The Plant Pantry

A web application for amateur gardeners growing their own food

BSc Hons Interactive Computing

Abstract

Gardening has is widely viewed as an advantageous activity for bolstering mental and physical health (Middleton, 2018; RHS, 2021). While always existing as a popular pastime for these reasons among others, gardening with the intention of growing edible produce has additionally gained popular acclaim given the current societal opinion towards health and the environment. Aspirations to eat a healthy diet, end food poverty, reduce food waste and follow food trends, such as Vegetarianism or Veganism, has made the concept of growing your own food more attractive to a wider audience.

Despite this interest, there is hesitancy to pursuing this activity due to lack of food education, with Beytes (2013) citing that reluctance to gardening largely stems from perceived time restraints, spatial limitations, and lack of knowledge. The aim of The Plant Pantry is to generate a full-stack web application for amateur gardeners who want to learn how to grow their own fruit, vegetables, herbs, and edible flowers, while mitigating perceived issues regarding growing food. This is done through enabling users to view planners, articles, and plant information in addition to recipes, which provides encouragement for users to experiment with their produce in the kitchen.

In addition to these areas, this application is highly collaborative. There is an interactive guide where users can receive tailored advice on what to grow based on their current situation and expertise and it is possible to curate lists for produce and recipes. There is functionality for adding and removing produce items to and from currently growing lists and wish-lists, while recipes can be added to liked recipes lists. Recipes can be uploaded by any user to the system and administrative privileges provide additional functionality.

A professional, organised approach to application development was executed during the implementation of this project, with a range of technologies applied to ensure its success. Requirements were gathered prior to beginning development, which informed the overall project plan established, and during the development process a range of programming technologies and software tools were utilised to manage and generate application code. The core elements include the MERN (MondoDB, Express, React, Node) framework (MongoDB, 2022a) for implementation, Azure DevOps Repos for version control, and Trello for project management.

An Agile-Waterfall Hybrid methodology was executed (Tan, 2019), followed by multiple rounds of user testing, which led to requirement changes and insight into the achievement of application objectives and user expectations.

Results obtained from the various testing practices carried out exhibit the effectiveness of the application created in achieving its aims and surpassing user expectations in a number of ways, from functionality to aesthetics. Improvements can always be made, which are discussed in this report, but the overarching findings display that the application is very successful and possesses potential for the future.

Acknowledgements

I wish to thank the lecturers I have had during my time studying BSc Hons Interactive Computing at Ulster University for their tutelage and support which has enabled me to develop my technical and personal skills.

In particular, I would like to express my gratitude to Professor Johnathon Wallace who has been a wonderful mentor throughout the course of this project, offering much appreciated guidance, support, encouragement, and wisdom.

My thanks extends to those who dedicated their time to assisting in requirements gathering and testing for this project, whose involvement has been of substantial importance and influence.

I would also like to thank those within the technology industry I have interacted with in any capacity during the course of my degree, whose influence has given me confidence to undertake this project and a career within this field. My appreciation especially goes to the Openreach IT E2E Testing and Tools team at BT, where my placement was carried out.

I give my greatest thanks to my family and friends, who have supported me through all my personal and academic endeavours over the years. I am grateful for having them as my champions and hope to continue to make them proud.

Contents

1.0: Requirement Control Document & Modification of the Project Plan

1.1: Final List of Requirements

The requirements identified below have been ordered into groups based on the feature they refer to. The MoSCoW rating process, influenced by developer and user feedback, and relative weighting have been used to determine the priority of each requirement.

MoSCoW has been used given its popularity, simplicity, and facilitation of changing requirements (Ahmad et. al., 2017; Girvan and Paul, 2017). Requirements are placed into one of four groups; must have, should have, could have, or won't have. This establishes the MoSCoW acronym and what elements the application is expected to possess (see Figure 1.1.1).

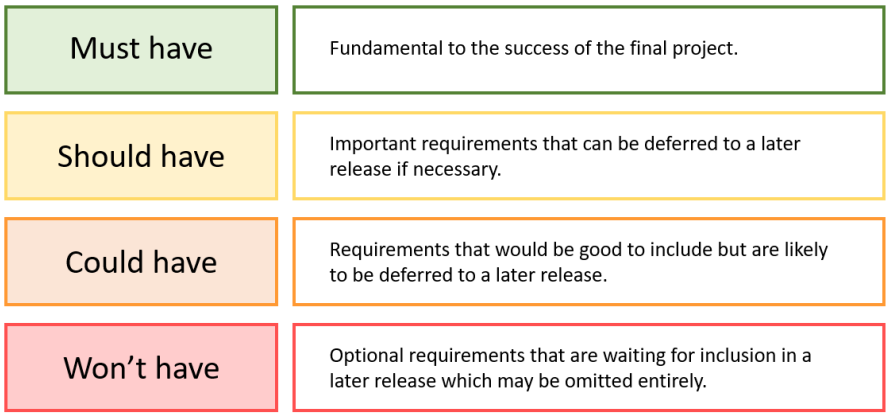


Figure 1.1.1: The four groups a requirement can be allocated to during the MoSCoW prioritisation process (Modified from: Ahmad et. al., 2017; Girvan and Paul, 2017).

As the MoSCoW method does not prioritise requirements at an individual capacity, relative weighting was utilised in conjunction with this strategy. Relative weighting gathers customer benefit, customer penalty, developer cost and developer risk values for each requirement. These are scored on a scale of one to nine, with one representing low impact. Percentages are calculated based on the values and the priority the requirement possesses is determined using the calculation:

Priority = Value %/ (Cost % * Relative Cost + Risk % * Relative Risk)

(Weigers, 1999)

After all requirements have had a priority calculated, the list is sorted in descending order.

The calculations for determining relative weighting are shown in Appendix B.1. The majority of requirements are dependent upon the successful execution of others, which additionally impacts requirement priority.

Each requirement has a level of risk associated with it and was calculated using a 5 x 5 risk matrix (see Figure 1.1.2). Within this matrix the score allocated to the likelihood of requirement being executed is multiplied by that of the impact of failure to establish a risk score:

Risk score = impact * probability

(HASpod, 2020; Zhou, 2016)

	Possibility				
Impact	1 (0 – 20%)	2 (21 – 40%)	3 (41 – 60%)	4 (62 – 80%)	5 (81 – 100%)
	Very unlikely	Low likelihood	Likely	Highly likely	Near certainty
5 (very high)	**5**	**10**	**15**	**20**	**25**
4 (high)	**4**	**8**	**12**	**16**	**20**
3 (medium)	**3**	**6**	**9**	**12**	**15**
2 (low)	**2**	**4**	**6**	**8**	**10**
1 (very low)	**1**	**2**	**3**	**4**	**5**

Extreme Risk (15 – 25)

High Risk (15 – 25)

Moderate Risk (4 - 6)

Low Risk (1 – 3)

Figure 1.1.2: Risk matrix used in the risk assessment for this project (Modified from: Zhou, 2016).

Determining a risk score for each constraint can provide valuable insight into the importance, through the impact weighting, and the complexity, through the possibility score, of each requirement. This can have a great influence on the attention and thought bestowed on every constraint involved in the application being created.

In terms of the risk scores allocated, they refer to the likelihood of the impact of failure and possibility requirement execution. Given that some of the requirements have not been implemented based on the MoSCoW and relative weighting prioritisation methods, their risk scores are based on their hypothetical implementation.

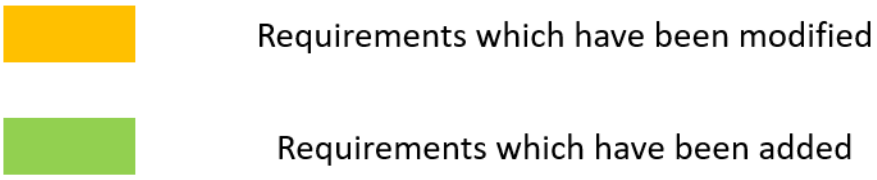


Figure 1.1.3: The colours used in sections 1.1.1 and 1.1.2 to represent changes made to the requirements in the transition from AP1 to AP2.

Requirements are listed within the table below in order of requirement ID, to ensure features are kept together and one prioritisation method is not applied over another to establish order. New and modified requirements can be identified using the colour code outlined in Figure 1.1.3.

1.1.1: Functional Requirements

ID	Requirement	Description	Dependencies	Risk Score	MoSCoW Rating	Relative Weighting	Modification Description
FR01	Account sign-up	User should be able to sign up for an account.	N/A	20	MUST	1.036	N/A
FR02	Account login	User should be able to log in to their account.	FR01	25	MUST	1.121	N/A
FR03	Account logout	User should be able to logout of their account.	FR02	16	MUST	1.684	N/A
FR04	Account update	User should be able to securely update their account details.	FR02	12	MUST	0.569	N/A
FR05	Account deletion	User should be able to delete their account.	FR02	9	MUST	2.142	N/A
FR06	Plant information	Information on fruit, vegetables, herbs, and edible flowers should be available to be read.	N/A	25	MUST	0.602	N/A
FR07	View plant encyclopaedia	User should be able to view a list of all fruit, vegetables, herbs, and edible flowers they can grow.	FR02, FR06	20	MUST	0.534	MoSCow Rating changed from 'SHOULD' to 'MUST'.
FR08	Plant encyclopaedia type filter	User should be able to filter the plant list by type.	FR07	8	MUST	0.505	MoSCow Rating changed from 'SHOULD' to 'MUST'.
FR09	Plant encyclopaedia search	User should be able to search for a particular plant by name.	FR07	6	MUST	0.581	MoSCow Rating changed from 'SHOULD' to 'MUST'.
FR10	View single plant	User should be able to view information for each specific plant.	FR07	20	MUST	0.565	MoSCow Rating changed from 'SHOULD' to 'MUST'.
FR11	Add plant to currently growing	User should be able to add plant to area to record what they are currently growing.	FR10	20	MUST	0.657	MoSCow Rating changed from 'SHOULD' to 'MUST'.
FR12	View currently growing	User should be able to view all plants they are currently growing.	FR01, FR11	16	MUST	0.906	MoSCow Rating changed from 'SHOULD' to 'MUST'.

FR13	Delete plant from currently growing	User should be able to remove plant from list of what they are currently growing.	FR11, FR12	12	MUST	1.428	MoSCow Rating changed from 'SHOULD' to 'MUST'.
FR14	Currently growing sort	User should be able to sort list of plants in currently growing list by type.	FR12	6	COULD	0.693	MoSCow Rating changed from 'SHOULD' to 'COULD'.
FR15	Add plant to wish-list	User should be able to save plant to their wish-list.	FR10	20	MUST	0.614	MoSCow Rating changed from 'COULD' to 'MUST'.
FR16	View wish-list	User should be able to view all plants in their wish-list.	FR01, FR15	16	MUST	1.071	MoSCow Rating changed from 'COULD' to 'MUST'.
FR17	Remove plant from wish-list	User should be able to remove plant from their wish-list.	FR15, FR16	20	MUST	1.339	MoSCow Rating changed from 'COULD' to 'MUST'.
FR18	Open interactive guide	User should be able to open the interactive guide.	FR02	20	MUST	0.987	N/A
FR19	Close interactive guide	User should be able to close the interactive guide.	FR18	25	MUST	1.254	N/A
FR20	Interactive guide parameters	User should be able to interact with a guide that can help determine what the best plants to start growing at that particular time based on criteria, such as the season and available space.	FR06, FR18	25	MUST	0.534	N/A
FR21	Interactive guide response changes	User should be able to go back and change their responses to questions asked during use of the interactive guide.	FR18	12	MUST	0.744	N/A
FR22	Interactive guide page redirect	During use of the interactive guide the user should be redirected to the appropriate pages within the application based on their contact with the interface.	FR18	8	SHOULD	0.434	MoSCow Rating changed from 'COULD' to 'SHOULD'.
FR23	View planner for all plants	User should be able to view a planner detailing when to sow and harvest all produce based on the current month.	FR06	15	MUST	0.642	N/A

FR24	View planner for current plants	User should be able to view a tailored version of the planner based on what they are currently growing.	FR06, FR11	10	SHOULD	0.586	N/A
FR25	Planner type filter	User should be able to view a planner of each type of produce.	FR06	9	SHOULD	0.820	New requirement added.
FR26	Planner search by name	User should be able to search the planner for produce by name.	FR06	6	SHOULD	0.793	New requirement added.
FR27	Access to articles	User should be able to view a list of helpful advice articles containing top tips.	FR02	5	MUST	0.848	N/A
FR28	View a single article	User should be able to view a single article.	FR27	6	MUST	0.768	N/A
FR29	View all comments on a single article	User should be able to view all the comments added to a single article.	FR28	6	COULD	0.540	N/A
FR20	Search articles using keywords	User should be able to search using keywords to find appropriate articles.	FR28	4	SHOULD	0.678	N/A
FR31	Comment on article	User should be able to add a comment to an article.	FR28	9	COULD	0.625	N/A
FR32	View all article comments	User should be able to view all comments they have made on articles.	FR31	6	COULD	0.773	N/A
FR33	Edit comment on articles	User should be able to edit a comment they have added to an article.	FR31	2	WON'T	0.357	N/A
FR34	Delete comment on articles	User should be able to delete a comment they have added to an article.	FR31	6	COULD	0.943	N/A
FR35	Location entry	User should be able to enter their location.	FR02	6	WON'T	0.737	MoSCow Rating changed from 'COULD' to 'WON'T'.
FR36	Tailored plant care	User should be able to view plant care advice based on the weather at their location.	FR35	9	WON'T	0.397	MoSCow Rating changed from 'COULD' to 'WON'T'.
FR37	Notifications for plant care	User should be able to receive notifications regarding plant care based on the weather.	FR06, FR35	6	WON'T	0.357	N/A
FR38	Recipe section	User should be able to view a list of recipes for dishes which can be made using produce grown.	FR02	15	MUST	0.865	MoSCow Rating changed from 'COULD' to 'MUST'.

FR39	View single recipe	User should be able to view an individual recipe.	FR38	20	MUST	0.623	MoSCow Rating changed from 'COULD' to 'MUST'.
FR40	Comment on recipe	User should be able to comment on a recipe.	FR39	9	COULD	0.422	N/A
FR41	View all comments on a single recipe	User should be able to view all comments made on recipes.	FR39	6	COULD	0.499	N/A
FR42	View all recipe comments	User should be able to view all comments they have made on recipes.	FR39	4	COULD	0.391	N/A
FR43	Edit recipe comment	User should be able to edit their comment on a recipe.	FR40	2	WON'T	0.269	MoSCow Rating changed from 'COULD' to 'WON'T'.
FR44	Delete recipe comment	User should be able to delete their comment on a recipe.	FR40	6	COULD	0.673	N/A
FR45	Recipe search by title	User should be able to search for recipes by title.	FR38	8	SHOULD	0.831	MoSCow Rating changed from 'COULD' to 'SHOULD'.
FR46	Recipe search by author	User should be able to search for recipes by author.	FR38	4	SHOULD	0.833	MoSCow Rating changed from 'COULD' to 'SHOULD'.
FR47	Recipe search by ingredient	User should be able to search for recipe by ingredient they can grow.	FR38	9	SHOULD	0.695	MoSCow Rating changed from 'COULD' to 'SHOULD'. Ingredient specified as one they can grow.
FR48	Recipe search by diet	User should be able to search for recipe by dietary requirement.	FR38	9	SHOULD	0.600	MoSCow Rating changed from 'COULD' to 'SHOULD'.
FR49	Add recipe to favourites	User should be able to favourite an individual recipe.	FR39	16	MUST	0.685	MoSCow Rating changed from 'COULD' to 'MUST'.
FR50	View favourite recipes	User should be able to view favourites list.	FR01, FR49	12	MUST	1.010	MoSCow Rating changed from 'COULD' to 'MUST'.

FR51	Remove recipe from favourites	User should be able to remove a recipe from their favourites list.	FR49, FR50	12	MUST	1.160	MoSCow Rating changed from 'COULD' to 'MUST'.
FR52	Recipe upload	User should be able to upload their own recipe.	FR02	15	MUST	0.483	MoSCow Rating changed from 'COULD' to 'MUST'.
FR53	View all recipe uploads	User should be able to view all recipes they have uploaded.	FR52	12	MUST	0.643	MoSCow Rating changed from 'COULD' to 'MUST'.
FR54	Edit recipe uploaded	User should be able to edit a recipe they have uploaded.	FR52	9	SHOULD	0.506	MoSCow Rating changed from 'COULD' to 'SHOULD'.
FR55	Remove recipe uploaded	User should be able to delete a recipe they have uploaded.	FR53	9	MUST	0.952	MoSCow Rating changed from 'COULD' to 'MUST'.
FR56	Access to all forum content	User should be able to access a forum to view community questions posted to the forum.	FR02	12	COULD	0.673	N/A
FR57	View single forum post	User should be able to view an individual question posted on the forum.	FR56	12	COULD	0.780	N/A
FR58	View all comments on single post	User should be able to view all comments for a forum question.	FR57	6	COULD	0.535	N/A
FR59	Add post to forum	User should be able to add a post to the forum.	FR56	9	COULD	0.883	N/A
FR60	View all forum posts	User should be able to see all posts they have added to the forum.	FR59	10	COULD	1.076	N/A
FR61	Edit forum post	User should be able to edit a post they have added to the forum.	FR59	6	WON'T	0.571	N/A
FR62	Remove forum post	User should be able to delete a post they have added to the forum.	FR60	4	COULD	0.773	N/A
FR63	Add comment to forum post	User should be able to add a comment to a post on the forum.	FR57	9	COULD	0.657	N/A
FR64	View all comments made on forum	User should be able to view all comments they have made to posts on the forum.	FR63	6	COULD	0.300	N/A

FR65	Edit comment on forum post	User should be able to edit a comment they have made on the forum.	FR63	4	WON'T	0.316	N/A
FR66	Remove comment on forum post	User should be able to delete a comment they have made on the forum.	FR64	6	COULD	0.714	N/A
FR67	Forum post search by author	User should be able to search for forum posts by author.	FR56	3	COULD	0.268	N/A
FR68	Forum post search by category	User should be able to search for forum posts by category.	FR56	4	COULD	0.929	N/A
FR69	Forum post search by title	User should be able to search for forum posts by title.	FR56	4	COULD	0.581	N/A
FR70	Links to purchase seeds	User should be able to access links to information on where to purchase seeds for particular plants.	FR07	2	WON'T	0.527	N/A
FR71	View quizzes	User should be able to view quizzes to test their plant knowledge.	FR02	8	WON'T	0.416	MoSCow Rating changed from 'COULD' to 'WON'T'.
FR72	Take a single quiz	User should be able to take quizzes to test their plant knowledge.	FR71	12	WON'T	0.382	MoSCow Rating changed from 'COULD' to 'WON'T'.
FR73	View results from single quiz	User results from taking a quiz should be recorded.	FR72	12	WON'T	1.012	MoSCow Rating changed from 'COULD' to 'WON'T'.
FR74	View results from all quizzes taken	User should be able to view their quiz results.	FR73	12	WON'T	0.685	MoSCow Rating changed from 'COULD' to 'WON'T'.
FR75	Access permissions	Users should be allocated an account type to enable access permissions.	FR01	25	MUST	0.964	N/A
FR76	Admin application overview	Admin users should be able to access a dashboard to monitor application activity.	FR75	20	MUST	0.505	New requirement added.
FR77	View admin user dashboard	Admin users should be able to access a dashboard to view and modify basic user information.	FR75	20	MUST	0.513	MoSCow Rating changed from 'COULD' to 'MUST'.
FR78	Admin toggle user permissions	Admin users should be able to update user permissions, to give or revoke admin access.	FR75	16	MUST	0.654	New requirement added.

FR79	Admin send user password reset email	Admin users should be able to send users a password reset email if they cannot remember their password.	FR75	12	COULD	0.693	New requirement added.
FR80	Admin delete user account	Admin users should be able to delete any user account.	FR75	12	MUST	0.771	New requirement added.
FR81	Admin content management	Admin users should be able to remove user generated content if it violates community guidelines.	FR75	16	SHOULD	0.666	MoSCow Rating changed from 'COULD' to 'SHOULD'.

The functional requirements with a MoSCoW priority of 'MUST' include the account, interactive guide, planner, article, recipe, and wish-list features. This functionality must be included within the project for it to be deemed as a success, while the forum may or may not be developed.

1.1.2: Non-Functional Requirements

ID	Requirement	Description	Dependencies	Risk Score	MoSCoW Rating	Relative Weighting	Modification Description
NR01	Accessibility based on personal ability	The application should be able to be used by people with visual, auditory, physical speech, cognitive and neurological disabilities.	N/A	25	MUST	0.785	N/A
NR02	Accessibility based on device	The application should be able to be used across various device screen sizes.	N/A	20	MUST	0.773	N/A
NR03	Security	The application should store user information in a secure manner.	N/A	25	MUST	0.559	N/A
NR04	Privacy	The application should only utilise user information for necessary tasks to carry out specific functionality.	N/A	25	MUST	0.798	N/A
NR05	Response time	The application should have a fast response time.	N/A	16	MUST	0.631	N/A
NR06	User friendly	The application should use natural language.	N/A	15	MUST	0.534	N/A
NR07	Consistency	The application should have a consistent user interface.	N/A	10	SHOULD	0.487	N/A
NR08	Application aesthetic	The user interface should be aesthetically pleasing.	N/A	10	SHOULD	0.692	N/A
NR09	Notification of success	The application should notify the user of successful actions.	N/A	25	MUST	0.814	N/A
NR10	Notification of failure	The application should notify the user of unsuccessful actions.	N/A	25	MUST	0.594	N/A
NR11	Availability	The application should be available for use 24/7.	N/A	12	COULD	0.739	N/A
NR12	Scalability	The application should be able to adapt to usage requirements, such as number of people using the application at the same time.	N/A	12	SHOULD	0.603	N/A
NR13	General information	Basic information about the application should be available, such as developer and version information.	N/A	12	COULD	1.153	MoSCow Rating changed from ‘SHOULD’ to ‘COULD’.

1.2: Requirements Evolution

Requirements are the cornerstone of application development and have been given great attention throughout this project, given the employment of the Agile-Waterfall Hybrid development methodology. A thorough requirement gathering process occurred at the inception of the project using individual interviews, brainstorming, focus groups and a survey.

At this stage several core ideas emerged, best demonstrated in the survey results (see Figures 1.2.1 and 1.2.2), which aided in growing functional and non-functional requirement lists for the initial prototype.

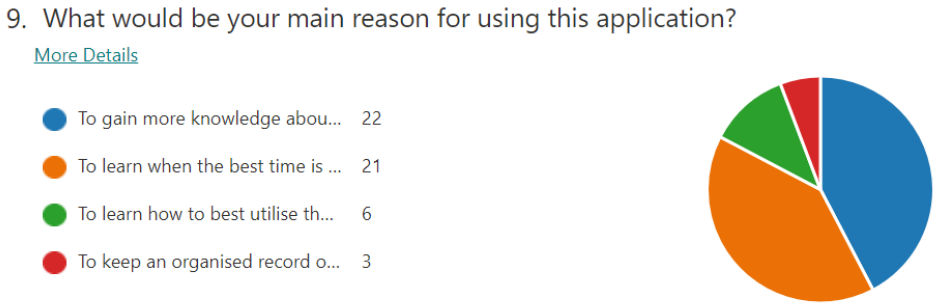


Figure 1.2.1: Results for question nine in the initial requirements gathering survey.

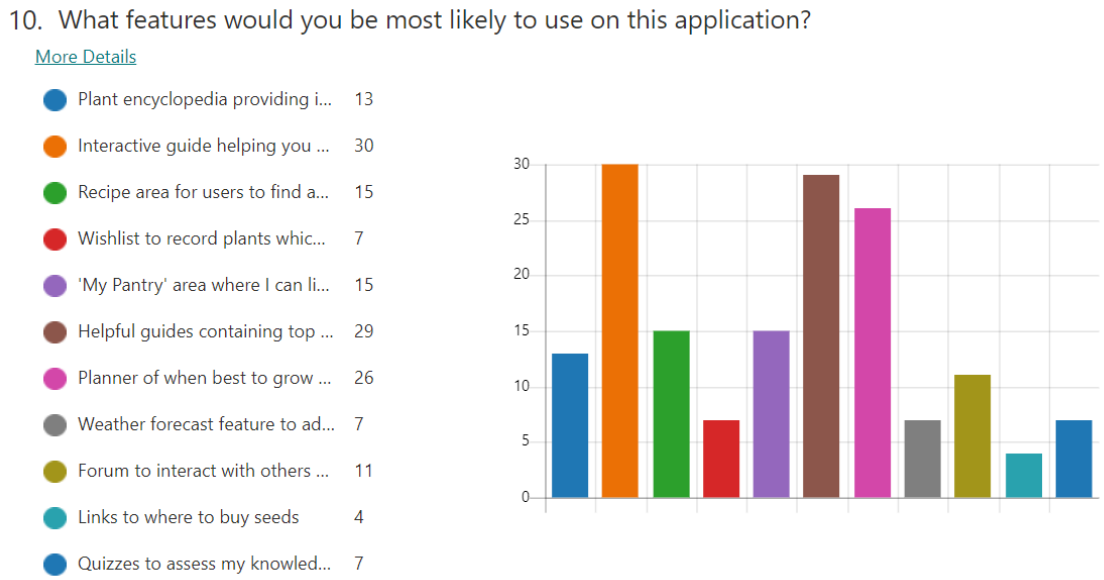


Figure 1.2.2: Results for question ten in the initial requirements gathering survey.

Upon completing the initial prototype, and each subsequent sprint, feedback was acquired using verification and validation methods, discussed further in later sections of this report. In most cases feedback pertained to the visual appearance of the application.

More constraints were added to the functional requirements list, after completing the prototype, relating to administrator and planner functionality (see Tables 1.2.1 and 1.2.2), as this had not been fully explored prior. Feature requirements established before development commenced were exhaustive and did not require attention beyond updating priorities.

ID	Requirement	Description	Dependencies
FR23	View planner for all plants	User should be able to view a planner detailing when to sow and harvest all produce based on growing location.	FR06
FR24	View planner for current plants	User should be able to view a tailored version of the planner based on what they are currently growing.	FR06, FR11
FR25	Planner type filter	User should be able to view a planner of each type of produce.	FR06
FR26	Planner search by name	User should be able to search the planner for produce by name.	FR06

Table 1.2.1: Functional requirements relating to the planner included in the application.

ID	Requirement	Description	Dependencies
FR75	Access permissions	Users should be allocated an account type to enable access permissions.	FR01
FR76	Admin application overview	Admin users should be able to access a dashboard to monitor application activity.	FR75
FR77	View admin user dashboard	Admin users should be able to access a dashboard to view and modify basic user information.	FR75
FR78	Admin toggle user permissions	Admin users should be able to update user permissions, to give or revoke admin access.	FR75
FR79	Admin send user password reset email	Admin users should be able to send users a password reset email if they cannot remember their password.	FR75
FR80	Admin delete user account	Admin users should be able to delete any user account.	FR75
FR81	Admin content management	Admin users should be able to remove user generated content if it violates community guidelines.	FR75

Table 1.2.2: Functional requirements relating to administrator permissions included in the application.

Requirement priorities were updated after the prototype was generated in light of knowledge acquired through feedback and developer experience. For instance, users believed that the plant wish-list was a strong companion feature to the currently growing section and that commenting on various pieces was less important than having a range of features enabling personalisation.

Time management was also a major factor in re-establishing priority, as completion of the prototype and a greater awareness of workload meant that the original priority ratings, while largely accurate, needed modified.

