**Honours Project Draft**

Kari McMahon

*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 people to express themselves and share experiences and traditions with others through the alternative format of cooking. But as we increasingly move more and more into a digital age the concept of the 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 offer 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.*

1. **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 people to express themselves and share experiences and traditions with others through the alternative format of cooking. But as we increasingly move more and more into a digital age the concept of the collaborative cookbooks could soon be lost in the onslaught of digital information if it is not brought into the 21st century. Currently there is no applications on the market that offers 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 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.

1. **Background**

Recipes and cookbooks inextricably link with fond memories. Whether it’s a memory of the recipe of your favourite apple pie you used to bake with your grandma, the delicious recipe for chocolate chip cookies you picked up at the charity bake sale or the recipe which holds the secret to your beloved penne arrabiata. Many of us store these memories away by placing our recipes on pieces of paper in the back of cookbooks, creating our own cookbooks and increasingly placing these recipes on technological devices. By storing these recipes it allows us to keep the recipes to look back on and alter as well as sharing with friends and family to develop new experiences and memories together. Although as we reach a point where many of us are storing and sharing our recipes using technology the possibility of collaborative cookbooks and recipes between groups of people could soon been lost. For generations recipe books have enabled others to add new recipes, alter recipes and view other recipes but as we move into this digital age we appear to be losing this tradition. 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 none of these apps offer the possibility to collaborate on cookbooks and recipes with friends, families, clubs or even with 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 A Historical Document**

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 recipes have played such a major part in society for so many years, cookbooks have often been seen as an informal historical document. 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 untold stories of these women from the cookbooks the author examined as well as getting others to consider cookbooks as worthy objects of serious textual analysis. The book demonstrated how recipe books told the history of that time but also how we could still maintain a connection with these recipe books with the author making a strong point “How deep are the connections between us – a 17th century mother and myself –despite the time that has elapsed.” The book showed we could still connect with the individual even though they lived in a time that was polar opposite to ours by bonding through the medium of cooking and recipes (Theophano, 2003). These papers and books mentioned demonstrated how cookbooks can be seen as a historical document which can tell the story of the time and be beneficial in helping others understand that time period through the analysis of these documents. We can see that cookbooks appear to stand the test of time as we can still connect with the documents that are hundreds of years old. To lose cookbooks would be losing a form of documentation that tell us the stories of society during that time period, that tell us about the varying diets of different cultures around the world and that tells about the food and cooking trends during a specific time period.

**2.2 Traditional Cookbooks And Their Benefits**

As well as cookbooks bringing benefits to historians and researchers by acting as an informal historic document, the success of the cookbooks survival over so many years is the many other benefits they provide in particular the collaborative aspects. For example the paper Intensifying Taste, Intensifying Identity: Collectively Through Cookbooks aimed to look at the negatives of community cookbooks by examining whether community cookbooks 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 by 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 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 and in particular community cookbooks. The author states that cookbooks “tell stories – autobiographical in some case, historical sometimes and perhaps factious or idealized in other instances.” A common theme in the book was the way cookbooks were used to voice stories and experiences as the author mentioned recipes were increasingly becoming “readable with great benefit to our knowledge of women’s experiences and discourses” (Bower, 1997). Although the benefits of community cookbooks can be best summarized by 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 book 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 and so to suddenly lose these benefits to 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 have 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. Therefore background research then turned to focus on cooking and recipe management on technological devices. The research was predominantly focussed on tablets as they were the devices that appeared to be increasingly more in use in the kitchen with AllRecipes.com stating in 2013 that social referrals came from tablet devices were up 787% from 2012 to 2013 as well as an increase in page views on the website from tablets (All Recipes, 2013). The recognition of the increasing amount of popularity for using tablets in the kitchen can be seen by the amount of kitchen accessories on offer for tablets such as tablet kitchen stands, covers to protect from spillages and styluses to use when cooking instead of touching the screen with dirty hands, 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*

The research was focussed on the applications available for tablets. These applications often fall into two categories either recipe discovery or recipe management with occasionally some apps merging the two of the categories. Below is an analysis of a few of the most popular cooking apps on the market at the moment.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Application Name** | **OS Type** | **Description** | **Features** | **Downfalls** |
| All The Cooks | All OS’s | A social cooking application | Enables users to find new recipes, write reviews as well as add new recipes and a few neat features like shopping lists and conversion features | No collaborative feature |
| Big Oven | All OS’s | A strong recipe management and discovery application. With over 8 million downloads and has won several awards. | Import recipes from websites  Take an image of a recipe and convert to a recipe. | No collaborative feature.  You have to pay for some functionalities. |

MORE ANALYSIS HERE.



*Figure 2: Current applications*

Many of these applications are very 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 that recipe apps are highly desired and used by a large amount of the population. The research also enabled us to see the successes of the current apps and what users currently like and don’t like with these apps which could work in addition to the collaborative features of this project.

1. **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. Therefore this project had no client interaction to gain the specifications for 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 had no client interaction to meet the core aim of the application the individual student had to create the specifications for the project based on own ideas, data gathering and research.

* 1. **What Is A Minimum Viable Product?**

A description of a minimum viable product can best be described by technopedia:

*“A minimum viable product (MVP) is the 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, Unknown)

By creating a minimum viable product it enables an individual to gauge the market need for a product by trying to maximise the amount of learning for the minimum amount of engineering hours. A minimum viable product helps the developers create products that are market led and that are desired by consumers by accelerated learning. This project aims to present a minimum viable product that demonstrates a collaborative recipe management application. The challenge in creating the specifications for this mvp is that there is a large amount of recipe applications on the market with a significant amount of features on offer. This already sets up a high expectation for the standard of recipes apps on the market. So the challenge in creating the specification for the product is setting minimum requirements that are do able in the time frame and are primarily focussed on the core aim of the product but also having enough functionality to put the app on a level playing field with current recipe applications.

* 1. **Initial Specification**

The initial specification for the project was developed based on the students own ideas and research from applications on the market. The specification document type selected for the project was of the format suggested by the IEEE requirements guidelines. The choice of using the formalised IEEE requirements over creating user stories[[1]](#footnote-1) was that the specifications were coming from the individual student and not the client therefore in this case formalised requirements seemed more appropriate than user stories which are very customer oriented.

