Sri Lanka Institute of Information Technology

Topic : Online Property Sales System

Group no : MLB_02.02.03

Campus : Malabe

Submission Date: 17/05/2022

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
IT21219252	Senadheera S.A.A.D	072-5060868
IT21217586	Sevindi P.V.D	076-5592898
IT21220074	Kumari M.M.P.M	071-4932552
IT21217654	Herath H.M.R.M.K	077-0870361
IT21218644	Palihena N.V	

Table of Contents

1. Functional requirements

- The Guest should be able to access the website.
- The Guest should be able to access the system.
- The Guest should be able to search the property.
- The Guest should be able to change the language.
- The guest should be able to compare the properties.
- The registered customer should be able to rent a property.
- The registered customer should be able to categorize the reviews.
- The registered customer should be able to add the favourites.
- The registered customer should be able to make the payment.
- The registered customer should be able to buy the property.
- The registered customer should be able to add reviews.
- The registered customer should be able to request a virtual tour.
- The registered customer should be able to compare the properties.
- The registered customer should be able to update the account.
- The registered customer should be able to view the account.
- The registered customer should be able to remove the account.
- The registered customer should be able to add the account.
- The seller should be able to view seller account.
- The seller should be able to update seller account.
- The seller should be able to get a general estimate.
- The seller should be able to sell a property.
- The seller should be able to rent a property.
- The seller should be able to lease a property.
- The seller should be able to update the property details.
- The admin should be able to accept virtual tours.
- The admin should be able to blacklist the user account.
- The owner should be able to add or remove admins.
- The owner should be able to refund the scammed customer.

2. Classes

- Guest
- Registered customer
- Seller
- Admin
- Property
- Language
- Favorites
- Payment
- Reviews
- Scammed customer
- Virtual tour

CRC Cards

Guest class	
Responsibility	Collaborators
Register to the website Add Language	Language

Language class	
Responsibility	Collaborators
Select language	

Registered customer class	
Responsibility	Collaborators
Log in to the system Add Language View customer account Update customer account Remove customer account Add customer account Blacklist user	 Language Admin

Seller class	
Responsibility	Collaborators
Login Logout Add Language Display seller details View seller account Update seller account Blacklist user	 Language Admin

Admin class	
Responsibility	Collaborators
Login Logout View account	

Property class	
Responsibility	Collaborators
Display property details Buy property Rent property Sell property Lease property Update property details Get general estimate Compare property Search property	 Seller Seller Seller

Favourite class	
Responsibility	Collaborators
Add favourites Add note Remove favourites	registered customer registered customer registered customer

