Topic : Identity Issuing Service

Group no : MLB_08.01_11

Campus : Malabe / ~~Metro~~ / ~~Matara~~ / ~~Kandy~~ / ~~Kurunegala~~ / ~~Kandy~~ / ~~Jaffna~~

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.

Registration No	Name	Contact Number
IT21160448	Akalanka P. A. A.	0703011142
IT21162978	Athukorala W. A. A. D. D. T.	0771606447
IT21160516	Chamikara T. A. D. G.	0713644084
IT21161742	Kumari P. D. L.	0753152110
IT21160202	Wijekoon W. M. S. R.	0768843142

Exercise 01

“DLMS” (Driving License Management System) is an online driving license issuing service. This system helps customers get their license easier and faster and driving school owners can advertise their course among the customers by posting classes in the courses section. When a visitor comes to the website for the first time, the homepage is displayed. The currency is in Sri Lankan Rupees, and the default language is "English." Users who want to alter their language can do so according to their preferences. User, admin, and driving school login methods are all displayed on the homepage. Users can choose and log in to the system with ease. They can make an account if they don't already have one. They can also change their password if they forget it. They can see the DLMS services that are available to them (Driving License Management System). Users can register to obatain a new license. When a visitor logs in as an admin or a driving school, they will be taken to the dashboard. They can manage their services from the dashboard. The admin user has complete control over the system's functions. He has the ability to analyze all current modifications in the system and accept or reject them according on the system's standards, rules, and regulations. For example, they can clarify user details before registering a new user to receive a new license, or they can reject a learner's firm that requests membership in DLMS with admin permission for their sector. They can also update the system, record all payment details, and generate reports based on the data. After then, the administrator can make more judgments by combining with other agents. Learner's advertising admin is in charge of all advertising for learners as well as other important advertisements on the website. They are also eligible for driving school student registrations and make appropriate payments from students who are known to be driving school students. They are also in charge of class scheduling and distributing notices pertaining to driving schools. This driving license management system is being developed as part of the internet and web technologies module assignment. So yet, we've just completed the system's basic functions. By the conclusion of the semester, we expect to have created a more innovative and userfriendly system.

Requirements

1. Guest search for the DLMS and search for services available.
2. Guest can register to the system as user or Driving License Advertiser. To register both type of users must provide the information requested by the system such as name, NIC, date of birth, address, email, and the user type.
3. Then registered user can login to the system using their credentials and can update the user details and view the current details.
4. Register user can browse and can identify the service types given by system such as new license registration, license renewal and applying for a foreign license and can view the service type details including service type, service ID and Service fees.
5. Register user must fill the application, submit the required documents, and make necessary payments according to the service type selected.
6. Register user can make payment via credit/ debit cards and make service registration successful and can add reviews, view reviews submitted by others about services.
7. To obtain a new license, a register user must complete examinations which consist of a written exam and a practical exam which can be selected using exam type and registered user can register to the relevant exam by providing exam ID, setting up convenient dates and times.
8. Registered user can view the scheduled exams, get the exam details and apply for postponements of exams. For the written exam, there are practice exam sessions provided by the system for the registered user to do.
9. For the practical exam, a register user can register for a driving school which are recommended by the system and view all the details of driving schools and courses published by driving schools.
10. A register user can give feedbacks/ comments and FAQ.
11. Driving school advertiser can create and publish the advertisements related to driving school.
12. Driving school advertiser can manage the profile related to driving school by updating the existing advertisements, update contact details and update packages.
13. Driving school advertiser can publish the course details and schedule the course classes.

14. Driving school advertiser can view customer payments and transactions related to driving school.
15. Driving school advertiser can answer to the customer questions related to driving school.
16. Admin can generate reports by validating the submitted details, documents, and payments by the user.
17. Admin can manage information and status which may include notices, usual schedules, and payments. Also, admin can manage FAQ and answer for the questions.
18. System agent act on reports generated by admin and update the user according to the report details such as registration confirmation/ rejection, examination dates, results and progress of license issuing via emails
19. System agent issues completed licenses.

Classes

1. User
2. Registered user
3. Driving school advertiser
4. Driving school
5. Course
6. Examination
7. Payment
8. Services
9. Agent
10. Feedback
11. Reviews
12. Report

CRC Card

Class name: User	
Responsibilities:	**Collaborations:**
Store User Details	
Display User Details	
Update User Details	

Class name: Register User	
Responsibilities:	**Collaborations:**
Store Register User ID	
Store Login Credentials	
Validate Registered User Credential	

Class name: Driving School Advertiser	
Responsibilities:	**Collaborations:**
Store Driving School Advertiser ID	
Store Login Credentials	
Validate Seller Credentials	

Class name: Driving School	
Responsibilities:	**Collaborations:**
Publish Advertisements	Driving school advertiser
Profile Management	Driving school advertiser
Publishing and Scheduling Courses	Driving school advertiser

Class name: Course	
Responsibilities:	**Collaborations:**
View Course Details	User
Update Course Details	Driving School Advertiser

Class name: Examination	
Responsibilities:	**Collaborations:**
Publishing Written Exam	
Conducting Practical Exam	Driving school
Publishing Practical Exam	

Class name: Services	
Responsibilities:	**Collaborations:**
Publish Service Types	
Display Service Details	
View Service Types and Details	

Class name: Payment	
Responsibilities:	**Collaborations:**
Store Card Details	Services
Store Payment Type	Services
Display payment Details	

