Topic : Laboratory Information Management System

Group no : KGL_01

Campus : Malabe / Metro / Matara / Kandy / Kurunegala / Kandy / Jaffna

Submission Date: 20/5/2022

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
IT21257018	Rathnayaka K.P.S.K	071 6378320
IT21254420	Dewpura D.D.W.C	076 4190521
IT21195570	Herath H.M.K.C.B	072 8625364
IT21256950	Dayananda N.C.E	071 7155007
IT21238444	Wimalarathna D.M.A.T	074 2135589

System Requirements for Laboratory Management System

1) Guest should register to the system by creating an account.
2) After registering user can login to the system by entering correct user credentials.
3) A registered customer can view their profile and can edit and manage their user account.
4) A registered user can view lab tests and make a reservation for a test.
5) Registered user can place the reservation through online payment. Payment can be done by using credit card, debit card or master card.
6) System should check and confirm payment details.
7) System should send a payment confirmation message and lab test reservation details to the user.
8) User can download lab test report by entering username and password.
9) Staff can update / edit lab test details, add / remove lab test and upload lab reports to the user profile.
10) Staff can manage user details and generate reports such as financial reports.
11) Users can make feedbacks and enquiries if they want.

Noun Analysis

Nouns in RED color

1. Guest should register to the system by creating an account.
2. After registering user can login to the system by entering correct user credentials.
3. A registered customer can view their profile and can edit and manage their user account.
4. A registered user can view lab tests and make a reservation for a test.
5. Registered user can place the reservation through online payment. Payment can be done by using credit card, debit card or master card.
6. System should check and confirm payment details.
7. System should send a payment confirmation message and lab test reservation details to the user.
8. User can download lab test report by entering username and password.
9. Staff can update / edit lab test details, add / remove lab test and upload lab reports to the user profile.
10. Staff can manage user details and generate reports such as financial reports.
11. Users can make feedbacks and enquiries if they want.

Verbs Analysis

Verbs in BLUE color

1. Guest should register to the system by creating an account.
2. After registering user can login to the system by entering correct user credentials.
3. A registered customer can view their profile and can edit and manage their user account.
4. A registered user can view lab tests and make a reservation for a test.
5. Registered user can place the reservation through online payment. Payment can be done by using credit card, debit card or master card.
6. System should check and confirm payment details.
7. System should send a payment confirmation message and lab test reservation details to the user.
8. User can download lab test report by entering username and password.
9. Staff can update / edit lab test details, add / remove lab test and upload lab reports to the user profile.
10. Staff can manage user details and generate reports such as financial reports.
11. Users can make feedbacks and enquiries if they want.

Rules for rejecting nouns

Guest – Redundant (with user)

System – Out of the scope

Account – Out of the scope

User - Redundant

User credentials – An attribute

Registered customer - Redundant

Profile – Out of the scope

Registered user - Class

Lab test - Class

Reservation - Class

Payment - Class

Card - Class

Lab test reservation – An event or Operation

Lab test report - Class

Username – An attribute

Password – An attribute

Staff - Class

User profile - Redundant

Financial reports – Meta Language

Feedback - Class

Report - Class

CRC Cards

Registered User	
Responsibilities	Collaboration
View, edit and manage user profile	
View and select required lab test	Lab test
Make a reservation	Reservation
Make a payment	Payment
Download lab report	Lab report

Reservation	
Responsibilities	Collaboration
Confirm reservation	Registered user
Display reservation details	

Lab test	
Responsibilities	Collaboration
Add/remove lab tests	Staff
Update lab test details	Staff
Display lab test details	

Payment	
Responsibilities	Collaboration
Confirm payment details	Registered user
View payment details	
Store payment details	
Validate payments	Card

