Reducing Environmental Impact in the Kitchen

Group 9: Preliminary Design Specification

1.1 Requirements Engineering

1.1.1 Introduction

In the first deliverable we described the motivation and rationale behind developing an application to help individuals reduce their food waste. To re-cap, food-waste contributes to climate change and directly reduces the availability of food. Climate change will have detrimental effects on our way of life and challenges our very existence. Although we strongly believe that the most obvious solution is to reduce our food-waste on a global level, currently alternative techniques are employed such as the use of fertilizers and making way for new areas of agriculture through deforestation, which in itself will contribute to climate change, reducing our food supply and returning to the origin of this vicious cycle. It is difficult for an individual to fathom the detrimental effects of food-waste as it is the collective waste of people as an entity that has deleterious effects. Therefore, our objective is to create an application that appeals to both those individuals who strive to be more eco-friendly, and those who are indifferent.

We have refined and developed the application functionality detailed in deliverable 1. Our current proposed solution tackles food waste via several features. Prior to going shopping, a list is created by selecting recipes and individual ingredients. This task is simple to undertake and hence incentivises meal-planning which is “one of the most effective ways to reduce food waste” [NWPP, 2018]. The user confirms the items purchased and is prompted to enter expiry dates for perishable items, to remind the user pre-expiry (670 thousand tonnes [Munbodh, 2018] of food are thrown out yearly due to the misinterpretation of expiry dates). These items are moved to the users’ 'inventory’ and placed in a virtual food cupboard so that the ‘recipe suggester’ functionality can suggest recipes that use the inventoried ingredients, with a minimum number of additional ingredients. This feature helps reduce food waste as it means all ingredients are used whilst also being attractive to non-eco-friendly users as it reduces meal costs. We take this a step further by helping people minimise their carbon footprint and maximise recyclability by raising awareness through a personal informatics section which displays the user’s data graphically. Once a user is aware of their environmental impact, they can seek to minimize their damage, using features such as sorting algorithms that sort recipes based on carbon footprint or making use of the relevant data associated with each individual food item.

We aim to develop a dynamic application to attract people who strive to be more eco-friendly, through features such as the ‘recipe suggester’ and by providing accurate information on food products, yet also lure the average person due to the simplification of meal planning, and the associated cost reduction. We plan to target students as they are in a pivotal point in their lives moving towards self-sufficiency in terms of cooking and shopping, yet also in a formative stage in habit creation. The University of Illinois states that “18- to 24-year-olds, especially [university] students, have a higher tendency to waste food” [Nikolaus, 2018]. Students also live on a limited budget and want to minimise costs.

Our application will initially be Tesco focused, but new supermarkets will be added in the future. The application is tailored to each supermarket whereby product data (quantities, carbon footprint, perishability, recyclability… etc) is specific and not generalised which in turn simplifies the user experience as well as allowing for more accurate and representative analysis. We chose Tesco for our pilot as it is the 1st ranked [Student Room, 2015]

supermarket of choice among students. Tesco also allows web scraping of their online grocery website and have specific data on carbon footprint and recyclability of their products.

1.1.2 Stakeholders

The primary stakeholders are the users of the application are students, as described earlier. External stakeholders include grocery retailers, distributors and suppliers, and farmers. The UK grocery retail market is valued at £177.5 billion [Vasquez, 2015] whilst £13 billion (7.3%) [Quested, 2015] of food that could have been eaten, was wasted. The farming market value in 2015 was £24 billion [NFU, 2017] and assuming a supermarket mark-up of 12% [Berry, 2008] then the amount food wasted in homes in the UK equates to 48% of the farming market value. These numbers are merely to highlight how a reduction in food waste would negatively impact the grocery retail and farming industries, among others.

It is clearly ambitious to believe that our application will lead to a significant reduction in food-waste on a national level to the point where these other industries will suffer although some decrease in food-production should be noted. A readjustment period could mean hardship for the people employed in these sectors, but will ultimately benefit, and highlight manufactures and retailers who do not over produce and who are more environmentally conscious in production and packaging. Supermarkets could also pressurise their suppliers to adapt or de-list non environmentally friendly producers. With the addition of new supermarkets, a sense of friendly rivalry between supermarkets to be the most eco-friendly could happen and lead to the disclosure of valuable product-related information. One possible short-term impact is that prices will increase. Ecological products and "eco-friendly products are more expensive than traditional products" and this directly affects our target users "students" who live on a limited budget. That said food-waste is projected to indirectly cause crop prices to rise "by approximately fivefold" [Jay, 2020]. So, the bigger question is what are you prepared to pay to contribute to a reduction in climate change?

1.1.3 Aims for Requirement Elicitation

Our app requires consistent use to make a difference, so it is imperative that we listen to our target audience and primary stakeholders – students – while constructing requirements. We decided that we would interview a sample of university students, our main aim being to find out the most common causes of food waste. In particular, we aim to:

- Find reasons why students end up buying too much food
- Learn good habits of those who do not waste food
- Identify foods which get wasted the most often and why
- Inquire about useful apps they already use often

Having completed these aims, we will have a better idea of the type of features we could implement to effectively reduce food waste in students, compare them to our first draft of ideas for features, and make changes. We can then express these features as smaller functional requirements. Later in the development cycle, we could come back to these same students and confirm that the features we have implemented are ones which help with the problems they mentioned here.

To verify that these requirements and features would be useful to a broader audience, we will then write a survey that will be sent to a larger sample of students. We wish to use the responses to verify that students would

gladly use the features constructed from the interviews, and that their use cases line up with the most common reasons that food is wasted. The aims for the survey are:

- Find a consensus on the main reason food is wasted
- Verify that our requirements match up with the needs of students
- Refine and tweak any features based on open-ended questions
- Reconsider planned features that seem to contrast with survey responses.

Additionally, we will also attempt to identify some application requirements which are independent of the features. We will use evidence from various sources to back up the requirement choices.

1.1.4 Requirements

Using the techniques described in 1.1.3 Aims for Requirement Elicitation, we created a requirements specification which can be found in 3.1 Requirements. We have split the requirements into functional and non-functional categories. Below we highlight the most important requirements and how we generated them.