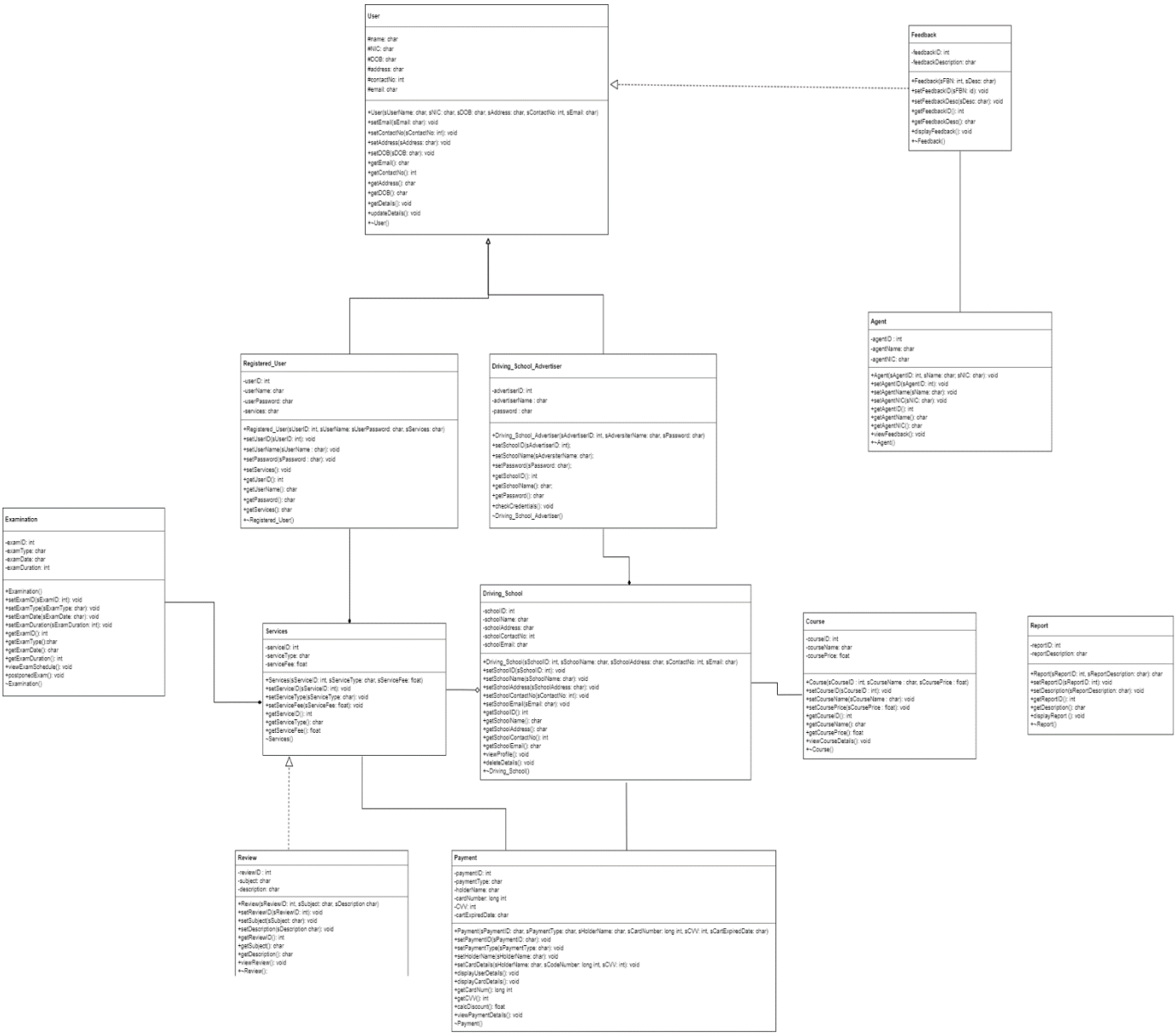
Class name: Agent	
Responsibilities:	**Collaborations:**
Store Agent ID	
Manage FAQ	Feedback

Class name: Feedback	
Responsibilities:	**Collaborations:**
Add Feedback/ Comments	User
View Feedback/ Comment	Agent

Class name: Reviews	
Responsibilities:	**Collaborations:**
Add Reviews	
View Reviews	

Class name: Report	
Responsibilities:	**Collaborations:**
Generate Report	
Display Report	

