Topic : Online tour guide system

Group no : KDY_04

Campus : Kandy

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**
IT21235092	Bandara D.M.L.P	0717309707
IT21299520	Kamalhewa Y.U	0776519985
IT21231728	Neelawala P.K.N.G.K.B	0775817204
IT21372162	Bandara D.M.A.T.D	0703385599
IT21260810	Godapolaarachchi G.A.J.G	0741921018

List down the requirements you have identified, for the system you need to implement.

- The system provides 24/7 service throughout the year.
- A visitor can overview the system. To use the system, unregistered user must register to the system by providing personal information such as Full name, Address, NIC, Email, contact number.
- System generates unique customer ID for each customer.
- A registered user can login to the system by providing username and password.
- Registered users can purchase trip packages using the system.
- Admin should be able to manage trip packages by adding, updating, and deleting trip packages.
- A trip package stores package details such as destination, accommodation, facilities, and price.
- System should generate unique package ID for each trip package.
- Customers can search the trip packages from destination, accommodations, and price.
- They can add their feedbacks that for system or trip packages.
- Admin can handle received feedbacks posted on the system.
- Registered users can place a booking by selecting a package.
- After the booking, unique booking ID and a bill is generated.
- Customers are needed to do payment for the selected packages.
- To proceed payment, customer must include payment details like payment type and card details.
- After the payment is done, report generate booking details, payment details, and confirmation details and emailed to the customer.

Noun & verb analysis

(Nouns)

- The system provides 24/7 service throughout the year.
- A visitor can overview the system. To use the system, unregistered user must register to the system by providing personal information such as Full name, Address, NIC, Email, contact number.
- System generates unique customer ID for each customer.
- A registered user can login to the system by providing username and password.
- Registered users can purchase trip packages using the system.
- Admin should be able to manage trip packages by adding, updating, and deleting trip packages.
- A trip package stores package details such as destination, accommodation, facilities, and price.
- System should generate unique package ID for each trip package.
- Customers can search the trip packages from destination, accommodations, and price.
- They can add their feedbacks that for system or trip packages.
- Admin can handle received feedbacks posted on the system.
- Registered users can place a booking by selecting a package.
- After the booking, unique booking ID and a bill is generated.
- Customers are needed to do payment for the selected packages.
- To proceed payment, customer must include payment details like payment type and card details.
- After the payment is done, report generate booking details, payment details, and confirmation details and emailed to the customer.

Classes

- Visitor
- Customer
- Trip package
- Admin
- Feedback
- Booking
- Payment
- Report

Other nouns

Redundant: unregistered user, registered user.

An event or an operation:

Outside the scope: system, year, email.

Meta language: they.

Attributes: personal information (full name, address, NIC, email, contact number), username, password, package details (destination, accommodation, facilities, price), unique customer ID, unique package ID, unique booking ID, date of booking, payment details (payment type, card details), confirmation details.

(Verbs)

- The system provides 24/7 service throughout the year.
- A visitor can overview the system. To use the system, unregistered user must register to the system by providing personal information such as Full name, Address, NIC, Email, contact number.
- System generates unique customer ID for each customer.
- A registered user can login to the system by providing username and password.
- Registered users can purchase trip packages using the system.
- Admin should be able to manage trip packages by adding, updating, and deleting trip packages.
- A trip package stores package details such as destination, accommodation, facilities, and price.
- System should generate unique package ID for each trip package.
- Customers can search the trip packages from destination, accommodations, and price.
- They can add their feedbacks that for system or trip packages.
- Admin can handle received feedbacks posted on the system.
- Registered users can place a booking by selecting a package.
- After the booking, unique booking ID and a bill is generated.
- Customers are needed to do payment for the selected packages.
- To proceed payment, customer must include payment details like payment type and card details.
- After the payment is done, report generate booking details, payment details, and confirmation details and emailed to the customer.

