Politecnico di Milano
A.A. 2015-2016

Software Engineering 2: myTaxiService

Requirements **A**nalysis and **S**pecifications **D**ocument

version 2.0

Matteo Bulloni (852676), Marco Cannici (852527)

4 December 2015

Contents

1 Introduction

1.1 Description of the given problem

We are going to project an application called myTaxiService, for both web and mobile, which has been requested by the government of a large city to optimize its taxi service. The main aim of this application is to allow people to call a taxi for a ride without actually having to make a phone call, but sending a request through the application instead, and to automate the handling and transmission of the requests from clients to drivers in such a way to make the service fast and efficient. The users of the application, after registering to the service, will be able to request an immediately needed ride, receiving fast notification of the incoming taxi code and estimated waiting time, or to make a reservation in advance for a ride, specifying its time, date, starting and ending point. These requests will be handled by a system which guarantees a fair management of taxi queues, obtained dividing the city into zones and handling requests depending on where they are coming from, and forwarding them to available taxis which are currently in the same area where the client needs to be picked up. There will thus be an application for taxi drivers, used to inform the system about their current state ("available" for a ride or "unavailable") and to handle the ride requests assigned by the system.

1.2 Identifying Stakeholders

- [S1] City's public transportation company: they want to optimize the service offered by taxis in this city by implementing this system;
- [S2] Taxi drivers: the application is going to influence and optimize the performance of their job;
- [S3] Users of the taxi service: they are interested in a system that can easily allow them to request taxis and reserve rides;

1.3 Actors Identifying

- [AC1] Guest: a person using myTaxiService application who hasn't signed up yet into the system;
- [AC2] Passenger user: a person who has registered to myTaxiService passenger application, and can in general send ride and reservation requests;
- [AC3] Driver user: the unique user account associated to a taxi driver, who can use all the functionalities provided by myTaxiService driver application; it is created by the administrator of the system;
- [AC4] Administrator: the person who is responsible for the system and the user accounts management, and in particular for creating the accounts for drivers;

1.4 Goals

[AC1] Being a guest user, myTaxiService application should only provide one functionality:

- [G1] User registration;

[AC2] From the perspective of the passenger user, we think that the application should provide the following features in both web and mobile applications:

- [G2] User login and logout;
- [G3] Sending a ride request for an immediately needed ride, specifying where to be picked up;
- [G4] Receiving notification that a ride request the user has sent has been accepted or queued, if no taxis are currently available;
- [G5 Sending ride reservation requests, specifying the origin and destination of the ride and the reservation time and date;
- [G6] Receiving notification that a reservation request the user has made has been accepted or refused;
- [G7] Receiving a reservation reminder notification 10 minutes before a reservation has to be performed;
- [G8] Checking the user's reservation history;
- [G9] Deleting ride reservations included in the history that have been made and not yet performed;
- [G10] Viewing and modifying the user's personal informations (e-mail address, password);

[AC3] Furthermore, from the perspective of the driver user, the following features will be provided (accessible from the specific driver application only):

- [G11] User authentication and logout;
- [G12] Informing the system about the taxi's current state: "available" to handle new ride requests, or "unavailable";
- [G13] Receiving ride requests, assigned by the system, to be handled immediately;
- [G14] Being able to accept or refuse the ride requests within a limited amount of time;
- [G15] Receiving reserved ride requests, assigned by the system and notified 10 minutes in advance with respect to the reservation time;

- [G16] Notify, at the end of each ride, if it was successfully completed or if the client couldn't be found where he should have been;

[AC4] To the system administrator, the application should provide the following capabilities:

- [G17] Creating, deleting and managing all the kind of user accounts (administrator, passenger, driver) and their attributes;

1.5 Domain properties

We assume that the following conditions hold in the analyzed world:

- [D1] The number of people to be served in a ride due to a single request doesn't exceed the capacity of passengers of a single taxi;
- [D2] Whenever a taxi arrives where he should pick up a client and he can't find him, he waits at most a few minutes, and then signals "missing client" on the application if he doesn't show up;
- [D3] Taxi drivers behave in general as honest workers: they won't keep refusing ride requests, also because at the end of each month the transportation company checks how much they earned and how many rides they accepted and refused, to evaluate their job performance;
- [D4] Whenever a driver accepts a ride request of any kind, he will actually and immediately start heading where he is requested to pick the passenger up;
- [D5] A driver taking care of an accepted ride request of any kind will always reach the spot where he is requested to pick up the passenger, around the time he is supposed to;
- [D6] Each taxicab owned by the transportation company is already provided with an embedded touch screen tablet device (from now on called taxi terminal), located on a protection case in the upper part of the taxi's gear and connected to the vehicle for battery supply, running an Android OS, connected to the internet and used to run the navigation system (which is basically Google Maps' API), and on which myTaxiService mobile application for drivers will be installed. We assume this choice has been made by the company for several reasons:
  - to ensure a comfortable usage of the applications by the driver without compromising the safety of the ride, in terms of attention paid to the road while driving;
  - to guarantee saftey in case of accidents; thanks to the case, the device will always remain stuck to the dashboard, to avoid hitting the driver or the passengers in case of a crash;

  - to prevent thievery acts; the device can't be removed from the protection case without the keys to unlock and open it;

  The transportation company operators can of course unlock the protection case and remove the tablet in case of need for repairing or particular software updates.

- [D7] A driver who accepts to offer a ride to a person asking for it on the side of the road (so not through myTaxiService application) will immediately set his current state on "unavailable", and will set it back on "available" only and immediately after completing the ride.

- [D8] Before the development of the application described in this document, ride and reservation requests could be made only by phone, and the forwarding to taxi drivers was handled by a series of dedicated human operators; no automated system was present.

1.6 Glossary

1.6.1 Definitions

- Driver application: myTaxiService application (mobile only) for driver users. Not publicly available, installed only on taxi terminals, allows drivers to inform the system about their state, to receive and handle ride and reserved ride requests, and to communicate the system when and how a ride coming from a request ends.

- Driver use's state: the state of a logged in driver user, which is set by the user himself to inform the system about his current situation, and can be set on "available" to handle new ride or reserved ride requests, or "unavailable". It is automatically set and locked on "unavailable" while a ride coming from a request is being performed, and automatically set back on "unavailable" just after the ride is communicated to be completed

- myTaxiService: the application discussed in this document, which aims to offer a fast and automated service for requesting and booking taxi rides, and to handle forwarding of requests from clients to drivers in an automatic and clever way.

- Notification: a generic informative message sent by the system to a driver or passenger user. Notifications are shown through popup windows in myTaxiService applications, and are shown only if the user is online when they are received. They can be divided into purely informative (I) and acceptable/refusable (A/R) notifications. Here we list the main situations for which we use the term "notification" sent to a

  - Passenger User:

    * to confirm acceptance or refusal of a reservation request (I);

    - to remind a reservation that has to be performed in the next 10 minutes (I);
    - to tell that a sent ride request has been accepted or queued (I);
    - to ask if the user wants to confirm or cancel a ride request after he has been informed that it has been queued (A/R);
  - Driver User:
  - to assign a ride request to be performed immediately (A/R);
  - to assign a reserved ride, to be performed in 10 minutes (A only);

- Passenger application: myTaxiService application (either web or mobile) for passenger users. Publicly available for download on mobile devices and accessible by web, allows the registered users to send ride and reservation requests, and to check and manage their reservation history.

- Reservation reminder notification: a notification sent to a passenger user who has reserved a ride, 10 minutes before the reservation shall be performed, to remind the reservation and to tell the incoming taxi code.

- Reservation request: a request that can be sent by a passenger user to book a taxi ride in the future. To send the request, the user is needed to specify a starting point, which is the point where the client wants to be picked up by the taxi, a destination for the ride, and a meeting time and date, which has to be at least 2 hours later than when the request is sent for it to be accepted. This kind of request can be cancelled under certain conditions (see [AS14]), and must be accepted by the assigned driver (after being accepted by the system and received by a driver, this kind of request is referred to as "reserved ride request")

- Reserved ride: a taxi ride that has been booked by sending a reservation request through myTaxiService application;

- Ride request: a request that can be sent by a passenger user for a ride immediately needed, it is myTaxiService's application equivalent of actually calling a taxi. a starting point, which is the point where the client wants to be picked up by the taxi, and a meeting time, which isn't actually specified by the client when sending the request, since it's supposed to be as soon as possible. This kind of requests are assigned to drivers, and can be either accepted or refused by them.

- Taxi terminal: a touch screen tablet device, running an android OS, connected to the internet, which is placed on the upper part of the car's dashboard in a sort of protection case, permanently connected to the vehicle for battery supply. It is used to run the navigation system (Google Maps) and myTaxiService applications.

1.6.2 Abbreviations

- [Gn]: n-goal
- [FRn]: n-functional requirement
- [Dn]: n-domain assumption
- [Sn]: n-stakeholder
- [Asn]: n-assumption
- [Acn]: n-actor

1.7 Reference Documents

- RASDs examples from past years;
- IEEE Recommended Practice for Software Requirements Specifications (Revision of IEEE Std 830-1993);

2 Overall Description

2.1 Product Perspective

We will develop the web and mobile passenger applications to offer the passenger users the same experience and functionalities on both. Anyone aiming to use any of the services provided by myTaxiService application as a passenger will need to complete a user registration first; no service will be available without being logged in as a registered user. After logging in, the user will immediately be able to easily send ride or reservation requests. An automated system running on the company's servers, which is constantly keeping trace of taxis positions and availability, will handle the incoming requests following a defined policy and will forward them to the drivers through their specific driver application. Drivers can receive and handle requests using the embedded terminal present on every taxi of the company, which is basically a tablet device running the specific mobile application for drivers. Each driver is associated with a unique driver user account, which is created by the system administrator and can be accessed on taxi's terminal using the personal driver ID and password, provided by the transportation company. After authentication, the driver will be able to set his state on "available" or "unavailable", to receive immediate ride requests and to accept or refuse them, to receive reserved ride requests, and to communicate whether an accepted ride has or hasn't been completed successfully.

2.2 User characteristics

No special skills will be required to correctly use the software. Both passenger and driver users should be able to easily understand how the application works and interact with it through a simple and intuitive interface. Especially in the driver application, since it has to be used while being on the road, the user will be required the least possible amount of attention and interaction to do what he needs to, to ensure the maximum safety of his person and of the passengers.

2.3 Constraints

2.3.1 Regulatory policies

The system we provide must meet the following regulatory policies:

- Taxi policy of the country where myTaxiService will be available;
- Privacy policy for the treatment of personal data stored in the system;
- Policy for the use of the external software involved in the application, e.g. Google Maps;
- Policy about the usage of cookies, to memorize the login details of the user on the application, for example;

2.3.2 Interfaces to other applications

The system should be able to interact with Google Maps' API , which is used by the central system to estimate the time needed by each taxi to reach his destination and in the passenger user's application to select the origin and destination points of a requested ride, when they need to be selected. It has also to be created an interface between the driver user's application and the existent navigation system running on embedded taxi terminals, which also uses Google Maps, and has to be interfaced with our application in order to allow the automatic transmission of the informations about the origin and destination points of a ride when it is accepted by the driver through myTaxiService application.

2.3.3 Parallel operations

The system must be developed to handle parallel and simultaneous requests from multiple users, both drivers and passengers, at the same time. A proper DBMS must be configured to manage multiple data accesses.

2.3.4 Documents related

- Feasibility study: in order to better understand which marketing and development plans shall be done to achieve the fixed goals;
- RASD: Requirement Analysis and Specification Document, to well-understand the given problem and to analyze in a detailed way which are our goals and how to reach them, defining proper requirements and specifications;
- DD: Design Document, to define the real structure of our mobile and web application;
- Code documentation: in order to keep track of the structure of the system we are going to develop and guarantee a much easier maintenance of it;
- Installation Manual: a guide to install the system core of myTaxiService;
- User Manual: a guide on how to use myTaxiService, that will be available for free download on both the website and the application;
- Testing Document: a report of the tests we will perform during the software development;

2.4 Assumptions

To declare how we solve ambiguities in the assignment text, and in general to state the starting points we considered in designing the application, here we provide a list of assumptions we made in developing our solution:

- [AS1] The service provided by the taxis of this company shall not be exclusively due to requests from the application: for instance, if a driver

currently in "available" state happens to find a person asking for a ride on the side of the road, he can answer positively and accept him or her as a client (see [D7] as a related assumption);

- [AS2] To guarantee system security and to ensure a fair behavior by the drivers, the driver application of myTaxiService has to be available for use directly and only on the tablet terminals present on each taxi vehicle, on which it will be installed by the company; the application won't be available for download from any kind of public source.
- [AS3] To ensure system security and avoid potential attacks and problems, a policy has to be followed such that accounts for driver users will be allowed to be created directly and only by the system administrator, with each username corresponding to the driver ID of the driver associated to that account.
- [AS4] The central system has to store data considering the city divided into several city zones, to keep organized and manage the informations about drivers and requests; a queue of available drivers has to be associated to each city zone, and shall be kept updated in real time, to forward requests rapidly.
- [AS5] The central system has to be developed to guarantee a fair management of taxi queues, and it has to do this by forwarding requests coming from passenger users starting from the first available driver in the queue of the corresponding city zone (i.e. the one who has been on "available" state for the longest time from his last state switch);
- [AS6] The central system is capable of offering the passenger users waiting for a taxi to come an estimate of the expected waiting time, calculated in real time through Google Maps, which takes into account also traffic situation, and is the same API used as navigation system on taxicabs;
- [AS7] The central system is capable of receiving GPS informations from each taxi of the company, to keep informations about their position updated (see 3.6.6 Accuracy);
- [AS8] In handling reserved ride requests, the system is able to manage the requests and queues in such a way that it is always ensured that a driver is available to handle each reserved ride request 10 minutes before the ride has to be performed;
- [AS9] If a driver user doesn't either accept or refuse a ride request within a certain defined time limit (15 seconds), the system has to consider it as refused by the driver, and forward it to the next user in the drivers' queue;
- [AS10] Driver users can't refuse a reserved ride request when they are assigned to it: to fulfill the government's transportation policy, implemented in order to provide a reliable reservation service, these requests must be accepted by the assigned drivers.

