



POLITECNICO
MILANO 1863

A.A 2015-2016

Software Engineering 2: "myTaxiService"

Requirements **A**nalysis and **S**pecifications
Document

Massimiliano Paci (mat. 852720)
Giovanni Patruno (mat. 852658)

6 November 2015

Contents

1	Introduction	4
1.1	Scope	4
1.2	Actors	4
1.3	Goals	4
1.4	Definitions, acronyms and abbreviations	6
1.4.1	Definitions	6
1.4.2	Acronyms	7
1.4.3	Abbreviations	7
1.5	References	7
1.6	Overview	7
2	Overall description	8
2.1	Product perspective	8
2.1.1	Quality of Service	8
2.2	User characteristics	8
2.3	Constraints	8
2.3.1	Regulatory policies	8
2.3.2	Hardware limitations	8
2.3.3	Documents related	9
2.4	Assumptions and dependencies	9
3	Specific Requirements	10
3.1	External Interface Requirements	10
3.1.1	User Interfaces	10
3.1.1.1	Home Page	10
3.1.1.2	Booking Page	11
3.1.1.3	List of Bookings	12
3.1.1.4	Driver's profile	13
3.1.1.5	Administration Panel	14
3.1.1.6	Passenger's notification	14
3.1.1.7	Taxi Driver's notification	15
3.1.2	API Interfaces	15
3.1.3	Hardware Interfaces	15
3.1.4	Software Interfaces	15
3.1.5	Communication Interfaces	15
3.1.6	Memory	16
3.2	Functional Requirements	16
3.2.1	[G1] Allow the registration on the platform	16
3.2.2	[G2] Allow user to log-in on the platform	16
3.2.3	[G3] Allow user to book a new trip	16
3.2.4	[G4] Allow user to modify one of his booked trip	17
3.2.5	[G5] Allow user to delete one of his booked trip	17
3.2.6	[G6] Allow user to receive a cost preview for a selected route	17

3.2.7	[G7] Allow user to view the EWT for a request	17
3.2.8	[G8] Allow user to release a feedback on a specific taxi driver	17
3.2.9	[G9] The application will notify taxi driver for a new ride	18
3.2.10	[G10] Allow taxi driver to inform the system about his availability	18
3.2.11	[G11] Allow taxi driver to accept or deny ride's requests	18
3.2.12	[G12] Allow administration to inspect all users' profiles .	18
3.3	Scenarios	18
3.3.1	Scenario 1: Passenger future-booking case	18
3.3.2	Scenario 2: Payment methods and feedback releasing . . .	19
3.3.3	Scenario 3: Passenger now-booking case	19
3.3.4	Scenario 4: Administration profile inspecting	19
3.4	Domain Properties	20
3.5	UML Models	20
3.5.1	Use case models	20
3.5.1.1	Visitor Registration	20
3.5.1.2	Visitor Login	22
3.5.1.3	Create booking	24
3.5.1.4	Modify booking	27
3.5.1.5	Delete booking	29
3.5.1.6	Feedback Releasing	31
3.5.1.7	Profile Management	33
3.5.1.8	Notification Management	34
3.5.1.9	Taxi driver availability management	35
3.5.2	Class Diagram	38
3.5.3	State Chart Diagram	39
4	Alloy	40
4.1	Alloy calculator results	45
4.2	Metamodel	47
4.3	Booking Insertion	48
4.4	Booking Deletion	50
4.5	Feedback Releasing	51
4.6	World	53
5	Appendix	54
5.1	Software and tool used	54
5.2	Working Hours	54

1 Introduction

1.1 Scope

The aim of this project is to create a platform, named **myTaxiService**, to optimize the taxi service of a large city. Passengers will be able to book a taxi for a certain route, view a cost preview and the relative waiting time.

Taxi drivers will be able to inform the system about their availability in a certain zone, confirm or deny a ride and consult the feedback of the passengers.

The city is divided in taxi zones and each one is associated to a queue of taxis. The system assigns every booking to an available driver in the ride's zone. If there is not a driver available in that specific zone, the system automatically assigns another available driver from another zone.

Taxi drivers will have access to the platform too: they will be able to check their queue state and also see statistics on their routes.

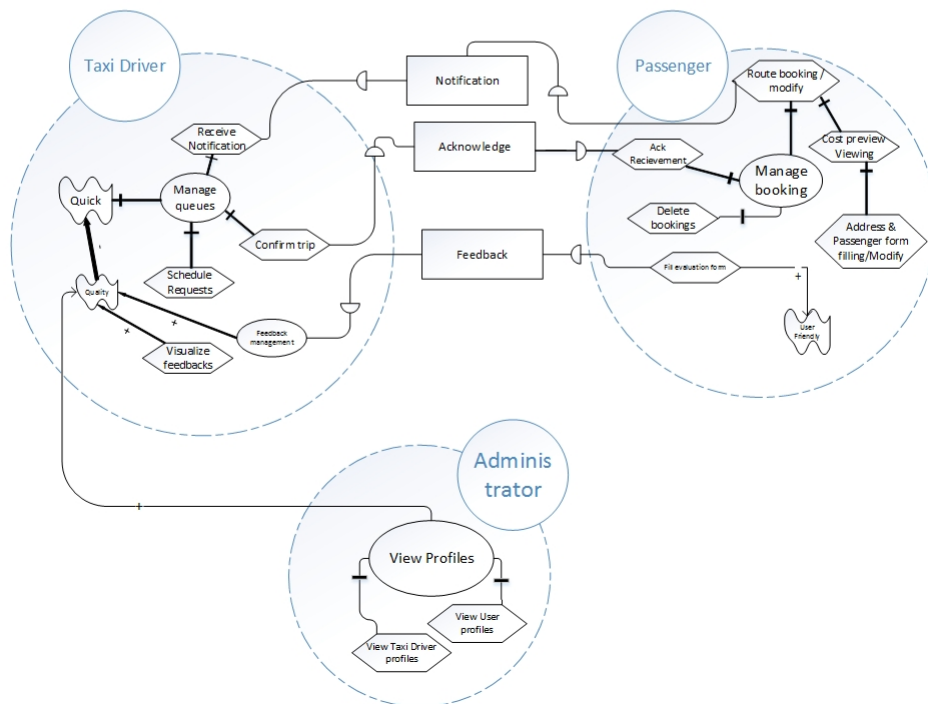
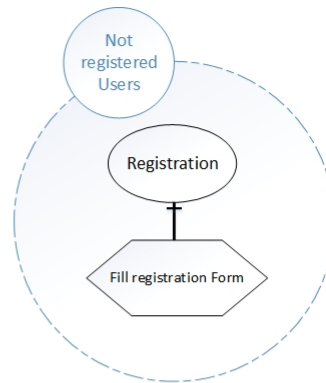
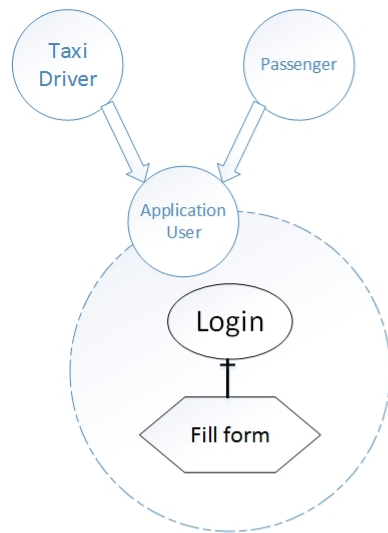
The platform will be scalable and it could be integrated with other applications to implement new functionalities.

1.2 Actors

- Visitor: everyone that enter to the website, but is not-registered: can only have access to the log in page
- User: who is regular registered on the platform. Can be Passenger, Taxi Driver or Administrator.
- Passenger: type of user that can create, modify and delete a trip for himself and/or other people; have access to a ride's preview and give a feedback to a taxi driver after a trip.
- Taxi Driver: type of user that inform the system about his availability, can accept or deny a booking and views his trips' queue state and his feedback.
- Administrator: type of user that have access to the whole system and is allowed to view all the trips and taxi driver's feedbacks
- Passenger: who really travels with a taxi

1.3 Goals

The system behaviour is also designed with an I* Modeling diagram, to provide a complete overview of the system, considering its goals, requirements and actors. This is only a way to represent every part of the diagram that will be discussed in next chapters.



The system has the following goals:

- [G1] Allow the registration on the platform
- [G2] Allow user to log-in on the platform
- [G3] Allow user to book a new ride
- [G4] Allow user to modify one of his booked ride
- [G5] Allow user to delete one of his booked ride
- [G6] Allow user to receive a cost preview for a selected route
- [G7] Allow user to view the EWT for a request
- [G8] Allow user to release a feedback on a specific taxi driver
- [G9] The application will notify taxi driver for a new ride
- [G10] Allow taxi driver to inform the system about his availability
- [G11] Allow taxi driver to accept or deny ride's requests
- [G12] Allow administration to inspect all users' profiles.
- [G13] For every zone the application will compute taxi driver's queue.

1.4 Definitions, acronyms and abbreviations

1.4.1 Definitions

- Ride: a taxi booking performed by a passenger.
- Zone: A delimited square area (0.2 km^2) at which belongs addresses.
- Passenger: who really travel with the taxi.
- Profile: Every registered user has got a profile which contains all necessary info needed by the platform. We have three types of profile corresponding to three type of users.
- Navigation bar: Every user in the platform has got a navigation bar that contains a set of links useful to manage their accounts like: Profile, settings and so forth.
- past-booking: A booking that regards a performed ride by the passenger.
- now-booking: A ride booking that is required from the user that needs a taxi in that moment.
- future-booking: A ride booking that is required from the user in a *second* moment.

1.4.2 Acronyms

- RASD: Requirements Analysis and Specification Document
- EWT: Estimated Waiting Time

1.4.3 Abbreviations

- [Gn]: n-goal
- [Rn]: n-functional requirement
- [Dn]: n-domain assumption
- iff: if and only if

1.5 References

- *Stakeholder specification document*
- *IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications.*
- *Alloy Documentation*

1.6 Overview

The document is structured in this following sections:

1. This section contains a generic introduction to the product and a prospective of high level specifications.
2. This section gives an overall view of the constraints and assumptions, also giving an idea of the possibles future implementations.
3. This section gives the specification of requirements for each goal, giving samples of the application usage and UML diagrams to understand better the working behaviour.
4. This section gives an Alloy application model and samples of some goal simulation.
5. This section describes the softwares used to redact this RASD document, and specifics the time spent for redact this document.

2 Overall description

2.1 Product perspective

The service will be available through a web application accessible either by PCs or mobile devices. Furthermore will be available a mobile app for Android, iOS and Windows Phone for a better usability

2.1.1 Quality of Service

- Security: myTaxiService will be a safe platform, all internet communication (Like Gps positions) will be encrypted.
- Usability: Platform interface will be user-friendly and all links will be reachable by maximum 3 clicks.
- Availability: Platform will be available 24/7
- Exception handling: Platform will have a structure designed to support any kind of exception related on all its functionalities e.g. User-duplicate exception, booking-duplicate exception, no-money on the credit card exception and so forth.
- Interoperability: myTaxiService will provide proxy interfaces to being integrated in a different system.
- Scalability: Design of the platform will be splitted in modules so that each of these modules could be updated separately.

2.2 User characteristics

The application is thought to reach passengers on one side and taxi drivers on the other side. Both these actors must be able to use a web browser (or a mobile app) and have access to internet.

2.3 Constraints

2.3.1 Regulatory policies

myTaxiService have to meet the taxi services policies and to have permits to get gps location of the device on which the application is running. The mobile app must comply with the policies of the various store (Play Store, Windows Store, App Store)

2.3.2 Hardware limitations

myTaxiService doesn't have any hardware limitation.

