FISHMO
Semi-Automated Aquarium Monitoring System

Software Test Report
Revision 2

April 9, 2015

Group 5

Rhett Amin - 1060085
Brandon Da Silva - 1057434
Dan Esteves - 1152328
Michael Liut - 1132938

Prepared for:
Computer Science 4ZP6 Capstone Project
Instructor: Dr. Rong Zheng
Fall/Winter 2014-2015

Contents

1 Introduction

1.1 Purpose of the Document

This document provides an overview of the test results determined from the test plan as well as analysis and descriptions of these tests.

1.2 Scope of the Testing (Approach)

This testing report primarily focuses on the analysis of the program through tests specified from the test plan. This is accomplished by means of functional, or unit testing whereby an extensive examination of the robustness of our system occurs. To guarantee the validity and correctness of our program, dynamic testing must be performed. This dynamic testing will provide us with a high degree of confidence that the implementation may be correct.

Due to the limitation of time constraints associated with taking this course in terms of this project, the test cases outlined in this document only consider the correctness of the software to be the test factor. To test for efficiency would go beyond the constraints and scope of this report.

1.3 Definitions, Acronyms, and Abbreviations

Term	**Definition/Acronym**
FISHMO SAAMS	FISHMO Semi-Automated Aquarium Monitoring System
Pi	The Raspberry Pi is a credit-card sized computer that plugs into your TV and a keyboard. It is a capable little computer which can be used in electronics projects, and for many of the things that your desktop PC does, like spreadsheets, word-processing and games.
User	A person who interacts with the FISHMO system
NOOBS	NOOBS (New Out Of the Box Software) is an easy operating system install manager for the Raspberry Pi.
GUI	Graphical User Interface
Camera Board	The Camera Board is a small PCB that connects to the CSI-2 camera port on the Raspberry Pi using a short ribbon cable. It provides connectivity for a camera capable of capturing still images or video recordings.
3D Printer	A process for making a physical object from a three-dimensional digital model, typically by laying down many successive thin layers of a material.

1.4 Revision History

Revision #	**Revision Date**	**Description of Change**	**Author**
0	March 20, 2015	Initial Test Report	All members
1	April 9, 2015	Final Revision	All members
2	April 15, 2015	Feedback Changes	All members

1.5 Organization

This document will initially review the test results, followed by the test summary, which is broken down into the following categories: usability, performance, stress and robustness. These sections will entail the testing of components, integration, acceptance, security and the performance of the system. Furthermore, each of these major tests are summarized and the results are shared and explained in detail.

The structure of this document is closely following the example provided by Dr. Zheng in lecture. Further more, we are specifically referencing Avenue: Example ¿ Test Report.

2 Test Results

This section will explain the components of robustness testing in its entirety. Furthermore, both complex and trivial cases will be described at a high level; additional details will be provided for more complex modules.

This section merges "Expected Results" and "Accepted Results" into "Results" as we believed it to be more convenient method of conveying the overall result to the reader. It should be said that there were no anomalies during testing and all results expressed below are valid.

Mocks were used throughout the testing process to simulate the user and to mimic their ability to control their personalized FISHMO system. Furthermore, you can assume that all entries into the database were tested by utilising mock objects and verified via SQL queries in our database by our software engineers.

2.1 Component (Unit) Testing

Our components include: a temperature sensor, a water flow sensor, a water pump, a RGB LED strip, a camera, a network adapter and a water heater. All of which are rigorously stress tested prior to deployment. Furthermore, our code must meet certain requirements prior to execution. These requirements will ensure that the communication between the individual peripherals and the Raspberry Pi are correct and are properly receiving the anticipated data/response. This testing will occur during the initial stage of compilation (when the system is turned on), once it passes, the system will become active and ready for use.

2.1.1 Test 1: Accessing Camera Input

Initial State: The initial state of the camera is it not sending the data to the terminal window.
Input: Input a python test script to tell the PI to stream the camera feed to be displayed on the PIs local IP.
Results: The script switches the state and enables the camera feed to be streamed from the PIs local IP. The results could be seen by accessing the IP address.

2.1.2 Test 2: Temperature Input

Initial State: The initial state of the of the temperature sensor is not active, and not sending any data to the PI.
Input: Once the PI is prompted via python test script, it will begin to record temperature data from its surroundings, converting from Fahrenheit to Celsius.
Results: The temperature input data is outputted in Celsius and displayed on the terminal window.

