FER POLITECNICO MILANO MÄLARDALENS HÖGSKOLA

Taxi Service
Requirements Definition

Version 2.0

Revision History

Date	Version	Description	Author
2012-11-01	0.1	Initial Draft	DSD
2012-11-02	0.2	Chapters 1, 2 Draft	Marko Coha
2012-11-02	0.3	Chapter 3 Draft	Luca Zangari
2012-11-02	0.4	Chapters 5,6 Draft	Igor Piljić
2012-11-02	0.5	Chapter 4 Draft	Leon Dragić
2012-11-02	1.0	First version finalized	Igor Piljić
2012-11-04	1.1	Add detailed requirements for use case TAXI2 based on the first iteration planning meeting	Lyudmil Angelov
2012-11-08	1.2	Updated requirements for use case TAXI2	Jelena Jerat
2012-11-15	1.3	Updated requirements for the second iteration: use cases TAXI1, TAXI2, TAXI5	Lyudmil Angelov
2012-12-05	1.4	Update requirements for use case TAXI1, TAXI6	Luca Zangari
2012-01-20	2.0	Final document review	Jelena Jerat

Table of Contents

1. Introduction

1.1 Purpose of this document

This document contains all the requirements for the Taxi service project. It is defined after collecting all the necessary requirements in order to start the implementation process. These requirements may change, or new ones may be added, because of the iterative development process of this project. Any changes should be documented in one of the revisions of the document. All the prototypes and the final product should be based on this document.

1.2 Intended Audience

The requirements document should be used by all team members, the supervisor and the customers. Team members should use the document when implementing any of the features of the system, and all parts of the system should work as is described in this document. Requirements can be changed and new ones can be added, if requested by the supervisor or the customers, and this document should keep track of all the changes.

1.3 Scope

This document will describe all the requirements for the Taxi service project, the characteristics of future users and constraints that can influence implementation of the project. Interaction of the system with users will be described using use cases. Technical and implementation details will not be covered in this document.

1.4 Definitions and acronyms

1.4.1 Definitions

Keyword	Definitions
UC	*Use case*
ETA	*Estimated time of arrival*

1.4.2 Acronyms and abbreviations

Acronym or abbreviation	Definitions
GPS	*Global positioning system*

1.5 References

Taxi service website: http://www.fer.unizg.hr/rasip/dsd/projects/taxi_service

2. Overall Description

2.1 Product Perspective

This product is a standalone system, and, as such, is not a component of a larger system.

2.2 Product Functions

The main purpose of this product is to improve the organization of the taxi service in Milan. Using this product, the customers should be able to order a taxi to their current location, or a location of choice, and receive a confirmation of the order. Taxi drivers should be able to receive an order with a location, and change the status of a taxi accordingly. Taxis should be tracked at all times and put into zones, depending on their current location.

2.3 User Characteristics

There will be several types of users. Customers, or end-users, will use an Android “Catch a Cab” application to order a taxi. This application should be designed in such a way that no educational level, experience and technical expertise will be required for use. Another type of users will be taxi drivers. They will use an Android “Dispatch” application to receive orders and act on them. They should be proficient in using the application, so some education will be necessary. However, no technical expertise should be required and the application should be as simple as possible. Finally, the product will also be used by dispatchers. They will have to have some technical expertise and experience to efficiently coordinate all the drivers.

2.4 Constraints

Some limitations that should be taken into consideration are related to security and reliability of the system. The system should be designed in such a way that it’s impossible for a third party to see or change data in any way. Such an intrusion could allow a third party to act as a taxi driver, and could cause significant loss for the taxi organization. Furthermore, the system should be available at all times, and resistant to database, network and other failures. Such an error could cause a loss of communication between taxi drives and dispatchers, and should be treated accordingly.

3. Requirements Description

3.1 Introduction

The purpose of this project is to provide a service that, using mobile applications, is able to manage the taxis in a metropolitan area and gives to the customers the possibility to book a taxi and obtain information about their bookings.

3.2 General requirements

The server side of the system must be constantly updated with Taxi position, in order to put every taxi in a different queue, belonging on different area in the city.
System must manage the calls and dispatch every call to the nearest queue with available taxi.
The client's applications should communicate with the server side using their internet connection.

