SVKM's NMIMS

School of Technology Management & Engineering,

Navi Mumbai

OBJECT-ORIENTED PROGRAMMING

SOCIETY MANAGEMENT SYSTEM

B. TECH - SEM. 3RD

(Computer Science & Business System)

CONTENTS

INTRODUCTION

Most cooperative societies and apartments have multiple buildings with many residents. Managing these societies can be a time consuming and mammoth task for the administration committee. However, these duties are essential for assuring the safety and convenience of residents. Society management apps go a long way in ensuring the smooth functioning of a building society and improving efficiency while reducing the committee's workload and manpower.

PROBLEM DEFINITION

Society matters involve a huge amount of paperwork for things like guest entry, staff entry, hall booking etc and a wide range of funds – maintenance charges, sinking fund, reserve fund, repair charges, working capital, etc. Maintaining and managing separate physical ledgers can be strenuous, tedious and prone to error. This is where we saw an immediate need for migration from paperwork to digitalised data handling.

Lack of transparency can lead to suspicion and disrupt harmony in society. Hence a portal that allows users to put all records regarding the society on the platform. And admins can permit residents to view all details, including purchases, repairs and maintenance, updates, etc. Therefore, everything gets uploaded on the cloud, no data is ever lost, and the trust of residents prevails.

We need to come up with a portal that provides safety, reliability and will help to store and access data for a huge span of time. Also, we need to come up with an interface that is beginner-friendly.

PROJECT DESCRIPTION

Many society applications are not good enough because they lack proper orientation. Even the data is cluttered. Our society management project helps the user to execute simple yet time-consuming tasks with just one click of the finger. Our project is designed in a simplistic manner that is user-friendly.

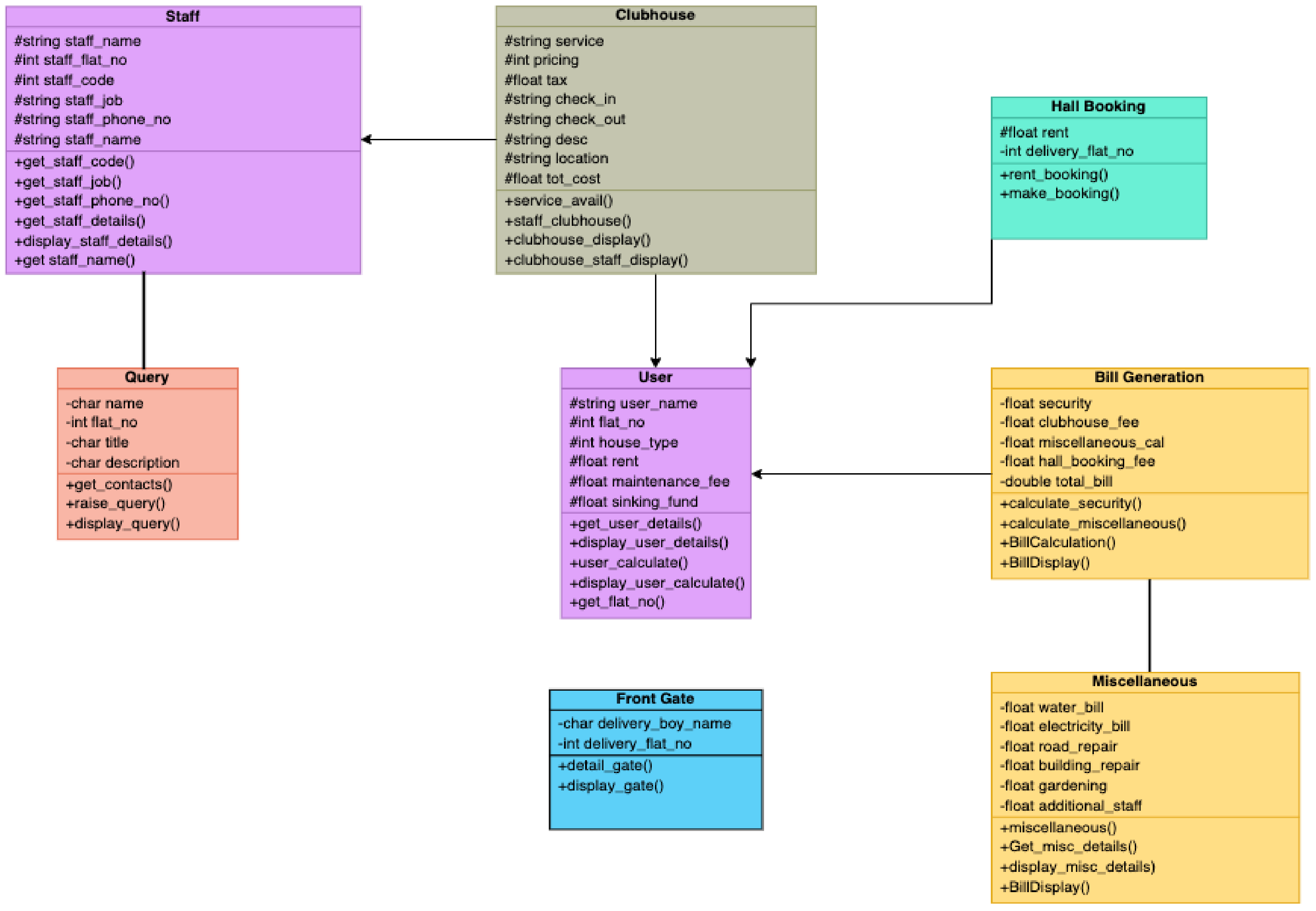
It allows the user to browse various functions of society such as accessing the clubhouse, booking a hall, and the most important one- to raise a complaint, if any. We also have the functionality of storing the front gate delivery records. These records can be displayed later or as and when required.

As mentioned earlier, in the clubhouse, the user can pick their favourite sports amenities for which is present in the society. Now, to maintain these clubhouse amenities, society needs maintenance staff. The clubhouse also stores the check-in and check-out timings of the staff.

Our hall management function allows residents and non-residents to book the halls for various time durations at different prices. The most important function of our society management system is the query function. Through this, the residents can access the contact details of the staff they need. This staff data includes the contact information of maids, plumbers and electricians. The query module also lets all the users file a complaint regarding society management if any.

All of this is simply coded in c++.

CLASS DIAGRAM



DETAILS OF PROJECT

MODULES OF CODE

STAFF MODULE:
It will allow the staff to enter their details like name, staff code, job, flat number, phone number and display all the details.

USER MODULE:

It will take the data such as the flat no, flat details of the members of the society. It will store the data in the society app. It also calculates the financial part of the society such as providing maintenance fees and sinking fees. The user module plays an important role in society management, it is linked to the clubhouse, front gate, bill generation, hall booking and query management modules.

CLUBHOUSE MODULE:
It allows the user of society to choose their clubhouse facility which they will be using and generate a bill according to it. For staff, it allows them to track their records along with check-in and check-out time.

FRONT GATE MODULE:
A systematic record of guest entry at the main gate. Uses the concept of file handling

BILL GENERATE MODULE:
Generates a final bill with the following break up:

- Receives user fee from the user module
  This includes the rent, maintenance, and sinking fund.
- Receives miscellaneous charges from the friend class miscellaneous.
  This includes water bill, electricity bill, road repair charges, building repair charges, gardening charges and payment of additional society staff
- Amount paid for hiring security for the society
- Charges to be paid for the clubhouse
- Charges to be paid if hall booking is done

  MISCELLANEOUS:
  This class is a friend class of the BILL GENERATE.
  It is used for calculating miscellaneous charges. This includes:
  - Water bill
  - Electricity bill

- Road repair charges
- Building repair charges
- Gardening charges
- Payment of additional society staff

HALL BOOKING MODULE:
It allows the society to rent out the society lawn to members and outsiders for functions.
Functions under this:

- Rates for rent
- Booking records

QUERY MODULE:
It allows the members to put out their grievances and to seek remedy for the same. It allows them to access important information on society's functioning.

- Contact of society help
- Grievances in general

TECHNIQUES USED

- **HYBRID INHERITANCE:**
  - There exist period a beginner-friendly interface multiple **inheritances** between user class, Bill Generate class, clubhouse class and Hall Booking class. The user class is the derived class and the others are the base classes.
  - There exists a **single level inheritance** between the Staff class and the Clubhouse class. Staff is the derived class and the Clubhouse is the base class.

- **FRIEND CLASS:**
  Bill Generate is the friend class of Miscellaneous class which means that it can now access the private members of the miscellaneous class.