Methods

Visitor -	Overview the system.
	Register to the system.
Customer -	Login to the system
	Place a booking
	Purchase for trip package
	Add feedbacks
	Search trip packages
Trip package –	Store details of the trip package
	Add trip packages to system
	Update trip packages
	Delete trip packages
Admin –	Manage the trip packages
	Handle the feedbacks
Feedback –	Store feedbacks
Booking –	Place a booking
	Generate booking ID and bill
Payment –	Add payment details
	Proceed payment
Report -	Generate booking details
	Generate payment details
	Generate confirmation details

Visitor	
Responsibilities	**Collaborators**
Register to the system Overview the system	Package

Customer	
Responsibilities	**Collaborators**
Login to the system Place a booking Make payment Search packages Add feedbacks	 Booking Payment Packages Feedbacks

Admin	
Responsibilities	**Collaborators**
Manage trip packages Handle feedbacks	packages feedbacks

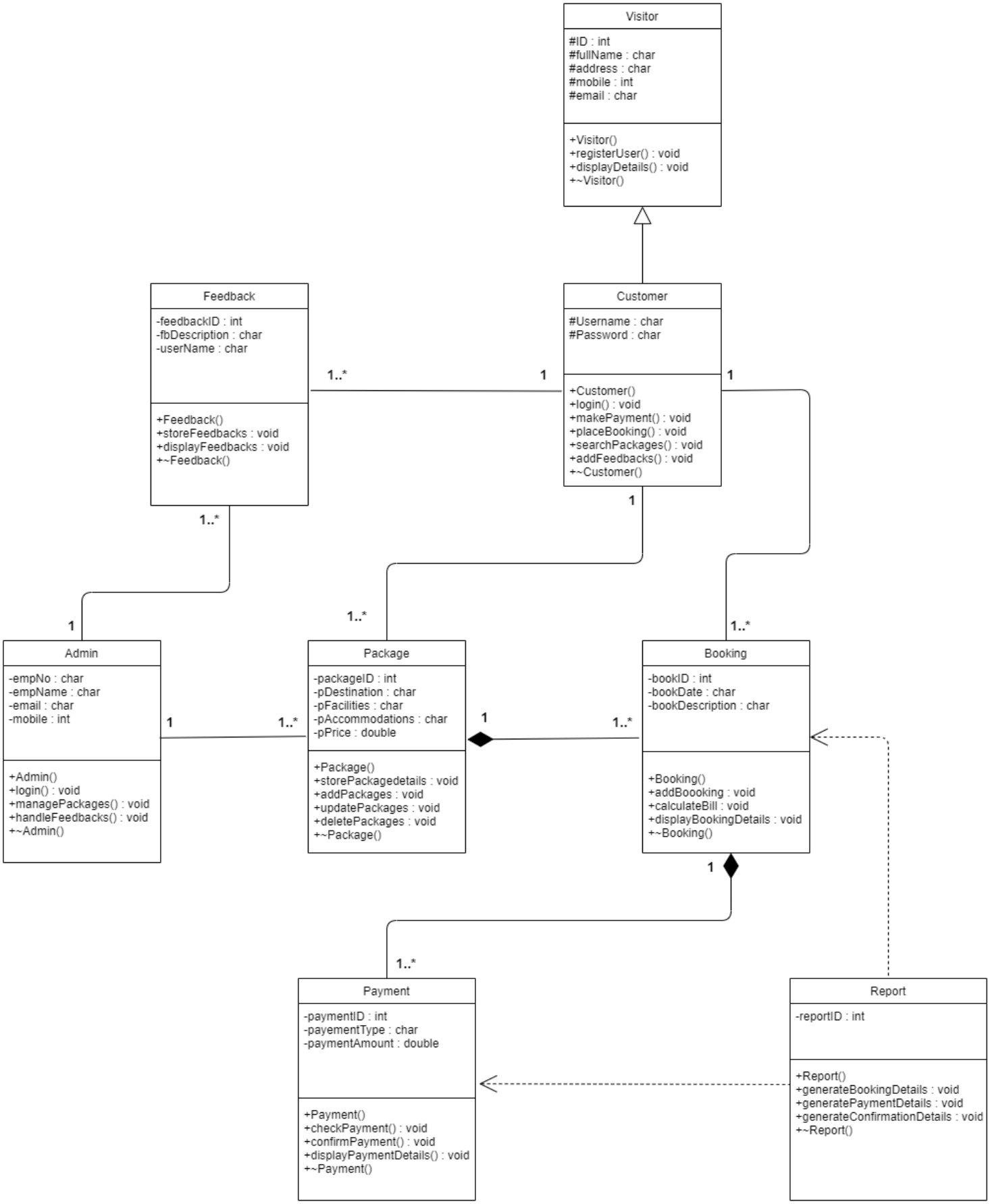
Packages	
Responsibilities	**Collaborators**
Store details of the trip package Add trip packages to system Update trip packages Delete trip packages	

