Topic : Boat Safari management System

Group no : MLB_02.02_11

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**
IT21211928	Gamlath W.A.V.K	0766883316
IT21214684	Kumarasiri O.A.K.U	0711245045
IT21208980	De Silva K.H.P.N	0763807475
IT21215056	Wickramasinghe W.A.D.L	0765706325
IT21213908	Samaranayake S.G.H.V	0702160983

System Requirements

- The guest user should be able to register to the system and create an account.
- The registered user and system admin should be able to login to account using credentials.
- The registered user and system admin should be able to reset password if they forgot.
- The system admin should be able to view total number of bookings they received.
- The registered user should be able to see the list of packages
- The registered user should be able to see the list of offers.
- As a registered customer can reserve package and place with prefer date and time
- After reservation can make payment with credit or debit card through payment portal
- The registered user should be able to give feedbacks
- The system admin must be able to provide solutions for their issues as soon as possible.
- The system admin should be able to add, remove and modify the system.
- manager can generate financial reports and send to admin
- System admin can view report and make necessary decisions.

Identified Classes

- guest user
- account
- registered user
- bookings
- packages
- offers
- payment
- feedbacks
- solutions
- reports

CRC Card

Class Name -: Account class	
Responsibilities	**Collaborations**
Store Guest user details	Guest user
Validate Registered user credentials	Registered user
Password reset	Registered user

Class Name -: Offers class	
Responsibilities	**Collaborations**

Class Name -: Report class	
Responsibilities	**Collaborations**
List of offers	Offers
List of packages	Packages

Class Name: Registered User	
Responsibilities	**Collaboration**
Register details Edit account details Check packages	Account Packages

Class Name: Booking	
Responsibilities	**Collaboration**
Making a booking Check for offers Confirm the booking	Registered User Offers Payment

Payment	
Responsibilities	**Collaborations**
View payment details	Registered user
Validate the payment	
Save payment details	

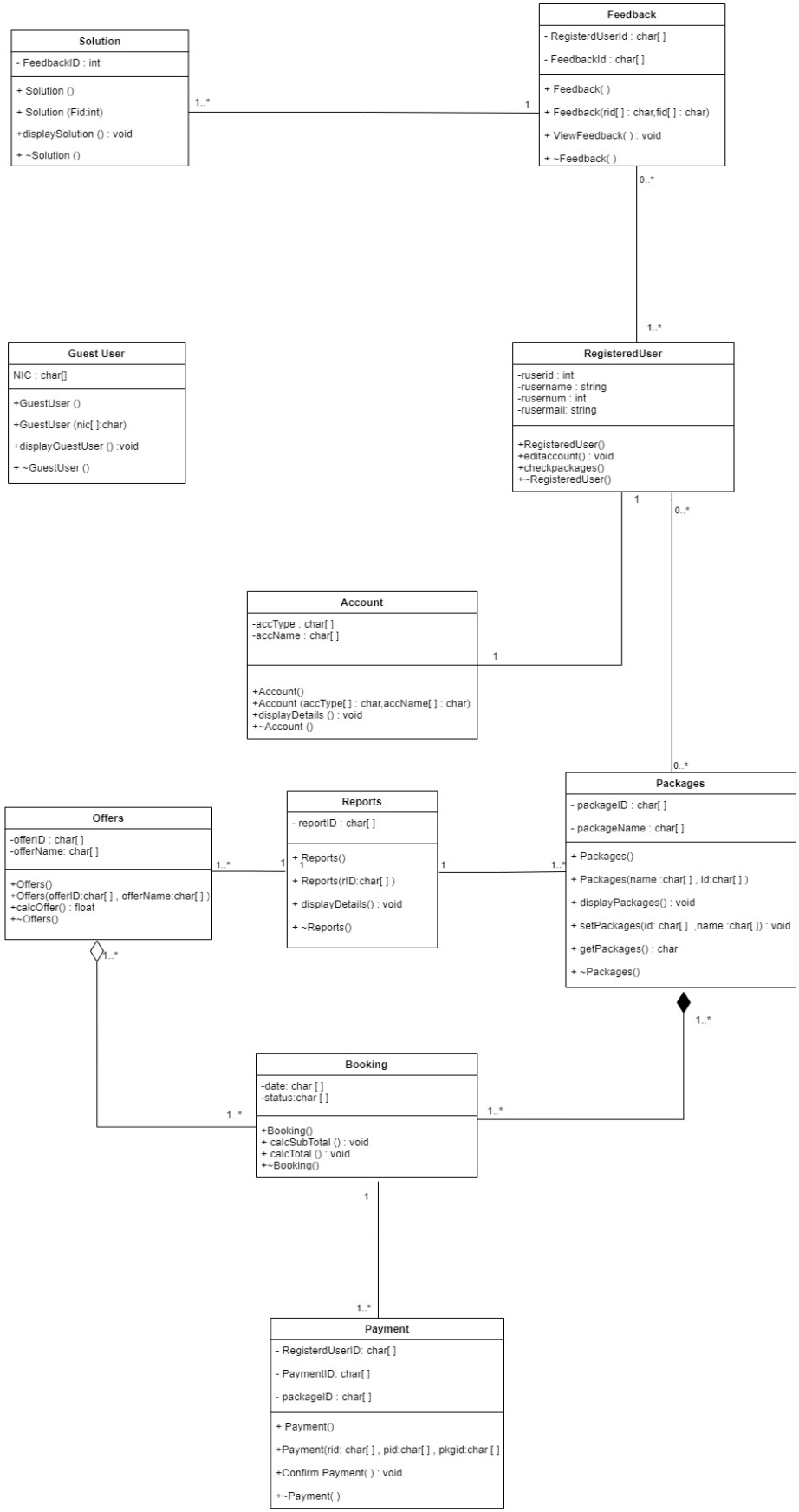
Feedback	
Responsibilities	**Collaborations**
View feedback	Registered user
Save feedback	