2.1.3 Test 3: Waterflow Input

Initial State: The initial state of the water flow input sensor remains in a default position where it does not send any data to the PI.
Input: Once the PI is prompted via python test script, it will begin to record the current flow rate through the sensor. It is then averaged by the number of rotations per second.
Results: The waterflow input data is displayed on the terminal window in litres/per minute.

2.1.4 Test 4: Heating Output

Initial State: The initial state of the heater is off. It is not receiving any current.
Input: Initialize a Python script to switch the state of the heater into the on position, accepting current.
Results: The heater will begin to generate heat at a steady rate.

2.1.5 Test 5: Adding/Removing Fish to/from the Database

Initial State: The InTank table on the server has one or more fish in it (for adding/removing), or is empty (for adding).
Input: Adding: Run a SQL command to insert a fish object row into the table. Removing: Run a SQL command to delete a fish object row from the table
Results: (Shown in Figures 1-3)

2.1.6 Test 6: RGB LED Strip

Initial State: The LEDs are off, and not producing any output current.
Input: Initialize a python script to switch the state of the LEDs into the on position and to specify the RGB colour.
Results: The LEDs will produce light and the colour specified by the user.

```
mysql> select * from Fish;
+----+-----------+--------------------------+---------+----------+-------------------------+
| ID | Name      | Picture                  | TempLow | TempHigh | Compatible              |
+----+-----------+--------------------------+---------+----------+-------------------------+
|  1 | Goldfish  | /images/goldfish.jpg     |      20 |       22 | NULL                    |
|  2 | Guppy     | /images/guppy.jpg        |      24 |       28 | Betta,Fighter,Angelfish |
|  3 | Angelfish | /images/Angelfish.png    |      24 |       28 | Betta,Fighter,Guppy     |
|  4 | Betta     | /images/Beta_fish.jpg    |      23 |       27 | Angelfish,Guppy         |
|  5 | Fighter   | /images/Fighter_Fish.jpg |      23 |       27 | Angelfish,Guppy         |
|  6 | Puffer    | /images/blowfish.jpeg    |      24 |       26 | NULL                    |
+----+-----------+--------------------------+---------+----------+-------------------------+
6 rows in set (0.00 sec)
```

Figure 1: The existing fish database

```
mysql> INSERT INTO Fish(Name, Picture, TempLow, TempHigh, Compatible)
       VALUES ('Shark', 'images/shark.jpeg', 12, 15, NULL);
Query OK, 1 row affected (0.01 sec)

mysql> select * from Fish;
+----+-----------+--------------------------+---------+----------+-------------------------+
| ID | Name      | Picture                  | TempLow | TempHigh | Compatible              |
+----+-----------+--------------------------+---------+----------+-------------------------+
|  1 | Goldfish  | /images/goldfish.jpg     |      20 |       22 | NULL                    |
|  2 | Guppy     | /images/guppy.jpg        |      24 |       28 | Betta,Fighter,Angelfish |
|  3 | Angelfish | /images/Angelfish.png    |      24 |       28 | Betta,Fighter,Guppy     |
|  4 | Betta     | /images/Beta_fish.jpg    |      23 |       27 | Angelfish,Guppy         |
|  5 | Fighter   | /images/Fighter_Fish.jpg |      23 |       27 | Angelfish,Guppy         |
|  6 | Puffer    | /images/blowfish.jpeg    |      24 |       26 | NULL                    |
|  7 | Shark     | images/shark.jpeg        |      12 |       15 | NULL                    |
+----+-----------+--------------------------+---------+----------+-------------------------+
7 rows in set (0.00 sec)
```

Figure 2: Adding fish to the database

```
mysql> DELETE FROM Fish WHERE ID = 7;
Query OK, 1 row affected (0.00 sec)

mysql> select * from Fish;
+----+-----------+--------------------------+---------+----------+-------------------------+
| ID | Name      | Picture                  | TempLow | TempHigh | Compatible              |
+----+-----------+--------------------------+---------+----------+-------------------------+
|  1 | Goldfish  | /images/goldfish.jpg     |      20 |       22 | NULL                    |
|  2 | Guppy     | /images/guppy.jpg        |      24 |       28 | Betta,Fighter,Angelfish |
|  3 | Angelfish | /images/Angelfish.png    |      24 |       28 | Betta,Fighter,Guppy     |
|  4 | Betta     | /images/Beta_fish.jpg    |      23 |       27 | Angelfish,Guppy         |
|  5 | Fighter   | /images/Fighter_Fish.jpg |      23 |       27 | Angelfish,Guppy         |
|  6 | Puffer    | /images/blowfish.jpeg    |      24 |       26 | NULL                    |
+----+-----------+--------------------------+---------+----------+-------------------------+
6 rows in set (0.00 sec)
```

Figure 3: Deleting fish from the database

2.1.7 Test 7: Network Adapter

Initial State: The network adapter is in a roaming state; meaning it is searching for Wifi (assuming it is powered).
Input: Raspberry Pi networking Python script used to switch the state of the Network Adapter from roaming to connected.
Results: The Raspberry Pi will have local network and internet access.

