



Concordia Institute for Information Systems Engineering (**CIISE**)

## System Design Specification Document

---

### Noble Team

*Danie(A)Zakerifar (40054463)*

**INSE 6260**

**Submitted To: Dr. Chun Wang**

## Table of Content

1	Overview .....	3
1.1	Goals and Objectives.....	3
1.2	Statement of Scope .....	3
1.3	Terminology & Definitions .....	3
1.3.1	Abbreviations .....	3
2	Functional Descriptions .....	4
3	Non-Functional Descriptions .....	5
3.1	Solution quality .....	5
3.2	Flexibility.....	5
3.3	Real time responsiveness .....	5
3.4	Reliability .....	5
4	System Architecture .....	6
5	Detailed Architecture .....	7
5.1	Process Scheduling .....	8
6	Detailed System Design .....	11
6.1	Graphic User Interfaces and a Basic Demo Scenario .....	11
6.2	Subsystem.....	13
6.3	Class Diagram .....	13
6.4	Sequence Diagram .....	16
6.5	Suggestions for Web Service Integration.....	16

## 1 Overview

### 1.1 Goals and Objectives

---

The purpose of system design specification documents is to completely describe the system at architecture level, including subsystems and their mapping, database mapping, interface design.

The design of the simulator should be done in order to satisfy following criteria:

1. Usability: The software simulator should be easy to use along with meeting all the customer requirements.
2. Efficiency: The software should be design considering the fact that computation part (#of chargers required, # of buses required, schedule of the buses) takes less time once the input is provided.

### 1.2 Statement of Scope

---

This software can fit to the any schedule. The constraints are that it can works for individuals' schedules and cannot consider the coverage of several schedule at the same time. But if we define some changes, we can improve this software to consider the whole schedule in the city.

The input is the schedule of East and West terminals. The output is the numbers of buses and chargers that we need in each terminals. It also produces the schedule for the each bus and charger.

The software simulates the process of traveling and charging to produce the outputs

### 1.3 Terminology & Definitions

---

#### 1.3.1 Abbreviations

HS: High Speed Charger

ON: Overnight Charger

## 2 Functional Descriptions

The Simulator software system is composed of the following basic components:

1. User Interface
2. Computational Part
3. Database

### User Interface

It will be a console based graphical user application that will take inputs from users like manufacture of charger, battery size of bus, cost of bus, chargers. Once input is provided by the user it will display the result accordingly.

### Computational Part

In this part, algorithm of our simulator will calculate number of buses required, number of chargers of each type at two terminus and suspected budget to complete the electrification of 211 bus route.

### Database

The database used as part of the software. Bus transit schedule, User inputs are stored in database.

### **3 Non-Functional Descriptions**

Software structural quality refers to how it meets non-functional requirements that support the delivery of the functional requirements, such as robustness or maintainability. It has a lot more to do with the degree to which the software works as needed.

#### **3.1 Solution quality**

---

The software should **show** us the number of buses that satisfy all the time slots in the schedule. The solution is for one day schedule trip.

#### **3.2 Flexibility**

---

Software should be able to fit with all the schedule from the other routs. For here we are just using rout 211 but we can adjust to any routs.

It is also compatible with every schedule. The only thing that we have to do is putting the schedule of bus in the file for reading through the software.

#### **3.3 Real time responsiveness**

---

For Windows 10 and Java Application, 8 gigs ram it should take 10 seconds. No application should be open, when they are using this application.

#### **3.4 Reliability**

---

If the user input the data according to the options that it has, the software should work properly. If the user input wrong data it should run it again.

