Topic : Vehicle Rental System

Group no : MLB_06.02_02

Campus : Malabe

Submission Date :05/20/2022

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**
IT21343520	Wijerama H.J.K.S.R	0764888184
IT21341922	Herath H.M.H.N	0713747929
IT21345500	T.D.T.L Thenuwara	0719882341
IT21334306	W.H.D.Wathsala	0713423059
IT21342608	Kumaravithana D.B	0703015048

1) System Requirements

1. A customer/Guest can visit the Vehicle Rental System website.
2. Unregistered customer (Guest) should sign up with the system before logging to the system. Then he/she can begin to take the service.
3. Unregistered customer provides name, NIC, address etc. as register details.
4. Registered customer (regular customer) can use the system using customer ID as usual.
5. The admin can add new, remove, restock vehicles to the system.
6. The customer can search available vehicles according to their requirements and favors at the moment.
7. A customer can book from vehicle rental system. A Booking can consist of multiple vehicles.
8. The customer selects a payment method for each booking (Cash/Card/PayPal).
9. Once the customer finalizes the booking, payment is getting validated, and booking ID is assigned to the particular booking.
10. The admin analyses the booking ID and assign staff accordingly.
11. The staff notifies the customer about the status of the booking.
12. The financial records are updated by admin.
13. The staff assures the booking is made and gives the notification to the driver about the booking.
14. The driver contacts the customer and drives the vehicle that were booked.
15. The customer is able to give feedbacks to the service received.
16. The admin is able to add, remove, update the system according to the feedbacks of customer.
17. The admin is able to assign and change the staff and the driver.

2) Noun & Verb Analysis

(Nouns are in RED color and Verbs are in BLUE color)

1. A customer/Guest can visit the Vehicle Rental System website.
2. Unregistered customer (Guest) should sign up with the system before log in to the system. Then he/she can begin to take the service.
3. Unregistered customer provides Name, NIC, Address etc. as register details.
4. Registered customer (regular customer) can use the system using customer ID as usual.
5. The admin can add new, remove, restock vehicles to the system
6. The customer can search available vehicles according to their requirements and favors at the moment.
7. A customer can book from vehicle rental system. A Booking can consist of multiple vehicles.
8. The customer selects a payment method for each booking (Cash/Card/PayPal).
9. Once the customer finalizes the booking, payment is getting validated, and booking ID is assigned to the particular booking.
10. The admin analyses the booking ID and assign staff accordingly.
11. The staff notifies the customer about the status of the booking.
12. The financial records are updated by admin.
13. The staff assures the booking is made and gives the notification to the driver about the booking
14. The driver contacts the customer and drives the vehicle that were booked
15. The customer is able to give feedbacks to the service received.
16. The admin is able to add, remove, update the system according to the feedbacks of customer.
17. The admin is able to assign and change the staff and driver.

Identified Nouns

- Customer - Class
- Unregistered customer (Guest) – Redundant
- Registered customer – Meta language
- Customer ID – Attribute
- Name, NIC, Address - Attribute
- Vehicle – Class
- Driver - Class
- Booking - Class
- Payment – Class
- Cash – Class (Inherited from “Payment” class)
- Card – Class (Inherited from “Payment” class)
- PayPal- Class (Inherited from “Payment” class)
- Booking ID - Attribute
- Staff - Class
- Financial Record - Class
- Admin - Class
- Feedback - Class

Identified Classes

1. Customer
2. Vehicle
3. Booking
4. Payment
5. Staff
6. Financial Record
7. Admin
8. Driver
9. Feedback
10. Cash
11. Card
12. PayPal

Identified Verbs

Customer

- Visit
- Search vehicles
- Register
- Book a vehicle
- Select a payment method
- Give feedbacks

Admin

- Add new vehicles
- Remove vehicles
- Restock vehicles
- Assign drivers, staff
- Change drivers
- Update system
- Analyses the booking ID
- Confirm the booking
- Update financial Report

Staff

- Notifies the customer
- Assure the booking

Driver

- Contact the customer
- Drive the vehicle that were booked

3) CRC Cards for the System

Customer	
Responsibility	**Collaborators**
Search vehicle	
Booking vehicle	
Payment of fees	
Giving Feedback	

Vehicle	
Responsibility	**Collaborators**
Add vehicles	
Remove vehicles	
Restock vehicles	
Update vehicles	

