Topic : Wild-Life Safari Trip Management System

Group no : MLB_06.01_04

Campus : Malabe

Submission Date : 16/05/2022

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**
IT21306440	Senarath S J U	0714985511
IT21312380	Saubhagya S.D.S.S.	0779018049
IT21309038	Shadhir A.M	0768824668
IT21312212	Wimalarathna S.D.A.N.	0763724334
IT21250088	Gunawardena K.S.S	0761943480

Introduction

Wildlife safari trip management system allows the explorers to conveniently reserve trip packages through online. The system allows everyone to explore the site, but it requires users to create an account and to be logged into the system to select and order a package

The system is designed to validate user accounts and allow customers to select packages add them to cart and then to pay and checkout, Also the system calculates the amount the user should pay checks for available discounts and displays the final amount.

Also, the system generates monthly reports so that it will be convenient to the stake holders to view the monthly sales and the system allows the IT staff to authorize access to the users also the IT staff will be managed by an IT administrator

The objective of the system is to create a user-friendly interface to the users to purchase their desires safari packages.

User Requirements

- An unregistered user can view site and create an account providing his/her details.
- Unregistered customers can visit the packages can contact us, about us and gallery pages without creating an account.
- A registered user can select package's view site and give feedbacks and suggestions.
- A user should login to the system with provided login credentials.
- Registered user can see booking history, manage cart and add or remove packages.
- A registered customer can make payments via credit card, debit card, PayPal and make reservation according to the trip.
- The user can get the information of total cost when making payments.
- The company has few employees such as drivers, IT staff, IT admin, managers to run the company.
- Every driver uses a vehicle to perform a trip.

- IT staff should login to the system with valid login credentials for collect details check and authenticate reservation to customer and verify payment.
- IT staff should update the system and handle the database.
- Users need to contact the IT staff if there are any issues regarding to the system.
- End of every month a report will be generated to the IT administrator.
- Every trip package will have accommodation vehicle food item and a guide.
- Employees are inspected and handled by owner.
- Owner can generate the sales reports and handle salary payments.

Noun Verb Analysis

- An unregistered user can view website and create an account providing his/her details.
- Unregistered customers can visit the packages can contact us, about us and gallery pages without creating an account.
- A registered user can select package's view site and give feedbacks and suggestions.
- A user should login to the system with provided login credentials.
- Registered user can see booking history, manage cart and add or remove packages.
- A registered customer can make payments via credit card, debit card, PayPal and make reservation according to the trip.
- The user can get the information of total cost when making payments.
- The company has few employees such as drivers, IT staff, IT administrator, managers to run the company.
- Every driver uses a vehicle to perform a trip.

- IT staff should login to the system with valid login credentials for collect details check and authenticate reservation to customer and verify payment.
- IT staff should update the system and handle the database.
- Users need to contact the IT staff if there are any issues regarding to the system.
- End of every month a report will be generated to the IT administrator.
- Every trip package will have accommodation, vehicle, food item and a guide.
- Employees are inspected and handled by owner.
- Owner can generate the sales reports and handle salary payments.

- **Nouns**

- Unregistered user
- Website
- Account
- Details
- Unregistered customer
- Package
- Contact us
- About us
- Gallery
- Account
- Register user
- Packages
- Site
- User
- System
- Login credentials
- Register user
- Booking history
- Driver
- Vehicle
- Trip
- IT staff
- System
- Login credentials
- Details
- Reservation
- Customer
- Payments
- IT staff
- System
- Database
- User
- IT staff
- Issues
- System
- Month

- Cart
- Packages
- Registered customer
- Payments
- Credit card
- Debit card
- PayPal
- Reservation
- Trip
- User
- Information
- Payments
- Company
- Employee
- Drivers
- IT staff
- IT administrator
- Manager
- Company

- Report
- IT administrator
- Trip
- Package
- Accommodation
- Vehicle
- Food items
- Guide
- Employee
- Owner
- Owner
- Sales reports
- payments

Class

- Account
- Packages
- Details
- Registered User
- Cart
- Payment
- Trip
- Employee
- Driver
- Vehicle
- Reservation
- Report
- Accommodation
- IT staff
- Feedback

Redundant

- Reservation
- Customer
- Payment
- System
- User
- IT Staff
- System
- IT administrator
- Trip
- Package
- Vehicle

- Employee
- Owner
- Sales report
- Payments
- Unregistered customer
- Account
- Package
- Site
- User
- Registered user
- Packages
- Registered customer
- Use
- Information
- Payments
- Company
- Trip
- IT staff
- System
- Login credential
- Details

Out of scope

- Unregistered user
- Website
- Contact us
- About us
- Gallery
- System

- Login credential
- Booking history
- Credit card
- Debit card
- PayPal
- Company
- IT administrator
- Manager
- Database
- Issues
- Month
- Food item
- Guide
- Owner

- **Verb**
  - View
  - Create
  - Providing
  - Visit
  - Creating
  - Select
  - Give feedback
  - Login
  - See
  - Manage
  - Add
  - Remove
  - Make
  - Get
  - Making
  - Have
  - Run
  - Uses
  - Perform
  - Collect
  - Check
  - Authenticate
  - Verify
  - Update
  - Handle
  - Need
  - Contact
  - Generate
  - Inspected
  - Handled

CRC Cards

Class Name : Account	
Responsibility	**Collaborations**
Store login credentials	
Store reserved package details	Reservations
Get discount according to account type	Payment
Store previous orders	Reservation
Check and validate account	

Class Name : Packages	
Responsibility	**Collaborations**
Store package details	
Display package cost	Cart
Display new packages and offers	