## 4 System Architecture

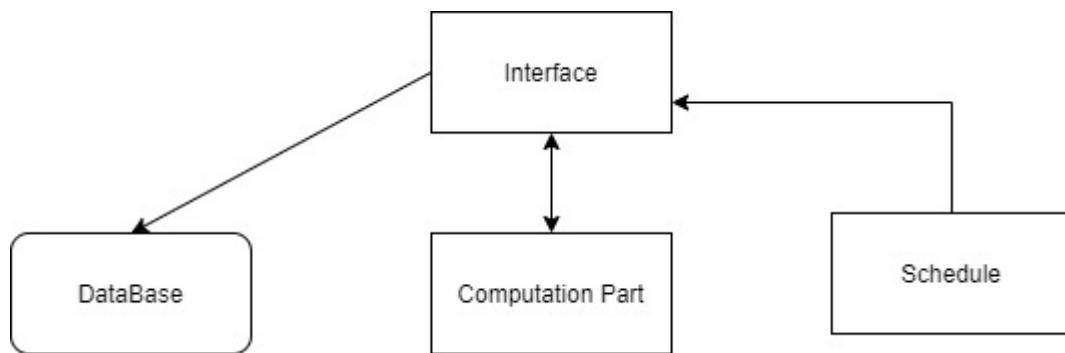


Figure 1 System Architecture

## 5 Detailed Architecture

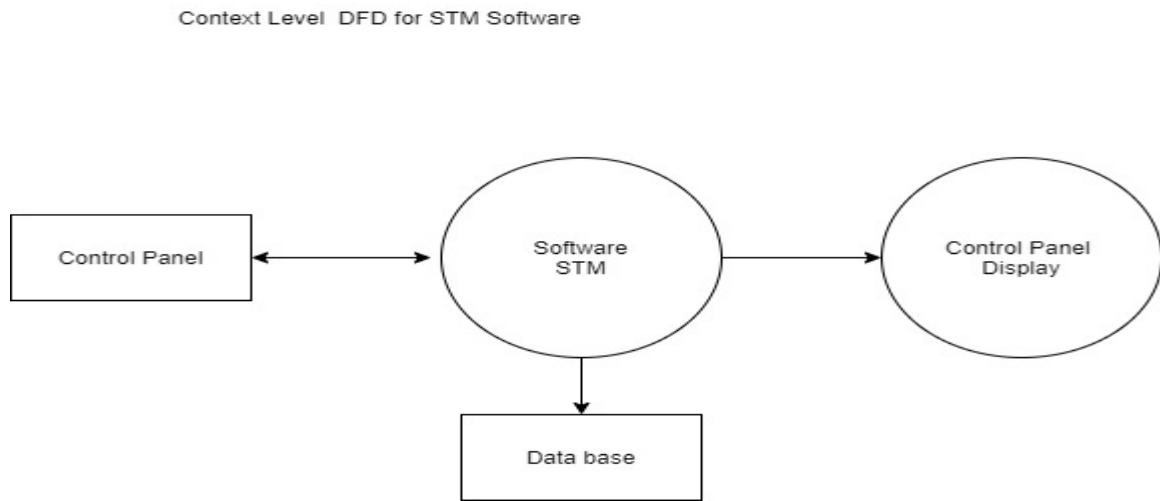


Figure 2 Context Level DFD STM Software

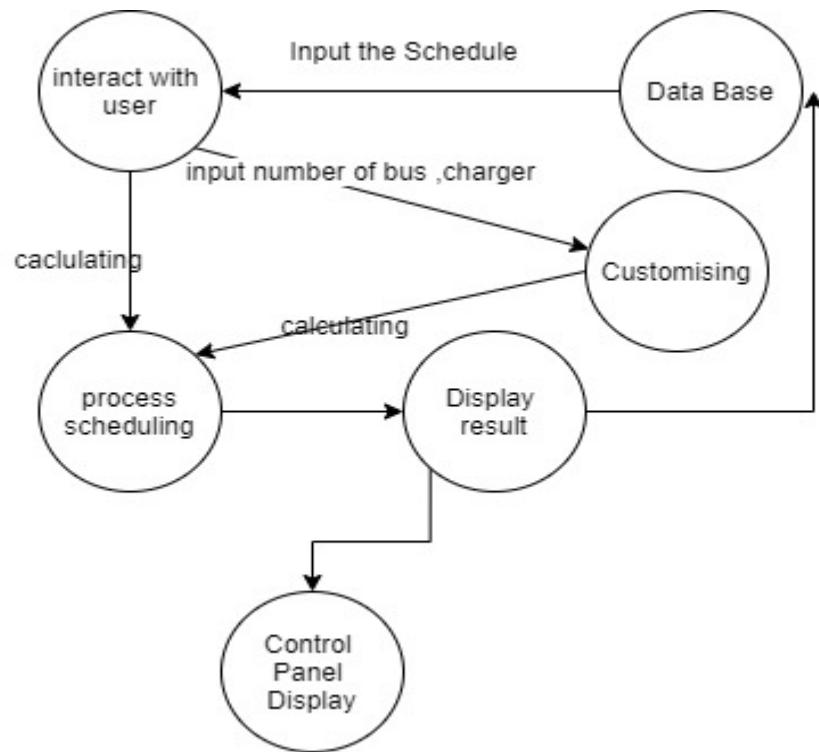
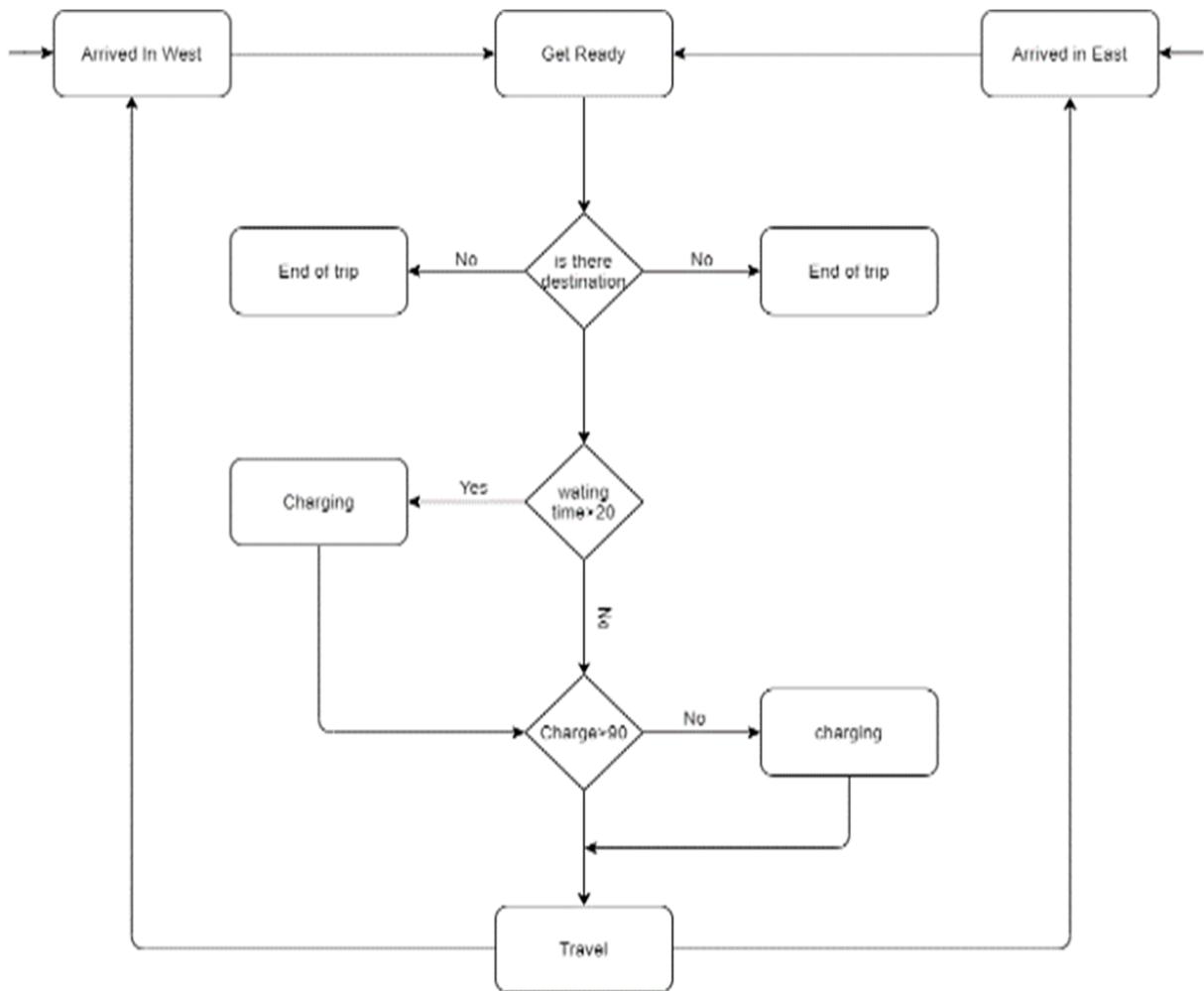


