Politecnico di Milano

A.A. 2015-2016

Software Engineering 2

"myTaxiService"

Requirements Analysis and Specification Document

(RASD)

Annalisa Rotondaro (mat. 854268)

Lorenzo Federico Porro (mat. 859093)

Contents

1 Introduction

1.1 Purpose

This document contains all the specifications and all the constraints that this project (see 1.2) requires. Inside will also be provided use cases and models (using UML and Alloy) for the main parts of the system to be developed. The goal of this document is to completely describe both the functional and non-functional requirements that needs to be taken in consideration in the next phases of the software lifecycle (when, described the design of the application in the design phase, implementing the code in the implementation phase, do the other phases, integration, validation and deployment).

1.2 Description of the given problem

The government of a large city aims at optimizing its taxi service. In particular, it wants to:

- simplify the access of passengers to the service;

- guarantee a fair management of taxi queues.

Passengers can request a taxi either through a web application or a mobile app. The system answers to the request by informing the passenger about the code of the incoming taxi and the waiting time. Taxi drivers use a mobile application to inform the system about their availability and to confirm that they are going to take care of a certain call. The system guarantees a fair management of taxi queues. In particular, the city is divided in taxi zones (approximately 2 km square each). Each zone is associated to a queue of taxis. The system automatically computes the distribution of taxis in the various zones based on the GPS information it receives from each taxi. When a taxi is available, its identifier is stored in the queue of taxis in the corresponding zone. When a request arrives from a certain zone, the system forwards it to the first taxi queuing in that zone. If the taxi confirms, then the system will send a confirmation to the passenger. If not, then the system will forward the request to the second in the queue and will, at the same time, move the first taxi in the last position in the queue. A user can also reserve a taxi by specifying the origin and the destination of the ride. The reservation has to occur at least two hours before the ride. In this case, the system confirms the reservation to the user and allocates a taxi to the request 10 minutes before the meeting time with the user. 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.

1.2.1 Intended Audience

RASD document is aimed, not only, to all developers and programmers who have to implement the requirements, but also to technicians and system analysts

who want to integrate other systems with the one here described, to testers, to project managers and also to stakeholders who can contribute requirements or knowledge to help understand the requirements. This paper is also a contractual base between the customer and the developers.

1.3 Scope

The aim of the project is to create both a web "myTaxiService.com" and a mobile "myTaxiServiceApp" applications for denizens and taxi drivers, as well as an efficient management system for taxi queues.

The application is aimed both to denizens of large cities and taxi drivers. The former will have a major quality-of-life improvement in the way they benefit of the city's taxi service, by having an intuitive and immediate interface to make taxi calls and to quicken whole process of taxi calls. The latter will have a fair queue system and management and also a fast and efficient way to start their everyday job through the mobile app.

1.3.1 Benefits

Main benefits for taxi drivers:

- Easy and immediate way to organize their job;

- Increment of their efficiency;

- Fair competition.

Main benefits for users:

- Immediate way to access the taxi service;

- The system is suitable for all audiences;

- The system is almost cost-free;

- This system make the service accessible from nearly anywhere.

1.3.2 The world and the machine

Here is shown the world and the machine model for myTaxiService project, in terms of world, machine and shared phenomena (the ones at the center):
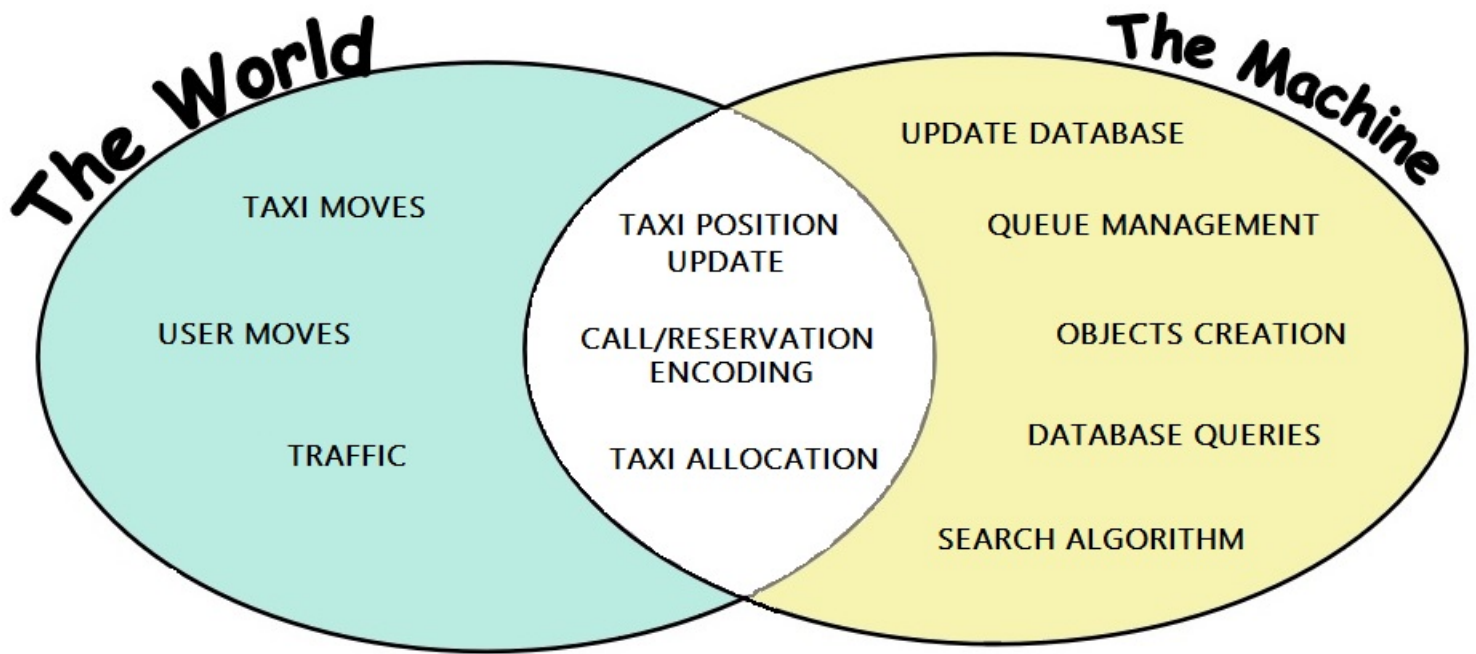
1.3.3 Goals

For this project the following goals have been identified:

1. Provide a personal page where users can see their active taxi reservations

2. For every taxi call a taxi should be on the place of the pickup within 10 minutes

3. Provide to drivers a system to manage their availability and incoming calls

4. The system should provide a fair management of taxi queues

5. Provide a notification system to let know users when a taxi is going to the pick up or to let know to drivers when a pick up is requested

6. Provide a call/reservation system to allow users to arrange pickups

1.4 Definitions, acronyms and abbreviations

1.4.1 Definitions

- **Call:** A taxicall requested immediately (if no distinction is made between reservations and call this refers simply to pickups);

- **Reservation:** A reservation of a taxi for a later pickup;

- **Request:** The request of a call or a reservation;

- **Notification:** A notification sent to drivers or users to communicate certain informations;

- **System:** The main system where the logic of the application is hosted;

- **Web app:** The site accesible by any browser;

- **Mobile app:** The application from which a user or a driver is able to access myTaxiService services;

- **Guest:** A user not registered yet;

- **Client:** Registered taxi clients;

- **User:** Registered People;

- **Driver:** Registered taxi drivers.

