Topic : **Online Land Sale System**

Group no : **MLB_01.01_08**

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**
IT21034718	Aththanayaka P.S.K	0771008349
IT21053900	Silva W.I.S	0785575691
IT21075544	Heshan J.A.C.I	0779768597
IT21069840	Devashika J.A.C.I	0702620205

Contents

System Requirements

- The system should be functioning 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 number
- Registered customers are of two types the are sellers & buyers where they can log in to the system by entering username and password
- They can buy and sell lands using the system.
- Land Sellers must be able to add land details such as location, price, and size to the system.
- Details must be confirmed by the administrator.
- The staff can delete or update the status of land details
- The system should generate an id for the lands after confirmation
- Merchants must pay a small commission per sale to the system before being placed on the system.
- After a sale is posted, the sale date and sale ID are generated for the sale.
- Buyers should be able to filter land by price, location, and rating.
- Buyers can choose land.
- After selection, a selection date and a selection id are generated.
- Both registered customers must make payment.
- Registered customers must enter their payment details, such as payment type, as a bank credit.
- After payment, "Pay ID" is generated for "sell ID" of sellers and "Land ID" of buyers.
- Once the payment is confirmed by a bank or other trusted sources, a sales details report for sellers and land details for buyers, as well as land details and payment details, are sent via email.

Noun & Verb Analysis

(NOUNS)

1. The system should be functioning every day.
2. Unregistered customers can overview the system to use the system. They must register with the system by providing details such as name, address, NIC, email and contact number
3. Registered customers are of two types the are sellers & buyers where they can log in to the system by entering username and password
4. They can buy and sell lands using the system.
5. Land Sellers must be able to add land details such as location, price, and size to the system.
6. Details must be confirmed by the staff.
7. The staff can delete or update the status of land details.
8. The system should generate an id for the lands after confirmation.
9. Sellers must pay a small commission per sale to the system before being placed on the system.
10. After a sale is posted, the sale date and sale ID are generated for the sale.
11. Buyers should be able to filter land by price, location, and rating.
12. Buyers can select land.
13. After selection, a selection date and a selection Id are Generated
14. Registered customers must make payment.
15. Registered customers must enter their payment details, such as payment type, as a bank credit.
16. After payment, "Payment ID" is generated for "sell ID" of sellers and "Land ID" of buyers.
17. Once the payment is confirmed by a bank or other sources, a sales details report for sellers and land details for buyers, as well as land details and payment details, are sent via email.

Identified Classes

- Unregistered customers
- Registered customers
- Sellers
- Buyers
- Land
- Staff
- Sales Details
- Payment Details

Reasons for rejecting other nouns

- **Redundant**: Sellers, Staff, Buyers
- **An Event or an operation:**
- **Outside scope of system**: System, Bank, sources, email
- **Meta-language:** They
- **An attribute**: Details (name, address, NIC, Email, contact number), Username, password

  Land Details (Location, price, size), status, land ID, commission, sale date, seller id, payment type, bank credit, payment ID

Noun & Verb Analysis

(VERBS)

1. The system should be functioning every day.
2. Unregistered customers can overview the system to use the system. They must register with the system by providing details such as name, address, NIC, email contact number
3. Registered customers are of two types the are sellers & buyers where they can log in to the system by entering username and password
4. They can buy and sell lands using the system.
5. Land Sellers must be able to add land details such as location, price, and size to the system.
6. Details must be confirmed by the staff.
7. The staff can delete or update the status of land details.
8. The system should generate an id for the lands after confirmation
9. Sellers must pay a small commission per sale to the system before being placed on the system.
10. After a sale is posted, the sale date and sale ID are generated for the sale.
11. Buyers should be able to filter land by price, location, and rating.
12. Buyers can select land.
13. After selection, a selection date and a selection id are generated.
14. Registered customers must make payment.
15. Registered customers must enter their payment details, such as payment type, as a bank credit.
16. After payment, "Payment ID" is generated for "sell ID" of sellers and "Land ID" of buyers.
17. Once the payment is confirmed by a bank or other sources, a sales details report for sellers and land details for buyers, as well as land details and payment details, are sent via email.

Methods

1. Unregistered Customer	Overview the system. Register to the system by providing details.
2. Registered Customer	Log into system by entering login details View the system.
3. Seller	Add land details. Pay commission. Sell lands.
4. Buyer	Selecting lands. Buy Lands Filter land details. Do payment for lands.
5. Lands	Generate land ID. Add land details Delete and update land details
6. Staff	Confirmed customer details. Delete and update land details.
7. Selling	Generate sell ID Calculate sell price
8. Payment	Generate payment ID Confirmed payment details. Make payment

CRC Cards

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

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

Seller	
Responsibility	Collaborators
Log in to the system	Registered Customer
Pay commission.	
Sell lands.	lands.

Buyer	
Responsibility	Collaborators
Log in to the system	Registered Customer
Buy Lands	lands.
Selecting lands.	lands.

