# Analysis

# Requirement

## MoSCow Prioritization

**Functional requirement:** The requirements of a system that are implementation within the system and we can actually use the corresponding functionalities is termed as functional requirement.

**Non-functional requirement:** The requirements of a system that are implemented as the behavior of the system over system functionalities is termed as non-functional requirement.

|  |  |  |  |
| --- | --- | --- | --- |
| **SN** | **Requirements** | **Functional**  **/Non-functional** | **MoSCow** |
| 1. | Login and Registration | Functional | Must have |
| 2. | View location | Functional | Must have |
| 3. | Hire vehicle | Functional | Should have |
| 4. | Rent vehicle | Functional | Should have |
| 5. | View fuel status | Functional | Should have |
| 6. | Add staff | Functional | Must have |
| 7. | Update staff | Functional | Must have |
| 8. | Delete staff | Functional | Must have |
| 9. | View users | Functional | Must have |
| 10. | Delete users | Functional | Must have |
| 11. | Efficiency | Non-functional | Must have |
| 12. | Security | Non-functional | Must have |
| 13. | Smooth performance | Non-functional | Could have |
| 14. | Reliable | Non-functional | Should have |
| 15. | Portability | Non-functional | Should have |

Table-1: Moscow prioritization

# Use-Case diagram

## Introduction

A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. In this context, the term "system" refers to something being developed or operated. System objectives can include planning overall requirements, validating a hardware design, testing and debugging a software product under development, creating an online help reference, or performing a consumer-service-oriented task.

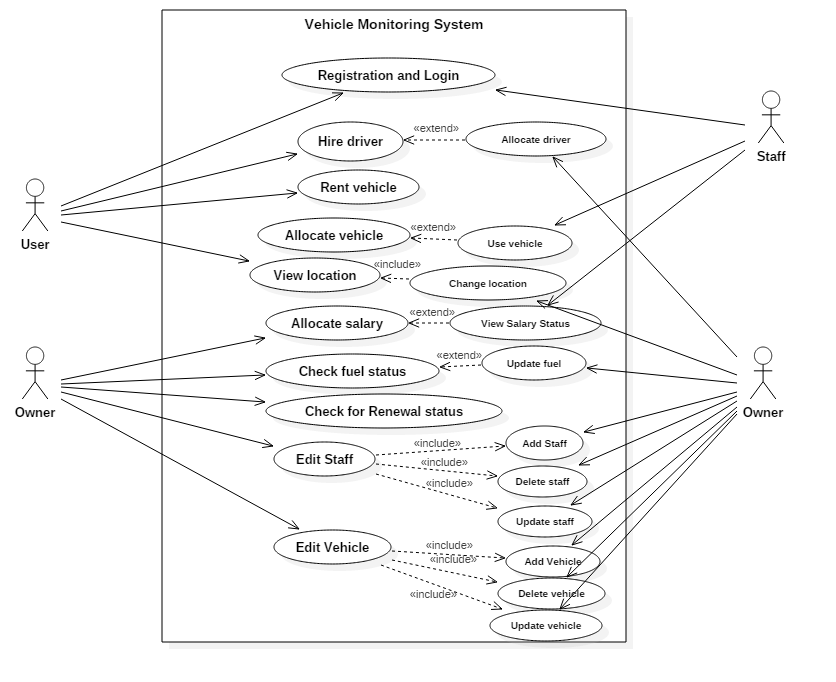


Figure 1: Use Case diagram

## Justification

### Advantages

For the scenarios below, use cases are ideal:

* Helps in detailing user interaction goals with a product.
* Relevant outlining and ensuring the requirements of a system.
* Determining the specific needs of a project.
* Modeling the basic flow of events in a use case.

### Disadvantages

* They do not capture the non-functional requirements easily.
* There might be a learning curve for the developer and/or specially, the client in using these use cases.
* Use case diagram is not suitable if we want to lay out step-by-step details. These diagrams summarize interactions, not explain them.

# Design

# Structural Model

# Class Diagram

A class diagram is an illustration of the relationships and source code dependencies among classes in the Unified Modeling Language (UML). In this context, a class defines the methods and variables in an object, which is a specific entity in a program or the unit of code representing that entity. Class diagrams are useful in all forms of object-oriented programming (OOP).

