Topic : Online Teacher Trainer

Group no : MLB_WD_CSNE_13_12

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**
IT21170652	Samarawickrama A. W. D. M	0713946766
IT21376986	Sisirakara W. H. D	0713460586
IT21454578	Indunuwan K. M. I. G. U	0702097576

Contents

Introduction

This project deals with developing an e-commerce website for Online Teacher Trainer. It provides the user with a catalogue of different courses and subjects available for study. In order to facilitate the online study, lecture recordings are provided to the user.

This is a project with the objective to develop a basic website where a teacher is provided with an E-learning application and also to know about the technologies used to develop such an application. An Online Teacher Trainer is a virtual store on the internet where teachers can browse the catalogue and select subjects and courses of interest. The selected courses and subjects may be enrolled. At that time, more information will be needed to complete the transaction. Usually, the customer will be asked to fill in selected course information, personal information, and payment method such as credit card number

System Requirements

- The System should function 24 x 7 x 365.
- Guest users can overview the system, to use the system, they must register with the system by providing details such as Name, Address, NIC, Email, contact.
- Registered users called as learners where they can log into the system by entering the correct username and password.
- They can enrol the courses relevant to the subject they wish to study using the system.
- Then learner can enrol in the selected course by paying the relevant course price. Payment can be paid through debit/credit or through PayPal.
- The learner should participate for the exam during the course.
- Instructor should teach the courses and provides exams.
- System should generate a unique id for the learners after confirming. System needs to store all the registered user details and needs to display the courses details.
- Registered users can edit and manage the account.

Noun & Verb Analysis

- The System should function 24 x 7 x 365.
- Guest users can overview the system, to use the system, they must register with the system by providing details such as Name, Address, NIC, Email, contact.
- Registered users called as learners where they can log into the system by entering the correct username and password.
- They can select the course relevant to the subjects they wish to study using the system.
- Then learner can enrol in the selected course by paying the relevant course price. Payment can be paid through debit/credit or through PayPal.
- The learner should participate for the exam during the course.
- Instructor should teach the courses and makes exams.
- System should generate a unique id for the learners after confirming. System needs to store all the registered user details and needs to display the courses details.
- Registered users can edit and manage the account.

Identified Classes

- Gust User
- Leaner (Resisted user)
- Instructor
- Course
- Subject
- Exam
- Payment

Reasons for rejecting other nouns

- Redundant : Registered users, Learners
- Outside scope of system : System, Account, Administer
- Meta-language : They
- An attribute : Details (Name, Address, NIC, Email, Contact), Username, Password, Debit/credit card, PayPal
- An Event or an operation : _____

Methods

- Gust User - Register to the system by providing details View the system
- Leaner - Login to the system by entering details.

  Enrol the course relevant to the selected subjects.

  Pay the course fee.

  Participate for the exam
- Instructor - Teaches the course

  Provides exams
- Course - Generate course Duration

  Update the course Details

  Calculate course price
- Subject - Generate subject ID

  Update the subject details
- Exam - Generate exam ID

  Update exam details
- Payment - Generate pay ID

  Check payment details

  Confirm payments

CRC Cards for the Online Teacher Trainer system

Guest User	
Responsibility	**Collaborators**
Register to the system	
Allow to view the course details	Course

Leaner	
Responsibility	**Collaborators**
Login to the system	
Enrol the course	Course
Participate for the exam	Exam
Pay the course fee	Payment

Instructor	
Responsibility	**Collaborators**
Teaches the course	Course
Provides exams	Exams

