Politecnico Di Milano
A.A. 2015/2016
Software Engineering 2: ”MyTaxiService”
Requirements Analysis and Specifications Document

Paramithiotti Andrea (Matr. 788794)
Rompani Andrea (Matr. 854052)
Zoia Lorenzo (Matr. 852392)

v2.0

Contents

1 Introduction

1.1 Purpose

This document represents the Requirement Analysis and Specification Document (RASD). The main goal of this document is to completely describe the system in terms of functional and non-functional requirements found by analysing the real needs of the customer.
In addition, this document outlines the constraints and the limits of the software and describes the typical use cases that will occur after the deployment.

This document is intended for all the developers and for the system analysts that need to integrate other systems with ours. It also could be used as a contractual basis between the customer and the developers.

1.2 Scope

The aim of the project is to create a brand new system for the management and organization of a city taxi service. This system offers a mobile application and a web interface in order to give the customer the possibility to benefit from the taxi service. Furthermore, the system provides an additional communication interface for the taxi driver.
The mobile application and the web interface accept requests and reservations for taxis from the users, with the possibility to organise a share ride among different users. The taxi driver is supposed to communicate his availability, acceptances and rejections of requests through the communication interface.

The system is created to simplify the access of passengers to the service, and to guarantee a fair management of taxi queues.

1.3 Definitions, acronyms, and abbreviations

1.3.1 Definitions

- **Zone**: One of the parts of a city. Every city is divided in a certain number of zones
- **Queue**: every zone has a queue of taxies and the system chooses the right taxies for the requests from these queues
- **Normal ride**: a ride for a single person
- **Shared ride**: a ride for many people where the fee is divided
- **Reservation**: a booked ride
- **Skynet**: the class which contains all the information that are useful for the system in order to manage the service

1.3.2 Acronyms

- **RASD**: Requirements Analysis and Specification Document
- **DB**: Database
- **API**: Application Programming Interface
- **OS**: Operating System

1.4 Actors

- **Person**: A generic person who is neither registered nor authenticated.
- **Registered User**: A person who is registered to the service.
- **Authenticated User**: A person who is authenticated to the service.
- **Passenger**: A person who is currently on a taxi ride.
- **Taxi Driver**: The person who drives a taxi.
- **System**: The software that manages the application interaction between all the actors.
- **Operator:** A employee of the MyTaxiService company who is able to assist the taxi driver and communicate with the administrative authority

1.5 Goals

- Allow a user to register and authenticate to the service
- Allow the user to request a taxi. Inform the taxi driver of the request and the position of the passenger
- Manage the taxis in order to fulfil all the requests
- Allow the user to reserve a ride at a specific date and time
- Allow some users to share a ride on the same route and the related cost
- Allow a Taxi Driver to manage an emergency during a ride

1.6 Document Overview

- **Introduction**: it gives a description of a document and some basic information about the software
- **Overall Description**: it gives a general information about the software with more focus about the constraints and the assumptions
- **Specific Requirements**: this part lists the requirements, the typical scenarios and the use cases. This section contains also some UML diagrams in order to facilitate the understanding of every functionality
- **Appendix**: this part contains some information about the attached ".als" file and also some screenshots
- **Revision**: Here are described the changes compared to the previous versions

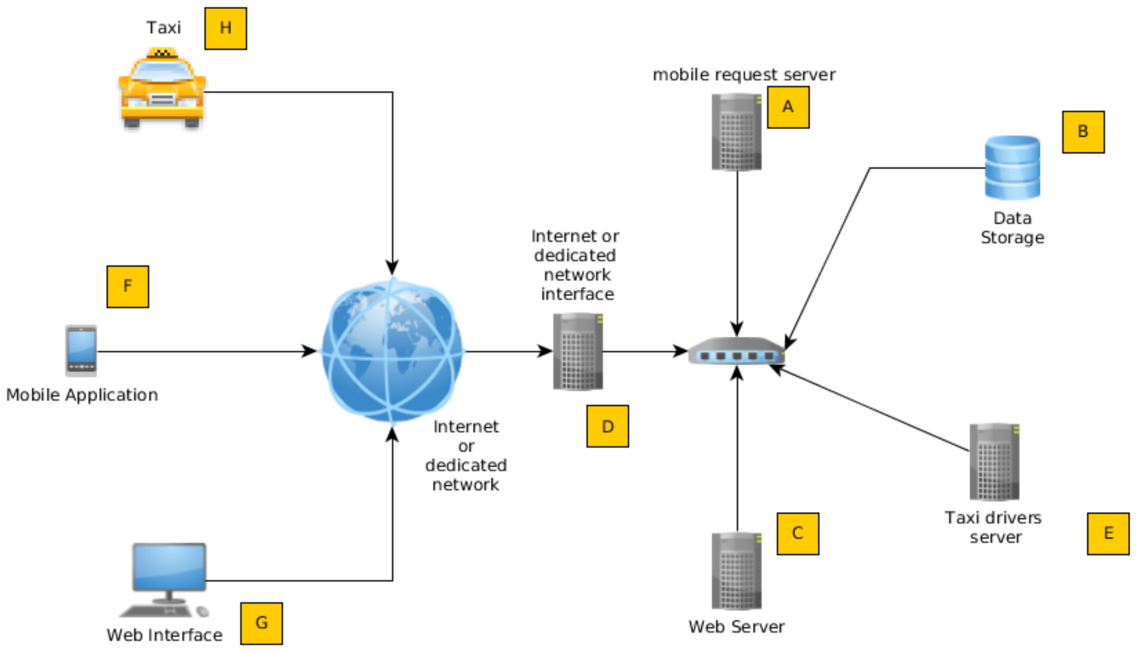
2 Overall Description

2.1 Product Perspective

The system is composed of:

A. A server that handles the requests from the mobile application
B. A data storage server used to store the needed data for the system to work properly
C. A web server that handles the requests from the web interface
D. An Internet interface used to link the system to the internet or a dedicated network
E. Taxi drivers server that handles the taxi drivers requests
F. A mobile application interface used to access the system services from an enabled mobile device
G. A web interface used to access the system services from an enabled web browser
H. A taxi terminal, a device that sends to the system the locations of the taxi and also acts as the driver's system interface

Diagram of the system showing its different parts and how they are connected.



2.1.1 System Interfaces

The system provides these interfaces:

- Taxi driver interface that:
    - Sends to the system the current location of the taxi
    - Accepts the commands concerning the taxi driver normal operations
- Mobile interface that:
    - Enables the registration of the users to the service
    - Enables the user authentication for the service access
    - Allows the user to request a taxi for a ride
    - Allows the user to reserve a taxi ride on a future date
    - Allows the user to engage in a shared taxi ride
- Web interface that:
    - Enables the user registration to the service
    - Enables the user authentication for the service access
    - Allows the user to request a taxi for a ride
    - Allows the user to reserve a taxi ride on a future date
    - Allows the user to engage in a shared taxi ride
- Internet or dedicated connection interface that:
    - Establishes the link from the mobile and web interfaces to the respective servers
    - Establishes the link from the taxi driver interfaces to the taxi drivers server

The system also provides a set of API to enable the expansion of the functionalities in the future

2.1.2 User Interfaces

The mobile interface:

- Must allow the user to insert string values and also, by using a map, to choose a location
- Must notify any error in an informative way, also showing the user a way to solve the error
- Any person with any school education level must be able use the mobile application services without any training

The web interface

- Must be compliant with the html 5 specifications and be readable and usable in any standardized resolution
- Must notify any error in an informative way, also showing the user a way to solve the error
- Any person with any school education level must be able use the web interface services without any training

The Taxi driver interface:

- To be able to use the driver’s specific services, the taxi driver’s interfaces must be easy to use also while driving (where highway code permits it) and while the taxi is still
- It must be able to send the position of the taxi to the system, manually or automatically
- Any taxi driver must learn every function in at most 1 hour of training

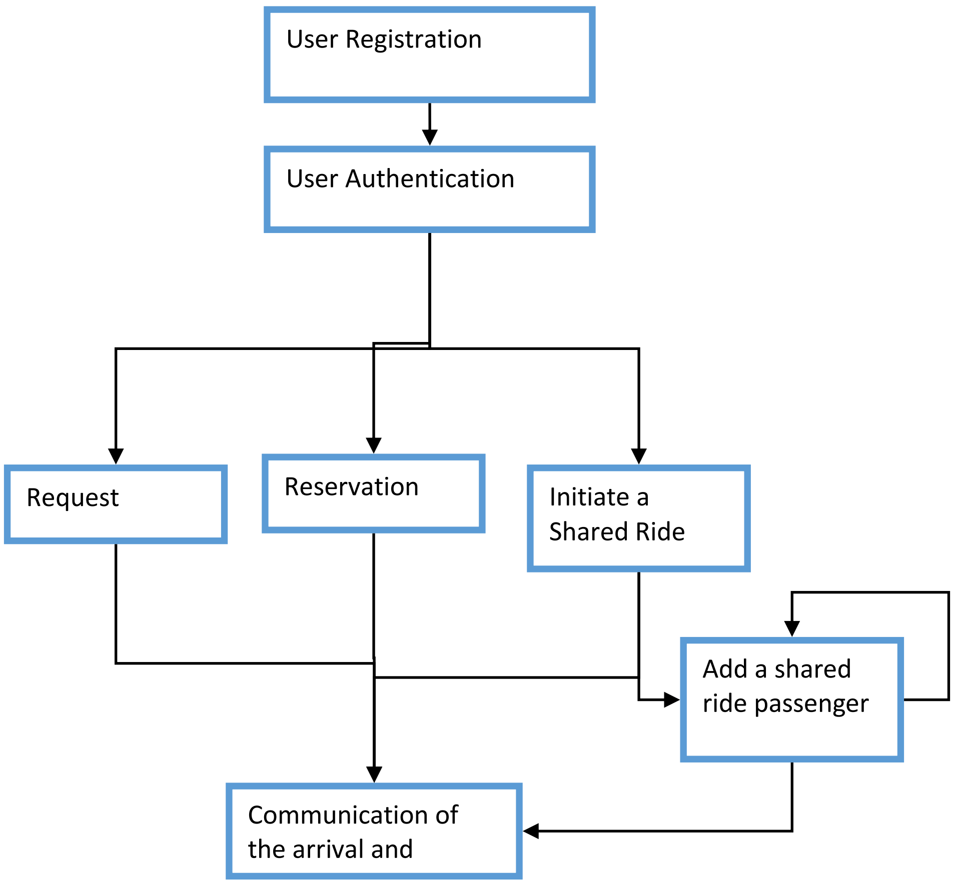
2.1.3 Communication Interfaces

In order to support the integration between the various parts of the system every protocol must use only standardized protocols to enable future modernization without compromising current operations

2.1.4 Operations

- The application is always interactive, except for the operation of the localization of the taxi
- The backup of the data must be done daily with the risk of losing at most 24 hours

2.2 Product Functions



2.3 User Characteristics

- Any technical expertise must not be necessary in order to access and use the mobile application and the web interface
- Every person with a primary school education level must be able to execute every action concerning the use of the operations in the taxi service
- The taxi drivers need to be instructed in order to be able to use their specific functionalities

2.4 Constraints

- Any request, reservation, and shared ride cannot be performed by a non authenticated user
- Any non authorized person cannot use the taxi driver specific functions
- Every taxi driver must be authenticated and authorized by the system administrator
- The system must always respond to a user’s request (positively or negatively)
- No downtime due to an internal system failure that is greater than 1% of the total time (of 1 year) is accepted
- Only a downtime (due to an internal error) smaller than the 1% of the total amount of working hours is accepted
- Two different users cannot simultaneously authenticate to the system with the same account
- If an area has a full queue, the system must send the taxi in the area with the least available taxies

2.5 Assumptions and Dependencies

- The mobile application must be executed and tested on Android, IOS and Windows Phone platforms
- The web interface must work correctly and be tested on any HTML 5 compliant browser
- The number of taxies must be minus or equal to the number of the zone multiplied for the maximum length of a taxies queue
- The number of taxies must be at least equal to the number of the zones
- The taxi driver can transport a number of people equal to the maximum number of people allowed by the certification of the vehicle
- The system does not manage the payment of a ride. It can only calculate the exact fee of each passenger during a shared ride. For all the other cases the payment must be performed, through the taximeter, directly by the taxi driver and the user. An automatic management of payments could be implemented in the future using the APIs