Feedback	
Responsibilities	**Collaborators**
Store feedbacks display feedbacks	

Booking	
Responsibilities	**Collaborators**
Add booking Calculate the bill Display the bill	Payment

Payment	
Responsibilities	**Collaborators**
Check payment details Confirm payment Display payment details	Package, customer

Report	
Responsibilities	**Collaborators**
Generate booking details Generate payment details Generate confirmation details	Package Payment


Visitor
#ID : int
#fullName : char
#address : char
#mobile : int
#email : char
+Visitor()
+registerUser() : void
+displayDetails() : void
+~Visitor()
Feedback
-feedbackID : int
-fbDescription : char
-userName : char
+Feedback()
+storeFeedbacks : void
+displayFeedbacks : void
+~Feedback()
Customer
#Username : char
#Password : char
+Customer()
+login() : void
+makePayment() : void
+placeBooking() : void
+searchPackages() : void
+addFeedbacks() : void
+~Customer()
1..*
1
1
1..*
1
1..*
1..*
1
Admin
-empNo : char
-empName : char
-email : char
-mobile : int
+Admin()
+login() : void
+managePackages() : void
+handleFeedbacks() : void
+~Admin()
Package
-packageID : int
-pDestination : char
-pFacilities : char
-pAccommodations : char
-pPrice : double
+Package()
+storePackagedetails : void
+addPackages : void
+updatePackages : void
+deletePackages : void
+~Package()
Booking
-bookID : int
-bookDate : char
-bookDescription : char
+Booking()
+addBoooking : void
+calculateBill : void
+displayBookingDetails : void
+~Booking()
1
1..*
1
1..*
1
1..*
Payment
-paymentID : int
-payementType : char
-paymentAmount : double
+Payment()
+checkPayment() : void
+confirmPayment() : void
+displayPaymentDetails() : void
+~Payment()
Report
-reportID : int
+Report()
+generateBookingDetails : void
+generatePaymentDetails : void
+generateConfirmationDetails : void
+~Report()

Header files

Visitor.h

```
#pragma once
class Visitor
{
   protected :
        int ID;
        char fullName[50];
        char address[50];
        int mobile;
        char email[20];

   public :
        Visitor();
        Visitor(int V_ID, const char Name[], const char adrs[], int phone, const char V_email[]);
        void login();
        void registerUser();
        void displayDetails();
        ~Visitor();

};
```

Customer.h

```
#pragma once
#include "Visitor.h"
#include "Package.h"
#include "Booking.h"
#include "Feedback.h"

class Package;
class Booking;
class Feedback;
#define SIZE 5

class Customer : public Visitor
{
   protected :
         char Username[30];
         char Password[15];

         Package *pkg[SIZE];
         Booking *bk[SIZE];
         Feedback *fb[SIZE];
```

```
   public :
        Customer();
        Customer(const char C_name[], const char C_pass[], int V_email_ID, const
char Name[], const char adrs[], int phone, const char V_email[]);
        void login();
        void makePayement();
        void placeBooking(Booking *bking);
        void searchPackages(Package *pk);
        void addFeedbacks(Feedback *feed);
        ~Customer();

};
```

Package.h

```
#pragma once
#include "Customer.h"
#include "Admin.h"
#include "Booking.h"
#define SIZE3 5


class Customer;
```

```
class Admin;
class Booking;
class Package
{
   private :
        int packageID;
        char pDescription[200];
        char pFacilities[100];
        char pAccommodation[100];
        double pPrice;

        Customer* custmr;
        Admin* admin;
        Booking* book[SIZE3];



   public :
        Package();
        Package(int book1, int book2, Customer* cus, Admin* adm);
        void storePackagedetails(int pID, const char pDes[], const char pFaci[],
const char pAcc[], double price, Customer* cus, Admin* adm);
        void addPackages();
        void updatePackages();
```

```
        void deletePackages();
        ~Package();
};
```

