Sri Lanka Institute of Information Technology

Assignment

MLB 05.02_01

Online Pet Care System

Information System and Data Modeling – IT1090

B.Sc. (Hons) in Information Technology

Assignment Cover Sheet

Topic : Online Pet Care System

Group no : Malabe 05.02_01

Campus : Malabe

Submission Date: 17/05/2022

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**
IT21299902	ZAKEY.M.S.M.A.	0701563283
IT21299452	DILHARA.W.M.A.	0772229907
IT21302480	DILSHAN.O.A.P.	0772930198
IT21301018	PADUKKA.P.V.G.G.	0710748638
IT21301704	DE SILVA.L.M.C.	0712761301

Description

The Online Pet care system facilitates people to acquire quality Pet care products at a convenient price and get assistance in the most important pet care services. The users can also make complaints and donations to protect our adorable animals. Furthermore, the system allows the users to get their products delivered to their doorstep within a short period. Both registered and non-registered customers can access our services and visit our pet care system.

Also, the system enables the users to check the availability of a vet and their desired products and make online and card payments. The objective of this system is to design an Online pet care system that would deliver quality service to our clients, allow the user to save their valuable time, get the exposure to a wide variety of pet products and make it more convenient for the clinical services through online payment methods.

Thus, this system is a helping hand to all the owners of four-legged animals to take care of their pets without any inconvenience with the assistance of our experienced team who considers the clients' little friends as their own.

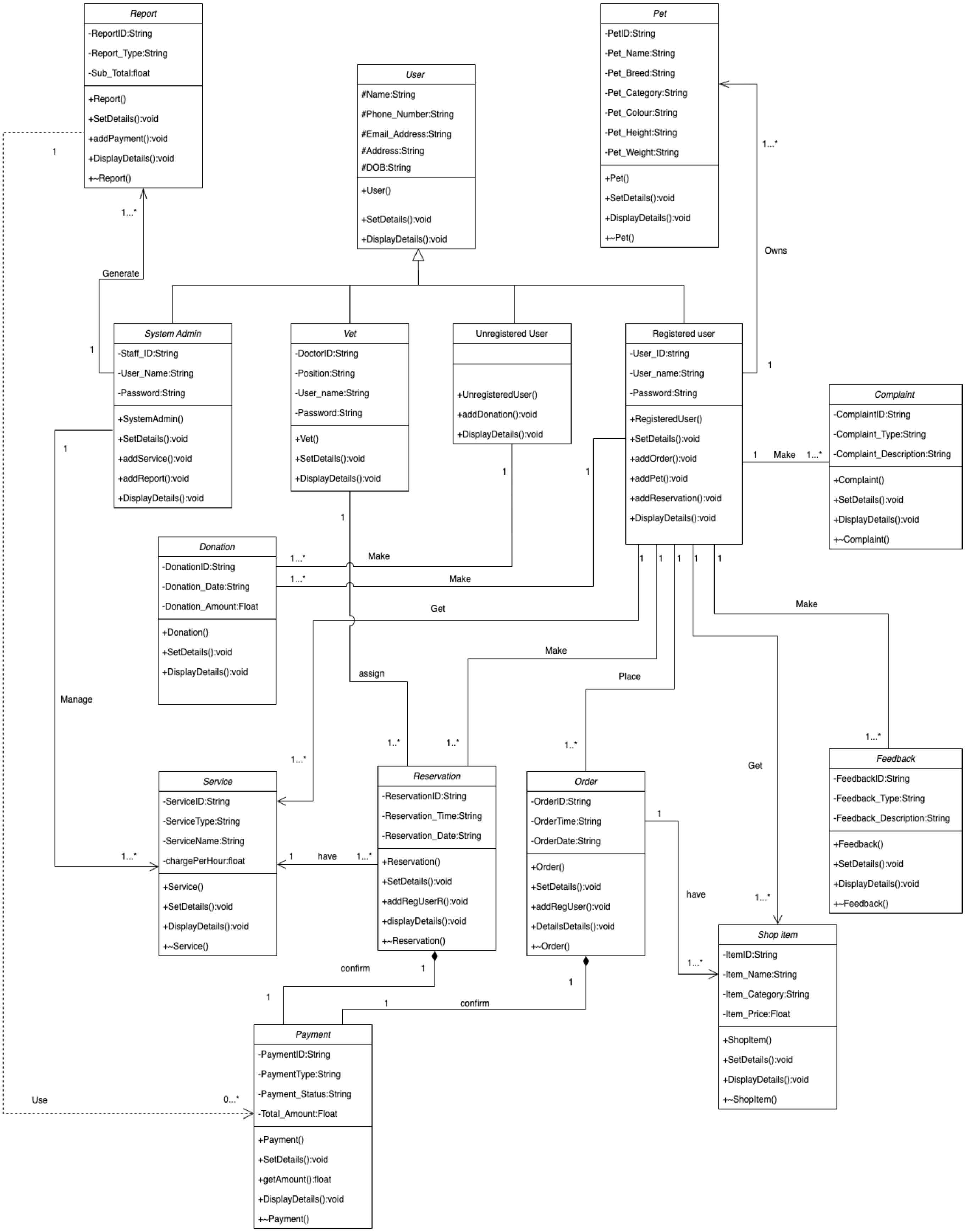
Requirement Analysis

1. The System should function 24/7/365.
2. Guest users can overview the system, to use the system, they must register with the system by providing details such as Name, Address, NIC, Email, contact and pet details.
3. Customer can register his all pets to the system using his profile.
4. Unregistered customer can register in the system by entering the details.
5. Users can login to the system by entering correct user credentials.
6. Registered customers can access any service provided by the system.
7. Registered Customers can access the website, view, and reserve available services and buy items form the shop.
8. Registered customer can edit and manage the account.
9. non-registered customer can only make donations, feedbacks, and pet harassment complaints.
10. Customers can make reservation through online transactions.
11. An available vet should be assigned to the relevant clinic reservation.
12. Payment can be paid through debit/credit or through PayPal for reservations.
13. A Customer can place an Order from online pet store. An Order consists of multiple items.
14. The customer can see the status of the Orders placed.
15. The customer specifies a payment method (credit card, debit card, PayPal, cash on delivery) for each order.
16. Once the customer confirms the order and the payment is validated the order is placed and items are updated.
17. Customers can get their products delivered.
18. After the payment is confirmed by bank or other trusted resources an invoice and appointment confirmation details emailed to the customer.
19. Customers can give their feedback and suggestions.
20. System admin can manage all the services, shop items, generate financial reports, mange feedbacks, handle customer complaints.
21. Customer can make complaints regarding the issues they faced.

Classes

1. Unregistered user
2. Registered user
3. Service
4. System Admin
5. Pet
6. Donation
7. Shop item
8. Order
9. Reservation
10. Payment
11. Feedback
12. Complaint
13. Vet
14. Report

CLASS DIAGRAM



Class Header Files

User.h

```
#ifndef USER
#define USER
class User //parent class
{
protected:

        char name[40];
        char Tp_No[15];
        char Email[30];
        char Address[80];
        char DOB[40];

public:
        User();
    User(const char U_name[], const char u_Tp_NO[],const char u_Email[] , const char
u_Address[], const char u_DOB[]); //default constructor
        void SetDetails1(const char U_name[], const char u_Tp_NO[],const char
u_Email[] , const char u_Address[], const char u_DOB[]);
//display function
        void DisplayUserDetails();
};
#endif
```

Unregistered_User.h

```
#ifndef UN
#define UN
#include "donation.h"
#include"user.h"
#define SIZE2 2
class Donation;
class Unregistered_User : public User{ //inheritence to user class
private:
    int numberOFDonations;
    Donation *d1[SIZE2];


public:
      Unregistered_User();
      Unregistered_User(const char name[], const char no[], const char add[]);
  void setUser(const char U_name[], const char u_Tp_NO[],const char u_Email[] ,
const char u_Address[], const char u_DOB[]);
      void addDonation(Donation* d2);
      void DisplayDetails(); //default costructor
      void DisplayUnRegDetails();
  void DisplayDetailsUser();
};
#endif
```

registered_user.h

```
#ifndef REGI
#define REGI

#include"user.h"
#include "Pet.h"
#include "order.h"
#include "reservation.h"
#include "shopitem.h"
#include "Service.h"
#include "complaint.h"
#include "donation.h"
#include "feedback.h"
#define SIZE7 5

class registered_user;
class order;
class Donation;
class Feedback;
class reservation;
class Service;
class Complaint;

class registered_user:public User {
    private:

    char userID[20];
    char user_Name[20];
    char password[20];
    shopitem *item[SIZE7];
    Pet *Pet1[SIZE7];
    order *o1[SIZE7]; //unidirectional connection with registered user and order
    int numberOForders;
    Service* S1[SIZE7];
      int numberOFService;
    Complaint* com[SIZE7];
      int noOfComplaints;
    Donation* dod[SIZE7];
    int noOFdonations;
    Feedback* feed[SIZE7];
    int noOFfeedbacks;
    reservation *r1[SIZE7];//unidirectional connection with registered user and
reservation
    int numberOFreservations;
    int numberOFitems;
    int numberOFpets;

    public:

    registered_user();
    registered_user(const char newuser_ID[],const char newuser_Name[],const char
newpassword[]);
    void setUser(const char U_name[], const char u_Tp_NO[],const char u_Email[] ,
const char u_Address[], const char u_DOB[]);
    //Parameterized constructor
    void addPet(Pet* p1);
    void addreservation(reservation *r2);
    void addorder(order *o2);
    void additems(shopitem *I1);
    void addService( Service* S2);
    void copli(Complaint* c);
    void addDonation(Donation* dod1);
    void addFeedback(Feedback* feed1);
    void DisplayDetailFeedback();
    void DisplayDetailsDonation();
    void DisplayDetailscomplaint();
      void displayDetailsService();
    void displayDetailsshopitem();
    void displayDetailsorder();
    void displayDetailsreservation();
    void DisplayDetailsUser();
    void DisplayPDetailsPet();
    void DisplayDetailsRegUser();
    ~registered_user();
};
#endif
```

Vet.h

```
#ifndef VET
#define VET

#include"user.h"
#include "reservation.h"
#define SIZE1 2
class Vet;
class reservation;

class Vet :public User {  //inheritance with user class

private:
      char DocId[20];
      char Position[50];
      char UserName[50];
      char Password[20];
    reservation *r1[SIZE1];
    int numberOFreservations;

public:
      Vet();
    Vet(const char d_DocId[], const char d_Position[] , const char d_UserNmae[]
,const char d_Password[]);
      void SetDetails(const char d_DocId[], const char d_Position[] , const char
d_UserNmae[] ,const char d_Password[]);
void setUser(const char U_name[], const char u_Tp_NO[],const char u_Email[] , const
char u_Address[], const char u_DOB[]);
    void addreservation(reservation *r2);
      void DisplayDocDetails();
    void displayDetailsReservation();
    ~Vet();
};
#endif
```

