



Topic : Online Property Sales System

Group no : MLB_WE_01.02_02

Campus : Malabe

Submission Date: 20/05/2022

We declare that this is our own work, and this Assignment does not incorporate without acknowledgment any material previously submitted by anyone else in SLIIT or any other university/Institute. And we declare that each one of us equally contributed to the completion of this Assignment.

Registration No	Name	Contact Number
IT21391668	H.N.D. Madhubhashana	0774686996
IT21357008	S.K.N.C. Karunanayaka	0716263256
IT21489914	H.A.S.A. Wijedasa	0726430848
IT21509186	H.G.L.R.Silva	0702155472
IT21575044	N.L.M. Fernando	0778197886

Requirements

- User should provide his/her details such as name, address to register into the system.
- Registered users can be divided as buyers and sellers.
- Buyer can search for properties through the application, and make an appointment with the seller.
- Seller can post his/her properties as advertisements through the application, can promote his/her advertisement, can request for seller verification, to post the advertisement seller should make a payment for the advertisement and he/she can specify the payment method (credit card, debit card, PayPal) for each advertisement.
- Seller can also search for properties and make appointments with other sellers.
- Seller can see the status of his/her advertisement and get a list of previous advertisements he posted through his profile
- Administrator can approve advertisement requests, appointments, and seller verification, can update, and maintain the system.
- System stores the sale details of property
- System updates the sale details of property
- System should maintain status of property

Classes

- ✓ Registered user
- ✓ Buyer
- ✓ Seller
- ✓ Advertisement
- ✓ Payment
- ✓ Payment Method
- ✓ Credit / Debit card
- ✓ PayPal
- ✓ Appointment
- ✓ Report
- ✓ Property
- ✓ Profile

CRC Cards

Registered User	
Responsibilities	Collaboration
Search Property	Advertisement

Buyer	
Responsibilities	Collaboration
Search Property	Advertisement
Make Appointment	Appointment

Seller	
Responsibilities	Collaboration
Post Advertisement	Advertisement
Request Advertisement Promotion	Advertisement
Request Verification	
Make Payment	Payment
Make Appointment	Appointment
Update Advertisement	

Advertisement	
Responsibilities	Collaborations
Generate Advertisement	Seller
Update Advertisement	Seller
Promote Advertisement	Seller

Payment	
Responsibilities	Collaborations
Store payment details	
Validate payment	

Calculate payment	
-------------------	--

Property	
Responsibilities	Collaborations
Store the sale details of property	
Update the sale details of property	
Maintain status of property	
Owner details	Seller

Report	
Responsibilities	Collaboration
List of advertisement	Advertisement
List of appointment	Appointment
List of verification request	Seller
List of property	Property

Credit/Debit Card	
Responsibilities	Collaborations
Check and validate card details	Registered user / Seller
Store card details	

Appointment	
Responsibilities	Collaboration
Schedule appointment	Buyer, Seller

Profile	
Responsibilities	Collaboration
Show the status of advertisement	Seller
Show the list of previous advertisement	Seller

PaymentMethod	
Responsibilities	Collaborations
Display Payment Method	
Add Payment Method	
Update Payment Method	
Delete Payment Method	
Payment Details	Payment

PayPal	
Responsibilities	Collaboration
Store PayPal account details	
Check details	



Codings

```
//IT21391668
```

```
//H.N.D. Madhubhashana
```

```
//Class for Registered User
```

```
Class registeredUser {
```

```
protected:
```

```
string username ;
```

```
string password ;
```

```
string email ;
```

```
string firstName ;
```

```
string lastName ;
```

```
int phone ;
```

public:

registeredUser(string m_username , string m_password , string m_email , string m_fistName .
string m_lastName , int m_phone);

void searchProperty();

~registeredUser() ;

};

//constructor

registeredUser :: registeredUser(string m_username , string m_password , string m_email ,
string m_fistName . string m_lastName , int m_phone) {

username = m_username ;

password = m_password ;

email = m_email ;

firstName = m_fistName ;

lastName = m_lastName ;

phone = m_phone ;

}

//destructor

registeredUser :: ~registeredUser(){

cout << "Deleting registeredUser" << endl ;

}

//Class for Buyer

Class Buyer: public registeredUser {

protected:

string username ;

string password ;

string email ;

string firstName ;

string lastName ;

int phone ;

int appointmentNumber ;

public:

Buyer(string m_username , string m_password , string m_email , string m_fistName . string
m_lastName , int m_phone);

void searchProperty();

void requestAppointment() ;

~Buyer() ;

};

