Topic : Online Fitness Trainer

Group no : MET_WD_06

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

Submission Date : 20/05/2022

We declare that this is our own work and this Assignment does not incorporate without acknowledgment any material previously submitted by anyone else in SLIIT or any other university/Institute. And we declare that each one of us equally contributed to the completion of this Assignment.

Registration No	**Name**	**Contact Number**
IT21264634	Sujitha.S	0742065382
IT21068478	Dharane.S	0778893615
IT21333170	Kaneson.K	0769638330
IT21369438	Fernando H.M.D	0789399405
IT21357244	Weerarathne M.A.D.M.S	0710567026

System Requirements for Online Fitness Trainer

- "Fit to be" Online trainer system allows users to be trained online and stay fit.
- Unregistered users can use online store and inspect basic fitness plans.
- To use online store as an unregistered user, the user should provide their full name, contact number, address, and email to the system.
- To be a registered user, additionally unregistered user must provide dob to calculate their age.
- System generates unique login id in the registrations process, which the registered users should use when they login to the system with their password.
- The system allows registered users to choose and enroll into fitness plans, diet plans, choose a fitness trainer, and request a nutritionist as preferred.
- The system keeps records about fitness plans and diet plans. Namely identification code, name, type, and price
- Registered users are allowed to schedule time for their chosen fitness plan.
- Diet plans are allocated for the registered users according to their age group.
- The system allows nutritionists and fitness trainers to join and provide their services to the registered users.
- In the registration process, nutritionist and fitness trainers must provide their name, email, address, contact number and qualifications to the system.
- System generates unique id for both fitness trainers and nutritionists after registration.
- Fitness trainers can create a profile and upload sample workout videos using the system.
- Nutritionists can create a profile and upload nutrition plans.
- The system keeps record about equipment and supplements in the inventory. Namely stock id, name, and price.
- Count of supplements and equipment are recorded separately.
- Customers are asked to confirm the order details when they buy products from the system.
- After finalizing, the payment of the order consists unique transaction id, quantity, amount and type.

- The administrators will be asked for their name, email, contact number and address when joining to the system while system provides them with a unique id.
- The system allows system administrators to generate reports (weekly, monthly, quarterly, annual), maintain inventory (add, update, or delete inventory items) and to change selected trainer and scheduled time according to the requests made by users.
- Administrator generate reports based on sales, order and payment.

Identified Classes

User

Unregistered User

Registered User

Plans

Fitness Plans

Diet Plans

Inventory

Equipment

Supplements

Payment

Fitness Trainers

Nutritionists

Administrator

Report

CRC CARDS

Unregistered user

Responsibility	Collaborations
Overview the system	
Buy supplements & equipment	Inventory
Make payment	Payment
Register to the system	Registered user
Inspect basic fitness plans	

Registered user

Responsibility	Collaborations
Login to the system	
Buy diet plans and fitness plans	plans
Select trainer	trainer
Select schedule time	
Request nutritionist	nutritionist

Plans

Responsibility	Collaborations
Keep the records of plans	Trainer, nutritionist ,administrator
Calculate price	

Fitness plans

Responsibility	Collaborations
Assign	trainer
Display details	

Diet plans

Responsibility	Collaborations
Assign	nutritionist
Display details	

Inventory

Responsibility	Collaborations
Check stock availability	
Store details	
Calculate sold stock	

Supplements

Responsibility	Collaborations
Display details	

Equipment

Responsibility	Collaborations
Display details	

Payment

Responsibility	Collaborations
Generate transaction id	
Check payment details	administrator
Confirm payment details	user

Fitness trainers

Responsibility	Collaborations
Enroll to the system	
Create profile	
Upload sample workout videos	

Nutritionist

Responsibility	Collaborations
Enroll to the system	
upload nutrition plans	diet plans
create profile	

