CV Tool

Supplementary Specification

Version <1.0>

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 03/04/19 | <1.0> | Supplementary specifications | Lăzăreanu Sabina |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

1. Introduction 4

2. Non-functional Requirements 4

2.1 Availability 4

2.2 Performance 4

2.3 Security 4

2.4 Testability 4

2.5 Usability 4

3. Design Constraints 4

Supplementary Specification

# Introduction

The introduction of the **Supplementary Specification** provides an overview of the entire document. The **Supplementary Specification** captures the system requirements that are not readily captured in the use cases of the use-case model. Such requirements include:

Quality attributes of the system to be built, including usability, reliability, performance, and supportability requirements.

# Non-functional Requirements

* *CV aesthetics,*
* *Source of stimulus: User*
* *Stimulus User pressing the “Create CV Button”*
* *Environment: being logged in and has correctly completed all the fields.*
* *Artifact: whole system*
* *Response: an email with generated CV is sent*
* *Response measure: the email is sent really quick*
* *Tactics:*

## Availability

The system should run 24/7 and everyone with internet access should be able to use it.

## Performance

In terms of performance, the system is supposed to run without any kind of lag, it should not freeze and execute everything as fast as possible.

## Security

When talking about security, the main features that have to be covered are the following:

* how well is the database built and how the passwords are stored(hashed or other methods);
* how many types of users exist and to what features they have access to;
* how correct is the stored data(the data should be validated before storing, in order to avoid errors regarding it’s length or type);

## Testability

The testing is going to be done using the mokito technique, which controls the application flow and creates objects that fit the requirements.

## Usability

Being a web app, it’s implied that it should work cross-platform, on different screen sizes and different devices. It should be accessible to anyone that needs it and it should have a user-friendly, intuitive interface.

# Design Constraints

The application is going to be developed using the Java language, using the Spring framework and Angular 6 for the user interface. Some of the design patterns that have to be included are the following: factory method, singleton, decorator and builder.