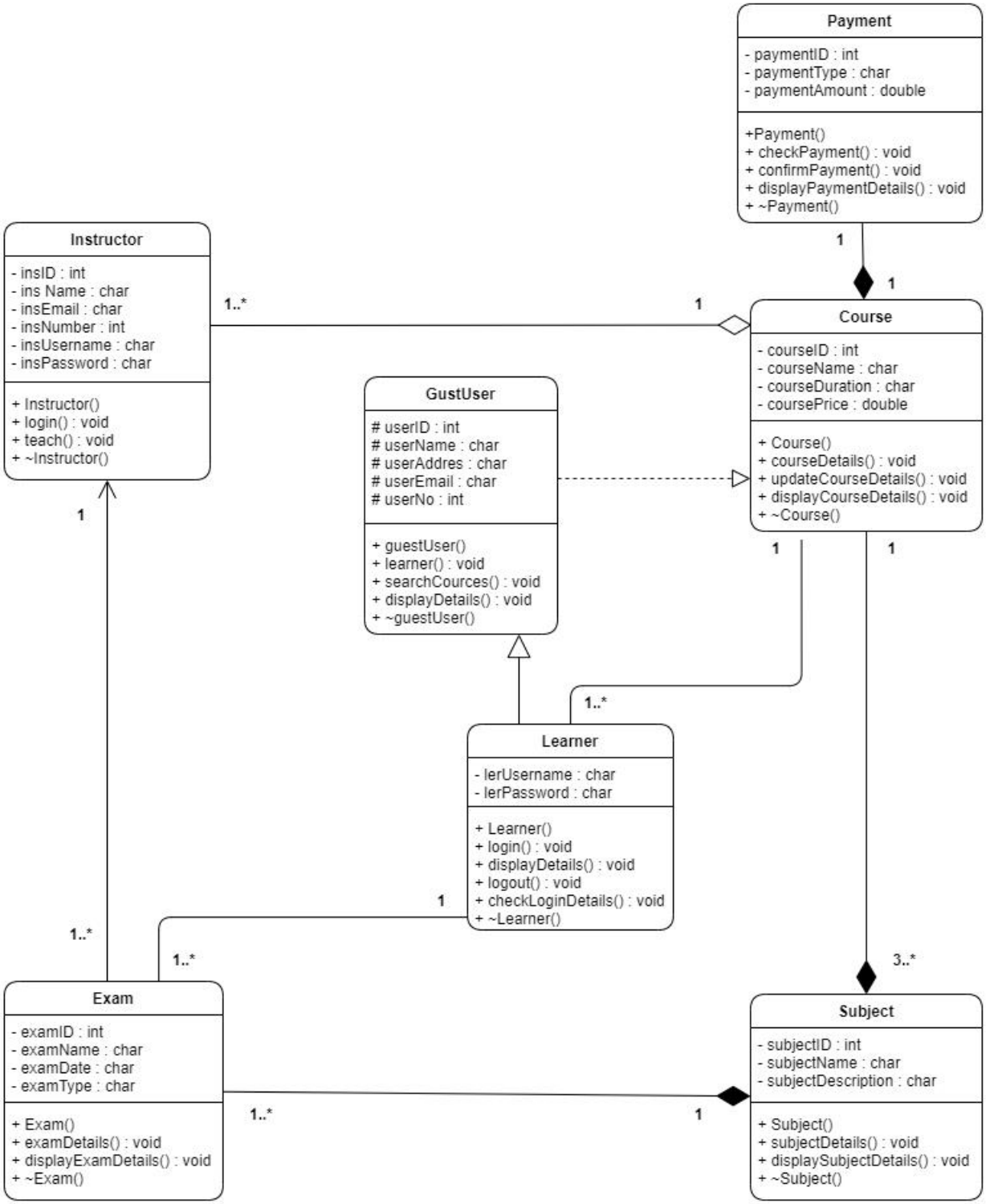
Course	
Responsibility	**Collaborators**
Generate course Duration	
Update the course Details	Subject
Calculate course price	Payment

Subject	
Responsibility	**Collaborators**
Generate subject ID	
Update the subject details	

Exam	
Responsibility	**Collaborators**
Generate exam ID	
Update exam details	Instructor

Payment	
Responsibility	**Collaborators**
Generate pay ID	
Check payment details	
Store the payment details	Course
Confirm payments	

Class Diagram (UML Notation)



Class Header Files

GuestUser.h

```
#include "Course.h"
class GuestUser
{
	protected:
		int userID;
		char userName[25];
		char userAddress[25];
		char userEmail[25];
		int userPhoneNo;

	public:
		GuestUser();
		GuestUser(int pID, const char pName[], const char pAddress[], const char
pEmail[], int pPhoneNo);
		void learner();
		void searchCources(Course *cou);
		virtual void displayDetails();
		~GuestUser();
};
```

