Sri Lanka Institute of Information Technology

Project Title : Online Examination System For Employees

Group Number : MLB_04.01_12

Campus : Malabe

Submission Date : 20th May 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.

	Student ID	Student Name	Email	Contact Number
1	IT21310478	Saranasuriya N V	it21310478@my.sliit.lk	0765944049
2	IT21253676	Swarnajith T H M P	it21253676@my.sliit.lk	076 5350429
3	IT21262654	Jayawardhana B M W P	it21262654@my.sliit.lk	076 1267141
4	IT21251382	Wijesingha W M R	it21251382@my.sliit.lk	076 4291793
5	IT21255960	Upathissa K W G M B	it21255960@my.sliit.lk	0718934434

Description

Online Examination System is a technology-driven way to simplify examination activities like defining exam patterns with question banks, defining exam timer, objective/ subjective question sections, conducting exams using the computer or mobile devices in a paperless manner.

The platform is used to conduct assessments, aptitude tests, psychometric tests and personality tests, entrance exams, hiring assessment tests. You can conduct an assessment using a computer, mobile, tablet devices. There is a facility to conduct offline assessments which can be synced with the main system after the assessment is completed.

Well specialized team of system administration and an attentive academic staff included in this system. Employees are let to login to the system through a web interface. Registered users are privileged to enroll into the exams and courses. A Single user can enroll into a single course at a time. While exams which corresponds to the course selected by the user are definitely included with the course. As an additional option certificates are also provided which can be a part of the course.

Users have the access to the proceed the registration, exam enrolments, course enrolments and rest of the payment by an online banking system linked with this system.

We provide secure browser technology, which prevents customers from opening any other windows while taking an online exam.

Requirements

- The system should work every day 24/7.
- Guest users can interact with the system to use the system, they must register with the system by providing personal details such as Name, Address, NIC, Email, contact.
- System admin checks the validity of the user registration details.
- Registered users are of two types called employee and marking lecturer where they can log into the system by entering the correct username and password
- After log in to the system, system should generate a unique id for the employee and marking lecturer.
- Registered user can update user details and view current details.
- An employee user should be able to view course types, enroll for a course, search exam. request to do the exam.
- A Marking lecturer user should be able to mark the papers and submit marks.
- Staff maintain a documentation which includes the details about User enrolments.
- Registered employee must do a payment for their course fee and exam fee
- They should make a payment through online by using PayPal, credit card or debit card.
- System should check the payment details and confirm the payment details and a transaction report should be sent to the registered employee and ‘Pay ID’ is generated to the ‘EMP ID’ of Employees and ‘ML ID’ of Marking Lecturer
- System admin can generate reports of the Exam details, Course details and payment details.
- Employee users and Marking lecturer users both can rate the system services and send feedback to the system and suggestions if unsatisfied.

Noun/Verb Analysis

- The system should work every day 24/7.
- Guest user can interact with the system to use the system, they must register with the system by providing personal details such as Name, Address, NIC, Email, contact.
- System admin checks the validity of the user registration details.
- Registered users are of two types called employee and marking lecturer where they can log into the system by entering the correct username and password
- After log in to the system, system should generate a unique id for the employee and marking lecturer.
- Registered user can update user details and view current details.
- An employee user should be able to view course types, enroll for a course, search exam. request to do the exam.
- A Marking lecturer user should be able to mark the papers and submit marks.
- Staff maintain a documentation which includes the details about User enrolments.
- Registered employee must do a payment for their course fee and exam fee
- They should make a payment through online by using PayPal, credit card or debit card.
- System should check the payment details and confirm the payment details and a transaction report should be sent to the registered employee and 'Pay ID' is generated to the 'EMP ID' of Employees and 'ML ID' of Marking Lecturers
- System admin can generate reports of the Exam details, Course details and payment details.
- Employee users and Marking lecturer users both can rate the system services and send feedback to the system and suggestions if unsatisfied.