- **CONSTRUCTOR AND DESTRUCTOR**
  This concept is used in the miscellaneous class to have a default value for some miscellaneous expenses.

- **FILE HANDLING:**
  File handling was incorporated in 2 classes:
  - **Front gate module:**
    Here, the details accepted at the front gate will be written in the file to have proper records for the same.
  - **Query Management module:**
    File handling is used here to store the grievances of the users.

- **PARAMETERIZED FUNCTIONS**
  One of the calculate functions in the miscellaneous class was parameterized.

- **ENCAPSULATION**
  The project actively used classes thereby applying the concept of encapsulation.

- **DATA HIDING**
  The data members and functions were written keeping in mind how and from where they need to be accessed.

TIMELINE

0 WEEK	16TH Aug-23RD Aug	Explore various domains and pick a suitable topic for the project Identify the concepts to be used and utilise this week to study them and get familiar with them
1 WEEK	23RD Aug-30TH Aug	Staff module
2 WEEK	30TH Aug-6TH Sept	Front gate security system module
3 WEEK	6TH Sept-13TH Sept	Clubhouse module
4 WEEK	13TH Sept-20TH Sept	Query management module
5 WEEK	20TH Sept-27TH Sept	Hall Booking module Bill Generation Module
6 WEEK	27TH Sept-4TH Oct	Bill Generation Module
7 WEEK	[4TH Oct-11TH Oct	Compilation of various files containing different functions to make the final project. Check for the coherence of the functions. Minor debugging of the program.
8 WEEK	11TH Oct-16TH Oct	Buffer week

PROJECT CODE

OOP_FRONTGATE.CPP:
Contains the Front Gate Module.

```cpp
#include <iostream>
using namespace std;

class Front_gate
{
    private:
        char delivery_boy_name[50];
        int delivery_flat_no;
    public:
        void detail_gate(); //used to get details of delivery person
        void display_gate(); //display details of delivery person
};

void Front_gate::detail_gate()
{
    cout<<"\nEnter delivery person's name: ";
    cin.ignore();
    cin.getline(delivery_boy_name,50);
    cout<<"Enter flat number: ";
    cin>>delivery_flat_no;
}

void Front_gate::display_gate()
{
    cout<<"\nDelivery person's name: "<<delivery_boy_name;
    cout<<"\nFlat number: "<<delivery_flat_no;
}
```

OOP_USERS.CPP:
Contains the User and Staff modules.

```
#include<iostream>
#include "oop_bill.cpp"
#include "oop_clubhouse.cpp"
#include "oop_hallbooking.cpp"
//including files to access class declarations given in other files
using namespace std;

class user:public BillGenerate,public clubhouse,public HallBooking
//inheriting multiple classes
{
    protected:
        string user_name;
        int flat_no;
        int house_type;
        float rent;
        float maintenance_fee;
        float sinking_fund;
    public:
        void get_user_details();
        void display_user_details();
        float user_calculate();
        void display_user_calculate();
        int get_flat_no()
        {
            return flat_no;
        };
};

void user::get_user_details()
{
    cout<<"\nEnter your name: ";
    cin.ignore();
    getline(cin,user_name);
    cout<<"Enter Flat number: ";
    cin>>flat_no;
    cout<<"House type:";
    cout<<"\n1. 1 BHK";
    cout<<"\n2. 2 BHK";
    cout<<"\n3. 3 BHK";
    cout<<"\n4. 3.5 BHK";
    cout<<"\n5. Row house";
    cout<<"\nEnter choice: ";
    cin>>house_type;

}

float user::user_calculate()
{
    float house_fee;
    if (house_type==1)
    {
        rent=15000;
        maintenance_fee=3500;
        sinking_fund=2000;
    }
    else if(house_type==2)
    {
        rent=30000;
        maintenance_fee=5000;
        sinking_fund=4000;
    }
    else if (house_type==3)
    {
        rent=45000;
        maintenance_fee=7000;
        sinking_fund=6000;
    }
    else
    {
        rent=60000;
        maintenance_fee=9000;
        sinking_fund=8000;
    }
    house_fee=rent+maintenance_fee+sinking_fund;
    return house_fee;
}
```

```
void user::display_user_calculate()
{
    cout<<"\nRent: "<<rent;
    cout<<"\nMaintenance fee: "<<maintenance_fee;
    cout<<"\nSinking fund: "<<sinking_fund;
}

void user::display_user_details()
{
    cout<<"\nName: "<<user_name;
    cout<<"\nFlat No. "<<flat_no;
    switch(house_type)
    {
        case 1:
            cout<<"\nHouse type is: 1 BHK";
            break;
        case 2:
            cout<<"\nHouse type is: 2 BHK";
            break;
        case 3:
            cout<<"\nHouse type is: 3 BHK";
            break;
        case 4:
            cout<<"\nHouse type is: 3.5 BHK";
            break;
        case 5:
            cout<<"\nHouse type is: Row House";
            break;
    }
}

class staff:public clubhouse
{
    protected:
        string staff_name;
        int staff_flat_no;
        int staff_code;
        string staff_job;
        string staff_phone_no;
    public:
        int get_staff_code()
        {
            return staff_code;
        }

        string get_staff_job()
        {
            return staff_job;
        }

        string get_staff_phone()
        {
            return staff_phone_no;
        }

        string get_staff_name()
        {
            return staff_name;
        }

        void get_staff_details()
        {
            cout<<"\nEnter staff name: ";
            cin.ignore();
            getline(cin,staff_name);
            cout<<"Enter Flat number: ";
            cin>>staff_flat_no;
            cout<<"Enter staff code: ";
            cin>>staff_code;
            cout<<"Enter staff job: ";
            cin.ignore();
            getline(cin,staff_job);
            cout<<"Enter staff contact number: ";
            cin>>staff_phone_no;
        }

        void display_staff_details()
        {
            cout<<"\nName: "<<staff_name;
            cout<<"\nFlat No. "<<staff_flat_no;
            cout<<"\nStaff code: "<<staff_code;
            cout<<"\nStaff job: "<<staff_job;
            cout<<"\nStaff Contact no: "<<staff_phone_no<<endl;
        }
};
```

OOP_CLUBHOUSE.CPP:
Contains the Clubhouse Module.

```
#include <iostream>
using namespace std;

class clubhouse
{
    protected:
        int choice;
        string service;
        int pricing;
        float tax;
        string check_in;
        string check_out;
        string desc;
        string location;
        float tot_cost;

    public:
        float service_avail(); //used by user to avail clubhouse service
        void staff_clubhouse(); //used to trace clubhouse staff
        void clubhouse_display(); //used to display clubhouse charges
        void clubhouse_staff_display();

};

float clubhouse::service_avail()
{
    cout<<"\n\tSERVICES";
    cout<<"\n0. Exit";
    cout<<"\n1. Pool Table";
    cout<<"\n2. Air Hockey";
    cout<<"\n3. Carrom";
    cout<<"\n4. Swimming Pool";
    cout<<"\n5. Gym Center";
    cout<<"\nEnter Your Choice:";
    cin>>choice;
    switch(choice)
    {
        case 0:
        {
            cout<<"Exiting...";
            break;
        }
        case 1:
        {
            service="Pool Table";
            pricing=200;
            tax=0.07*pricing;
            break;
        }
        case 2:
        {
            service="Air Hockey";
            pricing=200;
            tax=0.07*pricing;
            break;
        }
        case 3:
        {
            service="Carrom";
            pricing=100;
            tax=0.07*pricing;
            break;
        }
        case 4:
        {
            service="Swimming Pool";
            pricing=500;
            tax=0.07*pricing;
            break;
        }
        case 5:
        {
            service="Gym Center";
            pricing=1000;
            tax=0.07*pricing;
            break;
        }
    }
    cout<<"\nChoice recorded successfully"<<endl;
    tot_cost=pricing+tax; //used to store clubhouse fee

    return tot_cost; //needs to be returned in total calculation
}

void clubhouse::clubhouse_display()
{
    cout<<"\nService: "<<service;
    cout<<"\nPrice: "<<pricing;
    cout<<"\nTaxes Applicable(7%): "<<tax;
    cout<<"\nTotal Cost: "<<tot_cost;
}

void clubhouse::staff_clubhouse()
{
    cout<<"\nLocation: ";
    cin.ignore();
    getline(cin,location);
    cout<<"Check In Time: ";
    getline(cin,check_in);
    cout<<"Check Out Time: ";
    getline(cin,check_out);
    cout<<"Description: ";
    getline(cin,desc);
}

void clubhouse::clubhouse_staff_display()
{
    cout<<"\nLocation: "<<location;
    cout<<"\nCheck In Time: "<<check_in;
    cout<<"\nCheck Out Time: "<<check_out;
    cout<<"\nDescription: "<<desc<<endl;
}
```

