Topic : **Online Apartment Sales System**

Group no : **MLB_10.01_01**

Campus : **Malabe**

Submission Date : **05/18/2022**

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**
IT21185748	D.N.M. Dissanayake	0703847703
IT21186042	E.M.D.K.L. Ekanayake	0703248686
IT21186660	G.M.J.K. Gunathilake	0768953551
IT21185366	W.B.M.S. Sriyarathne	0756301144
IT21185434	E.M.T.S. Edirisingha	0754924558

Table of Contents

1) The System Requirements

- The System should function 24/7.
- Guest users can overview the system, to use the system, they must register with the system by supplying details such as Name, Address, NIC, Email, Contact.
- Registered customers can also be buyers, and both can log into the system by entering the correct username and password.
- They can ‘Buy’ apartments using the system.
- Staff should be able to add apartments details such as Location, Price, Facilities, and Conditions to the system.
- Details should be confirmed by the system managers.
- Administrator can remove or update the status of the apartment details.
- System should generate a unique ID for the apartment after confirming.
- 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, and Ratings.
- Buyers can place a reserving by selecting an apartment.
- After reserving, date of reserving and reserving ID is generated.
- Buyers must do payment for reserving.
- They must enter their payment details like payment type, card details.
- After the payment ‘Pay ID’ is generated to the ‘Sell ID’ of staff and ‘Reserve ID’ of buyers.
- After the payment is confirmed by bank or other trusted resources a report of the selling details for staff and reserving details for buyers and apartment details and payment details are emailed.

2) Identified Classes

- Guest user
- Registered Customer
- Buyer
- Staff
- Administrator
- Apartment
- Selling
- Reserving
- Payment
- Report

➢ Reasons for rejecting other nouns

- **Redundant:** Buyers, System managers
- **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, Ratings, Facilities and Conditions), status, unique ID (apartment ID), date of sale, sell ID, reserving date, reserving ID, payment type, card details, reserve ID, Pay ID

3) Methods

- Guest user – Register to the system by suppling details
  View the system

- Registered Customer – Login to the system by entering details

- Buyer - Buy apartments

  Search apartments by filtering requirements
  Place reserving
  Selecting apartments
  Do payment for apartment

- Staff - Log into the system
  Sell apartments
  Place selling

- Administrator – Log into the system
  Confirm apartment details
  Manage apartment details

- Apartment – Generate apartment ID
  Add apartment details
  Delete and update apartment details

- Selling – Generate sell ID
  Update the system
  Calculate sell price

- Reserving – Generate reserve ID
  Check availability of apartments
  Calculate reserving price

- Payment – Generate pay ID
  Check payment details
  Confirm payment

- Report – Generate reserving details
  Generate selling details
  Generate payment details

4) 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 view customer details	

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

Staff	
Responsibility	**Collaborators**
Log to system	
Sell apartments	Apartment

Administrator	
Responsibility	**Collaborators**
Log to system	
Confirm apartment details	Apartment

Apartment	
Responsibility	**Collaborators**
Add apartment details	Staff
Delete apartment details	Administrator
Update apartment details	Staff, Administrator

