Requirements Analysis and Specifications Document

for

Taxi Management System

Version 1.1

Prepared by Giuseppe Di Francesco & Domenico Iezzi

Summary

1. INTRODUCTION 3

1.1 Purpose 3

1.2 Description of the given problem 3

1.3 Actors 3

1.4 Goals 4

1.5 Domain Properties 4

1.6 Glossary 5

1.7 Assumptions 5

1.8 Proposed system 6

1.9 Stakeholders 7

1.10 Other Considerations 7

2. REQUIREMENTS 8

2.1 Functional 9

2.2 Non-Functional 10

2.2.1 **User Interface** 10

2.2.2 **Documentation** 13

2.2.3 **Architectural Considerations** 13

2.3 Other Considerations 14

3. SCENARIOS 15

4. USE CASES 18

4.1 Use Cases Diagrams 18

4.2 Use Cases Description 19

4.3 Class Diagrams 25

4.4 Sequence Diagrams 26

4.5 State Chart 31

5. ALLOY MODELLING 33

5.1 Results 35

6. WORLDS GENERATED 36

7. USED TOOLS 41

1. INTRODUCTION

1.1 Purpose

This document represent the Requirement Analysis and Specification Document (RASD). The main goal of this document is to completely describe the system in terms of functional and non-functional requirements, analyze the real need of the customer to modelling the system, show the constraints and the limit of the software and simulate the typical use cases that will occur after the development. This document is intended to all developer and programmer who have to implement the requirements, to system analyst who want to integrate other system with this one, and could be used as a contractual basis between the customer and the developer.

1.2 Description of the given problem

The aim of the project is to build an application for requesting taxi in a big city. User can queue for a taxi using the web or the smartphone application, and the systems answers with the code for the incoming taxi along with the waiting time. The system divides queues one for each zone of the city, while each of the taxis is assigned to a specific zone thanks to its GPS coordinates. Taxi driver uses a mobile application to inform the system of their availability and to confirm that they are going to take care of a certain call. When users queue to the app in a certain zone, system forwards the request to all taxis waiting in that zone. If the first one confirms, system will send a confirmation to the user. Else, the request will be moved to the second taxi in queue, and the first will be moved in the last position of the queue. Users can also reserve a taxi specifying origin and destination of their route, only if they do that at least two hours before the ride.

1.3 Actors

In our system, there are three actors:

- User: a person who can request a taxi to the system and optionally reserve it.
- Taxi Driver: a person who drives a Taxi
- Administrator: the responsible for the web application, who register the Taxi Drivers

1.4 Goals

For a User, the software “Taxi Management System” must provide those features:

- Request of a Taxi and visualization of the Waiting Time and Taxi Code
- Reserve a Taxi for a specific ride (Not less than 2 hours before)

Meanwhile the Taxi Drivers, with this software, must be able to:

- Update the information about his\her availability
- Accept/Decline a Call
- Log In to begin a work turn

1.5 Domain Properties

Now, we define some properties we suppose that the world we consider have:

- When a user make a call for a taxi, he\she actually needs only one taxi
- Only a User who needs a taxi will call for it
- Taxi position is obtained from its GPS coordinates and the driver won’t switch off his GPS when he is in work, neither he will uninstall the application.
- If the Taxi driver confirms the call, he will instantly begin to drive the road to reach the user.
- The Taxi Driver is telling the truth about his availability
- The user who reserve a taxi will be at the chosen location for the time he has selected, in the case of a reservation, or after the waiting time instead.
- There are enough taxies to cover all the zones and their requests, so there will always be a taxi available in the city.

1.6 Glossary

We define the meaning of some words that we will use frequently:

- **GPS:** The Global Positioning System (GPS) is a space-based navigation system that provides location and time information in all weather conditions, anywhere on or near the Earth where there is an unobstructed line of sight to four or more GPS satellites. We are using it to know the position of the taxies, as well as the user position if he has it enabled in his smartphone.
- **USER:** with "user" we mean every person that uses the application to Reserve or call a Taxi, so basically everyone who uses the application except the administrator and the taxi drivers.
- **ZONE:** Our app's city is divided into zones, approximately 2 km^2 each, in order to have multiple taxi queues and have a better management of requests. We use this term to denote one of those zones.
- **REQUEST (Referred to a Taxi):** A Request, when referred to a Taxi, is a call made by a user to the system to ask for a taxi.
- **CALL (Referred to the System):** A call is a notification of any kind that the system sends to a Taxi Driver when there is a Request for that Taxi, so he can accept or decline it.
- **RESERVATION:** A Reservation lets the user book a taxi for a specific hour
- **AVAILABILITY (Referred to a Taxi):** The Availability is a state of the taxi, that defines his possibility to receive a call.

1.7 Assumptions

Some things are not clear in the specification document, so we must make some assumptions:

- The document did not say anything about the traffic, so we assume that there are dedicated lanes for the Taxies in the City, or, at most, a little traffic. In this case, we will be able to use the road limit speed to calculate the waiting time for the user.
- If the Taxi Driver does not answer to the call after a certain time, we assume that he\she has not accepted. Therefore, the system must set a sort of "Timeout" after which the system must do a call to another taxi.
- We assume that the system has a special section for the taxi driver, which is hidden for the user. Of course, only the administrator can register a Taxi Driver and give him his access credentials. The Taxi Driver must also specify the number of the Taxi when he notifies his availability.

- The system assign a zone to every taxi in the moment he notifies his availability. We assume that if a taxi moves from a zone to another zone because of the user request, then the taxi is invited to return near his zone after the drive before he notifies his availability, especially when there are zones with more affluence.
- The document didn't say anything about who register the Taxi Drivers in the system, so we consider that there is a person responsible for this (the Administrator) that can log with special credential in his section.
- If a taxi have a malfunction when transporting a user, the user can simply do another call and take another taxi from that position. We assume that a taxi can't have a malfunction while moving toward the user (so before the user takes it), but we will consider this for a future improvement of the application.
- Even if every taxi driver has his own taxi assigned, we will associate the taxi number instead of the driver to our log system, to prevent some mistakes if a taxi exchange occurs. In fact, our system will be able, as an additional feature, to recognize if there will be some drivers that are not driving their own taxies.
- Zones are divided in squares: otherwise is very difficult to check them.
- The system must not call the same taxi two times for the same request (it can happen if the driver declined the call while he is the only taxi in the zone)

We think also that the user must know before if there are some taxies available near his\her position, so that he can think of make a call or not if he is in a hurry. Of course, we wanted to make the software minimal and very easy to use, but a *map or something with the availability of taxies will have to be shown before the reservation*.

This last function, though, will have a secondary priority among the base functionalities. More important is, instead, to create a confirmation task for the user if the waiting time would be more because of the lack of taxies in that zone.

1.8 Proposed system

We suggest making this Software initially as a Web Application and, after that, convert this into an application for Smartphones, in particular Android, iOS and Windows Mobile (in that developing order). This choice is made first because the Web application is the easiest way for a user to access to the service, and second because it is easy for a developer to convert a Web Application into a Smartphone App, for example with tools like Apache Cordova or Phonegap. Of course, the layout of the web application will be designed to run well on a Smartphone's Browser, so that the layout of the app won't be so much different.

As we said before, the design must be minimal for the user, which with a few button can insert his position, his name and request a Taxi. We are also thinking about using the GPS, if the user has it active on his smartphone, to retrieve his actual position. Another thing to consider is the way in which he can receive notifications: it is easy with a Smartphone app, but less easy when he is using a web app. Therefore, we are considering sending an E-Mail to the user, that he can easily check from any smartphone or PCs. We will also use the cookie system to save a pending session.

Instead, the Taxi Driver has a dedicated application, installed in his taxi (With the Taxi Code already memorized in the app), that shares the same database as the web app. After the login, he will receive notification in this terminal. Of course, it will have the GPS.

1.9 Stakeholders

The primary stakeholders are people who need to move within the city, so for example they can be tourists, people that cannot find an alternative public transport for a certain hour or just richer people that find it a more comfortable conveyance. Of course, having a faster and easiest way to call a taxi can increase its comfort and more people will use it. The other primary stakeholders are of course the taxi drivers, which will see it like a way to simplify their work and have more customers.

A secondary stakeholder is the people who committed us this work, for instance our University.

The other stakeholder is the city, which will become more “technologic”, and will appear more moved forward (advanced).

1.10 Other Considerations

Development of this application should follow three simple rules:

- **Easy to use:** the application interface and functions will be accessible for every user; it will follow common UI patterns for web application, so that advanced users will immediately feel confident with it, while unexperienced users will easily understand how to use main functions.
- **Stability:** all the functions should work out-of-the-box without errors.
- **Nice look and feel:** application will be nice to see, with a clean and well-designed interface

2. REQUIREMENTS

Taking into consideration the Domain properties and the assumptions written in the First Chapter, from our goals we can obtain our Requirements.

- **Request of a Taxi and visualization of the Waiting Time and Taxi Code** [User]
  - The System Must be able Obtain the Position of all Taxies in the Zone.
  - The System must obtain the start position of the user and associate it with the right zone, with his GPS if he has it active, or instead allow the user to insert the start position manually (in this case it must verify it).
  - The System must have the ability to make a queue of the taxies in the zone and iterate the call until an actually available taxi driver is found.
  - The System must be able to check the fastest road from the taxi position to the user position, then to calculate the approximate waiting time and show it to the user with the Taxi Code.
  - The System must be able to recognize if a zone has no taxi available. In that case , it must iterate the call to other queues.
- **Reserve a Taxi for a specific ride (Not less than 2 hours before)** [User]
  - The System must memorize the user information about the reservation.
  - The System must be able to check for available taxi in the source's zone 10 minutes before the start time, and when found send the confirmation with the taxi code to the user
- **Log In** [Taxi Driver]
  - The System Must Provide an authentication Service for the Taxi drivers.
  - The Software Administrator must be able to memorize the data about the Taxi Drivers and provide them login credentials.
- **Update the information about his\her availability** [Taxi Driver]
  - The System must automatically set as “not available” a Taxi Driver when he accepts a Call, but then it must allow the Driver to report his status either if he will be available again, or if he has some unexpected issues.
- **Accept a User Call** [Taxi Driver]
  - The System Must Allow the Driver to accept the ride or not.
  - The System must report the Driver decision to the user.

2.1 Functional

We have listed the main requirements, so here we show the features that the software must have according to the users that will use it.

USER can:

- Insert his Position (Using his GPS or manually)
- Request a Taxi and see the Waiting Time
- Make a Reservation for a certain time in a certain position (Min 10 minutes early and Max 2 hours early)
- Show a Map with the Taxies locations *(secondary)*

TAXI DRIVER can:

- Log In
- Report Availability to the System
- Accept or Decline the System Call
- See if he is in his zone or not
- End Their Turn (Which means to Logout)

ADMINISTRATOR can:

- Log In
- Add a Taxi Driver, this means to store his basic information (like his name, surname etc.) and to give him access credentials and a Taxi Code.
- Delete a Taxi Driver
- Move a Taxi Driver from one Zone to Another.

