



Cascianelli Silvia

Baldi Roberto

REQUIREMENT ANALYSIS AND SPECIFICATION DOCUMENT

MyTaxiService

V 1.1

CONTENTS

1	Introduction	4
1.1	Purpose.....	4
1.2	Scope	4
1.3	Actors	6
1.4	Stakeholders.....	6
1.5	Goals.....	6
1.6	Definitions, Acronyms and Abbreviations.....	8
1.6.1	Definitions	8
1.6.2	Acronyms	9
1.6.3	Abbreviations	9
1.7	References.....	10
1.8	Document Overview.....	10
2	Overall Description	11
2.1	Product perspective	11
2.2	User characteristics	11
2.3	Constraints	11
2.3.1	Regulatory policies.....	11
2.3.2	Parallel operations	12
2.4	Assumptions.....	12
2.5	Future possible implementations	13
3	Specific Requirements	14
3.1	External interface requirements	14

3.1.1	User interfaces	14
3.1.2	Hardware interfaces	22
3.1.3	Software interfaces	22
3.2	Functional requirements	23
3.2.1	Taxi driver	24
3.2.2	Passenger	25
3.3	Non-functional requirements	27
3.3.1	Performance Requirements	27
3.4	Scenarios	27
3.5	Use Cases	32
3.6	Class Diagram	36
3.7	Sequence diagrams	37
3.8	Activity Diagrams	41
3.9	Alloy Modelling	44
3.9.1	Metamodel	47

1 INTRODUCTION

1.1 PURPOSE

This is a Requirement Analysis and Specification Document (RASD). It aims at providing a complete description of the system to be designed, developed and tested, defining all the objectives the system is going to pursue and modelling scenarios and use cases in order to accurately examine and represent the expected behavior. In particular it should allow to conveniently identify the functional and not-functional requirements as well as the constraints of the project and all the relevant domain properties, in such a way that the system accomplishes all the pre-established goals. Hence this document should supply designers, developers and testers all the necessary guidelines to achieve a correct and complete outcome with respect to the implementation and the evaluation of the project in question. As regards the intended audience of this documents, it includes the current and eventually future developers of the system, (since this latter will be open to integrate subsequent additions and evolutions), as well as all the other stakeholders and interested parties in such a system. In fact, this document is also supposed to be used as a contractual basis between the Customer, who commissions the project, and the developers.

1.2 SCOPE

The project we will focus on is called MyTaxiService and it concerns the design of a system which is able to manage conveniently and efficiently the taxi service of a large city. Hence this project aims principally at simplifying the access of passengers to the service by exploiting the Net, but it also guarantees a fair management of taxi queues. To be more precise, MyTaxiService grants a potential passenger the possibility of requesting a taxi either through a web application or a mobile one and it assures the applicant to be informed about the code of the incoming taxi and the waiting time, provided that the system succeeds in allocating a taxi

for the request. Moreover MyTaxiService provides also the opportunity to demand for a reservation, as long as the booking occurs at least two hours before the expected ride; in this case the passenger must specify origin and destination of the ride as well as the meeting time during the booking process. In this way, the system will proceed automatically to allocate a taxi for the reservation 10 minutes before the scheduled time; at that point, the applicant receives a confirmation with the code of the incoming taxi and any information concerning an eventual delay. However, if there are not available taxis the allocation procedure fails also for an already booked reservation; in this eventuality the negative outcome associated with the reservation is reported to the applicant through an email, to apologize for the disruption. Hence, in general, MyTaxiService works in order to handle efficiently the incoming taxi requests. How the system manages to do this will be extensively discussed later in the document. Nevertheless, essentially MyTaxiService arranges every available taxi in a specific queue associated with the zone of the city the taxi is located at that moment. Thus, the system can always forward an incoming request to the first taxi of the appropriate queue, according with the zone the origin of the ride belongs to and by using the special functionalities of the mobile app dedicated to drivers. Whether the chosen taxi driver does not accept the sent request MyTaxiService records a penalty for him and puts his taxi at the bottom of the local queue; at the same time the system proceeds by forwarding the request to the following taxi, which has just become the first of the queue. If none among the first three chosen drivers accepts the request the system forces the fourth one with a mandatory service call. In fact MyTaxiService is designed to be efficient and so it cannot let a request missed unless there are not available taxi both in the local queue and in all the adjacent zones. In fact, if an incoming request refers to a zone in which the queue of available taxis is momentarily empty, MyTaxiService provides a taxi by forwarding the request to the longer queue among the ones of the adjacent zones. To conclude this argument, it has to be noted that each queue is scrolled cyclically and so a driver could receive the same request more than once and finally should be forced to accept it. Finally, as far as taxi driver are concerned, each of them must use the mobile app, logging in at the beginning of each shift; actually through the app a driver can notify the availability every time a ride has been concluded and the system can place it in the

opportune queue depending on its actual GPS position. Besides of this a taxi driver has to use the mobile app to accept an incoming call, which the system has conveniently sent to him, and also to signalize the occurrence of an urgent problem.

1.3 ACTORS

- Taxi drivers: Through the mobile app they can use specific functionalities of myTaxiService reserved exclusively to them.
- Passengers: They can use the mobile app or the web application to request a taxi at that moment, or eventually to reserve a taxi at least two hours before the time they need it.

1.4 STAKEHOLDERS

The stakeholders, whom this document is addressed to, are first of all the customers who have commissioned the realization of this project, specifically in this case the local government. However, among the stakeholders of the project we have to include also the designers, developers, testers and analysts of the system in question as well as the final users, although these latter are probably mostly interested in a practical approach.

1.5 GOALS

First of all, the System should provide

- **[G1]** a way to simplify the access of passengers to the service and improve the communication with taxi drivers
- **[G2]** a fair management of taxi queues
- **[G2]** an efficient getting through of all the incoming requests and reservations, according with taxi availability.

In this sense, **through the mobile application**, MyTaxiService should allow **each taxi driver**:

- **[G4]** to log with his/her access credentials
- **[G5]** to receive quickly an incoming request conveniently selected and forwarded to him/her by the system, as long as his/her taxi state appears available.
- **[G6]** to answer immediately an incoming request in order to advise the system that he/she is going to take care of the call in question.
- **[G7]** to notify immediately his/her taxi availability each time he/she concludes a ride or at the beginning of his/her working shift
- **[G8]** To notify immediately he/she is no more available if someone demands for a lift in loco, while the driver is waiting for a service call.
- **[G9]** to notify a brief interval of non-availability without incurring in penalization, as long as the amount of breaking time during a working shift last at most 7 minutes.
- **[G10]** to notify in real-time any kind of glitch or delay takes place in such a way that the system can analyze the occurred problem and eventually inform the customer.
- **[G11]** to be informed as soon as possible if a ride he/she is going to take care of has just been deleted

On the other hand, either **through the mobile or web application**, MyTaxiService should ensure that **each potential passenger**:

- **[G12]** can request a taxi
- **[G13]** receives an immediate confirmation regarding the outcome of his/her request; in particular whether the system succeeds in finding a taxi the confirmation must include detailed information about the code of the incoming taxi and the estimated waiting time.
- **[G14]** can call the request off as soon as he/she receives the confirmation of a successful outcome.
- **[G15]** can specify to have a special exigence or a number of passengers that requires to allocate two taxis instead of only one.
- **[G16]** can book a taxi reservation at least two hours before the desired meeting time

- **[G17]** receives a notification indicating his/her reservation has been accepted as soon as he/she completes the booking process; this notification contains also a code for the booking has been made. Anyway this notification does not assure the subsequent provision of the service
- **[G18]** receives a confirmation or a failure notice email as regards an already accepted reservation, ten minutes before the arranged time of the ride. In the first case the confirmation must include the code of the incoming taxi and detailed information about an eventual delay, whereas the failure notice consists in a message to apologize for the non-provision of the service.
- **[G19]** can cancel a reservation using the booking code or eventually can call the request off when he/she receives the confirmation of successful allocation

Finally, MyTaxiService should:

- **[G20]** Try to avoid to work on fakes or on duplication of a single request caused by errors or applicant impatience
- **[G21]** Discourage a taxi driver from not answering or from working badly
- **[G22]** supply a programmatic interface to enable the development of subsequent additional services on top of the basic one.

1.6 DEFINITIONS, ACRONYMS AND ABBREVIATIONS

1.6.1 Definitions

Taxi driver: a kind of user that, through the mobile app, can access his/her private area and utilize special functionalities of MyTaxiService, usable only by this type of user.

Passenger: the other kind of user, that can utilize MyTaxiService both through the mobile app or the web application; in particular these users are interested in exploiting the application to look for a taxi easily and quickly.

Local queue: This term refers to the organization of the city in different zone to cover with the taxi service; in effect a local queue is supposed to be the ordered set of available taxi associated to a specific zone; in particular this relation is based on the GPS current position of each available taxi.

Request: This term indicates that a passenger is requiring a ride because he/she needs a lift at the current moment

Reservation: This term indicates a ride that a passenger want to book for a certain moment successive with respect to the sending time. Precisely the required time must be at least two hours later the booking.

Unsuccessful call: This terms indicates a call forwarded by the system to a chosen taxi driver, who simply does not answer and consequently does not accept the ride associated with that call.

1.6.2 Acronyms

- RASD: Requirements Analysis and Specification Document.
- DB: Data Base.
- DBMS: Data Base Management System.
- API: Application Programming Interface.
- OS: Operating System.
- JVM: Java Virtual Machine.
- JEE: Java Enterprise Edition.

1.6.3 Abbreviations

- [Gn]: n-goal.
- [Rn]: n-functional requirement.
- [Dn]: n-domain assumption.

1.7 REFERENCES

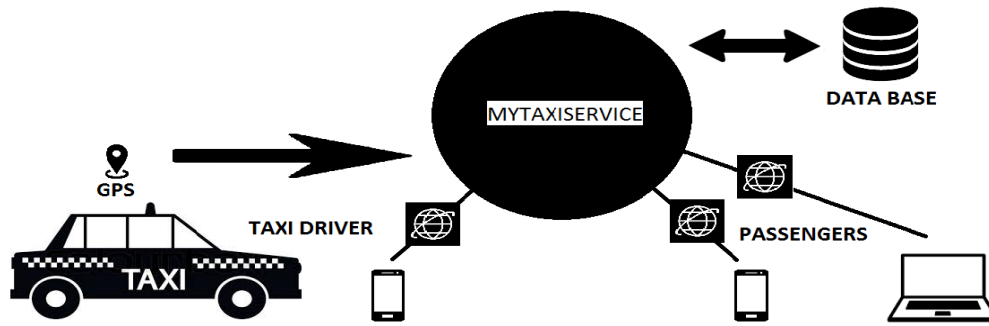
- Specification Document: MyTaxiService Project AA 2015-2016.pdf.
- IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications.
- IEEE Std 1016tm-2009 Standard for information Tecnology-System Design-Software Design Descriptions.

1.8 DOCUMENT OVERVIEW

The next chapter of this document, the Overall Description section, gives an overview of the product describing the informal requirements and the context in which this product operates. In particular it highlights the relevant assumptions concerning the environment and any constraint which limits; hence it represents also a baseline for the technical requirements specification presented in the following chapter. Precisely the third part , the Requirements Specification section, is written primarily for the developers and it provides a more technical explanation for the functionalities of the product as well as a clarification of all the non functional requirements, especially in terms of different kind of interfaces to use . Hence both sections of the document describe the same software product in its entirety, but are intended for different audiences and thus use different language and approaches.

