POLITECNICO MILANO 1863

myTaxiService

REQUIREMENT ANALYSIS AND SPECIFICATION DOCUMENT

Paolo Antonini 858242
Andrea Corneo 849793

version 1 – 6th November 2015

Contents

1
Introduction

1.1 Purpose

Improving the public transport is crucial for a modernizing city and Milan, as such, has already taken some steps in this direction. However, until now the taxi service has been overlooked, if not neglected. Recently, though, the government of the city requested a comprehensive study about possible ways to improve its taxi service. This document will outline a computer system which will hopefully help the city council to achieve this goal.

As a result, the natural addressee of this document is the municipal government itself, which will evaluate the feasibility of the project we are suggesting. Once the project is approved, this document, along with other ones that will follow, will constitute a solid guide to developers' work.

1.2 Scope

To be more specific, the council wants the access to the service to be simplified, as well as to guarantee a fair management of taxi queues, which will benefit both passengers and taxi drivers.

These targets can be reached by implementing myTaxiService, an inclusive computer system which lets users make an immediate or delayed reservation for taxi rides, and manages the requests in order to minimize the waiting time for all passengers. Users will be able to access the new service through both a mobile application and a website; nevertheless, the current, phone-based system continues to be operative, in order to meet the needs of those people who are unfamiliar with the improved service, and also of those who are not able to access it.

Moreover, taxi drivers will be supported in their activity with a mobile application as well, which will run on their smartphone and enables them to receive the requests for taxi rides.

1.3 Definitions, acronyms, and abbreviations

In order to avoid ambiguity, we find it appropriate to explicitly define some terms which will recur throughout the document:

Customer the person who makes use of the reservation service through one of the available methods (mobile application, website, phone call); customers can register to the service, thus gaining access to some convenient additional functions; otherwise they are considered as guests; they may be referred to as users as well.

Passenger the person travelling in the taxi; typically, but not necessarily, the passenger is also the customer who made the reservation.

Guest a customer who is not registered to the service; functionalities at his disposal are limited, but the main services are still available to him, as will be detailed farther in this document.

Mobile application a program designed to work on mobile devices, namely smartphones and tablets; in the specific context of this document, we have two or them, since both the customers and the taxi drivers need one.

Website in the specific context of this document, the website is a particular web page on the Internet, reachable with a browser (e.g. Microsoft Internet Explorer, Google Chrome, Mozilla Firefox, ...), by means of which a user can access to the service.

System in the specific context of this document, the system is the actual, computer-based provider of the service; notice that the two mobile applications, the website and phone calls are ways to access it.

Taxi queue list of taxis assigned to a specific area of the city; the policy that regulates the queue is a "First In, First Out" one, which means that the first taxi that is enlisted in the queue, is the first who receives a reservation, as well; there are some exceptions, which will be described farther in this document.

Area in order for the system to guarantee a fair management of taxi queues, the city is divided in portions of approximately 2 km^2, whose borders are well-defined by road intersections.

Reservation through the system, the user can reserve a taxi ride; actually, his reservation may be either immediate, meaning that he can ask for a taxi ride and receive immediately the estimated time of arrival, or delayed, which is to say that he books a ride by specifying the origin, the destination and the time of the needed ride.

1.4 References

Farther in this writing we will refer to some external documents, which are listed below:

Assignments 1 and 2 section 2, parts I and II;13th October 2015; Politecnico di Milano. This is to be considered as the high level description of the problem. Available at: `https://beep.metid.polimi.it/documents/3343933/d5865f65-6d37-484e-b0fa-04fcfe42216d`.

Legge quadro per il trasporto di persone mediante autoservizi pubblici non di linea number 21/1992; 15th January 1992. This is the current Italian law that regulates the taxi service. Available (only in Italian) at: `http://www.normattiva.it/uri-res/N2Ls?urn:nir:stato:legge:1992-01-15;21`.

Codice in materia di protezione dei dati personali number 196/2003; 30th June 2003. This is the current Italian law that law that regulates privacy issues in Italy. Available (only in Italian) at: `http://www.normattiva.it/uri-res/N2Ls?urn:nir:stato:decreto.legislativo:2003-06-30;196`.

1.5 Overview of the document

This document develops as follows. In section number 2 we give a general description of the factors that affect both myTaxiService system and its requirements. In particular, this section focuses on the functions which are provided by the system, the constraints we have to impose, the assumptions we make. Requirements are thoroughly analysed in section 4, which is the most technical one. Formal and informal graphical representations are given as a support. Eventually, section 5 contains some supporting information and material, including tables of contents and an index.