Booking	
Responsibility	**Collaborators**
Book	Vehicle
Status of booking	
Confirm booking	Payment

Payment	
Responsibility	**Collaborators**
Add payment details	Booking
Validate payment details	

Staff	
Responsibility	**Collaborators**
Notify the Customer about the status of the booking	Customer, Booking
Assures the booking is made	Booking

Financial Record	
Responsibility	**Collaborators**
Store details of transaction	Booking, Vehicles, Payment
Update	Booking, Vehicles, Payment

Admin	
Responsibility	**Collaborators**
Add new, remove, restock vehicles	Vehicle
Add, remove, update the system according to the feedbacks of customer.	Feedback
Update the financial report	Booking, Customer

Driver	
Responsibility	**Collaborators**
Contact the customer	Customer
Drive the vehicle that were booked	

Feedback	
Responsibility	**Collaborators**
Add feedback	Customer
Display feedback	

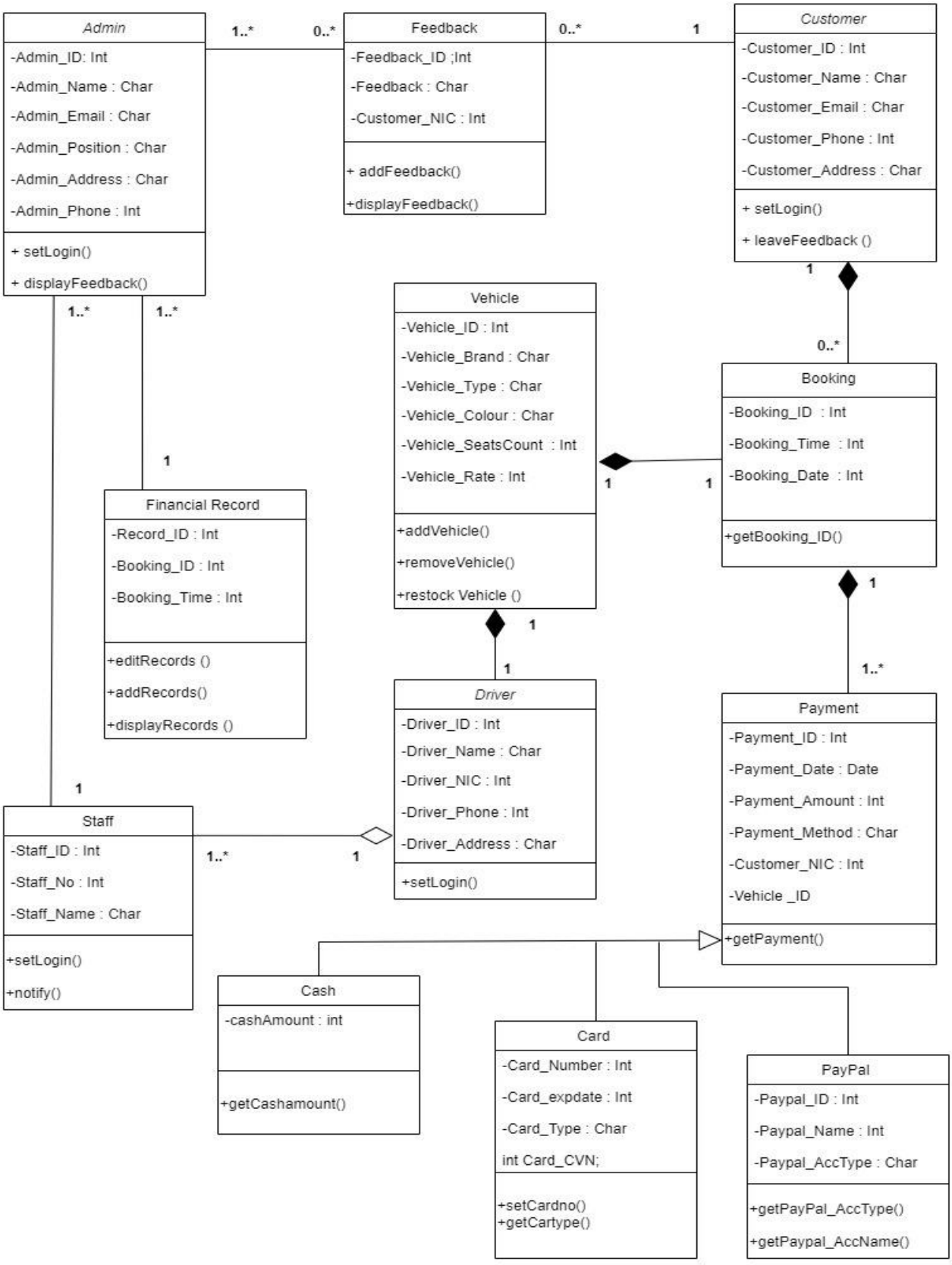
Cash	
Responsibility	**Collaborators**
Booking details done by cash Payment	Payment

