Software Architecture Document

Version 1.0

WEB-BASED CAR RENTAL BROKER MANAGEMENT SYSTEM

Prepared by

Name	Student-id	Email
Kiranmayie	40092284	2809kiran@gmail.com
Rajasekhar	40094479	rajasekhar.grr@gmail.com
Sahana	40092026	Sahana15shankar@gmail.com
Nandini	40105415	nandu.angel555@gmail.com
Vasu Dadhania	40103048	vasudadhania@gmail.com

Instructor:	Dr.Constantinos Constantinides
Course:	Software Design Methodologies (SOEN 6461-SS)
Date:	<09/18/2019>

Document history

Date	Version	Description	Author
09/15/2019	1.0	Introduction	VASU DADHANIA
10/05/2019	1.1	UML diagrams	NANDINI BANDLAMUDI
10/27/2019	2.0	Admin	Nandini Bandlamudi

Table of content

SOFTWARE ARCHITECTURE DOCUMENT

1.INTRODUCTION :

This document provides a high level overview and explains the architecture of the Distributed Development Monitoring and Mining system.
The document defines goals of the architecture, the use cases supported by the system, architectural styles and components that have been selected. The document provides a rationale for the architecture and design decisions made from the conceptual idea to its implementation.

1.1 PURPOSE :

The Software Architecture Document (SAD) provides a comprehensive architectural overview of the web based car rental broker management (CRBM). It presents a number of different architectural views to depict different aspects of the system.

1.2 SCOPE :

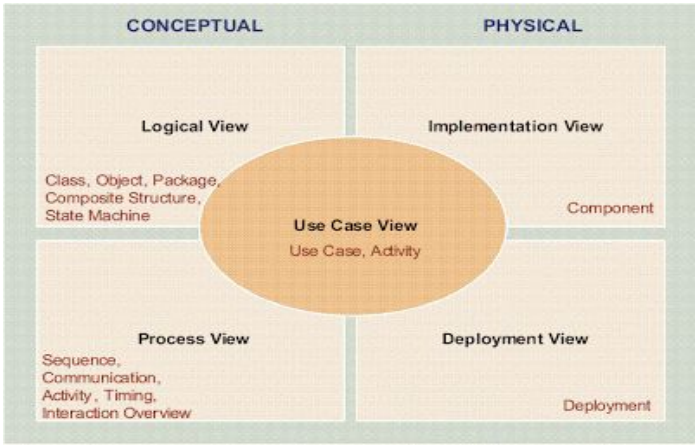
The scope of this SAD is to explain the architecture of the web based car rental broker management . This document describes the various aspects of the CRBM system design that are considered to be architecturally significant. These elements and behaviors are fundamental for guiding the construction of the CRBM system and for understanding this project as a whole. Stakeholders who require a technical understanding of the CRBM system are encouraged to start by reading the Project Proposal, Concept of Operations and Software Requirements Specification documents developed for this system .

1.3 DEFINITION ,ACRONYMS AND ABBREVIATIONS

- **Apache** – Web Server
- **ASP.NET** - Microsoft web platform
- **HTTP** – Hypertext Transfer Protocol
- **WWW** – World Wide Web
- **SAD -** Software Architecture Document
- **UML** – Unified Modeling Language
- **User -** This is any user who is registered on the CRBM website

2.Architectual Representation

This document details the architecture using the views defined in the “4+1” model [Kruchten]. The views used to document the CRBM system are:



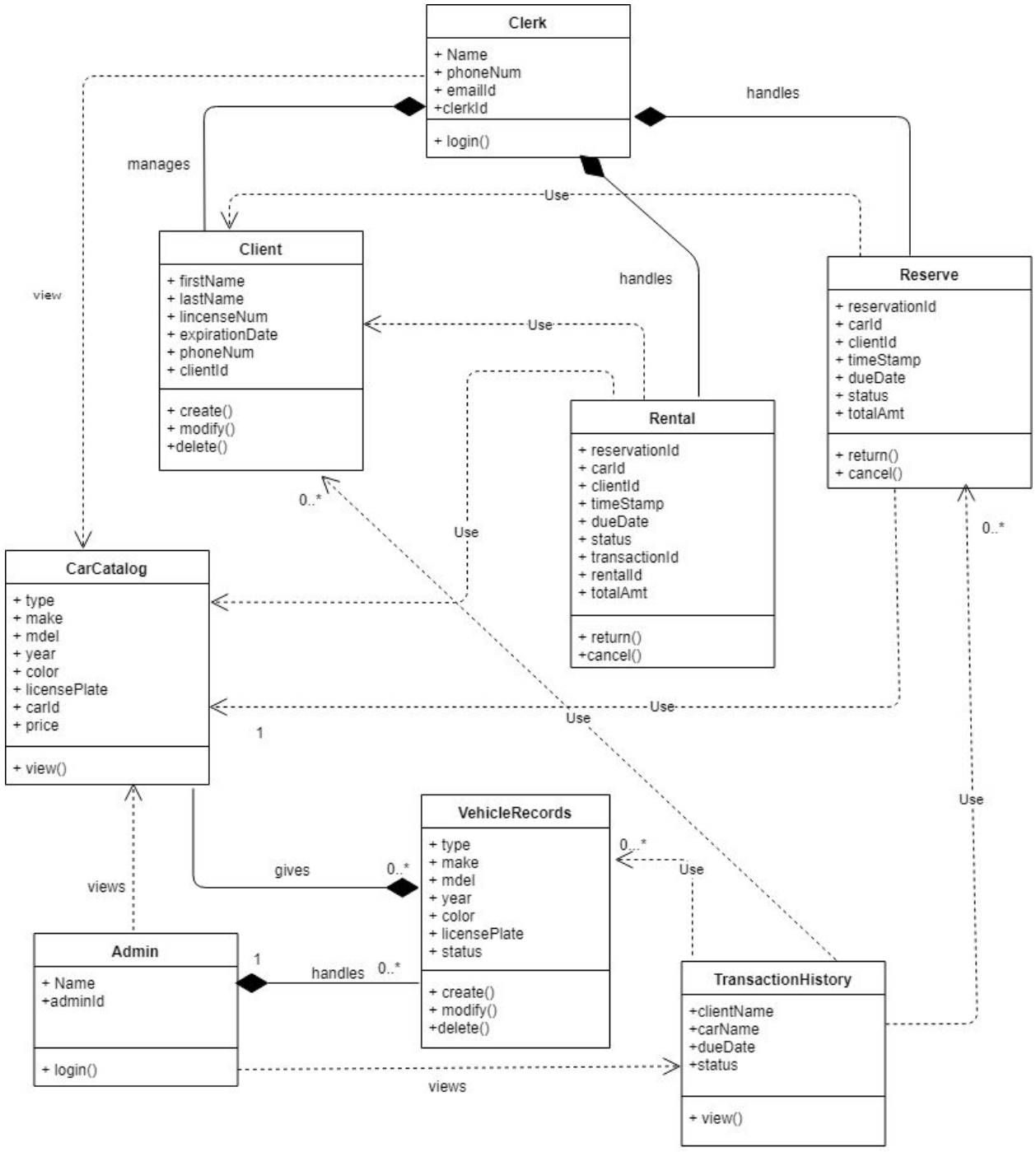
2.1 LOGICAL VIEW :

Audience: Designers.

Area: Functional Requirements: describes the design's object model. Also describes the most important use-case realizations and business requirements of the system. Logical view also concerned with the

functionality that the system provides to end users.

Related Artifacts: Design model (Class Diagram , Interaction Diagram)



2.2 DEPLOYMENT VIEW :

Audience: Deployment managers.

Area: Topology: describes the mapping of the software onto the hardware and shows the system's distributed aspects. Describes potential deployment structures, by including known and anticipated deployment scenarios in the architecture we allow the implementers to make certain assumptions on network performance, system interaction and so forth.

Related Artifacts: Deployment Diagram

2.3 PROCESS VIEW :

Audience: Senior managers.

Area: 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 .

