

# Software Architecture Document

Version 3.0

for

## Vehicle Renting System

Prepared by

|                         |                                                                                                       |
|-------------------------|-------------------------------------------------------------------------------------------------------|
| 1.Bhavana Bomma         | - 40071320 - <a href="mailto:bomma08@gmail.com">bomma08@gmail.com</a>                                 |
| 2.Divyaprabha Rajendran | - 40089282 - <a href="mailto:divyanajma@gmail.com">divyanajma@gmail.com</a>                           |
| 3.Harish Jayasankar     | - 40105791 - <a href="mailto:harishjayasankar28@gmail.com">harishjayasankar28@gmail.com</a>           |
| 4.Ishpreet Singh        | - 40093666 - <a href="mailto:ishpreetsingh923@gmail.com">ishpreetsingh923@gmail.com</a>               |
| 5.Mahy Salama           | - 40074737 - <a href="mailto:mahyhanysalama@gmail.com">mahyhanysalama@gmail.com</a>                   |
| 6.Swetha Chenna         | - 40092019 - <a href="mailto:swethachenna2018@gmail.com">swethachenna2018@gmail.com</a> [Team Leader] |

Instructor: Dr. C. Constantinides

Course: SOEN-6461

Date: 27/10/2019

|                                |                 |
|--------------------------------|-----------------|
| Vehicle Renting System         | Version: 3.0    |
| Software Architecture Document | Date:27/10/2019 |

### Document history

| Date       | Version | Description                                                                | Author           |
|------------|---------|----------------------------------------------------------------------------|------------------|
| 06/10/2019 | V2      | missing parts and format                                                   | all team members |
| 05/10/2019 | V1      | Use Case realization and sequence diagrams                                 | all team members |
| 10/27/2019 | V3      | Adding admin functionality related diagrams and reformatting class diagram | all team members |
|            |         |                                                                            |                  |

### Table of contents

|                                                      |    |
|------------------------------------------------------|----|
| 1. Introduction                                      | 4  |
| 2. Architectural representation                      | 5  |
| 7. Architectural requirements: goals and constraints | 7  |
| 8. Use case view (Scenarios)                         | 8  |
| 9. Logical view                                      | 8  |
| 10. Development (Implementation) view                | 10 |
| 11. Process view                                     | 10 |
| 12. Deployment (Physical) view                       | 10 |
| 13. Data view (optional)                             | 10 |
| 14. Quality                                          | 11 |

### List of figures

|                               |   |
|-------------------------------|---|
| Figure 1: The 4+1 view model. | 4 |
|-------------------------------|---|

|                                |                 |
|--------------------------------|-----------------|
| Vehicle Renting System         | Version: 3.0    |
| Software Architecture Document | Date:27/10/2019 |

## 1. Introduction

Vehicle Rental System is a web application which includes reserving ,renting and returning vehicles. The Software Architecture Document is requires while developing the web application, the significant decisions are captured here in the Software Architecture Document, it is the map of the software. We use it to see and understand the software's modules and components before digging into the code.

### Purpose

The Software Architecture Document provides a comprehensive architectural overview of the system, using a different architectural views to show different aspects of the system, the 4+1 view model that contains logical view for the designers to design a system that will provide the proper functionality for the end user, development view for programmers that basically deals with source code, executable files and component diagram for the implementation, process view for integrators that focus on how the system components communicate during the run time, physical view for deployment managers that considered with the physical connections between these components, and use case views the audiences are all the stakeholders of the system including the end users that illustrates the different scenarios and paths for the user to interact with the system.

### Scope

The Software Architecture Document applies to all stakeholders that meant to design and implement the functionality of the clerk and administrator in our vehicle rental system where architecture of each functionality will be specified.

### Definitions, acronyms, and abbreviations

Provides the definitions of all terms, acronyms, and abbreviations required to properly interpret the Software Architecture Document which will be used in complete project

### project's Glossary

| <u>Abbreviation</u> | <u>Definition</u>                                                                                                                                                                                                       |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SAD                 | Software Architecture Document, is a set of practices, techniques and types of representations used by software architects to record a software architecture. It is a modeling activity (Software architectural model). |

|                                |                 |
|--------------------------------|-----------------|
| Vehicle Renting System         | Version: 3.0    |
| Software Architecture Document | Date:27/10/2019 |

|     |                                                                                                                                                                                                                |
|-----|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|     | Architecture models can take various forms, including text, informal drawings, diagrams or other formalisms (UML).                                                                                             |
| UML | Unified Modeling Language, is a standardized modeling language enables developers to specify, visualize, construct and document the artifacts of a software system.                                            |
| UC  | Use Case, is a list of actions or event steps typically defining the interactions between a role (an actor in UML) and a system to achieve a goal. The actor can be a human, machine or other external system. |

## 2. Architectural representation

Describe the top-level architectural style of the system and the view model you will adopt.

Additionally describe what each individual view will provide. Many enterprise software systems are modeled using the 4+1 view illustrated in Figure 1.

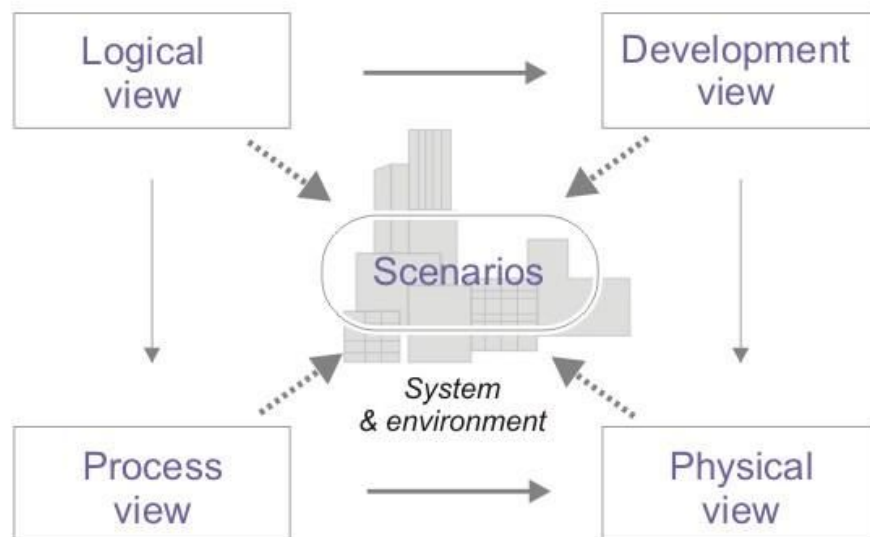


Figure 1: The 4+1 view model.

|                                |                 |
|--------------------------------|-----------------|
| Vehicle Renting System         | Version: 3.0    |
| Software Architecture Document | Date:27/10/2019 |

1. **Logical view** :It supports and provides clarification for the functional requirements, what services the system should provide to its end users, in this phase we try to get better knowledge of problem by talking to domain experts and other stakeholders, whatever decision taken here are independent of implementation decision. We elucidate logical view using Class diagram.
2. **Development view** :The development view focuses on organization of software modules in actual software. the software is organised as layers, each layer providing service to layer above it. we allocate effort required for development and maintenance in this view. it is represented using package diagram.
3. **Process view** : It is a dynamic view and it focuses on what will happen during run-time and also on non-functional requirements like performance. It addresses issues of concurrency and distribution, fault tolerance, process synchronization etc. It is explained using activity diagram.
4. **Physical view**:It focus on mapping software on to existing hardware.All Non functional requirements are taken care off in this view like performance, scalability etc. Its represented using deployment diagram.
5. **Use case view** (also known as Scenarios) : This view shows the interactions between actors/objects and the processes. In this view, we use a set of use cases or scenarios to depict the description of the system architecture.
6. **Data view** (optional): Audience: Data specialists, Database administrators. Describes the architecturally significant persistent elements in the data model .

Related Artifacts: **Data model**.

## 7. Architectural requirements: goals and constraints

Requirements are already described in SRS. In this section describe *key* requirements and constraints that have a significant impact on the architecture.

|                                |                 |
|--------------------------------|-----------------|
| Vehicle Renting System         | Version: 3.0    |
| Software Architecture Document | Date:27/10/2019 |

### **Functional requirements (Use case view)**

Refer to Use Cases or Use Case scenarios which are relevant with respect to the software architecture. The Use Cases referred to should contain central functionality, many architectural elements or specific delicate parts of the architecture.

The overview below refers to architecturally relevant Use Cases from the Use Case Model (see references).

