Topic : BUS SCHEDULING AND BOOKING SYSTEM

Group no : MLB_WE_01.01_02

Campus : Malabe

Submission Date: 19/05/2022

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**
IT21312762	Lakshitha W.H.M. U	0711911158
IT21306518	Jayasinghe J.I. B	0771893910
IT21291746	Madushanka J.G. N	0764144195
IT21344510	Sahnas M.J. F	0704118049
-	-	-

BUS SCHEDULING AND BOOKING SYSTEM

REQUIRMENTS

- Guests can use the URL to access the online bus booking and scheduling system and browse the we bsite.
- By supplying the essential basic information such as name, address, mobile number and email address, guests can register as passengers in the system.
- By supplying the essential basic information such as name, address, mobile number and email address, license expire date, guest can also register as a bus driver to the system.
- Then as a registered bus driver, driver can also register their bus details such as bus number, bus type, registered year in the system.
- After registering as a passenger, the guest can change his or her account information such as address, email, mobile number.
- Passengers can look up bus availability, bus routes, and bus schedules.
- Following the search, the passenger can book the bus ticket that he or she requires.
- The passenger is then taken to the payment page, which displays the complete payment information such as card number, cvv number, date of expire, card holder name and payment amount.
- To complete the transaction, the passenger can choose his or her chosen payment method like debit card payment, and credit card payment.
- The financial manager validates the payment and generate reports.
- Then complete and confirm the reservation and obtain an E-Ticket.
- If passenger need to cancel or reschedule ticket, he/she can request to cancellation or reschedule.
- System analyzer analysis the payment details and sends it to the system admin.
- System analyzer analyze the passenger feedback.
- System admin check and generate reports such as payment report, monthly income report, and passenger feedback report.
- System administrator can manage bus drivers and bus details with adding and removing relevant details such as driver name, contact number, address, bus number.
- System administrator can update the system with adding and removing bus times and bus routes.
- After the trip passenger can give feedback about the system.

ANALYSE NOUN

- Guests can use the URL to access the online bus booking and scheduling system and browse the we bsite.
- By supplying the essential basic information such as name, address, mobile number and email address, guests can register as passengers in the system.
- By supplying the essential basic information such as name, address, mobile number and email address, license expire date, guest can also register as a bus driver to the system.
- Then as a registered bus driver, driver can also register their bus details such as bus number, bus type, registered year in the system.
- After registering as a passenger, the guest can change his or her account information such as address, email, mobile number.
- Passengers can look up bus availability, bus routes, and bus schedules.
- Following the search, the passenger can book the bus ticket that he or she requires.
- The passenger is then taken to the payment page, which displays the complete payment information such as card number, cvv number, date of expire, card holder name and payment amount.
- To complete the transaction, the passenger can choose his or her chosen payment method like debit card payment, and credit card payment.
- The financial manager validates the payment and generate reports.
- Then complete and confirm the reservation and obtain an E-Ticket.
- If passenger need to cancel or reschedule ticket, he/she can request to cancellation or reschedule.
- System analyzer analysis the payment details and sends it to the system admin.
- System analyzer analyze the passenger feedback.
- System admin check and generate reports such as payment report, monthly income report, and passenger feedback report.
- System administrator can manage bus drivers and bus details with adding and removing relevant details such as driver name, contact number, address, bus number.
- System administrator can update the system with adding and removing bus times and bus routes.
- After the trip passenger can give feedback about the system.

IDENTIFY NOUN

1. Guests
2. online bus booking and scheduling system
3. name, address, mobile number and email address
4. passengers
5. name, address, mobile number and email address, license expire date
6. bus driver
7. Bus
8. driver - Redundant
9. bus number, bus type, registered year
10. bus routes
11. bus schedules
12. bus ticket
13. payment
14. card number, cv number, date of expire, card holder name and payment amount
15. cash payment, debit card payment, credit card payment and QR payment
16. financial manager
17. Ticket
18. system admin
19. System analyzer
20. Report
21. payment report, monthly income report, passenger feedback report and review report
22. System administrator
23. bus times
24. feedback

