

Requirement Analysis and Specifications Document

Rizzi Matteo & Claudio Scandella

December 4, 2015

Contents

1	Introduction	5
1.1	Purpose	5
1.2	Scope	5
1.3	Actors	5
1.4	References	6
1.5	Goals	6
1.6	Acronyms and abbreviations	6
1.6.1	Acronyms	6
1.6.2	Abbreviations	7
1.7	Overview	7
2	Overall Description	8
2.1	Product perspective	8
2.2	User characteristics	8
2.3	Constraints	8
2.3.1	Parallel Operations	8
2.3.2	Reliability Requirements	8
2.3.3	Safety and Security Considerations	8
2.4	Assumptions and Dependencies	8
3	Specific Requirements	10
3.1	Functional Requirements	10
3.1.1	[G1] Allow the registration of users to the service	10
3.1.2	[G2] Allow users and taxi drivers to login into the service	10
3.1.3	[G3] Users can request a taxi for a run either via web or mobile application	10
3.1.4	[G4] Users can reserve a taxi for a ride at least 2 hours before;	10
3.1.5	[G5] Users can cancel a reservation up to 10 minutes before the meeting time	11
3.1.6	[G6] Taxi drivers can inform the system about their availability and whenever a run of a client with reservation or request ends	11
3.1.7	[G7] Taxi drivers can either confirm or reject a certain incoming call	11
3.1.8	[G8] Taxi drivers can report if a client with reservation or request isn't at the meeting point at the meeting time	11
3.1.9	[G9] Allow taxi drivers to see the distribution of other taxis in the city in order to spread themselves uniformly	12
3.1.10	[G10] Inform taxi drivers in the zone where a call is coming from about the request	12
3.1.11	[G11] Users have the chance to move back or forward the meeting time of a reservation	12
3.2	External Interface Requirements	13

3.2.1	User Interfaces	13
3.2.2	API Interfaces	25
3.2.3	Hardware Interfaces	25
3.2.4	Software Interfaces	26
3.3	Scenarios	26
3.3.1	Matteo registers to myTaxiService	26
3.3.2	Jack needs a taxi to arrive at the San Siro Stadium	26
3.3.3	Claudio reserves a taxi for a job interview	26
3.3.4	Giorgio must go to the bank but the taxi crashes before the meeting	27
3.3.5	Riccardo desires to modify his reservation	27
3.3.6	Mario desires to delete a reservation	27
3.3.7	The taxi driver Alberto receives a call	28
3.3.8	Daniele has to report a client that did not show up at the meeting	28
4	UML Models	29
4.1	Use Case	29
4.1.1	Registration	30
4.1.2	Login	31
4.1.3	Request a taxi	31
4.1.4	Reserve a taxi	32
4.1.5	Modify reservation	33
4.1.6	Cancel reservation	34
4.1.7	Taxi Driver inform about availability	34
4.1.8	Accept/Reject calls	35
4.1.9	Report a client	35
4.1.10	Declare end of the ride for a client	36
4.2	Class Diagram	37
4.3	Sequence Diagram	38
4.3.1	Registration	38
4.3.2	User's login	39
4.3.3	Request a taxi	40
4.3.4	Reserve a taxi	41
4.3.5	Modify a reservation	42
4.3.6	Delete a reservation	43
5	Appendix	44
5.1	Alloy	44
5.1.1	Signatures	44
5.1.2	Facts	45
5.1.3	Assertions	45
5.1.4	Predicates	46
5.1.5	Result	46
5.1.6	Generated Worlds	47
5.2	Softwares and tools used	52

5.3	List of differences with previous versions	52
5.4	Hours of work	52

1 Introduction

1.1 Purpose

This document is the Requirement Analysis and Specification Document (RASD) for the myTaxiService application. The goal of this text is to describe the functional and/or non-functional requirements, helping the developers to implement properly all the functionalities of the new system.

1.2 Scope

We will project and implement myTaxiService, which is an application to improve the taxi service of a large city. The service aims to simplify the user's access to the service and to guarantee a fair management of the taxi queues. The system will offer the possibility to register a new user with all his credentials (name, age, address, email, ecc.). Once he is registered he can request a taxi for a ride either via web application or a mobile app. The user can also reserve a taxi specifying the origin and the destination at least two hours before the run. Taxi drivers inform the system about their availability and acknowledge incoming calls from potential passengers through a mobile application. Besides the specific user interfaces for passengers and taxi drivers, the system offers also programmatic interfaces to enable the development of additional services (e.g., taxi sharing) on top of the basic one.

1.3 Actors

- Guest: these actors can only see the login and registration pages. The system allows guests to sign up/login and nothing else.
- User: they are registered and logged in guests. They are able to use all the functions the application is designed for. They have a personal profile in which their informations are stored in, such as:
 1. Name;
 2. Surname;
 3. Email;
 4. Username;
- Taxi Driver (Special User): they are users who have been given a username and a password from the government of the city. The functions they can use are different from those of a normal User, even if they also possess a personal profile with all the informations the user has.

1.4 References

- Specification Document: Software Engineering 2 Project, AA 2015-2016 Assignments 1 and 2;
- IEEE Standard for Requirement Specification.

1.5 Goals

MyTaxiService has to satisfy the following list of goals:

- [G1] Allow the registration of users to the service;
- [G2] Allow users and taxi drivers to login into the service;
- [G3] Users can request a taxi for a run either via web or mobile application;
- [G4] Users can reserve a taxi for a ride at least 2 hours before;
- [G5] Users can cancel a reservation up to 10 minutes before the meeting time;
- [G6] Taxi drivers can inform the system about their availability and whenever a run of a client with reservation or request ends;
- [G7] Taxi drivers can either confirm or reject a certain incoming call;
- [G8] Taxi drivers can report if a client with reservation or request isn't at the meeting point at the meeting time;
- [G9] Allow taxi drivers to see the distribution of other taxis in the city in order to spread themselves uniformly;
- [G10] Inform taxi drivers in the zone where a call is coming from about the request;
- [G11] Users have the chance to move back or forward the meeting time of a reservation;

1.6 Acronyms and abbreviations

1.6.1 Acronyms

- RASD: Requirements Analysis and Specification Document;
- API: Application Programming Interface.
- UML: Unified Modeling Language.
- GPS: Global Positioning System.
- OS: Operating System.
- DB: DataBase.

1.6.2 Abbreviations

- [Gn]: n-goal.
- [Rn]: n-functional requirement.
- [Dn]: n-domain assumption.

1.7 Overview

The RASD is divided in 5 sections:

- Section 1: Introduction, it gives a description of document and some basic information about the software.
- Section 2: Overall Description, gives general information about the software product with more focus about constraints and assumptions.
- Section 3: Specific Requirements. This part focuses on functional requirements, external interfaces requirements and scenarios
- Section 4: UML Models. This part focuses on use cases, sequence and class diagrams.
- Section 5: Appendix. this part contains some information about the attached.als file and some described screenshot of software used to generate it. It also contains informations about how we created this document.

2 Overall Description

2.1 Product perspective

The service is not integrated with any other existing system and it is intended only for taxi drivers and for users. Furthermore, the application will provide modules or interfaces for the integration with potential new future services (e.g. taxi sharing).

2.2 User characteristics

Both users and taxi drivers should know basic functions of an application, either via browser or via mobile. In both cases, an Internet connection is required.

2.3 Constraints

2.3.1 Parallel Operations

The system must guarantee parallel operations from different users without interfering with the correct working of the application.

2.3.2 Reliability Requirements

Once the user has received the notification that his operation was successful, the system will guarantee that a taxi is reserved for him.

2.3.3 Safety and Security Considerations

The application must comply with the current privacy and security regulations.

2.4 Assumptions and Dependencies

- Users can request a taxi only if they have no active requests at the same time;
- Users can reserve a taxi, even if they have other reservations on that day (up to 3 at the same time), with at least ten minutes of distance between the two calls;
- After 3 reports within 30 days the user will be suspended for a month;
- There are at least 10 minutes between two requests;
- A reservation lasts until an hour after the meeting time;
- A taxi driver whose call is canceled will be moved in the first position of the proper queue;
- Taxi positions in the city are updated every minute;

- There is always an available taxi for each request or reservation;
- Taxi drivers are available only if the GPS tracks them inside the city;
- Taxis can reach every place in their zone, in less of five minutes;
- Taxi drivers always accept or refuse a request in little time.

3 Specific Requirements

3.1 Functional Requirements

3.1.1 [G1] Allow the registration of users to the service

- [R1] Visitors must not be already registered to the application to perform registration success;
- [R2] Visitors must choose an username that is not yet been selected by another user;
- [R4] The visitor must include all his credentials to perform the registration and also he has to accept the terms of the privacy's agreement.

3.1.2 [G2] Allow users and taxi drivers to login into the service

- [R1] The user must be registered to the application;
- [R2] The user must insert username and password correctly during the login phase;
- [R3] Wrong credentials don't give any access to the user;
- [R4] Visitors can't make any request before they log in;

3.1.3 [G3] Users can request a taxi for a run either via web or mobile application

- [R1] Users must be logged in to the application;
- [R2] Users must complete the request form in which they need to insert the address of their location;
- [R3] Users must complete any mandatory field of the form;
- [R4] The address inserted must be correct;

3.1.4 [G4] Users can reserve a taxi for a ride at least 2 hours before;

- [R1] Users must be logged in into the application;
- [R2] Users must complete every mandatory field of the form that he has to compile for the reservation, including address and dating time;
- [R3] The address must be correct;
- [R4] The dating must be at least two hours after the time the user compiles the form;
- [D1] The dating time must be between 00.00 and 23.59.