#### **Use cases realization**

| <b>Source</b>       | <b>Name</b>  | <b>Architectural<br/>relevance</b>      | <b>Addressed in:</b>                |
|---------------------|--------------|-----------------------------------------|-------------------------------------|
| Use case UC1 in SRS | View Catalog | Clerk: Wants to view the vehicle CatLog | 9.4.1 (Use case realization)<br>UC1 |

| <b>Source</b>       | <b>Name</b>          | <b>Architectural<br/>relevance</b>                                           | <b>Addressed in:</b>                |
|---------------------|----------------------|------------------------------------------------------------------------------|-------------------------------------|
| Use case UC2 in SRS | View Vehicle Details | Clerk: Wants to view the vehicle CatLog and view a particular vehicle detail | 9.4.2 (Use case realization)<br>UC2 |

| <b>Source</b>       | <b>Name</b>       | <b>Architectural<br/>relevance</b>                                                 | <b>Addressed in:</b>                |
|---------------------|-------------------|------------------------------------------------------------------------------------|-------------------------------------|
| Use case UC3 in SRS | Sorting/Filtering | Clerk: Wants to search the desired vehicle based on Sorting and filtering criteria | 9.4.3 (Use case realization)<br>UC3 |

|                                |                 |
|--------------------------------|-----------------|
| Vehicle Renting System         | Version: 3.0    |
| Software Architecture Document | Date:27/10/2019 |

| Source              | Name          | Architectural relevance                     | Addressed in:                       |
|---------------------|---------------|---------------------------------------------|-------------------------------------|
| Use case UC4 in SRS | create Client | Clerk: wants to create a new client record. | 9.4.4 (Use case realization)<br>UC4 |

| Source              | Name        | Architectural relevance                         | Addressed in:                       |
|---------------------|-------------|-------------------------------------------------|-------------------------------------|
| Use case UC5 in SRS | Edit Client | Clerk: Wants to edit the existing client record | 9.4.5 (Use case realization)<br>UC5 |

| Source              | Name          | Architectural relevance         | Addressed in:                       |
|---------------------|---------------|---------------------------------|-------------------------------------|
| Use case UC6 in SRS | Delete Client | Clerk: Wants to delete a client | 9.4.6 (Use case realization)<br>UC6 |

| Source              | Name                   | Architectural relevance                            | Addressed in:                    |
|---------------------|------------------------|----------------------------------------------------|----------------------------------|
| Use case UC7 in SRS | makeReservatioOrRental | Clerk: Requested to create a reservation or rental | 9.4.7 (Use case realization) UC7 |

|  |  |  |  |
|--|--|--|--|
|  |  |  |  |
|--|--|--|--|

|                                |                 |
|--------------------------------|-----------------|
| Vehicle Renting System         | Version: 3.0    |
| Software Architecture Document | Date:27/10/2019 |

|                     |               |                                                          |                                  |
|---------------------|---------------|----------------------------------------------------------|----------------------------------|
| Use case UC8 in SRS | Cancel/Return | Clerk: Requested to cancel a reservation/return a rental | 9.4.8 (Use case realization) UC8 |
|---------------------|---------------|----------------------------------------------------------|----------------------------------|

| Source              | Name                     | Architectural relevance                                      | Addressed in:                    |
|---------------------|--------------------------|--------------------------------------------------------------|----------------------------------|
| Use case UC9 in SRS | view Transaction History | Administrator: Wants to view the vehicle transaction history | 9.4.9 (Use case realization) UC9 |

| Source               | Name               | Architectural relevance                                                                                 | Addressed in:                     |
|----------------------|--------------------|---------------------------------------------------------------------------------------------------------|-----------------------------------|
| Use case UC10 in SRS | Transaction Search | Administrator: Should be able to search the transaction history as per client ,per vehicle,per due date | 9.5.0 (Use case realization) UC10 |

| Source               | Name                      | Architectural relevance                                                                     | Addressed in:                     |
|----------------------|---------------------------|---------------------------------------------------------------------------------------------|-----------------------------------|
| Use case UC11 in SRS | Create new vehicle record | Administrator: Should be able to create a new vehicle record with the necessary conditions. | 9.5.1 (Use case realization) UC11 |

| Source | Name | Architectural relevance | Addressed in: |
|--------|------|-------------------------|---------------|
|        |      |                         |               |



|                                |                 |
|--------------------------------|-----------------|
| Vehicle Renting System         | Version: 3.0    |
| Software Architecture Document | Date:27/10/2019 |

|                      |                       |                                                                                       |                                   |
|----------------------|-----------------------|---------------------------------------------------------------------------------------|-----------------------------------|
| Use case UC12 in SRS | Modify vehicle record | Administrator: Should be able to modify vehicle record with the necessary conditions. | 9.5.2 (Use case realization) UC12 |
|----------------------|-----------------------|---------------------------------------------------------------------------------------|-----------------------------------|

| Source               | Name           | Architectural relevance                                                     | Addressed in:                     |
|----------------------|----------------|-----------------------------------------------------------------------------|-----------------------------------|
| Use case UC13 in SRS | Delete Vehicle | Administrator: Should be able to delete vehicle using license plate number. | 9.5.3 (Use case realization) UC13 |

| Source               | Name           | Architectural relevance                                                                            | Addressed in:                     |
|----------------------|----------------|----------------------------------------------------------------------------------------------------|-----------------------------------|
| Use case UC14 in SRS | Catalog Search | Administrator: Should be able to search catalog for particular vehicle using license plate number. | 9.5.4 (Use case realization) UC14 |

### Non-functional requirements

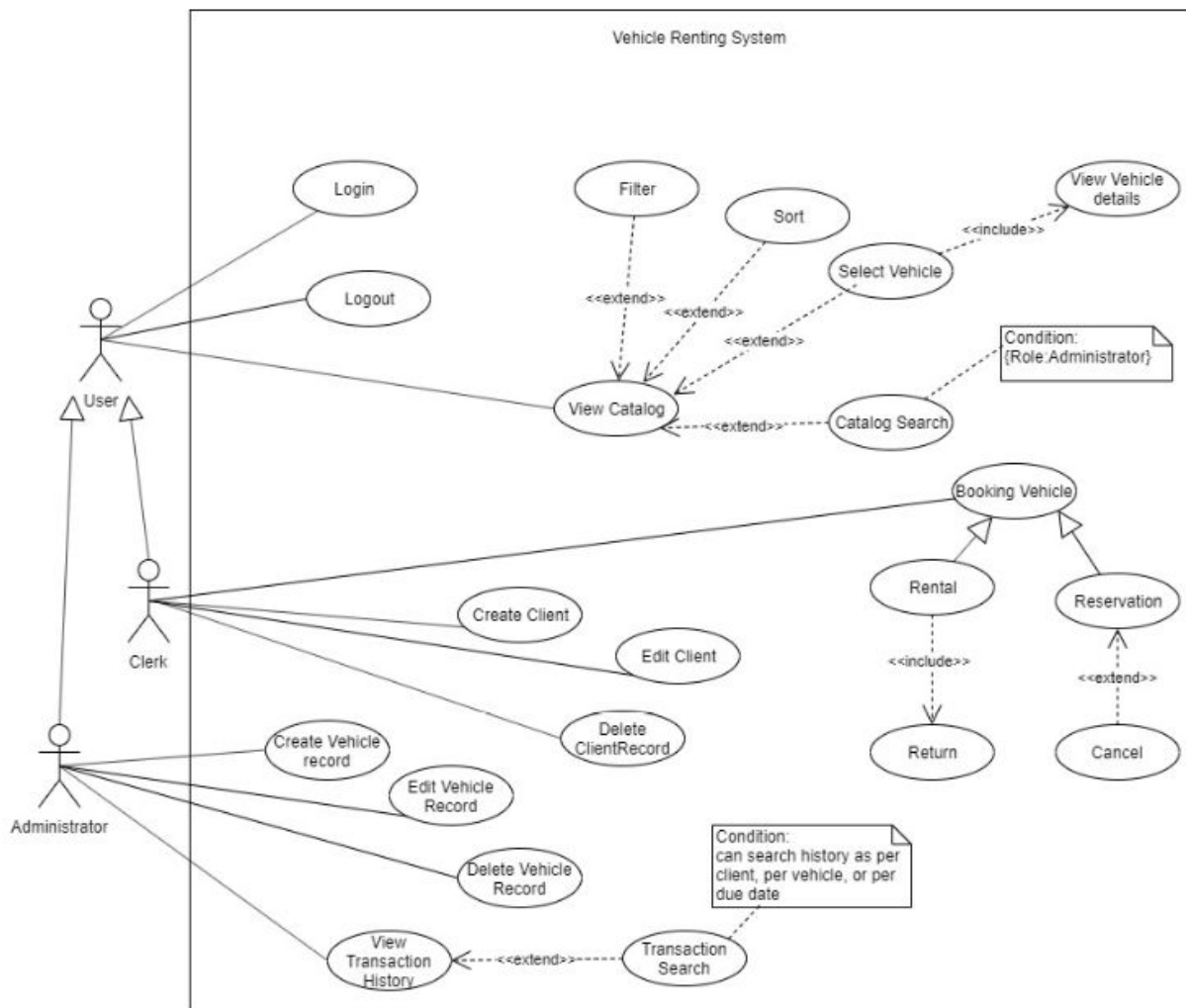
Describe the architecturally relevant non-functional requirements, i.e. those which are important for developing the software architecture. Think of security, privacy, third-party products, system dependencies, distribution and reuse. Also environmental factors such as context, design, implementation strategy, team composition, development tools, time to market, use of legacy code may be addressed.

|                                |                 |
|--------------------------------|-----------------|
| Vehicle Renting System         | Version: 3.0    |
| Software Architecture Document | Date:27/10/2019 |

Usually, the non-functional requirements are already in place and can be referenced here. This document is not meant to be the source of non-functional requirements, but to address them. Provide a reference per requirement, and where the requirement is addressed.