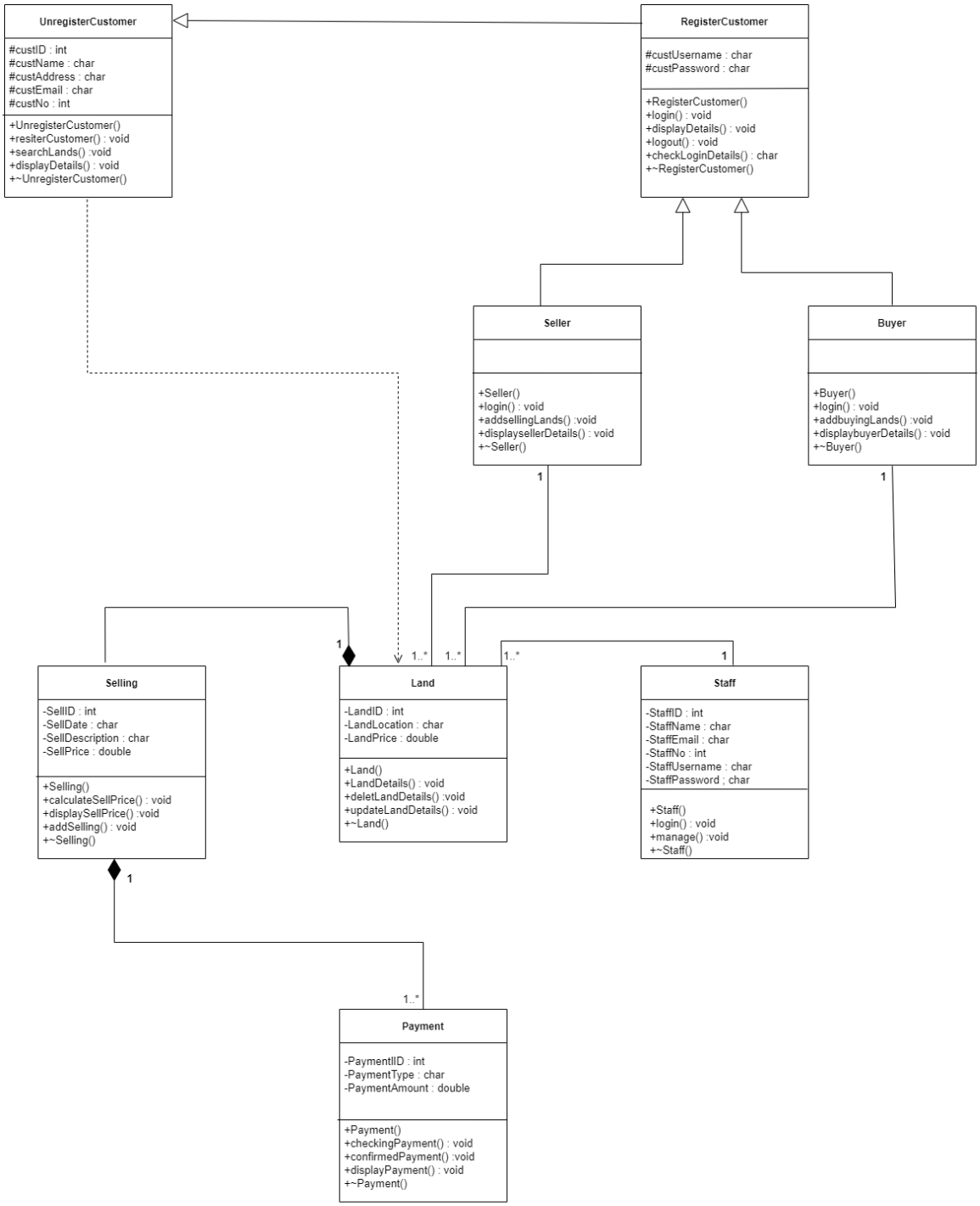
Lands	
Responsibility	**Collaborators**
Add land Details	Seller
Delete land Details	Staff
Update land Details	Seller, Staff

Staff	
Responsibility	**Collaborators**
Confirm lands details	Land
Delete land Details	Staff
Update Land Details	Seller, Staff

Selling	
Responsibility	Collaborators
Land selling	
Update the system	Land
Calculate the fee	

Payment	
Responsibility	Collaborators
Make a Payment	
Generate PayID	Selling
Confirm Payment Details	Seller, Buyer

Class Diagram (UML Notation)



Class Header Files and Class Cpp Files

GuestUser

```
#include<iostream.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 searchLands(Lands * pLan);
void registerUser();
virtual void displayDetails();
~GuestUser();
};

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::searchLands(Lands * pLan)
{
}
void GuestUser::registerUser()
{
}
void GuestUser::displayDetails()
{
}
GuestUser::~GuestUser()
{
//Destructor
}
```

Registered Customer

```
#include<iostream.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();
};

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

