Project name: Vehicle Renting System	**Version:** 1.0
Software Requirement Specifications	**Date:** 17 Sept, 2019

Software Requirements Specification

Version 1.0

for

SOEN6461 Coursework Project

Prepared by

Nirav Ashvinkumar Patel	40081268	niravjdn@gmail.com
Charles Jebalitherson Augustin Moses	40084105	charlesjebalitherson@gmail.com
Jemish Kishor Paghadar	40080723	jemish.paghadar27@gmail.com
Vikramjit Singh	40075774	vikramsandhu008@gmail.com
Avinash Damodaran	40078258	avinashdamu323@gmail.com

Project name: Vehicle Renting System	**Version:** 1.0
Software Requirement Specifications	**Date:** 17 Sept, 2019

Document history

Date	Version	Description	Author
09/18/2019	1.0	Initial Version	Nirav Patel
09/18/2019	1.0	Added Actor/Goal Table	Nirav Patel
09/22/2019	1.0	Added Introduction Section	Avinash Damodaran
09/23/2019	1.0	Added Non-Functional Requirements	Jemish Paghadar
09/23/2019	1.0	Added External Interface Figures and Description, Use Case Diagram and Domain Model	Nirav Patel
09/23/2019	1.0	Added Product perspective and functions	Charles Jebalitherson
09/24/2019	1.0	Added constraints, assumptions and dependencies	Charles Jebalitherson
09/24/2019	1.0	Added the Contract details for sequence diagram	Avinash Damodaran
09/25/2019	1.0	Added sequence diagrams for 6 contracts	Avinash Damodaran
09/25/2019	1.0	Added State Diagram	Jemish Paghadar
09/25/2019	1.0	Added sequence diagram for 3 contracts	Vikramjit Singh

09/05/2019	1.0	Updated Format - Refactoring	Nirav Patel
10/20/2019	2.0	Added Use case diagram, description of use case, and actor goal list	Nirav Patel
10/21/2019	2.0	Added Client View Specification by generating screenshots from bootsnip UI Generator	Nirav Patel
10/24/2019	2.0	Sequence Diagram and Contract Creation for manage vehicle records by admin	Avinash Damodaran
10/24/2019	2.0	Added Sequence Diagram and Contract details for vehicle reservation	Vikramjit Singh
10/24/2019	2.0	Updated Domain Model	Jemish Paghadar
10/25/2019	2.0	Added Contract for ReservationHistory View	Charles Jebalitherson
10/25/2019	2.0	Updated sequence diagram for ReservationHistory view	Charles Jebalitherson
10/25/2019	2.0	Added Images for Admin View for all out vehicles and checking vehicle availability	Nirav Patel

10/27/2019	2.0	Added Sequence Diagram and contract for checking vehicle availability	Nirav Patel
10/27/2019	2.0	Formatted document for final submission	Nirav Patel

List of figures

1. Introduction

SRS document provides a detailed overview of the entire Vehicle Rental System's high level functional and nonfunctional requirements.Intended Users of Vehicle Rental System are highlighted. The Vehicle Rental System is a web based application where the vehicle and clients records are stored in the database that provides a platform for a company to rent out its vehicle.

Purpose

Purpose of this document is to provide detailed description of the Vehicle Rental System. In order to provide better user experience, requirement elicitation is documented.
Necessary details of Vehicle Rental System in terms of functional and nonfunctional requirements are outlined. Details are portrayed using UML diagrams. System constraints, interface and interactions with other external applications are documented. Intended audience for the document is Developers, Designers, Business analysts and Maintainers who are known for improving better user experience. This document will serve them if requirements evolve and accommodation of more features are required.

Scope

The Vehicle Rental System provides a platform for the company to rent their vehicles to clients. The clerks can make vehicle reservation or cancel using this system for any particular clients. In addition clerk can create, modify and delete the records of clients. While the administrators using this system can manage the transactions history per client and can add vehicle records to the database.

Definitions, acronyms, and abbreviations

Terms	Definition
SRS	Software Requirement Specifications
UML	Unified Modeling Language

References

- Lecture Slides and Material provided by Dr. Constantine Constantinos.
- IEEE Software Engineering Standards Committee, "IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications", October 20, 1998.

2. Overall Description

Vehicle renting system is a high level web application designed to manage the vehicle and client information. The renting system performs operations such as view, search, updation and deletion of records in a catalog. It is a user friendly system structured to meet the needs of individual users of the system.

2.1. Product perspective

The main features of renting system includes viewing & searching catalog, renting, reserving and returning vehicles and administering the database. In addition to the aforementioned features, our application supports creation and cancellation of reservation, handles the return of vehicle. It also supports data filtering and sorting based on certain fields in the view. Users can drill down a transaction to view the detailed information (Eg: Vehicle availability, Due date for return).

2.2. Product functions

The vehicle module records information such as type, make, model, color and license plate number respectively. The client module stores information about client's first name, last name, driver's license and phone number. There are only two authorized users of the system namely clerk and administrator.

Clerk can view & search catalogs. In addition, the clerk can filter and sort the result set based on some criteria. He/ She can also create or cancel a reservation for a client, handle return of the vehicle. Each reservation creates a transaction along with time stamp. Administrator can query the history of transactions based on client, vehicle and due date.

2.3. User Characteristics

The authorized users of the system are clerks and administrators. A user must have a minimum level of higher education, basic knowledge of computers and experience in using web application. Additionally, the users must possess the domain knowledge of application and its limitations.

2.4. Constraints

- A minimum of 512 MB RAM, CPU of 1 Ghz, 128 MB of disk space and a web browser is required to view the application.
- Proper user authentication must be implemented to allow only authorized users to access the system.
- SSL certificate must be procured and configured in server from certificate authority to ensure a secure communication between client and server.

2.5. Assumptions and dependencies

- Web application must be connected to the internet to function properly.
- It is expected that, the web application is connected to the database for uninterrupted services.
- The server hosting the application must be up and running at all times.
- Users are assumed to have a basic knowledge on vehicles and client related documents (Eg. Validating the client's information such as driver's license authenticity).

3. Specific Requirements

This section contains all requirements in detail: Functional as well as non-functional requirements (quality attributes and constraints). The quality attributes are listed according to the ISO/IEC 25010 standard that classifies software quality in a structured set of characteristics and sub-characteristics.

