Topic : Online Fitness Trainer

Group no : MLB_09.02_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.

	Student ID	Student Name	Email	Contact Number
1	IT21341236	Rahubadda I.P.	it21341236@my.sliit.lk	0701263969
2	IT21328916	De Silva K.T.S.	it21328916@my.sliit.lk	0772021034
3	IT21326240	Athapaththu A.M.S.C.	it21326240@my.sliit.lk	0714782241
4	IT21296932	Silva T.C.D.	it21296932@my.sliit.lk	0717226025
5	IT21305696	Gunarathne M.M.S.U.	it21305696@my.sliit.lk	0716724148

Table of Contents

System Requirements

- This system should be functioning 24/7/365.
- Non-Registered users should be able to browse through the system and get all the details about packages and schedules.
- To purchase a package or to get any fitness advises the non-registered user must register by providing details such as name, address, Email, NIC, and contact number.
- Registered customers can log in to the system and for that they should use the correct username and the password.
- Registered customers can purchase packages, and schedule available time slots which are comfortable for them.
- Trainers should be able to add details such as types of packages, prices of packages, available time slots and what they offer in relevant packages.
- All the details should be validated by the administrator.
- Administrator should be able to update the system about packages, prices, and available time slots.
- All the details must be updated in a database.
- Registered customers must enter their payment types, and further details such as, card number, card holder's name and cvc number to purchase a package.
- After the registered customer do the payment, the payment must be validated by the bank or other trusted resources, and the customer should get a payment verification email with the details of the purchased package.

Noun & Verb Analysis (Nouns)

- This system should be functioning 24/7/365.
- Non-Registered users should be able to browse through the system and get all the details about packages and schedules.
- To purchase a package or to get any fitness advises the non-registered user must register by providing details such as name, address, Email, NIC, and contact number.
- Registered customers can log in to the system and for that they should use the correct username and the password.
- Registered customers can purchase packages, and schedule available timeslots which are comfortable for them.
- Trainers should be able to add details such as types of packages, prices of packages, available timeslots and what they offer in relevant packages.
- All the details should be validated by the administrator.
- Administrator should be able to update the system about packages, prices, and available timeslots.
- All the details must be updated in a database.
- Registered customers must enter their payment types, and further details such as, card number, card holder's name and cvc number to purchase a package.
- After the registered customer do the payment, the payment must be validated by the bank or other trusted resources, and the customer should get an email verifying the payment, with the details of the purchased package.

Identified Classes

- Non-Registered user
- Registered user
- Package
- Trainer
- Schedule
- Administrator
- Database
- Payment

Reasons for rejecting other nouns

1. **Redundant**
2. **An event or an operation**
3. **Outside scope of system –** system, bank
4. **Meta-Language -** they
5. **An attribute –** details, name, address, email, contact number, NIC, username, password, types, prices, card number, card holder's name, cvv number

Noun & Verb Analysis (Verbs)

- This system should be functioning 24/7/365.
- Non-Registered users should be able to browse through the system and get all the details about packages and schedules.
- To purchase a package or to get any fitness advises the non-registered user must register by providing details such as name, address, Email, NIC, and contact number.
- Registered customers can log in to the system and for that they should use the correct username and the password.
- Registered customers can purchase packages, and schedule available time slots which are comfortable for them.
- Trainers should be able to add details such as types of packages, prices of packages, available time slots and what they offer in relevant packages.
- All the details should be validated by the administrator.
- Administrator should be able to update the system about packages, prices, and available time slots.
- All the details must be updated in a database.
- Registered customers must enter their payment types, and further details such as, card number, card holder's name and cvc number to purchase a package.
- After the registered customer do the payment, the payment must be validated by the bank or other trusted resources, and the customer should get a payment verification email with the details of the purchased package.