Risk analysis is an important addition to the functional and non-functional requirements tables, and it is significant in relation to the prioritisation strategies. There are 24 functional requirements (see Table 1.2.3) and 8 non-functional requirements (see Table 1.2.5) which have been allocated a high risk score, meaning that these requirement tasks are likely to be executed within the application and would have a great impact if the task fails.

In terms of the functional requirements in Table 1.2.3, the requirements with the highest risk scores have been tackled in the initial prototype and the subsequent sprints as the majority of the tasks have been determined as high priority (see section 1.1.1). The greatest risk scores among these pertain to the ability to log in to the application, generating plant information, the interactive guide, and access permissions. These requirements are therefore largely fundamental to the perceived success of the application to the end users.

ID	Requirement	Risk Score
FR01	Account sign-up	20
FR02	Account login	25
FR03	Account logout	16
FR06	Plant information	25
FR07	View plant encyclopaedia	20
FR10	View single plant	20
FR11	Add plant to currently growing	20
FR12	View currently growing	16
FR15	Add plant to wish-list	20
FR16	View wish-list	16
FR17	Remove plant from wish-list	20
FR18	Open interactive guide	20
FR19	Close interactive guide	25
FR20	Interactive guide parameters	25
FR23	View planner for all plants	15
FR38	Recipe section	15
FR39	View single recipe	20
FR49	Add recipe to favourites	16
FR52	Recipe upload	15
FR75	Access permissions	25
FR76	Admin application overview	20
FR77	View admin user dashboard	20
FR78	Admin toggle user permissions	16
FR81	Admin content management	16

Table 1.2.3: Functional requirements with a high risk score signifying an extreme risk.

Two of the requirements identified in Table 1.2.3 as high risk have been newly introduced to the project requirements list. Mitigations must be considered for these to ensure that these new elements do not have a negative impact on the development process (see Table 1.2.4).

ID	Requirement	Risk Mitigation Strategy
FR76	Admin application overview	Consideration needs to be given to the information retrieved to provide this overview to ensure user privacy and application security. Existing functions should be utilised to check user admin credentials on both the front and back end of the application and the information should be generated during the API call not by the front end to reduce load time.
FR78	Admin toggle user permissions	Existing functions should be utilised to check user admin credentials for executing this functionality. On the front end it should be clear what permission each user has, with modals to prevent accidental changes. Admin users should be able to view separate lists of admin and general users to aid this.

Table 1.2.4: Risk mitigation strategies for the two new functional requirements with high risk scores.

A majority of the non-functional requirements outlined for the project possess high risk scores as well (see Table 1.2.5). Non-functional requirements typically carry great priority as they help to define the overall operation of the developed system. The largest scores relate to the requirements regarding accessibility, application security, and notifying users of task outcomes.

Despite not being task centric, these requirements are very important as they impact whether the system is inclusive and if users trust the application, so observing the mitigation strategies (see Table 1.2.6) has been highly important.

ID	Requirement	Risk Score
NR01	Accessibility based on personal ability	25
NR02	Accessibility based on device	20
NR03	Security	25
NR04	Privacy	25
NR05	Response time	16
NR06	User friendly	15
NR09	Notification of success	25
NR10	Notification of failure	25

Table 1.2.5: Non-functional requirements with a high risk score signifying an extreme risk.

ID	Requirement	Risk Mitigation Strategy
NR01	Accessibility based on personal ability	Consideration for the various types of users, based on elements such as cognitive abilities, physical capabilities, and technological background, must be included from the outset. Insight into how best to execute this can be attained through testing and user research.
NR02	Accessibility based on device	Ensure during the development process that stylings for both desktop and mobile devices are generated, and that they remain consistent for both aspect ratios.
NR03	Security	Security must be considered from the beginning of application development. Access permissions, encryption and authorisation are key factors which should be implemented, and thorough thought should be given to the features required to ensure security.
NR04	Privacy	Only necessary user data should be collected for functionality purposes and to ensure compliance with GDPR, alongside other appropriate laws.
NR05	Response time	It is vital to utilise technologies which support quick response times and to develop succinct code. Assessing the response time of actions and pages loading should be integrated into the testing process, through user testing and tools like Lighthouse.
NR06	User friendly	Thought must be given to the overall aesthetics of the application. Using natural language, establishing a complementary colour palette, and having a clear layout, among other criteria, should be integrated from the start of development.
NR09	Notification of success	Alerting the user of the successful outcome of their interactions with both the front-end and back-end of the application is necessary to ensure user satisfaction. In the front-end success should be denoted by the colour green, while in the back-end clear messaging should be used.
NR10	Notification of failure	Informing the user of erroneous or unsuccessful outcomes of their interactions with both the front-end and back-end of the application is necessary to ensure user satisfaction. In the front-end errors should be denoted by the colour red. Clear messaging should be used in responses from the back end.

Table 1.2.5: Risk mitigation strategies for the non-functional requirements identified as having high risk scores.

1.3: Project Plan Modifications

1.3.1: Work Breakdown Structure

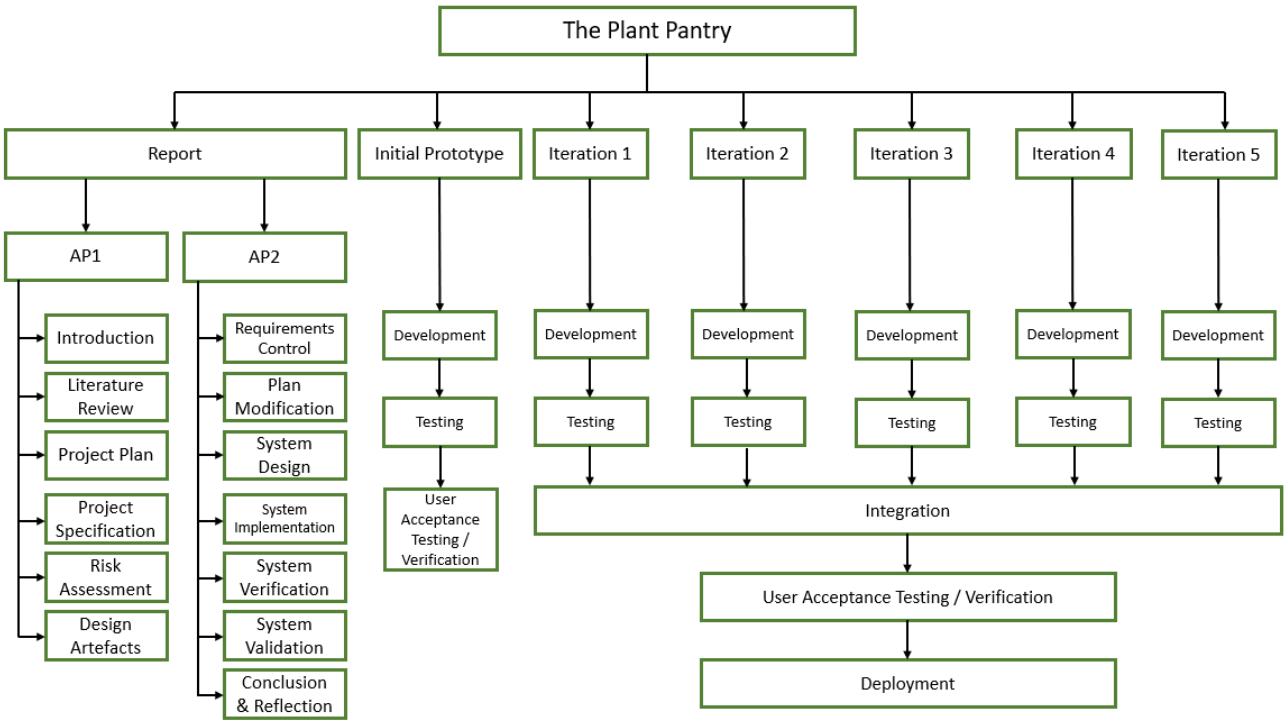


Figure 1.3.1.1: Updated work breakdown structure for the web application including all elements from AP1 and AP2.

1.3.2: Effort Estimation

		Time Estimation			
Action	**Description**	**Start**	**Finish**	**Days**	**Hours**
AP1 Report	Continuous additions to all sections during planning and development of the initial prototype.	23.09.2021	20.12.2021	88	132
Generate Design Artefacts	Creation of design artefacts included in AP1 report and used for prototype generation. Artefacts will include elements such as user personas, user stories, use cases, product backlog, entity-relationship diagrams, database schemas, low-fidelity and high-fidelity prototype designs and architecture diagrams.	31.10.2021	07.11.2021	8	16
Initial Prototype	Initial setup, common component creation and tackling greatest application risks. Account functionality, Plant information API generation, Articles, Planner, and Interactive Guide generated.	08.11.2021	28.11.2021	21	63
AP2 Report	Continuous additions to all sections during sustained development of the application.	10.01.2022	22.04.2022	102	153
Iteration 1	Changes based on user feedback from prior iteration. Plant encyclopaedia created alongside enhancements to features from the initial prototype. This is subject to change based on user feedback of the initial prototype.	17.01.2022	30.01.2022	14	42
Iteration 2	Changes based on user feedback from prior iteration. Currently growing list and plant wish-list added to user account section. Tasks are subject to change based on previous user feedback.	31.01.2022	13.02.2022	14	42
Iteration 3	Changes based on user feedback from prior iteration. Recipes area generated, alongside liked and uploaded recipes components in the user account section. Tasks are subject to change based on previous user feedback.	14.02.2022	27.02.2022	14	42
Iteration 4	Changes based on user feedback from prior iteration. Admin area generated. Tasks are subject to change based on previous user feedback.	28.02.2022	13.03.2022	14	42
Iteration 5	Changes based on user feedback from prior iteration. Focus on refactoring code and ensuring existence of accessibility features are met. Tasks are subject to change based on previous user feedback.	14.03.2022	27.03.2022	14	42
Final UAT Testing	Final feedback session with users to determine if the final version of the project meets user requirements.	28.03.2022	03.04.2022	7	21
Total				296	595

Table 1.3.2.1: Updated effort estimation for the project including all elements from AP1 and AP2.

1.3.3: Gantt Chart

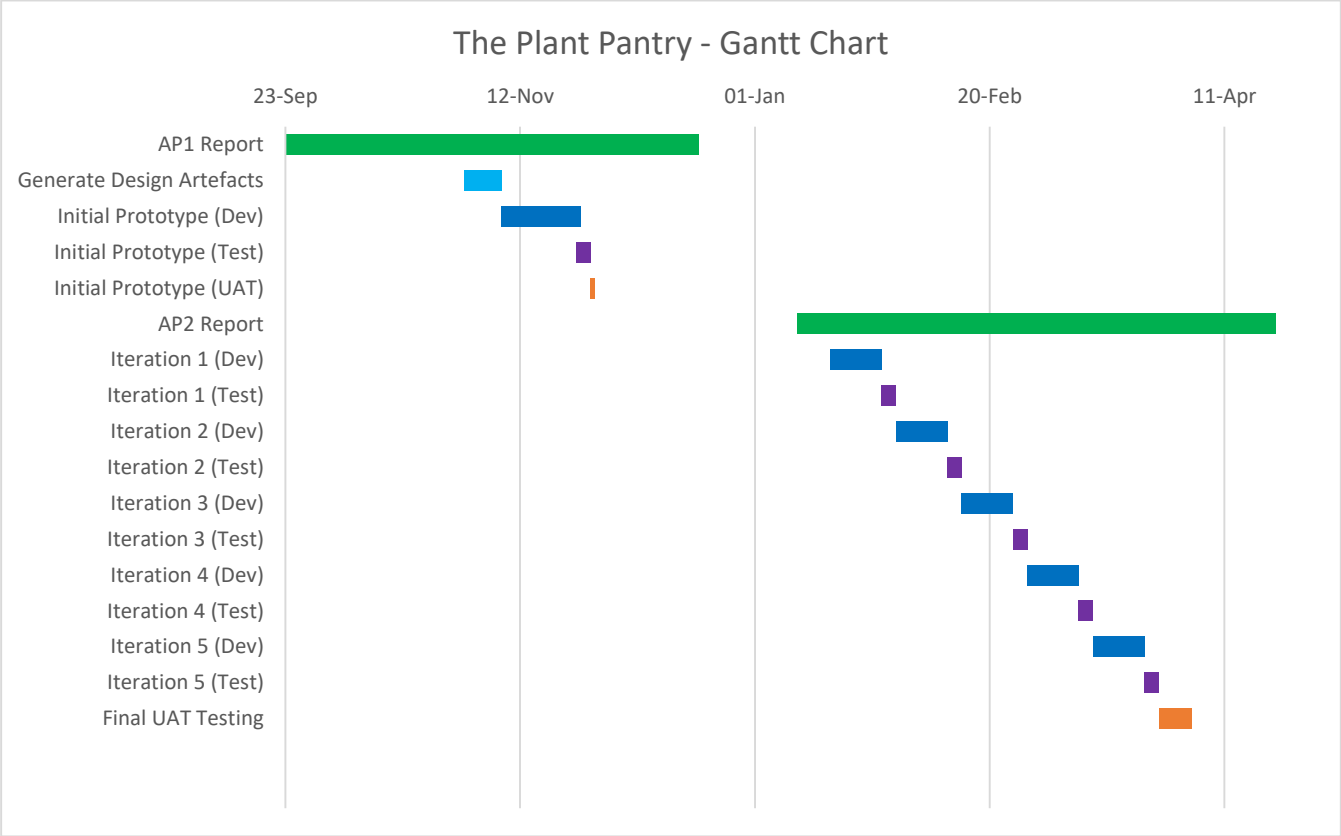


Figure 1.3.3.1: Gantt chart reflecting updated work breakdown and estimation for AP1 and AP2.

2.0: System Design

2.1: System Architecture Diagram

Selecting a strong system architecture was vital in establishing a stable foundation for this project, as all programmed elements of the application depend on this choice. From the outset, the MERN stack has been the favoured approach for a multitude of reasons.

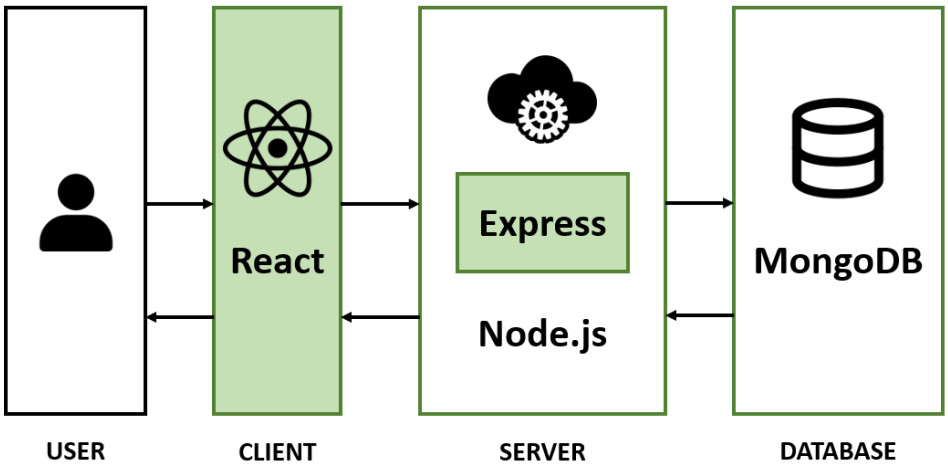


Figure 2.1.1: Diagram of the MERN system design architecture which has been utilised for application development (Modified from: MongoDB, 2022a).

This architecture utilises a Mongo database, with an Express server running in a Node.js environment, and a React front-end (see Figure 2.1.1), making up the MERN acronym. This style of system was chosen due to its popularity and how it encapsulates the entirety of the development cycle with the singular language of JavaScript (Ganguly, 2021; MongoDB, 2022a). It is widely acknowledged that React is a popular JavaScript library for developing large web applications which require superior performance speeds (Aggarwal and Verma, 2018).

Reasoning for this architecture design aligned with how most developers make this important selection: the language popularity, developer awareness of the language, required speed of development, and its overall suitability for the project (Meyerovich and Rabkin, 2013).

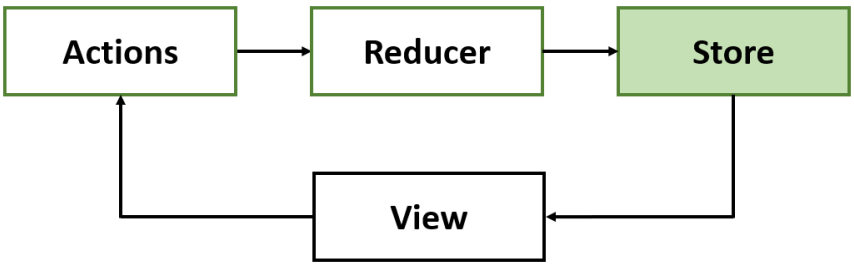


Figure 2.1.2: Diagram of the Redux architecture which has been utilised for application development within the React area of the MERN stack (Modified from: Vinugayathri, 2020).

Within the MERN stack, the end user interacts with the React front-end to make API calls to the Express server which accesses MongoDB. The server then returns obtained results to the Redux store of the application, which is used to update the React front-end, making up the Redux architecture (see Figure 2.1.2).

This is found within the client section illustrated in Figure 2.1.1 and is an advancement from the Flux architecture (Vinugayathri, 2020). Further discussion on the parts of these architectures can be found later in this report.

2.2: Interface Design

Throughout the design and development processes within the generation of this application, extensive consideration has been given to the overall appearance and accessibility.

2.2.1: Interface Wireframes and Prototypes

For the initial prototype, features were established in relation to mitigating two major project risks. These risks were the lack of technical knowledge required for development and the project possessing high amounts of unanticipated complexity. Features developed at this stage included the homepage, planner, articles, and interactive guide, alongside basic account functionality, such as sign-up and login.

Wireframes and prototypes discussed within this section will provide a focused overview on features generated beyond the initial prototype, with the full set of application wireframes and prototypes present within Appendix B. These features include the produce pages, recipe pages, admin area and an updated account section.

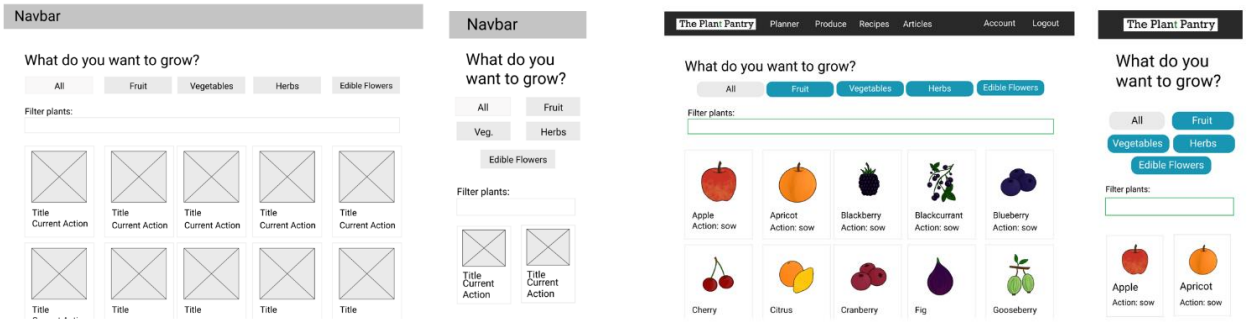


Figure 2.2.1.1: Wireframes and prototypes for desktop and mobile views of the produce page to be included in the application.

There are two main pages within the produce area of the application, the produce home page, and the single produce page, shown in Figures 2.2.1.1 and 2.2.1.2.

In terms of the produce home page (see Figure 2.2.1.1), there are a row of buttons along the top allowing the user to choose the type of produce they would like to view a list of, with the selected type possessing a different background colour. Beneath this follows a text input field, which can be utilised to filter the list of produce presented, and subsequently the produce list. Each produce item is displayed in the form of a card, a reusable component created by the developer, and contains a bespoke image of the item, the item name and the current action required at the time for looking after the plant. Each card acts as a link to the single produce page.

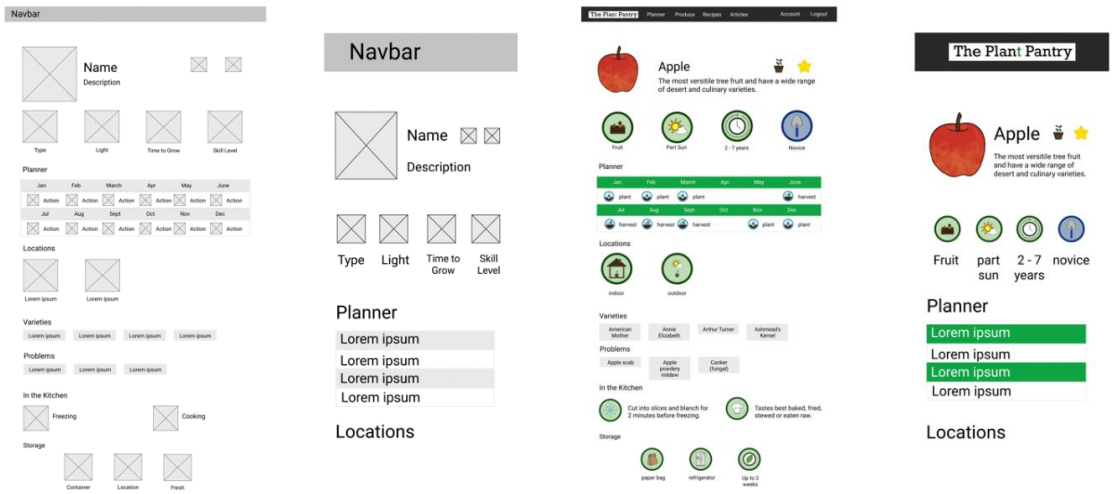


Figure 2.2.1.2: Wireframes and prototypes for desktop and mobile views of the single produce page to be included in the application.

The single produce page (see Figure 2.2.1.2) contains all information for a particular produce item, and the information on this page changes based on the item the user wants to view. Icons are used to represent values also displayed as text, such as the amount of light required and best locations for growing. Beside the produce name, used as the page title, there are two icons representing the ability for the user to add the item to their currently growing list or wish-list. If the item is not in the particular list the icon is a black and white outline, but if the item is contained in the corresponding list the icon is filled in with colour.

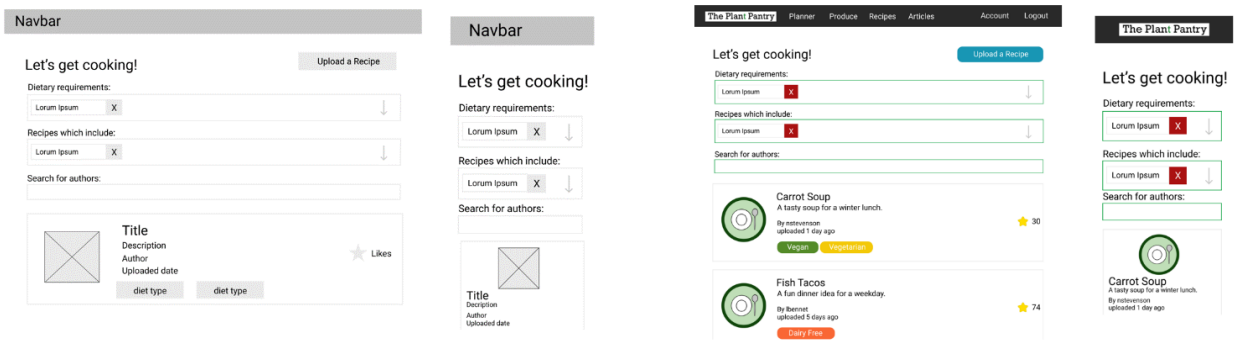


Figure 2.2.1.3: Wireframes and prototypes for desktop and mobile views of the recipes page to be included in the application.

There are three main pages within the recipe area of the application, the recipe home page, the recipe upload page, and the single recipe page, shown in Figures 2.2.1.3, 2.2.1.4 and 2.2.1.5. From the recipe home page (see Figure 2.2.1.3), the user is able to access the upload page through the button in the top right. There are then two select input fields, created by the developer, enabling the user to filter the recipes based on dietary requirements and desired produce to be included in recipes. In addition, there is a text input field for searching the recipes by title or author name. The recipe list follows this input section, with each recipe represented as a card, a different reusable component than that used for the produce home page which performs similar functionality. Key recipe information, such as the title, description, author name, and uploaded date is contained within the card, alongside the dietary requirement suitability. Each requirement represented by a particular colour of tag and the likes count can also be viewed at this stage.

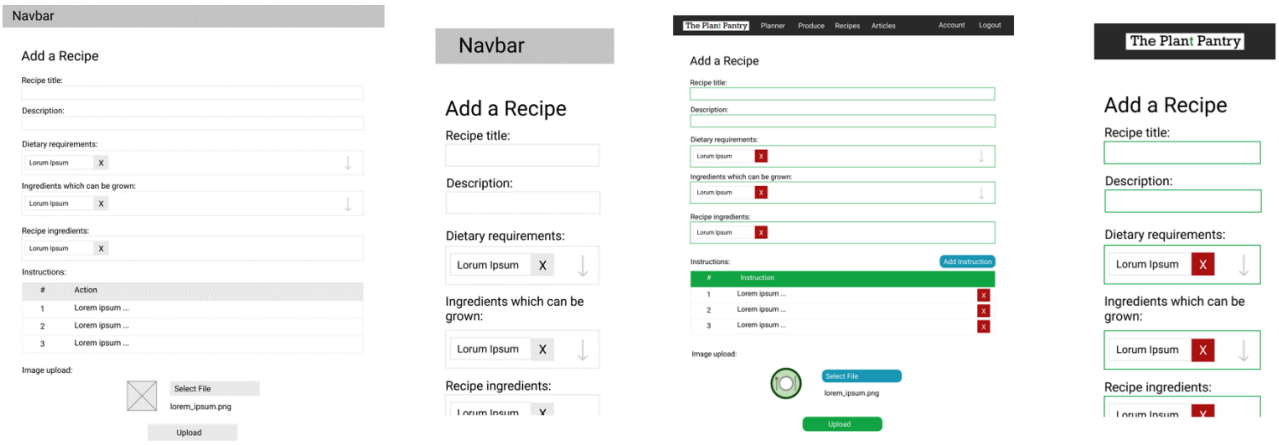


Figure 2.2.1.4: Wireframes and prototypes for desktop and mobile views of the upload recipe page to be included in the application.

As for the recipe upload page (see Figure 2.2.1.4), it consists of a form with multiple input fields of varying types. Title and description are simplistic text input fields, dietary requirements and ingredients that can be grown are multiple select input fields, the ingredients value is a multiple text input field, and the image upload is a file input field. The multiple select input and multiple text input fields are components created by the developer allowing numerous value entry, which are interpreted as arrays. The instructions section is also a uniquely created component for the application, with the appearance of a table. The user is able to add new rows to the table, with each row being allocated a number and an input field which can be updated to represent a step in the cooking process. Every row can be dragged to a different location, to change the order of instructions, and can be removed. This same form is used for adding and editing a recipe on the application.

