Requirements Analysis and Specification Document: myTaxiService

Chitti Eleonora, De Nicolao Pietro, Delbono Alex

October 25, 2015

Contents

Chapter 1

Introduction

1.1 Purpose

This document is the Requirement Analysis and Specification Document for the myTaxiService application. Its aim is to completely describe the system, its components, functional and non-functional requirements, constraints, and relationships with the external world and to provide typical use cases and scenarios for all the users involved. Further, this document will provide formal specification of some features of the applications.

This document is written for project managers, developers, testers and Quality Assurance. It may be useful also to users. It may be used in a contractual requirement.

1.2 Scope

The system is a taxi reservation and dispatching system for large cities. Its aim is to simplify the access of passengers to the service and to guarantee a fair management of taxi queues.

The system consists in a back-end server application (*myTaxi Server*), a web application front-end (*myTaxi Web*) and in a mobile application (*myTaxi Mobile*).

The system has 2 types of users: passengers and taxi drivers; it should allow the users to sign up and login with their credentials. The system has to know the location of both the passengers and the taxi drivers.

The system allows any passenger to request a taxi, informing him o her of the incoming taxi code and the estimated waiting time.

The system knows about the available taxi drivers and, when a request is incoming, informs one of them about the location of the available passenger; the taxi driver can either accept or deny the ride. If the taxi driver accepts the ride, the system sends a confirmation to the passenger, together with the estimated waiting time. If the taxi driver rejects the ride, the system looks for another taxi driver in the same way.

The system offers programmatic interfaces (APIs) to enable the development of additional services on top of the basic one.

The system is provided with two optional modules:

Taxi reservation allows the passenger to reserve a taxi by specifying the origin and the destination of the ride.

Taxi sharing allows the passengers to share a ride together, dividing the costs.

1.3 Definitions, acronyms, and abbreviations

RASD: Requirements Analysis and Specification Document.

System: the whole software system to be developed, comprensive of all its parts and modules.

Module: an optional software component which uses the core system APIs to provide additional features.

Passenger: the registered user who uses the service for a taxi ride.

Taxi driver: any taxi driver subscribed to the service.

User: any use (passenger or taxi driver) subscribed to the service.

1.4 References

- Project rules: "AA 2015-2016 Software Engineering 2 - Project goal, schedule and rules"

- Assignment: "Software Engineering 2 Project, AA 2015/2016 - Assignments 1 and 2"
- IEEE Std 830-1998: "IEEE Recommended Practice for Software Requirements Specifications"

1.5 Overview

This document is structured in three parts:

Chapter 1: Introduction. It provides an overall description of the system scope and purpose, together with some information on this document.

Chapter 2: Overall description. Provides a broad perspective over the principal system features, constraints, and assumptions about the users and the environment.

Chapter 3: Specific requirements. Goes into detail about functional and nonfunctional requirements. This chapter is arranged by feature.

Chapter 2

Overall description

2.1 Product perspective

The back-end stores its data in a RDBMS and can run on every platform that supports the JVM. The web applications works on any web server that supports PHP. The mobile application is supported by Android, iOS. The system provides APIs to extend its functionalities, e.g. :

- taxi reservation
- ride sharing
- online payments
- ...

2.1.1 User interfaces

The user interfaces must provide the following logical characteristics both in the mobile app and in the web interface:

- The possibility to choose the language used in the contents in every moment during every operation.
- A first screen in which the user must login to begin operations.
- A dashboard with links to every function in order to show the user the capabilities of the system and allow him to save time.

- A link to the dashboard in every screen.
- A reminder in the top bar to show the last taxi service called, with a link to a screen which displays the reserved taxi history

2.1.2 Hardware interfaces

The system has to deal with the dichotomy of the web user interaction and the mobile one. It is necessary to provide a common look and feel, without losing simplicity with the different hardware interfaces. For instance, the compilation of data fields has to be made with multiple choices in order to simplify the user's experience of the app. Same goes for the dimension of buttons that can not be too small.

2.1.3 Software interfaces

The required software products used by the systems are:

- MySQL 5.7 `http://dev.mysql.com`
- Java SE 8 `http://java.com`

2.2 Product functions

The system allows passengers to book a taxi and notify taxi drivers of the request.

