Sri Lanka Institute of Information Technology

Topic : Vehicle Rental System

Group no : MLB_WE_01.02_04

Campus : Malabe

Submission Date : 17 / 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**
IT21353116	Perera I.T.M.	075 280 5630
IT21576966	Weedagamaarachchi K.S.	076 242 0388
IT21350146	Hettiarachchi K.S.	071 693 6932

Table of Contents

Part 1 : Noun – Verb Analysis

Requirements Description

Nouns are highlighted in blue
Verbs are highlighted in green

These are a list of terms that appears in the description.

Guest – Any person who is not registered in the system
Customer – A registered customer
Agent – A person who lends vehicles that are rented to the customers
Itinerary – A record that includes data related to a rental

01) All Users

User Requirements

1. Any individual who can access the system (includes external parties) shall be able to search the vehicles available in any branch.
2. They shall be able to use available search filters to filter out the results.
3. If the required vehicle is not found, he/she should be able to suggest the vehicle via the system.
4. An unregistered customer (guest) shall be able to register as a customer.

System Requirements

1. The system shall provide access to search vehicles to all users.
2. The system shall not allow unregistered customers to reserve vehicles. A message shall be displayed requesting to register in the system.

02) Customer (Registered Customer)

User Requirements

1. A registered customer shall be able to reserve a vehicle for a future date via the system.
2. He/she shall be able to pay for the reservation using the available methods.
3. A customer shall be able to cancel a reservation before 72 hours from the pick-up date.
4. A registered customer shall be able to view information on the reservations the customer has done and his/her previous rentals.
5. The customer shall be able to make changes to his/her profile details.

System Requirements

1. The system shall give access to a customer who has logged in, to reserve a vehicle.
2. The system shall verify the login details of the customer before providing access.
3. The system shall allow a customer to cancel a reservation only before 72 hours from the pick-up date
4. The system shall update the customer's profile regularly

03) Agent

User Requirements

1. An agent shall be able to view the condition of the vehicles he/she has given
2. An agent shall be able to request for a list of itineraries of the particular vehicle rented by him/her.
3. An agent shall be able to view and update the user profile details.

System Requirements

1. The system shall provide access to view vehicle condition only for the agents who provides valid login credentials
2. The system shall forward a request for a list of itineraries of the particular vehicle rented by the agent, to the management
3. The system shall allow an agent to view and update the user profile details.

04) Officer-in-charge

User Requirements

1. Officer-in-charge must be able to confirm rental and return of vehicles via the system.
2. He/she shall be able to view the list of reservations and itineraries made by customers.

System Requirements

1. The system shall automatically retrieve the date and time from the machine when officer-in-charge confirms a rental / return.
2. When the officer-in-charge confirms rental/ return, the system shall automatically update the customer page, the vehicle page, the list of transactions, and if available, the driver's page and agent page, and the database.
3. Upon confirmation of return, the system shall automatically calculate the bill due for the customer.

05) System Admin

User Requirements

1. System admin shall be able to add and remove a customer, driver, or agent account.
2. System admin shall be able to perform updates to the system
3. System admin shall be able to request status reports from the system

System Requirements

1. The system shall provide access to a system admin only when valid user credentials are given.
2. The system shall provide access to admin to update the system.
3. The system shall provide system status reports on a daily basis, and when required.
4. The system shall be able to automatically backup the database.

06) Management

User Requirements

1. The management, upon login, shall be able to generate reports of their request.
2. The management shall be able to add and remove a customer, driver, or agent account.
3. The management shall be able to request status reports from the system
4. The management shall be able to view customer suggestions.

System Requirements

1. The system shall provide access to management to database only if valid credentials are provided.
2. The system shall retrieve data from the database and automatically create and display reports, based on the parameters provided by the management.
3. The system shall provide system status reports when a request was forwarded by the management.

07) Driver

User Requirements

1. A driver, upon login, shall be able to view the list of appointments he/she has, upon request.

System Requirements

1. The system shall provide access if only valid user credentials are provided.
2. Upon request, the system shall retrieve data from the database and create a report of list of appointments the driver has.