1.4.2 Acronyms

- **GUI:** Graphical User Interface;

- **API:** Application Programming Interface;

- **GPS:** Global PositioningSystem;

- **UML:** Unified Modelling Language;

- **JVM:** Java Virtual Machine;

1.4.3 Abbreviations

- **DB:** DataBase;

- **G:** Goal;

- **R:** Requirement;

- **FR:** Functional Requirement;

- **NFR:** Non Functional Requirement.

1.5 Overview

The document has the following structure:

- Section 1: contains a general overview of the problem and an high-level description of what the system is expected to do and of the setting in which the system will be introduced;

- Section 2: describes in detail the problem with a focus on interfaces and constraints that need to be faced, here are also detailed all the assumptions that are supposed to hold in redacting this document;

- Section 3: contains all the functional requirements for the system as well as UML models for the main parts that have to be implemented. The requirements stated here are divided by user class;

- Appendices: contains logic contraints and relations that could be useful during the implementation phase, modelled using Alloy;

- Supporting Informations: contains additional information on the tools and the documentation used to redact this document as well as information about its writing.

2 Overall description

2.1 Product perspective

MyTaxiService system is developed to be used either from any browser as a web app or from mobile devices through the mobile app, the software is not meant to be integrated with other existing systems so no interfaces of this kind are offered. The main system exchanges informations and data with web and mobile apps users through a common interface (this interface is hidden from the outside of the system). The mobile version of the software is going to be developed both for Android and IOs and thus it relies on GPS functions offered by Android and IOs devices to offer the position tracking service to taxi drivers and to auto-fill the pickup position field for users (even though for the latter is not mandatory) as specified in the following subsections. The system also offers programmatic API to allow in future the implementation of additional functions, some examples of these functions are presented below.

Here's depicted an high level general schema of the system and of its interfaces (through UML component diagram):
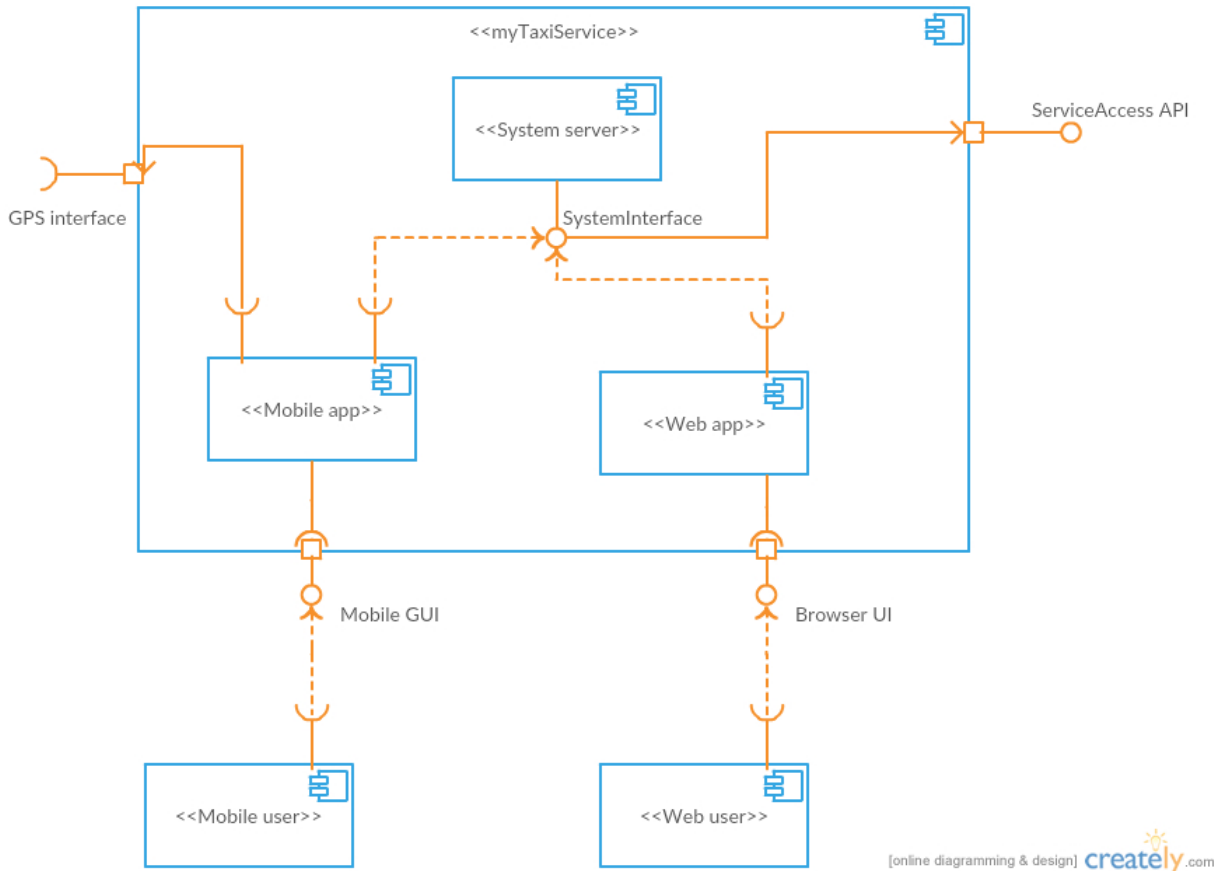
2.1.1 System interfaces

The system uses an internal interface "SystemInterface" common both to the web app and the mobile app, through which the app user-side can access the services offered by the main system.

2.1.2 User interfaces

Our system must have a simple interface to help user understand clearly what they have to do when they use the application from the mobile app or the web app. The service must be used by young and elderly people alike. To guarantee this the GUI must have only few elements with clearly stated words for buttons and forms. Any browser may reach the home page of the application, by connecting to the server address, that will have a significant name. The application have a significant name too to be found in the easiest way possible in the appstore of any mobile device.

2.1.3 Hardware interfaces

The mobile app should be supported by any Android device and IOs based device. Also the app should not interfere with system functions of such devices or with other applications running concurrently. Regarding this purpose the memory required to run the application must be limited within reasonables boundaries, ideally <10 Mb. The web app, instead, should be supported by all major browsers (Chrome, Edge,Opera, Firefox) at least. The main system will run on a server solution by Dell `http://www.dell.com/us/business/p/poweredge-t630/pd?~ck=anav`and as such refers to the manual for implementation purposes.

2.1.4 Software interfaces

DBMS

- Name: Microsoft SQL Server 2014 Express

- Scope: Manage the DB containing all user' s data

- Version number: 12.0.2000.8

- Source: `11.0.2100.60`

Application Server

- Name: GlassFish

- Scope: Manage connections and requests between users and the system

- Version number: 4.1.1

- Source: `https://glassfish.java.net/`

Operating System

- Name: Windows Server 2012 R2

- Scope: System that interact with the application

- Version number: February 2014

- Source: `https://www.microsoft.com/it-it/server-cloud/products/windows-server-2012-r2/overview.aspx`

Application Environment

- Name: Java

- Scope: VirtualMachine where the system will run on

- Version number: 8

- Source: `https://www.java.com/it/`

Android app

- Name: Android

- Scope: The environment where the android version of the app will run on

- Version number: 6.0

- Source: `https://www.android.com/intl/it_it/`

IOs app

- Name: IOs

- Scope: The environment where the IOs version of the app will run on

- Version number: 9

- Source: `http://www.apple.com/it/ios/?&cid=wwa-it-kwg-features`

2.1.5   Communications interfaces

Client-Server

- All the communications between users and the system are made by using HTTP protocol in the application level and TCP in the transport level.

