OBJECT ORIENTED PROGRAMMING (SE-203)

MTE PROJECT REPORT

SUBMITTED TO: Prof. Manoj Sethi

SUBMITTED BY: Shivam Bharti(2K19/SE/115)
Pawan(2K19/SE/083)

Project TITLE:

FLAT DEALING SYSTEM

AKNOWLEDGEMENT

We, (**Shivam Bharti(2k19/SE/115)**, **Pawan(2K19/SE/083)**) express our sincere gratitude to **Prof. Manoj Sethi** of Department of computer Science (Software Engineering) for his valuable guidance and timely suggestions during the entire duration of our dissertation work, without which the work would not have been possible. Finally, I would to thanks my friends for their advice and pointing out the mistakes.

CONTENTS

PROBLEM DEFINITION

The flat dealer has to face so many problems while managing the records of the buyers on paper. So, there’s a need to make a program that manages all the records efficiently and gives the user/buyer various facilities to put a flat on hold, show, edit a record, delete a record, etc. using map template for mapping flats holder details with flat number and insert into a file.

INTRODUCTION

This project is designed to save the data of all those customers who hire, lease or buy any kind of flats which will reduce manual work and help the dealer to save the records efficiently. This project is aimed at developing a flat dealing software. We are managing the records of buyers by the help of data structures. We are using ***Map template*** so that the flat holder details are mapped together with flat number. We have also used ***file handling and operator overloading*** so that we can write the details into the file as well as read the details through the file by calling upon an object, keep a non-volatile record of all the buyers efficiently and we have designed the software like that the dealer gets the details of all buyers sorted in whichever condition he may require, for e.g. – Buyers who want 2 BHK flat , or buyers who want a flat at a specific location , etc… We have designed various functions in this software that do specific tasks like we are able to edit a record if a buyer changes his mind for buying another type of another type of flat and we are also able to delete a record if in case dealing with a certain buyer is cancelled.

THEORY

- **Object Oriented Programming:**

Procedural programming is about writing procedures or functions that perform operations on the data, while object-oriented programming is about creating objects that contain both data and functions.

Classes and objects are the two main aspects of object-oriented programming. So, a class is a template for objects, and an object is an instance of a class. When the individual objects are created, they inherit all the variables and functions from the class.

→Constructors:

A constructor in C++ is a *special method* that is automatically called when an object of a class is created. To create a ***default constructor***, use the same name as the class, followed by parentheses `()`.

Constructors can also take parameters called as ***parametrized constructors*** (just like regular functions), which can be useful for setting initial values for attributes.

Just like functions, constructors can also be defined outside the class. First, declare the constructor inside the class, and then define it outside of the class by specifying the name of the class, followed by the scope resolution`::` operator, followed by the name of the constructor (which is the same as the class).

→Encapsulation:

The meaning of **Encapsulation**, is to make sure that "sensitive" data is hidden from users. To achieve this, we must declare class variables/attributes as `private` (cannot be accessed from outside the class). If we want others to read or modify the value of a private member, we can provide public **get** and **set** methods.

Friend Function: A friend function of a class is defined outside that class' scope but it has the right to access all private and protected members of the class. Even though the prototypes for friend functions appear in the class definition, friends are not member functions.

→Operator Overloading:

You can redefine or overload most of the built-in operators available in C++. Thus, a programmer can use operators with user-defined types as well.

Overloaded operators are functions with special names: the keyword "operator" followed by the symbol for the operator being defined. Like any other function, an overloaded operator has a return type and a parameter list.

→ File Handling:

The `fstream` library allows us to work with files.

To use the fstream library, include both the standard <iostream> **AND** the <fstream> header file:

ofstream	Creates and writes to files
ifstream	Reads from files
fstream	A combination of ofstream and ifstream: creates, reads, and writes to files

To create a file, use either the ofstream or fstream class, and specify the name of the file.

To write to the file, use the insertion operator (<<).

→Input/Output Operator Overloading in C++

C++ is able to input and output the built-in data types using the stream extraction operator >> and the stream insertion operator <<. The stream insertion and stream extraction operators also can be overloaded to perform input and output for user-defined types like an object.

Here, it is important to make operator overloading function a friend of the class because it would be called without creating an object.

• Map in C++ Standard Template Library (STL)

Maps are associative containers that store elements in a mapped fashion. Each element has a key value and a mapped value. No two mapped values can have same key values.

Some basic functions associated with Map:

begin() – Returns an iterator to the first element in the map
end() – Returns an iterator to the theoretical element that follows last element in the map.

pair insert(keyvalue, mapvalue) – Adds a new element to the map
erase(iterator position) – Removes the element at the position pointed by the iterator

→ map::insert()

is a built-in function in C++ STL which is used to insert elements with a particular key in the map container.

Syntax:

```
iterator map_name.insert({key, element})
```

Parameters: The function accepts a pair that consists of a key and element which is to be inserted into the map container. The function does not insert the key and element in the map if the key already exists in the map.
Return Value: The function returns an iterator pointing to the new element in the container.

→map::begin()

begin() function is used to return an iterator pointing to the first element of the map container. begin() function returns a bidirectional iterator to the first element of the container.

→map::erase()

is a built-in function in C++ STL which is used to erase element from the container. It can be used to erase keys, elements at any specified position or a given range.

Syntax for erasing a key:

```
map_name.erase(key)
```

Parameters: The function accepts one mandatory parameter *key* which specifies the key to be erased in the map container.

Return Value: The function returns 1 if the key element is found in the map else returns 0.

→map::end()

end() function is used to go through an iterator pointing to the end element of the map container. end() function returns a bidirectional iterator to the last element of the container.

- **STL Iterators:**

The concept of an iterator is fundamental to understanding the C++ Standard Template Library (STL) because iterators provide a means for accessing data stored in container classes such a vector, map, list, etc.

we can think of an iterator as pointing to an item that is part of a larger container of items. For instance, all containers support a function called begin, which will return an iterator pointing to the beginning of the container (the first element) and function, end, that returns an iterator corresponding to having reached the end of the container. In fact, you can access the element by "dereferencing" the iterator with a *, just as you would dereference a pointer.

To request an iterator appropriate for a particular STL templated class, you use the syntax.

```
std::class_name<template_paramter>::iterator name
```

e.g. if we want to use for map then syntax:

```
map<int,int>::iterator  itr;
```

OBJECTIVES

- To design a program which help the flat dealer to manage the records of buyers properly.
- Which help customers to choose, change or even delete their choice without any hesitation.
- Provide a service to customers to book their flats online.

SYSTEM REQUIREMENTS

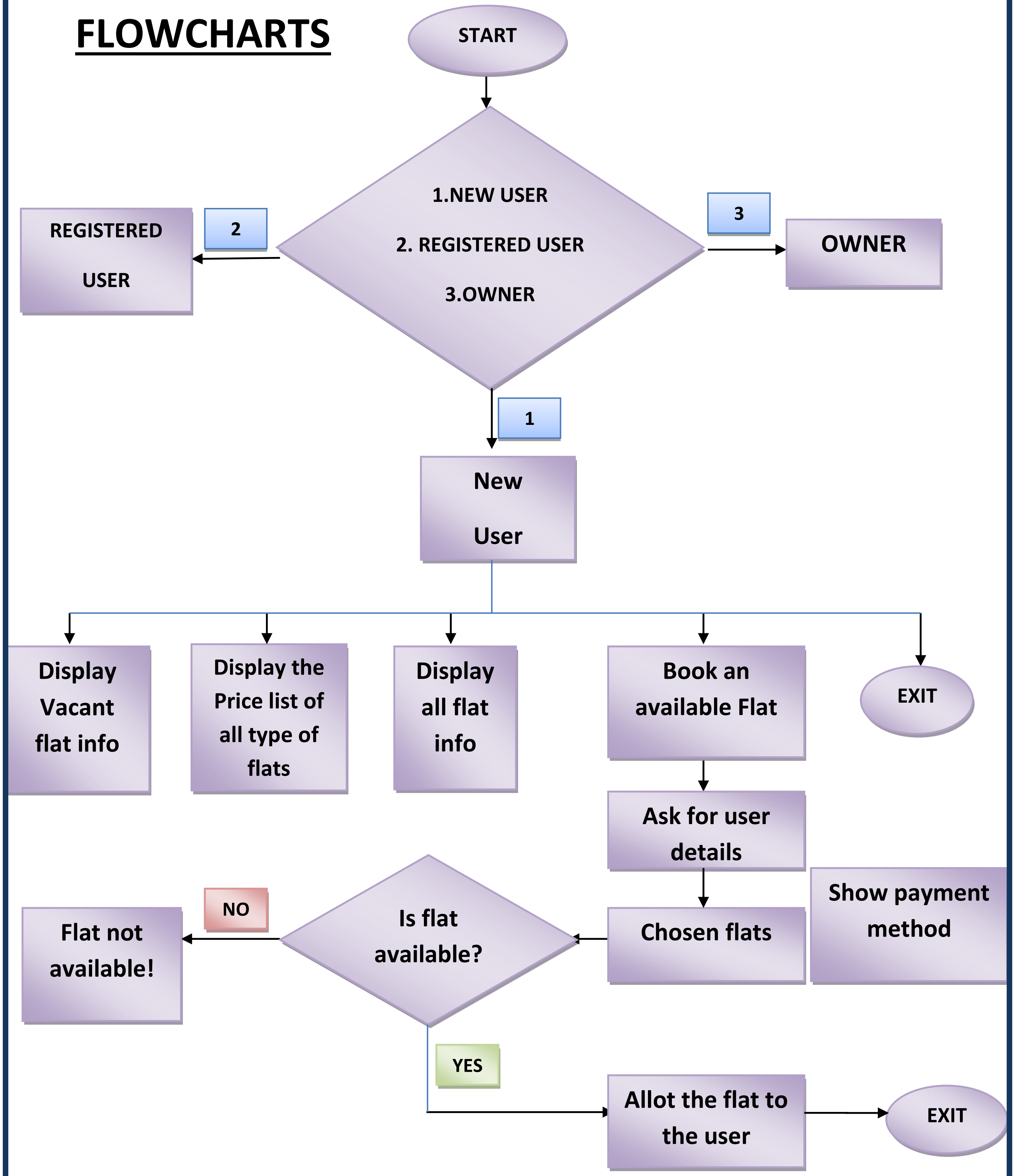
Min Hardware requirements:

- ➔ Processor: Pentium Dual-Core
- ➔ RAM: 2GB
- ➔ Hard Disk: 1 or 2 GB free space at least
- ➔ Operating System: Windows XP, Vista 7,8,10.