Figure 3: Level 1 DFD for STM Software

## 5.1 Process Scheduling

---

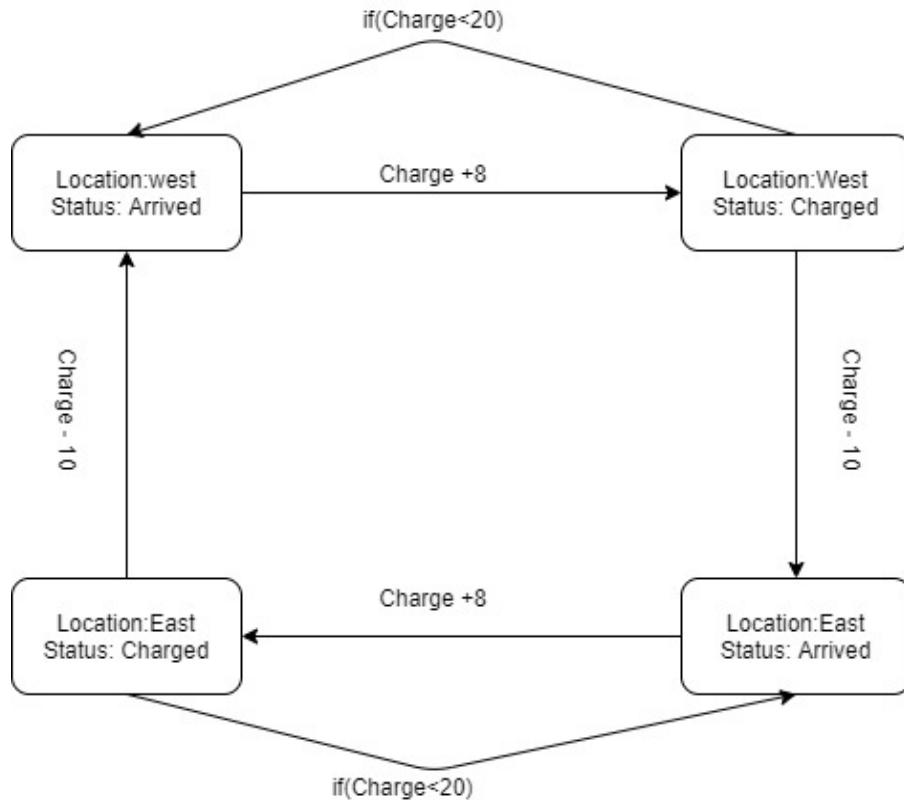
At first, we must find the algorithm. The algorithm that we are using is the following:



The formula for charging the battery of bus is,  $P(\text{kwh}) = E(\text{kw})T(\text{h})$

Where  $P(\text{kwh})$  = Increased battery level after charging,  
 $E(\text{kw})$  = Power of charger,  
 $T(\text{h})$  = Time spent for charging

The model of algorithm is given below.



