Concordia University
Department of Computer Science
and Software Engineering

Touch For Food

Test Plan

SOEN 490
Capstone Project
Fall 2012 – Winter 2013

CloudNine

Josh Hum (Team Leader)	9583157
Katrina Anderson	9106251
Cristian Asenjo	9280014
Christian Daher	9599673
Cynthia Donato	9353852
Mikhail Levkovsky	9583165
Patrick Modafferi	9401377
Ryan Nasr	9605614
Matthew Tam	9675701

Table of Contents

Touch For Food

Test Plan

Version 11.18

Revision History

Date	Rev.	Description	Author(s)
2012-09-17	0.00	Document Creation	Katrina Anderson
2012-10-31	1.01	Added to sprint 1 and contributed section 2	Patrick Modafferi
2012-10-31	1.02	Contributed to section 4.1 and 4.2.1	Mikhail Levkovsky
2012-11-01	1.03	Contributed to Section 4.2.4	Matthew Tam
2012-11-01	1.04	Reviewed Section 4.2.1	Matthew Tam
2012-11-01	1.05	Contributed Section 1	Josh Hum
2012-11-02	1.06	Added sections 5.3, 5.4 & 5.5	Cristian Asenjo
2012-11-02	1.07	Added section 3	Ryan Nasr
2012-11-02	1.08	Added section 4.2.3 minor corrections to 4.2.4	Christian
2012-11-03	1.09	Added section 4.2.2	Cynthia Donato
2012-11-03	1.10	Added section 5.1 & 5.2	Katrina Anderson
2012-11-03	1.11	Corrected sections 5.1, 5.2, 5.3 & References. Formatted Document	Katrina Anderson
2012-11-04	1.12	Corrected section 4.2.3	Cynthia Donato
2012-12-16	3.13	Corrected Section 4.2, Section 5.5	Mikhail Levkovsky
2012-12-20	4.14	Reviewed document and reduced redundancies	Katrina Anderson
2013-02-09	7.15	Updated Section 3.1	Mikhail Levkovsky
2013-02-09	7.16	Reviewed and updated document for submission	Cynthia Donato
2013-02-09	9.17	Updated document with current plan.	Katrina Anderson
2013-03-30	11.18	Reviewed Test Plan	Mikhail Levkovsky

1 Introduction

1.1 Purpose

The purpose of this test plan is to ensure a structured and organized way of testing the TFF software. This document describes various techniques and approaches that will be used in the testing effort. This Test Plan will define both the items that must be tested as well as the criteria giving rise to a successful test.

The Test Plan for TFF will define four main concerns:

- Mission and motivation for testing
- List of features to be tested.
- List of features not to be tested
- Testing approach and techniques
- Testing workflow

The Test Plan will provide clear testing definitions and guidelines to ensure a quality product.

1.2 Scope

The TFF system will be tested using three approaches: Unit testing, user interface testing and regression testing. For more details on each type of testing such as % coverage, success criteria, testing goals, etc., see *Section 4.2 Testing Techniques and Types*.

1.3 Intended Audience

The main audience of this test plan is the CloudNine team, who will all participate in the testing of TFF. Any supervisors and stakeholders should also be able to understand the basic plan.

1.4 Document Terminology and Acronyms

Refer to the SRS document - Appendix B Glossary and Appendix C Acronyms for a complete list of terms and definitions.

1.5 References

Please refer to Appendix A – References for a list of resources that may be referenced while preparing the Test Plan.

2 Evaluation Mission and Test Motivation

2.1 Background

In any software methodology testing is an important step. TFF is being developed using an iterative process, as such testing will be conducted each iteration. Testing frequently will increase code quality and mitigate risk. This section of the test plan will state the purpose of conducting testing activities and describe the reasoning behind the types of tests that were chosen for this project.

Refer to the Vision Document for more information regarding the history of TFF.

2.2 Evaluation Mission

The mission for the evaluation effort is to coordinate testing in a manner that reveals program defects, improves code quality and mitigates risk; specifically the risks involved with concurrent demands on the TFF software and security of sensitive client information.

2.3 Test Motivators

2.3.1 Risk Motivators

By testing early and often, bigger and more costly problems can be detected in early phase of development rather than later on in the project. The cost to repair a bug increases exponentially with respect to the lifecycle period in which it was created. Therefore, testing the functionalities iteratively, as they are being developed, will be a major factor in mitigating risk for the project.

