Topic : **Laboratory Information Management System**

Group no : **KDY_14**

Campus : **Kandy**

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**
IT21276200	Samarakoon S.M.R.S.B	0768916609
IT21323966	Munasinghe M.I.M.P	0766281711
IT21261350	Thiwanka Kalpage	0763573777
IT21280238	Herath H.M.V.A.H	0701855399
IT21359910	Samarathunga U.P.G.S.P	0765791119

Description of the requirements

- There are five users of our system, namely scientist, senior scientist, lab assistant, lab manager and the administrator.
- The main functions of the system are to add and manage samples, to report the details and observations of the tests conducted, validate the results, to add new laboratories when we build new ones and to maintain a precise inventory for each laboratory.
- Besides these functions, the administrator should have a separate interface when logging into the system where he/she can add new users and alter existing ones.
- When a new employee is being hired to our facility, the administrator should fill a form which consists of first name, last name, NIC, email, phone number, specializing field and position. Every user has a separate ID.
- First, the lab manager adds a new lab to the system by filling out a form consisting of the name of the senior scientist in charge of the lab and the research area.
- Add equipment by filling out a form with equipment name and quantity and equipment ID.
- When the scientist or senior scientist logs onto the system, he/she can add/alter samples. They will fill out a form consisting of sample name, sample type, location, description, storage condition, quantity and date collected and a sample id.
- Then both the roles can select a sample to test on of the relevant lab, conduct the test and fill out a form in the system which consists of test ID, test name, test type, test conducted by, validation status, description, observations, and conclusions. The sample id should also be included with the information of the conducted tests.
- After the test is completed, there must be an option to validate the conducted test which only can be done by the senior scientist. However, at the time of validation, if a scientist is logged into the system, when validating the test, a login window should pop up for a senior scientist to log in.

Identifies Classes

User

Administrator

Scientist

Senior Scientist

Lab Assistant

Lab Manager

Sample

Test

Equipment

Lab

CRC Cards

User	
Responsibility	Collaborators
Visit the website	
Login to the system	
Store User Details	Administrator

Lab Assistant	
Responsibility	Collaborators
Add Equipment	Equipment
Manage Equipment	Equipment

Administrator	
Responsibility	Collaborators
Add User	User
Manage User	User

Lab Manager	
Responsibility	Collaborators
Add Lab	Lab
Manage Lab	Lab

Scientist	
Responsibility	Collaborators
Add Sample	Sample
Conduct Test	Test
Manage Sample	Sample
Manage Test	Test
Viwe Equipment	Equipment

Senior Scientist	
Responsibility	Collaborators
Add Sample	Sample
Conduct Test	Test
Manage Sample	Sample
Validate Test	Test
Manage Test	Test
Viwe Equipment	Equipment

