

**Concordia University
Department of Computer Science
and Software Engineering**

Touch For Food

Vision Document

**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

1	Introduction	4
2	Positioning	4
2.1	Problem Statement	4
2.2	Product Position Statement	4
3	Stakeholder Descriptions	5
3.1	Stakeholder Summary	5
3.2	User Environment	5
4	Product Overview	6
4.1	Product Perspective.....	6
4.2	Assumptions and Dependencies	6
4.3	Needs and Features	7
4.4	Alternatives and Competition	8
5	Other Product Requirements	8
6	Appendix A References	9
7	Appendix B Glossary	10

Table of Figures

Figure 3-1	System Overview	6
------------	-----------------------	---

Touch For Food

Vision Document

Version 1.0

Revision History

[illegible]

1 Introduction

The software solution proposed in this document is a web application called Touch For Food (TFF), which is aimed at restaurants to increase their staff efficiency and to provide a higher level of service to their clients using mobile technologies. TFF will use NFC tag technology to allow users to view menus of subscribed restaurants, place orders and request assistance from their waiter/waitress. TFF will also allow restaurants to outsource tasks from staff to customers, personalising a client's experience and extending the reach of their advertising thanks to social media. This document is responsible for: describing the high level goals for TFF, identifying the stakeholders, providing an overview of the desired product and determining basic high level non-functional requirements [1].

2 Positioning

2.1 Problem Statement

The problem of	incorporating the convenience of smartphone technology and social media into a restaurant client's dining experience.
Affects	restaurant clientele with smartphones.
The impact of which is	clients are forced to wait for a waiter to physically appear at their table in order to place food orders, manage bills or request other services from staff. There is also no convenient way for clients to get unbiased up to date statistics about the restaurants they are frequenting.
A successful solution would be	to create an application accessible through a web browser in a mobile device and downloadable as an application for most smartphones. This would allow clients dining at a restaurant to order food, request services from the staff, integrate their dining experience with social networking sites and manage their bills.

2.2 Product Position Statement

For	the general public that own mobile devices.
Who	want to incorporate the convenience of mobile technology into their dining experience.
Touch for Food	is an application
That	will allow clients to place food orders, view food reviews, communicate with staff and manage their bills from their mobile devices.
Unlike	ordering through the waiter/waitress
Our product	is available on a device that clients already carry around with them everywhere. It will allow them to drastically speed up their dining experience by eliminating the need to wait for a waiter/waitress to physically come over to the table in order to communicate a need. Our product will also provide food reviews and statistics from other dinners and the ability to create your own review of the food and/or the restaurant once your dining experience is complete.

3 Stakeholder Descriptions

3.1 Stakeholder Summary

Name	Description	Responsibilities
Customer	A user who uses TFF either at the restaurant or as a social media tool.	-To use TFF to place an order, share the foods they ate, post reviews on meals.
Restaurant Manager	The person who is in charge of a restaurant.	-To provide Cloud9 the data that is necessary for certain aspects of TFF (i.e. menus, calorie count, specials...) -To provide the waiters/customers with the NFC tags/QR codes so TFF can be used in their restaurant.
Waiter	The individual that seats someone at a restaurant, take orders and brings the food.	-To provide the customers with the NFC tags/QR codes to use TFF in the restaurant. -To manage orders and bills.
Dr. Constantine Constantinides	Soen 490 coordinator and instructor.	-To provide timely feedback on the progress of our Capstone project based on our deliverables.
Dr. Olga Ormandjieva	Soen 490 customer stakeholder.	-To provide feedback on our design decisions based on our deliverables.
Online distributor	The platform that will allow TFF to be distributed to users (i.e. App Store, Google Play...)	-To allow users to download TFF on their respective platforms. -To provide a legally binding agreement between the user and TFF.
Team Cloud9	Software engineering students.	-Analyze possible solutions for the TFF software. -To create a software solution based on the designs of TFF. -Ensure a high quality software product that fulfills the needs that TFF describes.

3.2 User Environment

The target market of the TFF application is medium to frequent restaurant goers and the restaurants themselves. TFF consists of two types of tasks (for the restaurant goers) that require different environments. The first task is within a restaurant. Typically it takes 15-30 minutes to place an order at a restaurant and can take 5-10 minutes to get the attention of a waiter. The user can be in a rush (i.e. a business lunch) or just impatient. It may also be difficult to hail a waiter if the restaurant is experiencing a big rush of people. These two key issues make efficient use of the user's time a priority to TFF. TFF can provide an efficient means of communication between a user and his/her waiter. The second environment type is when the user is no longer in a hurry and has some spare time. The user can use TFF to communicate where/what they have eaten with their social circles. This can be at a restaurant or wherever there is a wireless network, but the important aspect is that the user is no longer pressed for time and can review their dining experience.

For the restaurants themselves the environment does not really change. The restaurants will use TFF to receive and respond to messages sent to them by their customers.

TFF is a mobile application, so a wireless network or Wi-Fi is required. Currently the largest mobile platforms are Android, iOS and Windows Phone. It is currently not feasible to develop for Blackberry or any other mobile platforms (i.e. Symbian). However, if the phone has a browser, it may be possible for Blackberry or other smartphone users to use the application via a mobile browser. Possible future integration in Quebec can be with the current major restaurant management system: Maitre'D.

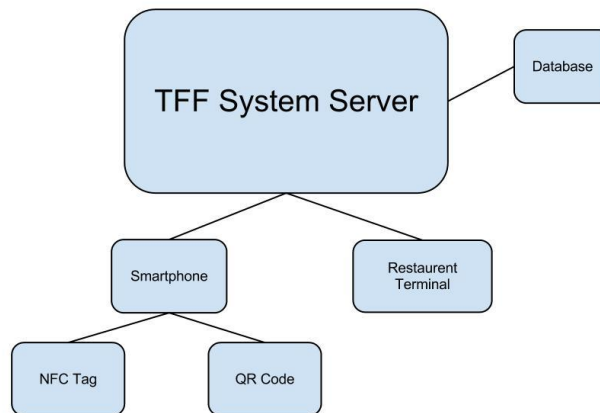


Figure 3-1 System Overview

4 Product Overview

4.1 Product Perspective

Each user will be connected to the TFF system using the QR code or the NFC tag that is placed on the restaurants table. The user can then interact with the system using his smartphone in order to do operations such as ordering food. The TFF system will be expected to do most of the computations on the server side since it is purely a client-server application. The server will be connected to the database, where all the data will be stored. The restaurant terminal will be the gateway for the restaurants to do certain operations on the system such as adding items to the menu and viewing orders placed, among others.

4.2 Assumptions and Dependencies

Assumptions	Dependencies
Internet or Data connection available	The TFF application uses an internet connection to communicate with the TFF system server.
Camera or NFC enabled smartphone	The TFF application can work using only an NFC tag, but if the user does not have an NFC enabled smartphone, a camera is required to scan the QR code.
The restaurant owns a computer connected to the internet	The restaurant terminal requires a computer with an internet connection in order to interact with the TFF system server.

4.3 Needs and Features

Need	Priority	Features	Planned Release (Milestone#)
Order food (customer)	High	Order management, bill management	
Make reservation (customer)	Medium	Food reservation, table reservation	
Call waiter (customer)	Low	Assistance	
Manage personal page (customer)	High	Review order history, managing friends, social networking	
View menu (customer)	High	Menu management	
Comment (customer)	High	Social networking, rating system	
View restaurant stats and reviews (customer)	High	Social networking, rating system	
Accountability, identify priorities for customers (restaurant)	Medium	Customer management	
Manage menu (restaurant)	High	Menu management	
Manage tables (restaurant)	Medium	Table reservation	
Manage bills (restaurant)	High	Bill management	
Restaurant page (restaurant)	High	Social networking, menu management, order management	
Restaurant reporting and statistics (restaurant)	Low	Reporting system	

4.4 Alternatives and Competition

Alternatives or Competition	Benefits	Disadvantages
Open Table	Review reserved table, comment, statistics, location based, smartphone app	Weak social networking, no food ordering/reservation, English language only
Evee	Reserve table, table management	No review, no comment, no food reservation, no smartphone app, not independent application
Google places	Review, comment, social networking, location based, smartphone app	No reservation system for food or table, no table management

5 Other Product Requirements

Architecturally, the TTF application has three main concerns: Speed, security and concurrency. Firstly, the time the mobile applications will take to query the TFF database for menus, orders, etc..... For the application to be useful data transfer should be fairly instant. Application services, such as menu lookup and ordering, should be available 99% of the time when users are connected to the internet through their mobile devices. Secondly, the TFF application may contain sensitive client information, such as food allergies or billing information, as such measures must be taken to protect restaurant clientele. Lastly, restaurants have their limitations in supply, staff and capacity, methods of managing simultaneous demands must be instituted.

TFF should be fairly intuitive; however, a user manual for the TFF application should be available online. The application will be distributed online through an official website and through app markets for Windows, Apple and Android mobile devices. Labeling specifying minimum system requirements must be provided.

6 Appendix A References

- [1] K. Anderson, C. Donato, J. Hum, M. Levkovsky, A. Lloyd, P. Modafferi, "F.S.T.S," Concordia University, Montreal, Canada, Vision Document, v6.16, 2012.

7 Appendix B Glossary

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

**Concordia University
Department of Computer Science
and Software Engineering**

Touch For Food

Supplementary Requirements & Specifications

**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

1	Introduction	6
1.1	Purpose	6
1.2	Scope.....	6
1.3	Definitions, Acronyms and Abbreviations	6
1.4	References.....	6
1.5	Overview.....	6
2	Functionality	6
2.1	Normal User Controls	6
2.2	Administrative Controls.....	6
3	Usability	6
3.1	Required training Time for a Normal User.....	6
3.2	Required Training Time for an Administrative User.....	6
4	Reliability.....	7
4.1	Categorization of Bug Rate.....	7
5	Performance	7
5.1	Response time for ordering food.....	7
5.2	Orders per second	7
5.3	Concurrent orders	8
5.4	Customer Capacity.....	8
5.5	Heavier load performance.....	8
5.6	Native app hard drive space.....	8
5.7	Real Time check-in.....	8
6	Supportability	8
6.1	Coding Standards	8
6.2	Naming Conventions	8
6.3	Maintenance Group Response Time.....	8
7	Design Constraints	9
7.1	Java for Android development.....	9
7.2	Objective-C for iPhone development.....	9
7.3	C# for Windows Phone development	9
7.4	HTML5/CSS3 with .Net for web application front end.....	9
7.5	Iterative development cycles (Agile).....	9
7.6	JIRA for Agile development issue tracking and management.....	9
7.7	SVN with Assembla server for version control	9

8	Online User Documentation and Help System Requirements	9
9	Purchased Components	10
10	Interfaces	10
10.1	User Interfaces	10
10.2	Hardware Interfaces	10
10.3	Software Interfaces	11
10.4	Communications Interfaces	11
11	Licensing Requirements.....	11
11.1	Mobile Licenses	12
11.2	Domain Name	12
11.3	Copyright	12
11.4	Legal disclaimer.....	12
Appendix A	References.....	13
Appendix B	Glossary	14

List of Figures

List of Tables

Table 4-1 Categorization of Bug Rate7

Supplementary Requirements & Specifications

Version 1.0

Revision History

Date	Rev.	Description	Author(s)
2012-09-17	0.0	Document Creation	Katrina Anderson
2012-09-20	0.1	Contributed Purchased Components, Supportability, and Glossary	Josh Hum
2012-09-22	0.2	Added reliability, online user documentation and help system requirements sections	Cristian Asenjo
2012-09-22	0.3	Added interfaces, licensing requirements, and legal, copyright and other notices	Ryan Nasr
2012-09-22	0.4	Added Introduction, Functionality and Usability	Matthew Tam
2012-09-22	0.5	Added Design Constraints and Performance	Patrick Modafferi
2012-09-23	0.6	Added Categorization of Bug Rate to Reliability, updated Glossary	Josh Hum
2012-09-25	0.7	Reviewed Ryan's part	Patrick Modafferi
2012-09-29	0.8	Reviewed Pat's part	Cristian Asenjo
2012-09-29	0.9	Made spelling, grammar and formatting corrections	Josh Hum
2012-09-30	0.10	Reviewed Cristian's part	Matthew Tam
2012-11-04	1.0	Reviewed doc and made minor corrections	Josh Hum

1 Introduction

1.1 Purpose

This document shall list the non-functional, technical, quality, legal, technological and other types of requirements pertaining to TFF. Other important functional requirements will also be included here. These requirements serve as a reference to ensure that the team adheres to performance, quality and other technical standards. It also makes sure that these aspects of the application are thought out and written early in the development process. Please refer to the user stories and the use case models for the major functional requirements.

1.2 Scope

The contents of this document discusses all additional requirements to accompany the use case and vision documents.

1.3 Definitions, Acronyms and Abbreviations

Please see Appendix B, of this document.

1.4 References

Please see Appendix A, References, of this document.

1.5 Overview

The supplementary specifications document is split into thirteen sections. It begins with the Functionality section, which describes the functional requirements of the TFF application in natural language. The next four sections outline the quality attributes of the system: Usability, Reliability, Performance and Supportability. The self-explanatory Design Constraints, Online User Documentation, Purchased Components and Interface requirements sections make up the center portion of the specifications document. The necessary legal and regulatory qualifications are outlined in the Licensing Requirements, Legal, Copyright and Other Notices and Applicable Standards sections, before finishing off with a Glossary of terms, acronyms and abbreviations.

2 Functionality

2.1 Normal User Controls

The TFF application shall allow a normal user to place their food orders to the restaurant, manage how the bill will be separated, call for assistance, manage their personal page by reviewing past meals and adding food friends, leave a customer satisfaction card, and review and recommend restaurants.

2.2 Administrative Controls

The TFF application shall allow an administrator to manage the items on their menus, table reservations, customer orders and bills; customize their restaurant page; view statistics and prioritize customers.

3 Usability

3.1 Required training Time for a Normal User

Since the TFF application is supposed to be intuitive to use, the training time for a normal user will be one day.

3.2 Required Training Time for an Administrative User

The TFF will introduce a new component that will expedite restaurant service. The level of service should not be dropped by using the application; therefore the training time for an administrator will be one day.

4 Reliability

The reliability of this app is dependent on the following factors:

- The Mean Time Between Failure (MTBF) of the web services server.
- The load on the server at any given time.
- The stability and speed of the data connection kept by the mobile device.
- The stability and speed of the Internet connection used by the restaurant.

Of those four factors, only the first two are directly within our control. Issues such as server maintenance and software upgrades can be done during the times the restaurant is closed in order to minimize the downtime of the system's functionality. This MTBF could be further minimized by using server redundancy solutions in order to be better prepared for situations such as hardware and power failures. Taking the previously mentioned issues into account, the MTBF of the system should be no less than 10 months.

Server load must also be taken into account, especially when multiple restaurants and customers are making use of the system at the same time. Load balancing techniques should be considered such as server side HTTP compression and HTTP caching, while on the client side caching and storage of static resources should be maximized in order to reduce requests made to the server.

4.1 Categorization of Bug Rate

Categorization	Description
Minor	A minor bug is categorized as an error that does not hamper the performance of TFF. Most errors in this category will have to do with the application's visual output.
Significant	A significant bug is categorized as an error that will return incorrect or undesirable results. Most errors in this category will be a result of faults in program logic.
Critical	A critical bug is categorized as an error that will cause unrecoverable errors, performance issues, or render parts/all of TFF unusable. Most errors in this category will be a result of hardware crashes or corrupted data.

Table 4-1 Categorization of Bug Rate

5 Performance

5.1 Response time for ordering food

Time elapsed between the customer submitting an order and the order appearing on the restaurant's order list should not exceed 10 seconds.

5.2 Orders per second

The system should be able to accommodate 2-3 orders per second for a brief time interval. For example, if a table of 12 decided to order all at the same time. For the brief 10-15 seconds of the table ordering, there could be multiple transactions per second.

5.3 Concurrent orders

The system should be able to support up to 25 simultaneous orders being made from inside the restaurant. This situation seems unlikely, but large groups of people could collude to arrange a simultaneous order.

