Topic : Online Medical Portal

Group no : MLB_05.01_05

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**
IT21287954	Manuditha P. M. M. S.	0710900426
IT21286728	Dassanayake D. M. P. J.	0710663066
IT21289484	K. D. R. Manditha	0712193303
IT21286346	Wijerathne D. A. M. Y. R.	0770244755
IT21286650	Senevirathna D. M. O. C.	0769980116

System Requirements

1. The system should function 24/7/365.

2. Guests can overview the system, to use the system, they must register with the system by providing

Such as Full name, Address, Email-Address, NIC, Blood-type, DOB and Gender.

3. After the registration, they receive a user account and they can set the user name and the password.

4. Patients can channel doctors and give feedbacks after the service.

5. Patients need to do the payment to channel a doctor.

6. Doctors can log in to the system using their doctor id and password.

7. Doctors check lab reports and conform the medicines that required by the patient.

8. The admin login to the system using admin id and the password.

9. Admin manage the user accounts such as deleting non-use accounts and conform the new accounts.

10. Admin also check and keep on track about the medicines, checking the available medicines and the

Medicines need to refill.

11. Admin can manage the system portals using the admin's account.

12. Admin also contact suppliers and work on supplies.

13. Suppliers can log into the system using their supplier id and the password

Noun & Verb Analysis

(NOUNS)

1. The system should function 24/7/365.

2. Guests can overview the system, to use the system, they must register with the system by providing

   Such as Full name, Address, Email-Address, NIC, Blood-type, DOB and Gender.

3. After the registration, they receive a user account and they can set the user name and the password.

4. Patients can channel doctors and give feedbacks after the service.

5. Patients need to do the payment to channel a doctor.

6. Doctors can log in to the system using their doctor id and password.

7. Doctors check lab reports and confirm the medicines that required by the patient.

8. The admin login to the system using admin id and the password.

9. Admin manage the user accounts such as deleting non-use accounts and confirm the new accounts.

10. Admin also check and keep on track about the medicines, checking the available medicines and the

    Medicines need to refill.

11. Admin can manage the system portals using the admin's account.

12. Admin also contact suppliers and work on supplies.

13. Suppliers can log into the system using their supplier id and the password.

Identified Classes

- Guest
- Patient
- Channel
- Doctor
- Feedback
- Payment
- Lab report
- Medicine
- Admin
- Suppliers

Reasons for rejecting other nouns

- Redundant - User
- An Event or an operation – Channel doctor
- Outside scope of system – System
- Meta-language – They , Service
- Attributes - Full name, Address, Email address, NIC, Blood-type, DOB and Gender, User Name, Password, Doctor id, Admin id, Non use accounts, New accounts, Admin's account, Supplier id

Noun & Verb Analysis

(VERBS)

1. The system should function 24/7/365.

2. Guests can overview the system, to use the system, they must register with the system by providing

Such as Full name, Address, Email-Address, NIC, Blood-type, DOB and Gender.

3. After the registration, they receive a user account and they can set the user name and the password.

4. Patients can channel doctors and give feedbacks after the service.

5. Patients need to do the payment to channel a doctor.

6. Doctors can log in to the system using their doctor id and password.

7. Doctors check lab reports and conform the medicines that required by the patient.

8. The admin login to the system using admin id and the password.

9. Admin manage the user accounts such as deleting non-use accounts and conform the new accounts.

10. Admin also check and keep on track about the medicines, checking the available medicines and the

Medicines need to refill.

11. Admin can manage the system portals using the admin's account.

12. Admin also contact suppliers and work on supplies.

13. Suppliers can log into the system using their supplier id and the password.

Methods

- Guest – Register to the system by providing details view the system.
- Patient – Login to the system by entering details
  Can channel the doctors
  Do the payments
  Give feedback.
- Channel – Schedule the time and date
  Display the charge
  Display the instructions.
- Doctor – Login to the system
  Check the lab reports
  Confirm the medicine.
- Feedback – Add feedback
  Display feedback
- Payment – Add payment details
  Display payment details.
- Lab reports – Store the patient's reports
  Display the reports.
- Medicine – Refill the medicine
  Display medicine list and quantity
  Update medicine details.
- Admin – Login to the system
  Manage the user accounts
  Manage the medicine
  Manage the system portals
  Handle the suppliers
- Supplier – Login to the system
  Supply the medicine
  Update the medicine prices up to date.

