Smart Waiter Test Report

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

April 15, 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

Table 1: Revision History Table

1 Introduction

1.1 Purpose of the Report

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

1.2 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. Automatic testing for modules in Smart-Waiter were not feasible, as mentioned in the Test Plan. [You should give a quick summary as to why it was not feasible here. —DS] This plan is fairly exhaustive, further testing will be performed for the final revision to ensure a completely smooth and intuitive user experience.

1.3 Organization

In section 1 [Use LaTeXproperly! Use refs! —DS] we provide the introduction to the test report. In section 2 we outline and provide the results of the system testing, including but not limited to: Database security testing, order transactions testing, account testing. Section 3 describes the Usability tests that were conducted and carried out by the Smart-Waiter team.

2 System Testing

2.1 Database Testing

2.1.1 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.

2.1.2 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.

Note: A mix of scripts and manual commands were required for database testing. The tests cannot be fully automated because it is not feasible, as mentioned in test plan.

[Explain what you mean when you say they cannot be fully automated. Are your scripts partial automation? —DS]

2.1.3 Test Factors

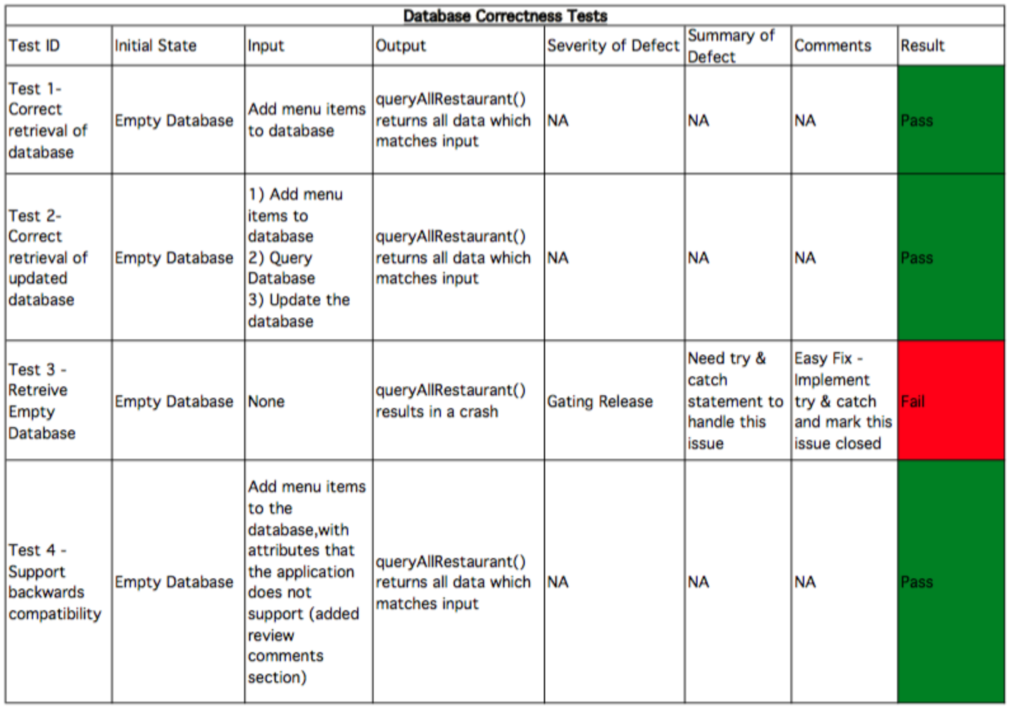
As per test plan, the following test factors will be considered for database testing;