Exercise 03


User
#name: char
#NIC: char
#DOB: char
#address: char
#contactNo: int
#email: char
+User(sUserName: char, sNIC: char, sDOB: char, sAddress: char, sContactNo: int, sEmail: char)
+setEmail(sEmail: char): void
+setContactNo(sContactNo: int): void
+setAddress(sAddress: char): void
+setDOB(sDOB: char): void
+getEmail(): char
+getContactNo(): int
+getAddress(): char
+getDOB(): char
+getDetails(): void
+updateDetails(): void
+~User()
Feedback
-feedbackID: int
-feedbackDescription: char
+Feedback(sFBN: int, sDesc: char)
+setFeedbackID(sFBN: id): void
+setFeedbackDesc(sDesc: char): void
+getFeedbackID(): int
+getFeedbackDesc(): char
+displayFeedback(): void
+~Feedback()
Agent
-agentID : int
-agentName: char
-agentNIC: char
+Agent(sAgentID: int, sName: char, sNIC: char): void
+setAgentID(sAgentID: int): void
+setAgentName(sName: char): void
+setAgentNIC(sNIC: char): void
+getAgentID(): int
+getAgentName(): char
+getAgentNIC(): char
+viewFeedback(): void
+~Agent()
Registered_User
-userID: int
-userName: char
-userPassword: char
-services: char
+Registered_User(sUserID: int, sUserName: sUserPassword: char, sServices: char)
+setUserID(sUserID: int): void
+setUserName(sUserName : char): void
+setPassword(sPassword : char): void
+setServices(): void
+getUserID(): int
+getUserName(): char
+getPassword(): char
+getServices(): char
+~Registered_User()
Driving_School_Advertiser
-advertiserID: int
-advertiserName : char
-password : char
+Driving_School_Advertiser(sAdvertiserID: int, sAdverstetName: char, sPassword: char)
+setSchoolID(sAdvertiserID: int);
+setSchoolName(sAdverseterName: char);
+setPassword(sPassword: char);
+getSchoolID(): int
+getSchoolName(): char;
+getPassword(): char
+checkCredentials(): void
~Driving_School_Advertiser()
Examination
-examID: int
-examType: char
-examDate: char
-examDuration: int
+Examination()
+setExamID(sExamID: int): void
+setExamType(sExamType: char): void
+setExamDate(sExamDate: char): void
+setExamDuration(sExamDuration: int): void
+getExamID(): int
+getExamType():char
+getExamDate(): char
+getExamDuration(): int
+viewExamSchedule(): void
+postponedExam(): void
~Examination()
Services
-serviceID: int
-serviceType: char
-serviceFee: float
+Services(sServiceID: int, sServiceType: char, sServiceFee: float)
+setServiceID(sServiceID: int): void
+setServiceType(sServiceType: char): void
+setServiceFee(sServiceFee: float): void
+getServiceID(): int
+getServiceType(): char
+getServiceFee(): float
~Services()
Driving_School
-schoolID: int
-schoolName: char
-schoolAddress: char
-schoolContactNo: int
-schoolEmail: char
+Driving_School(sSchoolID: int, sSchoolName: char, sSchoolAddress: char, sContactNo: int, sEmail: char)
+setSchoolID(sSchoolID: int): void
+setSchoolName(sSchoolName: char): void
+setSchoolAddress(sSchoolAddress: char): void
+setSchoolContactNo(sContactNo: int): void
+setSchoolEmail(sEmail: char): void
+getSchoolID(): int
+getSchoolName(): char
+getSchoolAddress(): char
+getSchoolContactNo(): int
+getSchoolEmail(): char
+viewProfile(): void
+deleteDetails(): void
+~Driving_School()
Course
-courseID: int
-courseName: char
-coursePrice: float
+Course(sCourseID : int, sCourseName : char, sCoursePrice : float)
+setCourseID(sCourseID : int): void
+setCourseName(sCourseName : char): void
+setCoursePrice(sCoursePrice : float): void
+getCourseID(): int
+getCourseName(): char
+getCoursePrice(): float
+viewCourseDetails(): void
+~Course()
Report
-reportID: int
-reportDescription: char
+Report(sReportID: int, sReportDescription: char): char
+setReportID(sReportID: int): void
+setDescription(sReportDescription: char): void
+getReportID(): int
+getDescription(): char
+displayReport (): void
+~Report()
Review
-reviewID : int
-subject: char
-description: char
+Review(sReviewID: int, sSubject: char, sDescription char)
+setReviewID(sReviewID: int): void
+setSubject(sSubject: char): void
+setDescription(sDescription char): void
+getReviewID(): int
+getSubject(): char
+getDescription(): char
+viewReview(): void
+~Review();
Payment
-paymentID: int
-paymentType: char
-holderName: char
-cardNumber: long int
-CVV: int
-cartExpiredDate: char
+Payment(sPaymentID: char, sPaymentType: char, sHolderName: char, sCardNumber: long int, sCVV: int, sCartExpiresDate: char)
+setPaymentID(sPaymentID: char): void
+setPaymentType(sPaymentType: char): void
+setHolderName(sHolderName: char): void
+setCardDetails(sHolderName: char, sCodeNumber: long int, sCVV: int): void
+displayUserDetails(): void
+displayCardDetails(): void
+getCardNum(): long int
+getCVV(): int
+calcDiscount(): float
+viewPaymentDetails(): void
~Payment()

Exercise 04

Source Code

main.cpp

```
/*
   IT Number: IT21160448, IT21162978, IT21160516, IT21161742, IT21160202
*/

#include <iostream>
#include "User.h"
#include "Report.h"
#include "Course.h"
#include "Driving_School.h"
#include "Driving_School_Advertiser.h"
#include "Service.h"
#include "Payment.h"
#include "Agent.h"
#include "Feedback.h"
#include "Review.h"
#include "Registered_User.h"
#include "Examination.h"

using namespace std;

int main() {
   // User
   cout << "----------------User----------------" << endl << endl;
   User *user1 = new User("bentennyson", "24343", "02/11/2000", "Colombo, Sri Lanka",
1111111111, "ben@gmail.com");
   user1->getDetails();

   cout << endl << endl << endl << endl;

   // report
   cout << "----------------Report----------------" << endl << endl;
   Report *report1 =  new Report(123, "New License: Rs. 25,000");
   report1->displayReport();

   cout << endl << endl << endl << endl;

   // course
   cout << "----------------Course----------------" << endl << endl;
   Course *course1 = new Course(12, "Light Weight Vehicle", 25000.00);
   course1->viewCourseDetails();
```

```
    cout << endl << endl << endl << endl;

    // driving school
    cout << "------------Driving School-------------" << endl << endl;
    Driving_School *driving_school1 = new Driving_School(587, "Rathna Learners", "Colombo, Sri
Lanka", 0111365145, "rathna@gmail.com");
    driving_school1->viewProfile();

    cout << endl << endl << endl << endl;

    // driving school advertiser
    cout << "-------Driving School Advertiser-------" << endl << endl;
    Driving_School_Advertiser *advertiser1 = new Driving_School_Advertiser("rathnaadvertiser1",
"789254", "12/08/1999", "Malabe, Sri Lanka", 1111111111, "rathnaadver@gmail.com", 324, "Julia
Ann", "pass123");
    advertiser1->getDetails();
    cout << endl;
    cout << "view advertiser" << endl;
    advertiser1->viewAdvertiser();
    Driving_School_Advertiser *advertiser2 = new Driving_School_Advertiser("Ameesha Akalanka",
"829654", "25/02/1994", "Kandy, Sri Lanka", 22222222, "rathnaadver2@gmail.com", 325,
"rathnaadvertiser2", "aka123");

    cout << endl << endl << endl << endl;

    // add advertiser to the driving school
    driving_school1->setAgent(advertiser1, advertiser2);
    driving_school1->viewAgent();

    cout << endl << endl << endl << endl;

    // registered user
    cout << "------------Registered User-------------" << endl << endl;
    Registered_User *registered_user1 = new Registered_User("Anna Perera", "890254",
"08/02/1990", "Kandy, Sri Lanka", 2222222222, "anna@gmail.com", 8792, "annap", "cat123",
"Renew License");
    registered_user1->showRegisteredUser();
    Registered_User *registered_user2 = new Registered_User("Jason Perera", "454354",
"23/12/19807", "Gamapaha, Sri Lanka", 3333333333, "jason@gmail.com", 23432, "jasonp",
"jason123", "New License");
        registered_user1->showRegisteredUser();
    cout << endl << endl << endl << endl;

    // services
    cout << "----------------Services----------------" << endl << endl;
    Service *service1 = new Service(120, "New License", 25000);
    Service *service2 = new Service(122, "License Renewal", 25000);
```

```
service1->displayServiceDetails();
service2->displayServiceDetails();

// add registered user to service
service1->setRegisteredUser(registered_user1, registered_user2);
service1->displayRegisteredUser();

cout << endl << endl << endl << endl;

// add service to driving school
cout << "------Add Service to Driving School------" << endl;
driving_school1->addService(service1);
driving_school1->displayService();

cout << endl << endl << endl << endl;

// payment
cout << "----------------Payment----------------" << endl << endl;
Payment *payment1 = new Payment(789, "VISA", "Ervil Lewis", 384238432, 890, "21/12/2021");
payment1->viewPaymentDetails();

cout << endl << endl << endl << endl;

// set examination
cout << "----------------Examination----------------" << endl << endl;
Examination *exam1 = new Examination(3423, "Practical", "21/12/2021", 30);
exam1->viewExamSchedule();

cout << endl << endl << endl << endl;

// add examination to services
cout << "----------Add Examination to service--------" << endl << endl;
service1->setExamination(exam1);
service1->viewExamDetails();

cout << endl << endl << endl << endl;

//agent
cout << "----------------Agent----------------" << endl << endl;
Agent *agent1 = new Agent(908, "Kevin Levin", "732874");
cout<<"Agent ID: "<<agent1->getAgentID()<<endl;
agent1->displayAgentDetails();
Agent *agent2 = new Agent(909, "Peter Parker", "939622");
cout<<"Agent ID: "<<agent1->getAgentID()<<endl;
agent2->displayAgentDetails();

cout << endl << endl << endl << endl;
```

```
//feedback
cout << "----------------Feedback----------------" << endl << endl;
Feedback *FB1 = new Feedback(12345);
FB1->setFeedbackDesc("Good", user1);
FB1->displayFeedback();

cout << endl << endl << endl << endl;

//review
cout << "----------------Review----------------" << endl << endl;
Review *review1 = new Review(550,"License","This is a good service");
review1->setDescription("This is a good service", service1);
cout << "Review ID: " <<review1->getReviewID() <<endl;
review1-> viewReview();
cout<<endl;
Review *review2 = new Review(551,"Service","Exellent service");
review1->setDescription("Exellent service", service2);
cout << "Review ID: " <<review2->getReviewID() <<endl;
review2-> viewReview();

cout << endl << endl << endl << endl;


cout << "----------------Remove Objects----------------" << endl << endl;
// remove examination
delete exam1;
// remove payment
delete payment1;
// remove services
delete service1;
delete service2;
// remove report
delete report1;
// remove course
delete course1;
// remove driving school advertiser
delete advertiser1;
// remove driving school
delete driving_school1;
// remove registered user
delete registered_user1;
delete registered_user2;
// remove agent
delete agent1;
delete agent2;
// remove user
delete user1;
//remove feedback
delete FB1;
```

```
    //remove review
    delete review1;
    delete review2;
}
```