In particular it lets users to:

- Passengers:
  - create an account
  - request a taxi
  - share a taxi with other passengers
- Taxi drivers:
  - create an account
  - accept or deny a lift request

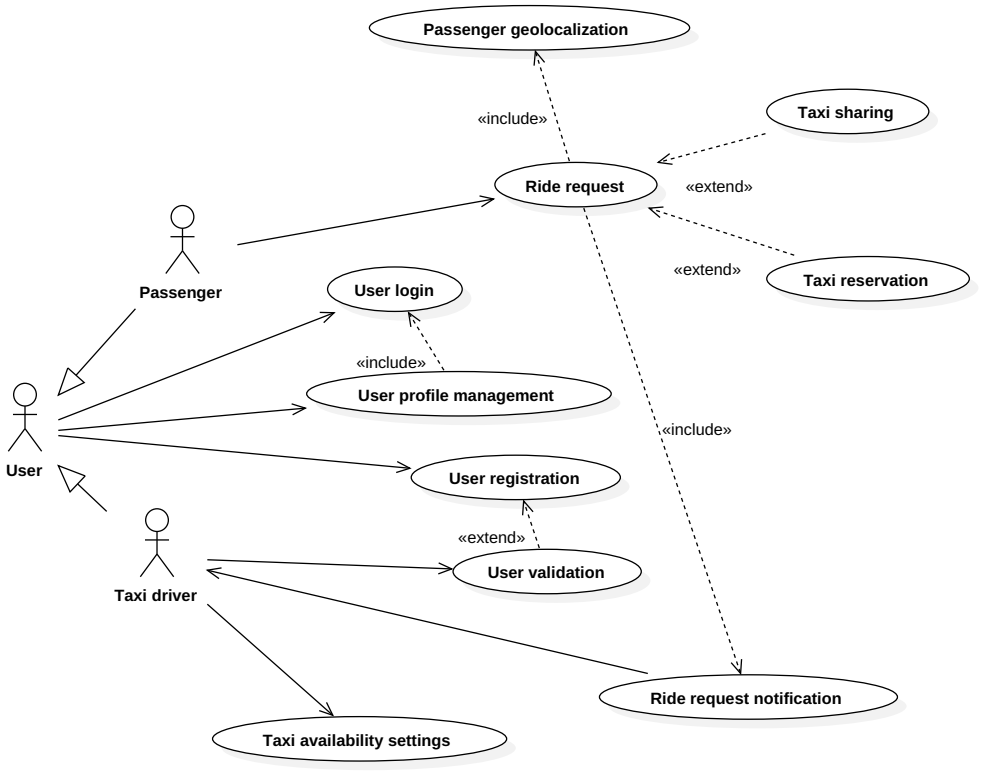

Figure 2.1: The comprehensive use-case diagram of all the functionalities implemented by the system.

2.3 User characteristics

The two kinds of users are *passengers* and *taxi drivers*. We suppose that both kinds of users have access to Internet.

Taxi drivers must be able to install and use the mobile application on their cellphone to answer the ride requests.

Passengers have to use the browser application or the mobile app.

2.4 Constraints

2.4.1 Regulatory policies

It's user responsibility to ensure that the use of the system complies with the local laws and policies.

The system must ask the user for the permission to acquire, store and process personal data and web cookies. The system must offer to the user the possibility to delete all the personal data.

2.4.2 Hardware limitations

the system has to run under the following conditions:

- App
  - 3G connection, at 2 Mb/s
  - 100 MB of free space
  - 2 GB of RAM
- Web application
  - Support for current version of IE, Firefox, Chrome, Safari, Opera as of 2013
  - 2 Mb/s Internet connections
  - 800x600 resolutions

2.4.3 Interfaces to other applications

2.4.4 Reliability requirements

The system must have a minimum availability of 98%.

2.4.5 Criticality of the application

The system is not employed in life-critical applications.

2.4.6 Safety and security considerations

The locations of the passenger and its destinations must be kept private unless the passenger chooses to share rides.

Only taxi drivers with a valid license must be able to use the service for security reasons.

2.5 Assumptions and dependencies

We assume that:

- the taxi is provided with a GPS navigator
- the taxi driver is able to reach the meeting point within 10 minutes from the given hour 90% of the times
- the taxi driver is able to reach the meeting point within 20 minutes from the given hour 100% of the times
- the passenger waits in the location until the taxi arrives
- the taxi driver picks up the correct passenger
- the taxi driver correctly updates his status (off shift, available, busy)
- the passenger specifies the correct location

2.6 Future extensions

The system will be implemented in order to offer the possibility of further extensions:

- Provide secure and reliable methods for ride payments. These functionalities will give the users the opportunity to pay for a taxi service using credit card informations stored in a secure way in their profiles.
- Offer a taxi rating system which allows the users to evaluate the specific taxi service provided.