Class Name: Guest User	
Responsibilities	**Collaboration**
Checking details Registration to the system	Account

Class Name: Solution	
Responsibilities	**Collaboration**
Focusing on a problem Identify the problem Problem solving	

Class Name: Packages	
Responsibilities	**Collaboration**
Add Packages Update Packages Package Details	

Class Diagram (UML Notation)



Codes

- Relationship with Solution and feedback

```
#include<iostream>
#include<cstring>
#define size 20
using namespace std;

class Solution
{
	protected://private
		char SolutionID[20];
		Feedback * Feedback1;

	public://public
	      Solution();
		void setSolution(char Sid[20]);
		void displaySolution();
		~Solution();

};

class Feedback{
	protected://private
		char RegisteredUserID[20];
		char FeedbackID[20];
		Solution * Sol[size];
	public://public
	      Feedback();
		void assigndetails(char rid[20],char fid[20]);
		void displayFeedback();
		~Feedback();

};
//default constructor
Feedback :: Feedback(){
	strcpy(RegisteredUserID, " " );
	strcpy(FeedbackID, " ");
}
/*constructor with parameter
void Feedback :: assignDetails(char pRid,char  pFid){


}

void Feedback :: displayFeedback(){

}
//destructor
Feedback :: ~Feedback(){


}*/
/class Solution
{
	protected://private
		char SolutionID[20];
		Feedback * Feedback[20];
```

```
        public://public
              Solution();
                 void setSolution(char Sid[20]);
                 void displaySolution();
                 ~Solution();

};
//default constructor
Solution :: Solution(){

        strcpy(SolutionID," ");

}
/*constructor with parameter
void Solution :: setSolution(){
}

}

void Solution :: displaySolution(){


}
//destructor
Solution :: ~Solution(){


}*/

int main()
{
        Solution *sol1;
        sol1 = new Solution();

        Feedback *feed1;
        feed1 = new Feedback();

        delete sol1;
        delete  feed1;

        return 0;


}
```

- Relationship with registered user, feedback, account and package

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

Class RegisteredUser{
	private:
		int reguserid;
		string regusername;
		int regusernum;
		string regusermail;
		Feedback*fback[size];
		Package * packs[size];
		Account*acc[size];

	public:
		RegisteredUser();
		RegisteredUser(string rid, string rname, int rnum, string rmail);
		Void editaccount();
		Void checkpackages();
		~RegisteredUser();
};
RegisteredUser::RegisteredUser (string id, string name, int num, string mail)
{
	reguserid = id;
	regusername = name;
	regusermail = email;
	regusernum = num;
}
void Customer::Customer(){
	cout<<"RegisteresUser id=" <<reguserid <<endl;
	cout<<"RegisteredUser name=" <<regusername <<endl;
	cout<<"RegisteredUser email=" <<regusermail <<endl;
	cout<<"RegisteredUser telephone number=" <<regusernum <<endl;
}
void RegisteredUser::editaccount() {}
void RegisteredUser::checkpackages() {}
int main()
{
	RegisteredUser regu1;
	Return 0;
}



class Feedback {
	private:
		int feedbackid;
		string rusername;
		Registereduser *ruser;
		Solution* sltion[size];
	public:
		Feedback();
		Feedback(int rid , string rname);
		void ViewFeedback();
};
Feedback::Feedback() {}
Feedback::Feedback(int rid, string rname)
{
feedbackid = pid;
```

