

G52GRP Final Group Report

Digital Chef: Collaborative Filtering in the Kitchen

gp09-jqg:

Dhruv Gairola (dxg09u) - Project Manager

Chris Head (cxh08u) - Quality Assurance Officer

Rob Miles (rxm08u) - Technical Director

Amy Jane Wesson (ajw08u) - Documentation Manager

Chenjue Xu (cxx09u) - Head of Design

supervised by

Dr Julie Greensmith (jqg)

December 4, 2009

Contents

1	An Introduction to the “Problem”	4
2	Expanded Problem Description	5
3	Research	6
3.1	BBC Recipes	6
3.2	RecipeZaar.com	7
3.3	Supercook.com	8
3.4	Allrecipes.com	9
3.5	JamieOliver.com	10
3.6	Jamie Oliver - 20 Minute Meals Iphone Application	11
4	Results of Technical Research	12
4.1	Platform Decisions	12
4.2	Technologies	13
4.3	Web Browser Options	14
4.4	The World Wide Web Consortium (W3C)	14
5	Collaborative Filtering Technology	15
5.1	About	15
5.2	System Pre-requisites	15
5.3	Algorithms	16
5.4	Non-Probabilistic algorithms	16
5.4.1	User-based Nearest Neighbour Algorithms	16
5.4.2	Item-based Nearest Neighbour Algorithms	17
5.4.3	Dimensionality Reduction Algorithms	17
5.5	Probabilistic Algorithms	18
5.6	General concerns about all algorithms	18
5.7	Acquiring Ratings- Design Decision	18
5.8	Rating Scales	18
5.9	Cold Start Issues	19
5.9.1	New User	19
5.9.2	New Item	19
5.9.3	New Community	19
5.10	Challenges with Collaborative Filtering	19
6	Product Specification	20
6.1	Minimum - v1	20
6.2	Realistic Best - v2	20
6.3	Ideal - v3	21
7	Initial design of the proposed system and its user interface	22
7.1	Version 2	22
7.2	Version 3	26

8	Implementation Options/Designs	27
8.1	Summary of Project Description and Specification	27
9	Implementation Decisions	28
9.1	Decision Influences	28
9.1.1	Aims	28
9.1.2	Design Principles	28
9.2	Decisions	29
9.2.1	The recipes App	29
9.2.2	URL Design	29
9.2.3	Model Design	29
9.2.4	View Design	29
9.2.5	Template Design	30
10	Implementation Results	31
10.1	Implementation Points of Interest	31
10.1.1	Collaborative filtering implementation	31
11	Testing	34
11.1	Obtaining Test Data	34
11.1.1	Requirements	34
11.1.2	Meeting the requirements	34
11.2	Running Tests	35
11.2.1	Website Functionality	35
11.2.2	Website Compatibility	38
11.2.3	Recipe Search Speed	38
12	Reflections on project	39
12.1	Technical Issues	39
12.2	Time Management Issues	39
12.3	Group Working Issues	39
12.4	Success of Project	40
13	Conclusion	40
14	Evaluation	41
15	Appendices	43
15.1	Timescale	43
15.2	Meeting Minutes	45

1 An Introduction to the “Problem”

The aim of this project is to develop a software kitchen assistant tool. This tool must be able to provide recipes which match a supplied list of available ingredients. This is similar to the BBC’s recipe search¹ (Fig 1), but is not constrained to just three search items. This also involves making recommendations and provides us with an avenue to make use of the collaborative filtering technology, which is the process of evaluating recipes through the opinions of other people. The recipe evaluation allows us to provide users with recipes they are likely to enjoy. Additionally, the recipe database can be community maintained, and aspects of social networking can be implemented to encourage user participation. The main draw of the project lies in its inherent flexibility, which extends from the plethora of expansion decisions which can be made to enhance user experience.

¹<http://www.bbc.co.uk/food/recipes/>

2 Expanded Problem Description

As a group we understood that the aim of the project is to develop a software kitchen assistant tool. The tool must be able to provide recipes which match a supplied list of available ingredients. The software should provide a number of matching recipes and rank the suggestions according to how well they match. The database will also take into account users own food preferences, this allows us to introduce collaborative filtering technology to manage the recommendations.

We have chosen to use extreme programming therefore making frequent and small releases. We have specified three versions, these being minimum, realistic and ideal.

For all three versions the interface of the kitchen assistant tool will be web-based and will allow the user to select at the least three ingredients. Once these ingredients are submitted it will return a list of recipes that are ranked in accordance to how well the recipe matches the selected ingredients.

Version 1 will be a web-based application with a simple interface using only HTML, CSS and Django template markup. The online database will contain several recipes that are searchable by ingredients.

Version 2 is an extended version of v1. The web-based interface will introduce JavaScript, AJAX (asynchronous JavaScript and XML) and JSON (JavaScript Object Notation). The online database for this version will contain a vast amount of recipes that are searchable by a combination of ingredients. With the introduction of collaborative filtering we intend to create user accounts that allows the user to rate recipes and then return recommendations based on previous ratings.

Version 3 extends v2. The web-based interface will remain the same however we will create a mobile optimised interface and an Iphone application. The online database for this version will include a very large amount of recipes that are automatically updated and maintained. Recipes will be searchable by combinations of ingredients and by other tags such as vegetarian, Italian, low fat, etc. Returned recipes will be returned based on past ratings and accumulated data from the entire database. Recipes will also give recommendations for several users i.e. A recipe that alice and bob will both like. This version will include a full user system with profiles and user-uploaded recipes. The application will support social media functions such as messenger, the possibility of rating, tagging and commenting on other recipes.

The project has be divided into 5 managerial areas. These being management, technical, design, quality assurance and documentation. Every member has been given the responsibility of one of these areas and is expected to ensure all targets are met.

3 Research

3.1 BBC Recipes



The screenshot shows the BBC Recipes search interface. At the top is the BBC logo and a search bar. Below this is a teal header with the word 'Recipes'. The main content area is titled 'Search recipes' and features a large image of a dish of mussels. To the right of the image are several search filters: a text input for 'Search by up to three ingredients or recipe', a dropdown menu for 'Find recipes by programme' (set to 'All programmes'), another dropdown menu for 'Find recipes by chef' (set to 'All chefs'), and checkboxes for 'quick recipes' and 'vegetarian'. At the bottom of the search filters is a blue 'Search recipes' button and a link to 'Advanced search'.

Figure 1: The BBC Food Recipe Search Page

BBC Recipes² (Fig 1) is a web application that allows the user to input up to three ingredients and returns a list of related recipes.

BBC Recipes has two search options, basic and advanced. The basic search allows the user to input up to three ingredients with the option to find the recipe by television program or by chef. This search also provides the tagging options of quick recipes and vegetarian. However the advanced recipe search includes a wider choice of search preferences such as, the preparation method, cuisine, season and dietary requirements. After looking over the source code its very obvious that this web-based application uses an online database using a query language to return recipe results.

One main attraction about this application is the advanced search option which includes a numerous amount of tag options. This option becomes quite useful when searching through a large database as this kind of search minimizes the results. A good example of this is a search including flour, butter and sugar with the season tag being Christmas and the dietary needs tag being nut-free returning only 14 recipes. However a search including just the three ingredients returns 400 recipes.

²[BBC Recipes, 24th November 2009 www.bbc.co.uk/food/recipes/]

However, the website has its flaws. For instance, the search field is a text box which takes text input, but does not validate the input until after the search. The validation matches the input with similar words, for example for flor it will return flour. However not all typographical errors are corrected to the intended ingredient, which is a logical consequence using such a search methodology. For example typing aooples, instead of apples returned the match allows. To solve this issue perhaps an automatic-complete feature could be implemented, where as the user types an ingredient, the user is presented with a variety of completed strings from which he/she can choose allowed ingredients.

The design of the website is attractive with a good use of colour and images. However the navigation of the website is slightly tedious, the reason for this being when expanding the actual recipes on the home page the webpages content increases and therefore leaving the user to have to scroll through the webpage to view its content.

3.2 RecipeZaar.com



Figure 2: The RecipeZaar.com Home Page

Recipezaar³ (Fig 2) is a website that includes a large database of recipes and user accounts. Recipes are searchable by recipes, cookbooks, ingredients and members.

The recipe search allows the user to input n number of ingredients. The input method is a text box which includes no validation. If the user inputs flor instead of flour the search returns nothing. Additionally, the query system used for search lacks intuition for inputs of many ingredients. For example if the user inputs flour, sugar, butter, eggs, the search asks users to refine their search by selecting from a generated list of more specific

³[RecipeZaar, 24th November 2009 <https://RecipeZaar.com>]

of ingredients. However, this returned list is extremely vague, mainly because users now have to select boxes such as low-sugar apple butter (which, incredibly, was the first in the list) where the system combined the query for butter and sugar. There is no checkbox for what one would have expected, like granulated sugar. Hence, such a system- where users have to input more specific ingredients before queries can be accepted- is ineffective for a search with many ingredients because the search now requires users to refine their search by selecting from a generated list which consists of mostly absurd ingredients (e.g. sodium-free-sugar-free peanut butter).

This website introduces the use of collaborative filtering, allowing users to rate recipes. Moreover, the website encourages social networking by allowing users to have user accounts where they do a variety of tasks, like submitting recipes. When viewing a members recipe there is also the option to send a private message, submit corrections, send the recipe to the users email address or mobile device and create a shopping list. The recipe page also directs the user to other recipes like the chosen recipe. The website also has a community webpage with forums for general discussions.

The interactive design of the website makes good use of JavaScript. The colour scheme is neutral with a good use of images.

3.3 Supercook.com

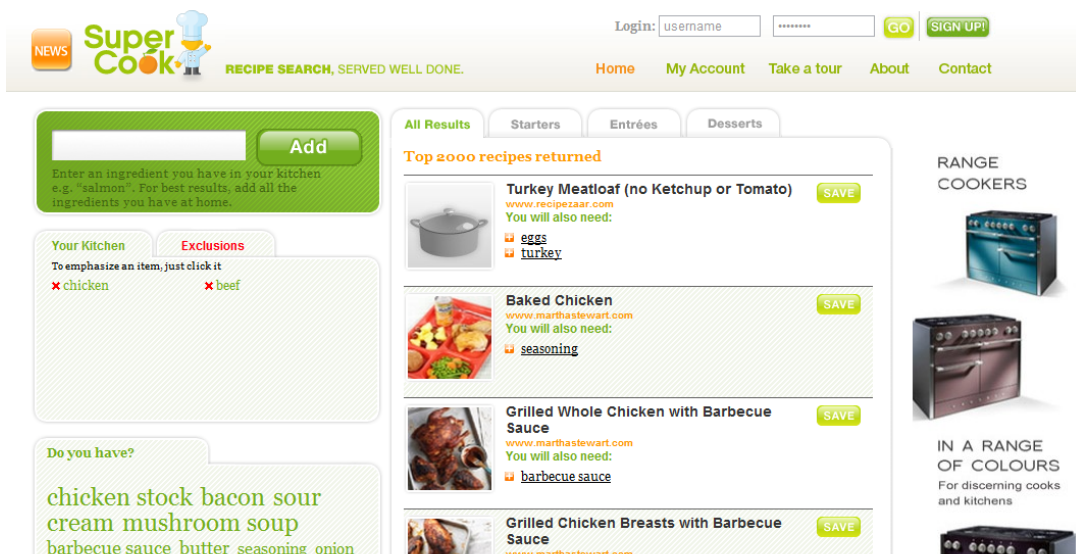


Figure 3: The Supercook.com Home Page

Supercook⁴ (Fig 3) is one of the best websites surveyed due to its immense functional-

⁴[Supercook, 17th February 2010 <https://Supercook.com>]

ity and features. Its search-box has the automatic-complete feature, which validates user ingredients. Moreover, adding multiple ingredients is done by putting the ingredients in a list one at a time on the left side of the page, while recipe results are seamlessly displayed on the right hand side of the page as ingredients are added.