## Policies for our software:

1. When to charge the bus?
  - if, (waiting time < 20 minutes & battery remaining > 90) then, no charging required
  - If, (waiting time >20 minutes & battery remaining >90) then, charge using high speed charger
  - If, (battery remaining <90) then, charge using low speed charger
2. Charging Time Required by slow speed charger  
At least 30 minutes
3. Charging Time Required by high speed charger  
At most 1 minutes
4. What is the charging rate of chargers?
  - ABB = 1 minute of charging will charge the battery to 7units
  - HELIOX = 1 minute of charging will charge the battery to 5 units
5. What is the utilization of bus battery?  
1km of distance will utilizes 1kwh of energy

The following are our requirements (numbers are in percent):

1. The charge of the battery should be always more than zero. This is because whenever the charge of battery become zero, it cannot move anymore. The formal language for that is  $(G(\text{StateOfCharging} > 0))$
2. Moreover, to make sure that the bus has enough energy to reach charger station, we have to define the limit for that. For doing that we consider that limit as the amount of energy needed for reaching from one station to another station. We consider that limit equals 10. So, the charge of battery should always be more than 10. The formal language for that is  $\neg(F(x < 10))$

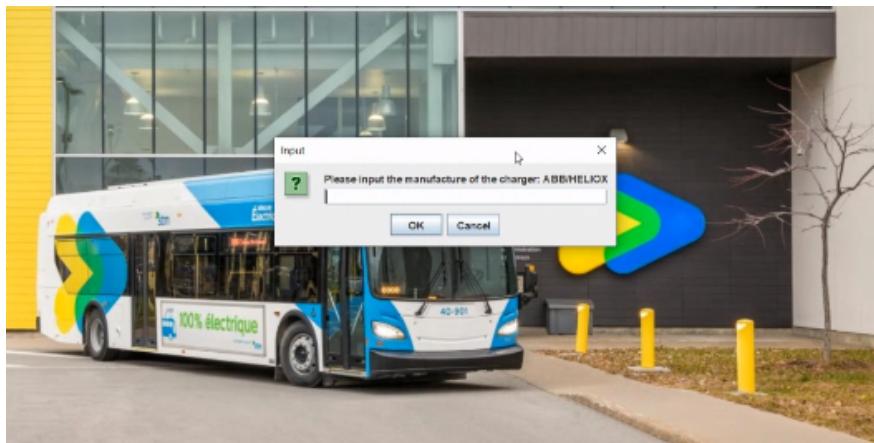
3. The energy consuming for one trip is 10 so for the condition that energy more than 10, we can have a trip with bus when it has at least 20. When is 20, after a one trip the energy become 10. So, it must go charging when it is 20. So the charge of battery often goes less than 20 but it always more than 10. The formal language for that is  $(GF(x < 20))$
4. The charge of the battery cannot go more than the capacity of battery which is 100. The formal language for that is  $F(StateOfCharger > 100)$  is false.
5. We cannot move anywhere until we have sufficient charging. The formal language for that is  $check\_ItIsSpec - p"!(status != charged \cup status = arrived)"$  Is true
6. The bus cannot be in two stations at the same time. The formal language to represent this is,  $!( location == west \text{ and } location == east )$
7. Whenever the charge of the battery become less than 20, it continue charging until it increase energy to 20. The formal language to represent this is,  $G(stateOfCharging < 20 \text{ and } status = charging) \rightarrow XX status = charging$
8. When charging become equals to 100, the charging stops. The formal language to represent this is  $(G(StateOfCharging == 100 \rightarrow b.charging.stops))$
9. The bus cannot arrive anywhere if does not move.
10. When bus arrived, there is no possibility that it stops for ever.

Then we used a model checker like NUSMV to check the our requirements are satisfied. After checking some time and improving model eventually we made an algorithm that satisfied our problem.