Card	
Responsibility	**Collaborators**
Booking details done by cash Payment	Payment

PayPal	
Responsibility	**Collaborators**
Payment details done through PayPal	Payment

➢ Exercise 1:

UML Class Diagram



➢ Exercise 2:

C++ Code

(vehicle.h File)

//---->>> Declaration of Classes <<<----

```
//Declaration of "Customer" class

class Customer
{
private:
        char Customer_ID[];
        char Customer_Name[30];
        char Customer_Email[30];
        int Customer_Phone;
        char Customer_Address[30];

        Feedback* feedback[SIZE];
        Booking* bookings[SIZE];

public:
        Customer();
        Customer(const char ID[], const char Name[], const char Email[], int Phone, const char Address[]);
        void leaveFeedback(Feedback* feedbacks);
        void addBookingDetails(Booking* bookings[]);
        void setLoging();
        ~Customer();
};
```

//Declaration of " Vehicle " class

```
class Vehicle
{
private:
	char Vehicle_ID[10];
	char Vehicle_Brand[30];
	char Vehiccle_Type[30];
	char Vehicle_Color[30];
	int Vehicle_SeatCount;
	int VEhicle_Rate;

	Booking* bookings[SIZE];
	Driver* drivers[SIZE];
public:
	Vehicle();
	Vehicle(const char ID[], const char Brand[], const char Type[], const char Color[], int sCount, int
Rate);
	void addBookingDetails(Booking* bookings[]);
	void addDriverDetails(Driver* drivers[]);
	void addVehicle();
	void removeVehicle();
	void restockVehicle();
	~Vehicle();
};
```

//Declaration of " Booking " class

```
class Booking
{
private:
	char Booking_ID[10];
	char Booking_Time[10];
	char Booking_Date[10];

	Payment* payments[SIZE];

public:
	Booking();
	Booking(const char ID[], const char Time[], const char Date[]);
	void addPaymentDetails(Payment* payments[]);
	void getBooking_ID();
	~Booking();
};
```

```
//Declaration of "Payment" class


class Payment
{
protected:
        char Payment_ID[10];
        char payment_Date[10];
        int Payment_Amount;
        char Payment_Method[10];
public:
        Payment();
        Payment(const char ID[], const char Date[], int Amount, const char Method[]);
        void getPayment();
        void displayPaymentDetails();
        ~Payment();
};


  //Declaration of " Staff" class


class Staff
{
private:
        char staff_ID[10];
        char staff_Name[30];

        Admin* admin[SIZE];

public:
        Staff();
        Staff(const char ID[], const char Name[]);
        void staffsAdmin(Admin* admins);
        void setLogin();
        ~Staff();
};



//Declaration of " Financial Record " class


class FinancialRecord
{
```

```
private:
    char Record_ID[10];

    Admin* admin;

public:
    FinancialRecord();
    FinancialRecord(const char ID[], Admin* admins);
    void editRecords();
    void addRecords();
    ~FinancialRecord();
};
```

//Declaration of " Admin " class

```
class Admin
{
private:
        char Admin_ID[10];
        char Admin_Name[30];
        char Admin_Email[30];
        char Admin_Position[30];
        char Admin_Address[30];
        int Admin_Phone;

        FinancialRecord* financialRecords;
        Staff* theStaff[SIZE];
        Feedback* feedback[SIZE];

public:
        Admin();
        Admin(const char ID[], const char Name[], const char Email[], const char Position[], const char
Address[], int Phone, FinancialRecord* records);
        void setLogin();
        void adminsStaff(Staff* staff);
        void adminsFeedback(Feedback* feedbacks);
        ~Admin();
};
```

