



POLITECNICO MILANO 1863

Politecnico di Milano
A.A. 2015/2016
Software Engineering 2: “*myTaxyService*”

Alessandro Baldassari (mat. 841561)
Alberto Bendin (mat. 841734)
Francesco Giarola (mat. 840554)

November 6, 2015

Contents

	Page
1 Introduction	1
1.1 Purpose	1
1.2 Scope	1
1.3 Identifying Stakeholders	1
1.4 Identifying Actors	1
1.5 Goals	2
1.6 Proposed system.....	3
1.7 Definitions, acronyms, and abbreviations.....	3
1.7.1 Definitions	3
1.7.2 Acronyms	3
1.7.3 Abbreviations	4
1.8 Reference Documents	4
1.9 Overview	4
2 Overall description.....	5
2.1 Product perspective	5
2.2 Product functions	5
2.3 User characteristics.....	5
2.4 Constraints.....	5
2.4.1 Regulatory Policies & Safety and security	5
2.4.2 Hardware limitations	5
2.4.3 Interfaces to other applications	5
2.4.4 Parallel operation	5
2.5 Assumptions and dependencies.....	6
3 Specific requirements.....	6
3.1 External Interface Requirements	6
3.1.1 User Interfaces	6
3.1.1.1 Sign-up	7
3.1.1.2 Login.....	8
3.1.1.3 Call a taxi	9
3.1.1.4 Plan a ride.....	10
3.1.1.5 Your reservations.....	11
3.1.1.6 Your profile.....	12
3.1.1.7 Manage jobs	13
3.1.1.8 Administrator panel	14
3.1.2 Hardware Interfaces.....	15
3.1.3 Communication Interfaces.....	15
3.2 Functional Requirements.....	15
3.2.1 [G1] The system has to guarantee to the user the possibility to access the service either through a web application or a mobile application.	15

3.2.2	[G2] The system has to guarantee a fair management of the queues in each taxi zone.	15
3.2.3	[G3] The system has to assign each taxi to the correct taxi zone using the GPS coordinates that receives from each taxi.	15
3.2.4	[G4] The system has to allocate a taxi for each request or reservation.	15
3.2.5	[G5] The system has to notify the users, both passengers and taxi drivers, about updates on taxi requests and reservations in which they are involved.	16
3.2.6	[G6] The system has to offer public APIs to enable the possibility to develop additional services on top of the basic ones.	16
3.2.7	[G7] The passenger shall be able to sign up to the service.	16
3.2.8	[G8] The passenger shall be able to log in to the service.	16
3.2.9	[G9] The passenger shall be able to request a taxi.	16
3.2.10	[G10] The passenger shall be able to delete a taxi request.	17
3.2.11	[G11] The passenger shall be able to create a reservation for a taxi ride.	17
3.2.12	[G12] The passenger shall be able to modify or delete a taxi reservation.	17
3.2.13	[G13] The passenger shall be able to enable taxi sharing option. .	17
3.2.14	[G14] The passenger shall be able to see historical data on his taxi rides.	17
3.2.15	[G15] Taxi drivers shall be able to log in to the service.	17
3.2.16	[G16] Taxi drivers must inform the system about their availability. .	18
3.2.17	[G17] Taxi drivers must confirm if they are going to take care of a certain call.	18
3.2.18	[G18] The administrator shall be able to add, edit and delete taxi drivers in the system.	18
3.2.19	[G19] The administrator shall be able to manage and supervise the operation of the whole system, including the real-time situation of all the queues and of the taxis.	18
3.3	The world and the machine.	18
3.4	Scenarios	19
3.4.1	Scenario 1	19
3.4.2	Scenario 2	19
3.4.3	Scenario 3	19
3.4.4	Scenario 4	19
3.4.5	Scenario 5	20
3.5	UML Models.	21
3.5.1	Use Case	21
3.5.1.1	Guest registers to <i>myTaxiService</i>	22
3.5.1.2	Guest logs in to <i>myTaxiService</i>	24
3.5.1.3	Passenger requests a taxi.	26
3.5.1.4	Passenger cancels a request for a taxi	28
3.5.1.5	Passenger creates a reservation for a taxi	29
3.5.1.6	Passenger edits or deletes a reservation for a taxi	30
3.5.1.7	Passenger reserves a ride with shared option	32
3.5.1.8	Passenger checks the history of his/her own requests	33
3.5.1.9	Taxi driver logs in to <i>myTaxiService</i>	34
3.5.1.10	Taxi driver sets his/her availability with the mobile application	35

	3.5.1.11	Taxi driver accepts or rejects a job with the mobile application	36
	3.5.1.12	System administrator manages the drivers list	38
	3.5.1.13	System administrator monitors the situation from the control center view	40
	3.5.2	Class Diagrams	41
	3.5.3	State Machine Diagrams	42
	3.5.3.1	Ride creation	42
	3.5.3.2	Modification of the request	42
	3.5.3.3	Cancellation of the request	43
3.6		Non Functional Requirements	44
	3.6.1	Performance Requirements	44
	3.6.2	Software System Attributes	44
	3.6.2.1	Availability & Reliability	44
	3.6.2.2	Security	44
	3.6.2.3	Maintainability	44
4		Alloy	45
5		Appendix	50
	5.1	Software and tools used	50
	5.2	Hours of work	50

1 Introduction

1.1 Purpose

The purpose of this document is to describe in a complete and sound way the application that will be developed and the domain in which it will run.

The intended audience for this document are the developers and programmers who have to implement the application, system and requirement analysts who want to integrate *myTaxiService* with their system or software, testers who have to determine whether the requirements have been satisfied in the application implementation, projects managers who have to plan, estimate and control the analysis and development processes and finally the users themselves. This document could be used as a contractual agreement between the costumer and the entity who develops the application.

1.2 Scope

myTaxiService is a new web and mobile application conceived to provide an immediate and user-friendly access to the taxi service of a large city; it aims at an overall improvement of the quality of the service offered.

This optimization is obtained thanks to the real-time interaction and feedback of all the parties involved in the service: taxi passengers can choose and book the ride, and the system will forward the request to the nearest available taxi drivers who can decide to take over the call; in this case the system will notify the client with the code of the incoming taxi and the waiting-time. The system guarantees a fair management of taxi queues. In particular, the city is divided in taxi zones and each zone is associated with its taxi queue. The system automatically computes the distribution of taxis in the various zones based on the GPS information it receives from each taxi. When a taxi is available, its identifier is stored in the queue of taxis in the corresponding zone. When a request arrives from a certain zone, the system forwards it to the taxis in the corresponding zone according to their order in the queue. If the taxi confirms, then the system will send a confirmation to the passenger. If not, then the system will forward the request to the second in queue and will move the first taxi in the last position of the queue. Additional features of the application are the possibility for the passengers to reserve a ride with an advance of at least two hours, choosing the origin and destination, and the option to possibly share the ride with someone else, thus dividing the cost of the service. The system confirms the reservation to the user and allocates a taxi to the request 10 minutes before the meeting time with the user. If more people are willing to share a ride from the same zone going in the same direction, then the system arranges the route for the taxi driver and defines the fare for every passenger informing all the users involved.

1.3 Identifying Stakeholders

The main direct stakeholders for this project are the government of the city and the taxi company which together have promoted the renewal of the software that manages the taxi service in the city. Of course once the system will be up and running the final stakeholders will be the users of the service that will provide an essential feedback on the new system.

1.4 Identifying Actors

- Clients: are the final users the taxi service is offered to. They can book the ride choosing among different options, for instance date and time, origin and destination locations and the possibility of sharing the trip with other customers.
- Taxi Drivers: represent the other category of users of the application, they can accept a call for a service or turn it down, thus allowing the whole system to be synchronized,

fast and efficient; moreover the system keeps the coordinates of the taxis automatically updated.

- Administrator: is the system manager. He supervises the city zones and their taxi queue. He is able to manage the list of taxi drivers in the system.

1.5 Goals

System goals of *myTaxiService* application:

- [G1] The system has to guarantee to the user the possibility to access the service either through a web application or a mobile application.
- [G2] The system has to guarantee a fair management of the queues in each taxi zone.
- [G3] The system has to assign each taxi to the correct taxi zone using the GPS coordinates that receives from each taxi.
- [G4] The system has to allocate a taxi for each request or reservation.
- [G5] The system has to notify the users, both passengers and taxi drivers, about updates on taxi requests and reservations in which they are involved.
- [G6] The system has to offer public APIs to enable the possibility to develop additional services on top of the basic ones.

List of the goals of *myTaxiService* application for taxi passengers:

- [G7] The passenger shall be able to sign up to the service.
- [G8] The passenger shall be able to log in to the service.
- [G9] The passenger shall be able to request a taxi.
- [G10] The passenger shall be able to delete a taxi request.
- [G11] The passenger shall be able to create a reservation for a taxi ride.
- [G12] The passenger shall be able to modify and delete a taxi reservation.
- [G13] The passenger shall be able to enable taxi sharing option.
- [G14] The passenger shall be able to see historical data on his taxi rides.

List of the goals of *myTaxiService* application for taxi drivers:

- [G15] Taxi drivers shall be able to log in to the service.
- [G16] Taxi drivers must inform the system about their availability.
- [G17] Taxi drivers must confirm if they are going to take care of a certain call.

List of the goals of *myTaxiService* application for the administrator:

- [G18] The administrator shall be able to add, edit and delete taxi drivers in the system.
- [G19] The administrator shall be able to manage and supervise the operation of the whole system, including the real-time situation of all the queues and of the taxis.

1.6 Proposed system

The enterprise web application is going to be developed from scratch, and will also provide the counterpart mobile version for all the main smartphone OSs on the market nowadays. It will be composed of a server, which runs the business logic, generates dynamic web pages and accesses the DBMS and on the other side there will be several clients who interact with the server using a web browser or the mobile application.

1.7 Definitions, acronyms, and abbreviations

1.7.1 Definitions

- Client (or Customer, Taxi passenger): is the user of the application that wants to use the taxi service.
- Taxi driver (or Taxi owner, employee): is the user of the application that together with the back-end system makes the service functional and constantly updated, s/he controls the work which is assigned to herself/himself accepting or rejecting the proposals of clients that the system forwards.
- Guest (or Visitor): is a non-registered user.
- Administrator: manages the system on behalf of the taxi company and of the government of the city.
- Ride: is a single taxi ride from a location to another one.
- Shared ride: a ride shared with other passengers arranged by the system; in this way clients that have common taxi lines can shrink the cost of the single ride.
- Location: are the GPS coordinates (or the street address) to unequivocally identify a place. It could be the position of a taxi, the origin or destination of a ride.
- Availability: is the “status” of a taxi driver, s/he can be available to take care of new jobs, thus the system should forward compatible requests, or can be unavailable, or occupied, busy, meaning that the taxi is temporarily off line with respect to the system, meaning that the system should not consider that taxi for requests assignment. The availability is set to “off” also when the taxi is carrying passengers.
- Queue: the queue of the available taxis in each zone of the city; it is updated by the system, based on the interaction with taxi drivers that can accept or reject the proposals forwarded by the system.
- Zone: the city is divided in several zones to better allocate and coordinate the taxi fleet.
- Request: it is intended as a real time call of a passenger that simply provides his/her own pickup position for an immediate service, without previous reservation.
- Reservation: it is a previously planned request, for which the client can specify the date and time, the origin and destination, the willingness to share the ride.

1.7.2 Acronyms

- RASD: Requirements Analysis and Specification Document.
- DBMS: DataBase Management System.
- DB: DataBase.

- OS: Operating Systems.
- API: Application Programming Interface.
- HW: HardWare.
- TCP: Transmission Control Protocol.
- HTTP: Hypertext Transfer Protocol.
- HTTPS: HTTP over SSL/HTTP Secure.
- SSL: Secure Sockets Layer.
- DMZ: Demilitarized Zone.

1.7.3 Abbreviations

- [G n]: n^{th} goal.
- [R n]: n^{th} functional requirement.
- [D n]: n^{th} domain assumption.

1.8 Reference Documents

- Specification document: myTaxiService project
- IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications.

1.9 Overview

- Section 1: Introduction, it gives a brief description of the purpose, functionalities and goals of the application.
- Section 2: Overall Description, focuses more in-depth on features of the software, constraints and assumptions.
- Section 3: Specific Requirements, this part lists requirements, typical scenarios and use cases, together with UML diagrams to provide a more easy-to-read insight at the several functionalities of the software.

2 Overall description

2.1 Product perspective

myTaxiService is a web and mobile application which is not integrated with any other existing system, it is independent and totally self-contained. It has the interface for an administration, except for that feature it is entirely user based. The application does provide some APIs for integration with future projects.

2.2 Product functions

The function summary that is necessary for this part can be taken directly from the section of the higher-level specification in the first part of this document *Scope* (see section 1.2) and *Goals* (see section 1.5).

2.3 User characteristics

No technical expertise is required to the intended users of the new service. Part of the developers' effort are invested exactly in designing a user-friendly, self-explanatory interface, yet keeping a modern look and feel to assure an innovative user experience that fully exploits the power of present web and mobile technologies.

2.4 Constraints

2.4.1 Regulatory Policies & Safety and security

myTaxiService fulfills all the required policies that regulate the public transportation, in particular the taxi-related ones. Besides it enforces all the requested security and privacy-connected regulations for web based transmission of information; it can make use of cookies to memorize some users' preferences and the browser can request to detect the user's position. The mobile application only requires basic permissions.