3.1. External Interface

- Login Interface

Figure 1: Login Page Interface

- Result View/Interface For Client Record

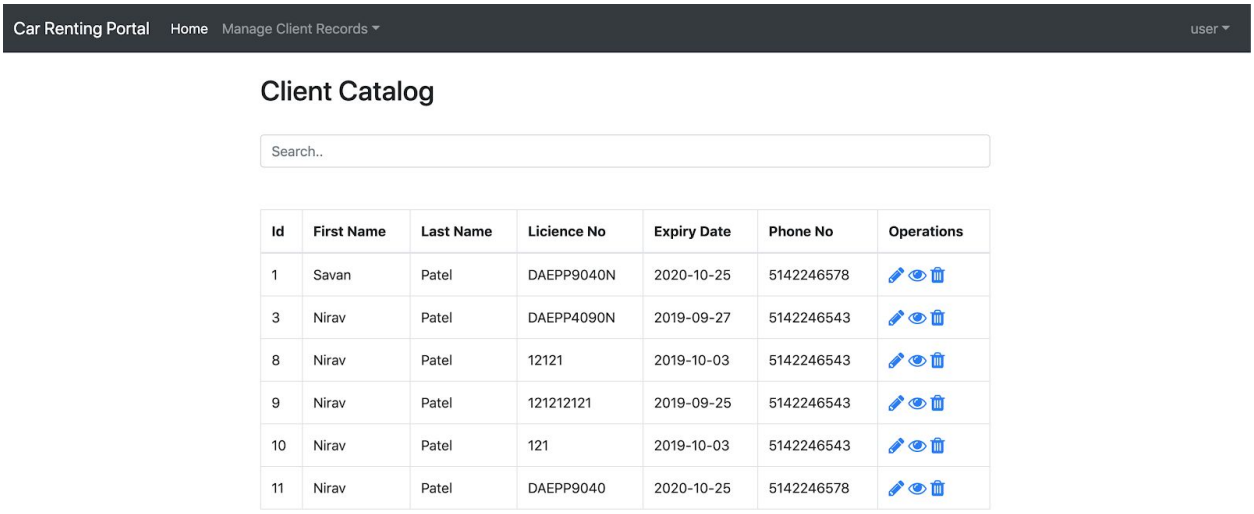


Figure 2: Client Catalog

- Add Client Record Form

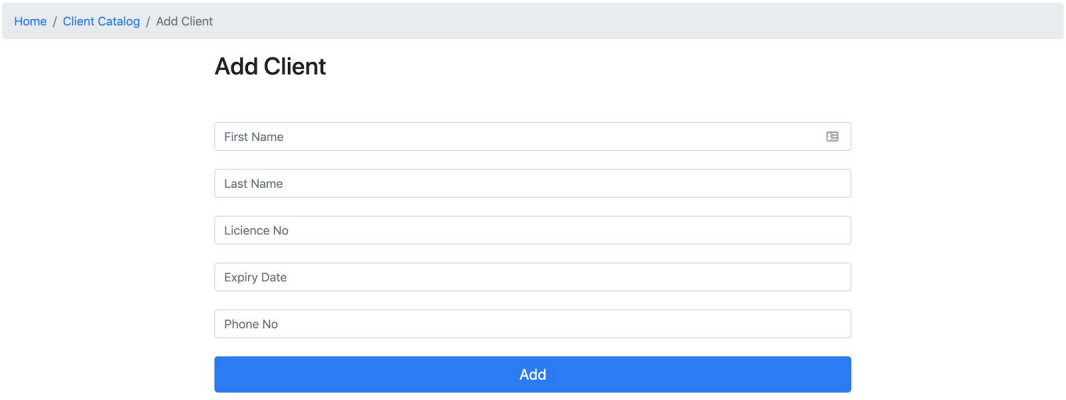


Figure 3: Add Client Record Form

- Update Client Record Form

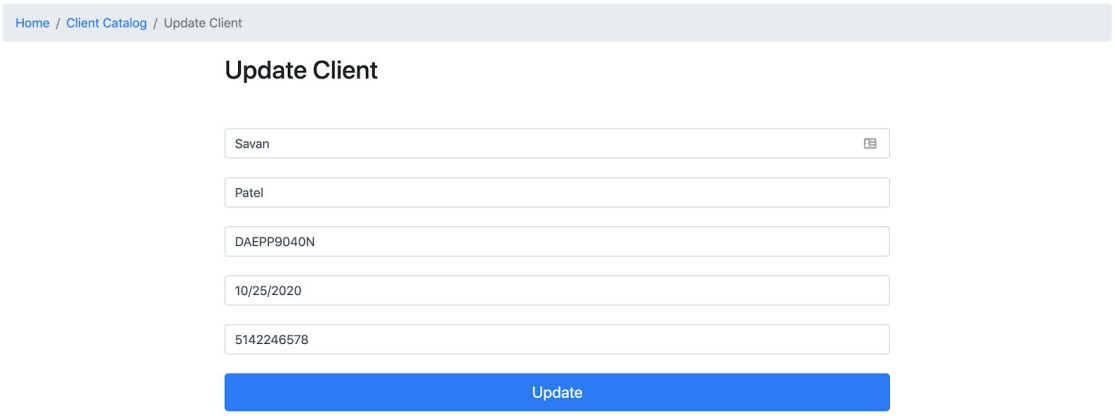


Figure 4: Update Client Record Form

- Client Record Detail View

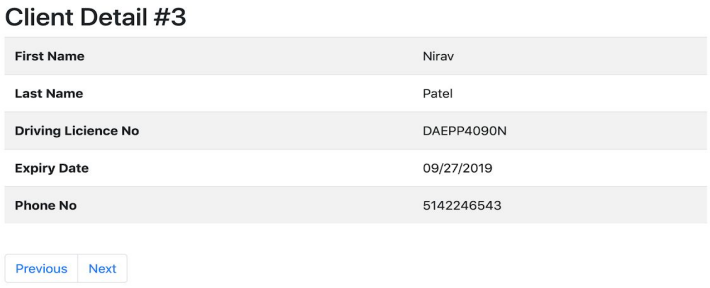