3.3 Functional requirements

Mobile application for taxi user shall be able to:

- Communicate with the server side, sharing constantly current position and status
- Receive from the server side the booking requests, accept or decline it
- Receive information about the booked destination

Server shall be able to:

- Receive constantly the taxi positions and statuses
- Divide the city in different areas and make a queue for every area
- Maintain the queues up to date with the taxi positions and statuses
- Receive orders from customer clients, and dispatch them to the nearest queues

Mobile application for customer user shall be able to:

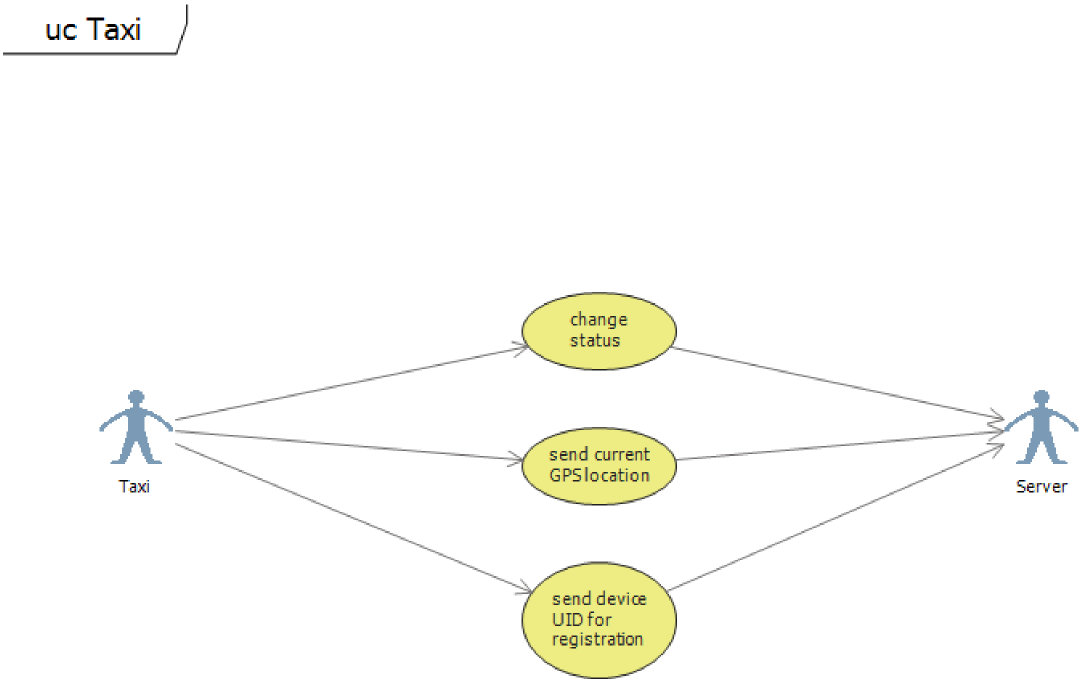
- Communicate with the server side, sending a booking request of a taxi
- Receive information about the booked taxi and its time of arrival

3.4 Non-functional requirements

- Mobile applications shall be developed for Android mobile platform
- Server application shall be developed using .NET technology
- Mobile application interface should be simple and user friendly
- The interfaces of communication should be simple and stable between the different components
- Server has to be able to work and communicate with large number of clients

4. Use Cases

4.1 Taxi UC



4.1.1 *Use case "change status"*

Use case ID	TAXI1
Name	*Change status*
Goal	*Change current taxi status*
Participating actors	Taxi and server
Precondition	
Main scenario	1. *The taxi driver has reached the end of his or her shift and would like to not be eligible to receive orders* 2. *Taxi driver clicks on the status button and sees their current status (see Figure 4.1.1.1)* 3. *The taxi driver switches their status from "On duty" to "Off duty" (see Figure 4.1.1.2)* 4. *The status change is communicated to the server*
Exceptions	*If there is no internet connection, taxi can't change its status*
Extensions	When an order is received the taxi status changes automatically from "available" to "busy", to come back to available, taxi driver needs to push the light on the button.
Dependent UC	