List of Classes Identified

Unregistered customer (Guest)
Registered customer (Customer)
Agent
Driver
Reservation
Itinerary
Report
Payment

Part 2 : CRC Cards

Guest	
Responsibilities	**Collaboration**
Register in the system	
Search vehicles	Vehicle
Suggest vehicles	

Customer	
Responsibilities	**Collaboration**
Store details of customers	
Search vehicles	Vehicle
Suggest vehicles	
Make reservations	Reservations
Make payment	Payment

Vehicle	
Responsibilities	**Collaboration**
Store details of vehicles	
Confirm rental of vehicle	Itinerary
Confirm return of vehicle	Itinerary

Agent	
Responsibilities	**Collaboration**
Store details of agents	
Lend vehicles	Vehicle

Driver	
Responsibilities	**Collaboration**
Store details of drivers	
Accommodate a rental	Itinerary

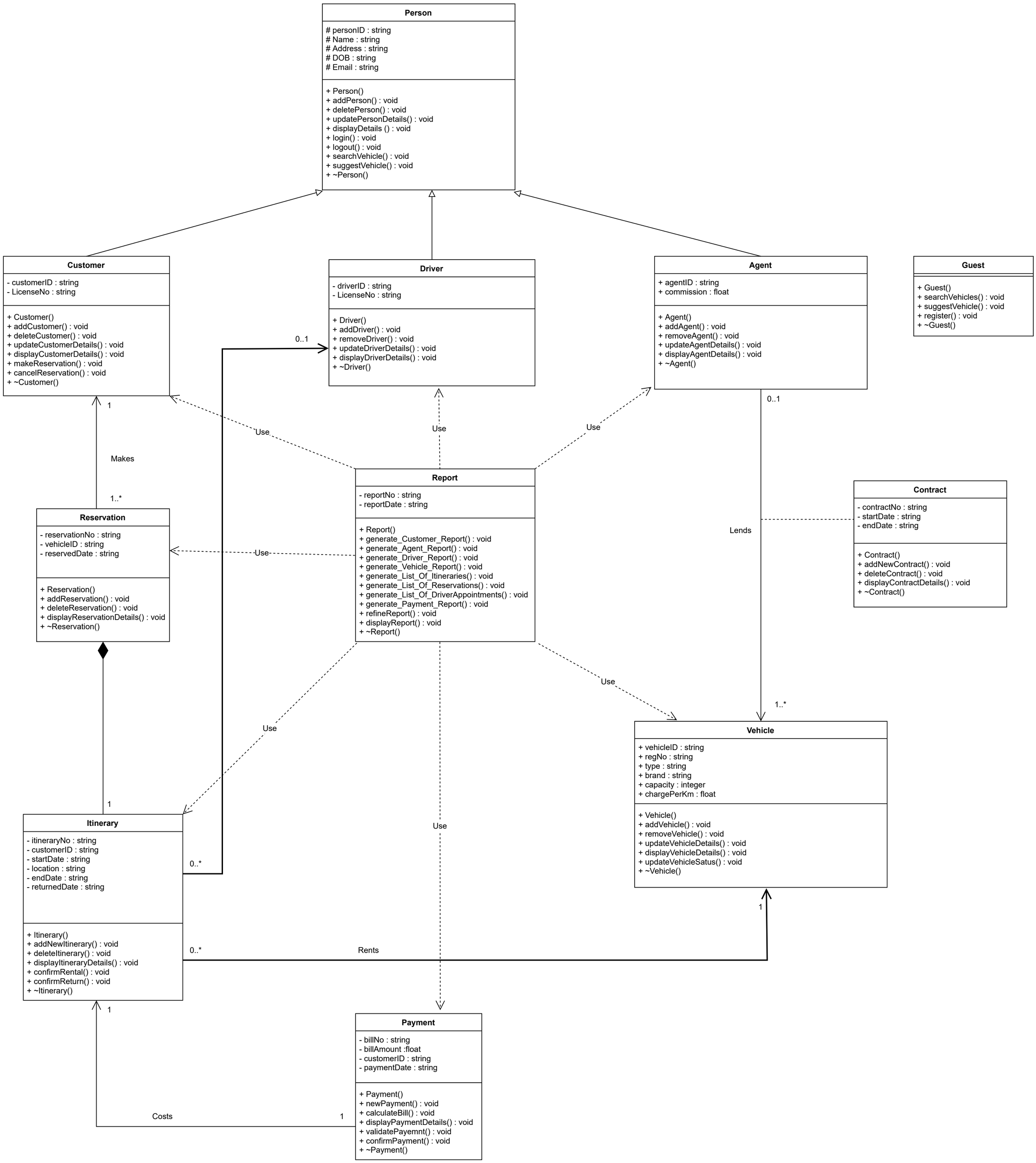
Reservation	
Responsibilities	**Collaboration**
Store details of reservations	
Create itineraries for reservations	Itinerary

