<FlashScore>

Supplementary Specification

Version <1.0>

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 03/04/2019 | 1.0 | Initial version | Dura Catalin |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

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:

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.]

This document will clarify the concerns about both functional and non-functional details of the application.

I will talk about issues such as system requirements to run the application properly, legal issues related to the application (to whom it addresses as well as the personal data to be processed for each client).

# Non-functional Requirements

*[Define system quality attributes in terms of scenarios according to the following template:*

* *Quality attribute definition*
* *Source of stimulus: the entity (human or another system) that generated the stimulus or event*
* *Stimulus: a condition that determines a reaction of the system*
* *Environment: the current condition of the system when the stimulus arrives*
* *Artifact: is a component that reacts to the stimulus. It may be the whole system or some pieces of it*
* *Response: the activity determined by the arrival of the stimulus*
* *Response measure: the quantifiable indication of the response*
* *Tactics*

*]*

Initial non-functional requirements:

The application will have 2 type of users:Client,Admin.

- In case of client use, the application will open and the user and password will be required to enter the registration. If login is required, the verification part of the system will be activated and confirmation of the correctness of the data will be requested and in case of confirmation the options for the client will be displayed. Each option will activate a possible specific component of the code to perform the operation. If registration is desired, verify the uniqueness of the data and then create the account.

- In the case of admin usage, the verification part of the system will be activated and confirmation of the correctness of the data will be requested and in case of confirmation the options for admin will be displayed. Each option will activate a possible specific component of the code to perform the operation.

## Availability

The app wants to be turned on always, being available to anyone with Internet access and a powerful device for accessing a standard web page.

## Performance

The application is intended to be simple and fast, running strictly as a scoring page and not as a live betting page or any real-time action other than logging in / registration.

## Security

To ensure security, a complex password and the uniqueness of each username will be required.

## Testability

The application is subject to a continuous test, being always available to customers, and is therefore under constant critical observation.

## Usability

In terms of usage, the application can be used by any sports enthusiast or company that needs real-time results, as well as a recent ranking or any other relevant information.

# Design Constraints

[This section needs to indicate any design constraints on the system being built. Design constraints represent design decisions that have been mandated and must be adhered to. Examples include software languages, software process requirements, prescribed use of developmental tools, architectural and design constraints, purchased components, class libraries, and so on.]

The application will be implemented using Java, with an SQL database in the back. The application will use the principles of DAO (Data Access Object), Multilayer, and other principles that we will talk about later will be used.