Topic : Bus Scheduling & Booking System

Group no : MLB_04.02_02

Campus : Malabe

Submission Date : 18/05/2022

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
IT21267222	EDIRISINGHE E.A.K.G.	0769618496
IT21251450	A.L.V.J. Atapattu	0762328714
IT21204784	W.D.O.D Wijerupa	0773391865
IT21267086	Hasara L.A.N	0704432319
IT21267772	Kumanayake I. A.	0766502778

Requirements

- Customers can visit the bus scheduling and booking system website.
- There are two types of customers as unregistered users and registered users.
- An unregistered user can become a registered user by creating an account providing details such as name, date of birth, NIC, email, address and contact number.
- Registered users can login to the system using their username and password.
- Both unregistered users and registered users can search available bus routes.
- Unregistered users and registered users can view bus schedules by selecting the starting city, destination city, date and bus type.
- A registered user can continue to make a reservation by selecting a bus operator and entering the number of seats required.
- Registered users can also view offers they are eligible for.
- After confirming the reservation, a registered user can make the payment using a debit card or credit card.
- A registered user can view the invoice generated after a successful payment.
- A registered user can rate the service and provide feedback.
- A registered user can update or cancel the reservation.
- Registered users can edit their profile details.
- Administrator manages bus routes and bus operators.
- Administrator manages offers.
- Administrator can generate list of reservations and list of payments made.
- Administrator can manage feedbacks and ratings.

Noun Verb analysis
(Nouns)

- Customers can visit the bus scheduling and booking website.
- There are two types of customers as unregistered users and registered users.
- An unregistered user can become a registered user by creating an account providing details such as name, date of birth, NIC, email, address and contact number.
- Registered users can login to the system using their username and password.
- Both unregistered users and registered users can search available bus routes.
- Unregistered users and registered users can view bus schedules by selecting the starting city, destination city, date and bus type.
- A registered user can continue to make a reservation by selecting a bus operator and entering the number of seats required.
- Registered users can also view offers they are eligible for.
- After confirming the reservation, a registered user can make the payment using a debit card or credit card.
- A registered user can view the invoice generated after a successful payment.
- A registered user can rate the service and provide feedback.
- A registered user can update or cancel the reservation.
- Registered users can edit their profile details.
- Administrator manages bus routes and bus operators.
- Administrator manages offers.
- Administrator can generate list of reservations and list of payments made.
- Administrator can manage feedbacks and ratings.

Identified classes

- Unregistered user
- Registered user
- Bus route
- Bus operator
- Offer
- Reservation
- Payment
- Feedback

Noun Verb analysis
(Verbs)

- Customers can visit the bus scheduling and booking system website.
- There are two types of customers as unregistered users and registered users.
- An unregistered user can become a registered user by creating an account providing details such as name, date of birth, NIC, email, address and contact number.
- Registered users can login to the system using their username and password.
- Both unregistered users and registered users can search available bus routes.
- Unregistered users and registered users can view bus schedules by selecting the starting city, destination city, date and bus type.
- A registered user can continue to make a reservation by selecting a bus operator and entering the number of seats required.
- Registered users can also view offers they are eligible for.
- After confirming the reservation, a registered user can make the payment using a debit card or credit card.
- A registered user can view the invoice generated after a successful payment.
- A registered user can rate the service and provide feedback.
- A registered user can update or cancel the reservation.
- Registered users can edit their profile details.
- Administrator manages bus routes and bus operators.
- Administrator manages offers.
- Administrator can generate list of reservations and list of payments made.
- Administrator can manage feedbacks and ratings.

CRC cards

Class Name: Unregistered User	
Responsibility	**Collaborators**
Register	
Search bus route	Bus Route
View bus schedule	Bus Operator

Class Name: Registered User	
Responsibility	**Collaborators**
Login	
Search bus route	Bus Route
View bus schedule	Bus Operator
Edit profile details	

Class Name: Bus Route	
Responsibility	**Collaborators**
Store bus route details	Bus operator

Class Name: Bus operator	
Responsibility	**Collaborators**
Store bus operator details	
Store bus schedule details	

Class Name: Offer	
Responsibility	**Collaborators**
Store offer details	
View offer	Registered User

