Software Architecture Document

Version 1.0

for

SOEN-6461-Team 11

Prepared by

Anusha Keralapura Thandavamurthy	40102962	kt.anusha21@gmail.com
Arvind Korchibettu Adiga	40105178	adiga1993@gmail.com
Basant Gera	40082433	basantgera29@gmail.com
Koteswara Rao Panchumarthy	40084998	kotichowdary18@gmail.com
Sai Charan Duduka	40103928	charan140494@gmail.com
Sourabh Rajeev Badagandi	40098471	sourabh.rajeev@gmail.com

Instructor: Dr. C. Constantinides

Course: SOEN-6461

Date: 03-12-2019

Document history

Date	Version	Description	Author
28-September-2019	1.0	Writing the software Architecture used throughout project	Basant Gera
21- october -2019	1.1	Updating Changes / Scenarios in iteration 2/Adding scenarios for Iteration 3	Basant Gera

Table of contents

List of figures

1. Introduction

The document will provide you an overview of the entire S***oftware Architecture*** for VMS (vehicle Renting management system) which help in renting vehicle according to user needs.

Purpose

The document provides you the architectural overview of VRMS (vehicle renting management system).The sole purpose of this management system is Rent vehicle according to user needs so that he can travel on the date he booked his/her reservation of vehicle and return back the same on the date he /she specified.

The document also capture and convey the significant architectural design which have been made in developing and designing the system.The documents tries to convey a system architect should involve in this project for better understanding of the problem which is represented in the system.

Scope

The scope of the document is to highlight the architecture of the VRMS which meets the desired requirements.

Definitions, acronyms, and abbreviations

VRMS	Vehicle Renting management system

2. Architectural representation

Architectural representation can be explained by carrying following objective in mind which are as follows :

1. **Registration** of user on behalf of client/ Administrator [Check if user wants rights to make username as Administrator].
2. **Login** of clerk on behalf of client to book a vehicle according to client's availability.
3. **Checking availability of** vehicle based on various parmenter listed on Vehicle catalog page.
4. Moving to **detailed view** vehicle info and next button functionality.
5. Booking a vehicle for a client with his/her available dates in the **booking form**.
6. **Manage user request** to see which vehicle are available and which are not.
7. Clerk can edit or modify the records for the clients.
8. Clerk can add/edit/delete the record for vehicle entry they are doing.
9. Administrator can add/edit/delete a vehicle from **vehicle master form**.
10. Administrator can view all the transactions done by clerks on **transaction screen** like Modify/Cancel/Returned by Client at the same time updated record will be shown in the transactions.The records can also be filtered out based on Due date/ Start date of vehicle and History per vehicle,client.

Figure 1 illustrated below shows the overall functionality / Design representation as per iteration 2.

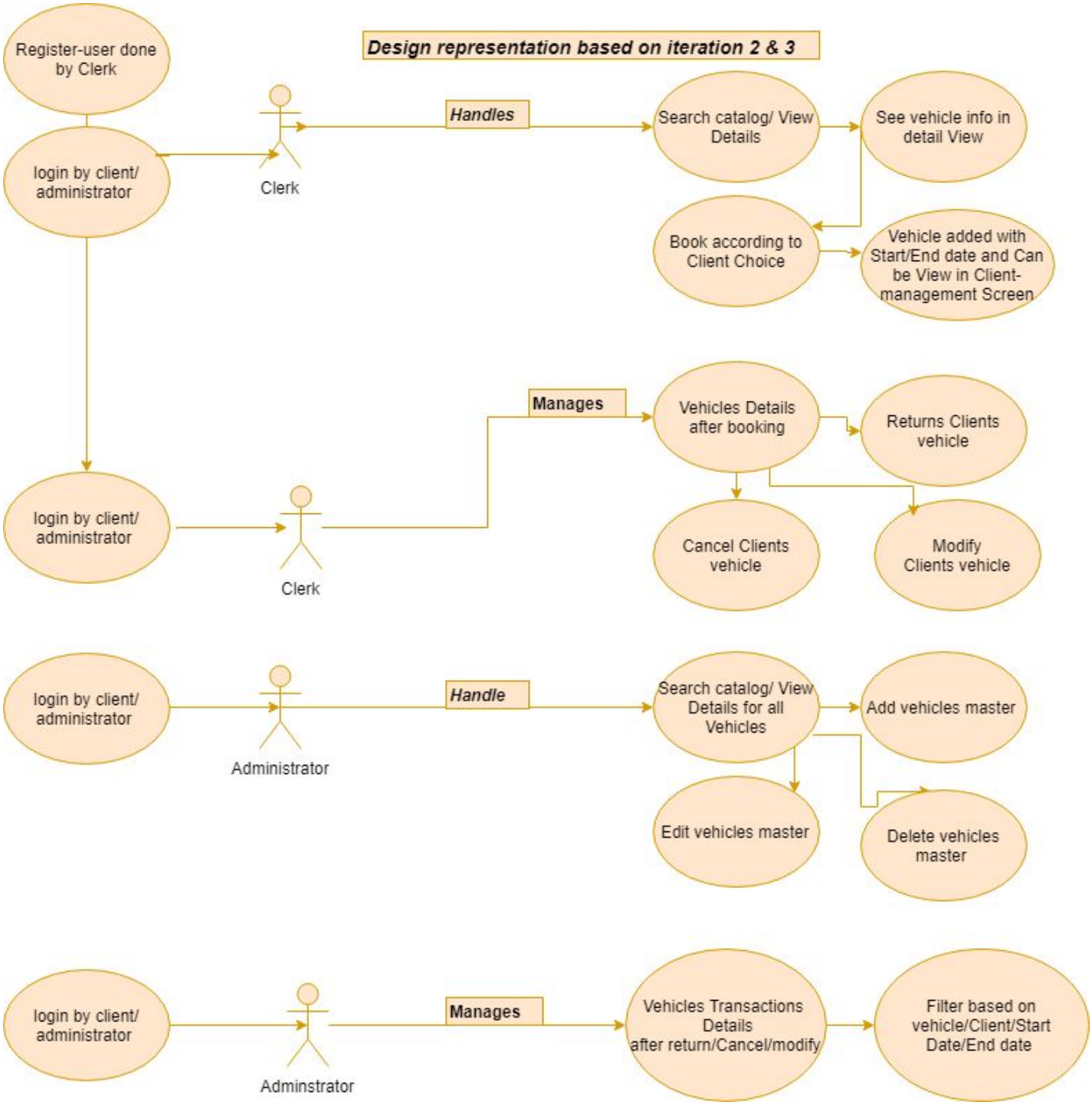

Design representation based on iteration 2 & 3
Register-user done by Clerk
login by client/ administrator
Clerk
Handles
Search catalog/ View Details
See vehicle info in detail View
Book according to Client Choice
Vehicle added with Start/End date and Can be View in Client-management Screen
login by client/ administrator
Clerk
Manages
Vehicles Details after booking
Returns Clients vehicle
Cancel Clients vehicle
Modify Clients vehicle
login by client/ administrator
Administrator
Handle
Search catalog/ View Details for all Vehicles
Add vehicles master
Edit vehicles master
Delete vehicles master
login by client/ administrator
Adminstrator
Manages
Vehicles Transactions Details after return/Cancel/modify
Filter based on vehicle/Client/Start Date/End date