2.1.6   Memory

System memory

- 1+ GB (as stated in the requirement of Microsoft SQL Server 2014 Express)

DB memory

- 50+ GB

2.1.7   Operations

- Every User can connect to the system but only registered ones can access the myTaxiService services.

2.2   Product functions

- Users and Taxi drivers can login and register into the system;

- Users can call and reserve taxis;

- A taxi driver can answer requests;

- Taxi drivers can manage their availability;

- The system manages the queues of taxis in every zone in a fair way.

2.3 User characteristics

The service could be accessed by any people having access to a mobile device (for the mobile app) or a browser (for the web app). The system offers an easy interface and as such every user with at least 5th grade and some experience with browsers and mobile devices should be able to use it correctly either from the web or mobile app.

2.4 Constraints

- **Regulatory policies:** Users destinations can be seen only by drivers and taxi numbers can be seen only by the user that requested the pickup;

- **Hardware limitations:** No limitations;

- **Interfaces to other applications:** The system offers API to allow to add more functionalities in the future;

- **Parallel operation:** The system is able to handle more connections at once;

- **Audit functions:** Not provided;

- **Control functions:** Not provided;

- **Higher-order language requirements:** the following languages are required: Java, Sql, HTML;

- **Signal handshake protocols:** Not provided;

- **Criticality of the application:** Managed by the administrator;

- **Safety and security considerations:** Not required.

2.5 Assumptions and dependencies

For everything specified in this document the following domain assumptions are supposed to hold:

- Hardware and/or software failures never occurs;

- For every call and/or answer the communication between parts of the system is always error-free;

- Drivers that accept a call always go to pick up that client;

- Drivers always accept reservation requests;

- When a driver accept a call he departs immediately;

- Car accidents that can delay a taxi never occurs;

- The maximum time needed to cross a single zone is 10 minutes;

- Users that call a taxi wait for the taxi, and take their trip without requesting other taxis;

- A single user don't make multiple reservations for the same day at the same hour;

- Users that reserve a taxi will be available at the time of the scheduled pick up;

- Users correctly notify the place of the pick up;

- Drivers correctly signal their availability through the app;

- Available taxi's position is known by GPS;

- For each reservation there is at least one taxi available that will answer that call;

- The taxi number (code) is a 4 digit number assigned by the transport authority of the city.

3 Specific Requirements

3.1 External interface requirements

3.1.1 User interfaces

Here are portrayed some mockups of user interfaces of both the mobile app and the web app
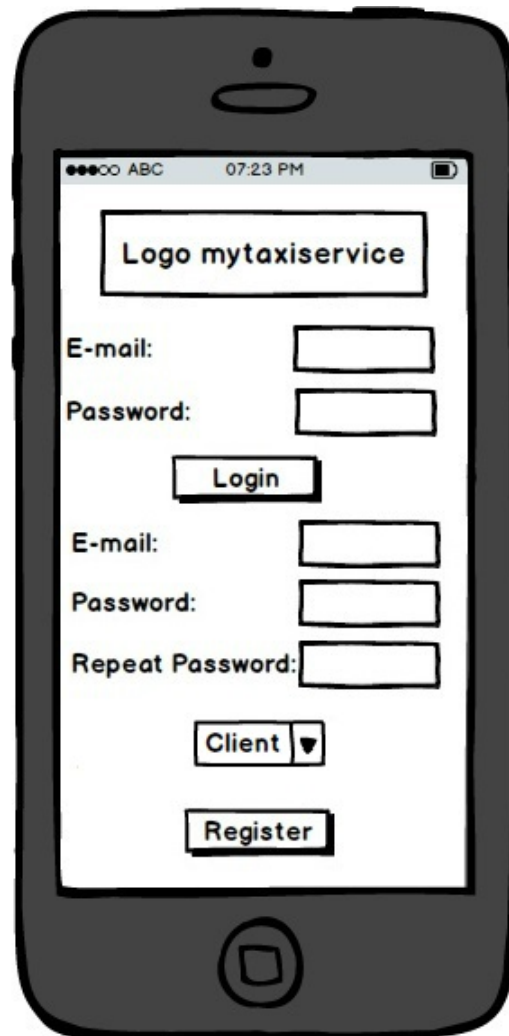
Figure 1: This is an example of the first screen of the application that appear when you download it and open it for the first time. This page allow you to register if you aren't registered yet or login if you registered yourself in the web application. You can either choose if you are a client or a taxi driver, and for the latter, you have to insert the code of the taxi in the pop up that appear later. This screen appears only the first time you access the mobile app because from themobile version all users and drivers are always logged in.
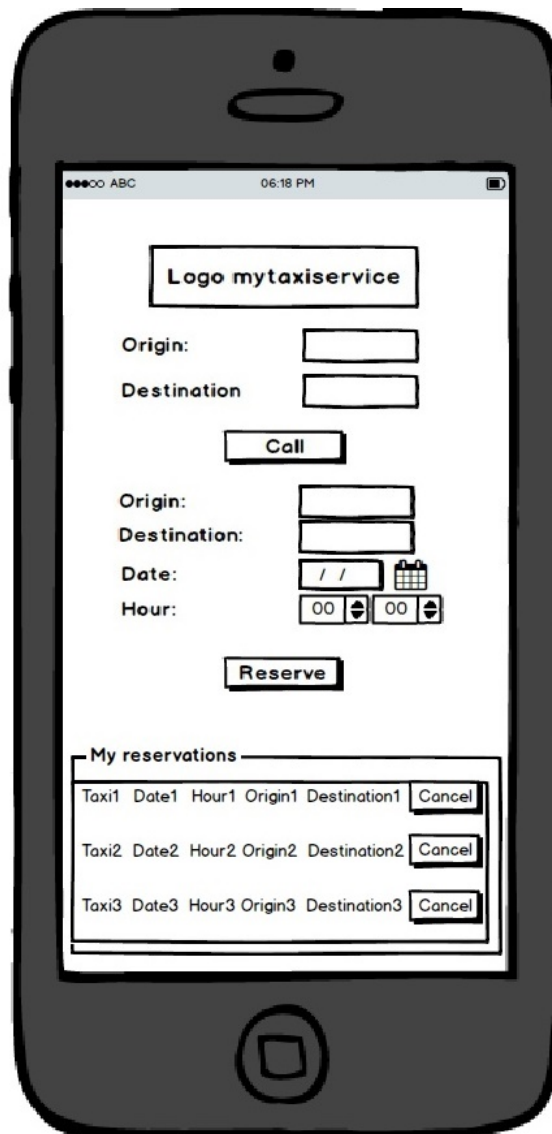
Figure 2: The personal page of the user. Here you can call or reserve a taxi just by putting the information in the respective field. You can also view the other reservation that you have made and cancel them if you don't want a taxi anymore.