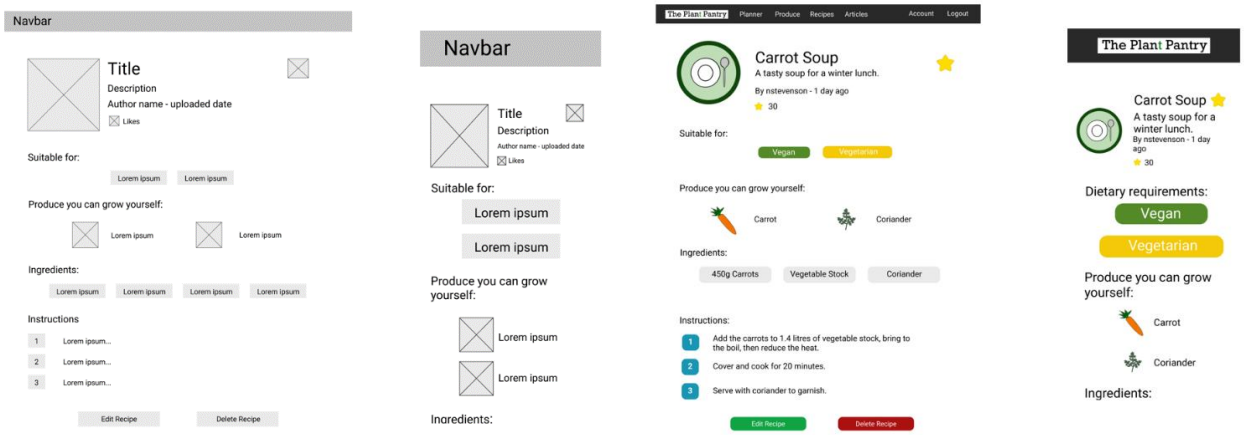


Figure 2.2.1.5: Wireframes and prototypes for desktop and mobile views of the single recipe page to be included in the application.

The last area within the recipe section is the single recipe page (see Figure 2.2.1.5). This contains all information regarding an individual recipe, using the requirement colour-coded tags and the produce images. The user can click the star icon at the top of the page to add or remove the chosen recipe from their favourites list, like with the single produce page if the recipe is in the list the star is colourful and if not, it is a black and white outline. The buttons at the bottom are present depending on permissions. If the user is the author, they have access to the edit and delete buttons, while users with administrative permissions can also see the delete button. These buttons are not visible otherwise.

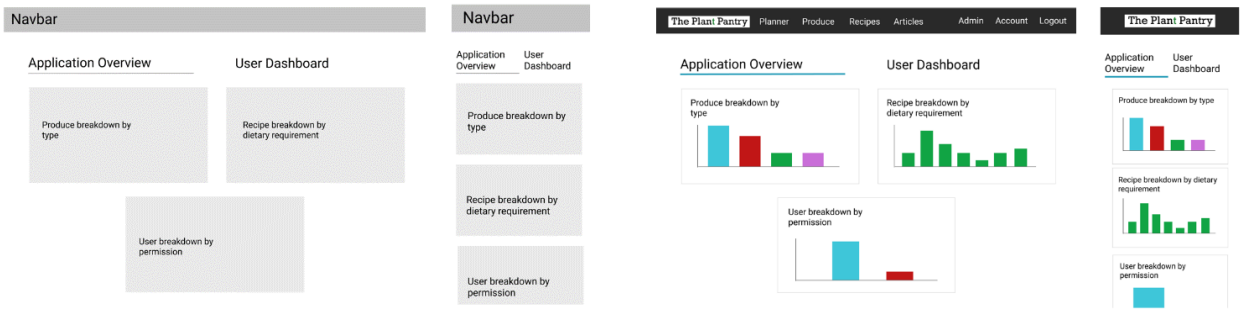


Figure 2.2.1.6: Wireframes and prototypes for desktop and mobile views of the application overview within the admin section to be included in the application.

If a user has administrative privileges, then they are given access to the admin area of the application, permitting them to view an overview of the application and a user dashboard. The application overview (see Figure 2.2.1.6) consists of three charts: the produce breakdown by type, the recipe breakdown by dietary requirement and the user breakdown by permission. The produce chart counts the number of fruit, vegetables, herbs, and edible flowers recorded in the application, the recipe chart shows the number of recipes identified as suitable for the different diets recorded in the system, while the user chart presents the number of general and admin users within the application. Using this visual representation is a clear way of presenting this information, and proportionally displaying the values, for example there be approximately double the recipes suitable for vegetarians than there are for vegans.

The user dashboard (see Figure B.2.24 and B.4.24 in Appendix B) is similar in style to the components used in the Planting Profile, where the user can manage their account information.

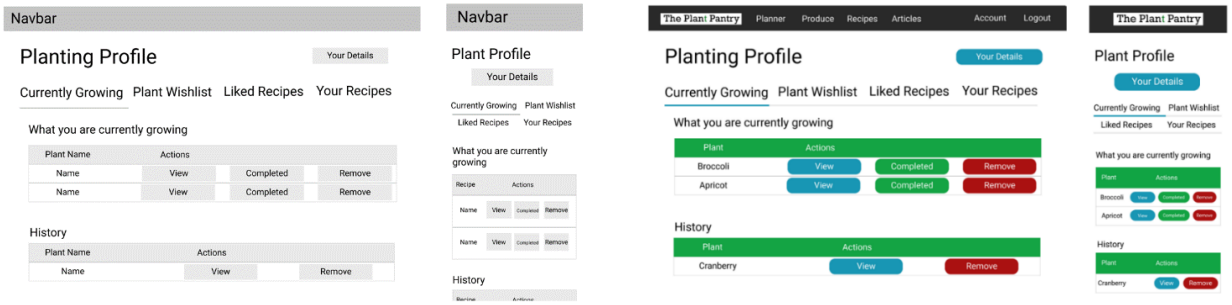


Figure 2.2.1.7: Wireframes and prototypes for desktop and mobile views of the currently growing list within the account section to be included in the application.

The currently growing list (see Figure 2.2.1.7) is largely reflective of the presentation of the account components used within the application, with some minor differences depending on the functionality required. In the case of this list there are two portions, the current and history sections. Within the current section a user can click the corresponding button in the table row to view the single produce page, mark the item as finished growing and delete the item from the list altogether. Marking a produce item as complete moves it to the history section.

From the wish-list (see Appendix B.2.31 and B.4.31) users can move items to the currently growing list. Within the Planting Profile users can also view the recipes they have liked (see Appendix B.2.32 and B.4.32) and the recipes they have uploaded (see Appendix B.2.33 and B.4.33), alongside updating their account information (see Appendix B.2.29 and B.4.29).

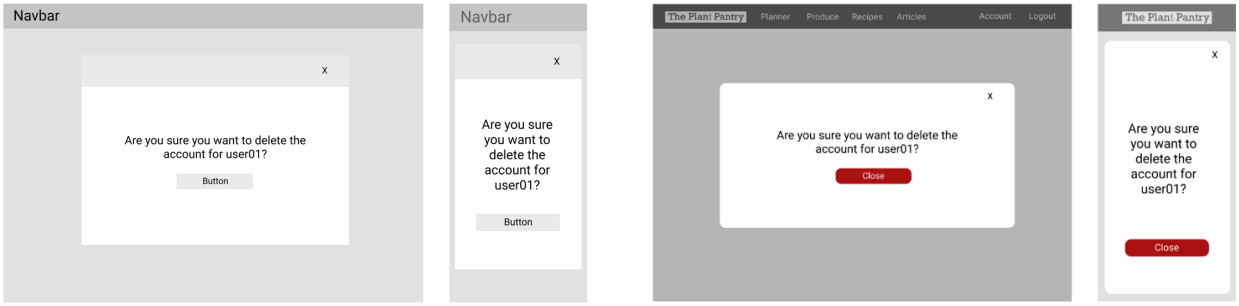


Figure 2.2.1.8: Wireframes and prototypes for desktop and mobile views of the admin delete user account modal within the admin section to be included in the application.

Modals are used throughout the application to fulfil various purposes, such as indicating that an event has occurred or asking the user to confirm an action. These modals overlay the information currently presented to the user, drawing user attention, and can always be closed by clicking the ‘x’ icon in the top right corner or clicking the modal background. For instance, when an admin user clicks to delete a particular user account from the user dashboard the modal shown in Figure 2.2.1.8 appears asking them to confirm this action, preventing accidental account deletion.

2.2.2: HCI and Usability Considerations

A fundamental part of the design process is considering the human computer interaction (HCI) a user has with the developed system to provide a superior user experience. Carroll notes how HCI has evolved as a discipline over the years (Carroll, 2012) to embody a mechanism which provides "explicit consideration of the needs, abilities, and preferences of their ultimate users" (Carroll, 1997, pp.67). The idea of "needs" and "abilities" refers to the accessibility concerns, while the concept of "preferences" relates to human centred design (HCD) principles, foundational aspects of HCI.

There are a range of existing usability and HCD standards which were considered during the design and development process for this application. These best practices share some common elements which can be identified upon evaluation and are reflective of strong professional practice. The most prominent existing principles for interface design are authored by Schneiderman (see Figure 2.2.2.1), Nielsen (see Figure 2.2.2.2), Krug (see Figure 2.2.2.3), and Babich (see Figure 2.2.2.4).

Figure 2.2.2.1: Schneiderman's Eight Golden Rules for Interface Design (Modified from: Schneiderman, 2016).

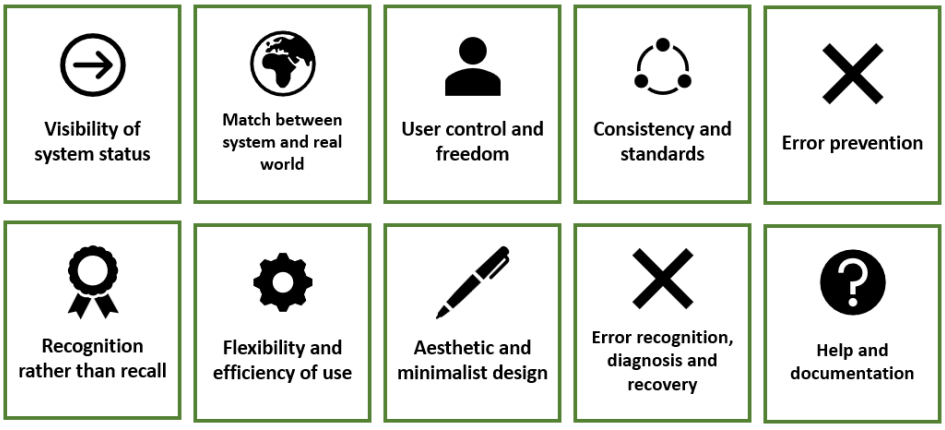


Figure 2.2.2.2: Nielsen's Ten Usability Heuristics for Interface Design (Modified from: Nielsen, 1994a).

Pages should be designed for scanning, not reading

- Create a clear visual hierarchy
- Take advantage of conventions
- Break pages up into clearly defined areas
- Make it obvious what is clickable
- Minimise noise

Users prefer not to have to think when making choices.
Simplistic navigation is paramount.

Testing is an important part of the process:

- It should be done throughout
- Test on more than just the target audience

Accessibility must be considered, and focus should be given to the areas with most impact.

Figure 2.2.2.3: Overview of Krug's laws of usability (Derived from: Krug, 2005).

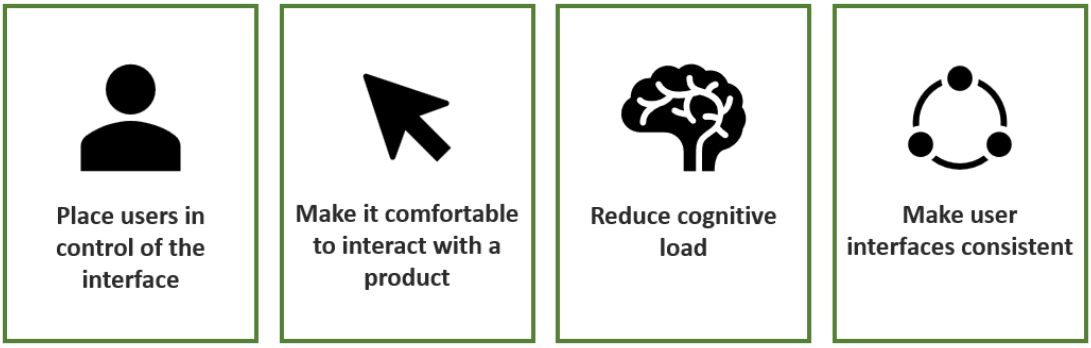


Figure 2.2.2.4: Babich's Four Golden Rules of user interface design (Modified from: Babich, 2019).

The rules dictated by Babich (see Figure 2.2.2.4) are inspired by Nielsen and Schneiderman and are extremely broad. Schneiderman's guidelines (see Figure 2.2.2.1), have a lot of crossovers for having so few rules, for instance reversal of actions and errors, and feedback and closure. For these reasons, the application will be closely aligned with Nielsen (see Figure 2.2.2.2) and Krug's (see Figure 2.2.2.3) standards. Nielsen has the greatest number of criteria out of all standards investigated, covering a variety of elements, and this level of specificity lends itself to developer accountability. In contrast to Nielsen, Krug's assertions have a more emotional perspective, which is highly valuable and connects well with other best practices regarding design (see Figures 2.2.2.5, 2.2.2.6 and 2.2.2.7).

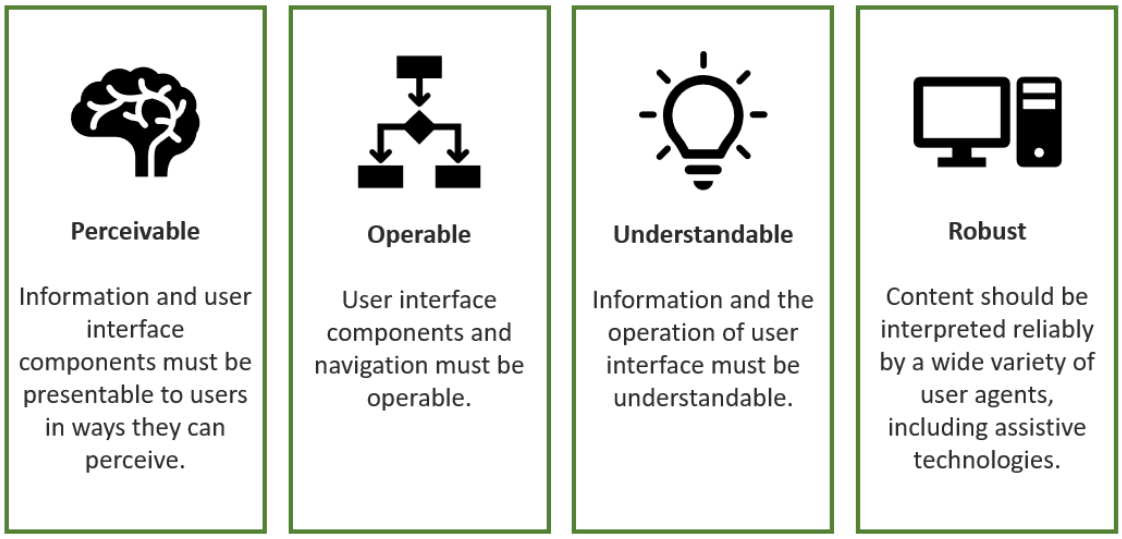


Figure 2.2.2.5: The principles of accessibility as defined in WCAG 2.1 (W3C, 2018).

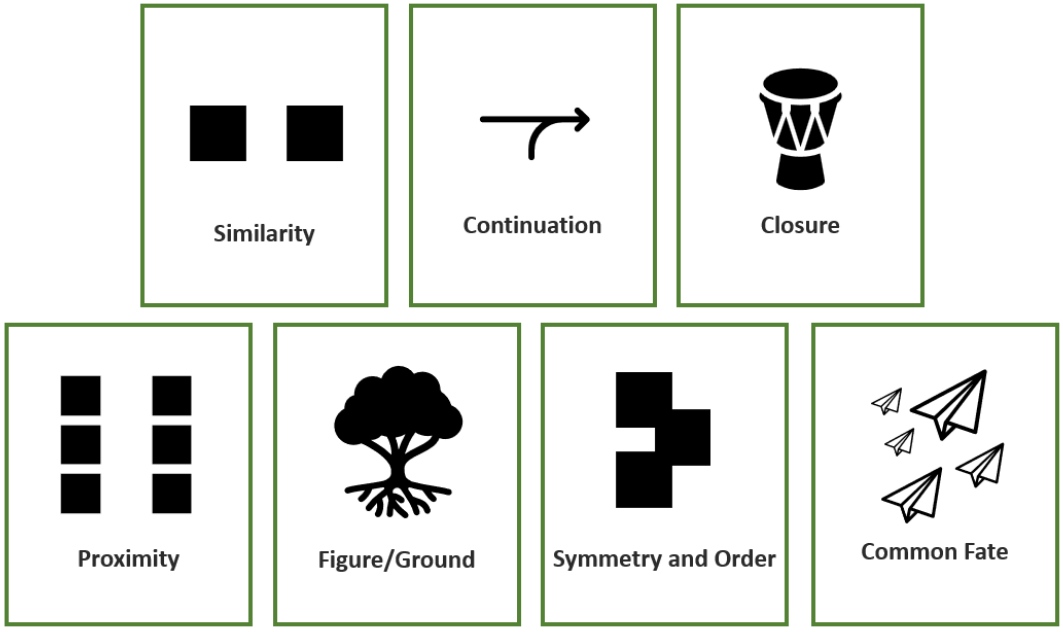


Figure 2.2.2.6: The Gestalt Principles of design (Modified from: Chapman, 2022).

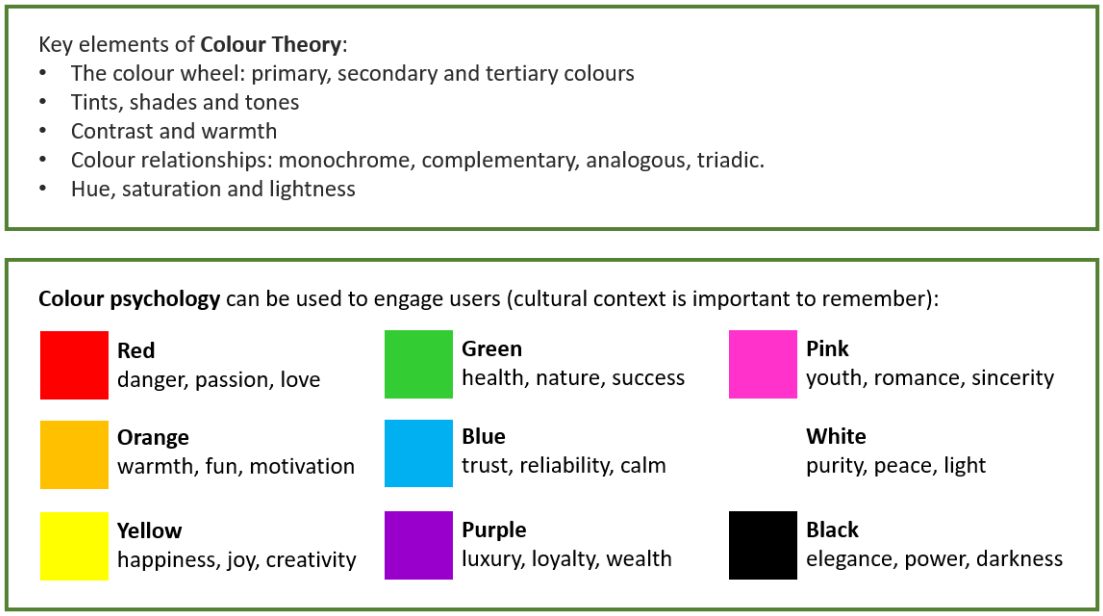


Figure 2.2.2.7: The central principles of colour theory and colour psychology outlined by Corrigan (2022) and Khazanova (2022).

There are three further areas considered within the development of the application: the Gestalt principles, the WCAG 2.1 guidelines, and the core ideas of colour theory and psychology. The WCAG (Web Content Accessibility Guidelines) are an international standard established to make web content accessible to people with a wide range of disabilities (W3C, 2018), with its guidelines divided into four key areas (see Figure 2.2.2.5) to ensure compliance. The most recent version of this is 2.1.

While the WCAG 2.1 strategies are solely concerned with accessibility, the policies surrounding the use of colour and the Gestalt principles are strongly focused on application aesthetics. The Gestalt principles, largely originating from Wertheimer's proposal in 1923, concentrate on the presentation and alignment of stationary and dynamic items (Chapman, 2022; Wertheimer, 2012), subsequently making the web application more user friendly. Informed use of colour is a key component of appealing to the users of the application, using colour psychology to emote and the concepts of colour theory to establish a vibrant colour palette (see Figure 2.2.2.7) across the project.

2.3: Data Support Design

Thorough attention has been dedicated to devising and implementing strong application security, alongside data validation which attributes to this. This security and validation is paramount in ensuring users have confidence in the system.

2.3.1: Data Security and Validation Considerations

Great consideration was given to establishing a secure product, particularly in regard to managing application data, and this was a central element within the development process.

Encryption of Variables

The secure-env npm package was valuable in the encryption of three important application variables: the MongoDB connection string, the JWT secret, and the password to the application email address used for sending password reset requests. This package is used to create an encrypted .env file through employing a user defined secret, which helps to prevent against npm package attacks, while using the AES 256 encryption standard (npm, 2022a).

Securing these variables in this way is vital due to the use of version control for maintaining and managing the application created. Storing this information in plain text would be a great security risk in the event of a system breach, which is why encryption was utilised.

User Permission Checks

In addition to authorisation checks enacted upon the reception of every API request, some endpoints also enforce permission checks. There are two types of checks that may be imposed: an admin check (see Figure 2.3.1.1), to establish if the user has administrative permissions, or a username check.

A username check is carried out in instances where deletions or updates are being made, for instance to a recipe uploaded as only the user who created the recipe should be able to edit or remove it. As for the admin checks, only users with administrative permissions should be allocated overarching privileges across the application, such as managing user permissions and viewing overall application information.

```
10. function checkAdmin(req) {
11.   let authVal = req.rawHeaders.indexOf("Authorization");
12.   let token = req.rawHeaders[authVal + 1];
13.   token = token.replace("Bearer ", "");
14.   let decodedToken = jwt.decode(token);
15.
16.   if (!decodedToken.admin) {
17.     return false;
18.   } else {
19.     return true;
20.   }
21. }
```

Figure 2.3.1.1: Lines 10 - 21 of the api-router.js file within the server directory. This function decodes the token send in the Authorization header of an API request and returns whether the user is an administrator.

Password Encryption

Secure password management is a fundamental part of creating any application requiring an account. The bcryptjs npm package is used to perform this when a user makes a new account or updates their existing information. The encrypted password is stored within MongoDB and checked against the input provided by the user upon login. A salt is provided by the developer for hashing the password supplied by the user, and increasing this value aids in making values impervious to brute-force attacks (npm, 2022b).

Only Returning Required Data

Key elements of the data protection process includes is only disclosing information when necessary and only providing required information. Information returned from API calls is either shown on the front-end of the application or is necessary to perform functionality, for example keeping account information in the Redux store which is used for making additional requests.

This is handled by projections used in the collection methods executed when accessing data held in the MongoDB cluster (see Figure 2.3.1.2). In addition to providing security, using projections on API requests can improve performance as returning smaller amounts of information is faster.

```
779. usersCollection
780.  .find({}, { username: 1, email: 1, admin: 1 })
781.  .toArray((err, result) => {
782.   if (result) {
783.         return res.status(200).json({ result });
784.       } else {
785.         return res
786.           .status(404)
787.           .json({ general: "Error making this request." });
788.       }
789.     });
```

Figure 2.3.1.2: Lines 779 – 789 of the api-router.js file within the server directory of the project. This is within the API call for acquiring a user list for the user dashboard within the admin section. Only the username, email and admin fields are necessary to be returned and this is declared in the projection on line 780.

Front-end and Back-end Data Validation

This form of validation occurs in both the front and back end of the project. In terms of the front-end, there are checks within component submit functions, called when a button is clicked, which return field specific messages if there are issues.

Such problems include empty fields and user input not matching regular expressions, appearing in red text underneath the corresponding fields (see Figure 2.3.1.3). Errors returned from the back end are also typically expressed in red text and may be input or request specific, for example there may be an issue retrieving information. Success messages are indicated with green text.

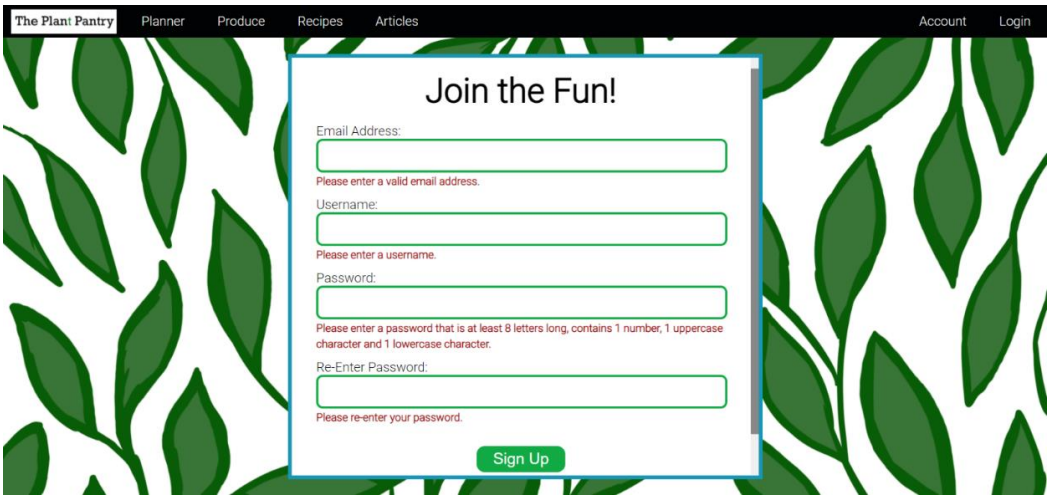


Figure 2.3.1.3: Screenshot of the SignUp application component when the 'Sign Up' button is clicked without any information being added to input fields. All fields are valid within this form so the errors highlighting this are displayed in red text beneath the corresponding field.

It is necessary that input checks are also present in the back end of the application in the event that requests are being made directly to the API and not through the application's user interface. These errors are reflected in the HTTP status code and message returned (see Figure 2.3.1.4). The most common error status used in the application is a 500 HTTP status, a generic error response used when requests cannot be made for other reasons.

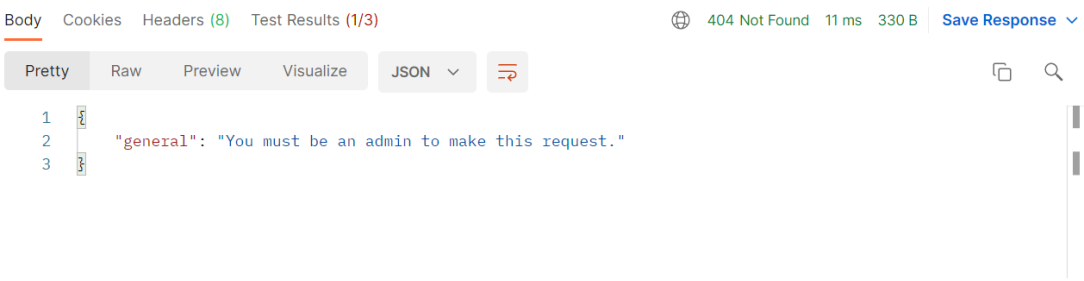


Figure 2.3.1.4: Screenshot of the response when a request is made to retrieve a list of all user accounts within the application when the user does not have administrative permissions. This is conveyed within the response message and the 404 HTTP status.