2.4.2 Hardware limitations

myTaxiService requires a computer or a smartphone connected to the internet. The mobile application requires the GPS module.

2.4.3 Interfaces to other applications

myTaxiService requires to access the Internet and the Google Maps APIs to provide map visualizations and map-related services.

2.4.4 Parallel operation

myTaxiService supports parallel operations from different users when working with the DB and when dealing with all the operations done by the user after the connection.

2.5 Assumptions and dependencies

- Entitled taxi drivers are inserted in the system by the administrator.
- Taxi zones are predetermined by the city government and will never change in the system time horizon.
- The rides that can be shared are only the reserved ones.
- Passengers sharing a ride will be matched only if they share the same pick up and destination zone.
- Once the ride has been created it's only possible to cancel it; thus if the passenger wants to modify a request he will need to delete the old and create a new one.
- If the taxi availability is set to "off", the system does not consider that particular taxi for jobs assignment.
- The login page for the users is unique, but according to their role they will be automatically redirected to the appropriate page for the passenger, employee or administrator.
- There are no dependencies between users.
- Passengers are not allowed to book overlapping taxi rides.
- Passengers pay only using their valid credit or debit card by linking it to their own profile.
- Passengers always have enough money to pay the taxi fare.
- The web application and the mobile counterpart share the same functionalities.
- The waiting time can vary depending on the current traffic condition.
- The taxi fleet is composed of only one model of car, thus all the cars have the same capacity.
- When a taxi is mobilized, it will reach the pick up location in the shortest possible time.
- All the actions performed by a taxi driver during working hours will be performed through the mobile application.
- Taxi drivers are assigned the zone by the system.
- Each taxi driver is associated with only one taxi.
- Locations provided by users are always correct.

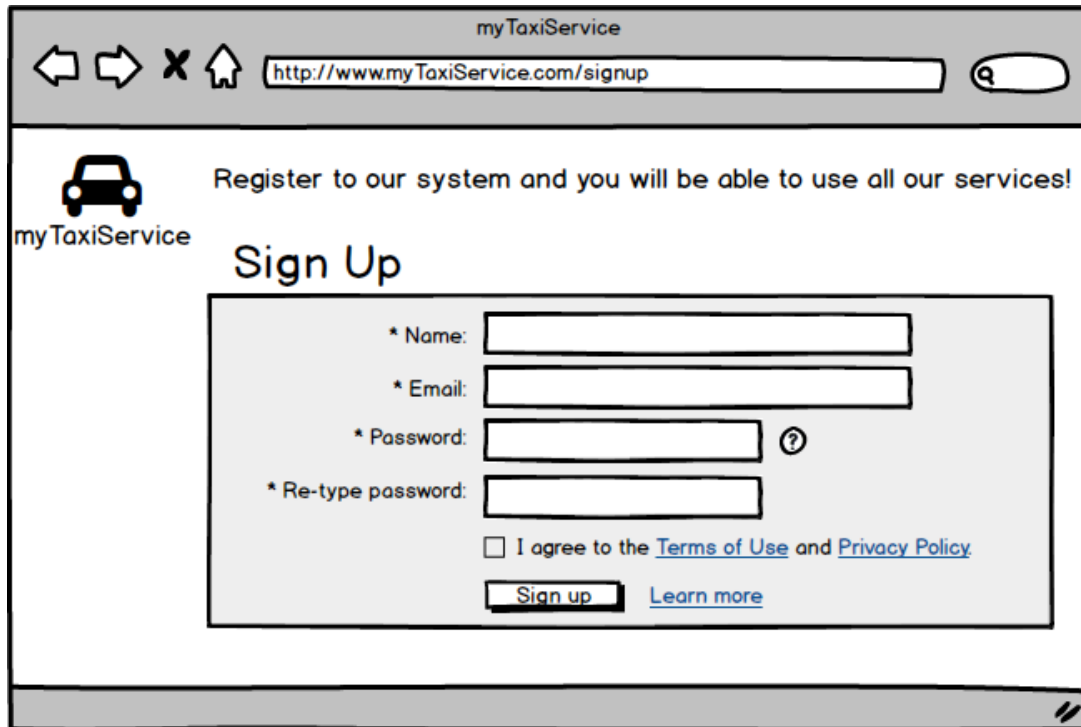
3 Specific requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

Shown below are some mock-ups that preview the user interface of the main features the system shall provide.

3.1.1.1 Sign-up This page presents the sign up form for the clients. The user will need to provide his personal data, home and email addresses, phone number, favorite payment option.

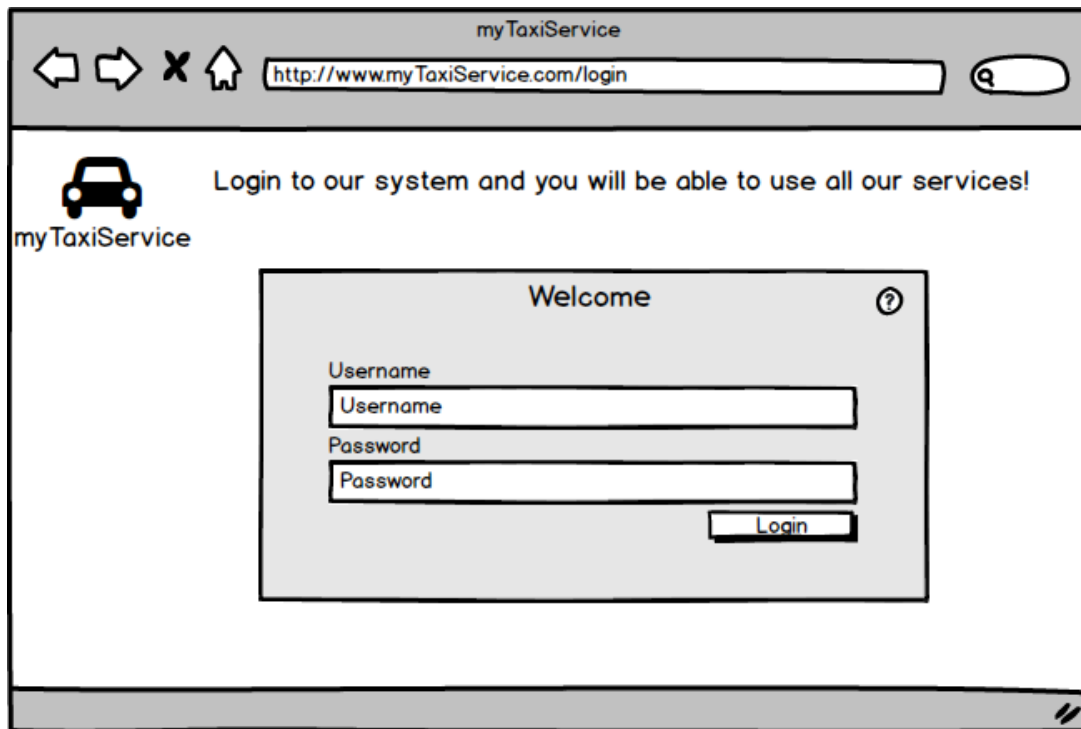


A web browser window titled "myTaxiService" with the address bar showing "http://www.myTaxiService.com/signup". The page features a car icon and the text "myTaxiService". Below this, it says "Register to our system and you will be able to use all our services!". The main heading is "Sign Up". The form includes fields for "Name:", "Email:", "Password:", and "Re-type password:". There is a checkbox for "I agree to the Terms of Use and Privacy Policy" and a "Sign up" button. A "Learn more" link is also present.



A mobile phone screen displaying the "myTaxiService" sign-up page. The status bar at the top shows "ABC" and "04:44 PM". The app icon is a car. The heading is "Sign up". The form includes fields for "Name:*", "Password:*", and "Email:*". A "Sign up" button is at the bottom.

3.1.1.2 Login This page shows the login form for the final users.



3.1.1.3 Call a taxi The user can ask for a taxi providing the pickup location through a complete address or through the automatic detection of his/her location thanks to the browser or the GPS. The user can also choose from an history of “recent addresses”.



3.1.1.4 Plan a ride The user can plan a ride in advance providing the pickup and drop off locations, the date and time, the willingness to share the ride.

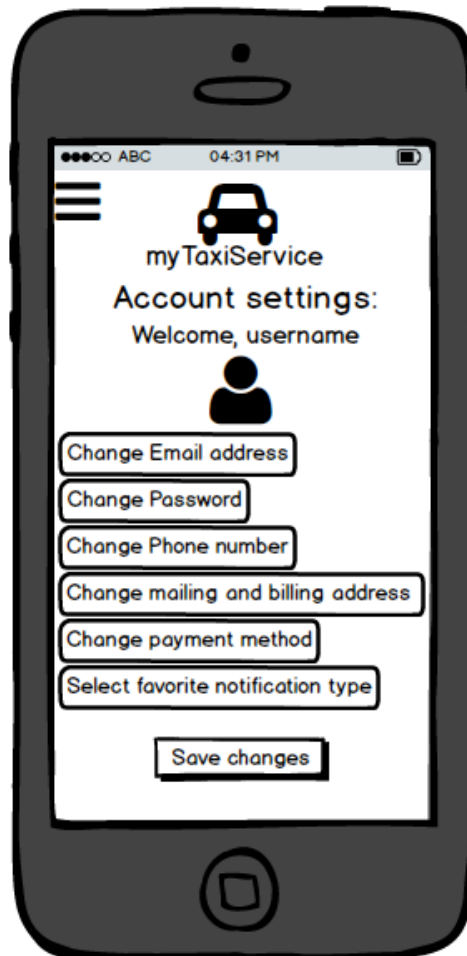
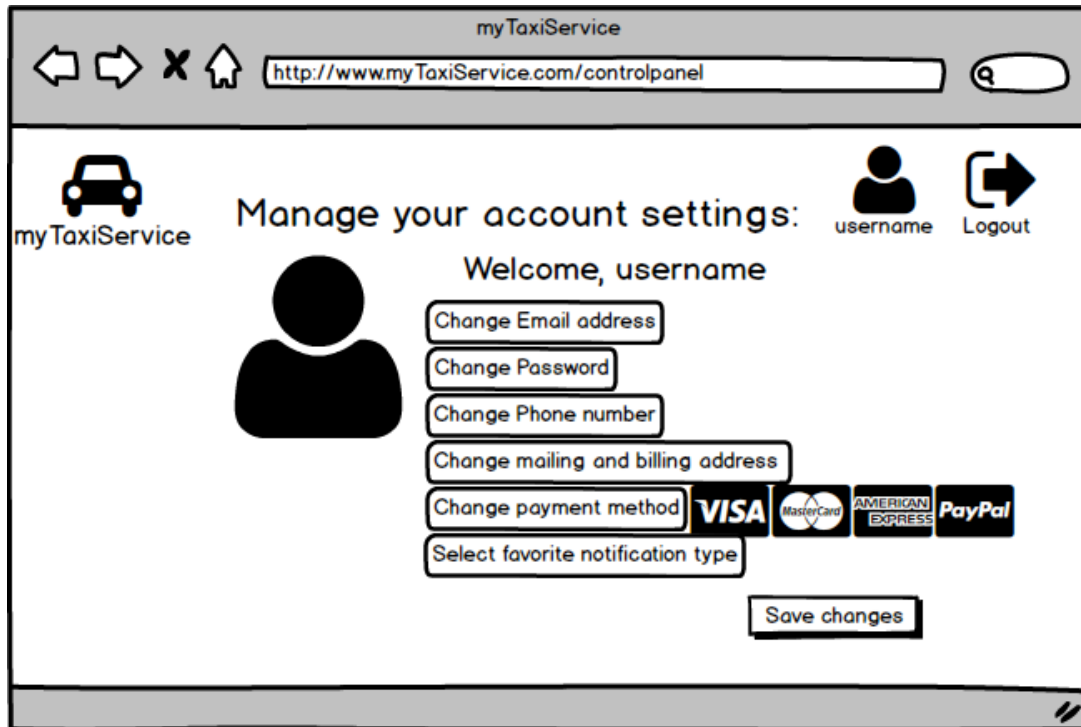
The image shows a web browser window for 'myTaxiService'. The address bar displays 'http://www.myTaxiService.com/bookride'. The page has a navigation bar with 'Request a taxi', 'Book a ride' (highlighted), and 'Your reservations'. A user profile icon is labeled 'username' and a 'Logout' button is present. The main heading is 'Make a reservation:'. The form includes fields for 'Pickup Address:*' (with a text input 'Enter a location', a 'Recent Addresses' dropdown, and a location pin icon), 'Drop off Address:*' (with a text input 'Enter a location', a 'Recent Addresses' dropdown, and a location pin icon), 'Date:*' (with a text input 'dd/mm/yyyy'), 'Time:*' (with a text input '12:00'), and 'Passengers:*' (with a text input '1'). There is a checkbox for 'Include shared rides' and a map showing a route between two points.

The image shows a mobile app interface for 'myTaxiService'. The status bar at the top shows 'ABC' and '11:14 AM'. The app has a hamburger menu icon and a car icon. The main heading is 'Make a reservation:'. The form includes fields for 'Pickup Address:*' (with a text input 'Enter a location' and a location pin icon), 'Drop off Address:*' (with a text input 'Enter a location'), 'Date:*' (with a text input 'dd/mm/yyyy'), 'Time:*' (with a text input '12:00'), and 'Passengers:*' (with a text input '1'). There is a checkbox for 'Include shared rides'.