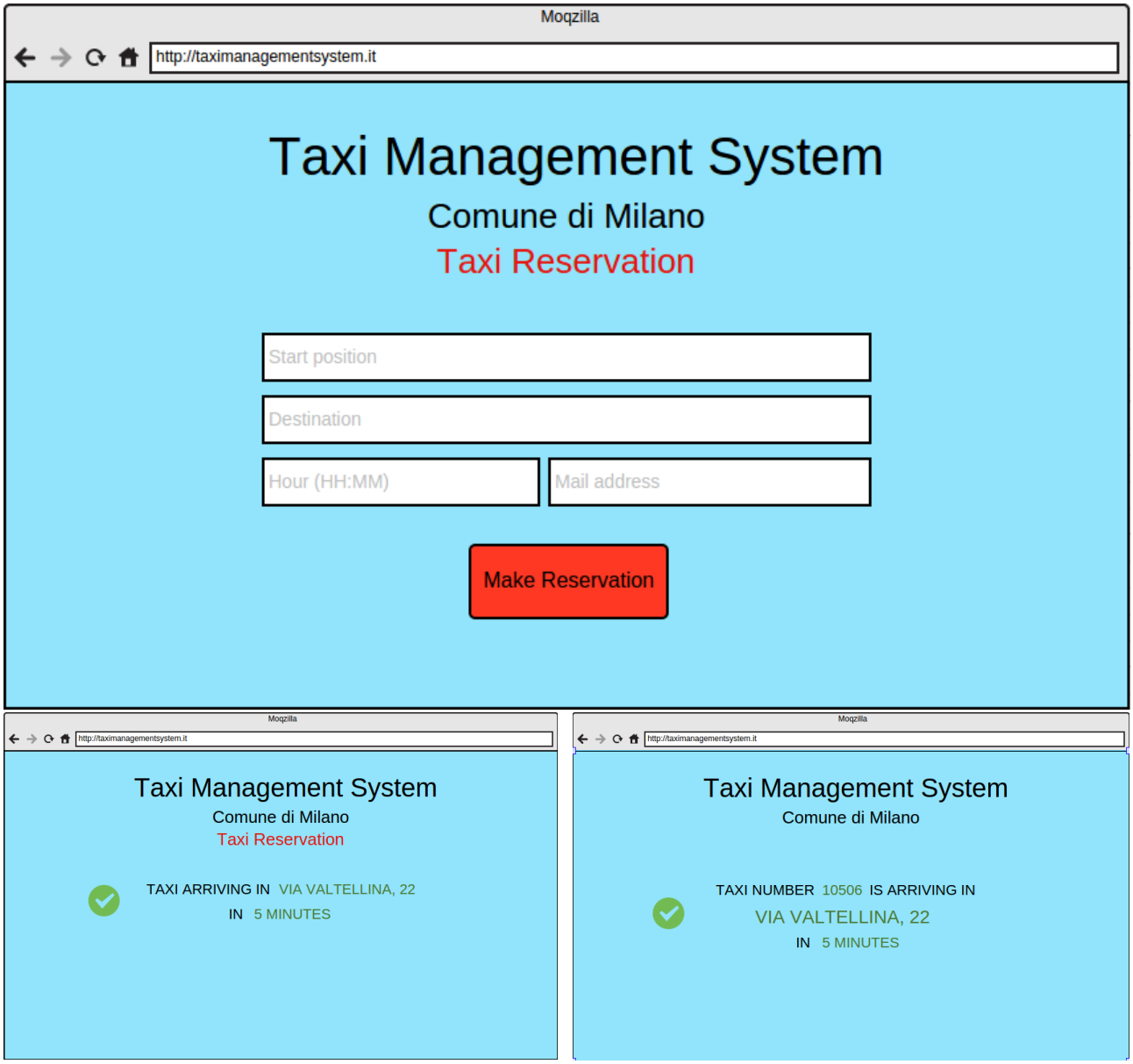
2.2 Non-Functional

2.2.1 User Interface

The Interface, as we said, will be simple and minimal, and mobile-oriented. When the user open the app, he will see a beautiful background with one main button: Reserve a Taxi. There will be of course also a text box to select the position, by texting the address (Manually or using the GPS). If the user writes an unrecognized address, after the click on the button, a red "X" appear, with the message "the address isn't in this city". If there is not a taxi in the zone, instead, a message will appear, saying that there are no taxies near his position and the waiting time could be long. So, if the user confirms, he will receive the waiting time information and the taxi number, otherwise the call will be cancelled.

For the Taxi Reservation function, there will be another button, which will show up a simple form with the destination, time and the E-Mail for the confirmation. Notice that for the Smartphone App, the e-mail insertion will be optional.
For the secondary requirement to show a map to the user, we can simply add another button "Show Map" which behavior is similar to the "Request a Taxi" one, but it will simply show the user a map with his position and the taxies all around. Then he can go back and decide to make a request to the system.

2.2.1.1 Examples of the Main View of the User. On the second one, the address is not found in the system.



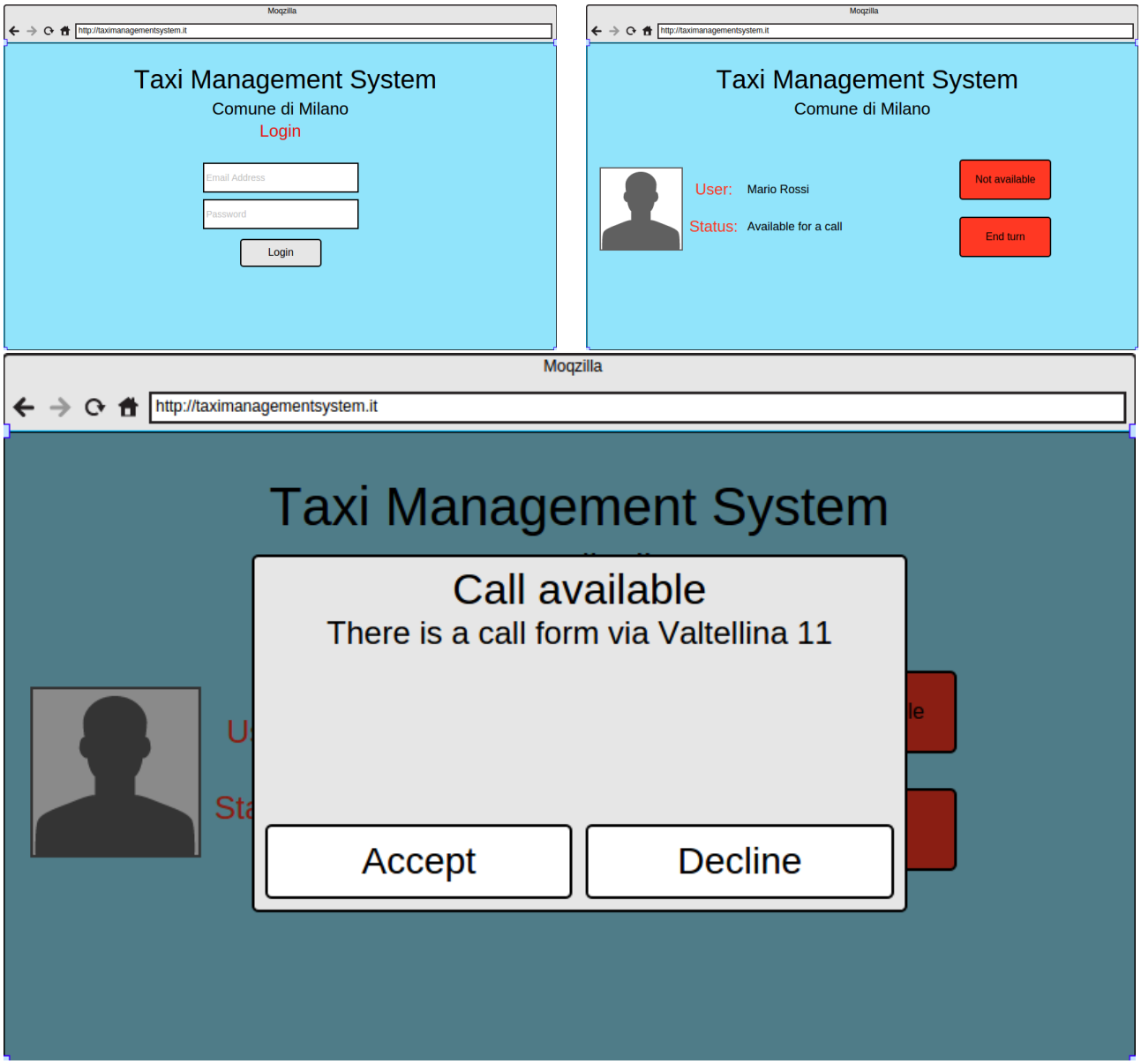
2.2.1.2 Other examples of the Main View of the User. First, there is the form for the taxi reservation, the last two appears when a taxi is requested.

For the Taxi we must think about a Car System, so there is a similar approach to the interface, simple and maybe with bigger buttons, to make easier for the driver to click them in every moment.

At the beginning a simple login form appears. After that, there is a simple view with a button “tell that you are available” and another one to End the Turn. Then there will be only a window telling that you are waiting for a call, and the button turns to “Tell you’re not available”.

When a call arrives, the address of the user appears with two buttons: ACCEPT and DECLINE. If he accepts, the availability automatically switch to “NO” and a new Button “End Ride” appears. A *red alert* will appear also if he is not in his zone.

We can think also to add some visual effects and sound notifications for the Taxi Driver to attract his attention when there is a Call or the red alert. If the on-board machine is using also other application, we can also build a notification system like the one on smartphones.



2.2.1.3 Examples of Views of the Taxi Board

For the Administrator, instead, after the login, similar to the Taxi Drivers one, there is a menu for the various functions. There will be an "insert Driver", with a form with all the information for the taxi driver to insert, or a "See all Drivers" function with a search field, in which he could delete or change someone's zone.

2.2.2 Documentation

We will write those documents to organize our work, so to do the best thing in the minor time possible:

- **RASD:** Requirement Analysis and Specification Document, which purpose has been described in the 1.1 Chapter.
- **DD:** Design Document, to define the real structure of our web application and its tiers.
- **Comments in the source code:** to make easier for anyone that wants to develop (or do maintenance) to understand the code.
- **Installation Manual:** a guide to install this Software.
- **User Manual:** a guide to use this Software.
- **Testing Document:** a report of our testing experience of another similar project

2.2.3 Architectural Considerations

To meet all of those requirements, we will use the MEAN framework to create our Web Application. MEAN is the acronym to "MongoDB, Express, AngularJS, Node", that are the four JavaScript framework we use to create our server and client website. In particular: Node is the core of our server; Express is a framework that helps us to use Node and his functionalities, providing us also more security; MongoDB is a minimal and efficient database system for Node; AngularJS is the Client framework to easily retrieve information from our database. Then, with "PhoneGap", we will create our mobile application.

The information that we will store in our Database will be shown on the Class Diagram below, but they will be clearer in the Design Document.

To run the WEB application a user will need, other than an internet connection, a PC or a Smartphone with an updated browser, while to run the App he will need an iOS, Android or Windows Mobile Smartphone. It's the same for the Taxi System, but with the addition of the mandatory GPS system.

2.3 Other Considerations

We will write down here additional considerations to make for this software

- Every Taxi has a taxi code, which depends on the physical taxi and not on his driver, even if every taxi belongs to a driver in particular. Therefore, we decided to associate the code to a taxi in a way that it's independent from the login, and that the Taxi Driver can't modify it from his system. This can be done associating a code to the app when it is installed in the Taxi system for the first time, so with a sort of first time operation that can't be undone, unless we uninstall and install again the application. (Note that in the domain we considered that a taxi driver won't do this, we can also think to make a blocked system in the future).
- We didn't think about the "cancel reservation" function, because we considered as an assumption that every user that reserve or call a taxi, actually needs it. It will be useful to give a user the possibility to cancel the reservation, so we will consider it as a possible additional feature in another version of the software.
- We think that starting from this base application, in the future we can build also a taxi-sharing system, starting from a simpler system without user logins, but also adding user login. In that case we think that the user login will be useful only to the taxi-sharing function, so that it will be optional.
- The system must check if a taxi has confirmed his availability outside his zone, so that a taxi driver is encouraged to return to its zone if needed. Note that an administrator can change a taxi zone in every moment, so this can be also a way to notify the driver that his zone has been changed.
- We choose 10 seconds as the waiting time to consider the call as "Declined". We will test this and see if it creates a good balance between the potential waiting for the user to receive a response against the probability that the driver cannot respond in only 10 seconds even if he wants.
- There is not a backup of the queue in the database, because there is not a certain logic for the creation of the queue, so even if the system crashes, the queue will be built again based on the available taxies on the respective zones. This choice is made to make the application faster.
- We haven't considered the destination of the user as important, but in the future we can consider it as a relevant information, especially if we are building a taxi-sharing system. It can be also useful for to anticipate taxi positions and organize zones (with sending alert for example) according to that information.