2.3.2: Data Structure Designs

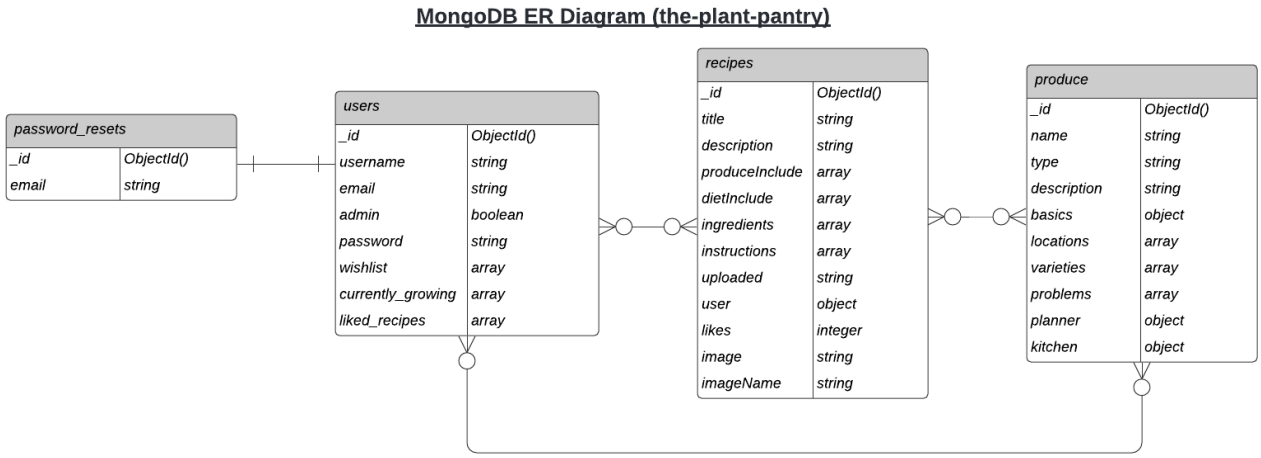


Figure 2.3.2.1: The ER Diagram for the web application developed using MongoDB, which contains four collections.

There are four collections held within the MongoDB cluster for the application: password_resets, users, recipes and produce (see Figure 2.3.2.1). The password_resets collection is utilised when a user has forgotten their password and a request has been made to change it. It contains a unique ID and holds the email of the user making the request. A check is carried out to ensure that the email exists for an account in the system before the record is made in the collection, and one user can make one reset request.

The users collection contains information for all end users of the application who have created an account. It contains ID, username, email, and password fields, alongside an admin field used to mark the user access level. The three array fields of wishlist, currently_growing, and liked_recipes are updated by users using information from the recipes and produce collections. One user can create many recipes, or none, and a single recipe is generated by one user. The liked_recipes field can contain zero or many recipes while the wishlist and currently_growing fields can have zero or many produce items.

A single recipe must have at least one produce item within the produceInclude array, and a single produce item can be in zero or more recipes.

This ER diagram has evolved from the one created when developing the initial prototype due to greater understanding or document databases, such as MongoDB, and the changing environment of this project, as some requirements were not implemented due to low priority. JSON schemas for these collections identified are outlined below, providing greater detail into particular areas of each collection, such as the object fields.

```
{
        "_id": ObjectId(),
        "email": string
}
```

Figure 2.3.2.2: JSON schema outlining information held for each password reset request made. The ID value matches the ID of the user who made the request, so unlike other records is not automatically generated on creation, instead it is provided.

Records from the password_reset collection (see Figure 2.3.2.2) are created from the Login and UserDashboard components within the React front-end of the application and interaction with the API endpoints interacting with this collection do not require the user to be logged in.

```
{
        "_id": ObjectId(),
        "username": string,
        "email": string,
        "admin": boolean,
        "password": string,
        "wishlist": array,
        "currently_growing": array,
        "liked_recipes": array,
}
```

Figure 2.3.2.3: JSON schema outlining information held for each user. The 'wishlist' array contains only string values while the 'currently_growing' and 'liked_recipes' arrays contain objects (see Figures 2.3.2.4 and 2.3.2.5). The ID value is automatically generated upon record creation.

Records from the users collection (see Figure 2.3.2.3) are created from the SignUp component within the React front-end. These records are used throughout most features within the application and the token generated and returned to a user upon successful login is based on the username and admin fields.

```
{
        "name": string,
        "completed": boolean
}
```

Figure 2.3.2.4: JSON schema outlining information held for each array item within the 'currently_growing' list in a user record (see Figure 2.3.2.3).

```
{
        "_id": ObjectId(),
        "title": string,
        "completed": boolean
}
```

Figure 2.3.2.5: JSON schema outlining information held for each array item within the 'liked_recipes' list in a user record (see Figure 2.3.2.3). The ID value is pulled from the record of the specific recipe to be included.

The wishlist, currently_growing, and liked_recipes fields (see Figures 2.3.2.3, 2.3.2.4 and 2.3.2.5) are used in the SingleProduce, SingleRecipe, Wishlist, CurrentlyGrowing and LikedRecipes React components

```
{
        "_id": ObjectId(),
        "title": string,
        "description": string,
        "produceInclude": array,
        "dietInclude": array,
        "ingredients": array,
        "instructions": array,
        "uploaded": string,
        "user": {
                "_id": ObjectId(),
                "username": string
        },
        "likes": string,
        "image": string,
        "imageName": string,
}
```

Figure 2.3.2.6: JSON schema outlining information held for each recipe. The 'produceInclude', 'dietInclude', 'ingredients' and 'instructions' arrays contain only string values. The ID value is automatically generated upon record creation and the ID value in the user object is pulled from the record of the specific user who created the entry.

A record can be added to the recipes collection (see Figure 2.3.2.6) from the RecipeForm component, and in addition can be updated from this component, within the React front-end. The RecipesHome, SingleRecipe, RecipeForm, LikedRecipes and YourRecipes components utilise this collection.

A seeder function exists within the server directory to populate the collection with five sample records derived from White (2012). All communications with endpoints interacting with this collection require the user to be signed in.

```
{
        "_id": ObjectId(),
        "name": string,
        "type": string,
        "description": string,
        "basics": {
        "light": string,
        "time_to_grow": string,
        "skill_level": string
        },
        "locations": array,
        "varieties": array,
        "problems": array,
        "planner": {
                "Jan": string or null,
                "Feb": string or null,
                "Mar": string or null,
                "Apr": string or null,
                "May": string or null,
```

```
                "Jun": string or null,
                "Jul": string or null,
                "Aug": string or null,
                "Sep": string or null,
                "Oct": string or null,
                "Nov": string or null,
                "Dec": string or null
        },
        "kitchen": {
                "freezing": string,
                "storage": {
                        "container": string,
                        "location": string,
                        "fresh": string
                },
                "cooking": string
        }
}
```

Figure 2.3.2.7: JSON schema outlining information held for each plant of the types of fruit, vegetable, herb, and edible flower. The 'locations', 'varieties' and 'problems' arrays contain only string values. The ID value is automatically generated upon record creation.

Records are only added to the produce collection through the seeder function, contained within the server directory of the project. The function inserts a dataset created by the developer, using information from Pollock (2002), Hessayon (1995 and 1997), and Allaway (2017), containing 132 plants. These plants consist of fruit, vegetables, herbs, and edible flowers which users will be inspired to grow through application use (see Figure 2.3.2.7).

This data is central to the functionality of the entirety of the project and all interactions with endpoints using this collection require the user to be logged into their account.

2.4: User Interaction Design

2.4.1: Use Case Diagrams

To ensure clarity, multiple use case diagrams have been illustrated to represent the entirety of the application instead of making one overarching diagram. The diagrams have been divided based on features developed (see Table 2.4.1.1), which aids in the grouping of requirements (see Appendix A).

Use Case ID	Features included	Related figure
1	Planner	Figure 2.4.1.1
2	Produce	Figure 2.4.1.2
3	Interactive Guide	Figure 2.4.1.3
4	Recipes	Figure 2.4.1.4
5	Articles	Figure 2.4.1.5
6	Admin	Figure 2.4.1.6
7	Account	Figure 2.4.1.7

Table 2.4.1.1: Table outlining which features are included in each use case diagram within this section. An ID has been allocated to each figure, utilised in requirement descriptions contained in Appendix A.

It is important to note that users with admin permissions possess access to all the functionality that a general user has in addition to executing other tasks. This is reflected in the use case diagrams as the icons representing the general and admin actors being connected by a solid arrow, the admin user flows into the general user (see Figures 2.1.4.1, 2.1.4.2, 2.1.4.3, 2.1.4.4, 2.1.4.5 and 2.4.1.6). The character representing the general user is then used to connect to the use cases.

Within the diagrams there are two types of dashed arrows, include and extend. The 'include' arrows denote functionality which is mandatory for inclusion to execute a particular use case, while the 'extend' arrows signify functionality which may be executed within a use case (Creately, 2021).

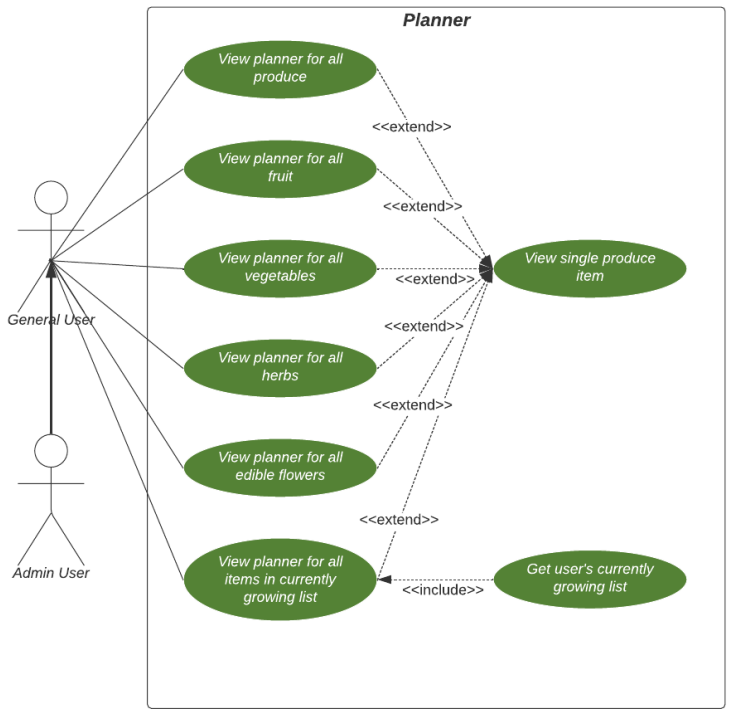


Figure 2.4.1.1: Use case diagram outlining functionality for the planner feature within the application.

The planner can be utilised in a range of ways (see Figure 2.4.1.1) and is accessible to both admin and general users. There are six views of the planner, which users can select from through applying a filter using buttons at the top of the planner page. The list displayed can be controlled to show either all produce, fruit, vegetables, herbs, edible flowers, or all produce items stored within the currently growing list of the user logged into the system. From the list presented in the planner, the user is able to navigate directly to the page containing all information for a specific produce item, indicated with the extend arrows, but this is optional functionality. Obtaining the user's currently growing list is necessary for viewing produce items within the planner that are contained in this list, which is conveyed with the include arrow in Figure 2.4.1.1.

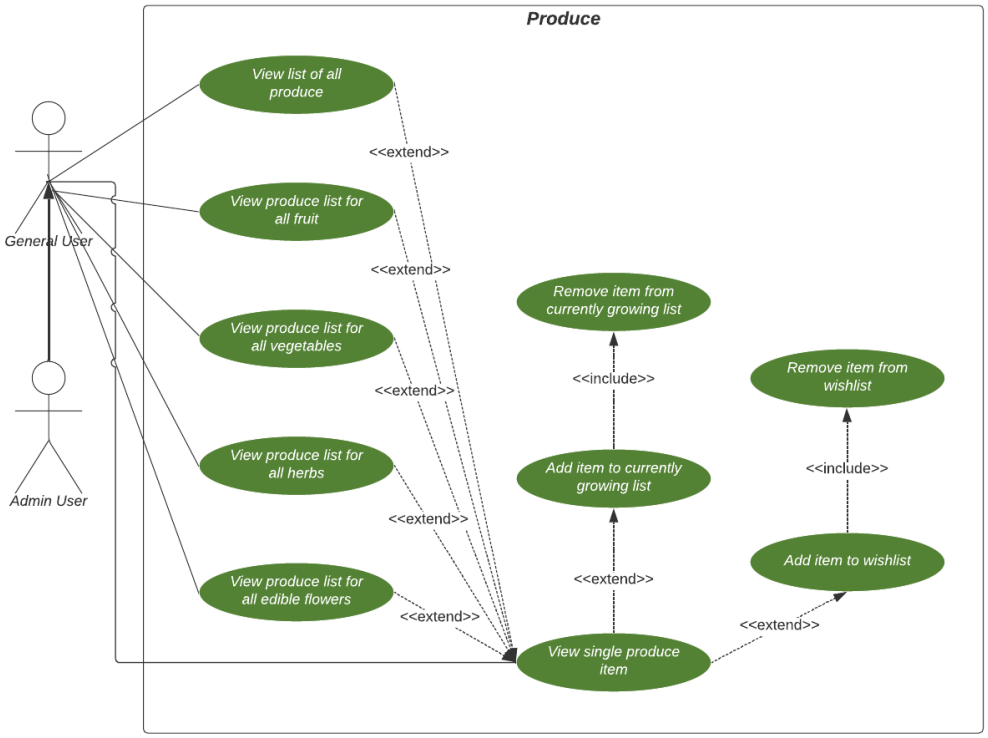


Figure 2.4.1.2: Use case diagram outlining functionality for the produce feature within the application.

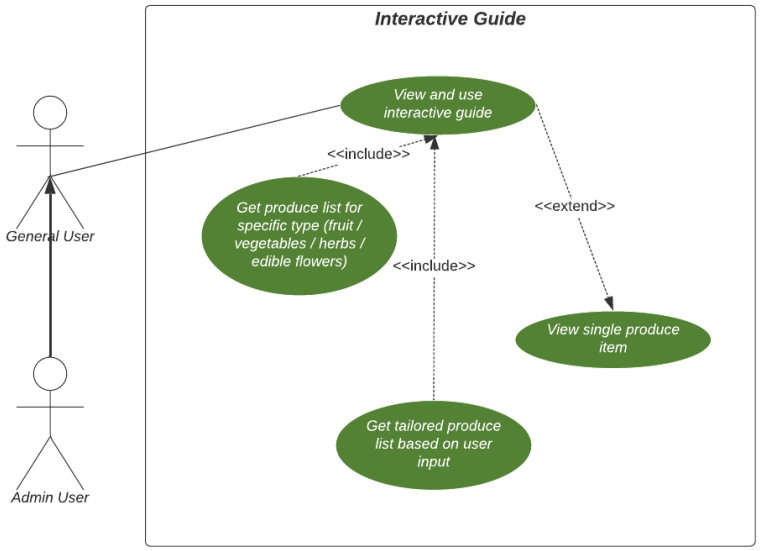


Figure 2.4.1.3: Use case diagram outlining functionality for the interactive guide feature within the application.

The produce feature is a cornerstone of the application and is a cornerstone of the overall functionality (see Figure 2.4.1.2). On the produce homepage the user can view a list of all produce items and can additionally filter this list by fruit, vegetables, herbs, and edible flowers. From here, navigation to the single produce item page is possible, which can also be done from the URL, shown by the extend arrows. The single produce page contains all information for a particular item and the user is able to add these to their currently growing list and wish-list, with the ability to remove them provided they already exist in the respective lists.

The interactive guide makes use of elements of produce functionality. In order to use the guide, the capability to retrieve a list of all produce, matching a specific type, and to obtain a tailored produce list, based on user input from the guide, is required as indicated in the diagram with the include arrows (see Figure 2.4.1.3). At the end of the guide, once the results have been generated, the user may choose to navigate to the single produce page for a particular item if desired.

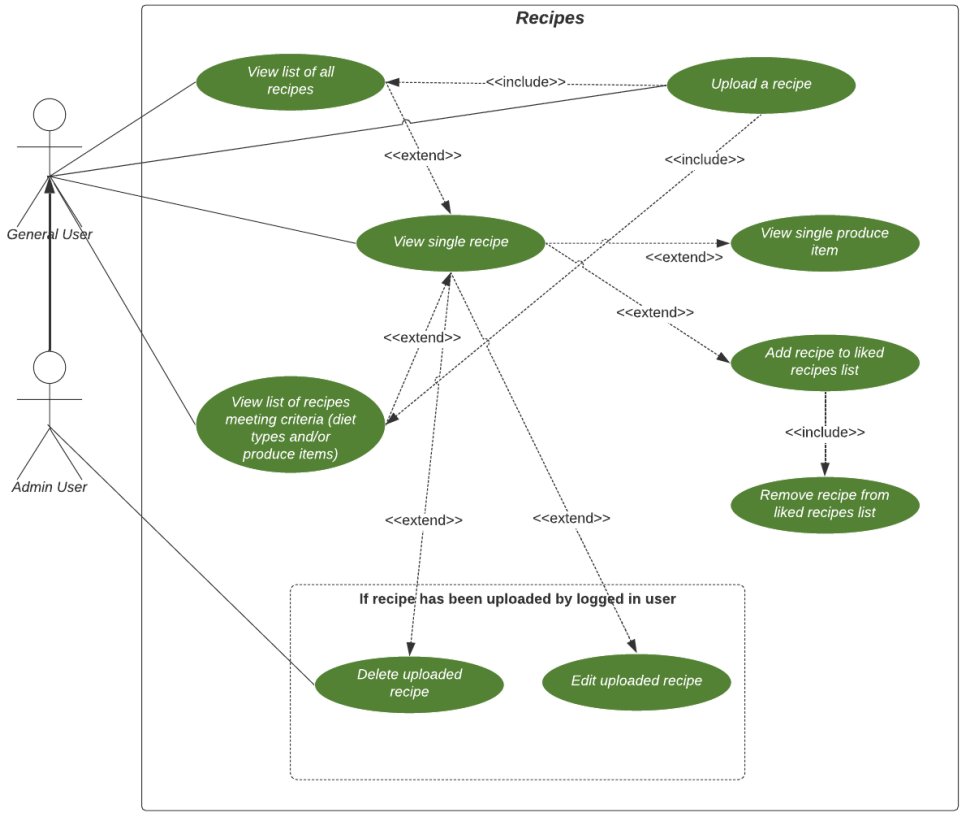


Figure 2.4.1.4: Use case diagram outlining functionality for the recipes feature within the application.

The recipes area in another key piece of the application, containing a wide range of actions (see Figure 2.4.1.4). A list of all recipes can be viewed and filtered, by dietary requirement and ingredient, and the user can navigate to the single recipe page from the recipe homepage, depicted with the extend arrows. Uploading recipes is required for these elements to operate. From a single recipe, a user can navigate to view a single produce item if it is included in the recipe as an ingredient and add a recipe to their liked recipes list, being able to remove it if it is already included. If the user logged in is viewing a single recipe and they are its author, they are able to edit and delete it, and admin users can delete any recipe in the application.

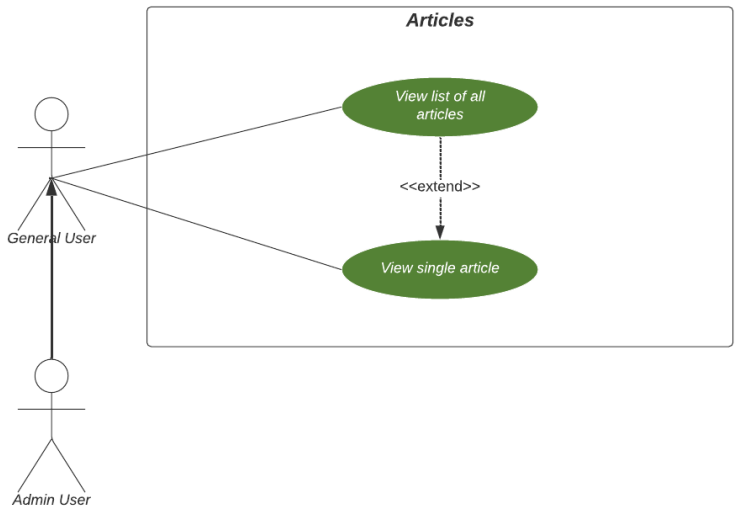


Figure 2.4.1.5: Use case diagram outlining functionality for the articles feature within the application.

There are minimal elements within the articles section as it largely consists of static components, not requiring API calls for additional functionality. From the articles homepage, where a full list of all articles within the application is shown, the user is able to view a single article (see Figure 2.4.1.5).

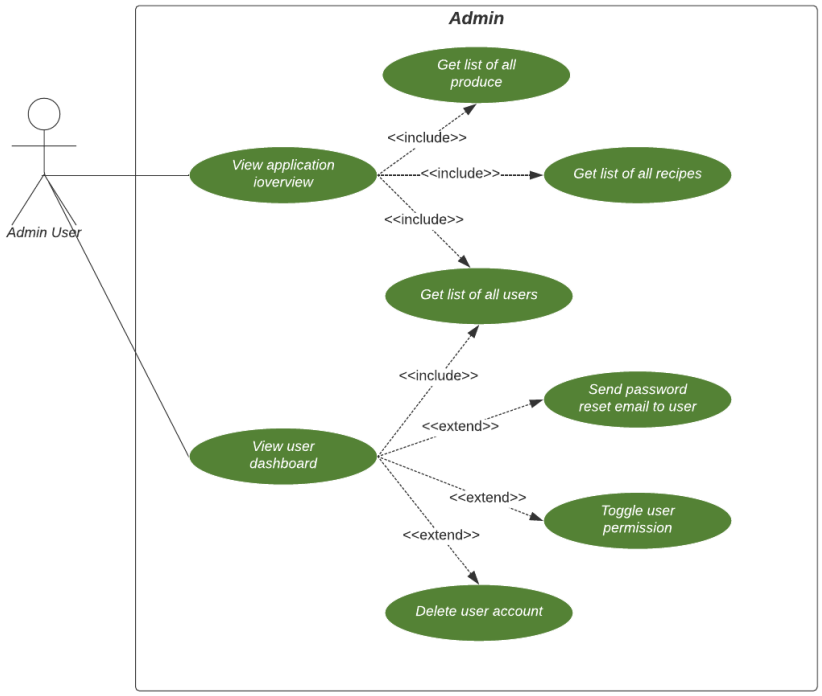


Figure 2.4.1.6: Use case diagram outlining functionality for the admin area within the application.

Within the admin area, the user can view an application overview and a user dashboard (see Figure 2.4.1.6). The application overview requires the retrieval of lists for all produce, recipes, and users, while the user dashboard must attain the list of all users. These requirements are represented by the include arrows. From the user dashboard it is possible to send password reset emails, toggle permissions and delete user accounts. Only admin users are able to access this functionality within the application.

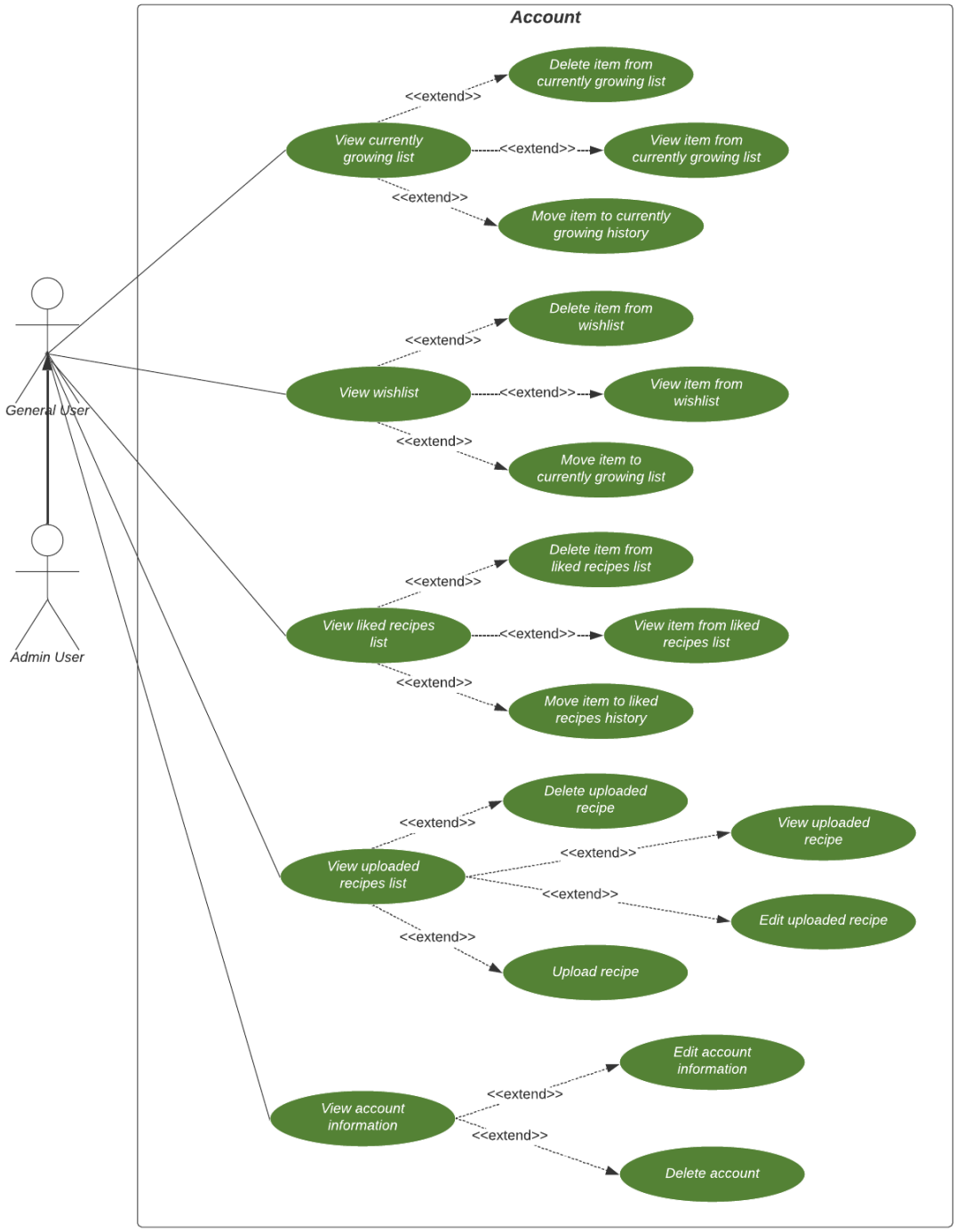


Figure 2.4.1.7: Use case diagram outlining functionality for the account feature within the application.

The account feature is vital for enabling users to maintain tailored lists of application infomration and to modify account data stored. There are five main sections to this; four relating to lists while the fifth is for controlling their personal data (see Figure 2.4.1.7). Within the currently growing list, a user is able to view items, delete items, and move items to their currently growing history. Similar functionality exists for the liked recipes list, while the wish-list enables produce items to be viewed, removed from the list or added to the currently growing list. The uploaded recipes section allows navigation to create a new recipe, view or edit existing recipes the user has uploaded, and delete recipes. As discussed, account infomration can be viewed, and the ability to edit personal data exists alongside functionality for deleting the user account.

2.5: Additional Design Artefacts

2.5.1: User Personas

Figure 2.5.1.1: User persona created for Rachel Watson, The Eco Warrior.

Figure 2.5.1.2: User persona created for Andy Greer, The Go-Getter.