Figure 1

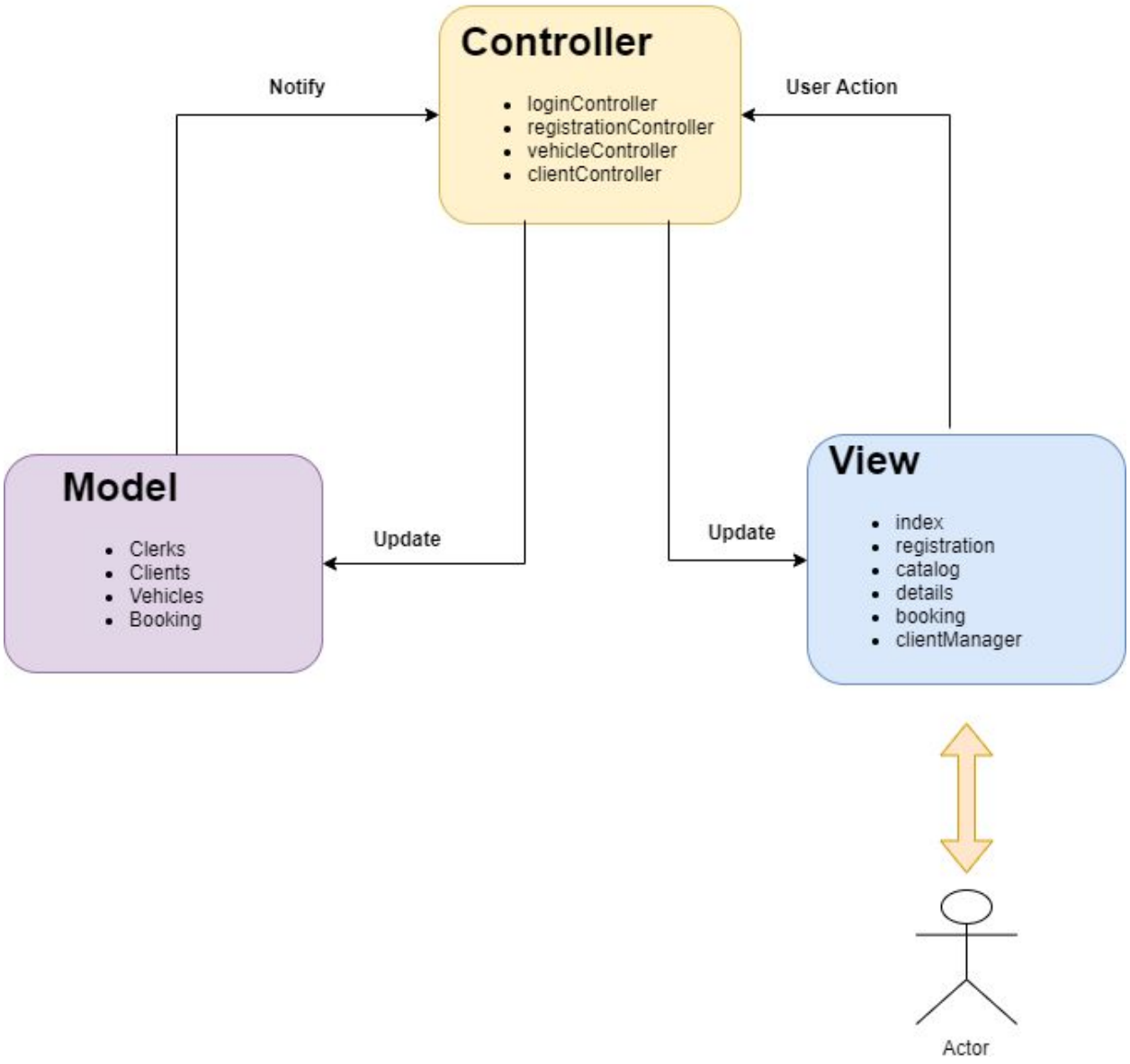


Figure 2: Software Architecture based on iteration 2

For Iteration 2, persistence is not supported, a temporary test class has been created to validate and verify the clerk functionalities that include:

1. Clerk Registration
2. Clerk Login
3. Catalog view with filter and sort options.
4. Vehicle detailed view.
5. Booking a car for the client.
6. Managing client records that include: handling return, cancellation and modification of order.
7. **Logical view** : Designers / **For iteration 3**

   Audience :

   Area Concerned :

Audience: Designers. The logical view is concerned with the functionality that the system provides to end-users. UML Diagrams used to represent the logical view include **Class diagram**, and **interaction diagrams** (**communication diagrams**, or **sequence diagrams**).

8. **Development view** : / **For iteration 3**

   Audience : Programmers

   Area Concerned :

(also known as Implementation view): Audience: Programmers. The development view illustrates a system from a programmer's perspective and is concerned with software management. This view is also known as the implementation view. It uses the UML Component diagram to describe system components. UML Diagrams used to represent the development view include the **Package diagram**.

9. **Process view** : / **For iteration 3**

   Audience : Integrators

   Area Concerned :

: Audience: Integrators. 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. The process view addresses concurrency, distribution, integrators, performance, and scalability, etc. UML Diagrams to represent process view include the **Activity diagram**.

10. **Physical view / For iteration 3**

    Audience : Deployment managers

    Area Concerned :

11. (also known as deployment view) : Audience: Deployment managers. The physical view depicts the system from a system engineer's point of view. It is concerned with the topology of software components on the physical layer, as well as the physical connections between these components. UML Diagrams used to represent physical view include the **Deployment diagram**. /**For iteration 3**