//constructor

Buyer :: Buyer(string m_username , string m_password , string m_email , string m_fistName .
string m_lastName , int m_phone) {

username = m_username ;

password = m_password ;

email = m_email ;

firstName = m_fistName ;

lastName = m_lastName ;

phone = m_phone ;

```
}
```

```
//destructor
```

```
Buyer :: ~Buyer(){
```

```
cout << "Deleting Buyer" << endl ;
```

```
}
```

```
//Class for Seller
```

```
Class Seller: public registeredUser{
```

```
protected:
```

```
string username ;
```

```
string password ;
```

```
string email ;
```

```
string firstName ;
```

```
string lastName ;
```

```
int phone ;
```

```
int appointmentNumber ;
```

```
int advertisementID ;
```

```
int requestNumber ;
```

```
string paymentID ;
```

```
public:
```

```
Seller(string m_username , string m_password , string m_email , string m_fistName . string  
m_lastName , int m_phone );
```

```
void requestAppointment();
```

```
void postAdvertisement() ;
```

```
void requestVerification() ;  
void makePayment() ;  
void getAdvertisementList(Profile *PRO)  
~Seller() ;  
  
};
```

```
//constructor
```

```
Seller :: Seller(string m_username , string m_password , string m_email , string m_fistName .  
string m_lastName , int m_phone ) {
```

```
username = m_username ;  
password = m_password ;  
email = m_email ;  
firstName = m_fistName ;  
lastName = m_lastName ;  
phone = m_phone ;  
}
```

```
//destructor
```

```
Seller :: ~Seller(){  
cout << "Deleting Seller" << endl ;  
}
```

```
#include <iostream>
```

```
using namespace std ;
```

```
int main () {
```

```
string m_username ;  
string m_password ;  
string m_email ;  
string m_firstName ;  
string m_lastName ;  
int m_phone ;
```

```
registeredUser R1 ;
```

```
Seller S1 ;
```

```
Buyer B1 ;
```

```
R1.registeredUser(string m_username , string m_password , string m_email , string m_fistName  
. string m_lastName , int m_phone);
```

```
S1.Seller(string m_username , string m_password , string m_email , string m_fistName . string  
m_lastName , int m_phone);
```

```
B1.Buyer(string m_username , string m_password , string m_email , string m_fistName . string  
m_lastName , int m_phone);
```

```
return 0 ;
```

```
}
```

```
//Student Register Number :- IT21357008
```

```
//Student Name :- S.K.N.C.Karunanayaka
```

```
//Bi-directional Association between the Seller and Property.
```

```
//Seller can publish one or more property. Property belongs to only one Seller.
```

```
#include<iostream>
```

```
#include<string>
```

```
#define SIZE 10
```

```

using namespace std;

class Seller;
class Property;

//Class Seller
class Seller {
    private:
        string seller_userFName;
        string seller_userLName;
        string seller_email;
        string seller_contNumber;
        //Bi-directional association with Property
        Property *property[SIZE];
        int noOfProperty;
    public:
        //constructor1
        Seller();
        //constructor2
        Seller(string pseller_userFName, string pseller_userLName, string pseller_email,
string pseller_contNumber);
        void addProperty(Property *P);
        //Display seller details
        void displaySeller();
        //Destructor
        ~Seller(){
            cout<<"Deleting Seller"<<endl<<endl;
        }
};

//class Property
class Property {
    private:
        string pro_id;
        string pro_type;
        string pro_name;
        string pro_price;
        string pro_location;
        string pro_description;
        //Bi-directional association with Seller
        Seller *S;
    public:
        //Constructor
        Property(string ppro_id, string ppro_type, string ppro_name, string
ppro_price, string ppro_location, string ppro_description, Seller *pS);

```

```

        //Display payment Details of Seller
        void displayProperty();
        //Destructor
        ~Property(){
            cout<<"Deleting Property"<<endl<<endl;
        }
};

Property::Property(string ppro_id, string ppro_type, string ppro_name, string ppro_price, string
ppro_location, string ppro_description, Seller *pS)
{
    pro_id = ppro_id;
    pro_type = ppro_type;
    pro_name = ppro_name;
    pro_price = ppro_price;
    pro_location = ppro_location;
    pro_description = ppro_description;
    S = pS;
    S->addProperty(this);
}

void Property::displayProperty()
{
    cout<<"*****"<<endl;
    cout<<" Property Details of the seller mentioned above >>"<<endl<<endl;
    cout<<"Proprty ID : "<< pro_id <<endl;
    cout<<"Proprty Type : "<< pro_type <<endl;
    cout<<"Proprty Name : "<< pro_name <<endl;
    cout<<"Proprty Price : "<< pro_price <<endl;
    cout<<"Proprty Location : "<< pro_location <<endl;
    cout<<"Proprty Description : "<< pro_description <<endl;
}

Seller::Seller(string pseller_userFName, string pseller_userLName, string pseller_email, string
pseller_contNumber)
{
    seller_userFName = pseller_userFName;
    seller_userLName = pseller_userLName;
    seller_email = pseller_email;
    seller_contNumber = pseller_contNumber;
    noOfProperty= 0;
}

void Seller::addProperty(Property *P)
{
    if(noOfProperty < SIZE){
        property[noOfProperty] = P;
    }
}