3. SCENARIOS

SCENARIO 1

1. Mikael is a taxi driver beginning his working day. He logs in into the system with the on-board terminal on his vehicle with the provided username and password. A welcome screen appears saying he is in queue for pending requests for zone A, but he is not in his zone already.
2. Mikael moves to zone A and then notifies his availability.
3. The application notifies Mikael about an user request, that appears to be in zone A, and a confirmation dialog to accept or decline the request appears. He decides to accept and begins driving to the given position. Mikael now is not available anymore for the system.
4. 4 minute after the confirmation, Mikael arrives, sees the user calling him and takes the user to his destination.
5. Mikael is outside his zone, but he choose to notify his availability to the system.
6. After some minutes, a red message appears inviting him to return to his zone, so he removes his availability, moves to his zone and notify his availability again, so that the system can change the queue.

SCENARIO 2

1. Steven is a tourist in the city of Milan. He doesn't feel confident about the public transport system, so he decides to take a taxi. He sees and advertisement about the Taxi Management System of Milan, so he downloads the app.
2. Steven opens the app and uses his GPS to retrieve his position, so he instantly sends a request.
3. Roberto, a Taxi Driver, is the first in the queue, so he's the first to be called, but he is speaking on the phone, so he declines the call.
4. Gianni, the second Driver, is temporarily outside the taxi and have forgotten to change the availability, so, after 10 seconds, the call has considered as declined.
5. Giovanna is the third Taxi Driver: she receives the Steven's call and accepts it.
6. Steven receives Giovanna's taxi code and the estimated waiting time for the Taxi arrival, which sets to 5 minutes.
7. After 5 and a half minutes Giovanna arrives and Steven recognize her by her Taxi Code. Steven moves his and to call Giovanna and tells her his destination. She drives him there.

SCENARIO 3

1. Many friends suggests Alex to try the new Taxi Reservation service that his city offers since some weeks. He decides to give it a try despite he does not feel at ease to use web application and smartphone for services.
2. He loads the web page with the browser on his PC, he first wants to ensure the service is working by opening up the map with the available taxies position. He inserts the position and click on the Show Taxies.
3. He has written the wrong address, so a red cross appeared with a message error. Then he writes the correct address and click "Show Taxies" again: a windows pops up showing many taxi markers on the map near him, so he feels more comfortable.
4. Alex decides to book a taxi for the evening, in order to go from his house to the stadium (Zone A): he selects reserve a taxi, and a form shows up with the fields Source, Destination, Hour, E-mail and a confirmation button. After compiling and sending the form, a message opens up saying that, some minutes before the ride, he will be notified by an e-mail about the availability of the taxi for the selected hour and his code.
5. 10 minutes before the meeting Matteo, a Taxi Driver in Zone B (near Zone A), receives a notification about Alex's ride: he accepts the call and begins driving to the meeting place.
6. Alex receives an e-mail containing Matteo's taxi code and a remainder of the selected time and place. Therefore, he goes to the right position.
7. Matteo arrived some minutes before the reservation, Alex sees the Taxi with the correct code and Matteo drives him to the destination point.

SCENARIO 4

1. Carla is a traveler that is thinking about requesting a taxi, but she is in a hurry.
2. She opens the web application without downloading the app and clicks on request a Taxi. She has the GPS on, so her position is automatically detected.
3. After some minutes, the system informs her that the waiting time can be of some more minutes because there are not available taxies in her zone, so Carla decides not to confirm the request.
4. She close the website and begin finding the correct metro.

SCENARIO 5

1. Mikael is a Taxi driver that begins his usual working day making the login to the screen. He is already in his zone (Zone B) and tells his availability.
2. Mikael receives a call for a ride, but he is currently busy and he has not turned off the availability, so he declines the call.
3. Mikael then receives a call for another user in "Via Golgi, 2" and accepts it.
4. Mikael arrives in "Via Golgi, 2" and sees the user. The user tells him his destination, so Mikael begins to drive.
5. While driving Mikael has a problem with the taxi, so he tells the user to get off the car, but he doesn't tell the system that he is available.
6. He calls for help and after some time he solves his problem, so he tells the system he is available again.
7. He doesn't receive a call until his turn ends, so he clicks on "End Turn" and the login screen appears on the taxi.

SCENARIO 6

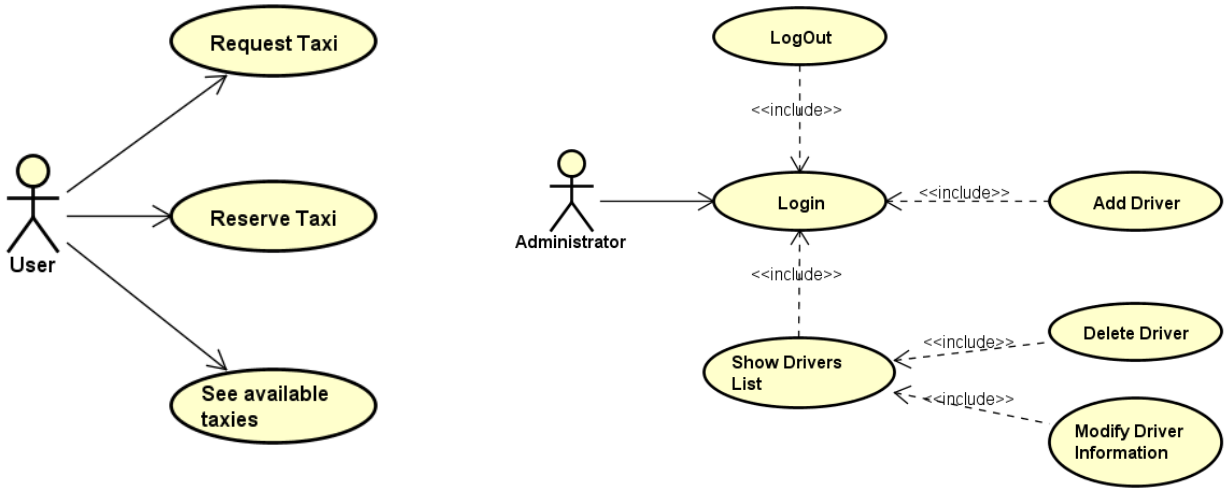
1. The Administrator receives a call from the direction: he must add a new taxi driver called “Matteo Pileri” in the database, move “Marco D’Alessandro” from Zone A to Zone C and remove “Mario Rossi” because he doesn’t work in Milano anymore.
2. The Administrator logs in with his admin credentials
3. The Administrator clicks to “add a new driver” and adds “Matteo Pileri”.
4. The Administrator goes to “View All Drivers” and types “Marco D’alessandro" in the search bar
5. He founds two namesake, so he calls the direction to know about his tax code to identify him.
6. He clicks on “Modify Driver Zone” and changes it from Zone A to Zone C.
7. He returns to the search bar and writes “Mario Rossi”. Then he founds him and delete him.
8. He calls the direction and confirms the operations. He also gives them the credential for the login of the Driver “Matteo Pileri”.

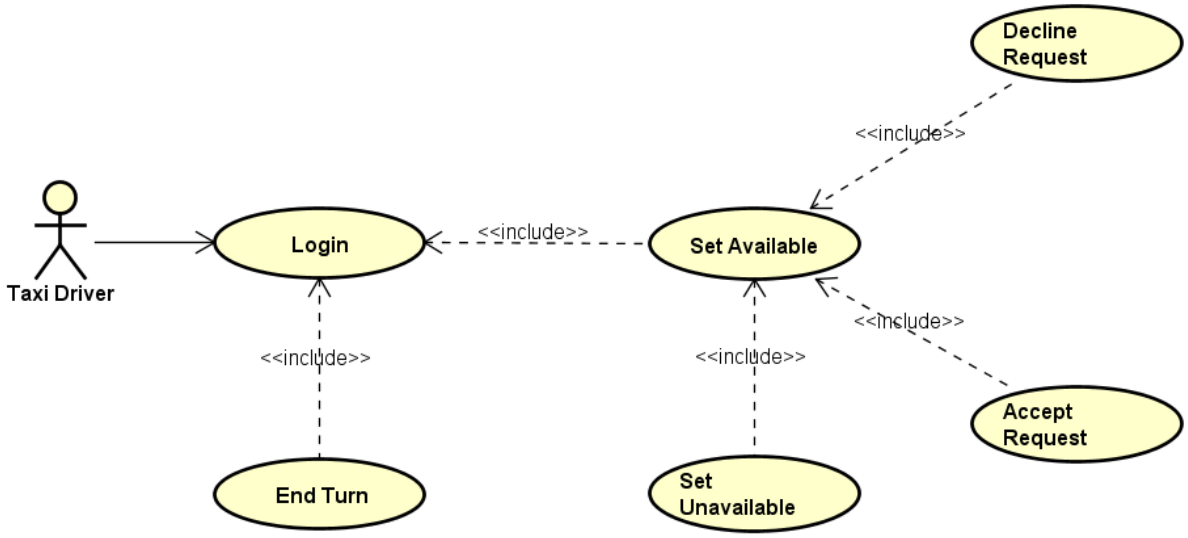
4. USE CASES

We now define use cases derived from our scenarios, in order to have a complete view of the system depicted in the scenarios. Use cases are:

- RequestTaxi
- ReserveTaxi
- ShowMap
- Login
- AcceptRequests
- DeclineRequests
- SetAvailable
- SetUnavailable
- EndTurn
- ShowDriversList
- AddDriver
- ModifyDriverInformation
- DeleteDriver
- Logout

4.1 Use Cases Diagrams





4.2 Use Cases Description

We want to refine the **RequestTaxi** use case:

Name	RequestTaxi
Actors	User
Entry Condition	User opens web page or smartphone application
Flow of Events	• Application presents the request form • User inserts his position on the text box of the form • User sends his request clicking the submit button of the form • The position is validated by the application • Application sends this request to the server • The Server calls the taxies in the zone • Application prompts to User information about taxi code and waiting time
Exit Condition	Use case terminates when User receives confirmation about taxi found.
Exceptions	If user enters a wrong address, system shows an error message below the text boxes and User cannot continue the reservation. If there are no taxies available in the user's zone, he must confirm the request so that the system can call taxies in other zones.

We want to refine the **ReserveTaxi** use case:

Name	ReserveTaxi
Actors	User
Entry Condition	User clicks Reserve Taxi button on the main screen
Flow of Events	• Application prompts the user a form containing source, destination, hour, e-mail and a confirmation button. • User compile the form with his data, then sends request using the appropriate button. • Application sends this request to the server. • The server register the reservation • Application notifies user that his request has been registered, and that 10 minutes before the meeting he will receive the taxi code
Exit Condition	Use case terminates when the reservation is transformed into a request and the user receives the confirmation with the taxi code
Exceptions	If user enters wrong adresses, system shows an error message below the text boxes and User cannot continue the reservation

We want to refine the **ShowMap** use case:

Name	ShowMap
Actors	User
Entry Condition	User clicks the button to show the map
Flow of Events	• User inserts the location where the maps should point in the text box • User clicks the button “Show Map” • Application shows User a map of the city with markers, each one representing an available taxi, in the zone specified by User
Exit Condition	There is no exit condition
Exceptions	If user enters a wrong address, system shows an error message below the text boxes and User cannot continue the reservation

We want to refine the **Login** use case:

Name	Login
Actors	Taxi Driver or Administrator
Entry Condition	Driver/Administrator opens his terminal or device
Flow of Events	• Application on the terminal prompts Driver/Administrator for his credentials • Driver/Administrator inserts username, password and click the submit button • If it's a driver, his log is stored as a Work Turn • Application show the main screen
Exit Condition	Use case terminates when main screen for Driver or Administrator appears

We want to refine the **SetAvailable** use case:

Name	SetAvailable
Actors	Taxi Driver
Entry Condition	Driver needs to have finished LogIn use case or the SetUnavailable use case
Flow of Events	• Driver click the button in the main screen to make himself available for requests • Application shows in main screen a message stating availability
Exit Condition	Use case terminates when Driver is correctly available for the system

We want to refine the **SetUnavailable** use case:

Name	SetUnavailable
Actors	Taxi Driver
Entry Condition	Driver needs to have finished the SetAvailable use case
Flow of Events	• Driver click the button in the main screen to make himself unavailable • Application shows in main screen a message stating the Driver is currently not available
Exit Condition	Use case terminates when Driver is unavailable for the system

We want to refine the **AcceptRequest** use case:

Name	AcceptRequest
Actors	Taxi Driver
Entry Condition	Driver needs to be available.
Flow of Events	• Application shows to Driver a request in his zone, along with a confirmation dialog that lets him decline or accept. • Driver accepts the request
Exit Condition	Use case terminates when Driver accepts the request
Special Requirements	Driver needs to accept the request in 10 seconds, else the request will be automatically declined

We want to refine the **DeclineRequest** use case:

Name	DeclineRequest
Actors	Taxi Driver
Entry Condition	Driver needs to be available.
Flow of Events	• Application shows to Driver a request in his zone, along with a confirmation dialog that lets him decline or accept. • Driver decline request and returns to main screen
Exit Condition	Use case terminates when Driver decline the request

We want to refine the **EndTurn** use case:

Name	EndTurn
Actors	Taxi Driver
Entry Condition	Driver should be on the main page
Flow of Events	• Driver clicks on the End Turn button • Application shows a confirmation Dialog
Exit Condition	Use case terminates when the logout information is stored and the application returns to the main login screen.