2.3.3 Documents related

- Requirements and Analysis Specification Document (RASD)

2.4 Assumptions and dependencies

In order to overcome deficiencies and / or disambiguation of the problem given, we assume that:

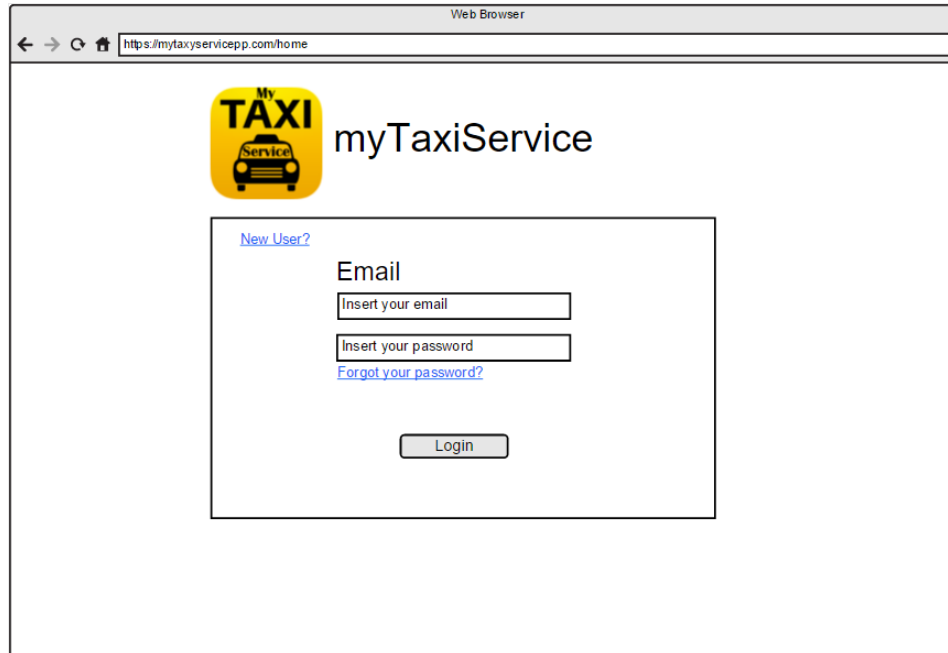
- There are always enough taxis in the city to meet all requests.
- Each Taxi Driver belongs to a specific zone.
- For a specific booking the system will assign the first available taxi driver in the zone, if there are no Taxi Drivers available the system will try to find one in the nearest zone and will put all the taxi drivers that have been refused the call in the back on the queue of their specific belonging zone.
- The feedback is released iff the passenger has performed the booked ride.
- Passenger may book for other people, so that, there could be booking with the same date but not with exactly the same date and time due to a possible presence of a user error.
- The taxi drivers will be already registered on the platform.
- The platform always receives precise (with an error of 0.02km^2) GPS informations in order to compute the distribution of taxis in the city.
- There will be a platform (Mobile application, Web application and so forth) to run the entire application.
- Different users must have different emails.
- Application must be used for a city with an area $\geq 0.02\text{km}^2$ in order to have at least 1 zone.
- Address must be associated to exactly one zone.

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

3.1.1.1 Home Page



The image shows a web browser window with the address bar displaying `https://mytaxyservicepp.com/home`. The page features a logo on the left consisting of a yellow square with a black car icon and the text "My TAXI Service". To the right of the logo is the text "myTaxiService". Below the logo and text is a registration and login form. The form contains a link "New User?" in blue. Under the heading "Email", there are two input fields: "Insert your email" and "Insert your password". Below these fields is a link "Forgot your password?" in blue. At the bottom of the form is a "Login" button.

This mock-up shows myTaxiService's Homepage. Here a visitor can register himself on the platform and an already registered user can log in.

3.1.1.2 Booking Page

The mock-up shows a web browser window with the URL `https://mytaxyservicepp.com/booking`. The user is logged in as Mario Rossi. The page title is "Book your ride!". The form includes fields for "Starting address" (with a "My pos" button), "Destination address", and "Number of passengers" (set to 2). A calendar shows the date 11/11/2015, and a time selector shows 11:55. There are radio buttons for "Book for now" (selected) and "Book for a specific date", and another set for "Credit Card" and "Cash" (with "Cash" selected). A map on the right shows a location pin. Below the map, it says "Cost preview: 15€" and "Estimated waiting time: 15 minutes". A "Book now!" button is at the bottom right.

Web Browser

← → ↻ 🏠 `https://mytaxyservicepp.com/booking`

Mario Rossi 👤 🔔 ⚙️

Book your ride!

Choose the starting address:

Choose the destination address:

Number of passengers: ▼

11/11/2015 📅 ☒ Book for now
☐ Book for a specific date

11:55 ⌚

📅 November 11, 2015

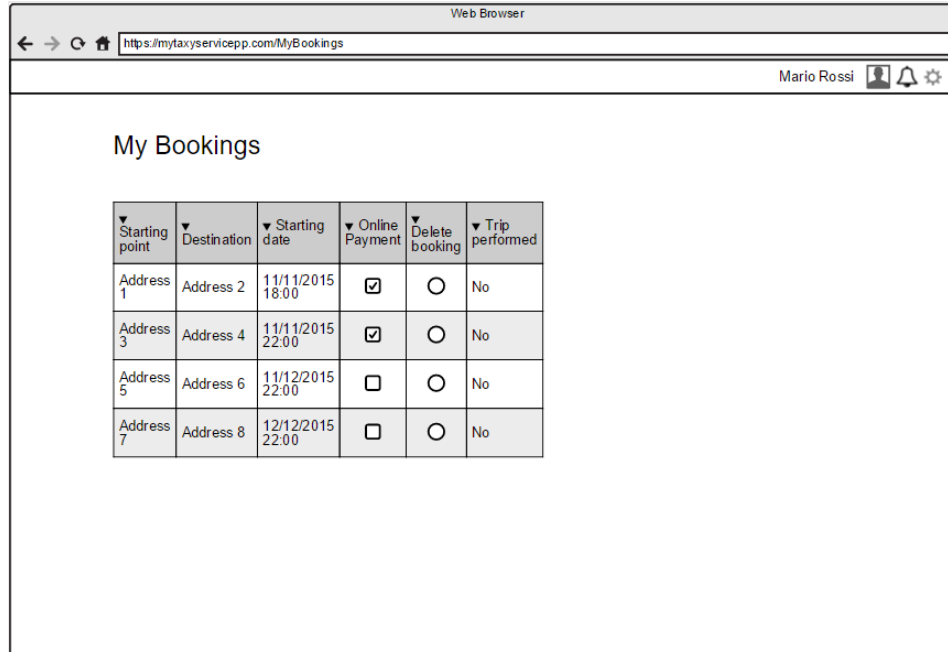
Su	Mo	Tu	We	Th	Fr	Sa
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

☐ Credit Card
☒ Cash

Cost preview: 15€
Estimated waiting time: 15 minutes

This mock-up shows the booking page of a Passenger user. He can write starting and destination address, number of passengers, date and time of the ride and how to pay. It shows also the cost preview of the selected route and the EWT

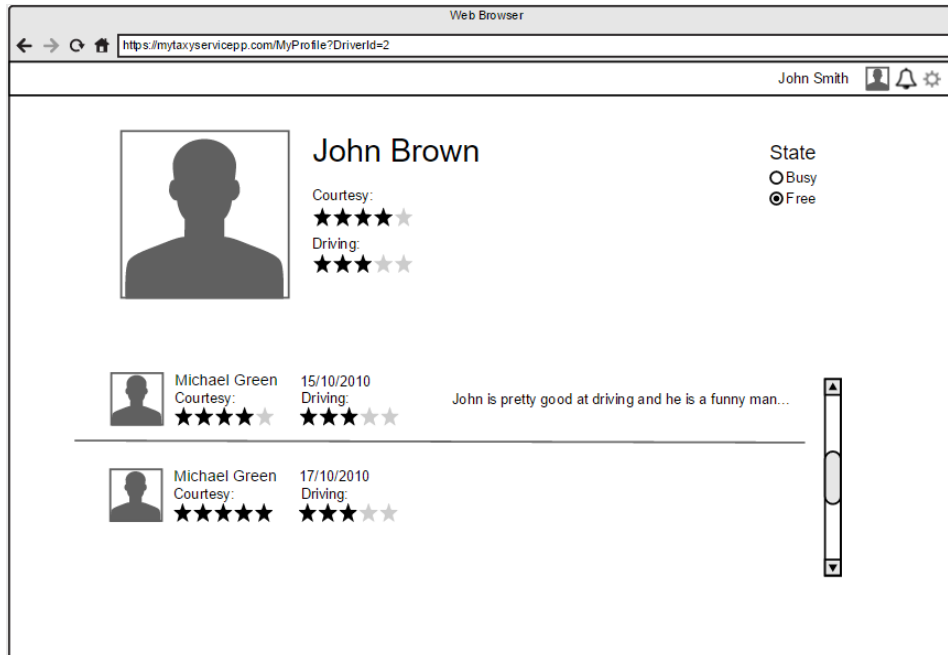
3.1.1.3 List of Bookings



▼ Starting point	▼ Destination	▼ Starting date	▼ Online Payment	▼ Delete booking	▼ Trip performed
Address 1	Address 2	11/11/2015 18:00	<input checked="" type="checkbox"/>	<input type="radio"/>	No
Address 3	Address 4	11/11/2015 22:00	<input checked="" type="checkbox"/>	<input type="radio"/>	No
Address 5	Address 6	11/12/2015 22:00	<input type="checkbox"/>	<input type="radio"/>	No
Address 7	Address 8	12/12/2015 22:00	<input type="checkbox"/>	<input type="radio"/>	No

This page can be viewed by the passenger and shows the list of past and future booking. By clicking on a *past-booking* he can access at the feedback page if the booking has not already a feedback associated. By clicking on a *future-booking* he can access at the page that shows his EWT, map and eventually, the driver.

3.1.1.4 Driver's profile



This mock-up shows a taxi driver's profile. There is a list of feedback of past rides and a radio button to choose the current availability of the driver.

3.1.1.5 Administration Panel

The mock-up shows a web browser window with the URL `https://mytaxyservicepp.com/AdministrationPanel/DriversList`. The page title is "Platform Users". There are two tabs: "Users" and "Taxi drivers". A search bar is present with the text "Insert name to find a user:" and a search icon. Below the tabs is a table with four columns: "Name", "Surname", "Zone", and "Busy". The table contains four rows of data. The "Busy" column has checkboxes, with the first and fourth rows checked.

▼ Name	▼ Surname	▼ Zone	▼ Busy
Name1	Surname1	Zone 1	<input checked="" type="checkbox"/>
Name2	Surname2	Zone 2	<input type="checkbox"/>
Name3	Surname3	Zone 3	<input type="checkbox"/>
Name4	Surname4	Zone 4	<input checked="" type="checkbox"/>

This mock-up shows the administrator Page with the list of all users, divided for typology (Passengers and Taxi Drivers)

3.1.1.6 Passenger's notification

The mock-up shows a notification box with the text "Your taxi has arriving". Below this, it says "Taxi driver's name: Christian White" and "Estimated waiting time: 8 minutes". At the bottom of the box is an "OK!" button.

Your taxi has arriving

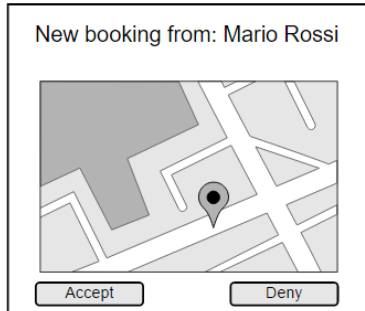
Taxi driver's name: Christian White

Estimated waiting time: 8 minutes

OK!

This mock-up shows the passenger's notification received for an oncoming ride

3.1.1.7 Taxi Driver's notification



This mock-up shows the Taxi driver's notification received before a ride. He can accept or deny the booking

3.1.2 API Interfaces

The application will use *OpenStreetMap* API for verify addresses validity and to show a map on the devices.

3.1.3 Hardware Interfaces

This product supports any kind of machine that has access to internet.

3.1.4 Software Interfaces

- Database Management System (DBMS):
 - Name: MySQL
 - Version: 5.7.10
 - Source: <http://dev.mysql.com/>
- Java Virtual Machine (JVM):
 - Name: JEE
 - Version: 7
 - Source: <http://www.oracle.com/>
- Operative System (OS): The product will be runnable on any OS that will support JVM.

