BSc (Hons) in Information Technology

Object Oriented Concepts – IT1050

Assignment 2

2023-June

Topic : **Automated Parking System**

Group no : **MLB_WD_CSNE_01.01_10**

Campus : **Malabe**

Submission Date : **06/14/2023**

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.

Name
T.RAVISHKA LAKSHAN
RANGANA WIJESINGHE
GAVISHKA SAHAN
KAVISHA RAJAPAKSHA
CHANIKA GAYASHAN

1) Requirements of the system

1)The system should support various types of vehicles, including cars, motorcycles, and bicycles.

2)Store information about a vehicle, type, license, plate number, color and Provide methods to update vehicle information.

3)Store vehicle owner’s (customer) details

4)When booking a parking slot user should fill in vehicle and customer details for the relevant vehicle.

5)When payment is made the system should be store customer details

6)Users should be able to pay for parking conveniently through various payment methods, including credit/debit cards, mobile payment apps, or pre-paid parking cards.

7)The system should provide secure and reliable payment processing to ensure user data and financial information are protected.

8)The payment details will be stored in the system.

9)A customer/user can visit the online automated parking system website/App.

10)A New user must register to the system and become registered to reserve the Parking slot.

11)Registered users can view the available parking slots.

12)Registered users can select and reserve the available parking slot.

13)When reserving a parking slot user should fill in vehicle details for the relevant vehicle.

14)Management can take reports on customers, vehicle and payment details.

15)The gates should be equipped with sensors to detect the presence of vehicles and prevent collisions.

16)The gates should be integrated with a reliable access control system to prevent unauthorized entry and exit.

17)The gates should be designed to accommodate high traffic volume during peak hours while maintaining optimal security and safety standards.

18)Staff members monitor the vehicle and parking space, maintain the vehicles if owner needs, handle the customer services and emergency.

19)Customer should enter the necessary personal information at the front gate.

20)In the front gate customer can get the parking slot number.

21)Customers can reserve a parking slot using mobile app, website or through customer service.

22)Unregistered customers should provide personal information relating to parking slot.

23)Unregistered customers should pay for parking slot at the exit gate.

24)Customers able to give feedback.

2) Classes

- Vehicle
- Payment
- Reservation
- Parking slot
- Entrance
- Exit
- Report
- Staff
- Unregistered customer
- Registered customer
- Customer
- feedback

CRC cards

Class name : vehicle	
Responsibility	**Collaborators**
Store information about vehicles	
Update vehicle information	
Booking relevant parking spaces	Parking slot
Register the vehicle	customer

Class name : payment	
Responsibility	**Collaborators**
Store payment details	
Provide reliable payment methods	
Generate the time period	
Add payment details	customer, vehicle
Calculate payment	reservation, customer, vehicle

Class name : reservation	
Responsibility	**Collaborators**
Reservation history managements	
Check and record availability of parking slots	Parking slot

Class name : parking slot	
Responsibility	**Collaborators**
Manage parking traffic	
Check reserved parking slot	reservation

Class name : entrance	
Responsibility	**Collaborators**
Record entrance time	
Check type of vehicle	vehicle
Check reservation ID	reservation

Class name : exit	
Responsibility	**Collaborators**
Record exit time	
Check exiting vehicles	vehicle
Send waited time to the system	payment
Calculate fee	payment

Class name : report	
Responsibility	**Collaborators**
Take report on customer details	
Take report on vehicle details	
Take report on payment details	
Take report on reservation details	

Class name : staff	
Responsibility	**Collaborators**
Monitor parking space	
Handle customer service	
Handle emergencies	
Handle report	report

