POLITECNICO DI MILANO

Master Degree course in Computer Science Engineering

Requirement Analysis and Specification Document (RASD)

Instructor: Prof. Elisabetta Di Nitto

Authors:

Brunato Andrea	851413
Costantini Lorenzo	852599
Dell’Orto Alessandro	853050

Academic year 2015-2016

Contents

1: Introduction

1.1 Purpose

The following RASD exposes the functionalities of EasyTaxi, the project that we are going to develop. The main goal of this document is to completely describe the system in terms of functional requirements, with the help of various UML diagrams and scenarios, and to point out constraints on the future implementation of the software.

This document is intended to all developer and programmer who have to implement the requirements, and could be used as a contractual basis between the customer and the developer.

1.2 Scope

Our software EasyTaxi aims at optimizing the taxi service in a large city, automatizing many procedures of the actual system.

The application will allow users to call a taxi from a mobile application or from the web, providing even options for shared rides between different users.

Non registered users will have only the possibility to make simple requests (also shared).

We will provide many more services for registered users, like the use of GPS for the automatic detection of the passenger, or the possibility to make reservations up until due days, with the possibility of sharing the cost of the ride with other people that has booked similar rides.

The system will successfully keep track of all requests and reservations and will grant a fair division of them between all available taxi drivers, as we will see in detail later in this document.

This application will use GPS tracking system in order to control the taxi locations and to show taxi drivers the routes they have to follow.

1.3 Actual System

Our work consists in the optimization and modernization of the public taxi service of a large city.

We assume that the actual system doesn't incorporate at all the use of mobile or web applications and that all the operations concerning the booking of a taxi by users are made only by calling employees working for the government of the city.

Those people handle all the requests that arrive from the users and they are responsible for searching free taxi drivers in a common database frequently updated by the employees themselves.

The actual database contains informations about the availability of taxi drivers and their area of competence. Taxi drivers are called by the government office in order to know if they are able to take care of a request or not when they have declared to be available.

Moreover taxi drivers can call the employees of the office for asking to be chosen as soon as possible when a request for a ride arrives.

We assume that the current system works really similar to the one we are going to implement, despite that the management of the service of the various city zones is done manually by human beings and not by an automatic system.

Instead, the taxi sharing service will be a totally new functionality.

1.4 Actors

Visitor: This actor represents the unregistered user. From the home page the visitor can login to the system, register himself to the service or start the procedure for a simple request (possibly shared).

Client/Registered User: This actor represents the client to our system who has successfully terminated the registration and so the system is storing some of his personal data. When he is successfully logged in, he can request or reserve a taxi and benefit of all the services provided by the application.

Taxi Driver: This actor represents the taxi driver correctly registered to the system. After he is successfully logged in, he can accept and manage reservation and requests.

1.5 Goals

- [G1] Allow a visitor to register himself to the system.
- [G2] Allow user to request a taxi through a web application.
- [G3] Allow user to request a taxi through a mobile application.
- [G4] Allow the user to make a taxi reservation.
- [G5] Allow the user to make shared rides.
- [G6] Allow the taxi driver to reply to user requests through a mobile application.
- [G7] Allow the taxi driver to login the system.
- [G8] Allow a visitor to login the system.
- [G9] Allow a client to delete a reservation already done.

1.6 Glossary

1.6.1 Definitions

- **Request/Simple Request/Immediate Request**: Action performed by a client or a visitor, who asks for a taxi which immediately arrives where the passenger is located.
- **Reservation/Booking**: Action performed by a client, who asks for a taxi specifying the position, the date and the hour of its arrival.
- **Shared Request**: Action performed by a client or a visitor who asks for a simple request, but setting also the shared ride option on.
- **Shared Reservation**: Action performed by a client who asks for a reservation, but setting also the shared ride option on.
- **Generic Shared Request**: This term indicates both the shared request and the shared reservation.

- **Generic Request**: This term indicates the act of a user who asks for a taxi, without distinguish between simple requests, reservations or shared generic shared requests.
- **Registered User/Users**: this term indicates the client and the taxi driver, i.e. everyone who is successfully registered to the system.
- **Available Taxi**: A taxi which hasn't accepted a generic request yet, but it's ready to receive messages from the system regarding new available rides.
- **Request Queue**: System component dealing with requests in order to manage in a fair way which available taxi contact for a new request.
- **Request Attributes**: This term indicates the number of clients involved in the ride, the number of passengers, the fee for the passengers and the route to follow.
- **Access Code**: Unique code necessary for the login of a taxi driver, released to him by the government of the city at the moment of his hiring.
- **Taxi Code**: Unique ID for a taxi car.

1.6.2 Acronyms

- **RASD**: Requirement Analysis and Specification Document
- **GPS**: Global Positioning System
- **DBMS**: Database Management System
- **OS**: Operating System

1.6.3 Other conventions

#: In the 3.1.1 section dedicated to mockups, # indicates that the word after the symbol is a variable that changes value for every different user who visualize the page.

1.7 Stakeholders

The principal stakeholder is the government of the city, who commissioned the project. For the moment there isn't a plan of having some services under payment by registered users, but in this case the earned money will be gained by the government.
Other stakeholders are the citizens and the taxi drivers who work in the city.

1.8 Reference Documents

- Specification Document: Software Engineering 2 Project, AA 2015-2016 Assignments 1 and 2
- Reference for Activity Diagram Syntax: http://www.uml-diagrams.org/activity-diagrams-reference.html

1.9 Document overview

This document is essentially structured in four part:

- Section 1: **Introduction**, that gives an high level description our software project, talking about its purpose and goals.
- Section 2: **Overall Description**, that gives more precise informations about the software product with a focus about constraints and assumptions. This sections also talks about the characteristics of the future users.
- Section 3: **Specific Requirements**, that contains functional and non functional requirements, typical scenarios and use cases, UML Activity Diagrams, Sequence Diagrams, State Chart Diagrams and Class Diagrams.
- Section 4: **Appendix**. This part contains everything related to Alloy and other final informations on our project.

2: Overall Description

2.1 Product perspective

The application will be released as a web application accessible by a browser and there will be a mobile application downloadable from the stores of the compatible operating systems.
The application is not independent because it provides interaction with a GPS system for location purposes, and uses SMS and e-mail external systems for notifications to the actors.
That application will not have any internal interface for administration but it will be only user based.
However there exists administrators working for the government of the city who can access the application database in order to manually add new taxi driver accounts.

2.2 User characteristics

The software is intended principally for the citizens and taxi drivers of the city. The formers by subscribing to the service will benefit of all the functionalities, and the latters will have their work simplified but they will have to change and adapt their usual habits with the introduction of the new computer system.
The choice of giving the possibility to visitors to do a simple request (or a shared request) is intended for the foreigners who will use the application seldom or even once in a lifetime. If some of them have emergencies or other needs, the application doesn't request compulsorily the registration for them, which would be really annoying.

2.3 Constraints

2.3.1 Regulatory policies

- The application will ask the user consent for its cookies management during the registration session (as indicated at the row ”I Agree With Terms And Conditions” in the mockup section for the registration).
- User sensible data won't be distributed to third parts to preserve user's privacy.
- The mobile version won't access to phone functionalities and personal data. In order to use GPS the application will explicitly ask the user's permission.

2.3.2 Hardware limitations

- User needs to have an internet connection to benefit of every service provided by our software (login, registration, generic requests).
- The client will need at least 20 MB of free space on the memory for the download of the mobile application.

2.3.3. Software limitations

- For the correct usage of the application it is recommended to have installed Java 7.
- The application could be used on any OS that supports browsers and DBMS.

2.4 Assumptions and Dependencies

2.4.1 Domain Assumptions

- A taxi can be available if and only if a driver is into that car, logged in the system and he isn't serving another user in the meantime.
- The mobile device of the taxi driver is an onboard computer. From that device he can login see the route he has to follow, accept requests and all his other functionalities described.
- City zones are a partition of the city, so they are disjointed.
- The payments are managed directed by the taxi driver, the system we implement won't deal with the taxi ride payments, apart the calculation and notifications of the fees.
- A ride can't terminate farther than 10 km from the nearest point of the city's frontier.
- A taxi which is serving an user can move without limitations from a zone of the city to another one. It can also conclude a ride outside the frontier of the city, but it's essential that the ride starts into one of the city zones and not outside it.
- We assume that the waits for the user for the confirmation of a request or reservation are not likely to be longer than 1 minute, which corresponds to four demands to taxis that says to be available and ready to work. So we don't manage the case in which the client has to wait a very long time for the confirmation of his request and the assignation of a taxi.
- We assume that the reservation can't fail; it means that during a period of ten minutes the system finds a taxi in the queue which accepts the ride.
- There is no online registration for taxi drivers. When a taxi driver wants to subscribe to the system, he/she has to go to the government office and there the responsible workers will give him the access code necessary to log in to the system.
- Every taxi can carry at most four passengers, and this is particularly important for the calculation of the feasibility of a generic shared request.

2.5 Future possible implementations

The possibility of searching taxis in farther city zones could be implemented for the reservation option when there aren't available taxis. Only for this case it could be useful to extend the search first to the adjoining zones of the zone where the user is waiting, and then also farther, until the estimated waiting time doesn't exceed a certain fixed value.

3: Specific Requirements

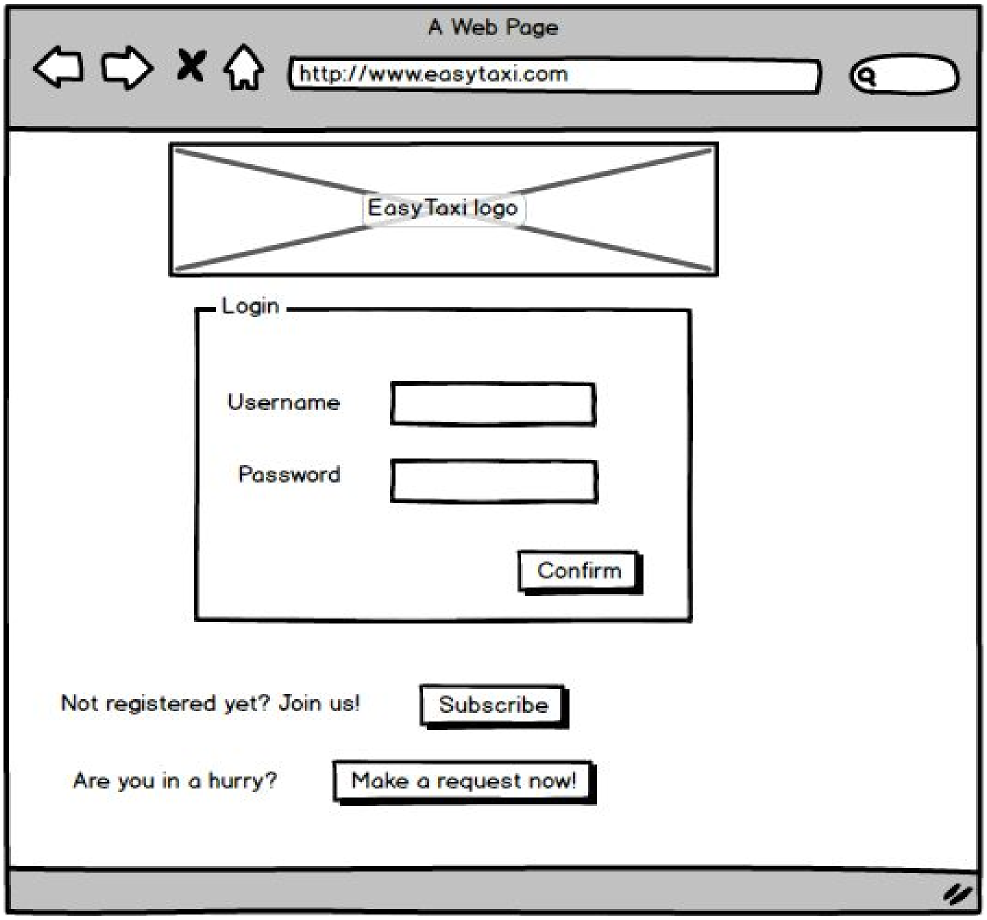
3.1 External Interface Requirements

3.1.1 User Interfaces

In this section are presented some mockup that shows how the web and mobile pages will be seen by the all the users of the software, both on the mobile systems and on the web. This is how the front-end of the final system will look like for every implemented functionality.

3.1.1.1 Visitor Homepage

This mockup regards the website homepage: from here a visitor can register himself, login to the system and has also the possibility to make an immediate taxi request.

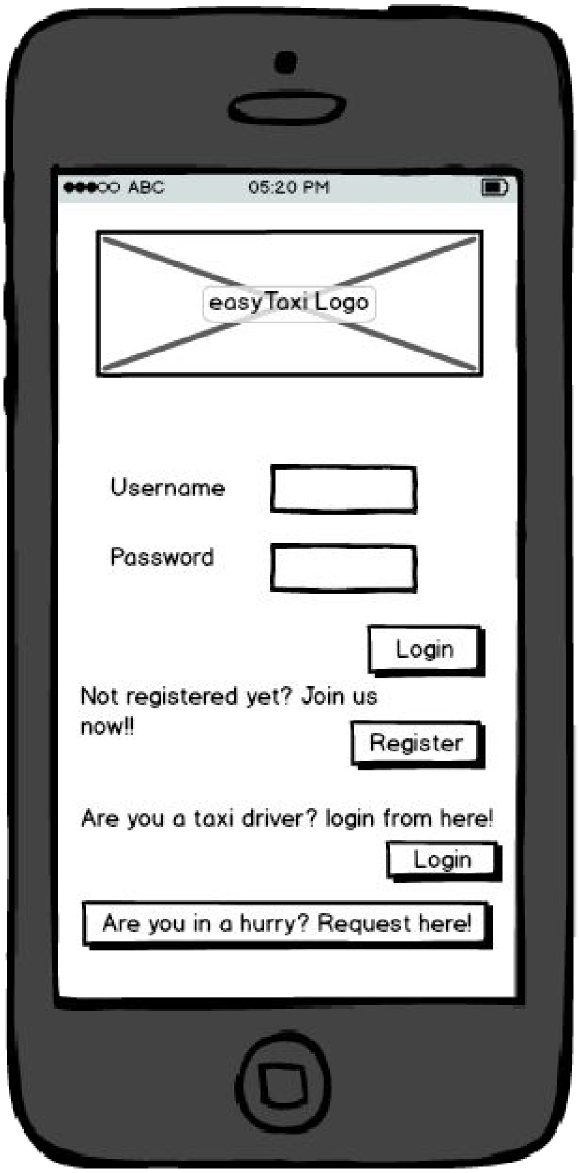


Not registered yet? Join us!

Are you in a hurry?

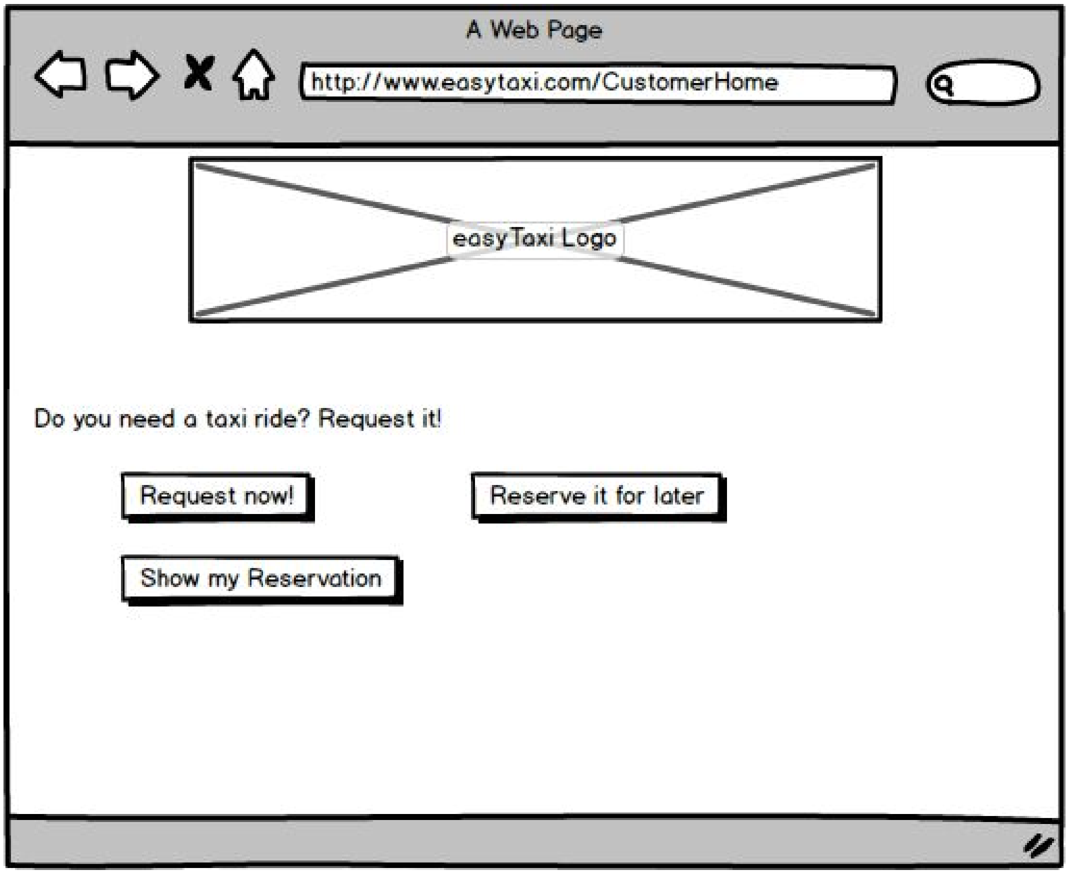
3.1.1.2 Visitor Mobile Homepage

This mockup shows the main screen of the visitor in the mobile application. The mobile application provides also the link to the login page for taxi drivers.