Figure 5: Client Record Detail View

- Vehicle Record Detail View

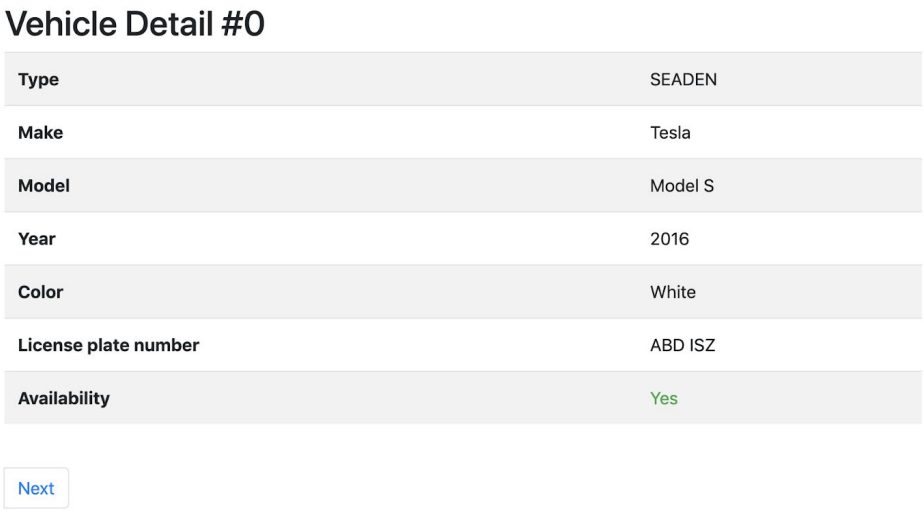


Figure 6: Vehicle Record Details View

- View Vehicle Record and Search based on different filter criteria

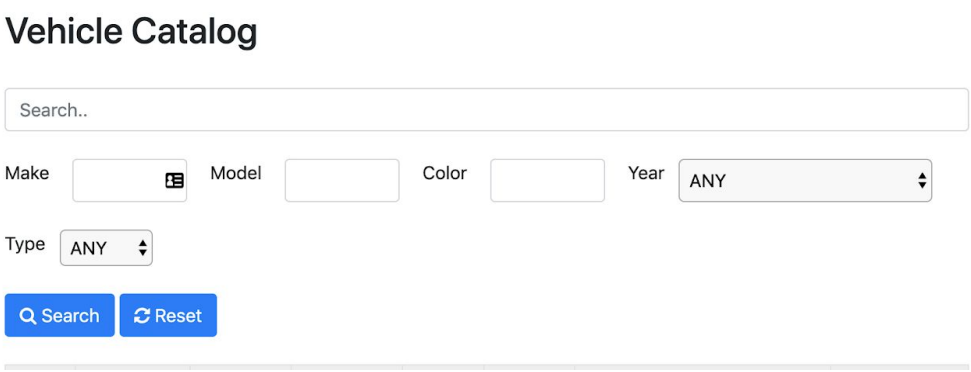


Figure 7: Vehicle Catalog Search View

- Reservation Catalog

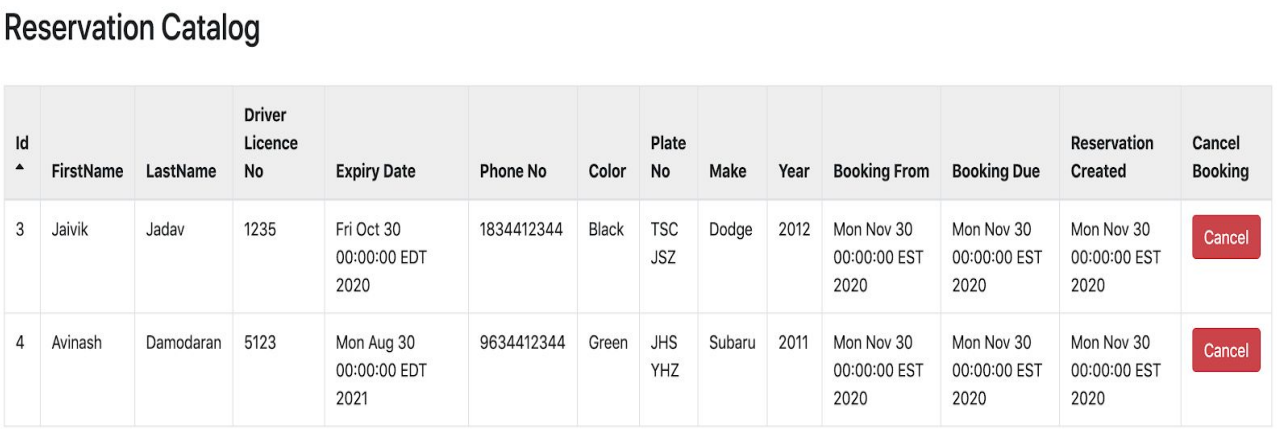


Figure 8: Reservation Catalog View

- Reservation Form

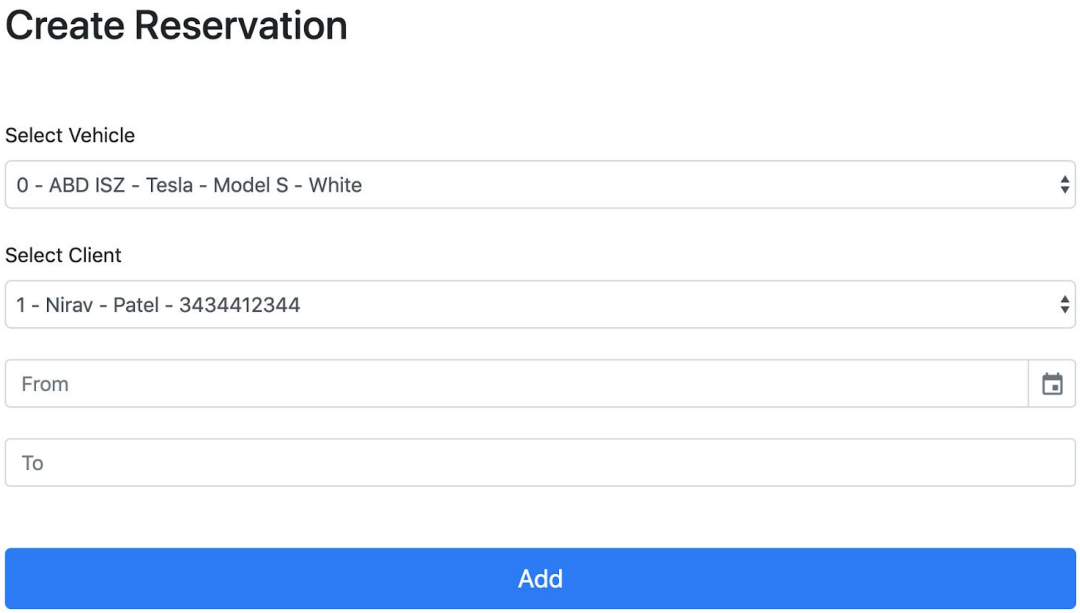
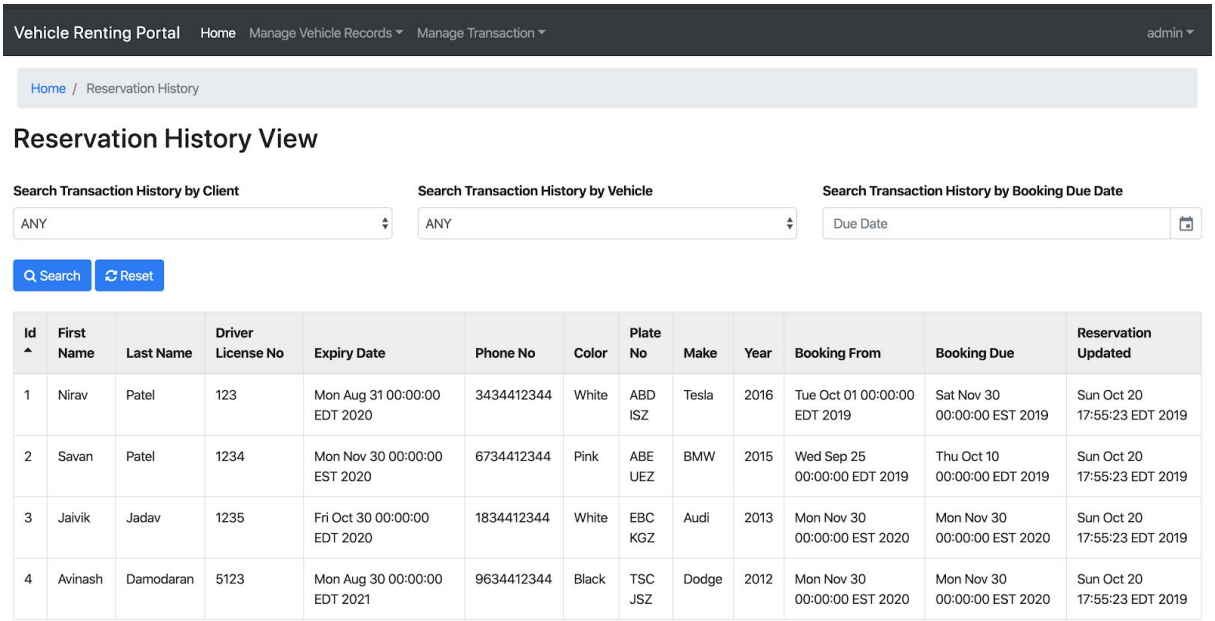