Type of inputs

- Each blank field requires an input of type text. No special characters are allowed, only letters (latin alphabet) and numbers (arab numbers, between

14
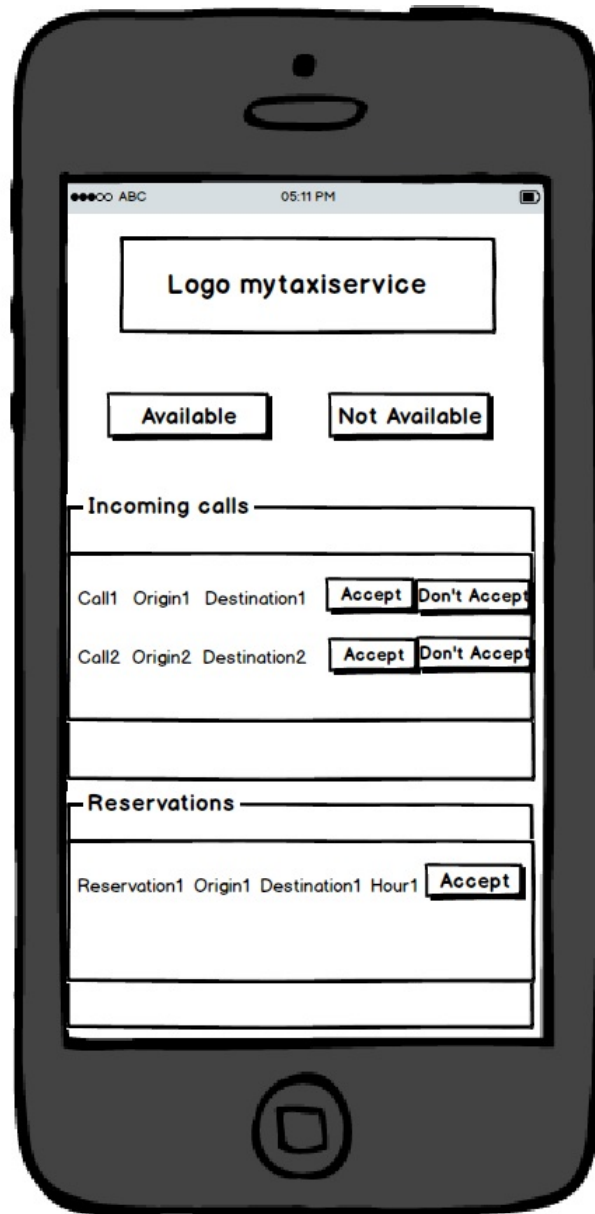
Figure 3: The personal page of the taxi driver. Here the driver can signal taht he is available or not available in the zone where he is. He can accept or don't accept an incoming call and he have to accept the resevations received by the system.
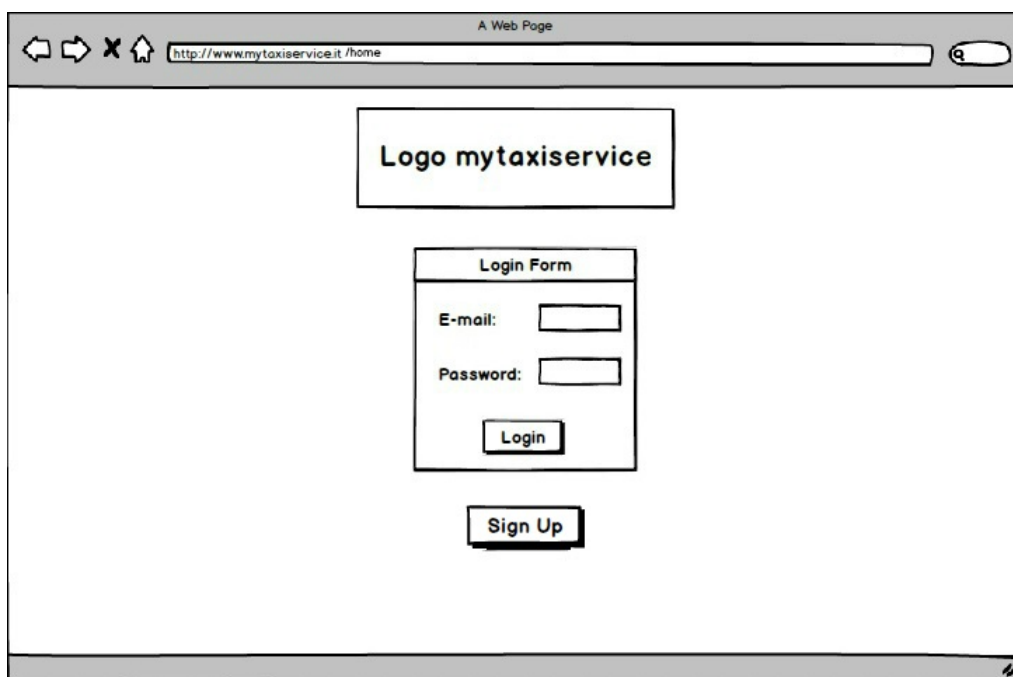
Figure 4: The home page of the web site of the system. Here you are have to either login or register if you want to use the service.

Figure 5: The registration page when you can insert all the necessary information to register to the service, una tantum.

Figure 6: The personal page of the web site of the client. Here you can, as the personal page of the application (fig 2), call or reserve a taxi, see and, if you want, cancel the reservations you have done in the past, if there are anyone.

0 and 9);

- The field date requires an input of type date in the format dd/mm/yy only arab numbers allowed. The day part (dd) must be inside the range 00 and 31. The part month (mm) must be inside the range 00 and 12. The part year (yy) must be in the range 00 and 99;

- Scroll down menus (hour, user type...) requires a direct input (via mouse or tap) one on the little arrow to expand the menus and one to select the option. Hour menu option are all integers numbers inside the range 0 and 12 for the left part and between 0 and 59 for the right part. User type menu (fig. 1) contains the two options: "client" and "taxi driver".

3.1.2 Software interfaces

The system offers an external programmatic interface to allow the implementation of new functions (like a taxi sharing function, a suggestion system or a removal system), the functionalities that this interface must offer are the following::

- **Route calculation:** given a User and a Request the methods returns the route of the trip: the starting and destination Zones as well as the Addresses;

- **Queue informations:** given a specific Zone the Queue of all taxis available and waiting in that zone is returned;

- **Requests informations:** returns all active requests (not yet accepted) of a given zone;

- **Position informations:** returns the exact position (GPS coordinates) of a User or a TaxiDriver;

- **User removal:** removes a specific User from the system, effectively deleting all his data from the system (email and password);

- **Driver removal:** removes a specific taxi driver from the system and deleting all his personal data from the system (email, password and taxi code).

3.2 Functional requirements

3.2.1 User class 1 (Client)

Requirement Name	R1 : REGISTRATION INTO THE SYSTEM
Goals	G1 : Provide a personal page where users can see their active taxi reservations
Preconditions	• Guest is not registered yet
Description	By clicking on the "Sign up" button in the home page the Guest is taken to the registration form where he must provide his valid e-mail and choose his password (each field is mandatory), after filling the registration form and clicking on the "register" button under the form the User is registered into the system and is now able to use all the services "myTaxiService" offers.
Use Case Name	RegistrationUser
Participating Actors	• Guest
Use Case	

Table 1: Requirement 1 with its related use case diagram

Requirement Name	R2: LOGIN INTO THE SYSTEM
Goals	G1 : Provide a personal page where users can see their active taxi reservations
Preconditions	• User is registered into the system • User is not yet logged in
Description	The User fills the form in the Home page with the e-mail and password he chose in the registration phase, if these are correct he is then taken to his personal page where he can see his active calls and call/reserve taxis.
Use Case Name	Login
Participating Actors	• User
Use Case	