OOP_HALLBOOKING.CPP:
Contains the Hall Booking Module.

```
#include<iostream>
using namespace std;

class HallBooking
{
    protected:
        float rent;
    public:
        void rent_booking(); //to show the rent values
        float make_booking(int);

};

void HallBooking::rent_booking()
{
    cout<<"\nAvailable packages: ";
    cout<<"\n1. 2 hours - Rs3000";
    cout<<"\n2. 3 hours - Rs.5000";
    cout<<"\n3. 5 hours - Rs.7000";
    cout<<"\n4. 7 hours - Rs.9000";
    cout<<"\n5. 9 hours - Rs.11000";
    cout<<"\n6. 24 hours - Rs.13000";
}

float HallBooking::make_booking(int flag) //flag sees if the user wants to
book the hall
{
    rent=0;
    if(flag==1)
    {
    int choice;
    cout<<"\nEnter the plan you want to pick: ";
    cin>>choice;
    switch(choice)
    {
        case 1:
            rent=3000;
            break;
        case 2:
            rent=5000;
            break;
        case 3:
            rent=7000;
            break;
        case 4:
            rent=9000;
            break;
        case 5:
            rent=11000;
            break;
        case 6:
            rent=13000;
            break;
        default:
            cout<<"\nInvalid package.";
    }
    cout<<"The hall is successfuly booked."<<endl;
    }
    return rent;
}
```

OOP_BILL.CPP:
Contains the bill Generation Module.

```
#include <iostream>
using namespace std;

class miscellaneous
{
    private:
        float water_bill;
        float electricity_bill;
        float road_repair;
        float building_repair;
        float gardening;
        float additional_staff;

    public:
        friend class BillGenerate; //BillGenerate can now access private
members of miscellaneous class
        miscellaneous() //constructor to give intial values
        {
            water_bill=0;
            electricity_bill=0;
            road_repair=1000;
            building_repair=5000;
            gardening=1500;
            additional_staff=7000;
        }
        void Get_misc_details()
        {
            cout<<"\nEnter the water bill: ";
            cin>>water_bill;
            cout<<"Enter the electricity bill: ";
            cin>>electricity_bill;
        }
        void display_misc_details()
        {
            cout<<"\nWater bill: "<<water_bill;
            cout<<"\nElectricity bill: "<<electricity_bill;
            cout<<"\nRoad repair costs: "<<road_repair;
            cout<<"\nBuilding repair costs: "<<building_repair;
            cout<<"\nCost of gardening: "<<gardening;
            cout<<"\nPayment of additional staff: "<<additional_staff;
        }
    ~miscellaneous() //destructor
    {
        ;
    };
};
```

```
class BillGenerate //used to generate bill
{
    private:
        float security;
        float clubhouse_fee;
        float miscellaneous_calc;
        float hall_booking_fee;
        double total_bill;

    public:
        void calculate_security(); //used to calculate amt for security
staff
        void calculate_miscellaneous(miscellaneous &);
        void BillCalculation(float,float,float);
        void BillDisplay(miscellaneous &);
};

void BillGenerate::calculate_security()
{
    int security_guards=10;
    security=security_guards*100*30; //salary of Rs. 100 per day. Assuming
30 days are there in a month.

}

void BillGenerate::calculate_miscellaneous(miscellaneous &misc)
{

miscellaneous_calc=misc.water_bill+misc.electricity_bill+misc.building_rep
air+misc.gardening+misc.road_repair+misc.additional_staff;
    //Sums up all of miscellaneous costs
}

void BillGenerate::BillCalculation(float user_fee,float clubhouse,float
hall_booking) //calculates total bill
{
    clubhouse_fee=clubhouse;
    hall_booking_fee=hall_booking;
total_bill=security+miscellaneous_calc+user_fee+hall_booking_fee+clubhouse
_fee;
}

void BillGenerate::BillDisplay(miscellaneous &misc) //displays bill
{
    cout<<"\nClubhouse fee: "<<clubhouse_fee;
    cout<<"\nSecurity fee: "<<security;
    misc.display_misc_details();
    if(hall_booking_fee!=0)
    {
        cout<<"\nHall booking charges: "<<hall_booking_fee;
    }
    cout<<"\nTotal bill is: "<<total_bill<<endl;
}
```

OOP_QUERY.CPP:
Contains the Query Management Module.

```
#include <iostream>
#include<bits/stdc++.h> //standard library to convert string to upper
character
#include "oop_users.cpp"
using namespace std;

class query
{
    private:
        char name[50];
        int flat_no;
        char title[30];
        char description[200];


    public:
        int choice;
        string temp;
        void get_contacts(staff s[], int);
        void raise_query();
        void display_query();
};
//will take the data for raise query
void query::raise_query()
{
    cout<<"\nResident Name: ";
    cin.ignore();
    cin.getline(name,50);
    cout<<"Flat No: ";
    cin>>flat_no;
    cout<<"Title: ";
    cin.ignore();
    cin.getline(title,30);
    cout<<"State your complaint: ";
    cin.getline(description,200);

}

void query::display_query()
{
    cout<<"\nResident Name: "<<name;
    cout<<"\nFlat No: "<<flat_no;
    cout<<"\nTitle: "<<title;
    cout<<"\nState your complaint: "<<description;
}
//will take data from s[] , compare different staff jobs and will display
the staff name,staff job and phone number
void query::get_contacts(staff s[], int staff_records)
{
    int flag=0;
    cout<<"\nHelp available:";
    cout<<"\n1. Maid"<<endl;
    cout<<"2. Plumber"<<endl;
    cout<<"3. Electrician"<<endl;
    cout<<"Enter choice: ";
    cin>>choice;
    switch(choice)
    {
    case 1:
        for(int a=0; a<staff_records; a++)
        {
            temp=s[a].get_staff_job();
            transform(temp.begin(), temp.end(),temp.begin(),
::toupper);//convert staff_job to upper string
            if(temp=="MAID")
            {
                flag=1;
                cout<<"\nStaff name: "<<s[a].get_staff_name();
                cout<<"\nStaff job: "<<temp;
                cout<<"\nPhone no: "<<s[a].get_staff_phone();
                cout<<endl;
            }
        }
        break;
    case 2:
        for(int a=0; a<staff_records; a++)
        {
            temp=s[a].get_staff_job();
            transform(temp.begin(), temp.end(),temp.begin(),
::toupper);//convert staff_job to upper string
            if(temp=="PLUMBER")
            {
                flag=1;
                cout<<"staff name: "<<s[a].get_staff_name();
                cout<<"Staff job: "<<temp;
                cout<<"\nPhone no: "<<s[a].get_staff_phone();
            }
        }
        break;
    case 3:
        for(int a=0; a<staff_records; a++)
        {
            temp=s[a].get_staff_job();
            transform(temp.begin(), temp.end(),temp.begin(),
::toupper);//convert staff_job to upper string
            if(temp=="ELECTRICIAN")
            {
                flag=1;
                cout<<"staff name: "<<s[a].get_staff_name();
                cout<<"Staff job: "<<temp;
                cout<<"\nPhone no: "<<s[a].get_staff_phone();
            }
        }
        break;
    default:
        cout<<"\nInvalid option";
        break;
    }
     if(flag==0)
     {
        cout<<"\nSorry, this staff is not available."<<endl;
     }
}
```

