BSc (Hons) in Information Technology

Object Oriented Concepts – 1050

Assignment 2

Year 1, Semester 2 **2023 - February**

Topic : Boat Safari Trip Management System

Group no : MLB_16.01_07

Campus : Malabe

Submission Date: 14/ June / 2023

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
IT22560926	PRABASHWARA R. P	0713529325
IT22601674	DILSHAN K. B	0774021236
IT22603104	DIAS M. P. U	0717031982
IT22602800	MUTHUKUDA ARACHCHIGE N. D	0703141890
IT22560094	RANASINGHE T. M. R	0781904889

Contents

1. Description

GreenHorizon operates a global boat safari company. Due to its high level of customer satisfaction, this service has numerous users worldwide. Additionally, Greenhorizon offers affordable packages and offers superior customer service to other boat services. They conduct specific activities including wedding photo shoots, photography, vlogging, and hotel arrangements.

Any user intends to make a reservation on the GreenHorizon website must first register as a customer. Both registered customer and unregistered customer can view packages and see feedbacks by online. After registration, only registered customer can add feedback and comments to the website. When customer make reservation, customer must submit both the customer's information and the details of package. After entering the details of the customer and packages, one can make payment using either a Visa or Mastercard or at the reception counter. Afterwards, the customer's selected payment information and account information are stored in the reservation report. The reservation report includes a payment invoice and a report on the customers.

Requirements

1. Feedback is part of the customer class, and feedback cannot exist without a customer. One customer can provide numerous feedback.
2. Customers have the capability to select and participate in multiple activities.
3. Payment class become a part of the Customer class, also payment class cannot exist without a customer class. One customer can make only one payment.
4. Package class become a part of the Payment class, also payment class cannot exist without a Package class. One package belongs to only one payment class.
5. Each package is associated with a single report.
6. Package class become a part of reservation class, also Package class can exist without a reservation class. One package belongs to one reservation.
7. The Family package and Premium package are categorized as subclasses of the Package class.
8. Each package can have one or multiple feedback.
9. Each payment has a corresponding one report.
10. Each customer is associated with a single reservation.
11. There is one report for each reservation.

2. Classes Identified

- Activity
- Customer
- Payment
- Reservation
- Feedback
- Package
- Report

3. CRC card

Activity	
Resposiblility	**Collaboration**
provide activity details	

Customer	
Resposiblility	**Collaboration**
Register to the system Subbmit Information	

Payment	
Resposiblility	**Collaboration**
Make payment	
Store package details	Report
Display Package details	Report
Generate payment reciepts	
Show availability	
Managing payment methods	

Reservation	
Resposiblility	**Collaboration**
Make reservation	Customer
Send reservation Details to report	Report

Package	
Resposiblility	**Collaboration**
Select package	
Store package details	Report

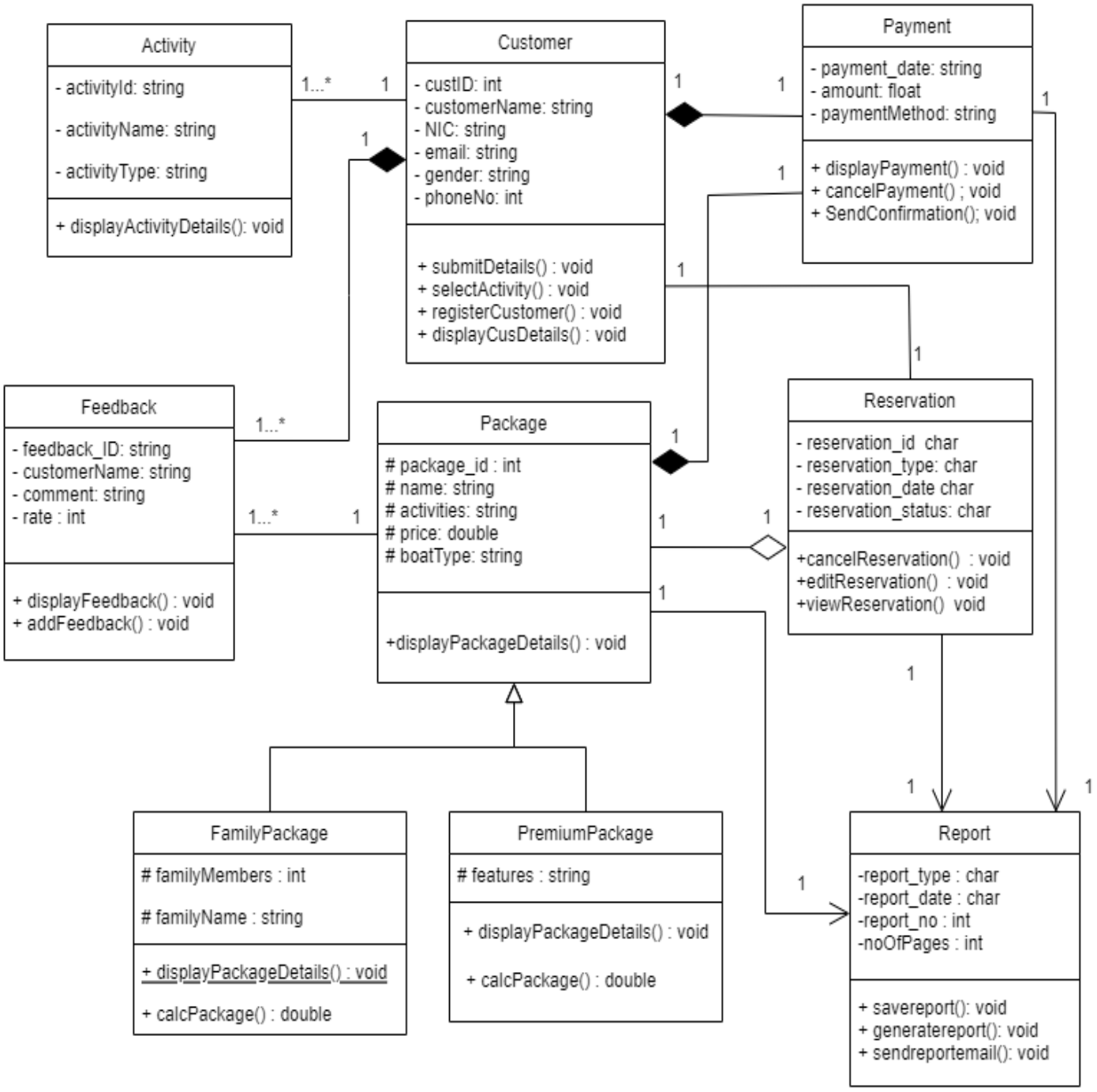
Feedback	
Resposiblility	**Collaboration**
Add feedback to the website	Customer
Display feedback to the customer	

