Feed Me!

by

Brett Toothman

A Design Freeze Submitted to
the Faculty of the Information Technology Program
in Partial Fulfillment of the Requirements for
the Degree of Bachelor of Science
in Information Technology

University of Cincinnati
College of Applied Science

June 2010

Feed Me!

by

Brett Toothman

Submitted to
the Faculty of the Information Technology Program
in Partial Fulfillment of the Requirements
for
the Degree of Bachelor of Science
in Information Technology

Brett Toothman	Date
Annu Prabhakar, Faculty Advisor	Date
Hazem Said, Ph.D. Department Head	Date

Acknowledgements/Dedication

This project is dedicated to my friends and family, whose support and assistance made this project possible.

I would also like to thank the open source communities. The shared knowledge and help of various forums, tutorials, guides, and blogs, made this project better then I ever could have on my own.

Table of Contents

List of Figures

Abstract

Feed Me! is a web application, designed for cooking enthusiasts. It blends the traditional cooking web page with social functions. Allowing users to submit recipes of their own, as well as explore recipes from other users with similar tastes and abilities. The application takes into account multiple factors when suggesting recipes such as: past recipes, tools available, and user ratings. The primary purpose of this application is to go beyond the traditional cookbook web site, and be able to make intelligent suggestions for cooks who may not otherwise find a suitable recipe. *Feed Me!* is not a site for the professional chef, it is designed with the “weekend cook” in mind; for the hobbyist who may not have an exact recipe in mind and for anybody who is just interested in cooking.

Feed Me!

1. Project Description

1.1 Problem: Current cooking recipe websites do not address the abilities of a novice cook (1). They offer the basic functionality of a cookbook, in the standard format with list of ingredients first and cooking directions second (2). This is perfect for talented chefs who just need a place to store recipes they created, but inexperienced cooks may need more help. The goal of this project was to create a site with the weekend cook in mind, the person who doesn't know exactly what dish they are looking for, but would like to try something new. Instead of simply being a place to store recipes, this new site would be focused more on being able to suggest a recipe to those who may not have one in mind.

1.2 Solution: The first part of the solution was to move away from the concept of users simply looking for recipes. A user can create a profile, which keeps track of what types of recipes they like, what tools they have available (pots, pans, crockpot, etc), and their general skill level. The site will follow a mantra of "A suggestion for everything", meaning instead of looking up a recipe, a user could look up an ingredient and get recipe suggestions. The same idea applies to users that look up tools ("I want to cook something in the wok tonight"), time ("I need something in 30min or less"), number of ingredients ("I need something I can cook with 5 ingredients or less"), or any other number of variables. This emphasis on suggestion also helps to teach the user basic cooking techniques; if the user looks up egg recipes, then there are also tutorials on how to boil an

egg, or how to separate the egg yolk from the egg whites, or what the term “scrambled” means (3).

1.3 Content: The recipes on the site are entirely user submitted, so an important part of the application is the ability to categorize and sort these recipes. The tagging system allows users to label their recipes with multiple keywords, so that recipes can be quickly grouped. These groupings help users narrow their searches, and allow the site to suggest other simular recipes (based on common tags). Every recipe will also have a rating and comments attached to them. This way users can vote on the quality of a particular recipe or leave feedback on the recipes strengths or weaknesses.

2. User Profiles

Users of the application will fall into one of three categories:

- Guests
- Registered Users
- Site Administrators

2.1 Guests: Guests are users that are not logged into the site. While a guest cannot use many of the sites primary features, they are still able to browse and search for recipes. They cannot however, submit recipes or comment on recipes until they log in.

2.2 Registered Users: As soon as a user logs into the site using a valid username and password, the site recognizes them as a registered user. This allows the site to provide additional tools to the user. Whereas a guest can only browse recipes, a registered user can submit recipes of their own, rate and comment on recipes, and use the cookbook feature.

2.3 Administrators: Administrators have all the same abilities of a registered user, with the added benefit of being able to control various aspects of the site. The administrators are responsible for moderating site content. Besides being able to delete users or recipes at their discretion, admins are also responsible for posting announcements and various other content on the home page.