3.1.5 [G5] Users can cancel a reservation up to 10 minutes before the meeting time

- [R1] Users must be logged in to the application
- [R2] The user must have made a reservation through the application;
- [R3] Users can cancel their reservation up to 10 minutes before the meeting time using the delete (red) button;
- [R4] The delete of a reservation is irreversible. There is no way to retrieve the data after deleting the order;
- [R5] Users must confirm the order cancelation by pushing the yes button in the pop-up shown after they push the delete button ;

3.1.6 [G6] Taxi drivers can inform the system about their availability and whenever a run of a client with reservation or request ends

- [R1] Taxi drivers must be logged in to their application;
- [R2] Taxi drivers must click the "Available Button" on the Taxi driver's application;
- [R3] The Taxi driver must have informed the system his client has ended his ride;

3.1.7 [G7] Taxi drivers can either confirm or reject a certain incoming call

- [R1] Taxi drivers must be logged in to the application;
- [R2] Taxi drivers must have received a call on their devices;
- [R3] Taxi drivers can accept the call by clicking the accept button on their devices;
- [R4] Taxi drivers can reject the call by clicking the reject button on their devices;
- [D1] Taxi drivers receive calls only if they are available;

3.1.8 [G8] Taxi drivers can report if a client with reservation or request isn't at the meeting point at the meeting time

- [R1] Taxi drivers must be logged in to the application;
- [R2] The GPS of the taxi driver confirms he is at the meeting's location;
- [R4] Taxi drivers should report to the system the client has not arrived using the form available in the application installed on their devices;

- [R5] All the form's fields must be compiled(if they are mandatory)
- [D1] The client's username (that is a field of the form) must be correct.

3.1.9 [G9] Allow taxi drivers to see the distribution of other taxis in the city in order to spread themselves uniformly

- [R1] Taxi drivers must be logged in to the application;
- [R2] Every taxi's position must be registered thanks to the GPS on their car;
- [R3] Only taxi codes are shown to other taxi drivers, not their exact position;
- [R4] Only available taxis are shown on other taxi drivers devices;
- [R5] Taxi drivers can see only taxis in their zone;

3.1.10 [G10] Inform taxi drivers in the zone where a call is coming from about the request

- [R1] Taxi drivers must be logged in;
- [R2] Taxi drivers must be in the zone where the call is coming from;
- [R3] The information arrives only to those Taxi drivers that are available;

3.1.11 [G11] Users have the chance to move back or forward the meeting time of a reservation

- [R1] Users must be logged in to the application;
- [R2] Users made a reservation that is not expired yet;
- [R3] Users can change the meeting time by clicking the modify button on the reservation and by changing the fields available in the form;
- [D1] The new meeting time must be between 00.00 and 23.59 and on the same day;
- [R4] Users must confirm the changes by clicking the confirm button at the end of the form;
- [R5] The new meeting time must be at least 2 hours after the time the change was confirmed;
- [R6] Users receives a notification after they have changed the meeting time;

3.2 External Interface Requirements

3.2.1 User Interfaces

Our interface is thought to be used via Web or via Application. In both cases, the user interface has the same design. We show in this document some mock-ups of the user's interface.

MyTaxiService

[Login](#)
[Sign up](#)

Name

Surname

E-mail

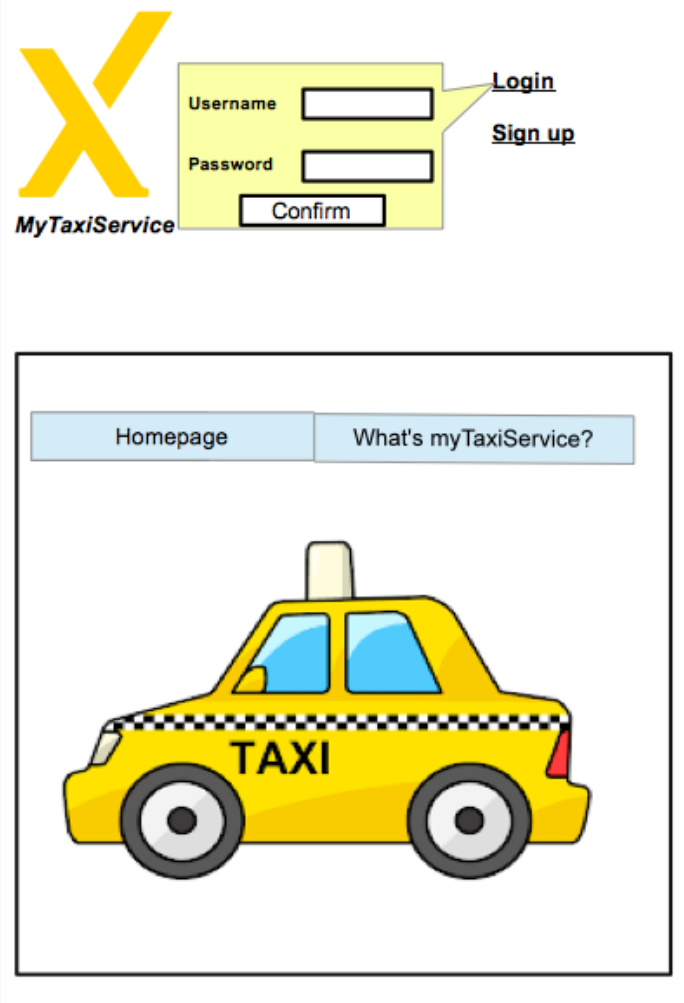
Username

Password

Confirm Password

☐ Accept the terms and the privacy agreement

Registration Page After the guest has clicked on the sign up button, The system will show him the "Registration Form", which he must compile in all of its fields. When the guest has filled all the fields of the form he will click on the confirm button, in order to send his data to the system. At the end, to verify his account he will also respond to an e-mail, sent by the system, in order to close this procedure.



Login If the guest is already a user he can click on the login button. Then the system will show him a pop-up in which he has to insert his Username and his Password(and click the confirm button), in order to enter inside his personal profile page.



MyTaxiService

USERNAME

Logout

PROFILE

REQUESTS

RESERVATIONS

Name MYNAME

Surname MYSURNAME

E-mail MYEMAIL@EMAIL.COM

Username USERNAME

Show User's Profile After the user has logged in the system will redirect him to his personal profile page, in which are available all the data he entered during the "sign up" phase, such as Name, Surname, E-mail address and Username.



USERNAME

Logout

PROFILE

REQUESTS

RESERVATIONS

Complete your request. Send us
your position and
a taxi will arrive as soon as possible

Confirm

User asks for a request Once The user has clicked on the "Requests" key then the system will redirect him to the Request form, in which the user must enter his position in order to get a taxi. After he inserted the address he clicks the Confirm Button and then he waits for the notification of the taxi's code and the meeting time.



USERNAME

Logout




PROFILE

REQUESTS

RESERVATIONS

Those are your current reservations

Date	Address	Destination	Operation
13/11/15 20.00	My Street	2 nd Avenue	 
14/11/15 05.00	My Street	Hospital	 
			

-  Create a new reservation
-  Modify a reservation
-  Delete a reservation

Reservation's Homepage Once the user has clicked the "Reservations" key the system will redirect him to the Reservations homepage in which he looks at all his active bookings. Then he has the chance to create(Green) a new reservation, or to modify/delete(Yellow/Red) a previous one(still active yet).



USERNAME

Logout

PROFILE

REQUESTS

RESERVATIONS

Complete your reservation. We only need those datas. Remember the meeting time must be at least two hours after this moment

Address

Destination

Meeting time

Date

Confirm

User asks for a booking Once the user has clicked the green button he will be redirected to the Reservation form in which he must insert address, destination, meeting time and date. Then he will click the confirm button in order to complete his booking.



USERNAME

Logout

PROFILE	REQUESTS	RESERVATIONS
---------	----------	--------------

Modify your reservation. Remember, you can only change the meeting time and the date



















Address	<input type="text"/>	Confirm
Destination	<input type="text"/>	
Meeting time	<input type="text"/>	
Date	<input type="text"/>	

User modifies a reservation Once the user clicks the yellow button the system redirects him to the form to modify only meeting time and date of the reservation. Then he pushes the confirm button in order to save the changes. Remember, once the changes are made, there is no way to regain previous data back.




USERNAME

Logout

PROFILE	REQUESTS	RESERVATIONS																
<div>Those are your current reservations</div> <table border="1"><thead><tr><th>Date</th><th>Address</th><th>Destination</th><th>Operation</th></tr></thead><tbody><tr><td>13/11/</td><td></td><td></td><td> </td></tr><tr><td>14/11/</td><td></td><td></td><td> </td></tr><tr><td></td><td></td><td></td><td></td></tr></tbody></table> <div> Create a new reservation  Modify a reservation  Delete a reservation</div>			Date	Address	Destination	Operation	13/11/			 	14/11/			 				
Date	Address	Destination	Operation															
13/11/			 															
14/11/			 															
																		

User deletes a booking Once the user clicks the red button, the system will show him a pop-up in which he's told if he's sure to delete this booking or not. The user must choose between Yes(green button) or No(Red button). If he chooses the first one the booking will be deleted, otherwise the system will redirect him to the Reservations Homepage.



1

USERNAME

Logout

MyTaxiService

REQUEST NOTIFICATION

Taxi number 1368 will arrive at your location at 10.35 p.m


NameMYNAME

SurnameMYSURNAME

E-mailMYEMAIL@EMAIL.COM

UsernameUSERNAME

User receives a notification of his call After the user made his request, he will be redirected to his Personal Profile Page, and in a couple of minutes(once the system found a taxi for him) he will receive a notification in which it's written the taxi's number and the arriving time of the taxi.