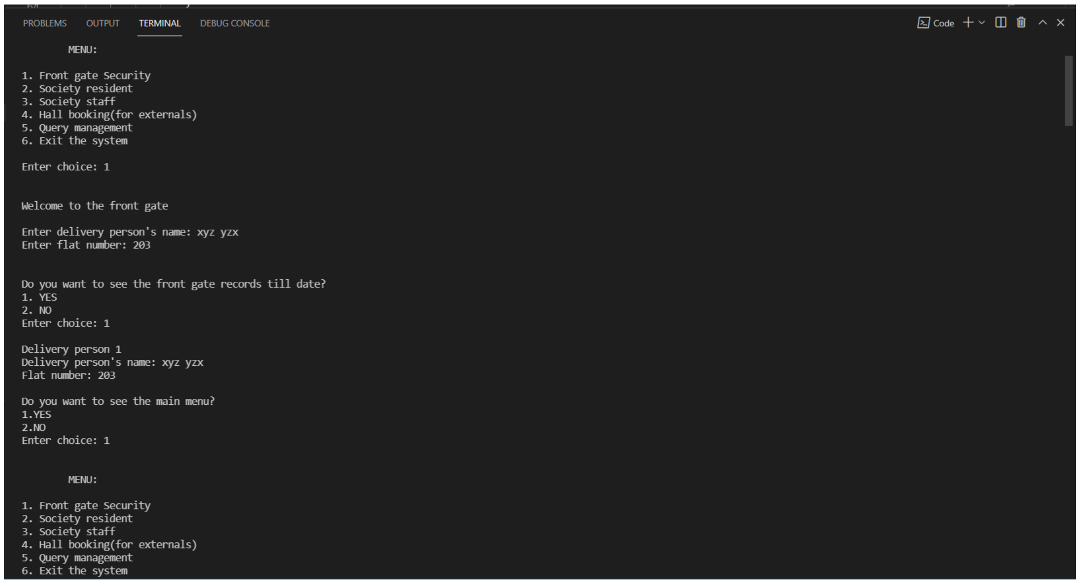
OOP_FINAL.CPP:
This has the int main and is used to run the code.

```
#include <iostream>
#include "oop_frontgate.cpp"
#include "oop_query.cpp"

#include<fstream> //for file handling
using namespace std;
fstream file; //file object for frontgate
fstream file2; //file object for query

int main()
{
    file.open("frontgate.dat", ios::out|ios::binary); // file for front
gate
    file.close();
    file2.open("query.dat", ios::out|ios::binary); // file for query
    file2.close();
    user u[300]; //assuming that the society has 300 flats/row houses in 
total
    staff s[50]; //assumming that the society has 50 staff memebers
    HallBooking h[25]; //assumming that the society can have 25 hall 
bookings for external users
    Front_gate f,temp;
    query q,temp1;

    int j=1;//used in printing of query/front gate
    int user_register,staff_register; //sees if the society resident is an 
already registered user
    int choice,main_menu_choice,query_choice; //used to store various 
choices
    int club_flag=0; //keeps track if the staff is from the clubhouse
    int user_count=0,staff_count=0,hallbooking_count=0; //keeps record of 
the number of users,staff,hallbookings registered till date respctively

    ch: //label for main menu
    cout<<"\n\n\tMENU:"<<endl;
    cout<<"\n1. Front gate Security";
    cout<<"\n2. Society resident";
    cout<<"\n3. Society staff";
    cout<<"\n4. Hall booking(for externals)";
    cout<<"\n5. Query management";
    cout<<"\n6. Exit the system"<<endl;
    cout<<"\nEnter choice: ";
    cin>>choice;
    cout<<endl;

    switch(choice)
    {
        case 1:
            int f_choice;
            cout<<"\nWelcome to the front gate"<<endl;
            file.open("frontgate.dat",ios::app|ios::binary); //opening 
file for writing
            if(file)
            {
                f.detail_gate();
                file.write((char *)&f,sizeof(f)); // write the object to a
file
            }
            file.close();
            file.open("frontgate.dat", ios::in|ios::binary); //open file
for reading
            cout<<"\n\nDo you want to see the front gate records till
date?";
            cout<<"\n1. YES \n2. NO\nEnter choice: ";
            cin>>f_choice;
            if(f_choice==1)
            {
                j=1;
                while(!file.eof())
                {
                    file.read((char*)&temp,sizeof(temp)); //reads from the
file
                    if(file)
                    {
                        cout<<endl<<"Delivery person "<<j;
                        temp.display_gate();
                        cout<<endl;
                        j++;
                    }
                }
            }
            file.close(); // close the file
            break;
        case 2:
            int society_choice,society_cont; //keeps track of society menu
option, if they want to see the society menu again
            int flat_num,user_record; //used to store flat no of existing
user to compare in the records, keeps track of the fact that the user at
present is a new user or an existing user
            float club_fee; //used to store the club fee
            float hall_fee;
            float user_fee;
            cout<<"\n\tSOCIETY RESIDENT VIEW\n";
            cout<<"\nHave you already registered on the society app?\n1.
NO\n2. YES";
```

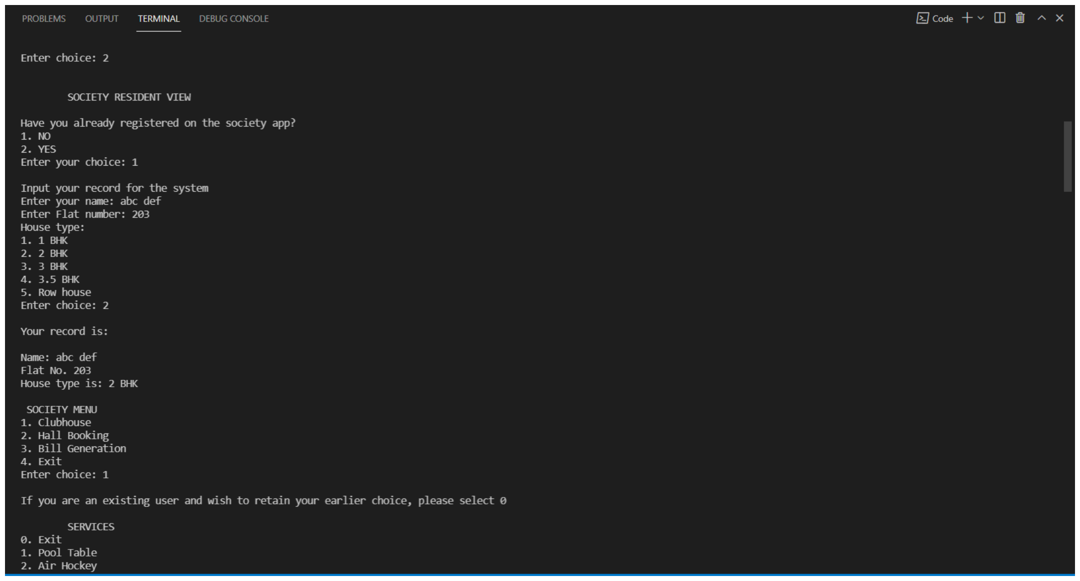
```
            cout<<"\nEnter your choice: ";
            cin>>user_register;
            user_record=user_count; //initially, user_record is set to 
point to the next society user
            if(user_register==1) //new user is there-take their info
            {
                cout<<"\nInput your record for the system";
                u[user_count].get_user_details();
            }
            else //existing user found, check for flat number
            {
                cout<<"\nEnter your flat number: ";
                cin>>flat_num;
                for(int i=0;i<user_count;i++)
                {
                    if(flat_num==u[i].get_flat_no())
                    user_record=i; //this now points to the record of the 
existing user
                }
            }
            cout<<"\nYour record is: "<<endl;
            u[user_record].display_user_details(); //displays user details
            do //society menu
            {
                cout<<"\n\n SOCIETY MENU";
                cout<<"\n1. Clubhouse";
                cout<<"\n2. Hall Booking";
                cout<<"\n3. Bill Generation";
                cout<<"\n4. Exit";
                cout<<"\nEnter choice: ";
                cin>>society_choice;
                if(society_choice==1) //clubhouse
                {
                    cout<<"\nIf you are an existing user and wish to 
retain your earlier choice, please select 0"<<endl;
                    club_fee=u[user_record].service_avail();
                }
                else if(society_choice==2) //hall booking
                {
                    int flag; //used to check if hall booking is requested
by the user or not
                    u[user_record].rent_booking();
                    cout<<"\n\nDo you want to book the hall?\n1. YES\n2.
NO";
                    cout<<"\nEnter choice: ";
                    cin>>flag;
                    hall_fee=u[user_record].make_booking(flag);
                }
                else if(society_choice==3)//bill generate
                {
                    miscellaneous misc;
                    misc.Get_misc_details();
                    u[user_record].calculate_security();
                    u[user_record].calculate_miscellaneous(misc);
                    user_fee=u[user_record].user_calculate();

u[user_record].BillCalculation(user_fee,club_fee,hall_fee);
                    cout<<"\n\nBILL";
                    u[user_record].display_user_calculate();
                    u[user_record].BillDisplay(misc);
                }
                else
                {
                    ; //user doesn't want to do anything after his record
creation
                }
                if(user_register==1) //if new record is added to the
system, the user count should increase by 1
                {
                    user_count+=1;
                }
                cout<<"\nDo you want to see the society menu again?\n1.
YES\n2. NO";
                cout<<"\nEnter choice: ";
                cin>>society_cont;
            }while(society_cont==1); //to see if the user wants to see the
society menu
            break;
        case 3:
            int staff_c; //stores code of registered staff
            int club_choice;  //sees if the staff is part of the clubhouse
            int staff_record; //keeps track of the fact that the staff at
present is a new staff or an existing staff
            cout<<"\n\tSTAFF VIEW"<<endl;
            cout<<"\nAre you a registered staff?";
            cout<<"\n1. No \n2. YES";
            cout<<"\nEnter choice: ";
            cin>>staff_register;

            staff_record=staff_count; //initially, staff_record is set to
point to the next society staff
            if(staff_register==1)
            {
                s[staff_count].get_staff_details();
            }
            else //existing staff found, check for staff code
            {
                cout<<"\nEnter your staff code: ";
                cin>>staff_c;
                for(int i=0;i<staff_count;i++)
                {
                    if(staff_c==s[i].get_staff_code())
                    staff_record=i; //this now points to the record of the
existing staff
                }
            }
            cout<<"\n\nAre you a part of the club house staff?\n1. YES
\n2. NO";
            cout<<"\nEnter your choice: ";
            cin>>club_choice;
            club_flag=0;
            if(club_choice==1) //staff is part of the clubhouse
            {
                s[staff_record].staff_clubhouse();
                club_flag=1; //sets flag to 1-staff is part of clubhouse
            }

            if(club_flag==1)
            {
                s[staff_record].display_staff_details();
                s[staff_record].clubhouse_staff_display();
            }
            else
            {
                s[staff_record].display_staff_details();
            }

            if(staff_register==1) //if new record is added to the system, the staff count should increase by 1
            {
                staff_count+=1;
            }
            break;
        case 4: // hall booking for external users
            int hall_flag; //to see if they want to book the hall
            float booking_fee;

            h[hallbooking_count].rent_booking(); //shows hall charges
            cout<<"\n\nDo you want to book the hall?\n1. YES\n2. NO";
            cout<<"\nEnter choice: ";
            cin>>hall_flag;
            if(hall_flag==1) //user books the hall
            {
                booking_fee=h[hallbooking_count].make_booking(hall_flag);
                cout<<"\nHall charges are: "<<booking_fee<<endl;
            }
            break;
        case 5:
            cout<<"\n\tQUERY MANAGEMENT\n";
            cout<<"\n1. Generate number of help";
            cout<<"\n2. Raise a query";
            cout<<"\nEnter choice: ";
            cin>>query_choice;
            if(query_choice==1)
            {
                q.get_contacts(s,staff_count);
            }
            else
            {
                int q_choice;
                cout<<"\nWelcome to the Query portal";
                file2.open("query.dat",ios::app|ios::binary); //open file
for writing
                if(file2)
                {
                    q.raise_query();
                    file2.write((char *)&q,sizeof(q));// write the object
to a file
                }
                file2.close();
                file2.open("query.dat", ios::in|ios::binary); //open file
for reading
                cout<<"\n\nDo you want to see all the queries? ";
                cout<<"\n1. YES \n2. NO\nEnter choice: ";
                cin>>q_choice;
                if(q_choice==1)
                {
                    j=1;
                    while(!file2.eof())
                    {
                        file2.read((char*)&temp1,sizeof(temp1)); //reads
from the file
                        if(file2)
                        {
                            cout<<endl<<"Query "<<j;
                            temp1.display_query();
                            cout<<endl;
                            j++;
                        }
                    }
                }
                file2.close(); // close the file
            }
            break;
        case 6:
            exit(0);
            break;
        default: //user enters invalid choice
            {
                cout<<"\nInvalid choice.Enter choice again.";
                goto ch;
            }
    }
    cout<<"\nDo you want to see the main menu?\n1.YES\n2.NO";
    cout<<"\nEnter choice: ";
    cin>>main_menu_choice;
    if(main_menu_choice==1) //to see if the user wants to go back to the
main menu
    {
        goto ch;
    }
    return 0;
}
```

