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”

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 search[1] (Fig 1), but is not constrained to just three search items. The software should, in essence, provide recipe matches to help the user make recipe decisions given the ingredients he/she currently has. This involves making recommendations and provides us with an avenue to make use of collaborative filtering technology. 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 and possibly AJAX but will also have a separate simpler web interface for mobile devices (with just the use of HTML and Django template markup). 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

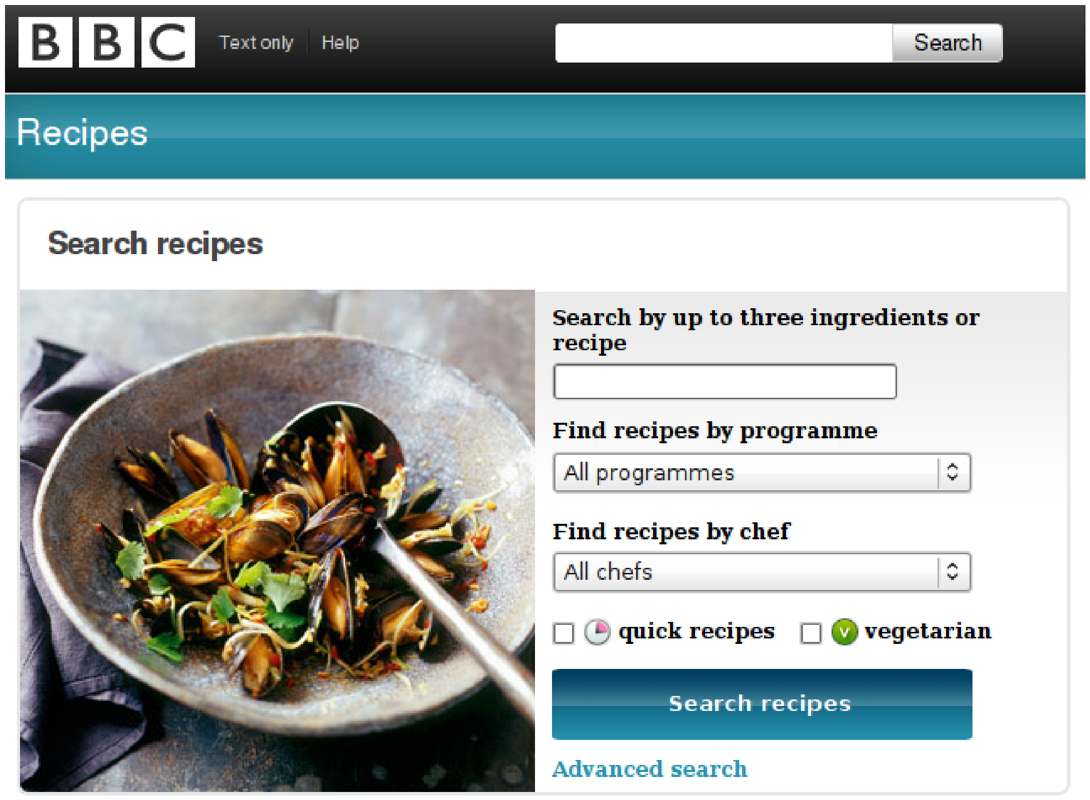


Figure 1: The BBC Food Recipe Search Page

BBC Recipes[2] (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.