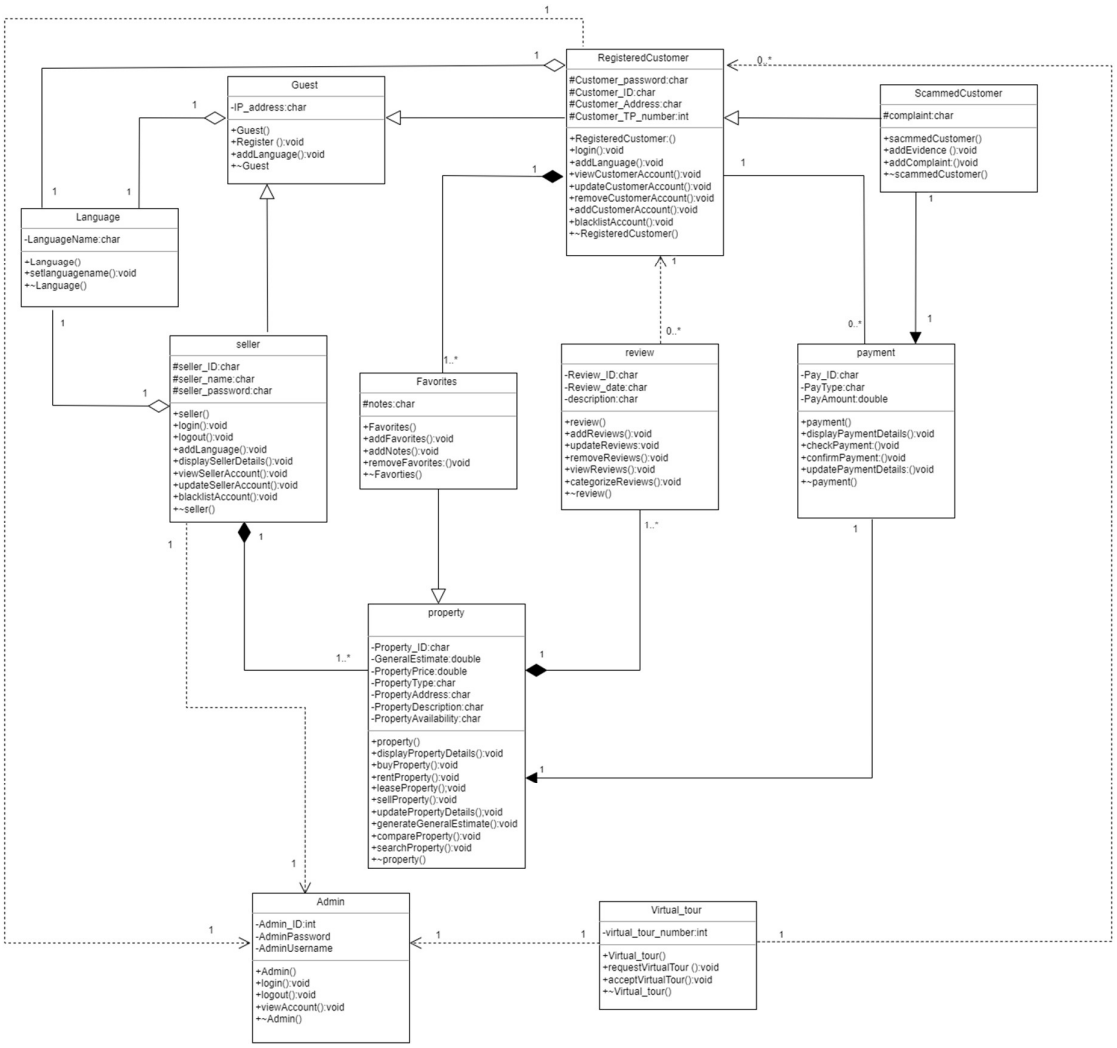
Payment class	
Responsibility	Collaborators
Display payment details Check payment Confirm payment Add payment Remove payment	Customer, Property Customer Customer Customer Customer

Scammed customer class	
Responsibility	Collaborators
Add evidence Add complaint	Payment

Reviews class	
Responsibility	Collaborators
Add reviews Update reviews Remove reviews View review Categories reviews	Registered customer,Property Registered customer Registered customer Registered customer

Virtual tour class	
Responsibility	Collaborators
Request virtual tour Accept virtual tour	Registered customer Admin

Class Diagram



Main.cpp

```
#include <iostream>
#include "Favorites.h"
#include "Scammedcustomer.h"
#include "Seller.h"
#include "virtualTour.h"

int main() {
 // Creation of Objects
 Guest *G = new Guest();
 Seller *S = new Seller();
 registeredCustomer *RC = new registeredCustomer();
 scammedCustomer *SC = new scammedCustomer();
 Admin *A = new Admin();
 Language *L = new Language();
 virtualTour *V = new virtualTour();
 favorites *F = new favorites();
 Property *P = new Property();
 Review *R = new Review();
 Payment *Py = new Payment();

 // Deleting Created Objects;
 delete G;
 delete S;
 delete RC;
 delete SC;
```

```
  delete A;

  delete L;

  delete V;

  delete F;

  delete P;

  delete R;

  delete Py;


  std::cout << " Program Completed Successfully\n";


  return 0;

}
```