3 Specific Requirements

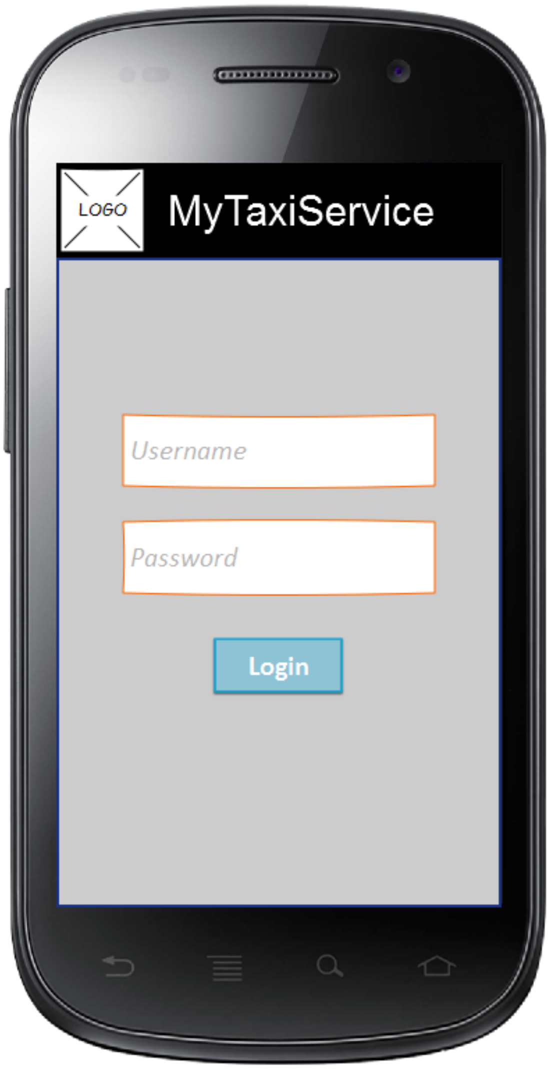
3.1 External Interfaces Requirements

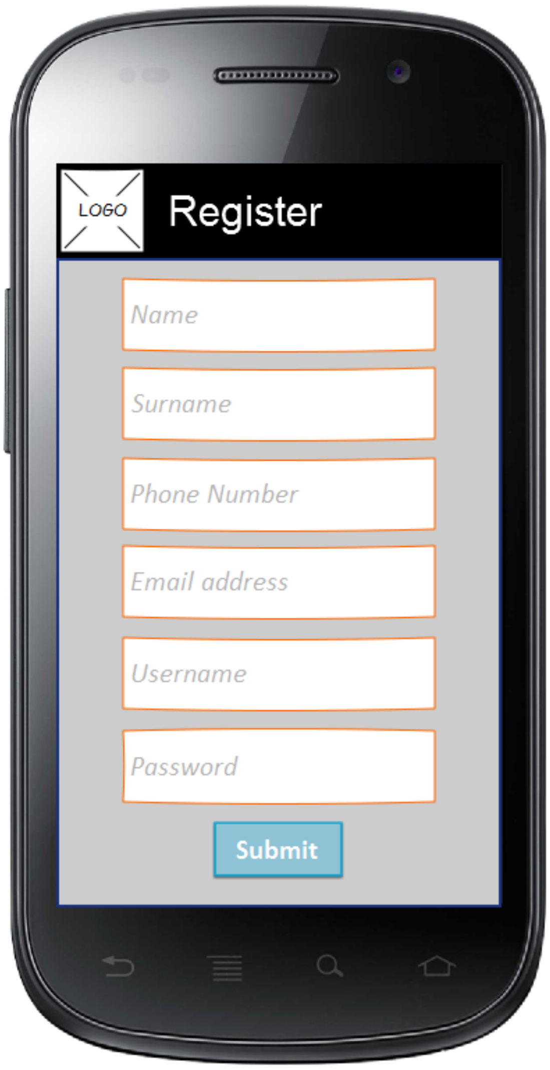
3.1.1 User Interfaces

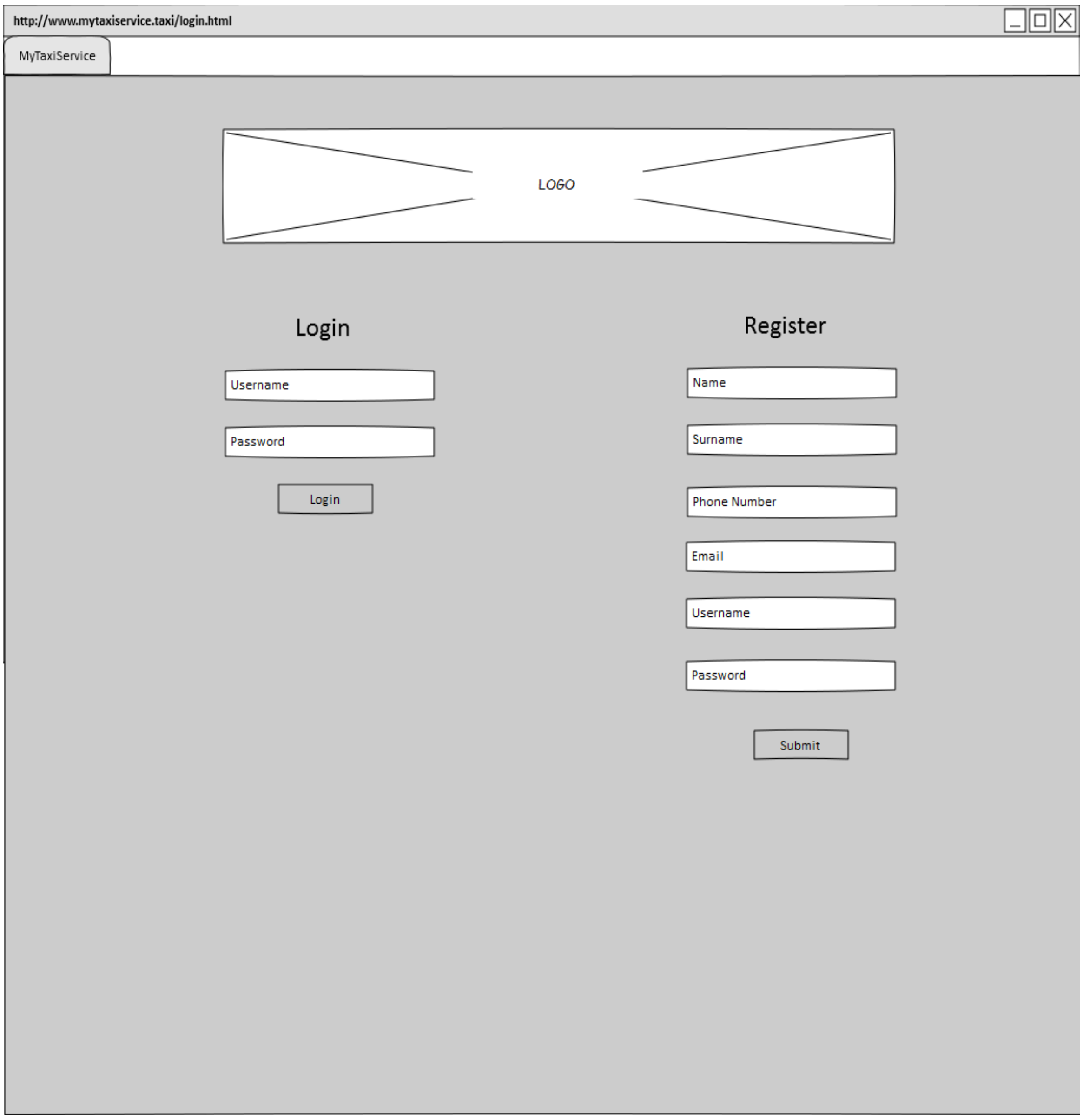
The mockups show the pages for the interaction with the users, through the web application or the mobile application.

3.1.1.1 Registration and Authentication

These mockups show the home page of the web application and the pages of the mobile application that allow a new user to register to the service or a registered user to authenticate.

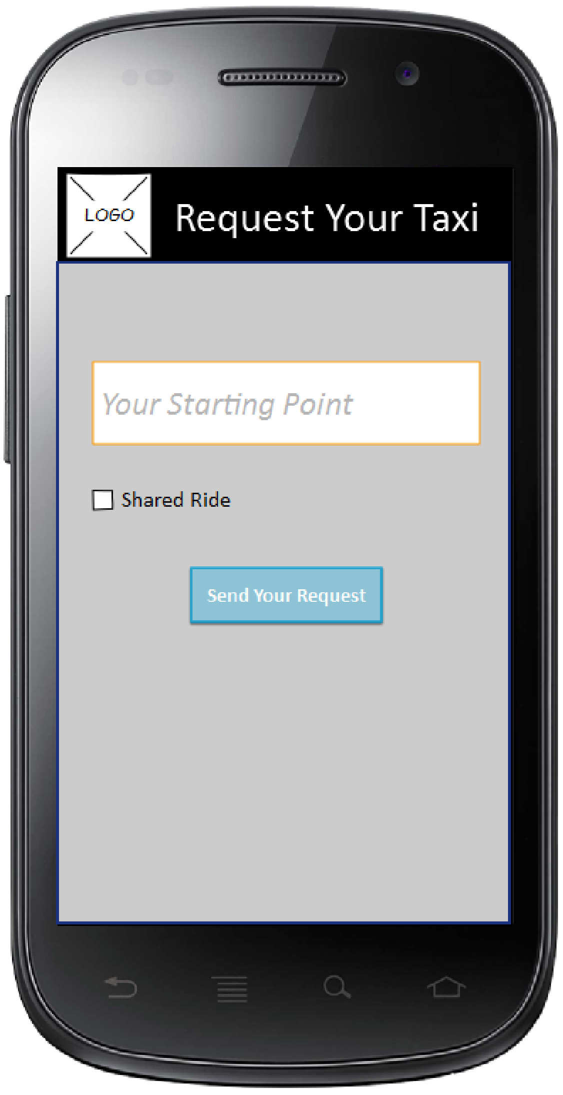


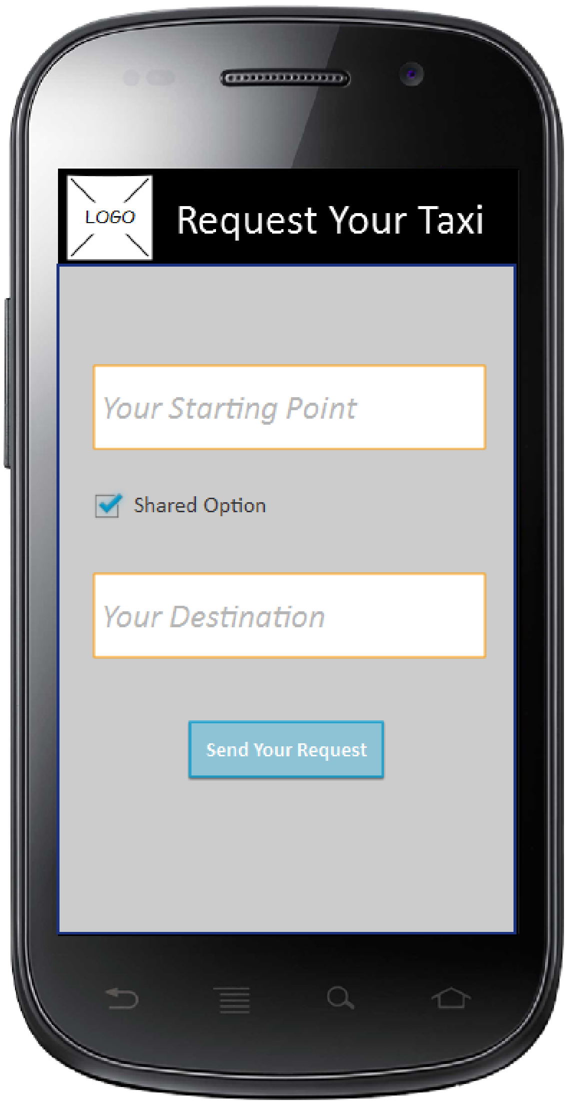


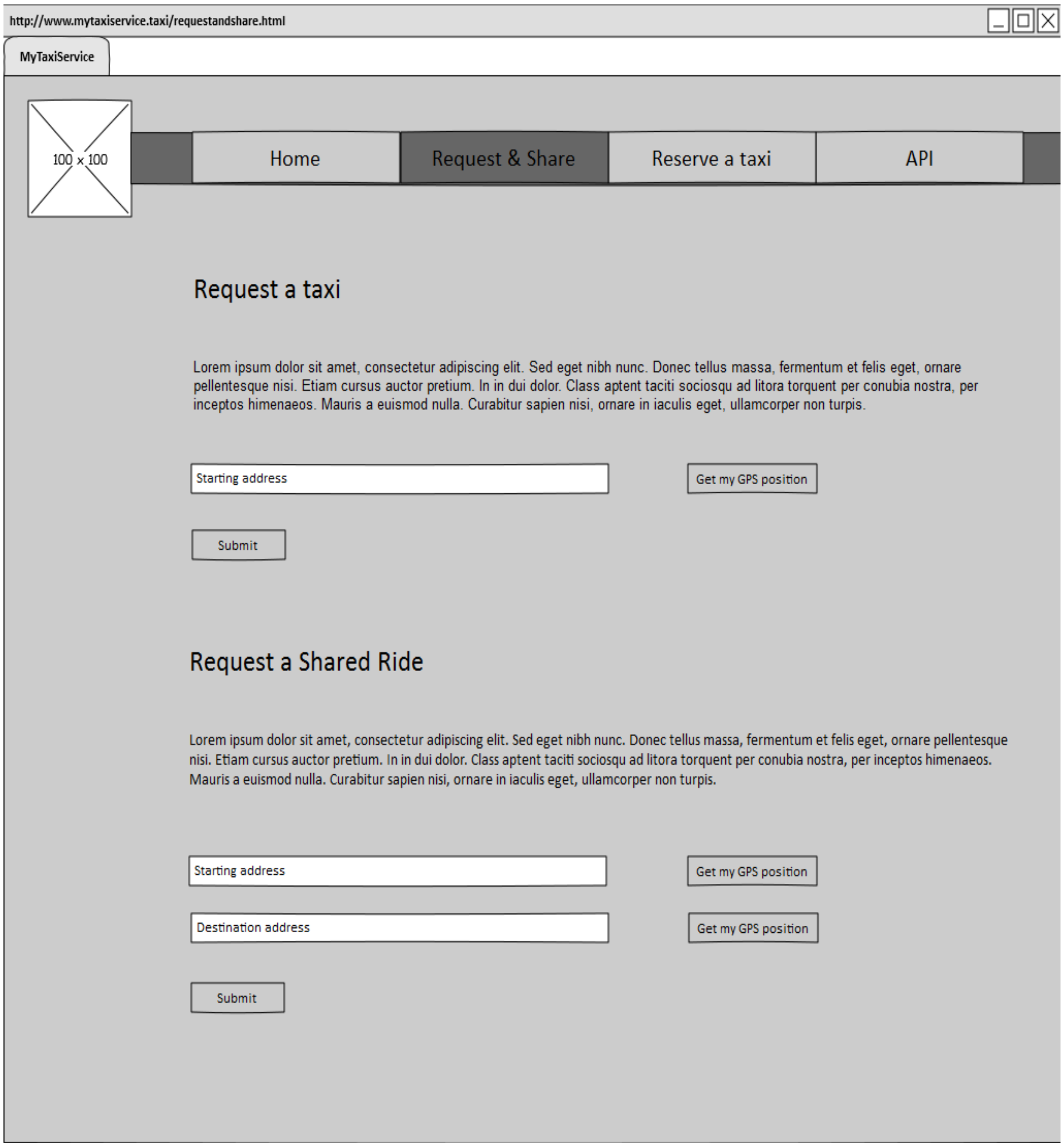

http://www.mytaxiservice.taxi/login.html
MyTaxiService
LOGO
Login
Username
Password
Login
Register
Name
Surname
Phone Number
Email
Username
Password
Submit

3.1.1.2 Request a taxi or a Shared ride

The mockups show the interfaces that allow the users to request a normal ride or a shared ride.
The two options are on the same page in the web application, while in the mobile application the user must select a flag in order to make the shared ride options appear.






http://www.mytaxiservice.taxi/requestandshare.html

Request a taxi

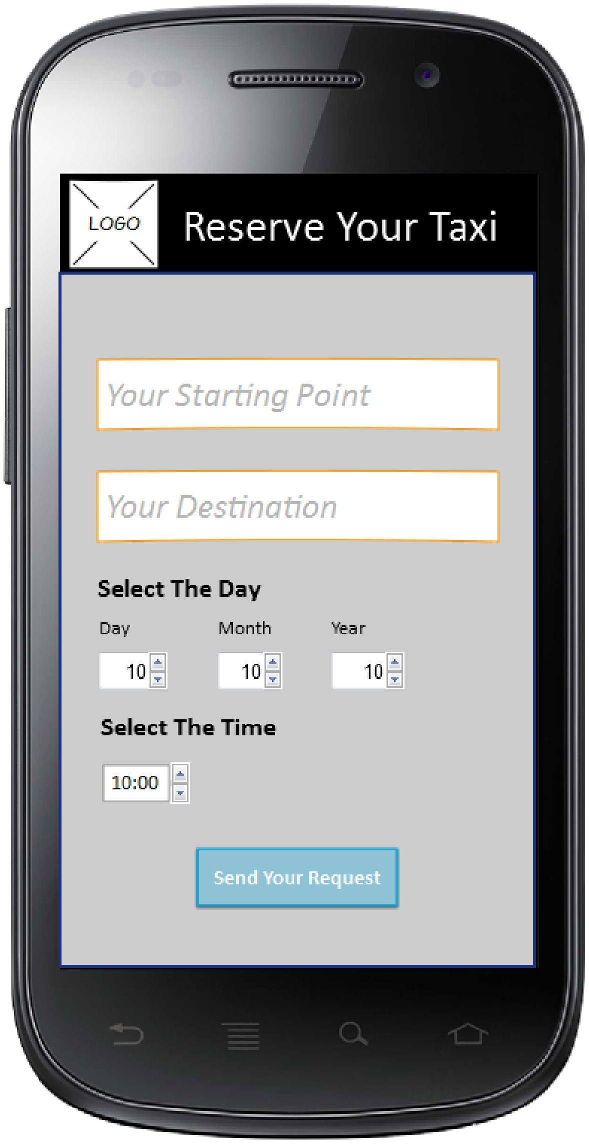
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed eget nibh nunc. Donec tellus massa, fermentum et felis eget, ornare pellentesque nisi. Etiam cursus auctor pretium. In in dui dolor. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Mauris a euismod nulla. Curabitur sapien nisi, ornare in iaculis eget, ullamcorper non turpis.