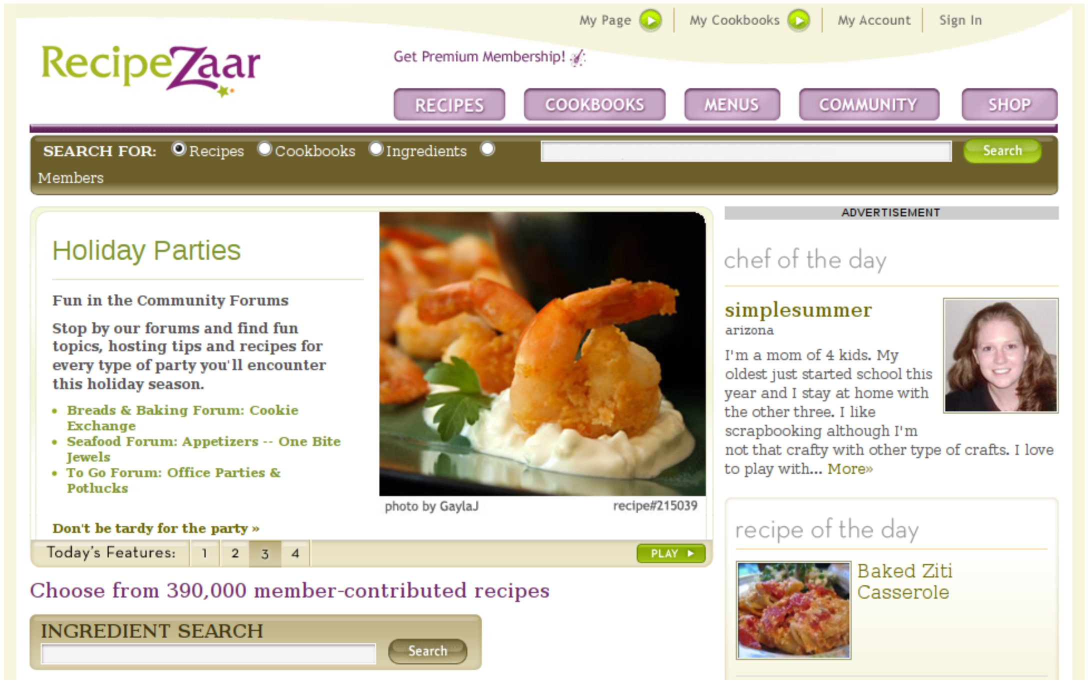
One main attraction about this application is the advanced search option which includes a numerous amount of tag options. This option becomes quite useful when searching through a large database as this kind of search minimizes the results. A good example of this is a search including flour, butter and 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 igredient, 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



Holiday Parties

Fun in the Community Forums

Stop by our forums and find fun topics, hosting tips and recipes for every type of party you'll encounter this holiday season.

- Breads & Baking Forum: Cookie Exchange
- Seafood Forum: Appetizers -- One Bite Jewels
- To Go Forum: Office Parties & Potlucks

Don't be tardy for the party »

photo by GaylaJ recipe#215039

chef of the day

simplesummer
arizona

I'm a mom of 4 kids. My oldest just started school this year and I stay at home with the other three. I like scrapbooking although I'm not that crafty with other type of crafts. I love to play with... More»

recipe of the day

Baked Ziti Casserole

Choose from 390,000 member-contributed recipes

Figure 2: The RecipeZaar.com Home Page

Recipezaar[3] (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

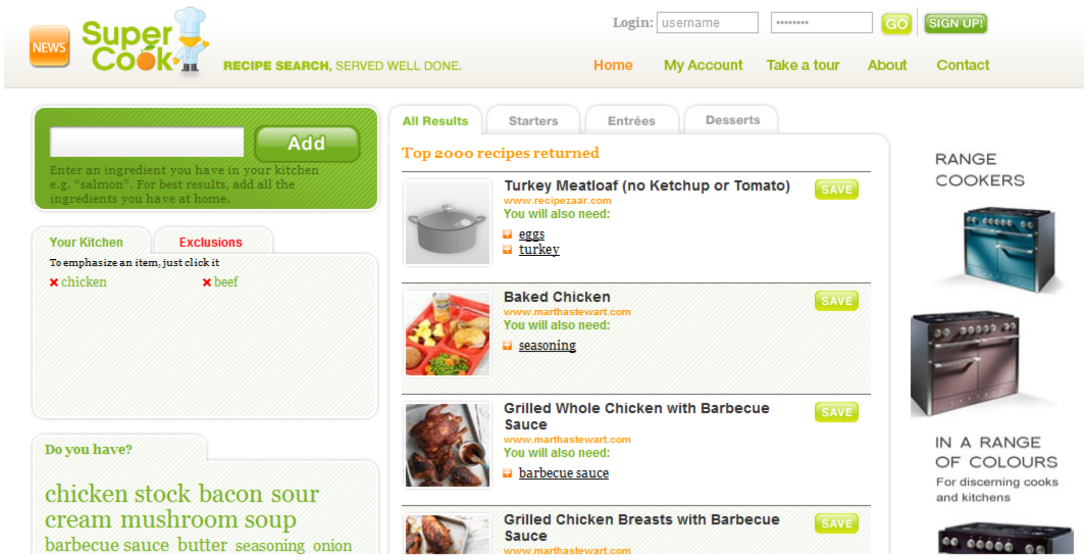
[3][RecipeZaar, 24th November 2009 `https://RecipeZaar.com`]

