Topic : Online Salon Booking System

Group no : MLB_01.01_07

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

Submission Date : 30th of May 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**
IT20605902	S.D. Samarasinghe	0702449678
IT20299552	L.M. Rubasinghe	0776400813
IT20604844	W.G.T.D. Thilakarathne	0776668613
IT20605834	H.R. Kotuwe Gedara	0712011693
IT20606060	Wanni Arachchige H.S.	0772137060

Exercise 01

01) Requirement Analysis

- User needs to register to the system as a Customer by entering valid details. If is not a registered user cannot book services and purchase products. After the registering they becomes the Customers of the system.
- User can view available services and products.
- User can add available products to shopping cart according to his preferences.
- Customer can select available services and products. And add products to shopping cart.
- Customer can fill the inquiry form according to their preferences and do a booking, they can choose Beauticians before booking services. One customer can do a one booking at once.
- Registered user can select products.
- Customer can add products to the shopping cart. One customer can add many products to the shopping cart at one time.
- After the Booking and Purchasing items, customers should make their advance or full payment using credit or debit cards.
- A customer can cancel the booking by visiting his profile and confirming the cancelation. He can request for refund within 5 days.
- The System Administrator can add new services, new items, delete and update and available services and products.
- He can reply for feedbacks and generate reports.
- Job seeker is a guest. He can search for jobs and apply for available job opportunities by filling the respective career form.
- Manager should login to the system as an admin. He can check income, manage payments, and check reports.
- Manager can assign new Beauticians.

02) Noun & Verb Analysis

- Nouns are Blue
- Verbs are Red

- Any User can register to the System by providing email and password.
- User can search services and products.
- User can view products and services, also User can add products to shopping cart.
- Customer can search services and products.
- Customer can select available products. And add products to shopping cart.
- Customer can fill the inquiry form according to their preferences and do a booking, they can choose Beauticians before booking services.
- After the Booking and Purchasing items, customers should make their advance or full payment by using credit or debit cards.
- A customer can cancel the booking by visiting his profile and confirming the cancelation. He can request for refund within 5 days.
- The System Administrator can add new services, new items, delete and update and available services and products.
- System Administrator can reply for feedbacks and generate reports.
- Job seeker can search for jobs and apply for available job opportunities by filling the respective career form.
- Manager should login to the system as an admin. He can check income, manage payments, and check reports.
- Manager can assign new Beauticians.

03) Identify Classes using Noun Verb Analysis

Nouns

- User
- System
- Email
- Password
- Customer
- Services
- Products
- Inquiry Form
- Booking
- Beautician
- Advance
- Full Payment
- System Administrator
- Job Opportunities
- Shopping Cart
- Career Form
- Manager
- Search Services
- Income
- Credit Card
- Debit Card
- Payments
- Reports
- Feedbacks
- New Services
- New Items

Identified Classes

- Products
- User
- Customer
- Services
- Report
- Beautician
- Payment

- Booking
- Feedback
- Shopping Cart

Rules of Rejecting Nouns

- Redundant – Visitor, User refers to the same person as “Customer”, Payment (Credit Card, Debit Card, Advance, Full Payment), Services (New Services), Products (New Items)
- Outside Scope of the System – System, System Administrator, Manager, Job Opportunities, Inquiry Form, Career Form
- Meta-Language – Visitor, User, Customer
- An Event or an Operation – Search Services
- Attributes – Email, Password, Income

Exercise 02

CRC Card

User	
Responsibility	**Collaborators**
Visit the Website	
Register to the System	
Search Products	Products
Search Services	Services
Add to Shopping Cart	Shopping Cart

Customer	
Responsibility	**Collaborators**
Search Services and View Services	Services
Search Products and View Products	Products
Purchase Products	Products
Do a Booking	Booking
Make the Payment	Payment
Add products to Shopping Cart	Shopping Cart
Give a Feedback	Feedback

Payment	
Responsibility	**Collaborators**
Store the Payment Details	Customer
Validate the accurateness of the Payment	Customer
Confirm the Payment	

Beautician	
Responsibility	**Collaborators**
Check for Bookings	Booking
Update the Availability of the Beautician	

Products	
Responsibility	**Collaborators**
Display Product Details	
Update the Availability of Product	

Services	
Responsibility	**Collaborators**
Display Service Details	
Redirects to Booking	Booking