SystemAdmin.h

```
#ifndef ADMIN
#define ADMIN
#include"user.h" //inheritence with user class
#include "Service.h"  //connecting service header file
#include "Report.h" //connecting report header file

//report size
#define SIZE3 2

//service size
#define SIZE4 4
class Service;

class SystemAdmin : public User {  //connecting inheritence to systemadmin

private:
      char StafId[20];
      char UserName[20];
      char password[20];
  Service* ser[SIZE4];  //uni directional association connection with Service class
  int numOFser;
  Report *rep[SIZE3];    //uni directional association connection with Report class
  int numOFrep;

public:
      SystemAdmin(); //default constrcuctor
      SystemAdmin(const char u_StaffId[] , const char u_UserName[] ,const char
u_Password[]);
  void setUser(const char U_name[], const char u_Tp_NO[],const char u_Email[] ,
const char u_Address[], const char u_DOB[]);
  void addService(Service* s1); //adding service details to admin
  void addReport(Report* r1); //adding report details to admin
      void DisplayAdminDetails();
  void DisplayServiceDetails();
  void DisplayReportDetails();
   void DisplayDetailsUser();
  ~SystemAdmin();

};
#endif
```

Pet.h

```
#ifndef PET
#define PET

class Pet  //implimenting pet class
{
private:
	char petID[10];
	char petName[20];
	char petBreed[20];
	char petCategoy[20];
	char petColor[20];
	char height[20];
	char weight[20];

public:
	Pet(); //default constructor
	Pet(const char id[] ,const char name[] ,const char breed[] ,const char
category[] ,const char color[] ,const char p_height[] ,const char p_weight[]);
	void DisplayPetDetails();

};
#endif
```

Report.h

```
#ifndef  REP
#define REP
#include "payment.h"

class Report  //implimenting report class
{
private:
       char reportId[20];
       char ReportType[30];
       float sub_Total;

public:
       Report(); //default constructore
       Report(const char ID[], const char type[], float total); //parameterized
constructor
       void DisplayDetails();
       void addpayment(payment *pay);
    ~Report();

};
#endif
```

complaint.h

```
#ifndef COM
#define COM
#include "registered_user.h"

class registered_user;
class Complaint;

class Complaint
{
private:
        char ComplaintID[20];
        char Complaint_Type[30];
        char Complaint_Description[200];
    registered_user* reg;
public:
        Complaint();
        Complaint(const char cid[], const char type[],const char description[]);
        void addRegUser(registered_user* preg);
        void Display_Details();
        ~Complaint();
};
#endif
```

feedback.h

```
#ifndef feedback
#define feedback
#include "registered_user.h"
// feedback class
class registered_user;
class Feedback {
private:
	char feedbackId[10];
	char feedbackType[50];
	char feedbackDescription[100];
	registered_user* reguser1;

public:
	Feedback();
  Feedback(const char fid[],const char fname[],const char fdes[]);
    void addRegUser(registered_user* reg1);
	void displayDetails();
};
#endif
```

donation.h

```
#ifndef DON
#define DON
#include "Unregistered_User.h"
#include "registered_user.h"

class Unregistered_User;

class Donation
{
private:
	char DonationID[30];
	char Donation_Date[30];
	float Donation_Amount;
	Unregistered_User* r1;
    registered_user* reguser;

public:
	Donation();
	Donation(const char id[], const char date[], float amount);
    void addUnregUser(Unregistered_User* r2);
    void addRegUser(registered_user* newreg);
	void Display_Details();
	~Donation();
};
#endif
```

Service.h

```
#ifndef SER
#define SER
#include "reservation.h"
#define SIZE6 2
class reservation;
class Service
{

private:
	char serviceId[20];
	char serviceType[30];
	char serviceName[80];
	float ChargPerhour;
	reservation* rev[SIZE6];
	int noOFreservation;

public:
	Service(); //default constructor
	Service(const char id[], const char type[], const char name[], float charge);
//parameterized constructor
	void addreservation(reservation *rev1);
	void DisplayDetailsReservation();
	void DisplayServDetails();
  ~Service();

};
#endif
```

shopitem.h

```
#ifndef SHOP
#define SHOP


class shopitem{
    private:
    char itemID[20];
    char item_Name[50];
    char item_Category[20];
    float item_Price;
    public:
    shopitem();
    shopitem(const char newitemID[],const char newitem_Name[],const char
newitem_Category[],float newitem_Price);
    void displayDetails();
    ~shopitem();

};

#endif
```

reservation.h

```
#ifndef RES
#define RES
#include "Vet.h"
#include "payment.h"
#include "registered_user.h"

class registered_user;
class Vet;
class reservation;


class reservation {
    private :
    char resesrvationID[10];
    char reservation_Time[20];
    char reservation_Date[20];
    payment *pay1;
    registered_user *c1;
    Vet *v1;
    public :
    reservation();
    reservation(const char newreservationID[],const char newreservation_Time[],const
char newreservation_Date[],payment *pay);
    void addVet(Vet *v2);
    void addRegUserR(registered_user *c3);
    void displayDetailsReservation();
    void displayDetailswithPayment();
    ~reservation();
};
#endif
```

order.h

```
#ifndef ORDER
#define ORDER
#include "payment.h"
#include "shopitem.h"
#include "registered_user.h"
#define SIZE8 4
class registered_user;
class order;

class order {
    private:
    char order_ID[10];
    char order_Time[20];
    char order_Date[20];
    payment *pay2;
    registered_user *c1;
    shopitem *I1[SIZE8];
    public:
    order();
    order(const char neworder_ID[],const char neworder_Time[],const char
neworder_Date[],payment *pay3);
    void addRegUser(registered_user *c2);
    void addshopitem(shopitem *I2,shopitem *I3,shopitem *I4,shopitem *I5);
    void displayDetails();
    void displayDetailsshopitems();
    ~order();
};
#endif
```

payment.h

```
#ifndef PAYMENT
#define PAYMENT
class payment {
    private:
    char paymentID[10];
    char paymentType[20];
    char payment_Status[20];
    float total_Amount;
    public:
    payment();
    payment(const char newpaymentID[],const char newpaymentType[],const char
newPayment_Status[],float newttotal_Amount);
float getamount();
void displayDetails();
    ~payment();
};

#endif
```

Class Cpp Files

user.cpp

```
#include "user.h"
#include<iostream>
#include<cstring>

using namespace std;


//implimenting the User parent class


//default constructor User
User::User() {
        strcpy(name, "");
        strcpy(Tp_No, "0000000000");
        strcpy(Email, "");
        strcpy(Address, "");
        strcpy(DOB, "00 / 00 / 0000");
}

User::User(const char U_name[], const char u_Tp_NO[],const char u_Email[] , const
char u_Address[], const char u_DOB[]){
    strcpy(name, U_name);
        strcpy(Tp_No, u_Tp_NO);
        strcpy(Email, u_Email);
        strcpy(Address, u_Address);
        strcpy(DOB, u_DOB);

}

//Parameterized constructor
void User::SetDetails1(const char U_name[], const char u_Tp_NO[], const char
u_Email[], const char u_Address[], const char u_DOB[]) {

        strcpy(name, U_name);
        strcpy(Tp_No, u_Tp_NO);
        strcpy(Email, u_Email);
        strcpy(Address, u_Address);
        strcpy(DOB, u_DOB);

}


//displaying details  function implimentation
void User::DisplayUserDetails() {

}
```

Unregistered_User.cpp

```
#include<iostream>
#include<cstring>
#include"Unregistered_User.h"
#define SIZE12 2
using namespace std ;

//default constructor
Unregistered_User::Unregistered_User() {

      strcpy(name, "");
      strcpy(Tp_No, "0000000000");
      strcpy(Email, "");
      strcpy(Address, "");
      strcpy(DOB, "00 / 00 / 0000");


}


void Unregistered_User::addDonation(Donation* d2) {
    if (numberOFDonations < SIZE12) {
        d1[numberOFDonations] = d2;
        numberOFDonations++;
    }
}

//impliment display function


void Unregistered_User::DisplayDetails() {

  cout << "--------" << endl;
    cout << "Name : " << name << endl;
    cout << "User Name : " << Tp_No << endl;
    cout << "Email_Address : " << Address << endl;

    for (int i = 0; i < numberOFDonations; i++)
        d1[i]->Display_Details();


    cout << "---------"<<endl;
}
void Unregistered_User::setUser(const char U_name[], const char u_Tp_NO[],const char
u_Email[] , const char u_Address[], const char u_DOB[]){
  strcpy(name, U_name);
      strcpy(Tp_No, u_Tp_NO);
      strcpy(Email, u_Email);
      strcpy(Address, u_Address);
      strcpy(DOB, u_DOB);
}
```

```
 void Unregistered_User::DisplayDetailsUser(){

      cout << "\n----------" << endl;
      cout << "Name      -->" << name << endl;
      cout << "Tp No     -->" << Tp_No << endl;
      cout << "Email     -->" << Email << endl;
      cout << "Address   -->" << Address << endl;
      cout << "BOD       -->" << DOB << endl<<endl;
      cout << "------" << endl;


}

void Unregistered_User::DisplayUnRegDetails() {

 //display function implimentation

}
```

registered_user.cpp

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

registered_user::registered_user(
//default constructor
){}

registered_user::registered_user(const char newuser_ID[],const char
newuser_Name[],const char newpassword[]){
    strcpy(userID,newuser_ID);
    strcpy(user_Name,newuser_Name);
    strcpy(password,newpassword);
    numberOForders=0;
    numberOFreservations=0;
}

void registered_user::addorder(order *o2){
    if(numberOForders<SIZE7){
        o1[numberOForders]=o2;
        numberOForders++;
    }//counting the orders
}
void registered_user::addreservation(reservation *r2){
    if(numberOFreservations<SIZE7){
        r1[numberOFreservations]=r2;
        numberOFreservations++;
    }//counting the reservations
}
void registered_user::additems(shopitem *I1){
        if(numberOFitems<SIZE7){
        item[numberOFitems]=I1;
        numberOFitems++;
    }//counting the reservations

}
void registered_user::addPet(Pet *p1){
    if(numberOFpets<SIZE7){
        Pet1[numberOFpets]=p1;
        numberOFpets++;
    }
}

void registered_user::addService(Service * S2) {
             if (numberOFService < SIZE7) {
                   S1[numberOFService] = S2;
                   numberOFService++;
             }//counting the orders
}

void registered_user::copli(Complaint* c)
{
    if (noOfComplaints < SIZE7)
        com[noOfComplaints] = c;
    noOfComplaints++;
}
void registered_user::addDonation(Donation* dod1)
{
    if (noOFdonations < SIZE7)
        dod[noOFdonations] = dod1;
    noOFdonations++;
}

void registered_user::addFeedback(Feedback* feed1)
{
    if (noOFfeedbacks < SIZE7)
        feed[noOFfeedbacks] = feed1;
    noOFfeedbacks++;
}
void registered_user::DisplayDetailFeedback()
{
    cout << "--------" << endl;
    cout << "User ID : " << userID << endl;
    cout << "User Name : " << user_Name << endl;
    cout << "Password : " << password << endl;

    for (int i = 0; i < noOFfeedbacks; i++)
        feed[i]->displayDetails();