```



```

        noOfProperty++;
    }
}

void Seller::displaySeller()
{
    cout<< " Seller First Name : "<<seller_userFName<<endl;
    cout<< " Seller Last Name : "<<seller_userLName<<endl;
    cout<< " Seller Email Address : "<<seller_email<<endl;
    cout<< " Seller Contact Number: "<<seller_contNumber<<endl<<endl;

    for(int i = 0; i < noOfProperty; i++)
    {
        property[i]->displayProperty();
    }
}

int main()
{
    //Bi-directional association of Seller and Property
    char sp;
    Seller *S1= new Seller("Nuwan","Ekanayaka","nuwaeka@gmail.com", "0712356237");
    Seller *S2= new Seller("Sugath","Aravindha","sugaravi@gmail.com","0776767678");

    Property *P1= new Property("L001","Land","Land for sale Gampaha","LKR
1700000","Gampaha","50 perch , Pure freehold deeds , Plumbing and well water , SLT cable
phone facilities , Electricity" , S1);
    Property *P2= new Property("H003","Home","Home for sale in Kandy","LKR
7500000","Kandy","20 Perch, 4 rooms,living room, dining room, kitchen, 2 attached
bathrooms",S1);
        cout<<"*****"<<endl;
        cout<<" Properties of >>" <<endl;
        S1->displaySeller();
        cin>>sp;
    delete S1;
    delete P1;
    return 0;
} //end of bi-directional association

```

//Uni-directional association between the Payment and PaymentMethod

//Payment has One or more Payment Method

```
#include<iostream>
```

```
#include<string>
```

```
using namespace std;
```

```
//class of PaymentMethod
```

```
class PaymentMethod{
```

```
    private:
```

```
        string payMethod_id;
```

```
        string payMethod_type;
```

```
    public:
```

```
        //Constructor1
```

```
        PaymentMethod();
```

```
        //Constructor2
```

```
        PaymentMethod(string ppayMethod_id, string ppayMethod_type)
```

```
        {
```

```
            payMethod_id = ppayMethod_id;
```

```
            payMethod_type = ppayMethod_type;
```

```
        }
```

```
        //Display Payment Method Details
```

```
        void displayDetails()
```

```
        {
```

```
            cout<<"Payment Method ID : "<<payMethod_id<<endl;
```

```
            cout<<"Payment Method Type : "<<payMethod_type<<endl;
```

```
        }
```

```
        //destructor
```

```
        ~PaymentMethod(){
```

```
            cout<<"Delete Payment Method"<<endl;
```

```
        }
```

```
};
```

```
//Class of Payment
```

```
class Payment{
```

```
    private:
```

```
        string pay_id;
```

```
        string pay_type;
```

```
        string pay_ammount;
```

```
        PaymentMethod *pymd;//an object of PaymentMethod as attribute
```

```
    public:
```

```
        //constructor
```

```
        Payment(string ppay_id, string ppay_type, string ppay_ammount,
```

```
        PaymentMethod *pm)
```

```
        {
```

```
            pay_id = ppay_id;
```

```

        pay_type = ppay_type;
        pay_ammount = ppay_ammount;
        pymd = pm;
    }
    //Details of Payment
    void displayDetails()
    {
        cout<<"Payment ID : "<<pay_id<<endl;
        cout<<"Payment Type : "<<pay_type<<endl;
        cout<<"Payment Ammount : "<<pay_ammount<<endl;
        pymd->displayDetails();
    }
    //Destructor
    ~Payment(){
        cout<<"Delete Payment"<<endl;
    }
};

int main()
{
    //uni-directional association of Payment and PaymentMethod
    char pp;
    PaymentMethod *pm = new PaymentMethod("CD001","Credit and Debit");
    pm->displayDetails();
    cout<<"*****"<<endl;
    Payment *p = new Payment("P2","Property Payment","LKR 2500000",pm);
    p->displayDetails();
    delete p;
    delete pm;
    cin>>pp;

    return 0;
} //end of uni-directional Association