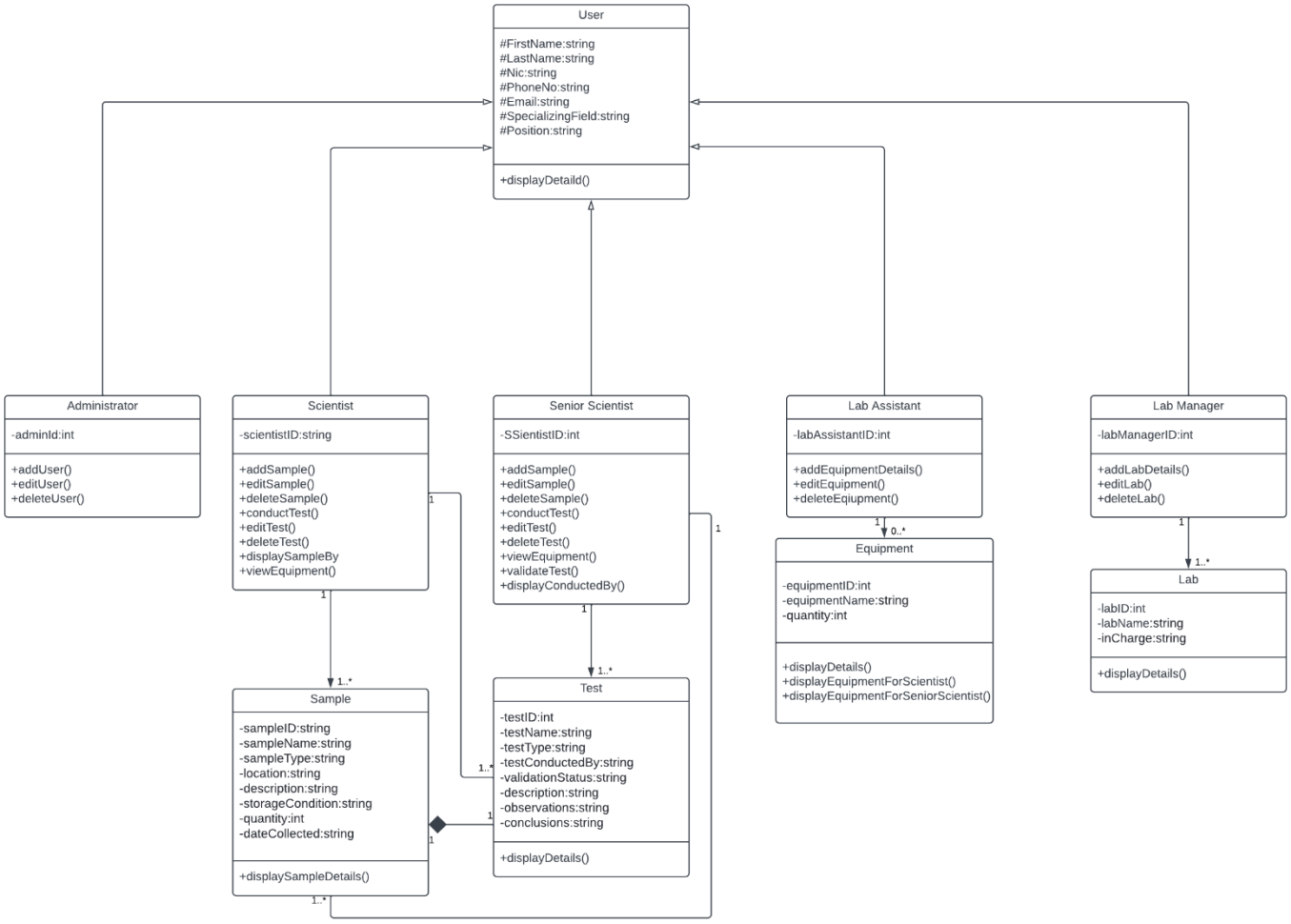
Sample	
Responsibility	Collaborators
Store Sample Details	Scientist
	Senior Scientist

Test	
Responsibility	Collaborators
Store Test Details	Scientist
	Senior Scientist

Equipment	
Responsibility	Collaborators
Store Equipment Details	Lab Assistant

Lab	
Responsibility	Collaborators
Store Lab Details	Lab Manager

Class Diagram



Individual contribution

Name	ID	Contribution
Samarakoon S.M.R.S.B	IT21276200	CRC Cards/Class Diagram/Coding: User, Administrator
Munasinghe M.I.M.P	IT21323966	CRC Cards/Class Diagram/Coding: Sample, Scientist
Thiwanka Kalpage	IT21261350	CRC Cards/Class Diagram/Coding: Lab Assistant, Equipment
Herath H.M.V.A.H	IT21280238	CRC Cards/Class Diagram/Coding: Test, Senior Scientist
Samarathunga U.P.G.S.P	IT21359910	CRC Cards/Class Diagram/Coding: Lab Manager, Lab

