Topic : Online Teacher Trainer

Group no : MLB_01.02_04

Campus : Malabe

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**
IT21196942	Hettiarachchige S.I	070 2971773
IT21197864	Kahatapitiya.K.K.O.Y	076 9311540
IT21194276	Muhtunayaka.J.P.V	0765658411
IT21195020	Rabukpotha.R.R.M.L.G	0789798437
IT21195334	Athukorala.D.T.A	0776864451

Functional Requirements of the system:

Registered customer (Student) :-

• Should be able to log in to the system

• Browse on Course Page

• Can enroll to Courses

• Make payment from credit or debit cards

• Check the Schedule

• Able to change account settings

• Attend to training sessions

• Contact lecturers or staff member

• Ask Questions (Support service)

• Explore Resources (eBooks, lecture recordings etc.)

• Can view the offers page

• Can access social media contacts

Non- registered user (Guest) :-

• Should be able to view home page

• Should be able to view course details in course page

• Contact staff member (Contact us)

• Ask Questions (Support service)

• Can access social media links

• Can be register into the system (sign up)

Instructor (Lecturer) :-

• Able to Log in to the system

• Explore resources (eBooks)

• Check the lecture schedules

• Can view student information

• Conduct lectures

• Should be able to change account settings

• Send notifications to students

• Communicate with students

• Contact stuff member (administrator)

Administrator :-

• Able to view students and lecturers details

• Assign students to specific courses

• Assign lecturers to specific courses

• Update lecture details

• Answer to questions (support services)

• Upload resources (lecture recordings ,e books)

• Upload FAQs

• Send notifications to students and lecturers

• Upload exam questions and marks

• Analyze student’s performance

• Analyze student’s participation for courses

• Maintain the data in the system

Technician (back end developer) : -

• Analyze system performance

• Back up information

• Update course details

• Update content of the system

• Adding new features and content into the system

• Maintain the security of the system

• Communicate with staff

Identified Classes:

1.Register customer(student)

2.Exam

3.Course

4.Technician.

5.Resources

6.Non-registered

7.Administrator

8.System

9.Schedule

10.Instuctor

CRC cards

Course

Responsibility	**Collaborators**
Store & update course details	
Get Lecturer details	Lecturer
Get resources details	Resources
Get student details	Student
Get offer details	Administrator
Get schedules	Schedules

Technician

Responsibility	**Collaborators**
Store & update Technician details	
Analyse system performance	System
Back up information	System
Update course details	Courses
Update content of the system	System

Adding new features and content into the system	System
Maintain the security of the system	System
Communicate with the staff and organization	

Administrator

Responsibility	Collaborator
Get students and lecturers details	Student/Instructor
Set students to specific courses	Student/System
Set lecturers to specific courses	Instructor/System
Update lecture details	Schedule
Answer questions (support services)	System
Upload FAQs	System
Upload resources	Resources
Get exam questions and marks	Exam
Maintain the data in the system	System

System

Responsibility	Collaborator
Send notifications to students and lecturers	Student/Lecturer
Analyze student's performance	Student
Analyze student's participation for courses	Student/Course
Assign students to courses	Course/Administrator
Assign lecturers to courses	Course/Administrator
Store and update system information	
Get questions	Administrator

Register customer(student)	
Responsibilities	**collaborations**
Access resources	Resources table

Browse on course page	System
enroll to Courses	Course table
Attend to training sessions	
Check the Schedule	Schedule
Ask Questions (support service)	Admin
Store and update student details	System

Exam	
Responsibilities	**collaborations**
Set Exam questions	System
Set exam marks	System
Sent exam questions	
Reseau exam answers	student

CRC cards

Schedule

Responsibility	Collaborators
Store and update schedule details	
Update the schedule	
Get session schedule	
Set batch wise schedule	
Get lecturer schedule	Lecturer
Set lecture hall/labs schedule	
Display schedule	