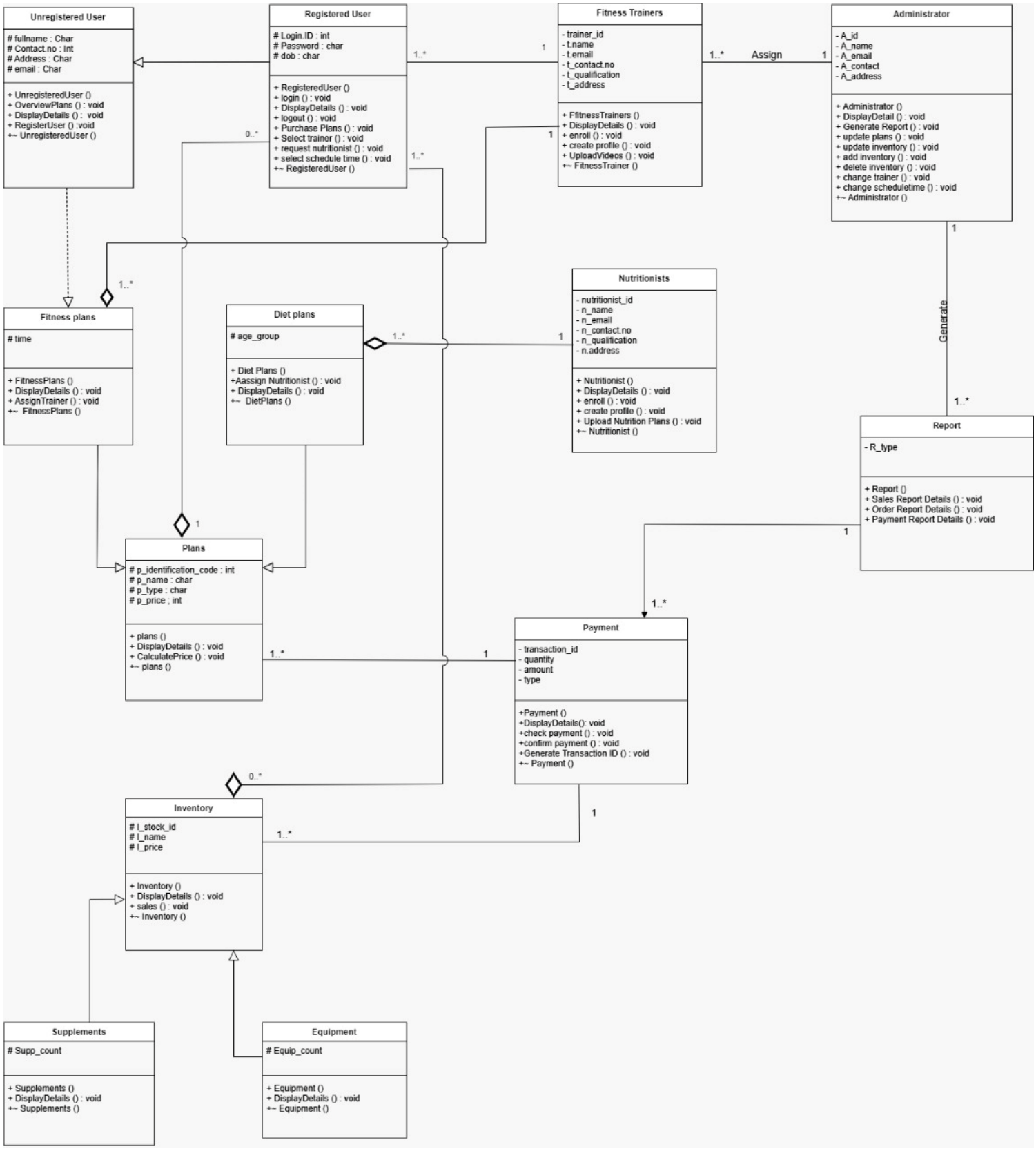
Administrator

Responsibility	Collaborations
generate reports	
update plan details	fitness plans, diet plans
maintain inventory	supplements, equipment
change trainer	trainer, registered customer
change schedule time	

Report

Responsibility	Collaborations
Generate selling details	
Generate ordering details	
Generate payment details	payment

Class Diagram (UML Notation)



Class Header Files

1.UnregisteredUser.h

```
//Dependency with fitnessPlans
//Inheritance with RegisteredUser

#include "fitnessPlans.h"

class UnregisteredUser
{
	protected:
		char fullName[20];
		int contactNo;
		char address[200];
		char email[50];
	public:
		UnregisteredUser();
		UnregisteredUser( char u_fname[], char u_address[], char u_email[],int
u_cno );
   void overViewPlans();
		void addfitnessPlans (fitnessPlans *f);
		void overviewPlans();
		void DisplayDetails();
		void registerUser();
		~UnregisteredUser();
};
```