Identified classes

- Guest User
- Registered user
- Employee
- Marking lecture
- Enrolment
- Exam
- Course
- Staff
- Payment
- Report

Reasons for rejecting other nouns

- **Redundant:** Guest user, employee, marking lecturer, Staff, Registered users
- **An Event or an operation:**
- **Outside scope of system:** System**,** System Admin, reports, databases, feedback
- **Meta-language:** They, Details (Exam, Course, payment)
- **An attribute:** Username (of Registered User), Password (of Registered User), Details (Name, Address, NIC, Email, Contact), Username, password, PayPal (of Payment), credit card (of Payment), debit card (of Payment), transaction report, unique id, course types, exam, papers, marks, User enrolments, course fee, exam fee, Pay ID, ML ID, EMP ID, payment type,

CRC Cards

Guest User	
Responsibilities	**collaborations**
Register to the system	
view the enrolling details	Enrolment

Registered User	
Responsibilities	**collaborations**
log in to the system	
view the enrolling details	Enrolment
Add and update user details	
Can select a exam depending on the course selected	Course, Exam

Employee	
Responsibilities	**collaborations**
Log in to the system	Registered User
Enroll for a Exam	Enrolment

Marking lecture	
Responsibilities	**collaborations**
Log in to the system	Registered User
Enroll for course to mark papers	Enrolment

Enrolment	
Responsibilities	**collaborations**
Display the details of the enrolment regarding the courses and exams they have registered.	Employee
Get details regarding enrolment date and time	Marking lecturer

Exam	
Responsibilities	**collaborations**
Search exam	
Enroll for a Exam	Enrolment
Calculate the Exam fee	

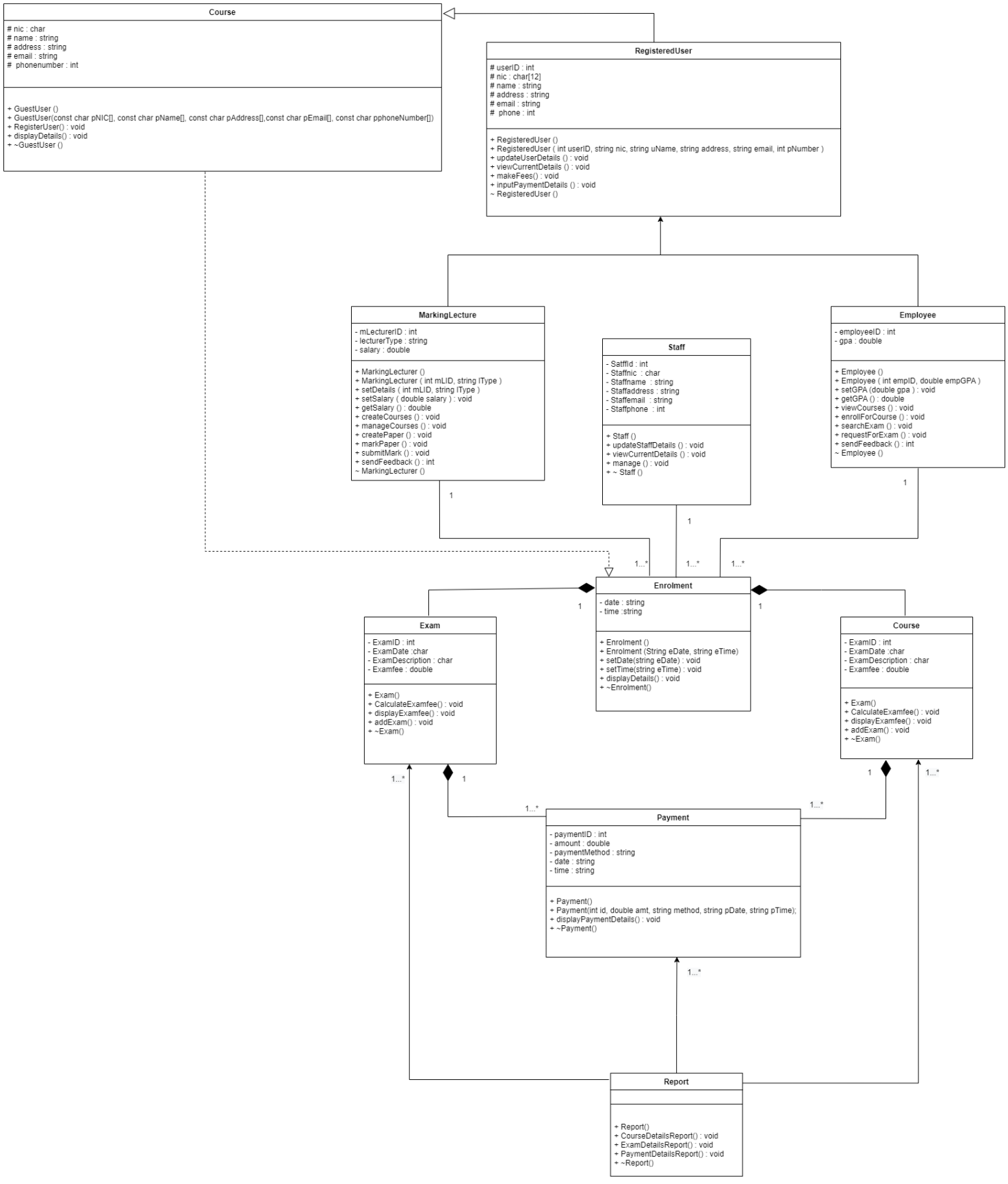
Course	
Responsibilities	**collaborations**
view course types	
Enroll for a course	Enrolment
Calculate the course price	

