

myTaxiService
Requirements Analysis and Specification
Document

Belluschi Marco, Cerri Stefano, Di Febbo Francesco

November 5, 2015

Contents

1	Introduction	3
1.1	Purpose	3
1.2	Scope	3
1.3	Definitions, acronyms, and abbreviations	4
1.4	References	4
1.5	Overview	5
2	Overall Description	6
2.1	Product perspective	6
2.2	Product functions	6
2.3	User characteristics	6
2.4	Constraints	7
2.4.1	Regulatory policies	7
2.4.2	Hardware limitations	7
2.4.3	Interfaces to other applications	8
2.4.4	Parallel operation	8
2.4.5	High-order language requirements	8
2.4.6	Reliability requirements	8
2.4.7	Criticality of the application	8
2.4.8	Safety and security considerations	8
2.5	Assumptions and dependencies	9
2.5.1	Assumption	9
2.6	Apportioning of requirements	9
3	Specific Requirements	10
3.1	External interface requirements	10
3.1.1	User interfaces	10
3.1.2	API interfaces	17
3.1.3	Hardware interfaces	17
3.1.4	Software interfaces	17
3.1.5	Communication interfaces	18
3.2	Functional requirements	18
3.2.1	Login	18
3.2.2	Registration	18

3.2.3	Ride request	19
3.3	Performance Requirements	19
3.4	Design constraints	19
3.5	Software system attributes	19
3.6	Other requirements	20
A	Appendix	21
A.1	Actors	21
A.2	Identifying stakeholders	21
A.3	Alloy	21
A.4	Software and tool used	26
A.5	Hours of works	26

Chapter 1

Introduction

1.1 Purpose

This document represent the Requirement Analysis and Specication Document (RASD). The main goal of this document is to completely describe the system in terms of functional and non-functional requirements, analyse 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 Scope

The system described in this document is a taxi service for large cities. The main goals of the system are: 1) simplify the access of passangers to the service 2) guarantee a fair management of taxi queues. The system is composed by a web application, a mobile application and a web server.

There are three types of actors that can use the system: visitors, taxi drivers and passengers. Visitors have only two operations allowed: log in or sign in. Passengers can use both the web application and the mobile application to request a taxi. Taxi drivers use only the mobile application to modify their status and to confirm to the system that they are going to take care of a certain request from a certain passenger.

The system, when a passenger request a taxi, informs an available taxi driver (FIFO mode) about the current position of that passenger. At this time the taxi driver has two options:

- accept : the system sends a notification to the passenger with the estimated waiting time

- reject : the system searches for another available taxi driver

The system allows also a passenger to:

- reserve a taxi by specifying the origin and the destination of the ride
- share a taxi with others (if possible) by specifying all the rides that he/she wants to share. In this case the system defines the cost of the ride for each passenger

Besides the specific user interfaces for passengers and taxi drivers, the system offers also APIs to enable the development of additional services on top of the basic one.

1.3 Definitions, acronyms, and abbreviations

Definitions

- User: person that uses the service applications.
- Visitor: user that has not registered nor logged in.
- Registered user: user that has registered to the service.
- Passenger: passenger registered to the service.
- Taxi driver: taxi driver registered to the service.
- System: the union of software and hardware to be developed and implemented.

Acronyms

Abbreviations

1.4 References

- Software Engineering 2 Project AA 2015/2016: Project Description And Rules
- Software Engineering 2 Project AA 2015/2016: Assignments 1 and 2 (RASD and DD)
- Software Engineering 2 Project AA 2015/2016: RASD-meteocal-example1
- Software Engineering 2 Project AA 2015/2016: RASD-meteocal-example2

1.5 Overview

This document is essentially structured in three parts:

- Section 1: Introduction: it gives a description of the document and some basic information about the system. It also identifies the stakeholders and the actors involved.
- Section 2: Overall Description: it gives general information about the software and hardware product, constraints and assumptions.
- Section 3: Specific Requirements: this is the core of the document. It describes the functional and non-functional requirements combined with some scenarios. There is also a class diagram that gives an overall representation of the system.

