Topic : Online Pet Care System

Group no : MLB_04.01_09

Campus : Malabe

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**
IT21258008	Gamage W.G.S.Y	077 065 6737
IT21264016	Helapalla K.O.P.S	077 552 0022
IT21263194	Gamage W.G.T	070 459 4487
IT21256264	Kumara B.D.A.N	076 472 0123
IT21263262	Abeykoon A.M.P.N.	076 059 8525

Contents

Requirements

- The system should function every day.
- To register they must provide their personal details such as Name, Email, Address and telephone number.
- System admin checks the validity of the user registration details
- The website should save their login details as cookies if user allows.
- A registered user should be able to register their pets, customise their appointments and view vet’s notes.
- Only registered users can purchase pet items and make the appointment .
- To make the payment, registered user have to enter payment details such as payment method, card number, name on card, expiry date and CVV number.
- After the payment is validated by the bank, registered users will receive their order id and an email as a confirmation of their order.
- registered user should be able to give ratings about the item to the system.
- A veterinary surgeon can view registered user's appointments and publish the vet's notes to the pet owner. If they want they can reschedule the appointments.
- Pet Store manager visits the login page, enters username and password. Then the system validates the details and the member successfully logs in to the system.
- Pet store manager can check the orders and assign the delivery staff.
- Support inquiries received through the contact form are handled by the pet store manager.
- The pet store manager assigns the system admin to solve customer inquiries.
- Delivery staff processes the order and assigns an available driver to deliver the item.
- System admin logs into the system as the main admin. Managing the database and keeping the system up to the date are done by the system admin. Also the admin takes care of system errors and user account problems.
- system admin can store details of items to the website.
- Both guest customers and registered customers can ask questions
- system admin can generate the monthly reports.

Noun & Verb Analysis

(NOUNS)

- The system should function every day.
- To register they must provide their personal details such as Name, Email, Address and telephone number
- System admin checks the validity of the registration details
- The website should save their login details as cookies if user allows.
- A registered user should be able to register their pets, customise their appointments and view vet's notes.
- Only registered users can purchase pet items and make the appointment .
- To make the payment registered user have to enter payment details such as payment method, card number, name on card, expiry date and CVV number.
- After the payment is validated by the bank, registered users will receive their order id and an email as a confirmation of their order.
- registered user should be able to give ratings about the item to the system.
- A veterinary surgeon can view registered user's appointments and publish the vet's notes to the registered users. If they want they can reschedule the appointments.
- Pet Store manager visits the login page, enters username and password. Then the system validates the details and the member successfully logs in to the system.
- Pet store manager can check the orders and assign the delivery staff.
- Support inquiries received through the contact form are handled by the pet store manager.
- The pet store manager assigns the system admin to solve customer inquiries
- Delivery staff processes the order and assigns an available driver to deliver the item.
- System admin logs into the system as the main admin. Managing the database and keeping the system up to the date are done by the system admin. Also the admin takes care of system errors and user account problems.
- system admin can store details of items to the website.
- Both guest customers and registered customers can ask questions.
- system admin can generate the monthly reports.

Identified Classes

- Guest customer
- Registered User
- Payment
- Item
- Pet store manager
- Order
- Delivery staff
- Driver
- Veterinary surgeon
- Appointments
- Pet

Reasons for rejecting nouns

- An attribute

  personal details (name,email,address,telephone)

  number,username,password),cookies,

  Payment number (card number,CVV number,name on card ,expiry date)

  Ratings,date

  order ID

- Redundant

  Member/user

  Registration details/personal details

  Website/system

  Item/pet item

  frequently asked questions/questions

- An event or an operation

  unlock certain features

  issue prescriptions

- Outside scope of system

  System

  database

  bank

  system admin

- Meta-language

  They

Noun & Verb Analysis

(VERBS)