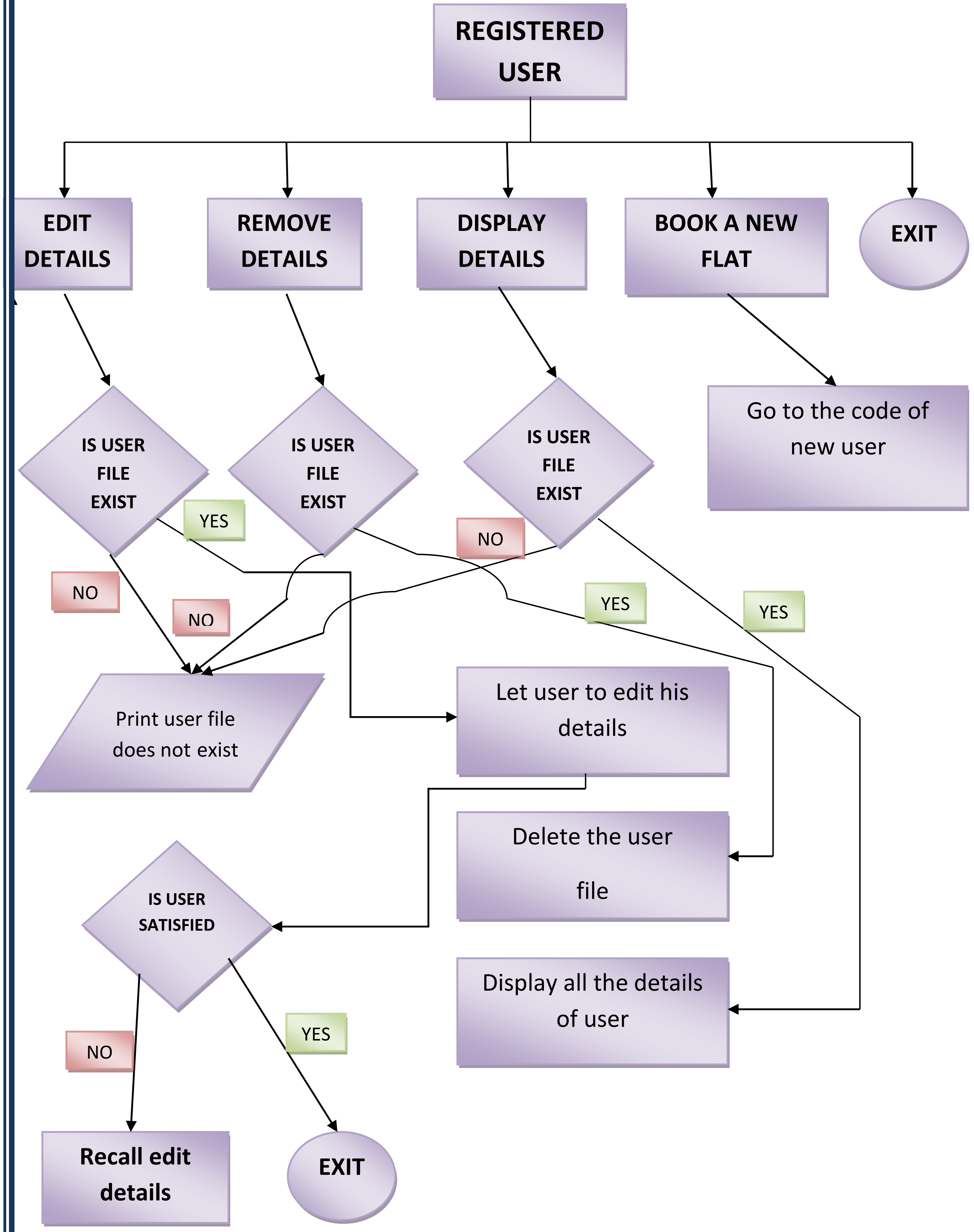
AIM

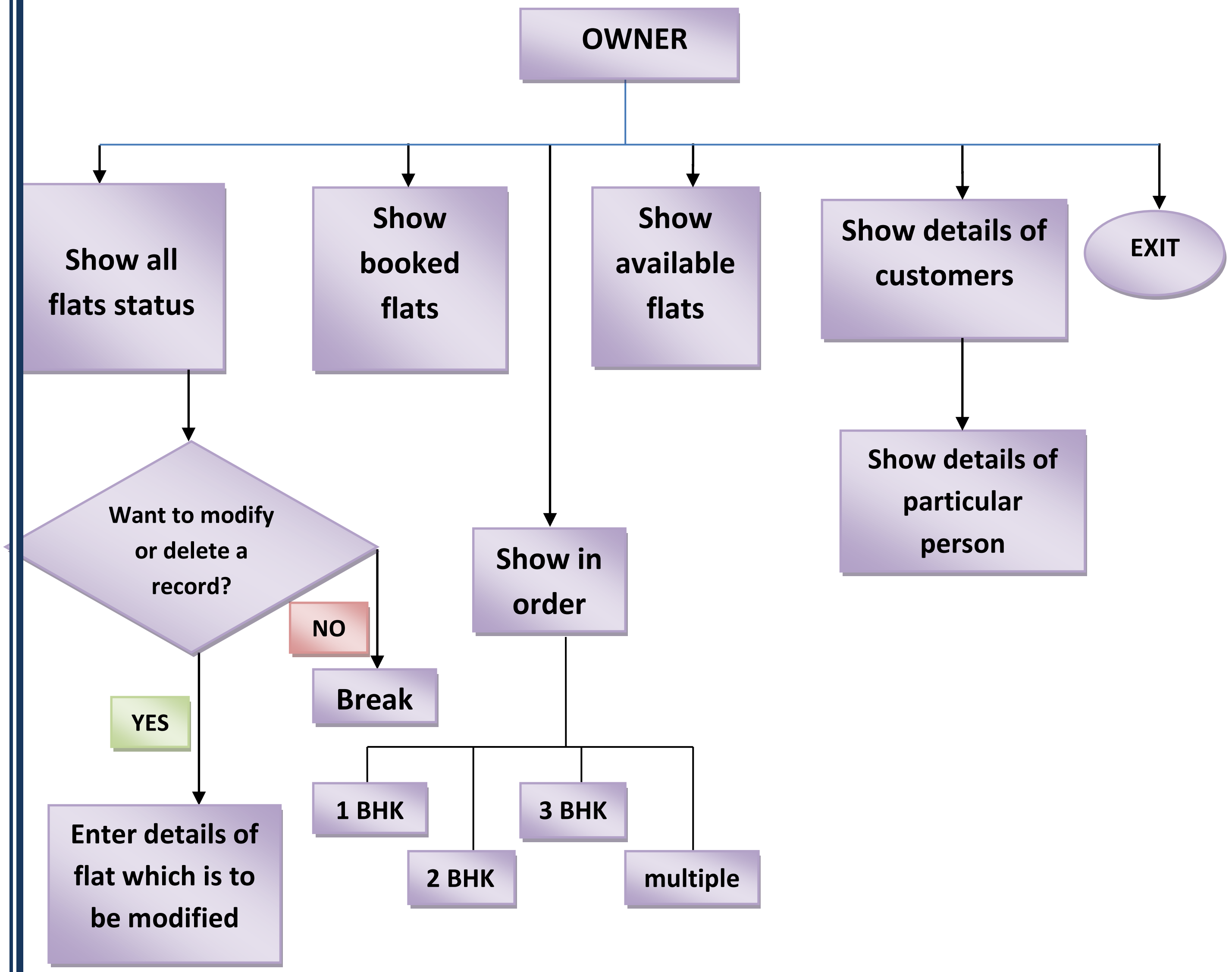
We are aiming to proper maintenance of records of buyers for owner and also to provide every information of flats and various facilities to our customers, so that they choose their desirable flats.

FLOWCHARTS



START
1.NEW USER
2. REGISTERED USER
3.OWNER
2
REGISTERED USER
3
OWNER
1
New User
Display Vacant flat info
Display the Price list of all type of flats
Display all flat info
Book an available Flat
EXIT
Ask for user details
Show payment method
Chosen flats
Is flat available?
NO
Flat not available!
YES
Allot the flat to the user
EXIT


REGISTERED USER
EDIT DETAILS
REMOVE DETAILS
DISPLAY DETAILS
BOOK A NEW FLAT
EXIT
IS USER FILE EXIST
IS USER FILE EXIST
IS USER FILE EXIST
Go to the code of new user
YES
NO
NO
NO
YES
YES
Print user file does not exist
Let user to edit his details
Delete the user file
IS USER SATISFIED
Display all the details of user
NO
YES
Recall edit details
EXIT


OWNER
Show all flats status
Show booked flats
Show available flats
Show details of customers
EXIT
Show details of particular person
Want to modify or delete a record?
NO
YES
Break
Show in order
1 BHK
2 BHK
3 BHK
multiple
Enter details of flat which is to be modified

EXPLAINATION OF MODULES

AND OUTPUT SCREENSHOTS

→Function Prototypes: -

Inside class flat:

flat() //default constructor to initialize member of class flat

flat(int room_no ,string fname ,string lname ,string address ,string phone) // parametrized const. to initialized the values to the member of class respt.

int get_room_no() // to return the flat number

string getfirstname() // to return the first name

string getlastname() // to return the last name

string getaddress() // to return the address

string getphone() // to return the Phone number

friend ofstream & operator<< (ofstream & ofs ,flat & fl);

//to write the data calling upon an object into the file

friend ifstream & operator>> (ifstream & ifs ,flat & fl);

//to read the data calling upon an object throughout the file friend ostream & operator<< (ostream & os ,flat & fl);

//to print the data calling upon an object from the file

Inside class Owner:

owner(); //default constructor to initialize member of class owner

void pricelist(); //to display the price menu of flats

flat add(); //to book a room

void displayvac(); //to display vacant flats

void display(); //to display the particular customer record

void rooms(); //to display alloted rooms

void edit(); //to edit the customer record

int check(int); //to check room status

void modify(int); //to modify the record

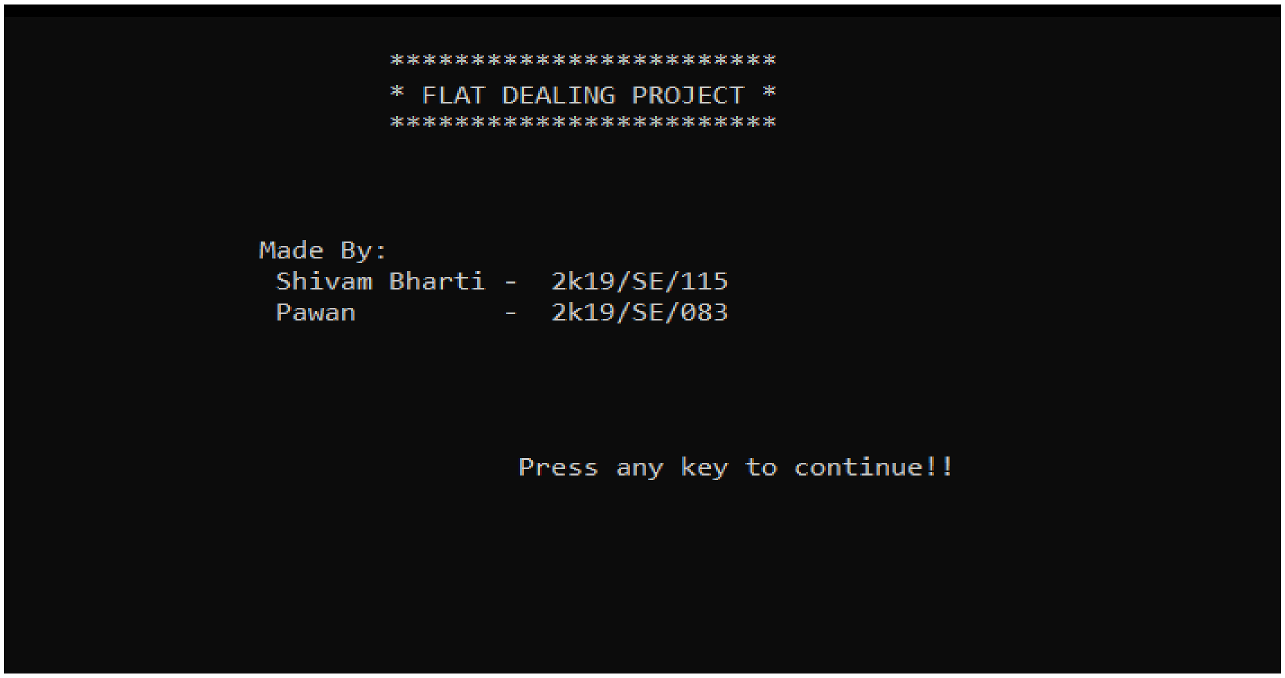
int delete_rec(int); //to delete the record

void displaybooked(); //to display booked flats

~owner(); // to deallocate the data assigned to an members .