2.4 Potential Users: The typical user of this application is the person who enjoys cooking, but isn't necessarily experienced. They know that they want to cook something, but may not know exactly what. They need a tool that can make suggestions based on various factors such as ingredients, price, and difficulty.

Examples of Users / Scenarios

- Someone cooking a romantic dinner for a date
- Someone cooking something to bring to friend's cookout
- College students who want to find a cost effective meal

2.5 Experience with Similar Applications: The user should at least be familiar with the layout and use of both print cookbooks and other online cooking sites. Both print versions and digital versions suffer from the same drawbacks; they act only as storage for recipes, however, knowing the general layout for a recipe (ingredients followed by directions) is helpful. A user has to already know the recipe they are looking for, and be able to follow whatever instructions and jargon are required. The *Feed Me!* application goes a step further in that, if the user is confused about certain parts of a recipe, it will suggest easier recipes or give better explinations of the complicated steps.

2.6 Technological Experience: Since this is a web application, the user has to be familiar with basic web standards such as: using a web browser, signing into a user account, creating an account profile, using hyperlinks. A knowledge of standard website layouts would be helpful as well; knowing the usual places for navigation, logins, and contact info.

3. Design Protocols

3.1 Interface: The primary focus of the site is organizing and displaying recipes, so the design of the site has the basic concepts of working with products. As with most sites, each page is broken down into three sections: the header the content and the footer. The header acts as consistent access to important tools. Whereas the content may change from page to page, the header only changes in a few instances (such as a user status changing from guest to logged in). This makes the header the ideal location for the search bar and user functions. The footer section is another constant among most pages. The footer acts

as a sitemap: it holds links to areas such as a help page, contact page, and general site info.

Navigation for the application is very limited. The site emphazises the use of tools to access reasources, instead of navigation. So instead of having a navigation bar or menu, the user can access the needed section through the tool provided. Users should appear under a very distinct category, and each category has a specific tool provided.

- Users looking for a specific recipe – search bar allows for entering all criteria
- Users wanting to browse recipes – the homepage holds recipe suggestions
- Users wanting to change user preferences, or submit recipe/media – User toolbar in header
- Administrators needing to change site settings – once authorized as admin, the admin tool appears in the header.
- Other – the footer provides a sitemap and links to any other type of user, including users looking for site information, or help on how to use the site.

3.2 Icons/Graphics: The application is designed with the weekend cook in mind, this means that the common user is not using it in any professional manner. While a cooking application does not draw in a younger crowd, it also is focused on the hobbyists and cooking enthusiasts. The graphics and icons on the application must reflect this by being not too cartoony, but at the same time, keep an informal feel. Bright colors, appropriate shading and rounded edges convey that message.

3.3 Color Scheme: Following the same design concepts as the graphics, the color scheme has to convey the informal but not cartoony look and feel. A very simple color scheme with a bright primary color and a weaker secondary conveys this message well. The site uses a bright orange for any headers, icons, links (things that need to attract attention) and a duller grey for the borders and backgrounds. The bulk of the content uses black text on a white background. This scheme makes the site very easy to read, and looks simple and uncluttered, while still allowing important elements to be easiliy identified.

3.4 Help: The footer provides access to the help pages. This allows the user to access the help section no matter where they are in the site. There will be a help section dedicated to processes such as: user creation, searching for recipes, submitting recipes, and working with the *My Cookbooks* feature. In case of larger problems or complaints there will also be a contact page where the user can contact a site administrator.

4. Technology

4.1 Language/Framework (Ruby on Rails): Which framework and language to use was the biggest decision for a project. Each different language has different tools, styles, and databases, in essence entirely different methods of creating a project. In the end Ruby on Rails was choosen as the best language and framework for the project. The selection of plugins available for the Rails community coincides very well with the feature list of this project. Rails also had the added advantage of working well in a muti-operating-system development environment. Some other advantages of the Rails framework is its focus on

an agile development cycle, which allows changes to the design of the project to happen quickly and easily (5).