}
void registered_user::DisplayDetailsDonation()
{
    cout << "---------" << endl;
    cout << "User ID : " << userID << endl;
    cout << "User Name : " << user_Name << endl;
    cout << "Password : " << password << endl;

    for (int i = 0; i < noOFdonations; i++)
        dod[i]->Display_Details();


}
void registered_user::DisplayDetailscomplaint()
{
    cout << "------" << endl;
    cout << "User ID : " << userID << endl;
    cout << "User Name : " << user_Name << endl;
    cout << "Password : " << password << endl;

    for (int i = 0; i < noOfComplaints; i++)
        com[i]->Display_Details();
  cout << "--------" << endl;

}
```

```
void registered_user::displayDetailsService()
{
  cout << "--------" << endl;
      cout << "User ID =" << userID << "\n";
      cout << "User Name =" << user_Name << "\n";
      cout << "Password =" << password << "\n";
      for (int i = 0; i < SIZE7; i++)
             S1[i]->DisplayServDetails();
      cout << "------------------\n";

}

void registered_user::displayDetailsorder(){

    cout << "--------" << endl;
    cout << "User ID =" << userID << "\n";
    cout << "User Name =" << user_Name<< "\n";
    cout << "Password =" << password << "\n\n";
    for( int i = 0; i < numberOForders; i++)
        o1[i]->displayDetails();
    cout << "---------------------\n" ;
  //displaying all the orders user got
}
void registered_user::displayDetailsreservation(){

    cout << "--------" << endl;
    cout << "User ID =" << userID << "\n";
    cout << "User Name =" << user_Name<< "\n";
    cout << "Password =" << password << "\n";
    for( int i = 0; i < numberOFreservations; i++)
        r1[i]->displayDetailsReservation();
    cout << "---------------------\n" ;
  //displaying all the reservations user got
}


//printing Register details with his pets
void registered_user::DisplayPDetailsPet() {

       cout << "-----------" << endl;
       cout << "User ID   --> " << userID << endl;
       cout << "User Name --> " << user_Name << endl << endl;

       for (int i = 0; i < numberOFpets; i++) {
              Pet1[i]->DisplayPetDetails();
       }
       cout << "---------" << endl;
 }

 void registered_user::DisplayDetailsUser(){

      cout << "\n----------" << endl;
      cout << "User ID   -->" << userID << endl;
      cout << "User name -->" << user_Name<<endl;
      cout << "Password  -->" << password << endl;
      cout << "Name      -->" << name << endl;
      cout << "Tp No     -->" << Tp_No << endl;
      cout << "Email     -->" << Email << endl;
      cout << "Address   -->" << Address << endl;
      cout << "BOD       -->" << DOB << endl<<endl;
      cout << "---------------" << endl;


}
void registered_user::displayDetailsshopitem(){

  cout << "--------" << endl;
    cout << "User ID =" << userID << "\n";
    cout << "User Name =" << user_Name<< "\n";
    cout << "Password =" << password << "\n";
    for( int i = 0; i < numberOFitems; i++)
        item[i]->displayDetails();
    cout << "---------------------\n" ;
}

void registered_user::DisplayDetailsRegUser(){
  cout << "--------" << endl;
      cout << "User ID =" << userID << "\n";
    cout << "User Name =" << user_Name<< "\n";
    cout << "Password =" << password << "\n";
  cout << "--------" << endl;
}

registered_user::~registered_user(){
    cout<< "\nuser closed";
}


void registered_user::setUser(const char U_name[], const char u_Tp_NO[],const char
u_Email[] , const char u_Address[], const char u_DOB[]){
    strcpy(name, U_name);
      strcpy(Tp_No, u_Tp_NO);
      strcpy(Email, u_Email);
      strcpy(Address, u_Address);
      strcpy(DOB, u_DOB);

}
```

Vet.cpp

```
#include<iostream>
#include<cstring>
#include"Vet.h" //linking vet header file
#define SIZE11 2
using namespace std ;

Vet::Vet() {

      strcpy(DocId,"0");
      strcpy(Position,"");
      strcpy(UserName,"");
      strcpy(Password , "");
}


//implimeting set details function

Vet::Vet(const char d_DocId[], const char d_Position[], const char d_UserNmae[],
const char d_Password[]) {

      strcpy(DocId,d_DocId);
      strcpy(Position,d_Position);
      strcpy(UserName,d_UserNmae);
      strcpy(Password,d_Password);
    numberOFreservations=0;
}

void Vet::SetDetails(const char d_DocId[], const char d_Position[], const char
d_UserNmae[], const char d_Password[]) {


  //implimenting the default constructor

      strcpy(DocId,d_DocId);
      strcpy(Position,d_Position);
      strcpy(UserName,d_UserNmae);
      strcpy(Password,d_Password);
}

void Vet::addreservation(reservation *r2){
    if(numberOFreservations<SIZE11){
        r1[numberOFreservations]=r2;
        numberOFreservations++;
    }
}
void Vet::setUser(const char U_name[], const char u_Tp_NO[],const char u_Email[] ,
const char u_Address[], const char u_DOB[]){
  strcpy(name, U_name);
```

```
      strcpy(Tp_No, u_Tp_NO);
      strcpy(Email, u_Email);
      strcpy(Address, u_Address);
      strcpy(DOB, u_DOB);
}

//implimenting display function
void Vet::DisplayDocDetails() {
      cout << "---------" << endl;
      cout << "Docotr ID -->" << DocId << endl;
      cout << "User name -->" << UserName << endl;
      cout << "Password  -->" << Password << endl;
      cout << "Name      -->" << name << endl;
      cout << "Positione -->" << Position << endl;
      cout << "Tp No     -->" << Tp_No << endl;
      cout << "Email     -->" << Email << endl;
      cout << "Address   -->" << Address << endl;
      cout << "BOD       -->" << DOB << endl;
      cout << "---------" << endl;

}
void Vet::displayDetailsReservation(){
  cout << "--------" << endl;
    cout << "Docotr ID -->" << DocId << endl;
    cout << "Positione -->" << Position << endl;
      cout << "User name -->" << UserName << endl;
      cout << "Password  -->" << Password << endl;

    for( int i = 0; i < numberOFreservations; i++)
        r1[i]->displayDetailsReservation();
    cout << "------------\n" ;
}

Vet::~Vet(){
    cout << "vet closed..." << endl ;
}
```

SystemAdmin.cpp

```
#include<iostream>
#include<cstring>
#include"SystemAdmin.h"  //connecting system admin  header file
#include "Service.h"  //connecting service header file
#include "Report.h"  //connecting report header file

#define SIZE5 4

using namespace std ;


//default constructor
SystemAdmin::SystemAdmin() {

       strcpy(StafId,"0");
       strcpy(UserName,"");
       strcpy(password,"");
}


//implimenting set details function
SystemAdmin::SystemAdmin(const char u_StaffId[], const char u_UserNmae[], const char
u_Password[]) {

       strcpy(StafId,u_StaffId);
       strcpy(UserName,u_UserNmae);
       strcpy(password,u_Password);
}

void SystemAdmin::addService(Service* s1){

       if(numOFser<SIZE5){
        ser[numOFser]=s1;
        numOFser++;
    }
}

void SystemAdmin::addReport(Report* r1) {

       if(numOFrep<SIZE5){
        rep[numOFrep]=r1;
        numOFrep++;
    }
}


//implimenting display function

void SystemAdmin::DisplayAdminDetails() {
  }

//Service + System Admin
void SystemAdmin::DisplayServiceDetails() {

      cout << "---------" << endl;
      cout << "Admin ID   --> " << StafId << endl;
      cout << "Admin Name --> " << UserName << endl << endl;

      for (int i = 0; i < numOFser; i++) {
             ser[i]->DisplayServDetails();
      }
      cout << "----------" << endl;
}



//Report  + System admin
void SystemAdmin::DisplayReportDetails() {

      cout << "---------" << endl;
      cout << "Admin ID   --> " << StafId << endl;
      cout << "Admin Name --> " << UserName <<endl << endl;

      for (int i = 0; i < numOFrep; i++) {
             rep[i]->DisplayDetails();
      }
      cout << "-------" << endl;
}
void SystemAdmin::setUser(const char U_name[], const char u_Tp_NO[],const char
u_Email[] , const char u_Address[], const char u_DOB[]){
  strcpy(name, U_name);
      strcpy(Tp_No, u_Tp_NO);
      strcpy(Email, u_Email);
      strcpy(Address, u_Address);
      strcpy(DOB, u_DOB);
}


 void SystemAdmin::DisplayDetailsUser(){

      cout << "\n-------" << endl;
      cout << "Staff ID   -->" << StafId << endl;
      cout << "User name -->" << UserName<<endl;
      cout << "Password  -->" << password << endl;
      cout << "Name      -->" << name << endl;
      cout << "Tp No     -->" << Tp_No << endl;
      cout << "Email     -->" << Email << endl;
      cout << "Address   -->" << Address << endl;
      cout << "BOD       -->" << DOB << endl<<endl;
      cout << "-------" << endl;


}

SystemAdmin::~SystemAdmin(){
  //destructor
}
```

Pet.cpp

```
#include<iostream>
#include "Pet.h"  //connecting pet header file
#include<cstring>
using namespace std ;


//default constructore
Pet::Pet() {

      strcpy(petID, "0");
      strcpy(petName, "");
      strcpy(petBreed, "");
      strcpy(petCategoy, "");
      strcpy(petColor, "");
      strcpy(height, "0.0 inch");
      strcpy(weight, " 0.0 KG");
}


//parameterized constructor
 Pet::Pet(const char id[], const char name[], const char breed[], const char
category[], const char color[], const char p_height[], const char p_weight[]) {

      strcpy(petID, id);
      strcpy(petName, name);
      strcpy(petBreed, breed);
      strcpy(petCategoy, category);
      strcpy(petColor, color);
      strcpy(height, p_height);
      strcpy(weight, p_weight);
}

void Pet::DisplayPetDetails() {
  cout << "-------" << endl;
      cout << "Pet ID          --> " << petID << endl;
      cout << "pet name        --> " << petName << endl;
      cout << "pet breed       --> " << petBreed  << endl;
      cout << "Pet category    --> " << petCategoy << endl;
      cout << "Pet color       --> " << petColor << endl;
      cout << "Pet height      --> " << height << endl;
      cout << "Pet weight      --> " << weight <<endl<< endl;
      cout << "--------" << endl;