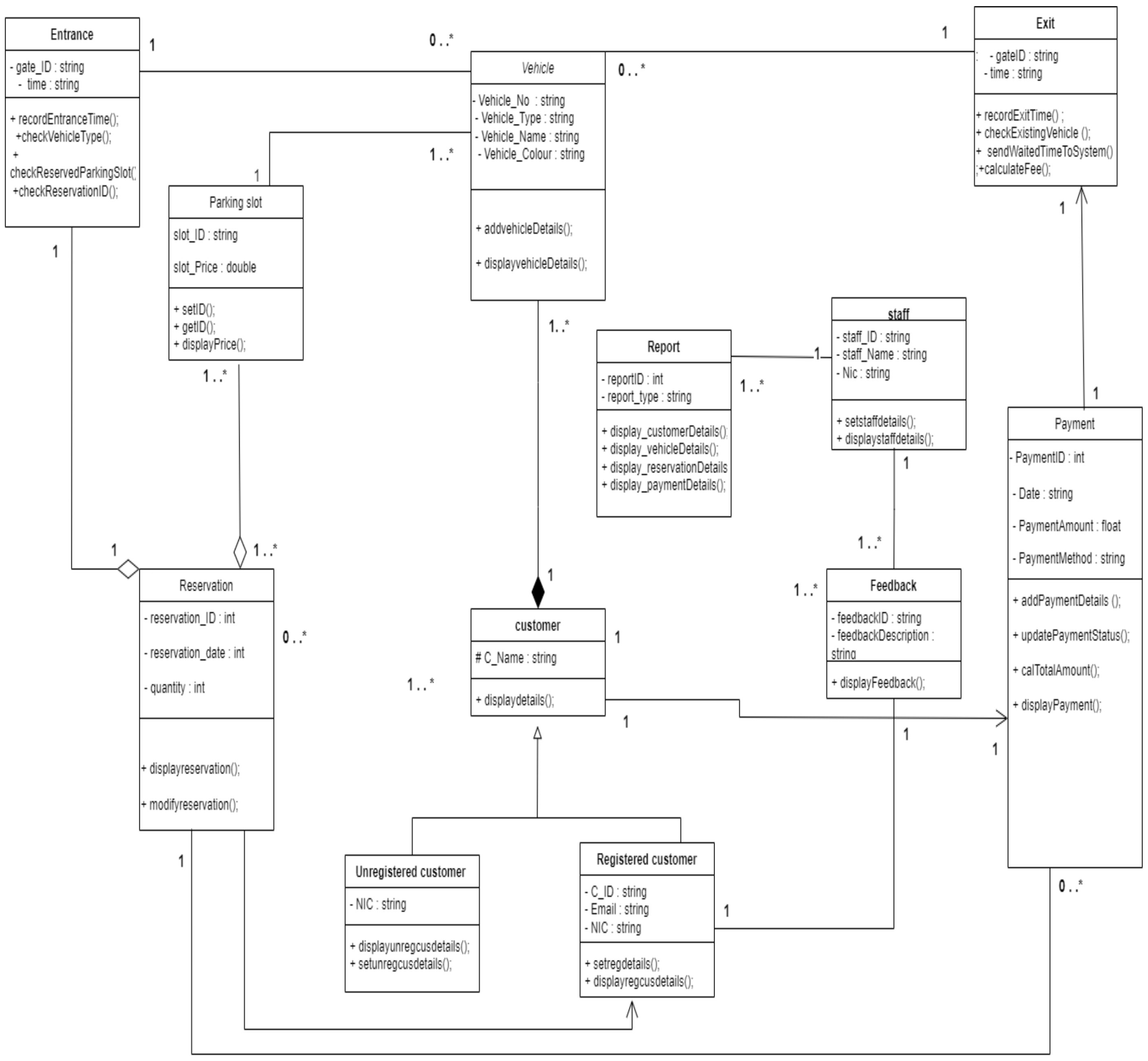
Class name : Customer	
Responsibility	**Collaborators**
Provide information	
Make payment	Payment
Reserve parking space	Reservation
Give feedback	Report

Class name : Registered customer	
Responsibility	**Collaborators**
Get parking slot	Parking slot
Get parking history	
Give feedback	Report

Class name : Unregistered customer	
Responsibility	**Collaborators**
Get parking slot	Parking slot
Make payment	Payment
Provide information	

