Topic : Online Apartment Sales System

Group no : MLB_WD_CSNE_07.01_02

Campus : Malabe

Submission Date : 15/10/2021

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**
IT21092992	Jayakody L. R	0784727974
IT21073878	Senasinghe T. U	0703391474
IT20630898	Edirisinghe C. O	0766685712
IT21093296	Dissanayake N. S. S	0713173544
IT21092688	Bandara R. M. L. K	0774551426

Contents

System Requirements 3
Noun & Verb Analysis 4
Identified Classes 5
Noun & Verb Analysis 6
Methods 7
CRC Cards 8
Class Diagram (UML Notation) 12
Class Header Files 13
- GuestUser.h 13
- RegisteredCustomer.h 13
- Seller.h 14
- Buyer.h 14
- Apartment.h 15
- Staff.h 16
- Selling.h 17
- Booking.h 17
- Payment.h 18
- Report.h 18

Class Cpp Files 19
- GuestUser.cpp 19
- RegisteredCustomer.cpp 20
- Seller.cpp 21
- Buyer.cpp 22
- Apartment.cpp 23
- Staff.cpp 25
- Selling.cpp 26
- Booking.cpp 27
- Payment.cpp 28
- Report.cpp 29

Main program 30
- Main.cpp 30

System Requirements

- The System should function 24/7/365
- Guest users can overview the system, to use the system, they must register with the system by providing details such as Name, Address, NIC, Email, contact.
- Registered customers are of two types called sellers and buyers where they can log into the system by entering the correct username and password
- They can 'Buy' or 'Sell' apartments using the system
- Sellers should be able to add apartment details such as Location, price, facilities, and utility price to the system
- Details should be confirmed by the administrator staff
- Staff can delete or update the status of the apartment details
- System should generate a unique id for the apartment after confirming
- Before placing it on the system sellers must pay a small sale fee to the system
- After placing the sale, date of sale and sell ID is generated to the selling.
- Buyers should be able to filter apartment from type, price, location, ratings, and utility price (maintenance fee).
- Buyers can place a booking by selecting an apartment
- After booking, date of booking and booking ID is generated.
- Both the registered customers must do a payment
- Registered customers must enter their payment details like payment type, card details.
- After the payment 'Pay ID' is generated to the 'sell ID' of sellers and 'book ID' of buyers
- After the payment is confirmed by bank or other trusted resources a report of the selling details for sellers and booking details for buyers and apartment details and payment details is emailed

Noun & Verb Analysis

(NOUNS)

- The System should function 24/7/365
- Guest users can overview the system, to use the system, they must register with the system by providing details such as Name, Address, NIC, Email, contact.
- Registered customers are of two types called sellers and buyers where they can log into the system by entering the correct username and password
- They can 'Buy' or 'Sell' apartments using the system
- Sellers should be able to add apartment details such as Location, price, facilities, and utility price to the system
- Details should be confirmed by the administrator staff
- Staff can delete or update the status of the apartment details
- System should generate a unique id for the apartment after confirming
- Before placing it on the system sellers must pay a small sale fee to the system
- After placing the sale, date of sale and sell ID is generated to the selling.
- Buyers should be able to filter apartment from type, price, location, ratings, and utility price (maintenance fee).
- Buyers can place a booking by selecting an apartment
- After booking, date of booking and booking ID is generated.
- Both the registered customers must do a payment
- Registered customers must enter their payment details like payment type, card details.
- After the payment 'Pay ID' is generated to the 'sell ID' of sellers and 'book ID' of buyers
- After the payment is confirmed by bank or other trusted resources a report of the selling details for sellers and booking details for buyers and apartment details and payment details is emailed.