4.2 Database (MySQL): The primary requirement for the database on this project is to be compatible with the Ruby on Rails framework. This did not really narrow the field much, but in the end MySQL won as the obvious choice. MySQL works well with Ruby on Rails, but also the added advantage of being lighttweight, quick, familiar to the development team, multiplatform, and of course free for non-business use.

4.3 Webserver (Mongrel): Mongrel is the web server that comes packaged with the Ruby on Rails framework. This makes it very lightweight and portable, making the project very easy to test on multiple systems, without changing the production code (5).

4.4 Plugins: Plugins act as a shortcut for the developer. They are packages of code that are already written to handle common tasks. User authentication is a classic example. Many web applications have a need for user authentication, and it is inefficient for every web developer to rewrite the code. It is much more efficient for a single developer to create a standalone user authentication module that can be included in multiple projects. This reduces the amount of work developers have to do, and because each module is utilized by multiple projects and maintained by a single developer, the module is much more stable than if it had been written just for a single app (6).

Plugins to be used:

- Paperclip : Image conversion / storage
- Sphinx / Thinking Sphinx : Full text searching
- Authlogic : User authentication

5. Proof of Design

Just how the current solution addresses the problem statement is much easier to see if the typical user of a cooking site is broken down into three seperate groups:

- The advanced cook
- The inexperienced cook that has has specific criteria in mind and needs recipe suggestions
- The inexperienced cook who has no idea at all of what they want

The problem statement suggests that far too many cooking sites address only the first group and completely ignore the last two. *Feed Me!* specifically focuses on these groups using tools specifically oriented towards each.

5.1 The Cook with Criteria:

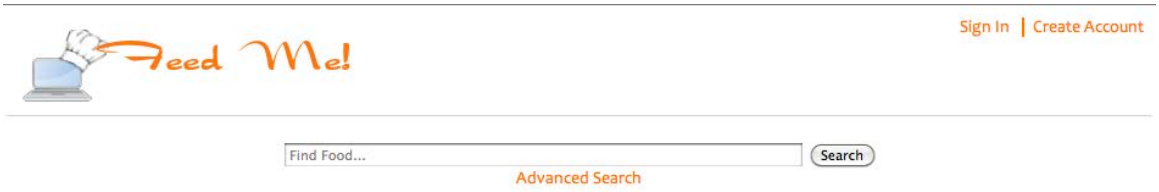

Figure 1: Site header

Figure 1 shows the header that sits at the top of every page in the site. This header provides the tools that allow a user to specify the criteria on which they would like to search. First of all, the search area allows the user to search by keywords in the recipe name, ingredients or tags. If the user needs any more specific searching capabilities, the advanced search section expands to allow for input on every aspect of the recipe, such as difficulty, serving sizes, and cook time.

Search capabilities are a stable of most cooking sites, but what really sets *Feed Me!* appart is the user section, and the ability to search based on a users profile. Once a user creates an account and logs in, they can set preferences such as general skill level, allergies, available tools; then they can search for recipes based on these preferences. Theres no point in returning a grilling recipe with peanut oil, to a user that is allergic to peanuts and has no grill.

5.2 The Cook Who Needs Suggestions: The user who has no idea what they want to cook, or just the user who is browsing the site in their spare time, is a surprisingly large user-base. These are the users that have a few minutes to kill and just want to quickly check the site to see if there's anything interesting, if the site does not immediatly deliver,

the user will move on. The home page of *Feed Me!* is designed exactly with this user in mind.

The rails framework has a tool called database migrations. Migrations implement a type of version control for the main structure of the project. Giving the ability to rollback unwanted versions of the project model, seeing the model revision history, and easily changing the model in the case of poor-design.

