IT1050 – Object Oriented Concepts

Topic : Online Fashion Store

Group no : MLB_01.01_04

Campus : Malabe

Submission Date : 15/10/2021

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
IT21033414	Pabasara Ganegoda	071 0178881
IT21034336	Dushmani Amarakoon	077 3650098
IT21041716	Sithanga Rashmika	076 3223349
IT21041334	Viraj Dissanayake	077 6851959
IT21040726	Harshangi Perera	071 8558347

System requirement

1. User can register to system by providing registration details.
2. User can login into the system by using username and password.
3. Unregistered customer needs to create an account if he needs to purchase any item.
4. Both the users can view, search, check ratings and feedbacks of items.
5. Registered customer can add one or more item to the cart.
6. Item price can be paid through Debit card or with credit card.
7. Customers recheck cart, can add more items or can remove items from the cart.
8. Finally Registered customer confirm the payment.
9. Customer can send feedback and inquiries.
10. Admin can add new items and delete items from the system.
11. Admin can check feedback, inquiries and replies feedback, inquiries.
12. Admin create reports (total sales, inventory reports) end of the month.

Noun verb analysis

Nouns

1. User can register to system by providing registration details.
2. User can login into the system by using username and password.
3. Unregistered customer needs to create an account if he needs to purchase any item.
4. Both the users can view, search, check ratings and feedbacks of items.
5. Registered customer can add one or more item to the cart.
6. Item price can be paid through Debit card or with credit card.
7. Registered customers recheck cart, can add more items or can remove items from the cart.
8. Finally Registered customer confirm the payment.
9. Registered customer can send feedback and inquiries.
10. Admin can add new items and delete items from the system.
11. Admin can check feedback, inquiries and replies feedback, inquiries.
12. Admin create reports (total sales, inventory reports) end of the month.

Verbs

1. User can **register** to system by ~~**providing**~~ registration details.
2. User can **login** into the system by using username and password.
3. Unregistered customer needs to ~~**create**~~ an account if he needs to purchase any item.
4. Both the users can **view**, **search**, **check** ratings and feedbacks of items.
5. Registered customer can **add** one or more item to the cart.
6. Item price can be **paid** through Debit card or with credit card.
7. Registered customers **recheck** cart, can **add** more items or can **remove** items from the cart.
8. Finally customer **Registered confirm** the payment.
9. Registered customer can **send** feedback and inquiries.
10. Admin can **add** new items and **delete** items from the system.
11. Admin can **check** feedback, inquiries and **replies** feedback, inquiries.
12. Admin **create** reports (total sales, inventory reports) end of the month.

Identifying classes using noun/ verb analysis

- User - class
- Admin - class
- Registered customer - class
- System - out of scope
- Member - redundant
- System - out of scope
- Username - attribute
- Password - attribute
- Unregistered customer - class
- He - Meta-language
- Account - class
- Item - class
- User - redundant
- Rating / Feedback/ Inquiries - class
- Item - redundant
- Registered customer - redundant
- Cart - class
- Item - redundant
- Item price - attribute
- Debit card - attribute of payment
- Credit card - attribute of payment
- Registered customer - redundant
- Cart - redundant
- Items - redundant
- Items - redundant
- Cart - redundant
- Payment - class

- Registered customer - redundant
- Rating / Feedback/ Inquiries - redundant
- Admin - redundant
- Items - redundant
- Items - redundant
- System - out of scope
- Admin - redundant
- Feedback - redundant
- Inquiries - redundant
- feedbacks - redundant
- inquiries - redundant
- admin - redundant
- sales report - attribute
- inventory report - attribute
- report - class

CRC Cards

Class name: User	
responsibilities	**collaborations**
Register to the system	Admin, Unregistered customer
Login to system	Admin, Registered customer

Class name: Admin	
responsibilities	**collaborations**
Add Items	Item
Delete Items	Item

Class name: Registered Customer	
responsibilities	**collaborations**
View/ check/ send ratings & feedbacks	Rating / Feedback/ Inquiries
Add/ Remove items to cart	Item