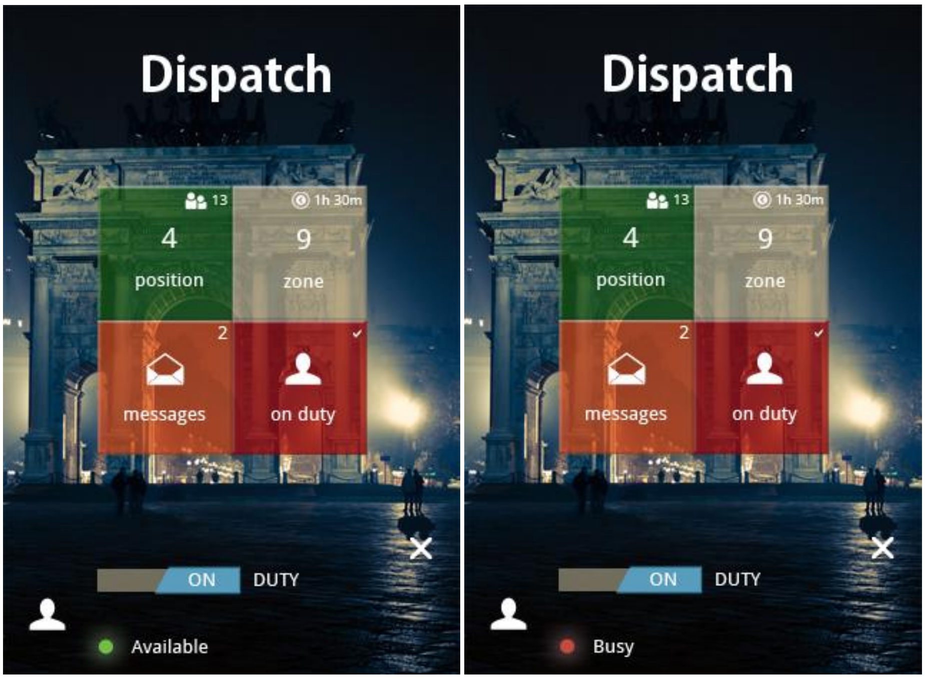


Figure 4.1.1.1: The taxi client user interface when they are on duty. When they are available to receive an order it looks like the screenshot on the left. If they are currently with a customer, they see the screenshot on the right.

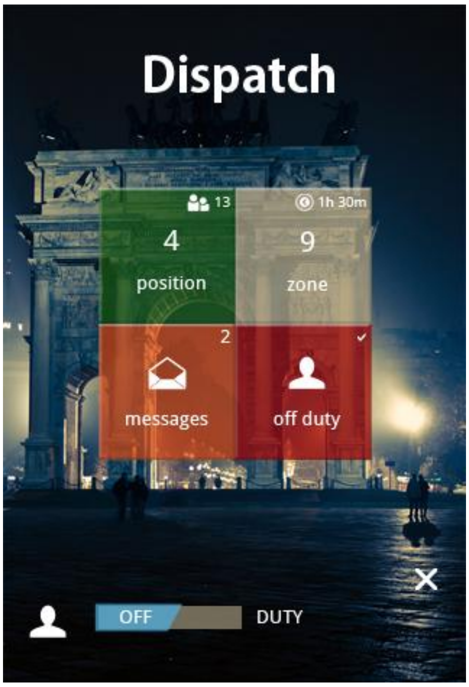


Figure 4.1.1.2: The taxi client user interface when they are off duty.

4.1.2 Use case "send current GPS location"

Use case ID	TAXI2
Name	*Send current GPS location*
Goal	*Update the location of the taxi*

Participating actors	Taxi and server
Precondition	The taxi needs to be “on duty”, The taxi device needs to have an internet connection, The server needs to be up, The taxi is registered in the system, The taxi device can get a GPS fix
Main scenario	1. *The taxi client application starts and shows the main screen* 2. *The application gets a fix of the GPS coordinates of the taxi at a regular time interval* *3. Each time a new fix is obtained, the application sends the latitude and longitude to the server through an HTTP POST request* *a. The URI of the request is “api/Taxis/id/Location”* *b. There are two parameters passed in the request* *i. “Latitude”, containing the latitude of the GPS fix* *ii. “Longitude”, containing the longitude of the GPS fix* 4. *The server updates its current location and responds with a status message “OK”* 5. *The taxi indicates that it has successfully connected to the server on the main screen (see Figure 4.1.2.1)*
Exceptions	*1. If the taxi is off duty, it does not report its current location* *2. If the taxi does not have an internet connection it indicates the problem on its main screen (see Figure 4.1.2.2)* *3. If the taxi cannot get a GPS fix it indicates the problem on its main screen (see Figure 4.1.2.2)* *4. If the server is down, the taxi indicates the problem on its main screen (see Figure 4.1.2.2)*
Extensions	
Dependent UC	TAXI3, TAXI1