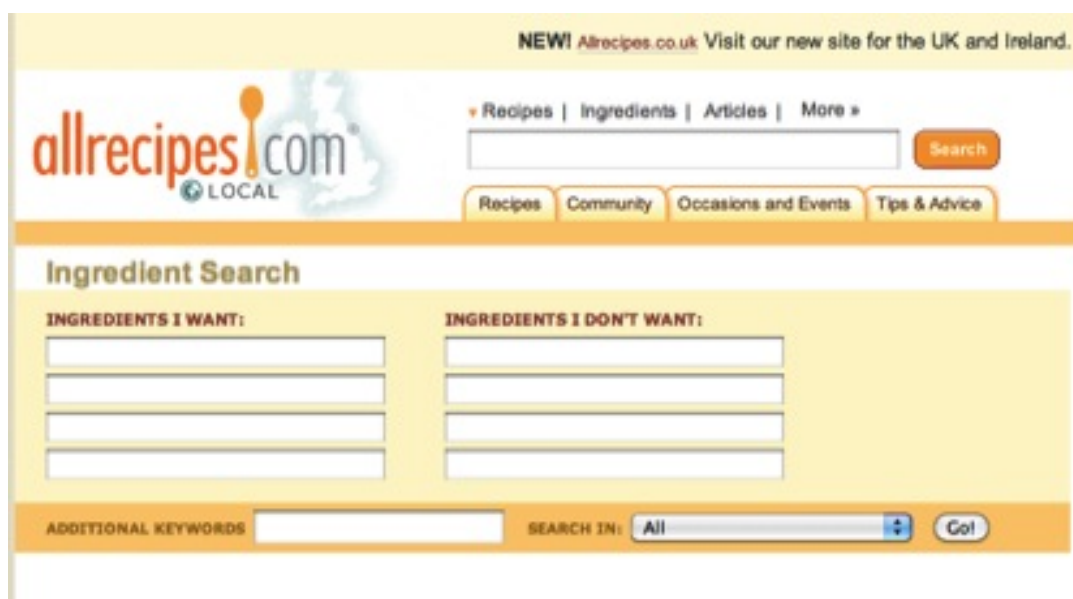
The results also inform you if you have all the ingredients you need to make the recipe. In case you do not, the returned recipe tells you what additional materials you will need in a very user friendly format. Moreover, there is a do you have area where if you have the additional ingredient displayed, then you can make a recipe without needing extra materials. This is useful because it provides a way for users to find potentially unique combinations of ingredients to complete a recipe based on the suggestion, which they would otherwise not have thought of.

You can also specify what ingredients to exclude in the search. For example, you may wish to exclude milk from the returned recipes if you are lactose intolerant.

Another atypical feature about this website is that the returned recipes are actually links to other websites, like recipezaar.com. So clicking a recipe results in redirection to another website. This introduces a problem of standards, because each website linked has its own way of displaying recipes, and the level of detail for the instructions. An improvement would have been for supercook.com to use its own database.

Moreover, the website is quite chaotic in its layout. The most jarring are the advertisements which occupy multiple areas on the webpage as opposed to a fixed area.

3.4 Allrecipes.com



The screenshot shows the Allrecipes.com website interface. At the top, there is a banner for 'NEW! Allrecipes.co.uk' and a navigation bar with links for 'Recipes', 'Ingredients', 'Articles', and 'More'. Below this is a search bar with a 'Search' button. The main section is titled 'Ingredient Search' and contains two columns of input fields: 'INGREDIENTS I WANT:' and 'INGREDIENTS I DON'T WANT:'. At the bottom, there is a section for 'ADDITIONAL KEYWORDS' and a 'SEARCH IN:' dropdown menu set to 'All', with a 'Go!' button.

Figure 4: The Allrecipes.com Home Page

Allrecipes⁵ (Fig 4) is a web application that allows the user to search for recipes using four ingredients they wish to search by, and four ingredients that they dont wish to search

⁵[Allrecipes, 23rd March 2010 <http://allrecipes.com>]

by. There's also the additional feature to add keywords and the ability to search using different tags, for example, Asian, breakfast and brunch, seafood etc

Recipes are returned and sorted by relevance, title or ratings. Each recipe has a rating, reviews and the ability to find recipes like the recipe chosen. The users submit the recipes including details such as, the prep time for the recipe, the cook time, the ready in time, servings, ingredients and directions.

Each user has a user account. The user account includes the user's home town, where they're currently living, the length of time that they've been a member, their cooking level, interests such as baking, Indian, Italian etc, cooks that they like, a recipe box, a blog and an about section.

This website has a good use of collaborative filtering. The colours used and the design of the website is minimum. The simplicity of the website allows the users to navigate freely. With the use of recipe of the day this gives the user reason to revisit the website on a regular basis.

3.5 JamieOliver.com



Figure 5: The JamieOliver.com Home Page

The JamieOliver⁶ website allows the user to search for recipes rather than inputting ingredients. The website returns a list of recipes meeting the text inputted into the search field. A recipe consists of ingredients and directions. This website doesn't use collaborative filtering as well as other recipe websites, however does include recipe comments, blogs and user profiles.

⁶[JamieOliver, 23rd March 2010 <http://www.jamieoliver.com>]

The overall design of this website is very attractive and has a good use of colours that really catch the eye. With the use of a slide show this makes the website more interactive.

3.6 Jamie Oliver - 20 Minute Meals Iphone Application



Figure 6: Jamie Oliver's 20 Minute Meals Main Interface

This application provides a list of recipes. Dishes include easy pasta, delicious soups, fast fish and many more. It then provides the user with a summary of the chosen recipe, ingredients that you'll need and the steps to follow. The images used look very appetising and handy videos are also available.

The application also provides the user with the capability of creating a shopping list using ingredients from the recipes and also adding their own shopping items.

4 Results of Technical Research

For this part of the project we looked into many different alternatives for suitable platforms, tools, technologies, algorithms and data structures. We mainly conducted our research using the internet however, we did use some of our own personal experience and preferences aswell to influence our decisions.

4.1 Platform Decisions

Microsoft Windows

The windows operating systems have long dominated the operating system industry. Approximately 90 % of users use Windows operating systems, chiefly Windows XP followed by Windows Vista. Its ease of use and engaging graphical interface is certainly an attraction. However, precisely due to its widespread usage, Windows is the prime target for malware. Microsoft however, does provide bug fixes and other help to stabilise the system. Moreover, most forms of software run on Windows.

If our group is to market our product to customers, it makes sense that we focus on Windows as the platform of choice since it is the most commonly used operating system. Moreover, if our project decides to make an application, it should be able to run in Windows, and since most software works on Windows, it is the clear choice.

Mac OSX

Although not as widely used as Windows, this operating system has a very encouraging user interface which is easy to pick up. It is claimed as being more secure than Windows, due to its UNIX base. However, recent reports suggest the Apples Snow Leopard system is less secure compared to Windows Vista and XP⁷. Of course we must take into account the comparatively fewer threats from malware on Mac OSX. Mac OSX also uses pre-emptive multitasking for all native applications to which decreases the incidence of multiple program crashes.

Linux

One of the biggest advantages of Linux over other operating systems is the Linux kernel which ensures a basic level of security. Its hardware requirements are also much lesser than Windows and Mac OSX. Additionally, Linux, being open source is a free system. Linux distributions like Ubuntu, also provide a friendly and graphical user interface for users to work with. However, latest hardware is typically slower to reach linux. Moreover, depending on the distribution, the learning curve of Linux might be daunting for users⁸.

⁷[<http://www.wired.com/gadgetlab/2009/09/security-snow-leopard/>].

⁸[<http://packratstudios.com/index.php/2008/04/06/the-pros-and-cons-of-linux-windows-and-osx/>]

4.2 Technologies

Django

A web framework based on Python language, Django is relatively easy to understand, Python being easier to program due to its natural language-like syntax. One of chief arguments for the use of Django concerned software reuse. Various existing libraries can be used to aid our software development efforts. The group software head also backed Django, and his recommendation was well received since the group could learn a new form of technology while benefiting from his expertise.

- Advantages
 - Our Technical Officer has experience developing with Django which is beneficial when developing and learning the language.
 - Python is an easy language to learn and use, with a focus on simplicity and ease of use whilst providing an elegant solution to the problem.
 - A variety of third party plugins coincide with our site's functionality, saving a lot of work by maximising code reuse.
- Disadvantages
 - Requires learning a new language.

Ruby on Rails

A web framework based on Ruby language, it allows users to create powerful applications using simple coding without compromising on the functionality of powerful languages⁹. The Rails framework also has many pre-defined libraries and functions that we may be able to use to our advantage.

- Advantages
 - Increased code reuse due to vast array of pre-defined libraries and functions available.
 - Also provides an easy and elegant solution to complex web programming problems.
- Disadvantages
 - Abstraction may mean sacrificing fine control even when it would be useful.
 - None of our group are familiar with the Ruby framework which may affect our pre-defined timetable and/or our time constraints.

⁹[<http://www.hosting.com/support/rubyonrails/faq/>]

PHP with SQL

This option was an attractive one, considering that members had some experience with PHP and SQL previously. Moreover Java, Python, C++, Ruby are normally used to create complex systems which is not necessary for us at this stage in the project.

- Advantages
 - Overall group experience with these languages.
 - Common technology means it is well supported with many tutorials and guides on usage.
- Disadvantages
 - Low level control means making large systems is complicated and difficult.

4.3 Web Browser Options

In theory, all web browsers are the same, and any page that works on one will work on all others, as long as they all support the same standards. In practice this is not the case and different browsers support different features. The main browsers right now are Microsoft Internet Explorer, Mozilla Firefox, Safari, Google Chrome and Opera¹⁰. Of these, the majority support the web standards codified by the World Wide Web Consortium (W3C)¹¹, so in order to easily gain compatibility with the largest number of web browsers, we will aim to write a W3C standard compliant site. We can test this with W3C's validator¹², which will save us having to debug the site in every browser we wish to support

4.4 The World Wide Web Consortium (W3C)

Tim Berners-Lee and the Inventor of the Web created W3C in October 1994.

The W3C mission is to lead the World Wide Web to its full potential by developing protocols and guidelines that ensure the long-term growth of the Web. W3C developed Web specifications, called Recommendations, that describe communication protocols.

The technical specifications are developed through a process designed to maximize consensus about the content of a technical report, to ensure high technical and editorial quality, and to earn endorsement by W3C and the broader community.

¹⁰http://www.w3schools.com/browsers/browsers_stats.asp

¹¹<http://www.w3.org>

¹²<http://validator.w3.org/>

5 Collaborative Filtering Technology

The final prototype of our project deals with users being able to rate recipes and receive recommendation of recipes they might like. There are a number of ways this system can be implemented with collaborative filtering technology. Collaborative filtering technology provides numerous prediction algorithms which we can use to implement such a system and acquire ratings.

5.1 About

Collaborative filtering (CF) is the process of evaluating items through the opinions of other people¹³. While the concept of collaboration is nothing new or novel- it is in human nature to take the opinions of other people- the technology itself was developed in the early 90s. The main difference between collaboration in everyday life and through the internet is that the internet allows us to utilise the opinion of large communities.

One of the earliest systems utilising the collaborative approach was Tapestry, developed by Xerox Parc¹⁴. Tapestry stored textual information, along with metadata and annotations about this information upon, which users could apply queries.

Users are recommended items based on their ratings for particular items. There are two ways a rating may be acquired- either explicitly or implicitly. Ratings are gathered explicitly when the user submits an opinion on an item, as opposed to implicit ratings which are inferred from user actions.