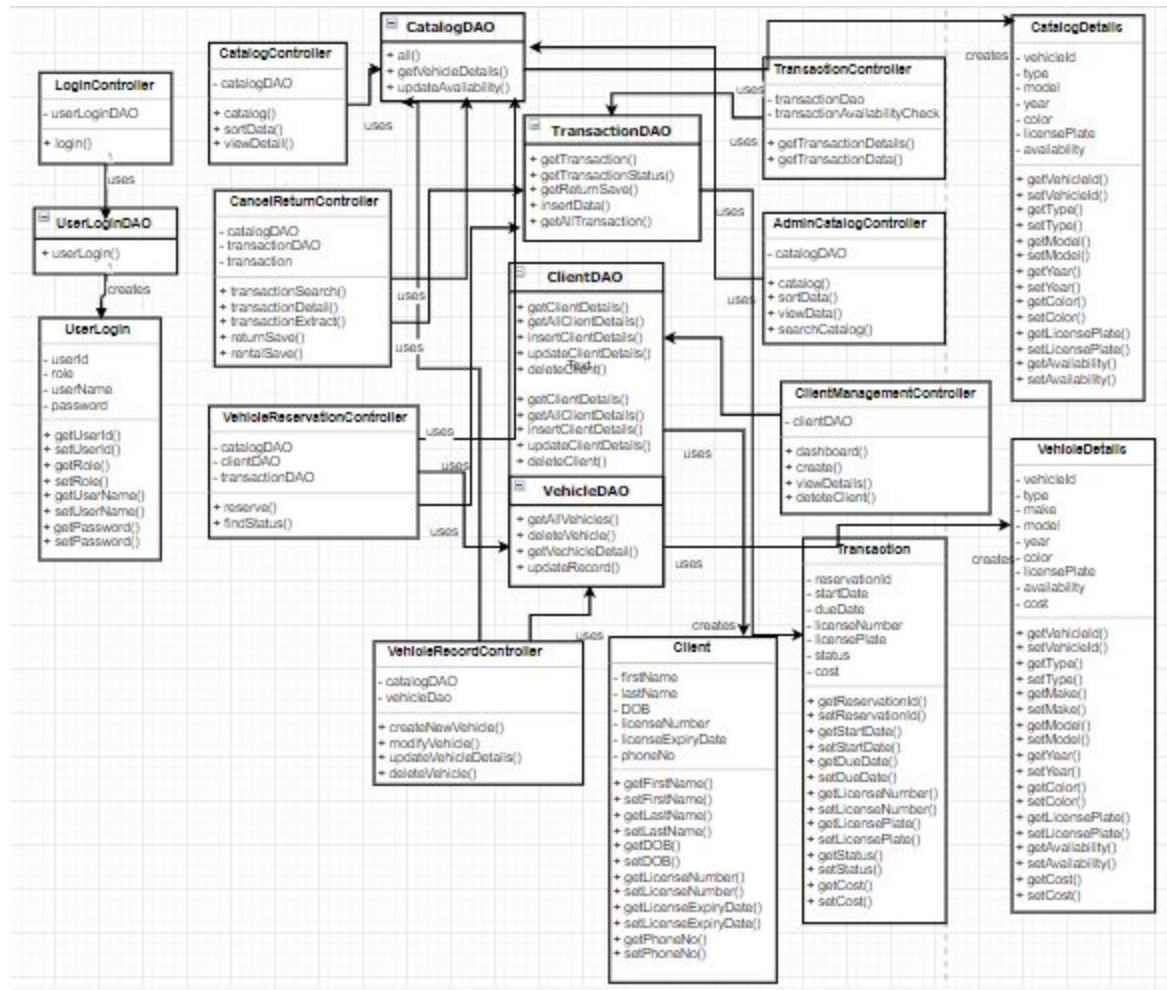
## 8. Use case view (Scenarios)

The scenarios (or functional view) represent the behavior of the system as seen by its actors. Use case scenarios describe sequences of interactions between actors and the system (seen as a black box) as well as between the system and external systems the *UML use case diagram* is used to capture this view.



|                                |                 |
|--------------------------------|-----------------|
| Vehicle Renting System         | Version: 3.0    |
| Software Architecture Document | Date:27/10/2019 |

## 9. Logical view



### 9.1 Layers, tiers etc.

Describe the top-level architecture style. Deploy a *UML class diagram*.

### 9.2 Subsystems

Describe the decomposition of the system in subsystems and show their relation.

### 9.3 Architecturally significant design packages

Describe packages of individual subsystems that are architecturally significant. For each package, include a subsection with its name, its brief description, and a diagram with all significant classes and packages contained within the package.

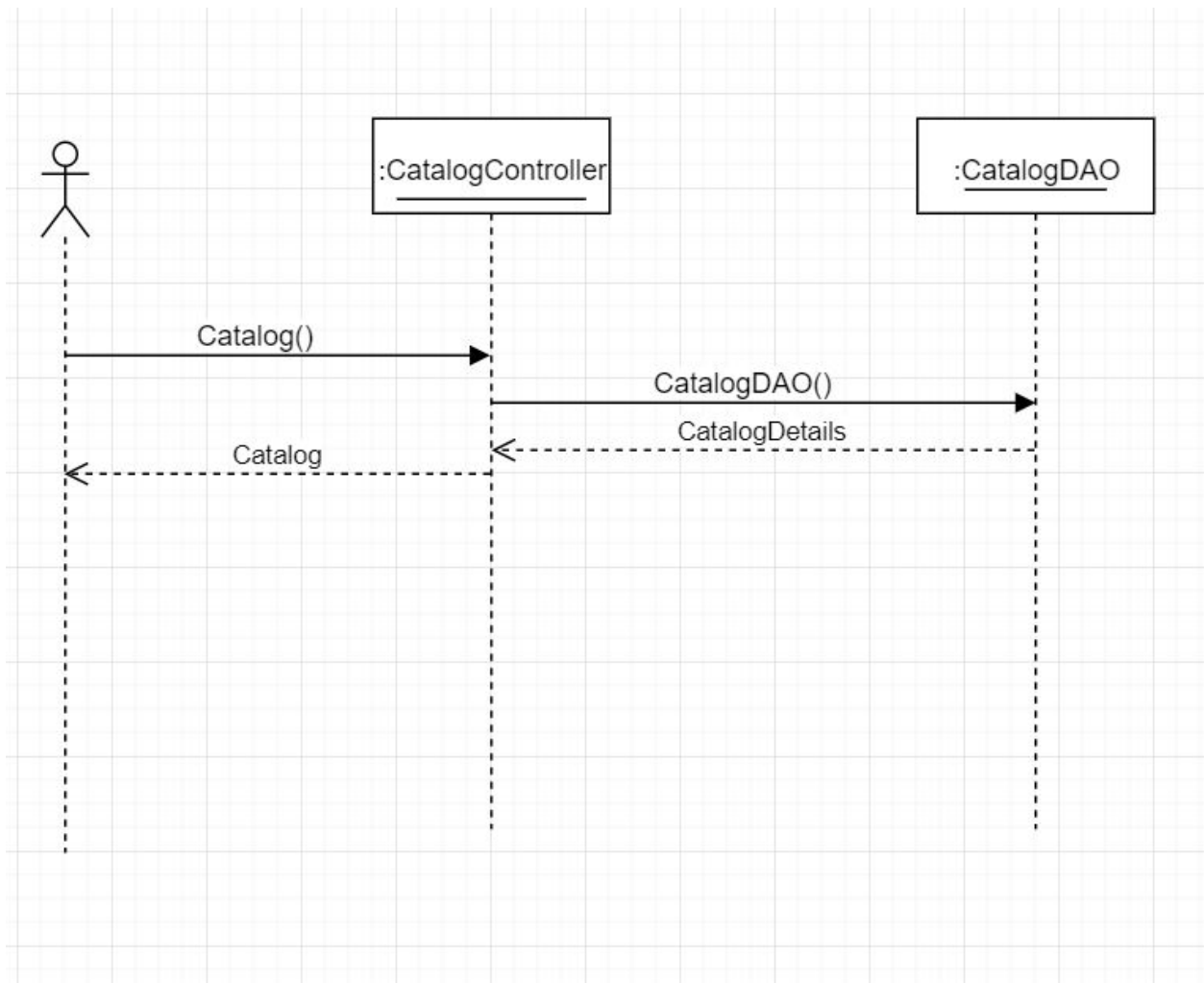
|                                |                 |
|--------------------------------|-----------------|
| Vehicle Renting System         | Version: 3.0    |
| Software Architecture Document | Date:27/10/2019 |

#### 9.4 Use case realizations

In this section you have to illustrate how use cases are translated into *UML interaction diagrams*. Give examples of the way in which the Use Case Specifications are technically translated into Use Case Realizations, for example, by providing a sequence-diagram. Explain how the tiers communicate and clarify how the components or objects used realize the functionality.