2 OVERALL DESCRIPTION

2.1 PRODUCT PERSPECTIVE



2.2 USER CHARACTERISTICS

Both type of final users, taxi drivers and passengers must be able to use a smartphone to download and utilize MyTaxiService app; eventually a passenger can choose to use a web browser and exploit MyTaxiService web application. Through MyTaxiService a passenger can easily look for a taxi without spending time or money for a phone call and, on the other side, a taxi driver can simplify his/her communication with the enterprise he/she works for.

2.3 CONSTRAINTS

First of all, in order to use successfully either the web application or the mobile app the users must be connected to the Network.

2.3.1 Regulatory policies

MyTaxiService must be subject to regulatory policies concerning:

- the handling of personal data pursuant to the Personal Data Protection Code;

- the exploiting of the software licenses as regards the developed code
- the use of the Network according to the rules of the contract stipulated with an Internet Service Provider

2.3.2 Parallel operations

MyTaxiService must supports parallel operations caused by simultaneous actions undertook from different users.

2.4 ASSUMPTIONS

As regards the relevant domain properties it is assumed that:

- **[D1]** Each taxi driver has an enterprise smartphone
- **[D2]** The enterprise grants a personal taxi identifier and password to each driver
- **[D3]** Each taxi driver logs in at the beginning of his working shift
- **[D4]** The number of taxis is sufficient to provide conveniently the city in question with the service
- **[D5]** Each taxi and each taxi driver of the enterprise respects all the rules concerning licenses, work contract, assurances and so on.
- **[D6]** All the information about taxis, taxi drivers, local queues as well as requests and reservations are properly stored in a DB, accessible also for any kind of control as regards taxis activities
- **[D7]** The city plan is mapped into a grid which can include all the areas despite of the city is not obviously a real rectangle
- **[D8]** Each taxi has a fully functioning GPS tracker, which is also properly connected to the system
- **[D9]** The system is able to receive continuously all the information coming from the external GPS tracker system
- **[D10]** Each taxi can transport a maximum of four passengers

- **[D11]** Each available taxi driver can eventually accept to pick a bystander up if this latter asks for a lift in loco
- **[D12]** Each taxi driver remembers to update conveniently his/her taxi state, if necessary
- **[D13]** Only the drivers of this urban enterprise are able to access and use the private area to receive assignments
- **[D14]** The e mail addresses inserted by applicants are valid

2.5 FUTURE POSSIBLE IMPLEMENTATIONS

MyTaxiService should be designed and developed with a particular attention to maintain it open for future possible evolution or additions, either in terms of adopted solutions and dedicated hardware either as regards to the implementation of new functionalities. For instance the development of the additional service of taxi sharing on top of the basic one.