Chapter 2

Overall Description

2.1 Product perspective

The system is composed by a web application, a mobile application and a web server. The web application runs on most common browsers, namely Chrome, Internet Explorer, Firefox, Safari. It needs a web server that supports PHP. The mobile application needs a platform supporting Android, iOS or Windows Phone. Both applications interact with a DBMS. Additional functionalities are provided through the use of APIs or interfaces, i.e. taxi reservation and the taxi sharing option.

2.2 Product functions

The system allows different kinds of user to perform different actions. In particular:

- Visitors can simply register or log in, thus becoming either a passenger or a taxi driver user.
- Passengers can request, reserve and share taxi rides.
- Taxi drivers can modify their availability status and respond (accept/refuse) to impending ride requests.

2.3 User characteristics

Registered users can be either passengers or taxi drivers.

The system wants to give both an easy way to interact, thus optimizing the taxi service. To do so, passengers must be able to install and use the mobile application, or use the web application. On the contrary, taxi drivers can only install and use the mobile app; besides, their cellphone must be provided with a GPS. All users must have access to the Internet.

2.4 Constraints

2.4.1 Regulatory policies

myTaxiService is a service provided by the public company responsible for public transportation in the city. The user, who reaches this service by web or mobile application, has to agree to License Agreement rather than Privacy policy and Terms of use at registration.

The user access and use of the services constitutes his/her agreement to be bound by these Terms, which establishes a contractual relationship between him/her and myTaxiService. If user does not agree to these Terms, he/she may not access or use the services. myTaxiService may immediately terminate these Terms or any services with respect to him/her, or generally cease offering or deny access to the Services or any portion thereof, at any time for any reason. myTaxiService collects the information provided by the user, for example when creating or making changes to services on demand, through contact with customer service or during other communications. This information may include: name, email, phone number, mailing address, profile picture, payment method, products required (for service delivery), delivery receipts and other information user choose to provide. The personal data will be used only to provide the services requested.

User is responsible for obtaining the data network access necessary to use the services. User mobile network's data and messaging rates and fees may apply if he/she accesses or uses the services from a wireless-enabled device. User is responsible for acquiring and updating compatible hardware or devices necessary to access and use the service and applications and any updates thereto.

myTaxiService does not guarantee that the services, or any portion thereof, will function on any particular hardware or devices. In addition, the services may be subject to malfunctions and delays inherent in the use of the Internet and electronic communications.

2.4.2 Hardware limitations

myTaxiService defines the minimum requirements for using web and mobile applications.

- *Web application*
Supported minimum version browsers: Chrome 25, Internet Explorer 10, Firefox 20, Safari 25. Other browsers may also work
Web access at the minimum speed of 1Mbps
- *Mobile application*
Operating system: Android, iOS, Windows Phone
Memory: 512MB RAM
Hard drive: 50MB of free space
GPS navigation system (only for taxi drivers)
Web access at the minimum speed of 1Mbps

2.4.3 Interfaces to other applications

myTaxiService is a stand-alone application and does not have to meet any interface to other applications. It is available on web and mobile stores.

2.4.4 Parallel operation

myTaxiService supports parallel operations cause of the nature of service. Many users can access to the service at same time thus system and database have to work with parallel requests.

2.4.5 High-order language requirements

myTaxiService requires the following high-order languages based on different platforms.

- *Web*
HTML 5 and CSS 3 standards.
- *Android*
Java 8
- *iOS*
Swift 2.0
- *Windows Phone*
C# 6.0

2.4.6 Reliability requirements

myTaxiService relies on network connections thus reliability issues are equivalent to performance issues. However, the applications should not corrupt server data as a result of its actions. The system has to guarantee whole-time availability.