Other:

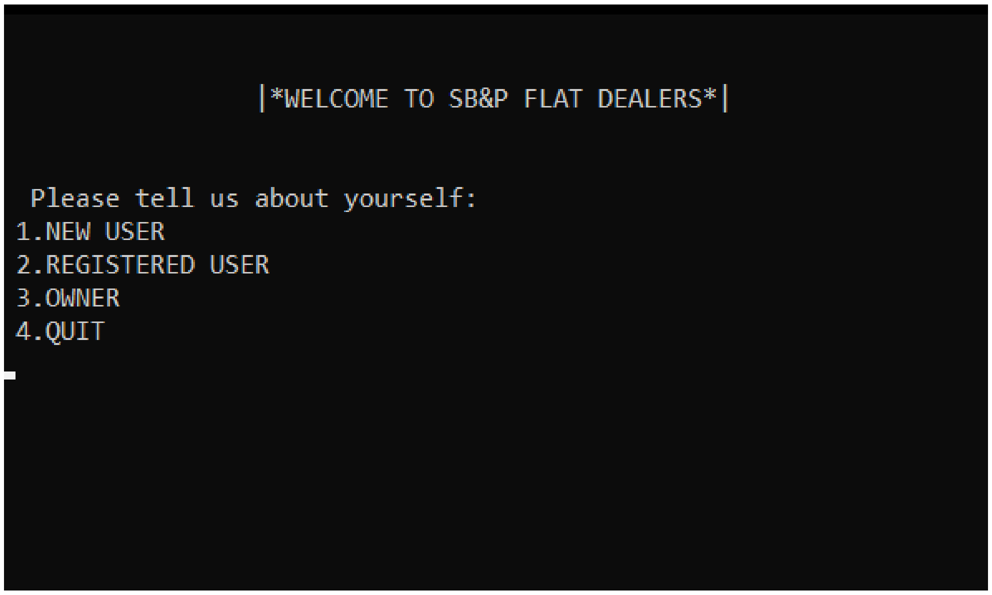
int main(); // the main body where program starts executing .


```
                    ************************
                    * FLAT DEALING PROJECT *
                    ************************



            Made By:
             Shivam Bharti -  2k19/SE/115
             Pawan         -  2k19/SE/083




                              Press any key to continue!!
```

❖ int main()

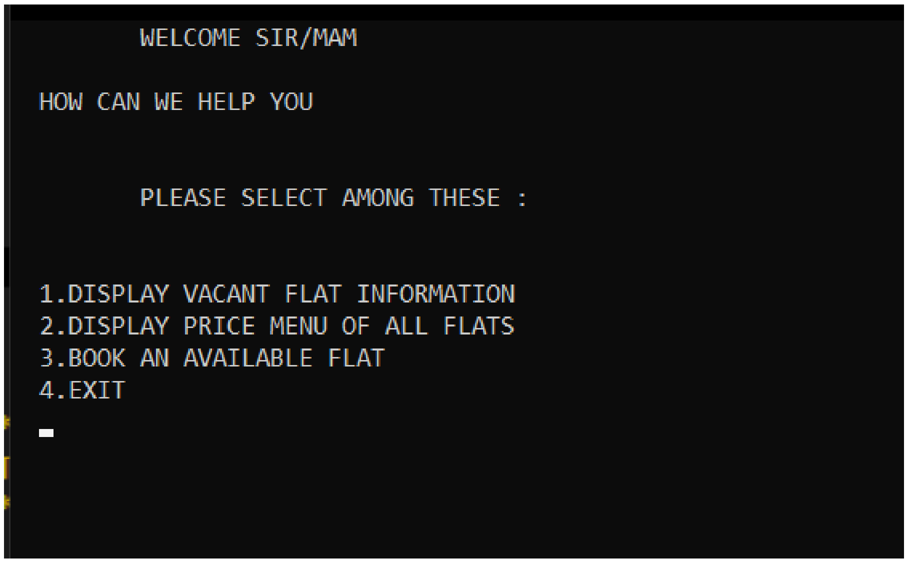
In this function gives the initial choice to the user. Here the user is first asked about himself that if he’s a new user, a registered user, or the owner of flats.
Then after choosing a particular option he’s welcomed by the program and further given the choices for accessing the details and facilities.

Also if the user is owner, then he’s asked for a password and the further choices are given only if the password matches to our pre entered string.



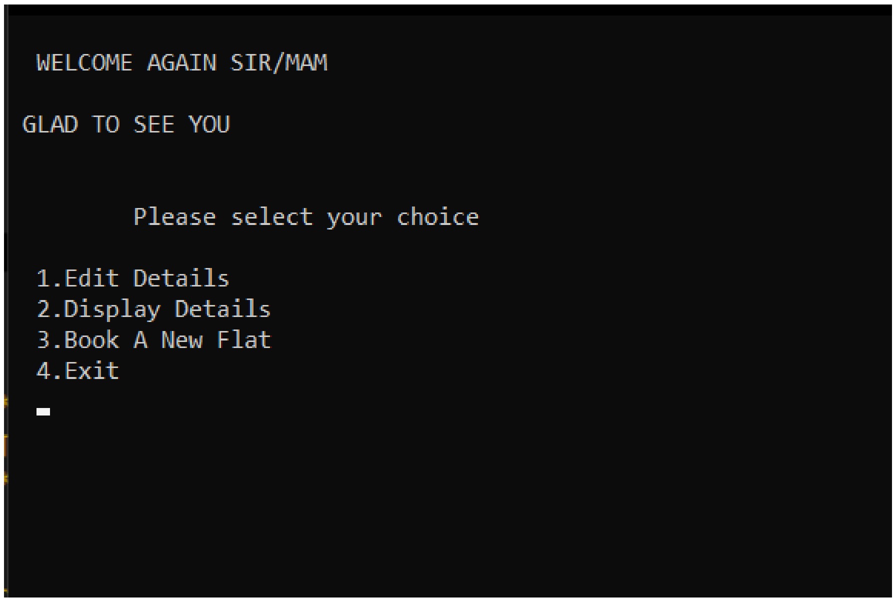
```
                 |*WELCOME TO SB&P FLAT DEALERS*|


 Please tell us about yourself:
1.NEW USER
2.REGISTERED USER
3.OWNER
4.QUIT
```

On choosing 1 :



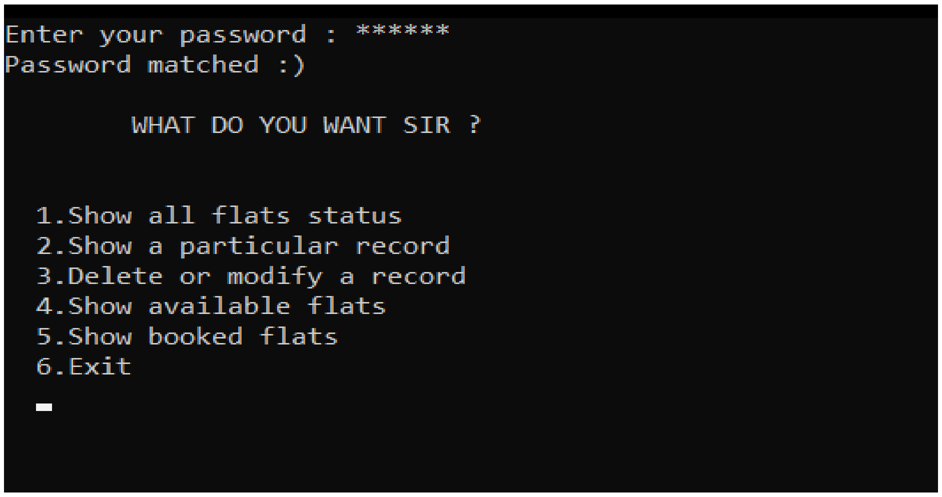
```
       WELCOME SIR/MAM

HOW CAN WE HELP YOU


       PLEASE SELECT AMONG THESE :


1.DISPLAY VACANT FLAT INFORMATION
2.DISPLAY PRICE MENU OF ALL FLATS
3.BOOK AN AVAILABLE FLAT
4.EXIT
```

On choosing 2:



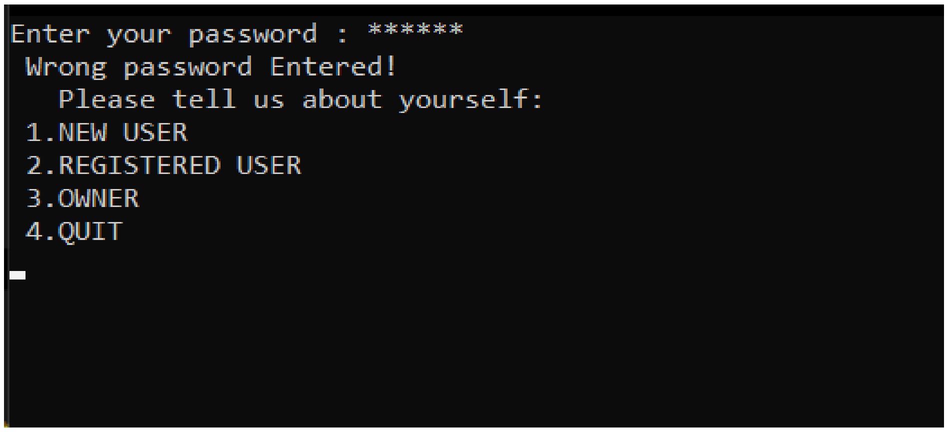
```
 WELCOME AGAIN SIR/MAM

GLAD TO SEE YOU


        Please select your choice

 1.Edit Details
 2.Display Details
 3.Book A New Flat
 4.Exit
```

On choosing 3 :



```
Enter your password : ******
Password matched :)

        WHAT DO YOU WANT SIR ?


  1.Show all flats status
  2.Show a particular record
  3.Delete or modify a record
  4.Show available flats
  5.Show booked flats
  6.Exit
```