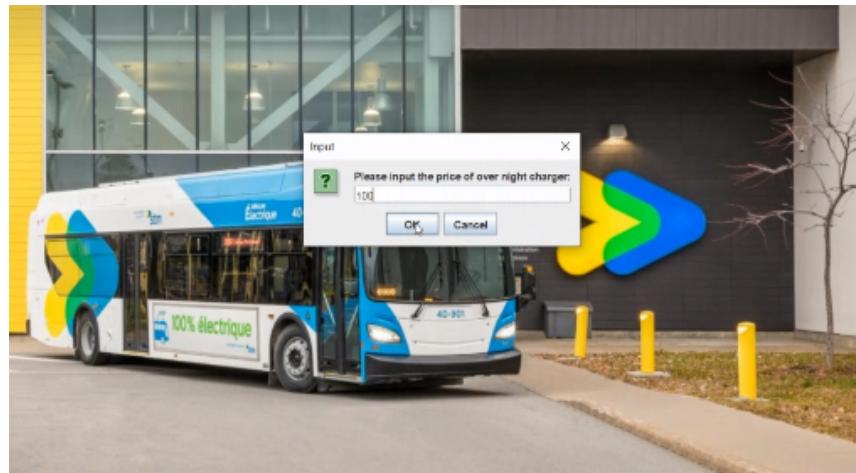
## 6 Detailed System Design

### 6.1 Graphic User Interfaces and a Basic Demo Scenario

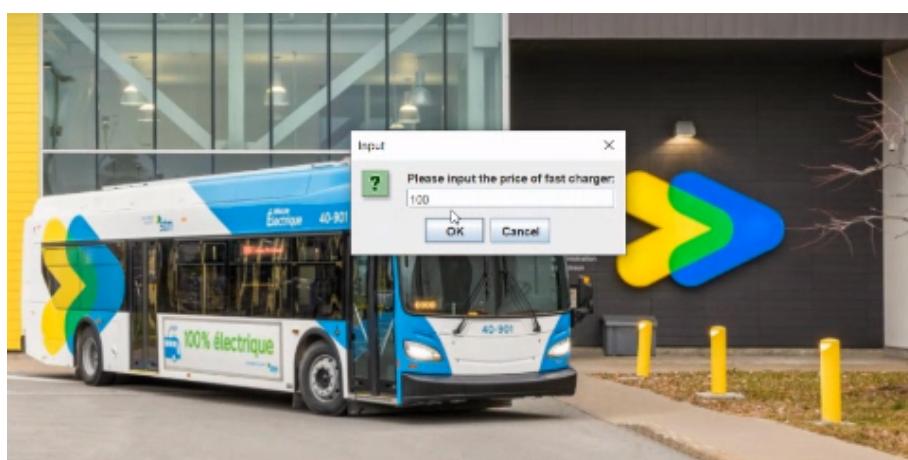
At first it asked user to input manufacture of Chargers between ABB and HELIOX



Then it asked user to input the price of overnight charger



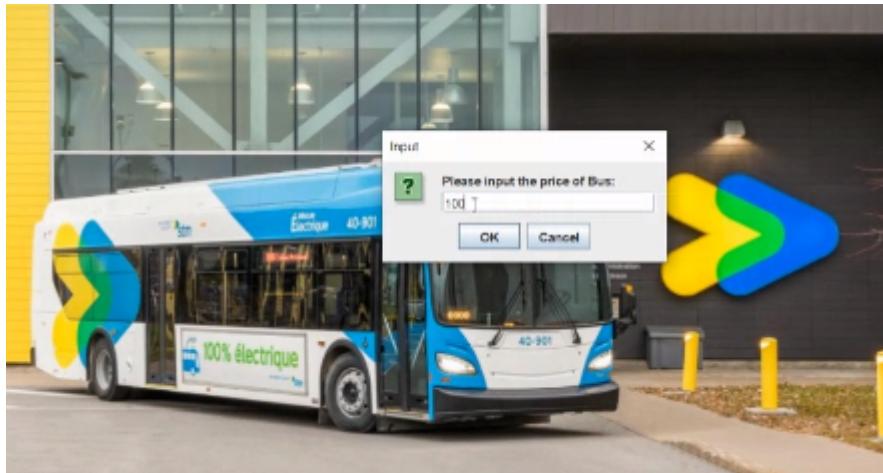
In the next step it asked user to input the price of fast chargers



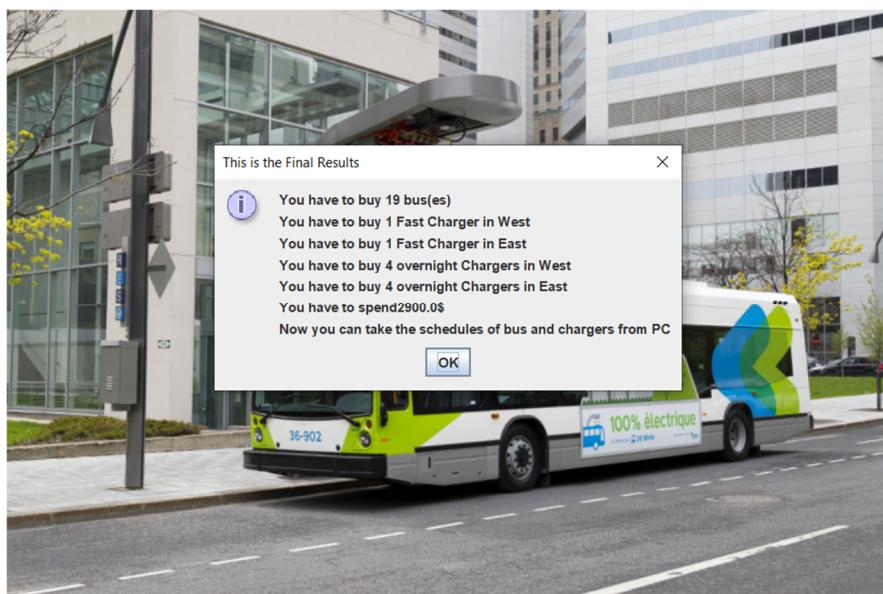
Then it ask user for the kind of bus.



Then it asked user to input the price of bus



And eventually it is going to show how many buses, and charger you need. It also say about the amount of money that you have to spend for that.