}
```

Report.cpp

```cpp
#include "SystemAdmin.h" // connecting system admin header file
#include "Report.h" //connecting report header file
#include<cstring>
#include<iostream>


using namespace std;


Report::Report() {

      strcpy(reportId, "0");
      strcpy(ReportType, "");
      sub_Total = 0.0;
}

Report::Report(const char ID[], const char type[], float total) {

      strcpy(reportId, ID);
      strcpy(ReportType, type);
      sub_Total = total;
}

void Report::DisplayDetails() {

  cout << "--------" << endl;
      cout << "\n Report ID   --> " << reportId << endl;
      cout << " Report Type --> " << ReportType << endl;
      cout << " sub total   --> " << sub_Total << endl << endl;
  cout << "--------" << endl;


}

void Report::addpayment(payment *pay){
    sub_Total=sub_Total+pay->getamount();
}

Report::~Report(){
  //Destructor
}
```

complaint.cpp

```cpp
#include <iostream>
#include <cstring>
#include"registered_user.h"
#include "complaint.h"
using namespace std;


Complaint::Complaint()
{

}
Complaint::Complaint(const char cid[], const char type[], const char description[])
{
	strcpy(ComplaintID, cid);
	strcpy(Complaint_Type, type);
	strcpy(Complaint_Description, description);
}

void Complaint::addRegUser(registered_user* preg){
	reg = preg;
	reg->copli(this);
}

void Complaint::Display_Details()
{
	cout << "---------" << endl;
	cout <<  "Complaint ID : " << ComplaintID << endl;
	cout << "Complaint Type : " << Complaint_Type <<endl;
	cout << "Complaint Description : " << Complaint_Description << endl;
	cout << "---------" << endl;
}

Complaint::~Complaint()
{
	//Destructor
}
```

feedback.cpp

```cpp
#include<cstring>
#include<iostream>
using namespace std;
#include"feedback.h"
//implementing the feedback class

Feedback::Feedback() {
      strcpy(feedbackId, "0");
      strcpy(feedbackType, "");
      strcpy(feedbackDescription, "");
}

Feedback::Feedback(const char fid[],const char fname[],const char fdes[]) {
      strcpy(feedbackId, fid);
      strcpy(feedbackType, fname);
      strcpy(feedbackDescription, fdes);
}
void Feedback::addRegUser(registered_user* reg1){
    reguser1 = reg1;
    reguser1->addFeedback(this);
}
void Feedback::displayDetails() {
    cout << "---------"<< endl;
      cout << "Feedback Id : " << feedbackId << endl;
      cout << "Feedback Type : " << feedbackType << endl;
      cout << "Feedback Description : " << feedbackDescription << endl;
    cout << "---------"<< endl;
}
```

donation.cpp

```cpp
#include<iostream>
#include<cstring>
#include "Unregistered_User.h"

#include "donation.h"
using namespace std;



Donation::Donation()
{

}

Donation::Donation(const char id[], const char date[], float amount)
{
    strcpy(DonationID, id);
    strcpy(Donation_Date, date);
    Donation_Amount = amount;

}

void Donation::addUnregUser(Unregistered_User* r2){
    r1 = r2;
    r1->addDonation(this);
}

void Donation::addRegUser(registered_user* newreg){
    reguser = newreg;
    reguser->addDonation(this);
}


void Donation::Display_Details() {

    cout << "----------" << endl;

    cout << "Donation ID : " << DonationID << endl;
    cout << " Donation Date : " << Donation_Date << endl;
    cout << " Donation Amount Date : " << Donation_Amount << endl;

    cout << "----------" << endl;

}

Donation::~Donation()
{
    //destructor
}
```

Service.cpp

```
#include "Service.h" //connecting service header file
#include<cstring>
#include<iostream>
#define SIZE13 2
using namespace std;


//defaul constructor
Service::Service() {

        strcpy(serviceId, "0");
        strcpy(serviceName, "");
        strcpy(serviceType, "");
        ChargPerhour = 0.0;
}


//parameterized constructore
Service::Service(const char id[], const char type[], const char name[], float
charge) {

        strcpy(serviceId,id);
        strcpy(serviceName,name);
        strcpy(serviceType,type);
        ChargPerhour = charge;


}
void Service::addreservation(reservation* rev1){
    if (noOFreservation < SIZE13)
        rev[noOFreservation] = rev1;
        noOFreservation++;

}

void Service::DisplayDetailsReservation()
{
        cout << "---------------------------------" << endl;
        cout << "Service ID : " << serviceId << endl;
        cout << "Service Type : " << serviceType << endl;
        cout << "Service Name : " << serviceName << endl;
        cout << "ChargPerhour : " << ChargPerhour << endl;

        for (int i = 0; i < noOFreservation; i++)
                rev[i]->displayDetailsReservation();

  cout << "--------" << endl;
}

void Service::DisplayServDetails() {
    cout << "---------" << endl;
      cout << "Service ID : " << serviceId << endl;
      cout << "Service Type : " << serviceType << endl;
      cout << "Service Name : " << serviceName << endl;
      cout << "ChargPerhour : " << ChargPerhour << endl;
      cout << "---------" << endl;

}

Service::~Service(){
  //destructor
}
```

shopitem.cpp

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

shopitem::shopitem(){

}

shopitem::shopitem(const char newitemID[],const char newitem_Name[],const char
newitem_Category[],float newitem_Price){
    strcpy(itemID,newitemID);
    strcpy(item_Name,newitem_Name);
    strcpy(item_Category,newitem_Category);
    item_Price=newitem_Price;
}

void shopitem::displayDetails(){
  cout << "--------" << endl;
    cout << "\nItem ID =" << itemID<< "\n" ;
    cout << "Item Name =" << item_Name<< "\n" ;
    cout << "Item Category =" << item_Category<< "\n" ;
    cout << "Item Price =" << item_Price<< "\n" ;
  cout << "--------" << endl;
}

shopitem::~shopitem(){
    cout<<"\n shop item closed...";
}
```

reservation.cpp

```
#include "Vet.h"
#include "reservation.h"
#include <iostream>
#include <cstring>
using namespace std;


reservation::reservation(){

}

reservation::reservation(const char newreservationID[],const char
newreservation_Time[],const char newreservation_Date[],payment *pay){
    strcpy(resesrvationID,newreservationID);
    strcpy(reservation_Time,newreservation_Time);
    strcpy(reservation_Date,newreservation_Date);
    pay1=pay;

}

void reservation::addVet(Vet *v2){
    v1=v2;
    v1->addreservation(this);
}

void reservation::addRegUserR(registered_user *c3){
    c1=c3;
    c1->addreservation(this);
}

void reservation::displayDetailsReservation(){
  cout << "--------" << endl;
   cout << "\n Reservation ID =" << resesrvationID << "\n" ;
   cout << " Reservation Time =" << reservation_Time << "\n" ;
   cout << " Reservation Date =" << reservation_Date<< "\n" ;
  cout << "--------" << endl;
}

void reservation::displayDetailswithPayment(){
  cout << "--------" << endl;
   cout << "\n Reservation ID =" << resesrvationID << "\n" ;
   cout << " Reservation Time =" << reservation_Time << "\n" ;
   cout << " Reservation Date =" << reservation_Date<< "\n" ;
   pay1->displayDetails();
  cout << "--------" << endl;
}

reservation::~reservation(){
   delete pay1;
   cout << "Reservation closed..." << endl;
}
```

order.cpp

```
#include "order.h"
#include <iostream>
#include <cstring>
using namespace std;
order::order(){
    pay2= new payment(" "," "," ",00.00);
}
order::order(const char neworder_ID[],const char neworder_Time[],const char
neworder_Date[],payment *pay3){
   strcpy(order_ID,neworder_ID);
   strcpy(order_Time,neworder_Time);
   strcpy(order_Date,neworder_Date);
   pay2=pay3;
}
void order::addRegUser(registered_user *c2){
       c1=c2;
   c1->addorder(this);
}

void order::addshopitem(shopitem *I2,shopitem *I3,shopitem *I4,shopitem *I5){
    I1[0]=I2;
    I1[1]=I3;
    I1[2]=I4;
    I1[3]=I5;
}

void order::displayDetails(){
  cout << "Order ID =" << order_ID << "\n" ;
  cout << "Order Time =" << order_Time << "\n" ;
  cout << "Order Date =" << order_Date << "\n" ;
  pay2->displayDetails();

}

void order::displayDetailsshopitems(){
  cout << "\nOrder ID =" << order_ID << "\n" ;
  cout << "Order Time =" << order_Time << "\n" ;
  cout << "Order Date =" << order_Date << "\n" ;
  for( int i = 0; i < SIZE8; i++)
        I1[i]->displayDetails();
    cout << "--------------------\n" ;

}
order::~order(){
  cout << "Order Closed...";
}
```

payment.cpp

```
#include "payment.h"
#include <iostream>
#include <cstring>
using namespace std;
payment::payment(){

}

payment::payment(const char newpaymentID[],const char newpaymentType[],const char
newPayment_Status[],float newttotal_Amount){
    strcpy(paymentID,newpaymentID);   strcpy(paymentType,newpaymentType);
    strcpy(payment_Status,newPayment_Status); total_Amount=newttotal_Amount;
}

float payment::getamount(){
    return total_Amount;
}

void payment::displayDetails(){
    cout << "--------" << endl;
    cout << "\n Payment ID =" << paymentID   << "\n" ;
    cout << " Payment Type =" << paymentType   << "\n" ;
    cout << " Payment Status =" <<  payment_Status  << "\n" ;
    cout << " Total Amount =" << total_Amount   << "\n" ;
    cout << "--------" << endl;
}
payment::~payment(){
    cout << "payment closed..." << endl;
}
```