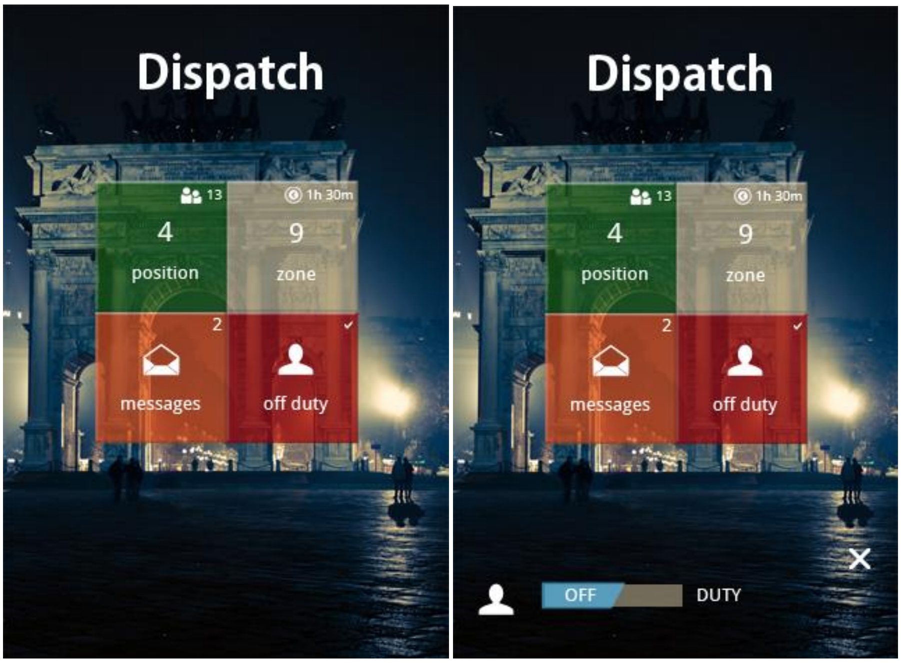
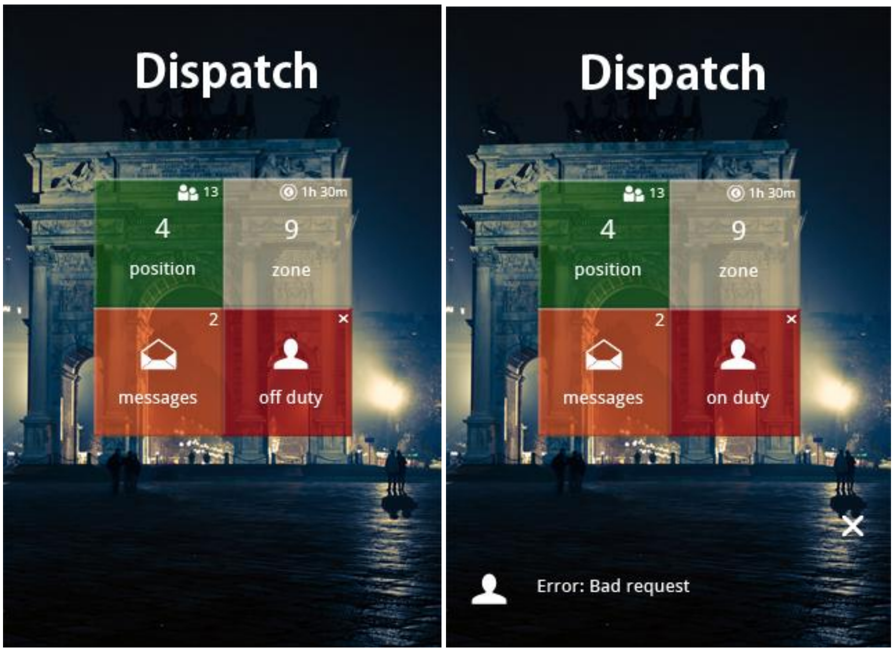


Figure 4.1.2.1: Taxi client user interface when coordinates are being successfully communicated to the server. On the left is the default view (notice the checkmark in the top-right corner of the status button). On the right is the view if the user clicks on the status button.



Error: Bad request

Figure 4.1.2.2: Taxi client user interface when coordinates are not being successfully communicated to the server. On the left is the default view (notice the “x” in the top-right corner of the status button). On the right is the view if the user clicks on the status button. The message displayed is different, depending on the error.