Language.h

```
#ifndef LANGUAGE_H
#define LANGUAGE_H



class Language {
  private:
    char languageName[15];
  public:
    Language();
    Language(const char tLanguageName[]);
    void setLanguageName();
    ~Language();
};


#endif
```

Language.cpp

```
#include "Language.h"
#include <cstring>

Language::Language() {
  strcpy(languageName,"");
}

Language::Language(const char tLanguageName[]) {
  strcpy(languageName,tLanguageName);
}

void Language::setLanguageName() {

}

Language::~Language() {

}
```

Payment.h

```
#ifndef PAYMENT_H
#define PAYMENT_H

#include "RegisteredCustomer.h"
//#include "Property.h"
class Property;
class registeredCustomer;

class Payment {
  private:
    char pay_ID[5];
    char payType[20];
    double payAmount;
    Property *pty;
    registeredCustomer *rc;
  public:
    Payment();
    Payment(char tPay_ID[], char tPayType[], double tPayAmount);
    void displayPaymentDetails(Property *tPty, registeredCustomer *tRc, char tPay_ID[], char
tPayType[],double tPayAmount);
    void checkPayment();
    void confirmPayment();
    void updatePaymentDetails();
    ~Payment();
};
#endif
```

Payment.cpp

```
#include "Payment.h"
#include <cstring>

Payment::Payment() {
  strcpy(pay_ID,"");
  strcpy(payType,"");
  payAmount = 0;
}

Payment::Payment(char tPay_ID[], char tPayType[], double tPayAmount) {
  strcpy(pay_ID,tPay_ID);
  strcpy(payType,tPayType);
  payAmount = tPayAmount;
}

void Payment::displayPaymentDetails(Property *tPty, registeredCustomer *tRc, char tPay_ID[], char
tPayType[],double tPayAmount) {

}

void Payment::checkPayment() {

}

void Payment::confirmPayment() {
```

```
}

void Payment::updatePaymentDetails() {

}

Payment::~Payment() {

}
```

Property.h

```
#ifndef PROPERTY_H

#define PROPERTY_H


#include "review.h"

#define SIZE 2



class Property {

  protected:

    char property_ID[5];

    double propertyPrice;

    double generalEstimate;

    char propertyType[15];

    char propertyAddress[100];

    char propertyDescription[250];

    char propertyAvailability[15];

    Review *review[SIZE];

  public:

    Property();

    Property(const char tProperty_ID[], double tPropertyPrice, double tGeneralEstimate, const char tPropertyType[], const char tPropertyAddress[], const char tPropertyDescription[], const char tPropertyAvailability[], const char tReview0_ID[],const char tReview_date0[],const char tDescription0[], const char tReview1_ID[],const char tReview_date1[],const char tDescription1[] );

    void displayPropertyDetails();

    void buyProperrty();

    void rentProperty();
```

```
    void leaseProperty();
    void sellProperty();
    void updatePropertyDetails();
    void generateGeneralEstimate();
    void compareProperty();
    void searchProperty();
    ~Property();
};

#endif
```