5.4 Customer Capacity

The system should be able to handle as many as 200 users browsing concurrently without noticeable loss of performance.

5.5 Heavier load performance

When experiencing heavier than expected traffic (see previous requirements), the system should continue to function but a slowdown of up to 50% should be expected.

5.6 Native app hard drive space

The native application should not occupy more than 20Mb of hard drive space on the user's phone.

5.7 Real Time check-in

When a user checks in to the restaurant, the event should be recorded and displayed on the restaurant owner's side of the software in real time.

6 Supportability

6.1 Coding Standards

- One tab indentation should be done within parenthesis of code blocks for clarity of encased code.
- New lines must be placed between "if", "else if", and "else" statements for better readability.

6.2 Naming Conventions

- All variable names will start with a lowercase letter and use camel casing.
- All method names will start with a lowercase letter and use camel casing.
- All class names will begin with an uppercase letter and use camel casing.
- All constants will be written in all uppercase letters using an underscore to separate words.

6.3 Maintenance Group Response Time

- All minor bugs/defects discovered will be addressed in a future iteration.
- All significant bugs/defects will be addressed on a case by case basis but will be scheduled for an appropriate iteration.
- All critical bugs/defects that are blocking development will be addressed within 48 hours.

Please see section 4.1 of this document for categorization of minor, significant, and critical bugs/defects.

7 Design Constraints

7.1 Java for Android development

When developing a wrapper/version for the application on the Android devices, the programming language enforced by this SDK is Java.

7.2 Objective-C for iPhone development

When developing a wrapper/version for the application on the iPhone devices, the programming language enforced by this SDK is Objective-C.

7.3 C# for Windows Phone development

When developing a wrapper/version for the application on the Windows Phone devices, the programming language enforced by this SDK is C#.

7.4 HTML5/CSS3 with .Net for web application front end

In order to create dynamic pages usable across all mobile devices targeted by the product, a web based application will be built using HTML5 with CSS3 and the .Net framework.

7.5 Iterative development cycles (Agile)

For the type of project being undertaken, it is preferable to have an iterative development cycle. This type of approach allows for modified requirements and progress tracking. Agile satisfies these needs.

7.6 JIRA for Agile development issue tracking and management

In order to better accommodate the previous requirement (Agile), the team will use an online issue tracker called JIRA. Team members will report bugs, assign tasks, log hours and track sprint progress with this tool.

7.7 SVN with Assembla server for version control

Developers will require an Assembla account along with Tortoise SVN software to update, commit and merge revisions of the software source code. This version control is crucial to the development cycle as it allows for many people to work on the same set of files as well as reverting the code to previous versions when needed.

8 Online User Documentation and Help System Requirements

The user documentation for customers will be available online (when accessed via the mobile browser) as well as offline (when accessed via the mobile application itself). For restaurants the user documentation will be available online, when accessed via a web browser, and offline through a PDF file.

The help system will provide an overview of the software based on the user's point of view (customer and restaurant).

The customer's help system will give an overview of the features available on the mobile site as well as the features available on the app. The following help topics should be addressed in the help system:

- Scanning of NFC tags or QR codes
- Creation of a user profile
- Logging in/out of a user profile
- Placing an order

- Contacting a waiter/waitress
- Paying for an order
- Creating a food review

The restaurant's help system should give an overview of the features available on the web site. The following help topics should be addressed in the help system:

- Adding/removing/modifying tables
- Enabling/disabling use of tables
- Adding/removing/modifying menu items
- Enabling/disabling menu items for order
- Adding/removing a TFF user to the blacklist

9 Purchased Components

NFC tags will need to be purchased for testing. iOS developer licenses will also be needed. Since only two members of the team own a Mac, we will only need two licenses. These licenses are provided free for university students. A one-time purchase of Jira and the Jira Agile plugin will also be needed to manage the software development process.

10 Interfaces

10.1 User Interfaces

The core system application will be based on a web application that will run on existing browsers such as Google Chrome, Safari, among others. There will also be wrappers to convert the web application to native applications for the iPhone, Android phones and Windows phones. There will be two main user interfaces, one for the customer and one for the restaurant.

The customer user interface will consist of a section view described by the following:

- Profile management
- Menu viewing
- Order management
- Bill management
- Table reservations
- Social networking section which will include commenting, restaurant statistics and reviews.

The restaurant user interface will consist of:

- Customer management
- Menu management
- Table management
- Restaurant page management
- Order viewing and management
- Reporting and statistics

10.2 Hardware Interfaces

Display and Graphics

A smartphone is necessary to view the TFF customer application. The restaurant will require a screen to view the TFF restaurant application.

Input devices

Smartphone touch screen for the TFF customer application and a keyboard and mouse for the TFF restaurant application.

Database and storage

‘MSSQL’ Database

Connection

The system shall use the available internet connection for collecting data from the users and communicating with the TFF server.

TFF System Server

The server hosting the application shall meet the following requirements:

CPU: Intel Xeon

RAM: Minimum - 1 GB, Recommended - 2GB

Operating System: Windows Server

Connection: Internet

TFF Customer Application

The smartphone using the application shall meet the following requirements:

- Camera or NFC enabled
- iPhone: 3gs and above
- Android: Running Android OS v2.3 and above
- Windows phone: Version 7.0 and above

Additional hardware

The smartphone will be using NFC tags or QR stickers to launch the TFF application on the customers' smartphone.

10.3 Software Interfaces

The TFF system will be communication with ‘MSSQL’ database on the server to store and process information. The system also will require the use of JavaScript on the client's browser. ‘ASP.NET’ will also be used on the client side to display and process user operation. Native applications will also be created for the iPhone, Android and Windows smartphones.

10.4 Communications Interfaces

The system shall use a data or internet connection to communicate with the TFF system server. It will also require NFC communication for NFC enabled smartphones.

11 Licensing Requirements

11.1 Mobile Licenses

The application will require licensing from the Apple developer program for the iPhone native app. It will also require an Android market license for the native Android applications and a Windows market license for the native Windows application.

11.2 Domain Name

The TFF system will require a domain name so that it is publicly accessible.

11.3 Copyright

The application name 'Touch For Food' will be copyrighted. TFF may not be distributed or modified without Cloud 9's consent.

11.4 Legal disclaimer

Cloud 9 is not responsible for allergic reactions or any food related illness.

Appendix A References

Appendix B Glossary

Category	Placeholder for a set of foods fitting under the same grouping or page on the menu (i.e. Breakfast or Desert).
Client	A client of a restaurant who is or can be a potential user of the TFF app
Customer	A customer of a restaurant who is or can be a potential user of the TFF app
Jira	The software process management system that Cloud9 is using
NFC	Near Field Communication
QR Code	A matrix barcode that can be scanned using a smartphone camera
Smartphone	Any mobile phone that has a data connection and internet browser. It must also have NFC capability or the ability to scan QR codes.
Social media	A website that allows interactions between people. Could be Facebook, Touch For Food profile, or any other social networking site.
TFF	Touch For Food
User	Anyone who uses the Touch for Food app. Can be a customer or the restaurant itself.

**Concordia University
Department of Computer Science
and Software Engineering**

Touch For Food

Software Architecture & Design

**SOEN 490
Capstone Project
Fall 2012 – Winter 2013**

Cloud Nine

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

1	Introduction.....	5
1.1	Purpose	5
1.2	Scope.....	5
1.3	Definitions, Acronyms and Abbreviations	5
1.4	References.....	5
1.5	Overview.....	5
2	Architectural Representation.....	5
2.1	Use Case View.....	5
2.2	Logical View.....	6
2.3	Process View	6
2.4	Physical/Deployment View	6
3	Architectural Goals and Constraints	6
3.1	3-Tier Architecture vs MVC.....	7
4	Use Case View	7
4.1	Actor Goal List	7
4.2	Use Case Model.....	8
4.3	Fully Dressed Use Cases.....	10
5	Logical View.....	19
5.1	Operational Contracts	19
5.2	Domain Model.....	19
5.3	Class Diagram.....	20
5.4	System Sequence Diagrams.....	22
5.5	Sequence Diagram	22
5.6	Communication Diagrams	23
6	Process View.....	24
6.1	Activity Diagrams.....	24
7	Physical View	27
7.1	Deployment Diagram.....	27
8	Size & Performance	27
9	Quality.....	28
Appendix A	References.....	29
Appendix B	Glossary	30

List of Figures

Figure 4-1 User Use Case Model.....	8
Figure 4-2 Restaurant Use Case Model	9
Figure 5-1 Domain Model	19
Figure 5-2 Class Diagram	21
Figure 5-3Populate menu Sequence Diagram.....	22
Figure 5-4 Collaboration Diagram for populating a menu from scratch	23
Figure 5-5 Collaboration Diagram for populating menu with pre existing cetegories and items.....	23
Figure 6-1 View Menu.....	24
Figure 6-2 Add Food Item to Order	25
Figure 6-3 Remove Food Item from Order	26
Figure 7-1 Deployment Diagram	27

List of Tables

Table 2-1 Diagram types used for each view.....	6
Table 4-1 Actor Goal List	7

Touch For Food

Software Architecture & Design

Version 1.6

Revision History

Date	Rev.	Description	Author(s)
2012-09-17	0.0	Document Creation	Katrina Anderson
2012-09-26	0.1	Contributed to Section 3 Architectural Goals and Constraints	Mikhail Levkovsky
2012-09-26	0.2	Contributed to Section 10 Quality	Mikhail Levkovsky
2012-09-28	0.3	Contributed to Section 1 and 2	Patrick Modafferi
2012-09-29	0.4	Contributed to Section 8 (Physical View) and 9 (Size & Performance)	Cristian Asenjo
2012-09-30	0.5	Contributed to Section 4 (Actor Goal List, Fully Dressed UC2.1 – UC2.4, Use Case Model)	Cynthia Donato
2012-09-30	0.6	Contributed to Section 4 UC1.1 Formatted section. Added Table of Figures.	Katrina Anderson
2012-10-01	0.7	Contributed to Section 5.2 and 5.3. added the figures to Table of Figures	Christian Daher/Ryan Nasr
2012-10-14	0.8	Contributed to Section 7.1	Matthew Tam
2012-10-20	1.0	Updated to sprint 1Contributed section 4.3	Patrick Modafferi/Mikhail Levkovsky
2012-10-26	1.1	Added Sequence Diagrams Populate Menu and updated actor goal list	Mikhail Levkovsky
2012-10-27	1.2	Added Collaboration Diagrams Populate Menu	Patrick Modafferi
2012-10-31	1.3	Added Use Case for updating customer profile. Fixed numbering for use case 3.1	Cynthia Donato
2012-11-03	1.4	Added Use Case for creating a user profile. Renumbered section 4.3 to have the user profile creation first and the user profile updating second.	Cristian Asenjo
2012-11-03	1.5	Added Use Case for login to personal profile. Renumbered section 4.3 to have the user login first and the user profile updating second	Matthew Tam
2012-11-04	1.6	Reviewed and made minor corrections	Josh Hum

1 Introduction

1.1 Purpose

The purpose of this document is determining on an architectural level how each part of the TFF application will work. These architectural diagram can represent inner workings of the system or higher level design. When it comes to implementing, the coder should refer to this document to save time and maintain consistency. Creating this document could be time consuming, but in the end, having clear documentation for the design and architecter has many advantages.

With these artifacts we can think of potential problems and identify design concerns before coding. This would improve overall quality of the system because ides and thoughts are properly organized. There are many types of diagrams some help determine the flow of the system and some explain how components interact with one another. The domain model and class diagram shows how objects are related. Finally, some diagrams show more physical elements like the deployment diagram.

Another reason for this documentation is scalability, mantainability and resusability. In order to allow for someone to potentially expand on this project, reuse it for something else or maintain the finished product, the documentation must be able to serve as a road map or a guide for any future developer.

1.2 Scope

The scope of the architecture artifacts cover the main components of the system. Each iteration, the components planned for that sprint, will be designed. The resulting architecture and design diagrams will be added to the document. In this way, the document will incrementally be built up to a final document containing diagrams for every component in the system.

1.3 Definitions, Acronyms and Abbreviations

Please see Appendix B, of this document.

1.4 References

Please see Appendix A, References, of this document.

1.5 Overview

The document is organized into each of the 5 main views for architecture; Use Case, Logical, Development, Process and Physical (See section 2 for more details). As needed, other items will be discussed such as Quality, Size and Performance. It will also discuss architecture styles and constraints as well as a comparative analysis to justify some decision that were taken. As a whole, the document should completely represent all the architecture of the system.

2 Architectural Representation

The architecture being used for this web application is a 3-tire architecture. See section 3 below for more details about why this architecture is chosen and what it provides the system with.

Whichever diagram is needed for each component will be created and added to this document. Here is a list of every type of diagram broken down into each of the 4+1 views:

2.1 Use Case View

Represents important requirements through fully detailed use cases.

2.2 Logical View

Determines how the layers, packages, classes and other software elements are organized in the system.

2.3 Process View

Illustrates processes and threads in the system. Could show how some elements interact and collaborate throughout a process.

2.4 Physical/Deployment View

Physically represents the components, processes, communications and important structures of the system.

Table 2-1 Diagram types used for each view

Use Case	Logical	Process	Physical
Actor-Goal List	Operational Contracts	Activity Diagrams	Deployment Diagram
Use Case Diagrams	Domain Model		
Use Cases	Class Diagram		
	System Sequence Diagrams		
	Sequence Diagrams		
	Communication Diagrams		

3 Architectural Goals and Constraints

TFF will be designed using a 3-tier client-server architecture. The reason for choosing a 3-tier architecture is to achieve:

- Decoupling of presentation, data and domain logic
- Allows for multiple people to work on different parts of the same project
- Promotes low coupling between different components
- Promotes organization and code reuse
- Promotes consistent and well defined interfaces between each layer

A well structured 3-tier architecture will also allow us to swap out or update any layer independently of the others. In order to achieve an even higher level of abstraction and separation of concerns the 3 tiers can themselves be layered and separated (a more general term of this architecture would then be n-tier architecture).

The 3 tiers consist of:

- Client – in our case it will be a thin client with little to no application logic
- Domain – contains the bulk of the domain logic
- Data – persistence layer with the user information

Some constraints of the 3-tier architecture are:

- Tracing end-to-end flows of data can be challenging as the system grows
- The requirement of having a network limits the areas of use of the application
- Inherit constraints of the network (i.e. slow connections, low bandwidth)

3.1 3-Tier Architecture vs MVC

Since our application necessitates a network connection and the physical separation of the machines that TFF is running on, an MVC architecture was not suited for the task. The entire TFF application needs to take into account the physical separation between all of the layers and while this is achievable in MVC, it is not the main focus of this architecture. MVC might still be incorporated within each layer of the 3-tier architecture, but cannot represent the system as a whole.