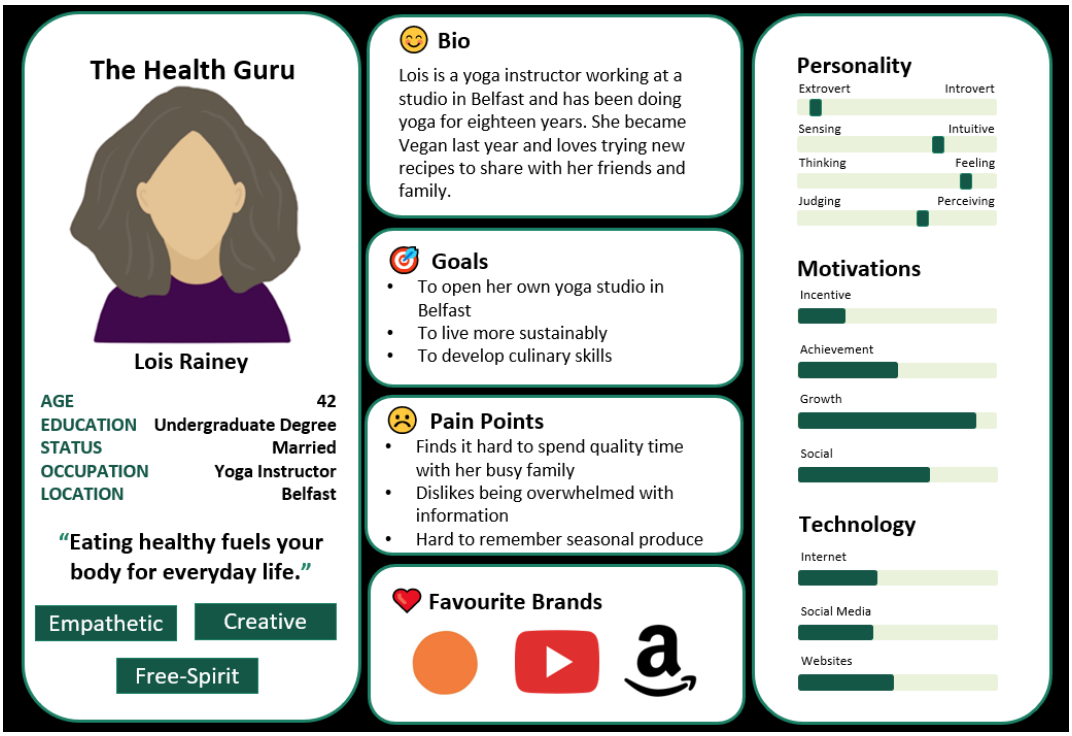


Figure 2.5.1.3: User persona created for Lois Rainey, The Health Guru.

User personas are widely viewed as one of the most effective mechanisms for achieving a high-quality product (Goltz, 2014). Personas were established during the creation of the initial prototype of the application and have remained a crucial part of the development process. Maintaining a strong sense of user focus is vital to the success of the application and has been made easier through establishing personas. These personas were generated through the combination of real-word user information, obtained during the initial requirements gathering process, and fictitious data to establish user intention for application usage (Faller, 2019).

Three personas were produced and contain a range of information (see Figures 2.5.1.1, 2.5.1.2 and 2.5.1.3), such as an overarching title, name, short biography, and core characteristics. Goals, pain points, and favourite brands alongside visual representations of personality traits, motivations and technological expertise have also been included. This information enables the construction of an application which aspires to create a positive user experience, as thorough consideration has been dedicated to user situations.

2.5.2: User Scenarios

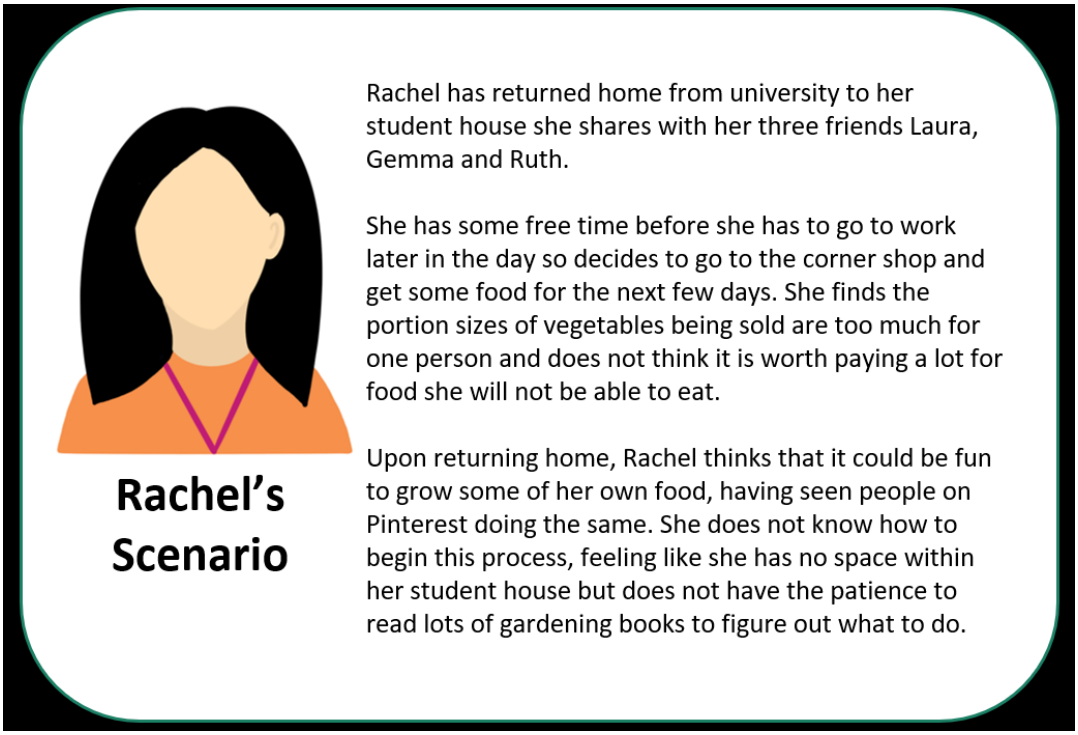


Figure 2.5.2.1: User scenario created for Rachel Watson (see Figure 2.5.1.1).

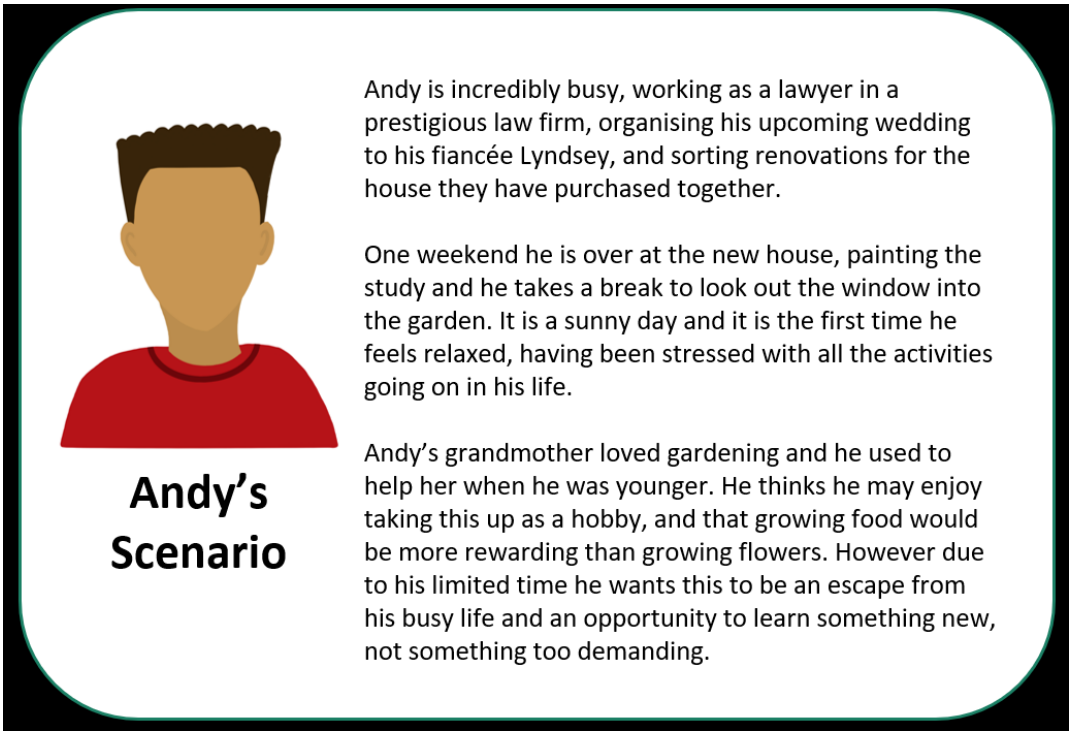


Figure 2.5.2.2: User scenario created for Andy Greer (see Figure 2.5.1.2).

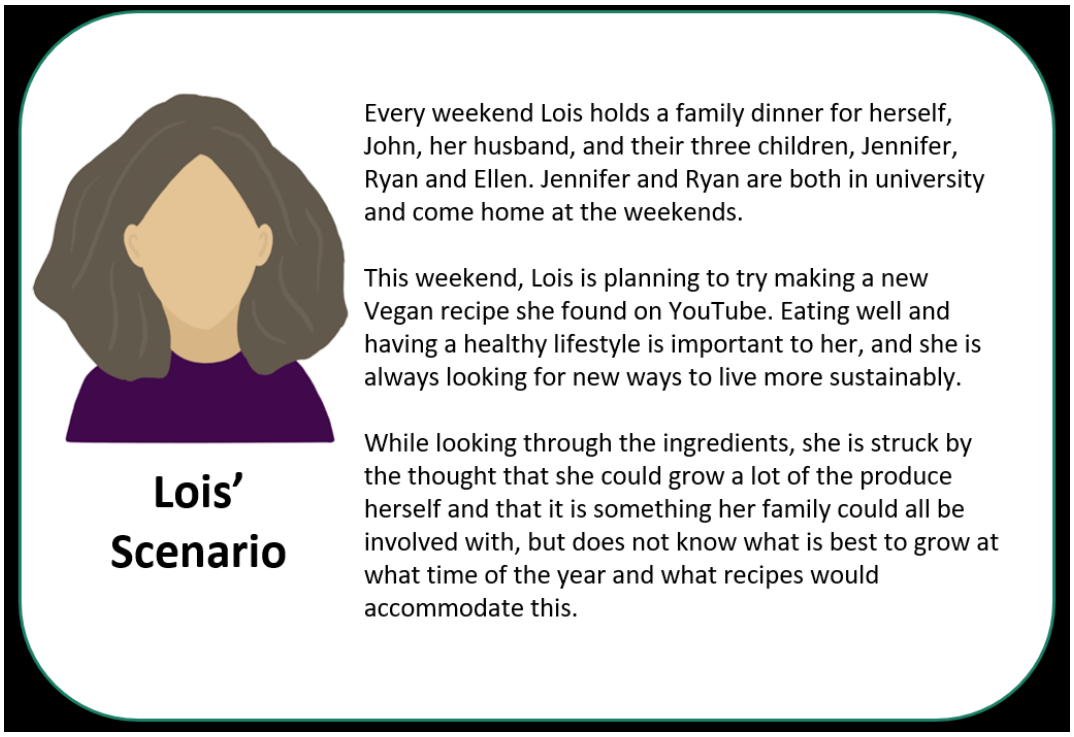


Figure 2.5.2.3: User scenario created for Lois Rainey (see Figure 2.5.1.3).

User scenarios are built upon the foundation of user personas established in section 2.5.1, and provide insight into when the personas would utilise the application (see Figures 2.5.2.1, 2.5.2.2 and 2.5.2.3). Goltz (2014) discusses the importance of goal-directed design, with personas and scenarios fundamental aspects of this. They describe personas at the starting place, with the scenario as the pathway to achieving a particular goal.

These scenarios are important for building developer empathy for the users (Faller, 2019), which aids in enhancing the user experience of the application as a relationship has been forged with the application end users. Scenarios further emphasise features most desirable to end users as they are based on real people and situations, enabling the development process to be streamlined using this method for prioritisation.

2.5.3: User Journey Maps

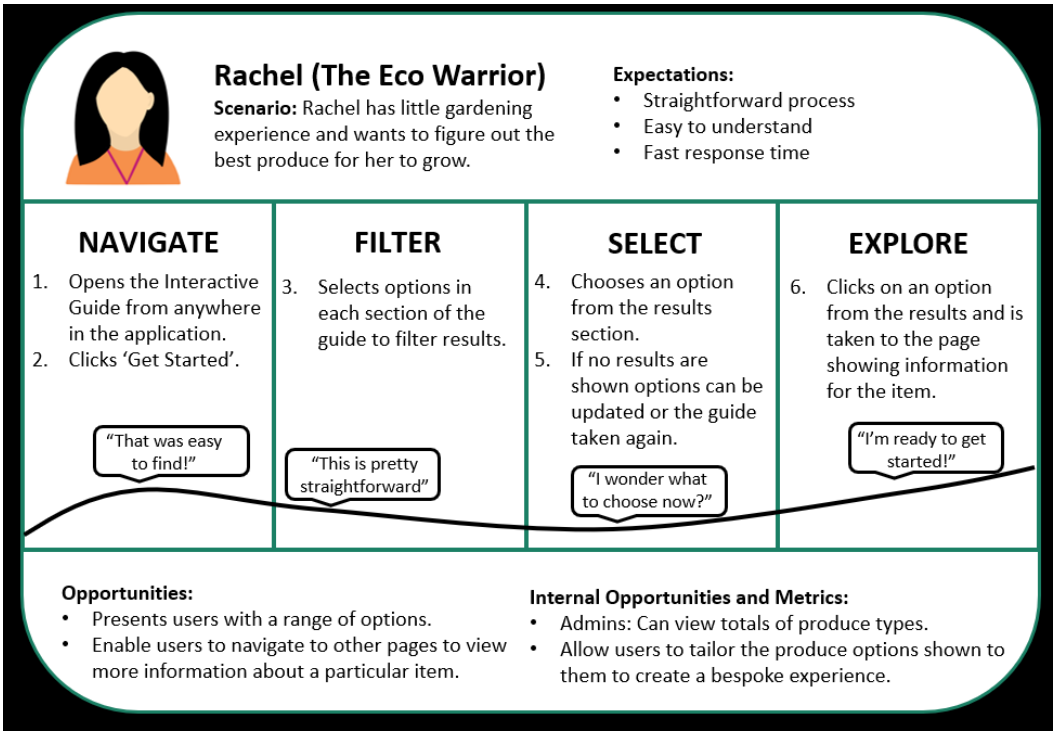


Figure 2.5.3.1: Journey Map created for Rachel Watson (see Figures 2.5.1.1 and 2.5.2.1).

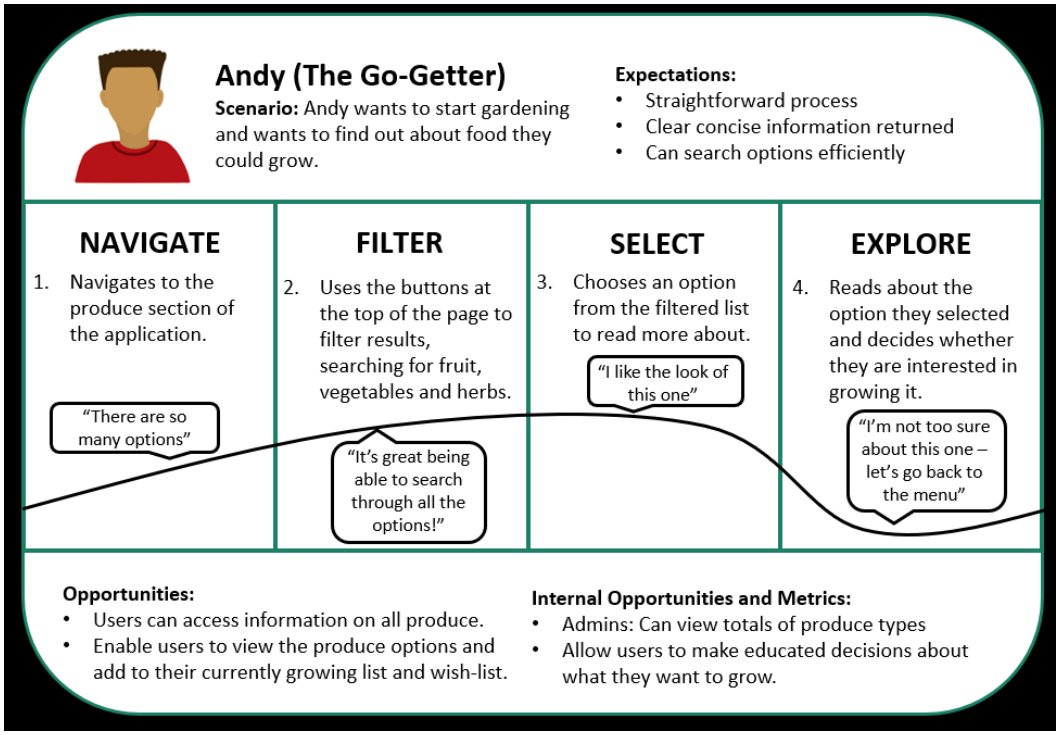


Figure 2.5.3.2: Journey Map created for Andy Greer (see Figures 2.5.1.2 and 2.5.2.2).

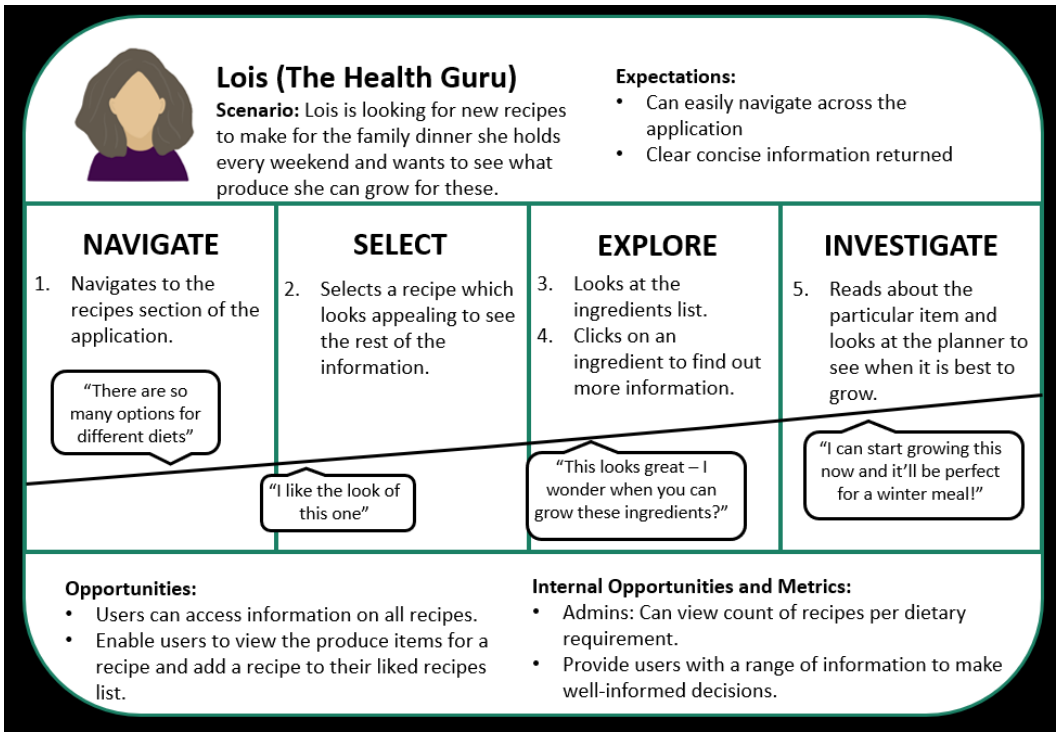


Figure 2.5.3.3: Journey Map created for Lois Rainey (see Figures 2.5.1.3 and 2.5.2.3).

Journey maps are an extension of the scenarios discussed in section 2.5.3, and identify the steps a user takes through the application to achieve their goal. In this way journey maps can be inserted into Goltz's illustration of goal-orientated design (2014) to reflect the actions taken to achieve the user goal. Therefore, the order of these elements would be persona, scenario, journey map and goal.

Journey maps are a visualisation of the process to achieving a goal (Gibbons, 2018) for each scenario created and user expectations are declared before each process pathway. This path is broken down further into phases, and across Figures 2.5.3.1, 2.5.3.2 and 2.5.3.3 there are four phases similarly titled. Actions, mindsets, and emotions are additionally plotted, valuable assets for observing user behaviour and thought processes (Gibbons, 2018). These are illustrated across all phases; actions are numbered, mindsets are declared in speech bubbles and the emotional flow is charted with a line. Opportunities and metrics are identified, based on the pathway executed, and identify core aspects of achieving a particular goal.

2.5.4: Application Sitemap

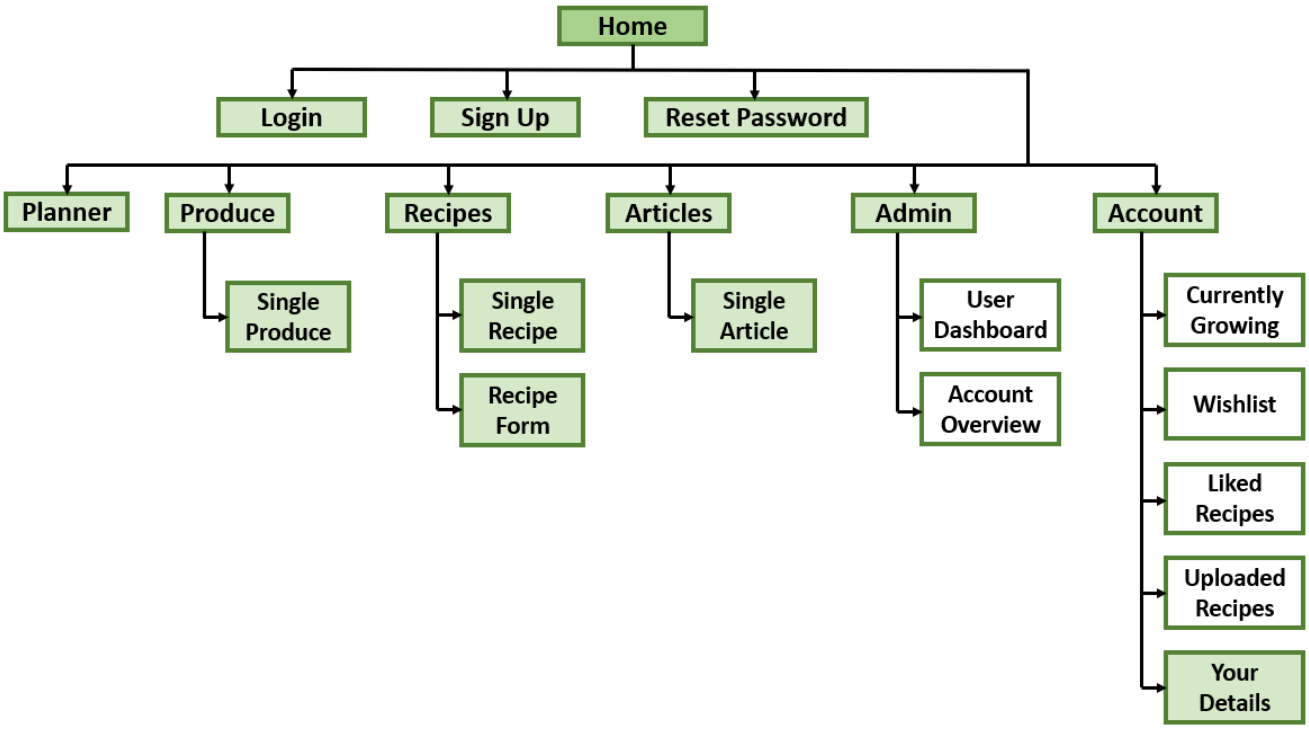


Figure 2.5.4.1: Sitemap representing the page structure of the application. Boxes with green backgrounds signify components acting as webpages, while boxes with a white background symbolise components which change the view of the parent component.

A sitemap is a simplistic way of viewing the application navigation and by extension is reflective of the features included. As the system is developed using React, components are used to establish various aspects of the application, including webpages, page sections and reusable elements included in these areas. In Figure 2.5.4.1, boxes with green backgrounds are components utilised as webpages while boxes with a white background refer to components representing sections within a page. For example, in the account page there are sections for currently growing, wish-list, liked recipes and uploaded recipes, and it is possible to navigate to the details page.

This diagram provides a basic overview of the core pathways through the application. Many additional links exist between elements, such as uploaded recipes can navigate to recipe form, currently growing to single produce, and single recipe to single produce. However, the existence of these links is dependant upon whether users have taken advantage of the system functionality, so have not been automatically included in this sitemap.

2.5.5: UML Activity Diagrams

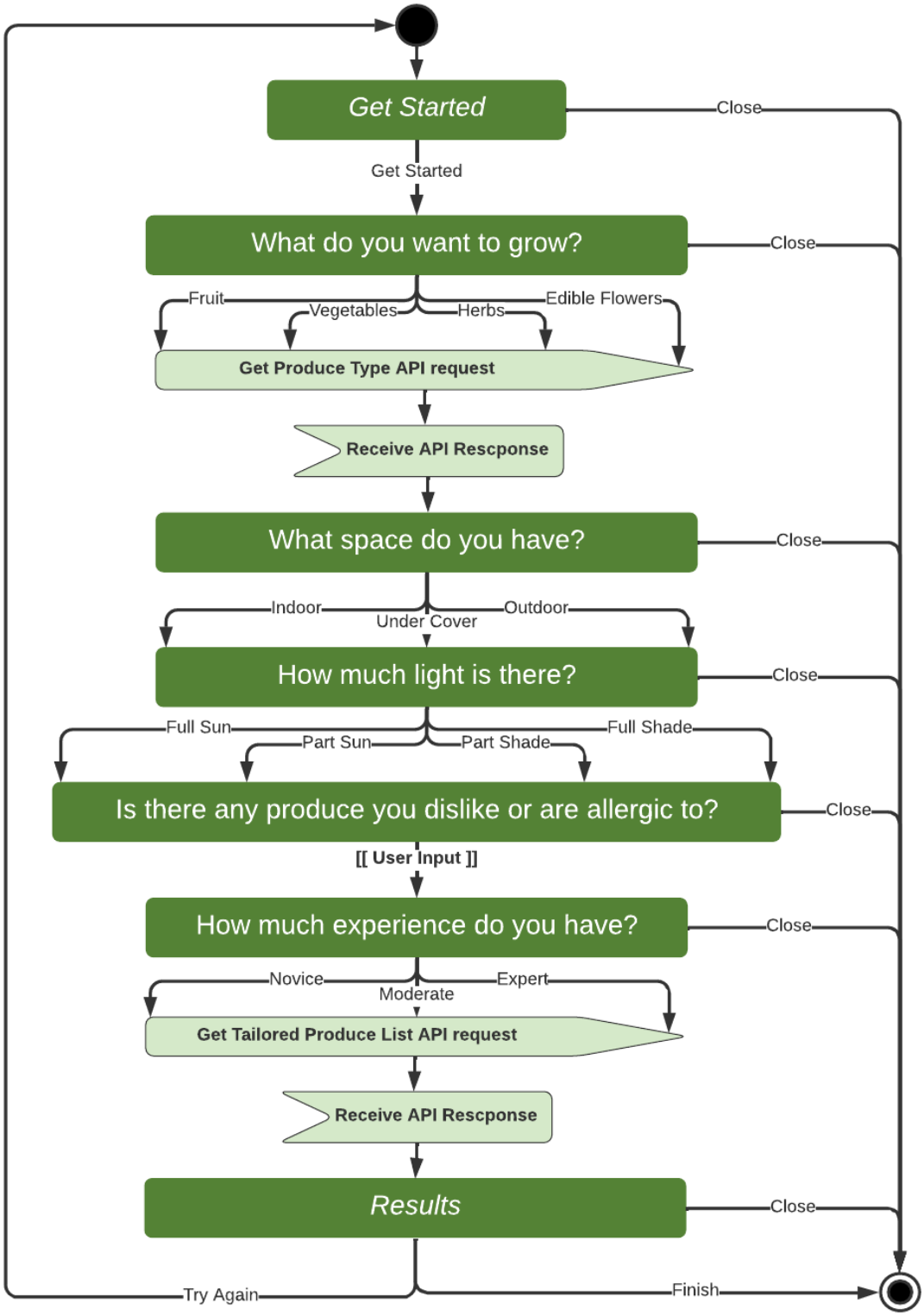


Figure 2.5.5.1: Activity diagram representing the user interactions with the interactive guide feature. The two API calls are also shown in the illustration with each dark green box representing a component within the guide.

