Topic : Online Vaccination Portal

Group no : MLB_04.02_12

Campus : Malabe

Submission Date:

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**
IT21269370	Pothuwila P. Y. R	0703661870
IT21274220	Wickramasinghe B. G. A	0785497798
IT21274534	Moraes V. J	0750472211
IT21144080	R. Dhinojjan	0770025951
IT21277290	G. K. N. Dulangi	0770431140

Contents

System Requirements

- The System Must be function 24hours in all 365 days.
- Guest users have ability to overview the system and in order to use the system, guests should register in the system by entering details such as Full Name , NIC, DOB, Contact number, Email address, residence address.
- There are 2 types of registered users, Vaccination unit and registered customers .they can log into the system by providing the relevant details like, username and password.
- Registered customers can reserve vaccines.
- Details should be confirmed by the admin staff units.
- Staff units can delete or update the status of the vaccines

- All the details provided by users will be collected and stored in a database
- Customers can reserve a vaccine by search and selecting a suitable option.
- After reserving, vaccine ID and order ID is generated.
- Customers must do a payment to confirm the reservation.
- Customers must provide their payment details like payment type, card type.
- After the payment is confirmed by bank, a report of the reserving details for registered customer, vaccine details and payment details is emailed.

Noun & Verb Analysis

(NOUNS)

- The System Must be function 24hours in all 365 days.
- Guest users have ability to overview the system and in order to use the system, guests should register in the system by entering details such as Full Name , NIC, DOB, Contact number, Email address, residence address.
- There are 2 types of registered users, vaccination unit and registered customers .they can log into the system by providing the relevant details like, username and password.

- Registered customers can reserve vaccines.
- Details should be confirmed by the staff units.
- Staff units can delete or update the status of the vaccines details
- All the details provided by users will be collected and stored in a database
- Customers can reserve a vaccine or a session by search and selecting a suitable option.
- After reserving, vaccine ID and order ID is generated.
- Customers must do a payment to confirm the reservation.
- Customers must provide their payment details like payment type, card type.
- After the payment is confirmed by bank, a report of the reserving details for registered customer, vaccine details and payment details is emailed.

Identified Classes

- Guest User
- Registered user
- Registered customer
- Vaccination unit
- Staff Unit
- reserving
- vaccine type
- Payment
- Reservation report

Noun & Verb Analysis

(VERBS)

- The System Must be function 24hours in all 365 days.
- Guest users have ability to overview the system and in order to use the system, guests should register in the system by entering details such as Full Name , NIC, DOB, Contact number, Email address, residence address.
- There are 2 types of registered users, vaccination unit and registered customers, they can log into the system by providing the relevant details like, username and password.
- Registered customers can reserve vaccines.
- Details should be confirmed by the admin staff units.
- Staff units can delete or update the status of the vaccines details
- All the details provided by users will be collected and stored in a database
- Customers can reserve a vaccine by search and selecting a suitable option.
- After reserving, date of, vaccine ID and order ID is generated.
- Customers must do a payment to confirm the reservation.
- Customers must provide their payment details like payment type, card type.
- After the payment is confirmed by bank, a report of the reserving details for registered customer, vaccine details and payment details is emailed.

Methods

- Guest User - Register to the system by entering details and Review the system
- Registered User - Register to the system by entering details and

Review the system

- Registered Customer
  - Login to the system by entering details
  - Reserve and buy vaccine
  - Search vaccine
  -Place a reservation
  -Selecting vaccine
  -Do payment for vaccine
- Vaccination unit
  - Sell vaccine
  Place a selling
- Staff Unit
  - Log into the system,
  Confirm vaccine details
  Manage vaccine details
- Selling
  - Generate Sell ID
  Update the system
  Calculate price
- Reserving
  - Generate order ID
  Check availability of vaccine
  Calculate reserving price
- Vaccination
  - Generate Vaccine ID
  Add payment details
  Delete and update vaccine details
- Payment
  - Generate pay ID
  Check payment details
  Confirm payments