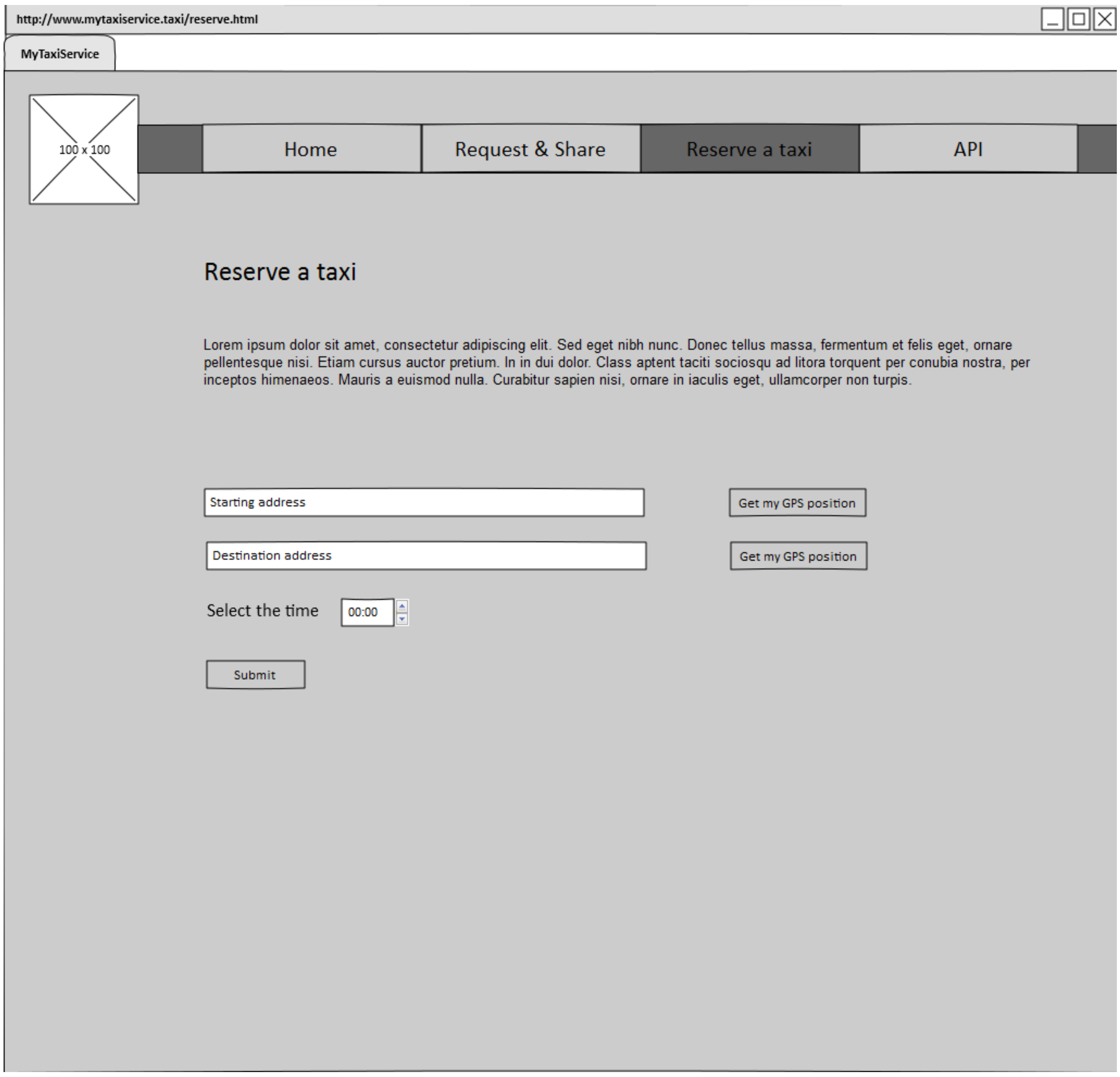
Request a Shared Ride

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed eget nibh nunc. Donec tellus massa, fermentum et felis eget, ornare pellentesque nisi. Etiam cursus auctor pretium. In in dui dolor. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Mauris a euismod nulla. Curabitur sapien nisi, ornare in iaculis eget, ullamcorper non turpis.

3.1.1.3 Reserve a taxi

The mockups show the interfaces that allow the users to make a reservation.





Reserve a taxi

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed eget nibh nunc. Donec tellus massa, fermentum et felis eget, ornare pellentesque nisi. Etiam cursus auctor pretium. In in dui dolor. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Mauris a euismod nulla. Curabitur sapien nisi, ornare in iaculis eget, ullamcorper non turpis.

Starting address | Get my GPS position

Destination address | Get my GPS position

Select the time: 00:00

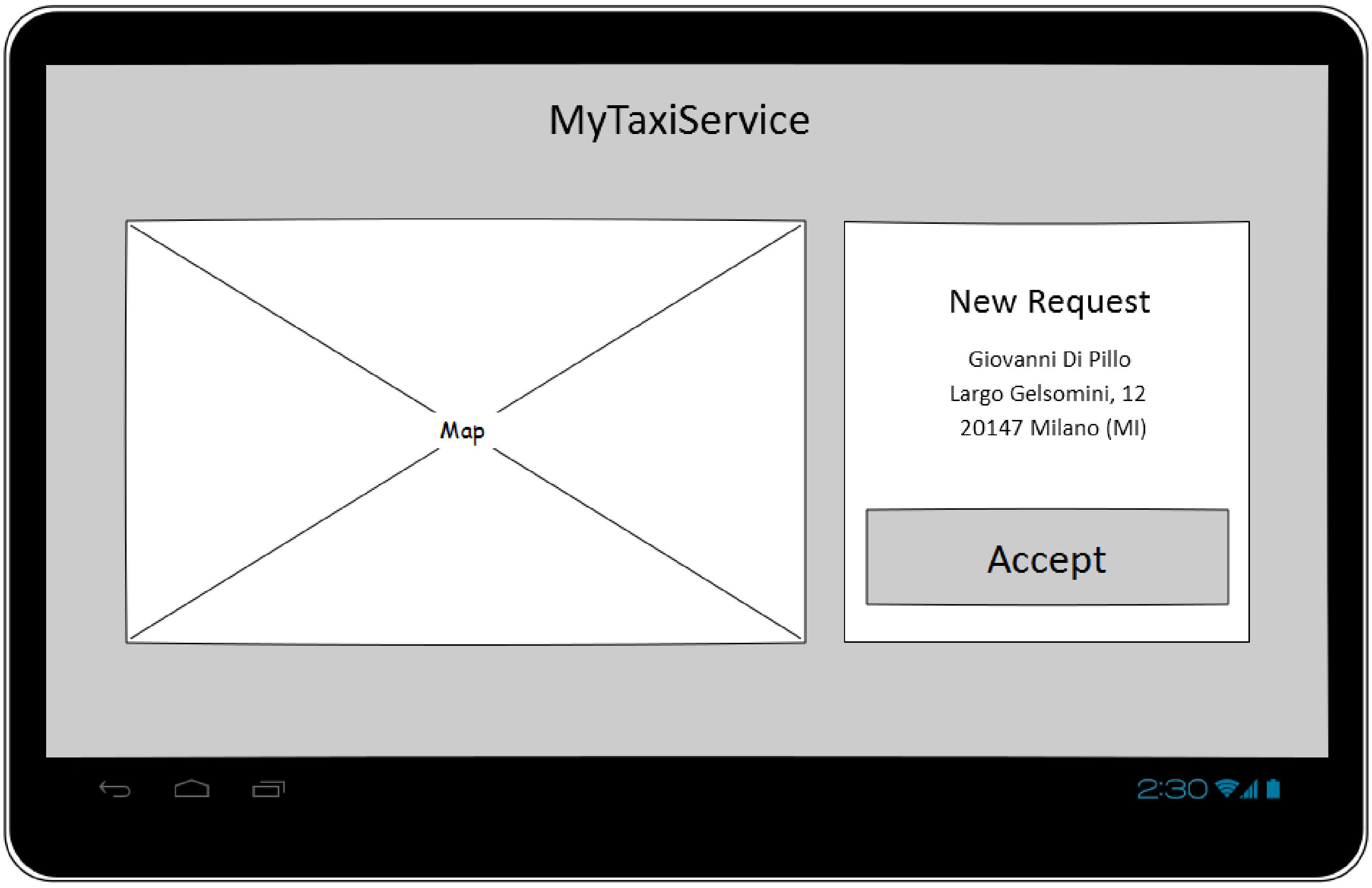
Submit

3.1.2 Taxi Driver Interfaces

The mockups show the pages on the mobile in the taxi for the interaction with the taxi driver.

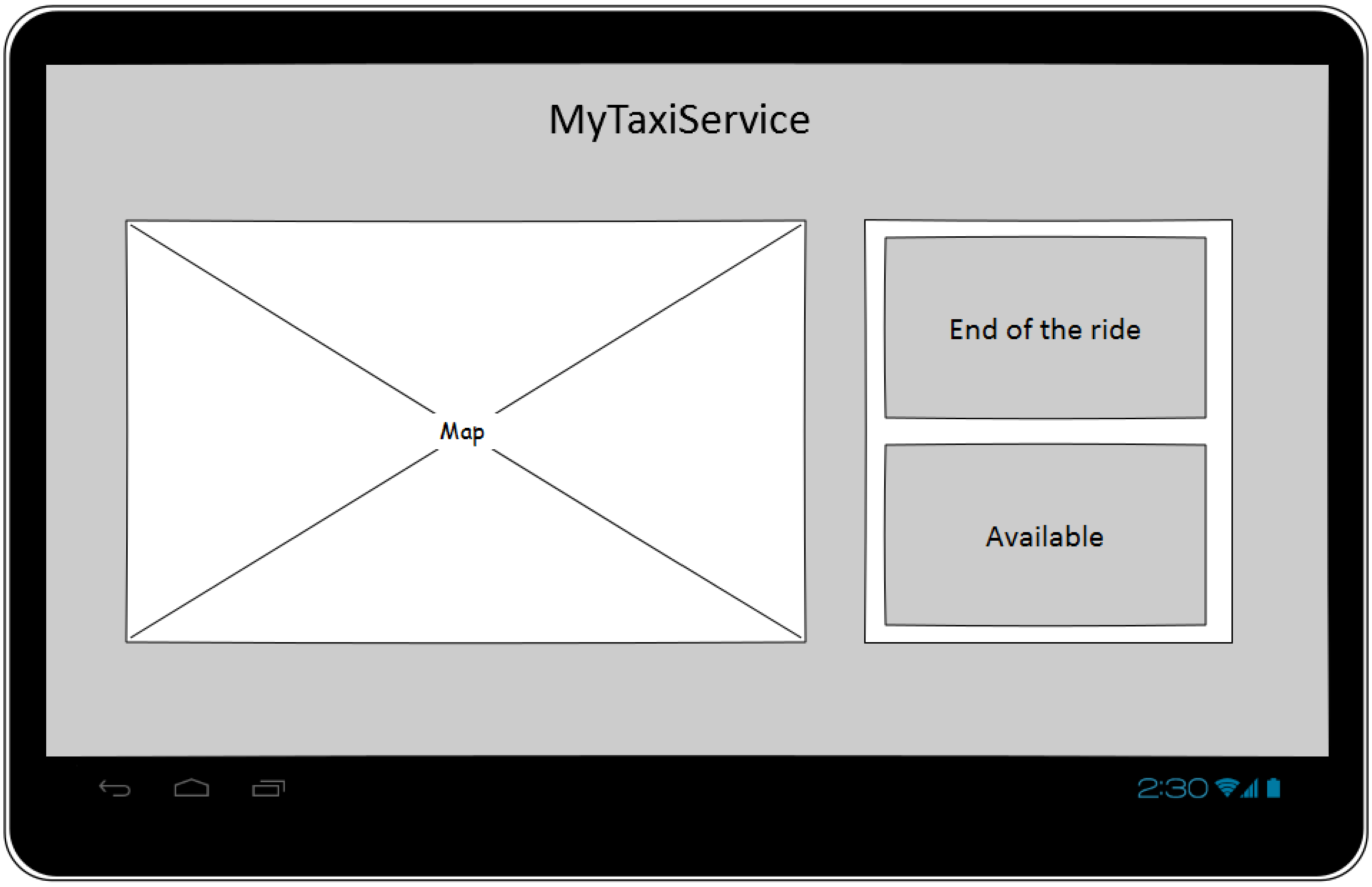
3.1.2.1 Accept a Request

The mockup shows the interface that allow the taxi driver to accept a request forwarded by the system.
The taxi driver visualizes a map with the position of the user and the information concerning the requested ride.
In order to accept the request the taxi driver must click on the button "Accept".



3.1.2.2 End of the Ride and Availability

The mockup shows the interface that is shown to the taxi driver while he is driving towards the destination with the passenger/s aboard. The map has a navigation function that guides the taxi driver and the buttons on the right are used to communicate that the ride is finished and to confirm the availability of the taxi after the ride.



3.1.3 API Interfaces

The web application offers a development area, indicated with “API” in the Index menu of the mockup.
In that area the developers can develop additional services on top of the basic ones.

3.1.4 Hardware Interfaces

Every taxi is prearranged with a special tablet that executes the software for the interaction with the taxi driver. An app that can perform phone calls must be installed in that table.
A generic personal computer or smartphone is sufficient for the interaction with the users.

3.2 Functional Requirements

3.2.1 Goal 1

Allow a user to register and authenticate to the service

Requirements

- The user must not be already registered
- The username chosen by the user in the registration phase must be unique
- The user must confirm his email address in order to become a registered user
- The authentication data inserted must be correct in order to authenticate
- If the user is not authenticated, he can only see the authentication/registration page

Domain Assumptions

- The personal data inserted in the registration phase must be correct and valid

3.2.2 Goal 2

Allow the user to request a taxi. Inform the taxi driver of the request and the position of the passenger

Requirements

- The user must be authenticated
- The taxi driver must accept the forwarded request using his mobile app
- The system must calculate the estimated time for the arrival of the taxi and inform the user
- The system must forward the request to an available taxi

Domain Assumptions

- The address inserted in the request must be correct and valid
- The user must get in the taxi by 10 minutes from its arrival
- The taxi must go to the requesting user as soon as he accepts the request

3.2.3 Goal 3

Manage the taxies in order to fulfil all the requests

Requirements

- The system puts the taxis at the bottom of the queue of the area where they finish the rides
- When a user makes a request, the system must send the nearest taxi available
- If there isn't any available taxi when a user makes a request, the system must warn the user
- A taxi driver can remove his availability and restore it whenever he wants, but that implies the loss of his priority in the queue

3.2.4 Goal 4

Allow the user to reserve a ride at a specific date and time

Requirements

- The user must be authenticated
- The system must dispatch a taxi for the reservation 10 minutes before the requested time
- If something happens to the taxi while driving towards the user, the system must warn the user
- The user must be able to cancel a reservation

Domain Assumptions

- The data inserted by the user must be valid and correct
- The taxi must be at the user's position at the requested time

3.2.5 Goal 5

Allow some users to share a ride on the same route and the related cost

Requirements

- The users must be authenticated
- In order to share a ride, the different users must be on the same route
- The system must calculate the fee for every user, dividing the total fee and weighting it on the amount of kilometres done by each of them
- There can't be more than 4 passengers in a taxi at the same time
- If a shared ride is made for a single user, the system calculates the fee as a normal ride

Domain Assumptions

- The data inserted by the users must be valid and correct

3.2.6 Goal 6

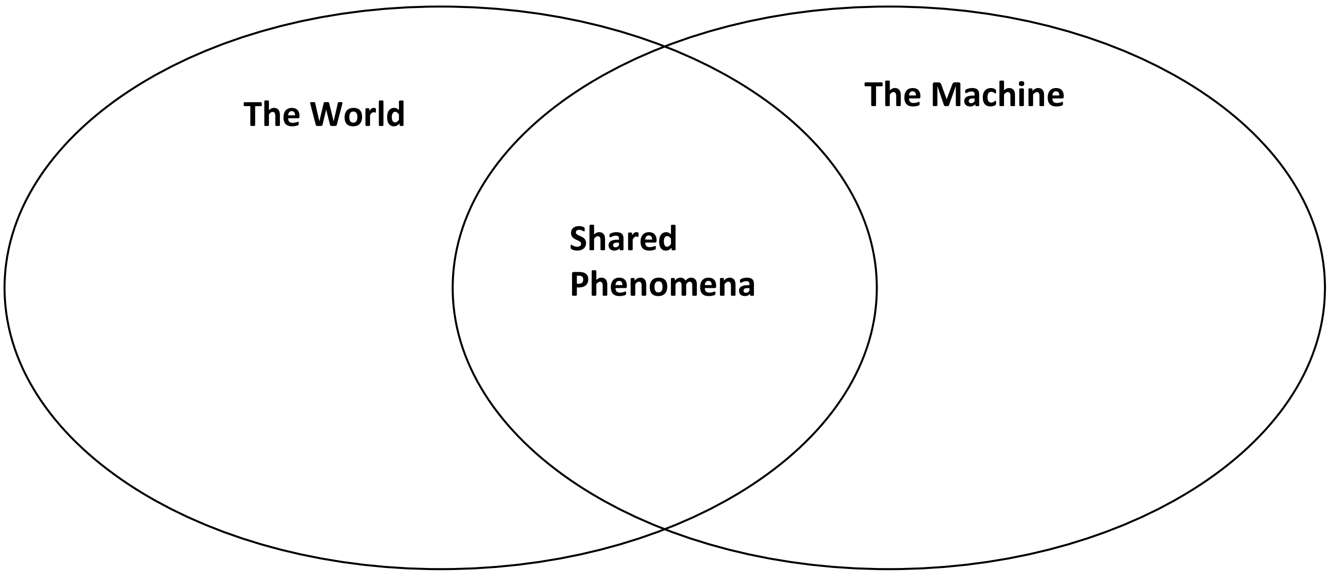
Allow a Taxi Driver to manage an emergency during a ride

Requirements

- Allow the taxi driver to call the emergency number and a tow truck
- Allow the taxi driver to cancel the current ride and, with the consent of the user, send a substitute taxi
- Allow the taxi driver to call an operator

3.3 The World and the Machine

For a first domain analysis of this application, we use “The World & Machine” model by M. Jackson & P. Zave. This approach let us divide the domain into three zones.
“The World” that contains all the phenomena that are observed only by the Environment.
“The Machine” that, otherwise, contains all the phenomena located entirely in the machine.
At the end, the “Shared Phenomena” that, as the name suggests, contains all the phenomena that are controlled by the world and observed by the machine or controlled by the machine and observed by the world.