```
// IT21053900  Silva W.I.S

#include<iostream.h>
class Seller :public RegisteredCustomer
{
private:
 int noOfLands;
Lands* sellLands[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 pnoOfLands);
 void addSellingLands(Lands* psellLands);
 void login();
 void displaySellerDetails();
 ~Seller();
};


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


Seller::Seller(const char usName[], const char usPwd[], int id,
const char name[], const char address[], const char email[], const
char tellno[], int pnoOfLands) :RegisteredCustomer(usName,
```

```
usPwd, id, name, address, email, telno)
{
noOfLands = pnoOfLands;
}


void Seller::addSellingLands(Lands* psellLands)
{
if (noOfLands < SIZE)
{
sellLands[noOfLands] = psellLands;
noOfLands++;
}
}


void Seller::login()
{
}


void Seller::displaySellerDetails()
{
}



Seller::~Seller()
{

}
```

// IT21053900 Silva W.I.S

Buyer

// IT21053900 Silva W.I.S

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

#define SIZE 5




class Buyer : public RegisteredCustomer
{
private:
 int noOflands;
 land* buyLands[SIZE];

public:

 Buyer();
 Buyer(cons char usName[], cons char usPwd[], int id, cons
char name[], cons char address[], cons char email[], cons char
telno[],int pnoOfLands);
 void addBuyingLands(Lands* pbuyLands);
 void login();
 void displayBuyerDetails();
 ~Buyer();
};
```

```
Buyer::Buyer()
{
noOfLands = 0;
}
Buyer::Buyer(cons char usName[], cons char usPwd[], int id, const
char name[], cons char address[], cons char email[], cons char
telno[], int pnoOfLands:RegisteredCustomer(usName,usPwd, id,
name, address, email, telno)
{
noOfLands = pnoOfLands;
}
void Buyer::addBuyingLands(Lands* pbuyLands)
{
if (noOfLands < SIZE)
{
buyLands[noOfLands] = pbuyLands;
noOfLands++;
}
}
void Buyer::login()
{
}
void Buyer::displayBuyerDetails()
{
}
Buyer::~Buyer()
{

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

// IT21053900 Silva W.I.S

Land

```
#include<iostream>
using namespace std;
class Land
{
private:
int landID;
char landLoctaion[50];
double landPrice;
int count = 0;
Selling* sell[SIZE2];
Seller* seller;
Buyer* buyer;
Staff* staff;
public:
land();
land(int sell1, int sell2,Seller*
pseller, Buyer* pbuyer,Staff* pstaff);
void landDetails(int lanID, const char lanLocation,
double lanPrice, const
char , Seller* pseller , Buyer* pbuyer , Staff* pstaff);
void deleteLandDetails();
void updateLandDetails();
void calculateLandPrice();
void displayLanDetails();
void checkAvailability();
~Land();
```

```
};



Land::Land()
{
}
Land::Land(int sell1, int sell2, Seller* pseller, Buyer* pbuyer, Staff* pstaff)
{
sell[0] = new Selling(sell1);
sell[1] = new Selling(sell2);
seller = pseller;
buyer = pbuyer;
staff = pstaff;
}
void Land::landDetails(int lanID, const char lanLocation,
double lanPrice,const
char Seller* pseller, Buyer* pbuyer, Staff* pstaff)
{
}
void Land::deleteLandDetails()
{
}
void Land::updateLandDetails()
{
}
void Land::calculateLandPrice()
{
}
void Land::displayLanDetails()
{
}
Land::~Land()
{
//Destructor
```

```
for (int i = 0; i < SIZE1; i++)
{
delete sell[i];
}

}
```

Staff

```
#include<iostream>
using namespace std;
#define SIZE 5
class Staff
{
private:
int staffID;
char staffName[20];
char staffEmail[20];
char staffNumber[10];
char staffUsername[20];
char staffPassword[20];
Lands* land[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( Land* pland);
~Staff();
```

```
};



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(Land* pland)
{
}
Staff::~Staff()
{
//Destructor
for (int i = 0; i < SIZE; i++)
{
```

```
delete land[i];
}
}
```

Selling

```
#include<iostream>
using namespace std;
#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();
};

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];
}
}
```

Payment

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

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();
};
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
}
```

Main program

```
#include "Selling"
#include "Seller"
#include "Buyer"
#include "Staff"
#include "Land"
#include "GuestUser"
#include "Payment"
#include "RegisteredCustomer"
#include <iostream>
using namespace std;
int main()
{
 //creation of object
 GuestUser* rg = new RegisteredCustomer(); // Object -
RegisteredCustomer class

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

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

 Land* land = new Land(); // Object - Land class

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

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




//methods

 rg->login();
 rg->displayDetails();



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


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



land->updateLandDetails();
land->checkAvailability();


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


payment->addSelling();
payment ->displaySelPrice();



delete rg;
delete seller;
delete buyer;
delete land;
 delete selling;
 delete payment;

 return 0; }
```