- [AS11] User registration is going to be required to use any of the functionalities offered by myTaxiService passenger application, in order to collect more detailed informations about the usage of the service and to protect the system from dangerous and bad user behaviors (e.g. ride requests spamming, never showing up after requesting a ride, ...), which can be detected and collected if coming from registered users.

- [AS12] Passenger users who have registered their account but haven't confirmed it yet through the confirmation email sent by the system to their email address shall not be able to use any of the application's functionalities, until they complete the account confirmation.

- [AS13] Reservation requests sent by passenger users less than 2 hours in advance with respect to the reservation's pick up time always have to be refused by the system.

- [AS14] Passenger users have to be allowed to delete ride reservations they have made and are yet to be performed, but only until 1 hour before the reservation has to be performed (i.e. when the ride shall actually start). Otherwise, the reservation can't be canceled.

- [AS15] There has to be a passenger users' waiting queue associated to each city zone, in which passenger users who make a ride request when there are no available taxis in their current city zone are put by the system, until a taxi becomes available and handles their request. When this happen, the user put in the queue has to be told he has been put in such a queue and asked if he still wants to confirm the ride request, or if he wants to cancel it. If the user doesn't answer within a limited amount of time (30 seconds), the request has to be considered as canceled.

- [AS16] MyTaxyService interface for administrator users has to be developed only for the web application. It is not required to provide the administrator interface and its functionalities on the mobile application.

2.5 Future possible implementations

- Shared rides: the transportation company has planned to optimize the proposed system in order to allow shared rides in the near future. The system will be able to plan the route the driver has to follow in a way that allows to pick up other passengers during the ride. In order to achieve this goal the system will also ask the destination of the ride when a user requests one, in the same way as it's already done for reservation requests, and the exact number of people that are included in the ride request.

- SMS notifications: the company also aims to implement a notification functionality that uses SMS to send reminder notifications to passengers which have reserved rides in the near future, and in general to allow users

without internet access to receive notifications; to achieve this, the passenger users will be requested to provide their mobile phone numbers, if they want to.

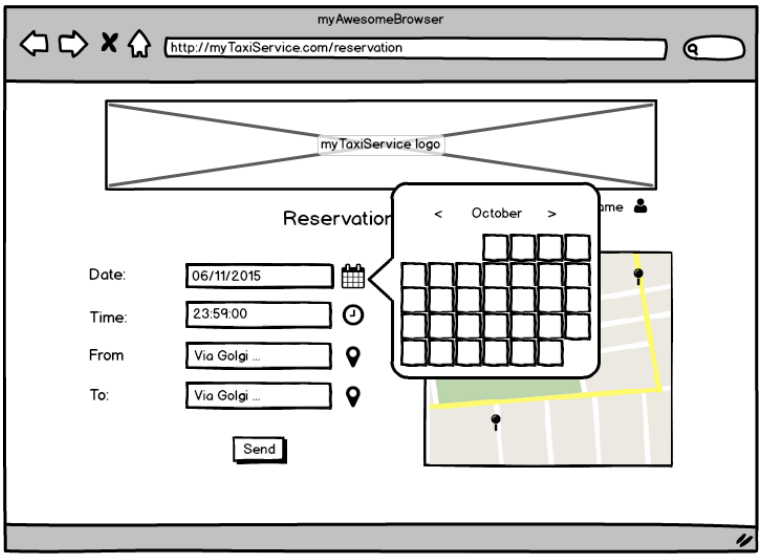
3 Specific requirements

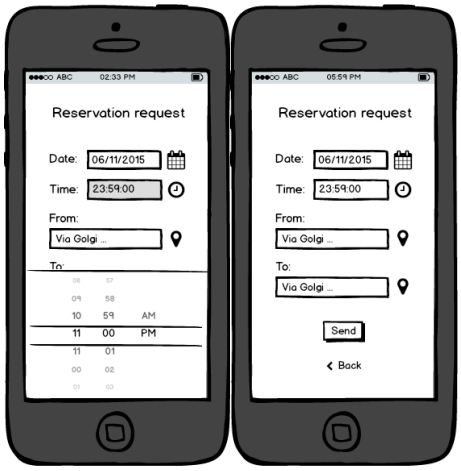
3.1 External Interface requirements

3.1.1 User interfaces

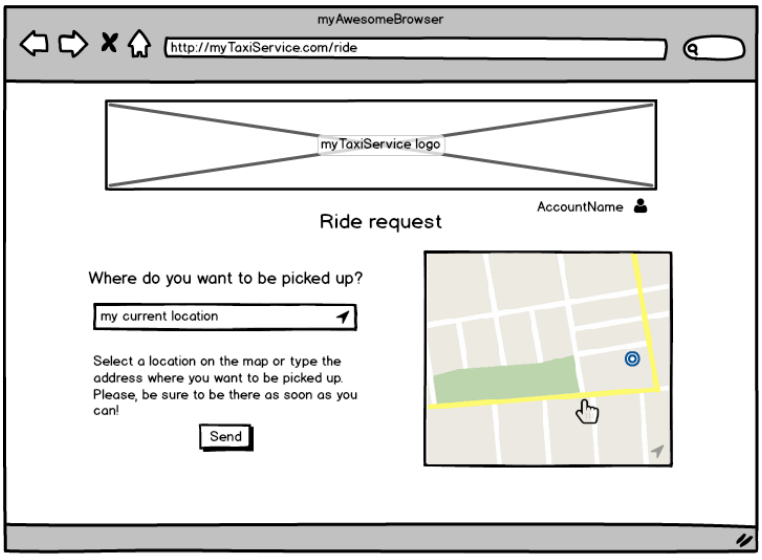
- [G3], [G5], [G8] User's homepage: this mockup shows how the user home page shall look like; from this page the user should be able to request or reserve a ride, and to check the list of reservations he/she has already requested.

- [G5] Ride reservation page: This mockup shows how a user can request a ride reservation. The user can select origin and destination locations by typing the address or by clicking on "from" or "to" icons and then directly on the map, and select date and time by selecting the value inside a popup.

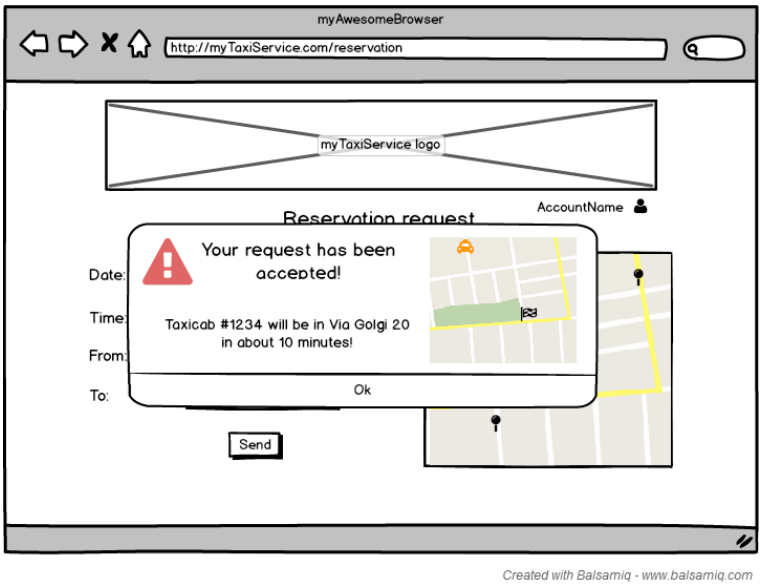




- [G3] Ride request page: This mockup shows how the user can request a ride. To select where he wants to be picked up, the user can choose to use his current location (default choice, provided by his GPS module, if there is one on the device he is currently using), or to select a different spot.

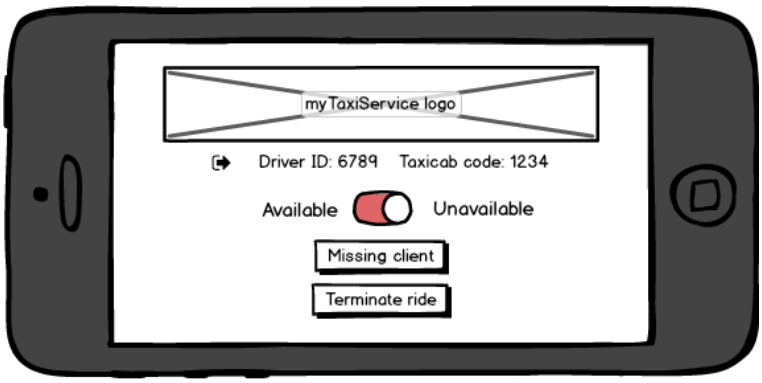


- [G4] Passenger user's notifications: This mockup shows how notifications are displayed when they are received from the system. Notifications are mainly used to provide informations about the code of the incoming taxi and the estimated waiting time when a requested ride is accepted, or in general about the acceptance or refusal of a request sent by the user, but can also be used to display other kinds of messages (see glossary entry "Notification" for more details).

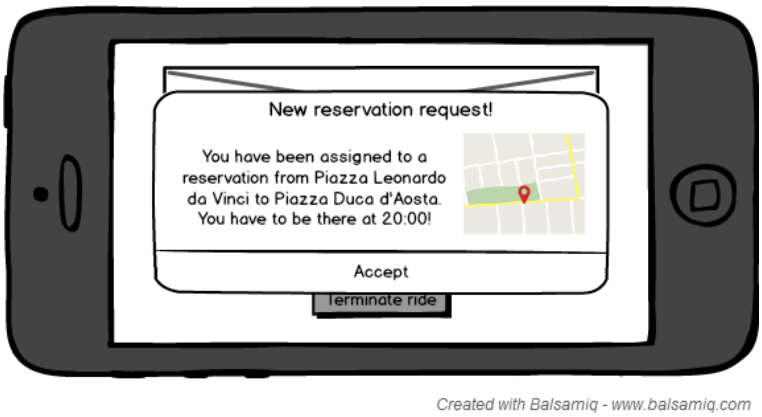


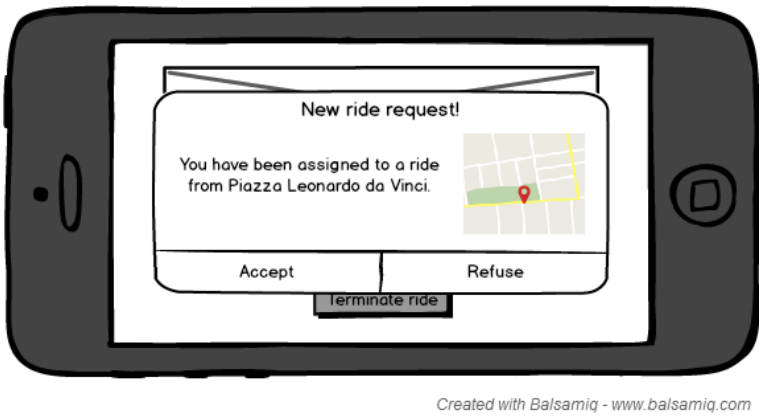
- [G8], [G9] Reservation history: This mockup shows how the user can view the list of reservations he has already requested. If the reservation hasn't already been performed, and only until 1 hour before the ride should be performed, the user can delete it.

- [G11], [G15] Driver user's homepage (during a ride): This mockup shows the homepage of the driver user after he has been assigned and has accepted a ride request. In this condition, he has to first communicate how the ride terminated (successfully or before even starting, because the client was missing) to be able to change his status back to "available" and to receive new requests.



