Requirements Analysis and Specification Document: myTaxiService

Chitti Eleonora, De Nicolao Pietro, Delbono Alex

October 28, 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.

RDBMS: Relational Data Base Management System.

JVM: Java Virtual Machine.

API: Application Programming Interface.

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 application 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 application:

- The possibility to choose the language used in every moment during every operation.

- A first screen that lets the user login in order 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 taxi drivers to take care of the request.

This is a list of what the users of the service can do.

- Passengers:

    - create an account
    - request a taxi
    - share a taxi with other passengers
    - reserve a taxi in advance for a specific hour

- Taxi drivers:

    - create an account
    - mark themselves as available or busy
    - accept or deny a lift request
    - get the exact position of the passenger, after having accepted a ride request
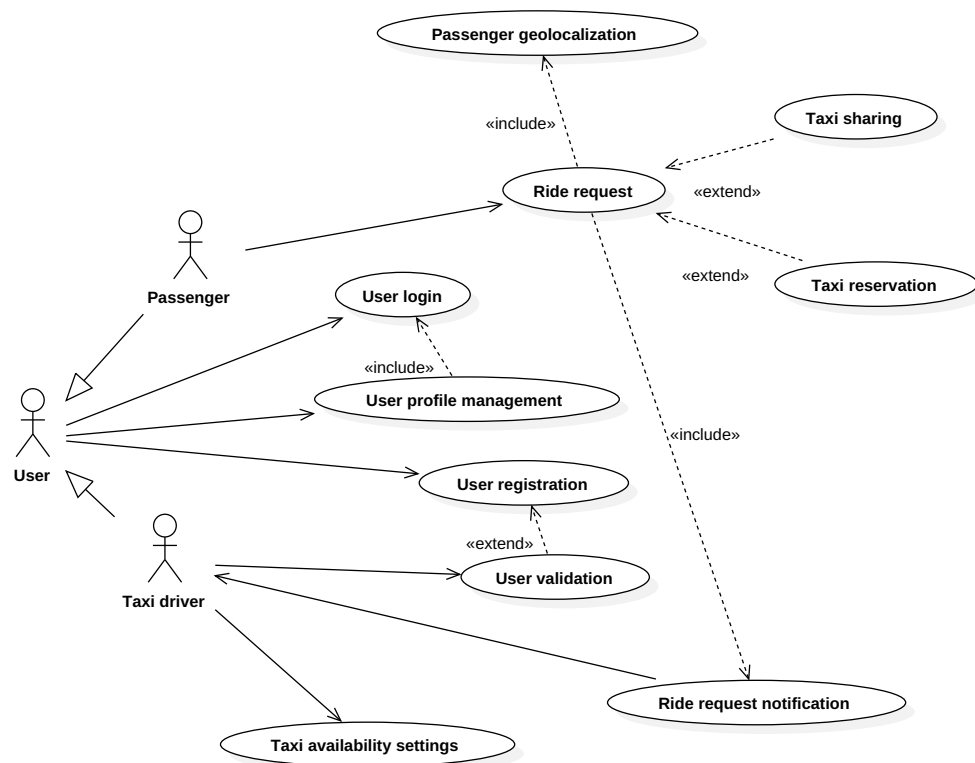
Figure 2.1: The comprehensive use-case diagram of all the functionalities implemented by the system.
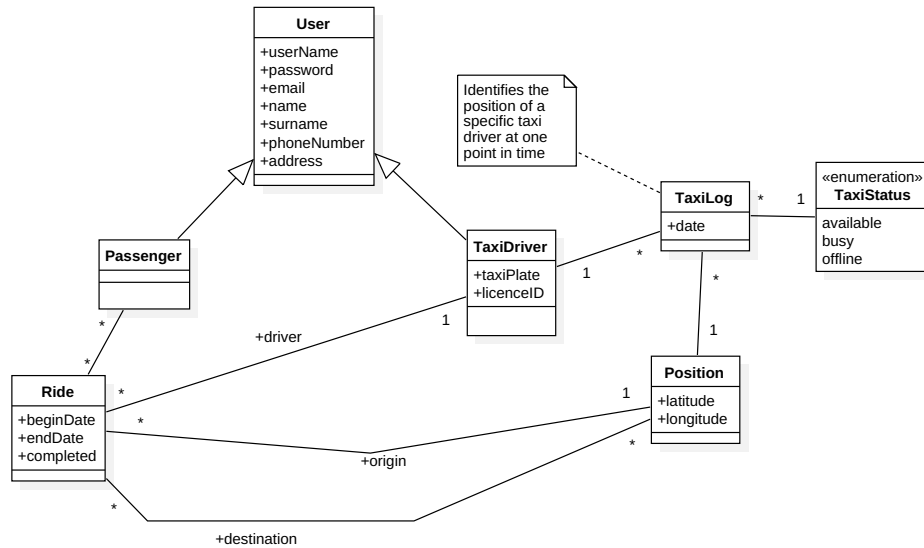
Figure 2.2: The comprehensive class diagram of the system.