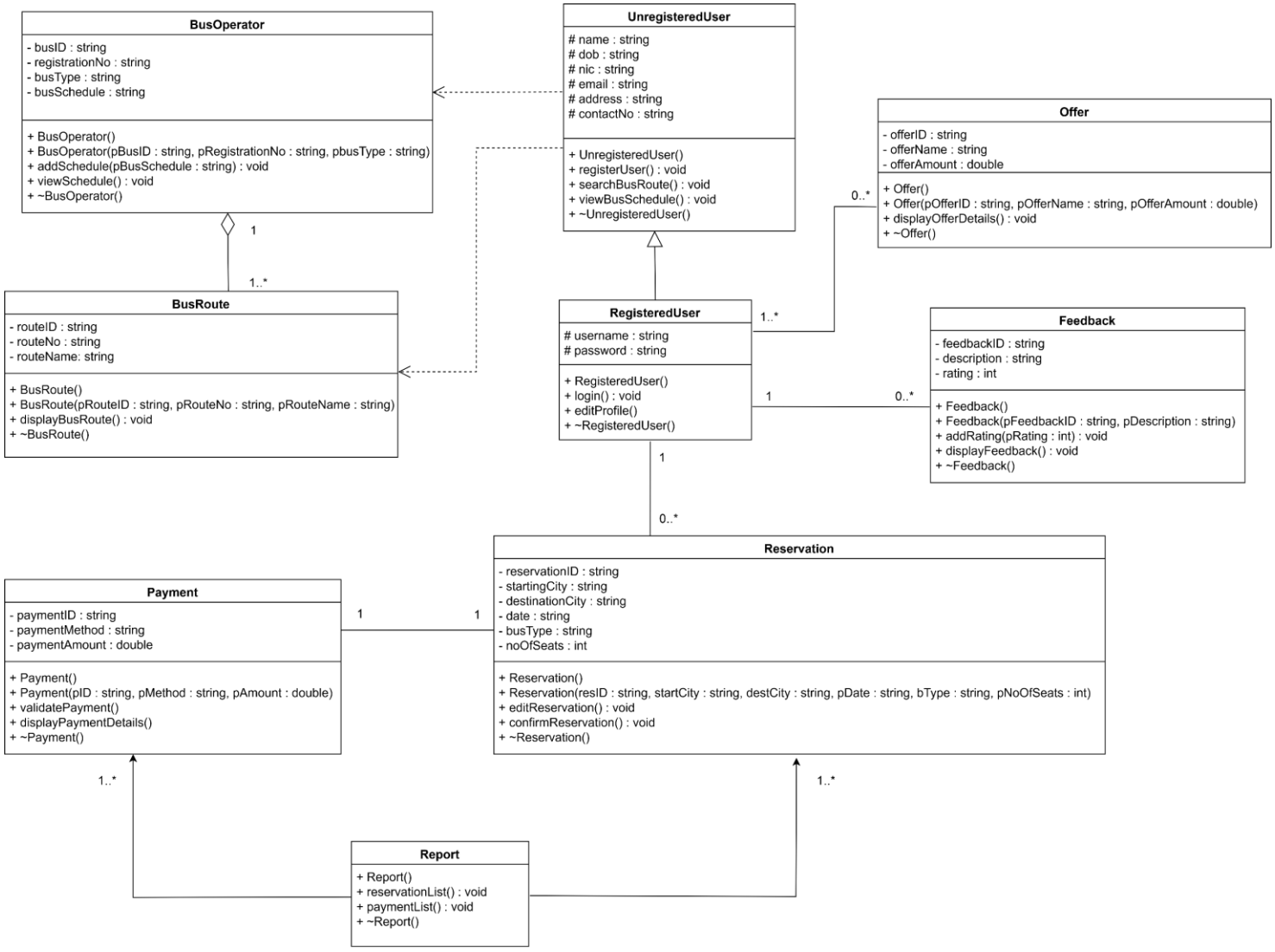
Class Name: Reservation	
Responsibility	**Collaborators**
Make reservation	Registered User
Confirm reservation	Payment
Edit reservation	Registered User

Class Name: Payment	
Responsibility	**Collaborators**
Store payment details	
Validate	
Generate invoice	Reservation

Class Name: Feedback	
Responsibility	**Collaborators**
Rate service	Registered User
Add feedback	Registered User
View feedback	