Issue

TAXIDRIVERUSER

Logout

PROFILE

CALLS ¹

AVAILABILITY

DISTRIBUTION

REPORT


USERNAME	ADDRESS	DESTINATION*	HOUR*
USERNAME	2ND AVENUE		

Accept

Decline

*Optional

Taxi driver's choice about an incoming call Taxi drivers that are available, are logged in to the application, and are tracked by the GPS in a certain zone will receive a user's request by the system(remember they will receive it one by one basing on the queue of that zone). Once a call arrives the system will redirect the taxi driver to the Call page on his device in which he has to click the "Accept" or the "Decline" button in order to acknowledge the system if they are going to take care of the request.



Issue

[TAXIDRIVERUSER](#)
[Logout](#)


PROFILECALLSAVAILABILITY

DISTRIBUTIONREPORT

If you have ended your ride please remember to click the "Available Button", in order to send you another call as soon as possible

AVAILABLE

Taxi Driver's availability Once a taxi driver(who is logged in to the application through his device) has ended his ride he MUST click the Availability key and , once the system has redirected him to the availability page, click on the available button, in order to have the chance to take care of further calls.



Issue

[TAXIDRIVERUSER](#)
[Logout](#)

PROFILECALLSAVAILABILITY

DISTRIBUTIONREPORT

This is the report area. If you had any issue with a Client, you're welcomed to compile this form

Username

Ride ID

Taxi Code

Comments*

*Optional

Confirm

Taxi driver report A taxi driver can click on the report key to report an issue with a client. The system will redirect him to the report form in which he must enter all the mandatory fields and ,if he wants, also the optional one, and then push the confirm button.

X
MyTaxiService

Issue [TAXIDRIVERUSER](#) [Logout](#)

PROFILE CALLS AVAILABILITY

DISTRIBUTION REPORT

If you have ended your ride click the End of the Ride Button. If you are now available, also remember to Click the Available button in the availability page

END OF THE RIDE

Taxi driver declares the end of the ride Once the taxi driver has carried the client to his destination, he must enter the “Calls” page and then clicks the end of the ride button to acknowledge the system his ride is ended.

3.2.2 API Interfaces

The system will use Location Services API supplied by Google.

3.2.3 Hardware Interfaces

Neither the mobile application nor the web site has any designated hardware, therefore they don't have any direct hardware interfaces. The physical GPS is managed by the GPS application in the mobile phone and the hardware connection to the database server is managed by the OS.

3.2.4 Software Interfaces

The mobile application communicates with the GPS application in order to get geographical information about where the taxi drivers are located, and with the database in order to get the information about the type of actor.

3.3 Scenarios

3.3.1 Matteo registers to myTaxiService

Matteo must arrive at the exam in time because it is his last exam before the degree. Today there is a TreNord strike so no trains are available. A friend tells him about the myTaxiService application. So he decides to download it on his mobile device. Once the application is downloaded he reads how the application works in the “What’s myTaxiService” page, then he sign up to the service. In order to do that he completes the registration form in which he enters name, surname, e-mail address, username and password and he accepts the privacy terms, then he clicks on the confirm button. At last, he opens his e-mail in order to verify his account and so using the application to get a cab.

3.3.2 Jack needs a taxi to arrive at the San Siro Stadium

Jack is an A.C Milan player and he will meet other players at Milanello where a bus will take them to the stadium. But, after lunch, he sits on his couch and he falls asleep. When he wakes up, the bus, that will take other players from Milanello to San Siro is already gone. So, he informs Mr. Mihajlovic he will arrive in taxi at the stadium. So, he opens the myTaxiService app on his mobile, he logs in entering his credentials (username and password); then he clicks on the “Request” key and the system shows him the Request form. He puts his address in the form and clicks the confirm button. In a matter of seconds the system sends him a notification with the taxi code and the waiting time. In a few minutes the taxi is at Jack’s house and he arrives at the stadium where he joins his mates.

3.3.3 Claudio reserves a taxi for a job interview

Claudio this morning has a job interview and he needs a taxi to reach the headquarters of the company that wants to sign him. He knows the meeting is at 10.30 a.m and it will last 2/3 hours because he is not the only candidate. So as soon as he leaves his house he log in to the myTaxiService application and he asks for a taxi entering his house’s address in the Request form. While waiting for the taxi he also decides to reserve a taxi for the return trip. So he clicks on the “Reservations” key and he clicks on the green button. Then the system redirects him to the reservation form in which he enters the meeting’s address, destination, meeting time and the date. Once his job interview is ended, few minutes before the time agreed, he receives a notification with the taxi code and the waiting time.

3.3.4 Giorgio must go to the bank but the taxi crashes before the meeting

Giorgio must go to the bank to withdraw money, but his bank, which is the only one he trusts, it's too far from his house. So, he opens the myTaxiService app and he makes the login. He makes a request for a taxi entering his house's address and clicking the confirm button in the form in the Requests page. He receives the notification that his taxi(code:1368) will arrive at 10.30 a.m at his house. But during the ride the taxi driver crashes into a tree. Luckily the taxi driver is ok. The taxi driver, so, signals to the application that his taxi is not available anymore by clicking the "Issue" button on his application. The system, so, assigns a new taxi to the client and he notifies the client the new taxi code(1638) and the new meeting time(10.35 a.m).

3.3.5 Riccardo desires to modify his reservation

Riccardo has reserved a taxi for himself because he has a date with Rebecca. The date is scheduled for 9 pm, but around 7 pm she tells Riccardo that she will arrive at 9.30 pm because she will finish late work. So he logs in the myTaxiService app and he clicks on the "Reservations" key. Then he clicks on the yellow button; so the system redirects him to the Reservation form in which he can change meeting time and date. Luckily, Rebecca informed him in time because the myTaxiService accepts reservations up to two hours after the confirm button was clicked. He enters the new meeting time in the form and he clicks the "confirm" button. Then, few minutes before the time agreed, he receives a notification with the taxi code and the waiting time.

3.3.6 Mario desires to delete a reservation

Mario has reserved a taxi for himself because he has to take two cars to the mechanic to change tires. But today there is no one at home except him so he needed a taxi that could bring him from the workshop to his home to take the second car. But his son Matteo discovers in the morning that the "Databases 2" lesson has been deleted. So, he returns home early and he tells his father he can help him to take one of the cars to the workshop. Then, Mario logs in to the myTaxiService app and he clicks the "Reservations" key. Once in the reservations page he clicks on the red button for his booking and the system shows him a pop-up in which he needs to click between yes or no to decide if he is sure to delete the reservation. He clicks yes and so his reservations is deleted.

3.3.7 The taxi driver Alberto receives a call

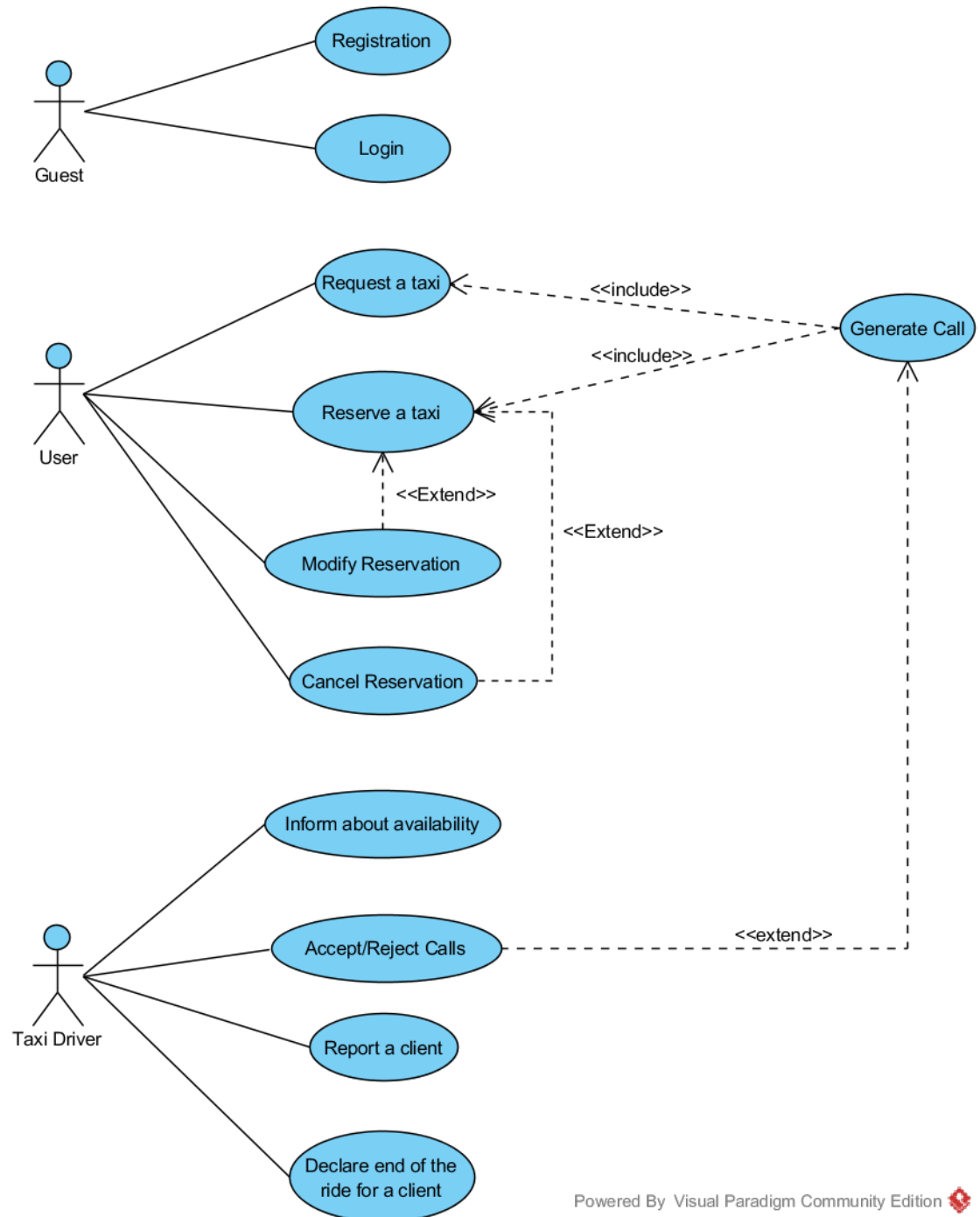
Alberto is a taxi driver and he's already logged in to the app. The system signals him a certain call from a client is coming. Alberto has to decide if he accepts or rejects this call. He decides to accept the call by clicking the green accept button on the "Calls" page. The client receives from the system a notification that Alberto's taxi is coming in 15 minutes. Alberto arrives at the location where the client is and he drives to the destination the user desires. Once they arrive Alberto signals to the system that the client has left his taxi by clicking the "Exit Button" on his device on the Homepage. Then, he goes to the Availability page by clicking on its key and he pushes the available button. The system then puts his taxi in the last position of the queue of the zone where Alberto is.