4.1.3 Use case “apply for taxi registration”

Use case ID	TAXI3
Name	*Apply for taxi registration*
Goal	*Add new taxi to the system*
Participating actors	Taxi and server
Precondition	
Main scenario	1. *Before starting to work as taxi, taxi needs to apply for taxi registration* 2. *System administrator adds a new taxi to the system*
Exceptions	*Taxi is already registered in the system* *Taxi registration is rejected*
Extensions	
Dependent UC	

4.1.4 Use case “accept or reject a customer order”

Use case ID	TAXI4
Name	*Accept or reject a customer order*
Goal	*Allow the taxi driver to refuse a customer order if they want to*
Participating actors	Taxi and server
Precondition	
Main scenario	1. *A customer orders a taxi* 2. *The server selects a taxi to service the request* 3. *The server sends the taxi the customer information* 4. *The taxi receives the order* 5. *The taxi driver chooses to accept or reject the order* 6. *The taxi driver’s response is sent to the server*
Exceptions	
Extensions	
Dependent UC	

4.1.5 Use case “get zone information”

Use case ID	TAXI5
Name	*Get zone information*
Goal	*Display to the taxi driver which zone they are in and how many people are waiting in the queue for the zone*
Participating actors	Taxi and server
Precondition	The taxi client is running and communicating its position to the

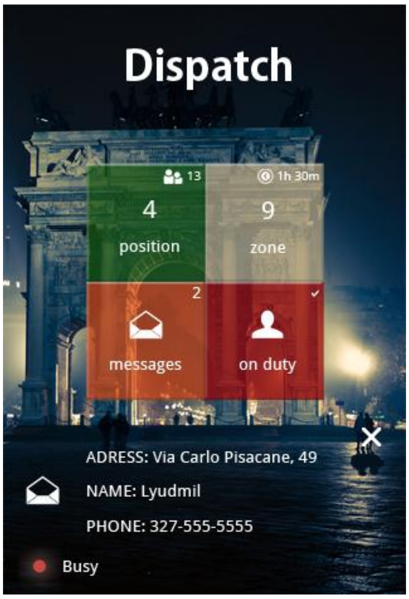
	server

Main scenario	1. *The taxi client sends its position to the server*
	2. *The server saves it, determines the zone, and responds to the request in the following format:*
	`{`
	`‘ZoneId’:<string>,`
	`‘ZoneName’:<string>,`
	`‘Position’:<integer>,`
	`‘status’:<string>`
	`}`
	Each value between angle brackets is calculated on the server. What we specify within the angle brackets is the type of data sent.
	3. *The taxi client parses the information and displays it in the zone button*

Exceptions	
Extensions	
Dependent UC	

4.1.6 Use case “receive orders”

Use case ID	TAXI6
Name	*Receive orders*
Goal	*Receive orders from customer clients*
Participating actors	Taxi , server, customer
Precondition	Customer client and taxi are in the same metropolitan area.
Main scenario	1. *The customer client sends an order from dispatch application.* 2. *The server gets the order and send it to a first taxi available in the nearest queue of the customer’s zone.* 3. *The taxi client show a notification that an order is been received* 4. *When the taxi driver push on messages button, he can see the order notification (see Figure 4.1.2.3)* 5. *The taxi client switches its status from “Available” to “Busy” (see Figure 4.1.2.3)* 6. *The status change is communicated to the server*
Exceptions	*If there is no internet connection, taxi can’t receive orders*
Extensions	
Dependent UC	TAXI2 , TAXI5