Itinerary	
Responsibilities	**Collaboration**
Store details of vehicle rentals	
Confirm renting out of vehicle	
Confirm return of vehicle	
Provide details for calculation of bill	Payment

Payment	
Responsibilities	**Collaboration**
Store details of payments	
Calculate bills	Itinerary
Validate payment	

Report	
Responsibilities	**Collaboration**
Generate customer report	Customer
Generate vehicle report	Vehicle
Generate agent report	Agent
Generate driver report	Driver
Generate list of previous itineraries of a customer	Itinerary, Customer
Generate list of reservations	Reservations
Generate driver appointments reports	Itinerary, Driver
Generate payment reports	Payment
Refine report details based on given parameters	
Display reports	

Part 3 : Class Diagram



Part 4 : Coding

Person.h

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

class Person
{
protected :
	string personID ;
	string Name;
	string Address;
	string DOB;
	string Email;

public :
	Person();
	Person(string pID, string p_name, string p_address, string p_DOB, string p_email);
	void addPerson();
	void deletePerson();
	void updatePersonDetails();
	void displayDetails();
	void login();
	void logout();
	void searchVehicle();
	void suggestVehicle();
	~Person();
};
```

Person.cpp

```
#include "Person.h"
#include <iostream>
#include <string>
using namespace std;

Person::Person() {
	personID = "P000";
	Name = "";
	Address = "";
	DOB = "DD/MM/YYYY";
	Email = "";
}

Person::Person(string pID, string p_name, string p_address, string p_DOB, string p_email)
{
	personID = pID;
	Name = p_name;
	Address = p_address;
	DOB = p_DOB;
	Email = p_email;
}

void Person::addPerson() {

}

void Person::deletePerson() {

}

void Person::updatePersonDetails() {

}

void Person::displayDetails() {
}

void Person::login() {
}

void Person::logout() {

}

void Person::searchVehicle() {

}
```

```
void Person::suggestVehicle() {

}

Person::~Person() {
	cout << "Destructor running for Person" << endl;
}
```

Customer.h

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

class Customer : public Person
{
private:
	string customerID;
	string LicenseNo;

public :
	Customer();
	Customer(string cID, string pID, string c_name, string c_address, string c_DOB,
string c_email, string c_license);
	void addCustomer();
	void deleteCustomer();
	void updateCustomerDetails();
	void displayCustomerDetails();
	void makeReservation();
	void cancelReservation();
	~Customer();
};
```

Customer.cpp

```
#include "Customer.h"
#include <iostream>
#include <string>
using namespace std;

Customer::Customer() {
	customerID = "C000";
	LicenseNo = "";
}

Customer::Customer(string cID, string pID, string c_name, string c_address, string c_DOB,
string c_email, string c_license)
{
	customerID = cID;
	personID = pID;
	Name = c_name;
	Address = c_address;
	DOB = c_DOB;
	Email = c_email;
	LicenseNo = c_license;
}

void Customer::addCustomer() {

}

void Customer::deleteCustomer() {

}

void Customer::updateCustomerDetails() {

}
void Customer::displayCustomerDetails() {

}

void Customer::makeReservation() {

}

void Customer::cancelReservation() {

}

Customer::~Customer() {
	cout << "Destructor running for Customer" << endl;
}
```