GIT LINK: https://github.com/AvantikaJalote/OOP_MINI_PROJECT.git

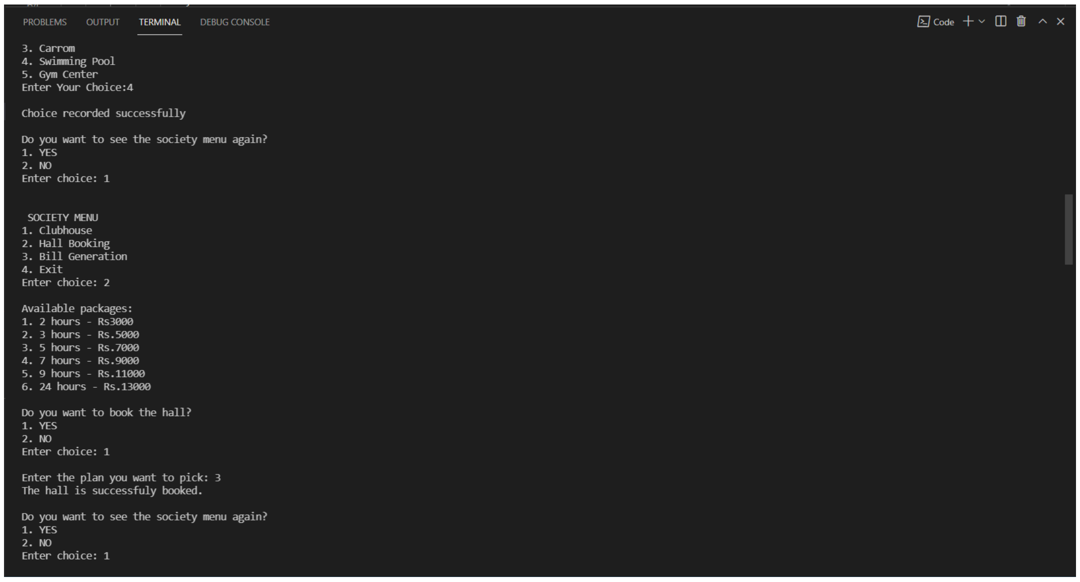
OUTPUT SCREENSHOTS



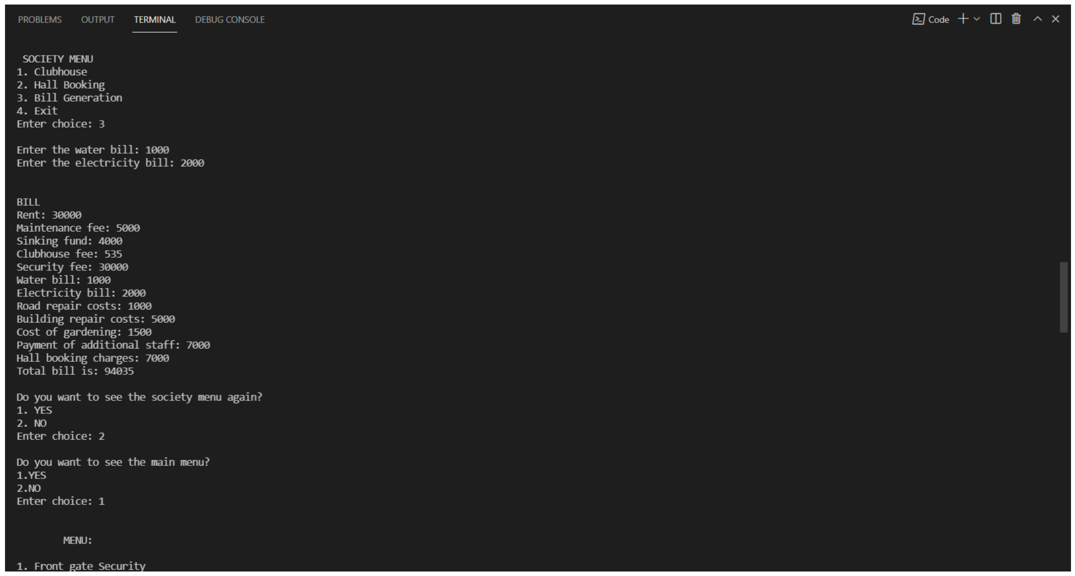
```
        MENU:

1. Front gate Security
2. Society resident
3. Society staff
4. Hall booking(for externals)
5. Query management
6. Exit the system

Enter choice: 1


Welcome to the front gate

Enter delivery person's name: xyz yzx
Enter flat number: 203


Do you want to see the front gate records till date?
1. YES
2. NO
Enter choice: 1

Delivery person 1
Delivery person's name: xyz yzx
Flat number: 203

Do you want to see the main menu?
1.YES
2.NO
Enter choice: 1


        MENU:

1. Front gate Security
2. Society resident
3. Society staff
4. Hall booking(for externals)
5. Query management
6. Exit the system
```