3.3.1 The World

Contains all the goals and the domains properties described above.

3.3.2 Shared Phenomena

Contains all the requirements described above. The most important ones for the application are shown here.

- For every Request, Call and Reservation the System must find the nearest Taxi and send it to the User Location
- The System must save the information about all the Reservations in order to allocate a Taxi in time
- The System must store the Credentials of all Registered Users in order to allow them to perform the authentication
- The System must organize all the Taxis using a queue for every zone

3.3.3 The Machine

- The System must have a GPS technology in order to perform the fee calculation

3.4 Scenarios

3.4.1 Scenario 1

Michael J. wants to go to the car park in order to meet Hemmet, but it is very far from him. He remembers that he has just installed a new taxi application called MyTaxiService. Therefore, he opens this new application but he realizes that the application needs a login. For this reason, considering that he has no credentials for the login, he decides to do the registration. He compiles all the mandatory fields and waits for the confirmation from the service. Once received, he can finally log in the application and request a Taxi.

3.4.2 Scenario 2

John has an appointment to the doctor and his car is broken, so he decides to use his new application to call a taxi and get to the studio.
He authenticates in the website and requests a taxi in the dedicated section. He fills the blank spaces with the mandatory information and waits for a confirmation. The system receives the request and forwards it to the first taxi in the queue of the area where John is. The taxi driver accepts the requests using his mobile application and starts driving towards John’s house. The system calculates the estimated time and warns John with a message. After a few minutes John gets in the taxi and communicates to the taxi driver his destination. The taxi driver is really fast so he gets to the doctor in a few minutes. John pays the fee and gets off the taxi, then the taxi driver communicates to the system that he is free again using the mobile application. The system knows that the queue in that area is already full, so it sends the taxi driver in another area where the queue is empty and it is more likely that his service will be needed.

3.4.3 Scenario 3

Dominic needs to go to Letty's home tomorrow at 7 pm because he wants to go racing with their friend Brian, so he reserves a taxi by using the mobile app on his internet connected device.
The system confirms the reservation to Dominic.
The next day at 6:50 pm the taxi arrives at Dominic's, but Dominic loses track of time and forgets the reservation and at 7:00 because he was weightlifting.
The driver waits until 7:05 then he goes away and communicates to the system that Dominic missed the appointment.
Letty, knowing Dominic's love of himself, calls him to check if he's on the way to her home.
Discovering what happened she decided to ask Brian to get to Dominic's at full speed.

3.4.4 Scenario 4

Isaac needs to call a taxi in order to get to the city center from his house. This month he has bought many expensive things at home so he would like to spend few money for the ride. He knows that there is the possibility to share a taxi ride with other users so he decides to try this particular service.
He uses his mobile application and selects the "Shared ride" option, then he inserts his address and the road where he wants to be taken to.
The system sends the request for the shared ride to the first taxi in Isaac's zone and the taxi driver accepts by using the mobile in the taxi.
The system calculates the estimated time for the taxi to get to Isaac's, then shows Isaac a message on the phone to inform him that his taxi will be there in a maximum of 12 minutes.
After exactly 12 minutes Isaac gets in the taxi and starts to play with his mobile phone.
In the meantime Lucy, who lives not too far from Isaac's, requests a shared ride through the web application because she needs to go shopping. She's on the route that Isaac is doing and her destination is two roads far from Isaac's. The system warns the taxi driver that another user has requested a shared ride and the address is near his position, so he accepts the request and makes a little detour to Lucy's. Meanwhile the system tells Lucy that the taxi is almost there, so she goes on the street.
After 2 minutes the taxi pulls over and lets Lucy get in. Both Lucy and Isaac are happy to share their ride and they get on very well too. When the taxi arrives to the destination requested by Isaac, the system calculates his fee so that he can pay the taxi driver. After a pair of minutes also Lucy is arrived and pays her part to the taxi driver.
The taxi is now free again, so the taxi driver informs the system that he is available again through the mobile app on the taxi.

3.4.5 Scenario 5

Gigi has a tennis lesson at 9 pm, so he uses his mobile application and selects the "Shared ride" option, then he inserts his address and the road where he wants to be taken to.
He wants for the taxi to be in front of his house at 8.30 pm, because he knows that a shared ride might take a while more than a normal request, but he still hopes that he will get someone on the road so that he can pay a bit less.
The system tries to send the request for the shared ride to the first taxi in Gigi's zone, but the queue for that zone is empty, so it forwards it to the first in the queue of the zone nearby. The taxi driver accepts by using the mobile in the taxi.
The system calculates the estimated time for the taxi to get to Gigi's, then shows him a message on the phone to inform him that his taxi will be there in a maximum of 15 minutes.
After 15 minutes Gigi gets in the taxi, which is still empty.
Unfortunately the taxi driver doesn't receive any other request during the ride, so Gigi gets to the tennis court without sharing the ride with anyone. The system calculates his fee, which corresponds to the total fee for the ride.

3.4.6 Scenario 6

Silvestro wants to go to the gym but his wife has taken the car so he needs to take a taxi.
He decides to share the ride because he hopes to meet someone else who is going to the same gym. He uses his mobile application and selects the “Shared ride” option, then he inserts his address and the road where he wants to be taken to. The system forwards the request to a taxi driver who is already driving in that area with a ”shared client” aboard, he accepts and the system sends Silvestro a message to inform him that the taxi would arrive in 5 minutes. Silvestro gets in the taxi after 6 minutes and is really happy to see that another client is already on the taxi and that he is going in the same gym where he trains. At the end of the ride the system calculates the fee of the two passengers, they pay and they get off the taxi together.

3.4.7 Scenario 7

Gigi finds place on a taxi and is going to meet his grandmother because today it is her birthday. The taxi is going straight on Zara Street when suddenly a van crosses the road with a red light. The taxi driver cannot avoid the crash, makes a hard braking and crashes into the side of the van.
Everyone is fine, but Gigi’s head hurts a little. Fortunately, the tablet is not broken, so the taxi driver pushes the emergency button and selects the option “Call 112”. The tablet makes the call to the emergency number and the taxi driver communicates what has happened. At the end an ambulance and a patrol are dispatched and sent to their position, so that Gigi can be brought to an hospital and the others involved can be seen by a doctor. The taxi driver now selects the options “End Ride” and “Call a tow truck” and has the taxi taken to a mechanic.

3.5 UML Models

3.5.1 Use Cases

3.5.1.1 Use Case 1

Use Case Name

Register to the service

Actors

- Person
- System

Goal

- Allow a user to register and authenticate to the service (this use case speaks only about the registration part)

Input Condition

- None

Event Flow

- The person gets access to the service
- The person inserts his name, surname, email, phone number, desired username and password
- The system accepts the data and confirms the registration

Output Condition

- The person becomes a registered user

Exceptions

- If the person is already registered, the system shows an error message and shows the authentication form again
- If the desired username already exists, the system asks for another one

Special Requirements

- The password must contain at least one number, one capital letter and one symbol and must be at least 8 characters long

3.5.1.2 Use Case 2

Use Case Name

Authenticate to the service

Actors

- Registered User
- System

Goal

- Allow a user to register and authenticate to the service (this use case speaks only about the registration part)

Entry Condition

- Null

Flow of Events

- The Registered User inserts the credentials for the log in
- The System checks the credentials and accepts the data

Exit Condition

- The Registered User is now an Authenticated User

Exceptions

- If the system refuses the credentials, it must send a notification to the Registered User in order to repeat the authentication procedure

3.5.1.3 Use Case 3

Use case name

Request a taxi

Actors

- System
- Taxi driver
- Authenticated User

Goal

- Allow the user to request a taxi. inform the taxi driver of the request and the position of the passenger

Input condition

- Null

Event flow

- The authenticated user fills in the forms in the web application or the mobile app with the address where he wants the taxi to be sent
- The system forwards the request to the first taxi in the queue of the area where the passenger is
- The taxi driver confirms the request using the mobile app
- The system informs the authenticated user about the code of the incoming taxi and the waiting time
- The taxi gets to the authenticated user

Output condition

- The authenticated user gets in the taxi

Exceptions

- If the authenticated user gives an invalid address, the system shows an error message and asks for a new address
- If there isn’t any free taxi in the area where the passenger is, the system selects the first taxi from the nearest area
- If there isn’t any free taxi at all, the system doesn’t allow the request and shows an error message
- If the taxi driver doesn’t confirm the request, the system forwards the request to another taxi driver
- If the taxi doesn’t manage to get to the passenger (on time/at all), the system informs the passenger, who must call another taxi

Special Requirements

- The city is already divided into Taxi Zones (approximately 2 km^2 each)
- For each zone there is a queue of taxi (it can be empty if there aren't any taxies in its zone)
- Every Taxi and taxi Driver must have an identification code

3.5.1.4 Use Case 4

Use Case Name

Manage the Taxi

Goal

- Manage the taxis in order to fulfil all the requests

Actors

- System
- Taxi Driver
- Taxi GPS Locator

Entry Condition

- A Taxi Driver has communicated to the System his availability. This situation can happen either when a taxi driver finishes a service or when he starts his work session

Flow of Events

- The System receives the GPS information from the Taxi GPS Locator
- The System calculates the zone in which the Taxi is waiting
- The System stores the identifier of the Tax into the queue that contains all the taxies of the particular zone as the last element

Exit Condition

- The Taxi Driver waits for a request for a new service from the System

Exceptions

- If the System doesn't receives the GPS Information from the Taxi GPS Locator, it has to send a notification to the Drive
- If a particular queue is full, the system automatically stores the Taxi into a different queue in order to equally distribute the taxis

in the city. Then send a notification to the Taxi Driver and he moves to the new zone

Special Requirements

- The city is already dived into Taxi Zone (approximately 2 km^2 each)
- For each zone there is a queue of taxi (it can be empty if there aren't any taxies in its zone)
- The Taxi Driver has to use a Mobile App for sending his availability
- Each Taxi and Drives must have an identification code

3.5.1.5 Use Case 5

Use case name

Reserving a ride

Goal

- Allow the user to reserve a ride at a specific date and time

Actors

- User
- System

Entry condition

- An authenticated user wants to make a reservation for a taxi

Flow of events

- User chooses a time and a date for the reservation and communicates it to the system
- The system checks the parameters of the reservation sent by the user
- The system confirms the reservation to the user

Exceptions

- If the system doesn't receive the complete information (at least date and time, starting location, destination) to reserve a ride the user receives a communication detailing the incompleteness error and the reservation process is restarted from the beginning
- If the date and time received from the user are in the past or less than 2 hours in the future then the user will receive a communication detailing the date error and the reservation process is restarted from the beginning
- If the user sends as starting location or a destination that aren't real addresses, the reservation process is stopped and restarted,

communicating to the user the wrong location error
- If the user doesn't show up at the agreed time, the taxi driver will wait for a maximum of 5 minutes then it's free to accept another request
- If no taxi is available due to a strike known in advance and in place at the time of the reservation then the user will be warned and prevents him to make a reservation for the strike day

Special Requirements

- The system must always return some information even if there is an error
- The system must accept reservations 24 hours a day

3.5.1.6 Use Case 6

Use Case Name

Request a shared ride

Goal

- Allow some users to share a ride on the same route and the related cost

Actors

- System
- Authenticated user/s
- Passenger/s
- Taxi driver

Entry Condition

- An authenticated user requests a shared ride

Flow of Events

- The authenticated user specifies the starting address and the destination address
- The system forwards the request to the first taxi from the queue of the area where the authenticated user is
- The taxi driver confirms the request using the mobile app
- The system informs the authenticated user about the code of the incoming taxi and the waiting time
- The taxi gets to the authenticated user
- The authenticated user gets in the taxi and becomes a passenger
- Other authenticated users who requested a shared ride get in the taxi during the ride
- The passenger gets to his destination
- The system calculates the passenger's fee

Exit Condition

- The passenger pays his part of the fee and gets off the taxi

Exceptions

- If there aren't other users who requested a shared ride, the passenger pays the full fee
- If the authenticated user who requests a shared ride is on the route of another shared ride already in progress, the system doesn't get the first taxi in the queue and sends that taxi instead
- If the authenticated user gives an invalid address, the system shows an error message and asks for a new address
- If there isn't any free taxi in the area where the passenger is, the system selects the first taxi from the nearest area
- If there isn't any free taxi at all, the system doesn't allow the request and shows an error message
- If the taxi driver doesn't confirm the request, the system forwards the request to another taxi driver
- If the taxi doesn't manage to get to the passenger (on time/at all), the system informs the passenger and sends another taxi

Special Requirements

- The city is already divided into Taxi Zones (approximately 2 km^2 each).
- For each zone there is a queue of taxi (it can be empty if there aren't any taxies in its zone).
- The Taxi Driver must use a Mobile App for sending his availability.
- Every Taxi and taxi Driver must have an identification code.