Interviews

Our first interviewee said that the main cause of food waste, from his experience, is food expiring unexpectedly. This was supported by the consensus of our survey, with around 55% of respondents ranking 'Items unexpectedly expiring' as the top cause for food waste in their experience. We drew from this that a primary use case for our system should be to keep track of when foods expire, and to alert users when they need to use up certain ingredients, generating requirements (F3.7), and giving it a high priority. Further group discussion led to the generation of requirements (F3.1), (F3.2), and (F3.3), on which the expiry date tracker is dependent. These requirements describe a food inventory system; we needed an easy way for users to keep track of the food they purchase and consume.

When asked to elaborate, the interviewee mentioned meat as an item that was particularly hard to keep track of. This led us to consider how we should represent quickly perishing items in our app. To strike a balance between smooth user experience and functionality, we decided to give each possible ingredient a ‘Low’, ‘Medium’ or ‘High’ perishability in the database (F2.1). Then, when a user adds a new item to their inventory, its perishability will be checked. If it’s High, they will be prompted to set an expiry alarm for that item specifically (F3.8). One interviewee also suggested a shopping list helper, which was a feature we were already thinking about implementing. This would integrate well with our other planned features, but could lead be difficult to implement under our time constraints, so we decided to write the requirement as one with low priority (F4.6).

Additionally, some of the initial interviews we did for the proposal were also valuable in drafting requirements. One interviewee who suggested to not make the app ‘pushy’ led to us considering user customisability options, such as allowing the user to disable push notifications (NF2.5, F3.9).

Survey

As part of the survey, we provided the participants with a list of statements and asked to which degree they agreed, in order to prioritise requirements that the users would believe to be valuable. The most one-sided statement we provided was “I would like to be able to cook more interesting meals”, with about 91% agreeing or strongly agreeing, in response to which we decided that users should be able to browse and save recipes (F4.1, F4.2, F4.3, F4.5).

In an open-ended question about feature ideas, the most submitted idea was a way to find recipes using leftover ingredients. This feature was also suggested through our interviews, and so we constructed the core requirement for it: (F4.4). As a group we also strongly agreed that this would be a good feature to implement, as it is a natural extension to the inventory feature – we could read the user's current inventory and use that to filter the recipe search. We also specified that this should prioritise recipes that use ingredients close to expiring, in order to provide a solution to the common problem of items unexpectedly expiring – we thought this was a natural extension of the expiry alarm feature, as a user would be notified of an item expiring soon, and the app would take them to recipes they can use that item in before it expires. Additionally, when asking participants to name apps that they use to help with their daily lives, all responses were free mobile applications (e.g. Notes, Apple Calendar, Lidl Plus, MyFitnessPal), which supported our initial idea for our system to also be a free mobile app (NF2.2).

Existing Work

As mentioned in the proposal, there are already many systems in the domain that use different approaches to reduce food waste. The Winnow Vision system, used in restaurants to track food waste, was praised by one of its clients, IKEA, for its ease of use, and specifically how it fits into their workflow "this frees up capacity for recording waste, minimizes time that we spend with the Winnow system" [Winnow, n.d.]. Although our system's stakeholders are completely different, it's still valuable to realise that people are more encouraged to use the system if it isn't obtrusive, which led us to add some requirements to smooth out the user experience – for example, for keeping the inventory tidy by automatically removing items (F3.5) or reminding the user to (F3.4). This also led to us modifying the expiry alarm (F3.7) to be optional, rather than requiring the user to set an alarm on each item they add.

Technical

We did some research into the platform and technologies we are going to use. We identified a range of requirements relating to the application and how it should be developed. Most of them are non-functional and they include ideas related to development and tools. A high priority requirement (NF1.1) says we must use the design pattern MVP (Model-View-Presenter). This was included after researching various Android design patterns described in an article [Quang Nguyen, 2017]. Additionally, (NF1.2) says we should use multiple threads in the application which came from a point made on [Android Developers Guide, 2020]. On top of this, (NF1.3), (NF1.4) and (NF1.5) refer to the development of the system and are included to ensure development is consistent between team members.

Ethical

Our planned features do not rely on collecting data from any users. With this in mind, we should follow the Data Protection Act (2018), which specifies that data collection should 'limited to only what is necessary' - which in our case is not at all. For this reason, all databases will be stored locally. (F2.1-2.4).

1.2 Preliminary Design and Prototyping

1.2.1 Use Case Diagram

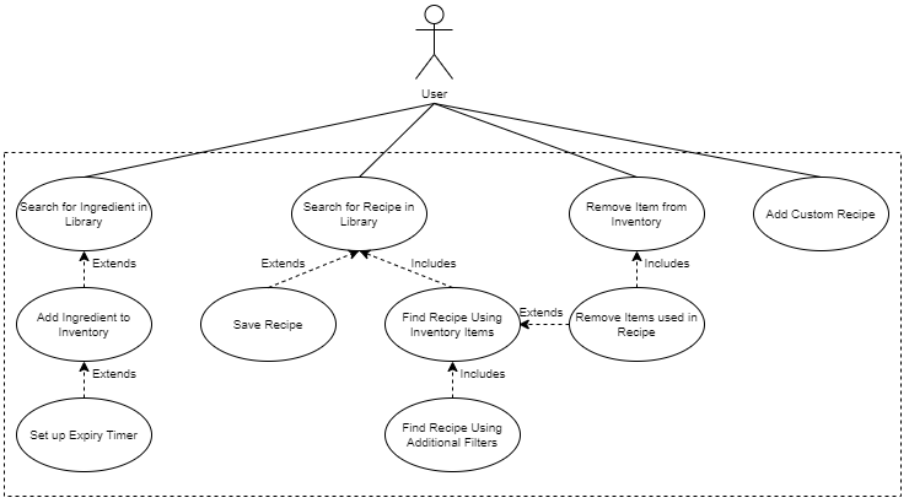


Figure 1 - Use case diagram.

The Use case diagram above in Figure 1 shows possible interactions between the User and the System, as well as the relationships between the different use cases. This diagram shows how our system meets the Users functional requirements. The next paragraph describes some examples of the use cases, justifying why are each necessary to meet our System Requirements, which were gathered from Student interviews and survey responses.

When a User Searches for an ingredient in the library they have the option to add this to their ingredient inventory if they want and in some cases this ingredient will need to be set up with am expiry timer, which is why they extend the 'Search for ingredient in Library' use case. This satisfies F2.2 because the system stores saved ingredients in an inventory database, and F3.2 which states a user must be able to add ingredients to their inventory. When a User searches for a recipe they will be shown in order that the User is closest to being able to make with their inventory, which satisfies requirement F4.4, and F4.1 as well as F4.3 because they can also filter and search through the recipe database. The User can also save recipes for later which is why it extends 'Search for Recipe in Library', this is necessary to satisfy the requirement F4.5.