Identified Classes

- Guest User
- Registered Customer
- Seller
- Buyer
- Apartment
- Staff
- Selling
- Booking
- Payment

Reasons for rejecting other nouns

- **Redundant:** sellers, staff, buyers
- **An Event or an operation:**
- **Outside scope of system:** System, Bank, trusted resources
- **Meta-language:** they
- **An attribute:** Details (Name, Address, NIC, Email, Contact), Username, password, Apartment Details (type, Location, price, facilities, and utility price, utility price (maintenance fee)), status, unique ID (apartment ID), sale fee, date of sale, sell id, booking date, booking ID, payment type, card details, pay ID

Noun & Verb Analysis

(VERBS)

- The System should function 24/7/365
- Guest users can overview the system, to use the system, they must register with the system by providing details such as Name, Address, NIC, Email, contact.
- Registered customers are of two types called sellers and buyers where they can log into the system by entering the correct username and password
- They can 'Buy' or 'Sell' apartments using the system
- Sellers should be able to add apartment details such as Location, price, facilities, and utility price to the system
- Details should be confirmed by the administrator staff
- Staff can delete or update the status of the apartment details
- System should generate a unique id for the apartment after confirming
- Before placing it on the system sellers must pay a small sale fee to the system
- After placing the sale, date of sale and sell ID is generated to the selling.
- Buyers should be able to filter apartment from type, price, location, ratings, and utility price (maintenance fee).
- Buyers can place a booking by selecting an apartment
- After booking, date of booking and booking ID is generated.
- Both the registered customers must do a payment
- Registered customers must enter their payment details like payment type, card details.
- After the payment 'Pay ID' is generated to the 'sell ID' of sellers and 'book ID' of buyers
- After the payment is confirmed by bank or other trusted resources a report of the selling details for sellers and booking details for buyers and apartment details and payment details is emailed.

Methods

- Guest User
  - Register to the system by providing details
    View the system
- Registered Customer
  - Login to the system by entering details
- Seller
  - Sell apartment,
    Place a selling,
    Pay the sell fee
- Buyer
  - Buy apartments
    Search apartments by filtering requirements
    Place booking
    Selecting apartments
    Do payment for apartment
- Apartment
  - Generate apartment ID
    Add apartment details
    Delete and update apartment details
- Staff
  - Log into the system,
    Confirm apartment details
    Manage Apartment details
- Selling
  - Generate sell ID
    Update the system
    Calculate sell price
- Booking
  - Generate book ID
    Check availability of apartments
    Calculate booking price
- Payment
  - Generate pay ID
    Check payment details
    Confirm payments

CRC Cards

Guest User	
Responsibility	**Collaborators**
Register to the system	
Allow to view the Apartments	Apartment

Registered Customer	
Responsibility	**Collaborators**
Can view the Apartments	Apartment
Add and update customer details	

Seller	
Responsibility	**Collaborators**
Log in to the system	Registered Customer
Sell apartments	Apartments
Pay the sell fee	

Buyer	
Responsibility	**Collaborators**
Log in to the system	Registered Customer
Buy apartments	Apartment
Search apartments	Apartment

Apartment	
Responsibility	**Collaborators**
Add apartment Details	Seller
Delete Apartment Details	Staff
Update Apartment Details	Seller, Staff

Selling	
Responsibility	**Collaborators**
Place selling	
Update the system	Apartment
Calculate the sell fee	