3.5.1.7 Use Case 7

Use Case Name

Manage an emergency

Actors

- Registered User
- System
- Taxi Driver

Goal

- Allow a Taxi Driver to manage an emergency during a ride

Entry Conditions

- The taxi driver is making a ride an suddenly an Emergency happens, so he is unable to continue the ride

Flow of Events

- The taxi driver pushes the emergency button on the tablet
- The taxi driver selects the option “End Ride”
- The system cancels the current ride
- The passenger wants an new taxi in order to continue his ride, so the taxi driver sends a request for a new taxi to the system by selecting the right option
- The system creates a new request, automatically executes it and sends a new to taxi to their position
- The taxi driver selects the option that can solve the emergency

Exit Condition

- The emergency is solved and the passenger gets in another taxi

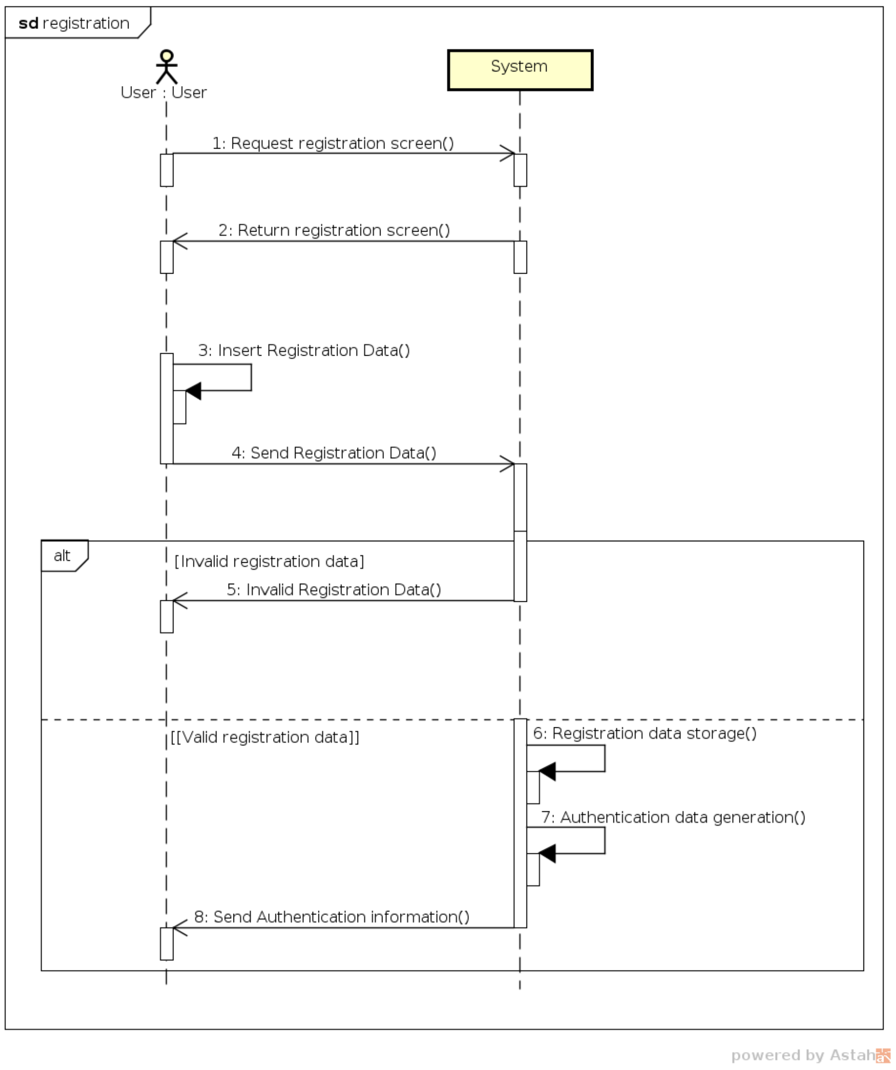
Exceptions

- If the passenger doesn’t want another taxi, he/she can go away
- If the emergency is not caused by the passenger, he/she is not supposed to pay the fee

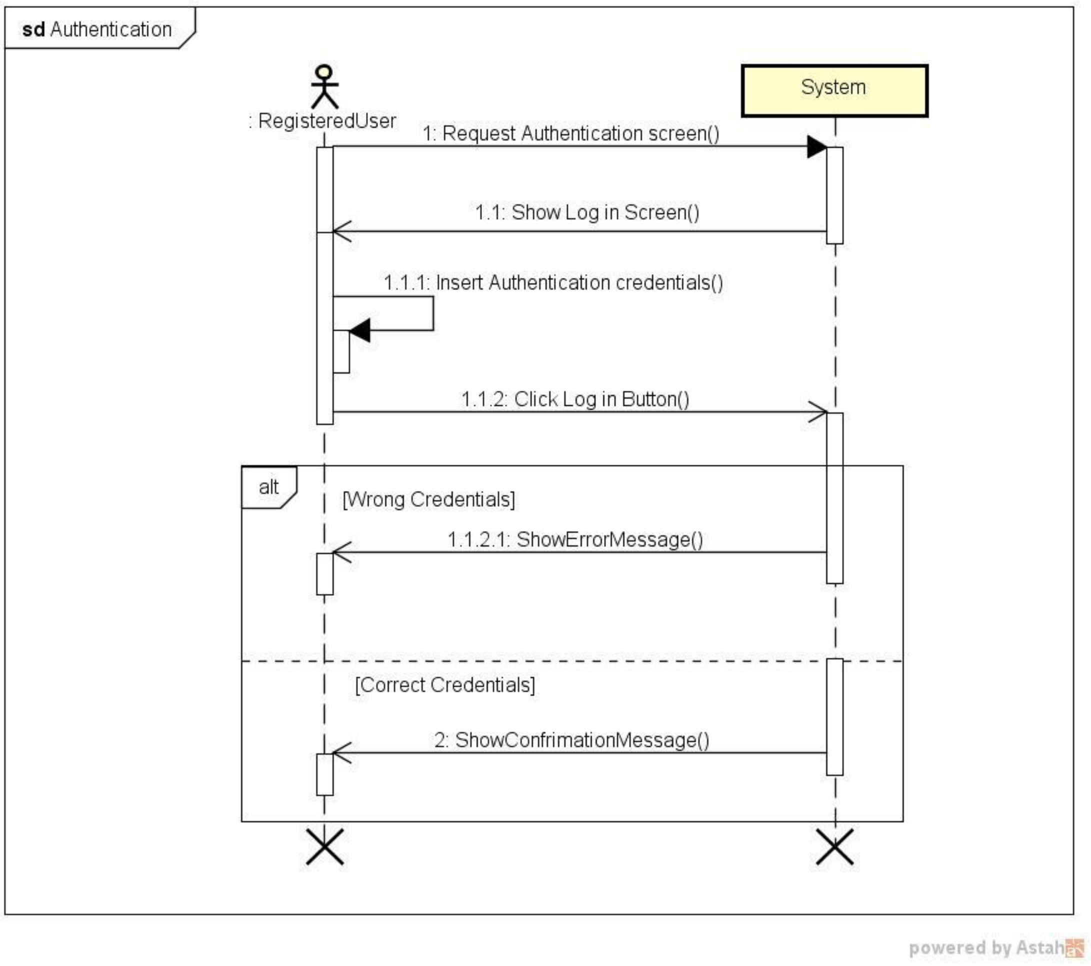
3.5.2 Sequence Diagram

Each sequence diagram is related to the use case with the same number.

3.5.2.1 Sequence 1

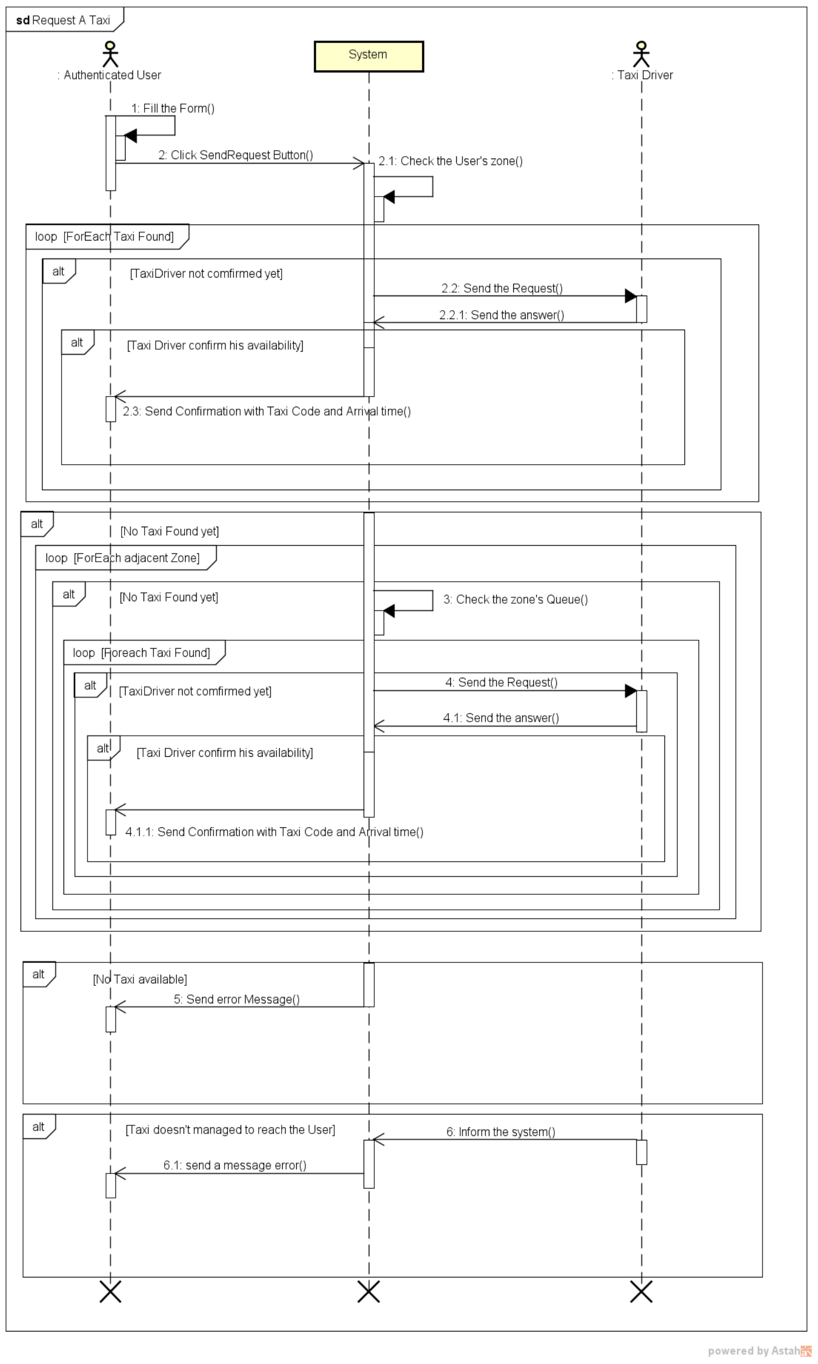

sd registration
User : User
System
1: Request registration screen()
2: Return registration screen()
3: Insert Registration Data()
4: Send Registration Data()
alt
[Invalid registration data]
5: Invalid Registration Data()
[[Valid registration data]]
6: Registration data storage()
7: Authentication data generation()
8: Send Authentication information()
powered by Astah

3.5.2.2 Sequence 2

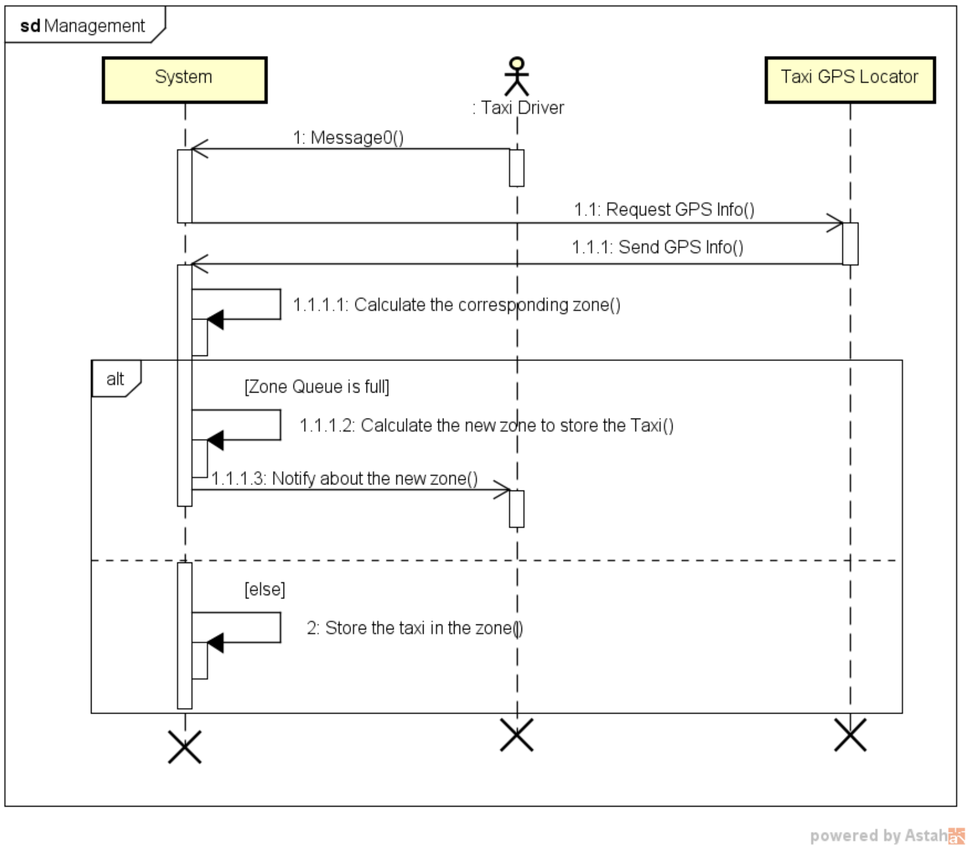


sd Authentication
: RegisteredUser
System
1: Request Authentication screen()
1.1: Show Log in Screen()
1.1.1: Insert Authentication credentials()
1.1.2: Click Log in Button()
alt
[Wrong Credentials]
1.1.2.1: ShowErrorMessage()
[Correct Credentials]
2: ShowConfrimationMessage()
powered by Astah

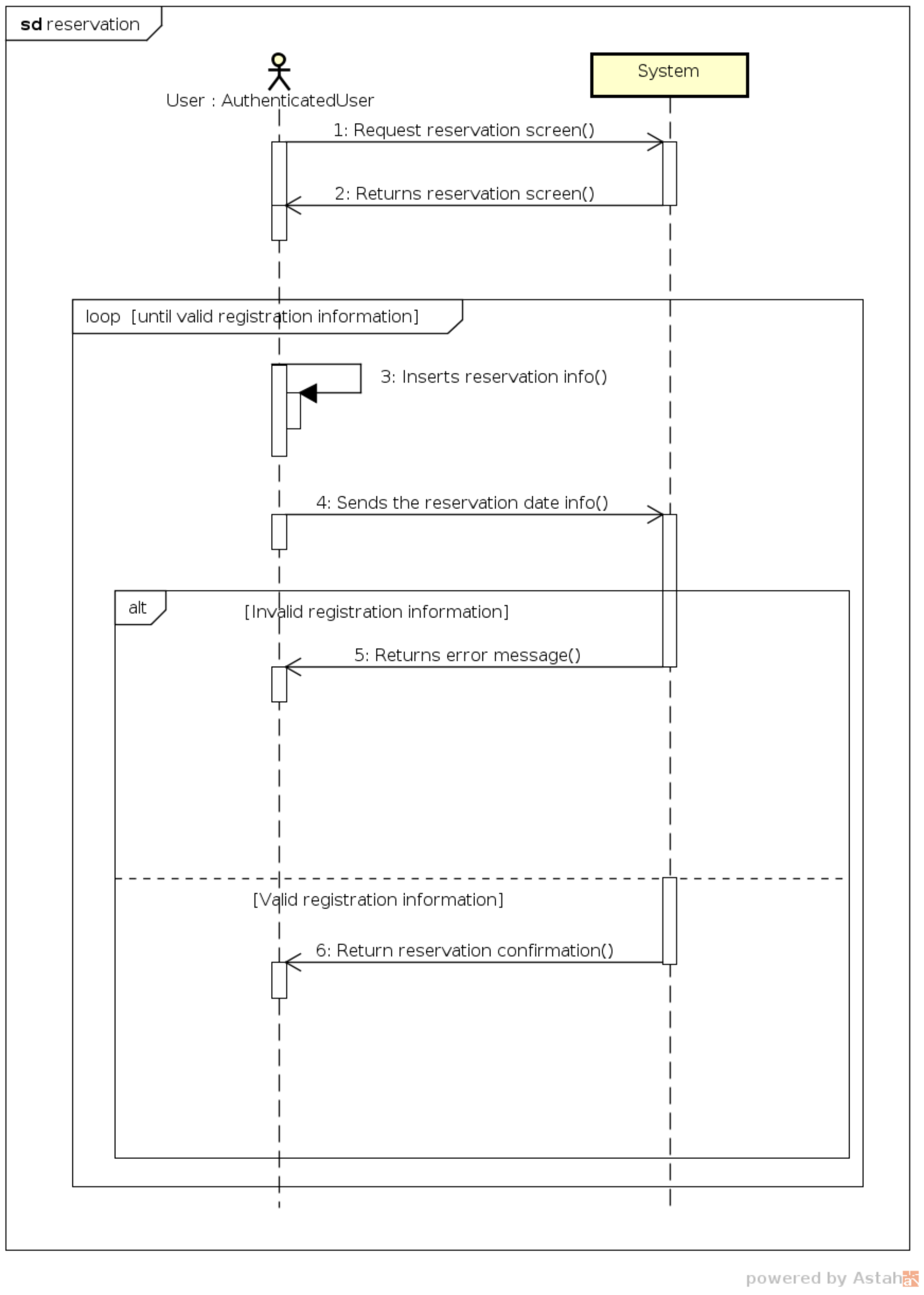
3.5.2.3 Sequence 3


sd Request A Taxi
: Authenticated User
System
: Taxi Driver
1: Fill the Form()
2: Click SendRequest Button()
2.1: Check the User's zone()
loop [ForEach Taxi Found]
alt [TaxiDriver not comfirmed yet]
2.2: Send the Request()
2.2.1: Send the answer()
alt [Taxi Driver confirm his availability]
2.3: Send Confirmation with Taxi Code and Arrival time()
alt [No Taxi Found yet]
loop [ForEach adjacent Zone]
alt [No Taxi Found yet]
3: Check the zone's Queue()
loop [Foreach Taxi Found]
alt [TaxiDriver not comfirmed yet]
4: Send the Request()
4.1: Send the answer()
alt [Taxi Driver confirm his availability]
4.1.1: Send Confirmation with Taxi Code and Arrival time()
alt [No Taxi available]
5: Send error Message()
alt [Taxi doesn't managed to reach the User]
6: Inform the system()
6.1: send a message error()
powered by Astah

