Politecnico di Milano

A.A. 2015-2016

Software Engineering 2: “MyTaxyService”

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

version 1.0

Matteo Martinelli (mat. 840538)

Simone Biffi (mat. 853393)

Pietro Astolfi (mat. 841044)

TABLE OF CONTENTS

1. Introduction

1.1 Description of the given problem

The aim of this project, myTaxiService, is to create a web and mobile application finalized to simplifying the access of passengers to the taxi service and, at the same time, guarantee a fair management of taxi queues.
In order to use the service, the passengers must sign up on the web or mobile application, while the taxi driver are pre-registered by the government who adopt this new cutting-edge system.
Once the user is registered, he should be able to make a request for a taxi or reserve a future ride. Furthermore, there should exist an additional feature which allows all the users to share a ride with others whose are willing to start a shared ride from the same zone going in the same direction.

While, the taxi driver must be able to inform the system about his availability and to accept or reject requests coming from passengers.

1.2 Actual system

The software house wants to create a new taxi service. We suppose that until now nothing has been created and we have to “develop” the entire application without using or modifying a previous system.

1.3 Goals

We thought about the possible users and what myTaxiService should provide them, so we planned that our system will give the following opportunities:

- [G1] Registration of a person.
- [G2] Request a taxi ride.
- [G3] Reserve a taxi ride.
- [G4] Create a shared ride.

- [G5] Join to a shared ride.
- [G6] Manage a taxi ride.
- [G7] Accept, or not, a passenger's request.
- [G8] Update the taxi driver availability.
- [G9] Taxi driver can set a ride as “Completed”

The system automatically should also perform these functions:

- [G10] Informing the passenger about taxi details.
- [G11] It ensures a fair management of taxi queues.

1.4 Domain properties and assumptions

We suppose that the following properties and assumptions hold in the analysed domain.

- If a passenger asks for a taxi, he gives the right information.
- A passenger requests a taxi only if he can take it
- If a passenger requests/reserves a taxi, he must waits in the specified location during the request/reservation.
- Taxi are pre-registered by the government and the credentials are delivered via post office or email.
- Once a taxi driver updates his availability status in a specific zone, he can't move to another one. (Example: if the availability signal has been sent in Zone A, the taxi driver must stay in Zone A)
- Taxi driver GPS are always on-line while the taxi driver is in “available” state.
- If a taxi driver accepts a request, he will complete it.
- Taxi drivers don't put themselves as available if they are giving a ride

- Taxi drivers set that a course has been completed only if it really is.
- The taxi driver always reaches the meeting point at most in 10 minutes for a reservation/shared ride.
- Every taxi driver owns a smartphone (used to run the MyTaxiService mobile application)
- A shared ride starts from one location and can reaches different location along the entire ride
- Queues have always at least one taxi available
- The city is divided in zone with fixed dimension ($2km^2$) (set up in the installation phase)

1.5 Definitions, acronyms and abbreviation

1.5.1 Definition

- **Request**: A request is an action performed by the passenger that allow him to ask for a taxi immediately
- **Reservation**: A reservation is an action performed by the passenger that allow him to reserve a taxi ride
- **User**: Who use the application, may be passengers or taxi drivers
- **Shared ride**: A taxi ride shared by different registered person
- **Available**: A taxi driver is available when it starts working and then, it can be selected from the system to complete a ride
- **System**: the entire software system to be developed

1.5.2 Acronyms

- **OS**: Operative system
- **UI**: User interface
- **UXD**: User experience design

- **UML**: Unified Modelling Language
- **DD**: Design document
- **RASD**: Requirements Analysis and Specification Document
- **API**: Application Programming Interface.

1.5.3 Abbreviation

- **KISS**: Keep It Simply and Stupid (Keep it simply smart)

1.6 Identifying stakeholders

Our main stakeholder is the professor who gave us this didactic project. The professor asks us to focus on the whole development process of a complex enterprise application, which involves the following phases: requirement analysis, design, testing and project reporting. The implementation phase has been outsourced and we do not have to focus on it, but actually, we think that the main goal is to show that we have understood the development process in all its parts and that we can carry out a project from the beginning to the end.

We will focus first on the main functionalities of the system, which are directly requested in the specification given to us by the professor, then we will try to extend the functionalities of the system in order to make the application as close as possible to an application that is ready to be launched in the market.

Starting from this consideration we can imagine that an hypothetical stakeholder for our system could be a government that wants to automate the life cycle of a taxi request making it easily accessible through modern internet services.

1.7 Future possible implementation

Some future implementation we have spotted out during the RASD phase are:

- Since the current location and destination location are requested, the system can automatically compute the relative cost and make possible a payment through the web/mobile application
- Another important feature should be that the passenger (and taxi driver) can provide a feedback (positive or negative) to each other. A feedback system can avoid, for example, fake requests (banning the passengers from the system)
- The history of all the rides could be kept stored in the database to provide to the user (for both passengers and taxi) statistics about mileage, hours spent in a taxi and how much money has been spent (or earned) so far

2. Actors identifying

- **Guest**: the guest is a person that is not already registered. He can only see the login page and optionally fulfils the registration form.
- **Passenger**: the passenger is a person registered to the system willing to utilize services provided by myTaxiService. He has the possibility to access his profile and then request, reserve or share a taxi.
- **Taxi driver**: the taxi driver is a person auto-registered by the government. He has the possibility to access his profile, the responsibility to give his availability and to accept, or not, requests from the passengers.

3. Requirements

We have determined the following requirements according to our analysis, so assuming that the domain properties, which are described above, holds for our purposes, we have written the following requirements in order to satisfy goals.

3.1 Goal requirements

- Registration of a person:
  - The system must provide an intuitive sign up functionality both on the web and mobile application.
- Request a taxi ride:
  - The system gives to the passengers the possibility to require a taxi through web or mobile application.
  - The system will allow the passenger to confirm, or not, a request submitted before.
- Reserve a taxi ride:
  - The system has to provide a function that allow passengers to reserve a taxi for the preferred date and time.
  - The system has to provide a function able to show the state of the reservations done by the passenger.
- Create a shared ride:
  - The system will allow the passenger to submit a “reservation with sharing”.
- Join to a shared ride.
  - The system will allow the passenger to join to an existent shared ride.
- Manage a taxi ride:
  - The system has to provide a function which allows the passenger

to monitor all pending rides and delete reservations and shared rides (disjoin).

- Accept, or not, a passenger's request:
    - The system will allow taxi drivers to accept or reject requests coming from passengers.
- Update the taxi driver availability:
    - The system has to provide a function that allow taxi drivers to notify the system when they change their availability status
- Taxi driver can set a ride as “Completed”
    - The taxi driver, once he finished a ride, it has to notify the system that a particular ride has been completed.
- Informing the passenger about taxi details:
    - The system has to provide a function that allow the applicant passenger to be informed about the code of the incoming taxi and the waiting time.
- Ensure a fair management of taxi queue:
    - The system has to provide a management of taxi queue that respect an adapted FIFO paradigm (The assigned taxi driver is the first one in the queue which meet the passenger's requirement).

3.2 Functional requirements

- **Guest**: he/she can
    - Sign up [G1]
- **Passenger**: he/she can
    - Log in through web or mobile application
    - Make a request for a taxi [G2]
    - Make a reservation for a drive [G3]

    - Create a shared ride [G4]
    - Delete a reservation accordingly to the imposed limits [G6]
    - Receive a confirmation question for a requested/reserved drive
    - Access to the “Share a ride” section
    - (Dis)join a ride from the “Share a ride” section [G5] [G6]
    - Monitor all his/her pending ride [G6]
- **Taxi driver**: he/she can
    - Log in through mobile application
    - Modify his availability status (available or not) [G8]
    - Accept or refuse the received ride requests [G7]
    - Notify the completion of a ride [G9]

3.3 Non functional requirements

We can divide the non-functional requirements into three groups:

SECURITY ISSUES

- Security
    - All the data inserted in database must be protected
        - Password must not be stored in plaintext but encrypted
    - The communication among user and system must be encrypted in order to protect every kind of transaction

CHARACTERISTIC OF PHYSICAL SYSTEM

- Performance: must be acceptable to guarantee a good grade of usability, in particular:
    - Every kind of API request must be completed within 1 second.

    - Once a user make a taxi request, he has to receive a response from a taxi driver in no more than 2 minutes
- Portability
    - The mobile application must be available for every platform (Android, iOS, Windows Phone)
    - The web application must be compatible with the most popular browser (Google chrome, Mozilla Firefox etc..)
- Availability
    - The system is running 24/7 so that a user can always requests a taxi.
    - Any kind of updates must not stop the normal operation
- Reliability
    - System shall have an availability of 99.99% (four nines) which imply a 52.56 minutes downtime per year
- Scalability
    - In case of overloading the system must be able to scale:
        - Horizontal scalability: in case the requests generate are not manageable, a load balancers is automatically configured and activated to distribute the application load
- Capacity
    - System should be able to manage 10000 requests/second
    - The system must be able to manage at least 5000 connected passengers and 1000 taxi driver at a given time
- Maintainability
    - The whole application code will be documented to well inform future developers of how application works and how it has been developed.

- Accuracy
    - The error deriving from the estimated arrival time computed by the system and the effective time must be less than 5 minutes.
    - The system automatically update the arrival time using the GPS location sent by the taxi driver.
- Accessibility
    - MyTaxiService can be easily used by every person who own a mobile phone with Android, iOS or Windows phone OS. For the passengers is also available a web application to make request or reserve/share a ride.
    - Both the mobile application and the web application can be used by any kind of user since the UI/UXD is very simple and intuitive (KISS paradigm must be adopted)
    - At the first login in the web application or in the mobile application, will be presented a simple and quick wizard tour to improve the ability of the users

HOW THE SYSTEM WORK IN UNEXPECTED/FAULT CONDITIONS

- Robustness
    - In case of failure the system must restart after 30 minutes

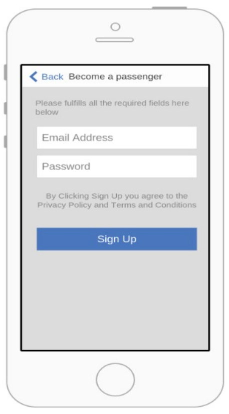
3.3.1 User interface

Here below there are some sketches made to clarify and let the reader creates a mental draw about how the application will be. Of course, we do not present all the possible user interface but only the most important. We want to emphasize that the web application will have the same functionalities and features offered by the mobile application (for the passengers only) and this entails that the user interface are quite similar from one to the other.

Log in and sign up

As specified during the documents, the only one who can register is the passenger while, both the taxi driver and the passengers can log in.

Both of them has provide email address and password and only the taxi driver must activate the switch “Taxi driver?”