For the Administrator, we define the **ShowDriversList** use case:

Name	ShowDriversList
Actors	Administrator
Entry Condition	Administrator should be logged in, viewing the main screen
Flow of Events	• Administrator click on the Show Drivers button • Application shows a list of Drivers registered into the system, each one with a taxi code associated and two buttons Modify and Delete
Exit Condition	Use case terminates when the drivers' list appears

We define the **AddDriver** use case:

Name	AddDriver
Actors	Administrator
Entry Condition	Administrator should be in the drivers' list page
Flow of Events	• Administrator clicks on the Add button • Application visualize a form with the Driver name and taxi code fields, including the submit button • Administrator compiles and sends the form with new driver data • Application shows a dialog stating that driver has been correctly added into the system.
Exit Condition	Data stored into the system
Exceptions	In case of invalid input (empty or too long), system shows an error asking user to insert a valid string.

We define the **DeleteDriver** use case:

Name	DeleteDriver
Actors	Administrator
Entry Condition	Administrator needs to have finished ShowDriversList use case
Flow of Events	• Administrator clicks the Delete button on the row corresponding to the driver he wants to delete • Application prompts a confirmation dialog to the Administrator • Administrator confirm deletion
Exit Condition	Driver succesfully removed from the system

We define the **ModifyDriverZone** use case:

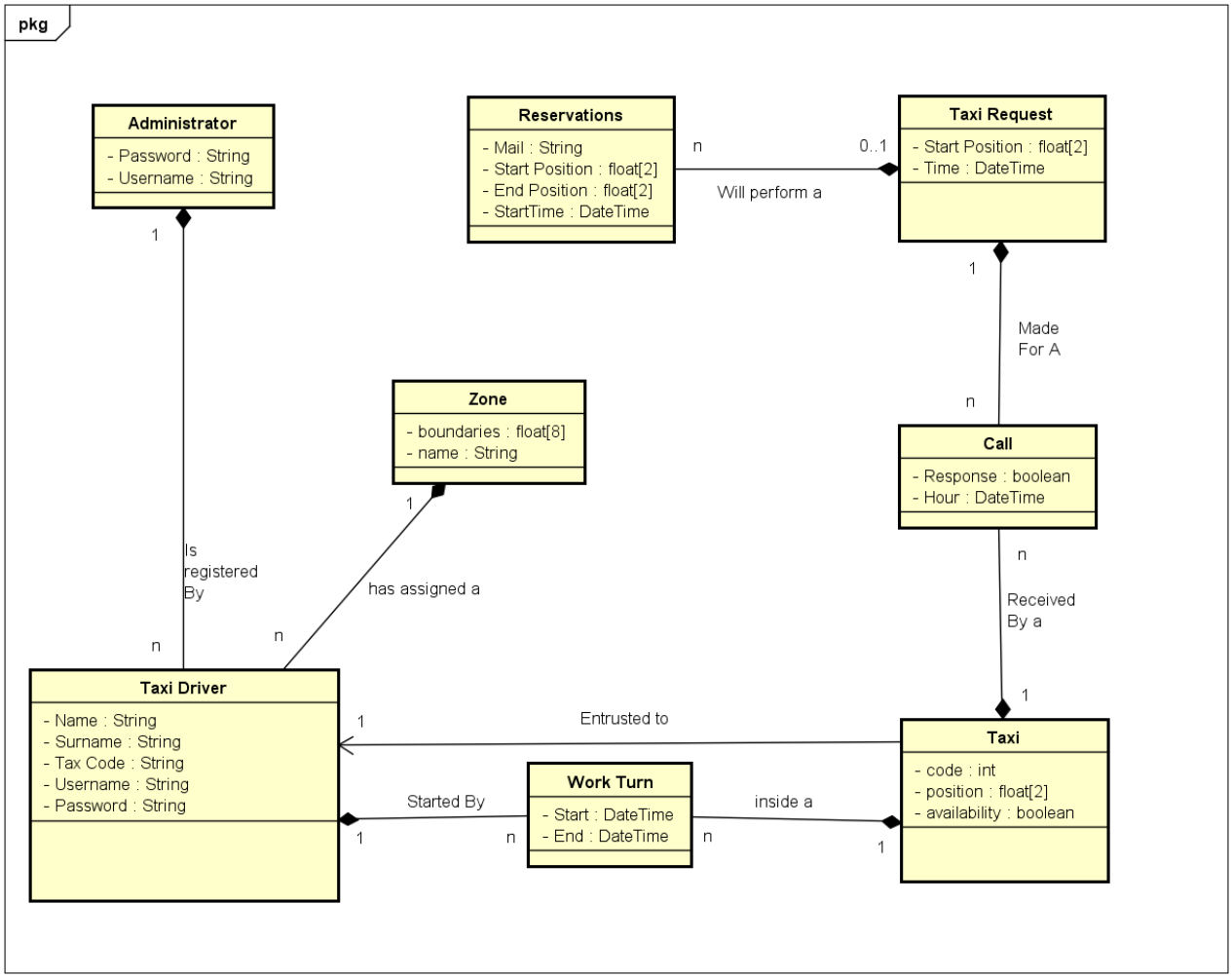
Name	ModifyDriverZone
Actors	Administrator
Entry Condition	Administrator needs to have finished ShowDriversList use case
Flow of Events	• Administrator clicks the Modify button on the row corresponding to the Driver • Application prompts a dialog with a form containing selected Driver's zone • Administrator changes zone • Administrator confirms changes clicking Submit button
Exit Condition	Data correctly changed
Exceptions	In case of invalid input (empty or too long), system shows an error asking user to insert a valid string.

We define the **Logout** use case:

Name	Logout
Actors	Administrator
Entry Condition	Administrator needs to be in the main screen
Flow of Events	• Administrator clicks on the logout button on the main screen • Application shows a confirmation Dialog • Administrator clicks Yes on the dialog
Exit Condition	There are No Exit Conditions

4.3 Class Diagrams

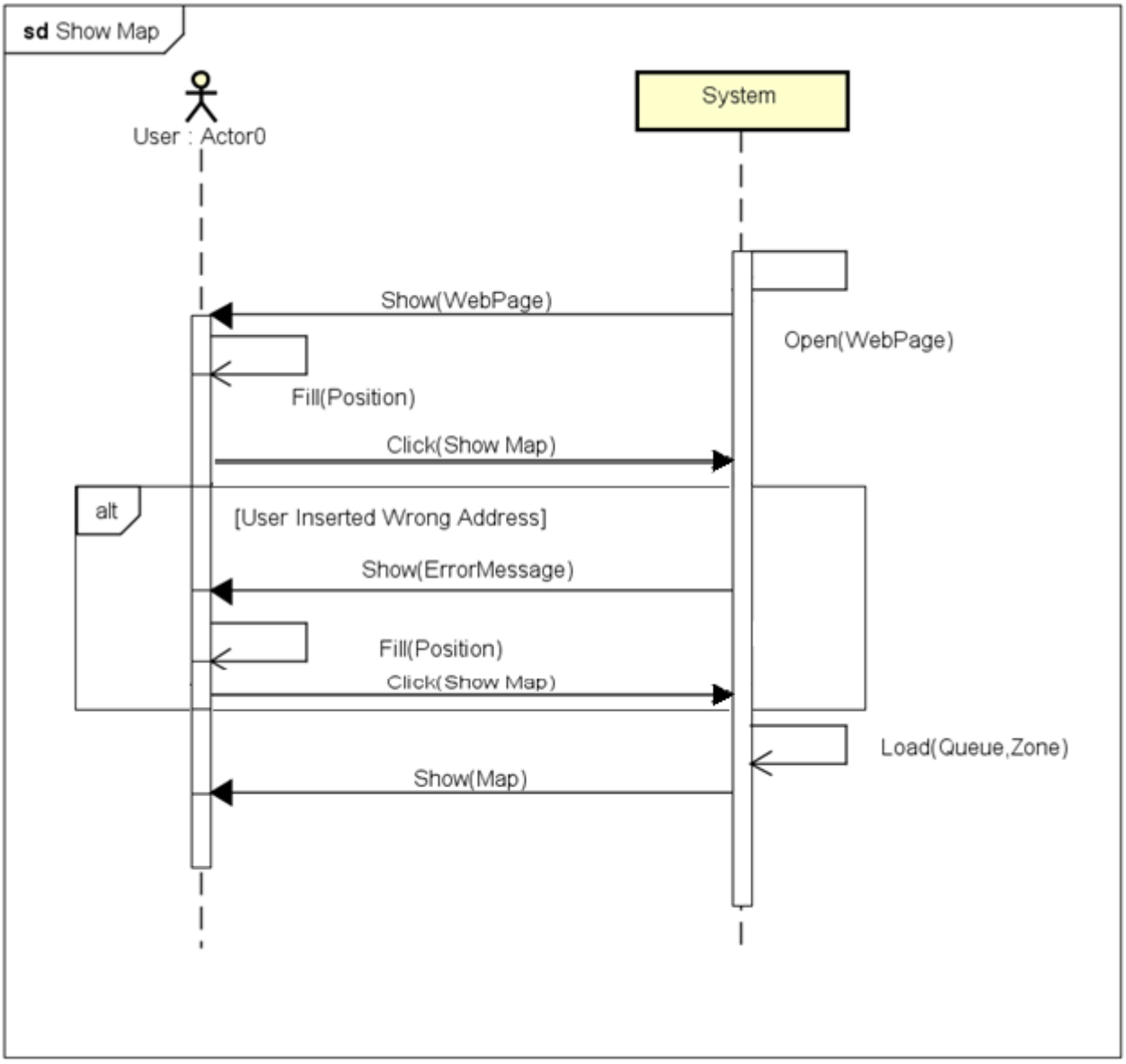
Here there is the initial Class diagram. Note that many things are missing, because we considered only the information that we wanted to store in our persistent database, while other information will be only temporary (for example users won't need to register in our system, but every time we consider he as a new user, and store only a temporary cookie to identify him). We also remember that this decision was made to make this application easier and faster to use. It won't be difficult, though, to add users in our application if we will need to implement some other function (like a feedback system for example).



