Politecnico di Milano
Academic Year 2015/2016
Software Engineering 2: "myTaxiService"
Requirement Analysis and Specification Document

Massimo Schiavo, Marco Edoardo Cittar

November 2, 2015

Contents

1 Introduction

1.1 Description of the problem

We will design and implement myTaxiService, a new web and mobile app to optimize the taxi service in big cities. It should simplify the access of passengers to the service and guarantee a fair management of taxi queues.

The users will have to register and login in order to use the application, then they can request a taxi and be informed about the code of the incoming taxi and the waiting time.

We suppose the city is divided in zones and every one of them has a queue of available taxis present in the zone, whose position is calculated according to GPS. After a request arrives, the system informs the first taxi in the queue of the zone from which the request came. If the taxi accept the request, the system sends the confirmation to the user. If not, the system will put it at the end of the queue and forward the request to the next available taxi.

To accept ride requests, taxi drivers will have to login through the mobile app like normal users. Then they can set themselves as available and receive ride requests.

A user can also reserve a taxi by inserting origin, destination and time of the ride. The request must be submitted at least two hours before. The system will allocate a taxi and notify the user 10 minutes before the ride.

1.2 Actors

- Guest: the guests are users who are not registered yet. They must sign themselves up into the system in order to use the features available to registered users.
- Registered user: this type of user, after successful login, has access to all the features of the application as a customer. They can request rides, be them simple or detailed, and receive notifications after a ride has been confirmed.
- Driver: they have the functionalities of both customer and worker, so they can set themselves as available and so can be notified when a new ride request arrives, but can also use the application as a registered user when they are not working.

1.3 Goals

myTaxiService should have these features:

- Guests should be able to:
  - [G1] Register themselves into the system
- Users should be able to:

  - [G2] Log into the system
  - [G3] See number of available taxis of the zone he's in
  - [G4] Make a request for a simple ride
  - [G5] Make a request for a detailed ride
  - [G6] Read the confirmation of the request

- Drivers should be able to:
  - [G2] Log into the system
  - [G7] Set themselves as available
  - [G8] Read and accept ride requests

- The system should:
  - [G9] Notify passengers after the confirmation of a normal request
  - [G10] Notify passengers 10 minutes before the ride reserved through a detailed request
  - [G11] Forward requests to the first taxi in queue
  - [G12] After 30 seconds, forward the request to the second taxi in queue and put the first at the end

1.4 Definitions, acronyms, abbreviations

1.4.1 Definitions

1.4.2 Acronyms

1.4.3 Abbreviations

- [Gn]: n-goal.

1.5 Reference documents

- Specification document: Assignments 1 and 2 (RASD and DD).pdf
- IEEE Standard For Requirement Specification.pdf

1.6 Document overview

This document is essentially structured in four part:

- Section 1: Introduction, it gives a description of document and some basical information about software.
- Section 2: Overall Description, gives general information about the software product with more focus about constraints and assumptions.

- Section 3: Specific Requirements, this part list requirements, typical scenarios and use cases. To give an easy way to understand all funcionality of this software, this section is filled with UML diagrams.
- Section 4: Appendix, this part contains some information about the attached .als file and some described screenshot of software used to generate it.

2 Overall Description

2.1 Product perspective

2.2 User characteristics

2.3 Constraints

2.4 Assumptions

- When the app or the web page is closed the user automatically logs out.
- If a driver sees a ride request, he always accepts it.
- Users can't cancel ride requests.
- If the system sends a notification the user will always receive it.
- Users can make an unlimited number of daily ride requests.
- In case of empty queue 10 minutes before a detailed ride, the application will always manage to find one within the time of the ride.
- Users insert existing addresses when making requests.