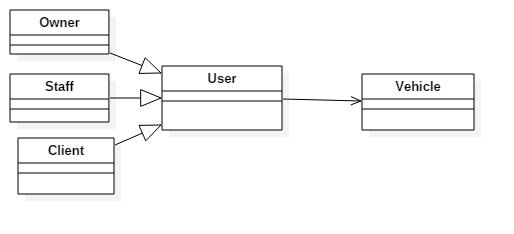


Figure 2:Class Diagram

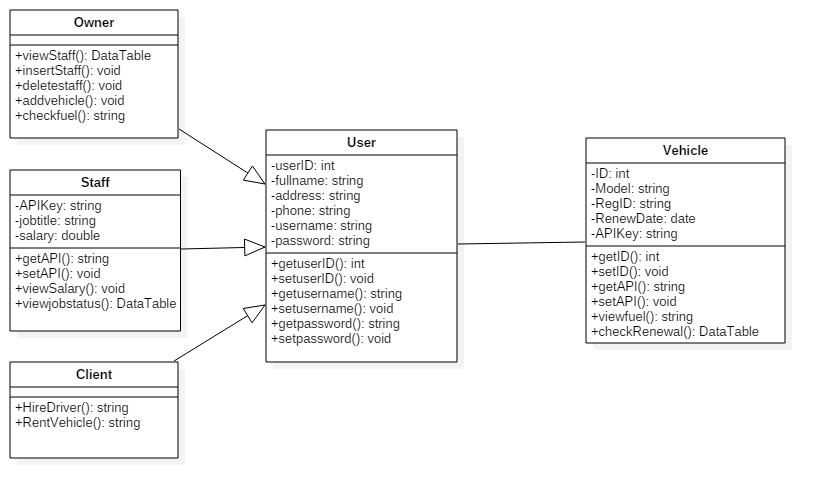


Figure 3:Final Class Diagram

## Justification

### Advantages

* It forces the programmer to think out the structure of his/her classes and how they will interact with each other before actually writing any code. This may lead to a more robust application.
* It provides a blueprint for maintenance programmers to get an overview of how the application is structured before examining the actual code. This may reduce maintenance time.

### Disadvantages

* The programmer may need to learn UML to build the class diagram in the first place.
* The time spent building the class diagram may add to overall development time.
* If the class diagram is overcomplicated, then it may be difficult to correlate with the actual code.

# ER Diagram

An entity relationship diagram (ERD) shows the relationships of entity sets stored in a database. An entity in this context is an object, a component of data. An entity set is a collection of similar entities. These entities can have attributes that define its properties. By defining the entities, their attributes, and showing the relationships between them, an ER diagram illustrates the logical structure of databases.ER diagrams are used to sketch out the design of a database.

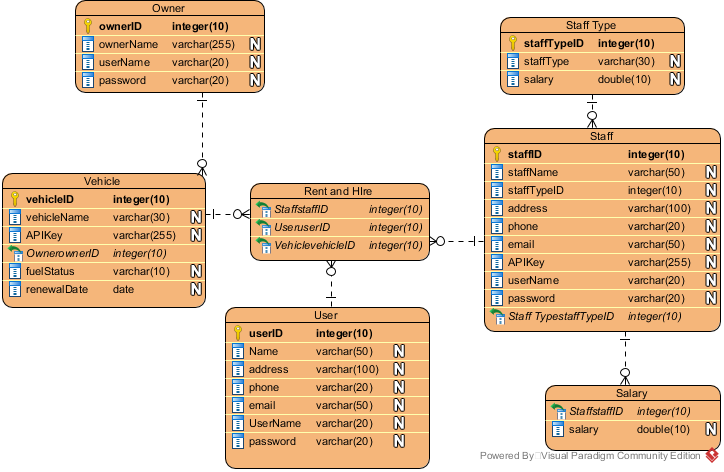


Figure 4: ER diagram

## Justification

### Advantages

* Conceptually it is very simple: ER model is very simple because if we know relationship between entities and attributes, then we can easily draw an ER diagram.
* Better visual representation: ER model is a diagrammatic representation of any logical structure of database. By seeing ER diagram, we can easily understand relationship among entities and relationship.
* Effective communication tool: It is an effective communication tool for database designer.
* Highly integrated with relational model: ER model can be easily converted into relational model by simply converting ER model into tables.
* Easy conversion to any data model: ER model can be easily converted into another data model like hierarchical data model, network data model and so on.

### Disadvantages

* Limited constraints and specification
* Loss of information content: Some information be lost or hidden in ER model
* Limited relationship representation: ER model represents limited relationship as compared to another data models like relational model etc.
* No representation of data manipulation: It is difficult to show data manipulation in ER model.
* Popular for high level design: ER model is very popular for designing high level design
* No industry standard for notation

# Behavioral model

# Activity Diagram

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc.

