Library Management System

Version 1.0

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 20/03/2018 | 1.1 | First attempt to write the Supplementary Specification document | Boros Hanniel |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

1. Introduction 4

2. Non-functional Requirements 4

2.1 Availability 4

2.2 Performance 4

2.3 Reusability 5

2.4 Testability 5

2.5 Usability 5

3. Design Constraints 6

# Introduction

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). Also, here are specified other requirements such as operating systems and environments, compatibility requirements and design constraints.

The Supplementary Specifications and the use-case model together capture a complete set of requirements on the system.

This specification defines the non-functional requirements of the system; such as availability, performance, security, testability and usability as well as functional requirements that are common across a number of use cases. (The functional requirements are defined in the Use Case Model.)

# Non-functional Requirements

Defining 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

## Availability

**Definition**

Availability defines the proportion of time that the system is functional and working. It can be measured as a percentage of the total system downtime over a predefined period. Availability will be affected by system errors, infrastructure problems, malicious attacks, and system load.

**Source of stimulus:** User

**Stimulus:** Connection to database

**Environment:** system is in the developing process

**Artifact:** Data Access Layer

**Response:** Application is functional and working

**Response measure:** amount of time the system is functional and working

**Tactics:** good system architecture, design tactics and security

## Performance

**Definition**

Performance is an indication of the responsiveness of a system to execute any action within a given time interval. It can be measured in terms of latency or throughput. Latency is the time taken to respond to any event. Throughput is the number of events that take place within a given amount of time.

**Source of stimulus:** User

**Stimulus:** read/update the database

**Environment:** developed system

**Artifact:** All Layers

**Response:** Database is read/updated

**Response measure:** time interval to execute any action (Latency/Throughput)

**Tactics:** performance techniques

## Reusability

**Definition**

Reusability is the probability that a component will be used in other components or scenarios to add new functionality with little or no change. Reusability minimizes the duplication of components and the implementation time. Identifying the common attributes between various components is the first step in building small reusable components for use in a larger system.

**Source of stimulus:** Developer

**Stimulus:** use of different code or components

**Environment:** project in development process

**Artifact:** almost all layers

**Response:** the use of multiple similar methods to implement tasks that have only slight variation

**Response measure:** reused components work properly

**Tactics:** focus on reusability when designing components

## Testability

**Definition**

Testability is a measure of how well system or components allow you to create test criteria and execute tests to determine if the criteria are met. Testability allows faults in a system to be isolated in a timely and effective manner.

**Source of stimulus:** Developer

**Stimulus:** run test modules

**Environment:** project in development process

**Artifact:** imply mostly the Business Logic Layer

**Response:** the tests are run

**Response measure:** if the test are successfull

**Tactics:** tets driven development

## Usability

**Definition**

Usability defines how well the application meets the requirements of the user and consumer by being intuitive, easy to localize and globalize, providing good access for disabled users, and resulting in a good overall user experience.

**Source of stimulus:** User

**Stimulus:** use the aplication

**Environment:** developed project

**Artifact:** Presentation Layer

**Response:** user wants to do specific actions

**Response measure:** user successfully complete his actions

**Tactics:** Interface Segregation Principle

# Design Constraints

This section lists any design constraints on the system being built.

**Platform Requirements**

The Library Management System shall operate on any personal computer having Windows or Linux-based operating system. The client portion shall require less than 200MB disk space and around 1GB RAM.

**Java Compatibility**

The application shall be compatible with the Java 8 VM runtime environment.