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



Enter an ingredient you have in your kitchen e.g. "salmon". For best results, add all the ingredients you have at home.

Figure 3: The Supercook.com Home Page

Supercook[4] (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.

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 stabalise 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 XP[5]. 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 users[6].

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

[6][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 languages[7]. 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 esay 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.

[7][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 Opera[8]. Of these, the majority support the web standards codified by the World Wide Web Consortium (W3C)[9], 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 validator[10], which will save us having to debug the site in every browser we wish to support

[8]`http://www.w3schools.com/browsers/browsers_stats.asp`
[9]`http;//www.w3.org`
[10]`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 people [11]. 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 Parc[12]. 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

[11]Schafer J.B., Frankowski D., Herlocker J., Sen S.: Collaborative Filtering Recommender Sytems

[12]Goldberg 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.

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

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 accurae. 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.

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

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.

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

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 4.

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

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 us composed of a weighted sum of the users ratings for items most similiar to i, described by Equation 5, where *itemSim()* is a measure of item similarity.

$$pred(u,i) = \frac{\sum_{j \subset relatedItems(u)} itemSim(i,j) \ \cdot \ r_{ui}}{\sum_{j \subset relatedItems(u)} itemSim(i,j)} \tag{5}$$

Adjusted-cosine similarity is the most popular and accurate[13] 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-based[14].

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.

[13]Schafer J.B., Frankowski D., Herlocker J., Sen S.: Collaborative Filtering Recommender Sytems

[14]Sarwar 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 algorithm [15] 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.

[15] Hoffmann 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 analysis [16] 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 products [17].

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

[17][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 - version3 [18]). 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 large database of recipes.
- Has a clean, attractive web interface.
- Has a seperate, simpler web interface for mobile devices.
- Recipes are searchable by combinations of ingredients.
- Has user accounts.
- Allows users to rate recipes.
- Gives recipe recommendations based on past ratings.

[18] It 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 commment 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

The version 1 website colour scheme is an amalgamation of white (for the background) and orange (for miscellaneous design features). The white background leverages on its simplicity and clarity, an important criteria for retaining the attention of the user. Moreover, the orange colour is reportedly stimulating for the users appetite[19]. The group also agreed on an orange logo design.