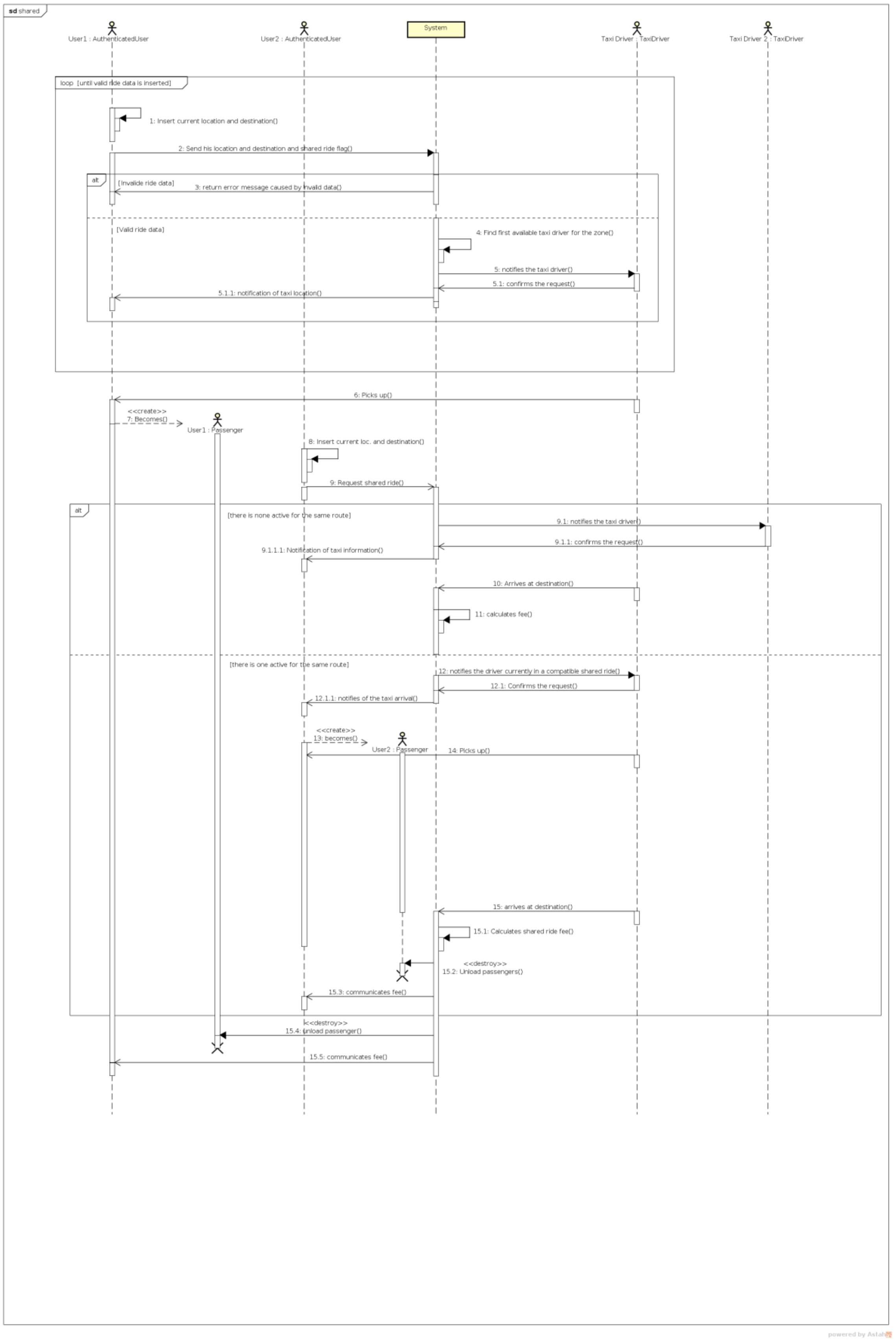
3.5.2.4 Sequence 4


sd Management
System
: Taxi Driver
Taxi GPS Locator
1: Message0()
1.1: Request GPS Info()
1.1.1: Send GPS Info()
1.1.1.1: Calculate the corresponding zone()
alt
[Zone Queue is full]
1.1.1.2: Calculate the new zone to store the Taxi()
1.1.1.3: Notify about the new zone()
[else]
2: Store the taxi in the zone()
powered by Astah

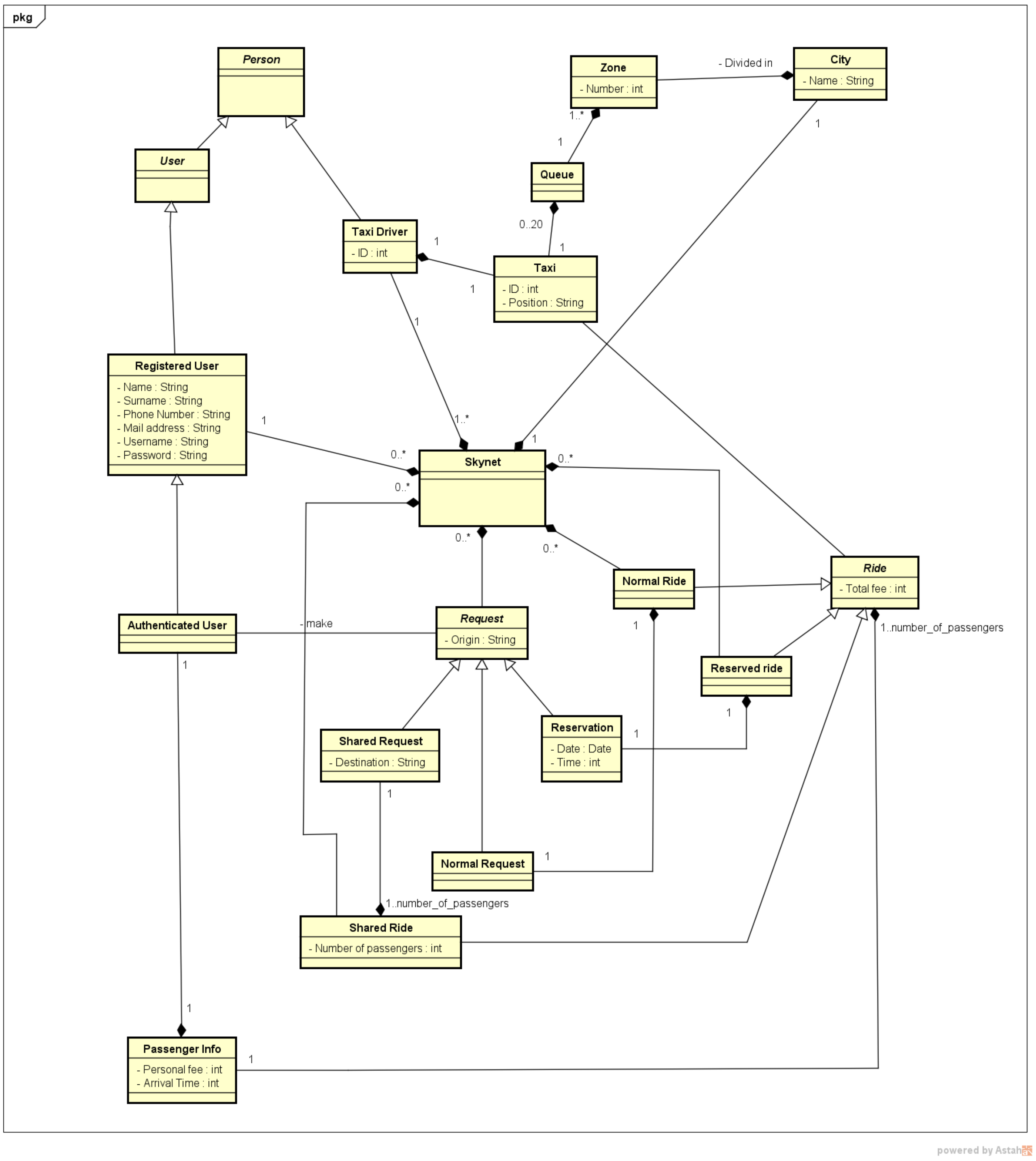
3.5.2.5 Sequence 5


sd reservation
User : AuthenticatedUser
System
1: Request reservation screen()
2: Returns reservation screen()
loop [until valid registration information]
3: Inserts reservation info()
4: Sends the reservation date info()
alt
[Invalid registration information]
5: Returns error message()
[Valid registration information]
6: Return reservation confirmation()
powered by Astah

3.5.2.6 Sequence 6


sd shared
User1 : AuthenticatedUser
User2 : AuthenticatedUser
System
Taxi Driver : TaxiDriver
Taxi Driver 2 : TaxiDriver
loop [until valid ride data is inserted]
1: Insert current location and destination()
2: Send his location and destination and shared ride flag()
alt
[Invalide ride data]
3: return error message caused by invalid data()
[Valid ride data]
4: Find first available taxi driver for the zone()
5: notifies the taxi driver()
5.1: confirms the request()
5.1.1: notification of taxi location()
6: Picks up()
<<create>>
7: Becomes()
User1 : Passenger
8: Insert current loc. and destination()
9: Request shared ride()
alt
[there is none active for the same route]
9.1: notifies the taxi driver()
9.1.1: confirms the request()
9.1.1.1: Notification of taxi information()
10: Arrives at destination()
11: calculates fee()
[there is one active for the same route]
12: notifies the driver currently in a compatible shared ride()
12.1: Confirms the request()
12.1.1: notifies of the taxi arrival()
<<create>>
13: becomes()
User2 : Passenger
14: Picks up()
15: arrives at destination()
15.1: Calculates shared ride fee()
<<destroy>>
15.2: Unload passengers()
15.3: communicates fee()
<<destroy>>
15.4: unload passenger()
15.5: communicates fee()
powered by Astah

3.5.3 Class Diagram



3.6 Nonfunctional Requirements

3.6.1 Performance Requirements

There are requirements for the improvement of the overall performances of the service:

- If a taxi doesn't accept a request in a maximum of 5 minutes, the system forwards the request to another taxi and puts the first one at the bottom of the queue
- The user must get in the taxi by 5 minutes from the arrival of the taxi
- In order to share a ride, the different users must be on the same route in a range of 500 meters

3.6.2 Logical database Requirements

The database must store every user's personal data communicated during the registration. Moreover it must store every info about the rides, the number of passengers, the total fees and the partial fees (in the case of a shared ride).
The information must be maintained for the time requested by the law of every country regarding the storing of data and must be accessible respecting the privacy laws of every countries.

3.6.3 Design Constraints

The design constraints will be inherited from the coding language that will be chosen by the developers, as no code language was imposed in the requirements.

3.6.4 Standard Compliance

Every component of the application must be in compliance with the current standards and must not use any proprietary technology.

3.6.5 Software System Attributes

3.6.5.1 Availability

The system must be available 24/7 and reachable anytime with an internet connection

3.6.5.2 Reliability

The maintenance of the system must be performed in 1 hour lasting shifts, as the system cannot be offline for more than 1 hour every 12 hours. In case of a failure the system must be online again in a maximum of 1 hour.

3.6.5.3 Maintainability

The application does not provide any specific API for the maintenance of the system, but it will be mandatory to well document the code, to inform the future developers of how the application works and how it was developed

3.6.5.4 Portability

The web application runs on any version of Windows from Windows 7 included, the mobile application runs on every version of Android after 4.0.0 included and on any version of iOS after iOS 4 included.
The web application runs on any browser which supports HTML5

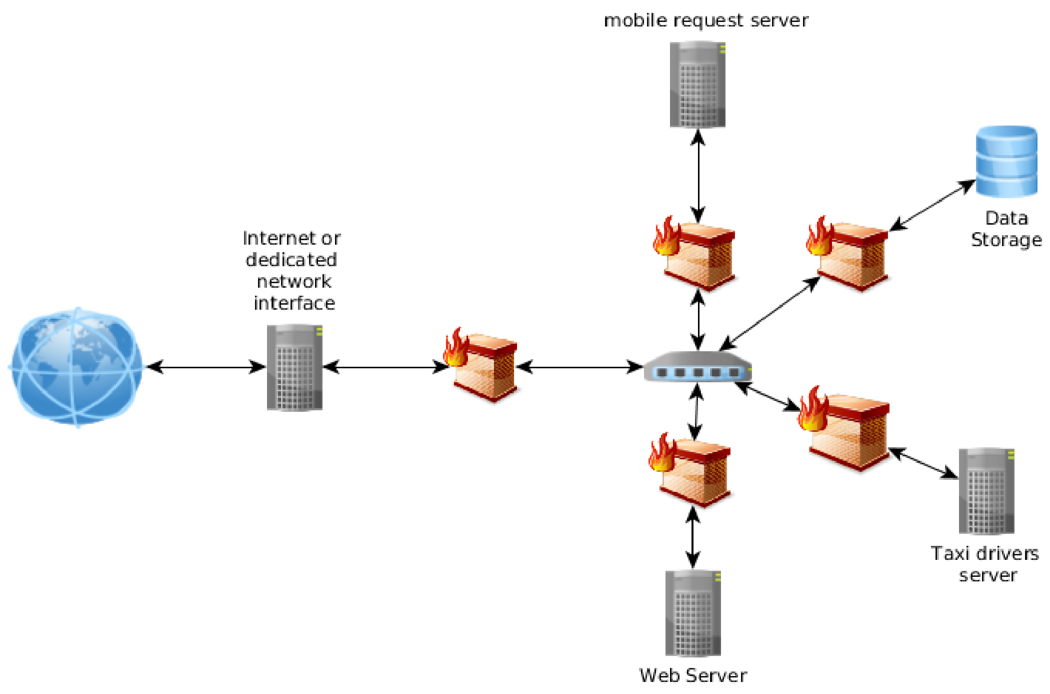
3.6.5.5 Security

<u>For the users</u>:
The users are required to choose a personal username and a password and it is up to them to protect and not divulgate these information. Our system guarantees the protection of the users' personal data according to the laws of the countries where the application is used.
The system requires a password composed at least of 8 characters containing a capital letter, a symbol and a number, in order to improve the security. The personal data are stored with a RSA 2048 bit encryption in order to prevent any theft in case of a breach in the system. Every communication with the user is performed by sending emails on the address that he/she is required to confirm after the registration, so that the security and the privacy of the communications depend only on the secrecy of the user's email.

<u>For the taxi driver</u>:
The taxi drivers, just like the taxies, are identified by a unique and personal identification number. The taxi driver is required to log in the application on the tablet installed in the taxi by using the identification number above at the beginning of every shift. In this way the interaction with a thief or a spiteful person is avoided. The personal number mentioned above is given directly to the taxi driver.

<u>For the system</u>:

The system is protected by an advanced firewall that prevents cyber attacks from the outside. There are also other two firewalls between the web server and the application server and between the application server and the database.
Another important secure method is the implementation of the https connection instead of the normal http to guarantee the confidentiality and the integrity of the communications and also mutual authentication. SSL is resistant to man in the middle attack (MITM) but needs a server certificate signed by a Certification Authority(CA).
The database is also implemented in a way that avoids the known SQL attacks.