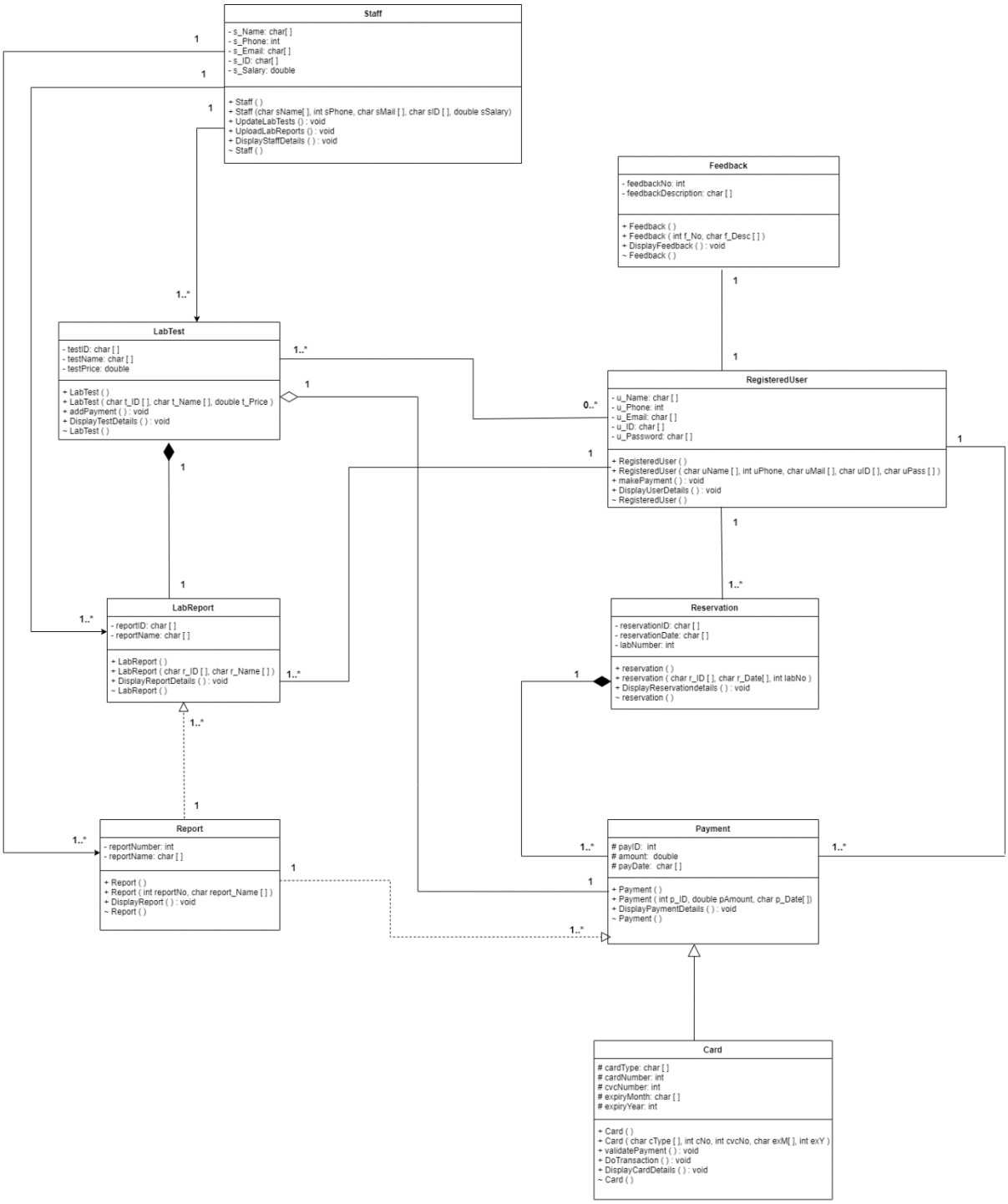
Lab report	
Responsibilities	Collaboration
Upload lab reports	Staff
Download lab reports	Registered user
Display lab reports	

Card	
Responsibilities	Collaboration
Do transactions	Staff
Display card details	

Staff	
Responsibilities	Collaboration
Manage lab report details	Lab report
Manage lab test details	Lab test
Manage user details	Registered user
Generate reports	Report

Feedback	
Responsibilities	Collaboration
Add feedbacks	Registered user
Store feedbacks	

Report	
Responsibilities	Collaboration
Store list of lab report details	Lab report
Store list of payment history	Payment