Family Package	
Resposiblility	**Collaboration**
Select package	
Store package details	
Offering discounts.	

Report	
Resposiblility	**Collaboration**
Store reservation details	Reservation
Store payment details	Payment
Store customer details	Customer

Premium Package	
Resposiblility	**Collaboration**
Select package	
Store package details	
Providing additional features	

4. Class diagram



5. Coding for the classes

```
#include<iostream>
#include<cstring>
#include<string>
#define SIZE 15
using namespace std;

//DIAS M.P.U - IT22603104
class Report
{
private:
	int report_no;
	int noOfPages;
	char report_type[25];
	char report_date[35];

public:
Report()
{
	report_no = 0;
	noOfPages = 0;
	strcpy_s(report_type,"");
	strcpy_s(report_date,"");
}
Report(const char preport_type[], const char preport_date[], int preport_no, int
pnoOfPages)
{
	strcpy_s(report_type, 25, preport_type);
	strcpy_s(report_date, 35, preport_date);
	report_no = preport_no;
	noOfPages = pnoOfPages;
}
```

```
void generatereport();
void savereport();
void sendreportemail();
~Report();
};

class Reservation
{
private:
	package* package1;
	Customer* customer1;
	Report* report1;
	char reservation_id[10];
	char reservation_type[15];
	char reservation_date[10];
	char reservation_status[10];
public:
Reservation()
{
	strcpy_s(reservation_id,"");
	strcpy_s(reservation_type, "");
	strcpy_s(reservation_date, "");
	strcpy_s(reservation_status, "");
}
Reservation(const char preservation_id[], const char preservation_type[], const char
preservation_date[],const char preservation_status[])
{
	strcpy_s(reservation_type,preservation_type);
	strcpy_s(reservation_id,preservation_id);
	strcpy_s(reservation_date,preservation_date);
	strcpy_s(reservation_status,preservation_status);
}
void reservationDetails();
void cancelReservation(package* package1);
void editReservation(package* package1);
~Reservation();
```

```
};
//DILSHAN K.B - IT22601674
class Customer {
private:
//static const int SIZE = 10; // Assuming SIZE is a constant
	int custID;
	string customerName;
	string NIC;
	string email;
	string gender;
	int phoneNo;
//Bi-directional Association relationship with activity and customer
Activity* activity1[SIZE];
//Compostion relationship with customer feedback
Feedback* feedback1[SIZE];
//Composition relationship with payment
Payment* payment1;

public:
Customer()
{
	custID = 0;
	customerName = "";
	NIC = "";
	email = "";
	gender = "";
	phoneNo = 0;
}
Customer(int CID, string Cname, string nic, string Email, string Gender, int PNo)
{
	custID = CID;
	customerName = Cname;
	NIC = nic;
	email = Email;
	gender = Gender;
	phoneNo = PNo;
};
```

```
void submitDetails();
void selectActivity(); //void selectActivity(Activity* A);
void registerCustomer();
void displayCusDetails();
~Customer();
};

class Activity {
private:
	string activityId;
	string activityName;
	string activityType;
//Bi-directional Association relationship with activity and customer
Customer* cus;

public:
Activity()
{
	activityId = "";
	activityName = "";
	activityType = "";
}
Activity(string ActId, string ActName, string ActType, Customer* cus)
{
	activityId = ActId;
	activityName = ActName;
	activityType = ActType;
};
void setActivityDetail();
void displayActivityDetails();
~Activity();
};
```

```
//MUTHUKUDA ARACHCHIGE N.D - IT22602800
class Feedback
{
private:
//Bi - directional Association relationship with package
	package* package1;
	string feedback_ID;
	string customerName;
	string comment;
	int rate;

public:
Feedback()
{
	feedback_ID = "";
	customerName = "";
	comment = "";
	rate = 0;
}
Feedback(string pfeedback_ID, string pcustomerName, string pcomment, int prate)
{
	feedback_ID = pfeedback_ID;
	customerName = pcustomerName;
	comment = pcomment;
	rate = prate;
}
void addFeedback(string fed_ID, string f_uname, string f_desc, int f_rate);
void displayFeedback();
~Feedback();

};
```

```
//RANASINGHE T.M.R - IT22560094
class Payment
{
private:
	string payment_date;
	float amount;
	string payment_method;
//an object of an report class -- Uni directional association relationship
Report* report1;

public:
Payment()
{
	payment_date  = "";
	amount = 0.00;
	payment_method = "";
}
Payment(string P_date, float P_amount, string P_method)
{
	payment_date = P_date;
	amount = P_amount;
	payment_method = P_method;

}
void displayPayment();
void cancelPayment();