Agent.h

```
/*
    IT Number: IT21162978
*/

#pragma once

#define SIZE1 2

    class Feedback;

    class Agent
    {
        private:
            int agentID;
            char agentName[50];
            char agentNIC[20];
            Feedback *FB[SIZE1];

        public:
            Agent(int sAgentID, const char sName[],const char sNIC[]);
            void setAgentID(int sAgentID);
            void setAgentName(const char sName[]);
            void setAgentNIC(const char sNIC[]);
            int getAgentID();
      void checkFeedback(Feedback *FB);
            char *getAgentName();
            char *getAgentNIC();
            void displayAgentDetails();
            void viewFeedback();
            ~Agent();
    };
```

Agent.cpp

```
/*
    IT Number: IT21162978
```

```
*/

#include<iostream>
#include<cstring>
#include "Agent.h"
#include "Feedback.h"

using namespace std;

Agent::Agent(int sAgentID, const char sName[],const char sNIC[]){
    agentID=sAgentID;
    strcpy(agentName,sName);
    strcpy(agentNIC,sNIC);
}

void Agent::setAgentID(int sAgentID){
    agentID=sAgentID;
}

int Agent::getAgentID(){
    return agentID;
}

void checkFeedback(Feedback *FB){

}

char *Agent::getAgentName(){
    return agentName;
}

char *Agent::getAgentNIC(){
    return agentNIC;
}

void Agent::displayAgentDetails(){
    cout << "Agent Name : " << agentName <<endl;
    cout << "Agent NIC : " <<agentNIC << endl;

}

void Agent::viewFeedback(){
}

Agent::~Agent(){
    cout << "Deleting Agent ID " <<agentID <<endl;
}
```

Course.h

```
/*
   IT Number: IT21160202
*/

#pragma once
#include "Driving_School.h"

class Driving_School;

class Course{
private:
   int courseID;
   char courseName[30];
   float coursePrice;
   Driving_School *driving_school;
public:
   Course(int sCourseID, const char sCourseName[], float sCoursePrice);
   void setCourseID(int sCourseID);
   void setCourseName(const char sCourseName[]);
   void setCoursePrice(float sCoursePrice);
   int getCourseID();
   char *getCourseName();
   float getCoursePrice();
   void viewCourseDetails();
   ~Course();
};
```

Course.cpp

```
/*
   IT Number: IT21160202
*/

#include <iostream>
#include <cstring>
#include "Course.h"

using namespace std;

Course::Course(int sCourseID, const char sCourseName[], float sCoursePrice){
   courseID = sCourseID;
   strcpy(courseName, sCourseName);
   coursePrice = sCoursePrice;
}

void Course::setCourseID(int sCourseID){
```

```
    courseID = sCourseID;
}

void Course::setCourseName(const char sCourseName[]){
    strcpy(courseName, sCourseName);
}

void Course::setCoursePrice(float sCoursePrice){
    coursePrice = sCoursePrice;
}

int Course::getCourseID(){
    return courseID;
}

char *Course::getCourseName(){
    return courseName;
}

float Course::getCoursePrice(){
    return coursePrice;
}

void Course::viewCourseDetails(){
    cout << "Course ID: " << courseID << endl;
    cout << "Course Name: " << courseName << endl;
    cout << "Course Price: " << coursePrice << endl;
}

Course::~Course(){
    cout << "Course is deleted!" << endl;
}
```