Figure 2.5.5.1 is an extension of the use case diagram for the interactive guide shown in Figure 2.4.1.3, showing the flow for how the guide can be used once opened by the user from any area of the application once successfully logged in.

A UML diagram was created for the interactive guide to aid in developer clarity for how this was to be implemented and to understand how the end users would interact with this feature (Lucidchart, 2022). Start and end nodes are identified by the black circles, reflecting the beginning and end of guide usage, while arrows with text signify options for accepted user input within the feature which enable access to the subsequent action or element.

The dark green rectangles represent actions and, in this diagram, reflect sections of the guide, the text within the rectangles are the section titles and questions used for the feature. There are two light green tags which appear one after the other in two areas of the activity diagram, the first is representative of an API call triggered by user interaction and the second connotes the response from the prior request.

There are two API requests made within this feature: to retrieve a list of all produce of a particular type chosen by the user and to attain a tailored produce list based on user input throughout the guide. The API call to obtain a produce type list acquires data used to populate the dropdown field in the dislikes and allergies section. The user is able to close the guide at any point during its execution and once the final results are obtained, it is possible to restart the interactive guide.

3.0 Project Plan and Requirement Specification

3.1 Reflection on Implementation Plan and Execution

Minimal adaptations were required to the implementation plan created for developing the initial prototype, with changes being reflective of the requirements prioritised not using additional services. For instance, the Weather API as the tailored plant care feature was not implemented at this stage as users valued other features greater.

The decisions made regarding languages, platforms and software used, remained strong and the selection of JWT as the authentication mechanism proved valuable during development.

Application progress occurred at a steady and consistent pace, with accountability for this being enforced by version control and the enforcement of an iterative development process. The completed system encompasses a wide assortment of code and image files (see section 3.4), and the levelled approach to implementation heavily contributed to the overall professional performance and generation of a well-rounded system.

3.2 Rationale for Tools and Technologies Implemented

An assortment of resources were required to successfully execute development for the desired application. These elements include programming, markup and styling languages, frameworks, an effective authentication mechanism, and a database management system. Chosen software solutions and essential physical hardware is also discussed. Tools and technologies implemented remained largely consistent with those used in the initial prototype, and the developer acquired deep understanding of facets used during this process.

3.2.1 Programming, Markup and Styling Languages

Selecting the best languages for development is vital for ensuring successful implementation of the application, both client and server aspects. Great consideration went into this process, with JavaScript emerging as the superior candidate. This was the case due to extensive developer experience with the language, its assimilation into both client and server frameworks, its overall global popularity and existence as a language best fitted for developing versatile web applications (Costa, 2021; Goel, 2021). Using JavaScript as the main programming language provides a level of consistency throughout the application as both client and server sides utilise the same core language.

In terms of the client aspect, the React framework has been implemented. Rationale for this includes the wealth of experience using this framework possessed by the developer, the efficient performance it provides (Casey, 2021; Costa, 2021) and the abundance of documentation available for support. The ES6 and JSX syntax is required for implementing React, determining the markup language, to generate components employing markup and logic (React, 2022). In addition, using the React library enables the use of Redux, a predictable state container which enables consistent application behaviour and allows systems to run in various environments (Redux, 2022). SCSS is used for styling and follows BEM (Block Element Modifier) notation (Rendle, 2016).

The React front-end is generated using Webpack, which creates application builds which can be tailored to meet the requirements of the system. It is a “static module bundler for modern JavaScript applications” (Webpack, 2022) and has enabled the creation of two different scripts, one for running the code in a development environment and another to create a version compiled to vanilla JavaScript which can be easily deployed.

As for the server side of the system, an Express server was developed and is flexible, extremely popular, and is based upon the Node.js runtime environment (Cost, 2021). This server creates a REST API (Representation State Transfer Application Programming Interface) for connecting the front-end to the database system selected, by exposing a set of endpoints which facilitate the transfer of information (Masse, 2014). Compliance to best practices has been necessary during development, which includes being developer-friendly, extensible, and secure (Doglio, 2015).

3.2.2 Frameworks

Information regarding the React and Express frameworks included in the application is explored in depth within the above section. Alongside these frameworks, it has been of utmost importance to establish an effective test plan for assessing the React components, leading to the use of Jest and Enzyme.

Jest is a simple JavaScript testing framework (Jest, 2022) which has been used to create a wide assortment of unit tests for this application in conjunction with Enzyme. Enzyme is able to expand upon Jest to enable React components to be fully tested (Enzyme, 2022), from the rendering to the functions included.

3.2.3 Authentication Mechanism

Prior to making the initial prototype there was great debate surrounding the best approach to handling authentication. It was decided to use JWT (JSON Web Tokens) on the Express server to create tokens as it enables a custom implementation for security, the payload can be adapted to include any variable and the duration can be set to any desired length of time. Its recognition as a widely used industry standard (JWT, 2022) is important as it aids in creating a trustworthy, professional authentication system.

3.2.4 Database Management System (DBMS)

Choosing a database system is a fundamental part of application design and development, as it feeds into the data structure design (see section 2.3.2) and the overall management of information. Using MongoDB proved a strong choice, as its nature as a NoSQL document data store delivers the ability to handle change and large volumes of content (AWS, 2022). Specifically using MongoDB Atlas, classed as a cloud platform due to the range of additional facilities it provides (MongoDB, 2022b), will prove beneficial in the long term when it comes to deployment as it allows access from anywhere in the world and is highly scalable.

3.2.5: Software Solutions

Visual Studio Code (VS Code) was fundamental to the development process as this software was used to write all the JavaScript code for the whole application. All scripts were run from the built-in terminal, to start the code base as well as run test files, and extensions including ES7+ React/Redux/React-Native snippets and Prettier were installed to make the development process more efficient.

Postman was heavily used during the process of creating the Express server, to ensure that endpoints established were returning expected results and that API connections were successful. It is a widely used platform for using and generating APIs, possessing a range of tools and documentation (Postman, 2022). Testing was also carried out within Postman and endpoint documentation produced.

MongoDB Compass was used to maintain and check information held within the application MongoDB collections. It provides a visual interface that can be used to query the data held within the cluster, supplying a faster way for the developer to follow the information flow than using the command line or browser.

Procreate is an illustration software utilised for generating every icon and image within the application. It was used to help visually establish the brand of the end product and deliver a bespoke, distinctive front-end. In the same arena, Figma was employed to generate wireframes and prototypes for both desktop and mobile views during the design stage of development.

As for version control, Azure DevOps Repos was the chosen mechanism as it integrates well with VS Code, has a broad range of functionality extending beyond other version control systems, enables straightforward code management, and in the long term facilitates the creation of build pipelines which could be used for application deployment. Trello was used for establishing and maintaining Kanban boards used during each iteration, which were built upon the product backlog containing all requirements gathered.

3.2.6: Physical Hardware

Without physical equipment the whole application would never have been developed and ensuring the possession of all necessary hardware has been important for additionally assuring developer health and safety through employing ergonomic equipment. Essential hardware included a laptop, mouse, and keyboard in addition to a headset. An iPad and Apple Pencil were required for using the Procreate software.

3.3 Use of Version Control

Successfully implementing a version control mechanism has been fundamental in ensuring consistent application development across all iterations. Azure DevOps Repos, as discussed in the previous section, was the mechanism used for maintaining the source code created. Great attention was given to the management of the version control system, with Gitflow selected as the chosen workflow.

Gitflow is a branching strategy, with a core set of branches which live longer than in other models and has bigger commits. It is an alternative to truck-based workflows, which are better suited to projects with CI/CD pipelines (Atlassian, 2022). A CI/CD pipeline does not exist at this stage, and has been considered for future development.

Use of version control has complied with the Gitflow model as two main branches have been the centripetal points of the system which are labelled 'develop' and 'master' (see Figure 3.3.1). The master branch contains the most complete edition of the code base, with commits to this branch occurring upon initial prototype and final sprint completion. The develop branch comprises of the most up to date, working version of the application, with all other branches being taken off this one.

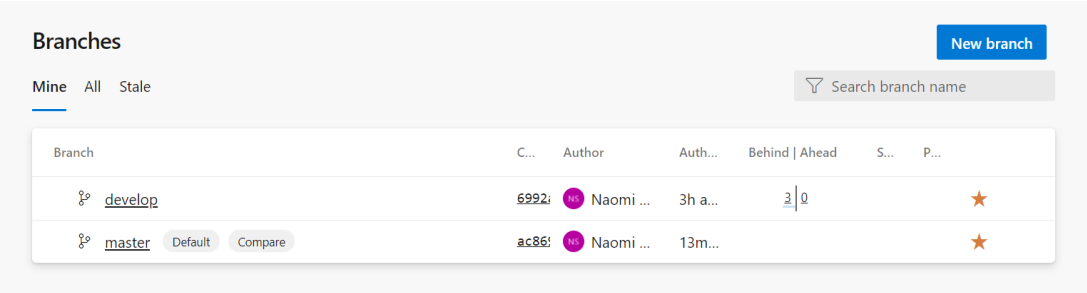


Figure 3.3.1: A screenshot of the branches page in the application repository on Azure DevOps Repos. At this point the develop and master branches exist, and are marked as favourites. All other branches are created from this page.

```
Naomi@DESKTOP-15O2DUV MINGW64 ~/Documents/DOCUMENTS/the-plant-pantry (develop)
$ git branch
* develop
  feature/account-forms
  feature/account-wishlist-growing
  feature/admin-area
  feature/articles
  feature/interactive-guide
  feature/multi-select-dropdown
  feature/navbar
  feature/planner
  feature/produce
  feature/produce-images
  feature/recipes
  fix/account-modals
  fix/actions-stylesheets-refactoring
  fix/env-encryption
  fix/final-uat-changes
  fix/redux-extension
  fix/sprint5-uat-testing
  master
```

Figure 3.3.2: A screenshot of results from the 'git branch' command executed within the VS Code terminal for the project. This presents a list of all branches that have been created throughout the application development and the alignment to the Gitflow workflow can be observed.

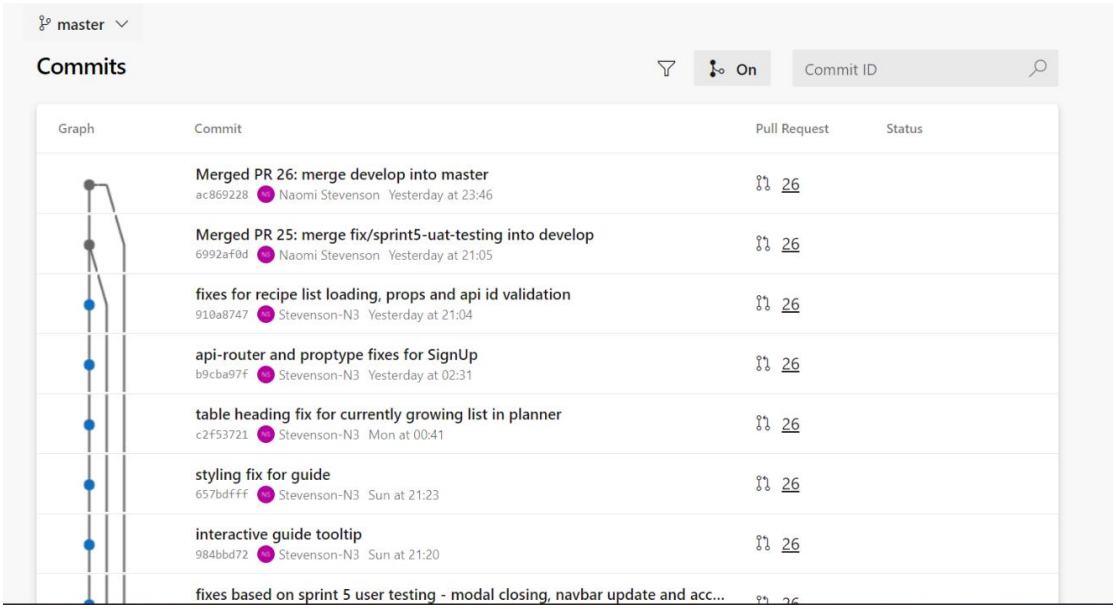


Figure 3.3.3: A screenshot of the commit history for the project as shown in the application repository on Azure DevOps Repos. At the top of this is the execution of the merge request from the develop branch into the master branch, marking the completion of the final sprint.

Additional branches created are prepositioned by the terms ‘feature’ or ‘fix’ to denote whether the branch is an implementation of a new requirement or fixes a bug in the system (see Figure 3.3.2). The text following the terms are a short reference to the content of the branch and are separated by dashes. Once these branches are completed, they are merged into ‘develop’ through pull requests, with only the develop branch being merged into the master branch (see Figure 3.3.3).

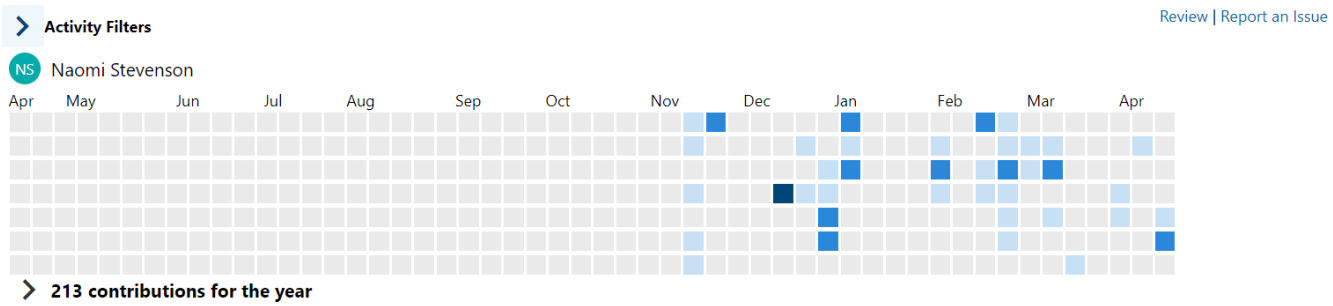


Figure 3.3.4: A screenshot of the contribution graph taken during the final stages of development. Consistent code generation is apparent given the frequency of blue boxes across the graph, with darker shades connoting more commits on the particular day.

Version control has been consistently used, as shown in the contribution graph (see Figure 3.3.4). It has provided developer accountability, application security and a professional approach to project management. In total there have been 165 commits and 25 pull requests created.

3.4 System Walkthrough

The following walkthrough provides a brief overview of how a user would interact with the developed system. Prototypes reflecting the appearance of all aspects of the application can be found in Appendix B.4.

The first page the user is greeted with is the homepage (see Figure 3.4.1). When the user tries to navigate to any other page, they are redirected to either the login or sign-up page if they are not signed in (see Figure 3.4.3).

Figure 3.3.1: Screenshots of the homepage for The Plant Pantry. On the left is the homepage when the user is not signed in and the right shows the appearance of the homepage upon successful login, with the updated navigation bar (see Figure 3.4.2) and the interactive guide icon being shown.

```
22. if (localStorage.getItem("tpp-token")) {
23.     var decoded = jwt.decode(localStorage.getItem("tpp-token"));
24.     if (decoded.exp > new Date().getTime() / 1000) {
25.       setIsLoggedIn(true);
26.     } else {
27.       setIsLoggedIn(false);
28.     }
```

Figure 3.4.2: Lines 22 – 28 of the Navbar.react.js file in the client directory. This is found in a function checking the presence of a valid user token, the existence of the token changes the navigation bar view.

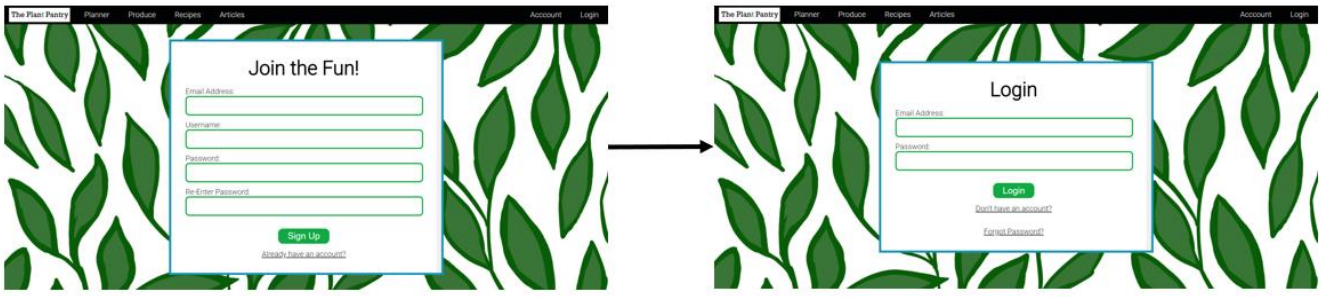


Figure 3.4.3: Screenshots of the sign-up and login pages. If a user signs up for an account, they are then directed to the login page to sign into the application.

The interactive guide can be accessed from anywhere in the application through the icon in the bottom right of any page once signed in. It appears as a modal on top of the current page the user is viewing and has a total of seven sections (see Figure 3.4.4).

Figure 3.4.4: Screenshots of all sections of the interactive guide. The results are for vegetables in an indoor space with full sun where the user dislikes Brussel Sprouts and has minimal experience growing produce (see Figure 3.4.5).

```
77. let searchParams =
78.       "?location=" +
79.       params.location +
80.       "&light=" +
81.       params.light +
82.       "&notInclude=" +
83.       params.notInclude +
84.       "&skill_level=" +
85.       skillLevelVal;
86.     this.props.getTailoredProduceList(params.type, searchParams);
```

Figure 3.4.5: Lines 77 – 86 of the InteractiveGuide.react.js file in the client directory. This is found in a function handling value selection from each part of the guide. Once all parts of the guide have been traversed the values collected from the user are added to a query string which is sent to a specific endpoint to obtain the results list.

Moving from left to right across the navigation bar, the next feature is the planner (see Figure 3.4.6). The user is able to view the actions required throughout the year to care for a range of produce items. Within the table, if an icon is hovered over a tooltip appears stating what action the icon refers to so the user does not have to remember the meaning of every icon.

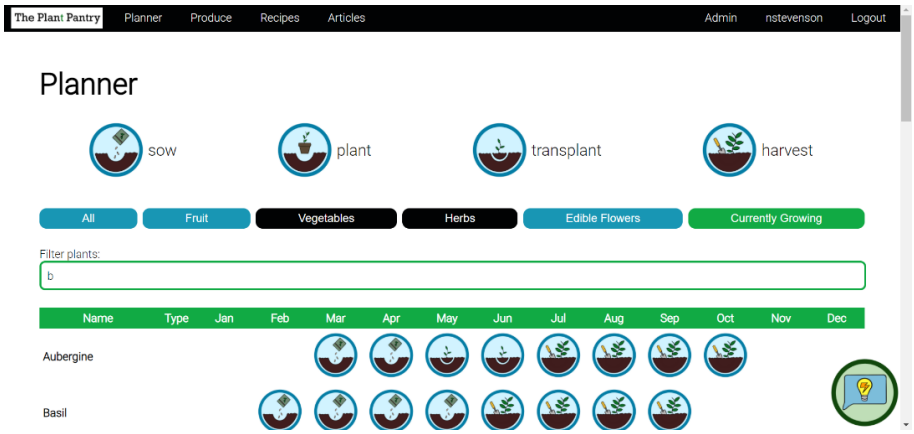


Figure 3.4.6: A screenshot of the planner page. The user has filtered by vegetables and herbs, and also is looking for plants containing the letter 'b'.

Next is the produce homepage, where the user can access a list of all produce items held in the system, which can be filtered based on user input (see Figure 3.4.7). From here, the user can navigate to a page containing all information held for a single produce item and add it to their currently growing list or their wish-list.

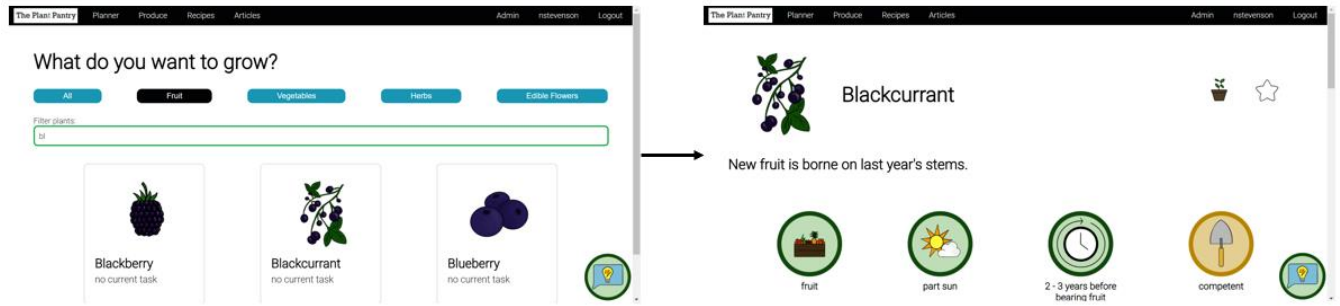


Figure 3.4.7: Screenshots of the produce homepage and the single produce page. The produce page has been filtered by type 'fruit' and the user is looking for items containing 'bl'. They then clicked on 'Blackcurrant' and are taken to its individual page where they have added it to their currently growing list.

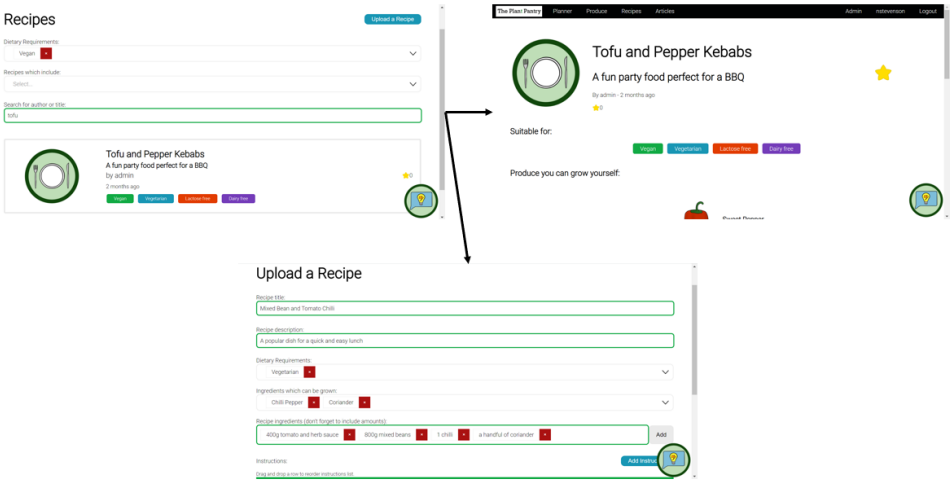


Figure 3.4.8: Screenshots of the recipe homepage, single recipe page and recipe form. The recipe page is filtering recipes by the diet of 'Vegan' and by value 'tofu'. The user then clicked on the recipe and added it to their liked recipes list. From the homepage they also clicked the upload recipe button and completed the form for a mixed bean and tomato chilli.

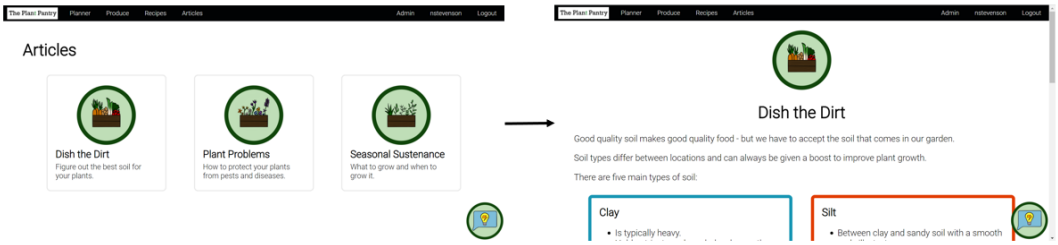


Figure 3.4.9: Screenshots of the articles page and single article page. The user clicked on the article 'Dish the Dirt' which took them to view the corresponding article.

Following this is the recipes area (see Figure 3.4.8). Like the produce section, the recipe homepage allows the user to access a list of all recipes contained in the system, which can be filtered based on user input. From here, the user can navigate to a page containing all information held for a single recipe and add it to their liked recipes list, or navigate to the recipe form where they can upload a recipe. If the user seeks to edit a recipe they have uploaded, the form used is identical and gets populated with existing recipe data. The articles section (see Figure 3.4.9) contains a list of articles, and the user is able to navigate to view an individual article from this page.

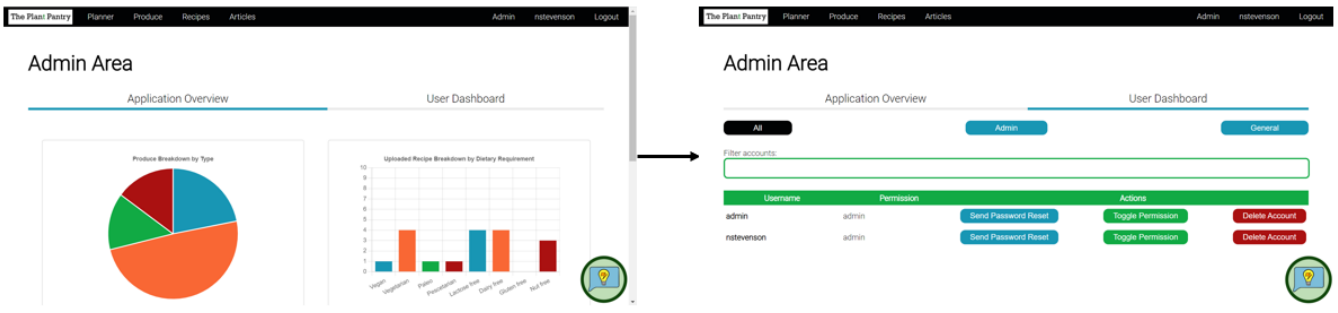


Figure 3.4.10: Screenshots of the application overview and user dashboard in the admin area.

If the user has administrative privileges, they can view the admin area containing an application overview and a user dashboard (see Figure 3.4.10). The application overview shows charts representing important values contained in the system, while the user dashboard allows the admin user to view a list of all users, send password reset requests, toggle user permissions, and delete any user account.

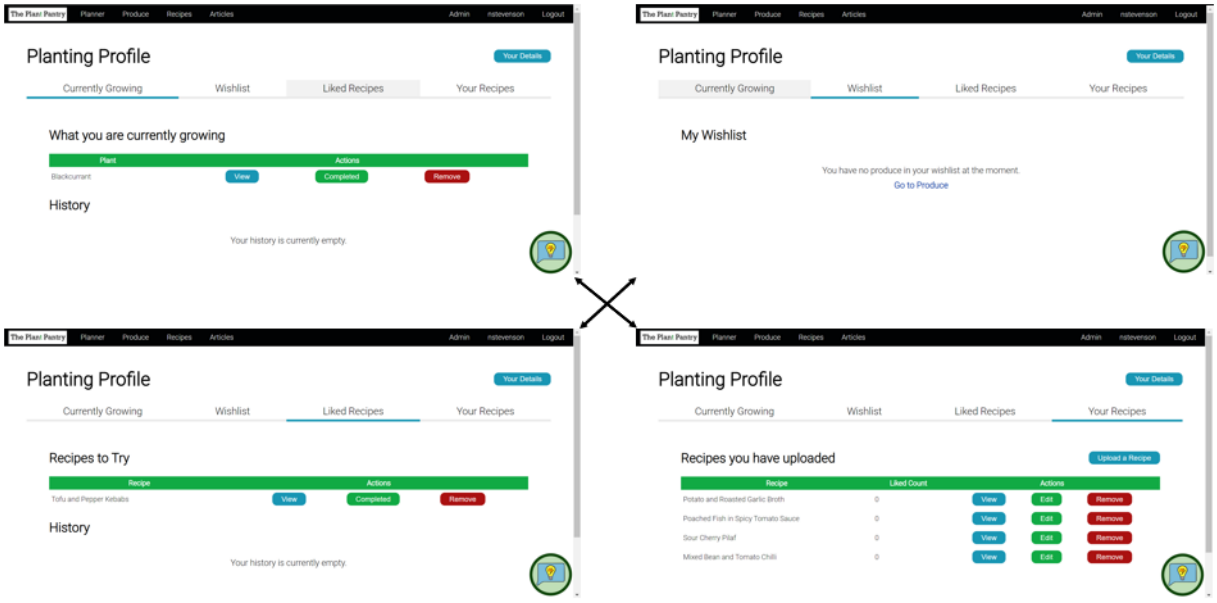


Figure 3.4.11: Screenshots of the account area, showing the currently growing list, wish-list, liked recipes and uploaded recipes. Items within these lists have been added through actions shown in Figures 3.4.7 and 3.4.8.