Staff	
Responsibilities	**collaborations**
Login to system	
Maintain a documentation which includes the details about User enrolments.	Enrolment

Payment	
Responsibilities	**collaborations**
Make a new payment	
Check payment details	Employee, Marking lecturer
Confirm payment details	
Generate Pay ID	Exam, Course

Report	
Responsibilities	**collaborations**
Generate Exam details	Exam
Generate Course details	Course
Generate Payment details	Payment

Class Diagram



Codings

GuestUser.h

```
class GuestUser
{
protected:
	char NIC[12];
	char Name[30];
	char Address[40];
	char Email[30];
	char phoneNumber[10];

public:
	GuestUser();
	GuestUser(const char pNIC[], const char pName[], const char pAddress[],
const char pEmail[], const char pphoneNumber[]);
	void registerUser();
	void displayDetails();
	~GuestUser();
};
```

GuestUser.cpp

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

GuestUser::GuestUser()
{
	strcpy(NIC, "");
	strcpy(Name, "");
	strcpy(Address, "");
	strcpy(Email, "");
	strcpy(phoneNumber, "");
}

GuestUser::GuestUser(const char pNIC[], const char pName[], const char pAddress[],
const char pEmail[], const char pphoneNumber[])
{
	strcpy(NIC, pNIC);
	strcpy(Name, pName);
	strcpy(Address, pAddress);
	strcpy(Email, pEmail);
	strcpy(phoneNumber, pphoneNumber);
}

void GuestUser::registerUser()
{

}

void GuestUser::displayDetails()
{

}

GuestUser::~GuestUser()
```

```
{
	//Destructor
}
```

RegisteredUser.h

```
class RegisteredUser {
protected:
    int userId;
    char nic[12];
    char name[20];
    char address[100];
    char email[30];
    char phone[10];

public:
    RegisteredUser();
    RegisteredUser(int pUserId, char pNic[12], char pName[20], char pAddress[100],
char pEmail[30], int pPhone[10]);
    void updateUserDetails();
    void viewCurrentDetails();
    void makeFees();
    void inputPaymentDetails();
    ~RegisteredUser();
};
```

RegisteredUser.cpp

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

RegisteredUser::RegisteredUser() {
    cout << "Default Constructor called" << endl;
}

RegisteredUser::RegisteredUser(int pUserId, char pNic[12], char pName[20], char
pAddress[100], char pEmail[30], int pPhone[10])
{
    this->userId = pUserId;
    this->nic[12] = pNic[12];
    this->name[20] = pName[20];
    this->address[100] = pAddress[20];
    this->email[30] = pEmail[20];
    this->phone[10] = pPhone[10];
}
RegisteredUser::~RegisteredUser()
{
    cout << "Destructor called" << endl;
}
```