3.1.5 Communication Interfaces

Protocol	Application	Port
TCP	HTTPS	443
TCP	DBMS	3306

3.1.6 Memory

The memory host requirements are:

- Storage memory: 4GB +
- RAM memory: 1GB +

The memory client requirements are:

- Storage memory: 256MB +
- RAM memory: 1GB +

3.2 Functional Requirements

3.2.1 [G1] Allow the registration on the platform

- [R1] Visitor must not be already registered.
- [R2] Visitor must choose an email not already used by another user.
- [R3] Visitor can only log in or register on the platform.
- [D1] Visitor must choose a valid email address.
- [D2] Visitor must choose a password of at least 8 characters and at least one digit.

3.2.2 [G2] Allow user to log-in on the platform

- [R1] Passenger must be already registered.
- [R2] Passenger must know the correct email and password.
- [R3] Wrong credentials must avoid the access.

3.2.3 [G3] Allow user to book a new trip

- [R1] Passenger must be logged in.
- [R2] Passenger must insert a starting address, a destination address, number of passengers and date of the booking.
- [R3] Passenger must confirm the booking.
- [R4] Starting address and destination address must be different.
- [D1] Date and time of booking must be in the future.
- [D2] Address must have the format: City, Address, Number.
- [D4] Cannot exist more than one booking with the same date and time.

3.2.4 [G4] Allow user to modify one of his booked trip

- [R1] User must be logged in.
- [R2] User must select one of his future booking.
- [R3] User must modify the field that he wants.
- [R5] User must confirm the updates.
- [R6] User can modify a booking until two hours before the ride
- [D1] Cannot exist more than one booking with the same date and time.

3.2.5 [G5] Allow user to delete one of his booked trip

- [R1] User must be logged in.
- [R2] User must select one of his future booking.
- [R3] User must confirm the the elimination.
- [D1] User can only delete his booking until 10 minutes before the ride.

3.2.6 [G6] Allow user to receive a cost preview for a selected route

- [R1] User must be logged in.
- [R2] User must select one of his future booking.

3.2.7 [G7] Allow user to view the EWT for a request

- [R1] User must be logged in.
- [R2] User must select one of his future booking.
- [D1] EWT can't be displayed until 10 minutes before the ride.

3.2.8 [G8] Allow user to release a feedback on a specific taxi driver

- [R1] User must be logged in.
- [R2] User must select one of his *past-booking ride* that has no feedback.
- [R3] User must have performed the ride associated to the booking.
- [D2] The feedback will be associated to the driver of that specific booking.

3.2.9 [G9] The application will notify taxi driver for a new ride

- [R1] The taxi driver must be logged in.
- [R2] The application will immediately notify taxi drivers of a *now-booking ride*.
- [R3] The application will notify taxi drivers 10 minutes before a *future-booking ride*.
- [R4] The application notifies taxi driver iff a user creates or modifies a booking.
- [D1] The notified taxi driver must be the first driver available of the queue of the booking's zone.

3.2.10 [G10] Allow taxi driver to inform the system about his availability

- [R1] The taxi driver must be logged in.
- [D1] Available times' must be in the working hours of taxi drivers.

3.2.11 [G11] Allow taxi driver to accept or deny ride's requests

- [R1] The taxi driver must be logged in.
- [D1] Taxi driver must be available in the request moment.
- [D2] taxi driver must be the first driver available of the queue of the ride's zone queue.

3.2.12 [G12] Allow administration to inspect all users' profiles

- [R1] Administrator must be logged in.

3.3 Scenarios

3.3.1 Scenario 1: Passenger future-booking case

Mary, John's wife, will arrive at Malpensa Airport next Moonday morning. John cannot go to pick her at that time, so he decides to book a taxi. John is already registered on myTaxiService platform, so he log in to book the ride.

Only three hours before the planned trip, John realize to have booked the ride from a wrong airport. Promptly open his mobile app for modify the booking (he has only one hour left to modify the booking).

Ten minutes before the ride Mario receives a confirm notification with taxi's ID that he sends to Mary.

She arrives in time at the meeting place and she arrives home very quickly with Jack, a funny and a little crazy taxi driver. At the end of the trip Mary as usual try to pay with cash but Jack says that the payment will be automatically charged to the account of her husband.

During the evening Mary and John enter again to myTaxiService portal to give a feedback to John: very good for *Codiality*, but not so good for *Driving style* about Jack.

3.3.2 Scenario 2: Payment methods and feedback releasing

Filomena, an old taxi driver, starts her day as usual but today she is particularly irritable: it rains and she didn't sleep very well.

The first trip early in the morning is with a business gentleman for about 20 minutes. At the end of the ride Filomena pick her smartphone to close the booking: the business man selected the payment by PayPal but the system gives a warning to Filomena because the man doesn't have enough money on his account.

The procedure provides that the driver has to ask the amount by cash and Filomena is very nervous about this! So begins a discussion with the passenger that, after minutes of tension, gives her the cash and get out. Immediately the business man logs on the platform and gives 1 star to the field *Cordiality* on Filomena's feedback page.

3.3.3 Scenario 3: Passenger now-booking case

Mario wants to reach his meeting but he's too late to go with public services so he decides to find an application to book a taxi really fast. He finds myTaxiService then he goes on the registration page and fill the form, when he confirms his e-mail, logs in and he books a taxi for his destination.

Because of the instantaneous need, the system advices as soon as possible John of the EWT, the estimated price and he's really happy because he could arrive in time at the meeting.

3.3.4 Scenario 4: Administration profile inspecting

Carl, an administrator of the platform wants to check the quality of the service that he's offering at his clients in the suburb zone. He types the name of the zone in the research form and starts to inspect all taxi driver's profile finding that in the suburb taxi drivers are too rude with their client, he reads all the user's comments and understand better the situation, after that he'll contact all the taxi drivers in the suburb zone and will take a discussion with them.

3.4 Domain Properties

In the context of our system we assume the truth of these conditions:

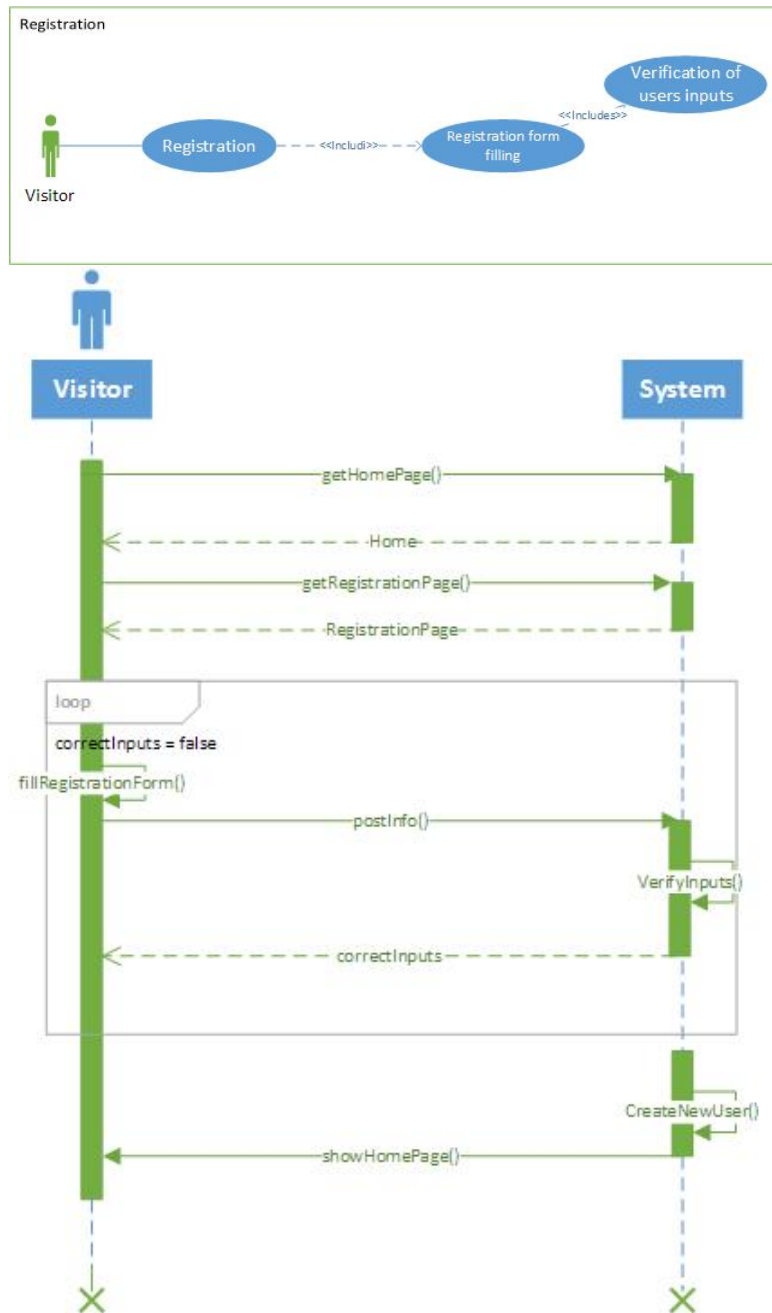
- The passenger can create several booking (also at the same time)
- The system can be used only by registered and logged in users, administrators and taxi driver.
- A user can reserve at least one seat and at most 6 seats for every ride.
- Bookings can be created until two hours before the ride.
- Bookings can be modified until two hours before the ride.
- Bookings can be cancelled until 15 minutes before the ride.
- Bookings are unique.

3.5 UML Models

3.5.1 Use case models

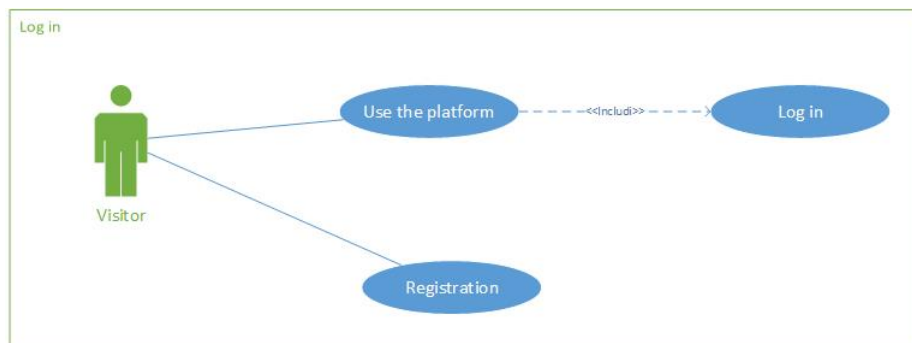
3.5.1.1 Visitor Registration

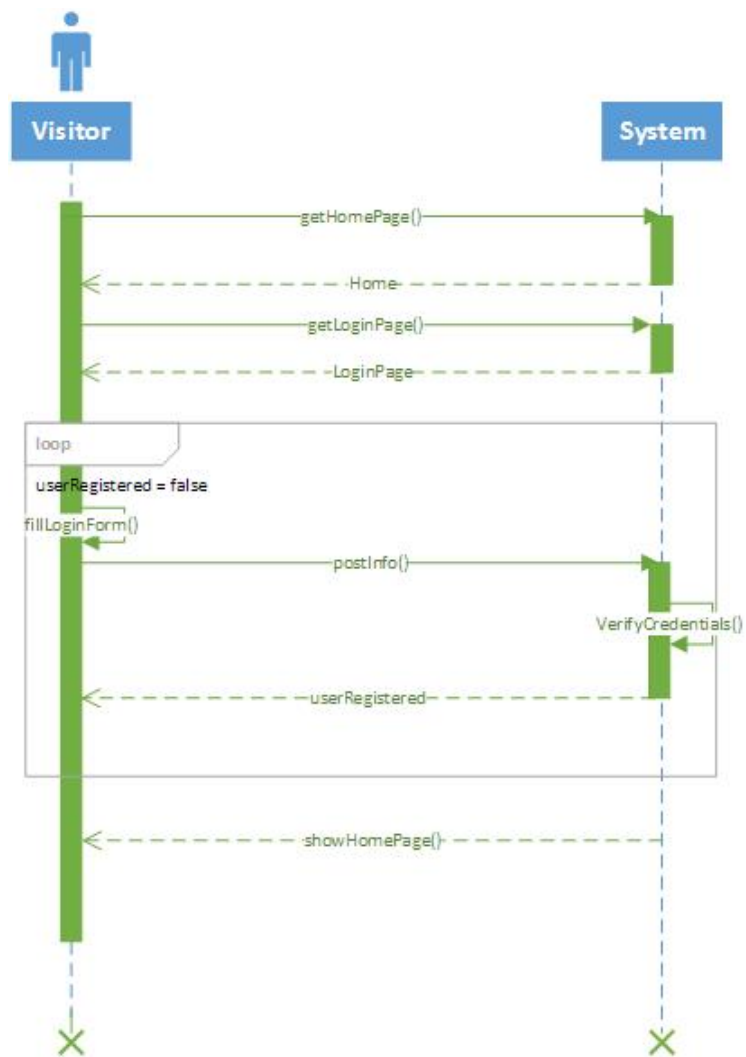
Actor	Visitor
Goal	G1
Input condition	NULL
Event flow	1. Visitor access to login page. 2. Visitor clicks on the New User button. 3. Visitor fills in at least mandatory fields. 4. Visitor clicks on submit button. 5. Application will save info with a storage system.
Output state	Visitor successfully ends registration process and he can become a User. From now on the visitor with that credentials can log in to the application and become a User start using the application.
Possible exceptions	1. Visitor doesn't fill at least mandatory fields. 2. e-mail address format is not valid 3. e-mail address already used