Learner.h

```
#include "GuestUser.h"
#include "Exam.h"
#define SIZE 5
class Learner :public GuestUser
{
	private:
		char lerUsername[10];
		char lerPassword[10];
		int noOfExams;
		Exam *exams[SIZE];
		Learner *learn;
```

```
        public:
                Learner();
                Lerner(const char plerUsername[], const char plerPassword[], int pID, const
char pName[], const char pAddress[], const char pEmail[], int pPhoneNo, int noOfExams,
Learner *plearn);
                void addExam(Exam *e);
                void login();
                void displayDetails();
                void logout();
                void checkLoginDetails();
                ~Learner();
};
```

Instructor.h

```
class Instructor
{
        private:
                int insID;
                char insName[25];
                char insEmail[25];
                char insSubject[10];
                char insUsername[10];
                char insPassword[10];

        public:
                Instructor();
                Instructor(int pinsID, const char pinsName[], const char pinsEmail[], const
char pinsSubject[], const char pinsUsername[], const char pinsPassword[]);
                void displayInstructor();
                void login();
                void teach();
                ~Instructor();
};
```

Course.h

```
#include "Instructor.h"
#include "Learner.h"
#define SIZE 5

class Course
{
	private:
		int courseID;
		char courseName[10];
		char courseDuration[10];
		double coursePrice;
		Instructor *ins[SIZE];
		int noOfLearners;
		Learner *ler[SIZE];
		Payment *pay[SIZE];

	public:
		Course();
		Course(int pcourseID, const char pcourseName[], const char
pcourseDuration[], double pcoursePrice, int noOfLearners, int no1, int no2);
		void addInstructor(Instructor *ins1, Instructor *ins2, Instructor *ins3);
		void addLearners(Learner *lern);
		void displayPayment();
		void CourseDetails();
		void updateCourseDetails();
		void displayCourse();
		~Course();

};
```

Subject.h

```
#include "Course.h"
#define SIZE 5;

class Subject
{
	private:
		int subjectID;
		char subjectName[10];
		char subjectDescription[100];
		Course *course[SIZE];
		Exam *exam[SIZE];

	public:
		Subject();
		Subject(int psubjectID, const char psubjectName[], const char
psubjectDescription[], int cos1, int cos2, int exm1, int exm2);
		void displayCourse();
		void displayExam();
		void subjectDetails();
		void displaySubjectDetails();
		~Subject();
};
```

Exam.h

```
#include "Instructor.h"
#include "Learner.h"

class Exam
{
	private:
		int examID;
		char examName[10];
		char examDate[10];
		char examType[10];
		Instructor *inst;
		Learner *lern;
```

```
        public:
                Exam();
                Exam(int pexamID, const char pexamName[], const char pexamDate[10],
const char pexamType[], Instructor *i, Learner *plern);
                void examInstructor();
                void displayExamDetails();
                ~Exam();

};
```

Payment.h

```
class Payment
{
        private:
                int paymentID;
                char paymentType[10];
                double paymentAmount;

        public:
                Payment();
                Payment(int ppaymentID, const char ppaymentType[], double
ppaymentAmount);
                void checkPayment();
                void confirmPayment();
                void displayPaymentDetails();
                ~Payment();
};
```

Class Cpp Files

GuestUser.cpp

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

GuestUser::GuestUser()
{
	userID = 0;
	strcpy(userName, "");
	strcpy(userAddress, "");
	strcpy(userEmail, "");
	userPhoneNo = 0000000000;
}

GuestUser::GuestUser(int pID, const char pName[], const char pAddress[], const char pEmail[], int pPhoneNo)
{
	userID = pID;
	strcpy(userName, pName);
	strcpy(userAddress, pAddress);
	strcpy(userEmail, pEmail);
	userPhoneNo = pPhoneNo;
}

void GuestUser::searchCources(Course *c)
{

}

void GuestUser::learner()
{

}

void GuestUser::displayDetails()
{
	cout<<"User ID : "<<userID<<endl;
```

```
	cout<<"User Name : "<<userName<<endl;
	cout<<"User Address : "<<userAddress<<endl;
	cout<<"User Email : "<<userEmail<<endl;
	cout<<"User Phone Number : "<<userPhoneNo<<endl;
	cout<<"**********************************"<<endl;

