Electric Vehicle Reservation System



**Software Design Document**

**for**

**EV Charging Network Reservation System**

Prepared by: Navjot Singh

40014477

Concordia University

16-03-2018

**Table of Content**

1 Overview 3

1.1 Purpose 3

1.2 Intended Audience and Reading Suggestions 3

1.3 Project Scope 3

1.4 References 3

2 Functional Descriptions 4

2.1 Operating Environment 4

2.2 Design and Implementation Constraints 5

2.3 Assumptions and Dependencies 5

3 Non-Functional Descriptions 6

3.1 Performance Requirements 6

3.2 Security Requirements 6

3.3 Software Quality Attributes 7

4 Detailed Design and Architecture 8

4.1 System Software Architecture 8

4.2 Design 8

4.3 Data Flow Diagram 10

5 Graphical User Interface 11

6 Class Diagram 13

7 Sequence Diagram 14

8 State Machine Diagram 15

9 Use Case Diagram 16

# Overview

## Purpose

The purpose of this document is to define and communicate the software requirement of “Electric Vehicle Reservation System”. The requirements are documented to means to provide a common understanding of stakeholders. The requirements will be verified through reviews. Diagrams and detailed use cases it contains apply UML notations and description.

## Intended Audience and Reading Suggestions

This document is for all the stakeholders (Developers, end-users, project manager, etc.). It provides the well-structured detailed documentation of architecture in well-mannered standard format, which assists them to understand what an application need to deliver. For more understanding refer to the reference section which includes all the details of various papers or reports that are used for preparing this document.

## Project Scope

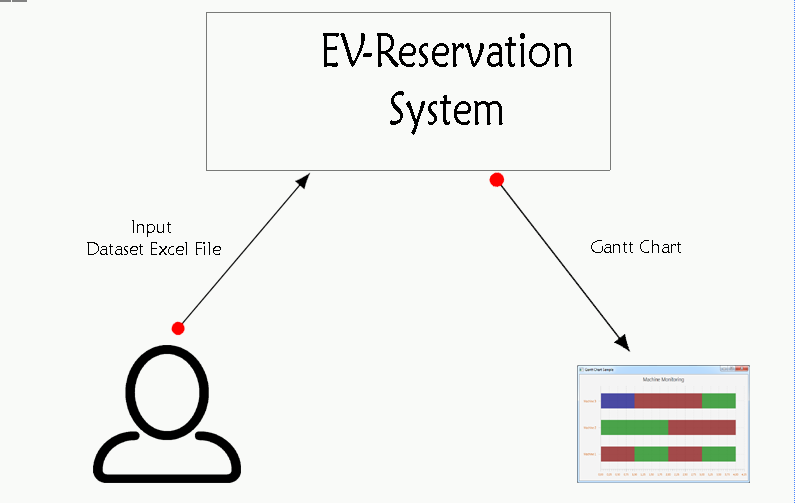
Electric Vehicle Reservation is an interactive standalone application with graphical interface. The goal is to develop a reservation system using the software engineering processes. Application will be tested with various test cases and techniques. Major purpose is of application is effective utilization of charging stations with respect to the customers. Application does not need any special expertise or skills.

## References

* <https://www.uml-diagrams.org/use-case-diagrams.html>
* <https://www.projectsmart.co.uk/requirements-gathering.php>
* <https://www.mockplus.com/blog/post/user-persona-template>
* <https://xtensio.com/user-persona/>
* <https://en.wikipedia.org/wiki/Charging_station>
* <https://pluginamerica.org/understanding-electric-vehicle-charging/>
* <https://www.draw.io/>

# Functional Descriptions

* Utilization of different stations charging points.
* Electric vehicle user will ask to reserve a charging point at one station which provides their compatible type of charger.
* Each station has only one type of charging points
* User will give their time window in which they want to charge their vehicle and distance that they want to travel after charging
* System will calculate the time required to charge the vehicle based on the given distance that user want to travel

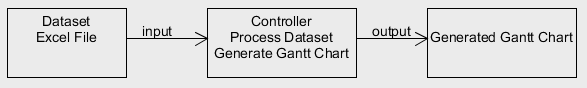


**Different Constraints:**

1. Charging point is selected based on the Type of Charger
2. System will reserve time slot if it is available in user given time window
3. If **time needed to charge is > time window**, system will reserve the full window
4. Reserved slot can be modified to utilize the charging point but it must be managed in effective manner so that maximum number of user can get time slot in their given window
5. Rejection of vehicle request is based on their given window, System will try to find a charging point if there is no availability even after modifying the assigned vehicles then System will reject that vehicle.
6. Finally, system assigned all the charging points and generates the Gantt chart which is outcome of system.