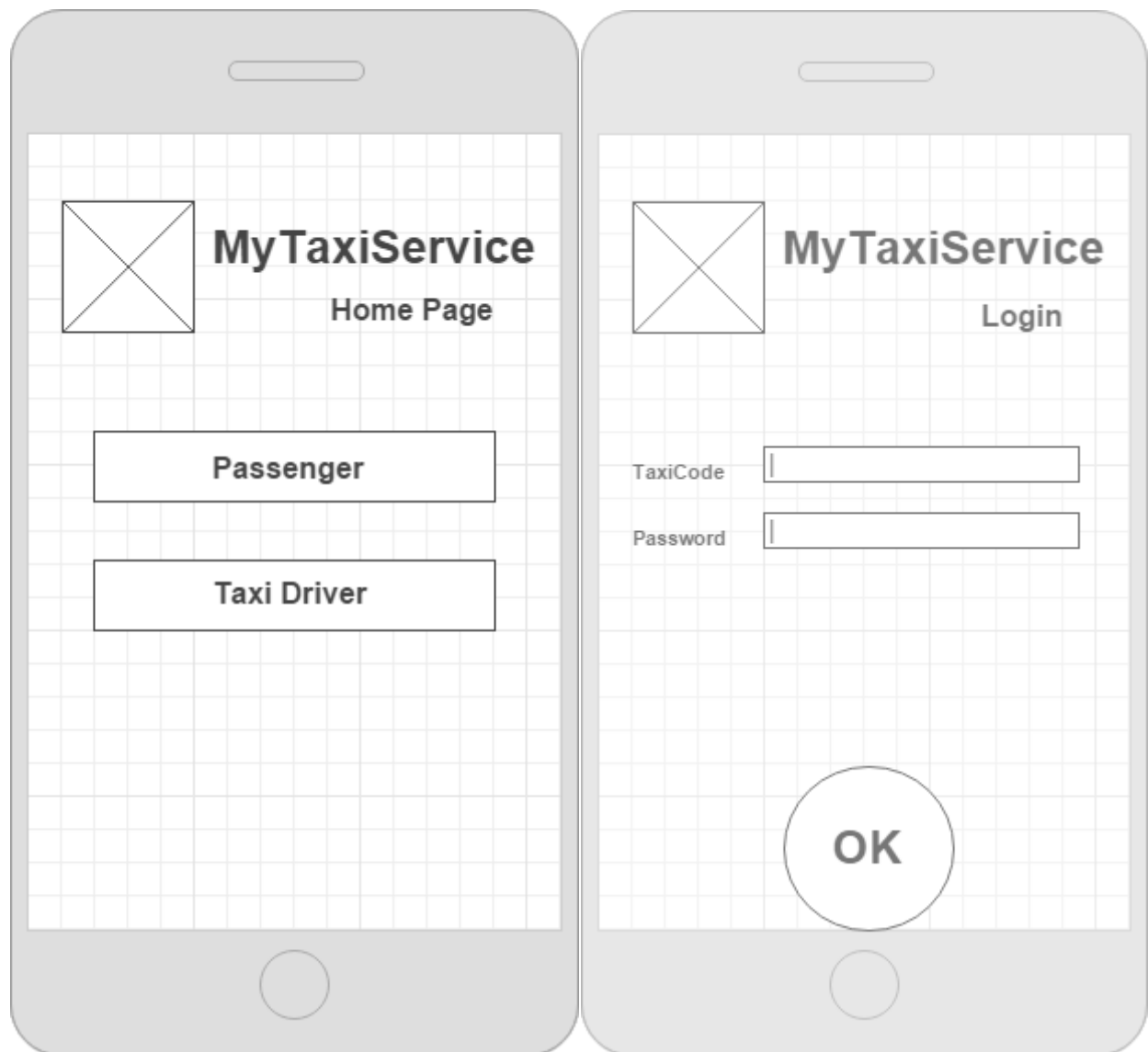
3 SPECIFIC REQUIREMENTS

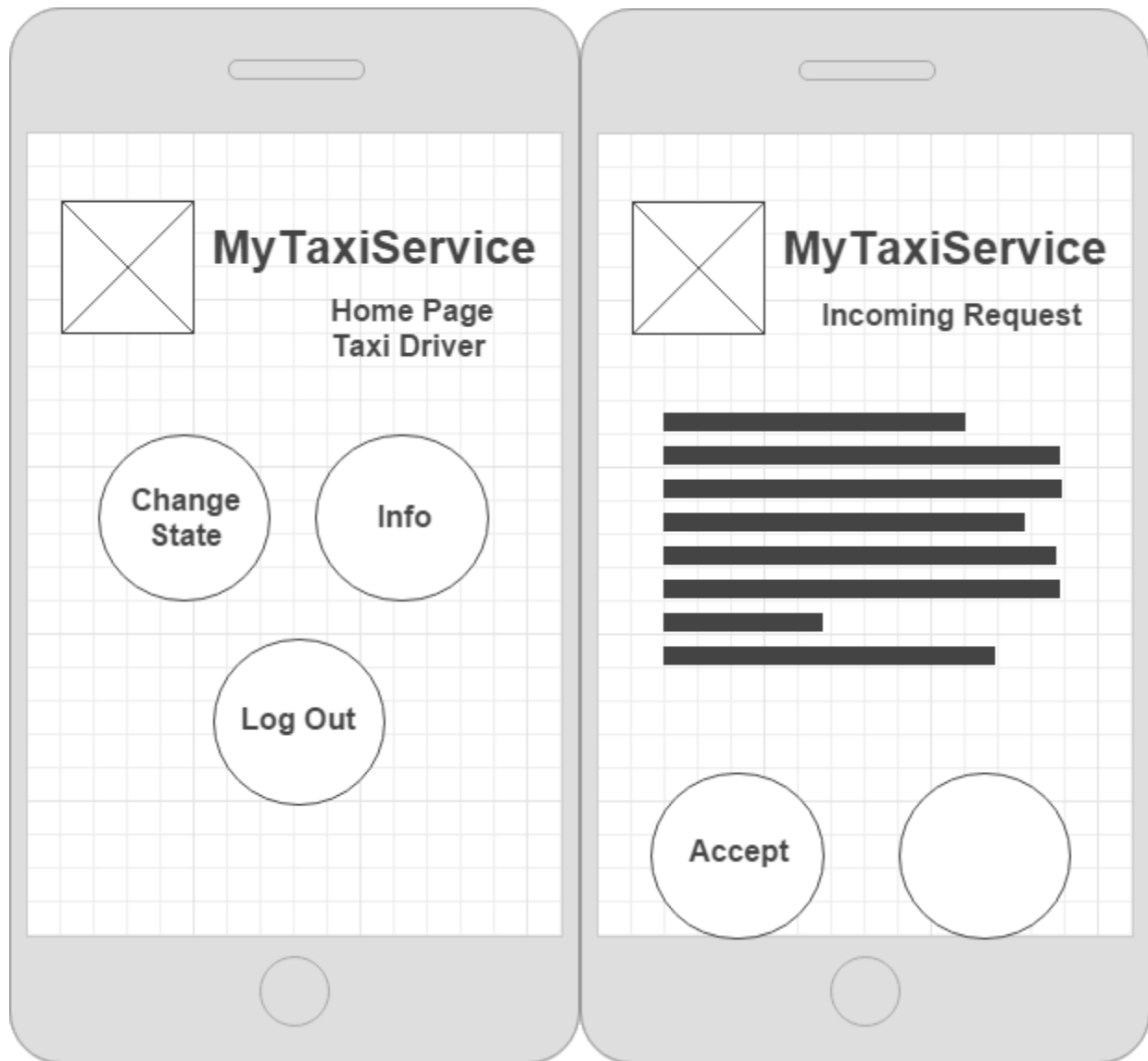
3.1 EXTERNAL INTERFACE REQUIREMENTS

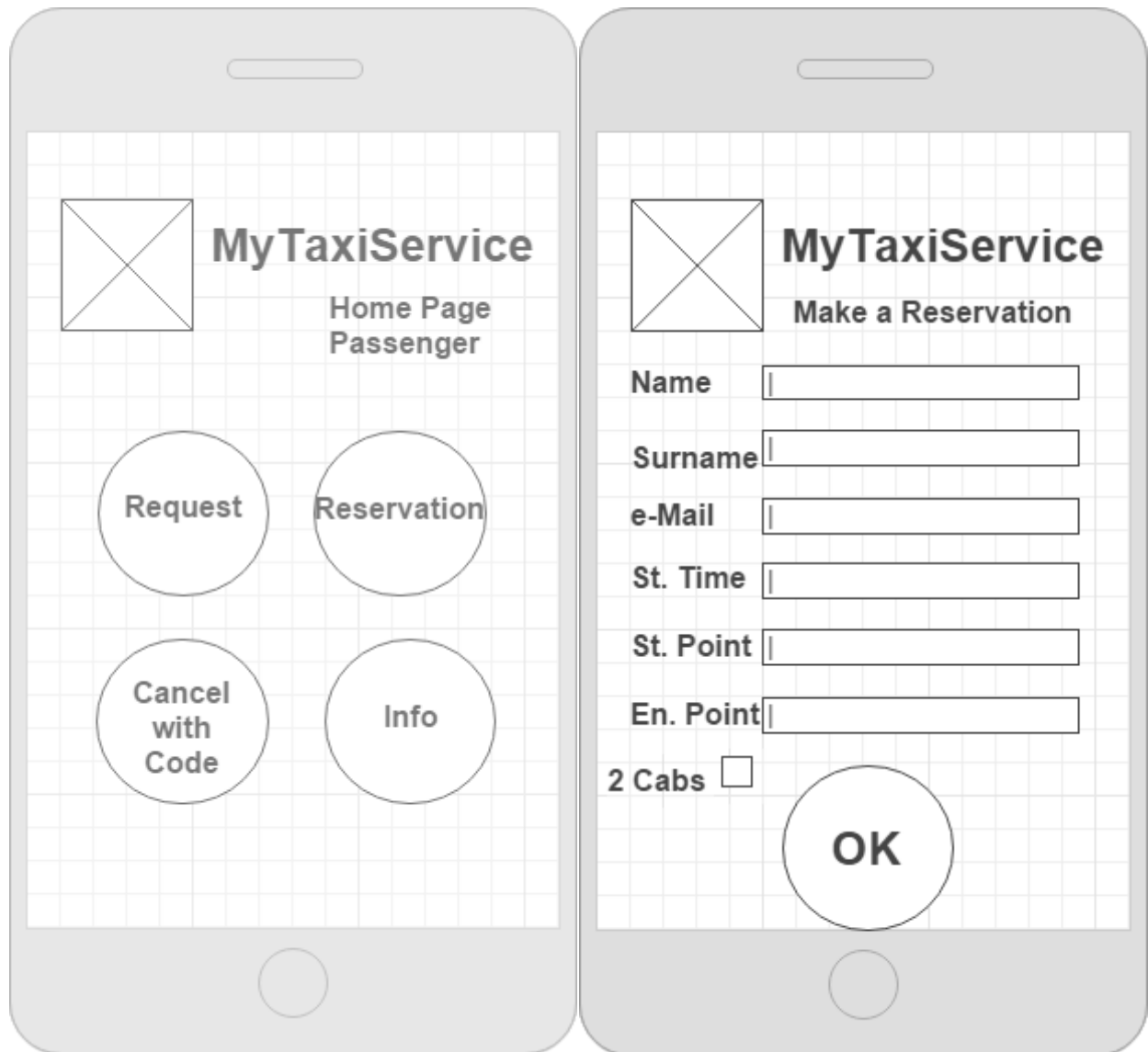
3.1.1 User interfaces


- **[R1]** Both the web application and the mobile app must have user-friendly interfaces with intuitive commands, easily usable. Moreover, the applications aim at granting a trivial to use real-time connection between potential passengers, who want to enjoy the service, and taxi drivers, that provide it. ([G1])

In this sense, here are presented some mockups that represent an idea about the mobile app screens or of the web application pages :









MyTaxiService

Make a Request

Name


Surname

e-Mail

Address

2 Cabs ☐

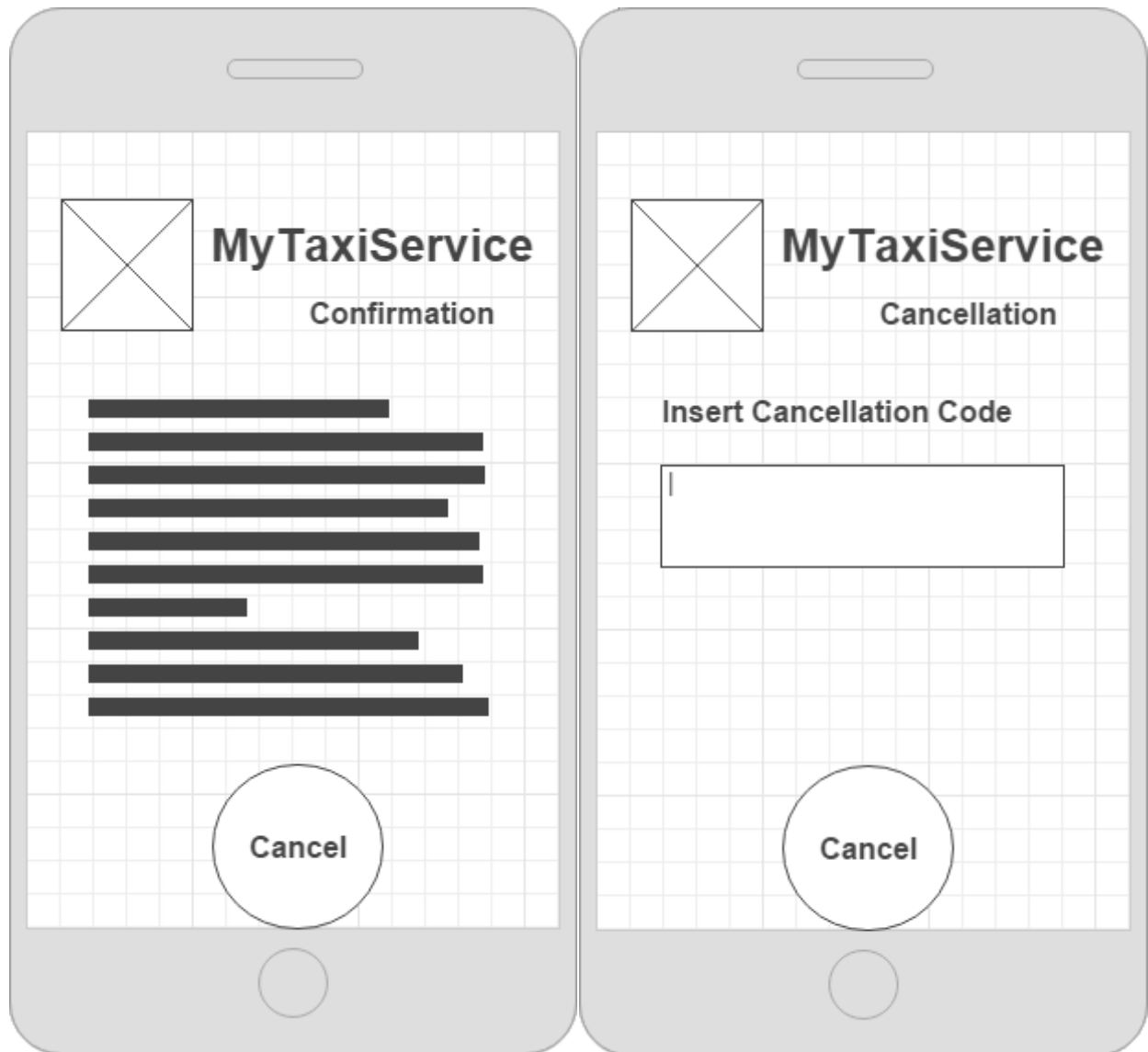
OK



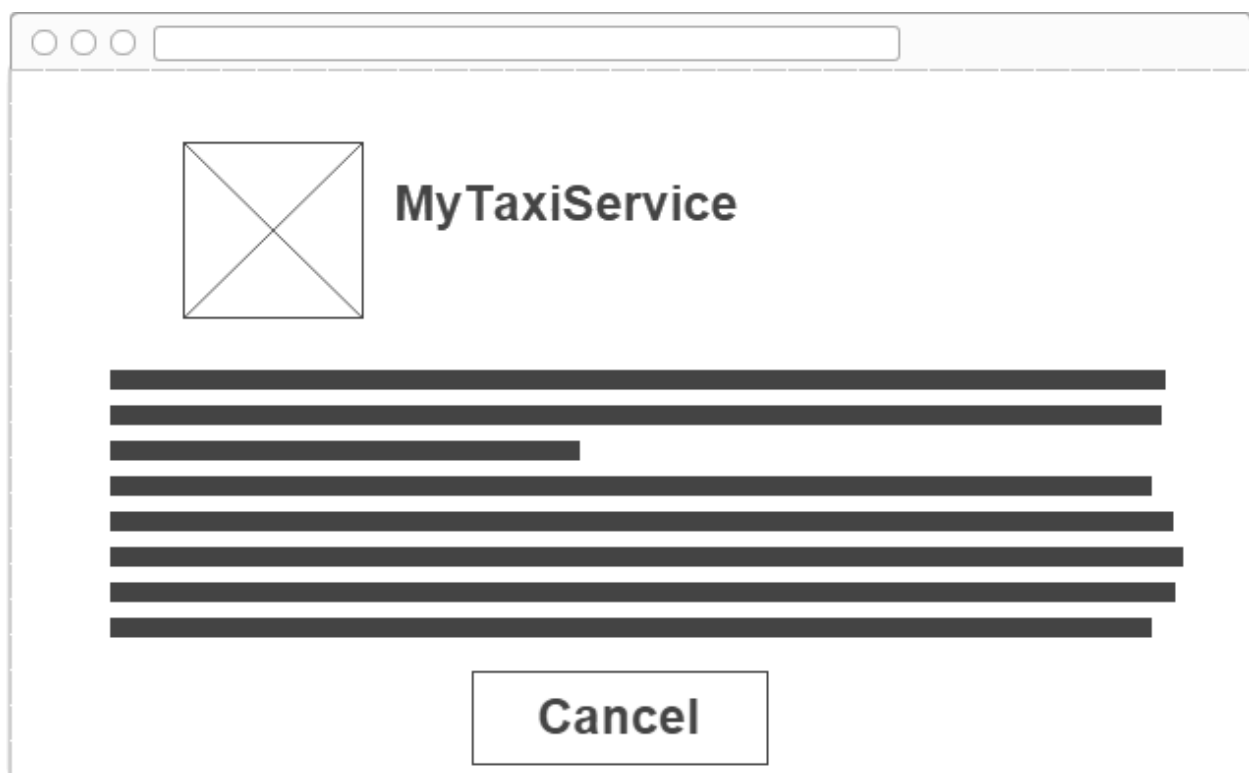
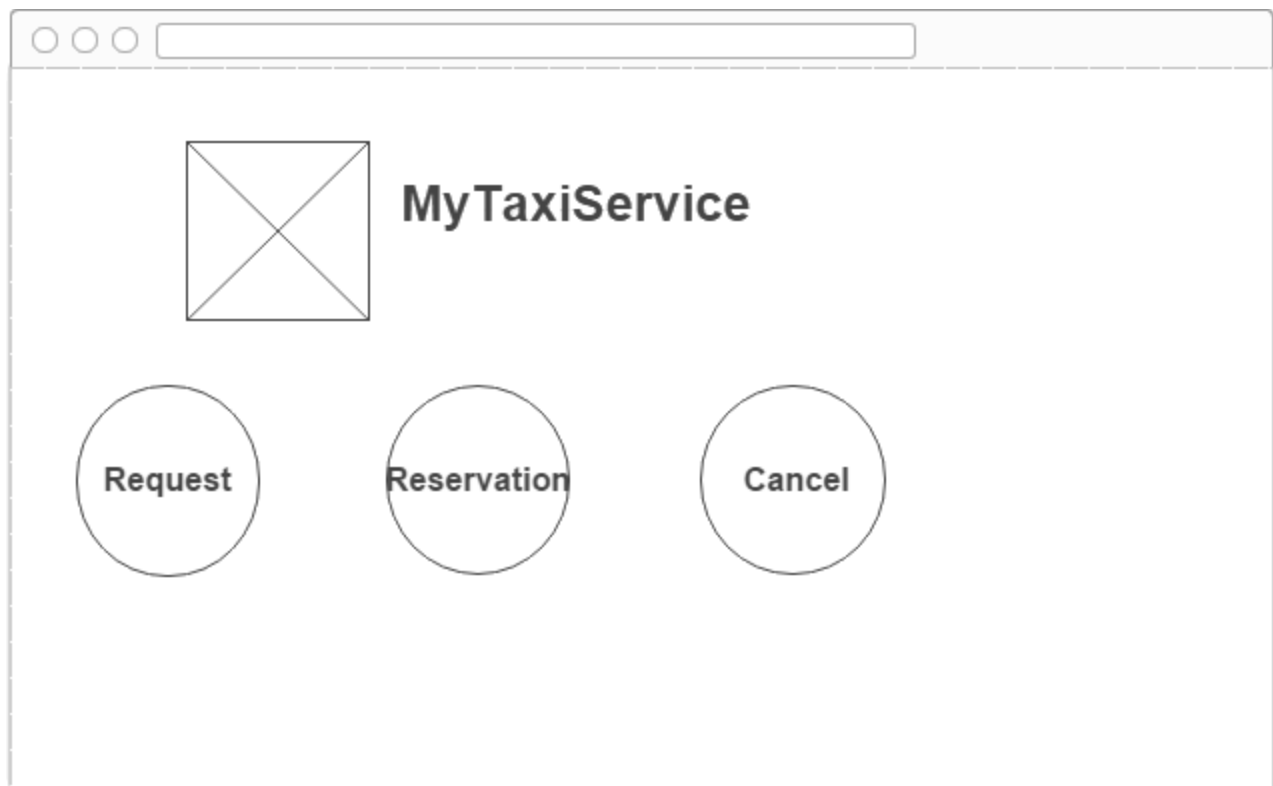
MyTaxiService