- [G10] Ride/Reserved ride request notifications: This mockup shows how notifications are displayed to the driver. These notifications contain informations about the ride he has been assigned to, and allow him to accept (both kinds) or refuse it (only for the ride requests).





3.2 Software Interfaces

- Database Management System (DBMS)
  - Name: MySQL;
  - Version: Enterprise Edition 5.7
  - Source: http://www.mysql.com/
- HTTP Server
  - Name: Apache HTTP Server
  - Edition: 2.4
  - Source: https://httpd.apache.org/;
- Mapping Service:
  - Name: Google Maps API
  - Version: Web / iOS / Android
  - Source: https://developers.google.com/maps/

3.3 Communications Interfaces

- TCP protocol (both HTTP and HTTPS applications running on standard ports) for client-server communication
- SSL encryption protocol for DBMS communication

3.4 Programmatic interfaces

The system shall be developed to offer also programmatic interfaces, with the main aim of allowing the services offered by the application to be easily extended through the development of new modules, for example to handle the possible future implementation of the ride sharing functionality, as predicted in section 2.5. Concretely, these programmatic interfaces will consist in public interfaces (e.g like the ones that are used in Java language) which allow to make calls to methods of the already existing classes of the project. All the methods and functions used by the system to perform his duty won't be explicitly defined in the essential class diagram provided in section 5.15.1, but in general the ones exposed as APIs will include some that allow to exploit functionalities like the ones listed next, among all the ones available internally on the system:

- User registration, login and logout;
- Sending ride and reservation requests as passenger user;
- Handling taxi's availability state, ride's state (terminated, missing client, ...) and request acceptance or refusal as driver user;
- Computation of taxi positions, city zones busyness, driver and passenger queues statuses, driver users' job performance evaluation, passenger users' reliability evaluation and other useful informative functions;

3.5 Functions

Here we list the functional requirements the system shall match, identified basing on IEEE Standard Requirement Specification document definition, paired with the goals proposed in the "Goals" section and divided by the category of actor or entity which is dealing with such requirement:

3.5.1 [AC1] Guest requirements

- [G1] User registration:
  - [FR 1.1] The application shall provide a registration functionality on its main page;
  - [FR 1.2] The registration process shall request the guest to insert a username, a password and an email address;
  - [FR 1.3] The system shall check the validity of such provided informations (see 3.6.7 Security section of performance requirements), and ask to insert them again if something is detected as not valid, specifying which of the fields generated the problem;
  - [FR 1.4] If the provided informations are valid, the system shall generate and send a registration confirmation email to the user, to be used to effectively activate the new account;

3.5.2 [AC2] Passenger user requirements

- [G2] User login and logout:
  - [FR 2.1] The application shall provide a login functionality for passenger users on its main page, allowing to login by typing username and password;
  - [FR 2.3] After successfully completing the login, the user shall be displayed his home page;
  - [FR 2.4] The application shall provide a logout functionality on the user's home page, accessible from the menu that opens by clicking on the "user account" visual icon (top right of the user's home page);
  - [FR 2.5] On the mobile application for passenger users, the user shall not have to perform the login again after the first time: the application shall keep him logged in later on, even after being closed. This unless the user actually performs a logout;
- [G3] Sending request for an immediately needed ride, specifying where to be picked up:
  - [FR 3.1] The user shall be able to access the "ride request" screen interface from his home page;
  - [FR 3.2] In the "ride request" screen, the application shall allow the user to send a ride request, specifying the position where he wants to be picked up by typing it into a box or selecting it on the city's map;
  - [FR 3.3] The user's position when requesting a ride shall be set by default as the one provided by his GPS module, if there is one on the device he is currently using;
  - [FR 3.4] The system shall check that the provided position is valid (an existing address or coordinate if typed in by hand) and inside the area in which the system is available, and ask to select it again if it is not;
  - [FR 3.5] The system shall prevent the user to send multiple ride requests: if the user has already requested a ride in the short past, that functionality shall be disabled until the current request is answered and completed;
- [G4] Receiving notification that a ride request the user has sent has been accepted or queued, if no taxis are currently available:
  - [FR 4.1] After sending a ride request, in case of a lack of available taxis, the system shall immediately notify the user he has been put into a waiting queue, showing a notification telling the ~~estimated waiting time for a taxi to free up~~ number of users preceding him in the queue; thus the user shall be asked to further confirm the request if he can wait a little, or to cancel it;

    - If the user confirms, or equally if the system didn't detect a lack of available taxis at the beginning, he shall be told to wait until a new notification comes;
    - If the user doesn't provide an answer until a fixed limited amount of time (30 seconds, as assumed in [AS15]), the request shall be considered as cancelled;
  - [FR 4.2] As soon as the sent request is accepted by a driver, the user shall receive and be shown a popup notification confirming that the request has been accepted, and showing the incoming taxi code and estimated waiting time;
  - [FR 4.3] On the mobile application, the notifications described above shall be shown as popup windows on the phone's screen even if the application is closed when they are received (only if the user had already logged in through the application of course);

- [G5] Sending ride reservation requests, specifying the origin and destination of the ride and the reservation time and date:
  - [FR 5.1] The user shall be able to access the "ride reservation" screen interface from his home page;
  - [FR 5.2] In the "ride reservation" screen, the application shall allow the user to send a ride reservation request, asking him to specify the origin and destination of the ride (to be specified in the same way as described above for [G3] requirements), and its time and date;
  - [FR 5.3] Time and date shall be selected from a popup box allowing to select consistent inputs only (i.e. not a date in the past or on a not existing day);
  - [FR 5.4] The system shall check that the provided origin and destination locations are valid and inside the area in which the system is available, and ask to select them again if they are not;

- [G6] Receiving notification that a reservation request the user has made has been accepted or refused:
  - [FR 6.1] A popup notification shall be shown right after the reservation request is sent, telling if it has been accepted or not;

- [G7] Receiving a reservation reminder notification 10 minutes before a reservation has to be performed:
  - [FR 7.1] Ten minutes before a reserved ride has to be performed, the user shall receive and be shown a reminder notification if he is logged into the web or mobile application in that moment (the mobile app always keeps the user logged in, as described in the last requirement of [G2]), telling him the incoming taxi code and the estimated waiting time;

  - [FR 7.2] On the mobile application, this reservation reminder notification above shall be shown as popup window on the phone's screen even if the application is closed when it is received (only if the user had already logged in through the application of course);

- [G8] Checking the user's reservation history:

  - [FR 8.1] The user shall be able to view his ride reservation history on a dedicated screen, accessible from his home page;
  - [FR 8.2] The history shall display all the reservations made by the user and accepted by the system that will be performed in the future, and the ones that have been completed in the last month;
  - [FR 8.3] The history shall be displayed divided into already performed and not yet performed rides:
    * Not yet performed reservations shall be displayed from the closest to happen to the farthest;
    * Already performed rides shall be displayed from the last one performed to the oldest one;
  - [FR 8.4] By clicking on a not yet performed reservation, all the ride details shall be shown;

- [G9] Deleting ride reservations included in the history that have been made and not yet performed;

  - [FR 9.1] Besides each not yet performed reservation there shall be a "delete" button (appears by sliding with the finger from right to left on the reservation in the mobile application), which allows to delete that reservation (asking further confirmation before completing the cancellation); this functionality shall be available only until 1 hour before the reserved ride has to be performed;

- [G10] Viewing and modifying the user's personal informations (e-mail address, password);

  - [FR 10.1] The user shall be able to view his profile informations; this shall be done by opening the "user account" menu through the visual icon on the top right of his home page and by selecting "view profile";
  - [FR 10.2] On the "view profile" screen, the user shall be able to modify his password and e-mail address (provided that he provides new valid ones);

3.5.3 [AC3] Driver user requirements

- [G11] User authentication and logout:

  - [FR 11.1] The application shall allow the driver user to log into the system from the main page, by inserting its driver ID and password;

- [G12] Informing the system about his current state: "available" to handle new ride requests, or "unavailable":

  - [FR 12.1] On the user's home page, the application shall allow the driver to inform the system when he is in "available" or "unavailable" state, by simply sliding with a finger on a switch between the two states;
  - [FR 12.2] The switch shall be set on "unavailable" by default right after the user has logged in;

- [G13] Receiving ride requests, assigned by the system, to be handled immediately:

  - [FR 13.1] The application shall notify the driver that the system has assigned him a ride or a reservation through a popup window, showing the ride's starting point;

- [G14] Being able to accept or refuse the request within a limited amount of time:

  - [FR 14.1] When a ride request is displayed, the application shall ask the driver to accept or refuse the assigned ride, giving a fixed limited amount of time to answer (10 seconds, as specified in [AS9]);
  - [FR 14.2] If the driver accepts the request, his state shall automatically be set on "unavailable", the screen shall switch to the taxi's navigation system application and his destination shall be displayed;
  - [FR 14.3] If the driver refuses the request or doesn't provide an answer within the given time, the system shall consider the request as refused and close the popup window, keeping the driver's state on "available";

- [G15] Receiving reserved ride requests, assigned by the system and notified 10 minutes in advance with respect to the reservation time:

  - [FR 15.1] When a reserved ride request is assigned to the driver by the system, a popup window shall be shown 10 minutes in advance, showing the reservation details;
  - [FR 15.2] Since this request can't be refused, after the driver accepts it the screen shall switch to the navigation system application, and the origin and destination of the ride shall be displayed on the map;

- [G16] Notify, at the end of each ride, if it was successfully completed or if the client couldn't be found where he should have been:

- [FR 16.1] When a ride or reserved ride request is accepted by the driver user, the "terminate ride" and "missing client" buttons shall become available on his application's main page;
- [FR 16.2] After tapping on one of the buttons (used to end a ride), the application shall communicate to the system that the current ride has ended, and the user's status shall be automatically set back to "available";

3.5.4 [AC4] Administrator requirements

- [G17] Creating, deleting and managing all the kind of user accounts (administrator, passenger, driver) and their attributes:
  - [FR 17.1] The administrator interface shall be accessible by logging in from myTaxiService web application only (as stated in [AS16]), through the usual login form used by passenger users;
  - [FR 17.2] The administrator interface shall allow him to create new passenger users providing a valid username, password and email;
  - [FR 17.3] The interface shall allow him to create new administrator users providing a valid username and a password;
  - [FR 17.4] The interface shall allow him to create new driver users providing a valid driver ID as a username and a password;
  - [FR 17.5] The interface shall allow him to formulate queries searching for existing users, displaying the results in a "query result" screen;
  - [FR 17.6] The interface of the "query result" screen shall allow him to delete any user (either admin, driver or passenger) and to edit any user's information;

3.6 System requirements

Here we list the non-functional requirements the system shall provide, paired with the goals proposed in the "Goals" section and divided by the category of actor or entity which is dealing with or providing such requirement:

3.6.1 Maintainability

- The system shall permit replacement and upgrade of the hardware without downtime;
- The system shall permit software upgrade without downtime;
- The MTTR (Mean Time to Repair) of the system shall not exceed 24 hours;

3.6.2 Reliability

- The MTTF (Mean Time To Failure) shall not exceed 1 year;
- Each release of the application should satisfy all the code tests before being released;

3.6.3 Availability

- The system should be available 24 hours a day, 7 days a week.

3.6.4 Capacity

- The system shall support without problems at least 2000 passenger users simultaneously connected;
- The system shall support without problems at least 500 driver users simultaneously connected;
- The system shall support without problems at least 1000 simultaneous requests;

3.6.5 Time Response

- The system shall provide a feedback for each user's action within a maximum of 3 seconds;
- Every time an event that interests the user occurs, the system shall notify him within a maximum of 3 seconds;

3.6.6 Accuracy

- The system shall update the informations about the position of each taxi in his database every 30 seconds;

3.6.7 Security

In order to guarantee the security of the whole system, the following requirements must be ensured:

- Passwords must be stored encrypted using hash functions;
- The system should accepts users' passwords only if they comply with certain complexity constraints: for instance, the password must have at least 8 characters, at least one digit and a non alphanumeric character;
- The system should validate each input of the user in order to protect the system from accidental or malicious usage, like for instance SQL injection;
- Communication between client and server must be encrypted using HTTPS protocol;

4 Scenarios Identifying

Here we list some of the possible scenarios of myTaxiService application usage, divided by the kind of actor they apply to:

4.1 [AC1] Guest scenario

4.1.1 Scenario: passenger user registration

David just got off from work, but he missed the last available bus to his home. So he downloads myTaxiService app on his smartphone, he opens it and registers a new account, providing his email, a valid username and password. He checks his email inbox folder, in which he finds myTaxiService registration confirmation email, he confirms the subscription and logs in on the application with his brand new account; he won't have to repeat this login operation again: the app will keep him logged in until he possibly chooses to log out. He can now perform all the activities of a registered passenger user.