- Correctness
- Performance
- Security
- Reliability

2.1.4 Correctness

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

2.1.5 Correctness Tests Results



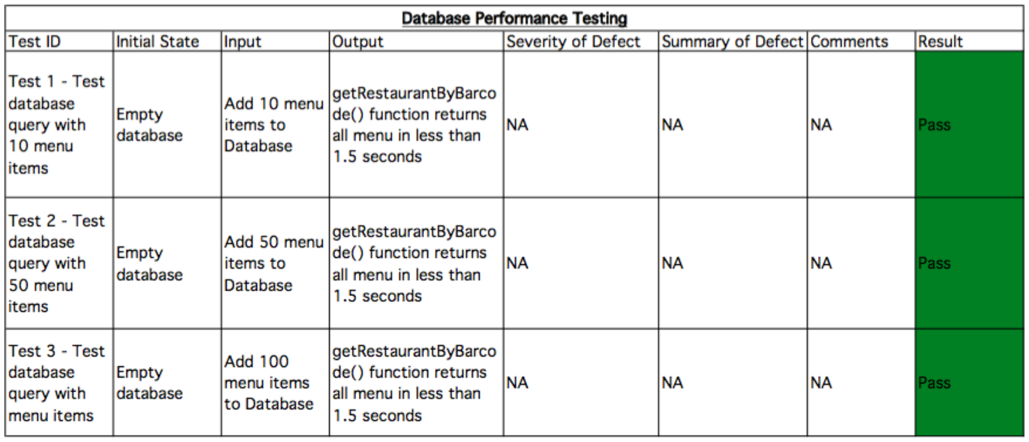
Database Correctness Tests

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

2.1.6 Performance

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

2.1.7 Performance Test Results



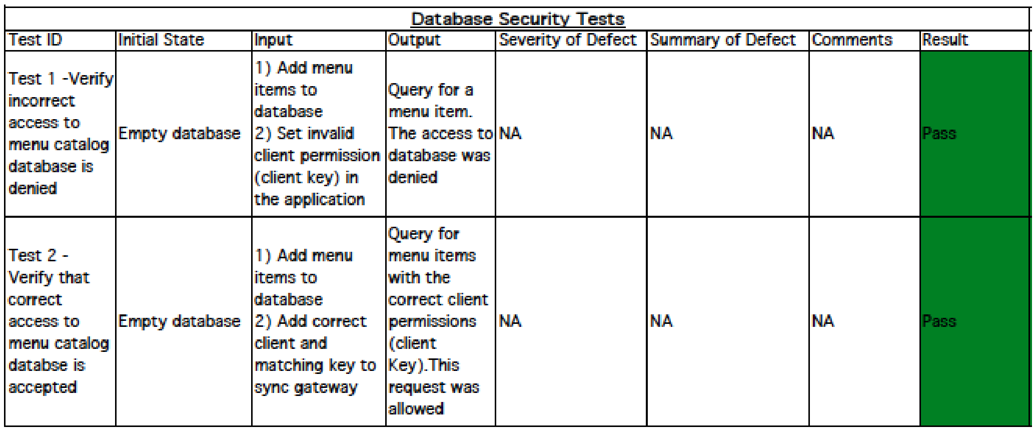
Database Performance Testing

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

2.1.8 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.