Class name : feedback	
Responsibility	**Collaborators**
Get feedback	customer

Exercise 1 – CLASS DIAGRAM



3) Exercise 02 – CODE

```
 #include <iostream>
#define SIZE 2


using namespace std;



// Customer class


class Customer
{
private:
    Feedback* feedback[SIZE];
    Report* report[SIZE];




public:
    void addfeedback(Feedback *feed);

};



// Feedback Class
// T R Lakshan



class Feedback
{
private:
    string feedbackID;
    string feedbackDescription;
    Customer* customer;

public:
    Feedback();
    Feedback(string fID, string fdescription, Customer* cus);
    void displayFeedback();
};

Feedback::Feedback()
{
    feedbackID = "";
    feedbackDescription = "";
    customer = nullptr;
}

Feedback::Feedback(string fID, string fdescription, Customer* cus)
{
    feedbackID = fID;
    feedbackDescription = fdescription;
    customer = cus;
    customer->addfeedback();
…
[22:31, 14/06/2023] Chanika: #include <iostream>
#include <cstring>
#define SIZE 2

using namespace std;



//customer class
//Sahan W A G
```

```
class customer {
protected:
	string C_name;

private:
	vehicle* vec[SIZE];



public:
	customer();
	customer(string* cname);
	void setregdetails();
	void displaydetails();
	void vehicleinfo(vehicle* v);
};



//customer class implementation

customer::customer()
{
	C_name = "";
}
customer::customer(string* cname)
{
	C_name = *cname;
}
```

```
//registeredcustomer class
//Sahan W A G


class registered_Customer : public customer {
private:
        string C_ID;
        string Email;
        string NIC;
        Reservation* res[SIZE];
        Feedback* fback[SIZE];
public:
        registered_Customer();
        registered_Customer(string cname, string cid, string email, string nic);
        void setunregdetails(string cid, string email, string nic);
        void displayregcusdetails();
        void reservationinfo(Reservation* rese);
        void feedbackinfo(Feedback* feeback);
        ~registered_Customer();
};

//registeredcustomer class implementation

registered_Customer::registered_Customer()
{
        C_ID = "";
        Email = "";
        NIC = "";
}
registered_Customer::registered_Customer(string cname, string cid, string email, string nic)
```

```
{
        C_ID = cid;
        Email = email;
        NIC = nic;
}
void registered_Customer::setunregdetails(string cid, string email, string nic)
{

}
registered_Customer::~registered_Customer()
{
        cout << "registered customer details deleted!" << endl;
}




//Unregisteredcustomer class
//Sahan W A G


class Unregistered_Customer : public customer {
private:
        string NIC;
public:
        Unregistered_Customer();
        Unregistered_Customer(string c_name,string nic);
        void setunregcusdetails(string nic);
        void displayunregcusdetails();
        ~Unregistered_Customer();
};
```

```
//Unregisteredcustomer class implementation

Unregistered_Customer::Unregistered_Customer()
{
	NIC = "";
}
Unregistered_Customer::Unregistered_Customer(string nic)
{
	NIC = nic;
}
void Unregistered_Customer::setunregcusdetails(string nic)
{

}
Unregistered_Customer::~Unregistered_Customer()
{
	cout << "Unregistered customer details deleted!" << endl;
}




//staff class
//Sahan W A G


class staff {
private:
	string staff_ID;
	string staff_Name;
```

```
        string NIC;
        report* rpt[SIZE];
public:
        staff();
        staff(string sid, string sname, string Snic);
        void viewreport();
        void setstaffdetails(string sid, string sname, string Snic)
        void displaystaffdetails();
};

//staff class implementation
staff::staff()
{
        staff_ID = "";
        staff_Name = "";
        NIC = "";
}

staff::staff(string sid, string sname, string Snic)
{
        staff_ID = sid;
        staff_Name = sname;
        NIC = Snic;
}

void staff::setstaffdetails(string sid, string sname, string Snic)
{


}
```

```
// Declaring Feedback Class
// T R Lakshan



class Feedback
{
private:
    string feedbackID;
    string feedbackDescription;
    Customer* customer;

public:
    Feedback();
    Feedback(string fID, string fdescription, Customer* cus);
    void displayFeedback();
    ~Feedback();
};

Feedback::Feedback()
{
    feedbackID = "";
    feedbackDescription = "";
    customer = nullptr;
}

Feedback::Feedback(string fID, string fdescription, Customer* cus)
{