2.3.2 Testing for Functionality and User Stories

The team will be following an agile methodology. In JIRA, an agile tracking tool, every feature is defined by a user story along with a set of subtasks. The team will add tests to these lists of tasks. By doing so, the test must be completed in order for the story to be considered closed or fixed. This will motivate our developers to test their code in order to be able to close the respective story.

2.3.3 Design and Customer Satisfaction

A major metric for success in the TFF application will be customer satisfaction and overall ease of use. The motivation behind many of our tests will be to validate that the team is building a user friendly product that will appeal to both demographics described.

2.3.4 Summary

It is important to keep regular updates on the test plan in order to satisfy the previously mentioned goals and to remember why testing is necessary. Risk, traceability and customer satisfaction are factors that will motivate the team to test thoroughly in order to maintain quality.

3 Target Test Items

3.1 Features to be Tested

The high level list of features to be tested is described below. These features are all related to at least one user story. All tests associated to these features must pass successfully in order to close the user stories associated to them. In other words, a story can only be defined as 'complete' if its related tests all pass. These tests are not limited to a development workstation. They should also be run on TFF supported web capable mobile devices.

- Menu management
- Table management
- Order management
- Order food
- ~~Reservations~~
- ~~Restaurant personal page~~
- Comment, rating and review system
- Customer management
- ~~Restaurant reporting and statistics~~
- Manage bills for restaurants
- Manage bills for customer
- The Output and Input Mappers (layer between the data store and the controllers)

3.2 Features Not to be Tested

Any auto-generated code from Visual Studio will not be tested. We will assume that these parts are already tested by the Visual Studio team; however, any changes made to the auto-generated code will be tested. In addition, there will not be any tests related to the operating system during development.

4 Test Approach

The testing for the TFF application will be done using three testing techniques: unit testing, user interface testing, and regression testing. This section of the test plan will indicate the objective, target items, testing goals, techniques, required tools, success criteria, failure contingencies and any special considerations involved with each testing methodology. For some tests, specific standards need to be followed in order to prevent false test results (i.e. different formatting in browsers) [**1**,**2**,**3**].

4.1 Initial Test Idea Catalogues and Other Reference Sources

Please refer to *Appendix A – References*, for the listing of existing resources that will be used to simulate the identification and selection of the specific tests to be conducted [**4**].

4.2 Testing Techniques and Types

4.2.1 Unit Testing

Technique Objective:	The objective of unit testing is to establish a basis for the successful completion of TFF functionalities. A user story can only be considered complete when its unit tests have successfully passed. Unit tests also ensure that new functionality does not break any existing ones.
Target Items	The features groups listed in Section 3.1 – Features to be Tested.
Testing Goal:	Unit tests must cover a minimum of 60% of the functionalities implemented by developers. Coverage will be measured by establishing the percent of statements included during the execution of unit tests. The Visual Studio IDE includes built in tools to find the block percent covered during the execution of a series of unit tests [**5**].
Technique:	Unit tests will be written and executed using the Visual Studio IDE [**5**].
Oracles:	Visual Studio will provide a detailed report of the unit test results [**5**].
Required Tools:	**Touch For Food –** Access to the system code is required to run unit tests. **Visual Studio 2010 IDE –** Required for writing and running unit tests.
Success Criteria:	An individual unit test will “pass” if its programmed objective has been completed without any errors.
Failure contingencies:	A unit test will fail if the execution of its programmed objective results in an error. An error can either be a logical one (i.e. wrong data is displayed) or a runtime one (i.e. null pointer exception). A story cannot be closed until its Unit Test has passed. The defect will be addressed in the current sprint if it is critical (i.e. inhibits progress of a task in the current sprint). If the defect is not critical it will be moved to the backlog to be re-evaluated during sprint planning.
Special Considerations:	Any auto generated code provided by Visual Studio will not be tested by the developers.

