Jobs Management Application

Version 1.0



Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 18.03.2018 | 1.0 | First version | Galeata Bianca-Floriana |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

1. Introduction 4

2. Non-functional Requirements 4

2.1 Availability 4

2.2 Performance 5

2.3 Security 6

2.4 Testability 8

2.5 Usability 9

3. Design Constraints 10

4. References 10

Supplementary Specification

# Introduction

This document captures the system requirements that are not distinguished from the use-case model or which has not been detailed in others documents. Some of the requirements included:

* Legal and regulatory requirements, including application standards.
* Quality attributes of the system to be built, including usability, reliability, performance, and supportability requirements.
* Other requirements such as operating systems and environments, compatibility requirements, and design constraints

# Non-functional Requirements

## Availability

*Definition:*

This quality attribute refers to the degree in which this application is in the specified operable and commutable state at the start of a mission, when the mission is called for at an unknown time [1]. More exactly, this attribute counts the time the application is functional or in other words, online. This can be defined as a probability, after the formula:

*mean time to failure / (mean time to failure + mean repair time) [2].*

*Source of stimulus:*

* internal,
* external.

*Stimulus:*

* omission,
* crash,
* timing,
* response.

*Environment:*

* normal,
* degraded.

*Artifact:*

* system processors,
* communications channels,
* storage,
* processes.

*Response:*

* record event,
* notify user or other systems,
* disable sources according to rules,
* make unavailable for period.

*Response measure:*

* availability time,
* degraded mode interval,
* repair time.

*Tactics* [3]*:*

* *Ping/echo*

One component issues a ping and expects to receive back an echo, within a predefined time, from the component under scrutiny. Can be used for a group of components or by a client to test the performance of the server.

* *Heartbeat (dead man timer).*

In this case one component emits a heartbeat message periodically and another component listens for it. If the heartbeat fails, the originating component is assumed to have failed and a fault correction component is notified.

The Jobs Management Application will encounter a small rate of failures, the percentage of availability being at least 99%. This thing will assure that anyone will have access from anywhere and in case that a failure will arise, this will be remediated before a user can remark.

## Performance

*Definition:*

Performance indicates how responsive an application is when an action and an interval of time are given. It can be measured in terms of latency or throughput [4].

*Source of stimulus:*

* independent sources (user etc.),
* within system.

*Stimulus:*

* periodic events arrive,
* sporadic events arrive,
* stochastic events arrive.

*Environment:*

* normal,
* overload.

*Artifact:*

* the system.

*Response:*

* processes stimuli,
* changes level of service.

*Response measure:*

* latency,
* deadline,
* throughput,
* jitter,
* miss rate,
* data loss.

*Tactics* [3]*:*

There are many tactics that can be applied to increase the performance. These can be divided according to the subjects that they address:

1. *RESOURCE DEMAND*
   * + - * *reducing latency*

reducing the resources required for processing an event stream,

reducing the number of events processed,

reducing or managing demand involve controlling the use of resources.

1. *RESOURCE MANAGEMENT*
   * + - * Introduce concurrency,
         * Maintain multiple copies of either data or computations,
         * Increase available resources.
2. *RESOURCE ARBITRATION*
   * + - * First-in/First-out,
         * Fixed-priority scheduling,
         * Dynamic priority scheduling,
         * Static scheduling.

This application will ensure a performance at least the same as the user requests. Most of the presented tactics will be used in different area of the project, in order to obtain the best result*.*

## Security

*Definition:*

The capability of a system to prevent malicious or accidental actions outside of the designed usage, and to prevent disclosure or loss of information [4]. Security can be characterized as a system providing nonrepudiation, confidentiality, integrity, assurance, availability, and auditing

*Source of stimulus:*

Individual or system that is:

* correctly identified,
* identified incorrectly,
* unknown identity who can be internal/external, authorized/unauthorized with access to limited or vast resources.

*Stimulus:*

* display data,
* change/delete data,
* access service,
* reduce availability.

*Environment:*

* online or offline,
* connected or disconnected,
* firewalled or open.

*Artifact:*

* system services,
* Data.

*Response:*

* authenticate user,
* hide identity,
* block access/allow access,
* grant/withdraw permission,
* record access, store data in format, informs user or system
* unexplainable high demand, restricts availability.

*Response measure:*

* time, effort or resources to circumvent with probability,
* probability of detecting attack,
* probability of identifying responsible party, services available in DOS attack, extent of damage,
* percentage of legitimate access denied.

