<Supermarket Manager>

Supplementary Specification

Version <1.0>

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| <22/03/23> | <1.0> | <First version> | <Marin Andreea> |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

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 presents the system requirements that are not readily captured in the use cases of the use-case model.

# 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*

*]*

## Availability

The system will be available for the admin of the supermarket and for the employees (regular users).

## Performance

## Security

The project comes with high security, giving strict permissions for each user in terms of operations that they can perform. The regular user cannot perform the operation that the admin performs and the admin cannot perform the operations assigned to the regular user.

## Testability

The system will present tests for different use cases in order to assure a good performance.

## Usability

The system will be used by two users. The admin of the supermarket and the regular user (employee).

# 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 system should respect the Object Oriented Paradigms should be designed and implemented using Spring Boot. The architecture will be a layered one, the system being composed of several separate horizontal layers that function together as a single unit of software and in which we can make changes to a layer without concerning about whether the change will affect the other layers as well.