Cadence Health

Test Specifications

Mobile Application

Version 1.1, 30/10/14

Table of Contents

1. Test Specifications 3

1.1. Search Function 3

1.2. Create, Update and Delete Data 4

1.3. Camera Function 4

1.4. Tag Recognition 5

1.5. Google Maps and Geo-location 6

1.6. ORAC Rating function 6

1.7. Customization and editing of meals and Ingredients 7

1.8. Validation Tutorial Test 8

1.9. Device Compatibility 8

1.10. Application Bugs 9

1.11. General Application Performance Test 10

1.12 Test Case Lifecycle 10

2. Test Plan 11

2.1. Features to Be Tested 11

2.2. Major Constraints 11

2.3. Test Strategy 12

2.4. Test Schedule 14

2.5. Testing Resources Assigned 14

2.6. Testing Milestones 15

2.7. Test Deliverables 16

3. Change Log 17

# Test Specifications

## Search Function

**Test Description**

Search function must be able to accurately find Information and provide a drop down of recommended searches.

**Input and Output Specifications**

| Step | Input Specification | Output Specification | Observed Results | Pass/ Fail |
| --- | --- | --- | --- | --- |
| 1. | Open Application | Application should respond by touch |  |  |
| 2. | Tap on Search Function to find specific information | Search Function should be seen on the front page. Touch should be responsive |  |  |
| 3. | Search for a description ex. ‘burger | Drop down of recommended searches should be seen |  |  |
| 4. | Tap on desired information | User should be able to find their desired result with a list of searches related to burger |  |  |
| 5. | Tapping on a selected result | User should be able to select their desired result to find further information such as map location and/or meal ingredients etc. |  |  |
| 6. | **Error Handling**: **Search Error** | **If error occurs during test, re-check any spelling mistakes and coding errors. Check whether data is available for the search, if not update the database**. |  |  |
| 7. | Complete Search | Application should seamlessly connect to the data servers, providing an accurate list of search items |  |  |

## Create, Update and Delete Data

**Test Description**

Admins must be able to create, update and/or delete data such as meals and ingredients for the application.

**Input and Output Specifications**

| Step | Input Specification | Output Specification | Observed Results | Pass/ Fail |
| --- | --- | --- | --- | --- |
| 1. | Open Database | Database should show appropriate data |  |  |
| 2. | Select a table to edit, update and/or delete | The user is presented with an appropriate form with the correct number of fields based on the tables selected |  |  |
| 3. | Fill in all the fields with appropriate data | Data should be added, updated and/or deleted |  |  |
| 4. | Apply changes | Changes should be applied without error and seen in the database |  |  |
| 5. | **Invalid sequences** | **If invalid sequences occur, check for any sequence errors. If errors persists consult a professional** |  |  |
| 6. | Complete changes | The page displayed will state that new, updated or deleted information was successfully posted to the system. New data should be available through the application |  |  |

## Camera Function

**Test Description**

Camera must be able to connect to the user’s smartphones camera and be able to take snapshots which shall be saved to the user’s smartphone gallery.

**Input and Output Specifications**

| Step | Input Specification | Output Specification | Observed Results | Pass/ Fail |
| --- | --- | --- | --- | --- |
| 1. | Open camera via the application | Users camera should load up | Camera successful | Pass |
| 2. | Take a picture of a meal | Taking pictures of meals should automatically be synced to the users smartphone gallery | Pictures were taken and synced to smartphones gallery | Pass |
| 3. | **Stress Test: User should be able to take multiple pictures without the application and camera crashing** | **Application and Camera should sustain the amount of pictures taken. Check whether the maximum amount of picture input is optimal** | Multiple Pictures were taken successfully without error | Pass |
| 4. | Close camera | User should be able to close the camera and be able to further tag their pictures/meals if needed | Camera successfully closed. However tagging is still in development | Fail |

## Tag Recognition

**Test Description**

Users must be able to tag their pictures of meals using social networks such as Facebook or Twitter. Tags must be recognized correctly so other users can view tagged meals

**Input and Output Specifications**

| Step | Input Specification | Output Specification | Observed Results | Pass/ Fail |
| --- | --- | --- | --- | --- |
| 1. | Open up the application and identify what pictures will be tagged | A clear option to ‘tag’ the pictures should be seen |  |  |
| 2. | User must correctly tag the pictures to the pictures properties | Tagging should result in recognition between the users pictures and its tags |  |  |
| 3. | Complete Tagging | Tags should be compiled so that other users are able to use keywords to find specific tagged meals. |  |  |

## Google Maps and Geo-location

**Test Description**

Geolocation should provide an accurate point of where the restaurant may be located as well as where the user is located.