9.4.1 UC1: 1,The instance of List<Catalog> instance is created.

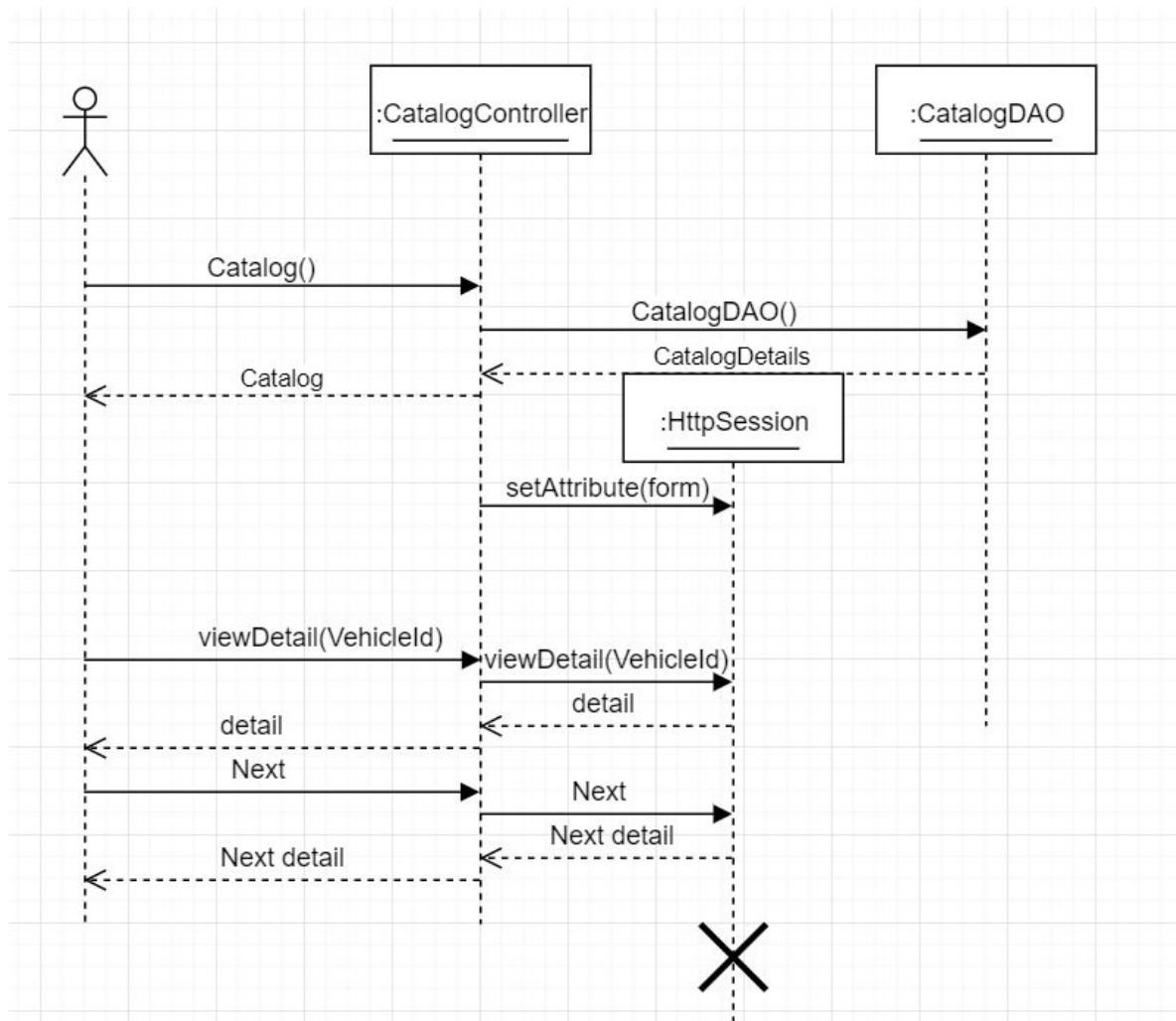
2,instance was associated with Vehicle details table.



|                                |                 |
|--------------------------------|-----------------|
| Vehicle Renting System         | Version: 3.0    |
| Software Architecture Document | Date:27/10/2019 |

9.4.2 UC2: 1,The instance of List<Catalog> is created.

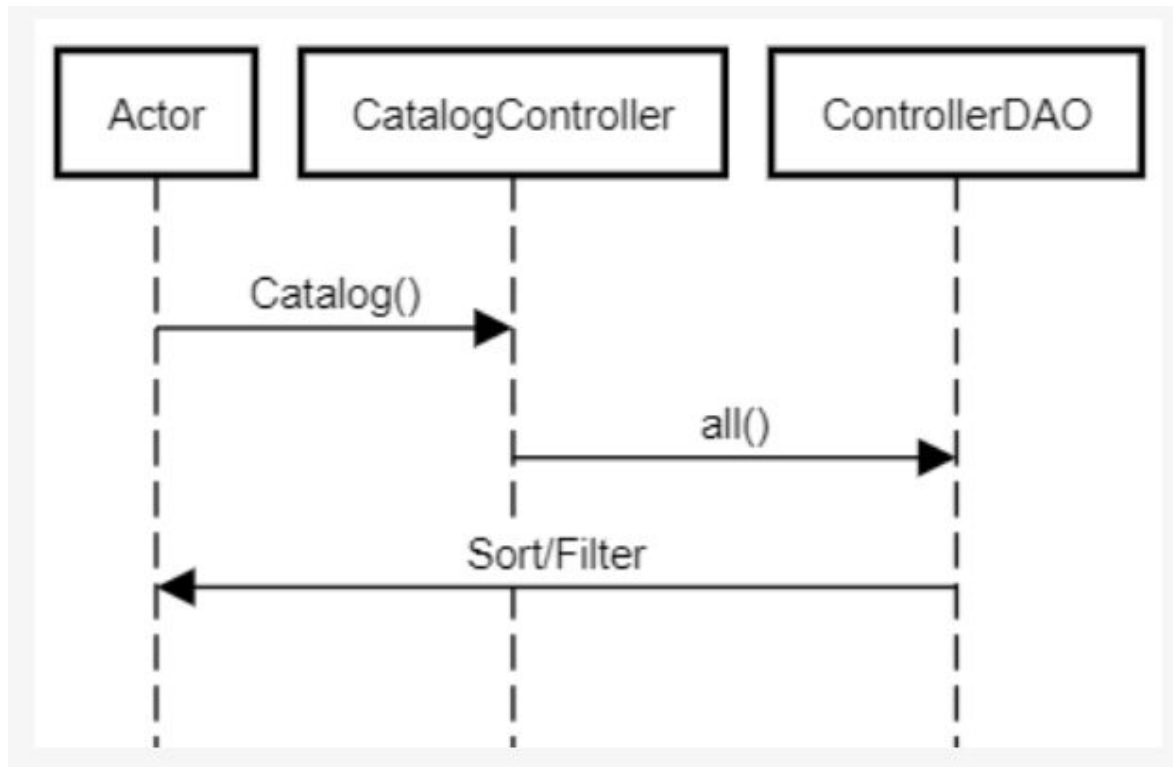
2,instance was associated with Vehicle details table.



9.4.3 UC3: 1. The instance of catalog is created

|                                |                 |
|--------------------------------|-----------------|
| Vehicle Renting System         | Version: 3.0    |
| Software Architecture Document | Date:27/10/2019 |

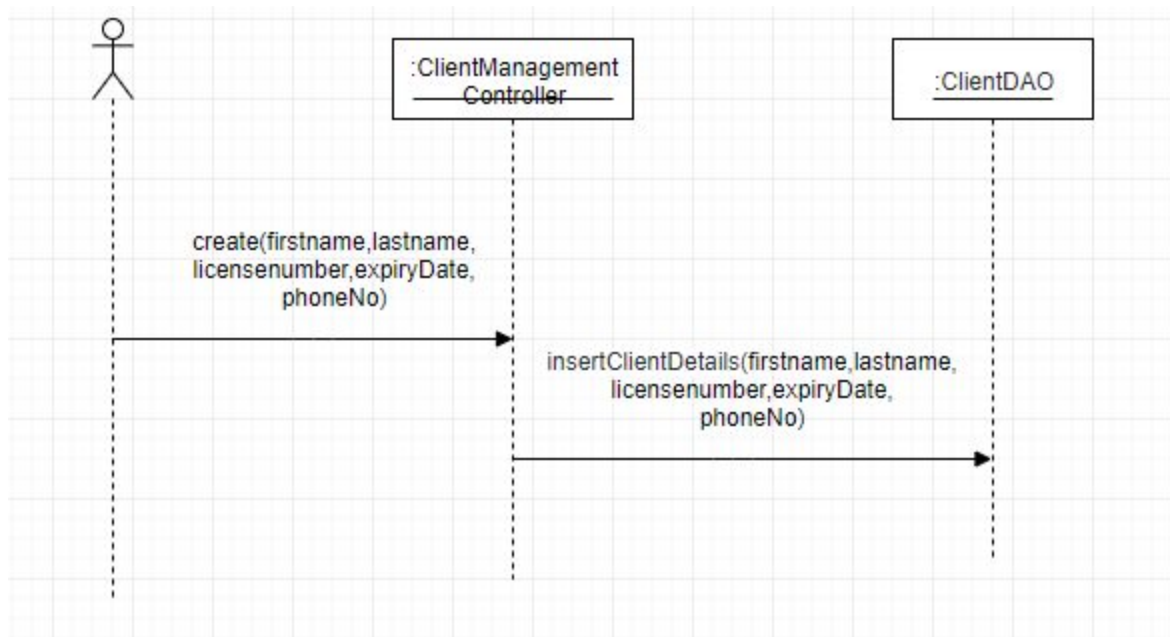
2. cl, is associated with vehicle sorting/filtering details



9.4.4 UC4: 1.With all the details instance of client will be created.

|                                |                 |
|--------------------------------|-----------------|
| Vehicle Renting System         | Version: 3.0    |
| Software Architecture Document | Date:27/10/2019 |

