02/14/2017

# Software Requirements Specification for Fishy Feeder System

Group 3 SRS


Authors

- **Adriana Rios**
- **David Rouse**
- **Michael Robertson**
- **Kayla Saythongkham**

## Table of contents


**I. Preface** 3

1.1 Introduction 3

*1.1.1 Purpose Statement* 4

1.2 Document Scope 4

1.3 Document conventions 4

*1.3.1 General conventions* 4

*1.3.2 Document levels style* 4

1.4 Intended audience 5

**II. Overall Description** 5

2.1 Product Perspective 5

*2.1.1 System Interfaces* 5

*2.1.2 User Interfaces* 5

*2.1.3 Hardware Interfaces* 5

*2.1.4 Software Interfaces* 6

*2.1.5 Communication Interfaces*

*2.1.6 Memory* 6

2.3 Production Function: Users 6

*2.2.1 Login* 6

*2.2.2 Schedule feedings* 6

*2.2.3 Feed on click* 6

**III. Functional Requirements** **6**

3.1 Use Case 1: Log in *6*

3.2. Use Case 2: Feed Fish 7

3.3 Use Case 3: Schedule a feeding 7

3.4 Use Case 4: View/Edit schedule and previous feedings 8

**IV. Non-Functional Requirements** 9

4.1 Requesting User Use 9

*4.1.1 Reliability* 9

*4.1.2 Robustness* 9

*4.1.3 Performance* 9

*4.1.4 Maintainability* 9

*4.1.5 Usability* 9

**V. Design & Implementation Constraints** **9**

5.1 Implementation Constraints: 10

*5.1.1 Connectivity* *10*

5.2 Design Constraints 10

*5.2.1 Database accuracy* *10*

5.3 Standards compliance 10

*5.3.1 User Authentication* *10*

# I. Preface

## 1.1 Introduction

Our project is an automated fish feeder targeted for users who are forgetful or have a busy schedule that prevents timely feeding of their pet fish. This is ideal for users who spend lots of time away from home for travel or have busy work or school schedules.

We will implement an app that allows users to set a time that may be repeated daily or weekly. Upon app interaction, the device will feed the fish. Users will also be reminded when the fish have been fed.

### 1.1.1 Purpose Statement

The purpose of this document is to detail the Fishy Feeder's required components and functioning.

## 1.2 Document Scope

The scope includes documentation on all aspects required of the Fishy Feeder. This document details what functions the Fishy Feeder will perform and what software and hardware will be required to perform required functions.

This document is intended as a comprehensive overview of the essential functionality and implementation of the Fishy Feeder. Herein is found descriptions of functional requirements for end users. An evaluation of the reliability, robustness,

performance, maintainability, and usability is provided in Section IV: Non-Functional Requirements.

## 1.3 Document conventions

### 1.3.1 General conventions

Visual consistency is maintained in this document through employment of the Arial font for the standard text and Times New Roman font for titles and subtitles. Technical terms are defined in the Definitions Subsection 1.5.

Title sections are numbered with roman numerals. Subsections are prefaced with the Indo-Arabic numbers 1, 2, … , 3 of the title section, followed by a period and the Indo-Arabic number of the subsection. Second level subsections are denoted similarly (e.g. “2.1.3” indicates title section 2, subsection 1, second level subsection 3).

Additionally, this document maintains compliance with IEEE standards.

### 1.3.2 Document levels style

- Title sections I through V are in font Times New Roman, size 16, bolded and underlined.
- Subsections are capitalized in font Times New Roman, size 12, bolded.
- Second level subsections are in font Times New Roman, size 11, bolded.
- General text is rendered in font Arial, size 11 with a 1.5 line spacing.

## 1.4 Intended audience

This SRS is intended primarily for developers, project managers (if any), and any personnel whose role involves direction of services staff or Fishy Feeder servers maintenance. Additionally, anyone making use of the Fishy Feeder may find this document beneficial in understanding the Fishy Feeder requirements and functioning.

# II. Overall Description

## 2.1 Product Perspective

We will implement an app that allows users to set a time that may be repeated daily or weekly. Upon app interaction, the device will feed the fish. Users will also be reminded when the fish have been fed.

### 2.1.1 System Interfaces

The system will be accessible via a website. The website will be addressable by directly typing in a URL.

### 2.1.2 User Interfaces

Any user can type in the website URL and access the webpage. However, the user needs a 1080p HD camera and high torque servo rigged up to drill feeder to implement services.

### 2.1.3 Hardware Interfaces

- Smart phones – can be used access the Fishy Feeder website.
- Computers – can be used to access the Fishy Feeder website.
- Server is housed in Raspberry Pi B
- 1080 HD camera
- Node.js

### 2.1.4 Software Interfaces

The website should work on all major web browsers (Google, Firefox, etc.).

### 2.1.5 Communication Interfaces

Communication between user and application is simply two-way. From the website, users will be reminded to feed fish via push notifications and of course through the fish camera, the feedback may be viewed. From the user, communication with the website includes scheduling feedings and button and toggle implementation to navigate the website.

### 2.1.6 Memory

The system should be able to keep track of a history of feedings to organize the fish's diet and help plan the next feeding.

### 2.2 Product Function: Developer

#### 2.2.1 Website

The website is the heart of this product. In order for users to be able to schedule feedings, the website must be reliable at all times for users and the feeder must respond.