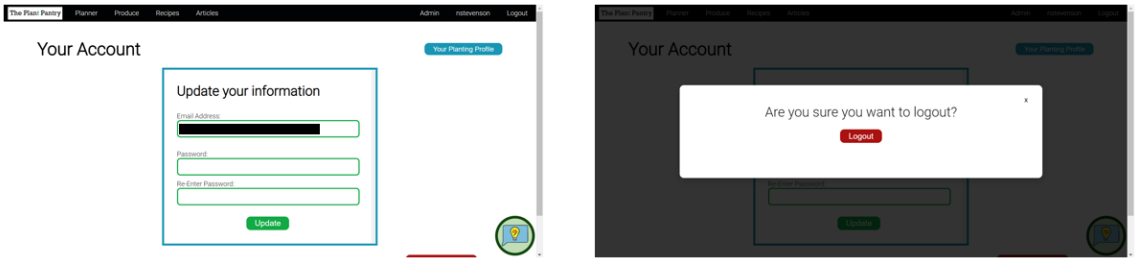


Figure 3.4.12: Screenshots of the account details area, accessed by the blue button at the top right in all screenshots in Figure 3.4.11, and the logout modal shown when 'Logout' is clicked in the navigation bar.

Every user has an account page, indicated in the navigation bar by their username. The user is able to view their currently growing list, wish-list, liked recipes and uploaded recipes from this section by clicking the corresponding tab at the top of the page (see Figure 3.3.11). From this area the user is able to navigate to a page where they can alter their personal information, such as their email address and password, or delete their account (see Figure 3.4.12).

When the user selects the 'Logout' link in the navigation bar the modal shown in Figure 3.4.12 is shown. If the logout button is clicked, they are then redirected back to the homepage and signed out of their account (see Figure 3.4.1).

3.5: Volume of Code Summary

A high volume of code was produced for this application, including test files for front-end components and functions. Figure 3.5.1 breaks this down into sections, and details additional information such as the number of scripts and dependencies used. Quantities have also been included for the images used within the project as these were generated by the developer and are bespoke to the application.

Area of Project	Measurement	Quantity
React code	Class components	30
	Functional components	3
	Arrow functional components	18
	Compound components	5
	JavaScript files containing functions and objects	3
React code unit tests	Test files	60
	Test cases	329
Images created by developer	Total images	161
	Fruit	27
	Vegetables	58
	Herbs	19
	Edible flowers	20
	Other images	37
Redux code	Action files	3
	Action creators	37
	Action functions	30
	Action Type Files	3
	Action Types	37
	Reducers	3
	Reducer cases	37
Redux code unit tests	Test cases	3
	Tests	37
Stylesheets	Total stylesheet files	22
Server	Total JSON files	5
	Seed functions	2
	Encrypted variables	3
	API endpoints	27
General	Scripts in package.json	12
	Dependencies	59
	Webpack configuration files	2
Total number of files for front-end		**340**
Total number of files for back-end		**17**
Total lines of code produced		**22826**

Figure 3.5.1: Table outlining the volume of code produced during the development of the project, broken down into front-end and back-end values with summative totals.

```
total files : 346
total code lines : 22826
total comment lines : 9
total blank lines : 2382
|      extension|     total code|  total comment|    total blank|percent|
-------------------------------------------------------------------------
|               |              3|              0|              0|  0.013|
|            .js|          14790|              9|           1826|     65|
|          .scss|           1262|              0|            177|    5.5|
|          .json|           6760|              0|            376|     30|
|          .html|             11|              0|              3|  0.048|
```

Figure 3.5.2: Contents of output file produced from running the Lines of Code (LOC) extension (lyzerk, 2020) in VS Code. The number of lines is broken down by file extension, with JSON files containing the most due to the dataset created for application use. This figure only includes files created, not modified (see Appendix C)

3.6: Application Security Considerations

In section 2.3.1 the privacy of data alongside validation methods were discussed in depth and strictly related to the overall handling of application data. Great thought was also given to the security of the application as a whole, with these areas discussed below.

Using JWT

The rationale for implementing JWT (JSON Web Tokens) has been outlined in section 3.2.3 of this report. JWT has successfully been applied to this application and errors regarding token expiry or lack of token presence are highlighted through the 401 Unauthorized HTTP status code returned from a failed request (see Figure 3.6.1).

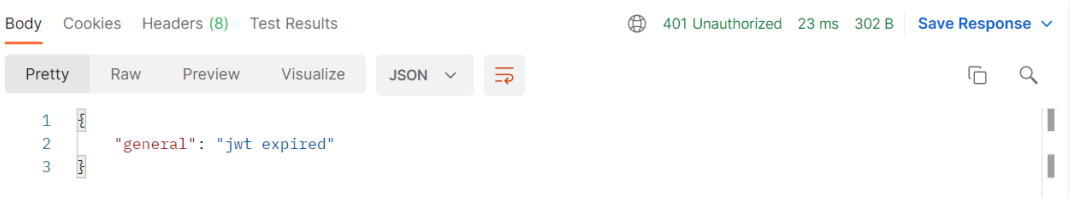


Figure 3.6.1: Screenshot of the response when a request is made to retrieve all the recipes for the application when the user's JWT token has expired. This is conveyed within the response message and the 401 HTTP status.

CORS Enabled

Cross-Origin Resource Sharing (CORS) allows API requests to avoid having to follow the same-origin policy imposed by web browsers. The application breaks this policy as the React front-end runs on port 3000 while the Express server runs on port 3010. This policy is defined as a "critical security mechanism" which helps avoid malicious web attacks (Mozilla, 2022). CORS is enabled in the back end of the application (see Figure 3.6.2) and the 'Access-Control-Allow-Origin' header is attached to all API requests to facilitate this.

```
1.  const express = require("express");
2.  const cors = require('cors');
3.  const apiRouter = require("./api-router");
4.
5.  function createExpressApp(database){
6.      const app = express();
7.      app.use(express.json());
8.      app.use(cors());
9.
10.     app.use("/api/v1.0", apiRouter(database));
11.
12.     return app;
13. }
14.
15. module.exports = createExpressApp;
```

Figure 3.6.2: The create-express-app.js file within the server directory of the project. This function generates the Express server, enabling CORS in the process.

Authorisation Header

The majority of API requests additionally require the presence of an 'Authorization' header to enable access to particular endpoints. This ensures that only users who are logged in, and therefore are in possession of valid user credentials, can access data from the MongoDB database through API calls.

Not all endpoints require this level of security (see Figure 3.6.3), for securing the login, accounts and forgot password API endpoints would render the application unusable.

```
96. router.use(
97.   checkJwt({ secret: global.env.JWT_SECRET, algorithms: ["HS256"] }).unless({
98.     path: [
99.       "/api/v1.0/login",
100.              "/api/v1.0/accounts",
101.              "/api/v1.0/forgot-password",
102.              "/api/v1.0/forgot-password/request",
103.            ],
104.          })
105.        );
```

Figure 3.6.3: Lines 96 – 105 of the api-router.js file within the server directory of the project. This checks the validity of the JWT token supplied by the user within the 'Authorization' header of the API request and prevents endpoint access unless the desired endpoint is within the path array shown.

Protected Routes

A ProtectedRoute component has been created to prevent the user from accessing features from the front end of the application unless they have successfully logged in to their account or, in the case of the admin section, have the appropriate access permission (see Figure 3.6.4).

This is managed through checking the login token supplied on login exists in localStorage, and the token is then decoded to check it has not expired and determine whether the user permission matches what is required for the specified route.

```
58.     <Route exact path="/" component={Homepage} />
59.     <Route exact path="/login" component={Login} />
60.     <Route exact path="/sign-up" component={SignUp} />
61.     <Route exact path="/forgot-password/:id" component={ForgotPassword} />
62.     <ProtectedRoute exact path="/planner" component={Planner} />
63.     <ProtectedRoute
64.       exact
65.       path="/produce/:produceName"
66.       component={SingleProduce}
67.     />
68.     <ProtectedRoute exact path="/produce" component={ProduceHome} />
69.     <ProtectedRoute
70.       exact
71.       path="/admin"
72.       component={Admin}
73.       adminRequired={true}
74.     />
```

Figure 3.6.4: Lines 58 –74 of the App.js file within the client directory of the project. Lines 58 – 61 display general routes which any user can access, lines 62 – 68 are protected routes which require the user to be logged in before navigating to, and lines 69 – 74 show a protected route which requires a user to be logged in and possess administrative permissions within the application.

4.0: System Verification

4.1 Reflection on the Verification Plan and Execution

The intention of conducting verification on the application was to ensure that the product generated met the requirements outlined at the beginning of the development process, see section 1.1. Determining outcomes from this type of testing is extremely clear as the requirements are either met or not.

Four methods were used for this process; application walkthroughs, code reviews, executing unit tests and assessing all API endpoints created. Verification occurred at the end of every iteration, enabling the developer to keep track of the compliance to requirements, specifically functional ones, and the successes of the implementations. This embodies Krug's (2005) belief that testing should be done throughout development. Maintaining this process throughout allowed issues to be caught early on, preventing bigger problems from being encountered in later stages and during the final round of application testing.

Results from the final walkthrough and API endpoint tests are shown in section 4.2 as each individual test within the technique possesses a value of pass, accept, or fail. Further evidence of API endpoint testing, alongside code reviews and unit tests will be discussed in section 4.3.

4.2 System Verification Results

4.2.1 Walkthrough

These are the results of the final application walkthrough conducted after completion of the final iteration and relate to the functional requirements defined. The intention during this was to test that upon projected successful interactions the system performed as expected (see Table 4.2.1.1) for all requirements successfully implemented. All tests cases identified have passed.

Test ID	Test Description	Retrieving / Sending Data to / from API	Req. Tested	Expected Result	Actual Result	Pass / Accept / Fail
T001	User is able to sign up for an account.	Sending	**FR01**	User successfully creates an account.	User successfully creates an account.	**Pass**
T002	User is able to log into account.	Sending	**FR02**	User successfully signs into their account.	User successfully signs into their account.	**Pass**
T003	User is able to log out of their account.	N/A	**FR03**	User successfully logs out of their account.	User successfully logs out of their account.	**Pass**
T004	User is able to update their account information.	Sending	**FR04**	User successfully updated information from their account page.	User successfully updated information from their account page.	**Pass**
T005	User is able to reset their account password.	Sending	**FR04**	User successfully updates their password through the password reset feature.	User successfully updates their password through the password reset feature.	**Pass**

T006	User is able to delete their account.	Sending	**FR05**	User successfully deletes account from the account page.	User successfully deletes account from the account page.	**Pass**
T007	User is able to view a list of all produce items.	Retrieving	**FR07**	User successfully views all produce from the produce page.	User successfully views all produce from the produce page.	**Pass**
T008	User is able to filter the list of produce items by type.	Retrieving	**FR08**	User successfully filters the items in the produce page by type.	User successfully filters the items in the produce page by type.	**Pass**
T009	User is able to filter the list of produce items by name.	Retrieving	**FR09**	User successfully filters the items in the produce page by name.	User successfully filters the items in the produce page by name.	**Pass**
T010	User is able to view a single produce item.	Retrieving	**FR10**	User successfully views a single produce item from the single produce page.	User successfully views a single produce item from the single produce page.	**Pass**
T011	User is able to view their currently growing list.	Retrieving	**FR12**	User successfully views currently growing list from their account page.	User successfully views currently growing list from their account page.	**Pass**
T012	User is able to add a produce item to their currently growing list.	Sending	**FR11**	User successfully adds a produce item to their currently growing list from the single produce page.	User successfully adds a produce item to their currently growing list from the single produce page.	**Pass**
T013	User is able to remove a produce item from their currently growing list.	Sending	**FR13**	User successfully removes a produce item from their currently growing list.	User successfully removes a produce item from their currently growing list.	**Pass**
T014	User is able to view their wish- list.	Retrieving	**FR16**	User successfully views wish-list from their account page.	User successfully views wish-list from their account page.	**Pass**
T015	User is able to add a produce item to their wish-list.	Sending	**FR15**	User successfully adds a produce item to their wish-list from the single produce page.	User successfully adds a produce item to their wish-list from the single produce page.	**Pass**
T016	User is able to remove a produce item from their wish-list.	Sending	**FR17**	User successfully removes a produce item from their wish-list.	User successfully removes a produce item from their wish-list.	**Pass**
T017	User is able to open the interactive guide.	N/A	**FR18**	User successfully opens the interactive guide.	User successfully opens the interactive guide.	**Pass**

T018	User is able to close the interactive guide.	N/A	**FR19**	User successfully closes the interactive guide.	User successfully closes the interactive guide.	**Pass**
T019	User is able to interact with the guide and receive tailored results based on input.	Sending	**FR20**	User successfully reaches the results section of the interactive guide and are presented with a list of produce.	User successfully reaches the results section of the interactive guide and are presented with a list of produce.	**Pass**
T020	User is able to go back and change answers provided to the interactive guide.	Sending	**FR21**	User is able to navigate to a previous section of the guide and change their answer.	User is able to navigate to a previous section of the guide and change their answer.	**Pass**
T021	User can click on a result item and be redirected to the singled produce page for the chosen item.	N/A	**FR20**	User successfully clicks a produce item shown in the guide results and is redirected to the single produce page for the item.	User successfully clicks a produce item shown in the guide results and is redirected to the single produce page for the item.	**Pass**
T022	User can view a planner for all produce.	Retrieving	**FR23**	User successfully views the planner page.	User successfully views the planner page.	**Pass**
T023	User can view a planner for all produce in their currently growing list.	Retrieving	**FR24**	User successfully filters the items in the planner to show what is in their currently growing list.	User successfully filters the items in the planner to show what is in their currently growing list.	**Pass**
T024	User can filter the planner by type.	Retrieving	**FR25**	User successfully filters the planner by type.	User successfully filters the planner by type.	**Pass**
T025	User can filter the planner by name.	N/A	**FR26**	User successfully filters the planner by name.	User successfully filters the planner by name.	**Pass**
T026	User can view a list of all articles.	N/A	**FR27**	User successfully views the articles page.	User successfully views the articles page.	**Pass**
T027	User can view a single article.	N/A	**FR28**	User successfully views a single article.	User successfully views a single article.	**Pass**
T028	User is able to view a list of all recipes.	Retrieving	**FR38**	User successfully views all recipes from the recipes page.	User successfully views all recipes from the recipes page.	**Pass**
T029	User is able to view a single recipe.	Retrieving	**FR38**	User successfully views a single recipe.	User successfully views a single recipe.	**Pass**

T030	User is able to filter recipes list by title.	N/A	**FR45**	User successfully filters the items in the recipes page by title.	User successfully filters the items in the recipes page by title.	**Pass**
T031	User is able to filter recipes list by author.	N/A	**FR46**	User successfully filters the items in the recipes page by author.	User successfully filters the items in the recipes page by author.	**Pass**
T032	User is able to filter recipes list by ingredient.	Retrieving	**FR47**	User successfully filters the items in the recipes page by ingredient from produce list.	User successfully filters the items in the recipes page by ingredient from produce list.	**Pass**
T033	User is able to filter recipes list by dietary requirement.	Retrieving	**FR48**	User successfully filters the items in the recipes page by dietary requirement.	User successfully filters the items in the recipes page by dietary requirement.	**Pass**
T034	User can view their liked recipes list.	Retrieving	**FR50**	User successfully views their liked recipes list from their account page.	User successfully views their liked recipes list from their account page.	**Pass**
T035	User can add recipe to their liked recipes list.	Sending	**FR49**	User successfully adds a recipe to their liked recipes list from the single recipe page.	User successfully adds a recipe to their liked recipes list from the single recipe page.	**Pass**
T036	User can remove recipe from their liked recipes list.	Sending	**FR51**	User successfully removes a recipe from their liked recipes list.	User successfully removes a recipe from their liked recipes list.	**Pass**
T037	User is able to upload a new recipe.	Sending	**FR52**	User successfully uploads a new recipe to the application using the recipe form.	User successfully uploads a new recipe to the application using the recipe form.	**Pass**
T038	User is able to view a list of all recipes they have uploaded.	Retrieving	**FR53**	User successfully views their uploaded recipes list from their account page.	User successfully views their uploaded recipes list from their account page.	**Pass**
T039	User can edit a recipe they have uploaded.	Sending	**FR54**	User successfully edits a recipe they uploaded using the recipe form.	User successfully edits a recipe they uploaded using the recipe form.	**Pass**
T040	User can delete a recipe they have uploaded.	Sending	**FR55**	User successfully deletes a recipe they uploaded.	User successfully deletes a recipe they uploaded.	**Pass**
T041	Users are allocated access permissions.	Sending	**FR75**	User successfully navigates to the admin page as they have admin permissions.	User successfully navigates to the admin page as they have admin permissions.	**Pass**

T042	Admin users can view an application overview of key information.	Retrieving	**FR76**	User successfully views the application overview on the admin page as they have admin permissions.	User successfully views the application overview on the admin page as they have admin permissions.	**Pass**
T043	Admin users are able to view a list of all users on the user dashboard.	Retrieving	**FR77**	User successfully views the user dashboard on the admin page as they have admin permissions.	User successfully views the user dashboard on the admin page as they have admin permissions.	**Pass**
T044	Admin users are able to toggle permissions for any user.	Sending	**FR78**	User successfully updates permission for a user on the admin page as they have admin permissions.	User successfully updates permission for a user on the admin page as they have admin permissions.	**Pass**
T045	Admin users can create a password reset request for any user account.	Sending	**FR79**	User successfully creates a password reset request for a user on the admin page as they have admin permissions.	User successfully creates a password reset request for a user on the admin page as they have admin permissions.	**Pass**
T046	Admin users can delete any user account.	Sending	**FR80**	User successfully deletes a user account from the admin page as they have admin permissions.	User successfully deletes a user account from the admin page as they have admin permissions.	**Pass**
T047	Admin users are able to delete any recipe uploaded.	Sending	**FR81**	User successfully deletes a recipe from the single recipe page as they have admin permissions.	User successfully deletes a recipe from the single recipe page as they have admin permissions.	**Pass**

Table 4.2.1.1: Results of final system walkthrough executed upon completion of the final sprint.

4.2.2 API Endpoint Testing

Testing API endpoints created by the Express server during the development of the system was carried out using Postman against all endpoints created for the system (see Table 4.2.2.1). Results for the final round of testing can be found below (see Table 4.2.2.2). All test cases pass apart from T030, T118 and T119 which are instead accepted. These three tests are for delete functionality where records are not deleted, as intended, but may provide the appearance that a deletion has occurred.

Endpoint ID	HTTP Verb	Relative Path	Endpoint Description
E001	POST	/accounts	Allows user to create an account for the application.
E002	POST	/login	Allows user to log in to their account for the application.
E003	POST	/forgot-password	Allows user to create a password reset request.
E004	POST	/forgot-password/request	Allows user to retrieve their password reset request information.
E005	DELETE	/forgot-password	Allows user to delete their password reset request.
E006	GET	/account	Allows user to retrieve their account information based on decoded value in token.
E007	GET	/accounts	Allows user to get list of all account information held in the application.
E008	GET	/accounts/stats	Allows user to get count of each user type within the application.
E009	DELETE	/accounts/:id	Allows user to delete their account or admin user to delete any account.
E010	PUT	/accounts/permissions/:id	Allows user to update account permissions for a particular user by ID.
E011	PUT	/accounts/:id	Allows user to update their account information by ID.
E012	GET	/produce	Allows user to get list of all produce held in application.
E013	GET	/produce/plant/:produceName	Allows user to get information for specific produce item by name.
E014	GET	/produce/:produceType	Allows user to get list of all produce of a specific type. Search parameters can be added to obtain a tailored list based on user input for type, name, location, light, and skill level values.
E015	GET	/currently-growing/:id	Allows user to get their currently growing list using their user ID.
E016	PUT	/currently-growing/:id	Allows user to update their currently growing list using their user ID.
E017	GET	/wishlist/:id	Allows user to get their wish-list using their user ID.
E018	PUT	/wishlist/:id	Allows user to update their wish- list using their user ID.
E019	GET	/liked-recipes/:id	Allows user to get their liked recipes list using their user ID.

E020	PUT	/liked-recipes/:id	Allows user to update their liked recipes list using their user ID.
E021	GET	/recipes	Allows user to get list of all recipes. Search parameters can be added to obtain a tailored list based on dietInclude and produceInclude values.
E022	GET	/user/recipes	Allows user to get list of all recipes they have uploaded to the application.
E023	POST	/recipes	Allows user to upload new recipe to the application.
E024	GET	/recipes/:id	Allows user to get specific recipe by ID value.
E025	PUT	/recipes/:id	Allows user to update specific recipe by ID value.
E026	DELETE	/recipes/:id	Allows user to delete specific recipe by ID value.
E027	DELETE	/user/recipes/:id	Allows user to delete all recipes they have uploaded to the application.

Table 4.2.2.1: Table showing an overview of all API endpoints created for the application. The HTTP path and path are listed, alongside a description. Each endpoint is also allocated an ID, used in Table 4.2.2.2.

Test ID	Test Description	Endpoint ID	Req. Tested	Elapsed Time	Expected Result	Actual Result	Pass / Accept / Fail
T001	Request made without Authorization token present	**E012 (any endpoints not E001, E002, E003 or E004)**	**NR03, NR04**	11 ms	401 error status containing object stating, "No authorization token was found".	401 error status containing object stating, "No authorization token was found".	**Pass**
T002	Request made with expired Authorisation token	**E012 (any endpoints not E001, E002, E003 or E004)**	**NR03, NR04**	25 ms	401 error status containing object stating "jwt expired".	401 error status containing object stating "jwt expired".	**Pass**
T003	User sign up with no email value	**E001**	**FR01**	16 ms	404 error status containing object stating "email, username and password values must be included in request".	404 error status containing object stating "email, username and password values must be included in request".	**Pass**

T004	User sign up with no username value	**E001**	**FR01**	14 ms	404 error status containing object stating “email, username and password values must be included in request”.	404 error status containing object stating “email, username and password values must be included in request”.	**Pass**
T005	User sign up with no password value	**E001**	**FR01**	9 ms	404 error status containing object stating “email, username and password values must be included in request”.	404 error status containing object stating “email, username and password values must be included in request”.	**Pass**
T006	User sign up with email already in system	**E001**	**FR01**	86 ms	404 error status containing object stating, “This email is already in use”.	404 error status containing object stating, “This email is already in use”.	**Pass**
T007	User sign up with invalid email	**E001**	**FR01**	91 ms	404 error status containing object stating, “Please enter a valid email address”.	404 error status containing object stating, “Please enter a valid email address”.	**Pass**
T008	User sign up with username already in system	**E001**	**FR01**	101 ms	404 error status containing object stating, “This username is already in use”.	404 error status containing object stating, “This username is already in use”.	**Pass**
T009	User sign up with invalid password	**E001**	**FR01**	119 ms	404 error status containing object stating, “Please enter a password that is at least 8 letters long, contains 1 number, 1 uppercase character and 1 lowercase character”.	404 error status containing object stating, “Please enter a password that is at least 8 letters long, contains 1 number, 1 uppercase character and 1 lowercase character”.	**Pass**

T010	Successful user sign up	**E001**	**FR01**	513 ms	201 success status containing object including message stating “Successfully added new user” and the new user ID.	201 success status containing object including message stating “Successfully added new user” and the new user ID.	**Pass**
T011	User sign up with no email value	**E002**	**FR02**	12 ms	404 error status containing object stating, “email and password values must be included in request”.	404 error status containing object stating, “email and password values must be included in request”.	**Pass**
T012	User sign up with no password value	**E002**	**FR02**	25 ms	404 error status containing object stating, “email and password values must be included in request”.	404 error status containing object stating, “email and password values must be included in request”.	**Pass**
T013	User sign up with invalid email	**E002**	**FR02**	129 ms	404 error status containing object stating, “Invalid email address entered”.	404 error status containing object stating, “Invalid email address entered”.	**Pass**
T014	User sign up with invalid password	**E002**	**FR02**	173 ms	404 error status containing object stating, “Invalid password entered”.	404 error status containing object stating, “Invalid password entered”.	**Pass**
T015	Successful user login	**E002**	**FR02**	373 ms	201 success status containing object including message stating “Successfully authenticated”, token and user object.	201 success status containing object including message stating “Successfully authenticated”, token and user object.	**Pass**
T016	Forgot password request with no email value	**E003**	**FR02, FR79**	12 ms	404 error status containing object stating, “email value must be included in request”.	404 error status containing object stating, “email value must be included in request”.	**Pass**

T017	Forgot password request with email that is not in system	**E003**	**FR02, FR79**	90 ms	404 error status containing object stating, “This email address does not exist in our system”.	404 error status containing object stating, “This email address does not exist in our system”.	**Pass**
T018	Forgot password request with email that request has already been made for	**E003**	**FR02, FR79**	363 ms	404 error status containing object stating, “A reset request has already been made for this email address”.	404 error status containing object stating, “A reset request has already been made for this email address”.	**Pass**
T019	Successful password reset request	**E003**	**FR02, FR79**	1827 ms	201 success status containing password reset request object which was created.	201 success status containing password reset request object which was created.	**Pass**
T020	Get forgot password request with no ID value	**E004**	**FR02, FR79**	9 ms	404 error status containing object stating, “ID value must be included in request”.	404 error status containing object stating, “ID value must be included in request”.	**Pass**
T021	Get forgot password request with invalid ID	**E004**	**FR02, FR79**	62 ms	404 error status containing object stating, “Invalid ID has been supplied”.	404 error status containing object stating, “Invalid ID has been supplied”.	**Pass**
T022	Get forgot password request with ID that does not exist	**E004**	**FR02, FR79**	49 ms	404 error status containing object stating, “Reset request does not exist”.	404 error status containing object stating, “Reset request does not exist”.	**Pass**
T023	Successful password reset request retrieval	**E004**	**FR02, FR79**	49 ms	200 success status containing password reset request object.	200 success status containing password reset request object.	**Pass**
T024	Delete forgot password request with no ID value	**E005**	**FR02, FR79**	10 ms	404 error status containing object stating, “ID, email and password values must be included in request”.	404 error status containing object stating, “ID, email and password values must be included in request”.	**Pass**