2.RegisteredUser.h

```
//Inheritance with UnregisteredUser
//Association with FitnessTrainers
#include "FitnessTrainers.h"
class RegisteredUser : public UnregisteredUser
{
	protected:
		int login_id;
		char password[10];
		char dob[15];
		FitnessTrainer *ft;
	public:
		RegisteredUser();
		RegisteredUser( int r_lid, char r_pword[], char r_dob[], FitnessTrainer *ftr );
		void login();
		void displayDetails();
```

```
                void logout();
                void purchasePlans();
                void selectTrainer();
                void requestNutritionist();
                void scheduleTime();
                ~RegisteredUser();
};
```

3. Administrator.h

```
#include "Report.h"
#include "FitnessTrainers.h"
#pragma once
//Uni directional Association with Fitnesstrainers
//Unidirectional association with Report

class Administrator
{
        private:
                char a_name[20];
                int a_id;
                char a_email[50];
                int a_contact;
                char a_address[100];
                Report *rep[100];
                FitnessTrainers *fitness[10];
        public:
                Administrator();
                Administrator(char admin_name[],int admin_id,char admin_email[],int
admin_contact,char admin_address[]);
                void displayDetails();
                void generateReports(Report *Repo);
                void updatePlans();
                void updateInventory();
                void addupInventory();
                void deleteInventory();
                void changeTrainer();
                void changeScheduleTime();
                void addTrainer ( FitnessTrainers *f );
                void addReport ( Report *r );
                ~Administrator();
};
```

4.Plans.h

```
#include "Plans.h"
#include"Payment.h"

//Inheritance with fitnessPlans
//Inheritance with dietPlans
//Association with Payment
//Aggregation with RegisteredUser
class Plans{
	protected:
		int p_identificationCode;
		char p_name[20];
		char p_type[20];
		int p_price;
		Payment *pay;
		RegisteredUser *rg [ 5 ];
	public:
		Plans();
		Plans(int plans_identificationcode,char plans_name[],char plans_type[],int
plans_price);
		void addRegisteredUser (RegisteredUser *rg1, RegisteredUser *rg2,
RegisteredUser *rg3, RegisteredUser *rg4, RegisteredUser *rg5);
		void displayDetails();
		void calculatePrice();
		~Plans();
};
```

5.fitnessPlans.h

```
#pragma once
#include "FitnessTrainers.h"
#include "Plans.h"
//Inheritance between Plans and fitnessPlans
//Dependency between UnregisteredUser and fitnessPlans
//Aggregation between FitnessTrainers and fitnessPlans


class fitnessPlans : public Plans
{
	protected:
		int time;
		FitnessTrainers *fitTrainer[5];
```

```
	public:
		fitnessPlans();
		fitnessPlans(int fitnessPlans_time);
		void displayDetails();
		void addtrainer ( FitnessTrainers *tra1, FitnessTrainers *tra2, FitnessTrainers
*tra3, FitnessTrainers *tra4, FitnessTrainers *tra5);
		void assignTrainer();
		~fitnessPlans();
};
```

6.dietPlans.h

```
#pragma once
#include "Nutritionists.h"
#include "Plans.h"
//Inheritance with Plans
//Aggregation with Nutritionists



class dietPlans : public Plans
{
	protected:
		int age_group;
		Nutritionists *nutri[5];
	public:
		dietPlans();
		dietPlans(int d_ageGroup);
		void addNutritionist(Nutritionists *nutri1,Nutritionists *nutri2,Nutritionists
*nutri3,Nutritionists *nutri4,Nutritionists *nutri5);
		void assignNutritionist();
		void displayDetails();
		~dietPlans();
};
```