Property.cpp

```
#include "Property.h"
#include <cstring>
#define SIZE 2



Property::Property() {
   strcpy(property_ID,"");
   propertyPrice = 0;
   generalEstimate = 0;
   strcpy(propertyType,"");
   strcpy(propertyAddress,"");
   strcpy(propertyDescription,"");
   strcpy(propertyAvailability,"");
   review[0] = new Review();
   review[1] = new Review();
}


Property::Property(const char tProperty_ID[], double tPropertyPrice, double tGeneralEstimate, const
char tPropertyType[], const char tPropertyAddress[], const char tPropertyDescription[], const char
tPropertyAvailability[], const char tReview0_ID[],const char tReview_date0[],const char
tDescription0[], const char tReview1_ID[],const char tReview_date1[],const char tDescription1[]) {
   strcpy(property_ID,tProperty_ID);
   propertyPrice = tPropertyPrice;
   generalEstimate = tGeneralEstimate;
   strcpy(propertyType,tPropertyType);
   strcpy(propertyAddress,tPropertyAddress);
```

```
    strcpy(propertyDescription,tPropertyDescription);
    strcpy(propertyAvailability,tPropertyAvailability);
    review[0] = new Review(tReview0_ID,tReview_date0,tDescription0);
    review[1] = new Review(tReview1_ID,tReview_date1,tDescription1);
}

void Property::displayPropertyDetails() {

}

void Property::buyProperrty() {

}

void Property::rentProperty() {

}

void Property::leaseProperty() {

}

void Property::sellProperty() {

}

void Property::updatePropertyDetails() {
```

```
}

void Property::generateGeneralEstimate() {

}

void Property::compareProperty() {

}

void Property::searchProperty() {

}

Property::~Property() {
  for(int i = 0;  i < SIZE; i++ ) {
    delete review[i];
  }
}
```

registeredCustomer.h

```
#ifndef REGISTEREDCUSTOMER_H
#define REGISTEREDCUSTOMER_H

#include "Guest.h"
#include "Language.h"
#include "Favorites.h"
#include "Admin.h"
#define SIZE 2

class favorites;
class Payment;

 class registeredCustomer:public Guest{

        protected:
                char customer_password[10];
                char customer_ID[5];
                char customer_Address[50];
                char customer_TP_number[10];
                favorites *CFavorites[SIZE];
    Language *sLanguage;
    Payment *CPayment[SIZE];
        public:
                registeredCustomer();
                registeredCustomer(const char tIP_address[], const char
tCustomer_password[],const char tCustomer_ID[],const char tCustomer_Address[],const char
```

```
tCustomer_TP_number[],const char tProperty_ID1[], double tPropertyPrice1, double
tGeneralEstimate1, const char tPropertyType1[], const char tPropertyAddress1[], const char
tPropertyDescription1[], const char tPropertyAvailibility1[],const char notes1[],const char
tProperty_ID2[], double tPropertyPrice2, double tGeneralEstimate2, const char tPropertyType2[],
const char tPropertyAddress2[], const char tPropertyDescription2[], const char
tPropertyAvailibility2[], const char notes2[] );


                void login();

                void viewCustomerAccount(Admin *admin);

                void updateCustomerAccount();

                void removeCustomerAccount();

                void addCustomerAccount();

                void blacklistAccount(Admin *admin);

    void addLanguage(Language *sLanguage);

    ~registeredCustomer();

 };


#endif
```

registeredCustomer.cpp

```
#include "RegisteredCustomer.h"

#include <cstring>

registeredCustomer::registeredCustomer()

{

strcpy(IP_address,"");

strcpy(customer_password,"");

strcpy(customer_ID,"");

strcpy(customer_Address,"");

strcpy(customer_TP_number,"");

CFavorites[0] = new favorites();

CFavorites[1] = new favorites();

}

registeredCustomer::registeredCustomer(const char tIP_address[], const char
tCustomer_password[],const char tCustomer_ID[],const char tCustomer_Address[],const char
tCustomer_TP_number[],const char tProperty_ID1[], double tPropertyPrice1, double
tGeneralEstimate1, const char tPropertyType1[], const char tPropertyAddress1[], const char
tPropertyDescription1[], const char tPropertyAvailibility1[],const char notes1[],const char
tProperty_ID2[], double tPropertyPrice2, double tGeneralEstimate2, const char tPropertyType2[],
const char tPropertyAddress2[], const char tPropertyDescription2[], const char
tPropertyAvailibility2[],const char notes2[])

{

strcpy(IP_address,tIP_address);
```

```
strcpy(customer_ID,tCustomer_ID);

strcpy(customer_password,tCustomer_password);

strcpy(customer_Address,tCustomer_TP_number);

CFavorites[0] = new favorites(tProperty_ID1, tPropertyType1, tPropertyAddress1,
tPropertyDescription1, tPropertyAvailibility1, tGeneralEstimate1, tPropertyPrice1, notes1);

CFavorites[1] = new favorites(tProperty_ID2, tPropertyType2, tPropertyAddress2,
tPropertyDescription2, tPropertyAvailibility2, tGeneralEstimate2, tPropertyPrice2, notes2);


}


void registeredCustomer::login(){


}


void  registeredCustomer::viewCustomerAccount(Admin *admin){


}


void registeredCustomer::updateCustomerAccount(){


}


void  registeredCustomer::removeCustomerAccount() {


 }


void registeredCustomer::addCustomerAccount() {
```

```
}

 void registeredCustomer::blacklistAccount(Admin *admin) {

}

void registeredCustomer::addLanguage(Language *sLanguage) {

}

registeredCustomer::~registeredCustomer(){
    for(int i = 0; i < SIZE; i++) {
       delete CFavorites[i];
  }
}
```

