myTaxiService

Requirement Analysis and Specification Document

Monica Magoni 854091

Alberto Cibari 852689

November 6, 2015

Contents

1 Introduction

1.1 Purpose

This document represents the Requirements Analysis and Specification Document (RASD). Its aim is to capture all the functional and non-functional requirements that the system-to-be has to respect, in order to satisfy the stakeholders goals, under certain domain properties. This document also contains Use Case Diagrams, Sequence Diagrams and Class Diagrams that can be useful to better understand how the system is organized.
Further, this document is a valid basis for system testing, verification and validation and has also a contractual value.

1.2 Scope

The aim of our project is to optimize the taxi service of a large city. Passengers, once registered to the application, will have the possibility to request a taxi through either the web application or the mobile application. When a request from a passenger arrives, the system-to-be will be able to look for an available taxi (through a fair management of the taxi queues) and send to the passenger the code of the incoming taxi and the waiting time.
The management of taxi queues is explained as follows: if the first taxi of the queue is not available, the system will forward the request to the second taxi of the queue and put the first taxi in the last position of the queue. This process will continue until the system finds an available taxi, in order to satisfy the passenger's request.
Moreover, our system will also offer a mobile application for taxi drivers so that they will be able to share their availability and confirm a certain call.
Another functionality that our system will offer to users is the reservation of a taxi. Users will have the possibility to reserve a taxi in advance, by specifying the origin and the destination of their journey. The reservation must occur at least two hours before the ride and the system will inform the

taxi driver ten minutes before the scheduled time.
Our system will provide some programmatic interfaces for the development of additional functions, such as taxi sharing.
The system will offer interfaces (APIs) to enable the developement of new services. One of these services is the 'Taxi Sharing' option.

1.3 Goals

1. Let the users create a profile.
2. Let the users manage their profile logging in into the system.
3. Let the users request a taxi.
4. Let the users reserve a taxi.
5. Let the taxi drivers answer the taxi requests and reservations.
6. Handle the taxi in the city tracking their positions.

1.4 Actors

The actors of our system are:

- **Guest**: a person who is not registered in the system. He can not use the features of the system until he registers.
- **User**: a person who is already registered in the system that uses the application to request or reserve a taxi.
- **Taxi Driver**: a taxi driver registered in the system that uses the application to inform the system of the calls he will take care of and to share his availability.
- **Administrators**: managers of the web and mobile applications. They manage internally the system and try to fix errors in case of system fault.

1.5 Definitions, acronyms, abbreviations

1.5.1 Definitions

In order to avoid confusion, we want to specify the definition of some words that will be often used in our documentation of the project.

- PASSENGER: for passenger we mean a person, already registered in the system, who has requested or reserved a taxi either through the web or the mobile applications.
- USER: a person who is already registered in the system that can use the application to request or reserve a taxi.
- GUEST: a person who is not registered in the system.
- REQUEST: message sent by the user's application to the system in order to require a taxi or make a reservation.
- CALL: a task received by the taxi drivers after a user made a request.
- TAXI ZONE: for taxi zone we mean a zone that is approximately of 2km2. Each taxi zone has a queue of taxi associated.
- QUEUE: for queue we mean the list of taxi available at a specific moment in a certain zone.
- NOTIFICATION: for notification we mean a message that could be send by the system to the user (in order to notify the taxi identifier that will satisfy his request and the waiting time) or by the system to a specific taxi driver (in order to notify that a user has a request or a reservation to satisfy).
- RESERVATION: the action of the user to book a taxi followed by a request to the system.
- TAXI SHARING: the service that allows passengers to share a taxi.

1.5.2 Acronyms

- RASD: Requirements Analysis and Specification Document.
- GPS: Global Positioning System.
- DBMS: Database Management System.
- SD: Sequence Diagram.

1.6 Stakeholders

The stakeholders of our project are:

- Municipal administrators of the city: they gave us this project in order to optimize the taxi service of the city.
- Users: they are involved in our project because they will use our system-to-be, in particular they will have the possibility to choose between the web application or the mobile application.
- Taxi companies: our system will offer a mobile application for taxi drivers.
- System administrators: they are the managers of our system.
- Developers: they will develop the system.
- Testers: they will have to check if our system respects all the requirements that have been identified.

1.7 Reference documents

- AA 2015-2016 Software Engineering 2, Assignment 1 and 2.
- AA 2015-2016 Software Engineering 2, Project goal, schedule and rules.
- IEEE Std 830-1998: IEEE Recommended Practice for Software Requirements Specifications.

1.8 Overview

Our document is organized in four main parts:

- Introduction: in this section, we give an overview of the scope and goals of our system-to-be. We also identify the main actors that will be involved in our system and give the basic definitions of some words we will often use in this document.
- Overall description: in this part, we try to focus our attention on constraints and assumptions concerning our system-to-be and the world around it. This section also considers some possible future implementations that could be added to our system.
- Specific requirements: this section is the body of our document. All the specific requirements that our system need are described here and they are associated with different kinds of diagrams, in order to create a model of the real system.
- Bibliography: in this part we specify the documents or books we have referred to.

2 Overall Description

2.1 Product perspective

The product is a web based application that interacts with a back-end server that handles data and stores them into a DBMS. The client application is supposed to be developed in both web and mobile form. Here there are the main features that are included in 'myTaxiService':

- Multi-platform: the back-end system can be used to develop the mobile and web applications.
- User account: the system allows the user to create an account in the system and provide features of updating and viewing profiles.
- Number of users being supported by the system: though the number is precisely not mentioned, the system is able to support a large number of online users at a time.
- Taxi reservation: the system allows the reservation of a taxi, giving priority of using that taxi to the user who made the reservation, instead of those who made only a request.

2.2 User characteristics

We expect that our user's application will be installed by users who are looking for an easy way to request or reserve a taxi and have the possibility to check their reservations whenever they want. The system is developed in order to make the user feel the same experience on both the web and the mobile applications with a similar look.
We expect that our taxi drivers' application will be used by taxi drivers in order to simplify their job: our system will offer a simple way to accept or deny a request and to share taxi drivers' locations.

2.3 Product functions

The main function of the system is to allow users to make a request or a reservation for a taxi and taxi drivers to take care of a specific request. This is a list of the main actions that our system will provide:

- Users:
  - Sign up into the system.
  - Log into the system.
  - Request a taxi.
  - Reserve a taxi.
  - Delete requests or reservations.
  - Visualize information about requests or reservations.
  - Manage their account.
- Taxi drivers:
  - Sign up into the system.
  - Log into the system.
  - Receive information from the system about the city's area they have to cover.
  - Notify the system of their availability.
  - Receive requests and reservations from the system.
  - Accept or decline requests.
  - Manage their account.

2.3.1 Use Case diagram

From the list of actions described above, we can derive a Use Case Diagram of our system.

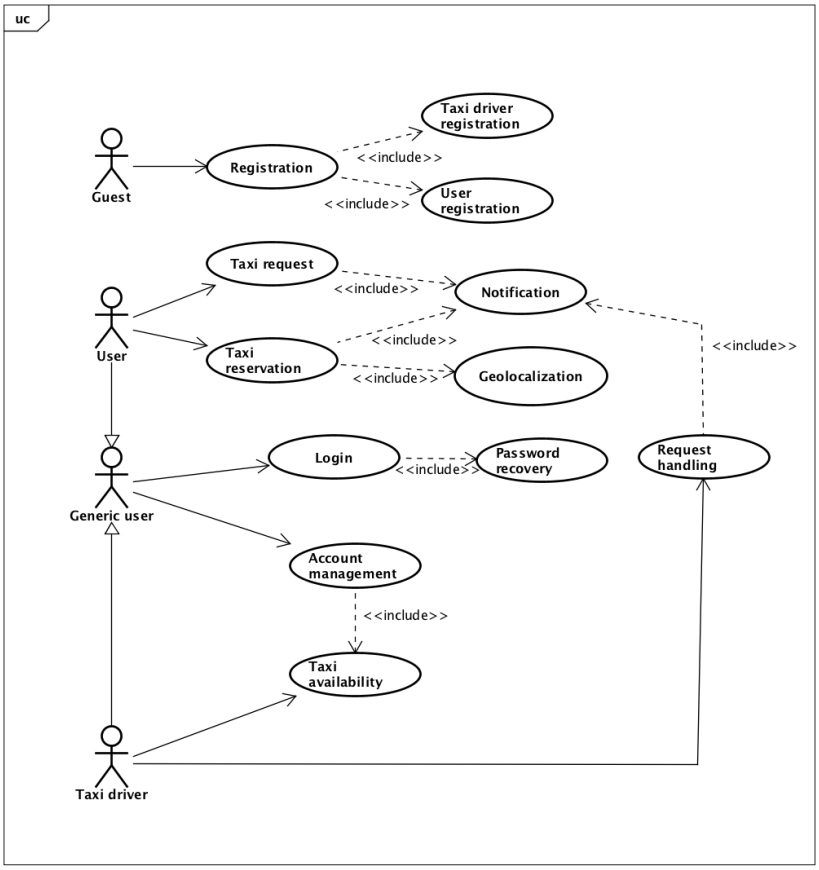


Figure 1: Use Case diagram

2.4 Assumptions and Dependencies

- The taxi queue of each area has a maximum length.
- If in a specific taxi queue there are too many taxi, the system must reschedule all the taxi's positions.