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 resolution

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, for example:

- Provide secure and reliable methods for ride payments. This functionality allow the passengers to pay for the rides using credit card information stored in a secure way in their profiles.

- Offer a taxi rating system which lets the passengers evaluate taxi drivers.

- Monitor passengers' 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 passenger, which gives them the opportunity to benefit from special offers. The score increases every time the passenger uses of the taxi service, proportionally with the distance covered.

Chapter 3

Specific requirements

3.1   External interface requirements

3.1.1   User interfaces

- Web application

  1. The web pages must adhere to the W3C standards. In particular, the software conforms to the HTML5 [1] and CSS [2] standards.
  2. The web pages must be accessible also by text-only browsers.

- Mobile app

  1. The iOS version must adhere to the iOS Human Interface Guidelines [3].
  2. The Android version must follow Android design guidelines [4].

- Server back-end

  1. The server back-end must be configurable by means of a configuration text file.

- The clients must have an UI that is accessible to disabled people.

- All clients must show to the logged user a dashboard which allows access to all functionalities.

3.1.2   Hardware interfaces

This project does not support any hardware interfaces.

3.1.3 Software interfaces

3.1.4 Communications interfaces

The clients communicate with the server via HTTPS requests (port 443).

3.2 System features

3.2.1 User registration

Purpose

Any user can subscribe through the web application or the mobile app.

In both cases the user has to fill a registration form and must agree to the personal data policy according his/her country privacy laws, otherwise the registration request is denied.

As soon as the user has submitted all the data, the system verifies the consistency of the information and a confirmation mail is sent to check the availability of the user mail-address. After this last check the registration ends successfully.

The system has two kind of registration forms (so two kind of registered users) one for passenger and one for taxi drivers.

The taxi driver registration is more restrictive and requires more user's information. The system registers a user that claims to be a taxi driver only if he is able to demonstrate it with a currently valid taxi license.

Scenario 1

Alice, a normal citizen without a car, has just discovered the existence of myTaxiService web application and she wants to use it. She opens the home of myTaxiService page on web and clicks on "passenger registration".

She gives all the informations required and she authorises the personal data treatment.

The system verifies the submitted information and sends a confirmation mail. Alice checks the mailbox, opens the mail and clicks on the "confirm e-mail" link.

The system informs Alice that the registration has ended with success.

Scenario 2

Bob is a taxi driver that wants to subscribe to myTaxiService application. He downloads the mobile application from his phone app-store and once he

opens it, he selects "taxi driver registration". He fills the form, enters his licence ID and authorises the personal data treatment.

However he forgets to write his phone number on the form so the system warns him about the forgetfulness. Only after the complete and correct filling of the form, and the personal data treatment authorisation, the registration is one step near the successfully end.

The system verifies the informations submitted and sends a confirmation mail. Bob checks the mailbox, opens the mail and clicks on "confirm e-mail".

The system informs Bob that the registration has ended with success.

Associated functional requirements

1. On registration, the user can choose to register as a passenger or as a taxi driver.

2. Passengers must provide the following information:

   - e-mail address
   - username
   - password
   - name
   - surname
   - address (optional filling)
   - phone number (optional filling)

3. Taxi drivers must provide the following information:

   - e-mail address
   - username
   - password
   - name
   - surname
   - address
   - phone number
   - taxi license ID
   - taxi number-plate

4. There mustn't be another user already subscribed with the same user-name or e-mail.

5. The username must match the regular expression
   "`[a-zA-Z][a-zA-Z0-9]{2,20}`"

6. The system accepts a password that contains at least one number and one capital letter and that has a minimum length of eight characters.

