Topic	: Online Land Sale System
Group no	: MLB_09.01_06
Campus	: Malabe
Submission Date	: 18/05/2022

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**
IT21175084	V.P.E.P.V. Pathirana	0788750055
IT21172700	M.J.E.M. Arachchi	0713382440
IT21173004	A.W.S Sandeepani	0763991246
IT21174162	E.M.D.T. Ekanayake	0771166407
IT21173240	H.M.M.D. Herath	0714946604

Contents

System Requirements

- The system should be operational 24 x 7.
- Guest users can view the system but must register to utilize it by entering information such as their name, address, NIC, email and phone number.
- Registered customers are divided into two categories. They are sellers and buyers, and they can log into the system by entering the necessary username and password.
- They can "Sell" or "Buy" lands using the system.
- Sellers should be able to add land details such as Location, Price, facilities, and utility price to the system.
- Details should be confirmed by the land manager.
- Land manager can delete or update the status of the land details.
- After validating the land, the system should generate a unique ID for it.
- After placing the sale, date of sale and sell ID is generated to the selling.
- Land should be able to be filtered by price, location, and ratings for buyers.
- Buyers can place a booking by choosing a land.
- After booking, date of booking and booking ID is generated.
- Both the registered customers must do a payment.
- Registered customers must input payment details such as payment method and card details.
- Registered customers must enter their customer details, bank details.
- After the payment 'Payment ID' is generated to the 'Seller ID' of sellers and 'book ID' of buyers.
- After the payment is validated by bank or other trusted resources a report of the selling details for sellers and booking details for buyers and land details and payment details is emailed.

Noun & Verb Analysis

(Nouns)

- The system should be operational 24 x 7.
- Guest users can view the system but must register to utilize it by entering details such as their name, address, NIC, email and phone number.
- Registered customers are divided into two categories. They are sellers and buyers, and they can log into the system by entering the necessary username and password.
- They can "Sell" or "Buy" lands using the system.
- Sellers should be able to add land details such as Location, Price, facilities, and utility price to the system.
- Details should be confirmed by the land manager.
- Land manager can delete or update the status of the land details.
- After validating the land, the system should generate a unique ID for it.
- After placing the sale, date of sale and sell ID is generated to the selling.
- Land should be able to be filtered by price, location, and ratings for buyers.
- Buyers can place a booking by choosing a land.
- After booking, date of booking and booking ID is generated.
- Both the registered customers must do a payment.
- Registered customers must input payment details such as payment method and card details.
- Registered customers must enter their customer details, bank details.
- After the payment 'Payment ID' is generated to the 'Seller ID' of sellers and 'book ID' of buyers.
- After the payment is validated by bank or other trusted resources a report of the selling details for sellers and booking details for buyers and land details and payment details are emailed.

Identified Classes

- Guest User
- Registered customer
- Seller
- Buyer
- Land
- Land manager
- Selling
- Booking
- Payment
- Report

Reasons for rejecting other nouns

- **Redundant:** Sellers, Land manager, Buyers
- **An event or an operation:**
- **Outside scope of system:** System, Bank, trusted resources
- **Meta – language:** they
- **An attribute:** Details (Name, Address, Email, phone number), Username, Password, Land Details (Location, Price, facilities, utility price), Status, unique ID (Land ID), Date of sale, Seller ID, Date of booking, Booking ID, Payment method, card details, Customer details, Bank details, Payment ID.

Noun & Verb Analysis

(Verbs)

- The system should be operational 24 x 7.
- Guest users can view the system but must register to utilize it by entering information such as their name, address, NIC, email and phone number.
- Registered customers are divided into two categories. They are sellers and buyers, and they can log into the system by entering the necessary username and password.
- They can “Sell” or “Buy” lands using the system.
- Sellers should be able to add land details such as Location, Price, facilities, and utility price to the system.
- Details should be confirmed by the land manager.
- Land manager can delete or update the status of the land details.
- After validating the land, the system should generate a unique ID for it.
- After placing the sale, date of sale and sell ID is generated to the selling.
- Land should be able to be filtered by price, location, and ratings for buyers.
- Buyers can place a booking by choosing a land.
- After booking, date of booking and booking ID is generated.
- Both the registered customers must do a payment.
- Registered customers must input payment details such as payment method and card details.
- Registered customers must enter their customer details, bank details.
- After the payment ‘Payment ID’ is generated to the ‘Seller ID’ of sellers and ‘book ID’ of buyers.
- After the payment is validated by bank or other trusted resources a report of the selling details for sellers and booking details for buyers and land 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 apartments
	-Place a selling
• Buyer	-Buy lands
	-Search lands by filtering requirements
	-Place a booking
	-Selecting lands
	-Do payment for lands
• Land	-Generate Land ID
	-Add land details
	-Delete and update land details
• Land Manager	-Log into the system
	-Confirm apartment details
	-Manage apartment details
• Selling	-Generate sell ID
	-Update the system
	-Calculate selling price
• Booking	-Generate Booking ID
	-Check available lands
	-Calculate Booking price
• Payment	-Generate payment ID
	-Check payment details
	-Confirm payments
• Report	-Generate Booking details
	-Generate selling details
	-Generate Payment details