### 2.3 Production Function: Users

#### 2.2.1 Login

Users should be able to login to the website in need of service.

#### 2.2.2 Schedule feedings

Users are allowed to set a repeating time to feed their fish.

#### 2.2.3 Feed on click

Users are able to feed fish on click.

# III. Functional Requirements

### 3.1 USE CASE #1: Log in

1. The user accesses the Fishy Feeder log in web page.
2. The system displays the log in web page with a grid for filling in a username and password along with a *log in* button. A *sign up* button is also displayed.
3. The user fills in the log in information and clicks *log in.*
4. The system verifies the information as valid and then logs in the user.
5. The system displays the user's main page with a livestream of the fish's bowl. *Feed fish, schedule feeding, view schedule and previous feedings*, and *log out* buttons are also displayed.
6. The user clicks *log out* button.
7. The system logs the user out and displays the log in web page.

**Variation #1: Sign up**

1.1. In step 3, the user clicks sign up button.

1.2. The system displays a grid for filling in a username, password, and any information needed to create an account along with a submit button.

1.3. The user fills in the sign up information and clicks submit.

1.4. Continue with step 4.

**Variation #2: Invalid login**

2.1. In step 4, the system verifies the information as invalid.

2.2. The system displays a message that the log in information was invalid.

2.3. Continue with step 2.

### 3.2 USE CASE #2: Feed fish

Before this use case can be initiated, the user has already accessed the user main page of the application.

1. The user selects *feed fish.*
2. The system presents a *feed* button and a *back* button.
3. The user clicks *feed* button.
4. The system dispenses food and displays message that the fish was fed.
5. User clicks *back* button.
6. The system displays the main page.

**Variation #1**

1.1. In step 3, the user clicks *back* button.

1.2. The system displays the main page.

### 3.3 USE CASE #3: Schedule a feeding

Before this use case can be initiated, the user has already accessed the user main page of the application.

1. The user selects *schedule feeding*.
2. The system presents a grid for filling in the date and time of feeding along with a *set feeding* and *back* button.
3. The user fills in the information and clicks *set feeding*.
4. The system verifies the information as valid and then saves it to the schedule.
5. The system displays a message of the date and time of the scheduled feeding.
6. The user clicks *back* button.
7. The system displays the main page.

**Variation #1**

1.1. In step 3, the user clicks *back* button.

1.2. The system displays the main page.

**Variation #2**

2.1. In step 3, the user enters invalid data for the date or time.

2.2. The system displays a message that the date or time input was invalid.

2.3. Continue with step 2.

### 3.4 USE CASE #4: View/edit schedule and previous feedings

Before this use case can be initiated, the user has already accessed the user main page of the application.

1. The user selects *view schedule and previous feedings*.
2. The system presents a calendar that is filled in with past feedings that say the date and time of the feeding as well as scheduled feedings data and time. The scheduled feedings have an *edit* button. A *back* button is also displayed.
3. The user clicks *back* button.
4. The system displays the main page.

**Variation #1: Edit a scheduled feeding**

1.1. In step 3, the user clicks *edit* button on a given scheduled feeding.

1.2. The system presents a grid for editing the date and time of feeding along with a *set feeding, delete,* and *back* button.

1.3. The user fills in the information and clicks *set feeding*.

1.4. The system verifies the information as valid and then saves it to the schedule.

1.5. The system displays a message of the date and time of the scheduled feeding.

1.6. The user clicks *back* button.

1.7. Continue with step 2.

**Variation #2: Delete a scheduled feeding**

2.1. In step 3, the user clicks *edit* button on a given scheduled feeding.

2.2. The system presents a grid for editing the date and time of feeding along with a *set feeding, delete,* and *back* button.

2.3. The user clicks *delete* button.

2.3. The system deletes the scheduled feeding from the schedule and displays a message saying that it was deleted.

2.4. The user clicks *back* button.

2.5. Continue with step 2.

# IV. Non-Functional Requirements

## 4.1 Fish owners / Users

### 4.1.1 Reliability

The user credentials database will be interacted with the Fishy Feeder database to keep the system secure. When someone attempts to log in, the system will then check with database that the username and password are in the system and match.

### 4.1.2 Robustness

The robustness of the system will be reliant on the developers to keep the system updated accordingly as well as the user or owner to take good care of the feeding device. The need for them to maintain their system is crucial for the functionalities to operate properly.

### 4.1.3 Performance

Performance of the system is reliant on a local host.

### 4.1.4 Maintainability

Some major priorities are maintaining the fish's live camera for feedback, making sure the feeder toggle functions properly by dropping enough food for the fish.

### 4.1.5 Usability

The web application provides a handful of buttons of navigate with. Pages and website design must be user friendly and considers human factors.

## 4.2 V. Design & Implementation Constraints

## 5.1 Implementation Constraints:

### 5.1.1 Connectivity

In order for the system to work as intended, it is assumed that users will have access to the internet via Wi-Fi or mobile data.

## 5.2 Design Constraints

### 5.2.1 Database Management

The Database must be able to work on the local servers, and maintainable by Fishy Feeder developers.

## 5.3 Standards compliance

### 5.3.1 User Authentication

Users must be authorized to log in to the system using their username and password information. Only fish owner users will have access to the visual model map functions of the website.