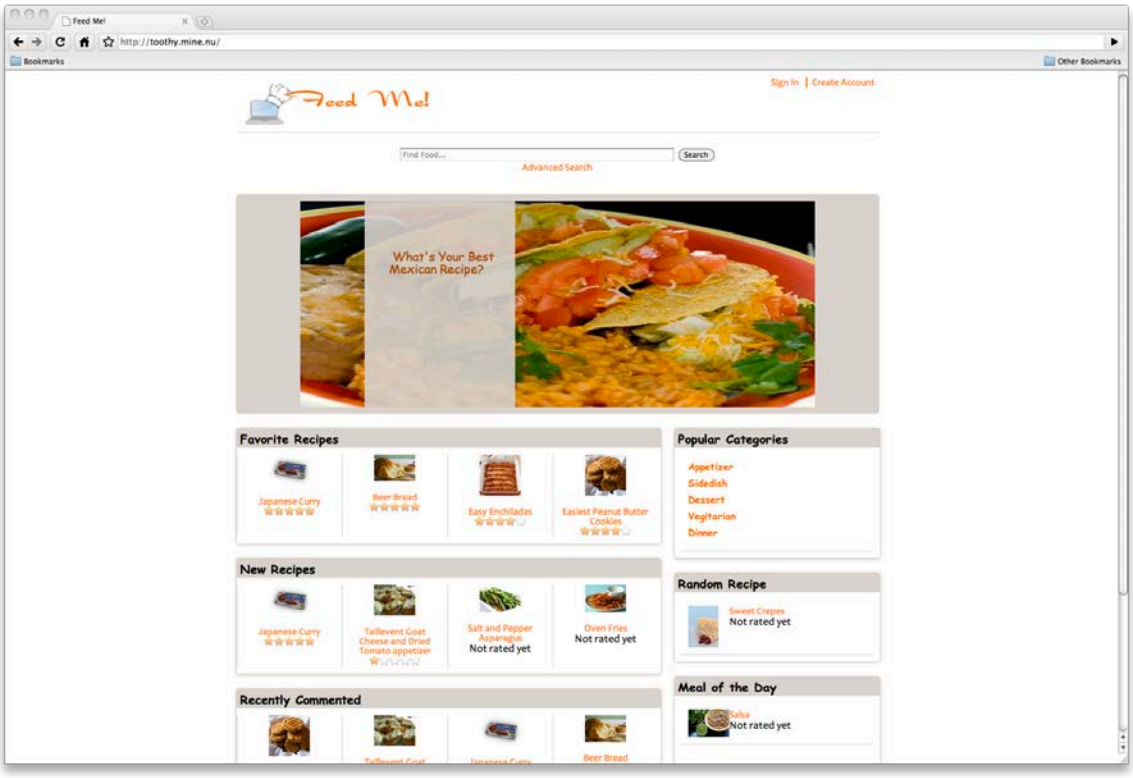

Figure 2: Feed Me! home page

Figure 2 shows the current homepage. This is the page that will greet users as soon as they arrive on the site. The header is the same as in the last section, but the content is what's important here. The large block on the top is the announcements section; this area is set aside for important messages or recipes of the day. It will run through a list of

announcements one at time, on a timer. So events such as contests, featured recipes, or any other content that may grab the users attention can be prominently displayed.

Below the announements will be the recipe lists. Here the site will suggests recipes such as featured recipes (turkey recipes on thanksgiving, cookies on valentines day, etc), highly rated recipes, and a meal of the day (meal meaning recipes for all courses of a meal). When a user logs into the site, it also adds some lists relevant only to that user. Such as recently viewed recipes, and recipes sugessted based on the users viewing history.