- If in a specific taxi queue there are too few taxi, the system must reschedule all the taxi's positions.
- Priority to reservations.
- Users can delete reservations within 10 minutes from the appointment.
- Taxi cars are provided of GPS and each of them is associated to a specific taxi identifier.
- Taxi drivers can delete accepted requests by contacting administrators, only if there is a problem to their car.
- The origin of every reservation must be within the boundaries of the city.
- The destination chosen by the passenger must be within 10Km from the city boundaries.
- If there are no taxi in a city zone for more than 30 minutes, the system will do a reschedule of the taxi's positions.
- Each time a user makes a reservation, he must specify the origin of the ride.
- If a taxi driver doesn't accept/deny a request received by the system within 1 minute, the system forwards the request to the next taxi driver in the queue.
- The taxi is able to reach the meeting location within 20 minutes from the request acceptance.

2.5 Constraints

2.5.1 Regulatory policies

- The system must work under the local laws and policies.

- The system must ask the user to acquire, store and process personal data.

2.5.2 Hardware limitations

- Taxi must have implemented a GPS device in order to track their car's location.
- Taxi driver and users must have a mobile device with the mobile application 'myTaxiService' installed. The minimum requirements for the application to be used are:
  - 3G connection
  - 512MB of RAM
  - 50MB of free disk space
- The web version of the application must work with the versions of IE, Chrome, Opera and Firefox released after 2010.

2.5.3 Performance requirements

- The system must reschedule the taxi positions within 5 minutes.
- The system must send the user's requests to the taxi drivers as soon as they are received.
- If a request arrives during the rescheduling process, the system must use the taxi configuration as it was before the reschedule.

2.5.4 Reliability requirements

- The system must have a minimum availability of 97%.
- The system can be shut down for maintenance only during night for at least 5 hours a month.

2.5.5 Security considerations

- The origin and destination of a user request must be kept secret.

2.5.6 Interfaces to other applications

- At the moment 'myTaxiService' has no interfaces with other applications. Possible implementations in the future.

2.6 Future possible implementation

- An additional service that could be integrated to our mobile application or web application is taxi sharing. This feature allows the passengers who wants to go to the same city area to share the taxi.

- Another useful function that could be added to our user's application is the possibility to pay via mobile after the taxi reaches his destination. In this way, the system will need to be interconnected to a bank's application.

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 Hardware Interfaces

The mobile version of the application for taxi drivers must provide a connection to the GPS device installed into the taxi.

3.1.2 Software Interfaces

Both the web application and the mobile application use the same web services to contact the system.

- DBMS:
  - MySQL v.5.7.8
- Application server:
  - JBoss Application Server v.7.1.1.Final
- Operating System (OS):
  - The web application must be able to run on any SO using any browser (IE, Chrome, Opera and Firefox) released after 2010.
  - The mobile applications must be available for devices running Android and iOS.

3.1.3 Communication Interfaces

The client application communicates with the back-end server via HTTPS (port 443). The back-end services connect to the DBMS on the default port.

3.2 System Functions

3.2.1 Guest Registration

Guests can subscribe to the system via mobile or web application. In both cases they have to fill a form with their personal data and authorize the recipient to use and process their personal details. If the guest accepts the conditions then he can complete the registration, otherwise it is canceled. Once a guest is registered, the system checks the consistency of the data inserted by the guest and if everything is correct a mail is sent to the new user.
The registration can also be done by taxi drivers and by a generic user of the service. Taxi drivers must complete the form with more specific data.

Use case

Actor	Unregistered user
Goal	Goal 1
Input condition	A guest chooses to create an account

Event Flow	1. On the registration page the guest chooses the user registration or the taxi driver registration; 2. The user fills the form with the information required; 3. The user accepts the terms and conditions for the service usage; 4. The user sends the form to the server clicking the "Register" button; 5. The user clicks the link on the confirmation e-mail sent by 'myTaxiService'.
Output condition	The system tells the user he is successfully registered and displays the user account page.
Exception	The user doesn't provide all the information required.

Functional requirements

- During the registration phase, guests must choose which kind of account to create: user or taxi driver. If the account is a user account then guests must provide:
  - Name and surname
  - Email address
  - Username
  - Password
  - Phone number

If the account is a taxi driver account, the guest must provide the same information of a user account, with in addiction:

  - Taxi Licence ID

- The username must be an alphabetic string.
- The password must be an alphanumeric string with at least a number and a uppercase character.
- The password's length must be between 8 and 16 characters.
- There can not be two accounts with the same e-mail and of the same type (user, taxi).
- The username is case-insensitive.
- Before sending the registration form, the guest will be asked to accept the terms and conditions contract.