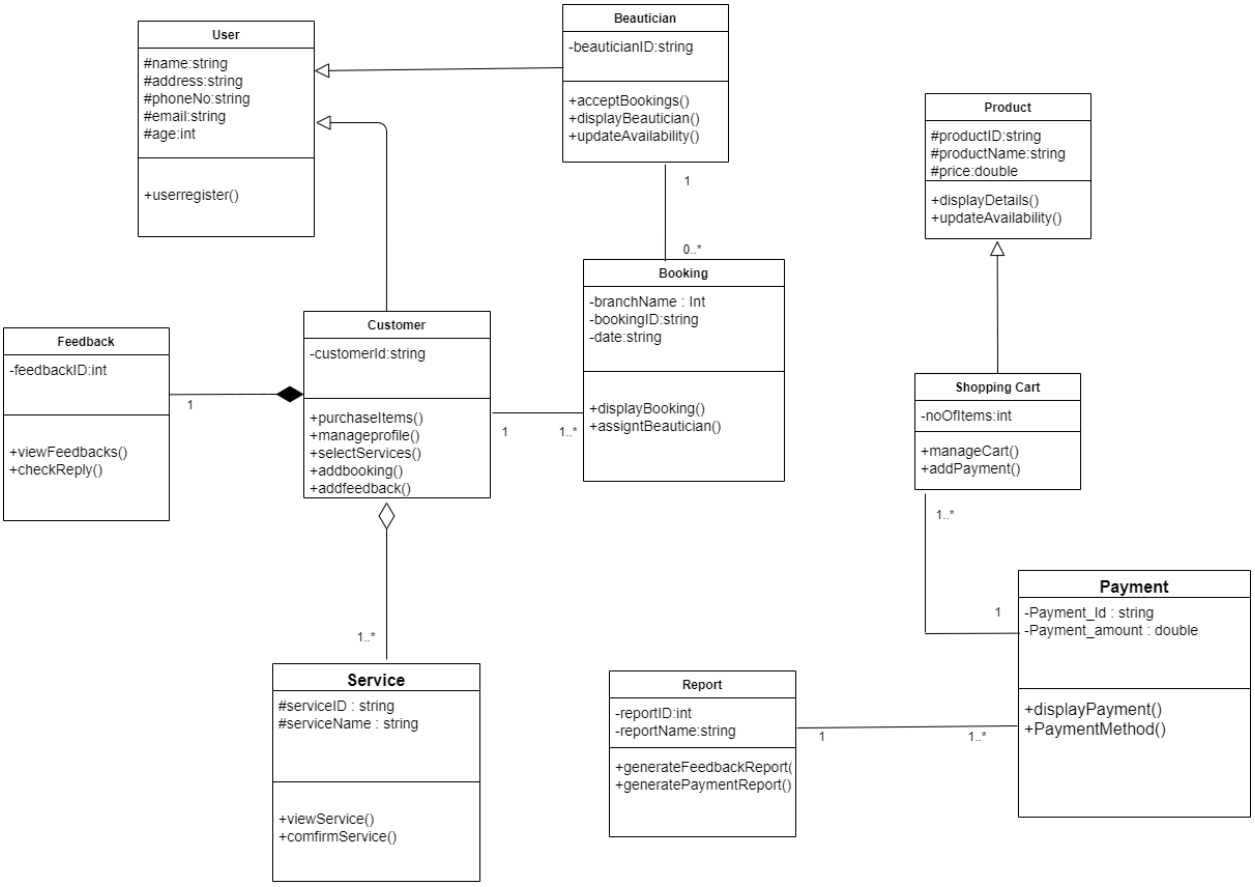
Booking	
Responsibility	**Collaborators**
Make a Booking	Customer
Assign a Beautician	Beautician
Confirm the Booking	Payment

Report	
Responsibility	**Collaborators**
Generate List of Available Products	Products
Generate List of Sold Products	Products
Generate Transaction Details	Payment

Feedback	
Responsibility	**Collaborators**
Manage feedbacks	Customer

Shopping Cart	
Responsibility	**Collaborators**
Manage shopping cart	

Exercise 03


User
#name:string
#address:string
#phoneNo:string
#email:string
#age:int
+userregister()
Beautician
-beauticianID:string
+acceptBookings()
+displayBeautician()
+updateAvailability()
Product
#productID:string
#productName:string
#price:double
+displayDetails()
+updateAvailability()
1
0..*
Booking
-branchName : Int
-bookingID:string
-date:string
+displayBooking()
+assigntBeautician()
Feedback
-feedbackID:int
+viewFeedbacks()
+checkReply()
1
Customer
-customerId:string
+purchaseItems()
+manageprofile()
+selectServices()
+addbooking()
+addfeedback()
1
1..*
Shopping Cart
-noOfItems:int
+manageCart()
+addPayment()
1..*
1
Payment
-Payment_Id : string
-Payment_amount : double
+displayPayment()
+PaymentMethod()
1..*
Service
#serviceID : string
#serviceName : string
+viewService()
+comfirmService()
Report
-reportID:int
-reportName:string
+generateFeedbackReport(
+generatePaymentReport()
1
1..*