4 Use Case View

4.1 Actor Goal List

Table 4-1 Actor Goal List

Actor	Goal
User	<ol style="list-style-type: none">1. Add Food Item to Order2. View Order3. Remove Food Item from Order4. Place Food Order5. Create User Profile6. Login to User Profile7. Update Customer Settings
Restaurant	<ol style="list-style-type: none">1. Populate Menu

4.2 Use Case Model

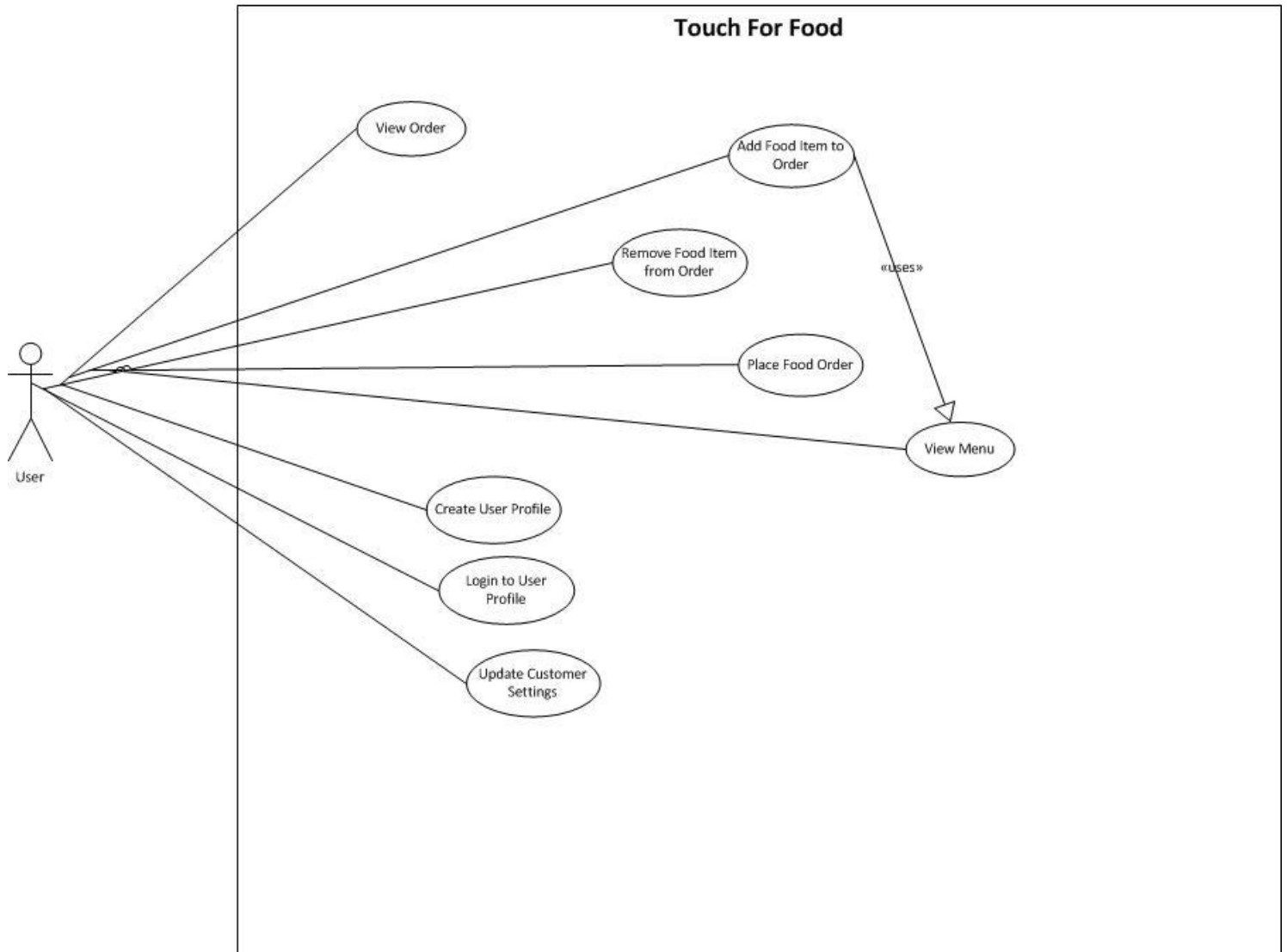


Figure 4-1 User Use Case Model

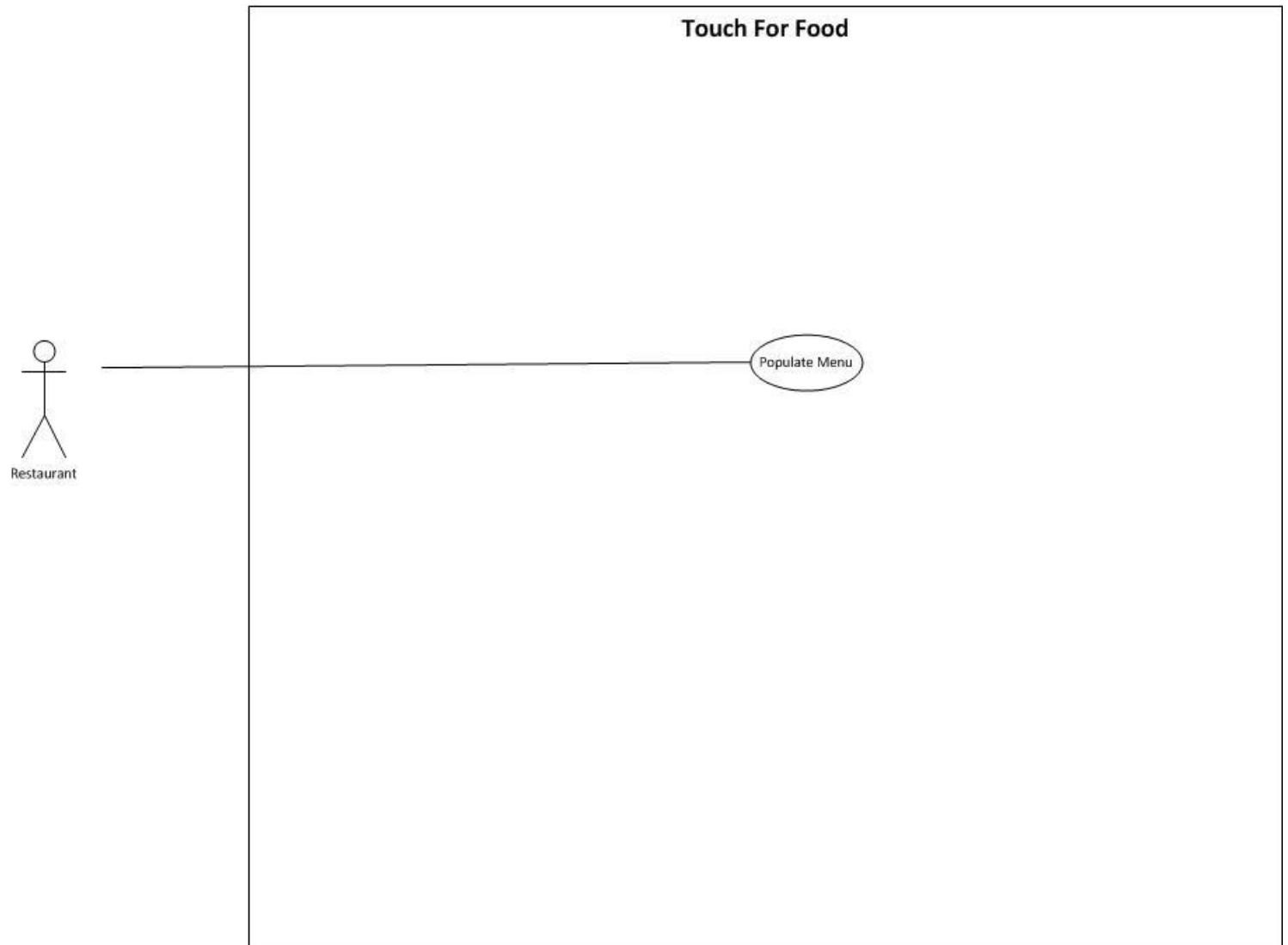


Figure 4-2 Restaurant Use Case Model

4.3 Fully Dressed Use Cases

UC1.1 View Menu

ID: UC1.1

Use Case: View Menu

Description: Restaurant client wishes to view all menu options.

Level: User Goal

Primary Actor: User

Supporting Actors: None

Stakeholders and Interests:

User: Their interest is to view their meal and beverage options for a restaurant.

Restaurant: Their interest is for clients to see meal and beverage options.

Pre-Conditions:

1. The user is logged into their Touch for Food account.
2. The user has scanned the NFC tag or QR code at the given restaurant.

Post Conditions:

Success end condition:

1. All menu options are displayed

Failure end condition:

1. No menu options are displayed
2. An error message is displayed

Minimal Guarantee:

1. An informative message is displayed to the user.

Main Success Scenario:

1. User initiates restaurant page search
2. System returns restaurant page
3. User navigates to the menu section of the restaurant page.
4. System displays restaurant menu in order of course.

Extensions:

4.a. Client chooses to sort menu by price

- 4.a.1 User selects to have menu items appear in order of price.
- 4.a.2 System displays restaurant menu items in order of price

4.b Client chooses to sort menu by popularity

- 4.b.1 User selects to have menu items appear in order of popularity.
- 4.b.2 System displays restaurant menu items in order of popularity

Special Requirements:

See Section 2 Functionality of Supplementary Specifications document

See Section 3 Usability of Supplementary Specifications document

See Section 4 Reliability of Supplementary Specifications document

See Section 5 Performance of Supplementary Specifications document

See Section 6 Supportability of Supplementary Specifications document

See Section 7 Design Constraints of Supplementary Specifications document

See Section 10 Interfaces of Supplementary Specifications document

UC2.1 Add Food Item to Order

ID:UC2.1

Use Case:Add Food Item to Order

Description:User wishes to add a menu item to their order.

Level:User-goal

Primary Actor:User

Secondary Actor: None

Stakeholders and Interests:

User: Their interest is to add a single menu item to their order.

Restaurant: Their interest is to receive an accurate and detailed food order from the customer.

Pre-Conditions:

1. The user is logged into their Touch for Food account.
2. The user is physically present at a restaurant registered with Touch for Food.
3. The user has scanned the NFC tag or QR code at the given restaurant.

Post Conditions:

Success end condition

1. A food item is added to the order along with its desired quantity.

Failure end condition:

1. The order remains unchanged.
2. The user is notified that the system failed to add the food item.

Minimal Guarantee

1. An informative message is displayed to the user.

Main Success Scenario:

1. User navigated to the View Menu section. See UC1.1
2. System displays the appropriate restaurant's menu.
3. User selects a food item and indicated that they would like to add it to their order.
4. System requests that the user provide a desired quantity.
5. User specifies the quantity.
6. System notifies the user that the item has been added to their order.

Extensions:

6.a The food item is no longer available

- 6.a.1 System notifies the user that the restaurant does not have enough food to fulfill their request.
- 6.a.2 System provide options for conflict resolution
 - 6.a.2.1 System provides the option to order a lesser quantity if available.
 - 6.a.2.2 System suggests a similar dish that is available.

Special Requirements:

- See Section 2 Functionality of Supplementary Specifications document
- See Section 3 Usability of Supplementary Specifications document
- See Section 4 Reliability of Supplementary Specifications document
- See Section 5 Performance of Supplementary Specifications document
- See Section 6 Supportability of Supplementary Specifications document
- See Section 7 Design Constraints of Supplementary Specifications document
- See Section 10 Interfaces of Supplementary Specifications document

UC2.2 View Food Order

ID:UC2.2

Use Case:View Food Order

Description:User wishes to view their current order.

Level:User-goal

Primary Actor:User

Secondary Actor: None

Stakeholders and Interests:

User: Their interest is to view their current order.

Restaurant: Their interest is to receive an accurate and detailed food order from the customer.

Pre-Conditions:

1. The user is logged into their Touch for Food account.

The user is physically present at a restaurant registered with Touch for Food.

The user has scanned the NFC tag or QR code at the given restaurant.

The user has at least one menu item in their order.

Post Conditions:

Success end condition

1. The order is accurately displayed.

Failure end condition:

1. The order remains unchanged.
2. The user is notified that the system failed to retrieve their order.

Minimal Guarantee

1. An informative message is displayed to the user.

Main Success Scenario:

1. User navigated to the View Order section.
2. System displays the users order.

Extensions:

N/A

Special Requirements:

See Section 2 Functionalityof Supplementary Specifications document

See Section 3 Usabilityof Supplementary Specifications document

See Section 4 Reliabilityof Supplementary Specifications document

See Section 5 Performanceof Supplementary Specifications document

See Section 6 Supportabilityof Supplementary Specifications document

See Section 7 Design Constraintsof Supplementary Specifications document

See Section 10 Interfacesof Supplementary Specifications document

UC2.3 Remove Food Item From Order

ID:UC2.3

Use Case:Remove Food Item from Order

Description:User wishes to remove an item from their order.

Level:User-goal

Primary Actor:User

Secondary Actor: None

Stakeholders and Interests:

User: Their interest is to remove a single menu item from their order.

Restaurant: Their interest is to receive an accurate and detailed food order from the customer.

Pre-Conditions:

1. The user is logged into their Touch for Food account.
2. The user is physically present at a restaurant registered with Touch for Food.
3. The user has scanned the NFC tag or QR code at the given restaurant.
4. The user has at least one menu item in their order.

Post Conditions:

Success end condition

1. A food item is removed from the order.

Failure end condition:

1. The order remains unchanged.
2. The user is notified that the system failed to remove the food item.

Minimal Guarantee

1. An informative message is displayed to the user.

Main Success Scenario:

1. User navigated to the View Order section. See UC2.3.
2. System displays the users order.
3. User selects a food item and indicated that they would like to remove it from their order.
4. System requests that the user confirms the removal of the item.
5. User confirms they would like to remove the item.

Extensions:

N/A

Special Requirements:

See Section 2 Functionalityof Supplementary Specifications document

See Section 3 Usabilityof Supplementary Specifications document

See Section 4 Reliabilityof Supplementary Specifications document

See Section 5 Performanceof Supplementary Specifications document

See Section 6 Supportabilityof Supplementary Specifications document

See Section 7 Design Constraintsof Supplementary Specifications document

See Section 10 Interfacesof Supplementary Specifications document

UC2.4 Place Food Order

ID:UC2.4

Use Case:Place Food Order

Description:User wishes to finalize their order and place it with the restaurant.

Level:User-goal

Primary Actor:User

Secondary Actor: None

Stakeholders and Interests:

User: Their interest is to place an accurate order quickly and effortlessly.

Restaurant: Their interest is to receive an accurate and detailed food order from the customer.

Pre-Conditions:

1. The user is logged into their Touch for Food account.
2. The user is physically present at a restaurant registered with Touch for Food.
3. The user has scanned the NFC tag or QR code at the given restaurant.

Post Conditions:

Success end condition

1. The food order is conveyed to the restaurant staff.

Failure end condition:

1. The existing draft order remains unchanged.
2. The user is notified that the system failed to convey it to the restaurant staff.

Minimal Guarantee

2. An informative message is displayed to the user.

Main Success Scenario:

1. User navigated to the View Order section. See UC2.3.
2. System displays the users order.
3. User indicates that they would like to place their order with the restaurant.
4. System notifies the user that the order has been placed successfully.

Extensions:

3.a The user indicates that they would like to clear their current order

- 3.a.1 System requests confirmation from the user to clear their current order.
- 3.a.2 The user confirms that they would like to clear their current order.

Special Requirements:

See Section 2 Functionalityof Supplementary Specifications document

See Section 3 Usabilityof Supplementary Specifications document

See Section 4 Reliabilityof Supplementary Specifications document

See Section 5 Performanceof Supplementary Specifications document

See Section 6 Supportabilityof Supplementary Specifications document

See Section 7 Design Constraintsof Supplementary Specifications document

See Section 10 Interfacesof Supplementary Specifications document

UC3.1 Populate Menu

ID:UC3.1

Use Case:Populate Menu

Description:User wishes to add categories and items to a menu.

Level:User-goal

Primary Actor:Client

Secondary Actor: None

Stakeholders and Interests:

User: Their interest is to be able to see what items are available for them to order

Restaurant: Their interest is to be able to create fully customized menus

Pre-Conditions:

1. The client is logged in to their TFF account
2. The system contains at least one menu with a default category
3. The client has navigated to the menu editor
4. The client has selected the option to create/edit a menu

Post Conditions:

Success end condition

1. The system saves the menu with at least one category and one menu item

Failure end condition:

1. The existing draft order remains unchanged.
2. The user is notified that the system failed and to contact support.

Minimal Guarantee

1. An informative message is displayed to the user.

Main Success Scenario:

1. The system takes the client to the selected menu page that contains at least the default category.
2. Client adds one or more categories to the menu.
3. Client chooses a category they would like to add items to
4. The system displays a list of existing menu items in the selected category.
5. Client adds one or more items to the menu's category.
6. Steps 2 to 5 can be repeated as much as needed
7. Client selects to finalize
8. The system saves the menu with all modifications

Extensions:

2.a The client chooses to create a new category

- 3.a.1 System displays a category creation form.
- 3.a.2 The user enters a name for the category and confirms the creation.
- 3.a.3 The system saves the category and adds it to the menu.