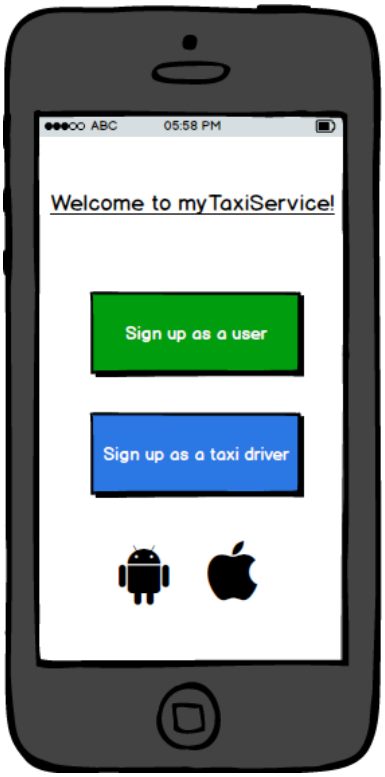


Figure 2: Taxi Registration Form

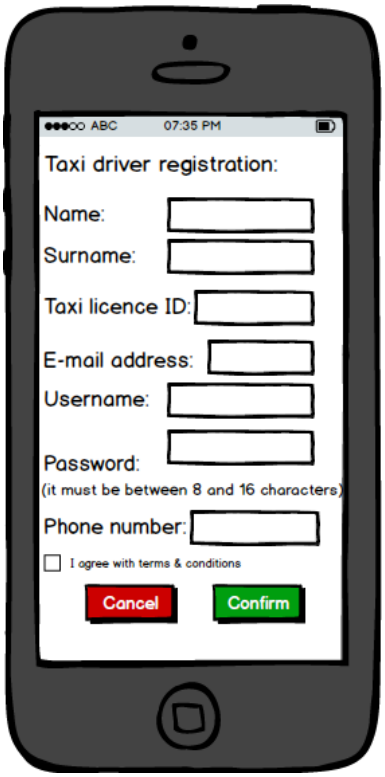


Figure 3: User registration Form

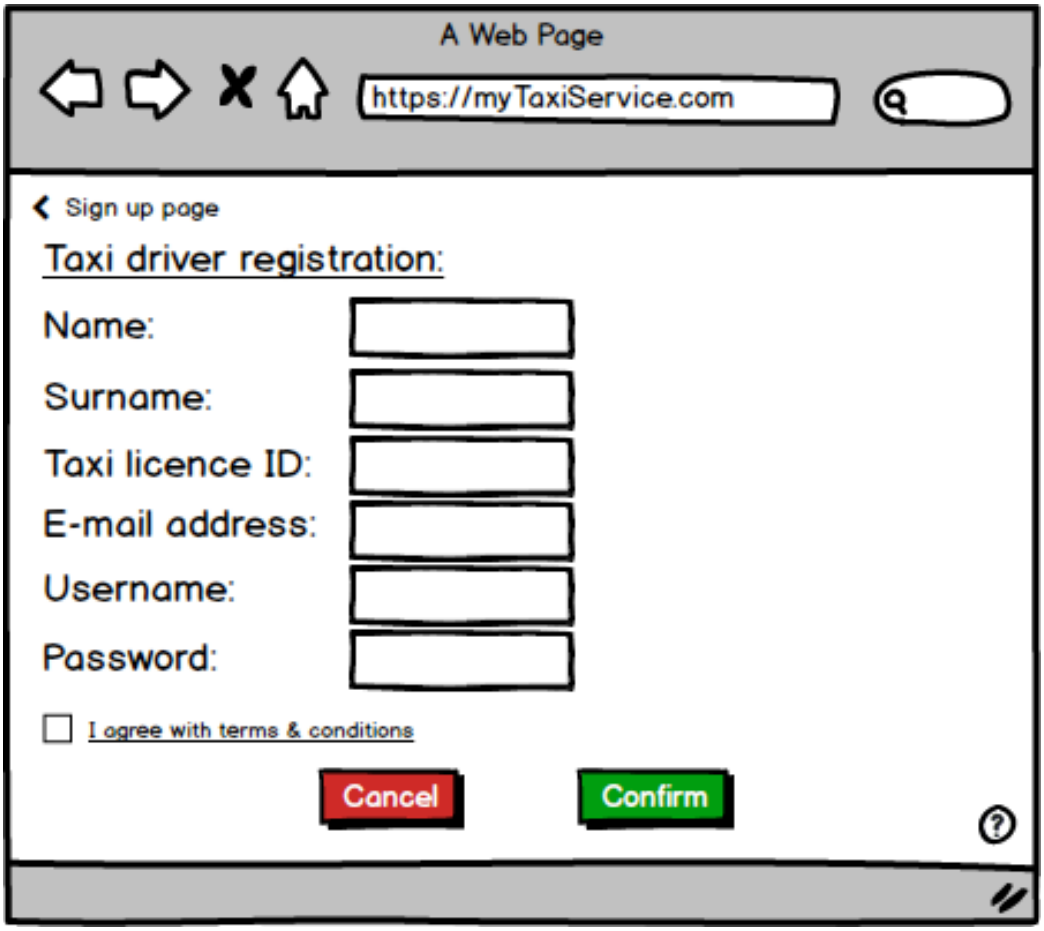


Figure 4: User registration form of the web application

3.2.2 User login/taxi driver login

Users must be logged in the system to use its services. During the login process users must choose the type of account (user or taxi), insert the username and the password. If everything is fine, then the users are redirected to their account page. In the 'user login' page, guests can be redirected to the 'registration' page.

Use case

Actor	Registered user

Goal	Goal 2
Input condition	A user chooses to login
Event Flow	1. On the main page of the application the user clicks the 'login' button; 2. The user fills the form presented with his credentials; 3. The user sends the form to the system for the credentials verification.
Output condition	The system tells the user he is successfully logged in and displays the user account page.
Exception	The user inserts wrong credentials (username/e-mail and password) and the system notifies him presenting the 'login' page again with an error message.

Functional requirements

- In the 'user field', users can insert the email or the username chosen during the registration phase.
- If a user makes a mistake while inserting the credentials, a suggestion field in the page pops up, and it will help the user to recover his credentials.
- After inserting the password for 3 times, the account freezes and the user receives an e-mail with the instructions to unfreeze the account.
- The 'username field' is case insensitive.

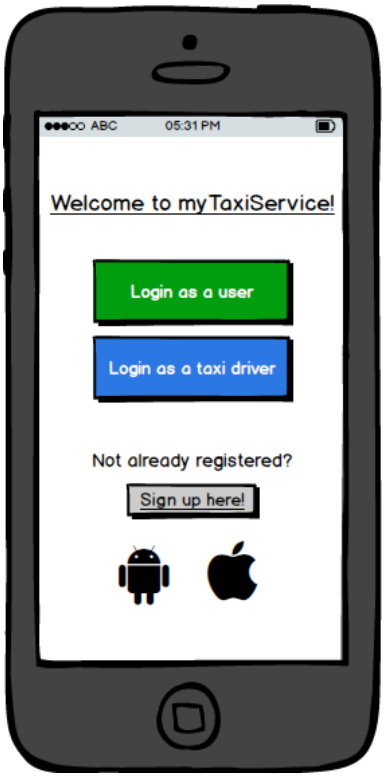


Figure 5: Login page of the mobile application

Figure 6: Login page of the web application

3.2.3 User home page/taxi driver home page

After the login process, users are redirected to their main page. Here they have the possibility to choose different actions to perform. For example, they can reset their password, change personal information or delete their account. If the user who reached this page is a generic user, he can choose the option of creating a request or a reservation. If the user is a taxi driver, there are others types of actions available: for instance, he can view the pending requests sent by the system and accept or decline them.

Functional requirements

- A generic user can:
  - Change password.
  - Change account information.
  - Make a taxi request.
  - Make a Reservation.
  - Delete requests or reservation.
  - Visualize all the requests/reservations done.
  - View the history of all the rides done.
- A taxi driver user can:
  - Change password.
  - Change account information.
  - Delete account.
  - View the history of all the rides done.
  - Activate/Deactivate taxi service.
  - View the pending request.
  - Track the taxi via GPS.
- During the 'password changing' phase, the old password should provided with the new one.
  - The new password must be an alphanumeric string with at least a number and a uppercase character.
  - The new password length must be between 8 and 16 characters.
  - The new password must be different from the old one.
- During the 'account deletion' phase, after the confirmation of the user, a mail is sent to him with a link that allows to deactivate the account.

- The deletion account function delete the account but keeps all the user information. The information are kept as history in the DB.
- Activate/Deactivate taxi service: the taxi driver use this function to tell the system to activate/deactivate the taxi service adding/removing him from the taxi queues in the city zones.
- A request received by a taxi driver is displayed in his main page for one minute, then it is passed to the next user in the city zone queue.

3.2.4 Activation/deactivation of the taxi services

When a taxi driver is visiting his home page in the application, he should be able to activate or deactivate his taxi service sending a notification to the system. When the system receives the notification from the user application about activating/deactivating the taxi system, it checks if the GPS installed in the car connected to the account is on. If there are no errors the taxi is put into a taxi zone queue/removed from the queue. And the system sends him a notification

Use case

Actor	Taxi Driver
Goal	Goal 2
Input condition	A taxi driver enters his account page and tries to change his status.
Event Flow	1. The taxi driver changes his status from his account page using the buttons in the account page.
Output condition	The system confirms the change sending a notification to the taxi driver.

Exception	• If the GPS on the car is not active, when the taxi driver tries to set is status to 'available', the application alerts the user to retry the operation.

Functional requirements

- The activation of the service can occur only if the GPS installed on the car is active, otherwise the taxi driver receives an alert on his application.
- If the system can not receive the GPS signal every 5 minutes, it sends a notification to the user.
- If the system can not receive the GPS signal for more than 20 minutes, the taxi service is deactivated.
- The taxi service can be deactivated only if the user isn't taking a request/reservation.

3.2.5 Request making

The user should be able to make a taxi request from the web application and the mobile application. If the request comes from the web application, the user has to tell the system the origin and the destination of the ride, otherwise they can be taken using the GPS of the mobile application. After the user makes the request, he receives a notification from the system with the identifier of the taxi and the possible arrival time of the taxi. Once the request is sent to the system, the system has to send it to the first taxi driver available, who can accept or deny the request.

Use case

Actor	Passenger
Goal	Goal 2
Input condition	A user chooses to make a taxi request from his account page
Event Flow	1. The user makes a request of a taxi inserting all the required information in the application page; 2. The user sends the request to the system using the "Send request" button.
Output condition	The user after a while receives the confirmation of the request.
Exception	• There are no available taxi in the area; • The user waits for the confirmation for more than 10 minutes; • When ten taxi drivers refuse the reservation, it is canceled.

Functional requirements

- The origin and the destination must be valid [View assumptions]. They must be within 10Km from the city boundaries.
- If the request is made using the web application, the system shall be provided with the passenger location inserted directly by the user.

- If the request is made using the mobile application, the system can use the GPS position to identify the user location.
- If the GPS is not available, the system tells the passenger to insert the correct origin location for the taxi ride.
- If the origin and/or the destination are not specified (neither from GPS nor from the user insertion), the user can not proceed to send the request.
- If a request is not accepted by a taxi driver, it is passed to the next driver in the queue of the city zone.