Collaborative filtering is especially relevant to our project since it addresses the user task of helping users find recipes that he/she might like. The functionality we would like our system to have is that of constrained recommendation, as opposed to dealing with prediction (where given a particular recipe, calculate its predicted rating). Constrained recommendation is suited to our project because it allows users to input particular constraints, in this case, the recipe ingredients, and from the list of recipes generated, recommend users the recipe. Provided the numerous number of recipes in the database, and given a users limited attention span, it would be logical to return a recommendation which the user is more likely to appreciate.

5.2 System Pre-requisites

However, this technology will work effectively provided the domain it is applied to meets certain criteria. In our case,

1. The database should hold many recipes If too few recipes then there is no point of recommendations.
2. There must be many ratings for a recipe For a useful recommendation, the recipe needs enough ratings. Assuming each user rates only few recipes, and the large

¹³Schafer J.B., Frankowski D., Herlocker J., Sen S.: Collaborative Filtering Recommender Systems

¹⁴Goldberg D., Nichols D., Oki B.M., Terry D.: Using Collaborate Filtering To Weave An Information Tapestry. Communication of the ACM, 35(12): pp 61-70

database of recipes, this implies that there must be a sufficiently large amount of users.

3. Users must rate multiple recipes. Else, we will be unable to relate recipes to each other.

5.3 Algorithms

Algorithms can either be probabilistic or non-probabilistic. Probabilistic algorithms use probability distributions to compute, for example ranked recommendation lists. However, non-probabilistic algorithms models are more popular with practitioners.

5.4 Non-Probabilistic algorithms

5.4.1 User-based Nearest Neighbour Algorithms

Predictions are generated for users, u , based on ratings from similar users, called neighbours. By analysing all the users neighbours, n , a rating can be predicted for a particular item, i , for the user. Neighbours who are more similar to the user will have a higher weight in calculating the rating. For instance, if we take the average of all the neighbours ratings for a particular item, we can generate a prediction elucidated by Equation 1, where r_{ni} is n 's rating for i .

$$pred(u, i) = \frac{\sum_{n \in neighbours(u)} r_{ni}}{\text{number of neighbours}} \quad (1)$$

However Equation 1 is not expressive enough because it ignores the reality that some neighbours have a higher level of similarity to u than others. If we rely more on such neighbours then our predictions are likely to be more accurate. Improving Equation 1 gives us Equation 2, where $userSim(u, n)$ measures the similarity between a specific user and a neighbour. This similarity can be described by the Pearson correlation.

$$pred(u, i) = \sum_{n \in neighbours(u)} userSim(u, n) \cdot r_{ni} \quad (2)$$

Still, if the neighbour's similarities do not add up to one, the prediction will be scaled wrongly. So we normalize Equation 2 to give Equation 3.

$$pred(u, i) = \frac{\sum_{n \in neighbours(u)} userSim(u, n) \cdot r_{ni}}{\sum_{n \in neighbours(u)} userSim(u, n)} \quad (3)$$

We have not yet taken advantage of users attitude towards ratings. Ratings from enthusiastic users against that of cynical users might be different, although both ratings might mean the same thing ("I like the recipe"). Hence, we need to *average adjust* for users' mean ratings, given by Equation 6.

$$pred(u, i) = \bar{r}_u + \frac{\sum_{n \in neighbours(u)} userSim(u, n) \cdot (r_{ni} - \bar{r}_n)}{\sum_{n \in neighbours(u)} userSim(u, n)} \quad (4)$$

Unfortunately, there are many caveats with this algorithm. If there are few ratings, then the matching of a users and a neighbours ratings will be biased, such that a particular neighbour will start heavily influencing the users neighbourhood. Additionally, the formula for this algorithm does not account for popular items. If numerous people agree that a particular recipe is excellent, then this data is less important than one in which users agree on a recipe of debatable taste. Moreover, naive implementations of this algorithm have linear time and memory requirements. Although, there are techniques which help alleviate such requirements (e.g. clustering) they also have their own limitations.

5.4.2 Item-based Nearest Neighbour Algorithms

Conceptually similar to user-based nearest neighbour algorithms, this algorithm focuses on acquiring ratings based on similarities between items. A prediction for a user u and item i is composed of a weighted sum of the users ratings for items most similar to i , described by Equation 5, where $itemSim()$ is a measure of item similarity.

$$pred(u, i) = \frac{\sum_{j \in relatedItems(u)} itemSim(i, j) \cdot r_{uj}}{\sum_{j \in relatedItems(u)} itemSim(i, j)} \quad (5)$$

Adjusted-cosine similarity is the most popular and accurate¹⁵ way to calculate similarity between items. It is very similar to the Pearson correlation that is used with user-based nearest neighbour algorithm. There exists reports which suggest that item-based nearest neighbour algorithms give more accurate predictions compared to user-based¹⁶.

A big disadvantage is the complexity of the algorithm, whose size could be the square of the number of items, although, there exists ways to prune the size of the model for example using a minimum number of coratings. However, pruning causes difficulty prediction. Additionally, items cannot have too few ratings as this may lead one item to heavily influence a prediction.

5.4.3 Dimensionality Reduction Algorithms

These algorithms help reduce the complexity of very large systems with a lot of items. They map the item space to underlying tastes of the user, which diminishes the system runtime complexity. Typically, a vector based technique like vector decomposition can be used to extract these tastes. This way, a prediction can be found reasonably.

However, techniques like vector decomposition require complex mathematical computation to produce the taste-space. Although heuristic methods help in updating this taste-space to avoid constant recalculation, software maintenance is hard due to this complexity. Reportedly this reduction in complexity is not a significant enough improvement.

¹⁵Schafer J.B., Frankowski D., Herlocker J., Sen S.: Collaborative Filtering Recommender Sytems

¹⁶Sarwar B., Karypis G., Konstan J.A., Riedl J.:Item-based Collaborative Filtering Recommendation Algorithms. Proceedings of the 10th international conference on World Wide Web. (2001) Hong Kong. ACM Press p.285-295

5.5 Probabilistic Algorithms

Probabilistic algorithms employ well established concepts of probability to make predictions. Bayesian-network models provide the preferred framework to generate reliance among users or items and can be implemented as decision trees to represent probability tables, for example. Expectation maximisation algorithm¹⁷ also uses Gaussian probability distributions to extract user underlying tastes.

One advantage of these algorithms is that not only can they help calculate the most probable rating but they also help compute the plausibility of the rating.

5.6 General concerns about all algorithms

If there are too few ratings then none of the algorithms will yield any useful data due to biasness. There are a number of techniques that help mitigate such problems. For example you can choose to discard items with ratings which are below a certain number although, this will reduce the scope of the collaborative filtering system.

5.7 Acquiring Ratings- Design Decision

There are two way to collect ratings- explicit and implicit. Explicit ratings are usually more accurate description of the users preferences however, this requires some input from the users end. It was previously believed that the users would rarely choose to spend their time rating an item, however, experience has refuted this belief as exemplified by websites such as YouTube. Moreover, incentives can be used to entice users into rating, for example, reward points.

In contrast, implicit ratings require no input from the user but they usually are not as accurate as explicit means. For example, the amount of time the user spends reading a recipe might be an implicit mean of ratings collection (the more time the user spends reading the recipe, the more he/she will likely like the recipe), but this may be inaccurate as the user may not like the recipe after reading it. However this uncertainty will be assuaged if you are able to collect a large amount of such implicit data.

5.8 Rating Scales

Another issue is selecting rating scales. If you provide a rating scale with more options, it will provide more information about the user. That is not to say that you provide a rating scale with an unnecessarily large amount of options, which may not add value to the system. Importantly, you must consider the needs of the users. If you provide too few ratings on the scale, users may find that they cannot express their opinions accurately.

¹⁷Hoffmann T.: latent Semantic Models for Collaborative Filtering. ACM Transactions on Information Systems (TOIS)(2004) 22(1)

5.9 Cold Start Issues

These issues refer to when the system is unable to provide a useful recommendation to the user due to an initial dearth of ratings. This can occur as the following scenarios:

5.9.1 New User

No specific predictions can be made to the new user since he/she has not made any ratings yet. This can be overcome, for example, by asking the user to rate some initial items before they can register the service or by obtaining demographic information from the user and matching his/her ratings with other users of similar demography.

5.9.2 New Item

The new item will initially have no ratings so it will not be recommended. This can be overcome by recommending items through non-CF techniques like content analysis¹⁸ or random selection of new items and asking for ratings on those items.

5.9.3 New Community

If there are no ratings then the system cannot provide recommendations to users who seek this service. User retention will diminish. A solution is to provide rating incentives to a small subset of the community, before expanding the service to the entire community.

5.10 Challenges with Collaborative Filtering

The obvious challenge is that of privacy. For a system provide accurate recommendations, it is necessary as much information about the user as possible. However, if a user divulges too much information, he/she is at risk if the central database containing this information is compromised. Additionally, the user should trust the particular website to not misuse user information.

Another issue is of trust. Users can purposely give artificial ratings which are not representative of their true opinions. Companies can manipulate recommender systems as well by overwhelming the system with favourable ratings of its own products¹⁹.

¹⁸r53 Sarwar B., Karypis G., Konstan J.A., Riedl J.:Incremental SVD-Based Algorithms for Highly Scaleable Recommender Systems. Proceedings of the Fifth International Conference on Computer and Information Technology (2002)

¹⁹[6 BBC News Online, "Sony Admits Using Fake Reviewer." June 4, 2001 <https://news.bbc.co.uk/1/hi/entertainment/film/1368666.stm>]

6 Product Specification

In response to our initial meeting with our client and after reviewing the "Problem Description", we have put together a brief specification stating the requirements for the software at three levels which have been agreed by our client. The levels can be considered as versions as they will be implemented in this way (i.e. Minimum - version 1, Realistic Best - version 2 and Ideal - version 3²⁰). These requirements will be used as a basis for the design of the respective versions and the designs will be created with these requirements in mind.

6.1 Minimum - v1

The requirements listed below are the minimum requirements to make the software work which does not include extra functionality desired by our client. This will be the version 1 of our prototype and we will use this as a base to build from. Furthermore, the creation of different versions with the intent to expand on previous versions will allow for continuity between languages and eliminating the need to 'start from scratch'.

- Contains several recipes in a database.
- Has a Web Interface.
- Recipes are searchable by ingredients (greater than 3 ingredients).

6.2 Realistic Best - v2

The requirements below are what we have identified as the most realistic outcome of the software within the time constraints. This would meet all the minimum requirements outlined by the client and some desirable ones as well. This will essentially be an expanded/upgraded version 1.

- Has a database of around 900 recipes.
- Has a clean, attractive web interface.
- Recipes are searchable by combinations of ingredients.
- Has user accounts.
- Allows users to rate recipes.
- Gives recipe recommendations based on past ratings.

²⁰It should be noted that due to the nature of creating a solution there are likely to be intermediate sub-versions e.g. 1.2, which will be used solely in the repository.

6.3 Ideal - v3

Below are the requirements for the software if it were to be the ideal solution to the problem. This list includes existing requirements from v1 and v2 as well. Note that it is likely that only few of these requirements will be implemented in the final version.

- Has a very large, automatically updated and maintained database of recipes.
- Has a clean, attractive and user-friendly web interface.
- Has a mobile device optimised interface and an iPhone application.
- Recipes are searchable by combinations of ingredients.
- Recipes are searchable by other tags: 'vegetarian', 'italian', 'low fat' etc.
- Has a full user system with profiles and user-uploaded recipes.
- Supports social media functions.
- Allows users to rate, tag and comment on recipes.
- Gives recipe recommendations based on past ratings and accumulated data from the entire user base.
- Gives recipe recommendations for several users i.e. 'A recipe that Alice and Bob both like'.

7 Initial design of the proposed system and its user interface

Based on the design of version 1, the current version is still a combination of the colour white (for the background) and orange (for miscellaneous design features). The white background leverages on its simplicity and clarify, this being an important criteria for retaining the attention of the user. Moreover, the orange colour is reportedly simulating for the users appetite²¹.