Class Name : Driver	
Responsibility	**Collaborations**
Involve in a trip	Trip
Store driver details	
Uses **v**ehicle	Vehicle
Check vehicle issues and maintenance	

Class Name : Vehicle	
Responsibility	**Collaborations**
Check driver ID	Driver
Add new vehicle details	IT staff
Delete non using vehicle details	IT staff
Update existing vehicle details	IT staff

Class Name : Reservation	
Responsibility	**Collaborations**
Store reservation details	Payment
Validate reservation	IT staff
Display reservation details	

Class Name : Payment	
Responsibility	**Collaborations**
Store payment details	Package
Place the reservation through online transaction methods	
Registered customers do their payments for selected package	Registered user
Will receive discounts	Account
Display payment details	
Online receipt will be generated	Registered user

Class Name : Trip	
Responsibility	**Collaborations**
Store trip details	
Receive a vehicle with a driver	Vehicle, Driver
Display trip details	

Class Name : Employee	
Responsibility	**Collaborations**
Check for system issues	
Check for user issues	Feedback
Store employee details	
Display monthly salary	
Display OT salary	

Class Name : Feedback	
Responsibility	**Collaborations**
Get feedbacks	

Class Name : Report	
Responsibility	**Collaborations**
Generate package details	Package
Create selling report	Cart
Make Payment details	Payment

Class Name : Accommodation	
Responsibility	**Collaborations**
Display Accommodation type	
Preparing place to stay	

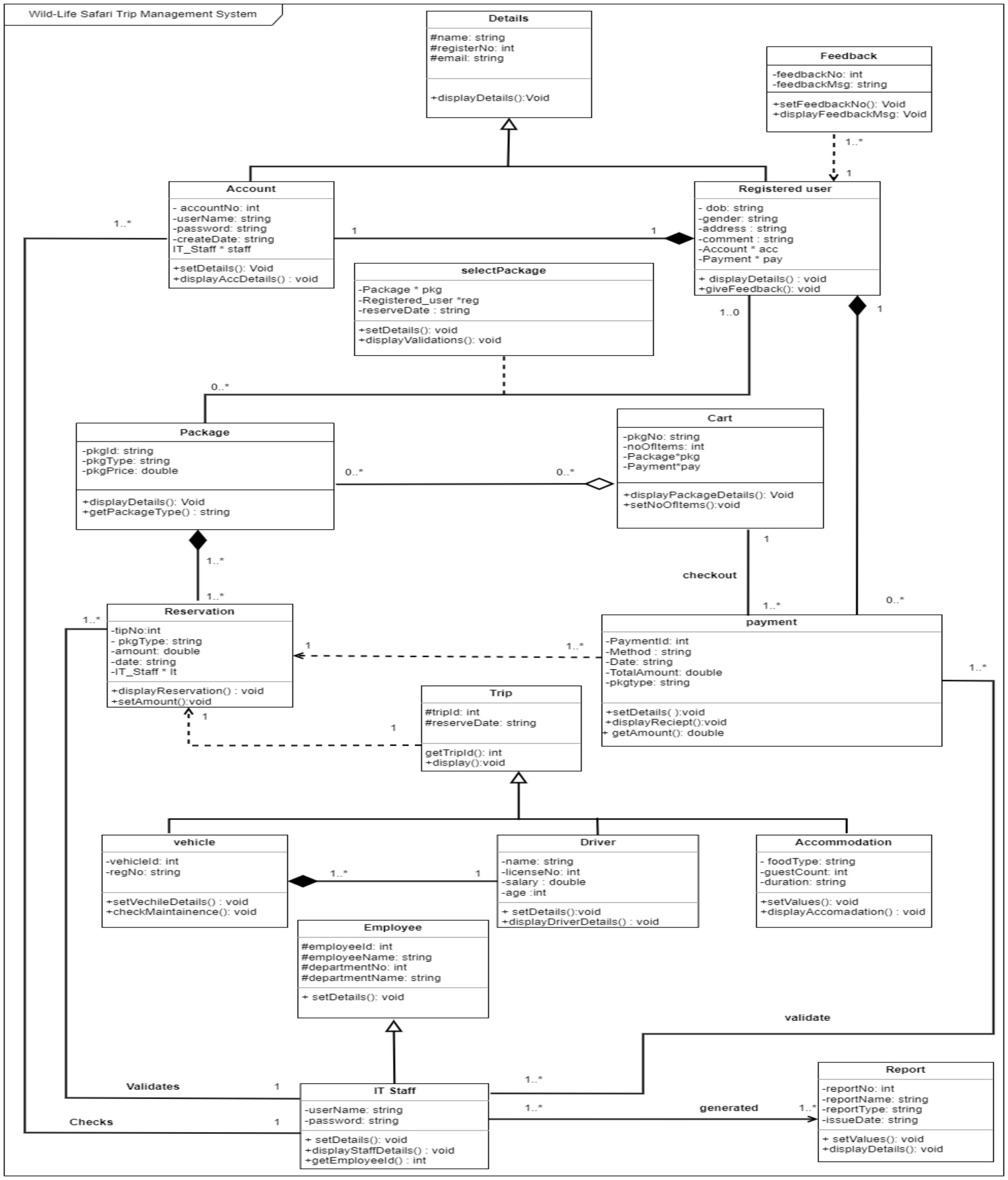
Class Name : IT staff	
Responsibility	**Collaborations**
Log in to the system	Account
Collect details from the User	Details
Check and Authenticate reservation to customer	
Verify Payment details	Payment
Update the system	
Handle the database in the system	
Generate monthly report	Report
Create the sales reports	Report
Handle salary payments	Payment

Class Name : Details	
Responsibility	**Collaborations**
Store user credential	Account
Store registered user details	
Retrieve details to it staff	