ADRESS: Via Carlo Pisacane, 49

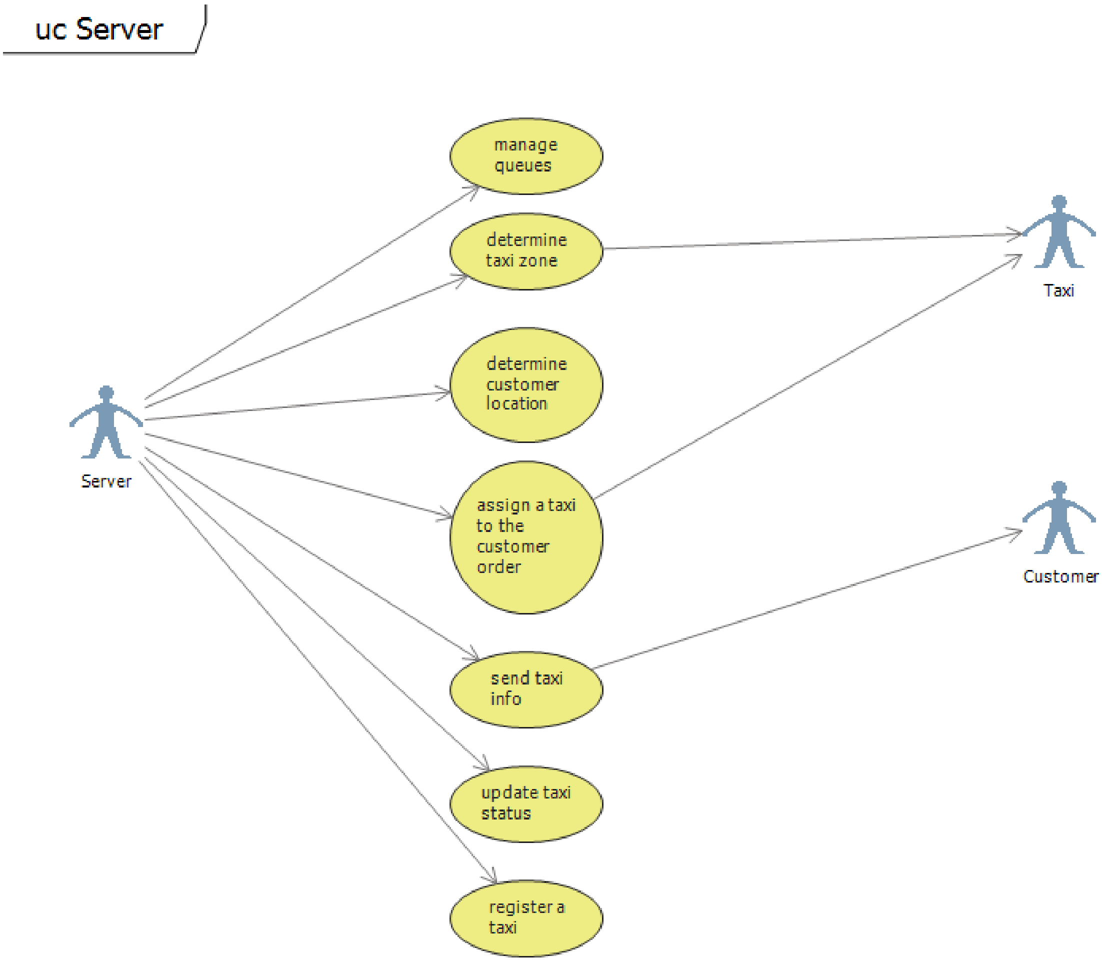
NAME: Lyudmil

PHONE: 327-555-5555

Busy

Figure 4.1.2.3: The taxi client user interface when an order is received. If the user clicks on the messages button he can see order’s detail. The taxi status changes from available to busy.

4.2 Server UC



4.2.1 Use case "manage queues"

Use case ID	SERVER1
Name	*Manage queues*
Goal	*Keep the queues for every city zone updated*
Participating actors	Server
Precondition	Change in the queue
Main scenario	1. *When the taxi enters a new city zone, it is automatically put at the end of the virtual queue of that zone* 2. *Whenever a taxi from the queue goes to "busy" or "off duty" status, it is removed from the queue* 3. *The server notifies the taxi about its position in the queue*
Exceptions	
Extensions	
Dependent UC	

4.2.2 Use case “determine taxi zone”

Use case ID	SERVER2
Name	*Determine taxi zone*
Goal	*Assigns taxi to the appropriate zone*
Participating actors	Server and taxi
Precondition	Taxi has sent its GPS location
Main scenario	1. *Server receives the GPS location of the taxi* 2. *Server determines the zone in which taxi currently is* 3. *Server sends the zone number to the taxi*
Exceptions	
Extensions	
Dependent UC	TAXI2

4.2.3 Use case “determine customer location”

Use case ID	SERVER3
Name	*Determine customer location*
Goal	*Determine the city zone of the customer*
Participating actors	Server
Precondition	Customer has made an order for the taxi
Main scenario	1. *Server receives the customer order* 2. *Server determines the zone of the customer*
Exceptions	
Extensions	
Dependent UC	CUSTOMER1

*4.2.4 Use case “*assign a taxi to the customer order*”*

Use case ID	SERVER4
Name	*Assign a taxi to the customer order*
Goal	*Dispatch the first taxi in the queue to the customer*
Participating actors	Server and taxi
Precondition	The server has determined the customer zone
Main scenario	1. *Server sends the customer order to the first taxi of the appropriate queue*
Exceptions	*Taxi refuses to take the order* *There is no taxis in the queue*
Extensions	

Dependent UC	SERVER3

4.2.5 *Use case* “send taxi info”

Use case ID	SERVER5
Name	*Send taxi info*
Goal	*Inform the customer about the taxi which will pick him up*
Participating actors	Server and customer
Precondition	The order has been assigned to a taxi
Main scenario	1. *Server sends the information about taxi and ETA and location to the customer*
Exceptions	
Extensions	
Dependent UC	SERVER4, CUSTOMER3

4.2.6 *Use case* “update taxi status”