1.2.2 Scenarios

SCENARIO 1: Adding an ingredient

Actor: User

Preconditions (Initial Assumptions):

- User viewing inventory page

Normal flow of events:

1. User clicks on '+' button in top right
2. User types in search bar the ingredient that they want to add
3. User presses 'Go' button
4. System shows ingredient searched
5. User types in quantity
6. User presses '+' button

7. System adds Ingredient to Ingredient Inventory
    a. If ingredient has perishability – User is prompted to add expiry date

Alternative flow of events:

4a. Ingredient is not in database
    4a1. No ingredients shown when User presses 'Go'
    4a2. Message shown 'no results'

System state on completion:

- Inventory updated to add new ingredient
- User returned to inventory screen

SCENARIO 2: Finding and saving a recipe

Actor: User

Preconditions (Initial Assumptions):

- User in viewing home page

Normal flow of events:

1. User clicks on 'recipes' button
2. System shows Wizard page
3. User scrolls through the recipes, which are ordered by how many additional ingredients they would be required to buy
4. User clicks on desired recipe
5. System shows recipe description
6. User double clicks screen to 'save' recipe
7. System saves recipe

Alternative flow of events:

5a. User does not find a recipe
    5a1. User creates custom recipe

System state on completion:

- Recipe added to saved recipe database
- User on that specific recipe screen

1.2.3 Prototype Testing

After gathering our requirements, we started designing some features to satisfy them, along with their corresponding user interface designs. By creating a prototype and giving it to stakeholders to test, we aimed to ensure that our users would feel comfortable navigating to different features in our app, and to ensure that important controls were intuitive and never confusing. We started by drawing wireframe designs for the important screens in the app along with a diagram showing how to navigate between, them; using both in conjunction to emulate the app for the stakeholders participating in the tests.

The diagram (Figure 2) describes how to get to any screen on the app. In the top-left corner of any screen, there will be a button that will allow the user to move one layer up in this hierarchy – labelled with a back arrow, or if the user is on the second layer, by a menu icon. This button will be part of a toolbar that will consistently be at the top of the screen and moving between screens in a particular layer will also be possible using this bar. The dotted arrows on this diagram represent dependencies of features, for example the recipe wizard is dependent on the library to search through to find recipes, and the ingredient inventory to filter the recipes.

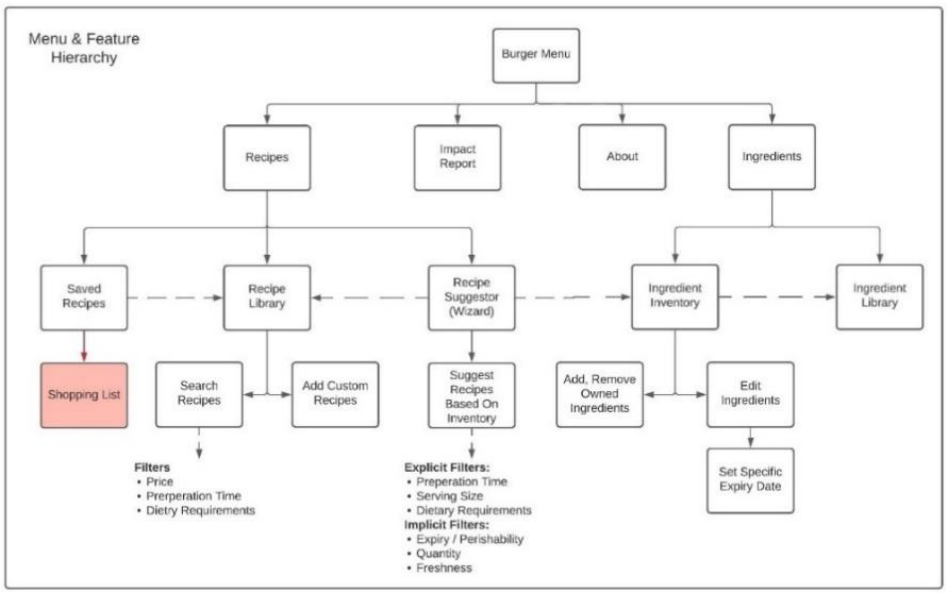


Figure 2 - Hierarchy for menus and features.

We showed the prototype screens to some students in a survey and asked some questions – firstly about the design of the interface in general, secondly about possible colour schemes, and finally about navigation with particular focus on the bar at the top of each screen – we want to make it clear to the user that by looking at this bar, they can find out where they are in the UI tree, how to go back, and where else they can go.

The first useful piece of information was screen orientation – when designing the user interface, we were under the assumption that the final app would appear in portrait mode only on the user's phone. Our survey responses supported this, with around 89% of participants expressing a preference for this orientation.

We also asked an open-ended question in the UI survey to find out features that could be frustrating in a UI in order to avoid them. A user commented on 'getting lost' in a menu, and another commenting on making 'important features immediately accessible' which led us to generate a requirement stating that any screen should be accessible in 5 clicks (NF2.4) in order to ensure that the user always felt in control when navigating the menu. Another user suggested that a 'central toolbar' was required, reinforcing our idea of the bar at the top being able to take the user back or to the menu at any point (F1.8).

Another valuable thing we learned from this survey was which colour scheme to implement. We presented the survey participants with a few different options that we thought suited the app and asked them to select the most visually appealing. The results were not very decisive, with a green-blue colour scheme getting the highest rating, but when asked to give feedback on the colour scheme in general, more than half of the users suggested to add white or a similar contrasting colour, with some commenting that it was 'too colourful'. We decided to stick with the first colour scheme, but also use a much lighter colour as the main background colour for the app.

Figure 3 - UI design and colours for the ingredients, recipes and burger menus.

To ensure that the layouts were clear, and the icons were intuitive, we asked the users some questions on how to navigate the UI. In the small group of participants, 100% of them guessed the correct place to click to add a new item, and to get to the main menu. We took these tests as evidence that our planned UI structure was intuitive to navigate, and went on to formalise this into requirements (F1.1 - F1.4). When asked to guess what the exclamation mark symbol on an item could mean, all of them were on the right track, guessing 'item expired' or 'item expiring soon' (F1.5). We considered these results conclusive evidence that this screen layout was not confusing, and that the icons used on it were meaningful and intuitive, even for a first-time user.