Class name: Unregistered Customer	
responsibilities	**collaborations**
Provide registration details	
Create a new account	

Class name: Account	
responsibilities	**collaborations**
Get registration details	Admin, Unregistered customer
Provide username & password	

Class name: Item	
responsibilities	**collaborations**
Add new items	
Add to cart	Cart
Delete items	
Provide report item details	Report
Display item details	

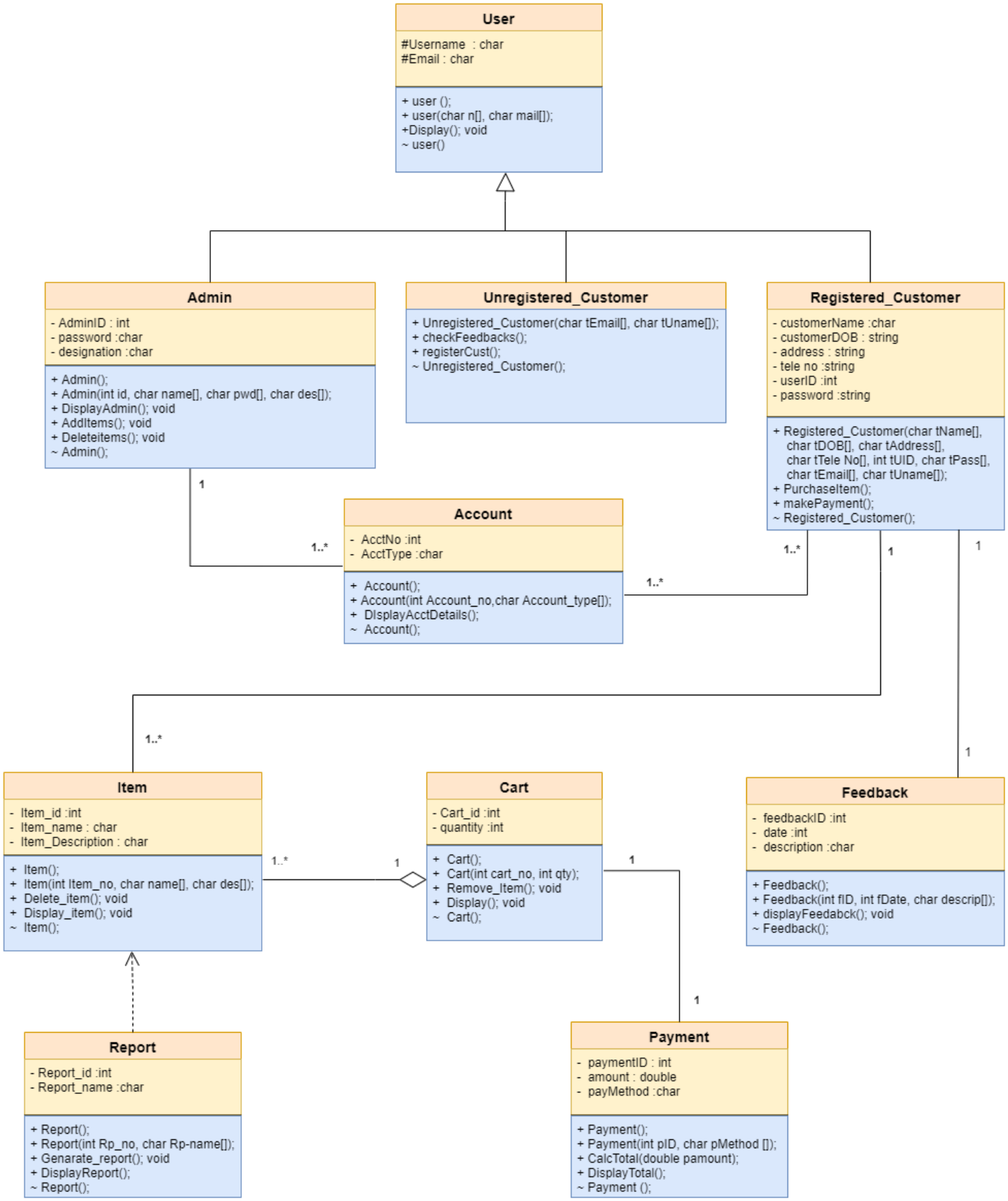
Class name: Rating / Feedback/ Inquiries	
responsibilities	**collaborations**
Receive feedbacks/ratings	Registered Customer
Display feedbacks/ratings	