2
Overall description

2.1 Product perspective

Before analysing in detail all the aspects of the system we are proposing, for the sake of clarity, it is useful to state its structure, and give mnemonic names to the different components.

The central system, to which the other components refer to, is called myTaxiService; this is to be accessed by the office of the municipality in charge of the service, who will administer it (for example, by adding and removing taxi drivers). Users can use a mobile application, myTaxiApp, or the website, myTaxiWeb, to communicate with the system. As of the taxi drivers, their activity is supported by another mobile application, named myTaxiAssist.

When it is clear from the context, however, we will refer to the whole ecosystem as myTaxiService too, by extension of the name of the main system.

In the following subsections we will always consider each component of the system separately.

2.1.1 System interfaces

myTaxiApp since it is a mobile application, it needs a smartphone or tablet to work on; a connection to the Internet is also needed. To reach the widest portion of population, we need the application to be available for AppleâĂŹs iOS, Android and Windows Phone devices.

myTaxiWeb given that a browser is always present in any operating system, the only requirement is that a connection to the Internet is available.

myTaxiAssist considering that there are some technological devices (now our interest is focused on the GPS device) on board of the taxi cab, the application should integrate with them; with this in mind, we decide to provide every taxi driver with an Android smartphone, which is the best platform to provide integrated services; as before, an Internet connection is needed.

myTaxiService we believe that a good trade-off between costs and performances is the installation of a server farm based on cloning

(Reliable Array of Cloned Services, or RACS. Obviously, for the whole ecosystem to work correctly, an Internet connection is needed.

2.1.2 *User interfaces*

myTaxiApp since the application is designated for mobile handsets, limited screen size and resolution will be a major design consideration. Moreover, navigability, effectiveness and ease of use are fundamental, so every function shall be fulfilled by the user within no more than (two) screens, and every screen shall contain a link to a help page as well.

myTaxiWeb here we have less constraints on resolution than in myTaxiApp, since the website is to be accessible only from personal computers. However, we need it to be lightweight, so that on a 7 Mbit/s connection (which we assume to be available everywhere in Milan) the page can fully load in no more than 2 seconds. Moreover, the requirements regarding the ease of use still hold, so also here every function shall be fulfilled within no more than (two) screens, and every screen shall contain a link to a help page.

myTaxiAssist this application is specifically designated for smartphones, so some resolution constraints hold. Furthermore, taxi drivers need to be extremely rapid while using it: after two hours of training, every taxi driver can exploit any function of the application in less than a minute. This is also possible thanks to the use of big (no more than four in a screen), self-explanatory icons. Every function shall be fulfilled within no more than (two) screens, and every screen shall contain a link to a help page.

myTaxiService as was stated before, some employees of the municipality shall be able to perform some administrative functions on the system, such as the insertion of new taxi drivers; this can be done through a webpage. As a consequence, no resolution constraint holds.

2.1.3 *Hardware interfaces*

Location services are of central importance for the operation of the system. However, the GPS coordinates are always provided to the applications or the website by the operating system. In particular:

myTaxiApp receives them from the APIs of the specific operating system;

myTaxiWeb gets them through the localization services provided by every reasonably modern web browser;

myTaxiAssist is provided with the localization by the GPS sensor placed in the taxi cab.

Nevertheless, since we cannot assume that a GPS sensor (or other technological solutions) are available and reliable in customersâĂŹ devices, the possibility to manually specify the address is given. On the contrary, it is reasonable to take for granted the accuracy of the GPS device on board of taxi cabs.

The same holds for Internet access: it is the operating system which provides a connection both to the applications and to the web browser.

2.1.4 Software interfaces

Due to the great variety of operating systems and browsers, each of which bearing a potentially different configuration, the website myTaxiWeb shall not require Java nor Adobe Flash plugins; it will be developed using HTML 5 technology, so a suitable web browser is needed (as of Microsoft Internet Explorer, al least version 7 is needed; on the other browsers, compatibility is taken for granted).

The system itself has to communicate with a database, which contains both the personal details of the taxi drivers, along with the information about taxi cabs, and the record of all the reservations (pending, accomplished, cancelled). A relational database is the most suitable solution.

On the other hand, the two applications, from this point of view, are rather self-contained, so no particular software interface is needed.

2.1.5 Communications interfaces

The myTaxiService server is HTTP-based and it contains also an SMTP component to send emails. Applications, obviously, send to and receive data from the system over the same HTTP protocol.