```
strcpy(rusername, rname);
}
void Feedback::viewfeedback() {}

int main()
{
	Feedback fdk1;
	Return 0;
}



class Package {
	private:
		char packageID;
		char packageName;
		Booking *bkng[size];
		Registereduser *ruser;
		Reports* rprt[size];

	public:
		Packages();
		Packages(char name[] , char id[]);
		void displayPackages();
		void setPackages(char id[] , char name[]);
		char getPackages();
		~Packages();
};


Packages::Packages(){}
Packages::Packages(char name[] , char id[])
{
	packageid = id;
	packagename = name;
}

void Package::displayPackages(){}
void Package::viewPackages(){}

int main{
Package pkg1;
}


Class Account{
	private:
		char accType;
		char accName;
		Registereduser *ruser;
	public:
		Account()
		Account(char atype[], char aname[]);
		Void displayDetails();
		~Account;
};
Account::Account()
{
	strcpy(accType,"");
	strcpy(accName,"");
}
Account::Account (char atype)
```

```
{
	strcpy(accType,atype);
}
Account::Account (char aname)
{
	strcpy(accName,aname);
}


void Account::displayDetails(){}
~Account::Account(){}

int main()
{
	Account acc1;
}
```

- Relationship with packages, registered user, reports and booking

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

class Booking{
	private:
		int bookingID();
		char date[];
		string status;

	public:
		Booking();
		void calcSubTotal();
		void calcTotal();
		~Booking();

};

Booking::Booking()
{
	strcpy(date,"");
	strcpy(status,"");
}

void Booking::calcSubTotal()
{

};

void Booking::calcTotal()
{

}

Booking::~Booking
{
	cout<<"Delete Booking"<<date<<status<<endl;
}

class Packages{
	private:
		char packageID[10];
		char packageName[20];
		Booking *Book[SIZE];

	public:
		Packages();
		Packages(char name[] , char id[]);
		void displayPackages();
		void setPackages(char id[] , char name[]);
		char getPackages();
		~Packages();
};
Packages::Packages()
{
	Book[0] = new Booking(00);
	Book[1] = new Booking(11);
}

Packages::Packages(int no1, int no2)
```

```
{
	Book[0] = new Booking[no1];
	Book[1] = new Booking[no2];
}
void displayBooking(){

}

void Packages::setPackages(char id[], char name[])
{

}

char Packages::getPackages()
{

}

Packages::~Packages()
{
	//Destructor
}

int main()
{
	Packages *myPackages;
	myPackages = new Packages(20,30);

	return 0;
}

class Reports{
	private:
		char reportID[];
		Packages *pack[2];

	public:
		Reports();
		Reports(char rID[]);
		void displayDetails();
		Reports *rep[2];
		~Reports();

};
Reports::Reports()
{
	strcpy(reportID,"");
}

Reports::Reports(char rID)
{
	strcpy(reportID,rID);
}

void Reports::displayDetails()
{

}

~Reports::Reports()
{

}
```

```
int main()
{
	Reports rep;
}

class RegisteredUser{
	private:
		int ruserID;
		string rusername;
		int rusernum;
		string rusermail;
		Packages *pkgs[2];

	public:
		RegisteredUser();
		void edita();
		void checkpackages();
		~RegisteredUser();

	RegisteredUser::RegisteredUser()
	{
		ruserID = 0;
		rusername = "AAAA";
		rusernum =  0;
		rusermail = "0";
	}

	void RegisteredUSer :: edita()
	{

		}

	void RegisteredUser ::checkpackages()
	{

	}

	RegisteredUser::~RegisteredUser()
	{
		//Destructor
	}

	int main()
	{
		RegisteredUser reg;
	}

};
```