Admin.h

```
#pragma once
#include "Package.h"
#include "Feedback.h"
#define SIZE1 5

class Package;
class Feedback;

class Admin
{
    private :
        int empNo;
        char empName[30];
        char email[20];
        int mobile;
```

```
        Package *admin[SIZE1];
        Feedback *fbk[SIZE1];

    public :
        Admin();
        Admin(int E_no, const char E_name[], const char E_email[], int E_mobile);
        void login();
        void managePackages(Package *apkg);
        void handleFeedbacks(Feedback *afdk);
        ~Admin();
};
```

Booking.h

```
#pragma once
#include "Customer.h"
#include "Payment.h"
#define SIZE5 2

class Customer;
class Payment;
class Booking
```

```
{
   private :
      int bookID;
      char bookDate[20];
      char bookDescription[100];
      int count=0;

      Customer *bcustmr;
      Payment *payment[SIZE5];


   public :
       Booking();
       Booking(int B_ID);
       Booking(int B_ID, const char B_date[], const char B_desc[],Customer *bcus,
int pay1, int pay2);
       void addBooking();
       void calculateBill(int id, const char ptyp[], double pAmt);
       void displayBookingDetails();
       ~Booking();

};
```

Feedback.h

```
#pragma once
#include "Admin.h"
#include "Customer.h"

class Customer;
class Admin;
class Feedback
{
   private :
        int feedbackID;
        char fbDescription[500];
        char userName[50];

        Customer* Fcus;
          Admin* Fadmin;

   public :
        Feedback();
        Feedback(int F_ID, const char F_des[], const char F_name[], Customer*
Fcust, Admin* Fadm);
        void storeFeedbacks();
        void displayFeedbacks();
```

```
        ~Feedback();

};
```

Payment.h

```
#pragma once

class Payment
{
   private :
        int paymentID;
        char paymentType[20];
        double paymentAmount;

   public :
        Payment();
        Payment(int P_ID, const char P_type[], double P_amnt);
        void checkPayment();
        void confirmPayment();
        void displayPaymeentDetails();
        ~Payment();
```

```
};
```

Report.h

```
#pragma once
#include "Booking.h"
#include "Payment.h"

class Report
{
   private :
        int reportID;

   public :
         Report();
         Report(int R_ID);
         void generateBookDetails(Booking *R_bk);
         void generatePaymentDetails(Payment *R_pay);
         void generateConfirmationDetails();
        ~Report();
};
```

Cpp files

Visitor.cpp

```
#include "Visitor.h"
#include <iostream>
#include <cstring>

Visitor::Visitor()
{
   ID=0;
   strcpy(fullName,"");
   strcpy(address,"");
   mobile=0;
   strcpy(email,"");

}

Visitor::Visitor(int V_ID, const char Name[], const char adrs[], int phone, const char V_email[])
{
   ID=V_ID;
   strcpy(fullName,Name);
   strcpy(address,adrs);
```

```
    mobile=phone;
    strcpy(email,V_email);
}

void Visitor :: login()
{

}

void Visitor :: registerUser()
{

}

void Visitor :: displayDetails()
{

}

Visitor :: ~Visitor()
{
    //destructor
}
```

Customer.cpp

```
#include "Customer.h"
#include <iostream>
#include <cstring>

Customer :: Customer()
{
   strcpy(Username,"");
   strcpy(Password,"");
}

Customer :: Customer(const char C_name[], const char C_pass[], int V_ID, const char Name[], const char adrs[], int phone, const char V_email[])
{
   strcpy(Username,C_name);
   strcpy(Password,C_pass);
}

void Customer :: login()
{

}
```

```
void Customer :: makePayement()
{

}

void Customer :: placeBooking(Booking *bking)
{

}

void Customer :: searchPackages(Package *pk)
{

}

void Customer :: addFeedbacks(Feedback *feed)
{

}

Customer :: ~Customer()
{
```

```
    for (int i=0; i<SIZE; i++)
    {
        delete pkg[i];
    }

}
```

Package.cpp

```
#include "Package.h"
#include<cstring>
#define SIZE4 2



Package::Package()
{


}

Package :: Package(int book1, int book2, Customer* cus, Admin* adm)
{
    custmr = cus;
    admin = adm;
```

```
    book[0]= new Booking(book1);
    book[1]= new Booking(book2);
}

void Package :: storePackagedetails(int pID, const char pDes[], const char pFaci[],
const char pAcc[], double price, Customer* cus, Admin* adm)
{

}

void Package :: addPackages()
{

}

void Package :: updatePackages()
{

}

void Package :: deletePackages()
{
```

```
}

Package :: ~Package()
{
   for(int i=0 ; i<SIZE4 ; i++)
     {
       delete book[i];
     }
}
```