## Vehicle Rental

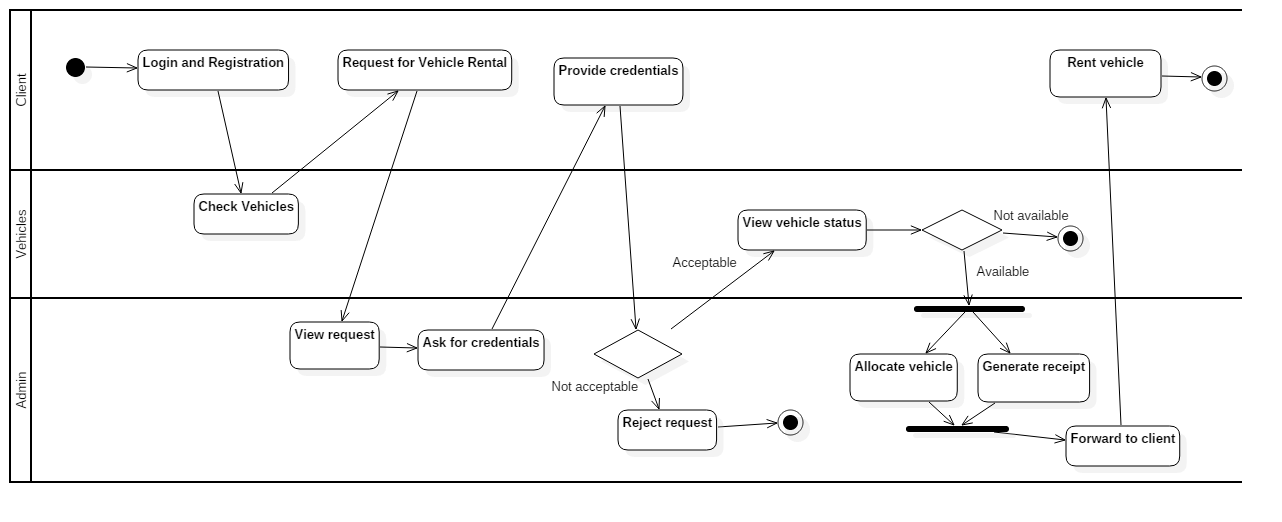


Figure 5:Activity Diagram for Rental

## Driver Hire

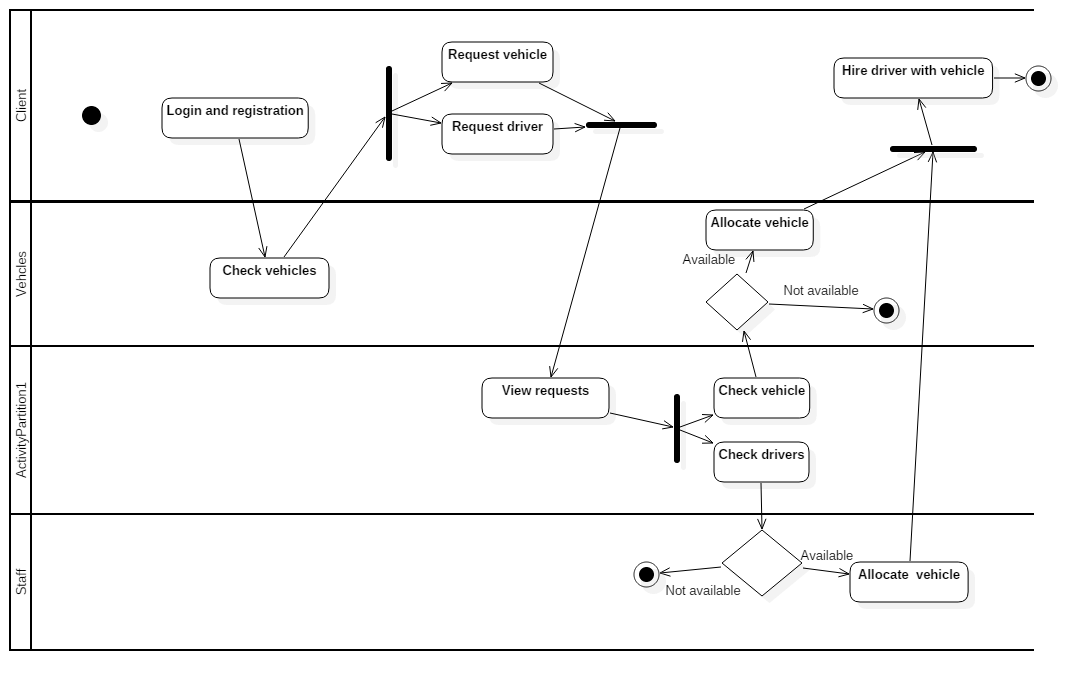


Figure 6: Hire driver with vehicle

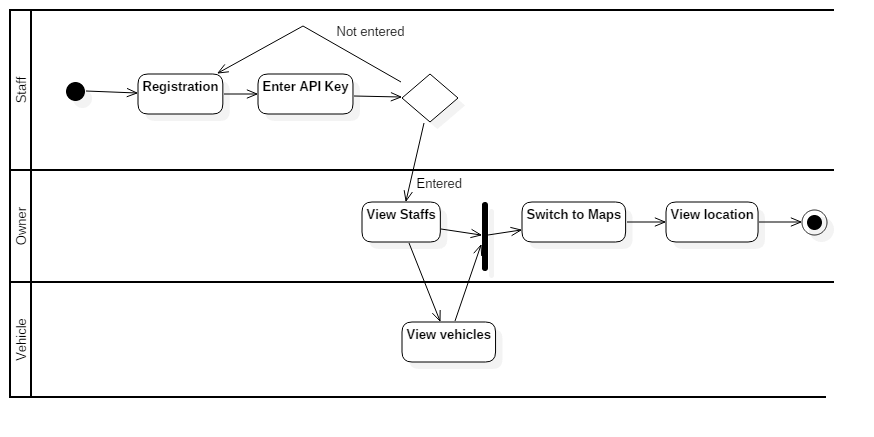


Figure : View location

## Justification

### Advantages

* UML modeling language included that these diagrams are normally easily comprehensible for both analysts and stakeholders.
* Since they are among the most user-friendly diagrams available, they are generally regarded as an essential tool in an analyst’s repertoire.
* Additionally, as stated above, activity diagrams allow an analyst to display multiple conditions and actors within a work flow through the use of swim-lanes. Swim-lanes, however, are optional as a single condition or actor is normally displayed without them.