7. The system must ask for the password twice.

8. The system accepts the password only if the same one is entered identically both times.

9. If the personal data treatment is not authorised, the subscription is canceled.

10. The system must allow the user to abort the registration process at any time.

11. Email confirmation process:

    (a) The subscription ends successfully when the user clicks on the link in the confirmation e-mail.

    (b) After two days without any answer, the systems sends another confirmation e-mail.

    (c) After seven days user's registration info are deleted and the user may re-try the registration process.

3.2.2   User login

Purpose

Any subscribed user can login to use myTaxiService services. The system requires the user to provide nickname and password, or e-mail and password, to log in correctly.

If the user doesn't remember his password, a new one is sent to the user's e-mail address. If the user clicks on the link in the e-mail, the password is reset and, as soon as he logs in, the system asks him to choose a new one.

Scenario 1

Bob opens the home page of myTaxiService on the web and clicks on "login". he's asked to enter the nickname or the e-mail and the password. He doesn't recall his nickname, so he enters the registration e-mail and the password. He clicks on "enter". Everything is correct so he can access to the services as a logged user.

Scenario 2

Alice opens the home of myTaxiService page on web and clicks on "login". She is asked to enter the nickname or e-mail and the password. She enters nickname and password and selects "enter". However, the entered password isn't correct so the system asks her to re-enter it. After some failed attempts she select "password forgot". The system sends to her a new password via e-mail. She enters that one and the access is permitted, but before successfully ending the login the system requires her to change the password immediately. Alice changes her password and finally logs in to the services.

Associated functional requirements

1. In order to log in, the user must insert either a nickname, or the registration e-mail, but not both.

2. On login, the system must grant to the user access to his/her account if and only if the following conditions are met:

   (a) The inserted nickname corresponds to a username of an existing user, or the inserted e-mail corresponds to the registration e-mail of an existing user.

   (b) The inserted password is the same of that of the user identified above.

3. The system sends an e-mail after the "password forgot" button is selected.

4. The system reset the user's password only after the link on e-mail is clicked.

5. If the password entered is wrong a new attempt can be made only 10 seconds later.

6. The system accepts a new password that contains al least one number and one capital letter and that has a minimum length of eight characters.

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 in-

formed 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 figures 3.1 and 3.2.

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.
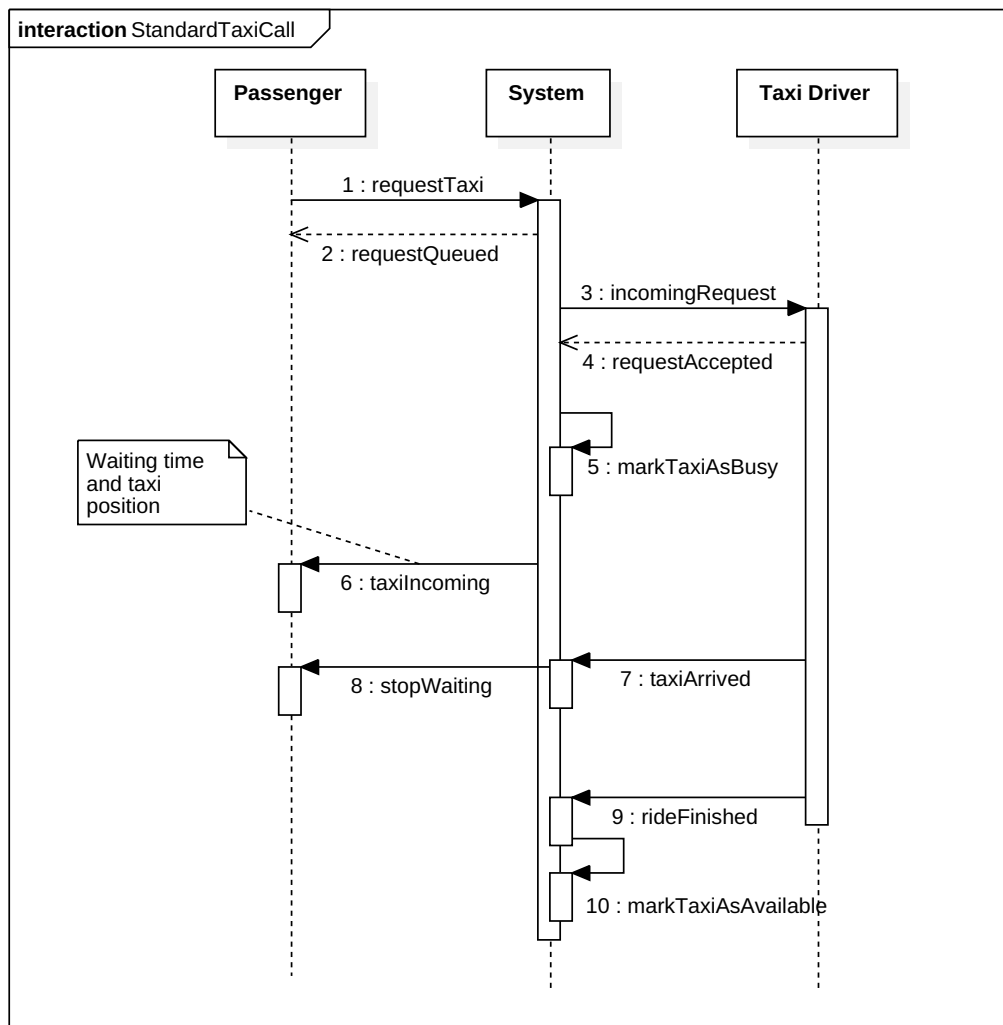
Figure 3.1: Sequence diagram of a successful taxi call, picked up by the first taxi driver called by the system.