## Operating Environment

Electric Vehicle Reservation System is a standalone application, which is developed in java8 programming language. It is a window based application which does not require any special environment.



## Design and Implementation Constraints

Application does not have any implementation or design related constraints. It has one major constraint which is related to the output format.

Output must be Gantt chart which will show the reserved station points.

## Assumptions and Dependencies

* Electric Vehicle recharge reservation system has two library dependencies.

1. For processing Excel file: Apache POI
2. For Gantt chart generation: JFree Chart.

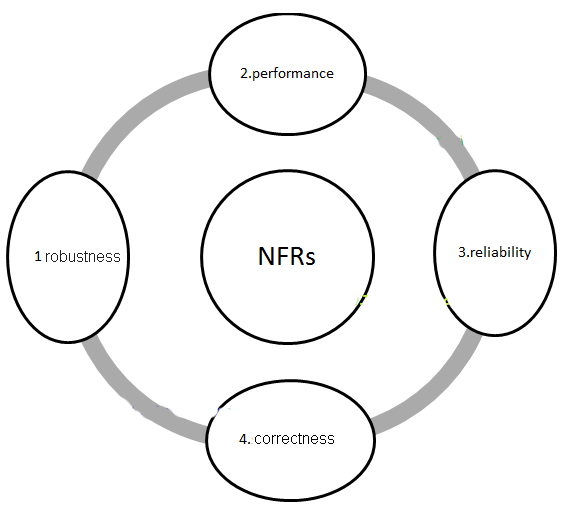
# Non-Functional Descriptions

## Performance Requirements

Application must be responsive in terms of number of vehicles at time need to reserve a charging point at ant station based on their level of charger.

## Security Requirements

Reservation information must be processed and stores in secured way so that during the updating of different time slots later will not break the system’s performance as well as anyone can not able to update the others reserved slots.



## Software Quality Attributes

* **Availability:**

System must be available all the time so that whenever any user need to reserve any timeslot they can reserve it.

* **Correctness:**

It basically related to the output part. After processing the user request in our case, it will be Dataset in the form of excel file system always generate effective as well as correct results in the form of Gantt Chart.

* **Flexibility:**

Application supposed to be extendible in a sense that may be later needing to integrate it with other components or adding new feature to it does not generate any issues.

* **Reliability:**

It refers to the failure-free software at selected interval of time. Electric Vehicle Reservation system should be reliable at any point of time so that end users able to reserve a time slot at any time.

* **Robustness:**

There are some cases when use give erroneous input to the system so system must able to handle those cases in such a way that which assists users to get back to the right place and system will perform as expected by the user.

* **Testability:**

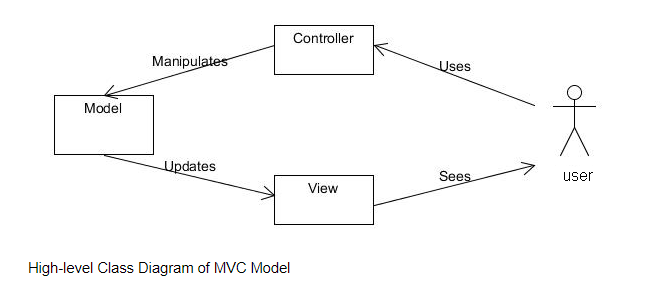
System should be testable so that before deliver to the end user it is validated and verified with numerous test cases. Testability refers to make the product viable solution. For requirement documentation there is review stage in which testing of requirements helps to identify the errors in requirement document.

# Detailed Design and Architecture

In this section, describe the system and/or subsystem(s) architecture for the project.

