

**Recipes For Life:**

***Bringing the tradition of cookbooks to the 21st Century***

**Kari McMahon**

**AC40001 Honours Project**

**BSc (Hons) Applied Computing**

**University of Dundee 2015**

**Supervisor: Dr Keith Edwards**

**Table of Contents**

[1. Introduction 5](#_Toc417646404)

[2. Background 5](#_Toc417646405)

[2.1 Cookbooks as Historical Documents 5](#_Toc417646406)

[2.2 Traditional Cookbooks and Their Benefits 6](#_Toc417646407)

[2.3 Current Recipe Applications 6](#_Toc417646408)

[3. Specification 8](#_Toc417646409)

[3.1 What Is A Minimum Viable Product? 8](#_Toc417646410)

[3.2 Initial Specification 8](#_Toc417646411)

[3.3 Data Gathering 8](#_Toc417646412)

[3.4 Final Specification 9](#_Toc417646413)

[3.5 Specification Flexibility and Managing Requirements 9](#_Toc417646414)

[4. Project Management 10](#_Toc417646415)

[4.1 Methodology 10](#_Toc417646416)

[4.2 Project Management Tools 11](#_Toc417646417)

[4.2.1 Initial Project Plan, Gantt Charts & Trello 11](#_Toc417646418)

[4.2.2 Sprint Backlogs 11](#_Toc417646419)

[4.2.3 Risk Assessment 11](#_Toc417646420)

[4.2.4 Supervisor Meetings & Minutes 12](#_Toc417646421)

[4.2.5 Log Book & Github 12](#_Toc417646422)

[4.3 Realities of the Methodology 12](#_Toc417646423)

[5. Design 13](#_Toc417646424)

[5.1 Application Operating System Selection 13](#_Toc417646425)

[5.2 Server Side Design 14](#_Toc417646426)

[5.2.1 Database Requirements & Selection 14](#_Toc417646427)

[5.2.2 Database Design 14](#_Toc417646428)

[5.2.3 Server Side Development Language Selection 14](#_Toc417646429)

[5.2.4 Server Side Development Tools 15](#_Toc417646430)

[5.3 Application Design 15](#_Toc417646431)

[5.3.1 Application Layout & Design Patterns 15](#_Toc417646432)

[5.3.2 Application Language Selection 16](#_Toc417646433)

[5.3.3 Application Development Tools 16](#_Toc417646434)

[5.4 Application & Server Communication 16](#_Toc417646435)

[5.5 Collaboration Feature 16](#_Toc417646436)

[5.6 Paper Prototypes 17](#_Toc417646437)

[5.7 Design Sketches 17](#_Toc417646438)

[5.8 Ethics 17](#_Toc417646439)

[5.9 Intended Audience 18](#_Toc417646440)

[5.10 Design Focus Group 18](#_Toc417646441)

[6. Implementation & Testing 18](#_Toc417646442)

[6.1 Changes from the Design Stage 18](#_Toc417646443)

[6.2 Interface Design 18](#_Toc417646444)

[6.2.1 Customized Dialogs 18](#_Toc417646445)

[6.2.2 Custom Listview 19](#_Toc417646446)

[6.2.3 Action Bar 19](#_Toc417646447)

[6.2.4 Challenges of Interface Design 19](#_Toc417646448)

[6.3 Database Implementation 20](#_Toc417646449)

[6.4 Application & Server Communication 20](#_Toc417646450)

[6.4.1 Sync Logic 20](#_Toc417646451)

[6.4.2 JSON 21](#_Toc417646452)

[6.4.3 Asynchronous Code 21](#_Toc417646453)

[6.4.4 Managing edits/deletes between syncs 22](#_Toc417646454)

[6.5 Image Handling 22](#_Toc417646455)

[6.5.1 AsyncTask 22](#_Toc417646456)

[6.5.2 Other Techniques 23](#_Toc417646457)

[6.6 Application 23](#_Toc417646458)

[6.6.1 Application Views 23](#_Toc417646459)

[6.6.2 Application Features 23](#_Toc417646460)

[6.7 Security 24](#_Toc417646461)

[6.8 Refactoring 24](#_Toc417646462)

[6.9 Unit Testing 24](#_Toc417646463)

[6.10 User Testing 24](#_Toc417646464)

[6.11 Black Box Tests 25](#_Toc417646465)

[6.12 White Box Testing (possibly delete) 25](#_Toc417646466)

[6.13 Usability Tests 25](#_Toc417646467)

[7. Evaluation 25](#_Toc417646468)

[7.1 Final Evaluation 25](#_Toc417646469)

[7.2 Summary of Results 26](#_Toc417646470)

[7.3 Discussion 26](#_Toc417646471)

[8. Final Product 26](#_Toc417646472)

[8.1 Front Screen 26](#_Toc417646473)

[8.2 Users Cookbooks 27](#_Toc417646474)

[8.3 Users Recipes 27](#_Toc417646475)

[8.4 Viewing A Recipe 27](#_Toc417646476)

[8.5 Editing A Recipe 27](#_Toc417646477)

[8.6 Searching/Exploring 28](#_Toc417646478)

[9. Critical Appraisal 28](#_Toc417646479)

[10. Conclusion 29](#_Toc417646480)

[11. Future 29](#_Toc417646481)

[12. Acknowledgements 29](#_Toc417646482)

[13. References 30](#_Toc417646483)

[14. Appendix Outline 32](#_Toc417646484)

**Recipes For Life:**

***Bringing the tradition of cookbooks to the 21st Century***

**Kari McMahon**

**AC40001 Honours Project**

**BSc (Hons) Applied Computing**

**University of Dundee 2015**

**Supervisor: Dr Keith Edwards**

*Abstract - This project details the design and development of Recipes for Life an Android application for collaborative recipe management. For generations collaborative cookbooks and recipes have been a way for individuals to express themselves and share experiences and traditions with others through the format of cooking. But as society increasingly moves more and more into a digital age the concept of collaborative cookbooks could soon be lost in the onslaught of digital information if it is not brought into the 21st century. Many recipe management applications are on offer but none enable the possibility to collaborate on cookbooks and recipes with friends, family and communities - a tradition which has been ingrained in society for many generations. Therefore this project aims to fit this gap in the market and presents a solution which brings the tradition of recipe books to the 21st century through a collaborative recipe management application.*

# Introduction

A recipe book is defined as a “book of directions explaining how to prepare and cook various kinds of foods” (Merriam-Webster, 2014). Although many successful cookbooks do not just represent directions but they represent people’s experiences and their traditions which we can identify with and get excited about often through tantalising pictures, interesting descriptions and innovative ideas (Ruhlman, 2012). For generations collaborative cookbooks and recipes have been a way for individuals to express themselves and share experiences and traditions with others through the format of cooking. But as society increasingly moves more and more into a digital age the concept of collaborative cookbooks could soon be lost in the onslaught of digital information if it is not brought into the 21st century. Currently there are no applications on the market that offer collaborative recipe management and therefore this project presents a solution to this; an Android application called Recipes For Life. This solution aims to bring the tradition of recipe books to the 21st century through collaborative recipe management. The following report outlines the design and development process of the solution as well as reflecting on the successes, challenges and lessons learnt from the various aspects of the project.

# Background

Recipes and cookbooks inextricably link with fond memories. Whether it’s an individual’s memory of their favourite apple pie recipe they baked with their grandma, a delicious chocolate chip cookie recipe they picked up at a charity bake sale or their beloved secret penne arrabiata recipe. Much of society stores these memories away by placing the recipes on scraps of paper, creating their own cookbooks and increasingly placing these recipes on technological devices. By storing these recipes it allows them to keep the recipes to look back on and alter as well as share with friends and family to develop new experiences and memories together. Although as society reaches a point where many individuals are storing and sharing recipes using technology the possibility of collaborative cookbooks and recipes between groups of people could soon be lost. For generations recipe books have enabled others to view our cookbooks and alter and add new recipes but as society moves into this digital age the tradition appears to be going to be lost. This can be seen when browsing through the Apple and Android application stores. There is a large amount of cooking and recipe management applications but these apps do not offer the possibility to collaborate on cookbooks and recipes with friends, families, clubs or even strangers who have similar interests. The recognition of this sparked interest into the research into traditional cookbooks and the benefits they provide as well as research into the current recipe applications on offer to help understand the viability of the project being presented.

## 2.1 Cookbooks as Historical Documents

Recipes have been a part of society for thousands of years with the earliest recollection of recipes being the De Re Coquinara which dates back to the 5th century AD. Since then recipes have played a major part in society for so many years with cookbooks now being seen as informal historical documents. Many papers have explored this topic including Mitchell’s paper Cookbooks as a Social and Historical Document – A Scottish Case Study. The paper examined whether Scottish cookbooks published between 1890 and 1990 are historical markers of major events and technological advances in society. The paper found that “although cookbooks might not record events in society as historical facts nevertheless their contents are often a response to historical events.” (Mitchell, 2001). Similarly in the book Eat My Words: Reading Women’s Lives through the Cookbooks They Wrote - the author examines cookery books from the US and UK from the 17th to mid-twentieth century. The book documents how women from “diverse backgrounds have found the homely cookbook a suitable place to record their stories and thoughts as well as their recipes.” The book aimed to tell the untold stories of these women from the cookbooks. The author used the book to examine these stories to get others to consider cookbooks as worthy objects of serious textual analysis through demonstrating how the recipe books told the history of that time. As well as this the author also demonstrated how recipe books could be used to maintain a connection between individual’s from different centuries and how this connection was formed exclusively through the medium of cooking - “How deep are the connections between us – a 17th century mother and myself –despite the time that has elapsed.” (Theophano, 2003). These papers and books demonstrate how cookbooks can be seen as historical documents which can tell the story of the time and are beneficial in helping others understand that time period through the analysis of these documents. It is clear that cookbooks appear to stand the test of time and that individuals can still connect with documents that are hundreds of years old. Losing cookbooks would mean losing documentation that can tell the story of a specific time period whether it’s learning about an individual’s specific story or learning about the varying diets or food/cooking trends for cultures all around the world in that specific time period.

## 2.2 Traditional Cookbooks and Their Benefits

Although the cookbook being seen as an informal historic document has brought its benefits to historians and researchers. The main key to the cookbooks survival and success over so many years is the benefits it brings to society and communities through its collaborative features. For example the paper Intensifying Taste, Intensifying Identity: Collectively Through Cookbooks aimed to look at the negatives of community cookbooks by examining whether they implicitly rebuke a social order that devalues women’s work by focussing on Lutheran church women. But in fact the author found the process of creating a collaborative cookbook gave a voice to the community through building the cookbook, organizing it, discussing experiences and producing and selling them. The author found the cookbook was not just a way to raise money but also a way to recognise each of the ladies knowledge and experience of cooking and share it with those who were interested. (Ferguson, 2012). Similarly the book Recipes for Reading contains a collection of essays that demonstrate the different stories available in cookbooks stating that they “tell stories – autobiographical in some case, historical sometimes and perhaps factious or idealized in other instances.” The book explored how cookbooks were used to voice stories and experiences and how the author could see that recipes were increasingly becoming “readable with great benefit to our knowledge of women’s experiences and discourses” (Bower, 1997). The appeal of collaborative cookbooks is the ability for an individual to input and share their story with others. What appeals to those who read these cookbooks is that they get to hear and experience these stories all through the medium of cooking. The benefits of community and collaborative cookbooks are best summarized in the book Eat My Words - “modifications and modernizations of old recipes and the invention of new dishes in a woman’s cookbook represent the combined effort of many people. Contributions may come from past circles, sometimes from one or more cultures and while we tend to think of cooking as a delight to our senses, the relationships formed through these culinary compositions are social, cultural and economic.” (Theophano, 2003). The benefits that are often re-iterated in the papers and books is that cookbooks enable us to have a voice, tell a story and share our experience and knowledge with others. Collaborative cookbooks have often been a way for communities to come together and bond, so to suddenly lose these benefits in the digital age would be a detrimental loss to cooking and to society. Therefore it seems necessary to try and move collaborative cooking into the digital age.

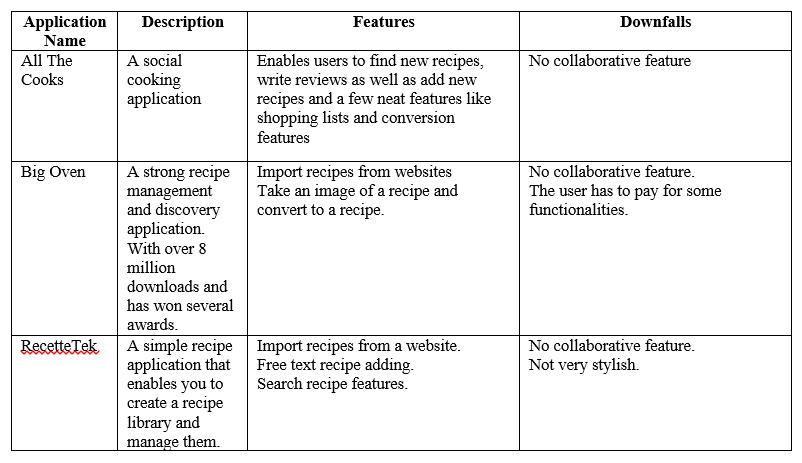
## 2.3 Current Recipe Applications

AllRecipes.com, an extremely popular recipe site, states that one-third of online cooks use smartphones to look up recipes. They found that 44% of cooks preferred using cooking websites over 19% who preferred to use traditional cookbooks and in the past 15 years the use of cooking websites has surged 207% (All Recipes, 2012). It is clear that technology is becoming the top cooking resource and although traditional collaborative cookbooks have been seen to have many benefits, they are beginning to be lost in the mass offerings of technological cooking resources. Tablets are now being used as a replacement to the traditional cookbook. AllRecipes.com in 2013 stated that social referrals from tablet devices to their website increased by 787% from 2012 to 2013 as well as a rise in page views on the website from tablets (All Recipes, 2013). The popularity of tablets in the kitchen can also be seen by the vast number of kitchen accessories on offer for tablets such as tablet kitchen stands and covers to protect from spillages, an example of this can be seen in figure 1. With accessories like these making their way onto the market and the statistics seen from AllRecipes.com tablets seem the way to go when creating technology applications for the kitchen.