ABC
05:20 PM
easyTaxi Logo
Username
Password
Login
Not registered yet? Join us now!!
Register
Are you a taxi driver? login from here!
Login
Are you in a hurry? Request here!

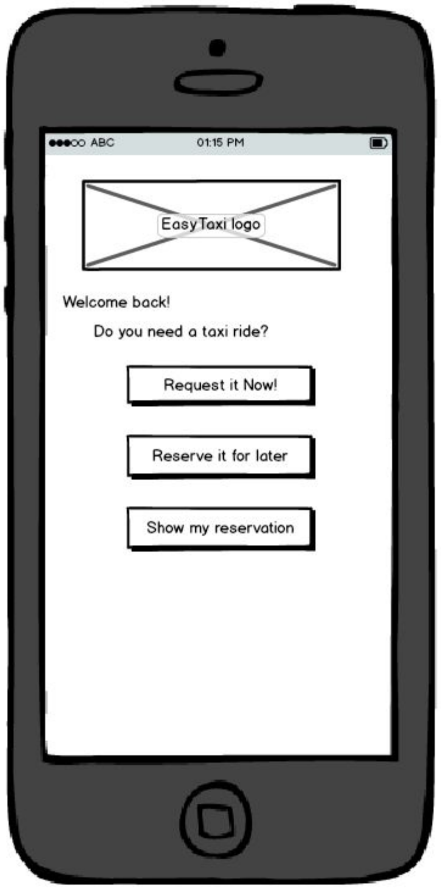
3.1.1.3 Client Web Homepage

This mockup represents the registered client homepage when logged in.



3.1.1.4 Client Mobile Homepage

This mockup shows the screen the client will see after having logged in.

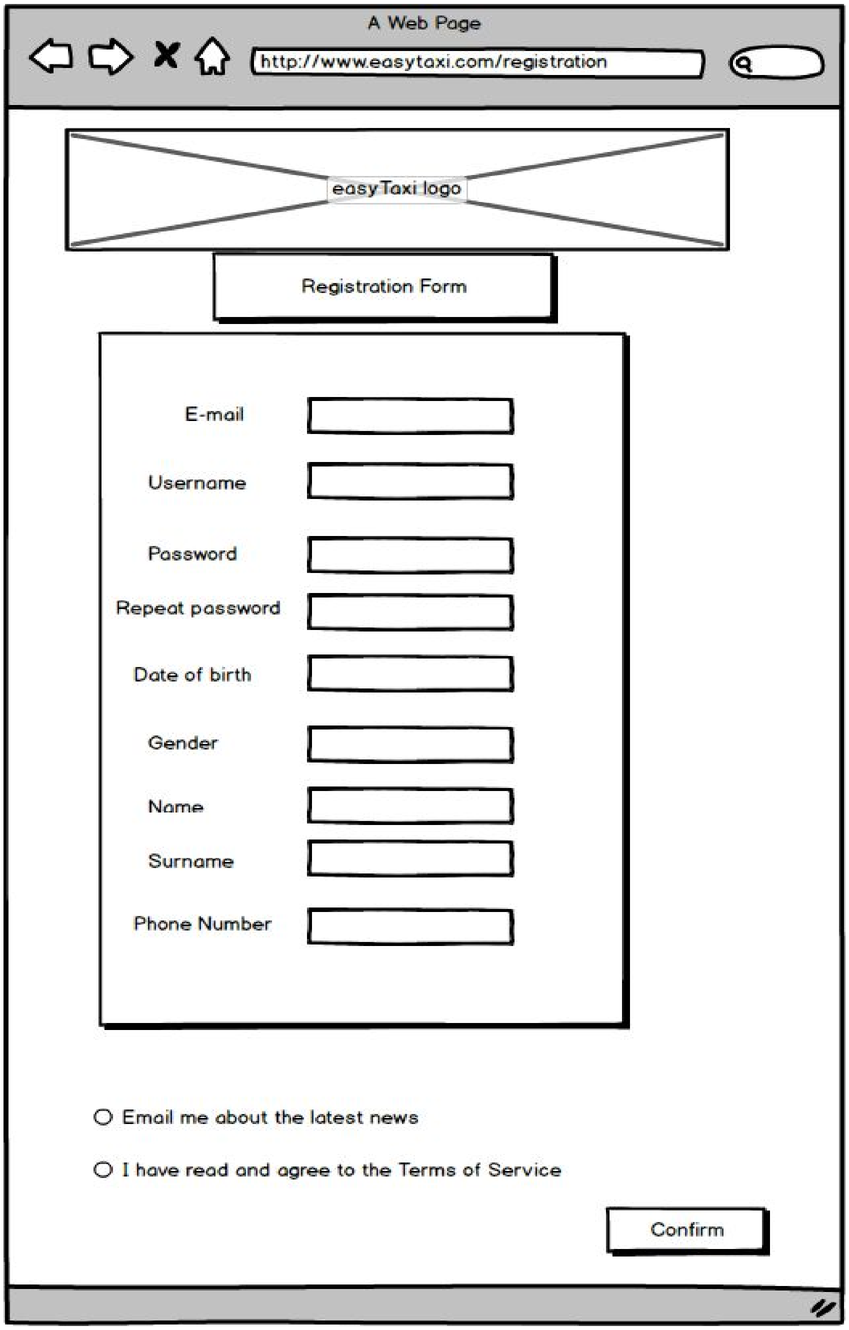


Welcome back!

Do you need a taxi ride?

3.1.1.5 Registration form

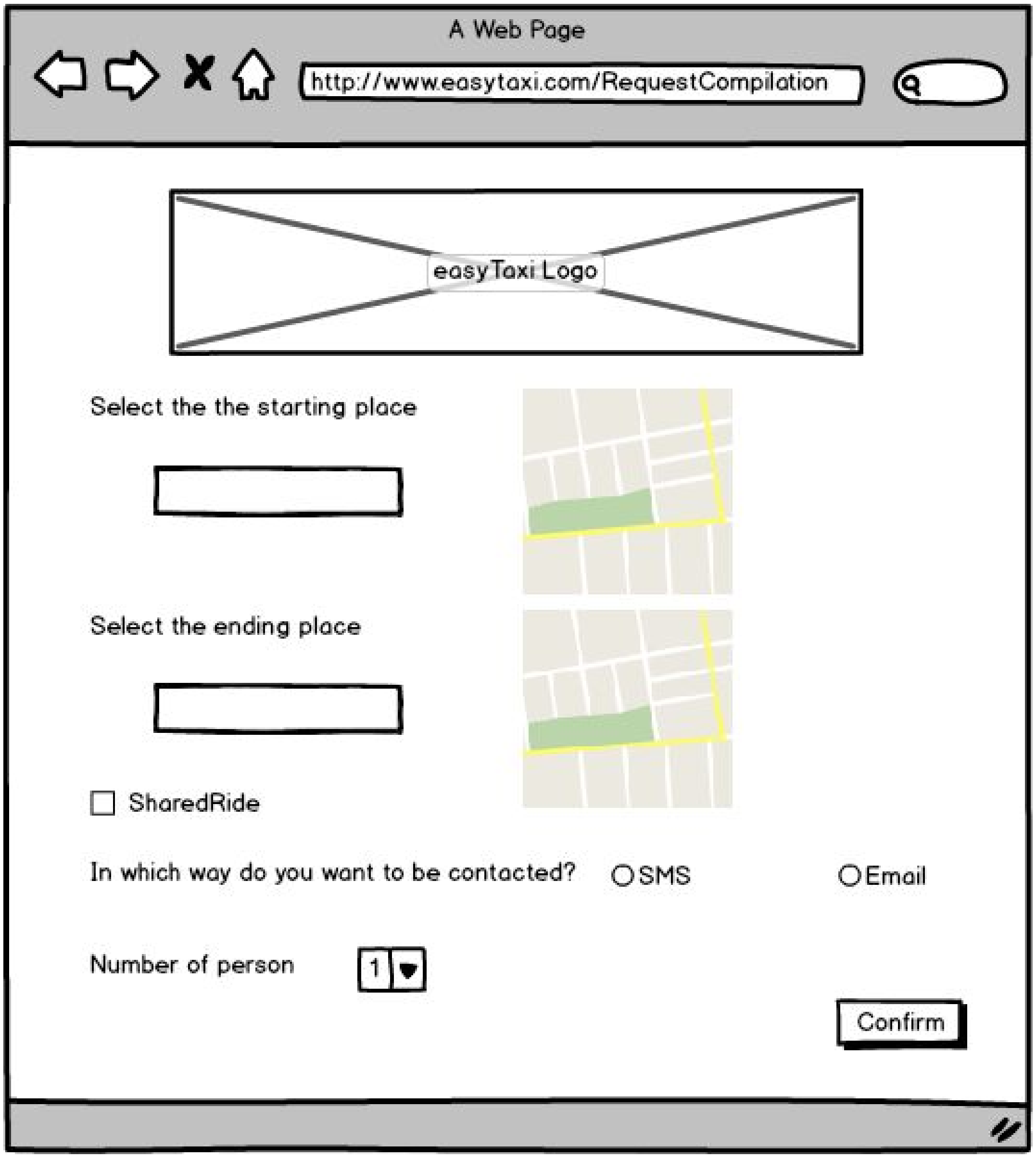
This mockup contains the registration form that the visitor will compile it in order to register himself to the system.



3.1.1.6 Client/Visitor Simple Request On Web Page

This mockup shows the request form the client/visitor has to fill in order to accomplish a request. Shared ride option may be selected.

Client:



A Web Page

http://www.easytaxi.com/RequestCompilation

easyTaxi Logo

Select the the starting place

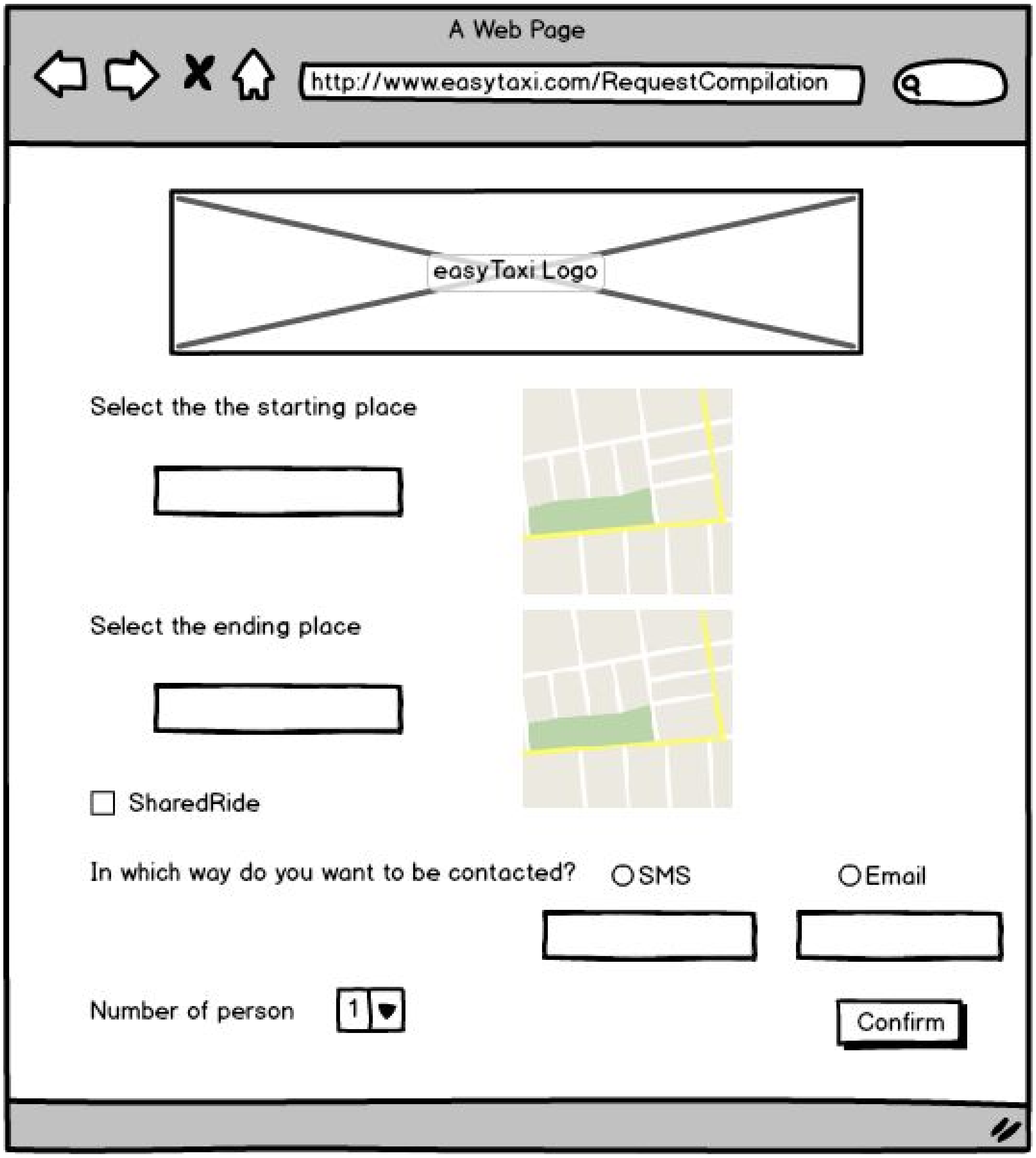
Select the ending place

□ SharedRide

In which way do you want to be contacted? ○SMS ○Email

Number of person 1

Visitor:



A Web Page

http://www.easytaxi.com/RequestCompilation

easyTaxi Logo

Select the the starting place

Select the ending place

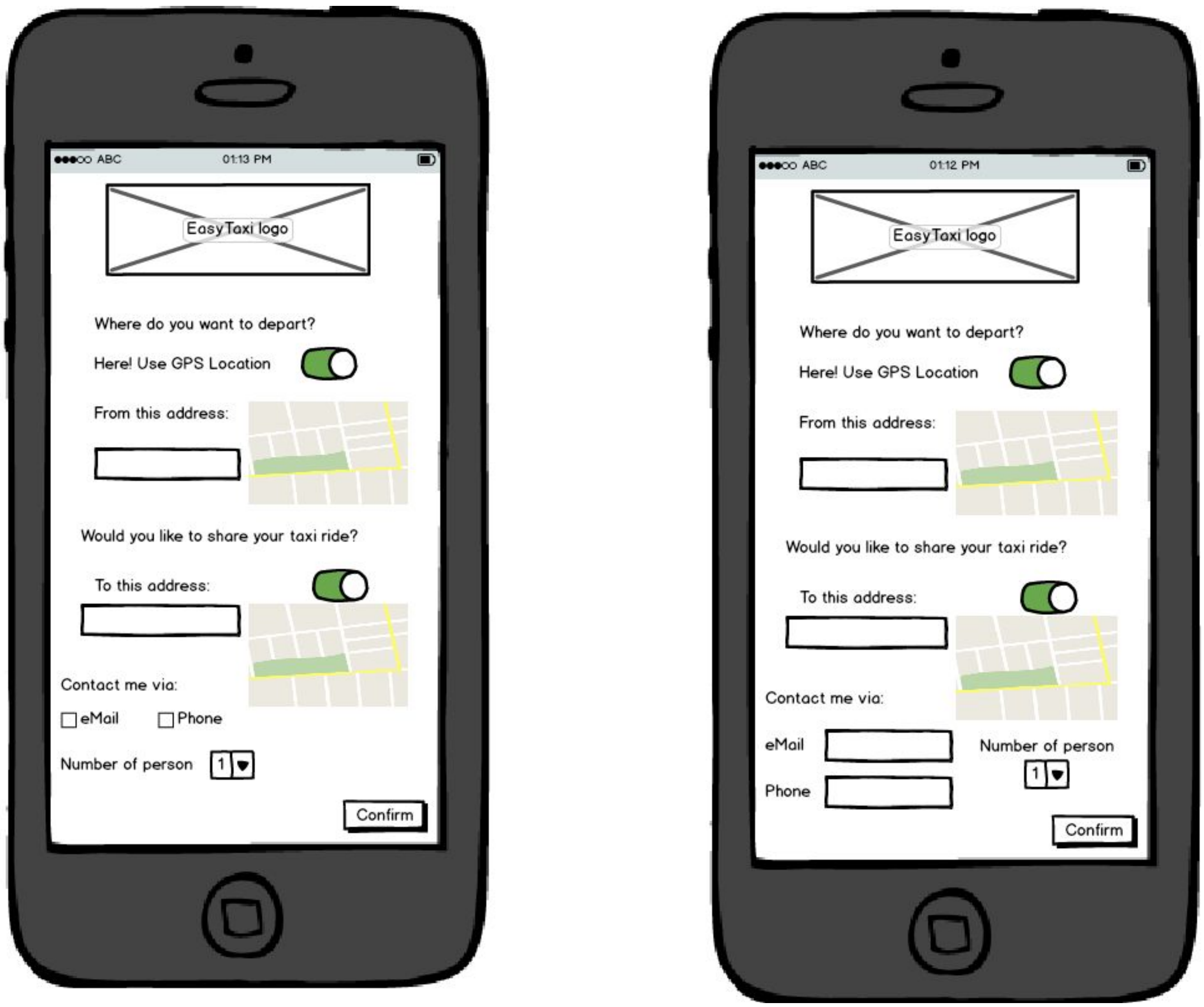
□ SharedRide

In which way do you want to be contacted? ○ SMS ○ Email

Number of person 1 ▼

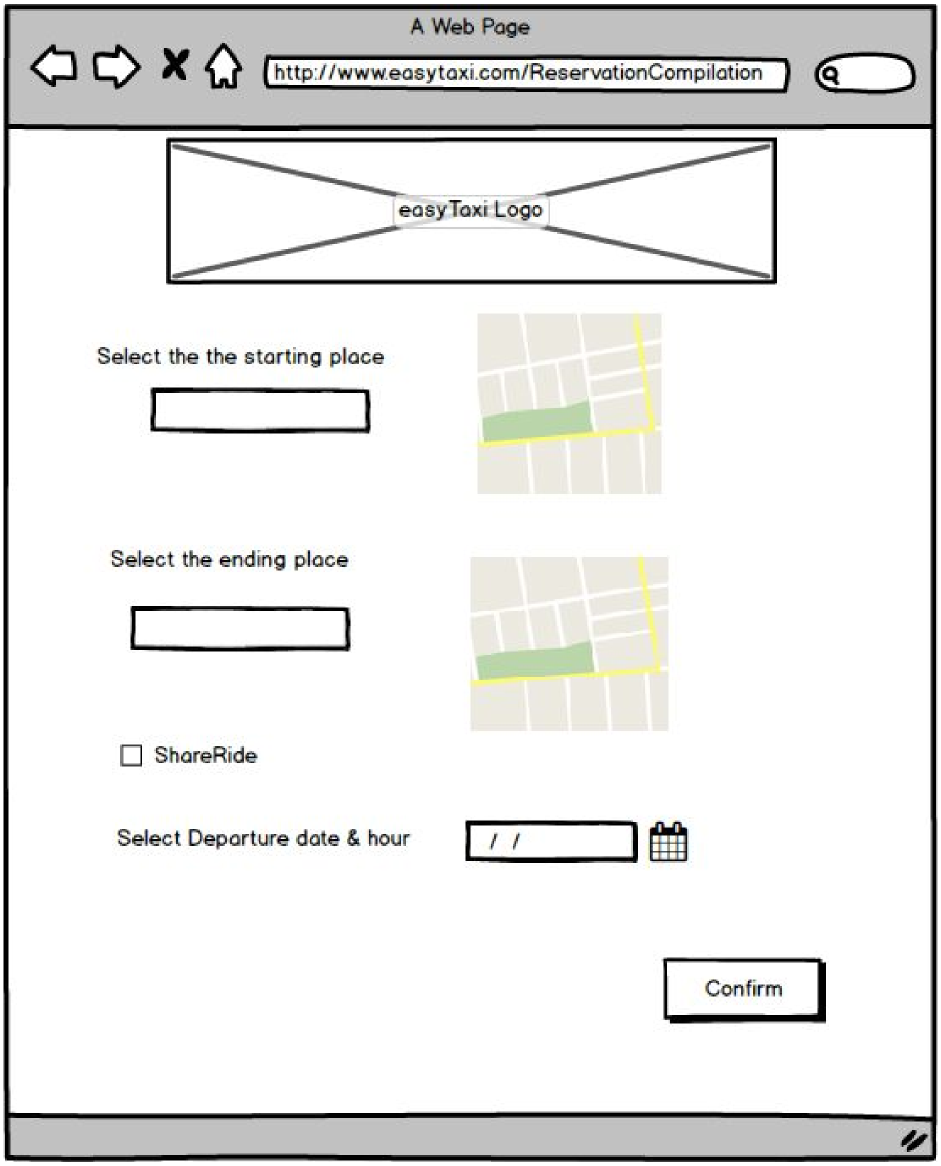
3.1.1.7 Client/Visitor Simple Request On Mobile Application

This mockup shows the form the client/visitor has to fill in order to accomplish a request.



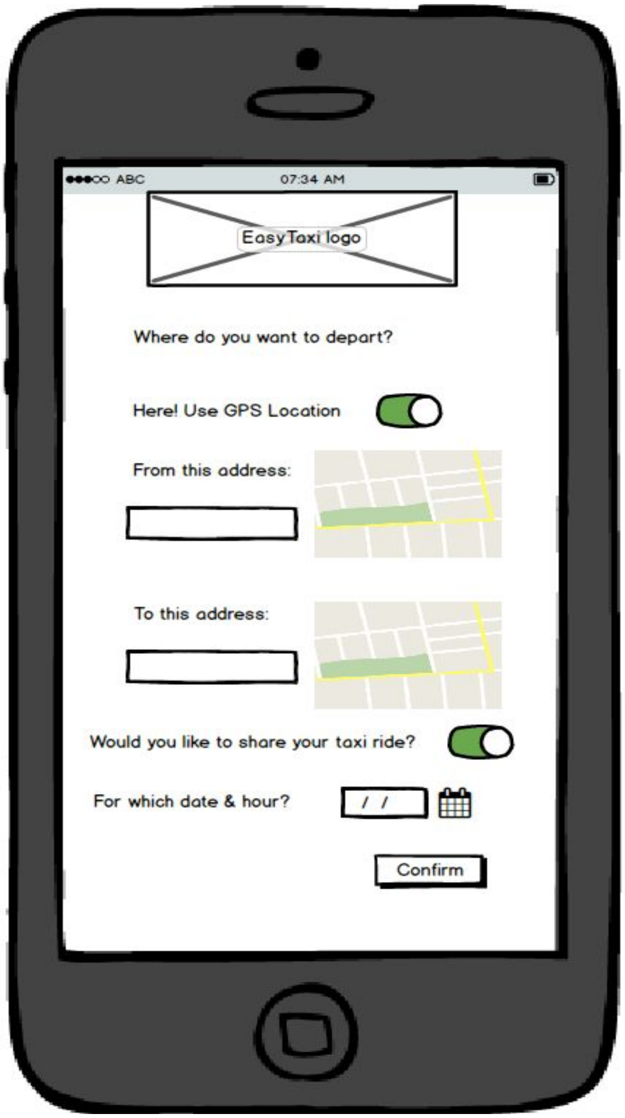
3.1.1.8 Client Reservation On Web Page

This mockup shows the form the client has to fill in order to accomplish a booking. Shared option may be enabled.


A Web Page
http://www.easytaxi.com/ReservationCompilation
easyTaxi Logo
Select the the starting place
Select the ending place
□ ShareRide
Select Departure date & hour
/ /
Confirm

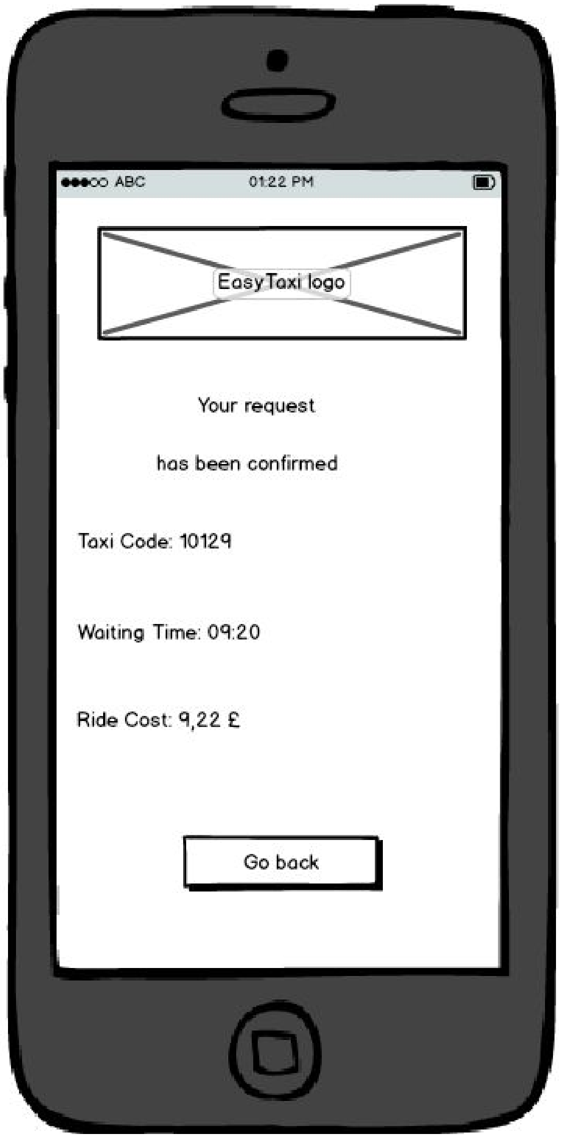
3.1.1.9 Client Reservation On Mobile Application

This mockup shows the form the user has to fill in order to accomplish a booking.



3.1.1.10 Client Generic Request Confirmation

This is the screen that appears to the client/visitor when a request has been accepted by a taxi driver. The same informations are written in the SMS or e-mail, in base of the choice of the client during the reservation



Your request

has been confirmed

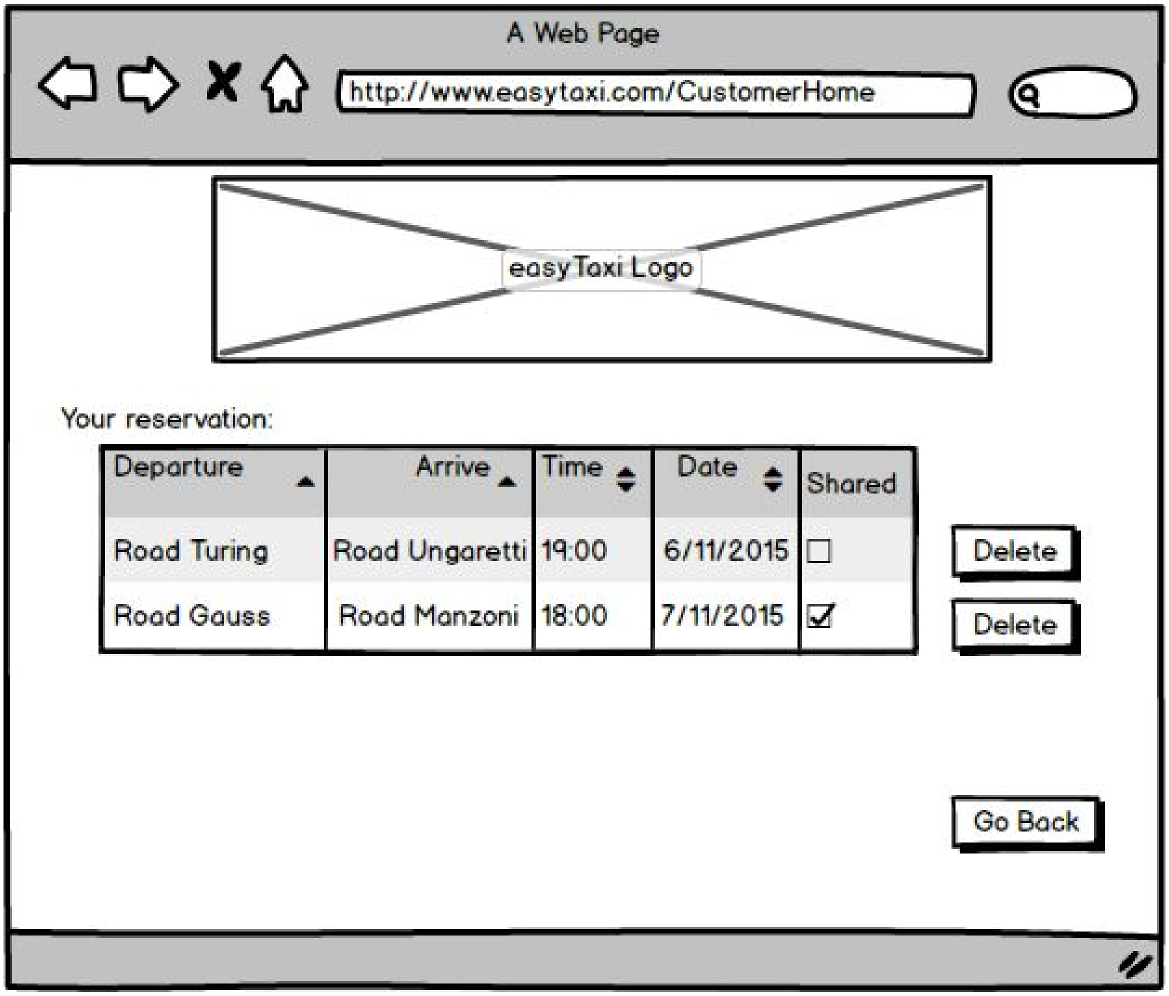
Taxi Code: 10129

Waiting Time: 09:20

Ride Cost: 9,22 £

3.1.1.11 Show Client Booked Rides

This mockup shows all client reservations. From this page the user has the possibility to delete his rides. The screen for the mobile application would be equal.

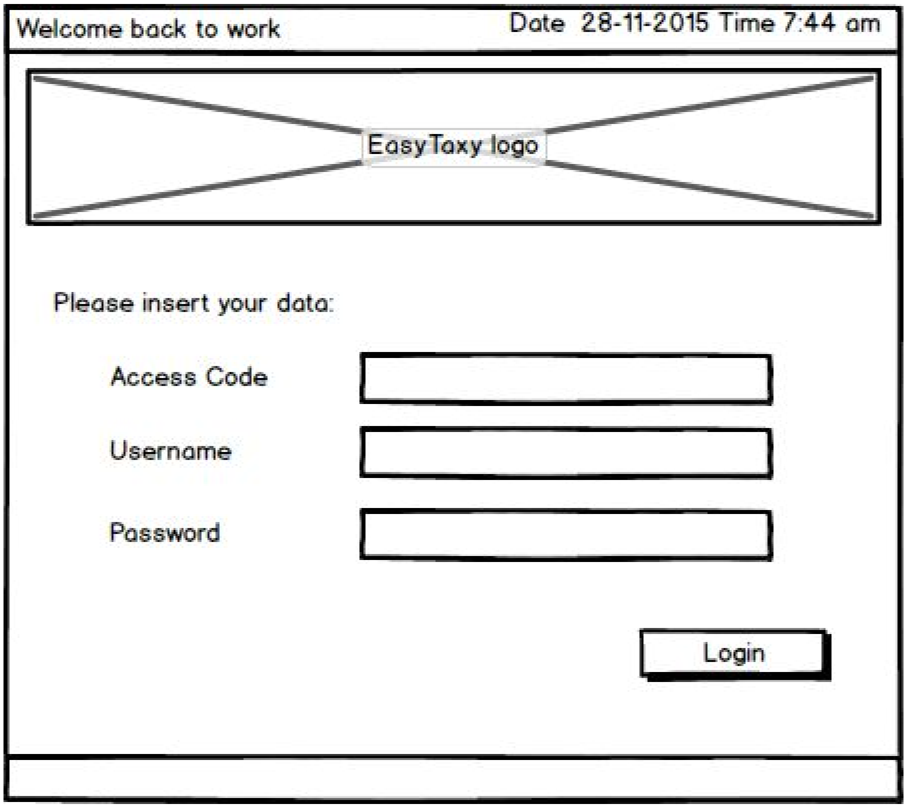


Your reservation:

Departure ▲	Arrive ▲	Time ⬍	Date ⬍	Shared
Road Turing	Road Ungaretti	19:00	6/11/2015	□
Road Gauss	Road Manzoni	18:00	7/11/2015	☑

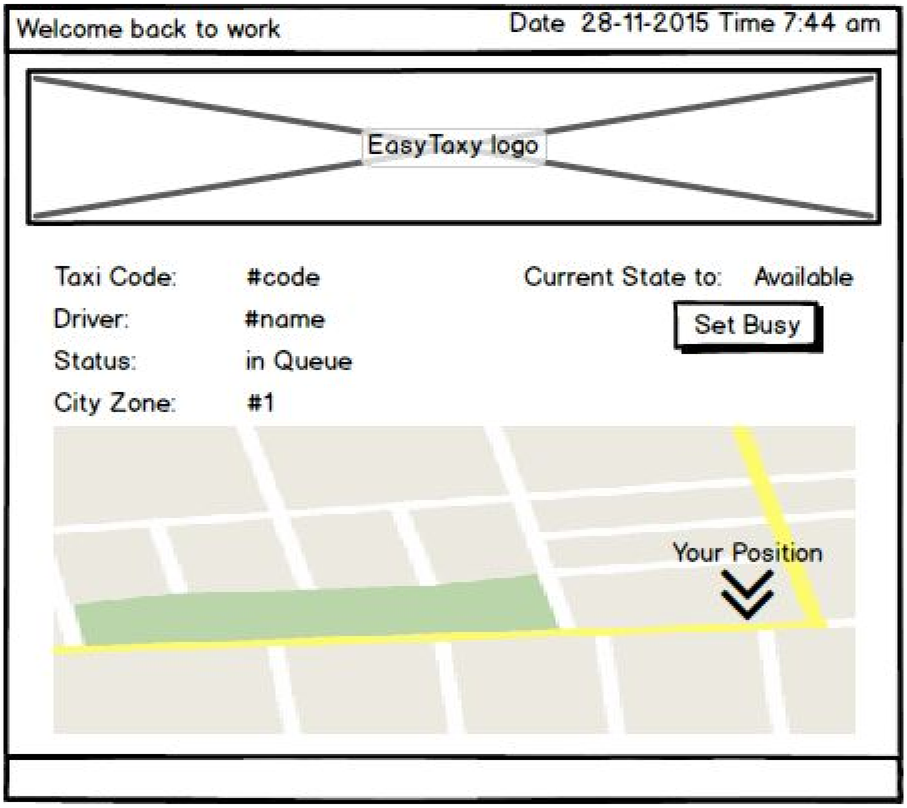
3.1.1.12 Taxi Driver Login Screen

This mockup shows the login form that appears on the onboard computer of the taxi driver.