```

```

//Inheritance between PaymentMethod and Credit
//Two Kind-of payment method

#include<iostream>
#include<cstring>

using namespace std;

class PaymentMethod{
protected:
    char payMethod_id[20];//CD001,CD002,PYPL001,PYPL002
    char payMethod_type[20];//Credit and Debit, Pay Pal
public:
    //default Constructor
    PaymentMethod()
    {
        cout<<"PaymentMethod Default Constructor
Called"<<endl<<endl;
    }
    //Overload Constructor
    PaymentMethod(char ppayMethod_id[],char ppayMethod_type[])
    {
        cout<<"PaymentMethod Overload Constructor
Called"<<endl<<endl;

        //Setting values
        strcpy(payMethod_id,ppayMethod_id);
        strcpy(payMethod_type,ppayMethod_type);
    }
    //Display Payment Method Details
    void displayDetails()
    {
        cout<<"Payment Method ID : "<<payMethod_id<<endl;
        cout<<"Payment Method Type :
"<<payMethod_type<<endl<<endl;
    }
    //destructor01
    ~PaymentMethod(){
        cout<<"Delete PaymentMethod Default Constructor
"<<endl<<endl;
    }
}

```

```

    }

};

class CreditDebit : public PaymentMethod{
protected:
    string card_type;
    string card_number;
    string card_holderName;
    string validityPeriod;
    int cvv;
public:
    CreditDebit(){
        cout<<"CreditDebit Default Constructor Called"<<endl<<endl;
    }
    CreditDebit(char ppayMethod_id[],char ppayMethod_type[],string pcard_type,
string pcard_number, string pcard_holderName, string pvalidityPeriod, int pcvv)

    {

        cout<<"This is CreditDebit class."<<" "<<"derived class from
PaymentMethod"<<endl<<endl;

        card_type = pcard_type;
        card_number = pcard_number;
        card_holderName = pcard_holderName;
        validityPeriod = pvalidityPeriod;
        cvv = pcvv;
    }

    void displayDetails(){
        cout<<" Card Type : "<<card_type<<endl;
        cout<<" Card Number : "<<card_number<<endl;
        cout<<" Card Holder Name : "<<card_holderName<<endl;
        cout<<" Card Validity Period : "<<validityPeriod<<endl;
        cout<<" Card CVV : "<<cvv<<endl<<endl;
    }
    ~CreditDebit(){
        cout<<"Delete CreditDebit Default Constructor "<<endl<<endl;
    }
};

int main(){
    PaymentMethod PM1; //PaymentMethod default constructor called
    PaymentMethod PM2("PYPL001","Pay Pal"); //PaymentMethod overload constroucter
called
    PM2.displayDetails();
}

```

```

        CreditDebit myCard1;//CreditDebit default constructor called
        CreditDebit myCard2("CD001","Credit and
Debit","Mastercard","2234567854326789","KUMARASIRI
JAGATH","09/26",267);//CreditDebit overload constructor called
        myCard2.displayDetails();
        myCard1.~CreditDebit();//creditDebit destructor called

        return 0;
}

```

```

//IT21575044
//N.L.M. Fernando
//Class for Report
Class Report{

private:
string advertisementnum ;
string appointmentnum ;

public:

void advertisementReport();
void getAdvertisementDetails() ;
void appointmentReport() ;
void getAppointmentDetails() ;
void propertyReport() ;
void getPropertyDetails() ;

};

void Report :: getAdvertisementDetails(Advertisement *A){
details = P-> getDetails() ;
}

void Report :: getAppointmentDetails(Appointment *Q){
details = Q-> getDetails() ;
}

getPropertyDetails(Property *P)
{
details = P-> getDetails() ;
}

```

```

//Class for Appointment
Class Appointment{

private:
string appointmentnum ;
string buyer_username ;
string seller_username ;


public:

void scheduleAppointment(Seller *S);
void scheduleAppointment((Buyer *B);
void getAppointmentDetails(Appointment *Q) ;

};


#include <iostream>
using namespace std ;

int main () {

string advertisementnum ;
string appointmentnum ;
string buyer_username ;
string seller_username ;

Report R1 , R2 ;
Appointment A ;

R1.getAdvertisementDetails(string advertisementnum) ;
R1.advertisementReport();

R2.getAppointementDetails(string appointmentnum) ;
R2.appointmentReport();

A.scheduleAppointment() ;

return 0 ;
}