CRC Cards

Class name: Guest	
Responsibilities:	**Collaborations:**
Register to the system	
Allow to view the medicine	Medicine

Class name: Channel	
Responsibilities:	**Collaborations:**
Schedule the time and date	Doctor, Patient
Display the charge	Payment
Display the instructions	Doctor, Patient

Class name: Patient	
Responsibilities:	**Collaborations:**
Can channel the doctors	Doctor
Give feedback	Feedback
Do the payments	Payment

Class name: Doctor	
Responsibilities:	**Collaborations:**
Login to the System	
Check the lab reports	Lab reports
Confirm the medicine	Medicine

Class name: Feedback	
Responsibilities:	**Collaborations:**
Add feedback	
Display feedback	

Class name: Payment	
Responsibilities:	**Collaborations:**
Add payment details	Medicine, Channel
Display payment details	

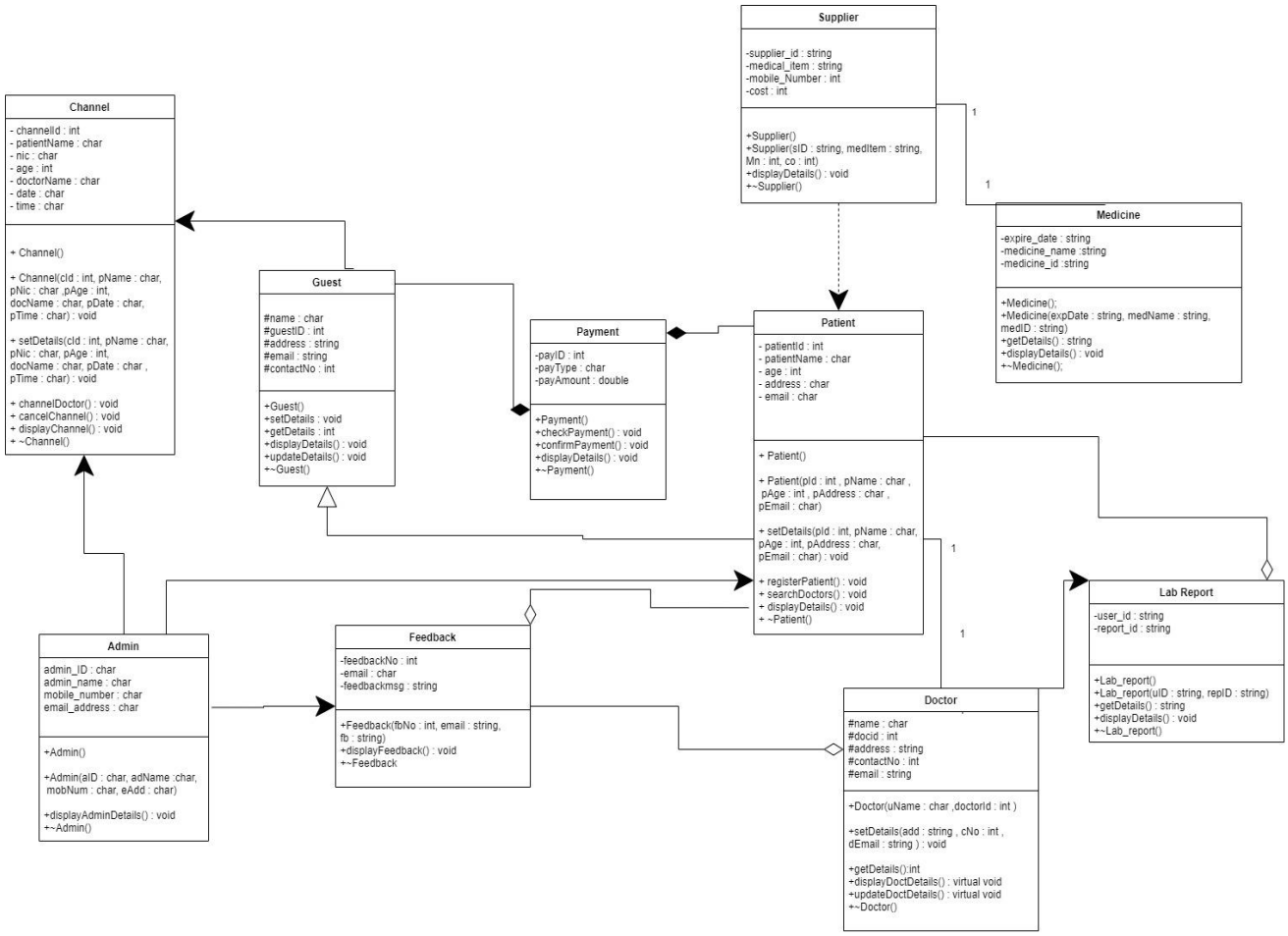
Class name: Lab reports	
Responsibilities:	**Collaborations:**
Store the patient's reports	Patient
Display the reports	

