Software Requirements Specification for

[1/23/2017]
[Version 0.1]

Prepared by:
Francisco Barros, Karem Ballesteros, Carlo García.

Table of Contents

Revision History

Version	Date	Name	Description
0.1	1/23/2017	T&W	Basic elements from IBM Watson's sdk and IBM's Analytics. Translator French, Spanish and Italian
0.2	1/24/2017	T&W	Speech to Text API Bluemix

1

Introduction

1.1 Overview

This document contains requirements of Medical Database system that is being developed for Puerta de Hierro Medical Center. The purpose of the document is to explain to public how is the new database will work, so that patients can use this database as a tool to transport their medical record.

This document doesn't contains any financial information nor medical programs provided by the system. Those are located in separate documents of the project.

The Medical Database system is a web-cloud to transport their medical record to any country they are visiting to be aware of any emergency. It provides a backup of their medical history located in a cloud, so that it could be read at any place in the world.

1.2 Goals and Objectives

Making the goals and objectives of the product explicit guides developers and give direction to the project. During the development of even a small system developers make hundreds of tiny decisions (consciously and unconsciously) not specifically addressed by the formal requirements. Knowing the goals and objectives will help them make the best decisions for the product.

Goals and objectives also help keep the project on track. Without a clear set of goals and objectives the scope of the project can grow or shift as new information arrives. Shifting the focus of the project isn't bad as long as it is done open and explicitly with everyone in agreement.

Example:
The three main goals of the innovative cloud medical services are:

1. Provide fast and efficient medical services.
2. Use feedback from users in order to make the database adapt to the user and make medical technology much easier and faster.
3. The system shouldn't require any extra effort on the part of the client.

1.3 Scope

Example:
The innovative cloud medical system will solicit feedback from readers including speech comments. Aggregate feedback from readers may be offered directly to other readers (i.e. articles might be rated), but unedited comments from readers will not automatically be made available to other readers. The reason for this is the quality of unedited comments is hard to control.

1.4 Definitions

This section defines potentially unfamiliar or ambiguous words, acronyms and abbreviations. If terms such as “shall”, “should” and “may” are used to indicate importance the meaning of these terms should be defined here.

Example:

Use case – describes a goal-oriented interaction between the system and an actor. A use case may define several variants called scenarios that result in different paths through the use case and usually different outcomes.

Scenario – one path through a use case

Actor – user or other software system that receives value from a use case.

Role – category of users that share similar characteristics.

Product – what is being described here; the software system specified in this document.

Project – activities that will lead to the production of the product described here. Project issues are described in a separate project plan.

Shall – adverb used to indicate importance; indicates the requirement is mandatory. “Must” and “will” are synonyms for “shall”.

Should – adverb used to indicate importance; indicates the requirement is desired but not mandatory.

May – adverb used to indicate an option. For example, “The system may be taken offline for up to one hour every evening for maintenance.” Not used to express a requirement, but rather to specifically allow an option.

Controls – the individual elements of a user interface such as buttons and check-boxes.

1.5 Document Conventions

Portions of this document that are incomplete will be marked with TBD. Each TBD item will have an owner and estimated date for resolving the issue.

1.6 Assumptions

In this section list any assumptions on which the requirements, as they are described here, depend. Assumptions are conditions, usually outside the control of the performing organization, that are taken for granted.

Be careful to only document assumptions that are outside the control of the performing organization. For example, the following is not a valid assumption for the requirements document: “We assume that all requirements will be documented.” You might be *assuming* this but there is no point in documenting it as an assumption here because it is something that is primary within the scope and control of the performing organization. The purpose of this section is to document assumptions that are outside the control of the

performing organization—conditions that should be noted because someone will be taking responsibility for ensuring they hold.

A distinction can also be made between assumptions that pertain to the requirements and those that pertain to the project as a whole. The software project management plan is the most logical place to document assumptions that pertain to the project as a whole.

Example:
It is assumed that the client has an ODBC compliant database installed and this database will be accessible from the machine where the system will run.

It is assumed that the web hosting ISP will allow server-side scripts to access the file system.

2 General Design Constraints

2.1 Product Environment

The innovative cloud medical system will be a component of the existing Puerta De Hierro webpage https://www.cmpdh.net/ our web application will serve as an extension of a medical staff such as a doctor in charge of different clients.

2.2 User Characteristics

It's important for developers to have a complete and accurate image of the end users. Even when the requirements of the user interface are described in detail the developer will still make many tiny design decisions during design and implementation. Knowing the general characteristics of the end users will help the developer make better decisions.

If the specific users of the system are know list them here. More likely there will be user roles or categories of uses. For each group of users list their responsibilities, characterize their knowledge of the domain and describe their characteristics including technical sophistication, background and education.

Prioritize users if necessary. This is especially important when user characteristics conflict. For example, if the system must accommodate experience users as well as novices it is important to know which should be given priority in case it's not possible to accommodate both groups.

2.3 Mandated Constraints

Ideally requirements will be specified in terms of functionality needed and developers will have free rein to design and implement a solution. In practice there are constraints on the eventual design and implementation.

Constraints may be mandated technologies. For example, the client may specify that a

specific database management system, programming language, and/or operating system be used.

Constraints limit design and implementation options.

Constraints might be absolute, desirable or optional. If constraints aren't absolute the motivation for the constraint should also be given.

2.4 Potential System Evolution

The software implies an ever-growing database with cognitive functions; the system will study and therefore learn from the user, enabling itself to adapt to different user experiences.