3.1.1.13 Taxi Driver Main Screen

This mockup shows the taxi driver main screen.



Welcome back to work

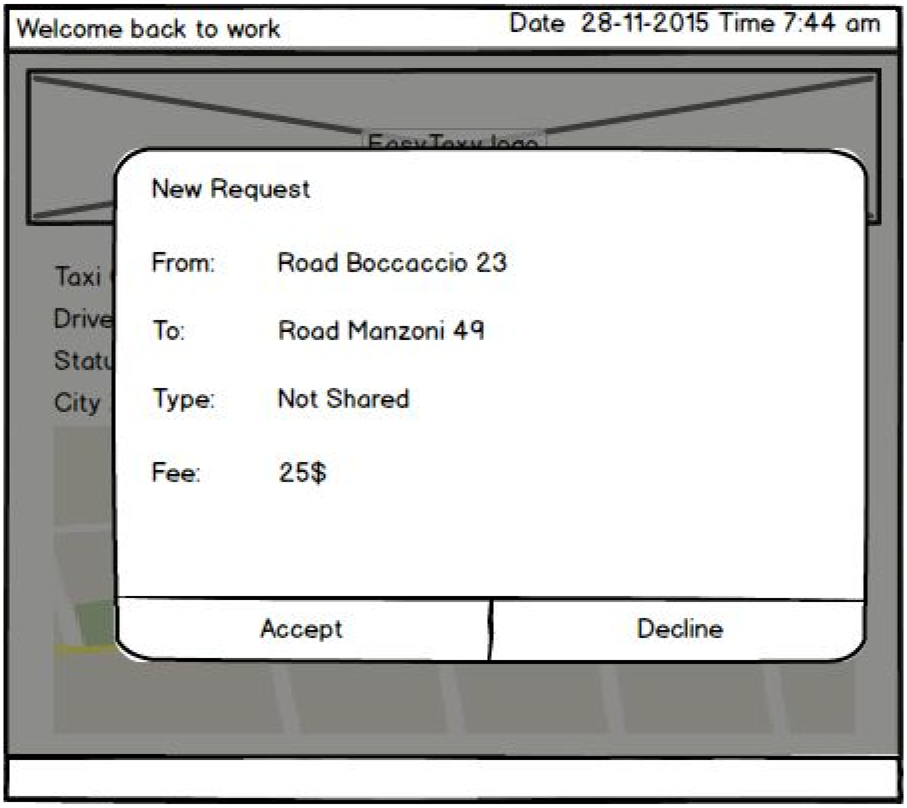
Date 28-11-2015 Time 7:44 am

Taxi Code:	#code
Driver:	#name
Status:	in Queue
City Zone:	#1

Current State to: Available

3.1.1.14 Taxi Driver Request Arrival Screen

On these mockup it’s reported the request the taxi driver can receive remaining in the main screen.



Welcome back to work

Date 28-11-2015 Time 7:44 am

New Request

From: Road Boccaccio 23

To: Road Manzoni 49

Type: Not Shared

Fee: 25$

3.1.2 Software Interfaces

- The application will provide programmatic interfaces for the future development of additional services on top of the basic one.

3.1.3 Communication Interfaces

- The web communications between client and system and through the application server will be performed thanks to the HTTP protocol.

3.2 Functional Requirements

REGISTRATION

- The system will provide a registration module to the visitor containing the following fields: username, password, name, surname, gender, date of birth, e-mail, phone number.
- The system will check the validity of the data inserted by the visitor in the module.

USER LOGIN

- The system will provide a login form containing the fields “username” and “password”.
- The system will check the validity of the data inserted by the visitor in the login form.

GENERIC REQUEST MANAGEMENT(valid both for requests and reservations)

- The system will grant request confirmations using sms or email notification according to the user choice.
- The system will manage all inconsistent generic requests compilation by refusing them and asking to compilate again the fields.
- After a ride has been accepted by a taxi driver, the system will show the waiting time for the arrival of the taxi and the code of the incoming taxi to the client.
- If a generic request contains the destination of the ride, after that ride has been accepted by a taxi driver, the system will show the fee to the client.

SIMPLE REQUEST MANAGEMENT

- The system will provide to visitors and clients a page containing a form for the compilation of a simple request or a simple shared request.

SHARED REQUEST / SHARED RESERVATION MANAGEMENT

- After the compilation of a shared request by client (or visitor) A, the system will search the most compatible shared ride between the current rides with shared option activated, and if it finds that the most compatible route shares more than the 80% of the path, the system will create a unique path for both the client(or visitor) A and for the other client (or visitor), otherwise the system will create a new request initially not shared for the client (or visitor) A.

- After having fixed a shared compatible route between two or more users, the system will calculate the shared ride cost, and notify it to the taxi driver and to the clients involved.
- After having fixed a shared compatible route, the system will notify the new route to the taxi driver.

RESERVATION MANAGEMENT

- The system will provide to clients a page containing a form for the compilation of a reservation or a shared reservation.
- The system will permit to the clients to view all their booked requests not yet happened.
- The system will start the procedure for the assignment of a taxi to a client 10 minutes before the start of the booked ride
- The system will permit a user to delete his reservation until it doesn't last less than 10 minutes to the start of the booked ride.

QUEUE MANAGEMENT

- The system will manage a generic request by sending a notification to an available taxi driver chosen according to a FIFO queue policy.
- The system will be able to move a taxi from a queue zone to another one, according to its GPS position.

GPS LOCALIZATION

- The system will allow taxi drivers to visualize his position in the city's map by using the GPS localization.
- The mobile application will permit the user to use GPS localization to express his position during a generic request compilation.

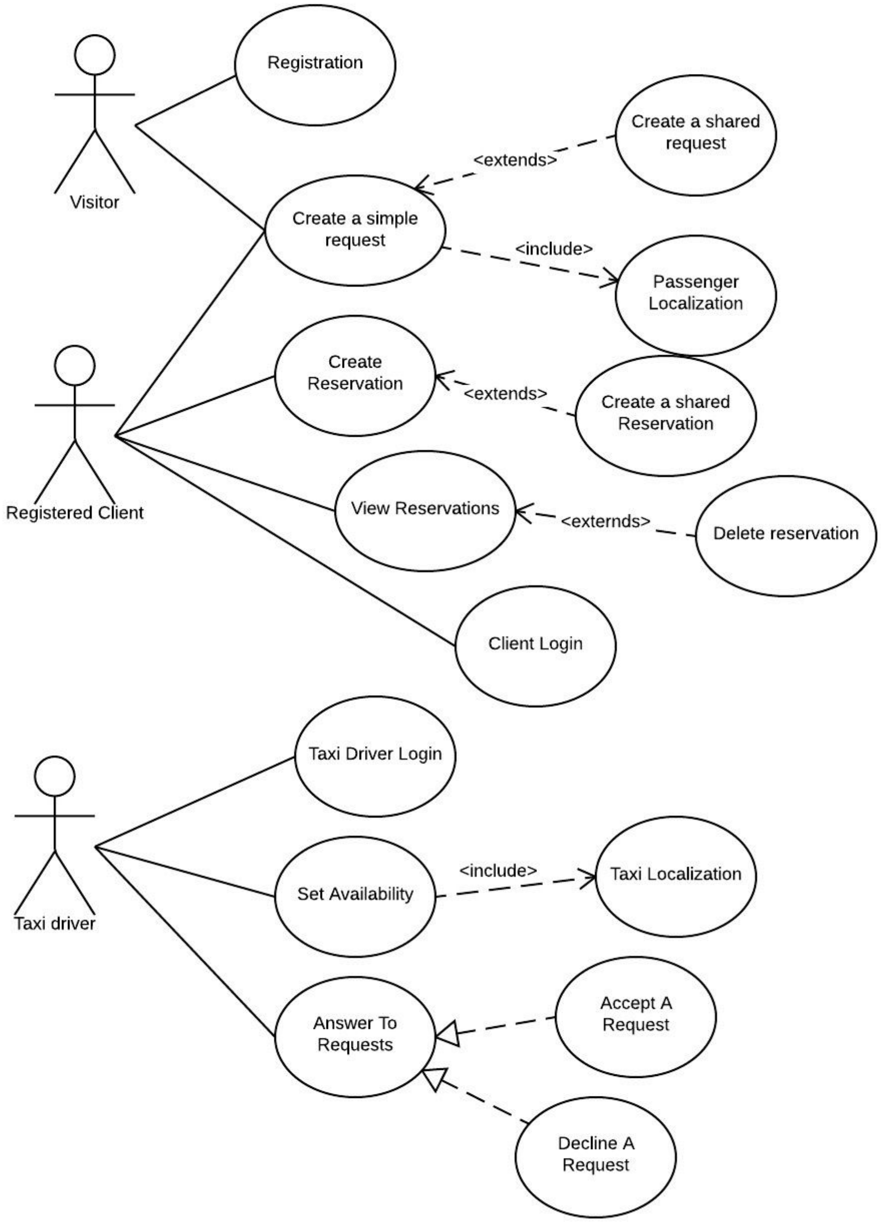
TAXI DRIVER FUNCTIONALITIES

- The system will let a taxi driver to change his state from "busy" to "available" and conversely.
- Taxi drivers can change his state to "available" when he isn't performing any ride and when the GPS verifies that the taxi is not out of the boundary of the city.
- The system will let the taxi driver to accept or refuse a generic request.
- A taxi driver will be provided of all the request attributes, that is all the available informations of a ride for deciding to accept it or not, apart the exact route to follow and the personal data of the client/visitor who made the request.
- After a ride has been accepted by a taxi driver, the system will show to him the route he has to follow.

TAXI DRIVER LOGIN

- The system will provide a login form to the taxi driver containing the fields "username" and "password" and "access code".
- The system will check the validity of the data inserted by the taxi driver in the login form.

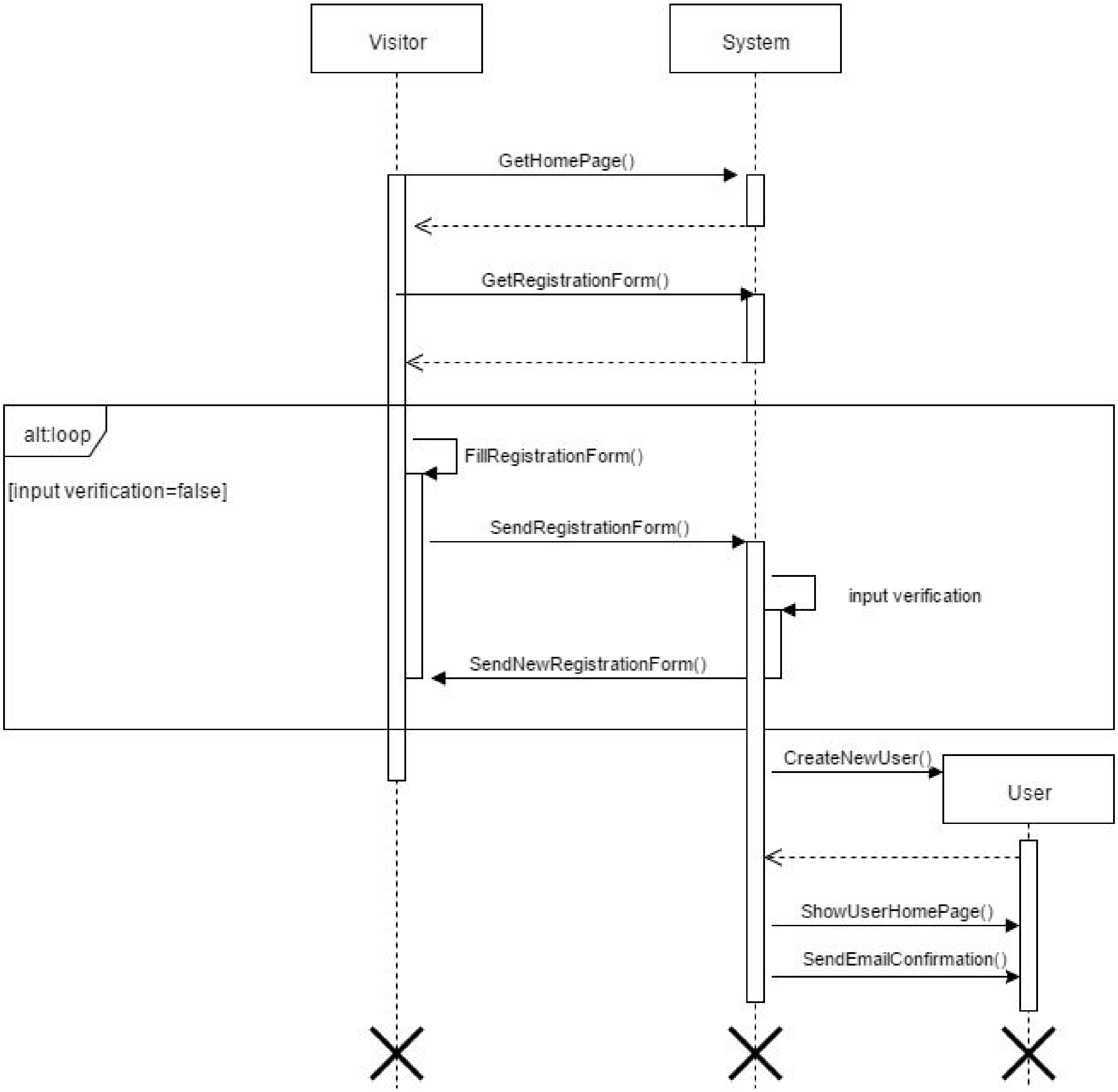
3.3. Use Case Diagram



3.4 Use Cases and Sequence Diagrams

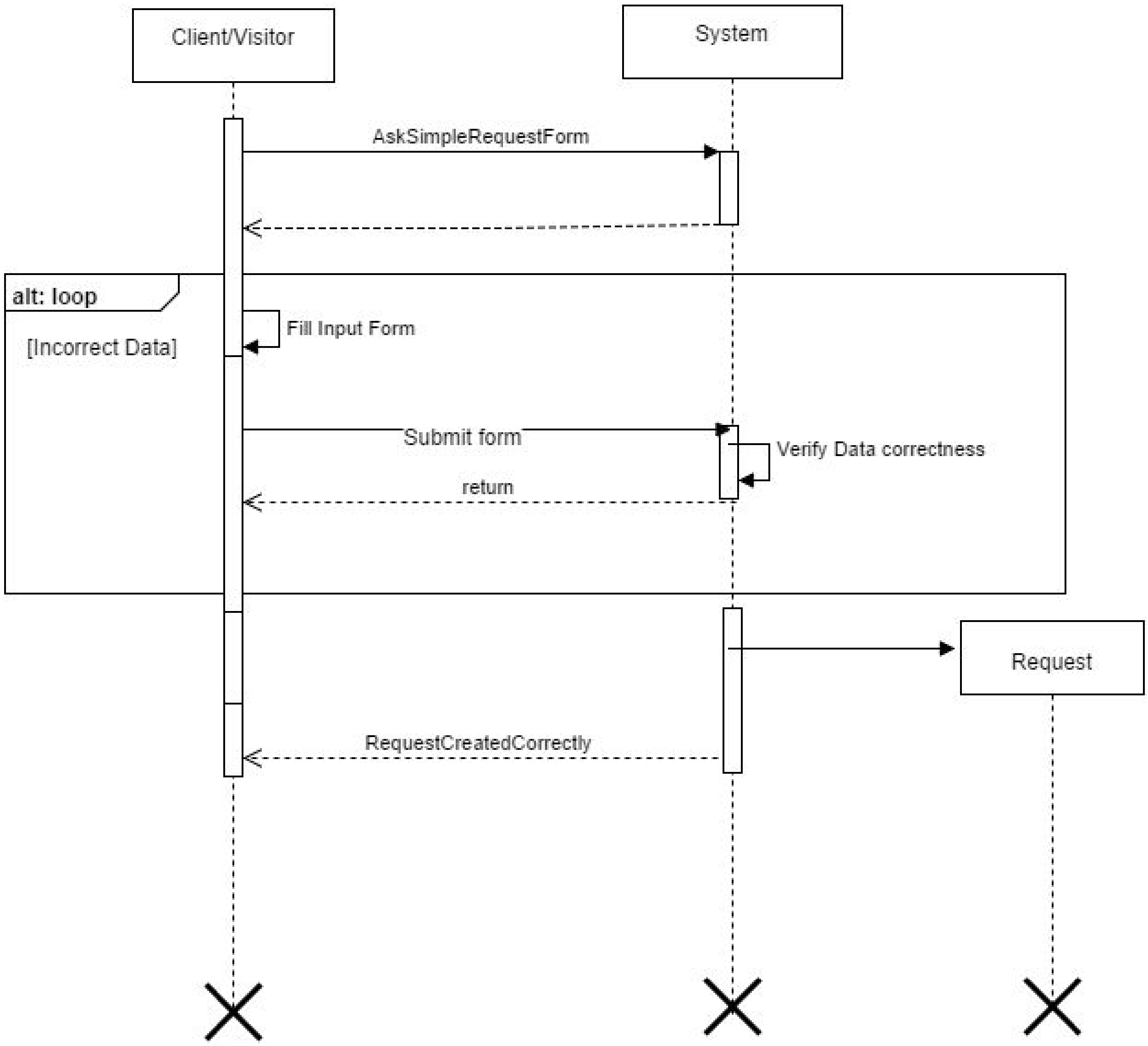
3.4.1 Registration to EasyTaxi

Actor	Visitor
Goal	[G1]
Precondition	The guest had never before registered into the system
Entry condition	This use case starts when the 'Subscribe' button is clicked on the homepage
Event flow	1)Visitor fills all the fields in the registration form. 2)Visitor must accept terms and conditions 3)Visitor can decide if subscribe to email updates services. 4)Visitor confirms his decisions 5)The system verifies that all data inserted are correct
Exit Condition	The system sends an email to confirm the registration.
Constraints	● The username inserted by the visitor must not match with another existent one. ● The e-mail inserted is valid and if it isn't already used by another user. ● The password must be long at least 6 characters. ● The string inserted in the "repeat password" field must be equal to the string inserted in the password field. ● The birth date must be inferior of the current date
Exceptions	One or more inserted data are incorrect: the visitor has to refill the non-valid fields