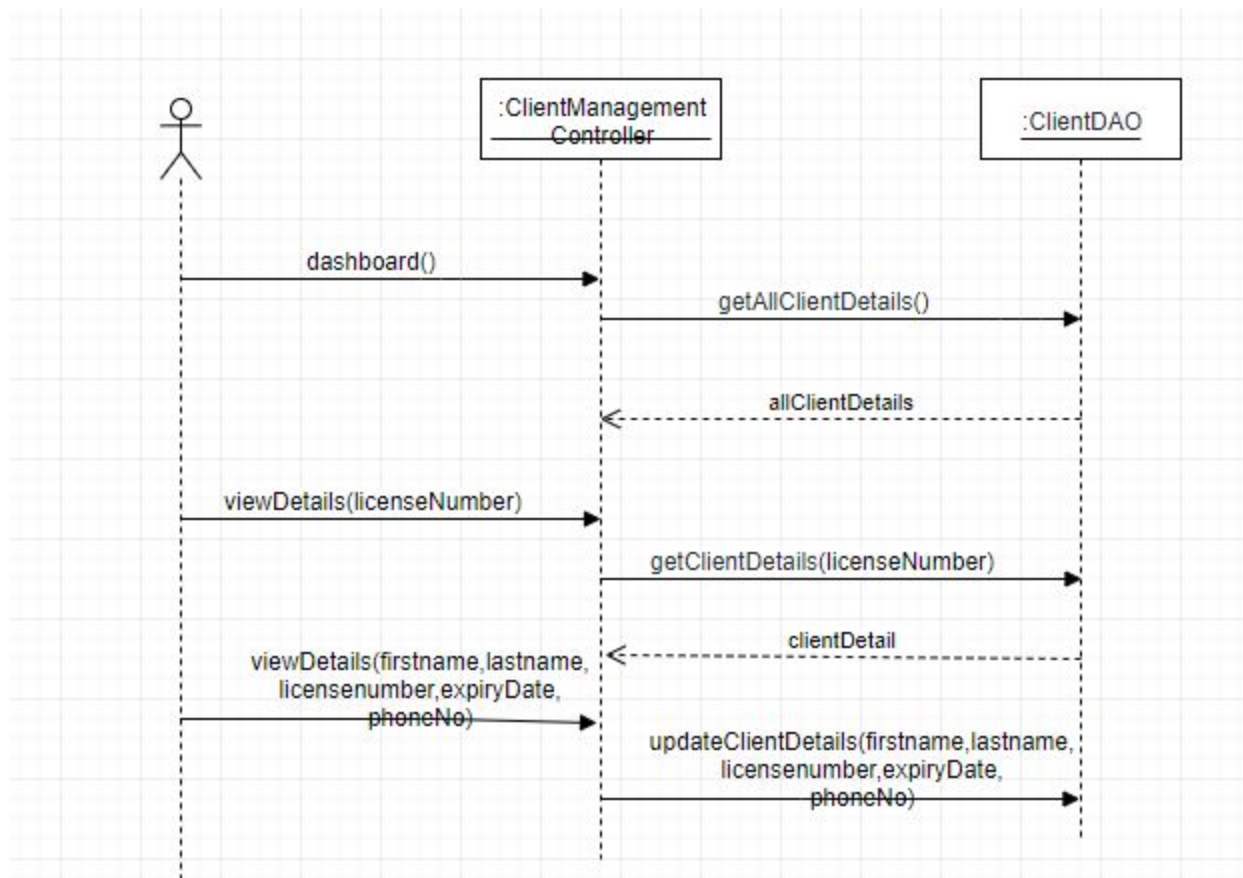
2.The details in the client instance will be added into database.



9.4.5 UC5: 1.With all the modified details instance of client will be created.

|                                |                 |
|--------------------------------|-----------------|
| Vehicle Renting System         | Version: 3.0    |
| Software Architecture Document | Date:27/10/2019 |

2.The details in the client instance will be added into database.

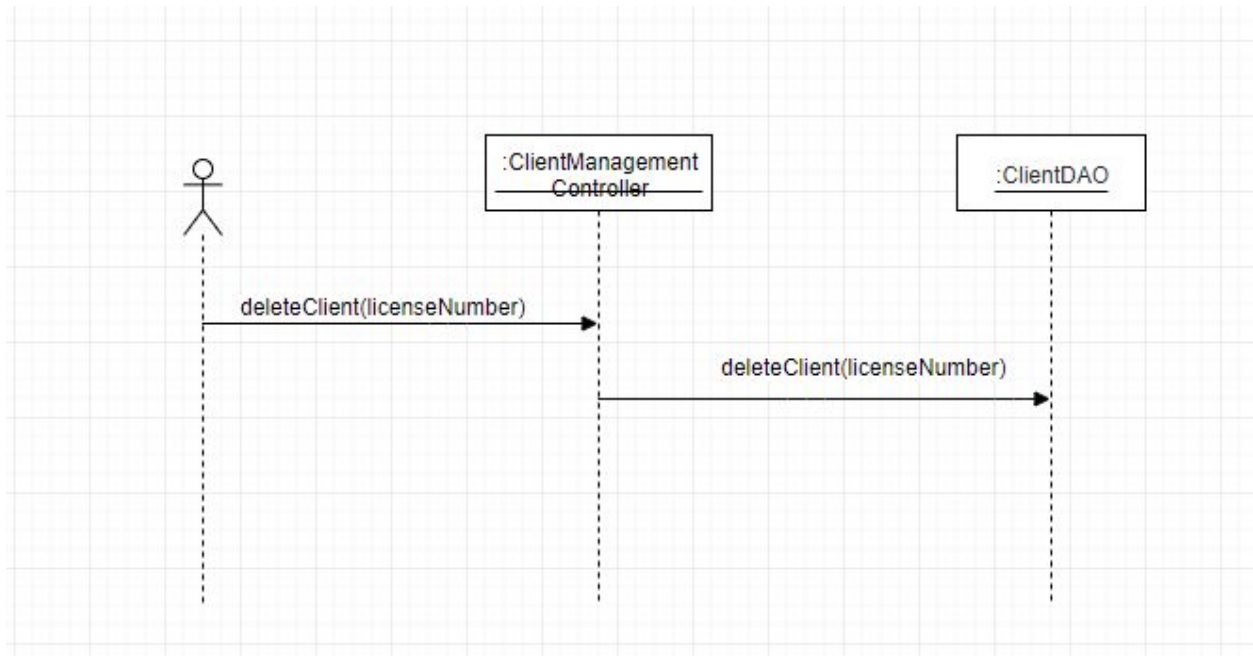


9.4.6 UC6: 1,The instance of client will be deleted.

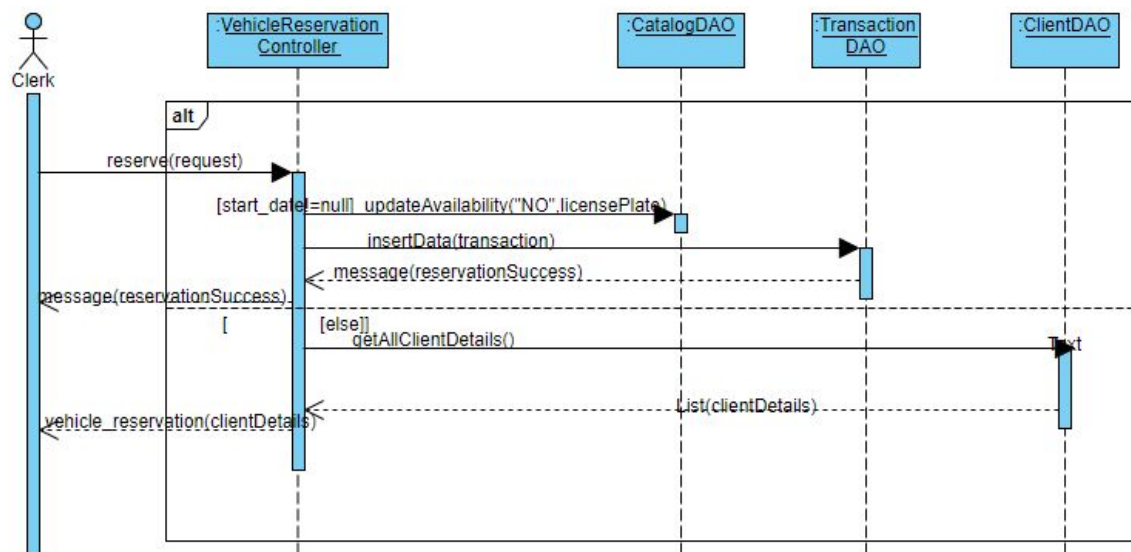
2,client record will be deleted from the database.



|                                |                 |
|--------------------------------|-----------------|
| Vehicle Renting System         | Version: 3.0    |
| Software Architecture Document | Date:27/10/2019 |