4.2 [AC2] Passenger user scenarios

4.2.1 Scenario: accepted ride request through mobile application

David and his girlfriend went to the cinema this evening, and since the movie finished late they missed the last available bus to their home. So David, who has already registered to myTaxiService, decides to call a taxi: he opens the app on his smartphone and just taps on "request a ride", and sends the request by tapping on "send request", keeping the position provided by his GPS module as the ride's desired starting point. After few seconds, he receives a confirmation notification, telling him that taxicab no. 17689 is on his way, and will meet him in about 3 minutes. Thus David taps on "ok" and waits for the cab to come and bring him and his girlfriend home, fast and safely.

4.2.2 Scenario: login and queued ride request through web application

Carla is home and has to get to her friend's Anna party, that is starting at 7 p.m. at her place. Carla's got no driving license, and she lives far from any public transportation spot, so she decides to call a taxi, to avoid being late. Carla is currently chatting with Anna on her laptop, so she decides to call the taxi using myTaxiService web application. She opens a new tab of her browser and types myTaxiService.com; being already registered to the service, she quickly logs in and clicks on "request a ride". She types her home's address as ride's starting point and sends the request. Since it's rush hour, it happens that there's currently no available taxi in that area of the city. So Carla immediately receives a notification which tells her she's been put into a waiting queue, and that she'll have to wait about 6 minutes for a cab to be available; she is asked if she wants to confirm or delete her request. She chooses to confirm it, since

she's got no hurry, and she waits. After a few minutes, a new notification is received and shown on her homepage, telling that taxicab no. 17446 will pick her up in about 4 minutes. She then goes out of her house and waits a little, until the taxi comes and takes her to her friends' party.

4.2.3 Scenario: ride reservation request with non valid and valid time

Philip is in Memphis, where he has been for some days to attend an important conference. He is boarding on his plane in 15 minutes, and he knows he'll land in about 1.5 hours. Since he'll land at rush hour, he wants to be sure that a taxi will be waiting for him outside the airport, to bring him home. To do this, he opens his myTaxiService app on his brand new iPhone; being a regular user of the service, he is already logged in. He taps on "reserve a ride", on his home page. He selects the date and time when he wants to be picked up, taps on "from" button and taps on the airport on the city's map shown; then he taps on "to" box and writes his home address. He then clicks "send", and the system immediately answers through a notification: he has set the reservation time too close (less than 2 hours in advance), so the application tells him the reservation can't be completed. Philip executes again the above steps, this time inserting a valid reservation time; after sending, he is immediately confirmed that the reservation has been successfully completed. Later, after landing, Philip receives a reminder notification on his phone, reminding him that taxicab no. 14599 will be waiting for him outside the airport in 10 minutes. He is thus able to collect his luggage and go meet the cab in the taxi waiting area.

4.2.4 Scenario: deleting ride reservation request

Philip is in Chicago, where he has been for some days to attend another important conference. He should board on his plane in 20 minutes, and he should land in about 5 hours as usual, so he has already made a reservation for a taxi to wait for him when he arrives at the airport. But suddenly, an announcement is made at the airport, telling that his flight will be delayed of an unknown amount of hours, due to some problems with the plane. So, Mario decides to delete his ride reservation. He does it by opening myTaxiService app on his smartphone, tapping on "reservation history", sliding his finger on the target reserved ride and tapping on the "delete" button appeared beside it. He then confirms the cancellation in the popup window that has just appeared, and sadly waits for news about his flight.

4.3 [AC3] Driver user scenarios

Mario is a taxi driver working for myTaxiService public company. The following scenarios are the most common ones that can happen during his working day:

4.3.1 Scenario: login, accepted and completed ride request

Mario has just started his shift for today, and he gets into his cab. He turns on the navigation system and the terminal, and he logs in the application using his personal driver ID and password. His state is set by default on "unavailable" when he logs in, so he immediately sets it on "available" and gets ready to work. After a minute, a ride request notification pops up on the terminal's screen: a client needs a ride from Central Station. Mario accepts the request by tapping on "accept", and the terminal immediately shows the navigation system, highlighting his destination on the map and the best route to follow. He can easily switch between the navigation app and myTaxiService driver app by just sliding horizontally with his finger on the terminal's screen. When he accepts the request, his taxi's state is automatically set on "unavailable". He starts heading to his destination, where in a couple of minutes he meets his client. He brings him where he needs to, gets paid and, back to the driver application on the terminal, taps on "terminate ride", which automatically sets the state on "available", and gets ready for the next ride.

4.3.2 Scenario: reserved ride request and missing client

Mario is in the middle of his working day, and he just completed a ride: his state is set back to "available". After some minutes, a reserved ride request appears on the taxi's terminal, telling him he should be in People's Square in 10 minutes, to pick up a client who reserved a ride to the Central Station. He taps on "ok" and the screen switches to the navigation app, immediately showing him the best route to follow to reach People's square, and then the Central Station. The taxi's state has been automatically set on "unavailable". So he heads to People's Square, where he waits a couple of minutes, but his passenger won't show up. He thus assumes he has renounced to his reserved ride, and so he taps on "missing client", which communicates the system about the fact and sets back the taxi's state to "available".

4.3.3 Scenario: state switching, ride request refusal, extraordinary ride, logout

Mario has just left a passenger at his destination, and has set back his state to "available". After a few seconds, a new ride request pops up on the application running on the terminal. Mario suddenly realizes he has to pee, since it's been sitting in the cab for 5 straight hours; so he taps on "refuse" and sets his state to "unavailable", and heads toward the closest public toilet. After doing what he needed, he comes back to his taxi and sets his state back to "available", but he sees a guy waving his hand towards him: he is asking for a ride. Thus, Mario immediately switches his state to "unavailable" and welcomes the new passenger, asking where he should bring him. After completing the ride and telling the application so, Mario realizes that his shift is over for today, and so he keeps the taxi's state on "unavailable", heading to the company's garage. Once reached, he logs out from the driver application and goes home.

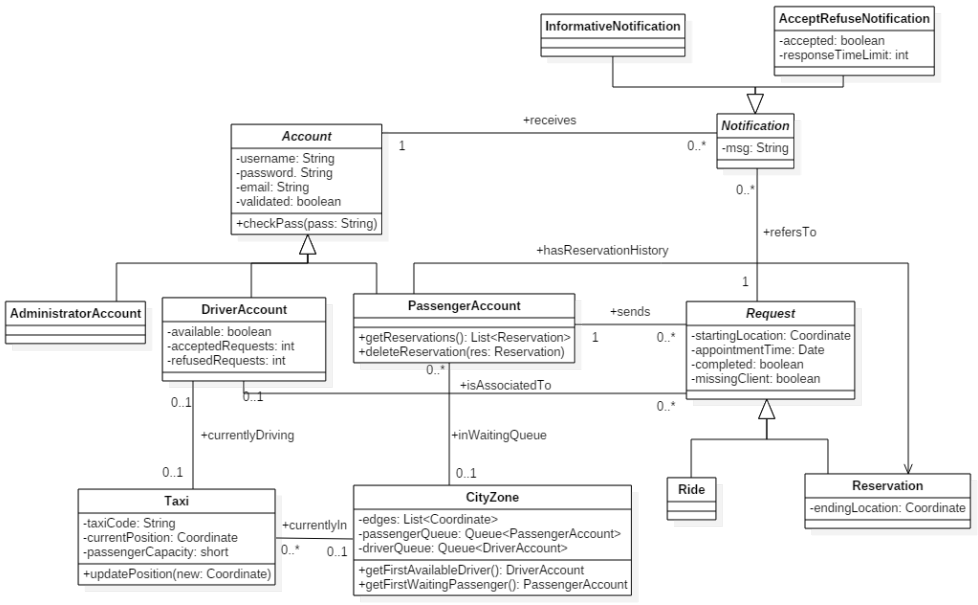
4.4 [AC4] Administrator scenario

4.4.1 Scenario: driver user cancellation and creation

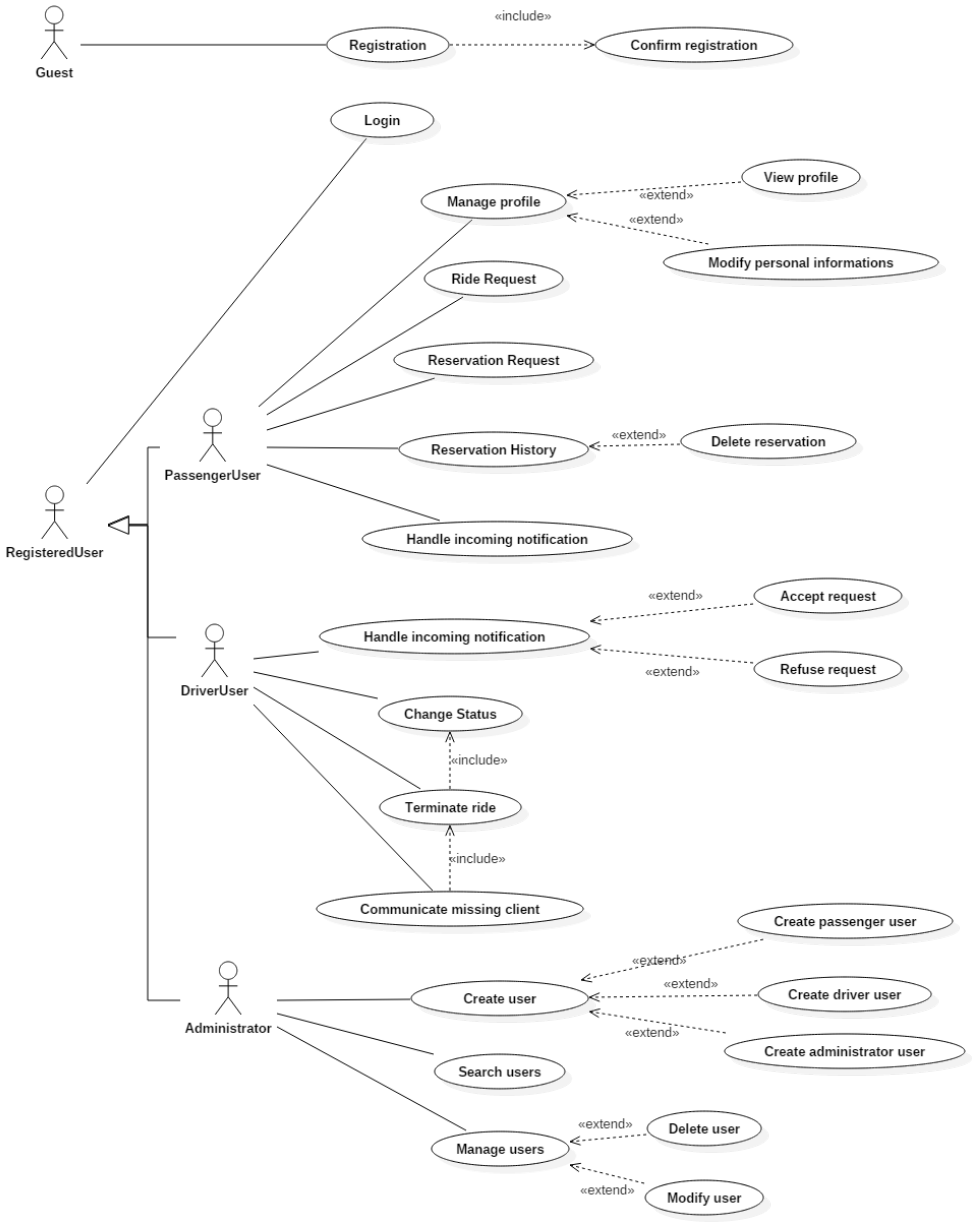
Bill, one of myTaxiService administrators, is told by his boss that the company has just fired the driver Mario R***i, with driver ID MR130477A, and has hired the new driver Pablo E*****r, with given ID PE011249A as a replacement. He is in charge to delete the driver user account of Mario and to create a new one for Pablo. He logs into the system with his administrator credentials and he searches for the driver user with username MR130477A; he deletes it by clicking on the corresponding button. Then he clicks on "create new user" button and creates a new driver user with username PE011249A, he sets a valid and safe password, and he confirms. He will then provide Pablo the password, and instruct him on how to complete the login on the driver application.