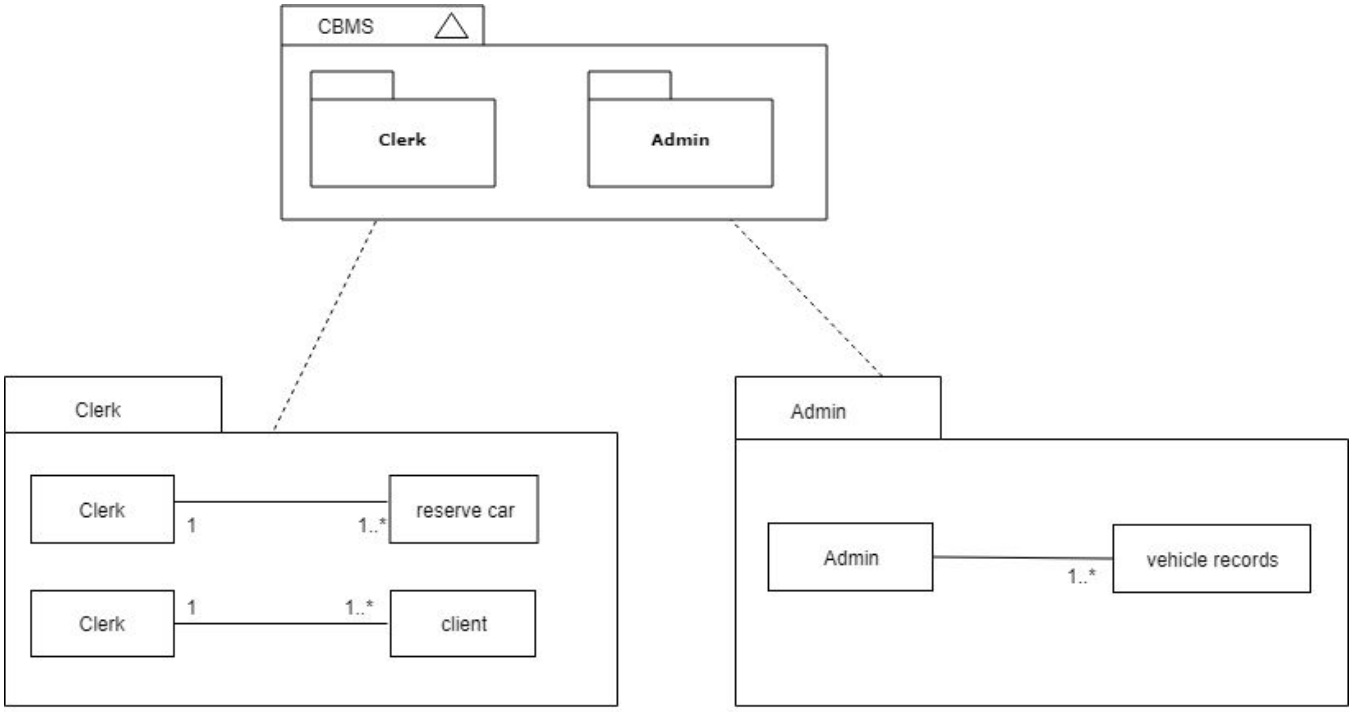
Related Artifacts: Activity Diagram

2.4 IMPLEMENTATION VIEW :

Audience: Developer .

Area: 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.

Related Artifacts: Package Diagram



2.5 USE CASE VIEW :

Audience: all the stakeholders of the system, including the end-users.

Area: describes the set of scenarios and/or use cases that represent some significant, central functionality of the system. Describes the actors and use cases for the system, this view presents the needs of the user and is elaborated further at the design level to describe discrete flows and constraints in more detail. This domain vocabulary is independent of any processing model or representational syntax (i.e. XML).

Related Artifacts : Use-Case Model, Use-Case documents .

3. ARCHITECTURE REQUIREMENTS :

3.1 FUNCTIONAL REQUIREMENTS:

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)

SOURCE	NAME	ARCHITECTURAL RELEVANCE	ADDRESSED IN
use case	create	Clerk Functionality	section 4
use case	rent	Clerk Functionality	section 4
use case	return	Clerk Functionality	section 4

(Functional Requirements)

3.2 NON FUNCTIONAL REQUIREMENTS :

In systems engineering and requirements engineering, a non-functional requirement (NFR) is a requirement that specifies criteria that can be used to identify the operation of a system, rather than specific behaviors. i.e. those which are important for developing the software architecture.

Some typical non functionality are performance (Response Time, Throughput) , Scalability , Secure , Data integrity , interoperability and many more .