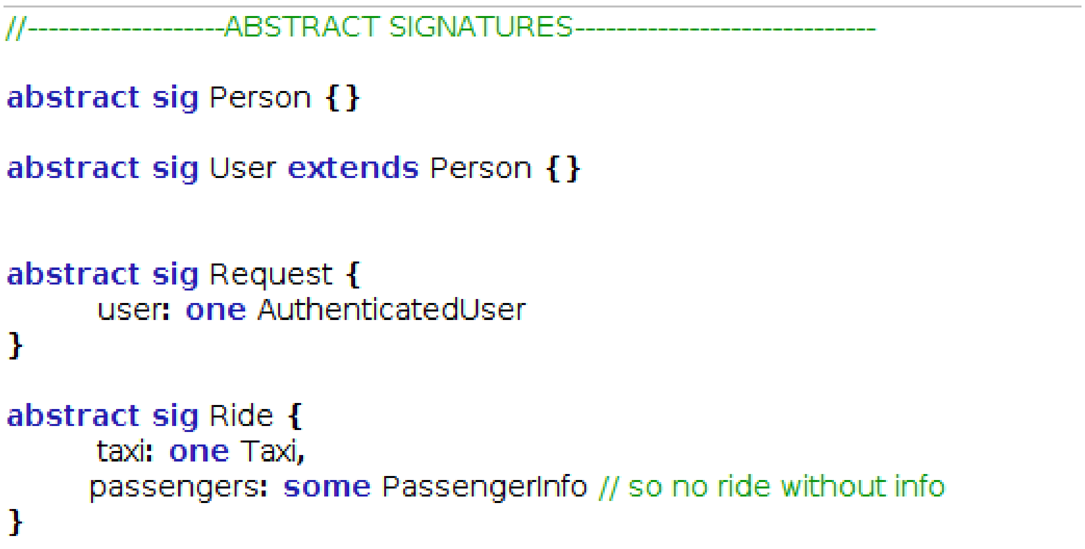


4 Appendix

4.1 Alloy

The complete alloy file (.als) can be found in the project Github repository.
The following alloy model was created using the class diagram.
This part shows all the code divided in the related parts: signatures, facts, assertions and predicates.

4.1.1 Signatures



```
//------------------ABSTRACT SIGNATURES----------------------------

abstract sig Person {}

abstract sig User extends Person {}


abstract sig Request {
	user: one AuthenticatedUser
}

abstract sig Ride {
	taxi: one Taxi,
	passengers: some PassengerInfo // so no ride without info
}
```

```
//-----------------DERIVED SIGNATURES-------------------------------
sig RegisteredUser extends User{}

sig AuthenticatedUser extends RegisteredUser {}

sig TaxiDriver extends Person {
	taxi: one Taxi
}

sig Taxi {}

sig Queue {
	taxies: set Taxi
}

sig Zone {
	queue: one Queue
}

sig City {
	zones: set Zone
}

sig NormalRequest extends Request {}

sig Reservation extends Request {}

sig SharedRequest extends Request {}

sig PassengerInfo {
	user: one AuthenticatedUser
}

sig NormalRide extends Ride {
	request: one NormalRequest
}

sig ReservedRide extends Ride {
	request:one Reservation
}

sig SharedRide extends Ride {
	requests: some SharedRequest
}

sig Skynet {
	registeredUsers:set RegisteredUser,
	cities: one City,
	taxidrivers: set TaxiDriver,
	requests: set Request,
	normalRides: set NormalRide,
	sharedRides:set SharedRide,
	reservedRides:set ReservedRide
}
```

4.1.2 Facts

```
//-----------------------FACTS ----------------------------------------------

fact noTaxiDriversSameTaxi{
	no t1:TaxiDriver,t2:TaxiDriver| t1.taxi=t2.taxi && !(t1=t2)
}

fact noTaxiWithoutDriver{
	all tax:Taxi| one driver:TaxiDriver | driver.taxi=tax
}

fact noForeverAloneQueue {
	all q:Queue | one zon:Zone | zon.queue=q
}

fact noMoreThanTwentyTaxiPerZoneQueue {
	all q:Queue | #q.taxies =<20
}

fact noUbiquitaxi{
	all que1:Queue, que2:Queue | !(que1=que2) implies ((que1.taxies & que2.taxies) = none)
}

fact noForeverAloneZone {
	all z:Zone | one c:City | z in c.zones
}

fact noMultipleNormalRequestsForUser {
	all r1:NormalRequest,r2:NormalRequest | !(r1=r2) implies !(r1.user=r2.user)
}

fact noMultipleSharedRequestsForUser {
	all r1:SharedRequest,r2:SharedRequest | !(r1=r2) implies !(r1.user=r2.user)
}

fact noNormalAndSharedRequestsForUserAtSameTime {
	all r1:SharedRequest,r2:NormalRequest | !(r1.user=r2.user)
}

fact noRideWithoutTaxi {
	all r:Ride | one t:Taxi | r.taxi=t
}

fact noTwoRidesWithSameTaxi {
	no r1:Ride, r2:Ride | r1.taxi =r2.taxi && !(r1=r2)
}

fact noPassengerInfoWithoutAssociatedRide {
	all p:PassengerInfo |one r:Ride | p in r.passengers
}
fact noMultiplePassengerInfoForSameAuthenticatedUser{
	all p1:PassengerInfo, p2:PassengerInfo | !(p1=p2) implies !(p1.user=p2.user)
}

fact noMultipleNormalRideWithSameRequest{
	all n1:NormalRide,n2:NormalRide | !(n1=n2) implies !(n1.request=n2.request)
}

fact noMultipleReservedRideWithSameRequest{
	all rr1:ReservedRide,rr2:ReservedRide | !(rr1=rr2) implies !(rr1.request=rr2.request)
}

fact noMultipleSharedRideWithSameSharedRequest{
	all sr1:SharedRide, sr2:SharedRide | !(sr1=sr2) implies !(sr1.request=sr2.request)
}

fact noNormalRideWithMultiplePassengersInfo{
	all n:NormalRide | #n.passengers=1
}

fact noNormalRideWithAPassengerInfoThatIsNotAPassenger{
	all n:NormalRide | all p:PassengerInfo | p in n.passengers implies p.user=n.request.user
}

fact noSharedRideWithAPassengerInfoThatIsNotAPassenger{
	all sr:SharedRide | all p:PassengerInfo |(p in sr.passengers)implies  p.user in sr.requests.user
}

fact noReservedRideWithAPassengerInfoThatIsNotAPassenger{
	all rr:ReservedRide | all p:PassengerInfo | p in rr.passengers implies p.user=rr.request.user
}
```

4.1.3 Predicates

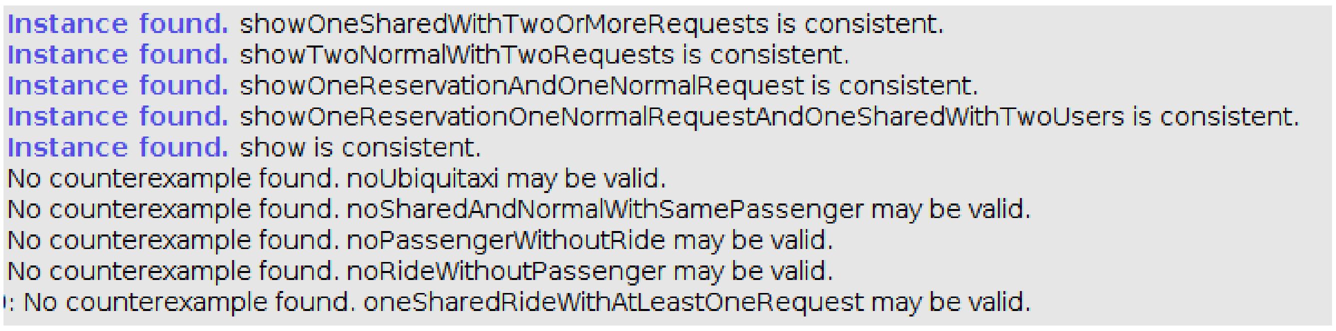
```
//------------------------------------PREDICATES----------------------------------------------
//---------------ONE SHARED WITH TWO REQUESTS FOR THE RIDE--------------
pred showOneSharedWithTwoOrMoreRequests [sr:SharedRequest]{
        one s:SharedRide| sr in s.requests && #SharedRide=1 && #Skynet=1 && #s.passengers>1 && #s.passengers<5
        #NormalRequest=0
        #ReservedRide=0
}
run showOneSharedWithTwoOrMoreRequests for 10

//---------------TWO NORMAL WITH TWO SEPARATE REQUESTS FOR THE RIDE--------------
pred showTwoNormalWithTwoRequests{
        #Skynet=1
        #NormalRide=2
        #SharedRequest=0
        #Reservation=0
        #Zone=3
        #Taxi=3
}
run showTwoNormalWithTwoRequests for 10

//---------------ONE RESERVED RIDE AND ONE NORMAL RIDE--------------
pred showOneReservationAndOneNormalRequest{
        #Skynet=1
        #NormalRide=1
        #SharedRequest=0
        #ReservedRide=1
        #Zone=3
        #Taxi=3
}
run showOneReservationAndOneNormalRequest for 10

//---------------ONE RESERVATION AND ONE SHARED RIDE WITH TWO USERS (AND ONE OF THEM HAS THE RESERVATION--------------
pred showOneReservationOneNormalRequestAndOneSharedWithTwoUsers{
        one s:SharedRide | some r:Reservation |  #s.passengers>1 && #s.passengers<5 implies (r.user in s.passengers.user)
        all s:SharedRequest | one sr:SharedRide | s in sr.requests
        #Skynet=1
        #NormalRide=0
        #NormalRequest=0
        #SharedRequest=2
        #Reservation=1
        #ReservedRide=0
        #Zone=3
        #Taxi=3
}
run showOneReservationOneNormalRequestAndOneSharedWithTwoUsers for 10
```

```
//--------------------GENERAL SYSTEM-----------------------------------------------------

pred show (){
        #Skynet=1
        #Zone<12
        #Taxi>5
        #City=1
        #AuthenticatedUser>2
        #Ride>2
}
run show for 11
```

4.1.4 Assertions

```
//-----------------------------------ASSERTS---------------------------------------------

assert noUbiquitaxi{
	all que1:Queue, que2:Queue | !(que1=que2) implies ((que1.taxies & que2.taxies) = none)
}
check noUbiquitaxi


assert noSharedAndNormalWithSamePassenger{
	all s:SharedRide, n:NormalRide | !(n.request.user in s.passengers.user)
}

check noSharedAndNormalWithSamePassenger

assert noPassengerWithoutRide{
	all p:PassengerInfo,r:Ride | p in r.passengers
}

check noPassengerWithoutRide


assert noRideWithoutPassenger{
	all r:Ride | #r.passengers>0
}

check noRideWithoutPassenger


assert oneSharedRideWithAtLeastOneRequest{
	all sr:SharedRide| #sr.requests>0
}
check oneSharedRideWithAtLeastOneRequest
```

