Sri Lanka Institute of Information Technology

Topic: Online Pharmacy Portal

Group Number : MLB_09.01_01

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**
IT21169380	I.M.H.G. Thuduvage	0771886641
IT21171260	L.H.M.J.C. Lansakara	0742873390
IT21171956	P.B. Sewwandi	0763988972
IT21169144	R.Y.D. Karunarathne	0788095559
IT21172632	S.A.D.S.A. Priyankara	0774749770

Object oriented concepts

B.Sc (Hons) in Information Technology

Exercise 1:

1) User Requirements

1. An Unregistered user in an online pharmacy system needs to first register providing details such as name, mobile number, e-mail address.
2. Then the Registered user can log in to the system using his/her credentials.
3. If there are some mistakes in the registration details the registered user can edit user details in the user account.
4. A Registered User can choose the medicines, medical devices, wellness items and traditional remedies with selecting.
5. A Registered User can upload prescriptions as images or notes with frequency, fulfillment, and new shipping address.
6. A Registered User can enter additional delivery information such as store pickup, express delivery.
7. The Registered User can view the status of the shopping cart and order summery.
8. The Registered User can edit the items from the cart.
9. The Registered User selects the payment method (credit/debit, cash) and enters payment details for place the order.
10. The Registered User confirms the order and after the payment validation, order confirm.
11. The Registered User views the payment successful message and order confirmation.
12. The admin login to the system.
13. The admin generates a listed orders and checks the details.
14. The manager approves the finalized report after processing to the system according to the list of items that the admin generates.
15. The system admin can add new items, update items, edit available status according to the available items that the stock has.

2) Noun & verb Analysis

(Noun are in red and Verb are in blue),

1. An Unregistered user in an online pharmacy system needs to first register providing details such as name, mobile number, e-mail address.
2. Then the Registered user can log in to the system using his/her credentials.
3. If there are some mistakes in the registration details the registered usercan edit user details in the user account.
4. A Registered User can choose the medicines, medical devices, wellness items and traditional remedies with selecting.
5. A Registered User can upload prescriptions as images or notes with frequency, fulfillment, and new shipping address.
6. A Registered User can enter additional delivery information such as store pickup, express delivery.
7. The Registered User can view the status of the shopping cart and order summery.
8. The Registered User can edit the items from the cart and process to the checkout.
9. The Registered User selects the payment method (credit/debit, cash) and enters payment details for place the order.
10. The Registered User confirms the order and after the payment validation, order confirm.
11. The Registered User views the payment successful message and order confirmation.
12. The admin login to the system.
13. The admin generates a listed orders and checks the details.
14. The manager approves the finalized report after processing to the system according to the list of items that the admin generates.
15. The system admin can add new items, update items, edit available status according to the available items that the stock has.

3) Identified classes using Noun Analysis,

Identified classes.

- Registered User
- Item
- order
- Payment
- Card - (Inherited from Payment)
- Cart
- Report
- Delivery

Exercise 2:

CRC Cards for the Online Pharmacy portal system

Registered User	
Responsibility	**Collaborators**
Store login details	
Upload Prescription	
Add items	Cart
Update Profile	
Cancel Order	Order

Item	
Responsibility	**Collaborators**
Display ordered medicine details	Order
Update medicines details	

Order	
Responsibility	**Collaborators**
View Order	
Store details about available stock	Item
Store details about who requested the order	
Add order details	Item
Available items status	Item
Confirm Order	Payment

Payments	
Responsibility	**Collaborators**
Calculate payment	Order
View payment	Registered User
Store Payment Details	
Validate	