Confirm a Reservation

Cancellation Code







3.1.2 Hardware interfaces

- **[R2]** MyTaxiService must interact with the GPS tracker system that computes the exact position of each cab instant by instant.([G2])

3.1.3 Software interfaces

- **[R3]** MyTaxiService must interact with the DBMS of a data base containing all the relevant information about the taxis. Among them the mutable attributes about each taxi concern its state, its current position and the amount of penalties of its driver.

- Open Source Relational Database Management System (DBMS):
 - Name: MySQL.
 - Version: 5.7
 - Source: <http://www.mysql.it/>
- Java Virtual Machine (JVM).
 - Name: JEE
 - Version: 7
 - Source: <http://www.oracle.com/technetwork/java/javaee/tech/index.html>

Java Enterprise Edition (JEE), formerly Java 2 Enterprise Edition or J2EE, is a collection of APIs which are not included in the standard JDK but provide functionality which is useful for many server applications.

- Open Source Application server: (to support a distributed environment)
 - Name: Glassfish.
 - Version: 4.1.
 - Source: <https://glassfish.java.net/>
- Operating System (OS).
MyTaxiService web application as well as the mobile app must be able to run on any OS which supports JRE.

3.2 FUNCTIONAL REQUIREMENTS

In order to achieve all the pre-established goals the functional requirements are strictly connected with them, as it will be clear in the following statements:

- **[R4]** MyTaxiService must consider the city as a grid of squared zone. Each zone encloses an area of 2km^2 and thus has a maximum of 8 adjacent zones. Every available taxi is appended by the system at the bottom of the local queue, depending on the GPS position of the taxi at that moment. **([G2])**
- **[R5]**. MyTaxiService must handle all the simultaneous requests and reservations of different passengers and must evaluate the addresses specified in each of them; in this way the system forwards a call to the proper cab, according with the local taxi queue suitable for accomplishing that assignment. **([G3])**
- **[R6]** MyTaxiService starts a new taxi search for an already allocated call if the driver who has accepted to take care of that assignment notifies any kind of problem through the app. **([G3])**
- **[R7]** MyTaxiService refuses an incoming request or fails the automatic taxi allocation associated with a previous reservation, if and only if both the local queue suitable for that ride and all the adjacent ones are empty at that moment. **([G3])**
- **[R8]** Every time a taxi search regards a zone where the local queue is not empty the allocation must be achieved successfully, in the worst case through a mandatory assignment. This rule holds also if a ride has the origin that belongs to a zone where the local taxi queue is empty at the moment, as long as at least one of the adjacent zone has an available taxi. However, in this case the system forwards the call to the longer local queue among the ones of the adjacent zones. **([G3])**
- **[R9]** When the first taxi driver of a local queue does not accept a call, this attempt of taxi allocation fails; so MyTaxiService proceeds immediately by forwarding the same incoming call to the following taxi of the queue; actually for the system this latter has become the new first taxi of the queue whereas the old one is demoted at the bottom. This procedure works cyclically on the queue, so the same taxi can receive the same call

more than once or even in the end it should be obliged to accept it. In fact, the system counts the times it has already forwarded the call in question and, if none of the first three taxis has accepted the ride, MyTaxiService sends a mandatory assignment to the following available cab in order to assure to accomplish the allocation. ([G3])

- **[R10]** MyTaxiService cannot accept more than a request with the same credentials and the same origin address for a while of 30 minutes ([G20])
- **[R11]** MyTaxiService cannot accept another reservation with the same credential indicating a time that does not differ from the other one of at least one hour([G20])

3.2.1 Taxi driver

It is necessary to introduce other requirements which could guarantee the interaction with taxi drivers and the knowledge of the taxi states instant by instant.

- **[R12]**MyTaxiService mobile app must have a specific area dedicated to the taxi drivers of the enterprise. This part of the app is accessible only through a log in operation; hence at the beginning of each shift, the driver must fill in two fields, respectively with his taxi identifier and his password. As soon as the log in concludes successfully, the system sets the state of the taxi in question automatically as available. Therefore, MyTaxiService uses the information obtained by the GPS tracker of the taxi to arrange it in the proper local queue. ([G4][G7])
- **[R13]** After the log in operation MyTaxiService app allows a taxi driver to enter his private area. Here there are command that permit the driver to change the state of his cab, choosing among “not available”, “available” or “occupied”. ([G7][G8][G9])
- **[R14]** Only the system can associate a taxi also with another possible state which is “unusable”. This state is set by the system if and only if the driver notifies a glitch or any kind of problem by using the special command “URGENT PROBLEM”. This command also allows the driver to be phoned directly by a responsible for problem solving. Anyway, the cab remains in “unusable” state until the hitch is concretely worked out. ([G10])
- **[R15]** The system tries to combine an assignment with a taxi basically by forwarding an incoming call on the MyTaxiService app of the chosen driver. This “false” call lasts 30

seconds and during this while a driver should accept simply by tapping and dragging the icon of ACCETP REQUEST right. Although this call does not open any conversation, immediately after the acceptance the driver receives a notification on the app that specifies origin of the ride and personal data of the applicant to make sure of his/her identity. The notification may inform the driver in case he is going to take care of a previously scheduled reservation. **([G5][G6])**

- **[R16]** The state associated with a taxi changes automatically to “occupied” every time the driver accepts an incoming call from the system. **([G6])**
- **[R17]** Every time a passenger uses the proper command on the web page or on the app to cancel a ride that a taxi driver has already accepted, this latter is immediately informed by the system with a notification through MyTaxiService app. **([G11])**
- **[R18]** MyTaxiService monitors a taxi which state is not available. A taxi can use this state for a maximum of 7 minutes during a working shift. If this time has run out and the state is still not available the system assigns a great penalty to the driver, updating the appropriate counter in the DB. **([G21])**
- **[R19]** MyTaxiService records every unsuccessful “call”, and the taxi in question is moved back from the first position to the bottom of the local queue in which is located. Besides of this, the system punishes the taxi driver by adding a penalty on the appropriate counter in the DB.

3.2.2 Passenger

The following requirements concern the functionalities that MyTaxiService grants to passengers

- **[R20]** Both the web application and the mobile app of MyTaxiService provides the passenger a specific form to fill out in order to request a taxi; its fields are the origin of the ride and some personal data of the applicant, actually the name, surname, and a valid email address, all necessary to complete the request. Moreover, the passenger may tick a box to specify the need of two taxis, because of the number of actual passengers or for any other special exigence. **([G12], [G15])**

- **[R21]** MyTaxiService can allocate two taxis for a specific ride with respect to the specification in the request/reservation itself. Anyway the system allocates the taxis as well as for two different rides with the only constraint that the search is confirmed and successfully concluded if and only if both the taxis have been found. Hence, if only one of the two taxis is found the taxi driver who has already accepted the call is quickly informed through a notification that the ride must be deleted. **([G15])**
- **[R22]** As soon as a taxi has been found for a certain ride, a confirmation screen or a notification is provided to the passenger, respectively through the web application or the mobile app. This confirmation contains the code of the incoming taxi and the expected waiting time, as well as a command that enables the passenger to cancel immediately the just allocated ride; moreover, in case of web application, the information about the ride to take place are sent also by e-mail. **([G13], [G14])**
- **[R23]** Both the web application and the mobile app of MyTaxiService provides the passenger a specific form to book a taxi reservation; It includes name surname, e-mail address of the applicant as well as the origin of the ride, as usual, but additionally it is necessary to indicate destination and meeting time to schedule. Furthermore MyTaxiService allows a passenger to choose a meeting time which is at least two hours later than the time he/she performs the booking. Finally the booking process ends showing a screen or sending a notification that confirms uniquely the receipt of that reservation and that indicates a code associated with that booking. **([G16])(G17)**
- **[R24]** Both the web application and the mobile app have an area in which a passenger can insert a booking code to call that reservation off **([G19])**
- **[R25]** After a successful allocation of a taxi associated with a previous reservation, the passenger receives a notification on the app or an email, respectively if he/she used the mobile app or the web application for the booking process. However these messages contain a confirmation for the ride and indicate the code of the incoming taxi as well as a possible delay with respect to the scheduled time. Moreover, the email contains a link to cancel the ride whereas the app provides as usual a command to cancel it as well.

Finally, if the allocation fails, the message via notification or email apologizes for the non-provision of the expected service. **[[G18] [G19]]**

3.3 NON-FUNCTIONAL REQUIREMENTS

3.3.1 Performance Requirements

Performance must be acceptable to guarantee a good grade of usability. We assume the response time of the system is in general close to zero, so the performance are essentially bounded by user internet connections.

Anyway the taxi search operation is strictly related to real allocation attempts, hence for this kind of operation the system tries to assure a response time within 5 minutes from the receipt.

3.4 SCENARIOS

Scenario 1

Luca has to go to the airport, but he does not want to leave his car in the airport park; so he downloads MyTaxiService app on his smartphone and uses the application to request a taxi, specifying in the mandatory blanks his name, surname and a valid email address as well as the origin of the ride. As soon as the system finds an available taxi, Luca receives a confirmation with the code of the incoming taxi and the waiting time for its arrival.

Scenario 2

After few days, Luca has to come back home, so, before his return flight departure, he decides to use MyTaxiService app to reserve a cab that picks him up at the airport, three hours later. He looks for the special area of the app dedicated to the reservation service; here he fills in the fields regarding his name surname and e-mail last and he decides to insert also his mobile number this time. Moreover, since he is demanding for a taxi reservation, he has to specify the origin and destination of the ride as well as the desired meeting time. At this point, he receives

simply a confirmation of receipt of reservation, in which there is a code associated with his reservation. Later, ten minutes before the time scheduled during the booking, Luca receives a new email from MyTaxiService; so, looking at his mailbox he reads the confirmation email containing the code of the incoming taxi.