Review.h

```
#ifndef REVIEW_H
#define REVIEW_H

class registeredCustomer;

class Review{
        private:
                char review_ID[5];
                char review_date[10];
                char description[20];

        public:
                Review();
   Review(const char tReview_ID[],const char
   tReview_date[],const char tDescription[]);
                void addReview(registeredCustomer *C);
                void updateReview(registeredCustomer *registeredcustomer);
                void removeReview(registeredCustomer *registeredcustomer);
                void categorizeReviews();
                ~Review();
};

#endif
```

Review.cpp

```
#include"review.h"
#include <cstring>

Review::Review() {
  strcpy(review_ID,"");
  strcpy(review_date,"");
  strcpy(description,"");
}

Review::Review(const char tReview_ID[],const char tReview_date[],const char tDescription[])

{
  strcpy(review_ID,tReview_ID);
  strcpy(review_date,tReview_date);
  strcpy(description,tDescription);
}

void Review::addReview(registeredCustomer *registeredcustomer)
{

}
void Review::updateReview(registeredCustomer *registeredcustomer)
{

}
```

```
void Review::removeReview(registeredCustomer *registeredcustomer)
{


}
void Review::categorizeReviews()
{


}
Review::~Review(){


}
```

Seller.h

```
#ifndef SELLER_H

#define SELLER_H


#include "Guest.h"

#include "Property.h"

#include "Language.h"

#include "Admin.h"

#define SIZE 2



class Seller: public Guest{

private:

    char SellerID[5];

    char Sellername[50];

    char Sellerpassword[50];

    Property *sProperty[SIZE];

    Language *sLanguage;

public:

    Seller();

    Seller(const char tIP_address[], const char tSellerID[], const char tSellername[],const char tSellerpassword[],const char tProperty_ID1[], double tPropertyPrice1, double tGeneralEstimate1, const char tPropertyType1[], const char tPropertyAddress1[], const char tPropertyDescription1[], const char tPropertyAvailibility1[], const char tReview0_ID[],const char tReview_date0[],const char tDescription0[], const char tReview1_ID[],const char tReview_date1[],const char tDescription1[], const char tProperty_ID2[], double tPropertyPrice2, double tGeneralEstimate2, const char tPropertyType2[], const char tPropertyAddress2[], const char tPropertyDescription2[], const char tPropertyAvailibility2[],const char tReview2_ID[],const char tReview_date2[],const char tDescription2[], const char tReview3_ID[],const char tReview_date3[],const char tDescription3[]);
```

```
    void Login();
    void Logout();
    void addLanguage(Language *sLanguage);
    void ViewSellerAccount();
    void UpdateSellerAccount();
    void BlacklistSellerAccount(Admin *admin);
    void DisplaySellerDetails();
    ~Seller();
};

#endif
```