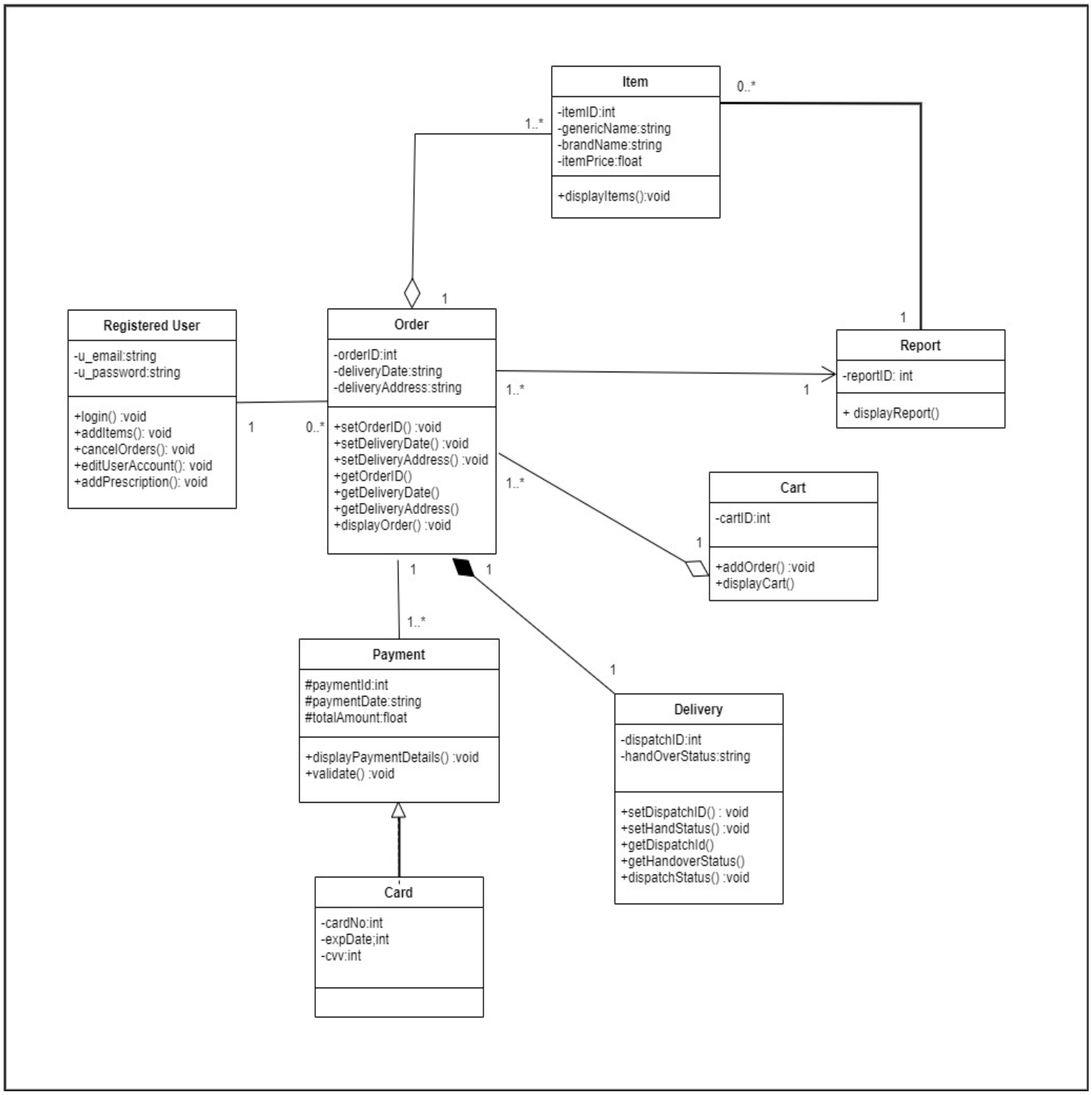
Card - (Inherited from Payment)	
Responsibility	**Collaborators**
Store card payment Details	Payment

Cart	
Responsibility	**Collaborators**
Edit Order	Order
View payment	Registered User
Store Payment Details	
Validate	

Report	
Responsibility	**Collaborators**
List of Orders	Order
List of available Stock	Item
Cash flow	Payment

Delivery	
Responsibility	**Collaborators**
Item Identity	Order
Check Receiver Details	Registered User
Total Qty	Order
Payment Details (already done or not)	Payment

Exercise 3:
<u>Class diagram for the Online Pharmacy portal system</u>



Exercise 04

C++ Coding

RegisteredUser.h

```
#pragma once
#include <string>
#include "order.h"
class Order;

#define SIZE 2

using namespace std;

class RegisteredUser{
  private:
    string u_emailAddress;
    string u_password;
    Order *order[SIZE];

  public:
    RegisteredUser();
    void login(string email, string pw);
    void addItems();
    void cancelOrders();
    void editUserAccount();
    void addPrescription();
    ~RegisteredUser();
};
```