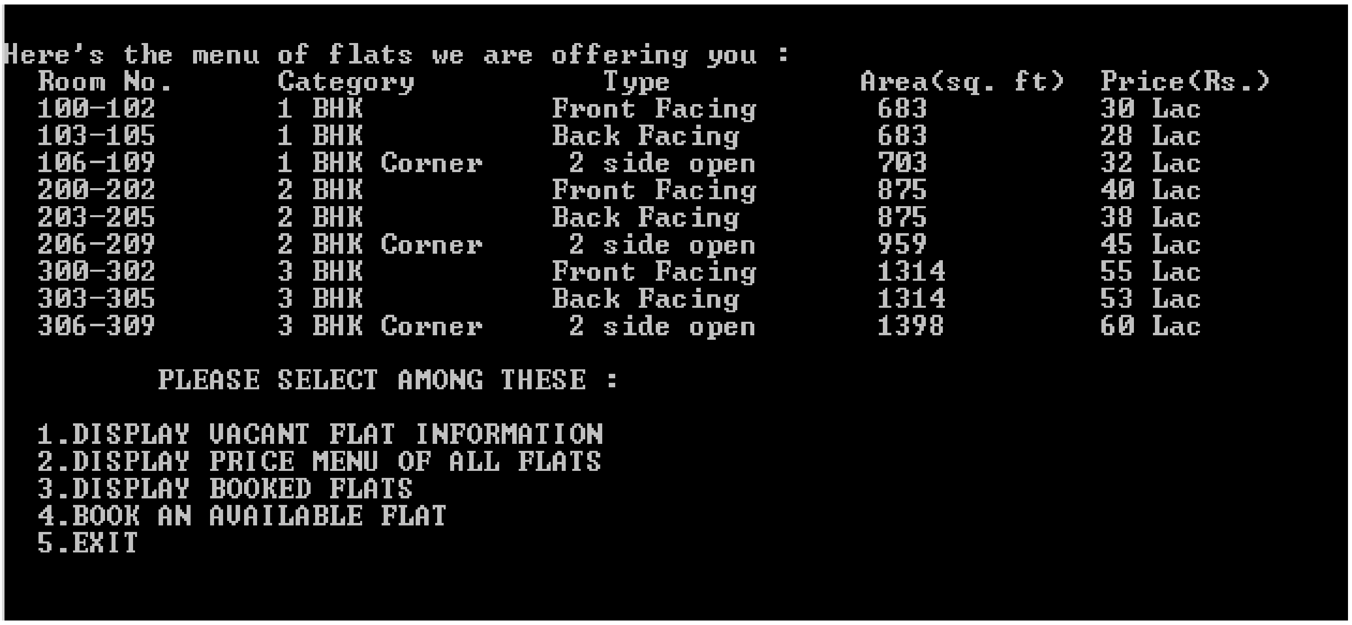
If entered wrong!....



```
Enter your password : ******
 Wrong password Entered!
   Please tell us about yourself:
 1.NEW USER
 2.REGISTERED USER
 3.OWNER
 4.QUIT
```

❖ void pricelist()

This function is used to give an idea of our flats to our new user. It displays the price list of all the flat types.
It displays various details in a tabular form such as room number, category, type of flat, area covered by flat, price for buying.

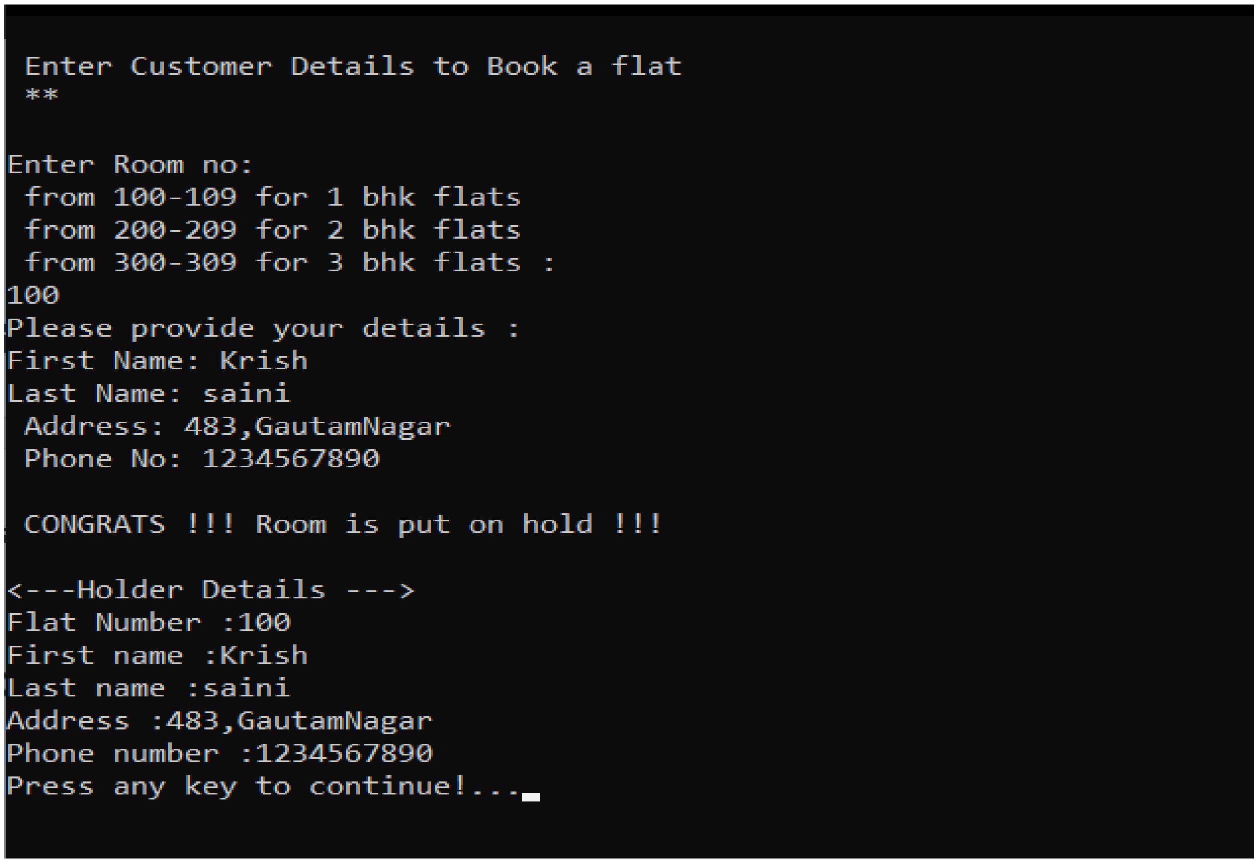


```
Here's the menu of flats we are offering you :
  Room No.      Category          Type           Area(sq. ft)  Price(Rs.)
  100-102       1 BHK           Front Facing       683           30 Lac
  103-105       1 BHK           Back Facing        683           28 Lac
  106-109       1 BHK Corner     2 side open       703           32 Lac
  200-202       2 BHK           Front Facing       875           40 Lac
  203-205       2 BHK           Back Facing        875           38 Lac
  206-209       2 BHK Corner     2 side open       959           45 Lac
  300-302       3 BHK           Front Facing       1314          55 Lac
  303-305       3 BHK           Back Facing        1314          53 Lac
  306-309       3 BHK Corner     2 side open       1398          60 Lac

         PLEASE SELECT AMONG THESE :

  1.DISPLAY VACANT FLAT INFORMATION
  2.DISPLAY PRICE MENU OF ALL FLATS
  3.DISPLAY BOOKED FLATS
  4.BOOK AN AVAILABLE FLAT
  5.EXIT
```

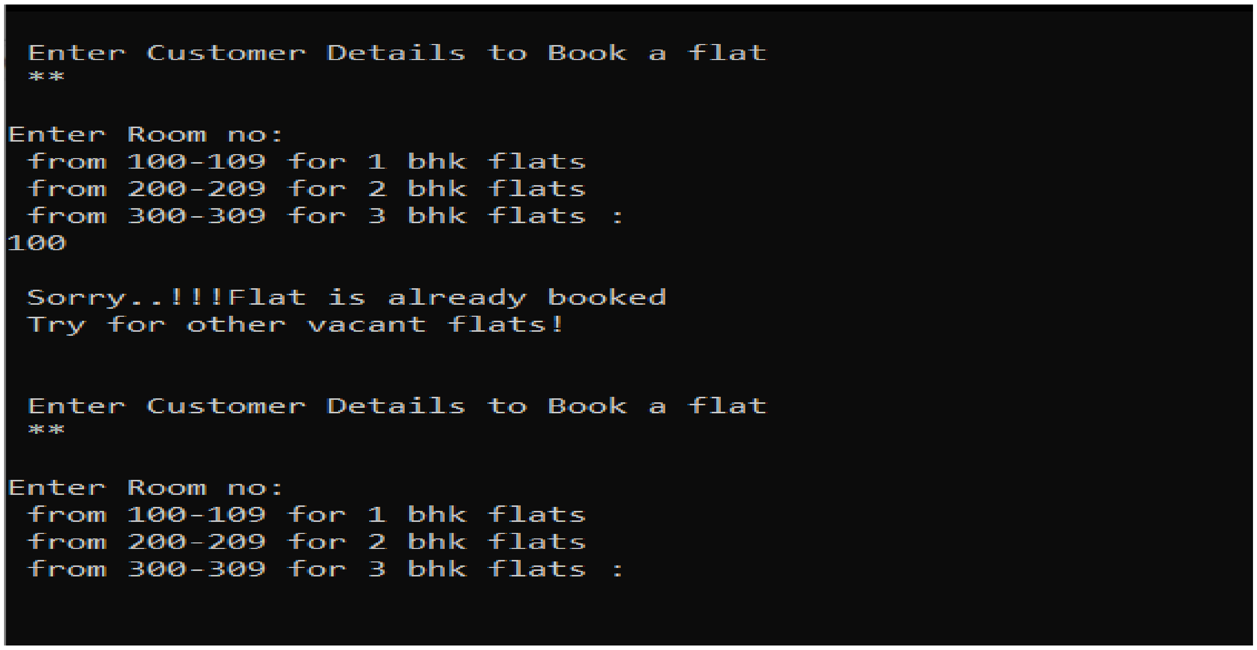
❖ flat add()

This function is used to add a new user record to our file of records using STL map.

It first checks that the room entered by the user should not be pre booked. If it is available then it takes the details from the user such as his name, permanent address and mobile number. Then it calls built in insert() function of STL map for inserting a new node for new user using iterator template.
After that it displays a message that the flat is put on hold.
We have made counters for keeping a check on number of flats which are vacant. So, this module also decrements the counter of specific flat type which is put on hold by user.



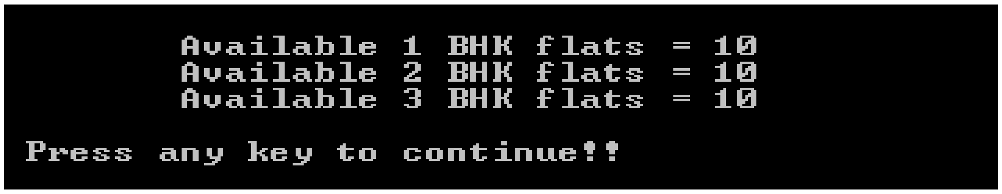
```
 Enter Customer Details to Book a flat
 **

Enter Room no:
 from 100-109 for 1 bhk flats
 from 200-209 for 2 bhk flats
 from 300-309 for 3 bhk flats :
100
Please provide your details :
First Name: Krish
Last Name: saini
 Address: 483,GautamNagar
 Phone No: 1234567890

 CONGRATS !!! Room is put on hold !!!

<---Holder Details --->
Flat Number :100
First name :Krish
Last name :saini
Address :483,GautamNagar
Phone number :1234567890
Press any key to continue!...
```



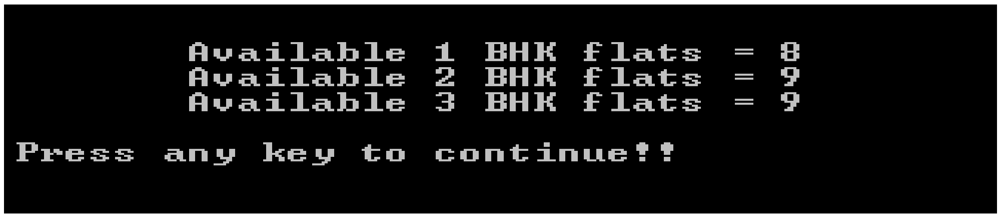
```
 Enter Customer Details to Book a flat
 **

Enter Room no:
 from 100-109 for 1 bhk flats
 from 200-209 for 2 bhk flats
 from 300-309 for 3 bhk flats :
100

 Sorry..!!!Flat is already booked
 Try for other vacant flats!


 Enter Customer Details to Book a flat
 **

Enter Room no:
 from 100-109 for 1 bhk flats
 from 200-209 for 2 bhk flats
 from 300-309 for 3 bhk flats :
```

❖ void displayvac()

This function displays the number of vacant flats of each type, i.e. 1 BHK, 2BHK or 3 BHK.
This function uses the values of counter variables which are decremented when a flat is booked, and are incremented when a flat gets vacant, i.e. when a booking gets cancelled.