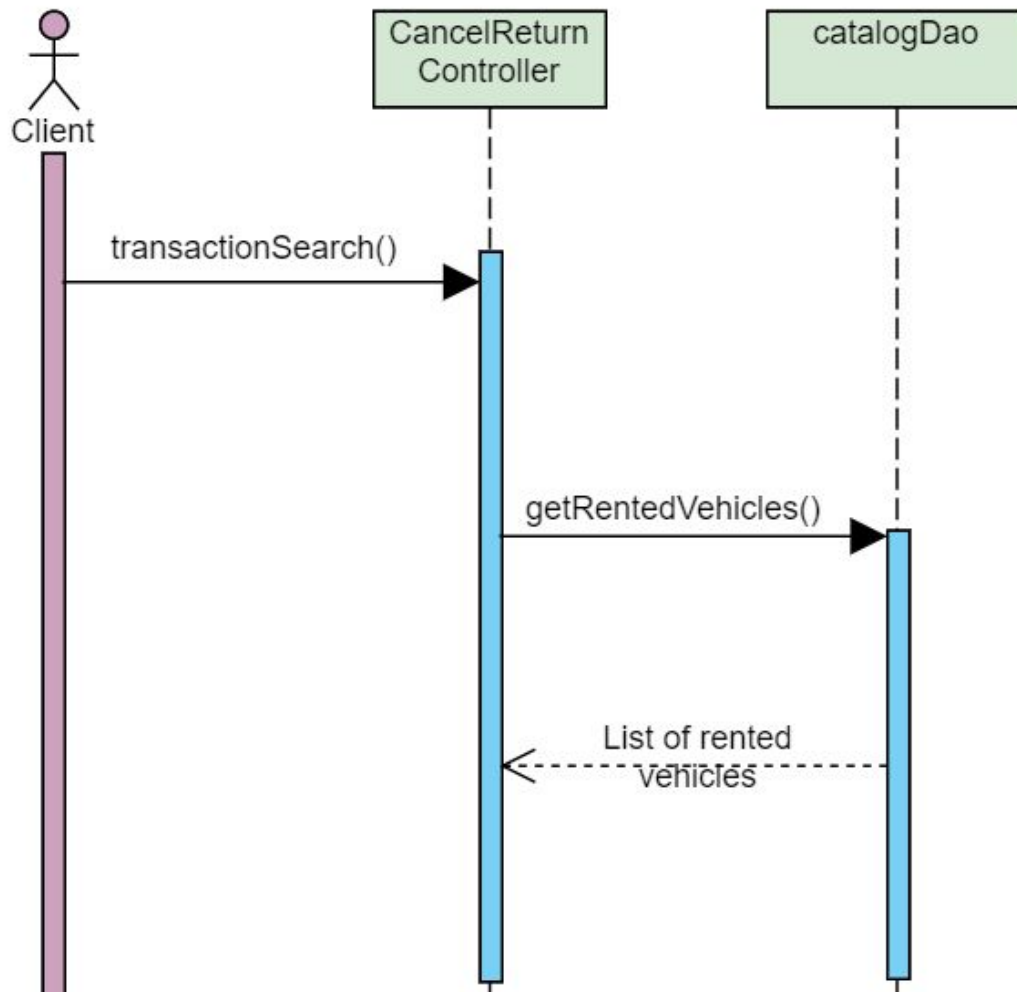
- 9.4.7 UC7: 1, A reservation/rental record will be created.  
 2, The vehicle availability will be changed to “NO”.



|                                |                 |
|--------------------------------|-----------------|
| Vehicle Renting System         | Version: 3.0    |
| Software Architecture Document | Date:27/10/2019 |

9.4.8 UC8: 1, A Cancel/return record will be created.

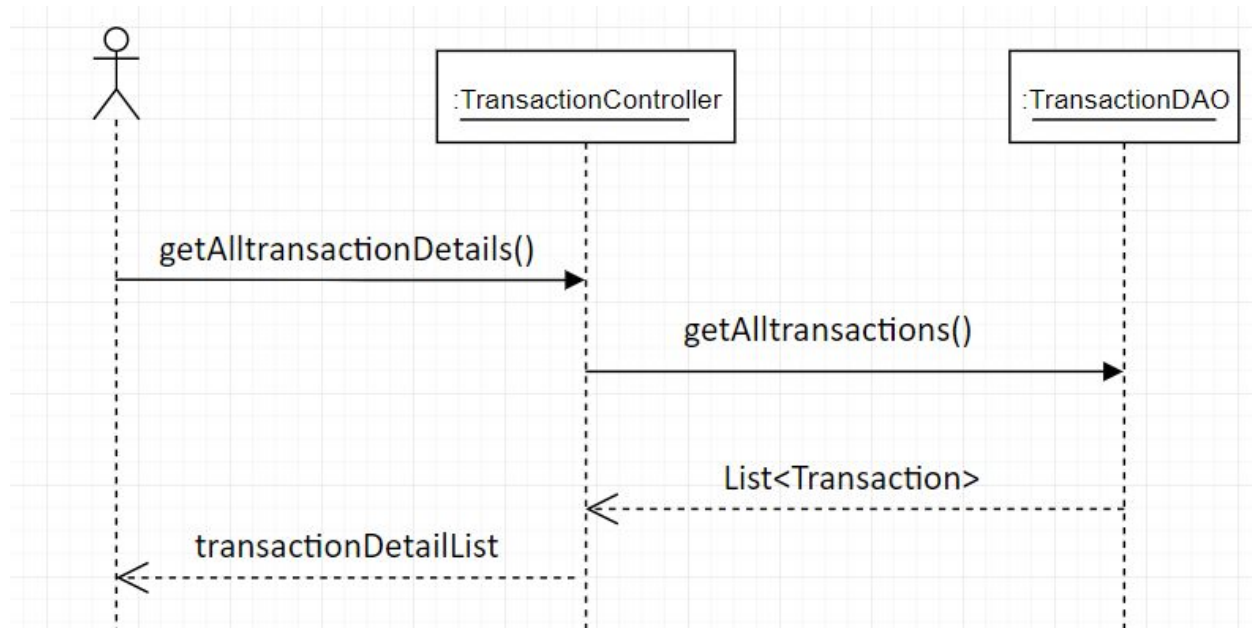
2, The vehicle availability will be changed to "YES".



9.4.9:- UC9:

1. The list of instances of Transactions will be created.
2. All the transaction history records will be fetched from the database.
3. The instances attributes will be updated with data from database.

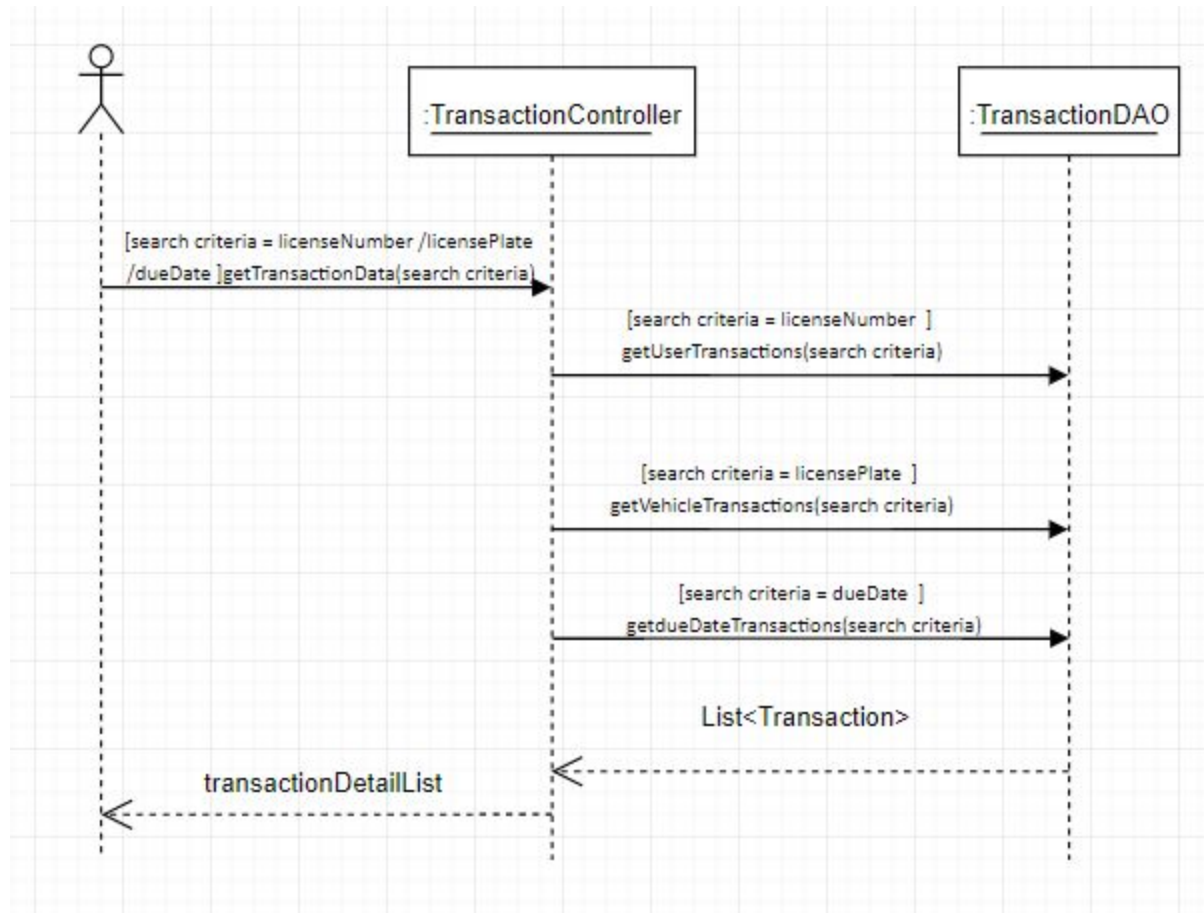
|                                |                 |
|--------------------------------|-----------------|
| Vehicle Renting System         | Version: 3.0    |
| Software Architecture Document | Date:27/10/2019 |



9.5.0 :- UC10:

1. The list of instances of Transactions will be created.
2. All the transaction history records with respect to search criteria will be fetched from the database.
3. The instances attributes will be updated with data from database.
4. Then data will be displayed on the screen.