RegisteredUser.cpp

```
#include <string>
#include <iostream>
#include "registeredUser.h"

using namespace std;


RegisteredUser::RegisteredUser(){

}

void RegisteredUser::login(string email, string pw){
  u_emailAddress = email;
  u_password = pw;
}

void RegisteredUser::addItems(){

}
void RegisteredUser::cancelOrders(){

}
void RegisteredUser::editUserAccount(){

}
void RegisteredUser::addPrescription(){

}

RegisteredUser::~RegisteredUser(){
  cout << "Registered User Deleted";
}
```

Item.h

```
#pragma once
#include<iostream>
#include <string>
#include "report.h"
class Report;

using namespace std;

class Item{
  private:
    int itemId;
    string genericName;
    string brandName;
    float itemPrice;
    Report *report;

  public:
    Item();
    Item(int id, string genName, string brand, float price);
    void displayItems();
    ~Item();


};
```

Item.cpp

```cpp
#include <string.h>
//#define SIZE 2
#include "item.h"

using namespace std;

Item::Item(){

}


Item::Item(int id, string genName, string brand, float price){
  itemId = id;
  genericName = genName;
  brandName = brand;
  itemPrice = price;
}

void Item::displayItems(){
  cout << "Item ID: " << itemId << endl;
  cout << "Generic Name: " << genericName << endl;
  cout << "Brand Name: " << brandName << endl;
  cout << "Item Price: " << itemPrice << endl;
  cout << "**************************************************" <<
endl;
};

Item::~Item(){
  cout << "Items are Deleted" << endl;
}
```

Order.h

```
#pragma once
#include <string>
#include "item.h"
#include "registeredUser.h"
#include "payment.h"
#include "delivery.h"
#define SIZE 2

class RegisteredUser;
class Payment;

class Order{
  private:
    Item *item[SIZE];
    RegisteredUser *regUser;
    Payment *payment[SIZE];
    Delivery *delivery;
    int orderId;
    string deliveryDate;
    string deliveryAddress;

  public:
    Order();
    Order(int id, string date, string address);
    Order (int id, string status);
    void setOrderId(int id);
    void setDeliveryDate(string date);
    void setDeliveryAddress(string address);
    int getOrderId();
    string getDeliveryDate();
    string getDeliveryAddress();
    void displayOrder();
    void addItem(Item *item1, Item *item2);
    ~Order();
};
```

Order.cpp

```
#include <iostream>
#include <string>

#include "order.h"
#include "item.h"
#include "delivery.h"

class Delivery;

using namespace std;

Order::Order(){}

Order::Order(int id, string status){
  Item *item[2];
  delivery = new Delivery(1, "abc");


}


Order::Order(int id, string date, string address){
  orderId = id;
  deliveryDate = date;
  deliveryAddress = address;
}


void Order::setOrderId(int id){

}
void Order::setDeliveryDate(string date){

}
void Order::setDeliveryAddress(string address){

}

int Order::getOrderId(){
  return orderId;
}
```

```
string Order::getDeliveryDate(){
  return deliveryDate;
}
string Order::getDeliveryAddress(){
  return deliveryAddress;
}

void Order::addItem(Item *item1, Item *item2){
  item[0] = item1;
  item[1] = item2;
}

void Order::displayOrder(){
  cout << "Order Id: " << orderId << endl;
  cout << "Delivery Date : " << deliveryDate << endl;
  cout << "Delivery Address : " << deliveryAddress << endl;
  cout << "_________________________________________________" <<
endl;


 for(int i = 0; i < SIZE; i++){
    item[i]  -> displayItems();
  }

}

Order::~Order(){
  cout << "Orders are Deleted." << endl;
}
```

Payment.h

```
#pragma once
#include <string>
#include "order.h"
class Order;

using namespace std;

class Payment{
  protected:
    int paymentId;
    string paymentDate;
    float totalAmount;
    Order *order;

  public:
    Payment();
    Payment(int id, string date, float total);

    void displayPaymentDetails();
    void validate();
    ~Payment();
};
```