2.4.7 Criticality of the application

myTaxiService relies on network systems and servers. Scheduled downtime is acceptable. This system requires a generator backup and redundant power in the event of failover.

2.4.8 Safety and security considerations

myTaxiService guarantees secure communications through AES encryption algorithms.

2.5 Assumptions and dependencies

2.5.1 Assumption

- Passenger requests a ride from web or mobile applications
- Passenger sets a correct meeting point
- Passenger sets a correct destination
- Taxi driver reaches the meeting point
- Taxi driver picks up the correct passenger
- Accurate taxi driver's locations are known by GPS
- Taxi driver reports correctly his availability
- The city is divided in taxi zones
- The taxi queue in a zone contains only taxi drivers available in that zone
- Taxi driver confirms or denies a passenger request call

2.6 Apportioning of requirements

Chapter 3

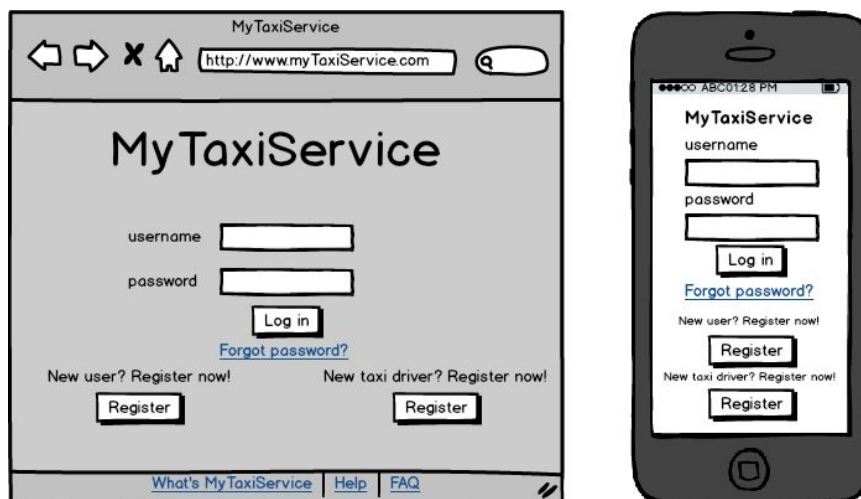
Specific Requirements

3.1 External interface requirements

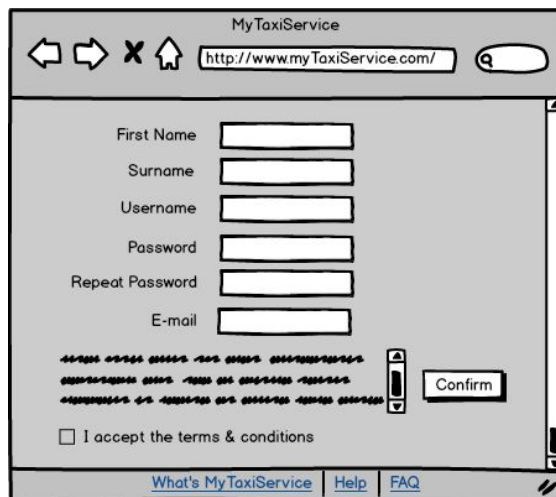
3.1.1 User interfaces

The interface of MyTaxiService can be for web application and mobile application. Here will be presented some of the most important pages and screens of MyTaxiService.

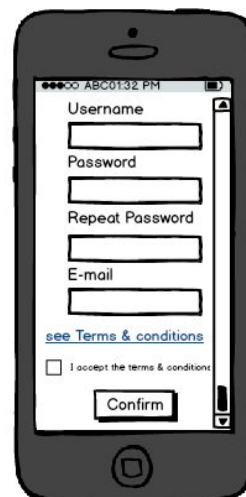
Log in: In the figure below is shown MyTaxiService's homepage



Registration passenger: View of the visitor that wants to register as a passenger



A desktop browser window titled "MyTaxiService" showing a registration form. The address bar displays "http://www.myTaxiService.com/". The form includes input fields for "First Name", "Surname", "Username", "Password", "Repeat Password", and "E-mail". Below these fields is a "Confirm" button. A checkbox labeled "I accept the terms & conditions" is positioned above the footer. The footer contains links for "What's MyTaxiService", "Help", and "FAQ".