Code

```
#include <iostream>
#include <cstring>

using namespace std;

class User
{
protected:
	string FirstName;
	string LastName;
	string Nic;
	string PhoneNo;
	string SpecializingField;
	string Position;

public:
	User() {}

	User(string ufname, string ulname, string unic, string upno, string usf, string upos)
	{
		FirstName = ufname;
		LastName = ulname;
		Nic = unic;
		PhoneNo = upno;
		SpecializingField = usf;
		Position = upos;
	}
```

```
        void displayDeatails()
        {
                cout << "First Name : " << FirstName << endl;
                cout << "Last Name : " << LastName << endl;
                cout << "Nic : " << Nic << endl;
                cout << "PhoneNo : " << PhoneNo << endl;
                cout << "Specializing Field : " << SpecializingField << endl;
                cout << "Position : " << Position << endl;
        }

        ~User()
        {
                cout << "--> Delete User" << endl;
        }

};

class Administrator :public User
{
private:
        int adminid;

public:

        Administrator() {}

        Administrator(string ufname, string ulname, string unic, string upno, string usf,
string upos, int aid) :User(ufname, ulname, unic, upno, usf, upos)
```

```
	{
		adminid = aid;
	}

	void addUser() {};

	void editUser() {};

	void deleteUser() {};

	void displayDeatails()
	{
		cout << "First Name : " << FirstName << endl;
		cout << "Last Name : " << LastName << endl;
		cout << "Nic : " << Nic << endl;
		cout << "PhoneNo : " << PhoneNo << endl;
		cout << "Position : " << Position << endl;
		cout << "Admini Id : " << adminid << endl;
	}

	~Administrator() {
		cout << "--> Delete Administrator" << endl;
	}

};

class Lab
{
```

```
private:
        string Incharge;
        string Labname;
        int LabID;
public:

        Lab();

        Lab(string incharge, string labname, int labid)
        {
                Incharge = incharge;
                Labname = labname;
                LabID = labid;
        }

        void displaydetails()
        {
                cout << "Incharge : " << Incharge << endl;
                cout << "Labname : " << Labname << endl;
                cout << "Lab ID : " << LabID << endl;
        }

};

class LabManager : public User
{
private:
        int labManagerID;
```

```
        Lab* lab;
public:

        LabManager();

        LabManager(string ufname, string ulname, string unic, string upno, string usf,
string upos, int lmid, Lab* l) :User(ufname, ulname, unic, upno, usf, upos)
        {
                labManagerID = lmid;
                lab = l;
        }

        void addlabDetails()
        {
                cout << "Lab Manager ID :" << labManagerID << endl;
                lab->displaydetails();
        }

        void displayLabManagerDetails() {
                cout << endl;
                cout << "First Name : " << FirstName << endl;
                cout << "Last Name : " << LastName << endl;
                cout << "Nic : " << Nic << endl;
                cout << "PhoneNo : " << PhoneNo << endl;
                cout << "Position : " << Position << endl;
                cout << "Lab Manager Id : " << labManagerID << endl;
                cout << endl;
        }
```

```
        ~LabManager() {}

        void addlab();

        void editlab();

        void deletelab();

};

class Scientist;
class SScientist;

class Equipment
{
private:
        string equipmentID;
        string equipmentName;
        int quantity;
public:

        Equipment();

        Equipment(string iequipmentID, string iname, int iquantity)
        {
                equipmentID = iequipmentID;
                equipmentName = iname;
                quantity = iquantity;
```

```
	}

	void displayDetails()
	{
		cout << endl;
		cout << " Equipment ID : " << equipmentID << endl;
		cout << " Equipment Name : " << equipmentName << endl;
		cout << " Quantity : " << quantity << endl;
		cout << endl;
	}

	~Equipment()
	{
		cout << "--> Delete Equipment " << endl;
	}
};

class LabAssistant : public User
{
private:
	int labAssistantID;
	Equipment* eqmt; //an object of Equipment as attribute
public:
	LabAssistant();
	LabAssistant(string ufname, string ulname, string unic, string upno, string usf,
string upos, int ilabAssistantID, Equipment* e) :User(ufname, ulname, unic, upno, usf,
upos)
	{
		labAssistantID = ilabAssistantID;
```

```
        eqmt = e;
}

void addEquipmentDetails()
{
        cout << " Lab_Assistant's ID : " << labAssistantID << endl;
        eqmt->displayDetails();
}

void displayLabAssistantDetails() {
        cout << endl;
        cout << "First Name : " << FirstName << endl;
        cout << "Last Name : " << LastName << endl;
        cout << "Nic : " << Nic << endl;
        cout << "PhoneNo : " << PhoneNo << endl;
        cout << "Position : " << Position << endl;
        cout << "Lab Assistant Id : " << labAssistantID << endl;
        cout << endl;
}

~LabAssistant()
{
        cout << "--> Delete Lab_Assistant " << endl;
}
void addEquipment();

void editEquipment();
```

```
        void deleteEquipment();


};


class Scientist;


class Sample {
private:
        string sampleID;
        string sampleName;
        string sampleType;
        string location;
        string description;
        string storageCondition;
        int quantity;
        string dateCollected;
public:
        Sample() {}//default constructor


        Sample(string sampID, string sampName, string sampType, string sampLocation,
string sampDescription, string sampStorageCondition, int sampQuantity, string
sampDateCollected) {
                sampleID = sampID;
                sampleName = sampName;
                sampleType = sampType;
                location = sampLocation;
                description = sampDescription;
                storageCondition = sampStorageCondition;
                quantity = sampQuantity;
```

```
            dateCollected = sampDateCollected;

        }//overloaded constructor

        void displaySampleDetails() {
            cout << endl;
            cout << "Sample details" << endl;
            cout << "Sample ID : " << sampleID << endl;
            cout << "Sample Name : " << sampleName << endl;