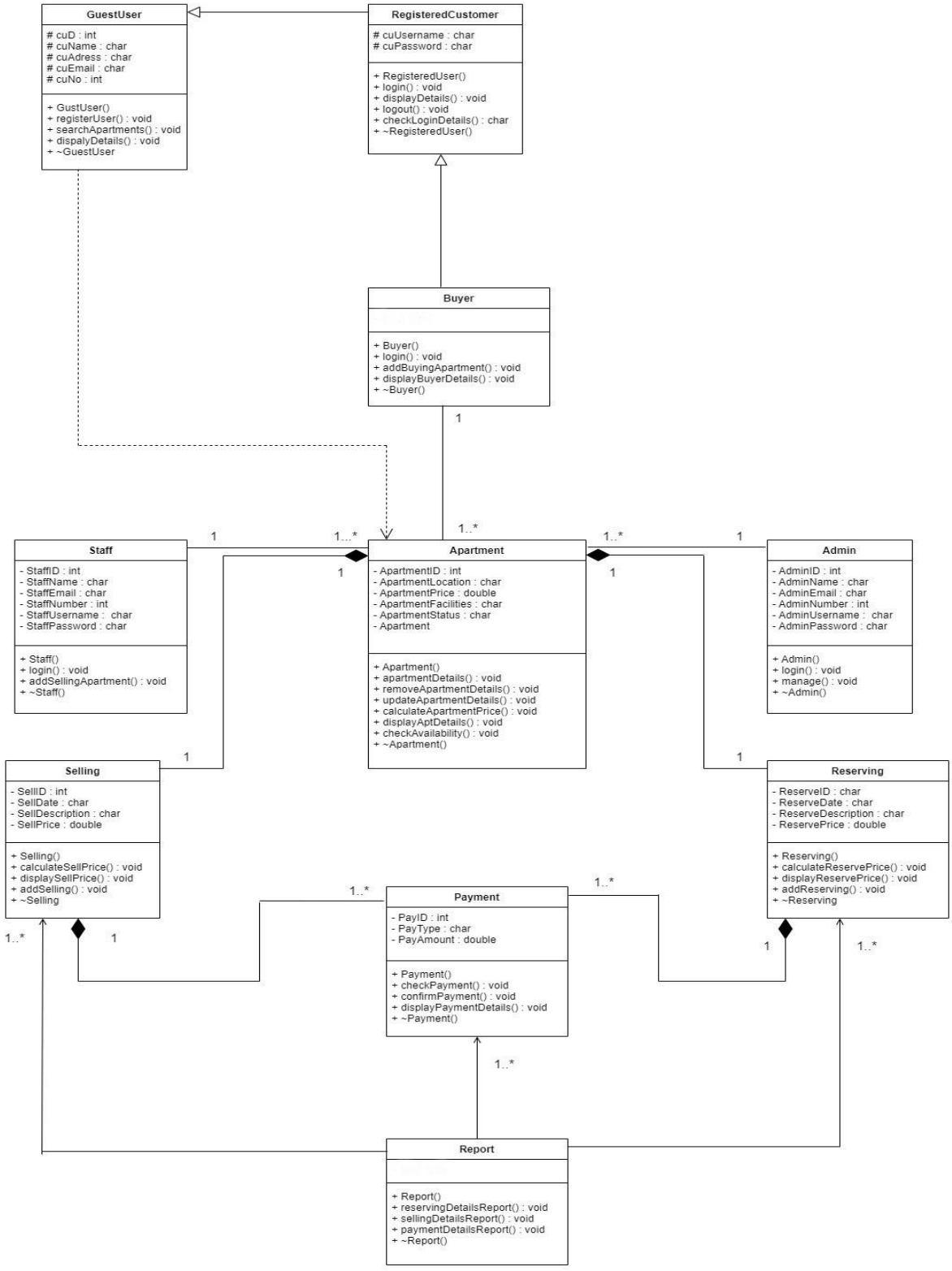
Selling	
Responsibility	**Collaborators**
Place selling	
Update the system	Apartment
Calculate the sell price	

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

Payment	
Responsibility	**Collaborators**
Make new payment	
Generate Pay ID	Selling, Reserving
Check payment details	Staff, Buyer
Confirm payment details	

Report	
Responsibility	**Collaborators**
Generate reserving details	Reserving
Generate selling details	Selling
Generate payment details	Payment

5) Class Diagram (UML Notation)



6) Class Header files

- **GuestUser.h**

```
#include "Apartment.h"


class GuestUser
{
protected:
	int cuID;
	char cuName[20];
	char cuAddress[30];
	char cuEmail[30];
	char cuphoneNumber[10];


public:
	GuestUser();
	GuestUser(int pcuid, const char pcuName[], const char pcuAddress[], const char pcuEmail[], const char cuPHno[]);
	void searchApartments(Apartment* pApt);
	void registerUser();
	void login();
	virtual void displayDetails();
	~GuestUser();


};
```