Staff
- s_Name: char[ ]
- s_Phone: int
- s_Email: char[ ]
- s_ID: char[ ]
- s_Salary: double
+ Staff ( )
+ Staff (char sName[ ], int sPhone, char sMail [ ], char sID [ ], double sSalary)
+ UpdateLabTests ( ) : void
+ UploadLabReports ( ) : void
+ DisplayStaffDetails ( ) : void
~ Staff ( )
Feedback
- feedbackNo: int
- feedbackDescription: char [ ]
+ Feedback ( )
+ Feedback (int f_No, char f_Desc [ ])
+ DisplayFeedback ( ) : void
~ Feedback ( )
LabTest
- testID: char [ ]
- testName: char [ ]
- testPrice: double
+ LabTest ( )
+ LabTest (char t_ID [ ], char t_Name [ ], double t_Price)
+ addPayment ( ) : void
+ DisplayTestDetails ( ) : void
~ LabTest ( )
RegisteredUser
- u_Name: char [ ]
- u_Phone: int
- u_Email: char [ ]
- u_ID: char [ ]
- u_Password: char [ ]
+ RegisteredUser ( )
+ RegisteredUser (char uName [ ], int uPhone, char uMail [ ], char uID [ ], char uPass [ ])
+ makePayment ( ) : void
+ DisplayUserDetails ( ) : void
~ RegisteredUser ( )
LabReport
- reportID: char [ ]
- reportName: char [ ]
+ LabReport ( )
+ LabReport (char r_ID [ ], char r_Name [ ])
+ DisplayReportDetails ( ) : void
~ LabReport ( )
Reservation
- reservationID: char [ ]
- reservationDate: char [ ]
- labNumber: int
+ reservation ( )
+ reservation (char r_ID [ ], char r_Date[ ], int labNo)
+ DisplayReservationdetails ( ) : void
~ reservation ( )
Report
- reportNumber: int
- reportName: char [ ]
+ Report ( )
+ Report (int reportNo, char report_Name [ ])
+ DisplayReport ( ) : void
~ Report ( )
Payment
payID: int
amount: double
payDate: char [ ]
+ Payment ( )
+ Payment (int p_ID, double pAmount, char p_Date[ ])
+ DisplayPaymentDetails ( ) : void
~ Payment ( )
Card
cardType: char [ ]
cardNumber: int
cvcNumber: int
expiryMonth: char [ ]
expiryYear: int
+ Card ( )
+ Card (char cType [ ], int cNo, int cvcNo, char exM[ ], int exY)
+ validatePayment ( ) : void
+ DoTransaction ( ) : void
+ DisplayCardDetails ( ) : void
~ Card ( )

Payment.h

```
class Payment
{
protected:
	int payID;
	double amount;
	char payDate[10];

public:
	Payment();
	Payment(double pay);
	Payment(int p_ID, double pAmount, char p_Date[]);
	void Display();
	void DisplayPaymentDetails();
	~Payment();
};
```

Payment.cpp

```
#include "Payment.h"
#include <cstring>
#include <iostream>
using namespace std;

Payment::Payment()
{
	payID = 1001;
	amount = 1500.0;
	strcpy_s(payDate, "2022-09-01");
}

Payment::Payment(double pay)
{
}

Payment::Payment(int p_ID, double pAmount, char p_Date[])
{
	payID = p_ID;
	amount = pAmount;
	strcpy_s(payDate, p_Date);
}

void Payment::Display()
{
}

void Payment::DisplayPaymentDetails()
{
	cout << "Payment ID : " << payID << endl
		<< "Amount : " << amount << endl
```

```
		<< "Pay Date : " << payDate << endl;
}


Payment::~Payment()
{
	cout << "Destructed" << endl;
}
```

Card.h

```
#include  "Payment.h"
class Card: public Payment
{
protected:
	char cardType[20];
	int cardNumber;
	int cvcNumber;
	char expiryMonth[10];
	int expiryYear;

public:
	Card();
	Card(char cType[], int cNo, int cvcNo, char exM[], int exY);
	void validPayment();
	void DoTransaction();
	void DisplayCardDetails();
	~Card();
};
```