Figure 7: New Logo for Version 2

The logo for version 1, however, has been replaced. The new logo (Fig 7) also uses the letters "Digi Chef" as the banner, but instead of just typing the banner in a simple font, we use several ingredients to sketch up these letters. Therefore, the logo can explain our website's main function, search by ingredients and you will get recipes for delicious food.

7.1 Version 2

Version 2 is an upgraded version of the original version containing more functions (described by the Product Specification). With the use of technology such as JavaScript, the web interface looks more professional.

For version 2, our website has a larger database that contains over 900 recipes. For such amount of recipes it would be effort for the user is we were still using the drop down menu's for the search. Therefore, we have implemented a fancy search that auto completes text inputted and makes each ingredient a tag.

As shown in Fig 9, the user is able to search for recipes using more than three ingredients. When the user inputs an ingredient he/she is required to press enter to tokenise each ingredient. If the user wishes to remove the ingredient from the list, this is done by clicking the close icon at the top of the token.

Furthermore, for version two, users are allowed to create their own account to rate the recipes. So a register section is presented in the right column of the website, below this section is a part that exhibits the user of the week.

²¹[<http://desktoppub.about.com/cs/colorselection/p/orange.htm>]

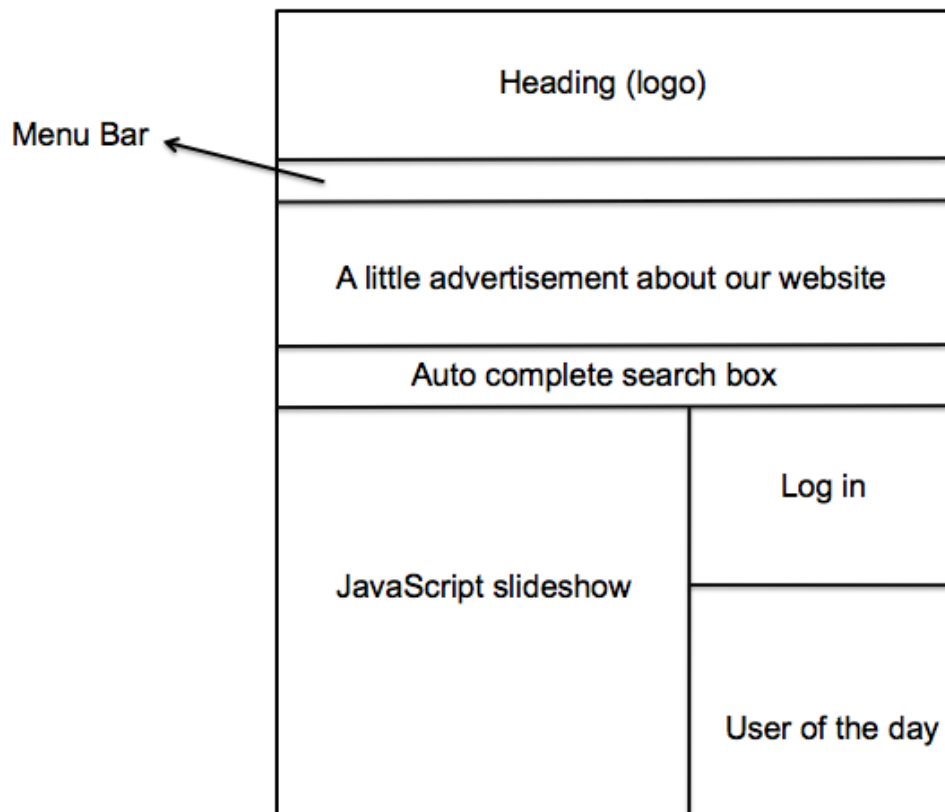


Figure 8: Layout of the Home Page



Figure 10: Auto Complete Search Box

Fig 10 shows the JavaScript slideshow, the slideshow demonstrates a list of randomly

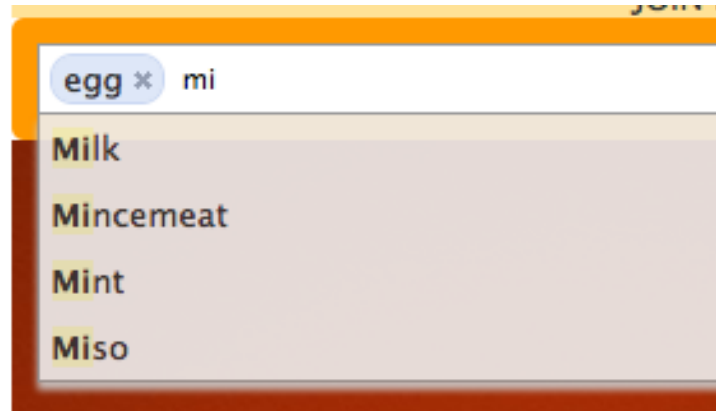


Figure 9: Auto Complete Search Box

selected recipes with their image and link.

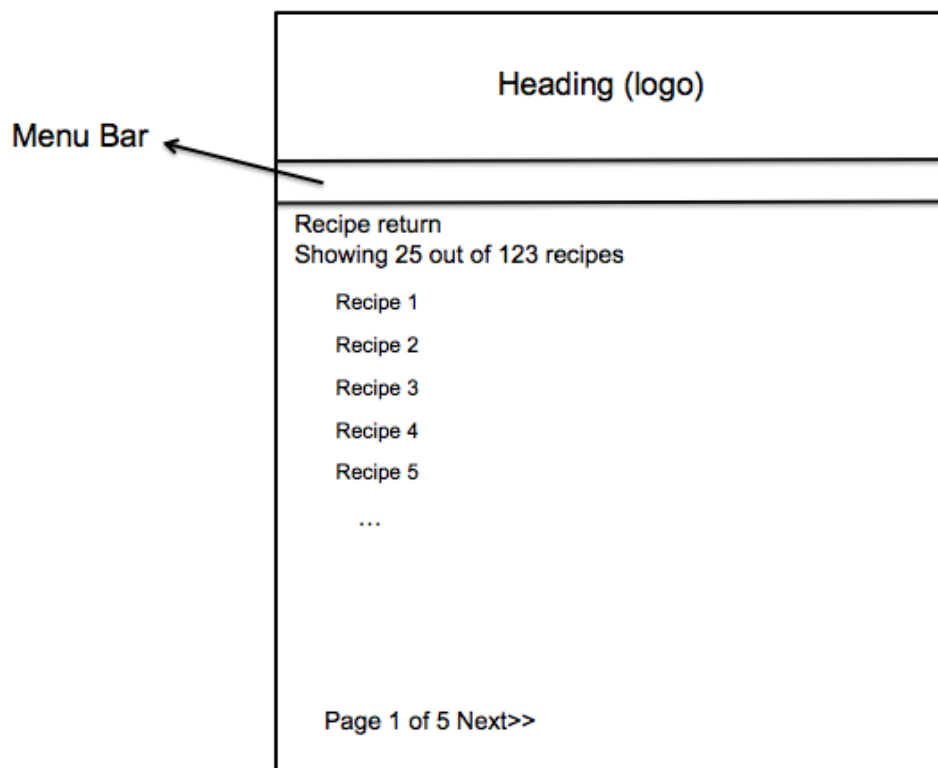


Figure 11: Layout of the Recipe List Page

Fig 11 contains a list of returned recipes matching the list of inputted ingredients. Each recipe will contain at least one of the specified ingredients. Upon recipe selection,

the specific recipe page will be displayed.

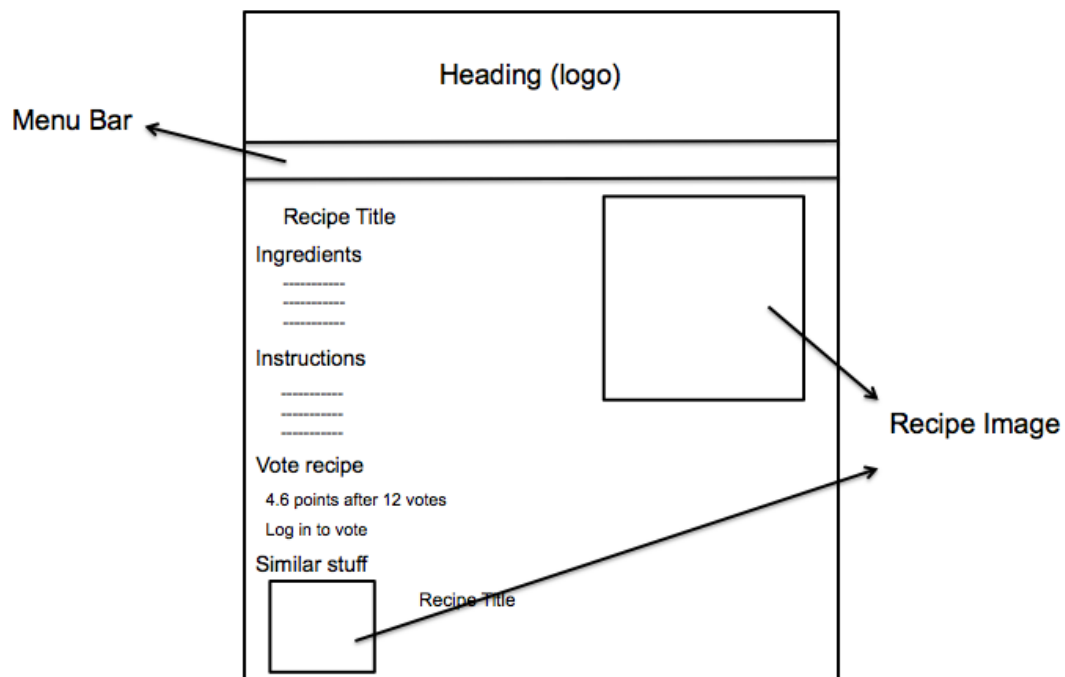


Figure 12: Layout of the Recipe Page

The recipe page (Fig 12) contains recipe details, this including the recipe name, an image, ingredients, instructions and the recipe tags.

With the implementation of Collaborative Filtering we have included a list of similar recipes and the ability to like, hate or clear a vote when the user is logged in. The similar recipes section returns up to five related recipes together with their images.

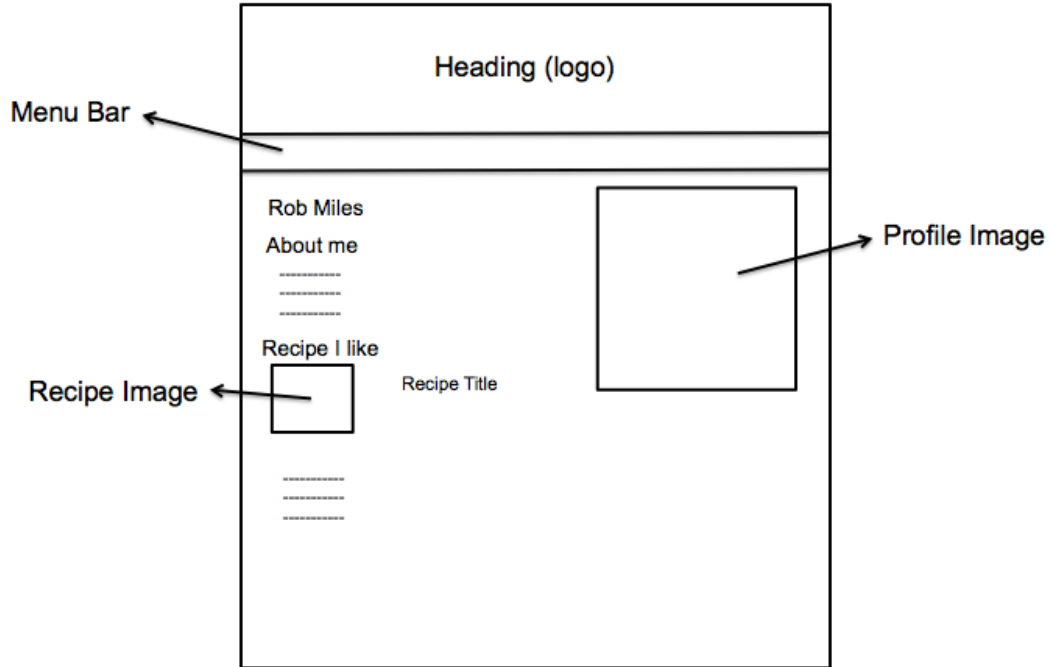


Figure 13: Layout of the Profile Page

When a user registers with Digi Chef, each user is provided with their own profile page (Fig 13). The profile page includes an about me section and a list of recipes that the user has liked.