CRC Cards

Guest User	
Responsibility	**Collaborators**
Register to the system	
Allow to view the lands	Land

Registered Customer	
Responsibility	**Collaborators**
Can find a land	Land
Upload and update customer details	

Seller	
Responsibility	**Collaborators**
Log in to the system	Registered Customer
Sell Lands	Lands

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

Land	
Responsibility	**Collaborators**
Add land details	Seller
Delete land details	Land Manager
Update land details	Seller, Land Manager

Selling	
Responsibility	**Collaborators**
Land selling	
System upgrades	Land
Determine the selling fee	

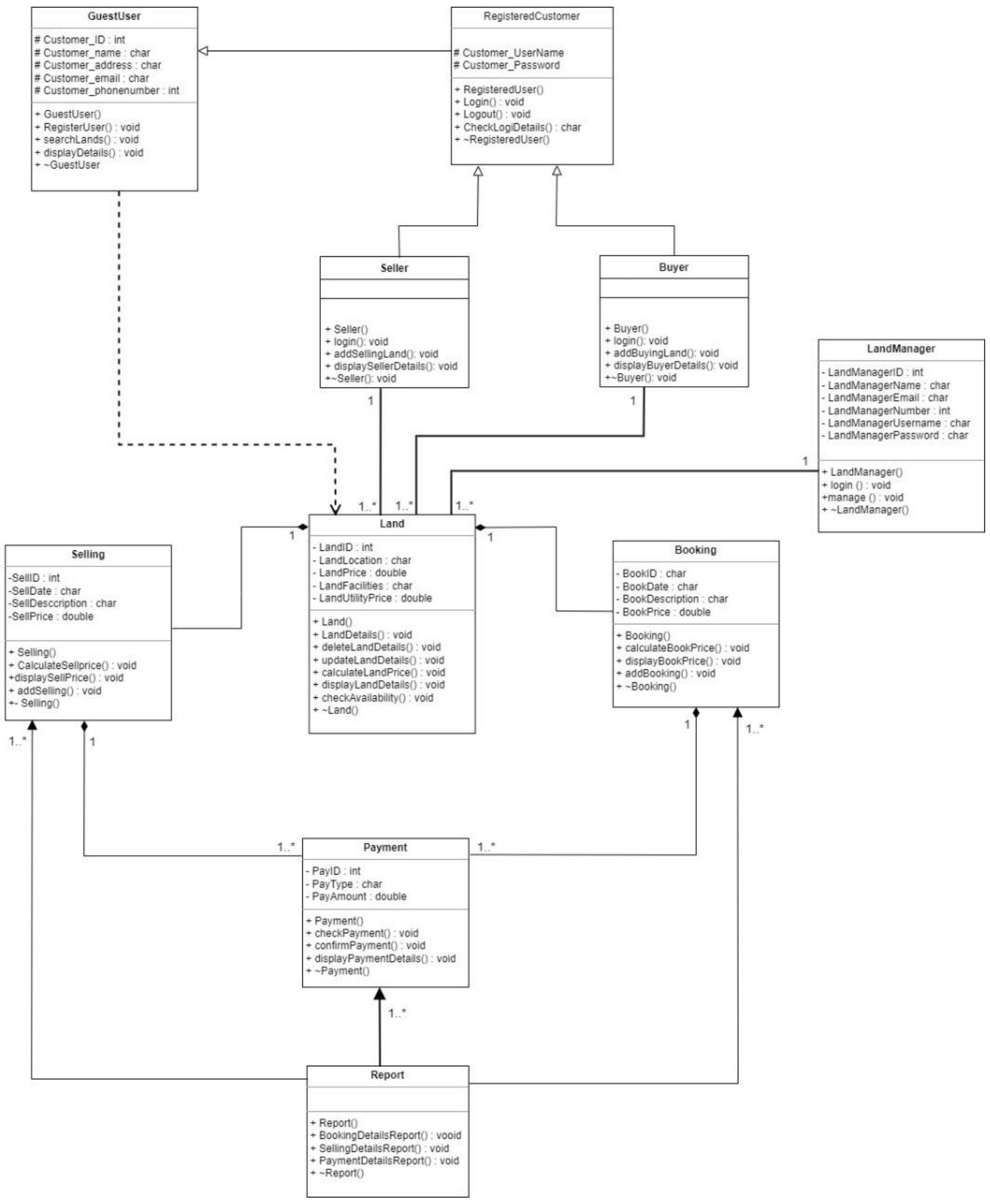
Land Manager	
Responsibility	**Collaborators**
Log in to the system	
Confirm land details	Land
Manage land details (Delete and update land details)	Land

