Smart Waiter Test Report

Meraj Patel #1137491
Pavneet Jauhal #1149311
Shan Perera #1150394

April 24, 2016

Contents

List of Tables

Revision History

Date	Comments
March 20, 2016	Test results added
March 20, 2016	Account Tests added
March 21, 2016	Order Transactions added
March 21, 2016	Introduction Created
April 21, 2016	Added test results for customization. Addressed Dan's comments

Table 1: Revision History Table

1 Changes Made

- Addressed all of Dan's comments
- Added item selection and customization test results
- Added traceability to functional requirements
- Added tables numbers
- Modified account and order transaction test results

2 Introduction

2.1 Purpose of the Report

This section will provide an introduction and general outline of the Smart-Waiter test report.

2.2 Scope of Testing

The test report primarily focuses on the overall correctness of the software with regard to the test cases provided. Many of the test cases provided are in the form of functional dynamic tests. This plan is fairly exhaustive, further testing will be performed for the final revision to ensure a completely smooth and intuitive user experience.

2.3 Organization

In section 2 we provide the introduction to the test report. In section 3 we outline and provide the results of the system testing, including but not limited to: Database security testing, order transactions testing, account testing. Section 4 describes the Usability tests that were conducted and carried out by the Smart-Waiter team.

3 System Testing

3.1 Database Testing

3.1.1 Purpose

Database testing is critical component in correlation with Smart Waiter application. Testing was conducted to ensure users are able to retrieve and send information regarding restaurant orders.

3.1.2 Structural Test

Database testing was exclusively performed from structural testing point of view. The goal of this component of testing was to verify that the database queries always return valid and correct data, under all circumstances. Specifically, a separate database was created for testing purposes, because any failed test case could affect the state of the original database.

3.1.3 Test Factors

As per test plan, the following test factors will be considered for database testing;

- Correctness
- Performance
- Security
- Reliability

3.1.4 Correctness

In this section of database testing, tester had to verify that the application receives complete and correct data from the couchbase server.

3.1.5 Correctness Tests Results

Database Correctness Tests							
Test ID	Initial State	Input	Output	Severity of Defect	Summary of Defect	Comments	Result
Test 1- Correct retrieval of database	Empty Database	Add menu items to database	queryAllRestaurant() returns all data which matches input	NA	NA	NA	Pass
Test 2- Correct retrieval of updated database	Empty Database	1) Add menu items to database 2) Query Database 3) Update the database	queryAllRestaurant() returns all data which matches input	NA	NA	NA	Pass
Test 3 - Retreive Empty Database	Empty Database	None	queryAllRestaurant() results in a crash	Gating Release	Need try & catch statement to handle this issue	Easy Fix - Implement try & catch and mark this issue closed	Fail
Test 4 - Support backwards compatibility	Empty Database	Add menu items to the database,with attributes that the application does not support (added review comments section)	queryAllRestaurant() returns all data which matches input	NA	NA	NA	Pass

Table 2: Correctness Test Results

3.1.6 Performance

In this section of database testing, tester had to verify that the data query was performed in a reasonable amount of time.

3.1.7 Performance Test Results

Database Performance Testing							
Test ID	Initial State	Input	Output	Severity of Defect	Summary of Defect	Comments	Result
Test 1 - Test database query with 10 menu items	Empty database	Add 10 menu items to Database	getRestaurantByBarcode() function returns all menu in less than 1.5 seconds	NA	NA	NA	Pass
Test 2 - Test database query with 50 menu items	Empty database	Add 50 menu items to Database	getRestaurantByBarcode() function returns all menu in less than 1.5 seconds	NA	NA	NA	Pass
Test 3 - Test database query with menu items	Empty database	Add 100 menu items to Database	getRestaurantByBarcode() function returns all menu in less than 1.5 seconds	NA	NA	NA	Pass

Table 3: Performance Test Results

3.1.8 Security

In this section of database query testing, tester had to verify that database allows correct access control. Database penetration testing was not required because we will leverage couchbase security implementation, testing and certifications.

3.1.9 Security Test Results

Database Security Tests							
Test ID	Initial State	Input	Output	Severity of Defect	Summary of Defect	Comments	Result
Test 1 -Verify incorrect access to menu catalog database is denied	Empty database	1) Add menu items to database 2) Set invalid client permission (client key) in the application	Query for a menu item. The access to database was denied	NA	NA	NA	Pass
Test 2 - Verify that correct access to menu catalog databse is accepted	Empty database	1) Add menu items to database 2) Add correct client and matching key to sync gateway	Query for menu items with the correct client permissions (client Key).This request was allowed	NA	NA	NA	Pass

Table 4: Security Test Results

3.1.10 Reliability

In this section of database testing, we tested the robustness and reliability of the database.

