# Software Testing Report

**For**

**Menu Thingz Mobile Application**

**(Bajan Tech)**

**Prepared by**

**KOD inc**

| | | |
|---|---|---|
| **Ottor Mills** | **ID#: 180917** | **Email: 180917@gist.edu.cn** |
| **David Thomas** | **ID#: 180912** | **Email: 180912@gist.edu.cn** |
| **Nicoy Smith** | **ID#: 180902** | **Email: 180902@gist.edu.cn** |
| **Kenneth Anglin** | **ID#: 180907** | **Email: 180907@gist.edu.cn** |

| | |
|---|---|
| Course Instructor: | Thomas Canhao Xu |
| Course: | SWEN3010 |
| Date: | April 30, 2019 |

# Table of Contents

# 1. INTRODUCTION

The Menu Thingz Application is an application that provides English translations to Chinese menus in the Global Institute of Software Technology canteens. Additionally, the application describes each meal on the menu based on their ingredients and allow students to order directly from their phone requesting a specific meal to be delivered. The stakeholders of the system are the International University of the West Indies (UWI) Students & Personnel, the Restaurant owners in the GIST College Canteen. The UWI Students are the main target audience of the system as the application was designed to cater to their day to day interactions with the chinese menus in the cafeteria.

## 1.1 Objectives

One of the objectives of testing the Menu Things application is to first test the application as if it were a student ordering food from the Gist cafeteria, to see if a foreign student can successfully place an order. With testing the Menu Thingz application the expected results were a list of malfunctioning features and any vulnerabilities in security and possible points of attack by hackers or users with malicious intent,

## 1.2 Testing Strategies

Testing the Menu Thingz Android Application was done using the automated testing tool known as Calabash which is used to test android applications using the Gherkin language. This tool was paired with the Black Box Testing technique to carry out effective tests.

## 1.3 Scope

Testing will be conducted on the Menu Thingz Android Application version 1.0 using calabash version 2.3.1 on the Android Api level 26.

## 1.4 Reference Material

- Menu Things Software Requirements Specification Document
- IEEE Software Testing Template
- Cucumber/ Calabash Testing Documentation

## 1.5 Definitions and Acronyms

- Calabash - Automation Testing Framework
- APK - Android Application KIT
- Black Box Testing - A Type of Testing
- Equivalence Partitioning - Black Box Testing Technique
- Boundary Value Analysis - Black Box Testing Technique
- BDD - Behavior Driven Development
- UI - User Interface
- OS - Operating System

## 2.TEST ITEMS

- Software Requirements Specification - Functional , non-functional requirements of the Menu Thingz Applications.

## 3. FEATURES TO BE TESTED

- Menu Thingz Login
- Menu Thingz Login
- Adding Items to Orders
- View Items in Order
- Removing Items from Orders
- Order Placement

## 4. FEATURES NOT TO BE TESTED

- Internal Implementation
- Internal Structure
- Internal Design

# 5. APPROACH

Testing the Menu Thingz Application. was done using the automated testing framework Calabash. Calabash is an open source Acceptance Testing framework that allows you to write and execute tests for iOS and Android Apps. It's an Automated User Interface Framework that allows tests to be written in Ruby using Cucumber. Calabash works by enabling automatic UI interactions within an application such as pressing buttons, inputting text, validating responses, etc. While this is a great first step in automated UI acceptance test automation, the real benefits can be gained when Calabash tests are executed on real mobile devices. Calabash tests can be configured to run on hundreds of different Android and iOS devices, providing real-time feedback and validation across many different form factors, OS versions and hardware specs. Calabash, which is considered as Behavior-Driven Development (BDD) test automation framework, means application behaviors are specified and tested to see if done correctly. Writing tests with Calabash can be done easily because the language syntax is human-readable and a tester does not need coding experience or skills that are typically required for software developers.