Seller.cpp

```
#include"Seller.h"

#include<cstring>

#define SIZE 2


Seller::Seller()

{

  strcpy(IP_address,"");

  strcpy(SellerID,"");

  strcpy(Sellername,"");

  strcpy(Sellerpassword,"");

  sProperty[0]= new Property();

  sProperty[1]= new Property();

}


Seller::Seller(const char tIP_address[], const char tSellerID[], const char tSellername[],const char
tSellerpassword[],const char tProperty_ID1[], double tPropertyPrice1, double tGeneralEstimate1,
const char tPropertyType1[], const char tPropertyAddress1[], const char tPropertyDescription1[],
const char tPropertyAvailibility1[], const char tReview0_ID[],const char tReview_date0[],const char
tDescription0[], const char tReview1_ID[],const char tReview_date1[],const char tDescription1[],
const char tProperty_ID2[], double tPropertyPrice2, double tGeneralEstimate2, const char
tPropertyType2[], const char tPropertyAddress2[], const char tPropertyDescription2[], const char
tPropertyAvailibility2[],const char tReview2_ID[],const char tReview_date2[],const char
tDescription2[], const char tReview3_ID[],const char tReview_date3[],const char tDescription3[])

{

  strcpy(IP_address,tIP_address);

  strcpy(SellerID,tSellerID);

  strcpy(Sellername,tSellername);

  strcpy(Sellerpassword,tSellerpassword);
```

```
  sProperty[0] = new Property(tProperty_ID1,tPropertyPrice1,tGeneralEstimate1,tPropertyType1,
tPropertyAddress1,tPropertyDescription1,tPropertyAvailibility1, tReview0_ID,tReview_date0,
tDescription0,tReview1_ID,tReview_date1, tDescription1);

  sProperty[1] = new
Property(tProperty_ID2,tPropertyPrice2,tGeneralEstimate2,tPropertyType2,tPropertyAddress2,tPro
pertyDescription2,tPropertyAvailibility2, tReview2_ID, tReview_date2,
tDescription2,tReview3_ID,tReview_date3, tDescription3);

}


void Seller::Login() {


}


void Seller::Logout(){


}



void Seller::addLanguage(Language *sLanguage){


}



void Seller::ViewSellerAccount(){


}


void Seller::UpdateSellerAccount(){
```

```
}



void Seller::BlacklistSellerAccount(Admin *admin) {


}


void Seller::DisplaySellerDetails() {


}


Seller::~Seller() {
  for(int i = 0; i < SIZE; i++) {
    delete sProperty[i];
  }
}
```

Admin.h

```
#ifndef ADMIN_H
#define ADMIN_H

class Admin {
private:
    char adminID[5];
    char adminPassword[50];
    char adminUsername[50];
public:
    Admin();
    Admin(const char tAdminID[], const char tAdminPassword[],const char tAdminUsername[]);
    void Login();
    void Logout();
    void ViewAccount();
    ~Admin();
};

#endif
```

Admin.cpp

```
#include "Admin.h"
#include <cstring>

Admin::Admin()
{
  strcpy(adminID,"");
  strcpy(adminUsername,"");
  strcpy(adminPassword,"");
}
Admin::Admin(const char tAdminID[], const char tAdminPassword[],const char tAdminUsername[])
{
  strcpy(adminID,tAdminID);
  strcpy(adminPassword,tAdminPassword);
  strcpy(adminUsername,tAdminUsername);
  }

void Admin::Login(){
}

void Admin::Logout(){
}

void Admin::ViewAccount(){
}
```

```
Admin::~Admin(){
}
```

Guest.h

```
#ifndef GUEST_H
#define GUEST_H

class Guest {
  protected:
    char IP_address [15];
  public:
    Guest();
    Guest(const char tIP_address[]);
    void Register();
    ~Guest();
};

#endif
```