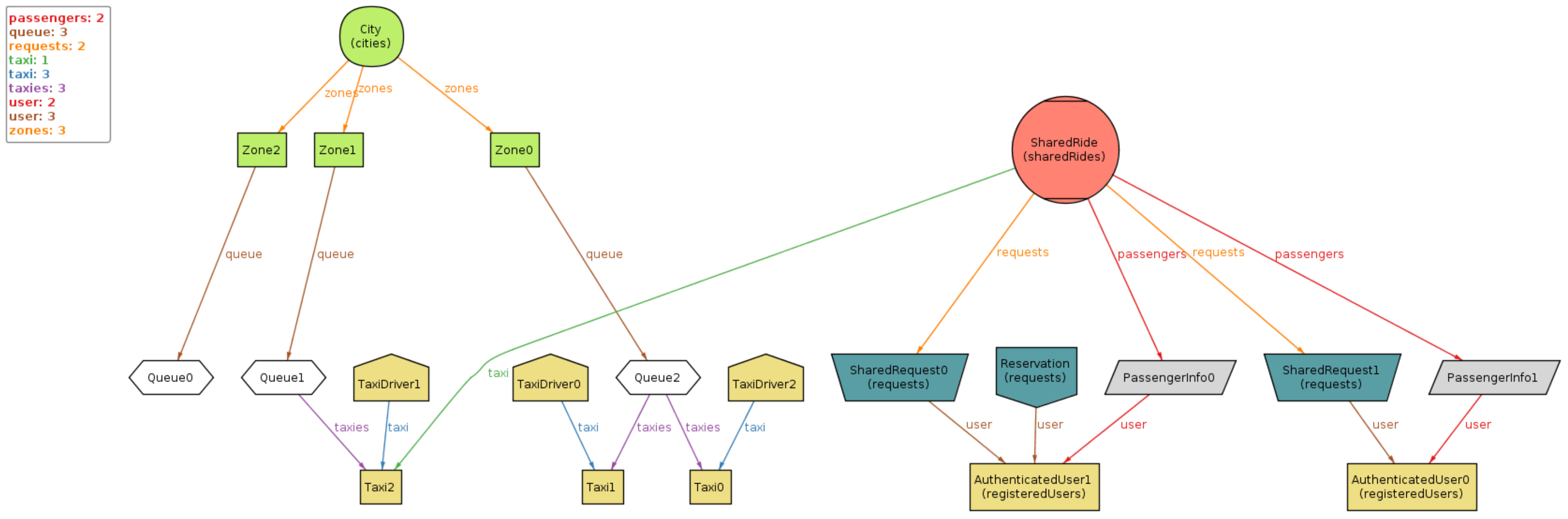
4.1.5 Results

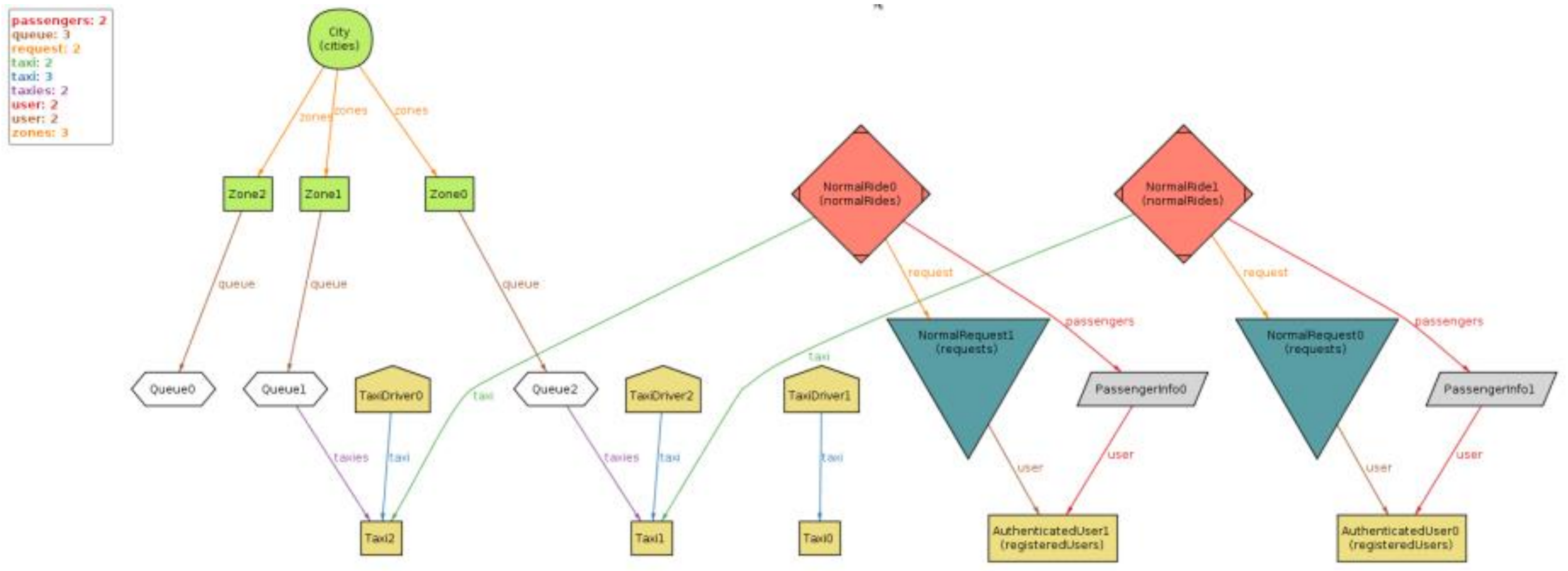


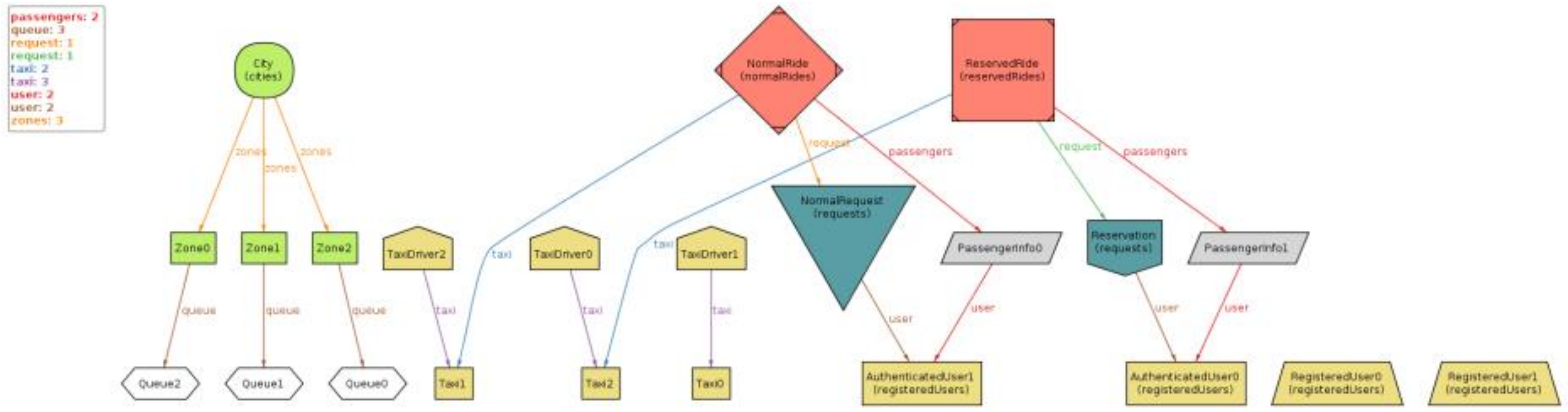
```
Instance found. showOneSharedWithTwoOrMoreRequests is consistent.
Instance found. showTwoNormalWithTwoRequests is consistent.
Instance found. showOneReservationAndOneNormalRequest is consistent.
Instance found. showOneReservationOneNormalRequestAndOneSharedWithTwoUsers is consistent.
Instance found. show is consistent.
No counterexample found. noUbiquitaxi may be valid.
No counterexample found. noSharedAndNormalWithSamePassenger may be valid.
No counterexample found. noPassengerWithoutRide may be valid.
No counterexample found. noRideWithoutPassenger may be valid.
: No counterexample found. oneSharedRideWithAtLeastOneRequest may be valid.
```

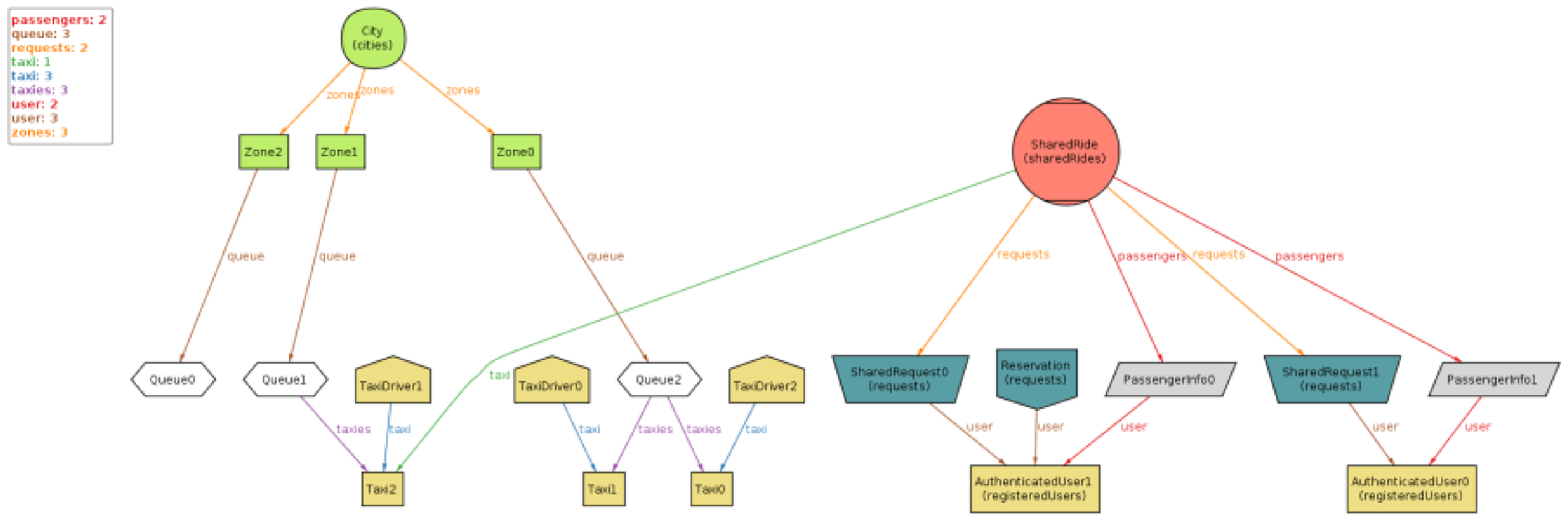
4.2 Worlds Generated

In this parts there are some worlds generated from the predicates above in Alloy.
The worlds are displayed in the same order as the predicates above.

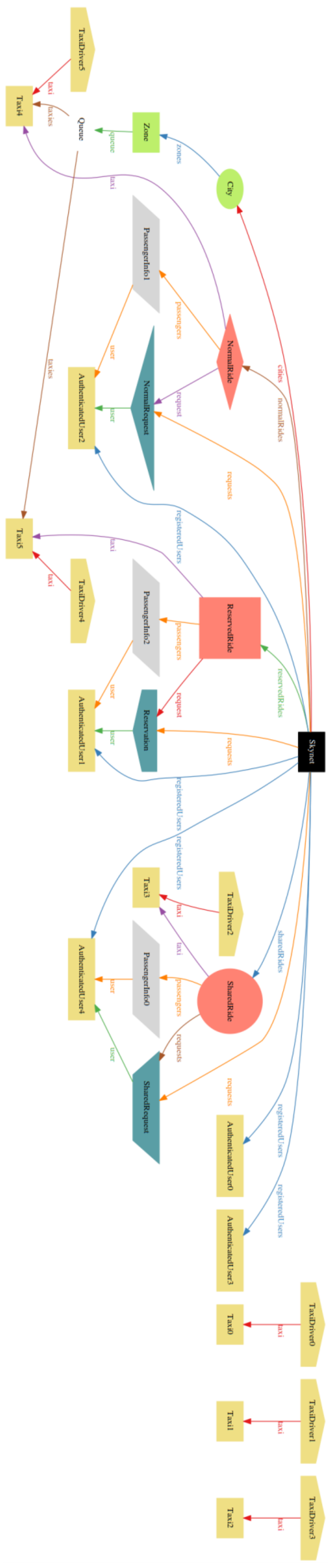

passengers: 2
queue: 3
requests: 2
taxi: 1
taxi: 3
taxies: 3
user: 2
user: 3
zones: 3
City (cities)
zones
zones
zones
Zone2
Zone1
Zone0
queue
queue
queue
Queue0
Queue1
Queue2
TaxiDriver1
TaxiDriver0
TaxiDriver2
taxies
taxi
taxi
taxies
taxies
taxi
Taxi2
Taxi1
Taxi0
SharedRide (sharedRides)
taxi
requests
passengers
requests
passengers
SharedRequest0 (requests)
Reservation (requests)
PassengerInfo0
SharedRequest1 (requests)
PassengerInfo1
user
user
user
user
user
AuthenticatedUser1 (registeredUsers)
AuthenticatedUser0 (registeredUsers)


passengers: 2
queue: 3
request: 2
taxi: 2
taxi: 3
taxies: 2
user: 2
user: 2
zones: 3
City (cities)
zones
zones
zones
Zone2
Zone1
Zone0
queue
queue
queue
Queue0
Queue1
Queue2
TaxiDriver0
TaxiDriver2
TaxiDriver1
taxies
taxi
taxies
taxi
taxi
taxi
taxi
Taxi2
Taxi1
Taxi0
NormalRide0 (normalRides)
NormalRide1 (normalRides)
request
passengers
request
passengers
NormalRequest1 (requests)
NormalRequest0 (requests)
PassengerInfo0
PassengerInfo1
user
user
user
user
AuthenticatedUser1 (registeredUsers)
AuthenticatedUser0 (registeredUsers)


passengers: 2
queue: 3
request: 1
request: 1
taxi: 2
taxi: 3
user: 2
user: 2
zones: 3
City
(cities)
zones
zones
zones
Zone0
Zone1
Zone2
queue
queue
queue
Queue2
Queue1
Queue0
TaxiDriver2
TaxiDriver0
TaxiDriver1
taxi
taxi
taxi
taxi
taxi
Taxi1
Taxi2
Taxi0
NormalRide
(normalRides)
ReservedRide
(reservedRides)
request
passengers
request
passengers
NormalRequest
(requests)
PassengerInfo0
Reservation
(requests)
PassengerInfo1
user
user
user
user
AuthenticatedUser1
(registeredUsers)
AuthenticatedUser0
(registeredUsers)
RegisteredUser0
(registeredUsers)
RegisteredUser1
(registeredUsers)



passengers: 2
queue: 3
requests: 2
taxi: 1
taxi: 3
taxies: 3
user: 2
user: 3
zones: 3
City
(cities)
zones
zones
zones
Zone2
Zone1
Zone0
queue
queue
queue
Queue0
Queue1
Queue2
TaxiDriver1
TaxiDriver0
TaxiDriver2
taxies
taxi
taxi
taxies
taxies
taxi
Taxi2
Taxi1
Taxi0
SharedRide
(sharedRides)
taxi
requests
passengers
requests
passengers
SharedRequest0
(requests)
Reservation
(requests)
PassengerInfo0
SharedRequest1
(requests)
PassengerInfo1
user
user
user
user
user
AuthenticatedUser1
(registeredUsers)
AuthenticatedUser0
(registeredUsers)



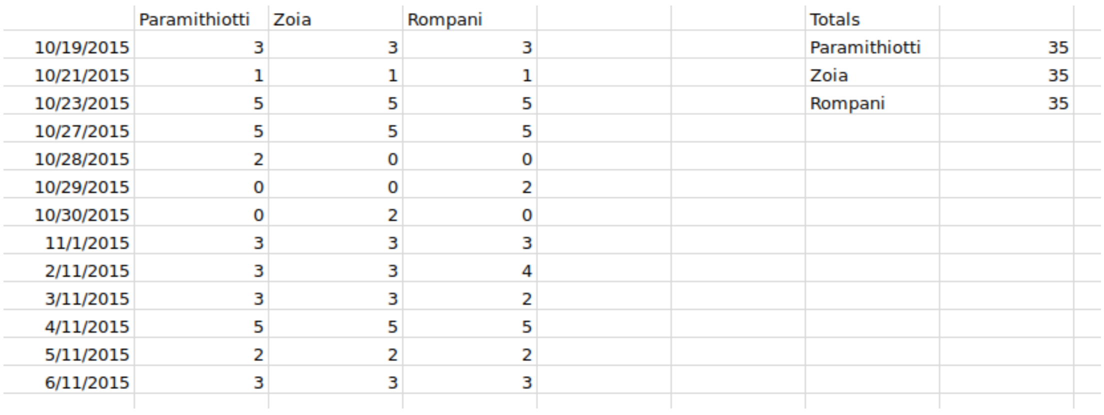
Skynet
City
cities
zones
Zone
queue
Queue
taxies
Taxi4
taxi
TaxiDriver5
NormalRide
normalRides
passengers
PassengerInfo1
user
request
NormalRequest
user
AuthenticatedUser2
requests
registeredUsers
Taxi5
taxi
taxi
TaxiDriver4
ReservedRide
reservedRides
passengers
PassengerInfo2
user
request
Reservation
user
AuthenticatedUser1
requests
registeredUsers
registeredUsers
Taxi3
taxi
TaxiDriver2
taxi
sharedRides
SharedRide
passengers
PassengerInfo0
user
AuthenticatedUser4
requests
user
SharedRequest
requests
registeredUsers
AuthenticatedUser0
registeredUsers
AuthenticatedUser3
Taxi0
taxi
TaxiDriver0
Taxi1
taxi
TaxiDriver1
Taxi2
taxi
TaxiDriver3

4.3 Softwares and Tools used

- **Microsoft Word 2010/2013** (https://products.office.com/it-it/word): text editor used to redact and to format the RASD
- **Astah Community** (http://astah.net/editions/community): software used for the creation of the class diagrams and the sequence diagrams
- **Pencil** (http://pencil.evolus.vn/): open-source prototyping tool used for the creation of the mockups
- **Alloy Analyzer 4.2** (http://alloy.mit.edu/alloy/): software used to prove the consistency of the model
- **Google Drive** (https://www.google.com/intl/it_it/drive/): software used for saving on the cloud the documents we were redacting
- **Github** (https://github.com/): revision control tool used for avoiding revision problems while working on the same files
- **Adobe Photoshop CC/CS6** (http://www.adobe.com/it/products/photoshop.html): image editing software

Even if the version control tool offered by Github works perfectly, we have preferred to work on Google Drive in order to save our files faster and automatically, as we have worked together most of the time.
Github was used only for saving big parts of the project and for uploading the final version.

4.4 Hours of work



	Paramithiotti	Zoia	Rompani			Totals	
10/19/2015	3	3	3			Paramithiotti	35
10/21/2015	1	1	1			Zoia	35
10/23/2015	5	5	5			Rompani	35
10/27/2015	5	5	5				
10/28/2015	2	0	0				
10/29/2015	0	0	2				
10/30/2015	0	2	0				
11/1/2015	3	3	3				
2/11/2015	3	3	4				
3/11/2015	3	3	2				
4/11/2015	5	5	5				
5/11/2015	2	2	2				
6/11/2015	3	3	3				

5 Revision

All the new functionalities and elements that were added in the revision are listed here.

5.1 Introduction

5.1.1 Actors

The definition of “Operator”

5.1.2 Goals

A specific Goal for the management of an Emergency

5.2 Overall Description

5.2.1 Assumptions and Dependencies

New assumptions about the payment

5.3 Specific Requirements

5.3.1 External Interfaces Requirements

5.3.1.1 Hardware Interfaces

The constraint “An app that can perform phone calls must be installed in that table” in order to perform phone calls from the tablet leaning on an existing app.

5.3.2 Functional Requirements

5.3.2.1 Goal 2

The point “The taxi must go to the requesting user as soon as he accepts the request” of the Requirements has been moved into the Domain Assumptions.

5.3.2.2 Goal 3

The point "A taxi driver can remove his availability and restore it whenever he wants, but that implies the loss of his priority in the queue"

5.3.2.3 Goal 4

The point "The taxi must be at the user's position at the requested time" of the Requirements has been moved into the Domain Assumptions.
The point "The user must be able to cancel a reservation" was added to the Requirements

5.3.2.4 Goal 6

The definition of all the requirements of the new goal

5.3.3 Scenarios

5.3.3.1 Scenario 7

A new Scenario in order to describe the emergency Goal

5.3.4 Use Cases

5.3.4.1 Use Case 7

A new Use Case in order to describe the emergency Goal

6 Revision Table

Version	Date	Modifications
1.0	2015/11/06	First version
2.0	2015/12/04	See (5 Revision)