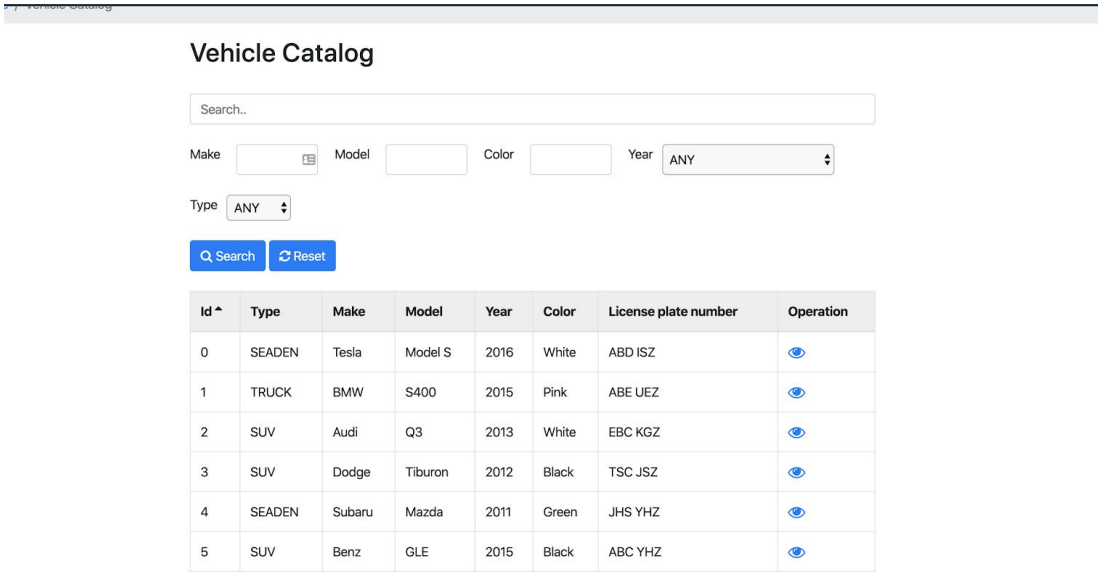
Figure 9: Create Reservation Form

- Reservation History View For Admin

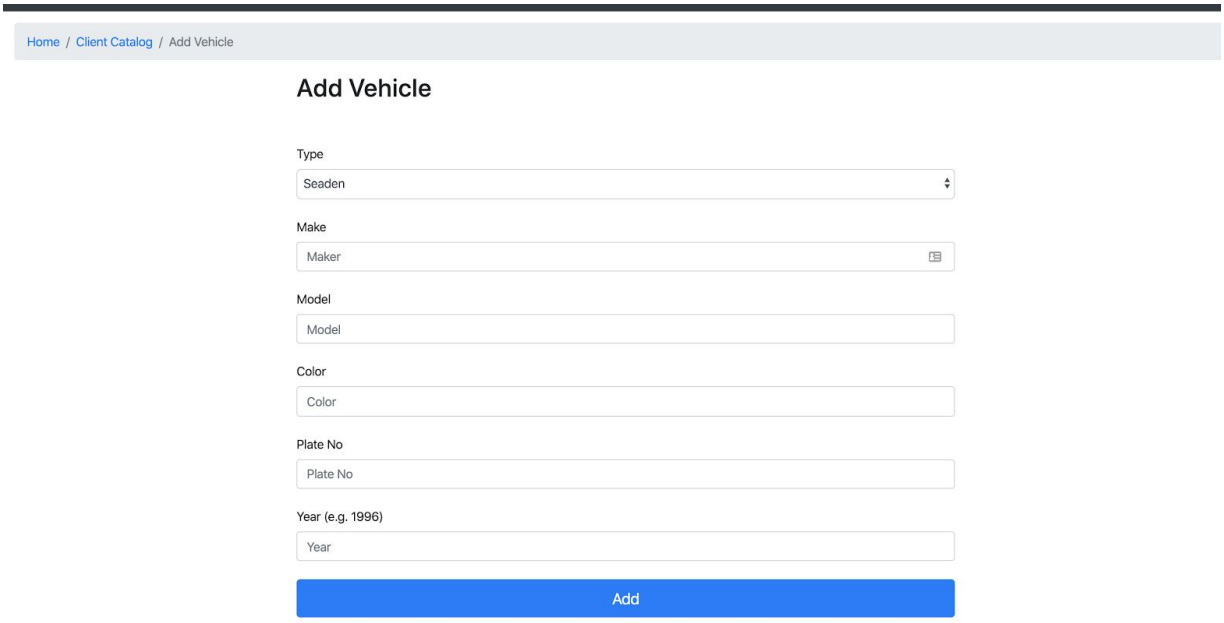


Id	First Name	Last Name	Driver License No	Expiry Date	Phone No	Color	Plate No	Make	Year	Booking From	Booking Due	Reservation Updated
1	Nirav	Patel	123	Mon Aug 31 00:00:00 EDT 2020	3434412344	White	ABD ISZ	Tesla	2016	Tue Oct 01 00:00:00 EDT 2019	Sat Nov 30 00:00:00 EST 2019	Sun Oct 20 17:55:23 EDT 2019
2	Savan	Patel	1234	Mon Nov 30 00:00:00 EST 2020	6734412344	Pink	ABE UEZ	BMW	2015	Wed Sep 25 00:00:00 EDT 2019	Thu Oct 10 00:00:00 EDT 2019	Sun Oct 20 17:55:23 EDT 2019
3	Jaivik	Jadav	1235	Fri Oct 30 00:00:00 EDT 2020	1834412344	White	EBC KGZ	Audi	2013	Mon Nov 30 00:00:00 EST 2020	Mon Nov 30 00:00:00 EST 2020	Sun Oct 20 17:55:23 EDT 2019
4	Avinash	Damodaran	5123	Mon Aug 30 00:00:00 EDT 2021	9634412344	Black	TSC JSZ	Dodge	2012	Mon Nov 30 00:00:00 EST 2020	Mon Nov 30 00:00:00 EST 2020	Sun Oct 20 17:55:23 EDT 2019

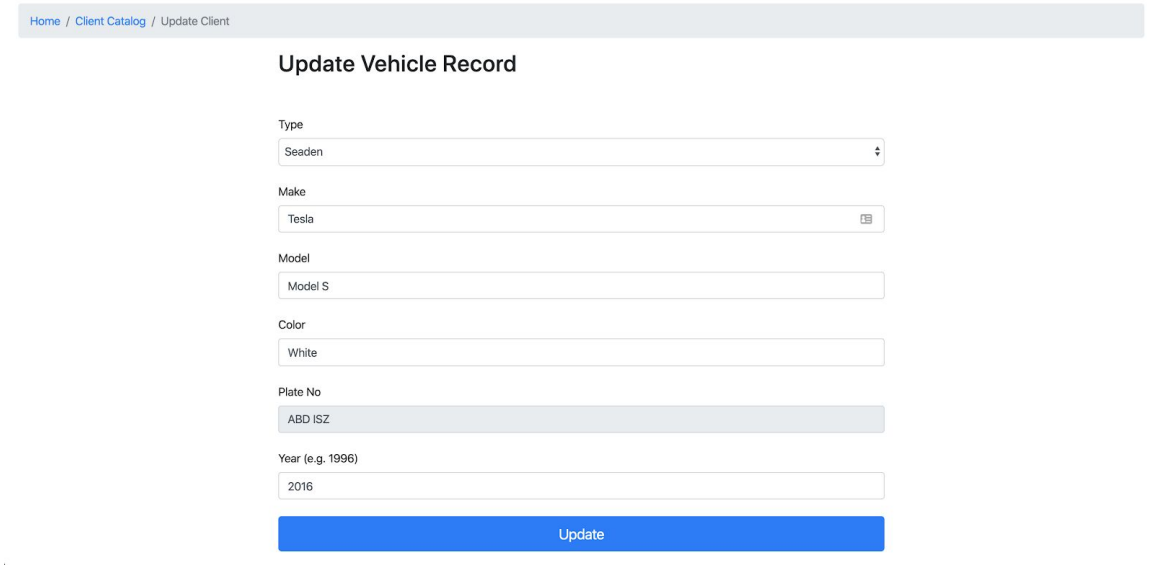
- Vehicle Catalog for Clerk



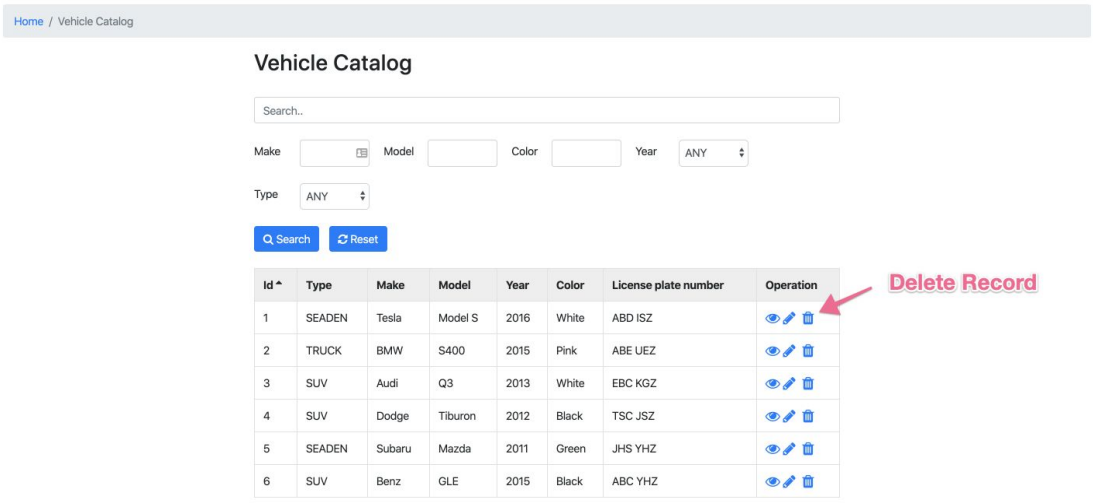
- Create Vehicle Record Form



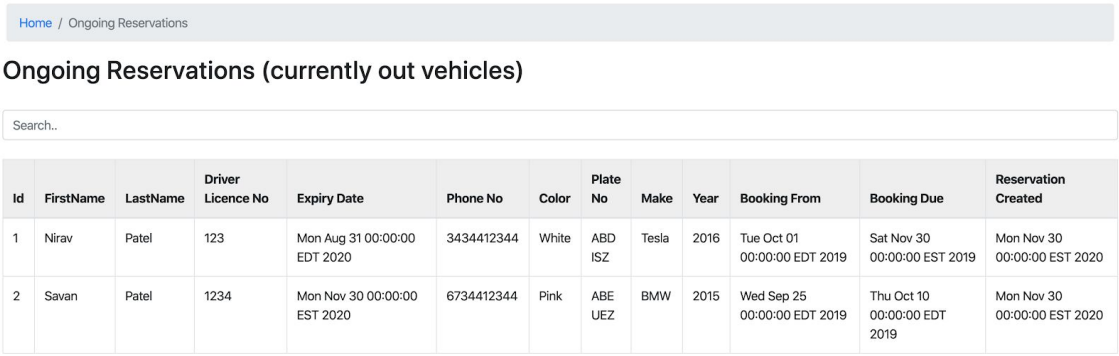
- Update Vehicle Record Form



- View Vehicle Record For Admin



- Admin View For All out Vehicle



- Admin View For Checking Vehicle Availability

3.2. Functional Requirements

Functional requirements capture the intended behavior of the system. This section contains the Actor Goal List and the Use Case view.

- Actor Goal List

Actor	Goal
Clerk	View Vehicle Catalog
	Search Vehicle Catalog
	Manage(create, update, delete, return) reservation

	View Vehicle Details
	Sort View Order of Catalog
	Manage(create, update, delete) client record
	View Due Vehicle By Date
System	Persist Database
Admin	View History of Reservations per client, per vehicle, per due date
	Manage Vehicle Record
	View Vehicle Catalog
	View Reservations by due date
	View Vehicle Availability by given date range
	View All out vehicles