2.2 Integration (System) Testing

This test method will ensure that software design objectives are met and ensures that the software, as a complete entity, complies with operational requirements. This will be tested manually whereby we use a mock user account to run the test cases below. In the test cases, we use every function and element of the system to ensure its accuracy and correctness. This test will pass if and only if the system requirements are satisfied. However, if the test fails, it does not meet one the required functionalities and therefore will flag an error notifying the user.

2.2.1 Test 1: Accessing Camera Input

Initial State: The initial state of the camera is it not sending the data over to the server to be pulled by the website.
Input: The user would click the play button on the display panel, which will tell the server to tell the PI to stream the camera feed. The website will take the IP address and display it under the display controls.
Results: The camera feed is streamed on the display control panel of the website.

2.2.2 Test 2: Temperature Input

Initial State: The initial state of the temperature sensor is not active, and not sending any data to the server.
Input: The user logins into the website, pinging the server to tell the PI to start streaming data to the server.
Results: The website pulls the data for the temperature from the server to display under the current temperature label.

2.2.3 Test 3: Waterflow Input

Initial State: The initial state of the water flow input sensor remains in a default position where it does not send any data to the server.
Input: The user turns on the waterflow switch on the website, which tells the server to ping the PI to turn it on, and to send its data back.
Results: The water flow display will show that the waterflow is on, and will show the value of the water flow sensor as in updates.

2.2.4 Test 4: Heating Output

Initial State: The initial state of the heater is off. It is not receiving any current.
Input: The user adjusts the temperature control to be above the current reading from the sensors.
Results: The current temperature reading raises slowly as the heater is turned on and heating the tank. It continues to be on and heating the tank until the desired temperature is met and at that point the heater will shut off.

2.2.5 Test 5: Adding/Removing Fish to/from the Database

Initial State: The user's virtual tank has 1 or more fish in it (for adding/removing), or is empty (for adding).
Input: The user clicks the add button to add a fish to the tank (adding) or the user clicks the "x" button on the tank panel above any of the fish cards to remove the fish.
Results: The corresponding fish will either be added or removed from the database via the add/delete buttons shown on each of the fish cards (Figures 4 and 5).

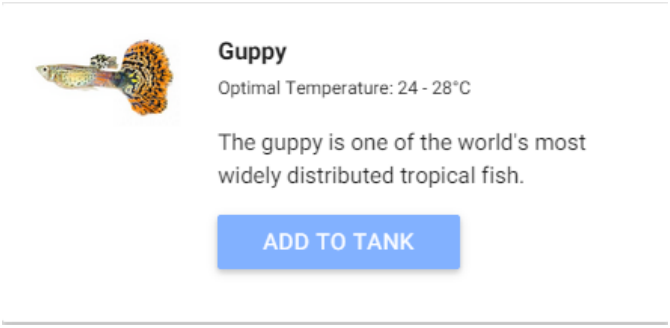

Figure 4: Fish card from the database to add to the virtual tank

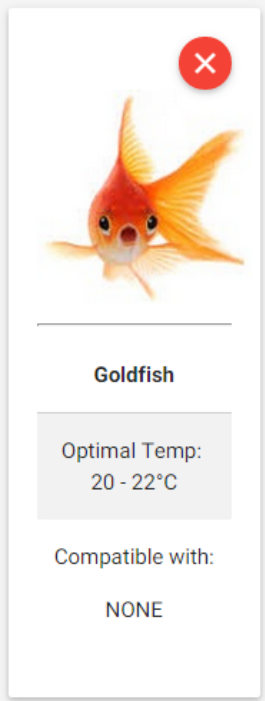

Figure 5: Fish card of a fish in the tank with the delete option in the top right

2.2.6 Test 6: RGB LED Strip

Initial State: The LEDs are off, and not producing any output current.
Input: The user turns on the lights and chooses a colour to display via the dropdown on the web application.
Results: The LEDs will turn on producing the light specified by the user. This can be viewed by either being in the same location as the FISHMO or via the camera feed on the web application.