Class Name: Report	
Responsibility	**Collaborators**
List of Reservations	Reservation
List of Payments	Payment

Exercise 1


BusOperator
- busID : string
- registrationNo : string
- busType : string
- busSchedule : string
+ BusOperator()
+ BusOperator(pBusID : string, pRegistrationNo : string, pbusType : string)
+ addSchedule(pBusSchedule : string) : void
+ viewSchedule() : void
+ ~BusOperator()
UnregisteredUser
name : string
dob : string
nic : string
email : string
address : string
contactNo : string
+ UnregisteredUser()
+ registerUser() : void
+ searchBusRoute() : void
+ viewBusSchedule() : void
+ ~UnregisteredUser()
Offer
- offerID : string
- offerName : string
- offerAmount : double
+ Offer()
+ Offer(pOfferID : string, pOfferName : string, pOfferAmount : double)
+ displayOfferDetails() : void
+ ~Offer()
BusRoute
- routeID : string
- routeNo : string
- routeName: string
+ BusRoute()
+ BusRoute(pRouteID : string, pRouteNo : string, pRouteName : string)
+ displayBusRoute() : void
+ ~BusRoute()
RegisteredUser
username : string
password : string
+ RegisteredUser()
+ login() : void
+ editProfile()
+ ~RegisteredUser()
Feedback
- feedbackID : string
- description : string
- rating : int
+ Feedback()
+ Feedback(pFeedbackID : string, pDescription : string)
+ addRating(pRating : int) : void
+ displayFeedback() : void
+ ~Feedback()
Payment
- paymentID : string
- paymentMethod : string
- paymentAmount : double
+ Payment()
+ Payment(pID : string, pMethod : string, pAmount : double)
+ validatePayment()
+ displayPaymentDetails()
+ ~Payment()
Reservation
- reservationID : string
- startingCity : string
- destinationCity : string
- date : string
- busType : string
- noOfSeats : int
+ Reservation()
+ Reservation(resID : string, startCity : string, destCity : string, pDate : string, bType : string, pNoOfSeats : int)
+ editReservation() : void
+ confirmReservation() : void
+ ~Reservation()
Report
+ Report()
+ reservationList() : void
+ paymentList() : void
+ ~Report()
1
1..*
1..*
0..*
1..*
1
0..*
1
0..*
1
1
1..*
1..*

Contribution

	Student ID	Student Name	Individual Contribution
1	IT21267222	EDIRISINGHE E.A.K.G.	CRC & Class diagram: • UnregisteredUser • RegisteredUser Code: • UnregisteredUser.h • UnregisteredUser.cpp • RegisteredUser.h • RegisteredUser.cpp • main.cpp
2	IT21251450	A.L.V.J. Atapattu	CRC & Class diagram: • Payment Code: • Payment.h • Payment.cpp • main.cpp
3	IT21204784	W.D.O.D Wijerupa	CRC & Class diagram: • Offer Code: • Offer.h • Offer.cpp • main.cpp
4	IT21267086	Hasara L.A.N	CRC & Class diagram: • Reservation • Feedback

			Code: • Reservation.h • Reservation.cpp • Feedback.h • Feedback.cpp • main.cpp
5	IT21267772	Kumanayake I. A.	CRC & Class diagram: • BusRoute • BusOperator • Report Code: • BusRoute.h • BusRoute.cpp • BusOperator.h • BusOperator.cpp • Report.h • Report.cpp • main.cpp

Exercise 2

UnregisteredUser.h

```
#pragma once
#include "BusRoute.h"
#include "BusOperator.h"

class UnregisteredUser
{
protected:
    char name[10];
    char dob[10];
    char nic[12];
    char email[20];
    char address[40];
    char contactNo[10];

public:
    UnregisteredUser();
    void registerUser(const char pName[], const char pDob[], const char pNic[],
const char pEmail[], const char paddress[], const char pContactNo[]);
    void searchBusRoute(BusRoute *br);
    void viewBusSchedule(BusOperator *bo);
    ~UnregisteredUser();
};
```