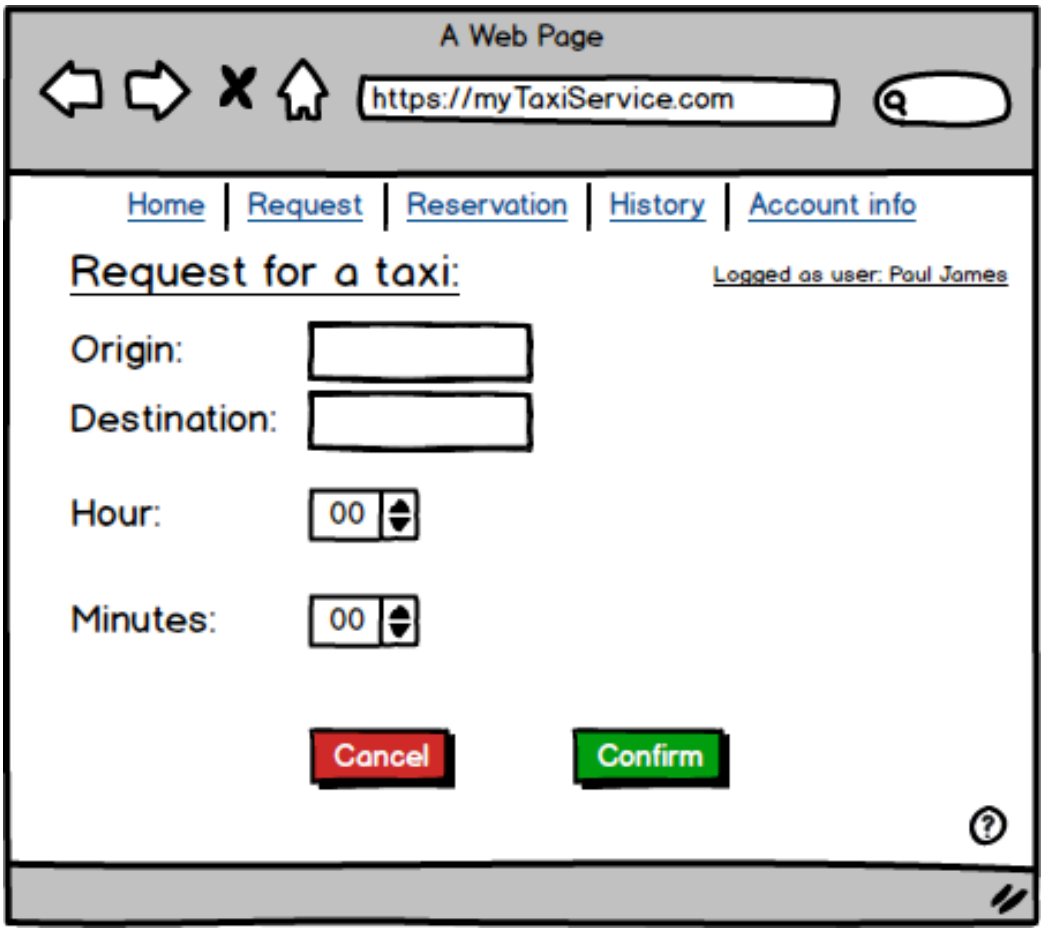


Figure 7: Request page of the web application

3.2.6 Reservation making

The user should be able to make a taxi request from the web application and the mobile application. The reservation must contain the origin and the destination of the ride. The user has to specify also the date and hour of the reservation. If the values for the reservation are valid, the user can send the request to the system. Then the user will receive a notification with the taxi identifier.

Use case

Actor	Passenger
Goal	Goal 4
Input condition	A user chooses to make a taxi reservation from his account page;
Event Flow	1. The user makes a request of a taxi inserting all the required information in the application page; 2. The user sends the reservation to the system using the "Make reservation" button.
Output condition	The user receives the confirmation of the reservation 10 minutes before the time chosen during the request making.
Exception	• There are no available taxi in the area; • When ten taxi drivers refuse the reservation, it is canceled.

Functional requirements

- The origin and the destination must be valid [View assumptions]. They must be within 10Km from the city boundaries.
- If the GPS is available the user can use the current position as origin of the ride.
- The date and the time of the reservation must be valid values.
- The system does not allow reservation after 24 hours from the current system time.
- The system sends a notification to the user 10 minutes before the hour chosen by the user for the ride. The notification contains the taxi identifier.

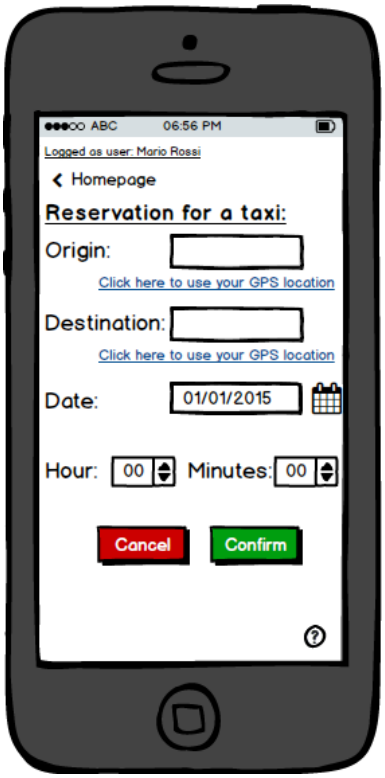


Figure 8: Reservation making page

3.2.7 Taxi driver notification handling

After a taxi driver activates the taxi service, he can receive a notification request for a taxi ride. He can accept or refuse the request. If the request is accepted, he has to go to the origin location indicated in the notification.

Functional requirements

- After the taxi driver receives a notification, if the system does not receive an answer within 1 minute, the notification is passed to the following driver in the taxi queue.
- If there are no available taxi drivers in a city zone, the system looks for an available taxi for a maximum of 10 times.
- If the system can not find an available taxi, it sends a notification of failure to the user.
- When the taxi reaches the origin zone, the system asks him to confirm the begin of the ride.
- When the taxi reaches the destination zone, the system asks him to confirm the end of the ride.
- After the taxi driver confirms the begin of a ride, the system sets his status to 'busy'.
- After the taxi driver confirms the end of a ride, the system sets his status to 'available'.
- When a ride is completed, the system stores the data of the ride into the DB.

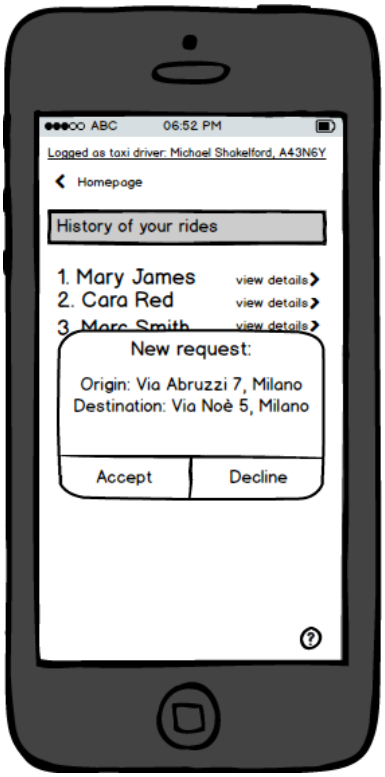


Figure 9: Notification request incoming

3.2.8 Zone queue handling

The system has to manage the queue of taxi in all the different city zones. It has to schedule the requests and send them to the right taxi.

Functional requirements

- The queues use the FIFO policy.
- When a taxi connects to the system and sets its status to 'available', the system insert it into a queue.
- The queue is identified by the GPS positions of the city zones: taxi are put in the queues according to their GPS location.

3.3 Scenarios

Scenario 1: User Registration

Bob is a tourist who just came in Gotham city. He discovers that the city has a system of taxi reservation using a mobile application. So he downloads the mobile app and opens it. The first page presented is the login page, but he hasn't an account yet. So he taps the registration button and he is redirected to the registration page. There he inserts his name and surname, his email and his username and password. He decides to leave the address and phone fields empty. Before sending the registration form, he accepts the terms and conditions contract, and sends the registration. After a while he receives a mail with a link for the confirmation of the account creation.

Scenario 2: Taxi driver registration

Alice is a taxi driver. She wants to participate in the city government project 'myTaxiService'. At her home she enters her browser in the main page of the myTaxiService web site. She isn't registered yet, so the click on the registration button that redirects her to the registration page. Here she enters her personal data, her username, password, and her taxi licence ID. Once filled the registration form, she accepts the terms and conditions of the service and submits the registration. A confirmation mail is sent to her with a link to complete the registration. She clicks it and she is redirected to the login page.

Scenario 3: User login

Charlie has to go to the airport for his holiday trip. At home on his web browser he enters the myTaxiService site. In the login page he enters his username and password, chosen during the registration phase. Once entered correctly the required values, he clicks the login button and visualizes his home page.

Scenario 4:User login error

Daniel wants to enter the myTaxiService site to view the log of all the requests he accepted. In the login page he enters his username and password, but unfortunately they are wrong. So he retries again but writes them wrong again. So he decides to try the recovery password system. In the login page he clicks on the 'Recovery' button. The system tells him it sents an e-mail with the new password auto-generated for entering the account. After Daniel receives the mail, he retries to enter into the system using the new password. After clicking onto the Login button he visualizes his homepage

Scenario 5: Activation of taxi service

Emily is beginning her work day as a taxi driver. She uses the myTaxiService system to handle the requests for a taxi. To enable her taxi to receive request she has to activate the service via myTaxiService application. Once entered her home page, after the login, she clicks the button for the activation. But she receives an alert regarding the GPS signal: "the system can't track the taxi GPS signal. The service can not be activated". So she turns on the GPS device in the car and retries the activation. This time the system finds the GPS signal and activate her service, putting her taxi in the queue of the city zone where che is in.

Scenario 6: Taxi request

It's a sunny day and Michael, who loves swimming, decides to go to the open air swimming-pool of his city. Unfortunately, the swimming-pool is quite far from his home and so he decides to make a taxi request. He remembers that some days ago, he registered to the 'myTaxiService' application. So, he logs in from his mobile application and from his home page, he clicks on the button 'make a taxi request'. The GPS on his mobile device is activated and so, after few seconds, he receives a notification containing the taxi identifier, that has been chosen by the system in order to satisfy his request, and the waiting time expected.