Exercise 04

```
#include <iostream>
#include<cstring>
using namespace std;

//classes declaration
class Products;
class  User;
class Customer;
class Services;
class Report;
class Beautician;
class Payment;
class Booking;
class Feedback;
class Shopping_cart;

//base class(use inheritance)
class User {
    protected:
        string name;
        string  phoneNo;
        string email;
        int age;
        string address;

    public:
        User();//default constructor
        User(string pname, string pphoneNo, string pemail, int page, string
paddress);//overloaded constructor
        void userregister();
        ~User() {}//destructor called
};

User::User() {}

User::User(string pname, string pphoneNo, string pemail, int page, string paddress) {
        name = pname;
        phoneNo = pphoneNo;
        email = pemail;
        age = page;
        address = paddress;
}
void User::userregister() {}


class Report {
    private:
        int reportID;
        string reportName;
        Payment* payreport;

    public:
        Report() {}//default constructor
```

```
        Report(int RID, string rreportName) {
            reportID = RID;
            rreportName = reportName;
        }//overloaded constructor

        void generatepaymentReport(Payment* prep);
        ~Report() {}//destructor called

};

class Feedback {
    private:
        int feedbackID;

    public:
        Feedback();//default constructor
        Feedback(int pfeedbackID);//overloaded constructor
        void viewfeedback() {
            cout << "======================||Feedback||======================" << endl
<< endl;
            cout << "Excelent Service" << endl << endl;;
            cout << "++++++++++++++++++++++++++++++++++++++++++++++++++++++++" << endl;
        }
        void checkreply();
        ~Feedback() {}//destructor called

};

Feedback::Feedback(int pfeedbackID) {
    feedbackID = pfeedbackID;

}


class Services {
    protected:
        string serviceID;
        string serviceName;

    public:
        Services();//default constructor
        Services(string pserviceID, string serviceName);//overloaded constructor
        void viewService() {
            cout << "======================||Service||======================" << endl <<
endl;
            cout << "Service ID: " << serviceID << endl;
            cout << "Service Name: " << serviceName << endl;
        }
        void confirmService();
        ~Services() {}//destructor called

};

Services::Services() {}

Services::Services(string pserviceID, string pserviceName) {

        serviceID = pserviceID;
```

```
}


class Customer :public User {
    private:
        Booking* booking;
        Feedback* feedback[1];//composition relationship
        Services* cusservice[2];
        string CustomerId;

    public:
        Customer();//default constructor
        Customer(string pname, string pphoneNo, string pemail, int page, string paddress,
string pCustomerId, int serno, int feedno);//overloaded constructor
        void selectservice(Services* s1, Services* s2) {
            cusservice[0] = s1;
            cusservice[1] = s2;
        }
        void addbooking(Booking* cusb);
        void addfedback() {
            feedback[1]->viewfeedback();
        }
        void purchaseitems();
        void manageprofile();
        ~Customer() {}//destructor called
};

Customer::Customer() {}

Customer::Customer(string pname, string pphoneNo, string pemail, int page, string
paddress, string pCustomerId, int serno1, int feedno) :User(pname, pphoneNo, pemail,
page, paddress) {

        feedback[0] = new Feedback(feedno);
        CustomerId = pCustomerId;
 }


class Beautician : public User {
    private:
        string beauticianID;
        Booking* bbooking[2];

    public:
        Beautician();//default constructor
        Beautician(string pname, string pphoneNo, string pemail, int page, string
paddress, string pbeauticianID);//overloaded constructor
        void acceptbooking(Booking* beau);
        void updateAvailability();
        void displayBeautician() {
            cout << endl << "==================||Beautician Details||==============" <<
endl << endl;;
            cout << "Beautician Name: " << name << endl;
            cout << "Beautician ID: " << beauticianID << endl;
            cout << "Email: " << email << endl;
        }
        ~Beautician() {}//destructor called
```

```
};

Beautician::Beautician() {}

Beautician::Beautician(string pname, string pphoneNo, string pemail, int page, string
paddress, string pbeauticianID) :User(pname, pphoneNo, pemail, page, paddress)
{
    beauticianID = pbeauticianID;
}


class Booking {
    private:
        Customer* cus;
        Beautician* beautician;
        string bookingID;
        string branchName;
        string date;

    public:
        Booking();//default constructor
        Booking(string pbookingID, string pbranchName, string pdate, Customer* pcustomer,