Class Name : Cart	
Responsibility	**Collaborations**
Store package detail	Package
Edit cart items	

Class Name : Registered User	
Responsibility	**Collaborations**
Log in to the system	Account, Details
View usage history	Account, Details
Select package	Package
View trip details	Trip, Details
Select members of trip	Trip, Details
Make payment via preferred	Payment
Give feedback and suggestion	Feedback
Contact IT staff if there is an issue	IT staff

Class Diagram



Implementation

Accomodation.h

```
#pragma once
#include <iostream>
#include <string>
#include "Trip.h"
using namespace std;

class Accomodation : public Trip {
private:
    string foodType;
    int guestCount;
    string duration;
public:
    void setValues(string date, string type, int count, string time)
    {
        reserveDate = date;
        foodType = type;
        guestCount = count;
        duration = time;
    }
    void displayAccomodation()
    {
        cout <<
"*********************************************************************************
*******************" << endl;
        cout << "Trip sheduled on" << reserveDate << "your accomodation details will
be" << endl;
        cout << "You choosed food type : " << foodType << endl;               cout
<< "For " << guestCount << " person" << endl;
        cout << "The days of trip will be : " << duration << endl;
        cout <<
"*********************************************************************************
*******************" << endl;
    }
};
```

Account.h

```
#pragma once
#include <iostream>
#include <string>
#include "Details.h"
#include "IT_Staff.h"
using namespace std;

class Account : public Details {
private:
    int accountNo;
    string userName;
    string password;
    string createDate;
    ITStaff* staff;
    string sid;
public:
    Account()
    {
        cout << "Constructor is running" << endl;
    }
    void setDetails(string pname, int regno, string mail, int no, string uname, string
psd, string date)
    {
        name = pname;
        registerNo = regno;
        email = mail;
        accountNo = no;
        userName = uname;
        password = psd;
        createDate = date;
    }
    void displayAccDetails()
    {
        cout << "****************************************************" << endl;
        cout << "Your Details are given below.." << endl;
        cout << "Name :" << name << endl;
        cout << "Register number :" << registerNo << endl;
        cout << "Email ID :" << email << endl;
        cout << "Account number :" << accountNo << endl;
        cout << "User Name :" << userName << endl;
        cout << "Password :" << password << endl;
        cout << "Created Date :" << createDate << endl;
        cout << "***************************************************" << endl;
    }
    ~Account()
    {
        cout << "Running Deconstructor" << endl;
    }

};
```

Cart.h

```cpp
#pragma once
#include<iostream>
#include<string>
#include "Package.h"
#include "Payment.h"
using namespace std;
//creating class
class Cart
{
private:
    string pkgNo;
    int noOfItems;

    Package* pkg;
    Payment* pay;
    string pkgtype;

public:
    Cart() {}
    void displayPackageDetails(Package* pkg)
    {

        pkg->displayDtails();
    }
    void setNoOfItems(int nItems)
    {
        noOfItems = nItems;
    }
    void checkoutPaymentType(Payment* pay)
    {
        pkgtype = pay->getPayType();
    }
};
```

Details.h

```
#pragma once
#include <string>
#include <iostream>
using namespace std;

//creating class

class Details
{
protected:
    string name;
    int registerNo;
    string email;
public:
    //default constructor
    Details()
    {
        cout << "setting default values" << endl;
        name = "xxxxxx";
        registerNo = 0;
        email = "abc@gmail.com";
    }
    //display fucntion
    virtual void displayDetails()
    {
        cout << name << "" << registerNo << email << "" << endl;
    }

    ~Details()
    {
        cout << "**" << endl;
        cout << "deleting details" << endl;
        cout << "**" << endl;
    }

};
```

Driver.h

```
#pragma once
#include <iostream>
#include <string>
#include <cstring>
#include "Trip.h"
using namespace std;

class Driver : public Trip {
private:
    string name;
    int licenseNo;
    double Salary;
    int Age;
public:
    //set values
    Driver(string Dname, int licNo, double salary, int age)
    {
        name = Dname;
        licenseNo = licNo;
        Salary = salary;
        Age = age;
    }
    //displaying details
    void displaydriverDetails()
    {
        cout << "============================" << endl;
        cout << "Driver Name : " << name << endl;
        cout << "Driver License : " << licenseNo << endl;
        cout << "Salary: " << Salary << endl;
        cout << "Age: " << Age << endl;
        cout << "============================" << endl;


    }
    //deconstructors
    ~Driver()
    {
        cout << "deleting Driver" << endl;

    }
};
```

Employee.h

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

class Employee {
protected:
    int employeeId;
    string employeeName;
    int departmentNo;
    string departmentName;
public:
    Employee()
    {
        cout << "Constructor is running" << endl;
        employeeId = 001;
        employeeName = ("Janindu");
        departmentNo = 101;
        departmentName = ("IT dep");
    }
    void setEmployeeDetails(int EempID, string EempName, int EdepNo, string
        EdepName)
    {
        employeeId = EempID;
        employeeName = EempName;
        departmentNo = EdepNo;
        departmentName = EdepName;
    }

};
```

Feedback.h

```
#pragma once
#include <string>
#include <iostream>
using namespace std;

class Feedback {
private:
    int feedbackNo;
    string feedbackMsg;
public:
    Feedback()
    {
        cout << "Constructor runs" << endl;
        feedbackNo = 0;
    }
    int getFeedbackNo()
    {
        return feedbackNo;
    }
    string getFeedback()
    {
        return feedbackMsg;
    }
    void displayFeedbackMsg()
    {
        cout << "Feedback No " << feedbackNo << " message " << feedbackMsg << " has 
been accepted,Thank you for contacting us." << endl;
    }
    ~Feedback()
    {
        cout << "Deconstructor is runnig" << endl;
    }
};
```

IT_Staff.h