UnregisteredUser.cpp

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

UnregisteredUser::UnregisteredUser()
{
}

void UnregisteredUser::registerUser(const char pName[], const char pDob[], const
char pNic[], const char pEmail[], const char paddress[], const char pContactNo[])
{
    strcpy(name, pName);
    strcpy(dob, pDob);
    strcpy(nic, pNic);
    strcpy(email, pEmail);
    strcpy(address, paddress);
    strcpy(contactNo, pContactNo);
}

void UnregisteredUser::searchBusRoute(BusRoute *br)
{
    br->displayBusRoute();
}

void UnregisteredUser::viewBusSchedule(BusOperator *bo)
{
    bo->viewSchedule();
}

UnregisteredUser::~UnregisteredUser()
{
}
```

RegisteredUser.h

```
#pragma once
#include "UnregisteredUser.h"
#include "Reservation.h"
#include "Offer.h"
#include "Feedback.h"
#define SIZE1 5
#define SIZE2 5
#define SIZE3 5

class RegisteredUser : public UnregisteredUser
{
protected:
    char username[20];
    char password[20];

    //class relationship
    Reservation *rsv[SIZE1];
    Offer *ofr[SIZE2];
    Feedback *fbk[SIZE3];

public:
    RegisteredUser();
    RegisteredUser(const char pUsername[], const char pPassword[]);
    void login();
    void editProfile();
    void addReservation(Reservation *prsv[]);
    void viewOffer(Offer *pofr[]);
    void addFeedback(Feedback *pfbk[]);
    ~RegisteredUser();
};
```

RegisteredUser.cpp

```cpp
#include "RegisteredUser.h"
#include <iostream>
#include <cstring>
using namespace std;

RegisteredUser::RegisteredUser()
{
}

RegisteredUser::RegisteredUser(const char pUsername[], const char pPassword[])
{
    strcpy(username, pUsername);
    strcpy(password, pPassword);
}

void RegisteredUser::login()
{
}

void RegisteredUser::editProfile()
{
}

void RegisteredUser::addReservation(Reservation *prsv[])
{
    for (int i = 0; i < SIZE1; i++)
    {
```

```
        rsv[i] = prsv[i];
    }
}

void RegisteredUser::viewOffer(Offer *pofr[])
{
    for (int i = 0; i < SIZE2; i++)
    {
        ofr[i] = pofr[i];
    }
}

void RegisteredUser::addFeedback(Feedback *pfbk[])
{
    for (int i = 0; i < SIZE3; i++)
    {
        fbk[i] = pfbk[i];
    }
}

RegisteredUser::~RegisteredUser()
{
    cout << "Deleting registered user " << username << endl;
}
```

BusRoute.h

```
#pragma once

class BusRoute
{
private:
    char routeID[6];
    char routeNo[10];
    char routeName[30];

public:
    BusRoute();
    BusRoute(const char pRouteID[], const char pRouteNo[], const char
pRouteName[]);
    void displayBusRoute();
    ~BusRoute();
};
```

BusRoute.cpp

```
#include "BusRoute.h"
#include <iostream>
#include <cstring>
using namespace std;

BusRoute::BusRoute()
{
}

BusRoute::BusRoute(const char pRouteID[], const char pRouteNo[], const char
pRouteName[])
{
    strcpy(routeID, pRouteID);
    strcpy(routeNo, pRouteNo);
    strcpy(routeName, pRouteName);
}

void BusRoute::displayBusRoute()
{
    cout << "Route ID : " << routeID << endl;
    cout << "Route No : " << routeNo << endl;
    cout << "Route Name : " << routeName << endl;
}

BusRoute::~BusRoute()
{
    cout << "Deleting bus route " << routeID << endl;
}
```

BusOperator.h

```
#pragma once
#include "BusRoute.h"
#define SIZE 5

class BusOperator
{
private:
    char busID[5];
    char registrationNo[8];
    char busType[20];
    char busSchedule[50];

    //class relationship
    BusRoute *br[SIZE];

public:
    BusOperator();
    BusOperator(const char pBusID[], const char pRegistrationNo[], const char
pbusType[]);
    void addSchedule(const char pBusSchedule[]);
    void viewSchedule();
    void addBusRoute(BusRoute *br[]);
    ~BusOperator();
};
```

BusOperator.cpp

```
#include "BusOperator.h"
#include <iostream>
#include <cstring>
using namespace std;