## System Software Architecture

In this section, describe the overall system software and organization. Include a list of software modules (functions, classes), computer languages, and programming computer-aided software engineering tools. Use structured object-oriented diagrams that show the various segmentation levels down to the lowest level. Include a narrative that expands on and enhances the understanding of the functional breakdown.

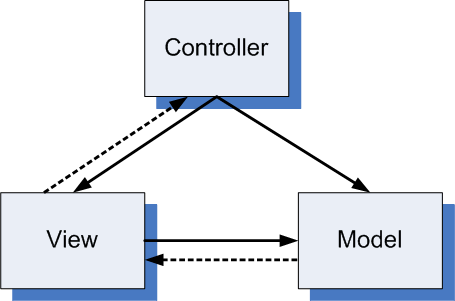


## Design

**Architectural design**

The architecture taken for Electric Vehicle Reservation system is the Model View Controller model. The MVC architecture is consists of 3 separate components, called the model, view and controller.

The **model** is base of the game where the game’s state and player data is managed. All the computations that are performed during the game are done in the model component as well as all data that needs to be processed. Moreover, any step to change the game’s state is also handled by the model. Whenever there are changes made to the model, the model updates the view automatically.



The **view** is the graphical user interface (GUI) of the game and displays the data from the model. Whenever the model makes any change, the view update itself. The view also gets updated by the controller, as the controller conduct simple data verification on user input and updates the view in the form of a pop-up if any errors are find out during any operation. Different exercises of the view can be developed to present the same data in different manners.

The **controller** is the component that players use to interact with the game. All players data is captured through the controller, which it then passes to the model. The controller also translates all mouse clicks or game events and it decide which part of the model needs to be manipulated. Basic all input validations are performed by controller, which updates the view if an error is detected.

The MVC architecture gives the three components to be developed differently from one another and they can be done in parallel. At any time, any component can be updated without affecting the other components if their application program interface stays the same. With a common API, the 3 models can be integrated into one game and different versions of the models can be combined in the game. This architecture gives facility for many different versions of views to be developed, which could be used to make variations of the game based on different themes.

**Major Modules:**

* Dataset (Excel File)
* Gantt Chart (Output)

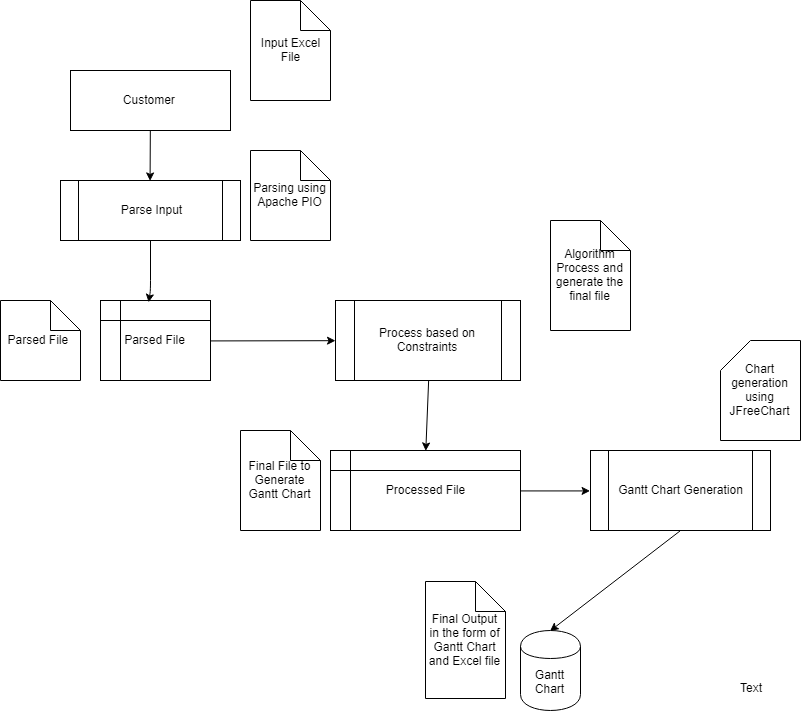
**Major Classes:**

* Vehicle
* Chart
* Controller

**External Interfaces:**

* Apache POI
* JFreeChart

## Data Flow Diagram

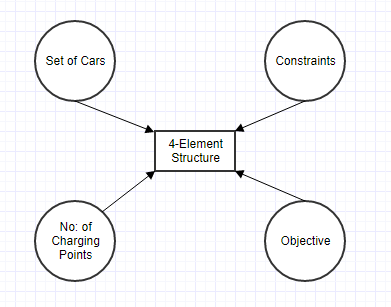
******

## Algo approach for Problem

After careful study of the problem I concluded that problem comes under the category of **partial overlapped system**.