7.1 Version 1

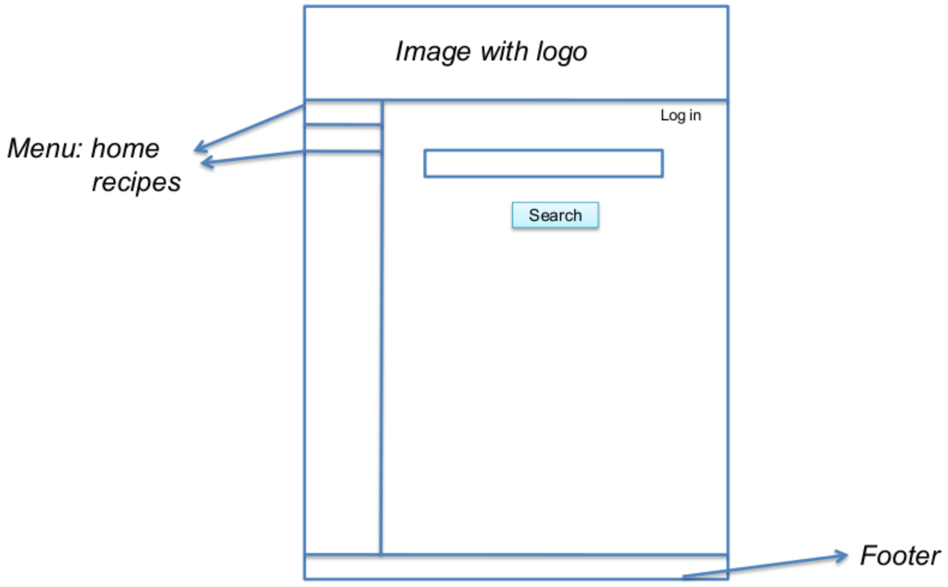


Figure 4: Layout of the Home Page

The merit of the version 1 website design is in its simplicity. Upon accessing the website, (Fig 4) users are presented there are three drop-down menus which allows people to select ingredients. The drop-down menus are transversely positioned to accommodate the vertical cascading of the ingredients of the menus. Users are provided with three menus to select ingredients and submit a search. This then links to the recipe list page. Additionally, the recipes button in the sidebar links the page to the complete list of recipes alphabetically.

[19][http://desktoppub.about.com/cs/colorselection/p/orange.htm]

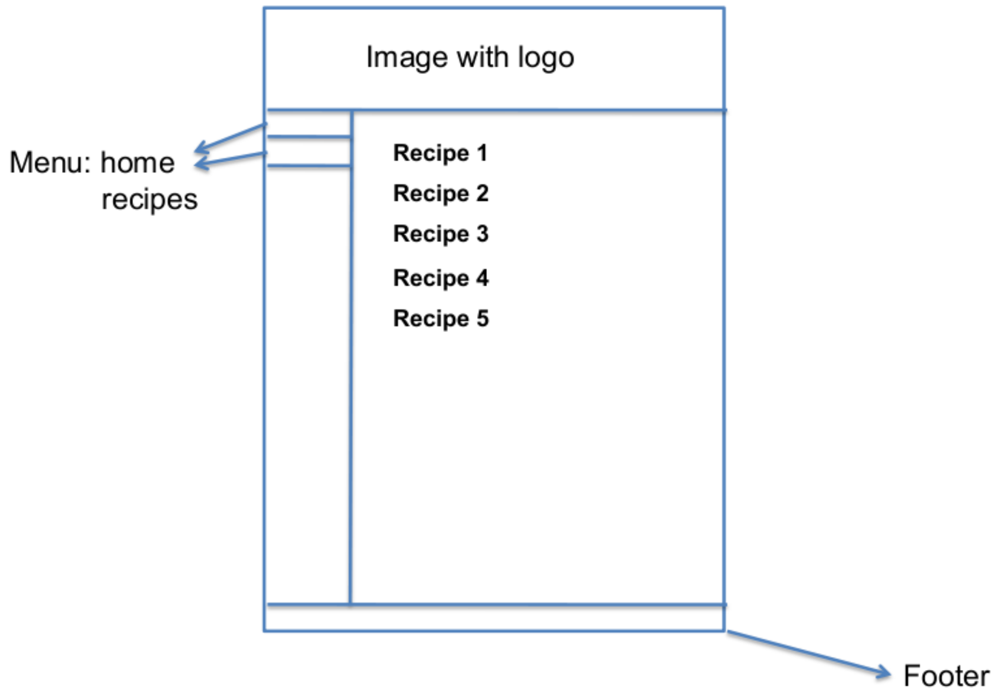


Figure 5: Layout of the Recipe List Page

The recipe list page (Fig 5) contains a list of recipes with at least one of the three ingredients. However, the recipe may contain other ingredients which were not specified by the user. Upon recipe selection, the specific recipe page will be displayed.