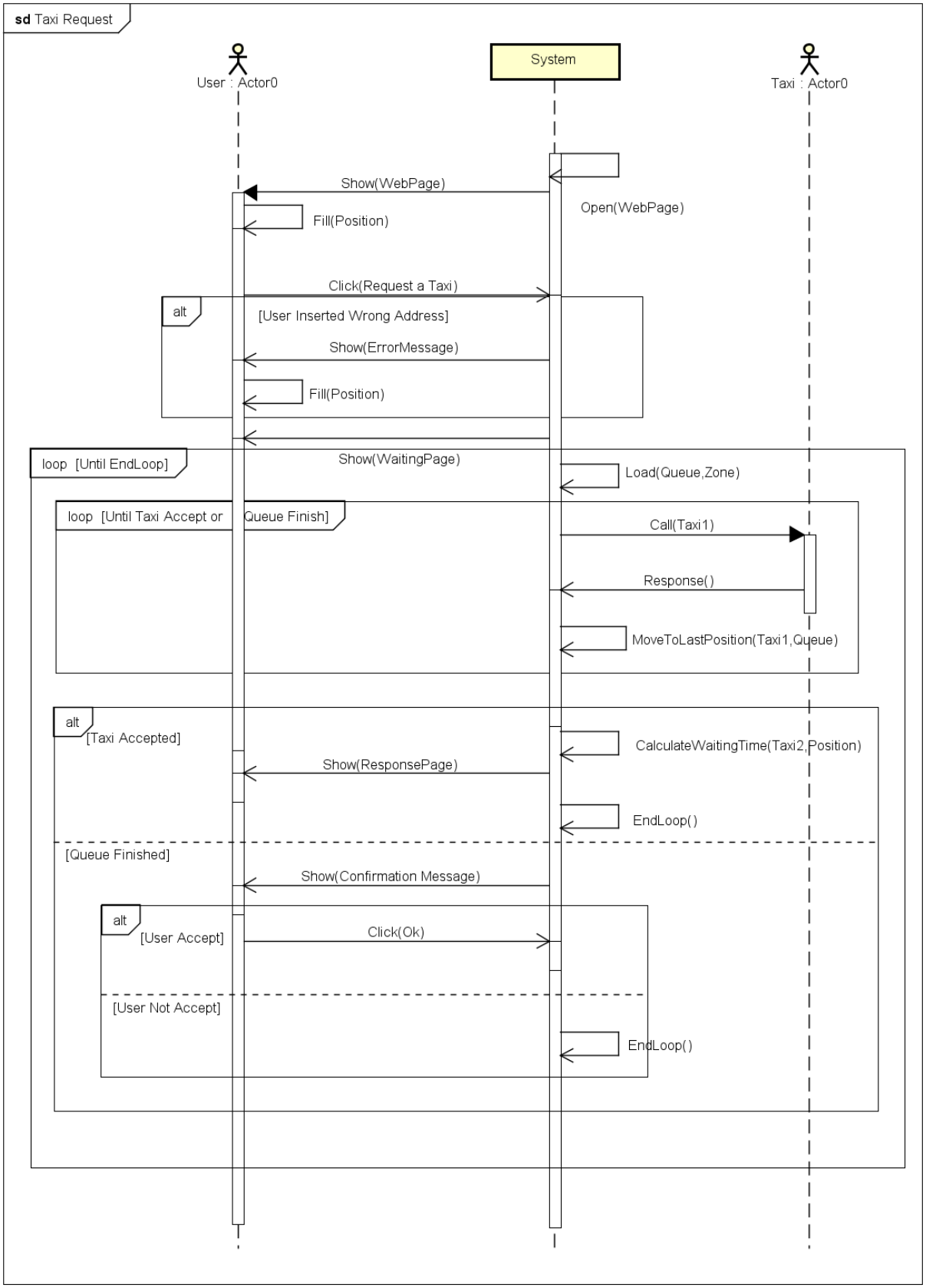
For Instance, the class Work Turn is a way to keep trace of the Taxi Driver Logins, and represent, as the name says, a turn of Work of a Taxi Driver.