2.b The user selects a category from a list of previously created ones

- 3.a.1 System adds the selected category to the menu.

5.a The client chooses to create a new menu item

- 3.a.1 System displays a menu item creation form.
- 3.a.2 The user enters a name, description and price for the menu item and confirms the creation.
- 3.a.3 The system saves the category and adds it to the menu.

5.b The user selects a menu item from a list of previously created ones

- 3.a.1 System adds the selected category to the menu.

Special Requirements:

- See Section 2 Functionality of Supplementary Specifications document
See Section 3 Usability of Supplementary Specifications document
See Section 4 Reliability of Supplementary Specifications document
See Section 5 Performance of Supplementary Specifications document

See Section 6 Supportability of Supplementary Specifications document
See Section 7 Design Constraints of Supplementary Specifications document
See Section 10 Interfaces of Supplementary Specifications document

UC4.1 Create User Profile

ID:UC4.1

Use Case:Create User Profile

Description:User wishes to create a user profile.

Level:User-goal

Primary Actor:User

Secondary Actor: None

Stakeholders and Interests:

User: Their interest is to create a user profile.

TFF: Our interest is to be sure that users can easily and efficiently create a user profile.

Pre-Conditions:

1. The user is at the Create User Profile page.

Post Conditions:

Success end condition

1. The user profile is created correctly.

Failure end condition:

1. The user profile is not created.
2. The user is notified that the system failed to create a new user profile.

Minimal Guarantee

1. An informative message is displayed to the user.

Main Success Scenario:

1. User navigated to the Create User Profile section.
2. System displays an empty user information form.
3. User fills out the user information form and indicates it would like to confirm the creation of the profile.
4. System notifies the user that the profile was created.

Extensions:

3.a The user indicates that they would like to cancel without creating the profile

- 3.a.1 System redirects to the main Touch For Food page.

Special Requirements:

See Section 2 Functionality of Supplementary Specifications document
See Section 3 Usability of Supplementary Specifications document
See Section 4 Reliability of Supplementary Specifications document
See Section 5 Performance of Supplementary Specifications document
See Section 6 Supportability of Supplementary Specifications document
See Section 7 Design Constraints of Supplementary Specifications document
See Section 10 Interfaces of Supplementary Specifications document

UC4.2 Login to Personal Profile

ID:UC4.2

Use Case:Login to Personal Profile

Description:User wishes to log in to their personal profile.

Level:User-goal

Primary Actor:User

Secondary Actor: None

Stakeholders and Interests:

User: Their interest is to view their personal profile.

TFF: Our interest is to be sure that users can easily and efficiently access their personal profile.

Pre-Conditions:

1. The user is at the User Login page.

Post Conditions:

Success end condition

1. The user's personal profile page is displayed.

Failure end condition:

1. The user's personal profile is not displayed.
2. The user is notified that the system failed to display their personal profile.

Minimal Guarantee

1. An informative message is displayed to the user.

Main Success Scenario:

1. User navigates to the User Login section.
2. System displays two text fields for username and password.
3. User inputs their username and password and attempts to login.
4. System verifies the username and password.
5. User views their personal profile.

Extensions:

3.a The user inputs the wrong username and/or password

- 3.a.1. System verifies the username and password.
- 3.a.2. System redirects user to the user login page with an error message: "Username or Password is incorrect, please enter your credentials again".

3.b The user forgets their password

- 3.b.1. User clicks on the "Forgot Password" link
- 3.b.2. System redirects user to the Forgot Password section
- 3.b.3. System displays options to recover their password

3.c The user wants to create an account

- 3.c.1. Refer to UC4.1 Create User Profile

Special Requirements:

- See Section 2 Functionality of Supplementary Specifications document
- See Section 3 Usability of Supplementary Specifications document
- See Section 4 Reliability of Supplementary Specifications document
- See Section 5 Performance of Supplementary Specifications document
- See Section 6 Supportability of Supplementary Specifications document
- See Section 7 Design Constraints of Supplementary Specifications document
- See Section 10 Interfaces of Supplementary Specifications document

UC4.3 Update Customer Settings

ID:UC4.3

Use Case:Update Customer Settings

Description:User wishes to update their personal profile.

Level:User-goal

Primary Actor:User

Secondary Actor: None

Stakeholders and Interests:

User: Their interest is to update their personal profile.

TFF: Our interest is to be sure that users can easily and efficiently update their profile.

Pre-Conditions:

1. The user is logged into their Touch for Food account.

Post Conditions:

Success end condition

1. The user profile is update correctly.

Failure end condition:

1. The account user account information remains unchanged.
2. The user is notified that the system failed to update their profile.

Minimal Guarantee

1. An informative message is displayed to the user.

Main Success Scenario:

1. User navigated to the Edit Profile section.
2. System displays an editable version of the user profile.
3. User makes necessary updated to name, email and/or display name and indicates that they would like ot save the changes.
4. System notifies the user that the profile has been updated.

Extensions:

3.a The user indicates that they would like to cancel without saving

- 3.a.1 System redirects to the user profile page.

Special Requirements:

See Section 2 Functionalityof Supplementary Specifications document

See Section 3 Usabilityof Supplementary Specifications document

See Section 4 Reliabilityof Supplementary Specifications document

See Section 5 Performanceof Supplementary Specifications document

See Section 6 Supportabilityof Supplementary Specifications document

See Section 7 Design Constraintsof Supplementary Specifications document

See Section 10 Interfacesof Supplementary Specifications document

5 Logical View

5.1 Operational Contracts

5.2 Domain Model

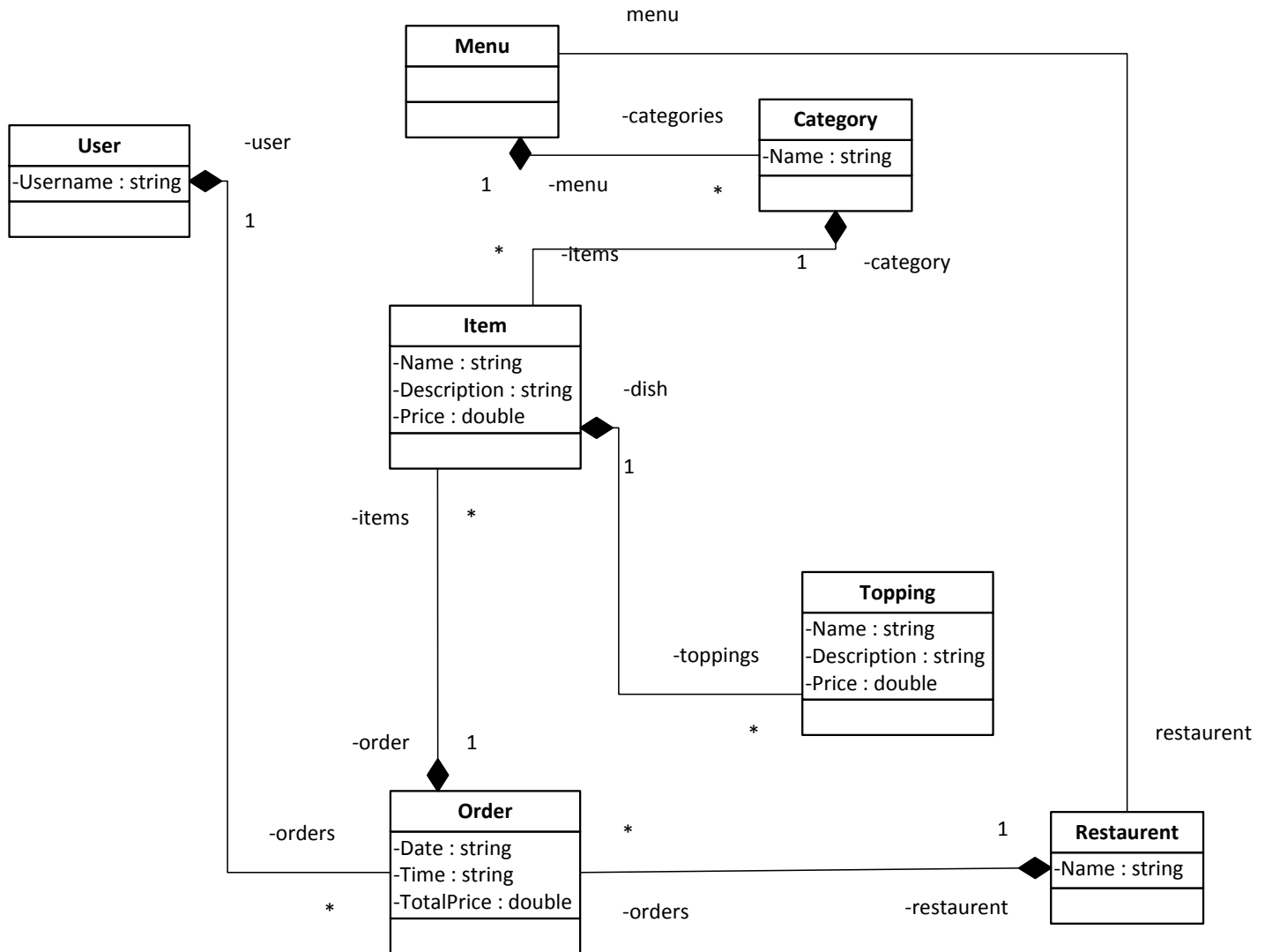
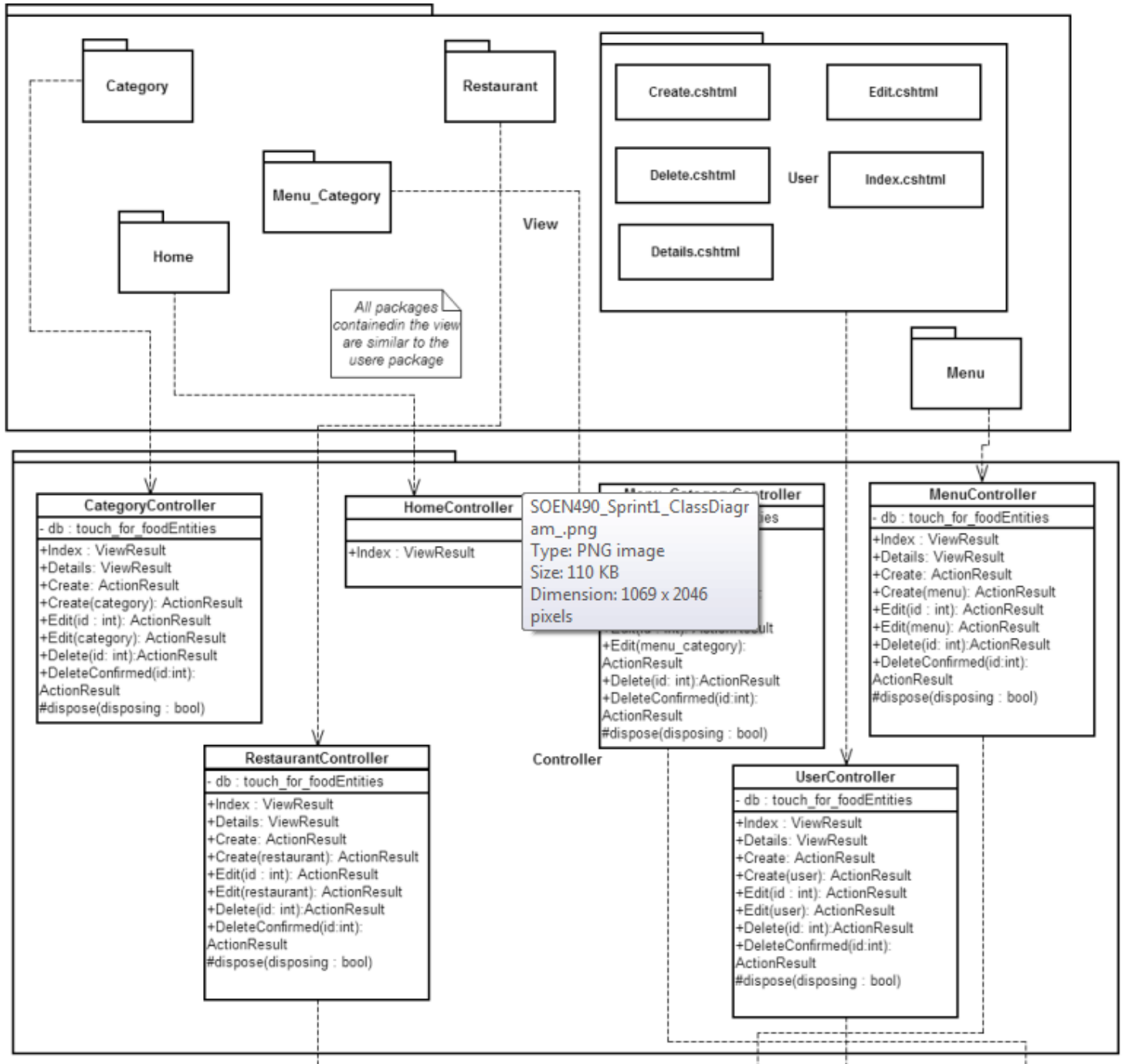