4.2.2 User Interface Testing

Technique Objective:	The objective of user interface testing is to exercise the event-driven nature of the application and log the respective results. It verifies that the user is able to execute each user story in an effective manner. UI testing will also test the access methods of TFF, such as mouse movement and tab keys to ensure that navigation is properly executed.
Target Items	The features groups listed in Section 3.1 – Features to be Tested.
Testing Goal:	100% coverage of the TFF screens
Technique:	Developers will write a minimum of one test case per user story. Each test case must then be manually executed by the developer to determine if it passes or fails.
Oracles:	After the manual execution of each test the tester will indicate if the test case has passed or failed in the test report.
Required Tools:	User Interface testing will be performed on multiple devices, such as desktops, laptops~~, tablets~~ and smart phones. The restaurant management aspect of the application is meant to be viewed from a computer browser and will therefore be tested on Safari, Internet Explorer 9, Google Chrome and Mozilla Firefox browsers. Restaurant menus, order summaries, reviews and bill management are meant to be viewed from a cell phone application and will therefore be tested on Android, iOS ~~and Windows phones~~.
Success Criteria:	A test will be considered as "passed" if the output of the manual test follows the flow of events indicated in the *Description* and *Expected Results* section of the respective test case being executed. No errors should appear at any of the steps.
Failure contingencies:	A test will be considered as "failed" if the output of the manual test does not follow the flow of events indicated in the *Description* and *Expected Results* section of the respective test case being executed. In addition, if any errors should appear during the execution of a test, the test is considered to have failed. If a test fails the tester will investigate the cause: If failure of a test indicates that there is an error in the system then the tester will create a defect to rectify the error. If failure of the test is due to the test no longer being valid, the tester will adjust the parameters of the test.
Special Considerations:	Not all properties of the TFF can be accessed or tested through UI testing on a particular device (cell phone vs. browser). It is important to distinguish which scenario is associated to each particular device in order to provide efficient testing. The test case should specify the *Target Platform.*

4.2.3 Regression Testing

Technique Objective:	The objective of regression testing is to discover any bugs after implementing new features in the system. This testing verifies that any changes to the existing TFF system do not impact existing functionality.
Target Items	The feature groups listed in Section 3.1 – Features to be tested
Testing Goal:	60% for unit tests 100% for UI tests
Technique:	Unit regression testing will be executed with Visual Studio IDE [**5**]. UI regression testing will be done manually. Both programmed and manual tests will be executed at the end of each sprint.
Oracles:	For unit tests, Visual Studio will provide a detailed report of the unit test results [**5**]. For UI tests, the tester will indicate if the test has passed or failed in the test report.
Required Tools:	**Touch For Food** – Tests require system code to run **Visual Studio 2010 IDE** – Required to run unit tests **Web Browser(Chrome, IE, Firefox, Safari)** –Required to run UI tests **Web Capable Mobile Devices (Smartphones ~~Tablets~~)** – Required to run UI tests
Success Criteria:	A regression test will "pass" if it successfully meets the criteria of its programmed and acceptance objectives. There should be no conflicts with the objectives. TFF can only be considered complete if it passes all regression tests.
Failure contingencies:	A regression test will "fail" if any of the criteria of its unit tests or UI tests fail. At the end of a sprint, all regression tests must pass for the sprint to be concluded successfully. If a test fails the tester will investigate the cause: If failure of a test indicates that there is an error in the system then the tester will create a bug to rectify the error. If failure of the test is due to the test no longer being valid, the tester will adjust the parameters of the test.
Special Considerations:	Some parts of the TFF may not be accessible for tests with Visual Studio

Note: Usability testing was removed due to time constraints.