CRC Cards

Guest User	
Responsibility	**Collaborators**
Register to the system	
Ability to view the Vaccines	vaccines

Registered User	
Responsibility	**Collaborators**
Can view the vaccines	vaccines
Add and update customer details	

Vaccination Unit	
Responsibility	**Collaborators**
Log in to the system	Registered user
Sell vaccines	vaccines

Registered customer	
Responsibility	**Collaborators**
Log in to the system	Registered user
Buy vaccines	vaccines
Search vaccines	vaccines

Vaccines	
Responsibility	**Collaborators**
Add vaccine Details	Vaccination unit
Delete vaccine Details	Staff unit
Update vaccine Details	Vaccination unit, Staff unit

Selling	
Responsibility	**Collaborators**
Place selling	
Update the system	Vaccine

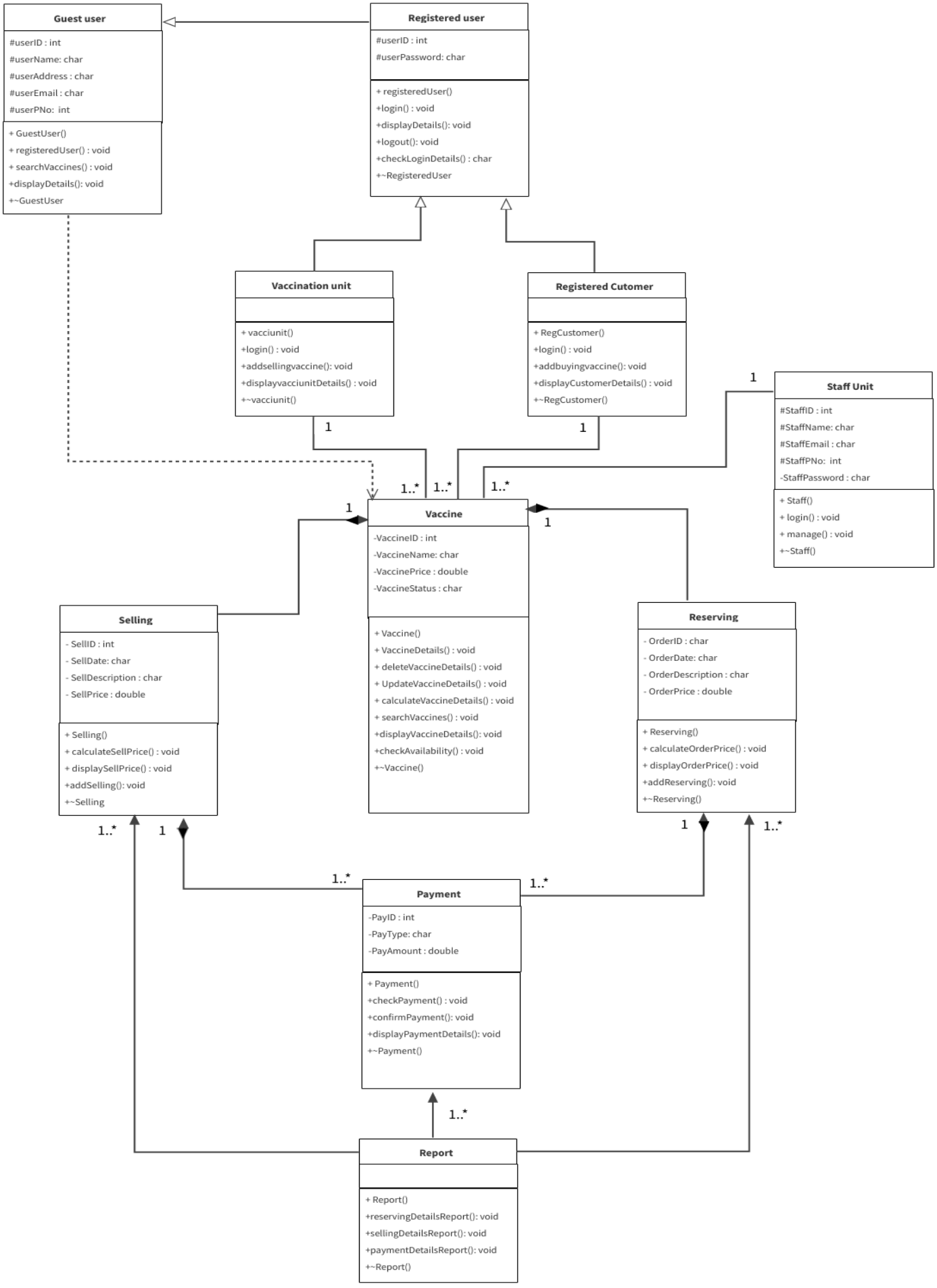
Staff unit	
Responsibility	**Collaborators**
Login to system	
Confirm vaccines details	vaccines