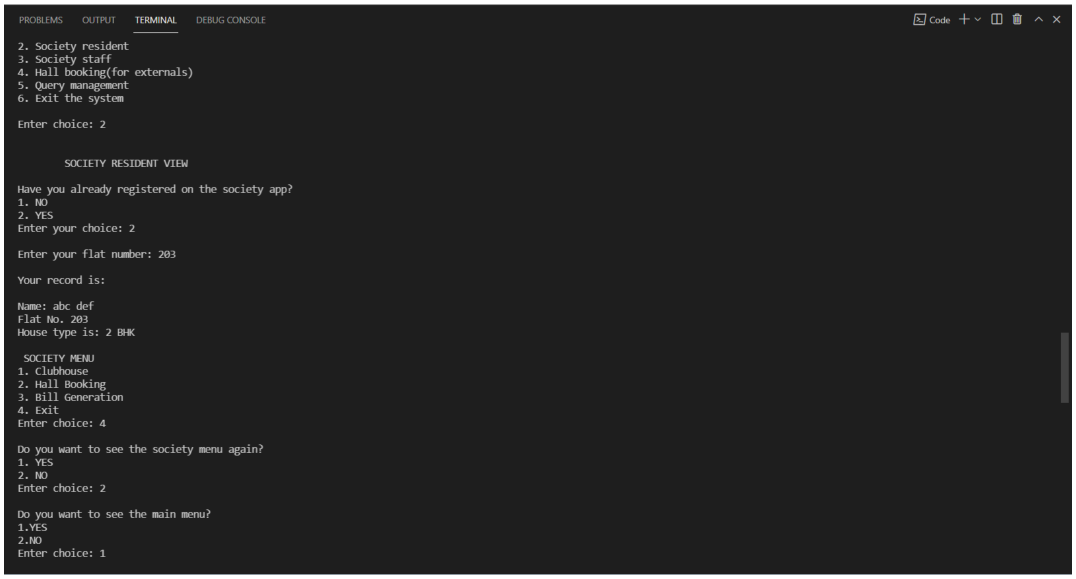
```
Enter choice: 2


        SOCIETY RESIDENT VIEW

Have you already registered on the society app?
1. NO
2. YES
Enter your choice: 1

Input your record for the system
Enter your name: abc def
Enter Flat number: 203
House type:
1. 1 BHK
2. 2 BHK
3. 3 BHK
4. 3.5 BHK
5. Row house
Enter choice: 2

Your record is:

Name: abc def
Flat No. 203
House type is: 2 BHK

 SOCIETY MENU
1. Clubhouse
2. Hall Booking
3. Bill Generation
4. Exit
Enter choice: 1

If you are an existing user and wish to retain your earlier choice, please select 0

        SERVICES
0. Exit
1. Pool Table
2. Air Hockey
```



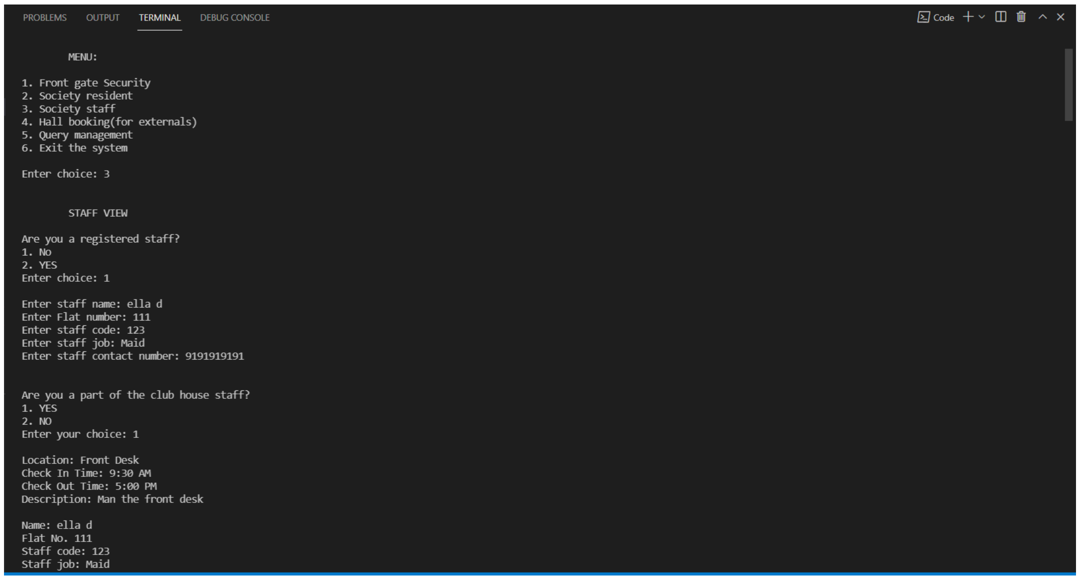
```
3. Carrom
4. Swimming Pool
5. Gym Center
Enter Your Choice:4

Choice recorded successfully

Do you want to see the society menu again?
1. YES
2. NO
Enter choice: 1


 SOCIETY MENU
1. Clubhouse
2. Hall Booking
3. Bill Generation
4. Exit
Enter choice: 2

Available packages:
1. 2 hours - Rs3000
2. 3 hours - Rs.5000
3. 5 hours - Rs.7000
4. 7 hours - Rs.9000
5. 9 hours - Rs.11000
6. 24 hours - Rs.13000

Do you want to book the hall?
1. YES
2. NO
Enter choice: 1

Enter the plan you want to pick: 3
The hall is successfuly booked.

Do you want to see the society menu again?
1. YES
2. NO
Enter choice: 1
```



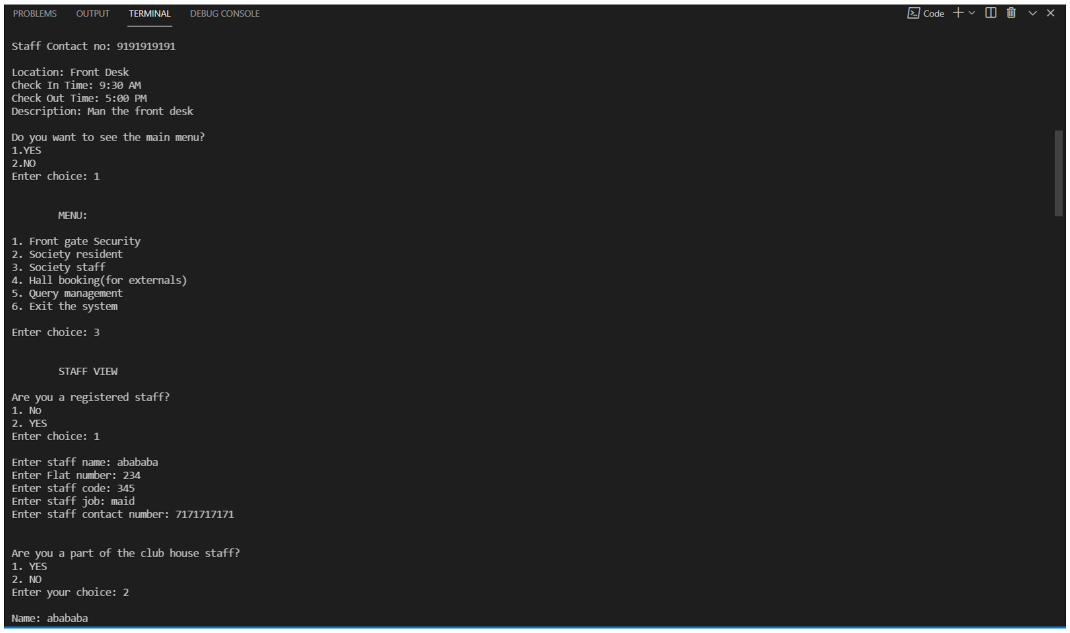
```
 SOCIETY MENU
1. Clubhouse
2. Hall Booking
3. Bill Generation
4. Exit
Enter choice: 3

Enter the water bill: 1000
Enter the electricity bill: 2000


BILL
Rent: 30000
Maintenance fee: 5000
Sinking fund: 4000
Clubhouse fee: 535
Security fee: 30000
Water bill: 1000
Electricity bill: 2000
Road repair costs: 1000
Building repair costs: 5000
Cost of gardening: 1500
Payment of additional staff: 7000
Hall booking charges: 7000
Total bill is: 94035

Do you want to see the society menu again?
1. YES
2. NO
Enter choice: 2

Do you want to see the main menu?
1.YES
2.NO
Enter choice: 1


        MENU:

1. Front gate Security
```