Employee.h

```
#include "RegisteredUser.h"
    class Employee : public RegisteredUser {

    protected:
        int employeeID;
        double GPA;

    public:
        Employee();
        Employee(int pemployeeID, double pGPA);
        int getEmployeeID();
        void setGPA(double pGPA);
        double getGPA();
        void enrollForCourse();
        void searchExam();
        void requestForExam();
        void sendFeedback();
        ~Employee();
};
```

Employee.cpp

```
#include <iostream>
#include <string>

#include "Enrolment.h"
#define SIZE1 2
#define SIZE2 2

using namespace std;

Enrolment::Enrolment() {
 cout << "Default constructor of Enrollment called" << endl;
}
Enrolment::Enrolment(string eDate, string eTime) {
 date = eDate;
 time = eTime;
}

Enrolment(int ex1, int ex2, int Course1, int course2,Employee* pEmployee,
MarkingLecturer* pMarkingLecturer,Staff* pStaff);
{
  Exam[0] = new Exam(ex1);
  Exam[1] = new Exam(ex2);

  Course[0] = new Course(Course1);
  Course[1] = new Course(Course2);

  Employee = pEmployee;
  MarkingLecturer = pMarkingLecturer;
  Staff = pStaff;
}

Enrolment::~Enrolment()
```

```
{
  //Destructor
  for (int i = 0; i < SIZE1; i++)
    {
      delete Course[i];
    }
  for (int i = 0; i < SIZE2; i++)
    {
      delete Exam[i];
    }
}
```

MarkingLecture.h

```
#include "RegisteredUser.h"
class MarkingLecture : public RegisteredUser {

private:
    int markingLectureId;
    char lecturerType[10];
    double salary;
public:
    MarkingLecture();
    void setMarkingLectureId(int pMarkingLectureId);
    void setLecturerType(char pLecturerType[10]);
    void setSalary (double pSalary);
    double getSalary();
    void createCourses();
    void manageCourses();
    void createPaper();
    void markPaper();
    void submitMark();
    ~MarkingLecture();
};
```

MarkingLecture.cpp

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

MarkingLecture::MarkingLecture() {
    cout << "Default Constructor called" << endl;
}

void MarkingLecture::setMarkingLectureId(int pMarkingLectureId)
{
    this->markingLectureId = pMarkingLectureId;
}

void MarkingLecture::setLecturerType(char pLecturerType[10])
{
    this->lecturerType[10] = pLecturerType[10];
}

void MarkingLecture::setSalary(double pSalary)
```

```
{
    this->salary = pSalary;
}

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

Enrolment.h

```
#include <string>
#include "Course.h"
#include "Exam.h"
#include "Employee.h"
#include "MarkingLecturer.h"
#include "Staff.h"

#define SIZE1 5
#define SIZE2 5

class Enrolment {
  private:
    string date;
    string time;

    Course* Course[SIZE1];
    Exam* Exam[SIZE2];
    Employee* Employee;
    MarkingLecturer* MarkingLecturer;
    Staff* Staff;

  public:
     Enrolment();
     Enrolment(string eDate, string eTime);
     void setDate(string eDate);
     void setTime(string eTime);
     void displayDetails();
     Enrolment(int ex1, int ex22, int Course1, int course2,Employee* pEmployee,
MarkingLecturer* pMarkingLecturer,Staff* pstaff);
     ~Enrolment();

};
```