1.3 System Architecture and Models

1.3.1 Architecture & Design Patterns

As mentioned in the system proposal, we are going to use the MVP (Model View Presenter) design pattern as a general architectural structure. It is designed to improve separation of concerns between the UI and business logic, which makes testing and development easier. The model refers to the data required to display a page; we will wrap the model inside a data repository system. The view is a passive interface which exposes functionality for changing the UI and retrieving user input; we will implement Android specific API code in the view. Finally, the presenter acts as a 'middle-man' between the model and view which logically controls the flow of data from the model to the view. We refer to the presenter and view together as a contract since they are commonly used in a one-to-one relationship.

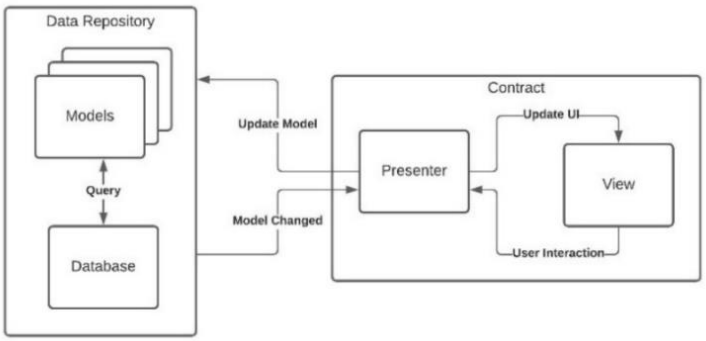


Figure 4 - Generalised HLA for system architecture.

Every page in the application will be represented by a contract behind the scenes with - potentially - many models. We will define all contracts as Java interfaces. This way all functionality is clearly listed (without implementation) and the intent of the contract is obvious to the programmer. Furthermore, we can implement test versions of these interfaces (which do not include an actual UI) for use in unit testing; testing improves the reliability of our system and thus maintains professional integrity and quality of our system.

Our HLA for the entire project (Figure 5) combines both the features and the generalised HLA, turning it into a diagram presenting most of what the project has in it. We will be making use of multiple databases for recipes, ingredients, search history in order for the user's most optimal experience with the app and easy access via the suggestion feature – depending on each user, the suggestions will vary and be personalized. We will also make use of a User Satisfaction database which collects information from the impact report. At the same time, the Inventory, Home Page, Menu, Search and Shopping list pages will all be part of the UI, having a backend with databases as well as an interface.

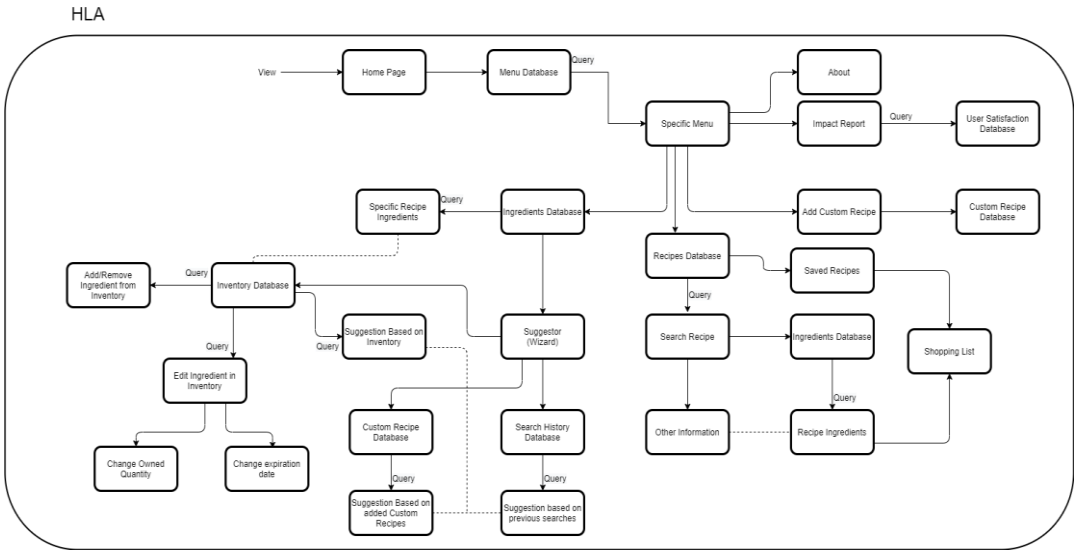


Figure 5 - HLA for entire project.

We created a UML class diagram for the ingredient inventory menu. The presenter contains methods for controlling the general state / intent of the application such as 'refreshIngredients'. These methods are invoked through an event call-back setup inside the view. For example, if the user taps an ingredient, an event will be created which calls the presenters 'ingredientTapped' method. The view exposes methods for showing and hiding various UI components. The presenter handles the flow of data from the model to the view and it gets data from the inventory repository. The model class 'Ingredient' is used to define an ingredient and it contains attributes like name and description. The view must know about this model class so it can see the exposed attributes. The ingredient adapter is an Android specific class which converts the ingredient model data into a form which the view's Android code can use [Android Adapter Reference, 2020].

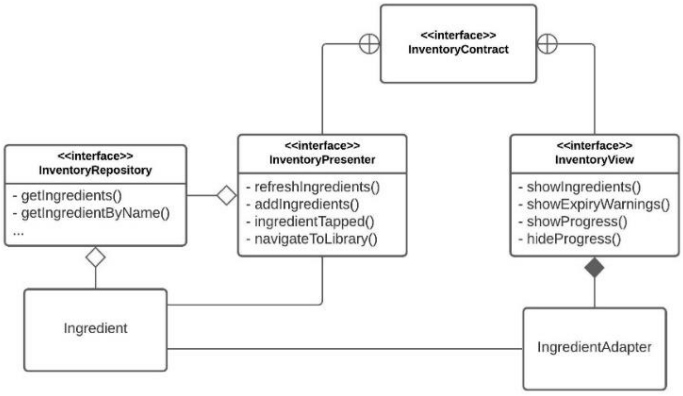


Figure 6 - UML Class Diagram for ingredient inventory contract.

The diagram (Figure 7) shows a sequence diagram for scenario one (1.2.2 Scenarios): the user adding an ingredient to their inventory. It clearly shows how the user interacts with the view which in turn interacts with the presenter to inform it of user events.