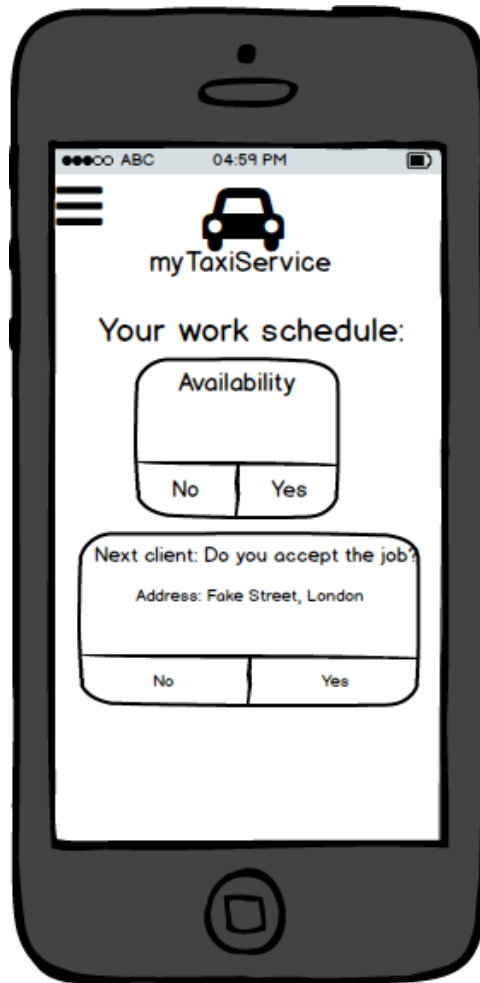
3.1.1.5 Your reservations The user can see both the active reservations and the past ones. S/he can edit the active reservations within the established time frame before the meeting time.



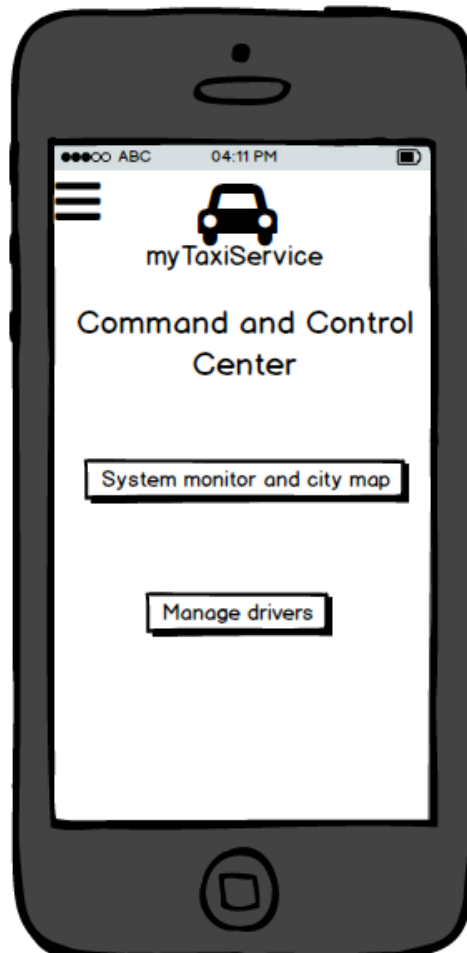
3.1.1.6 Your profile The user can edit the personal profile modifying the password, phone number, email address, permanent address, payment method.



3.1.1.7 Manage jobs The taxi driver's home page: s/he can accept or reject the requests forwarded by the system and set her/his own "availability".



3.1.1.8 Administrator panel The administrator's home page: s/he can monitor the whole system or manage taxi drivers inserting new employees in the system DB or modifying/deleting existing ones.



3.1.2 Hardware Interfaces

myTaxiService does not require and does not support any additional HW interface, it is enough a compatible smartphones or any supported browser on a computer.

3.1.3 Communication Interfaces

myTaxiService uses the TCP transport protocol and HTTP/HTTPS over SSL application layer protocol to guarantee the top of the line security on the matter of transmission of private data.

3.2 Functional Requirements

3.2.1 [G1] The system has to guarantee to the user the possibility to access the service either through a web application or a mobile application.

- [D1] The system must offer the same functionalities to the final user in both cases.

3.2.2 [G2] The system has to guarantee a fair management of the queues in each taxi zone.

- [R1] The system is able to control the queue of taxis in every zone and enforce the predetermined priority rules to guarantee a fair management of the queues.
- [R2] If the taxi driver on top of the queue accepts a job then the system allocates the taxi for the specific request and therefore updates the queue.
- [R3] If the taxi driver on top of the queue rejects the job then the system updates the queue placing the second taxi as first and the one that rejected the job in the last position of the queue of the same zone.
- [D1] Each taxi queue is never empty.

3.2.3 [G3] The system has to assign each taxi to the correct taxi zone using the GPS coordinates that receives from each taxi.

- [R1] The system is able to map the position of the taxi fleet and assign each taxi to a predetermined zone of the city according to its position.
- [D1] The GPS coordinates are always available and correct.

3.2.4 [G4] The system has to allocate a taxi for each request or reservation.

- [R1] In case of a request the system allocates a taxi as soon as a driver in the corresponding zone accepts the job.
- [R2] In case of a reservation the system allocates the taxi 10 minutes before the meeting time with the client.
- [R2] The system computes the matching of people in shared rides taking into account the number of passengers from each reservation; the total number of passengers cannot exceed the capacity of the taxi.
- [D1] There is always at least one available taxi.

3.2.5 [G5] The system has to notify the users, both passengers and taxi drivers, about updates on taxi requests and reservations in which they are involved.

- [R1] The awaiting passenger is notified by the system if the taxi is delayed for any reason.
- [R2] The passenger is notified when other passengers sharing the same route join the ride or cancel the reservation; the user is also updated with the recomputed taxi fare.
- [R3] The system notifies the taxi driver allocated to a request if the correspondent passenger has deleted the reservation.
- [R4] The system allows passengers to choose the type of notification they will receive among text message, email and mobile push notification.
- [D1] The taxi driver will always receive only mobile push notifications.
- [D2] The user will provide his preference on at least one type of notification.

3.2.6 [G6] The system has to offer public APIs to enable the possibility to develop additional services on top of the basic ones.

- [R1] The system offers APIs to third party applications using web APIs as technology.
- [R2] The system replies using current industry standards.
- [D1] Access to APIs functionalities is provided only using the HTTPS protocol.

3.2.7 [G7] The passenger shall be able to sign up to the service.

- [R1] The user cannot register more than once.
- [R2] The user must choose a username not already used by someone else.
- [R3] Unregistered users can only see the sign up and login page.
- [R4] The user must fill in all the mandatory fields of the proposed form to successfully complete the registration process.
- [D1] Phone number, email address and credit card data used for the registration are mandatory and must be valid.

3.2.8 [G8] The passenger shall be able to log in to the service.

- [R1] If the user hasn't the correct login information cannot access the system.
- [R2] If the user provides correct credentials the system will grant him/her the access.
- [D1] The system does not implement a password retrieval mechanism.

3.2.9 [G9] The passenger shall be able to request a taxi.

- [R1] The user can ask for a taxi simply providing his/her own position.
- [R2] The system is able to dispatch the request of the client to the right queue according to his/her position.
- [R3] The system has to answer a taxi request by informing the passenger about the code of the taxi and the waiting time.

- [R4] When a passenger books a ride s/he is given a code that uniquely identifies her/him in front of the taxi driver.
- [D1] The user knows that the capacity of the taxi is 4 passengers.
- [D1] If there are more than 4 passengers the user has to make two or more requests.

3.2.10 [G10] The passenger shall be able to delete a taxi request.

- [R1] The passenger can withdraw the request within 30 seconds after the call.
- [R2] A penalty fee is automatically collected if the user deletes the request after the allowed time frame.
- [D1] Passengers always wait for the requested taxi.

3.2.11 [G11] The passenger shall be able to create a reservation for a taxi ride.

- [R1] The user can customize the reservation, specifying the date and time of the ride, the origin and destination and the number of passengers.

3.2.12 [G12] The passenger shall be able to modify or delete a taxi reservation.

- [R1] The user can check his/her own reservations and edit or delete them.
- [D1] If the user deletes a reservation in the last 10 minutes before the meeting time (the taxi has already been allocated), then there is a fixed penalty fee automatically deducted from the user's credit card.

3.2.13 [G13] The passenger shall be able to enable taxi sharing option.

- [R1] The user can enable the sharing of a reservation.
- [R2] The system computes the matching of people in shared rides taking into account the number of passengers from each reservation; the total number of passengers cannot exceed the maximum capacity of the taxi.
- [D1] This is only valid for reservations booked in advance.

3.2.14 [G14] The passenger shall be able to see historical data on his taxi rides.

- [R1] The system keeps track of the recent routes and addresses.
- [R2] The user can select locations from past rides to plan new ones.

3.2.15 [G15] Taxi drivers shall be able to log in to the service.

- [R1] If the user hasn't the correct login information cannot access the system.
- [R2] If the user provides correct credentials the system will grant him/her the access.
- [D1] The system does not implement a password retrieval mechanism.
- [D2] Only legally employed taxi drivers have the login information to access the system.

3.2.16 [G16] Taxi drivers must inform the system about their availability.

- [R1] The system allows taxi drivers to set their availability.
- [R2] The system allows to specify the reason of the unavailability.
- [D1] All the actions performed by a taxi driver during working hours will be performed through the mobile application.
- [D2] Taxi drivers always remember to switch their availability.

3.2.17 [G17] Taxi drivers must confirm if they are going to take care of a certain call.

- [R1] The system allows taxi drivers to accept or reject a call.
- [D1] Taxi drivers always remember to accept or reject a call.

3.2.18 [G18] The administrator shall be able to add, edit and delete taxi drivers in the system.

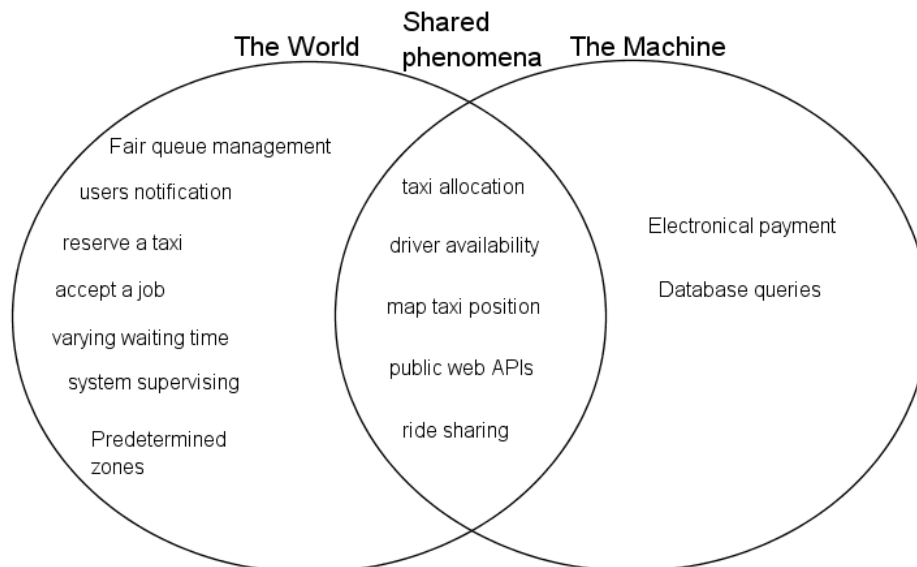
- [R1] The system allows to the administrator to add, edit and delete taxis.
- [D1] The administrator only inserts in the system already verified entitled taxi drivers.

3.2.19 [G19] The administrator shall be able to manage and supervise the operation of the whole system, including the real-time situation of all the queues and of the taxis.

- [R1] The system allows to the administrator to have a glance view of the system situation.
- [D1] The visualization is read only.

3.3 The world and the machine

“The World & The Machine” model by M. Jackson & P. Zave has been used as an additional domain analysis of *myTaxiService* application. This approach identifies the portion of “The World” affected by “The Machine”, the portion of system to be developed, and the intersection (“Shared Phenomena”) between the world and the application, that are all world information known or managed directly by the application. Here is a simplified version of the model.



3.4 Scenarios

3.4.1 Scenario 1

Bob lives in Milan, a very crowded and traffic congested city, for this reason he decided not to buy a car. When he needs to move around the city he finds very convenient the taxi service, less crowded than other public transport means. By now he has learned how to use all the different functionalities offered by *myTaxiService* mobile application. Whenever and wherever he needs he simply start the application on his mobile phone and he requests a taxi in a matter of seconds simply providing his location through the built-in GPS of his smartphone. In a very short time, usually no more than a couple of minutes, he receives the notification that confirms the take on responsibility of his call with the code of the incoming taxi and the estimated waiting time. Meanwhile the system reserves on the credit card he linked the minimum fee in case of cancellation. Once the taxi arrives Bob exhibits the identification code to the taxi driver and is taken to his destination; finally the system automatically computes the fare based on the distance covered and deducts the correct amount of money from his credit card.