```
#pragma once
#include <iostream>
#include <string>
#include "Employee.h"
using namespace std;
class ITStaff :public Employee
{
private:
    string Username;
    string Password;
public:
    int getEmployeeID()
    {
        return employeeId;
    }

    //Constructor
    ITStaff(string Uname, string ppassword, int pID, string PName, int DNo, string
Dname)
    {
        Username = Uname;
        Password = ppassword;
        employeeId = pID;
        employeeName = PName;
        departmentNo = DNo;
        departmentName = Dname;
    }
    //Destructor
    ~ITStaff()
    {
        cout << "Deleted Successfully" << endl;
    }

    void displayDetails()
    {
        cout << " ITStaff Account Details" << endl;
        cout << "@====================================@" << endl;

        cout << "USERNAME          \t" << Username << endl;
        cout << "PASSWORD          \t" << Password << endl;
        cout << "Employee ID       \t" << employeeId << endl;
        cout << "Employeee Name    \t" << employeeName << endl;
        cout << "Department Number \t" << departmentNo << endl;
        cout << "Department Name   \t" << departmentName << endl;

        cout << "@====================================@" << endl;
    }
    void setDetails(string Uname, string password)
```

```
        {
            Username = Uname;
            Password = password;
        }
    };
```

Package.h

```
#pragma once
#include <string>
#include <iostream>
#include "Resevation.h"
using namespace std;

//creating class
class Package {
private:
    string pkgId;
    string pkgType;
    double pkgPrice;
    Reservation* res;
public:
    //setting default values
    Package()
    {
        pkgId = "p000";
        pkgType = "abcd";
        pkgPrice = 0000.00;
        res = new Reservation();
    }
    // overloading constructor
    Package(string pId, string pType, double pPrice)
    {
        pkgId = pId;
        pkgType = pType;
        pkgPrice = pPrice;
    }
    void displayDtails()
    {
        cout << "================================" << endl;
        cout << "pkgId:" << pkgId << endl;
        cout << "pkgType:" << pkgType << endl;
        cout << "pkgPrice:" << pkgPrice << endl;
        cout << "================================" << endl;
    }
    string getPackageType()
    {
        return pkgType;
    }
    ~Package()
    {
        cout << "Delete Package" << endl;
    }
};
```

Payment.h

```
#pragma once
// Payment

#include <iostream>
#include <string>
#include "Cart.h"
#include "IT_Staff.h"
using namespace std;
class Payment {
private:
    int PaymentId;
    string Method;
    string Date;
    double TotalAmount;
    string pkgType;
    ITStaff* stf[20];
public:
    Payment()
    {
        cout << "Constructor is running" << endl;
        PaymentId = 0;
        Method = "None";
        Date = "00-00-0000";
        TotalAmount = 00.00;
        pkgType = "None";


    }
    Payment(int pID, string Mtd, string Dt, double tamount, string Pt)
    {
        PaymentId = pID;
        Method = Mtd;
        Date = Dt;
        TotalAmount = tamount;
        pkgType = Pt;
    }
    void displayPaymentDetails()
    {
        cout << "*******" << endl;
        cout << "Your Details are given below" << endl;
        cout << "Payment ID :  " << PaymentId << endl;
        cout << "Payment Method :  " << Method << endl;
        cout << "Payment Date :  " << Date << endl;
        cout << "Total Amount :" << TotalAmount << endl;
        cout << "Package Type:" << pkgType << endl;
        cout << "******" << endl;

    }
    double getAmount()
```

```
        {
            return TotalAmount;
        }
        string getPayType()
        {
            return pkgType;
        }

        ~Payment()
        {
            cout << "Deleting Payment" << PaymentId << endl;
        }
    };
```

Registerd_User.h

```
#pragma once
#include <iostream>
#include "Details.h"
#include "Account.h"
#include "Payment.h"
#include "Feedback.h"
#include <string>

class Registered_User : public Details
{
private:
    string dob;
    string gender;
    string address;
    string comment;
    int fno;
    Account* acc;
Payment* pay; public:
    //setting default value
    Registered_User()
    {
        dob = "00/00/0000";
        gender = "ggggg";
        address = "abc";
        comment = "xxxx";
        fno = 0;
        acc = new Account();
        pay = new Payment();

    }

    //set valuesby overloading constuctor
    Registered_User(string pname, int pregisterNo, string pemail, string pdob,
string pgender, string paddress, string pcomment)
    {
        name = pname;
        registerNo = pregisterNo;
        email = pemail;
        dob = pdob;
        gender = pgender;
        address = paddress;
        comment = pcomment;
    }
    //displaying details
    void displayDetails()
    {
        cout << "============================" << endl;
        cout << "Name : " << name << endl;
        cout << "Registered number : " << registerNo << endl;
```

```
            cout << "Email : " << email << endl;
            cout << "Date of Birth : " << dob << endl;
            cout << "Gender : " << gender << endl;
            cout << "Address : " << address << endl;
            cout << "============================" << endl;


        }
        //deconstructors
        ~Registered_User()
        {
            cout << "deleting register user" << endl;

        }
        //give feedback

        void giveFeedback(Feedback* fdb)
        {
            cout << "Enter feedback number (integer) : ";
            cin >> fno;
            fno = fdb->getFeedbackNo();

            cout << "Enter feedback : ";
            cin >> comment;
            comment = fdb->getFeedback();