Scenario 7: Taxi reservation

Lorena is planning a visit to the local museum of the city. She loves planning every single detail of her trip and so she decides to make a taxi reservation in advance. She opens the web application of 'myTaxiService' and so she logs in. From the home page, she clicks on 'make a taxi reservation'; then, due to the fact that her GPS is not activated, she selects the origin and the destination of her ride. She has also to specify the date and the hour of the ride. After some seconds, she receives a notification from the application that confirms her reservation and indicates the taxi identifier that will take care of her journey.

Scenario 8: Taxi driver request acceptance

Alessia is a taxi driver, already registered in the system, and she is waiting for a request at the parking of city zone 2. She set her status as 'available' from her taxi driver application. Suddenly, she receives a notification on her device: it's a request from a user. She clicks on the button 'accept' and, after viewing where the user is located, she is ready to go to the origin of the ride.

3.4 Class Diagram

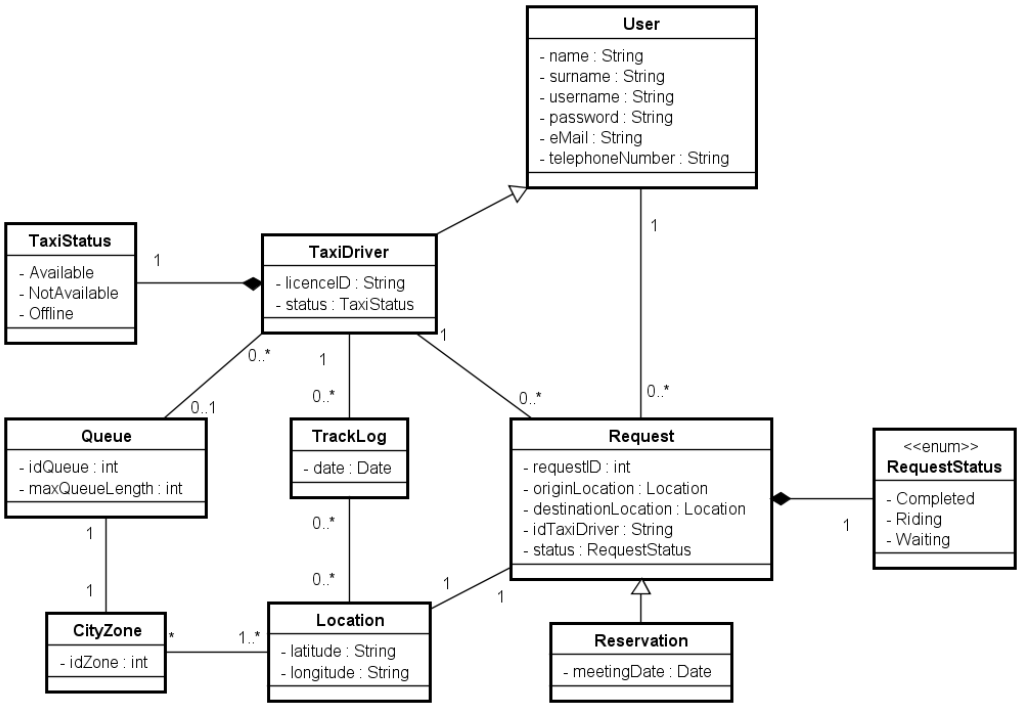


Figure 10: Class Diagram

3.5 Function sequence diagrams

User registration

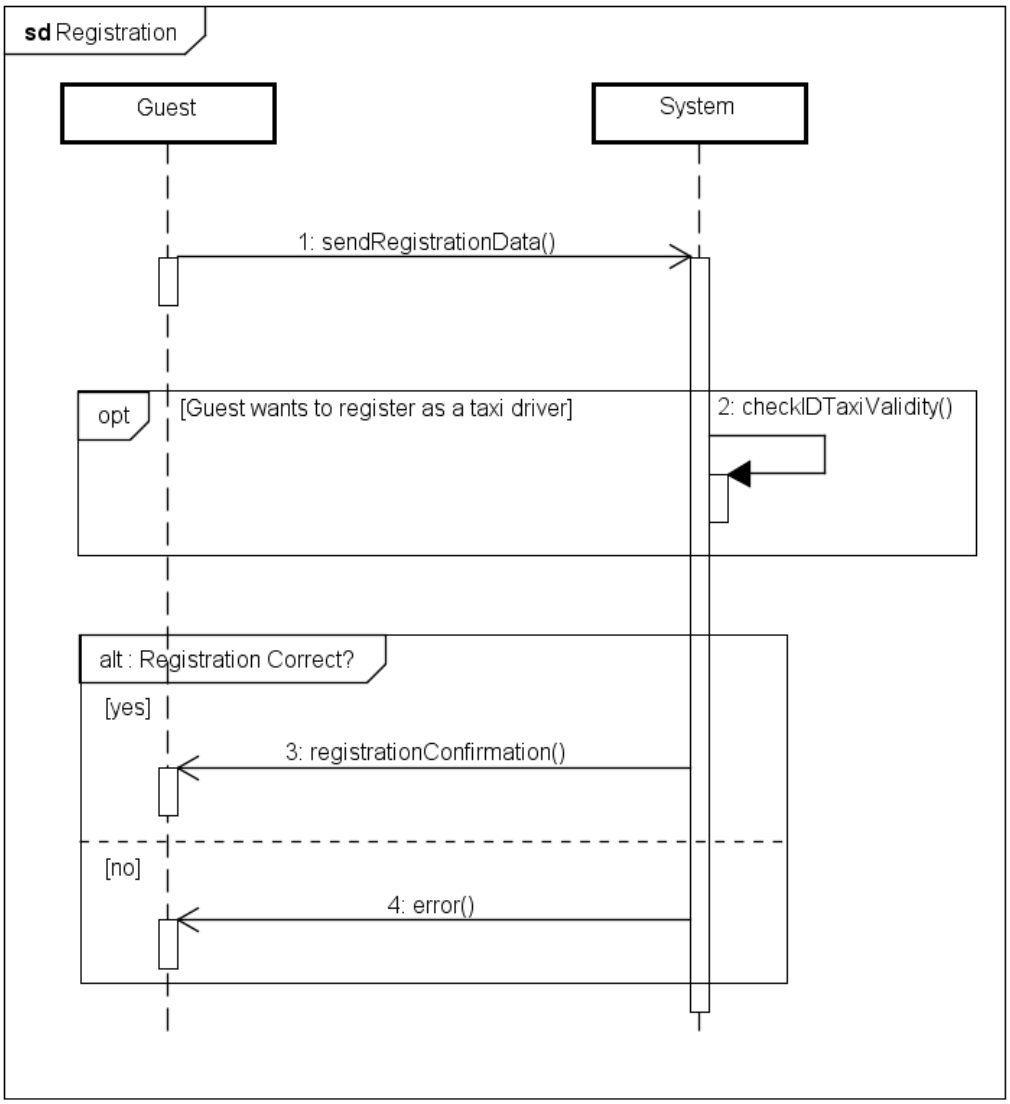


Figure 11: SD - User registration

User login

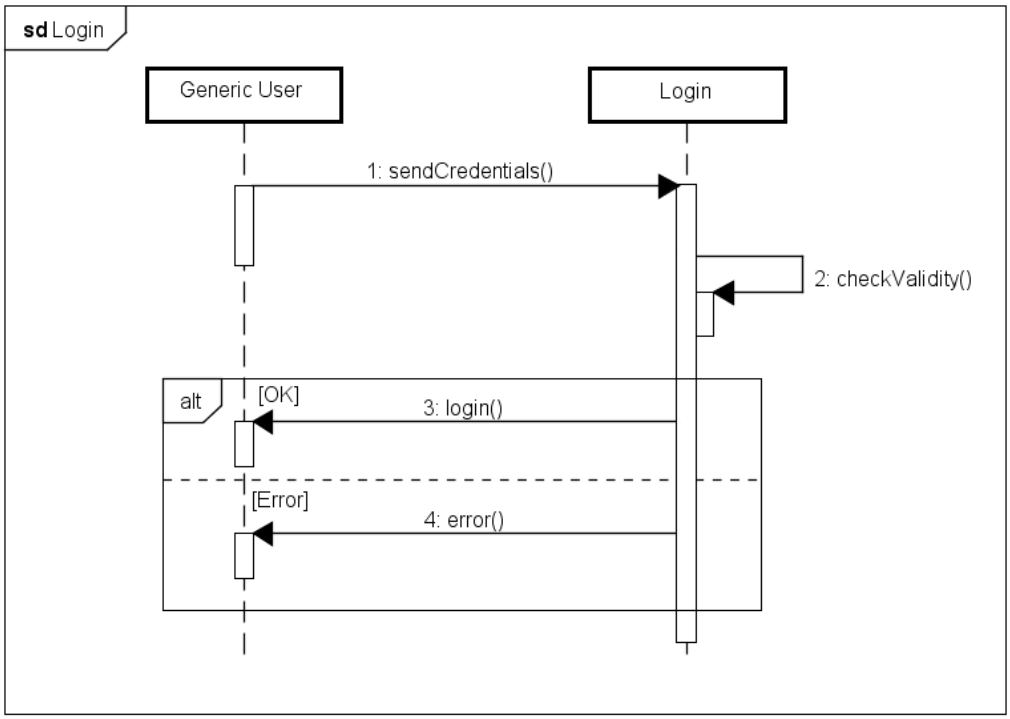


Figure 12: SD - User login

Taxi request

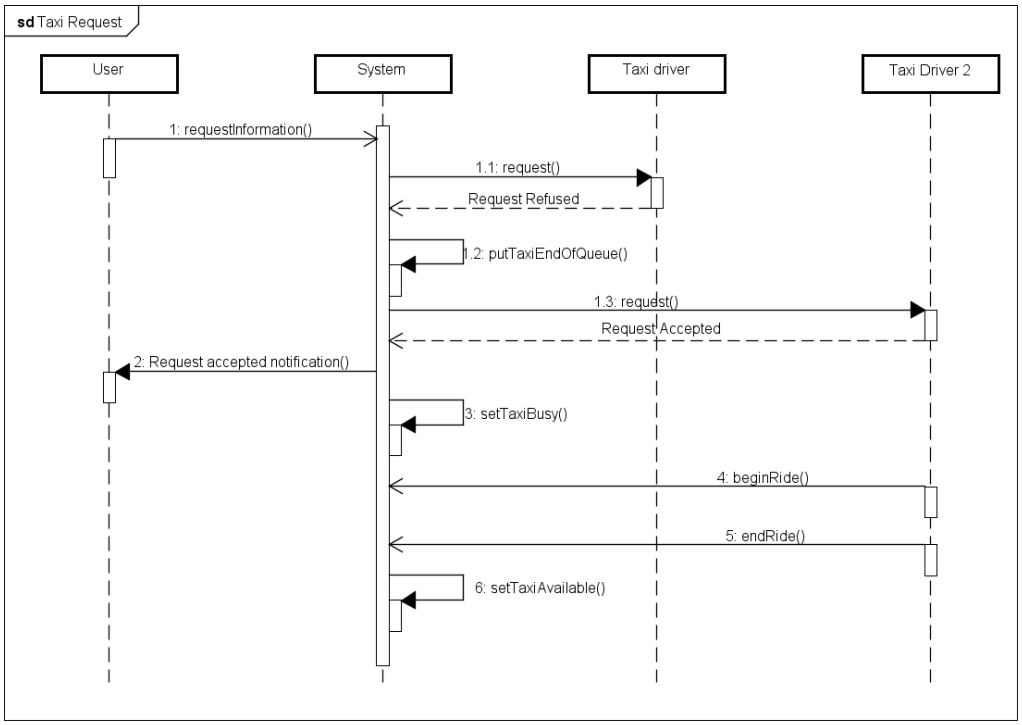


Figure 13: SD - Taxi request

Taxi reservation

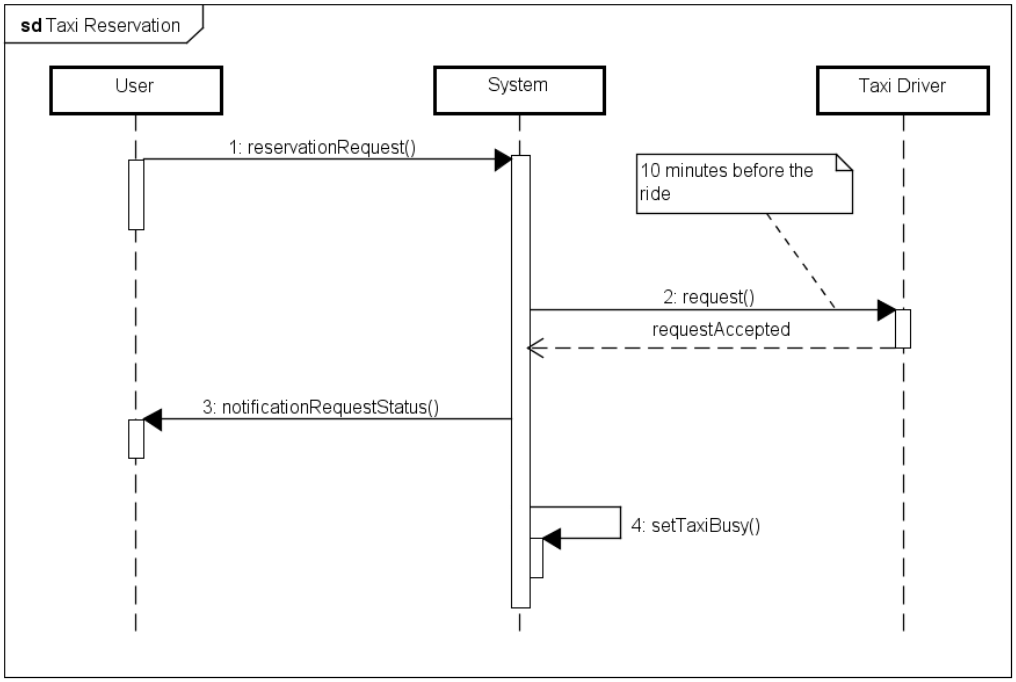


Figure 14: SD - Taxi reservation

3.6 Software system requirements

3.6.1 Availability

The system shall be available 24 hours per day, 365 days per year. It must have a minimum availability of 97

3.6.2 Portability

The software product can be installed in any operating system that supports the java virtual machine and its dependent components.

3.6.3 Maintainability