- The system should function every day.
- To register they must provide their personal details such as Name, Email, Address and telephone number.
- System admin checks the validity of the user registration details
- The website should save their login details as cookies if user allows.
- A registered user should be able to register their pets, customise their appointments and view vet's notes.
- Only registered users can purchase pet items and make the appointment .
- To make the payment, registered user have to enter payment details such as payment method, card number, name on card, expiry date and CVV number.
- After the payment is validated by the bank, registered users will receive their order id and an email as a confirmation of their order.
- registered user should be able to give ratings about the item to the system.
- A veterinary surgeon can view registered user's appointments and publish the vet's notes to the pet owner. If they want they can reschedule the appointments.
- Pet Store manager visits the login page, enters username and password. Then the system validates the details and the pet store manager successfully logs in to the system.
- Pet store manager can check the orders and assign the delivery staff.
- Support inquiries received through the contact form are handled by the pet store manager.
- The pet store manager assigns the system admin to solve customer inquiries.
- Delivery staff processes the order and assigns an available driver to deliver the item.
- System admin logs into the system as the main admin. Managing the database and keeping the system up to the date are done by the system admin. Also the admin takes care of system errors and user account problems.
- system admin can store details of items to the website.
- Both guest customers and registered customers can ask questions
- system admin can generate the monthly reports.

Methods

● Guest customer	- Register to the system by providing details
	search pet store items
	view the pet store
	ask questions
● Registered User	- register their pets by providing details
	login to the system
	view the appointment
	update the appointment
	delete the appointment
	register pet
	purchase pet item
	ask questions
	logout from system
● Payment	- validate payment details
● Item	- add items
	update items
	restock items
● Pet store manager	- check orders and notify the delivery staff
	add order
	handling the support inquiries
	assign the system admin to solve customer inquiries
● Order	- place the order
	status of the order
	confirm the order
	remove the order
● Delivery staff	- process the order
	assign the driver

● Driver	- deliver the item
● Veterinary surgeon	-view registered user's appointments
	issue prescriptions
	reschedule an appointment
	login to the system
	logout from system
● pet	- displayDetails
● Appointments	- display details

CRC cards

Guest customer	
Responsibilities	collaborations
Register to the system Ask questions view the pet store	

Registered User	
Responsibilities	collaborations
ordering items from the pet store register their pets customise their appointment view vet’s note ask questions	order,item pets Appointments

Payment	
Responsibilities	collaborations
validate payment details	Item

Item	
Responsibilities	collaborations
Store the detail of item	

Pet store manager	
Responsibilities	collaborations
Login to the system check the order assign delivery staff	Order Delivery staff

Order	
Responsibilities	collaborations
customise their orders	

Delivery staff	
Responsibilities	collaborations
process the order	Order
assign the driver	Driver