## 5.1 Black Box Testing

Black box testing also known as Behavioral Testing, is defined as a testing technique in which functionality of the Application Under Test  is tested without looking at the internal code structure, implementation details and knowledge of internal paths of the software. This type of testing is based entirely on software requirements and specifications. In BlackBox Testing the is focus on inputs and outputs of the software system without bothering about internal knowledge of the software program. Initially, the requirements and specifications of the system are examined, then the tester chooses valid inputs (positive test scenario) to check whether Software under test processes them correctly. Also, some invalid inputs are chosen to verify that the Software under testis able to detect them, the tester determines expected outputs for all those inputs, the software tester constructs test cases with the selected inputs, the test cases are executed, software tester compares the actual output with the expected outputs, defects if any are fixed and re-tested.

Advantages of Black Box testing are: Unbiased tests because the designer and tester work independently, tester is free from any pressure of knowledge of specific programming languages to test the reliability and functionality of an application / software, facilitates identification of contradictions and vagueness in functional specifications, test is performed from a user’s point-of-view and not of the designer’s and test cases can be designed immediately after the completion of specifications.

## 5.1.1 Black Box Testing Techniques

In equivalence-partitioning technique only one condition from each partition is tested. This is because we are assuming that all the conditions in one partition will be treated in the same way by the software. If one condition in a partition works, we assume all of the conditions in that partition will work, and so there is little point in testing any of these others. Similarly, if one of the conditions in a partition does not work, then assume that none of the conditions in that partition will work, there is little point in testing any more in that partition.

Boundary value analysis, tests the behavior of a program at the boundaries. When checking a range of values, after selecting the set of data that lie in the valid partitions, next is to check how the program behaves at the boundary values of the valid partitions. Boundary value analysis is most common when checking a range of numbers. For each range, there are two boundaries, the lower boundary (start of the range) and the upper boundary (end of the range) and the boundaries are the beginning and end of each valid partition. The Tester should design test cases which exercises the program functionality at the boundaries, and with values just inside and just outside the boundaries. The assumption is that if the program works correctly for these extreme cases, then it will work correctly for all values in between the valid partition. Testing has shown that defects that arise when checking a range of values the most defects are near or at the boundaries.

## 6. PASS / FAIL CRITERIA

**Scenario 1**

**Feature: Register Functionality**

**Scenario: I should be able to register for an account**

**When I enter my username and I enter my password**

**Then I press "Register"**

**Then I should be a registered user**

**Then I should see "Restaurants List"**

**Test Cases for Scenario 1**

1. Check system behaviour when valid username and password is entered.
2. Check system behaviour when a weak password is entered.
3. Check system behaviour when password is left blank and Register is clicked.
4. Check system behaviour when special characters are entered.
5. Check system behaviour when an existing users credentials are entered

| ID | Scenario | Steps | Test Data | Expected Result | Actual Result | Pass/ Fail |
|---|---|---|---|---|---|---|
| T1 | Check Register with valid Username and Password | 1. Go to App<br>2. Select Register<br>3. Enter Username<br>4. Enter Password<br>5. Click Register | Username: otboss<br>Password: password | User Should Be able to be Registered | User Registers Successfully | Pass |

| | | | | | | |
|---|---|---|---|---|---|---|
| T2 | Check User Register With a weak password | 1. Go to App<br>2. Select Register<br>3. Enter Username<br>4. Enter Password<br>5. Click Register | Username: otboss<br>Password: p | User Should not be able to Register | User is able to Register | Fail |
| T3 | Check User Register With blank data | 1. Go to App<br>2. Select Register<br>3. Enter Username<br>4. Enter Password<br>5. Click Register | Username:<br>Password: | User Should not be able to Register | User is able to Register | Fail |
| T4 | Check User Login With Special Characters | 1. Go to App<br>2. Select Register<br>3. Enter Username<br>4. Enter Password<br>5. Click Register | Username:\"$<br>Password:\"$<br><br>Username: !<br>Password: !<br><br>Username: " !"<br>Password: " !" | User Should not be able to Register | User is able to Register | Fail |
| T5 | Check system behaviour when an existing users credentials are entered | 1. Go to App<br>2. Select Register<br>3. Enter Username<br>4. Enter Password<br>5. Click Register | Username: otboss<br>Password: password | User Should not be able to Register | User is able to Register | Fail |