The database must be backed up 2 times per day, so the logs of taxi services are not lost due to malfunction.

3.6.4 Security

The software product must provide secure storage of the passwords of its users. This can be achieved by using any cryptographic techniques.

3.7 Alloy

```
/*
 * DEFINING SIGNATURES
 */

/*
 * General signatures
 */
sig Date{}
sig Location{}

/*
 * Enum that defines the possible status of a taxi driver
 * in the system
 */
enum TaxiStatus{
  AVAILABLE,
  NON_AVAILABLE,
  OFFLINE
}

/*
 *  Enum that defines the possible status of a request
```

```
23   * in the system
24   */
25  enum RequestStatus{
26    WAITING,
27    RUNNING,
28    COMPLETED
29  }
30
31  sig User{
32    request: set Request
33  }
34
35  sig TaxiDriver{
36    request: set Request,
37    status:  one TaxiStatus
38  }
39
40  sig CityZone{
41    borders: set Location,
42    queue: one Queue
43  }{
44    //Each zone must have at least three points (three locations)
45    #borders ≥ 3
46  }
47
48  /*
49   * Signature defining the queue associated to a CityZone
50   * and containing the list of the taxi in the zone
51   */
52  sig Queue{
53    maxQueueLength: one Int,
54    zone: one CityZone,
55    taxiList: set TaxiDriver
```

```
} {
  //The number of taxi must be lesser
  //than the maximum length of the queue
  #taxiList ≤ maxQueueLength

  //The maximum length of the queue must be equal or greater than 0
  maxQueueLength ≥ 0
}

sig TrackLog{
  taxiDriver: one TaxiDriver,
  date: one Date,
  location: set Location
}

/*
 * Signature defining a request with its associations
 * with the users and the locations
 */
sig Request{
  requestStatus: one RequestStatus,
  originLocation: one Location,
  private destinationLocation: lone Location,
  taxiDriver: lone TaxiDriver,
    user: one User
}{
  originLocation != destinationLocation
}

sig Reservation extends Request{
  destination: one Location,
} {
  originLocation != destination
```