Table 2: Requirement 2 with its related use case diagram
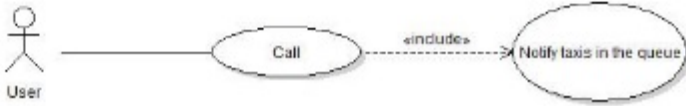
Requirement Name	R3: CALL A TAXI
Goals	G2 : For every taxi call a taxi should be on the place of the pickup within 10 minutes G5: Provide a notification system to let know users when a taxi is going to the pick up or to let know to drivers when a pick up is requested G6: Provide a call/reservation system to allow users to arrange pickups
Preconditions	· User is logged in
Descriptions	The User fills the call form on the left side of the personal page with the origin and destination of his ride (all fields are mandatory). After clicking onto the "call" button the system starts the call procedure and notifies the first taxi waiting in the queue about the call. When the call is accepted (or refused) by a driver (view R9) a popup notifies him that a taxi is coming (or not coming). If a taxi is incoming the popup will also tell the ETA and the code of the taxi.
Use Case Name	CallTaxi
Participating Actors	· User
Use Case	

Table 3: Requirement 3 with its related use case diagram

Requirement Name	R4: RESERVE A TAXI
Goals	G1 : Provide a personal page where users can see their active taxi reservations G5: Provide a notification system to let know users when a taxi is going to the pick up or to let know to drivers when a pick up is requested G6: Provide a call/reservation system to allow users to arrange pickups
Preconditions	· User is logged in · Hour of the reservation is at least 2 hour later than current time
Description	The User fills the reservation form on the right side of the personal page with origin and destination of the ride, date and hour desired for the pickup (all fields are mandatory). After clicking onto the "reserve" button the system starts the reservation procedures, a popup confirms the reservation and the call appears under the section "My reservations" with all the relevant informations.
Use Case Name	ReserveTaxi
Participating Actors	· User
Use Case	

Table 4: Requirement 4 with its related use case diagram

Requirement Name	R5: DELETE A RESERVATION
Goals	G1 : Provide a personal page where users can see their active taxi reservations G5: Provide a notification system to let know users when a taxi is going to the pick up or to let know to drivers when a pick up is requested G6: Provide a call/reservation system to allow users to arrange pickups
Preconditions	- User is logged in - A reservation have been made by the User - Hour of the reservation is at least 2 hour later than current time
Description	The User clicks onto the cancel button next to the reservation he wants to cancel under "My Reservations" section, then a popup asks to the User if he's sure about his decision. By clicking onto the "cancel" button inside the popup. Another popup then confirms the cancellation and the reservation is deleted form the section "My Reservations"
Use Case Name	DeleteReservation
Participating Actors	- User
Use Case	

Table 5: Requirement 5 with its related use case diagram

3.2.2 User class 2 (Taxi driver)

Requirement Name	R6: REGISTRATION INTO THE SYSTEM
Goals	G3: Provide to drivers a system to manage their availability and incoming calls
Preconditions	• Driver is not yet registered
Description	The driver after downloading the application fills the form for register himself , he must provide e-mail and password, clicks onto combobox and select "Taxi driver". He have to provide also the taxi number assigned by the government authority (all the fields are mandatory). After he clicks onto the register button he will be able to accept calls and reservations.
Use Case Name	RegistrationTaxiDriver
Participating Actors	• Taxi Driver
Use Case	

Table 6: Requirement 6 with its related use case diagram

25

Requirement Name	R7: BECOME AVAILABLE
Goals	G3: Provide to drivers a system to manage their availability and incoming calls G4: The system should provide a fair management of taxi queues
Preconditions	· Driver is logged in · Driver is currently not available · Driver is inside one zone of the city
Description	When the driver is not available and is inside one valid zone of the city he can become available to answer calls and reservations by clicking the "Available" button inside his personal page. He is then placed in the last position of the queue of the zone he is currently inside (position known by GPS) and can now receive notifications about incoming calls.
Use Case Name	BecomeAvailable
Participating Actors	· Taxi Driver
Use Case	

Table 7: Requirement 7 with its related use case diagram

Requirement Name	R8: BECOME UNAVAILABLE
Goals	G3: Provide to drivers a system to manage their availability and incoming calls G4: The system should provide a fair management of taxi queues
Preconditions	· Driver is logged in · Driver is currently available · Driver is currently waiting into one queue of the city
Description	When the driver is waiting into a queue he can decide to become unavailable by clicking onto the "Not available" button inside his personal page. He is then removed from the queue and he can no longer receive any notification about incoming calls.
Use Case Name	BecomeUnavailable
Participating Actors	· Taxi Driver
Use Case	

Table 8: Requirement 8 with its related use case diagram

Requirement Name	R9: ANSWER AN INCOMING CALL
Goals	G2 : For every taxi call a taxi should be on the place of the pickup within 10 minutes G3: Provide to drivers a system to manage their availability and incoming calls G4: The system should provide a fair management of taxi queues G5: Provide a notification system to let know users when a taxi is going to the pick up or to let know to drivers when a pick up is requested G6: Provide a call/reservation system to allow users to arrange pickups.
Preconditions	• Driver is available • Driver is waiting in the queue of the zone where the call is requested
Description	When a taxi call is issued the system forwards the request to the first taxi waiting in the queue of that zone. A notification is sent to his mobile device and the details of the call appears under the "incoming calls" section of his personal page, he can then accept or don't accept (by clicking onto the respective button near the call) or don't choose neither. If he accepts, he is removed from the queue and leaves to go to pick up the client. If he refuses the call, he is removed from the queue and put in the last position of the queue of the same zone and the call is forwarded to the next taxi waiting in the queue. If he doesn' t answer within 1 minute from the notification the call is considered refused. When he accepts a call his status is automatically set to "not available". If for a certain call the queue ends without any driver accepting the call, the client that requested the pickup is notified that no taxis are available at the moment.
Use Case Name	AnswerCall
Participating Actors	• Taxi Driver
Use Case	