Booking	
Responsibility	**Collaborators**
Place a booking	
Check availability of lands	Land
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 "Land.h"
class GuestUser
{
  protected:
    int Customer_ID;
    char Customer_name [20];
    char Customer_address [30];
    char Customer_email [30];
    char Customer_phonenumber [10];

  public:
    GuestUser ();
    GuestUser (int pcustid, const char pcustName [], const char
pcustAddress [], const char pcustEmail [],
    const char custPHno []);
    void searchLands (Land * pLnd);
    void RegisterUser ();
    virtual void displayDetails ();
    ~GuestUser ();
};
```

RegisteredCustomer.h

```
#include"GuestUser.h"
class RegisteredCustomer: public GuestUser
{
protected:
      char customer_Username [10];
      char customer_Password [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 "Land.h"
#define SIZE 5
class Seller :public RegisteredCustomer
{
private:
 int noOfLands;
 Land* sellLnd [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 addSellingLand (Land* psellLnd);
 void login ();
 void displaySellerDetails ();
 ~Seller ();
};
```

Buyer.h

```
#include "RegisteredCustomer.h"
#include "Land.h"
#define SIZE 5
class Buyer : public RegisteredCustomer
{
private:
 int noOfLands;
 Land* buyLnd [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
pnoOfLands);
 void addBuyingLand (Land* pbuyLnd);
 void login ();
 void displayBuyerDetails ();
 ~Buyer ();
};
```

Land.h
```
#include "LandManager.h"

#define SIZE1 2
#define SIZE2 2

class Land {
  private:
    int Land_ID;
    char Land_Location [50];
    double Land_Price;
    char Land_Facilities [50];
    double Land_UtilityPrice;
    int count = 0;

    Booking* book [SIZE1];
    Selling* sell [SIZE2];
    Seller* seller;
    Buyer* buyer;
    LandManager* landmanager;

  public:
    Land ();
    Land (int sell1, int sell2, int book1, int book2, Seller* pseller, 
Buyer* pbuyer, LandManager* plandmanager);
    void LandDetails (int LdID, const char LdLocation, double LdPrice, 
const char lndFacility, double LdUtiprice, const   char LdStatus, Seller * 
pseller, Buyer* pbuyer, LandManager* landmanager);
    void deleteLandDetails ();
    void updateLandDetails ();
    void calculateLandPrice ();
    void displayLdDetails ();
    void checkAvailability ();
    ~Land ();
};
```

LandManager.h
```
#include "Land.h"

#define SIZE 5

class LandManager
{
private:
      int LandManagerID.
      char LandManagerName [20];
      char LandManagerEmail [20];
      char LandManagerNumber [10];
      char LandManagerUsername [20];
      char LandManagerPassword [20];
```

```
	Land* lnd[SIZE];

public:
	LandManager ();
	LandManager (int pLandManagerID, const char pLandManagerName [],
const char pLandManagerEmail [], const char pLandManagerNumber [], const
char pLandManagerUsername [], const char pLandManagerPassword []);
	void login (const char LandMngUsername, const char LandMngPassword);
	void manage (Land* plnd);
	~LandManager ();

};
```

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*landbok[], Selling*landsell[], Payment*landpay []);
	void bookingDetailsReport ();
	void sellingDetailsReport ();
	void paymentDetailsReport ();
	~Report ();
};
```

Class Cpp Files

GuestUser.cpp

```
#include "GuestUser.h"

#include <cstirng>


GuestUser::GuestUser ()
{
  Customer_ID = 0;
  strcpy (Customer_name,"");
  strcpy (Customer_address,"");
  strcpy (Customer_email,"");
  strcpy (Customer_phonenumber,"0000000000");
}

GuestUser::GuestUser (int pcustid, const char pcustName [], const char
pcustAddress [], const char pcustEmail [], const char custPHno [])
{
  custID = pcustid;
  strcpy (customer_name, pcustName);
  strcpy (customer_address, pcustAddress);
  strcpy (customer_email, pcustEmail);
  strcpy (customer_phonenumber, custPHno);
}

void GuestUser::searchLand(Land * pld)
{

}

void GuestUser::RegisterUser ()
{
}

void GuestUser::displayDetails ()
{
}

GuestUser::~GuestUser ()   //Destructors
{

}
```