Class name: Cart	
responsibilities	**collaborations**
Add/ remove items	Item
Display details	
Provide payment details	Payment

Class name: Payment	
responsibilities	**collaborations**
Making new payment	
Check payment details	Registered customer, Cart
Get purchase details	

Class name: Report	
responsibilities	**collaborations**
Generate sales report	
Generate inventory report	
Display report	

UML diagram


User
#Username : char
#Email : char
+ user ();
+ user(char n[], char mail[]);
+Display(); void
~ user()
Admin
- AdminID : int
- password :char
- designation :char
+ Admin();
+ Admin(int id, char name[], char pwd[], char des[]);
+ DisplayAdmin(); void
+ AddItems(); void
+ Deleteitems(); void
~ Admin();
Unregistered_Customer
+ Unregistered_Customer(char tEmail[], char tUname[]);
+ checkFeedbacks();
+ registerCust();
~ Unregistered_Customer();
Registered_Customer
- customerName :char
- customerDOB : string
- address : string
- tele no :string
- userID :int
- password :string
+ Registered_Customer(char tName[], char tDOB[], char tAddress[], char tTele No[], int tUID, char tPass[], char tEmail[], char tUname[]);
+ PurchaseItem();
+ makePayment();
~ Registered_Customer();
Account
- AcctNo :int
- AcctType :char
+ Account();
+ Account(int Account_no,char Account_type[]);
+ DIsplayAcctDetails();
~ Account();
Item
- Item_id :int
- Item_name : char
- Item_Description : char
+ Item();
+ Item(int Item_no, char name[], char des[]);
+ Delete_item(); void
+ Display_item(); void
~ Item();
Cart
- Cart_id :int
- quantity :int
+ Cart();
+ Cart(int cart_no, int qty);
+ Remove_Item(); void
+ Display(); void
~ Cart();
Feedback
- feedbackID :int
- date :int
- description :char
+ Feedback();
+ Feedback(int fID, int fDate, char descrip[]);
+ displayFeedabck(); void
~ Feedback();
Report
- Report_id :int
- Report_name :char
+ Report();
+ Report(int Rp_no, char Rp-name[]);
+ Genarate_report(); void
+ DisplayReport();
~ Report();
Payment
- paymentID : int
- amount : double
- payMethod :char
+ Payment();
+ Payment(int pID, char pMethod []);
+ CalcTotal(double pamount);
+ DisplayTotal();
~ Payment ();
1
1..*
1..*
1..*
1
1
1..*
1
1..*
1
1
1