RegisteredCustomer.h

```
#pragma once
#include"GuestUser.h"

class RegisteredCustomer : public GuestUser
{
protected:
	char cuUsername[10];
	char cuPassword[10];

public:
	RegisteredCustomer();
	RegisteredCustomer(const char pcuUsername[], const char
pcuPassword[], int pcuid, const char pcuName[], const char pcuAddress[],
const char pcuEmail[], const char pcuNo[]);
	void displayDetails();
	void displayBuyerDetails();
	void login();
	void logout();
	char checkLoginDetails();
	~RegisteredCustomer();
};
```

Buyer.h

```
#include "RegisteredCustomer.h"
#include "Apartment.h"

#define SIZE 3

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();

};
```

- **Staff.h**

```
#pragma once
#include "Apartment.h"
#define SIZE
class Staff {

private:
	int StaffiD;
	char StaffName[20];
	char StaffEmail[20];
	int StaffNumber;
	char StaffUsername[20];
	char Password[20];
	Apartment* Apt[SIZE];

public:
	Staff();
	Staff(int id, char sname[], char email, int number, char uname[], char
pword[]);
	void login(char name[], char password[]);
	void addSellingApartment(Apartment* Apt[SIZE]);
	~Staff();

};
```

- **Administrator.h**

```
#pagma once

#include "Apartment.h"

#define SIZE


class Admin {


private:

	int AdminID;

	char AdminName[20];

	char AdminEmail[20];

	int AdminNumber;
```

```
        char AdminUsername[20];
        char AdminPassword[20];
        Apartment * Apt[SIZE];


public:
        Admin();
        Admin(int id, char aname[], char email, int number, char uname[], char
pword[]);
        void login(char name[], char password[]);
        void manage(Apartment* Apt[SIZE]);
        ~Admin();
};
```

- **Apartment.h**

```
#pragma once
#include <iostream>
#include "Selling.h"
#include "Reserving.h"
#include "Staff.h"
#include "Admin.h"
#define SIZE 3

class Apartment {

private:

    int ApartmentID;
    char ApartmentLocation[20];
    char ApartmentFacilities[20];
    char ApartmentStatus[10];
    Selling* sell;
    Reserving* reserve[SIZE];
    Staff* staff;
    Admin* admin;

public:

    Apartment();
    Apartment(int id, const char location[], const char facilities[], const
char status[], const char preserveID1[], const char preserveDate1[], const
char preserveDescription1[], double preservePrice1, const char
preserveID2[], const char preserveDate2[], const char
preserveDescription2[], double preservePrice2, Admin* gadmin, Staff*
gstaff);
```

```
        void Apartmentdetails();
        void calculateApartmentprice();
        void displayApartmentdetails();
        void checkAvalibility();
        void deleteApartmentDetails();
        void updateApartmentDetails();

        ~Apartment();

    };
```

- **Selling.h**

```
#include <iostream>
#pagma once
include "Payment.h"

class Selling {

private:
        int SellID;
        char SellDate[20];
        char SellDescription[20];
        double Sellprice;
        Payment* pay;

public:
        Selling();
        Selling(int id, char date[], char description[], double price);
        void CalculateSellPrice();
        void DisplaySellPrice();
        void addselling();
        void addpayment();

        ~Selling();


        };
```

- **Reserving.h**

```
#include"Payment.h"


#define SIZE 3


class Reserving {
private:
	char ReserveID[10];
	char ReserveDate[20];
	char ReserveDescription[50];
	double ReservePrice;
	int count = 0;
	Payment* payment[SIZE];


public:
	Reserving();
	Reserving(const char preserveID[], const char preserveDate[], const char
preserveDescription[], double preservePrice, int PID, const char PpayType[],
double PpayAmount);
	void calculateReservePrice(int id, const char pType[], double pAmt);
	void displayReservePrice();
	void addReserving();
	~Reserving();


};
```