Methods

- Non-Registered user - Browse the system
  Register to the system by providing details
- Registered user - Log in to the system using credentials
  Purchase packages
  Add payment details
- Package - Generate package ID
- Trainer - Log in to the system using credentials
  Add package details
  Add schedule details
- Schedule - Update timeslots
- Administrator - Validate log in details
  Validate package, and schedule details
  Update package and schedule details
- Database - Update details about registered customers
  Update details about packages and schedules
- Payment - Check payment details
  Confirm payment details

CRC Cards

Non-Registered User	
Responsibilities	**Collaborations**
Browse the system	
Register to the system by providing details	

Registered User	
Responsibilities	**Collaborations**
Log in to the system using credentials	
Purchase packages	Package
Add payment details	Payment

Package	
Responsibilities	**Collaborations**
Generate package ID	Package

Trainer	
Responsibilities	**Collaborations**
Log in to the system using credentials	
Add package details	Package
Add schedule details	Schedule

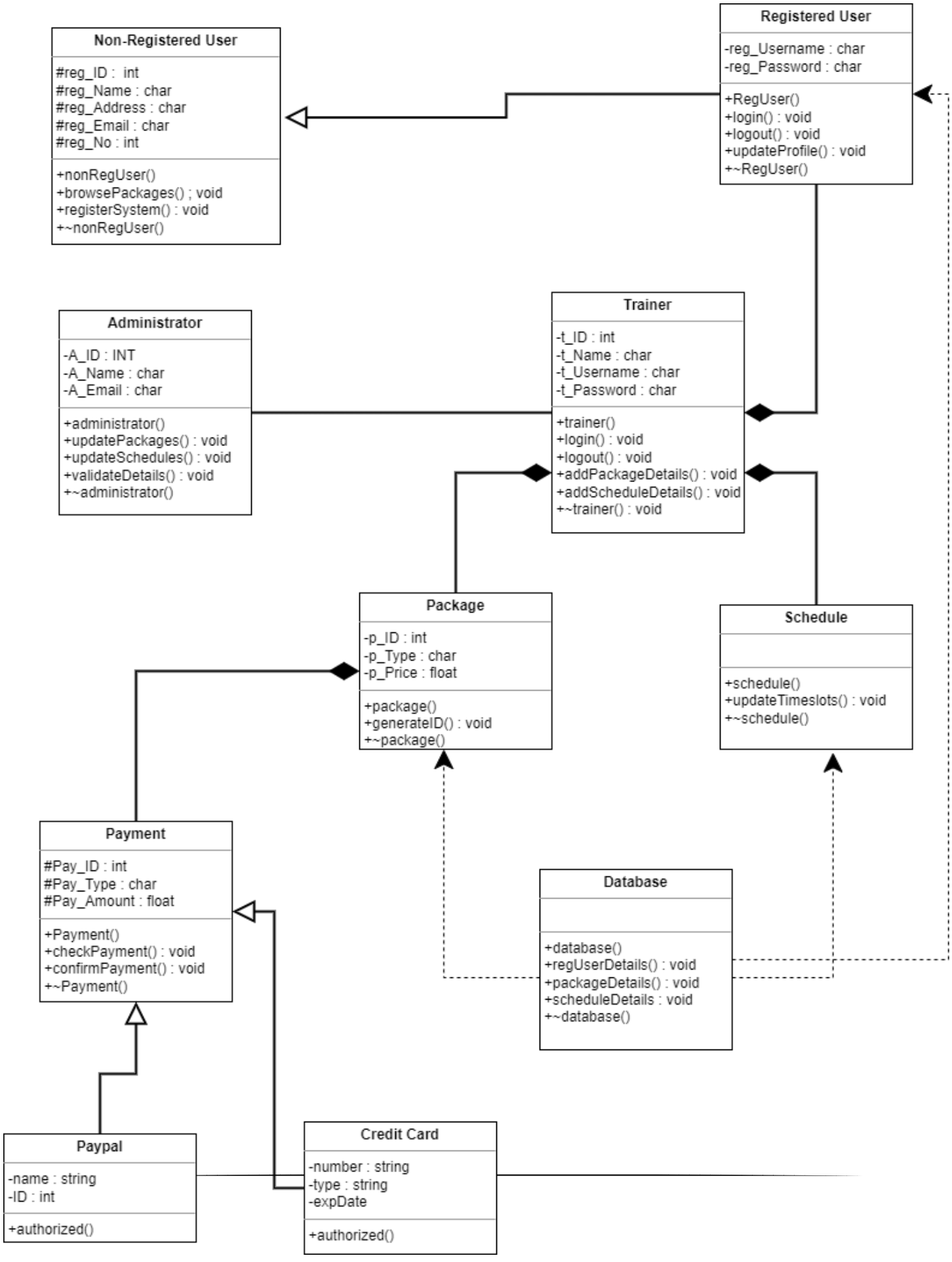
Schedule	
Responsibilities	**Collaborations**
Update timeslots	

Administrator	
Responsibilities	**Collaborations**
Validate log in details	
Validate package, and schedule details	Package, Schedule
Update package and schedule details	Package, Schedule