**Input and Output Specifications**

| Step | Input Specification | Output Specification | Observed Results | Pass/ Fail |
| --- | --- | --- | --- | --- |
| 1. | Open up the application and identify nearby restaurants based on the users input | A clear location of where the user is located and nearby restaurants should open up via google maps |  |  |
| 2. | Find Address | Google maps should be interactive enough to locate the address and other details such as phone numbers. |  |  |
| 3. | **Test Boundaries** | **Check whether google maps are linked properly and shows the correct location and address. Complete this step using an emulator with internet access** |  |  |
| 4. | Complete geolocation search | User must be satisfied with the available information |  |  |

## ORAC Rating function

**Test Description**

An ORAC rating should provide users an easy to understand scale to rate their meals. The ORAC rating should be based on ORAC values and should easily recognize a meals ingredients and generate an accurate rating.

**Input and Output Specifications**

| Step | Input Specification | Output Specification | Observed Results | Pass/ Fail |
| --- | --- | --- | --- | --- |
| 1. | Search meals to find ORAC rating | ORAC rating should be clear, precise and accurate for the user to understand |  |  |
| 2. | Identify ORAC rating | ORAC rating should be scaled by either a percentage or a 1-10 scale so the user can understand |  |  |
| 3. | **Test Boundaries** | **Test whether the ORAC rating shown is easy for the user to understand. Also recheck how accurate ORAC rating is by using several internet resources. Complete this step by using various test subject on real life users and interview their impression on the ORAC function** |  |  |
| 4. | Identify how accurate the ORAC rating is based on the ingredients listed | Even the slighted change of ingredients should still result in an accurate ORAC rating |  |  |

## Customization and editing of meals and Ingredients

**Test Description**

Meal and Ingredient customization should allow

**Input and Output Specifications**

| Step | Input Specification | Output Specification | Observed Results | Pass/ Fail |
| --- | --- | --- | --- | --- |
| 1. | Open up the application and attempt to customize Meal and ingredients | A clear function for users to allow customization and editing ingredients and meal should be seen |  |  |
| 2. | Using various scenarios change the input of the meal and ingredients | Application should allow the customization and editing of meals and ingredients |  |  |
| 3. | **Stress Test** | **Input various spelling errors and random generated info to identify whether the application can pick up any “spam” generated by the user** |  |  |

## Validation Tutorial Test

**Test Description**

Tutorial should load when the application is loaded for the first time. The tutorial should be understandable and should teach new users on how to use the application.

**Input and Output Specifications**

| Step | Input Specification | Output Specification | Observed Results | Pass/ Fail |
| --- | --- | --- | --- | --- |
| 1. | Download application and load for the first time | Loading the application for the first time should prompt the user to start a tutorial |  |  |
| 2. | Accept the tutorial | The tutorial should guide the user in a step by step fashion |  |  |
| 3. | **Validation Test: Check whether the tutorial actually teaches the user** | **Tutorial should easily be available and clear for users to understand. At the end the user must clearly understand on how to use the application** |  |  |

## Device Compatibility

**Test Description**

Proportional Display test. Application must display proportionally on all devices including tablets. Compatible on all Android devices for now

**Input and Output Specifications**

| Step | Input Specification | Output Specification | Observed Results | Pass/ Fail |
| --- | --- | --- | --- | --- |
| 1. | Open application using various emulators, smartphones and tablet devices | Application should load up on tested devices | Application tested on Android emulators were successful | Pass |
| 2. | Development team test functions on various devices | All functions should be working correctly | Developed function are working correctly | Pass |
| 3. | **Error Handling** | **If errors persist on different devices, check the resolution of the application. Check whether the application is operable on the OS. Update the OS if necessary, update any software fixes.** |  |  |

## Application Bugs

**Test Description**

To test random application crash, freeze or functional bugs.