Back Become a passenger

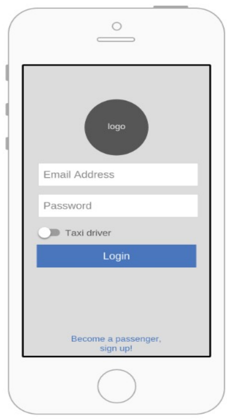
Please fulfills all the required fields here below

Email Address

Password

By Clicking Sign Up you agree to the Privacy Policy and Terms and Conditions

Sign Up


logo

Email Address

Password

Taxi driver

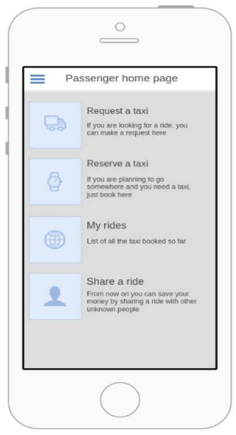
Login

Become a passenger, sign up!

Homepage after login (passenger and taxi driver)

The passenger (left figure) can perform exactly three operation and watch the list of all the rides booked, shared or requested.

On the other hand, a taxi driver, after being logged in, can set himself as available such that the system can assign to him a request.


Passenger home page

Request a taxi

If you are looking for a ride, you can make a request here

Reserve a taxi

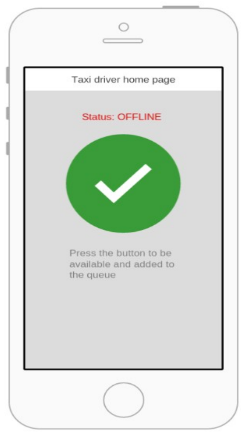
If you are planning to go somewhere and you need a taxi, just book here

My rides

List of all the taxi booked so far

Share a ride

From now on you can save your money by sharing a ride with other unknown people

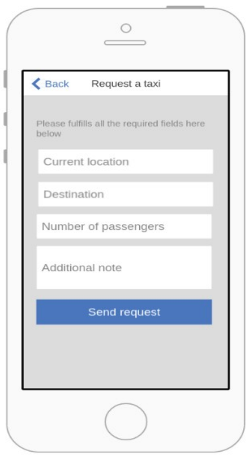

Taxi driver home page

Status: OFFLINE

Press the button to be available and added to the queue

Passenger makes a request/reservation/shared ride

The user from the menu can choose among these three options. Every screen shown by the application requires to the user to fulfils all the fields in order to having a successful submission.


Back Request a taxi

Please fulfills all the required fields here below

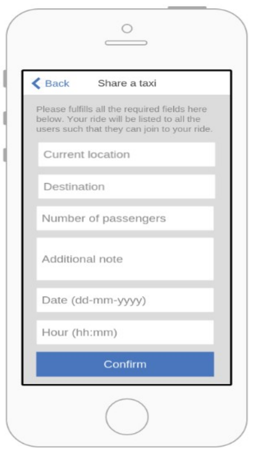
Current location

Destination

Number of passengers

Additional note

Send request


Back Share a taxi

Please fulfills all the required fields here below. Your ride will be listed to all the users such that they can join to your ride.

Current location

Destination

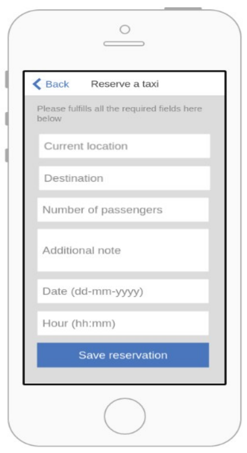
Number of passengers

Additional note

Date (dd-mm-yyyy)

Hour (hh:mm)

Confirm


Back Reserve a taxi

Please fulfills all the required fields here below

Current location

Destination

Number of passengers

Additional note

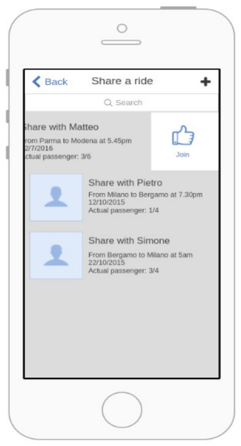
Date (dd-mm-yyyy)

Hour (hh:mm)

Save reservation

Passenger joins/disjoin to/from a shared ride

A passenger can join to a ride created by someone else, or, after being successful joined to one shared ride, remove himself from it.


Back Share a ride +

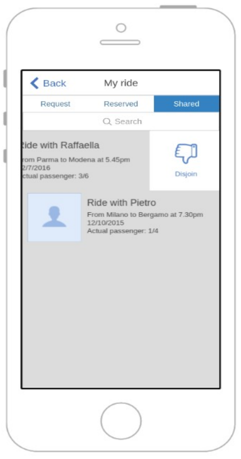
Search

hare with Matteo
rom Parma to Modena at 5.45pm
2/7/2016
ctual passenger: 3/6

Join

Share with Pietro
From Milano to Bergamo at 7.30pm
12/10/2015
Actual passenger: 1/4

Share with Simone
From Bergamo to Milano at 5am
22/10/2015
Actual passenger: 3/4


Back My ride

Request Reserved Shared

Search

ide with Raffaella
rom Parma to Modena at 5.45pm
2/7/2016
ctual passenger: 3/6

Disjoin

Ride with Pietro
From Milano to Bergamo at 7.30pm
12/10/2015
Actual passenger: 1/4

3.3.2 Documentation

We will draft these documents to well-organize our work in the way to be as much efficient as possible:

- **RASD**, Requirement Analysis and Specification Document, to well-understand the given problem and to analyse in a detailed way which are our goals and assumptions. It also contains an overall description of the system (using scenarios and use-cases) and the models describing requirements and specifications.

- **DD**, Design Document, which contains a functional description of the system using models such as Entity-Relationship and UML diagrams.
- **Installation manual:** which explains how to deploy the web-application, mobile application and the database
- **User manual**: A simple guide which explain to the users how to use the main functionalities of the application.
- **Testing report**, which contains the results of the testing activity performed on a system developed by another group.

3.4 Constraints

- The mobile application have to be as light as possible, maximum size is fixed around 10MB;
- The web application has to be compatible with most of the browsers and devices (mobile phone, tablet, PC);
- The chosen database have to be distributed by Oracle;
- The functions using maps or more in general GPS geolocation have to be based on Google services.

4. Specifications

We write down here some specification that should make clearer how we can reach the goal explained above:

- Every time a request or reservation has been completed, the system will remove it from the database
- Taxi drivers have to confirm within 2 minutes otherwise the system will automatically reject.
- Once the system notifies to the passenger, he has to confirm within 2 minutes otherwise the system will automatically delete the request.
- The system must not allow a user to have more than 1 request active
- When a taxi driver set himself as available, the system push him into the queue of his current zone according to the FIFO policy
- When a taxi driver is picked up from the queue, it must confirm:
    - If it accepts, he is removed from the queue (Unavailable status automatically set)
    - Otherwise, if he rejects, he is moved to the tail of the queue.
- When a user rejects a request (i.e. because the waiting time doesn't meet his requirements), the system notifies the taxi driver and places him in the queue at the same position of before.
- The system allows passenger to sign up with a specific email address at most one time
- When a passenger disjoins from a shared ride, the system will simply delete the destination of that passenger
- 10 minutes before a reserved or shared ride occurs, the system assigns them to a taxi driver and notifies it to the related passengers (more than one in case of shared ride)
- System must not allow a passenger to create a reservation which will occur within 2 hours.

- System allows any passengers to join/disjoin themselves from a shared ride until 15minutes before the starting time.
- System allows any passengers to delete a reservation until 15minutes before the ride will occur.
- The location of the taxi is automatically calculated through the GPS location

5. Scenarios

Here there are some possible scenarios:

- Marc, a resident of Old York, doesn't have a car and he needs to go from his apartment to the one of his friend Frank when the public transport aren't available. So, surfing on Internet looking for a phone number in order to book a taxi comes across MyTaxiService
  On the homepage there is a clear explanation of the functionalities of the service and he likes so decides to sign up.
  Marc fulfills all the requested fields in the registration form and, after the system confirms him the successful registration, he is able to see the home page.
  Afterward, he decide to perform a request for a taxi entering his current location and the destination point that allow Marc to reach Frank.
  The application notifies Mark about the succeed of the operation and it also provides the waiting time and the taxi code. Everything meets the Marc's requirements and he confirms the ride.
- After having utilized the web application, Marc reminds himself that Frank has only one bedroom and he cannot accommodate him for the night. Satisfied by the service offered by MyTaxyService, he decides to download the mobile application and log into it.
  At the first glance, he notices the button "Reserve a taxi" who will allow him to book a taxi ride. The application will show automatically a form which must be filled by Marc. He enters the departure time, the departure location, the destination and finally he sends the request. A notification alerts Marc that the reservation succeed and 10 minutes before the drive the system will assign him a taxi.
  After 2 hours while Marc and Frank were watching a movie, Frank falls asleep. The Frank's mother offers a ride back home to Marc, but he has already booked a taxi. Looking at the rules, he reads that the lower bound to cancel a reservation is 15 minutes before the departure time that, as booked, would arrive in an hour.

In conclusion, he takes his mobile phone and he goes into the application in the “My ride” section which is dedicated to show a list of request/reserved and shared ride. Through the appropriate button he cancels the reservation, and the system informs him about the successful annulment

- John, the taxi driver, received a notification from the government of Old York which specify that a new taxi system has been introduced. Moreover, in the announcement he can find a brief but clear explanation about how the new system works and the credential to log into the application.
  Obviously, John gets curious and he immediately decide to download the mobile application and sign into to exploit all the features which myTaxiService is made up.
  As per usual, John starts his night shift waiting a customer call when he realized to have downloaded the MyTaxiService application. He goes into it, and once reached the home page he sets his status as available. After just a couple of minutes the application inform him about a new ride far away from his current location but, anyway, he accepts the request and he waits the confirmation from the system.
  On the other side, the passenger receives the notification with the waiting time which doesn't completely satisfy him so the passenger decide to halt the request.
  Of course, the taxi driver receives that the process has been stopped and he does not start the ride.
  By the way, the taxi driver is still available and able to manage a new request which comes up after few minutes. The new request turned out well and he drives directly to the passenger's location specified by the application.
- Anne, Marion e George are three expert researcher invited to a conference in Old York. Everybody arrives from different city and they didn't meet each other so far.
  Knowing the end time of the conference, George decides to reserve a taxi but, in order to save as much money as possible, he chooses to use the sharing option.