**Major Goals:**

* Maximize the utilization of charging points



Implementing this problem with partial overlapping approach is a best approach for effective allocation of charging points.

But this is needs an optimization algorithm to managing the transfer rate after initial allocation. Due to that I have another approach to implement first basic allocation and after integrating optimization to the solution.

**Initial Approach:**

Implementation based on some rules:

* Finding set of requests for common time window which is based on common charging point.
* Allocating charging point to request which is having maximum need of point, which leads to minimize transfer.
* Random selection if there is condition of two requests with same parameters and not enough charging station or point available.

**Final** **Algorithm**

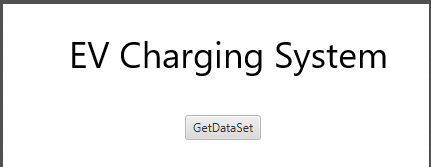
**Steps:**

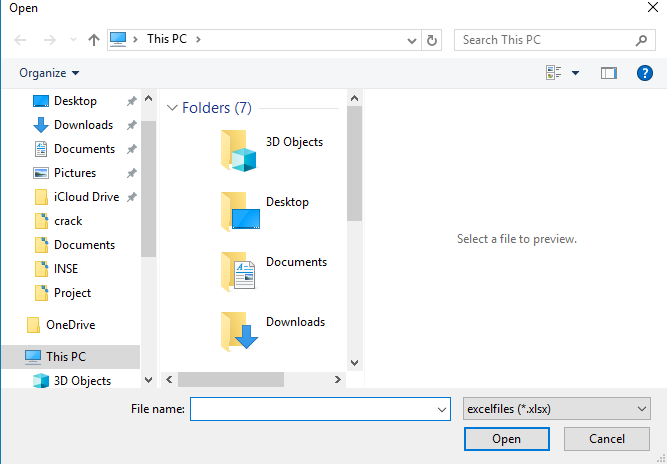
1. **Filter the valid client requests (based on duration)**
2. **Lists of Car based on type (Tesla, Chev, Nissan)**
3. **Sorting of all based on Miles (reversed-highest) and then based on Prefer Start Time**
4. **Processing all for best option (All tesla first goes to Supercharger)**
5. **Output generated on Gantt Chart**
6. **All the valid processed requests stored in excel file.**

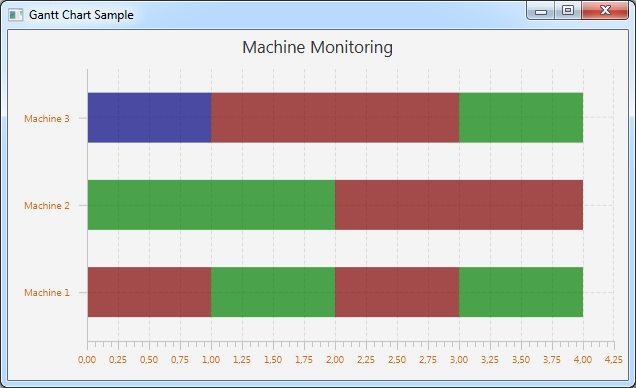
# Graphical User Interface

These are the main user interfaces of Electric Vehicle Reservation System.