        }


    };
```

Report.h

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

class Report {
private:
    int reportNo;
    string reportName;
    string reportType;
    string issueDate;
public:
    Report()
    {
        cout << "Report default constructor called";
        reportNo = 10;
        reportName = ("user");
        reportType = ("pdf");
        issueDate = "04/04/2022";

    }
    Report(int PrepNo, string PrepName, string PrepType, string PiDate)
    {
        reportNo = PrepNo;
        reportName = PrepName;
        reportType = PrepType;
        issueDate = PiDate;
    }

    void displayReport()
    {
        cout << "Report Number : " << reportNo << endl;
        cout << "Report Name : " << reportName << endl;
        cout << "Report Type : " << reportType << endl;
        cout << "issue Date : " << issueDate << endl;

    }
};
```

Resevation.h

```
#pragma once
#include<iostream>
#include "Payment.h"
#include "Trip.h"
#include "IT_Staff.h"
#include<cstring>
using namespace std;

//class creation of reservation
class Reservation
{
private:
    int tripNo;
    string pkgId;
    double amount;
    string date;
    ITStaff* It;
public:
    //seting default values to class
    Reservation()
    {
        tripNo = 0;
        pkgId = "000";
        amount = 00.0;
        date = "00/00/0000";

    }
    //seting values by overloading constuctors
    Reservation(string ppkgId, string pdate)
    {

        pkgId = ppkgId;
        date = pdate;

    }
    //did not overloade trip and amount because they are taken as references
    //setting amount
    void setAmount(Payment* pay)
    {
        amount = pay->getAmount();
    }
    //setting trip
    void setId(Trip* tr)
    {
        tripNo = tr->getTripId();

    }
    void diplayReservation()
    {
        cout << "=======================" << endl;
        cout << "Trip number : " << tripNo << endl;
        cout << "Package ID : " << pkgId << endl;
        cout << "Amount : " << amount << endl;
        cout << "Date : " << date << endl;
        cout << "=======================" << endl;
    }

    //destructor for delete reservation
    ~Reservation()
    {
        cout << "deleting Reservation " << endl;
    }


};
```

Selectpackage.h

```
#pragma once
#include<iostream>
#include<cstring>
#include "Registerd_User.h"
#include "Package.h"
using namespace std;

//creating Association  class
class selectpackage
{
private:
    Package* pkg;
    Registered_User* reg;
    string reserveDate;
public:
    //default constructor for set default values
    selectpackage()
    {
        reserveDate = "00/00/0000";
    }

    //function to set details
    void setDetails(string preserveDate)
    {
        reserveDate = preserveDate;
    }

    //validating package selection
    void diplayValidation()
    {
        if (pkg->getPackageType() == "")
        {
            cout << "invalid package!, try again" << endl;

        }
        else
        {
            cout << "successfull selection" << endl;
        }
    }

    //destructor for delete details of select packages
    ~selectpackage()
    {
        cout << "deleting select package details" << endl;

    }


};
```

Trip.h

```cpp
#pragma once
#include<iostream>
#include<string>
using namespace std;

//creating inheritance class
class Trip
{
protected:
    int tripId;
    string reserveDate;
public:
    //setting default valuesby default constructor
    Trip()
    {
        tripId = 0;
        reserveDate = "00/00/0000";

    }
    //overloading constructor to set value
    Trip(int ptripId, string preserveDate)
    {
        tripId = ptripId;
        reserveDate = preserveDate;

    }
    //display details
    virtual void display()
    {
        cout << "=========================" << endl;
        cout << "Trip Id : " << tripId << endl;
        cout << "Date : " << reserveDate << endl;
        cout << "=========================" << endl;
    }
    //returning tripId
    int getTripId()
    {
        return tripId;
    }

    //destructor to delete data
    ~Trip()
    {
        cout << "deleting trip" << endl;

    }

};
```

Vehicle.h

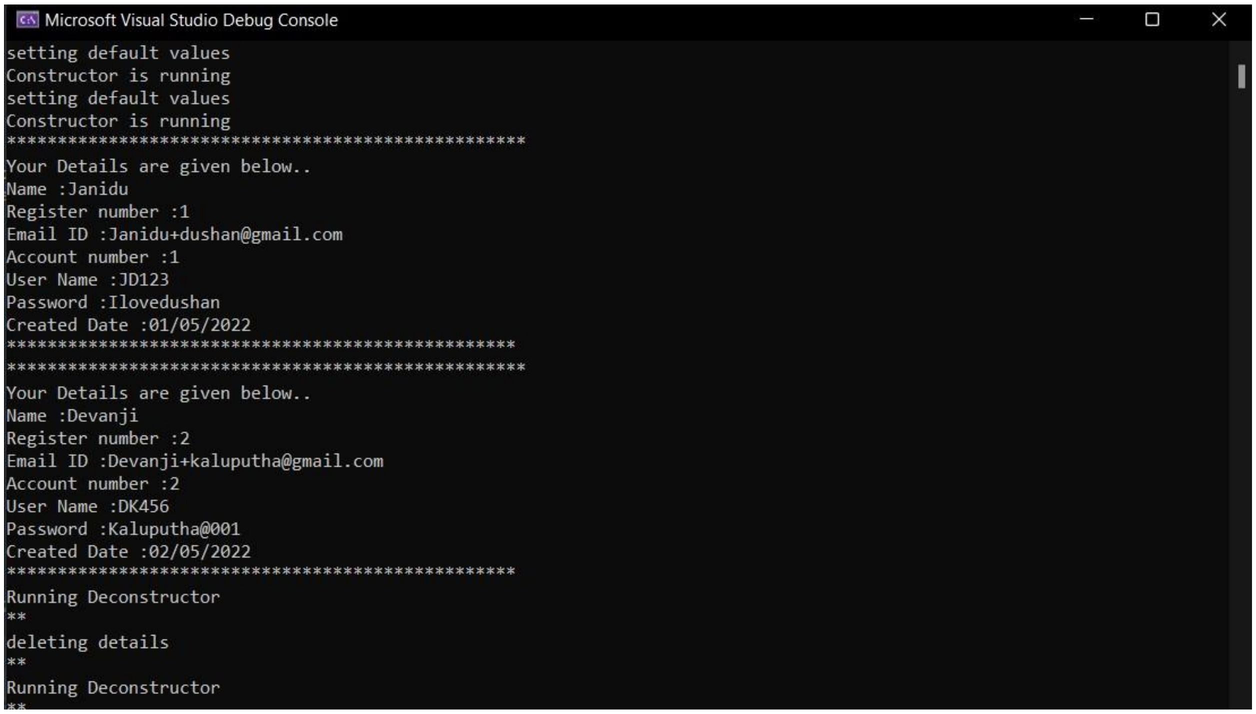
```cpp
#pragma once
#include "Driver.h"
#include <iostream>
#include <string>

using namespace std;

class Vehicle {
private:
    int VehicleID;
    string RegNO;
    Driver* drv[2];
public:
    Vehicle()
    {
        VehicleID = 0;
        RegNO = "000";
    }
    //set values
    Vehicle(int vehicleID, string regNo)
    {
        VehicleID = vehicleID;
        RegNO = regNo;
    }

    void setVehicleDetails(int vehicleID, string regNo) {
        VehicleID = vehicleID;
        RegNO = regNo;
    }

    //displaying details

    void displayDetails()
    {
        cout << "=============================" << endl;
        cout << " Vehicle ID Number : " << VehicleID << endl;
        cout << " Register Number   : " << RegNO << endl;
        cout << "=============================" << endl;

    }
    ~Vehicle()
    {
        cout << "Deleting Vehicle" << endl;

    }

};
```

