G52GRP Interim Group Report

gp09-jqg[*]

November 30, 2009

Contents

[*]Dhruv Gairola, Chris Head, Rob Miles, Amy Jane Wesson and Chenjue Xu, supervised by Julie Greensmith

1 An Introduction to the "Problem"

"You arrive home after a long day at work, open the fridge to find ... a single egg, one lonely onion and some almost mouldy cheese ..." 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], but is not constrained to just three search items. The software should provide a number of matching recipes and rank the suggestions according to how well they match. The recipe database can be online and community maintained and take into account the user's own food preferences. To achieve this collaborative filtering techniques can be used to provide and manage the recommendations. As an additional extra, lists of required items can be created and exported to a mobile phone as a shopping list. In addition there is the potential to link the software to an online supermarket such as tesco.com to calculate the current cost of each created meal.

[1][bbc.co.uk/food]

2 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 1.0, Realistic Best - version 2.0 and Ideal - version3.0 [2]). 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.

2.1 Minimum - v1.0

The requirements listed below are the minimum requirements to make the software work. This does not include extra functionality desired by our client. This will be regarded as version 1.0 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 hopefully allow us to keep continuity between languages and structures removing 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).

2.2 Realistic Best - v2.0

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 1.0 .

- 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.

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

2.3 Ideal - v3.0

Below are the requirements for the software if it were to be the ideal solution to the problem. This list includes existing requirements from v1.0 and v2.0 at a higher level and all desirable requirements as well. It is likely that only few or none of these requirements will be fulfilled in the final version that is submitted.

- 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'.

3 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.

3.1 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[3]. 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 [4].

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

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

3.2 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[5]. 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.

[5][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.

3.3 Server setup

Why web based nd not downloadable app. The implementation decisions made concerned the web scripting language and web framework. We considered a variety of scripting languages to work with. For the prototype website, we decided to make use of HTML. It provides an ideal outlet for group members to acquire a feel of the project by working very simply and speedily. Other more complex client side scripting languages were considered for the prototype, namely Javascript and VBScript.(cons of html before pros) The web framework provides us a way to reduce overhead associated with web development, for example

4 Implementation Options

4.1 Implementation Options/Designs:

Summary of Project Description and Specification

The description of the project to be completed is rather vague, however, it does state that the format of the solution is to be a database which can either be implemented 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 (which can also be seen in the Problem Specification). This can be implemented in many different styles/languages which we have also discussed extensively. We have created the Problem Specification 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 will make the solution easily upgradable for the members of our group or possibly external groups/people in the future. This therefore means that to keep to this method of creating the solution we need to find a language that has all the functionality to complete all of the requirements and also work easily with a database and with some sort of web programming language such as HTML .

Language Choices

Taking into account the requirements and also our experience as a group regarding programming languages, our Technical Officer suggested that we use the language Django as the backbone of the solution with a HTML webpage implementation for the GUI. We came to this outcome as the Django language is very compatible to the solution that we require. It has all the functionality that is required to implement all of the requirements and can implement the bulk of the solution (i.e. collaborative filtering) with ease. Django is also designed to work closely with a database and also contains many other features such as tagging which makes it an ideal language to use for the solution. We have decided to create the database in a simple MySQL style as we are all familiar with this implementation and also MySQL provides all the functionality required for the proposed solution. Django also works very closely with MySQL structured databases.

Operating Systems

If we are to use the proposed language (Django), operating system portability should not be an issue as Django is a multi-platform language, although the solution will initially be intended to be used on the Microsoft Windows platform. This is another advantage of using this language. We have decided to initially aim the solution to be developed for the Microsoft Windows platform as it currently has the highest market-share and is therefore the most popular Operating System. This will allow us to have a larger potential customer base if the solution were to be released.

Additional Hardware/Software

The solution is to be an online web-based application with a database backing. This being true, it will be necessary for a server to be used to host the webpage. We have been granted some server space for our application from an external Web Design Consultant who is an associate of the Technical Officer. There are no other software requirements for the software to run.