3.5.1.2 Visitor Login

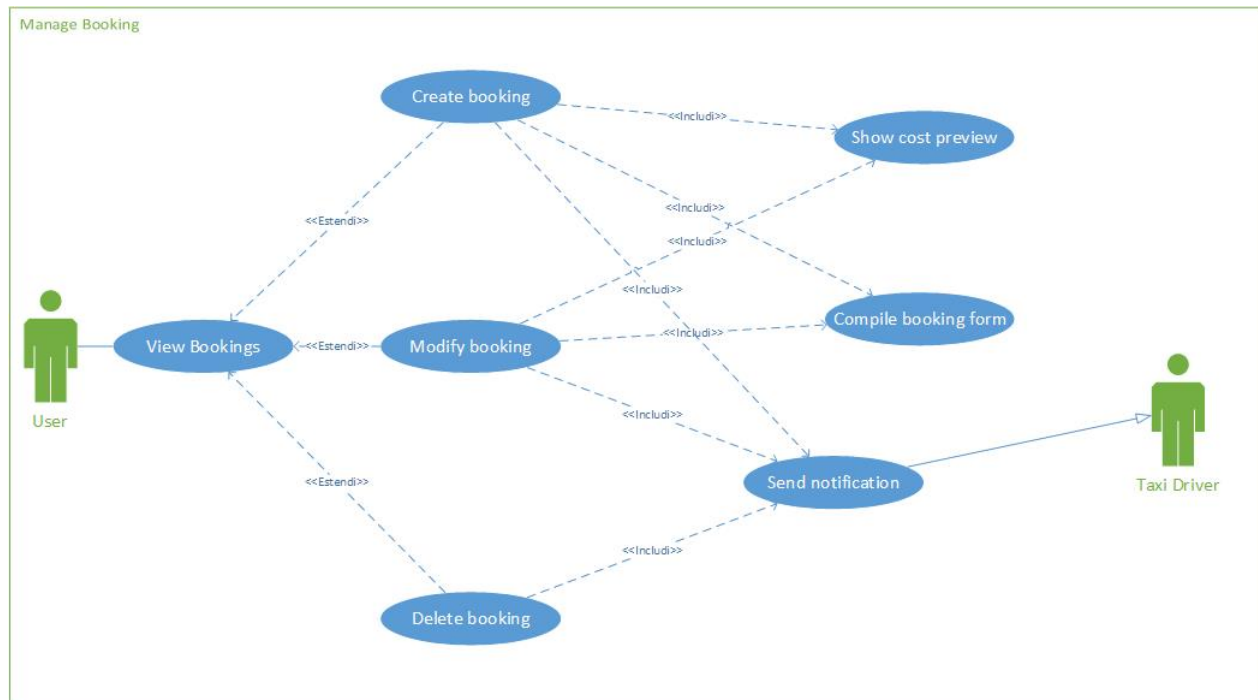
Actor	Visitor
Goal	G2
Input condition	Visitor is already registered in the platform
Event flow	1. Visitor access to login page. 2. Visitor insert is corrected registered credentials. 3. Visitor logs in and become a User
Output state	Visitor succesfully logs in can become a User. From now he can start using the application.
Possible exceptions	1. Credentials are wrong.

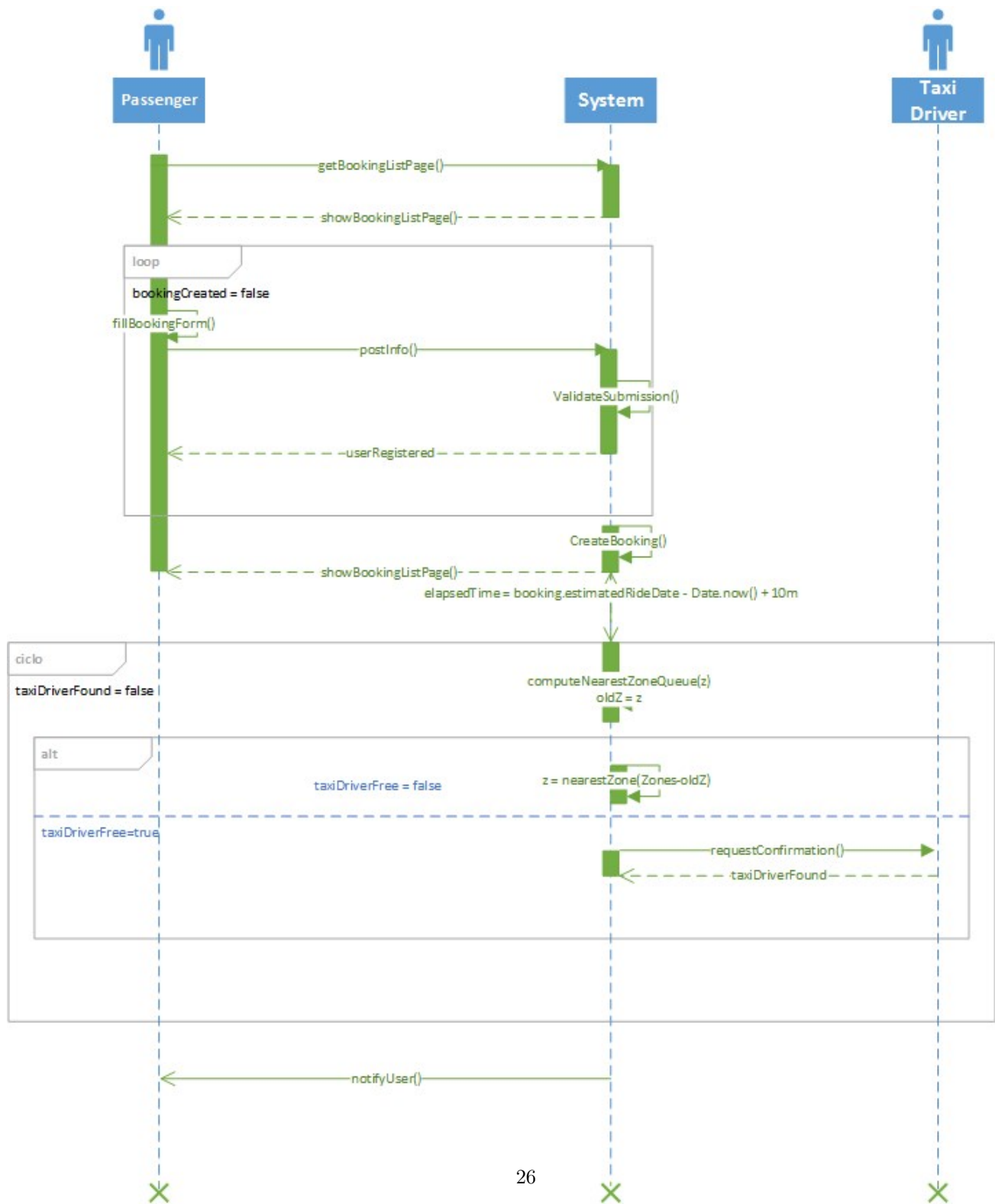




3.5.1.3 Create booking

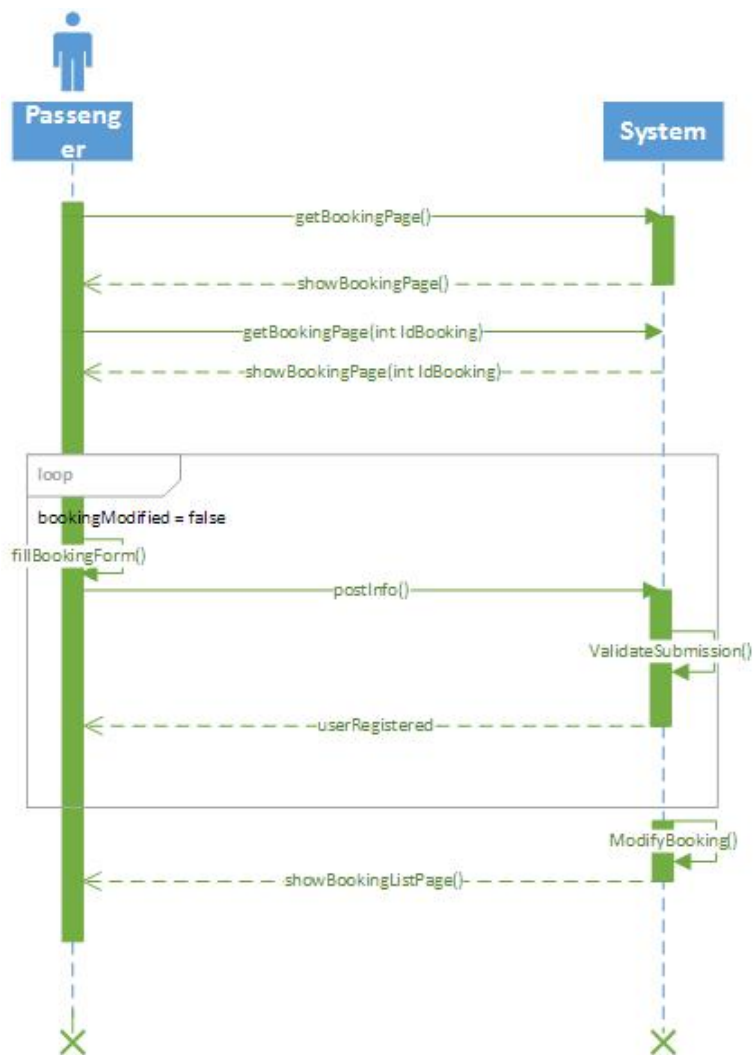
Actor	User
Goal	G3, G6, G13
Input conditions	Visitor is already registered in the platform and logged in as a User.
Event flow	<ol style="list-style-type: none">1. User access to the booking page.2. User create a new booking, inserting starting point, destination, date passengers and views the cost preview; or modify/delete an existing booking showed in a list.3. User commits the operation by clicking on the submit button.4. For the starting zone the application will compute taxi driver' s queue.5. System will assign the first taxi driver in the zone's queue.
Output state	User has successfully created / modified / deleted a booking. He's redirect in the booking list page showing a notification that inform him for the operation's success.
Possible exceptions	<ol style="list-style-type: none">1. There is an exact booking with the same: Starting point, Destination, date and passengers.2. There are no taxi driver in the queue.