Driving_School_Advertiser.h

```
/*
    IT Number: IT21160448
*/

#pragma once
#include "User.h"

class Driving_School_Advertiser :public User {
private:
    int advertiserID;
    char advertiserName[20];
    char password[20];
public:
```

```
    Driving_School_Advertiser (const char sUserName[], const char sNIC[], const char sDOB[], const
char sAddress[], long int sContactNo, const char sEmail[], int sAdvertiserID, const char
sAdversiterName[], const char sPassword[]);
    void setSchoolID(int sAdvertiserID);
    void setSchoolName(const char sAdversiterName[]);
    void setPassword(const char sPassword[]);
    int getSchoolID();
    char *getUserName();
    char *getPassword();
    void checkCredentials();
    void viewAdvertiser();
    ~Driving_School_Advertiser();
};
```

Driving_School_Advertiser.cpp

```
/*
    IT Number: IT21160448
*/

#include <iostream>
#include <cstring>
#include "Driving_School_Advertiser.h"
#include "User.h"

using namespace std;

Driving_School_Advertiser::Driving_School_Advertiser (const char sUserName[], const char sNIC[],
const char sDOB[], const char sAddress[], long int sContactNo, const char sEmail[], int
sAdvertiserID, const char sAdversiterName[], const char sPassword[]):User(sUserName, sNIC,
sDOB, sAddress, sContactNo, sEmail){
    advertiserID = sAdvertiserID;
    strcpy(advertiserName, sAdversiterName);
    strcpy(password, sPassword);
}

void Driving_School_Advertiser::setSchoolID(int sAdvertiserID){
    advertiserID = sAdvertiserID;
}

void Driving_School_Advertiser::setSchoolName(const char sAdversiterName[]){
    strcpy(advertiserName, sAdversiterName);
}

void Driving_School_Advertiser::setPassword(const char sPassword[]){
    strcpy(password, sPassword);
```

```
}

int Driving_School_Advertiser::getSchoolID(){
    return advertiserID;
}

char *Driving_School_Advertiser::getUserName(){
    return userName;
}

char *Driving_School_Advertiser::getPassword(){
    return password;
}

void Driving_School_Advertiser::checkCredentials(){

}

void Driving_School_Advertiser::viewAdvertiser(){
    cout << "Advertiser ID: " <<  advertiserID << endl;
    cout << "Advertiser Name: " << advertiserName << endl;
    cout << "Advertiser Passowrd: " << password << endl << endl;
}

Driving_School_Advertiser::~Driving_School_Advertiser(){
    cout << "Driving school advertiser is deleted!" << endl;
}
```

Driving_School.h

```
/*
    IT Number: IT21160202
*/
#pragma once
#include "Course.h"
#include "Payment.h"
#include "Driving_School_Advertiser.h"
#include "Service.h"

#define SIZE 2
class Course;
class Payment;
class Service;

class Driving_School{
private:
```

```
    int schoolID;
    char schoolName[20];
    char schoolAddress[30];
    long int schoolContactNo;
    char schoolEmail[20];
    Course *course;
    Payment *payment;
    Driving_School_Advertiser *advertiser[SIZE];
    Service *service;
public:
    Driving_School(int sSchoolID, const char sSchoolName[], const char sSchoolAddress[], long
int   sContactNo, const char sEmail[]);
    void setSchoolID(int sSchoolID);
    void setSchoolName(const char sSchoolName[]);
    void setSchoolAddress(const char sSchoolAddress[]);
    void setSchoolContactNo(long int sContactNo);
    void setSchoolEmail(const char sEmail[]);
    void setAgent(Driving_School_Advertiser *advertiser1, Driving_School_Advertiser *advertiser2);
    int getSchoolID();
    char *getSchoolName();
    char *getSchoolAddress();
    int getSchoolContactNo();
    char *getSchoolEmail();
    void viewProfile();
    void deleteDetails();
    void viewAgent();
    void deleteAgent();
    void addService(Service *service1);
    void displayService();
    ~Driving_School();
};
```

Driving_School.cpp

```
/*
    IT Number: IT21160202
*/
#include <iostream>
#include <cstring>
#include "Driving_School.h"
#include "Driving_School_Advertiser.h"
#define SIZE 2

using namespace std;
```

```
Driving_School::Driving_School(int sSchoolID, const char sSchoolName[], const char
sSchoolAddress[], long int sContactNo, const char sEmail[]){
    schoolID = sSchoolID;
    strcpy(schoolName, sSchoolName);
    strcpy(schoolAddress, sSchoolAddress);
    schoolContactNo = sContactNo;
    strcpy(schoolEmail, sEmail);
}

void Driving_School::setSchoolID(int sSchoolID){
    schoolID = sSchoolID;
}

void Driving_School::setSchoolName(const char sSchoolName[]){
    strcpy(schoolName, sSchoolName);
}

void Driving_School::setSchoolAddress(const char sSchoolAddress[]){
    strcpy(schoolAddress, sSchoolAddress);
}

void Driving_School::setSchoolContactNo(long int sContactNo){
    schoolContactNo = sContactNo;
}

void Driving_School::setSchoolEmail(const char sEmail[]){
    strcpy(schoolEmail, sEmail);
}

int Driving_School::getSchoolID(){
    return schoolID;
}

char *Driving_School::getSchoolName(){
    return schoolName;
}

char *Driving_School::getSchoolAddress(){
    return schoolAddress;
}

int Driving_School::getSchoolContactNo(){
    return schoolContactNo;
}

char *Driving_School::getSchoolEmail(){
    return schoolEmail;
}
```

```
void Driving_School::viewProfile(){
    cout << "School ID: " << schoolID << endl;
    cout << "School Name: " << schoolName << endl;
    cout << "School Address: " << schoolAddress << endl;
    cout << "School Contact Number: " << schoolContactNo << endl;
    cout << "School Email: " << schoolEmail << endl;
}

void Driving_School::deleteDetails(){

}

void Driving_School::setAgent(Driving_School_Advertiser *advertiser1, Driving_School_Advertiser
*advertiser2){
    advertiser[0] = advertiser1;
    advertiser[1] = advertiser2;
}

void Driving_School::viewAgent(){
    for(int i = 0; i < SIZE; i++){
        advertiser[i]->viewAdvertiser();
    }
}

void Driving_School::deleteAgent(){
    for(int i = 0; i < SIZE; i++){
        advertiser[i]->~Driving_School_Advertiser();
    }
}

void Driving_School::addService(Service *service1){
    service = service1;
}

void Driving_School::displayService(){
    service->displayServiceDetails();
}

Driving_School::~Driving_School(){
    cout << "Driving school deleted!" << endl;
}
```