5 UML Models

5.1 Class Diagrams

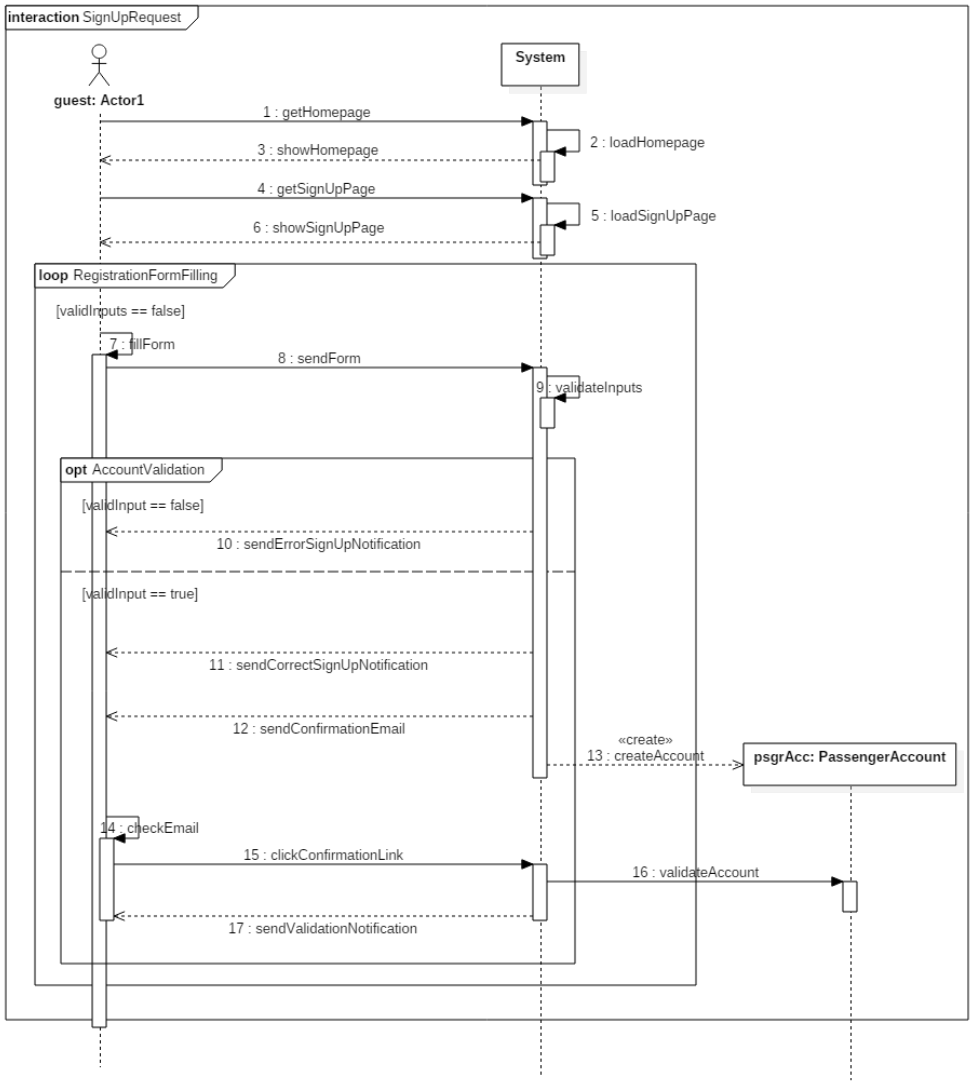


5.2 Use cases


Guest
Registration
«include»
Confirm registration
Login
Manage profile
View profile
«extend»
«extend»
Modify personal informations
Ride Request
Reservation Request
PassengerUser
Reservation History
«extend»
Delete reservation
RegisteredUser
Handle incoming notification
«extend»
Accept request
Handle incoming notification
«extend»
Refuse request
DriverUser
Change Status
«include»
Terminate ride
«include»
Communicate missing client
Create passenger user
«extend»
Create user
«extend»
Create driver user
«extend»
Create administrator user
Administrator
Search users
«extend»
Delete user
Manage users
«extend»
Modify user

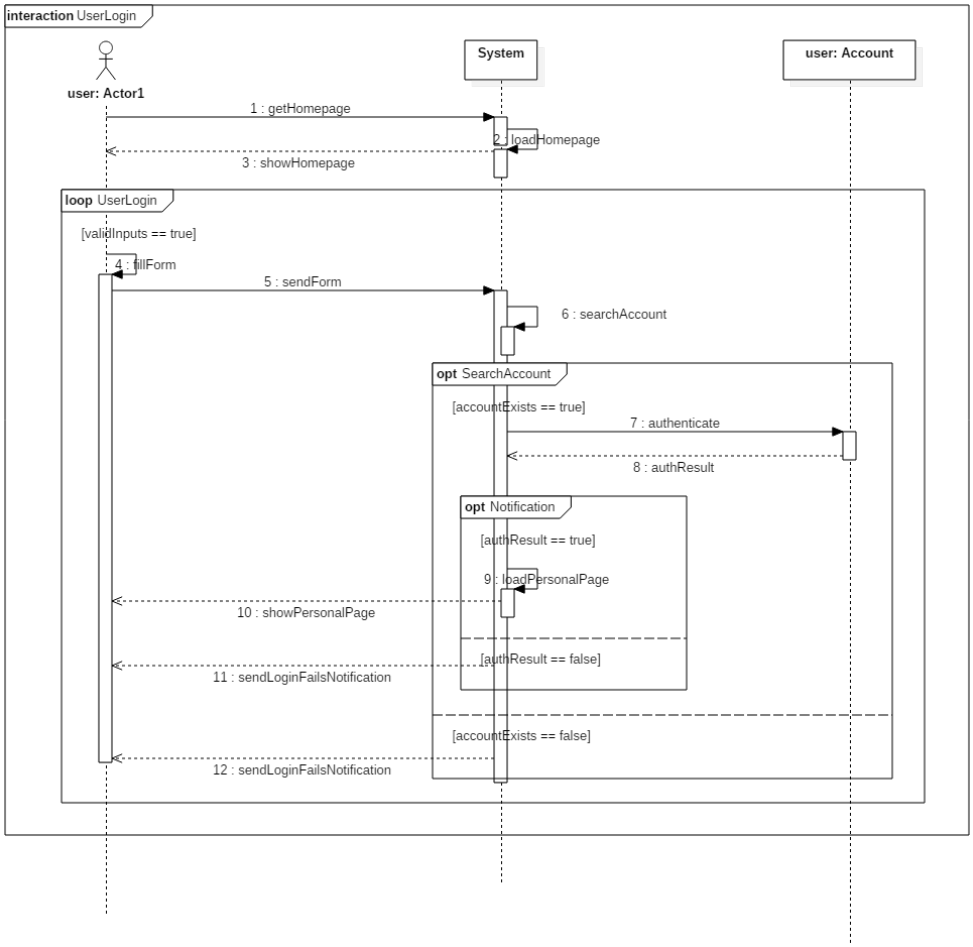
5.2.1 Guest registers to myTaxiService

Actor	[AC1] Guest user
Goal	[G1]
Input Conditions	No conditions
Event Flow	1. The guest user clicks on "Sign up" button on the main page of the application; 2. The system shows him the registration form containing the fields: email, username, password, confirm password; 3. The guest user fills all the fields; 4. The guest user clicks on "Register" button; 5. The system creates a new passenger user account that has to be verified in order to be used; 6. The guest user receives an email in which a confirmation link is provided; 7. The user confirms his subscription by clicking on the confirmation link, activating his account;
Output Conditions	The guest user will now be able to log into the system.
Exceptions	1. One or more fields are left blank; 2. The provided email is not well formed; 3. The provided email has been already used; 4. The provided user has already been used; 5. The provided password doesn't match the one inside the "confirm password" field; 6. The provided password doesn't match the complexity constraints; All the exceptions are handled by informing the user of the problem and then the event flow shall restart from point 2.


interaction SignUpRequest
guest: Actor1
System
1 : getHomepage
2 : loadHomepage
3 : showHomepage
4 : getSignUpPage
5 : loadSignUpPage
6 : showSignUpPage
loop RegistrationFormFilling
[validInputs == false]
7 : fillForm
8 : sendForm
9 : validateInputs
opt AccountValidation
[validInput == false]
10 : sendErrorSignUpNotification
[validInput == true]
11 : sendCorrectSignUpNotification
12 : sendConfirmationEmail
«create»
13 : createAccount
psgrAcc: PassengerAccount
14 : checkEmail
15 : clickConfirmationLink
16 : validateAccount
17 : sendValidationNotification

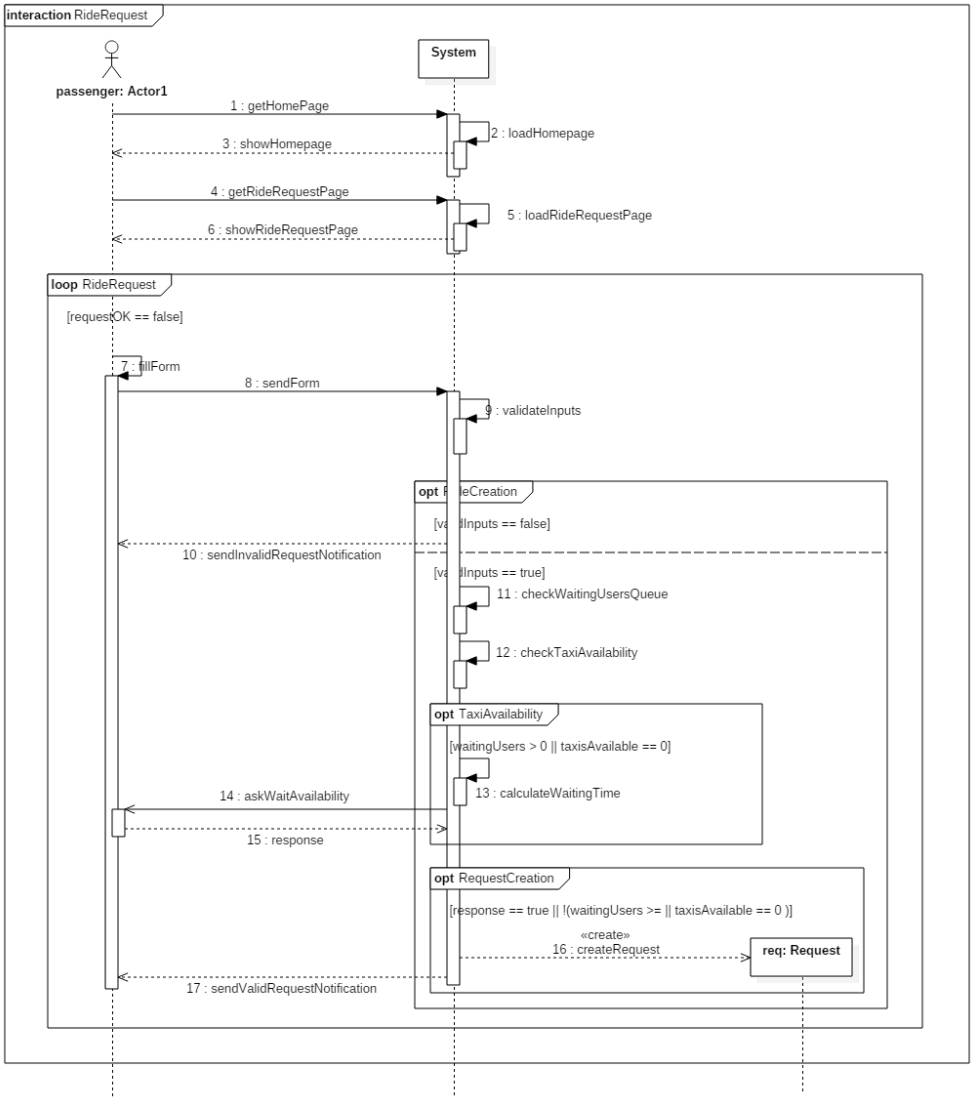
5.2.2 User logs in

Actor	[AC2] Passenger User, [AC3] Driver User, [AC4] Administrator
Goal	[G2]
Input Conditions	The user must own a valid account
Event Flow	1. The user, from the main page of the application, fills Username and Password fields; 2. The user clicks on the "Login" button; 3. The user is redirected to his home page;
Output Conditions	The user will be able to use all the functionalities offered to passenger users.
Exceptions	1. One or more fields are left blank; 2. The provided username doesn't exist; 3. The password doesn't match the one paired with the provided username; All the exceptions are handled by informing the user of the problem and then the event flow shall restart from point 2.


interaction UserLogin
user: Actor1
System
user: Account
1 : getHomepage
2 : loadHomepage
3 : showHomepage
loop UserLogin
[validInputs == true]
4 : fillForm
5 : sendForm
6 : searchAccount
opt SearchAccount
[accountExists == true]
7 : authenticate
8 : authResult
opt Notification
[authResult == true]
9 : loadPersonalPage
10 : showPersonalPage
[authResult == false]
11 : sendLoginFailsNotification
[accountExists == false]
12 : sendLoginFailsNotification