## 6.2 Subsystem

Devices

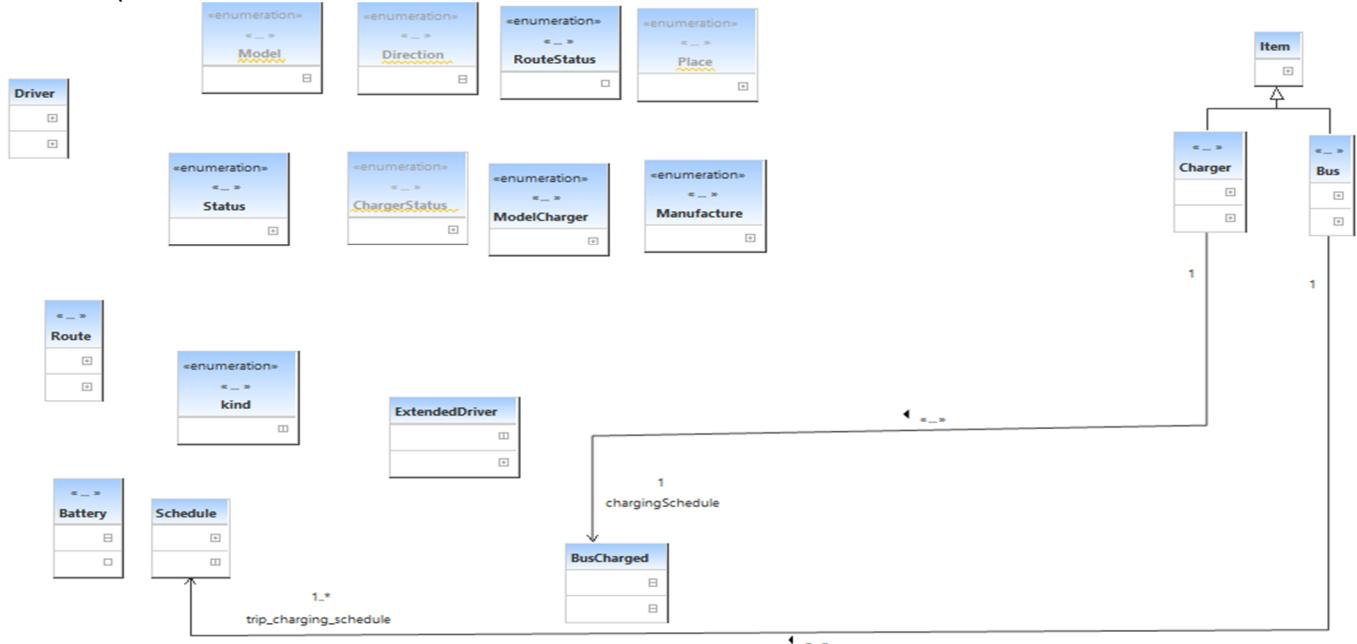
Property

Interconnection

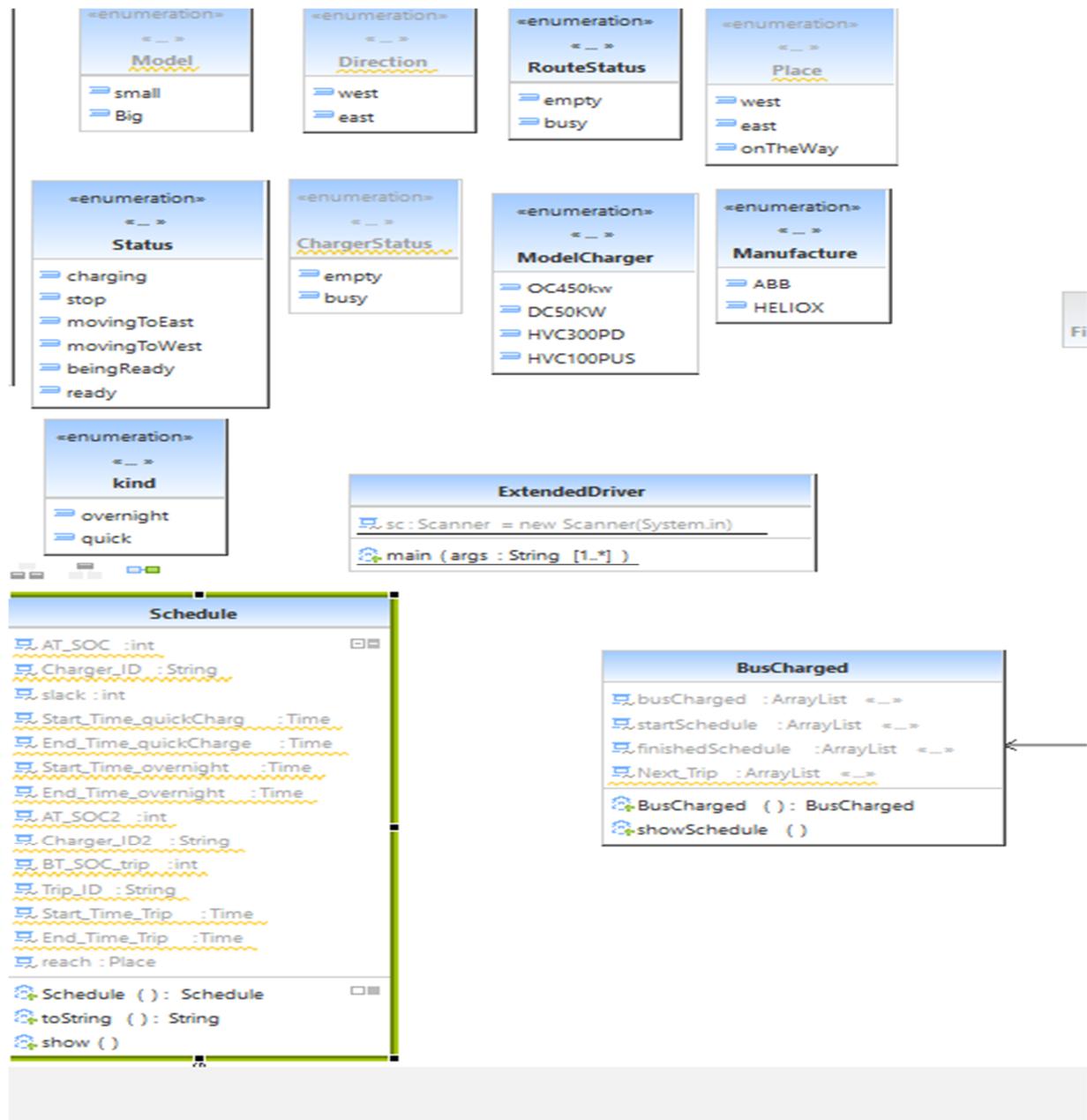