Figure 1: Belkin Kitchen Tablet Mount

To help the student understand if there was room in the market for the collaborative recipe application being proposed research was focussed on current popular applications available for tablets. The student explored a few applications from big name companies and a few smaller independent applications. The student explored independent applications as she would be more likely to be competing against those. Figure 2 outlines a few of these applications with their features and downfalls. Many of these applications are popular with a high amount of downloads and star ratings. But all of these applications lacked a collaborative feature that enabled users to set up and maintain shared cookbooks. The research showed there is room on the market for the project and also showed that recipe apps are highly desired and used by a large amount of the population. The research also enabled the student to see the successes of current apps and see what users currently liked and disliked about these apps, which could then be possible features that would work in addition to the collaborative features of this project.

Figure 2: Current Applications Research

# 3. Specification

This project was the idea of the University of Dundee Applied Computing student Kari McMahon who also undertook the design and development of the project. The core aim of the project was to create a minimum viable product that enabled users to collaborate on recipes together in shared cookbooks.

Since the project was the student’s idea there was no client interaction to gain the specifications for the minimum viable product. Therefore the student created the specifications based on a combination of own ideas, data gathering and research.

## 3.1 What Is A Minimum Viable Product?

*“A minimum viable product (MVP) is the most pared down version of a product that can still be released. An MVP has three key characteristics:*

* *It has enough value that people are willing to use it or buy it initially*
* *It demonstrates enough future benefit to retain early adopters*
* *It provides a feedback loop to guide future development*

*The catch to this development technique is that it assumes that early adopters can see the vision or promise the final product and provide the valuable feedback needed to guide developers forward. “*

* (Technopedia, no date)

A minimum viable product (MVP) is a way for developers to create a product that is market led and desired by consumers through accelerated learning. An MVP is producing a product that has just enough functionality to gauge whether there is a market need for this product. The issue with creating this project as a minimum viable product is that there is already a significant amount of recipe applications on the market with many of these already having a large amount of functionalities. Therefore the bar is already set high for the market expectations of what recipe apps can do. The challenge in creating an MVP specification for the project is having the majority of the requirements predominantly focussed on collaboration (the apps core aim). As well as having enough additional functionalities that will put the app on a level playing field with current recipe applications on the market.

## 3.2 Initial Specification

The initial requirements were in an informal list. The requirements were in two categories functional and non-functional. They were set out with shall, should and may; meaning the requirements that are required, recommended and optional. The initial requirements proved as a starting point for understanding the applications functionalities. With a list of possible requirements gathered from research, a survey was then created to learn more about the demographics for the application and to help prioritise and find new functionalities based on the target markets needs and desires. An example of these initial requirements can be seen in figure 3.

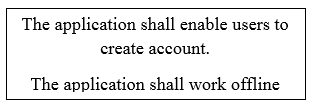
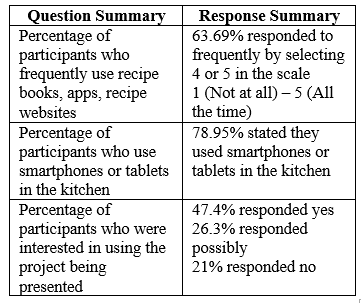


Figure : Requirements specification example for functional and non-functional requirements

## 3.3 Data Gathering

To help better understand the target market and gather specifications for the application, a survey was created and placed on the website SogoSurvey.com. SogoSurvey was selected over popular survey sites like Google Forms or SurveyMonkey because it was free to use, had a wide range of different question types and had useful analysis tools. The survey asked for anyone over the age of 18 who was interested in cooking and cooking with technology to fill out the survey. The survey was distributed via email, on cooking forums and through communication with those interested in the project. The aim of the survey was to understand the applications target market and gather information to help create the final specification. The survey laid out the 17 possible functionalities that had been set out in the student’s initial specification. The participants were then asked to rate these functionalities in order of usefulness and suggest any other functionalities.

The survey received 19 responses from a wide range of ages with 13 of the participants being female and 6 being male. Several of the participants in the survey were from a cooking background where their occupations were a pastry chef, dietician and private chef. In figure 4 a summary of some of the responses from questions in the survey can be seen.

Figure 4: Summary of some of the survey responses

The responses from the survey showed the participants had a clear interest in cooking using technology, the majority liked the idea of the application being presented and they were from a wide range of backgrounds which was useful for gaining a varied response. Therefore the responses relating to the rating of application features were taken into consideration when developing the final specification document. The survey responses were used in connection with the initial requirements document to develop the final specification. The data gathering was beneficial in showing that there was interest in the project being presented and therefore it was worth pursuing. Having the respondents contribute towards the requirements document helped create a market led specification document. The full survey results can be found in the appendix.

## 3.4 Final Specification

The final specification was created using data gathered from the survey, the student’s ideas and research. The student examined the ranking of requirements via a frequency table generated by SoGoSurvey. The ranking of the requirements determined the priority in the specification document. Occasionally requirements that were not ranked highly by participants still ended up high in the final requirements document because they were essential to the creation of the application. The requirements marked with shall were the requirements essential to creating a minimum viable product. The full specification document can be found in the appendix and an example requirement can be seen in figure 5.

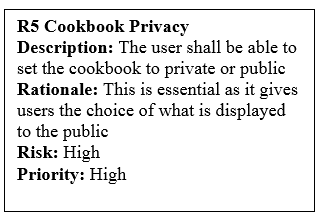


Figure : Example Final Requirement

## 3.5 Specification Flexibility and Managing Requirements

The type of formalised requirements document discussed in section 3.4 is often rigid and fixed. Since the project was the student’s idea and was not a contractual agreement with a client this enabled some flexibility in the specification document. To help create flexibility with the specifications the technique of a task board was used. The task board was popularized by the Agile methodology and enables “a snapshot of the current sprint backlog allowing everyone to see which tasks remain to be started, which are in progress and which are done” (Bowes, 2014). The key benefit of the task board is its simplicity, the developer can easily move tasks around if priorities change and add/remove new tasks if the developer realises they are missing a requirement. The task board’s simplicity means it easily provides a daily snapshot of progress (Agile Alliance, no date). In the project the student used the task board by splitting the formal requirements into smaller requirements in a user story[[1]](#footnote-1) format and then storing them in the online task board Trello. The requirements are then stored in the same priority as listed in the specification document. The higher they are on the Trello list then the higher priority the requirement is. The tasks are then marked with a colour – red for to do, yellow for in progress and green for done. This marking helped for an easy snapshot of progress. Trello makes requirements easy to move around and helps make the project more adaptable to change.

When the final specification was set and moved to Trello, the requirements were then managed on Trello throughout the project. Using a task board helped create a project that was flexible and information led instead of a process where requirements were fixed and based on assumptions made at the start of the project. An example scenario was requirement R1 (Recipe management) was higher priority than R2 (Account creation). At the development stage it was clear creating an account should be developed before recipe management as users accounts are linked to recipes. So by utilising Trello the student could easily log on and move the requirements around based on this information. Also splitting the requirements into smaller and more manageable chunks, as seen in figure 6, made tasks easier to achieve and provided clarity on the priority of some requirements. For example due to time constraints creating a recipe and cookbook were more important tasks to achieve than deleting a recipe; the use of Trello enabled the student to easily address and manage these priorities. Often in development it became clear that a new requirement, which hadn’t been clear at the initial requirement stage, might need to be added. Using the task board made it simple to add or remove requirements based on new information.

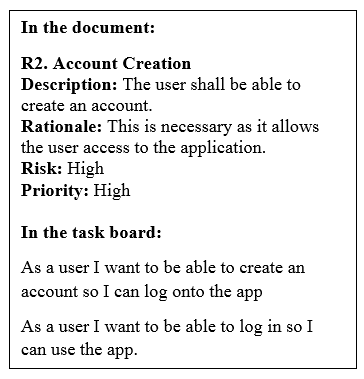


Figure 6: Example comparison of requirement and user story

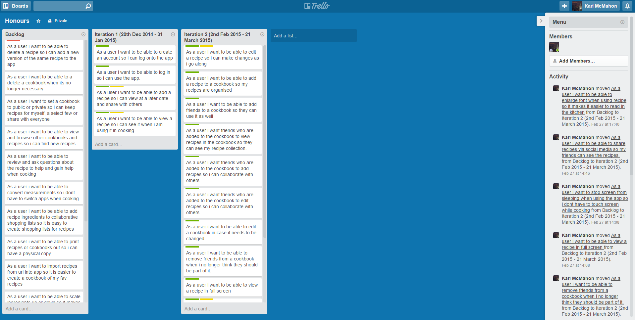


Figure : Trello Task Board

# 4. Project Management

## 4.1 Methodology

A software development methodology is a “framework that is used to structure, plan, and control the process of developing an information system” (IT Knowledge Portal, no date). Recipes for Life is a large project and it was necessary to select a software development methodology that would help structure and maintain the project. There are two core software development methodologies – waterfall and agile. Waterfall is the traditional software development process; it is a linear and rigid approach that does not embrace the inevitable changes or revisions that often occur within projects (IT Knowledge Portal, no date). Whereas Agile is a set of development processes which are flexible to change, encourages working code over documentation and allows frequent opportunities to assess the direction of the project throughout the development lifecycle (Highsmith & Cockburn, 2001). The Agile approach enables a flexible project which can be evaluated honestly and realistically through interaction and collaboration at frequent intervals.

An Agile approach was most suitable for this project as requirements, design and evaluation could easily change as the project was explored in more depth and therefore a software development process that enabled flexibility was essential. The most popular Agile methodology is SCRUM which is a very customer centric approach and is based on high levels of customer and team collaboration. This project lacked an official customer and was being developed by an individual so it was necessary to find a flexible Agile development approach that had less focus on client and team collaboration. An Agile approach suitable for this type of task is the iterative development process. The iterative development process allows a system to be developed in iterative cycles. The process starts with an initial planning stage where the initial requirements and design are set. Then iterations are incorporated in “mini-project” cycles which repeat until the product is ready for delivery. At the end of each iteration the individual or team has an opportunity to re-evaluate the project and re-organise and change aspects of the project to fit the needs of the team, individual or client. The iterative approach is very flexible and helps create a final product that is user centred and information led (Bittner & Spence, 2006). For the needs of the project the iterative approach was adopted and built upon to create an iterative strategy appropriate for the project. The diagram in figure 9 outlines this strategy.

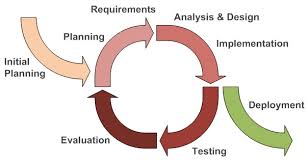


Figure 8: Iterative Development Process [[2]](#footnote-2)

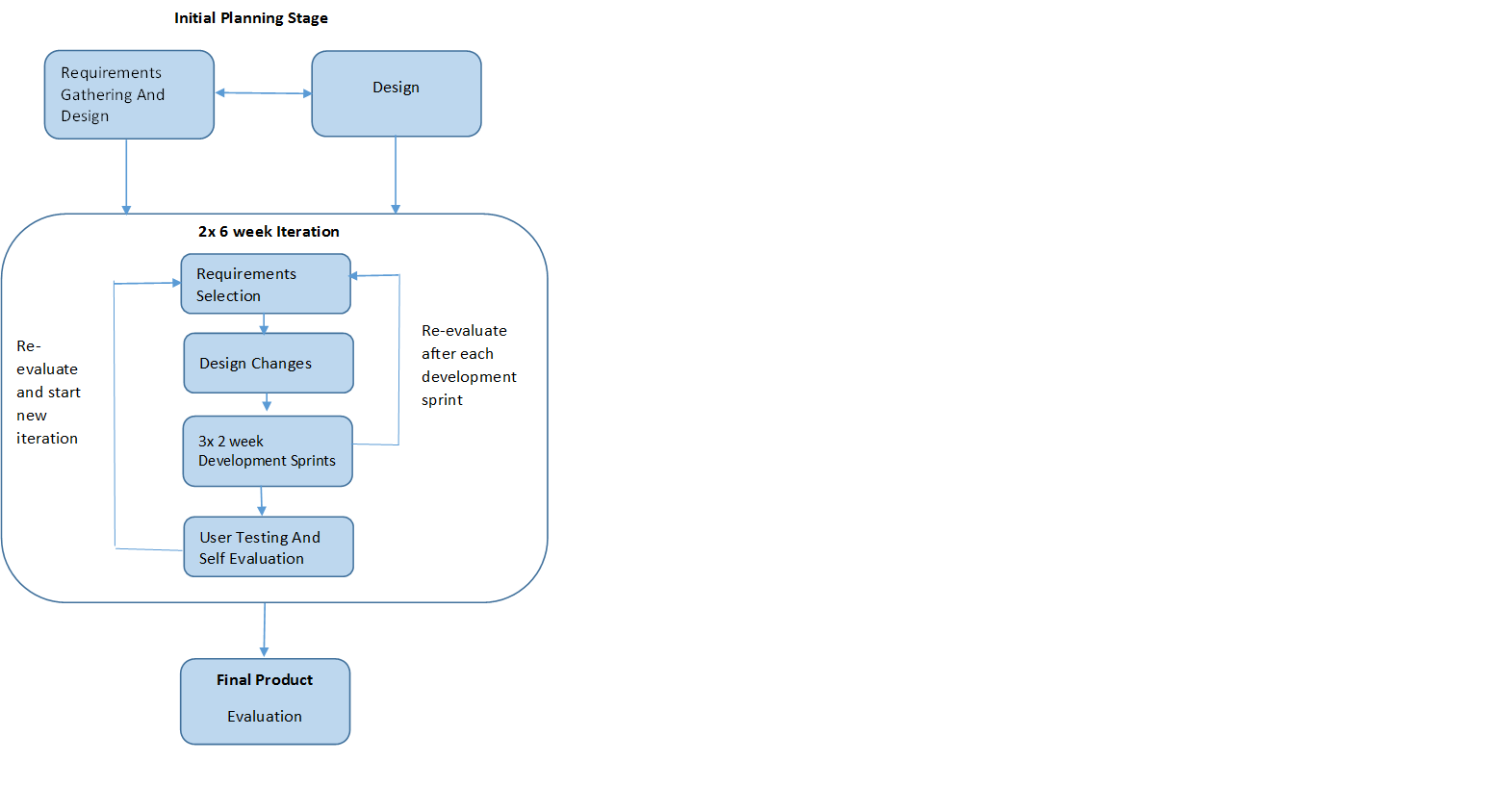


Figure 9: Iterative Strategy For Project

## 4.2 Project Management Tools

### 4.2.1 Initial Project Plan, Gantt Charts & Trello

At the beginning of the project an initial plan was developed by the student outlining an overview of the project plan for September 2014 till May 2015. The key function of the plan was to provide the supervisor with a guideline of the estimated project timeline. This plan was then further developed into a Gantt chart at the start of the project. Gantt charts give a graphical illustration of a schedule that helps to plan and coordinate tasks (Rouse, 2007). Gantt charts are frequently used as management tools which facilitate visualisation of the overall project. The student chose to employ a Gantt chart as it provided a snapshot of the overall project plan and allowed tasks to be visualised and milestones to be identified over the course of the year. The Gantt chart was utilised at regular intervals to evaluate overall project priorities and measure progress. Trello which was discussed in section 3.5 was used for a lower level project view when the student required assessing in detail whether specific requirements had been achieved, were progressing or were still to be tackled. The combination of the Gantt chart and Trello meant it was easy to see project progress and re-prioritise if necessary. This contributed to good organisation throughout the project. The initial project plan, Gantt charts and a link too Trello can be found in the appendix.