### Disadvantages

* UML modeling language include that activity diagrams have the potential to become overly complex because their user-friendly nature may lend itself to an all-inclusive description.
* Another aspect of activity diagram is that they may not be used in lieu of a state diagram or sequence diagram because activity diagrams do not give detail about how objects behave or how objects collaborate.

# UI design

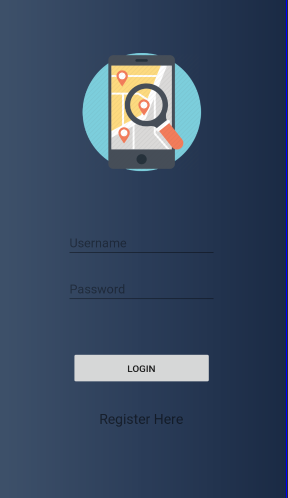


Figure 8: Login form

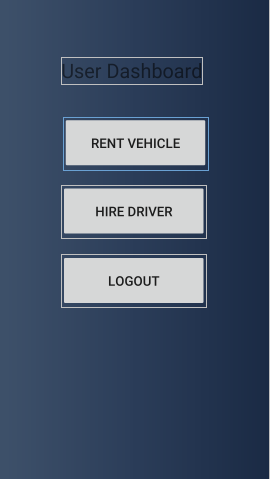


Figure 9: User dashboard

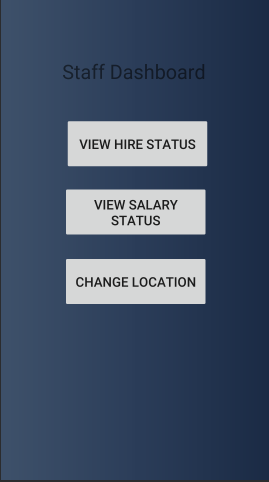


Figure 10: Staff dashboard

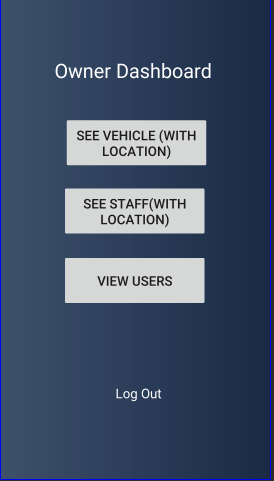


Figure 11: Owner dashboard