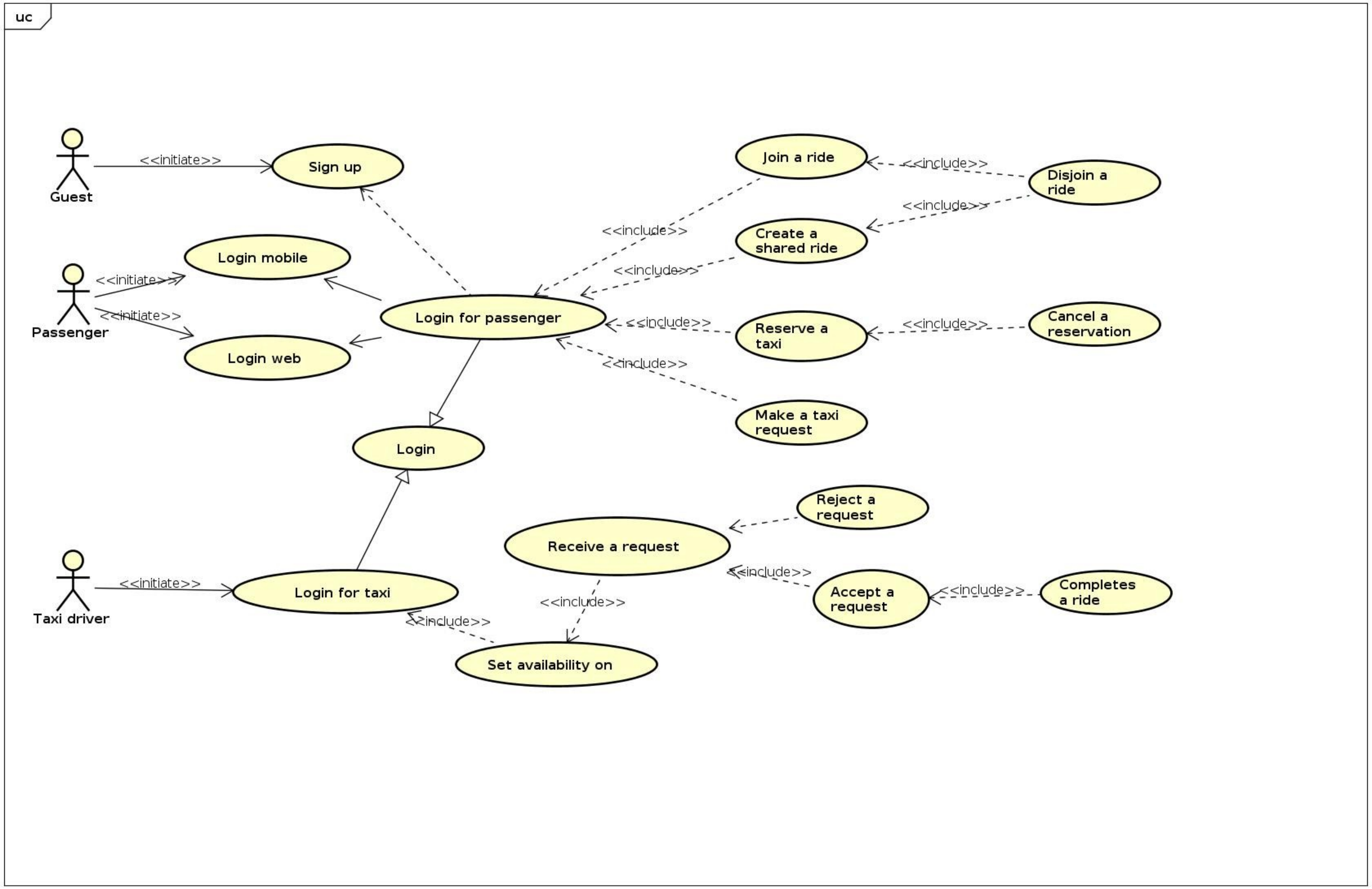
By clicking on the menu in the “Share a ride” section, he is able to filter through all the inserted rides but, since there is no rides that satisfy what he was looking for, he creates a new one fulfilling all the required information hoping someone else will need the same drive. Pushed on the button “Confirm”, the system notify him that the ride has been inserted properly.
Anna and Marion land later than George, and like him, they think how to come back to the hotel after the end of the conference. As described in the website of the conference, every participants would accommodate in the same hotel, therefore, Both Anna and Marion take the own mobile phone and they start looking for a shared ride using MyTaxiService.
Both of them, after having correctly filtered all the available drives, find the George's ride and either Marion and Anna joins to the shared ride by clicking to the appropriate button “Join”.
At noon, Anna meets casually a closer friend which invite her to an happy hour after the congress. Knowing that it's 1.35pm and the ride will occur at 19pm, Anna got plenty of time to disjoin from the shared ride. She goes to the “My ride” section and selecting the “shared ride”, she can easily remove herself from the ride by pressing the button “disjoin”. The system notifies her about the successful operation while George and Marion are still sharing the ride.

6. UML Models

6.1 Use Case Diagram

We can derive from the scenarios identified in the previous paragraph:

- Passenger signs up in web application
- Passenger logs in in web application
- Passenger signs up in mobile application
- Passenger signs in in mobile application
- Taxi driver logs in
- Passenger makes a taxi request
- Taxi driver accepts/rejects a request from a passenger
- Taxi driver sets his availability
- Taxi driver completes a ride
- Passenger sends a reservation request
- Passenger cancels a reservation request
- Passenger creates a shared ride
- Passenger joins to a shared ride
- Passenger removes himself from a shared ride


uc
Guest
Passenger
Taxi driver
<<initiate>>
<<initiate>>
<<initiate>>
<<initiate>>
Sign up
Login mobile
Login web
Login for passenger
Login
Login for taxi
Set availability on
Receive a request
Reject a request
Accept a request
Completes a ride
Join a ride
Disjoin a ride
Create a shared ride
Reserve a taxi
Cancel a reservation
Make a taxi request
<<include>>

6.2 Use Case Description

Here below, we describe in a detailed way the main use cases. Some of them may be omitted because less relevant or mostly similar to other already described.
All the references related to “button” or every other kind of input forms are only hypothesis in order to make the situation clearer and to help the reader to draw a visual picture in his mind.

Name	**Passenger signs up in web application**
Actors	Guest
Entry Conditions	The guest doesn't have any credentials to perform a log in.
Flow of events	• The guest open the menu in the web page and click on the “Sign up” link. • The system shows a page with the required fields: email address and password • The guest fulfills the input form and press the button “Send registration” • The system checks, confirms and redirect the passenger to the home page.
Exit conditions	The submitted informations are stored in the database and the guest is now able to log in.
Exceptions	If the guest types a wrong password or he doesn't fill the mandatory fields, an error will be shown.

Name	**Passenger logs in the web application**
Actors	Passenger
Entry	The passenger has already the credentials, this imply the

Conditions	registration phases has been already performed (either via web or mobile application)
Flow of events	• The passenger open the menu in the web page and click on the “Log in” link. • The system shows all the input required for a successful login (Email address and password) • The passenger fills all of them and press the “Log in” button • The system redirect the passenger in the personal page
Exit conditions	The passenger is now logged into the application
Exceptions	The inserted informations are wrong and an error will be shown.

Name	**Taxi driver logs in**
Actors	Taxi driver
Entry Conditions	The taxi driver has the credentials provided by the MyTaxiService office and it turns on the smartphone.
Flow of events	• The taxi driver press on the “Log in” link • The system shows all the input required for a successful login (Email, password) • The taxi driver fills all of them, he activates the “taxi driver” switch and finally press the “Log in” button • The system redirect the taxi driver in the personal page
Exit conditions	The taxi driver is logged into the application and it can perform all the available actions

Exceptions	The inserted credentials are wrong and an error will be shown.

Name	**Taxi driver completes a ride**
Actors	Taxi driver
Entry Conditions	The taxi driver is running a ride and he reaches the destination point (last destination in case of shared ride).
Flow of events	• The taxi driver press on the “Ride completed” button in the home page • The system delete the relative ride
Exit conditions	The taxi driver is now in idle and ready to set himself available
Exceptions	The taxi driver presses the button before he reach the destination discovered by a cross examination between the GPS location and the destination location.

Name	**Passenger makes a taxi request**
Actors	Passenger
Entry Conditions	The passenger is logged in and he needs a taxi
Flow of events	• The passenger press the button “Request a taxi” in the home page • The system shows all the input required for a successful submission: ○ Number of passengers ○ Current location ○ Destination

	◦ Additional note • The passenger send the request through a “send request” button and he waits the system assigns to him a taxi.
Exit conditions	• A taxi driver has been properly assigned to the passenger. In case the assigned taxi doesn't meet the passenger's requirements, the passenger can reject the confirmation and halts the process
Exceptions	• The passenger has already a request active and not yet completed. The system halts the process • The inserted fields may not be correct (number of passengers less than 0 or greater than a maximum constant). In this case an error will be shown

Name	**Taxi driver accepts/rejects a request from a passenger**
Actors	Taxi driver
Entry Conditions	The taxi driver is available and able to process a new incoming request (or reservation) from the system
Flow of events	• The taxi driver gets notified (and picked up from the queue) by the system and the mobile application shows all the information about the ride to be performed. • The taxi driver through two buttons (V and X) can accept or reject the request.
Exit conditions	• The taxi driver accepts the request ◦ The passenger reject the confirmation (because maybe the arrival time is too late). In this case the request is halted. ◦ The passenger confirms and the taxi driver starts

	the ride • Otherwise if the taxi driver rejects the request, the system will select the next taxi in the queue (following a FIFO policy)
Exceptions	In case the taxi driver doesn't accept or reject the request within 2 minutes, the request is automatically rejected.

Name	**Taxi driver sets his availability on**
Actors	Taxi driver
Entry Conditions	The taxi driver is logged in the mobile application and wants to start his shift or he has just finished a ride. The taxi driver is not available yet.
Flow of events	• In the home page of the application the taxi driver can press a button which allows him to make himself available • The system adds the taxi to the queue of his current zone
Exit conditions	The taxi driver has been added to the queue and it is ready to be picked up from the system when a new request comes up.
Exceptions	If the taxi driver is assigned to a ride, the system will show an error.