Codes

Header Files

```
//Declaring user class
#pragma once
class user{
  protected:
  char Username[20];
  char Email[30];

  public:
  //constructors and distructors
  user(){};
  user(const char n[], const char mail[]);
  ~user();

  //other methods
  void Display();
   };



//Declaring Admin class
#include"user.h"
class Admin:public user{
  private:
  int AdminID;
  char password[8];
  char designation[10];

  public:
  //constructors and distructors
  Admin(){};
  Admin(int id,const char name[],const char pwd[],const char des[]);
  ~Admin();

  //other methods
  void DisplayAdmin();
  void AddItems(int qty);
  void DeleteItems(int qty);

   };



//Declaring unregistered_Customer class
#pragma once
#include "User.h"
class Unregistered_Customer : public User
{
public:
       //Constructor and destructor
       Unregistered_Customer(const char tEmail[300], const char tUname[50]);
       ~Unregistered_Customer();

       //other methods
       void checkFeedback();
       void registerCust();
   };
```

```
//Declaring Registered_Customer class
#pragma once
#include "User.h"
#include "Item.h"
#include "Account.h"
class Registered_Customer : public User
{
protected:
	char customerName[50];
	char customerDOB[20];
	char address[200];
	char teleNo[15];
	int userID;
	char password[50];
public:
	//Constructor and Destructor
	Registered_Customer(const char tName[50], const char tDOB[20], const char
tAddress[200], const char tTellNo[15], int tUID, const char tPass[50], const char
tEmail[300], const char tUname[50]);
	~Registered_Customer();

	//Other Methods
	void purchaseItems();
	void makePayments();
    };
```

```
//Declaring Account class
#include "Registered_customer.h"
#include "Admin.h"
class Account {

private:
	int Account_no;
	char Account_type[20];
	Registered_customer* cs1;
	Admin* ad1;

public:
	Account();
	Account(int AcctNo, const char AcctType[], Registered_customer* cs, Admin* ad);
	void DisplayAccDetails();
	~Account();
    };
```

```
//Declaring Feedback class
#include "Registered_customer.h"
class Feedback
{
private:
	int feedbackID;
	int date;
	char description[20];
	Registered_customer* cs1;
public:
	Feedback();
	Feedback(int fID, int fDate, const char descrip[], Registered_customer* cus);
	void displayFeedback();
	~Feedback();
    };
```

```
//Declaring Item class
#pragma once
#include "Cart.h"
#include "Registered_customer.h" // there is not this header in my program, this part
done by another guy

class Item
{
private:
	int Item_id;
	char Item_name [30];
	char Item_Decsription[50];
	Registered_customer * cs1;


public:
	Item();
	Item(int Item_no, const char name[], const char des[], Registered_customer *
cus );// destructor  association relationship with registered customer
	void Delete_item();
	void Display_item();
	~Item();//destructor
    };
```

```
//Declaring Cart class
#pragma once
#include "Payment.h"// there is not this header in my program, this part done by
another guy

#include "Item.h"
class Cart
{
private:
	int Cart_id;
	int quantity;
	Payment * pay1;
	Item * Itm;


public:
	Cart();// default constructor
	Cart( int cart_no , int qty , Payment* pay , Item * Im); // overload
constructor and association relationship with payment class and aggregration
relationship with item class
	void Remove_Item();
	void Display();
	~Cart(); //destructor
    };
```

```
//Declaring Payment class
#include "Cart.h"

class Payment
{
private:
	int paymentID;
	double amount;
	char payMenthod[10];
	Cart* c_art1;
public:
	Payment();
	Payment(int pID, const char pMethod[], Cart* c_art);
	void CalcTotal(double pamount);
	void displayTotal();
	~Payment();
    };
```

```
//Declaring Report clas
#include "Item.h"
class Report{

private :
  int Report_id;
  char Report_name[30];



public :
  Report();
  Report(int Rp_no, const char Rp_name[]);
  void Genarate_report(Item *i);
  void DisplayReport();
  ~Report();
    };
```