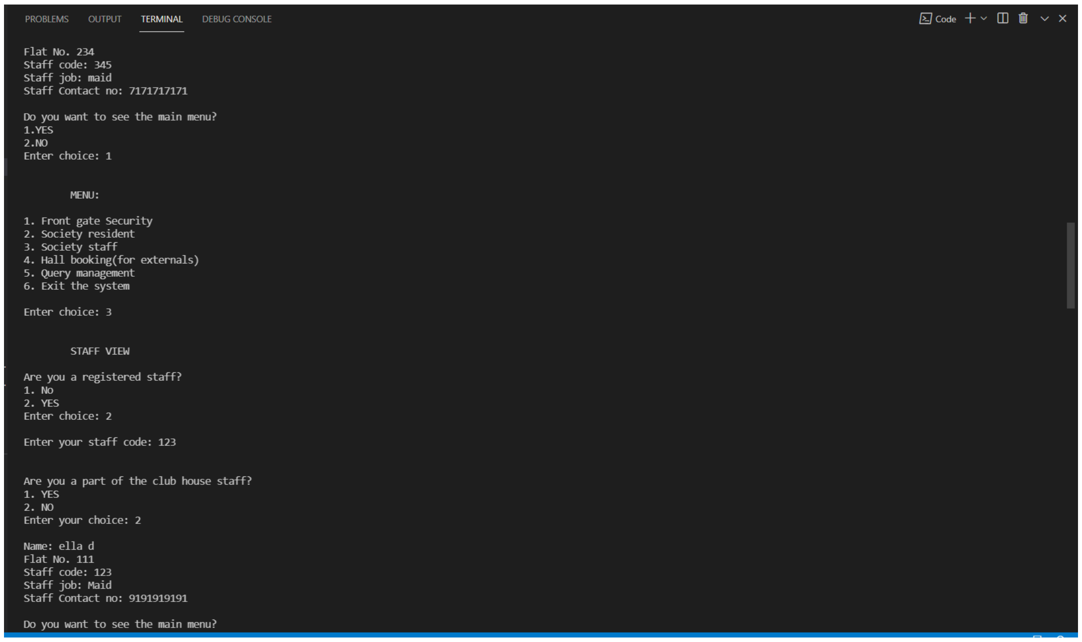
```
2. Society resident
3. Society staff
4. Hall booking(for externals)
5. Query management
6. Exit the system

Enter choice: 2


        SOCIETY RESIDENT VIEW

Have you already registered on the society app?
1. NO
2. YES
Enter your choice: 2

Enter your flat number: 203

Your record is:

Name: abc def
Flat No. 203
House type is: 2 BHK

 SOCIETY MENU
1. Clubhouse
2. Hall Booking
3. Bill Generation
4. Exit
Enter choice: 4

Do you want to see the society menu again?
1. YES
2. NO
Enter choice: 2

Do you want to see the main menu?
1.YES
2.NO
Enter choice: 1
```



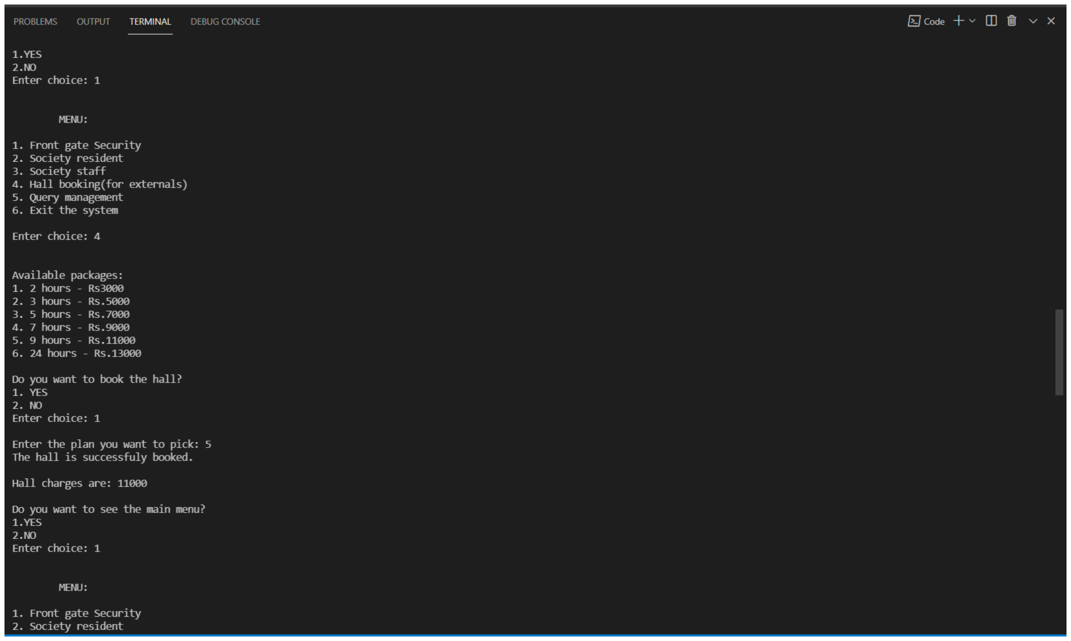
```
        MENU:

1. Front gate Security
2. Society resident
3. Society staff
4. Hall booking(for externals)
5. Query management
6. Exit the system

Enter choice: 3


        STAFF VIEW

Are you a registered staff?
1. No
2. YES
Enter choice: 1

Enter staff name: ella d
Enter Flat number: 111
Enter staff code: 123
Enter staff job: Maid
Enter staff contact number: 9191919191


Are you a part of the club house staff?
1. YES
2. NO
Enter your choice: 1

Location: Front Desk
Check In Time: 9:30 AM
Check Out Time: 5:00 PM
Description: Man the front desk

Name: ella d
Flat No. 111
Staff code: 123
Staff job: Maid
```



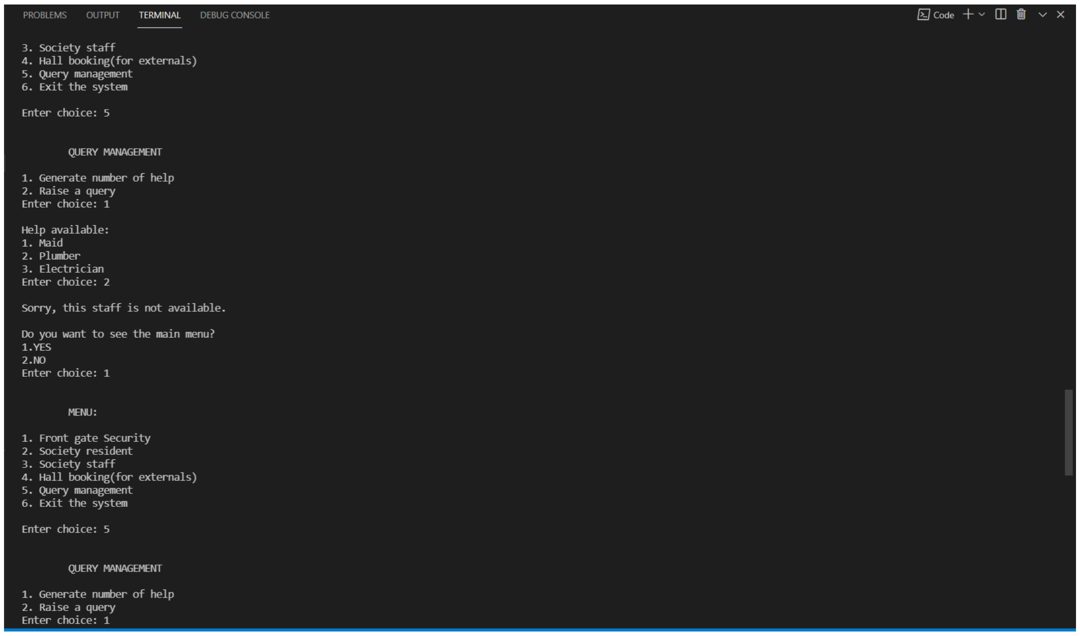
```
Staff Contact no: 9191919191

Location: Front Desk
Check In Time: 9:30 AM
Check Out Time: 5:00 PM
Description: Man the front desk

Do you want to see the main menu?
1.YES
2.NO
Enter choice: 1


        MENU:

1. Front gate Security
2. Society resident
3. Society staff
4. Hall booking(for externals)
5. Query management
6. Exit the system

Enter choice: 3


        STAFF VIEW

Are you a registered staff?
1. No
2. YES
Enter choice: 1

Enter staff name: abababa
Enter Flat number: 234
Enter staff code: 345
Enter staff job: maid
Enter staff contact number: 7171717171


Are you a part of the club house staff?
1. YES
2. NO
Enter your choice: 2

Name: abababa
```