Instructor

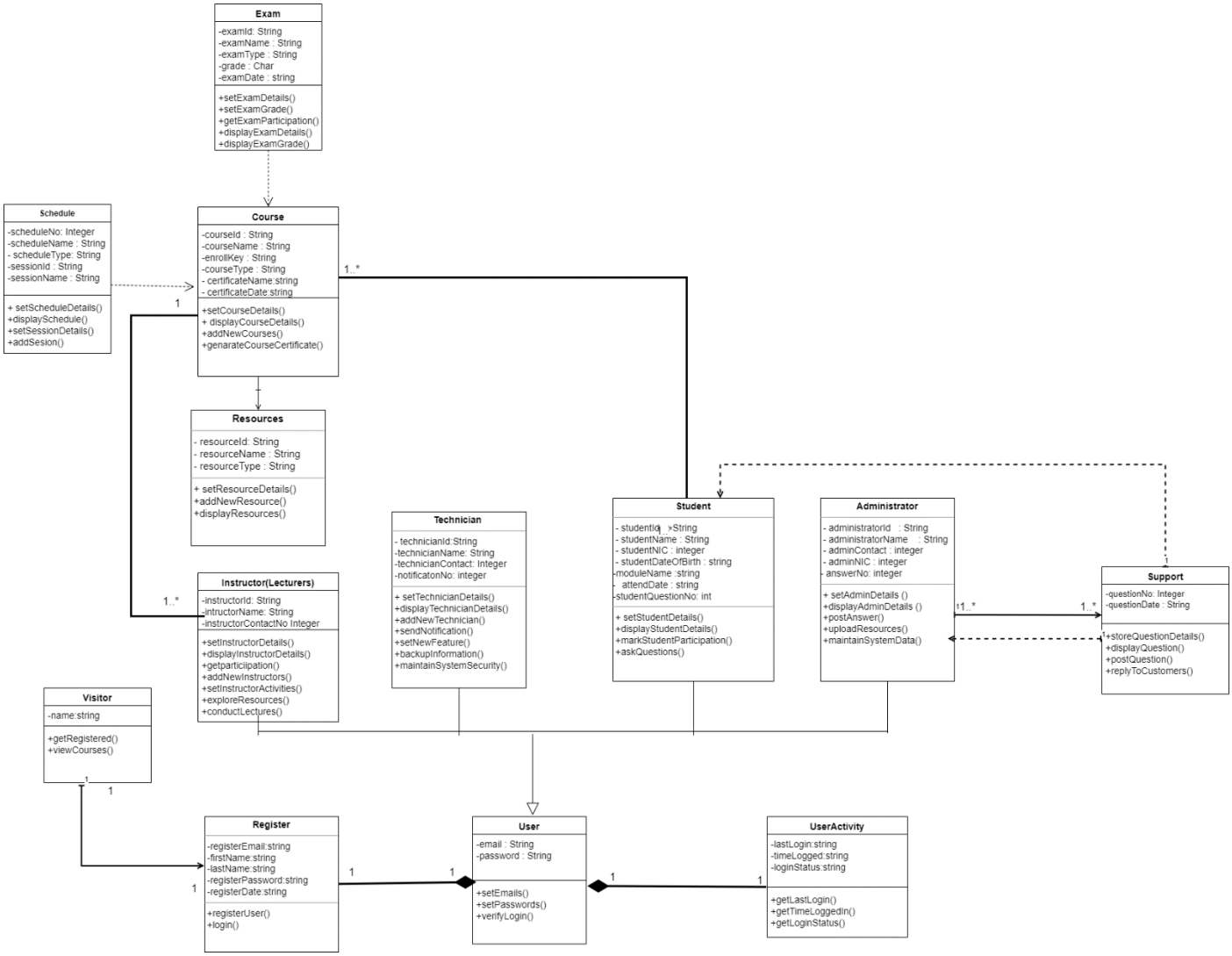
Responsibility	Collaborators
Explore resources	Resources
Check lecture schedules	Schedule
View student information	Student
Send notification to students	Student/System
Contact administrator	Administrator
Communicate with students	Student
Conduct lectures	System

CRC Cards

Resources

Responsibilities	Collaborations
Store and update resource details	
Manage resources	
Add new resources	
Allocate resources to courses	Course
Get resources	