3.3.8 Daniele has to report a client that did not show up at the meeting

Daniele is a taxi driver and he has just accepted a call coming from a client (He already logged in). He drives to the client's location but when he arrives the client isn't there. He waits for a couple of minutes, then he signals to the system the client has not showed up at the meeting time by clicking on the Report key. So, Daniele compiles the form for the report that the client hasn't showed up entering the taxi code, the username of the client and why is he reporting the client and then he clicks the confirm button. Then, a report point is added to the user's profile. Luckily, this is not its third, because in that case he would have been suspended from the service. The taxi driver then informs the system of his availability and waits for another call.

4 UML Models

4.1 Use Case



4.1.1 Registration

Actors	Guest
Goals	[G1]
Input Conditions	NULL
Output Conditions	<ol style="list-style-type: none">1. With his device the guest opens the myTaxiService app2. clicks the sign up button3. fills in all fields of the form and clicks confirm button4. application sends the confirmation email5. guest verifies his account by clicking on the link sent by e-mail6. application save the data in the DB
Exceptions	<ol style="list-style-type: none">1. visitor is already a user2. one or more mandatory fields are not valid3. username chosen already used by another user4. e-mail chosen already associated to another user

4.1.2 Login

Actors	Guest, User, Taxi Driver
Goals	[G2]
Input Conditions	User must be registered to the service
Output Conditions	<ol style="list-style-type: none">1. With his device the guest opens the myTaxiService app2. clicks the log in button3. The system shows a pop-up in which the guest must insert username and password4. The guest fills the fields and clicks confirm button5. System redirects the user to his personal profile page(Taxi driver or User)
Exceptions	<ol style="list-style-type: none">1. Username not valid2. Password not valid

4.1.3 Request a taxi

Actors	User
Goals	[G3]
Input Conditions	User must be logged in to the application
Output Conditions	<ol style="list-style-type: none">1. User clicks the Request key2. The system shows the Request form3. Users fill in the address field of the form and click confirm button4. System search for first taxi available5. User receives a notification
Exceptions	<ol style="list-style-type: none">1. Address not valid

4.1.4 Reserve a taxi

Actors	User
Goals	[G4]
Input Conditions	<ol style="list-style-type: none">1. User must be logged in to the application2. This must happen at least two hours before the meeting
Output Conditions	<ol style="list-style-type: none">1. User clicks the Request key2. The system shows Reservations HomePage3. User clicks green button to create a new reservation4. The system shows the Reservations form5. Users fill in all the fields of the form and click confirm button
Exceptions	<ol style="list-style-type: none">1. Address not valid2. Destination not valid3. Meeting time not valid4. Date not valid

4.1.5 Modify reservation

Actors	User
Goals	[G11]
Input Conditions	<ol style="list-style-type: none">1. User must be logged in to the application2. This must happen at least two hours before the new meeting time3. User must have reserved a taxi
Output Conditions	<ol style="list-style-type: none">1. User clicks the Request key2. The system shows Reservations HomePage3. User clicks yellow button on his reservation to change it4. The system shows the Reservations form5. Users fill in all the fields available(Meeting time and Date) of the form and click confirm button
Exceptions	<ol style="list-style-type: none">1. Meeting time not valid2. Date not valid

4.1.6 Cancel reservation

Actors	User
Goals	[G5]
Input Conditions	<ol style="list-style-type: none">1. User must be logged in to the application2. This must happen at least ten minutes before the meeting time3. User must have reserved a taxi
Output Conditions	<ol style="list-style-type: none">1. User clicks the Request key2. The system shows Reservations HomePage3. User clicks red button on his reservation to delete it4. The system shows a pop-up to have the confirmation of the delete5. Users clicks the yes button to delete the reservation
Exceptions	NULL

4.1.7 Taxi Driver inform about availability

Actors	Taxi Driver
Goals	[G6]
Input Conditions	<ol style="list-style-type: none">1. Taxi Driver must be logged in.
Output Conditions	<ol style="list-style-type: none">1. Taxi Driver clicks on the availability key2. The system redirects him to the availability page3. The driver clicks the Available button
Exceptions	NULL

4.1.8 Accept/Reject calls

Actors	Taxi Driver
Goals	[G7]
Input Conditions	<ol style="list-style-type: none">1. Taxi Driver must be logged in.2. Taxi Driver must be available
Output Conditions	<ol style="list-style-type: none">1. The system redirects the Taxi Driver to the calls page2. Taxi driver choose between accept or decline a certain call
Exceptions	NULL

4.1.9 Report a client

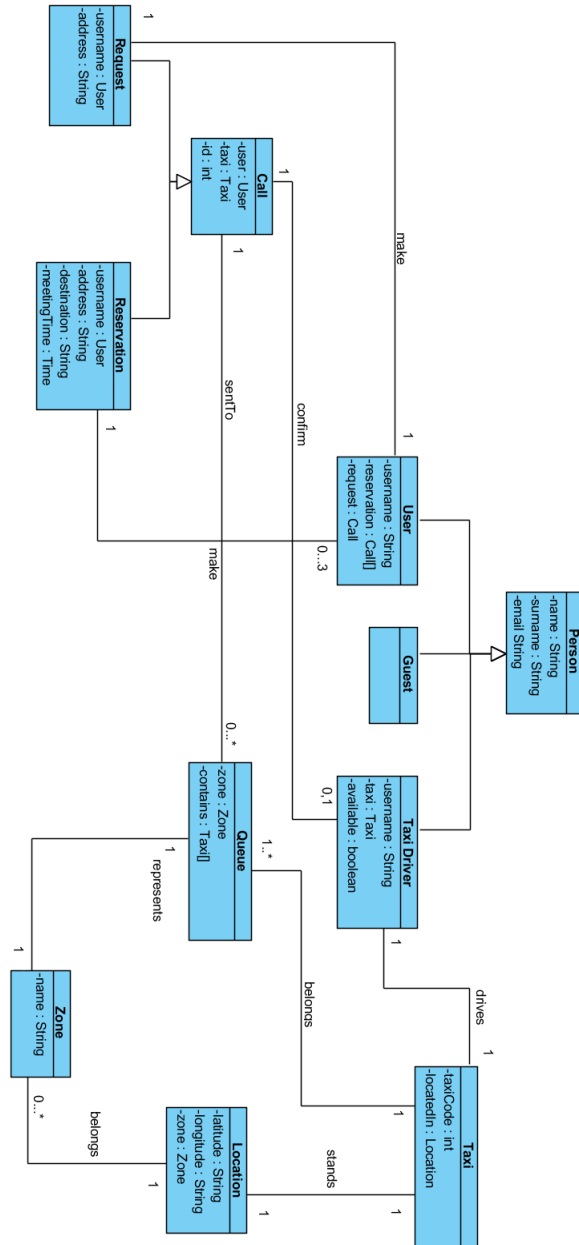
Actors	Taxi Driver
Goals	[G8]
Input Conditions	<ol style="list-style-type: none">1. Taxi Driver must be logged in.2. Taxi Driver must not be available
Output Conditions	<ol style="list-style-type: none">1. Taxi drivers clicks on the Report key2. The system redirects him to report form3. Taxi Drivers fill in all the fields (User's Username, id of the ride and if necessary optional comment field) of the form and click confirm button
Exceptions	<ol style="list-style-type: none">1. User's username must be correct2. Id must be valid

4.1.10 Declare end of the ride for a client

Actors	Taxi Driver
Goals	[G6]
Input Conditions	<ol style="list-style-type: none">1. Taxi Driver must be logged in.
Output Conditions	<ol style="list-style-type: none">1. Taxi Driver clicks on the Calls key2. The system redirects him to the Calls page3. The driver clicks the “End of the ride” button
Exceptions	NULL