Examination.h

```
/*
```

```
  IT Number: IT21161742
*/
#pragma once
class Examination{
  private:
  int examID;
    char examType[20];
    char examDate[20];
    int examDuration;

public:
  Examination(int sExamID, const char sExamType[], const char sExamDate[], int sExamDuration);
    void setExamID(int sExamID);
    void setExamType(const char sExamType[]);
    void setExamDate(const char sExamDate[]);
    void setExamDuration(int sExamDuration);
    int getExamID();
    char *getExamType();
    char *getExamDate();
    int getExamDuration();
    void viewExamSchedule();
    void postponedExam();
    ~Examination();

};
```

Examination.cpp

```
/*
    IT Number: IT21161742
*/

#include "Examination.h"
#include <iostream>
#include <cstring>

using namespace std;

Examination::Examination(int sExamID, const char sExamType[], const char sExamDate[], int
sExamDuration){
    examID = sExamID;
    strcpy(examType, sExamType);
    strcpy(examDate,sExamDate);
    examDuration = sExamDuration;
}

void Examination::setExamID(int sExamID){
    examID = sExamID;
```

```
}

void Examination::setExamType(const char sExamType[]){
    strcpy(examType,sExamType);
}

void Examination::setExamDate(const char sExamDate[]){
    strcpy(examDate,sExamDate);
}

void Examination::setExamDuration(int sExamDuration){
    examDuration = sExamDuration;
}

int Examination::getExamID(){
    return examID;
}

char *Examination::getExamType(){
    return examType;
}

char *Examination::getExamDate(){
    return examDate;
}

int Examination::getExamDuration(){
    return examDuration;
}

void Examination::viewExamSchedule(){
    cout << "Exam ID : " << examID << endl;
    cout << "Exam Type : " << examType << endl;
    cout << "Exam Date : " << examDate << endl;
    cout << "Exam Duration : " << examDuration << endl;
}

void Examination::postponedExam(){

}

Examination::~Examination()
{
    cout << "Delete Examination ID " << examID << endl;
}
```

Feedback.h

```
/*
   IT Number: IT21162978
*/

#pragma once
#define SIZE2 5

class Agent;
class User;

class Feedback
{
   private:
      int feedbackID;
      char feedbackDescription[200];
      Agent *Ag[SIZE2];

   public:
      Feedback(int sFBN);
      void setFeedID(int sFeedback);
      void setFeedbackDesc(const char sDesc[], User *user);
      int getFeedbackID ();
   void markAgent(Agent *Ag[SIZE2]);
      char *getFeedbackDesc();
      void displayFeedback();
      ~Feedback();
};
```

Feedback.cpp

```
/*
   IT Number: IT21162978
*/

#include <iostream>
#include <cstring>
#include "Feedback.h"
#include "User.h"

using namespace std;

Feedback::Feedback(int sFBN){
   feedbackID=sFBN;
}

void Feedback::setFeedID(int sFeedback){
   feedbackID=sFeedback;
```

```
}

void Feedback::setFeedbackDesc(const char sDesc[], User *user1){
    user1->getDetails();
    strcpy(feedbackDescription,sDesc);
}

int Feedback::getFeedbackID(){
    return feedbackID;
}

void markAgent(Agent *Ag[SIZE2]){

}

char *Feedback::getFeedbackDesc(){
    return feedbackDescription;
}

void Feedback::displayFeedback(){
    cout<<"Feedback ID: "<<feedbackID<<endl;
    cout<<"Feedback Description: "<<feedbackDescription<<endl;
}

Feedback::~Feedback(){
    cout << "Deleted Feedback ID " <<feedbackID <<endl;
}
```

Payment.h

```
/*
    IT Number: IT21160516
*/

#pragma once
#include "Driving_School.h"
#include "Service.h"

class Driving_School;
class Service;

class Payment{
    private :
        int paymentID;
        char paymenType[10];
        char holderName[50];
        long int cardNumber;
```

```
        int CVV;
        char CardExpiredDate[20];
        Driving_School *driving_school;
        Service *service;
    public :
        Payment(int sPaymentID, const char sPaymentType[], const char sHolderName[], long int
sCardNumber, int sCVV, const char sCardExpiredDate[]);
        void setPaymentID(int sPaymentID);
        void setPaymentType(const char sPaymentType[]);
        void setHoldersName(const char sHolderName[]);
        void setCardDetails(const char sHolderName[],long int sCardNumber, int sCVV);
        void displayUserDetails();
        void displayCardDetails();
        int getCardNum();
        int getCVV();
        float calcDiscount();
        void viewPaymentDetails();
        ~Payment();
};
```

