Sri Lanka Institute of Information Technology

Topic : Wild-Life Safari Management System

Group no : MLB_02.01_05

Campus : Malabe

Submission Date : 2022.05.16

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**
IT21206900	Peiris P.A.S	0711034797
IT21211164	Gayasri B.H.D.B	0771138876
IT21211478	Bhagya E.M.P	0775846166

System Requirements

1. A user needs to register to the system providing details such as full name, address, phone number.
2. A user can visit the website and explore tours.
3. A user should login to the website providing login details.
4. A registered customer can select a package for a preferred tour.
5. One package has only one tour.
6. A customer can make a booking for the selected package.
7. A booking must require a payment.
8. A customer can make payments online through the website using credit cards, debit cards and online money transfer.
9. Once the customer confirms the booking and the payment is validated, the booking is made.
10. A customer can send feedback about the service.
11. Administrator must login to the system to do tasks.
12. Administrator can update tour details and package details.
13. Administrator should generate a list of new bookings.
14. Administrator can respond to customer feedback.
15. Feedbacks can be viewed by administrator and customers.

Identified classes using Noun Verb Analysis

- User
- Registered Customer
- Tour
- Package
- Booking
- Payment
- Administrator
- Feedback
- Report

CRC Cards

User Class	
Responsibilities	**Collaborations**
Store user details	
Display user details	
Explore tours	Tour

Customer Class	
Responsibilities	**Collaborations**
Provide login details	
Edit profile details	
Give feedback	Feedback

Tour Class	
Responsibilities	**Collaborations**
Store tour details	
Display tours	

Package Class	
Responsibilities	**Collaborations**
Store package details	
Display package details	

Booking Class	
Responsibilities	**Collaborations**
Store booking details	
Make booking	Customer
Confirm booking	Payment
Get booking details	Customer

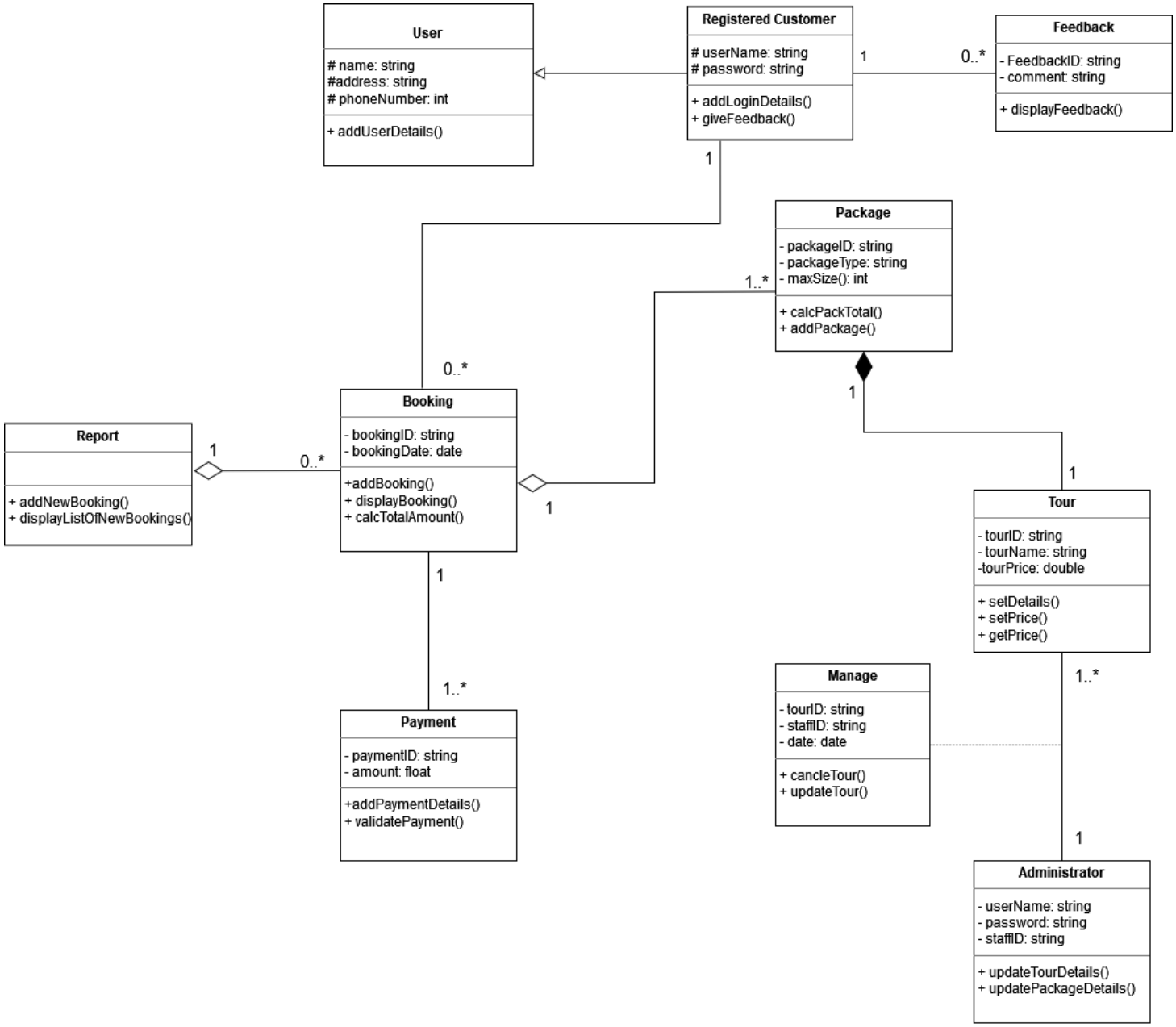
Payment Class	
Responsibilities	**Collaborations**
Store payment details	
Validate	
Make a payment	Customer