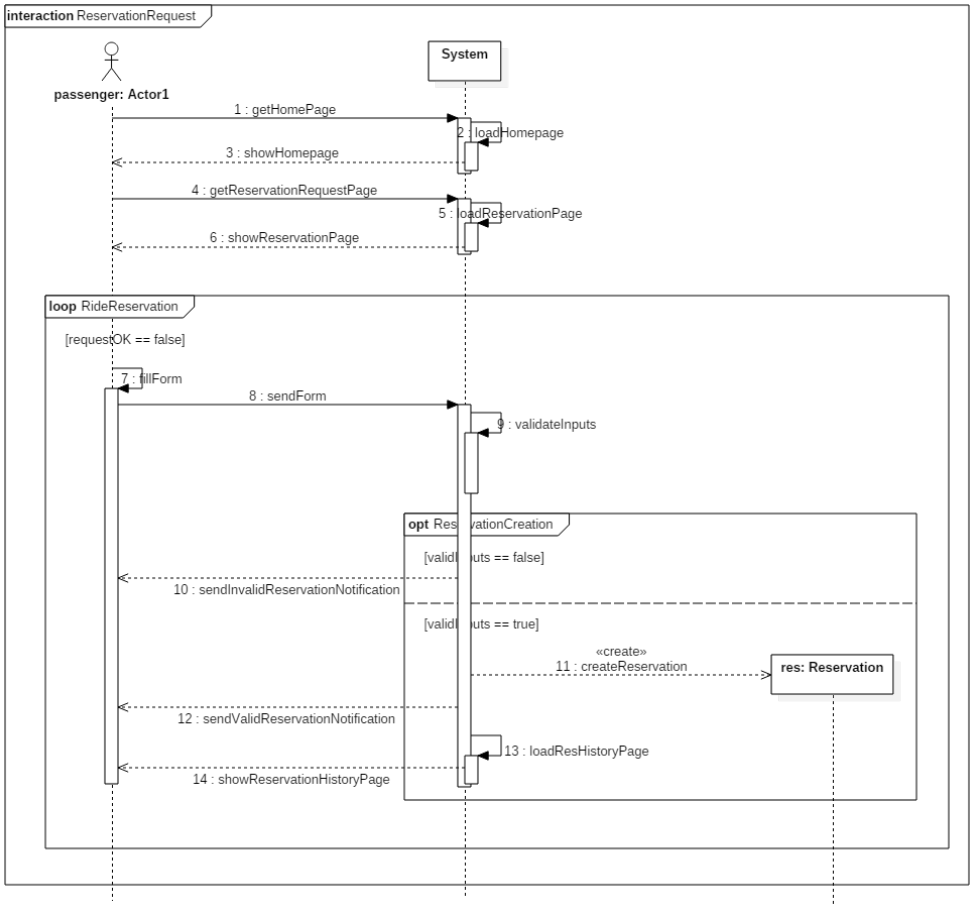
5.2.3 Passenger user requests a ride

Actor	[AC2] Passenger User
Goal	[G3]
Input Conditions	1. The user must be logged in; 2. The user mustn't have already sent a ride request which hasn't been completed yet;
Event Flow	1. The user, from his main page, clicks on the button "Request a ride now"; 2. The system shows him a page containing a map that the user can use to select the meeting point and a box in which he can directly type the address; the box is filled by default with the location provided by the GPS module, if there is one on the user's device; 3. The user selects where he wants to be picked up by either leaving his current position, typing a different address in the form box, or just selecting a location on the map; 4. The user clicks on "Send" button; 5. The user receives a notification that tells him if the requested ride has been correctly saved in the system or if the provided data are not valid;
Output Conditions	The user is now waiting to receive the notification that a driver has taken in care of his ride, and won't be able to make another ride request until the ride is completed.
Exceptions	1. The selected point or address is outside the area in which myTaxiService is available; 2. The typed address is invalid; 3. If there aren't available taxi at the moment, the system informs the user about the situation communicating him how much users are waiting the availability of a taxi and the estimated time of the queue, asking the user if he wants to wait or not. If one of the previous exceptions is thrown, the user will be informed and then the flow will restart from point 2.


interaction RideRequest
System
passenger: Actor1
1 : getHomePage
2 : loadHomepage
3 : showHomepage
4 : getRideRequestPage
5 : loadRideRequestPage
6 : showRideRequestPage
loop RideRequest
[requestOK == false]
7 : fillForm
8 : sendForm
9 : validateInputs
opt RideCreation
[validInputs == false]
10 : sendInvalidRequestNotification
[validInputs == true]
11 : checkWaitingUsersQueue
12 : checkTaxiAvailability
opt TaxiAvailability
[waitingUsers > 0 || taxisAvailable == 0]
13 : calculateWaitingTime
14 : askWaitAvailability
15 : response
opt RequestCreation
[response == true || !(waitingUsers >= || taxisAvailable == 0)]
«create»
16 : createRequest
req: Request
17 : sendValidRequestNotification

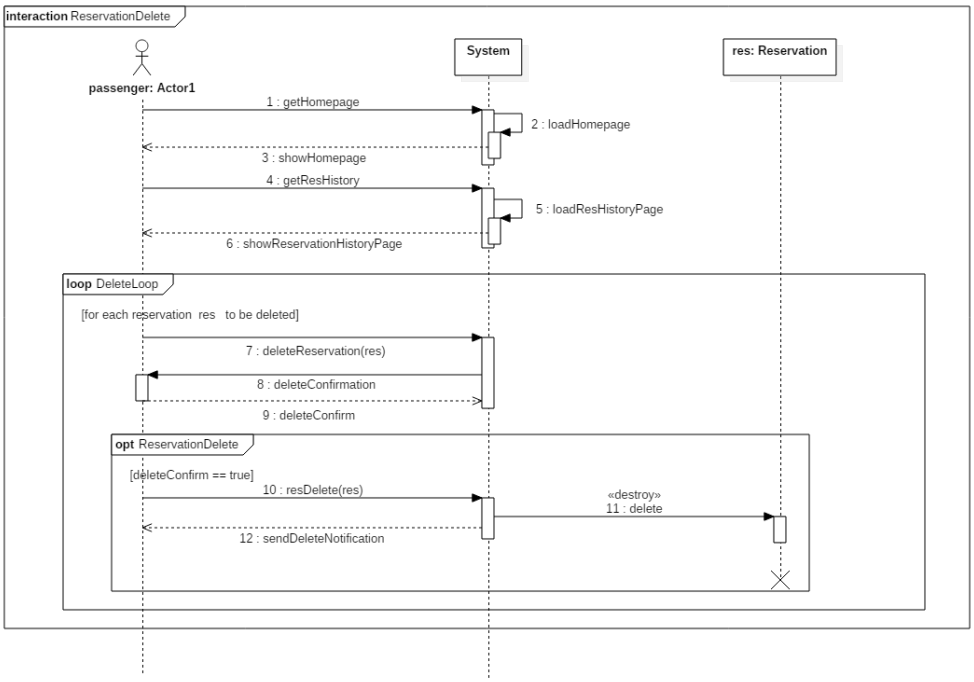
5.2.4 Passenger user requests a reservation

Actor	[AC2] Passenger User
Goal	[G5]
Input Conditions	The user must be logged in
Event Flow	1. The user, from his main page, clicks on "Reserve a ride" button; 2. The system shows a page containing a form and a map which the user can use to specify starting and ending locations for his ride; 3. The user selects when the reservation has to be scheduled (by specifying date and time) and the starting and ending locations; 4. The user presses "Send" button;
Output Conditions	A message is shown telling that the reservation has been scheduled and the user will be redirected to his reservation history page.
Exceptions	1. At least one of the provided locations is either invalid or outside the area in which myTaxiService is available; 2. The provided date is not valid or the request has been made less than 2 hours in advance; If one of the previous exceptions is thrown, the user will be informed and then the flow will restart from point 2.


interaction ReservationRequest
passenger: Actor1
System
1 : getHomePage
2 : loadHomepage
3 : showHomepage
4 : getReservationRequestPage
5 : loadReservationPage
6 : showReservationPage
loop RideReservation
[requestOK == false]
7 : fillForm
8 : sendForm
9 : validateInputs
opt
[validInputs == false]
10 : sendInvalidReservationNotification
[validInputs == true]
«create»
11 : createReservation
res: Reservation
12 : sendValidReservationNotification
13 : loadResHistoryPage
14 : showReservationHistoryPage

5.2.5 Passenger user deletes a reservation

Actor	[AC2] Passenger User
Goal	[G9]
Input Conditions	1. The user must be logged in; 2. The user must have at least one not yet performed reservation in his history;
Event Flow	1. The user clicks on the "Reservation history" button on his home page; 2. The application will show, on top of the page, the list of reservations he has already requested but not yet performed; 3. The user can choose to directly click on the "Delete" button beside the reservation (in this case the flow will continues from 6) or to click on the reservation; 4. The details of the selected reservation are shown; 5. The user clicks on "Delete" button; 6. A confirmation message is shown, asking the user to confirm or cancel the operation; 7. The user clicks on "OK" button;
Output Conditions	The reservation is deleted from the system, and the user is redirected to his reservation history page.
Exceptions	1. The user tries to delete a reservation that has to be performed in less than 1 hour (see [AS14]);

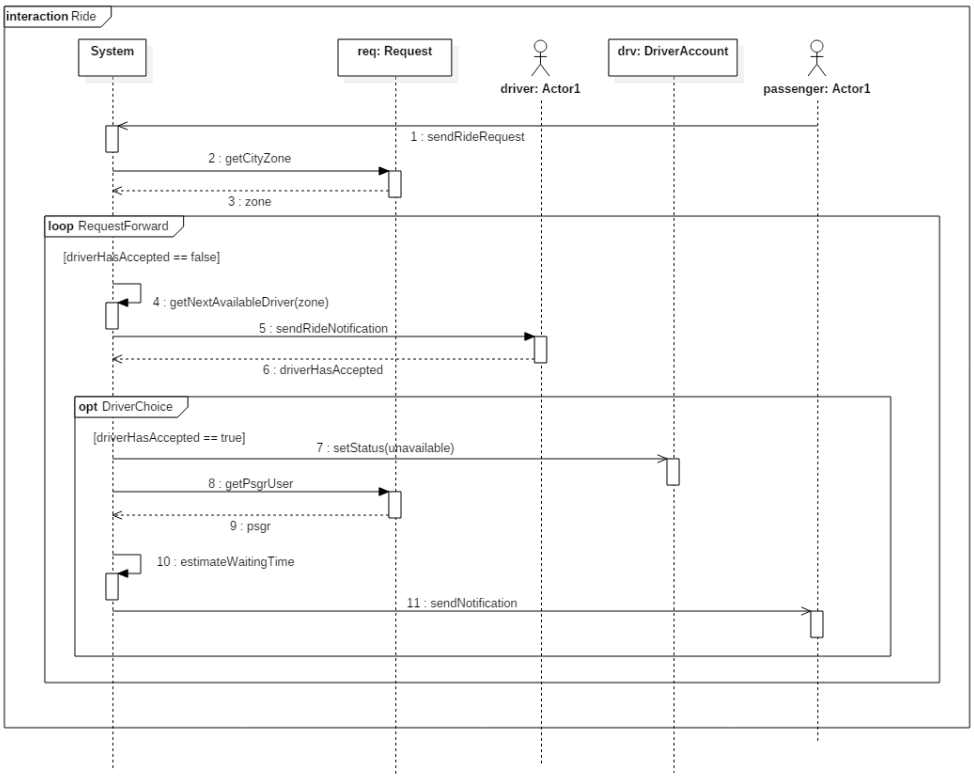


5.2.6 Passenger user receives a ride confirmation notification

Actor	[AC2] Passenger User
Goal	[G14]
Input Conditions	1. The user must be logged in; 2. The user must have requested a ride; 3. The ride must have been accepted by a driver;
Event Flow	1. The user is shown a notification sent by the system, through a popup containing the incoming taxicab code and the estimated waiting time, telling that a driver is going to pick him up at the meeting point; 2. The user click on "Ok" button;
Output Conditions	The popup disappears from the screen and the user is back on whatever page he was before the notification arrived;
Exceptions	No exceptions.

5.2.7 Driver user receives a ride request notification

Actor	[AC3] Driver User
Goal	[G13]
Input Conditions	1. The driver user must be logged in; 2. A ride has been requested by a passenger user in the city zone the driver is currently in; 3. The driver user's status must be on "available" and he must be the first available driver in that city zone's drivers queue;
Event Flow	1. The driver user is notified by a popup appearing on his main page; 2. The driver has a few seconds to decide if he wants to take care of the notified ride by clicking on "Accept" button, or to decline it by clicking on "Refuse" button;
Output Conditions	The notification disappears from the screen and, if the driver has accepted the request, his status is set on "unavailable"; otherwise, it remains on "available".
Exceptions	1. The driver user doesn't provide a choice within the maximum time limit. The system will consider the request as refused by the driver.