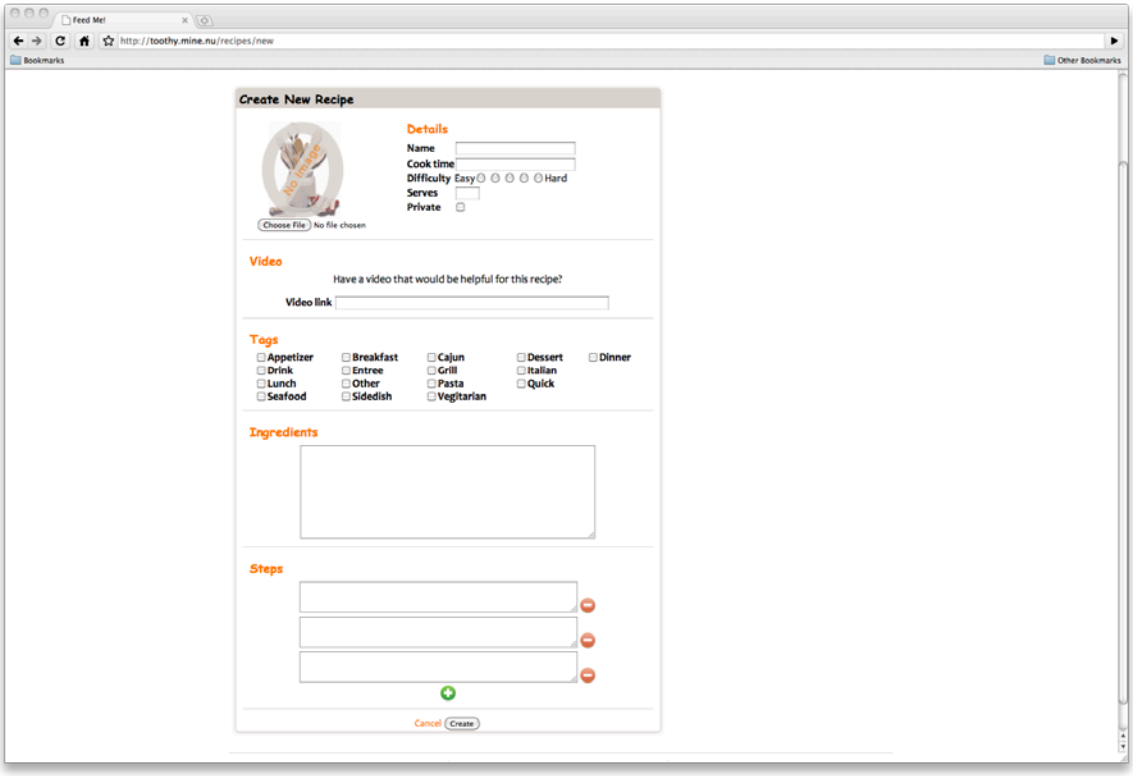

Figure 3: Recipe submission

Figure 3 shows the recipe submission page. This is the page users are brought to when adding a new recipe to the site. The main section here allows users to set recipe details such as the name, ingredients and steps of a recipe while the sections in the sidebar let the

user set any recipe associations. This is where the user would link to any helpful tutorials or media.

