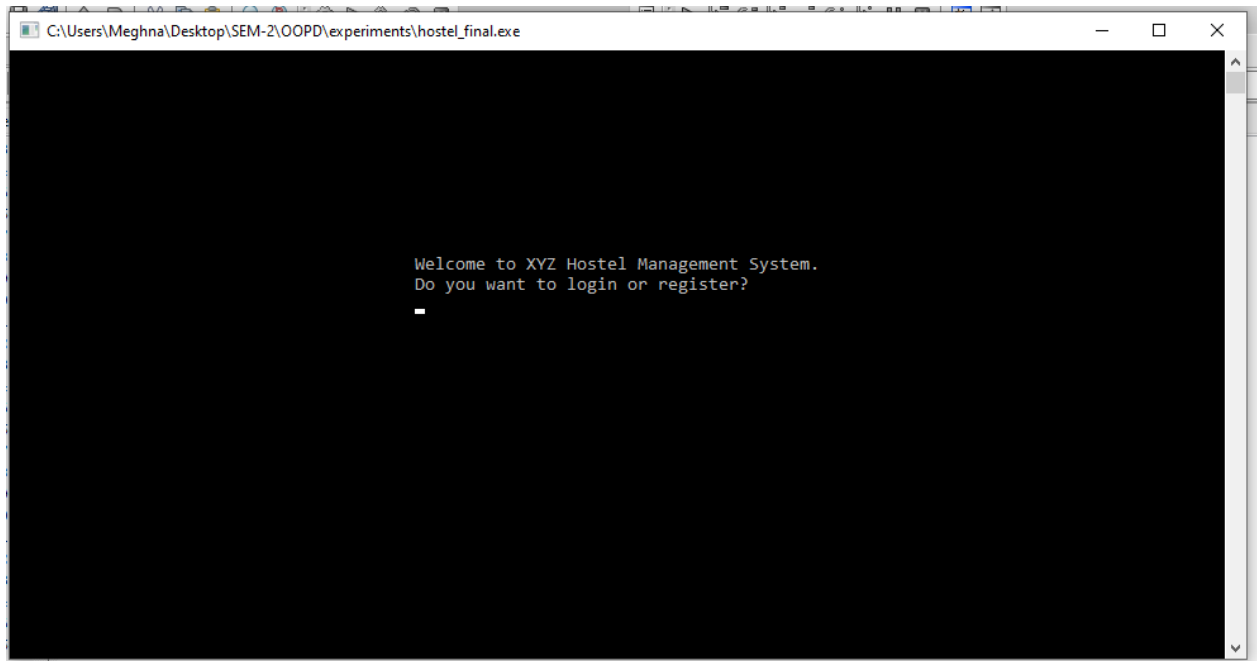
int main()
//main function
{
    login l;
    //creating an instance of login
    l.login_or_reg();