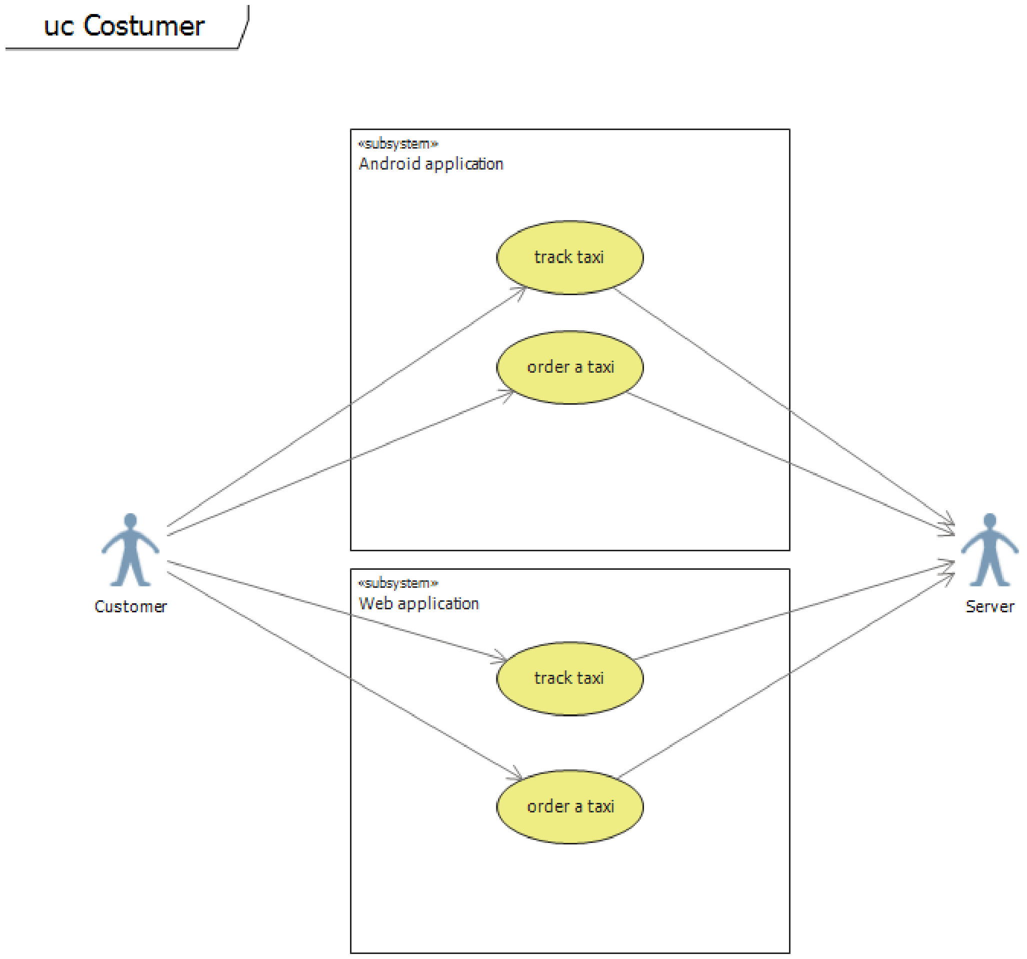
Use case ID	SERVER6
Name	*Update taxi status*
Goal	*Update the status of the taxi on the server*
Participating actors	Server
Precondition	Taxi changed its status
Main scenario	1. *Server receives the status change from taxi* 2. *Server updates the queue*
Exceptions	
Extensions	
Dependent UC	TAXI1

4.2.7 *Use case* “register a taxi”

Use case ID	SERVER7
Name	*Register a taxi*
Goal	*Add new taxi to the system*
Participating actors	Server
Precondition	Taxi had sent the application for registration
Main scenario	1. *Server receives the application* 2. *Server adds taxi to the system*
Exceptions	*The taxi is already in the system* *The application is rejected*

Extensions	
Dependent UC	TAXI3

4.3 Customer UC



4.3.1 Use case "order a taxi"

Use case ID	CUSTOMER1
Name	*Order a taxi*
Goal	*Order a taxi to customer location*
Participating actors	Server and customer
Precondition	
Main scenario	1. *Customer clicks the button for ordering a taxi* 2. *Customer chooses number of passengers and other order details* 3. *Application sends the GPS coordinates and order details*

	to the server
Exceptions	
Extensions	
Dependent UC	

4.3.2 Use case “order a taxi”

Use case ID	CUSTOMER2
Name	*Order a taxi*
Goal	*Order a taxi to customer location*
Participating actors	Server and customer
Precondition	
Main scenario	1. *Customer uses web application to select his/hers position on the map* 2. *Customer chooses number of passengers and other order details* 3. *The position and order details are sent to the server*
Exceptions	
Extensions	
Dependent UC	