### 4.2.2 Sprint Backlogs

A sprint backlog is a list of the tasks and requirements to be completed within the sprint (Mountain Goat Software, no date). For this project sprint backlogs were used for the 2 week development sprints that occurred within the iterations. Requirements would be selected from the product backlog for a 2 week sprint and then these requirements would be split into small tasks on the sprint backlog with an estimated time to spend to achieve these tasks. Each day in the sprint the amount of hours spent on the task would be logged and monitored. The sprint backlog helped show the progress of development and helped the student analyse the amount of time being spent on requirements. In this way the student could track on average how quickly the requirements were being developed. This made weekly task commitments more realistic as what could be achieved in a given time frame was explicit and this assisted the student with setting realistic time frames for specific tasks. The sprint backlog provides flexibility in that if an individual has underestimated what they can do in 2 weeks then new requirements can be easily added. Sprint backlogs are a simplistic but effective method to monitor and analyse development progress providing a snap shot of day to day progress. These sprint backlogs were created in excel and can be found in the appendix.

### 4.2.3 Risk Assessment

For a project to be successful it is essential to outline the possible risks and problems within the project at an early stage and create contingency plans for these risks. This is common practice within the software development industry. For the project a risk assessment was created at the research and design stage. It outlined the risks for the project in order of priority with contingency plans for each risk. The prioritisation of the risk is based on a value from 1 to 10 representing the likelihood of the risk occurring and the loss if the risk did occur. These two values are multiplied together to give the severity which is used to calculate the priority. Risk assessments enable developers to think ahead about any risks that might have a serious impact on the success of the project and help create solutions to best limit them. This is essential in a large project like Recipes for Life where a risk such as losing files at a late stage could be detrimental to the success of the project. Therefore it was essential to develop a risk assessment for the project. The risk assessment can be found in the appendix.

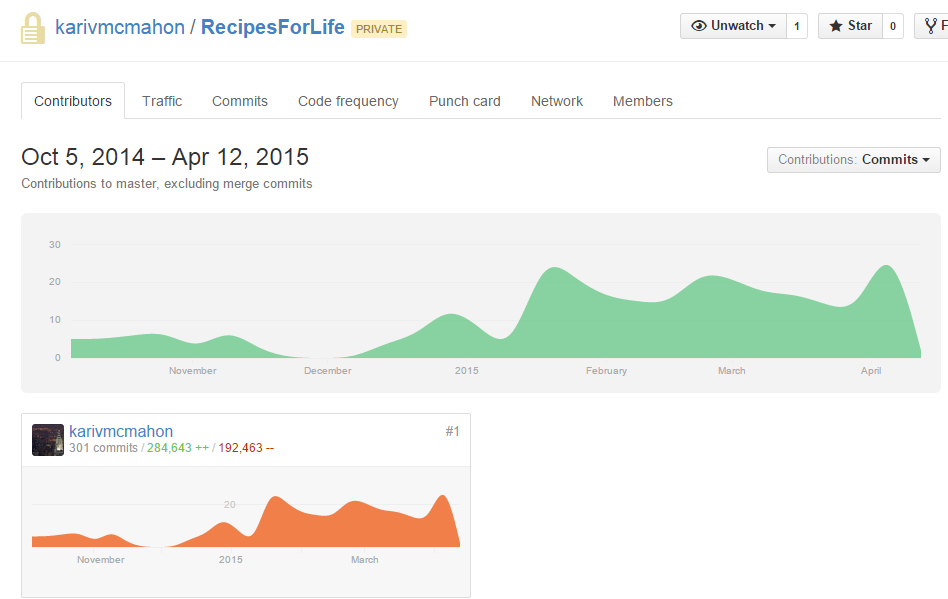
### 4.2.4 Supervisor Meetings & Minutes

Supervisor meetings were scheduled weekly throughout the majority of the project. A supervisor meeting is a useful way to help manage the project because the student receives expert guidance and an objective perspective. This perspective is useful when the student is struggling to make progress and to deal with the challenges inherent in project development. An objective perspective provides suggestions and potential strategies that the student had not previously considered. At each meeting the student has a chance to present their work and based on this the supervisor can give advice and guidance on the student’s project or their progress. This is a useful way to address any progress or project issues at an early stage. The meetings give the student an opportunity to benefit from the supervisor’s knowledge and experience which is invaluable. Minutes were kept to enable reflection on the meetings and overall project process. These can be found in the appendix.

### 4.2.5 Log Book & Github

A log book was maintained by the student throughout the project as an informal diary of work. The log book enables the student to write about the tasks achieved on a day to day basis as well as their rough notes, ideas or research throughout the project. The log book was a really useful and informal way to document the daily progress and vision of the project.

Github was selected by the student as the versioning control system to manage all the files for the project. This enabled versions of files to be stored in case any files needed to be recovered. Github enables access to files wherever there is an internet connection, the ability to revert to old versions and the ability to store a variation of file types. Github was selected over other versioning control systems like Subversion or CVS because the student already had experience with Github and had a private account on Github. As well as this Github is also well supported with over 6 million people using the site (Github, no date) so there is a large amount of support available if any difficulties occur. Github’s availability offers flexibility and therefore based on these reasons was selected for file management. Github also acted as another informal logbook with daily commit messages logging day to day progress. The visualisations produced on Github based on the students commits helped visualise the student’s overall progress.



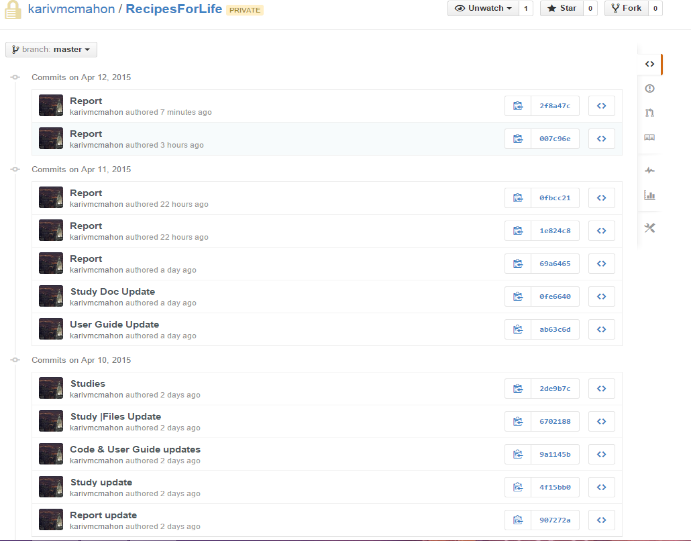


Figure : Github Project Visualisations

## 4.3 Realities of the Methodology

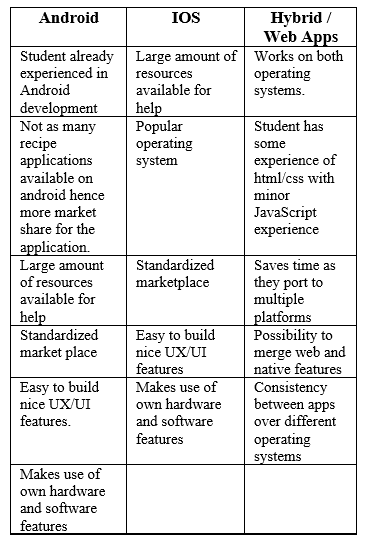
The student followed the iterative strategy laid out in section 4.1 and figure 9. Although some aspects of the strategy were modified as the project progressed. For example in the strategy it was set out that user testing sessions would take place after the 2 iterations but after the first 6 week iteration there was not sufficient functionality in the application to justify user testing being undertaken at that stage. As a consequence in the second iteration two user testing sessions had to be undertaken ,one early in the iteration and one late into the iteration. Also in the plan it stated there would be 2x 6 week iterations but in the second iteration an extra 2 weeks had to be included to enable more functionality to be added to the application. Although the iterative strategy did not go exactly to plan because of its flexibility the adaptations could be made without causing too many problems allowing the needs of the student to be met.

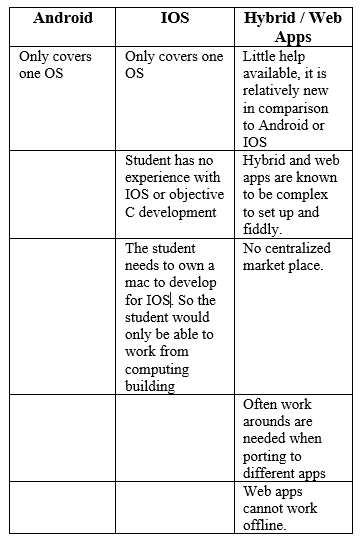
# 5. Design

Before implementation could start on the project several design decisions needed to be made which are discussed in the following sections.

## 5.1 Application Operating System Selection

The application for the project was being developed predominantly for tablets as they are becoming increasingly popular for use in the kitchen. This necessitated a tablet operating system being selected prior to development. In terms of tablet operating systems there are three main options which can be developed for. These are Android, IOS or Hybrid / Web apps. In 2014 IDC.com stated the worldwide smartphone operating system market share for Q2 2012 was 84.7% for Android, 11.7% for IOS and others making up 3.7% of the market share with the predominant vendors being Samsung with 24.4% market share and Apple with 11.7% market share. IDC.com stated the tablet OS market share was 53.8% Apple and Android 42.7% (IDC, 2014), (IDC, 2014), (McCracken, 2013). In terms of the two major operating system competitors it is quite a close call between them but Android comes out slightly stronger having a wider overall reach over Apple. This data was taken into consideration with the advantages and disadvantages of each operating system based on the student’s previous experience and research. These advantages and disadvantages are outlined in tables 1 and 2.

Table 1: Advantages of Android, IOS and Native Web Apps[[3]](#footnote-3)

Table 2: Disadvantages of Android, IOS and Native Web Apps[[4]](#footnote-4)

Android was the leading operating system in market share in smartphones in the last year. Android also had the largest amount of advantages with the least amount of disadvantages based on the research and the student’s previous experience. Therefore Android was the natural choice to develop for. Although hybrid/web apps were also a strong contender for the development of the application. The main reason for not selecting this option was the lack of support and documentation available which was a concern to the student especially when working on such a large project.

## 5.2 Server Side Design

### 5.2.1 Database Requirements & Selection

Database consideration played a significant role in the design stage as the application handles a reasonable amount of data from multiple users. The student was already aware that Android devices come with a SQLite database built into the phone; this enables individuals to retrieve data from the database directly without being connected to the internet. A SQLite database on the phone means applications can be very responsive as there is no time waiting to be connected to the server and applications can work offline. It was important for the student that Recipes For Life could work offline because the users may want to retrieve recipes to look for ingredients while in the supermarket or may not have wifi in their kitchen. But it was also important for the student to have a database on a centralized server because users can be contributing to other cookbooks or searching for other recipes. This means it’s necessary to insert and retrieve this new data from a central database and from the users’ phones so that all the users can access new recipes, cookbooks etc. Therefore this meant having the database on each of the individual’s phones and a central database on the server and a sync functionality between them.

Although SQLite is built into the Android device, a new and upcoming NoSQL database has been developed for Android as well called Couchbase. The student explored the SQLite and Couchbase options before deciding which would be used on the device. SQLite is a lightweight, self-contained database that is memory efficient and can handle terabyte sized databases which makes it suitable to be embedded on smartphone devices (SQLite, no date). It is a full SQL implementation which makes it very easy to use if the developer already has experience of SQL. Whereas Couchbase is a NoSQL database solution for the Android device. NoSQL databases are schema free, can handle large volumes of data, are suitable for scaling and have high availability and strong disaster recovery (MongoDB, no date). Couchbase at first seemed the better choice for the project because the application may need to handle a large amount of data and may need to scale in the future which Couchbase offers. Couchbase also states they offer a sync functionality between the phone and the server which would be a useful tool for this application. The issue with Couchbase was it was relatively new and lacking significantly in support and documentation in comparison to SQLite. To help make the choice the student drew out the basic database design for the project and found it was hard to visualise the projects database in NoSQL, so taking this into consideration with the lack of support and documentation for Couchbase the student chose SQLite to be used on the device.

With SQLite selected to be on the device, a SQLite or SQL database needed to be chosen to be used on the central server. There wasn’t much difference between selecting a SQLite or SQL database as they are both equally suitable and it was mainly down to the student’s preference. The student selected a SQL database on the server because they have more robust and sophisticated database management tools which are free in comparison to SQLite management tools which are very lightweight. Another reason for this choice is SQL databases tend to be more supported for querying from server side scripts such as a C# or PHP script than SQLite.

### 5.2.2 Database Design

After selecting the database an in-depth database design was developed. The design of the database went through multiple iterations in the design phase. It was important that all the tables in the database had a timestamp which would be used to track the differences between the application and server databases which would allow for an accurate sync between them. The database design was complex as a large amount of tables needed to be connected to the recipe table to allow for collaboration and detailed recipes. A snapshot of the database design can be seen in figure 11 and the full database design can be found in the appendix.