Figure 5-1 Domain Model

5.3 Class Diagram



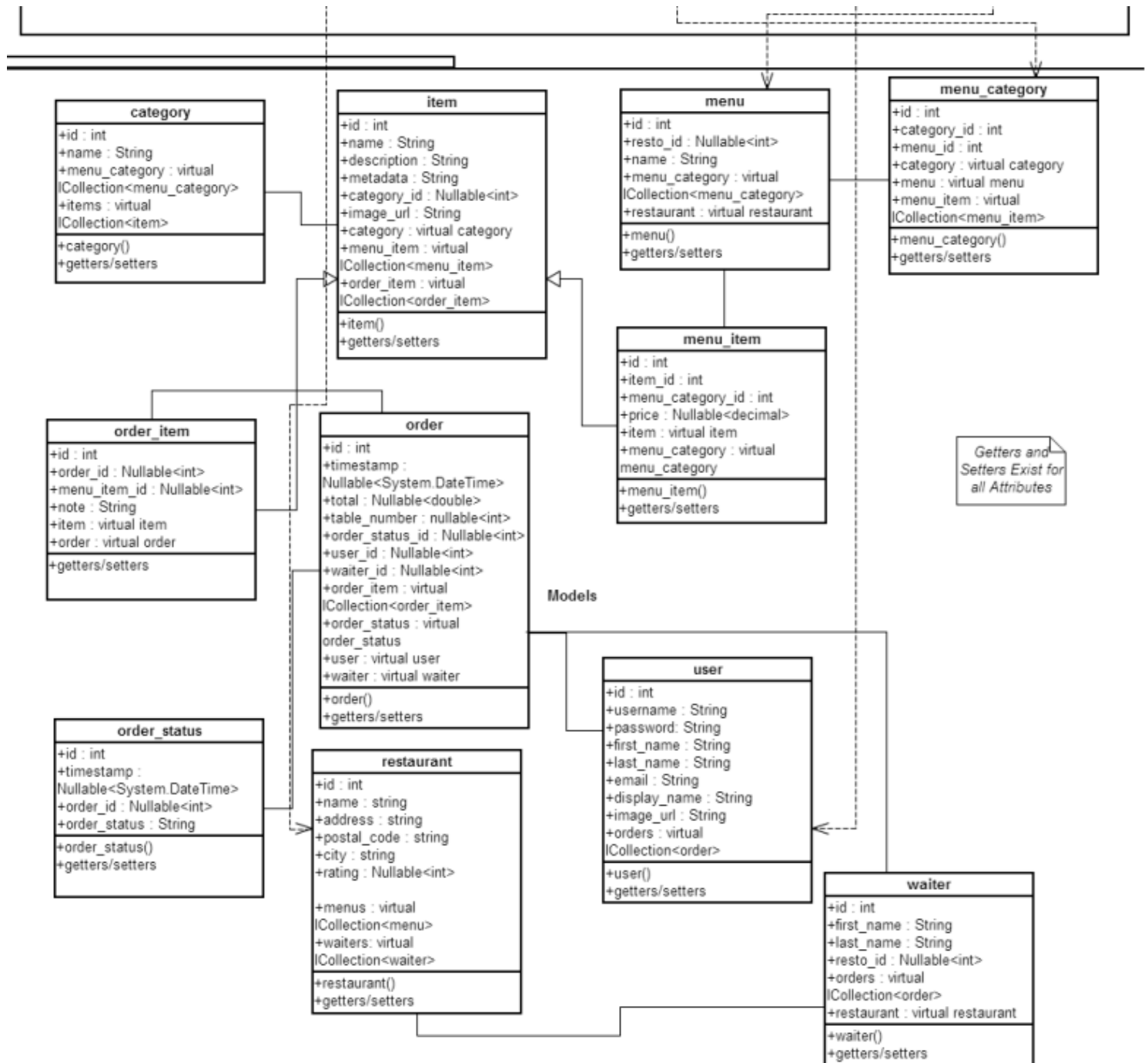
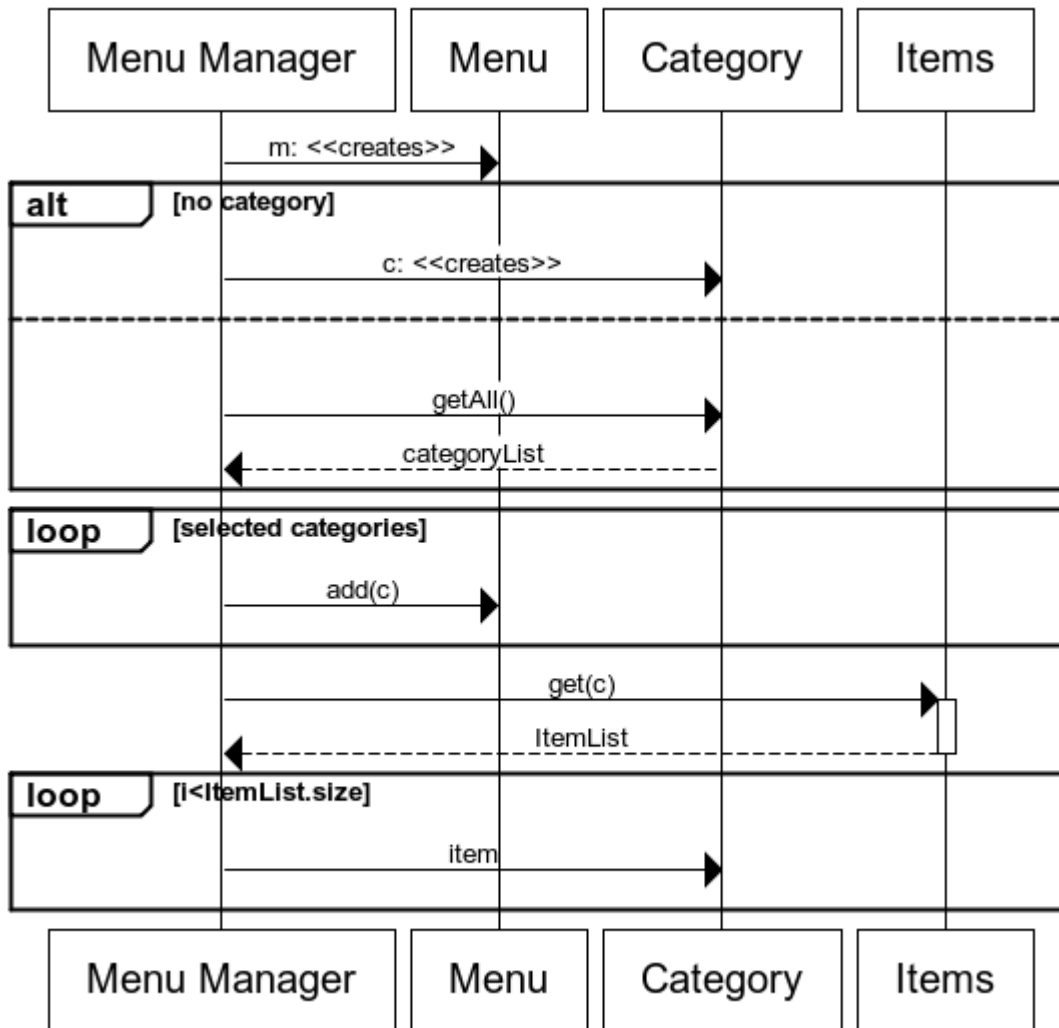


Figure 5-2 Class Diagram

5.4 System Sequence Diagrams

5.5 Sequence Diagram



www.websequencediagrams.com

Figure 5-3Populate menu Sequence Diagram

5.6 Communication Diagrams

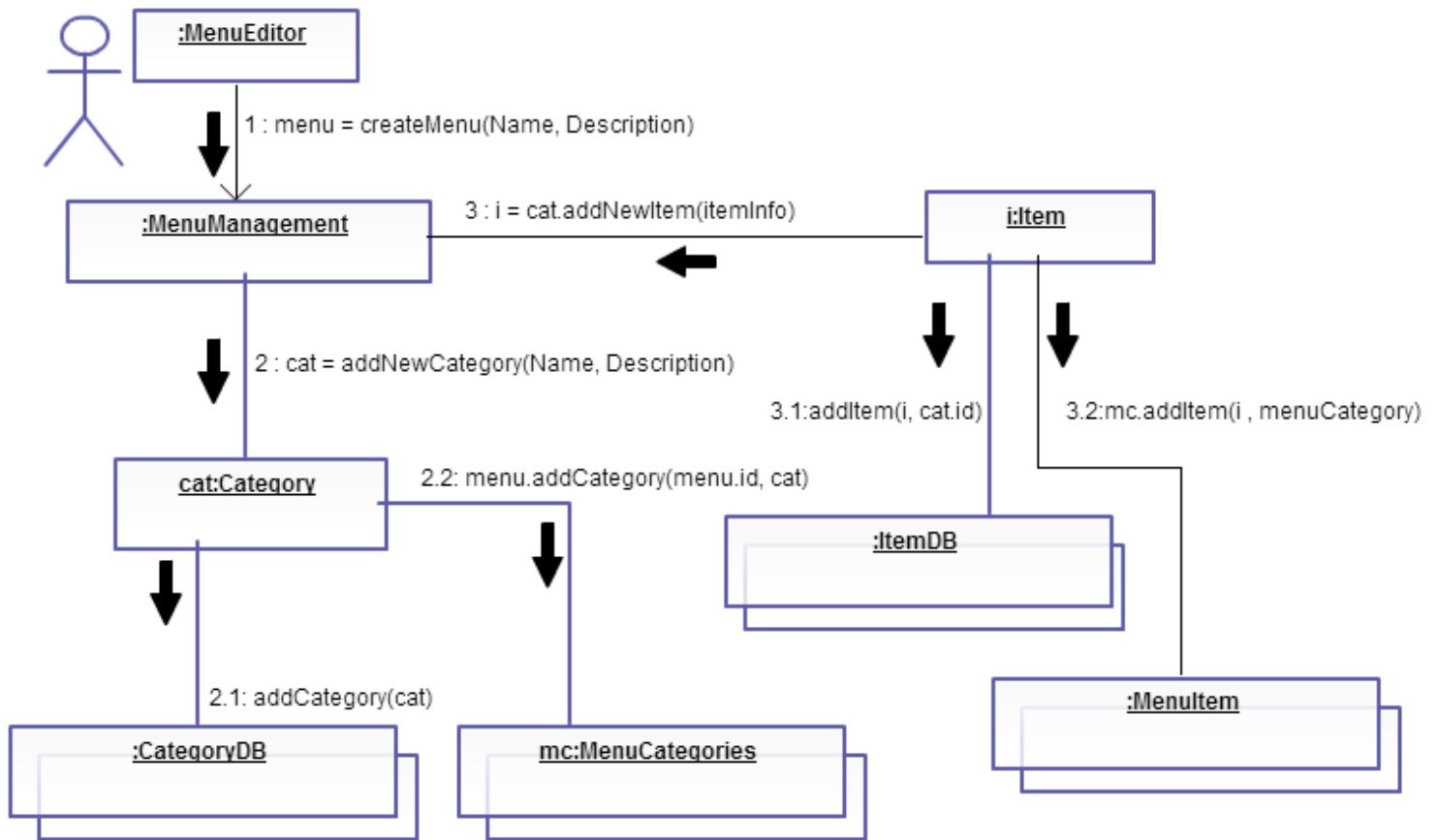


Figure 5-4 Collaboration Diagram for populating a menu from scratch

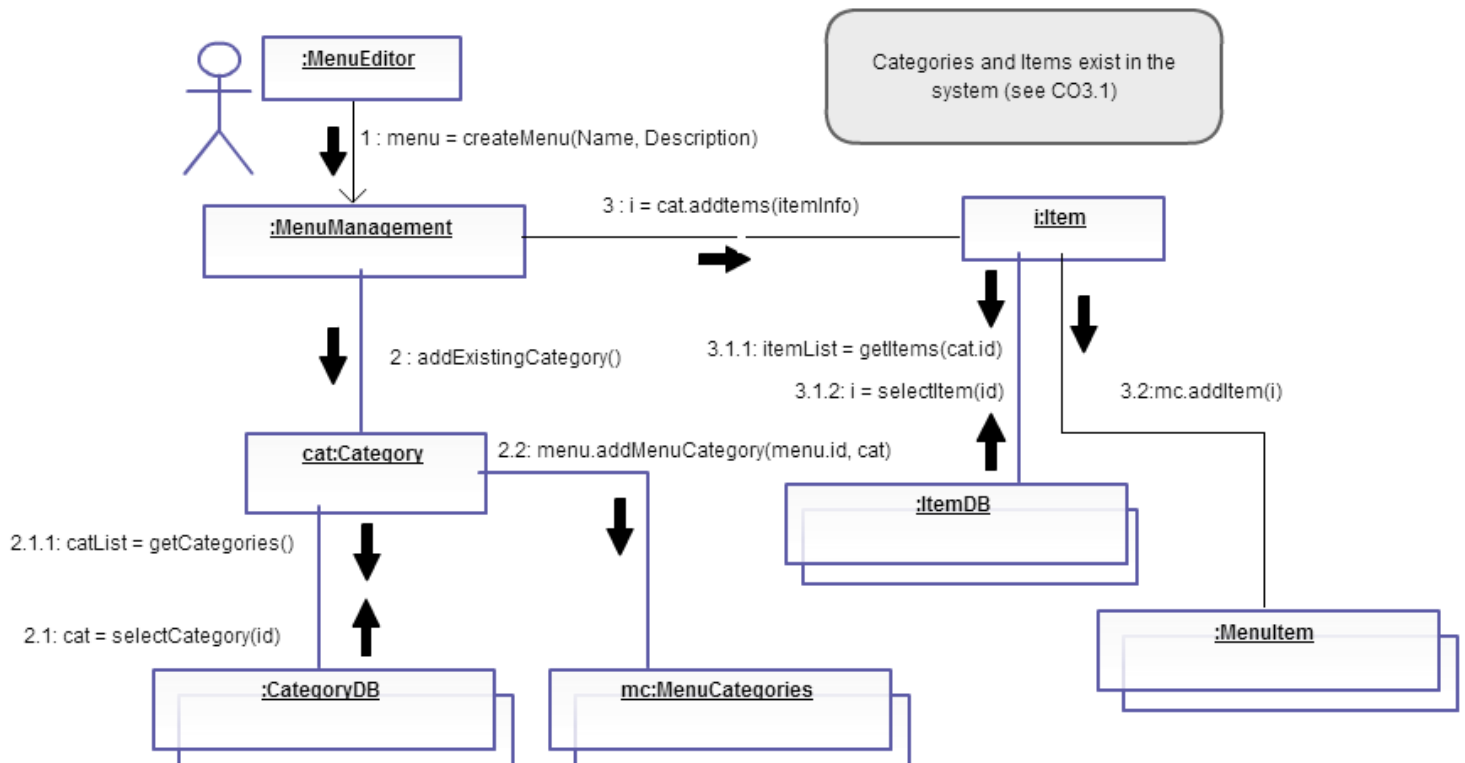


Figure 5-5 Collaboration Diagram for populating menu with pre existing categories and items