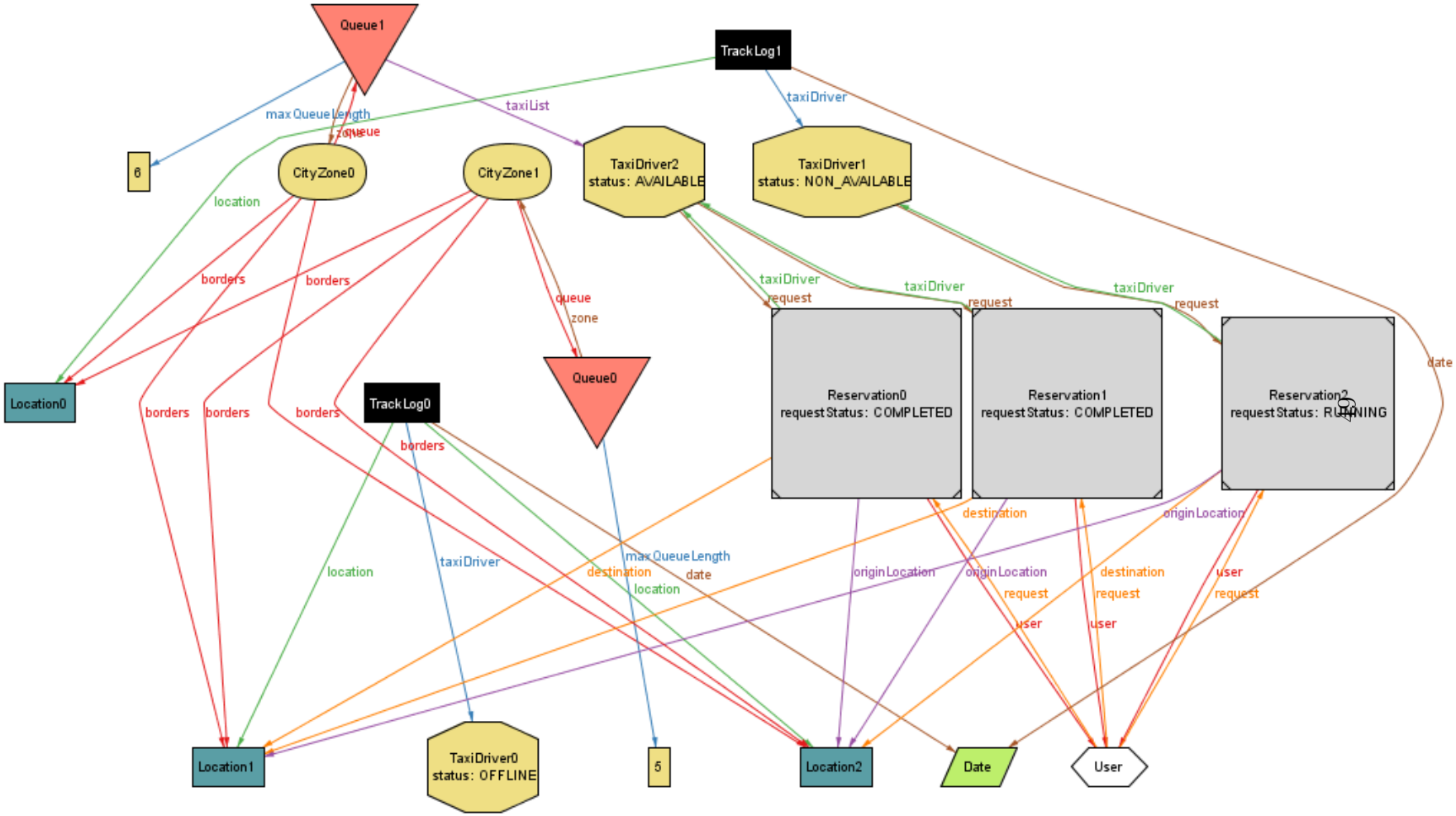
```
89  }
90
91
92  /*
93   * DEFINING FACTS
94   */
95
96  /*
97   * Every cityZone has only one queue and every queue refers
98   * to only one cityZone
99   */
100 fact AssociationQueueZone{
101   all q: Queue,  z: CityZone |
102     (z.queue = q) iff (q.zone = z)
103 }
104
105 /*
106  * A taxi driver is in only a queue
107  * if and only if his status is 'available'
108  */
109 fact AvailableTaxiStatus{
110   all t : TaxiDriver |
111     ( t.status = AVAILABLE) iff (
112       one q : Queue | t in q.taxiList
113     )
114 }
115
116 /*
117  * If a request is associated to a taxi driver, then the
118  * taxi driver must be associated to that request
119  */
120 fact AssociationTaxiRequest{
121   all t : TaxiDriver, r : Request |
```

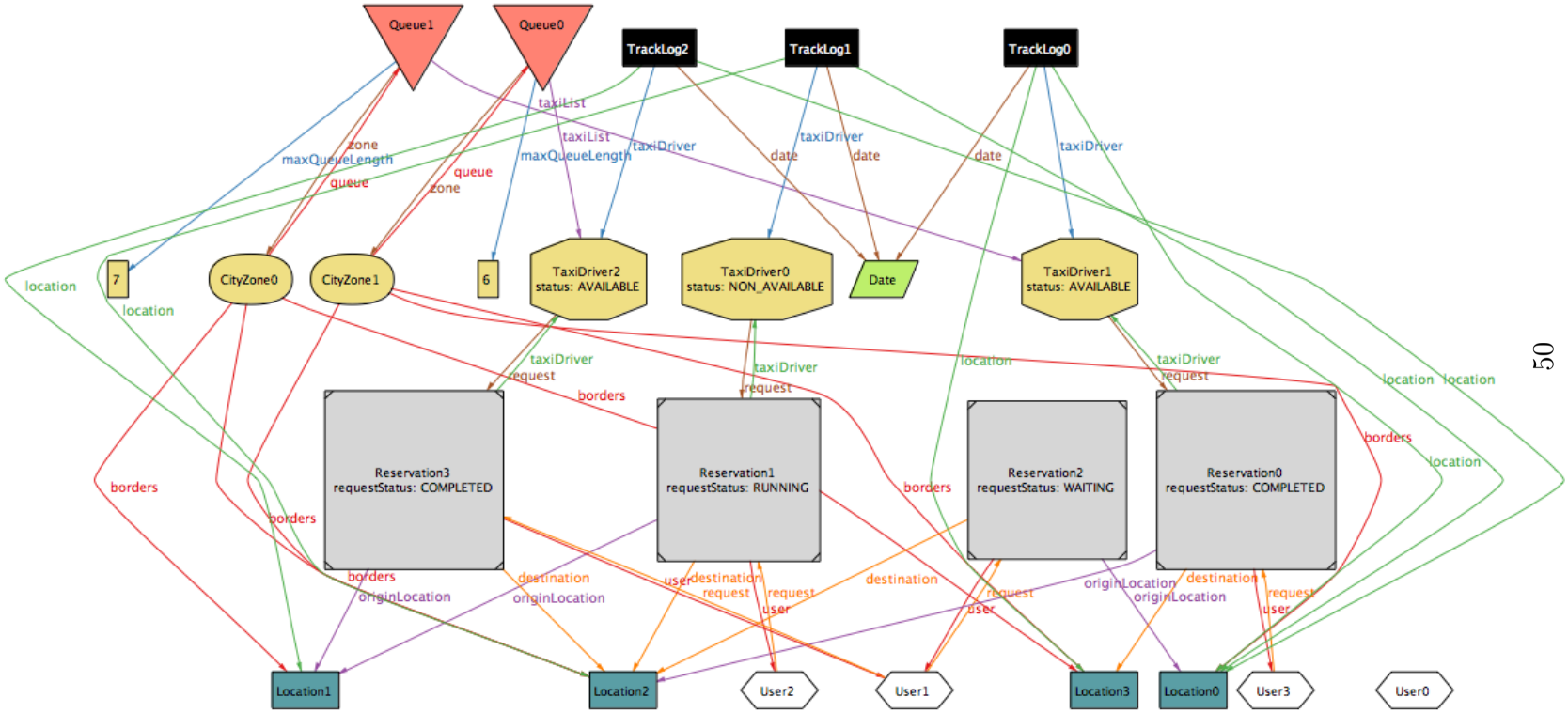
```
122     (r.taxiDriver = t) iff (r in t.request)
123 }
124
125 /*
126  * If a request is associated to a user, then the user
127  * must be associated to that request
128  */
129 fact AssociationUserRequest{
130   all u: User, r: Request |
131     (r.user = u) iff (r in u.request)
132 }
133
134 /*
135  * There can not exist two logs that refer
136  * to the same taxi driver and the same date
137  */
138 fact NoLogSameDayPerTaxiDriver{
139   no l1, l2: TrackLog |
140     (l1 != l2) and (l1.taxiDriver = l2.taxiDriver) and (l1.date =
    l2.date)
141 }
142
143 /*
144  * A user can be associated to only a request
145  * which status is 'running' or 'waiting'
146  */
147 fact OnlyOneWaitingOrRunningRequestPerUser{
148   no u: User,  r1, r2: Request |
149     (r1 != r2) and (r1.user = u) and (r2.user = u) and
150     (
151       (r1.requestStatus = WAITING or r1.requestStatus = RUNNING)
152       and (r2.requestStatus = WAITING or r2.requestStatus =
    RUNNING)
```