7.2 Version 3

The ideal version of our product involves the implementation of a variety of possible functionalities. For instance, for the web based interface, users could choose to click tags and get a list of recipes that contains that particular tag. Additionally, recipes could be made searchable by not just ingredients but also, for example, the type of cuisine (Chinese dish) or whether recipes are vegetarian or non-vegetarian. Moreover, a mobile application could be developed.

Social networking would be another possibility, with users being able to interact with other users and leave comments on the profile page as well as the recipe page. Users might be able to upload their own recipes and receive ratings from other users. The possibility for improvements are abundant.

8 Implementation Options/Designs

8.1 Summary of Project Description and Specification

The project description implies the implementation of some form of a database, either online (i.e. on a website) or offline (i.e. run on a local machine as an executable program). There are no specific details as to which language, layout or structure etc, are to be used when creating the solution. There are also no details as to which Operating System the solution should be created for and whether additional software/hardware is allowed. Considering this we have taken it open ourselves to discuss and choose what we thought was a suitable target platform and have also discussed availability of software/hardware needed to create the solution. During an initial meeting with our client concerning the problem specification/requirements it has become clear that the preferred solution is to create a website with a database backbone. This can be implemented in many different styles/languages which we have also discussed extensively.

We have created the Problem Specification (section 6 on page 20) with the intent that at each stage, the structure and format of the solution allows successive stages to be completed without changing the entire structure too much. This prototyping adheres to a large extent with the concept of Extreme Programming, where frequent releases introduce checkpoints where new customer requirements can be adopted. Additionally, the focus will be on upgrading versions of prototypes. To accommodate this dynamic character of our project, we need a framework which ties in the database with all the different elements of the website.

9 Implementation Decisions

9.1 Decision Influences

9.1.1 Aims

The aims of version 1 have a strong influence over implementation decisions. They are as follows;

1. To be a complete working release of a usable piece of software
2. To allow the team to familiarise themselves with the tools and systems to be used for later versions
3. To explore the capabilities of those systems, to inform and inspire later decisions.

9.1.2 Design Principles

The project is being developed using a version of the Extreme Programming (XP) Methodology. XP's software development principles have an impact on the software design principles of projects developed using it.

Similarly the Web Framework Django has its own set of design philosophies²² which also influence the project's design principles.

The principles of XP and Django are quite similar and complement one another quite well, so it is possible to abide by both sets of principles without contradictions.

Some of the XP/Django principles that have the most influence on implementation decisions are listed below;

DRY Don't Repeat Yourself

Every piece of knowledge must have a single, unambiguous, authoritative representation within a system.

YAGNI You Aren't Gonna Need It

Always implement things when you *actually* need them, never when you just *foresee* that you need them

Maximise Code Reuse If two bits of code look similar, move them out into a more general function. If two functions do a similar thing, merge them. This keeps redundancy low.

Both XP and Django have a strong basis in philosophy and principles, and while they both leave the developer the freedom to choose their implementation decisions, they are designed to work best with implementations that follow their principles.

²²<http://docs.djangoproject.com/en/dev/misc/design-philosophies/>

9.2 Decisions

9.2.1 The recipes App

Recipes are the only thing that version 1 does, so it would make some sense to simply have the project as a whole perform the recipe functions, and not use any apps. However, the functionality of the site was put in a **recipes** app for two reasons. Firstly, it is best practice in Django to have all code in conceptually distinct, reusable apps, to maximise potential code reuse. Secondly, as one of the aims of this version is to introduce the team to working with Django, and apps are a major part of Django, it made sense to use apps even if they are not strictly necessary.

9.2.2 URL Design

The URL design is intended to be very simple and readable. In accordance with Django URL design principles, there are no filename extensions in URLs.

9.2.3 Model Design

The only implementation decision of note in the model design is the use of a python **property** to handle recipe tags. A **property** is a python language construct that behaves as though it is a class variable, but behind the scenes calls a getter or setter function when it is fetched or assigned to. This was used because, although it makes the model less readable, it makes all of the code that deals with the model far more readable, and it is this code which is more complex and benefits more from simplification.

9.2.4 View Design

The `recipe_list` View

There are 2 views that simply show a list of recipes:- **recipes_all** (the view of all recipes on the system) and the results section of **search** (the view of all recipes that meet the search terms). In order to maximise code reuse, the functionality of displaying a list of recipes was taken out into a separate **recipe_list** view, which is called by both **recipes_all** and **search**.

The `search` View

The search is deliberately the simplest search possible that meets the specifications. The set of results is simply the set of recipes which contain any of the ingredients searched for. This will be radically improved in later releases.

9.2.5 Template Design

“The most powerful – and thus the most complex – part of Django’s template engine is template inheritance. Template inheritance allows you to build a base “skeleton” template that contains all the common elements of your site and defines blocks that child templates can override.”

– Django’s Template Documentation²³

Template Inheritance provides a good opportunity to maximise code reuse, but it was not used in version 1 in an attempt to keep template design simple.

²³<http://docs.djangoproject.com/en/dev/topics/templates/>

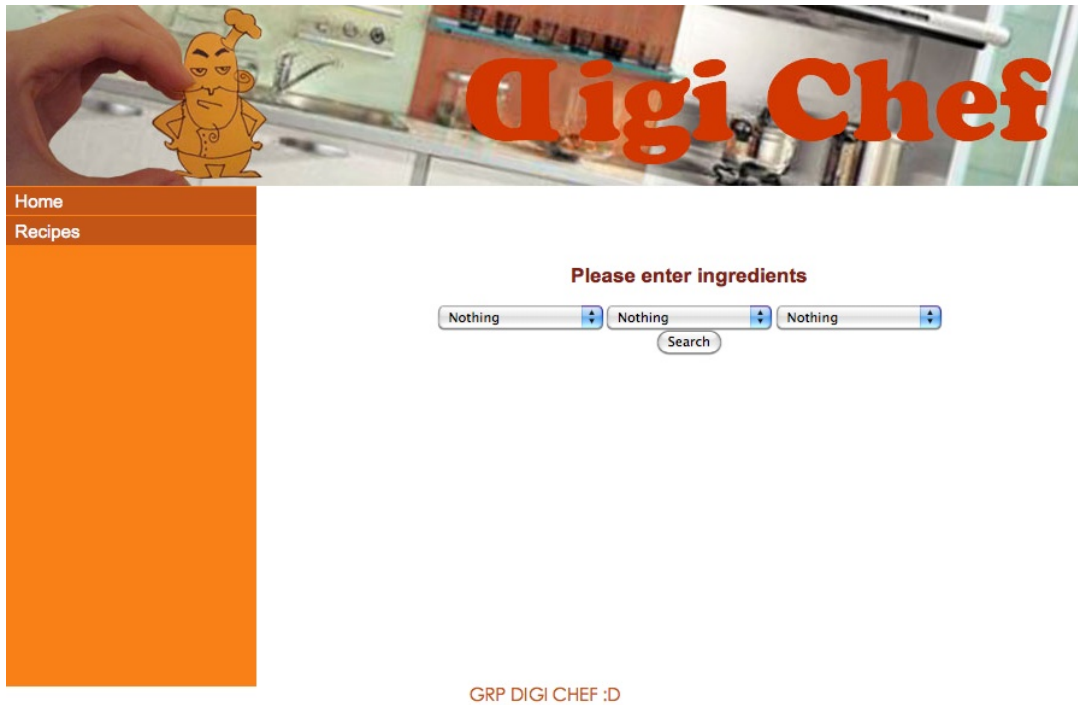


Figure 14: Homepage

10 Implementation Results

Version 1 largely adheres to the design specification.

As mentioned, white and orange are the colours of choice, along with an appealing logo. The design is simple and effective in meeting the needs of version 1. (Fig 14) shows the drop down menus for ingredient selection with the search option.

(Fig 15) shows the result of ingredient selection by the user. The recipe list was generated based on the ingredients selected by the user.

(Fig 16) shows the result of clicking on a recipe. Notice the Recipes button on the far left hand side of the web page. Clicking that generates the list of all recipes in the database.

The group thoroughly scoured the website of bugs by doing exhaustive testing of all the possible ingredients and checking if the returned list matches the input ingredients. Exhaustive testing was possible for version 1 due to the small database and only three ingredient boxes. However, as our system gets more complicated, this form of testing will be unwieldy. A better method of testing will be required then.

10.1 Implementation Points of Interest

10.1.1 Collaborative filtering implementation

The implementations of collaborative filtering methods in the module `django-recommender` were not as suitable for our needs as they first appeared, and had to be modified.