The initial requirements were an informal list of requirements. The requirements were in two categories functional and non-functional and were set out with shall, should and may. Shall meaning the requirements that should definitely be developed. Should meaning the requirements that should be developed if there is time and may meaning the requirements that could be may be developed if there is time. The initial requirements proved as a starting point for understanding the applications functionalities. With a list of possible application requirement functionalities gathered from research, a survey was then created to learn more about the demographics for the application and to help prioritise the functionalities and find any new functionalities based on the target markets needs and desires. An example of these initial requirements can be seen below:

The application shall enable users to create account.

The application shall work offline

*Figure 3: Requirements specification example for functional and non-functional requirements*

* 1. **Data Gathering**

A survey was placed on the website SogoSurvey.com. SogoSurvey was selected over other popular survey sites like Google Forms or SurveyMonkey as it was free to use, gave a selection of different questions types and provided good analysis tools. The survey asked for anyone over the age of 18 who was interested in cooking particularly those interested in the use of technology when cooking to fill out the survey. The survey was distributed via email, on cooking forums and through communication with those who were interested in the project. The survey aimed to understand people’s interests in using technology to cook with and the devices and applications they use to do this to help better understand the market being targeted. As well as present the 17 possible functionalities set out in the initial specifications to be rated in order of usefulness as well as welcome any other functionality suggestions.

The survey received 19 responses with 13 of the participants being female and 6 being male with the participant’s ages ranging from 18 – 78. Several of the participants in the survey were from a cooking background where their occupations were a pastry chef, dietician and private chef. A summary of some of the question responses can be seen below.

|  |  |
| --- | --- |
| **Question Summary** | **Response Summary** |
| Percentage of participants who frequently use recipe books, apps, recipe websites | 63.69% responded to frequently by selecting 4 or 5 in the scale  1 (Not at all) – 5 (All the time) |
| Percentage of participants who use smartphones or tablets in the kitchen | 78.95% stated they used smartphones or tablets in the kitchen |
| Percentage of participants who were interested in using the project being presented | 47.4% responded yes  26.3% responded possibly  21% responded no |

*Figure ?: Summary of some of the survey responses*

The responses from the survey showed the participants gathered had a clear interest in cooking using technology, the application being presented and from a wide range backgrounds. Therefore they seemed the right match to take into consideration their rating of the applications features. These ratings were then used in connection with the initial requirements document to create the final requirements document. Some interesting additional features were also suggested in the survey such as calculate ingredient pricing and calculate nutrition information which weren’t in the initial document. The data gathering was really beneficial to backing up the interest in the product and helping to create a market led requirements document. The full survey results can be found in the appendix.

* 1. **Finalised Specification**

The finalised specification was then created based on the data gathering from the survey, the student’s ideas and research. This was done by a frequency table generated by SoGoSurvey which showed the amount of people who ranked a certain requirement at that ranking e.g. rank 1 is the most useful requirement. This was then considered with the initial requirements document to create the final document. Some of the requirements that were not ranked so highly by the participants ended up high in the final requirements as they were essential to the creation of the application. The requirements in the document that are marked as shall are the requirements essential to creating a minimum viable product. The full specification document can be found in the appendix and an example requirement is seen below in figure ? .

**R5 Cookbook Privacy**

**Description:** The user shall be able to set cookbook to private or public

**Rationale:** This is essential as it gives users the choice of who is displayed to the public

**Risk:** High

**Priority:** High

*Figure 4: Example of finalised requirement*

* 1. **Creating Specification Flexibility And Managing Requirements**

Creating a formalised requirements document as discussed in section 3.4 is often associated as something rigid and fixed. Since the project was an individual’s idea and not a contractual agreement with anyone this enabled some flexibility in the document. To help create flexibility the technique of using a task board which is popular in the Agile methodology was used. The flexibility was created by splitting the formalised requirements document into smaller requirements in a user story style and then storing them in an online task board called Trello in the same priority as listed in the document. The Trello task board allows for flexibility as you can easily move a requirements priority around based on changes in the project and therefore this makes the project more adaptable to change.

The task board was used to manage the project requirements for the rest of the project after the finalised specification was set. This meant requirements for the project priority could easily be moved around based on new information when developing or redesigning aspects of the application. This flexibility helped create a project that was led by information gained throughout the process instead of requirements simply being fixed and based on assumptions at start of the project. An example of a scenario when this was really useful is the requirements for R1 Recipe Management was above the requirement R2 Account Creation and when it came to the development stage it was clear to see that creating an account should be developed over recipe management as a user’s account was often linked with recipe management. Splitting the requirements into smaller more manageable requirements seen in figure 4 was also really useful as it split tasks into easier to manage chunks and enabled more flexibility for example in the time scale of development it appeared that creating a recipe and then creating a cookbook were more important tasks to achieve than deleting a recipe. So having the requirements split up like this on the task board allowed an easy change of priorities. As time went on the project sometimes it was clear requirements needed to be added as that were not initially thought about in the final specification document and so the task board enabled an easy add and removal of requirements based on new information.