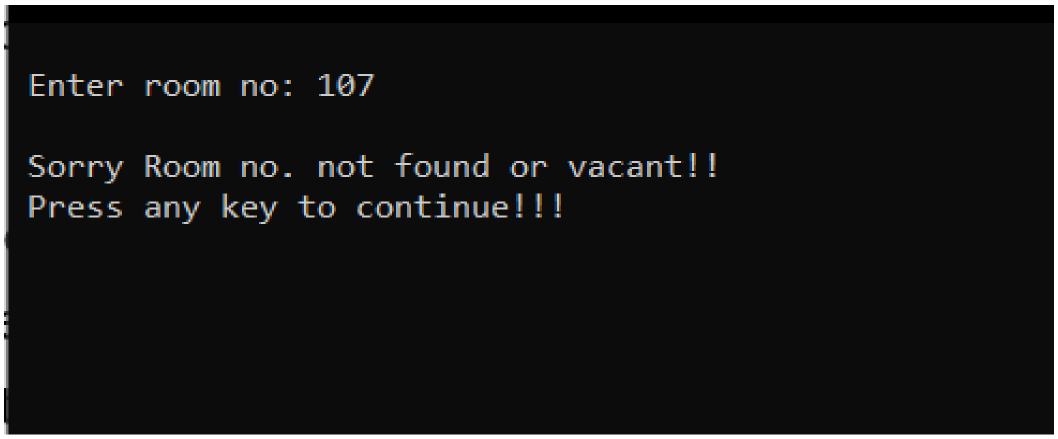
```
        Available 1 BHK flats = 10
        Available 2 BHK flats = 10
        Available 3 BHK flats = 10

 Press any key to continue!!
```



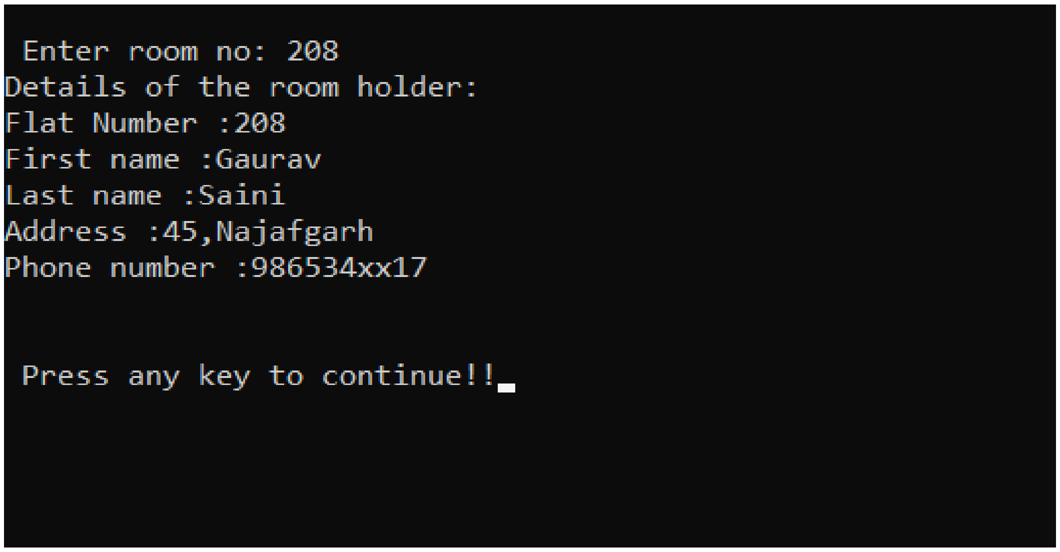
```
        Available 1 BHK flats = 8
        Available 2 BHK flats = 9
        Available 3 BHK flats = 9

Press any key to continue!!
```

❖ void display()

This function displays the record of a particular buyer. It takes the flat number from the user for which he/she wants the record, and then searches it with the records present in the file. If present, it displays the information of that buyer.



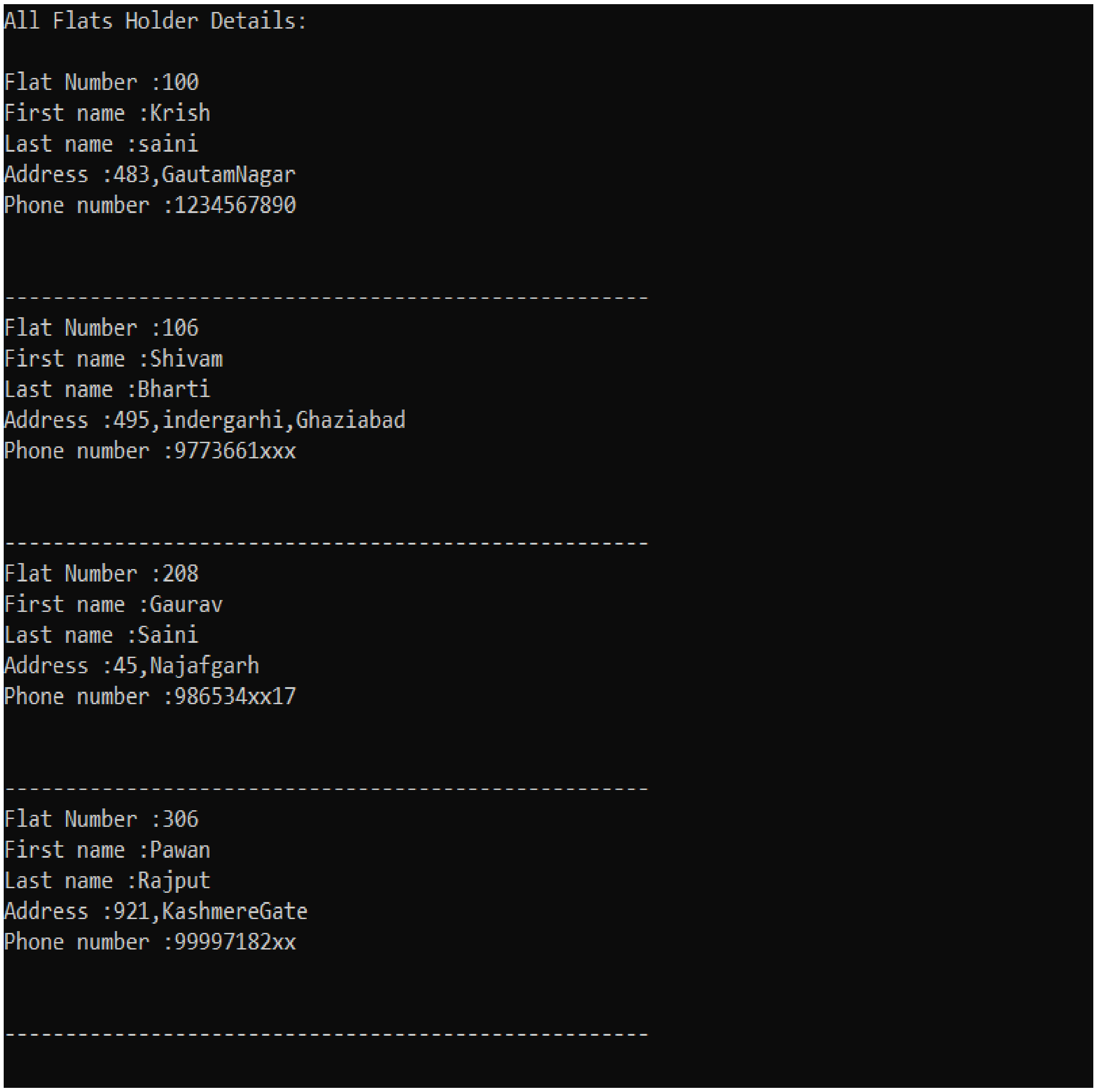
```
Enter room no: 107

Sorry Room no. not found or vacant!!
Press any key to continue!!!
```



```
 Enter room no: 208
Details of the room holder:
Flat Number :208
First name :Gaurav
Last name :Saini
Address :45,Najafgarh
Phone number :986534xx17


 Press any key to continue!!
```

❖ void rooms()

This function is used to display the details of all the buyers irrespective of their flat types. This displays the records in the sorted order of the flat numbers.

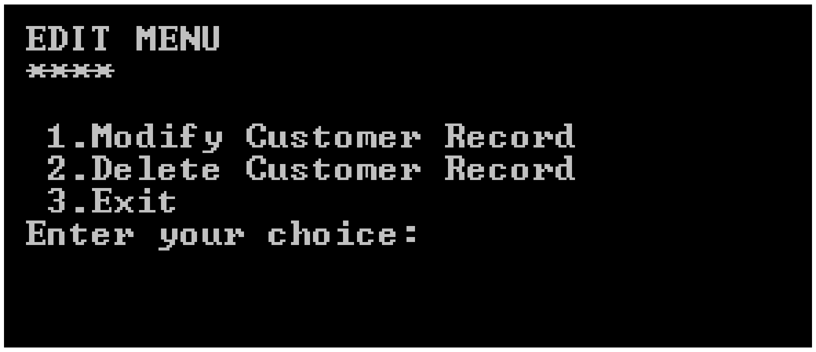


```
All Flats Holder Details:

Flat Number :100
First name :Krish
Last name :saini
Address :483,GautamNagar
Phone number :1234567890


-----------------------------------------------------
Flat Number :106
First name :Shivam
Last name :Bharti
Address :495,indergarhi,Ghaziabad
Phone number :9773661xxx


-----------------------------------------------------
Flat Number :208
First name :Gaurav
Last name :Saini
Address :45,Najafgarh
Phone number :986534xx17


-----------------------------------------------------
Flat Number :306
First name :Pawan
Last name :Rajput
Address :921,KashmereGate
Phone number :99997182xx


-----------------------------------------------------
```

❖ void edit()

This function is used to edit a particular record. This facility is provided to registered user and the owner only.

It further provides two choices to user, i.e. to modify a record or to delete a record and calls the respective functions according to the choice.



```
EDIT MENU
*****

 1.Modify Customer Record
 2.Delete Customer Record
 3.Exit
Enter your choice:
```

❖ int check(int)

This function is used to check whether entered record is present in file or not.