    //calling login page
    l.login_con();
    //hostelite or admin access
}

```

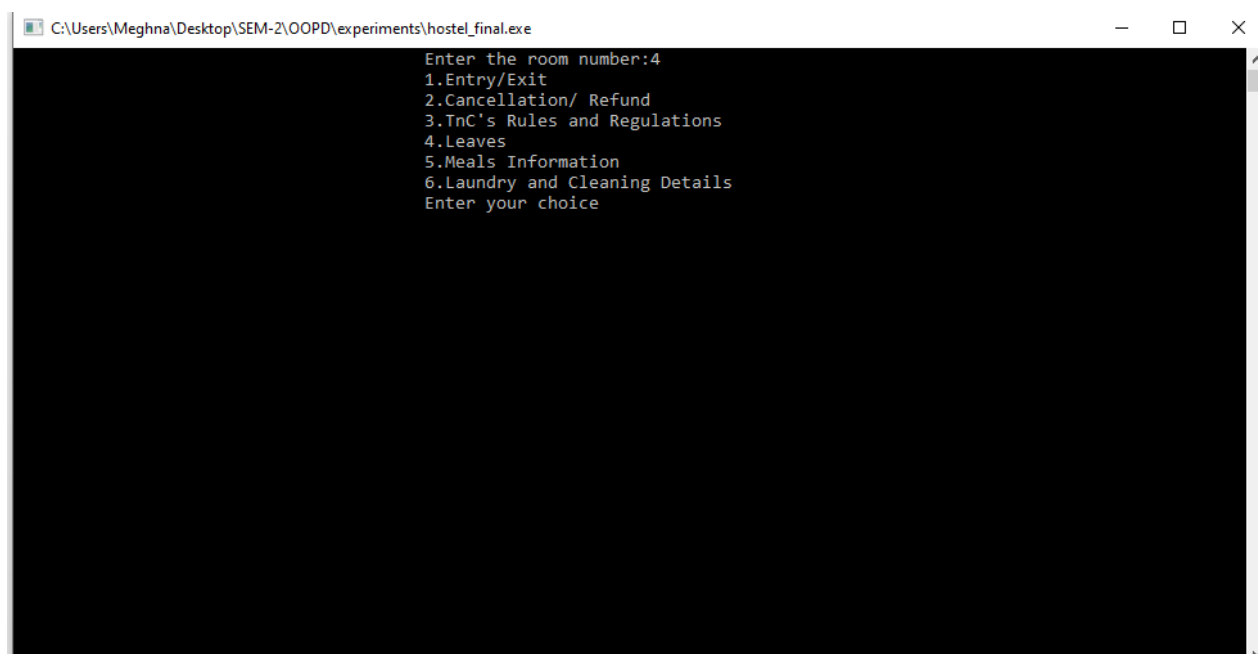