Agent.h

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

class Agent : public Person
{
private :
	string agentID;
	double commission;

public :
	Agent();
	Agent(string aID, string pID, string a_name, string a_address, string a_DOB, string a_email, double a_commission);
	void addAgent();
	void removeAgent();
	void updateAgentDetails();
	void displayAgentDetails();
	~Agent();
};
```

Agent.cpp

```
#include "Agent.h"
#include <iostream>
#include <string>
using namespace std;

Agent::Agent()
{
	agentID = "A000";
	commission = 0;
}

Agent::Agent(string aID, string pID, string a_name, string a_address, string a_DOB, string a_email, double a_commission)
{
	agentID = aID;
	personID = pID;
	Name = a_name;
	Address = a_address;
	DOB = a_DOB;
	Email = a_email;
	commission = a_commission;
}

void Agent::addAgent() {

}

void Agent::removeAgent() {

}

void Agent::updateAgentDetails() {

}

void Agent::displayAgentDetails() {

}

Agent::~Agent() {
	cout << "Destructor running for Agent" << endl;
}
```

Driver.h

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

class Driver : public Person
{
private :
	string driverID;
	string LicenseNo;

public :
	Driver();
	Driver(string dID, string pID, string d_name, string d_address, string d_DOB, string d_email, string d_license);
	void addDriver();
	void removeDriver();
	void updateDriverDetails();
	void displayDriverDetails();
	~Driver();
};
```

Driver.cpp

```
#include "Driver.h"
#include <iostream>
#include <string>
using namespace std;

Driver::Driver()
{
	driverID = "D000";
	LicenseNo = "";
}

Driver::Driver(string dID, string pID, string d_name, string d_address, string d_DOB, string d_email, string d_license)
{
	driverID = dID;
	personID = pID;
	Name = d_name;
	Address = d_address;
	DOB = d_DOB;
	Email = d_email;
	LicenseNo = d_license;
}

void Driver::addDriver()
{
}

void Driver::removeDriver()
{
}

void Driver::updateDriverDetails()
{
}

void Driver::displayDriverDetails()
{
}

Driver::~Driver()
{
	cout << "Destructor running for Driver" << endl;
}
```

Guest.h

```
#pragma once

class Guest
{
public:
	Guest();
	void searchVehicle();
	void suggestVehicle();
	void Register();
	~Guest();
};
```

Guest.cpp

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

Guest::Guest()
{
}

void Guest::searchVehicle()
{
}

void Guest::suggestVehicle()
{
}

void Guest::Register()
{
}

Guest::~Guest()
{
	cout << "Destructor running for Guest" << endl;
}
```

Vehicle.h

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

class Vehicle
{
private :
	string vehicleID;
	string regNo;
	string type;
	string brand;
	int capacity;
	double chargePerKm;

public :
	Vehicle();
	Vehicle(string vID, string v_regNo, string v_type, string v_brand, int v_capacity,
double v_charge);
	void addVehicle();
	void removeVehicle();
	void updateVehicleDetails();
	void displayVehicleDetails();
	void updateVehicleSatus();
	~Vehicle();
};
```

Vehicle.cpp

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

Vehicle::Vehicle()
{
	vehicleID = "V000";
	regNo = "";
	type = "";
	brand = "";
	capacity = 0;
	chargePerKm = 0;
}

Vehicle::Vehicle(string vID, string v_regNo, string v_type, string v_brand, int v_capacity, double v_charge)
{
	vehicleID = vID;
	regNo = v_regNo;
	type = v_type;
	brand = v_brand;
	capacity = v_capacity;
	chargePerKm = v_charge;
}

void Vehicle::addVehicle()
{
}

void Vehicle::removeVehicle()
{
}

void Vehicle::updateVehicleDetails()
{
}

void Vehicle::displayVehicleDetails()
{
}

void Vehicle::updateVehicleSatus()
{
}

Vehicle::~Vehicle()
{
	cout << "Destructor running for Vehicle" << endl;
}
```

Contract.h