Visitor
System
GetHomePage()
GetRegistrationForm()
alt:loop
[input verification=false]
FillRegistrationForm()
SendRegistrationForm()
input verification
SendNewRegistrationForm()
CreateNewUser()
User
ShowUserHomePage()
SendEmailConfirmation()

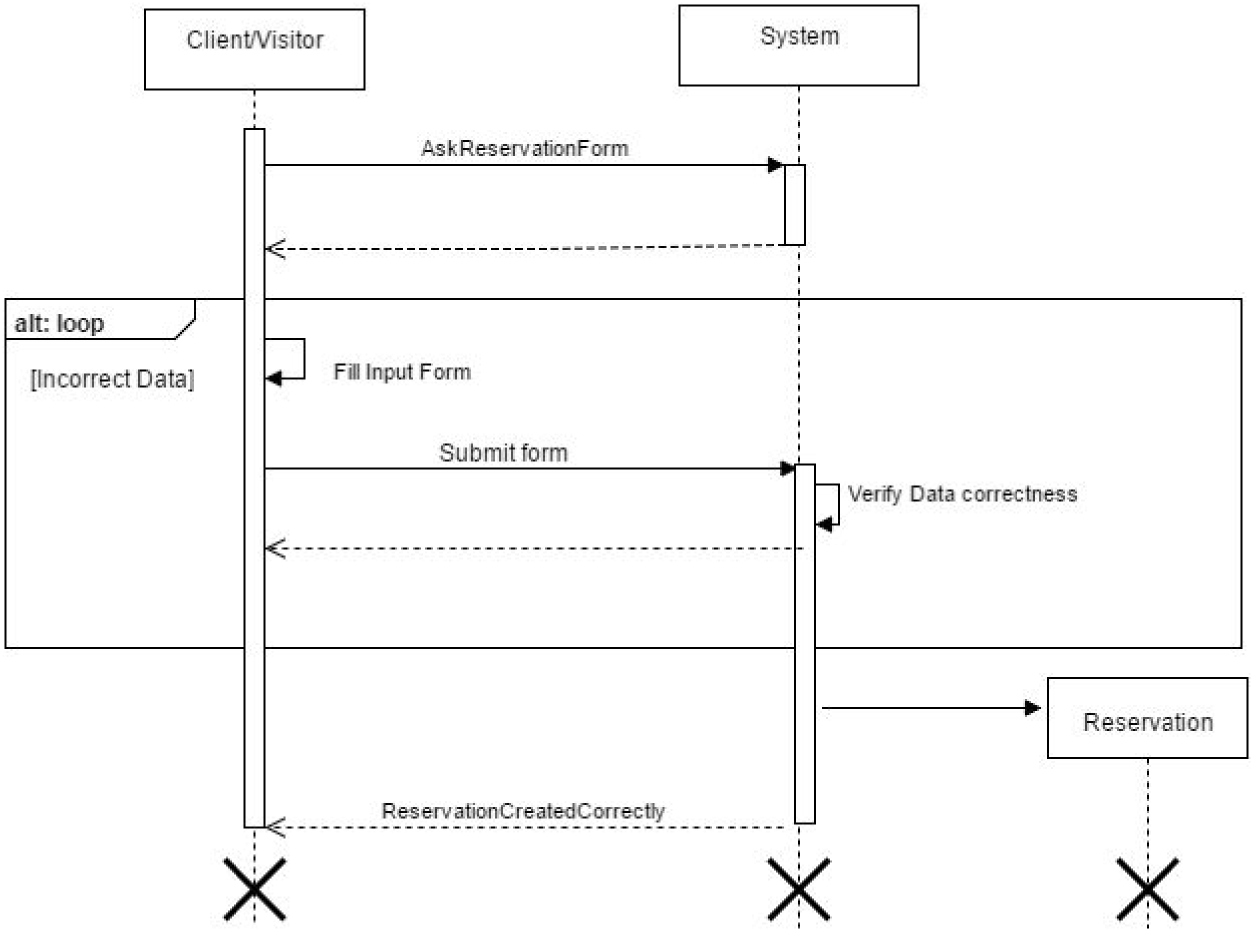
3.4.2 Create A Simple Request

Actor	Client/Visitor
Goal	[G2] [G3] [G5]
Precondition	The visitor must be on the visitor web or mobile homepage.
Entry condition	This use case starts when the ‘Request now!’ button is clicked on the visitor homepage.
Event flow	1)The system displays compilation request page. 2)The visitor/client writes the starting place of his ride or, if possible, let the GPS to do it automatically. 3)The visitor/client decides to activate the shared ride option. 4)The visitor/client can select the destination of the ride. 5)The visitor/client selects the number of passengers of his ride. 6)The visitor/client decides to be contacted by email or sms for the future confirmation of the request. 7)The visitor/client confirms his choices. 8)The system verifies the correctness and the validity of the data inserted.
Exit Condition	The system send an email or an sms to the visitor/client saying the request has been accepted.
Constraints	● The visitor/client cannot leave neither the “select starting place” field, nor the choice of been contacted by email or sms, as empty fields. ● The visitor/client cannot specify as the starting place of a ride a position outside the frontier of the city ● The destination, if it has been specified by the visitor/client, must not be farther than 10 km from the nearest frontier of the city. ● If the shared ride option has been activated, the visitor/client cannot leave empty the “destination” field. ● The visitor/client must specify a number of passengers included between 1 and 4 in the case of not shared request. ● If the shared option is activated, the number of passengers must be between 1 and 3. ● The visitor has to write the SMS or the email, because these data are not registered in the system. The client only chooses the way to be contacted by clicking on his favourite choice.
Exceptions	The data inserted into the system doesn’t correspond to any existent location or the locations are not acceptable or some mandatory fields are missing: in this case, the system ask to the visitor/client to modify the invalid fields.


Client/Visitor
System
AskSimpleRequestForm
alt: loop
[Incorrect Data]
Fill Input Form
Submit form
Verify Data correctness
return
Request
RequestCreatedCorrectly

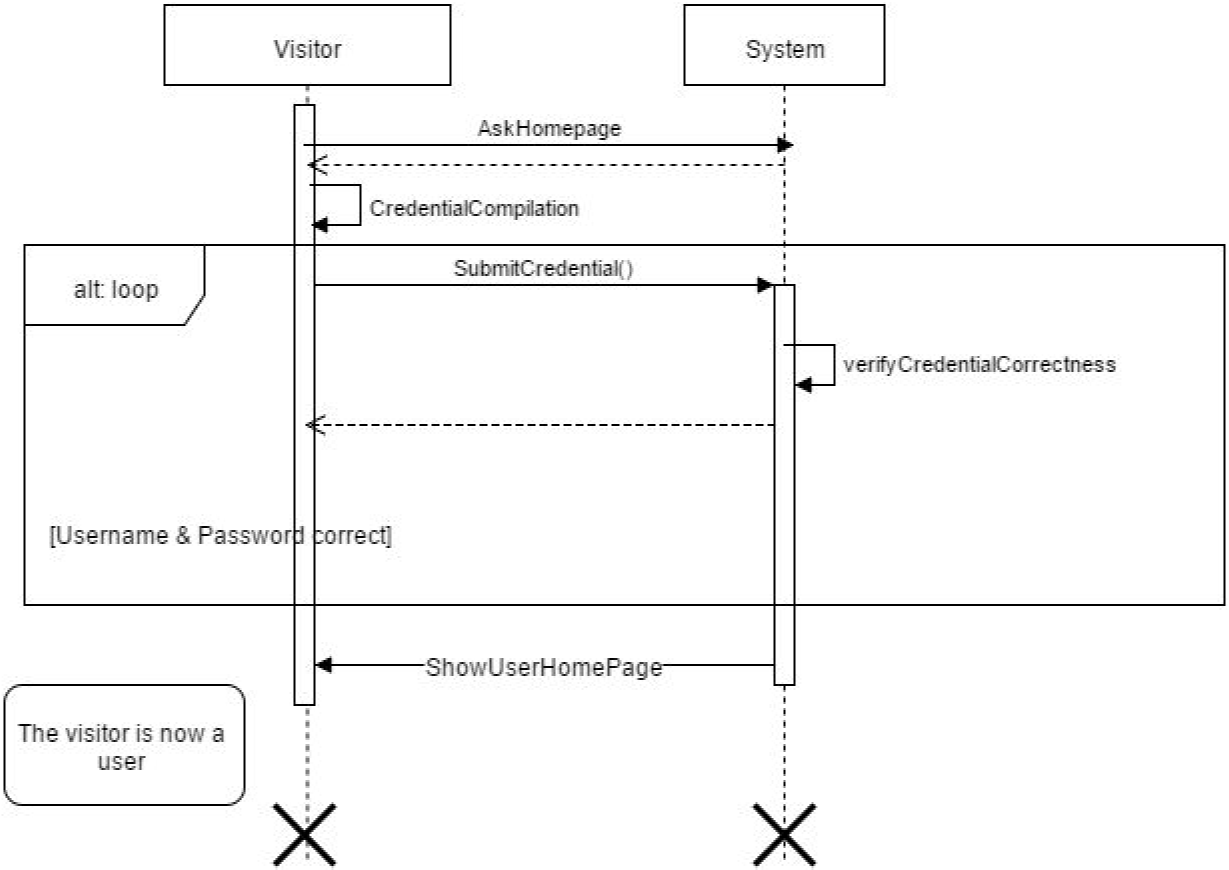
3.4.3 Create A Reservation

Actor	Client
Goal	[G4][G5]
Precondition	The registered client has already logged into the system
Entry condition	This use case starts when the ‘Reserve it for later!’ button is clicked on the registered client homepage.
Event flow	1)The system displays reservation compilation page. 2)The client writes the starting place of his ride or, if possible, let the GPS to do it automatically. 3)The client specifies the destination, the number of passengers, the date and the hour of the ride he intends to book. 4)The client decides to activate or not the shared ride option. 5)The system verifies that the inserted data satisfy the constraints and that are syntactically valid.
Exit Condition	The system shows to the registered client a popup with the confirm message of the reservation.
Constraints	● The module must be compiled in every part and no text input form can be left empty; the only non mandatory field is the choice for the activation of the shared ride. ● The reservation date and hour must be included between two hours after the request has been sent and two days before it. ● The user cannot specify as the starting place of a ride a position outside the frontier of the city ● The destination, if it has been specified by the user, must not be farther than 10 km from the nearest frontier of the city. ● The user must specify a number of passengers included between 1 and 4 in the case of not shared reservation. ● If the shared option is activated, the number of passengers must be between 1 and 3.
Exceptions	The data inserted into the system doesn’t correspond to any existent location or the locations are not acceptable or some mandatory fields are missing: in this case, the system ask to the client to modify the invalid fields.


Client/Visitor
System
AskReservationForm
alt: loop
[Incorrect Data]
Fill Input Form
Submit form
Verify Data correctness
Reservation
ReservationCreatedCorrectly

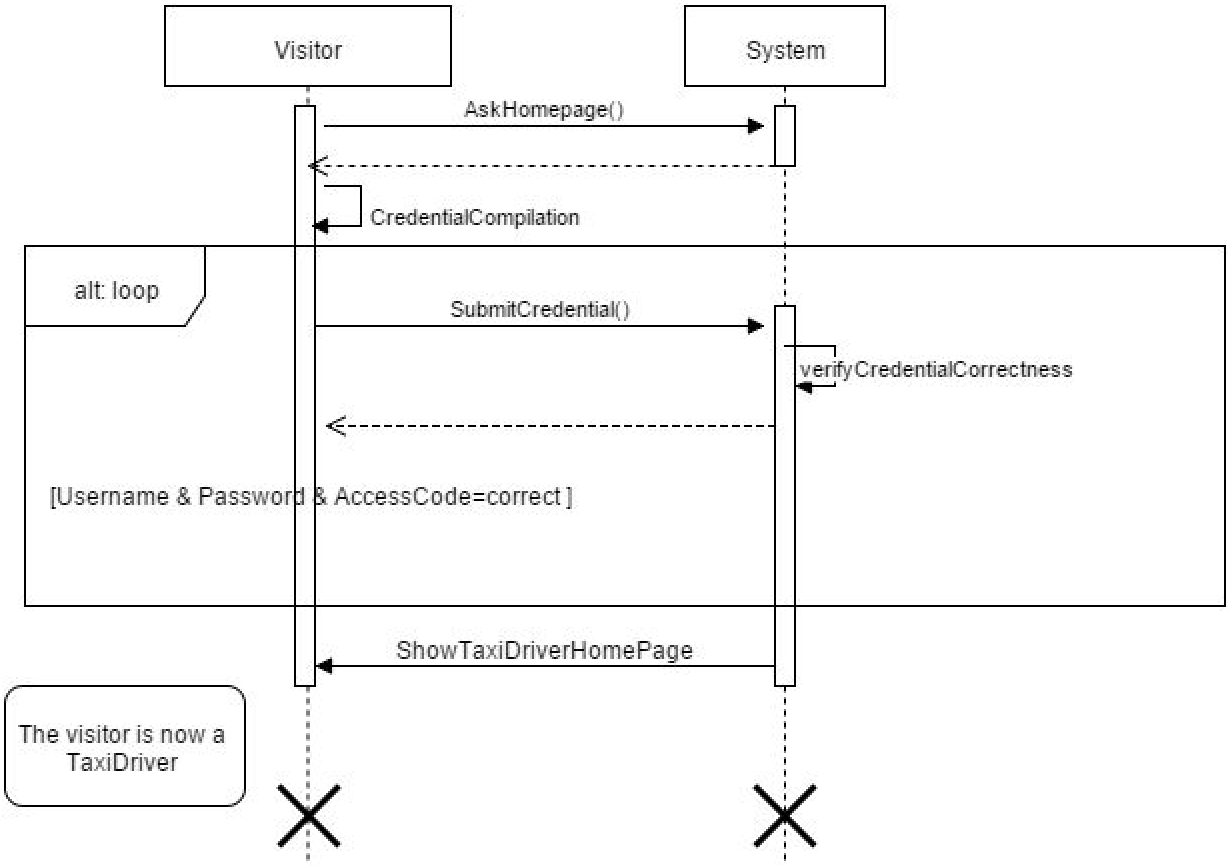
3.4.4 Client Login

Actor	Visitor
Goal	[G8]
Precondition	The visitor has successfully registered into the system
Entry condition	This use case starts when the ‘Login’ button is clicked on the homepage
Event flow	1)The visitor fills the username and password text input form. 2) The visitor presses on ‘Confirm’ button 3) The system verifies that all data inserted are valid and that satisfy the constraints.
Exit Condition	The system redirects the visitor into the client homepage.
Constraints	● There must be correspondence between the username and the password inserted. ● Username and password input form cannot be left empty.
Exceptions	The system finds out that the data are not existent or incorrect: the visitor can’t log in and he has to fill again the two fields.


Visitor
System
AskHomepage
CredentialCompilation
alt: loop
SubmitCredential()
verifyCredentialCorrectness
[Username & Password correct]
ShowUserHomePage
The visitor is now a user

3.4.5 Taxi Driver Login

Actor	Taxi Driver
Goal	[G7]
Precondition	The taxi driver has already an existent account.
Entry condition	This use case starts when the a taxi driver is on the main screen on the mobile device and clicks on the 'Login' button for taxi driver's login.
Event flow	1)System redirects the taxi driver into taxi driver login screen. 2) The taxi driver fills the text input with his user data (username, password and access code). 3)The taxi driver press the 'Confirm' button. 4)The system checks data correctness.
Exit Condition	System redirect the registered taxi driver into his homepage.
Constraints	● There must be correspondence between the username, the password and the access code inserted. ● All the fields are mandatory and cannot be left empty.
Exceptions	The system finds out that the data are not existent or incorrect: the visitor can't log in and he has to fill again all the three fields.