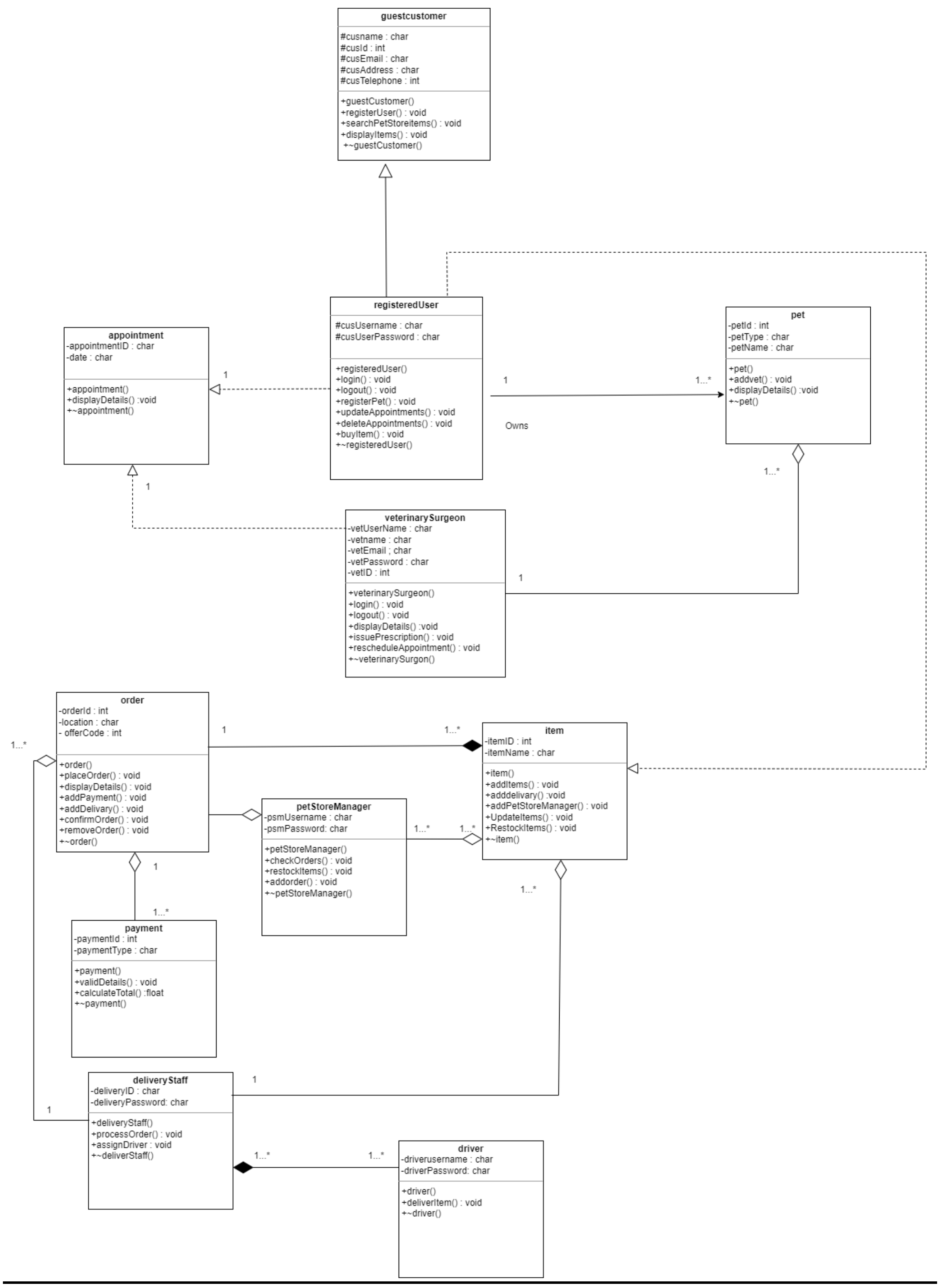
Driver	
Responsibilities	collaborations
deliver the item	Item

Veterinary surgeon	
Responsibilities	collaborations
view registered user's appointments	Appointments
reschedule an appointment	Appointments

Appointments	
Responsibilities	collaborations

Pet	
Responsibilities	collaborations

Class Diagram (UML Notation)