- Use case view

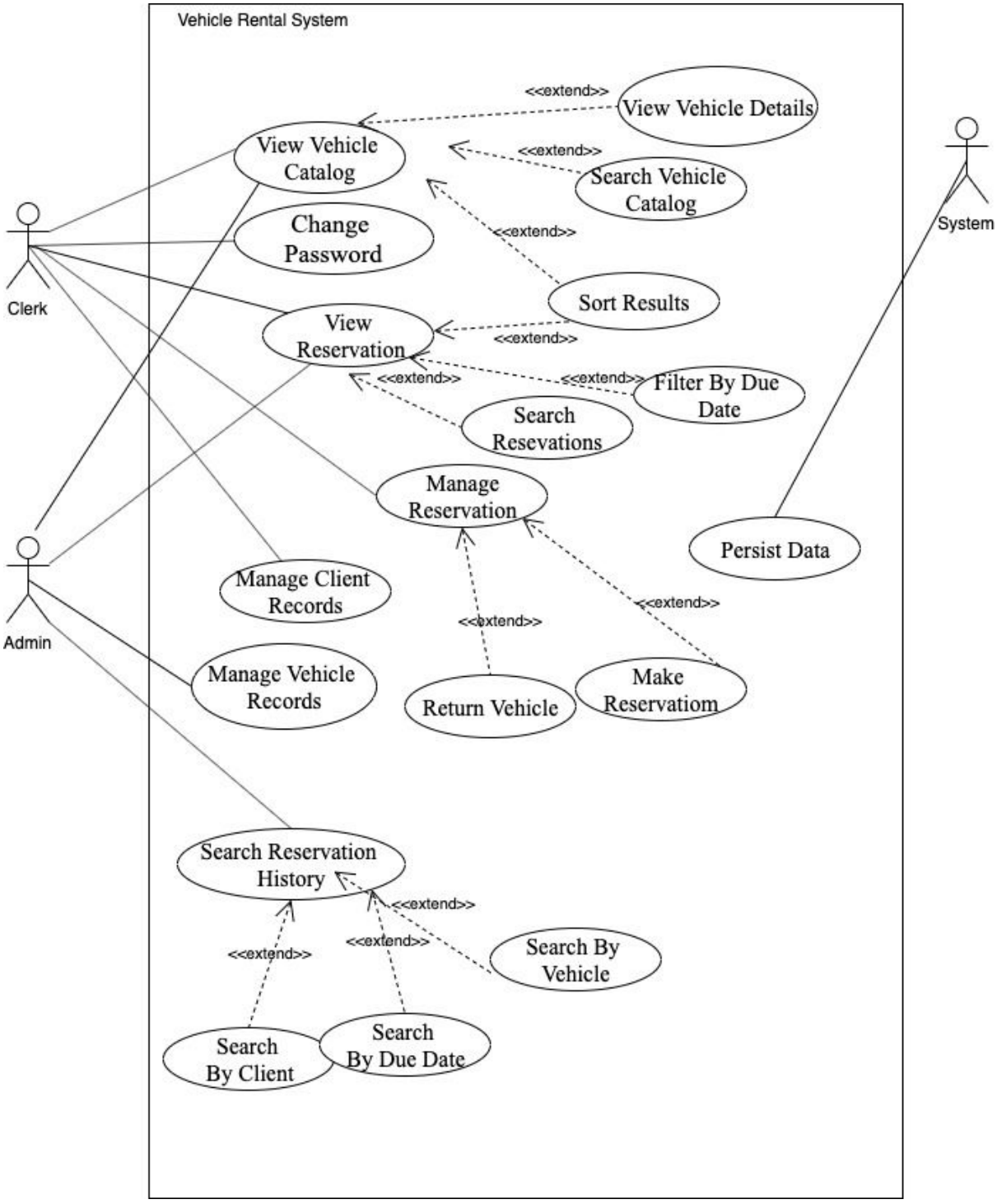


Figure 10: Use Case Model

Description of Individual use case is described below:

Use case	Description
Login	The clerk shall be able to log in using the correct username and password
View Vehicle Catalog	The clerk and admin shall be able to view Vehicle Catalog in an interactive manner, from where the clerk can perform certain operations and filter them.
Search Vehicle Catalog	The clerk and admin shall be able to filter Vehicle catalog based on several filter criteria such as Car less than 3-year-old, Car with black colors, Car with particular plate no.
Manage(create, update, delete, return) reservation	The clerk shall be able to manage reservations.
View Vehicle Details	The clerk shall be able to view vehicle details and also its availability in this view.
Sort View Order of Catalog	The clerk shall be able to sort order of catalog based on its attributes.
Manage(create, update, delete) client record	The clerk shall be able to manage client records
View Due Vehicle By Date	The clerk shall be able to see the due date of the vehicle that is rented out or has been reserved.
Search Reservation History	The admin shall be able to view vehicle reservations history by vehicle, client, and its due date.
View Reservation	The clerk and admin should be able to view reservations.
Manage Vehicle Records	The admin shall be able to manage(create, update, delete) vehicle records.

3.3. Non-functional requirements

Performance Efficiency

It shows the response of the system to perform certain actions for a certain period of time and represents the performance relative to the amount of resources used under stated conditions. It can be characterized in mainly three categories such as time behavior, resource utilization and capacity.
Time behaviour represents the degree to which the response and processing times and throughput rates of a product or system, when performing its functions, meet requirements. Resource utilization represents the degree to which the amounts and types of resources used by a product or system, when performing its functions, meet requirements and capacity represents the degree to which the maximum limits of a product or system parameters meet requirements.
Our vehicle renting system shall respond in a few seconds to the clerk and users. This web application must be connected to the internet to function properly.

Compatibility

The degree to which a product, system or component can exchange information with other products, systems or components, and/or perform its required functions, while sharing the same hardware or software environment.
Our System shall provide co-existence as it will perform its required functions efficiently while sharing a common environment and resources with other systems, without detrimental impact on any other system and it will also be interoperable so that two or more systems can exchange and use information such as vehicle catalog information etc.

Usability

Usability is the degree to which a software can be used by specified consumers to achieve quantified objectives with effectiveness, efficiency, and satisfaction in a quantified context of use. It is one of the most important attributes, because, unlike in cases with other attributes, users can see directly how well this attribute of the system is worked out.
This system shall be easy to use for input preparation, operation, and interpretation of output and shall provide consistent user interface standards or conventions with our other frequently used systems. It will also be easy for new or infrequent users to learn to use the system as It will provide learnability, operability, user error protection, accessibility and user interface aesthetics. Our system shall increase user confidence and satisfaction by providing user friendly navigation.

Reliability

Reliability is an attribute of the system responsible for the ability to continue to operate under predefined conditions. Our system shall work under different working environments and different conditions. It shall also define maturity by satisfying the needs for reliability under normal operation.
Vehicle renting system shall be available, accessible when required for use and operational as it is indented to operate despite the presence of hardware or software faults. Moreover, in the event of an interruption or failure, this system will recover the data directly affected and re-establish the desired state of the system.

Security

It is a measure of the system's ability to resist unauthorized usage while still providing its services to legitimate users. It is also one of the important quality attributes as it is responsible for the ability of the system to reduce the likelihood of malicious or accidental actions as well as the possibility of theft or loss of information.
Our system shall provide confidentiality by making data accessible only to those authorized who have access through password privacy and prevent unauthorized access and modification of data. Thus, security can be characterized as a system providing nonrepudiation, confidentiality, integrity, assurance, availability, and auditing.

Maintainability

It is the ease with which a product can be maintained in order to correct defects or their cause, repair or replace faulty or worn-out components without having to replace still working parts, prevent unexpected working conditions, maximize a product's useful life, maximize efficiency, reliability, and safety, meet new requirements, make future maintenance easier, or cope with a changed environment.

This system shall be maintainable because it is composed of discrete components such that a change to one component has minimal impact on other components. It shall also be reusable as module that is developed with high cohesion and low coupling can be used in more than one system, or building other assets. It will provide the effectiveness and efficiency with which it is possible to assess the impact on a product or system of an intended change to one or more of its parts, or to diagnose a product for deficiencies or causes of failures, or to identify parts to be modified. Moreover, it will also be effectively and efficiently modified and tested without introducing defects or degrading existing product quality.

Portability

The degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another. It involves transferring installed program files to another computer of basically the same architecture. Our system shall provide abstraction by which a system can effectively and efficiently be adapted for different or evolving hardware, software or other operational or usage environments.

Design Constraints

This section is used to specify constraints on the system design imposed by external standards, regulatory requirements, or project limitations. Examples of design constraints include the use of a particular programming language or framework (or versions thereof), a specific operating system, or references to a standard reference architecture.

- A minimum of 512 MB RAM, CPU of 1 Ghz, 128 MB of disk space and a web browser is required to view the application.
- SSL certificate must be procured and configured in server from certificate authority to ensure a secure communication between client and server.
- Web application must be connected to the internet to function properly.
- It is expected that, the web application is connected to the database for uninterrupted services.
- The server hosting the application must be up and running at all times.
- Users are assumed to have a basic knowledge on vehicles and client related documents (Eg. Validating the client's information such as driver's license authenticity).

(On-line) user documentation and help

- **Version Control System**
  To have a common repository for all project files available and updated remotely, distributed version control system(Git) is maintained.
  Clone the repository from below link
  https://github.com/niravjdn/SDM-Project.git

- **Google Docs**
  Google Docs brings your documents to life with smart editing and styling tools to help you easily format text and paragraphs.
  Link below is to access software requirement specification(SRS) document
  https://docs.google.com/document/d/1Nf2htqwNipwpwBkuJHgIIEKss1c2Wyc_30OqOcDhJsU/edit?usp=sharing

Link below is to access software architecture document(SAD)
https://docs.google.com/document/d/1qzFCg2DJoRVz6zVaxPKkzUvPD7Dt12LYFW3MdrxW1-k/edit?usp=sharing

Purchased components

This vehicle renting project is open source, and hence no component of it will be purchased.

Licensing Requirements

This system will be released under a GPL license and will be open-source.

Legal, Copyright and Other Notices

Our system is not a copyright trademark.

4. Analysis Model

This section contains a list of all analysis models used in developing specific requirements. Each model has an introduction and a narrative description.