Reserving	
Responsibility	**Collaborators**
Place a reserving	
Check availability of vaccines	vaccine
Calculate the vaccine price	

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

Payment	
Responsibility	**Collaborators**
Make a new payment	
Generate Pay ID	Selling, Reserving
Check payment details	Vaccination unit, Registered customer
Confirm payment details	

Class Diagram (UML Notation)



Class Header Files

1. **GuestUser.h**

```
#include "vaccines.h"
class GuestUser
{
protected:
int userID;
char userName[20];
char userAddress[30];
char userEmail[30];
char userphoneNumber[10]

public:
GuestUser();
GuestUser(int puserid, const char puserName[], const char
puserAddress[], const char puserEmail[] ,const char userPHno[]);
void searchvaccines(vaccines* pApt);
void registerUser();
virtual void displayDetails();
~GuestUser();
};
```

2. **RegisteredUser.h**

```
#include"GuestUser.h"
class RegisteredUser : public GuestUser
{
 protected:
        int UserID;
        char UserPassword[15];
 public:
        RegisteredUser();
        RegisteredUser(int UID, const char UPassword[], const char UName[], const char
UserAddress[], const char TelNo[], const char UserEmail[]);
        void login();
        void displayDetails();
        void logout();
        char checkLoginDetails();
        ~RegisteredUser();
};
```

3. RegisteredCustomer.h

```
#include "RegisteredUser.h"
#include "Vaccine.h"
#define SIZE 5
class RegCustomer : public RegisteredUser
{
  private:
      int NoOfVaccines;
      Vaccine* buyVacc[SIZE];
  public:
      RegCustomer();
      RegCustomer(int Userid[], const char Userpw[], const char
      name[], const char Telno[], const char email[], const char
address[], int NoofVaccines);
      void addbuyingvaccine(Vaccine* Buyvaccine);
      void login();
      void displayCustomerDetails();
      ~RegCustomer();
};
```

4. VaccinationUnit.h

```
#include "Registereduser.h"
#include "vaccines.h"
#define SIZE 5
class VaccinationUnit :public Registereduser
{
private:
 int noOfvaccines;
 vaccines* VaccinationUnitApt[SIZE];
public:
 VaccinationUnit ();
VaccinationUnit( int userid[], const char userPassword[], int userid, const
char username[], const char useraddress[], const char useremail[], const char
usertelno [], int pnoOfvaccines);
 void addSellingvaccine(vaccines* pvacciunitApt);
 void login();
 void displayvacciunitDetails();
 ~vaccinationUnit();
      };
```