Payment.cpp

```cpp
#include <iostream>
#include "payment.h"

using namespace std;

Payment::Payment(){

}

Payment::Payment(int id, string date, float total){
  paymentId = id;
  paymentDate = date;
  totalAmount = total;
}

void Payment::displayPaymentDetails(){
  cout << "Payment ID: " << paymentId << endl;
  cout << "Total Amount : " << totalAmount << endl;
}

void Payment::validate(){}

Payment::~Payment(){
  cout << "Payments are Deleted";
}
```

Card.h

```
#pragma once
#include "payment.h"

class Card: public Payment{
  private:
    int cardNo;
    int expDate;
    int cvv;
};
```

Cart.h

```
#pragma once
#include "order.h"

#define SIZE 2

class Cart{
  private:
    Order *order[SIZE];
   // int cartId;

  public:
    Cart();
    void addOrder(Order *order1, Order *order2);
    void displayCart();
    ~Cart();

};
```

Cart.cpp

```
#include <iostream>
#include <string>
#include "cart.h"
#include "order.h"


#define SIZE 2

using namespace std;

Cart::Cart(){
  Order *order[2];
}

void Cart::addOrder(Order *order1, Order *order2){
  order[0] = order1;
  order[1] = order2;
}

void Cart::displayCart(){
  for (int i = 0; i < SIZE; i++){
    order[i]  -> displayOrder();
  }
}

Cart::~Cart(){
  cout << "Cart is Deleted" << endl << endl;
}
```

Report.h

```
#pragma once
#include "item.h"

#define SIZE 2

class Item;

class Report{
  private:
    int reportId;
    Item *item[SIZE];

  public:
    Report(int id);
    void displayReport();
    ~Report();
};
```

Report.cpp

```
#include <iostream>
#include "report.h"

using namespace std;

Report::Report(int id){}

void Report::displayReport(){}

Report::~Report(){
  cout << "Reports are deleted";
}
```

Delivery.h

```
#pragma once
#include <string>

using namespace std;

class Delivery{
  private:
    int dispatchId;
    string handoverStatus;

  public:
    Delivery(int id, string status);
    void setDispatchId(int id);
    void setHandoverStatus(string status);
    int getDispatchId();
    string getHandoverStatus();
    void dispatchStatus();
    ~Delivery();
};
```

Delivery.cpp

```
#include "delivery.h"

Delivery::Delivery(int id, string status){
  dispatchId = id;
  handoverStatus = status;
}

void Delivery::setDispatchId(int id){
  dispatchId = id;
}
void Delivery::setHandoverStatus(string status){
  handoverStatus = status;
}

int Delivery::getDispatchId(){
  return dispatchId;
}
string Delivery::getHandoverStatus(){
  return handoverStatus;
}

void Delivery::dispatchStatus(){}
```

main program

```
#include "card.h"
#include "cart.h"
#include "delivery.h"
#include "item.h"
#include "order.h"
#include "payment.h"
#include "registeredUser.h"
#include "report.h"
#include <iostream>
#include <string>
#define SIZE 2

using namespace std;

int main() {
  Order *order = new Order();
  Cart *c1 = new Cart();
  Item *item = new Item();

  Order *o1 = new Order(001, "05/25", "Kandy");
  Order *o2 = new Order(002, "06/02", "Galle");

  c1->addOrder(o1, o2);

  Item *i1 = new Item(100, "Panadol", "Indian", 200.00);
  Item *i2 = new Item(101, "Strepsills", "Indian", 35.00);
```

```
    Item *i3 = new Item(103, "Niwaran", "Indian", 25.00);
    Item *i4 = new Item(104, "Belcid", "Finland", 450.00);

    order->addItem(i1, i2);

    c1->addOrder(o1, o2);



    delete c1; // delete cart

    o1->addItem(i1, i2);
    o2->addItem(i3, i4);

    o1->displayOrder();
    o2->displayOrder();

    delete order;

    cout << "\n";

    i1->displayItems();
    i2->displayItems();
    i3->displayItems();
    i4->displayItems();

    // delete items
    delete i1;
    delete i2;

    // delete orders
    delete o1;
    delete o2;

    return 0;
}
```