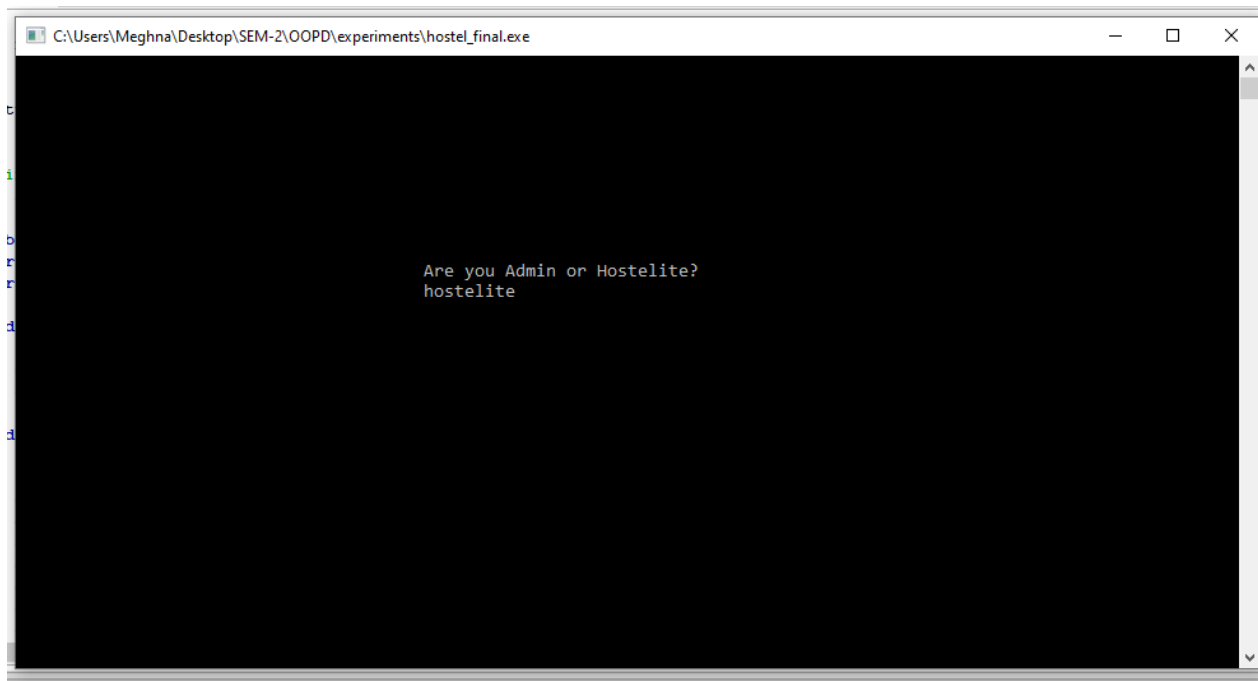
.

14. Output Screenshots

1. Login/Register Page

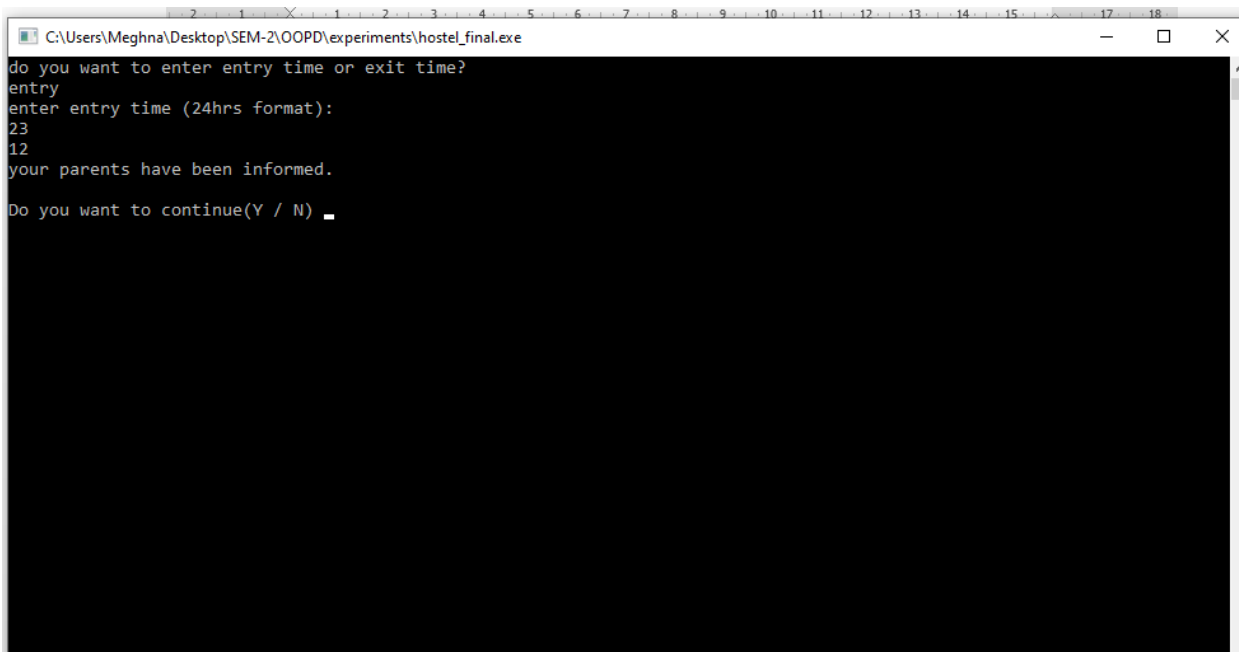


2. Student Interface