Administrator	
Responsibilities	**Collaborations**
Provide login details	
Update packages	package
Update tour details	tour
respond to customer feedback	feedback

Feedback Class	
Responsibilities	**Collaborations**
Add feedback	customer
Display feedback	

Report Class	
Responsibilities	**Collaborations**
List of new bookings	booking

Class Diagram



Coding for the classes in class diagram

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

//Payment class
class Payment
{
   private:
      String paymentID;
      float amount;

   public:
       Payment();
       void addPaymentDetails();
      void validatePayment();
      ~Payment();
};


Payment::Payment(string payID, float amnt)
{
          paymentID = payID;
          amount = amnt;
}

void Payment::addPaymentDetails(){

}

void Payment::validatePayment(){

}

Payment::~Payment(){

}


//Booking class
class Booking
{
    private:
```

```
        string bookingID;
        string bookingDate;
        Payment *pay;

    public:
        Booking(string BID, date bDate, Payment *p);
        void addBooking();
        void displayBooking();
        float calcTotalAmount();
        ~Booking();
};

Booking::Booking(string BID, string bDate, Payment *p){
            bookingID = BID);
            bookingDate = bDate);
            pay = p;
}

void Booking::addBooking(){

}

void Booking::displayBooking(){

}

float Booking::calcTotalAmount(){

}

Booking::~Booking(){
            cout << "Deleting Booking" << bookingID << endl;
}



//Report class
class Report
{
    private:
        Booking * book[2];

    public:
            Report();
```

```
        void addNewBooking(Booking *b1, Booking *b2);
    void displayListOfNewBookings();
    ~Report();
};

Report::Report(){

}

void Report::addNewBooking(Booking *b1, Booking *b2){
        book[1] = b1;
        book[2] = b2;
}

string Report::displayListOfNewBookings(){
        for(int i = 0; i < 2; i++)
                book[i]->displayBooking();
}

Report::~Report(){

}
```

```
//User Class
class user
{
      protected:
            string name;
            string address;
            int phoneNumber;

     public:
              user();
              user(string uName, string uAddress, int uPhNo);
                        void addUserDetails();
   ~user;
};

user::user(){

}

user::user(string uName, string uAddress, int uPhN){
  name = uName;
  address = uAddress;
  phoneNumber = uPhNo;
}

void user:: addcustomerDetails(){

}

user::~user(){

}

//class registered customer

class RegisteredCustomer: public user
{
          protected:
                string userName;
                string password;
                Feedback *fb;

          public:
                RegisteredCustomer();
```

```
            RegisteredCustomer(string RegCusUsername, string   RegCusPassword,);
            void addLoginDetails();
             void addFeedback(Feedback *fb) ;
            ~RegisteredCustomer;
};

RegisteredCustomer::RegisteredCustomer(){

}

RegisteredCustomer::RegisteredCustomer(string RegCusUsername, string
RegCusPassword){
        userName = RegCusUsername;
        password = RegCusPassword;
  }
void RegisteredCustomer::addLoginDetails(){

}

void RegisteredCustomer::addFeedback(Feedback *fb){

}

RegisteredCustomer::~RegisteredCustomer(){

}

//class feedback
class  feedback:
{
      private:
            string FeedbackID;
            string comment;
             RegisteredCustomer *cus;

      public:
             feedback();
             feedback(string fID, string fcomment, RegisteredCustomer *rCus);
             void displayFeedback;
             ~feedback;
};

feedback::feedback(){
 }
```

```
feedback::feedback(string fID, string fcomment, RegisteredCustomer *rCus){
        FeedbackID = fID;
        comment = fcomment;
}

void feedback::displayFeedback(){

}

feedback::~feedback(){

}



//Tour Class
class Tour{
    private:
        string TourID;
        string TourName;
        double TourPrice;

    public:
        Tour();
        void setDetails(string tID, string tName);
        void setPrice(double tPrice);
        double getPrice();
        ~Tour;
};