OPT: Around 30 minutes before the time scheduled during the booking, Luca has already landed and he is waiting for his luggage at the baggage hall of the airport; here, unexpectedly, he runs into his friend Alessandro, who offers him a lift to come back home together. Thus, Luca opens MyTaxiService App and call his reservation off in the area dedicated to cancellation, by using the reservation code which was enclosed in the email of the confirmation of receipt.

Scenario 3

This morning, as usual, Chiara has to go to work; unfortunately her car will not start, so she uses her laptop to look for a cab on MyTaxiService web application. Hence, she compiles the requesting section, by inserting her personal data (name, surname, e-mail) and her address as origin of the ride. As soon as the system finds a taxi for her zone, she receives a confirmation that advises her about the code of the incoming taxi and the expected waiting time of 20 minutes due to the traffic.

ALT1: Although she thinks she will be late at work because of this long waiting time, she does not have other opportunities and so she waits the incoming taxi.

ALT2: As she thinks 20 minutes of waiting are too many in order to arrive at work in time, she annuls the request using the specific command and calls his colleague Luca to ask for a lift.

Scenario 4

Andrea is a taxi driver who works for the company that uses MyTaxiService to handle the taxi service. So, this afternoon, at the beginning of his shift, he takes his enterprise smartphone and

he logs in by using his credentials, actually his taxi identifier and his own password, in order to enter his private area. After the log in the state of his cab is set as available by default; therefore, the system puts Andrea's cab at the bottom of the queue of the zone in which his taxi is located at the moment, referring to the position obtained by the cab GPS tracker. At a certain point, while Andrea is waiting for a request to take care of, he receives a service call, conveniently forwarded to him by the system.

ALT1: He accepts it simply by answering the service call on his MyTaxiService app. As a result, the system sends him a notification with the address where he has to pick the passenger up and appends also the personal information of the applicant for any useful purpose. Moreover, since Andrea has accepted the ride, the system automatically set the state of Andrea's cab as occupied, in order to prevent it from being associated to any queue of available taxi, until any change occurs. Finally, as soon as Andrea ends his ride, he notifies the system his availability, by changing his taxi state through the app.

ALT2: This time, Andrea decides not to accept the ride because he wants a less stressful shift. Hence, he ignores the incoming call. Anyway this choice has some consequences; indeed he obtains a penalty to be recorded on the appropriate counter in the DB and besides of this, his taxi is demoted at the bottom of the local queue it belongs to.

Scenario 5

Andrea carries on working hard every day, accepting promptly every incoming request, but today he feels it will be his lucky day. So, while is waiting for a ride to accomplish, he decides to change his state to "not available" and gets out of his cab to buy a lottery ticket.

ALT1: After this brief break he gets into his cab, but, while he is going to set his state back to available, a bystander comes across Andrea's taxi. The bystander, who needs a lift for the mall, demands for a ride and Andrea accepts to pick him up to the mall. So, first of all, Andrea sets his taxi state as occupied and then he starts the ride.

ALT2: During this break Andrea runs into his old friend Chiara. A long while has passed since they met last time, so they chat for about 10 minutes. Unfortunately, when Andrea gets back into his cab he has received a notification on the app, advising him that he has been penalized because his taxi state is still not available after more than the maximum admissible of 7 minutes for shift.

Scenario 6

After some months, while Andrea is waiting for a service call in his taxi, his cab undergoes a collision. Since Andrea's taxi was still the incident is not so serious, anyway the vehicle shows some damages at the car body. In this situation, Andrea must advise his superiors about the occurred problem and has to stop answering to any eventual request forwarded by the system. Hence, he opens the application and uses the command URGENT PROBLEM on his app to be phoned by a responsible as soon as possible. In such a circumstance, the system automatically proceeds to set the state of Andrea's taxi as UNUSABLE, indicating the cab has some kind of hitch.

Scenario 7

Lidia wants to visit her grandmother but she does not have a driving license, so she decides to ask for a cab; she takes her smartphone and downloads MyTaxiService app. Then she moves to the request section and here she inserts her data and the origin address of the desired ride.

When MyTaxiService starts to analyse the local queue, according with the address indicated by Lidia, it finds out that the proper queue is empty; hence, the system tries to search a taxi in the adjacent zones.

ALT1: In particular, since most of the adjacent queues are not empty, MyTaxiService considers the longer one to forward Lidia's request and allocate a cab for her ride.

ALT2: Unfortunately, also all adjacent areas have empty queues at the moment, and consequently Lidia's request must be rejected; so, in the end, the negative outcome is notified to Lidia through the app.

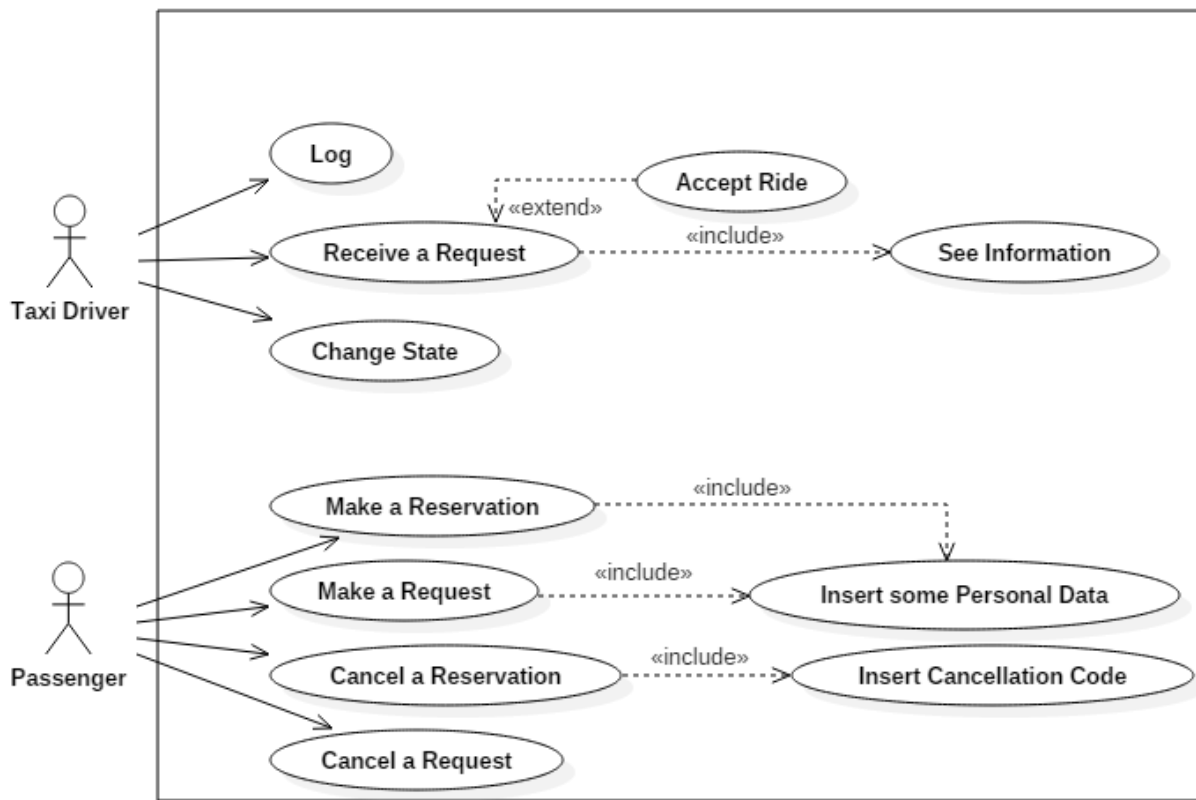
Scenario 8

Giorgio has to go to the train station with all his family in 3 hours, so he wants to make a request for a cab. He opens MyTaxiService web application through his web browser and fill in the reservation form. Actually, he inserts his personal data and the required information about the ride to book; furthermore he ticks the appropriate box to specify that he needs two taxis instead of a single one, because his family is composed by 7 people. However, his booking process concludes successfully. Later on, ten minutes before the time scheduled in the reservation, the system tries to allocate two taxis for Giorgio's ride; hence the system forwards two calls sequentially to the same local queue, with the only constraint that both must be allocated to confirm the ride.

ALT1: The system succeeds in finding two available taxis and so it sends a confirmation email with the code of the incoming taxis to Giorgio, specifying a delay of 5 minutes with respect to the previously scheduled time, due to traffic.

ALT2: When the system look for two available taxis for Giorgio, it concludes successfully the first allocation but then it finds out that both the local queue in question and all the adjacent ones are empty. Hence, since it cannot satisfy Giorgio's reservation, sends him an email to notify the failure and the non-provision of the expected service. Furthermore, the system informs the taxi driver who has already accepted the call that his ride has been deleted.

3.5 USE CASES



1.

Name:	The passenger makes a reservation for a ride
Information:	This use case does not analyse the case where the passenger cancels the ride after that he receives the information about the ride, because this procedure is equal to the cancellation of a request.
Path:	<ol style="list-style-type: none">1. The user opens the reservation section.2. The user inserts his data (name, surname and e-mail).3. The passenger inserts a starting and an ending points, an hour for the ride.4. The passenger receives a confirmation and some information of the cab.5. The passenger receives the information about the ride 10 minutes before it.
Alternative path:	<ol style="list-style-type: none">5. The passenger opens the cancellation area.6. The passenger inserts the code to cancel the reservation.
Exceptions:	We have to consider exceptions for the network part and for possible errors with the insertion of the cancellation code.

2.

Name:	The passenger makes a request for a ride
Information:	We consider the case where we find a cab for the ride.
Path:	<ol style="list-style-type: none">1. The user opens the request section.2. The user inserts his data (name, surname and an e-mail).3. The passenger inserts a starting point for the ride.4. The passenger receives a confirmation and some information about the incoming cab.
Alternative path:	<ol style="list-style-type: none">5. The passenger decides to cancel the request from the option in the confirmation.
Exceptions:	We have to consider exception for the network part.

3.

Name:	The taxi driver logs to the system.
Information:	The taxi driver is register.
Path:	<ol style="list-style-type: none">1. The taxi driver opens the login area.2. The taxi driver logs himself with his right credential.
Alternative path:	
Exceptions:	We have to consider exception for the network part and for wrong credential in the login phase.

4.

Name:	The taxi driver receives and manages a request
Information:	The taxi driver is the first one of the queue and for this reason he receives a request; he usually can answer or ignore the call, but sometimes the system forwards a mandatory assignment and then the driver is forced to accept the request. Anyway, normally the driver has to answer the incoming call within 30 seconds, because the system has a deadline for this kind of actions.
Path:	<ol style="list-style-type: none">1. The taxi driver receives a request.2. The taxi driver accepts the request.3. The taxi driver opens the requests area.4. The system changes the state of the taxi driver.5. The taxi driver opens his private area.6. The taxi driver change again his state at the end of the ride.
Alternative path:	<ol style="list-style-type: none">3. The taxi driver ignores the request.4. The system adds a penalty to the taxi driver.5. The system moves the taxi driver at the bottom of the queue.
Exceptions:	We have to consider exceptions as regards the network part.

5.

Name: The taxi driver changes his state

Information: The taxi driver sometimes can change his state to not available but in case of some glitches, he must use the command Urgent problem to contact immediately his manager.