- Monitor user reliability, recording when they do not show up at the meeting point or how many minutes later they arrive, in order to limit their access to the service when their reliability is bad.
- Create a fidelity score for every user, which gives the opportunity to benefit from special offers. The score increases every time the user makes use of the taxi service, proportionally with the distance covered.

Chapter 3

Specific requirements

3.1 External interface requirements

3.1.1 User interfaces

3.1.2 Hardware interfaces

3.1.3 Software interfaces

3.1.4 Communications interfaces

3.2 System features

3.2.1 User registration

3.2.2 User login

3.2.3 Standard taxi call

Purpose

Any subscribed passenger shall be able to request a taxi either through the web application or the mobile app. After the request, the passenger is informed by the system about the waiting time and the code of the incoming taxi.

Requests shall be forwarded to available and active taxi drivers in the same zone of the passengers. Taxi drivers shall be able to accept or reject an

incoming request.

Scenario 1

Alice needs a taxi. She opens the myTaxiService mobile app on her phone, and selects "Call a taxi". She authorizes the application to access her GPS data, checks on the map that her position is correct and confirms the request.

The system forwards the request to Bob, the first taxi driver in Alice's taxi zone. Bob decides to accept the call: a map of Alice's position gets displayed on Bob's phone and the navigator starts.

The position of Bob is transmitted from Bob's phone to the system, which computes the ETA for the incoming taxi and shows it to Alice. Bob arrives and picks Alice up. Then he confirms that the passenger is on board.

When the ride is over, Bob taps on "Finish ride" so that the system knows that Bob is ready for another ride.

Scenario 2

Luke needs a taxi. He requests it and the request is forwarded to the first taxi driver in queue, Chewbacca. Chewbacca decides to reject the incoming request, so the system puts him at the bottom of the queue.

Luke's request gets forwarded to the new first taxi driver in queue, Han Solo. Han accepts the request and comes to pick up Luke.

Response sequence

The response sequence is illustrated in figure 3.2.3.

Associated functional requirements

1. The system must localize the passenger before he or she makes a taxi request.
    (a) (App) If GPS info is available and the passenger can be tracked within a radius of 50 m, then the passenger is presented with the option of using the current GPS position.
    (b) (App) If GPS info is not available or the precision is less than 50 m, then the app requests the passenger to insert a valid address.