- Relationship with payment and booking

```
#include <iostream>
#include <cstring>

using namespace std;

class Payment
{
 private :
             char registerdUserId[20];
             char paymentId[10];
             char packageId[10];


             Booking * booking;

 public :
             Payment();
             Payment(char rregisterdUserId[], char ppaymentId[], char
ppackageId[],Booking * bbooking);
             void confirmPayment(Booking * bbookig);
             ~Payment();

};

class Booking
{
      private :
                   char date[5];
                   char status[10];


                   Payment * pay[SIZE];



      public :
                   Booking();
                   Booking(char ddate[], char sstatus[]);
                   //void clacSubTotal();
                   void calcTotal (Payment * ppay);
                   ~Booking();
}
Payment::Payment() {

             strcpy(registerdUserId,"");
             strcpy(paymentId,"");
             strcpy(packageId,"");

}

Payment::Payment(char rregisterdUserId[], char ppaymentId[], char
ppackageId[],Booking * bbooking) {

             strcpy(registerdUserId,rregisterdUserId);
             strcpy(paymentId,ppaymentId);
             strcpy(packageId,ppackageId);

             Booking = bbooking;
```

```
}

void Payment::confirmPayment(Booking * bbooking) {

}

Payment::~Payment() {

}

Booking::Booking() {

              strcpy(date,"");
              strcpy(status,"");


}

Booking::Booking(char ddate[], char sstatus,Payment * ppay) {

              strcpy(date,ddate);
              strcpy(status,sstatus);

              Payment = ppay

}
void Booking:calcSubTotal()
{

}
void Booking::calcTotal()
{

}

int main ()
{
       Paymnent p1;

       Booking b1;
}
```

- Relationship with offers and booking

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

//part
class Booking{
	private:
		char date[10];
		char status[10];

	public:
		Booking(const char pDate[],const char pStatus[]){
			strcpy(date,pDate);
			strcpy(status,pStatus);
		};
		void calcSubTotal(){

		};
		void calcTotal(){

		};
		~Booking(){
			cout<<" Booking Destructor called!!"<<endl;
		};

};

//whole
class Offers{
	private:
		char offerID[20];
		char offerName[20];
		Booking *book[2];

	public:
		Offers(){
			book[0] = new Booking("2022.05.18","pending");
			book[1] = new Booking("2022.05.19","pending");
		};
		Offers(const char pOfferID[],const char pOfferName[]){
			strcpy(offerID,pOfferID);
			strcpy(offerName,pOfferName);
		};
		float calcOffer(){

		};
		~Offers(){
			cout<<" Offer Destructor called!!"<<endl;
		};
};

int main(void)
{
	Offers *cust1 = new Offers();

	delete cust1;

	Booking *book1 = new Booking("2022.05.18","pending");
	Booking *book2 = new Booking("2022.05.19","pending");
```

```
        delete book1;
        delete book2;

        return 0;
}
```

- Relationship with offers and reports

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

class Reports;
class Offers;
//offers calss
class Offers{
	private:
		char offerID[10];
		char offerName[20];
		Reports *report[3];

	public:
		Offers(){
			//not used. because i sent values from the objects.
		};
		Offers(const char pOfferID[],const char pOfferName[]){
			strcpy(offerID,pOfferID);
			strcpy(offerName,pOfferName);
		};

		float calcOffer(){

		};

		~Offers(){
			cout<<" Offers Destructor called!!"<<endl;
		};
};

class Reports{
	private:
		char reportID[10];
		Offers *offer;

	public:
		Reports(){
			//not used. because i sent values from the objects.
		};

		Reports(char rID[], Offers *pOffer){
			strcpy(reportID,rID);
			offer = pOffer;

		};

		void displayDetails(){

		};

		~Reports(){
			cout<<" Reports Destructor called!!"<<endl;
		};

};
int main (void)
{
```

```
        Offers *offer1 = new Offers("001","package 1");
        Offers *offer2 = new Offers("002","package 2");

        Reports *report1 = new Reports("010",offer1);
        Reports *report2 = new Reports("011",offer2);

        delete offer1;
        delete offer2;

        delete report1;
        delete report2;

        return 0;


}
```

- Guest user

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

class GuestUser{
	protected ://private
		char NIC[10];

	public ://public
		GuestUser ();
		  GuestUser (char nic[10]);
		  void displayGuestUser();
		  ~GuestUser();

};
//default constructor
GuestUser :: GuestUser(){

	strcpy(NIC,"");
}
//constructor with parameter
GuestUser :: GuestUser(char pnic[10]){

	strcpy(NIC, pnic);
}
void GuestUser :: displayGuestUser(){

}
//destructor
GuestUser :: ~GuestUser(){

}

int main()
{
	GuestUser  g;
}
```