6 Process View

6.1 Activity Diagrams

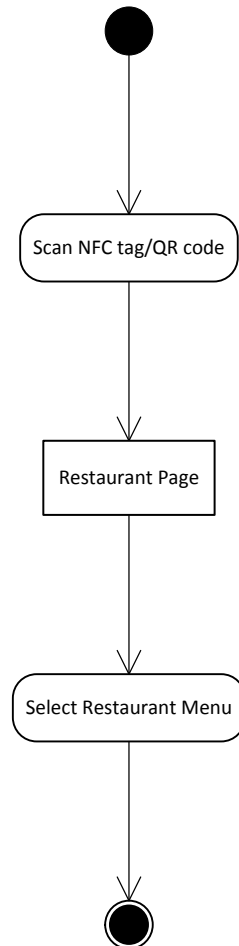


Figure 6-1 View Menu

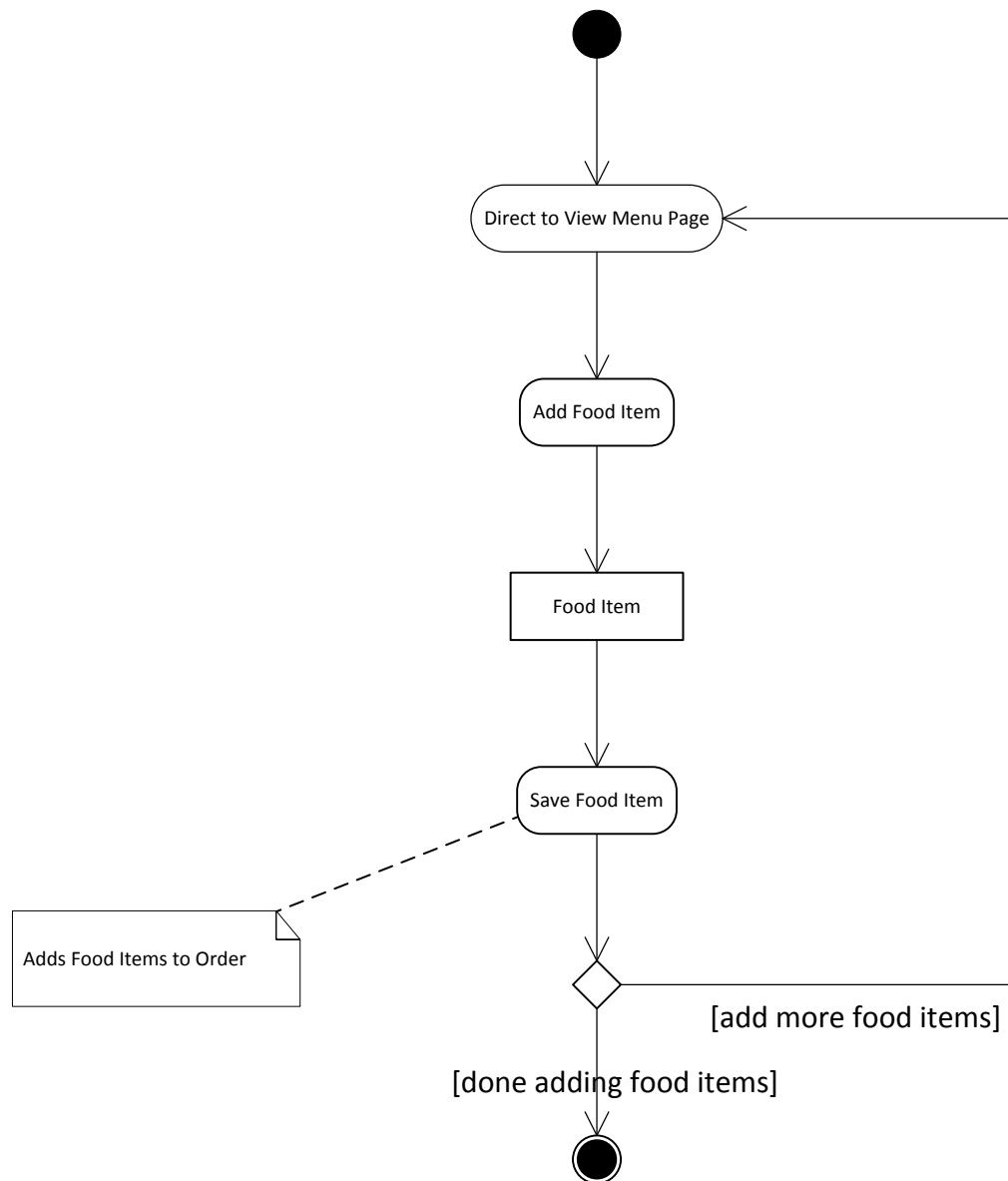


Figure 6-2 Add Food Item to Order

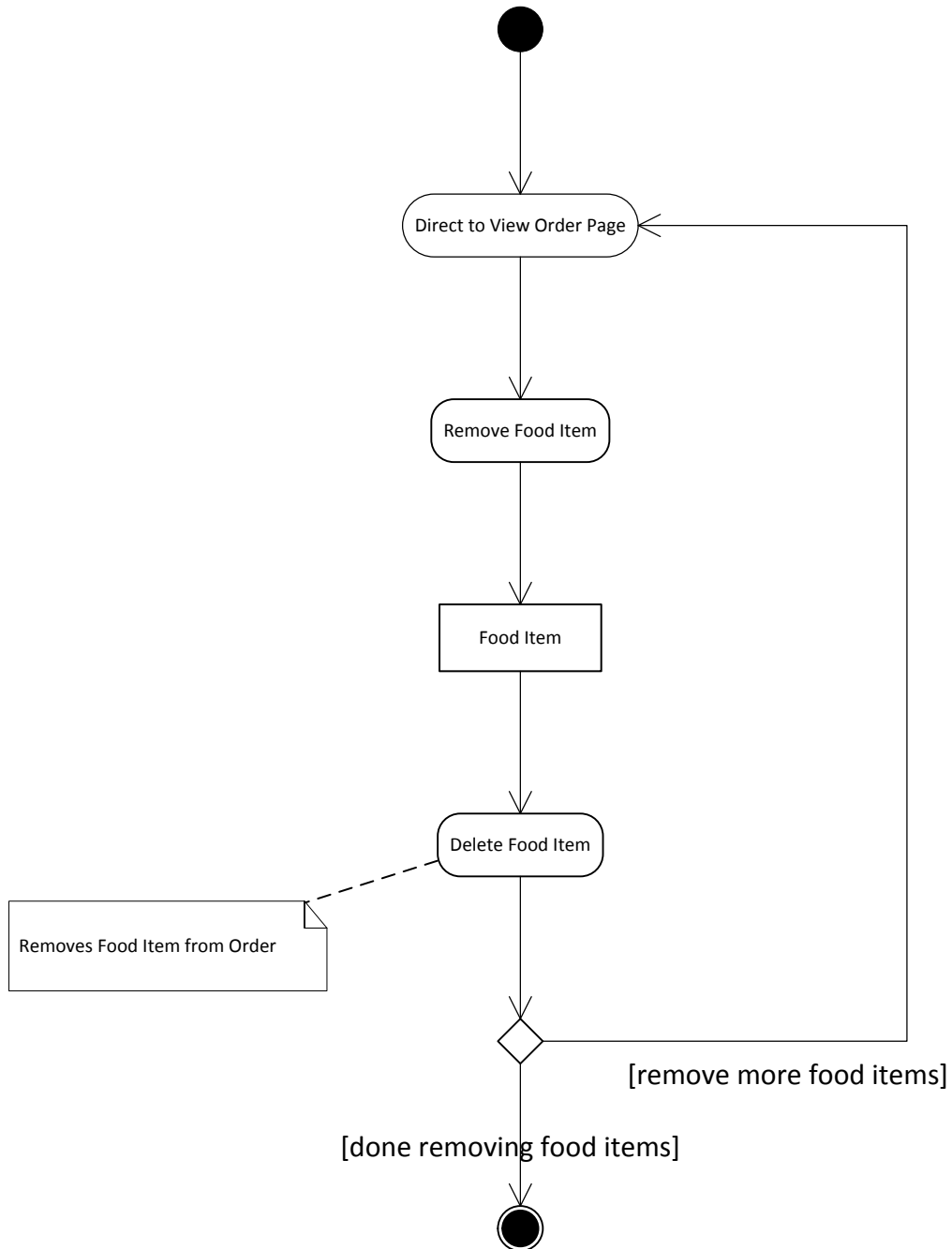


Figure 6-3 Remove Food Item from Order

7 Physical View

7.1 Deployment Diagram

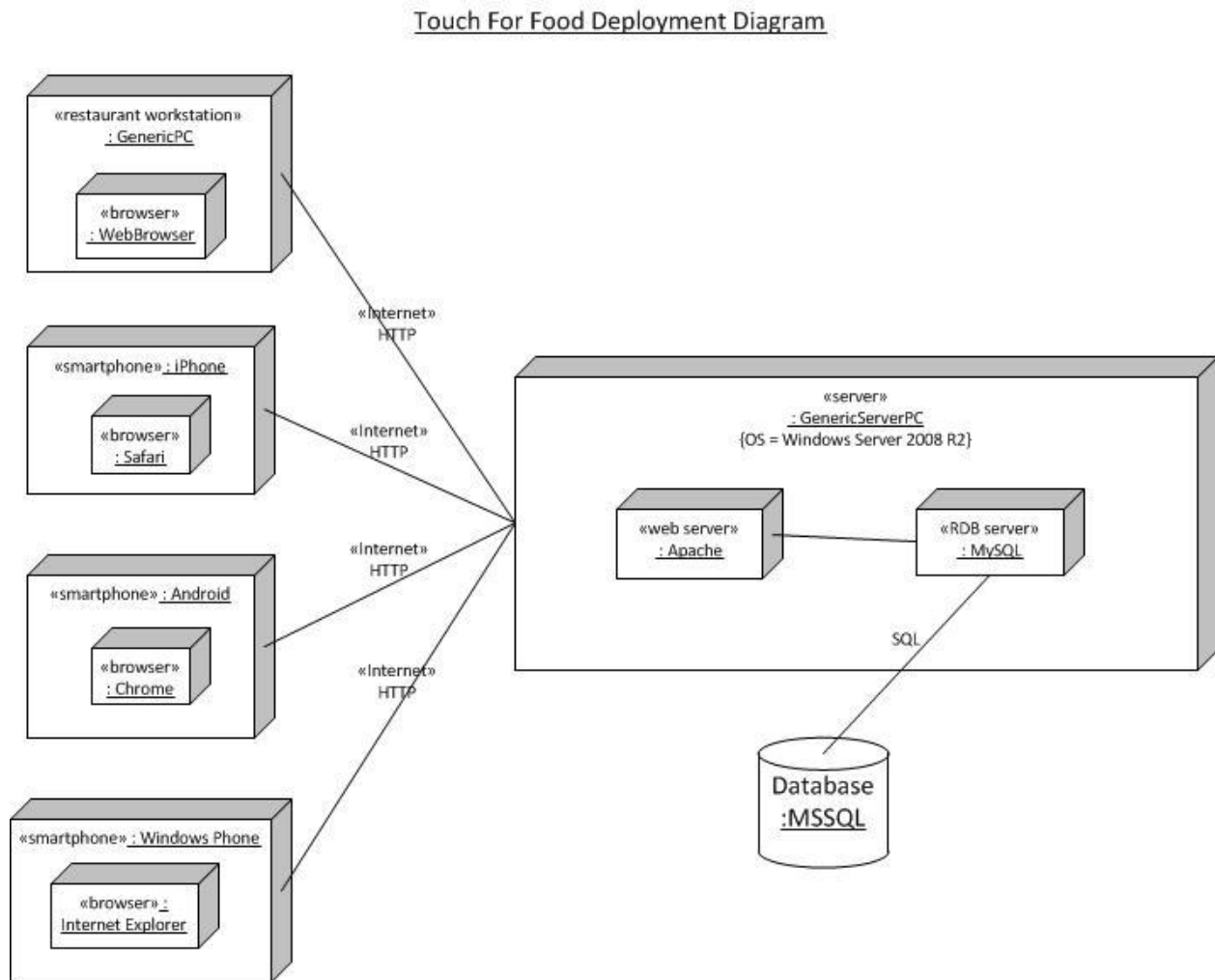


Figure 7-1 Deployment Diagram

8 Size & Performance

From a backend point of view, the element that will impact the performance the most will be the .Net framework along with the MSSQL database server. There are a lot of functions that we will need which are already included in the framework and as such are optimized to work better with the MSSQL database server, as opposed to a third party solution like MySQL. Additionally, the .Net framework includes functions and tools that can be used to form relations between different elements in the most efficient way possible.

In terms of the frontend, the performance will greatly depend on the power the device accessing the mobile application (or website) actually has. In order to mitigate this potential performance hit, a JavaScript library such as JQuery or Dojo will be used. Additionally, if the user is accessing the system through the mobile application, the use of native (i.e. compiled) code will also help the overall performance of it.

The size of the system will initially be small, with one server to provide the order services. When/if the system begins to grow, then more servers will have to be deployed in order to maintain a balanced load across all of them.

9 Quality

TFF is being developed with quality in mind. The architectural designs have been discussed and made to ensure that the quality of TFF is high.

Maintainability, scalability, security and portability are the 4 focuses of TFF. The code base needs to be able to be flexible and easily maintained. Bugs should be easy to locate within TFF's code base. New and different components must be easy to add/modify. This will be ensured by keeping a consistent architecture as well as deploying necessary GRASP patterns.

Since TFF will be exposed to the internet, and it collects user data, security will also be a main focus. Steps will be taken to ensure that malicious users and hackers do not get access to the system and are not able to abuse it.

Appendix A References

Appendix B Glossary

See SRS – Appendix B Glossary for the complete TFF list of terms and definitions.

**Concordia University
Department of Computer Science
and Software Engineering**

Touch For Food

User Interface Requirements

**SOEN 490
Capstone Project
Fall 2012 – Winter 2013**

Cloud Nine

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

1	Introduction	4
1.1	Purpose	4
1.2	Scope.....	4
1.3	Intended Audience	4
1.4	Document Terminology and Acronyms.....	4
1.5	References.....	4
2	Evaluation Mission and Test Motivation	5
2.1	Background.....	5
2.2	Evaluation Mission	5
2.3	Test Motivators	5
3	Target Test Items	6
3.1	Features to be Tested	6
3.2	Features Not to be Tested	6
4	Test Approach	7
4.1	Initial Test Idea Catalogs and Other Reference Sources.....	7
4.2	Testing Techniques and Types.....	7
5	Testing Workflow	11
5.1	Test Phase Table	11
5.2	Test Phase Gantt Chart	12
5.3	Test Management Tools.....	13
5.4	Test Slippage	13
5.5	Defects	13
	Appendix A References	14
	Appendix B Glossary	15

Touch For Food

User Interface Requirements

Version 1.11

Revision History

Date	Rev.	Description	Author(s)
2012-09-17	0.0	Document Creation	Katrina Anderson
2012-10-31	1.0	Added to sprint 1 and contributed section 2	Patrick Modafferi
2012-10-31	1.1	Contributed to section 4.1 and 4.2.1	Mikhail Levkovsky
2012-11-01	1.2	Contributed to Section 4.2.4	Matthew Tam
2012-11-01	1.3	Reviewed Section 4.2.1	Matthew Tam
2012-11-01	1.4	Contributed Section 1	Josh Hum
2012-11-02	1.5	Added sections 5.3, 5.4 & 5.5	Cristian Asenjo
2012-11-02	1.6	Added section 3	Ryan Nasr
2012-11-02	1.7	Added section 4.2.3 minor corrections to 4.2.4	Christian
2012-11-03	1.8	Added section 4.2.2	Cynthia Donato
2012-11-03	1.9	Added section 5.1 & 5.2	Katrina Anderson
2012-11-03	1.10	Corrected sections 5.1, 5.2, 5.3 & References. Formatted Document	Katrina Anderson
2012-11-04	1.11	Corrected section 4.2.3	Cynthia Donato

1 Introduction

1.1 Purpose

The purpose of the Test Plan is to ensure a structured and organized way of testing the TFF software throughout each iteration. It describes various techniques and approaches that will be used in the testing effort. The 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 as well as 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. It is based on the IEEE 829 Test Plan Format [1].

1.2 Scope

The four approaches to testing will include unit testing, user interface testing, usability 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 the Test Plan is the Cloud Nine team. It is intended to guide the developers (who are also the testing team) in developing and testing the TFF software. Any supervisors and stakeholders should also be able to understand the basic plan.

1.4 Document Terminology and Acronyms

Please refer to the *SRS document - Appendix B: Glossary* for a listing 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 lifecycle, independent on what methodology is chosen, testing is an important step. This section will elaborate on what the objective of testing and the test plan is. We will also describe what motivators are in play for selecting the types of tests that are discussed in this document and will be implemented during development.

Refer to the Vision Document for more information about the product “TFF”

2.2 Evaluation Mission

The mission of this testing plan effort is to [2]:

- coordinate testing
- contain defect
- track testing progress
- improve quality and user satisfaction

2.3 Test Motivators

2.3.1 Risk Motivators

By testing early and often, we can be sure to avoid bigger and more costly problems in development phases later on in the project. As we know the cost to repair a bug increases exponentially with respect to the lifecycle period in which it was found. 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 (Traceability)

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. By doing this, the developer is motivated to test so they can mark the progress and close issues.

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. Since our application will attempt to please a general audience, it will be crucial to perform usability tests on a carefully selected demographic. The motivation behind many of our tests will be to validate that the team is building a user friendly product that will appeal to as vast of a spectrum as possible.

2.3.4 Summary

In 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 give the team a reason to test and 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 these tests must pass successfully to be able to close the story 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 be ran on TFF supported hardware such as a compliant Windows Phone, an Android device and the iPhone 4s or above.

- Menu management
- Table management
- Order management (order food, manage orders)
- Reservations
- Restaurant personal page
- Comment, rating and review system
- Customer management
- Restaurant reporting and statistics
- Manage bills for restaurants
- Manage bills for customer

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.

There will not be any tests related to the operating system during development.

4 Test Approach

The testing for the TFF application will be approached using four testing techniques: unit testing, user interface testing, usability 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. Auto generated code by Visual Studio will not be tested [2,3]. For some tests specific standards need to be followed [4,5,6] in order to not receive any false test results (i.e. different formatting in browsers).

4.1 Initial Test Idea Catalogs 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.

4.2 Testing Techniques and Types

4.2.1 Unit Testing

Technique Objective:	The objectives of unit testing are to establish a basis for the successful completion of TFF functionalities and user stories. A user story can only be considered complete when its unit tests have successfully passed. Unit tests also ensure that new functionality do not break any existing ones.
Target Items	The features groups listed in Section 3.1 – Features to be Tested.
Testing Goal:	60% Coverage 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.
Technique:	Unit tests will be written in the Visual Studio IDE [3] and will run from there as well.
Oracles:	After the execution of unit tests, Visual Studio [3] will then provide us with a detailed report of the unit test results.
Required Tools:	Touch For Food – Access to the system code is required to run unit tests. Visual Studio 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. A user story cannot be completed without the successful pass of its unit tests.
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). Any story where its respective unit test has failed must be moved to the backlog to be re-evaluated at a later date.
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 allows us to verify that the user is able to execute each user story in manner that reflects the business functions and requirements through the various windows of the application. UI testing will also allow us to test the access methods such as mouse movement and tab keys to ensure that the field to field navigation is properly executed.
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 developer will indicate if the Test Case has passed or failed in our test report.
Required Tools:	User Interface testing will be performed using an Android Galaxy S3, an iPhone 4S and a Windows phone which is yet to be determined. 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.
Success Criteria:	A test will be considered as a “pass” 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.
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.