Main program

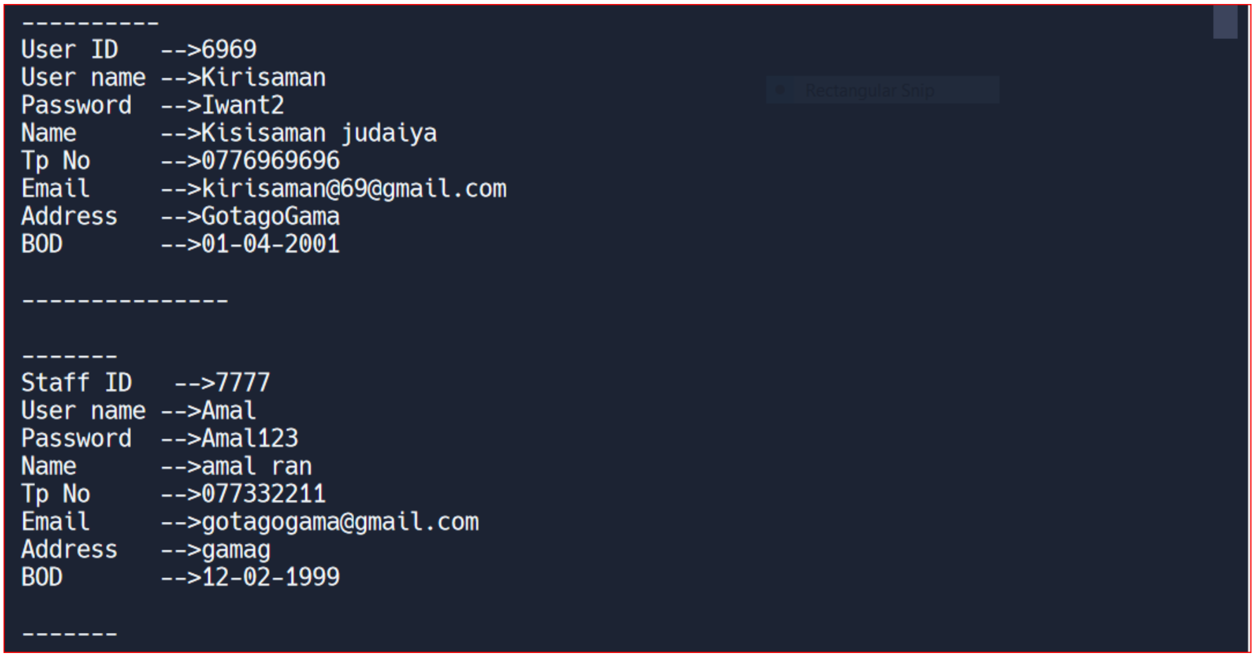
Main.cpp

```
//connecting header files

#include<cstring>
#include<iostream>
#include "Unregistered_User.h"
#include "Vet.h"
#include "SystemAdmin.h"
#include "registered_user.h"
#include "Pet.h"
#include "user.h"
#include "Service.h"
#include "Report.h"
#include "order.h"
#include "reservation.h"
#include "donation.h"
#include "complaint.h"
#include "feedback.h"


using namespace std;

int main()
{
  //registered user object creation
  registered_user *reg1= new registered_user("6969","Kirisaman","Iwant2");

  //setting values for user class
  reg1->setUser("Kisisaman
judaiya","0776969696","kirisaman@69@gmail.com","GotagoGama","01-04-2001");

  //registered user print details
    reg1->DisplayDetailsUser();

 //system admin object creation
  SystemAdmin *staff1=new SystemAdmin("7777","Amal","Amal123");

  //setting values for user class
  staff1->setUser("amal ran","077332211","gotagogama@gmail.com","gamag","12-02-
1999");

  //system admin print ddetails
  staff1->DisplayDetailsUser();

  //unregistered user object creation
  Unregistered_User *unuser1=new Unregistered_User;

  //setting values for user class
  unuser1->setUser("ffre","---","---","---","---");

  //unregistered user print details
  unuser1->DisplayDetailsUser();

  //vet object creation
  Vet *vet1=new Vet("v0001","helth doc","dr.amarasinhe","45rtgs");

  //setting values for user class
  vet1->setUser("amarasinhe
rupahinhe","077635274","amara69@gmail.com","colombo","21-05-2022");

  //vet print details
  vet1->DisplayDocDetails();

  //pet object creation
  Pet *pet1=new Pet("pet001","jonny","pitbull","dog","red","60cm","5KG");
  Pet *pet2=new Pet("pet002","kitty","saico","cat","blue","20cm","2KG");

  //assigning pet details to registered user
  reg1->addPet(pet1);
  reg1->addPet(pet2);

  //pet print details with registered user details
  reg1->DisplayPDetailsPet();

  //report object creation
  Report *rep1=new Report("rep001","income",300.00);

  //assiging report details to registered user
  staff1->addReport(rep1);

  //report print details with registered user details
  staff1->DisplayReportDetails();

  //payment object creation
  payment *pay001=new payment("pay001","credit","done",400.00);

  //payment print details
  pay001->displayDetails();

  //add payment amount  to report
  rep1->addpayment(pay001);

  //print report with new total amount
  rep1->DisplayDetails();

  //service object creation
  Service *s001=new Service("s001","health care","medince",300.00);

  //assigning service details to system admin
  staff1->addService(s001);

  //service print details with system admin details
  staff1->DisplayServiceDetails();

  //complaint object creation
  Complaint *c0001=new Complaint("c001","service","service awl");
```

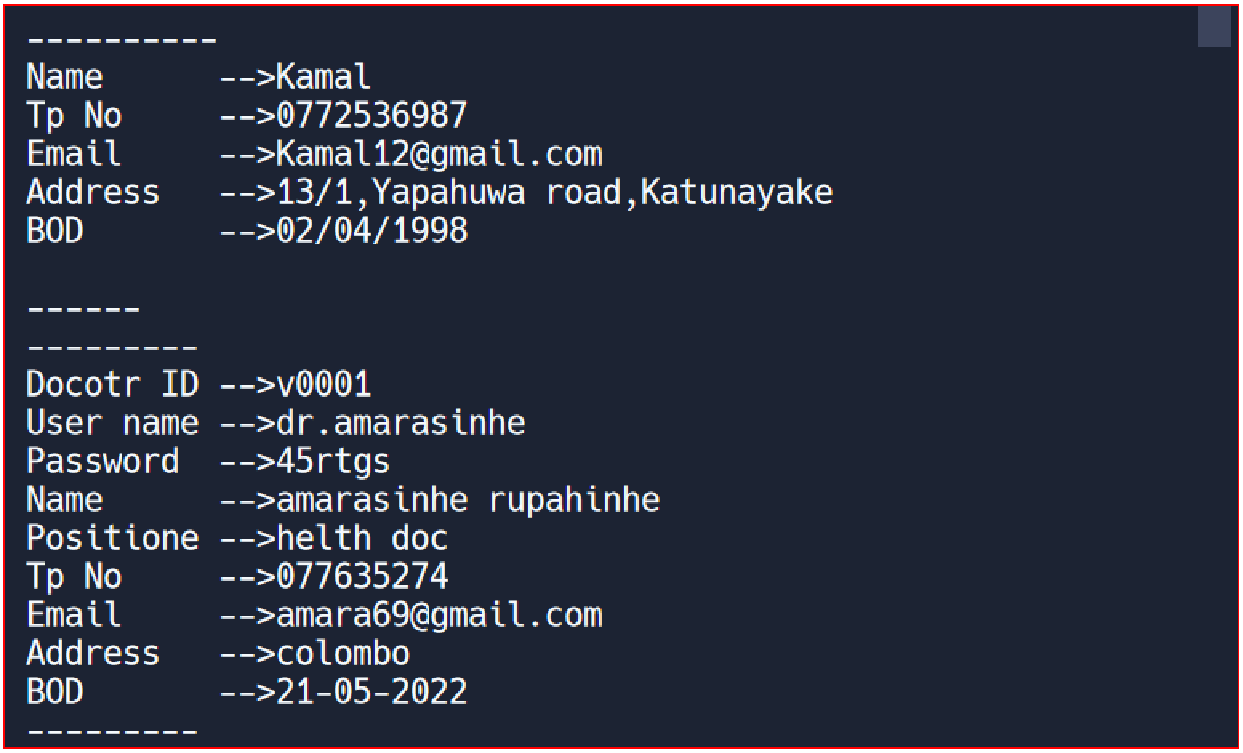
```
reg1->copli(c0001);
reg1->DisplayDetailscomplaint();

//donation object creation
Donation *d001= new Donation("d0001","12-04-2022",4000.00);
reg1->addDonation(d001);
reg1->DisplayDetailsDonation();

//donation object creation
 Donation *d002= new Donation("d0002","12-04-2022",5000.00);

//assing donation details to registered user
 unuser1->addDonation(d002);

//reservation print detalis with registered user details
 unuser1->DisplayDetails();

//reservation object creation
reservation *re001=new reservation("re001","10:10:10","12-04-2022",pay001);

 //assing reservation details to vet
vet1->addreservation(re001);

//reservation print detals with vet
vet1->displayDetailsReservation();

//assing reservation details to registered user
reg1->addreservation(re001);

//reservation print detals with registered user details
reg1->displayDetailsreservation();

//print reservation details with payment details
re001->displayDetailswithPayment();

//payment object creation
payment *pay002=new payment("pay002","paypal","done",5000.00);

//payment print details
pay002->displayDetails();

//add payment amount  to report
rep1->addpayment(pay002);

//print report with new total amount
rep1->DisplayDetails();

//order object creation
order *or001=new order("or001","12:12:12","12-12-2022",pay002);

//assing order details to registered user
reg1->addorder(or001);

//order print detals with registered user details
reg1->displayDetailsorder();

//print order details with payment details
or001->displayDetailswithPayment();
```

```
  //shop item object creation
  shopitem *it001=new shopitem("it001","belt","assesoires",500.00);

  //assing shop item details to registered user
  reg1->additems(it001);

  //shop item print detals with registered user details
  reg1->displayDetailsshopitem();


  //feedback object creation
  Feedback *feed1=new Feedback("f0001","Service","service nam maru");

  //assing feedback details to registered user
  reg1->addFeedback(feed1);

  //feedback print detals with registered user details
  reg1->DisplayDetailFeedback();

  delete feed1;
  delete it001;
  delete or001;
  delete re001;
  delete d001;
  delete d002;
  delete c0001;
  delete s001;
  delete rep1;
  delete pet1;
  delete pet2;
  delete vet1;
  delete unuser1;
  delete staff1;
  delete reg1;
    return 0;
};
```