UML Class Diagram Image File

https://drive.google.com/file/d/1S5Ny8vvo_EklUlRnB9KQ-RHfBgcyKR2C/view?usp=sharing coding parts

```
#include <iostream>
#include <cstring>
#define SIZE2 2
using namespace std;

//class implementation
//class appointment

class appointment {
private:
    char appointmentID[30];
    char date[20];
public:
    appointment();
    appointment(const char pappointmentID[30], const char pdate[20]);
    void displayDetails();
    ~appointment();
};

//class guestCustomer
class guestCustomer {

protected:
    int cusId;
    char cusName[50];
    char cusEmail[40];
    char cusAddress[40];
    int cusTelephone;


public:

    guestCustomer();
    guestCustomer(int CID, const char Name[50], const char
        Email[40], const char Address[40], int telephone);
    void registerUser();
    void serchPetStoreItems();
     void displayItems();
   void displayDetails();
    ~guestCustomer();
};

//class veterinarySurgeon
class veterinarySurgeon {
private:
    char vetUserName[40];
    char vetName[40];
    char vetEmail[30];
    char vetPassword[30];
    int vetId;

public:
    veterinarySurgeon(const char pvetUserName[40],const  char
pvetName[40],const char pvetEmail[30], const char pvetPassword[30], int
pvetId);
    void login();
    void logout();
    void issuePrescription();
    void RescheduleAppointment(appointment* appoin);
    void displayDetails();
    ~veterinarySurgeon();
};
//class pet
#define SIZE2 2
class pet {
private:
    int petId;
    char petType[40];
    char petName[40];
    veterinarySurgeon* vet[SIZE2];

public:
    void addvet(veterinarySurgeon* vet1, veterinarySurgeon* vet2);
    pet();
    pet(int ppetId, const char ppetType[40], const char ppetName[40]);
    void displayDetails();
    ~pet();


};
//class payment
class payment
{
private:
    int paymentId;
    char paymentType[40];

public:

    payment(int ppaymentId, const char ppaymentType[40]);
    void validDetails();
    float calculateTotal(float price, float discount);
    ~payment();

};
```

```
//class driver
class driver
{
private:
    char driverUsername[50];
    char driverPassword[50];

public:
    driver();
    driver(const char username[50], const char password[50]);
    void deliverItem();
    ~driver();
};

//class deliveryStaff
#define SIZE 2
class deliveryStaff
{
private:
    char deliverId[10];
    char deliverPassword[50];

    driver* driv[SIZE];

public:
    deliveryStaff();
    deliveryStaff(const char Id[10],const char password[50], const char
driverUsername_1[],const char driverPassword_1[],const char
        driverUsername_2[],const char driverPassword_2[]);

    void processOrder();
    void assignDriver();
    void addDeliveryStaff();
    ~deliveryStaff();

};
//class order
#define SIZE2 2
class order
{
private:
    int orderId;
    char location[40];
    int offerCode;
    deliveryStaff* del[SIZE];
    payment* pay[SIZE2];
```

```
public:
    order();
    order(int porderId, const char plocation[40], int pofferCode);
    void addDelivary(deliveryStaff* del1, deliveryStaff* del2);
    void addPayment(payment* pay1, payment* pay2);
    void deliverItem();
    void placeOrder();
    void displayDetails();
    void confirmOrder();
    void removeOrder();
    ~order();
};

//class petStoreManager
class petStoreManager {

private:
    char psmUsername[40];
    char psmPassword[40];
    order* odr[SIZE];

public:

    petStoreManager(const char ppsmUsername[40], const char ppsmPassword[40]);
    void addOrder(order* odr1, order* odr2);
    void checkOrders();
    void restokItems();
    ~petStoreManager();
};

//class item
#define SIZE 2
class item
{
private:
    int itemId;
    char itemName[40];
    deliveryStaff* del[SIZE];
    order* ord[SIZE];
    petStoreManager* PSM[SIZE];
public:
    item();
    item(int pitemId, const char pitemName[40], int porderID_1, const char
plocation_1[40], int offerCode_1, int porderID_2, const char plocation_2[40],
int offerCode_2);
    void adddelivary(deliveryStaff* del1, deliveryStaff* del2);
    void addPetStoreManager(petStoreManager* PSM1, petStoreManager* PSM2);
    void addItems();
```

```cpp
    void UpdateItems();
    void restockItems();
    ~item();
};

//class registeredUser
class registeredUser : public guestCustomer
{
protected:
    char cusUsername[50];
    char cusPassword[50];
    pet* pe;

public:
    registeredUser(const char pcusUsername[50],const char pcusPassword[50],int
CID ,const char Name[50] , const char Email[40] ,const char Address[40] , int
telephone);
    void registerPet();
    void updateAppointment();
    void deleteAppointment();
    void buyItem(item* itm);
    ~registeredUser();
};

//functions implementation

//appointment class function implementation

appointment::appointment()
{
    strcpy(appointmentID, "");
    strcpy(date, "");


}
appointment::appointment(const char pappointmentID[30], const char pdate[20])
{
    strcpy(appointmentID, pappointmentID);
    strcpy(date, pdate);


}
void appointment::displayDetails()
{
    cout <<"Appointment ID :" << appointmentID << endl;
    cout << "Appintment Date :" << date << endl;
}
```

```
appointment::~appointment()
{

}
//guestCustomer class functions implemnetation

//default constructor

guestCustomer::guestCustomer()
{
    cusId = 0;
    strcpy(cusName, "");
    strcpy(cusEmail, "");
    strcpy(cusAddress, "");
    cusTelephone = 0;

}
// constructor with parameters

guestCustomer::guestCustomer(int cID, const char name[50], const char
email[40], const char address[40], int telephone)
{
    cusId = cID;
    strcpy(cusName, name);
    strcpy(cusEmail, email);
    strcpy(cusAddress, address);
    cusTelephone = telephone;
}

void guestCustomer::registerUser()
{

}

void guestCustomer::serchPetStoreItems()
{

}

void guestCustomer::displayItems()
{


}
void guestCustomer :: displayDetails(){
  cout << "the customer Id is: " << cusId << endl;
    cout << "the customer name is: " << cusName << endl;
    cout << "the customer email is: " << cusEmail << endl;
    cout << "the customer address is : " << cusAddress <<
 endl;
    cout << "the telephone number is : " << cusTelephone << endl;
}
guestCustomer::~guestCustomer()
{

}


//veterinarySurgeon class funcions implementation
veterinarySurgeon::veterinarySurgeon(const char pvetUserName[40], const char
pvetName[40], const char pvetEmail[30],const char pvetPassword[30], int
pvetId) {
    strcpy(vetUserName, pvetUserName);
    strcpy(vetName, pvetName);
    strcpy(vetEmail, pvetEmail);
    strcpy(vetPassword, pvetPassword);
    vetId = pvetId;
}
void veterinarySurgeon::login() {}

void veterinarySurgeon::logout() {}
void veterinarySurgeon::issuePrescription() {}
void veterinarySurgeon::RescheduleAppointment(appointment* appoin) {}
void veterinarySurgeon::displayDetails()
{
    cout << "vet username is: " << vetUserName << endl;
    cout << "vet name is: " << vetName << endl;
    cout << "vet email is: " << vetEmail << endl;
    cout << "vet ID is : " << vetId << endl;

}
veterinarySurgeon :: ~veterinarySurgeon() {}

//pet class functions implementation

void pet::addvet(veterinarySurgeon* vet1, veterinarySurgeon* vet2) {
    vet[0] = vet1;
    vet[1] = vet2;
};

pet::pet()
{

}

pet::pet(int ppetId,const char ppetType[40],const char ppetName[40])
{
    petId = ppetId;
    strcpy(petType, ppetType);
    strcpy(petName, ppetName);

}

void pet::displayDetails()
{
    cout << "Pet ID is: " << petId << endl;
    cout << "Pet type is: " << petType << endl;
    cout << "Pet name is: " << petName << endl;
}

pet::~pet()
{