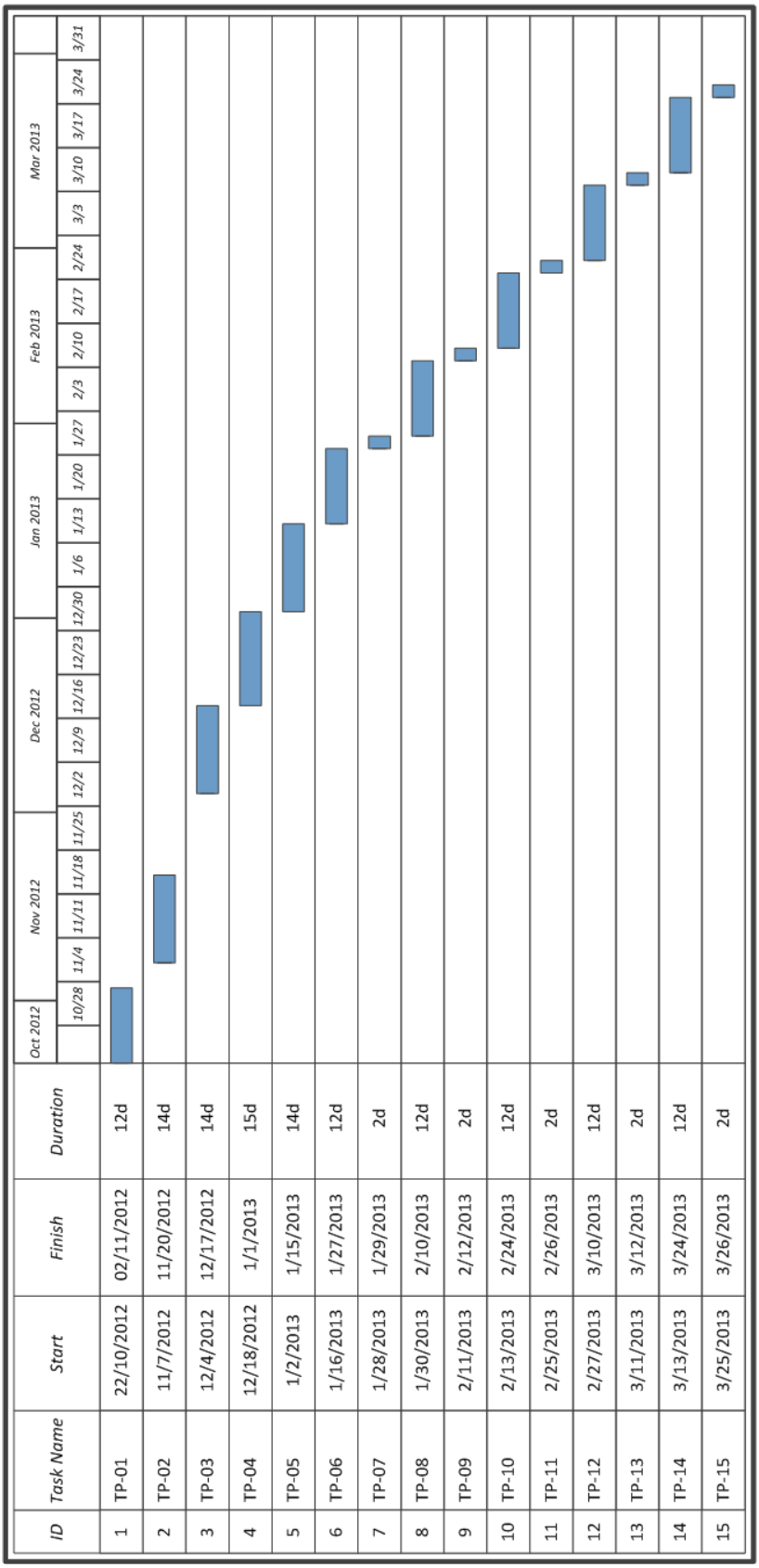
5 Testing Workflow

Section 5 of this test plan document will list the test phases of the TFF system. It will detail approximate testing time spans and illustrating them along with any dependencies in a Gantt chart. In addition, this document will also describe the management tools we plan to use during testing, as well as how we plan to handle test slippage and defects [**6**].

5.1 Test Phase Table

Test Phase ID	Phase Description	Phase Period
TP-01	**Iteration 1**: Run unit tests and fix bugs that are found. Update documents accordingly.	October 22nd – November 2nd
TP-02	**Iteration 2**: Run unit tests and fix bugs that are found. Update documents accordingly.	November 7th – 20th
TP-03	**Iteration 3**: Run unit tests and fix bugs that are found. Update documents accordingly.	December 4th – 17th
TP-04	**Iteration 4**: Run unit tests and fix bugs that are found. Update documents accordingly.	December 18th – January 1st
TP-05	**Iteration 5**: Run unit tests and fix bugs that are found. Update documents accordingly.	January 2nd – 15th
TP-06	**Iteration 6**: Run unit tests and fix bugs that are found. Update documents accordingly.	January 16th – 27th
TP-07	**Iteration 6:** Run user interface tests and fix bugs that are found. Update documents accordingly	January 28th –29th
TP-08	**Iteration 7**: Run unit tests and fix bugs that are found. Update documents accordingly.	January 30th – February 10th
TP-09	**Iteration 7:** Run user interface tests and fix bugs that are found. Update documents accordingly	February 11th – 12th
TP-10	**Iteration 8**: Run unit tests and fix bugs that are found. Update documents accordingly.	February 13th – 24th
TP-11	**Iteration 8:** Run user interface tests and fix bugs that are found. Update documents accordingly	February 25th – 26th
TP-12	**Iteration 9**: Run unit tests and fix bugs that are found. Update documents accordingly.	February 27th – March 10th
TP-13	**Iteration 9:** Run user interface tests and fix bugs that are found. Update documents accordingly	March 11th – 12th
TP-14	**Iteration 10**: Run unit tests and fix bugs that are found. Update documents accordingly.	March 13th – 24th
TP-15	**Iteration 10:** Run user interface tests and fix bugs that are found. Update documents accordingly	March 25th – 26th