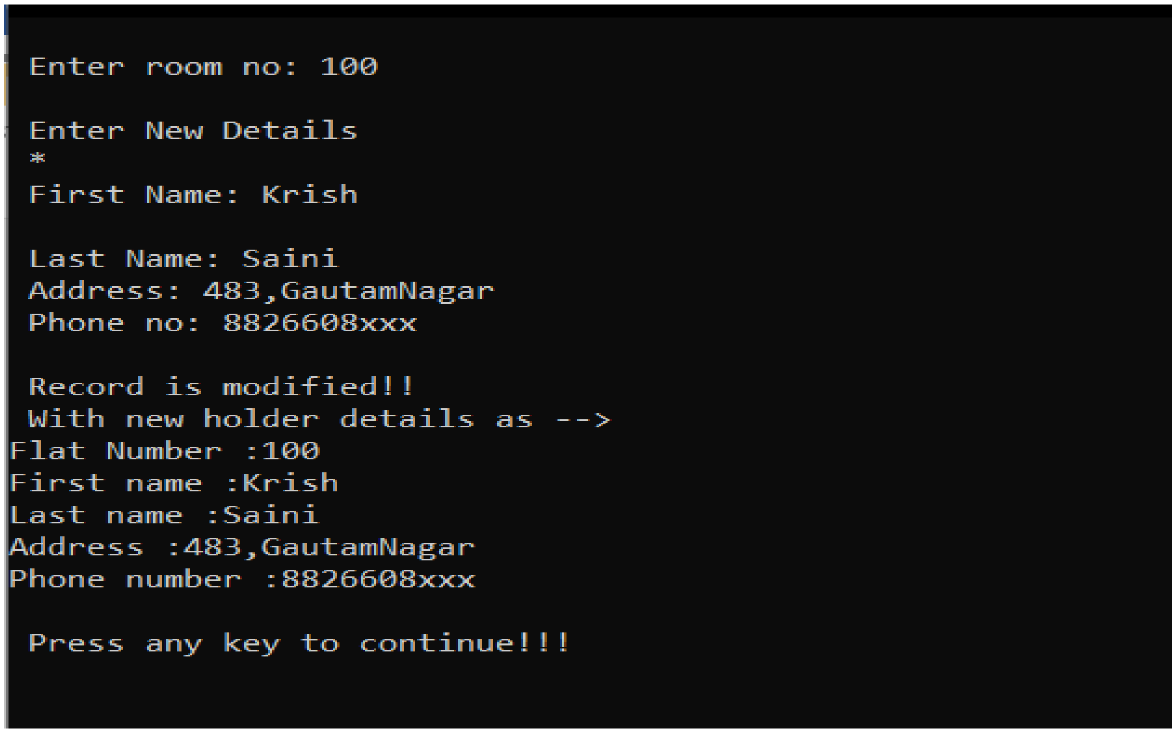
It matches the entered room number with all the room numbers using the iterator template for STL map and returns some changed variable if it is found.

This function is useful because it prevents overwriting of same room number and also shows the errors when the searched record is not present.

❖ void modify(int)

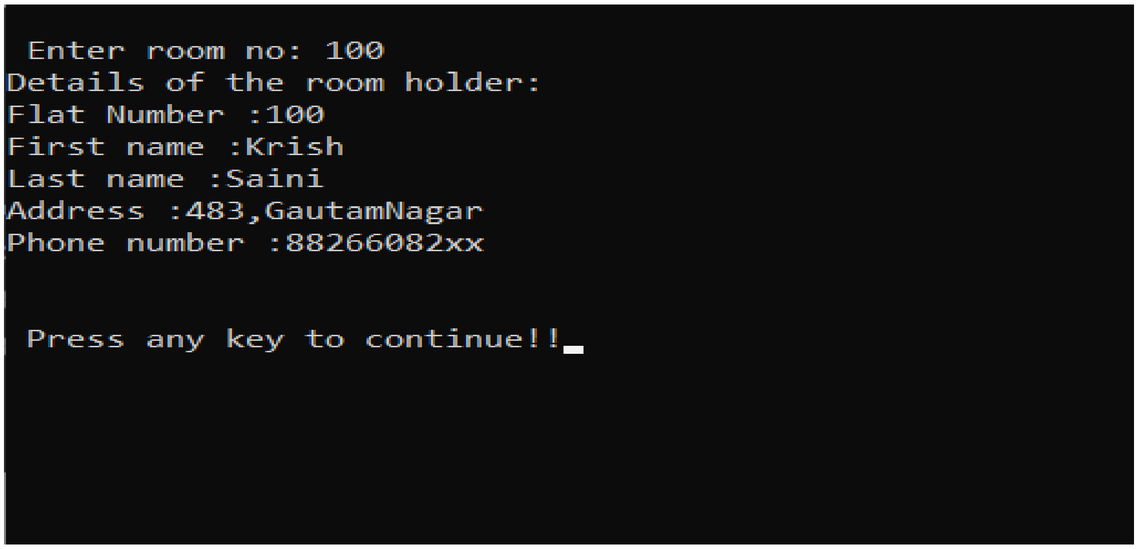
This function is used to modify the details of a particular customer. This is called inside the edit() function.

It receives a room number from the user for which he wants to modify the details. It then searches the record with same room number in the file. If found, it allows the user to overwrite the details and updates them in the file.



```
 Enter room no: 100

 Enter New Details
 *
 First Name: Krish

 Last Name: Saini
 Address: 483,GautamNagar
 Phone no: 8826608xxx

 Record is modified!!
 With new holder details as -->
Flat Number :100
First name :Krish
Last name :Saini
Address :483,GautamNagar
Phone number :8826608xxx

 Press any key to continue!!!
```

On checking particular record for flat no. 100 after editing

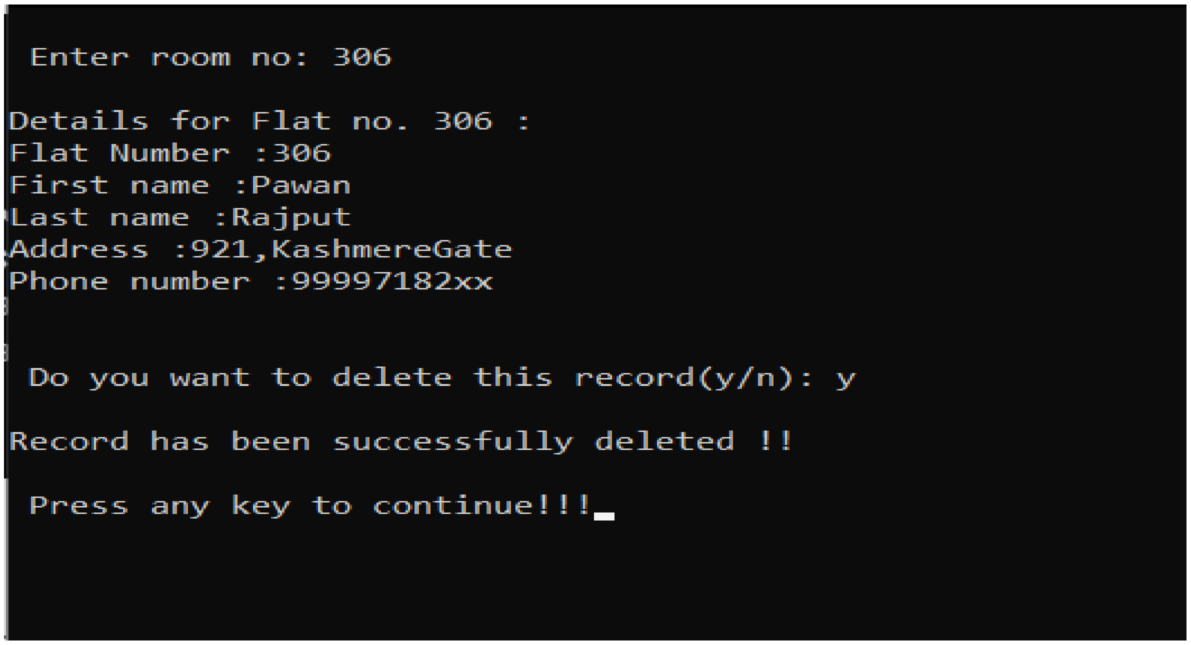


```
 Enter room no: 100
Details of the room holder:
Flat Number :100
First name :Krish
Last name :Saini
Address :483,GautamNagar
Phone number :88266082xx


 Press any key to continue!!_
```

❖ int delete_rec(int)

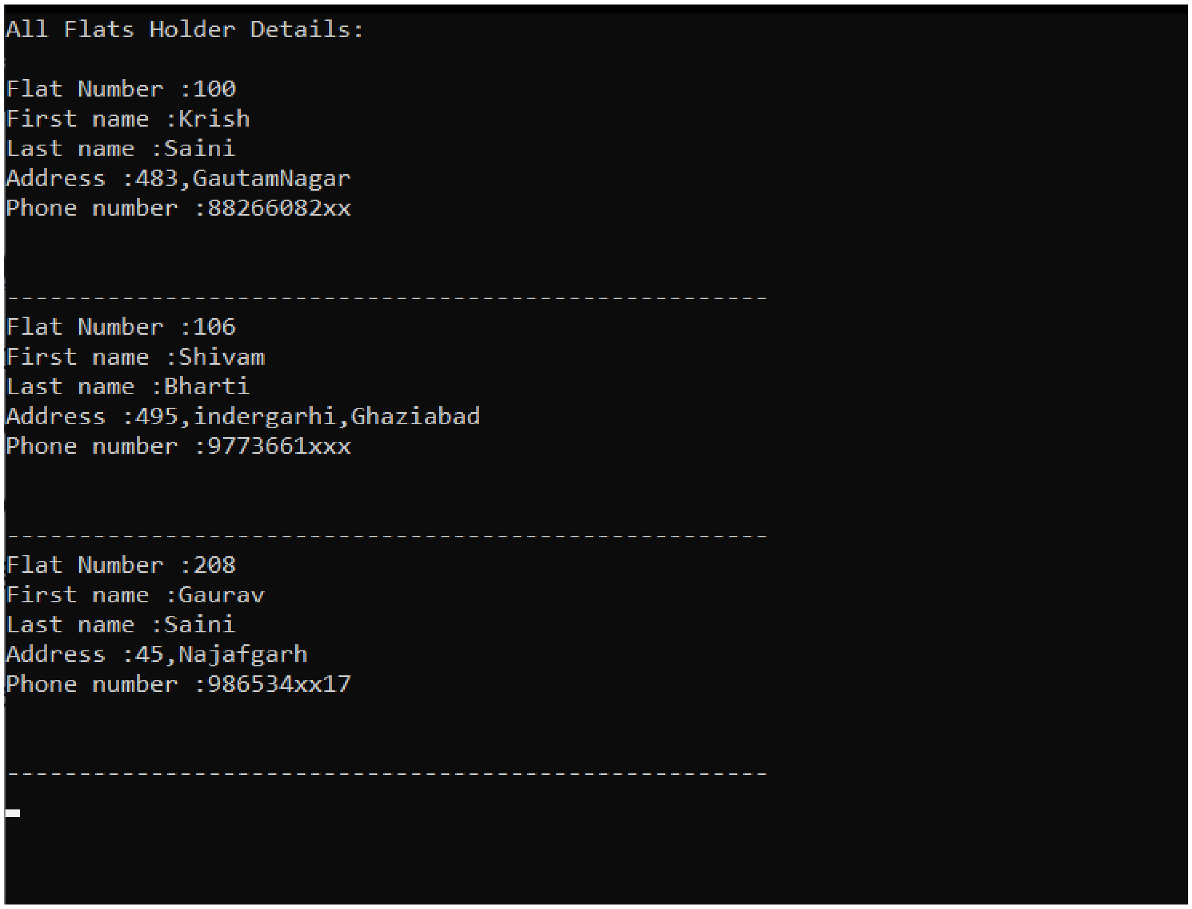
This function is used to delete the details of a particular customer. This is called inside the edit() function.