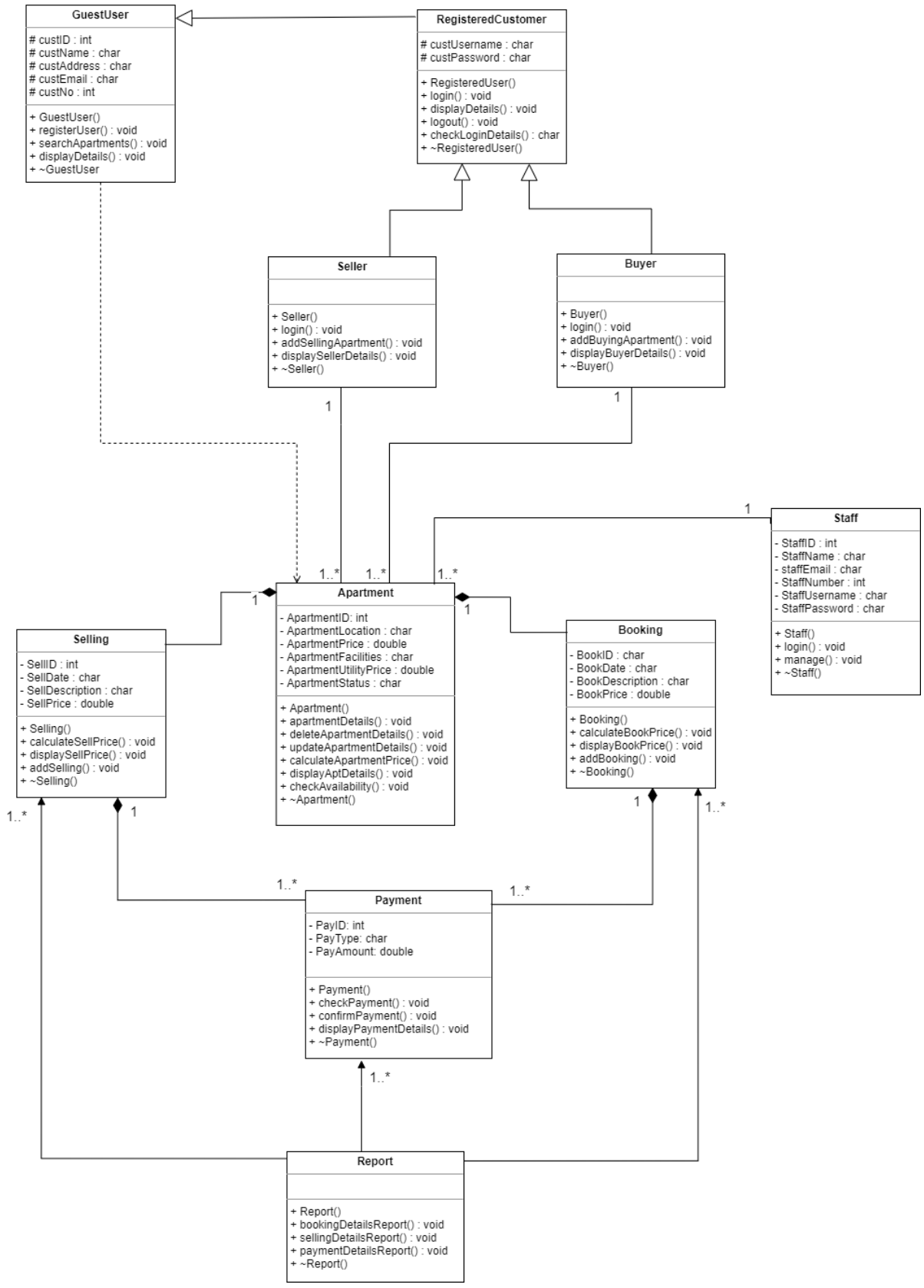
Staff	
Responsibility	**Collaborators**
Login to system	
Confirm apartment details	Apartment

Booking	
Responsibility	**Collaborators**
Place a booking	
Check availability of apartments	Apartment
Calculate the apartment price	

Report	
Responsibility	**Collaborators**
Generate Booking details	Booking
Generate selling details	Selling
Generate Payment details	Payment

Payment	
Responsibility	**Collaborators**
Make a new payment	
Generate Pay ID	Selling, Booking
Check payment details	Seller, Buyer
Confirm payment details	

Class Diagram (UML Notation)



Class Header Files

GuestUser.h

```
#include "Apartment.h"
class GuestUser
{
protected:
	int custID;
	char custName[20];
	char custAddress[30];
	char custEmail[30];
	char custphoneNumber[10];

public:
	GuestUser();
	GuestUser(int pcustid, const char pcustName[], const char
pcustAddress[], const char pcustEmail[] ,const char custPHno[]);
	void searchApartments(Apartment* pApt);
	void registerUser();
	virtual void displayDetails();
	~GuestUser();
};
```