4.2.3 Usability Testing

Technique Objective:	The objective of Usability testing is to evaluate the level of ease towards learning and using the TFF interface. It allows us to evaluate whether our application meets the expectations of the user and has an efficient user interface.
Target Items	The feature groups listed in Section 3.1 – Features to be tested
Testing Goal:	Usability tests are to be run on by at least 3 people who are not involved in the development cycle of the application. This will help us evaluate how efficiently any regular user can interact with the application. 100% Coverage of the TFF Screens
Technique:	Developers will create a number of fully dressed test cases that will be performed by the users. We will then measure how fast he/she can go through the test cases and record the results in our test report.
Oracles:	Direct observation will be used to document the performance of the usability tests.
Required Tools:	Touch For Food – Test requires system to be running Google Chrome –Required to run Usability testing of the web version Android, iPhone and Windows Device – Required to run Usability testing of the mobile version
Success Criteria:	A usability test will “pass,” if all users (testers and stakeholders) determine that the outcomes listed in the <i>Expected Results</i> sections of a test case are executed satisfactorily.
Failure contingencies:	An acceptance test will fail if either a tester or a stakeholder has determined that it has not completed the <i>Expected Results</i> criteria satisfactorily. Any problems found shall be added as defects in the Jira repository. Until the end of the iteration, developers and testers must work to implement the changes required for usability tests to pass.
Special Considerations:	It is important that the test cases indicate on which device (mobile or browser) should be used to test the respective usability scenario.

4.2.4 Regression Testing

Technique Objective:	The objective of regression testing is to discover any bugs after implementing new features to the system. This testing verifies that any changes to the existing TFF system do not impact any existing functionality.
Target Items	The feature groups listed in Section 3.1 – Features to be tested
Testing Goal:	60% for Unit and Integration tests 100% for Acceptance, System and UI tests
Technique:	Regression will be done through the use of the Visual Studio IDE [3], where tests can be programmed and rerun at any time. The programmed tests will address Unit and System tests and UI tests will be done manually. Both programmed and manual tests will be performed at the end of each iteration.
Oracles:	Visual Studio will display the results of the programmed tests, indicating which tests failed and at which method.
Required Tools:	Touch For Food – Tests require system code to run tests Visual Studio IDE – Required to run Unit and System tests Google Chrome – Required to do UI tests Android Device – Required to do 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 programmed and acceptance objectives are not met. This means that there was an error in one of the tests where the system did not behave properly. At the end of an iteration, newly implemented changes must pass all regression tests to be concluded as a successful iteration.
Special Considerations:	Some parts of the TFF may not be accessible for tests with Visual Studio

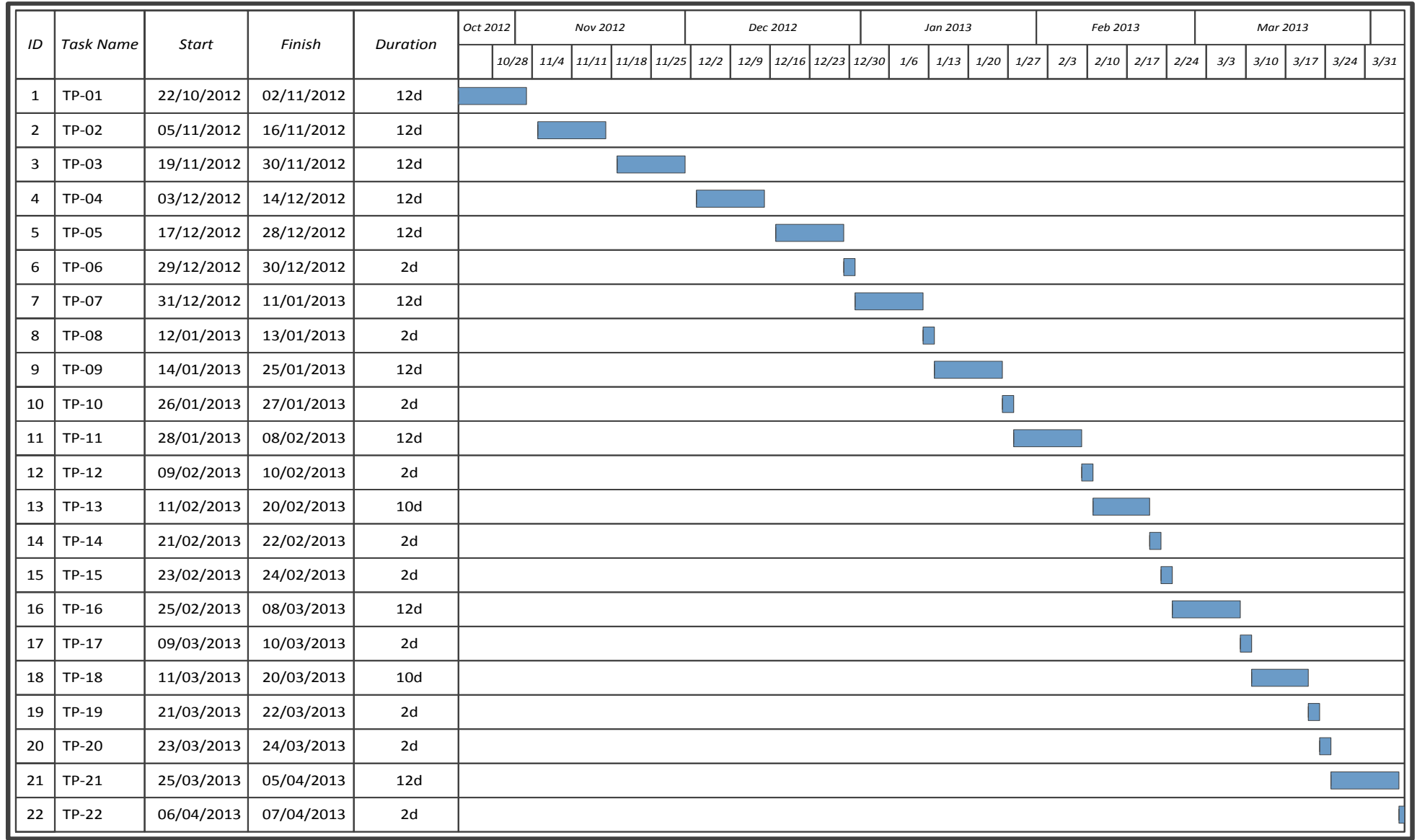
5 Testing Workflow

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 [2].

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 22 nd – November 2 nd
TP-02	Iteration 2: Run unit tests and fix bugs that are found. Update documents accordingly.	November 5 th – 16 th
TP-03	Iteration 3: Run unit tests and fix bugs that are found. Update documents accordingly.	November 19 th – 30 th
TP-04	Iteration 4: Run unit tests and fix bugs that are found. Update documents accordingly.	December 3 rd – 14 th
TP-05	Iteration 5: Run unit tests and fix bugs that are found. Update documents accordingly.	December 17 th – 28 th
TP-06	Iteration 5: Run user interface tests and fix bugs that are found. Update documents accordingly	December 29 th – 30 th
TP-07	Iteration 6: Run unit tests and fix bugs that are found. Update documents accordingly.	December 31 st – January 11 th
TP-08	Iteration 6: Run user interface tests and fix bugs that are found. Update documents accordingly	January 12 st – 13 nd
TP-09	Iteration 7: Run unit tests and fix bugs that are found. Update documents accordingly.	January 14 th – 25 th
TP-10	Iteration 7: Run user interface tests and fix bugs that are found. Update documents accordingly	January 26 th – 27 th
TP-11	Iteration 8: Run unit tests and fix bugs that are found. Update documents accordingly.	January 28 th – February 8 th
TP-12	Iteration 8: Run user interface tests and fix bugs that are found. Update documents accordingly	February 9 th – 10 th
TP-13	Iteration 9: Run unit tests and fix bugs that are found. Update documents accordingly.	February 11 th – 20 th
TP-14	Iteration 9: Run user interface tests and fix bugs that are found. Update documents accordingly	February 21 st – 22 nd
TP-15	Iteration 9: Run usability tests for web application and fix bugs that are found. Update documents accordingly	February 23 rd – 24 th
TP-16	Iteration 10: Run unit tests and fix bugs that are found. Update documents accordingly.	February 25 th – March 8 th
TP-17	Iteration 10: Run user interface tests and fix bugs that are found. Update documents accordingly	March 9 th – 10 th
TP-18	Iteration 11: Run unit tests and fix bugs that are found. Update documents accordingly.	March 11 th – 20 th
TP-19	Iteration 11: Run user interface tests and fix bugs that are found. Update documents accordingly	March 21 st – 22 nd
TP-20	Iteration 11: Run usability tests for phone application and fix bugs that are found. Update documents accordingly	March 23 rd – 24 th
TP-21	Iteration 12: Run unit tests and fix bugs that are found. Update documents accordingly.	March 25 th – April 5 th
TP-22	Iteration 12: Run user interface tests and fix bugs that are found. Update documents accordingly	April 6 th – 7 th

5.2 Test Phase Gantt Chart



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 user stories and directly associate them to their respective tests. 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. Finally, Microsoft Word will be used to back up the data obtained from Jira as well as report on our testing progress.

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 [2].

5.5 Defects

Any defects encountered during the project will be documented in Jira. These defects will be analyzed to establish quality metrics [2].

Appendix A References

- [1] Systeme Evolutif Limited. (2012) TEST PLAN OUTLINE (IEEE 829 FORMAT). [Online]. <http://www.computing.dcu.ie/~davids/courses/CA267/ieee829mtp.pdf>
- [2] K. Anderson, C. Donato, J. Hum, M. Levkovsky, A. Lloyd, P. Modafferi, *"Testing Plan, Cases & Reports"*. Montreal, QC: Concordia University, 2012.
- [3] Microsoft. (2012) Visual Studio 2010. [Online]. [http://msdn.microsoft.com/en-us/library/dd831853\(v=vs.100\).aspx](http://msdn.microsoft.com/en-us/library/dd831853(v=vs.100).aspx)
- [4] Microsoft. (2012) ASP.NET MVC 3 Testing. [Online]. http://msdn.microsoft.com/en-us/vs2010trainingcourse_aspnetmvc3testing.aspx
- [5] The jQuery Foundation. (2012) jQuery; Write less, do more. [Online]. <http://jquery.com>
- [6] W3C. (2012) HTML & CSS. [Online]. <http://www.w3.org/standards/webdesign/htmlcss.html>

Appendix B Glossary

**Concordia University
Department of Computer Science
and Software Engineering**

Touch For Food

Management

**SOEN 490
Capstone Project
Fall 2012 – Winter 2013**

Cloud Nine

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

1	List of User Stories	5
2	AHP.....	6
3	Iteration 1 Plan	6
3.1	Planned Activities & Person-Hour Estimation	6
3.2	Person-Hour Estimation.....	6
3.3	Code Quality Goals.....	7
4	Iteration 1 Report	8
4.1	Person-Hour Work Log	8
4.2	Cumulative Velocities vs. Time.....	9
4.3	Cumulative Flow Diagram.....	9
4.4	Measurement Report.....	10
4.5	Retrospective	10
Appendix A	References.....	12
Appendix B	Glossary	13

List of Figures

Figure 1-1 Backlog Stories	5
Figure 2-1 AHP Graph.....	6
Figure 4-1 Person-Hour Work Log.....	8
Figure 4-2 Cumulative Velocities vs. Time	9
Figure 4-3 Cumulative Flow Diagram.....	9
Figure 4-4 Code Analysis report.....	10

List of Tables

Table 1-1Planned Activities.....	6
Table 1-2 Person-Hour Estimation	7
Table 1-3 Code Quality Goals	7
Table 2-1 Completed User Stories.....	10

Touch For Food

Management

Version 1.0

Revision History

Date	Rev.	Description	Author(s)
2012-11-03	1.0	Document Creation	Josh Hum
2012-11-04	1.1	Update doc	Josh Hum

1 List of User Stories

The following is the list of user stories in the backlog. Stories will be chosen to work on at the beginning of each sprint based on the results of the AHP graph as well as if they are blockers or not.

































Backlog			Create Sprint
	 CAP-26 View Menu		3
	 CAP-25 Order Food		13
	 CAP-29 Manage Order		5
	 CAP-33 Make Reservation		13
	 CAP-34 Call Waiter		1
	 CAP-36 Restaurant Statistics and Reports		21
	 CAP-37 View Restaurant Statistics and Reviews		8
	 CAP-38 Customer Management and Accountability		3
	 CAP-39 Leave Review		3
	 CAP-40 Sign in/Manage Tables		3
	 CAP-41 Customer Bill Management		8
	 CAP-42 Restaurant Bill Management		8
	 CAP-43 Restaurant Page		13
	 CAP-27 Manage Menu		8
	 CAP-35 Manage Personal Profile		34
	 CAP-28 Set Up DB		2

Figure 1-1 Backlog Stories

2 AHP

The AHP method was used to determine priority of user stories. The following graph shows the results of building AHP cost-value matrixes. High priority stories are between the y-axis and the first diagonal line. Medium priority stories fall between the two diagonal lines and low priority stories come between the second diagonal line and the x-axis.

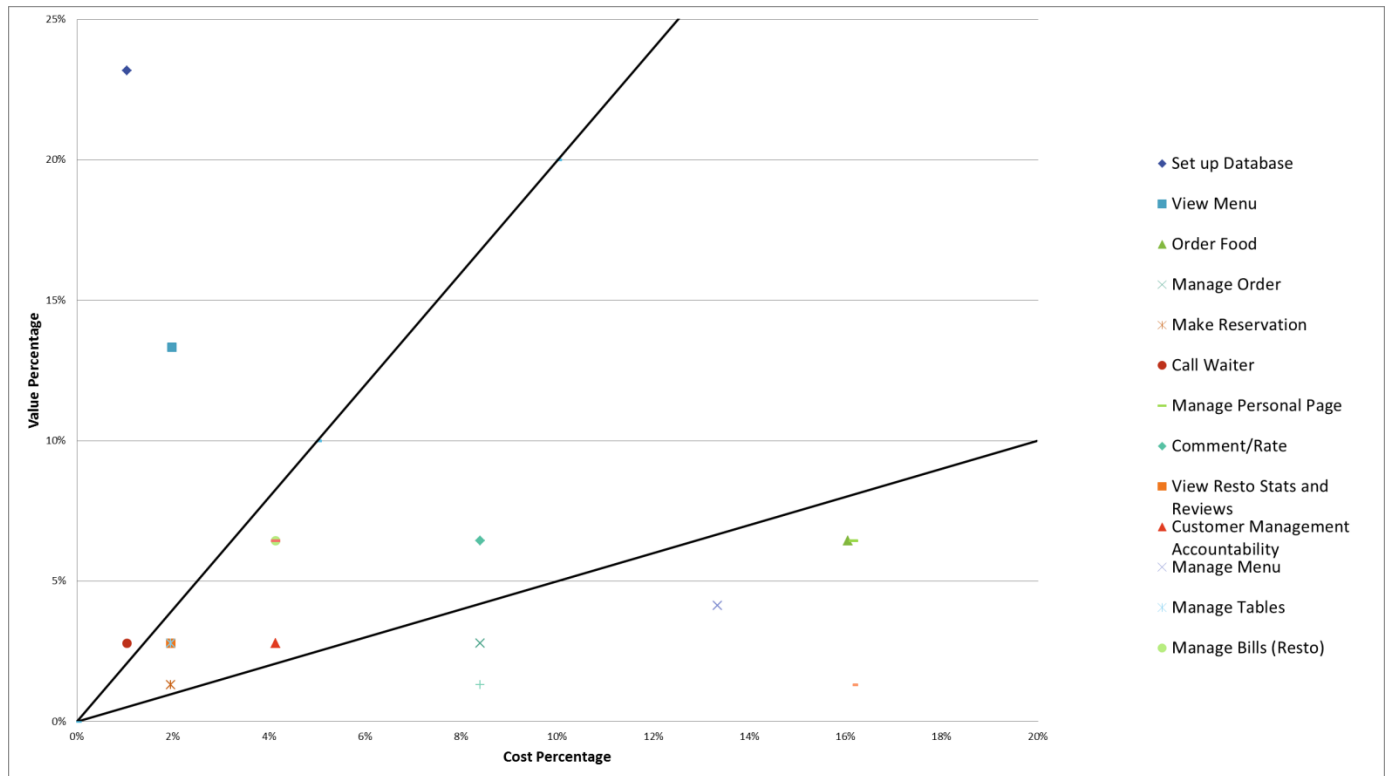


Figure 2-1 AHP Graph

3 Iteration 1 Plan

Iteration 1 is the first iteration where coding development began. The period of time before Iteration 1 was used for planning, meetings, setup and configuration of each team member's development environment.