}

GuestUser::~GuestUser()  //Destructor
{
	//cout<< "Destructor called" <<endl;
}
```

Leaner.cpp

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

Learner::Learner()
{
	strcpy(lerUsername, "");
	strcpy(lerPassword, "");
	noOfExams =0;
}

Learner::Lerner(const char plerUsername[], const char plerPassword[], int pID, const char
pName[], const char pAddress[], const char pEmail[], int pPhoneNo, int pnoOfExams,
Learner *plearn) : GuestUser(pID, pName[], pAddress[], pEmail[], pPhoneNo)
{
	strcpy(lerUsername, plerUsername);
	strcpy(lerPassword, plerPassword);
	noOfExams = pnoOfExams;
	learn = plearn;
	learn->addLearners();
}

void Lerner::addExam(Exam *e)
{
```

```
	if (noOfExams < SIZE)
	{
		exam[noOfExams] = e;
		noOfExams++;
	}
}

void Learner::displayDetails()
{

}

void Learner::login()
{

}

void Learner::logout()
{

}

void Learner::checkLoginDetails()
{
	//return 0;
}

Learner::~Learner()  //Destructor
{
	//cout<< "Lerner Delete" <<endl;
}
```

Instructor.cpp

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

Instructor::Instructor()
{
	insID = 0;
	strcpy(insName, "");
	strcpy(insEmail, "");
	strcpy(insSubject, "");
	strcpy(insUsername, "");
	strcpy(insPassword, "");
}

Instructor::Instructor(int pinsID, const char pinsName[], const char pinsEmail[], const char pinsSubject[], const char pinsUsername[], const char pinsPassword[])
{
	insID = pinsID;
	strcpy(insName, pinsName);
	strcpy(insEmail, pinsEmail);
	strcpy(insSubject, pinsSubject);
	strcpy(insUsername, pinsUsername);
	strcpy(insPassword, pinsPassword);
}

void Instructor::displayInstructor()
{
	cout<< "Instructor ID : " <<insID <<endl;
	cout<< "Instructor Name : " <<insName <<endl;
	cout<< "Instructor Email : " <<insEmail <<endl;
	cout<< "Instructor Subject : " <<insSubject <<endl;
	cout<< "*******************************"<<endl;
}

void Instructor::login()
{

}
```

```
void Instructor::teach()
{

}

Instructor::~Instructor()  //Destructor
{
	//cout<< "Instructor delete" <<endl;
}
```

Course.cpp

```
Course::Course()
{
	courseID = 0;
	strcpy(courseName, "");
	strcpy(courseDuration, "");
	coursePrice = 0.00;
	noOfLearners =0;
}

Course::Course(int pcourseID, const char pcourseName[], const char pcourseDuration[],
double pcoursePrice, int pnoOfLearners, int no1, int no2)
{
	courseID = pcourseID;
	strcpy(courseName, pcourseName);
	strcpy(courseDuration, pcourseDuration);
	coursePrice = pcoursePrice;
	noOfLearners = pnoOfLearners;
	pay[0] = new Payment(no1);
	pay[1] = new Payment(no2);
}

void Course::addLearners(Learner *lern)
{
	if (noOfLearners < SIZE)
	{
		ler[noOfLearners] = lern;
```

```
                noOfLearners++;
        }
}

void Course::displayPayment()
{
        for (int i=0; i<SIZE, i++)
        {
                pay[i]->displayPaymentDetails();
        }
}

void Course::addInstructor(Instructor *ins1, Instructor *ins2, Instructor *ins3)
{
        ins[0] = ins1;
        ins[1] = ins2;
        ins[2] = ins3;
}

void Course::CourseDetails()
{
        cout<<"Course ID : "<<courseID<<endl;
        cout<<"Course Name : "<<courseName<<endl;
        cout<<"Course Duration : "<<courseDuration<<endl;
        cout<<"Course Price : "<<coursePrice<<endl;
        cout<<"**********************************"<<endl;

}

void Course::displayCourse()
{
        for (int i=0; i<SIZE, i++)
        {
                ins[i]->displayInstructor();
        }

        for (int i=0; i<noOfLearners, i++)
        {
                ler[i]->displayDetails();
        }
}
```