//Declaration of " Driver" class

```
class Driver
{
private:
        char Driver_ID[10];
        char Driver_Name[30];
        char Driver_NIC[12];
        int Driver_Phone;
        char Driver_Address[30];

        Staff* staff[SIZE];
```

```
public:
        Driver();
        Driver(const char ID[], const char Name[], const char NIC[], int Phone, const char Address[]);
        void addStaffDetails(Staff* staff[]);
        void displayDriver();
        void setLoging();
        ~Driver();
};
```

//Declaration of " Feedback" class

```
class Feedback
{
private:
        char Feedback_ID[10];
        char Desc[200];

        Admin* admin[SIZE];
        Customer* customer;

public:
        Feedback();
        Feedback(const char ID[], const char Fback[], Customer* customers);
        void feedbacksAdmin(Admin* admins);
        void addFeedback();
        ~Feedback();
};
```

//Declaration of " Cash" class

```
class Cash:public Payment
{
private:
        float Cash_Amount;

public:
        Cash();
        Cash(float amount);
        ~Cash();

};
```

//Declaration of "Card" class

```
class Card:public Payment {
private:
        int Card_Number;
        char Card_Type[10];
        char Exp_Date[10];
        int Card_CVN;
public:
        Card();
        Card(int card_no, const char card_type[], const char exp_date[], int cvn);
        void card_authorize();
        ~Card();
};
```

//Declaration of " PayPal" class

```
class PayPal:public Payment {
private:
        char Paypal_AccType[20];
        char Paypal_Name[20];
        int Paypal_ID;
public:
        PayPal();
        PayPal(const char accType[], const char paypal_name[], int paypal_id);
        void paypal_authorize();
        ~PayPal();
};
```

(vehicle.cpp File)

//---->>> Implementation of Classes <<<----

//Implementation of "Customer" class

```
#include "Customer.h"

Customer::Customer()
{
	strcpy_s(Customer_ID, "");
	strcpy_s(Customer_Name, "");
	strcpy_s(Customer_Email, "");
	Customer_Phone = 0;
	strcpy_s(Customer_Address, "");
}

Customer::Customer(const char ID[], const char Name[], const char Email[], int Phone, const char Address[])
{
	strcpy_s(Customer_ID, ID);
	strcpy_s(Customer_Name, Name);
	strcpy_s(Customer_Email, Email);
	Customer_Phone = Phone;
	strcpy_s(Customer_Address, Address);
}

void Customer::leaveFeedback(Feedback* feedbacks)
{
}
```

```
void Customer::addBookingDetails(Booking* bookings[])
{
}

void Customer::setLoging()
{
}

Customer::~Customer()
{
	cout << "Customer deleted" << endl;
	for (int i = 0; i < SIZE; i++) {
		delete bookings[i];
	}
}
```

//Implementation of "Vehicle" class

```
#include "Vehicle.h"
#include "Driver.h"
#include "Booking.h"

Vehicle::Vehicle()
{
	strcpy_s(Vehicle_ID, "");
	strcpy_s(Vehicle_Brand, "");
	strcpy_s(Vehiccle_Type, "");
	strcpy_s(Vehicle_Color, "");
	Vehicle_SeatCount = 0;
	VEhicle_Rate = 0;
}

Vehicle::Vehicle(const char ID[], const char Brand[], const char Type[], const char Color[], int sCount, int Rate)
{
	strcpy_s(Vehicle_ID, ID);
	strcpy_s(Vehicle_Brand, Brand);
	strcpy_s(Vehiccle_Type, Type);
	strcpy_s(Vehicle_Color, Color);
	Vehicle_SeatCount = sCount;
	VEhicle_Rate = Rate;
}

void Vehicle::addBookingDetails(Booking* bookings[])
{
}

void Vehicle::addDriverDetails(Driver* drivers[])
```

```
{
}

void Vehicle::addVehicle()
{
}

void Vehicle::removeVehicle()
{
}

void Vehicle::restockVehicle()
{
}

Vehicle::~Vehicle()
{
	cout << "Vehicle delete" << endl;
	for (int i = 0; i < SIZE; i++) {
		delete bookings[i];
		delete drivers[i];
	}
}
```