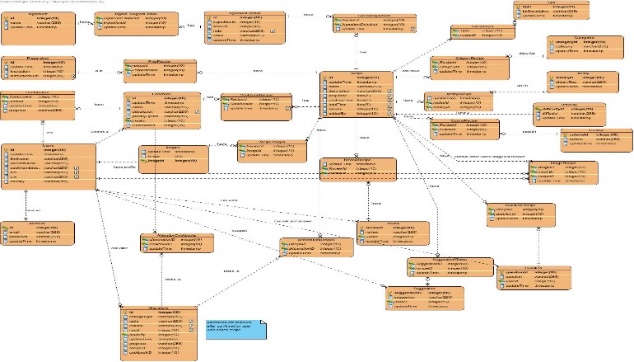


Figure 11: Database Design

### 5.2.3 Server Side Development Language Selection

On the server side it was necessary to have scripts that would be called by the application to insert and retrieve data from the central database. The two main options for this was PHP and ASP.net/C#. PHP is a “widely used open source general purpose scripting language” (PHP, no date) and ASP.net is a “development framework for building web sites with html, css, javascript and server side scripting” (W3C, no date). Both of these languages are widely used and have strong support networks so the choice of the language predominantly came down to experience. The student was experienced in both languages but had used PHP with very simple Android syncing in the past and found it straightforward so therefore felt it was suitable to use again.

### 5.2.4 Server Side Development Tools

Microsoft SQL Server Management Studio was chosen to manage the central database as it is a SQL server database which was provided by the School of Computing for use in the project. Microsoft SQL Server Management Studio was selected as it is recommended to use with SQL server databases, it is free for students and it's a premiere Microsoft product so has a lot of support and documentation and is easy to set up. To write and manage the PHP scripts notepad++ was selected as it’s a lightweight, fast and simple tool which has a lot of code editing features built in that seemed suitable for managing the PHP scripts (Web master Format, 2009).

## 5.3 Application Design

### 5.3.1 Application Layout & Design Patterns

The application was designed with a Model-View-Controller design pattern as it is heavily database orientated and the MVC pattern enables separation of the business logic and view. The benefits this pattern brings is that it limits code duplication and enforces code re-use making code more flexible and easy to test functionality independently (Kotek, 2002). The application will have classes split into the packages model, view and controllers (beans). Figure 12 demonstrates the mvc design pattern.

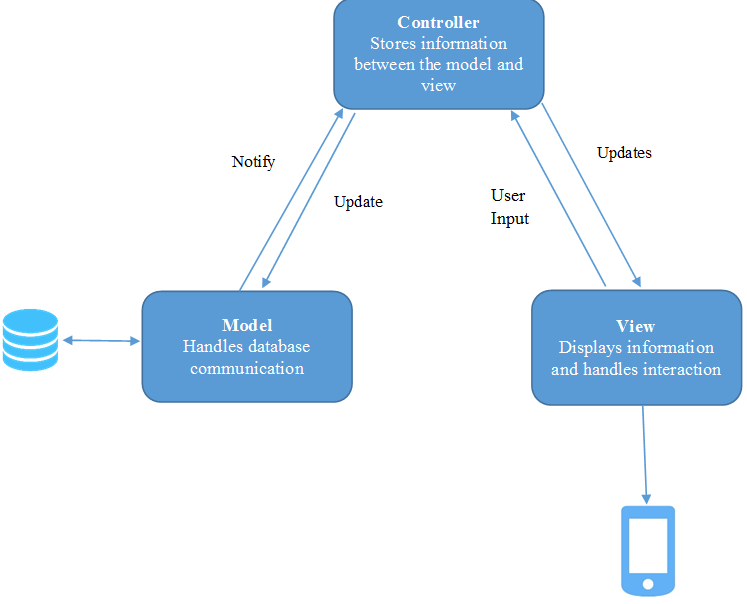


Figure 12: MVC Design Pattern

The application will have a model package which contains classes that purely send and retrieve data from the database. The controller contains information beans which will store the information that is being sent too or retrieved from the database. The view will contain all the classes (activities) which the user will interact with and retrieve the data from user input which will then be stored in the controller. This pattern is demonstrated in the UML class diagram created for the application which is seen in figure 13 and in the appendix. This class diagram represents the classes the student thought would be used during the implementation at the design stage.

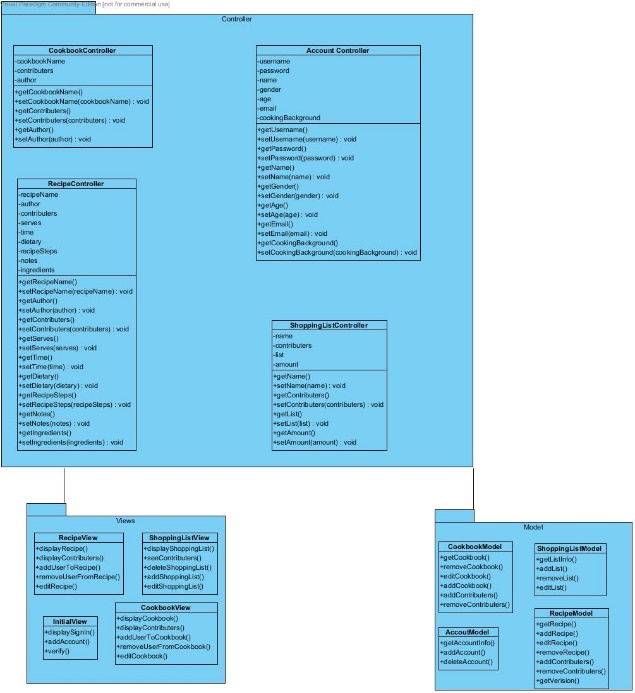


Figure 13: UML Class Diagram

### 5.3.2 Application Language Selection

Java is an object oriented programming language which is platform ubiquitous and has a vast array of 3rd party libraries available for use (IBM, no date). Java has been around since 1995 and therefore a significant number of developers have worked with java and a significant amount of documentation is available for Java (Oracle, no date). Google selected Java as the language to be used to develop Android applications and therefore is the language that was used to develop the application. To write unit tests for the application Junit was selected as the android test suites are based on Junit (Android, no date). Junit is a unit testing framework for the Java Programming Language (JUnit, no date), it links with the android test suites which makes it easy to write and run tests for Android applications.

### 5.3.3 Application Development Tools

There are two IDEs available for Android development – Eclipse and Android Studio. Android Studio is an intellji editor that is set to replace Eclipse at some point in the future but it was still in its beta stages while the student was designing the application (Android, no date). Eclipse is the original Android IDE of choice, there is sufficient support for issues in eclipse and it is stable. However the Eclipse editor is also bulky and can be very slow when doing development work. The deciding factor between Eclipse and Android studio was stability and support. Eclipse was more reliable for these factors. The student also already had Eclipse set up and had experience using it, so these factors influenced the choice of the IDE.

## 5.4 Application & Server Communication

As previously discussed the application should be able to work online and offline. Therefore a sync functionality is required between a centralized database on the server and a database on the phone. To show how the sync should work a diagram is displayed in figure 14.

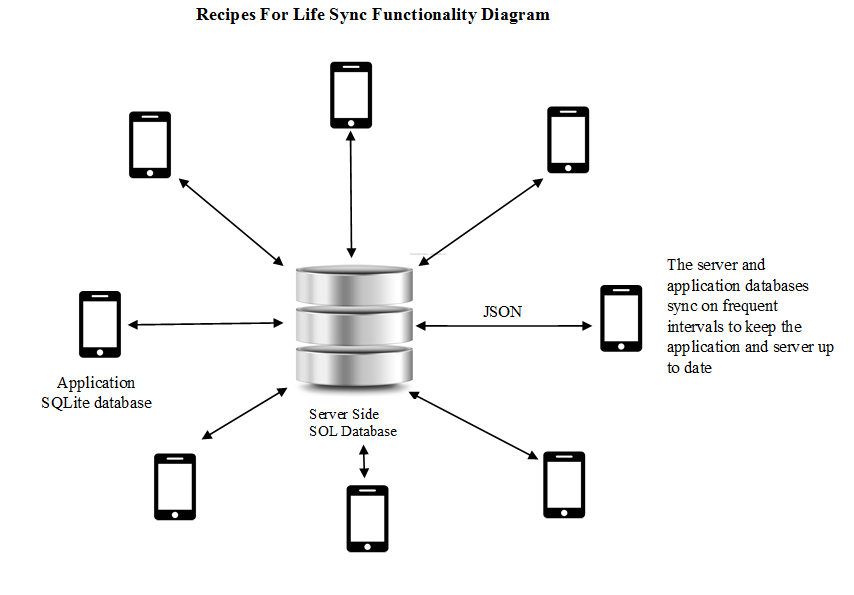


Figure 14: Application & Server Database Communication

This system involves having a database on the server and a Sqlite database on the phone. Each table in the database stores a timestamp and differences between the server and phone timestamps to help find the rows that need to be sent or retrieved to one of the databases. These rows are often sent through a data exchange format between a server side application and the device. To make the syncing possible it was important to select a data exchange format that would be easy to use within the application and server side code.

The two main choices for the data exchange formats were XML and JSON. XML is a markup language for documents containing structured information such as text or images with some information about the role this data plays like headers or paragraphs (Walsh, no date). JSON is a lightweight data interchange format that is easy for humans to understand but is also easy for machines to parse and generate (JSON.org, no date). The choice selected was JSON as it is easy to read and it is really simple to create and parse JSONs in both PHP and Java therefore it seemed more straightforward to use JSON than XML.

## 5.5 Collaboration Feature

An important aspect of the application is collaboration and therefore the student considered this in depth at the design stage. The student designed collaboration to work by allowing the user to add other individuals to a cookbook. The individual who is added to the cookbook will then see the cookbook on their own shelf. The user will then be able to add, delete and edit recipes that are in the cookbook to help create collaboration.

The student also explored additional collaboration features that could be used to compliment this feature such as a versioning control system for recipes, cloning of recipes so if the user wants to change a recipe drastically instead of editing it they can clone it and adding reviews or tips to a recipe for small changes instead of the user having to edit a recipe. The student also explored the idea of users being able to ask to join cookbooks. But these possible additional features being achievable were dependent on development capabilities and time constraints at the implementation stage.

## 5.6 Paper Prototypes

Paper prototypes are a simple and efficient way to explore ideas at the design stage. The student used paper prototypes when trying to develop a design for the application. It enabled the student to visualise the application without spending too much time on the finer details (Medero, 2007). The paper prototypes went through several iterations often being re-developed when new information was discovered at the research or design stage. The paper prototypes were a cheap and quick way to explore design ideas and were beneficial in the design of the application.

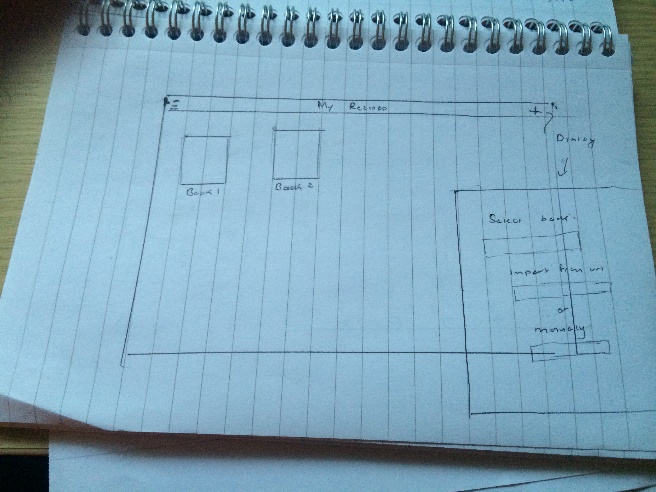
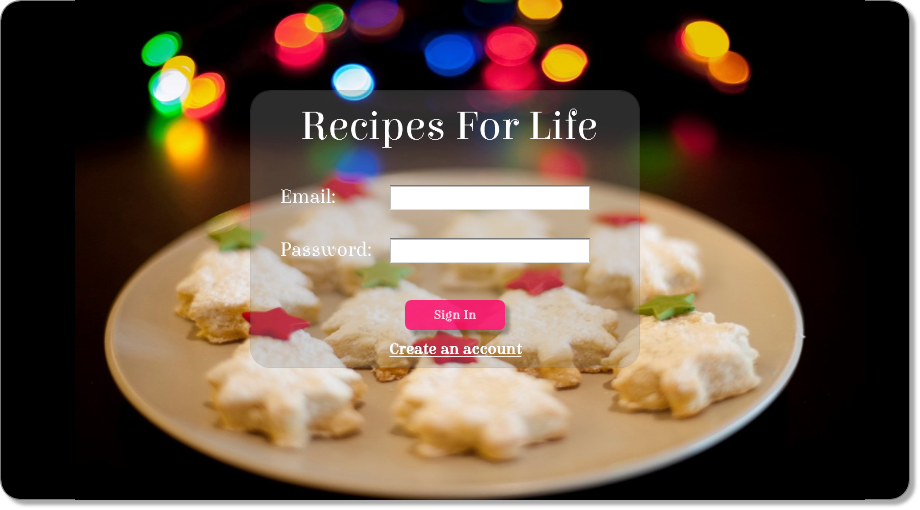


Figure : Paper Prototype

## 5.7 Design Sketches

Once the finalised design sketches were created on paper they were then moved onto Axure. Axure helps create a more polished design sketch by exploring widgets, placeholders and font styles (Axure, no date). This makes it very easy to create sketches that resemble a real application. Axure helped the student pull out additional ideas and possible challenges with the design. Axure was useful because the student could load the design and easily make changes to the application based on feedback or requirement changes which is helpful when developing a user centred design.