    for (int i = 0; i < SIZE2; i++)
    {
        delete vet[i];
    }
}

//payment class functions implementation
payment::payment(int ppaymentId,const char ppaymentType[40]) {
    paymentId = ppaymentId;
    strcpy(paymentType, ppaymentType);
}
void payment::validDetails() {}

float payment::calculateTotal(float price, float discount) {
    return (price - discount);
}
payment::~payment()
{

}

//driver class function implementation
driver::driver(const char username[50], const char password[50])
{
    strcpy(driverUsername, username);
    strcpy(driverPassword, password);

}

void driver::deliverItem()
{

}
```

```
driver::~driver()
{

}


//deliveryStaff class functions impelmentation
deliveryStaff::deliveryStaff()
{
    strcpy(deliverId, "");
    strcpy(deliverPassword, "");
}
deliveryStaff::deliveryStaff(const char Id[10],const char password[50], const
char driverUsername_1[],const char driverPassword_1[], const char
    driverUsername_2[], const char driverPassword_2[])
{
    strcpy(deliverId, Id);
    strcpy(deliverPassword, password);

    driv[0] = new driver(driverUsername_1, driverPassword_1);
    driv[1] = new driver(driverUsername_2, driverPassword_2);
}

void deliveryStaff::processOrder()
{

}

void deliveryStaff::assignDriver()
{

}

deliveryStaff::~deliveryStaff()
{
    for (int i = 0; i < SIZE; i++)
    {
        delete driv[i];
    }

}
//order class functions implementation
order::order(int porderId, const char plocation[40], int pofferCode)
{
    orderId = porderId;
    strcpy(location, plocation);
    offerCode = pofferCode;
```

```
}
void order::addDelivary(deliveryStaff* del1, deliveryStaff* del2) {
    del[0] = del1;
    del[1] = del2;
}

void order::addPayment(payment* pay1, payment* pay2) {
    pay[0] = pay1;
    pay[1] = pay2;
}

void order::deliverItem() {}
void order::placeOrder() {}
void order::displayDetails()
{
    cout << "order ID is " << orderId << endl;
    cout << "Location is " << location << endl;
    cout << "offercode is " << offerCode << endl;

}
void order::confirmOrder() {}
void order::removeOrder() {}
order::~order()
{

}

//petStoreManager class functions implementation
petStoreManager::petStoreManager(const char ppsmUsername[40], const char
ppsmPassword[40])
{
    strcpy(psmUsername, ppsmUsername);
    strcpy(psmPassword, ppsmPassword);

}
void petStoreManager::addOrder(order* odr1, order* odr2)
{
    odr[0] = odr1;
    odr[1] = odr2;
}
void petStoreManager::checkOrders()
{

}
void petStoreManager::restokItems()
{
}
petStoreManager::~petStoreManager()
{

}
//item class fuctions implementation
item::item()
{
 itemId=0;
 strcpy(itemName,"");
}
item::item(int pitemId, const char pitemName[40], int porderID_1, const char
plocation_1[40], int offerCode_1, int porderID_2, const char plocation_2[40],
int offerCode_2)
{
    itemId = pitemId;
    strcpy(itemName, pitemName);
    ord[0] = new order(porderID_1, plocation_1, offerCode_1);

    ord[1] = new order(porderID_2, plocation_2, offerCode_2);
}

void item::adddelivary(deliveryStaff* del1, deliveryStaff* del2) {
    del[0] = del1;
    del[1] = del2;
}

void item::addPetStoreManager(petStoreManager* PSM1, petStoreManager* PSM2)
{
    PSM[0] = PSM1;
    PSM[1] = PSM2;
}
void item::addItems() {}
void item::UpdateItems() {}
void item::restockItems() {}
item::~item()
{

}
//registeredCustomer class functions implementation

//constructor with parameteres
registeredUser::registeredUser(const char pcusUsername[50],const char
pcusPassword[50],const int CID ,const char Name[50] ,const char Email[40]
,const char Address[40] , int telephone)
{
    strcpy(cusUsername, pcusUsername);
    strcpy(cusPassword, pcusPassword);
```

```
  cusId = CID;
  strcpy(cusName,Name);
  strcpy(cusEmail,Email);
  strcpy(cusAddress,Address);
