<Car Service Appointment Application>

Supplementary Specification

Version <1.0>

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| <dd/mmm/yy> | <x.x> | <details> | <name> |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

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.]

# 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

* Availability is expressed as the ratio of the available system time to the total working time
* Source of stimulus: existence of the system
* Stimulus: software update
* Environment: fully working system
* Artifact: the whole application
* Response: updating the application
* Response measure: the time it takes for the update to be completed
* Make updates efficient by updating smaller parts of the software application

## Performance

* Performance shows the response of the system when performing certain actions for a certain period of time
* Source of stimulus: user
* Stimulus: searching for an appointment
* Environment: the appointments are currently displayed in a certain order
* Artifact: the appointment module
* Response: refreshing the page by displaying only the items that match the search
* Response measure: the time in which the response is given
* Finding a good algorithm for performing the search

## Security

* Security is responsible for reducing the likelihood of malicious or accidental actions, theft or loss of information
* Source of stimulus: malicious user
* Stimulus: attempting to access private information without being authenticated
* Environment: the application is in normal functioning mode
* Artifact: the security/data validation layer
* Response: prevent user from having access to private information
* Response measure: the consistency of the blocking measures in response to different attacking methods
* Password encryption, encapsulation

## Testability

* Testability shows how the system performs according to certain criteria
* Source of stimulus: user
* Stimulus: entering wrong data formats
* Environment: the application is in normal functioning mode
* Artifact: unit tests
* Response: validate data and respond accordingly
* Response measure: the wideness of cases covered by the tests
* Creating tests for as many as possible use case scenarios

## Usability

* Usability is defined by the efficiency of the interaction between the user and the system
* Source of stimulus: User
* Stimulus: finding the register page
* Environment: the application is in normal functioning mode
* Artifact: UI
* Response: page searched is rendered
* Response measure: the amount of actions needed in order to reach the goal (i.e. number of clicks)
* Making every action accessible within at most 2 clicks

# 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 development tools chosen for this application are the Django framework (using Python) and JavaScript for building the UI. The advantages of Django are its scalability, security and speed. It offers the possibility of using many useful libraries which gives versatility. However, Django also provides an architectural constraint since it is based on the MVT pattern.