Name	**Passenger sends a reservation request**
Actors	Passenger
Entry Conditions	The passenger is logged in and he will need a taxi in the next few hours/days
Flow of	• The passenger press the button “Reserve a taxi” in

events	the home page • The system shows all the input required for a successful submission: ○ Number of passengers ○ Current location ○ Destination ○ Additional note ○ Hour and date • The passenger confirms pushing a button ("Save reservation")
Exit conditions	The passenger received the confirmations from the system
Exceptions	• If the specified hour and date are within two hours from now, the reservation cannot be completed. • The inserted fields may not be correct (number of passengers less than 0 or greater than a maximum constant). In this case an error will be shown

Name	**Passenger cancels a reservation request**
Actors	Passenger
Entry Conditions	The passenger wants to delete a reservation
Flow of events	• The passenger press to the button "My rides" which allows him to watch the list of all his pending rides. • By choosing the right section ("Reserved") he can see all the reserved ride with all the information about the reservation

	• He selects one reservation ride and swiping left through a “X” button can remove it.
Exit conditions	The reservation has been properly deleted
Exceptions	• If the passenger tries to delete a reservation which will occur within 15 minutes from now, an error will be shown and the process will not be completed

Name	**Passenger creates a shared ride**
Actors	Passenger
Entry Conditions	The passenger wants to make a sharing request with other unknown people
Flow of events	• The passenger from the home page goes to the “Share a ride” section through an appropriate button • Now the passenger click on the “+” button to add a shared ride • The system will show all the requested fields to complete the request ○ Current location ○ Destination ○ Number of passenger ○ Hour and date • The passenger sends the request through a “Confirm” button.
Exit conditions	The passenger received the confirmations from the system
Exceptions	• If the specified hour and date are within two hours

	from now, the request cannot be completed. • The inserted fields may not be correct (number of passengers less than 0 or greater than a maximum constant). In this case an error will be shown

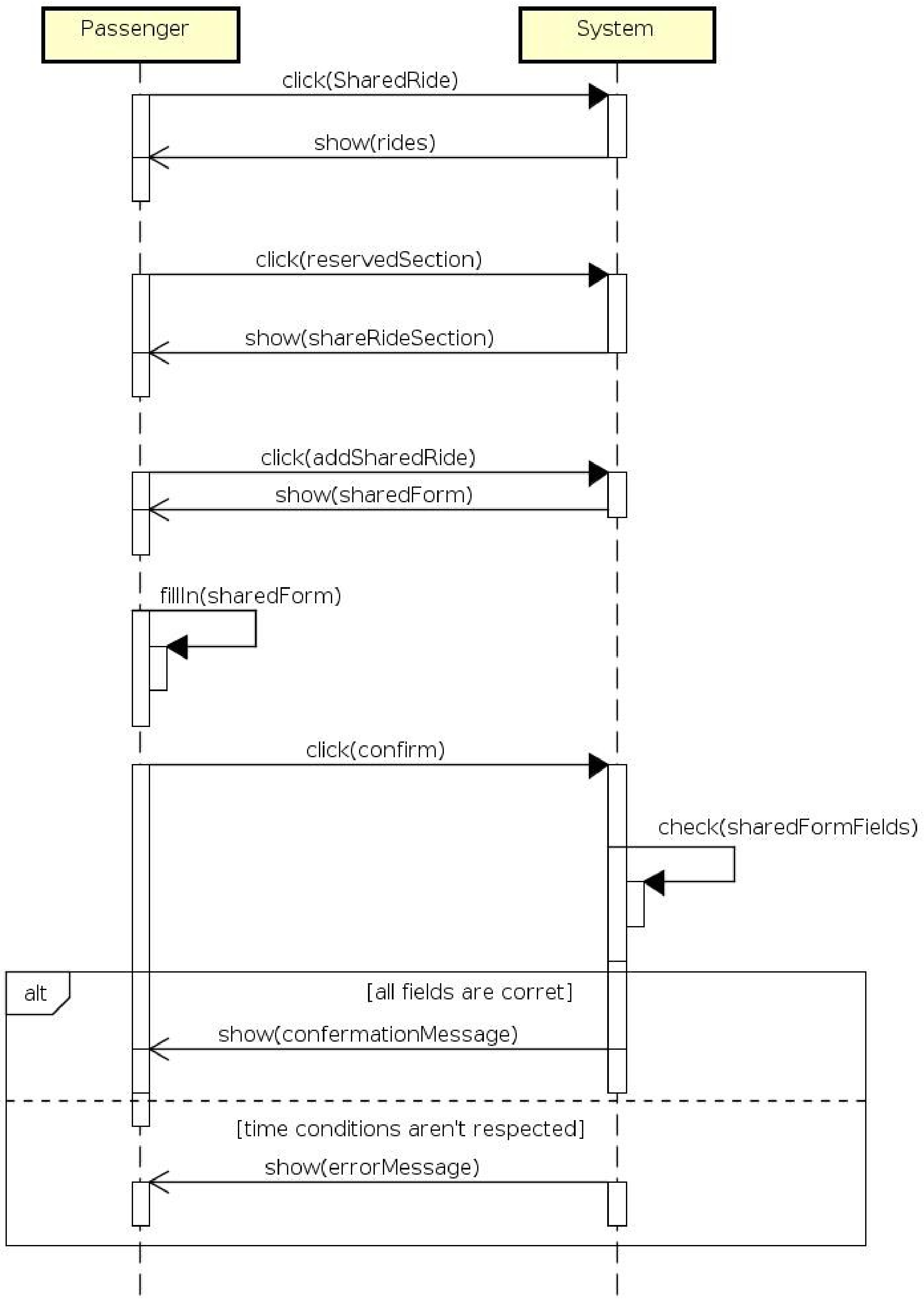
Name	**Passenger joins to a shared ride**
Actors	Passenger
Entry Conditions	A passenger is willing to share a ride with someone else
Flow of events	• From the home page he goes to the “Share a ride” section through an appropriate button. • Through an appropriate search field which allows the passenger to specify: ○ Current location ○ Destination ○ Number of passenger ○ Hour and date • The system will compute and match the requirements with all the shared ride inserted in the system so far. • The system will show a list of all the results and thanks to a “join” button allows the passenger to share a ride specifiying a possible different destination
Exit conditions	The passenger shares a taxi with someone else saving money and reducing personal fees.
Exceptions	• If the passenger tries to join to a shared ride which will occur within 15 minutes from now, an error will be shown and the process will not be completed

	(This exception should not happen since the system would not return this ride)

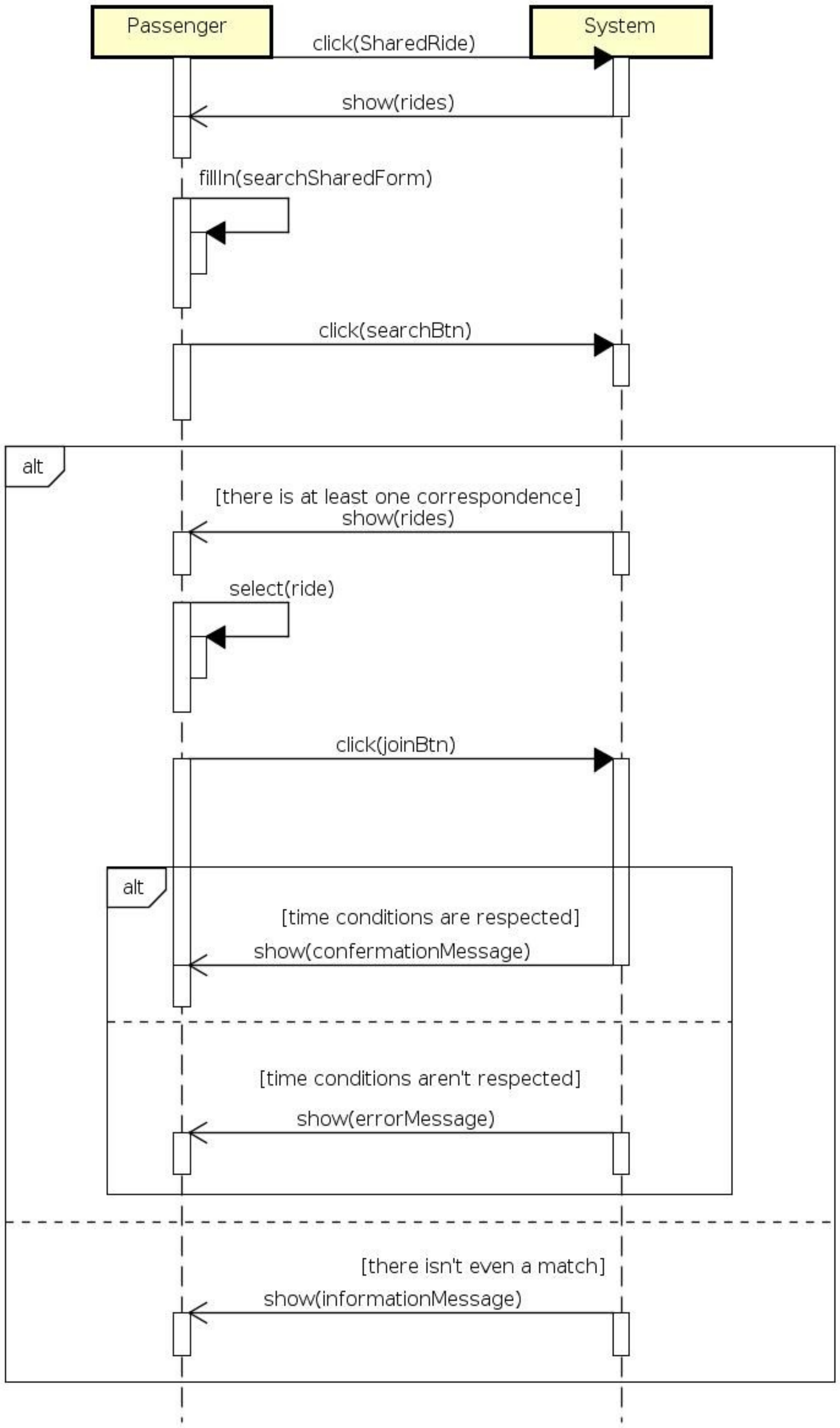
Name	**Passenger removes himself from a shared ride**
Actors	Passenger
Entry Conditions	A passenger wants to remove himself from a sharing ride joined previously.
Flow of events	• From the home page he goes to the “My rides” section through an appropriate button and select the “Shared” tab • He can see the list of all the shared ride which has been joined in a previous situation • Now, pressing to the “disjoin” button corresponding to the chosen shared ride, the passenger will not share this ride anymore.
Exit conditions	The passenger is removed from the shared ride
Exceptions If t	The passenger tries to remove himself from a shared ride which will occur within 15 minutes from now, an error will be shown and the process will not be completed