Tour::Tour(){
  TourID = 0;
  TourName = "";
  TourPrice = 0.0;
}

void Tour::setDetails(string tID, string tName){
  TourID = tID;
  TourName = tName;
}

void Tour::setPrice(double tPrice){
  TourPrice = tPrice;
}
```

```
double Tour::getPrice(){
  return TourPrice;
}

Tour::~Tour(){}

//Package Class
class Package{
    private:
            Tour * tours;
            string packageID;
            string packageType;
            int maxSize;
     public:
            package(string packID, string packType, int size);
            void addPackage(string tID, string tName, double tPrice);
            double calcPackTotal();
            ~Package(void);
};

Package::Package(string packID,string packType, int size){
            packageID = packID;
            packageType = packType;
            maxSize = size;
            tours = new Tour[size];
            count = 0;
}

void Package::addItem(string tID, string tName, double tPrice){
  if(count<maxSize){
    tour[count].setDetails(tID,tName);
    tour[count].setPrice(tPrice);
    count++;
  }
  else{
    cout<<"cannot enter anymore packages"<<endl;
  }
}

double Package::calcPackTotal(){
  double total = 0;
  for(int i+0; i<maxSize; i++){
    total = tours[i].getPrice();
```

```
  }
  return total;
}

Package::~Package(){}

//class Administrator
class Administrator{
    private:
            Package * pack;
            string username;
            string password;
            string staffID;
    public:
            Administrator();
            Administrator(string aUsername, string aPassword, string aStaffID);
            void updateTourDetails();
            void updatePackageDetails();
};

Administrator::Administrator(){

}

Administrator::Administrator(string aUsername, string aPassword, string aStaffID, Package *
pack){
  username = aUsername;
  password = aPassword;
  staffID = aStaffID;
  pack = apack;
}

void Administrator::updateTourDetails(){

}
void Administrator::updatePackageDetails(){

}
Administrator::~Administrator(){

}

//class Manage
class Manage{
```

```
    private:
            string tourID;
            string staffID;
            Tour*Tr;
            Administrator*Admin;
    public:
            Manage();
            Manage(string mTourID, string mstaffID, Tour*mtour, Administrator*mAdmin);
            Manage(string mTourID, string mstaffID);
            void cancleRequests();
            void updateTour();
            ~Manage();
};

Manage:: Manage(){

}

Manage:: Manage(string mTourID, string mstaffID, Tour*mtour, Administrator*mAdmin){
  TourID = mTourID;
  staffID = mStaffID;
  Tr = mtour;
  admin = mAdmin;
}

void Manage:: cancleRequests(){

}

void Manage:: updateTour(){

}

Manage:: ~Manage(){

}
```

```
//main program
int main()
{
	Payment *P1 = new Payment("P001", 15000.00);
	Payment *P2 = new Payment("P002", 20000.00);

	Report * R1 = new Report();

	Booking * B1 = new Booking("B001", "2022-05-16", P1);
	Booking * B2 = new Booking("B002", "2022-05-17", P2);

	R1->addNewBooking(B1, B2);
	R1->displayListOfNewBookings();

	B1->displayBooking();
	B2->displayBooking();

	delete P1;
	delete P2;
	delete B1;
	delete B2;
	delete R1;



	  user us("Janaka", "13/A,Colombo 7",0713456789);
	  us.addcustomerDetail();

	  RegisteredCustomer RegCus("Janka33","Jan@5344");
	  RegCus.addLoginDetails();
	  RegCus.giveFeedback();

	  feedback fd("f1322","GG");
	  fd.displayFeedback();



	  double totalPackPrice = 0;

	  Package pk("p13","One-Day",1);
	  pk.addItem("t123","wilpattuwa",1200.00);
	  totalPackPrice = pk.calcPackTotal();
	  cout<<"The price of the package is : "<<totalPackPrice<<endl;
```

```
        Administrator Administ("Malith", "Ma@3313", "s3313");
        Administ.updateTourDetails();
        Administ.updatePackageDetails();

        Manage mng("t123","s3313");
        mng.cancleRequests()
        mng.updateTour()

        return 0;
}
```

Individual Contribution

Student ID	Student Name	Classes
IT21206900	Peiris P.A.S	User class, Registered Customer class, Feedback class
IT21211164	Gayasri B.H.D.B	Booking class, payment class, Report class
IT21211478	Bhagya E.M.P	Tour class, Packages class, Administrator class, Manage class