Payment.cpp

```
/*
    IT Number: IT21160516
*/

#include <iostream>
#include <cstring>
#include "Payment.h"

using namespace std;

Payment::Payment(int sPaymentID, const char sPaymentType[], const char sHolderName[], long
int sCardNumber, int sCVV, const char sCardExpiredDate[]){
    paymentID = sPaymentID;
    strcpy(paymenType, sPaymentType);
    strcpy(holderName, sHolderName);
    cardNumber = sCardNumber;
    CVV = sCVV;
    strcpy(CardExpiredDate, sCardExpiredDate);
}

void Payment::setPaymentID(int sPaymentID){
    paymentID = sPaymentID;
}

void Payment::setPaymentType(const char sPaymentType[]){
    strcpy(paymenType, sPaymentType);
```

```
}

void Payment::setHoldersName(const char sHolderName[]){
    strcpy(holderName, sHolderName);
}

void Payment::setCardDetails(const char sHolderName[],long int sCardNumber, int sCVV){
    strcpy(holderName, sHolderName);
    cardNumber = sCardNumber;
    CVV = sCVV;
}

void Payment::displayUserDetails(){
    cout << "Payment ID: " << paymentID << endl;
    cout << "Payment Type: " << paymenType << endl;
    cout << "Holder Name: " << holderName << endl;
}

void Payment::displayCardDetails(){
    cout << "Card Holder Name: " << holderName << endl;
    cout << "Card Number: " << cardNumber;
    cout << "CVV: " << CVV;
}

int Payment::getCardNum(){
    return cardNumber;
}

int Payment::getCVV(){
    return CVV;
}

float Payment::calcDiscount(){

}

void Payment::viewPaymentDetails(){
    cout << "Payment ID: " << paymentID << endl;
    cout << "Payment Type: " << paymenType << endl;
}

Payment::~Payment(){
    cout << "Payment is deleted!" << endl;
}
```

Registered_User.h

```
/*
```

```
    IT Number: IT21160448
*/
#pragma once
#include "User.h"

class Registered_User: public User {
private:
    int userID;
    char name[50];
    char userPassword[20];
    char services[30];
public:
    Registered_User();
    Registered_User(const char sUserName[], const char sNIC[], const char sDOB[], const char
sAddress[], long int sContactNo, const char sEmail[], int sUserID, const char sName[], const char
sUserPassword[], const char sServices[]);
    void setUserID(int sUserID);
    void setUserName(const char sUserName[]);
    void setPassword(const char sPassword[]);
    void setServices(const char sServices[]);
    int getUserID();
    char *getUserName();
    char *getPassword();
    char *getServices();
    void showRegisteredUser();
    ~Registered_User();
};
```

Registered_User.cpp

```
/*
    IT Number: IT21160448
*/

#include <iostream>
#include <cstring>
#include "User.h"
#include "Registered_User.h"

using namespace std;

Registered_User::Registered_User(){

}
```

```
Registered_User::Registered_User(const char sUserName[], const char sNIC[], const char sDOB[],
const char sAddress[], long int sContactNo, const char sEmail[], int sUserID, const char sName[],
const char sUserPassword[], const char sServices[]):User(sUserName, sNIC, sDOB, sAddress,
sContactNo, sEmail){
    userID = sUserID;
    strcpy(name, sName);
    strcpy(userPassword, sUserPassword);
    strcpy(services, sServices);
}

void Registered_User::setUserID(int sUserID){
    userID = sUserID;
}

void Registered_User::setUserName(const char sName[]){
    strcpy(name, sName);
}

void Registered_User::setPassword(const char sUserPassword[]){
    strcpy(userPassword, sUserPassword);
}

void Registered_User::setServices(const char sServices[]){
    strcpy(services, sServices);
}

int Registered_User::getUserID(){
    return userID;
}

char *Registered_User::getUserName(){
    return userName;
}

char *Registered_User::getPassword(){
    return userPassword;
}

char *Registered_User::getServices(){
    return services;
}

void Registered_User::showRegisteredUser(){
    cout << "User ID: " << userID << endl;
    cout << "Name: " << userName << endl;
    cout << "Services: " << services << endl;
}
```

```
Registered_User::~Registered_User(){
    cout << "Registered user is deleted" << endl;
}
```

Report.h

```
/*
    IT Number: IT21160202
*/
class Report{
private:
    int reportID;
    char reportDescription[30];

public:
    Report(int sReportID, const char sReportDescription[]);
    void setReportID(int sReportID);
    void setDescription(const char sReportDescription[]);
    int getReportID();
    char *getDescription();
    void displayReport();
    ~Report();
};
```

Report.cpp

```
/*
    IT Number: IT21160202
*/

#include <iostream>
#include <cstring>
#include "Report.h"

using namespace std;

Report::Report(int sReportID, const char sReportDescription[]){
    reportID = sReportID;
    strcpy(reportDescription, sReportDescription);
}
```

```
void Report::setReportID(int sReportID){
    reportID = sReportID;
}

void Report::setDescription(const char sReportDescription[]){
    strcpy(reportDescription, sReportDescription);
}

int Report::getReportID(){
    return reportID;
}

char *Report::getDescription(){
    return reportDescription;
}

void Report::displayReport(){
    cout << "Report ID: " << reportID << endl;
    cout << "Report Description: " << reportDescription << endl;
}

Report::~Report(){{
    cout << "Report is deleted!" << endl;
}}
```

Review.h

```
/*
    IT Number: IT21162978
*/

#pragma once
class Service;

class Review{
    private:
        int reviewID;
        char subject[100];
        char description[200];

    public:
        Review(int sReviewID, const char sSubject[], const char sDescription[]);
```

```
        void setReviewID(int sReviewID);
        void setSubject(const char sSubject[]);
        void setDescription(const char sDescription[],Service *service1);
        int getReviewID();
        char *getSubject();
        char *getDescription();
        void viewReview();
        ~Review();
};
```

Review.cpp

```
/*
   IT Number: IT21162978
*/

#include <iostream>
#include "Review.h"
#include "Service.h"
#include <cstring>

using namespace std;

Review::Review(int sReviewID, const char sSubject[], const char sDescription[]){
   reviewID = sReviewID;
   strcpy(subject,sSubject);
   strcpy(description,sDescription);
}
void Review::setReviewID(int sReviewID){
   reviewID=sReviewID;
}

void Review::setSubject(const char sSubject[]){
   strcpy(subject, sSubject);
}

void Review::setDescription(const char sDescription[],Service *service1){
   service1->displayServiceDetails();
   strcpy(description,sDescription);
}

int Review::getReviewID(){
   return reviewID;
}

char *Review::getSubject(){
```

```
    return subject;
}

char *Review::getDescription(){
    return description;
}

void Review::viewReview(){
    cout <<"Review : " <<description <<endl;
}

Review::~Review(){
    cout <<"Deleted Review ID " << reviewID <<endl;
}
```