Database	
Responsibilities	**Collaborations**
Update details about registered customers	Registered user
Update details about packages and schedules	Package, Schedule

Payment	
Responsibilities	**Collaborations**
Check payment details	Payment
Confirm payment details	Payment

Class Diagram (UML Notation)



Class Header Files

Nonreguser.h

```
#include <iostream>
#include <cstring>

using namespace std;

class nonRegUser {
protected:
    int regid;
    char regname[20];
    char regaddress[50];
    char regemail[30];
    int regno;
public:
    nonRegUser();
    nonRegUser(int rid, char rname[], char raddress[], char remail[], int rno);
    void browsepackages();
    void registersystem();
    ~nonRegUser();


};

//register user
```

reguser.h

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

class RegUser :public nonRegUser {
private:
    DataBase* database;
    char regUsername[20];
    char regPassword[20];
public:
    RegUser();
    RegUser(char rusername[], char rpassword[]);
    void login();
```

```
    void logout();
    void updateprofile();
    ~RegUser();

};
```

//Payment

payment.h

```
#include <cstring>

class Payment {
protected:
    int Payid;
    char Paytype[15];
    float Payamount;
public:
    Payment();
    Payment(int* pid, char* ptype[], float* pa);
    bool checkpayment();
    void confirmpayment();
    ~Payment();
};
```

//Paypal

paypal.h

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

class Paypal :public Payment {
private:
    char Name[20];
    int Id;
public:
    Paypal(int* pid, char* ptype[], float* pa, char n[], int i);
    bool authorized();
    ~Paypal();

};
```

//credit card

creditcard.h

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

class CreditCard :public Payment {
private:
    char CardNumber[17];
    char Ccord[10];
    char exdate[10];
public:

    CreditCard(int* pid, char* ptype[], float* pa, char cnumber[], char cc[],char edate[]);
    bool authorized();
    ~CreditCard();
};
```

//package

package.h

```
#include <cstring>
#include "database.h"
#include "payment.h
class Package {
private:
    DataBase* database;
    Payment* pay[1];
    int packid;
    char packtype[20];
    float packprice;
public:
    Package(int paid,char patype[],float paprice);
    void generateid();
    ~Package();
};
```

//Trainer

trainer.h

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

```
#include "package.h
#include "reguser.h"
#include "shedule.h



class Trainer {
    Administrator* admin[1];
    Package* pack[1];
    RegUser* regu[1];
    Schedule* schedu[1];
    int trainerid;
    char trainername[20];
    char traineruserna[20];
    char trainerpassword[10];

public:
    Trainer();
    Trainer(int tid,char tna[],char tuna[],char tpass[]);
    void login(RegUser* regu[]);
    void logout();
    void addpackagedetails(Package* pack[]);
    void addscheduledetails();
    ~Trainer();

};


//Administrator
```

administrator.h

```
#include <cstring>
#include "trainer.h"
class Administrator {
private:
    Trainer* trainer;
    int adminid;
    char adminname[20];
    char adminemail[30];

public:
    Administrator(int aid, char ana[], char aemail[]);
    void updatePackages();
```

```
    void updateschedules();
    void validatedetails();
    ~Administrator();
};
```

```
//Schedule
```

schedule.h

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

class Schedule {
private:
    DataBase* database;
public:
    Schedule();
    void updatetimeslots();
    ~Schedule();
};
```

```
//DataBase
```

database.h

```
#include <cstring>

class DataBase {
public:
    DataBase();
    void reguserdetails();
    void packagedeteils();
    void scheduledetails();
    ~DataBase();

};
```

Class Cpp Files

Nonreguser.cpp

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

nonRegUser::nonRegUser() {
    regid = 0;
    strcpy(regname,"");
    strcpy(regaddress,"");
    strcpy(regemail,"");
    regno=0;
}
nonRegUser::nonRegUser(int rid, char rname[], char raddress[], char remail[], int rno)
{
    regid = rid;
    strcpy(regname,rname);
    strcpy(regaddress,raddress );
    strcpy(regemail,remail);
    regno = rno;