interaction StandardTaxiCall (with refusal)

| Passenger | System | Taxi Driver 1 | Taxi Driver 2 |

1 : requestTaxi

2 : requestQueued

3 : incomingRequest

4 : requestRefused

5 : dequeueTaxiDriver

Waiting time
and taxi
position

6 : incomingRequest

7 : requestAccepted

8 : markTaxiAsBusy

9 : taxiIncoming

10 : taxiArrived

11 : stopWaiting

12 : rideFinished

13 : markTaxiAsAvailable

Figure 3.2: Sequence diagram of a taxi call when the first taxi driver to receive the request refuses the call and the second one accepts it.

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.

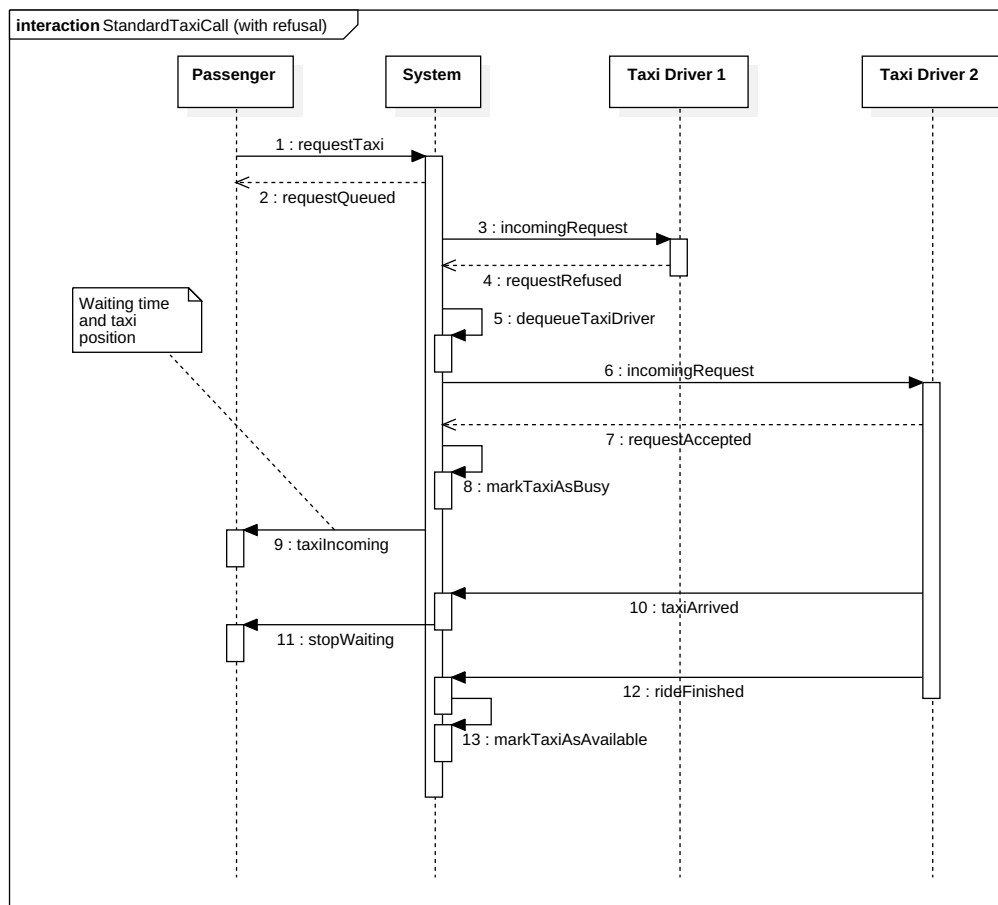3. The system must ask the passenger for confirmation before delivering the taxi request.