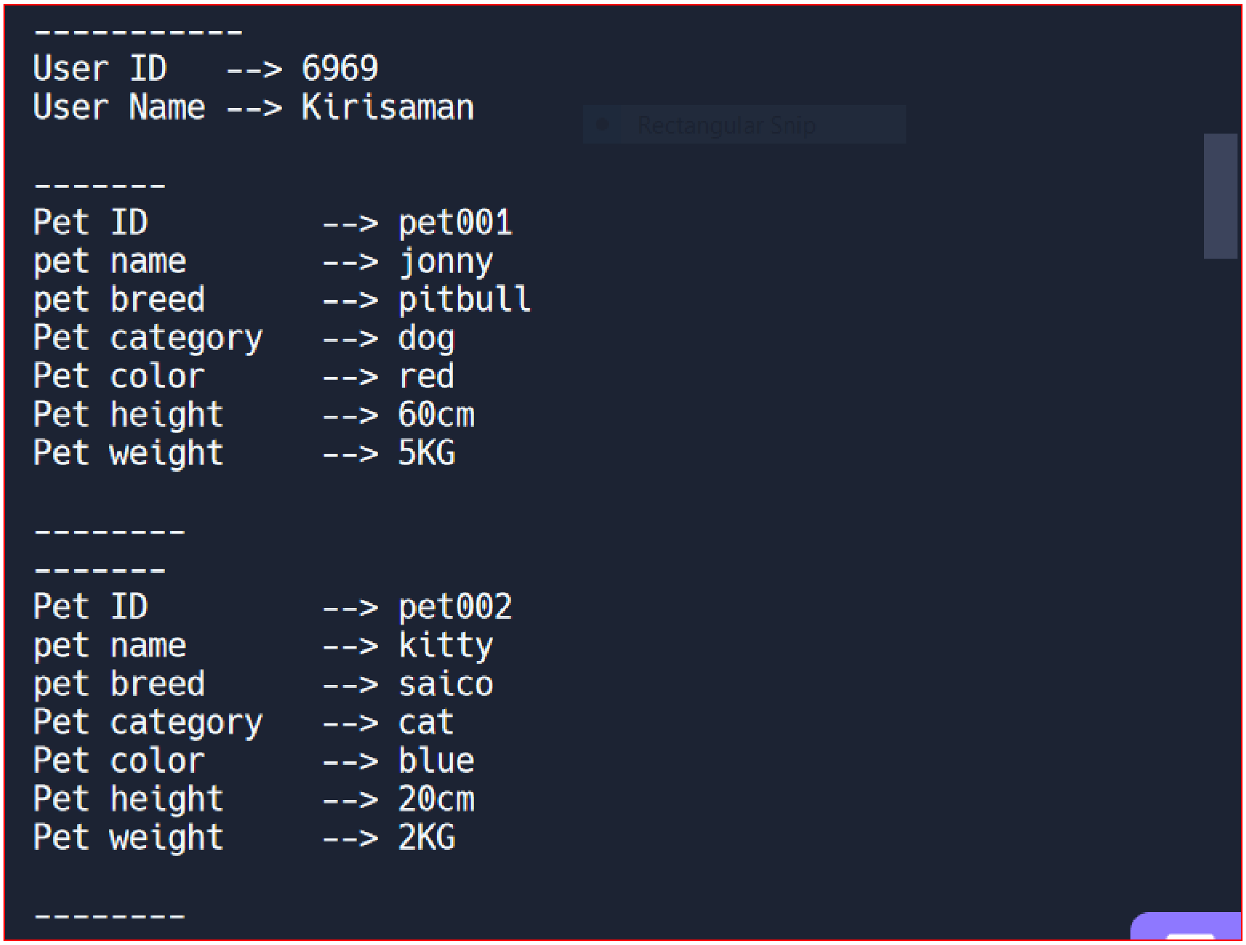
Sample Outputs



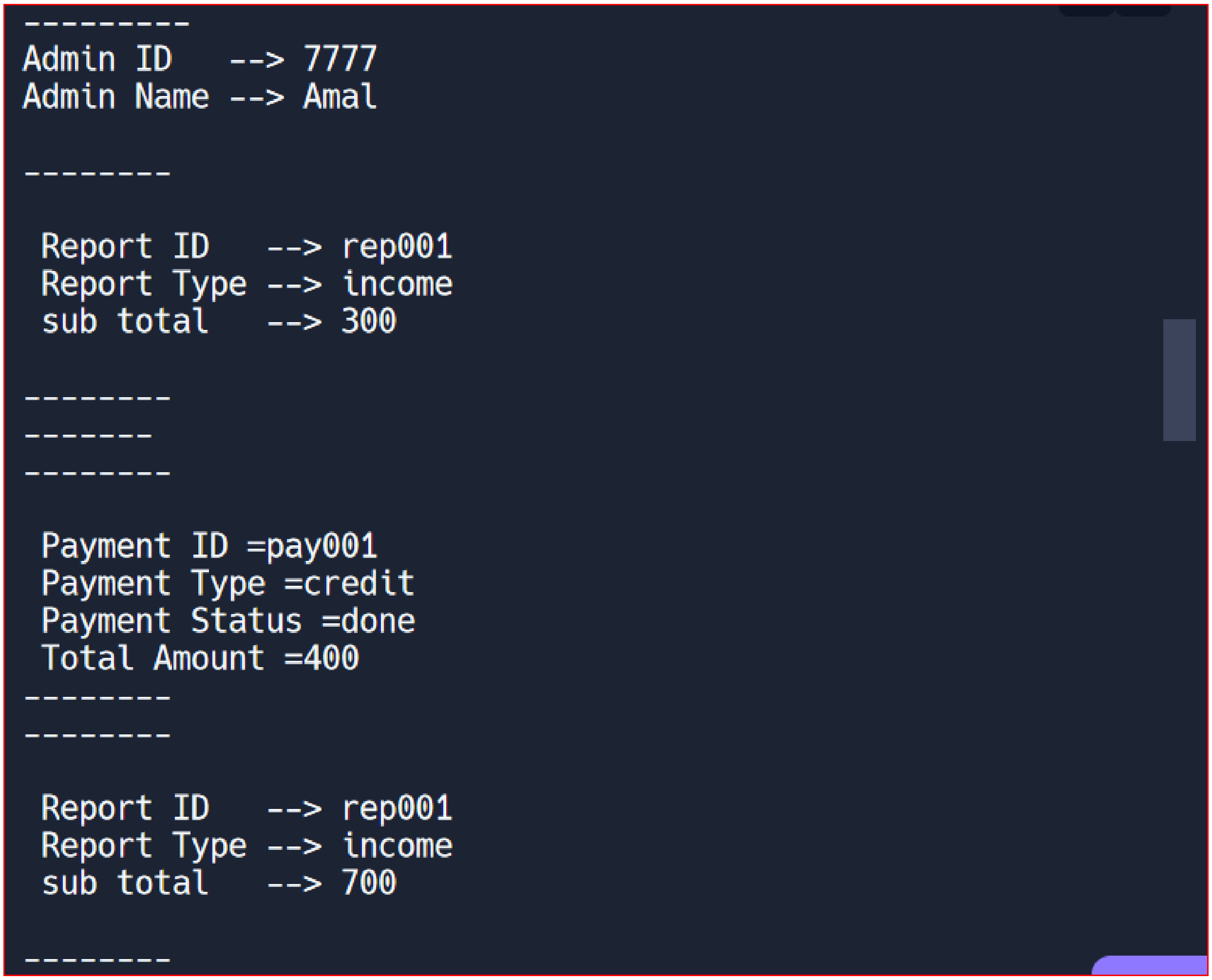
```
----------
User ID   -->6969
User name -->Kirisaman
Password  -->Iwant2
Name      -->Kisisaman judaiya
Tp No     -->0776969696
Email     -->kirisaman@69@gmail.com
Address   -->GotagoGama
BOD       -->01-04-2001

---------------

-------
Staff ID   -->7777
User name -->Amal
Password  -->Amal123
Name      -->amal ran
Tp No     -->077332211
Email     -->gotagogama@gmail.com
Address   -->gamag
BOD       -->12-02-1999

-------
```



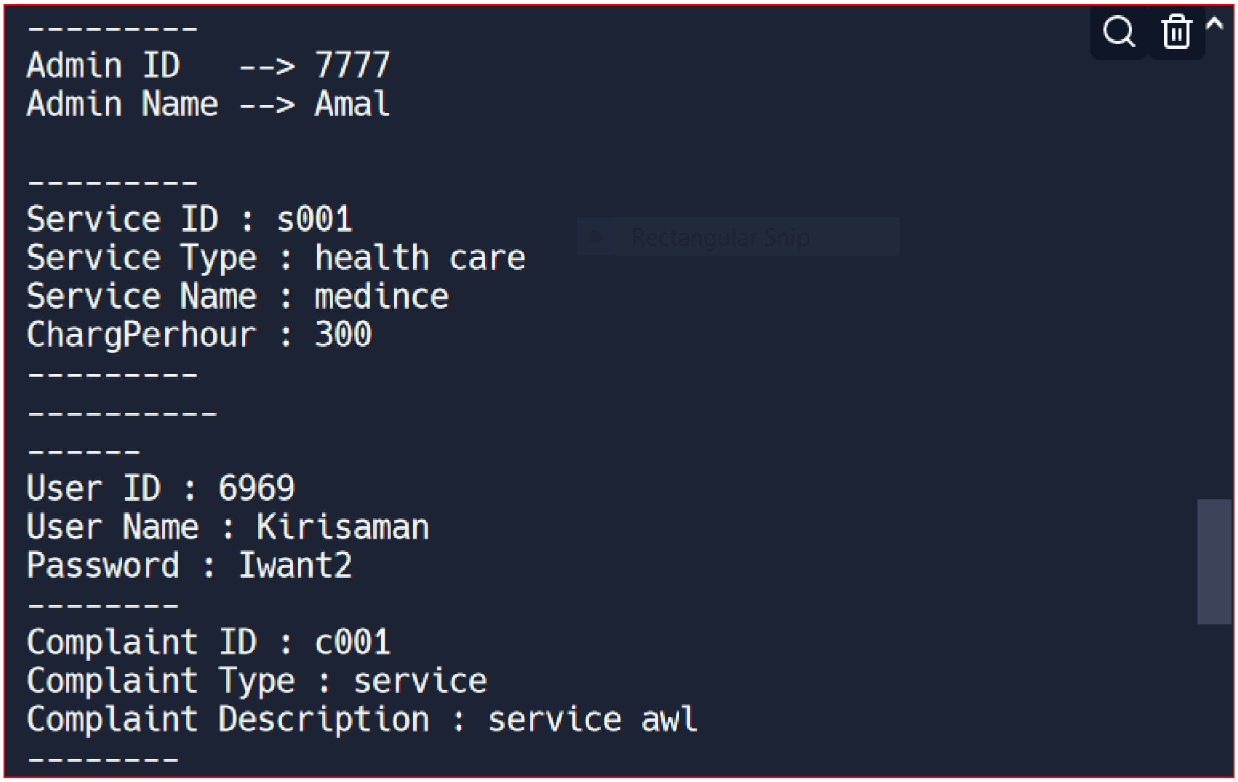
```
----------
Name      -->Kamal
Tp No     -->0772536987
Email     -->Kamal12@gmail.com
Address   -->13/1,Yapahuwa road,Katunayake
BOD       -->02/04/1998

------
---------
Docotr ID -->v0001
User name -->dr.amarasinhe
Password  -->45rtgs
Name      -->amarasinhe rupahinhe
Positione -->helth doc
Tp No     -->077635274
Email     -->amara69@gmail.com
Address   -->colombo
BOD       -->21-05-2022
---------
```