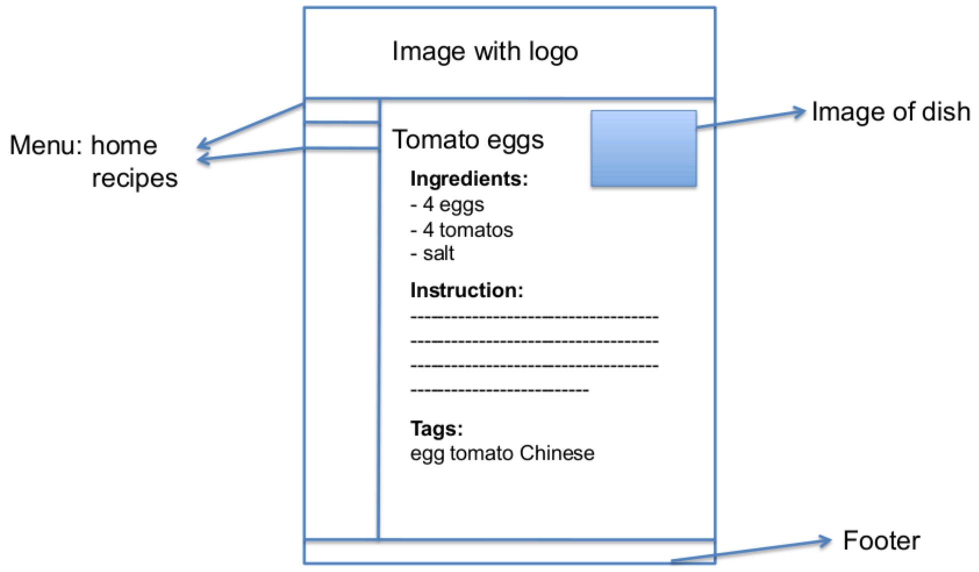


Figure 6: Layout of the Recipe Page

The recipe page (Fig 6) contains recipe details, for example recipe name, ingredients, the instruction, and recipe tags.

One design limitation of version 1 is that the website only has three drop-down menus, and the user is unable to type in ingredients. Being a prototype, the version 1 database is fitted with few recipes hence such functionality is redundant.

7.2 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 JAVA Script, the web interface will look more polished. Users will be able to enter text data into ingredient selection text boxes, which will have the tab completion feature for ingredients instead of using a drop-down menu. There will also be a larger database of ingredients hence justifying the use of tab completion as apposed to drop-down menus.

Additionally, for version 2, users are allowed to have web accounts. The benefits of the account include the user having access to past recipes which he/she rated and more importantly receive recommendations of recipes they might like (this uses collaborative filtering technology). In addition, users could choose to click tags and get a list of recipes that contains that particular tag.

Additional website link may likely be created, for example a link to the most popular recipes. The website is also expected to be optimised for mobile phone viewing.

7.3 Version 3

The ideal version of our product involves the implementation of a variety of possible functionalities. For example, a mobile application could be developed. Recipes could be made searchable by not just ingredients but also, for instance, the type of cuisine (Chinese dish) or whether recipes are vegetarian or non-vegetarian.

Social networking is another possibility, with users being able to interact with other users and leave comments of the pages of other users. 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 17) 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 philosophies[20] 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 chose their implementation decisions, they are designed to work best with implementations that follow their principles.

[20]`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 Documentation[21]

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.