*Tactics* [3]*:*

* *Authenticate users*

Ensure that a system or a person is who say it is.

* *Authorize users*

Ensure that just an authorized user can access and modify data and services.

* *Maintain data confidentiality*

Confidentiality is usually achieved by applying encryption to data in order to protect them from unauthorized access.

* *Maintain integrity*

Data should be delivered as intended.

* *Limit exposure*

Limitation of services on each host in order to reduce any weaknesses.

* *Limit access*

If an unknown source tries to access the system, it’s access should be limited.

The security of this application will be assured from different points of view. The first thing is that each user has access to his own set of data and can obtain information that are private about another user. An account is needed in order to have access to data in the first place. This account has a password and a username in order to authorize the right user. Also limit exposure will be ensuring, in order to minimize the range of a possible attack.

## Testability

*Definition:*

This measure show how easy is to create test for a system and how quickly a failure can be demonstrated.

*Source of stimulus:*

* unit tester,
* integration tester,
* system tester,
* client acceptance tester,
* user.

*Stimulus:*

* analysis,
* architecture,
* design,
* class,
* integration,
* delivery.

*Environment:*

* development time,
* design time,
* compile time,
* deploy time.

*Artifact:*

* design,
* code,
* application.

*Response:*

* access state values,
* computes values,
* prepare test environment.

*Response measure:*

* percent of statements executed,
* probability of failure if a fault exists,
* time to perform tests,
* time to prepare test environment.

*Tactics* [3]*:*

* *Record/playback*

Saving the information that is crossing an interface and using it as the input or output for tests.

* *Separate interface from implementation*

Replace some components’ implementation in order to test the rest of the system*.*

* *Specialize access routes/interfaces*

Set variable values for a component through a test or independent from its normal execution.

* *Built-in monitors*

Components that supervise the system.

Different tactics will be used in order to obtain a testable application. Tests will be made for each level of logic, UI and database operations. The number of tests will increase with the development of the application.

## Usability

*Definition:*

This quality measure how intuitive, easy and simple is the user’s interaction with the application.

*Source of stimulus:*

* User/

*Stimulus:*

* learn system features,
* use system efficiently,
* minimize error impact,
* adapt system.

*Environment:*

* runtime or configuration time.

*Artifact:*

* system.

*Response:*

* learn features,
* use system efficiently,
* minimize error impact,
* comfort.

*Response measure:*

* task time,
* number of errors,
* number of problems solved,
* user satisfaction,
* gain of user knowledge,
* ratio of successful operations,
* amount of time/data lost.

*Tactics* [3]*:*

* *Separate the user interface from the rest of the application*

The UI may change more often that other components, so a separation between UI and the rest of the application will ease the later modifications.

* *Maintain a model of the task*

The system can provide assistance to the user.

* *Maintain a model of the user*

The system can configure the UI personalized for user time response.

* *Maintain a model of the system*

The system can inform the user about the duration of a task.

The application will have a friendly UI, simple and capable to learn a user how it works from the first utilization. Information about user’s error will be display, for example when incorrect data is introduced.

# Design Constraints

* Software language

The language used for this application will be JavaScript. The reason for this decision is that the language offer flexibility both for front-end and back-end. Also, there are various framework which can make the interaction with the language much easier.

For the database will be used a relational type, more exactly MySQL. The choose is made based on many years of SQL existence and the standards it offers.

* Developmental tool

The used tool will be WebStorm from JetBrains, because the easiness that a programmer can work with it, the integrated features that offer and the support for many languages.

* Architectural constrains

The used architecture will be client-server. This separation is desired because the components pats have different task that routinely perform. The client will hand the information entered by a user and the server will manage the storage of this.

# References

[1]Availability, from <https://en.wikipedia.org/wiki/Availability> [19.03.2018]

[2] Quality Attributes, from <https://mperry.github.io/2012/02/15/quality-attributes.html> [19.03.2018]

[3] Software Architecture in Practice, from <http://www.ece.ubc.ca/~matei/EECE417/BASS/> [20.03.2018]

[4] Chapter 16: Quality Attributes, from <https://msdn.microsoft.com/en-us/library/ee658094.aspx> [19.03.2018]

[5] Understanding Quality Attributes, from <https://www.cs.unb.ca/~wdu/cs6075w10/sa2.htm> [19.03.2018]