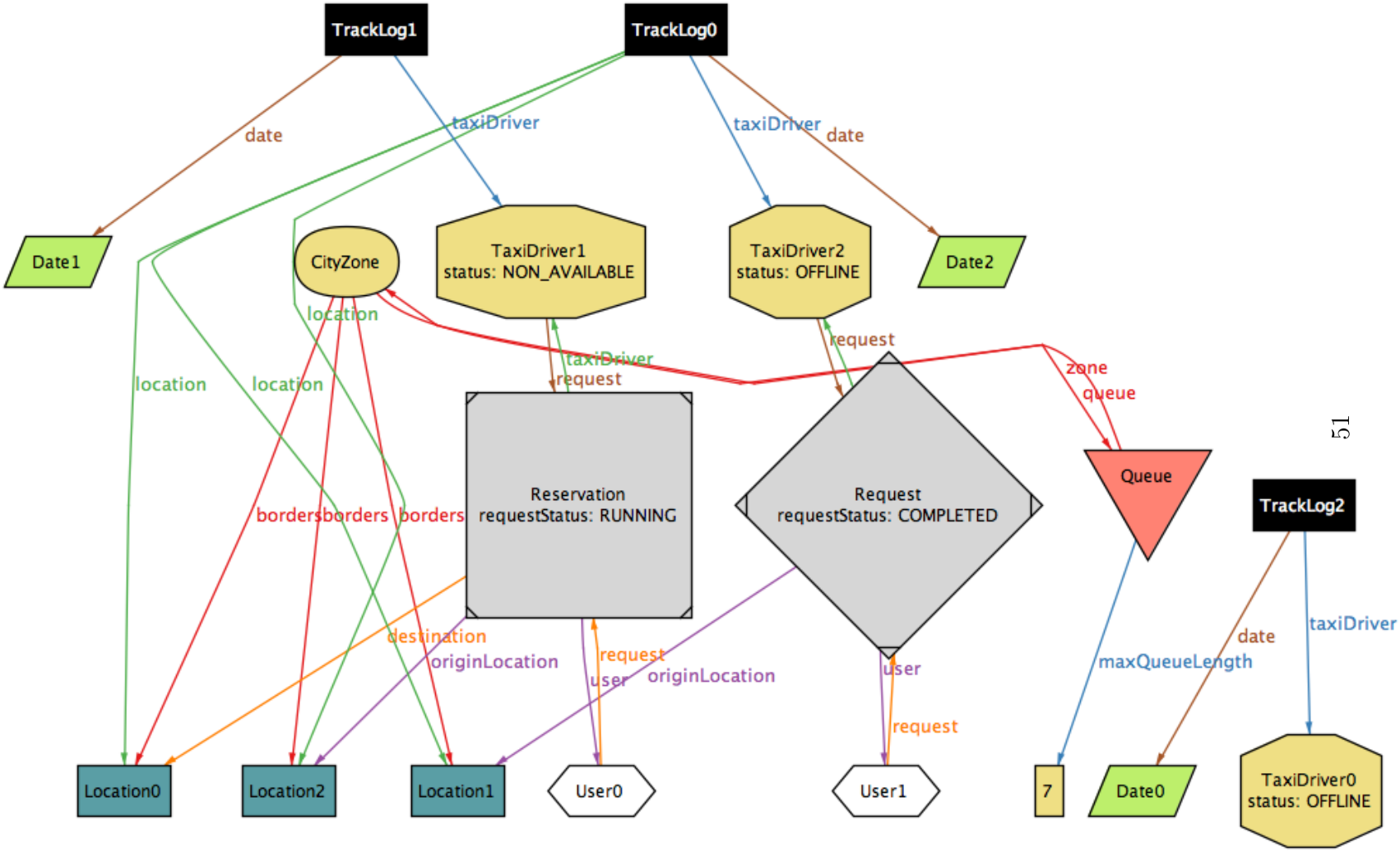
```
153     )
154 }
155
156 /*
157  * A taxi driver can be associated to only a request
158  * which status is 'running'
159  */
160 fact OnlyOneWaitingOrRunningRequestPerTaxi{
161   no t : TaxiDriver,  r1, r2: Request |
162     (r1 != r2) and (r1.taxiDriver = t) and (r2.taxiDriver =
    t) and
163     (r1.requestStatus = RUNNING and r2.requestStatus = RUNNING)
164 }
165
166 /*
167  * If the status of a request is 'running', it means that
168  * the status of the taxi driver who is associated
169  * to that request must be 'non available'.
170  * If the status of the taxi driver is 'non available'
171  * then there must be only one request which is 'running'
172  * assoicated to that driver
173 */
174 fact AvalabilityOfTaxiDriver{
175   all r : Request, t : TaxiDriver |
176     (r.requestStatus = RUNNING) and (r.taxiDriver = t)
177     implies (t.status = NON_AVAILABLE)
178   all t : TaxiDriver |
179     t.status = NON_AVAILABLE
180     implies (one r : Request |r.requestStatus = RUNNING and (r.taxiDriver =
    t))
181 }
182
183 /*
```

```
184  * If the status of a taxi driver is 'non available' or 'offline'
185  * it means that he isn't in any queue of the city zones.
186 */
187 fact taxiNotAvailableInQueues{
188   all t : TaxiDriver |
189     (t.status = NON_AVAILABLE or  t.status = OFFLINE)
190     implies ( no q: Queue |(t in q.taxiList)  )
191 }
192
193 /*
194  * If the status of a taxi driver is 'offline'
195  * it means that he is not associated to any request which
196  * status is 'running' or 'waiting'
197 */
198 fact taxiNotAvailableInQueues{
199   all t : TaxiDriver |
200     (t.status = OFFLINE) implies (
201       no r: Request |( t = r.taxiDriver) and
202       (r.requestStatus = RUNNING or r.requestStatus = WAITING)
203     )
204 }
205
206 /*
207  * If the status of a request is 'waiting',
208  * it means that there are no taxi drivers associated to.
209  * If the status of the request is 'running' or 'completed',
210  * there is only one taxi driver who is associated to.
211 */
212
213 fact TaxiRequestAssociationWaiting{
214   all r : Request | (r.requestStatus = WAITING)
215     implies (no t: TaxiDriver | r.taxiDriver = t and r in t.request)
216   all r : Request | (r.requestStatus = RUNNING
```

```
217     or r.requestStatus = COMPLETED)
218       implies (one t: TaxiDriver | r.taxiDriver = t and r in t.request)
219 }
220
221
222 pred showCompeteScenario{
223 }
224
225 run showCompeteScenario
```


Queue1
TrackLog1
taxiList
taxiDriver
maxQueueLength
queue
6
CityZone0
CityZone1
TaxiDriver2
status: AVAILABLE
TaxiDriver1
status: NON_AVAILABLE
location
borders
queue
zone
taxiDriver
request
date
Location0
Queue0
Reservation0
requestStatus: COMPLETED
Reservation1
requestStatus: COMPLETED
Reservation2
requestStatus: RUNNING
TrackLog0
destination
originLocation
user
Location1
TaxiDriver0
status: OFFLINE
5
Location2
Date
User


Queue1
Queue0
TrackLog2
TrackLog1
TrackLog0
taxiList
zone
maxQueueLength
queue
taxiDriver
date
location
7
CityZone0
CityZone1
6
TaxiDriver2
status: AVAILABLE
TaxiDriver0
status: NON_AVAILABLE
Date
TaxiDriver1
status: AVAILABLE
request
borders
Reservation3
requestStatus: COMPLETED
Reservation1
requestStatus: RUNNING
Reservation2
requestStatus: WAITING
Reservation0
requestStatus: COMPLETED
destination
originLocation
user
Location1
Location2
User2
User1
Location3
Location0
User3
User0


TrackLog1
TrackLog0
date
taxiDriver
taxiDriver
date
Date1
CityZone
TaxiDriver1
status: NON_AVAILABLE
TaxiDriver2
status: OFFLINE
Date2
location
location
location
taxiDriver
request
request
zone
queue
borders
borders
borders
Reservation
requestStatus: RUNNING
Request
requestStatus: COMPLETED
Queue
TrackLog2
destination
originLocation
request
user
originLocation
user
request
maxQueueLength
date
taxiDriver
Location0
Location2
Location1
User0
User1
7
Date0
TaxiDriver0
status: OFFLINE

4 Bibliography

[1] Software Engineering 2 project, *AA 2015-2016 Software Engineering 2, Assignment 1 and 2*

[2] Software Engineering 2 project , *AA 2015-2016 Software Engineering 2, Project goal, schedule and rules*

[3] IEEE Std 830-1998, *IEEE Recommended Practice for Software Requirements Specifications*

[4] Alloy, *http://alloy.mit.edu/alloy/documentation.html*

Appendix

Tools used

- ShareLaTeX: to write this document. [1]
- GitHub: to share our work. [2]
- Astah: to create Sequence Diagrams, UML Diagrams and Use Case models. [3]
- Balsamiq: to create mockups. [4]
- Alloy Analyzer: to simulate specifications and perform verification. [5]

Hours of work

- Alberto Cibari: 39 hours
- Monica Magoni: 39 hours

[1] `https://it.sharelatex.com`
[2] `https://github.com`
[3] `http://astah.net`
[4] `https://balsamiq.com`
[5] `http://alloy.mit.edu/alloy/`