```
do you want to enter entry time or exit time?
entry
enter entry time (24hrs format):
12
30
entry time: 12:30
Do you want to continue(Y / N) _
```



```
do you want to enter entry time or exit time?
entry
enter entry time (24hrs format):
23
12
your parents have been informed.
Do you want to continue(Y / N) _
```

```
C:\Users\Meghna\Desktop\SEM-2\OOPD\experiments\hostel_final.exe
do you want to enter entry time or exit time?
exit
enter exit time (24hrs format):
14
22
Exit time: 14:22
Do you want to continue(Y / N) _
```

```
C:\Users\Meghna\Desktop\SEM-2\OOPD\experiments\hostel_final.exe
do you want to enter entry time or exit time?
exit
enter exit time (24hrs format):
22
12
you are not allowed to leave the hostel as per the rules.
Do you want to continue(Y / N)
```



```
C:\Users\Meghna\Desktop\SEM-2\OOPD\experiments\hostel_final.exe
Payment status: enter 1 if completed, or 0 if not:
0
Do you want to continue(Y / N)
```

```
C:\Users\Meghna\Desktop\SEM-2\OOPD\experiments\hostel_final.exe
Payment status: enter 1 if completed, or 0 if not:
1
Enter days left before joining:
45
100% Refund.
Do you want to continue(Y / N) _
```

```
C:\Users\Meghna\Desktop\SEM-2\OOPD\experiments\hostel_final.exe
Payment status: enter 1 if completed, or 0 if not:
1
Enter days left before joining:
22
80% Refund.
Do you want to continue(Y / N)
```

```
C:\Users\Meghna\Desktop\SEM-2\OOPD\experiments\hostel_final.exe
Payment status: enter 1 if completed, or 0 if not:
1
Enter days left before joining:
12
50% Refund
Do you want to continue(Y / N) _
```

```
C:\Users\Meghna\Desktop\SEM-2\OOPD\experiments\hostel_final.exe

He/She shall take proper care of the furniture and fixtures handed over to him / her.
The Administrator has the right to enter and inspect the rooms at any time.
8. All matters relating to differences among students and complaints about the servants shall be brought to the notice of the Property Manager, who will take action as may be necessary.
9. Students must switch off the lights, fans AC and Bathroom Geyser in their rooms every time they go out and take precautions to economize electricity consumption.
Air-condition can be used 24x7 but it must be switched off if there is no one in the room.
10. Charges for any damages to the property as well as to the furniture and fixtures caused by student/students negligence will be recovered from the student staying in the said premises.
11. Student should not drive nails, screws etc. into the wall or doors. No repair shall be done by the students themselves. They should raise a ticket on hostel's Application or approach the respective Property Manager who will arrange for repairs.
12. Visitors are not allowed to enter any room. They have to sit at the common area.
13. All the facilities including T.V., Magazines, Newspaper, Internet etc., if misused, shall be discontinued without given any notice and disciplinary action will be taken against the students involved.
14. Before leaving the PG Accommodation, a student must pay all dues and hand over the charges of rooms and other material in satisfactory condition to the respective Property Manager of the premises.
15. If any student is found misbehaving and misconducting himself, he/she will be expelled from the Accommodation immediately and the deposit paid by him / her will be forfeited.
16. Permission letter for night outs from parents should be mailed at XYZhostel@gmail.com and approval of the same should be submitted to respective Property Manager before 20:30hrs.
17. No music system is allowed in the premises.
18. Any complaint (indecent behavior/noisy) from the neighbors/society will result in severe action.
19. Ragging is strictly prohibited inside the premises.

Do you want to continue(Y / N)
```

```
C:\Users\Meghna\Desktop\SEM-2\OOPD\experiments\hostel_final.exe

Please enter number of leaves:
23
Leave days: 23
You will receive your leave grant within a span of 48 hours.
Thank you!