            cout << "Sample Type : " << sampleType << endl;
            cout << "Location : " << location << endl;
            cout << "Description : " << description << endl;
            cout << "Storage Condition : " << storageCondition << endl;
            cout << "Quantity : " << quantity << endl;
            cout << "Date Collected : " << dateCollected << endl;
            cout << endl;
        }

        ~Sample() {
            cout << "--> Delete Sample" << endl;
        }//destructor
};

class Test;

class Scientist : public User {
private:
        Sample* sampNew; //association relationship
```

```
        Test* test1;


        string scientistID;
        Sample* samp;
public:
        Scientist() {}//default constructor
        Scientist(string ufname, string ulname, string unic, string upno, string usf, string
upos, string sID, Sample* s) : User(ufname, ulname, unic, upno, usf, upos) {
                scientistID = sID;
                samp = s;
        }//overloaded constructor


        void displaySampleBy() {
                cout << "Sample added by :" << scientistID << endl;
                samp->displaySampleDetails();
        }


        void displayScientist() {
                cout << "Scientist : " << scientistID << endl << endl;
        }


        void displayScientistDetails() {
                cout << endl;
                cout << "First Name : " << FirstName << endl;
                cout << "Last Name : " << LastName << endl;
                cout << "Nic : " << Nic << endl;
                cout << "PhoneNo : " << PhoneNo << endl;
                cout << "Specializing Field : " << SpecializingField << endl;
                cout << "Position : " << Position << endl;
```

```
            cout << "Scientist Id : " << scientistID << endl;
            cout << endl;
        }

        void addSample(Sample* sampNew) {} //one way association relation between
scintist and sample
        void editSample() {}

        void addTest(Test* test1) {} //bi directional association between scientist and test
        void editTest() {}

        void deleteSample() {}

        void deleteTest() {}

        void viewEquipment() {}

        ~Scientist() {
            cout << "--> Delete Scientist" << endl;
        }

};

class Test
{
private:
        int testID;
        string testName;
        string testType;
```

```
        string testConBy;
        string validStatus;
        string description;
        string observation;
        string conclusion;
public:

        Test();

        Test(int tID, string tName, string tType, string tConBy, string
                ValSt, string desc, string obs, string concl)
        {
                testID = tID;
                testName = tName;
                testType = tType;
                testConBy = tConBy;
                validStatus = ValSt;
                description = desc;
                observation = obs;
                conclusion = concl;
        }

        void displayDetails()
        {
                cout << endl;
                cout << "Test ID : " << testID << endl;
                cout << "Test Name : " << testName << endl;
                cout << "Test Type : " << testType << endl;
```

```
            cout << "Test Conducted By : " << testConBy << endl;
            cout << "Validation Status : " << validStatus << endl;
            cout << "Discription : " << description << endl;
            cout << "Observation : " << observation << endl;
            cout << "Conclusion : " << conclusion << endl;
            cout << endl;
        }
};

class SScientist : public User
{
private:
        Sample* samp_New; //association relationship
        Test* test_1;
        int ssID;
        Test* conduct; //an object of Test as attribute
        // Sample *add ;//an object of Sample as attribute

public:
        SScientist() {}
        SScientist(string ufname, string ulname, string unic, string upno, string usf, string
upos, int pssID, Test* c) :
            User(ufname, ulname, unic, upno, usf, upos)
        {
            ssID = pssID;
            conduct = c;
        }

        void displayConductedBy()
```

```
{
        cout << "Conducted By :" << ssID << endl;
        conduct->displayDetails();
}

void displaySeniorScientistDetails() {
        cout << endl;
        cout << "First Name : " << FirstName << endl;
        cout << "Last Name : " << LastName << endl;
        cout << "Nic : " << Nic << endl;
        cout << "PhoneNo : " << PhoneNo << endl;
        cout << "Specializing Field : " << SpecializingField << endl;
        cout << "Position : " << Position << endl;
        cout << "Senior Scientist Id : " << ssID << endl;
        cout << endl;
}

void addSample(Sample* samp_New) {}

void editSample() {}

void deleteSample() {}

void conductTest(Test* test_1) {}

void EditTest() {}

void deleteTest() {}
```

```
        void viewEquip() {}


        void validateTest() {}

};


int main() {

        //creating objects

        User* u1 = new User("Sandalu", "Samarakoon", "200032501069", "0776588766",
"Cancer", "User");

        Administrator* a1 = new Administrator("Sandalu", "Samarakoon",
"200112343212", "0706588789", "", "Administrator", 1000);

        Lab* lb1 = new Lab("Gavesh", "cancer", 03);

        LabManager* m1 = new LabManager("Gavesh", "Samarathunga",
"200112343212", "0766382391", "", "Lab Manager", 1004, lb1);


        Equipment* e1 = new Equipment("4001", "Intensity Modulated Radiation Therapy
/ IMRT", 3);

        LabAssistant* l1 = new LabAssistant("Thiwanka", "Kalpage", "200133951026",
"0760357846", "", "Lab Assistant", 1003, e1);


        Sample* s1 = new Sample("5001", "Leukemia", "Complete blood sample",
"Storage wing B", "promising sample", "freezing", 3, "2020-02-22");

        Scientist* sci1 = new Scientist("Manuja", "Munasinghe", "200112343212",
"0766241711", "Cancer", "Scientist", "1001", s1);


        Test* t1 = new Test(6001, "name1", "type1", "Mr.ABC", "valid", "decription",
"observation", "conclusion");

        SScientist* ss1 = new SScientist("Viduni", "Herath", "20004567890",
"0706578765", "Cancer", "Senior Scientist", 1002, t1);


        //calling methods //FROM
```