  cusTelephone = telephone;



}

void registeredUser::registerPet()
{

}

void registeredUser::updateAppointment()
{

}

void registeredUser::deleteAppointment()
{

}

void registeredUser::buyItem(item* itm)
{

}

registeredUser::~registeredUser()
{

}



//main programme

    int main(){

  //appointment object creation
appointment *n_appointment = new appointment("100", "05/08/2022");
cout<<"Appointment:"<<endl;
cout<<endl;
    n_appointment->displayDetails();
```

```
 //Guest customer object creation
   guestCustomer *n_guestCustomer = new guestCustomer(05 , "Kamal" ,
"kamalbandara2000@gmail.com" , "Maralanda,Kurunegala" , 077-2345656);

   cout <<endl;
  cout << "========================================" <<   endl;
  cout << endl;

   //veterinary surgeon object creation
    veterinarySurgeon *n_veterinarySurgeon = new veterinarySurgeon(
"lakshan55", "lakshan" ,"lakshan77@gmail.com", "5566004", 9988);
       cout<<"Veterinary Surgeon:"<<endl;
      cout<<endl;
   n_veterinarySurgeon ->displayDetails();

   //pet object creation
   pet *n_pet = new pet(006, "Dog", "Shadow");
    cout <<endl;
  cout << "========================================" <<   endl;
  cout << endl;

   cout<<"Pet:"<<endl;
    cout<<endl;
   n_pet->displayDetails();

   //payment object creation
   payment *n_payment = new payment(443335,"credit");

  //driver object creation
  driver *n_driver = new driver("kamal12345","bhv25454");
    cout <<endl;
  cout << "========================================" <<   endl;
  cout << endl;

  //deliveryStaff object creation
  deliveryStaff *n_deliveryStaff = new deliveryStaff("001","abc1245",
"kamal12345", "bhv25454","nuwan546","mnb4515");

  //order object creation
  order * n_order = new order(0012,"colombo",1234);

    deliveryStaff  *d1=new deliveryStaff();
    deliveryStaff *d2=new deliveryStaff();

  n_order->addDelivary(d1,d2);

  payment *pay1=new payment(001,"credit");
  payment *pay2=new payment(002,"debit");
  n_order->addPayment(pay1,pay2);
  cout<<"Order:"<<endl;
  cout<<endl;
 n_order->displayDetails();


  //petStoreManager object creation
   cout <<endl;
  cout << "=======================================" <<   endl;
  cout << endl;

  petStoreManager *pet_m=new petStoreManager("akila456", "ahsvd456");

  order *o1=new order(0012, "colombo", 1234);
  order *o2=new order(0013, "kalutara", 456);

  pet_m ->addOrder(o1,o2);

  //item object creation
  item *item_1=new item(001,"biscuit" , 111, "kalutara", 110, 004, "colombo",
554);

  deliveryStaff *d_staff1=new deliveryStaff();
  deliveryStaff *d_staff2=new deliveryStaff();

  item_1 ->adddelivary(d_staff1,d_staff2);

  petStoreManager *petStore_m1=new petStoreManager("124avishka", "4578shgvd");
   petStoreManager *petStore_m2=new petStoreManager("546nuwan", "45671khg");

   item_1 ->addPetStoreManager(petStore_m1,petStore_m2);
   delete item_1;

   //register User object creation
     cout<<"Register User:"<<endl;
    cout<<endl;
   registeredUser*register_u = new registeredUser("4527kavidu",
"12345ghgyn",5444,"pawan","pawan@gmail.com","pannipitiya",83330990);
    register_u->displayDetails();

    delete n_appointment;
    delete n_guestCustomer;
    delete n_veterinarySurgeon;
    delete n_pet;
    delete n_payment;
    delete n_driver;
    delete n_order;
    delete n_deliveryStaff;
    delete pet_m;
    delete item_1;
    delete register_u;

  return  0;
}
```

Individual contribution

Registration No	Name	Contribution
IT21258008	Gamage W.G.S.Y	Appointment class guestCustomer class
IT21264016	Helapalla K.O.P.S	veterinarySurgeon class payment class
IT21263194	Gamage W.G.T	Pet class RegisteredUser class
IT21256264	Kumara B.D.A.N	Driver class petStoreManager class item class
IT21263262	Abeykoon A.M.P.N.	Order class deliveryStaff class