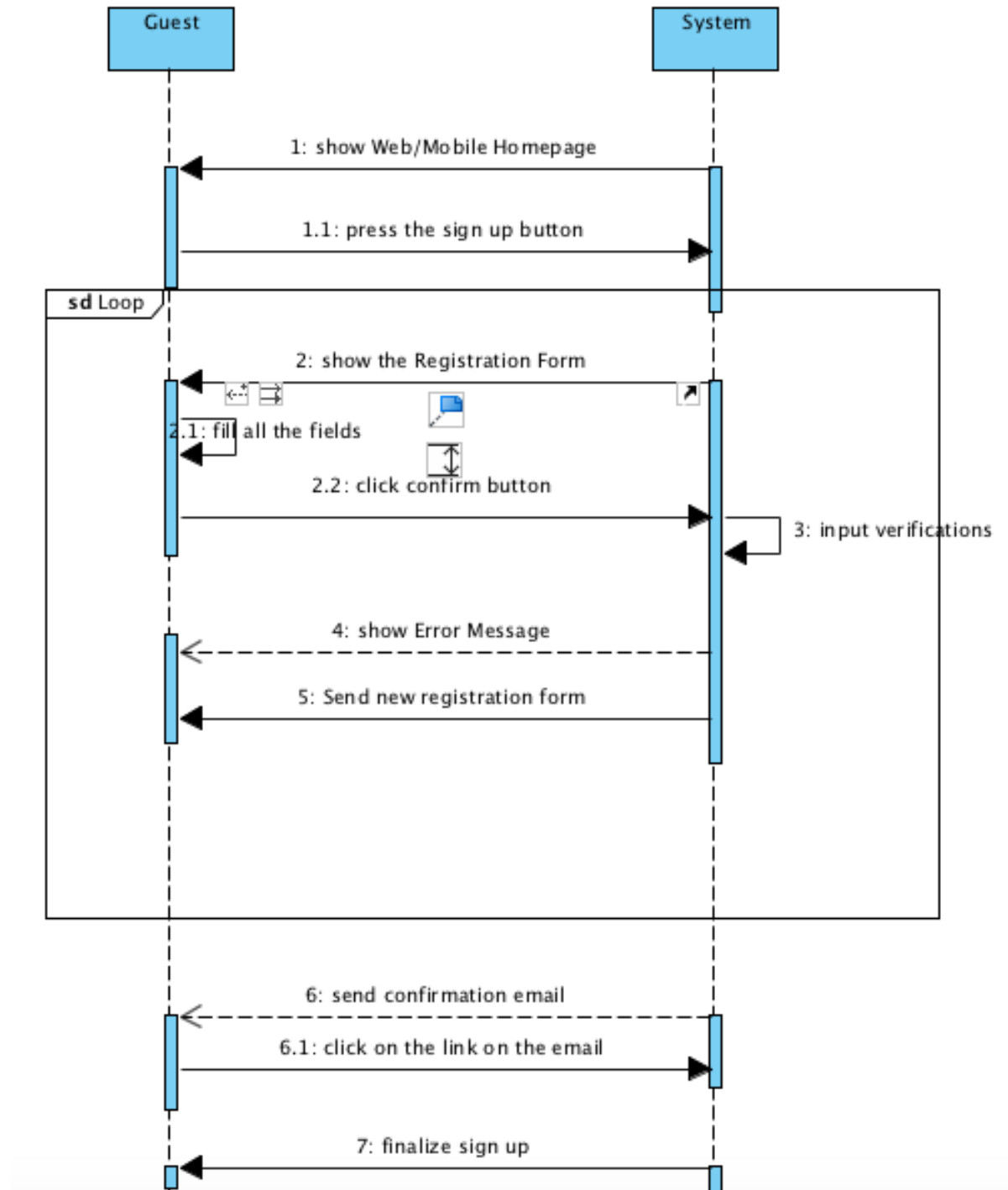
4.2 Class Diagram

This is the class diagram of our application. This diagram will be updated during the developing process with the inclusion of everything that is missing, especially the methods.