Class Diagram



Code

Header file.

```
#include<iostream>

using std::string;


//-----Implementing Classes-----


//class Course

class Course{

	protected:

		string courseId;

		string courseName;

		string enrollKey;

		string courseType;

		string certificateNo;

		string certificateName;

		string certificateDate;

		Instructor *instructor;//BI-DIRECTIONAL ASSOCIATION

		Student *student;//BI-DIRECTIONAL ASSOCIATION

		Resources *resources;//UNI-DIRECTIONAL


	public:

		Course();

		Course(string id, string name, sring key, string type,Instructor *i,Student *s,Resources *r);

		void displayCourseDetails();

		void addNewCourses();

		void genarateCourseCertificate(string id, string name, string type, string c_no, string c_name, string c_date);
```

```
                ~Course();
};

//class schedule
class Schedule{
  private:
        int scheduleNo;
        string scheduleName;
        string scheduleType;
        int sessionId;
        string sessionName;

   public:
        Schedule();
        Schedule(int s_no, string s_name, string s_type);
        void displaySchedule();
        void setSessionDetails(int s_id, string se_name,Course *c);
        void addSession();
        ~Schedule();
};

//class Resources;
class Resources{
   private:
     string resourceId;
     string resourceName;
     string resourceType;
```

```
    public:
        Resources();
        Resources(string r_id, string r_name, string r_type);
        void addNewResource();
        void displayResources();
        ~Resources();
};

//class Student ;
class Student: public User{
  private:
        string studentId;
        string studentName;
        int studentNIC;
        string studentDateOfBirth;
        string moduleName;
        string attendDate;
        int studentQuestionNo;
                Course *course;//BI-DIRECTIONAL ASSOCIATION

  public:
                Student();
        Student(string s_id, string s_name, int s_nic, string s_dob,Course *c,string em, string pswd,Register *rg,UserActivity *ua:):User(em,pswd,*rg,*ua);
      void displayStudentDetails();
      void markStudentParticipation(string s_id, string s_name, int s_nic,string m_name, string a_date);
      void askQuestions(string s_id, string s_name, string stq_no,Support *sp);
      ~Student();
```

```
};


//class Technician;
class Technician: public User{
   private:
          string technicianId;
      string technicianName;
      int technicianContact;
      int notificatonNo;
      string newFeature;


   public:
          Technician();
      Technician(string t_id, string t_name, int t_contact,string em, string pswd,Register
*rg,UserActivity *ua:):User(em,pswd,*rg,*ua);
      void displayTechicianDetails();
      void addNewTechnician();
      void sendNotification(string t_id, string t_name,int n_no);
      void setNewFeature();
      ~Technician();
};


//class Instructor;
class Instructor: public User{
   private:
      string instructorId;
      string intructorName;
      int instructorContactNo;
                  Course *course;//BI-DIRECTIONAL ASSOCIATION
```

```
    public:
        Instructor();
    Instructor(string i_id, string i_name, int i_contactno,Course *c,string em, string pswd,Register
*rg,UserActivity *ua):User(em,pswd,*rg,*ua);
    void displayInstructorDetails();a
    void getparticiipation();
    void addNewInstructors();
    void setInstructorActivities();
    ~Instructor();
};

//class Administrator;
class Administrator : public User{
    private:
    string administratorId;
    string administratorName;
    int adminContact;
    int adminNIC;
    Support *support;//UNI-DIRECTIONAL ASSOCIATION

    public:
        Administrator();
    Administrator(string a_id,string a_name,int a_contact, int a_nic,string em, string pswd,Register
*rg,UserActivity *ua:):User(em,pswd,*rg,*ua);
    void displayAdminDetails()
            void postAnswers(string a_id,Support *sp);
            void uploadResources();
    void maintainSystemData();
```

```
        ~Administrator();
};

//class Support;
class Support{
    private:
        int questionNo;
        string questionDate;

    public:
            Support();
        Support(int q_no, string q_date,Administrator *a);
        void displayQuestion();
        void postQuestion();
        void replyToCustomers();
        ~Support();
};

//class Exam;
class Exam{
    private:
        string examId;
        string examName;
        string examType;
        char grade;
        string examDate;

    public:
```

```
        Exam();
    Exam(string e_id, string e_name, string e_type, string e_date);
    void setExamGrade(string e_id,string e_name, char gr,Course *c);
    void getExamParticipation();
    void displayExamDetails();
    void displayExamGrade();
    ~Exam();
};

//class User;
class User{
  protected:
        string email;
    string password;
                Register *register;//COMPOSITION
                UserActivity *useractivity;//COMPOSITION

  public:
        User();
        User(string em, string pswd,Register *rg,UserActivity *ua);
    void setEmail(string em);
    void setPassword(string pswd);
    string verifyLogin();
    ~User();
};

//class UserActivity;
class UserActivity{
```

```
    private:
        string lastLogin;
        string timeLogged;
        string loginStatus;

    public:
            UserActivity();
        string getLastLogin();
        string getTimeLogged();
        string getLoginStatus();
        ~UserActivity();
};

//class Register;
class Register{
    private:
        string registerEmail;
        string firstName;
        string lastName;
        string registerPassword;
        string registerDate;

    public:
            Register();
        Register(string r_email, string f_name, string l_name,string r_pswd, string r_date);
        void login();
        ~Register();
};
```

```
//class Visitor;
class Visitor{
   private:
      string name;
                  Register *register;//UNI-DIRECTIONAL ASSOCIATION

   public:
         Visitor();
      void signUp(string nm, Register *rg);
      void viewCourse();
      ~Visitor();
};
```