Browser Options

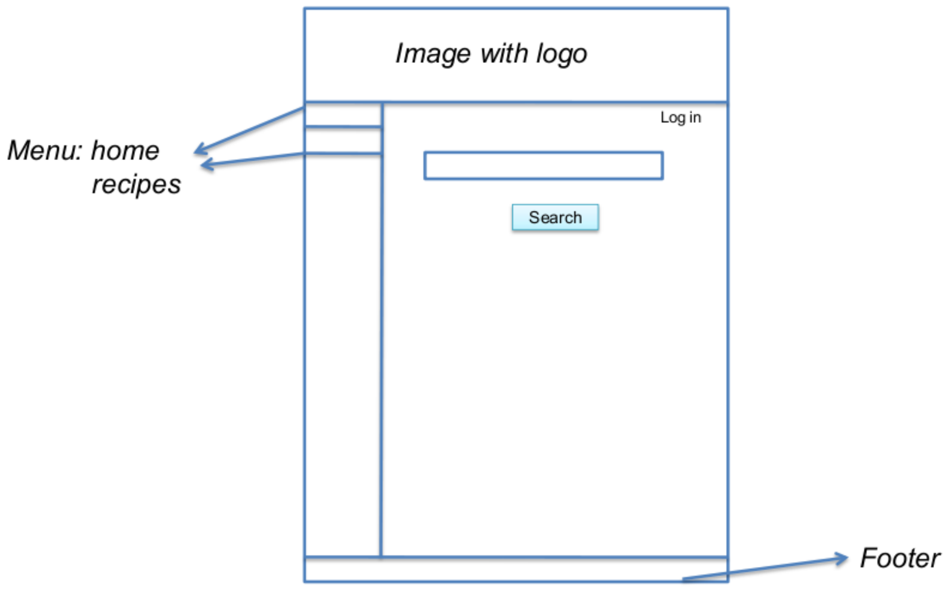

Figure 1: Layout of the Home Page

5 Initial design of the proposed system and its user interface

The basic color of the website is white and a serious orange. The main background of the website is white, since it makes the view clear and simple which would comfort the users. According to research, this warm color orange is stimulating, not only stimulating emotions but also appetite. Therefore, in order to make the website stand out and unique, the logo would be orange as well.

5.1 Version 1

Digichef is a with a web interface base on a web framework. Upon accessing the website, (Fig 1) users are presented there are three drop-down menus which allows people to select ingredients. The drop-down menus are transverse because the ingredients are related. After the user selected three ingredients and clicked on search, the page will be substituted to the page of recipe list. If the button recipes from the sidebar is clicked, moreover, the page will change to the whole list of all recipes ordering by alphabet.

The recipe list page (Fig 2) contains a list of recipes with at least one of the three ingredients. In each recipe, other ingredients would be included as well, nevertheless. Once one recipe is selected, the web will connect to the exact recipe page.

The recipe page (Fig 3) contains the details of the recipe, recipe name, ingredients, the instruction, and tags. The tags are mainly about the ingredients of the recipe and the region of the recipe. One of the basic limitation of version 1 is that the website only has three drop-down menu for the user, that is, the user could not type in ingredients.