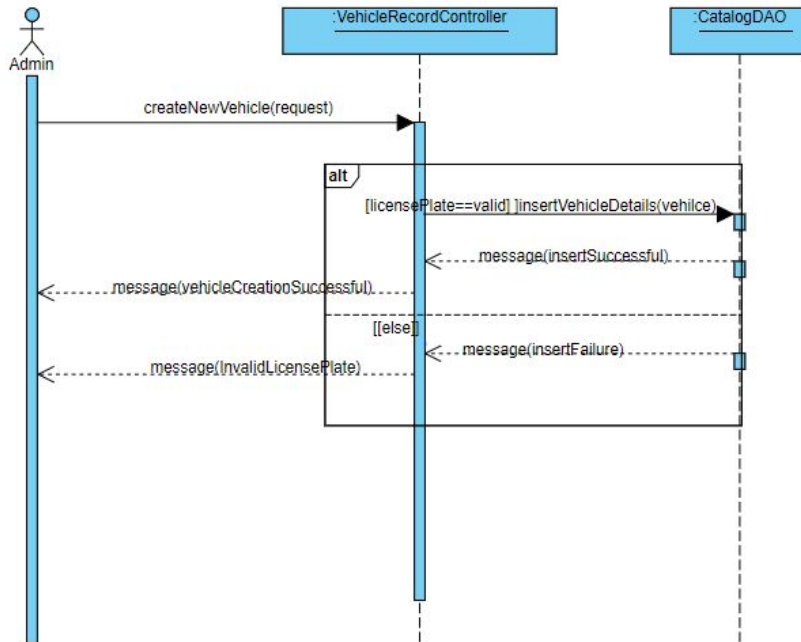
|                                |                 |
|--------------------------------|-----------------|
| Vehicle Renting System         | Version: 3.0    |
| Software Architecture Document | Date:27/10/2019 |



#### 9.5.1 :- UC11:

1. An instance of Catalog details will be created.
2. The input details provided by the administrator is validated.
3. Given details are inserted into the database, provided the license plate is not duplicated.
4. A result message is displayed at the same page accordingly.

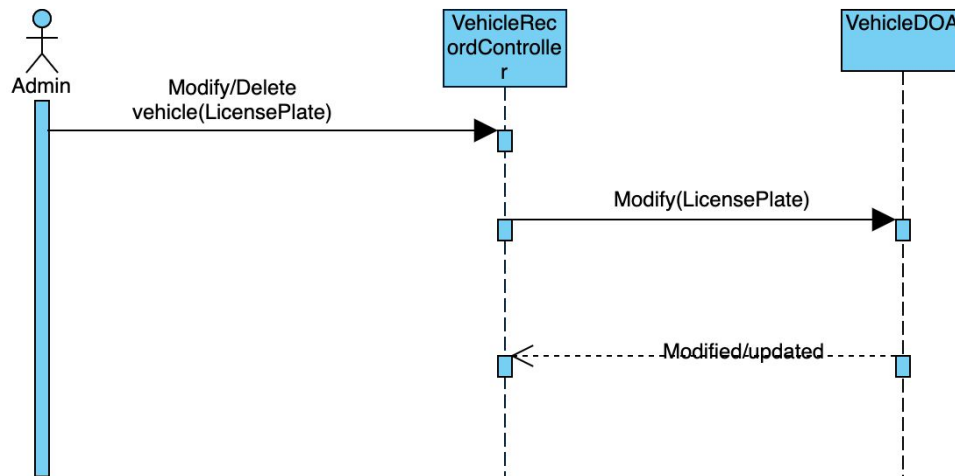
|                                |                 |
|--------------------------------|-----------------|
| Vehicle Renting System         | Version: 3.0    |
| Software Architecture Document | Date:27/10/2019 |



#### 9.5.2: UC12

- 1.An instance of Catalog Details will be created.
- 2.Based on the selected vehicle details, information is modified.
3. Selected vehicle record is modified successfully.
4. vehicle record is updated and a result message is displayed.

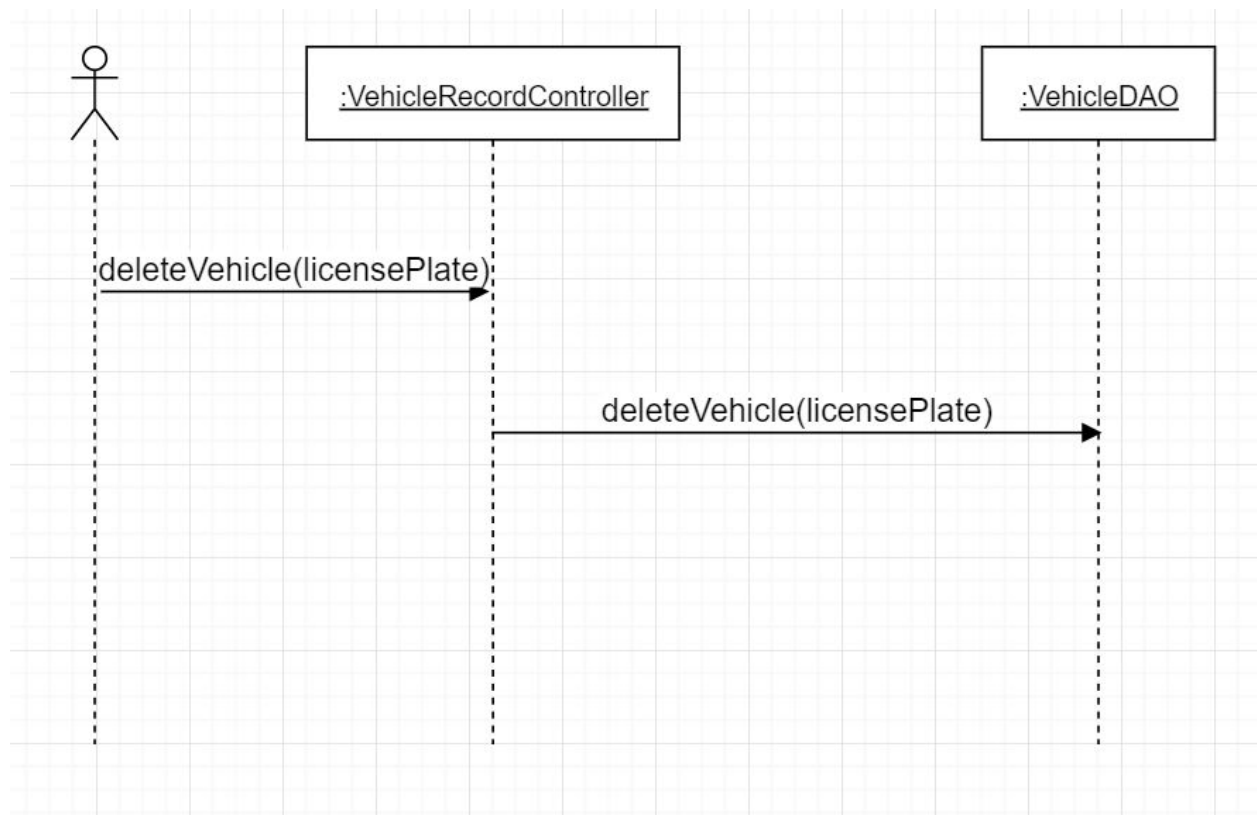
|                                |                 |
|--------------------------------|-----------------|
| Vehicle Renting System         | Version: 3.0    |
| Software Architecture Document | Date:27/10/2019 |



#### 9.5.3:-UC13

1. The instance of vehicle will be deleted.
2. The vehicle record will be deleted from the database.
3. The vehicle record will be deleted from the view catalog page.