2.3 Acceptance Testing

This test method will be conducted to determine whether or not a system satisfies the acceptance criteria and to help the customer determine whether or not they accept the system. This method of testing will be conducted by utilizing a group of independent users in a controlled setting (a test group), thus ensuring the utmost accurate results. The system will be demonstrated and utilized by small sample population. Their results to the system will then satisfy our criterion for a pass of fail of each test stated below. Furthermore, the system must meet every requirement for the tests to pass, otherwise it will fail.

2.3.1 Test: User Acceptance

If the user accepts the system as a viable tool to take care of their fish, the test will pass.The test fails if the FISHMO SAAMS system is not a viable tool to take care of the users fish.
Initial State: The user has no experience or knowledge with the FISHMO SAAMS system.
Input: Beta testing, the user applies the FISHMO SAAMS system to their current aquarium and uses it on a daily basis.
Results: FISHMO SAAMS is proven to be an adequate aquatic monitoring system therefore the system passes the acceptance test.

2.4 Security Testing

This section is to test the robustness of both the design and implementation of our software to protect the data and resources contained in and controlled by it. This directly applies to our database, as we require ample security for users personal account information and login credentials. Data must be encrypted using industry standard ciphers.

2.4.1 Test 1: Customer Login Integrity

Initial State: The user is idle on the login screen.
Input: The user types in their username and password.
Results: The user logs in, and is brought to the dashboard.

2.4.2 Test 2: Raspberry Pi Security

Initial State: The Raspberry Pi is active and the user is logged into the FISHMO SAAMS system.
Input: The user attempts to access the Pis localhost.
Results: Regardless of input, direct connection to the PI is inaccessible by anyone outside of the FISHMO SAAMS team. The user attempt is rejected as it is protected through RSA encryption specific to the key the team members have.

2.4.3 Test 3: Accessing the Dashboard without Logging In

Initial State: The user opens up their browser.
Input: The user types in the fishmo.ddns.net URL, with /dashboard.php afterword to access the dashboard.
Results: Due to session variables created when a user gets logged in with PHP, it can prevent users who are not logged in from accessing the dashboard page. If a session variable is not detected, the user is redirected back to the login page.

2.4.4 Test 4: Penetrating the database with malicious input

Initial State: The user opens up their browser.
Input: The user goes to the fishmo.ddns.net URL and attempts to put in malicious commands into the text boxes to hack the FISHMO database.
Results: The user is redirected back to the login page where they are asked to log in once again. The potential malicious code is handled by the script specified in Figure 6 to avoid any unwanted penetration into the database.

```
<?php
// Takes care of any 'dangerous' characters
if (get_magic_quotes_gpc()) {
  $process = array(&$_GET, &$_POST, &$_COOKIE, &$_REQUEST);
  while (list($key, $val) = each($process)) {
    foreach ($val as $k => $v) {
      unset($process[$key][$k]);
      if (is_array($v)) {
        $process[$key][stripslashes($k)] = $v;
        $process[] = &$process[$key][stripslashes($k)];
      } else {
        $process[$key][stripslashes($k)] = stripslashes($v);
      }
    }
  }
  unset($process);
}
?>
```

Figure 6: Code for handling any "dangerous" characters

2.5 Performance Testing

This test method is to verify that the user's software experience transitions smoothly with no latency issues and minimalist response times.

2.5.1 Test 1: FISHMO SAAMS Stress

Initial State: FISHMO SAAMS is active and under normal performance load.
Input: After initial setup, a python script will be executed to ensure execution of the commands in reasonable time frame. The input will stress the system through switching modes quickly while all functions are active to test the performance of the system.
Results: The system return the status of the PI through the stress test. The system behaves smoothly and responds quickly to the users actions.

2.5.2 Test 2: FISHMO SAAMS Database

Initial State: FISHMO SAAMS is idling.
Input: After initial setup, a MySQL script will be executed to ensure the database is capable of handling high load queries. The input will stress the system through adding/removing large amounts of fish to and from the database test performance of the database.
Results: The system return the status of the PI through the stress test. The system behaves smoothly and responds quickly to the users actions.

2.5.3 Test 3: FISHMO SAAMS Responsiveness

Initial State: The user is idle on the login page.
Input: The user performs actions on the website, such as adding fish, clicking the account tab, removing fish, changing control states, etc.
Results: The user experiences a smooth experience using the website.