5.2 Test Phase Gantt Chart



ID	Task Name	Start	Finish	Duration
1	TP-01	22/10/2012	02/11/2012	12d
2	TP-02	11/7/2012	11/20/2012	14d
3	TP-03	12/4/2012	12/17/2012	14d
4	TP-04	12/18/2012	1/1/2013	15d
5	TP-05	1/2/2013	1/15/2013	14d
6	TP-06	1/16/2013	1/27/2013	12d
7	TP-07	1/28/2013	1/29/2013	2d
8	TP-08	1/30/2013	2/10/2013	12d
9	TP-09	2/11/2013	2/12/2013	2d
10	TP-10	2/13/2013	2/24/2013	12d
11	TP-11	2/25/2013	2/26/2013	2d
12	TP-12	2/27/2013	3/10/2013	12d
13	TP-13	3/11/2013	3/12/2013	2d
14	TP-14	3/13/2013	3/24/2013	12d
15	TP-15	3/25/2013	3/26/2013	2d

5.3 Test Management Tools

Three tools will be used to organize and document the testing activities for this project. The central tool we will be using to manage this project will be JIRA; with it, we will create tests and directly associate them to their respective user stories. The creation and execution of the tests will be assigned to group members and can be viewed in a to-do style list format. The documentation of traceability matrices, User Stories vs. Use Cases and Test Cases vs. Use Cases, will be done using Microsoft Excel. Microsoft Excel will also be used to document the results of our test execution. Finally, Microsoft Word will be used to back up the data obtained from JIRA.

5.4 Test Slippage

Test slippage is known as the result of testers being unable to meet testing goals (usually due to time constraints). The TFF testing workflow will help see if test slippage has occurred. If slippage has occurred, then it is the responsibility of the developers and testers to report to the stakeholders, in order to establish which defects are tolerable or to reduce the set of features, so as to be able to deliver a viable system [**6**].

5.5 Defects

A defect is defined as improper behaviour in the application caused by either developer error or incorrect workflow. Defects may or may not terminate program execution depending on severity. Defect can cause any of the following: logical inconsistencies, crashes, unwanted side effect. Any defects encountered during the project will be documented in JIRA. Critical defects should be addressed during the testing phase of the current sprint and all other defects are placed in the backlog to be addressed in future sprints. The amount of defects will be used to establish quality metrics [**6**].

Appendix A References

[1] Microsoft. (2012) ASP.NET MVC 3 Testing. [Online]. http://msdn.microsoft.com/en-us/vs2010trainingcourse_aspnetmvc3testing.aspx

[2] The jQuery Foundation. (2012) jQuery; Write less, do more. [Online]. http://jquery.com

[3] W3C. (2012) HTML & CSS. [Online]. http://www.w3.org/standards/webdesign/htmlcss.html

[4] Systeme Evolutif Limited. (2012) TEST PLAN OUTLINE (IEEE 829 FORMAT). [Online]. http://www.computing.dcu.ie/~davids/courses/CA267/ieee829mtp.pdf

[5] Microsoft. (2012) Visual Studio 2010. [Online]. http://msdn.microsoft.com/en-us/library/dd831853(v=vs.100).aspx

[6] K. Anderson, C. Donato, J.Hum, M. Levkovsky, A. Lloyd, P. Modafferi, *"Testing Plan, Cases & Reports"*. Montreal, QC: Concordia University, 2012.