void SendConfirmation();
~Payment();
};
```

```
//PRABASHWARA R.P -IT22560926
class package
{
protected:

	int package_id;
	string name;
	string activities;
	double price;
	string boatType;

// Bi- directional Relationship between Feedback and package
Feedback* feedback1[SIZE];

//Composition relationship between payment and package (1:1)
Payment* payment1;

//Aggregation relationship between Reservation and package
Reservation* reservation1;

//Uni-directional association between Report And package
Report* report1;

public:
package()
{
	package_id = 0;
	name = "";
	activities = "";
	price = 0.0;
	boatType = "";
}
```

```
package(int pId, string pName, string act, double prz, string bType) {
	package_id = pId;
	name = pName;
	activities = act;
	price = prz;
	boatType = bType;
}
void displayPackageDetails();

~package()
{
//calling destructor for Composition relationship  (1:1)
	delete payment1;
}
};

class FamilyPackage: public package {
protected:
	int familyMembers;
	string familyName;
public:
FamilyPackage()
{
	familyMembers = 0;
	familyName = " ";
}
FamilyPackage(int pId, string pName, string act, double prz, string bType,
int fMembers, string fName)
:package(pId, pName, act, prz, bType) {

	familyMembers = fMembers;
	familyName = fName;
}
```

```
void displayPackageDetails();
double calcPackage();
~FamilyPackage();

};

class premiumPackage: public package {
protected:
	string features;
public:
premiumPackage()
{
	features = "";
}
premiumPackage(int pId, string pName, string act, double prz, string bType,
string ftrs)
: package(pId, pName, act, prz, bType) {

	features = ftrs;

}
void displayPackageDetails();
double calcPackage();
~premiumPackage();
};
```

```
int main()
{
Report* report2;
report2 = new Report();

Reservation *reservation2;
reservation2 = new Reservation();

Customer* customer2;
customer2 = new Customer();

Activity* activity2;
activity2 = new Activity();

Feedback* feedback2;
feedback2 = new Feedback();

Payment* payment2;
payment2 = new Payment();

FamilyPackage* familypackage2;
familypackage2 = new FamilyPackage();

premiumPackage* premiumpackage2;
premiumpackage2 = new premiumPackage();



return 0;
}
```

6. Individual contributions

IT22560926 - PRABASHWARA R.P

As a leader, I was gathering team members to discuss our scenario and select classes using noun-verb analysis. I was creating the CRC card structure and instructing them to fix errors and rebuild it. After the CRC stage, the workload is divided into several parts and assigned to team members. I created the package class and the respective Family package and premium package, which are inheritance relationships. and decided which relationships interact with my own class and implemented them. I also implemented objects for Family package and premium package sub classes

IT22601674 - DILSHAN K.B

I carried out noun-verb analyses individually for the scenario and found class (other members also did it individually). Then I discuss with my group members, add rejection rules, and find the essential class for the diagram. I contributed to the design and development of the coding for the activity and customer classes. Further, I implemented an object for each class.

IT22603104 - DIAS M.P.U

I have contributed to develop the classes and objects for Report and Reservation and implementing CRC Cards. Throughout my contributions, I have collaborated with my other team members to identify the suitable classes through Noun-verb analysis.

IT22602800 - MUTHUKUDA ARACHCHIGE N.D

As a team member I assisted in identifying the nouns in the scenario and proceeded to create CRC cards specifically for the feedback class. Furthermore, the code for the feedback class was implemented.

IT22560094 - RANASINGHE T. M. R

During our group project, my main contribution was in assisting team members with identifying the nouns and verbs in our scenario. Additionally, I took the initiative to draw CRC cards to represent the responsibilities and collaborators of the Payment, Customer, and Report classes.

Furthermore, creating a class diagram as a teamwork and help team members to identify multiplicity of these relationships. Finally, I actively participated in coding by implementing the Payment class with constructors and destructors, and successfully instantiated objects for testing and integration.