3.4.2 Scenario 2

Alice has just moved to Milan and hasn't got any car. For this reason, she looks on the Internet to understand which options she has to move around the city with public transports and she finds *myTaxiService*'s website that presents the taxi service. She decides to sign up to make use of the service, so she provides her personal data including the data of her credit card and selects "email" as notification option. Once the 2-step verification process is successfully completed she logs in with the credentials chosen during the registration process. She is now operative and wants to try out the service, so she enters the booking page to make a reservation for the next day. She fills in the pickup address with her own and the destination field with the museum she wants to visit and sets the date and time, enabling the shared ride choice and finally submits the request. The system confirms the reservation process went through.

3.4.3 Scenario 3

Alice is invited to her friend's Bob birthday party held the next evening at Bob's house, so she logs in to the *myTaxiService*'s website to reserve a taxi for tomorrow night. Unfortunately, she doesn't recall her friend's address, but she remembers that the previous week they took a taxi together to his house, so she browses the history of her own reservation and from there recognizes Bob's house address, then Alice uses it as drop off location and completes the reservation. Unluckily the following morning Bob phones to Alice and warns her that the party is postponed indefinitely due to bad weather forecast. Alice then decides to delete the reservation for taxi for the night, so she logs in to the website and cancel the booking without paying any fees.

3.4.4 Scenario 4

Mike is a taxi driver in Milan and its company relies on *myTaxiService* for the management of the work. He received the access codes to the system once he had been employed. Now every morning he starts his daily work routine by starting the *myTaxiService* app on his phone and heading to the assigned zone with his taxi to pick up the first client. He switches on the "availability" to work and starts receiving job proposals. Mike accepts them almost always; he only takes the liberty to switch off the system for a 30-minute lunch break. Often it happens that he has to pick up two clients in the same area for a shared ride because they are headed in the same direction. The shared ride feature has been having an huge approval among the clients since the implementation of the *myTaxiService* application. At the end of the day Mike stops

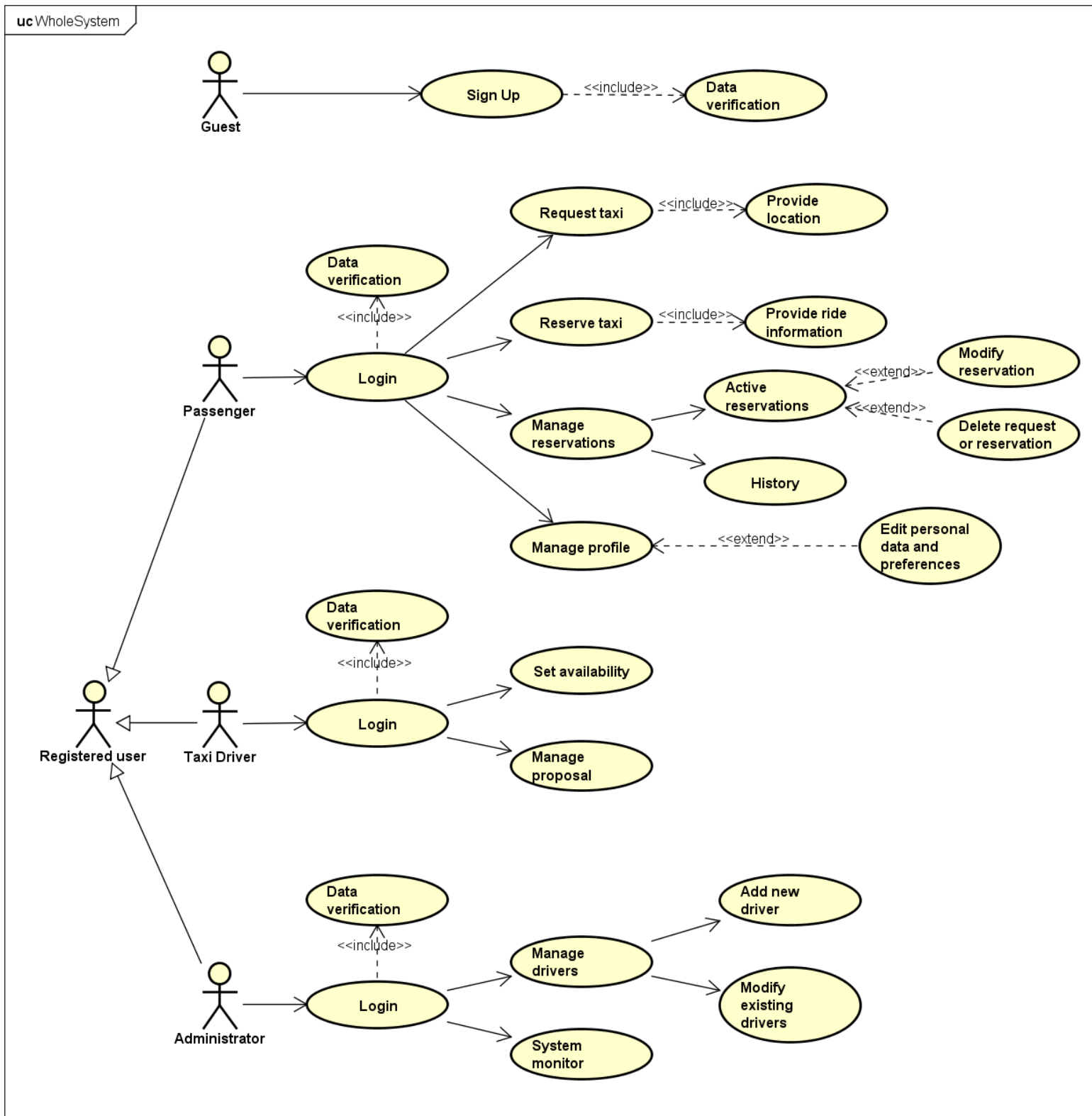
the flow of job proposals setting the “availability” back again to off and returns to his family at home.

3.4.5 Scenario 5

Jack is the manager of the taxi company in Milan and therefore he is the administrator of the *myTaxiService* system. He spends his days monitoring the activity of the system thanks to the supervision functionality of the system. He can access it anytime anywhere thanks to the double interface for web browser and mobile application. Of course he prefers the bigger monitor of his computer when he’s in the office with respect to the small display of the smartphone, but he is more relaxed knowing that he is notified if something goes wrong even when he is outside the office. Often it happens that he needs to insert in the system new just employed drivers; in that case he collects all the information about the person and register the user with the specific functionalities for the drivers.

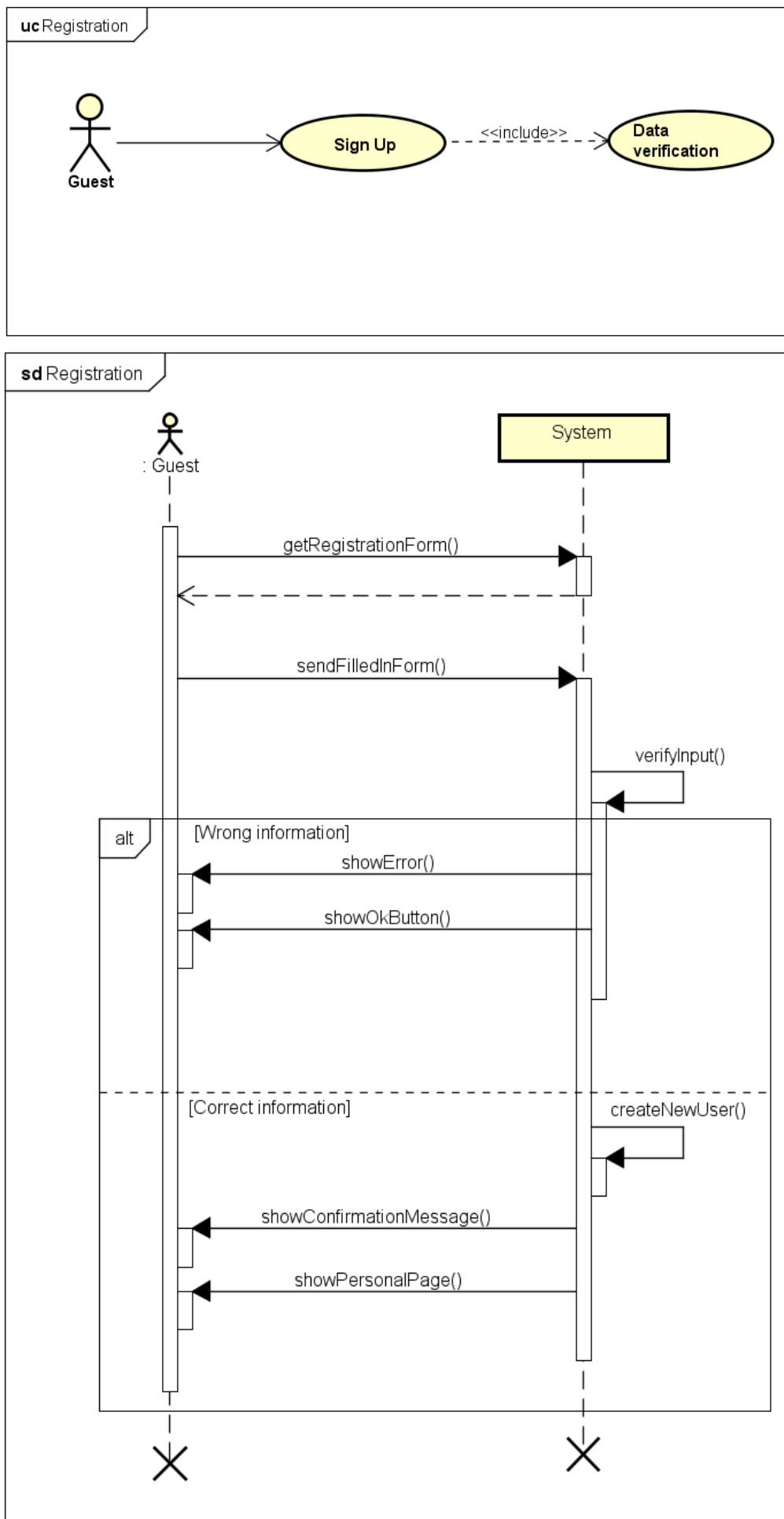
3.5 UML Models

3.5.1 Use Case



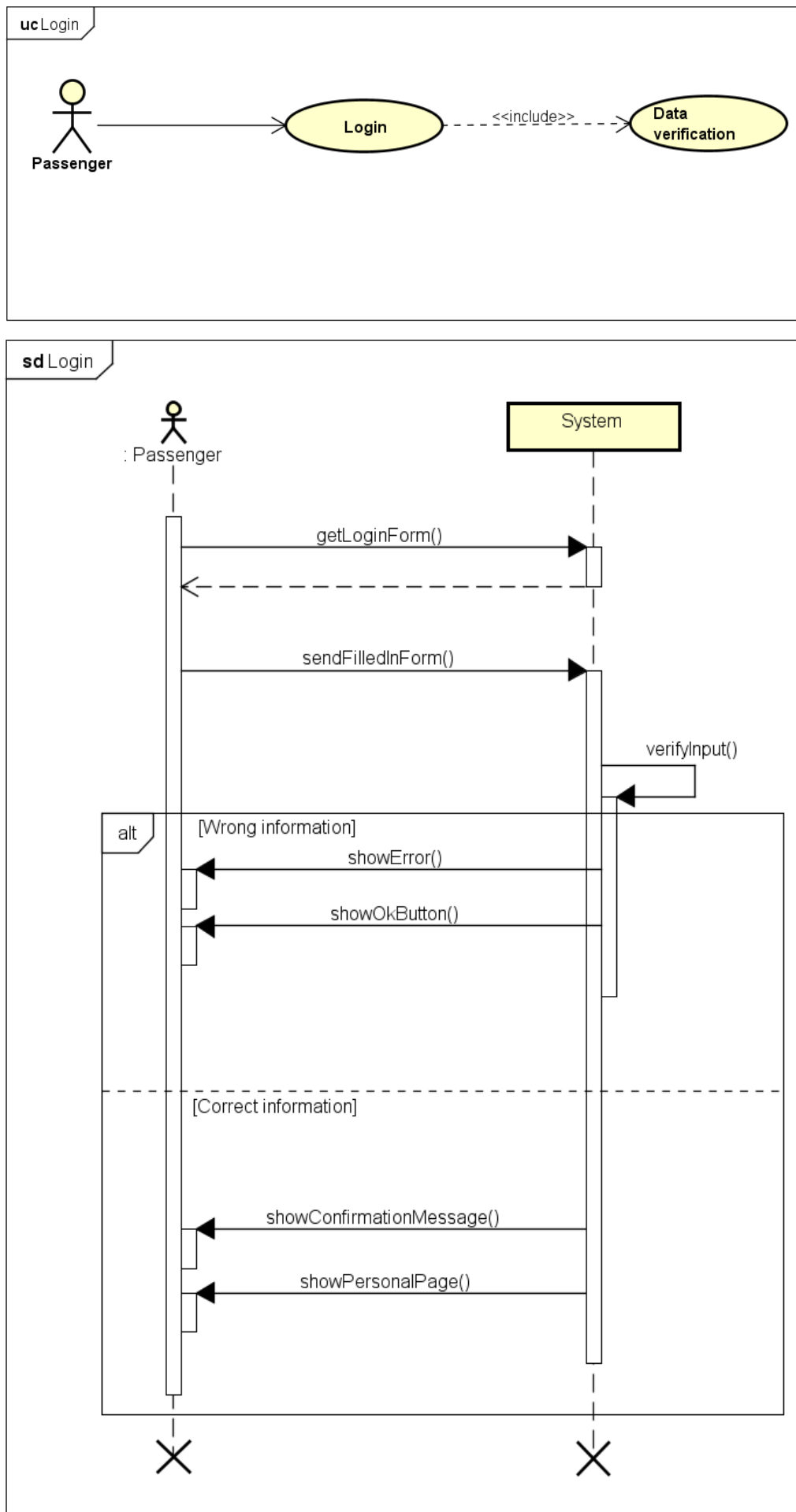
3.5.1.1 Guest registers to *myTaxiService*