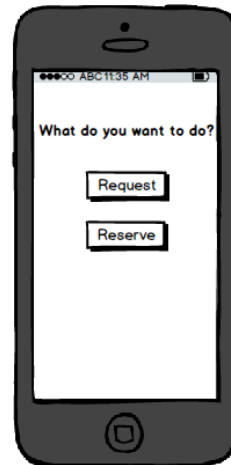
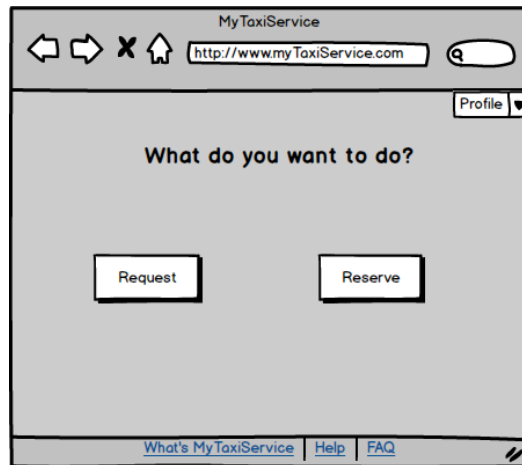


A mobile phone screen displaying the same registration form. The status bar at the top shows "ABC01:32 PM". The form fields are "Username", "Password", "Repeat Password", "E-mail", and a "Confirm" button. A link for "see Terms & conditions" and a checkbox "I accept the terms & conditions" are also visible.

Registration taxi Driver: View of the visitor that wants to register as a taxi driver

The illustration shows two devices displaying a registration form for 'MyTaxiService'. The desktop browser window on the left has a title bar with navigation icons and a search bar containing 'http://www.myTaxiService.com/'. The form fields are: First Name, Surname, License ID, Password, Repeat Password, and E-mail. Below these is a decorative horizontal line, a 'Confirm' button, and a checkbox labeled 'I accept the terms & conditions'. The footer contains links for 'What's MyTaxiService', 'Help', and 'FAQ'. The smartphone on the right displays the same form with a status bar at the top showing 'ABC0133 PM'. The form layout is adapted for the smaller screen, with the checkbox and 'Confirm' button positioned below the 'E-mail' field. The footer links are also present.

Passenger view: View of the passenger



Request a taxi: View of the passenger when he/she requests a taxi

The image shows two views of the 'Request a taxi' form. On the left is a desktop browser view, and on the right is a mobile phone view.

Desktop View:

- Browser address bar: <http://www.myTaxiService.com/>
- Page title: MyTaxiService
- Form title: Request a taxi
- Origin: use maps
- ☒ Share
- Destination: use maps
- Number of passenger:
- Confirm
- Text: Please remember to wait the taxi in your position
- Footer: [What's MyTaxiService](#) [Help](#) [FAQ](#)

Mobile View:

- Status bar: ABC0133 PM
- Form title: Request a taxi
- Origin: maps
- ☐ Share
- Number of passenger:
- Confirm
- Text: Please remember to wait the taxi in your position

Reserve a taxi: View of the passenger when he/she reserves a taxi

The image shows two views of the 'Reserve a taxi' form. On the left is a desktop browser view, and on the right is a mobile phone view.