}
void nonRegUser::browsepackages() {

}
void nonRegUser::registersystem() {

}
nonRegUser::~nonRegUser() {
    cout << "non register user deleted" << endl;
}
```

reguser.cpp

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


RegUser::RegUser() {
    strcpy(regUsername, "");
    strcpy(regPassword, "");
```

```
}
RegUser::RegUser(char rusername[],char rpassword[]) {
    strcpy(regUsername, rusername);
    strcpy(regPassword, rpassword);
}
void RegUser::login() {

}
void RegUser::logout() {

}
void RegUser::updateprofile() {

}
RegUser::~RegUser() {
    cout << "register user delete" << endl;
}
```

payment.cpp

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


Payment::Payment() {
    Payid = 0;
    strcpy(Paytype, "");
    Payamount = 0;
}
Payment::Payment(int* pid, char* ptype[], float* pa) {
    Payid = *pid;
    strcpy(Paytype,*ptype);
    Payamount = *pa;

}
bool Payment::checkpayment() {
    return false;
}
void Payment::confirmpayment() {

}
Payment::~Payment() {
```

```
    cout << "Payment delete" << endl;
}
```

paypal.cpp

```
#include <cstring>
#include "paypal.h"
#include "payment.h"


Paypal::Paypal(int* pid, char* ptype[], float* pa, char n[], int i):Payment(pid,ptype,pa) {
    strcpy(Name, n);
    Id = i;
    cout << "constructor executed" << endl;
}
bool Paypal::authorized() {
    return false;

}
Paypal::~Paypal() {
    cout << "Paypal delete" << endl;
}
```

creditcard.cpp

```
#include <cstring>
#include "creditcard.h"
#include "payment.h"

CreditCard::CreditCard(int* pid, char* ptype[], float* pa, char cnumber[], char cc[],
char edate[]):Payment(pid,ptype,pa) {
    strcpy(CardNumber, cnumber);
    strcpy(Ccord, cc);
    strcpy(exdate, edate);
    cout << "constructor executed" << endl;

}
bool CreditCard::authorized() {
    return false;
}
```

```
CreditCard::~CreditCard() {
    cout << "Creditcard delete" << endl;
}
```

package.cpp

```
#include <cstring>
#include "package.h"
#include "payment.h

Package::Package(int paid, char patype[], float paprice) {
    packid = paid;
    strcpy(packtype, patype);
    packprice = paprice;

}
void Package::generateid() {

}
Package::~Package() {
    cout << "package delete" << endl;
    for (int i = 0; i < 1; i++) {
        delete pay[i];
    }
}
```

trainer.cpp

```
#include <cstring>
#include "trainer.h"
#include "administrator.h"
#include "package.h
#include "reguser.h"
#include "shedule.h


Trainer::Trainer() {
    trainerid = 0;
    strcpy(trainername, "");
    strcpy(traineruserna, "");
```

```
    strcpy(trainerpassword, "");

}
Trainer::Trainer(int tid, char tna[], char tuna[], char tpass[]) {
    trainerid = tid;
    strcpy(trainername,tna);
    strcpy(traineruserna,tuna);
    strcpy(trainerpassword,tpass);

}
void Trainer::login(RegUser* regu[]) {

}
void Trainer::logout() {

}
void Trainer::addpackagedetails(Package* pack[]) {

}
void Trainer::addscheduledetails() {

}
Trainer::~Trainer() {
    cout << "Trainer delete" << endl;
    for (int i = 0; i < 1; i++) {
        delete admin[i];
        delete pack[i];
        delete regu[i];
        delete schedu[i];
    }

}
```

administrator.cpp

```
#include <cstring>
#include "trainer.h"
#include "administrator.h"

Administrator::Administrator(int aid, char ana[], char aemail[]) {
    adminid = aid;
    strcpy(adminname, ana);
```

```
    strcpy(adminemail, aemail);

}
void Administrator::updatePackages() {

}
void Administrator::updateschedules() {

}
void Administrator::validatedetails() {

}
Administrator::~Administrator() {
    cout << "administrator delete" << endl;

}
```

schedule.cpp

```
#include <cstring>
#include “schedule.h”
#include “database.h”


Schedule::Schedule() {
    DataBase* database;
    cout << "constructor executed" << endl;
}
void Schedule::updatetimeslots() {

}
Schedule::~Schedule() {
    cout << "schedule delete" << endl;
}
```

database.cpp

```
#include <cstring>
```

```
#include “database.h

DataBase::DataBase() {

}
void DataBase::reguserdetails() {

}
void DataBase::packagedeteils() {

}
void DataBase::scheduledetails() {

}
DataBase::~DataBase() {
    cout << "database delete " << endl;
```