28

Table 9: Requirement 9 with its related use case diagram

3.2.3 Class diagram

Here is attached a simple view of the system structure, made by a class diagram. This high level diagram represents the entities of the system. In the Design Document the class diagram will be more detailed regarding the implementation logic and the system structure.
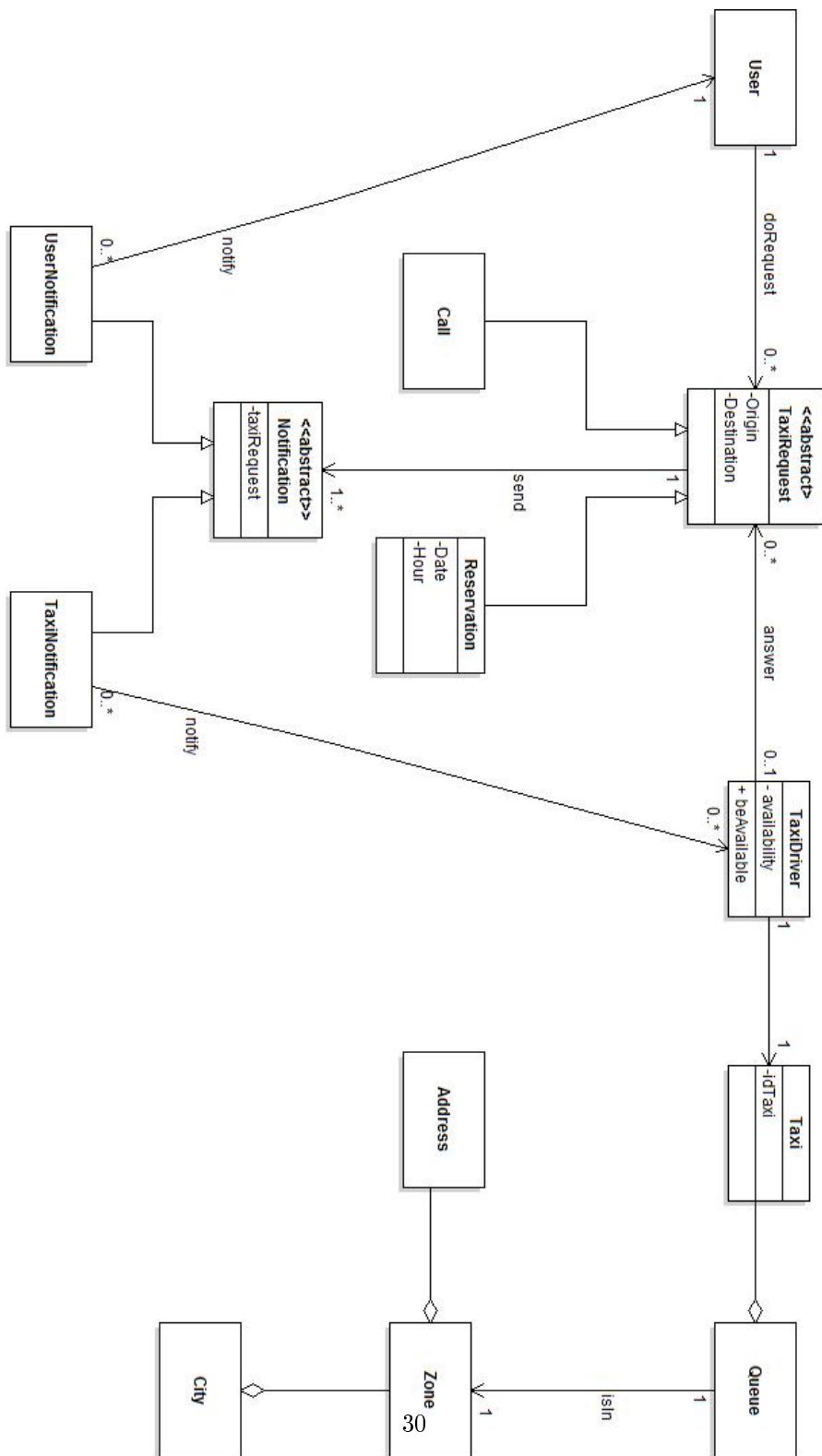
Figure 7: Class diagram

3.2.4 Scenarios

The following scenarios are derived from therequirements specified above, a sequence diagram is also tied to some scenarios to better represent the flow of events.

Scenario 1 It is 2 am and Marge is located in an area where the public transport service is over. She then decides to see if there is any taxi that will bring her home. She remembers that her friend told her about myTaxiService , an application that let you either call or book a taxi from anywhere in the city with just one click. Even late at night. She downloads the application, registers herself and fills the call form with her position. Homer, one of the few taxi drivers who like to do night shifts, is in the same area where Marge is located. Since he is the first drive waiting in the queue of that zone, he receives a notification of a call from the system that he accepts immediately. The application then notifies Marge that the taxi number 3587 will come to pick her up with a scheduled time of 5 minutes.
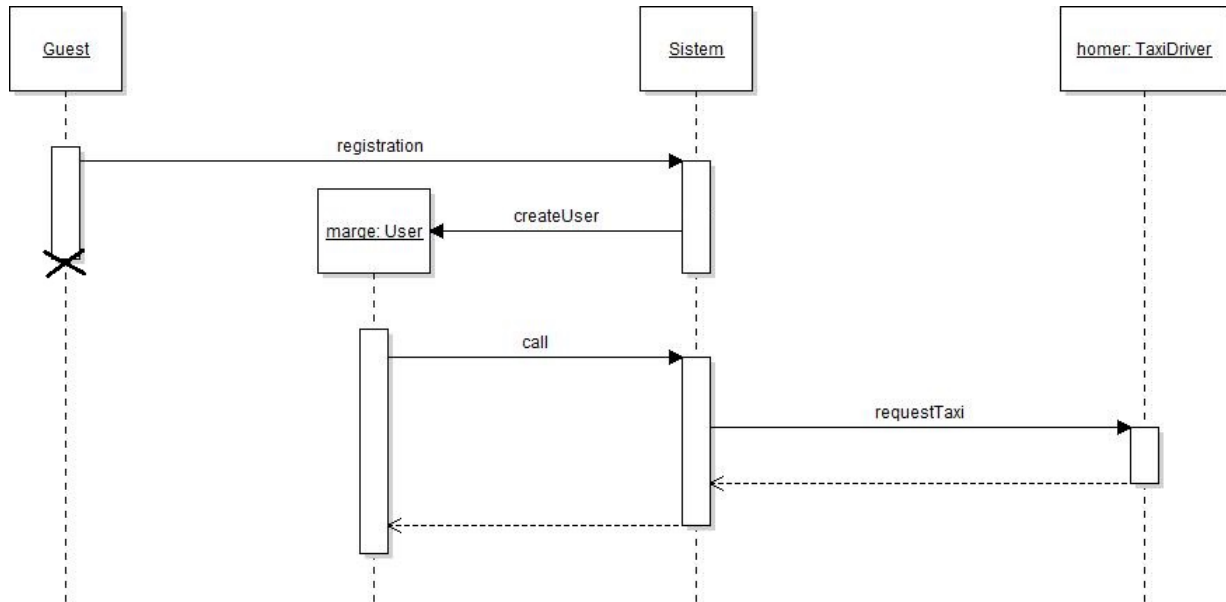
Figure 8: Sequence diagram related to the scenario number 1

Scenario 2 Holly is accustomed to using the application myTaxiService that he had downloaded a year ago. He often uses it to look for a taxi to bring him to his working place across the town. As usual, he opens the app on his phone and inserts the street where he is located. Benji' s phone receive the notification of the call but, because he is busy in the bathroom of the bar he

is used to visit, he refuses the call. The request is therefore transferred to the second taxi driver queuing in the same zone, Oliver, who accepts the call almost immediately, without pulling up the minute to respond. Holly after 1 minute and 10 seconds receives the notification, that a taxi with the number 6792 will come to pick up him under house within about 7 minutes, on his phone. Holly, breathing a sigh of relief, knows that he will not be late to his business meeting.

Scenario 3 Fred is waiting his girlfriend Wilma down her house, in the meanwhile he looks for a taxi with his myTaxiService app to go to a party with her. He opens the application, puts in the street where he is waiting Wilma and awaits. After 10 minutes, he receives a notification from the application that informs him that there are no taxis available. In fact tonight there is an important football match and thus is difficult for them to find an available taxi because they are all busy bringing other customers to their destinations. Fred is left with the only option of calling is best friend Barney and ask him for a lift.

Scenario 4 It's 9 am and Bart knows that in the afternoon he has to go to a swimming competition across the town. From his computer he goes on myTaxiService web page that is coach suggested him earlier. He signed up, logged in and filled the reservation form with the origin andt he destination of his ride as well as the date and the time he needs to leave: he chooses to set the time at 4 pm because his competition will start an hour later. He is confident because by receiving the confirmation of the reservation the service is guaranteed. About 10 minutes before the fixed hour he receives a notification that a taxi is on his way to Bart' s home to pick him up.