**In the document:**

**R2. Account Creation**

**Description:** The user shall be able to create an account.

**Rationale:** This is necessary as it allows the user access to the application.

**Risk:** High

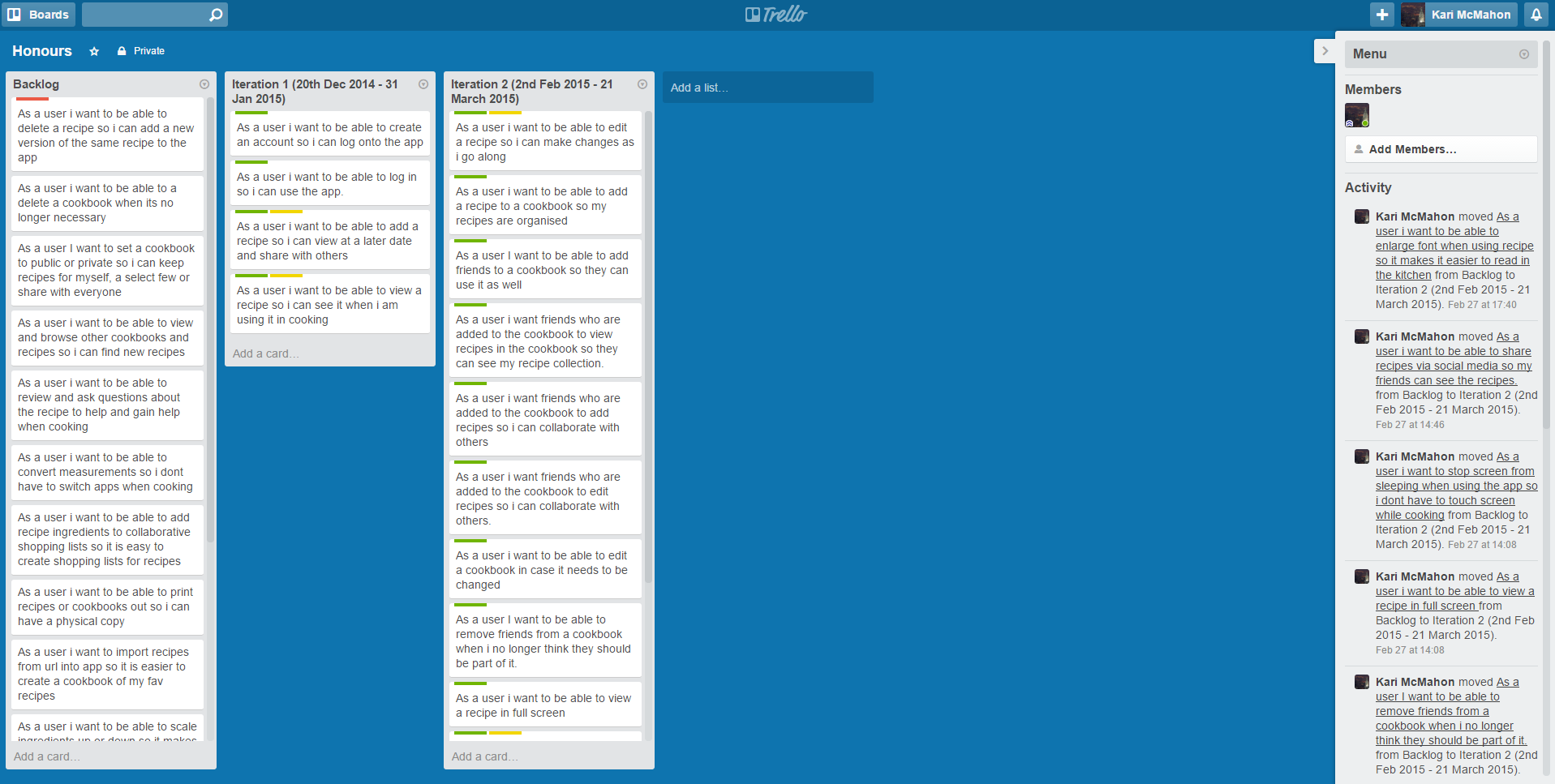
**Priority:** High

**In the task board:**

As a user I want to be able to create an account so I can log onto the app

As a user I want to be able to log in so I can use the app.

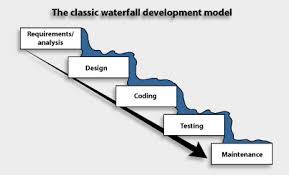
*Figure 4: Contrast of document and task board requirements*



*Figure 5: Trello task board for project*

1. **Project Management**
   1. **Methadology**

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, n.d.). 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, n.d.). Whereas Agile is a set of development processes which are flexible to change, encourages working code over documentation and frequently take 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 a lot 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 was this less client and team oriented. An Agile approach that was suitable for this was the iterative development process. The iterative development process allows you to develop a system in iterative cycles. The process starts with an initial planning stage where the initial requirements and design are set. Then iterations essentially “mini-projects” cycles occur until the product is ready for the 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 the fit the needs of the team, individual or client. The iterative approach is very flexible and helps creates a final product that is user centred and information led (Bittner & Spence, 2006).For the needs of the project the iterative approach was built on to create an iterative methodology strategy appropriate for the project, the diagram in figure ? outlines this strategy.

[[2]](#footnote-2)[[3]](#footnote-3)

[[4]](#footnote-4)

*Figure 6: Images representing waterfall, scrum and iterative software development processes*



*Figure 7: Project Iterative Strategy*

**4.2 Project Deliverables – Necessary ?**

The main deliverables for the project was the recipe application for Android devices, the server side SQL database and the C#/ASP.net code which inserts or retrieves a JSON of database details to sync the phone and server. Several other deliverables were to be handed in for the project:

* Requirements document
* Gantt chart
* User manual
* Proof of testing and evaluation.
* Source code for the application and server side code.
* Poster and presentation
* Log book.
* Supervisor minutes.
* Ethics documents.

As well as any other documents the student used throughout the project and felt was relevant to the hand in. These deliverables are all included in the appendix.

* 1. **Project Management Tools**
     1. **Initial Project Plan, Gantt Chart & Trello**

At the beginning of the project an initial plan was drawn out by the student to help outline the overall project view between September 2014 and May 2015 to the project supervisor. This document was then developed further into a Gantt chart at the start of the project. The Gantt chart enables an individual to help visualise tasks and milestones for the project over the year. The Gantt chart was predominantly used as a project overview which was updated and reviewed every so often to help the student understand whether the project was on track and if changes needed to be made to aid progress. The gantt chart changed over many iterations and figures ? and ? show the difference between the first and second gantt chart.

BEGINNING GANTT CHART FIGURE  
FINAL GANTT CHART

For a lower level view of overall project progress trello was used which was also mentioned in section 3.5 . Trello stored all the requirements for the project and whether they had been achieved, in progress or still to be done. Trello enabled a quick analysis of whether the project was on track just by having a glance at the amount of requirements achieved at that point in time.

**4.3.2 Sprint Backlogs**

A sprint backlog is a list of the tasks and requirements to be completed within the sprint (Layton, 2012). 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. This analysis helped inform later requirement selections for other sprints and commitments for project as the student could see on average how quickly they developed requirements. The sprint backlog is also very flexible because if an individual underestimates what they can do in 2 weeks they can easily add new requirements. Sprint backlogs are really simple and effective way to monitor and analyse development progress by giving a snap shot of day to day progress.

IMAGE OF BACKLOG

* + 1. **Risk Assessment**

For a project to be successful it is essential to outline the possible risks and potential problems within the project at an early stage and create contingencies plans for these risks. This is a common practice within the software development industry. A risk assessment was created at the research and design stage of the project. The risk assessment outlined the risks for the project in order of priority with contingencies for the risks. For the top 3 risks two contingencies are listed and for the rest of the risks one contingency is listed. The prioritisation of the risk is based on the likelihood of the risk occurring times by the loss if the risk did occur. By planning contingences early it allows you to put systems in place to limit the risks for example one risk would be losing the application source code files and so to limit this a contingency would be to have an area where the files are backed up and this area should be used frequently. A risk assessment is a really good practice that can help individuals mitigate risks that could have had a serious impact on the success of the project.