2.1.9 Security Test Results



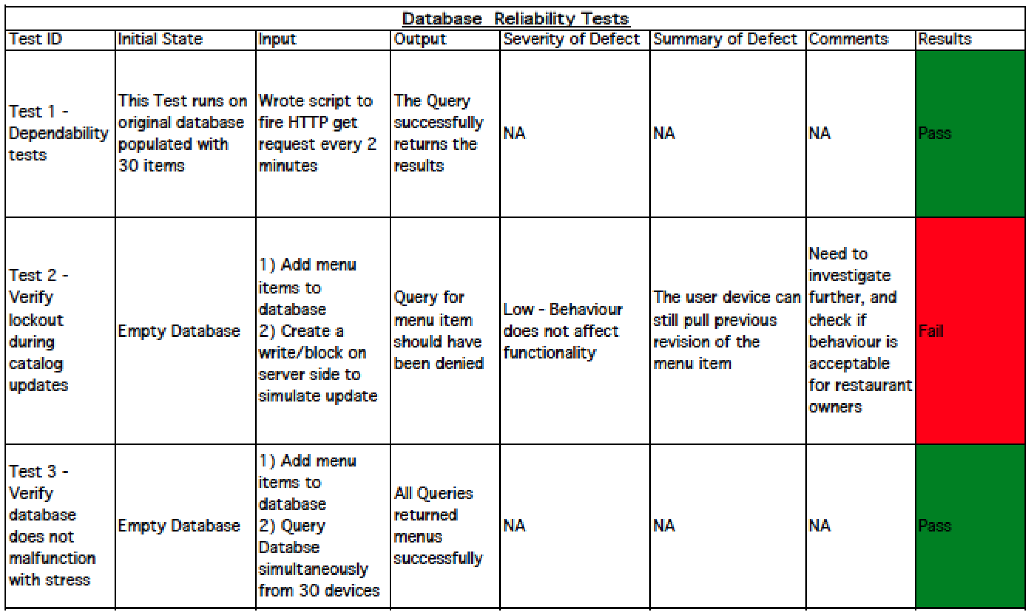
Database Security Tests

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

2.1.10 Reliability

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

2.1.11 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 2 - Verify lockout during catalog updates	Empty Database	1) Add menu items to database 2) Create a write/block on server side to simulate update	Query for menu item should have been denied	Low - Behaviour does not affect functionality	The user device can still pull previous revision of the menu item	Need to investigate further, and check if behaviour is acceptable for restaurant owners	Fail
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

[Test 2's result seems like it would be a major problem. —DS]

2.2 Barcode Scanning

2.2.1 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.

2.2.2 Test Factors

- Correctness
- Performance
- Ease of use

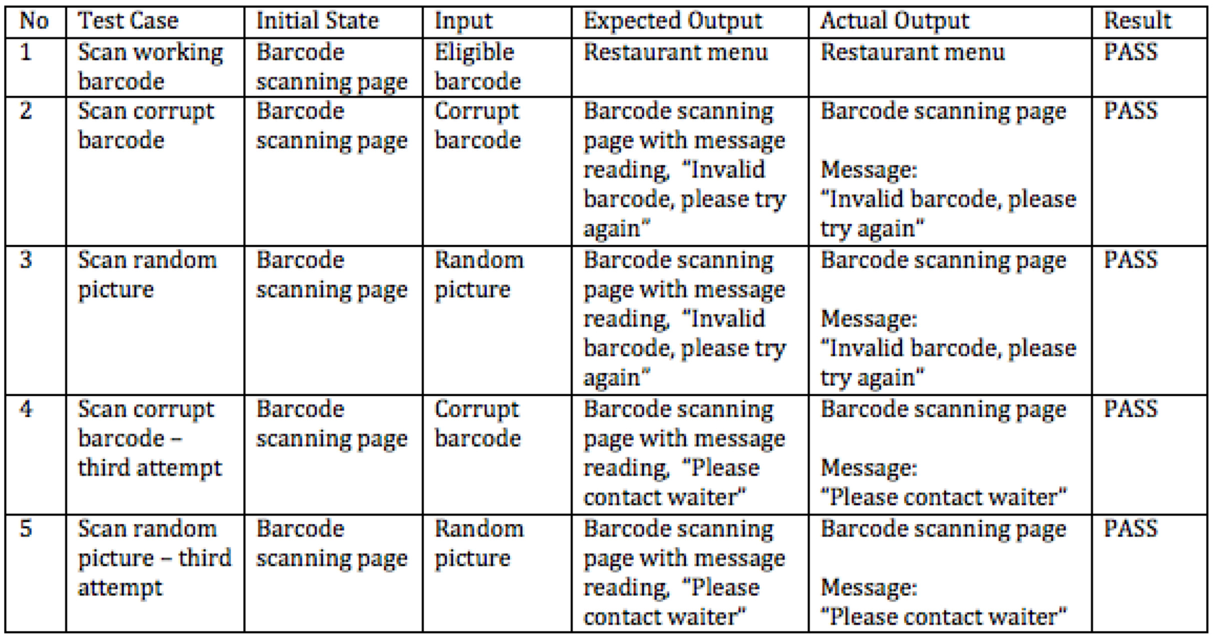
2.2.3 Correctness, Performance and Ease of use

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

2.2.4 Functional Unit Test

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

2.2.5 Test Results



No	Test Case	Initial State	Input	Expected Output	Actual Output	Result
1	Scan working barcode	Barcode scanning page	Eligible barcode	Restaurant menu	Restaurant menu	PASS
2	Scan corrupt barcode	Barcode scanning page	Corrupt barcode	Barcode scanning page with message reading, “Invalid barcode, please try again“	Barcode scanning page Message: “Invalid barcode, please try again“	PASS
3	Scan random picture	Barcode scanning page	Random picture	Barcode scanning page with message reading, “Invalid barcode, please try again“	Barcode scanning page Message: “Invalid barcode, please try again“	PASS
4	Scan corrupt barcode – third attempt	Barcode scanning page	Corrupt barcode	Barcode scanning page with message reading, “Please contact waiter“	Barcode scanning page Message: “Please contact waiter“	PASS
5	Scan random picture – third attempt	Barcode scanning page	Random picture	Barcode scanning page with message reading, “Please contact waiter“	Barcode scanning page Message: “Please contact waiter“	PASS

[What about scanning deprecated/obsolete barcodes? —DS]

2.3 Accounts

2.3.1 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.