Desktop View:

- Browser address bar: <http://www.myTaxiService.com/>
- Page title: MyTaxiService
- Form title: Reserve a taxi
- Origin: use maps
- Destination: use maps
- Time: hours/minute: ☐ Share
- Number of passenger:
- Text: Remember that you can reserve a taxi 2 hours before the ride
- Confirm
- Footer: [What's MyTaxiService](#) [Help](#) [FAQ](#)

Mobile View:

- Status bar: ABC0134 PM
- Form title: Reserve a taxi
- Origin: maps
- Destination: maps
- Time: hours/minute: ☐ Share
- Number of passenger:
- Text: Remember that you can reserve a taxi 2 hours before the ride
- Confirm

Taxi driver view: View of the taxi driver



Taxi driver notification: Notification that the taxi driver, chosen by the system, sees when a passenger request a ride.



Passenger notification : Notification that the passenger see when a taxi accept the ride



3.1.2 API interfaces

MyTaxiService uses Google Maps APIs in order to show to the passengers and to the taxi driver their position in the city. This API is continuous update and works for all the OS and browsers web supported by MyTaxiService. More information are available on the site: <https://developers.google.com/maps/>

3.1.3 Hardware interfaces

MyTaxiService doesn't support any hardware interfaces.

3.1.4 Software interfaces

- Database Management System (DBMS):
Name: MySQL
Version: 5.7
Source: <http://www.mysql.it/>
- Java Virtual Machine (JVM):
Name: JEE
Version: 7
Source: <http://www.oracle.com/technetwork/java/javasee/tech/index.html>

- Application server:
Name: Glassfish
Version: 4.1.1
Source: <https://glassfish.java.net/>

3.1.5 Communication interfaces

- Protocol: TCP Service: HTTPS Port : 443
- Protocol: TCP Service: HTTP Port : 80
- Protocol: TCP Service: DBMS Port : 9247

3.2 Functional requirements

3.2.1 Login

The visitor landing on myTaxiService site or app has the possibility to provide username (or email) and password to access the service, if he/she is already registered to the service. Submitted wrong informations will be notified by the system to the visitor. User who doesn't remember him/her login informations can ask the system to provide him/her the username and a new password. The visitor not registered can reach registration page.

Scenarios

1. Alan opens myTaxiService mobile app. He is not registered to the service indeed he wants to. He taps on "Register" in the bottom of screen. Now, providing his personal informations, he can register to the service.
2. Berry is a new user. He opens myTaxiService website but doesn't remember his password to access the service so he clicks on "Forgotten username and password?". He has to insert the mail provided at registration and if it is correct the system will send the username and a new password to the email.
3. Charlie is familiar with myTaxiService website and he wants to use also the mobile app. He knows that he can enter either username or email and password. He clicks on Login. It is successful.

3.2.2 Registration

The visitor, from the home page of the service both web or mobile application, can reach the registration page. The user can choose between two kinds of registration, one as a passenger the other as a taxi driver. Both registrations require providing personal informations but the latter needs more data and a valid taxi license. After the visitor agrees to Terms of use, Privacy policy,

and License Agreement, and his/her informations verified, he/she receives a confirmation email. The visitor, after the confirmation of email, is able to use the service.

Scenarios

1. Daisy is a taxi driver. myTaxiService has been just released. She wants to subscribe the service so she reaches the web site. She provides all her personal informations and her taxi license credentials. The system verifies her informations and sends her an email confirmation. She confirms her mail. Now Daisy is registered to the service as a taxi driver but she has to use her mobile application to be able to use the service. She downloads the application on his phone and, if it is provided with a GPS navigation system, use the service.
2. Ellie hasn't got a car but she has to commute to work. She opens the application on her phone and taps on "Register". She provides her personal information.....

3.2.3 Ride request

3.3 Performance Requirements

myTaxiService will perform 95% of the operations within 4 second; the total amount of the operations within 10 seconds. The system should ensure at least 2000 passengers connected and 500 taxi drivers connected.

3.4 Design constraints

myTaxiService wants to reach most of taxi drivers and passengers, requiring minimum specifications for devices. Taxi drivers, registered to the system, have to use their own devices provided with GPS navigation system to perform the service. Mobile applications have to offer backward compatibility.