RegisteredCustomer.h

```
#include"GuestUser.h"
class RegisteredCustomer :public GuestUser
{
protected:
	char custUsername[10];
	char custPassword[10];

public:
	RegisteredCustomer();
	RegisteredCustomer(const char pcustUsername[], const char
pcustPassword[], int pcustid, const char pcustName[], const char
pcustAddress[], const char pcustEmail[], const char pcustNo[]);
	void displayDetails();
	void login();
	void logout();
	char checkLoginDetails();
	~RegisteredCustomer();
};
```

Seller.h

```
#include "RegisteredCustomer.h"
#include "Apartment.h"
#define SIZE 5
class Seller :public RegisteredCustomer
{
private:
    int noOfApartments;

    Apartment* sellApt[SIZE];

public:
    Seller();
    Seller(const char usName[], const char usPwd[], int id, const
char name[], const char address[], const char email[], const char
telno[], int pnoOfApartments);
    void addSellingApartment(Apartment* psellApt);
    void login();
    void displaySellerDetails();
    ~Seller();
};
```

Buyer.h

```
#include "RegisteredCustomer.h"
#include "Apartment.h"
#define SIZE 5
class Buyer : public RegisteredCustomer
{
private:
    int noOfApartments;

    Apartment* buyApt[SIZE];

public:
    Buyer();
    Buyer(const char usName[], const char usPwd[], int id, const
char name[], const char address[], const char email[], const char
telno[],int pnoOfApartments);
    void addBuyingApartment(Apartment* pbuyApt);
    void login();
    void displayBuyerDetails();
    ~Buyer();
};
```

Apartment.h

```
#include "Booking.h"
#include "Selling.h"
#include "Seller.h"
#include "Buyer.h"
#include "Staff.h"

#define SIZE1 2
#define SIZE2 2


class Apartment
{
private:
	int apartmentID;
	char apartmentLoctaion[50];
	double apartmentPrice;
	char apartmentFacilities[50];
	double apartmentUtilityPrice;
	char apartmentStatus[50];
	int count = 0;

	Booking* book[SIZE1];
	Selling* sell[SIZE2];
	Seller* seller;
	Buyer* buyer;
	Staff* staff;

public:
	Apartment();
	Apartment(int sell1, int sell2, int book1, int book2,Seller*
pseller, Buyer* pbuyer,Staff* pstaff);
	void apartmentDetails(int aptID, const char aptLocation,
double aptPrice, const char aptFacility, double aptUtiPrice, const
char aptStatus , Seller* pseller , Buyer* pbuyer , Staff* pstaff);
	void deleteApartmentDetails();
	void updateApartmentDetails();
	void calculateApartmentPrice();
	void displayAptDetails();
	void checkAvailability();
	~Apartment();

};
```

Staff.h

```
#include "Apartment.h"
#define SIZE 5
class Staff
{
private:
	int staffID;
	char staffName[20];
	char staffEmail[20];
	char staffNumber[10];
	char staffUsername[20];
	char staffPassword[20];

	Apartment* apt[SIZE];

public:
	Staff();
	Staff(int pstaffID, const char pstaffName[], const char
pstaffEmail[], const char pstaffNumber[], const char
pstaffUsername[], const char pstaffPassword[]);
	void login(const char stfUsername, const char stfPsword );
	void manage( Apartment* papt);
	~Staff();

};
```

Selling.h

```
#include"Payment.h"
#define SIZE 2

class Selling {
private:
	int SelID;
	char SelDate[20];
	char SelDescription[50];
	double SelPrice;
	int count = 0;

	Payment* payment[SIZE];

public:
	Selling();
	Selling(int pselID, const char pseldate[], const char
pseldescription[], double pselprice, int pay1, int pay2);
	void calculateSellPrice(int id, const char pType[], double
pAmt);
	void displaySelPrice();
	void addSelling();
	~Selling();
};
```