RegisteredCustomer.cpp

```
#include "RegisteredCustomer.h"
#include <cstring>
RegisteredCustomer::RegisteredCustomer ()
{
	strcpy (Username, "");
	strcpy (Password, "");
}
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()

{

noOfLands = 0;

}

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

{

noOfLands = pnoOfLands;
}
void Seller::addSellingLand (Land* psellLnd)
{
if (noOfLands < SIZE)
{
sellLnd[noOfLands] = psellLnd;
noOfLands++;
}
}
void Seller::login ()
{
}
void Seller::displaySellerDetails ()
{
}
Seller::~Seller () //Destructor
{
}
```

Buyer.cpp

```
#include "Buyer.h"
Buyer::Buyer ()
{
noOfLands = 0;
}
Buyer::Buyer (const char usName [], const char usPwd [], int id, const
char name [], const char address [], const char email [], const char telno
[], int pnoOfLands):RegisteredCustomer(usName,usPwd, id, name, address,
email, telno)
{
noOfLands = pnoOfLands;
}
void Buyer::addBuyingLand (Land* pbuyLnd)
{
if (noOfLands < SIZE)
{
buyLnd[noOfLands] = pbuyLnd;
noOfLands++;
}
}
```

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

Land.cpp

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

Land::Land ()
{

}

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

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

  seller = pseller;
  buyer = pbuyer;
  landmanager = plandmanager;
}


void Land::LandDetails (int LdID, const char LdLocation, double LdPrice,
const char LdFacility, double LdUtiPrice, Seller* pseller, Buyer* pbuyer,
LandManager* landmanager)
{

}

void Land::deleteLandDetails ()
{

}
```

```
void Land::updateLandDetails ()
{

}

void Land::calculateLandPrice ();
{

}

void Land::displayLdDetails ();
{

}

void Land::checkAvailability ();

Land::~Land ()    //destructor
{

  for (int i = 0; i < SIZE1; i++)
    {
      delete book[i];
    }
  for (int i = 0; i < SIZE2; i++)
    {
      delete sell [i];
    }
}
```

<u>LandManager.cpp</u>

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

LandManager::LandManager ()
{
    LandManagerID = 0;
    strcpy (LandManagerName "");
    strcpy (LandManagerEmail "");
    strcpy (LandManagerNumber "0000000000");
    strcpy (LandManagerUsername "");
    strcpy (LandManagerPassword "");
}

LandManager::LandManager (int pLandManagerID, const char pLandManagerName
[], const char pLandManagerEmail [], const char pLandManagerNumber [],
const char pLandManagerUsername [], const char pLandManagerPassword [])
{
    LandManagerID = pLandManagerID;
    strcpy (LandManagerName, pLandManagerName);
    strcpy (LandManagerEmail, pLandManagerEmail);
    strcpy (LandManagerNumber, pLandManagerNumber);
    strcpy (LandManagerUsername, pLandManagerUsername);
    strcpy (LandManagerPassword, pLandManagerPassword);
}

void LandManager::login (const char LandmngUsername, const char
LandmngPassword)
{

}
void LandManager::manage(Land* papt)
{

}
LandManager::~LandManager()  //Destructor
{
    for (int i = 0, i < SIZE; i++)
    {
        delete lnd[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 "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* landbbok[], Selling* landsell[],
Payment*landpay[])
{
	for (int i = 0; i < SIZE1; i++)
	{
		book[i] = landbbok[i];
	}
	for (int j = 0; j < SIZE2; j++)
	{
		sell[j] = landsell[j];
	}
	for (int k = 0; k < SIZE3; k++)
	{
		pay[k] = landpay[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 "LandManager.h"

#include "Land.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

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

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

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

 LandManager* landmanager = new LandManager (); // Object - LandManager 
class

 Report* report = new Report (); // Object - Report class
//----Method Calling------
 rg->login ();
 rg->displayDetails ();
 seller->login ();
 seller->displaySellerDetails ();
 buyer->login ();
 buyer->displayBuyerDetails ();
 lnd->updateLanDetails ();
 lnd->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 lnd;
 delete selling;
 delete booking;
 delete report;
 return 0;
}
```

Individual Contribution

	Student ID	Student Name	Individual Contribution
1	IT21175084	V.P.E.P.V Pathirana (Leader)	• LandManager.h • LandManager.cpp • Selling.h • Selling.cpp
2	IT21172700	M.J.E.M Arachchi	• Seller.h • Seller.cpp • Buyer.h • Buyer.cpp
3	IT21173004	A.W.S Sandeepani	• GuestUser.h • GuestUser.cpp • Land.h • Land.cpp
4	IT21174162	E.M.D.T Ekanayake	• Booking.h • Booking.cpp • Payment.h • Payment.cpp
5	IT21173240	H.M.M.D Herath	• RegisteredCustomer.h • RegisteredCustomer.cpp • Report.h • Report.cpp