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

Purpose

After a taxi driver has informed the system about his/her availability (section 3.2.5) he/she will be able to receive ride request notifications. The taxi driver receives the notification on his/her cell phone and has one minute to accept or reject the request. If after one minute the request is not answered, the system will consider the request as refused.

Scenario 1

Travis, who suffers from chronic insomnia, every night waits in his taxi for an incoming request. When a user requests a taxi, the system processes the request and decides which taxi driver to send.

For a new incoming request, the system chooses Travis, who is on the top of the queue and in the same taxi zone of the passenger. The system sends him a notification on his mobile app. Travis reads the message which reports location and time of the meeting and decides to accept the request pushing the appropriate button on the display.

The system acknowledges the decision and waits for Travis to notify that the passenger is on board (last part of 3.2.3).

Scenario 2

The system receives a new incoming request from a user and selects Frank Martin as designated taxi driver (section 3.2.5). The system sends him a notification and waits for the reply.

Frank has done a very demanding ride and decides to refuse the request using the interface of the app. The system receives Frank's decision and looks for another available taxi on the same taxi area.

Response sequence

The use case associated to the response sequence is shown in figure 3.2.

Associated functional requirements

1. The taxi has to be in the taxi zone queue.

2. The GPS on the mobile phone fo the taxi driver is available.

3. The GPS position of the taxi must be in the same area of the user's.

4. The app must show the request notification to the taxi driver.

5. The system allows the taxi driver to accept or decline the request using the app.

6. The app must produce sounds or visual notifications as the taxi driver has chosen.

7. The system must use timeouts in order to prevent infinite pending requests.

8. The system has to provide the taxi driver with information about the time and place of the meeting with the passenger.

9. The system gives to the taxi driver one minute to reply. Else, the request is automatically refused.

3.2.5   Taxi availability handling

Purpose

When a taxi driver is ready for a ride, he/she shall be abòe to notify the system using his mobile phone. With the app he/she can change his status from "not in service" to "in service".

When the system receives the status update from the taxi driver, it inserts him/her in the right queue for his/her taxi zone using the information from the taxi GPS if the new status is "in service", or it removes him/her from the queue if the new status is "not in service".

After a ride, a taxi driver has to notify, using the dedicated section of the app, that the ride is over. The system shall reinsert the taxi into the right queue.

The queues are FIFO and when a taxi driver refuses a ride, the system moves the taxi to the bottom of the same queue.

When a taxi driver accepts a ride, the system marks the taxi as busy and removes it from the top of the queue.

Scenario 1

Ernie gets in his taxi, ready to start his working day. He takes out his phone from his pocket and after logging in he changes his status in "in service".

The system acquires the changing and retrieves the GPS position of Ernie's taxi. It analyses the data and puts the taxi in the right queue.

After thirty minutes the system has an incoming request from the Ernie's zone. The first taxi on the top of the queue is Ernie's. The system sends a notification to Ernie and waits for a reply.

Ernie receives the request and accepts the meeting. The system pops Ernie from the top of the queue and marks it as busy.

When Ernie accomplishes the ride, he notifies the system, which retrieves the new GPS position and inserts Ernie's in the queue.

Response sequence

The response sequence associated with this functionality is shown in figure 3.3.