SOURCE	NAME	ARCHITECTURAL RELEVANCE	ADDRESSED IN
users	Performance	The Web Application responds within 0.5 to 1 sec	
vision	scalability	The Web Application create any number of	section 2

		clients	
users	security	Client has been authenticated	section 5

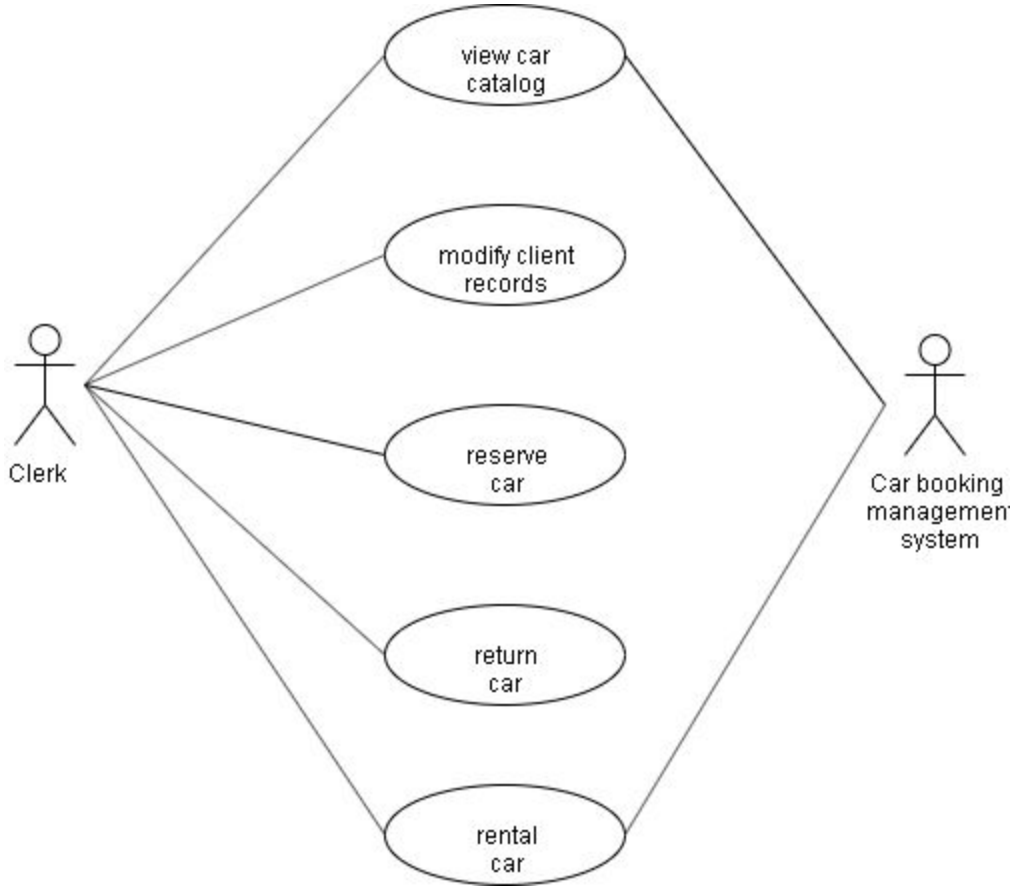
(Non Functional Requirements)

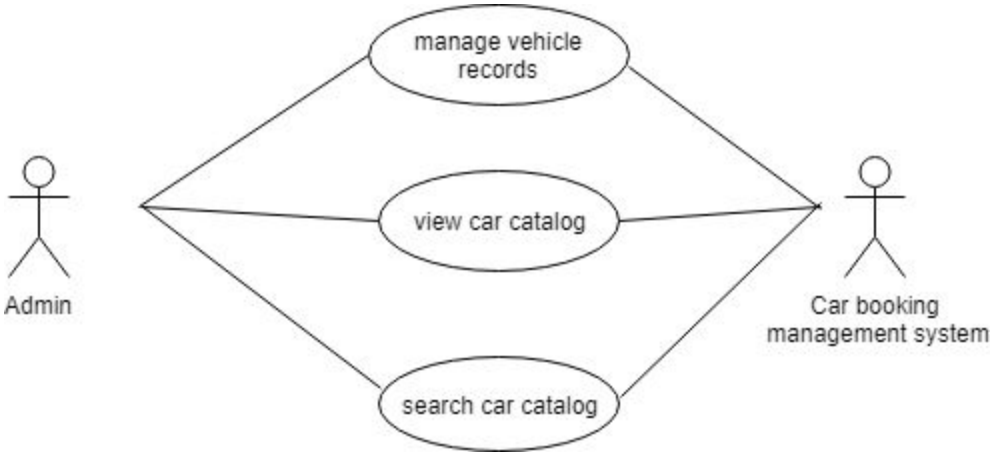
4 . USE CASE VIEW :