6.3 Sequence Diagrams

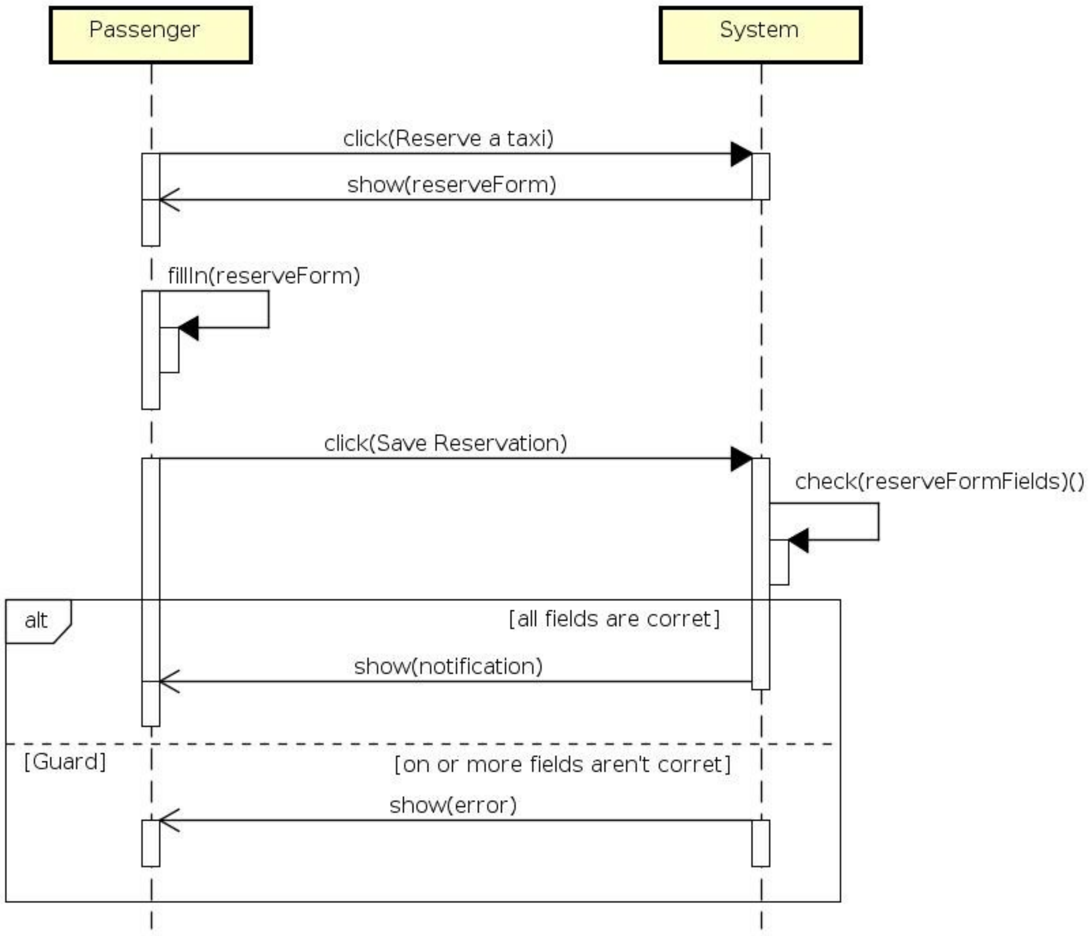
6.3.1 Passenger creates a shared ride



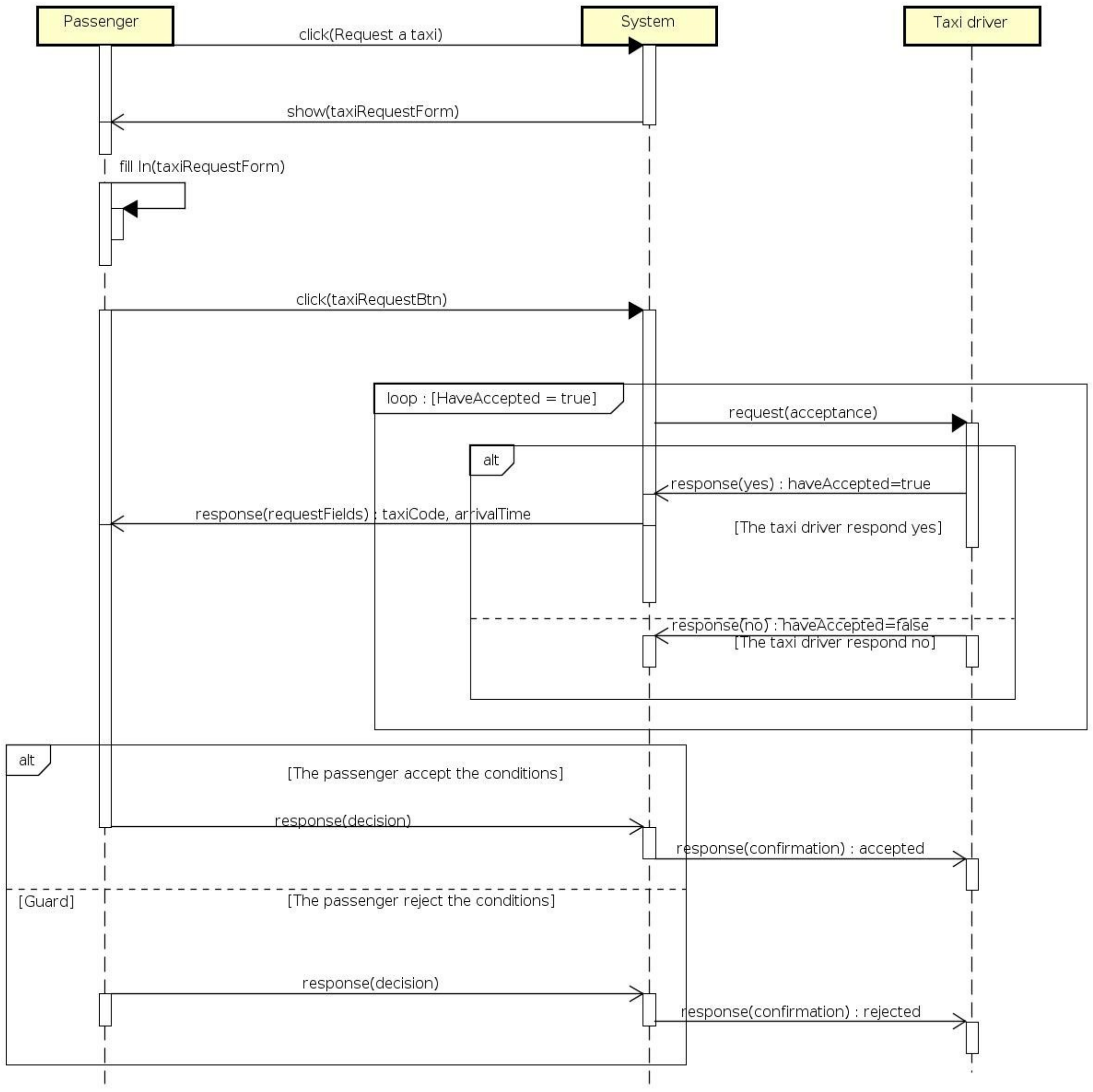
6.3.2 Passenger joins a shared ride



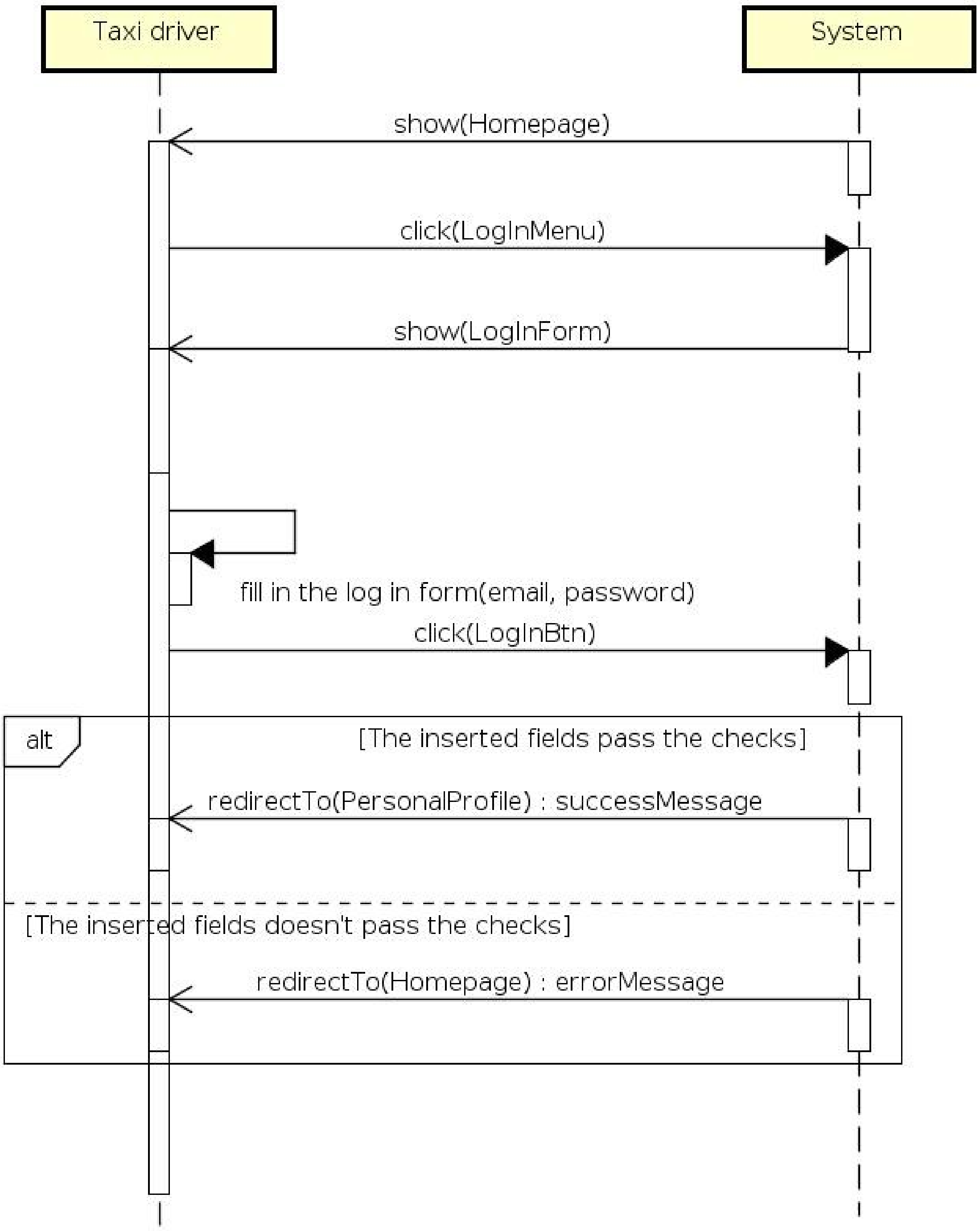
6.3.3 Passenger sends a reservation request