**Input and Output Specifications**

| Step | Input Specification | Output Specification | Observed Results | Pass/ Fail |
| --- | --- | --- | --- | --- |
| 1. | Open up the application and load up various functions | Functions should load up |  |  |
| 2. | Development team testing or group testing | Using development team testing/group testing identify the procedures for testing flaws |  |  |
| 3. | Various user scenarios must be identified and produced | Following the scenarios, various flaws may be identified |  |  |
| 4. | Exit application | Close application and compile a list of flaws that have to be fixed |  |  |

## General Application Performance Test

**Test Description**

To test the loading of the application, the responsiveness of functions and touch sensitivity.

**Input and Output Specifications**

| Step | Input Specification | Output Specification | Observed Results | Pass/ Fail |
| --- | --- | --- | --- | --- |
| 1. | Open up the application | Application must load within a satisfactory limit |  |  |
| 2. | Analyze touch sensitivity | Touch sensitivity must be satisfactory |  |  |
| 3. | Open up the various functions | Functions must be responsive; they must load within a satisfactory limit and must not crash or stall. |  |  |
| 4. | Close Application | Application must close properly. Document the performance of the application |  |  |

## Test Case Lifecycle

Ideally, test cases will be raised and recorded when they are not going to be fixed immediately. Any new updates on new test cases will be followed through the lifecycle.

|  |  |
| --- | --- |
| **Step** | **Description** |
| Identify Test Case | Ensure Test case is properly identifies and can be reproduced |
| Prioritize Test Case | Prioritize test case based on severity defect. How much the test case will affect the project |
| Analyze Test Case | Analyze the test case based on acceptance criteria and implementation details. Consider what should be done to resolve the test case |
| Resolve Test Case | Implement changes and/or remediate failing test. Attempt any stress test if necessary |
| Verify Resolution | Execute tests to verify if the case is resolved and no regression is seen |
| Close Test Case | Close the test case, update any notable information into the test specification document |

# Test Plan

This is the Master Test Plan for the Cadence Health Project based on the ‘Healthy Food’ smartphone Application. This plan will cover several levels of testing such as, Specific Functionality Test, Security Test, Accessibility Test and Performance Test. The primary focus of this plan is to ensure that testing will be strategic to ensure the most effective and efficient development output.

## Features to Be Tested

The following is a list of the areas to be focused on during testing of the application. The list has been grouped based on the type of testing.

*Unit Test*

* Search Function
* Create, Update and Delete Data
* Camera Function
* Tag Recognition
* Google Maps and Geolocation
* ORAC Rating Function
* Customization and editing of meals and Ingredients

*Integration Test*

* Device Compatibility
* Validation Tutorial Test

*System Tests*

* Application Bugs
* General Application Performance Test

*Stress Test*

* Stress Test on Camera Function
* Stress Test for Input

## Major Constraints

* Testing will only occur through Android stimulators. The final release of the application will only be on the Android OS for now
* Due to limited time we may not be able to complete a thorough test analysis on the application, thus not all test cases will be fully completed by the end of the project
* Due to the time frame of the project, a few components may not be able to be fully completed and thus testing will not be able to continue
* The application project will only be on the beta stage by the end of the project.
* As there are only two programmers in the project, most testing will be allocated to them
* Unforeseen error conditions, tests cases will not test for error conditions that may arise outside the DBMS/Application

## Test Strategy

The purpose of this test strategy is to create a shared understanding of the overall targets, approach, tools and timing of test activities. The objectives are:

* Achieve higher quality output and short lead times, effectiveness and efficiency
* Minimize Time and overhead costs
* Carry deliverables on a frequent and timely manner
* Close teamwork with team, client and users
* Continuous integrations
* Frequent changes to design
* Receive and revise feedbacks

The test strategy should guide us through the common obstacles with a clear view on how to evaluate the application. Testing starts with the exploration of the requirements and what the customer really wants by elaborating on the test specification from different perspectives. Testing becomes a continuous and integrated process where all stakeholders of the project are involved.