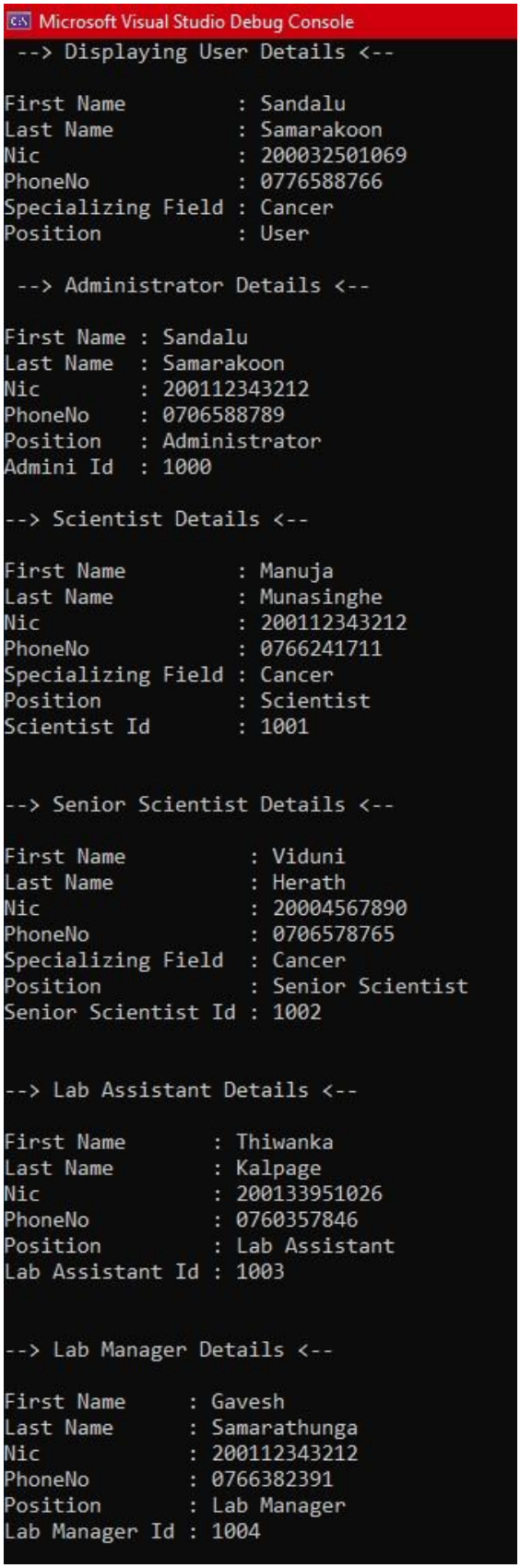
```
        cout << " --> Displaying User Details <--" << endl << endl;
        u1->displayDeatails();// void User:: displayDeatails()
        cout << endl;
        cout << " --> Administrator Details <--" << endl << endl;
        a1->displayDeatails();//void Administrator:: displayDeatails()
        cout << endl;

        cout << "--> Scientist Details <--" << endl;
        sci1->displayScientistDetails();
        cout << endl;

        cout << "--> Senior Scientist Details <--" << endl;
        ss1->displaySeniorScientistDetails();
        cout << endl;

        cout << "--> Lab Assistant Details <--" << endl;
        l1->displayLabAssistantDetails();
        cout << endl;

        cout << "--> Lab Manager Details <--" << endl;
        m1->displayLabManagerDetails();
        cout << endl;

        cout << "**********************************************" << endl <<
endl;
        cout << " --> Display Lab Details <--" << endl;
        cout << endl;
        lb1->displaydetails();//void displaydetails()
        cout << endl;
```

```
        cout << " --> Display Lab Details added by <--" << endl << endl;
        m1->addlabDetails(); //void addlabDetails()
        cout << endl;
        cout << "**************************************************" << endl <<
endl;
        cout << " --> Displaying Equipment Details <--" << endl;
        e1->displayDetails();//void Equipment::displayDetails()
        cout << " --> Displaying Equipment Details added by<--" << endl << endl;
        l1->addEquipmentDetails(); //void LabAssistant::addEquipmentDetails()
        cout << "**************************************************" << endl <<
endl;

        cout << " --> Display Sample Details <--" << endl;
        s1->displaySampleDetails(); //void Sample::displaySampleDetails()
        cout << endl;
        cout << " --> Display Sample Added by <--" << endl << endl;
        sci1->displaySampleBy();//void Scientist::displaySampleBy()
        cout << endl;
        cout << "**************************************************" << endl <<
endl;
        cout << " --> Displaying Test Details <--" << endl;
        t1->displayDetails(); //void Test::displayDetails()
        cout << endl;
        cout << " --> Display Test Conducted by <--" << endl << endl;
        ss1->displayConductedBy(); //void SScientist::displayConductedBy()
        cout << endl;
        cout << "**************************************************" << endl <<
endl;
```

```
//deallocate memory
        delete u1;
        delete a1;
        delete lb1;
        delete m1;
        delete e1;
        delete l1;
        delete s1;
        delete sci1;
        delete t1;
        delete ss1;

}
```