Associated functional requirements

1. The system knows the information about the GPS installed on the taxi.

2. The mobile app has to offer to the taxi driver the "change status" function and the "ride complete" function.

3. The system must notify the taxi when a request is incoming.

4. The system inserts the taxi in the right queue and changes it if the taxi changes area.

interaction TaxiAvailabilityHandling

The System | TaxiDriver

1 : inService

2 : queue Taxi

3 : incomingRequest

4 : requestAccepted

5 : dequeue Taxi

6 : taxiArrived

TaxiDriver
returns
"inService"

7 : rideFinished
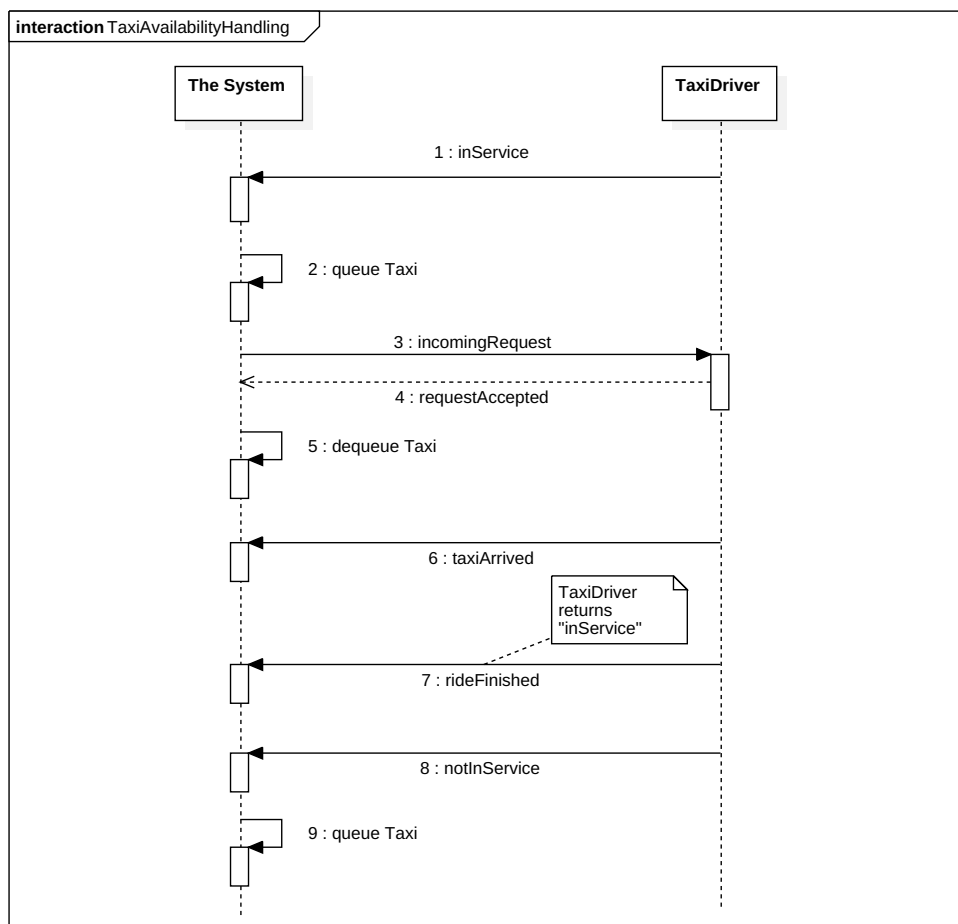
8 : notInService

9 : queue Taxi

Figure 3.3: Sequence diagram of an accepted taxi call.

24

5. For every new request, the system chooses the taxi driver on the top of the queue.

6. The taxi driver has to reply in one minute to the request, else the ride is automatically refused.

7. The system manages the queues using the FIFO policy.

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

The response sequence is illustrated in figure 3.4.

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.