```
void Course::updateCourseDetails()
{

}

Course::~Course() //Destructor
{
	//cout<< "Course Delete" <<endl;
	for (int i=0; i<SIZE, i++)
	{
		delete pay[i];
	}
}
```

Subject.cpp

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

Subject::Subject()
{
	subjectID = 0;
	strcpy(subjectName, "");
	strcpy(subjectDescription, "");
}

Subject::Subject(int psubjectID, const char psubjectName[], const char
psubjectDescription[], int cos1, int cos2, int exm1, int exm2)
{
	subjectID = psubjectID;
	strcpy(subjectName, psubjectName);
	strcpy(subjectDescription, psubjectDescription);
	course[0] = new Course(cos1);
	course[1] = new Course(cos2);
	exam[0] = new Exam(exm1);
	exam[1] = new Exam(exm2);
}
```

```
void Subject::displayCourse()
{
	for (int i=0; i<SIZE; i++)
	{
		course[i]->CourseDetails();
	}
}

void Subject::displayExam()
{
	for (int i=0; i<SIZE; i++)
	{
		exam[i]->displayExamDetails();
	}
}

void Subject::subjectDetails()
{

}

void Subject::displaySubjectDetails()
{

}

Subject::~Subject()  //Destructor
{
	//cout<< "Subject Delete" <<endl;
	for (int i=0; i<SIZE; i++)
	{
		delete course[i];
	}

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

Exam.cpp

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

Exam::Exam()
{
	examID = 0;
	strcpy(examName, "");
	strcpy(examDate, "");
	strcpy(examType, "");
}

Exam::Exam(int pexamID, const char pexamName[], const char pexamDate[10], const char
pexamType[], Instructor *i, Learner *plern)
{
	examID = pexamID;
	strcpy(examName, pexamName);
	strcpy(examDate, pexamDate);
	strcpy(examType, pexamType);
	inst = i;
	lern = plern;
	lern->addExam();
}

void Exam::examInstructor()
{
	cout<< "Exam ID : " <<examID <<endl;
	cout<< "Exam Name : " <<examName <<endl;
	inst-> displayInstructor();
	cout<< "*******************************"<<endl;
}

void Exam::displayExamDetails()
{
	cout<< "Exam ID : " <<examID <<endl;
	cout<< "Exam Name : " <<examName <<endl;
	cout<< "Exam Date : " <<examDate <<endl;
	cout<< "Exam Type : " <<examType <<endl;
	cout<< "*******************************"<<endl;
}
```

```
Exam::~Exam()  //Destructor
{
	//cout<< "Destructor called" <<endl;
}
```

Payment.cpp

```
#include "Payment.h"
#include <cstring>
Payment::Payment()
{
	paymentID= 0;
	strcpy(paymentType, "");
	paymentAmount = 0.00;
}
Payment::Payment(int ppaymentID, const char ppaymentType[], double ppaymentAmount)
{
	paymentID= ppaymentID;
	strcpy(paymentType, ppaymentType);
	paymentAmount = ppaymentAmount;
}
void Payment::checkPayment()
{

}
void Payment::confirmPayment()
{

}
void Payment::displayPaymentDetails()
{

}

Payment::~Payment()  //Destructor
{
	//cout<< "Destructor called" <<endl;
}
```

Main program

Main.cpp

```
#include "Learner.h"
#include "GuestUser.h"
#include "Instructor.h"
#include "Course.h"
#include "Exam.h"
#include "Payment.h"
#include "Subject.h"

#include <iostream>
using namespace std;

int main()
{
//---- Object creation ------

        Instructor *instruct = new Instructor();  // Dynamic Object - Instructor class
        Course *course = new Course();           // Dynamic Object - Course class
        GuestUser *lerner = new Learner();               // Dynamic Object - Learner class
        Exam *exam = new Exam();               // Dynamic Object - Exam class
        Payment *paym = new Payment();          // Dynamic Object - Payment class
        Subject *subj = new Subject();          // Dynamic Object - Subject class

//----Method Calling------

        instruct->login();
        instruct->displayInstructor();

        course->updateCourseDetails();
        course->CourseDetails();

        lerner->searchCources();
        lerner->displayDetails();

        exam->displayExamDetails();
        exam->examInstructor();

        subj->displaySubjectDetails();
```

```
        paym->checkPayment();
        paym->confirmPayment();
        paym->displayPaymentDetails();


//----Delete Dynamic objects------

        delete course;
        delete instruct;
        delete exam;
        delete lerner;
        delete exam;
        delete subj;
        delete paym;


        return 0;
}
```