Class name: Supplier	
Responsibilities:	**Collaborations:**
Login to the system	
Supply the medicine	Admin
Update the medicine prices up to date	

Class name: Medicine	
Responsibilities:	**Collaborations:**
Refill the medicine	Admin
Display medicine list and quantity	
Update medicine details	

Class name: Admin	
Responsibilities:	**Collaborations:**
Login to the System	
Manage the user accounts	Patient, Doctor, Supplier
Manage the medicine	Medicine, Supplier
Manage the system portals	
Handle the suppliers	Suppliers

Class Diagram (UML Notation)



Code files

Patient.h

```
class Patient
{
private:
		int patientId;
		char patientName[40];
		int age;
		char address[70];
		char email[40];

public:
		Patient();
		Patient(int pId, char pName[40], int pAge, char pAddress[70], char pEmail[40]);
		void setDetails(int pId, char pName[40], int pAge, char pAddress[70], char pEmail[40]);
		void registerPatient();
		void searchDoctors();
		void displayDetails();
		~Patient();

};
```

Patient.cpp

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

Patient::Patient()
{
		patientId = 0;
		strcpy(patientName, "");
		age = 0;
		strcpy(address, "");
		strcpy(email, "");
}

Patient::Patient(int pId, char pName[40], int pAge, char pAddress[70], char pEmail[40])
{
		patientId = pId;
		strcpy(patientName, pName);
		age = pAge;
		strcpy(address, pAddress);
		strcpy(email, pEmail);
}

void Patient::setDetails(int pId, char pName[40], int pAge, char pAddress[70], char pEmail[40])
{
		patientId = pId;
		strcpy(patientName, pName);
		age = pAge;
		strcpy(address, pAddress);
		strcpy(email, pEmail);
}

void Patient::registerPatient()
{
}
```

```
void Patient::searchDoctors()
{
}



void Patient::displayDetails()
{
	cout << "Patient Details\n" << endl;
	cout << "Patient ID : " << patientId << endl;
	cout << "Patient Name: " << patientName << endl;
	cout << "Patient Age : " << age << endl;
	cout << "Patient Address : " << address << endl;
	cout << "Patient Email : " << email << endl;
}


Patient :: ~Patient()

{

	cout << "\nDestructor runs on Patient class" << endl;

}
```

Channel.h

```
class Channel
{
	private:
		int channelId;
		char patientName[40];
		char nic[15];
		int age;
		char doctorName[50];
		char date[20];
		char time[8];

	public:
		Channel();
		Channel(int cId, char pName[40], char pNic[15], int pAge, char docName[50], char pDate[20], char pTime[8]);
		void setDetails(int cId, char pName[40], char pNic[15], int pAge,  char docName[50], char pDate[20], char pTime[8]);
		void channelDoctor();
		void cancelChannel();
		void displayChannel();
		~Channel();
};
```

Channel.cpp

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

Channel::Channel()
{
	channelId = 0;
	strcpy(patientName, "");
	strcpy(nic, "");
	age = 0;
	strcpy(doctorName, "");
```

```
		strcpy(date, "");
		strcpy(time, "");
}

Channel::Channel(int cId, char pName[40], char pNic[15], int pAge, char docName[50], char pDate[20], char pTime[8])
{
		channelId = cId;
		strcpy(patientName, pName);
		strcpy(nic, pNic);
		age = pAge;
		strcpy(doctorName, docName);
		strcpy(date, pDate);
		strcpy(time, pTime);
}

void Channel::setDetails(int cId, char pName[40], char pNic[15], int pAge,  char docName[50], char pDate[20], char pTime[8])
{
		channelId = cId;
		strcpy(patientName, pName);
		strcpy(nic, pNic);
		age = pAge;
		strcpy(doctorName, docName);
		strcpy(date, pDate);
		strcpy(time, pTime);
}

void Channel::channelDoctor()
{

}

void Channel::cancelChannel()
{

}

void Channel::displayChannel()
{
		cout << "\nChannel Details\n" << endl;
		cout << "Channel id : " << channelId << endl;
		cout << "Patient name : " << patientName << endl;
		cout << "Patient nic : " << nic << endl;
		cout << "Patient age :" << age << endl;
		cout << "Doctor name : " << doctorName << endl;
		cout << "Channel date : " << date << endl;
		cout << "Channel time : " << time << endl;
}

Channel :: ~Channel()
{
		cout << "\nDestructor runs on channel class" << endl;
}
```