**Test Strategy Guidelines**

|  |  |
| --- | --- |
| **Standard** | **Description** |
| Shared Responsibility | Everyone in the team is responsible for quality and assurance of the project |
| Data Management | Production data must be analyzed before used for testing |
| Test Management | Test cases, code, documents and data must be treated with the same importance as the Application System. |
| Test Completion | Attempt to complete all types of testing including Unit, System, Integration, Stress as far as feasible |

**Approach**

When executing the test specifications the following points should be administrated at all times

* Testing must be iterative
* Testers should be flexible
* Testers cannot rely wholly on test specifications, they should be able to identify new test cases and problems whenever possible
* Tested should work closely with other team members and developers
* Have a wide range of skills
* Focus on what to test and not how to test
* Focus on sufficient and straightforward results
* Focus on exploratory testing
* Define when to continue or stop testing before delivering the system to the client.

**Testing Levels**

The testing for the application project will consist of Unit, system, integration and stress tests levels. It would be ideal if testing is completed before the project deadline. However with the time line established, a complete test execution will most likely not be completed before reaching the client. This is due to the fact that the application will still be in development phase or at least beta phase by the end of the due date. As such, many of the functions may not be fully tested until after the due date. It is hoped that we complete as much of the test cases and scenarios as possible as long as the team participation is achieved.

Unit testing will be done by the developers and programmers and will be approved by the team leader. Proof of testing such as descriptions, defect information and test case lists must be provided by the programmers to the team leader before unit testing will be accepted.

System and Integration testing will be performed by the team leader with assistance of the developers and other team members. These tests will be performed last after Unit testing is completed. Programs will enter into system and integration test after all critical defects have been corrected. Stress testing will be performed by the team members with the assistance of actual end users to receive necessary user feedback.

**Test Management Strategy**

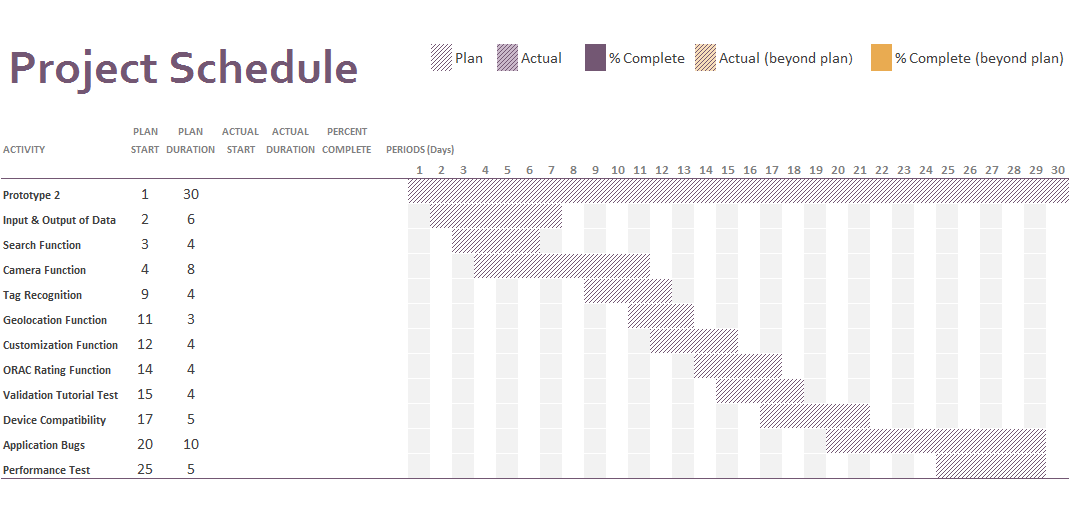
The test plan, test cases, test scenarios and bug reports should be stored in a systematic way and should be stored in the same place. Resources should be uploaded on to Github and be available anytime.

Meetings are essential for the management of test specifications. The team will meet twice every week to evaluate progress and get the team up to date. The team will identify error trends and problems as early as possible to minimize delay. The test team leader will organize a date with the client to recount any progress on the test specifications. These meeting will be schedules before each deliverable. Additional meetings can be called as required for emergency situation

## Test Schedule

The estimated time line of the competition of this project is approximately 5 months. While the accepted testing will take no longer than 1 month, therefore it is necessary for testing to planned accordingly and accurately to avoid any delays which will disrupt the overall development of the application. Testing is to be done parallel with the current application process.