12. **Use case view**

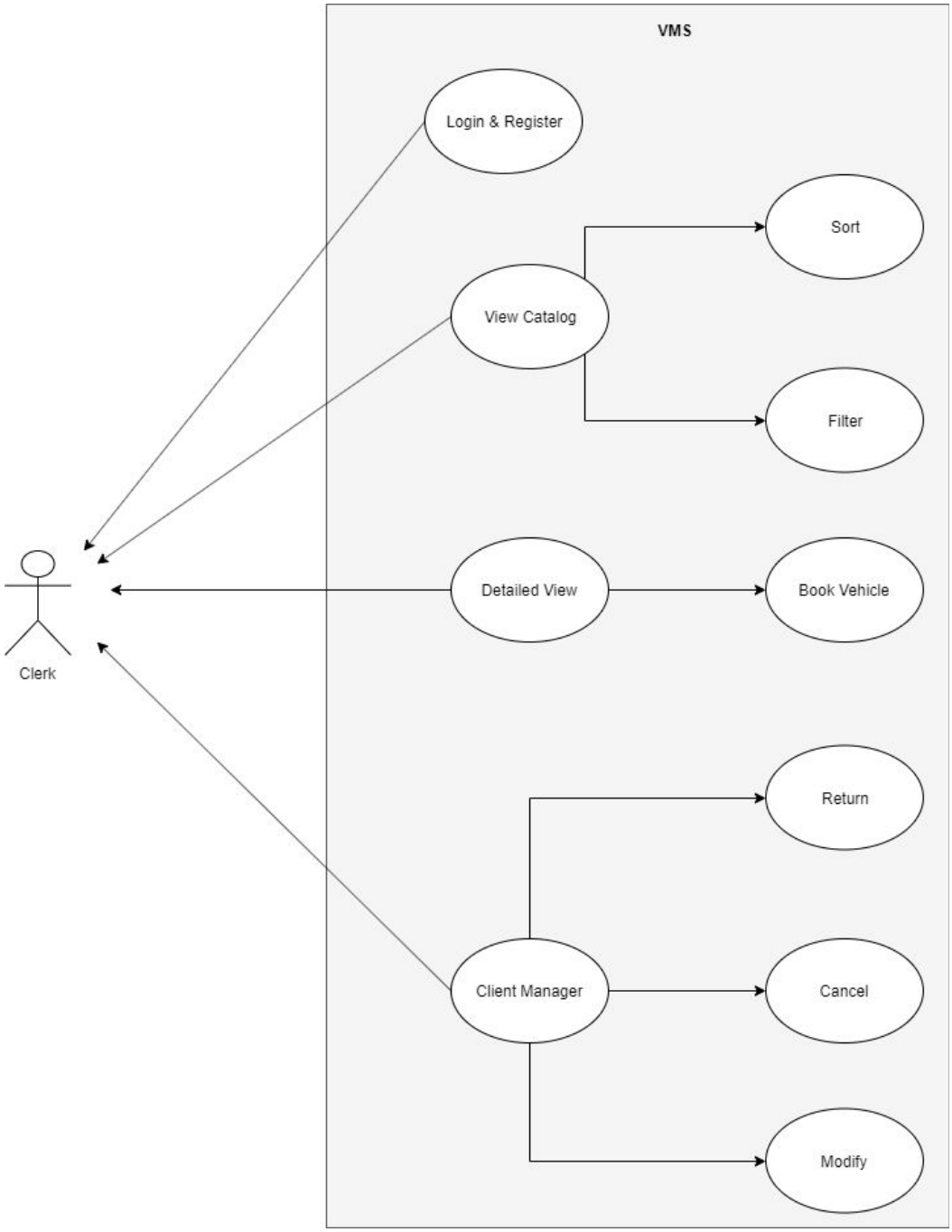


Figure 3. **USE CASE VIEW OF VRMS**

1. **Login and Registration:** This use case describes how a user can register and login to the Vehicle Management System. Username and Password is used for

authorization to gain access into the system.Encryption was maintained for the password.

2. **View Catalog:** This use case describes how clerks can view the vehicles added to the system. It also supports filtering and sorting of the vehicle list to enhance user interaction.
3. **Detailed View:** This use case allows the clerk to view the complete details of the selected vehicle(Make, Model, Type, Year, Color, Licence Plate) and also check if it is available or not. It also supports a navigation button to navigate vehicles in detailed view.
4. **Book Vehicle:** This use case allows a clerk to reserve a vehicle for his client, booking is done by storing all the necessary information of client(first name, last name, licence number, licence validity, and phone number). The system also maintains a timestamp of when the booking was done.
5. **The Client Manager :** This use case describes how client records can be managed. Return of vehicle , cancellation of a booking and modification of a booking is handled in this use case.
6. **Vehicle Master** : This use case describes how administrator can add vehicles to the system. It also supports filtering vehicle list to enhance user interaction.
7. **Transactions Screens** : This use case describes how administrator can view the information which is maintained and viewed on transactions screens. It also supports filtering the list via various parameter sp that administrator can view overall current availability of the vehicle.

Audience : All the stakeholders of the system

Area Concerned :

(also known as Scenarios) : Audience: all the stakeholders of the system, including the end-users. The description of the architecture is illustrated using a small set of use cases, or scenarios which become a fifth view. The scenarios describe sequences of interactions

between objects, and between processes. They are used to identify architectural elements and to illustrate and validate the architecture design. They also serve as a starting point for tests of an architecture prototype. Related Artifacts : **Use-Case Model**. /**For iteration 3**

13. Audience: Data specialists, Database administrators. Describes the architecturally significant persistent element s in the data model . Related Artifacts: **Data model**.
14. 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. **For iteration 3**

Functional requirements (Use case view)

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

Source	**Name**	**Architectural relevance**	**Addressed in:**
Use case(s) or scenario(s).	**Name of case(s) or scenario(s).**	**Description on why this use case or scenario is relevant to the architecture.**	**Section number where this use case or scenario is addressed in this document.**
User Login	User able to login via username and password than :	We selected Login page because we need to store who log in and maintain the	**SRS Document Figure:*4.2***

	1.**Success** : Logged In successfully. 2.**Failure** : Not able to login since username and password didn't not matched and redirected back to the login page.	user name for each user so that we can maintain the history and know what a user in doing after logging and booking a vehicle on behalf of someone.	
User Registration	User can register his information based on which general information is asked by the software.And than can try for login. 1. **Success** : Able to provide all the details to register basic information. 2.**Failure** :If not saved properly can not able to login.	Since user can logged in it should have a master when he/she can give his/her basic details and than can perform log In.	**SRS Document** **Figure: Use case:4.2**
Vehicle Searching page	User can search based on the following selections : Make,Type,Year and Model.Following things will happen :	A user after logging will be landed on the vehicle search page.Where he can check what kind of vehicle he/she would	**SRS Document** **Figure:*UseCase 4.2, Figure 4.8***

	1.You can select search button and filter based on the following condition. 2.Apart from the dataset which comes based on filtering can be sorted based on pressing the button respectively.Like ascending/Descending based on [a-z],[z-a] [alphabets] and [Lowest-Highest] ,[highest-lowest] [numbers]	be looking for according to his choice or can perform sorting based on alphabets on make model type and year and can be checked and can be opened or select via view details button.Since a user looking for a vehicle can look for book and he/she can book on clicking view details button.	
Detailed View	Based on view detail button user can user can check his/her vehicle detail and can also go back to the screen he came from or confirm the booking bu going on the booking view page.	In Detailed view you can edit/add the vehicle details and check the availability also.since checking the availability is an important aspect and you can proceed to click on book now	**SRS Document** **Figure:*UseCase 4.1***

		where you can give start and end date.	
Booking View	user can put the start and return date and can book the vehicle.	In booking view you need to give user general info and the most important things the start and return date of the vehicle when you are renting for a particular period.	**SRS Document Figure:*UseCase 4.10***
Client Page	All the reservation of the following vehicles would be shown on the clients page with their dates and user can add/edit-modify/Return vehicles.	Client pages manages booking done by user/clerk. You can add/edit/return/cancel the booking which you have done.This page shows data in tabular table.	**SRS Document Figure: ?**
Clients Modify	Clients can modify/add and return the vehicles on this page.	You can add/edit/return and cancel the vehicle you rented in edit mode.	**SRS Document Figure: ?**
Vehicle Master	Administrator can Filter and then Add/ Modify/Delete the vehicle details.		**SRS Document Figure: ?**

Transaction Screen	Administrator can Filter and view history based on client/Vehicle info or by particular start/end date.		**SRS Document Figure: ?**

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.

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.

Source	**Name**	**Architectural relevance**	**Addressed in:**
e.g. Vision, SRS.	Name of requirement.	Description on why this requirement is relevant to the software architecture.	Section number where this requirement is addressed in this document.

15. 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.

Use Case No.	**Use Case**	**Type**
UC-1	Register	Non-critical
UC-2	Login	Non-critical
UC-3	View Catalog	Non-critical
UC-4	Search Catalog	Critical
UC-5	View Vehicle Details	Non-critical
UC-6	Book Vehicle	Critical
UC-7	Manage Client Records	Critical
UC-8	View Bookings	Non-critical
UC-9	Search Bookings	Critical
UC-10	Manage Vehicle Record	Critical

Logical view

The logical view captures the functionality provided by the system; it illustrates the collaborations between system components in order to realize the system's use cases. Describe the architecturally significant logical structure of the system. Think of decomposition in tiers and subsystem. Also describe the way in which, in view of the decomposition, Use Cases are technically translated into Use Case Realizations.