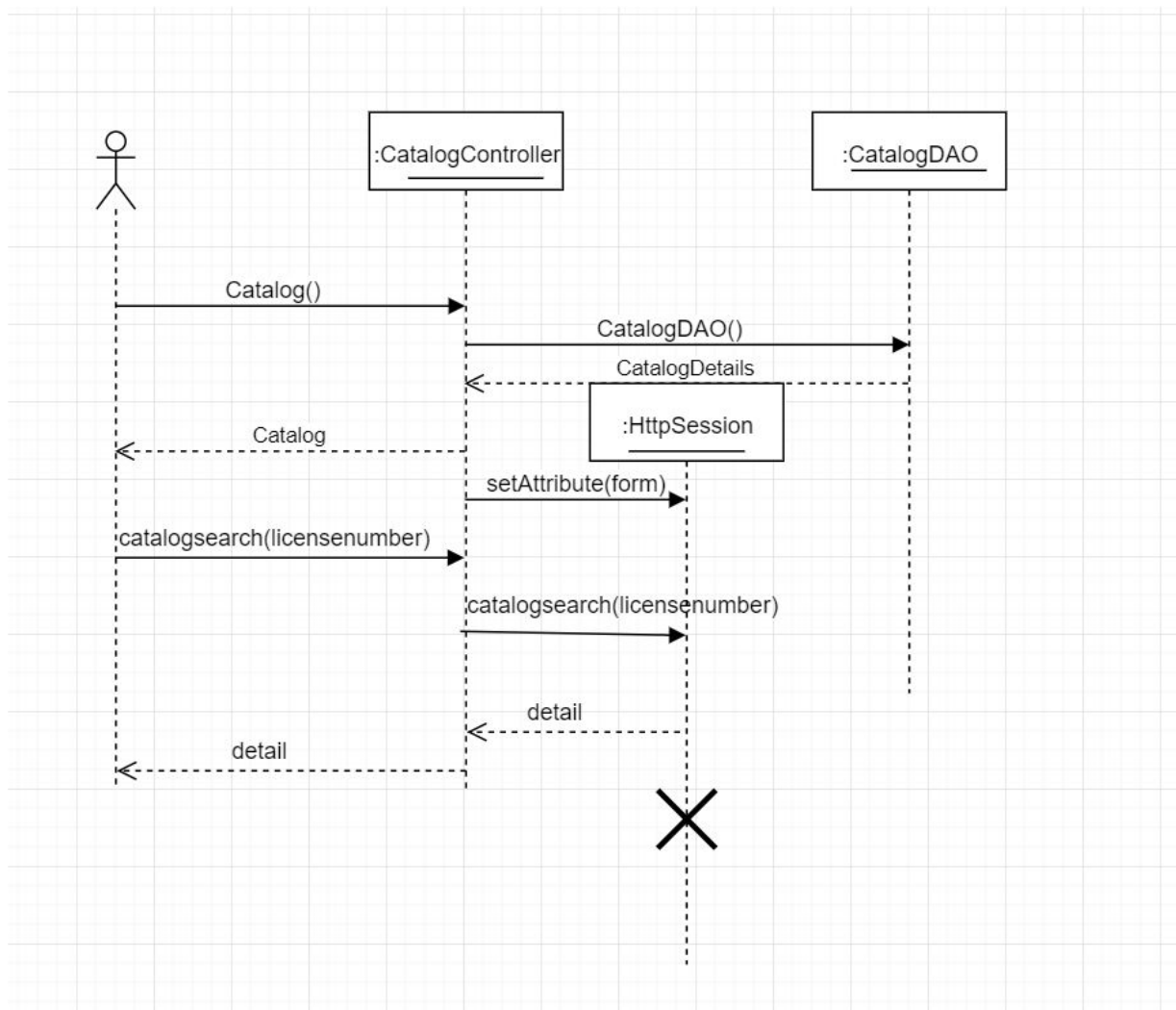
|                                |                 |
|--------------------------------|-----------------|
| Vehicle Renting System         | Version: 3.0    |
| Software Architecture Document | Date:27/10/2019 |



#### 9.5.4:-UC14

1. The instance of List<Catalog> instance is created.
2. instance was associated with Vehicle details table.
3. instance was associated with httpsession.
4. The system displays the desired result on the basis of search criteria.

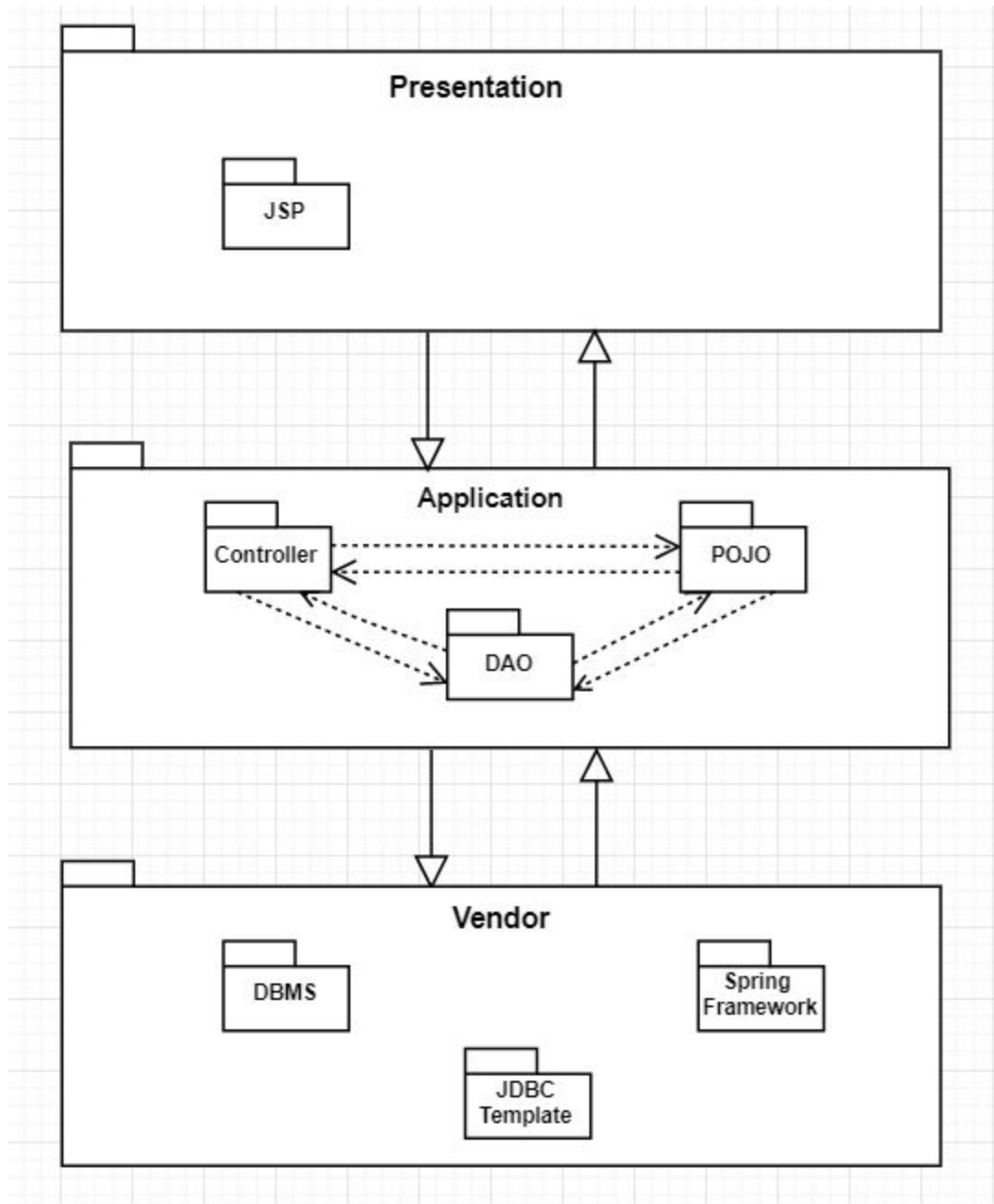
|                                |                 |
|--------------------------------|-----------------|
| Vehicle Renting System         | Version: 3.0    |
| Software Architecture Document | Date:27/10/2019 |





|                                |                 |
|--------------------------------|-----------------|
| Vehicle Renting System         | Version: 3.0    |
| Software Architecture Document | Date:27/10/2019 |

## 10. Development (Implementation) view



### Reuse of components and frameworks

Describe any third-party or home-made components and frameworks that will be reused.

|                                |                 |
|--------------------------------|-----------------|
| Vehicle Renting System         | Version: 3.0    |
| Software Architecture Document | Date:27/10/2019 |

## 11. Process view

The process view deals with the dynamic aspects of the system, explains the system processes and how they communicate, and focuses on the runtime behavior of the system.

## 12. Deployment (Physical) view

The deployment (or physical) view illustrates the physical components of the architecture, their connectors and their topology. Describe the physical network and hardware configurations on which the software will be deployed. This includes at least the various physical nodes (computers, CPUs), the interaction between (sub)systems and the connections between these nodes (bus, LAN, point-to-point, messaging, SOAP, http, https). Use a *UML deployment diagram* to capture this view.

| Name              | Type       | Description               |
|-------------------|------------|---------------------------|
| Name of the node. | Node type. | Technical specifications. |

## 13. Data view (optional)

An enterprise software system would additionally require a data view. The data view describes the data entities and their relationships. Deploy an *Entity-Relationship (ER) Model* to represent this view. Note that the ER model is not part of the UML specification. Additionally you can deploy a UML class diagram to represent the data view where classes would correspond to data entities.

## 14. Quality

|                                |                 |
|--------------------------------|-----------------|
| Vehicle Renting System         | Version: 3.0    |
| Software Architecture Document | Date:27/10/2019 |

A description of how the software architecture contributes to the quality attributes of the system as described in the ISO-9126 (I) standard. **For example:** The following quality goals have been identified:

#### Scalability:

- Description : System's reaction when user demands increase
- Solution : The system supports several workload management techniques.

#### Reliability, Availability:

- Description : Transparent failover mechanism, mean-time-between-failure
- Solution: All possible error exceptions are handled to check that all inputs are in the correct formats using try-catch mechanism and supported by several test cases.
- 

#### Portability:

- Description : Ability to be reused in another environment
- Solution : The system is designed to be easily moved from one computing environment to another, such as browsers and different operating systems.

#### Security:

- Description : Authentication and authorization mechanisms
- Solution : The system has a login form to verify if the clerk and administrator have the authorization for using the system or not, it only allows the registered predefined clerks and admins to use the application, so no malicious or theft could occur.