3 Specific Requirements

3.1 External interface requirements

3.2 Functional Requirements

3.2.1 Goal1

3.2.2 Goal2...

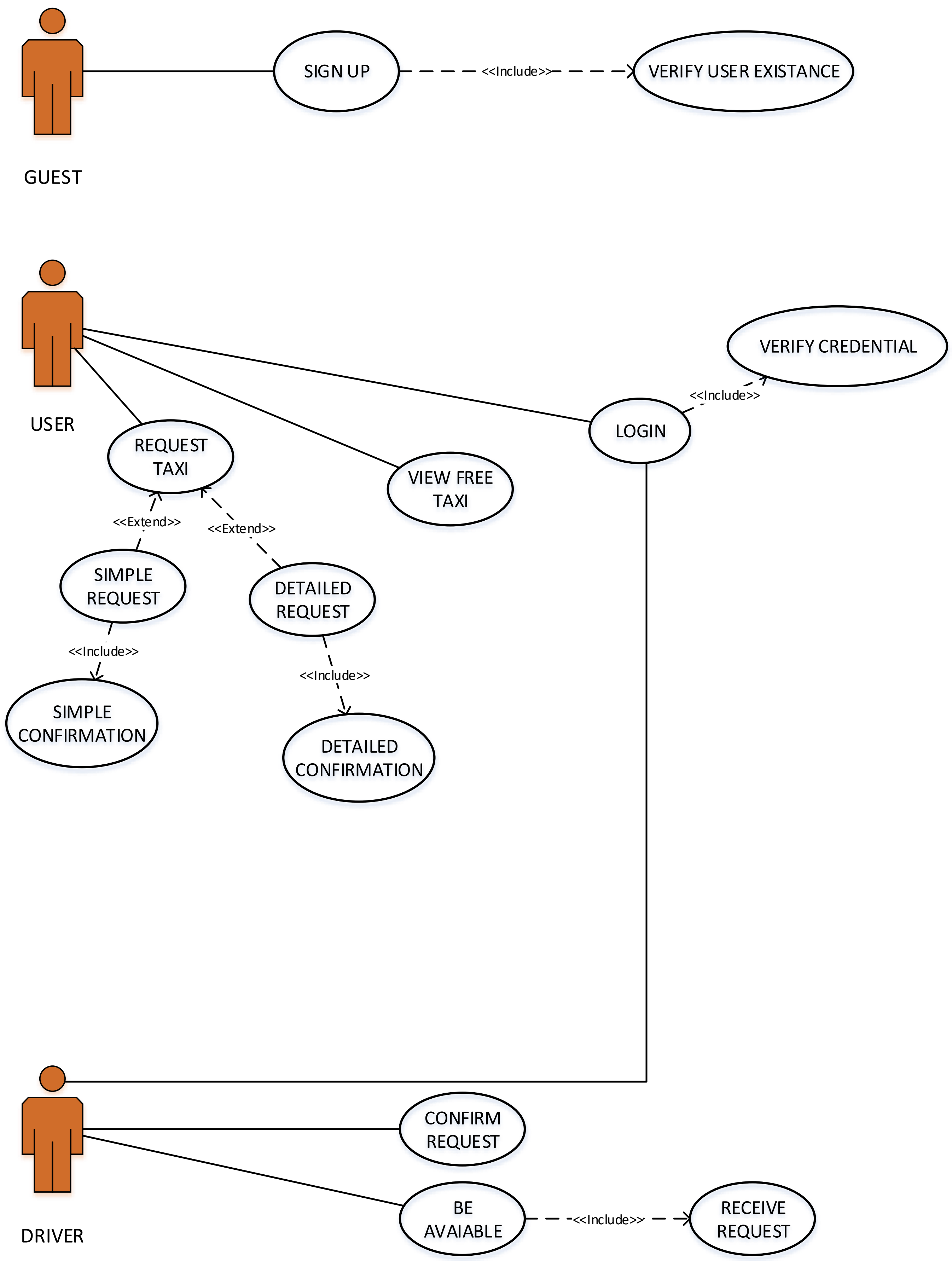
3.3 The world and the machine

3.4 Scenarios

3.5 UML models

3.5.1 Use case diagram

USE CASE


GUEST
SIGN UP
<<Include>>
VERIFY USER EXISTANCE
USER
VERIFY CREDENTIAL
<<Include>>
LOGIN
REQUEST TAXI
VIEW FREE TAXI
<<Extend>>
<<Extend>>
SIMPLE REQUEST
DETAILED REQUEST
<<Include>>
<<Include>>
SIMPLE CONFIRMATION
DETAILED CONFIRMATION
DRIVER
CONFIRM REQUEST
BE AVAIABLE
<<Include>>
RECEIVE REQUEST