6. Resources

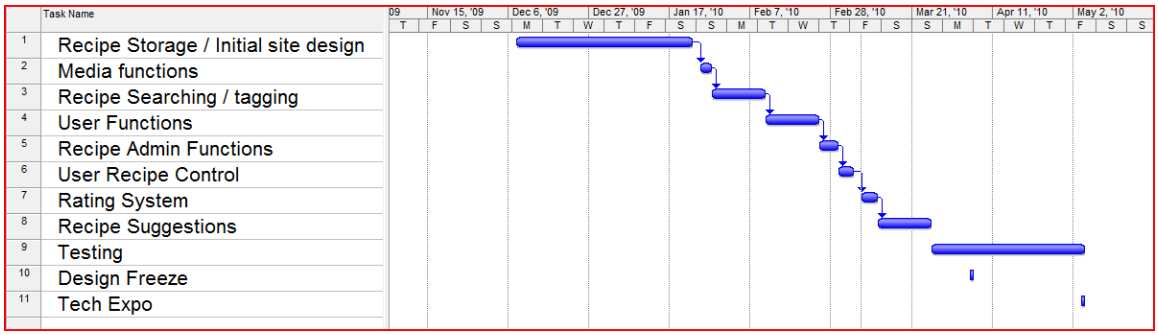

Figure 4: Timeline

6.1 Timeline: The timeline for this project, as depicted in figure 4, follows a very simple lifecycle. The first major portion of the project was planning and design. This covers the initial problem statement, as well as the major design choices, such as: sitemap, database structure, user analysis, and major tools being used. The second part of the project was to setup the development environment, the backup systems, and the tools to access the project. The final portion of the project will be the implementation. Each section of the project will have its own lifecyle. The flexibility of the Ruby on Rails framework allows each section to inherit the major design aspects from the rest of the project. This simplifies the design process because most of the projects design elements are created in the first design phase, while each sub section only needs minor adjustments. The Rails framework also automates the testing process with built in unit testing tools.

Type	Name	Personal Price	Production Price
Database	MySQL	Free	599$ / year (7)
Web Framework	Ruby on Rails	Free	Free
HTTP Server	Mongrel	Free	Free
Project Hosting	GitHub	Free	50$ / year (8)
Server OS	Ubuntu	Free	780$ / year (9)
Domain Name	DynDNS	Free	30$ / year (10)
	Total:	**-**	**1429$ / year**

Figure 5: Table of projected costs for the project

6.2 Budget: Figure 5 shows the budget layout for the project. Primarily the tools and services used either in the project itself, or for hosting the project. The budget does not show some of the tools actually used for the development of the project. It is important to note that there are two aspects to project hosting. The first is the location where the actual code is being maintained and the second is the actual production server where the final product is being hosted. Github is the service being used for the first part, and acts as a repository, providing global access to the project (so it can be quickly downloaded and worked on from any machine), as well as version control and bug tracking services. A privately run server will host the final product, but in order to access that server from the web, a domain name provided by DynDNS.com will be needed.

7. Testing

The testing was broken up into four primary sections. While there are many other types of testing available, these sections are given priority, because these tests focus on stability and useability.

7.1 Compatability Testing: The first stage of testing was to make sure that the site worked in the most common environments. For a web application these environments are web browsers such as: Internet Explorer, Chrome/Safari, and Firefox. The application

does use some newer technology, which doesn't display equally in all browsers (CSS3 and HTML5), but in these cases it was important that the application degraded gracefully for older browsers.

7.2 Useability Testing: This test series focused on having test subjects evaluate the user interface of the application. They are given a list of actions and, without guidence, are expected to complete them. The test requires that there be an observer to record where users have trouble, where they seem to expect certain elements to be, and what the test subjects general reactions are. This helps fix deign flaws and improve the overall useability of the application.

7.3 Load Testing: This was one of the more automated tests, in order to check the applications stability and reliability, a script fills the application with test data. Once the site was filled with data, there are tools to check how the site reacts when under a user load. Checking statistics such as how much each user slows down the database or webserver.

7.4 Alpha/Beta Testing: The final testing process involves allowing people to use the site before it is released to production. The alpha stage refers to testing by the development team; which because of their knowledge of the application, can have a better expectation of what should happen in a certain scenario. The beta stage follows the same process, except people outside the development team are used.

8. Risk Management

8.1 Technology Risk: The primary concern here is if a development machine fails in a way that results in complete or partial loss of the project. Besides the obvious loss of work, a loss in this way can also be harmful in that, because re-written code is not guarenteed to be the same is the original, testing must be re-run on existing code as well. In order to mitigate this risk, the project implements a two tier backup scheme. The first precaution is the development machine is backed up nightly to an external hard drive. As a second precaution the project is hosted with an external company, which implements their own backup architecture.

8.2 People Risk: Harmful or unwanted code could be added to the project, damaging the overall stability of the web application. This risk comes from the unreliabily of the development team. No matter how careful the developer may be, occasional human error is unavoidable. Careful testing helps detect these problems, and in case a problem is found the project is held in a version control system. This allows branching of the project for any experimental code, as well as revision history and the option to rollback to previous versions.