7.Inventory.h

```
//Association between payment
//Inheritance with Equipments
//Inheritance with Supplements
//Aggregation with RegisteredUser
```

```
#pragma once
#include "Payment.h"
class Inventory
{
	protected:
		int stock_id;
		char stock_name[20];
		int stock_price;
		Payment *pay1;
		RegisteredUser *RG [5];
	public:
		Inventory();
		Inventory(int id_stock,char name_stock[],int price_stock);
		void displayDetails();
		void addRegisteredUser ( RegisteredUser *RG1, RegisteredUser *RG2,
RegisteredUser *RG3, RegisteredUser *RG4, RegisteredUser *RG5);
		void sales();
		~Inventory();
};
```

8.Equipments.h

```
#pragma once
#include "Inventory.h"
//Inheritance with Inventory

class Equipments : public Inventory
{
	protected:
		int equip_count;
	public:
		Equipments();
		Equipments(int equipments_count);
		void displayDetails();
		~ Equipments();
};
```

9.Supplements.h

```
//Inheritance with Inventory

#include "Inventory.h"

class Supplements : public Inventory
{
	protected:
		int supp_count;
	public:
		Supplements();
		Supplements(int supplements_count);
		void displayDetails();
		~Supplements();
};
```

10.FitnessTrainers.h

```
//Aggregation with fitnessPlans
//Association with Administrator
//Association with RegisteredUser

#pragma once
#include "Administrator.h"
#include "RegisteredUser.h"

class FitnessTrainers
{
	private:
		char t_name[20];
		char t_email[50];
		int t_id;
		char t_qualification[100];
		char t_address[100];
		Administrator *admini;
		RegisteredUser *Reg[10];
	public:
		FitnessTrainers();
		FitnessTrainers(char trainerName[],char trainerEmail[],int trainerId,char
trainerQualification[],char trainerAddress[], Administrator *admin);
		void displayDetails();
```

```
                void enroll();
                void createProfile();
                void uploadVideos();
                void addRegisteredUser( RegisteredUser *r);
                ~FitnessTrainers();
};
```

11.Nutritionists.h

```
//Aggregation with dietPlans

#ipragma once
class Nutritionists
{
        private:
                char n_name[20];
                char n_email[50];
                int n_id;
                char n_qualification[100];
                char n_address[100];
        public:
                Nutritionists();
                Nutritionists(char nutriName[],char nutriEmail[],int nutriId,char
nutriQualification[],char nutriAddress[]);
                void displayDetails();
                void enroll();
                void createProfile();
                void uploadNutritionPlans();
                ~Nutritionists();
};
```

12.Report.h

```
#include "Payment.h"
#include "Administrator.h"
//Association with Administrator
//Association with payment


class Report
{
        private:
                char r_type[20];
                Payment *pay;
                Administrator *Admin;
```

```
        public:
                Report();
                Report( char rtype[], Administrator *ad );
                void displayDetails();
                void salesReportDetails();
                void orderReportDetails();
                void paymentReportDetails();
                ~Report();
};
```

13.Payment.h

```
#pragma once
#include "Inventory.h"
#include "Plans.h"

//Unidirectional association with Plans
//Unidirectional association with Inventory
//Unidirectional association with Report

class Payment{
        private:
                int transaction_id;
                int quantity;
                int amount;
                char type[20];
                Plans *plan[10];
                Inventory *Invent[100];
        public:
                Payment();
                Payment(int trans_id,int count,int price,char pay_type[],char
plan_name[],char stock[]);
                void displayDetails();
                void checkPayment();
                void confirmPayment();
                void generateTransactionId();
                ~Payment();
};
```

Class Cpp Files

1.UnregisteredUser.cpp

```
#include "UnregisteredUser.h"
#include<cstring>
using namespace std;

UnregisteredUser::UnregisteredUser()
{
	strcpy( fullName,"");
	strcpy( address,"");
	strcpy( email,"");
	contactNo = 0;
}

UnregisteredUser::UnregisteredUser( char u_fname[], char u_address[], char u_email[],int
u_cno )
{
	strcpy(fullName,u_fname);
	strcpy(address,u_address);
	strcpy(email,u_email);
	contactNo = u_cno;
}

void UnregisteredUser::DisplayDetails()
{

}

void UnregisteredUser::overViewPlans()
{

}

void UnregisteredUser::registerUser()
{

}

UnregisteredUser::~UnregisteredUser()
{
	cout << "Destructor runs" << endl;

}
```