3.3 Performance requirements

To be considered acceptable the system should satisfy the following performance constraints:

- The application must support at least 100 users (either from browser or mobile app) simultaneously;

- At least 90% of all requests must reach a taxi within 5 seconds;

- At least 90% of the answers coming from drivers must reach the respective user within 5 seconds;

- Mobile app loading must not take more than 15 seconds;

- Web app loading must not take more than 20 seconds.

- The system should be available 24 hour a day 7 days a week;

Appendices

Alloy

The model we specified represents the relationship between antities inside a single zone, we assumed not too important to model also the relationship that links taxi to different zones. Also the model represent the situation in a given moment where a taxi either is available or he isn't, so we didn't make any distinction between direct calls and reservations, we referred to both of them as "Requests". The code is modeled with the entities "Queue" and "Node", these entitities are just placeholders and are meant only to give a more clear representation of the queue behaviour, in fact each node is associated with a driver and is this one that is related to the other entities of the model. Through this model we understood how to correctly manage the beahviour of the queue and how to link notifications with the physical parts of the system such as users and drivers.

```
sig User {}

sig TaxiDriver {
    available : one Boolean
    }

sig Request {
    accepted : one Boolean,
    user : one User,
    driver : one TaxiDriver
    }

sig TaxiNotification {
    driver : lone TaxiDriver
    }

sig UserNotification {
    user : lone User
    }

sig Node {
    current: one TaxiDriver,
    next: lone Node,
    }

sig Queue {
    root: lone Node
    }

enum Boolean {true, false}
```

//Request accepted contraint

```
fact AcceptedRequest {
    (all r : Request | no t : TaxiDriver | r.accepted = false && r.driver = t)
    ||
    (all r : Request | some t : TaxiDriver | r.accepted = true && r.driver = t)
    }
```

//Busy taxi contraint

```
fact NotAvailable {
    (all t : TaxiDriver | no r : Request | t.available = true && r.driver = t)
    ||
    (all t : TaxiDriver | some r : Request | t.available = false && r.driver = t)
    }
```

//A user in a certain moment has only one active call

```
fact OneUserCall {
    (all u : User | lone r : Request | r.user = u)
    }
```

//A Taxi can have only a single call active in a certain moment

```
fact OneTaxiCall {
    (all t : TaxiDriver | lone r : Request | r.driver = t)
    }
```

//Users cannot have more than one notification at a given time

```
fact OneNotificationOneUser {
    (all u : User | lone n : UserNotification | n.user = u)
    }
```

```
//Drivers cannot have more than one notification at a given time

fact OneTaxiOneNotification {
    (all t : TaxiDriver | lone n : TaxiNotification | n.driver = t)
    }

//Taxi Notifications are sent to the first taxi in the queue
fact FirstNotification {
    (all t : TaxiNotification | all q : Queue | t.driver = q.root.current)
    }
//User Notifications are sent only to User with an active request

fact NotificationOnlyActive {
    (all n : UserNotification | some r : Request | n.user = r.user && r.accepted = true )
    }
```

```
//Queue Management

fact QueueNotReflexive {
    (no n : Node | n = n.next)
    }

fact AllNodeBelongsToSomeQueue {
    (all n : Node | one q : Queue | n in q.root.*next)
    }

fact NextNotCyclic {
    (no n : Node | n in n.^next)
    }

fact AllTaxisInQueueAreAvailable {
    (all n : Node | no t : TaxiDriver | n.current = t && t.available = false)
    &&
    (all t : TaxiDriver | some n : Node | t.available = true <=> n.current = t)
    }

fact NodeTaxiIsUnique {
    (all n1 : Node | all n2 : Node | all t : TaxiDriver |
    (t in n1.current) and (t in n2.current) => (n1 = n2))
    }

assert NoTaxiAvailableIsOutsideQueue {
    (no t : TaxiDriver | all n : Node | n.current = t && t.available = false)
    }

assert NotificationOnlyToFirstTaxi {
    (no n : TaxiNotification | all q : Queue | q.root.current != n.driver)
    }
```

Generated Worlds

Here are given the worlds generated by the code above for three different cases:

World 1

```
pred show {
    #User = 2
    #TaxiNotification = 0
    #Queue = 1
    #TaxiDriver = 4
    #Request = 2
    }

run show for 6
```
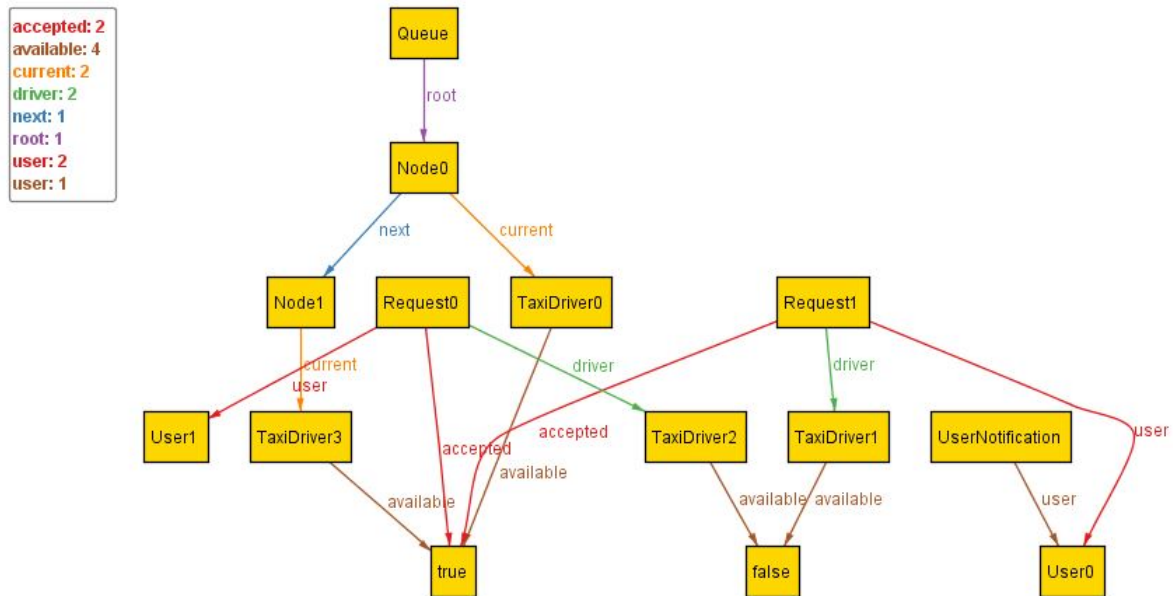

Figure 9: In this first world generated there are 2 Users and 4 Taxis available. Only 2 users do a request to take a taxi and so 2 taxis are marked as unavailable and thus they are not in any queue. The other two taxis however are available and are waiting in the queue of the zone.

