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

Customer the person who makes use of the reservation service through one of the available methods (mobile application, web-

site, 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

qwertyuiop

```
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
```