Beautician* pbeautician);//overloaded constructor
        void assignBeautician(Beautician* B);

        void displayBooking() {
            cout << "++++++++++++++++++++++++++++++++++++++++++++++++++++++++" << endl;
            cout << endl << "======================||Booking||======================" <<
endl << endl;
            cout << "Booking ID: " << bookingID << endl;
            cout << "Branch Name: " << branchName << endl;
            cout << "Appointment Date: " << date << endl << endl;
        }
        ~Booking() {}//destructor called

};

class Customer;
Booking::Booking() {}

Booking::Booking(string pbookingID, string pbranchName, string pdate, Customer*
pcustomer, Beautician* pbeautician) {

        bookingID = pbookingID;
        branchName = pbranchName;
        date = pdate;
        cus = pcustomer;
        beautician = pbeautician;
}

class Shopping_cart;

class Products {
    private:
        string productID;
        string productName;
        float price;
```

```
    public:
        Products() {}//default constructor
        Products(string pproductID, string pproductName, float pprice);//overloaded
constructor
        void displayproduct();
        ~Products() {}//destructor called

};

Products::Products(string pproductID, string pproductName, float pprice)
{
        productID = pproductID;
        productName = pproductName;
        price = pprice;
}


class Shopping_cart :public Products {
    private:
        Payment* paycart;
        int noOfitems;

    public:
        Shopping_cart();//default constructor
        Shopping_cart(string PProductId, string PProductname, float Price, int
pnoOfitems);//overloaded constructor
        void managecart();
        void addpayment(Payment* p);
        ~Shopping_cart() {}

};

Shopping_cart::Shopping_cart(string pproductID, string pproductName, float pprice, int
pnoOfitems) :Products(pproductID, pproductName, pprice) {

        noOfitems = pnoOfitems;
}


class Payment {
    private:
        Shopping_cart* pcart;
        Report* payreport;
        string  Payment_Id;
        double Payment_amount;
    public:
        Payment();//default constructor
        Payment(string pPayment_Id, float pPayment_amount, Shopping_cart* cart, Report*
p);//overloaded constructor
        void displaypayment() {
            cout << endl << "======================||Payment||======================" <<
endl << endl;
            cout << "Payment ID: " << Payment_Id << endl;
            cout << "Payable Amount: " << Payment_amount << endl << endl;
        }
        ~Payment() {}//destructor called
```

```
};

Payment::Payment(string pPayment_Id, float pPayment_amount, Shopping_cart* cart, Report*
p) {

        Payment_Id = pPayment_Id;
        Payment_amount = pPayment_amount;
        pcart = cart;
        payreport = p;

}


int main() {

        Customer* C1 = new Customer("Nipuni Sooriarachchi", "0768959018",
"nipuni.s@gmail.com", 30, "89/1,Bataduwa,Galle", "CUS01", 01, 23);
        Beautician* BT = new Beautician("Sandali Fernando", "0773456738",
"sandali.salonj@gmail.com", 33, "45/2,Arangala,Malabe", "BEAU01");
        Booking* B1 = new Booking("BK01", "Nugegoda", "2021/06/04", C1, BT);
        Shopping_cart* SC = new Shopping_cart("PRO01", "Oriflame", 3700, 1);
        Report* Re = new Report(1001, "DAY01");
        Payment* P = new Payment("PAY01", 1000, SC, Re);
        Products* Pr = new Products("PRO02", "Oriflame", 1000);
        Services* Ser = new Services("SER01", "Hair Cutting");
        Feedback* feed = new Feedback(01);
        User* user = new User("Senalii Mendis", "0786387632", "senali.m@gmail.com", 44,
"123/3,Mavittara,Kottawa");

        B1->displayBooking();//Booking::displayBooking() called
        Ser->viewService();//Services::viewService() called
        BT->displayBeautician();//Beautician::displayBeautician() called
        P->displaypayment();//Payment::displaypayment() called
        feed->viewfeedback();//Feedback::viewfeedback() called

        //deallocate memory(release memory)
        delete C1;
        delete P;
        delete Ser;
        delete BT;
        delete B1;
        delete feed;
        delete Re;
        delete SC;
        delete Pr;
        delete user;

    return 0;

}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

======================||Booking||=======================

Booking ID: BK01
Branch Name: Nugegoda
Appointment Date: 2021/06/04

======================||Service||=======================

Service ID: SER01
Service Name:

====================||Beautician Details||===============

Beautician Name: Sandali Fernando
Beautician ID: BEAU01
Email: sandali.salonj@gmail.com

======================||Payment||=======================

Payment ID: PAY01
Payable Amount: 1000

======================||Feedback||=======================

Excelent Service

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```