Guest.cpp

```
#include "Guest.h"
#include <cstring>

Guest::Guest()
{
  strcpy( IP_address,"");
}

Guest::Guest(const char tIP_Address[])
{
  strcpy( IP_address,tIP_Address);
}

void Guest::Register()
{

}

Guest::~Guest() {

}
```

virtualTour.h

```
#include "Admin.h"
#include "RegisteredCustomer.h"



class virtualTour {
private:
    int virtualTourNumber;
public:
    virtualTour();
    virtualTour(int tVirtualTourNumber);
    void requestVirtualTour(registeredCustomer *rCustomer);
    void acceptVirtualTour(Admin *admin);
    ~virtualTour();
};
```

virtualTour.cpp

```
#include "virtualTour.h"
#include <cstring>

virtualTour::virtualTour()
{
  virtualTourNumber = 0;
}

virtualTour::virtualTour(int tVirtualTourNumber) {
  virtualTourNumber = tVirtualTourNumber;
}

 void virtualTour::requestVirtualTour(registeredCustomer *rCustomer) {

}

void virtualTour:: acceptVirtualTour(Admin *admin) {

}

virtualTour::~virtualTour(){

}
```

Scammedcustomer.h

```
#include"RegisteredCustomer.h"

#include "Payment.h"


class scammedCustomer:public registeredCustomer

{

        private:

                char complaint[100];

  Payment *pt;


        public:

                        scammedCustomer();

                        scammedCustomer(const char pComplaint[],const char tIP_address[], const char tCustomer_password[],const char tCustomer_ID[],const char tCustomer_Address[],const char tCustomer_TP_number[]);


    void addEvidence();

                        void addComplaint();

                        ~scammedCustomer();

};
```

ScammedCustomer.cpp

```
#include "Scammedcustomer.h"
#include <cstring>

scammedCustomer::scammedCustomer() {
  strcpy(IP_address,"");
  strcpy(customer_password,"");
  strcpy(customer_ID,"");
  strcpy(customer_Address,"");
  strcpy(customer_TP_number,"");
        strcpy(complaint," ");
}

scammedCustomer::scammedCustomer(const char tIP_address[],const char pComplaint[], const char tCustomer_password[],const char tCustomer_ID[],const char tCustomer_Address[],const char tCustomer_TP_number[])
{
  strcpy(IP_address,tIP_address);
  strcpy(customer_ID,tCustomer_ID);
  strcpy(customer_password,tCustomer_password);
  strcpy(customer_Address,tCustomer_Address);
        strcpy(complaint,pComplaint);
}

void scammedCustomer::addEvidence() {

}
```

```
void scammedCustomer::addComplaint() {


}


scammedCustomer::~scammedCustomer() {


}
```

Favourites.h

```
#ifndef FAVORITES_H

#define FAVORITES_H


#include "Property.h"


class favorites: public Property {

  private:

        char notes[50];


  public:

    favorites();

    favorites(const char pID[], const char pType[],const char pAddress[], const char pDescription[],const char pAvailability[], double pGeneralestimate,double pPrice, const char pNotes[]);

        void addFavorites();

        void addNotes();

        void removeFavorites();

        ~favorites();

};


#endif
```

Favorites.cpp

```
#include "Favorites.h"
#include <cstring>

favorites::favorites() {
        strcpy(notes,"");
        strcpy(property_ID,"");
  propertyPrice = 0;
  generalEstimate= 0;
  strcpy(propertyType,"");
  strcpy(propertyAddress,"");
  strcpy(propertyDescription,"");
  strcpy(propertyAvailability,"");
}

favorites::favorites(const char pID[], const char pType[],const char pAddress[], const char
pDescription[],const char pAvailability[], double pGeneralestimate,double pPrice, const char
pNotes[]){

        strcpy(notes,pNotes);
        strcpy(property_ID,pID);
        strcpy(propertyType,pType);
        strcpy(propertyAddress,pAddress);
        strcpy(propertyDescription,pDescription);
        strcpy(propertyAvailability,pAvailability);
        generalEstimate= pGeneralestimate;
        propertyPrice = pPrice;
```

```
}
void favorites::addFavorites(){

}

void favorites::addNotes(){

}

void favorites::removeFavorites(){

}

favorites::~favorites(){

}
```