OnlineTeachertraining.cpp

```
#include<iostream>
#include"online teacher training.h"
using std::string;

//-----Implementing Functions----
//class Course
Course::Course(string id, string name, sring key, string type,Instructor *i,Student *s,Resources *r){
         courseId = id;
         courseName = name;
         enrollKey = key;
         courseType = type;
  ins = i;
  stu = s;
  res = r;
```

```
}
void Course::displayCourseDetails(){
        cout<<"Course ID: "<<courseId<<endl;
        cout<<"Course Name: "<<courseName<<endl;
        cout<<"Enroll Key: "<<enrollKey<<endl;
        cout<<"Course Type: "<<courseType<<endl;
}
void Course::addNewCourses(){
        cout<<"Input Course ID: ";
        cin>>courseId;
        cout<<"Input Course Name: ";
        cin>>courseName;
        cout<<"Input Enroll Key: ";
        cin>>enrollKey;
        cout<<"Input Course Type: ";
        cin>>courseType;
}
void Course:: genarateCourseCertificate(string id, string name, string type, string c_no, string
c_name, string c_date){}
Course::~Course(){}

//class Schedule
Schedule::Schedule(int s_no, string s_name, string s_type){
          scheduleNo = s_no;
   scheduleName = s_name;
   scheduleType = s_type;
}
void Schedule::displaySchedule(){
        cout<<"Schedule Number: "<<scheduleNo<<endl;
```

```
	cout<<"Schedule Name: "<<scheduleName<<endl;
	cout<<"Schedule Type: "<<scheduleType<<endl;
}
void Schedule::setSessionDetails(int s_id, string se_name,Course *c){}
void Schedule::addSession(){
	cout<<"Input Schedule Number: ";
	cin>>scheduleNo;
	cout<<"Input Schedule Name: ";
	cin>>scheduleName;
	cout<<"Input Schedule Type: ";
	cin>>scheduleType;
}
Schedule::~Schedule(){}

//class Resources
Schedule::Resources(string r_id, string r_name, string r_type){
	resourceId = r_id;
   resourceName = r_name;
   resourceType = r_type;
}
void Schedule::addNewResource(){
	cout<<"Input Resource ID: ";
	cin>>resourceId;
	cout<<"Input Resource Name: ";
	cin>>resourceName;
	cout<<"Input Resource Type: ";
	cin>>resourceType;
}
```

```
void Schedule::displayResources(){
        cout<<"Resource ID: "<<resourceId<<endl;
        cout<<"Resource Name: "<<resourceName<<endl;
        cout<<"Resource Type: "<<resourceType<<endl;
}
Resources::~Resources(){}

//class Student
Student::Student(string s_id, string s_name, int s_nic, string s_dob,Course *c,string em, string pswd,Register *rg,UserActivity *ua:):User(em,pswd,*rg,*ua){
        studentId = s_id;
   studentName = s_name;
   studentNIC = s_nic;
   studentDateOfBirth = s_dob;
 cour = c;
}
void Student::displayStudentDetails(){
        cout<<"Student ID: "<<studentId<<endl;
        cout<<"Student Name: "<<studentName<<endl;
        cout<<"Student NIC: "<<studentNIC<<endl;
        cout<<"Student Date of Birth: "<<studentDateOfBirth<<endl;
}
void Student::markStudentParticipation(string s_id, string s_name, int s_nic,string m_name, string a_date){}
void Student::askQuestions(string s_id, string s_name, string stq_no,Support *sp){}
Student::~Student(){}

//class Technician
Technician::Technician(string t_id, string t_name, int t_contact,string em, string pswd,Register *rg,UserActivity *ua:):User(em,pswd,*rg,*ua){
```

```
        technicianId = t_id;
    technicianName = t_name;
    technicianContact = t_contact;
}
void Technician::addNewTechnician(){
        cout<<"Input Technician ID: ";
        cin>>technicianId;
        cout<<"Input Technician Name: ";
        cin>>technicianName;
        cout<<"Input Technician Contact Number: ";
        cin>>technicianContact;
}
void Technician::sendNotification(string t_id, string t_name,int n_no){
        technicianId = t_id;
    technicianName = t_name;
    notificatonNo = n_no;
}
void Technician::setNewFeature(){}