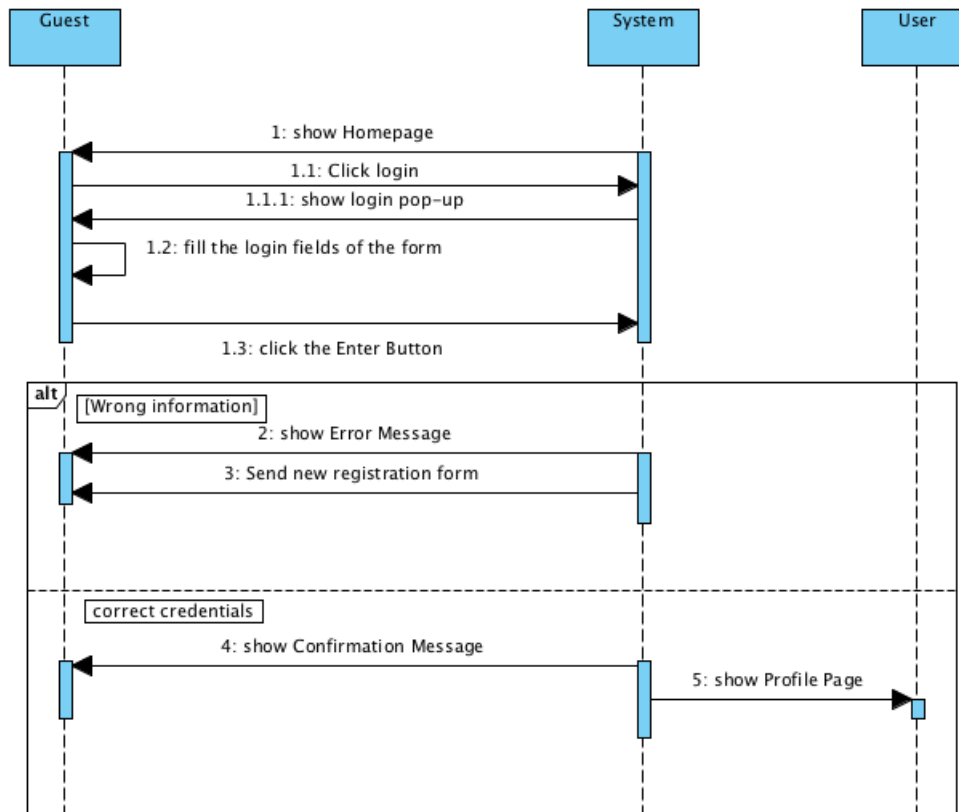


4.3 Sequence Diagram

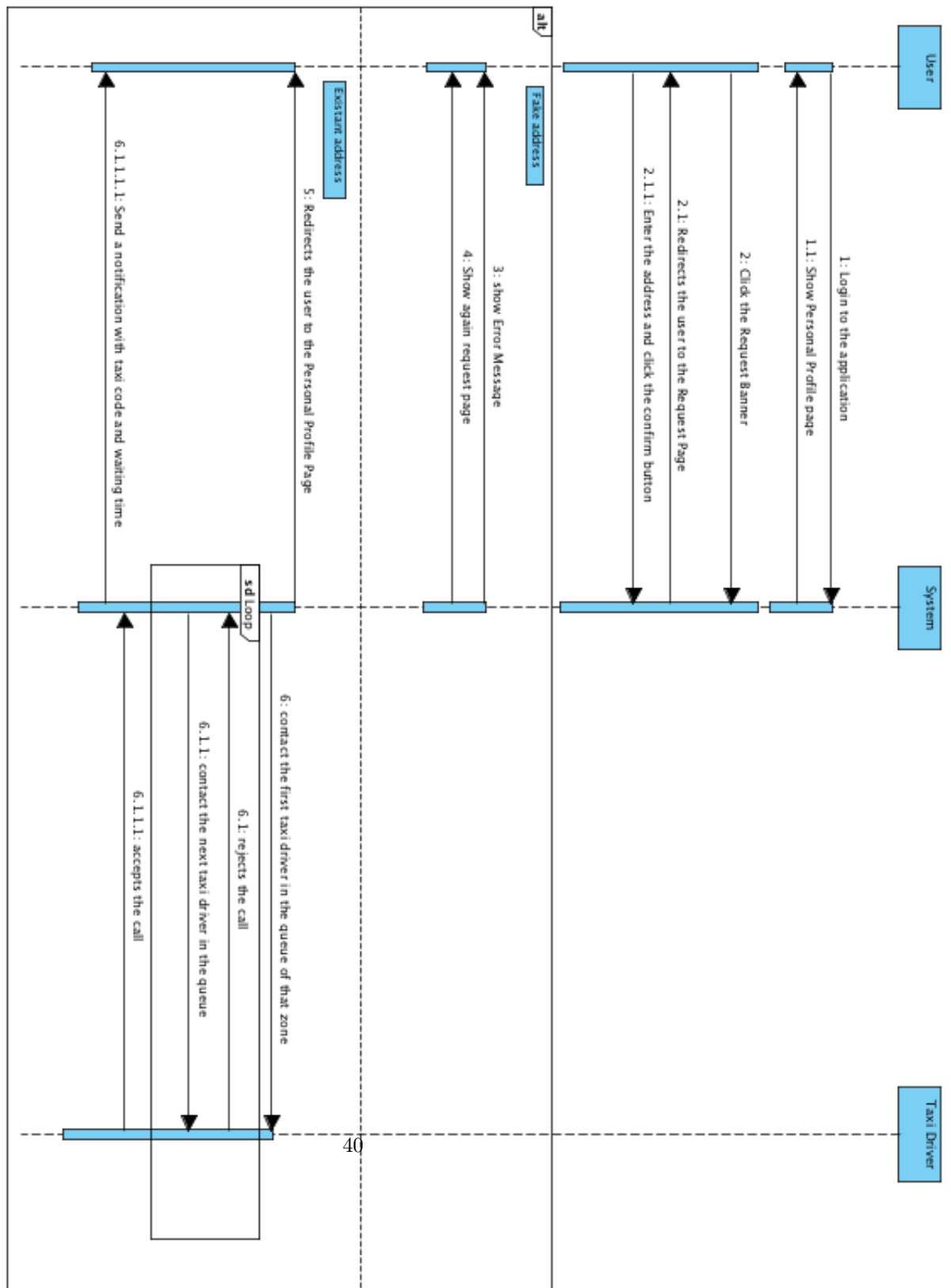
4.3.1 Registration



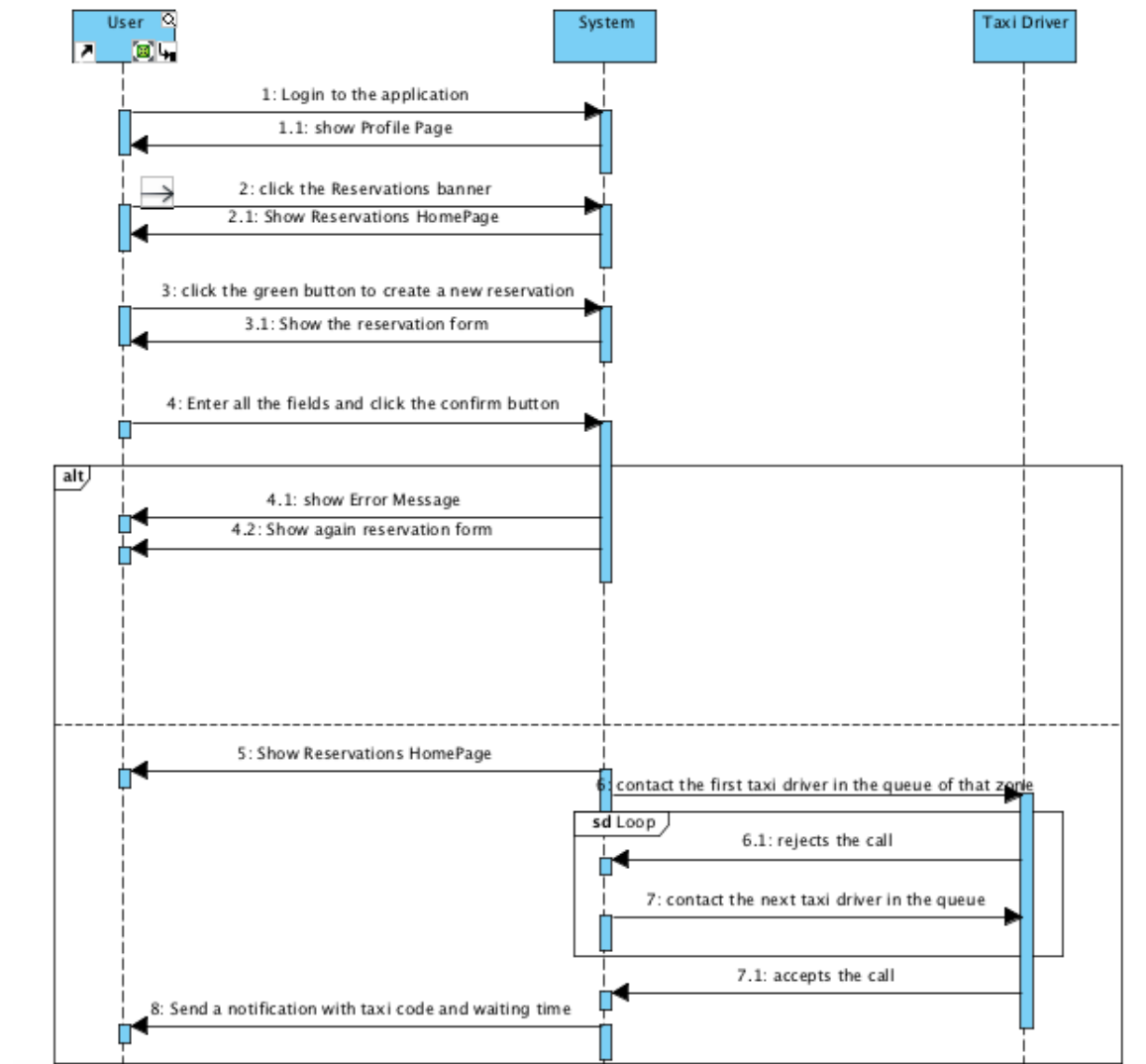
4.3.2 User's login



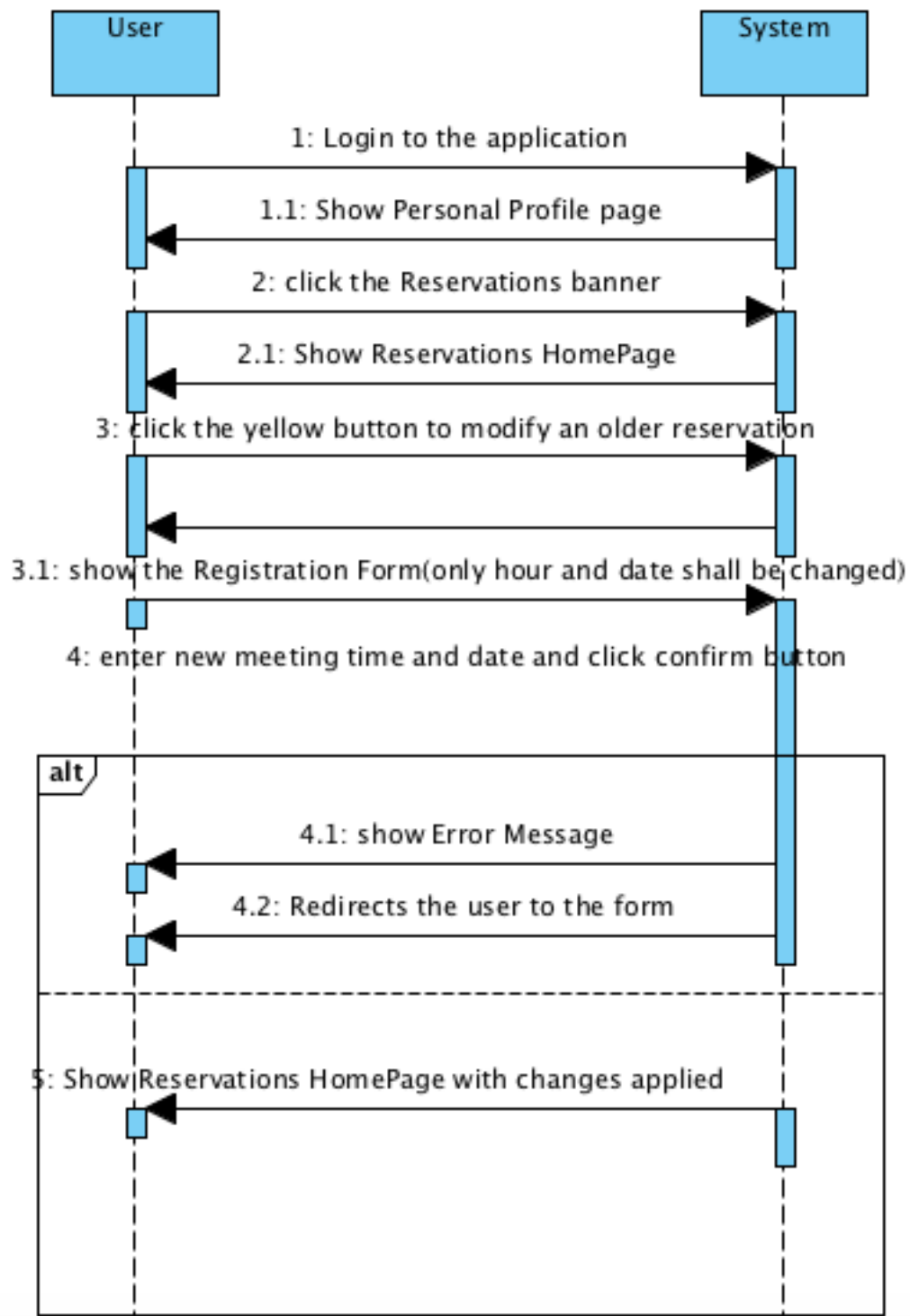
4.3.3 Request a taxi



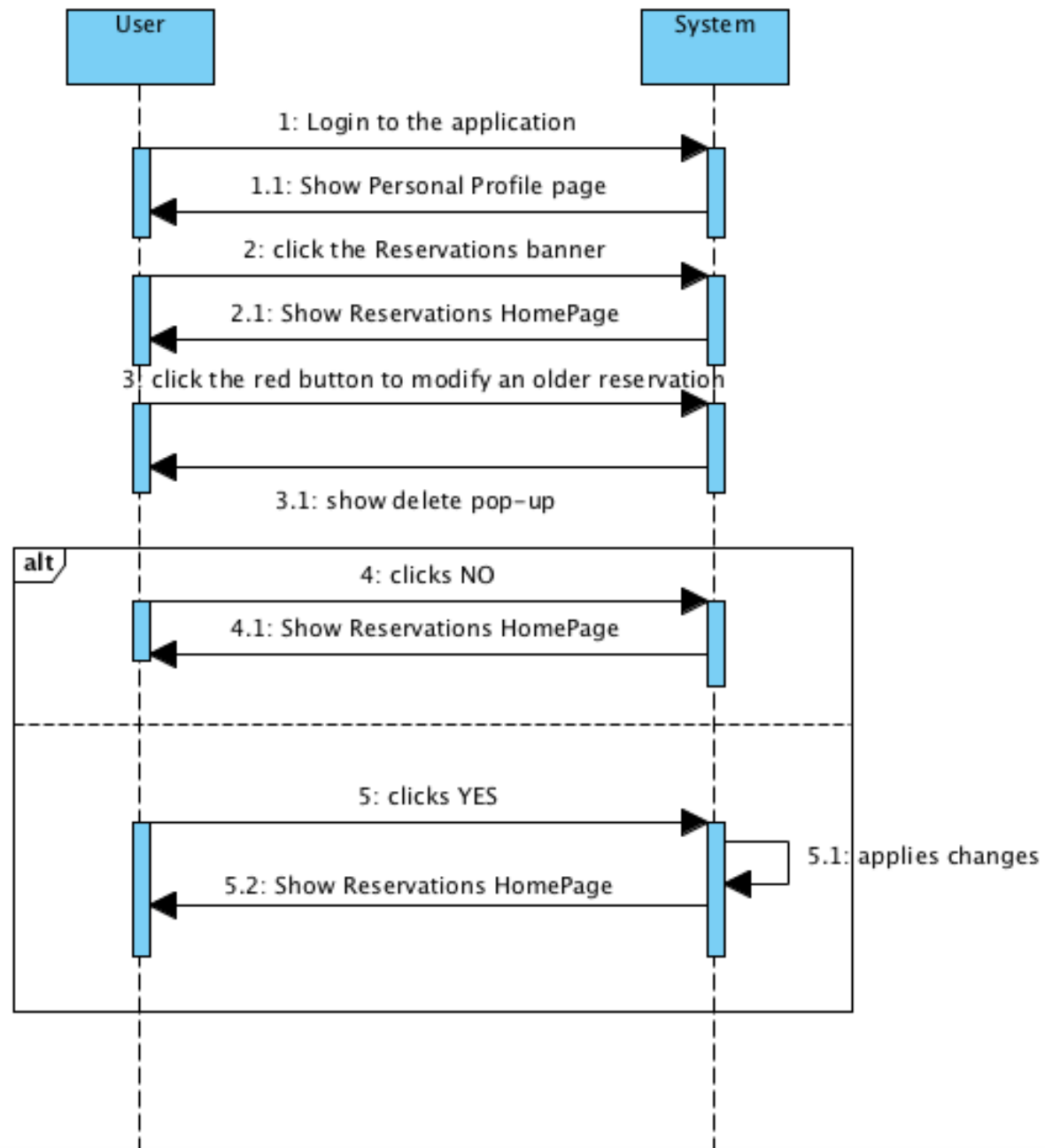
4.3.4 Reserve a taxi



4.3.5 Modify a reservation



4.3.6 Delete a reservation



5 Appendix

5.1 Alloy

Now we will prove whether the Class Diagram we made can be consistent by using Alloy Analyzer and we will also provide some Worlds generated to show relations between Classes and to clear the vision of how the system work. The Alloy code is divided in sections:

- Signatures;
- Facts;
- Assertions;
- Predicates;

5.1.1 Signatures

```
module alloy

// SIGNATURES

sig Taxi { locatedIn: one Location }
sig Location { isIn: one Zone }
sig Zone { }
sig Queue { zone: one Zone, contains: set Taxi }
abstract sig Person {}
sig User extends Person { reservation: set Reservation, request: lone Request }
sig Guest extends Person {}
sig TaxiDriver extends Person { possess: one Taxi }
abstract sig Call { associatedTo: one TaxiDriver, sendTo: one Queue }
sig Request extends Call {}
sig Reservation extends Call { date: one Date }
sig Date { }
```

5.1.2 Facts

```
//FACTS

fact UserCanHaveAtMost3Reservation { all u:User | #u.reservation<=3 }

fact EveryZoneHaveTheCorrispondentQueue { all z:Zone | one q:Queue | z in q.zone }

fact EveryLocationContainsATaxi { all l:Location | one t:Taxi | l in t.locatedIn }

fact EveryTaxiHaveExactlyOneTaxiDriver { all t:Taxi | one td:TaxiDriver | (t in td.possess) }

fact EveryCallIsMadeByOneUser { all c:Call | one u:User | (c in u.reservation || c in u.request) }

fact EveryDateIsReferredToAReservation { all d:Date | some r:Reservation | d in r.date }

fact EveryReservationOfSameUserHaveDifferentDate
  { all r0,r1:Reservation | all u:User | (r0!=r1 && r0 in u.reservation && r1 in u.reservation) implies (r0.date!=r1.date) }

fact ATaxiBelongsToOneQueue { all t:Taxi | one q:Queue | t in q.contains }

fact TaxiInTheSameZoneAreInTheSameQueue