Main.cpp

```
#include <iostream>
#include <string>
#include "Accomodation.h"
#include "Account.h"
#include "Cart.h"
#include "Details.h"
#include "Driver.h"
#include "Employee.h"
#include "feedback.h"
#include "IT_Staff.h"
#include "Package.h"
#include "Payment.h"
#include "Registerd_User.h"
#include "Report.h"
#include "Resevation.h"
#include "selectpackage.h"
#include "Trip.h"
#include "Vehicle.h"
using namespace std;
int main()
{
    //Accomodation Class object creation
    Accomodation acmo1, acmo2;

    acmo1.setValues("10/05/2022", "Non Veg food", 5, "2 days");
    acmo2.setValues("15/05/2022", "Veg food", 10, "3 days");

    acmo1.displayAccomodation();
    cout << endl;
    acmo2.displayAccomodation();

    //Account Class object cretion
    Account acc1, acc2;

    acc1.setDetails("Janidu", 1, "Janidu+dushan@gmail.com", 1, "JD123",
"Ilovedushan", "01/05/2022");
    acc2.setDetails("Devanji", 2, "Devanji+kaluputha@gmail.com", 2, "DK456",
"Kaluputha@001", "02/05/2022");

    acc1.displayAccDetails();
    acc2.displayAccDetails();

    acc1.~Account();
    acc2.~Account();


    //Cart Class object creation
    Package* pkg10 = new Package;
    Package* pkg20 = new Package;
```