## 6.3 Class Diagram

The following picture is the diagram of all class.

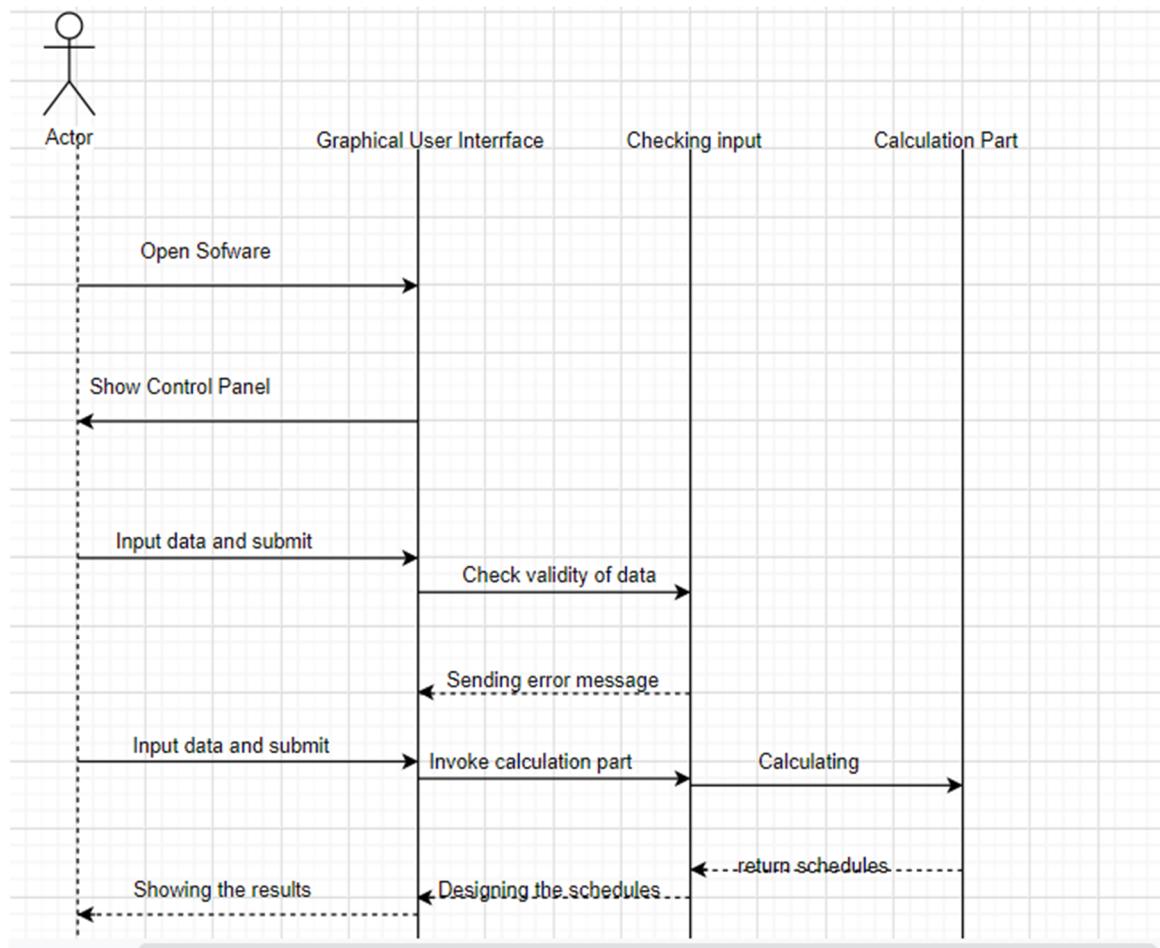
The next picture are the classes with their attribut and methods.