The Gantt charts shows the properties that will be tested over the 1 month testing period. The main application testing will start once development on the second prototype starts. The Gantt chart will be updated throughout the development of the application.

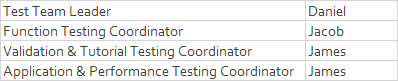


## Testing Resources Assigned

Testing of the core functions of the applications will be done through the Android emulator which is also being used for the development of the application. However a few test such as various smartphone and tablet compatibility will be tested on various devices. Tutorial testing will also need co-ordinance between the development team, client and target markets (the users).

For now, initial testing will not require special resources beyond the android emulator. However there is a possibility we may need certain resources in the near future.

However we have assigned staffing to allow an organized environment for testing. The Table below outlines the staffing resources assigned.

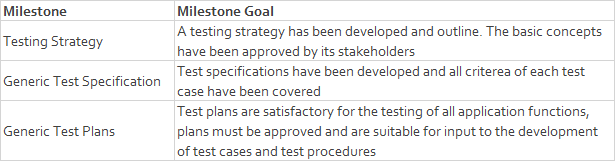


## Testing Milestones

We have split the testing milestones into 5 different phases. The diagram below demonstrates the milestones for overall testing of the application.

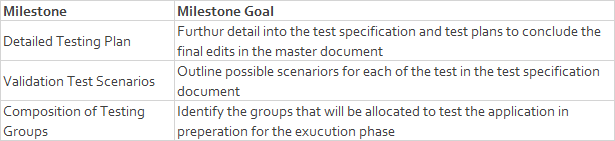
**Phase 1**: Conceptual phase

Date: 25/09/14 – 6/10/14



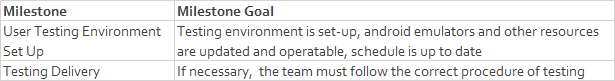
**Phase 2:** Planning phase

Date: 6/10/14 – 20/10/14



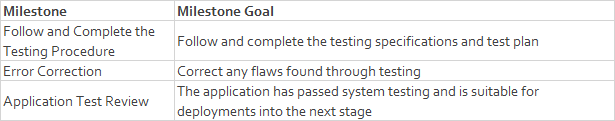
**Phase 3**: Preparation phase

Date: 20/10/14 – 27/10/14



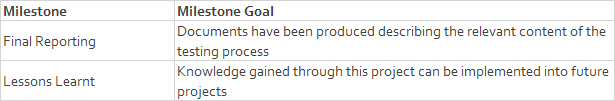
**Phase 4**: Execution phase

Date: 27/10/14 – 24/12/14



**Phase 5**: Closing phase

Date: 24/12/14 – 1/12/14



## Test Deliverables

The following list will outline the test deliverables that will be provided:

* Test Specifications
* Master Test Plan
* Project Schedule and Milestones
* Test Procedures (If applicable)
* Scenario Testing
* Stress Testing
* Error reports and summaries (fail/pass)
* Test Observations
* Development Test Checklists
* Application prototypes

# Change Log

|  |  |  |
| --- | --- | --- |
| **Content** | **Changes** | **Changed by/ Date** |
| * 1. Search Function | * Added Error Handling for Search | Rohan 30/10 |
| 1.2 Create, Update and Delete Data | * Added Invalid Sequences | Rohan 30/10 |
| 1.5 Google Maps and Geolocation | * Added Testing Boundaries | Rohan 30/10 |
| 1.6 ORAC Rating | * Added Testing Boundaries | Rohan 30/10 |
| 1.7 Customization and editing meals… | * Added Stress Test for Input | Rohan 30/10 |
| 1.9 Device Compatibility | * Added Error Handling Step | Rohan 30/10 |
| 1.12 Test Case Lifecycle | * Added Test Case Lifecycle Table for new Test Cases | Rohan 30/10 |
| Test Plan 2.1 | * Added 2.1 Features to be Tested * Unit Test, Integration Test, System Test, Stress Test | Rohan 30/10 |
| Test Plan 2.2 | * Added 2.2 Major Constraints * Time constraints, Project Constraints | Rohan 30/10 |
| Test Plan 2.3 | * Added 2.3 Test Strategy * Test Strategy Guidelines * Approach * Testing Levels * Test Management Strategy * Meetings | Rohan 30/10 |