Service.h

```
/*
    IT Number: IT21160516 and IT21161742
*/

#pragma once
#include "Payment.h"
#include "Registered_User.h"
#include "Examination.h"

#define SIZE3 2

class Driving_School;
class Payment;
class Registered_User;
class Examination;

class Service{
    private:
        int serviceID;
        char serviceType[50];
        float serviceFee;
        Payment *payment;
        Registered_User *registered_user[SIZE3];
        Examination *exam;
    public:
        Service();
        Service(int sServiceID,const char sServiceType[], float sServiceFee );
        void setServiceID(int sServiceID);
        void setServiceType(const char sServiceType[]);
        void setServiceFee(float sServiceFee);
        int getServiceID();
```

```
        char *getServiceType();
        void displayServiceDetails();
        float getServiceFee();
        void setRegisteredUser(Registered_User *registeredUser1, Registered_User
*registeredUser2);
        void displayRegisteredUser();
        void setExamination(Examination *exam1);
        void viewExamDetails();
        ~Service();
};
```

Service.cpp

```
/*
   IT Number: IT21160516 and IT21161742
*/

#include <iostream>
#include <cstring>
#include "Service.h"

using namespace std;

Service::Service(){

}

Service::Service(int sServiceID,const char sServiceType[], float sServiceFee){
   serviceID = sServiceID;
   strcpy(serviceType,sServiceType);
   serviceFee=sServiceFee;
}

void Service::setServiceID(int sServiceID){
   serviceID = sServiceID;
}

void Service::setServiceType(const char sServiceType[]){
   strcpy(serviceType,sServiceType);
}

void Service::setServiceFee(float sServiceFee){
   serviceFee=sServiceFee;
}

int Service::getServiceID(){
   return serviceID;
}
```

```
char *Service::getServiceType(){
    return serviceType;
}

void Service::displayServiceDetails(){
    cout<<"Service ID : " << serviceID<<endl;
    cout<<"Service Type : " << serviceType<<endl;
    cout<<"Service Fee : " << serviceFee<<endl;
}

float Service::getServiceFee(){
    return serviceFee;
}

void Service::setRegisteredUser(Registered_User *registeredUser1, Registered_User
*registeredUser2){
    registered_user[0] = registeredUser1;
    registered_user[1] = registeredUser2;
}

void Service::displayRegisteredUser(){
    for(int i = 0; i < SIZE; i++){
        registered_user[i]->showRegisteredUser();
    }
}

void Service::setExamination(Examination *exam1){
    exam = exam1;
}

void Service::viewExamDetails(){
    exam->viewExamSchedule();
}

Service::~Service(){
    cout<<"Delete Service ID " << serviceID << endl;
}
```

User.h

```
/*
    IT Number: IT21160448
*/
#pragma once
class User {
    protected :
        char userName[30];
```

```
        char NIC[20];
        char DOB[20];
        char address[100];
        long int contactNo;
        char email[30];
    public :
        User();
        User(const char sUserName[], const char sNIC[], const char sDOB[], const char sAddress[],
long int sContactNo, const char sEmail[]);
        void setName(const char sUserName[]);
        void setEmail(const char sEmail[]);
        void setContactNo(long int sContactNo);
        void setAddress(const char sAddress[]);
        void setDOB(const char sDOB[]);
        char *getEmail();
        long int getContactNo();
        char *getAddress();
        char *getDOB();
        void getDetails();
        void updateDetails();
        ~User();
};
```

User.cpp

```
/*
   IT Number: IT21160448
*/
#include <iostream>
#include <cstring>
#include "User.h"
using namespace std;

User::User(){

}

User::User(const char sUserName[], const char sNIC[], const char sDOB[], const char sAddress[],
long int sContactNo, const char sEmail[]){
   strcpy(userName, sUserName);
   strcpy(NIC, sNIC);
   strcpy(DOB, sDOB);
   strcpy(address, sAddress);
   contactNo = sContactNo;
   strcpy(email, sEmail);
}
```

```
void User::setName(const char sUserName[]){
    strcpy(userName, sUserName);
}

void User::setEmail(const char sEmail[]){
    strcpy(email, sEmail);
}

void User::setContactNo(long int sContactNo){
    contactNo = sContactNo;
}

void User::setAddress(const char sAddress[]){
    strcpy(address, sAddress);
}

void User::setDOB(const char sDOB[]){
    strcpy(DOB, sDOB);
}

char *User::getEmail(){
    return email;
}

long int User::getContactNo(){
    return contactNo;
}

char *User::getAddress(){
    return address;
}

char *User::getDOB(){
    return DOB;
}

void User::getDetails(){
    cout << "Username: " << userName << endl;
    cout << "NIC: " << NIC << endl;
    cout << "DOB: " << DOB << endl;
    cout << "Address: " << address << endl;
    cout << "Contact Number: " << contactNo << endl;
    cout << "Email: " << email << endl;
}

void User::updateDetails(){
```

```
}

User::~User(){
    cout << "User is deleted!" << endl;
}
```

Individual Contribution.

	Student ID	Student Name	Individual Contribution
01	IT21160448	Akalanka P.A.A	**CRC** • User • Registered User • Driving School Advertiser **CODE** 1. Main.cpp 2. User.h 3. User.cpp 4. RegisteredUser.h 5. RegisteredUser.cpp 6. DrivingSchoolAdvertiser.h 7. DrivingSchoolAdvertiser.cpp
02	IT21162978	Athukorala W. A. A. D. D. T.	**CRC** • Agent • Feedback • Review **CODE** 1. Main.cpp 2. Agent.h 3. Agent.cpp 4. Feedback.h 5. Feedback.cpp 6. Review.h 7. Review.cpp
03	IT21160202	Wijekoon W. M. S. R.	**CRC** • Report • Course • Driving School **CODE** 1. Main.cpp 2. Report.h 3. Report.cpp 4. Course.h 5. Course.cpp 6. Driving_School.h 7. Driving_School.cpp
04	IT21160516	Chamikara T. A. D. G.	**CRC** • Payment • Service

			CODE 1. Main.cpp 2. Payment.h 3. Payment.cpp 4. Service.h 5. Service.cpp
05	IT21161742	Kumari P. D. L.	**CRC** • Examination • Service **CODE** 1. Main.cpp 2. Examination.h 3. Examination.cpp 4. Service.h 5. Service.cpp