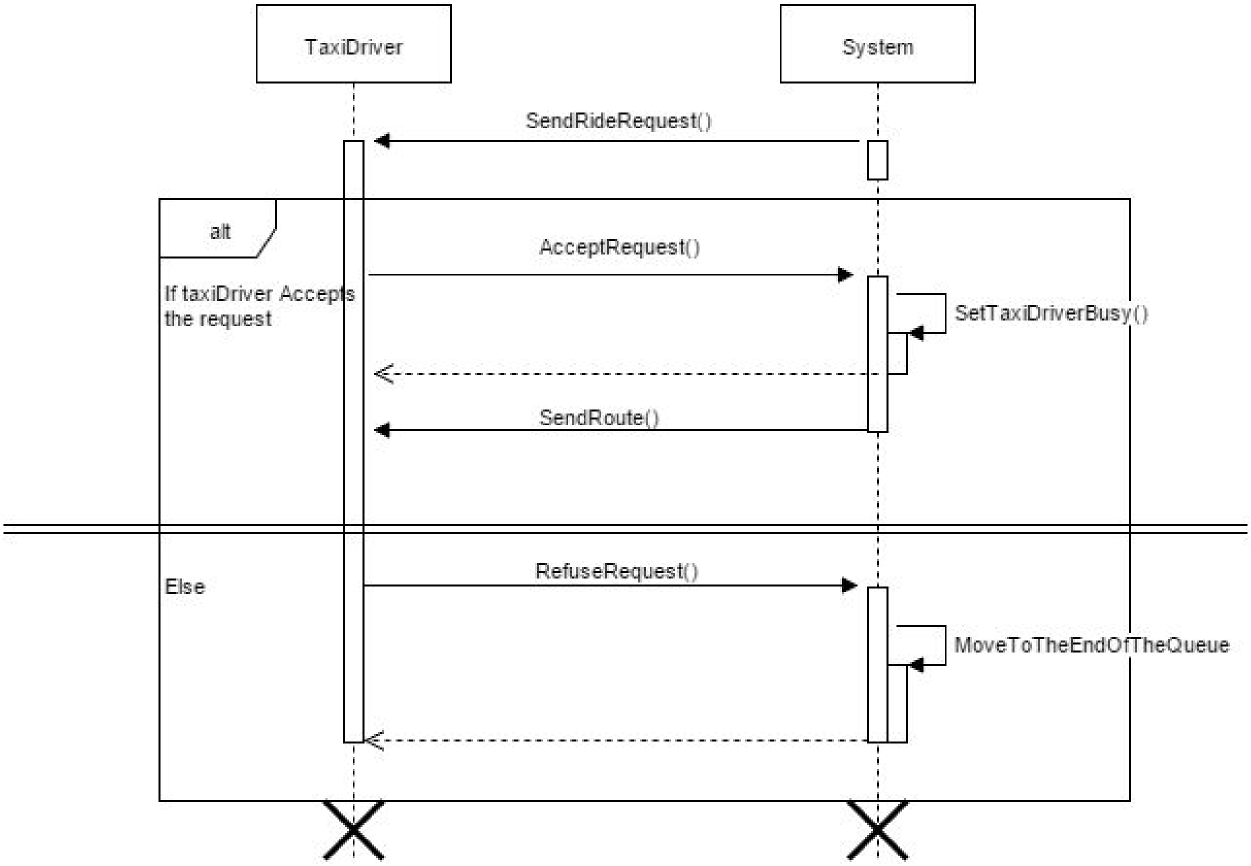

Visitor
System
AskHomepage()
CredentialCompilation
alt: loop
SubmitCredential()
verifyCredentialCorrectness
[Username & Password & AccessCode=correct]
ShowTaxiDriverHomePage
The visitor is now a TaxiDriver

3.4.6 Accept A Generic Request

Actor	Taxi Driver
Goal	[G6]
Precondition	The taxi driver is logged in and his state is set to “available” and he is in the first position of the queue of the zone where the passenger is located
Entry condition	This use case starts when a generic request appears on the taxi driver mobile device, showing the starting place of the ride and possibly the destination of the ride and its cost.
Event flow	1)The taxi driver accepts the request by clicking on the correspondent button. 2)The system set the taxi as busy and remove it from the queue of the zone.
Exit Condition	The system let the taxi driver visualize the route he has to follow in order to reach the first client
Constraints	Taxi drivers has 15 seconds to accept or refuse the ride.
Exceptions	If 15 seconds has passed from the reception of the request, the taxi is moved at the end of the queue of the zone where it is located, as if the taxi driver had declined the request

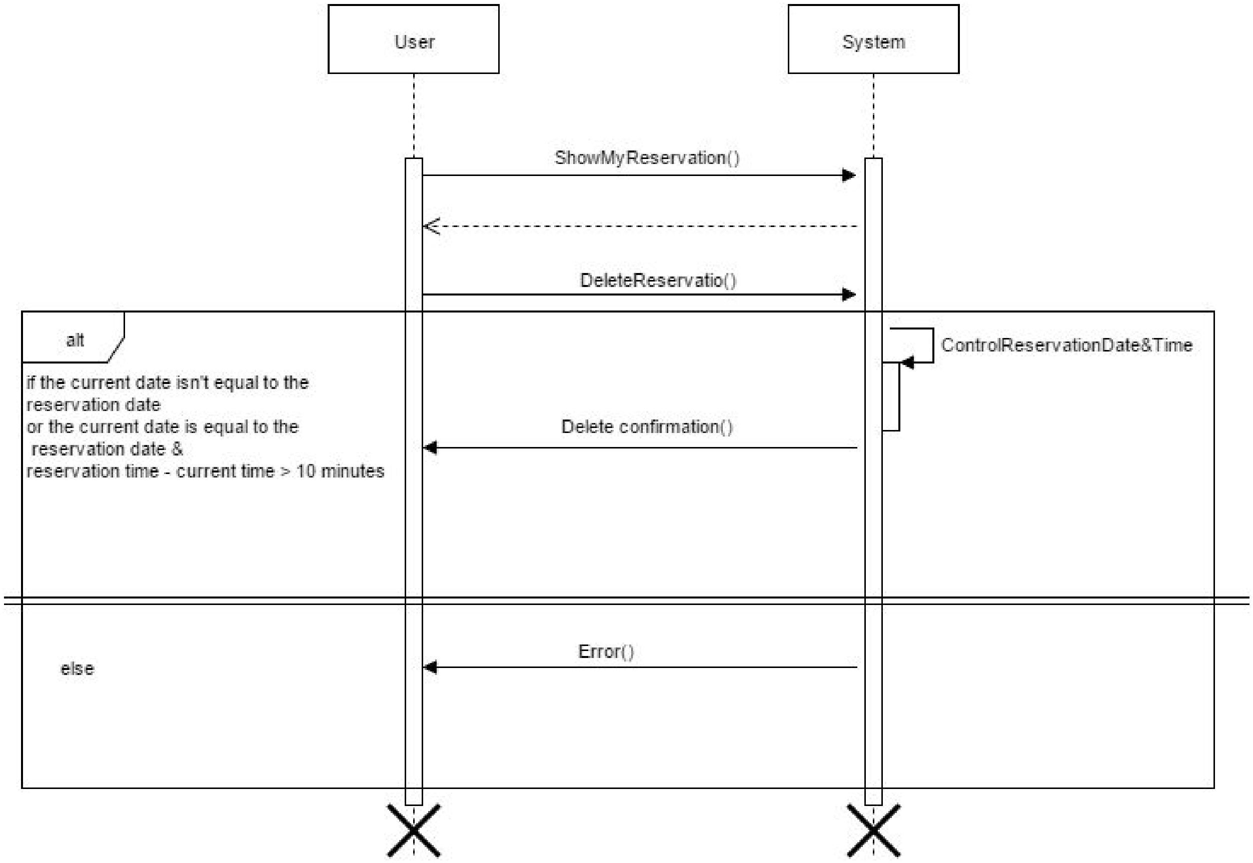
3.4.7 Decline A Generic Request

Actor	Taxi Driver
Goal	[G6]
Precondition	The taxi driver is logged, his state is set to “available” and he is in the first position of the queue of the zone where the passenger is located
Entry condition	This use case starts when a generic request appears on the taxi driver mobile application, showing the starting place of the ride and, possibly, the destination of the ride and its cost.
Event flow	The taxi driver declines the request by clicking on the correspondent button.
Exit Condition	The system moves the taxi from the top of the queue to the end of it.
Constraints	Taxi drivers has 15 seconds to accept or refuse the ride.
Exceptions	If 15 seconds has passed from the reception of the request, the taxi is moved at the end of the queue of the zone where it is located, as if the taxi driver had declined the request

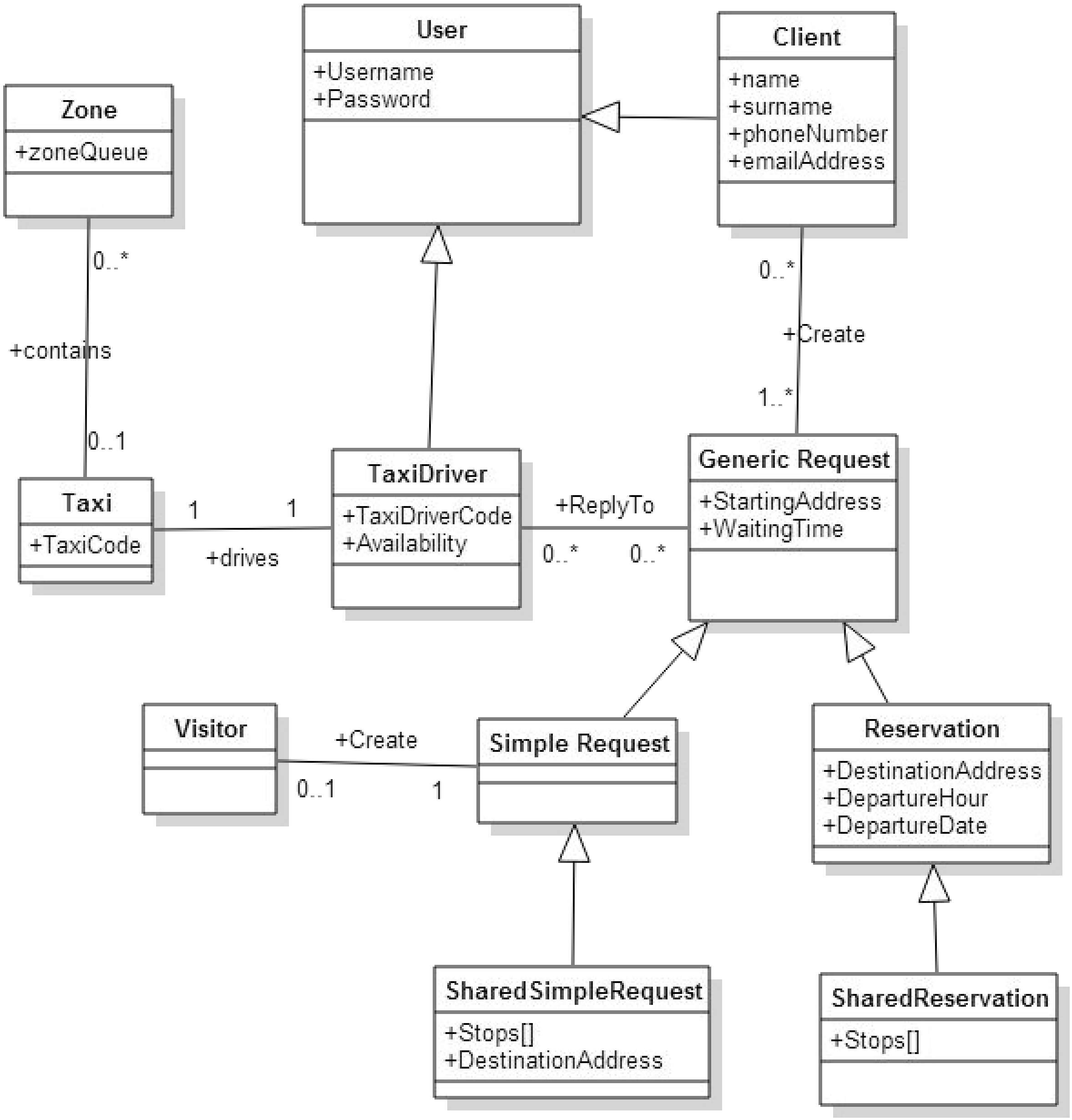

TaxiDriver
System
SendRideRequest()
alt
AcceptRequest()
If taxiDriver Accepts the request
SetTaxiDriverBusy()
SendRoute()
Else
RefuseRequest()
MoveToTheEndOfTheQueue

3.4.8 Delete A Reservation

Actor	Client
Goal	[G9]
Precondition	The registered client has already logged into the system and has booked at least one ride for the immediate future.
Entry condition	This use case starts when the ‘Show My Reservations’ button is clicked on the client homepage.
Event flow	1)The system shows to the client the list of future booked rides. 2)The client clicks on the “delete” button corresponding to the reservation that the client wants to delete.
Exit Condition	The system cancel from the database the reservation and the result is shown on the page.
Constraints	A client cannot delete reservations that are going to start less than minutes after the current time.
Exceptions	The system doesn’t permit to delete those reservations, the delete button will not be clickable.


User
System
ShowMyReservation()
DeleteReservatio()
alt
ControlReservationDate&Time
if the current date isn't equal to the
reservation date
or the current date is equal to the
reservation date &
reservation time - current time > 10 minutes
Delete confirmation()
else
Error()