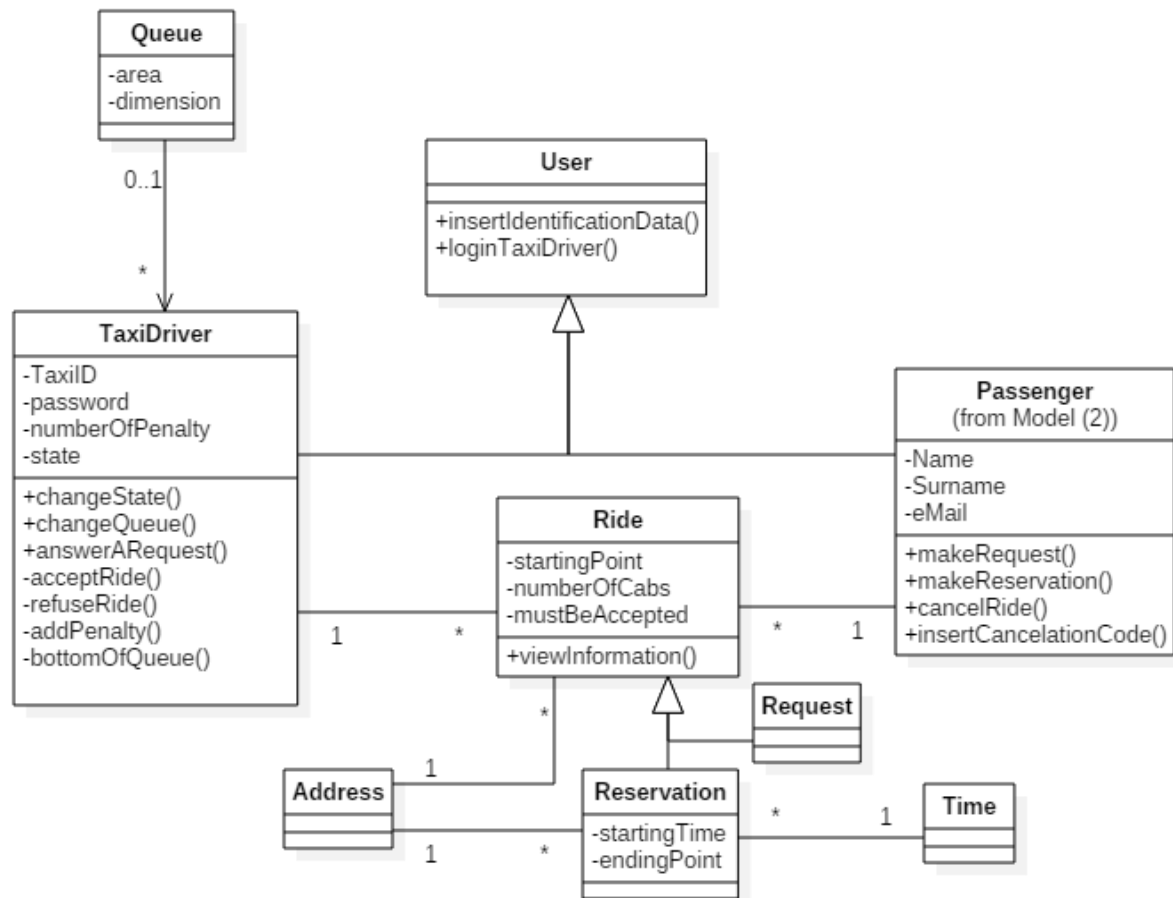
Path:

1. The taxi driver opens his account page.
2. The taxi driver change his state to not available.
3. The taxi driver uses the command Urgent Call
4. The taxi driver notifies the manager about the problem.
5. The system set the taxi as unusable
6. At the end, the taxi driver changes again his state if the problem is solved.

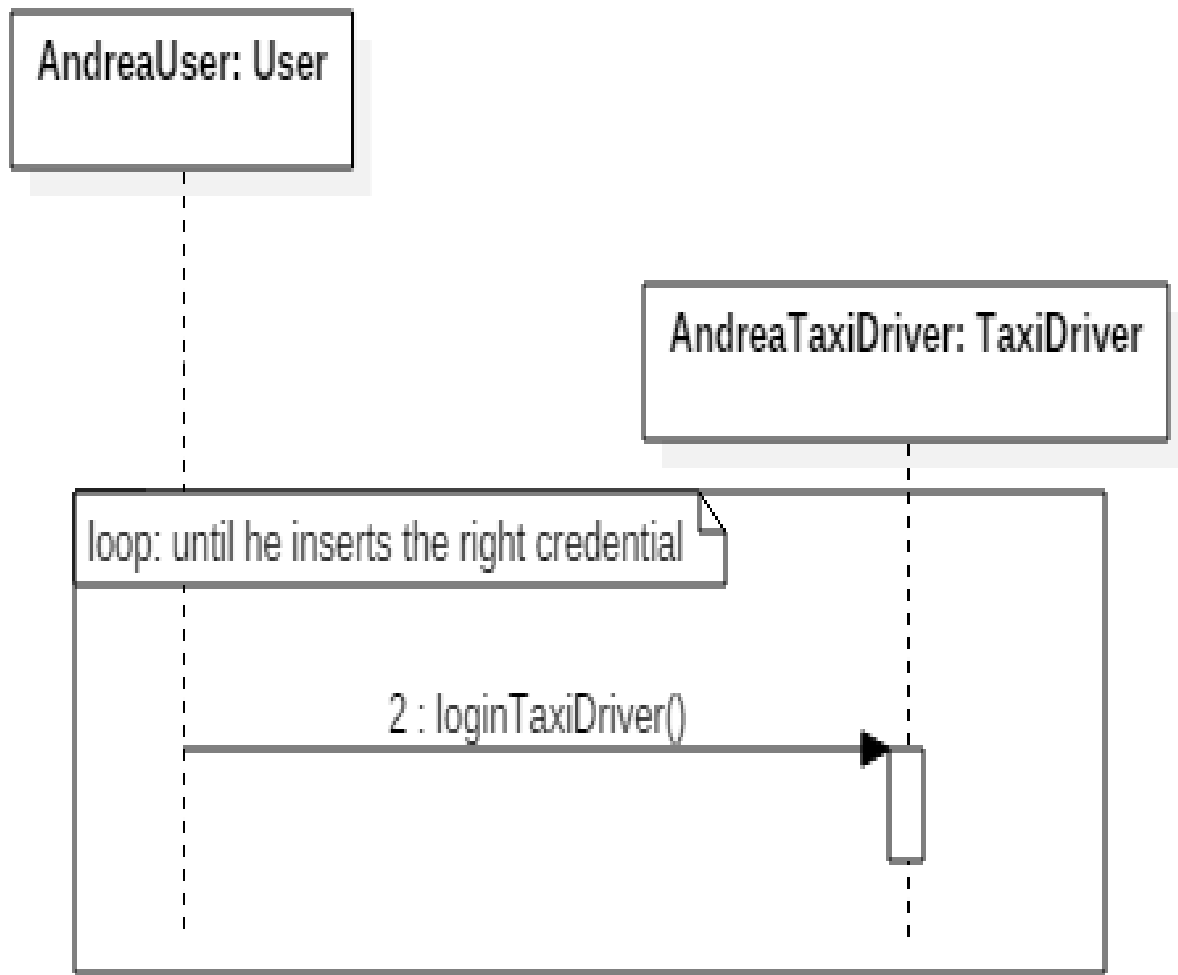
Alternative
path:

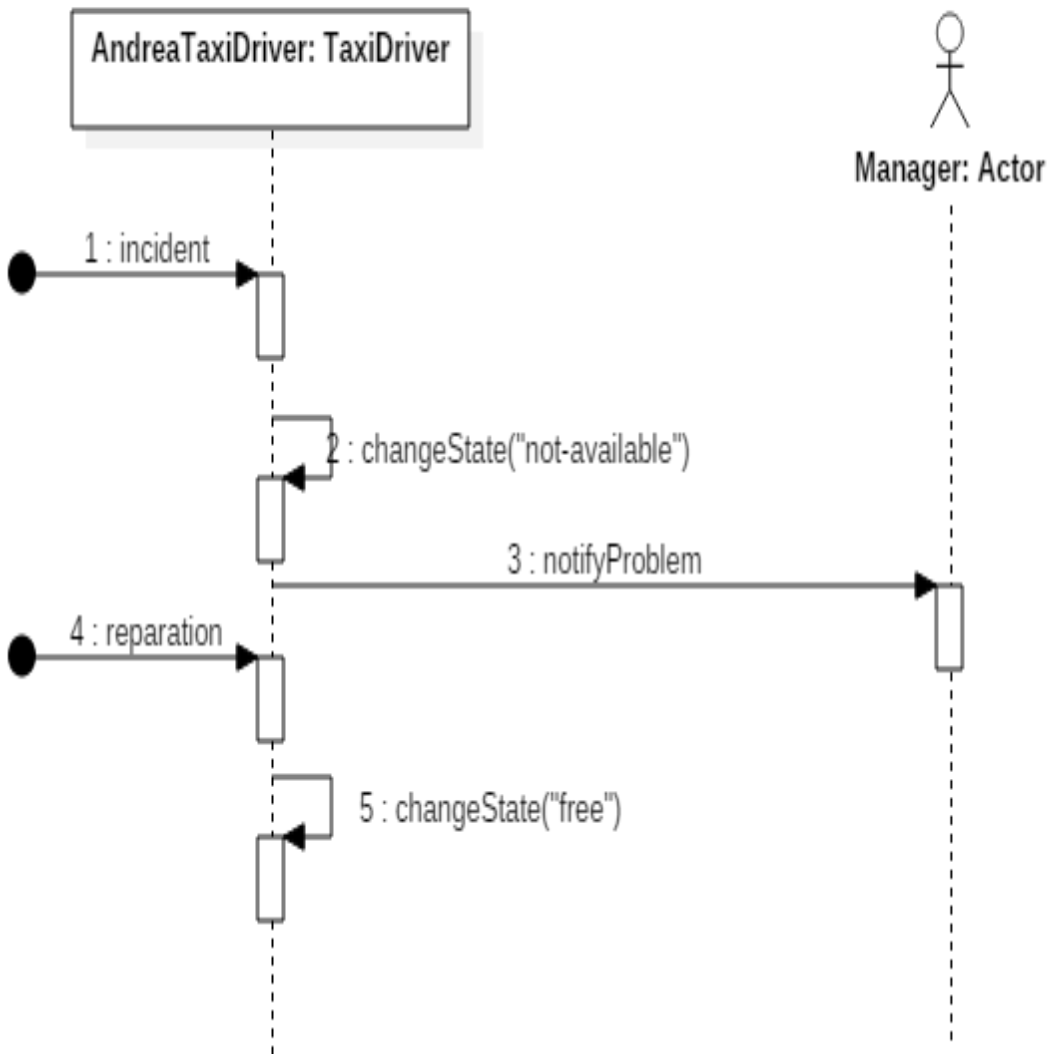
Exceptions: We have to consider exception for the network part.