The Use Case diagram is one of the Unified Modeling Language (UML) Behavioral diagrams that can be used to describe the goals of the users and other systems that interact with the system that is being modeled. They are used to describe the functional requirements of a system, subsystem or entity and present a simple but compelling picture of how the system will be used.

The Use Case diagram is used to describe the goals that users or other systems want to achieve from interacting with the system. They always describe the goal from the Actors' perspective, the details of the Use Case will describe the goal with more precision. Use Cases will often act as the basis for the definition of Test Cases.

Use case diagram for the CRBM is shown as : (use case diagram)

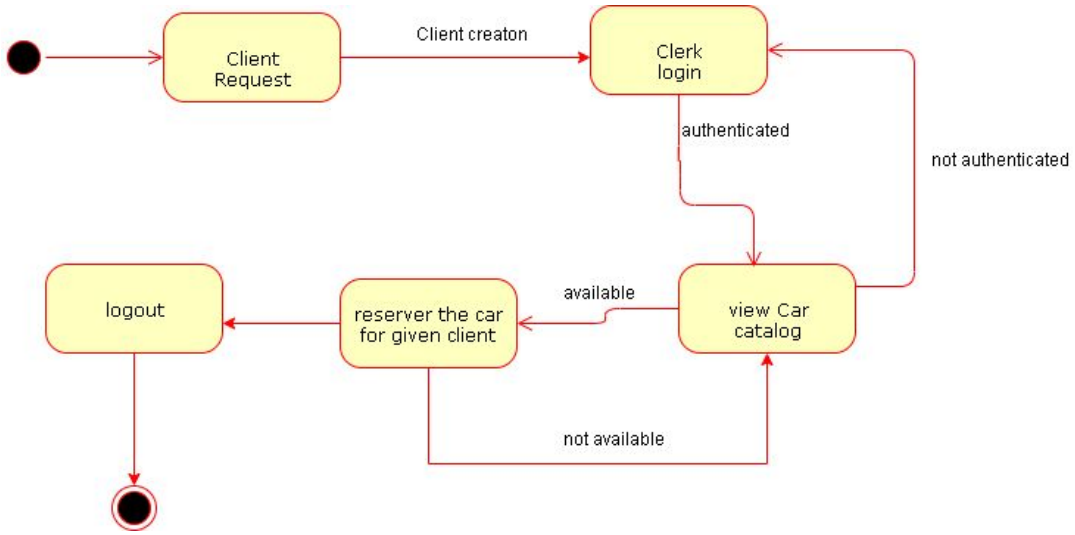

view car catalog
modify client records
reserve car
return car
rental car
Clerk
Car booking managemen system