4.4 Sequence Diagrams

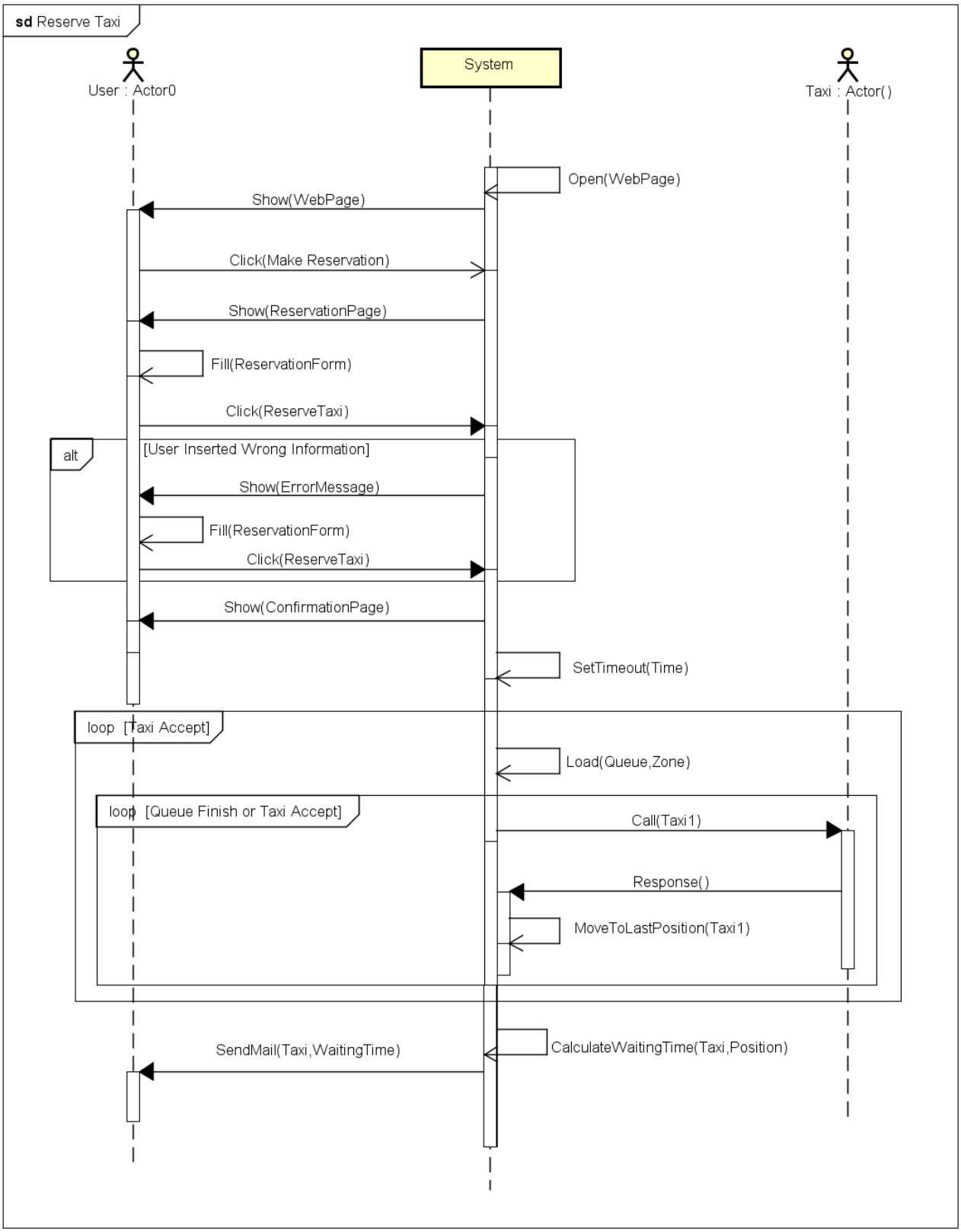
- **Show Map**



- **Request Taxi**

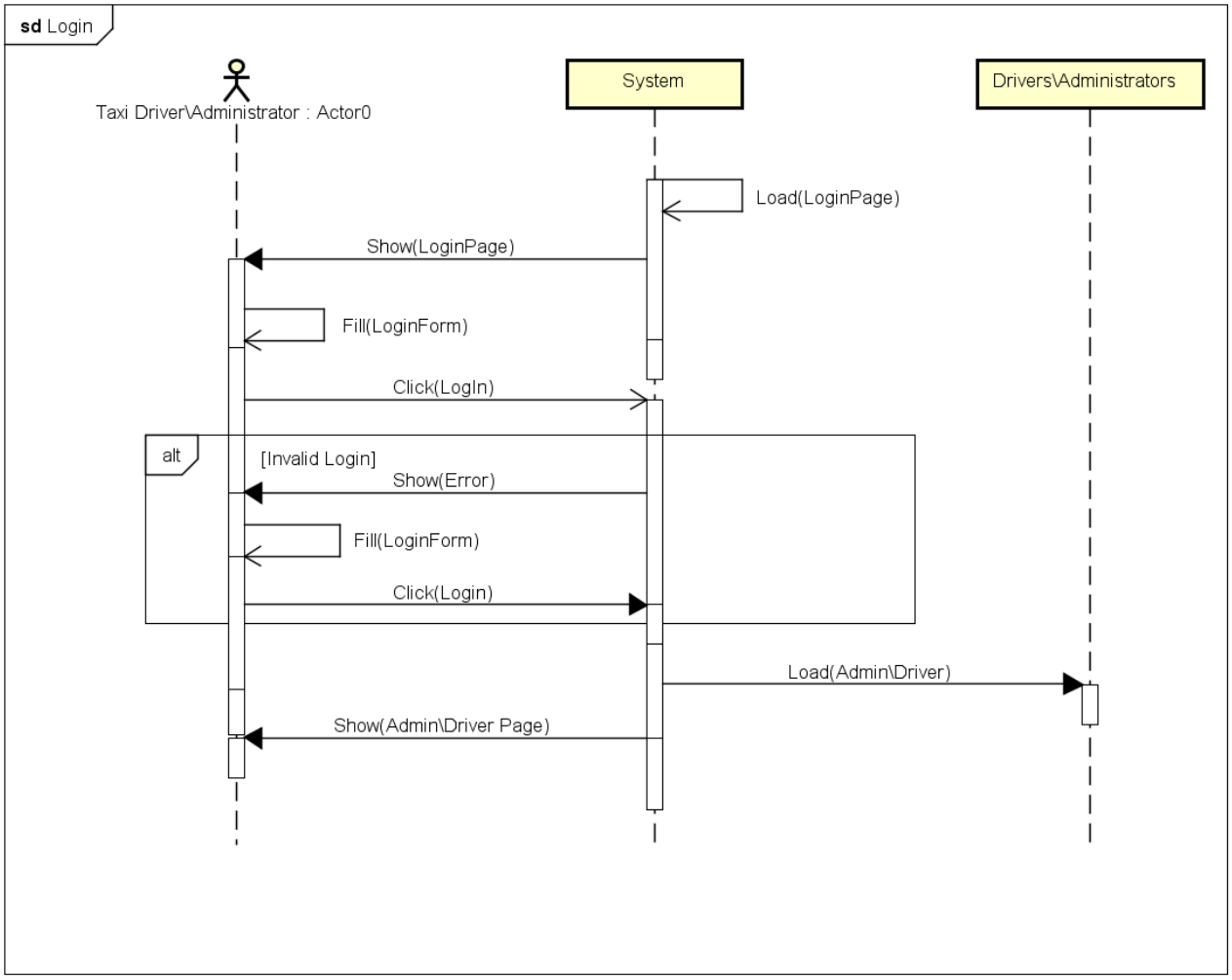


- **Reserve Taxi**



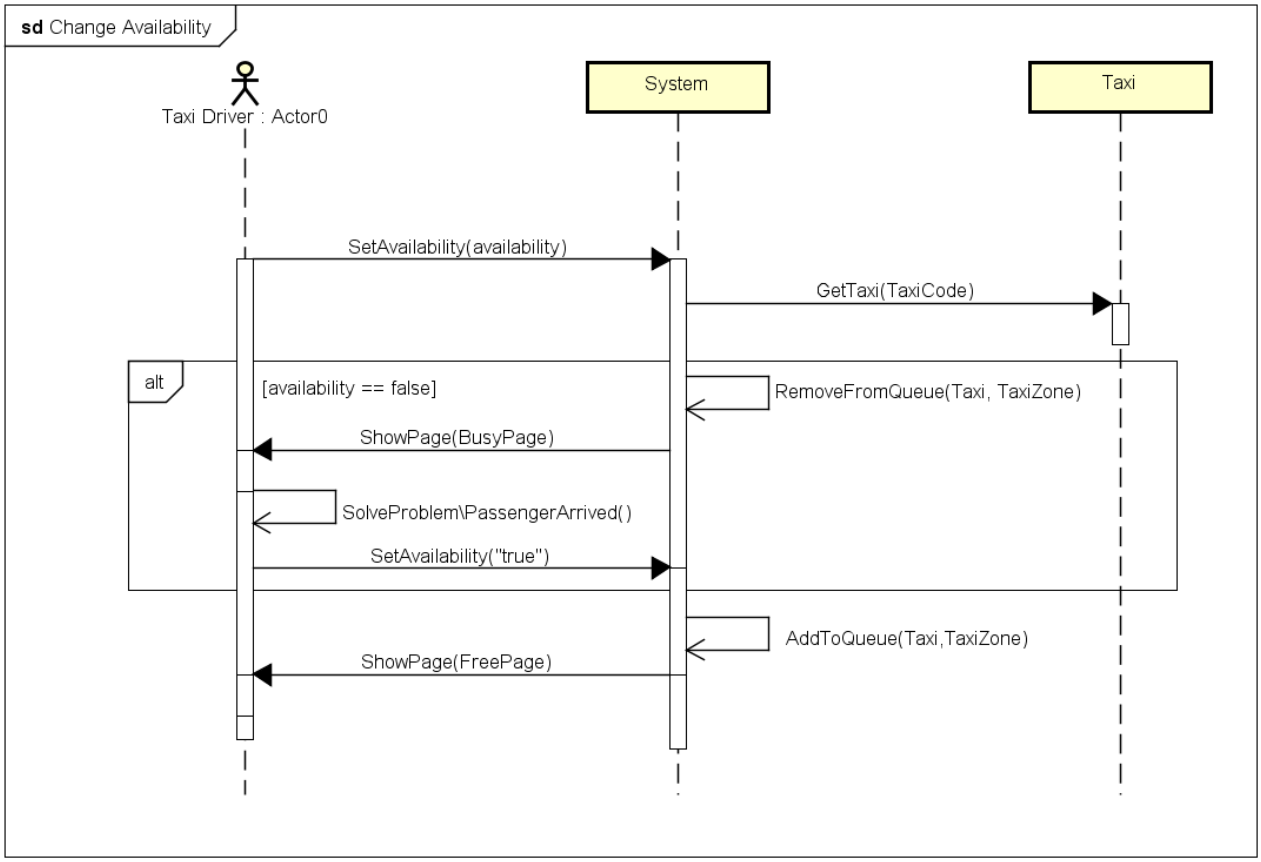
- **Login**

Notice that in this diagram a thing is missing: when a Driver makes a login, there is also a call in the "Log" part of databases, which is a place where all the taxi turns are stored. We are careless about that because it's not a fundamental part of the application.



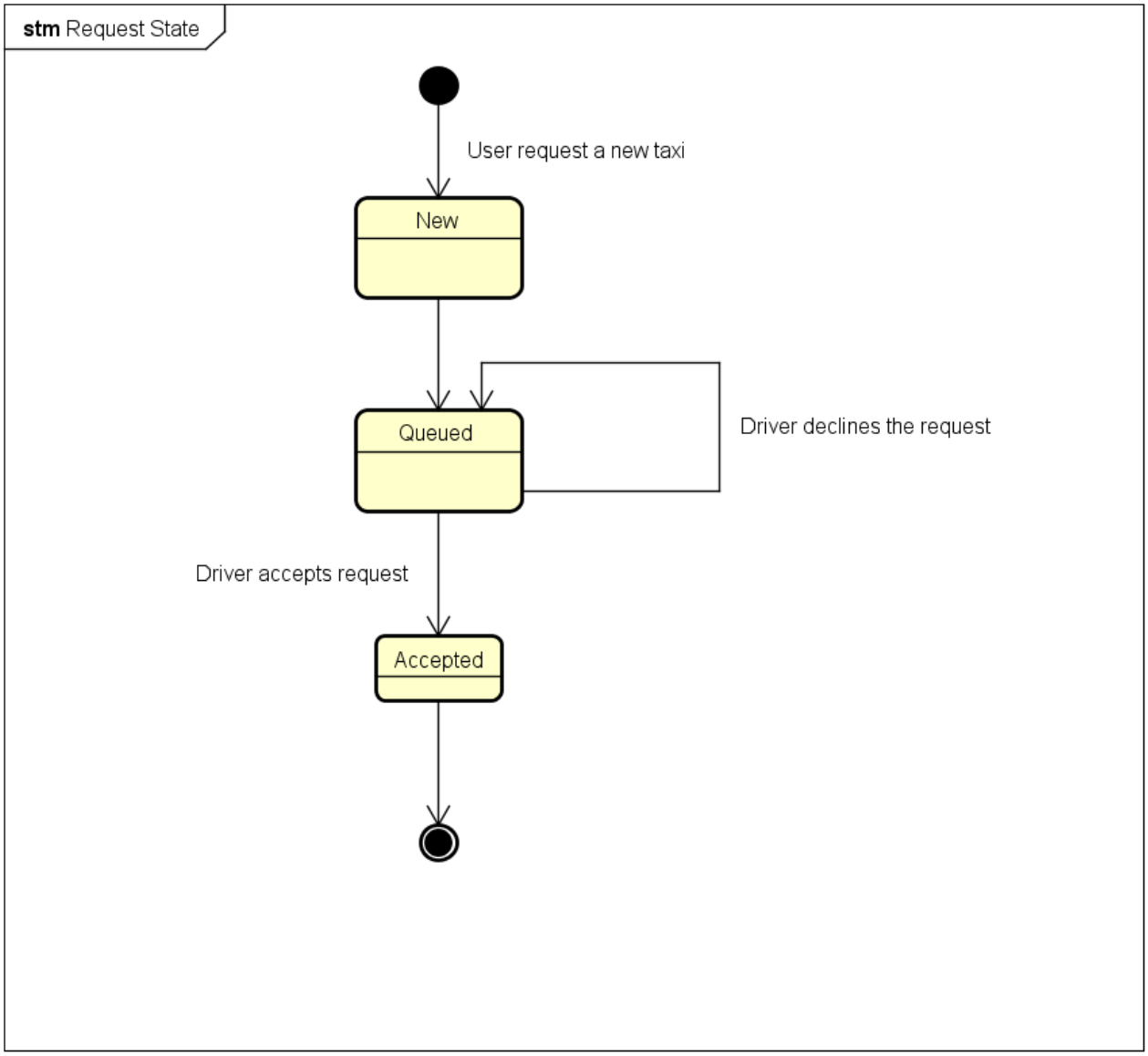
- **Change Availability**

We united two different use cases in one, because they are similar



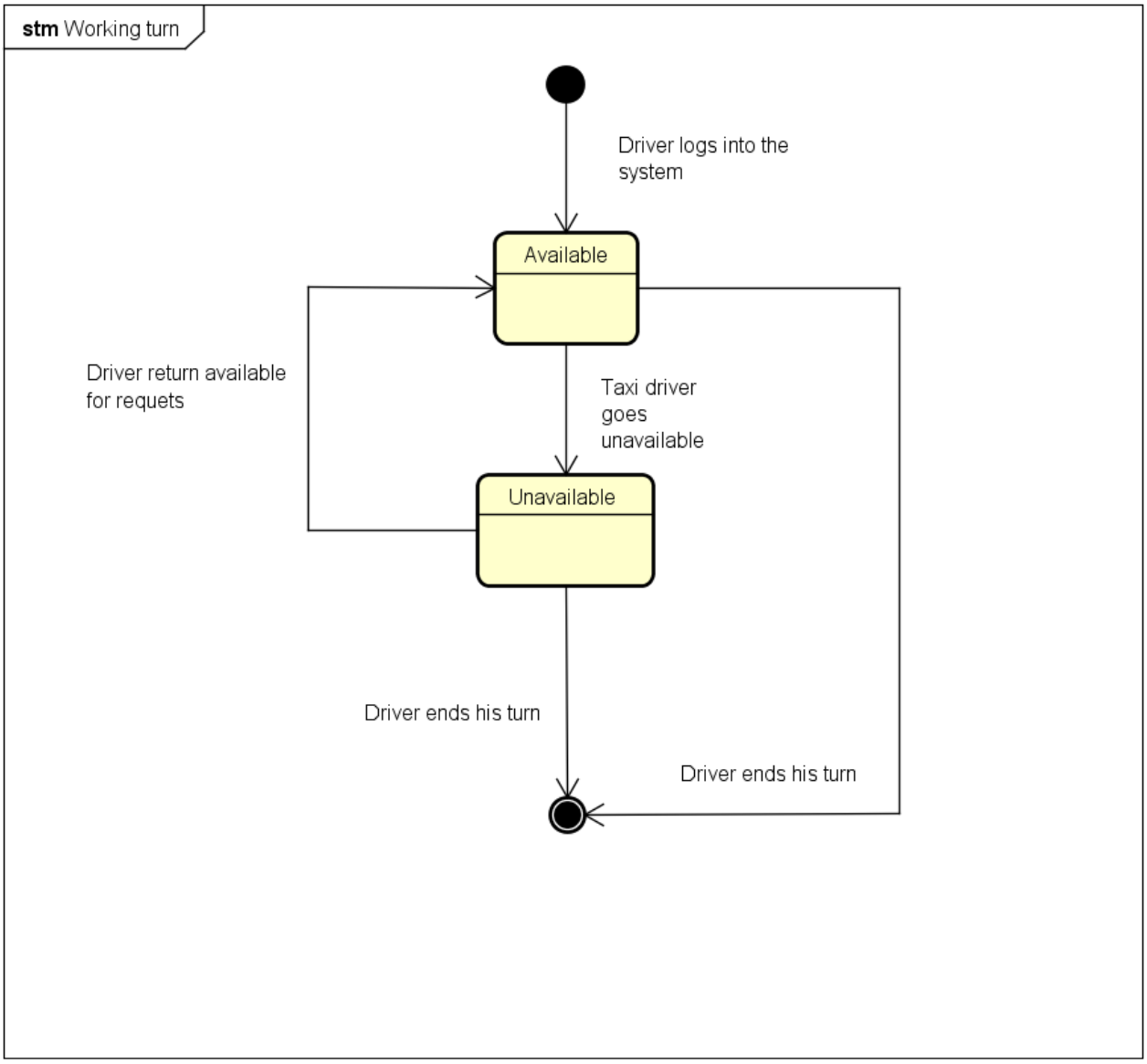
4.5 State Chart

- **Taxi request**



Notice that for the Reservation taxi there is the exactly same diagram, because when a call is made to the taxi, he can't distinguish whether it is a reservation or a request. In fact, ten minutes before the reservation time, the system behavior is the same.