5. Vaccine.h

```
#include "Reserving.h"
#include "Selling.h"
#include "VaccinationUnit.h"
#include "RegisteredCustomer.h"
#include "StaffUnit.h"
#define SIZE1 2
#define SIZE2 2
class Vaccine
{
private:
int VaccineID;
char VaccineName[20];
double VaccinePrice;
char VaccineStatus[50];
int count = 0;

Reserving* Vaccine[SIZE1];
Selling* sell[SIZE2];
VaccinationUnit* VaccinationUnit;
RegisteredCustomer* RegisteredCustomer;
StaffUnit* staffUnit;

public:

  Vaccine();
  Vaccine(int sell1, int sell2, int Vaccine1, int Vaccine2,VaccinationUnit*
pVaccinationUnit, RegisteredCustomer* pRegisteredCustomer,StaffUnit* pStaffUnit);
  void VaccineDetails(int VaccineID , const char  VaccineName,
```

```
double VaccinePrice,  const char VaccineStatus, VaccinationUnit* pVaccinationUnit ,
RegisteredCustomer* pRegisteredCustomer , StaffUnit* pstaffUnit);
  void deleteVaccineDetails();
  void updateVaccineDetails();
  void calculateVaccinePrice();
  void searchVaccines();
  void displayVaccineDetails();
  void checkAvailability();
  ~Vaccine();
};
```

6. Selling.h

```
#include"Payment.h"
#define SIZE 2
class Selling {
private:
int SellID;
char SellDate[20];
char SellDescription[50];
double SellPrice;
int count = 0;
Payment* payment[SIZE];
public:
Selling();
Selling(int psellID, const char pselldate[], const char
pselldescription[], double psellprice, int pay1, int pay2);
void calculateSellPrice(int id, const char pType[], double
pAmt);
void displaySellPrice();
void addSelling();
```

```
~Selling();

};
```

7. **Reserving.h**

```
#include"Payment.h"
#define SIZE 2
class Reserving
{
  private:
        char OrderID[10];
        char OrderDate[15];
        char OrderDescription[100];
        double OrderPrice;
        int count = 0;
        Payment* payment[SIZE];
  public:
        Reserving();
        Reserving(const char vOrderID[], const char vOrderDate[],const char
        vOrderDescription[], double vOrderPrice, int payt);
        void calculateOrderPrice( char vType[], int vAmount);
        void displayOrderPrice();
        void addReserving();
        ~Reserving();
};
```

8. **Payment.h**

```
class Payment
{
private:
int payID;
char payType[20];
double payAmount;
public:
Payment();
Payment(int ppayID,const char ppayType[],double ppayAmount);
void checkPayment();
void confirmPayment();
void displayPaymentDetails();
~Payment();
```

```
};
```

9. **Report.h**

```
#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* vreserve[], Selling* vsell[], Payment* vpay[]);
      void reservingDetailsReport();
      void sellingDetailsReport();
      void paymentDetailsReport();
      ~Report();
};
```

Class cpp files

Guestuser.cpp

```
#include "GuestUser.h"
#include <cstring>
GuestUser::GuestUser()
{
userID = 0;
strcpy(userName, "");
strcpy(userAddress, "");
strcpy(userEmail, "");
strcpy(userphoneNumber, "0000000000");
}
GuestUser::GuestUser(int pusertid, const char puserName[], const char
puserAddress[], const char puserEmail[], const char userPHno[])
{
userID = puserid;
strcpy(userName, puserName);
strcpy(userAddress, puserAddress);
strcpy(userEmail, puserEmail);
strcpy(userphoneNumber, userPHno);
}
void GuestUser::searchvaccines(vaccines* pApt)
{
}
void GuestUser::registerUser()
{
}
void GuestUser::displayDetails()
{
}
GuestUser::~GuestUser()
{
//Destructor
}
```