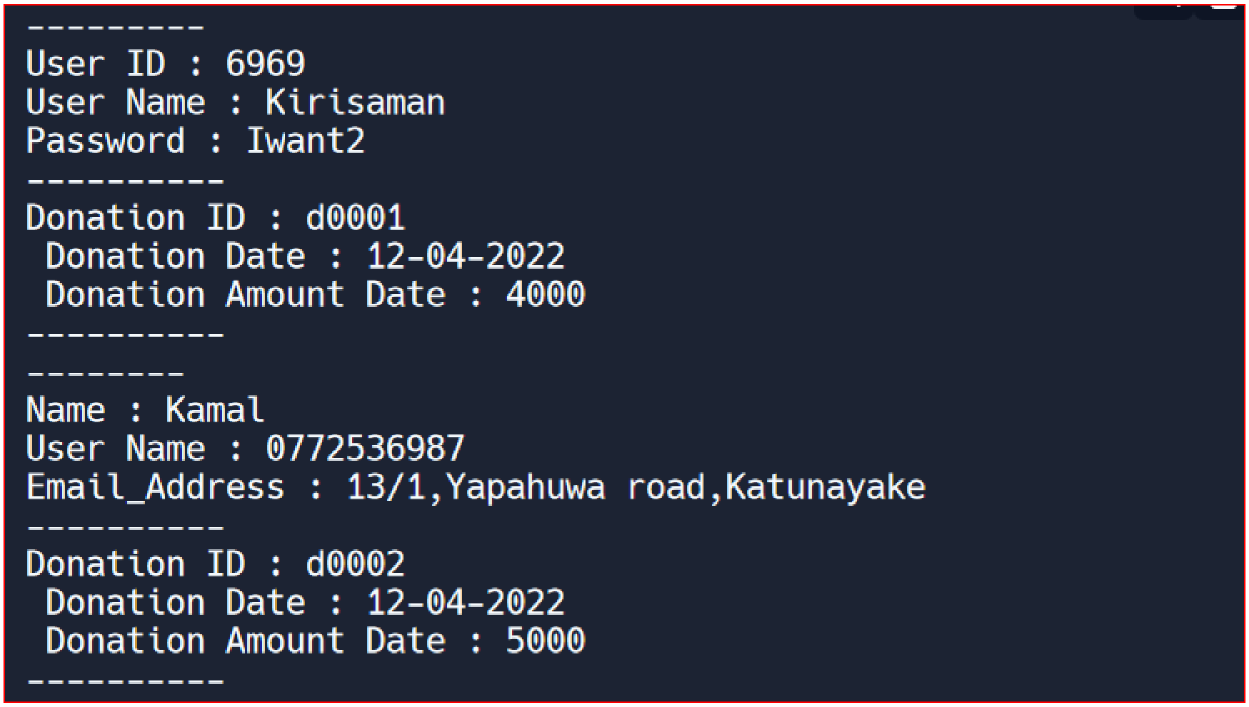
```
-----------
User ID   --> 6969
User Name --> Kirisaman

-------
Pet ID          --> pet001
pet name        --> jonny
pet breed       --> pitbull
Pet category    --> dog
Pet color       --> red
Pet height      --> 60cm
Pet weight      --> 5KG

--------
-------
Pet ID          --> pet002
pet name        --> kitty
pet breed       --> saico
Pet category    --> cat
Pet color       --> blue
Pet height      --> 20cm
Pet weight      --> 2KG

--------
```



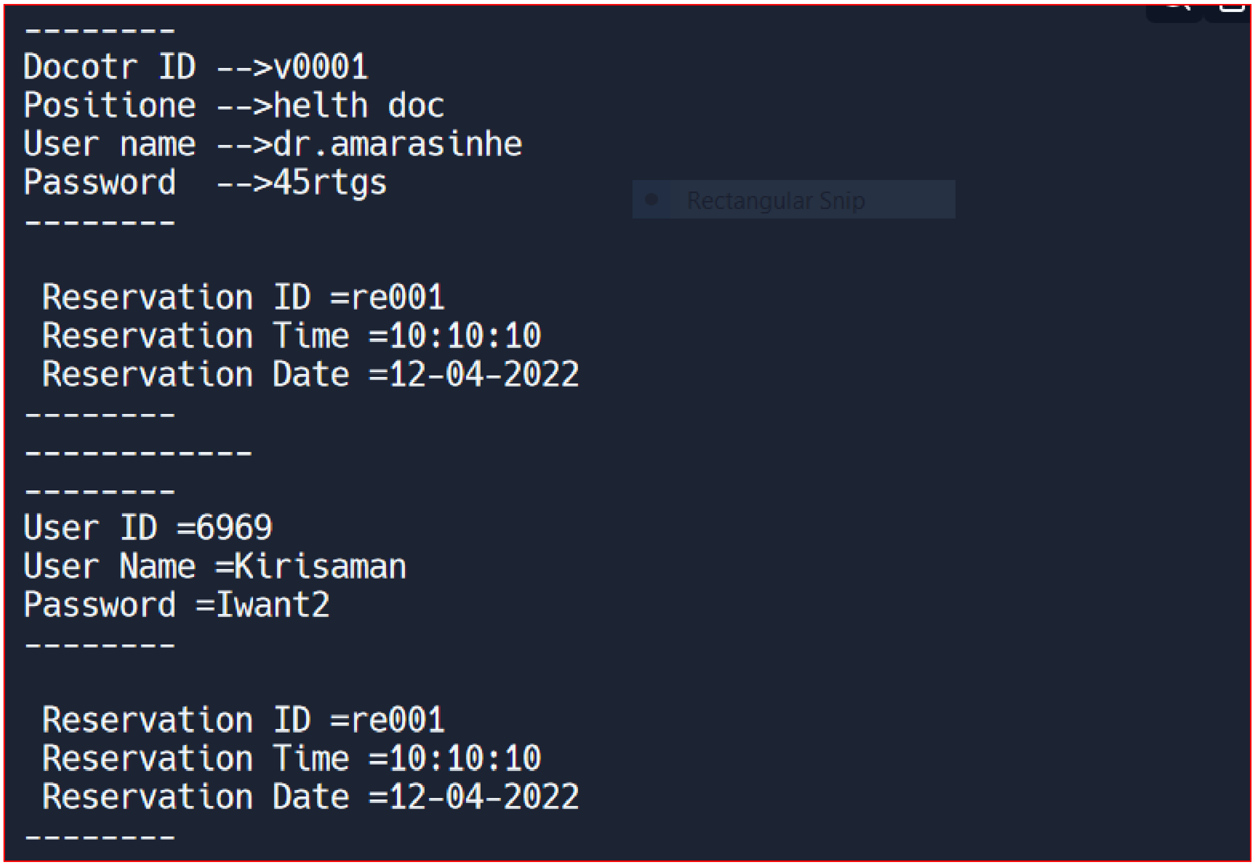
```
---------
Admin ID   --> 7777
Admin Name --> Amal

--------

 Report ID   --> rep001
 Report Type --> income
 sub total   --> 300

--------
-------
--------

 Payment ID =pay001
 Payment Type =credit
 Payment Status =done
 Total Amount =400
--------
--------

 Report ID   --> rep001
 Report Type --> income
 sub total   --> 700

--------
```



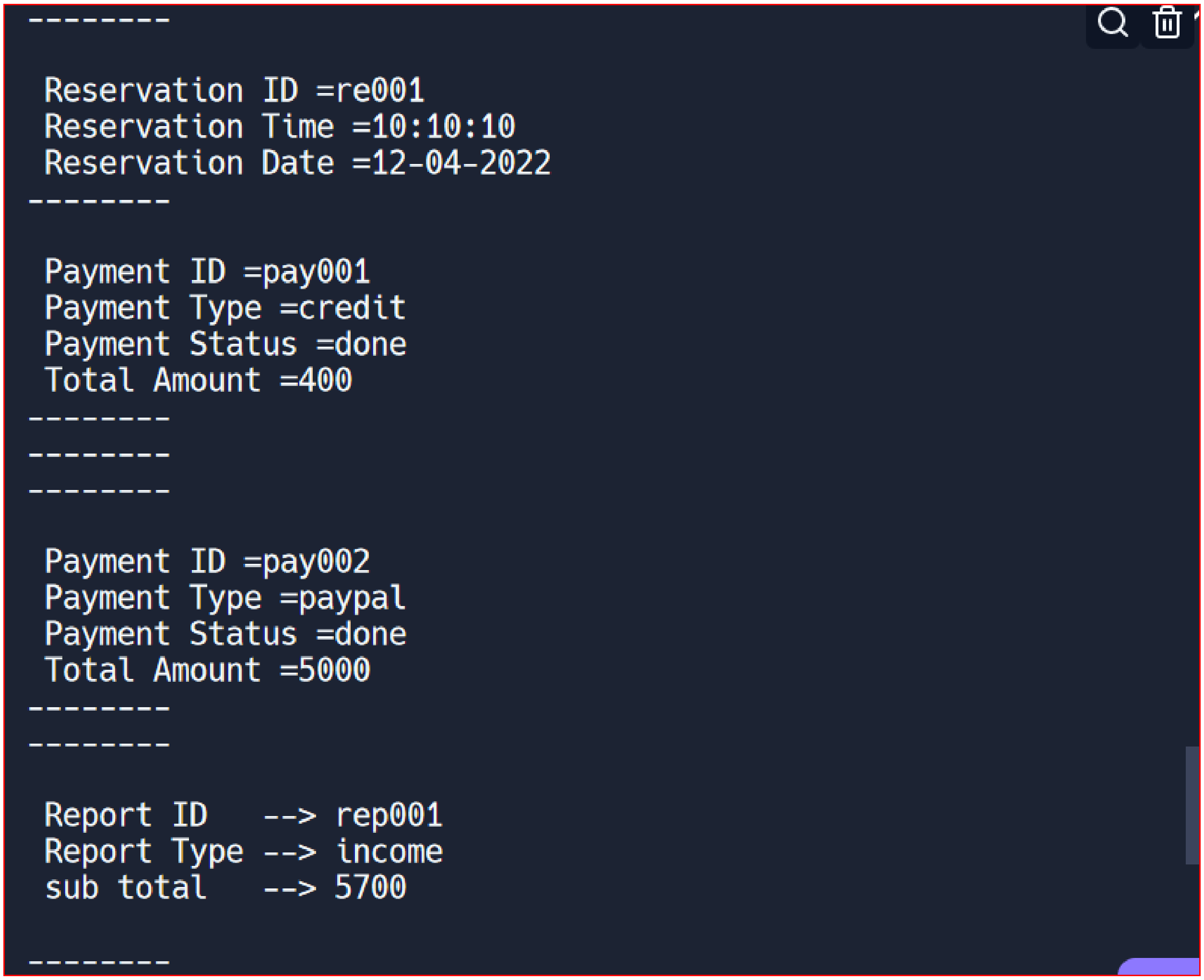
```
---------
Admin ID   --> 7777
Admin Name --> Amal

---------
Service ID : s001
Service Type : health care
Service Name : medince
ChargPerhour : 300
---------
----------
------
User ID : 6969
User Name : Kirisaman
Password : Iwant2
--------
Complaint ID : c001
Complaint Type : service
Complaint Description : service awl
--------
```