**Scenario 2**

**Feature: Login Functionality**

**Scenario: As a valid user I can log into my app**

**Given: I am a registered user**

**When I enter my username and I enter my password**

**Then I press "Login Now"**

**Then I should see "Restaurants List"**

**Test Cases for Scenario 2**

1. Check system behaviour when valid username and password is entered.
2. Check system behaviour when an in*valid* password is entered.
3. Check system behaviour when password is left blank and Login is clicked.
4. Check system behaviour when special characters are entered.

| ID | Scenario | Steps | Test Data | Expected Result | Actual Result | Pass/ Fail |
|---|---|---|---|---|---|---|
| T1 | Check Login with valid Username and Password | 1. Go to App<br>2. Select Login<br>3. Enter Username<br>4. Enter Password<br>5. Click Login | Username: otboss<br>Password: password | User Should Be able to Login | User Logs in Successfully | Pass |
| T2 | Check User Login With an Invalid password | 1. Go to App<br>2. Select Login | Username: otboss<br>Password: password1 | User Should not be able to Login | User is not able to Log in | Pass |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | 3. Enter Username<br>4. Enter Password<br>5. Click Login | | | | |
| T3 | Check User Login With blank data | 1. Go to App<br>2. Select Login<br>3. Enter Username<br>4. Enter Password<br>5. Click Login | Username:<br>Password: | User Should not be able to Login | User is able to Login | Fail |
| T4 | Check User Login With Special Characters | 1. Go to App<br>2. Select Login<br>3. Enter Username<br>4. Enter Password<br>5. Click Login | Username:\"$<br>Password:\"$ | User Should not be able to Login | User is able to Login | Fail |

**Scenario 3**

**Scenario: The app should allow the user to add items to their order**

**Given: I am a registered user**

**When I click on a restaurant**

**Then I should see menu items**

**Then I touch the menu item**

**Then I see the menu item in my orders list**

**Test Cases for Scenario 3**

1. Check system behaviour when adding items to the order list.
2. Check system behaviour when adding multiple items to the order list.

| ID | Scenario | Steps | Test Data | Expected Result | Actual Result | Pass/ Fail |
|---|---|---|---|---|---|---|
| T1 | Check system behaviour when adding items to the order list. | 1. Go to App<br>2. Select Restaurant<br>3. Select Menu Item<br>4. Add Item<br>5. Go to Orders | Restaurant : Live Love Home<br><br>Menu Item: Chicken With Pasta | Item Should be Added to User's Order List | Item added Successfully | Pass |

|  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|
| T2 | Check system behaviour when adding multiple items to the order list. | 1. Go to App<br>2. Select Restaurant<br>3. Select Menu Item<br>4. Add Item<br>5. Select Add more<br>6. Select Menu Item<br>7. Go to Orders | Restaurant : Live Love Home<br><br>Menu Item: Chicken With Pasta<br><br>Menu Item:Rice With Pork<br><br>Menu Item: Fresh Mush | Item Should be Added to User's Order List | Items added Successfully | Pass |

**Scenario 4**

**Scenario: The app should allow the user to remove items to their order**

**Given: I am a registered user**

**When I click on Orders**

**Then I should see menu items**

**Then I touch Remove item**

**Then I see the menu item has been removed in my orders list**

**Test Cases for Scenario 4**

1. Check system behaviour when Removing items to the order list.

| ID | Scenario | Steps | Test Data | Expected Result | Actual Result | Pass/ Fail |
|---|---|---|---|---|---|---|
| T1 | Check system behaviour when Removing items to the order list. | 1. Go to App<br>2. Go to Order list<br>3. Select Menu Item<br>4. Select Remove | Restaurant : Live Love Home<br><br>Menu Item: Chicken With Pasta | Item Should be Added to User's Order List | Item removed Successfully | Pass |