RISK ASSESMENT

* + 1. **Supervisor Meetings And Minutes**

Supervisor meetings were scheduled once a week for the majority of the project. A supervisor meeting is a useful way to help manage the project because you have an outside perspective. Another perspective is useful when the student is struggling with challenges that are halting project progress as a different perspective often gives new suggestions and strategies that hadn’t originally crossed their mind. At each meeting you have a chance to present your work and because of this the supervisor can give advice and guidance on the student’s project or their progress. This is a great way to address any progress or project issues early on. Each meeting gives the student an opportunity to learn from the supervisor’s knowledge and experience which is a great tool to have. Minutes were kept to enable reflection on these meetings and can be found in the appendix.

* + 1. **Log Book**

A log book was maintained 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.

**4.3.6 Github**

For the project it was necessary to have an area to manage all the files for the project incase anything went wrong and files needed to be recovered - Github was selected for this. Github is a versioning control tool.It enables access to files wherever there is an internet connection, the ability to revert to old versions and the ability to store a range of files whether it’s code or a word document. The student already had experience with Github from past projects and has a private account on the site. Github is also well supported with over 6 million people using the site (Github, n.d.). Therefore there is a lot of support available if any difficulties were to occur during the project. For these reasons Github seemed the appropriate choice for file management. Although github is not just a file management tool, it is also a great tool for managing projects. The daily commit messages enable an informal log of day to day work and visualisations are produced on Github based on commits outline the students’ progress. This can be very useful for understanding project progress and understanding an individual’s work load and work amount. The figures below show some of the visualisations that are produced.

* + 1. **Realities Of The Methadology ?**

1. **Design**

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

**5.1 Operating System Selection**

The application for the project is being developed predominantly for tablets as they are increasingly becoming more popular for use in the kitchen. This meant a tablet operating system needed to be selected before development. In terms of tablet operating systems there are three main options you can develop for which 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, n.d.), (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 previous experience and research. These advantages and disadvantages are outlined in the figures ? and ? below:

**Figure 1: Advantages of Android, IOS and Native Web Apps[[5]](#footnote-5)**

|  |  |  |
| --- | --- | --- |
| **Android** | **IOS** | **Hybrid / Web Apps** |
| Already experienced in Android development | Large amount of resources available for help | Works on both operating experience |
| Not as many recipe applications available on android hence more market share for the application | Popular operating system | Has some experience of html/css with minor javascript experience |
| Large amount of resources available for help | Standardized marketplace | Save time as they port to multiple platforms |
| Standardized market place | Easier to build nicer UX/UI features | Merge web and native features |
| Easier to build nicer UX/UI features | Make use of own hardware and software features | Consistency between apps |
| Make use of own hardware and software features |  |  |

**Figure 2: Disadvantages of Android, IOS and Native Web Apps1**

|  |  |  |
| --- | --- | --- |
| **Android** | **IOS** | **Hybrid / Web Apps** |
| Only covers one OS | Only covers one OS | Little help available, it’s relatively new in comparison to Android or IOS |
|  | No experience with ios or objective C development | Complex to set up and fidgety |
|  | Need to own a mac to develop so would only be able to work from computing building | No centralized market place. |
|  |  | Often work arounds are needed when porting to different apps |
|  |  | Web apps cannot work offline |

Android was the leading operating system in market share in smartphones in last year as well as having the largest amount of advantages with the least amount of disadvantages based on the research so Android was a natural choice to develop for. Although hybrid/web apps were also a strong choice but the main reason for not selecting this option was the lack of support available at the moment which was a concern especially for a large project like this. The device selected for testing and presenting the application is a Samsung Galaxy Tab S4 as Samsung is the largest Android vendor according to IDC.com (IDC, 2014), so it felt best to test on a Samsung device.

**5.2 Server Side Design**

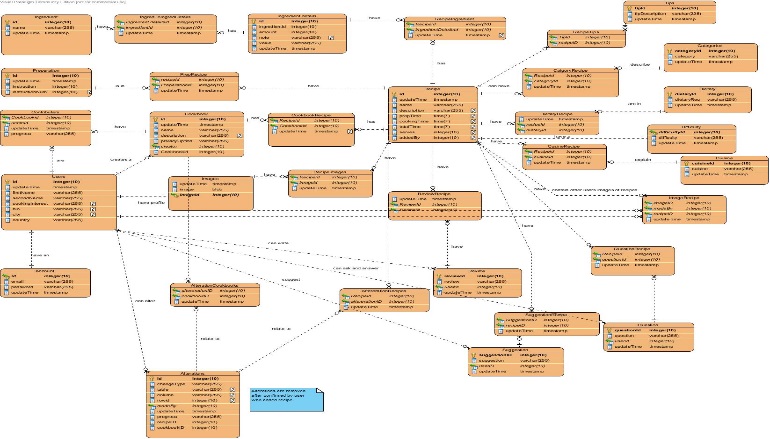
**5.2.1 Database Requirements & Selection**

Database consideration was a large section of the design stage of the project as the application will be handling a lot of data from multiple users. Android devices have a SQLite database built into the phone which enables individuals to retrieve data from the database without having to be connected to the internet this means a responsive application as there is no wait times trying to retrieve data from a server and it can also work offline. The application needed to be able to work offline so users could access the recipes where wifi might not be available. The application also be able to retrieve data from other individuals who may be contributing to your cookbooks therefore this meant a centralized server would be needed to store all the individuals who are using the application’s data. This meant having the database on the phone and a database on a server with a sync functionality between them.

SQLite is a lightweight, self-contained database that is memory efficient and can handle terabyte-size databases this makes it a suitable database to be embedded on smartphone devices (SQLite, n.d.). It is a full SQL implementation making it easy to use SQLite if an individual already has experience with SQL. At first it appeared that SQLite was the only database option to be used on the phone as it is embedded into every Android device but in fact Couchbase offers a NoSQL database solution suitable for the Android device. NoSql databases are databases which are schema free and can handle large volumes of data which is appropriate in the age of big data. NoSql databases are suitable for scaling and are easy to replicate meaning high availability and strong disaster recovery (MongoDB, n.d.). Couchbase was considered for the project as the application may hold a large amount of data in the future and may need a database which is scalable and Couchbase also offers a sync functionality between the phone and server. Although Couchbase’s phone database was relatively was relatively new and lacked support and documentation in comparison to SQLite and as the database design was explored further the database was hard to visualise as a NoSql database therefore it seemed more appropriate to use SQLite. With a SQLite database on the phone, a SQL or SQLite database had to be used on the server. SQL was chosen on the server as SQL tends to have more robust database management tools than SQLite and SQL databases are often more supported for querying from server side scripts such as a C# or PHP script. Although either choice would have been equally suitable, it was mainly due to preference.