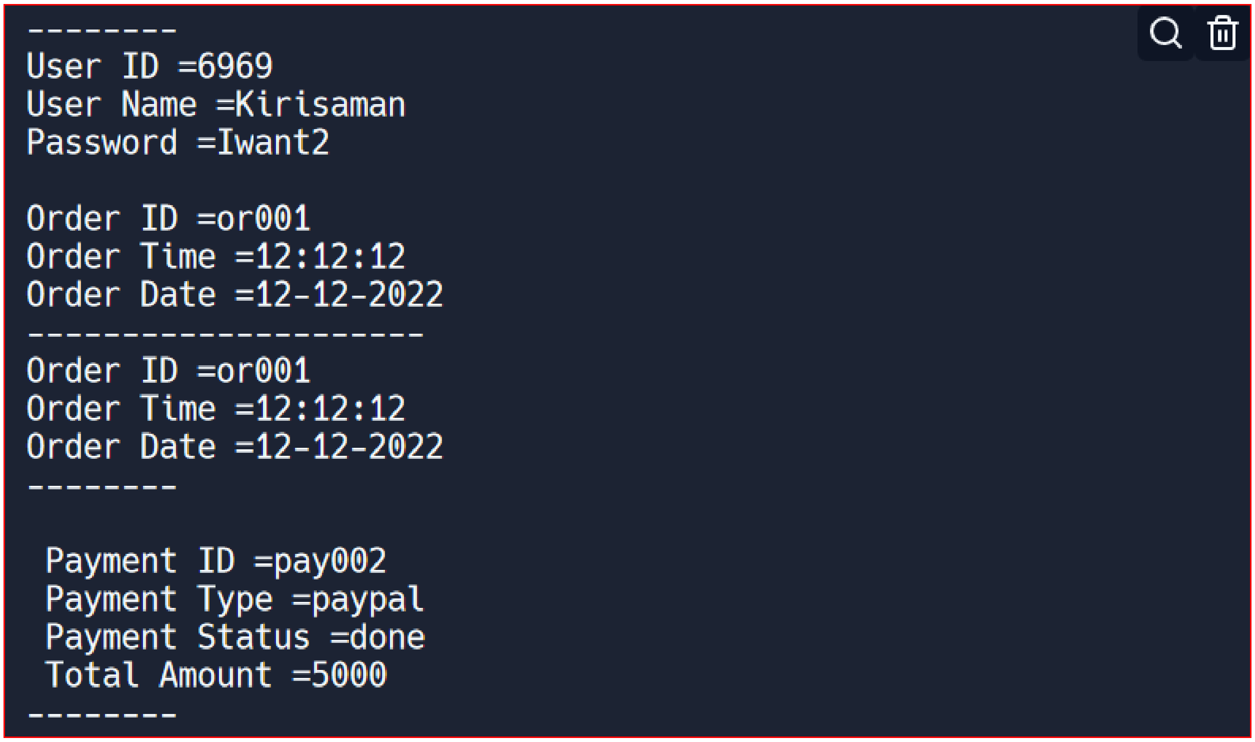
```
---------
User ID : 6969
User Name : Kirisaman
Password : Iwant2
----------
Donation ID : d0001
 Donation Date : 12-04-2022
 Donation Amount Date : 4000
----------
--------
Name : Kamal
User Name : 0772536987
Email_Address : 13/1,Yapahuwa road,Katunayake
----------
Donation ID : d0002
 Donation Date : 12-04-2022
 Donation Amount Date : 5000
----------
```



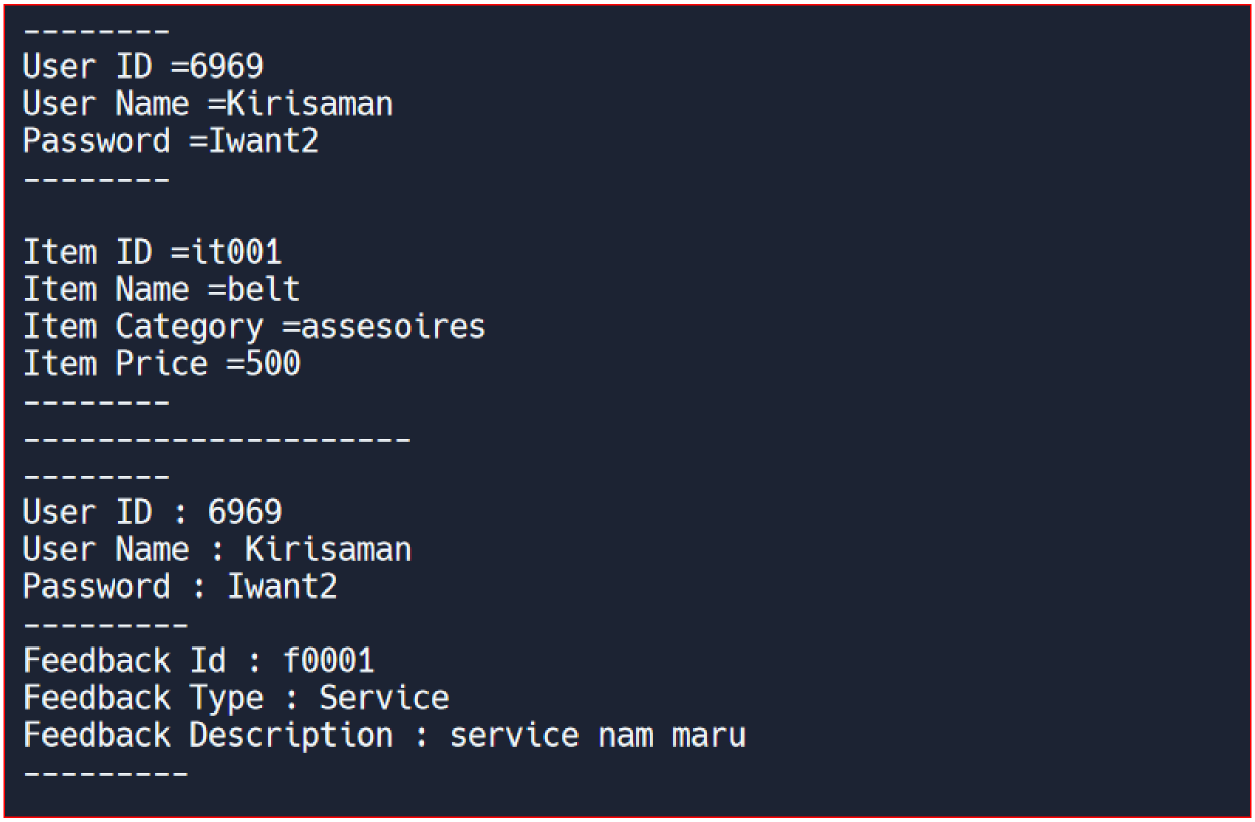
```
--------
Docotr ID -->v0001
Positione -->helth doc
User name -->dr.amarasinhe
Password  -->45rtgs
--------

 Reservation ID =re001
 Reservation Time =10:10:10
 Reservation Date =12-04-2022
--------
------------
--------
User ID =6969
User Name =Kirisaman
Password =Iwant2
--------

 Reservation ID =re001
 Reservation Time =10:10:10
 Reservation Date =12-04-2022
--------
```



```
--------

 Reservation ID =re001
 Reservation Time =10:10:10
 Reservation Date =12-04-2022
--------

 Payment ID =pay001
 Payment Type =credit
 Payment Status =done
 Total Amount =400
--------
--------
--------

 Payment ID =pay002
 Payment Type =paypal
 Payment Status =done
 Total Amount =5000
--------
--------

 Report ID   --> rep001
 Report Type --> income
 sub total   --> 5700

--------
```



```
--------
User ID =6969
User Name =Kirisaman
Password =Iwant2

Order ID =or001
Order Time =12:12:12
Order Date =12-12-2022
---------------------
Order ID =or001
Order Time =12:12:12
Order Date =12-12-2022
--------

 Payment ID =pay002
 Payment Type =paypal
 Payment Status =done
 Total Amount =5000
--------
```



```
--------
User ID =6969
User Name =Kirisaman
Password =Iwant2
--------

Item ID =it001
Item Name =belt
Item Category =assesoires
Item Price =500
--------
---------------------
--------
User ID : 6969
User Name : Kirisaman
Password : Iwant2
---------
Feedback Id : f0001
Feedback Type : Service
Feedback Description : service nam maru
---------
```

Individual Contribution

IT No	Name	Contribution
IT21299902	**ZAKEY.M.S.M.A.**	• **Created USER ->REGISTERED USER inheritance and implemented the code.** • **Created USER ->UNREGISTERED USER and implemented the code.** • **Created USER -> SYSTEM ADMIN inheritance and implemented the code.** • **Created USER -> VET inheritance and implemented the code.** • **Created SYSTEM ADMIN-> REPORT uni-directional association and implemented the code.** • **Created REGISTERED USER -> PET uni-directional association and implemented the code.** • **Help to draw crc cards.** • **Help to identify user requirements and Noun Verb analysis.**
IT21299452	**DILHARA.W.M.A.**	• **Created REGISTERED USER -> RESERVATION bi-directional association and implemented the code.** • **Created RESERVATION -> PAYMENT composition and implemented the code.** • **Created REGISTERED USER -> ODER bi-directional association and implemented the code.** • **Created ORDER -> PAYMENT composition and implemented the code.** • **Created REPORT -> PAYMENT dependency and implemented the code.** • **Created SYSTEM ADMIN -> SERVICE unidirectional and implemented the code.** • **Drew the finalize class diagram.**

		• Help to draw the CRC cards. • Help to identify user requirements and Noun Verb analysis. • Implement the main.cpp.
IT21302480	DILSHAN.O.A.P.	• Created REGISTERED USER -> COMPLAINT bi-directional association and implemented the code. • Created UNREGISTERED USER -> DONATION bi-directional association and implemented the code. • Created REGISTERED USER -> SERVICE uni-directional and implemented the code.
IT21301018	PADUKKA.P.V.G.G.	• Created REGISTERED USER -> FEEDBACK bi-directional association and implemented the code. • Created REGISTERED USER -> DONATION bi-directional association and implemented the code. • Created RESERVATION -> SERVICE uni-directional association and implemented the code. • Help to draw the CRC cards. • Help to identify user requirements and Noun Verb analysis.
IT21301704	DE SILVA.L.M.C.	• Created VET -> RESERVATION bidirectional association and implemented the code. • Created REGISTERED USER -> SHOP ITEM uni-directional association and implemented the code. • Created SHOP ITEM -> ORDER uni-directional association and implemented the code.