Layers, tiers etc.

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

Subsystems

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

Architecturally significant design packages

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

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.

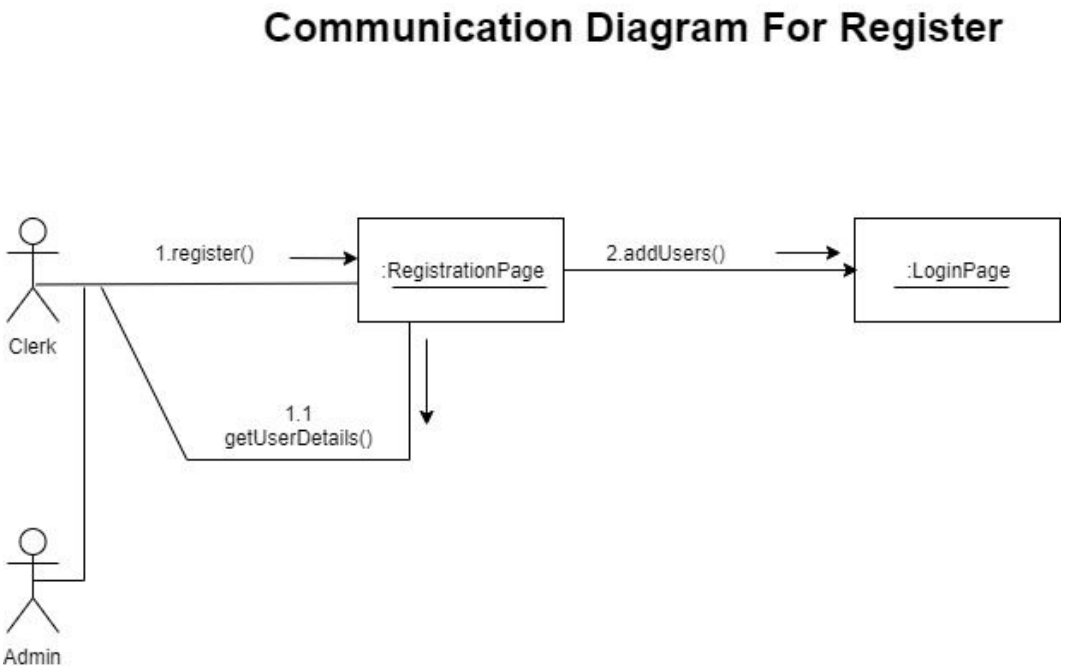


Figure 4: CD-1

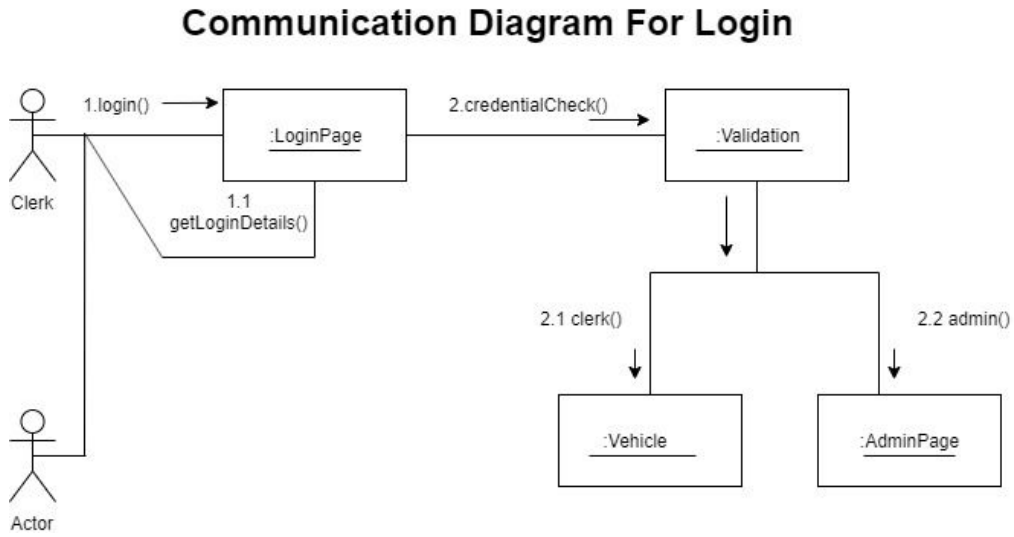


Figure 5: CD-2

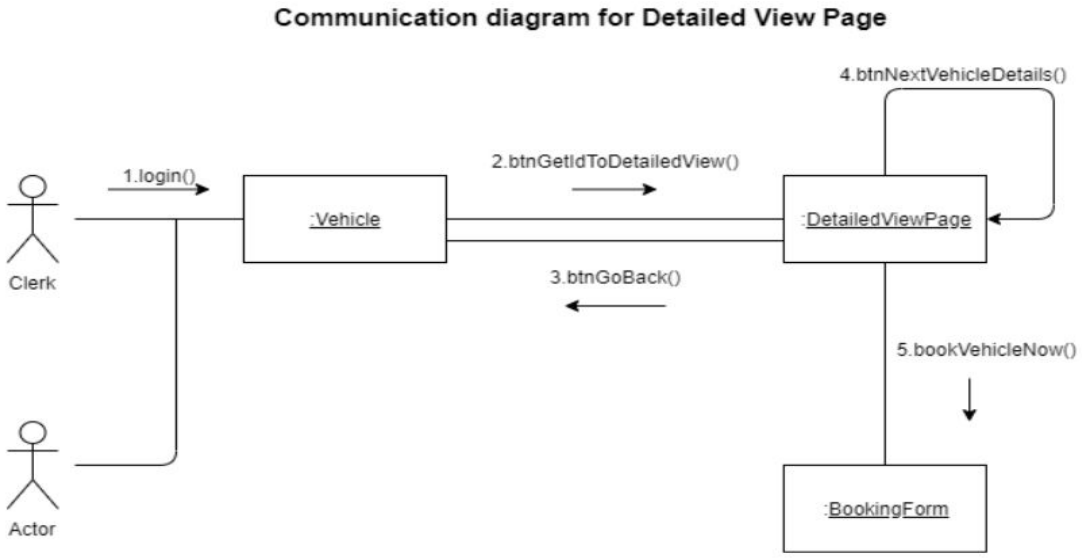

Communication diagram for Detailed View Page
4.btnNextVehicleDetails()
1.login()
2.btnGetIdToDetailedView()
:Vehicle
:DetailedViewPage
Clerk
3.btnGoBack()
5.bookVehicleNow()
Actor
:BookingForm