**Scenario 5**

**Scenario: The app should allow the user to place their order**

**Given: I have items in order list**

**When I click on Orders**

**Then I click on MAKE ORDER**

**Then I should see a message in my notifications**

**Test Cases for Scenario 5**

1. Check system behaviour when placing an order.
2. Check system behaviour when placing an order with empty order list.

| ID | Scenario | Steps | Test Data | Expected Result | Actual Result | Pass/ Fail |
|---|---|---|---|---|---|---|
| T1 | Check system behaviour when placing an order. | 1. Go to App<br>2. Go to Orders List<br>3. Click Make Order<br>4. Receive Notification | Menu Item: Chicken With Pasta<br><br>Menu Item:Rice With Pork<br><br>Menu Item: Fresh Mush | Receive Notification that the order has been placed. | Notification Received | Pass |
| T2 | Check system behaviour when placing an order with empty order list. | 1. Go to App<br>2. Go to Orders List<br>3. Click Make Order<br>4. Receive Notification | Menu Item: | Receive no Notification | Notification not Received | Pass |

# 7. TESTING PROCESS

## 7.1 Test Deliverables

- Test Input
- Calabash Test Logs
- Calabash Scripts
- Calabash Feature Files

## 7.2 Testing Tasks

- Ensure the Ruby Version Manager RVM is installed on the system
- Ensure the Ruby environment is installed on the system.
- Ensure the appropriate Ruby Gems have been installed
- Ensure the appropriate emulator is installed. It should come packaged with the installation of the Android Studio software. If the tester is using a physical device, make sure the device is connected.
- Ensure that the application has internet permissions allowed.
- Ensure the Calabash testing framework is installed on the system.
- Ensure the feature files that are testing the features have been completed.
- Ensure that the tester has access to a release version of the application

## 7.3 Resources

The following resources were used in the testing process:

- MenuThingz.apk
- Pixel XL Emulator
- Calabash 2.3.1
- IEEE812 Testing Doccument ‘
- Visual Studio Code

# 8. ENVIRONMENTAL REQUIREMENTS

## 8.1 Hardware

- Mac OS
- Windows OS
- Android Device with API level of at least 26

## 8.2 Software

- Command Prompt with Administrator (SUDO) Access
- Calabash for Android version 2.3.1
- Android Studio
- Visual Studio Code

## 8.3 Assumptions and Recommendations

Based on testing carried out on the Menu Thingz application can be improved by setting a minimum password input length of at least 6 characters so that users accounts are less susceptible to hackers, also by making mandatory that users have unique usernames and make users register with their emails for validity purposes or another thing that can also be done is to implement a dual email and username login approach also validating the users email is also a necessity and placing a hold on the account establishing until it is validated happens. An example is Facebook. You can login to facebook with your username, your email or even your phone number. The form fields can also be improved by limiting the length of user’s inputs. Adding a signing out option for the application as well as displaying an error message when trying to make an order with an empty item list will also improve the application.

## 9. Test Results

Testing was done on the vulnerability of the login and sign up requirements and the ability to carry out requirements defined in the Menu Thingz Software Requirements Specification Document. Firstly testing the login and sign up features show that the application allows for users to have the same username and allows the user to enter weak passwords. Weak Passwords can make your account information vulnerable to hackers. Another issue was detected where the form fields have no max length value, meaning the form fields allow the user to input passwords and usernames of any length. The Menu Thingz application also does not have an option for users to sign out which made testing the application more difficult. However the application allows users to successfully: Signup , Login into the application, View Restaurant Listings meaning the system allows the user to view a list of restaurants available, View Menus meaning the system  allows the user to view the menus from restaurants, Request Order meaning the system allows the user make order requests from their phones, Request Delivery meaning the system provides a delivery option, Send SMS meaning the system compiles the order information  into an SMS to be sent to the restaurant manager, Remove orders meaning the system allows the user to remove an order that they have created.