2.RegisteredUser.cpp

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

RegisteredUser::RegisteredUser()
{
	login_id = 0;
	strcpy(password,"");
	strcpy(dob,"");
}

RegisteredUser::RegisteredUser( int r_lid, char r_pword[], char r_dob[], FitnessTrainer *ftr)
{
	login_id = r_lid;
	strcpy(password,r_pword);
	strcpy(dob,r_dob);
}

void RegisteredUser::displayDetails()
{

}

void RegisteredUser::login()
{

}

void RegisteredUser::logout()
{

}

void RegisteredUser::purchasePlans()
{

}

void RegisteredUser::requestNutritionist()
{

}

void RegisteredUser::scheduleTime()
{
```

```
}

void RegisteredUser::selectTrainer()
{

}

RegisteredUser::~RegisteredUser()
{
	cout << "Destructor runs" << endl;

}
```

3. Administrator.cpp

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

Administrator::Administrator(){
	strcpy(a_name,"");
	a_id=0;
	strcpy(a_email,"");
	a_contact=0;
	strcpy(a_address,"");
}
Administrator::Administrator(char admin_name[],int admin_id,char admin_email[],int
admin_contact,char admin_address[]){
	strcpy(a_name,admin_name);
	a_id=admin_id;
	strcpy(a_email,admin_email);
	a_contact=admin_contact;
	strcpy(a_address,admin_address);
}

void Administrator::displayDetails(){

}
void Administrator::{

}
void Administrator::updatePlans(){

}
void Administrator::updateInventory(){
```

```
}
void Administrator::addupInventory(){

}
void Administrator::deleteInventory(){

}
void Administrator::changeTrainer(){

}
void Administrator::changeScheduleTime(){

}

void addTrainer ( FitnessTrainers *f ){

}
void addReport ( Report *r ){

}
Administrator::~Administrator(){
        cout << "Destructor runs" << endl;
}
```

4.Plans.cpp

```
#include "Plans.h"
#include<cstring>
using namespace std;

Plans::Plans()
{
        p_identificationCode=0;
        strcpy(p_name,"");
        strcpy(p_type,"");
        p_price=0;
}

Plans::Plans (int plans_identificationcode,char plans_name[],char plans_type[],int
plans_price)
{
        p_identificationCode=plans_identificationcode;
        strcpy(p_name,plans_name);
        strcpy(p_type,plans_type);
```

```
	p_price=plans_price;
}

void Plans::displayDetails()
{

}

void Plans::calculatePrice()
{

}

void Plans::addRegisteredUser (RegisteredUser *rg1, RegisteredUser *rg2, RegisteredUser
*rg3, RegisteredUser *rg4, RegisteredUser *rg5)
{
	rg[0] = rg1;
	rg[1] = rg2;
	rg[2] = rg3;
	rg[3] = rg4;
	rg[4] = rg5;

}

Plans::~Plans()
{
	cout << "Destructor runs" << endl;

}
```

5.fitnessPlans.cpp

```
#include "fitnessPlans.h"
#include<cstring>
using namespace std;

fitnessPlans::fitnessPlans()
{
	time=0;
}

fitnessPlans::fitnessPlans(int fitnessPlans_time)
{
   time=fitnessPlans_time;
}
```

```
void fitnessPlans::displayDetails()
{

}

void fitnessPlans::assignTrainer()
{

}

fitnessPlans::~fitnessPlans()
{
	cout<<"Destructor runs"<<endl;
}

void fitnessPlans::addtrainer ( FitnessTrainers *tra1, FitnessTrainers *tra2, FitnessTrainers
*tra3, FitnessTrainers *tra4, FitnessTrainers *tra5){
	fitTrainer[0] = tra1;
	fitTrainer[1] = tra2;
	fitTrainer[2] = tra3;
	fitTrainer[3] = tra4;
	fitTrainer[4] = tra5;
}
```