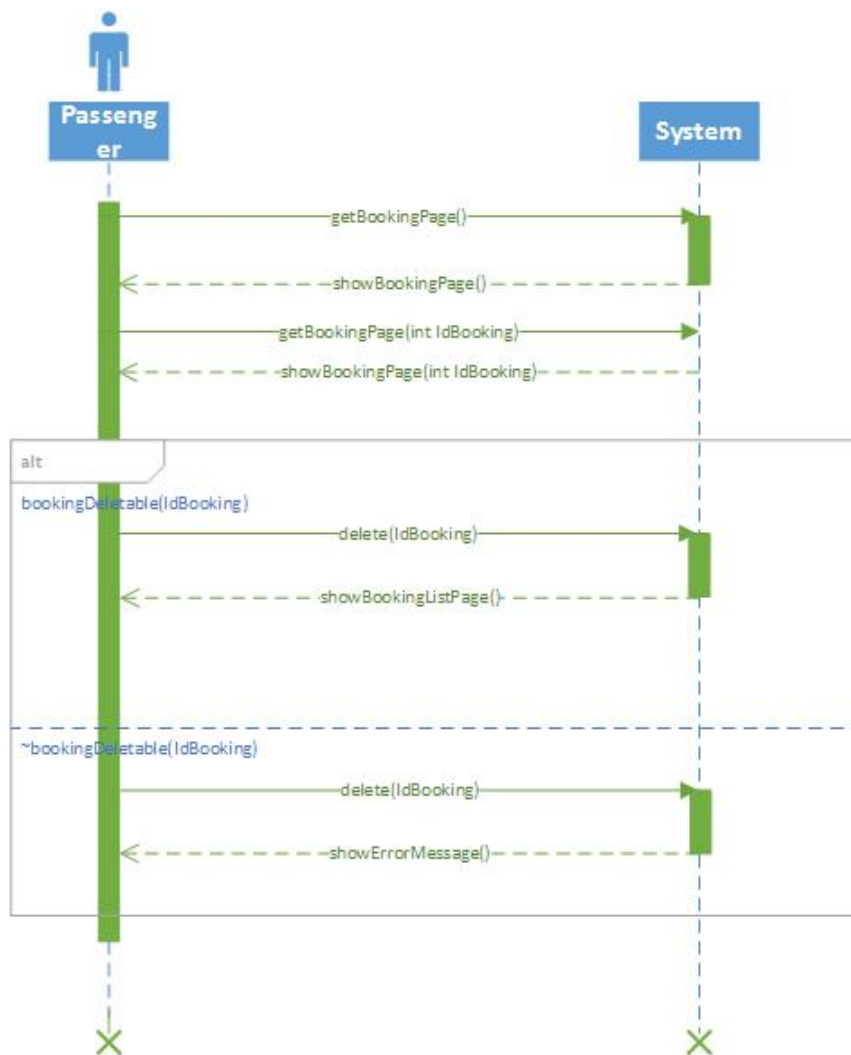
3.5.1.4 Modify booking

Actor	User
Goal	G4, G6, G13
Input conditions	Visitor is already registered in the platform and logged in as a User
Event flow	<ol style="list-style-type: none">1. User access to the booking page.2. User modify an existing booking showed in a list, editing one or more of the following fields: starting point, destination, date or passengers; then views the cost preview3. User commits the operation by clicking on the submit button.
Output state	User has succesfully modified a booking. He/she's redirect in the booking list page showing a notification that inform him for the operation's success
Possible exceptions	<ol style="list-style-type: none">1. There is an exact booking with the same: Starting point, Destination, date and passengers.



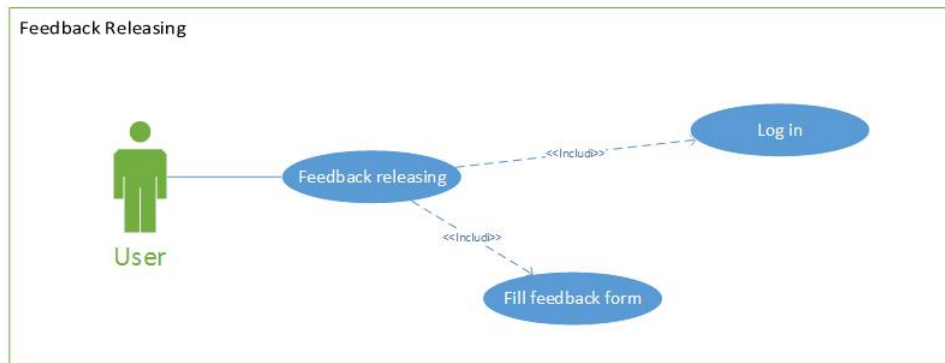
3.5.1.5 Delete booking

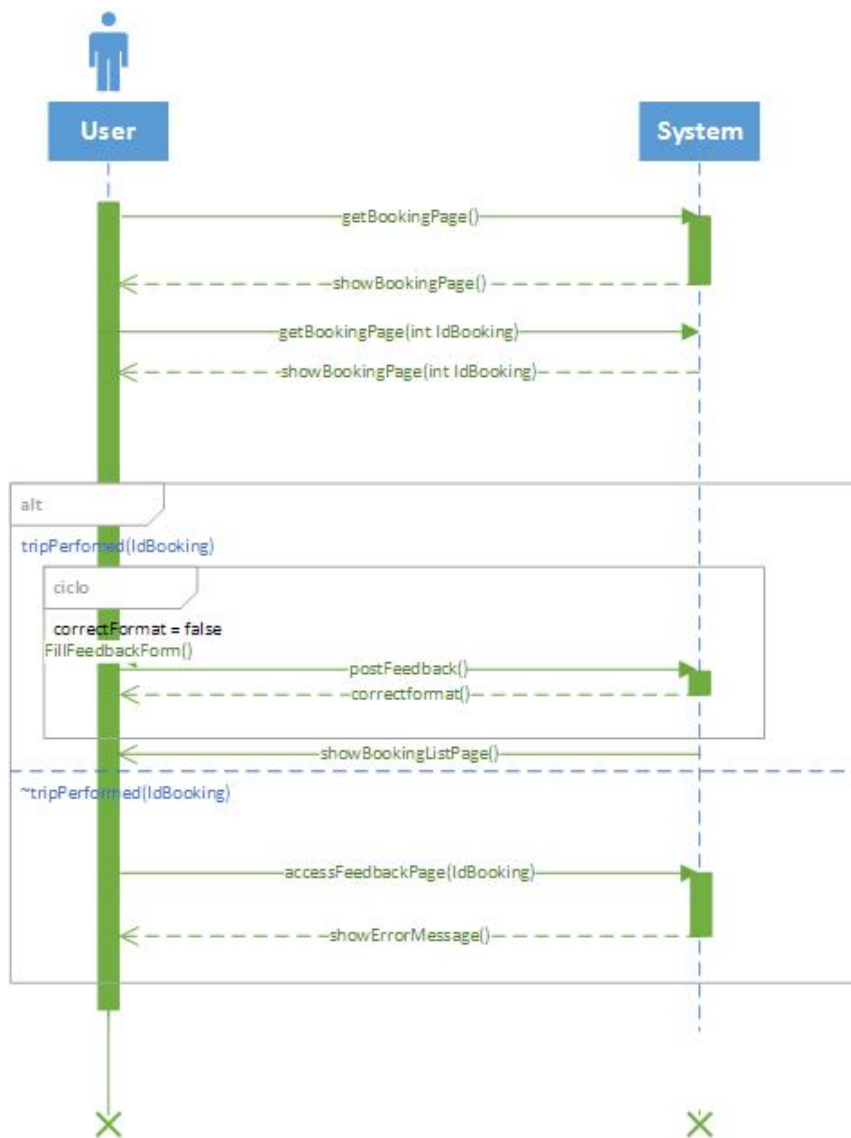
Actor	User
Goal	G5
Input conditions	Visitor is already registered in the platform and logged in as a User
Event flow	<ol style="list-style-type: none">1. User access to the booking page.2. User delete an existing booking showed in a list.3. User commits the operation by clicking on the submit button.
Output state	User has succesfully deleted a booking. He/she's redirect in the booking list page showing a notification that inform him for the operation's success
Possible exceptions	



3.5.1.6 Feedback Releasing

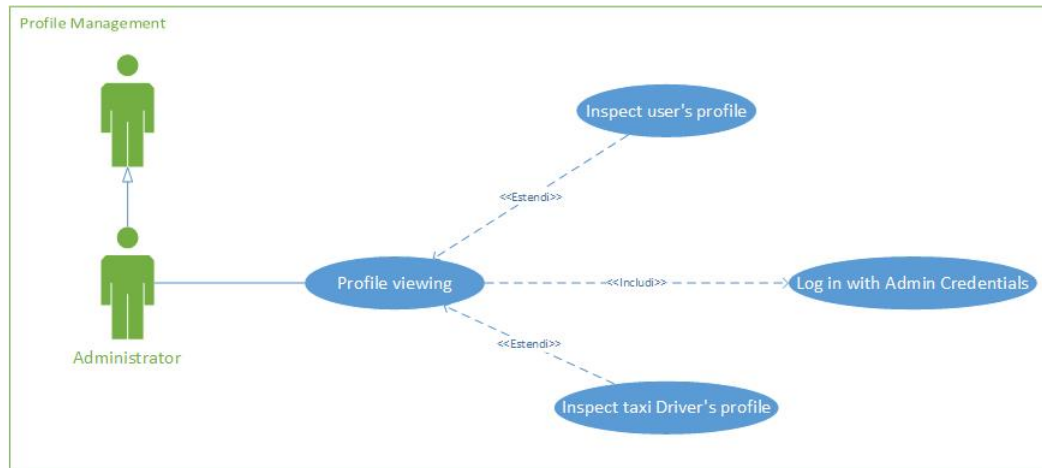
Actor	User
Goal	G8
Input conditions	Visitor is already registered in the platform and logged in as a User
Event flow	<ol style="list-style-type: none">1. User access to the booking page.2. User select a booking that he just used to travel.3. User fill the form concerning taxi driver that was involved in that trip, releasing: Driver style evaluation, Cordiality evaluation and a comment.
Output state	User has succesfully created / modified / deleted a booking. He/she's redirect in the booking list page showing a notification that inform him for the operation's success
Possible exceptions	<ol style="list-style-type: none">1. The format of the comment is not accepted by the platform (e.g. too long)