4.1. Sequence Diagram

Contract CO1: viewVehicleCatalog
Operation: viewVehicleCatalog()
Cross References: Use Case: View Vehicle Catalog.
Preconditions: Clerk is authenticated
Postconditions:

1. Catalog was associated with View (association formed)

Contract CO2: sortVehicleCatalog
Operation: sortVehicleCatalog()
Cross References: Use Case: Sort Vehicle Catalog.
Preconditions: Clerk is authenticated
Postconditions:

1. Instance of Sort Catalog **sr** is created (instance creation).
2. **sr** was associated with Catalog (association formed).
3. **sr.vehicles** is set to a list of elements from Catalog with matching provided specifications. (attribute modifications).
4. **sr** was associated with View (association formed).

Contract CO3: viewVehicleDetail
Operation: viewVehicleDetail()
Cross References: Use Case: View Vehicle Detail.
Preconditions: Clerk login session is present
Postconditions:

1. An attribute of vehicle **dt** is created based on the vehicle selected on the vehicle listing page(association formed)

Contract CO4: addClientRecord
Operation: addClientRecord ()
Cross References: Use Case: Adding Client Record.
Preconditions: Clerk login session is present
Postconditions:

1. An instance of client **c** was created (instance creation)
2. **c** was associated with clientCatalog (association formed)
3. **c** attributes defined with keyworded arguments from spec list. (attribute modification)

Contract CO5: modifyClientRecord
Operation: modifyClientRecord (spec)
Cross References: Use Case: Adding Client Record.
Preconditions: Clerk login session is present
Postconditions:

1. An instance of client **c** was retrieved from clientCatalog based on specifications provided (association formed)
2. **c** attributes updated with appropriate arguments from spec list. (attribute modification)

Contract CO6: deleteClientRecord
Operation: deleteClientRecord (id)
Cross References: Use Case: Delete Client Record.
Preconditions: Clerk login session is present
Postconditions:

1. An instance of client **c** was removed from clientCatalog based on specifications provided (association broken)

Contract CO7: createReservation
Operation: createReservation ()
Cross References: Use Case: Create Client Reservation
Preconditions: Clerk login session is present
Postconditions:

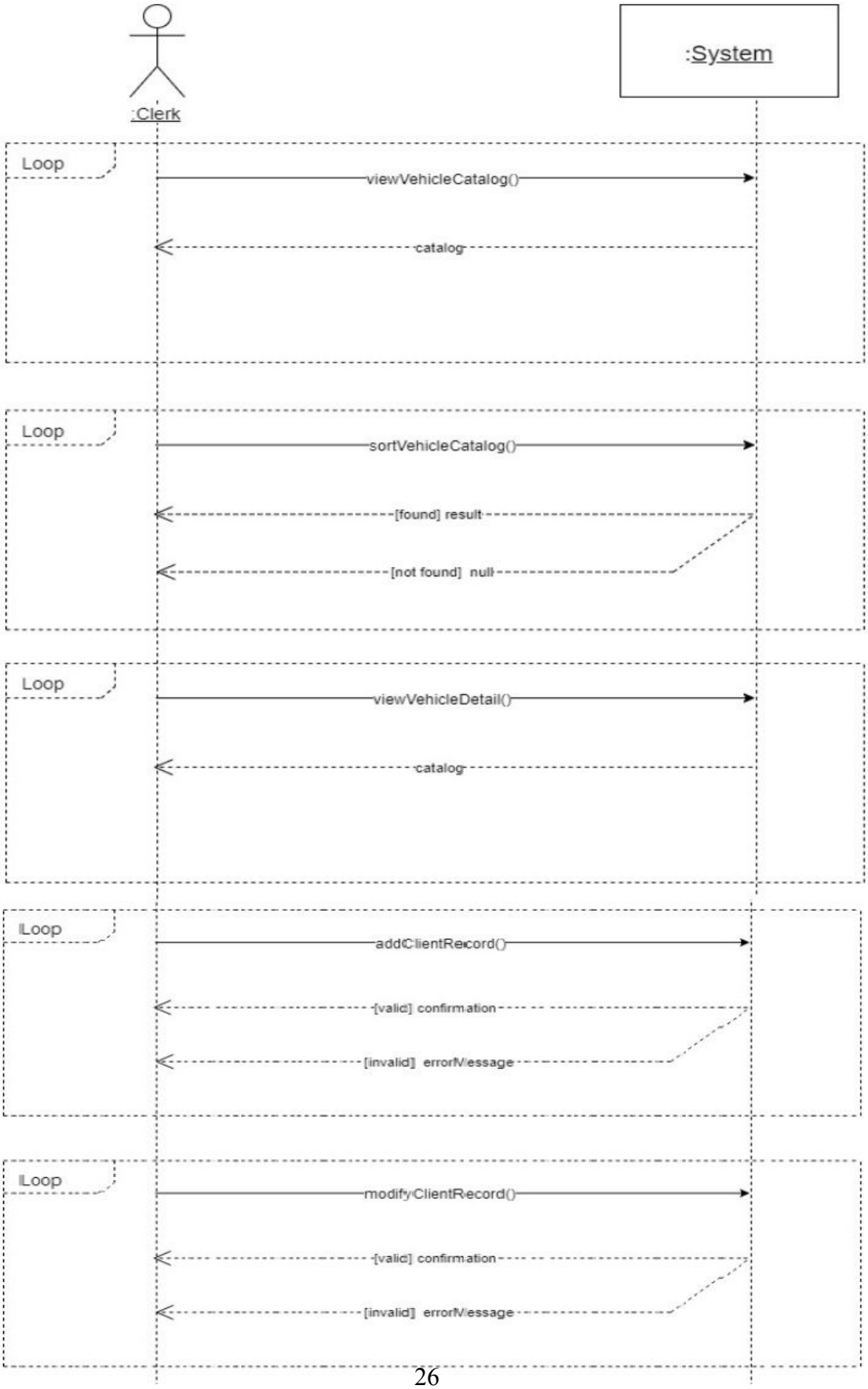
1. An instance of vehicle **vr** was retrieved from vehicleCatalog based specification provided. (association formed)
2. **vr.rented** was set to true (attribute modification)

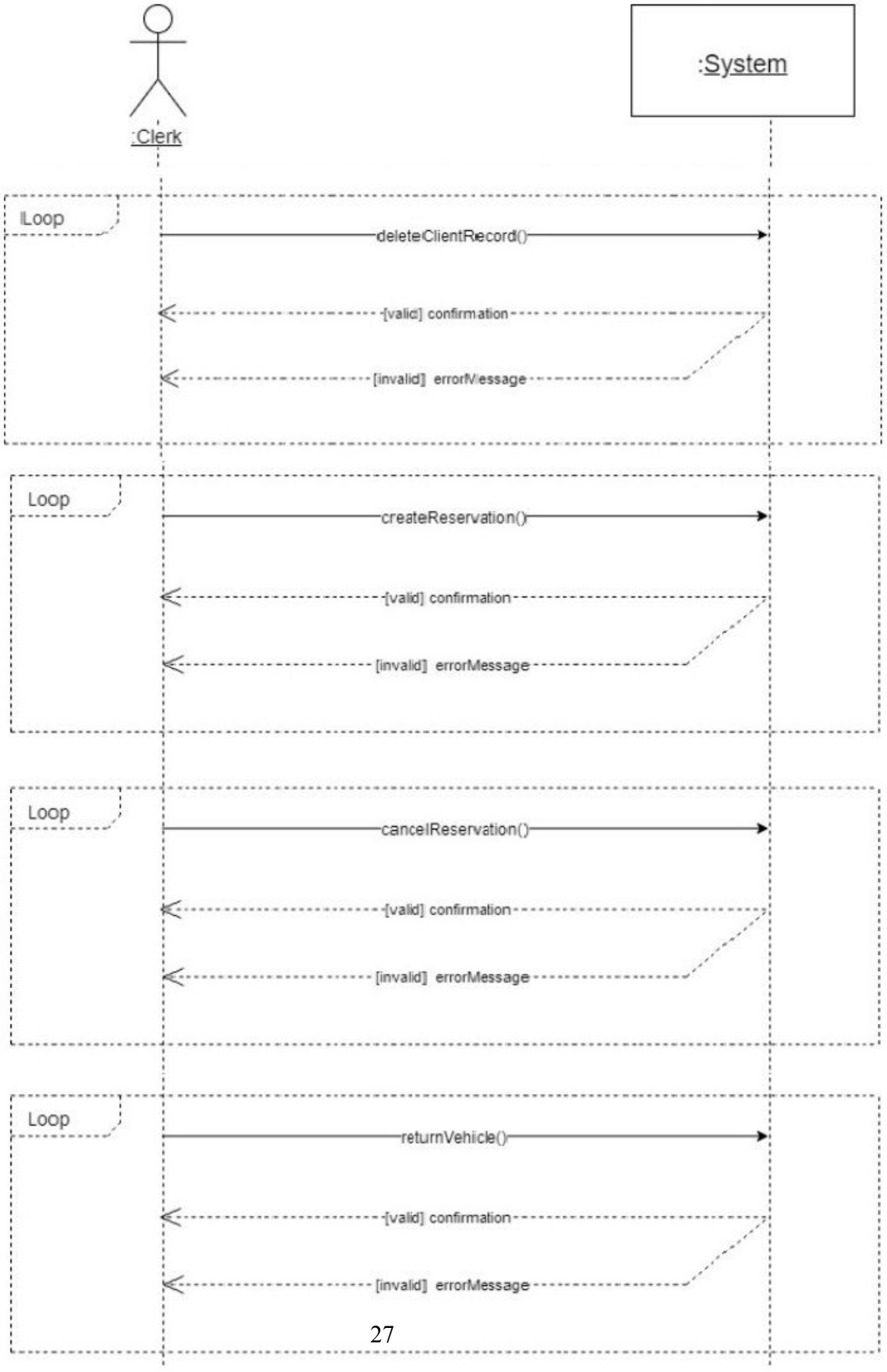
Contract CO8: cancelReservation
Operation: cancelReservation (spec)
Cross References: Use Case: Cancel Client Reservation.
Preconditions: Clerk login session is present
Postconditions:

1. An instance of vehicle **cr** was retrieved from vehicleCatalog based specification provided. (association formed)
2. **cr.rented** was set to false (attribute modification)

Contract CO9: returnVehicle
Operation: returnVehicle (id)
Cross References: Use Case: Return Vehicle.
Preconditions: Clerk login session is present
Postconditions:

1. An instance of vehicle **rv** was retrieved from vehicleCatalog based specification provided. (association formed)
1. **rv.rented** was set to false (attribute modification)


:Clerk
:System
Loop
viewVehicleCatalog()
catalog
Loop
sortVehicleCatalog()
[found] result
[not found] null
Loop
viewVehicleDetail()
catalog
Loop
addClientRecord()
[valid] confirmation
[invalid] errorMessage
Loop
modifyClientRecord()
[valid] confirmation
[invalid] errorMessage


:Clerk
:System
Loop
deleteClientRecord()
[valid] confirmation
[invalid] errorMessage
Loop
createReservation()
[valid] confirmation
[invalid] errorMessage
Loop
cancelReservation()
[valid] confirmation
[invalid] errorMessage
Loop
returnVehicle()
[valid] confirmation
[invalid] errorMessage

Contract CO10: createVehicleRec
Operation: createVehicleRec ()
Cross References: Use Case: Create Vehicle Record
Preconditions: Administrator login session is present
Postconditions:

1. An instance of vehicle **v** was created (instance creation)
2. **v** was associated with vehicleCatalog (association formed)
3. **v** attributes defined with appropriate arguments from spec list (attribute modification)

Contract CO11: modifyVehicleRec
Operation: modifyVehicleRec (spec)
Cross References: Use Case: Modify Vehicle Record.
Preconditions: Administrator login session is present
Postconditions:

1. An instance of vehicle **v** was retrieved from vehicleCatalog based on specifications provided (association formed)
2. **v** attributes updated with appropriate arguments from spec list (attribute modification)

Contract CO12: deleteVehicleRec
Operation: deleteVehicleRec (id)
Cross References: Use Case: Delete Vehicle Record.
Preconditions: Administrator login session is present
Postconditions:

1. An attribute of vehicle **v** was removed from vehicleCatalog based on specifications provided (association broken)

Contract CO13: viewVehicleCatalog
Operation: viewVehicleCatalog ()
Cross References: Use Case: View Vehicle Catalog.
Preconditions: Administrator login session is present
Postconditions:

1. Catalog was associated with View (association formed)

Contract CO14: searchVehicleCatalog
Operation: searchVehicleCatalog()
Cross References: Use Case: Search Vehicle Catalog.
Preconditions: Administrator login session is present
Postconditions:

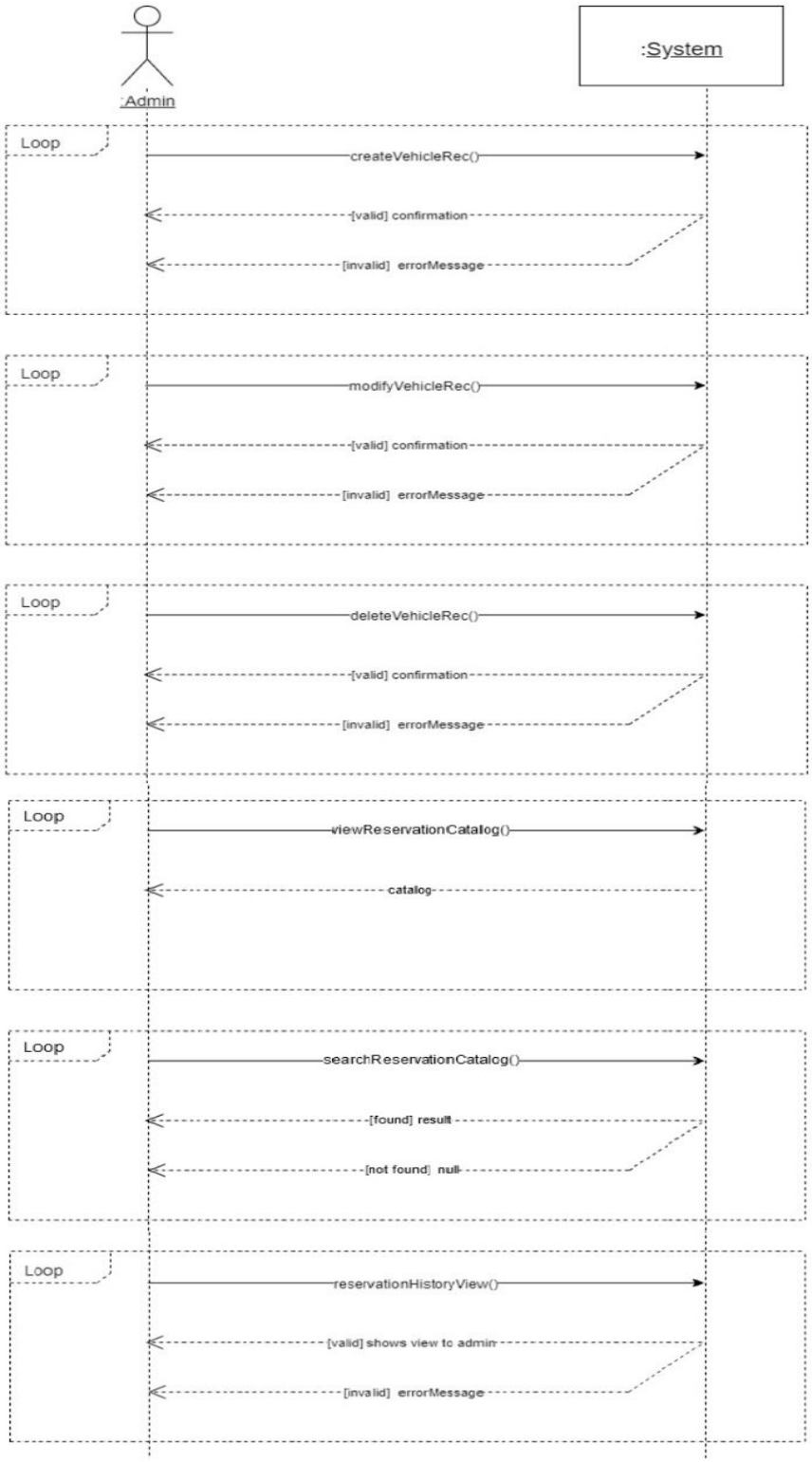
1. An instance of Search Catalog **sc** was created (instance creation)
2. **sc** was associated with vehicleCatalog (association formed)