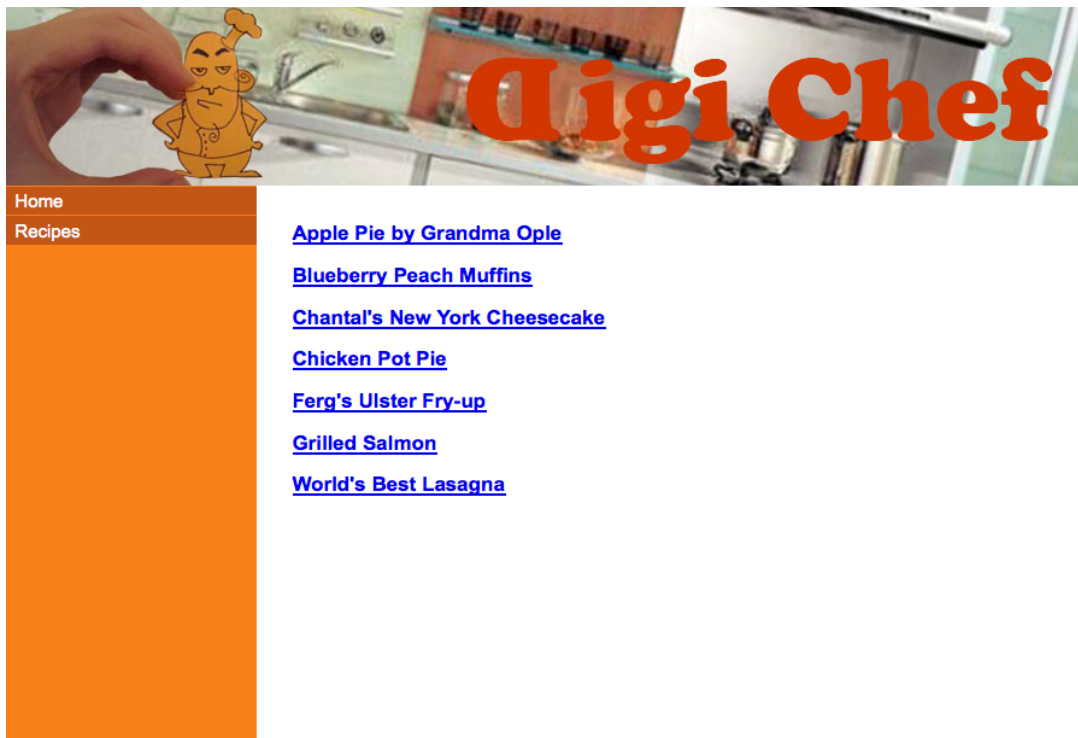


Figure 15: Search results

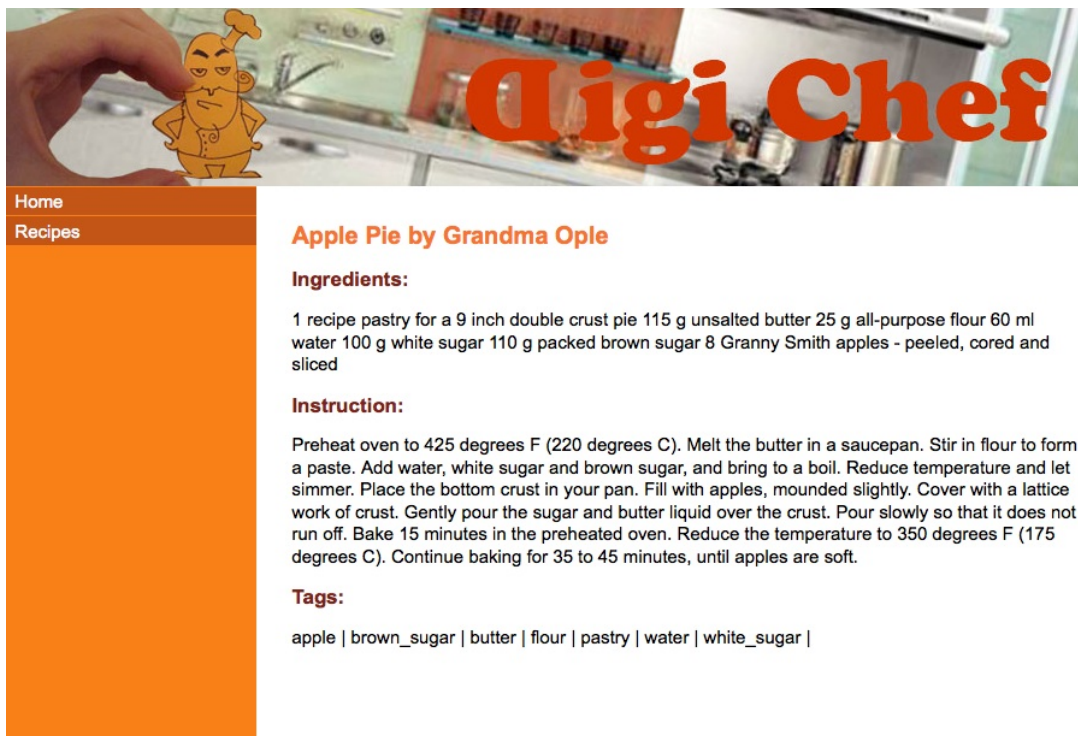


Figure 16: Recipe page

Some of these modifications were simple, for example there was no easy way to search for a recipe based on a set of tags, which we needed to implement search. A new method was adapted from an existing method which found items similar to a given item. The original method used the tags of the given object to do the similarity comparison, so the new method skipped the stage of extracting the tags from the given object and had the tag list be passed in as an argument in place of the object.

Others were more difficult. The method designed to recommend recipes to users was designed to work on tagged users. The idea here was that users would tag themselves with things they like. For example in a film rating system, action fans would tag themselves with ‘action’, the same ‘action’ tag applied to action movies. In this context that made little sense, as people are rarely ‘fans’ of individual ingredients, and no-one is going to want to tag themselves ‘onion’. So I wrote my own rating prediction function based on the equations in the collaborative filtering research section of the report, from scratch.

The method I used was the final user-based nearest neighbour method in the research:

$$pred(u, i) = \bar{r}_u + \frac{\sum_{n \in neighbours(u)} userSim(u, n) \cdot (r_{ni} - \bar{r}_n)}{\sum_{n \in neighbours(u)} userSim(u, n)} \quad (6)$$

A direct translation of this function into code was implemented. This proved to be extremely inefficient, taking approximately 1.5 to 2.0 seconds to rate each recipe, suggesting a total recommendation time for the database of 24 to 31 minutes. This was obviously unacceptable, so optimisations were sought.

A lot of things were being unnecessarily recalculated each time. Some of these things, like average vote for a user, or a 1-dimensional array of all votes for a user, could be calculated once per vote and stored in the database, so these calculations were moved to the voting part of the operation. This introduced a small delay in voting, which is far less of a problem than it would be here, because voting is done with AJAX, so the user isn’t interrupted, and voting is very fast anyway. These re-factorings made the method run on one tenth the original time.

Some things, like the sum of similarity ratings to neighbours, had to be calculated each time recommendations were sought, but not for each recipe, so these were moved to be pre-calculated before the loop over recipes. Moving out the similarity sum calculations doubled the running speed again, and pre-calculating individual user similarities caused further drastic speed increases. The final implementation operated 171 times faster than the original.

11 Testing

Testing the product is an essential component of any good software development, and this section deals not only with the details of the various tests which we conducted on our software, but also highlights the issue of obtaining suitable test data to run the tests on.

11.1 Obtaining Test Data

11.1.1 Requirements

In order for testing to be effective for a database-driven project of this type it was necessary to have a large database of recipes, for several reasons.

Firstly, performance and scalability, while not explicitly in the specification, are implicit goals for any software implementation. A large database is required to test the system's performance on large quantities of data.

Secondly, collaborative filtering systems work better the more data they have. Filters, by definition, act by filtering out large amounts of data leaving only the best matches. Having more data both increases the chance of good matches existing in the dataset, and increases the amount of available information that the system can use to find those matches.

The database also had to consist of real data. A large database of randomly generated fake recipes would have been easy to create, and could be used to test scalability and performance, but would not have worked for testing filtering quality. The only way to test the *quality* of the output of a collaborative filtering system is for a human to assess it for utility and reasonableness. A human would find it difficult to assess the quality of the output from a system that used a random database.

After assessing the datasets of other recipe sites and the technical properties of the platform it was decided that a database of approximately 1000 recipes would be suitable for testing. Due to the size of this requirement it was decided that manual data entry was off limits, as it would be far too time consuming.

For collaborative filtering to be possible, the recipes also need to be tagged with what ingredients they contain, as it is this data that the system uses to draw connections between related recipes.

Finally, the design team requested that the recipes have images related to them, to improve the aesthetic qualities of the site and differentiate intuitively between recipes.

So to summarise the technical requirements;

A database of about 1000 real recipes, each with relevant semantic metadata representing their ingredients, and a relevant image, *without* any human input.

11.1.2 Meeting the requirements

Obtaining raw recipe data was achieved by use of a recursive web-spider. All recipe detail pages of the website `AllRecipes.com` were downloaded as html files. The required

data were scraped from the resulting 941 html files using a python scraping script, which used the XML/HTML parsing package BeautifulSoup to extract the data from the files in bulk, reformatted the data a little, and interfaced directly with the django database API to input the recipes into the database.

Applying tags to represent ingredients was less simple. The solution would require automated semantic analysis, as the ingredients are listed in an unknown format. For example, the ingredient ‘onion’ could be listed in many different ways; ‘1 onion’, ‘One onion’, ‘One large onion’, ‘1 chopped onion’, ‘Onion (peeled and chopped)’, ‘1 cup chopped onion’, ‘3 onions’. To solve this problem a novel approximate solution was invented, making use of the semantic knowledge engine TrueKnowledge, which exposes an XML web API to developers. Every word in the ingredients sections of every page was extracted, de-pluralised and put into a `set` data structure, to prevent duplication of API queries, of which a limited number are allowed in a given timeframe. Each word was then loaded into an API query which effectively asked ‘*is <word> food?*’. Recipes were then tagged with every word in their ingredients list which was considered to be food. In this way the system was able to know that words like ‘chopped’, ‘peeled’, and ‘cup’ do not warrant tags, while ‘onion’ does. The system is somewhat approximate, as strings such as ‘30g onion powder’ will be tagged simply ‘onion’, but it is good enough for testing data.

To find suitable images, a custom script was written to query Google Image Search. Since Google’s image API is licensed for live web services not bulk data acquisition, it could not be used. Google implements some measures to prevent Google Image Search from being accessed programmatically, but these were circumvented by User Agent spoofing. Initially there were issues with images too large, too small, or on occasion too obscene to be used, but soon the right settings were found to ensure medium-sized, safe images. The intention was to provide images which were distinct and food related, but the system consistently surpasses that, producing images which are generally attractive, professionally photographed and surprisingly accurate, even for complex recipes. The images are of many different aspect ratios, which is useful for the design team, as the images in a production system would be provided by the users, so it is important to ensure that the page styling works with arbitrary images.

11.2 Running Tests

To make the testing of our product comprehensive, it was decided that we run three types of broad based tests dealing with website functionality, website compatibility and recipe search speed.

11.2.1 Website Functionality

The functionality of the website needs to be tested. It is of dire importance that the different functions of our website adhere to their specifications and that the navigation around the website works as specified.

The functionality of the website will be tested using black-box-testing. The testing will primarily be done by using three different input data for each area where input is

accepted. The three different inputs will be one of each: valid, invalid and extreme. Valid data is data that is usually accepted by the system and would be 'normal' data by a user. Invalid data is data that is not accepted by the system and should create some kind of error message. Extreme data is valid data that is on either end of the spectrum of what is normally accepted. An example of this could be testing whether a data field, which asks for a password of up to 12 characters, will actually accept 12 characters. The output of entering these three different input data will then be compared to the expected outcome. If the outcomes match then that particular test has been passed. Below is the formal test plan that outlines the actual tests and what the expected outcomes of these tests are.

The tests listed 1.1 through 1.3 concern the search box, 2.1 through 2.5 concern user accounts and 3.1 through 3.2 are miscellaneous general tests.

Test Id	Description and Process	Test type and Data	Expected Outcome
1.1	Test that the search box will accept and auto-complete with three common, tagged ingredients by entering the three ingredients and clicking the search button.	Valid/normal data. Data - <i>Cheese, Egg, Milk.</i>	The search bar should accept the input data with the auto-completion function and the browser should move to the recipe results page for that query showing relevant recipes.
1.2	Test that the search box accepts 10 common tagged ingredients by entering them into the search bar	Extreme data. Data - <i>Cheese, Egg, Bacon, Milk, Onion, Mushroom, Tomato, Salt, Butter, Flour.</i>	The search bar should accept the input data with the auto-completion function and the browser should move to the recipe results page for that query showing relevant recipes.
1.3	Test that the search function will run with no ingredients by clicking the search button while the box is empty.	Extreme data. Data - <i>nothing</i>	Clicking the search button should make the browser redirect the user to the recipe results page where no recipes are shown.

Test Id	Description and Process	Test type and Data	Expected Outcome
2.1	Test that users are able to login to their accounts.	Valid data. <i>valid username, valid password</i>	The account can be accessed and user is directed to the home page to begin search. Additionally, user can now view his profile from the menubar.
2.2	Users can like or dislike recipes, or cancel their vote.	Valid data.	Recipe score is updated accordingly and if the user likes the recipe, it should now appear on his profile page and generate new recommendations. Users can only rate on recipe if he is logged into his account.
2.3	Test if users can like the same recipe twice.	Invalid data.	The recipe can only be liked once, provided the user is logged into his account.
2.4	Test if users can submit a recipe.	Valid data.	The recipe now appears in the database.
2.5	Test whether users can edit their profiles.	Valid data.	New profile page should be updated accordingly.
Test Id	Description and Process	Test type and Data	Expected Outcome
3.1	Test that new recipes are loaded into the slideshow with each page refresh.	Valid data.	Not only new recipes must be loaded but their images must allow users to link to the recipe detail page.
3.2	Test that all links are valid	Valid data.	All links work as per normal.

All the tests were passed. Although an exhaustive testing was not possible, due to the large scope of possibilities, we did test our product on over 25 users, who thoroughly examined our website and were unable to find any functionality faults with it.

11.2.2 Website Compatibility

Before we proceed with testing our product on different browsers, we need to determine the relative popularity of the different browsers.