3.5.1.1 Sign up

Actor	Guest
Goal	[G1]
Entry conditions	NULL
Flow of events	1. Guest on the login page clicks on the “Register” button to start the registration process. 2. Guest fills in at least all the mandatory fields. 3. Guest clicks on “Submit” button.
Exit conditions	Guest successfully ends the registration process and become a User. From now on he will be able to log in with the credentials submitted before and start using the application.
Exceptions	1. Some mandatory field is not filled. 2. Chosen username is already taken. 3. Email is already in use. The exceptions are handled by showing an error and making the guest repeat the registration process from point 2 of the event flow.

3.5.1.2 Log in

Actor	User
Goal	[G2]
Entry conditions	Guest must be already registered into the system.
Flow of events	1. User inserts username and password. 2. User clicks on the “Log in” button.
Exit conditions	The credentials are succesfully validated and the user is logged into the application.
Exceptions	An error is shown when the credentials inserted are not correct, making the user repeat the login process.

3.5.1.3 Simple request

Actor	User
Goal	[G4]
Entry conditions	User must be logged in.
Flow of events	1. User clicks on the “Request a taxi” button 2. Chooses “Simple ride”
Exit conditions	User successfully makes a request which is still pending until a driver accepts it.
Exceptions	None. If there’s no taxi, the system waits until one becomes available.

3.5.1.4 Detailed request

Actor	User
Goal	[G5]
Entry conditions	User must be logged in.
Flow of events	1. User clicks on the “Request a taxi” button 2. Chooses “Detailed ride” 3. Fills the required fields, that are origin, destination and time of the ride 4. Clicks on “Submit”
Exit conditions	User succNone. essfully makes a request which is still pending until 10 minutes before the specified time, then the system allocates a taxi to the ride.
Exceptions	If the user inputs incorrect or If there is no taxi in that zone at the moment, the application waits until one becomes available.

3.5.1.5 View free taxis

Actor	User
Goal	[G3]
Entry conditions	User must be logged in.
Flow of events	1. User clicks on the “Taxi availability” button.
Exit conditions	The application shows the number of available taxis in the zone the user is in, according to GPS.
Exceptions	None.