//Implementation of "Booking" class

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

Booking::Booking()
{
	strcpy_s(Booking_ID, "");
	strcpy_s(Booking_Time, "");
	strcpy_s(Booking_Date, "");
}

Booking::Booking(const char ID[], const char Time[], const char Date[])
{
	strcpy_s(Booking_ID, ID);
	strcpy_s(Booking_Time, Time);
	strcpy_s(Booking_Date, Date);
}

void Booking::addPaymentDetails(Payment* payments[])
{
}

void Booking::getBooking_ID()
{
```

```
}

Booking::~Booking()
{
	cout << "Booking deleted" << endl;
	for (int i = 0; i < SIZE; i++) {
		delete payments[i];
	}

}
```

//Implementation of "Payment" class

```
#include "Payment.h"


Payment::Payment()
{
	strcpy_s(Payment_ID, "");
	strcpy_s(payment_Date, "");
	Payment_Amount = 0;
	strcpy_s(Payment_Method, "");
}

Payment::Payment(const char ID[], const char Date[], int Amount, const char Method[])
{
	strcpy_s(Payment_ID, ID);
	strcpy_s(payment_Date, Date);
	Payment_Amount = Amount;
	strcpy_s(Payment_Method, Method);
}
```

```
void Payment::getPayment()
{
}

void Payment::displayPaymentDetails()
{
}

Payment::~Payment()
{
	cout << "Payment details deleted" << endl;

}
```

//Implementation of "Staff" class

```
#include "Staff.h"
#include "Admin.h"

Staff::Staff()
{
	strcpy_s(staff_ID, "");
	strcpy_s(staff_Name, "");
}

Staff::Staff(const char ID[], const char Name[])
{
	strcpy_s(staff_ID, ID);
	strcpy_s(staff_Name, Name);
}
```

```
void Staff::staffsAdmin(Admin* admins)
{
}

void Staff::setLogin()
{
}

Staff::~Staff()
{
	cout << "TheStaff deleted" << endl;
}
```

//Implementation of "Financial Record" class

```
#include "FinancialRecord.h"
#include "Admin.h"

FinancialRecord::FinancialRecord()
{
	strcpy_s(Record_ID, "");
}

FinancialRecord::FinancialRecord(const char ID[], Admin* admins)
{
```

```
	strcpy_s(Record_ID, ID);
	admin = admins;
}

void FinancialRecord::editRecords()
{
}

void FinancialRecord::addRecords()
{
}

FinancialRecord::~FinancialRecord()
{
	cout << "TheFinancialRecords deleted" << endl;
}
```

//Implementation of "Admin" class

```
#include "Admin.h"
#include "FinancialRecord.h"
#include "Staff.h"
#include "Feedback.h"

Admin::Admin()
{
```

```
	strcpy_s(Admin_ID, "");
	strcpy_s(Admin_Name, "");
	strcpy_s(Admin_Email, "");
	strcpy_s(Admin_Position, "");
	strcpy_s(Admin_Address, "");
	Admin_Phone = 0;
}

Admin::Admin(const char ID[], const char Name[], const char Email[], const char Position[], const char
Address[], int Phone, FinancialRecord* records)
{
	strcpy_s(Admin_ID, ID);
	strcpy_s(Admin_Name, Name);
	strcpy_s(Admin_Email, Email);
	strcpy_s(Admin_Position, Position);
	strcpy_s(Admin_Address, Address);
	Admin_Phone = Phone;

	financialRecords = records;
}

void Admin::setLogin()
{
}

void Admin::adminsStaff(Staff* staff)
{
}

void Admin::adminsFeedback(Feedback* feedbacks)
{
}

Admin::~Admin()
{
	cout << "Admin deleted" << endl;
}
```

//Implementation of "Driver" class

```
#include "Driver.h"
#include "Staff.h"
```

```
Driver::Driver()
{
	strcpy_s(Driver_ID, "");
	strcpy_s(Driver_Name, "");
	strcpy_s(Driver_NIC, "");
	Driver_Phone = 0;
	strcpy_s(Driver_Address, "");
}

Driver::Driver(const char ID[], const char Name[], const char NIC[], int Phone, const char Address[])
{
	strcpy_s(Driver_ID, ID);
	strcpy_s(Driver_Name, Name);
	strcpy_s(Driver_NIC, NIC);
	Driver_Phone = 0;
	strcpy_s(Driver_Address, Address);
}

void Driver::addStaffDetails(Staff* staff[])
{
}

void Driver::displayDriver()
{
}

void Driver::setLoging()
{
}

Driver::~Driver()
{
	cout << "Driver deleted" << endl;
}
```

//Implementation of "Feedback" class