NOUN VERB ANALYSIS FOR IDENTIFY CLASSES

1. Guests -Class
2. online bus booking and scheduling system – Outside scope of system
3. name, address, mobile number and email address -attributes
4. passengers- Class
5. name, address, mobile number and email address, license expire date – attributes
6. bus driver – Class
7. Bus - Class
8. driver - Redundant
9. bus number, bus type, registered year - attributes
10. bus routes - attributes
11. bus schedules – Attributes
12. bus ticket - Class
13. payment - Class
14. card number, cv number, date of expire, card holder name and payment amount -Attributes
15. cash payment, debit card payment, credit card payment and QR payment -Attributes
16. financial manager - Class
17. Ticket - Redundant
18. system admin- Redundant
19. System analyzer – Outside scope
20. Report – Boundary class
21. payment report, monthly income report, passenger feedback report and review report - Attributes
22. System administrator – Class
23. bus times - attributes
24. feedback – An event or an operation

CLASSES FOR THE SYSTEM

Guests

Passengers

Bus

Bus driver

Financial manager

Payment

Report (Boundary class)

Bus ticket

Administrator

CRC CARD FOR CLASSES

Guest	
Responsibilities	Collaborators
Register to the online bus booking and scheduling system	

Passenger	
Responsibilities	Collaborators
Book and schedule a bus ticket	
Change the account	

Bus Driver	
Responsibilities	Collaborators
Login to the online bus booking and scheduling system	
Register the bus details	Bus

Payment	
Responsibilities	Collaborators
Add payment details	Passenger

Financial Manager	
Responsibilities	Collaborators
Validate payment	Payment
Check payment report	Report

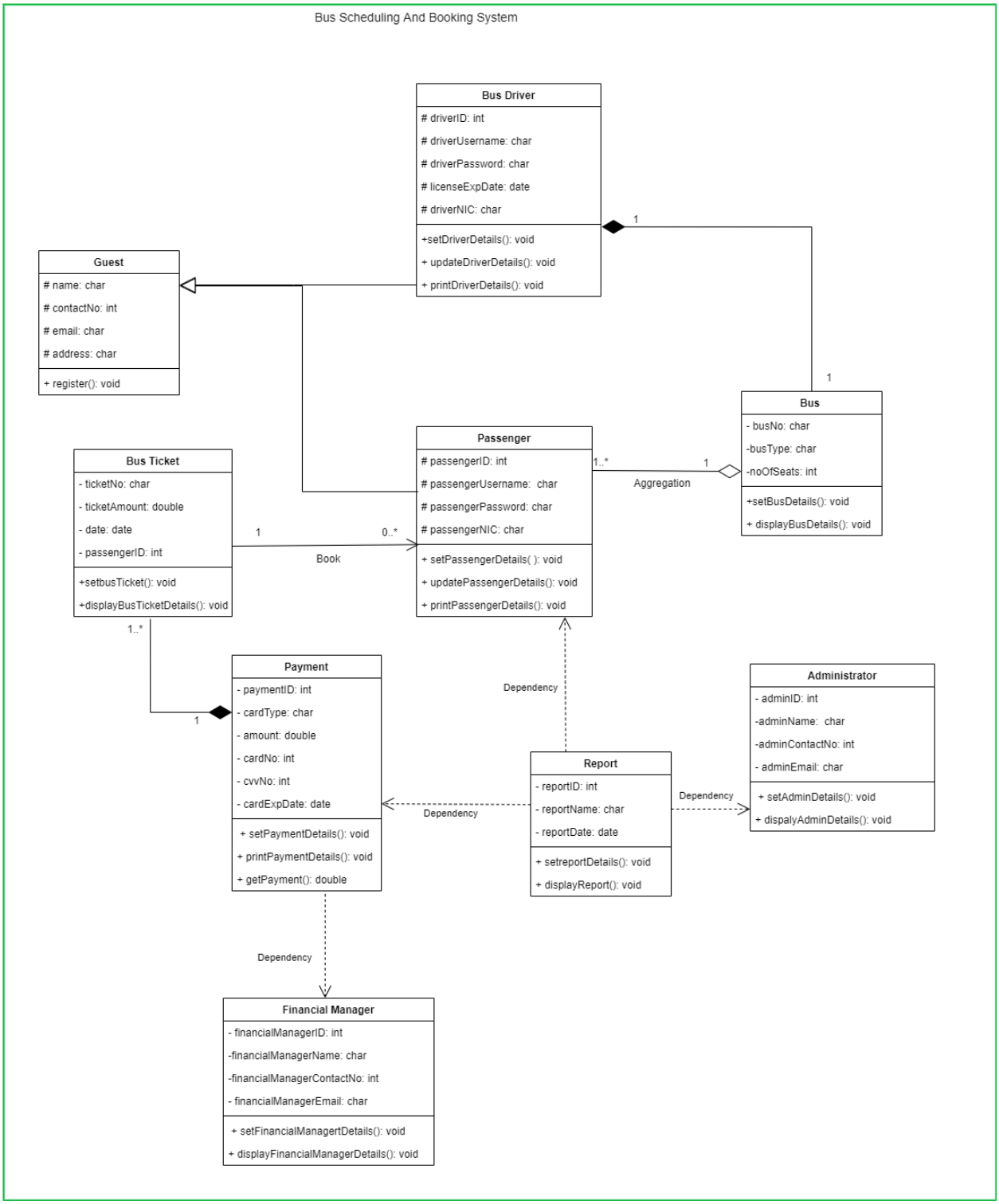
Report	
Responsibilities	Collaborators
List the payment details	
List the passenger feedback	administrator
Generate monthly income	