<pre> &lt; ... &gt; Charger <b><u>overnight :ModelCharger = ModelCharger.DC50KW</u></b> <b><u>priceOfOvernightCharger :double =_&gt;</u></b> <b><u>fast :ModelCharger = ModelCharger.DC50KW</u></b> <b><u>ID:int</u></b> <b><u>westNumberOfOverNightCharging :int = 0 &lt;=_&gt;</u></b> <b><u>power :double</u></b> <b><u>model :ModelCharger</u></b> <b><u>manufacture :Manufacture</u></b> <b><u>id :String</u></b> <b><u>place :Place</u></b> <b><u>status :ChargeStatus</u></b> <b><u>chargerSchedule :ArrayList&lt;ArrayList&lt;ArrayList&lt;Time&gt;&gt; = new ArrayList&lt;ArrayList&lt;ArrayList&lt;Time&gt;&gt;()</u></b> <b><u>busCharged :ArrayList&lt;new ArrayList&lt;&gt;() &lt;=_&gt;</u></b> <b><u>schedule :ArrayList&lt;new ArrayList&lt;Time&gt;() &lt;=_&gt;</u></b> <b><u>startOfCharging :Time &lt;=_&gt;</u></b> <b><u>scheduleWest :ArrayList&lt;new ArrayList&lt;Time&gt;() &lt;=_&gt;</u></b> <b><u>scheduleEast :ArrayList&lt;new ArrayList&lt;Time&gt;() &lt;=_&gt;</u></b> <b><u>busList :Bus [1..*] &lt;=_&gt;</u></b> <b><u>priceOfFastCharger :double &lt;=_&gt;</u></b> <b><u>eastNumberOfFastCharging :int = 0 &lt;=_&gt;</u></b> <b><u>eastNumberOfOverNightCharging :int = 0 &lt;=_&gt;</u></b> <b><u>westNumberOfFastCharging :int = 0 &lt;=_&gt;</u></b> <b><u>endOfCharging :Time &lt;=_&gt;</u></b> <b><u>Charger () : Charger</u></b> <b><u>getModel () : ModelCharger</u></b> <b><u>getPower () : double</u></b> <b><u>setPower (p : double) : void</u></b> <b><u>setId (id : String) : void</u></b> <b><u>toString () : String</u></b> <b><u>equals (otherObject : Object) : boolean</u></b> </pre>	<pre> &lt; ... &gt; Route <b><u>ID :int</u></b> <b><u>time :Time</u></b> <b><u>direction :Direction</u></b> <b><u>Id :String</u></b> <b><u>status :RouteStatus</u></b> <b><u>bus :Bus</u></b> <b><u>distance :int = 40</u></b> <b><u>place :Place</u></b> <b><u>busList :Bus [1..*] &lt;=_&gt;</u></b> <b><u>stateOfChargingBeforeTrip :double &lt;=_&gt;</u></b> <b><u>stateOfChargingAfterTrip :double &lt;=_&gt;</u></b> <b><u>makeItBusy ()</u></b> <b><u>toString () : String</u></b> <b><u>Route (t:Time) : Route &lt;=_&gt;</u></b> <b><u>Route (t:Time , s: Direction) : Route &lt;=_&gt;</u></b> <b><u>Route (r:Route) : Route &lt;=_&gt;</u></b> <b><u>equals (otherObject : Object) : boolean</u></b> </pre>	<pre> &lt; ... &gt; Bus <b><u>IDB :int = 0</u></b> <b><u>travelingTime :Time = new Time(01,00,00)</u></b> <b><u>consuming :double = 40</u></b> <b><u>timeBeingReady :Time = new Time(00,06,00)</u></b> <b><u>startOfCharging :Time &lt;=_&gt;</u></b> <b><u>departureTime :Time &lt;=_&gt;</u></b> <b><u>normalWaitingTime :Time &lt;=_&gt;</u></b> <b><u>schedule :Route [1..*] &lt;=_&gt;</u></b> <b><u>start :Time</u></b> <b><u>battery :Battery</u></b> <b><u>status :Status</u></b> <b><u>id :String</u></b> <b><u>place :Place</u></b> <b><u>consumingperkm :double = consuming/Route.distance</u></b> <b><u>waitingTime :int</u></b> <b><u>limit :double = 50</u></b> <b><u>endOfCharging :Time &lt;=_&gt;</u></b> <b><u>nextDepartureTime :Time &lt;=_&gt;</u></b> <b><u>arrivalTime :Time &lt;=_&gt;</u></b> <b><u>busready :Time &lt;=_&gt;</u></b> <b><u>EndingTime :Time &lt;=_&gt;</u></b> <b><u>Bus () : Bus</u></b> <b><u>beingReady ()</u></b> <b><u>travelingEmpty ()</u></b> <b><u>toString () : String</u></b> <b><u>ToString1 () : String</u></b> <b><u>Bus (s:Place) : Bus</u></b> <b><u>Bus (m:Model , d:double , p:Place , t:Time) : Bus</u></b> <b><u>Bus (b:Bus) : Bus</u></b> <b><u>traveling (r:Route)</u></b> <b><u>quickCharging (ch:Charger , ac:Time , nextTrip :Route)</u></b> <b><u>overNightCharging (ch:Charger , ac:Time)</u></b> <b><u>periodInMinute (t1:Time , t2:Time) : int</u></b> </pre>
<p style="text-align: center;"><b>Item</b></p> <pre> <b><u>price :double</u></b> <b><u>place :Place</u></b> </pre>		



## 6.4 Sequence Diagram



## 6.5 Suggestions for Web Service Integration

We can use JavaScript to connect this software with web application.