Output

Details of added users



Relationship between lab assistant and lab

```
**************************************************

 --> Display Lab Details <--

Incharge       : Gavesh
Labname        : cancer
Lab ID         : 3

 --> Display Lab Details added by <--

Lab Manager ID :1004
Incharge       : Gavesh
Labname        : cancer
Lab ID         : 3

**************************************************
```

Relationship between Equipment and lab assistant

```
**************************************************

 --> Displaying Equipment Details <--

 Equipment ID   : 4001
 Equipment Name : Intensity Modulated Radiation Therapy / IMRT
 Quantity       : 3

 --> Displaying Equipment Details added by<--

 Lab_Assistant's ID : 1003

 Equipment ID   : 4001
 Equipment Name : Intensity Modulated Radiation Therapy / IMRT
 Quantity       : 3

**************************************************
```

Added Sample Details and relationship between scientist

```
**************************************************

 --> Display Sample Details <--

Sample details
Sample ID          : 5001
Sample Name        : Leukemia
Sample Type        : Complete blood sample
Location           : Storage wing B
Description        : promising sample
Storage Condition  : freezing
Quantity           : 3
Date Collected     : 2020-02-22


 --> Display Sample Added by <--

Sample added by :1001

Sample details
Sample ID          : 5001
Sample Name        : Leukemia
Sample Type        : Complete blood sample
Location           : Storage wing B
Description        : promising sample
Storage Condition  : freezing
Quantity           : 3
Date Collected     : 2020-02-22


**************************************************
```

Added test details and relationship between senior scientist and test

```
**************************************************

 --> Displaying Test Details <--

Test ID           : 6001
Test Name         : name1
Test Type         : type1
Test Conducted By : Mr.ABC
Validation Status : valid
Discription       : decription
Observation       : observation
Conclusion        : conclusion


 --> Display Test Conducted by <--

Conducted By :1002

Test ID           : 6001
Test Name         : name1
Test Type         : type1
Test Conducted By : Mr.ABC
Validation Status : valid
Discription       : decription
Observation       : observation
Conclusion        : conclusion


**************************************************
```

Deleting Objects

```
--> Delete User
--> Delete Administrator
--> Delete User
--> Delete User
--> Delete Equipment
--> Delete Lab_Assistant
--> Delete User
--> Delete Sample
--> Delete Scientist
--> Delete User
--> Delete User
```