  { all t1,t2:Taxi | some q:Queue | (t1!=t2 implies (t1.locatedIn.isIn=t2.locatedIn.isIn iff (t1 in q.contains && t2 in q.contains))) }

fact TaxiLocatedInAZoneIsInTheQueueThatRefersThatZone
  { all t:Taxi | some q:Queue | (t.locatedIn.isIn in q.zone && t in q.contains) }

fact ACallIsAssociatedToATaxiThatIsInTheQueueWhereTheCallIsSentTo
  { all c:Call | all q:Queue | q in c.sendTo implies (c.associatedTo.possess in q.contains) }

fact ATaxiDriverCanAcceptOnlyARequestOrReservationAtTheSameTime
  { all td:TaxiDriver | lone c:Call | td in c.associatedTo }
```

5.1.3 Assertions

```
// ASSERTIONS

assert UserCanHaveAtMost3Reservation { no u:User | #u.reservation>3 }
check UserCanHaveAtMost3Reservation

assert EveryZoneHaveExactlyOneQueue
  { (no z:Zone | all q:Queue | z not in q.zone) && (no z:Zone | some disj q0,q1:Queue | q0.zone=z && q1.zone=z) }
check EveryZoneHaveExactlyOneQueue

assert EveryTaxiHaveExactlyOneTaxiDriver { no t:Taxi | all td:TaxiDriver | (t not in td.possess) }
check EveryTaxiHaveExactlyOneTaxiDriver

assert EveryCallIsMadeByOneUser { no c:Call | all u:User | (c not in u.reservation && c not in u.request) }
check EveryCallIsMadeByOneUser

assert ThereIsNoTaxiThatBelongsToMoreThanOneQueue { no t:Taxi | some disj q0,q1:Queue | (t in q0.contains && t in q1.contains) }
check ThereIsNoTaxiThatBelongsToMoreThanOneQueue
```

5.1.4 Predicates

```
// PREDICATES

pred show() {}
run show for 20 but exactly 5 Zone, exactly 10 TaxiDriver, exactly 1 Guest

pred show3ReservationPerUser()
{ some disj u0: User, r0: Reservation, rq: Request | (r0 in u0.reservation) and (rq in u0.request) }
run show3ReservationPerUser for 15 but exactly 7 Call, exactly 4 Queue, exactly 9 Taxi

pred showDifferentDateForReservationOfSameUser()
{ some disj u0, u1: User, d0, d1: Date | d0 in u0.reservation.date && d1 in u0.reservation.date &&
  d0 in u1.reservation.date && d1 in u1.reservation.date }
run showDifferentDateForReservationOfSameUser for 10 but exactly 7 Reservation, exactly 3 Zone
```

5.1.5 Result

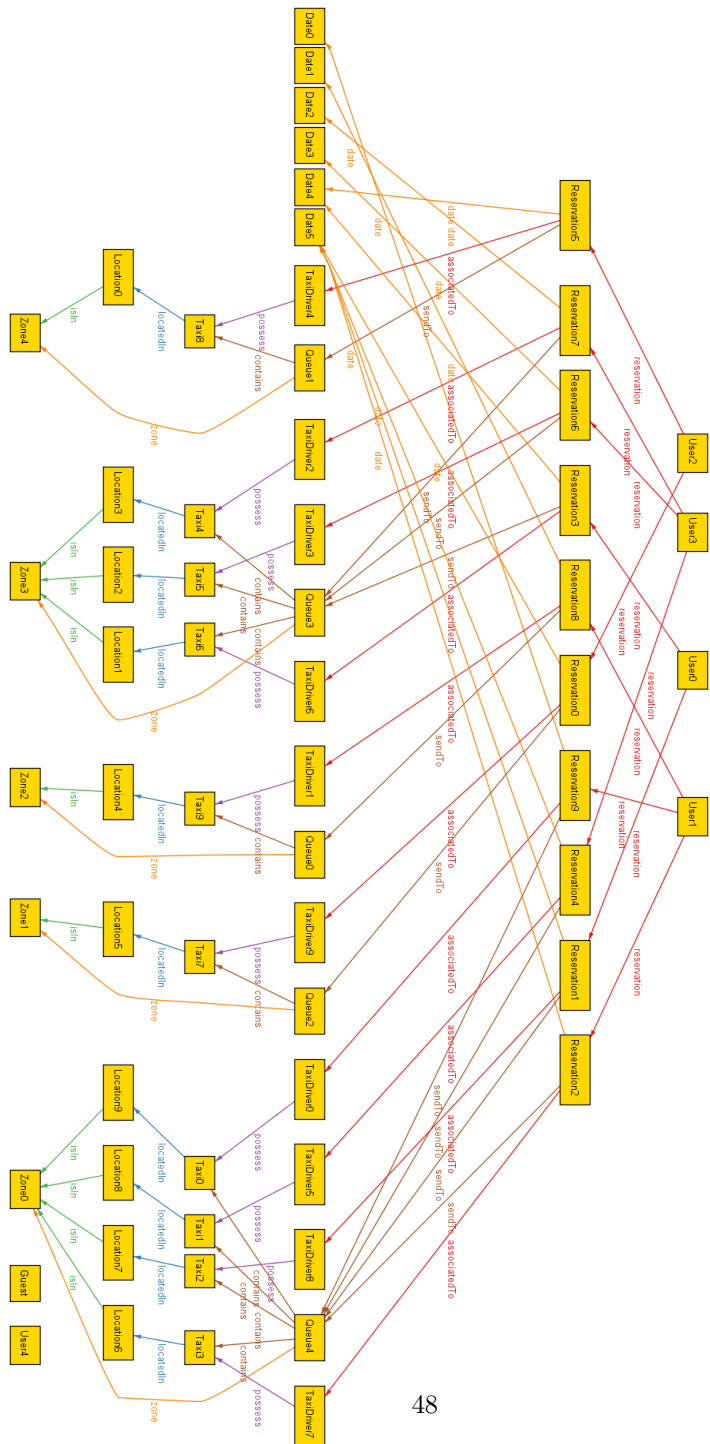
Here is the result from the analyzer that states our system is consistent.