Bus Ticket	
Responsibilities	Collaborators
Reserve bus ticket	
Reschedule bus ticket	
Cancel bus ticket	

Administrator	
Responsibilities	Collaborators
Update bus details	
Manage passenger details	Passenger
Manage bus driver details	Bus driver

Bus	
Responsibilities	Collaborators
Add bus	
Remove bus	

CLASS DIAGRAM FOR BUS SCHEDULING AND BOOKING SYSTEM



CODE FOR BUS SCHEDULING AND BOOKING SYSTEM

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

class Guest{
protected:
char name[30];
char email[30];
long int  contactNo;
char address[30];
public:
  Guest();
  Guest(char gName[], char gEmail[], long int gContactNo, char gAddress[] );
  void Register();
};
class Passenger : public Guest {
protected:
 int passengerID;
 char passengerUsername[15];
 char passengerPassword[10];
 char passengerNIC[12];
public:
 Passenger();
 Passenger(int pID,char pName[], char pEmail[],  long int pContactNo, char pAddress[], char
pUsername[], char pPassword[],  char pNIC[]);
 void setPassengerDetails();
 void printPassengerdetails();
```

```
 void updatePassengerDetails();
 ~ Passenger();
};
class BusDriver: public Guest {
protected:
 int driverID;
 char driverUsername[15];
 char driverPassword[10];
 char licenseExpDate[10];
 char driverNIC[12];
public:
  BusDriver();
  BusDriver(int bID, char bName[], char bEmail[], long int bContactNo, char bAddress[], char
bUsername[], char bPassword[],  char bNIC[], char licenseDate[]);
  void setBusDriverDetails();
  void printBusDriverDetails();
  void updateBusDriverDetails();
  ~ BusDriver();
};
//Busticket class
class Busticket{
  private:
    char ticketNo[20];
    double ticketAmount;
    int passengerID;
    Passenger *traveller1; //uni directional assosiations.
  public:
    Busticket();
    Busticket(char ticketNum[],double tPrice,int pID);
```