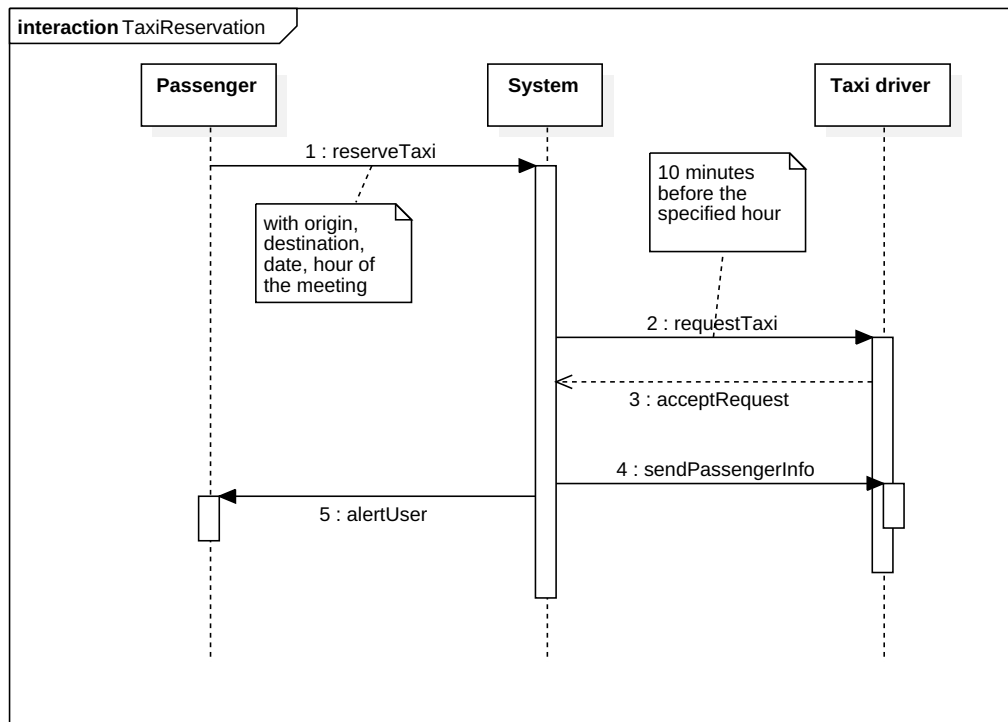6. The system lets the passenger enter only valid dates and times.

Figure 3.4: Sequence diagram of a successful taxi reservation.

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

Purpose

Every subscribed passenger shall be able to activate the ride sharing function in the mobile app or in the web interface. When this mode is enabled, the passenger has to provide the system also with a destination for the ride.

The system, before allocating a taxi, inserts the pending ride in the set of shared rides and tries to identify possible sharing solutions. With an

adequate algorithm it can propose every feasible sharing solution to the user.

The first user who reserves a taxi is called the "owner" of the ride. The passenger can choose to join one of the pending rides or to refuse the proposal, becoming the owner of a new ride.

In the first case the system informs the owner of the ride. The new passenger has to go to the meeting point with the other user in time. If he/she is not able to do that the taxi does not wait for him/her.

In the second case the system allocates the taxi (3.2.5) and applies the standard procedure 3.2.3. If there is another user who wants to share the ride with compatible options, the system allows it and redirects this last user to the meeting point with the new user and informs the previous user and the taxi driver.

Scenario 1

Batman needs a taxi and decides to enable the sharing mode on his cell phone. The app requires also the destination point of Batman's travel.

When Batman submits the form, the system matches the path of Batman's ride with every pending ride started from Batman's area. It finds only one compatible ride and sends it to Batman.

Batman finds that it is Joker's pending ride and decides to refuse the share. Then the system allocates a new taxi.

A new user, Robin, is in Batman's area and wants to take a taxi. He decides to enable the sharing function and the system proposes him Batman's and Joker's ones. He chooses Batman's: the system, which has already communicated location and time to Robin, notifies Batman and the taxi driver that there is an additional passenger.

When the taxi arrives, the taxi driver confirms from his mobile app how many passengers he has picked up.

Response sequence

The response sequence is illustrated in figure

Associated functional requirements

1. The system has to know the starting point of the new passenger in order to provide physical sharing solutions. It has to compute the estimated walking time to reach the meeting location and compare it with the estimated taxi arrival time.

2. The user can enable the sharing function both in mobile app and in web interface.

3. The user can choose between possible sharing solutions or a new taxi.

4. The system has to communicate to the taxi driver the possible presence of the new passenger.

5. The system must be able to analyse the current sharing situation.

6. The system must evaluate the number of seats available in every taxi.

7. The taxi driver can insert the number of passengers who have been picked up.

8. After the user chooses to enable the sharing mode, the system has to specify, for every sharing ride, the location and the time.

3.2.8   User profile management

Purpose

Any subscribed user can view or update his profile information.

The system allows taxi drivers to use the service only if they load a new medical certificate every two years. A reminder e-mail is sent to the taxi driver three months before the expiration.

Scenario 1

Alice, a myTaxiService passenger without a car, wants to count how many times she has used a taxi in the last month. She opens the home page of myTaxiService page on the web site and clicks on "login". After she has logged in correctly, she clicks on "load profile" and all the information about her account appears on screen, including the taxi request list.

Scenario 2

Gabriele's girlfriend has discovered his password but he doesn't want her to know where he has been. So he decides to change his password immediately. He opens the app on her cell phone, he selects "load profile" and after he selects "modify password". The system asks him to enter the old password and the new one two times to avoid errors. Once he has verified that everything's ok, he selects "done", and the system confirms that the password has been successfully set.