Bluetooth is required by myTaxiAssist application, since this is the integration technology between taxi driversâĂŹ handsets and the devices on board of their taxy cabs. If the connection is lost, arguably because the taxi driver has left the cab, the system is notified, and the driver is considered âĂIJoff dutyâĂİ until the reconnection.

2.1.6 Memory constraints

Since this system is not aimed to data analysis, we have no need of great memory availability. As a consequence, this is not to be considered a major constraint.

2.1.7 Operations

The data processing features of the system are limited, since they are not needed. Every day at 3AM, however, backup operations are performed.

2.2 Product functions

Before moving on with our analysis, it is appropriate to provide a summary of the major functions that the software will perform.

Passengers can request a taxi either through a web application (myTaxiWeb) or a mobile app (myTaxiApp). As soon as the request is received by the system (myTaxiService), it is immediately forwarded to the nearest waiting taxi driver. If he accepts, the system answers to the request by providing the passenger with the code of the incoming taxi and the waiting time. If the request is rejected, or ignored, the system looks for another taxi driver. In particular conditions, a user can also reserve a taxi for a later ride, by specifying its time, origin and destination.

Taxi drivers use a mobile application (myTaxiAssist) to inform the system about their availability and to confirm that they are going to take care of a certain call.

2.3 User characteristics

There are three types of users that interact with the system: users of the mobile application or website, taxi drivers and administrators.

Since no educational level can be assumed for the users of myTaxiApp and myTaxyWeb, we need both to be as easy to use as possible, which means that every screen shall have a precise help page. Obviously a very basic web and technological knowledge is needed to benefit from the service.

The taxi drivers will follow a mandatory two-hour course, to gain the necessary expertise to use the application myTaxyAssist.

The administrators' functionalities are limited, but they also need a two-hour course to gain the necessary knowledge.

2.4 Constraints

The current Italian law that regulates the taxi service (REFERENCE) allows all the modifications to the service that we are proposing. However, since we process personal data, we are subject to the Italian law concerning the protection of personal data (REFERENCE).

2.5 Assumptions and dependencies

We have to make some assumptions, since the high level description of the project given by the city council is ambiguous and incomplete:

- the number and distribution of taxi cabs in service are always sufficient to serve any request within 30 minutes;
- the reservation for a delayed ride can be cancelled before its confirmation (that is, before the 10 minutes limit);

- to cross an area, a taxi needs about 10 minutes;
- if the user reserves a taxi, he means to take it;
- a taxi driver who accepts a request, cannot refuse to accept the client;
- the origin of the ride must be within the borders of the Metropolitan City of Milan;
- we have only taxi cabs for four people (plus the driver); if the group is more numerous, then a proper number of cabs is reserved (one every four people);
- a taxi can bring no more than four people; if the reservation is for a larger group, then an appropriate number of taxi cabs (that is N/4 +1) is allocated;
- a pending request is automatically forwarded to the next taxi driver after (2) minutes;

2.6 Apportioning of requirements

The system, as it is outlined in this document, can be expanded in many ways. The city council explicitly mentions a taxi sharing functionality, which means that a user can share a taxi ride with others if possible, thus splitting the cost of the ride too. We also suggest that the system, through myTaxiAssist application, can record payments. A relatively easy integration may be with PayPal payment system, which allows fast and seamless online money transactions without disclosing bank account details. PayPal registered users can be charged directly and can pay by using their own handset.