```
    void setBusTicketDetails();
    ~Busticket();
};
//paymet class
class Payment{
  private:
    int paymentID;
    char cardType[70];
    double amount;
    int cardNo;
    int cvvNo;
    char cardExpDate[20];
    Busticket *busticket;   //Composition Relationship
  public:
    Payment();
    Payment(int paymentID,char cType[],double total,int cardNo,int cvvNo,char cardExp[]);
    void setPaymentDetails();
    void printPaymentDetails();
    ~Payment();
};
class Administrator{
            private:
                        int adminID;
                        char adminName;
                        int adminContactNo;
                        char adminEmail;

            public:
        Administrator();
```

```
                Administrator(int aID,char aName,int aContactNo , char aEmail);
                void setadminDetails(int aID,char aName,int aContactNo, char aEmail);
                void displayadminDetails();
                ~Administrator ();
};
class Report {
  private:
    int reportID;
    char reportName[20];
    char reportDate[10];
  public:
        Report();
        Report(int rID,char rName,char rDate);
    setreportDetails();
    displayreport(Administrator*r);
        ~Report();
};
//class finance Manager
class FinanceManager : public Payment{
        protected:
                int financialManagerID;
                char financialManagerName[30];
                int financialManagerContactNo;
                char financialManagerEmail[50];
        public:
                void financeManager();
                void financeManager(int fID,char fName[],int fContactNo,char fEmail[]);
                void setFinanceManagerDetails();
                void displayFinanceManagerDetails();
```

```
                ~FinanceManager();
};
//class Bus
class Bus:public BusDriver{
        protected:
                char busNo[10];
                char busType[20];
                int noOfSeats;
        public:
                void BusDetails();
                void BusDetails(char pbusNo[],char pbusType[],int pnoOfSeats);
                void setBusDetails();
                void displayBusDetails();
                void updateBusDetails();
                ~Bus();
};
//defualt constructor for guest class
Guest::Guest(){
   strcpy(name, "");
   strcpy(email, "");
   contactNo = 0;
   strcpy(address, "");
}
//Constructors with paramenters for guest class
Guest::Guest(char gName[], char gEmail[], long int gContactNo, char gAddress[]){
        strcpy(name, gName);
   strcpy(email, gEmail);
   contactNo = gContactNo;
   strcpy(address, gAddress);
```

```
}
//defualt constructor for passenger class
Passenger::Passenger(){
        passengerID = 0;
   strcpy (passengerUsername, "");
   strcpy (passengerPassword, "");
   strcpy (passengerNIC, "");

}
//Constructors with paramenters for Passenger class
Passenger::Passenger(int pID,char pName[], char pEmail[], long int pContactNo, char pAddress[], char pUsername[], char pPassword[],  char pNIC[]){
        passengerID = pID;
        strcpy(name, pName);
   strcpy(email, pEmail);
   contactNo = pContactNo;
   strcpy(address, pAddress);
   strcpy (passengerUsername, pUsername);
   strcpy (passengerPassword, pPassword);
   strcpy (passengerNIC, pNIC);
}
void Passenger::setPassengerDetails(){
        cout<<"Passenger ID:";
        cin>> passengerID;
        cout<<"Passenger Username:";
        cin>> passengerUsername;
        cout<<"Passenger Password:";
        cin>> passengerPassword;
```

```
        cout<<"Passenger NIC:";
        cin>> passengerNIC;
}
void Passenger::printPassengerdetails(){
        cout << "Name of the customer is : "<< name << endl;
   cout << "ID of the customer is : "<< passengerID << endl;
        cout << "Email of the customer is : "<< email << endl;
        cout << "Password of the customer is : "<< passengerPassword << endl;
        cout << "Address of the customer is : "<< address << endl;
        cout << "Phone of the customer is : "<< contactNo << endl;
        cout << "NIC of the customer is : "<< passengerNIC<< endl;
        cout << "Username of the customer is : "<< passengerUsername << endl;
}
void Passenger::updatePassengerDetails(){
        cout<<"Passenger new ID:";
        cin>> passengerID;
        cout<<"Passenger new Username:";
        cin>> passengerUsername;
        cout<<"Passenger mew Password:";
        cin>> passengerPassword;
        cout<<"Passenger new NIC:";
        cin>> passengerNIC;

}
//Destructor for passenger class
Passenger::~Passenger()
{
        cout<<"Destructor runs"<<endl;
}
```