Doctor.h

```
class Doctor
{
		private:
		char name[50];
		int docid;
		char address[30];
		int contactNo;
		char email[30];

		public:
		Doctor(const char uName[], int doctorId); void setDetails(char add[30], int cNo, char dEmail[30]);
		int getDetails();
		virtual void displayDoctDetails();
		virtual void updateDoctDetails();
		~Doctor();
};;
```

Doctor.cpp

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

Doctor::Doctor(const char uName[], int doctorId)
{
strcpy(name, uName);
docid = doctorId;
}

void Doctor::setDetails(char add[30], int cNo, char dEmail[30])
{
}

int Doctor::getDetails()
{
}

void Doctor::displayDoctDetails()
{
}

void Doctor::updateDoctDetails()
{
}

Doctor::~Doctor()
{
}
```

Feedback.h

```
class Feedback
{
		private:
		int feedbackNo;
		char email[40];
		char feedbackmsg[40];
```

```
        public:
        Feedback(int fbNo, char mail[30], char fb[30]);
        void displayFeedback();
        ~Feedback();

};
```

Feedback.cpp

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

Feedback::Feedback(int fbNo, char mail[30], char fb[30])
{
feedbackNo = fbNo;
strcpy(email, mail);
strcpy(feedbackmsg, fb);
}

void Feedback::displayFeedback()
{
}

Feedback::~Feedback()
{
}
```

Admin.h

```
class Admin{
        private:
                char admin_ID[20];
                char admin_name[50];
                char mobile_number[15];
                char email_address[50];
        public:
                Admin();
                Admin(char aID[], char adName[], char mobNum[], char eAdd[]);
                void displayAdminDetails();
                ~Admin();
};
```

Admin.cpp

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

Admin::Admin()
{
        strcpy(admin_ID,"");
        strcpy(admin_name,"");
        strcpy(mobile_number,"000000000000000");
        strcpy(email_address,"");
```

```
}
Admin::Admin(char aID[], char adName[], char mobNum[], char eAdd[])
{
	strcpy(admin_ID,aID);
	strcpy(admin_name,adName);
	strcpy(mobile_number,mobNum);
	strcpy(email_address,eAdd);
}
void Admin::displayAdminDetails()
{

}
Admin::~Admin()
{

}
```

Supplier.h

```
class Supplier{

	private:
		string supplier_id;
		string medical_item;
		int mobile_Number;
		int cost;
	public:
		Supplier();
		Supplier(string sID, string medItem, int Mn, int co);
		void displayDetails();
		~Supplier();
};
```

Supplier.cpp

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

Supplier::Supplier()
{
	supplier_id = "";
	medical_item = "";
	mobile_Number = 0;
	cost = 0;
}

 Supplier::Supplier(string sID, string medItem, int Mn, int co)
{
	supplier_id = sID;
	medical_item = medItem;
	mobile_Number = Mn;
	cost = co;
}

void Supplier::displayDetails()
{
```

```
}

Supplier::~Supplier()
{

}
```

Guest.h

```
class Guest
{
private:
char name[50]; int guestID; char address[50];
char email[20]; int contactNo;
public:
Guest(const char gName[], int gId);
void setDetails(char add, int cNo, char gEmail); int getDetails();
virtual void displayDetails(); virtual void updateDetails();
~Guest();
};
```

Guest.cpp

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

Guest::Guest(const char gName[], int gId)
{
 strcpy(name, gName);
 gId = guestID;
}

void Guest::setDetails(char add, int cNo, char gEmail)
{
}

int Guest::getDetails()
{
}

void Guest::displayDetails()
{
}

void Guest::updateDetails()
{
}

Guest::~Guest()
{
}
```

Payment.h

```
class Payment
{
private:
int payID;
char payType[20];
double payAmount;

public:
Payment();
Payment(int pID, const char payType[], double payAmount);
void checkPayment();
void confirmPayment();
void displayDetails();
void updateDetails();
~Payment();
};
```

Payment.cpp

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

Payment::Payment(int pID, const char ppayType[], double ppayAmount)
{
payID = pID;
strcpy (payType, ppayType);
payAmount = ppayAmount;

  }

void Payment::checkPayment()
{

}

void Payment::confirmPayment()
{

}
void Payment::displayDetails()
{

}


Payment::~Payment()
{

}
```