RegisteredUser.cpp

```
#include "RegisteredUser.h"
#include <cstring>
RegisteredUser::RegisteredUser()
{
	strcpy( UserID, "");
	strcpy( UserPassword, "");
}
RegisteredUser::RegisteredUser(int UID, const char UPassword[], const char
UName[], const char UserAddress[], const char TelNo[], const char UserEmail[]) :
GuestUser(guestid, guestName, guestAddress, guestEmail, guestPNo)
{
  strcpy(UserID, guestUsername);
  strcpy(UserPassword, guestPassword);
}
void RegisteredUser::displayDetails()
{

}
void RegisteredUser::login()
{

}
void RegisteredUser::logout()
{

}
char RegisteredUser::checkLoginDetails()
{
  return 0;
}
RegisteredUser::~RegisteredUser()
{
	cout << "Destructor runs" << endl;
}
```

RegisteredCustomer.cpp

```
#include "RegisteredCustomer.h"

RegCustomer::RegCustomer()
{
	NoOfVaccines = 0;
}

RegCustomer::RegCustomer (int Userid[], const char Userpw[], const char
	name[], const char Telno[], const char email[], const char
address[], int NoofVaccines):RegisteredUser(int UID, const char UPassword[],
const char UName[], const char UserAddress[], const char TelNo[], const char UserEmail[])
{
	NoofVaccines = NoOfVaccines;
}

void RegCustomer::addbuyingvaccine(Vaccine* vbuyVacc)
{
  if (NoofVaccines < SIZE)
  {
	buyvacc[NoofVaccines] = vbuyvacc;
	NoofVaccines ++;
  }
}

void RegCustomer::login()
{

}

void RegCustomer::displayCustomerDetails()
{

}

RegCustomer::RegCustomer()
```

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

Vaccineunit.cpp

```
#include "VaccinationUnit.h"
VaccinationUnit::VaccinationUnit()
{
noOfvaccines = 0;
}
VaccinationUnit::VaccinationUnit(const char userName[], const char userPassword[], int
userid,
const char username[], const char useraddress[], const char useremail[], const
char usertelno[], int pnoOfvaccines) :Registereduser(userName,
userPassword, userid, username, useraddress, useremail, usertelno)
{
noOfvaccines = pnoOfvaccines;
}
void VaccinationUnit::addSellingvaccine(vaccines* pvacciunitApt)
{
if (noOfvaccines < SIZE)
{
vacciunitApt[noOfvaccines] = pvacciunitApt;
noOfvaccines++;
}
}
void VaccinationUnit::login()
{
}
void Seller::displayvacciunitDetails()
{
}
VaccinationUnit::~VaccinationUnit()
{
//Destructor
}
```

Vaccine.cpp

```
#include "Vaccine.h"

#define SIZE1 2

#define SIZE2 2

Vaccine::Vaccine()

{

}

Vaccine::Vaccine(int sell1, int sell2, int Vaccine1, int Vaccine2,

VaccinationUnit* pVaccinationUnit, RegisteredCustomer* pRegisteredCustomer, StaffUnit*
pstaffUnit)

{

sell[0] = new Selling(sell1);

sell[1] = new Selling(sell2);

Order[0] = new Reseving(Order1);

Order[1] = new Reseving(Order2);

VaccinationUnit = pVaccinationUnit;

RegisteredCustomer = pRegisteredCustomer;

staffUnit = pstaffUnit;

}

void Vaccine::VaccineDetails(int VaccineID, const char VaccineName,

double VaccinePrice,const char VaccineStatus, VaccinationUnit* pVaccinationUnit,
RegisteredCustomer* RegisteredCustomer, StaffUnit* pstaffUnit)

{

}

void Vaccine::deleteVaccineDetails()

{
```

```
}
void Vaccine::updateVaccineDetails()
{
}
```

Selling.cpp

```
#include "Selling.h"
#include<cstring>
Selling::Selling()
{
SellID = 0;
strcpy(SellDate, "");
strcpy(SellDescription, "");
SelPrice = 0;
}
Selling::Selling(int psellID, const char pselldate[], const char
pselldescription[], double psellprice, int pay1, int pay2)
{
SellPrice = psellprice;
strcpy(SellDate, pselldate);
strcpy(SellDescription, pselldescription);
SellID = psellID;
}
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];
}
}
```