2.3.2 Functional Dynamic Test

As per our test plan, manual functional dynamic tests [“were” —DS] we 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.

2.3.3 Test Factors

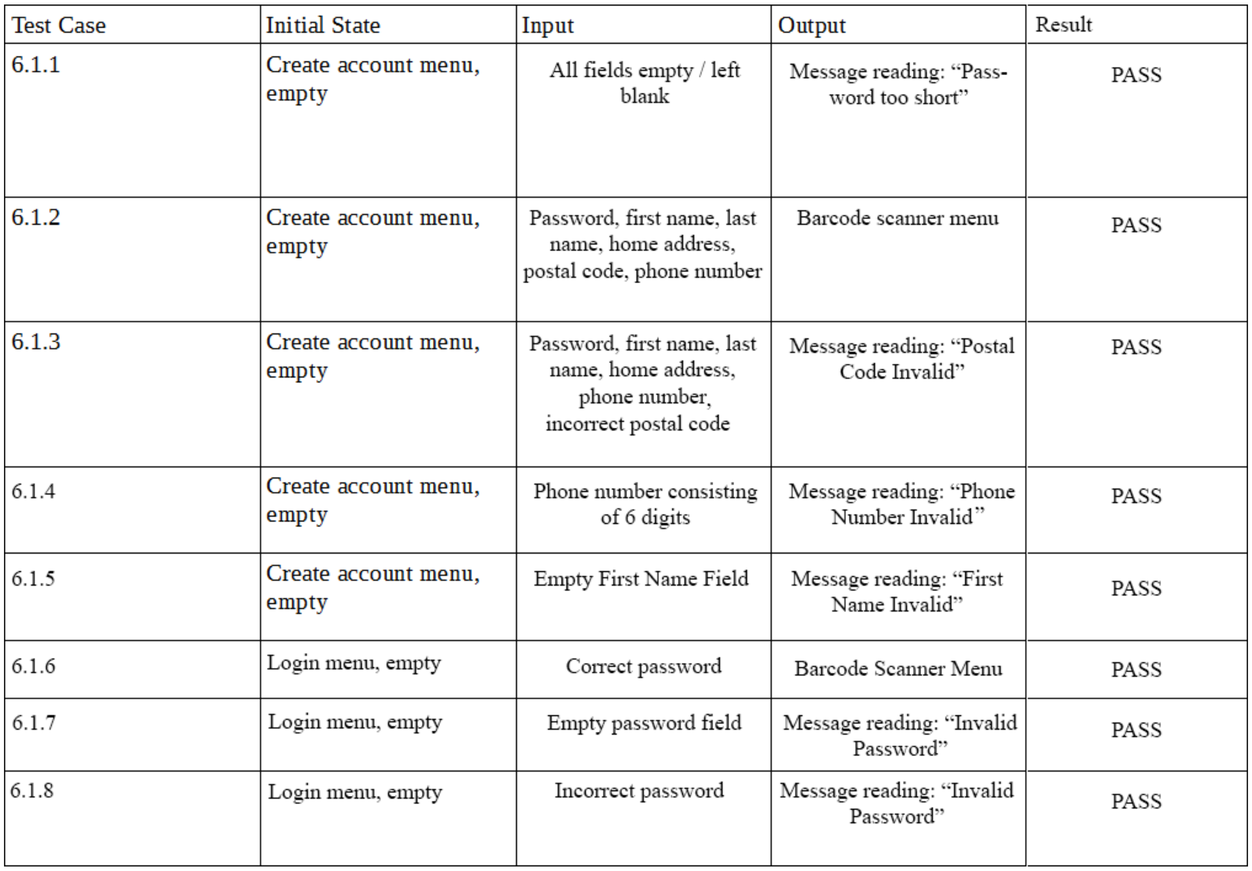
As per test plan, the following test factors will be considered for database testing;

- Correctness
- Data Integrity

2.3.4 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.