```
#pragma once
#include "Agent.h"
#include "Vehicle.h"
#include <string>
using namespace std;

class Contract
{
private :
	string contractNo;
	Agent* a;
	Vehicle* v;
	string startDate;
	string endDate;

public :
	Contract();
	Contract(string cntrct_No, Agent* Agent, Vehicle* Vehicle, string cntrct_startDate,
string cntrct_endDate);
	void addNewContract();
	void deleteContract();
	void displayContractDetails();
	~Contract();
};
```

Contract.cpp

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

Contract::Contract()
{
	contractNo = "000000";
	startDate = "DD/MM/YYYY";
	endDate = "DD/MM/YYYY";
}

Contract::Contract(string cntrct_No, Agent* Agent, Vehicle* Vehicle, string cntrct_startDate,
string cntrct_endDate)
{
	contractNo = cntrct_No;
	a = Agent;
	v = Vehicle;
	startDate = cntrct_startDate;
	endDate = cntrct_endDate;
}

void Contract::addNewContract()
{
}

void Contract::deleteContract()
{
}

void Contract::displayContractDetails()
{
}

Contract::~Contract()
{
	cout << "Destructor running for Contract" << endl;
}
```

Reservation.h

```
#pragma once
#include <string>
#include "Customer.h"
#include "Itinerary.h"
using namespace std;

class Reservation
{
private :
	string reservationNo;
	string reservedDate;
	Customer* c;
	Itinerary* i;


public :
	Reservation();
	Reservation(string R_No, string R_date, Customer* Cus, Itinerary* Iti);
	void addReservation();
	void deleteReservation();
	void displayReservationDetails();
	~Reservation();
};
```

Reservation.cpp

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

Reservation::Reservation()
{
	reservationNo = "R0000";
	reservedDate = "DD/MM/YYYY";
	i = new Itinerary();
}

Reservation::Reservation(string R_No, string R_date, Customer* Cus, Itinerary* Iti)
{
	reservationNo = R_No;
	reservedDate = R_date;
	c = Cus;
	i = Iti;
}

void Reservation::addReservation()
{
}

void Reservation::deleteReservation()
{
}

void Reservation::displayReservationDetails()
{
}

Reservation::~Reservation()
{
	delete i;
	cout << "Destructor running for Reservation" << endl;
}
```

Itinerary.h

```
#pragma once
#include <string>
#include "Vehicle.h"
#include "Driver.h"
using namespace std;


class Itinerary
{
private :
	string ItineraryNo;
	Vehicle* V;
	Driver* D;
	string startDate;
	string location;
	string endDate;
	string returnedDate;

public :
	Itinerary();
	Itinerary(string I_No, Vehicle* Vehicle, string I_startDate, Driver* Driver, string
I_location, string I_endDate, string I_returnedDate);
	void addNewItinerary();
	void deleteItinerary();
	void displayItineraryDetails();
	void confirmRental();
	void confirmReturn();
	~Itinerary();
};
```

Itinerary.cpp

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

Itinerary::Itinerary()
{
	ItineraryNo = "I0000";
	startDate = "DD/MM/YYYY";
	location = "";
	endDate = "DD/MM/YYYY";
	returnedDate = "DD/MM/YYYY";
}


Itinerary::Itinerary(string I_No, Vehicle* Vehicle, string I_startDate, Driver* Driver, string
I_location, string I_endDate, string I_returnedDate)
{
	ItineraryNo = I_No;
	V = Vehicle;
	startDate = I_startDate;
	D = Driver;
	location = I_location;
	endDate = I_endDate;
	returnedDate = I_returnedDate;
}

void Itinerary::addNewItinerary()
{
}

void Itinerary::deleteItinerary()
{
}

void Itinerary::displayItineraryDetails()
{
}

void Itinerary::confirmRental()
{
}

void Itinerary::confirmReturn()
{
}

Itinerary::~Itinerary()
{
	cout << "Destructor running for Itinerary" << endl;
}
```