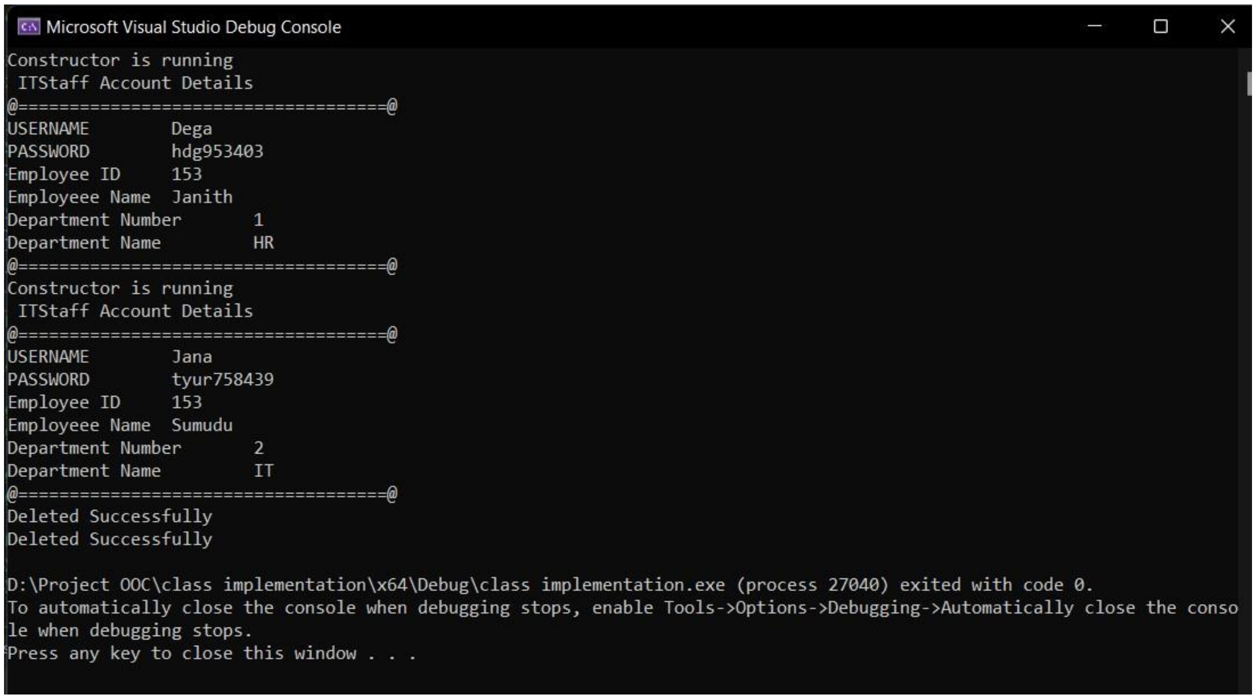
```
Cart* crt1 = new Cart();
Cart* crt2 = new Cart();


crt1->setNoOfItems(2);
crt2->setNoOfItems(3);

crt1->displayPackageDetails(pkg10);
crt2->displayPackageDetails(pkg20);

//Driver Class object creation
Driver* Dvr1 = new Driver("Sena", 234567, 20000, 24);
Dvr1->displaydriverDetails();

Driver* Dvr2 = new Driver("Bandara", 897641, 21500, 56);
Dvr2->displaydriverDetails();
delete Dvr1;
delete Dvr2;

//Feedback Class object creation
Feedback fb1, fb2;

fb1.displayFeedbackMsg();
fb2.displayFeedbackMsg();

//IT_Staff Class object creation
ITStaff* IT1 = new ITStaff("Dega", "hdg953403", 00231, "Janith", 01, "HR");
IT1->displayDetails();

ITStaff* IT2 = new ITStaff("Jana", "tyur758439", 00231, "Sumudu", 02, "IT");
IT2->displayDetails();
delete IT1;
delete IT2;

//Package Class object creation
Package pkg1("P001", "Gold", 1500.00);
Package pkg2("P002", "Premium", 2000.00);

pkg1.displayDtails();
pkg2.displayDtails();

pkg1.~Package();
pkg2.~Package();

//Payment Class object creation
Payment* P1 = new Payment(1012, "Credit card", "03-04-2022", 25000.00, "Gold");
Payment* P2 = new Payment(1015, "Credit card", "07-04-2022", 35000.00,
    "Silver");
P1->displayPaymentDetails();
```

```
        P2->displayPaymentDetails();
        cout << "*******" << endl;
        delete P1;
        delete P2;

        //Registerd_User Class object creation
        Feedback* fdb1 = new Feedback;
        Feedback* fdb2 = new Feedback;
        Registered_User* reg_user1 = new Registered_User("Devanja", 1, "vali@gamil.com",
    "04/07/2000", "Female", "No21 valivita rd,malabe", "null");
        reg_user1->giveFeedback(fdb1);
        reg_user1->displayDetails();
        reg_user1 -> ~Registered_User();

        Registered_User* reg_user2 = new Registered_User("Bandara", 1,
    "devkaluR@gamil.com", "21/07/2000", "Male", "No88 vihara rd,malabe", "null");
        reg_user2->giveFeedback(fdb2);
        reg_user2->displayDetails();
        delete reg_user2;

        //Report Class object creation
        Report* r1 = new Report(1, "Sales", "Word doc", "03-07-2022");
        Report* r2 = new Report(2, "vehicle", "PDF", "04-07-2022");
        r1->displayReport();
        r2->displayReport();
        cout << "******" << endl;

        //Resevation Class object creation
        Payment* pay1 = new Payment;
        Trip* tr1 = new Trip;
        Reservation* resev1 = new Reservation("001", "07/02/2022");
        resev1->setAmount(pay1);
        resev1->setId(tr1);
        resev1->diplayReservation();
        delete resev1;

        Payment* pay2 = new Payment;
        Trip* tr2 = new Trip;
        Reservation* resev2 = new Reservation("002", "09/05/2022");
        resev2->setAmount(pay2);
        resev2->setId(tr2);
        resev2->diplayReservation();
        delete resev2;

        //Vehile Class object creation
        Vehicle* Vehi1 = new Vehicle(23, "A001");
        Vehi1->displayDetails();
        delete Vehi1;


        Vehicle* Vehi2 = new Vehicle(24, "A002");
        Vehi2->displayDetails();
        delete Vehi2;


        return 0;
    };
```