Figure 6:CD-3

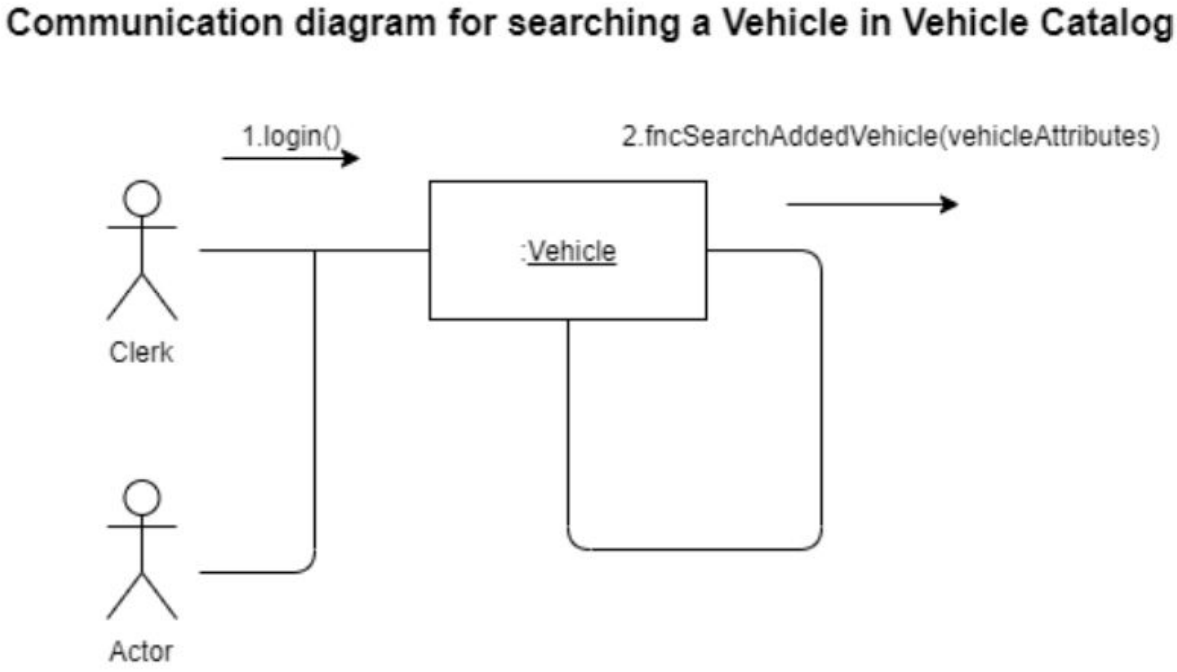

Communication diagram for searching a Vehicle in Vehicle Catalog
1.login()
2.fncSearchAddedVehicle(vehicleAttributes)
:Vehicle
Clerk
Actor

