Software Requirements Specifications

Alberto Pirovano Alessandro Vetere

October 26, 2015

Software Requirements Specifications of *myTaxiService* project.

Contents

1 Introduction **3**
1.1 Purpose . . . 3
1.2 Scope . . . 3
1.3 Definitions, acronyms, and abbreviations . . . 3
1.4 Software Overview . . . 3
1.5 References . . . 3

2 Overall Description **4**
2.1 Product Perspective . . . 4
2.2 Product Features . . . 4
2.3 User Classes and Characteristics . . . 4
2.4 Operating Environment . . . 4
2.5 Design and Implementation Constraints . . . 5
2.6 Assumptions and Dependencies . . . 5

3 Specific requirements **6**
3.1 External interface requirements . . . 6
3.1.1 User interfaces . . . 6
3.1.2 Hardware interfaces . . . 6
3.1.3 Software interfaces . . . 6
3.1.4 Communications interfaces . . . 6
3.2 TODO: System Features . . . 6
3.2.1 System Feature 1 . . . 6
3.2.1.1 Description and Priority . . . 6
3.2.1.2 Stimulus/Response Sequences . . . 6
3.2.1.3 Functional Requirements . . . 7
3.2.1.4 Scenarios . . . 7
3.2.1.5 UML Diagrams . . . 7
3.2.1.6 Alloy Models . . . 7
3.3 Performance requirements . . . 7
3.4 Design constraints . . . 7
3.5 Software system attributes . . . 7
3.6 Other requirements . . . 7

1 Introduction

TODO: Introduction to *myTaxiService* SRS document.

1.1 Purpose

The purpose of this document is to serve as a guide to software designers, developers and testers who are responsible for the engineering of the *myTaxiService* project. It should give the engineers all of the information necessary to design, develop and test the software. In addition, it is an important tool for *myTaxiService* stakeholders to check whether the product matches their initial demands, and to serve as a further contractual basis.

1.2 Scope

This document provides an overall description of the system, showing its functional and non-functional requirements, constraints and external interfaces. It consists of natural language descriptions, UML diagrams and Alloy models.

1.3 Definitions, acronyms, and abbreviations

TODO

1.4 Software Overview

The aim of *myTaxiService* is to improve taxi service usage and management in a large city. The project will simplify the access of passengers to the service and optimize the management of taxi queues. This software will lead to several benefits for taxi drivers, passengers and the government of the city. The software will simplify the access to the service therefore increasing the number of users, giving benefit to either the drivers and the government. Also, it will optimize the management of taxi queues shortening the passengers waiting time and distributing equally users to taxi drivers.

1.5 References

This section contains references to external sources that have been taken into account while writing this document.

- Specification Document: Software Engineering 2 Project, AA 2015-2016, Assignment 1 and 2
- IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications
- Applied Software Project Management Andrew Stellman & Jennifer Greene

2 Overall Description

This section contains an overall description of *myTaxiService* software.

2.1 Product Perspective

The software is a completely new product, not based on other previous ones; in addition it relies on GPS cars signal to handle the service. Several programmatic API are provided in order to permit further improvements and expansions. In this way services like taxi sharing could be built on the top of this API.

2.2 Product Features

MyTaxyService provides functions for both passengers and taxi drivers. A passenger can make a taxi request to the system, that has to answers informing the passenger about the code of the incoming taxi and the waiting time. On the other hand, taxi drivers should inform the system about their availability and confirm that they are going to take care of a certain call. The system guarantees a fair management of taxi queues. When a request arrives from a certain zone, the system forwards it to the first taxi queuing in that zone. If the taxi confirms, then the system will send a confirmation to the passenger. If not, then the system will forward the request to the second in the queue and will, at the same time, move the first taxi in the last position in the queue.

2.3 User Classes and Characteristics

This software is dedicated both to passengers and taxi drivers. The main focus is on the passenger, which is a final user of the software, whereas the taxi driver is only an intermediate user. Therefore passengers interface should be more attractive and well-finished, while the taxi drivers interface should more usable. Passengers can request a taxi either through a web application or a mobile app. Taxi drivers use a mobile application to inform the system about their availability and to confirm that they are going to take care of a certain call. Under a certain point of view, we could consider even an additional user class which is the one of the developers that will be using the project API to develop further services based on the provided ones. This last class needs an extensive documentation about the programmatic interfaces they will integrate in their development, but no proper user interface is provided for them.

2.4 Operating Environment

This software will operate on different machines in different environment. It TODO: Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.

2.5 Design and Implementation Constraints

TODO: Describe any items or issues that will limit the options available to the developers. These might include: corporate or regulatory policies; hardware limitations (timing requirements, memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; parallel operations; language requirements; communications protocols; security considerations; design conventions or programming standards (for example, if the customers organization will be responsible for maintaining the delivered software).

2.6 Assumptions and Dependencies

TODO: List any assumed factors (as opposed to known facts) that could affect the requirements stated in the SRS. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project, unless they are already documented elsewhere (for example, in the vision and scope document or the project plan).

3 Specific requirements

TODO

3.1 External interface requirements

TODO

3.1.1 User interfaces

TODO

3.1.2 Hardware interfaces

TODO

3.1.3 Software interfaces

TODO

3.1.4 Communications interfaces

TODO

3.2 TODO: System Features

TODO: This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.

3.2.1 System Feature 1

TODO

3.2.1.1 Description and Priority TODO: Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).

3.2.1.2 Stimulus/Response Sequences TODO: List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.

3.2.1.3 Functional Requirements

TODO: Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use TBD as a placeholder to indicate when necessary information is not yet available. TODO: Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.

3.2.1.4 Scenarios

3.2.1.5 UML Diagrams

3.2.1.6 Alloy Models

- REQ-1:
- REQ-2:

3.3 Performance requirements

TODO

3.4 Design constraints

TODO

3.5 Software system attributes

TODO

3.6 Other requirements

TODO