**5.2.2 Database Design**

After the database was selected the tables and columns for the database were designed. The database design went through quite a few iterations as the application design developed. The recipe table is connected to many tables like preparation and ingredients which involve linking tables as recipes can often have more than one preparation step or ingredient. This type of design created quite a significant amount of tables. The design of the database involved all possible tables needed for the application but not all tables were used in the application based on what could be implemented in the time frame. All tables needed a timestamp as this would be used to track differences in date between the application and the server which would be essential to syncing changes for the database. This database design is used for both the SQL and SQLite database.



**Figure 3: Database Design**

**5.2.3 Language Selection & Development Tools**

**5.2.3.1 Language Selection**

On the server side it was necessary to have server side scripts that would be called by the application to insert and retrieve data from the 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, n.d.) and ASP.net is a “development framework for building web sites with html, css, javascript and server side scripting” (W3C, n.d.). 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 Android syncing in the past and found it straightforward so therefore felt it was suitable to use again.

**5.2.3.2 Development Tools**

The development tools to be used for managing the server side application was Microsoft SQL Server Management Studio as the database was a MSSQL database which was provided by the School Of Computing for use in the project. Therefore Microsoft SQL Server Management Studio was chosen as the tool to manage the database as it is recommended to use with this database, it is free for students, 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 its lightweight and simple tool which has a lot of code editing features built in that seemed suitable for managing the PHP scripts.

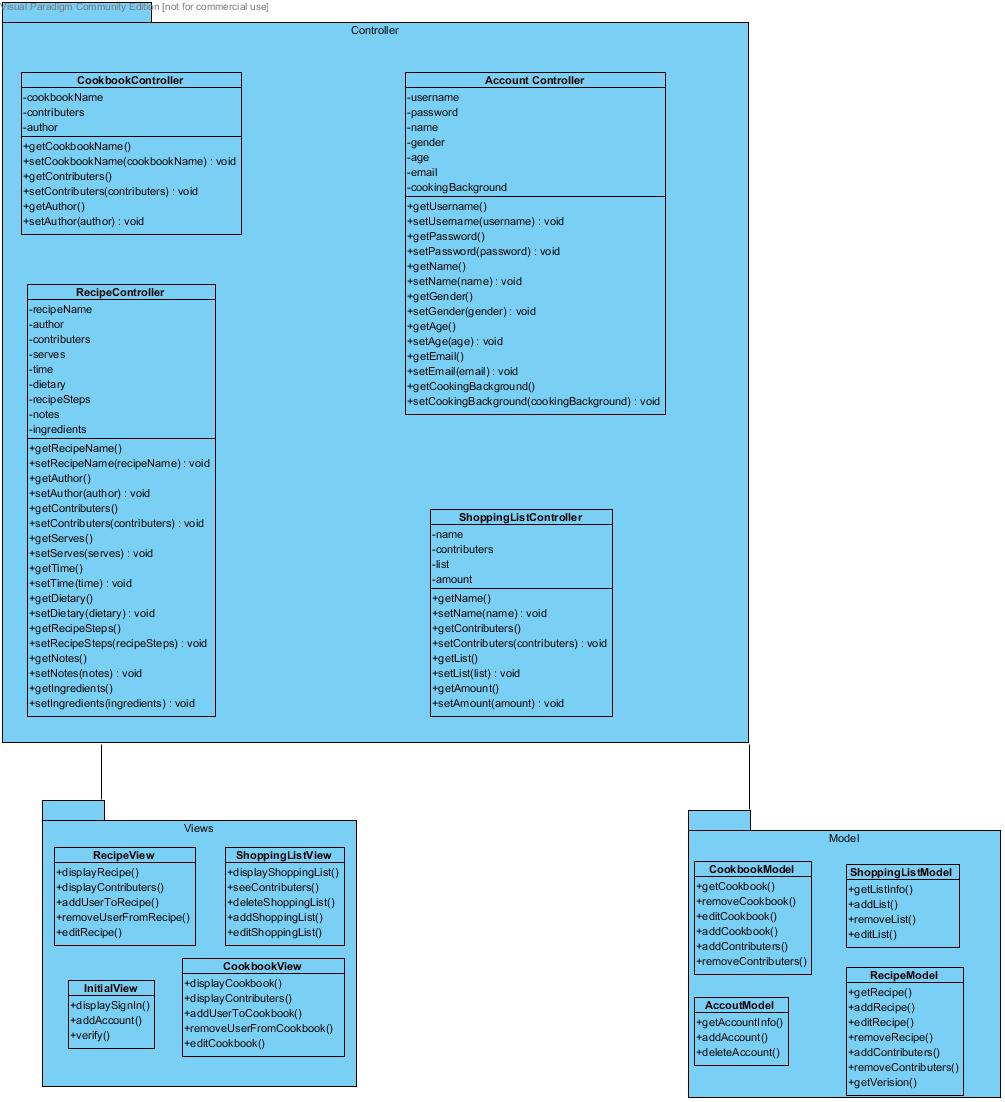
**5.3 Application Design**

**5.3.1 Application Layout & Design Patterns**

The application will be designed with a Model-View-Controller pattern as the application is heavily database focussed 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 ? demonstrates the model view controller design pattern.



The application will have a model package which contains classes that purely send and retrieve data from the database. The controller will be information beans which will store information that is being sent too or retrieved from the database. The view will have all the classes (activities) which the user will interact with and retrieve the data from user input which will be stored in the controller. The model view controller pattern creates a strong separation of business logic and view. This pattern is demonstrated in the UML class diagram created for the application which is in figure ? . This class diagram represents the classes that will be thought to be used during the implementation at the design stage.



**5.3.2 Language Selection & Development Tools**

Java is an object oriented programming language which is platform ubiquitous and has a vast array of 3rd libraries available for use (IBM, n.d.). Java has been around since 1995 and therefore a significant amount of developers work with java and a huge amount of documentation is available (Oracle, n.d.). Google selected Java as the language to be used to develop Android applications and therefore Java is the language that must be 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, n.d.). Junit is a unit testing framework for the Java Programming Language (JUnit, n.d.), its links with the android test suites makes it easy to test and write tests for Android applications.

There are two IDE’s 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 is still in its beta stages at the application design stage (Android, n.d.). Eclipse is the original Android IDE of choice, there is a lot of support for issues in eclipse and it is stable but 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 support and stability. The student also had Eclipse set up and was experienced using it, so this also affected the choice of the IDE.

