Day Health Manager Test Plan

Version 1.0

Prepared by 4yourhealth

Senior Project 2015

April 23, 2015

Table of Contents

1. Introduction

This document provides an overview of our plans, methodologies, and execution strategies for testing the Day Health Manager system. Within it, we introduce the reader to our planned processes for testing, our criteria by which testing begins, pauses, and is considered complete for each type of tests run.

2. Test Objective

Our objective is to ensure proper functionality of the Day Health Manager system. This involves ensuring each component does its part individually and as an integrated system. Also ensure Day Health Manager system meets all specified requirements agreed to by the team and Trillium our customer.

3. Process Overview

The following represents the overall flow of the testing process:

1. Identify the functionality to be tested.
2. Identify which particular test(s) will be used to test each module.
3. Review the test data and test cases to ensure that the unit has been thoroughly verified and that the test data and test cases are adequate to verify proper operation of the unit.
4. Identify the expected results for each test.
5. Document the test case configuration, test data, and expected results.
6. Perform the test(s).
7. Document the test data, test cases, and test configuration used during the testing process. This information shall be submitted via a Day Health Manager Test Report.
8. Successful unit testing indicates that the unit is eligible for component integration/system testing.
9. Unsuccessful testing requires a bug report to be filed.
10. Test documents and bug reports shall be submitted. Any specifications to be reviewed or revised will be handled.

4. Testing Process

a. **Organize Project** - Create a Day Health Test Plan.
b. **Design/Build System Test** - Identify test cycles, cases, entrance & exit criteria, expected results, etc. The team will identify test objectives, conditions, and the types of setup/data preparation required. The test conditions are derived from the Software Requirement Specifications Document.
c. **Design/Build Test Procedures** - Set up procedures for bug-tracking and test tools.
d. **Build Test Environment** - Construct sample data for use by the system to ensure functionality of the Day Heath Manager system.

e. **Execute System Tests –** Run the tests and document the results via bug-tracking software and test result forms.
f. **Signoff** - All pre-determined exit criteria have been achieved.

5. Testing Strategy

For our testing strategy, we will write unit, integration, system, and acceptance tests. The following template will be used when testing activities are performed.

<table>
<tr><td>Tested By:</td><td></td></tr>
<tr><td>Test Type</td><td></td></tr>
<tr><td>Test Case Number</td><td></td></tr>
<tr><td>Test Case Name</td><td></td></tr>
<tr><td>Test Case Description</td><td></td></tr>
<tr><td>Item(s) to be tested</td><td></td></tr>
<tr><td>1</td><td>
<table>
<tr><td>Specifications</td><td></td></tr>
<tr><td>Input</td><td>Expected Output/Result</td></tr>
<tr><td></td><td></td></tr>
<tr><td>Procedural Steps</td><td></td></tr>
<tr><td>1</td><td></td></tr>
<tr><td>2</td><td></td></tr>
<tr><td>3</td><td></td></tr>
<tr><td>4</td><td></td></tr>
<tr><td>5</td><td></td></tr>
<tr><td>6</td><td></td></tr>
<tr><td>7</td><td></td></tr>
</table>
</td></tr>
<tr><td>2</td><td>
<table>
<tr><td>Specifications</td><td></td></tr>
<tr><td>Input</td><td>Expected Output/Result</td></tr>
<tr><td></td><td></td></tr>
<tr><td>Procedural Steps</td><td></td></tr>
<tr><td>1</td><td></td></tr>
</table>
</td></tr>
</table>

	2	
	3	
	4	
	5	
	6	
	7	

5.1. Unit Testing

Unit tests are preformed to test the functionality of a class or object on a method by method level. We choose to utilize JUnit for creation of our unit tests when working with our backend model for testing.

5.1.1. White Box Testing

We have chosen to create our testing of our Javascript front end using the tool Karma and Jasmine for our testing framework. This will use our inner knowledge of the system code to enable us to assemble a broad range of tests that will cover a large portion of our system.

5.1.2. Black Box Testing

We will be running acceptance testing from the viewpoint of an end user using their system through the design of several use case tests. Use case tests are where a specific feature is targeted and the results of using that feature are recorded as a pass or the results were as the user expected or fail when something unexpected occurred. We will also be gathering feedback of the quality of Day Health Manager based on the ease of use for the typical end user.

5.2. Integration Testing

We will be testing the integration of our Clinician Portal, our database backend system, and also the integration of the Session App with the rest of our system so that all data collection and manipulation requirements are able to be fulfilled and to ensure every part of our system is able to work together. This will involve the use of Karma and Jasmine to run our integration test cases.

5.2.1. Incremental Testing

The process we plan on following is using incremental testing for our integration testing so that the integration process can begin as each feature is ready. This will continue until all sections are completed.

5.3. System Testing

5.3.1. System Testing

This phase will begin once the entire system is finished and all integration tests are passing. The system will be deployed in a "production like" environment to be sure the entire system will work together.

