**CMPS411**

**Software Requirements Specification (SRS) Template**

**Social Foodies**

**Software Requirements Specification**

**Document**

**Version: 1** **Date: 11/29/12**

# 1. Introduction

## 1.1 Purpose

This SRS document contains a detailing of the structure and purpose of the CS 411 group project. Its intended readers should include, but not be limited to the professors involved in organizing the course.

## 1.2 Scope

The scope of this course includes production of one software project completed under the name YouChew. The product will enable users to find, view and leave feedback for food venues through the social network site, facebook. The goal of the project is to recreate the complexity of well known food reviewing sites on a small scale with the aim of gaining knowledge and proficiency of the enabling technologies.

## 1.4 Overview

Overall Viewpoint of the Project...................................................................2

Project Requirements.....................................................................................3

Project Management......................................................................................4

Document Approval.......................................................................................5

# 

# 2. The Overall Description

## 2.1 Product Perspective

## SiteMap.png

The product can be best summarized with the preceeding document that helps show the overall structure of the product from a high level. The diagram is not a functional requirement specification, but is intended to present the original intention of the project.

### 2.1.1 System Interfaces

### The project does rely on one external system interface, The FourSquare API, for its data needs. Sufficent document on the specifics in how this interfact is used is left within the project. Queries are also made to the Facebook and MapBox API as well.

### 2.1.2 Interfaces

The project's interface is presented through the website on which it is hosted. Users can interact with the site using a browser and traveling to the correct url. Navigation and functionality are clearly visible throughout the site. Future updates might include web controls that enable users to utilize ease of access tools.

### 

### 2.1.3 Hardware Interfaces

The project may run on a server provided by 1and1 hosting technologies. The hardware of the server includes a 2.1Ghz Quad-Core AMD processor, 4 GB RAM, 750GB of hard disk storage, and software RAID 1 running on a standard Windows 2008 Server operating system.

### 2.1.4 Software Interfaces

Our project connects to the FourSquare API to retrieve venue information. We specifically target the Venues Platform for data access. When accessing the platform, API calls are made to a userless endpoint. OAuth is used to verify calls made by requiring the query include the application's client id and client secret it which are provided by FourSquare upon application registration. Various endpoints exist within the Venues platform and many accept numerous optional parameters for greater control in creating specific queries. More information about the API and documentation can be found at developer.foursquare.com.

The project connects to the facebook API for user interaction with venues. The site asks that the user be logged into a facebook account to leave input for venues. Specific documentation can be found at developers.facebook.com.

The MapBox API is also utilized by our project for displaying venues on maps. Documentation can be found at mapbox.com/developers/api.

### 2.1.5 Communications Interfaces

Our project is deployed in the .NET framework specifically using ASP.NET MVC 4 for project structure. The .NET Framework 4.0+ and IIs server are required within the operating environment.

### 2.1.6 Memory Constraints

No specific testing has been performed on the project, however the environment should have at least 2 GB of RAM for optimal use.

## 2.2 Product Functions

The site requires internet access to perform HTML request queries to an external APIs.

## 2.3 User Characteristics

The intended user base of this project includes anyone with access to the internet. Our site is designed to provide users with fast results to queries about food venues around their area. Viewing and search can be done without an account, but leaving feedback requires logging in with facebook.

## 3. Specific Requirements

**3.1 External Interfaces**

## 3.3 Performance Requirements

*This subsection specifies both the static and the dynamic numerical requirements placed on the software or on human interaction with the software, as a whole. Static numerical requirements may include:*

*(a) The number of terminals to be supported*

*(b) The number of simultaneous users to be supported*

*(c) Amount and type of information to be handled*

*Static numerical requirements are sometimes identified under a separate section entitled capacity.*

*Dynamic numerical requirements may include, for example, the numbers of transactions and tasks and the amount of data to be processed within certain time periods for both normal and peak workload conditions.*

*All of these requirements should be stated in measurable terms.*

*For example,*

*95% of the transactions shall be processed in less than 1 second*

*rather than,*

*An operator shall not have to wait for the transaction to complete.*

*(Note: Numerical limits applied to one specific function are normally specified as part of the processing subparagraph description of that function.)*

## 3.4 Logical Database Requirements

*This section specifies the logical requirements for any information that is to be placed into a database. This may include:*

1. *Types of information used by various functions*
2. *Frequency of use*
3. *Accessing capabilities*
4. *Data entities and their relationships*
5. *Integrity constraints*
6. *Data retention requirements*

*If the customer provided you with data models, those can be presented here. ER diagrams (or static class diagrams) can be useful here to show complex data relationships. Remember a diagram is worth a thousand words of confusing text.*

## 3.5 Design Constraints

*Specify design constraints that can be imposed by other standards, hardware limitations, etc.*