- **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**

```
#pragma once
#include"Selling.h"
#include"Reserving.h"
#include"Payment.h"
#define SIZE1 5
#define SIZE2 5
#define SIZE3 5

class Report
{
private:
	Reserving* reserve[SIZE1];
	Selling* sell[SIZE2];
	Payment* pay[SIZE3];

public:
	Report();
	Report(Reserving* preserve[], Selling* psell[], Payment* ppay[]);
	void ReservingDetailsReport();
	void sellingDetailsReport();
	void paymentDetailsReport();
	Report();

};
```

7) Class Cpp Files

- **GuestUser.cpp**

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


GuestUser::GuestUser()
{
	cuID = 0;
	strcpy_s(cuName, "");
	strcpy_s(cuAddress, "");
	strcpy_s(cuEmail, "");
	strcpy_s(cuphoneNumber, "0000000000");
}
GuestUser::GuestUser(int pcuid, const char pcuName[], const char pcuAddress[],
const char pcuEmail[], const char cuPHno[])
{
	cuID = pcuid;
	strcpy_s(cuName, pcuName);
	strcpy_s(cuAddress, pcuAddress);
	strcpy_s(cuEmail, pcuEmail);
	strcpy_s(cuphoneNumber, cuPHno);
}
void GuestUser::searchApartments(Apartment* pApt)
{
}
void GuestUser::registerUser()
{
}
void GuestUser::login() {
```

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

- **RegisteredCustomer.cpp**

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

RegisteredCustomer::RegisteredCustomer()
{
	strcpy_s(cuUsername, "");
	strcpy_s(cuPassword, "");
}
RegisteredCustomer::RegisteredCustomer(const char pcuUsername[], const
char pcuPassword[], int pcuid, const char pcuName[], const char
pcuAddress[], const char pcuEmail[], const char pcuNo[])

{
	strcpy_s(cuUsername, pcuUsername);
	strcpy_s(cuPassword, pcuPassword);
}
void RegisteredCustomer::displayDetails()
{
}
void RegisteredCustomer::displayBuyerDetails() {

}
void RegisteredCustomer::login()
{
}
void RegisteredCustomer::logout()
{
}
char RegisteredCustomer::checkLoginDetails()
{
	return 0;
}
RegisteredCustomer::~RegisteredCustomer()
{
```

```
        //Destructor
    }
```

- **Buyer.cpp**

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


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];
    }
}
```

- **Staff.cpp**

```
#include "Staff.h"
#include "Apartment.h"
#include <cstring>
#define SIZE

using namespace std;

Staff::Staff() {

   StaffiD = 0;
   strcpy(StaffName, "");
   strcpy(StaffEmail, "");
   StaffNumber = 0;
   strcpy(StaffUsername, "");
   strcpy(Password, "");

}

Staff::Staff(int id, char sname[], char email[], int number, char uname[],
char pword[]) {

   StaffiD = id;
   strcpy(StaffName, sname);
   strcpy(StaffEmail, email);
   StaffNumber = number;
   strcpy(StaffUsername, uname);
   strcpy(Password, pword);

}

void Staff::login(char name[], char password[]) {

}
void Staff::addSellingApartment(Apartment* Ap[SIZE]) {

}
```

```
Staff::~Staff() {

   //Destructor
}
```

- **Administrator.cpp**

```
#include "Admin.h"
#include "Apartment.h"
#include <cstring>
#define SIZE

using namespace std;

Admin::Admin() {

      AdminiD = 0;
      strcpy(AdminName, "");
      strcpy(AdminEmail, "");
      AdminNumber = 0;
      strcpy(AdminUsername, "");
      strcpy(Password, "");

}

Admin::Admin(int id, char aname[], char email[], int number, char uname[], char
pword[]) {

      AdminiD = id;
      strcpy(AdminName, aname);
      strcpy(AdminEmail, email);
      AdminNumber = number;
      strcpy(AdminUsername, uname);
```