8.3 Schedule Risk: A milestone in the development life cycle could be missed, delaying the next step of the project, or leading to unsatisfactory results in the previous step with the possible consequence of not meeting a deliverable. It is assumed that various parts of the project will be more difficult than others, so in order to protect against any delays two weeks have been set aside for testing purposes and other finalizing procedures. This time

can also be allocated to any units in the project that were not given enough time in the schedule or that do not meet quality requirements.

8.4 Structure Risk: A model or other underlying structure of the web application could be poorly implemented or designed, leading to overall instability of the project. The version control system helps here to a degree, but in the case of a structural weakness, the entire project could be unstable without a way to roll back. So in order to mitigate this risk a tool in the rails framework called database migrations is helpful. Migrations implement a type of version control for the main structure of the project. Giving the ability to rollback unwanted versions of the project model without loosing progress to the other parts of the project. This helps the developer see the model revision history, and easily make changes to the model in the case of poor-design.

Risk	**Probabilty**	**Impact**	**Mitigation**
Technology Risk	Low	High	Risk Transference – Backup scheme
People Risk	High	Low	Risk Avoidance – Version control and testing procedures
Schedule Risk	Medium	Medium	Risk Mitigation – Schedule buffer
Structure Risk	Low	Medium	Risk Mitigation – Migrations

9. Deliverables

1. **Recipe Storage** – Basic storage functions, including creating new, editing, viewing and destroying recipes.
    a. **Admin functions** – Ability for a site admin to remove and manage recipes, users and media
    b. **Searching / Sorting** – Allows for recipes to be searched for and filtered based on different criteria
    c. **Recipe tagging** – Each recipe can be tagged with multiple keywords, allowing for easier searching and suggestion based features
2. **Media Functions**
    a. **Images** – One image per recipe, so that users can see what the finished product looks like
3. **User Functions**
    a. **My Cookbook** – Helps users organize important recipes into categories
        i. **Recent recipes**
        ii. **Favorite recipes**
        iii. **Recipes to try**
        iv. **Personal recipes / private recipes**
    b. **Searching based on user preferences / history** – Ability for the site to suggest recipes based on settings in a user profile or user history
    c. **Admin over owned recipes** – Users have admin rights over recipes they upload, including edit and removal privileges
4. **Social Functions**

a. **Rating system** – Users can rate a recipe

b. **Comments** – Users can post comments and suggestions for each recipe

Works Cited

1. Nielsen, Jakob. *Designing Web Usability*. Inianapolis: New Riders Publishing, 2000.

2. *Cooks.com*. The Fournet Information Network, n.d. Web. 27 Oct. 2009. <http://www.cooks.com/>.

3. *Better Homes and Gardens New Cook Book*. 11th ed. New York: Bantam Books, 1997.

4. Prabhakar, Annu. Personal interview. 27 Oct. 2009.

5. “RubyonRails.org." *Ruby on Rails*. 37 Signals, n.d. Web. 27 Oct. 2009. <http://rubyonrails.org>.

6. Russell, Mark. "23 Amazing and Open Source Ruby on Rails Applications." *CoolCreation*. Website Design Blog, 12 Feb. 2009. Web. 27 Oct. 2009. <http://www.coolcreation.co.uk/website-design-blog/ 23-amazing-and-open-source-ruby-on-rails-applications/>.

7. "MySQL Online Store." *MySQL*. Sun Microsystems, n.d. Web. 2 Dec. 2009. <http://globalspecials.sun.com/store/mysql/DisplayHomePage/ ?resid=3uN8mQoBAkgAAAukxw0AAAAW&rests=1259812136229>.

8. "Github Pricing and Signup." *GitHub*. Logical Awesome, n.d. Web. 2 Dec. 2009. <http://github.com/plans>.

9. "Ubuntu for Server Support." *Ubuntu.com*. Canonical, n.d. Web. 2 Dec. 2009. <http://shop.canonical.com/product_info.php?products_id=532>.

10. "DynDNS Pricing." *DynDNS*. Dynamic Network Services Inc., n.d. Web. 2 Dec. 2009. <http://www.dyndns.com/services/pricing/>.