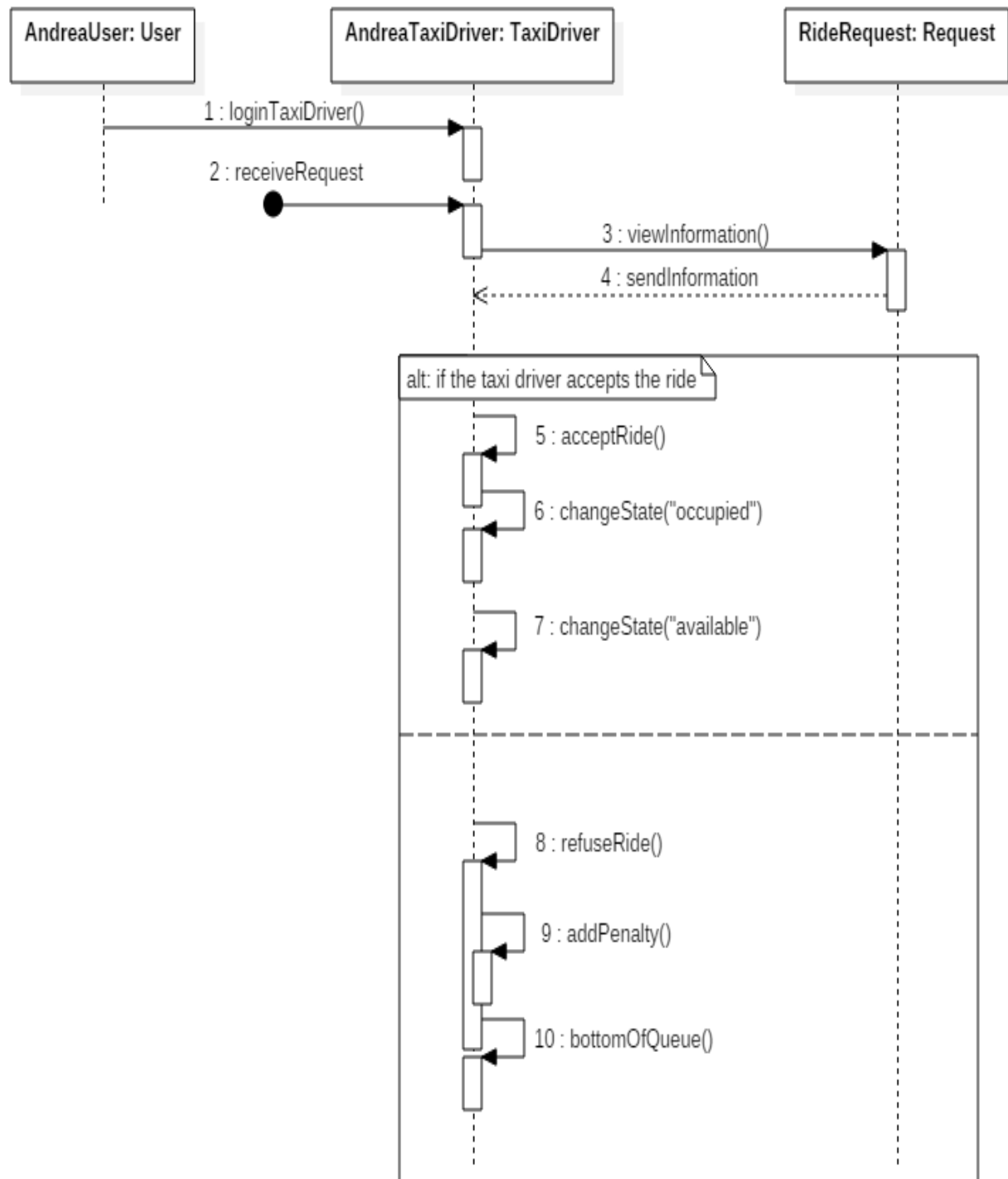
3.6 CLASS DIAGRAM

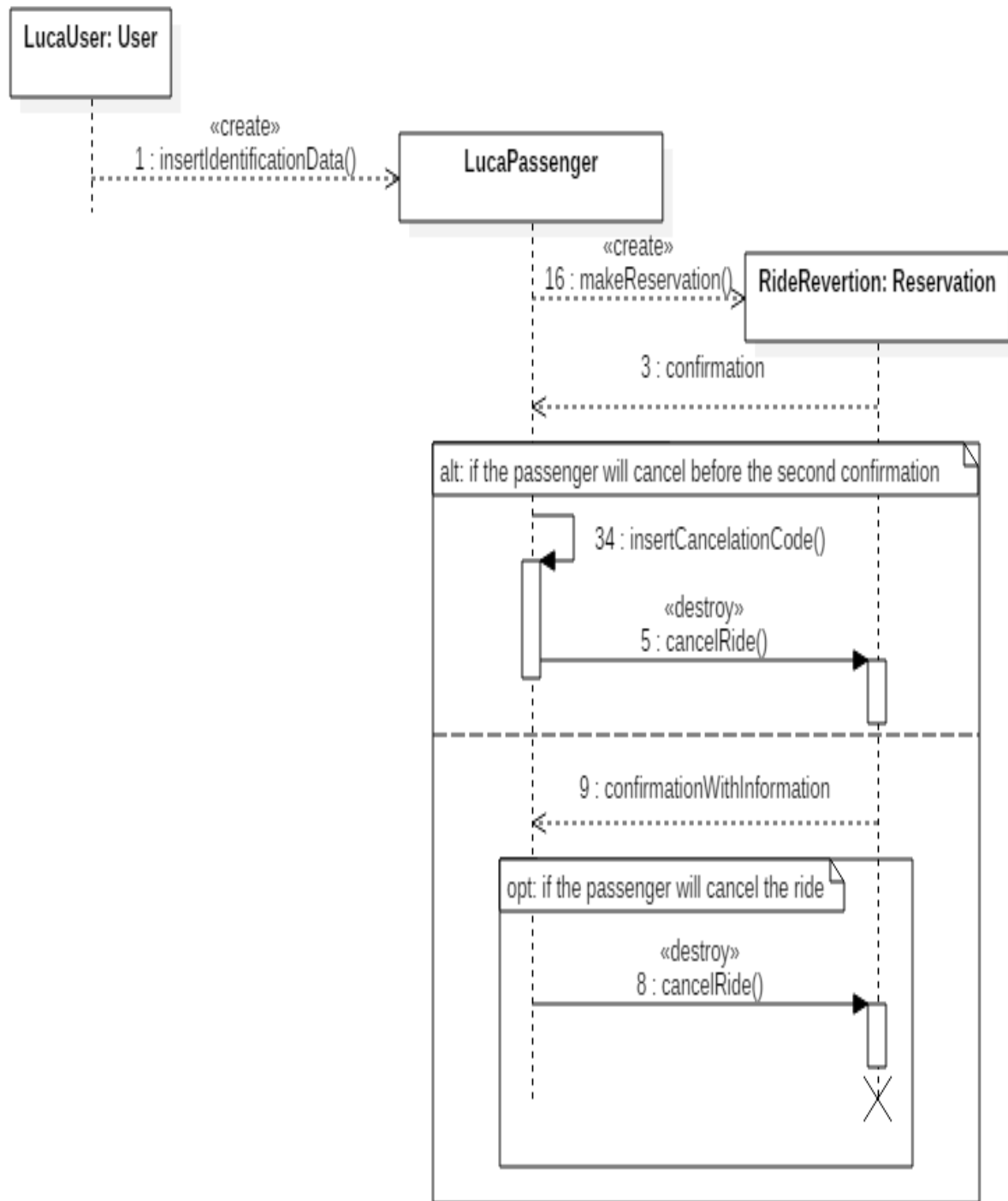


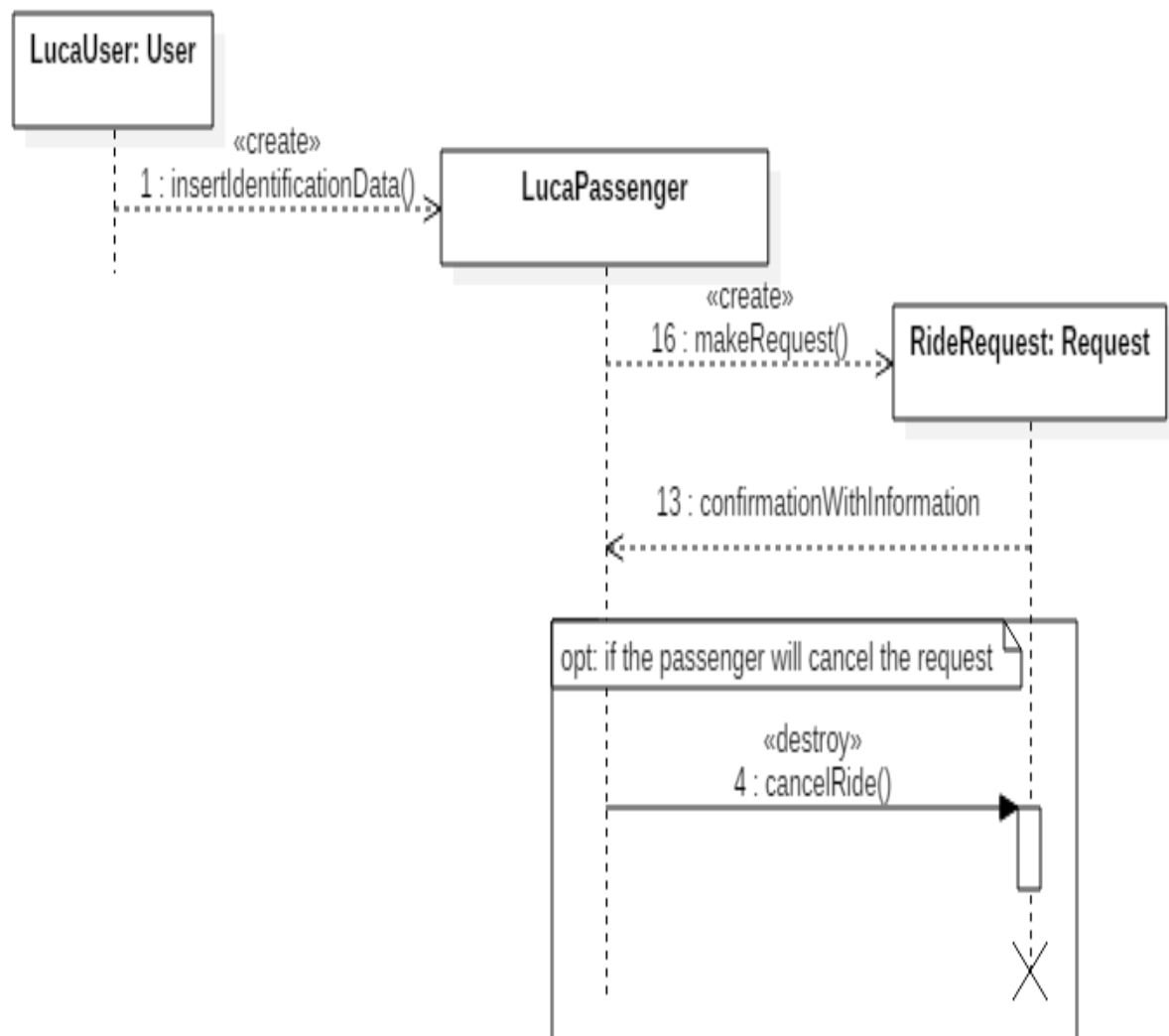
3.7 SEQUENCE DIAGRAMS



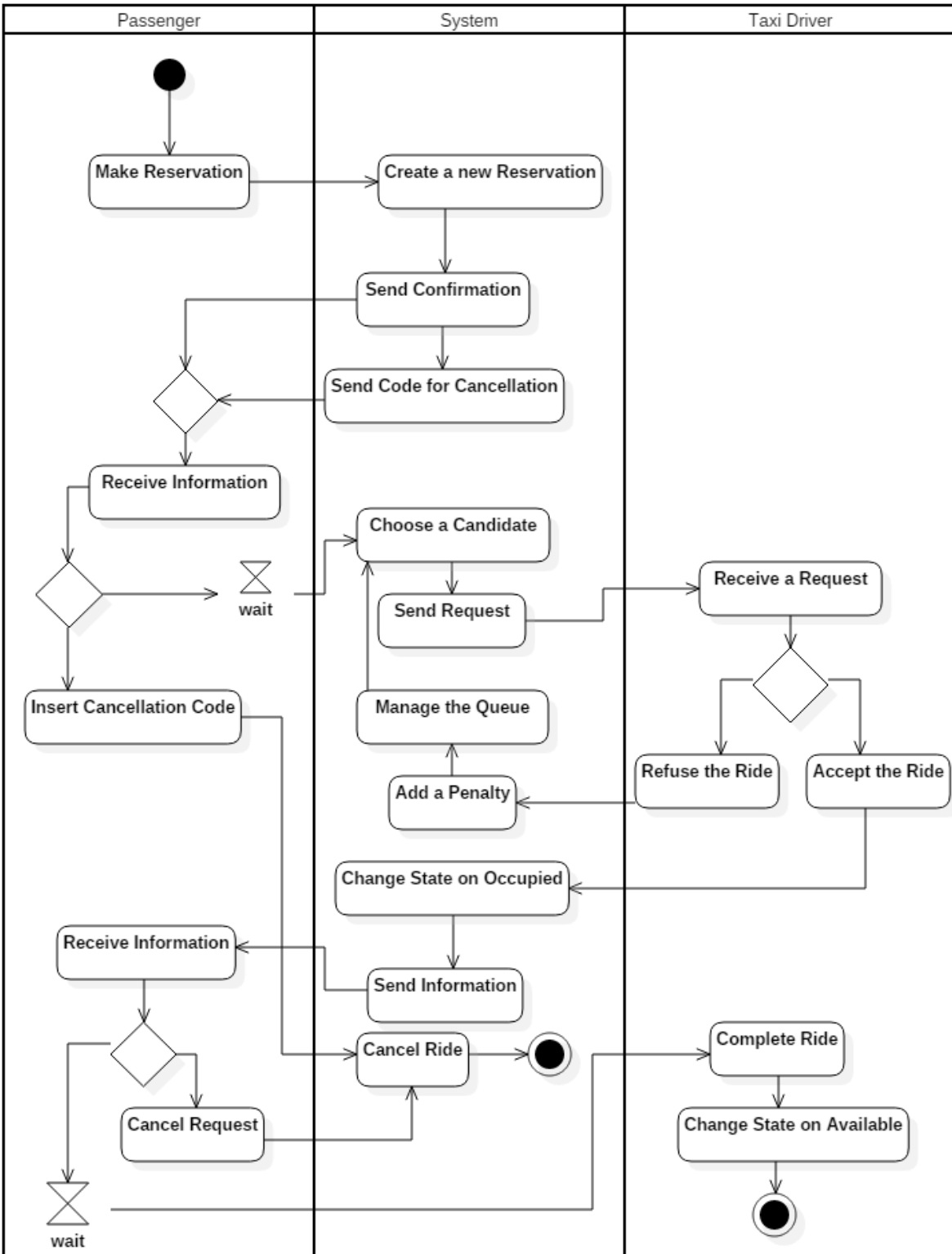


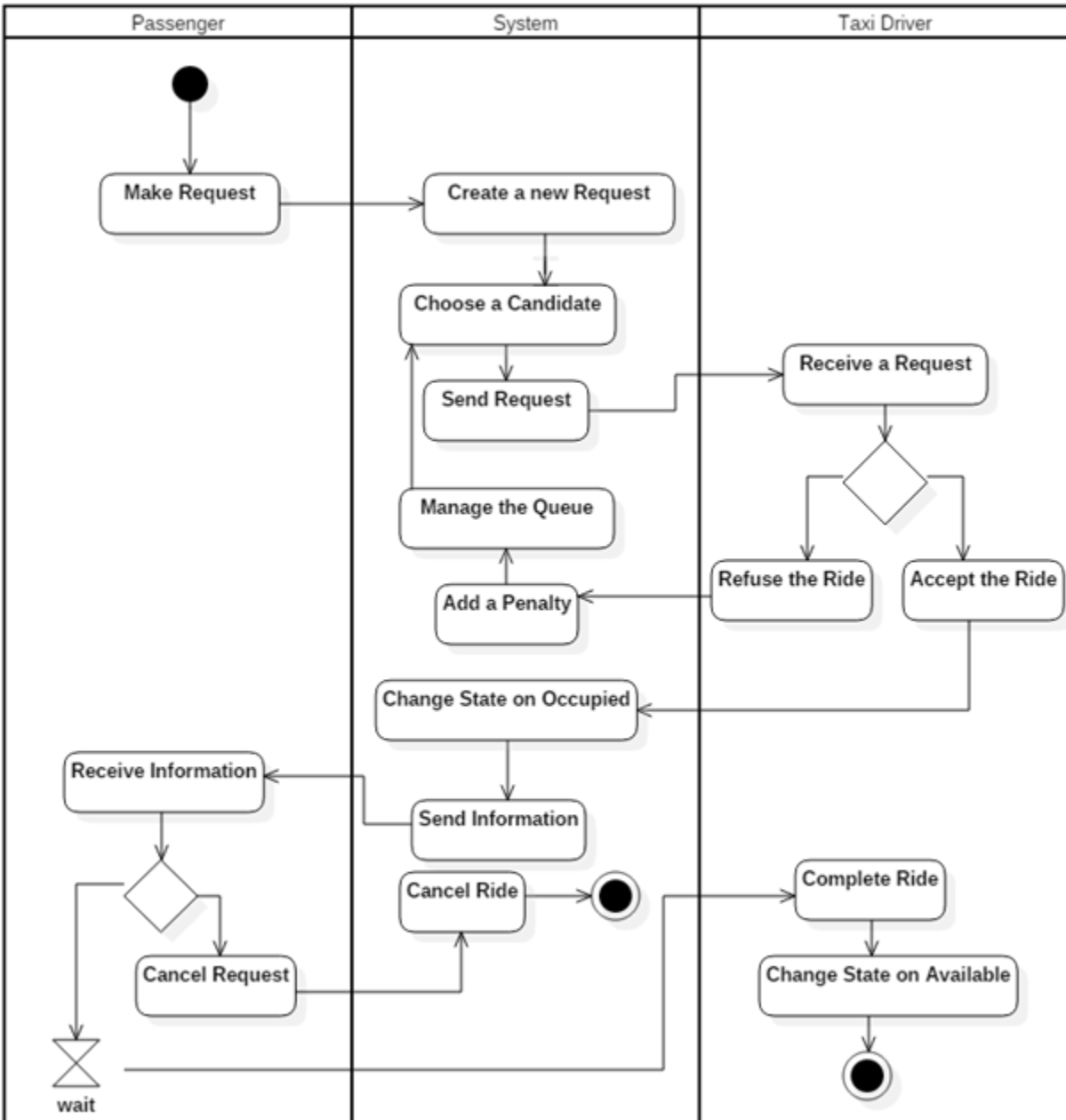






3.8 ACTIVITY DIAGRAMS





3.9 ALLOY MODELLING

```
//SIGNATURES

abstract sig User{}

sig Time{}

sig Addr{
  zone: one Area
}

abstract sig Ride{
  origin: one Addr,
  applicant: one Passenger,
  driver: lone TaxiDriver
}

sig Request extends Ride{}

sig Reservation extends Ride{
  destination: one Addr,
  time: one Time
}

sig Area{}

sig Queue{
  drivers: set TaxiDriver,
  zone: one Area,
  adj: set Area,
  length: one Int
}{#adj<=8 and length=#drivers}

enum State {avail,notAv,occ,unus,offDuty}

sig TaxiDriver extends User{
  ID: one TaxiID,
  state: one State,
  currentZone: lone Area,
  currentRide: lone Ride
}{#currentRide=1 iff state=occ}

sig TaxiID{}

sig PersonalData{}

sig Passenger extends User{
  PD: one PersonalData,
  rides: set Ride
}
```

//FACTS

```
fact RideWithDriver{
all r:Ride | #r.driver=1 iff some d:TaxiDriver | d.currentRide=r
}
```

```
fact AllAvailableInAQueue{
all d: TaxiDriver | d.state=avail implies some q: Queue | d in q.drivers and d.currentZone= q.zone
}
```

```
fact noTwoQueuesforTheSameArea{
no disj q1,q2 :Queue | q1.zone=q2.zone
}
```

```
fact noTwoDuplicatedRequests{
no disj r1, r2 :Request | r1.applicant=r2.applicant and r1.origin=r2.origin and r1.driver!=r2.driver
}
```

```
fact noTwoIncompatibleReservations{
no disj res1,res2:Reservation | res1.applicant=res2.applicant and res1.time=res2.time and res1.origin!=res2.origin
}
```

```
fact noTwoDuplicatedReservations{
no disj res1,res2:Reservation | res1.applicant=res2.applicant and res1.time=res2.time and res1.origin=res2.origin and res1.driver!=res2.driver
}
```

```
fact AvailableDriversNotHaveARide{
all dr: TaxiDriver | dr.state=avail implies #dr.currentRide=0
}
```

```
fact NoTwoRequestsWSameDriver{
all disj r1,r2:Request , dr: TaxiDriver | dr.currentRide=r1 implies r2.driver!=dr
}
```

```
fact NotAllocatedWOAvailability{