3.5 Software system attributes

Reliability The mean time between failures (MTBF) shall exceed 3 months.

Availability In order to maintain the system up-to-date and secure, myTaxiServer schedules downtime periods where will be executed routine operations. The service should be available 99% of the time.

Security myTaxiService to ensure service availability and data protection use:

- AES cryptography algorithm on network operations
- Data are encrypted and stored in backup drives to prevent system failure
- Login authentication. Users, after the registration, have to confirm their e-mail with the security code sent to the e-mail write in the registration form
- SQL injection detection

Server architecture will be implemented separating data from application. Application server must be separated from database and from the web server. All architectures are divided by firewalls.

Maintainability To ensure an easy maintenance of the software, it must be well-documented and written following coding patterns.

Portability Web programming ensures a wide target of browser. Mobile applications instead, cause of different languages and devices, have to be written following coding patterns for easy portability. Availability of the service is ensured by hardware and software limitations in Section 2.4.2.

3.6 Other requirements

Appendix A

Appendix

A.1 Actors

There are three types of actors that use MyTaxiService:

- visitors: they can only log in or sign up
- passengers: they can request, reserve or share a taxi
- taxi drivers: they can accept/deny a ride and modify their status

A.2 Identifying stakeholders

The main stakeholder of the project is the government of a large city. The government, with the help of the transport council, decided to improve the actual taxi service with MyTaxiService. With MyTaxiService the stakeholders want to:

- symplify the access of passangers to the service
- guarantee a fair management of taxi queues
- give the possibility to the passengers to reserve a taxi
- give the possibility to a passenger to share a taxi with other passengers

A.3 Alloy

Here the Alloy code for create MyTaxiService model

```
// ALLOY CODE FOR MYTAXSERVICE
module MyTaxiService
// Defines Bool, True, False
open util/boolean
```

// Dates are expressed as the number of seconds from 1970-01-01

//SIGNATURES

```
sig Strings{}

abstract sig User {
  email : one Strings,
  emailConfirmed: one Bool,
  username: one Strings,
  password: one Strings,
  name: one Strings,
  surname: one Strings,
  address: lone Strings,
  telephoneNumber: lone Strings
}

sig Passenger extends User{
}

sig TaxiDriver extends User{
  licenseID: one Int,
  taxiPlateNumber: one Strings,
  taxiCode: one Int,
  numberOfSeats: one Int,
  status: one TaxiDriverStatus
}
{
  taxiCode > 0
  licenseID > 0
  numberOfSeats > 0
}

abstract sig TaxiDriverStatus {}
sig READY extends TaxiDriverStatus {}
sig BUSY extends TaxiDriverStatus {}
sig OFFLINE extends TaxiDriverStatus {}

sig Float{
}

sig Position {
  latitude: one Float,
  longitude: one Float,
  zone: one TaxiZone,
}

abstract sig RideStatus{}
sig ONGOING extends RideStatus {}
sig COMPLETED extends RideStatus {}

sig Ride {
  startPosition:one Position,
  endPosition:one Position,
  startDate:one Int,
  endDate:lone Int,
  status: one RideStatus,
  taxiDriver: one TaxiDriver,
  passengers: some Passenger,
  numOfPassengers: one Int,
  requests: some RideRequest
}
{
  #requests > 0
  startDate > 0
  startDate < endDate
  startPosition≠endPosition
  numOfPassengers ≤ taxiDriver.numberOfSeats
}
```

```

        #passengers ≤ numOfPassengers
        #endDate=0 iff status= ONGOING
        #endDate=1 iff status= COMPLETED
    }

    abstract sig RideRequestStatus{}
    sig PENDING extends RideRequestStatus {}
    sig ACCEPTED extends RideRequestStatus {}

    sig RideRequest{
        startPosition: one Position,
        endPosition: lone Position,
        requestDate: Int,
        ride: lone Ride,
        //passenger that requests the ride
        passenger: one Passenger,
        //additional passengers specified in the request
        numberOfPassengers: one Int,
        taxiDriver: lone TaxiDriver,
        status: one RideRequestStatus,
        isShared: one Bool
    }
    {
        endPosition ≠ startPosition
        isShared = False implies #endPosition= 0
        (#ride=0 or #taxiDriver=0) iff status = PENDING
        (#ride=1 and #taxiDriver=1) iff status = ACCEPTED
        requestDate>0
        numberOfPassengers > 0
    }

    sig ReserveRideRequest extends RideRequest{
        startDate: one Int
    }
    {
        #endPosition=1
    }

    sig TaxiZone{
        zoneId: Int,
        queue: one Queue,
        positions: set Position
    }

    sig Queue {
        zone: one TaxiZone,
        taxiDrivers: set TaxiDriver
    }

    // FACTS

    // users must not have same username or same e-mail
    fact UniqueUser{
    no u1, u2: User | (u1 ≠ u2 and (u1.username = u2.username or u1.email = u2.email))
    }

    // taxi drivers must not have same licenseID or same taxiCode
    fact UniqueTaxiDriver{
    no t1, t2: TaxiDriver | (t1 ≠ t2 and (t1.licenseID = t2.licenseID or t1.taxiCode=t2.taxiCode))
    }

    //zones must not have same zoneId
    fact UniqueTaxiZone {
        no z1, z2: TaxiZone | ( z1 ≠ z2 and z1.zoneId = z2.zoneId)
        queue = ~zone
    }

    //if a taxi driver has the status READY, he/she has to put into some queues