Enrolment.cpp

```
#include <iostream>
#include <string>

#include "Enrolment.h"
#define SIZE1 2
#define SIZE2 2

using namespace std;

Enrolment::Enrolment() {
 cout << "Default constructor of Enrollment called" << endl;
}
```

```
Enrolment::Enrolment(string eDate, string eTime) {
 date = eDate;
 time = eTime;
}

Enrolment(int ex1, int ex2, int Course1, int course2,Employee* pEmployee,
MarkingLecturer* pMarkingLecturer,Staff* pStaff);
{
  Exam[0] = new Exam(ex1);
  Exam[1] = new Exam(ex2);

  Course[0] = new Course(Course1);
  Course[1] = new Course(Course2);

  Employee = pEmployee;
  MarkingLecturer = pMarkingLecturer;
  Staff = pStaff;
}

Enrolment::~Enrolment()
{
  //Destructor
  for (int i = 0; i < SIZE1; i++)
    {
      delete Course[i];
    }
  for (int i = 0; i < SIZE2; i++)
    {
      delete Exam[i];
    }
}
```

Exam.h

```
#include "Payment.h"
#define SIZE 5
class Exam {
    private:
    int ExamID;
    char ExamDate[20];
    char ExamDescription[50];
    double Examfee;
    int count = 0;

    Payment* payment[SIZE];

  public:
    Exam();
    Exam(int pExamID, const char pExamdate[], const char pExamdescription[], double
pExamfee, int pay1, int pay2);
    void calculateExamfee(int id, const char pType[], double
    pAmt);
    void displayExamfee();
    void addExam();
    ~Exam();

};
```

Exam.cpp

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

Exam::Exam()
{
  ExamID = 0;
  strcpy(ExamDate, "");
  strcpy(ExamDescription, "");
  Examfee = 0;
}

Exam(int pExamID, const char pExamdate[], const char pExamdescription[], double
pExamfee, int pay1, int pay2);
{
  Examfee = pExamfee;
  strcpy(ExamDate, pExamdate);
  strcpy(ExamDescription, pExamdescription);
  ExamID = pExamID;
}

void Exam::calculateExamfee(int id, const char pType[], double
pAmt)
{
  if (count < SIZE)
  {
    payment[count] = new Payment(id, pType, pAmt);
    count++;
  }
}

void Exam::displayExamfee()
{
}

void Exam::addExam()
{
}

Exam::~Exam()
{
  //Destructor
  for (int i = 0; i < SIZE; i++)
  {
    delete payment[i];
  }
}
```

Course.h

```
#include "Payment.h"
#define SIZE 5

  class Course {
    private:
    char CourseID[10];
    char CourseDate[20];
    char CourseDescription[50];
    double CoursePrice;
    int count = 0;

    Payment* payment[SIZE];

  public:
    Course();
    Course(const char pCourseID[], const char pCourseDate[],const
    char pCourseDescription[],double pCoursePrice,int pay1, int pay2);
    void calculateCoursePrice(int id, char pType[], double pAmt);
    void displayCoursePrice();
    void addCourse();
    ~Course();
};
```

Course.cpp

```
#include "Payment.h"
#include "Course.h"
#include<cstring>

Course::Course()
{
  strcpy(CourseID, "");
  strcpy(CourseDate, "");
  strcpy(CourseDescription, "");
  CoursePrice = 0;
}

Course::Course(const char pCourseID[],const char pCourseDate[], const
char pCourseDescription[], double pCoursePrice, int pay1, int pay2)
{
  strcpy(CourseID, pCourseID);
  strcpy(CourseDate, pCourseDate);
  strcpy(CourseDescription, pCourseDescription);
  CoursePrice = 0;
}

void Course::calculateCoursePrice(int id, char pType[], double pAmt)
{
  if (count < SIZE)
  {
    payment[count] = new Payment(id, pType, pAmt);
    count++;
  }
}
```

```
void Course::displayCoursePrice()
{
}

void Course::addCourse()
{
}

Course::~Course()
{
  //Destructor
  for (int i = 0; i < SIZE; i++)
  {
    delete payment[i];
  }
}
```

Staff.h