8 commands were executed. The results are:

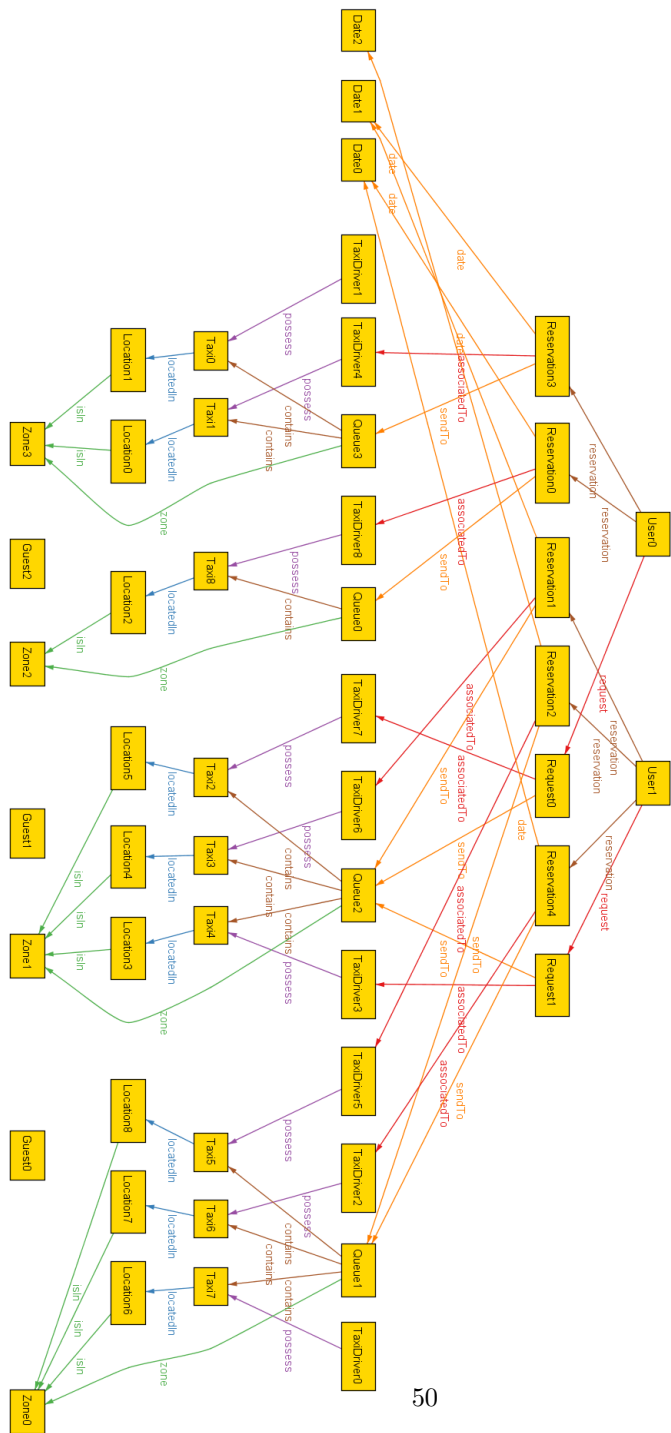
- #1: No counterexample found. UserCanHaveAtMost3Reservation may be valid.
- #2: No counterexample found. EveryZoneHaveExactlyOneQueue may be valid.
- #3: No counterexample found. EveryTaxiHaveExactlyOneTaxiDriver may be valid.
- #4: No counterexample found. EveryCallIsMadeByOneUser may be valid.
- #5: No counterexample found. ThereIsNoTaxiThatBelongsToMoreThanOneQueue may be valid.
- #6: **Instance found.** show is consistent.
- #7: **Instance found.** show3ReservationPerUser is consistent.
- #8: **Instance found.** showDifferentDateForReservationOfSameUser is consistent.

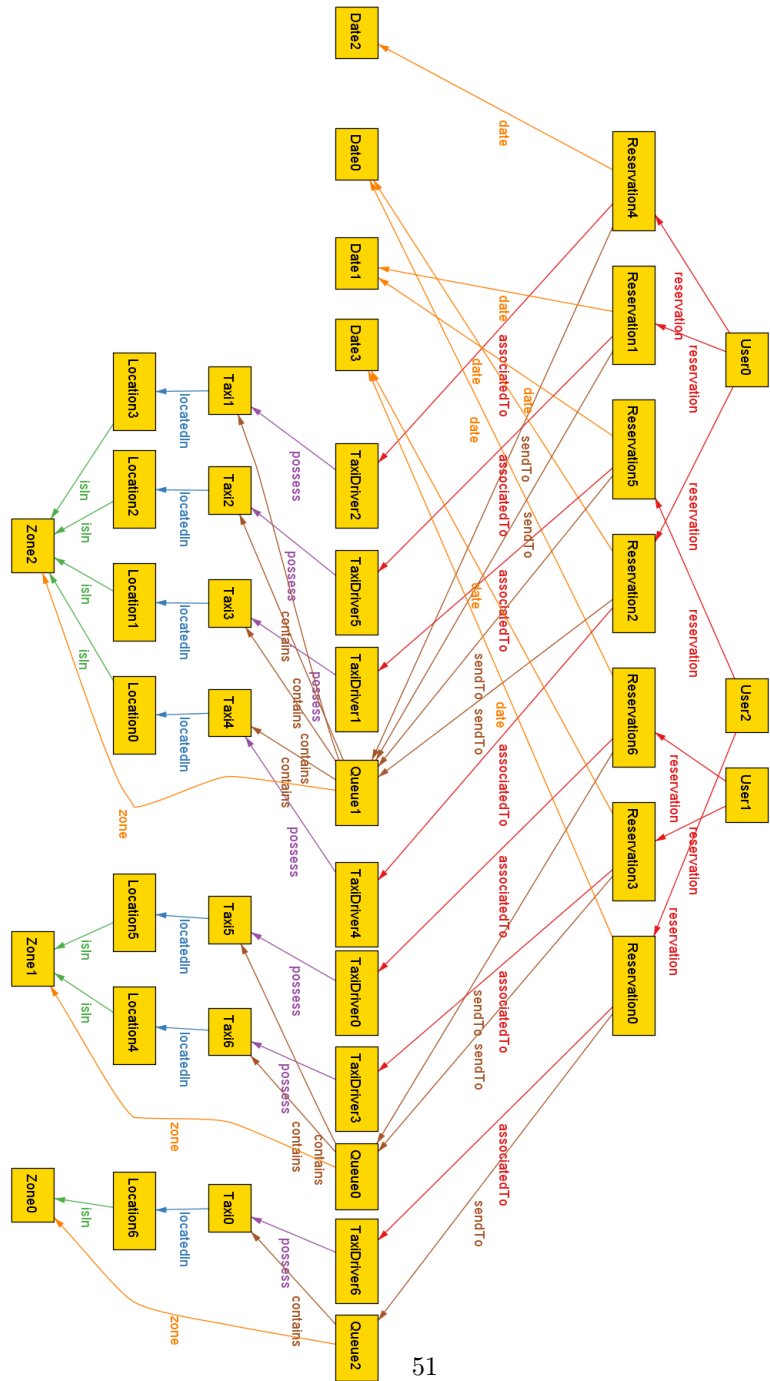
5.1.6 Generated Worlds

Finally the Worlds generated. General view of classes and their relations:



The following Worlds represent scenarios that show how the constraints about number of reservation per user and date of reservation for a user are not violated.





5.2 Softwares and tools used

We used these list of software and tools to create the RASD:

- Word Online (<https://office.live.com/start/Word.aspx?omkt=it-IT>): to redact this document;
- Lyx(<http://www.lyx.org/>): to format the document;
- Alloy Analyzer (<http://alloy.mit.edu/alloy/>): to prove the consistency of our model;
- Open Office (<http://www.openoffice.org>): we used the drawing module of OpenOffice to draw our mock-ups;
- Visual Paradigm (<http://www.visual-paradigm.com>): to draw Use Cases, Sequence Diagrams and Class Diagram.

5.3 List of differences with previous versions

This is the RASD 1.1 version. We edited some little things from RASD 1.0(RASD FINALE in our repository):

- Added the issue button in every mock-up regarding the taxi driver.
- Removed one requirement for goals 1 and 8 because they were useless.

5.4 Hours of work

This is the time spent to redact this document:

- Matteo Rizzi: 30 hours;
- Claudio Scandella: 30 hours;