all r :Ride | one q:Queue | r.origin.zone= q.zone and q.length=0 and all q1: Queue | q1.zone in q.adj and q1.length=0 implies #r.driver=0
}
```

```

// ASSERTIONS and PREDICATES

assert NoOccupiedWOaRide{
no d:TaxiDriver | d.state=occ and #d.currentRide=0
}
check NoOccupiedWOaRide

assert NoAvailableDriverWOQueue{
all d:TaxiDriver | d.state=avail implies some q:Queue |d in q.drivers
}
check NoAvailableDriverWOQueue

pred show(){ }run show for 10
run show for 30

pred loginTaxiDriver(d, d1:TaxiDriver){
//precondition
d.state=offDuty and
//postcondition
d1.ID=d.ID and
d1.state=avail
} run loginTaxiDriver

pred changeState(d, d1:TaxiDriver, s: State){
//precondition
some s_old:State | s_old!=offDuty and d.state=s_old and
//postcondition
d1.ID=d.ID and
d1.state=s
} run changeState

pred acceptARide(d,d1:TaxiDriver, r:Ride){
//precondition
d.state=avail and
//postcondition
d1.ID=d.ID and
d1.state=occ and d1.currentRide=
} run acceptARide

```

Executing "Check NoOccupiedWOaRide"
 Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
 2542 vars. 222 primary vars. 4518 clauses. 312ms.
 No counterexample found. Assertion may be valid. 16ms.

Executing "Check NoAvailableDriverWOQueue"
 Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
 2539 vars. 222 primary vars. 4484 clauses. 141ms.
 No counterexample found. Assertion may be valid. 16ms.

Executing "Run show for 10"
 Solver=sat4j Bitwidth=4 MaxSeq=7 SkolemDepth=1 Symmetry=20
 32736 vars. 1710 primary vars. 77377 clauses. 875ms.
 Instance found. Predicate is consistent. 250ms.

Executing "Run show for 30"
 Solver=sat4j Bitwidth=4 MaxSeq=7 SkolemDepth=1 Symmetry=20
 548116 vars. 13530 primary vars. 1443587 clauses. 9384ms.
 Instance found. Predicate is consistent. 14968ms.

Executing "Run changeState"
 Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
 2653 vars. 235 primary vars. 4764 clauses. 281ms.
 Instance found. Predicate is consistent. 94ms.

Executing "Run loginTaxiDriver"
 Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
 2542 vars. 225 primary vars. 4486 clauses. 125ms.
 Instance found. Predicate is consistent. 62ms.

i