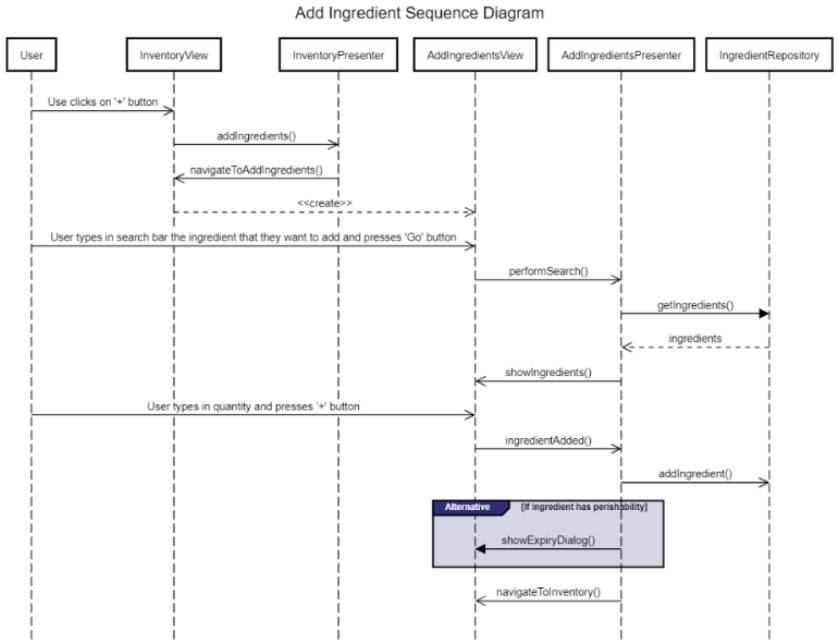


Figure 7 - Sequence diagram for adding an ingredient.

1.3.2 System Components

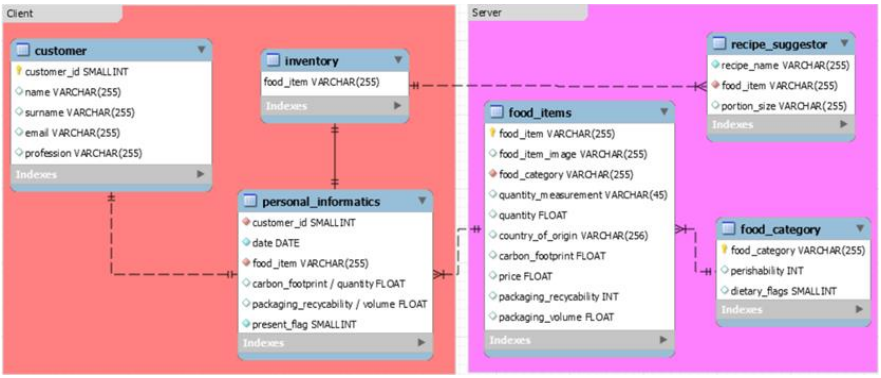


Figure 8 - EER Diagram for Database Design.

We have decided to use a local database system in our application, namely SQLite. The largest motivation behind this decision stems from the feasibility of a remote database. We would like to produce a working system which meets the aims and goals discussed in the proposal, and although our design architecture easily allows for both a remote and local database, it will be much easier to produce a functional product with a local database. On top of this, we are not handling personal data about our users which inherently reduces the risk of data security issues.

1.3 Progress Report

1.3.1 Alignment to Original Programme of Work

Thanks to the programme of work and work plan set out in the first deliverable, the project has progressed efficiently and consistently throughout the requirement gathering and advanced design stage. As decided in the work plan and carried out while completing the first deliverable, progress has been evaluated regularly and by all team members on a weekly basis. This allowed us to ensure we were on track for this deliverable and reassess the task to be completed in the next week/sprint as appropriate. Trello, was used again as our primary tool for group management and task progress tracking, as planned in our programme of work and work management. While the Gantt chart was followed reasonably closely, due to a week extension being given, the timeline for the finalisation of the documentation was extended to this additional week. This allowed us extra time to gather user feedback on the proposed user interfaces including potential colour schemes to better tailor the user interface to the stakeholders needs as well as finish all necessary components of the design of our system and the documentation. Apart from the week extension, we have stuck to the original timeline and intend to do so for the next deliverable.

1.3.2 Challenges Faced

While the majority of this stage was carried out with ease, the ever-developing coronavirus pandemic situation has continued to bring challenges to our work. The main issue that arose as a result of the pandemic was the UK's second national lockdown, reducing capacity for in person social interactions, during the time we wished to gather requirements from stakeholders to better shape our design of the project to better fit our stakeholders needs. In order to remedy this, we opted for conducting virtual interviews and creating an online questionnaire which was distributed to stakeholders. Even despite our solution of moving everything online, we were still relatively limited by who we perform interviews with meaning they were largely conducted with friends of our team which could result in some bias of interview responses.

Another problem our team faced because of the pandemic, was the UK government guidance for students to return home in the penultimate week of this task. While not all team members went home in this period, the guidance extended the period of disruption caused by team members returning home and being inactive for a day or more meaning whole team meetings were not always possible. As a result, we decided to meet more frequently, up to twice a week whenever a large portion of the group was available in the hopes that all members of the team would be updated at some point each week. At the same time, for those who were not available, there were consistent updates posted in our group chats and on Trello, allowing them to be kept up to date remotely.

1.3.3 Evaluation of Current Methodology

The process undertaken to complete this deliverable has been very efficient with work levels kept consistent throughout. This was likely due to having a better plan of which tasks will be completed in which weeks and by whom. Our iterative approach has allowed us to work more flexibly and encourage the project to be shaped more easily by the stakeholders needs. While some core features could be described in the requirements before completing stakeholder interviews, a large portion of our features were collected from stakeholder feedback and added to our requirements int eh following sprint. Despite the ongoing situation with the coronavirus pandemic, we were able to collect a number of requirements from stakeholder interviews and surveys conducted through video calls, COVID secure in person interviews and online forms. Whilst this has expanded the scope of our project, this has not been too significant so we believe our system will still be achievable with scope to expand in the future.

We hope to continue our current methodology in the future as it has allowed us to be flexible. Whilst the waterfall method would have constrained us to work on one section at a time and to continue 'forward' progress in spite of feedback, the iterative Agile methodology chosen has allowed us to adapt and return to elements of the design upon receiving more stakeholder engagement and for different members of the group to work simultaneously on different sections of the design. This ultimately has allowed us to work more efficiently and complete the design in less time as any inconsistencies in the design can be identified and rectified in the weekly scrum meetings.