4.3.3 Use case “track a taxi”

Use case ID	CUSTOMER3
Name	*Track an order*
Goal	*Track an existing order status.*
Participating actors	Server and customer
Precondition	Customer has sent an order to the server
Main scenario	1. *Customer selects the current order in the Customer application* 2. *Server sends the customer order details such as position of the taxi, ETA and other related information*
Exceptions	
Extensions	
Dependent UC	

4.3.4 Use case “track a taxi”

Use case ID	CUSTOMER4
Name	*Track an order*
Goal	*Track an existing order status.*
Participating actors	Server and customer
Precondition	Customer has sent an order to the server Customer is logged in on the web application
Main scenario	3. *Customer selects the current order* 4. *Server sends the customer order details such as position of the taxi, ETA and other related information*
Exceptions	
Extensions	
Dependent UC	

5. Requirements Definition

5.1 Requirement Group Definitions

Identification	Requirement Group	Rem.
CA	Customer application	
WA	Web application for customers	
TA	Taxi application	
SER	Server	
NFR	Non-functional requirements	

5.2 Requirement Sources

Source	Description	Rem.
CTM	Customer	
SYS	System	
DEV	Developer	

5.3 Requirement definitions

Identity	Status	Priority	Description	Source
			Android application for customers	
CA-1	I	1	User can send request for a taxi	CTM
CA-2	I	1	User can receive information about taxi that is assigned to pick him up and the estimated time of arrival (order could be made by both Android or web application)	CTM
CA-3	A	3	User can change the phone number from which the order is made	CTM
CA-4	D	3	User can place a postponed order	CTM
			Web application for customers	
WA-1	A	3	Customer can register and use web application	CTM
WA-2	A	3	User can send request for a taxi	CTM
WA-3	A	3	User can track the order made by the web application or Android application	CTM
			Android application for taxis	
TA-1	I	1	User sends information about his position periodically	SYS
TA-2	I	1	User can receive request to pick up customer	CTM
TA-3	I	1	User can receive information about zone he is currently in	CTM
TA-4	I	1	User can change its status	SYS
TA-5	H	2	User can accept or decline call	CTM
TA-6	I	2	User can see if connection with server is established	DEV
TA-7	H	2	User can register taxi	SYS
			Server	
SER-1	I	1	Server can determine client location	SYS
SER-2	I	1	Server can determine zone based on position	SYS
SER-3	I	1	Server can notify customer	CTM
SER-4	I	1	Server can manage queues	SYS
SER-5	I	1	Server can assign taxi to customer	CTM
SER-6	I	1	Server can update taxi status	SYS
SER-7	I	2	Server can recognize taxis by its id	SYS
SER-7	H	3	Server can calculate which taxi is nearest to certain position	DEV

			Non-functional requirements	
NFR-1	I	1	System has to be reliable always	CTM
NFR-2	I	1	Mobile applications are developed for Android OS	SYS
NFR-3	I	1	Mobile applications should be user friendly	DEV
NFR-4	I	1	System should be able to work with large number of customers	CTM

Requirement status:

I = initial (this requirement has been identified at the beginning of the project),
D = dropped (this requirement has been deleted from the requirement definitions),
H = on hold (decision to be implemented or dropped will be made later),
A = additional (this requirement was introduced during the project course).

Requirement priority:

1-high priority
2-medium priority
3-low priority

5.3.1 Change Log

Identity	Action	Date	Comments
CA-3	Added	15.11.2012.	Additional requirement added for customers who use tablets/phones without SIM card
WA-1	Added	28.12.2012.	Added web application to the requirements
WA-2	Added	28.12.2012.	Added web application to the requirements
WA-3	Added	28.12.2012.	Added web application to the requirements
CA-2	Modified	28.12.2012.	User should be able to track the orders made by both Android and Web application

Requirement status:

D = dropped (this requirement has been deleted from the requirement definitions),
H = on hold (decision to be implemented or dropped will be made later),
A = added (this requirement was introduced during the project course).
R = resurrected (dropped or on hold requirement was reactivated)

6. Future Development

Initial version of product will consist only of features needed for core functionality. Additional features were considered and discussed, but will be added after the main goals are accomplished. Some of these additional features are:

- Taxi sharing
- Advertisement system
- Customer registration
- Keep statistics about taxis and customers
- Show directions to taxi driver
- Connect applications with social networks

Final product could be developed for other operating systems besides Android, such as iOS or WP7.