****

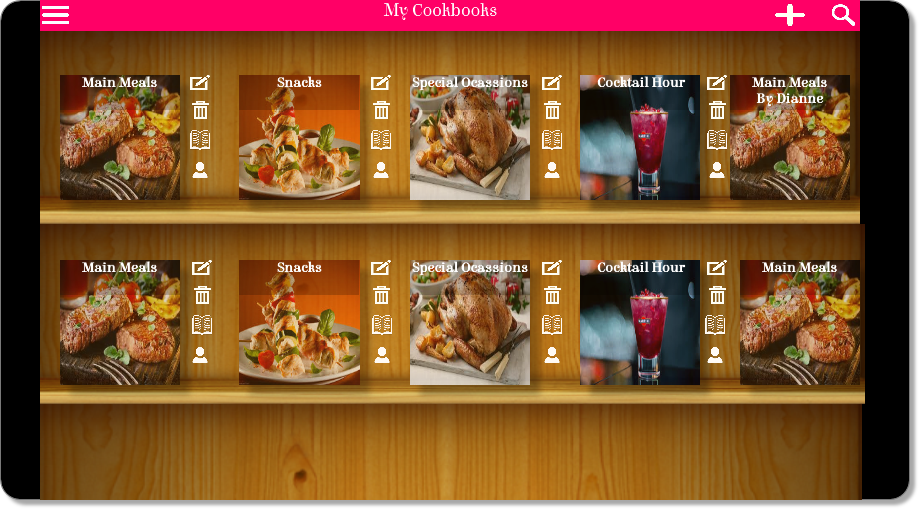
****

Figure 16: Design Sketches

In the design sketches the student viewed it as important to create an application that had a unique style. She wanted to represent cookbooks as users would see them on a day to day basis. The student did this by incorporating shelves which the cookbooks would sit on, similar to what the user would have at home. The image provided by the user for a cookbook is placed in a square book like shape. The title provided by the user is then overlaid at the top of the image which makes it appear like the books title. These books are then placed on the shelf once created.

The application design is icon centric with familiar icons being used to help create a system that has a match to the real world. The colour scheme for the application is bright pink and white because it creates a good contrast, helps catch attention and is aesthetically pleasing.

## 5.8 Ethics

The application aims to be user centred, this involves continuous user feedback throughout the project. To have continuous user feedback an ethics application has to be submitted and accepted. The submitted ethics form outlines the various techniques involving users that will be used to aid the creation of a user centred application. The techniques chosen for this project was an anonymous survey, interviews/focus groups, user testing and evaluation. An anonymous survey was used to gather information about the target market and help gather/prioritise requirements for the application, interviews/focus groups were used to gain opinions on design sketches, user testing to understand how usable and easy to understand the application is and an evaluation against a popular recipe app to see if the application could compete on the market. Ethics enables the student to undertake user studies in the correct manner. Ethical approval for the project was gained in November 2014; this meant the project could have users involved throughout which would aid the development of a user centred application.

## 5.9 Intended Audience

The intended audience for the application is any individual with an interest in cooking, particularly those with an interest in using technology when cooking. The student created persona’s which represent various different segments of the intended audience. These personas can be found in the appendix.

## 5.10 Design Focus Group

A focus group was undertaken at the design stage of the project to help create a user centred application. The focus group had 6 participants, 4 female and 2 male between the ages 18 to 60. The focus group was very informal with the Axure design sketches and the project idea being presented to the group and their feedback was noted down. The overall comments were that the application was clear, consistent, interesting and colourful with some minor changes to be made. There was issues with a few icons being hard to understand, for example the change font size icons. Participants also struggled to read headers where pink font was used. This feedback was then taken into consideration in the implementation stage.

# 6. Implementation & Testing

## 6.1 Changes from the Design Stage

During the implementation some of the plans from the design stage changed due to new information in the project. The main change was at the start of the implementation phase when the student went to set up scripts to communicate with the server database. When she tried to use PHP with MS SQL database she recognised it wasn’t set up to do this. The student recognised the issue during the holiday period where the staff who maintain the server would be on holiday. Instead of waiting for this to be fixed and halting development time, the student chose to write the scripts using ASP.net and C# as it was setup to access MS SQL database and she had used ASP.net/C# in the past. Although she did find a small learning curve with this choice as it had been a while since she had used ASP.net and C#. This learning curve meant it took some time to write the initial scripts for syncing but once this hurdle was overcome using ASP.net/C# was straightforward.

## 6.2 Interface Design

Android provides developers with UI components and controls that enable the development of Graphical User Interfaces for the application. These components and controls are very helpful to developers who want to create GUI’s. The challenge with these tools is they can sometimes lack flexibility, making it difficult to create unique or stylish applications. This has been seen as one of Android’s setbacks in comparison to IOS. They recognised this issue and are now in the middle of tackling it with the latest release of the Android 5.0 SDK in 2014. The new release provides a new style for apps. As discussed in section 5.7 it was important for the student to create a design that was unique, stylish and helped display her vision for the application. The student capitalises on some of the new features released in Android 5.0 as well as some older features and alternative hacks to create a unique and interesting interface. The following sections outline how some of the features were implemented to help create this interface.

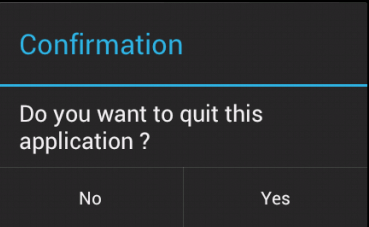
### 6.2.1 Customized Dialogs

Dialogs are small windows which appear and prompt users to make decisions and enter information (Android, no date). In the design sketches of the application it featured a significant amount of dialogs. This is so the user does not have to constantly be going back and forth between different pages to make minor decisions. For example:

The user could be viewing their cookbooks and they want to add a new one:

1. They press the add button on the cookbook screen
2. A dialog appears
3. They fill in the cookbook information
4. They press add and the dialog closes.
5. The new cookbook appears on the shelf.

By having dialogs it makes it easy to have a page (activity) which is centred on cookbooks and have all the functionality relating to cookbooks appearing in dialog boxes on the same page. This is instead of the user going in and out of multiple activities to achieve functionalities relating to cookbooks which can be much more time consuming and confusing for the user. Android comes with a pre-built standard dialog that can be displayed to the user. Although the standard dialog is very different from the style of dialog evident in the design sketches. To achieve a dialog as designed – a rectangle shape with rounded corners was written in code in an xml file which could be understood by Android. The rectangle colours were set to be transparent so users could see somewhat behind the dialog and give the appearance of it appearing over the main page. A basic Android’s dialogs window is then set to a layout containing the rectangle as a background. This helped create a more individual design for the whole application. Figure 17 shows a comparison between the standard Android dialog and the application dialog.



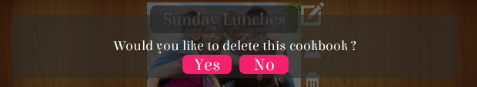


Figure : Comparison between Android dialog and app dialog

### 6.2.2 Custom Listview

“A listview is a view group that displays a list of scrollable items. The list items are automatically inserted to the list using an adapter that pulls content from a source such as an array or database query and converts each item result into a view that's placed into the list.” (Android, no date) In the design sketches a core aspect of the design was to make the cookbooks and recipes appear on a bookshelf to give the idea that these recipes were part of the users own personal collection like their kitchen book shelf at home. To create this effect a custom list view needed to be used because the basic listview only supports a list of strings which isn't sufficient for the application. A custom list view is made possible by a custom array adapter that links a sophisticated layout containing images and text to a listview. The steps involved in creating the bookshelf look:

1. Set up a basic listview
2. Create a layout which will be displayed for each item
   1. Set background of item to a shelf.
   2. Create a transparent textview containing the book or recipe title over an image to make the image appear like a book.
   3. Set icons next to the book to show possible actions you can achieve with the book
3. Create an adapter that adapts individual items to this layout
4. Set the listview to this adapter
5. Display listview

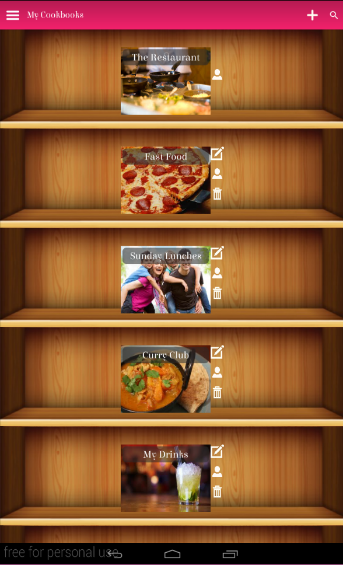


Figure : Cookbook ListView

The custom listview helped bring a unique look to the application that aims to remind users of cooking at home.

### 6.2.3 Action Bar

The action bar is a popular tool in applications as it is a way to support consistent navigation within an application. Action bars make important actions more prominent and accessible in a predictable manner (Android, no date) which is why it was selected to be used in the student’s application. The action bar provides consistency as it stays put throughout the application enabling the user to easily get back to the homepage, search or log out wherever they are in the app. Actions and titles on the bar can also change depending on which activity the user is in. For example when the user is in the cookbook activity, the action add a cookbook is available on the action bar whereas in the recipe view activity the action to share a recipe is available. This gives the application flexibility and consistency.



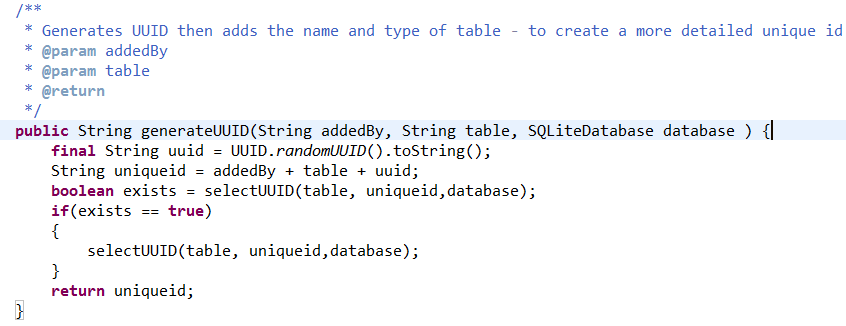
Figure : App Action Bar

### 6.2.4 Challenges of Interface Design

The challenges with creating this interface design are that many of the new features presented in SDK 5.0 have patchy documentation and support, particularly for the action bars, navigation bar and searchview which made it difficult to create these aspects of the design. The Android GUI tools can sometimes be limiting. For example occasionally setting a custom typeface for the components had to be done through a hack. This was because not all the tools supported the use of custom typefaces and the student wanted a consistent interface. Another challenge with creating a unique GUI like this was it was time consuming to create the GUI for a certain tablet screen size and due to this it meant it was too challenging for the student to explore a responsive design for other screen sizes in the time constraints. In the future the student would like to work on the design of the application further to help create a more responsive design for different screen sizes and also work on a better design for the landscape orientation of the tablet.

## 6.3 Database Implementation

The implementation of the database remains essentially the same as the ER diagram with the removal of some tables which were not used and a few changes to the columns. The most important changes in terms of the database implementation were that it was impossible to track rows based on ID because a row of data on the phone would be a different id to the same row of data on the server, something that was not fully recognised till the implementation stage. To overcome this issue a new column was added to the main tables called unique id which would make it easier to track rows between databases. A unique id for a row is made up of a randomly generated UUID, the users email address and the table its being inserted into. The application will always check if the unique id already exists before inserting it into the database, if it does exist it will generate another. This is done to make sure every row has a truly unique id.



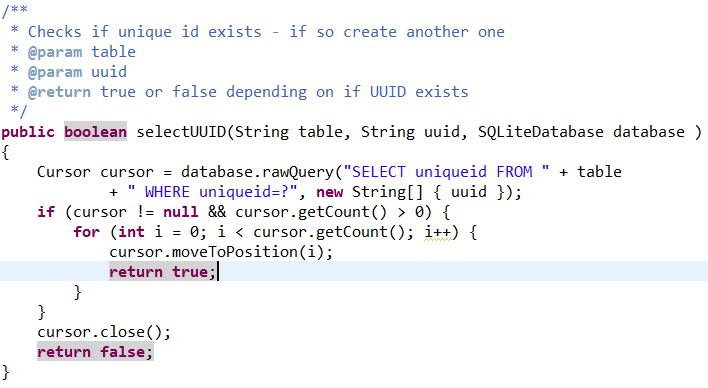


Figure : Unique ID Creation Code

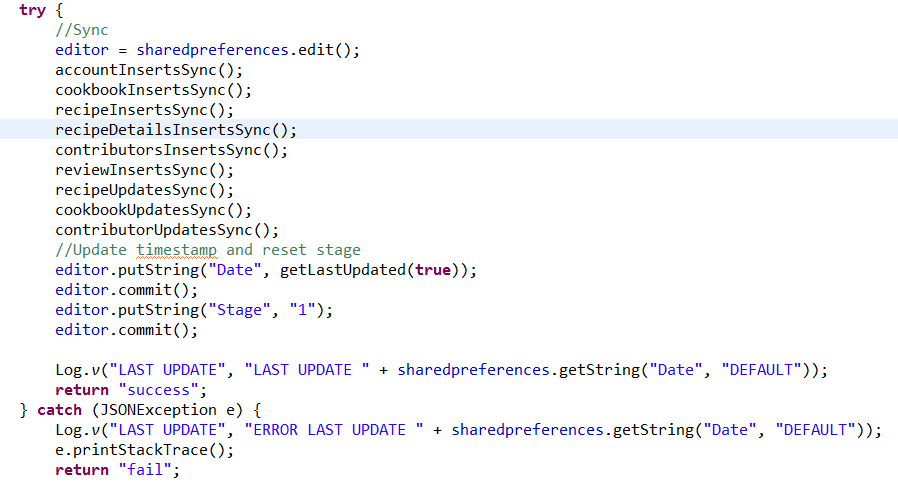
## 6.4 Application & Server Communication

### 6.4.1 Sync Logic

The implementation of the sync code was a challenging aspect of the application. The sync logic was based on differences between timestamps. The sync logic uses shared preferences which is a framework which allows you to save and retrieve persistent key value pairs. The shared preferences are used to store the timestamps for retrieving and deleting data from the server. When the app is installed a shared preference called date is allocated the datetime “2015-01-01 12:00:00” this is because all data on the server has a timestamp greater than this date. Therefore by having this timestamp on the first install allows the user to receive all the data from the server when you first download the app. Below outlines how the sync logic works between the app and the server using the shared preference and timestamps:

* The shared preference Date is installed with a timestamp of “2015-01-01 12:00:00”
* A http post request to a specific script on the server is made to receive all rows from the server database with a timestamp greater than “2015-01-01 12:00:00”. These rows are then inserted or updated in the application database with the timestamp “2015-01-01 12:00:00”.
* A call to the application database is then made for any rows added or updated after “2015-01-01 12:00:00”. These rows are then placed in a json which are sent in a http post request to a script in the server which updates or inserts these rows in the server with the timestamp “2015-01-01 12:00:00”.
* If no errors occurred the shared preference timestamp is then updated to the current time and the same is then done on the next sync with the new timestamp.
* Note: All the rows updated or inserted in the application that haven’t come from the server are inserted or updated with the current timestamp. This makes it easy to then find rows that have been inserted or updated in between syncs.
* A shared preference called stage, tracks the stage the sync got to, so that if the sync fails it will start from the last stage it reached. This means it doesn’t retrieve data that was already successfully retrieved and inserted.