1.3.4 Risk Re-assessment

As previously identified, one of the main risks to plan to mitigate going forward with our system is that of personnel shortfalls. The risk that one of the team members falls ill or is affected by the coronavirus pandemic or anything else in some way is of high probability. Therefore, we plan to re-distribute or re-assign the workload amongst the group to provide the most suitable task for an affected member dependent on their situation, with clear online communication throughout the period they have difficulties. This is made possible by our agile working methodology

After starting to design our system, more risks have become apparent. We started to realise that our scope from the proposal may have been too ambitious, and so while collecting requirements from stakeholders we made a conscious effort to limit the ones we implement to ones that are supported by a lot of different sources, and prioritising those that a lot of stakeholders agree on. Nonetheless, feature-creep is a risk that we should continue to be aware of as we move on to developing our app, and to mitigate this risk, we should focus on implementing the highest priority requirements first, and only moving on to the lower ones when we know we have the time to properly integrate them without disrupting the ones we already have.

Another risk to consider is potential miscommunication amongst group members. Within a lockdown environment, communicating online across multiple platforms such as Microsoft teams, GitHub and on our build tool for the code itself, can easily lead to confusion. Our main method of preventing this is the use of Trello, this tool makes it easier to keep track of who oversees which tasks, thus providing an easy visual representation of what each member must do to contribute to the project, avoiding overlap or missed objectives.

Also, as we will be moving on to start programming soon, we should be more cautious of potential issues with the software we use. Without a backup or version control system, a crash in the IDE software could result in a lot of lost work. For this reason, we should vigilantly make use of GitHub as we start developing the software and make backups of our project often. This also applies to any surrounding paperwork but continuing to store our work on the Microsoft Teams cloud as well as saving locally often should be sufficient mitigation for this risk.

2 References

- NFU, 2017, 'Farming's value to Britain - key facts' [Accessed online 12/12/2020]:
  - https://www.nfuonline.com/news/latest-news/farmings-value-to-britain-key-facts/
- Berry, T, 2nd December 2008, 'On average, how much do stores mark up their products' [Accessed online 12/12/2020]:
  - https://www.entrepreneur.com/answer/221767
- Quested, T, January 2017, 'House Food Waste in the UK, 2015' [Accessed online 12/12/2020]:
  - http://www.wrap.org.uk/sites/files/wrap/Household_food_waste_in_the_UK_2015_Report.pdf
- Vasquez, J, 22nd December 2015, 'Retail Foods' [Accessed Online 12/12/2020]:
  - https://apps.fas.usda.gov/newgainapi/api/report/downloadreportbyfilename?filename=Retail%20Foods_London_United%20Kingdom_12-22-2015.pdf
- Student Room, 2015, 'Best supermarket for students' [Accessed Online 16/12/2020]:
  - https://www.thestudentroom.co.uk/content.php?r=13833-best-supermarket-students
- Nikolaus, C, 22nd August 2018, 'Why are young adults wasting so much food? Study looks at perceptions and food behaviors' [Accessed Online 16/12/2020]:
  - https://www.sciencedaily.com/releases/2018/08/180822122832.htm
- Munbodh, E, 8th October 2018), 'Tesco axes "best before" dates from 116 more lines and it could save you £470' [Accessed Online on 16/12/2020]:
  - https://www.mirror.co.uk/money/tesco-remove-best-before-dates-13382848
- NWWF, EPA, 11th July 2018, 'Meal Planning' [Accessed Online on 12/12/2020]:
  - https://stopfoodwaste.ie/resource/meal-planning
- Android, Android Developer Guides [Accessed Online on 22/11/20]:
  - https://developer.android.com/guide
- Quang Nguyen, Apr 3, 2017, 'Architecture patterns in Android' [Accessed Online on 22/11/20]:
  - https://android.jlelse.eu/architecture-patterns-in-android-abf99f2b6f70
- Android, Adapter Reference [Accessed Online on 11/12/20]:
  - https://developer.android.com/reference/android/widget/Adapter
- Winnow, 'Save time and money. Capture, track and manage your food waste with ease' [Accessed Online on 18/10/20]:
  - https://www.winnowsolutions.com/vision
- Gov.uk, 'Data Protection', [Accessed Online on 16/12/20]:

- https://www.gov.uk/data-protection

3 Appendices

3.1 Requirements

Functional Requirements				
1 – User Interface				
Ref	**Description**	**Dependencies**	**Priority**	**Source**
F1.1	The application must have a screen which displays all items in the user's ingredient inventory. Along with the ingredient name, its icon and amount should also be displayed.	F3.1	H	Prototype Testing
F1.2	The application must have a screen which displays all recipes available in the database. There should also be an option to prioritise recipes that can be made using ingredients in current inventory.	F4.1, F4.4	H	Prototype Testing
F1.3	The application must have a screen which displays the user's monthly carbon footprint, by totalling all ingredients added to the inventory in that month.	F3.1	L	Prototype Testing
F1.4	The application must have a screen which contains a menu that leads to the inventory, recipe and carbon footprint screens.	F1.1, F1.2, F1.3	H	Prototype Testing
F1.5	The application should display ingredients with an associated Expiry Reminder at the top of the list, sorted by nearest to expiry first, along with the time remaining until their set Expiry Date. Items expiring within the next day should show an exclamation mark warning.	F1.1, F4.2	L	Prototype Testing, Group Discussion
F1.6	The application should change the colour of the border of each ingredient depending on their Perishability.	F1.1, F4.1	L	Group Discussion
F1.7	The application should group instances of the same ingredient into one slot in the inventory. Then, after clicking on the slot, the application should display a menu of all instances with their respective amounts, dates added, and Expiry Reminders if they have them, along with a description of that ingredient.	F1.1, F2.2, F4.2	M	Group Discussion
F1.8	The application must have a bar at the top of each screen, containing a back button, and buttons for all other screens accessible from the current one.	-	H	Prototype Testing
2 – Database				
Ref	Description	Dependencies	Priority	Source
F2.1	The application must store a local database of ingredients. An ingredient will consist of: Name, Description, Picture, Perishability (i.e. 'Low', 'Medium', 'High'), Dietary Flags (e.g. Vegetarian, Vegan, etc.), Quantity Measurement (e.g. Kg, Ml), Carbon Footprint ($g\ CO_2\ e$), Price.	-	H	Group Discussion, Ethical Considerations
F2.2	The application must store a local database of the user's current inventory. A record will consist of: Ingredient (foreign key from F2.1), Date Added, Amount, (Specific Expiry Date).	F2.1	H	Group Discussion, Ethical Considerations

