# Introduction

**Summary:**

This is a design document intent for readers to know more about the project so that they a perception about the software before using it.

## Document overview

This document contains the organization, the specifications, the conception, and verification tests of software development project.

It covers the following goals:

* Functionality of the software
* Logic and calculations used
* Testing approach used

## Scope

### Identification

This document applies to the software model developed using Spring Boot and thymeleaf template engine for the candidate-exercise project.

### Overview

The project was intended to add a way for a user to select either a single animal type or all animal types and rate the animals in the user-selected group using the business logic.

**1.2.3** **Abbreviations**

FSD-Functional Specification Document

CDN-Content Delivery Network

STS-Spring Test Suite

IDE-Integrated Development Environment

RTM- Requirements Traceability Matrix

# Project Management

The section describes the organizational structure of the candidate-exercise project.

## Team – human resources

The team is described in the diagram below.

## Responsibilities

The team of the project has the following responsibilities:

**Myself-Responsible to render the project as per FSD requirement**

## Customer -User involvement

From the beginning itself all the required details was provided and the requirements was enough for complation.

## Engineering environment

WorkStation-Ununtu(16.04 ,Xenial)

IDE-Eclipse with STS

Server-Tomcat

Port-8080

Version Control-Git Hub

## Other Resources

There is no particular resource needed for the project such as a calibrated measurement tool or a simulator.

# Specifications

1. Add another page similar to the one displayed by home.htm at /rate .

2. Add a way for a user to select either a single animal type or all animal types.

3. Rate the animals in the user-selected group using the above business logic.

4. Display the animals in the user-selected group, their characteristics, and their ratings on the /rate page, ordered by highest rated animal to lowest rated animal.

5. Calculate and display some statistics on the user-selected group: minimum rating, maximum rating, and average rating.

## States

Software works in three states:

* Starting: the software loads its components;
* In use: all the functionalities of the software are available to the users;
* Stopping: the software is being stopped.

## Performance

The core purpose of the project is to add a way for a user to select either a single animal type or all animal types and rate the animals in the user-selected group using the business logic.

In order to allocate the selection criteria a Dropdown menu was made and then allow to filter it with the selected item. I have made the following interfaces for filtration to display the intended result.

**public** List<Animal> findByDog();

**public** List<Animal> findByCat();

**public** List<Animal> findByRabbit();

**public** List<Animal> findBySnake();

In order to facilitate the custom sort a java comparator is used and below mentioned is the interface used for the same.

**public** List<Animal> getSortedanimalByRate();

In order to show the maximum and minimum ratings, I have used the below interfaces.

**public** List<Animal> maxRating();

**public** List<Animal> mimimumRating();

Below interfaces are used to display the maximun and minimum rating of each animal.

**public** List<Animal> dogMimimumRating();

**public** List<Animal>dogMaxRating();

**public** List<Animal> catMimimumRating();

**public** List<Animal>catMaxRating();

**public** List<Animal> rabMimimumRating();

**public** List<Animal>rabMaxRating();

**public** List<Animal> snakMimimumRating();

**public** List<Animal>snakMaxRating();

Below are the interfaces that is used to display the average of whole animal and that of each individual animal.

**public** **double** averageRating();

**public** **double** averageDogRating();

**public** **double** averageCatRating();

**public** **double** averageSnakeRating();

**public** **double** averageRabbitRating();

**Logic to Calculate the Rating**

Given offence, defense, health and items

Rating:

If defense greater than 10

(Offence\*2)-(items) +defense+ health

Else

(Offence\*2)-(items) +defense+ round(health/2)

## System environment

The software will support Mac, Linux and Windows OS. The software can be run on any IDE like eclipse or IntelliJ. In order to run the system require the following

1) Java

2)Maven

3)Jquery CDN or min.js file

4)STS

* 1. **Assumption**

A new variable called “rate” has created to enter the rating and there by facilitate the custom sort.

**1.12 Git Hub**

The copy the code is uploaded is git too and the reference for the same is “https://github.com/kevin8519/candidiate-exercise”

# Verification

## Test Plan

### Test environment

The environment used to test the software was linux (Ubuntu) and tomcat server on port 8080 which is the default one.

### Tests description

Manual Testing has done to make sure that the product has worked as per FSD requirement. Print Screen png files has been attached in order to show the expected features as mentioned in the FSD.

## Rationale for decision

After executing a test, the decision is defined according to the following rules:

* **OK:** The test sheet is set to "OK" state when all steps are in "OK" state. The real result is compliant to the expected result.
* **NOK:** The test sheet is set to "NOK" state when all steps of the test are set to "NOK" state or when the result of a step differs from the expected result.
* **NOT RUN:** Default state of a test sheet not yet executed.
* **NOT COMPLETED:** The test sheet is set to "Not Completed" state when at least one step of the test is set "Not Run" state.

## Results

Please refer RTM and Test case Documents

## Requirements Traceability Matrix(RTM)

This table gives the traceability between requirements and tests, and the method of test and the same is attached for your reference.

The verification methods of the requirements are defined below:

* Inspection (I): control or visual verification
* Analysis (A): verification based upon analytical evidences
* Demonstration (D): verification of operational characteristics, without quantitative measurement
* Test (T): verification of quantitative characteristics with quantitative measurement

For each requirement of the FSD, a verification method is defined. Method is abbreviated I, A, D or T.