6.3.4 Passenger make a taxi request (taxi driver accept)

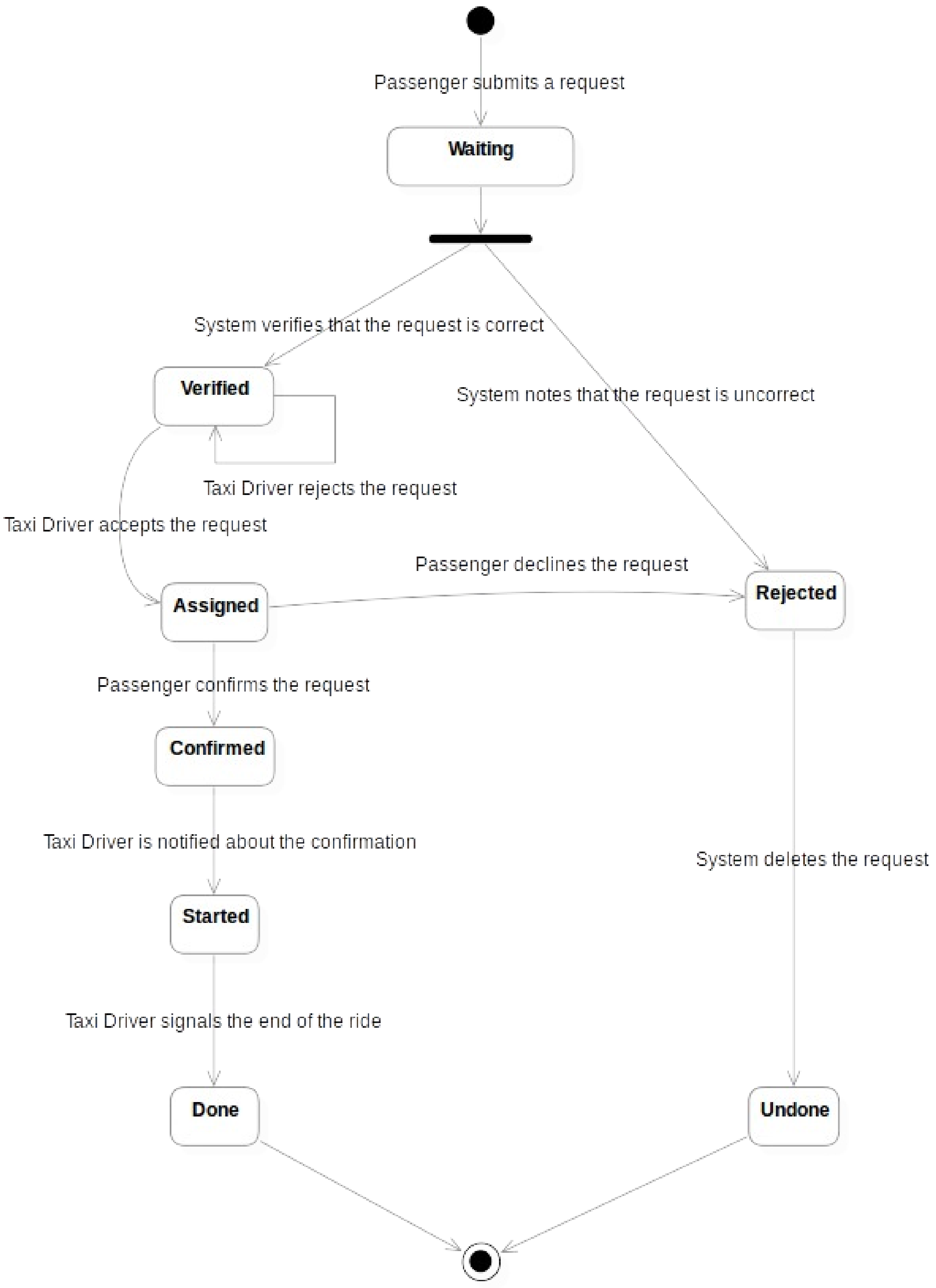

Passenger
System
Taxi driver
click(Request a taxi)
show(taxiRequestForm)
fill In(taxiRequestForm)
click(taxiRequestBtn)
loop : [HaveAccepted = true]
request(acceptance)
alt
response(yes) : haveAccepted=true
response(requestFields) : taxiCode, arrivalTime
[The taxi driver respond yes]
response(no) : haveAccepted=false
[The taxi driver respond no]
alt
[The passenger accept the conditions]
response(decision)
response(confirmation) : accepted
[Guard]
[The passenger reject the conditions]
response(decision)
response(confirmation) : rejected

6.3.5 Taxi driver login

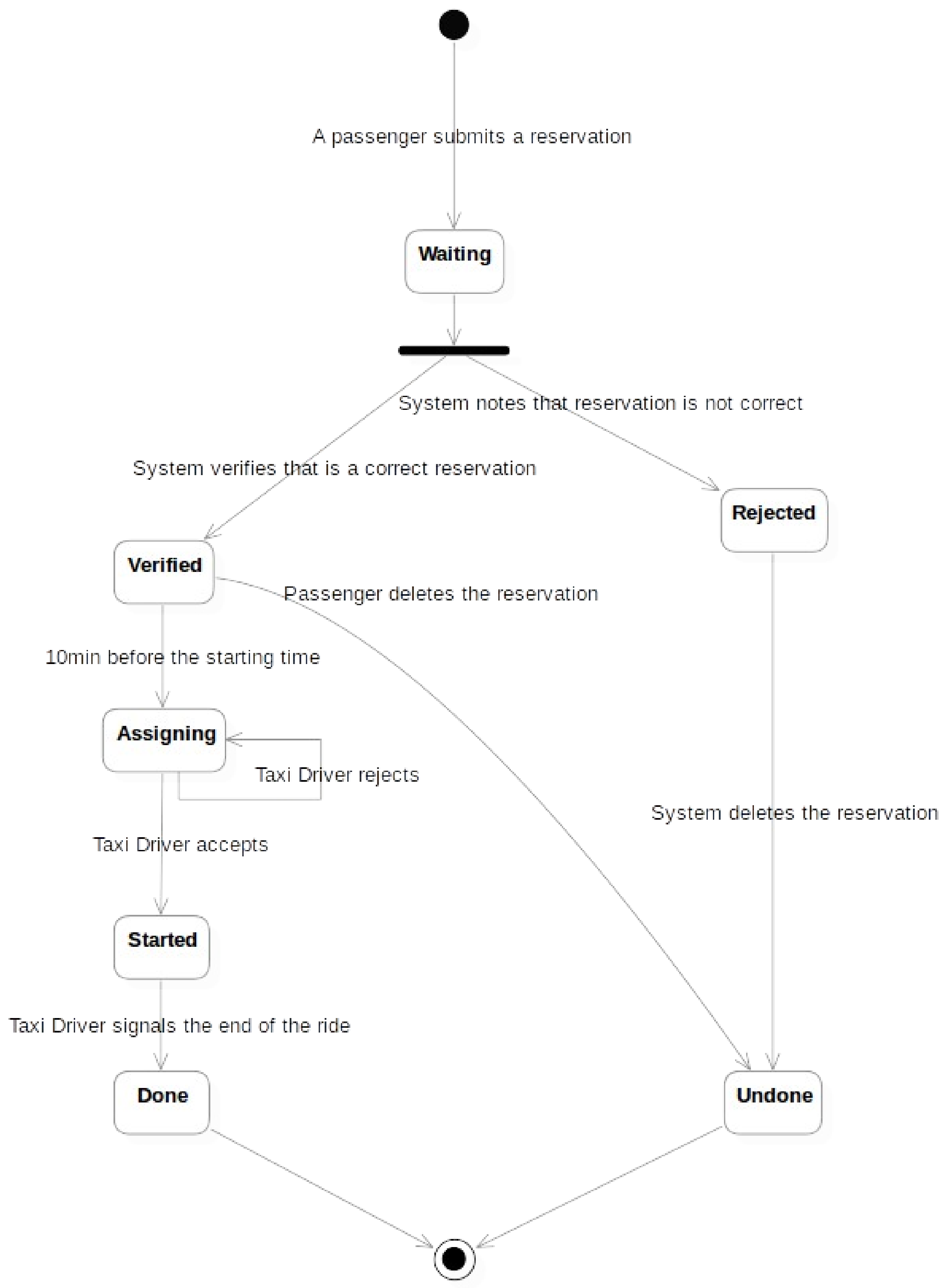

Taxi driver
System
show(Homepage)
click(LogInMenu)
show(LogInForm)
fill in the log in form(email, password)
click(LogInBtn)
alt
[The inserted fields pass the checks]
redirectTo(PersonalProfile) : successMessage
[The inserted fields doesn't pass the checks]
redirectTo(Homepage) : errorMessage

6.4 State Chart Diagrams

6.4.1 Passenger makes a taxi request

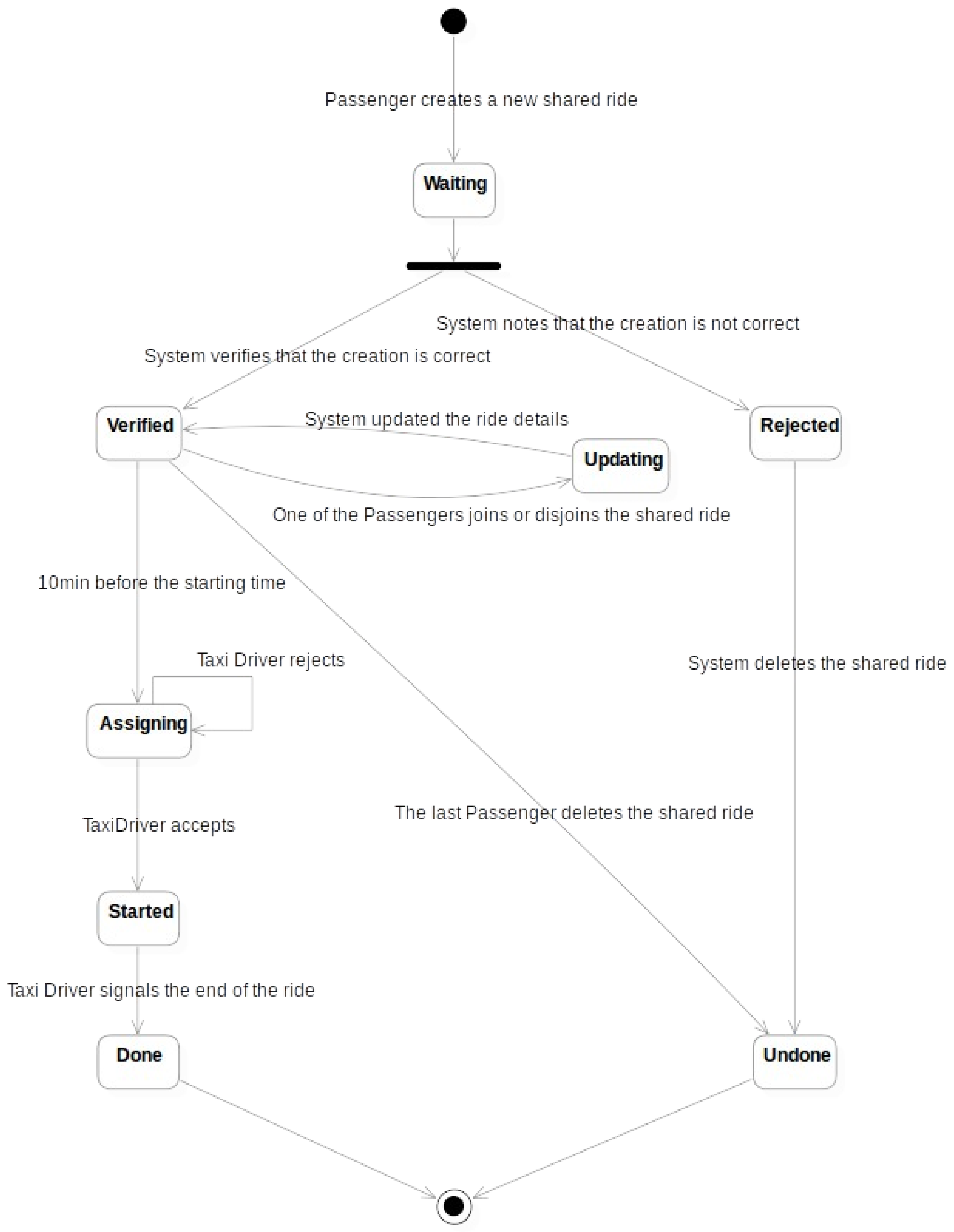

Passenger submits a request
Waiting
System verifies that the request is correct
Verified
System notes that the request is uncorrect
Taxi Driver rejects the request
Taxi Driver accepts the request
Passenger declines the request
Assigned
Rejected
Passenger confirms the request
Confirmed
Taxi Driver is notified about the confirmation
System deletes the request
Started
Taxi Driver signals the end of the ride
Done
Undone