5.4. Acceptance Testing

This will involve the Use Case Scenarios and the review process at each release. The purpose is to prove that each requirement is being meet in a usable way by the Day Health team.

5.4.1. Acceptance Testing

Part of the acceptance testing will be performed by 4yourhealth in performing all the Use Case Scenarios, and major feature scenarios will be selected to be performed by the Day Health team to ensure the features are usable by our targeted end users.

6. Entry and Exit Criteria

This section describes the general criteria by which testing commences, temporarily stopped, resumed and completed within each testing phase.

6.1. Unit Testing

Unit test cases shall be designed to test the validity of the Day Health Manager system functionality for each method.

6.1.1. Black Box Phase

Black box testing typically involves running the system with the range of possible input types in order to verify that it results in the right outputs using the software as an end-user would expect. We will be using Use Case Scenarios during the black box phase.

6.1.1.1. Black Box Entry Criteria

The Black Box Entry Criteria will rely on the component specification, and user interface requirements. Things that must be done on entry to the Black Box stage:

- The program must be runnable with the Graphic User Interface (GUI)
- All button functionality has to be in place
- Use case scenarios need to be generated based on the GUI and use cases to monitoring the use of every button together so that all possible button pair combinations are executed.

6.1.1.2. Black Box Exit Criteria

The Black Box Exit Criteria listed below explains what needs to be completed in-order to exit Black Box phase. To exit the Black Box phase 100% success rate must be achieved. Things that must be done upon exiting the Black Box stage:

- Use case scenarios all executed and results recorded
- Bug report entries entered for any button functionality that does not perform as expected

6.1.2. White Box Phase

White Box testing will be used to focus on key internal areas of Day Health Manager's structure. All programing errors will be recorded and sorted depending on what the state of the program is after execution.

6.1.2.1. White Box Entry Criteria

The testing team will use White Box testing to go over and verify that the specified features work as intended. The components that are tested will be tested separately to be certain which area of the program any bugs in the system are located in. To ensure successful White Box testing certain criteria must be meet.

- Program complies without error
- Module functions are all written

6.1.2.2. White Box Exit Criteria

Once each section that is in line for testing is completed 100% can we start going over the exit criteria checklist. The following is the checklist that has to be completed in order to exit the White Box testing phase.

- All functions have been tested
- Each test was documented, how was the section tested and the results
- Any and all bugs have been properly inputted into the bug tracker and the fixes have been tested

6.2. Integration Test

These tests will be done once each module is finished to ensure that different pieces of the system are all capable of functioning correctly together.

6.2.1. Integration Test Entry Criteria

In order for integration testing to begin the following criteria must first be meet.

- Both modules being integrated together must be fully coded.
- All unit tests for both modules must have been run, and any bugs found recorded into the bug tracker.
- Any fixes from unit testing must be finished and tested.

6.2.2. Integration Test Exit Criteria

For exiting the integration testing phase the following criteria needs to be completed.

- Clearly mark all bugs found in testing
- Modules are ready for System Testing. Any stubs or hooks that were made for testing are removed
- All Modules have had integration testing run with all other modules that are interacted with in the system.

6.3. System Test

The System Test criteria access for purpose in accomplishing unit testing, accomplishing integration testing, developing software system, and demonstrating the presence of a testing environment that produces the staging environment.

6.3.1. System Test Entry Criteria

This is the final stage for testing before the final release to ensure the system will work in its production environment prior to release.

- A production like environment must be setup and ready for system staging.
- All Black Box testing must be finished and found bugs must be fixed.
- All White Box testing must be finished and found bugs must be fixed.
- All Integration Testing must be finished and found bugs must be fixed.

6.3.2. System Exit Criteria

The Exit Criteria should meet and verify that all requirements are met and working.

- All errors and bugs from System Testing must be fixed and tested.

6.4. Acceptance Test

This phase of testing is designed to ensure every requirement as specified in the System Requirement Specification document has been meet by the system, and that the end users are able to perform the needed functionality.

6.4.1. Acceptance Test Entry Criteria

Beginning this phase of testing will follow our product release cycle and will involve some of the use case scenarios that highlight the specific user stories finished during each production cycle. To qualify for entry into acceptance testing the following criteria must be meet.

- Specific user stories have been finished and are able to be accessed through the GUI
- Use case scenarios have been completed for the desired features and selected by the team for inclusion in the tests Day Health to perform
- All previously found bugs in the system have been reported in the bug tracker

6.4.2. Acceptance Test Exit Criteria

This phase will be running along side other phases of testing but will be the last phase of testing to complete. For exit from the acceptance testing phase the following criteria must be meet.

- All use case scenarios have been run
- All bugs found have been properly reported and fixed with the fixes having been tested
- Sign off received on product by AJ for all requirements having been meet.