Do you want to continue(Y / N)
```

C:\Users\Meghna\Desktop\SEM-2\OOPD\experiments\hostel_final.exe

Day	Lunch	Dinner
Monday	Rajma Chawal	Chole Bature
Tuesday	Mix Veg	Tawa pulao
Wednesday	Paneer Masala	Dal Makhni
Thursday	Malai Kofta	Pav Bhaji
Friday	Dry Bhindi	Manchurian
Saturday	Aloo sabji	Rajma Chawal

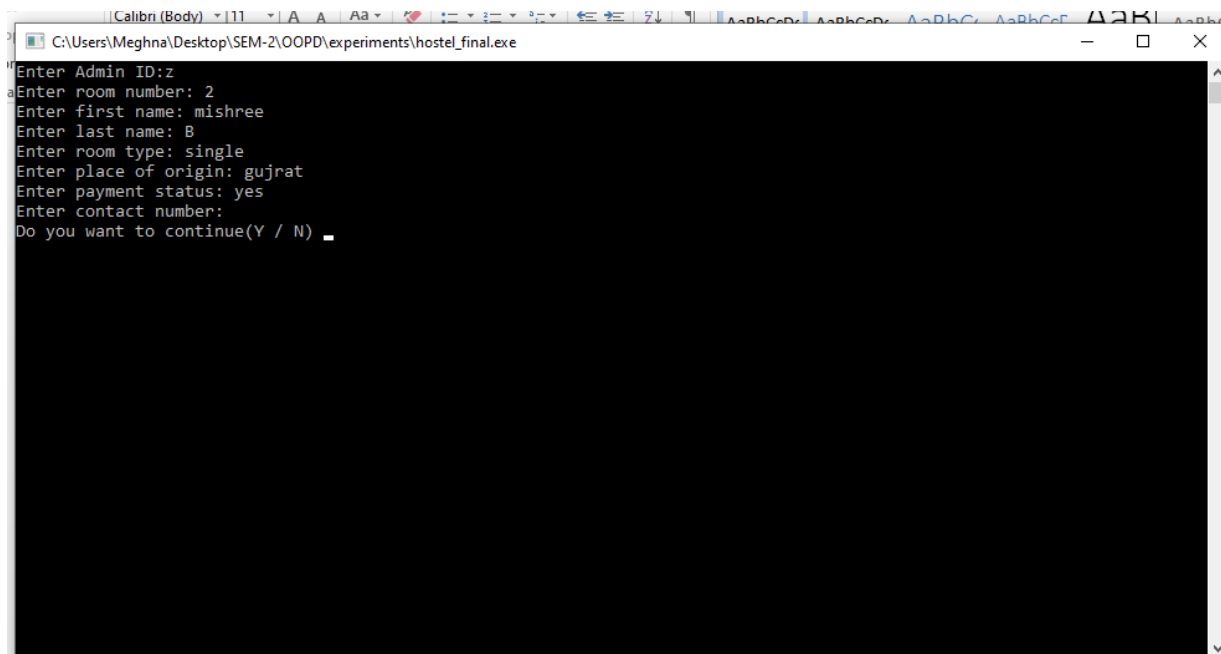
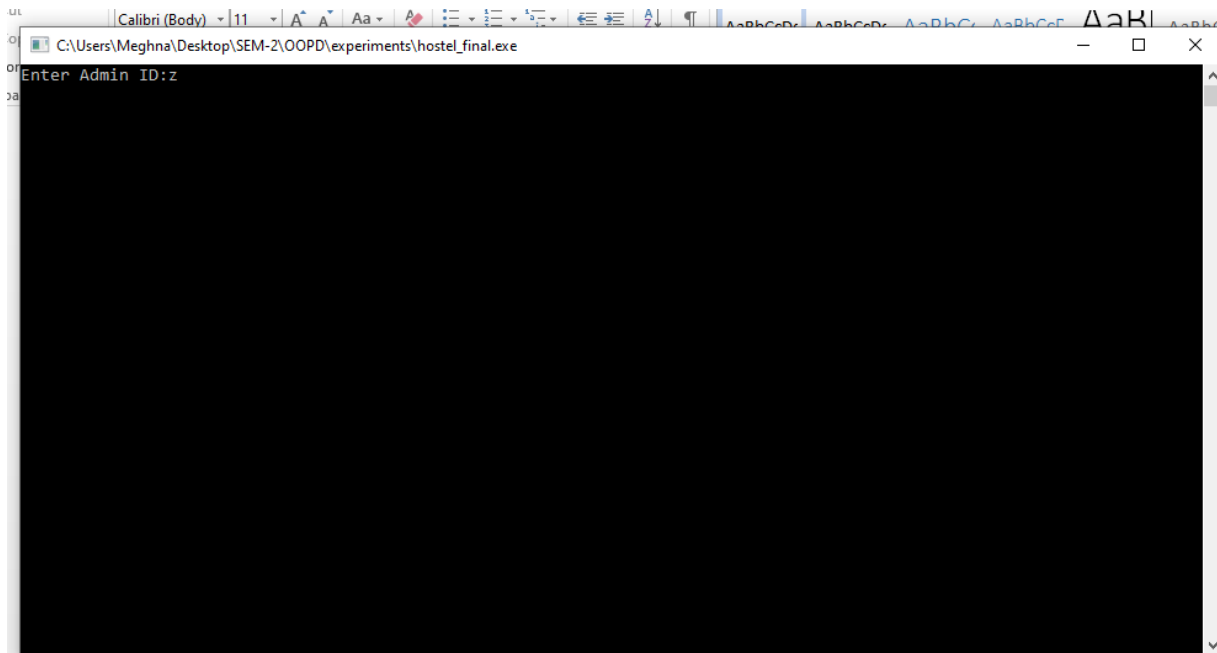
C:\Users\Meghna\Desktop\SEM-2\OOPD\experiments\hostel_final.exe

Do you require laundry done?
yes
Do you require Room cleaning done?
yes
Do you want to continue(Y / N) _

C:\Users\Meghna\Desktop\SEM-2\OOPD\experiments\hostel_final.exe

Enter 1 to view accommodation data base.
Enter 2 to update accommodation data base.
Enter 3 to check students requirements (laundry/cleaning).
Enter 4 to update meals information.
Enter your choice

3. Admin Interface