[21] `http://docs.djangoproject.com/en/dev/topics/templates/`

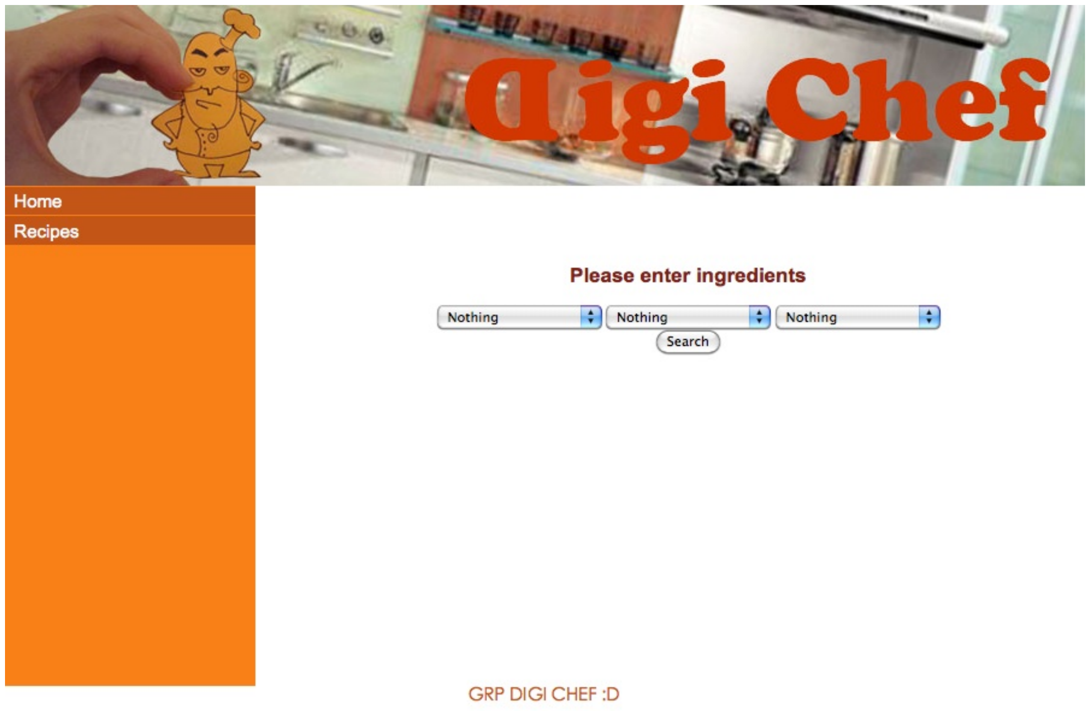


Figure 7: Homepage

10 Implementation Results

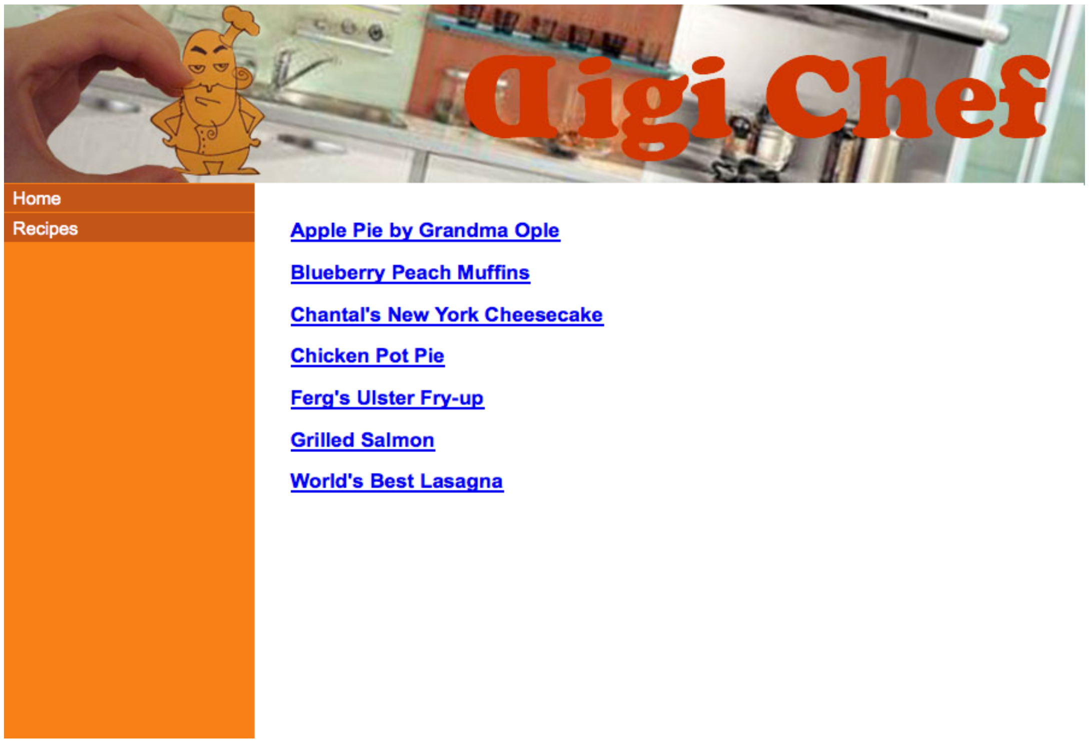
Version 1 largely adheres to the design specification.