```



```

fact QueuesForReadyTaxiDriver{
    all t: TaxiDriver | ((t.status = READY) iff (some q: Queue | t in q.taxiDrivers))
}

//if a taxi is in a queue must be only in one of them
fact TaxiDriverInOnlyOneQueue {
    all t: TaxiDriver | (lone q: Queue | t in q.taxiDrivers)
}

//a taxi must be BUSY during the time of the ride
fact BusyDuringRide {
    all t: TaxiDriver, r: Ride | (r.taxiDriver = t and #endDate=0)
    implies (t.status= BUSY)
}

//a passenger cannot take two ride at the same time
fact noPassengerOverlapRide {
    all p: Passenger, r1, r2: Ride | (p in r1.passengers and p in r2.passengers and r1 ≠
r2)
    implies (r1.endDate < r2.startDate or r2.endDate < r1.startDate)
}

//a taxi driver cannot take two ride at the same time
fact noTaxiDriverOverlapRide {
    all t: TaxiDriver, r1, r2: Ride | (t in r1.taxiDriver and t in r2.taxiDriver and r1 ≠
r2)
    implies (r1.endDate < r2.startDate or r2.endDate < r1.startDate)
}

// only ACCEPTED Ride Request can have a Ride
fact RideWithOnlyAcceptedRideRequest{
    all r: Ride, rr: r.requests | rr.ride = r and rr.status = ACCEPTED
}

//a Ride Request cannot be in two different Ride
fact RideWithOnlyAcceptedRideRequest{
    no r1,r2 :Ride | r1≠r2 and (r1.requests=r2.requests)
}

//A ride that has more than one RideRequest must have only RideRequest shared
fact RideWithRequestsSharing {
    all r: Ride | (#r.requests>1)
    iff (all rr:r.requests | (rr.isShared = True ))
}

//if a ride refer to a ride request the taxi driver must be the same
fact taxiDriverUniqueRideReferRideRequest {
    all rr: RideRequest | rr.ride.taxiDriver = rr.taxiDriver
}

//if a ride refer to a ride request the passenger of the Ride Request must be in the passenger of the Ride
fact taxiDriverUniqueRideReferRideRequest {
    all rr: RideRequest , r:Ride | rr.ride = r implies rr.passenger in r.passengers
}

//a passenger cannot take another request when is in a ongoing ride
fact noPassengerOverlapRideRequest {
    all p: Passenger, r1, r2: RideRequest | (p = r1.passenger and p = r2.passenger and r1 ≠
r2)
    implies (r1.ride.endDate < r2.ride.startDate or r2.ride.endDate <
r1.ride.startDate)
}

//the request date of a request ride must be before the start date of a ride
fact reserveDateBeforeRide{
    all r: RideRequest | r.requestDate<r.ride.startDate
}