Payment.h

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


class Payment
{
private :
	string billNo;
	Itinerary* i;
	double billAmount;
	string customerID;
	string paymentDate;

public :
	Payment();
	Payment(string bill, double amount, string cID, string billDate, Itinerary* I);
	void newPayment();
	void calculateBill();
	void displayPayemntDetails();
	void validatePayment();
	void confirmPayment();
	~Payment();
};
```

Payment.cpp

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

Payment::Payment()
{
	billNo = "000000";
	billAmount = 0;
	customerID = "C000";
	paymentDate = "DD/MM/YYYY";
}

Payment::Payment(string bill, double amount, string cID, string billDate, Itinerary* I)
{
	billNo = bill;
	billAmount = amount;
	customerID = cID;
	paymentDate = billDate;
	i = I;
}

void Payment::newPayment()
{
}

void Payment::calculateBill()
{
}

void Payment::displayPayemntDetails()
{
}

void Payment::validatePayment()
{
}

void Payment::confirmPayment()
{
}

Payment::~Payment()
{
	cout << "Destructor running for Payment" << endl;
}
```

Report.h

```
#pragma once
#include "Customer.h"
#include "Agent.h"
#include "Driver.h"
#include "Vehicle.h"
#include "Itinerary.h"
#include "Reservation.h"
#include "Payment.h"
#include <string>
using namespace std;

class Report
{
private :
        int reportNo;
        string reportDate;

public :
        Report();
        Report(int rprtNo, string rprt_date);
        void generate_Customer_Report(Customer* c);
        void generate_Agent_Report(Agent* a);
        void generate_Driver_Report(Driver* d);
        void generate_Vehicle_Report(Vehicle* v);
        void generate_List_Of_Itineraries(Itinerary* i);
        void generate_List_Of_Reservations(Reservation* r);
        void generate_List_Of_DriverAppointments(Itinerary* i);
        void generate_Payment_Report(Payment* p);
        void refineReport();
        void displayReport();
        ~Report();
};
```

Report.cpp

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

Report::Report()
{
	reportNo = 0;
	reportDate = "DD/MM/YYYY";
}

Report::Report(int rprtNo, string rprt_date)
{
	reportNo = rprtNo;
	reportDate = rprt_date;
}

void Report::generate_Customer_Report(Customer* c)
{
}

void Report::generate_Agent_Report(Agent* a)
{
}

void Report::generate_Driver_Report(Driver* d)
{
}

void Report::generate_Vehicle_Report(Vehicle* v)
{
}

void Report::generate_List_Of_Itineraries(Itinerary* i)
{
}

void Report::generate_List_Of_Reservations(Reservation *r)
{
}

void Report::generate_List_Of_DriverAppointments(Itinerary* i)
{
}

void Report::generate_Payment_Report(Payment* p)
{
}
```

```
void Report::refineReport()
{
}

void Report::displayReport()
{
}

Report::~Report()
{
        cout << "Destructor running for Report" << endl;
}
```

Main.cpp

```
#include <iostream>
#include "Agent.h"
#include "Customer.h"
#include "Driver.h"
#include "Contract.h"
#include "Guest.h"
#include "Itinerary.h"
#include "Person.h"
#include "Payment.h"
#include "Reservation.h"
#include "Report.h"
#include "Vehicle.h"
#include <string>
using namespace std;

int main() {

	//Creating Objects

	Customer* c = new Customer();
	Agent* a = new Agent();
	Driver* d = new Driver();
	Vehicle* v = new Vehicle();
	Contract* cntrct = new Contract();
	Reservation* r = new Reservation();
	Itinerary* i = new Itinerary();
	Payment* pymnt = new Payment();
	Report* rprt = new Report();
	Person* p = new Person();
	Guest* g = new Guest();


	//Deleting Objects

	delete c;
	delete a;
	delete d;
	delete v;
	delete cntrct;
	delete r;
	delete i;
	delete pymnt;
	delete rprt;
	delete p;
	delete g;

	return 0;
}
```

Individual Contribution

Student ID	Student Name	Classes and CRC Cards
IT21353116	Perera I.T.M.	Itinerary, Reservation, Payment, Report, Person
IT21576966	Weedagamaarachchi K.S.	Customer, Driver, Guest
IT21350146	Hettiarachchi K.S.	Agent, Vehicle, Contract