World 2

```
pred show {
  #User = 2
  #TaxiNotification = 1
  #Queue = 1
  #TaxiDriver = 4
  #Request = 2
}

run show for 5
```
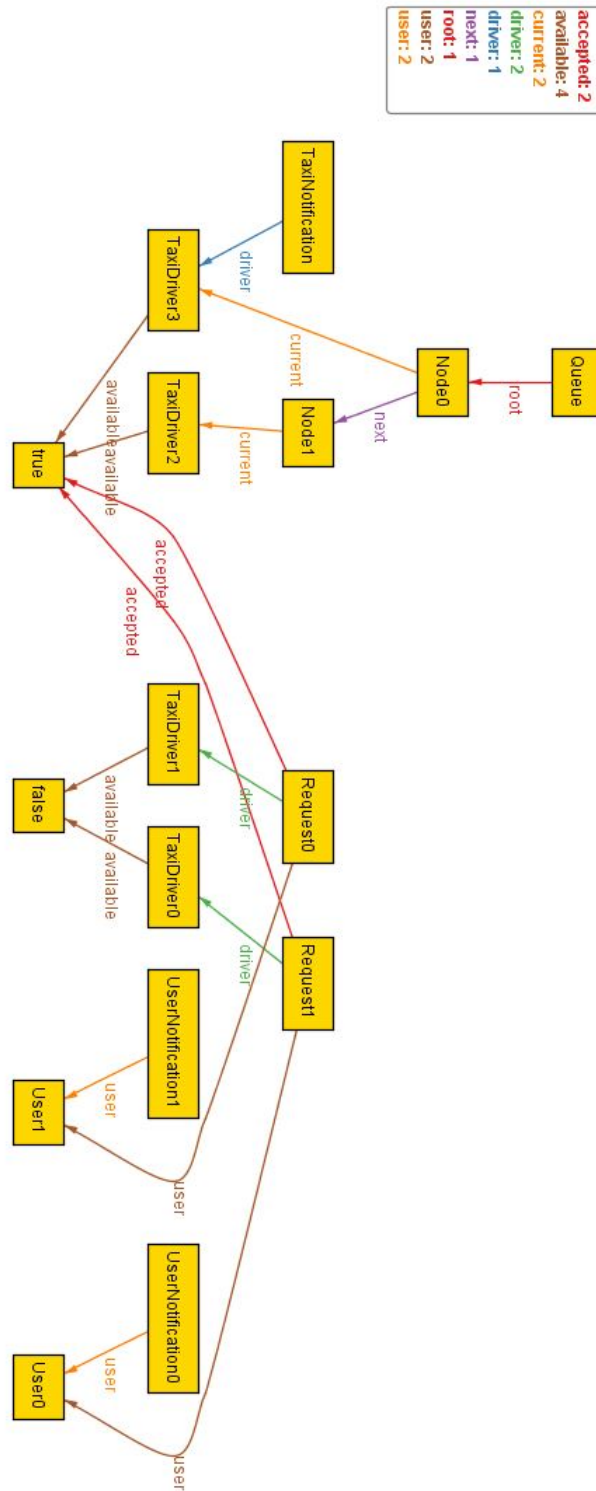
Figure 10: This world is similar to the first one but now there is a new notification of a call, the notification is forwarded only to the first taxi waiting in the queue.
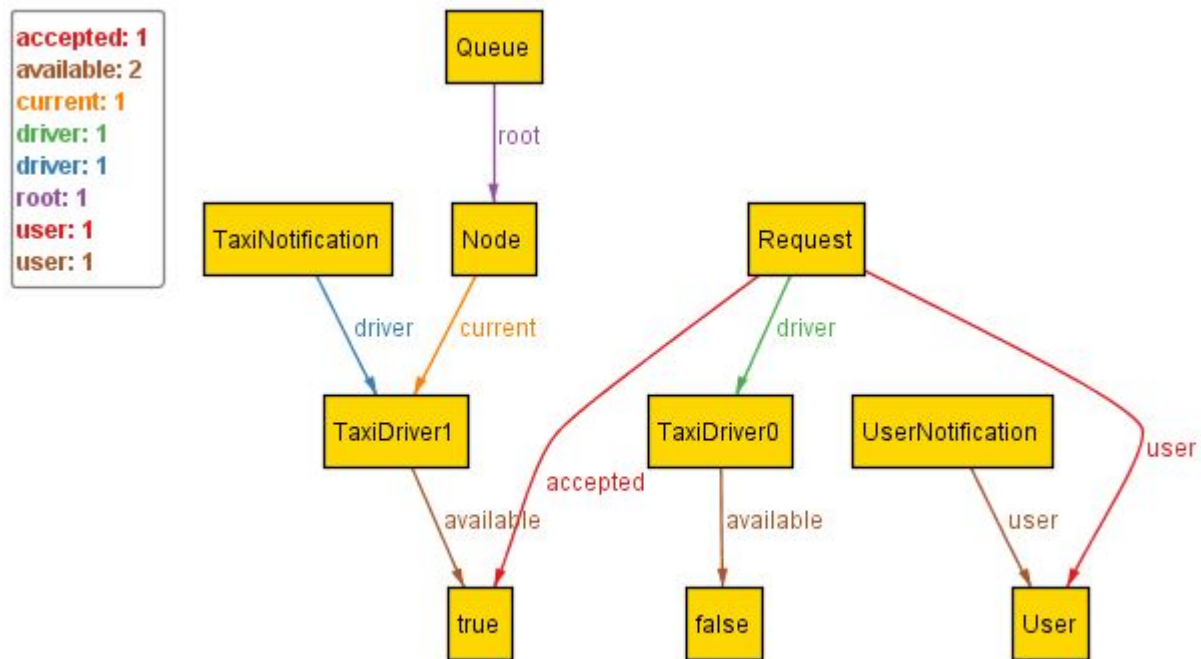
World 3

Figure 11: This is a more clear representation of the world 2 with a lesser number of entities

4 Supporting Informations

4.1 Glossary

- **GUI:** In computer science, a graphical user interface is a type of interface that allows users to interact with electronic devices through graphical icons and visual indicators

- **GPS:** The Global Positioning System 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

- **JVM:** A Java virtual machine is an abstract computing machine that enables a computer to run a Java program.

- **API:** In computer programming, an application programming interface (API) is a set of routines, protocols, and tools for building software applications;

- **Android:** Android is a mobile operating system currently developed by Google, based on the Linux kernel and designed primarily for touchscreen mobile devices such as smartphones and tablets;

- **IOs:** iOs is a mobile operating system created and developed by Apple Inc. and distributed exclusively for Apple hardware;

- **IEEE:** The Institute of Electrical and Electronics Engineers is a professional association. Its objectives are the educational and technical advancement of electrical and electronic engineering, telecommunications, computer engineering and allied disciplines;

- **Programming language:** A programming language is a formal constructed language designed to communicate instructions to a machine, particularly a computer;

- **Alloy:** Alloy is a language for describing structures and a tool for exploring them.

4.2 References

- **IEEE Std. 830:** standard for the redaction of the RASD document;

- **Alloy documentation:** from MIT, to draw alloy models;

- **LaTeX documentation:** to support the redaction of this document;

- **LyX documentation:** to redact this document;

- **Dell products catalog:** to define the type of machine that will run the server;

- **Microsoft server software catalog and documentation:** to define the software to be used inside the server;

- **Android documentation:** to define the interactions with android devices;

- **IOs documentation:** todefine the interaction with IOs devices.

4.3 Tools used

- **L$_Y$X:** used to write this documentation;

- **violetUML:** to make UML models;

- **Creately:** web app, used to make more UML models;

- **Balsamiq:** used to make mockups of the interfaces

- **Alloy analyzer:** used to write Alloy code andgenerate models

- **DropBox:** used to share materials between members of the group;

- **GitHub:** used to share materials between members of the group and to deliver this document.

4.4 Workload

The time spent to redact this document is approx:
Lorenzo Federico Porro: 41 hours
Annalisa Rotondaro: 41 hours