In Android you can run the application virtually using an emulator. The emulator selection for the project is Genymotion. Genymotion is an android emulator which is trusted by 1500000 developers (Genymotion, n.d.). This is the alternative in comparison to the Android emulators provided and from past experience of using both there is a significant difference in speed with Genymotion being a lot faster.

**5.4 Application & Server Communication**

As already discussed the application should be able work online and offline. Therefore a sync functionality is needed 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 ? .



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 timestamps on the server and phone are used to find rows that need to be sent or retrieved between the database and the device when internet is available. 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 data exchange format 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 header or paragraph (Walsh, n.d.). JSON is a lightweight data interchange format that easy for humans but is also easy for machines to parse and generate (JSON.org, n.d.). The choice selected was JSON as it is readable and in both PHP and Java is really simple to create and parse JSON code therefore it seemed more straightforward to use JSON than XML.

**5.5 The Collaboration Feature – Needs to be mentioned how it will work !!!**

**5.5 Refactoring & Testing Strategy**

NEEDS TO BE WRITTEN

**5.6 Paper Prototypes**

Paper prototypes are quick and easy way to throw down ideas. They are simple to create and easy to throw away which is great in the early stages of a project as you can easily visualise your application without spending too much time on the finer details. To help draw out requirements and application design several iterations of paper prototypes were drawn. Examples can be seen in the figure ? below:

IMAGE OF PAPER PROTOTYPES.

**5.7 Design Sketches**

Once a finalised design was in place on paper, Axure was used to create a more detailed design prototype. Axure helped turn the sketch into a realistic looking application. In Axure it’s easy to work with different

fonts, images and button styles which help make it easy to make design decisions for the application. Visualising the application in this way caused changes in design and pulled out additional ideas and possible challenges with some areas of the design. Axure was really useful because it is easy load up the design and make minor changes based on feedback or application requirement changes which is essential in creating a user centred application.

**Theory behind shelf design and stuff.**

****

**Figure 4: Design Sketches**

**5.8 Ethics**

The project/application is user centred so it necessary to be able to design and implement the application based on user feedback and to do this an ethics form has to be submitted. An 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 were 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 to gain opinions on design sketches and the application idea, user testing to understand how usable and easy to understand the application and an evaluation against a popular recipe app to see if it could compete on the market. Ethics enables you to undertaken user studies in the correct manner. Ethics for the project was approved in October 2014 this meant the project could have users involved throughout which would aid the development of a user centred application.

**5.9 Intended Audience & Persona’s**

The intended audience for the application is any individual with an interest in cooking and particularly those with an interest in using technology with cooking. The participant group aims for the studies involving users were individuals from the ages of 18-60 who have an interest in cooking. It was hoped the participant group would have an equal gender split. At the design stage very basic personas were created to represent the different demographic groups the application could have to help others understand the intended audience of the application

**5.9 Design Focus Group**

A focus group was undertaken at the design stage of the application to help create a user centred application. The focus group had 6 participants, 4 female and 2 male between the ages 18 and 60. The focus group was very informal, the design sketches created on Axure and the project idea was 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 such as areas where the pink font was used as headers was sometimes hard to read and issues with some of the icons such as the ones to change font size. This feedback was then taken into consideration in the implementation stage

1. **Implementation & Testing**

**6.1 Changes From The Design**

At the start of the implementation phase there was a minor issue with PHP on the School Of Computing Zeno server not being setup to retrieve and insert data from Microsoft SQL Server databases. This issue was recognised during the holiday period where the staff who maintains the server was on holiday. Instead of waiting for this to be fixed and halting development time, the student choose to write server side scripts using ASP.net and C# as it was setup to access Microsoft SQL Server database. There was a small learning curve with this choice as it had been a while since the student had used ASP.net and C# which meant it took some time to write the initial scripts for syncing but once this hurdle was achieved using ASP.net/C# was straightforward.

**6.2 Interface Design**

Android provides developers with UI components and controls to allow developers to build a graphical user interface for the application. These tools are very helpful for helping developers create GUI’s but are often very limiting in design which has often been seen as one Android’s setbacks in comparison to IOS. It has often been harder to create unique and stylish interfaces in Android which has been recognised by Android and they are currently in the middle of tackling with the latest release of Android 5.0 SDK .in October 2014. Some of the features that are part of the Android 5.0 SDK have been capitalised on in the project to help create a unique and interesting interface similar to the design sketches as well as older features and alternate hacks. Below outlines some of the features that helps create this feature:

**6.2.1 Customised dialogs**

Dialogs are small windows which appear and prompt users to make decisions and enter information (Android, n.d.). In the design sketches of the application it is very dialog heavy so the user does not have to constantly be going back and forth between different pages to make small 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.

The task described meant the user could stay on the one page to achieve a task instead of flicking between pages which is a lot more time-consuming. Android comes with a pre-built standard dialog that can be displayed to the user but the standard dialog is very different from the way it was visualised in the design sketches. So 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.

Standard dialog image vs Custom app dialog image

**6.2.2 Custom Typography, Icons & Buttons**

Part of the applications uniqueness comes from the typography used which is a free for use typography called Elsie which is listed in a document stating all the image and font rights in the application in the appendix. To use this font in the application, a custom typeface needed to be used below is a code sample of how this was done in the application:

CUSTOM TYPEFACE EXAMPLE

Buttons for the application were also customised to create the unique style of the application by creating rounded corners and setting the custom typeface to the button text. Icons were used throughout the application to represent buttons. Many popular applications often use icons to simplify interfaces such as plus sign to represent add and pen and paper symbol to represent which can be seen in the application. This helps create a modern style in the application and keep it in tone with the design standards of modern apps.

CUSTOM BUTTONS & ICONS

**6.2.3 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, n.d.) 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 custom collection like on their kitchen book shelf at home. To create this effect a custom list view needed to be used as a basic listview only supports a list of strings; 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 like textview over an image to make it appear booklike
   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

DISPLAY IMAGE OF LIST

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 the app often through navigation bars. Action bars make important actions more prominent and accessible in a predictable manner (Android, n.d.). The action bar provides consistency as a navigation bar stays put throughout the application enabling the user to easily get back to the homepage or log out of the application. The search bar also remains consistent enabling users to search for recipes, cookbooks or users wherever they are in the app. The actions on the action bar and titles change depending on where you are in the app. On the cookbook page there is an action to add a cookbook whereas on the recipe view page there is an action to share a recipe. This gives the application flexibility and consistency

IMAGE OF ACTION BAR

**6.2.4 Challenges**

The challenges with creating this interface design is many of the new features presented in SDK 5.0 have patchy documentation and support particularly for the action bars – navigation bar and searchview making it difficult to create these aspects of the design. The Android GUI tools can sometimes be limiting for example setting a custom typeface for a title in the action bar has to be done through a hack because for some reason Android didn’t enable this on the action bar but did in the activities and navigation bar. Another challenge with creating a GUI like this is it is very time consuming and trying to create this GUI and a responsive design was too challenging within the time and in the future it would be good to explore ways to make the layout more responsive to different device sizes.