Medicine.h

```
class Medicine{
        private:
                char expire_date;
                char medicine_name;
                char medicine_id;
        public:
                Medicine();
                Medicine(char expDate, char medName, char medID);
                char getDetails();
                void displayDetails();
                  ~Medicine();
};
```

Medicine.cpp

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

Medicine::Medicine()
  {
        expire_date = "";
        medicine_name = "";
        medicine_id = "";
  }

Medicine::Medicine(char expDate, char medName, char medID)
  {
        expire_date = expDate;
        medicine_name = medName;
        medicine_id = medID;
  }

char Medicine::getDetails()
  {

  }

void Medicine::displayDetails()
  {

  }

Medicine::~Medicine()
  {

  }
```

LabReport.h

```
class Lab_report{
		private:
				char user_id;
				char report_id;

		public:
				Lab_report();
				Lab_report(char uID, char repID);
				char getDetails();
				void displayDetails();
	~Lab_report();
};
```

LabReport.cpp

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

Lab_report::Lab_report()
  {
		user_id ="";
		report_id ="";
  }
Lab_report::Lab_report(char uID, char repID)
  {
		user_id =uID;
		report_id =repID;
  }

char Lab_report::getDetails()
  {

  }

void Lab_report::displayDetails()
  {

  }

Lab_report::~Lab_report();
  {

  }
```

Main Program

main.cpp

```
#include "Admin.h"
#include "Channel.h"
#include "doctor.h"
#include "feedback.h"
#include "Lab_report.h"
#include "Medicine.h"
#include "Patient.h"
#include "Supplier.h"
#include "Guest.h"
#include "Payment.h"

#include <iostream>
using namespace std;

int main()
{
 //creating objects

   Doctor* doctor;
   doctor= new Doctor("Dr.Wijerathna", 1990); //Create Doctor object

   Feedback fbc(35, "example@xyz.com", "feedback1"); //Create Feedback object

   Medicine* medicine;
   medicine= new medicine("12/4/2023", "oxygen", "146"); //Creating medicine object

   Lab_report* labReport;
   labReport= new labReport("35198", "lab198j"); //Creating lab report object

   Patient p1(23, "Kasun", 23, "fsdfds", "0712345678", "hasdsahbd@gmail.com");

   Channel c1(01, "Kasun", "4343432", 24, "4324", "DR can", "2022", "12.56");

   Guest* guest;
   guest = new guest("Perera", 1988); //Create Guest object

   Payment * pay;
   pay = new Payment("VISA" , "Nimal Perera",); //Create Payment object

   Admin *login = new Admin();
            login->Admin("IT","Walter","0712178345","Walterwhite@gmail.com"); //creating Admin object


 Supplier *details = new Supplier();

            details->Supplier("SID123","medicines","0712193303","1200"); //creating suppier object

 //calling methods

   doctor->setDetails("123/4, Kandy Rd, Warakapola", 0772341867, "example@abc.com"); //set details to attributes
   doctor->displayDoctDetails(); //display normal user details

   fbc.displayFeedback(); //display feedback

   labReport->setDetails("35198", "lab198j"); //seting details to attributes
```

```
    labReport->displayDetails(); //displaying the normal user's details

    medicine->setDetails("12/4/2023", "oxygen", "146"); //setting details to attributes
    medicine->displayDetails(); //displaying the normal user's details

    p1.displayDetails();

    c1.displayChannel();

    guest->setDetails("Main street, Mathara", 0775894875, "abcdehjf@xyzh.com"); //set details to attributes
    guest->displayDetails(); //display normal user details

   //display admin deatils
   login->displayAdminDetails();

   //display supplier details
   details->diplaySupplierDetails();

    //delete dynamic variables
    delete doctor;
    delete labReport;
    delete medicine;
    delete guest;
    delete pay;
    delete Admin;
    delete c1;
    delete p1;
    delete fbc;
    delete supplier;
   return 0;
}
```