```
C:\Users\Meghna\Desktop\SEM-2\OOPD\experiments\hostel_final.exe
Room no , Laundry , Cleaning
01,yes,yesRoom no , Laundry , Cleaning
02,y,yRoom no , Laundry , Cleaning
01,yes,noRoom no , Laundry , Cleaning
04,yes,yes

Do you want to continue(Y / N) _
```

```
C:\Users\Meghna\Desktop\SEM-2\OOPD\experiments\hostel_final.exe
Enter day: monday
Enter Lunch menu: rice
Enter dinner Menu: chinese
Enter day: tuesday
Enter Lunch menu: malaikofta
Enter dinner Menu: noodles
Enter day: wednesday
Enter Lunch menu: rajmachawal
Enter dinner Menu: dosa
Enter day: thursday
Enter Lunch menu: aloosabji
Enter dinner Menu: pavbhaji
Enter day: friday
Enter Lunch menu: mysoredosa
Enter dinner Menu: idli
Enter day: saturday
Enter Lunch menu: uttapam
Enter dinner Menu: bhindi
Do you want to continue(Y / N)
```

15. List of Errors

1. Couldn't take an input using getline because of combined use of cin and getline.
2. Received an error while implementing operator overloading because of insufficient arguments from different objects.
3. Couldn't append data in a specific file type so had to use ate for the same.
4. Received error for invalid use of non-static data member.
5. Received error when files were declared outside of the scope of functionalities.
6. Using inbuilt functions but were showing out of scope error therefore included necessary files.
7. Array of objects was a hard concept to implement due to an error of no matching function and complexity in inheritance.

16. Scope of Improvement

Like every successful project with no boundaries ever, even our project has a lot of scope of improvement and ends that haven't been completely tied due to increasing complexity.

- Rate your experience for future system improvements and feedback could be taken as an input using friend function.
- UI/UX could be worked upon and a cleaner looking interface could be created to show more from the design side of the field.
- User friendliness could be improved such as taking number inputs rather than making the user type words and displaying as much as clear information as possible.
- More functions could be declared for more flexibility between classes and objects.
- Template could be used to clarify a class of parent that could've been reused for things like automated calls, reviews and permission transfer which is happening as an external activity in our current project.
- Array of objects could be included for taking multiple updates of accommodation file by the admin.
- Student functionalities could also use array of objects for proper functioning.
- Refund and cancellation can be improvised upon by customizing it with a personalized touch using operator overloading in finding individual paybacks.
- Other functions can be involved in for payment and outer communication access with the hostel.

17. References

1. <https://www.edureka.co/blog/file-handling-in-cpp/>
2. <https://www.ibm.com/docs/en/zos/2.4.0?topic=reference-inheritance-c-only>
3. <https://www.programiz.com/cpp-programming/polymorphism>
4. <https://stackify.com/oop-concept-abstraction/>
5. https://www.tutorialspoint.com/cplusplus/cpp_classes_objects.htm
6. Book: Bjarne Stroustrup, “The C++ Programming Language” (4th Edition), Addison-Wesley, May 2013
7. Book: Debasish Jana, “C++ and Object-Oriented Programming Paradigm”, 3rd Edition, 2014