Web Browsers by market share	
1 Firefox	46.5%
2 Internet Explorer 8	14.7%
3 Chrome 4	11.6%
4. Internet Explorer 7	11.0%
5 Internet Explorer 6	9.6%
6 Safari	3.8%
7 Opera	2.1%

Broadly, we would assign a priority as to which browser the website should be tested on most thoroughly. Firefox, Internet Explorer 6, 7 and 8 and Chrome are assigned the highest priority. Our website successfully passed the tests on the aforementioned browsers, also, including Safari. Moreover, the test was carried out on different screen widths, and passed that as well.

11.2.3 Recipe Search Speed

Testing of the search speed is a convoluted task, because if we are to arrive at an exact big-oh complexity of the search, this would involve dealing with the Tanimoto coefficient and with tf-idf (term frequency-inverse document frequency) weighting which is used in text mining to compare the ingredients with tags. Rather than dealing directly with such nuanced topics, we can, however suggest an indirect big oh derivation by evaluating how the search works. First, all the tags for each object are collected and put into a matrix, by using a *for* loop and this is done for all objects (recipes). Then, a *for* loop is run on each tag for a particular recipe, which compares the tag with the specified ingredient based on some Tanimoto computation which returns a value. Then, if this value is sufficiently high, it is appended (along with the related recipe) to an array, which contains all the recipes which match the input ingredients. The value in the array with the highest value appears higher up in the search. Hence, this gives a speed described by the following Big-Oh notation: $O(\text{number of tags} * \text{tanimoto value for each recipe} * \text{appending to list} * \text{sorting the array})$. We could have tried evaluating the efficiency of our search experimentally but it would have been of little use due to variations in database caching and other server performance variations. If observational evaluation is used to assess the search speed, the speed is indistinguishable from the search speed of similar commercial websites, given our database which contains over 900 recipes.

12 Reflections on project

12.1 Technical Issues

The project required that members familiarise themselves with new forms of technology, for example Django (a python web framework). It was also necessary to regain familiarity with scripting and web-styling languages such as JavaScript and CSS. Inevitably, most of the time invested was spent resolving conflicts, a natural part of the learning process. Web tutorials and books were the main sources of learning.

Members were encouraged to reuse tested pieces of software whenever possible, in accordance with our design principles. This ensured that we minimised the incidence of errors in our code. However, it was still necessary for members to understand the underlying structure of the code which was reused.

For example, integrating multiple JavaScript applications in the web page resulted in issues such as dealing with Mootools library version conflicts, conflicts between different libraries, and "On-load" function conflicts, to name a few. It was necessary for designers to truly understand the workings of the code.

Additionally, there were some Django issues which arose, like improving the search efficiency for recipes. Even setting up a web server was a convoluted task, where members had to deal with bureaucracy and university red-tape. It was decided instead, that a local server be used.

Regardless, the team solved such issues with great panache, requiring minimal guidance and relying on each others support to collectively prevail as a team.

12.2 Time Management Issues

Beneath the disciplined facade suggested by the groups milestone charts, it was, admittedly, a herculean task negotiating the different deadlines. The main obstacle involved dealing with a variety of university assignments which were doled out on members in an unpredictable manner. However, the solution presented itself in due course- instead of tweaking the milestone chart, some members carried a heavier burden of the project when another members coursework was due, after which these roles were exchanged, to allow the first member time to complete his assignment. It was serendipitous that our members had enrolled in different modules which made this an effective solution.

12.3 Group Working Issues

The team has a clear management structure which has prevented any serious issues. Moreover, this enables members to take responsibility of their areas, thereby ensuring accountability and alleviating conflicts. This would have, ideally, also ensured an equitable distribution of workload amongst members as each sub-leader would be responsible for his/her area of the project. Weekly meetings were organised by the project manager and each week, the progress of the group was tracked. Pair work was also encouraged, in accordance with the principles of extreme programming. This again ensured that members became more familiar with each others presence and worked well together.

For example, the design sub-team frequently met up for interface design discussions. The software team also frequently made use of VOIP (voice over internet protocol) and instant messaging internet technology like Skype to work with each other.

12.4 Success of Project

One of the two main aims of the project was to meet the specifications laid out by the user at the beginning of development. These specifications were met quite early on, and we went on to surpass the original specifications, expanding our own version of the specification to include extra, more advanced features. By this measure the project is deemed to be a huge success.

The second main aim of the project was to learn group programming methodologies, and find a way to work together cohesively as a unit, learning to be more effective team developers. This aim was met well, as our clear management structure, consistent weekly meetings, and good use of group collaboration tools like subversion and basecamp shows.

13 Conclusion

The success of version 1 of our prototype within the set deadlines is a testament to the group's dedication to the project. This achievement also provides valuable affirmation to having set a reasonable timeplan and project specification.

Version 2 was met before the deadline, and there was time to implement a few extra features from the "Ideal System" Version 3 specification. Although it was a very challenging specification, the group had ambitious plans for further developments that could be made, and the structure of the project is designed to be extensible to allow such future features to be implemented. The project already surpasses many commercial services in its more advanced features, and with more development time we believe that it could lead the field.

14 Evaluation

At the basic level our project has not only met, but far surpassed the criteria we set for it to achieve. However, we need to take a broader based approach at our evaluation of our projects success by comparing it with similar commercial products which are currently on the market.

Our target was for the project to have all features from the realistic best version (v2) and a few from the ideal version (v3), which it easily met. With a sizable database composed of over 900 recipes, our product rivals many commercial products of similar scope.

Moreover, our product offers auto-complete based search of ingredients, which gives users an early affirmation when they attempt to search for a recipe based on their ingredients. This is unlike most similar products on the market, like BBC recipes and RecipeZaar.com. Supercook.com does offer auto-complete of ingredients, however, our website does it a step better because we allow the ingredient to form a token, which is visually more appealing than the one used on Supercook.com. (refer to diagram).



Figure 17: The auto complete search field for Supercook.com

Additionally, our project avoids clutter, which most websites of similar scope are plagued with, for example recipeZaar.com. With a users short attention span, purging superfluous information from our website prevents distracting users when they use our services.

Our index page features a slideshow which, upon refreshing the page, generates a completely new list of featured recipes, which you can quickly browse through pictorially, which encourages more ambitious users to create recipes which they would normally have never chanced upon. Serendipity is an important part of our website. As another example, by setting up an account with us, users are recommended recipes based on collaborative filtering technology, which provides users with seemingly coincidental recipes which they are likely to enjoy. This feature rivals many commercial websites, some of which, like BBC Recipes, do not even provide this functionality.

We have also used taken human behaviour into account when designing the website, decorating our interface with appetite stimulating colours like orange. Moreover, our banner spells Digichef using fruits and vegetables, alongside our mascot, which is a chef-robot. The central idea behind this was that if we are successful in stimulating the users appetite while at the same time, marketing our mascot , we will be able to

build a customer base who are more likely to repeatedly visit our website. No other website features a mascot, which in our opinion, presents an untapped resource to use for marketing purposes.²⁴

In addition, we allow users to set up an account with us, where users can upload their own recipes. There might be a potential problem here if users are allowed to submit any recipe of substandard quality. To solve this, we implemented a voting feature, where users can express their opinion on a recipe. Allowed more time, our group could have implemented more social networking features like chat to provide an involved user experience.

The search returns a list of recipes; each recipe contains a list of ingredients, instructions, and the ability to vote the recipe and also includes recipes similar to the recipe selected. Every recipe includes an image; for testing purposes we collected images from Google Image Search using a custom script and stored these as URLs within the database. Although this works well for the current version, if we had more time and a larger design team we would have been able to create a database of images for the recipes.

The banner uses different ingredients to spell DigiChef; this design emphasizes the purpose of the website. The colour scheme used is white, yellow and orange. The white background leverages on its simplicity and clarity, this is an important criteria for retaining the attention of the user. More over, the orange colour is reportedly simulating for the users appetite.

We have also implemented user accounts using a module called auth that handles users and their accounts. Users create a user account specifying their username, password, an about them section, their gender, email account, and their first and last name. This function has been implemented using a module called registration. The profile page includes the about them section, recipes that the user has liked and recommendations. Having a user account allows the user to rate recipes that they like or hate, these ratings are taken into consideration when the user searches for a recipe and returns only recipes that the user is most likely going to like. These functionalities introduce the use of Collaborative Filtering, this being something that not all of the recipe websites mentioned in our research include.

²⁴[Link to research, 31st March 2010 <http://desktoppub.about.com/cs/colorselection/p/orange.htm>]

15 Appendices

15.1 Timescale

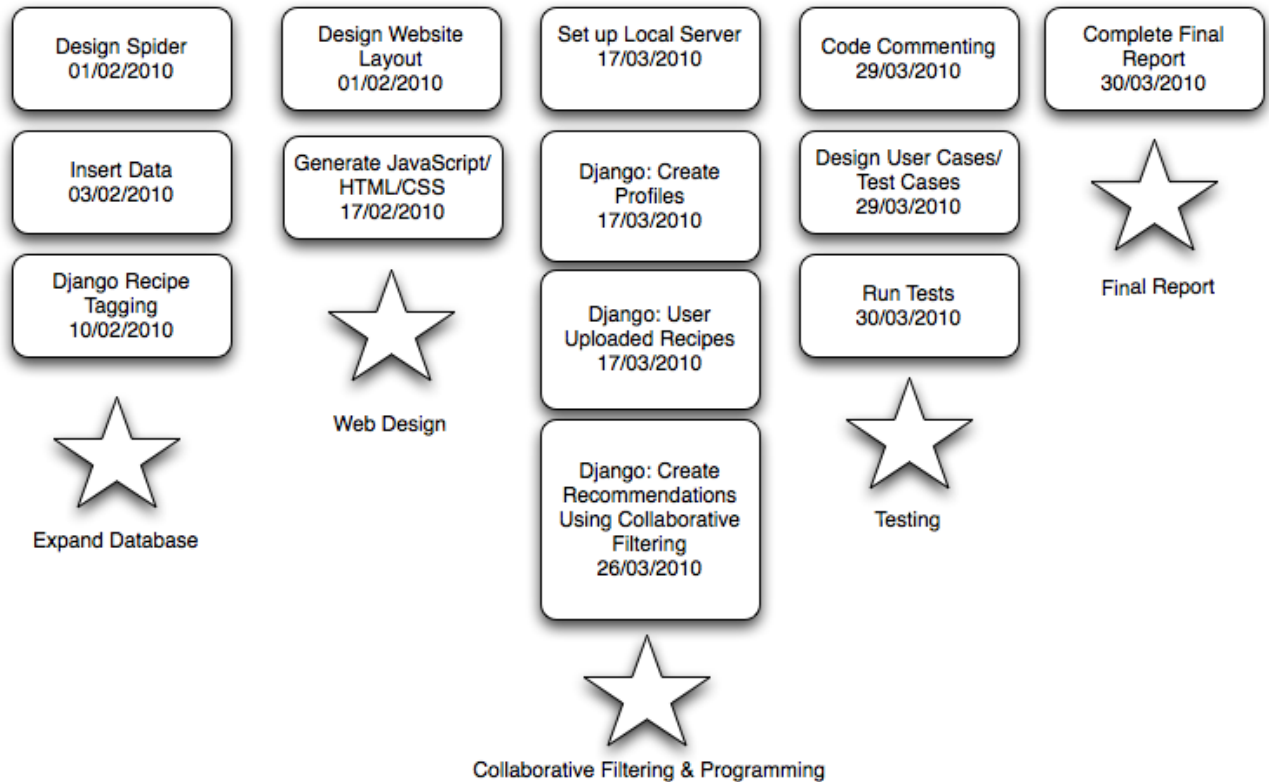


Figure 18: Project Timescale

The project timeline can be decomposed to 4 key deadlines:

1. Interim report submission (04/12/2009)
2. Final report due (01/04/2010)
3. Open day (05/05/2010)
4. Presentation day (07/05/2010)

Since our group is adopting the methodology of prototyping, a milestone chart is to be made for each of the three versions which conform to the above deadlines. Above is the detailed timeline of the version 2/3 prototype expressed as a project milestone chart (Fig 18). Each oval represents a downward progression to the milestones, represented by stars. The dates represent the deadlines that the group deemed appropriate for each progression. For this version the group focused on designing a clean, attractive and user-friendly web interface taking into consideration simple design principles. This version also introduces the use of JavaScript and Collaborative Filtering. Thereafter, tasks will

be allocated to each member where the individual heads of the project will take charge of their respective domains. Deadlines of specific tasks are as elucidated by the above. Notice that version 2/3 is to be completed before the deadline for the final report which exemplifies the fact that our chart takes into account the major deadline dates.