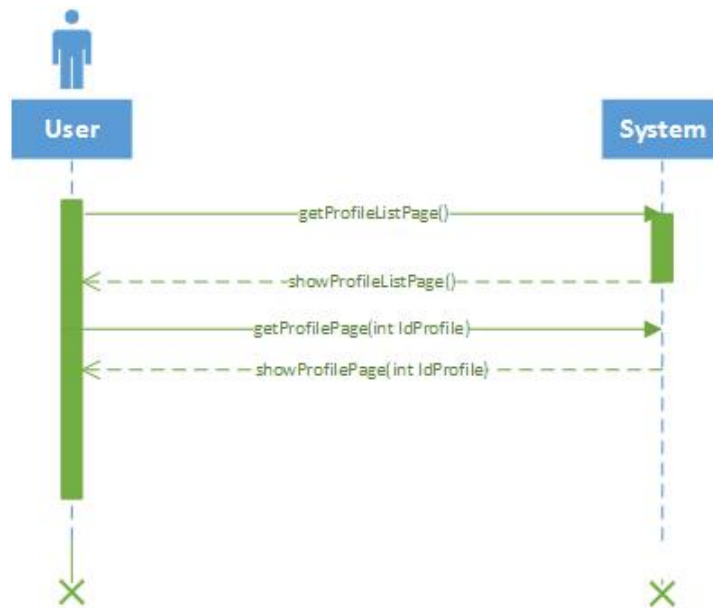




3.5.1.7 Profile Management

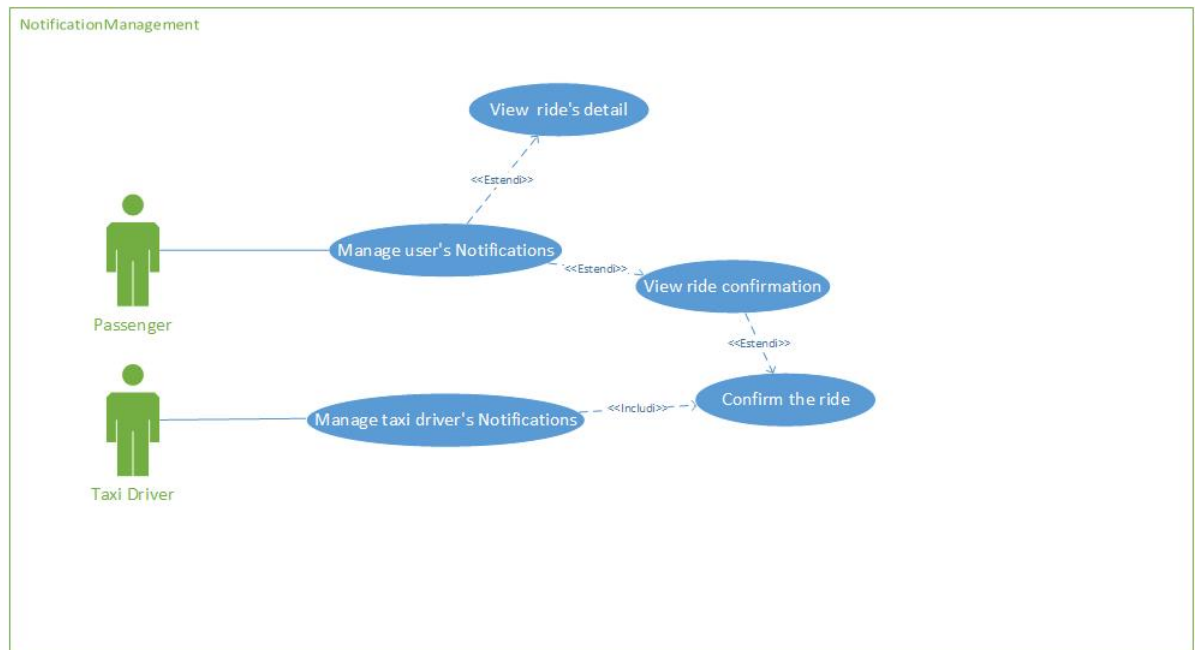
Actor	User
Goal	G12
Input conditions	Visitor is already registered in the platform and logged in as an Administrator or he he's on a taxi driver terminal.
Event flow	<ol style="list-style-type: none"> 1. Clicking on the profile button in the navigation bar taxi driver has access to his/her profile where he/she can find his/her ratings and passenger's comments . 2. Admin has access to the list of users page and select the interested one, could use the search module. 3. Clicking on a row (one row per user) he/she has access to the profile page of that specific user.
Output state	Admin / Taxi driver has successfully access to a profile page.
Possible exceptions	NULL.





3.5.1.8 Notification Management

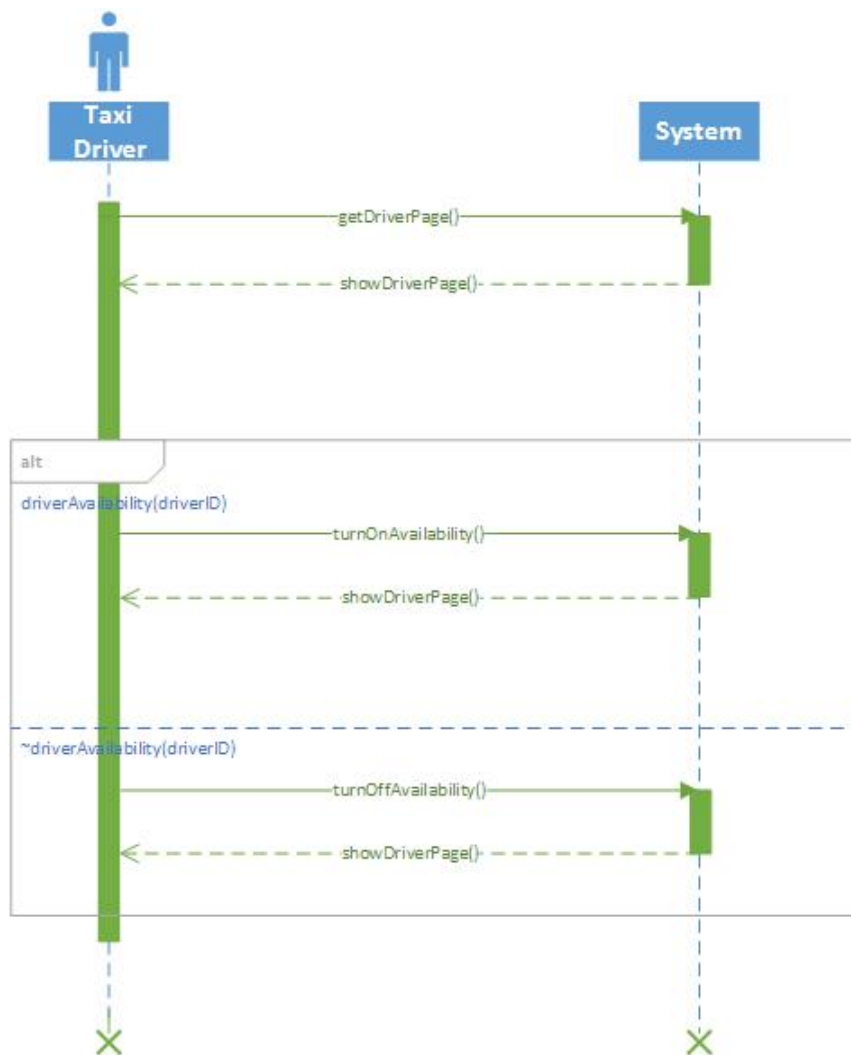
Actor	User, Taxi Driver
Goal	G7, G9
Input conditions	Visitor is already registered in the platform and logged in as a user or he/she's on a taxi driver terminal.
Event flow	<p>1. 10 minutes before the ride start time the user receives a message that reminds him the generic info about the trip and EWT .</p> <p>2. When the taxi driver receives a trip request from the platform, he/she is notified about the info and choose to accept or to deny the job.</p>
Output state	Admin / Taxi driver has successfully access to a profile page.
Possible exceptions	NULL.



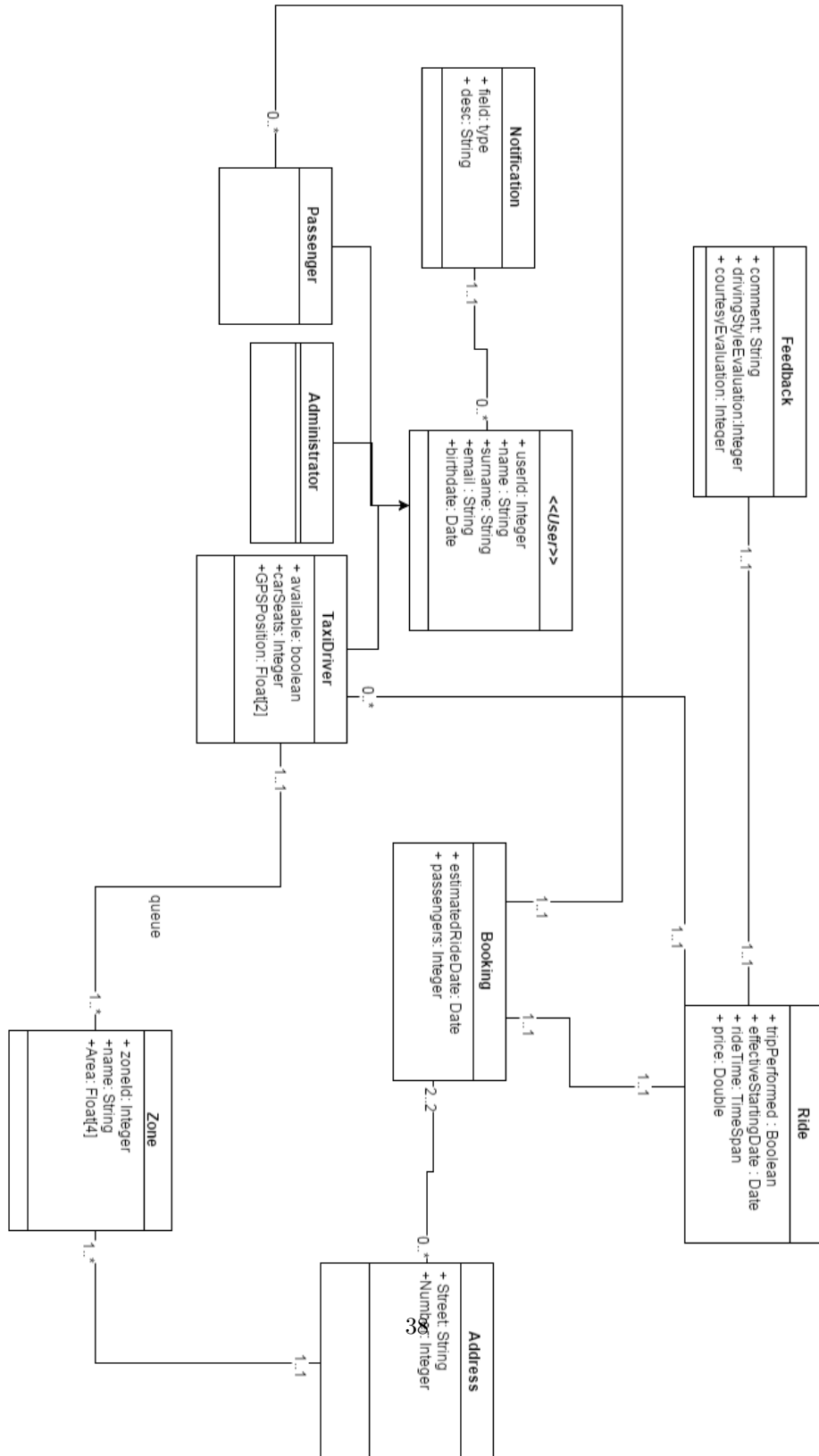
3.5.1.9 Taxi driver availability management

Actor	Taxi Driver
Goal	G10
Input conditions	Taxi driver is on his terminal.
Event flow	Taxi driver have access to his/her profile by clicking on his/her name on the navigation bar 2. He tick the status from busy to free
Output state	The taxi driver is now available for the platform and he would be called for a trip
Possible exceptions	NULL.