Actor	Guest
Goal	[G7]
Input Condition	NULL
Event Flow	<ol style="list-style-type: none">1. Guest on the home page clicks on “Sign up”.2. Guest fills in at least all mandatory fields.3. Guest clicks on the “confirm” button.4. The application saves the data in the DB.
Output Condition	Guest successfully completes the registration process and becomes a registered user. From now on s/he can login to the application using her/his credentials and use <i>myTaxiService</i> .
Exceptions	<ol style="list-style-type: none">1. The guest is already a registered user.2. One or more mandatory fields are not valid or missing.3. The username chosen is already used by another user.4. The email chosen is already associated to another user. <p>All exceptions are handled alerting the guest of the problem through an error message.</p>



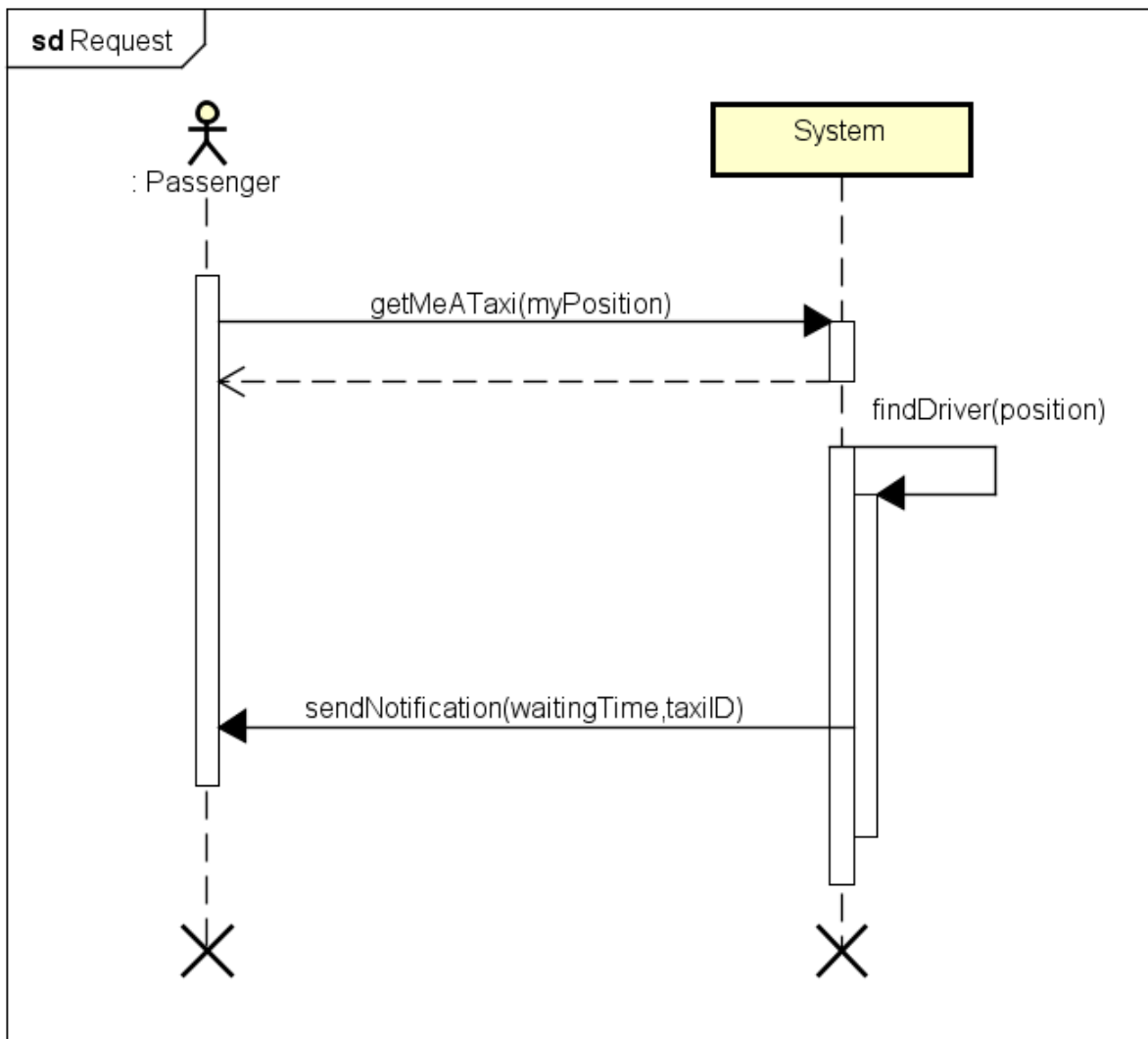
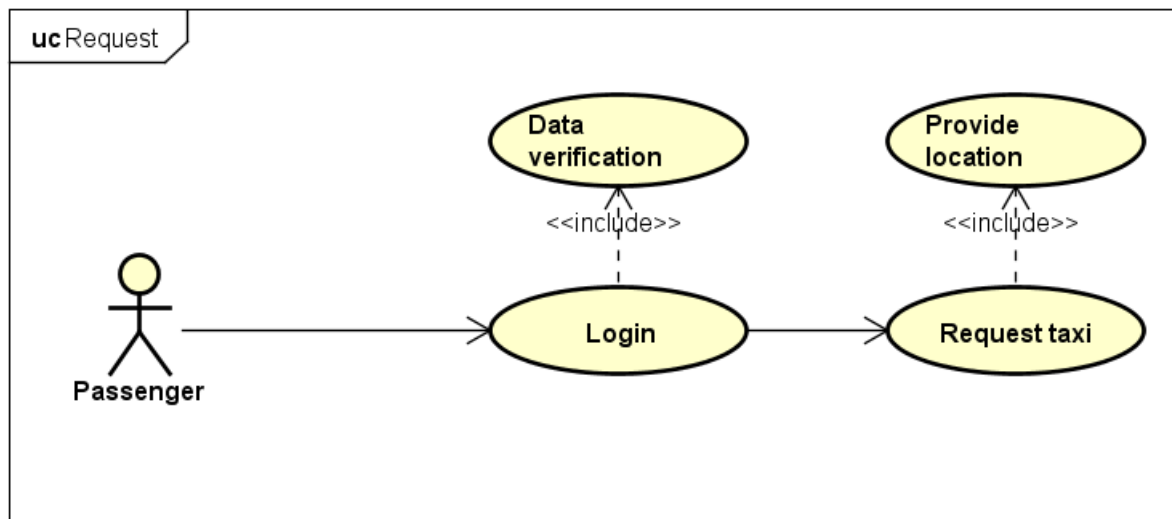
3.5.1.2 Guest logs in to *myTaxiService*

Actor	Guest, Passenger
Goal	[G8]
Input Condition	The user must be already registered.
Event Flow	<ol style="list-style-type: none">1. <i>myTaxiService</i> shows the login page.2. Guest completes the form inserting correct username and password.
Output Condition	<ol style="list-style-type: none">1. <i>myTaxiService</i> verifies the credentials of the guest and if correct shows the home page for the registered user.2. Guest is promoted to registered user.
Exceptions	If username and/or password are incorrect the system notifies this to the guest through an error message.



3.5.1.3 Passenger requests a taxi

Actor	Passenger
Goal	[G9]
Input Condition	The user must be already logged in.
Event Flow	The user requests a taxi using any of the options to provide his/her own location.
Output Condition	<ol style="list-style-type: none">1. <i>myTaxiService</i> starts the process in order to find a taxi for the user.2. Once the system has allocated a taxi it notifies the user.
Exceptions	NULL



3.5.1.4 Passenger cancels a request for a taxi

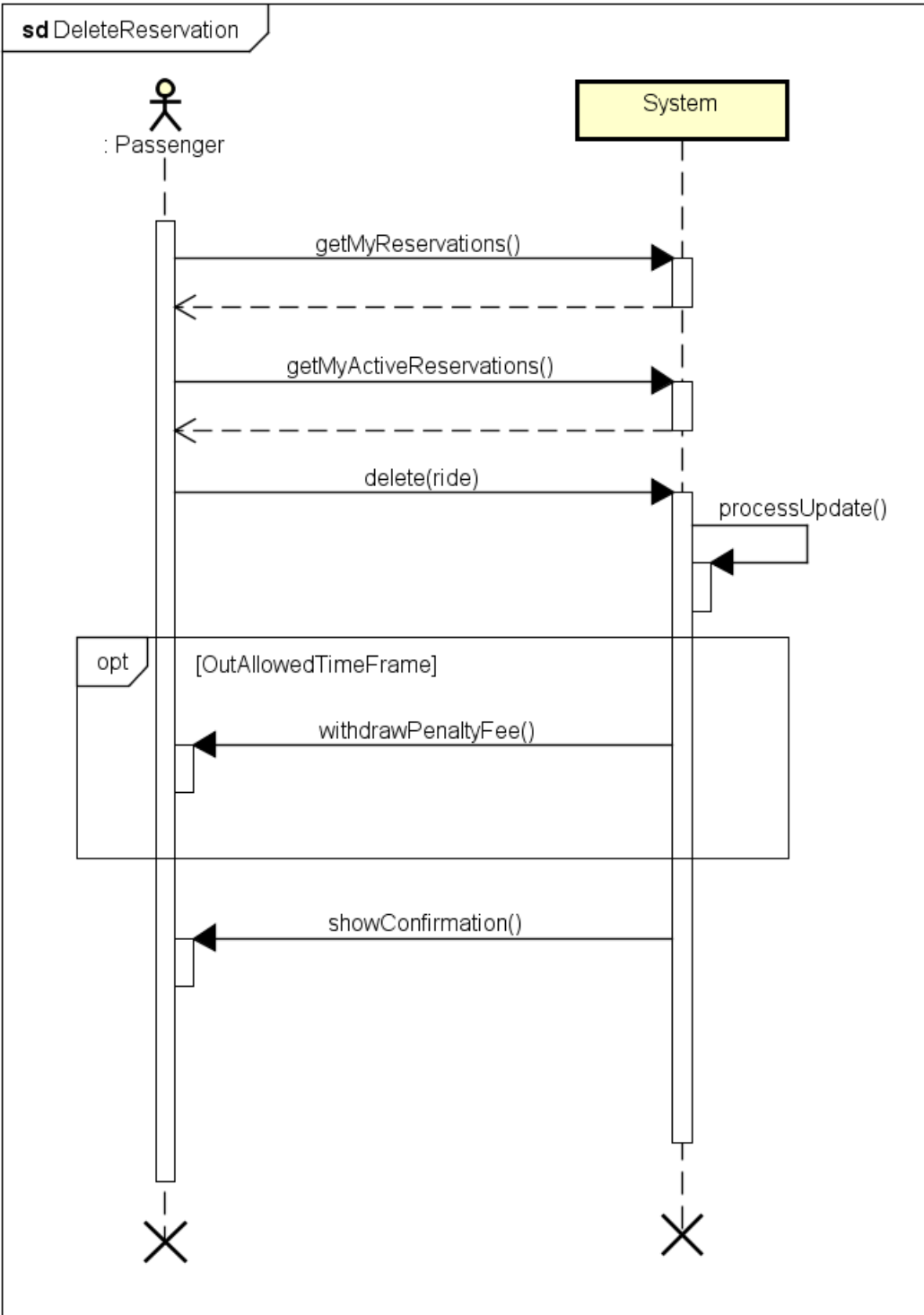
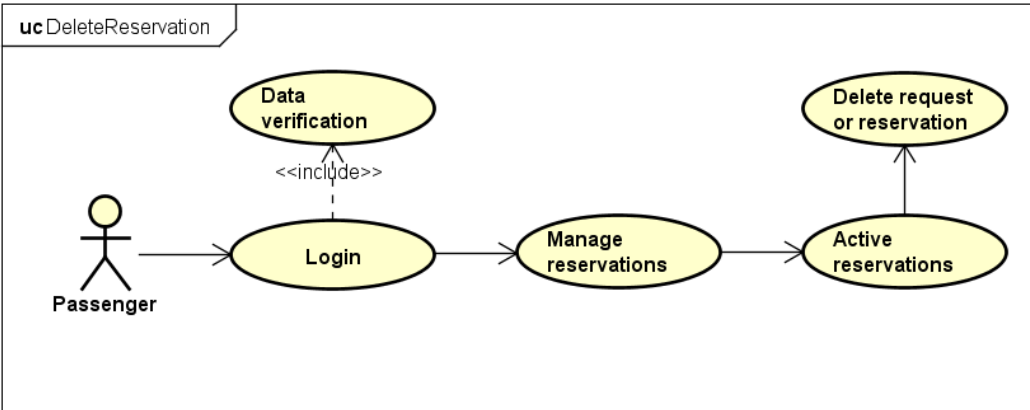
Actor	Passenger, Taxi driver
Goal	[G10]
Input Condition	<ol style="list-style-type: none">1. The user must be already logged in.2. The user must have a pending request for a taxi.
Event Flow	The user deletes the pending ride.
Output Condition	<i>myTaxiService</i> notifies the corresponding taxi driver that the job is canceled.
Exceptions	If the cancellation has been made after the allowed time frame the system applies a penalty fee to the user.