Booking.h

```
#include"Payment.h"
#define SIZE 2

class Booking {
private:
	char BookID[10];
	char BookDate[20];
	char BookDescription[50];
	double BookPrice;
	int count = 0;

	Payment* payment[SIZE];
public:
	Booking();
	Booking(const char pbookID[], const char pbookDate[],const
char pbookDescription[],double pbookPrice,int pay1, int pay2);
	void calculateBookPrice(int id, char pType[], double pAmt);
	void displayBookPrice();
	void addBooking();
	~Booking();
};
```

Payment.h

```
class Payment
{
private:
	int payID;
	char payType[20];
	double payAmount;

public:
	Payment();
	Payment(int pID,const char ppayType[],double ppayAmount);
	void checkPayment();
	void confirmPayment();
	void displayPaymentDetails();
	~Payment();
};
```

Report.h

```
#include"Selling.h"
#include"Booking.h"
#include"Payment.h"
#define SIZE1 5
#define SIZE2 5
#define SIZE3 5

class Report
{
private:
	Booking* book[SIZE1];
	Selling* sell[SIZE2];
	Payment* pay[SIZE3];
public:
	Report();
	Report(Booking* pbbok[], Selling* psell[], Payment* ppay[]);
	void bookingDetailsReport();
	void sellingDetailsReport();
	void paymentDetailsReport();
	~Report();
};
```

Class Cpp Files

GuestUser.cpp

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

GuestUser::GuestUser()
{
	custID = 0;
	strcpy(custName, "");
	strcpy(custAddress, "");
	strcpy(custEmail, "");
	strcpy(custphoneNumber, "0000000000");
}

GuestUser::GuestUser(int pcustid, const char pcustName[], const char
pcustAddress[], const char pcustEmail[], const char custPHno[])
{
	custID = pcustid;
	strcpy(custName, pcustName);
	strcpy(custAddress, pcustAddress);
	strcpy(custEmail, pcustEmail);
	strcpy(custphoneNumber, custPHno);

}

void GuestUser::searchApartments(Apartment* pApt)
{

}

void GuestUser::registerUser()
{

}

void GuestUser::displayDetails()
{

}

GuestUser::~GuestUser()
{
	//Destructor
}
```

RegisteredCustomer.cpp

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

RegisteredCustomer::RegisteredCustomer()
{
	strcpy(custUsername, "");
	strcpy(custPassword, "");
}

RegisteredCustomer::RegisteredCustomer(const char pcustUsername[],
const char pcustPassword[], int pcustid, const char pcustName[],
const char pcustAddress[], const char pcustEmail[], const char
pcustNo[]) : GuestUser(pcustid, pcustName, pcustAddress, pcustEmail,
pcustNo)
{
	strcpy(custUsername, pcustUsername);
	strcpy(custPassword, pcustPassword);
}

void RegisteredCustomer::displayDetails()
{

}

void RegisteredCustomer::login()
{

}

void RegisteredCustomer::logout()
{

}

char RegisteredCustomer::checkLoginDetails()
{
	return 0;
}

RegisteredCustomer::~RegisteredCustomer()
{
	//Destructor
}
```

Seller.cpp

```
#include "Seller.h"


Seller::Seller()
{
	noOfApartments = 0;
}

Seller::Seller(const char usName[], const char usPwd[], int id,
const char name[], const char address[], const char email[], const
char telno[], int pnoOfApartments) :RegisteredCustomer(usName,
usPwd, id, name, address, email, telno)
{
	noOfApartments = pnoOfApartments;
}

void Seller::addSellingApartment(Apartment* psellApt)
{
	if (noOfApartments < SIZE)
	{
		sellApt[noOfApartments] = psellApt;
		noOfApartments++;
	}
}

void Seller::login()
{
}

void Seller::displaySellerDetails()
{
}

Seller::~Seller()
{
	//Destructor
}
```