Card.cpp

```
#include "Card.h"
#include "Payment.h"
#include <cstring>
#include <iostream>
using namespace std;


Card::Card()
{
	strcpy_s(cardType,"Credit Card");
	cardNumber = 4261398;
	cvcNumber = 345;
	strcpy_s(expiryMonth, "July");
	expiryYear = 2028;
}

Card::Card(char cType[], int cNo, int cvcNo, char exM[], int exY)
{
```

```
	strcpy_s(cardType, cType);
	cardNumber = cNo;
	cvcNumber = cvcNo;
	strcpy_s(expiryMonth, exM);
	expiryYear = exY;
}

void Card::validPayment()
{
}

void Card::DoTransaction()
{
}

void Card::DisplayCardDetails()
{
	cout << "Card Type : " << cardType << endl
		<< "Card Number : " << cardNumber << endl
		<< "CVC Number : " << cvcNumber << endl
		<< "Expiry Month : " << expiryMonth << endl
		<< "Expiry Year : " << expiryYear << endl;
}


Card::~Card()
{
	cout << "Destructed" << endl;
}
```

Reservation_l.h

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

class Reservation_l
{
private:
	Payment *pay[SIZE];
	char reservationID[5];
	char reservationDate[10];
	int labNumber;

public:
	Reservation_l();
	Reservation_l(double pay1, double pay2, char r_ID[], char r_Date[], int labNO);
	void DisplayReservationDetails();
	~Reservation_l();
};
```

Reservation_l.cpp

```
#include "Reservation_l.h"
#include <cstring>
#include <iostream>
using namespace std;


Reservation_l::Reservation_l()
{
	pay[0] = new Payment(1800.00);
	pay[1] = new Payment(1500.00);
	strcpy_s(reservationID, "898");
	strcpy_s(reservationDate, "2022.09.19");
	labNumber = 10;
}

Reservation_l::Reservation_l(double pay1, double pay2, char r_ID[], char r_Date[], int
labNO)
{
	pay[0] = new Payment(pay1);
	pay[1] = new Payment(pay2);
	strcpy_s(reservationID, r_ID);
	strcpy_s(reservationDate, r_Date);
	labNumber = labNO;

}

void Reservation_l::DisplayReservationDetails()
{
	for (int i = 0; i < SIZE; i++) {
		pay[i]->Display();
	}
}


Reservation_l::~Reservation_l()
{
	cout << "Reservation close" << endl;
	for (int i = 0; i<SIZE; i++)
		delete pay[i];
	cout << "The End" << endl;

}
```

LabTest.h

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

class LabTest
{
private:
	LabReport *l_report[SIZE];
	Payment *payment[SIZE];
	char testID[5];
	char testName[20];
	double testPrice;

public:
	LabTest();
	LabTest(int lrep1, int lrep2, char t_ID[], char t_Name[], double t_Price);
	void addPayment(Payment *payment1, Payment *payment2);
	void DisplayTestDetails();
	~LabTest();
};
```

LabTest.cpp

```
#include "LabTest.h"
#include <cstring>
#include <iostream>
using namespace std;


LabTest::LabTest()
{
	l_report[0] = new LabReport(101);
	l_report[1] = new LabReport(102);
	strcpy_s(testID, "T2314");
	strcpy_s(testName, "PCR");
	testPrice = 1500.0;
}

LabTest::LabTest(int lrep1, int lrep2, char t_ID[], char t_Name[], double t_Price)
{
	l_report[0] = new LabReport(lrep1);
	l_report[1] = new LabReport(lrep2);
	strcpy_s(testID, t_ID);
	strcpy_s(testName, t_Name);
	testPrice = t_Price;
}
```

```
void LabTest::addPayment(Payment * payment1, Payment * payment2)
{
	payment[0] = payment1;
	payment[1] = payment2;
}

void LabTest::DisplayTestDetails()
{
	for (int i = 0; i < SIZE; i++) {
		l_report[i]->DisplayReportDetails();
	}
	for (int i = 0; i < SIZE; i++) {
		payment[i]->DisplayPaymentDetails();
	}
}


LabTest::~LabTest()
{
	cout << "LabReport close" << endl;
	for (int i = 0; i < SIZE; i++) {
		delete l_report[i];
	}
	for (int i = 0; i < SIZE; i++) {
		delete payment[i];
	}

}
```