3 Nonfunctional Requirements

Nonfunctional requirements are properties the system must have. Nonfunctional requirements tend to be orthogonal to functional requirements. For example a system may have the nonfunctional requirement that it be offline no more than 15 minutes at a time and not more than ½ hour each week. The realization of this requirements isn't limited to one spot in the code. This nonfunctional requirement crosscuts some or all functional requirements.

3.1 Usability Requirements

It's hard to image a software system that doesn't have usability as one of its highest nonfunctional quality requirements. It's not enough to just say that the system should be usable though. Usability requirements must be stated in a quantifiable and testable way.

One method of specifying usability requirements is to specify efficiency, effectiveness and satisfaction goals for specific scenarios of use (section 4) carried out by representative users (section 2.2). A simpler alternative is to design a survey to measure user satisfaction and get consensus on who will take the survey and what will be considered an acceptable aggregate score.

3.2 Operational Requirements

This section describes the general characteristics of the physical environment for the product.

Example:
The users' environment is noisy so the system shouldn't depend on the user hearing audible output.

3.3 Performance Requirements

The main performance characteristics are speed and capacity (memory). Performance requirements are usually stated as a function of the number of concurrent users. Use this

section to state the performance requirements of the system as a whole. If specific transactions have their own performance requirements state these requirements below along with the description of the feature.

Example:
System startup time should be less than 3 seconds. With 30 concurrent users no operation should take more than 5 seconds and 95% of the operations should take less than 2 seconds.

3.4 Security Requirements

Access to data and features may be limited to specific users. There may also be a requirement to keep an audit trail of system use. This section describes the security requirements including the levels and what needs to be protected.

3.5 Safety Requirements

The system may affect the safety of the larger environment. For example, there are limits on the intensity of stray electromagnetic radiation from electronic devices used in hospitals. Potential safety concerns should be investigated and documented in this section.

3.6 Legal Requirements

Some security and safety requirements may also be legal requirements. For example, federal law protects confidentiality of medical records.

Being confidentiality a core duty of medical practice, the personal information shared by patients will be kept private and therefore will only be available to the selected health care providers unless the patient allows otherwise.

Example:
Student social security numbers will not be visible to other students.

3.7 Other Quality Attributes

There are specific sections above for non-functional quality attributes such as security, performance, etc. In this section describe any other non-functional quality attributes such as portability, availability, etc.

3.8 Documentation and Training

An important part of the total system is the documentation and training that is provided with the system. This section should describe the types and quantity of documentation and training that will be provided with the product.

3.9 External Interface

External interfaces may be user interfaces or software interfaces.

3.9.1 User Interface

The requirements document shouldn't contain design or implementation details. The *logical* user interface should be described here. The description here shouldn't unnecessarily constrain design and implementation options.

The general personality of the interface should be described here. For example, should the interface convey a conservative, professional, authoritative or fun attitude? What is the look and feel? Style? User characteristics were given above in section 2.2 but it may be helpful to characterize the average user here as adult, teenager or child.

User interfaces may contain a mixture of media types. It may be helpful to describe the desired/permissible user interface in terms of media elements.

Should the interface be intuitive or will training be provided. If training is required what types of training will be provided (online help, separate user manual, formal classroom training).

Ease-of-use requirements should be stated in a way that can be verified. For example, "the product should be easy to use" isn't verifiable because it's impossible to define "easy" in a measurable way. Must better is "75% of users will be able to use 80% of the features within 20 seconds without prior training".

3.9.2 Software Interface

The operations defined by use case 4-7 below will also be available programmatically via XML over HTTP. The exact protocol is described the WSDL document at xyz.

4 System Features

The core requirements of the system are listed in this section. This template recommends organizing requirements by features rather than use cases. Features are system behaviors from the user's point-of-view. The requirements of a feature are described by one or more use cases plus any additional narration that is necessary

Features should be ranked and listed in priority order. Priority is determined by cost, risk and value. To prevent arguments over the exact values of these measures this template recommends using the values: high, medium and low. There should be a written understanding how the priorities listed here are used to determine what order features are delivered and what determines essential features, desirable features and optional features.

If you are following a time-boxed or incremental development process, any formula used to consider what order to implement features should consider not only the priority defined by cost, risk and value but the features impact on the design and architectures of the system. It may be beneficial to implement a low priority feature before a higher priority feature if it is an important architectural component.

4.1 Feature: <title>

Example Title: Feature: Speech to Text

Including the title in the heading causes it to show up in the table of contents above. This makes it easy to find the requirements associated with a feature.

Note, it may be better to organize requirements into separate groups or subcategories. For example, if there are different user interfaces for different user classes it may be helpful to organize the user interface specifications above and the features in this section into separate categories, one for each class of user. Categories may also be defined by mode or function.

4.1.1 Description and Priority

A short description provides an introduction helpful for understanding the detailed requirements in the next two sections.

Each feature should include estimates for: cost, risk and value. The developer estimates cost and risk, the user estimates the value of the feature.

Example:
Cost: medium
Risk: low
Value: high

4.1.2 Use Case: <title>

Example Title: Use Case: Store Medical Information

Each use case should have a short descriptive title that provides a convenient label for discussing the use case.

A use case captures the system requirements in the form of dialog between the system and actor. There may be more than one use case for each feature.

4.1.3 Additional Requirements

Include in this section additional functional and nonfunctional requirements not specified in the use case(s) above.

4.2 Feature: <title>

4.2.1 Description and Priority

Cost:
Risk:
Value:

4.2.2 Use Case: <title>

4.2.3 Additional Requirements