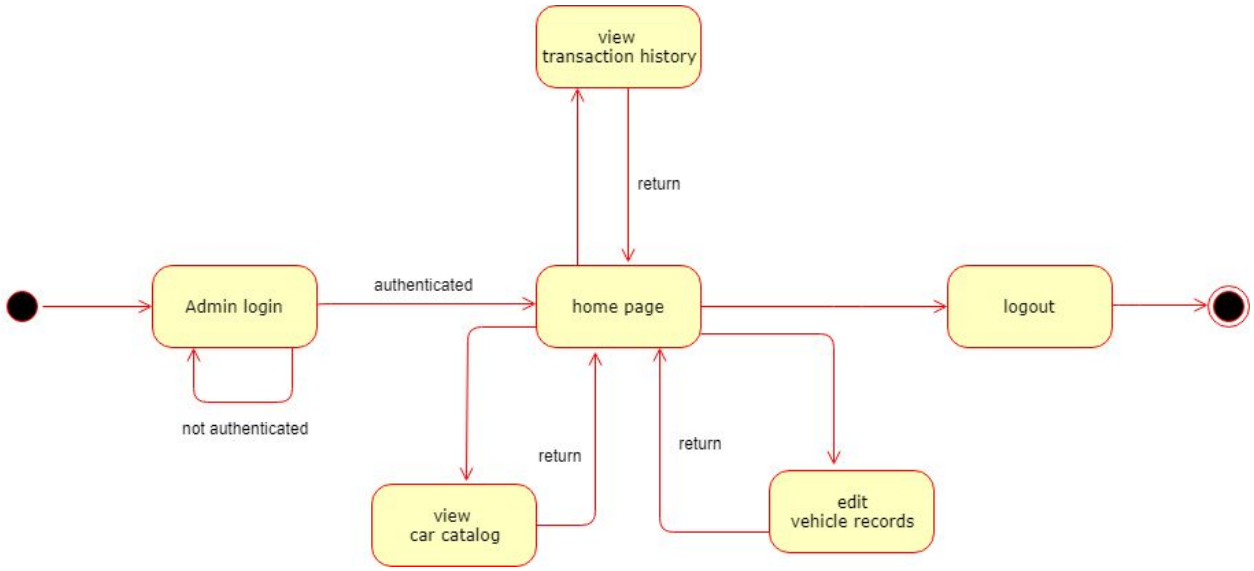

manage vehicle records
view car catalog
search car catalog
Admin
Car booking management system

5 . LOGICAL VIEW :

It focuses on functional requirements and the main building blocks and key abstractions decomposing of the system. The view basically is an abstraction of the system's functional requirements.It is typically used for object-oriented modeling from where the static and dynamic system structures emerge. The logical view specifies system decomposition into conceptual entities (such as objects), and connections between them (such as associations). This view helps to understand the interactions between entities in the problem space domain of the application and their potential variation.

Logical diagrams is shown as : (state diagram)





6. DEVELOPMENT VIEW :

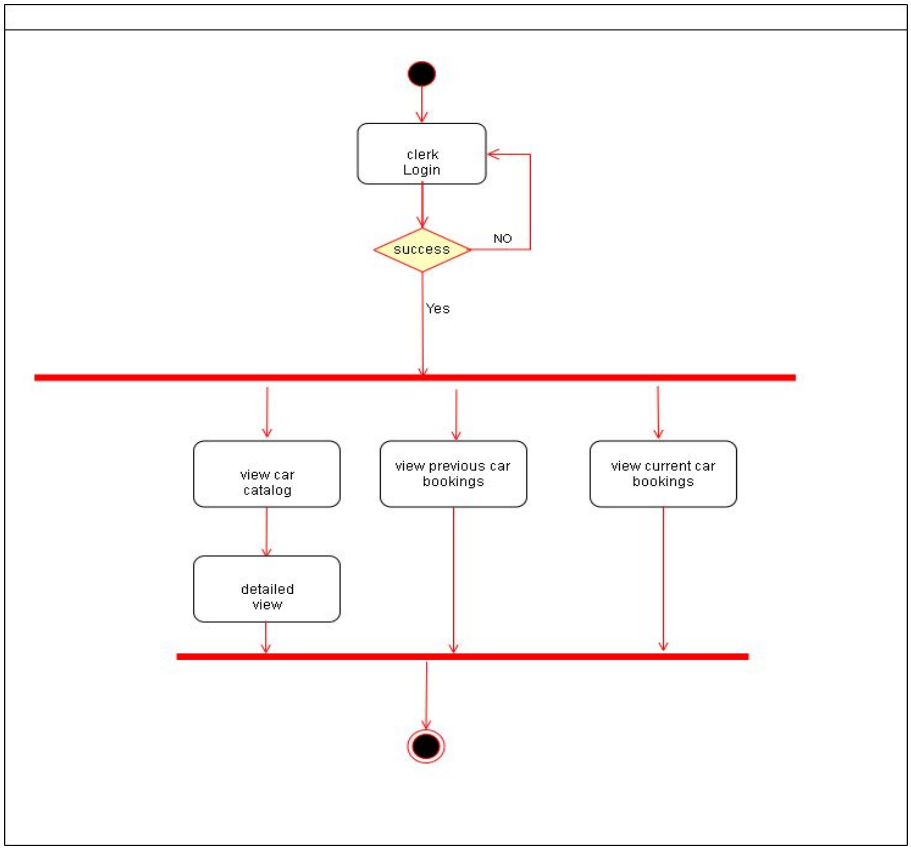
The development (or implementation) view describes the components used to assemble the system. Component diagram is used to capture the view and it is shown as