Scenario 3

A recall mail is sent to Bob, a myTaxiService user that has registered as taxi-driver, to notice him that he has to update his profile with a new medical certificate. A week later, after the medical checkup that ensures his good health, he opens the app on his cell phone, he selects "load profile", then "modify". He enters the new medical certificate info, then confirms selecting "done". The system confirms that the medical certificate has been successfully inserted.

Associated functional requirements

1. Logged passengers can:

   - view user profile
   - modify an information
   - delete account
   - view the latest taxi request
   - view taxi requests list

2. Logged taxi drivers can:

   - view user profile
   - modify an information
   - delete account
   - view the latest taxi request accepted
   - view requests accepted list

3. The system sends a recall mail three months before the expiration of the two years' validity of the last medical certificate submission.

4. Delete the account:

   (a) The system allows to delete the user's account with an option on screen.

   (b) After "delete account" is selected, the user has to ensures his decision selecting on screen "proceed": the system deletes the account only after user's confirmation.

   (c) When an account i deleted also all the informations in it are lost.

5. Modify the profile:

(a) The system allows to change one or more information on the profile: any information can be changed, even the nickname or e-mail.

(b) The system must verify the consistence of the modified information. It has to exist and it has to be unique (i.e. there mustn't be another user that has already entered the same info).

(c) If the e-mail is changed The system sends a confirmation mail on the new address.

(d) The system replaces the old information with the new one as soon as it has been verified.

(e) The system accepts a new password that contains al least one number and one capital letter and that has a minimum length of eight characters.

(f) The system accept a new password only if the old one has been submitted correctly before.

(g) The system asks for the new password two times, when a user wants to change it.

(h) The system accepts the new password only if the same one was entered twice.

(i) The system must allow the user to abort the modification at any time.

(j) The old information isn't replaced until the "done" choice isn't selected by the user.

3.2.9 Programmatic interface

Purpose

The back-end software component shall expose programmatic interfaces to ease the creation of modules that implement new function (e.g. taxi reservation, taxi sharing).

The programmatic interface will consist in Java public interfaces, classes and methods. Modules are required to use this public interface to carry out their functions, and will be limited in what they can do by this library.

The public interface shall provide safe and controlled access to the data, throwing suitable exceptions when an illegal action is attempted by a module.

Associated functional requirements

1. Creation of a new user (taxi driver or passenger)

2. Modification of user profile

3. Reservation of a taxi for a specific path in a specific time

4. Updating the location of an user

5. Taxi availability and status handling

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

The system is not currently designed to run in a distributed environment, but it may be the case in future versions. The reliability of the system is strictly related to the reliability of the server it runs on.

3.5.2 Availability

1. The system must guarantee an availability of 98%.

3.5.3 Security

1. All the communications between server and clients must be protected by strong encryption using the SSL protocol.

2. All attempts of establishing an unsecure communication channel (e.g. plain HTTP) with the server must be refused.

3. Users' passwords must not be stored in plain text in the database: instead, they must be hashed and salted.

4. The system must log all login attempts with IP addresses for at least 7 days.

5. The modules must state clearly which data they will need to read and write.

3.5.4 Maintainability

The code should be well documented using JavaDoc in order to be understood and fixed by developers later. The system must provide a configurable logging function for debugging purposes.

The development of the software will follow the object-oriented Model-View-Controller pattern and the separation of concerns principle.

The build from source code in the Version Control System will be completely automated, as well as unit testing.

The system will support modular extensions in order to have a stable core and divide the core development from the external modules, debugging them separately.

3.5.5 Portability

The back-end server software will be written in Java. It must run on every platform that supports the JVM, and extensive testing will be carried out on every OS to make sure that the system's really portable.

There will not be host-dependent dependencies.

The web application shall run on every modern browser. The mobile application must be supported by the last 2 major versions of Android and iOS.

3.6   Other requirements

Bibliography

[1] W3C, *HTML5 - W3C Recommendation 28 October 2014*, `http://www.w3.org/TR/html5/`.

[2] W3C, *Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification*, `http://www.w3.org/TR/CSS21/`

[3] Apple, *iOS Human Interface Guidelines*, `https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/`

[4] Google, *Android Developers - Design* `https://developer.android.com/design/index.html`