2.3.5 Test Results



Test Case	Initial State	Input	Output	Result
6.1.1	Create account menu, empty	All fields empty / left blank	Message reading: “Pass-word too short”	PASS
6.1.2	Create account menu, empty	Password, first name, last name, home address, postal code, phone number	Barcode scanner menu	PASS
6.1.3	Create account menu, empty	Password, first name, last name, home address, phone number, incorrect postal code	Message reading: “Postal Code Invalid”	PASS
6.1.4	Create account menu, empty	Phone number consisting of 6 digits	Message reading: “Phone Number Invalid”	PASS
6.1.5	Create account menu, empty	Empty First Name Field	Message reading: “First Name Invalid”	PASS
6.1.6	Login menu, empty	Correct password	Barcode Scanner Menu	PASS
6.1.7	Login menu, empty	Empty password field	Message reading: “Invalid Password”	PASS
6.1.8	Login menu, empty	Incorrect password	Message reading: “Invalid Password”	PASS

2.4 Order Transactions

2.4.1 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.

2.4.2 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. This is so any unexpected errors that may come up from unusual input is identified and fixed so that this does not add any extra work or effect the day-to-day operations of our restaurant partners. The two test cases that are performed to identify any unexpected behaviour is 6.2.1 and 6.2.7 in the table below. The 6.2.1 test is performed by rapidly clicking different sides that are available in the sides menu before the application transitions to the next activity. The 6.2.7 test is performed by selecting toppings for an entree menu item. Then editing the toppings using the function available in the order summary page, the toppings are changed and the side is changed and added to cart multiple times. These tests yielded no strange behaviour, and the application performed as expected.

2.4.3 Test Factors

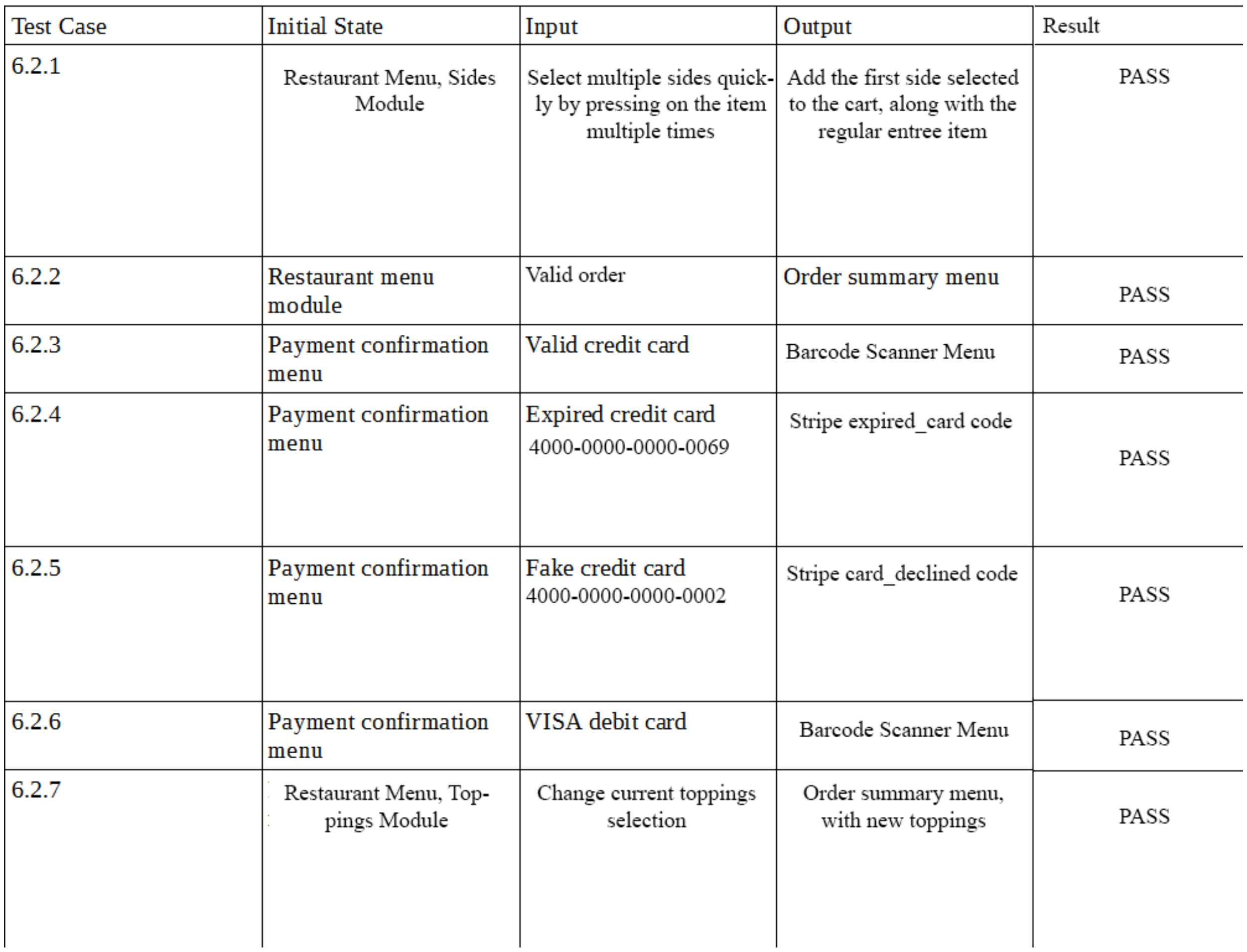
As per test plan, the following test factors will be considered for database testing;