6.4.2 Passenger sends a reservation request

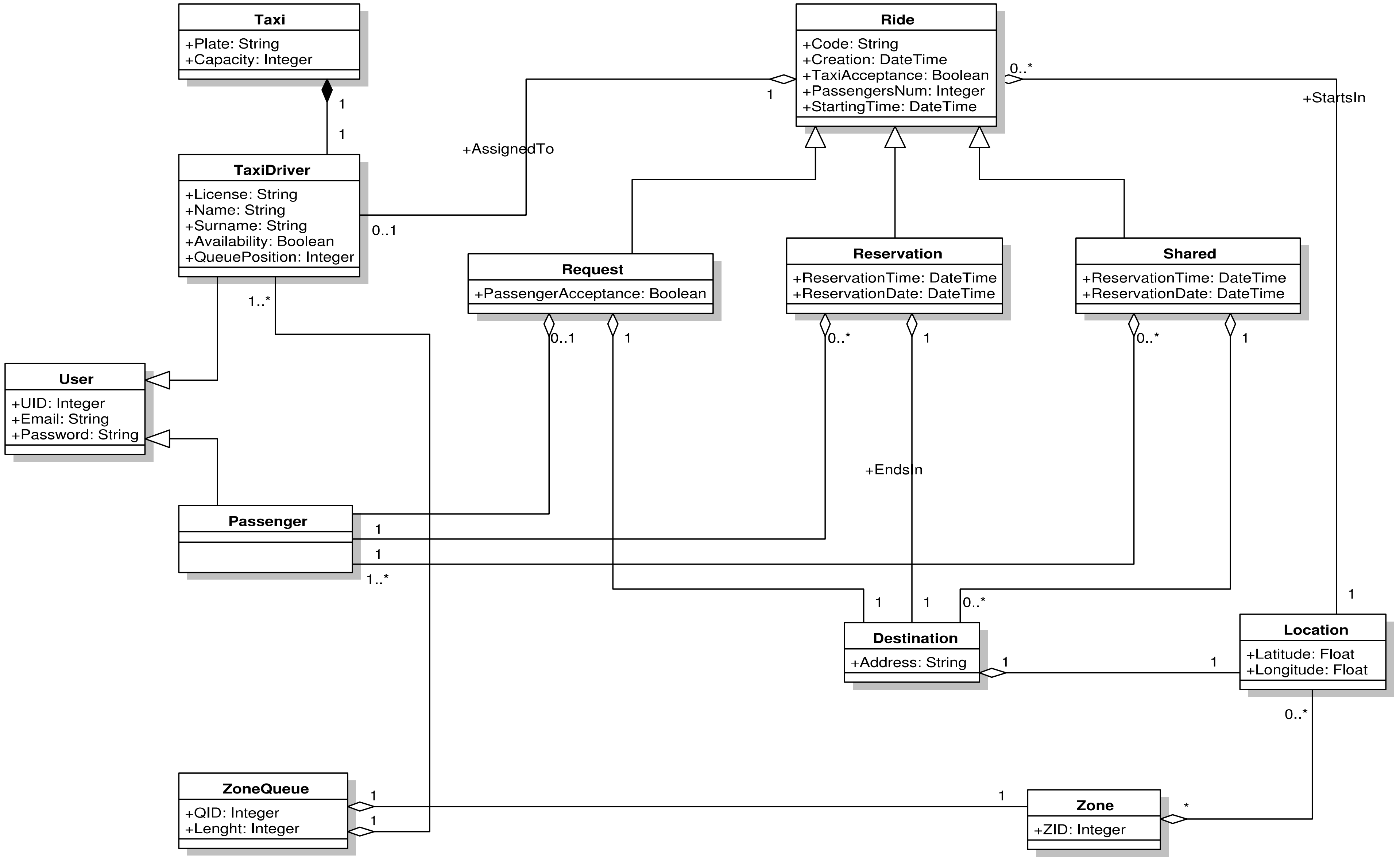


A passenger submits a reservation
Waiting
System notes that reservation is not correct
System verifies that is a correct reservation
Rejected
Verified
Passenger deletes the reservation
10min before the starting time
Assigning
Taxi Driver rejects
System deletes the reservation
Taxi Driver accepts
Started
Taxi Driver signals the end of the ride
Done
Undone

6.4.3 Passenger creates a shared ride


Passenger creates a new shared ride
Waiting
System notes that the creation is not correct
System verifies that the creation is correct
Verified
System updated the ride details
Updating
Rejected
One of the Passengers joins or disjoins the shared ride
10min before the starting time
Taxi Driver rejects
System deletes the shared ride
Assigning
The last Passenger deletes the shared ride
TaxiDriver accepts
Started
Taxi Driver signals the end of the ride
Done
Undone

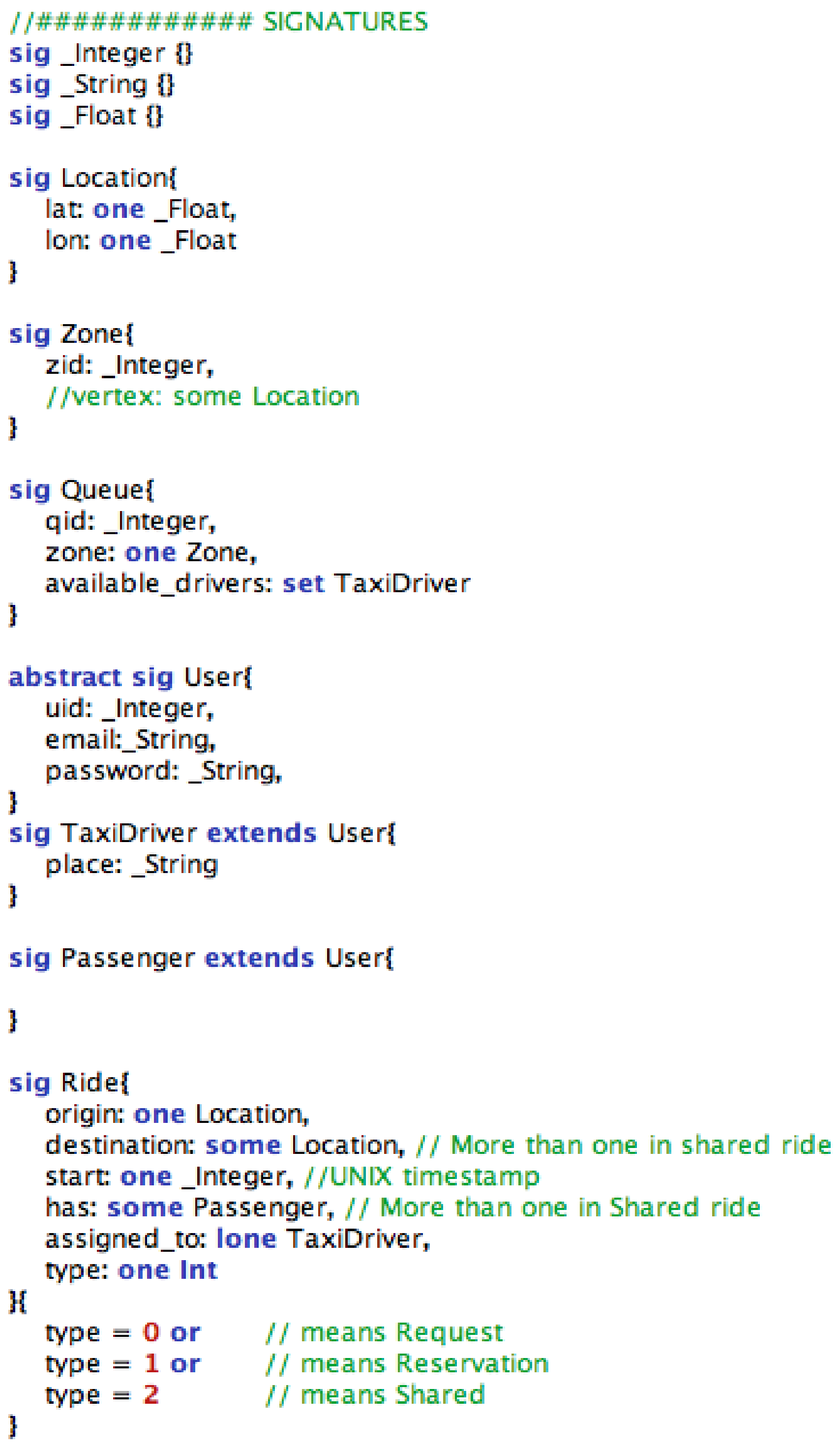
6.5 Class Diagram



7. Alloy

Every worlds generated by Alloy is a particular instance of time among all the possible configurations.