3.1 Planned Activities & Person-Hour Estimation

The two stories planned for Iteration 1 are Cap-35 – Manage Personal Profile and Cap-27 – Manage Menu. We chose to start with them because they provide a foundation for other user stories to be developed. Our initial

Table 3-1Planned Activities

User Story ID	Total Story Points	Related Use Cases
CAP-35, CAP-27, CAP-28	44.00	UC1.1, UC2.5

3.2 Person-Hour Estimation

The Expert Judgement Method was used to calculate person-hour estimations. Estimations were made for the worst case, most likely case, and best case. The worst case and best case are +30% and -30% The expected case was then calculated with the formula:

Expected case = $1/6([\text{worst case}] + 4[\text{most likely case}] + [\text{best case}])$

Table 3-2 Person-Hour Estimation

User Story ID	Worst Case	Most Likely Case	Best Case	Expected Case
US35	110	90	75	90.8
US27	30	20	15	20.8
US28	10	7	3	6.8
Total (ph)	150	117	93	118.4
Velocity (ph/day)	10.7	8.4	6.6	8.5
Velocity (ph/team member/day)	1.2	0.9	0.7	0.9

3.3 Code Quality Goals

As this is the first iteration, the following goals are based on previous software development experience. We will attempt to meet the following goals with the combination of our code and the .NET Framework.

Table 3-3 Code Quality Goals

Metric	Level
Cyclomatic Complexity (CC)	Average (or lower)
Maintainability Index (MI)	Average (or higher)
Class Coupling (CCP)	Average (or lower)
Lines of Code (LOC)	Average (or lower)
Depth of Inheritance (DOI)	Average (or lower)

4 Iteration 1 Report

4.1 Person-Hour Work Log

The following table shows the person-hour work log for each team member and activity during this iteration. This table was generated from the Jira management system.

























Start Date: 22/Oct/12 End Date: 4/Nov/12 [Change] (UNREGISTERED)															
Issue			Total	Christian Daher	Cristian Asenjo	Cynthia Donato	Josh Hum	Katrina Anderson	Matthew Tam	Mikhail Levkovsky	Patrick Modafferi	Ryan Nasr	Total		
	CAP-8	Documentation	0.5h	0h	0h	0h	0h	0h	0h	0.5h	0h	0h	0.5h		
	CAP-11	CAP-8 ↳ SAD	3.5h	0h	0h	0h	0h	0h	0h	1h	2.5h	0h	3.5h		
	CAP-15	CAP-14 ↳ Email	13.333h	0h	0.083h	1.25h	6.5h	4h	0.5h	0h	1h	0h	13.333h		
	CAP-16	CAP-14 ↳ iPhone	3.417h	0h	0h	0h	2.5h	0.917h	0h	0h	0h	0h	3.417h		
	CAP-17	CAP-14 ↳ Google Groups	0.667h	0h	0.083h	0h	0h	0h	0.583h	0h	0h	0h	0.667h		
	CAP-18	Meetings	1h	0h	0h	1h	0h	0h	0h	0h	0h	0h	1h		
	CAP-19	CAP-18 ↳ Weekly Group Meeting	6.75h	0h	0h	1h	1h	0h	2.75h	0h	2h	0h	6.75h		
	CAP-23	Setup	7.5h	0h	0.5h	3h	0h	0h	0h	0h	4h	0h	7.5h		
	CAP-24	CAP-8 ↳ Proposal	0.5h	0h	0h	0h	0.5h	0h	0h	0h	0h	0h	0.5h		
	CAP-26	View Menu	4.867h	0h	4.867h	0h	0h	0h	0h	0h	0h	0h	4.867h		
	CAP-27	Manage Menu	0.05h	0h	0h	0h	0h	0h	0h	0h	0h	0.05h	0.05h		
	CAP-44	CAP-8 ↳ Test Plan	14.042h	4.5h	0h	1h	1.75h	3h	1h	0.75h	2h	0.042h	14.042h		
	CAP-47	CAP-27 ↳ Menu Editor Main Page	2h	0h	0h	0h	0h	0h	0h	2h	0h	0h	2h		
	CAP-53	DB structure for menu, item and category is faulty.	2h	0h	0h	0h	0h	0h	0h	2h	0h	0h	2h		
	CAP-56	CAP-18 ↳ Other (GoogleHangouts/Mini Team Meetings/Emergency Meetings/SCRUM)	9.333h	2h	1.75h	1h	1.75h	2.833h	0h	0h	0h	0h	9.333h		
	CAP-57	CAP-8 ↳ Management Docs	18h	0h	0h	0h	17h	1h	0h	0h	0h	0h	18h		
	CAP-59	CAP-35 ↳ SAD 4.3 Login Page Use Case	0.5h	0h	0h	0h	0h	0h	0.5h	0h	0h	0h	0.5h		
	CAP-60	CAP-35 ↳ Customer Settings Page Programming	7.5h	0h	0h	7.5h	0h	0h	0h	0h	0h	0h	7.5h		
	CAP-61	CAP-35 ↳ SAD 4.3 Customer Settings Page Use Case	0.417h	0h	0h	0.417h	0h	0h	0h	0h	0h	0h	0.417h		
	CAP-62	CAP-35 ↳ Customer Sign-up Page Programming	5h	0h	5h	0h	0h	0h	0h	0h	0h	0h	5h		
	CAP-63	CAP-35 ↳ SAD 4.3 Customer Sign-up Use Case	0.5h	0h	0.5h	0h	0h	0h	0h	0h	0h	0h	0.5h		
	CAP-64	CAP-35 ↳ Customer Profile Page View Programming	0.5h	0h	0h	0.5h	0h	0h	0h	0h	0h	0h	0.5h		
	CAP-65	CAP-35 ↳ SAD 5.3 Class Diagram	3h	0h	0h	3h	0h	0h	0h	0h	0h	0h	3h		
	CAP-66	CAP-35 ↳ SAD 4.1 & 4.2 Update	1h	0h	0h	0h	1h	0h	0h	0h	0h	0h	1h		

Figure 4-1 Person-Hour Work Log

4.2 Cumulative Velocities vs. Time

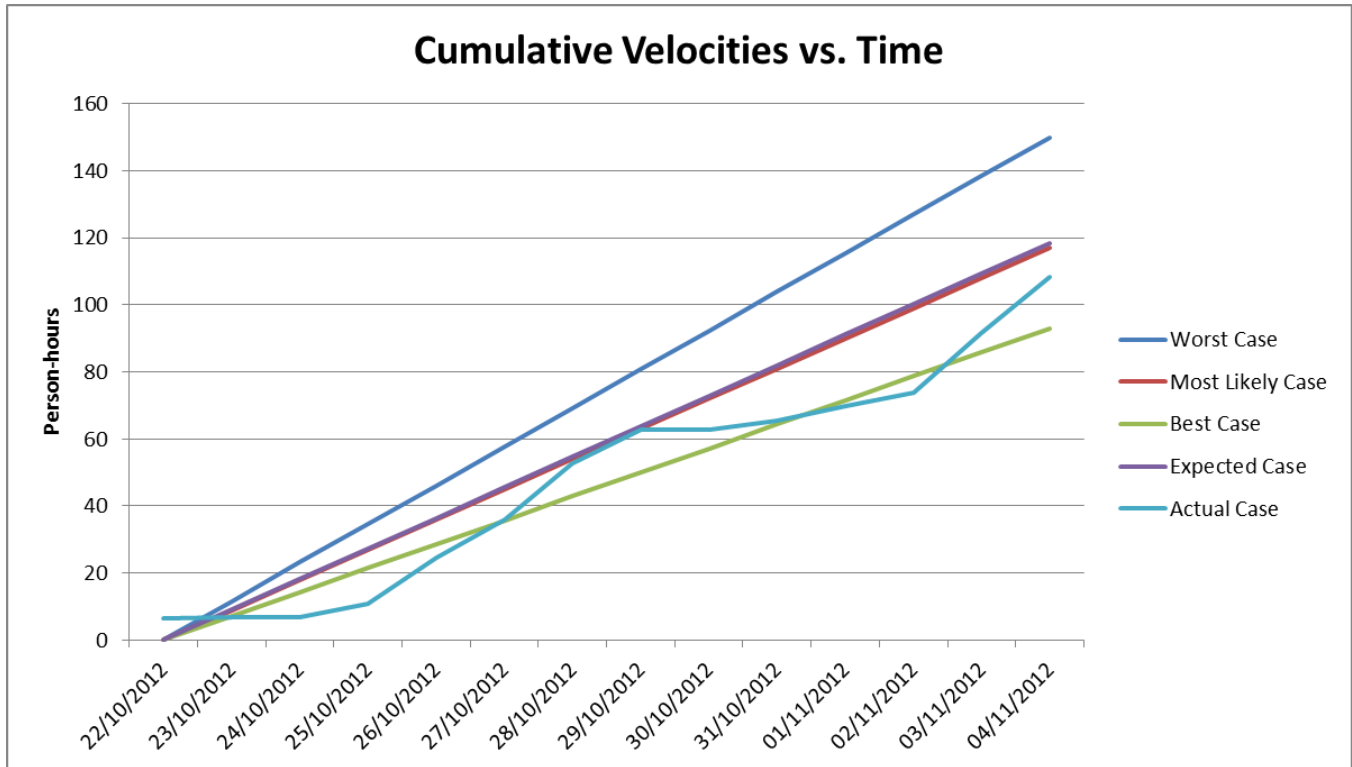


Figure 4-2 Cumulative Velocities vs. Time

4.3 Cumulative Flow Diagram

The following diagram shows the three stories and how the work progressed. Blue tasks have not been started, purple tasks are in progress and green represents a completed story. This diagram was generated in Jira.

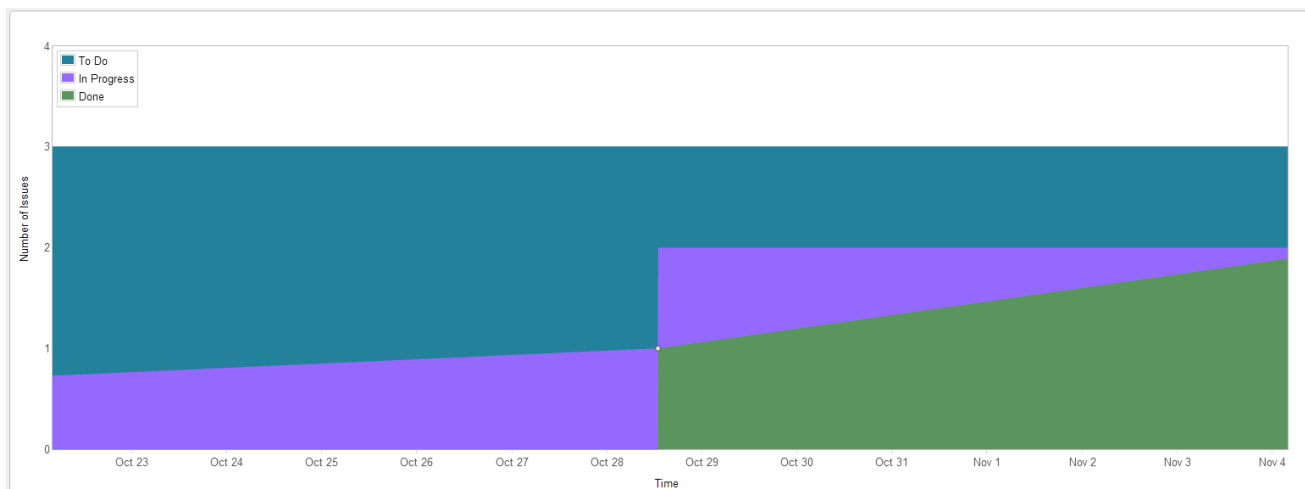


Figure 4-3 Cumulative Flow Diagram

4.4 Measurement Report

4.4.1 Code Quality Analysis

The following report was generated after analyzing the code. All our minimum goals were met while some were exceeded.

Analysis tool used: Code Metrics Viewer
Found at: <http://visualstudiogallery.msdn.microsoft.com/9f35524b-a784-4dbc-bd7b-6babd7a5a3b3>
Version: 1.5.3
Last updated: 2/5/2012

1	Scope	Hierarchy	Maintainability Index	Cyclomatic Complexity	Depth of Inheritance	Class Coupling	Lines of Code
2	ModuleScope	TouchForFood.dll	88	286	58	3	413
3	NamespaceScope	TouchForFood	90	4	9	2	7
4	TypeScope	MvcApplication	90	4	9	2	7
5	MemberScope	Application_Start() : void	80	1	3		3
6	MemberScope	MvcApplication()	100	1	1		1
7	MemberScope	RegisterGlobalFilters(GlobalFilterCollection) : void	94	1	2		1
8	MemberScope	RegisterRoutes(RouteCollection) : void	82	1	3		2
9	NamespaceScope	TouchForFood.Controllers	78	76	36	3	187
10	TypeScope	CategoryController	76	12	12	3	33
21	TypeScope	HomeController	96	2	2	3	3
24	TypeScope	Menu_CategoryController	69	20	29	3	42
35	TypeScope	MenuController	73	16	30	3	38
46	TypeScope	RestaurantController	76	12	12	3	33
57	TypeScope	UserController	75	14	15	3	38
58	MemberScope	Create() : ActionResult	89	1	2		2
59	MemberScope	Create(user) : ActionResult	69	2	8		6
60	MemberScope	Delete(int) : ActionResult	77	1	5		3
61	MemberScope	DeleteConfirmed(int) : ActionResult	71	1	8		5
62	MemberScope	Details(int) : ViewResult	77	1	5		3
63	MemberScope	Dispose(bool) : void	87	1	3		2
64	MemberScope	Edit(int) : ActionResult	77	1	5		3
65	MemberScope	Edit(user, HttpPostedFileBase) : ActionResult	59	4	11		11
66	MemberScope	Index() : ViewResult	83	1	5		2
67	MemberScope	UserController()	92	1	2		1
68	NamespaceScope	TouchForFood.Models	92	206	21	2	219
69	TypeScope	category	92	9	4	1	11
79	TypeScope	item	92	19	6	1	21
99	TypeScope	menu	93	11	5	1	12
111	TypeScope	menu_category	93	13	5	1	14
125	TypeScope	menu_item	93	13	4	1	13
139	TypeScope	order	92	25	8	1	27

Figure 4-4 Code Analysis report

4.5 Retrospective

In Iteration 1, we completed US27 and US28. This amounts to 10 out of the planned 44 story points. Therefore, it was not a successful iteration. The following table shows the completed user stories:

Table 4-1 Completed User Stories

User Story ID	User Story	Story Points
CAP-27	As a restaurant manager, I would like to create, update, delete and view customized menus.	8
CAP-28	As a developer, I would like to set up the database in order to be able to begin building the system.	2

Although the story US35 could not be completed, it was partially done and progress was made. However, for the next sprint, we will adjust the amount of story points we take on based on our current velocity.

There were both positives and negatives to be found in this iteration. We plan on learning from our mistakes in this iteration and build on the positives to make the next iterations a success.

Positives:

- Finished the critical story of setting up the database
- Overcame obstacles together as a team

Negatives:

- Did not complete all the user stories planned
- Could have better communication
- Missed out on certain management metrics

105.9 person-hours were spent working in Iteration 1. This was close to the 118 person-hour estimate. However, it is below half of the initial budgeted person-hours scheduled. We attribute this to two causes. First, this iteration was started while the developers were right in the middle of midterms. We expect to accomplish much more and spend more time working in the following iteration due to less time constraints. Secondly, the team is not yet fully accustomed to the use of the management system. Thus the work times logged are much lower than the actual work times. There seem to be quite a few data entry errors as well. We will look to correct this in the following iteration.

Appendix A References

Appendix B Glossary

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