Technician::~Technician();


//class Instructor
Instructor::Instructor(string i_id, string i_name, int i_contactno,Course *c,string em, string
pswd,Register *rg,UserActivity *ua:):User(em,pswd,*rg,*ua){
        instructorId = i_id;
    intructorName = i_name;
    instructorContactNo = i_contactno;
   cour = c;
}
void Instructor::displayInstructorDetails(){
```

```
	cout<<"Instructor ID: "<<instructorId<<endl;
	cout<<"Instructor Name: "<<intructorName<<endl;
	cout<<"Instructor Contact number: "<<instructorContactNo<<endl;
}
void Instructor::getparticiipation(){}
void Instructor::addNewInstructors(){
	cout<<"Input Instructor ID: ";
	cin>>instructorId;
	cout<<"Input Instructor Name: ";
	cin>>intructorName;
	cout<<"Input Instructor Contact Number: ";
	cin>>instructorContactNo;
}
void Instructor::setInstructorActivities(){}
Instructor::~Instructor(){}

//class Administrator
Administrator::Administrator(string a_id,string a_name,int a_contact, int a_nic,string em, string pswd,Register *rg,UserActivity *ua:):User(em,pswd,*rg,*ua){
	administratorId = a_id;
   administratorName = a_name;
   adminContact = a_contact;
   adminNIC = a_nic;
}
void Administrator::displayAdminDetails(){
	cout<<"Student ID: "<<administratorId<<endl;
	cout<<"Student Name: "<<administratorName<<endl;
	cout<<"Student NIC: "<<adminContact<<endl;
	cout<<"Student Date of Birth: "<<adminNIC<<endl;
```

```
}
        void postAnswers(string a_id,Support *sp);

Administrator::~Administrator(){}

//class Support
Support::Support(int q_no, string q_date,Administrator *a){
        questionNo = q_no;
    questionDate = q_date;
   admin = a;
}
void Support::displayQuestion(){
        cout<<"Question Number: "<<questionNo<<endl;
        cout<<"Question Date: "<<questionDate<<endl;
}
void Support::postQuestion(){}
void Support::replyToCustomers(){}
Support::~Support(){}

//class Exam
Exam::Exam(string e_id, string e_name, string e_type, string e_date){
        examId = e_id;
    examName = e_name;
    examType = e_type;
    examDate = e_date;
}
void Exam::setExamGrade(string e_id,string e_name, char gr,Course *c){
        examId = e_id;
```

```
    examName = e_name;
    grade = gr;
  cour = c;
}
void Exam::getExamParticipation(){}
void Exam::displayExamDetails(){
            cout<<"Exam ID: "<<examId<<endl;
            cout<<"Exam Name: "<<examName<<endl;
            cout<<"Exam type: "<<examType<<endl;
            cout<<"Exam Date: "<<examDate<<endl;
}
void Exam::displayExamGrade(){
            cout<<"Exam ID: "<<examId<<endl;
            cout<<"Exam Name: "<<examName<<endl;
            cout<<"Grade: "<<grade<<endl;
}
Exam::~Exam(){}

//class User
User::User(string em, string pswd,Register *rg,UserActivity *ua){
            email = em;
    password = pswd;
   reg = rg;
    useract = ua;
}
void User::setEmail(string em){
            email = em;
}
```

```
void User::setPassword(string pswd){
        password = pswd;
}
string User::verifyLogin(){}
User::~User(){}

//class UserActivity
UserActivity::UserActivity(string ll, string tl,string ls){
        lastLogin = ll;
   timeLogged = tl;
   loginStatus = ls;
}
string UserActivity::getLastLogin(){}
string UserActivity::getTimeLogged(){}
string UserActivity::getLoginStatus(){}
UserActivity::~UserActivity(){}

//class Register
Register::Register(string r_email, string f_name, string l_name,string r_pswd, string r_date){
        registerEmail = r_email;
   firstName = f_name;
   lastName = l_name;
        registerPassword = r_pswd;
   registerDate = r_date;
}
void Register::login(){}
Register::~Register(){}
```

```
//class Visitor
Visitor::Visitor(string v_name){
        name = v_name;
}
void Visitor::signUp(string nm, Register *rg){}
void Visitor::viewCourse(){}
Visitor::~Visitor(){}
```