This logic went through several iterations during implementation till reaching that final stage. This was mainly because of the students understanding of the best way to approach the sync because there was little information on the topic. The original approach was too simply use comparisons between timestamps in the database but this often caused duplicates of rows to be added. This approach was then filtered into using a shared preference for the application and server timestamps and then using these preferences with the datetime in the database to retrieve and send rows and again this method occasionally caused duplicates. This approach was then filtered into the one discussed above but a shared preference was stored for every script and whether it was for an insert or for an update therefore this meant there was 8 shared preferences being stored for the sync code which seemed unnecessary. The student then discovered she could simplify it by just storing one shared preference and updating its timestamp once all syncs were completed.



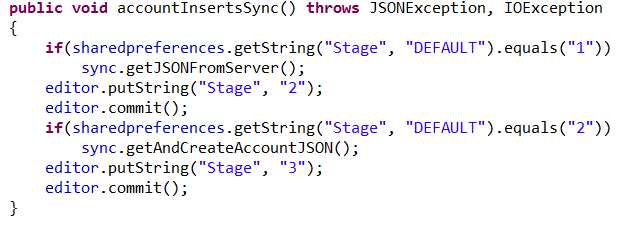


Figure : Sync Code Excerpts

### 6.4.2 JSON

JSON was the chosen format for sending data to and from the server. The JSON is made up of name value pairs. The name states the column the data will be inserted into and the value states the data to be inserted into that column. An example of a cookbook JSON that is being sent to the server can be seen in figure 22. As seen in the figure, the value for an image is a base64 value. This base64 value is then decoded into a byte array when it reaches the script or application which is then ready to be inserted into the database as a blob. The JSON’s can get quite large when sending a lot of rows to and from the server but are still handled well in the application for the amount of data that is currently stored. There is concerns if the size of the data stored in the database was too significantly increase that the built in JSON parser in Java may not be able to handle the parsing fast enough and so another JSON library for java might need to be considered such as Jackson or Google Gson which was not thought about at the design stage. Similarly the ASP.net JSON parser being used can only handle at max a 2GB JSON and so if the JSONs were too get much larger than this another JSON parser or scripting language may need to be considered. This challenge with ASP.net was not recognised till late in the development phase.

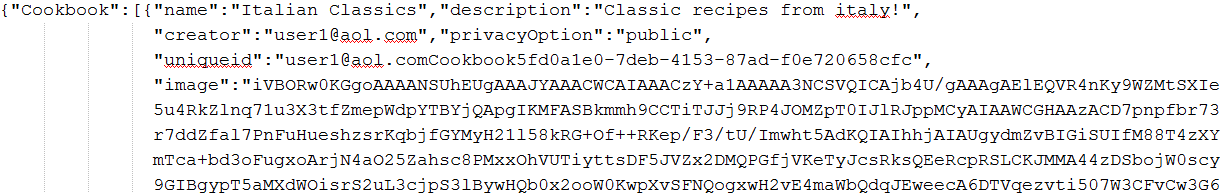


Figure : JSON Example

### 6.4.3 Asynchronous Code

During the implementation it was found that the sync code was taking some time to retrieve and send data to the server. To resolve this issue Asynctask was used. This meant the sync code could be performed in the background with the results of the sync being published to the UI without having to manipulate the UI thread (Android Guru, 2014). After implementing Asynctask for the sync code the application ran much quicker.

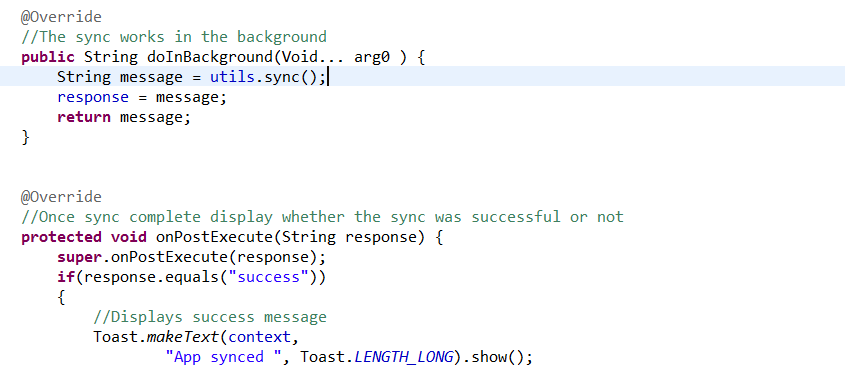


Figure : Async Code Excerpt

### 6.4.4 Managing edits/deletes between syncs

#### 6.4.4.1 Handling Edits

In the majority of the tables in the database there was a column called updateTime which stores the timestamps of when rows were inserted into the database. To handle edits between the two databases a second column was added called changeTime which stored the timestamps of updates in rows. Therefore when a sync occurs, the code checks for inserts by comparing the shared preference date against the updateTime column and checks for updates by comparing date against the changeTime column. This way enables updates between the databases to be handled.

#### 6.4.4.2 Handling deletes

Handling deletes between the server and application is challenging because if the user selects to delete an item in the application and it is instantly deleted, then there is no way of communicating on the next sync that the row has been deleted and therefore should be deleted from the server database as well. To handle this issue a progress column was placed in some of the tables in the databases. So when a row is inserted into the application the progress column for that row is set to “added” and then when a row is set to be deleted instead of deleting the row, it is updated and the progress is marked as “deleted”. This update is then synced with the server on the next sync and the server now has the row marked with deleted. Then hypothetically a server job will remove rows marked as deleted at a certain point each day to clear the database. The student wanted to set up server jobs to do this and wrote the queries which would perform this task but unfortunately the software to do this was not on SQL Server Management Studio and the student did not have permissions to put it on the server. The student sought assistance from the individual who maintains the server but time constraints prevented this issue being resolved.

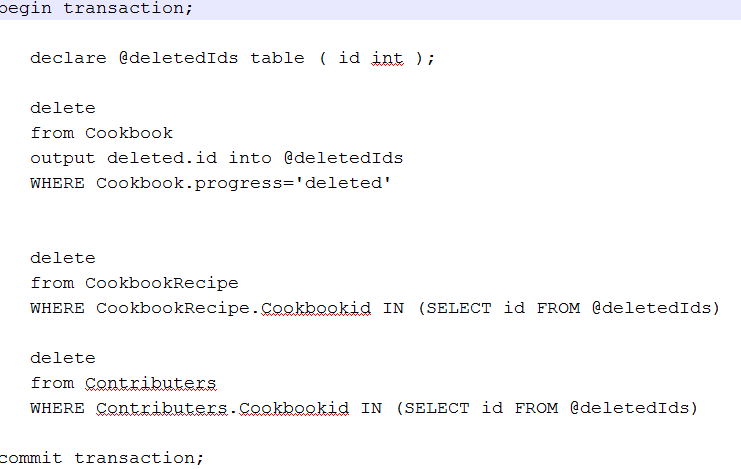


Figure : Example Delete Query

## 6.5 Image Handling

The application handles images which the user will select to represent their cookbooks or recipes. Android limits the amount of memory an application can use. In older devices like the G1 it was 16MB and in newer devices like Droid and Nexus One it is 24MB – 32MB (Dubroy, 2011). Based on past experiences with Android applications the student was aware of how much memory images can take up in an Android application and how easily you can receive out of memory errors if bitmaps are not efficiently handled. Therefore it was important to the student to handle the images correctly.

### 6.5.1 AsyncTask

AsyncTask was used to load images because it enables images to be loaded off the UI thread. It is important to do this as the time it takes to load images is unpredictable and is based on a number of factors such as speed of reading from disk or network, size of image, power of CPU etc (Android, no date). AsyncTask was appropriate for loading one image onto a page and this method was used in activities where only one image is seen. A concurrency issue occurs when multiple images are being loaded into the application using AsyncTask for example in the situation of a listview , so to be able to load the images off the UI thread but handle the issue of concurrency a modified version of Fedor Vlasov image loader code (Vlasov, 2014) was used. The image loader loads the images off the UI thread. This is done through the use of a memory cache. Images are placed in the cache with a generated uniqueid and when they are ready to be loaded into an imageview they are retrieved from the cache. This is much more memory and speed efficient as when the activity is called again all the images are already in the cache and therefore are quicker to retrieve.

### 6.5.2 Other Techniques

To make sure the handling of images was as efficient as possible several techniques were used. When the user selects to load an image into the app, the image will be retrieved and compressed for max quality. This version is then stored in the database. When retrieving the image from the database to display, the best sample size for the required image size is calculated and set for the image so it is as efficient as possible without losing quality. These techniques mean the images still show in the application at a good quality while not taking up to much memory.

## 6.6 Application

### 6.6.1 Application Views

The collaboration feature of the application means there are different views for different users. The following sections describe these views.

#### 6.6.1.1 Creator View

When a user creates a cookbook they are set as the creator. The creator is able to view, edit and delete cookbooks as well as manage contributors. Managing contributors is the ability to view, add and delete contributors in the cookbook. The creator can then go into the cookbook and add, view, edit and delete recipes.

#### 6.6.1.2 Contributor View

When a user is a contributor to a cookbook they can see the cookbook on their shelf. The user can view who the other contributors are but cannot delete or add other contributors. The user is also not allowed to edit or delete the cookbook. The user can view, edit or delete current recipes in the book as well as add new recipes to the cookbook therefore creating a collaborative environment.

### 6.6.2 Application Features

The following sections outline some of the interesting features implemented in the application.

#### 6.6.2.1 Recipe view features

When the user views a recipe several features were put in place to enhance the users cooking experience. The user can select to switch the screen sleep off which then enables the individual to sit the recipe on the kitchen surface and look at the recipe without having to touch the screen with dirty fingers to switch it back on from sleeping. Occasionally users can struggle with the font size or the size of the recipe page in the application, so to help with this issue the recipe view page enables users to easily increase the font size or set the recipe to take up the full screen. When the user has added a recipe it is also easy to share the recipe onto other social networks for friends to see. When the user chooses to share a recipe – an image of the recipe and a message asking their friends to come check it out is sent out on their chosen social network.

#### 6.6.2.2 Cloning & Reviewing A Recipe

Since the application is collaborative and enables users to easily add, edit or delete recipes in a shared cookbook. Occasionally there are situations where users may not want to edit a recipe. When they have only a few changes or too many changes to make and do not want to ruin the original recipe. In the application a clone feature was added, so the users can clone the original recipe under a new name into a selected cookbook which allows them to have the same recipe and make changes without altering the original. In the cases when an individual may want to make a minor change or suggestion to the recipe but might not want to explicitly edit it, a review feature was added to the application which enables the user to review or add suggestions to the recipe for all to see.

#### 6.6.2.3 Searching & exploring for recipes

Many users use recipe applications to find new recipes or help inspire some new cooking creations. Recipes are tagged with information like cuisine, difficulty and dietary requirements so it is easy to find recipes based on these categories. The user can explore for recipes in the application – this feature acts like a filter, it enables the user to select a category such as cuisine and look for only Italian recipes. The explore section is there to help inspire users who are looking for something new but not exactly sure what. While search is used in the application when users are looking for a specific recipe, cookbook or user. The search will find recipes, cookbooks or users containing the search query and return the results.

#### 6.6.2.4 Privacy Settings

Privacy settings for cookbooks enable the user to choose whether they want a cookbook to be public or private. This means they can have a cookbook for only personal use, for use with friends or to share with the world. This option offers great flexibility for the user and for the app as it can be used as a recipe management app, a recipe finder app or as both. If the cookbook is set to public then it can be found through the search but unless they are a contributor or the creator of the book, the user who searched it can only view the recipe and not edit or delete it.

.

## 6.7 Security

Security was an important factor for the student to consider in the implementation of the application. It was important to use regex’s to validate the users email and password when they are creating an account to make sure it is actually a user signing up and not a robot. The regex made sure the length of the password was between 6 – 12 characters and contained 1 digit, 1 special character and at least 1 upper or lower case character. This regex was used to help try and create a complex password that would be difficult for a computer to guess. On top of using a regex the password was then salted and hashed before being inserted into the database. The PBKDF2 algorithm was used to do the hashing – it then applies HMAC to the password with a salt value and then repeats the process many times in this case 1000 times. The amount of iterations makes password cracking more difficult because of the additional computational work and therefore makes the password more secure. All the queries to the database are parametrized to help limit against SQL injection and make the application more secure

## 6.8 Refactoring

It is important in any software development project to create code which is readable and maintainable and one way to ensure this is for developers to refactor their code throughout development. The student was aware of this and at the design stage created a refactoring plan outlining the aims for refactoring the code throughout the project.. The student then referenced this plan throughout the implementation stage to try and create maintainable and readable source code. As well as the refactoring strategy the student also added packages which contained related classes e.g. the view package contains all the classes relating to the user interface. In the res folder packages were not allowed. So to help the student categorise these files and make them easier to understand, the file format category\_filename.extension was used. The student also created two utility classes which contained common methods for code that had been duplicated among several classes. This was done to limit code duplication.

## 6.9 Unit Testing

Unit tests were created throughout the implementation stage to test the functionality of features in the application. Unit tests are useful as they are quick to write, independent and can quickly be run for a sanity check when the developer makes any changes to the code and wants to make sure it doesn’t affect any of the current code. 51 unit tests were written. The majority of the tests were used to test the database functionality. The student didn’t want this to affect the application’s database and so a mock database was created. A mock database is a replica of the application database containing some mock data. This allows for tests to be run without changing the application database. There was little documentation on how to do this for Android Junit tests and so this was quite challenging to implement. The use of writing unit tests throughout the implementation was a great way to test the app functionality quickly. Unit tests are often simple and easy to read, so if the developer finds themselves writing an overly complex or confusing test this can indicate a bad code smell and helps find areas where the code should be simplified and refactored which is a really useful tool to have.