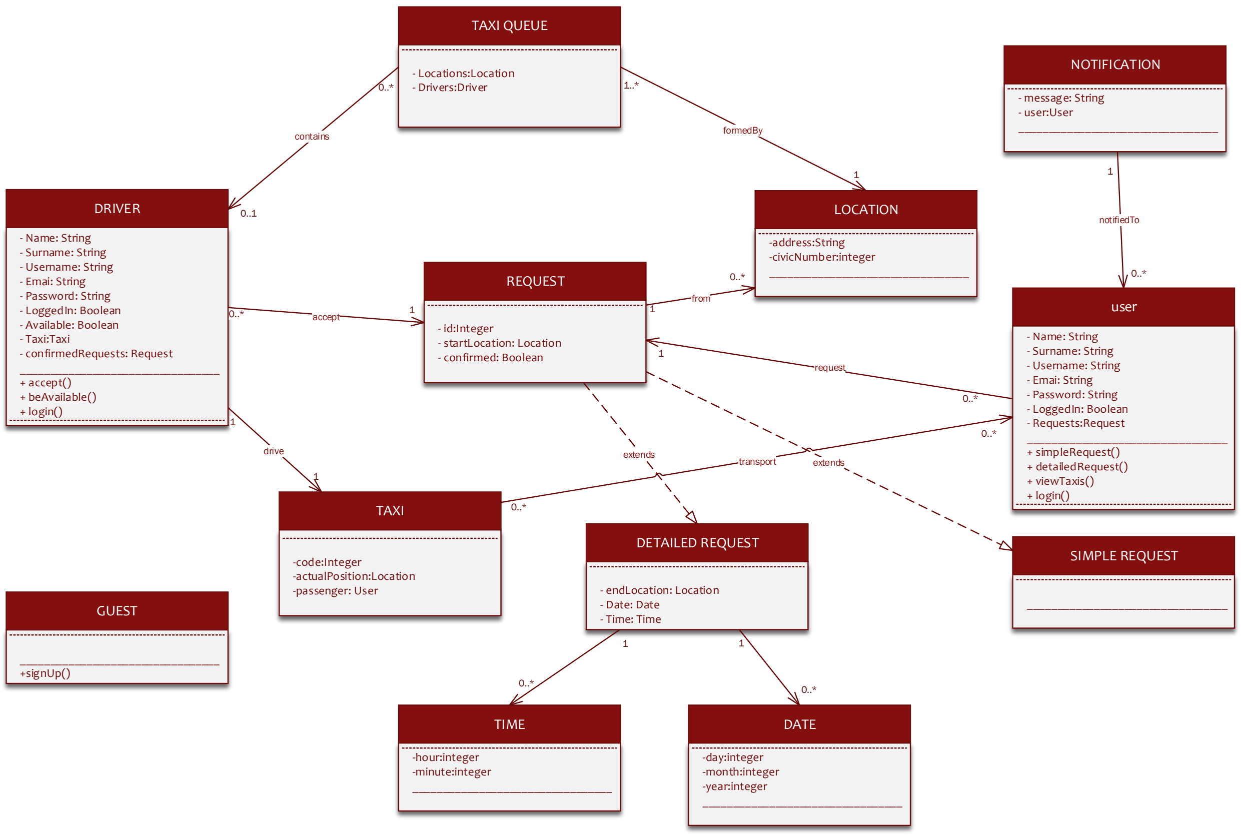
3.5.1.6 Be available

Actor	Driver
Goal	[G7]
Entry conditions	Driver must be logged in and not already available.
Flow of events	1. Driver clicks on the “Available” button.
Exit conditions	The driver is set as available, put into the queue of taxi in his zone and can now receive ride requests.
Exceptions	If the driver is already available the appication shows an error.

3.5.1.7 Confirm request

Actor	Driver
Goal	[G8]
Entry conditions	Driver must be logged in and available. Also he must have been notified by the system for a ride request.
Flow of events	1. Driver opens notification. 2. Driver accepts it by clicking on the “Accept” button.
Exit conditions	The driver is now in charge of the ride and the system informs the user about the taxi details.
Exceptions	If the driver doesn’t accept within 30 seconds the request is forwarded to the next available driver and this one is put at the end of the queue of his zone.

3.5.2 Class diagram


TAXI QUEUE
- Locations:Location
- Drivers:Driver
NOTIFICATION
- message: String
- user:User
DRIVER
- Name: String
- Surname: String
- Username: String
- Emai: String
- Password: String
- LoggedIn: Boolean
- Available: Boolean
- Taxi:Taxi
- confirmedRequests: Request
+ accept()
+ beAvailable()
+ login()
LOCATION
-address:String
-civicNumber:integer
REQUEST
- id:Integer
- startLocation: Location
- confirmed: Boolean
user
- Name: String
- Surname: String
- Username: String
- Emai: String
- Password: String
- LoggedIn: Boolean
- Requests:Request
+ simpleRequest()
+ detailedRequest()
+ viewTaxis()
+ login()
TAXI
-code:Integer
-actualPosition:Location
-passenger: User
DETAILED REQUEST
- endLocation: Location
- Date: Date
- Time: Time
SIMPLE REQUEST
GUEST
+signUp()
TIME
-hour:integer
-minute:integer
DATE
-day:integer
-month:integer
-year:integer
contains
0..*
0..1
formedBy
1..*
1
notifiedTo
1
0..*
accept
0..*
1
from
1
0..*
request
1
0..*
drive
1
1
transport
0..*
0..*
extends
extends
1
0..*
1
0..*

3.5.3 State machine diagram

3.6 Non functional requirements