Cadence Health

Test Specifications

Mobile Application

Version 1, 7/10/14

Table of Contents

1. Test Specifications 3

1.1. Search Function 3

1.2. Create, Update and Delete Data 3

1.3. Camera Function 4

1.4. Tag Recognition 5

1.5. Google Maps and Geo-location 5

1.6. ORAC Rating function 6

1.7. Customization and editing of meals and Ingredients 6

1.8. Validation Tutorial Test 7

1.9. Device Compatibility 7

1.10. Application Bugs 8

1.11. General Application Performance Test 8

2. Test Plan 9

2.1. Test Schedule 9

2.2. Testing Resources Assigned 10

2.3. Testing Milestones 10

2.4. Test Deliverables 12

# 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. | 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. | 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 |  |  |
| 2. | Take a picture of a meal | Taking pictures of meals should automatically be synced to the users smartphone gallery |  |  |
| 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 |  |  |
| 4. | Close camera | User should be able to close the camera and be able to further tag their pictures/meals if needed |  |  |

## 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. | 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. | 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 |  |  |

## 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 |  |  |
| 2. | Development team test functions on various devices | All functions should be working correctly |  |  |

## 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 |  |  |

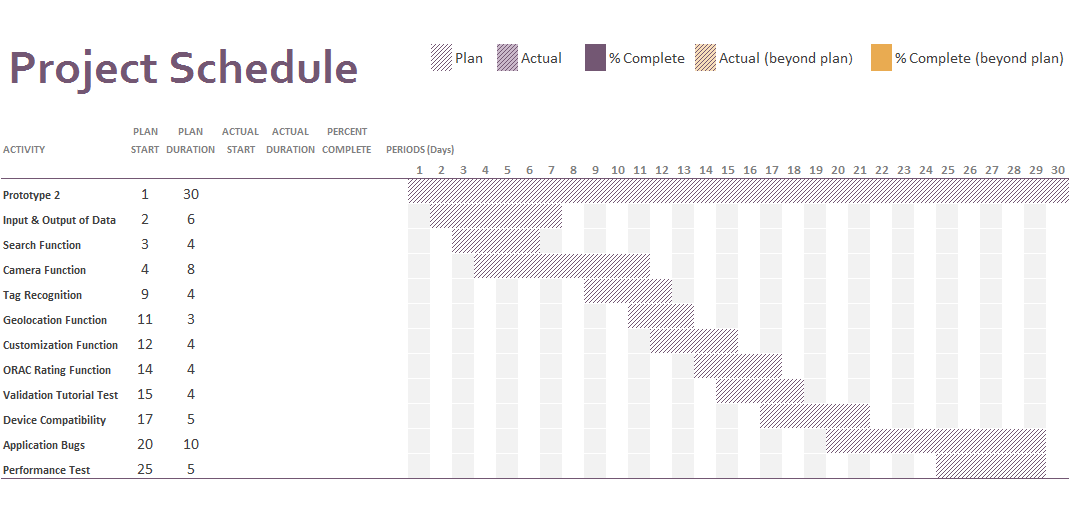
# Test Plan

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.

## 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.

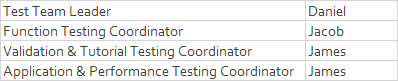


## Testing Resources Assigned

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.

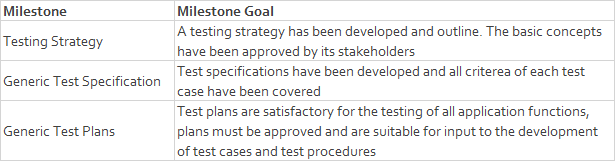


## Testing Milestones

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

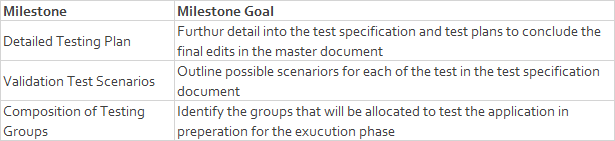
**Phase 1**: Conceptual phase

Date: 25/09/14 – 6/10/14



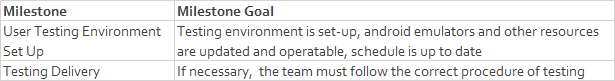
**Phase 2:** Planning phase

Date: 6/10/14 – 20/10/14



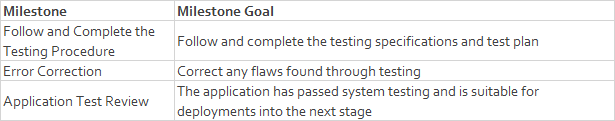
**Phase 3**: Preparation phase

Date: 20/10/14 – 27/10/14



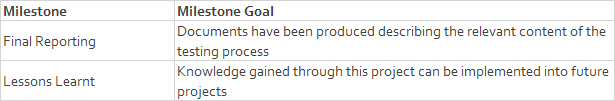
**Phase 4**: Execution phase

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