F2.3	The application must store a local database of recipes. A recipe will consist of: Name, Description, Picture, Ingredients, Preparation Time (i.e. 'Short', 'Medium', 'Long'), Number of Servings, Instructions.	F2.1	H	Group Discussion, Ethical Considerations
F2.4	The application must store a local database of saved recipes.	F2.1	H	Group Discussion, Ethical Considerations
3 – Ingredients				
Ref	**Description**	**Dependencies**	**Priority**	**Source**
F3.1	The application must allow the user to view a persistent inventory of the ingredients they currently have.	F2.1	H	Student Interviews
F3.2	The application must allow the user to add new ingredients to their inventory.	F3.1	H	Student Interviews
F3.3	The application must allow the user to remove old ingredients from their inventory.	F3.1	H	Student Interviews
F3.4	The application should remind the user, with a push notification, every week to manually remove used and expired ingredients from the inventory.	F3.3	M	Existing work in the domain: Winnow Vision
F3.5	The application should automatically assume that quickly perishing ingredients are expired or not present after one month.	F3.1	M	Existing work in the domain: Winnow Vision
F3.6	The application should allow the user to override the assumption that an ingredient is expired or not present.	F3.5	M	Group Discussion
F3.7	The application must allow the user to set a detailed expiry date reminder for a specific ingredient.	F3.1	H	Student Interviews
F3.8	The application should ask the user if they would like to set an expiry date reminder if they add a 'High' perishability ingredient to the inventory.	F3.7	M	Student Interviews
F3.9	The application should notify the user of potential ingredient expiration.	F3.7	M	Student Interviews
F3.10	The application must allow the user to view a library of ingredients.	F2.1		Group Discussion
4 – Recipes				
Ref	**Description**	**Dependencies**	**Priority**	**Source**
F4.1	The application must allow the user to view a library of recipes.	F2.3	H	Survey Responses
F4.2	The application must allow the user to add custom recipes to the library.	F4.1	H	Survey Responses
F4.3	The application must allow the user to search for recipes in the library and apply filters to: price, preparation time and dietary requirements.	F4.1	H	Survey Responses
F4.4	The application must be able to suggest recipes to the user based on their current inventory. The application will determine recipes by considering expiry / perishability, quantity and freshness of ingredients in the ingredient inventory. Additionally, the user can apply explicit filters for preparation time and dietary requirements.	F4.1, F3.1	H	Student Interviews
F4.5	The application must allow the user to save recipes they wish to use so that they can refer to them later.	F4.1	H	Survey Responses

F4.6	The application may allow the user to generate a shopping list of missing ingredients from their current list of saved recipes.	F4.5	L	Student Interviews

Non-Functional Requirements				
1 – Development				
Ref	**Description**	**Dependencies**	**Priority**	**Source**
NF1.1	The system must be implemented using the MVP (Model-View-Presenter) design pattern.	-	H	Architecture patterns in Android [Quang Nguyen, 2017]
NF1.2	The application should make use of multi-threading where possible.	-	M	Android Developers Guide
NF1.3	Fundamental aspects of the application should be thoroughly tested using unit tests (which can be run at any time).	-	M	Group Discussion
NF1.4	The source code for the application should include informative comments about its indented use and functionality.	-	M	Group Discussion
NF1.5	Changes to the source code should be performed on separate branches of the Git repository.	-	M	Group Discussion
2 – System Quality				
Ref	**Description**	**Dependencies**	**Priority**	**Source**
NF2.1	The user-interface should not be unresponsive for more than five seconds.	NF1.2	M	Group Discussion
NF2.2	The application must free for all users.	-	H	Survey Responses
NF2.3	The application should follow the Android Developer Guidelines.	NF2.1	M	Android Developer Guides, 2020
NF2.4	All user interface menus / screens should be accessible from the start menu / screen within a maximum of five clicks.	-	M	Group Discussion
NF2.5	Users must have the option to disable push-notifications from the application.	-	H	Interviews from Proposal, Group Discussion
3 – Security & Ethics				
Ref	**Description**	**Dependencies**	**Priority**	**Source**
NF3.1	All collected data must follow comply with the Data Protection Act.	-	H	Data Protection Act, 2018
NF3.2	The team must comply with the ACM Code of Conduct.	-	H	ACM Council, 2018