.CPP Files

```
//implementation of user class
#include"user.h"
#include<iostream>
#include<cstring>

using namespace std;

user::user(const char n[], const char mail[]){
  strcpy(Username,n);
  strcpy(Email,mail);

}

user::~user(){
  cout<<"User "<<Username<<" deleted"<<endl;
}

void user::Display(){
  cout<< "Username : " <<Username<<endl;

    }
```

```
//implementation of Admin class
#include"Admin.h"
#include<iostream>
#include<cstring>

using namespace std;
Admin::Admin(int id,const char name[],const char pwd[],const char des[]){
  AdminID = id;
  strcpy(Username,name);
  strcpy(password,pwd);
  strcpy(designation,des);

}

Admin::~Admin(){
  cout<< "Admin "<< AdminID << " deleted"<<endl;
  }

void Admin::DisplayAdmin(){
  cout<< "Admin ID : "<<AdminID<<endl;
  Display();
  cout<< "Password : " <<password<<endl;
  cout<< "Designation : "<<designation<<endl;

}

void Admin::AddItems(int qty){
   cout<<qty<<" items are added"<<endl;

}

void Admin::DeleteItems(int qty){
  cout<<qty<<" items are deleted"<<endl;

   }




//implementation of Unregistered_Customer class
#include "Unregistered_Customer.h"
#include <iostream>
#include <cstring>

using namespace std;

Unregistered_Customer::Unregistered_Customer(const char tEmail[300], const char
tUname[50]) : User (tEmail, tUname) {
       cout << "Unregistered Customer" << endl;
}

Unregistered_Customer::~Unregistered_Customer() {
       cout << "Unregistered Customer Removed" << endl;
   }
```

```
//implementation of Registered_Customer class
#include "Registered_Customer.h"
#include <iostream>
#include <cstring>

using namespace std;

Registered_Customer::Registered_Customer(const char tName[50], const char tDOB[20],
const char tAddress[200], const char tTellNo[15], int tUID, const char tPass[50],
const char tEmail[300], const char tUname[50]) : User(tEmail, tUname) {
	strcpy_s(customerName, tName);
	strcpy_s(customerDOB, tDOB);
	strcpy_s(address, tAddress);
	strcpy_s(teleNo, tTellNo);
	strcpy_s(username, tUname);
	strcpy_s(password, tPass);
}

Registered_Customer::~Registered_Customer() {
	cout << "Registered Customer " << username << " Deleted" << endl;
}



//implementation of Account class
#include <iostream>
#include <cstring>
#include "Account.h"
using namespace std;

Account::Account()
{
  Account_no = 0;
  strcpy (Account_type,"");

}
Account::Account(int AcctNo , const char AcctType[] ,Registered_customer * cs , Admin
* ad)
{
  Account_no = AcctNo;
  strcpy(Account_type, AcctType);
  cs1 = cs;
  ad1 = ad;
}
void Account:: DisplayAccDetails()
{
  cout<<"................................."<<endl;
  cout<<"Account number :" <<Account_no<< endl;
  cout<<"Account Type :" <<Account_type<< endl;

  cout<<"................................."<< endl;

}
Account:: ~Account()
{
  cout<<"Account destructor called !!"<<endl;
}
```

```
// implementation of Feedback class
#include "Feedback.h"
#include<iostream>
#include<cstring>
using namespace std;

Feedback::Feedback()
{
}

Feedback::Feedback(int fID, int fDate, const char descrip[], Registered_customer* cus)
{
	feedbackID = fID;
	date = fDate;
	strcpy_s(description, descrip);
}

void Feedback::displayFeedback()
{
}

Feedback::~Feedback()
{
	cout << "Feedback destructor caled !" << endl;
}
```

```
// implementation of Item class
#include "Item.h"
#include<iostream>
#include<cstring>
using namespace std;
Item::Item()
{
	Item_id = 0;
	strcpy_s(Item_name ,"");
	strcpy_s(Item_Decsription , "");

}

Item::Item(int Item_no, const char name[], const char des[], Registered_customer *
cus)
{
	Item_id = Item_no;
	strcpy_s(Item_name, name);
	strcpy_s(Item_Decsription, des);
	cs1 = cus;
}

void Item::Delete_item()
{

}
```

```
void Item::Display_item()
{
	cout << "....................................." << endl;
	cout << " Item ID :" << Item_id << endl;
	cout << " Item Name :"<< Item_name <<endl;
	cout << " Item_Decsription" << Item_Decsription << endl;
	cout << "....................................." << endl;

}

Item::~Item()
{
	cout << "Item destructore called !!" << endl;
}



// implementation of Cart class
#include "Cart.h"
#include<iostream>
using namespace std;
Cart::Cart()
{
	 Cart_id = 0;
	 quantity = 0;


}

Cart::Cart(int cart_no, int qty , Payment * pay , Item* Im) //
{
	Cart_id = cart_no;
	quantity = qty;
	pay1 = pay;
	Itm = Im;
}

void Cart::Remove_Item()
{
}


void Cart::Display()
{
	cout << "....................................." << endl;
	cout << "Cart ID :" << Cart_id << endl;
	cout << "Quantity :" << quantity << endl;
	pay1->Displaytotal();
	cout << "....................................." << endl;
}

Cart::~Cart()
{
	cout << "Cart destructore called !!" << endl;
}
```

```
//implementation of Payment class
#include "Payment.h"
#include "Cart.h"
#include<iostream>
#include<cstring>
using namespace std;

Payment::Payment()
{
}

Payment::Payment(int pID, const char pMethod[], Cart* c_art)
{
	paymentID = pID;
	strcpy_s(payMenthod, pMethod);
	c_art1 = c_art;
}

void Payment::CalcTotal(double pamount)
{
	amount = pamount;
	int total;
	total = c_art1 * amount;
}

void Payment::displayTotal()
{
	cout << "Total Amunt is :" << total << endl;
}

Payment::~Payment()
{
	cout << "Payment destructor caled !" << endl;
}
```

```
// implementation of Report class
#include <iostream>
#include <cstring>
#include "Report.h"
using namespace std;

  Report::Report()
  {
    Report_id = 0;
    strcpy (Report_name,"");
  }
  Report::Report(int Rp_no ,const char Rp_name[])
  {
    Report_id = Rp_no;
    strcpy (Report_name,Rp_name);
  }
  void Report::Genarate_report(Item *i)
  {

  }
```

```
void Report::DisplayReport()
  {
    cout<<"................................."<<endl;
    cout<<"Report ID :" <<Report_id<< endl;
    cout<<"Report Name :" <<Report_name<< endl;

  cout<<"................................."<< endl;

  }
  Report::~Report()
  {

  }
```