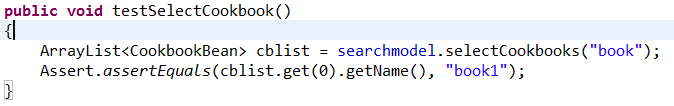


Figure : Unit Test Example

## 6.10 User Testing

It was important to the student to have user involvement throughout the design & implementation stages of the application. One way in which the student made sure to do this was through user testing. The student held two informal user testing sessions with 3 participants at each session. At the sessions the student presented the app to the participants, they played around with the features and the student noted any comments made or any issues or difficulties the participants had. The two sessions were used for different stages of the app. The first session was held on the 22nd of February 2015 when the application had quite a basic interface and the second session was held on the 9th of March 2015 when the application had an improved GUI interface and some additional features. The sessions were useful in gaining important feedback about the application which went towards improving the app throughout the implementation stage. The general consensus from the user testing sessions was they liked the style and ideas of the app. In the first session the participants mainly struggled with the speed on the pages where a sync was also being done and didn’t understand why there was fixed images. In the second session the previous issues were resolved and new issues had arisen such as they couldn’t see the cursor highlight, they didn’t understand the time picker and the images that were being taken via the camera were being displayed sideways. These issues were then resolved for the final product. Common issues in both sessions were that participants didn’t understand what hints were and therefore they needed to be removed, participants were inputting short passwords which reminded the student to implement password security and that participants all struggled with the Android keyboard but unfortunately this was a feature the student couldn’t improve. All the feedback from the sessions played an instrumental part in creating a more stable and usable final product and was incredibly useful in finding errors with the application which the developer missed.

## 6.11 Black Box Tests

Black box tests are a good way to test the application from the user’s point of view. Black box tests help test the function of the application and are a good way to test GUI functionalities (Beal, no date). Throughout the implementation small black box tests were done where the student knew the inputs and outputs that would occur in the application and checked it against what was expected. Once the implementation was complete the student created a formal black box test document for all the main functionalities of the application with the inputs and outputs and tested it to make sure everything was working as it should. This document can be found in the appendix.

## 6.12 White Box Testing (possibly delete)

White box tests, test the internal structure of the application (Webopedia, no date) . The white box tests for the application were not formally documented but were done throughout implementation using log cat to see the data passing between the activities to help understand if the correct data was being sent. This type of testing was essential to helping debug the application.

## 6.13 Usability Tests

The student tried to keep in mind Nielsen’s 10 usability heuristics (Nielsen, 1995) when developing the application to help develop and create a usable application. At the end of implementation the student tested the heuristics against the application. In the appendix is a list of the heuristics, a description of them and if and how the application meets them.

# 7. Evaluation

The application was evaluated throughout the project with a focus group at the design stage and user testing sessions at the implementation phase. When the implementation was completed an evaluation was carried out on the final product. This evaluation was to help the student understand if the product developed was easy to use and could compete with current products on the market. The following sections will outline the process and results of this evaluation as well as discuss the results.

## 7.1 Final Evaluation

The evaluation for the final product aimed to compare a competing product on the market to the student’s application. The procedure for the evaluation was as follows:

1. The participant reads the information sheet and signs the consent form.
2. The participant fills out a demographic questionnaire.
3. The study and task sheets are explained to the participant.
4. The participant is given the first task sheet and is asked to do the following tasks in the first application. The student observes and takes notes.
5. They are given a SUS form to fill out about the application and a break
6. The participant is given the second task sheet and is asked to do the following tasks in the second application. The student observes and takes notes.
7. They are then given a SUS form to fill out about the application and a break.
8. The student then asks a few short questions about each of the applications and asks the participant their preference between the two applications.

For the evaluation half the participants will start with Recipes For Life and half the participants will start with RecetteTek. This is to prevent the learner effect. Each of the applications have slightly varied tasks with the main tasks such as adding, deleting and editing recipes remaining the same. This is because the applications have a few different functionalities and these functionalities need to be evaluated in order to facilitate a fair evaluation for both applications.

For the evaluation the student looked for an application to compare with Recipes For Life. She wanted an application that was for Android and was a close fit to Recipes For Life’s functionalities for a fair evaluation. She looked at applications created by an individual or a small team of developers over one created by a major company like BigOven as the apps from big companies have an overwhelming amount of functionalities that wouldn’t allow for a fair comparison with Recipes For Life. The application for the comparison was RecetteTek. RecetteTek is a simple application that allows the user to create a recipe library and manage their recipes easily. RecetteTek has 4.3 stars overall from 1,108 reviews and has had over 50,000 downloads. This was a straightforward application that had the same level of functionalities as Recipes For Life and has clearly been successful. Comparing Recipes For Life to a successful application available on the app store will help the student to understand if it could compete with products currently on the market.

The evaluation between the applications is done by comparing the average SUS scores for each of the applications and calculating the number of individuals who prefer Recipes For Life against those who prefer RecetteTek. A SUS score is calculated from the System Usability Scale which is a ‘quick and dirty’ reliable tool for measuring usability. It consists of a 10 item questionnaire with five response options from strongly agree to strongly disagree. The system usability scale is very easy to administer, can give reliable results even with small sample sizes and is valid (Usability.gov, no date). For this reason the system usability scale was chosen to evaluate the usability of each app.

10 participants between the ages of 18 and 59 took part in the evaluation. They were recruited mainly through communication about the project. 4 of the participants were involved in the focus group and/or user testing sessions and for the others it was their first time using the application. This created some variety in the results. The participants varied in interest with regard to cooking, interest in recipe websites, apps and technology in general. 3 of the participants were male and 7 were female.

In the original ethics plan for the evaluation the student stated their intention to measure the time taken on each application. However differences in the applications made it unfair to attempt to measure this. Different tasks or functionalities were shorter or longer than other functionalities in the two apps making time comparison unfair.

## 7.2 Summary of Results

As stated in section 7.1 the comparison of the two applications is based on a comparison of SUS scores and a comparison of the participant’s application preferences and comments. A SUS score is calculated by taking the number from the five response option selected for each question (1-strongly disagree to 5-strongly agree). If the question number is odd the student will take one away from the selected response for the question. If the question number is even the student will take the selected response away from 5. The student then adds all the scores for the 10 questions up and then multiplies by 2.5 to get the SUS score out of a 100. Based on 500 studies they found the average SUS score to be above 68 and anything below 68 is below average. The process to understanding a good SUS score is similar to grading on a curve. To achieve an A grade on a SUS score it must receive over 80.3 which is also the point where users are likely to be recommending the product to a friend. If the SUS score is at 68 it would achieve a C grade and anything below 51 is an F putting the product in the bottom 15% (Sauro, 2011).

The results from the evaluation found that the average SUS score for RecetteTek was 52.5 with it receiving scores ranging from (25 – 87.5). The average SUS score for Recipes For Life was 83.25 with it receiving scores ranging from (65-100). Common comments from participants about Recipes For Life was that it was colourful and they found the view of recipes very nice. Many of the participants found the collaboration, cloning and sharing features very good and appreciated a lot could be done with the application. Common comments from participants for RecetteTek was that the users liked the import functionality of recipes from the search engine but found the application to be plain and unattractive and didn’t like how entering a recipe was all free text fields. Overall all the participants preferred Recipes For Life over RecetteTek.

## 7.3 Discussion

The evaluation clearly showed that Recipes For Life could compete with similar applications, it was preferred by all participants when compared to a popular application that has over 50,000 downloads. The evaluation demonstrated that taking a user centred approach throughout the project led to an easy to use product which achieved an A grade in the SUS. The SUS is a valid measurement of usability which has been shown to effectively distinguish between usable and unusable products. The student found it interesting that RecetteTek, a popular application on the app store with strong reviews, in this study received a low usability score. It was just under the criteria for an F grade and received very few positive comments from study participants. Some participants found RecetteTek easier to use because it has less features than Recipes For Life. However they recognised this meant less could be done using RecetteTek. The student is of the opinion that Recipes For Life receiving a high SUS score and being the application of preference by all participants demonstrates that the project has been a success and is worth pursuing in more depth in the future.

# 8. Final Product

The following sections outline the main features of the final product.

## 8.1 Front Screen

The first screen to be displayed to the user handles signing up and signing in. It enables the user to log in or create an account

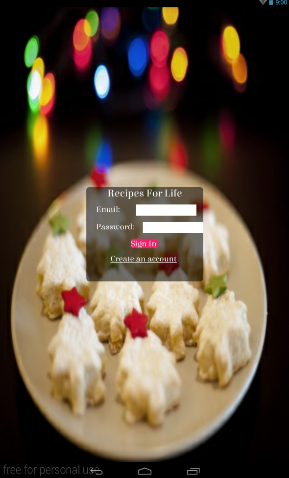


Figure : App Front Screen

## 8.2 Users Cookbooks

When the user logs in the first screen they see is the cookbook shelf. This is a shelf which contains the users own cookbooks and shared cookbooks. From this point the user can add, edit or delete a cookbook as well as manage cookbook contributors. Note: The user can only edit or delete a cookbook if they are the owner of the cookbook.

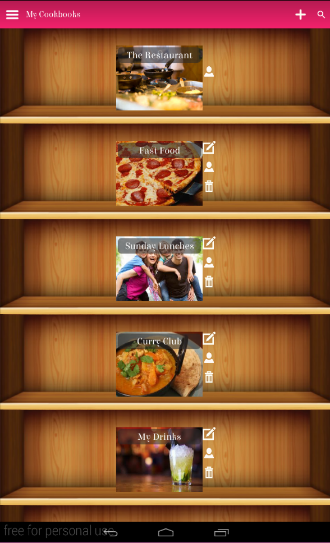


Figure : Users Cookbooks

## 8.3 Users Recipes

When the user wants to view a cookbooks recipe they click on the cookbook and it will take the user to a shelf of recipes contained in the cookbook. From this point the user can choose to add a new recipe or view, edit or delete a current recipe.



Figure : Users Recipes

## 8.4 Viewing A Recipe

To view a recipe the user clicks on the recipe on the shelf. This opens a page which shows the recipe. The user can clone the recipe into their own cookbook if they want a copy. The user can also review the recipe with alterations, tips or feedback to aid collaboration. At the top of the recipe view there is several icons to aid viewing the recipe when cooking.

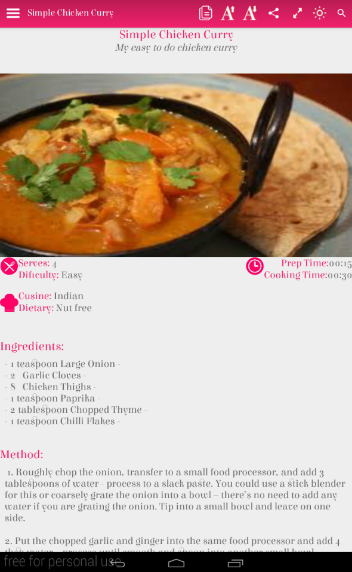


Figure : Recipe View

## 8.5 Editing A Recipe

To edit a recipe the user clicks the pen and paper icon on the recipe shelf this will take the user to a new screen where they can edit different sections of the recipe.



Figure : Recipe Edit View

## 8.6 Searching/Exploring

The user can search for a specific recipe, cookbook or user using the search bar at the top of the application. This takes the user to a page of results containing this query which they can use to navigate to their desired query. When the user does not have a specific idea in mind but wants to be inspired they can explore for recipes by difficulty, cuisine and dietary requirements as well as exploring for public cookbooks. The explore feature is used to help inspire the users cooking ideas.

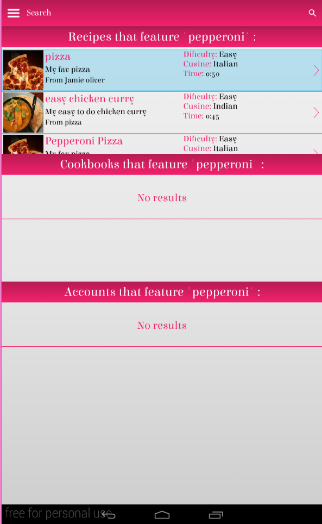


Figure : Search View



Figure : Explore View

# 9. Critical Appraisal

Overall the student views Recipes For Life as a successful project. It was the preferred application of all participants, it received a high SUS score and the requirements essential to achieving the minimum viable product were met. Recipes For Life achieved its core aim of being a recipe management application with collaboration functionality. The participants in the evaluation and in user testing responded positively to the collaboration functionality in the application and this is further evidence the core aim of the app has been achieved.

Although the project overall was considered successful by the student. There were a number of lessons learned, challenges faced and issues dealt with. All of these can be utilised to influence and improve the way projects are developed by the student in the future. The main issue was time constraints. To build the very basics of a recipe application there is a lot of underlying development work to achieve basic adding, deleting, viewing and editing functions for recipes. Achieving the basic functionality combined with trying to develop the code to sync the basic functionality with the server database was very time consuming. A lot of the first iteration was focussed on setting up and getting the basic account and recipe management features created and synced.

The student did not accurately estimate the time required to re-learn Android development, learn how to write the sync code and generally become familiar with all aspects of Android development. As a consequence the student spent a lot more time on the basics than originally planned. There was a concern throughout the project that the student would not have more than the basic features of the application developed by the deadline. Primarily this was due to how time consuming syncing the code was. However by the end of the project the student did achieve the majority of the functionalities originally specified. But from the experience she learnt when working with a new technology or concept, which is central to the application, it is better not to over commit and if feasible to start with a smaller project.

A key challenge within the project was related to project scope. Current recipe applications have a lot of features and society generally has high expectations of what recipe applications can achieve. This was a challenge throughout the project for the student as a lot of her time was focussed on creating the basic functionalities. There were concerns throughout that although a lot of work was put in to the basic features and the collaboration aspect, that if the application didn’t have common features that are in other recipe applications it wouldn’t receive positive responses from users. Especially when compared to other recipe applications or compared to users expectations. This meant re-prioritising of requirements so features like searching/exploring were developed and less focus was spent on developing the collaboration feature. The student would have liked to explore the collaboration feature further to look at a versioning control system for shared edited recipes as well as a way to keep track of the users who were editing or deleting recipes. The student would have also liked to have enabled users to ask to join cookbooks instead of only enabling the owner of the cookbook to add them. Unfortunately due to time constraints and project scope this was not possible.