3.1.11 Reliability Automated Test Results

Database Reliability Tests							
Test ID	Initial State	Input	Output	Severity of Defect	Summary of Defect	Comments	Results
Test 1 - Dependability tests	This Test runs on original database populated with 30 items	Wrote script to fire HTTP get request every 2 minutes	The Query successfully returns the results	NA	NA	NA	Pass
Test 3 - Verify database does not malfunction with stress	Empty Database	1) Add menu items to database 2) Query Databse simultaneously from 30 devices	All Queries returned menus successfully	NA	NA	NA	Pass

Table 5: Reliability Test Results

3.2 Barcode Scanning

3.2.1 Purpose

Barcode scanning tests were conducted to make sure users are able to scan a barcode with minimal attempts. Also, to check if appropriate messages are displayed according to each test case.

3.2.2 Test Factors

- Correctness

- Performance
- Ease of use

3.2.3 Correctness, Performance and Ease of use

Testing is performed to ensure barcode scanning provides correct results in an efficient and timely manner. As well to make sure the user is easily able to scan the barcode and is provided helpful messages in regards to errors encountered.

3.2.4 Functional Test Factors

Test cases were conducted to ensure the following functional requirements are met:

- The product shall scan a QR code to access the restaurants menu.
- The product shall provide error codes

3.2.5 Functional Unit Test

As per our test plan, functional unit tests were conducted to assess test cases. Doing so replicates real world usage.

3.2.6 Test Results

Test Case	Initial State	Input	Output	Result
Scan working barcode	Barcode scanning page	Eligible barcode	Restaurant menu	PASS
Scan corrupt barcode	Barcode scanning page	Corrupt barcode	Barcode scanning page Message - "Invalid barcode. Please try again"	PASS
Scan random picture	Barcode scanning page	Random picture	Barcode scanning page Message - "Invalid barcode. Please try again"	PASS
Scan corrupt barcode – third attempt	Barcode scanning page	Corrupt barcode	Barcode scanning page Message - "Invalid barcode. Please contact host"	PASS
Scan deprecated barcode	Barcode scanning page	Deprecated barcode	Barcode scanning page Message - "Invalid barcode. Please try again"	PASS
Scan deprecated barcode – third attempt	Barcode scanning page	Deprecated barcode	Barcode scanning page Message - "Invalid barcode. Please contact host"	PASS

Table 6: Barcode Scanning Test Results

3.3 Accounts

3.3.1 Purpose

Account creation and login tests were performed to ensure Smart-Waiter users are able to create an account quickly while still adhering to the account constraints set by Smart-Waiter. This is also to ensure proper error checking is implemented in the applications account related modules.

3.3.2 Functional Dynamic Test

As per our test plan, manual functional dynamic tests were performed to assess the following test cases. This allows the system to be exhaustively tested which will minimize or completely erase errors in real world usage.

3.3.3 Test Factors

As per test plan, the following test factors will be considered for Accounts testing;

- Correctness
- Data Integrity

3.3.4 Correctness and Data Integrity

In this section of Account testing, the tester had to verify that all data being passed is in the correct format and that the passwords were being stored properly so that user logins can be authenticated.

3.3.5 Functional Test Factors

Test cases were conducted to ensure the following functional requirements are met:

- The product shall allow the user to register an account for complete access to the products payment services.
- The product shall provide error codes

3.3.6 Test Results

Test Case	Initial State	Input	Output	Result
Successful account creation	Account creation page	Correct password length, first name, last name, address, postal code and phone number	Advance back to sign in screen	PASS
Unsuccessful account creation	Account creation page	Incorrect password length	Message – "Password length too short"	PASS
Unsuccessful account creation	Account creation page	First name field blank	Message – "First name invalid"	PASS
Unsuccessful account creation	Account creation page	Last name field blank	Message – "Last name invalid"	PASS
Successful sign in	Sign in page	Correct password	Proceed to barcode scan page	PASS
Unsuccessful sign in – invalid password	Sign in page	Invalid password	Message - "invalid password"	PASS

Table 7: Accounts Test Results

3.4 Item Selection and Customization

Smart-Waiter needs to ensure a user is able to select, customize and add item to cart.

3.4.1 Dynamic Unit Test

Manual black box tests will be performed to assess item customization and for the ability to add to cart. This test will be in the form of unit testing. Various test cases described below will be conducted. This is effective as it replicates real world usage and will provide a census of expected number of successful attempts to evaluate functional test factors.

3.4.2 Test Factors

- Correctness
- Reliability
- Ease of use

3.4.3 Functional Test Factors

Test cases were conducted to ensure the following functional requirements are met:

- The product will allow the user to view their current order
- The product shall provide error codes
- The product will allow a user to request special instructions
- The product will allow users to remove items from their order

3.4.4 Test Results

Test Case	Initial State	Input	Output	Result
Select menu item	Menu items page	User clicks item	Item customization page	PASS
Add item to cart	Special instructions page	User clicks checkmark	Message - "Added item to cart" Item visible on cart page	PASS
Add toppings	Toppings page	User clicks on toppings checkboxes. Adds item to cart.	Item name along with item toppings displayed on cart page	PASS
Add side order	Side order page	User clicks on side order name. Adds item to cart	Item name along with side order displayed on cart page	PASS
Modify item toppings	Cart page	User clicks customize icon. Proceed to click on toppings checkboxes	Item name along with modified item toppings displayed on cart page	PASS
Modify side order	Cart page	User clicks customize icon. Proceed to click on side order name.	Item name along with modified side order displayed on cart page	PASS
Delete item	Cart page	User clicks delete icon	Item is removed off cart page	PASS

Table 8: Item Selection and Customization Test Results

3.5 Order Transactions

3.5.1 Purpose

Order transactions are a critical part of Smart-Waiter, any unforeseen bugs or errors could effect the operations of the restaurant. Therefore, these modules of Smart-Waiter were rigorously tested to ensure no unexpected behaviour and a smooth and intuitive experience. Tests were performed to ensure Smart-Waiter users are able to order from the restaurants menu quickly without having to wait for a server to arrive. This is also to ensure proper error checking is implemented in the applications transaction related modules.