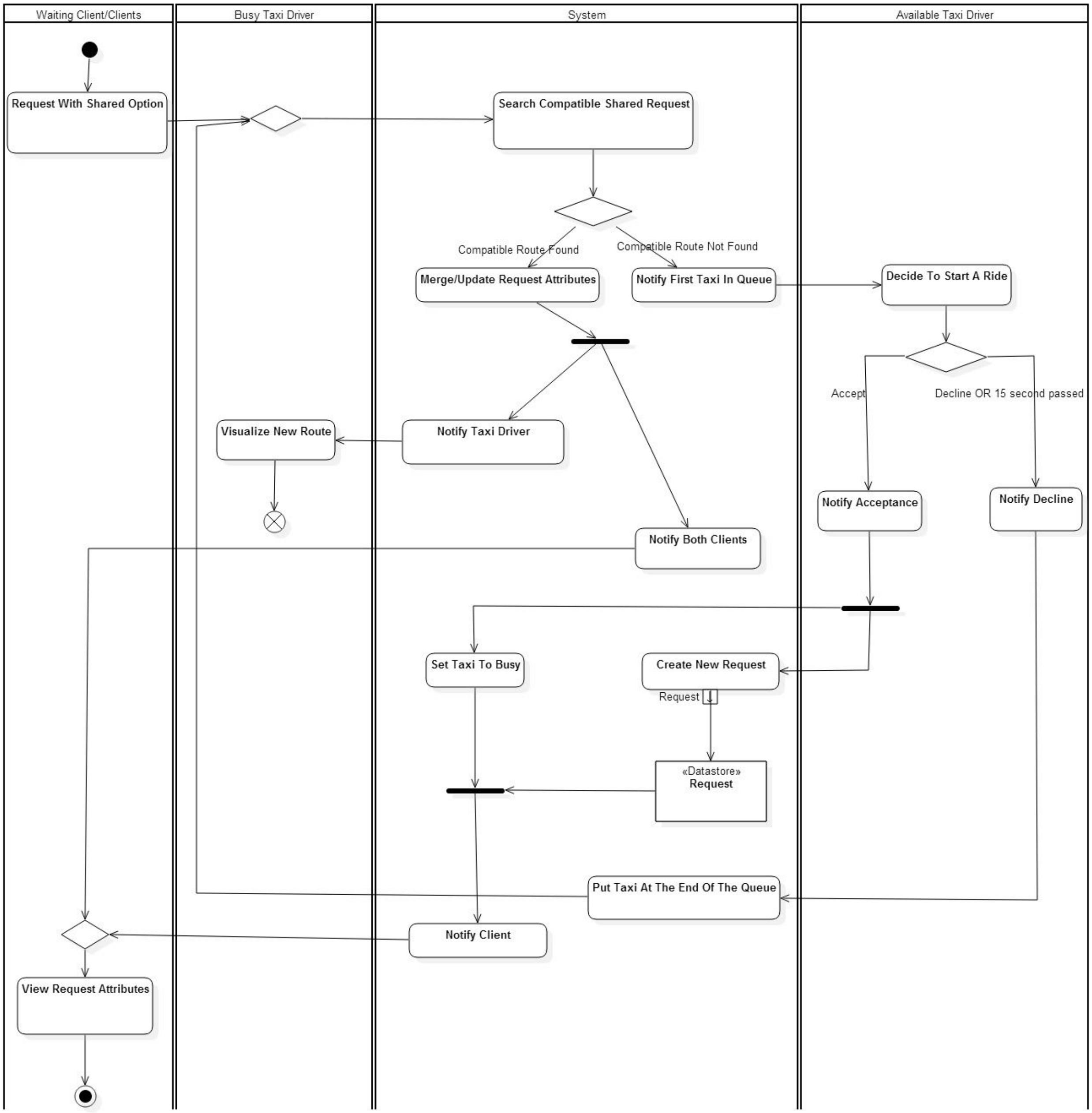
3.5 Requirement Level Class Diagram



3.6 Activity Diagrams

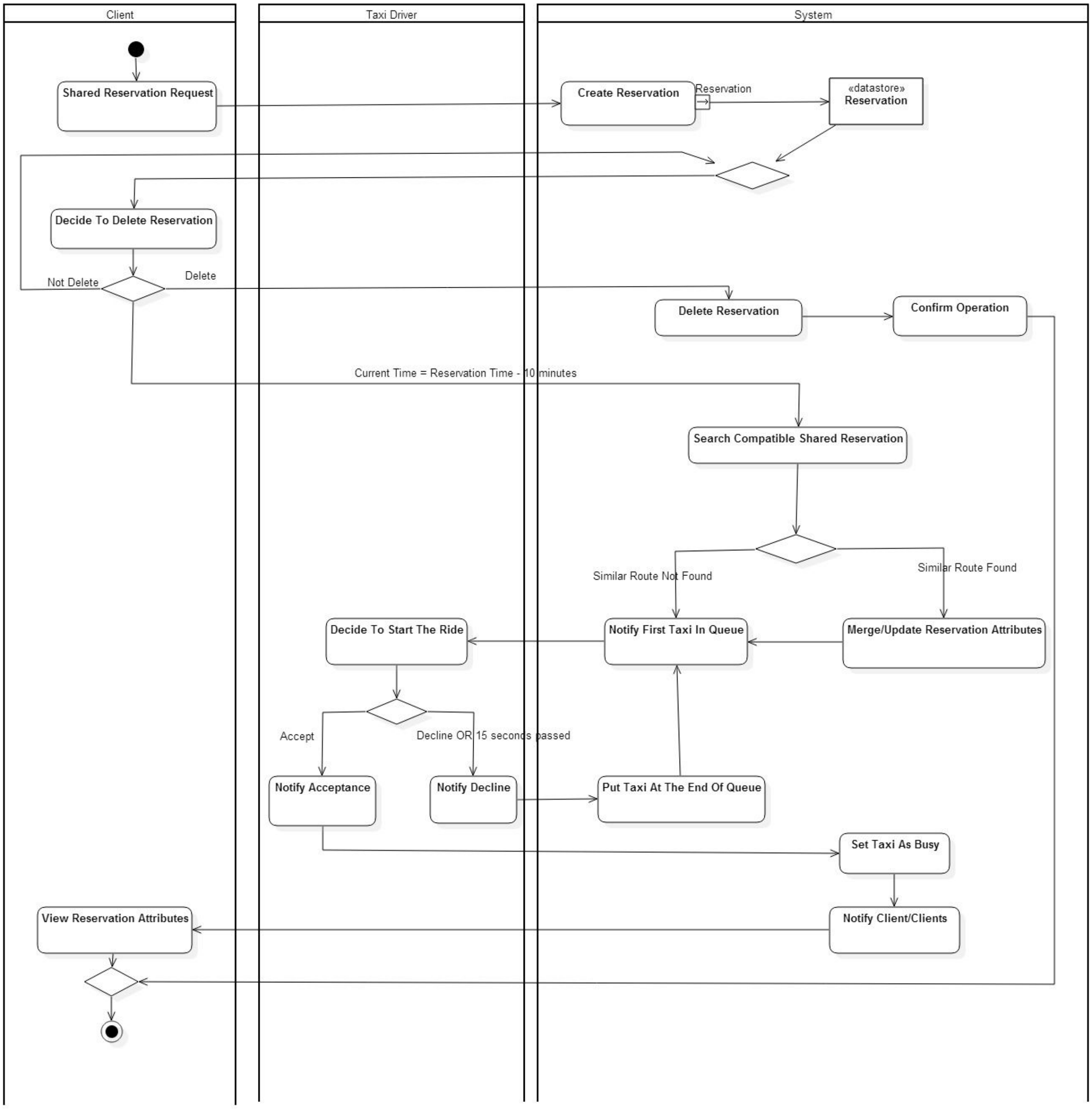
3.6.1 Shared Request

This activity diagram shows the flow of processes that define the shared request use case and the interaction between the system, the client, a generic available taxi driver and a busy taxi driver who is currently doing a similar route to the one requested by the client.



3.6.2 Shared Reservation

This activity diagram shows the flow of processes related to the shared reservation use case and the interactions between the system, the client and a generic available taxi driver, as shown on the swimlane label.



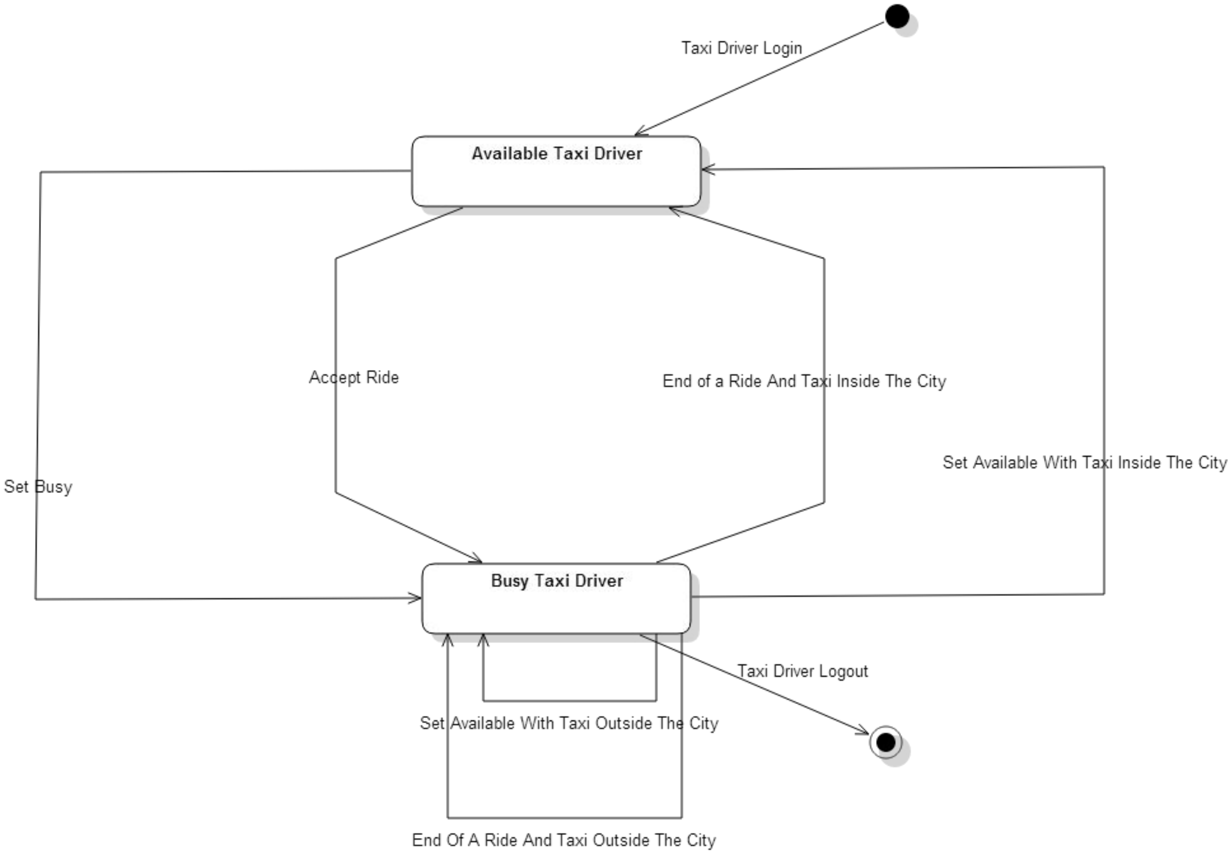
Client
Taxi Driver
System
Shared Reservation Request
Create Reservation
Reservation
«datastore»
Reservation
Decide To Delete Reservation
Not Delete
Delete
Delete Reservation
Confirm Operation
Current Time = Reservation Time - 10 minutes
Search Compatible Shared Reservation
Similar Route Not Found
Similar Route Found
Decide To Start The Ride
Notify First Taxi In Queue
Merge/Update Reservation Attributes
Accept
Decline OR 15 seconds passed
Notify Acceptance
Notify Decline
Put Taxi At The End Of Queue
Set Taxi As Busy
View Reservation Attributes
Notify Client/Clients

Some additional notes about activities:

- Merge/Update Request Attributes (or Reservation): it consists in the merge of all the attributes of the requests/reservations of the two clients involved in the ride at a data level, that are the sum of the number of passengers, the new cost of the ride, the starting place, the destination and the intermediate stops of the unified route.
- Notify Client/Clients: the system will notify two clients if a unified route will be found, otherwise if the system doesn’t find a common compatible route, it will notify the only client who does that route.
- The cycle near “Decide To Delete Reservation” models the fact that the client can decide at every moment to delete one of his reservation, but it can’t be possible anymore after the difference between reservation time and current time becomes less than ten minutes.

3.7 State Machine Diagram

The following statechart diagram shows how and when a taxi driver can change his state from available to busy and conversely, highlighting entry and exit conditions (transitions going out from start state and going into the final state) and exceptions.


Taxi Driver Login
Available Taxi Driver
Accept Ride
End of a Ride And Taxi Inside The City
Set Available With Taxi Inside The City
Set Busy
Busy Taxi Driver
Taxi Driver Logout
Set Available With Taxi Outside The City
End Of A Ride And Taxi Outside The City

3.8 Scenarios

3.8.1 Registration

A new academic year is starting and Federico knows he'll probably need some taxi rides in order to move in time from university to his home.

After having searched online some taxi services, he finds EasyTaxi application.

He downloads it from the store and after having installed it on his mobile phone, he proceeds to register.

He inserts his personal data, such his name, surname, ecc… and then his username and password.

Then he takes a piece of paper and write his user data down because he knows he has a terrible memory and probably he'll forget easily his password.

He clicks on 'Register' button and his mail app pops up: a new mail has been received.

Federico opens it: it's a confirmation e-mail from EasyTaxi service.

He clicks on it and his browser application is automatically opened, it displays the announce his registration has been successfully recognized by the system.

3.8.2 Simple Request

Valeria has just finished her lessons at the University of Milan. This evening she's going to attend to a concert and in order to arrive in time she wants to take a taxi.

Half Moon Run are an awesome band and she has been waiting for months to this special event.

Valeria opens from her laptop Chrome web application and goes to the EasyTaxi website. After having logged in, she clicks on the "Request a taxi now!" button and make her request, choosing to be contacted via SMS. Then, because of her tiring lessons she takes her headphones out of her bag and starts listening to some entertaining music.

Meanwhile Bruno, a taxi driver who has been inactive for a while and has set his system state to 'available' just before, look to his mobile system and sees he has just received Valeria's new request. He accepts it and goes to the address in which Valeria has said she will be located to.

Valeria receives an SMS with the confirmation of the accepted request, indicating the waiting time for the arrival of the taxi, the taxi code and the cost of the ride. He takes her to Circolo Magnolia, she pays him. An exciting evening is waiting for the girl.

3.8.3 Non-Shared Reservation

Davide and Eleonora are really tired. It has been a long, very long evening at Alfred's and all what they intend to do now is coming back home. Their friends left a few hours before and now nobody can give them a lift.

Eleonora is a little bit worried and so she asks Davide what are they going to do.

Davide tells her: <<No problem honey, this afternoon I used my new mobile application easyTaxi and, after having logged in, I booked a taxi ride to this place for 11:00 P.M and it will be here in a few minutes>>

Eleonora replies:<<Thank god, good job Davide>>
Paolo, a taxi driver recently employed has found a lot of traffic and arrives at Alfred’s five minutes later. He apologizes and the couple say that there is no problem.
So Paolo proceeds to carry the couple home.
Once arrived to the destination, Davide pays him and disappears in the night with his girl.

3.8.4 Shared Simple Request

Beniamino has just opened his mobile application to request a taxi ride. He is a very punctual person and all that he wants is to arrive in time to Emanuela’s house.
He has always been a little bit scary of taxis because, when he was child, he used to watch a scary movie called “Crazy Taxi”: however he really can’t avoid this simple solution to go to her girlfriend’s home.
He logs into the application and press on the “Request it now!” button and thinks that this new mobile app is fantastic. He can take taxi with other people and divide the ride cost.
Miriana, a taxi driver whose state is set to ‘Available’ sees the new request on his mobile system, but she is going to finish her working hours and she decides to decline the reservation.
Matteo, an available taxi driver, receives the new shared request and decides to accept it.
Meanwhile Mattia, who has recently graduated, is a little bit late for his working interview. He is really in a hurry!
He logs into the system from the mobile application, turns on his GPS and compiles as fast as possible the request form enabling the shared option.
While Matteo has just started his ride, the system notifies to him that the route is only slightly changed because another passenger has joined the ride and that at the next turn he will have to stop and let Mattia enter in the car with Beniamino. Both Beniamino, Mattia and Matteo are notified by the change of the fees of the ride and the two passengers are very happy because they are going to share the cost ride.
So Matteo will carry Beniamino to Emanuela’s house in perfect time, and will continue the ride with Mattia as indicated by the route visible on the mobile system.

3.8.5 Shared Reservation/Taxi Driver Login

Dave, a taxi driver, tries to log in on the car mobile system at 7 o’clock pm, after having just entered in his car and turned it on.
But Dave hasn’t already taken a coffee, he is still half asleep and he insert the wrong access code.
The system ask to him to refill the login form, and Dave at this point decides to have a coffee.
After a while he makes again the login, this time without errors, and the system shows his home page with the current state set to available.
Unexpectedly, Dave receives immediately a notification by the system: he has to decide to accept the shared request or not in just 15 seconds. It must be a shared reservation because the system shows him instantly the presence of two different clients to be served.

Dave accepts the work and the system send him the route he has to follow to pick up the first passenger.

3.8.6 Delete A Reservation

Yesterday Marco made a booking request for today at 5 o’clock PM.
Today Marco decides to use the bus instead of the EasyTaxi service, so he logs in his personal account by the mobile EasyTaxi’s app, clicks on the “Show My Reservation” button and deletes his reservation.
Luckily, Marco delete his request at 4.45 PM so the system allows him to perform the operation.

3.9 Non Functional Requirements

3.9.1 Performance Requirements

- The system should be reactive and able to manage many requests simultaneously. It is allowed a maximum of 2 seconds of wait between the acceptance of a ride by a taxi driver and the communication to all the interested users.
- Every web page should be visualized in maximum 1.5 second in case of user’s optimal internet connection, and this is mostly important for taxi drivers who have only 15 seconds in total to decide the acceptance of a ride.

3.9.2 Security

- User credentials must be protected in order to allow only the effective owner to access to his account. To prevent intrusions, every time a visitor wants to access the service he’ll be asked his username and his password, while taxi drivers login is made more secure by the compulsory insertion of their personal access code, in addition to their username and password.
- Basic security checks about SQL injection will be performed.

3.9.3 Availability

The service should be available every day of the year, and it will be unavailable only for a quarter of hour once a week on late night for maintenance purpose and data backup.

3.9.4 Usability

- The maximum number of clicks in order to use any functionality of the system won’t be superior to 4, as could be seen on the mockup section of this document.
- The usability of application increases with the possibility of using a GPS integration, but a client can achieve the same goals without it. With the GPS set to active, a client can avoid to specify manually the address of his position to the system during the compilation of generic requests, letting the GPS to do this automatically.

- The user interface should be as intuitive as possible in order to facilitate the using of the service even for the people less predisposed to technology.

3.9.5 Portability

- The mobile application must be compatible both with IOS, Android and Windows Mobile.
- The internet website must be compatible with the following internet browsers: Internet Explorer, Opera, Microsoft Edge, Mozilla Firefox, Safari, Google Chrome.

3.9.6 Scalability

The system should adapt well to an unexpected growth of user requests that could be possible in case of big events in the city or during periods of massive tourism. So the system should support at least a number of simultaneous requests equal to two times the number of taxi drivers registered to the system.

3.9.7 Reliability

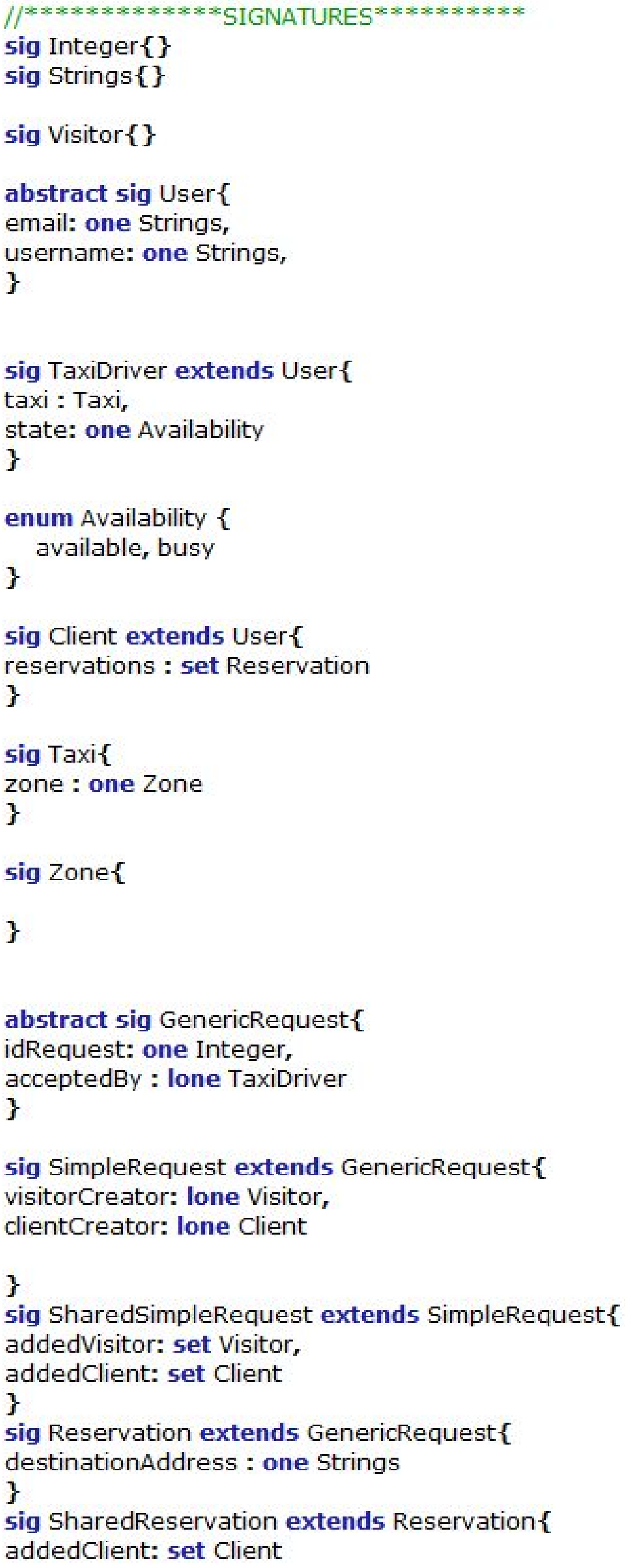
Data should be replicated in order to avoid data losses in case of system faults. Those operations are done during the period when the system stops to be active once a week.