The student found balancing the development of the application and the sync code to be challenging. Writing the basic sync code for the application was very time consuming and as a consequence there was little time to improve and refactor the sync code. Nor was there time to generalise the code so that it was less specific to the project. Reflecting on the project it was a mistake to not refactor and improve the sync code more and had time allowed this would have been a priority.

It would have been ideal to have achieved an equal gender split in the evaluation but it was not possible to recruit as many male as female participants within the time constraints. Evaluation and user testing involving only those really interested in cooking and recipe apps and websites may have produced more accurate results.

# 10. Conclusion

Overall Recipes For Life proved to be an exciting, challenging and rewarding experience. Throughout the project the student capitalised on existing skills and developed new skills particularly working with new technologies. The student managed to incorporate a wide range of skills developed over her 4 years of study at the University of Dundee to create a stable and user centred product. Project management skills enabled the student to overcome challenges throughout the project. The evaluation demonstrated that Recipes For Life could compete with existing applications on the market. The student is looking forward to exploring this further by placing Recipes For Life on the app store in the future and seeing how it is received. The project was extremely rewarding and represents an invaluable experience which will benefit her for years to come.

# 11. Future

The main feature the student would like to explore further is collaboration. The student recognises that the collaboration on recipes could benefit from a versioning control system where users could track who made changes, see old versions of the recipes and retrieve recipes if accidently deleted. This was a feature the student was very interested in pursuing in the project but simply just didn’t have time to fit it in. Other features that could help create collaboration is allowing users to request to be added into a cookbook, allowing users to add images of recipes into the reviews to show others how the recipe turned out and having a messaging system between users in the shared cookbook.

Work on a more responsive design for the application to make it more suited for small screen smart phones and so that it can be used in landscape mode would be beneficial. As would exploring importing of recipes from search engines and the creation of shopping lists from recipes as features in the app. These features were originally laid out in the requirements and are popular on many recipe apps however were not achievable within this project.

Overall there is significant scope for further development which the student hopes to pursue over time.

# 12. Acknowledgements

The student would like to thank her supervisor Dr Keith Edwards for his continuous support, guidance and feedback throughout the project. She would also like to thank the participants, involved in the study, all of whom played a significant role in the development of a successful final product. Finally the student would like to extend her thanks to her mother for her proof reading skills and support throughout the project. It is also important to acknowledge those who made their images available online which under the creative commons license were available for use within the application.

# 13. References

Agile Alliance, no date. *Task Board.* [Online]   
Available at: http://guide.agilealliance.org/guide/taskboard.html  
[Accessed 5 April 2015].

Agile Development Tools, no date. *Iterative Development.* [Online]   
Available at: http://agile-development-tools.com/wp-content/uploads/2010/10/iterative-development1.png  
[Accessed 5 April 2015].

All Recipes, 2012. *The Measuring Cup : What Families Are Eating Then And Cooking 1997 - 2012 Then And Now.* [Online]   
Available at: http://press.allrecipes.com/wp-content/uploads/AR\_July2012\_MeasuringCup\_Fnl\_HR3.pdf  
[Accessed 20 October 2014].

All Recipes, 2013. *DIgital Food Trends Tablets.* [Online]   
Available at: http://press.allrecipes.com/wp-content/uploads/Allrecipes-May-2013-Measuring-Cup-TabletTrends.pdf  
[Accessed 20 October 2014].

Android Guru, 2014. *What is AsyncTask in Android?.* [Online]   
Available at: http://programmerguru.com/android-tutorial/what-is-asynctask-in-android/  
[Accessed 7 April 2015].

Android, no date. *Action Bar.* [Online]   
Available at: http://developer.android.com/guide/topics/ui/actionbar.html  
[Accessed 15 March 2015].

Android, no date. *Android Studio Overview.* [Online]   
Available at: http://developer.android.com/tools/studio/index.html  
[Accessed 15 March 2015].

Android, no date. *ListView.* [Online]   
Available at: http://developer.android.com/guide/topics/ui/dialogs.html  
[Accessed 15 March 2015].

Android, no date. *ListView.* [Online]   
Available at: http://developer.android.com/guide/topics/ui/layout/listview.html  
[Accessed 15 March 2015].

Android, no date. *Processing Bitmaps Off The UI Thread.* [Online]   
Available at: http://developer.android.com/training/displaying-bitmaps/process-bitmap.html  
[Accessed 22 March 2015].

Android, no date. *Testing Android.* [Online]   
Available at: http://developer.android.com/tools/testing/testing\_android.html  
[Accessed 15 March 2015].

Axure, no date. *Axure Features.* [Online]   
Available at: http://www.axure.com/features  
[Accessed 6 April 2015].

Beal, V., no date. *Black Box Testing.* [Online]   
Available at: http://www.webopedia.com/TERM/B/Black\_Box\_Testing.html  
[Accessed 8 April 2015].

Bittner, K. & Spence, I., 2006. *Managing Iterative Software Development Process.* s.l.:Addison-Wesley Professional.

Bower, A. L., 1997. *Recipes For Reading: Community Cookbooks, Stories, Histories.* s.l.:University Of Massachusetts Press.

Bowes, J., 2014. *Agile concepts: the Scrum Task Board.* [Online]   
Available at: http://manifesto.co.uk/agile-concepts-scrum-task-board/  
[Accessed 05 April 2015].

Budiu, R., no date. *Mobile: Native Apps, Web Apps, and Hybrid Apps.* [Online]   
Available at: http://www.nngroup.com/articles/mobile-native-apps/  
[Accessed 14 September 2014].

Dubroy, P., 2011. *Memory Management For Android Apps.* [Online]   
Available at: http://dubroy.com/memory\_management\_for\_android\_apps.pdf  
[Accessed 22 March 2015].

Ferguson, K., 2012. Intensifying Taste, Intensifying Identity: Collectivity through Community Cookbooks. *Signs,* 37(3), pp. 695-717.

Genymotion, no date. *Genymotion.* [Online]   
Available at: http://www.genymotion.com/  
[Accessed 12 November 2014].

Github, no date. *About Github.* [Online]   
Available at: https://github.com/about  
[Accessed 12 November 2014].

Gorbsky, M., 2013. *Native vs. HTML5 Applications: Which Approach is Best?.* [Online]   
Available at: http://www.seguetech.com/blog/2013/04/09/native-vs-html-applications-best-approach  
[Accessed 11 November 2014].

Highsmith, J. & Cockburn, A., 2001. Agile Software Development: The Business Of Innovation. *Computer,* 34(9), p. 7.

IBM, no date. *IBM Knowledge Centre: Advantages Of Java.* [Online]   
Available at: https://www-01.ibm.com/support/knowledgecenter/ssw\_aix\_61/com.ibm.aix.performance/advantages\_java.htm  
[Accessed 15 March 2015].

IDC, 2014. *Smartphone OS Market Share, Q2 2014.* [Online]   
Available at: http://www.idc.com/prodserv/smartphone-os-market-share.jsp  
[Accessed 11 November 2014].

IDC, 2014. *Smartphone Vendor Market Share, Q2 2014.* [Online]   
Available at: http://www.idc.com/prodserv/smartphone-market-share.jsp  
[Accessed 11 November 2014].

IT Knowledge Portal, no date. *Software Development Methodologies.* [Online]   
Available at: http://www.itinfo.am/eng/software-development-methodologies/  
[Accessed 08 March 2015].

JSON.org, no date. *Introducting JSON.* [Online]   
Available at: http://json.org/  
[Accessed 15 March 2015].

JUnit, no date. *JUnit.* [Online]   
Available at: http://junit.org/  
[Accessed 15 March 2015].

Kotek, B., 2002. *MVC design pattern brings about better organization and code reuse.* [Online]   
Available at: http://www.techrepublic.com/article/mvc-design-pattern-brings-about-better-organization-and-code-reuse/  
[Accessed 13 November 2014].

McCracken, H., 2013. *Who’s Winning, iOS or Android? All the Numbers, All in One Place.* [Online]   
Available at: http://techland.time.com/2013/04/16/ios-vs-android/  
[Accessed 11 November 2014].

Medero, S., 2007. *Paper Prototyping.* [Online]   
Available at: http://alistapart.com/article/paperprototyping  
[Accessed 06 April 2015].

Merriam-Webster, 2014. *Cookbook.* [Online]   
Available at: http://www.merriam-webster.com/dictionary/cookbook  
[Accessed 10 November 2014].

Mitchell, J., 2001. Cookbooks as a social and historical document. A Scottish case study. *Food Service Technology,* 1(1), pp. 13-23.

MongoDB, no date. *NoSql Databases Explained.* [Online]   
Available at: http://www.mongodb.com/nosql-explained  
[Accessed 15 March 2015].

Mountain Goat Software, no date. *Sprint Backlog.* [Online]   
Available at: http://www.mountaingoatsoftware.com/agile/scrum/sprint-backlog  
[Accessed 12 April 2015].

Nielsen, J., 1995. *10 Usability Heuristics For User Interface Design.* [Online]   
Available at: http://www.nngroup.com/articles/ten-usability-heuristics/  
[Accessed 8 April 2015].

Oracle, no date. *The History Of Java Technology.* [Online]   
Available at: http://www.oracle.com/technetwork/java/javase/overview/javahistory-index-198355.html  
[Accessed 15 March 2015].

PHP, no date. *PHP Documentation - Introduction.* [Online]   
Available at: http://php.net/manual/en/intro-whatis.php  
[Accessed 15 March 2015].

Rouse, M., 2007. *Gantt Chart.* [Online]   
Available at: http://searchsoftwarequality.techtarget.com/definition/Gantt-chart  
[Accessed 5 April 2015].

Ruhlman, M., 2012. *So You Want To Write A Cookbook.* [Online]   
Available at: http://ruhlman.com/2012/02/so-you-want-to-write-a-cookbook/  
[Accessed 10 November 2014].

Sauro, J., 2011. *Measuring Usability With The System Usability Scale (SUS).* [Online]   
Available at: http://www.measuringu.com/sus.php  
[Accessed 12 April 2015].

SQLite, no date. *SQLite Features.* [Online]   
Available at: https://sqlite.org/features.html  
[Accessed 15 March 2015].

Technopedia, no date. *Minimum Viable Product.* [Online]   
Available at: http://www.techopedia.com/definition/27809/minimum-viable-product-mvp  
[Accessed 01 March 2015].

TechTarget, no date. *User Story.* [Online]   
Available at: http://searchsoftwarequality.techtarget.com/definition/user-story  
[Accessed 01 March 2015].

Theophano, J., 2003. *Eat My Words : Reading Women's Lives Through The Cookbooks They Wrote.* Ill edition ed. s.l.:Palgrave MacMillan Trade.

Usability.gov, no date. *System Usability Scale.* [Online]   
Available at: http://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html  
[Accessed 8 April 2015].

Vlasov, F., 2014. *GitHub - LazyList.* [Online]   
Available at: https://github.com/thest1/LazyList  
[Accessed 22 March 2015].

W3C, no date. *Asp.net Tutorial.* [Online]   
Available at: http://www.w3schools.com/aspnet/  
[Accessed 15 March 2015].

Walsh, N., no date. *A Technical Introduction To XML.* [Online]   
Available at: http://www.xml.com/pub/a/98/10/guide0.html?page=2#AEN58  
[Accessed 15 March 2015].

Web master Format, 2009. *Review Of Notepad++.* [Online]   
Available at: http://webmasterformat.com/tools/design-and-development/notepad-plus  
[Accessed 6 April 2015].

Webopedia, no date. *White Box Testing.* [Online]   
Available at: http://www.webopedia.com/TERM/W/White\_Box\_Testing.html  
[Accessed 8 April 2015].

# 14. Appendix Outline

1. **Requirements** 
   1. Final Requirements Document
   2. Trello Details
2. **Project Management**

2.1 Initial Project Plan

2.2 Gantt Charts

2.1.1 Initial Gantt Chart

2.1.2 Final Gantt Chart

2.3 Sprint Backlogs

2.4 Supervisor Minutes

2.5 Risk Assessment

2.6 Github Link

1. **UML Diagrams**

3.1 Class Diagram

3.2 Database ER Diagram

3.3 Actual database ER diagram after implementation

1. **Design Sketches**

4.1 Paper prototypes

4.2 Axure Design Sketches

1. **Source Code**

5.1 Application Source Code

5.2 Website Server Script Source Code

5.3 Accessing Central Database

5.4 Delete queries to be used when server jobs is available on the database.

1. **Testing & Refactoring**

6.1 Blackbox Testing

6.2 Accessing Unit Tests

6.2 Refactoring Strategy

6.3 Usability Tests

1. **Ethics**

7.1 Ethics Document

7.2 Ethical Approval Document

7.3 Updated Task Sheet

7.4 Feedback Questions

1. **Study Results**

8.1 Survey Results

8.2 Design Focus Group Results

8.3 User Testing Session Results

8.3.1 User Testing Session 1 Notes

8.3.2 User Testing Session 2 Notes

8.4 Evaluation Results

8.5 Summary of Participants Demographics Details

8.6 Scans of SUS, Consent & Demographic Forms

8.6.1 Scans of Participants SUS Documents

8.6.2 Scans of Participants Consent Forms

8.6.3 Scans of Participants Demographic Forms

1. **User Guide**

9.1 User Guide

9.2 Images Used In User Guide

1. **Misc**

10.1 Licenses for images, fonts etc used in the application

10.2 Screenshots and figures used in report

10.3 Intended Audience Criteria & Personas

1. **Poster**
2. **Demo Video**
3. **Copy of report in pdf**

13.1 Report

13.2 Screenshots and figures used in report

1. **Mid project report**

1. User Story - A user story is a tool used in Agile development to capture a description of a software feature from an end-user perspective. (TechTarget, no date) [↑](#footnote-ref-1)
2. (Agile Development Tools, no date) [↑](#footnote-ref-2)
3. (Budiu, no date)**,** (Gorbsky, 2013)**,** (McCracken, 2013) [↑](#footnote-ref-3)
4. (Budiu, no date)**,** (Gorbsky, 2013)**,** (McCracken, 2013) [↑](#footnote-ref-4)