```

```
// IT21509186
// H.G.L.R.Silva
//Inheritance between PaymentMethod and paypal
#include<iostream>
#include<cstring>
using namespace std;
class PaymentMethod{
protected:
char payMethod_id[20];
char payMethod_type[20];// Pay Pal
public:
//default Constructor
PaymentMethod()
{cout<<"PaymentMethod Default Constructor Called"<<endl<<endl;}
//Overload Constructor
PaymentMethod(char ppayMethod_id[],char ppayMethod_type[])
{
cout<<"PaymentMethod Overload Constructor Called"<<endl<<endl;
//Setting values
strcpy(payMethod_id,ppayMethod_id);
```



```

strcpy(payMethod_type,ppayMethod_type);}

//Display Payment Method Details
void displayDetails()
{cout<<"Payment Method ID : "<<payMethod_id<<endl;
cout<<"Payment Method Type : "<<payMethod_type<<endl<<endl;}

//destructor01
~PaymentMethod(){
cout<<"Delete PaymentMethod Default Constructor "<<endl<<endl;
}
};

class paypal :public PaymentMethod{
public:
string user_id;
string password;
int payment_id;
public:
paypal(){
cout<<"paypal Default Constructor Called"<<endl<<endl;
}

paypal(char ppayMethod_id[],char ppayMethod_type[],string puser_id, string ppassword,int
ppayment_id)
{
cout<<"This is paypal."<<" "<<"derived class from PaymentMethod"<<endl<<endl;
user_id = puser_id;
password = ppassword;
payment_id = ppayment_id;
}

void displayDetails(){
cout<<" User id : "<<user_id<<endl;
cout<<" password : "<<password<<endl;
}
}

```

```

cout<<" Payment id"<<payment_id<<endl;}

~paypal(){
cout<<"Delete paypal Default Constructor "<<endl<<endl;}

};

int main(){

    PaymentMethod P1; //default

    PaymentMethod P2("PYPL001","Pay Pal"); //over

    P2.displayDetails();


    paypal mypay1;

    paypal mypay2("CD001","credit and Debit","P001","23456789","PAY4567");

    mypay2.displayDetails();

    myPay1.~CreditDebit();


    return 0;
}


//Class for Payment

Class Payment{

protected:
string payment_type;
string payment_amount ;
string payment_id ;

public:
void displayDetails();
void displayPayMethod() ;
void getAdvertisementDetails() ;
void getPaymentMethods(Payment Method *PAY) ;
};

void Payment :: getAdvertisementDetails(Advertisement *A){
details = A-> getAdDetails() ;
}

```

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

int main () {
    Payment Pay ;

    Pay.getPaymentMethods() ;
    Pay.displayPaymentMethods() ;
    Pay.displayDetails() ;

    return 0 ;

}
```

//IT21489914 – Wijedasa H.A.S.A

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

//class for profile
class Profile
{
    private:
        string name;
        string address;
        string contactNumber;
        int No_of_Advertisements;
        string advertisementStatus;
```

```

public:
    Profile(string sellerName, string sellerAddress,string          contactNo, int advertiementsCount,
string Status );

    void displayAdvertisementList();

    void displayStatus(string advertisementStatus);

    ~Profile();
};

```

```

Profile:: Profile(string sellerName, string sellerAddress,string          contactNo, int
advertiementsCount, string Status ){

    strcpy(name,sellerName);

    strcpy(address,sellerAddress);

    strcpy(contactNumber,contactNo);

    No_of_Advertisements = advertisementCount;

    strcpy(advertisementStatus,Status);
}

```

```

int main() {

```

```

    Profile prof1("Ajith","No1,Borupana,Rathmalana","0713356734",3,"Posted");

```

```

}

```

```

#include <iostream>

```

```

#include<cstring>

```

```

using namespace std;

```

```

// class for advertisement
class Advertisement
{
    private:
        string advertisementID;
        string propertyType;
        float price;
        string location;
        string contactNumber;
        Seller *seller;

    public:
        Advertisement (string Ad_ID, string P_type, float Price,string Location,string contactNo,Seller
*nseller);

        void generateAdvertisement();
        void promoteAdvertisment();
        void updateAdvertisement();
        ~Advertisment();
};

int main() {
    Advertisement Ad("12334656898","House",19000000.00,"Malabe","0774567321","Ajith");
}

```

Registration Number	Name	Contribution
IT21391668	H.N.D. Madhubhashana	Registered User , Buyer , Seller
IT21357008	S.K.N.C. Karunanayaka	Payment Method , Credit/Debit card , Property
IT21489914	H.A.S.A.Wijedasa	Advertisement , Profile
IT21509186	H.G.L.R.Silva	Payment , Paypal
IT21575044	N.L.M. Fernando	Report , Appointment