- **Taxi Driver States**



5. ALLOY MODELLING

Now we are taking our class diagram and see if it is consistent using "Alloy Analyzer". We report below the code used and some World generated by our predicates to prove that our model is consistent.

```
// Signatures

// We create the Boolean with Integers: 1=true, 0=false
sig Bool{
     isTrue: Int
}{
     isTrue=1 || isTrue=0
}

sig Administrator {}

sig Zone{}

sig Position{
     //This Zone isn't in the Class Diagram,
     //but it will be calculated by the System
     PositionZone: one Zone
}

sig TaxiDriver{
     InsertedBy: one Administrator,
     AssignedTo: one Zone
}

sig Taxi{
     Driver: one TaxiDriver,
     Available: one Bool,
     TaxiPosition: one Position
}

sig WorkTurn{
     TaxiDriver: one TaxiDriver,
     TaxiDriven: one Taxi
}

sig TaxiRequest {
     StartPosition: one Position
}

sig Call{
     TaxiCalled: one Taxi,
     Request: one TaxiRequest ,
     Response: one Bool
}

sig Reservations{
     Request: lone TaxiRequest
}
```

```
// Facts

fact NoSameRequestsFromReservations{
  // Two Reservations can't make the same Request
  no x,y :Reservations | x!=y and x.Request=y.Request
}

fact OnlyOneTaxiForADriver{
  // A Taxi can be entrusted to only one Driver
  no x,y: Taxi | x!=y and x.Driver=y.Driver
}

// We never call the same taxi two times for the same request
fact NoTaxiCalledTwoTimesWithSameRequest {
   no disj x,y: Call | (x.TaxiCalled = y.TaxiCalled) and
                                          (x.Request=y.Request)
}

//Only One Taxi Can Accept a Call
fact OnlyOnePositiveResponse {
  no disj x,y: Call | ((x.Request=y.Request) and
                (x.Response.isTrue=1) and  (y.Response.isTrue=1))
}

//To Avoid Aving Many Bool Objects
fact OnlyTwoBools{
no disj x,y: Bool | x.isTrue=y.isTrue
}

//A Taxi Driver Must have at least one Turn Associated to be Available
fact OnlyLoggedTaxiCanBeAvailable{
     no x:Taxi | x.Available.isTrue=1 and
                           x.Driver not in (WorkTurn.TaxiDriver)
}

// There is always a taxi available
fact AtLeastOneAvailableTaxi {
     some t:Taxi | t.Available.isTrue=1
}
```

```
// Predicates

// The Others Are All Connected, so this will show a complete world
pred showEntireWorld(){
      #Call>0 and #WorkTurn>0 and #Reservations>0
}

run showEntireWorld for 2

// Here we are not focusing on administrators, and
// considering only one zone, for better readability
run showEntireWorld for 3 but exactly 1 Administrator, 1 Zone

pred show{}

run show for 3
```

5.1 Results

Those are the outputs of our analyzer for the three shows commands:

```
Executing "Run showEntireWorld for 2"
  Solver=sat4j Bitwidth=4 MaxSeq=2 SkolemDepth=1 Symmetry=20
  1093 vars. 104 primary vars. 1883 clauses. 12ms.
  Instance found. Predicate is consistent. 9ms.
```

```
Executing "Run showEntireWorld for 3 but exactly 1 Administrator, 1 Zone"
  Solver=sat4j Bitwidth=4 MaxSeq=3 SkolemDepth=1 Symmetry=20
  2371 vars. 172 primary vars. 4281 clauses. 17ms.
  Instance found. Predicate is consistent. 14ms.
```

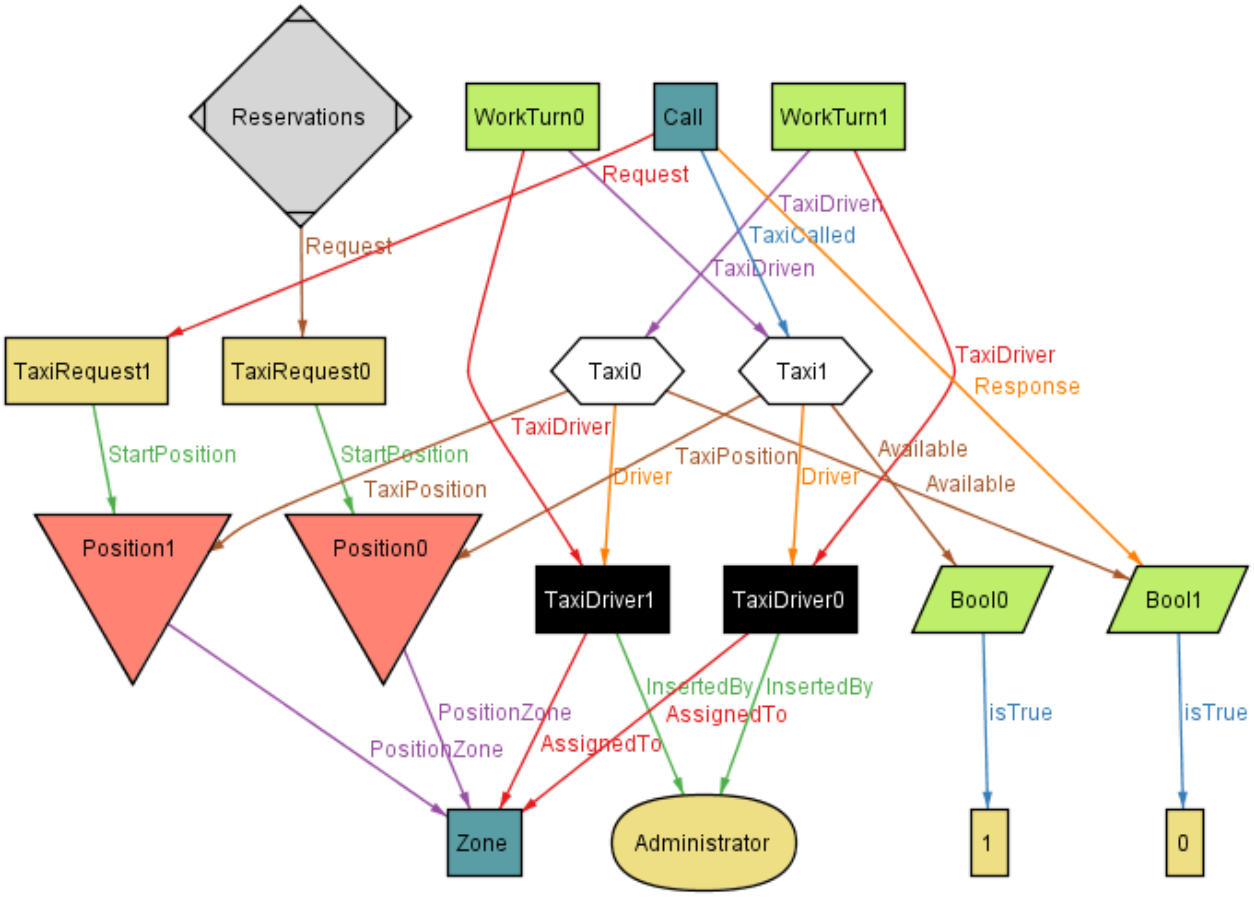
```
Executing "Run show for 3"
  Solver=sat4j Bitwidth=4 MaxSeq=3 SkolemDepth=1 Symmetry=20
  2414 vars. 195 primary vars. 4384 clauses. 13ms.
  Instance found. Predicate is consistent. 16ms.
```

6. WORLDS GENERATED

The First Scheme is a complete overview of the System, focusing on the data stored in our Database. We can't isolate cases because we have constraints (for example we can't do "run 3 but 0 administrator, instead there won't be Drivers neither), but we will show a second scheme with some minor approximations but higher number.

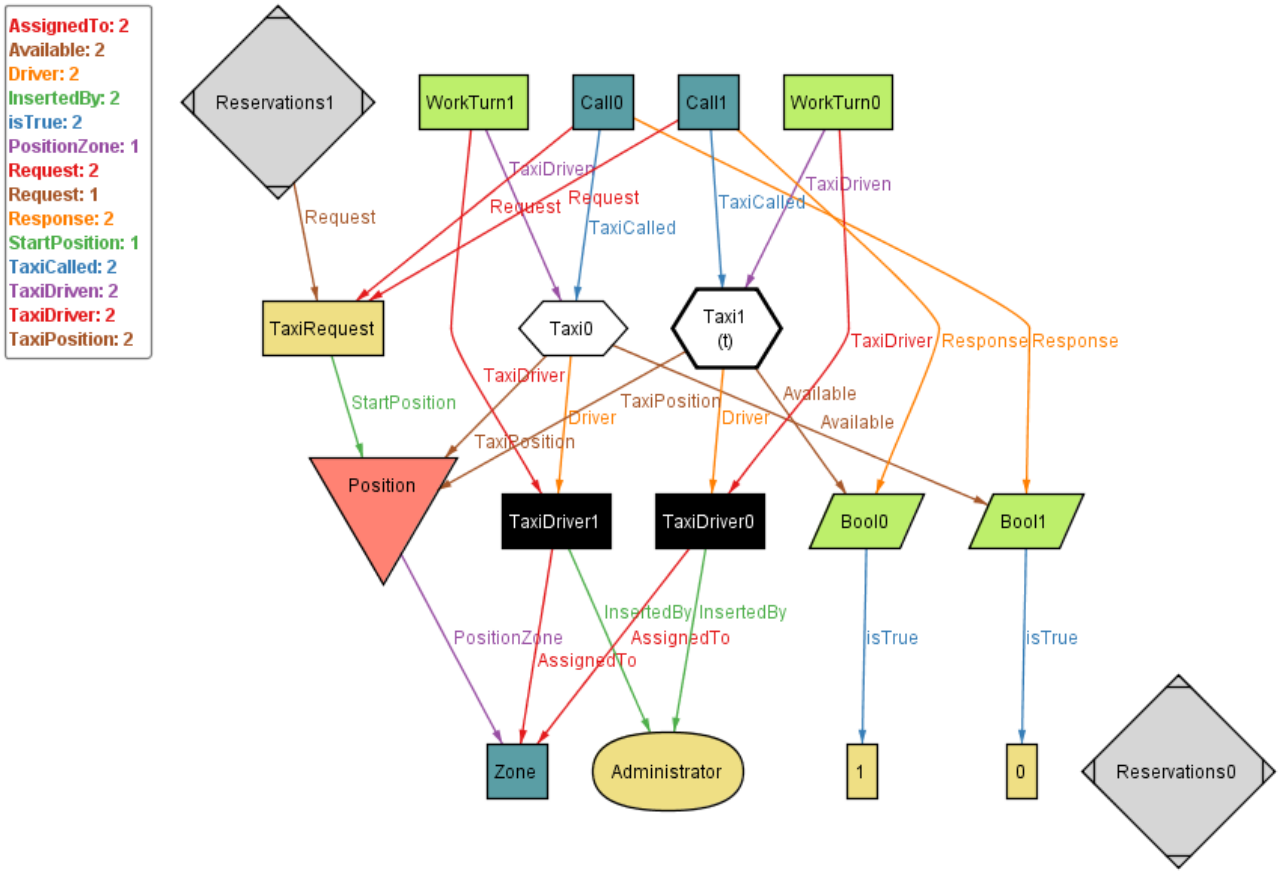
In this first scheme, we can notice that two Drivers have switched their Taxi: this is fine because we haven't any constraint about a Driver that makes the login onto another Taxi (But it's rare).

We also have a Taxi Request (0) that doesn't have any calls: this means that this request is pending, so, in this precise moment, the user is waiting. Also, the other request has a negative response, so it is pending too, but again it's perfectly fine, even if it's a rare situation.