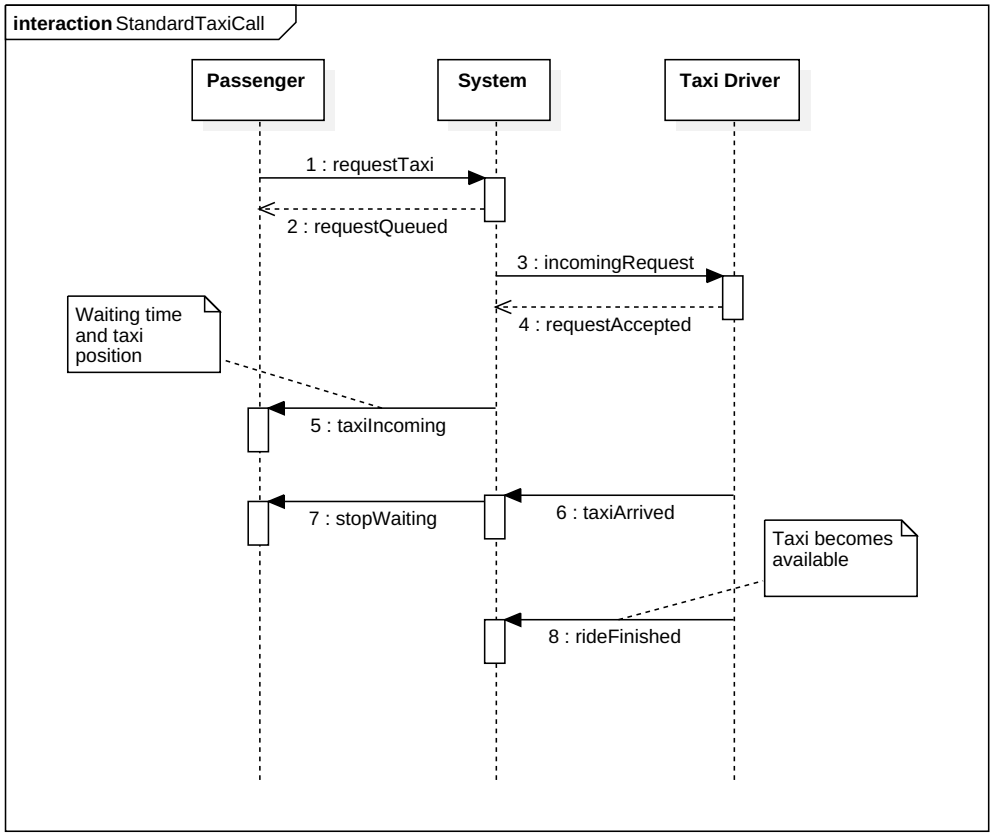

Figure 3.1: Sequence diagram of a successful taxi call.

Then the address is shown on a map and the passenger can confirm its position.

(c) (Web) In the web application the user is always requested to insert a valid address. Then the address is shown on a map and the user can confirm its position.

2. When the system knows the passenger's position, it presents the passenger the option to request a taxi.

3. The system must ask the passenger for confirmation before delivering the taxi request.

4. After the passenger's confirmation of a taxi request, the request is de-

livered to the first taxi in the queue for the taxi zone in which the user is located.

5. Taxi requests are forwarded only to active taxi drivers which are located in the same taxi zone, who are not currently busy.

6. Taxi requests are processed in order of arrival.

7. When a taxi driver receives a request, the mobile application shows a notification and emits a sound.

8. When receiving a request, the taxi driver is presented with the possibility of accepting or denying it.

9. After having accepted a request, the taxi driver is automatically marked as "busy".

10. After having accepted a request, the mobile application shows to the taxi driver a map with the location of the passenger, and automatically starts navigation instructions on the app preferred by the taxi driver (Google Maps, Waze, TomTom, etc.).

11. After the taxi request is accepted, the app shows the passenger a map with the current position of the incoming taxi.

12. After the taxi request is accepted, the passenger is informed about the ETA for the incoming taxi.

13. The ETA for the incoming taxi is fetched from the server every 90 s.

14. The ETA for the incoming taxi is computed by the system considering the distance from the taxi to the passenger and the current traffic conditions.

15. When the taxi is within 20 m from the passenger, the app asks the taxi driver to confirm that the passenger is aboard.

16. When the taxi driver confirms that the passenger is aboard, the taxi request is marked as fulfilled and the system stops to show the waiting time to the passenger.

17. The taxi driver is presented by the mobile application with the option to end the ride.

18. When the taxi driver ends the ride, he or she is marked as available and gets re-inserted in the taxi queue of the taxi zone where he or she is located.

3.2.4 Ride request notification to the taxi driver

3.2.5 Taxi availability handling

3.2.6 Taxi reservation

Purpose

Any subscribed passenger shall be able to reserve a taxi for a ride at a predefined time. The passenger has to specify in advance the origin and the destination of the ride, along with the starting date and time.

Scenario 1

John McClane will need a taxi to get to the airport tomorrow morning. He opens the web application of myTaxiService and decides to book a taxi for 6:00 AM for a ride from his home to the airport. He confirms the request.

The morning after, at 5:50 AM, the first taxi driver in the queue gets McClane's request and accepts it. He comes to pick up McClane and brings him to the airport.

Response sequence

Associated functional requirements

1. The system presents the passenger with the option to reserve a taxi.

2. The system asks the passenger the origin and the destination of the ride.

3. Origin and destination must be valid addresses.

4. If GPS info is present and accurate within 50 m, the passenger can specify "current position" as the destination of the ride.

5. The system asks passenger for the date and time of the ride.

6. The system lets the passenger enter only valid dates and times.

7. The system lets the passenger reserve a taxi from 48 hours to 2 hours before the actual ride time.

8. 10 minutes before the specified arriving time, the system allocates a taxi for the passenger by putting a request in the queue as described in subsection 3.2.3.

9. After the request is accepted, the passenger gets notified with the ETA of the incoming taxi along with its position.

3.2.7 Ride sharing

3.2.8 User profile management

3.3 Performance requirements

The system must support at least 1000 connected passengers at once, and at least 500 simultaneously active taxi drivers at any given time. 95% of requests shall be processed in less than 5 s; 100% of requests shall be processed in less than 10 s.

There is no limit on the total number of registered users.

3.4 Design constraints

3.5 Software system attributes

3.5.1 Reliability

3.5.2 Availability

3.5.3 Security

3.5.4 Maintainability

3.5.5 Portability

3.6 Other requirements