BusOperator::BusOperator()
{
}

BusOperator::BusOperator(const char pBusID[], const char pRegistrationNo[], const
char pbusType[])
{
    strcpy(busID, pBusID);
    strcpy(registrationNo, pRegistrationNo);
    strcpy(busType, pbusType);
}

void BusOperator::addSchedule(const char pBusSchedule[])
{
    strcpy(busSchedule, pBusSchedule);
}

void BusOperator::viewSchedule()
{
}

void BusOperator::addBusRoute(BusRoute *pbr[])
{
```

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

BusOperator::~BusOperator()
{
    cout << "Deleting bus operator " << busID << endl;
}
```

Offer.h

```
#pragma once
#include "RegisteredUser.h"

class Offer{
  private:
  char offerID[5];
  char offerName[20];
  double offerAmount;
  RegisteredUser *reguser;

  public:
  Offer();
  Offer(const char pOfferID[], const char pOfferName[], double pOfferAmount,
RegisteredUser *pregister);
  void displayOfferDetails();
  ~Offer();
};
```

Offer.cpp

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

using namespace std;

    Offer::Offer(){
        strcpy(offerID , "");
        strcpy(offerName , "");
        offerAmount = 0;
    }

    Offer::Offer(const char pOfferID[], const char pOfferName[], double
pOfferAmount, RegisteredUser *pregister){
        strcpy(offerID , pOfferID);
        strcpy(offerName , pOfferName);
        offerAmount = pOfferAmount;
        reguser = pregister;
    }

    void Offer::displayOfferDetails(){
        cout << "Offer ID : " << offerID << endl;
        cout << "Offer Name : " << offerName <<endl;
    }

    Offer::~Offer(){
         cout << "Deleting offer " << offerID << endl;
     }
```

Reservation.h

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

class Reservation{
    private:
    char reservationID[7];
    char startingCity[20];
    char destinationCity[20];
    char date[10];
    char busType[20];
    int noOfSeats;
    RegisteredUser *reguser;
    Payment *payment;

    public:
    Reservation();
    Reservation(const char preservationID[],const char pstartingCity[],const
char pdestinationCity[],const char pdate[],const char pbusType[],int pnoOfSeats);
    void addReservation(Payment *ppay , RegisteredUser *preg);
    void editReservation();
    void confirmReservation();
    ~Reservation();
};
```

Reservation.cpp

```
#include "Reservation.h"
#include <iostream>
#include <cstring>
using namespace std;

Reservation::Reservation(){
        strcpy(reservationID,"");
        strcpy(startingCity,"");
        strcpy(destinationCity,"");
        strcpy(date,"");
        strcpy(busType,"");
        noOfSeats = 0 ;
    }
Reservation ::Reservation(const char preservationID[],const char
pstartingCity[],const char pdestinationCity[],const char pdate[],const char
pbusType[],int pnoOfSeats){
        strcpy(reservationID,preservationID);
        strcpy(startingCity,pstartingCity);
        strcpy(destinationCity,pdestinationCity);
        strcpy(date,pdate);
        strcpy(busType,pbusType);
        noOfSeats = pnoOfSeats ;
    }
void Reservation ::addReservation(Payment *ppay , RegisteredUser *preg){

}
void Reservation :: editReservation(){

}
void Reservation ::confirmReservation(){
```

```
}
Reservation ::~Reservation(){
        cout<<"";
    }
```

Payment.h

```
#pragma once
#include "Reservation.h"
class Payment
{
private:
    char paymentID[6];
    char paymentMethod[10];
    double paymentAmount;

  //class relationship
  Reservation *res;

public:
    Payment();
    Payment(const char pID[], const char pMethod[], double pAmount, Reservation
*pres);
    void validatePayment();
    void displayPaymentDetails();
    ~Payment();
};
```

Payment.cpp

```
#include "Payment.h"
#include <iostream>
#include <cstring>
using namespace std;

Payment::Payment()
{
    strcpy(paymentID, "");
    strcpy(paymentMethod, "");
    paymentAmount = 0;
};

Payment::Payment(const char pID[], const char pMethod[], double pAmount,
Reservation *pres)
{
    strcpy(paymentID, pID);
    strcpy(paymentMethod, pMethod);
    paymentAmount = pAmount;
  res = pres;
};

void Payment::validatePayment()
{

};

void Payment::displayPaymentDetails()
{
    cout << "Payment ID is :" << paymentID << endl;
    cout << "Payment Method is :" << paymentMethod << endl;
```

```
    cout << "Payment Amount :" << paymentAmount << endl;
}

Payment::~Payment()
{
    cout << "!PR" << endl;
}
```