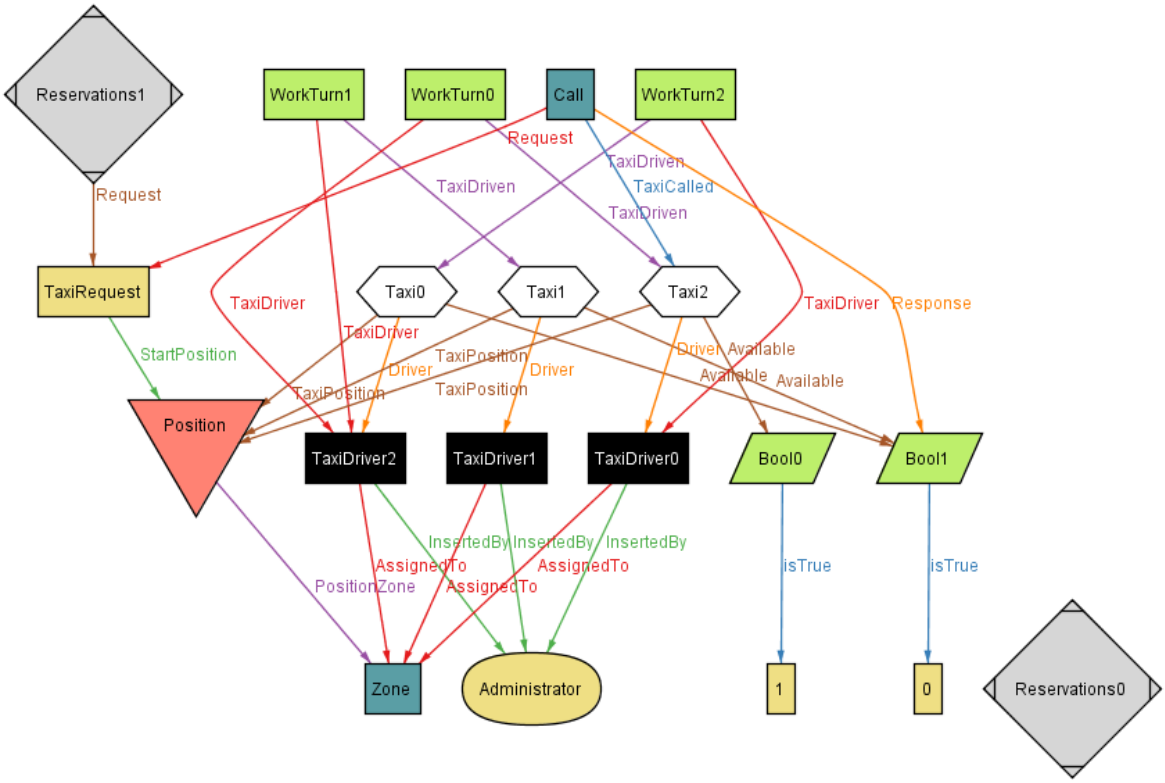
6.1 First Diagram: Complete Overview of Our World

In this other iteration we've made, there is a call made at a Taxi that isn't available, which is perfectly fine because, again, probably the call was made when he was available, and maybe he is already driving the user to destination, or it's an old call. For the same reason, a call for a Taxi here is independent from the zone in which he is in the moment we are considering..



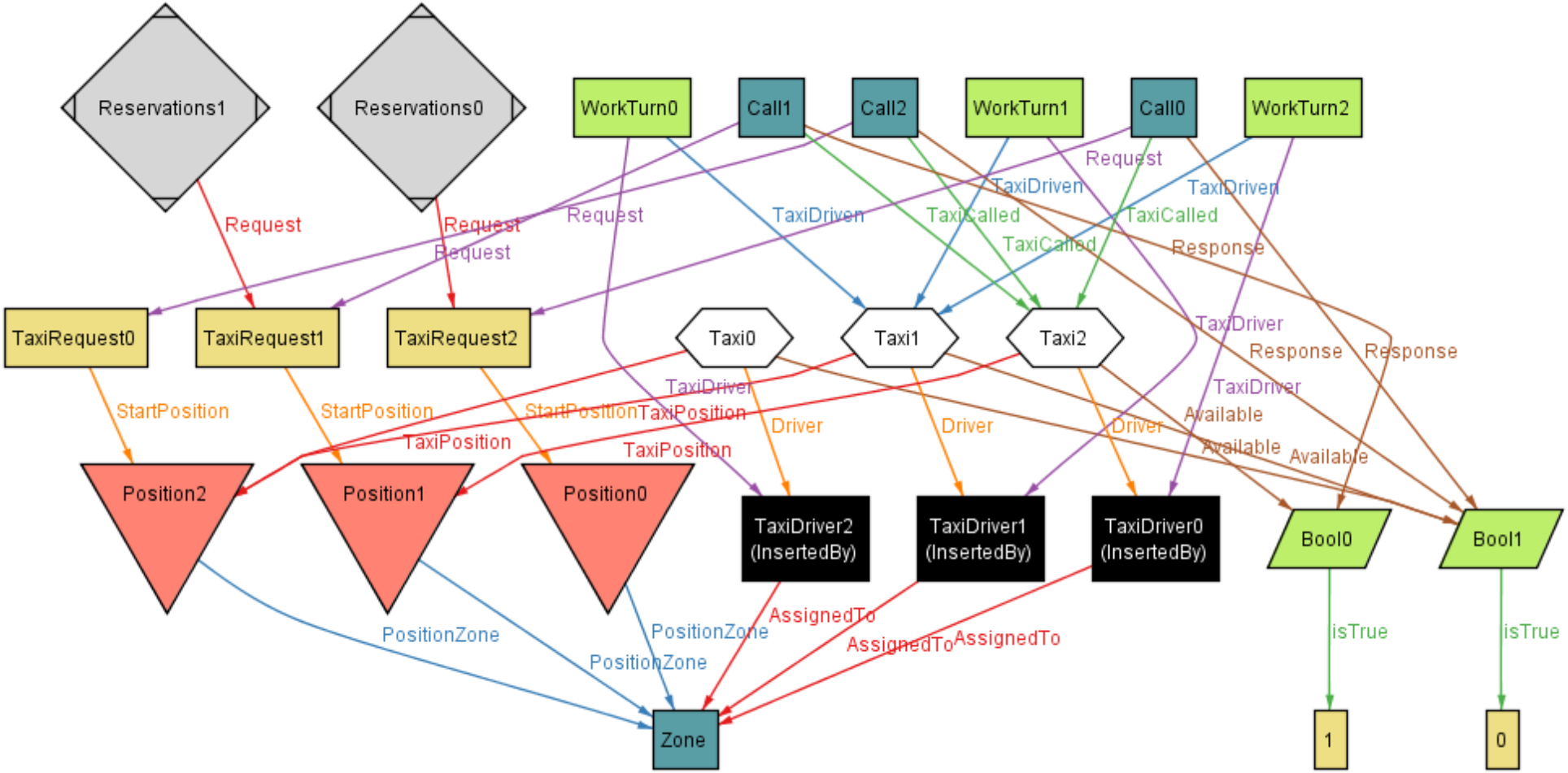
6.2 Complete Overview of Our World: Second Iteration

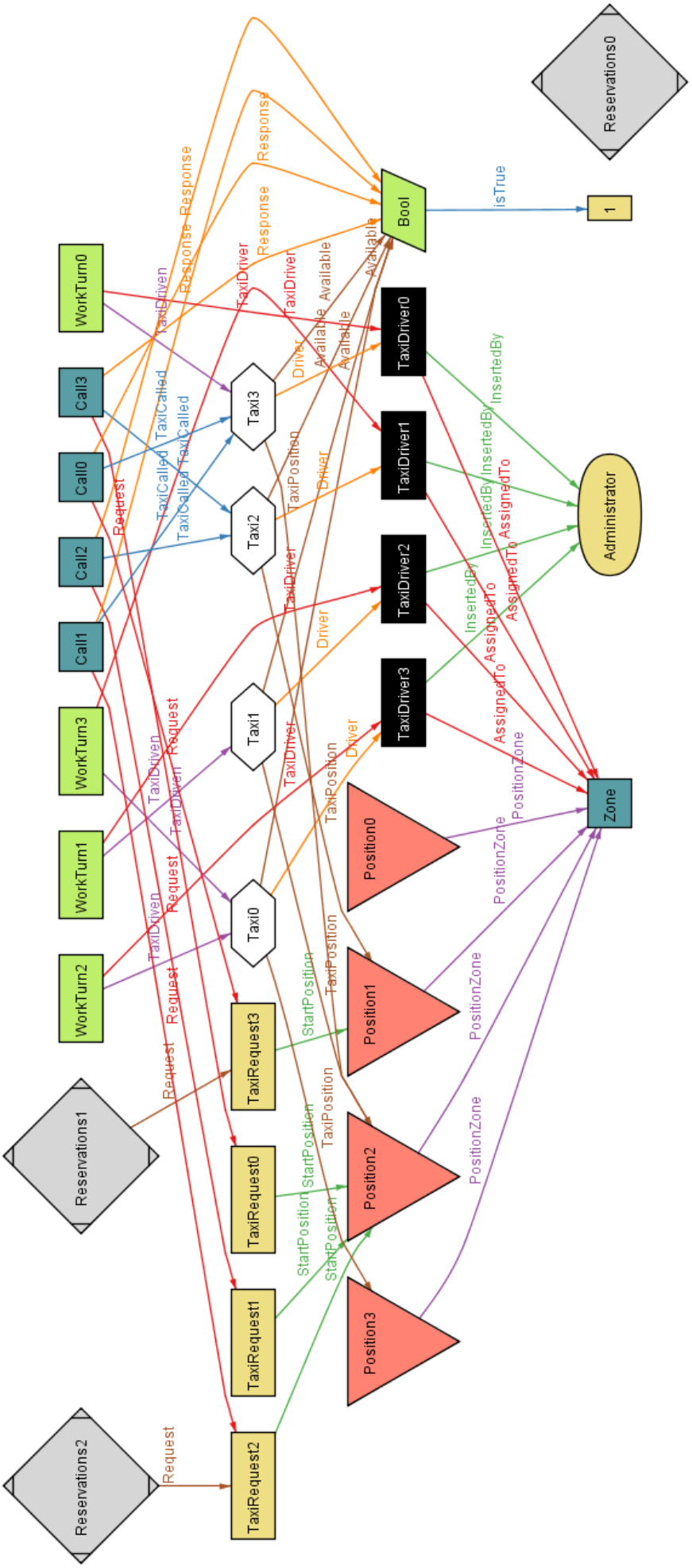
This is the Diagram considering only one Zone and one Administrator. Notice that There is a Pending Reservation, that is fine because it can be because the reservation time isn't arrived yet. Notice also that there are Taxies driven by more Taxi Driver: that's also fine because this scheme represent an history of the drives, so of course it means that a taxi has been driven by two drivers in two different times. We also did an observation for the Position: this is an improbably case where many Taxies are in the same position, but if we mean "near" it can be possible.



6.3 Diagram with only One Zone and One Administrator

We will show also some other, less readable, cases.


Reservations1
Reservations0
WorkTurn0
Call1
Call2
WorkTurn1
Call0
WorkTurn2
Request
Request
Request
Request
Request
TaxiDriven
TaxiDriven
TaxiDriven
TaxiCalled
TaxiCalled
TaxiCalled
Response
TaxiRequest0
TaxiRequest1
TaxiRequest2
Taxi0
Taxi1
Taxi2
TaxiDriver
Response
Response
TaxiDriver
TaxiDriver
StartPosition
StartPosition
StartPosition
TaxiPosition
TaxiPosition
TaxiPosition
Driver
Driver
Driver
Available
Available
Available
Position2
Position1
Position0
TaxiDriver2
(InsertedBy)
TaxiDriver1
(InsertedBy)
TaxiDriver0
(InsertedBy)
Bool0
Bool1
PositionZone
PositionZone
PositionZone
AssignedTo
AssignedTo
AssignedTo
isTrue
isTrue
Zone
1
0


Reservations0
1
isTrue
Bool
Response
Response
Response
Response
Available
Available
Available
Available
TaxiDriver
TaxiDriver
TaxiDriver
TaxiDriver
TaxiDriver0
TaxiDriver1
TaxiDriver2
TaxiDriver3
InsertedBy
InsertedBy
InsertedBy
InsertedBy
AssignedTo
AssignedTo
AssignedTo
AssignedTo
Administrator
Zone
PositionZone
PositionZone
PositionZone
PositionZone
Position0
Position1
Position2
Position3
StartPosition
StartPosition
StartPosition
StartPosition
TaxiPosition
TaxiPosition
TaxiPosition
TaxiPosition
Driver
Driver
Driver
Driver
TaxiRequest3
TaxiRequest0
TaxiRequest1
TaxiRequest2
Request
Request
Request
Request
Reservations1
Reservations2
Taxi3
Taxi2
Taxi1
Taxi0
TaxiCalled
TaxiCalled
TaxiCalled
TaxiCalled
TaxiDriven
TaxiDriven
TaxiDriven
TaxiDriven
Call3
Call0
Call2
Call1
WorkTurn0
WorkTurn3
WorkTurn1
WorkTurn2

7. USED TOOLS

- **Microsoft Office Word 2013** : to write this Document
- **Astah Professional**: to do all the Graphs, except the Alloy Words
- **Paint**: to draw the example interfaces
- **Alloy Analyzer 4.2**: to do the Alloy Analysis and World Graphs
- **Moqups:** online software to do the new Graphic interfaces previews