It receives a room number from the user for which he wants to delete the details. It then searches the record with same room number in the file using iterator template for STL map. If found, it allows the user to delete that record from the file using built in function erase() of map template . And then it increments the counter of that specific flat type which means one more flat of that type is vacant.



```
 Enter room no: 306

Details for Flat no. 306 :
Flat Number :306
First name :Pawan
Last name :Rajput
Address :921,KashmereGate
Phone number :99997182xx


 Do you want to delete this record(y/n): y

Record has been successfully deleted !!

 Press any key to continue!!!
```

After deleting the record then showing all record :

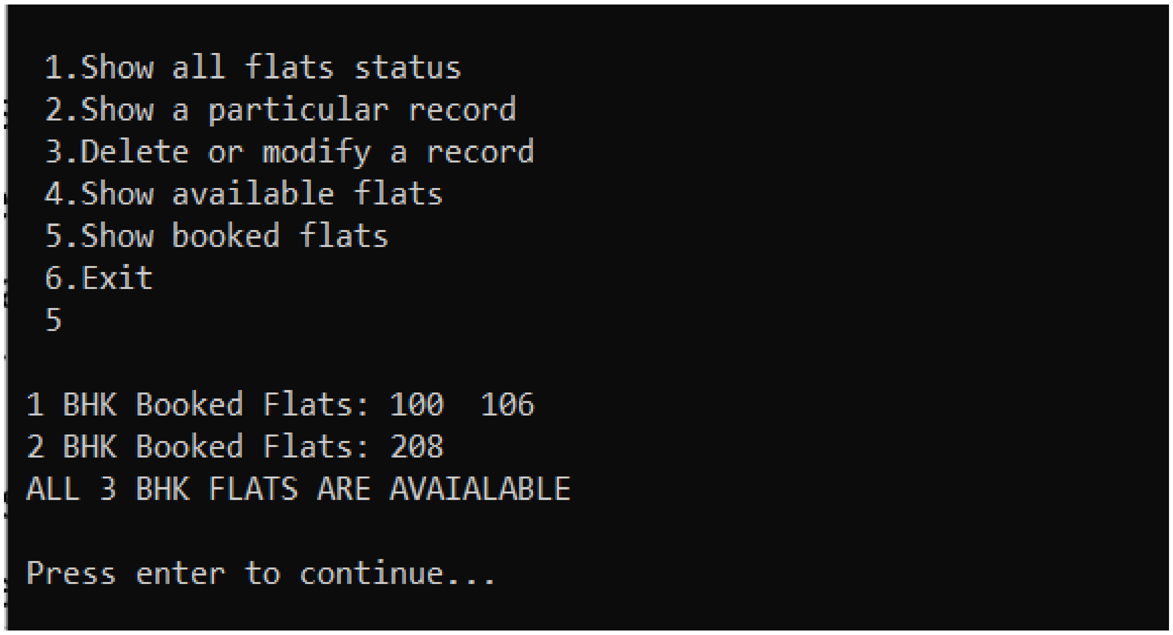


```
All Flats Holder Details:

Flat Number :100
First name :Krish
Last name :Saini
Address :483,GautamNagar
Phone number :88266082xx


----------------------------------------------------
Flat Number :106
First name :Shivam
Last name :Bharti
Address :495,indergarhi,Ghaziabad
Phone number :9773661xxx


----------------------------------------------------
Flat Number :208
First name :Gaurav
Last name :Saini
Address :45,Najafgarh
Phone number :986534xx17


----------------------------------------------------
```

❖ void displaybooked()

This function is used to display the room numbers of booked flats of each type.

This accesses the file which contains room numbers as data mapped with key using iterator template ,displays the content under specific headings by comparing the room numbers (which we get using function get_room_no() of class flat).



```
 1.Show all flats status
 2.Show a particular record
 3.Delete or modify a record
 4.Show available flats
 5.Show booked flats
 6.Exit
 5

1 BHK Booked Flats: 100  106
2 BHK Booked Flats: 208
ALL 3 BHK FLATS ARE AVAIALABLE

Press enter to continue...
```

CONCLUSION AND FUTURE WORK

It provides the simplest, cheapest and a less time and energy consuming way of dealing. In old days, the Dealer may have to keep records of the properties manually. This method of keeping the records is quiet time consuming and less efficient. Also there are more chances of mistakes by keeping the records manually as human beings are habitual of doing mistakes.

So with the help of this, Flat Dealing Software, the chances of mistakes becomes very few. Also it is very efficient method of keeping the records of flats sold as well as unsold flats. It consumes very less time as compared to manual method. The admin i.e. the Flat Dealer itself will handle the whole software. He'll manage the database as well as all the records. And also he can hand it over to the user to update his details and check out the varieties as his data is secured with password which is privately entered.

Also in future, we can add billing system in this project which as of now was a tough job to implement. Then the owner will be able to keep a record of all the money transactions as well efficiently. And then the flat will be booked only after the user pays some amount of money.