6.dietPlans.cpp

```
#include "dietPlans.h"
#include<cstring>
using namespace std;

dietPlans::dietPlans()
{
	age_group=0;
}

dietPlans::dietPlans(int d_ageGroup)
{
	age_group=d_ageGroup;
}

void dietPlans::assignNutritionist()
{

}
```

```
void dietPlans::displayDetails()
{

}

void dietPlans::addNutritionist(Nutritionists *nutri1,Nutritionists *nutri2,Nutritionists
*nutri3,Nutritionists *nutri4,Nutritionists *nutri5){
        nutri[0] = nutri1;
        nutri[1] = nutri2;
        nutri[2] = nutri3;
        nutri[3] = nutri4;
        nutri[4] = nutri5;
         }

dietPlans::~dietPlans()
{
        cout<<"Destructor runs"<<endl;
}
```

7.Inventory.cpp

```
#include "Inventory.h"
#include <cstring>
using namespace std;

Inventory::Inventory(){
        stock_id=0;
        strcpy(stock_name,"");
        stock_price=0;
}
Inventory::Inventory(int id_stock,char name_stock[],int price_stock){
        stock_id=id_stock;
        strcpy(stock_name,name_stock);
        stock_price=price_stock;
}
void Inventory::displayDetails(){

}
void Inventory::sales(){

}

void addRegisteredUser ( RegisteredUser *RG1, RegisteredUser *RG2, RegisteredUser *RG3,
RegisteredUser *RG4, RegisteredUser *RG5)
{
```

```
        RG[0] = RG1;
        RG[1] = RG2;
        RG[2] = RG3;
        RG[3] = RG4;
        RG[4] = RG5;

}
Inventory::~Inventory(){
        cout << "Destructor runs" << endl;
}
```

8.Equipments.cpp

```
#include "Equipments.h"
#include <cstring>
using namespace std;

Equipments::Equipments(){

        equip_count=0;

}
Equipments::Equipments(int equipments_count){

        equip_count=equipments_count;

}
void Equipments::displayDetails(){

}
Equipments::~Equipments(){

        cout << "Destructor runs" << endl;

}
```

9.Supplements.cpp

```
#include "Supplements.h"
#include <cstring>
using namespace std;
```

```
Supplements::Supplements(){

	supp_count=0;

}
Supplements::Supplements(int supplements_count){

	supp_count=supplements_count;

}
void Supplements::displayDetails(){

}
Supplements::~Supplements(){

	cout << "Destructor runs" << endl;

}
```

10.FitnessTrainers.cpp

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

FitnessTrainers::FitnessTrainers(){
	strcpy(t_name,"");
	strcpy(t_email,"");
	t_id=0;
	strcpy(t_qualification,"");
	strcpy(t_address,"");
}

FitnessTrainers::FitnessTrainers(char trainerName[],char trainerEmail[],int trainerId,char
trainerQualification[],char trainerAddress[], Administrator *admin){
	strcpy(t_name,trainerName);
	strcpy(t_email,"");
	t_id=trainerId;
	strcpy(t_qualification,trainerQualification);
	strcpy(t_address,trainerAddress);
}

void FitnessTrainers::displayDetails(){

}
```

```
void FitnessTrainers::enroll(){

}
void FitnessTrainers::uploadVideos(){

}
void FitnessTrainers::createProfile(){

}
FitnessTrainers::~FitnessTrainers(){
        cout << "Destructor runs" << endl;
}
```

11.Nutritionists.cpp

```
#include "Nutritionists.h"
#include <cstring>
using namespace std;

//Aggregation between dietPlans

Nutritionists::Nutritionists(){
        strcpy(n_name,"");
        strcpy(n_email,"");
        n_id=0;
        strcpy(n_qualification,"");
        strcpy(n_address,"");
}

Nutritionists::Nutritionists(char nutriName[],char nutriEmail[],int nutriId,char
nutriQualification[],char nutriAddress[]){
        strcpy(n_name,nutriName);
        strcpy(n_email,"");
        n_id=nutriId;
        strcpy(n_qualification,nutriQualification);
        strcpy(n_address,nutriAddress);
}

void Nutritionists::displayDetails(){

}
void Nutritionists::enroll(){

}
void Nutritionists::uploadNutritionPlans(){

}
```

```
void Nutritionists::createProfile(){

}
Nutritionists::~Nutritionists(){
	cout << "Destructor runs" << endl;
}
```

12.Report.cpp

```
#include "Report.h"
#include<cstring>
using namespace std;


