<Dog shelter>

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 application is easy to use, because with the help of the interface, it says to the users anything they need to know, or what to do in order for system to work smoothly. Anyone who wants to adopt or to give their dog for adoption can use the application. The users must be very honest when writing their description or the dogs description, because this can be crucial for the adoption process. If they are not, the staff can decide to delete the user.

[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

* The quality attributes are realized non-functional requirements used to evaluate the performance of a system
* Source of stimulus: the users and the staff will be humans, but in the case where a user wants to donate to the shelter, the application will have to decide if he has enough money in his account.
* Stimulus: with acceptance of the staff, the data base will be modify. For example, if a user wants to adopt a dog, he needs permission from the staff, and if it is given to him, then that dog will no longer appear in the shelter database.
* Environment: If the stimulus (the staff) will not do anything, then the system will remain the same as before.
* Artifact: If the staff changes something like adding another accessory, then the database of the system will be changed.
* Response: If some user ask request for something and at least one person from the staff is logged in, then the response will be quick enough.
* Response measure: If there are 2 users that wants to adopt one dog, then the staff will be looking at their descriptions and decide which one of them are more suitable for the dog

*[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 application is available for anyone who wants to be an owner of a dog, and has the necessary requirements and for anyone who can’t take care of their dogs, and have to find his dog a new home.

## Performance

The system’s performance is very high, because there are no other system that is created for this purpose, and because there are around 600 millions of dogs on the streets, it is necessary.

## Security

## Testability

## Usability

The usability is very high, because the database change immediately as the staff approve some requests of modify some aspects.

# 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.]