15.2 Meeting Minutes

Minutes 14.09.2009

The meeting took place at 1:30pm on the above date. The meeting location was in the computer science atrium.

Present

- Julie Greensmith (**Supervisor**)
- Amy (**Chair**)
- Dhruv (**Minutes Taker**)
- Jue
- Rob
- Chris

Tasks completed

1. Elect group leader
2. System Specifications- Done, consented by Dr Greensmith
3. Collaborate using Base Camp.
4. Dr Greensmith joined the Base Camp network.

Issues discussed

- Django doesnt seem like cheating- its just reusing what is available. Plus, clients just want to see the functioning product. Focus isnt on how youve made it.
- Ideally, develop a Facebook application along with the website. Add to ideal column in specification list.
- How do you get a massive userbase?
- Dr Greensmith endorses the realistic level of the specification as a good target.
- Ideal level of specification will be a great achievement, results can possibly be published.
- Binary matching of ingredients to recipes of specific type? There exists technology to support such a venture.
- Input from machine learning + user feedback, as a combination.
- Crowdsourcing technology.

- Quality of interface separates the minimum specification level from the realistic level.
- Ideal list– Has a full user system with profiles and user-uploaded recipes seems hard to do.
- How to measure extent of collaboration? Comparing the size of the userbase?
- Collaborative filtering?
- Ideal list– Gives recipe recommendations for several users i.e. A recipe that alice and bob will both like. Hard to do but its an awesome goal!
- Gantt charts are not as useful as Milestone markers.
- Identify end points and how to get there.
- The more you document, the easier to write up later.
- Keep a progress log.
- Design decisions need to be justified.
- Seems like everything needs to be justified.
- Set up blog?
- Focus on good planning.
- Have a backup plan.

To do

1. Milestone markers
2. Ideas for what the prototype is going to look like.
3. Assigning leadership roles for specific areas.
4. Decide on using Django?

Minutes 07.10.2009

The meeting took place at 1:30pm on the above date. The meeting location was in the computer science atrium.

Present

- Julie Greensmith (**Supervisor**)
- Dhruv (**Chair**)
- Amy (**Minute Taker**)
- Jue
- Rob
- Chris

Issues discussed

- To gain trust as a group and communicate.
- Relationship with Julie Greensmith is client based.
- Julie is passionate about cooking and finds that the BBC website is very limited as it only allows you to input 3 ingredients.
- Perhaps make the software portable.
- Focus on functionality, but ensure you have a good GUI.
- By Christmas we should be delivering a prototype, which is basic (build onto this).
- Make the software not just a database website. Include collaborative filtering - users, profiles, preferences, community based approach. This will give the software the WOW factor!
- Aims have to be achievable.

To do

1. Complete a specification before the next formal meeting.

Minutes 09.10.2009

The meeting took place at 1:30pm on the above date. The meeting location was in the computer science atrium.

Present

- Julie Greensmith (**Supervisor**)
- Amy (**Chair**)
- Jue (**Minute Taker**)
- Rob
- Dhruv

Tasks Completed

1. Milestone chart is completed for version one.
2. Job roles have been assigned.

Issues discussed

- There was a slight misunderstanding of how to present the milestone chart. After discussing this with Dr Greensmith we now understand that as long as we understand the milestone chart thats all that really matters.
- Since we are using Django to build the database all validation is included within the design.
- Ensure we have set deadlines for all tasks.
- Management heads for each section of the project. This includes documentation, quality assurance, technical, design and management.

To do

1. Begin to think about design, either the website or the database.
2. Begin to look at Django.

Minutes 21.10.2009

The meeting took place at 12:30pm on the above date. The meeting location was in the computer science atrium.

Present

- Amy (**Chair**)
- Rob (**Minute Taker**)
- Dhruv
- Jue

Apologies

- Chris is unable to attend today's meeting due to training.

Issues discussed

- Amy suggested we work on milestones and presented a template.
- Brainstorming of milestones headers, database, web interface etc...
- Discussion of possibilities for a server. Can we run Django on the school's servers? Would we be able to get an old machine and use that as our server? Or the possibility of funding for 3rd party hosting?
- Amy suggested a Django meeting on Tuesday - Time to be confirmed.

To do

1. Rob to post good websites to learn Django on basecamp.
2. All group members to read up on Django.

Minutes 11.11.2009

The meeting took place at 1:30pm on the above date. The meeting location was in the computer science atrium.

Present

- Julie Greensmith (**Supervisor**)
- Dhruv (**Chair**)
- Amy (**Minute Taker**)
- Jue
- Rob
- Chris

Tasks completed

1. Archived minutes from all previous meetings.
2. Plan of the website.
3. Group members have gone through the Django tutorial.

Issues discussed

- For version 1 we have agreed on a website design in order to begin web development next week.
- Group members have gone through the Django tutorial, however some group members thought this very time consuming.

To do

1. To get implementation ideas together in order to begin web development.
2. To work on the Django framework.

Minutes 18.11.2009

The meeting took place at 1:30pm on the above date. The meeting location was in the computer science atrium.

Present

- Julie Greensmith (**Supervisor**)
- Dhruv (**Chair**)
- Amy (**Minute Taker**)
- Jue
- Rob
- Chris

Tasks completed

1. Work on the Django framework has now been completed.
2. The choice of languages to be used for the website have also been confirmed.

Issues discussed

- The website will be constructed using HTML and CSS, this is for version 1. Ideas to use JavaScript and Ajax for later versions have been discussed.
- The deadline for the Interim report is within the next couple of weeks, we need to allocate a section for each group member to complete.

To do

1. Group members need to complete their respective part of the report.

Minutes 27.01.2010

The meeting took place at 1:30pm on the above date. The meeting location was in computer science atrium.

Present

- Dhruv (**Minute Taker**)
- Amy
- Rob
- Jue

Apologies

- Chris is not able to attend the meeting.

Issues discussed

- Informal meetings to be had on Friday.
- Time to be decided later.
- This Friday's time will hopefully be at 1pm.
- Agreed on milestone chart.
- Database expansion complete.

To do

1. Amy and Jue – Have web page design ready by Friday.
2. Rob and Dhruv – Expand django knowledge.
3. Chris – Come for next meeting
4. Setup server coming Friday
5. Decide on design on Friday and start coding

Minutes 29.01.2010

The meeting took place at 1:30pm on the above date. The meeting location was in the computer science atrium.

Present

- Dhruv (**Chair**)
- Jue (**Minute Taker**)
- Amy
- Rob

Apologies

- Chris is unable to attend today's meeting.

Issues discussed

- The website needs to be expanded with new function, such as the suggestion of related recipe and the registration of users' own account.
- Some basic ideas of the functionality of the website, for instance, users can rate a recipe as well as make their comment on it once they reach a recipe page.
- Run a server to support the website.

To do

1. Rob and Dhruv set up a server.
2. Amy and Jue draw some sketch of the website.

Minutes 03.02.2010

The meeting took place at 1:30pm on the above date. The meeting location was in supervisor Julie's office, c11 of the computer science building.

Present

- Julie Greensmith (**Supervisor**)
- Dhruv (**Chair**)
- Jue (**Minute Taker**)
- Amy
- Rob

Apologies

- Chris is unable to attend today's meeting.
- Discuss about the feedback of the interim report.

n

Issues discussed

- Further report needs more proof reading, since there are lots of silly mistakes in the interim report, meaningless brackets and type errors.
- Some topics are mentioned in the report but without any brief explanation.

To do

1. Rob and Dhruv set up a server.
2. Amy and Jue draw some sketch of the website.

Minutes 12.02.2010

The meeting took place at 1:30pm on the above date. The meeting location was in the computer science atrium.

Present

- Dhruv (**Chair**)
- Jue (**Minute Taker**)
- Amy
- Rob
- Chris

Tasks completed

1. Plan of the website.
2. Rob has finished the background jobs of the database.

Issues discussed

- For version 2 we have agreed on a website design and function in order to begin web development next week.
- Server needs to be accepted by the school office.

To do

1. Amy and Jue – work out the brief CSS structure of the website.
2. Rob – Solve the server problem.

Minutes 24.02.2010

The meeting took place at 1:30pm on the above date. The meeting location was in the computer science atrium.

Present

- Dhruv (**Chair**)
- Jue (**Minute Taker**)
- Amy
- Rob

Apologies

- This week is mainly concentrate on the CMP coursework

Issues discussed

- Our web server is still not confirmed by the school office, so we decide to discuss it later.

To do

1. Jue – Do the bibtex reference.
2. Amy – Do the basic structure of the website.
3. Rob – Do the background search function.

Minutes 03.03.2010

The meeting took place at 1:30pm on the above date. The meeting location was in the computer science atrium.

Present

- Julie Greensmith(**Supervisor**)
- Dhruv (**Chair**)
- Jue (**Minute Taker**)
- Amy
- Rob
- Chris

Tasks completed

1. The index of the website if finished but it not quite professional.

Issues discussed

- Julie the supervisor taught us about what a normal professional website would be like.
- The index page of the website need to be modified to be more like official website.
- A slideshow is decided to be made.

To do

1. Amy – Focus on the new css of the website.
2. Dhruv – Do the java script slideshow.
3. Jue – Do the result page.

Minutes 10.03.2010

The meeting took place at 1:30pm on the above date. The meeting location was in the computer science atrium.

Present

- Julie Greensmith(**Supervisor**)
- Dhruv (**Chair**)
- Jue (**Minute Taker**)
- Amy
- Rob
- Chris

Tasks completed

1. A functional index page is finished, including the function of search for multiple ingredients creating user own profile.
2. A stylish java script slideshow has been modified.

Issues discussed

- The slideshow needs to be added to the index page.
- A auto complete search box is decided to be used.
- There is a problem that once the browser window is resized, all parts of the website move around instead of stay in their own position. This needs to be solved.

To do

1. Dhruv – add the java script slideshow to the website.
2. Rob – Improve the website background function.
3. Chris – Design a questionnaire for the user.

Minutes 17.03.2010

The meeting took place at 1:30pm on the above date. The meeting location was in the computer science atrium.

Present

- Dhruv (**Chair**)
- Jue (**Minute Taker**)
- Amy
- Rob
- Chris

Tasks completed

1. A working java slideshow is added to the index of the website.

Issues discussed

- Different parts of the report is arranged to each person.
- Jue: Updated design of the system and interface and modify the minutes
- Amy: Implementation - XHTML & CSS and what's been achieved in comparison to the specification.
- Chris: Testting.
- Dhruv: JavaScript & Collaborative Filtering and Reflective comments.
- Rob: Django.

To do

1. Dhruv – solve the website problem.
2. Every body starts on the report.

Minutes 24.03.2010

The meeting took place at 1:30pm on the above date. The meeting location was in the computer science atrium.

Present

- Dhruv (**Chair**)
- Jue (**Minute Taker**)
- Amy
- Rob
- Chris

Tasks completed

1. The problem that every parts move around once the browser is resized is solved.
2. Rob adds a function to the background, therefore we can have image for each recipe we have in the database.
3. The javaScripts slideshow is work proper now.

Issues discussed

- The website still need the result page and profile page.

To do

1. More style work on the website.
2. Needs to move on the report.