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 7) shows the drop down menus for ingredient selection with the search option.

(Fig 8) shows the result of ingredient selection by the user. The recipe list was generated based on the ingredients selected by the user.

(Fig 9) 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.



Apple Pie by Grandma Ople

Blueberry Peach Muffins

Chantal's New York Cheesecake

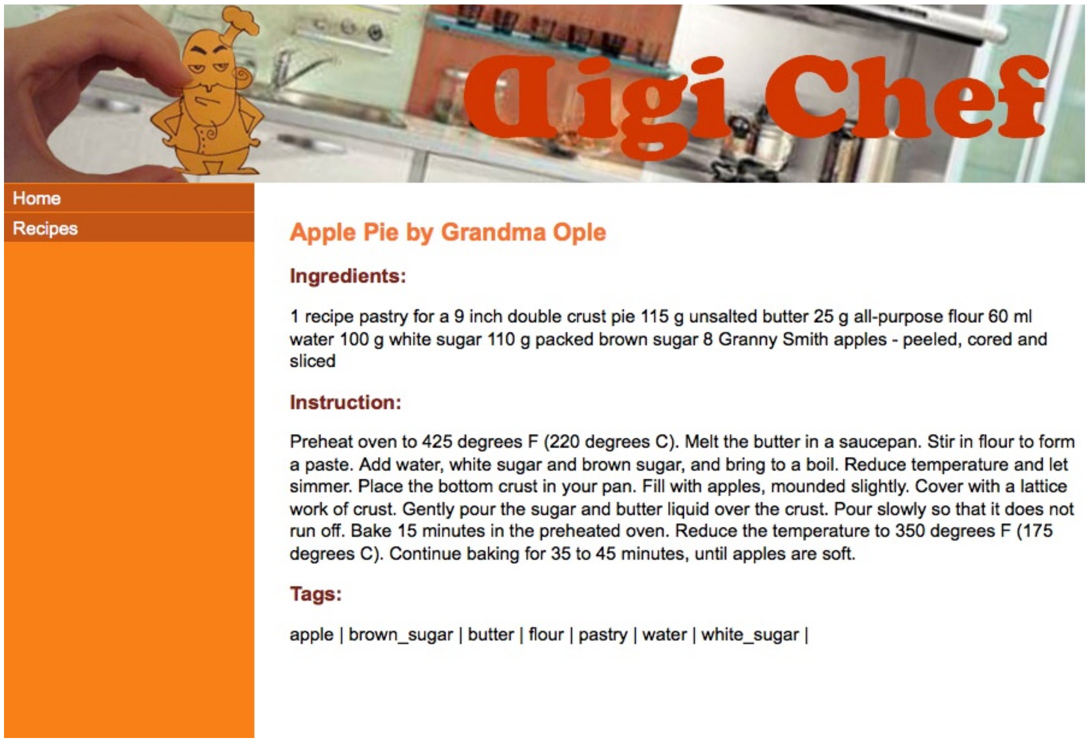
Chicken Pot Pie

Ferg's Ulster Fry-up

Grilled Salmon

World's Best Lasagna

Figure 8: Search results



Apple Pie by Grandma Ople

Ingredients:

1 recipe pastry for a 9 inch double crust pie 115 g unsalted butter 25 g all-purpose flour 60 ml water 100 g white sugar 110 g packed brown sugar 8 Granny Smith apples - peeled, cored and sliced

Instruction:

Preheat oven to 425 degrees F (220 degrees C). Melt the butter in a saucepan. Stir in flour to form a paste. Add water, white sugar and brown sugar, and bring to a boil. Reduce temperature and let simmer. Place the bottom crust in your pan. Fill with apples, mounded slightly. Cover with a lattice work of crust. Gently pour the sugar and butter liquid over the crust. Pour slowly so that it does not run off. Bake 15 minutes in the preheated oven. Reduce the temperature to 350 degrees F (175 degrees C). Continue baking for 35 to 45 minutes, until apples are soft.

Tags:

apple | brown_sugar | butter | flour | pastry | water | white_sugar |

Figure 9: Recipe page

11 Testing

11.1 Testing Data

Obtaining suitable testing data for the project was technical challenge of substantial proportions. This section deals with the requirements decisions we made and the technical solutions we used.

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 Solutions

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.

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 conflicts, multiple library conflicts, "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 effeciency 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 serendipituous 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 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 toghether. 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

The aim of the project was to achieve the ideal level version.

13 Conclusion

The success of version 1 of our protype 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. Having overcome the initial hurdle of acquiring a feel of the technologies involved, the group can look forward to expanding version 1 to versions 2 and 3.

14 Appendices

14.1 Timescale

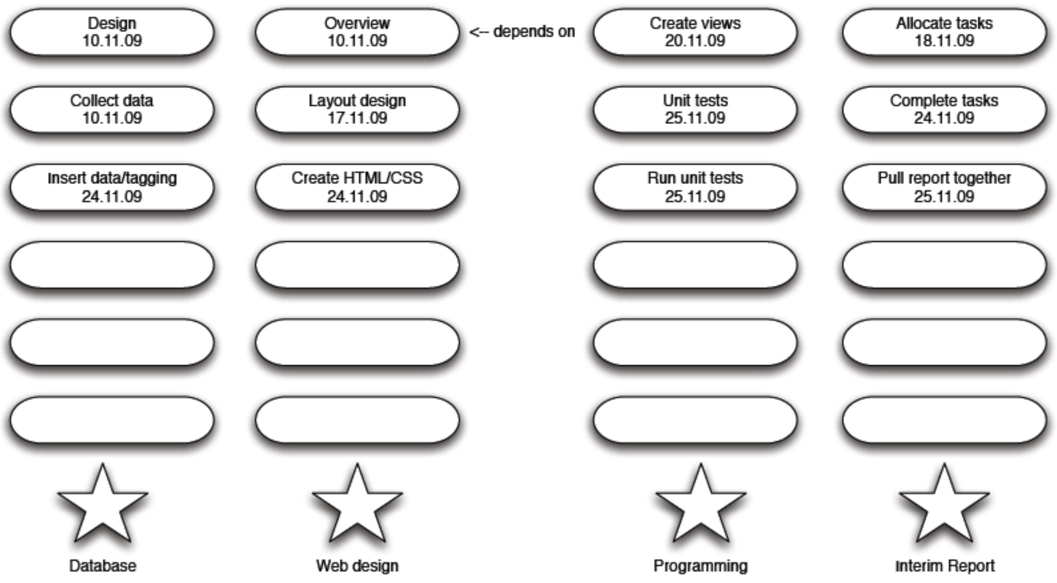


Figure 10: Project Timescale

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 1 prototype expressed as a project milestone chart (Fig 10). 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. The group decided early on to plan out the web and database design together, which is essentially the meat of our project. 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 1 is to be completed before the deadline for the interim report which exemplifies the fact that our chart takes into account the major deadline dates.

Due to the dynamic nature of our project, it was decided that further milestone charts will be designed as we progress from the completion of one version to another. Moreover, it is unwise to have planned out charts for future versions before even acquiring a feel of the project.

14.2 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 recipesseems hare 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 todays 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.