* 1. **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 was 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 wasn’t fully recognised till implementation. 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.

**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 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 you 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 has a timestamp installed of “2015-01-01 12:00:00”
* A call to a specific script on the server is made to receive all rows from the server database with 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 and sent 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.

This logic went through several iterations during implementation till reaching that final stage. This was mainly because of understanding the best way to approach the sync was confusing. 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 shared preference for the application and a shared reference for the server 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 and the student discovered she could simplify it by just storing one shared preference and updating it’s timestamp once all syncs were completed.

MAYBE DISCUSS MORE ? FIGURES SHOWING CODE AND OLD WAYS ?

**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 to data will be inserted 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 ? . In the JSON at the name value pair for image a base64 value is used to represent the byte array of the image. 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 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 libraries for java may 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.

EXAMPLE JSON

**6.4.3 Asynchronous Code**

During the implementation it was found that the sync code was taking some time to retrieve and send data from the server. To resolve this issue asynctask was used so that the sync code could be performed in the background and then the result of the sync could be published on UI thread. AsyncTask is useful as it enables code to be performed in the background and results to be shown in UI without having to manipulate. After implementing asynctask for my sync code the application ran much quicker.

**6.4.4 Managing edits/deletes between syncs**

**6.4.4.1 Handling edits**

To handle edits between the two databases a second timestamp was added to the rows in the database. In the majority of the tables in the database there was a column called updateTime which stored the insert timestamps of rows and there was a column called changeTime which stored the timestamps of updates in rows. Therefore the inserted rows are being synced the shared preference date is compared to the updateTime column and when the updated rows are being synced the shared preferences date is compared to the changeTime column. This way enables updates between the databases to be handled.

**6.4.4.2 Handling deletes**

Handling deletes between a server database and an application database is challenging because if the user selects to delete an item in the application and the application then instantly deletes this row from the application database then there is no way of communicating on the next sync that the row has been deleted and therefore should also be deleted from the server database. To handle this issue a progress column was placed in some of the tables of the database. When a row is inserted into the application the progress 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 then marked as “deleted”. This update is then synced with the server on the next sync and the server now has the row marked. Then hypothetically a server job will remove rows marked as delete at a certain point each day to clear the database.

SERVER JOB ?

* 1. **Image Handling**

The application handles images which the user will select for their cookbook or recipe front covers and images to represent their recipes and these images are then stored in the database. 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 run in to out of memory errors if bitmaps are not efficiently handled therefore it was important to handle the images correctly.

* + 1. **AsyncTask**

AsyncTask was used to load images because it enables the application to load images off the UI thread. It is important to load images off the UI thread as the time it takes to do this 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, n.d.). AsyncTask was appropriate for loading one image onto a page and this method was used in the recipe view and edit recipe pages where only one image is seen. A concurrency issue occurs when multiple images are being loaded into the application for example in the situation of a listview and so to be able to do load the images off the UI thread but handle the issue of a 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 listview and therefore are quicker to retrieve.

* + 1. **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 app retrieves the image and then compresses the image for max quality and this version of the image is stored in the database. When retrieving the image from the database to display the best sample size for the required image size is then calculated and set for the image so that the image is as efficient as possible without losing quality. These techniques mean the images still show in the application at a good quality while still not taking up to much memory.

Memory Size Compare To Other Apps

* 1. **Application**
     1. **Application Views**

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

**6.6.1.1 Creator View**

When a user creates a cookbook they are set as the creator. So the user logs on and views the cookbook they can choose to delete the cookbook, edit the cookbook and manage contributors. Managing contributors means viewing current contributors, adding contributors who can access the cookbook or deleting contributors that the user no longer wants to be able to access the cookbook. The user can then go into the cookbook and view, edit and delete recipes inside their cookbook. If the user sets the cookbook to private then it is only accessible to them and the set contributors. If the cookbook is set to public then it can be found through the search but unless they are contributor or creator of the book the user who searched can only view the recipe and not edit or delete it.

**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.

* + 1. **Application Features**

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

* + - 1. **Recipe features**

When the user views a recipe some features were put in place to enhance the users cooking experience. The user can choose to switch screen sleep off which then enables the individual to sit the recipe on the kitchen surface and just look at the recipe in the application without having to touch the screen with dirty fingers to switch it back on from sleeping. Sometimes users can struggle with the font size or the size of page in applications so in the recipe view page the application enables users to easily increase the font size or set the recipe to take up the full size of the screen so it is easy to see in the kitchen. When the user has added a recipe it is also easy to share the recipe onto other social networks for friends to see from the recipe view. When the user chooses to share a recipe the image of the recipe and a message asking their friends to come check out the recipe in the app is sent out for friends to see on their chosen social network. The app also enables the user to upload images straight from the device so as soon as the user is finished cooking the recipe, they can add the recipe with a photo taken on the device and then share that photo out to their friends.

**6.6.2.2 Cloning and reviewing a recipe**

Since the application is collaborative this means users can easily add new recipes to a shared cookbook or edit or delete recipes in the shared cookbook. Although sometimes there is a situation where the user may not want to edit the recipe as they have a lot of changes to make and they don’t want to ruin the original recipe. So they can clone the recipe under a new name into the same cookbook where they have the same recipe but they can make changes to it without having to ruin the original recipe. In other cases an individual may want to make a minor change or suggestion to the recipe but might not want to explicitly edit the recipe. For this scenario the user can view the recipe and at the bottom of the recipe view insert a review or suggestion for the recipe for all too see.

**6.6.2.3 Searching and 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 is like a filter it enables the user to select a category such as cuisine and look for only Italian recipes. The explore section is their to help inspire users who are looking for something new but not exactly sure what. Whereas search is used in the application for 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 if they want 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 the application can be used as a recipe management, a recipe finder app or as both.