    feedbackID = fID;
    feedbackDescription = fdescription;
    customer = cus;
    customer->addfeedback();
}

void Feedback::displayFeedback(){}



//Declaring class Report
// T R Lakshan


class Report
{
private:
    int reportID;
    string report_type;

public:
    Report();
    Report(int reID, string reType);
    void display_customerDetails();
    void display_vehicleDetails();
    void display_reservationDetails();
    void display_paymentDetails();
    ~Report();
};

Report::Report()
{
    reportID = 0;
    report_type = "";
}
```

```
Report(int reID, string reType)
{
    reportID = reID;
    report_type = reType;
}

void Report::display_customerDetails(){};
void Report::display_paymentDetails(){};
void Report::display_reservationDetails(){};
void Report::display_vehicleDetails(){};

/*
// Customer class edit

class Customer
{
private:
    Feedback* feedback[SIZE];
    Report* report[SIZE];




public:
    void addfeedback(Feedback *feed);

};*/




//Declaring parking slot class
```

```
//M C G DEVINDA



Class Parking_Slot
{
private:
    string Slot_ID;
    double Slot_price;
    Vehicle *vehicle;

public:
    Parking_Slot();
    Parking_Slot(string slot_id, double slot_price);
    string getID();
    void setID();
    void displayPrice();
    ~Parking_Slot();
};

Parking_Slot::Parking_Slot()
{
    Slot_ID = "";
    Slot_price = 0.0;
}

Parking_Slot(string slot_id, double slot_price)
{
    slot_id = slot_id;
    Slot_price = slot_price;
}

void Parking_Slot::getID(){}
void Parking_Slot::setID(){}

void Parking_Slot::displayPrice(){}



/*

// vehicle class edit


class Vehicle

{

private:

   Parking_Slot * parkingslot[SIZE];

   Entrance * entrance[Size];


public:


};  */





//Declaring reservation class

//M C G DEVINDA



Class reservation;

{

private:

   int reservation_ID;

   string reservation_date;

   int quantity;

   Customer *mgr;

   Parking_Slot *parkingslots[SIZE];
```

```
    Payment *pay;
    Entrance *entrances[SIZE];

public:

    reservation();
    reservation(int Rid, string Rdate, int rquantity);
    void displayreservation();
    void modifyreservation();
    void add_parkingslot(Parking_Slot* s1, Parking_Slot* s2)
    {
        parkingslots[0] = s1;
        parkingslots[1] = s2;
    }


    void add_entrances(Entrance * e1, Entrance * e2)
    {
        entrances[0] = e1;
        entrances[1] = e2;
    }
    ~reservation();

};

reservation::reservation()
{
    reservation_ID = 0;
    booking_date = 0;
    quantity = 0;
}
```

```
reservation::reservation(int Rid, int Rdate, int Rquantity)
{
    reservation_ID = Rid;
    booking_date = Rdate;
    quantity = Rquantity;
}

void reservation::add_entrances(){}
void reservation::add_parkingslot(){}

/*
// Class Payment edit

class Payment
{
private:
    reservation * reservations[SIZE];
    Entrance * entraces[SIZE];

public:
    void addentrance(Entrance * p1, Entrance * p2)
    {
        entances[0] = p1;
        entances[1] = p2;
    }



};
*/
```

```
// Declaring class Entrance

// Rangana Wijesinghe



class Entrance

{

private:

    string gate_ID;

    string time;

    Vehicle * vehi_cle;


public:

    Entrance();

    Entrance(string gateid, string etime);

    void recordEntranceTime();

    void checkVehicleType();

    void checkReservedParkingSlot();

    void checkReservationID();

    ~Entrance();


};

Entrance::Entrance()

{

    gate_ID = "";

    time = "";

}


Entrance::Entrance(string gateid, string etime)

{

    gate_ID = gateid;

    time = etime;
```

```
}

void Entrance::checkReservationID(){}
void Entrance::checkReservedParkingSlot(){}
void Entrance::checkVehicleType(){}
void Entrance::recordEntranceTime(){}



// Declaring class Exit
// Rangana Wijesinghe




class Exit
{
private:
    string gate_ID;
    string time;

public:
    Exit();
    Exit(string gateid, string extime);
    void recordExitTime();
    void sendWaitedTime(int waitedTime);
    float calculateFee(int duration);
    ~Exit();

};

Exit::Exit()
{
    gate_ID = "";
    time = "";
}

Exit::Exit(string gateid, string extime)
{
    gate_ID = gateid;
    time = extime;
}

void Exit::recordExitTime(){}
void Exit::sendWaited(){}
float Exit::calculateFee(){}




//declaring Vehicle
//Kavisha Rajapaksha



class Vehicle
{
private:
            string Vehicle_No;
            string Vehicle_Type;