Feedback.h

```
#pragma once
#include "RegisteredUser.h"

class Feedback{
    private:
    char feedbackID[6];
    char description[20];
    int rating;
    RegisteredUser *user;

    public:
    Feedback();
    Feedback(const char pfeedbackID[] ,const char pdescription[], RegisteredUser
*pregister);
    void addRating(int pRating);
    void displayFeedback();
    ~Feedback();
};
```

Feedback.cpp

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

using namespace std;

  Feedback::Feedback(){
        strcpy(feedbackID , "");
        strcpy(description , "");
    }
  Feedback::Feedback(const char pfeedbackID[] ,const char pdescription[] ,
RegisteredUser *pregister){
        strcpy(feedbackID , pfeedbackID);
        strcpy(description , pdescription);
        user = pregister;
    }
    void Feedback:: addRating(int pRating){

    }
    void Feedback:: displayFeedback(){

    }
   Feedback ::~Feedback(){
        cout<<"";
    }
```

Report.h

```
#pragma once
#include "Payment.h"
#include "Reservation.h"
#define SIZE1 5
#define SIZE2 5

class Report
{
private:
    Payment *pmt[SIZE1];
    Reservation *res[SIZE2];

public:
    Report(Payment *p[], Reservation *r[]);
    void reservationList();
    void paymentList();
    ~Report();
};
```

Report.cpp

```cpp
#include "Report.h"

Report::Report(Payment *p[], Reservation *r[])
{
    for (int i = 0; i < SIZE1; i++)
    {
        pmt[i] = p[i];
    }

    for (int i = 0; i < SIZE2; i++)
    {
        res[i] = r[i];
    }
}

void Report::reservationList()
{
}

void Report::paymentList()
{
}

Report::~Report()
{
}
```

main.cpp

```
#include <iostream>
#include "UnregisteredUser.h"
#include "RegisteredUser.h"
#include "BusRoute.h"
#include "BusOperator.h"
#include "Offer.h"
#include "Reservation.h"
#include "Payment.h"
#include "Feedback.h"
#include "Report.h"

int main() {
    UnregisteredUser *unregUser;

    unregUser->registerUser("Kavindu", "02-04-2000",
"200008456789","kavindugithmin@gmail.com","No.22/1,Parakrama
Mawatha,Maradana","0769645678");

    RegisteredUser *regUser = new RegisteredUser("kavinduEdirisinghe",
"jimmy21");

    BusRoute *busroute[2];

    busroute[0] = new BusRoute("BR001", "01", "Colombo - Kandy");
    busroute[1] = new BusRoute("BR002", "17", "Panadura - Kandy");

    BusOperator *busop = new BusOperator("B001", "ND-9021", "Semi Luxury");

    busop->addBusRoute(busroute);
```

```
    regUser->viewBusSchedule(busop);

    Offer *offer = new Offer("O001", "New Year", 2000, regUser);

    offer->displayOfferDetails();

    Reservation *reservation[2];

    reservation[0] = new Reservation("R00001","Colombo","Kandy","2022-05-
25","Luxuary",2);
    reservation[1] = new Reservation("R00002","Matara","Kandy","2022-05-
27","Luxuary",1);

    reservation[0]->confirmReservation();
    reservation[1]->confirmReservation();

    Payment *payment[2];

    payment[0]= new Payment("PR0001","VISA",3000,reservation[0]);
    payment[1]= new Payment("PR0002","AMEX",4000,reservation[1]);

    payment[0]->displayPaymentDetails();
    payment[1]->displayPaymentDetails();

    Feedback *feed = new Feedback("FB0023","Great service",regUser);
    feed->displayFeedback();

    Report *report = new Report(payment, reservation);

    delete unregUser;
    delete regUser;

    delete busop;
```

```
    delete busroute[0];
    delete busroute[1];

    delete offer;

    delete reservation[0];
    delete reservation[1];

    delete feed;

    delete payment[0];
    delete payment[1];

    delete report;

    return 0;
}
```