myTaxiApp

myTaxiWeb

myTaxiAssist

myTaxiService

3
APPENDIX

```
module myTaxiService


-----  SIGNATURES  ------------------------------------------------------------

abstract sig Person {
--  name: one String,
--  surname: one String,
--  phoneNumber: one String,
}

sig Customer extends Person {}

sig Operator extends Person {}

sig RegisteredCustomer extends Customer {
    email: one String,
    addressRecord: set Address,
}

sig TaxiDriver extends Person {
    driverLicenceNumber: one String,
}

sig Taxi {
--  taxiID: one String,
--  licencePlate : one String,
    onDuty: one Boolean,
    gpsLocation: one GPS,
    driver: one TaxiDriver,
}

sig Boolean {}

sig GPS{}

sig Address {
--  town: one String,
--  street: one String,
--  number: one String,
    isIn: one Area,
}
```

```
sig Date {}

sig TaxiRequest {
    date: one Date,
    origin: one Address,
    destination: one Address,
    allocatedTaxi: some Taxi, -- there may be more than 4 passengers
    customer: one Customer,
    numOfPassengers: one Int
} {
    origin != destination
}

sig TaxiReservation {
    reservationDate: one Date,
    requestDate: one Date,
    origin: one Address,
    destination: one Address,
    taxiRequest: one TaxiRequest,
    customer: one RegisteredCustomer,
    numOfPassengers: one Int
} {
    requestDate = taxiRequest.date
    origin = taxiRequest.origin
    destination = taxiRequest.destination
    customer = taxiRequest.customer
    reservationDate != requestDate  -- In Java, Date class contains time as well.
    numOfPassengers = taxiRequest.numOfPassengers
}

sig Intersection {}

sig Area {
    adjAreas: some Area,
    taxiQueue: set Taxi,
    borders: some Intersection,
    contains: some Address,
}

sig FaultReport {
    operator: one Operator,
    taxiInvolved: one Taxi,
}



-----  FACTS  ----------------------------------------------------------------
--Facts are constraints that restrict the model. RESTR on PROPERTIES
--Facts are part of our specification of our system.
--Any configuration that is an instance of the specification has to satisfy all the facts.


--A taxi driver is registered only once
fact singleTaxiDriverProprieties {
    no disj td1, td2: TaxiDriver |
        ((td1.driverLicenceNumber & td2.driverLicenceNumber) = none)
```

```
}

--A user can register only once
fact registerOnlyOnce {
    no disj rc1, rc2: RegisteredCustomer |
        ((rc1.email & rc2.email) = none)
}


-- All TaxiDrivers have one and only one taxi.
fact oneTaxiPropriety {
    no disj t1, t2: Taxi | t1.driver = t2.driver
}

-- Adjacency is a symmetric, non reflexive property.
fact adjacentProprieties {
    all a, b: Area | (a in b.adjAreas) iff (b in a.adjAreas)  -- symmetry
    all a: Area | a not in a.adjAreas  -- non reflexivity
}

-- A taxi belongs to one and only one taxi queue.
fact noTaxiUbiquity {
    all disj a1, a2: Area | (a1.taxiQueue & a2.taxiQueue) = none
}

-- An address is located in a specific area.
fact noAddressesUbiquity {
    all disj a1, a2: Area | (a1.contains & a2.contains) = none
}

-- If an address is in an area, then that area contains that address.
fact inverseFunction1 {
     isIn = ~ contains
}

-- A customer can have only one active request.
fact oneActiveRequest {
    all disj r1, r2: TaxiRequest |
        r1.date = r2.date implies ((r1.customer & r2.customer) = none)
}

-- A taxi is assigned at most one request at the same time.
fact requestAssignation {
    all disj tr1, tr2: TaxiRequest |
        tr1.allocatedTaxi = tr2.allocatedTaxi implies ((tr1.date & tr2.date) = none)
}

-- A request can correspond to at most one reservation.
fact oneReservationOneRequest {
    no disj t1, t2: TaxiReservation | (t1.taxiRequest & t2.taxiRequest) != none
}

--To each request there are a sufficient number of taxi and no more
fact numberOfTaxi {
    no r: TaxiRequest |
        #allocatedTaxi = rem[r.numOfPassengers, 4]
}
```

```
-----  ASSERTIONS  ------------------------------------------------------------
--Constraints that were intended to follow from facts of the model
--Assertions state the properties that we expect to hold

-- No multiple allocations of the same taxi to the same request.
assert noMultipleAllocations {
    all disj t1, t2: TaxiRequest |
        (t1.date = t2.date) implies ((t1.allocatedTaxi & t2.allocatedTaxi) = none)
}

check noMultipleAllocations

--A customer can have only one reservation at the same moment
assert oneActiveReservation {
    all disj r1, r2: TaxiReservation |
        (r1.requestDate = r2.requestDate) implies ((r1.customer & r2.customer) = none)
}

check oneActiveReservation

/*

-----  PREDICATES  ------------------------------------------------------------
--A predicate is a named constraint with zero or more arguments
--When it is used, the arguments are replaced with the instantiating expressions

-- Show a world where a request is due to a reservation.
pred showReservationRequest() {
    all res: TaxiReservation | some req: TaxiRequest | res.taxiRequest = req
}
run showReservationRequest for 2

*/


pred show() {
    #Customer > 0
    #RegisteredCustomer = 0
--  #TaxiDriver  > 0
--  #Taxi  > 0
--  #Boolean  = 0
--  #GPS > 0
--  #Address  > 0
--  #Date  > 0
    #TaxiRequest  > 0
--  #TaxiReservation  =0
--  #Intersection  > 0
--  #Area > 0

    #Operator  = 0
    #FaultReport  = 0
}
run show
```