Report::Report()
{
	strcpy(r_type,"");
}

Report::Report( char rtype[], Administrator *ad ){
	strcpy (r_type,rtype);
}

void Report::displayDetails()
{

}

void Report::orderReportDetails()
{

}

void Report::paymentReportDetails()
{

}

void Report::salesReportDetails()
{

}
```

```
Report::~Report()
{
	cout << "Destructor runs" << endl;
}
```

13.Payment.cpp

```
#include "Payment.h"
#include "Plans.h"
#include <cstring>
using namespace std;

Payment::Payment(){
	transaction_id=0;
	quantity=0;
	amount=0;
	strcpy(type,"");
}
Payment::Payment(int trans_id,int count,int price,char pay_type[],char plan_name[],char
stock[])
{
	transaction_id=trans_id;
	quantity=count;
	amount=price;
	strcpy(type,pay_type);
}
void Payment::displayDetails(){

}
void Payment::checkPayment(){

}
void Payment::confirmPayment(){

}
void Payment::generateTransactionId(){

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

Main program

Main.cpp

```
#include "UnregisteredUser.h"
#include "RegisteredUser.h"
#include "Report.h"
#include "Plans.h"
#include "fitnessPlans.h"
#include "dietPlans.h"
#include "Payment.h"
#include "FitnessTrainers.h"
#include "Nutritionists.h"
#include "Inventory.h"
#include "Supplements.h"
#include "Equipments.h"
#include "Administrator.h"
#include <iostream>
using namespace std;

int main(){
	//object creation

	RegisteredUser *rg = new RegisteredUser();
	Plans *fp = new fitnessPlans();
	Plans *dp = new dietPlans();
	Inventory *supp = new Supplements();
	Inventory *equip = new Equipments();
	fitnessPlans *fp1 = new fitnessPlans();
	FitnessTrainers *t1 = new FitnessTrainers();
	FitnessTrainers *t2 = new FitnessTrainers();
	FitnessTrainers *t3 = new FitnessTrainers();
	FitnessTrainers *t4 = new FitnessTrainers();
	FitnessTrainers *t5 = new FitnessTrainers();
	dietPlans *dp1 = new dietPlans();
	Nutritionists *n1 = new Nutritionists();
	Nutritionists *n2 = new Nutritionists();
	Nutritionists *n3 = new Nutritionists();
	Nutritionists *n4 = new Nutritionists();
	Nutritionists *n5 = new Nutritionists();



	fitnessPlans *fp2 = new fitnessPlans();
	UnregisteredUser *ur = new UnregisteredUser();

	Report *report = new Report;
```

```
        Payment *payment = new Payment;
        Administrator *admin = new Administrator;

        ur->addfitnessPlans();
        ur->DisplayDetails();
        ur->overviewPlans();
        ur->registerUser();
        fp1->addtrainer(t1,t2,t3,t4,t5);
        dp1->addNutritionist(n1,n2,n3,n4,n5);
        fp->displayDetails();
        dp->displayDetails();
        supp->displayDetails();
        equip->displayDetails();
        fp1->displayDetails();
fp1->assignTrainer();
dp1->displayDetails();
t1->createProfile();
t1->displayDetails();
t1->enroll();
t1->uploadVideos();
n1->createProfile();
        n1->displayDetails();
        n1->enroll();
        n1->uploadNutritionPlans();
        report->displayDetails();
        report->orderReportDetails();
        report->paymentReportDetails();
        report->salesReportDetails();
        payment->checkPayment();
        payment->confirmPayment();
        payment->displayDetails();
        payment->generateTransactionId();
        admin->addReport();
        admin->addTrainer();
        admin->addupInventory();
        admin->changeScheduleTime();
        admin->changeTrainer();
        admin->deleteInventory();
        admin->displayDetails();
        admin->generateReports();
        admin->updateInventory();
        admin->updatePlans();
        ur->registerUser();
        ur->DisplayDetails();
        rg->login();
        rg->logout();
        rg->purchasePlans();
```

```
rg->registerUser();
rg->requestNutritionist();
rg->scheduleTime();
rg->selectTrainer();

delete rg;
delete fp;
delete dp;
delete supp;
delete equip;
delete dp1;
delete fp1;
delete fp2;
delete ur;
delete report;
delete payment;
delete admin;

}
```