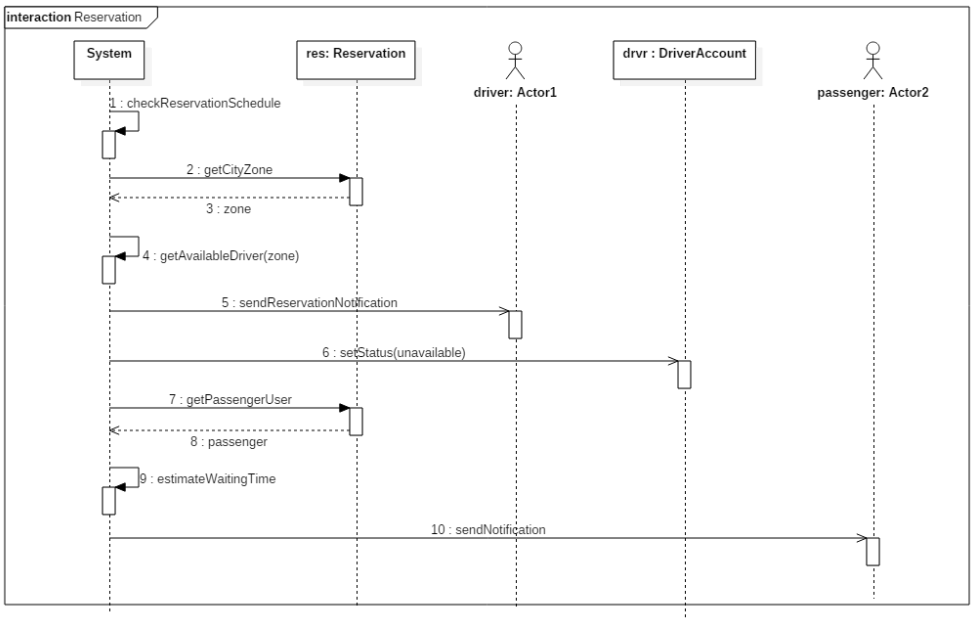

interaction Ride
System
req: Request
driver: Actor1
drv: DriverAccount
passenger: Actor1
1 : sendRideRequest
2 : getCityZone
3 : zone
loop RequestForward
[driverHasAccepted == false]
4 : getNextAvailableDriver(zone)
5 : sendRideNotification
6 : driverHasAccepted
opt DriverChoice
[driverHasAccepted == true]
7 : setStatus(unavailable)
8 : getPsgrUser
9 : psgr
10 : estimateWaitingTime
11 : sendNotification

5.2.8 Passenger user receives a reservation reminder notification

Actor	[AC2] Passenger User
Goal	[G6]
Input Conditions	1. The user must be logged in; 2. The user must have requested a reservation that is going to occur in 10 minutes; 3. The reservation has been assigned to a driver;
Event Flow	1. The user is notified by the system, through a popup containing the incoming taxicab code, that his reservation shall be performed in 10 minutes, and that a driver is going to pick him up at the meeting point; 2. The user clicks on "Ok" button;
Output Conditions	The popup disappears from the screen and the user is back on whatever page he was before the notification arrived;
Exceptions	No exceptions.

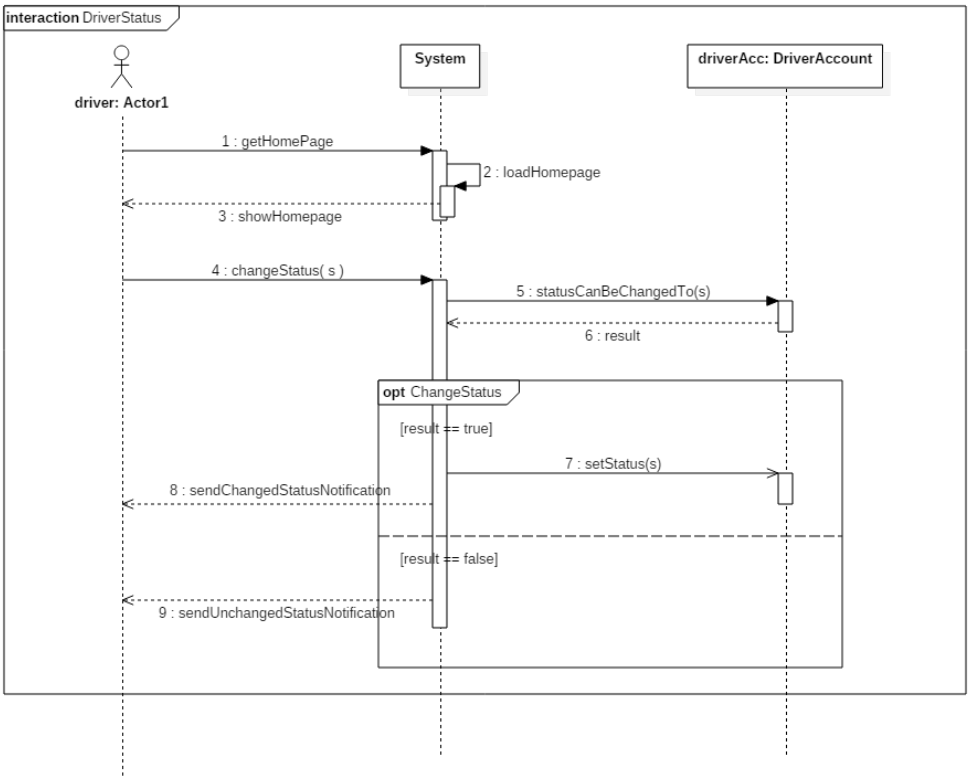
5.2.9 Driver user receives a reservation request notification

Actor	[AC3] Driver User
Goal	[G15]
Input Conditions	1. The driver must be logged in; 2. The driver user must be available; 3. A scheduled reservation has to be performed now;
Event Flow	1. The user, from the main page of application, is notified by a popup that appears on the screen; 2. The user click on "Ok" button;
Output Conditions	The notification disappears from the screen and the status of the driver user changes to unavailable.
Exceptions	No exceptions



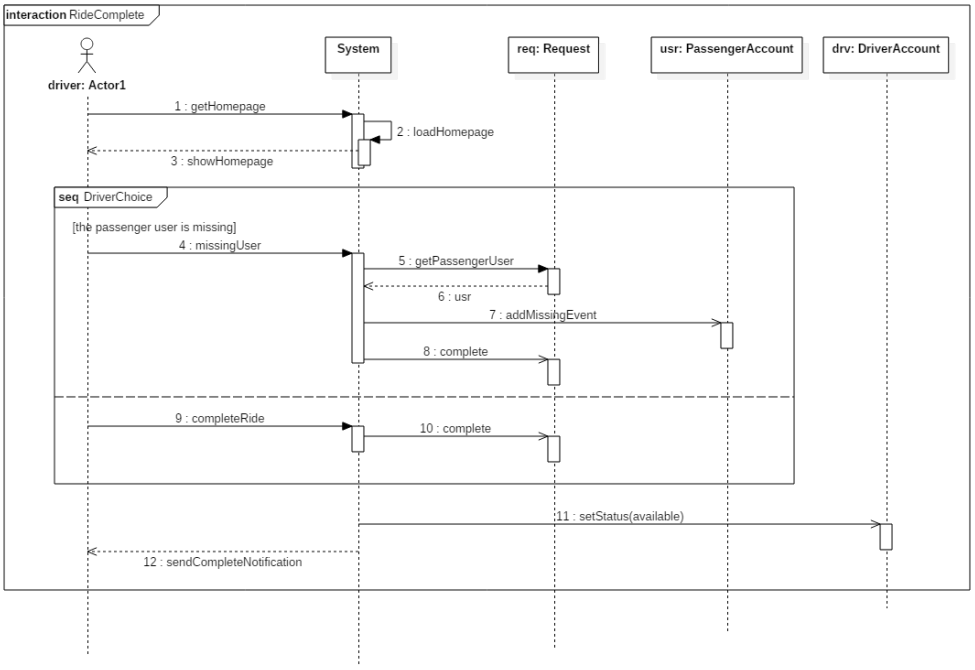
5.2.10 Driver user changes his status

Actor	[AC3] Driver User
Goal	[G12]
Input Conditions	1. The driver user must be logged in; 2. The driver user mustn't be performing a ride;
Event Flow	1. The driver user swipes on the "available/unavailable" switcher button on his main page;
Output Conditions	The availability status changes to "available"/"unavailable"
Exceptions	No exceptions


interaction DriverStatus
driver: Actor1
System
driverAcc: DriverAccount
1 : getHomePage
2 : loadHomepage
3 : showHomepage
4 : changeStatus(s)
5 : statusCanBeChangedTo(s)
6 : result
opt ChangeStatus
[result == true]
7 : setStatus(s)
8 : sendChangedStatusNotification
[result == false]
9 : sendUnchangedStatusNotification

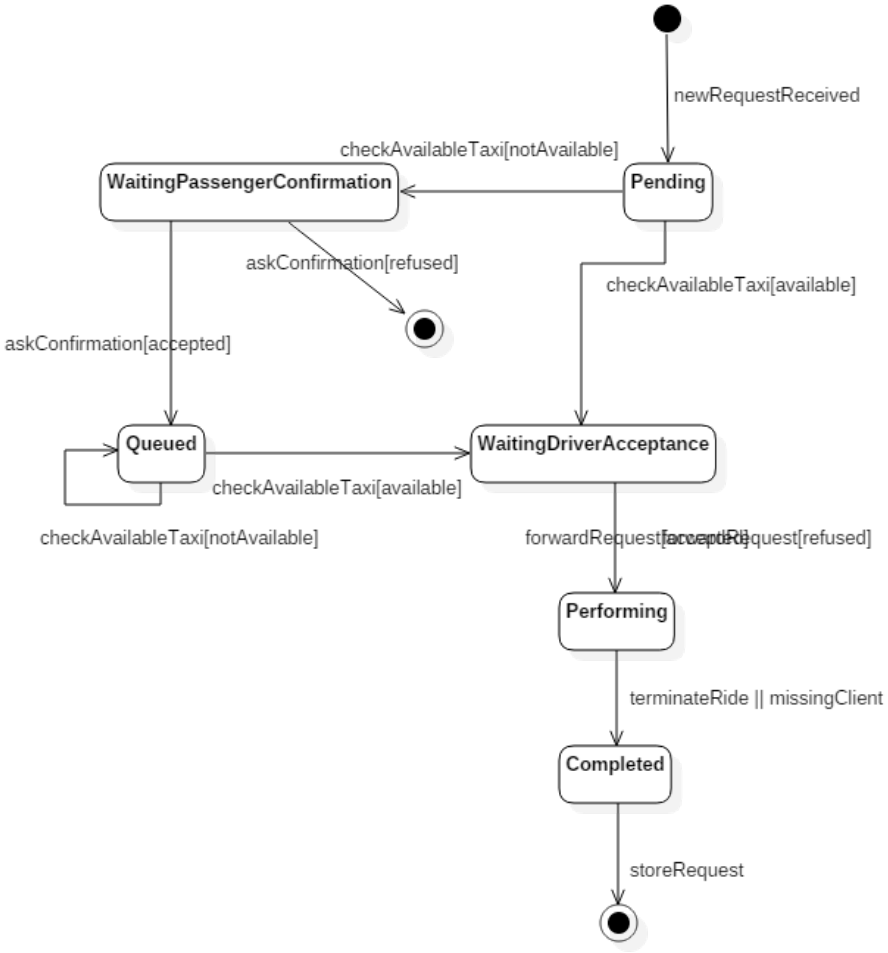
5.2.11 Driver user completes the current ride

Actor	[AC3] Driver User
Goal	[G16]
Input Conditions	1. The driver must be logged in; 2. The driver must be currently performing a ride; 3. The driver user has reached the meeting point with the client but can't find him, or has just successfully brought him to his destination;
Event Flow	1. From his main page the driver user presses "Missing client" if he hasn't found the user who requested the ride at the meeting point; otherwise he taps on "Terminate ride"; 2. The driver user's status is automatically set on "available";
Output Conditions	The driver user is now able to receive new ride and reserved ride requests.
Exceptions	No exceptions



5.3 Statechart Diagrams

Here we provide a statechart diagram describing the life cycle of the object "Ride Request" presented in the class diagram.