```
#include "Enrolment.h"
#define SIZE 5

class Staff{
  protected:
    int SatffId;
    char Staffnic[12];
    char Staffname[20];
    char Staffaddress[100];
    char Staffemail[30];
    char Staffphone[10];

    Enrolment* apt[SIZE];

  public:
    Staff();
    Staff(int pStaffId, char pStaffNic[12], char pStaffName[20], char
pStaffAddress[100], char pStaffEmail[30], int pStaffPhone[10]);
    void updateStaffDetails();
    void viewCurrentDetails();
    void manage( Enrolment* papt);
    ~Staff();

};
```

Staff.cpp

```
#include "Enrolment.h"
#include "Staff.h"

#include <iostream>
using namespace std;

Staff::Staff() {
    cout << "Default Constructor called" << endl;
}
```

```
Staff::Staff(int pStaffId, char pStaffNic[12], char pStaffName[20], char
pStaffAddress[100], char pStaffEmail[30], int pStaffPhone[10])
{
    this->SatffId = pStaffId;
    this->Staffnic[12] = pStaffNic[12];
    this->Staffname[20] = pStaffName[20];
    this->Staffaddress[100] = pStaffAddress[20];
    this->Staffemail[30] = pStaffEmail[20];
    this->Staffphone[10] = pStaffPhone[10];
}

void Staff::manage(Enrolment* papt)
{
}

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

Payment.h

```
#include <string>
#include "Exam.h"
#include "Course.h"

class Payment {
  private:
     int paymentID;
     double amount;
     string paymentMethod;
     string date;
     string time;

     Exam* Exam;
     Course* Course;

  public:
     Payment();
     Payment(int id, double amt, string method, string pDate, string pTime);
     void displayPaymentDetails();
     ~Payment();
};
```

Payment.cpp

```
#include <iostream>
#include <string>
#include "Payment.h"
#include "Exam.h"
#include "Course.h"

using namespace std;

Payment::Payment() {
   cout << "Default Constructor of Payment called" << endl;
}
```

```
Payment::Payment(int id, double amt, string method, string pDate, string pTime) {
   paymentID = id;
   amount = amt;
   paymentMethod = method;
   date = pDate;
   time = pTime;
}

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

Report.h

```
#include"Exam.h"
#include"Course.h"
#include"Payment.h"
#define SIZE1 5
#define SIZE2 5
#define SIZE3 5

class Report
{
  private:
    Course* Course[SIZE1];
    Exam* Exam[SIZE2];
    Payment* pay[SIZE3];
  public:
    Report();
    Report(Course* pbbok[], Exam* psell[], Payment* ppay[]);
    void CourseDetailsReport();
    void ExamDetailsReport();
    void PaymentDetailsReport();
    ~Report();
};
```

Report.cpp

```
#include "Report.h"
Report::Report()
{
  for (int i = 0; i < SIZE1; i++)
  {
    Course[i] = 0;
  }
  for (int j = 0; j < SIZE2; j++)
  {
    Exam[j] = 0;
  }
  for (int k = 0; k < SIZE3; k++)
  {
    pay[k] = 0;
  }
}

Report::Report(Course* pCourse[], Exam* pExam[], Payment* ppay[])
{
  for (int i = 0; i < SIZE1; i++)
```

```
  {
    Course[i] = pCourse[i];
  }
  for (int j = 0; j < SIZE2; j++)
  {
    Exam[j] = pExam[j];
  }
  for (int k = 0; k < SIZE3; k++)
  {
    pay[k] = ppay[k];
  }
}

void Report::CourseDetailsReport()
{
}

void Report::ExamDetailsReport()
{
}

void Report::paymentDetailsReport()
{
}

Report::~Report()
{
    //Destructor
    for (int i = 0; i < SIZE1; i++)
    {
        delete Course[i] ;
    }
    for (int j = 0; j < SIZE2; j++)
    {
        delete Exam[j] ;
    }
    for (int k = 0; k < SIZE3; k++)
    {
        delete pay[k] ;
    }
}
```

Main.h