```

```

//the reserve date of a reserve ride must be before the start date of a ride and after the request date
fact reserveDateBeforeRide{
    all r: ReserveRideRequest | r.startDate<r.ride.startDate and r.startDate>r.requestDate
}

// ASSERTION

//all taxi in at maximum one queue
assert TaxiDriverInOneQueue {
    all t: TaxiDriver | (lone q: Queue | t in q.taxiDrivers)
}

check TaxiDriverInOneQueue
//No counterexample found. Assertion may be valid

//No another ride if the taxi driver is busy
assert noAnotherRideIfTaxiDriverBusy {
    all r1, r2: Ride | (r1.taxiDriver=r2.taxiDriver and r1 ≠ r2)
    implies (r1.endDate < r2.startDate or r2.endDate < r1.startDate)
}

check noAnotherRideIfTaxiDriverBusy
//No counterexample found. Assertion may be valid

//No another ride if the passenger is going in another ride
assert noAnotherRideIfPassengerIsGoingInAnotherRide {
    all r1, r2: Ride | (r1.passengers=r2.passengers and r1 ≠ r2)
    implies (r1.endDate < r2.startDate or r2.endDate < r1.startDate)
}

check noAnotherRideIfPassengerIsGoingInAnotherRide
//No counterexample found. Assertion may be valid

// PREDICATES

pred showNormalRequest(){
    #Passenger =1
    #Ride = 1
    #TaxiDriver = 1
}

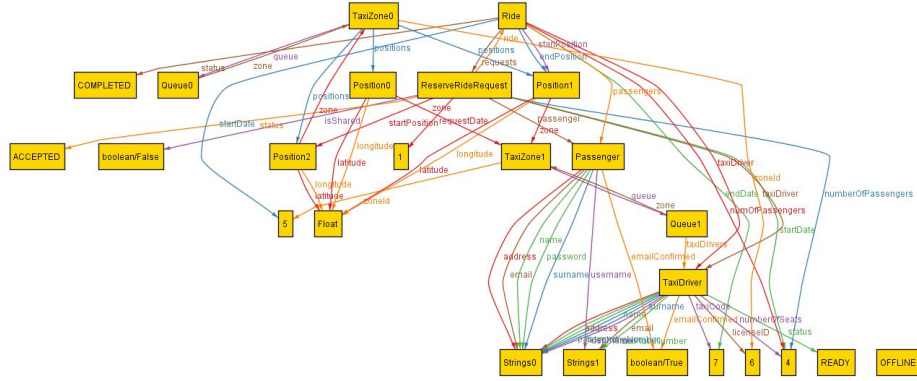
run showNormalRequest for 3

pred show(){
    #Passenger ≥ 2
    #Ride ≥ 2
    #TaxiDriver ≥ 2
    #{x: Ride | #x.requests>1} ≥1
    #{x: RideRequest | x.isShared = True} > 1
}

run show for 4

```

Here the world generated by the command (run showNormalRequest for 3):



A.4 Software and tool used

- LaTeX (<http://www.latex-project.org/>) : to redact and to format this document
- Balsamiq Mockups (<http://balsamiq.com/products/mockups/>): to create mockups
- Alloy Analyzer 4.2 (<http://alloy.mit.edu/alloy/>): to prove the consistency of the model

A.5 Hours of works