6 Appendix

6.1 Alloy

6.1.1 Alloy model

```
module myTaxiService

// ### Primitive types ###

abstract sig Boolean {}
one sig TRUE extends Boolean {}
one sig FALSE extends Boolean {}

sig Strings {}

sig Coordinate {
        isInside: lone CityZone
}

sig Date {}

// ### Signatures ###

abstract sig Account {
        username: Strings,
        email: Strings,
        validated: Boolean,
}

sig AdministratorAccount extends Account {}
{
        validated = TRUE
}

sig PassengerAccount extends Account {
        sends: set Request,
        hasReservationHistory: set Reservation,
        inWaitingQueue: lone CityZone
}

sig DriverAccount extends Account {
        available: Boolean,
        currentlyDriving: lone Taxi
}{
                validated = TRUE
        }

abstract sig Notification {
        refersTo: Request,
        receiver: Account
}

sig InformativeNotification extends Notification {}

sig AcceptRefuseNotification extends Notification {
        accepted: Boolean,
}

abstract sig Request {
        startingLocation: Coordinate,
        appointmentTime: Date,
        completed: Boolean,
        missingClient: Boolean,

        isAssociatedTo: lone DriverAccount
```

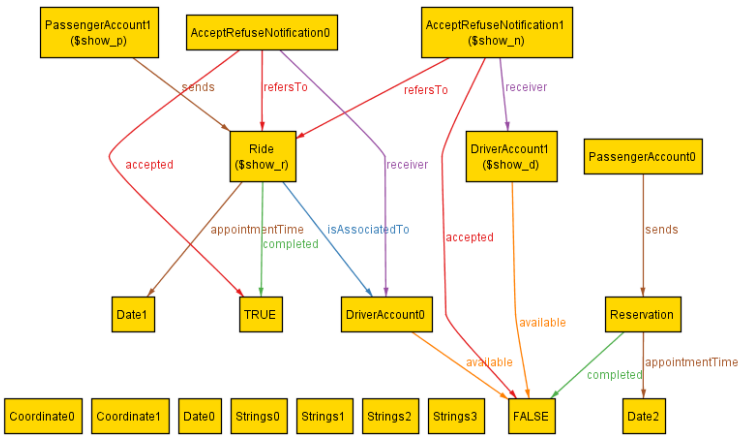
```
}

sig Ride extends Request {}

sig Reservation extends Request {
        endingLocation: Coordinate
}{
                startingLocation ≠ endingLocation
        }

sig CityZone {
        edges: some Coordinate,
        passengerQueue: set PassengerAccount,
        driverQueue: set DriverAccount
}{
                #edges ≥ 3
}

sig Taxi {
        taxiCode: Strings,
        currentPosition: Coordinate,

        currentlyIn: lone CityZone,
}

// ### Facts ###

fact RelationsCardinality{
        all r: Request | #r.~sends = 1
        all t: Taxi | #t.~currentlyDriving ≤ 1
}

//A passenger account that has not been validated yet
//cannot take part in any relations
fact NotValidatedAccountCannotDoNothing{
        no psgr: PassengerAccount | psgr.validated = FALSE
                and (#psgr.sends > 0 or #psgr.inWaitingQueue > 0
                        or #psgr.hasReservationHistory > 0)
}

//Usernames, email addresses and taxi IDs are unique
fact UniqueUsernameEmail{
        no disj a1, a2 : Account | a1.username = a2.username
                ∨ a1.email = a2.email
        no disj t1, t2: Taxi | t1.taxiCode = t2.taxiCode
}

//If a taxi is in a certain CityZone, its position must be inside that
//CityZone
fact TaxiPosition{
        all t: Taxi, z: CityZone | t.currentlyIn = z iff
                t.currentPosition.isInside = z
}

//Reservations that are inside the history of a PassengerAccount,
//must have that same passenger account as sender
fact ReservationHistoryConsistency{
        all p: PassengerAccount, r: Reservation | r in
p.hasReservationHistory implies r.(~sends) = p
}

//If a driver is availabe he must be driving a taxi
fact DriverAvailableIsDriving{
        all d: DriverAccount | d.available = TRUE implies
                #d.currentlyDriving = 1
}

//If a driver is inside the queue of a certain CityZone, he has to be
```

```
//available and must be driving a taxi that is inside that same CityZone
fact DriverQueueZone {
        all d: DriverAccount, z: CityZone | d in z.driverQueue implies
        ( #d.currentlyDriving = 1 and d.currentlyDriving.currentlyIn = z
        and d.available = TRUE )
}

//If a passenger is in the waiting queue of a certain zone, there must be
//a request, whose starting location is inside that CityZone, that is not
//yet completed and not associated to any driver
fact PassengerQueueZone{
        all p: PassengerAccount, z: CityZone | p in z.passengerQueue iff
                ( #incompleteRequestedRide[p] = 1 and
                #incompleteRequestedRide[p].isAssociatedTo = 0 and
                incompleteRequestedRide[p].startingLocation.isInside = z)
}

//For each passenger user, there can be at most one ride request
//not yet completed
fact PassengerIncompletedRide{
        all p: PassengerAccount | #incompleteRequestedRide[p] ≤ 1
}

//If a request is not yet completed then the missingClient field must
//be false
fact MissingClientIncompletedRequest{
        all r : Request | r.completed = FALSE implies r.missingClient = FALSE
        all r : Request | r.missingClient = TRUE implies r.completed = TRUE
}

//For each driver, there can be only one request associated to him that
// is not yet completed
fact IncompleteDriverRequests{
        no d: DriverAccount, disj r1, r2: Request | r1 in d.(~isAssociatedTo)
                and     r2 in d.(~isAssociatedTo) and r1.completed = FALSE
                and r2.completed = FALSE
}

//All the requests that are completed are associated to a driver
fact CompletedRequestDriver{
        all r: Request | r.completed = TRUE implies #r.isAssociatedTo = 1
}

//There cannot be two or more requests associated to a driver with the
// same appointmentTime
fact DuplicatedRequestDriver{
        no d: DriverAccount, disj r1, r2: Request | r1.isAssociatedTo = d and
        r2.isAssociatedTo = d and r1.appointmentTime = r2.appointmentTime
}

//There cannot be two or more requests sent by a passenger at the
//same appointmentTime
fact DuplicatedRequestDriver{
        no p: PassengerAccount, disj r1, r2: Request | r1 in p.sends and
                r2 in p.sends and r1.appointmentTime = r2.appointmentTime
}



//For each request associated to a driver there must be a notification
//sent to the same driver that has been accepted
fact NotificationForEachDriverRequest{
        all r: Request, d: DriverAccount | r.isAssociatedTo = d implies
                (one n: AcceptRefuseNotification | n.accepted = TRUE
                and n.receiver = r.isAssociatedTo and n.refersTo = r)
}

//For each reservation there cannot be a refused notification that
```

```
//refers to it
fact NoRefusedNotificationForAssignedReservation{
        all r: Reservation | ( no n: AcceptRefuseNotification |
        n.refersTo = r and n.accepted = FALSE )
}

// ### Functions ###
fun requestedRide[p: PassengerAccount] :
        set Ride { p.sends & Ride }

fun incompleteRequestedRide[p: PassengerAccount] :
        set Ride { requestedRide[p]  & {r: Ride | r.completed = FALSE} }

//### Predicates ###

//run show for 10 but 2 Notification, 2 Request, 0 Taxi, 1 CityZone,
//2 Coordinate
pred show{
        some d: DriverAccount, p:PassengerAccount, r: Request,
        n: AcceptRefuseNotification | r.isAssociatedTo ≠ d and
        r in p.sends and r.completed = TRUE and n.accepted = FALSE and
        n.refersTo = r and n.receiver = d
}

//run showDriverNotification for 10 but 1 PassengerAccount,
//1 DriverAccount,  2 Account, 1 Taxi, 1 CityZone, 1 Coordinate,
//1 Request, 1 Date, 1 Notification, 2 Strings
pred showDriverNotification [d: DriverAccount, r: Request]{
        r.isAssociatedTo = d
}

//run showTaxiQueueZone for 10 but 1 DriverAccount, 1 Account,
//0 Notification, 1 Strings, 3 Coordinate, 0 Date
pred showTaxiQueueZone [ d: DriverAccount, t: Taxi, z: CityZone ] {
        t.currentlyIn = z and d.currentlyDriving = t and d in z.driverQueue
        and #CityZone = 1 and #Taxi = 1
}

//run showPassengerQueueZone for 10 but 1 PassengerAccount,
//1 DriverAccount,  2 Account, 0 Notification, 0 Reservation, 0 Taxi,
//1 CityZone, 1 Ride, 1 Date, 3 Coordinate
pred showPassengerQueueZone [p: PassengerAccount, z: CityZone] {
        p in z.passengerQueue
}
```

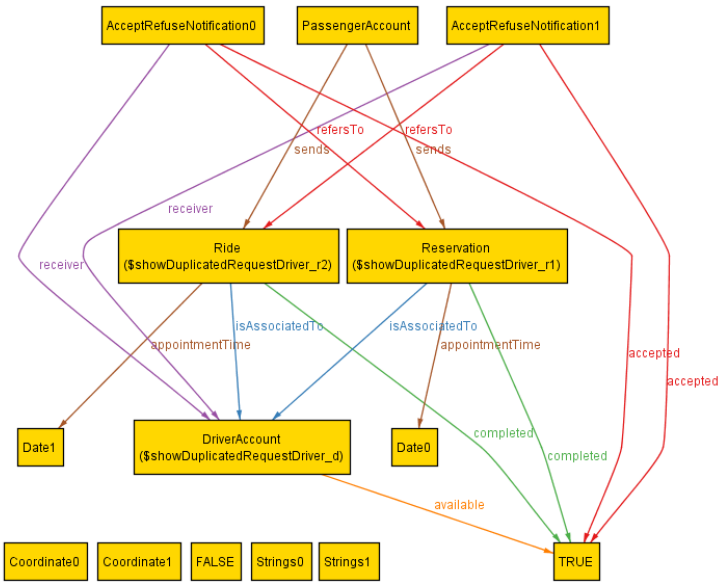
6.1.2 Generated worlds

The following is a general world, created with the predicate "show", that shows a scenario in which there is a ride request sent by a passenger which is "completed". The system has sent a notification to the driver DriverAccount1 who has decided to refuse it, so a new notification has been sent to another driver, DriverAccount0, who has accepted the ride, and the request has thus been associated to him. Some relationships have been hidden to allow a better readability.

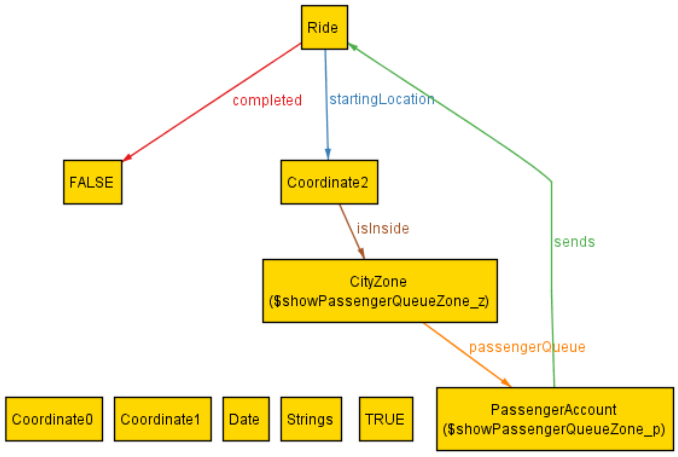


The following are some significant worlds generated by the Alloy analyzer that highlight some facts that hold in the model.

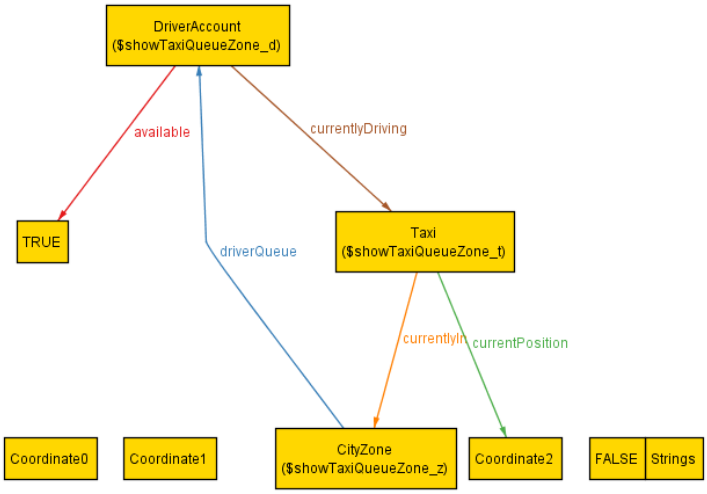
- This world, generated with "showDriverNotification" predicate, shows that if a passenger has sent a request and that request is associated to a driver, the system must have sent a notification to the driver associated to the request and the notification must have been accepted. Since a driver cannot take care of two requests at the same time, the appointment time of the two requests must be different.



- This world, generated with "showPassengerQueueZone" predicate, shows that if a passenger is in the waiting queue of a certain zone, he must have sent a ride request with a starting location that is located inside that zone.



- This world, generate with "showDriverQueueZone" predicate, shows that if a driver is inside the drivers' queue of a certain zone, he must be in "available" status and must be driving a taxi which is located in that city zone.



6.2 Softwares and tools used

- Google Docs (docs.google.com): to redact the document
- Lyx (lyx.org): to format the document
- StarUML (staruml.io): to create Use Case Diagrams, Class Diagrams, Sequence Diagrams and Statechart Diagrams
- Balsamiq (balsamiq.com): to create interface mockups
- Alloy Analyzer (alloy.mit.edu/alloy): to prove the consistency of our model

6.3 Hours of work

Here is how long it took to redact this document:

- Matteo Bulloni: ~33 hours
- Marco Cannici: ~33 hours

7 Changes made before DD

- Added functional requirements enumeration;
- Little change on [FR 4.1] , due to unfeasibility of the proposed feature (check [FR 4.1] for more details);