3.2 Wireframe Diagrams

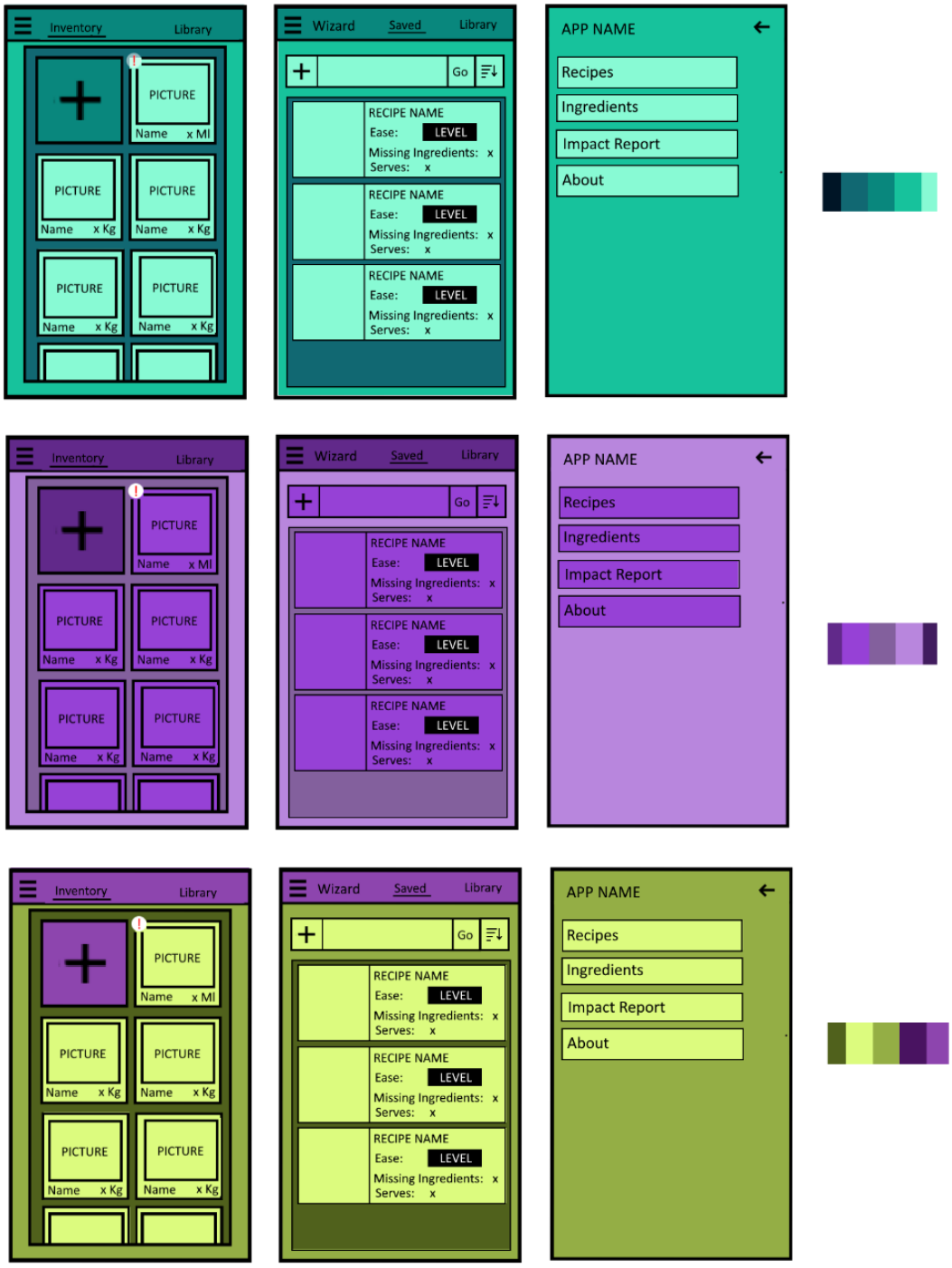


Figure 9 - UI colour schemes that were shown to stakeholders in a questionnaire

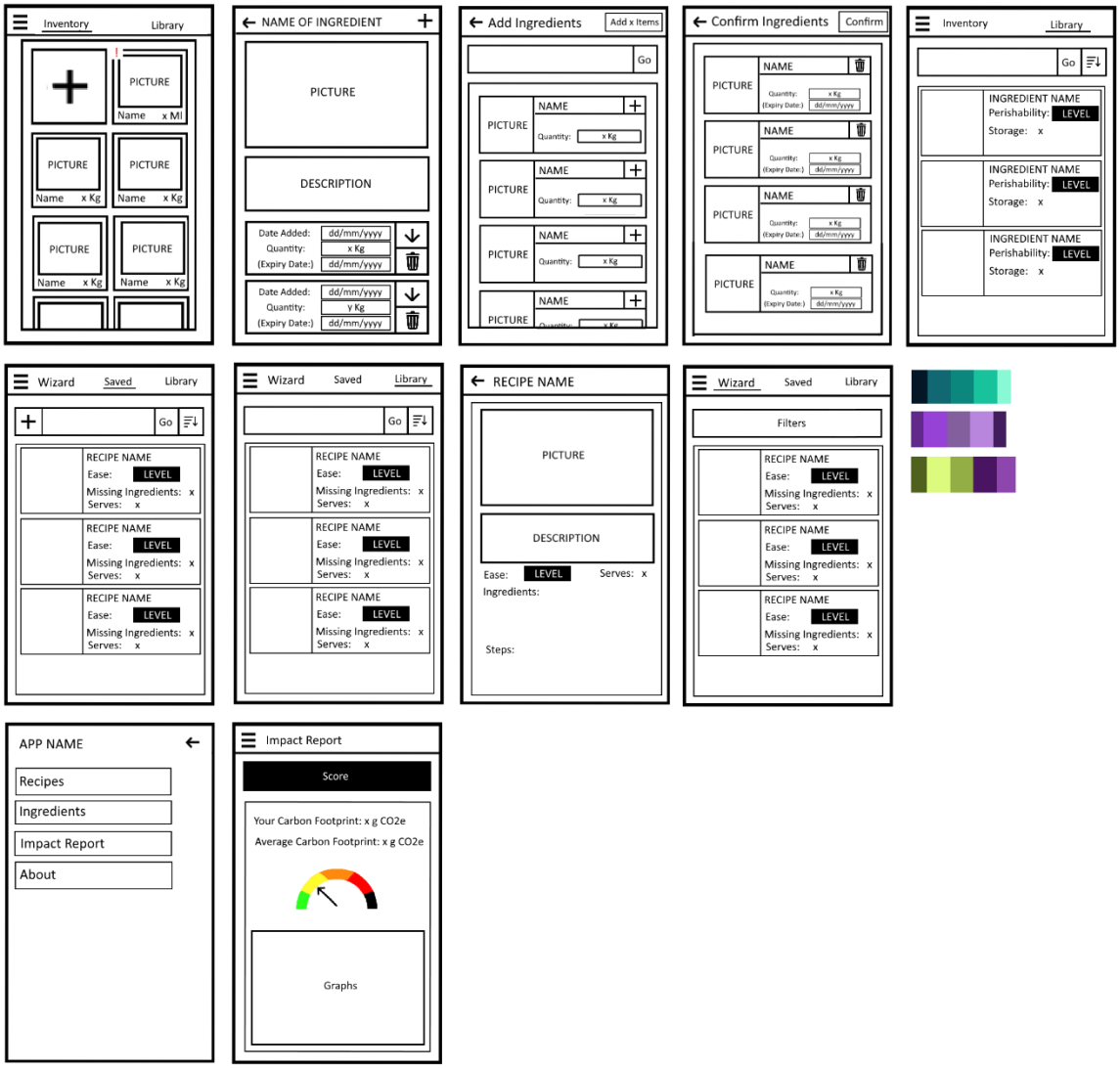
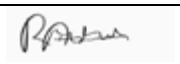
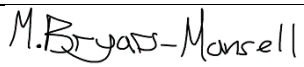
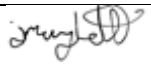
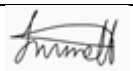


Figure 10 - UI prototype in black and white

4 Contribution Form

Name	Contribution	Signature
Richard Andrews	8	
Max Bryars-Mansell	10	
Jakob Aylott	8	
Teodora Dinca	8	
Lucy Emmett	8	
Jac Griffiths	10	
Nicolas Sanchez	10