```
#include "GuestUser.h"
#include "GuestUser.cpp"
#include "RegisteredUser.h"
#include "RegisteredUser.cpp"
#include "Employee.h"
#include "GuestUser.cpp"
#include "MarkingLecture.h"
#include "MarkingLecture.cpp"
#include "Course.h"
#include "Course.cpp"
#include "Enrolment.h"
#include "Enrolment.cpp"
#include "Exam.h"
#include "Exam.cpp"
#include "Staff.h"
#include "Staff.cpp"
#include "Payment.h"
#include "Payment.cpp"
#include "Report.h"
#include "Report.cpp"


#include <iostream>

using namespace std;
int main()
{
  //Objects Creation
       GuestUser* guser = new GuestUser();
       //Methods Calling
       guser->registerUser();
       guser->displayDetails();

  Employee* emp = new Employee();
       emp->enrollForCourse();
       emp->searchExam();
       emp->requestForExam();
       emp->sendFeedback();


  RegisteredUser * rgUser1 = new RegisteredUser(100, "200174982345", "Amal",
"Colombo, Srilanka",  "amal@gmail.com", "0718723645");
    RegisteredUser * rgUser2 = new RegisteredUser(200, "200047924345", "Suneth",
"Kandy, Srilanka",  "suneth@gmail.com", "07692836445");

  MarkingLecture * mlec1 = new MarkingLecture();
  MarkingLecture * mlec2 = new MarkingLecture();


  Course* Course = new Course(); // Object - Course class
  Course->addCourse();
  Course->displayCoursePrice();

  Enrolment* enr1 = new Enrolment("11 / 11 / 2000", "11.11");
  Enrolment* enr2 = new Enrolment("11 / 09 / 2000", "09.11");
  Enrolment* enr3 = new Enrolment(); // Object - Enrolment class
```

```
  Exam* Exam = new Exam(); // Object - Exam class
  Exam->addExam();
  Exam->displayExamfee();

  Staff * Staff1 = new Staff(111, "123456789101", "Pawan", "Horana, Srilanka",
"pawan@gmail.com", "0775520022");
  Staff * Staff2 = new Staff(112, "198765432102", "Dinusha", "Kalaniya, Srilanka",
"dinusha@gmail.com", "0777283834");

  Payment* payment1 = new Payment(100, 1500.0, "PAYPAL", "21.03.2022", "10.34");
  Payment* payment2 = new Payment(200, 1300.0, "Online VISA","01.06.2022","12.28");

  Report* rpt = new Report(); // Object - Report class
  rpt->CourseDetailsReport();
  rpt->ExamDetailsReport();
  rpt->PaymentDetailsReport();


  //Delete Dynamic Objects
  delete guser;

  delete emp;

  delete rgUser1;
  delete rgUser2;

  delete mlec1;
  delete mlec2;

  delete Course;

  delete enr1;
  delete enr2;
  delete enr3;

  delete Exam;

  delete Staff1;
  delete Staff2;

  delete payment1;
  delete payment2;

  delete rpt;

  return 0;
}
```

Individual Contribution

	Student ID	Student Name	Class Diagram	Codings
1	IT21310478	Saranasuriya N V	• Enrollment • Staff	• Enrollment.h • Enrollment.cpp • Staff.h • Staff.cpp
2	IT21253676	Swarnajith T H M P	• Course • Exam	• Course.h • Course.cpp • Exam.h • Exam.cpp
3	IT21262654	Jayawardhana B M W P	• Registered User • Marking Lecture	• RegisteredUser.h • RegisteredUser.cpp • MarkingLecture.h • MarkingLecture.cpp
4	IT21251382	Wijesingha W M R	• Payment • Report	• Payment.h • Payment.cpp • Report.h • Report.cpp
5	IT21255960	Upathissa K W G M B	• Guest User • Employee	• GuestUser.h • GuestUser.cpp • Employee.h • Employee.cpp