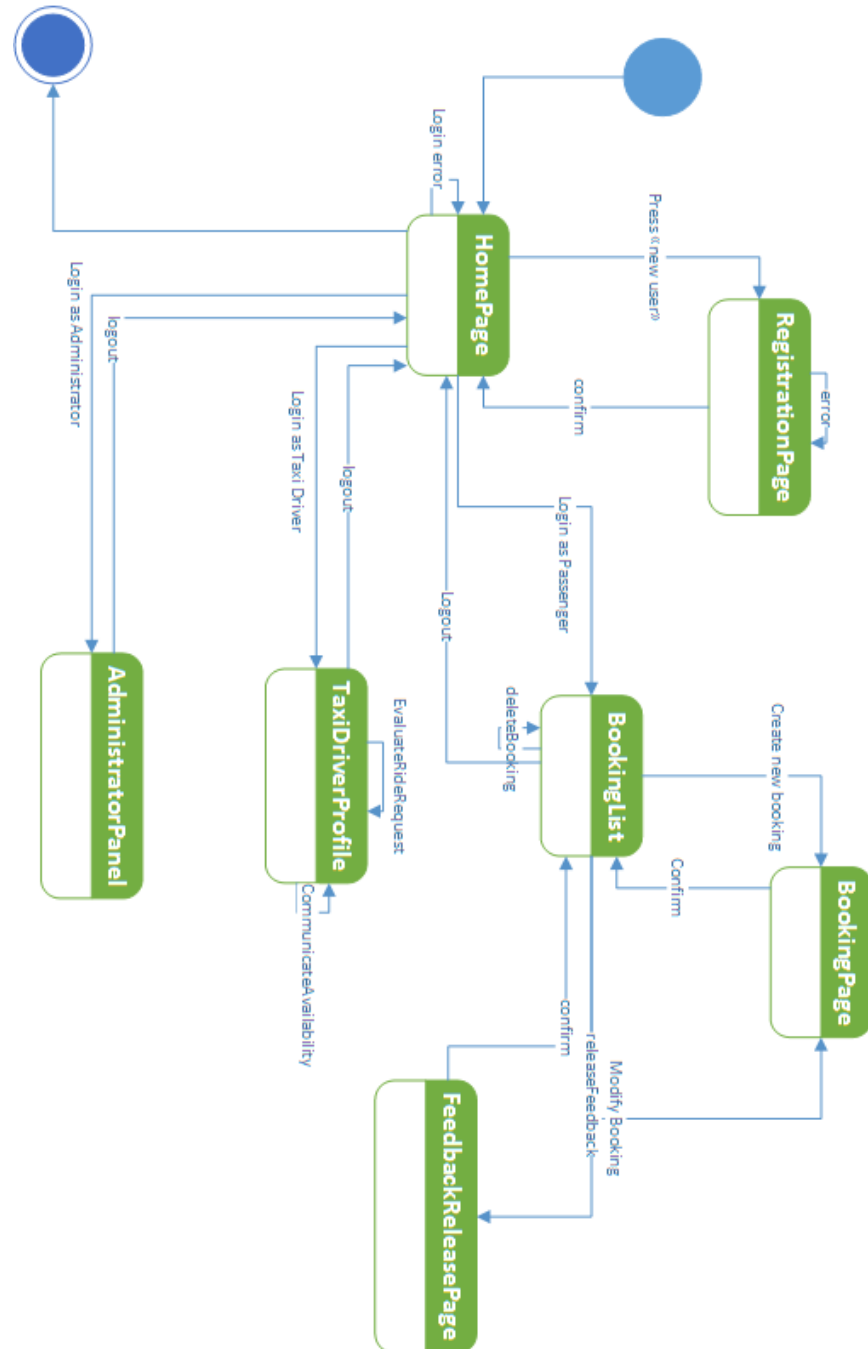




3.5.2 Class Diagram



3.5.3 State Chart Diagram



4 Alloy

In this paragraph we've reported the alloy code used for the analysis and some examples of worlds generated with the analyzer. In this paragraph is reported the alloy code used for modelling analysis of our application and samples of worlds generated by the Analyzer


```

/*****
BASIC SIGNATURES
*****/

sig Date{} //Date data structure, with time and timezone
sig Email{}
sig Integer{}

/*****
SIGNATURES
*****/

//abstract class of user
abstract sig User{
    notificationList: set Notification,
    email: one Email
}

sig Passenger extends User{
    bookingList: set Booking,
    releasedFeedback: set Feedback
}

sig TaxiDriver extends User{
    receivedFeedbacks: set Feedback,
    associatedZone: one Zone,
    carSeats: one Integer
}

sig Administrator extends User{
}

sig Booking{
    startingAddress: one Address,
    destinationAddress: one Address,
    estimatedRideDate: one Date,
    ride: one Ride,
    driver: lone TaxiDriver,
    passenger: one Passenger
}

sig Ride{
    associatedBooking: one Booking
}

sig Address{
    zone: one Zone
}

sig Zone{
    driverList: set TaxiDriver,
    addressList: set Address
}

```

```

sig Notification{
    notifiedUser: one User
}

sig Feedback{
    associatedRide: one Ride
}

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

fact noDuplicateBooking{
    no disj b1,b2: Booking | (b1.startingAddress=b2.startingAddress) and
    (b1.destinationAddress=b2.destinationAddress) and
    (b1.passenger=b2.passenger) and (b1.estimatedRideDate=b2.estimatedRideDate)
}

fact noDuplicateUser{
    no disj u1,u2: User | (u1.email=u2.email)
}

fact userConsistence
{
    all u1, u2: User | u1.email = u2.email iff u1.notificationList = u2.notificationList
}

fact bookingConsistence
{
    //A booking is on the user's booking list iff he is the creator of that booking
    all p: Passenger, b: Booking | b in p.bookingList iff b.passenger = p
    //A Booking must have starting address and destination address different
    all b: Booking | b.startingAddress != b.destinationAddress
}

fact feedbackStructure
{
    all f:Feedback | some r:Ride | r = f.associatedRide
}

fact feedbackConsistence
{
    //A feedback can be released iff the user has performed his ride
    all f:Feedback | some b: Booking | f.associatedRide = b.ride
    //iff some r:Ride | r.associatedBooking = b
    //All feedbacks must have a releaser and a receiver that are associated to a
    //booking in wich one is the passenger and one is the driver
}

fact carSeatsConstraint
{
    all t: TaxiDriver | #t.carSeats > 0 and #t.carSeats <= 6
}

```

```

/*****
  PREDICATES
*****/

pred show()
{
  #Passenger > 2
  #TaxiDriver > 1
  #Feedback > 1
  #Ride > 1
}
//Booking inserting in the list bookingList of Passenger (bookingList)
pred bookingInserting(b: Booking, bookingList: set Booking)
{
  b not in bookingList implies bookingList = bookingList + b
}

//Booking deleting
pred bookingDeleting(b: Booking, bookingList: set Booking)
{
  b in bookingList implies bookingList = bookingList - b
}

//Feedback inserting
pred feedbackReleasing(feedbackReceivedByTaxiDriver: set Feedback, r: Ride, b: Booking, f: Feedback)
{
  f not in feedbackReceivedByTaxiDriver and b = r.associatedBooking
  implies feedbackReceivedByTaxiDriver = feedbackReceivedByTaxiDriver + f
}

/*****
  RUNS
*****/
run show for 5

run bookingInserting for 2

run bookingDeleting for 2

run feedbackReleasing for 2

```

```

/*****
ASSERTIONS
*****/

assert noBookingDuplication
{
  no disj b1, b2 : Booking | (b1.startingAddress=b2.startingAddress) and
  (b1.destinationAddress=b2.destinationAddress) and (b1.passenger=b2.passenger) and
  (b1.estimatedRideDate=b2.estimatedRideDate)
}

check noBookingDuplication for 5

assert noUsersWithTheSameNotificationList
{
  no disj u1, u2: User | u1.notificationList = u2.notificationList
}

check noUsersWithTheSameNotificationList for 5

assert bookingCreation
{
  all b: Booking, p1: Passenger | (b not in p1.bookingList) and (bookingInserting[b,p1.bookingList]) implies (b in p1.bookingList)
}
check bookingCreation for 5

assert bookingDeletion
{
  all b:Booking, p1: Passenger | (b in p1.bookingList) and (bookingDeleting[b, p1.bookingList]) implies (b not in p1.bookingList)
}
check bookingDeletion for 5

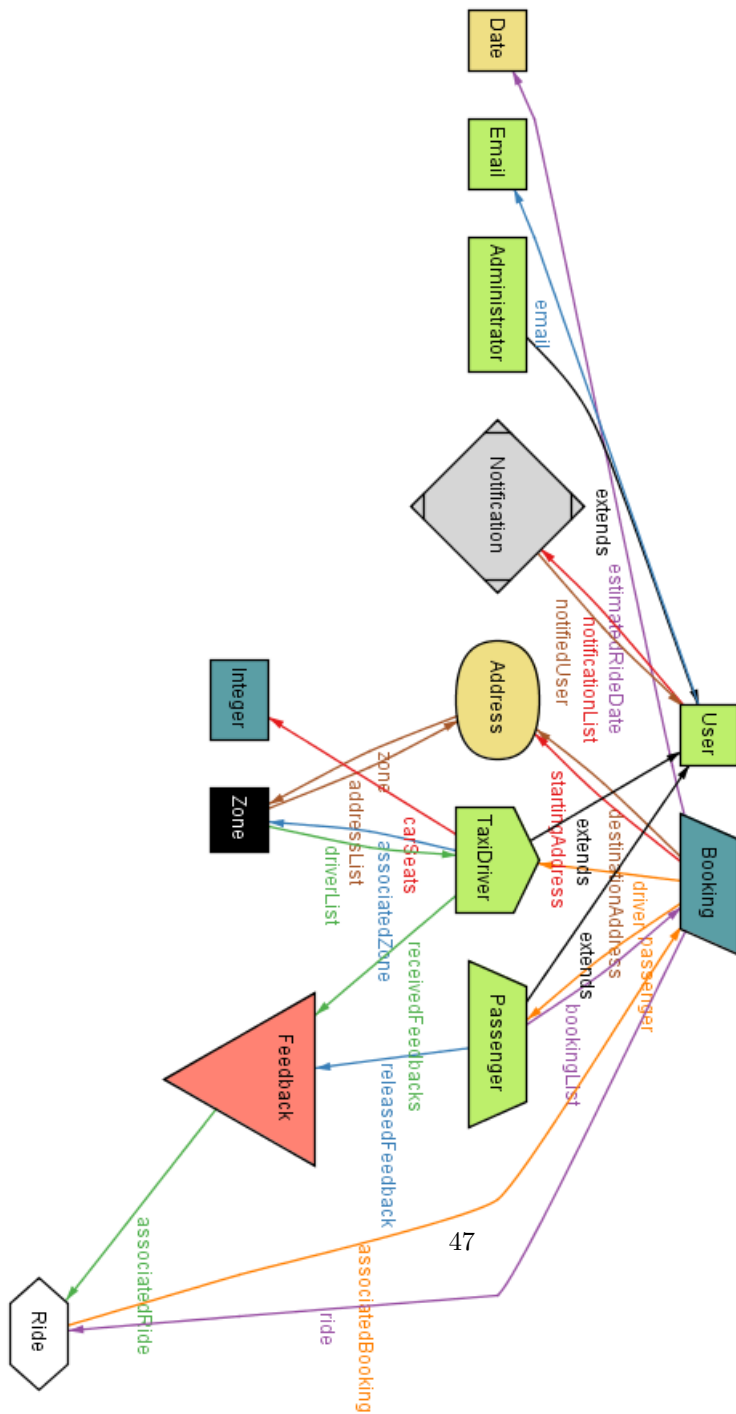
//There not exists a feedback for a specific booking that doesn't have assigned an existing ride
assert feedbackExistence
{
  all f:Feedback | one r:Ride | f.associatedRide = r
}
check feedbackExistence for 5

```

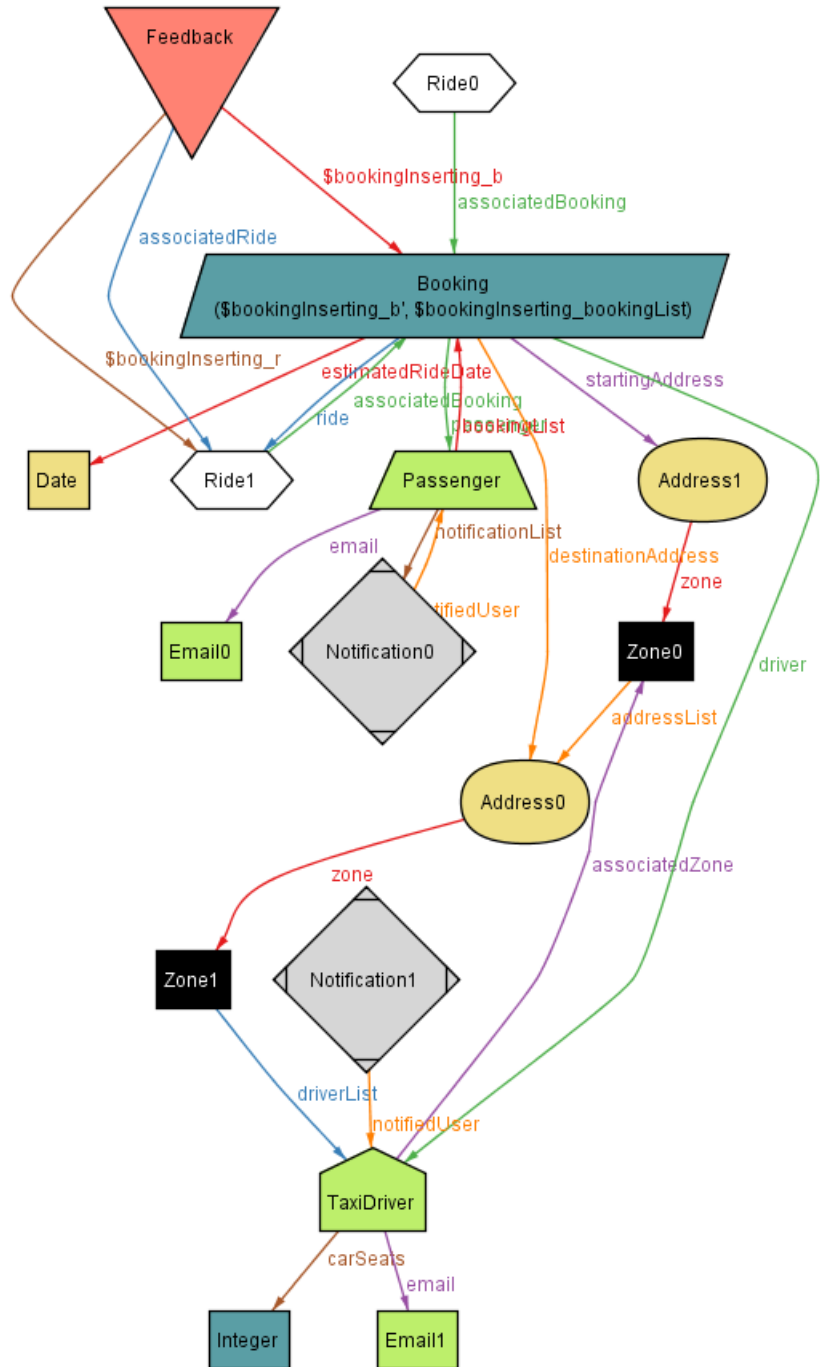
4.1 Alloy calculator results

```
#1: Instance found. show is consistent.  
#2: Instance found. bookingInserting is consistent.  
#3: Instance found. bookingDeleting is consistent.  
#4: Instance found. feedbackReleasing is consistent.  
#5: No counterexample found. noBookingDuplication may be valid.  
#6: No counterexample found. noUsersWithTheSameNotificationList may be valid.  
#7: No counterexample found. bookingCreation may be valid.  
#8: No counterexample found. bookingDeletion may be valid.  
#9: No counterexample found. feedbackExistence may be valid.
```