Buyer.cpp

```
#include "Buyer.h"

Buyer::Buyer()
{
	noOfApartments = 0;
}

Buyer::Buyer(const char usName[], const char usPwd[], int id, const
char name[], const char address[], const char email[], const char
telno[], int pnoOfApartments):RegisteredCustomer(usName,usPwd, id,
name, address,  email,  telno)
{
	noOfApartments = pnoOfApartments;
}

void Buyer::addBuyingApartment(Apartment* pbuyApt)
{
	if (noOfApartments < SIZE)
	{
		buyApt[noOfApartments] = pbuyApt;
		noOfApartments++;
	}
}

void Buyer::login()
{

}

void Buyer::displayBuyerDetails()
{

}

Buyer::~Buyer()
{
	//Destructor
	for (int i = 0; i < SIZE; i++)
	{
		delete buyApt[i];
	}
}
```

Apartment.cpp

```
#include "Apartment.h"
#define SIZE1 2
#define SIZE2 2

Apartment::Apartment()
{

}

Apartment::Apartment(int sell1, int sell2, int book1, int book2,
Seller* pseller, Buyer* pbuyer, Staff* pstaff)
{
	sell[0] = new Selling(sell1);
	sell[1] = new Selling(sell2);

	book[0] = new Booking(book1);
	book[1] = new Booking(book2);

	seller = pseller;
	buyer = pbuyer;
	staff = pstaff;
}

void Apartment::apartmentDetails(int aptID, const char aptLocation,
double aptPrice, const char aptFacility, double aptUtiPrice, const
char aptStatus, Seller* pseller, Buyer* pbuyer, Staff* pstaff)
{

}

void Apartment::deleteApartmentDetails()
{

}

void Apartment::updateApartmentDetails()
{

}
```

```
void Apartment::calculateApartmentPrice()
{

}

void Apartment::displayAptDetails()
{

}

void Apartment::checkAvailability()
{

}

Apartment::~Apartment()
{
	//Destructor
	for (int i = 0; i < SIZE1; i++)
	{
		delete book[i];
	}

	for (int i = 0; i < SIZE2; i++)
	{
		delete sell[i];
	}
}
```

Staff.cpp

```
//#include "Apartment.h"
#include "Staff.h"
#include<cstring>

Staff::Staff()
{
	staffID = 0;
	strcpy(staffName, "");
	strcpy(staffEmail, "");
	strcpy(staffNumber, "0000000000");
	strcpy(staffUsername, "");
	strcpy(staffPassword, "");
}

Staff::Staff(int pstaffID, const char pstaffName[], const char
pstaffEmail[], const char pstaffNumber[], const char
pstaffUsername[], const char pstaffPassword[])
{
	staffID = pstaffID;
	strcpy(staffName, pstaffName);
	strcpy(staffEmail, pstaffEmail);
	strcpy(staffNumber, pstaffNumber);
	strcpy(staffUsername, pstaffUsername);
	strcpy(staffPassword, pstaffPassword);
}

void Staff::login(const char stfUsername, const char stfPsword)
{
}

void Staff::manage(Apartment* papt)
{
}

Staff::~Staff()
{
	//Destructor
	for (int i = 0; i < SIZE; i++)
	{
		delete apt[i];
	}
}
```

Selling.cpp

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

Selling::Selling()
{
	SelID = 0;
	strcpy(SelDate, "");
	strcpy(SelDescription, "");
	SelPrice = 0;
}

Selling::Selling(int pselID, const char pseldate[], const char
pseldescription[], double pselprice, int pay1, int pay2)
{
	SelPrice = pselprice;
	strcpy(SelDate, pseldate);
	strcpy(SelDescription, pseldescription);
	SelID = pselID;
}

void Selling::calculateSellPrice(int id, const char pType[], double
pAmt)
{
	if (count < SIZE)
	{
		payment[count] = new Payment(id, pType, pAmt);
		count++;
	}
}

void Selling::displaySelPrice()
{
}

void Selling::addSelling()
{
}

Selling::~Selling()
{
	//Destructor
	for (int i = 0; i < SIZE; i++)
	{
		delete payment[i];
	}
}
```

Booking.cpp

```
#include "Payment.h"
#include "Booking.h"
#include<cstring>
Booking::Booking()
{
	strcpy(BookID, "");
	strcpy(BookDate, "");
	strcpy(BookDescription, "");
	BookPrice = 0;
}