LabReport.h

```
class LabReport
{
private:
	char reportID[5];
	char reportName[20];

public:
	LabReport();
	LabReport(int lrep);
	LabReport(char r_ID[], char r_Name[]);
	void DisplayReportDetails();
	~LabReport();
};
```

LabReport.cpp

```
#include "LabReport.h"
#include <cstring>
#include <iostream>
using namespace std;


LabReport::LabReport()
{
	strcpy_s(reportID, "LR001");
	strcpy_s(reportName, "PCR Report");
}

LabReport::LabReport(int lrep)
{
}

LabReport::LabReport(char r_ID[], char r_Name[])
{
	strcpy_s(reportID, r_ID);
	strcpy_s(reportName, r_Name);
}

void LabReport::DisplayReportDetails()
{
	cout << "Lab Report ID : " << reportID << endl
		<< "Lab Report Name : " << reportName << endl;
}


LabReport::~LabReport()
{
	cout << "Destructed" << endl;
}
```

RegisteredUser.h

```
#include "Payment.h"
#define SIZE 2
class RegisteredUser
{
private:
	char u_Name[20];
	int u_Phone;
	char u_Email[20];
	char u_ID[4];
	char u_Password[15];
	Payment *pay[SIZE];

public:
	RegisteredUser();
	RegisteredUser(char uName[], int uPhone, char uMail[], char uID[], char uPass[]);
	void makePayment(Payment *pay);
	void DisplayUserDetails();
	~RegisteredUser();
};
```

RegisteredUser.cpp

```
#include "RegisteredUser.h"
#include <cstring>
#include <iostream>
using namespace std;


RegisteredUser::RegisteredUser()
{
	strcpy_s(u_Name, "Amal");
	u_Phone = 0712363276;
	strcpy_s(u_Email, "amal@gmail.com");
	strcpy_s(u_ID, "U001");
	strcpy_s(u_Password, "1234A");
	Payment *pay;
}

RegisteredUser::RegisteredUser(char uName[], int uPhone, char uMail[], char uID[], char
uPass[])
{
	strcpy_s(u_Name, uName);
	u_Phone = uPhone;
	strcpy_s(u_Email, uMail);
	strcpy_s(u_ID, uID);
	strcpy_s(u_Password, uPass);
}

void RegisteredUser::makePayment(Payment * pay)
```

```
{
}

void RegisteredUser::DisplayUserDetails()
{
	cout << "User Name : " << u_Name << endl
		<< "User Phone Number : " << u_Phone << endl
		<< "User Email : " << u_Email << endl
		<< "User ID : " << u_ID << endl
		<< "User Password : " << u_Password << endl;
}


RegisteredUser::~RegisteredUser()
{
	cout << "Destructed" << endl;
}
```

Staff.h

```
#include  "LabTest.h"
#include  "LabReport.h"
#define SIZE 2
class Staff
{
private:
	char s_Name[20];
	int s_Phone;
	char s_Email[10];
	char s_ID[5];
	double s_Salary;
	LabTest *test[SIZE];
	LabReport *lReport[SIZE];

public:
	Staff();
	Staff(char sName[], int sPhone, char sMail[], char sId[], double sSalary);
	void UpdateLabTest(LabTest *test);
	void UploadLabReport(LabReport *lReport);
	void DisplayStaffDetails();
	~Staff();
};
```