```
	strcpy(Password, pword);


}


void Admin::login(char name[], char password[]) {


}
void Admin::manage(Apartment* Ap[SIZE]) {


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

- **Apartment.cpp**

```
#include "Apartment.h"
#include "Selling.h"
#include <cstring>
#include <iostream>
#define SIZE 2


using namespace std;

Apartment::Apartment() {

	ApartmentID = 0;
	strcpy_s(ApartmentLocation, "");
	strcpy_s(ApartmentFacilities, "");
	strcpy_s(ApartmentStatus, "");
	sell = new Selling();
	reserve[0] = new Reserving();
	reserve[1] = new Reserving();
	reserve[2] = new Reserving();
}

Apartment::Apartment(int id, const char location[], const char facilities[],
const char status[], const char preserveID1[], const char preserveDate1[],
const char preserveDescription1[], double preservePrice1, const char
preserveID2[], const char preserveDate2[], const char
preserveDescription2[], double preservePrice2, Admin* gadmin, Staff* gstaff)
{
```

```
   ApartmentID = id;
   strcpy_s(ApartmentLocation, "location");
   strcpy_s(ApartmentFacilities, "facilities");
   strcpy_s(ApartmentStatus, "status");
   sell = new Selling();
   reserve[0] = new Reserving();
   reserve[1] = new Reserving();

   admin = gadmin;
   staff = gstaff;

}

void Apartment::Apartmentdetails()
{

}
void Apartment::displayApartmentdetails()
{
}
void Apartment::updateApartmentDetails()
{
}
void Apartment::checkAvalibility() {

}
void Apartment::deleteApartmentDetails() {

}
void Apartment::calculateApartmentprice(){

}
Apartment::~Apartment() {

   //Destructor
}
```

- **Selling.cpp**

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

using namespace std;

Selling::Selling() {

   SellID = 0;
   strcpy(SellDate, "");
   strcpy(SellDescription, "");
```

```
   Sellprice = 0;

}

Selling::Selling(int id, char date[], char description[], double price) {

   SellID = id;
   strcpy(SellDate, date);
   strcpy(SellDescription, description);
   Sellprice = price;

}

Selling::addpayment() {

   pay[0] = new Payment(pid, ptype, pamount);


}

void Selling::DisplaySellPrice() {

   details[0]->displayPaymentDetails();

}
```

➢ Reserving.cpp

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

   Reserving::Reserving()
   {
       strcpy_s(ReserveID, "");
       strcpy_s(ReserveDate, "");
       strcpy_s(ReserveDescription, "");
       ReservePrice = 0;
       payment[0] = new Payment();
       payment[1] = new Payment();
   }
   Reserving::Reserving(const char preserveID[], const char preserveDate[],
   const char preserveDescription[], double preservePrice, int PID, const
   char PpayType[], double PpayAmount)
   {
       strcpy_s(ReserveID, preserveID);
       strcpy_s(ReserveDate, preserveDate);
       strcpy_s(ReserveDescription, preserveDescription);
       ReservePrice = 0;
       payment[0] = new Payment(PID, PpayType ,PpayAmount);
```

```
    payment[1] = new Payment(PID, PpayType, PpayAmount);
}
void Reserving::calculateReservePrice(int id, const char pType[], double
pAmt)
{
    if (count < SIZE)
    {
        payment[count] = new Payment(id, pType, pAmt);
        count++;
    }
}
void Reserving::displayReservePrice()
{
}
void Reserving::addReserving()
{
}
Reserving::~Reserving()
{
    //Destructor
    for (int i = 0; i < SIZE; i++)
    {
        delete payment[i];
    }
}
```

- **Payment.cpp**

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

Payment::Payment()
{
      PayID = 0;
      strcpy_s(PayType, "");
      PayAmount = 0;
}
Payment::Payment(int PID, const char PpayType[], double PpayAmount)
{
      PayID = PID;
      strcpy_s(PayType, PpayType);
      PayAmount = PpayAmount;
}
void Payment::checkPayment()
{
}
void Payment::confirmPayment()
{
}
```