```
//defualt constructor for BusDriver class
BusDriver::BusDriver()
{
        driverID = 0;
   strcpy (driverUsername, "");
   strcpy (driverPassword, "");
   strcpy(licenseExpDate, "");
   strcpy (driverNIC, "");

}
//Constructors with paramenters for BusDriver class
BusDriver::BusDriver(int bID, char bName[], char bEmail[], long int bContactNo, char bAddress[], char bUsername[], char bPassword[],  char bNIC[], char licenseDate[])
{
        driverID = bID;
        strcpy(name, bName);
   strcpy(email, bEmail);
   contactNo = bContactNo;
   strcpy(address, bAddress);
   strcpy (driverUsername, bUsername);
   strcpy (driverPassword, bPassword);
   strcpy(licenseExpDate, licenseDate);
   strcpy (driverNIC, bNIC);
}
void BusDriver::setBusDriverDetails()
{
        cout<<"Passenger ID:";
        cin>> driverID;
        cout<<"Passenger Username:";
```

```
        cin>> driverUsername;
        cout<<"Passenger Password:";
        cin>> driverPassword;
        cout<<"Passenger NIC:";
        cin>> driverNIC;
        cout<<"License expire date:";
        cin>>licenseExpDate;
}
void BusDriver::printBusDriverDetails(){
        cout << "Name of the customer is : "<< name << endl;
        cout << "ID of the customer is : "<< driverID << endl;
   cout << "Email of the customer is : "<< email << endl;
        cout << "Password of the customer is : "<< driverPassword << endl;
        cout << "Address of the customer is : "<< address << endl;
        cout << "Phone of the customer is : "<< contactNo << endl;
        cout << "NIC of the customer is : "<< driverNIC<< endl;
        cout << "Username of the customer is : "<< driverUsername << endl;
}
 void BusDriver::updateBusDriverDetails(){
        cout<<"Passenger new ID:";
        cin>> driverID;
        cout<<"Passenger new Username:";
        cin>> driverUsername;
        cout<<"Passenger mew Password:";
        cin>> driverPassword;
        cout<<"Passenger new NIC:";
        cin>> driverNIC;
        cout<<"New License expire date:";
        cin>>licenseExpDate;
```

```
}
//destructor for BusDriver class
BusDriver::~ BusDriver(){
        cout<<"destructor tuns"<<endl;
}
//implementation of payment constructor
Payment::Payment(){
 paymentID=0;
 strcpy(cardType,"");
 amount=0.00;
 cardNo=0;
 cvvNo=0;
 strcpy(cardExpDate,"");
}
Payment::Payment(int paymentID,char cType[],double total,int cardNum,int cvvNo,char cardExp[]){
 paymentID=paymentID;
 strcpy(cardType,cType);
 amount=total;
 cardNo=cardNum;
 strcpy(cardExpDate,cardExp);
}
//destructor of payment
Payment::~Payment(){

}
//implemetation of busticket constructor
Busticket::Busticket(){
        strcpy(ticketNo,"");
```

```
   ticketAmount=0.00;
   passengerID=0;
}
Busticket::Busticket(char ticketNum[],double tPrice,int pID){
        strcpy(ticketNo,ticketNum);
   ticketAmount=tPrice;
   passengerID=pID;
}
//destructor of busticket
Busticket::~Busticket(){

}
Administrator::Administrator(){

}
Report::Report(){

}
void Administrator::setadminDetails(int aID,char aName,int aContactNo,char aEmail){
        adminID = aID;
        adminName = aName;
        adminContactNo = aContactNo;
        adminEmail = aEmail;
}
void Administrator::displayadminDetails()
{
        cout<< "admin ID= "<<adminID<<endl;
        cout   <<"admin name="<<adminName<<endl;
         cout  <<"admin contact No="<<adminContactNo<<endl;
```

```
        cout   <<"admin Email="<<adminEmail<<endl;
}
Report::setreportDetails(){