3.5.1.5 Passenger creates a reservation for a taxi

Actor	Passenger
Goal	[G11]
Input Condition	The user must be already logged in.
Event Flow	The user fills in the form for the reservation of a ride customizing the parameters.
Output Condition	<ol style="list-style-type: none">1. <i>myTaxiService</i> records the new reservation in its DB.2. The system notifies the user of the successful reservation.
Exceptions	NULL

3.5.1.6 Passenger edits or deletes a reservation for a taxi

Actor	Passenger
Goal	[G12]
Input Condition	<ol style="list-style-type: none">1. The user must be already logged in.2. The user must have an active reservation.
Event Flow	The user modifies some parameters of his/her reservation, possibly also deleting it.
Output Condition	<ol style="list-style-type: none">1. <i>myTaxiService</i> records the changes in the reservation in its DB.2. The system notifies the user of the successful changes.
Exceptions	The system collects the penalty fee if the user deleted the ride beyond the allowed time-frame.



3.5.1.7 Passenger reserves a ride with shared option

Actor	Passenger
Goal	[G13]
Input Condition	The user must be already logged in.
Event Flow	The user fills in the form for the reservation of a ride customizing the parameters, in particular enabling the sharing of the ride.
Output Condition	<ol style="list-style-type: none">1. <i>myTaxiService</i> records the changes in the reservation in its DB.2. The system notifies the user of the successful reservation.
Exceptions	NULL

3.5.1.8 Passenger checks the history of his/her own requests

Actor	Passenger
Goal	[G14]
Input Condition	The user must be already logged in.
Event Flow	The user accesses the page of the history of his/her own rides.
Output Condition	The system returns the result of the corresponding query on the DB to the user.
Exceptions	NULL

3.5.1.9 Taxi driver logins to *myTaxiService*

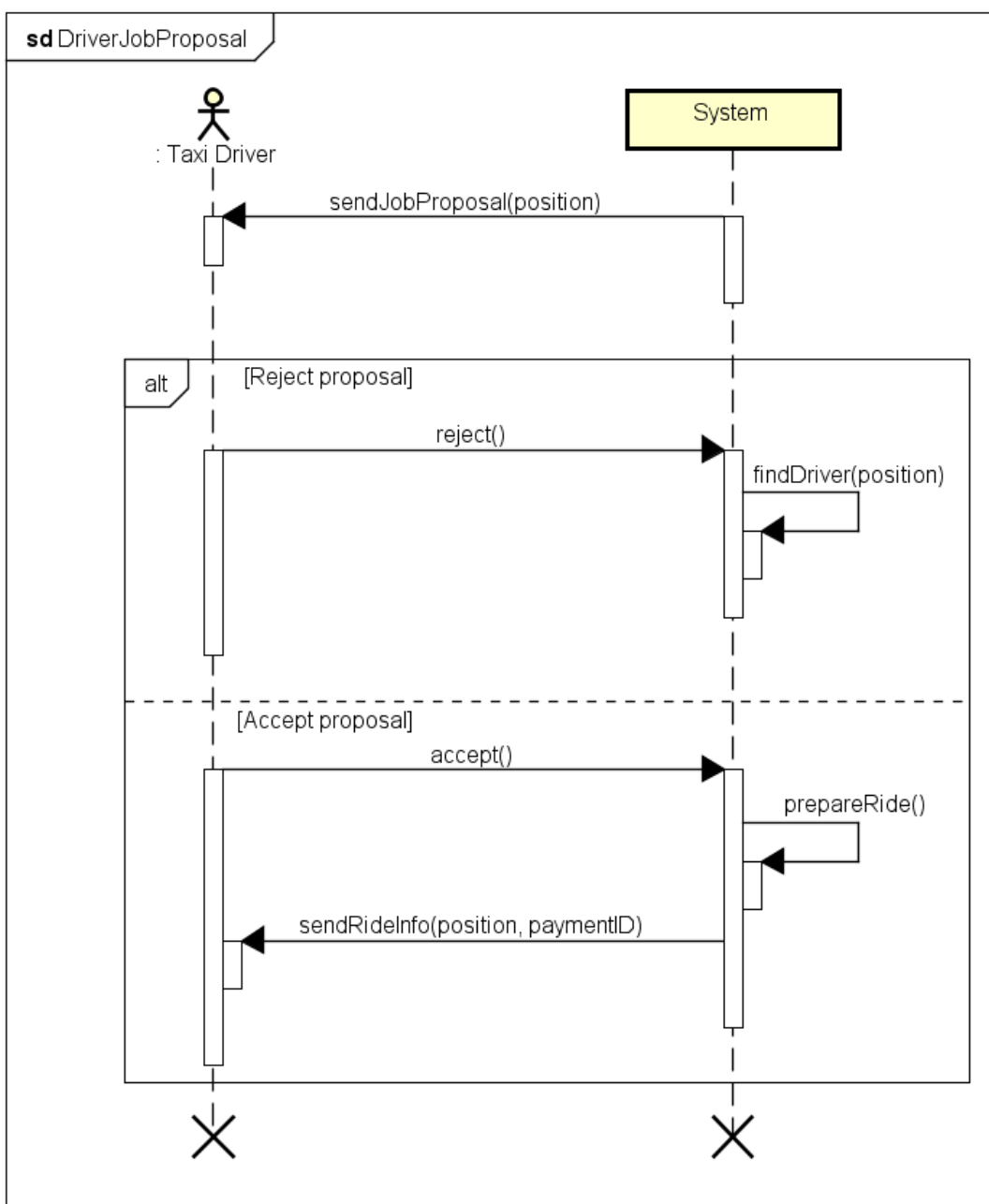
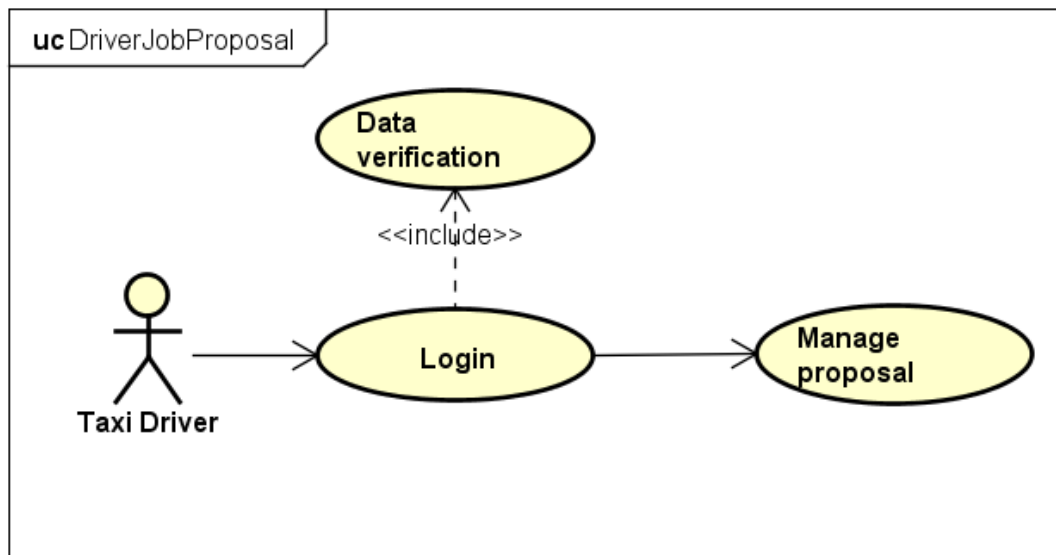
Actor	Taxi driver
Goal	[G15]
Input Condition	The user must be already registered (inserted in the system by the administrator).
Event Flow	<ol style="list-style-type: none">1. <i>myTaxiService</i> shows the login page.2. The user completes the form inserting correct username and password.
Output Condition	<ol style="list-style-type: none">1. <i>myTaxiService</i> verifies the credentials of the employee and if correct shows the home page for the registered user.2. Guest is promoted to registered user.
Exceptions	If username and/or password are incorrect the system notifies this to the guest and shows the login page again.

3.5.1.10 Taxi driver sets his/her availability with the mobile application

Actor	Taxi driver
Goal	[G16]
Input Condition	The user must be already logged in.
Event Flow	<ol style="list-style-type: none">1. <i>myTaxiService</i> shows the home page for the driver.2. The user sets his/her own availability to receive job proposals as s/he prefers.
Output Condition	The system receives the setting of the user and operates accordingly.
Exceptions	NULL

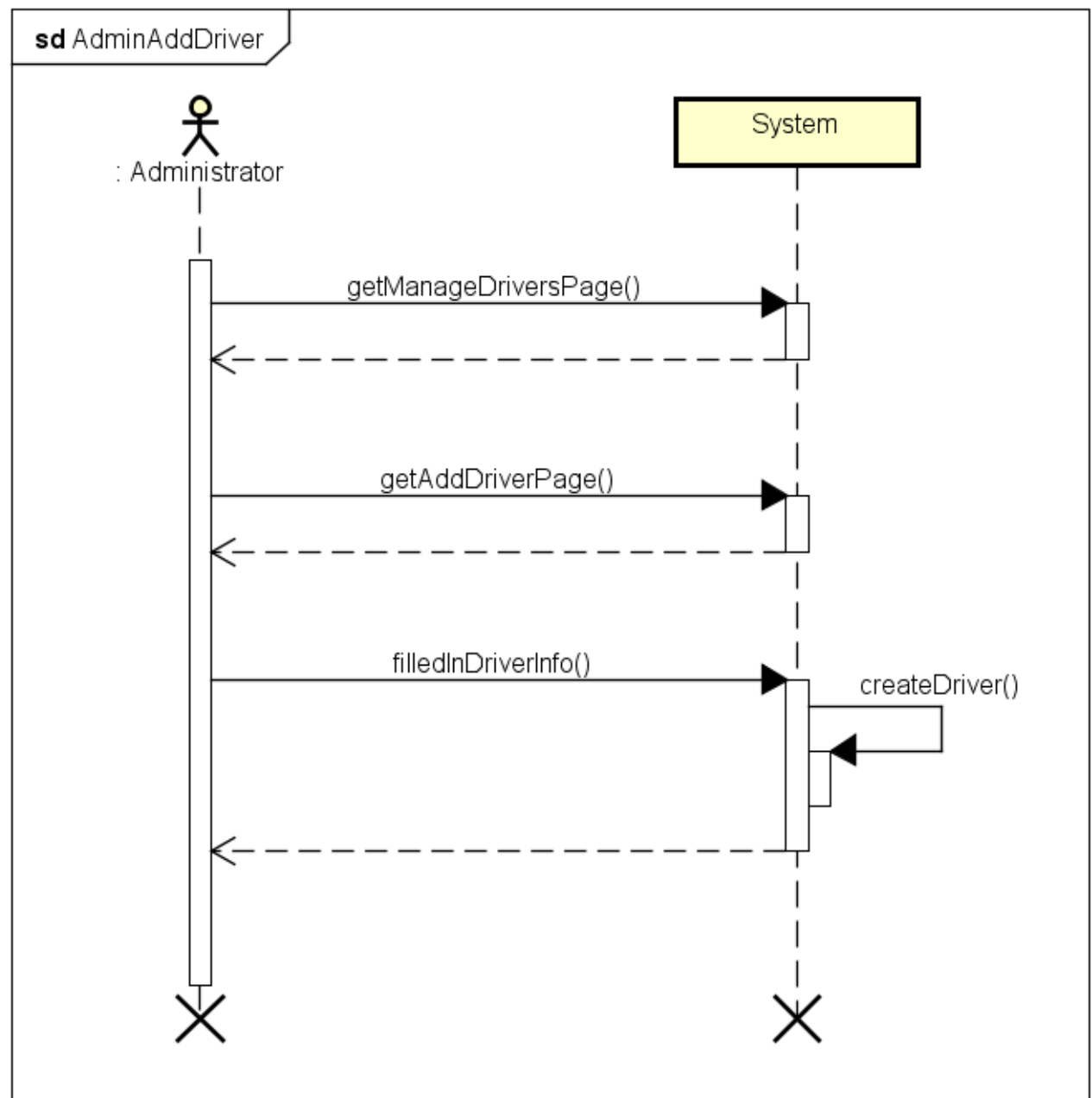
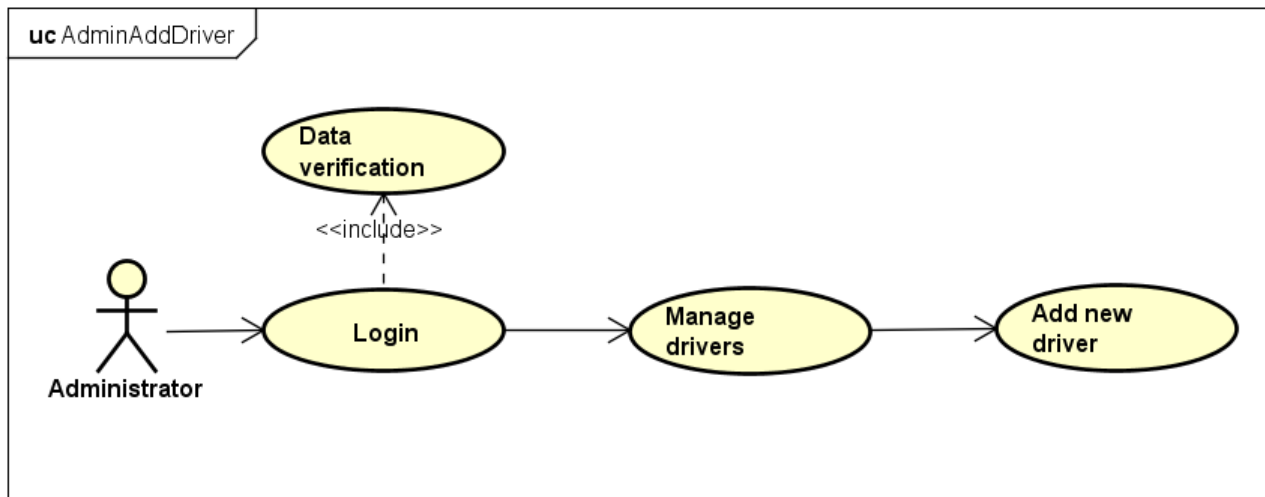
3.5.1.11 Taxi driver accepts or rejects a job with the mobile application