Main.cpp

```
//constractor User
  User *u1 = new User("shaliimeshi@gmail.com","Iloveteacher001",ua1);
     u1->setEmail(string em);
     u1-> setPassword(string pswd);
     u1-> verifyLogin();

    User *u2 = new User("ovins@yahoo.com","Blackeyes534",ua2);
    u2->setEmail(string em);
     u2-> setPassword(string pswd);
     u2-> verifyLogin();

    User *u3 = new User("Lakindu004@gmail.com","Sleepingbeauty@3",ua3);
    u3->setEmail(string em);
     u3-> setPassword(string pswd);
     u3-> verifyLogin();

   User *u4 = new User("Thisara.Ins@sllitex.co","Makeit001@",ua4);
    u4->setEmail(string em);
```

```
  u4-> setPassword(string pswd);
  u4-> verifyLogin();

User *u5 = new User("Sachi.Ins@sllitex.com","23456@INS",ua5);
 u5->setEmail(string em);
  u5-> setPassword(string pswd);
  u5-> verifyLogin();

User *u6 = new User("Rivi.Ins@sllitex.com","Dostudy123",ua6);
 u6->setEmail(string em);
  u6-> setPassword(string pswd);
  u6-> verifyLogin();

User *u7= new User("oshan@sllitex.com","Aid001@",ua7);
 u7->setEmail(string em);
  u7-> setPassword(string pswd);
  u7-> verifyLogin();

User *u8 = new User("gaveesh@sllitex.com","Bluewhite05",ua8);
 u8->setEmail(string em);
  u8-> setPassword(string pswd);
  u8-> verifyLogin();

User *u9 = new User("suren@sllitex.com","#2suren",ua9);
 u9->setEmail(string em);
  u9-> setPassword(string pswd);
  u9-> verifyLogin();
```

```
//constractor UserActivity
   UserActivity *ua1 = new UserActivity("17:00:00","14:45:00","Offline");
    ua1->getLastLogin();
    ua1-> getTimeLogged();
    ua1-> getLoginStatus();

UserActivity *ua2 = new UserActivity("20:00:00","19:45:00","Offline");
     ua2->getLastLogin();
    ua2-> getTimeLogged();
    ua2-> getLoginStatus();

UserActivity *ua3 = new UserActivity("16:00:00","","Online");
      ua3->getLastLogin();
    ua3-> getTimeLogged();
    ua3-> getLoginStatus();

 UserActivity *ua4 = new UserActivity("09:54:00","","Online");
      ua4->getLastLogin();
    ua4-> getTimeLogged();
    ua4-> getLoginStatus();
 UserActivity *ua5 = new UserActivity("18:00:00","18:30:00","Offline");
      ua5->getLastLogin();
    ua5-> getTimeLogged();
    ua5-> getLoginStatus();
 UserActivity *ua6 = new UserActivity("16:00:00","","Online");
```

```
    ua6->getLastLogin();
   ua6-> getTimeLogged();
   ua6-> getLoginStatus();
 UserActivity *ua7 = new UserActivity("16:45:00","19:00:00","Offline");
    ua7->getLastLogin();
   ua7-> getTimeLogged();
   ua7-> getLoginStatus();
 UserActivity *ua8 = new UserActivity("16:00:00","","Online");
    ua8->getLastLogin();
   ua8-> getTimeLogged();
   ua8-> getLoginStatus();
 UserActivity *ua9 = new UserActivity("10:29:00","14:40:00","Offline");
    ua9->getLastLogin();
   ua9-> getTimeLogged();
   ua9-> getLoginStatus();

//constractor Register
Register *rg1 = new Register(u1,"Shaliniya","Imeshini","2012-07-10");
   rg1->login();
  Register *rg1 = new Register(u1,"Ovini","Yazara","2010-04-14");
   rg1->login();
  Register *rg1 = new Register(u1,"Lakindu","Gathsara","2020-02-05");
   rg1->login();
//constractor Visitor
Visitor *v1 = new Visitor("Janka Pruthuvi");
    v1-> signUp(string nm, Register *rg);
    v1-> viewCourse();
Visitor *v2 = new Visitor("Deshitha Thejan");
```

```
    v2->signUp(string nm, Register *rg);

    v2-> viewCourse();

}
```

Contribution

Name and IT no	Contribution
IT21196942 Hettiarachige.S.I.	Classes- Course class Technician Class User Class User Activity class Register class Visitor class
IT21197864 Kahatapitiya.K.K.O.Y	Class- Schedule Class Instructor Class
IT21194276 Muhtunayaka.J.P.V	Class- Administrator Class Support Class
IT21195020 Rabukpotha.R.R.M.L.G	Class- Student class Exam class
IT21195334 Athukorala.D.T.A	Class-Resource class