Figure 7: CD-4

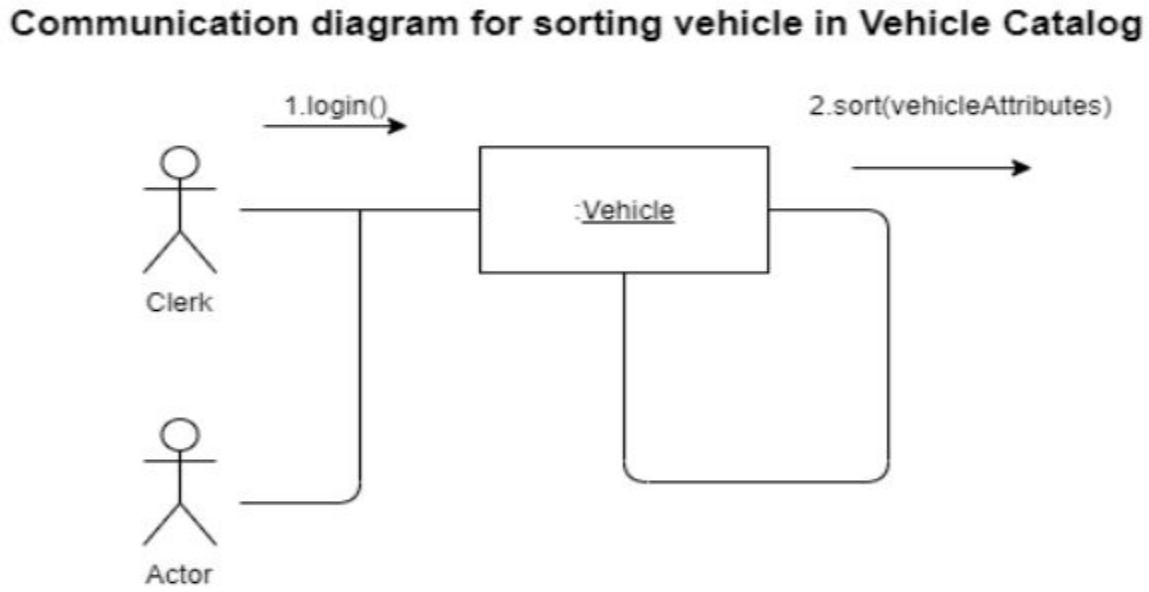


Figure 8: CD-5

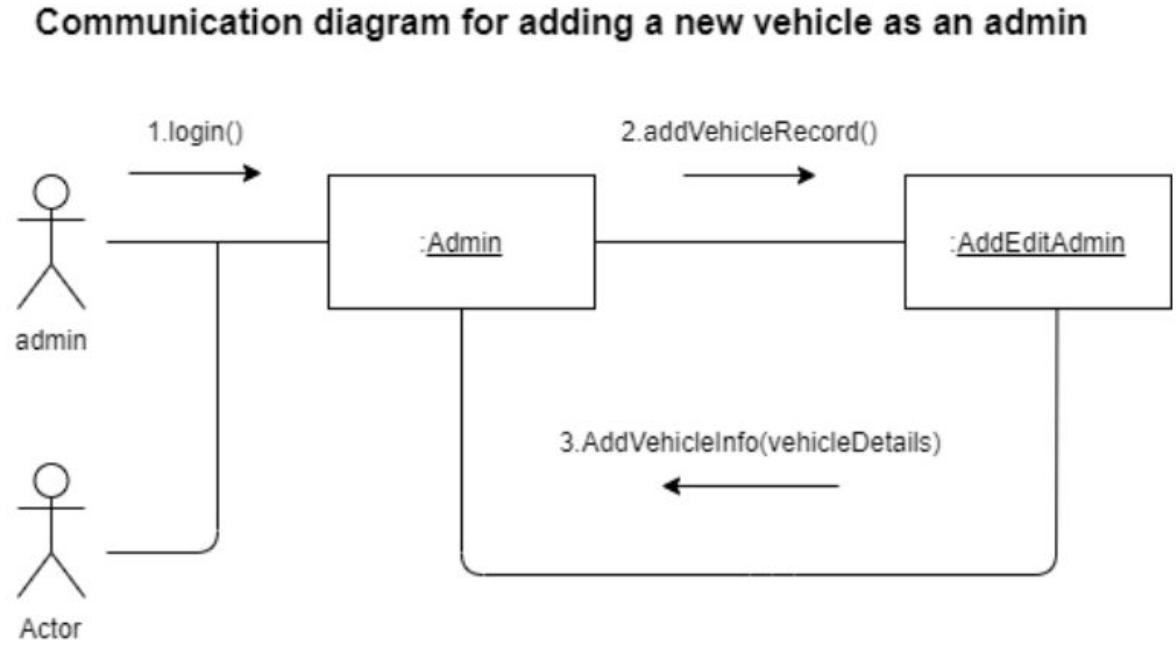


Figure 9: CD-6

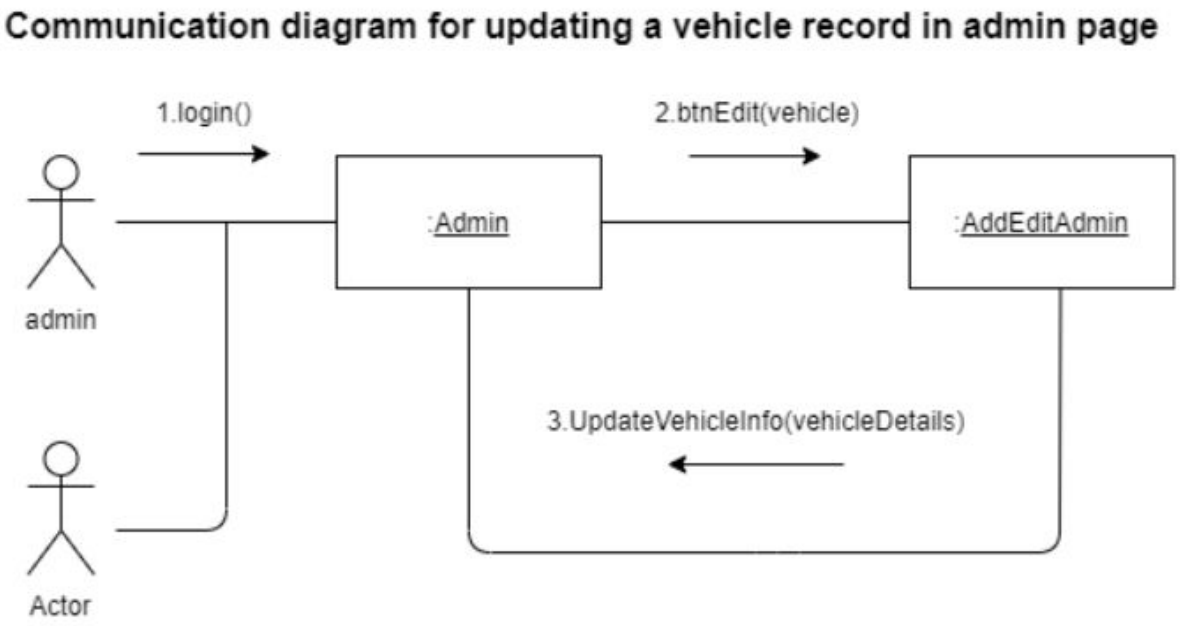


Figure 10: CD-7

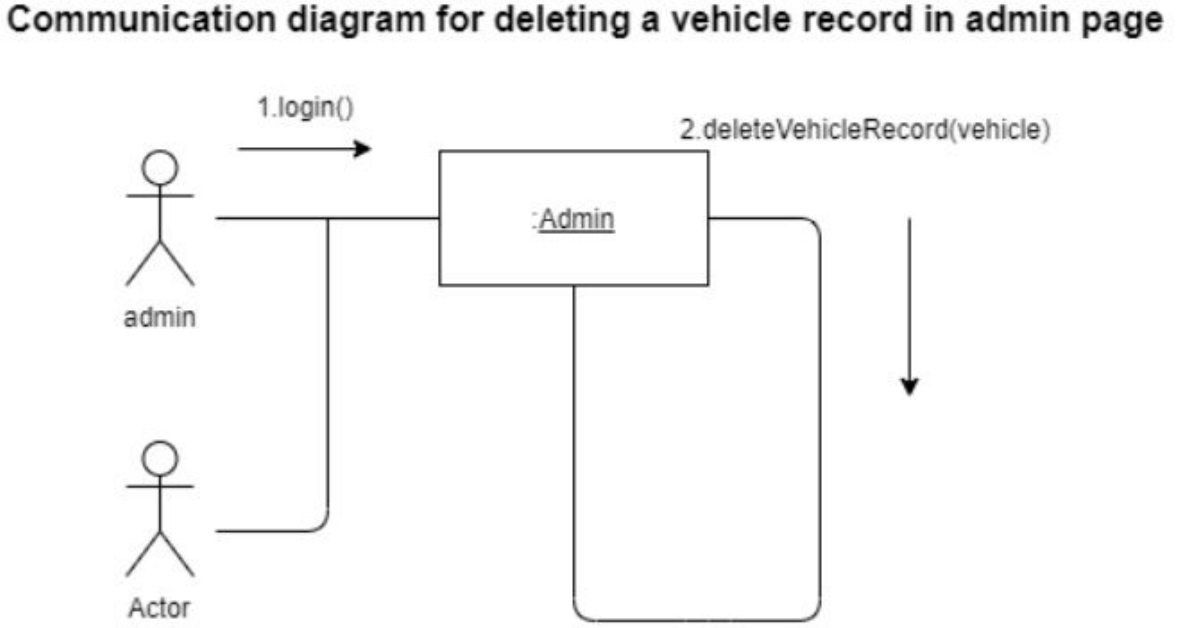


Figure 11: CD-8

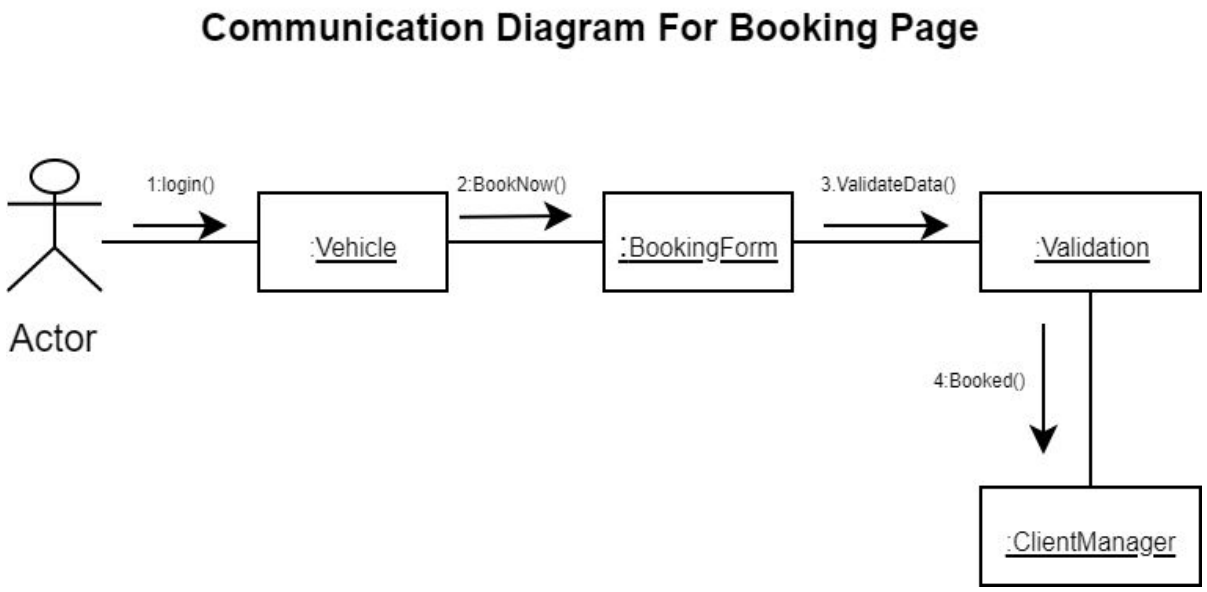


Figure 12: CD-9

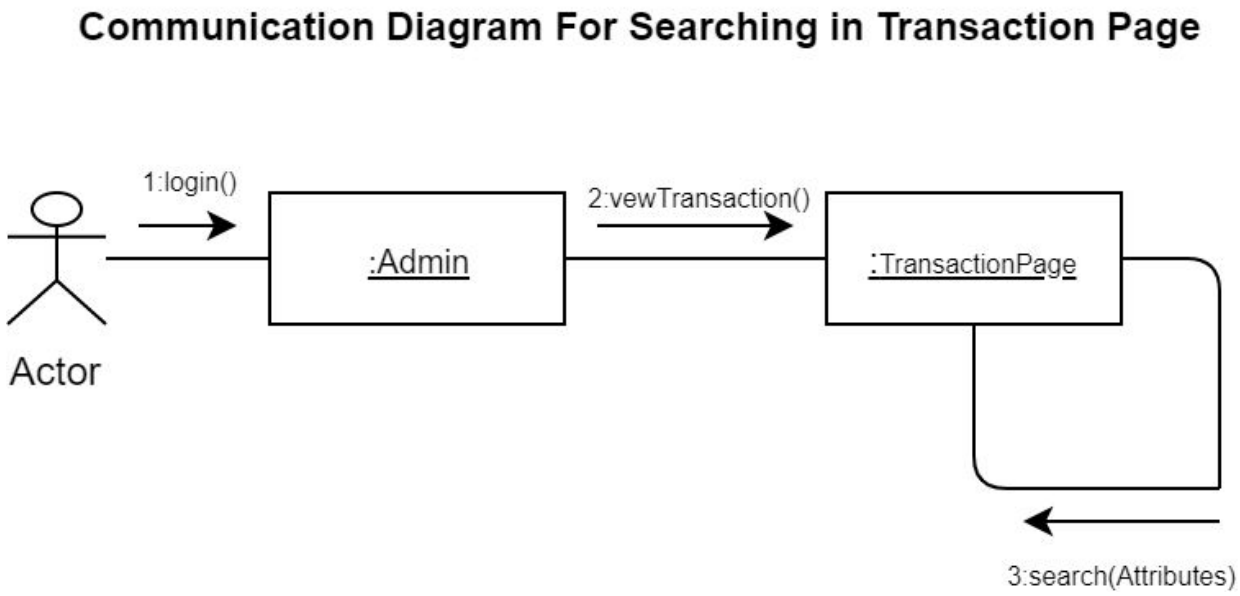


Figure 13: CD-10

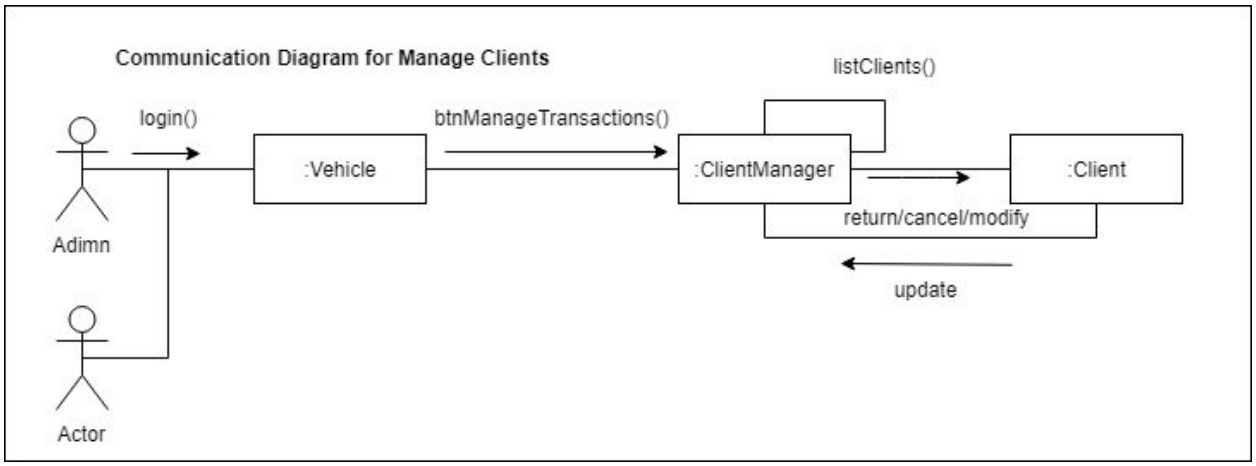


Figure 14: CD-11

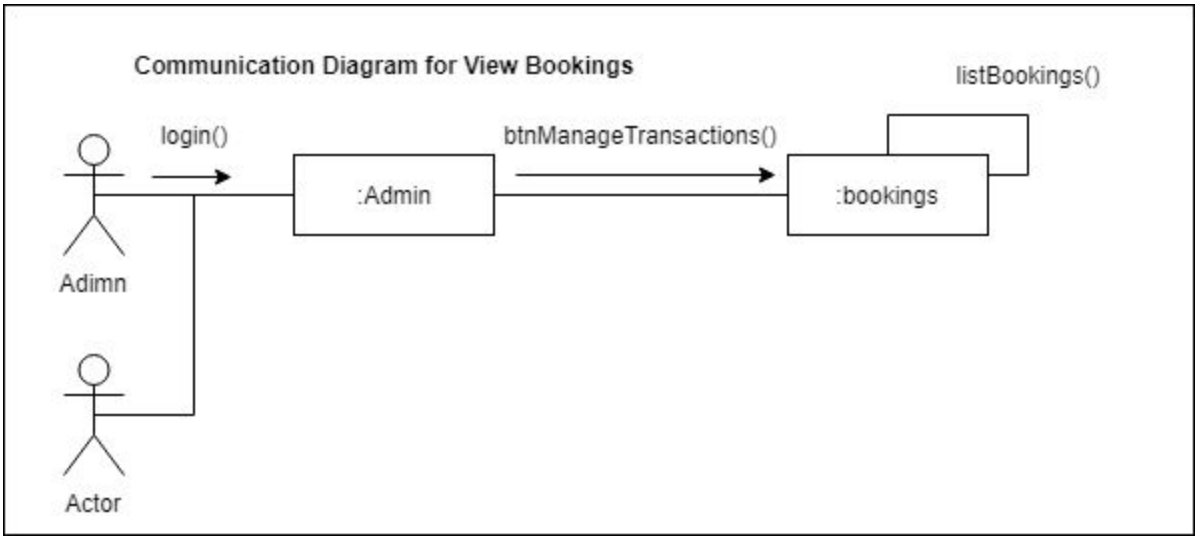


Figure 15: CD-12

Functional Requirements and Use case Mappings:

		USE CASES									
		UC-1	UC-2	UC-3	UC-4	UC-5	UC-6	UC-7	UC-8	UC-9	UC-10
FUNCTIONAL REQUIREMENTS	REQ-001	X									
	REQ-002		X								
	REQ-003			X							
	REQ-004				X						
	REQ-005				X						
	REQ-006					X					
	REQ-007					X					
	REQ-008							X			
	REQ-009							X			
	REQ-010							X			
	REQ-011							X			
	REQ-012						X				
	REQ-013						X				
	REQ-014										X
	REQ-015									X	
	REQ-016									X	