* 1. **Error Handing**

Errors can often occur in applications if not properly handled so it was important to the student to put in place as much error handling as possible to make the application as stable as possible. Within the application input type boxes such as number boxes for fields which should only contain numbers are used to make sure the appropriate values are being placed in the appropriate boxes. There is several error checks in the application checking that input boxes which should not be left empty by not enabling the user to proceed to the next dialog until the value is filled. Confirmation dialogs are put in place for when the user is using the application to confirm what they are doing before they make a big change such as deleting a recipe. Many try catches are used to catch possible errors such as sql exceptions or json exceptions that could occur within the application. Transactions are used to enable all related rows or no rows are inserted into the database. Timeouts are in place in connections to the server scripts to make sure the application is not left requesting a webpage for too long perhaps in the situation where the server is down for maintenance.

* 1. **Security**

Security was an important factor to consider in the implementation of the application. It was important to use regex’s to validate the users email and password were valid when creating they are creating account to make sure it is actually a user signing up to the application and not a robot. The regex for the password 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 – the algorithm 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 making the password more secure. All the queries to the databases use parametrized queries to help limit against SQL injection and make the application more secure

* 1. **Refactoring**

In the design stage a refactoring plan was laid out with the following aims – capitals for classes, comments explaning class and method, method starts with lowercase, comments to explain any tricky parts, remove any code duplication, classes to big split up.

To keep the code maintainable and readable several other strategies were put place. An additional package was created to store which held utility classes which didn’t specifically fit model, view or controller packages. These were often utility classes that aided some function in the view or model. In the res folder such as the layout folder – folders and packages are not allowed. To help categorise these files and make them easier to understand the file format category\_filename.extension was used. Two general utility classes were used to store methods which were being duplicated among several classes to help limit code duplication.

* 1. **Unit Testing**

Unit tests were created throughout the implementation to test the functionality of features of 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. Number of tests written. The majority of the tests tested the database functionality and too make sure this didn’t affect the application’s database – 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 affecting any of the functionality of the app. There was little clear documentation on how to implement a mock database 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.

Example Unit Test

* 1. **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 with 3 participants at each of the user testing sessions. 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. Two sessions were used for different stages of the app. The first session was held on the 22nd 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 final product as well as feedback from the first session going towards improving the application for the second session. 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 a fixed image. In the second session these issues were resolved from the first session and the issues participants had in the second session was they couldn’t see the cursor highlight, they didn’t understand the time picker and 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.

* 1. **Black Box Tests**

Black box tests are a good way to test the application from the point of the view of the user. Black box tests help test the function of the application and are a good way to test GUI functionalities. 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 as 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.

Doc:

Add A Cookbook

Add A Recipe

Edit A Cookbook

Edit A Recipe

Delete A Cookbook

Delete A Recipe

Add A Contributor

Delete A Contributor

Increase/decrease font

Screen sleep

Share to twitter

Full screen

Clone a recipe

Create an account

Log off

* 1. **White Box Tests**

White box tests are tests which test the internal structure of the application It tests how data passes between the application. The white box tests were not formally documented but were done throughout implementation using log cat to see the data passing between the activities to help see if the correct data was being sent. This type of testing was essential to helping debug the application.

* 1. **Usability Tests or Evaluation ?**

The student tried to keep in mind Nielsen’s 10 usability heuristics 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. Below is a list of the heuristics, a description of them and how the application meets them[[6]](#footnote-6).

|  |  |  |
| --- | --- | --- |
| **Heuristic** | **Description** | **Application** |
| **Visibility of system status** | The system should always keep users informed about what is going on, through appropriate feedback within reasonable time. | The application keeps the user informed by telling the user if the actions are completed successfully e.g. if an issue occurred editing a recipe or if there is no internet connection a message will appear telling the user this. |
| **Match between system and the real world** | The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order. | The application mainly use icons that follows application conventions for icons such as a plus sign for add and a dustbin for delete. Occasionally a symbol like the font symbols did not represent the real world as much as the student would have liked but this was due to the icons available with the right licenses for use. |
| **User control and freedom** | Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo. | **? Maybe close on the dialogs** |
| **Consistency and standards** | Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions. | Icons and phrases maintain the same throughout the application to maintain consistency and they are common icons and phrases to meet common conventions. |
| **Error prevention** | Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action. | The application has lots of error checking place for example it checks that textboxes which should not be empty aren’t empty and tells the user to fill it in before accepting the information or checking the password meets the correct criteria before expecting. |
| **Recognition rather than recall** | Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate. | All options are made visible in the application page and are easy to access. No option is inside another option so it should be straightforward to see the option and access it. |
| **Flexibility and efficiency of use** | Accelerators -- unseen by the novice user -- may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions. | There wasn’t really a need for accelerators in the application. Recent search suggestions were stored to speed up users recent searches. In the future it might be good to have an accelerator for frequent users to easily access frequent ingredients or method steps. |
| **Aesthetic and minimalist design** | Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility. | The application is as straight forward as possible with no irrelevant information in the dialogues. |
| **Help users recognize, diagnose, and recover from errors** | Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution. | Error messages are all in plain language outlining the issues. Although they could suggest more constructive solutions. |
| **Help and documentation** | Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large. | A user guide was created to help the user use the application but a help guide in the application in the future would also be useful |

1. **Evaluation**

**Discuss ongoing evaluations – prototyping etc**

**7.1 Final Evaluation**

**Comparison- Why other was picked**

**Participants – Getting, describe, amount**

**Process – SUS , feedback, swaping application to avoid favourability**

**7.2 Results**

**7.3 Discussion**

1. **Final Product**

**8.1 My Cookbooks**

**8.2 My Recipes**

**8.3 Recipe View**

**8.4 Recipe Edit**

**8.4 Recipe Contributors**

* 1. **Search**

**8.6 Explore**

1. **Critical Appraisal**
2. **Conclusion**
3. **Future**

**Acknowledgements**

**References**

**Appendix outline**

**Front cover and page stuff**

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, n.d.)

   [↑](#footnote-ref-1)
2. https://snailonabike.files.wordpress.com/2010/05/waterfall.jpg [↑](#footnote-ref-2)
3. http://www.mountaingoatsoftware.com/system/asset/file/17/ScrumLargeLabelled.png [↑](#footnote-ref-3)
4. http://agile-development-tools.com/wp-content/uploads/2010/10/iterative-development1.png [↑](#footnote-ref-4)
5. (Budiu, n.d.)**,** (Gorbsky, 2013)**,** (McCracken, 2013) [↑](#footnote-ref-5)
6. http://www.nngroup.com/articles/ten-usability-heuristics/ [↑](#footnote-ref-6)