### 3.5.1 Standards Compliance

*Specify the requirements derived from existing standards or regulations. They might include:*

*(1) Report format*

*(2) Data naming*

*(3) Accounting procedures*

*(4) Audit Tracing*

*For example, this could specify the requirement for software to trace processing activity. Such traces are needed for some applications to meet minimum regulatory or financial standards. An audit trace requirement may, for example, state that all changes to a payroll database must be recorded in a trace file with before and after values.*

## 3.6 Software System Attributes

*There are a number of attributes of software that can serve as requirements. It is important that required attributes by specified so that their achievement can be objectively verified. The following items provide a partial list of examples. These are also known as non-functional requirements or quality attributes.*

*These are characteristics the system must possess, but that pervade (or cross-cut) the design. These requirements have to be testable just like the functional requirements. Its easy to start philosophizing here, but keep it specific.*

### 3.6.1 Reliability

*Specify the factors required to establish the required reliability of the software system at time of delivery. If you have MTBF requirements, express them here. This doesn’t refer to just having a program that does not crash. This has a specific engineering meaning.*

### 3.6.2 Availability

*Specify the factors required to guarantee a defined availability level for the entire system such as checkpoint, recovery, and restart. This is somewhat related to reliability. Some systems run only infrequently on-demand (like MS Word). Some systems have to run 24/7 (like an e-commerce web site). The required availability will greatly impact the design. What are the requirements for system recovery from a failure? “The system shall allow users to restart the application after failure with the loss of at most 12 characters of input”.*

### 3.6.3 Security

*Specify the factors that would protect the software from accidental or malicious access, use, modification, destruction, or disclosure. Specific requirements in this area could include the need to:*

1. *Utilize certain cryptographic techniques*
2. *Keep specific log or history data sets*
3. *Assign certain functions to different modules*
4. *Restrict communications between some areas of the program*
5. *Check data integrity for critical variables*

### 3.6.4 Maintainability

*Specify attributes of software that relate to the ease of maintenance of the software itself. There may be some requirement for certain modularity, interfaces, complexity, etc. Requirements should not be placed here just because they are thought to be good design practices. If someone else will maintain the system*

### 3.6.5 Portability

*Specify attributes of software that relate to the ease of porting the software to other host machines and/or operating systems. This may include:*

1. *Percentage of components with host-dependent code*
2. *Percentage of code that is host dependent*
3. *Use of a proven portable language*
4. *Use of a particular compiler or language subset*
5. *Use of a particular operating system*

*Definitions of the quality characteristics not defined in the paragraphs above follow.*

*• Correctness - extent to which program satisfies specifications, fulfills user’s mission objectives*

*• Efficiency - amount of computing resources and code required to perform function*

*• Flexibility - effort needed to modify operational program*

*• Interoperability - effort needed to couple one system with another*

*• Reliability - extent to which program performs with required precision*

*• Reusability - extent to which it can be reused in another application*

*• Testability - effort needed to test to ensure performs as intended*

*• Usability - effort required to learn, operate, prepare input, and interpret output*

## 3.7 Organizing the Specific Requirements

*For anything but trivial systems the detailed requirements tend to be extensive. For this reason, it is recommended that careful consideration be given to organizing these in a manner optimal for understanding. There is no one optimal organization for all systems. Different classes of systems lend themselves to different organizations of requirements in section 3. Some of these organizations are described in the following subclasses.*

### 3.7.1 System Mode

*Some systems behave quite differently depending on the mode of operation. When organizing by mode there are two possible outlines. The choice depends on whether interfaces and performance are dependent on mode.*

### 3.7.2 User Class

*Some systems provide different sets of functions to different classes of users.*

### 3.7.3 Objects

*Objects are real-world entities that have a counterpart within the system. Associated with each object is a set of attributes and functions. These functions are also called services, methods, or processes. Note that sets of objects may share attributes and services. These are grouped together as classes.*

### 3.7.4 Feature

*A feature is an externally desired service by the system that may require a sequence of inputs to effect the desired result. Each feature is generally described in as sequence eof stimulus-response pairs.*

### 3.7.5 Stimulus

*Some systems can be best organized by describing their functions in terms of stimuli.*

### 

### 3. 7.6 Response

*Some systems can be best organized by describing their functions in support of the generation of a response.*

# Change Management Process

Our team uses Git for source control and revision via an open source repository provided by GitHub. Work loads are provisioned during meetings at which members agree on tasks distribution that best matches the skill set possessed. The project is freely available for download from the same site. Requirements are ultimately completely flexible since the team is essentially its own client. Changes that do occur are informally noted through email, phone, or other electronic messenger service.

# Document Approvals

Approval of this document will go to Prof. Ghassan Alkadi.