Your pet everywhere

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.]

The project’s name is “Your pet everywhere”. This application will be useful for all the pet owners. It is so har to keep in mind all the dates, so let’s let this application to manage this for you.

I have to design and implement and layer application which should be able to manage all the pet’s staff for a pet owner. There will be two types of users: the regular one and the administrator. The regular user will be able to make an account and after that he can login for make accounts for his pets. If he doesn’t have a pet yet but he wishes one, he will be able to make a profile. This profile will be verified by the administrator and the user will receive an animal breed. If he had a pet, he will access his pet profile and there he will can see his pet medical history and he will receive notification about his pet needs. Also, he can go shopping things for his pet.

All the date that will be stored in the database must be cheeked.

# 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*

*]*

In system engineering and requirements engineering, a non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. They are contrasted with functional requirements that define specific behavior or functions. The plan for implementing *functional* requirements is detailed in the system design. The plan for implementing *non-functional* requirements is detailed in the system architecture.

Source of stimulus: the regular user and the administrator will be humans

Stimulus: If a someone wants to make an account he has to receive the administrator permission. Also, if he doesn’t have a pet yet, he can make a profile for himself and the administrator will give him an animal breed.

Environment: Is the administrator doesn’t do something nothing happens, so the system will remain the same as before.

Artifact: If the user makes a note about his pet, or the administrator accept an user account there will be changes in database.

Response: the user has to wait for administrator permission to make an account

Response measure:

## Availability

This application is for everyone who needs help to manage his pets stuff.

## Performance

## Security

## Testability

## Usability

# 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.]

I will implement a Java application. The connection with the database will be make using MySQL Workbench. The development tool which I will use is Eclipse.