}
Report::displayreport(Administrator*r){

}
//implementing default construtor for BusDetails
void Bus::BusDetails(){
        strcpy(busNo,"");
        strcpy(busType,"");
        noOfSeats=0;
}
//implementing parameter construtor for BusDetails
void Bus::BusDetails(char pbusNo[],char pbusType[],int pnoOfSeats){
        strcpy(busNo,pbusNo);
        strcpy(busType,pbusType);
        noOfSeats=pnoOfSeats;
}
//implementing default constructor for financialManager
void FinanceManager::financeManager(){
        financialManagerID=0;
        strcpy(financialManagerName,"");
        financialManagerContactNo=0;
        strcpy(financialManagerEmail,"");
}
//implementing parameter constructor for financialManager
void FinanceManager::financeManager(int fID,char fName[],int fContactNo,char fEmail[]){
```

```
        financialManagerID=fID;
        strcpy(financialManagerName,fName);
        financialManagerContactNo=fContactNo;
        strcpy(financialManagerEmail,fEmail);
}
void Bus::setBusDetails(){
        cout<<"Bus No:";
        cin>>busNo;
        cout<<"Bus Type:";
        cin>>busType;
        cout<<"No of Seats:";
        cin>>noOfSeats;
}
void Bus::displayBusDetails(){
        cout<<"Bus Number:"<<busNo<<endl;
        cout<<"Bus Type:"<<busType<<endl;
        cout<<"Number of Seats:"<<noOfSeats<<endl;
}
void Bus::updateBusDetails(){
        cout<<"New Bus Number:";
        cin>>busNo;
        cout<<"New Bus Type:";
        cin>>busType;
        cout<<"New Number of Seats:";
        cin>>noOfSeats;
}
//implementing destructor for Bus
Bus::~Bus(){
        cout<<"Destructor runs in the Program"<<endl;
```

```
}
void FinanceManager::setFinanceManagerDetails(){
	cout<<"Finance Manager ID:";
	cin>>financialManagerID;
	cout<<"Finance Manager Name:";
	cin>>financialManagerName;
	cout<<"Finance Mnager Contact Number:";
	cin>>financialManagerContactNo;
	cout<<"Finance Mnager E-mail";
	cin>>financialManagerEmail;
}
void FinanceManager::displayFinanceManagerDetails(){
	cout<<"Finance Mnager ID:"<<financialManagerID<<endl;
	cout<<"Finance Manager Name:"<<financialManagerName<<endl;
	cout<<"Finance Manager Contact Number:"<<financialManagerContactNo<<endl;
	cout<<"Finance Manager Email:"<<financialManagerEmail<<endl;
}
//main programm
int main(){
	Payment *p1 = new Payment();
	Busticket *Bt1 = new Busticket();
	Administrator *a1 = new Administrator();
	Report *r1 = new Report();
	Bus*b1=new Bus();
	FinanceManager *f1=new FinanceManager;
	Guest *g1 = new Guest(); //creating object for Guest class
	//creating object for passenger class
	Passenger *traveller1= new Passenger(1,"Nuwan Jayathilake","nuwan@gmail.com",
94771,"Malabe","NuwanJ0001","nuwanJ@2022", "199676232574" );
```

```
            cout <<"--- Display Passengers Details --" << endl;
    traveller1->printPassengerdetails();
    cout << endl << endl;
    cout <<"--- Input Passengers Details --" << endl;
                traveller1->setPassengerDetails();
                cout << endl << endl;
    cout <<"--- Update Passengers Details --" << endl;
    traveller1->updatePassengerDetails();
    cout << endl << endl;



            //creating object for BusDriver class
            BusDriver *driver1= new BusDriver(1,"Ruwan Senarathne","ruwans@gmail.com",
94718855621,"Kandy","RuwanS0001","RuwanS@2022", "199090456611", "02/09/2028");


  cout <<"--- Display Bus Driver Details --" << endl;
    driver1->printBusDriverDetails();
    cout << endl << endl;
    cout <<"--- Input Bus Driver Details --" << endl;
                driver1->setBusDriverDetails();
                cout << endl << endl;
    cout <<"--- Update Bus Driver Details --" << endl;
    driver1->updateBusDriverDetails();
    cout << endl << endl;
    delete traveller1;
    delete driver1;
            return 0;
}
```