```
#include "Feedback.h"
#include "Admin.h"


Feedback::Feedback()
{
	strcpy_s(Feedback_ID, "");
	strcpy_s(Desc, "");
}

Feedback::Feedback(const char ID[], const char Fback[], Customer* customers)
{
	strcpy_s(Feedback_ID, ID);
	strcpy_s(Desc, Fback);
	customer = customers;
}

void Feedback::feedbacksAdmin(Admin* admins)
{
}

void Feedback::addFeedback()
{
}

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

//Implementation of "Cash" class

```
#include "Cash.h"

Cash::Cash()
{
	Cash_Amount = 0;
}

Cash::Cash(float amount)
{
	Cash_Amount = amount;
}

Cash::~Cash()
{
	cout << "Cash delete" << endl;
```

```
}



//Implementation of "Card" class


#include "Card.h"

Card::Card()
{
	Card_Number = 0;
	strcpy_s(Card_Type, "");
	strcpy_s(Exp_Date, "");
	Card_CVN = 0;

}

Card::Card(int card_no, const char card_type[], const char exp_date[], int cvn)
{
	Card_Number = card_no;
	strcpy_s(Card_Type, card_type);
	strcpy_s(Exp_Date, exp_date);
	Card_CVN = cvn;
}

void Card::card_authorize()
{
}

Card::~Card()
{
	cout << "Card delete" << endl;
}
```

//Implementation of "PayPal" class

```
#include "PayPal.h"

PayPal::PayPal()
{
	strcpy_s(Paypal_AccType, "");
	strcpy_s(Paypal_Name, "");
	Paypal_ID = 0;
}

PayPal::PayPal(const char accType[], const char paypal_name[], int paypal_id)
{
	strcpy_s(Paypal_AccType, accType);
	strcpy_s(Paypal_Name, paypal_name);
	Paypal_ID = paypal_id;
}

void PayPal::paypal_authorize()
{
}

PayPal::~PayPal()
{
	cout << "PayPal delete" << endl;
}
```

(main.cpp File)

```
#include"Admin.h"
#include"Booking.h"
#include"Card.h"
#include"Cash.h"
#include"Customer.h"
#include"Driver.h"
#include"Feedback.h"
#include"FinancialRecord.h"
#include"Payment.h"
#include"PayPal.h"
#include"Staff.h"
#include"Vehicle.h"


#include<iostream>
using namespace std;

int main() {

	//Creat objects to each classes

	Admin* Admin1 = new Admin();
	Booking* Booking1 = new Booking();
	Customer* Customer1 = new Customer();
	Driver* Driver1 = new Driver();
	Feedback* Feedback1 = new Feedback();
	Payment* Payment1 = new Payment();
	TheFinancialRecords* TheFinancialRecords1 = new TheFinancialRecords();
	TheStaff* TheStaff1 = new TheStaff();
	Vehicle* Vehicle1 = new Vehicle();

	//Delete dynamic variables

	delete Admin1, Booking1, Customer1, Driver1, Feedback1, Payment1, TheFinancialRecords1,
TheStaff1, Vehicle1;

	return 0;
}
```

Individual Contribution

	Student ID	Student Name	Individual Contribution
1	IT21343520	Wijerama H.J.K.S.R	CRC Card: Customer, Feedback UML Notation: Customer, Feedback C++ Code: Customer, Feedback
2	IT21341922	Herath H.M.H.N	CRC Card: Booking, Vehicle UML Notation: Booking, Vehicle C++ Code: Booking, Vehicle
3	IT21345500	T.D.T.L Thenuwara	CRC Card: Admin, Financial Record UML Notation: Admin, Financial Record C++ Code: Admin, Financial Record
4	IT21334306	W.H.D.Wathsala	CRC Card: Payment, Cash, Card, PayPal UML Notation: Payment, Cash, Card, PayPal C++ Code: Payment, Cash, Card, PayPal

5	IT21342608	Kumaravithana D.B	CRC Card: Staff, Driver UML Notation: Staff, Driver C++ Code:Staff, Driver