4: Alloy

4.1 Alloy

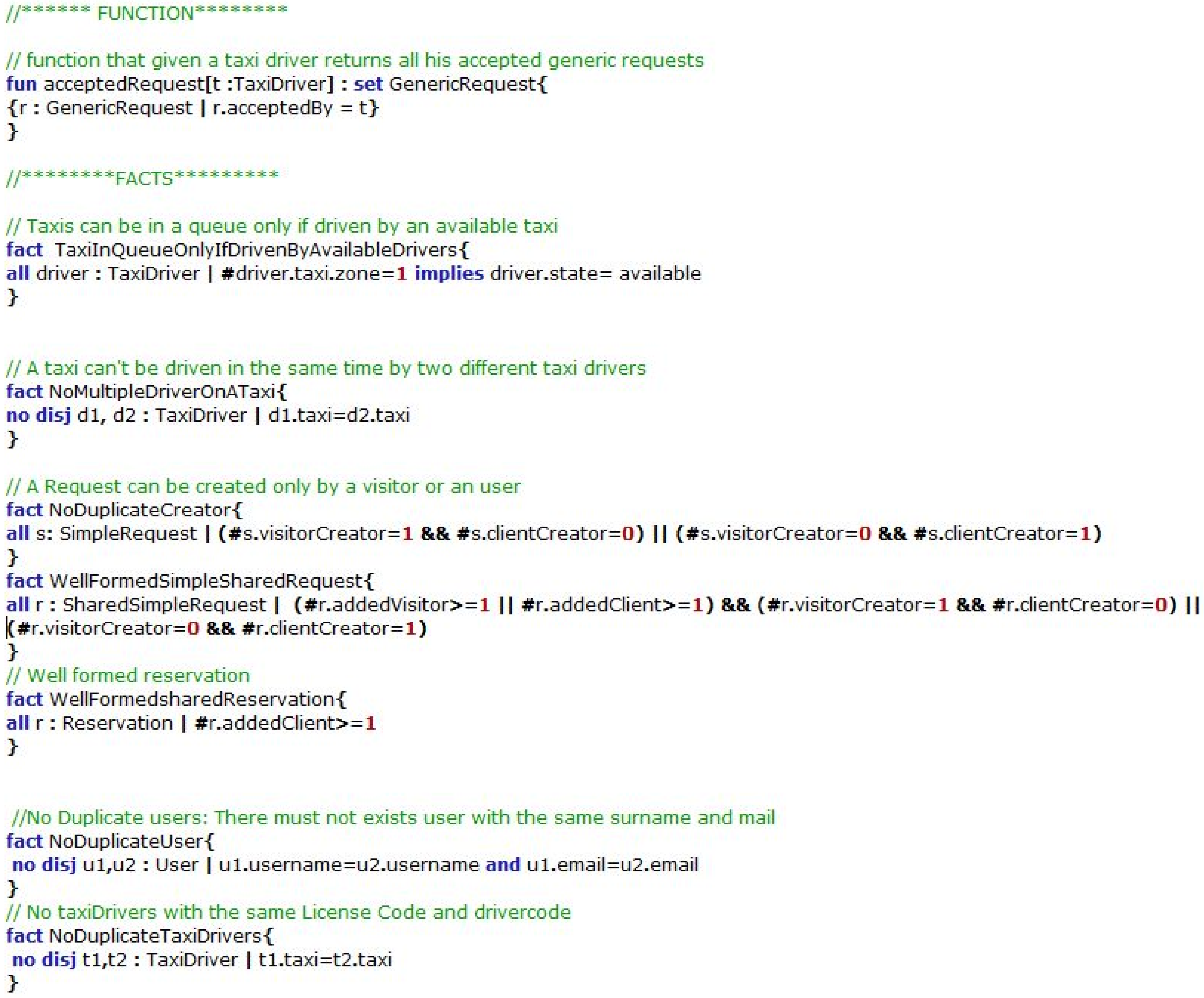
The following alloy model presented is created using the class diagram. We try to divide the code in part dividing signature from fact, assert and predicates. In the last part there is the generated world.

4.1.1 Entity Implementation and Signature



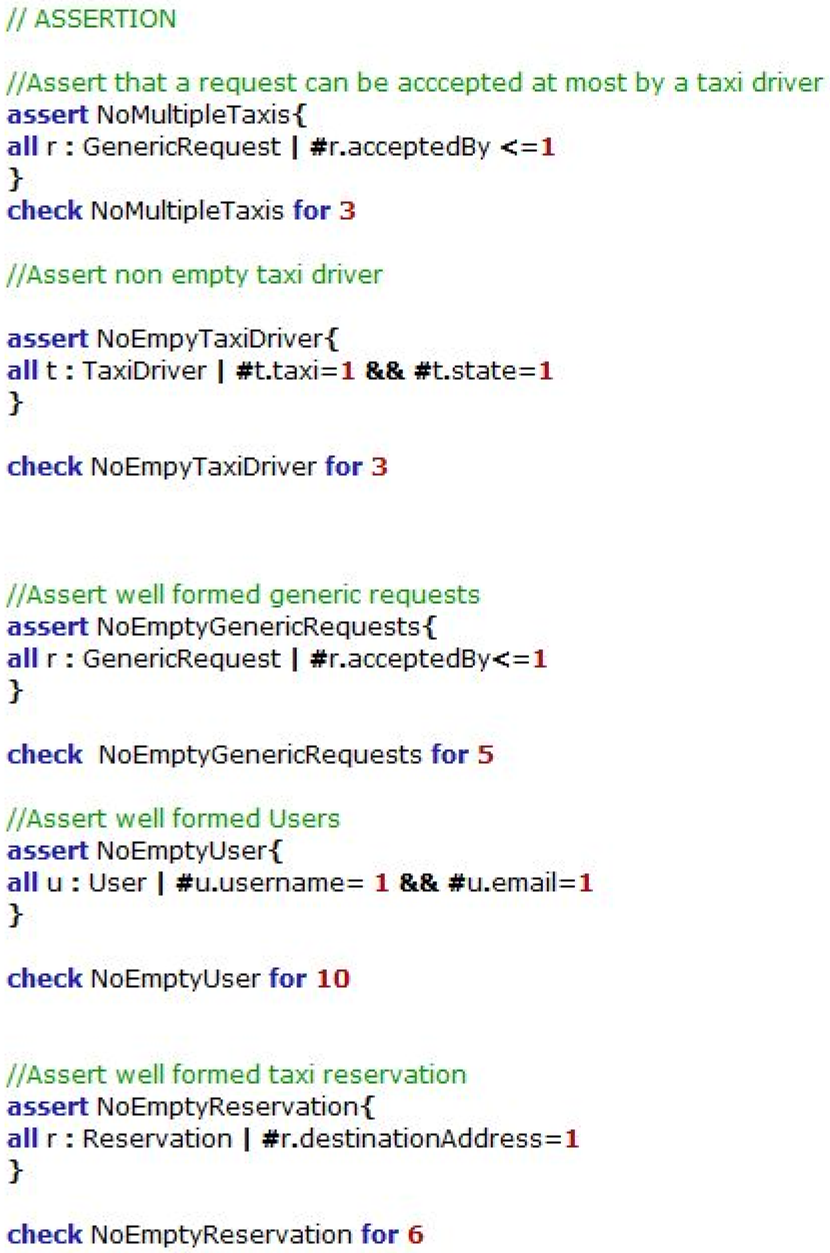
```
//**************SIGNATURES***********
sig Integer{}
sig Strings{}

sig Visitor{}

abstract sig User{
email: one Strings,
username: one Strings,
}


sig TaxiDriver extends User{
taxi : Taxi,
state: one Availability
}

enum Availability {
   available, busy
}

sig Client extends User{
reservations : set Reservation
}

sig Taxi{
zone : one Zone
}

sig Zone{

}


abstract sig GenericRequest{
idRequest: one Integer,
acceptedBy : lone TaxiDriver
}

sig SimpleRequest extends GenericRequest{
visitorCreator: lone Visitor,
clientCreator: lone Client

}
sig SharedSimpleRequest extends SimpleRequest{
addedVisitor: set Visitor,
addedClient: set Client
}
sig Reservation extends GenericRequest{
destinationAddress : one Strings
}
sig SharedReservation extends Reservation{
addedClient: set Client
```

4.1.2 Fact



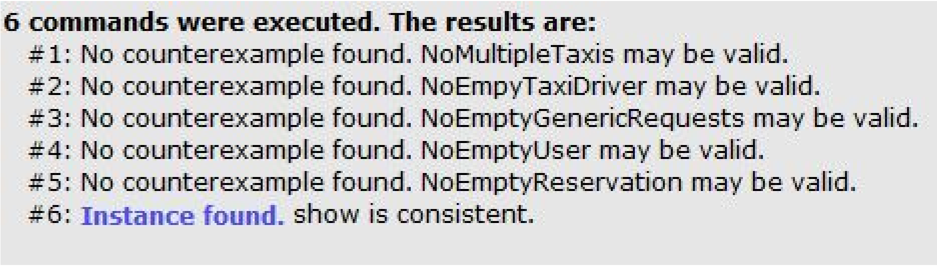
```
//****** FUNCTION********

// function that given a taxi driver returns all his accepted generic requests
fun acceptedRequest[t :TaxiDriver] : set GenericRequest{
{r : GenericRequest | r.acceptedBy = t}
}

//********FACTS*********

// Taxis can be in a queue only if driven by an available taxi
fact  TaxiInQueueOnlyIfDrivenByAvailableDrivers{
all driver : TaxiDriver | #driver.taxi.zone=1 implies driver.state= available
}


// A taxi can't be driven in the same time by two different taxi drivers
fact NoMultipleDriverOnATaxi{
no disj d1, d2 : TaxiDriver | d1.taxi=d2.taxi
}

// A Request can be created only by a visitor or an user
fact NoDuplicateCreator{
all s: SimpleRequest | (#s.visitorCreator=1 && #s.clientCreator=0) || (#s.visitorCreator=0 && #s.clientCreator=1)
}
fact WellFormedSimpleSharedRequest{
all r : SharedSimpleRequest |  (#r.addedVisitor>=1 || #r.addedClient>=1) && (#r.visitorCreator=1 && #r.clientCreator=0) ||
(#r.visitorCreator=0 && #r.clientCreator=1)
}
// Well formed reservation
fact WellFormedsharedReservation{
all r : Reservation | #r.addedClient>=1
}


 //No Duplicate users: There must not exists user with the same surname and mail
fact NoDuplicateUser{
 no disj u1,u2 : User | u1.username=u2.username and u1.email=u2.email
}
// No taxiDrivers with the same License Code and drivercode
fact NoDuplicateTaxiDrivers{
 no disj t1,t2 : TaxiDriver | t1.taxi=t2.taxi
}
```

4.1.3 Assert



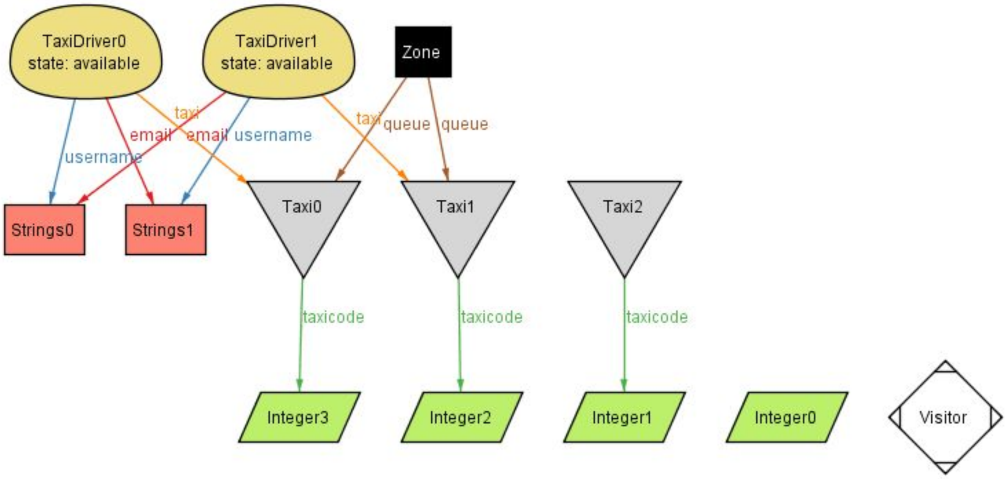
```
// ASSERTION

//Assert that a request can be acccepted at most by a taxi driver
assert NoMultipleTaxis{
all r : GenericRequest | #r.acceptedBy <=1
}
check NoMultipleTaxis for 3

//Assert non empty taxi driver

assert NoEmpyTaxiDriver{
all t : TaxiDriver | #t.taxi=1 && #t.state=1
}

check NoEmpyTaxiDriver for 3



//Assert well formed generic requests
assert NoEmptyGenericRequests{
all r : GenericRequest | #r.acceptedBy<=1
}

check  NoEmptyGenericRequests for 5

//Assert well formed Users
assert NoEmptyUser{
all u : User | #u.username= 1 && #u.email=1
}

check NoEmptyUser for 10


//Assert well formed taxi reservation
assert NoEmptyReservation{
all r : Reservation | #r.destinationAddress=1
}

check NoEmptyReservation for 6
```

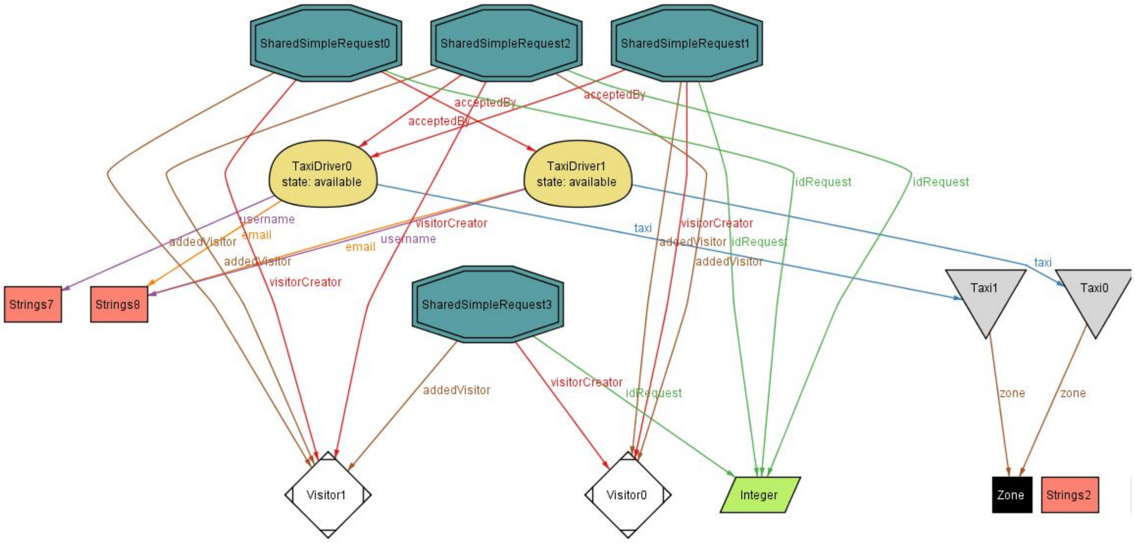
4.1.4 Result



```
6 commands were executed. The results are:
  #1: No counterexample found. NoMultipleTaxis may be valid.
  #2: No counterexample found. NoEmpyTaxiDriver may be valid.
  #3: No counterexample found. NoEmptyGenericRequests may be valid.
  #4: No counterexample found. NoEmptyUser may be valid.
  #5: No counterexample found. NoEmptyReservation may be valid.
  #6: Instance found. show is consistent.
```

4.1.5 Generated worlds





4.2 Software and tool used

- StarUml for the activity diagrams and the state chart diagrams
- Draw.io for the sequence diagrams, use case diagrams and the class diagram
- Balsamiq Mockups 3 for mockups

4.3 Hours of works

Alessandro Dell'Orto: 30 h
Andrea Brunato: 32 h
Lorenzo Costantini: 30 h