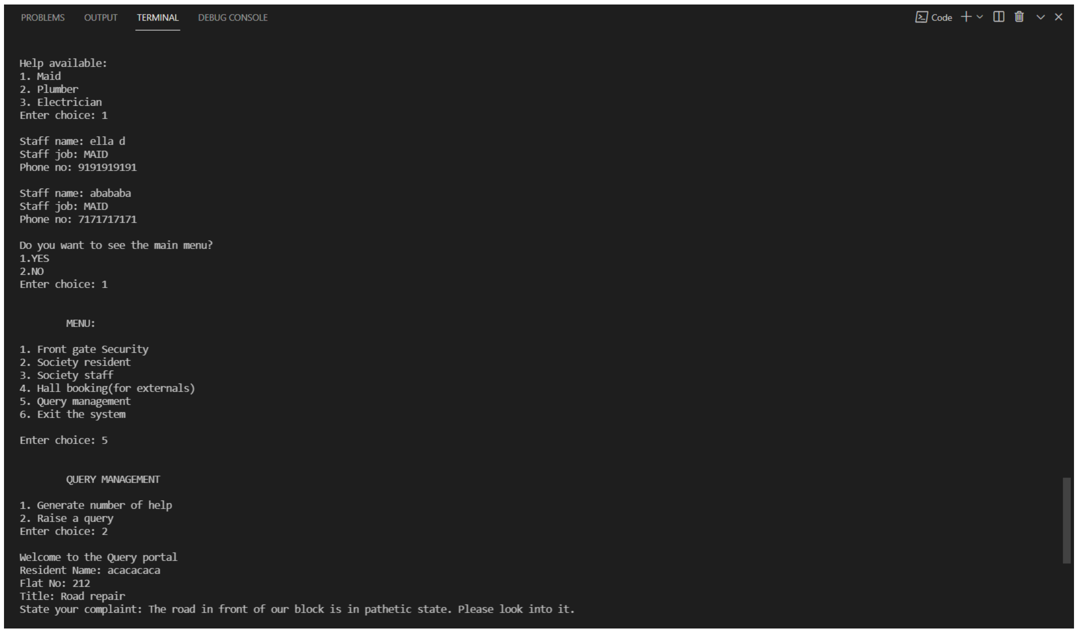
```
Flat No. 234
Staff code: 345
Staff job: maid
Staff Contact no: 7171717171

Do you want to see the main menu?
1.YES
2.NO
Enter choice: 1


        MENU:

1. Front gate Security
2. Society resident
3. Society staff
4. Hall booking(for externals)
5. Query management
6. Exit the system

Enter choice: 3


        STAFF VIEW

Are you a registered staff?
1. No
2. YES
Enter choice: 2

Enter your staff code: 123


Are you a part of the club house staff?
1. YES
2. NO
Enter your choice: 2

Name: ella d
Flat No. 111
Staff code: 123
Staff job: Maid
Staff Contact no: 9191919191

Do you want to see the main menu?
```



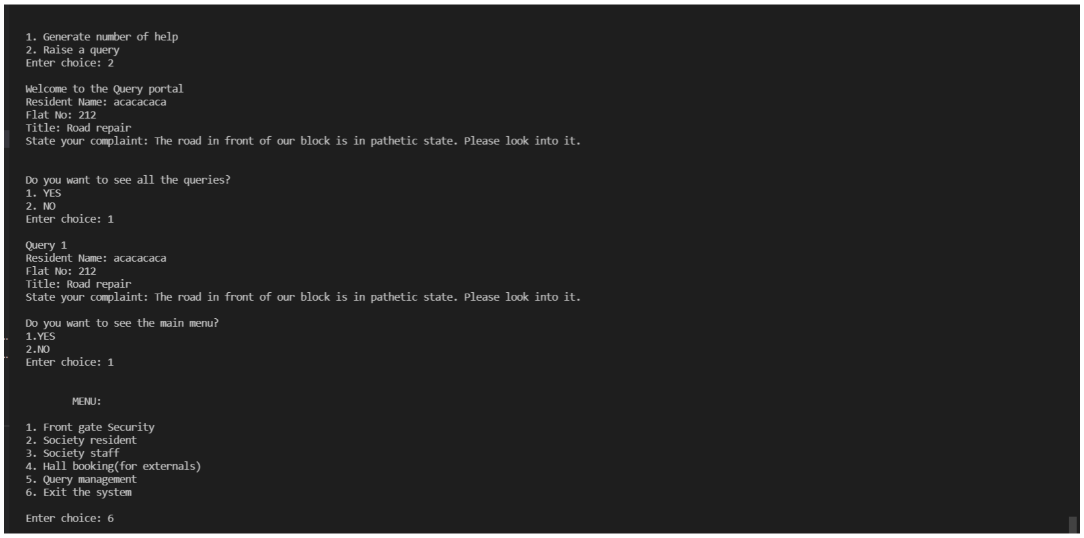
```
1.YES
2.NO
Enter choice: 1


        MENU:

1. Front gate Security
2. Society resident
3. Society staff
4. Hall booking(for externals)
5. Query management
6. Exit the system

Enter choice: 4


Available packages:
1. 2 hours - Rs3000
2. 3 hours - Rs.5000
3. 5 hours - Rs.7000
4. 7 hours - Rs.9000
5. 9 hours - Rs.11000
6. 24 hours - Rs.13000

Do you want to book the hall?
1. YES
2. NO
Enter choice: 1

Enter the plan you want to pick: 5
The hall is successfuly booked.

Hall charges are: 11000

Do you want to see the main menu?
1.YES
2.NO
Enter choice: 1


        MENU:

1. Front gate Security
2. Society resident
```



```
3. Society staff
4. Hall booking(for externals)
5. Query management
6. Exit the system

Enter choice: 5


        QUERY MANAGEMENT

1. Generate number of help
2. Raise a query
Enter choice: 1

Help available:
1. Maid
2. Plumber
3. Electrician
Enter choice: 2

Sorry, this staff is not available.

Do you want to see the main menu?
1.YES
2.NO
Enter choice: 1


        MENU:

1. Front gate Security
2. Society resident
3. Society staff
4. Hall booking(for externals)
5. Query management
6. Exit the system

Enter choice: 5


        QUERY MANAGEMENT

1. Generate number of help
2. Raise a query
Enter choice: 1
```



```
Help available:
1. Maid
2. Plumber
3. Electrician
Enter choice: 1

Staff name: ella d
Staff job: MAID
Phone no: 9191919191

Staff name: abababa
Staff job: MAID
Phone no: 7171717171

Do you want to see the main menu?
1.YES
2.NO
Enter choice: 1


        MENU:

1. Front gate Security
2. Society resident
3. Society staff
4. Hall booking(for externals)
5. Query management
6. Exit the system

Enter choice: 5


        QUERY MANAGEMENT

1. Generate number of help
2. Raise a query
Enter choice: 2

Welcome to the Query portal
Resident Name: acacacaca
Flat No: 212
Title: Road repair
State your complaint: The road in front of our block is in pathetic state. Please look into it.
```



```
1. Generate number of help
2. Raise a query
Enter choice: 2

Welcome to the Query portal
Resident Name: acacacaca
Flat No: 212
Title: Road repair
State your complaint: The road in front of our block is in pathetic state. Please look into it.


Do you want to see all the queries?
1. YES
2. NO
Enter choice: 1

Query 1
Resident Name: acacacaca
Flat No: 212
Title: Road repair
State your complaint: The road in front of our block is in pathetic state. Please look into it.

Do you want to see the main menu?
1.YES
2.NO
Enter choice: 1


        MENU:

1. Front gate Security
2. Society resident
3. Society staff
4. Hall booking(for externals)
5. Query management
6. Exit the system

Enter choice: 6
```

CONCLUSION

Through this project, we were able to create a society management system with various simple and complex functions. We were also able to apply various concepts of C++. The end product was a system that functions as we expect it to with all the functionalities that we wanted to implement at the start. This activity brings us one step closer to having a completely digitalised, user-friendly and competent Society Management System.