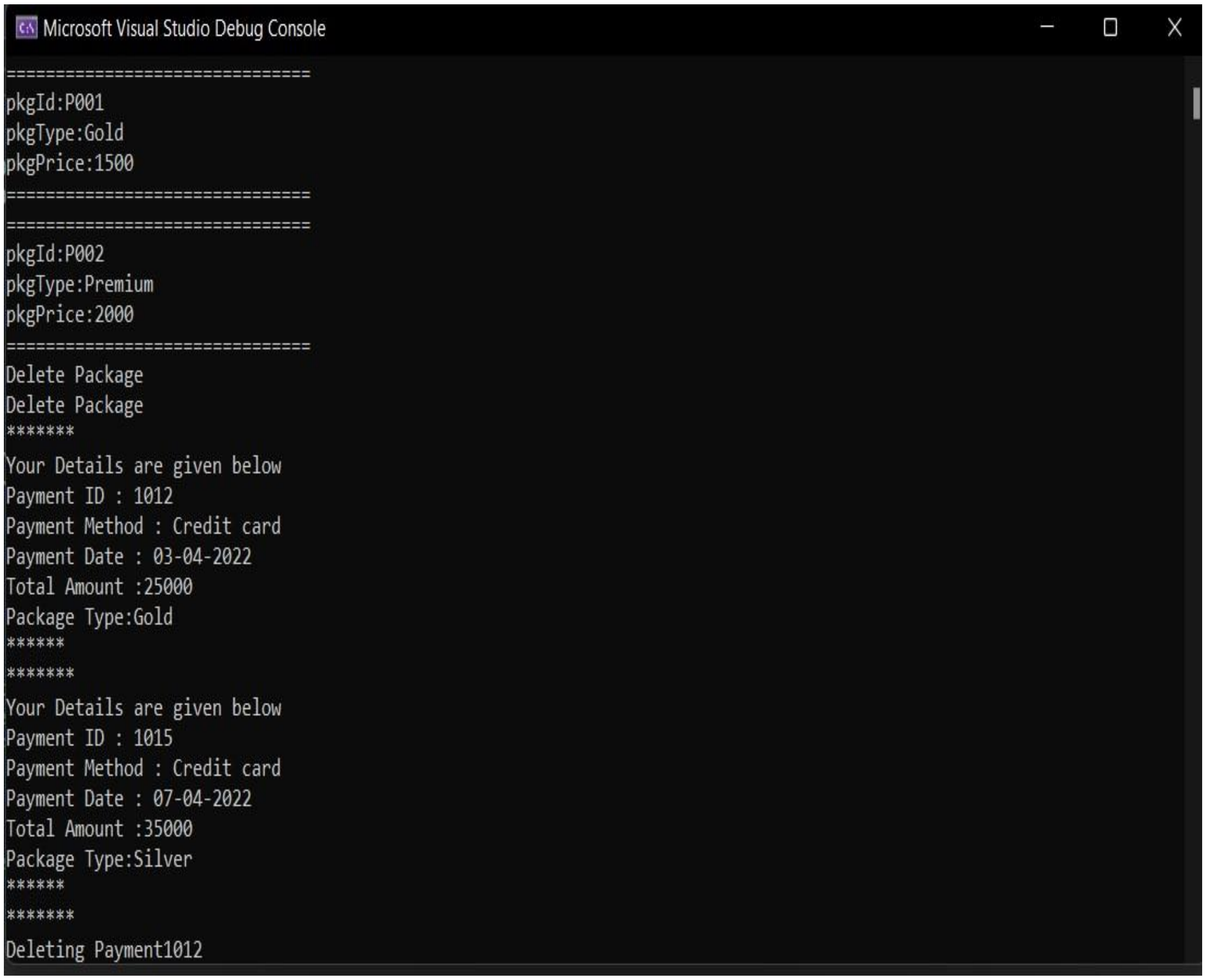
Microsoft Visual Studio Debug Console

```
setting default values
Constructor is running
setting default values
Constructor is running
***************************************************
Your Details are given below..
Name :Janidu
Register number :1
Email ID :Janidu+dushan@gmail.com
Account number :1
User Name :JD123
Password :Ilovedushan
Created Date :01/05/2022
**************************************************
***************************************************
Your Details are given below..
Name :Devanji
Register number :2
Email ID :Devanji+kaluputha@gmail.com
Account number :2
User Name :DK456
Password :Kaluputha@001
Created Date :02/05/2022
**************************************************
Running Deconstructor
**
deleting details
**
Running Deconstructor
```



Microsoft Visual Studio Debug Console

```
Constructor is running
 ITStaff Account Details
@====================================@
USERNAME        Dega
PASSWORD        hdg953403
Employee ID     153
Employeee Name  Janith
Department Number       1
Department Name         HR
@====================================@
Constructor is running
 ITStaff Account Details
@====================================@
USERNAME        Jana
PASSWORD        tyur758439
Employee ID     153
Employeee Name  Sumudu
Department Number       2
Department Name         IT
@====================================@
Deleted Successfully
Deleted Successfully

D:\Project OOC\class implementation\x64\Debug\class implementation.exe (process 27040) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .
```



Microsoft Visual Studio Debug Console

```
===============================
pkgId:P001
pkgType:Gold
pkgPrice:1500
===============================
===============================
pkgId:P002
pkgType:Premium
pkgPrice:2000
===============================
Delete Package
Delete Package
*******
Your Details are given below
Payment ID : 1012
Payment Method : Credit card
Payment Date : 03-04-2022
Total Amount :25000
Package Type:Gold
******
*******
Your Details are given below
Payment ID : 1015
Payment Method : Credit card
Payment Date : 07-04-2022
Total Amount :35000
Package Type:Silver
******
*******
Deleting Payment1012
```

Individual Contribution

Registration No	Name	Contribution
IT21306440	Senarath S J U	Contributed to crate the user requirement. Contributed to perform the noun verb analysis. Contributed to create Driver, Vehicle and Reservation in CRC Cards. Contributed to create the class diagram. Contributed to create and implement Payment, Employee and Report Class and created objects and implemented main program for the above classes.
IT21312380	Saubhagya S.D.S.S.	Contributed to crate the user requirement. Contributed to perform the noun verb analysis. Contributed to create Details, Registered User and Cart in CRC Cards. Contributed to create the class diagram. Contributed to crate and implement selectpackage,

		Registered_User, Trip, Reservation Class and created objects and implemented main program for the above classes.
IT21309038	Shadhir A.M	Contributed to crate the introduction Contributed to identify the classes in noun verb analysis Contributed to create Account, Packages and Feedback in CRC Cards. Contributed to create the class diagram. Contributed to create and implement Account, Accommodation and Feedback Class and created objects and implemented main program for the above classes.
IT21312212	Wimalarathna S.D.A.N.	Contributed to crate the user requirement. Contributed to perform the noun verb analysis. Contributed to create Report, Accommodation, and IT_Staff in CRC Cards. Contributed to create the class diagram.

		Contributed to create and implement IT_Staff, Driver and Vehicle Class and created objects and implemented main program for the above classes.
IT21250088	Gunawardena K.S.S	Contributed to create the user requirement. Contributed to perform the noun verb analysis. Contributed to create Payment, Trip and Employee in CRC Cards. Contributed to create the class diagram. Contributed to create and implement Packages, Details and Cart Class and created objects and implemented main program for the above classes.