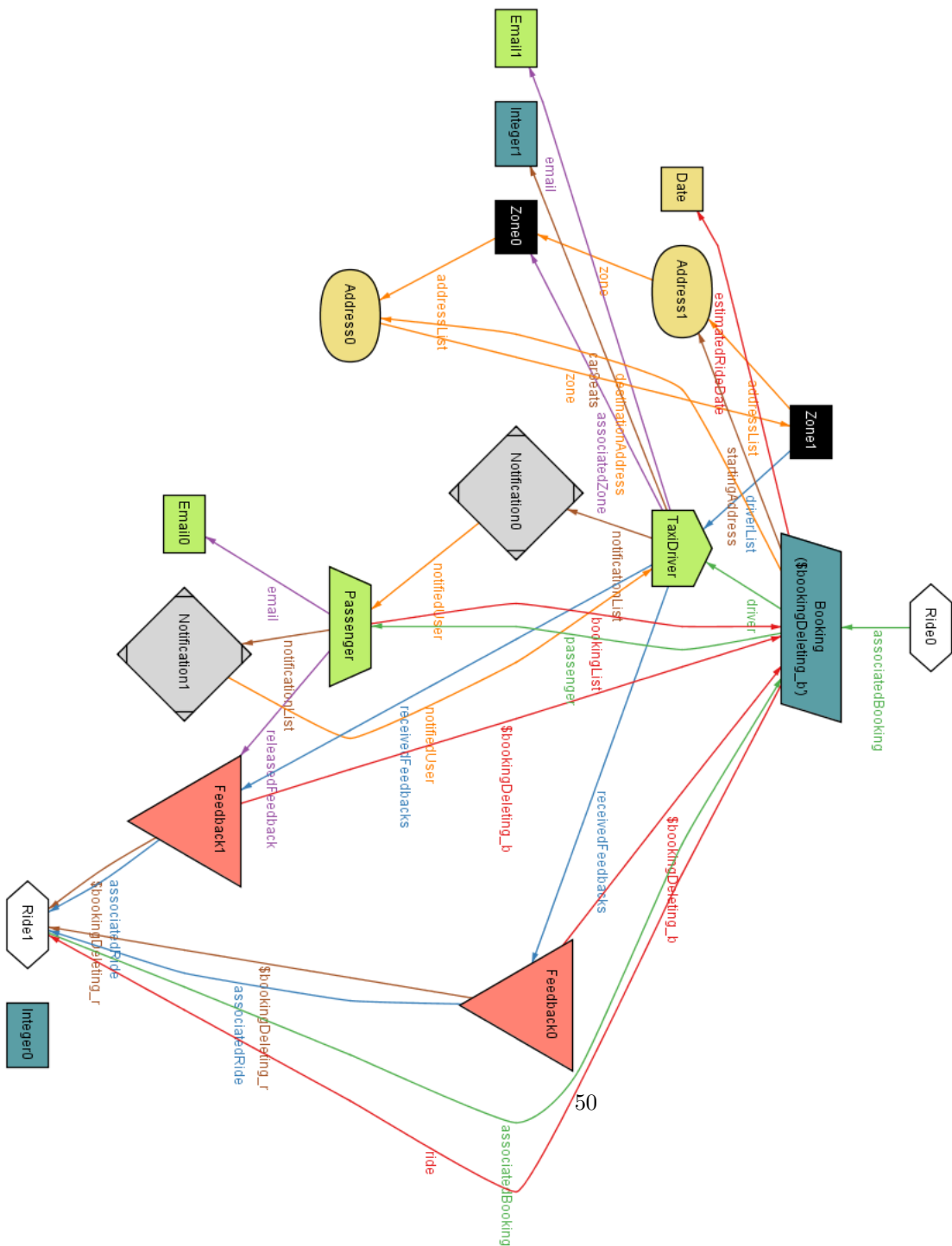

4.2 Metamodel



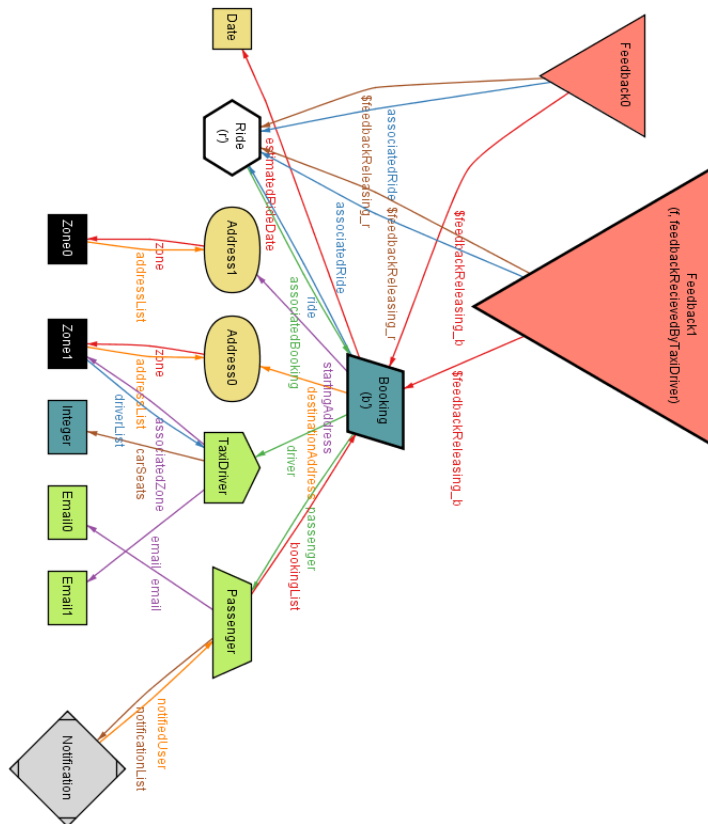
4.3 Booking Insertion



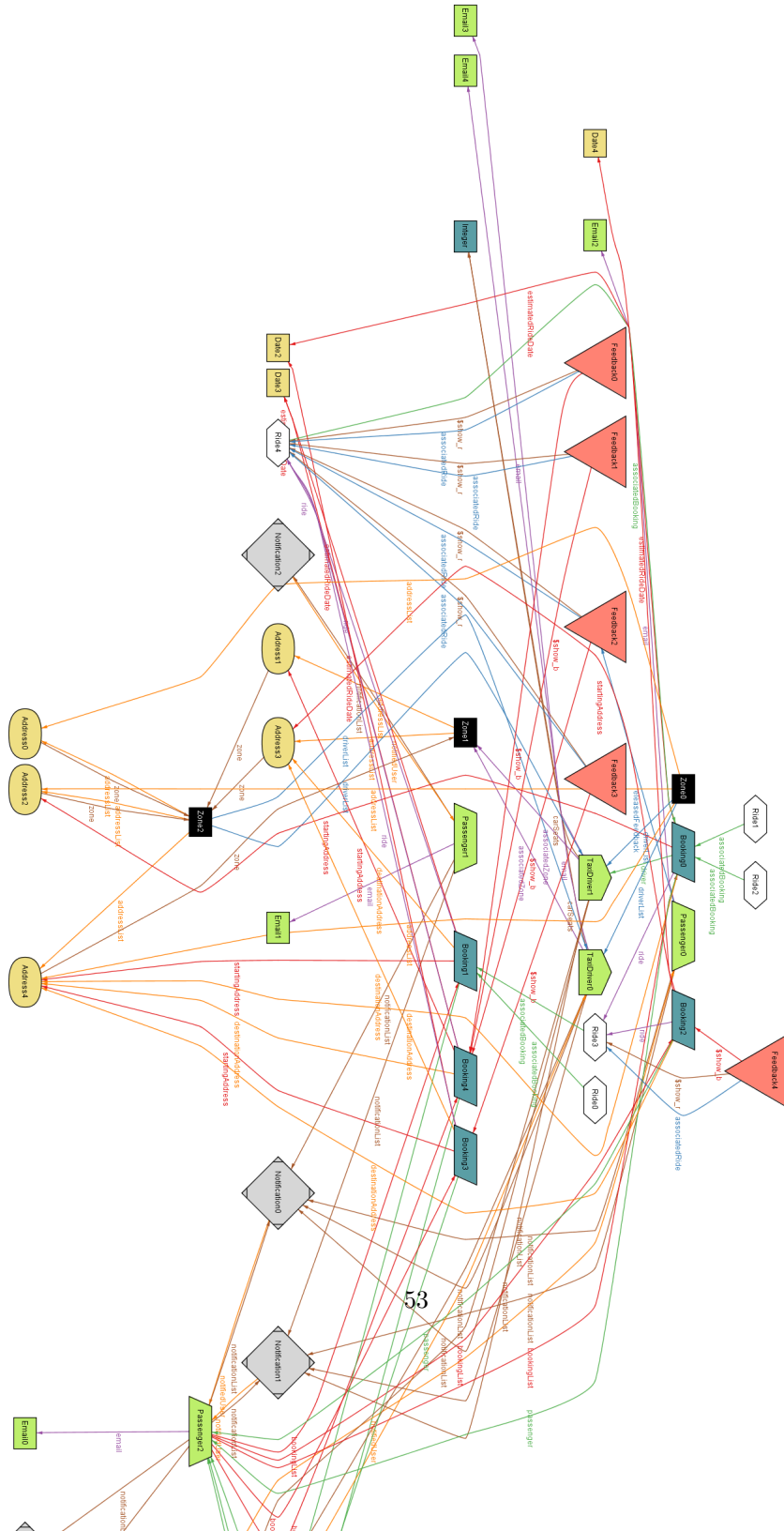
4.4 Booking Deletion



4.5 Feedback Releasing



4.6 World



5 Appendix

5.1 Software and tool used

In order to redact this document we have used the following softwares:

- – Name: Visio
– Version: 2013
– Scope: Draw all UML charts
– Reference: <https://products.office.com/it-it/visio/>
- – Name: Alloy Analyzer
– Version: 4.2
– Scope: Alloy modeling
– Reference: <http://alloy.mit.edu/alloy/>
- – Name: moqups
– Version: -
– Scope: Mockup creation
– Reference: <https://moqups.com/>

5.2 Working Hours

The RASD document was written in more or less 50 hours divided between the two elements of the group:

- Massimiliano Paci: 25 hours
- Giovanni Patruno: 25 hours

Each element of the group didn't work on specific sections but worked on the whole document