Annexure Code

```
#include<iostream>

#include<conio.h>

#include<fstream>

#include<stdio.h>

#include<string.h>

#include<map>

#include<cstdlib>


using namespace std;

int count_1 = 10;

int count_2 = 10;

int count_3 = 10;

int chance = 1;



class flat

{
```

```
private:
    int room_no;
    string fname;
    string lname;
    string address;
    string phone;



public:
    flat() {}
    flat(int room_no, string fname, string lname, string address, string
phone)
    {
        this->room_no = room_no;
        this->fname = fname;
        this->lname = lname;
        this->address = address;
        this->phone = phone;
    }

    int get_room_no() { return room_no; }
    string getfirstname() { return fname; }

    string getlastname() { return lname; }

    string getaddress() { return address; }

    string getphone() { return phone; }



    friend ofstream& operator<< (ofstream& ofs, flat& fl);

    friend ifstream& operator>> (ifstream& ifs, flat& fl);

    friend ostream& operator<< (ostream& os, flat& fl);

};

class owner

{

private:

    map<int, flat> flatss;

public:

    owner();


    flat add();         //to book a room

    void displayvac();   //to display vacant flats

    void display();      //to display the particular customer record

    void rooms();        //to display alloted rooms
    void edit();        //to edit the customer record

    int check(int);      //to check room status

    void modify(int);    //to modify the record

    int delete_rec(int); //to delete the record

    void pricelist();

    void displaybooked();

    ~owner();

};


void owner::displayvac()

{

    system("cls");

    cout << "\n\t\t SOCIETY - DWARKA ENCLAVES \nADDRESS - DWARKA
SECTOR  -24" << endl;

    cout << "\n\t" << "Available 1 BHK flats = " << count_1;

    cout << "\n\t" << "Available 2 BHK flats = " << count_2;

    cout << "\n\t" << "Available 3 BHK flats = " << count_3 << endl;

}

void owner::pricelist()

{

    cout << "\nHere's the menu of flats we are offering you : ";
    cout << "\n  Room No.\tCategory\t   Type   \t  Area(sq.
ft)\tPrice(Rs.)";

    cout << "\n  100-102\t1 BHK\t\tFront Facing\t   683\t\t30 Lac";

    cout << "\n  103-105\t1 BHK\t\tBack Facing\t   683\t\t28 Lac";

    cout << "\n  106-109\t1 BHK Corner\t 2 side open\t   703\t\t32 Lac";

    cout << "\n  200-202\t2 BHK\t\tFront Facing\t   875\t\t40 Lac";

    cout << "\n  203-205\t2 BHK\t\tBack Facing\t   875\t\t38 Lac";

    cout << "\n  206-209\t2 BHK Corner\t 2 side open\t   959\t\t45 Lac";

    cout << "\n  300-302\t3 BHK\t\tFront Facing\t   1314\t\t55 Lac";

    cout << "\n  303-305\t3 BHK\t\tBack Facing\t   1314\t\t53 Lac";

    cout << "\n  306-309\t3 BHK Corner\t 2 side open\t   1398\t\t60 Lac";

    getch();


}




flat owner::add()

{

    system("cls");

    int r, flag;

    string fn, ln, adrs, phn;
    ofstream fout;
label:
    cout << "\n Enter Customer Details to Book a flat";
    cout << "\n **";
    cout << "\n\nRoom no:\n from 100-109 for 1 bhk flats \n from 200-
209 for 2 bhk flats \n from 300-309 for 3 bhk flats : \n";
    cin >> r;
    if ((r >= 100 && r <= 109) || (r >= 200 && r <= 209) || (r >= 300 && r
<= 309))
    {

        flag = check(r);

        if (flag == 1)
        {
            cout << "\n Sorry..!!!Flat is already booked";
            cout << "\n\Try for other vacant flats!\n\n";
            goto label;
        }

        else
```

```
{
    if (r >= 100 && r <= 109)
    {
        count_1 -= 1;
    }
    if (r >= 200 && r <= 209)
    {
        count_2 -= 1;
    }
    if (r >= 300 && r <= 309)
    {
        count_3 -= 1;
    }
    cout << "First Name: ";
    cin >> fn;
    cout << "Last Name: ";
    cin >> ln;
    cout << " Address: ";
    cin >> adrs;
    cout << " Phone No: ";
    cin >> phn;
```

```
            flat acc(r, fn, ln, adrs, phn);

            flatss.insert(pair<int, flat>(acc.get_room_no(), acc));

            fout.open("Record.dat", ios::app);


            map<int, flat>::iterator itr;

            for (itr = flatss.begin(); itr != flatss.end(); itr++)

            {

                fout << itr->second;

            }

            fout.close();

            system("cls");

            cout << "\n CONGRATS !!! Room is put on hold !!!\n\n<---Holder
Details --->\n";

            return acc;

        }


        cout << "\n Press any key to continue!!";

        getch();


    }

    else
    {
        cout << "\n Wrong room number entered !!\n";
        getch();
    }

}

void owner::display()
{
    system("cls");
    int r, flag = 0;
    cout << "\n Enter room no: ";
    cin >> r;

    map<int, flat>::iterator itr = flatss.find(r);
    if (itr != flatss.end())
    {
        flag = 1;
        cout << "Details of the room holder: \n" << itr->second;
    }
    if (flag == 0)
        cout << "\n Sorry Room no. not found or vacant!!";

    cout << "\n\n Press any key to continue!!";
    getch();
}

void owner::rooms()
{
    system("cls");

    map<int, flat>::iterator itr;
    cout << "All Flats Holder Details: " << "\n\n";
    for (itr = flatss.begin(); itr != flatss.end(); itr++)
    {
        cout << itr->second << "\n";
        cout << "\n----------------------------------------------------\n";
    }
    getch();
}
```

```
int owner::check(int r)
{
    map<int, flat>::iterator itr = flatss.find(r);
    if (itr != flatss.end())
    {
        return 1;
    }
    return 0;
}

void owner::edit()
{
    system("cls");
    int choice, r;

    cout << "\n EDIT MENU";
    cout << "\n *";
    cout << "\n\n  1.Modify Customer Record";
    cout << "\n  2.Delete Customer Record";
    cout << "\n  3.Exit";
    cout << "\n Enter your choice: ";
    cin >> choice;

    system("cls");


    switch (choice)
    {
    case 1:cout << "\n Enter room no: ";
        cin >> r;
        modify(r);
        break;
    case 2:cout << "\n Enter room no: ";
        cin >> r;
        int c;
        c = delete_rec(r);
        if (c == 1)
        {
            cout << "\nRecord has been successfully deleted !!\n";
        }
        break;
    case 3: break;

    default: cout << "\n Wrong Choice!!";

    }


    cout << "\n Press any key to continue!!!";

    getch();

}


void owner::modify(int r)

{

    string fn, ln, add, phn;

    int flag = 0;

    map<int, flat>::iterator itr = flatss.find(r);

    if (itr != flatss.end())

    {

        cout << "\n Enter New Details";

        cout << "\n *";

        cout << "\n First Name: ";

        cin >> fn;

        cout << "\n Last Name: ";

        cin >> ln;
        cout << " Address: ";

        cin >> add;

        cout << " Phone no: ";

        cin >> phn;


        flat acc(r, fn, ln, add, phn);

        itr->second = acc;

        cout << "\n Record is modified!!\n With new holder details as --
>\n";

        flag = 1;

        cout << acc;

    }

    if (flag == 0)

        cout << "\n Sorry Room no. not found or vacant!!";

}


int owner::delete_rec(int r)

{

    int flag = 0;

    char ch;

    map<int, flat>::iterator itr = flatss.find(r);
    if (itr != flatss.end())

    {

        cout << endl << "Details for Flat no. " << itr->first << " :\n" << itr-
>second;

        flag = 1;

    }


    if (flag == 1)

    {

        cout << "\n\n Do you want to delete this record(y/n): ";

        cin >> ch;


        if (ch == 'n' || ch == 'N')

        {

            cout << "Okay!Exiting..... ";

            return 0;

        }

        else if (ch == 'y' || ch == 'Y')

        {

            flatss.erase(r);

            if (r >= 100 && r <= 109)
            {
                count_1 += 1;
            }
            if (r >= 200 && r <= 209)
            {
                count_2 += 1;
            }
            if (r >= 300 && r <= 309)
            {
                count_3 += 1;
            }
            return 1;
        }
    }
    else
    {
        cout << "Flat is vaccant till now! ";
    }

}
void owner::displaybooked()
```

```
{
    int ch_1 = 0, ch_2 = 0, ch_3 = 0;
    int checkk_1 = 0, checkk_2 = 0, checkk_3 = 0;

    map<int, flat>::iterator itr;
    for (itr = flatss.begin(); itr != flatss.end(); itr++)
    {
        if (itr->second.get_room_no() >= 100 && itr-
>second.get_room_no() <= 109)
        {
            if (ch_1 == 0)
            {
                cout << "\n 1 BHK Booked Flats:";
                ch_1 = 1;
            }
            cout << " " << itr->second.get_room_no() << " ";
            checkk_1++;
        }
        if (itr->second.get_room_no() >= 200 && itr-
>second.get_room_no() <= 209)
        {
            if (ch_2 == 0)

            {

                cout << "\n 2 BHK Booked Flats:";

                ch_2 = 1;

            }

            cout << " " << itr->second.get_room_no() << " ";

            checkk_2++;

        }

        if (itr->second.get_room_no() >= 300 && itr-
>second.get_room_no() <= 309)

        {

            if (ch_3 == 0)

            {

                cout << "\n 3 BHK Booked Flats:";

                ch_3 = 1;

            }

            cout << " " << itr->second.get_room_no() << " ";

            checkk_3++;

        }


    }
    if (checkk_1 == 0)cout << "\n ALL 1 BHK FLATS ARE AVAIALABLE";

    if (checkk_2 == 0)cout << "\n ALL 2 BHK FLATS ARE AVAIALABLE";

    if (checkk_3 == 0)cout << "\n ALL 3 BHK FLATS ARE AVAIALABLE\n";

    cout << "\n Press enter to continue...\n";

    getch();


}



ofstream& operator<<(ofstream& ofs, flat& fl)

{

    ofs << fl.room_no << endl;

    ofs << fl.fname << endl;

    ofs << fl.lname << endl;

    ofs << fl.address << endl;

    ofs << fl.phone << endl;

    return ofs;

}

ifstream& operator>>(ifstream& ifs, flat& fl)

{
    ifs >> fl.room_no;
    ifs >> fl.fname;
    ifs >> fl.lname;
    ifs >> fl.address;
    ifs >> fl.phone;
    return ifs;

}
ostream& operator<<(ostream& os, flat& fl)
{
    os << "Flat Number :" << fl.get_room_no() << endl;
    os << "First name :" << fl.getfirstname() << endl;
    os << "Last name :" << fl.getlastname() << endl;
    os << "Address :" << fl.getaddress() << endl;
    os << "Phone number :" << fl.getphone() << endl;
    return os;
}


owner::owner()
{
    flat acc;
    ifstream infile("Record.dat", ios::in);
    if (!infile)
    {
        //cout<<"Error in Opening! File Not Found!!"<<endl;
        return;
    }
    while (!infile.eof())
    {
        infile >> acc;
        flatss.insert(pair<int, flat>(acc.get_room_no(), acc));
    }
    infile.close();
}

owner::~owner()
{
    ofstream outfile;
    outfile.open("Record.dat", ios::trunc);
    map<int, flat>::iterator itr;
    for (itr = flatss.begin(); itr != flatss.end(); itr++)
    {
        outfile << itr->second;
    }
    outfile.close();
}



int main()
{
    flat f;
    owner own;
    system("cls");
    cout << "\n\t\t\t************************";
    cout << "\n\t\t\t* FLAT DEALING PROJECT *";
    cout << "\n\t\t\t************************";

    cout << "\n\n\n\n\t\tMade By:";
    cout << "\n\t\t Shivam Bharti -  2k19/SE/115 ";
    cout << "\n\t\t Pawan         -  2k19/SE/083 ";
    cout << "\n\n\n\n\n\t\t\t\tPress any key to continue!!";
    getch();
    system("cls");

    cout << endl << endl << "\t \t |*WELCOME TO SB&P FLAT
DEALERS*|" << endl << endl;

    cout << endl;

    int n, p, re, o;

    while (n != 4)

    {

        cout << "  Please tell us about yourself:\n 1.NEW USER \n
2.REGISTERED USER  \n 3.OWNER \n 4.QUIT" << endl;

        cin >> n;

        switch (n)

        {

        case 1:
                                                                         syste
m("cls");

            cout << "\t WELCOME SIR/MAM" << endl;

            cout << "\n  HOW CAN WE HELP YOU " << endl;


            do

            {

                cout << "\n\n\t PLEASE SELECT AMONG THESE :" << endl <<
endl;
```

```
cout << "\n  1.DISPLAY VACANT FLAT INFORMATION";

cout << "\n  2.DISPLAY PRICE MENU OF ALL FLATS";

cout << "\n  3.BOOK AN AVAILABLE FLAT";

cout << "\n  4.EXIT" << endl;

cout << "  ";

cin >> p;

cout << endl;

switch (p)

{

case 1: own.displayvac();

    cout << "\n Press any key to continue!!";

    getch();

    break;

case 2: own.pricelist();

    break;

case 3: f = own.add();

    cout << f;

    break;

case 4: cout << "\n  THANKS FOR VISITING\n" << endl;

    break;
```

```
            default: cout << "SORRY !, YOU MADE A WRONNG CHOICE " <<
endl;
              break;


            }


          } while (p != 4);


          break;
        case 2:
                                                                         syste
m("cls");
          cout << "\n  WELCOME AGAIN SIR/MAM" << endl;
          cout << "\n GLAD TO SEE YOU" << endl;
          do
          {
            cout << "\n\n\t Please select your choice" << endl << endl;
            cout << "  1.Edit Details";
            cout << "\n  2.Display Details";
            cout << "\n  3.Book A New Flat";
            cout << "\n  4.Exit";
            cout << "\n  ";

            cin >> re;

            switch (re)

            {

            case 1: own.edit();

              break;

            case 2: own.display();

              break;

            case 3: f = own.add();

              cout << f;

              break;

            case 4: cout << "\n  THANKS FOR VISITING" << endl;

              break;

            default: cout << "SORRY !, YOU MADE A WRONNG CHOICE " <<
endl;

              break;

            }

          } while (re != 4);

          break;

        case 3: {system("cls");

          int i;
```

```
char ch1, password[10];

cout << "Enter your password : ";

for (i = 0; i < 10; i++)

{

    ch1 = getch();

    password[i] = ch1;


    if (ch1 != 13)      //13 is ASCII of Enter key

        printf("*");

    if (ch1 == 13)

        break;

}

password[i] = '\0';

if ((strcmp(password, "031929")) == 0)

{

    system("cls");

    cout << "Password matched :)";

    cout << "\n\n\tWHAT DO YOU WANT SIR ? " << endl << endl;

    do

    {

        cout << "\n  1.Show all flats status";
cout << "\n  2.Show a particular record";
cout << "\n  3.Delete or modify a record";
cout << "\n  4.Show available flats";
cout << "\n  5.Show booked flats ";
cout << "\n  6.Exit";
cout << "\n  ";
cin >> o;
switch (o)
{
case 1: own.rooms();
   break;
case 2: own.display();
   break;
case 3: own.edit();
   break;
case 4: own.displayvac();
   break;
case 5: own.displaybooked();
   break;
case 6: break;
```

```
                    default: cout << "\n SORRY, YOU MADE A WRONG
CHOICE!!";
                      break;
                    }
                  } while (o != 6);


              }
              else
                cout << "\n Wrong password Entered! ";
              break;
            }
            case 4: cout << "\n  THANKS FOR VISITING " << endl;
              break;
            default: cout << "SORRY !, YOU MADE A WRONG CHOICE " << endl;
              break;
     }
     }
       getch();
      return 0;
       }
```

REFERENCES

- ➢ https://mtnlhousing.org/upcoming23.php
- ➢ https://www.magicbricks.com/property-for-sale
- ➢ https://www.geeksforgeeks.org/
- ➢ http://www.cplusplus.com/reference/map/map/
- ➢ https://www.programiz.com/cpp-programming/operator-overloading
- ➢ https://www.cprogramming.com/tutorial/stl/iterators.html
- ➢ https://www.tutorialspoint.com/cplusplus/cpp_friend_functions.htm

RESEARCH PAPER INCLUDED:

[1] Wei Wei, Zhengwei Lou. "Design and Implementation of Hotel Room Management System". School of Computer Science and Engineering, Xi'an University of Technology. 2019 IEEE Symposium Series on Computational Intelligence (SSCI) December 6-9 2019.

[2] Ashwin Urdhwareshe. "Object-Oriented Programming and its Concepts". Department of Technology Management, University of Bridgeport, Bridgeport, CT 06604, USA. International Journal of Innovation and Scientific Research ISSN 2351-8014 Vol. 26 No. 1 Aug. 2016.