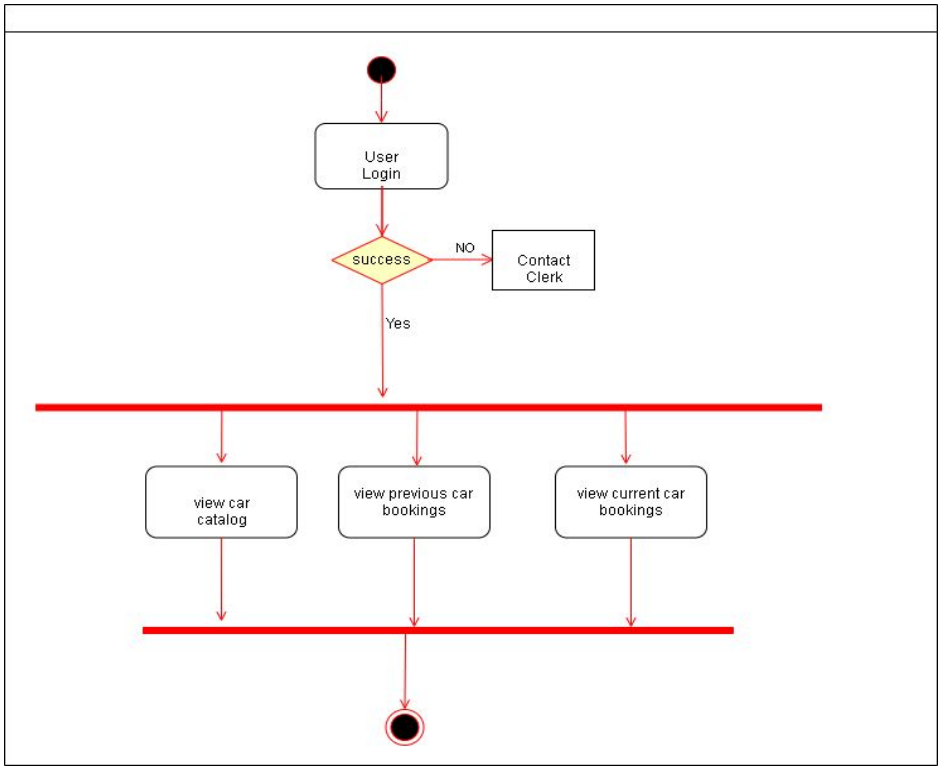
Reuse of components and frameworks :