3 Test Summary

3.1 Component Testing

This set of testing involved empirically testing each of the components of the FISHMO system on their own, separate from the inputs/outputs of the other components. This is to test the functionality of each interactive component

before going into integration to ensure their validity.

The component tests conducted

In summary all the results from the component testing section (refer to Section 2.1) have been successful in generating appropriate outcomes. See the table below for the list of component tests, and the status of their test.

Table 1 - Pass/Fail Test Results of the Component Testing

Test Name:	**Test Statues (Pass/Fail):**
Accessing Camera Input	Pass
Temperature Input	Pass
Waterflow Input	Pass
Heating Output	Pass
Adding/Removing Fish to/from the Database	Pass
RGB LED Strip	Pass

3.2 Integration Testing

Integration testing occurs in software test life cycle after unit testing. This is the phase in which the software and hardware modules are combined and tested as a complete system. This is crucial to ensuring harmony between our software and hardware components.

The integration tests conducted

In summary all the results from the integration testing section (refer to Section 2.2) have been successful in generating appropriate outcomes. See the table below for the list of integration tests, and the status of their test.

Table 2 - Pass/Fail Test Results of the Integration Testing

Test Name:	**Test Statues (Pass/Fail):**
Accessing Camera Input	Pass
Temperature Input	Pass
Waterflow Input	Pass
Heating Output	Pass
Adding/Removing Fish to/from the Database	Pass
RGB LED Strip	Pass
Network Adapter	Pass

3.3 Acceptance Testing

FISHMO SAAMS has been determined to be a viable alternative to aquatic system care. After careful monitoring it has been decided by the test group (used in acceptance testing) that the device is more than sufficient to monitor and maintain the overall effectiveness of a manual labourer who would routinely care for the aquatic system. Through both customer feedback and sales we will be able to determine whether or not the system is accepted in a market environment.

The acceptance test conducted

In summary all the results from the acceptance testing section (refer to Section 2.3) have been successful in generating appropriate outcomes. See the table below for the list of acceptance tests, and the status of their test.

Table 3 - Pass/Fail Test Results of the Acceptance Testing

Test Name:	**Test Statues (Pass/Fail):**
User Acceptance	Pass

3.4 Security Testing

Through the security testing, the security of the customer's information regarding their personal information as well as their product information on the database is dependent on the robustness of the encryption algorithm. By using industry standard ciphers, we are able to safely encrypt users' information onto the database.

Since this testing is being done during the beta phase of development, in-depth testing and analysis of the security of the system would be handled then most likely by a third party specialist to ensure validity. The beta phase of development would open the FISHMO product and service to select group of consumers, and rolling it out then would be the time to test the security among that group of customers.

The security tests conducted

In summary all the results from the security testing section (refer to Section 2.4) have been successful in generating appropriate outcomes. See the table below for the list of security tests, and the status of their test.

Table 4 - Pass/Fail Test Results of the Security Testing

Test Name:	**Test Statues (Pass/Fail):**
Customer Information Accessibility	Pass
Raspberry Pi Security	Pass
Accessing the Dashboard Without Logging In	Pass

3.5 Performance Testing

Performance Testing is key in ensuring that our user experience is a good one. Here we test for the response time of sensors, activation of components and the performance of the FISHMO unit overall as an entire system. We perform various tests on the system to ensure accessing information is both fast and fluid. With regular firmware/software updates as well as our customized python scripts to verify performance, we have created a user experience that is responsive, allowing the user to use the system without a hitch. The only limiting factor would be the user's internet connection, or if the server is unavailable for maintenance.

The performance tests conducted

In summary all the results from the performance testing section (refer to Section 2.5) have been successful in generating appropriate outcomes. See the table below for the list of performance tests, and the status of their test.

Table 5 - Pass/Fail Test Results of the Performance Testing

Test Name:	**Test Statues (Pass/Fail):**
FISHMO SAAMS Stress	Pass
FISHMO SAAMS Database	Pass
FISHMO SAAMS Responsiveness	Pass

3.6 Summary of Test Changes

Due to the high degree of testing and planning performed throughout the development lifecycle of the FISHMO SAAMS system, we achieved a high pass rate while analyzing the system using the test cases above. With this success allowed us to focus our efforts to further develop the interactivity and enhance the aesthetic aspects of our system.

4 List of Tables

5 List of Figures