1. System will ask user to input an excel file



1. File open Dialogue assists users to choose a file
2. Gantt chart is the final output of system



# Class Diagram

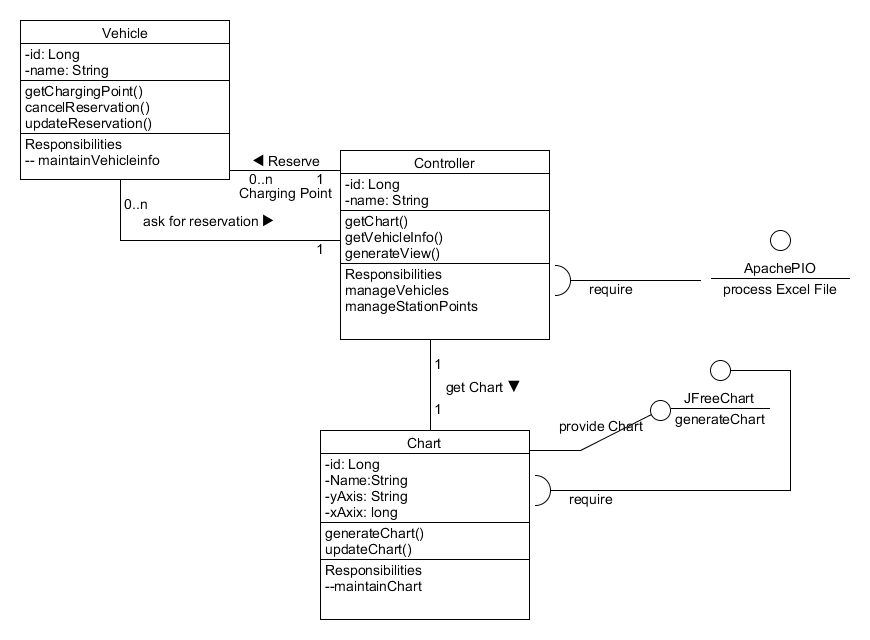
Class Diagram for Electric Vehicle Reservation System

**Major Classes:**

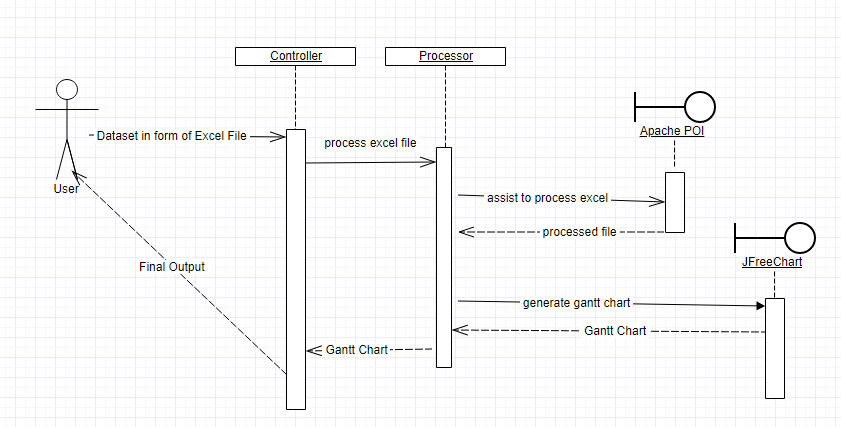
* Vehicle
* Chart
* Controller

**Major Interfaces:**

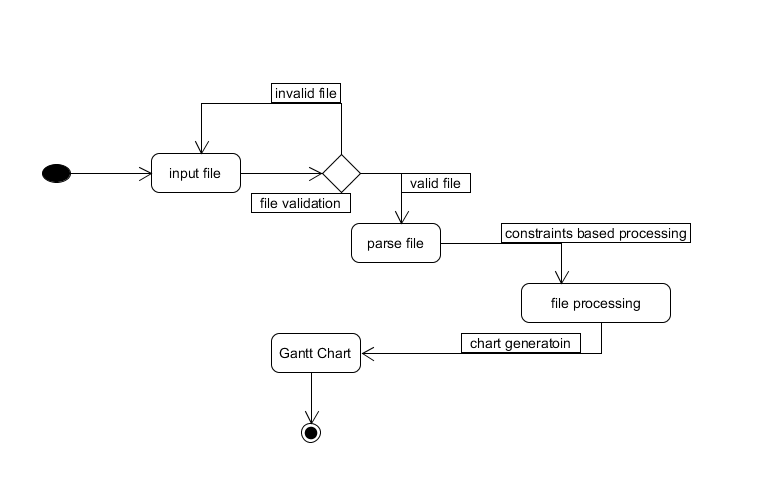
* Apache PIO
* JFreeChart



# Sequence Diagram



# State Machine Diagram



# Use Case Diagram