- Correctness
- Reliability
- Data Integrity

2.4.4 Correctness, Reliability and Data Integrity

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.

2.4.5 Test Results



Test Case	Initial State	Input	Output	Result
6.2.1	Restaurant Menu, Sides Module	Select multiple sides quickly by pressing on the item multiple times	Add the first side selected to the cart, along with the regular entree item	PASS
6.2.2	Restaurant menu module	Valid order	Order summary menu	PASS
6.2.3	Payment confirmation menu	Valid credit card	Barcode Scanner Menu	PASS
6.2.4	Payment confirmation menu	Expired credit card 4000-0000-0000-0069	Stripe expired_card code	PASS
6.2.5	Payment confirmation menu	Fake credit card 4000-0000-0000-0002	Stripe card_declined code	PASS
6.2.6	Payment confirmation menu	VISA debit card	Barcode Scanner Menu	PASS
6.2.7	Restaurant Menu, Toppings Module	Change current toppings selection	Order summary menu, with new toppings	PASS

[Why aren't you creating your tables in LaTeX? Your table is being cut off. —DS]

3 Usability Test

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

3.1 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.

3.2 Methodology

3.2.1 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

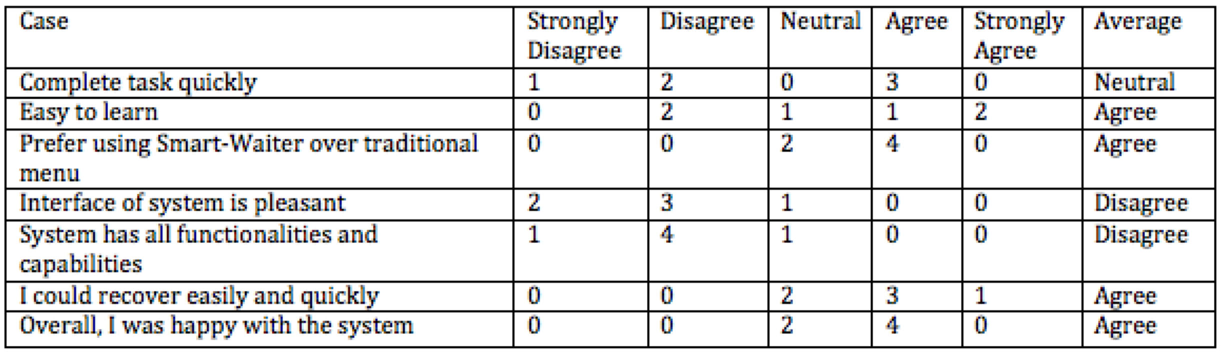
3.2.2 Questionnaire

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

3.3 Testing Results

3.3.1 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

3.3.2 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

3.4 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.

[To reiterate: Create your tables in LaTeX, it will make things clearer. Also, why do none of your tables/figures have captions? —DS]