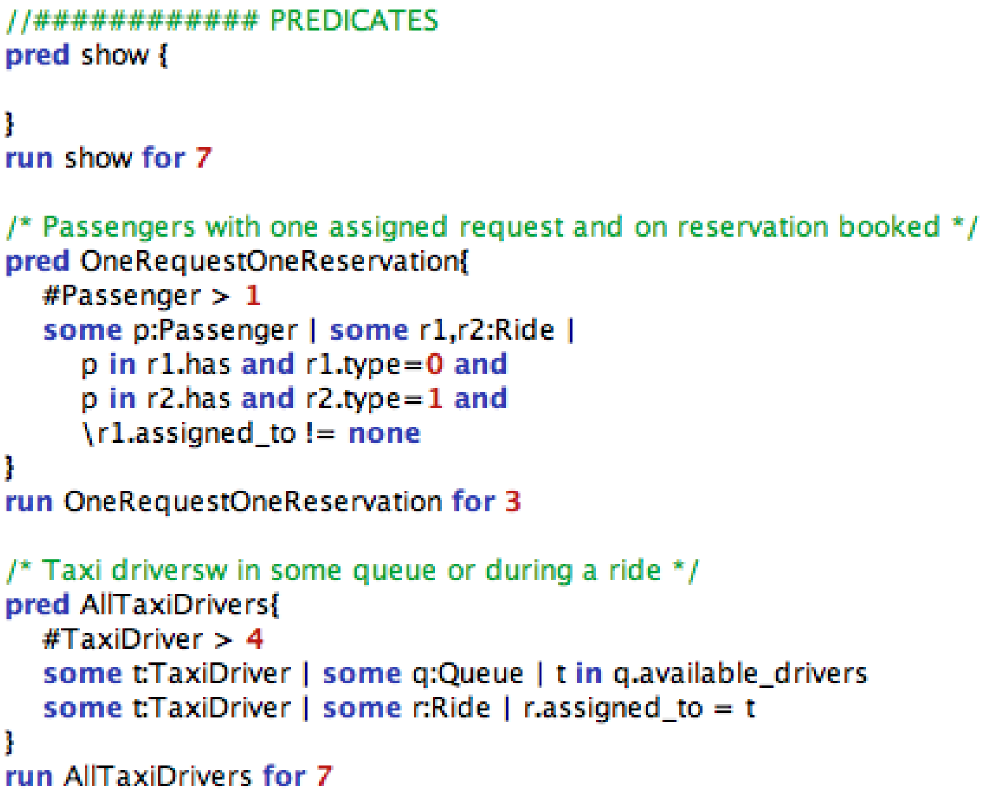
7.1 Modeling



```
//############ SIGNATURES
sig _Integer {}
sig _String {}
sig _Float {}

sig Location{
    lat: one _Float,
    lon: one _Float
}

sig Zone{
    zid: _Integer,
    //vertex: some Location
}

sig Queue{
    qid: _Integer,
    zone: one Zone,
    available_drivers: set TaxiDriver
}

abstract sig User{
    uid: _Integer,
    email:_String,
    password: _String,
}
sig TaxiDriver extends User{
    place: _String
}

sig Passenger extends User{

}

sig Ride{
    origin: one Location,
    destination: some Location, // More than one in shared ride
    start: one _Integer, //UNIX timestamp
    has: some Passenger, // More than one in Shared ride
    assigned_to: lone TaxiDriver,
    type: one Int
}{
    type = 0 or      // means Request
    type = 1 or      // means Reservation
    type = 2         // means Shared
}
```

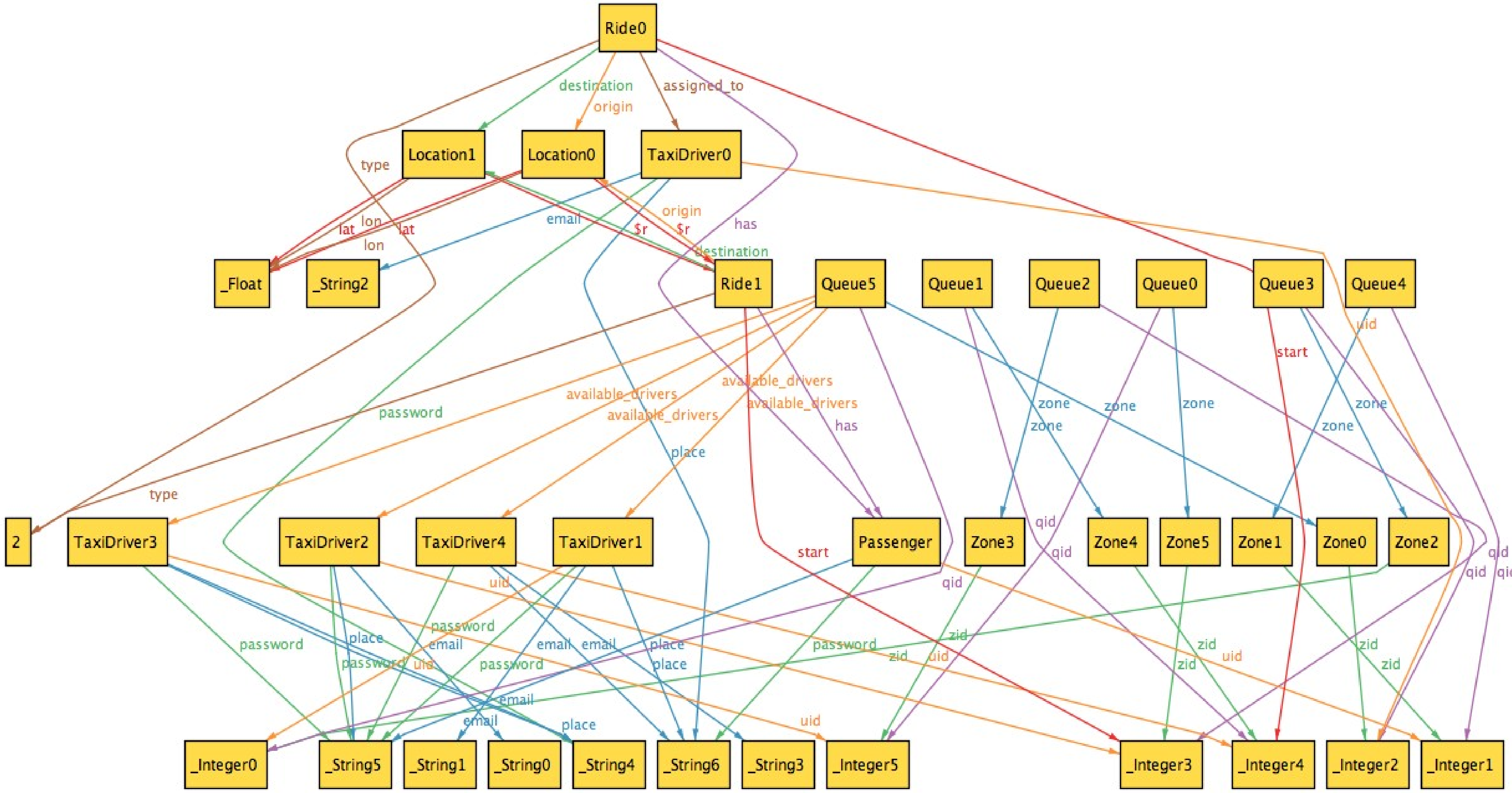
```
//############ FACTS
fact UnivocityProperties{
    /* TaxiDriver and passenger must have different ids and emails*/
    all disj p1,p2: Passenger | p1.uid != p2.uid and p1.email != p2.email
    all disj t1,t2: TaxiDriver | t1.uid != t2.uid and t1.email != t2.email
    all p: Passenger,t:TaxiDriver | p.uid != t.uid

    /* Zone with different ids */
    all disj z1,z2: Zone | z1.zid != z2.zid

    /* Queue with different ids */
    all disj q1,q2: Queue | q1.qid != q2.qid
}

fact QueueProperties{
    /* One zone -> One Queue */
    all q1, q2 : Queue |  q1!=q2  impliesq1.zone  != q2.zone
    /* One queue -> One zone */
    all z : Zone | one q: Queue | q.zone = z
}

fact TaxiDriverProperties{
    /* One taxi which is assigned to a ride, must not belong to any queue */
    all t:TaxiDriver,r:Ride,q:Queue | r.assigned_to=t implies t not in q.available_drivers

    /* One driver must be at most in one queue at a given time */
    no disj q1,q2: Queue, t:TaxiDriver |  t in q1.available_drivers and t in q2.available_drivers

    /* A taxi driver can be assigned to at most one generic ride at once */
    no disj r1,r2:Ride | r1.assigned_to=r2.assigned_to
}

fact RideProperties{
    /* Any request has exactly one passenger */
    all r:Ride | r.type=0 implies #(r.has) = 1

    /* Any reservation has exactly one passenger */
    all r:Ride | r.type=1 implies #(r.has) = 1

    /* Number of destination must be equal to the number of passenger */
    all r:Ride | #(r.destination) = #(r.has)

    /* Every destination must have different origin and destination*/
    all r:Ride | r.origin not in r.destination
}

fact LocationProperties{
    /* Every location is assigned to a location */
    all l:Location | some r:Ride | l in r.origin or l in r.destination
}

fact ZoneProperties{
    /* Every zone has N = 4 vertex */
    //all z:Zone | #(z.vertex) = 4
}
```

```
//############ ASSERTS
/* There are no taxidriver which has assigned two different rides */
assert noTaxiDriverTwoRidesAssigned{
    no t:TaxiDriver | some disj r1,r2:Ride | r1.assigned_to = t and r2.assigned_to = t
}
check noTaxiDriverTwoRidesAssigned

/* There no exist taxi driver which are in two different queues at a given time  */
assert noTaxiDriverAreInTwoDifferentQueue{
    no t:TaxiDriver | some disj q1,q2:Queue |
        t in q1.available_drivers and
        t in q2.available_drivers
}
check noTaxiDriverAreInTwoDifferentQueue

/* There no exist passenger with more than one request */
assert noPassengerWithMoreThanOneRequest{
    no disj r1,r2:Ride | r1.type=0 and r2.type=0 and r1.has = r2.has
}
check noPassengerWithMoreThanOneRequest

/* There no exist passenger with more than one ride with a taxi driver assigned at a given time*/
assert noPassengerWithMoreThanTwoRideAssigned{
    no p:Passenger | some disj r1,r2:Ride |
        p in r1.has and p in r2.has and
        r1.assigned_to != none and r2.assigned_to!=none
}
check noPassengerWithMoreThanTwoRideAssigned
```



```
//############ PREDICATES
pred show {

}
run show for 7

/* Passengers with one assigned request and on reservation booked */
pred OneRequestOneReservation{
   #Passenger > 1
   some p:Passenger | some r1,r2:Ride |
      p in r1.has and r1.type=0 and
      p in r2.has and r2.type=1 and
      \r1.assigned_to != none
}
run OneRequestOneReservation for 3

/* Taxi driversw in some queue or during a ride */
pred AllTaxiDrivers{
   #TaxiDriver > 4
   some t:TaxiDriver | some q:Queue | t in q.available_drivers
   some t:TaxiDriver | some r:Ride | r.assigned_to = t
}
run AllTaxiDrivers for 7
```

7.2 World Generated



8. Used tools

- **Libre office**: to redact and to format this document;
- **Proto.io**: to create the UI sketches;
- **Alloy Analyzer 4.2**: to prove the consistency of our model;
- **Asta**: to create Use Cases Diagrams;
- **Visual Paradigm 10 Community Edition**: to create Sequence Diagrams and State Charts;
- **StarUML**: to create Class Diagrams

9. Hours of works

- **Matteo Martinelli**: 28 hours {13 hours to write part of the documents, 3 hour for the user interface, 2 hour for the Use case Diagram, 10 hours for Alloy}
- **Simone Biffi**: 21 hours {13 hours to write part of the documents, 6 hours for Sequence diagram, 2 hour for the Use case Diagram}
- **Pietro Astolfi**: 21 hours {13 hours to write part of the documents, 3 hours for Class diagram, 5 hours for State chart diagram}
- **Together**: 4 hours to think how to describe every aspect of the project, 6 hours for revising everything and make some corrections.