Actor	Taxi driver
Goal	[G17]
Input Condition	<ol style="list-style-type: none">1. The user must be already logged in.2. The user's availability must be set to "on".
Event Flow	<ol style="list-style-type: none">1. <i>myTaxiService</i> shows the home page for the driver.2. <i>myTaxiService</i> notifies the user about a job proposal that could fit him/her.3. The user chooses whether accepting or rejecting the proposal.
Output Condition	The system receives the setting of the user and operates accordingly.
Exceptions	NULL



3.5.1.12 System administrator manages the drivers list

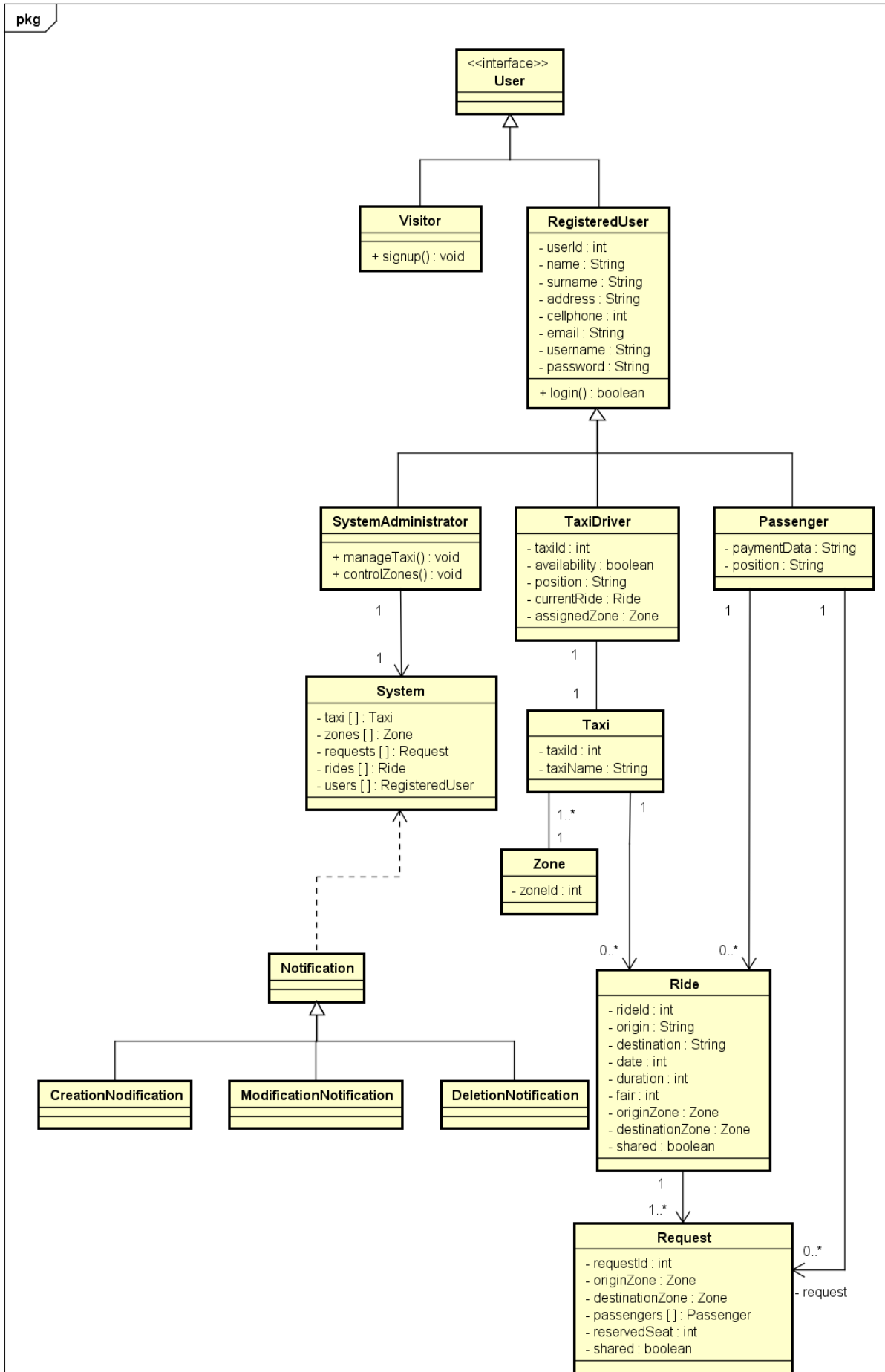
Actor	Administrator
Goal	[G18]
Input Condition	The user must be already logged in.
Event Flow	<ol style="list-style-type: none">1. <i>myTaxiService</i> shows the home page for the administrator.2. The user accesses the function to manage the taxi drivers.3. The system shows the list of drivers.4. The user decides to add, modify or delete a driver.
Output Condition	The system receives the setting of the user and operates accordingly.
Exceptions	NULL



3.5.1.13 System administrator monitors the situation from the control center view

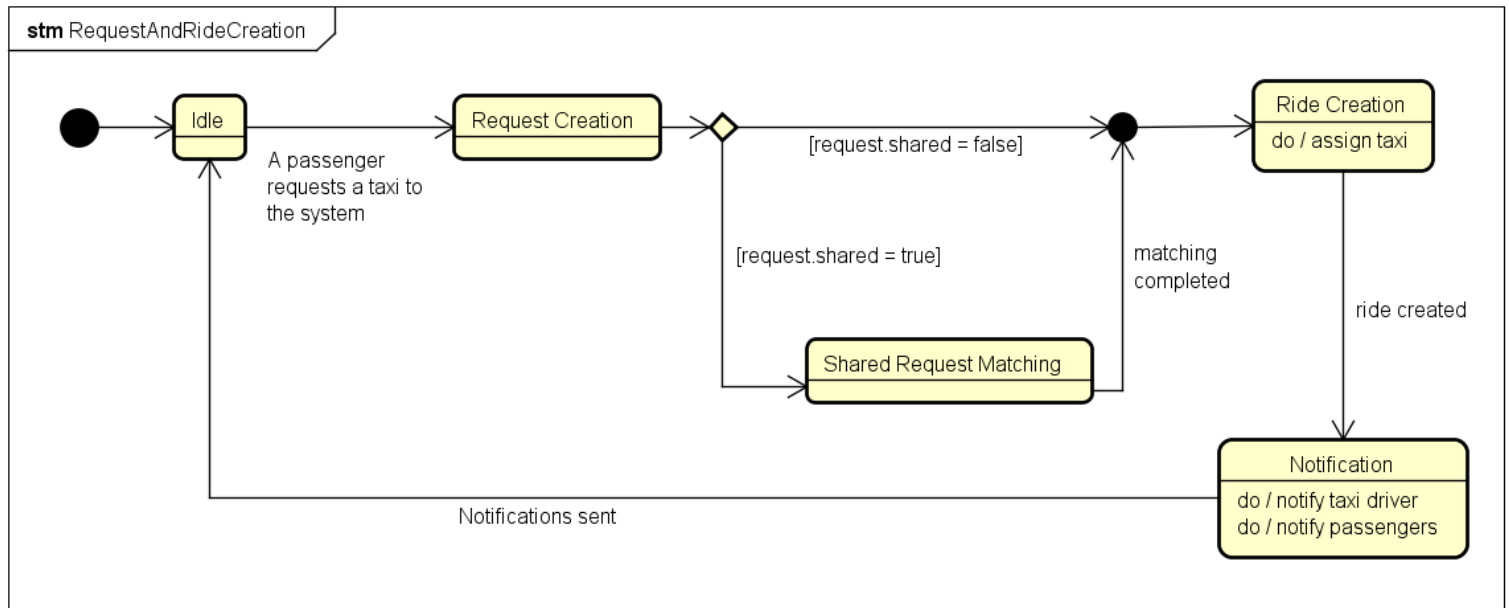
Actor	Administrator
Goal	[G19]
Input Condition	The user must be already logged in.
Event Flow	<ol style="list-style-type: none">1. <i>myTaxiService</i> shows the home page for the administrator.2. The user accesses system monitor.
Output Condition	The system shows the current situation of the service.
Exceptions	NULL