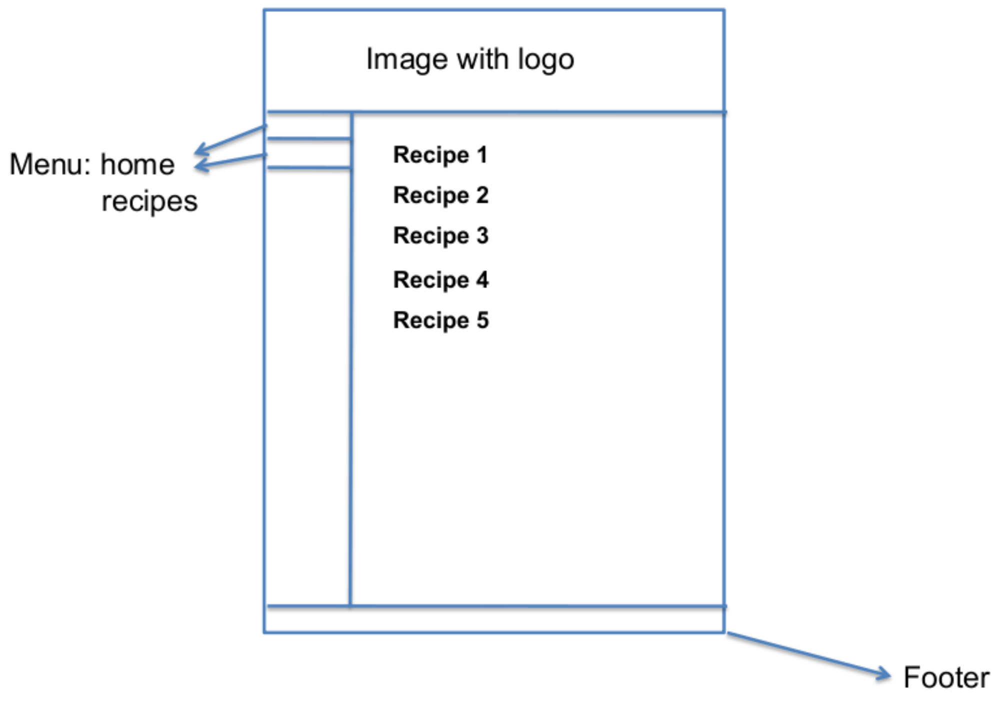

Figure 2: Layout of the Recipe List Page

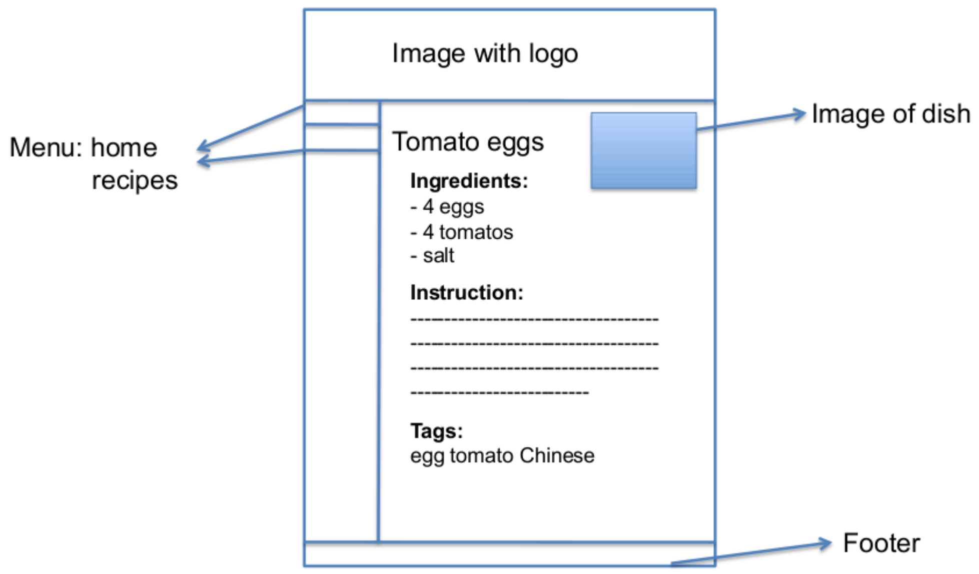

Figure 3: Layout of the Recipe Page

The reason we only have three drop-down menus at this point is that the version 1 is quite small, and we also only have limited number of recipes. Therefore, for version 2, the home page will have three text-fields instead. Furthermore, the web interface is also need to be more clear and attractive in the future.

5.2 Version 2

Version 2 is an upgraded version of the original version containing more functions (as shown in the Product Specification). With the use of JAVAScript, the web interface will be more professional and attractive. Users, moreover, could enter text data regarding their ingredients into a text box which uses tab completion 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 a drop-down menu. Additionally, for version 2, users are allowed to create their own account. With this account, a user can have their own history of recipes that they rate. Each recipe would be rated from 1 star to 5 stars. Additionally, the home page will contain not only the area for search of recipes but also the area made of brief look of the most popular recipes. And once the user has logged in, the home page will also have an area that gives suggest recipes based on past rating. One typical improvement for the home page is that, the search-text-field allows user to search by combination of ingredients. Thus, user could search recipes by typing multiple ingredients separated by commas or spaces. And the result of the recipe list is ordered according how matching the ingredients are. For the recipe page, with the image of the recipe, a new area called You might like would be also added. This area is also made up of a list of brief look of recipes. These recipes are chosen from the database using collaborative filtering techniques. In addition, users, at this point, could click the tags and get a list of recipes that contains this tag. The last but not the least, we will also have a simple web view for mobile phone and other portable equipment (iTouch, psp, etc). This mobile view is only simple black and white appearance with the same function as the version 1, which could be accessed through the internet.

5.3 Version 3

The ideal version, which is the third version of digichef, would not only have a more friendly website based on version 2 but also a mobile application that user could actually install. And the recipes are searchable by more than ingredients, for instance, the region of this recipe (Chinese dish) or other category like vegetarian. With the increasing of the whole database and tag system, the home page would also have the area contains the most popular tags. When the user click on a typical tag, the website would update with a list of recipes which ordered by rates. Once the user want to rate recipe, the webpage would open a feedback dialog box which allows user to rate it vary from one star to five stars, and also allows user to leave their own command about this recipes. And a number of the latest command would be showed on the exact recipe page. A huge improvement of version 3 is that user could upload their own recipes. When they construct the new recipe, they need to provide the name, the ingredients, the instruction and also the tag to classify the recipe. Thus, the new version of recipe page would be same as version 2 but with authors name[6].

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

6 Implementation Decisions

6.1 Decision Influences

6.1.1 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.

6.1.2 Design Principles

The project is being developed using a version of the Extreme Programming 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[7] 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.

[7] `http://docs.djangoproject.com/en/dev/misc/design-philosophies/`

6.2 Decisions

6.2.1 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.

6.2.2 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.

6.2.3 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.

6.2.4 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.

6.2.5 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[8]

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

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.