```
void Payment::displayPaymentDetails()
{
}
Payment::~Payment()
{
}
```

- **Report.cpp**

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

Report::Report()
{
    for (int i = 0; i < SIZE1; i++)
    {
        reserve[i] = 0;
    }
    for (int j = 0; j < SIZE2; j++)
    {
        sell[j] = 0;
    }
    for (int k = 0; k < SIZE3; k++)
    {
        pay[k] = 0;
    }
}
Report::Report(Reserving* preserve[], Selling* psell[], Payment* ppay[])
{
    for (int i = 0; i < SIZE1; i++)
    {
        reserve[i] = preserve[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::ReservingDetailsReport()
{
}
void Report::sellingDetailsReport()
{
}
void Report::paymentDetailsReport()
{
}
Report::~Report()
```

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

8) Main Program

```
#include <iostream>
#include <cstring>
#include "Reserving.h"
#include "Selling.h"
#include "Staff.h"
#include "Buyer.h"
#include "Admin.h"
#include "Apartment.h"
#include "GuestUser.h"
#include "Payment.h"
#include "RegisteredCustomer.h"
#include "Report.h"

using namespace std;

int main()
{

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

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

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

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

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

	Reserving* reserving = new Reserving(); // Object - Booking class

	Admin* admin = new Admin(); // Object - Staff class

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

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

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

	apt->updateApartmentDetails();
	apt->checkAvalibility();

	selling->addSelling();
	selling->DisplaySellPrice();

	reserving->addReserving();
	reserving->displayReservePrice();
```

```
        report->ReservingDetailsReport();
        report->sellingDetailsReport();
        report->paymentDetailsReport();

        //----Delete Dynamic objects----
        delete rg;
        delete staff;
        delete buyer;
        delete apt;
        delete selling;
        delete reserving;
        delete report;

        return 0;

}
```

9) Contribution

1. **Student Name: D.N.M. Dissanayake**
   Student ID: IT21185748

   - Identify noun and verbs in system requirements document.
   - Identify Guest User and Registered Customer Classes in noun verb analysis document.
   - Create CRC Cards for Guest User and Registered Customer classes.
   - Draw class diagram for Guest User and Registered Customer classes.
   - Write and develop cording for the Guest User and Registered Customer classes.

2. **Student Name: E.M.D.K.L. Ekanayake**
   Student ID: IT21186042

   - Identify noun and verbs in system requirements document.
   - Identify Staff and Selling Classes in noun verb analysis document.
   - Create CRC Cards for Staff and Selling classes.
   - Draw class diagram for Staff and Selling classes.
   - Write and develop cording for the Staff and Selling classes.

3. **Student Name: G.M.J.K. Gunathilake**
   Student ID: IT21185366

   - Identify noun and verbs in system requirements document.
   - Identify Admin and Payment in noun verb analysis document.
   - Create CRC Cards for Admin and Payment classes.
   - Draw class diagram for Admin and Payment classes.
   - Write and develop cording for the Admin and Payment classes.

4. **Student Name: W.B.M.S. Sriyarathne**
   Student ID: IT21185366

   - Identify noun and verbs in system requirements document.
   - Identify Apartment and Report Classes in noun verb analysis document.
   - Create CRC Cards for Apartment and Report classes.
   - Draw class diagram for Apartment and Report classes.
   - Write and develop cording for the Apartment and Report classes.

5. **Student Name: E.M.T.S. Edirisingha**
   Student ID: IT21185434

   - Identify noun and verbs in system requirements document.
   - Identify Buyer and Reserving Classes in noun verb analysis document.
   - Create CRC Cards Buyer and Reserving classes.
   - Draw class diagram for Buyer and Reserving classes.
   - Write and develop cording for the Buyer and Reserving classes.