Booking::Booking(const char pbookID[],const char pbookDate[], const
char pbookDescription[], double pbookPrice, int pay1, int pay2)
{
	strcpy(BookID, pbookID);
	strcpy(BookDate, pbookDate);
	strcpy(BookDescription, pbookDescription);
	BookPrice = 0;
}

void Booking::calculateBookPrice(int id, char pType[], double pAmt)
{
	if (count < SIZE)
	{
		payment[count] = new Payment(id, pType, pAmt);
		count++;
	}
}
void Booking::displayBookPrice()
{

}
void Booking::addBooking()
{

}
Booking::~Booking()
{
	//Destructor
	for (int i = 0; i < SIZE; i++)
	{
		delete payment[i];
	}
}
```

Payment.cpp

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

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

Payment::Payment(int pID, const char ppayType[], double ppayAmount)
{
	payID = pID;
	strcpy(payType, ppayType);
	payAmount = ppayAmount;
}

void Payment::checkPayment()
{

}

void Payment::confirmPayment()
{

}

void Payment::displayPaymentDetails()
{

}

Payment::~Payment()
{
	//Destructor
}
```

Report.cpp

```
#include "Report.h"
Report::Report()
{
	for (int i = 0; i < SIZE1; i++)
	{
		book[i] = 0;
	}
	for (int j = 0; j < SIZE2; j++)
	{
		sell[j] = 0;
	}
	for (int k = 0; k < SIZE3; k++)
	{
		pay[k] = 0;
	}
}

Report::Report(Booking* pbbok[], Selling* psell[], Payment* ppay[])
{
	for (int i = 0; i < SIZE1; i++)
	{
		book[i] = pbbok[i];
	}
	for (int j = 0; j < SIZE2; j++)
	{
		sell[j] = psell[j];
	}
	for (int k = 0; k < SIZE3; k++)
	{
		pay[k] = ppay[k];
	}
}

void Report::bookingDetailsReport()
{
}

void Report::sellingDetailsReport()
{
}

void Report::paymentDetailsReport()
{
}

Report::~Report()
{
	//Destructor
	for (int i = 0; i < SIZE1; i++)
	{
		delete book[i] ;
	}
	for (int j = 0; j < SIZE2; j++)
	{
		delete sell[j] ;
	}
	for (int k = 0; k < SIZE3; k++)
	{
		delete pay[k] ;
	}
}
```

Main program

Main.cpp

```
#include "Booking.h"
#include "Selling.h"
#include "Seller.h"
#include "Buyer.h"
#include "Staff.h"
#include "Apartment.h"
#include "GuestUser.h"
#include "Payment.h"
#include "RegisteredCustomer.h"
#include "Report.h"

#include <iostream>
using namespace std;

int main()
{


    //---- Object creation ------


    GuestUser* rg = new RegisteredCustomer(); // Object -
RegisteredCustomer class

    RegisteredCustomer* seller = new Seller(); // Object  - seller
class

    RegisteredCustomer* buyer = new Buyer(); // Object - buyer class

    Apartment* apt = new Apartment(); // Object - Apartment class

    Selling* selling = new Selling(); // Object  - Selling class

    Booking* booking = new Booking(); // Object  - Booking class

    Staff* staff = new Staff(); // Object  - Staff class

    Report* report = new Report(); // Object - Report class
```

```
    //----Method Calling------
    rg->login();
    rg->displayDetails();


    seller->login();
    seller->displaySellerDetails();

    buyer->login();
    buyer->displayBuyerDetails();

    apt->updateAptDetails();
    apt->checkAvailability();

    selling->addSelling();
    selling->displaySelPrice();

    booking->addBooking();
    booking->displayBookPrice();

    report->bookingDetailsReport();
    report->sellingDetailsReport();
    report->paymentDetailsReport();


    //----Delete Dynamic objects------
    delete rg;
    delete seller;
    delete buyer;
    delete apt;
    delete selling;
    delete booking;
    delete report;


    return 0;
}
```