Staff.cpp

```
#include "Staff.h"
#include <cstring>
#include <iostream>
using namespace std;


Staff::Staff()
{
	strcpy_s(s_Name, "Kamal");
	s_Phone = 0716342633;
	strcpy_s(s_Email, "kamal@gmail.com");
	strcpy_s(s_ID, "S1912");
	s_Salary = 100000.0;
}

Staff::Staff(char sName[], int sPhone, char sMail[], char sId[], double sSalary)
{
	strcpy_s(s_Name, sName);
	s_Phone = sPhone;
	strcpy_s(s_Email, sMail);
	strcpy_s(s_ID, sId);
	s_Salary = sSalary;
}

void Staff::UpdateLabTest(LabTest *test)
{
}

void Staff::UploadLabReport(LabReport *lReport)
{
}

void Staff::DisplayStaffDetails()
{
	cout << "Staff Name : " << s_Name << endl
		<< "Staff Phone Number : " << s_Phone << endl
		<< "Staff Email : " << s_Email << endl
		<< "Staff ID : " << s_ID << endl
		<< "Staff Salary : " << s_Salary << endl;
}


Staff::~Staff()
{
	cout << "Destructed" << endl;
}
```

Feedback.h

```
class Feedback
{
private:
	int feedbackNo;
	char feedbackDescription[20];

public:
	Feedback();
	Feedback(int f_No, char f_Desc[]);
	void DisplayFeedback();
	~Feedback();
};
```

Feedback.cpp

```
#include "Feedback.h"
#include <cstring>
#include <iostream>
using namespace std;


Feedback::Feedback()
{
	feedbackNo = 1010;
	strcpy_s(feedbackDescription, "Exellent");
}

Feedback::Feedback(int f_No, char f_Desc[])
{
	feedbackNo = f_No;
	strcpy_s(feedbackDescription, f_Desc);
}


void Feedback::DisplayFeedback()
{
	cout << "Feedback Number : " << feedbackNo << endl
		<< "Feedback Description : " << feedbackDescription << endl;
}

Feedback::~Feedback()
{
	cout << "Destructed" << endl;
}
```

Report.h

```
#include "Payment.h"

class Report
{
private:
	int reportNumber;
	char reportName[20];

public:
	Report();
	Report(int reportNo, char report_Name[]);
	void DisplayReportDetails();
	~Report();
};
```

Report.cpp

```
#include "Report.h"
#include <cstring>
#include <iostream>
using namespace std;


Report::Report()
{
	reportNumber = 104;
	strcpy_s(reportName, "Financial Report");
}

Report::Report(int reportNo, char report_Name[])
{
	reportNumber = reportNo;
	strcpy_s(reportName, report_Name);
}

void Report::DisplayReportDetails()
{
	cout << "Report Number : " << reportNumber << endl
		<< "Report Name : " << reportName << endl;
}


Report::~Report()
{
	cout << "Destructed" << endl;
}
```

Main.cpp

```
#include "stdafx.h"
#include <iostream>
#include <cstring>

#include "Payment.h"
#include "Card.h"
#include "RegisteredUser.h"
#include "LabReport.h"
#include "LabTest.h"
#include "Reservation_l.h"
#include "Staff.h"
#include "Feedback.h"
#include "Report.h"


using namespace std;


int main()
{
	Payment *payment1 = new Payment();
	payment1->DisplayPaymentDetails();

	Card *card1 = new Card();
	card1->DisplayCardDetails();

	RegisteredUser *user1 = new RegisteredUser();
	user1->DisplayUserDetails();

	Feedback *feed1 = new Feedback();
	feed1->DisplayFeedback();

	LabReport *l_repoort1 = new LabReport();
	l_repoort1->DisplayReportDetails();

	Staff *staff1 = new Staff();
	staff1->DisplayStaffDetails();

	Report *report1 = new Report();
	report1->DisplayReportDetails();

	Reservation_l *reserv1 = new Reservation_l();
	reserv1->DisplayReservationDetails();

	LabTest *test1 = new LabTest();
	test1->DisplayTestDetails();

	Payment *payment1 = new Payment();
	Payment *payment2 = new Payment();
	payment1->DisplayPaymentDetails();
	payment2->DisplayPaymentDetails();


	delete payment1;
	delete card1;
```

```
        delete user1;
        delete feed1;
        delete l_repoort1;
        delete staff1;
        delete report1;
        delete reserv1;
        delete payment1;
        delete payment2;

    return 0;
}
```