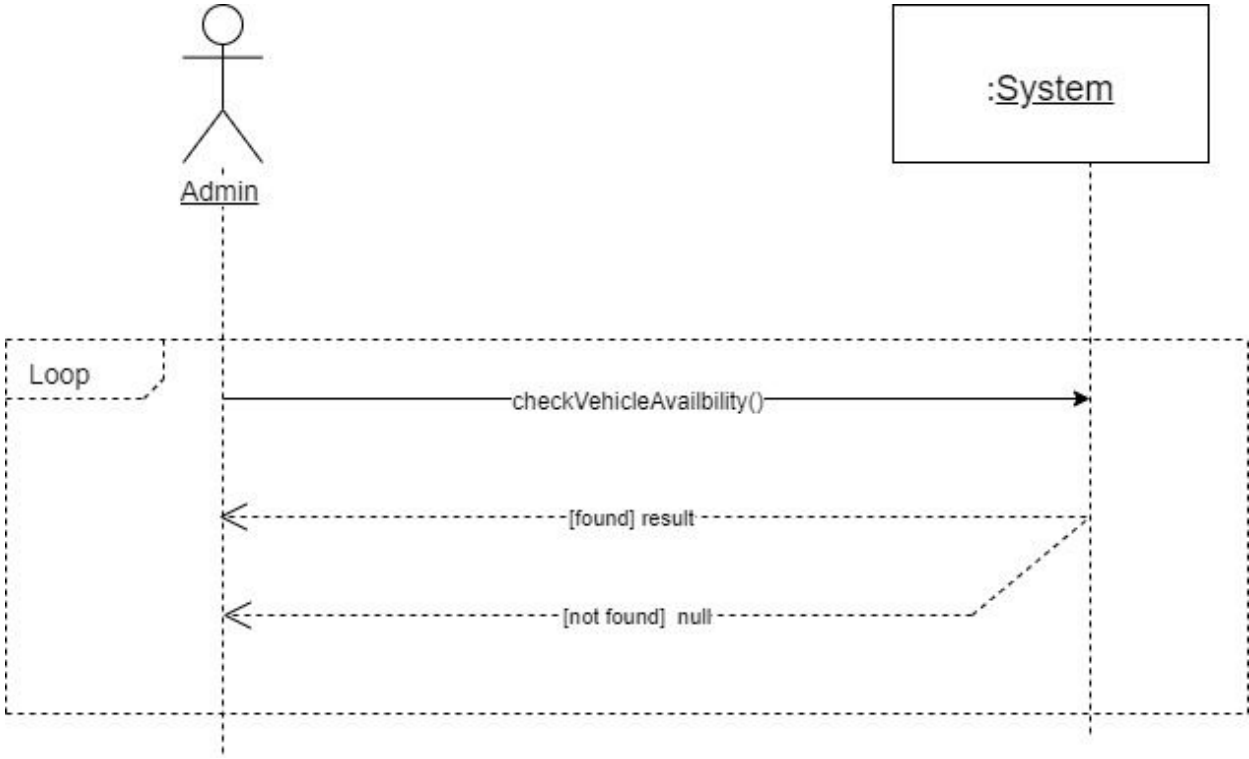
Contract CO15: reservationHistoryView
Operation: reservationHistoryView ()
Cross References: Use Case: Client Reservation.
Preconditions: Admin is authenticated.
Postconditions:

1. An instance of ReservationHistory (rh) collection is created.
2. An instance of rh was retrieved from ReservationHistoryService (rhs) based on the specifications or constraints provided (association formed).
3. Attibutes of ModelAndView (mv) were initialized

Contract CO16: checkVehicleAvailbility
Operation: checkVehicleAvailbility (spec)
Cross References: Use Case: Check Vehicle Availability
Preconditions: Administrator login session present
Postconditions:

1. An instance of vehicle v was created (instance creation)
2. **v** was associated with vehicleCatalog (attribute modification)


:Admin
:System
Loop
createVehicleRec()
[valid] confirmation
[invalid] errorMessage
Loop
modifyVehicleRec()
[valid] confirmation
[invalid] errorMessage
Loop
deleteVehicleRec()
[valid] confirmation
[invalid] errorMessage
Loop
viewReservationCatalog()
catalog
Loop
searchReservationCatalog()
[found] result
[not found] null
Loop
reservationHistoryView()
[valid] shows view to admin
[invalid] errorMessage



4.2. State Diagram

State diagrams are used to give an abstract description of the behavior of a system. This behavior is analyzed and represented as a series of events that can occur in one or more possible states. The state diagram for car reservation is shown below.

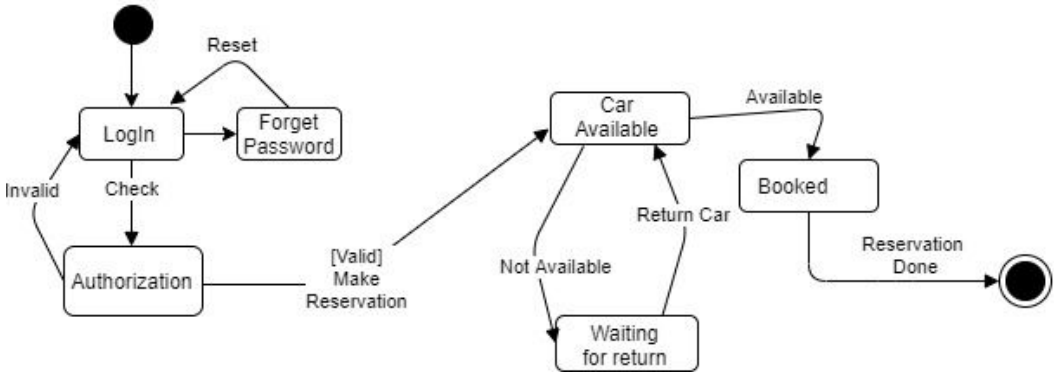


Figure 16: State Diagram

4.3. Domain Model

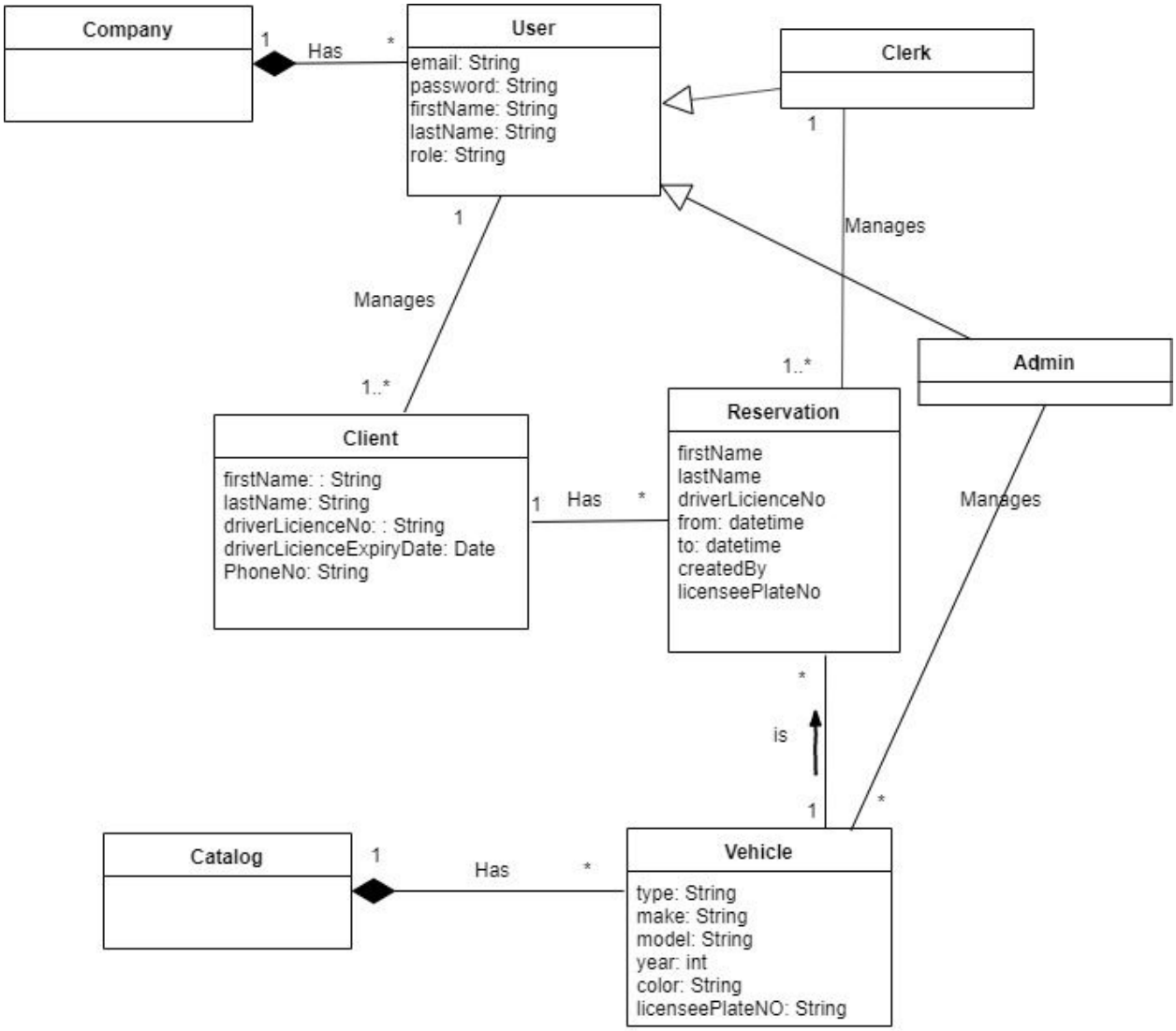


Figure 17: Domain Model