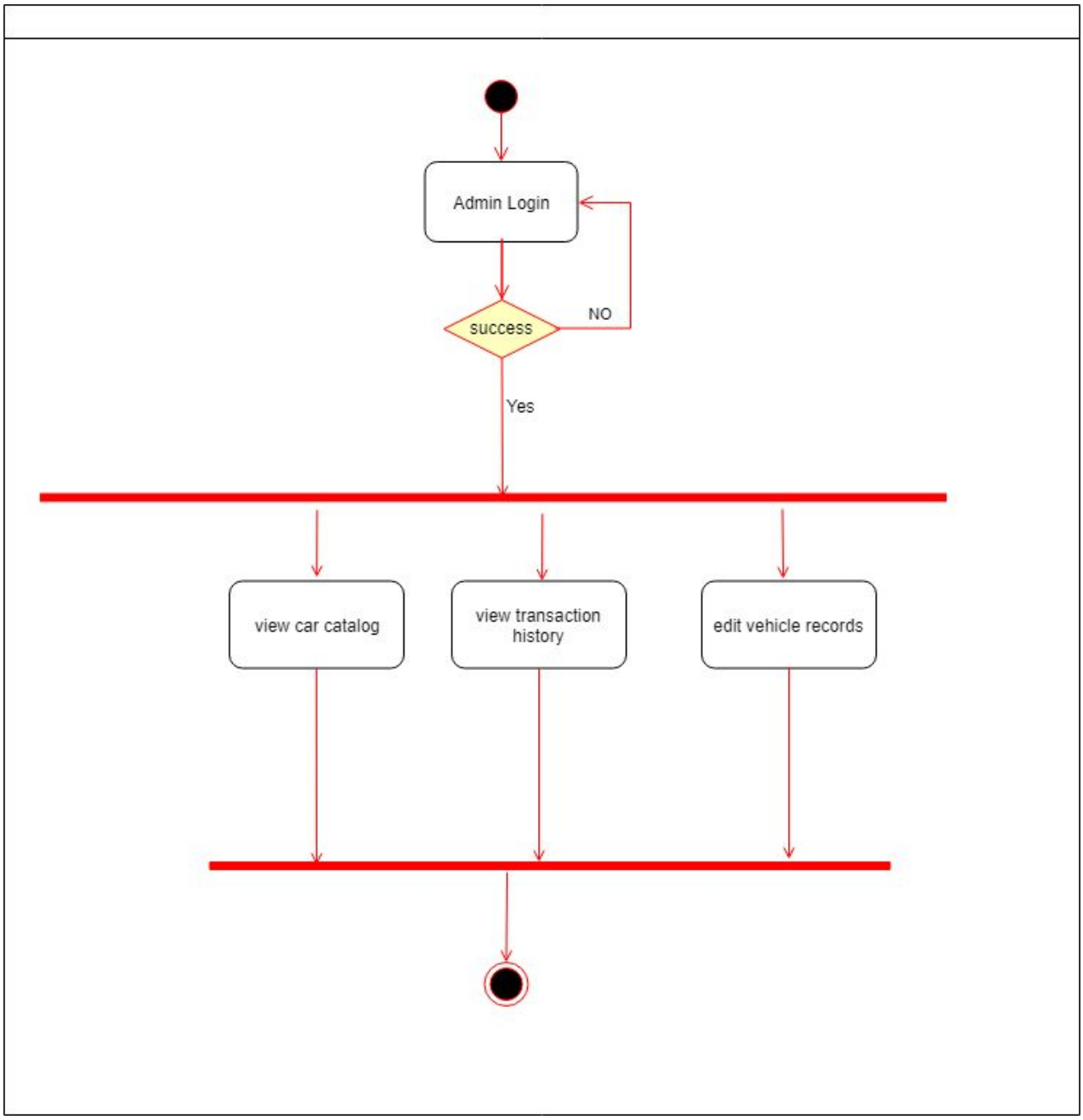
Component frameworks are sets of well-defined interfaces that establish the protocols for component cooperation within the framework. These protocols are used by component developers to ensure the interoperability of their components within the scope of a given framework.

7. PROCESS VIEW :

There is only one process view of the system, which illustrates the process decomposition of the system, including the mapping of classes and subsystems on to processes and threads The process view is refined during each iteration. The activity diagram for the CRBM is shown as :


clerk
Login
success
NO
Yes
view car
catalog
view previous car
bookings
view current car
bookings
detailed
view


User
Login
success
NO
Contact
Clerk
Yes
view car
catalog
view previous car
bookings
view current car
bookings



8. DEPLOYMENT VIEW :

Deployment diagram is a structure diagram which shows the architecture of the system as deployment (distribution) of software artifacts to deployment targets. The deployment (or physical) view illustrates the physical components of the architecture, their connectors and their topology. It shows the physical network and hardware configurations on which the software will be deployed.

NAME	TYPE	DESCRIPTION
Installation	Tomcat Server	Technical Specifications

9. QUALITY :

A quality attribute is a measurable or testable property of a system that is used to indicate how well the system satisfies the needs of its stakeholders.In other words, a quality attribute (aka as Non-Functional Requirements) is what makes a system good with respect to a specific stakeholder. An example of a QA is how fast the function must be performed or how resilient it must be to an erroneous input, the time to deploy the product, or a limitation on operational costs. For example some QA are performance , Security , modifiability , reliability , usability .

Example of quality attribute scenario :

Choosing a general scenario: **Availability**

Source: Internal/external: people, hardware, software, physical infrastructure or environment.

Stimulus: Fault: omission, crash, incorrect timing, incorrect response

Artifact(s): Processors, communications channels, persistent storage, processes

Environment: Normal operation, startup, shutdown, repair mode, degraded operation, overloaded operation

Response: Prevent the fault from becoming a failure.