Figure 16: Functional Requirements and Use Case mapping

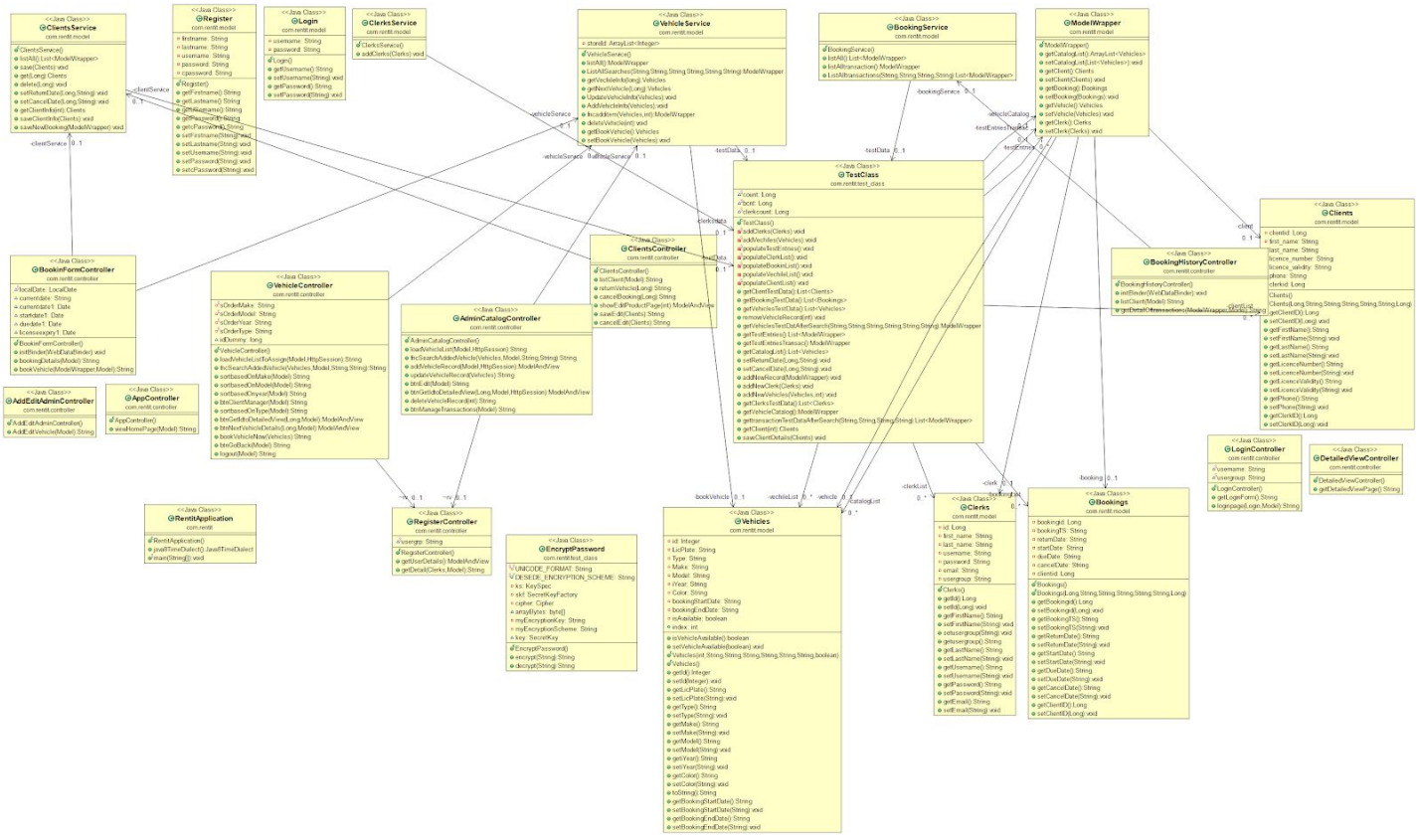
Use case and Communication diagram mapping:

		COMMUNICATION DIAGRAM											
		CD-1	CD-2	CD-3	CD-4	CD-5	CD-6	CD-7	CD-8	CD-9	CD-10	CD-11	CD-12
USE CASES	UC-1	X											
	UC-2		X										
	UC-3					X							
	UC-4				X	X							
	UC-5			X									
	UC-6									X			
	UC-7											X	
	UC-8												X
	UC-9										X		
	UC-10						X	X	X				

Figure 17: Use Case and Communication diagram mapping

16. Development (Implementation) view

The development (or implementation) view describes the components used to assemble the system. Use a *UML component diagram* to capture this view.



Reuse of components and frameworks

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

17. Process view

The process view illustrates the system's processes, focusing on the runtime behavior of the system. The view illustrates parallelism and concurrency. Deploy a *UML activity diagram* to capture this view.

18. Deployment (Physical) view

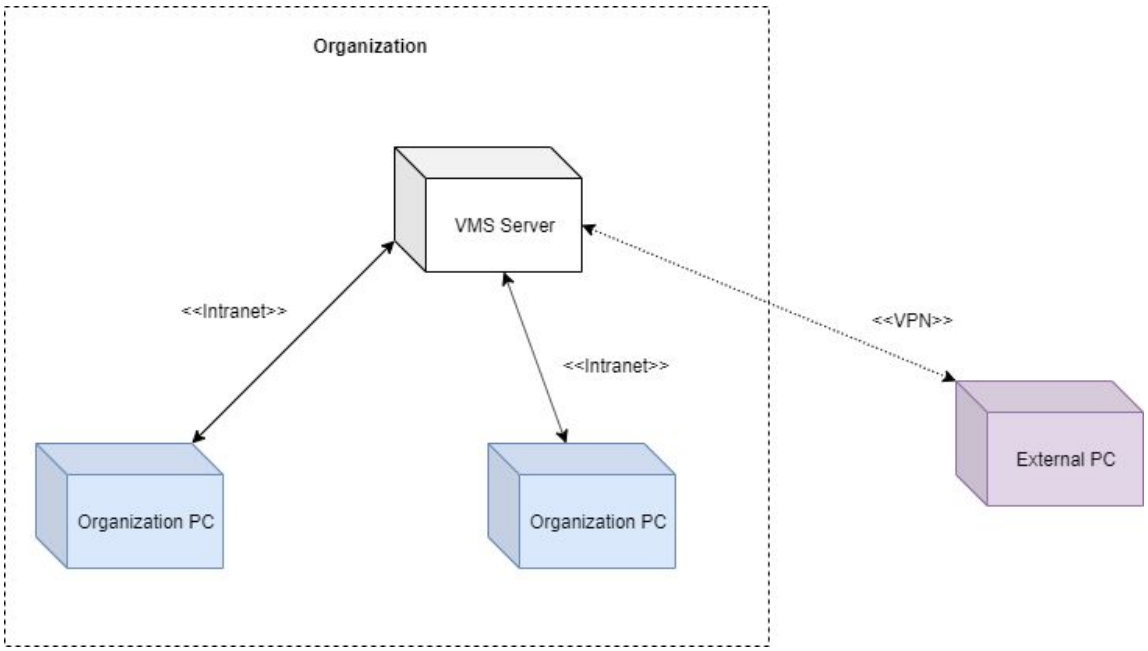


Figure 12.

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, http). Use a *UML deployment diagram* to capture this view.

Name	**Type**	**Description**
Name of the node.	Node type.	Technical specifications.

19. 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.

20. Quality

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 : J2EE application servers support several workload management techniques

Reliability, Availability:

- Description : Transparent failover mechanism, mean-time-between-failure
- Solution : : J2EE application server supports load balancing through clusters

Portability:

- Description : Ability to be reused in another environment
- Solution : The system me be fully J2EE compliant and thus can be deployed onto any J2EE application server

Security:

- Description : Authentication and authorization mechanisms
- Solution : J2EE native security mechanisms will be reused.