3.5.2 Functional Dynamic Test

As per our test plan, manual functional dynamic tests we performed to assess the following test cases. In a few of these test cases, manual testing for unexpected behaviour was performed by passing unexpected input.

3.5.3 Test Factors

As per test plan, the following test factors will be considered for database testing;

- Correctness
- Reliability
- Data Integrity
- Data security
- Ease of use

3.5.4 Correctness, Reliability, Data Integrity, Security and Ease of Use

In this section of Order Transaction testing, the tester had to verify that all data being passed is in the correct format and valid. Multiple tests to verify that the credit card information being passed is valid are performed. The tests are also used to check the reliability of the Stripe API and overall error testing. Some test cases check for unexpected behaviour in the Sides and Toppings module of the application by performing unexpected input. This is done to test the overall reliability of the Orders module.

3.5.5 Functional Test Factors

Test cases were conducted to ensure the following functional requirements are met:

- The product will allow users to pay for their order using a credit card
- The product shall provide error codes
- The product will allow users to place their order.

3.5.6 Test Results

Test Case	Initial State	Input	Output	Result
Proceed to payment with items in cart	Cart page. Items in cart	Click checkmark	Advances to confirm order page	PASS
Proceed to payment with no items in cart	Cart page. No items in cart	Click checkmark	Message – "Cart is empty. Please add items before checkout"	PASS
Proceed to payment page with items in cart	Confirm order page	Click checkmark	Advances to credit card information page	PASS
Input valid credit information	Credit card information page	Input valid credit card number, name, expiration date, CVC	Message – "order sent" Return to account sign in page	PASS
Input invalid credit card information	Credit card information page	Input invalid credit number or name or expiration date or CVC	Message – "Invalid card details"	PASS

Table 9: Order Transaction Test Results

4 Usability Test

Usability tests are conducted to assess the user's ability to complete routine tasks, and acquire their impression of the application.

4.1 Summary

To retrieve insightful results, participants were asked to complete a series of tasks and answer a brief questionnaire afterwards.

The first usability test has already been conducted on February 4, 2016. A total of six participants were gathered to conduct this test. To replicate an adequate demographic, three participants chosen are experienced using android applications, while the remaining three have little to no experience.

The proceeding sections provide insight and results of the usability test conducted.

4.2 Methodology

4.2.1 Tasks conducted

Participants were given a list of tasks to complete including:

- Task 1: Create and login to account
- Task 2: Scan barcode to retrieve menu
- Task 3: Customize and add items to cart
- Task 4: View cart
- Task 5: Delete item
- Task 6: Modify item
- Task 7: Confirm and pay for order

4.2.2 Questionnaire

Participants were asked to rate from 1 to 5 (1 - strongly disagree, 5 - strongly agree), provided the following statements:

1. I was able to complete the task quickly using the system
2. It was easy to learn how to use the system
3. I prefer using Smart-Waiter over ordering in a traditional sense
4. The interface of the system was pleasant
5. The system has all the functions and capabilities I expect it to have
6. Whenever I made a mistake using the system, I could recover easily and quickly
7. Overall I was happy using the system

4.3 Testing Results

4.3.1 Questionnaire Results

Case	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	Average
Complete task quickly	1	2	0	3	0	Neutral
Easy to learn	0	2	1	1	2	Agree
Prefer using Smart-Waiter over traditional menu	0	0	2	4	0	Agree
Interface of system is pleasant	2	3	1	0	0	Disagree
System has all functionalities and capabilities	1	4	1	0	0	Disagree
I could recover easily and quickly	0	0	2	3	1	Agree
Overall, I was happy with the system	0	0	2	4	0	Agree

Table 10: Questionnaire Results

4.3.2 User Feedback

After completing the usability test, we asked participants for feedback in terms of their experience. Specifically we asked for their; likes, dislikes and recommendations.

Likes

- Convenient for ordering take out at restaurant
- Ability to customize items and send special instructions
- Ease of use (according to experienced android application users)

Dislikes

- Look of GUI
- Unable to modify account settings
- Unable to save receipt

Recommendations

- Add settings page
- Offer ability to email receipt

4.4 Conclusion

Conducting this usability test definitely helps our team in terms of adjusting requirements to meet user recommendations. Specifically the following changes will be implemented:

- Create settings page
- Improve GUI
- Allow users to view order history

After implementation of these additions, a second usability test will be conducted. New participants will be gathered in order to provide unbiased results.