Main program

```
#include <iostream>
#include<cstring>
#include"user.h"
#include"Admin.h"
#include"Unregistered_customer.h"
#include"Registered_customer.h"
#include"Account.h"
#include"Feedback.h"
#include"Item.h"
#include"Cart.h"
#include"Payment.h"
#include"Report.h"


using namespace std;

int main() {
  user*u1;
  u1 = new user("dush1","dush@gmail.com");
  u1->Display();

  Admin*a1;
  a1 = new Admin(1,"jak1","123Cool","Admin");
  a1->DisplayAdmin();

  a1->AddItems(10);
  a1->DeleteItems(2);

  Registered_Customer* RC1 = new Registered_Customer("Harshangi", "25/05/2001",
"piliyandala", "7638456483", 0001, "Harshi123", "harshangiperera@gmail.com",
"Harshi001");

  Unregistered_Customer* UC1 = new
Unregistered_Customer("dushmaniamarakoon@gmail.com", "dushii");

  Registered_Customer* cus;
  Payment* pmt1;
  Item* it1;

  // cart object create
  Cart* crt1;
  crt1 = new Cart(121212, 5, pmt1, it1);
```

```
    crt1->Remove_Item();
    crt1->Display();

    // Item object create
    Item* it1;
    it1 = new Item(101, "Nike T-Shirt", "blue color collar t shirt ", cus);
    it1->Delete_item();
    it1->Display_item();

    Registered_customer* cst;
    Admin* adm;


    Report* rep1;
    rep1 = new Report(100, "Stock");
    rep1->Genarate_report(Item * i);
    rep1->DisplayReport();

    Account* acct;
    acct = new Account(500, "admin", cst, adm);
    acct->DisplayAccDetails();

    Payment p1;
    Feedback f1;


    p1.displayTotal();

    f1.displayFeedback();


    //dynamic object delete
    delete u1;
    delete a1;

    delete RC1;
    delete UC1;

    delete it1;
    delete crt1;

    delete rep1;
    delete acct;

    return 0;

}
```