Admin.cpp

```
#include "Admin.h"
#include<cstring>

Admin :: Admin()
{

}

Admin ::Admin(int E_no, const char E_name[], const char E_email[], int E_mobile)
{
```

```
    empNo=E_no;
    strcpy(empName,E_name);
    strcpy(email,E_email);
    mobile=E_mobile;

}

void Admin :: login()
{



}

void Admin :: managePackages(Package *apkg)
{

}

void Admin :: handleFeedbacks(Feedback *afdk)
{

}
```

```
Admin :: ~Admin()
{
   for (int i=0; i<SIZE1; i++)
   {
      delete admin[i];
   }
}
```

Booking.cpp

```
#include "Package.h"
#include<cstring>

Booking :: Booking()
{

}

Booking :: Booking(int B_ID)
{
```

```
  bookID=B_ID;
}

Booking :: Booking(int B_ID, const char B_date[], const char B_desc[],Customer
*bcus, int pay1, int pay2)
{
  bookID=B_ID;
  strcpy(bookDate,B_date);
  strcpy(bookDescription,B_desc);
  bcustmr=bcus;
}

void Booking :: addBooking()
{

}

void Booking :: calculateBill(int id, const char ptyp[], double pAmt)
{
  if(count<SIZE5)
  {
    payment[count] = new Payment(id,ptyp,pAmt);

    count++;
```

```
  }
}

void Booking :: displayBookingDetails()
{

}

Booking :: ~Booking()
{
  for (int i = 0; i < SIZE5; i++)
    {
       delete payment[i];
    }

}
```

Feedback.cpp

```
#include "Feedback.h"
#include<cstring>

Feedback :: Feedback()
```

```
{


}

Feedback :: Feedback(int F_ID, const char F_des[], const char F_name[],
Customer* Fcust, Admin* Fadm)
{
  feedbackID=F_ID;
  strcpy(fbDescription,F_des);
  strcpy(userName,F_name);

  Fcus=Fcust;
  Fadmin=Fadm;
}

void Feedback :: storeFeedbacks()
{


}

void Feedback :: displayFeedbacks()
{


}
```

```
Feedback :: ~Feedback()
{

}
```

Pyament.cpp

```
#include "Payment.h"
#include <cstring>

Payment :: Payment()
{

}

Payment :: Payment(int P_ID, const char P_type[], double P_amnt)
{
  paymentID=P_ID;
  strcpy(paymentType,P_type);
  paymentAmount=P_amnt;

}
```

```
void Payment :: checkPayment()
{

}

void Payment :: confirmPayment()
{

}

void Payment :: displayPaymeentDetails()
{

}

Payment :: ~Payment()
{

}
```

Report.cpp

```
#include "Report.h"
#include <cstring>

Report :: Report()
{

}

Report :: Report(int R_ID)
{
   reportID=R_ID;
}

void Report :: generateBookDetails(Booking *R_bk)
{



}

void Report :: generatePaymentDetails(Payment *R_pay)
{
```

```
}

void Report :: generateConfirmationDetails()
{

}

Report :: ~Report()
{

}
```

Main program

main.cpp

```
#include "Customer.h"
#include "Package.h"
#include "Booking.h"
#include "Admin.h"
#include "Feedback.h"
#include "Payment.h"
#include "Report.h"
int main ()
{

   Visitor *vc = new Customer();

   Package *pk = new Package();

   Booking *bk = new Booking();

   Payment *pmnt = new Payment();

   Admin *ad = new Admin();
```

```
  Feedback *fd = new Feedback();

  Report *rp = new Report();



vc->login();

pk->updatePackages();
pk->deletePackages();

bk->addBooking();
bk->displayBookingDetails();

ad->login();

pmnt->checkPayment();
pmnt->confirmPayment();
pmnt->displayPaymeentDetails();

fd->storeFeedbacks();
fd->displayFeedbacks();
```

```
  rp->generateConfirmationDetails();




  delete vc;
  delete bk;
  delete ad;
  delete pmnt;
  delete fd;
  delete rp;

    return 0;
}
```