            string Vehicle_Name;
            string Vehicle_Colour;
            Parking_slot* parkslt;
            Entrance* Entrce;
            Exit* ext;
            Payment* payt;

public:
        Vehicle();

        Vehicle(string VehiNO, string Type, string Name, string Colour);

        void addVehicle();

        void displayVehicle();

};




Vehicle::Vehicle(){}


Vehicle::Vehicle(string VehiNO , string Type, string Name, string Colour)

{

        Vehicle_No = VehiNO;

        Vehicle_Type = Type;

        Vehicle_Name = Name;

        Vehicle_Colour = Colour;

}



void Vehicle::addVehicle(){}


void Vehicle::displayVehicle(){}







//declaring Payment

//Kavisha Rajapaksha



class Payment
```

```
{
private:
        int PaymentID;
        string date;
        float PaymentAmount;
        string PaymentMethod;
        Exit* ext;
        Reservation* resv;



public:
        Payment();
        Payment(int PID, string Pdate, float PAmount, string PMethod);
        void addPaymentDetails();
        void updatePaymentStatus();
        void calTotalAmount();
        void displayPayment();
};

Payment::Payment(){}

Payment::Payment(int PID,string Pdate, float PAmount, string PMethod)
{
        PaymentID = PID;
        date = Pdate;
        PaymentAmount = PAmount;
        PaymentMethod = PMethod;


}

void Payment::addPaymentDetails(){}
void Payment::updatePaymentStatus(){}

void Payment::calTotalAmount(){}

void Payment::displayPayment(){}




int main()
{
   //customer class objects
   //Sahan W A G


   string C_name = "jack";
         string C_ID = "C001";
         string Email = "Jack45@gmail.com";
         string NIC = "954358795v";
         registered_Customer* c001 = new registered_Customer(C_name, C_ID, Email, NIC);

         c001->displaydetails();
         c001->displayregcusdetails();
         c001->setregdetails();



         string c_name = "John";
         string NIC = "JohnW@gmail.com";
         Unregistered_Customer* c002 = new Unregistered_Customer(C_Name, NIC);

         c002->displaydetails();
         c002->displayunregcusdetails();
```

```
        c002->setunregcusdetails();



        //staff class objects
//Sahan W A G



string staff_ID = "S001";
        string staff_Name = "Julian";
        string NIC = "985469857V";
        staff* s001 = new staff(staff_ID, staff_Name, NIC);

        s001->displaystaffdetails();
        s001->setstaffdetails();



// Creating Feedback Class objects
// T R Lakshan



Feedback* F1 = new Feedback();
Feedback* F2 = new Feedback();



// Creating Report Class objects
// T R Lakshan



Report R1;
R1.Report(10, "Customer");
```

```
// Creating class Exit objects

// Rangana Wijesinghe



Exit ex1;

ex1.Exit("Gate1", "10.00");



// Creating class Entrance objects

// Rangana Wijesinghe



Entrance en1;

en1.Entrance("Gate2", "01.00");



// Creating reservation class objects

// M C G DEVINDA



reservation reserve1;

reserve1.reservation(30, "2023.06.18", 1);



// Creating Parking_Slot class objects

// M C G DEVINDA



Parking_Slot parkS1;

parkS1.Parking_Slot("S005", 300.00);
```

```
   //Vehicle class object
           //Kavisha Rajapaksha



           Vehicle vehicle1("000A", "Car", "BMW", "White");


           vehicle1.addVehicle();
           vehicle1.displayVehicle();


           //Payment class object
           //Kavisha Rajapaksha



           Payment payment1(1, "1/1/2023", 300.00, "cash");
           payment1.addPaymentDetails();
           payment1.updatePaymentStatus();
           payment1.calTotalAmount();
           payment1.displayPayment();
           delete c001;
           delete c002;
           delete s001;
 return 0;
}
```

No	IT number	Name	Contribution

1		T.RAVISHKA LAKSHAN	CRC Cards: Feedback, Report C++ Code: Feedback, Report UML Notation: Feedback, Report
2		RANGANA WIJESINGHE	CRC Cards: Entrance, Exit C++ Code: Entrance, Exit UML Notation: Entrance, Exit
3		GAVISHKA SAHAN	CRC Cards: Customer(Registered, Unregistered), Staff C++ Code: Customer(Registered, Unregistered), Staff UML Notation: Customer(Registered, Unregistered), Staff
4		KAVEESHA RAJAPAKSHA	CRC Cards: Payment, Vehicle C++ Code: Payment, Vehicle UML Notation: Payment, Vehicle
5		CHANIKA GAYASHAN	CRC Cards: Parking Slot, Reservation C++ Code: Parking Slot, Reservation UML Notation: Parking Slot, Reservation