3.5.2 Class Diagrams

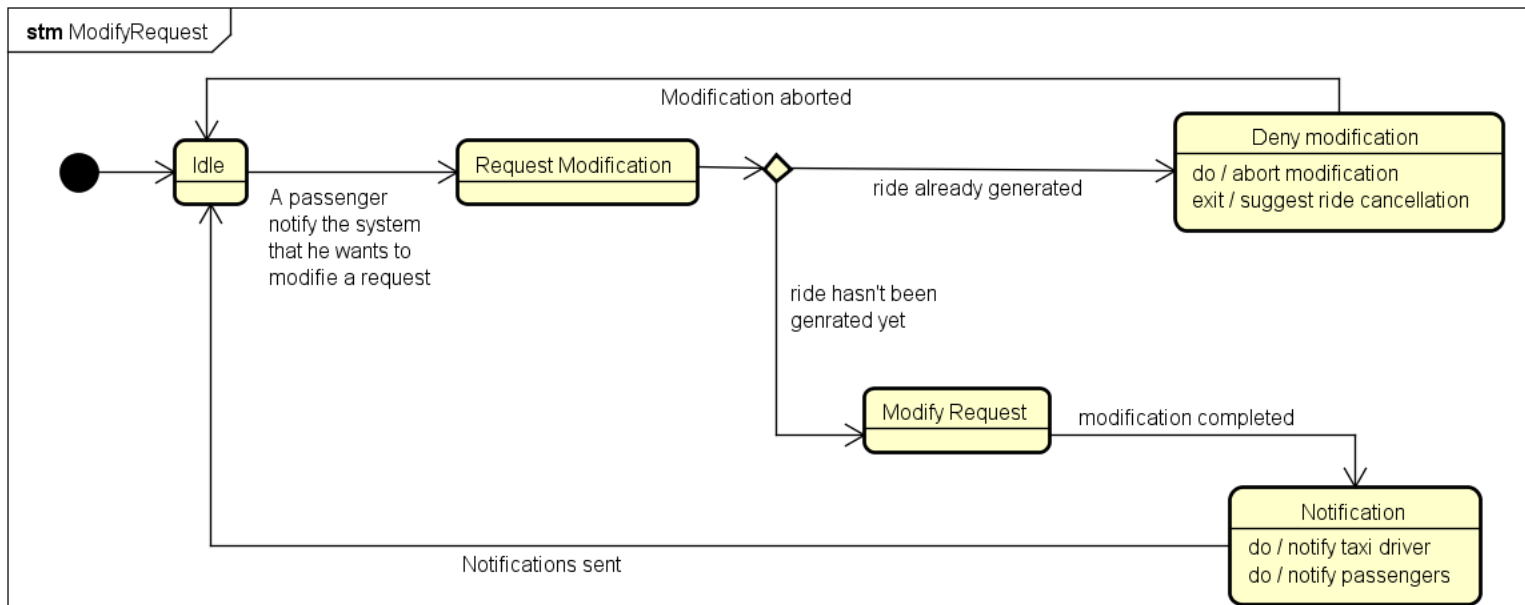


3.5.3 State Machine Diagrams

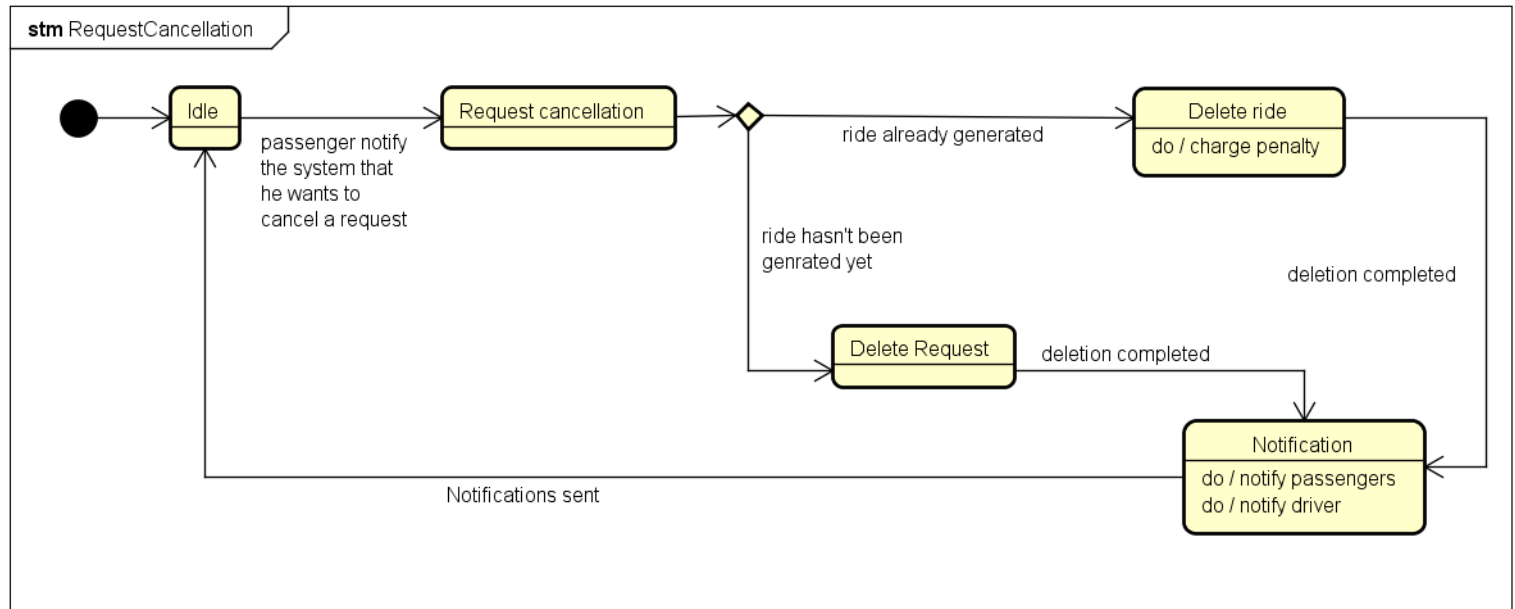
3.5.3.1 Ride creation The following state machine shows the steps of the process that controls the creation of the ride after the reception of the request or reservations. In particular it shows how the system takes into account the shared ride matching.



3.5.3.2 Modification of the request The following state machine shows the steps of the process that controls the modification of the ride. In particular it shows the different behavior depending on whether or not the ride has already been created.



3.5.3.3 Cancellation of the request The following state machine shows the steps of the process that controls the cancellation of the ride. In particular it shows the different behavior depending on whether or not the ride has already been created.



3.6 Non Functional Requirements

3.6.1 Performance Requirements

The service must guarantee an high level of usability, also in terms of processing and response time of the several functions. The objective is to avoid the situation in which the server represents the bottleneck of the entire system when there are many connections at the same time. To prevent this the server must be correctly chosen and calibrated to handle a large amount of connections in situations of heavy loads.

3.6.2 Software System Attributes

3.6.2.1 Availability & Reliability *myTaxiService* system must be always accessible. The server that hosts the service is online 24/7 and is equipped with an autonomous power supply backup system.

3.6.2.2 Security *myTaxiService* offers a high grade of privacy and security thanks to the protocols that it enforces. As already stated above it requires a 2-factor verification process during the registration; in this way the account is confirmed by two codes that are sent to both the provided email address and phone number. In this way it's verified that all the means used to communicate with the user really belong to him/her. The application also guarantees an appropriate level of confidentiality and integrity of the sensitive data exchanged on the Web with the server. The DB on the server is behind a DMZ in which is located the web server; firewalls with different rules are used to separate the different "zones" of access rights and permissions in the network.

3.6.2.3 Maintainability The software itself does not require any particular periodic update, unless the taxi company wants to add some functionalities. The only feature that could require some maintenance are the APIs offered by the system, but again, in this document and all the other ones related to *myTaxiService* that will be attached, future developers can find all what they could need to understand how the system works and then how to update some sections of the software.

4 Alloy

```
open util/boolean

sig string {}

sig Date {}

// Model definition
sig User {}

sig Visitor extends User {}

sig RegisteredUser extends User {
    userId : one Int ,
    name: one string ,
    surname: one string ,
    address: one string ,
    cellphone: one string ,
    username: one string ,
    password: one string ,
    mail: one string
}

sig SystemAdministrator extends RegisteredUser {
    system: one System
}

sig TaxiDriver extends RegisteredUser {
    availability: one Bool ,
    position: lone string ,
    currentRide: lone Ride ,
    taxi: one Taxi
}

sig Passenger extends RegisteredUser {
    paymentData : one string ,
    position : lone string ,
    currentRide: lone Ride ,
    requests: set Request
}

sig System {}

sig Taxi {
    taxiId: one Int ,
    taxiName: one string ,
    taxiDriver: one TaxiDriver ,
    currentZone: one Zone ,
    rides: set Ride
}
```

```
{
    taxiId > 0
}

sig Zone{
    zoneId: Int,
    taxis : set Taxi
}
{
    zoneId > 0
}

sig Ride{
    rideId: one Int,
    origin: one string,
    destination: one string,
    date : one Date,
    fair: one Int,
    originZone: one Zone,
    destinationZone: one Zone,
    shared: one Bool,
    associatedRequests: set Request,
    associatedPassengers : set Passenger
}
{
    rideId > 0
    fair > 0
}

sig Request{
    requestId: one Int,
    origin: one string,
    destination: one string,
    originZone: one Zone,
    destinationZone: one Zone,
    date: one Date,
    passengers: set Passenger,
    shared: one Bool
}
{
    requestId > 0
}

sig Notification{
    notificationId: one Int
}

sig CreationNotification extends Notification{}

sig ModificationNotification extends Notification{}
```

```
sig DeletionNotification extends Notification {}

// Model constraints

fact noEmptyRegisteredUser{
    all u: RegisteredUser | (#u.userId=1) and (#u.name=1) and (#u
        ↪ .surname=1) and (#u.username=1) and(#u.password=1) and
        ↪ (#u.mail=1)
}

fact noEmptySystemAdministrator{
    all s : SystemAdministrator | (#s.system=1)
}

fact noEmptyTaxiDriver{
    all t : TaxiDriver | (#t.availability=1) and (#t.taxi=1)
}

fact noEmptyPassenger{
    all p: Passenger | (#p.paymentData=1)
}

fact noEmptyTaxi{
    all t : Taxi | (#t.taxiId=1) and (#t.taxiName=1) and (#t.
        ↪ taxiDriver=1)
}

fact noEmptyZone{
    all z : Zone | (#z.zoneId=1)
}

fact noEmptyRide{
    all r : Ride | (#r.rideId=1) and (#r.origin=1) and (#r.
        ↪ destination=1) and (#r.date=1) and (#r.fair=1) and (#r.
        ↪ originZone=1) and (#r.destinationZone =1) and (#r.
        ↪ shared=1)
}

fact noEmptyRequest{
    all r : Request | (#r.requestId=1) and (#r.originZone=1) and (#r.
        ↪ destinationZone=1)
}

fact noDuplicateRegisteredUser{
    no disj u1,u2 : RegisteredUser | (u1.username = u2.username)
        ↪ and (u1.mail = u2.mail)
}

fact noDuplicateTaxi{
```

```
    no disj t1,t2 : Taxi | (t1.taxiId = t2.taxiId)
}

fact noDuplicateNotification{
    no disj n1,n2 : Notification | (n1.notificationId = n2.
    ↪ notificationId)
}

fact noDuplicateRequest{
    no disj r1,r2 : Request | (r1.requestId = r2.requestId)
}

fact noDuplicateRide{
    no disj r1,r2 : Ride | (r1.rideId = r2.rideId)
}

fact noDuplicateZone{
    no disj z1,z2 : Zone | (z1.zoneId = z2.zoneId)
}

fact NoMultipleTaxiPerTaxiDriver{
    no disj t1,t2 : Taxi | (t1.taxiDriver = t2.taxiDriver)
}

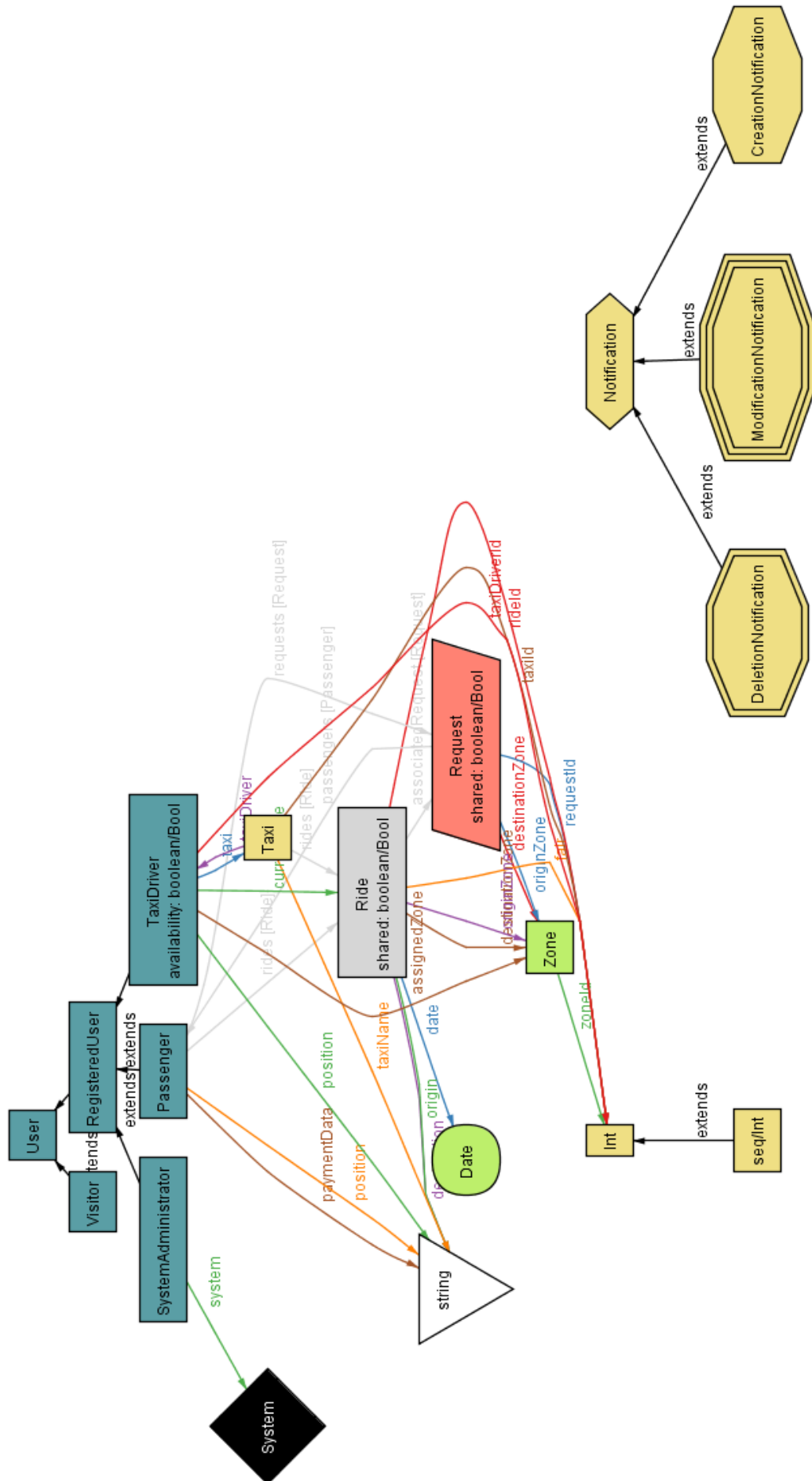
fact noMultipleCurrentRideForPassenger{
    all p1,p2 : Passenger | (p1=p2) implies (p1.currentRide = p2.
    ↪ currentRide)
}

fact noOriginEqualDestinationRequest{
    no disj r1,r2 : Request | (r1.origin = r2.destination)
}

fact noOriginEqualDestinationRide{
    no disj r1,r2 : Ride | (r1.origin = r2.destination)
}

pred show{}

run show for 5
```



5 Appendix

5.1 Software and tools used

- TeXstudio 2.10.4 (<http://www.texstudio.org/>) to redact and format this document.
- Astah Professional 7.0 (<http://astah.net/editions/professional>): to create Use Cases Diagrams, Sequence Diagrams, Class Diagrams and State Machine Diagrams.
- Alloy Analyzer 4.2 (<http://alloy.mit.edu/alloy/>) to prove the consistency of the model.
- Balsamiq Mockups 3.2.4 (<http://balsamiq.com/products/mockups/>) to create the mock-ups.

5.2 Hours of work

The time spent to redact this document:

- Baldassari Alessandro: 35 hours.
- Bendin Alberto: 35 hours.
- Giarola Francesco: 35 hours.