Reserving.cpp

```
#include "Payment.h"
#include "Reserving.h"
#include<cstring>
Reserving::Reserving()
{
	strcpy(OrderID, "");
	strcpy(OrderDate, "");
	strcpy(OrderDescription, "");
	OrderPrice = 0;
}
Reserving::Reserving(const char vOrderID[], const char vOrderDate[],const char
	vOrderDescription[], double vOrderPrice, int payt);
{
	strcpy(OrderID, vOrderID);
	strcpy(OrderDate, vOrderDate);
	strcpy(OrderDescription, vOrderDescription);
	OrderPrice = 0;
}
void Reserving::calculateOrderPrice(int id, char payt[], double
vamount)
{
  if (count < SIZE)
  {
	payment[count] = new Payment(id, payt, vamount);
	count++;
  }
}
void Reserving::displayOrderPrice()
{

}
void Reserving::addReserving()
{

}
Reserving::~Reserving()
{
  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 ppayID, const char ppayType[], double ppayAmount)
{
payID = payID;
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++)
  {
      Reserve[i] = 0;
  }

  for (int j = 0; j < SIZE2; j++)
  {
      Sell[j] = 0;
  }

  for (int x = 0; x < SIZE3; x++)
  {
      Pay[x] = 0;
  }
}

Report::Report(Reserving* vReserve[], Selling* vSell[], Payment*
vPay[])
{
  for (int i = 0; i < SIZE1; i++)
  {
      reserve[i] = vReserve[i];
  }

  for (int j = 0; j < SIZE2; j++)
  {
      sell[j] = vSell[j];
  }

  for (int x = 0; x < SIZE3; x++)
  {
      pay[x] = vPay[x];
  }
}
```

```
void Report::reservingDetailsReport()
{

}
void Report::sellingDetailsReport()
{

}
void Report::paymentDetailsReport()
{

}

Report::~Report()
{
  for (int i = 0; i < SIZE1; i++)
  {
      delete reserve[i];
  }

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

  for (int x = 0; x < SIZE3; x++)
  {
      delete pay[x];
  }
}
```

Main Programme

```
#include "GuestUser.h"
#include "RegisteredUser.h"
#include "RegisteredCustomer.h"
#include "VaccinationUnit.h"
#include "Vaccine.h"
#include "Selling.cpp"
#include "Reserving.h"
#include "Payment.h"
#include "Report.h"
#include <iostream>
using namespace std;
int main()
{
 //---- Object creation ------

GuestUser* rg = new RegisteredCustomer(); // Object - RegisteredCustomer class
 RegisteredCustomer* vaccinationUnit = new vaccinationUnit(); // Object - vaccinationUnit class
 RegisteredUser*RegisteredCustomer  = new RegisteredCustomer(); // Object - RegisteredCustomer class
 Vaccine*Vaccine = new Vaccine(); // Object - Vaccine class
 Selling* selling = new Selling(); // Object - Selling class
 Reserving* Reserving = new Reserving(); // Object - Reserving class
 StaffUnit* staffUnit = new Staff(); // Object - StaffUnit class
 Report* report = new Report(); // Object - Report class
```

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

vaccinationUnit->login();
vaccinationUnit->displayvaccinationUnitDetails();
RegisteredCustomer->login();
RegisteredCustomer->displayRegisteredCustomerDetails();
Vaccine->updateVaccineDetails();
Vaccine->checkAvailability();
selling->addSelling();
selling->displaySellPrice();
Reserving->addReserving();
Reserving->displayOrderPrice();
report->ReservingDetailsReport();
report->sellingDetailsReport();
report->paymentDetailsReport();
//----Delete Dynamic objects------
delete rg;
delete vaccinationUnit;
delete RegisteredCustomer;
delete Vaccine;
delete selling;
delete Reserving;
delete report;
return 0;
}
```