T025	Delete forgot password request with no email value	**E005**	**FR02, FR79**	7 ms	404 error status containing object stating, “ID, email and password values must be included in request”.	404 error status containing object stating, “ID, email and password values must be included in request”.	**Pass**
T026	Delete forgot password request with no password value	**E005**	**FR02, FR79**	6 ms	404 error status containing object stating, “ID, email and password values must be included in request”.	404 error status containing object stating, “ID, email and password values must be included in request”.	**Pass**
T027	Delete forgot password request with invalid ID	**E005**	**FR02, FR79**	8 ms	404 error status containing object stating, “Invalid ID has been supplied”.	404 error status containing object stating, “Invalid ID has been supplied”.	**Pass**
T028	Delete forgot password request with invalid password	**E005**	**FR02, FR79**	30 ms	404 error status containing object stating, “Please enter a password that is at least 8 letters long, contains 1 number, 1 uppercase character and 1 lowercase character”.	404 error status containing object stating, “Please enter a password that is at least 8 letters long, contains 1 number, 1 uppercase character and 1 lowercase character”.	**Pass**
T029	Delete forgot password request with ID for user which does not exist in system	**E005**	**FR02, FR79**	37 ms	404 error status containing object stating, “User does not exist”.	404 error status containing object stating, “User does not exist”.	**Pass**
T030	Delete forgot password request with reset request ID which does not exist in system	**E005**	**FR02, FR79**	186 ms	404 error status containing object stating, “Reset request does not exist”.	204 success status containing empty object.	**Accept**
T031	Successful password reset request deletion	**E005**	**FR02, FR79**	167 ms	204 success status containing empty object.	204 success status containing empty object.	**Pass**

T032	Successful account request	**E006**	**FR04, FR12, FR16, FR50, FR53**	69 ms	200 success status containing object including token and user object.	200 success status containing object including token and user object.	**Pass**
T033	Accounts request where user is not an admin	**E007**	**FR75, FR77**	9 ms	404 error status containing object stating, “You must be an admin to make this request”.	404 error status containing object stating, “You must be an admin to make this request”.	**Pass**
T034	Successful accounts request	**E007**	**FR75, FR77**	40 ms	200 success status containing object including result.	200 success status containing object including result.	**Pass**
T035	Account stats request where user is not an admin	**E008**	**FR75, FR76**	9 ms	404 error status containing object stating, “You must be an admin to make this request”.	404 error status containing object stating, “You must be an admin to make this request”.	**Pass**
T036	Successful account stats request	**E008**	**FR75, FR76**	163 ms	200 success status containing object including result.	200 success status containing object including result.	**Pass**
T037	Delete account request with invalid ID	**E009**	**FR05, FR75, FR80**	8 ms	404 error status containing object stating, “Invalid ID has been supplied”.	404 error status containing object stating, “Invalid ID has been supplied”.	**Pass**
T038	Delete account request where user does not exist in system	**E009**	**FR05, FR75, FR80**	43 ms	404 error status containing object stating, “User does not exist”.	404 error status containing object stating, “User does not exist”.	**Pass**
T039	Delete account request where user ID does not match current user	**E009**	**FR05, FR75, FR80**	42 ms	404 error status containing object stating, “You do not have permission to make this request”.	404 error status containing object stating, “You do not have permission to make this request”.	**Pass**
T040	Delete account request where user is not admin	**E009**	**FR05, FR75, FR80**	40 ms	404 error status containing object stating, “You do not have permission to make this request”.	404 error status containing object stating, “You do not have permission to make this request”.	**Pass**

T041	Successful account deletion where it is the user's account	**E009**	**FR05, FR75, FR80**	105 ms	204 success status containing empty object.	204 success status containing empty object.	**Pass**
T042	Successful account deletion where user is admin	**E009**	**FR05, FR75, FR80**	124 ms	204 success status containing empty object.	204 success status containing empty object.	**Pass**
T043	Toggle permission request with invalid ID	**E010**	**FR75, FR78**	8 ms	404 error status containing object stating, "Invalid ID has been supplied".	404 error status containing object stating, "Invalid ID has been supplied".	**Pass**
T044	Toggle permission where user is not admin	**E010**	**FR75, FR78**	18 ms	404 error status containing object stating, "You must be an admin to make this request".	404 error status containing object stating, "You must be an admin to make this request".	**Pass**
T045	Toggle permission where user ID does not exist	**E010**	**FR75, FR78**	57 ms	404 error status containing object stating, "User does not exist".	404 error status containing object stating, "User does not exist".	**Pass**
T046	Successful toggle permission	**E010**	**FR75, FR78**	86 ms	201 success status containing object including message stating "User permission successfully updated".	201 success status containing object including message stating "User permission successfully updated".	**Pass**
T047	Update account request with invalid ID	**E011**	**FR04**	30 ms	404 error status containing object stating, "Invalid ID has been supplied".	404 error status containing object stating, "Invalid ID has been supplied".	**Pass**
T048	Update account request with no email value	**E011**	**FR04**	32 ms	404 error status containing object stating, "email and password values must be included in request".	404 error status containing object stating, "email and password values must be included in request".	**Pass**
T049	Update account request with no password value	**E011**	**FR04**	16 ms	404 error status containing object stating, "email and password values must be included in request".	404 error status containing object stating, "email and password values must be included in request".	**Pass**

T050	Update account request where user ID does not match current user	**E011**	**FR04**	23 ms	404 error status containing object stating, “You do not have permission to make this request”.	404 error status containing object stating, “You do not have permission to make this request”.	**Pass**
T051	Update account request with email already in system	**E011**	**FR04**	121 ms	404 error status containing object stating, “This email address is already in use”.	404 error status containing object stating, “This email address is already in use”.	**Pass**
T052	Update account request where user ID does not exist	**E011**	**FR04**	72 ms	404 error status containing object stating, “User does not exist”.	404 error status containing object stating, “User does not exist”.	**Pass**
T053	Successful account update	**E011**	**FR04**	554 ms	201 success status containing object including result.	201 success status containing object including result.	**Pass**
T054	Successful retrieval of all produce information	**E012**	**FR06, FR07, FR09, FR20, FR23**	238 ms	200 success status containing object including result.	200 success status containing object including result.	**Pass**
T055	Successful retrieval of single produce item information for Tomato	**E013**	**FR06, FR10**	45 ms	200 success status containing object including result.	200 success status containing object including result.	**Pass**
T056	Successful retrieval of all produce for type fruit	**E014**	**FR06, FR08, FR20, FR25**	48 ms	200 success status containing object including result.	200 success status containing object including result.	**Pass**
T057	Successful retrieval of all produce which are vegetables, can be grown indoors in part sun, for novices who dislike carrots	**E014**	**FR06, FR08, FR20, FR25**	42 ms	200 success status containing object including result.	200 success status containing object including result.	**Pass**
T058	Currently growing list retrieval where user ID is invalid	**E015**	**FR12**	10 ms	404 error status containing object stating, “Invalid ID has been supplied”.	404 error status containing object stating, “Invalid ID has been supplied”.	**Pass**

T059	Currently growing list retrieval where username does not match current user	**E015**	**FR12**	48 ms	404 error status containing object stating, “You do not have permission to make this request”.	404 error status containing object stating, “You do not have permission to make this request”.	**Pass**
T060	Currently growing list retrieval where ID does not exist in system	**E015**	**FR12**	45 ms	404 error status containing object stating, “User does not exist”.	404 error status containing object stating, “User does not exist”.	**Pass**
T061	Successful retrieval of user currently growing list	**E015**	**FR12**	59 ms	200 success status containing object including result.	404 error status containing object stating, “Invalid ID has been supplied”.	**Pass**
T062	Currently growing list update where user ID is invalid	**E016**	**FR11, FR13**	14 ms	404 error status containing object stating, “Invalid ID has been supplied”.	404 error status containing object stating, “Invalid ID has been supplied”.	**Pass**
T063	Currently growing list update with no name value	**E016**	**FR11, FR13**	7 ms	404 error status containing object stating, “name value is required in this request”.	404 error status containing object stating, “name value is required in this request”.	**Pass**
T064	Currently growing list update where username does not match current user	**E016**	**FR11, FR13**	75 ms	404 error status containing object stating, “You do not have permission to make this request”.	404 error status containing object stating, “You do not have permission to make this request”.	**Pass**
T065	Currently growing list update where ID does not exist in system	**E016**	**FR11, FR13**	64 ms	404 error status containing object stating, “User does not exist”.	404 error status containing object stating, “User does not exist”.	**Pass**
T066	Currently growing list update where produce name does not exist in system	**E016**	**FR11, FR13**	42 ms	404 error status containing object stating, “Produce item is not in our system”.	404 error status containing object stating, “Produce item is not in our system”.	**Pass**
T067	Successful currently growing list update	**E016**	**FR11, FR13**	155 ms	200 success status containing object including result.	200 success status containing object including result.	**Pass**

T068	Wish-list retrieval where user ID is invalid	**E017**	**FR16**	8 ms	404 error status containing object stating, “Invalid ID has been supplied”.	404 error status containing object stating, “Invalid ID has been supplied”.	**Pass**
T069	Wish-list retrieval where username does not match current user	**E017**	**FR16**	45 ms	404 error status containing object stating, “You do not have permission to make this request”.	404 error status containing object stating, “You do not have permission to make this request”.	**Pass**
T070	Wish-list retrieval where ID does not exist in system	**E017**	**FR16**	46 ms	404 error status containing object stating, “User does not exist”.	404 error status containing object stating, “User does not exist”.	**Pass**
T071	Successful retrieval of user wish-list	**E017**	**FR16**	49 ms	200 success status containing object including result.	200 success status containing object including result.	**Pass**
T072	Wish-list update where user ID is invalid	**E018**	**FR15, FR17**	8 ms	404 error status containing object stating, “Invalid ID has been supplied”.	404 error status containing object stating, “Invalid ID has been supplied”.	**Pass**
T073	Wish-list update with no name value	**E018**	**FR15, FR17**	10 ms	404 error status containing object stating, “name value is required in this request”.	404 error status containing object stating, “name value is required in this request”.	**Pass**
T074	Wish-list update where username does not match current user	**E018**	**FR15, FR17**	79 ms	404 error status containing object stating, “You do not have permission to make this request”.	404 error status containing object stating, “You do not have permission to make this request”.	**Pass**
T075	Wish-list update where ID does not exist in system	**E018**	**FR15, FR17**	65 ms	404 error status containing object stating, “User does not exist”.	404 error status containing object stating, “User does not exist”.	**Pass**
T076	Wish-list update where produce name does not exist in system	**E018**	**FR15, FR17**	49 ms	404 error status containing object stating, “Produce item is not in our system”.	404 error status containing object stating, “Produce item is not in our system”.	**Pass**

T077	Successful wish-list update	**E018**	**FR15, FR17**	184 ms	200 success status containing object including result.	200 success status containing object including result.	**Pass**
T078	Liked recipes list retrieval where user ID is invalid	**E019**	**FR50**	6 ms	404 error status containing object stating, “Invalid ID has been supplied”.	404 error status containing object stating, “Invalid ID has been supplied”.	**Pass**
T079	Liked recipes list retrieval where username does not match current user	**E019**	**FR50**	63 ms	404 error status containing object stating, “You do not have permission to make this request”.	404 error status containing object stating, “You do not have permission to make this request”.	**Pass**
T080	Liked recipes list retrieval where ID does not exist in system	**E019**	**FR50**	40 ms	404 error status containing object stating, “User does not exist”.	404 error status containing object stating, “User does not exist”.	**Pass**
T081	Successful retrieval of user liked recipes list	**E019**	**FR50**	45 ms	200 success status containing object including result.	200 success status containing object including result.	**Pass**
T082	Liked recipes list update where user ID is invalid	**E020**	**FR49, FR51**	51 ms	404 error status containing object stating, “Invalid ID has been supplied”.	404 error status containing object stating, “Invalid ID has been supplied”.	**Pass**
T083	Liked recipes list update where recipe ID is invalid	**E020**	**FR49, FR51**	9 ms	404 error status containing object stating, “Invalid recipe ID has been supplied”.	404 error status containing object stating, “Invalid recipe ID has been supplied”.	**Pass**
T084	Liked recipes list update with no id value	**E020**	**FR49, FR51**	8 ms	404 error status containing object stating, “id value is required in this request”.	404 error status containing object stating, “id value is required in this request”.	**Pass**
T085	Liked recipes list update where username does not match current user	**E020**	**FR49, FR51**	106 ms	404 error status containing object stating, “You do not have permission to make this request”.	404 error status containing object stating, “You do not have permission to make this request”.	**Pass**

T086	Liked recipes list update where ID does not exist in system	**E020**	**FR49, FR51**	193 ms	404 error status containing object stating, “User does not exist”.	404 error status containing object stating, “User does not exist”.	**Pass**
T087	Liked recipes list update where recipe ID does not exist in system	**E020**	**FR49, FR51**	49 ms	404 error status containing object stating, “Recipe is not in our system”.	404 error status containing object stating, “Recipe is not in our system”.	**Pass**
T088	Successful liked recipes list update	**E020**	**FR49, FR51**	421 ms	200 success status containing object including result.	200 success status containing object including result.	**Pass**
T089	Successful retrieval of all recipes	**E021**	**FR38, FR47, FR48**	72 ms	200 success status containing object including result.	200 success status containing object including result.	**Pass**
T090	Successful retrieval of all recipes which include the tomato produce item	**E021**	**FR38, FR47, FR48**	42 ms	200 success status containing object including result.	200 success status containing object including result.	**Pass**
T091	Successful retrieval of all recipes which include the tomato produce item and is a Vegan diet	**E021**	**FR38, FR47, FR48**	46 ms	200 success status containing object including result.	200 success status containing object including result.	**Pass**
T092	Successful retrieval of all recipes which include the Vegan diet	**E021**	**FR38, FR47, FR48**	52 ms	200 success status containing object including result.	200 success status containing object including result.	**Pass**
T093	Successful retrieval of all recipes uploaded by current user	**E022**	**FR46, FR53**	60 ms	200 success status containing object including result.	200 success status containing object including result.	**Pass**
T094	Recipe upload where title field is missing	**E023**	**FR52**	6 ms	404 error status containing object stating, “Error inserting new recipe due to missing fields”.	404 error status containing object stating, “Error inserting new recipe due to missing fields”.	**Pass**

T095	Recipe upload where title field is a number	**E023**	**FR52**	18 ms	404 error status containing object stating, “title, description, uploaded, imageName and image must be strings”.	404 error status containing object stating, “title, description, uploaded, imageName and image must be strings”.	**Pass**
T096	Recipe upload where produceInclude is empty	**E023**	**FR52**	10 ms	404 error status containing object stating, “produceInclude, instructions and ingredients must not be empty”.	404 error status containing object stating, “produceInclude, instructions and ingredients must not be empty”.	**Pass**
T097	Recipe upload where likes field is a string	**E023**	**FR52**	8 ms	404 error status containing object stating, “likes value must be a number”.	404 error status containing object stating, “likes value must be a number”.	**Pass**
T098	Recipe upload where user ID field is a number	**E023**	**FR52**	8 ms	404 error status containing object stating, “user value must be an object containing _id and username string values”.	404 error status containing object stating, “user value must be an object containing _id and username string values”.	**Pass**
T099	Successful recipe upload	**E023**	**FR52**	46 ms	200 success status containing object including result.	200 success status containing object including result.	**Pass**
T100	Single recipe retrieval where ID is invalid	**E024**	**FR39**	8 ms	404 error status containing object stating, “Invalid ID has been supplied”.	404 error status containing object stating, “Invalid ID has been supplied”.	**Pass**
T101	Single recipe retrieval where recipe does not exist in system	**E024**	**FR39**	43 ms	404 error status containing object stating, “Recipe does not exist”.	404 error status containing object stating, “Recipe does not exist”.	**Pass**
T102	Successful single recipe retrieval	**E024**	**FR39**	42 ms	200 success status containing object including result.	200 success status containing object including result.	**Pass**
T103	Recipe update where ID is invalid	**E024**	**FR39**	11 ms	404 error status containing object stating, “Invalid ID has been supplied”.	404 error status containing object stating, “Invalid ID has been supplied”.	**Pass**

T104	Recipe update where recipe ID does not exist	**E024**	**FR39**	130 ms	404 error status containing object stating, “Recipe does not exist”.	404 error status containing object stating, “Recipe does not exist”.	**Pass**
T105	Recipe update where user ID for recipe does not match current user	**E024**	**FR39**	81 ms	404 error status containing object stating, “You do not have permission to make this request”.	404 error status containing object stating, “You do not have permission to make this request”.	**Pass**
T106	Recipe update where title field is missing	**E025**	**FR54**	13 ms	404 error status containing object stating, “Error inserting new recipe due to missing fields”.	404 error status containing object stating, “Error inserting new recipe due to missing fields”.	**Pass**
T107	Recipe update where title field is a number	**E025**	**FR54**	21 ms	404 error status containing object stating, “title, description, uploaded, imageName and image must be strings”.	404 error status containing object stating, “title, description, uploaded, imageName and image must be strings”.	**Pass**
T108	Recipe update where produceInclude is empty	**E025**	**FR54**	30 ms	404 error status containing object stating, “produceInclude, instructions and ingredients must not be empty”.	404 error status containing object stating, “produceInclude, instructions and ingredients must not be empty”.	**Pass**
T109	Recipe update where likes field is a string	**E025**	**FR54**	11 ms	404 error status containing object stating, “likes value must be a number”.	404 error status containing object stating, “likes value must be a number”.	**Pass**
T110	Recipe update where user ID field is a number	**E025**	**FR54**	25 ms	404 error status containing object stating, “user value must be an object containing _id and username string values”.	404 error status containing object stating, “user value must be an object containing _id and username string values”.	**Pass**
T111	Successful recipe update	**E025**	**FR54**	144 ms	200 success status containing object including result.	200 success status containing object including result.	**Pass**

T112	Delete single recipe where ID is invalid	**E026**	**FR55, FR81**	13 ms	404 error status containing object stating, “Invalid ID has been supplied”.	404 error status containing object stating, “Invalid ID has been supplied”.	**Pass**
T113	Delete single recipe where user ID for recipe does not match current user	**E026**	**FR55, FR81**	106 ms	404 error status containing object stating, “You do not have permission to make this request”.	404 error status containing object stating, “You do not have permission to make this request”.	**Pass**
T114	Delete single recipe where user is not admin	**E026**	**FR55, FR75, FR81**	102 ms	404 error status containing object stating, “You do not have permission to make this request”.	404 error status containing object stating, “You do not have permission to make this request”.	**Pass**
T115	Delete single recipe where recipe does not exist in system	**E026**	**FR55, FR81**	74 ms	404 error status containing object stating, “Recipe does not exist”.	404 error status containing object stating, “Recipe does not exist”.	**Pass**
T116	Successful single recipe deletion	**E026**	**FR55, FR81**	117 ms	204 success status containing empty object.	204 success status containing empty object.	**Pass**
T117	Delete all user recipes where ID is invalid	**E027**	**FR05**	31 ms	404 error status containing object stating, “Invalid ID has been supplied”.	404 error status containing object stating, “Invalid ID has been supplied”.	**Pass**
T118	Delete all user recipes where user ID does not match current user	**E027**	**FR05**	87 ms	404 error status containing object stating, “You do not have permission to make this request”.	204 success status containing empty object.	**Accept**
T119	Delete all user recipes where user ID does not exist in system	**E027**	**FR05**	91 ms	404 error status containing object stating, “User does not exist”.	204 success status containing empty object.	**Accept**
T120	Successful user recipes deletion	**E027**	**FR05**	85 ms	204 success status containing empty object.	204 success status containing empty object.	**Pass**

Table 4.2.2.2: Results of final system walkthrough executed upon completion of the final sprint. Endpoint IDs are allocated from those mapped in Table 4.2.2.1.xs

4.3 Additional Evidence

4.3.1 Automated API Endpoint Testing

In addition to manually testing endpoints created (see Figure 4.2.2.2) it is possible to generate automated test suites within Postman. While evaluating each endpoint independently, running multiple tests consecutively can provide insight into how they interact with each other. Test suites performed are grouped by feature and data returned in responses can be stored as global variables. These can be reused in subsequent endpoint tests.

The test cases executed (see Figures 4.3.1.1, 4.3.1.2, 4.3.1.3, 4.3.1.4, 4.3.1.5 and 4.3.1.6) follow intended successful API calls across the entirety of the application. Each request has between one and three test cases, depending on the anticipated response. The first test case checks the response status, the second the type of value returned, typically an array or object, and the third verifies key names returned in objects (see Figure 4.3.1.1).

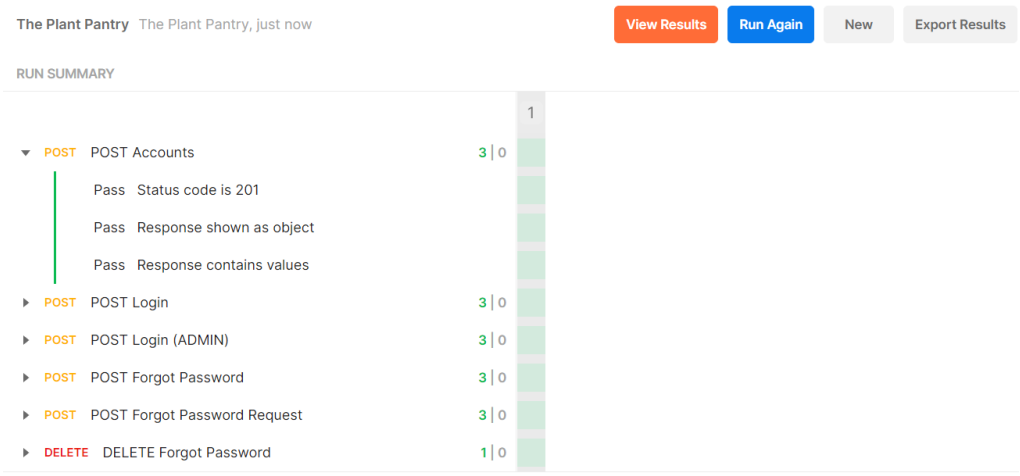


Figure 4.3.1.1: Execution of automated tests for account functionality relating to signing up, logging in, and resetting passwords.

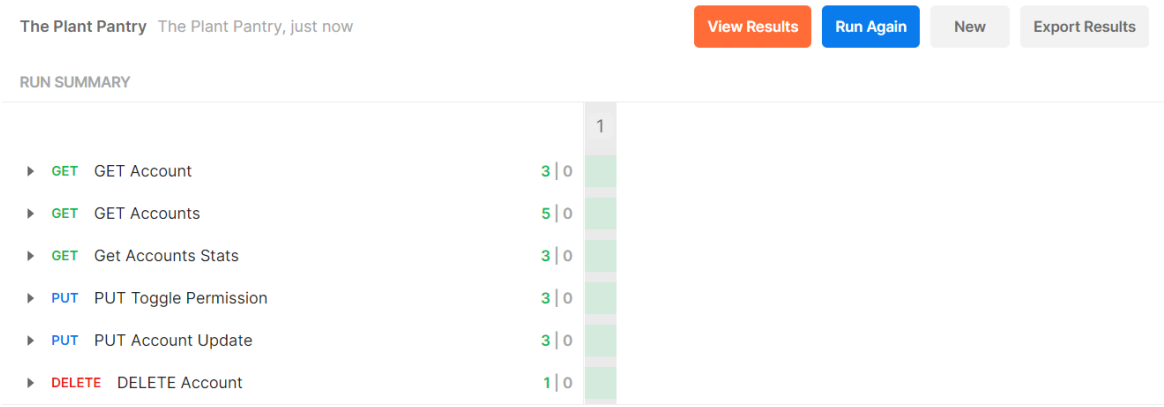


Figure 4.3.1.2: Execution of automated tests for account and admin functionality relating to retrieving, updating, and deleting account information, getting information regarding all accounts, and updating account permissions.

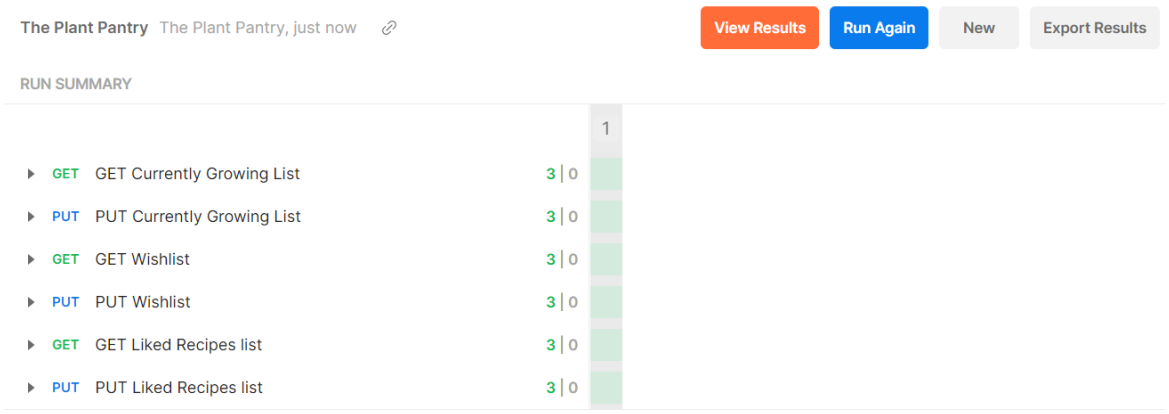


Figure 4.3.1.3: Execution of automated tests for account functionality relating to retrieving and updating currently growing lists, wish-lists, and liked recipes lists.

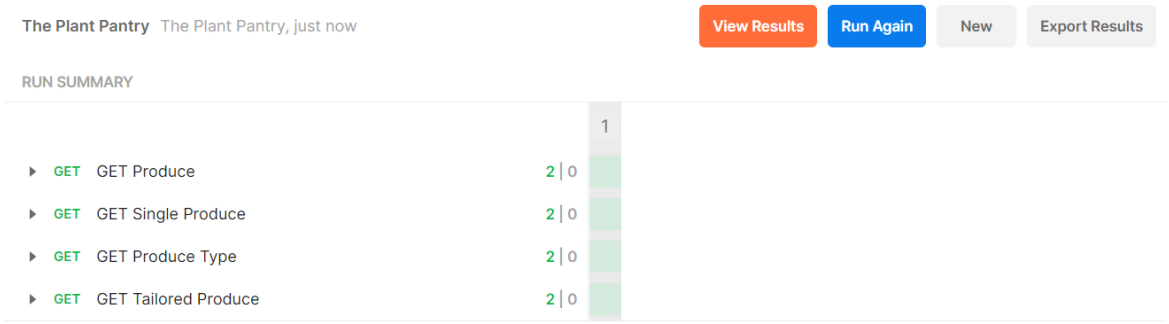


Figure 4.3.1.4: Execution of automated tests for produce functionality relating to retrieving produce lists and single produce items.

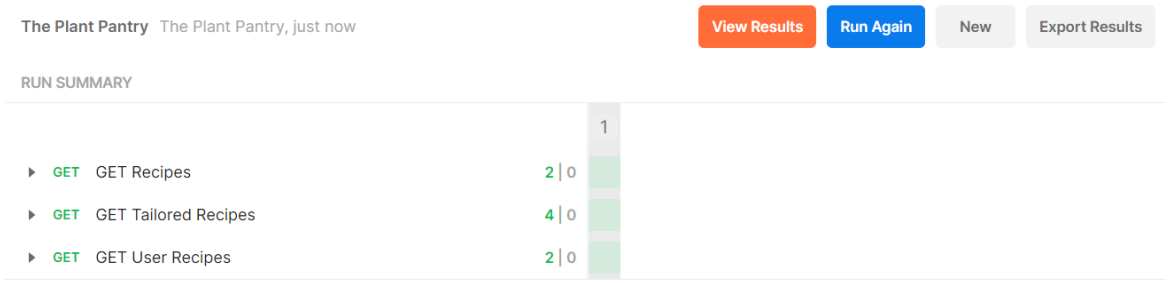


Figure 4.3.1.5: Execution of automated tests for recipe functionality relating to retrieving recipe lists.

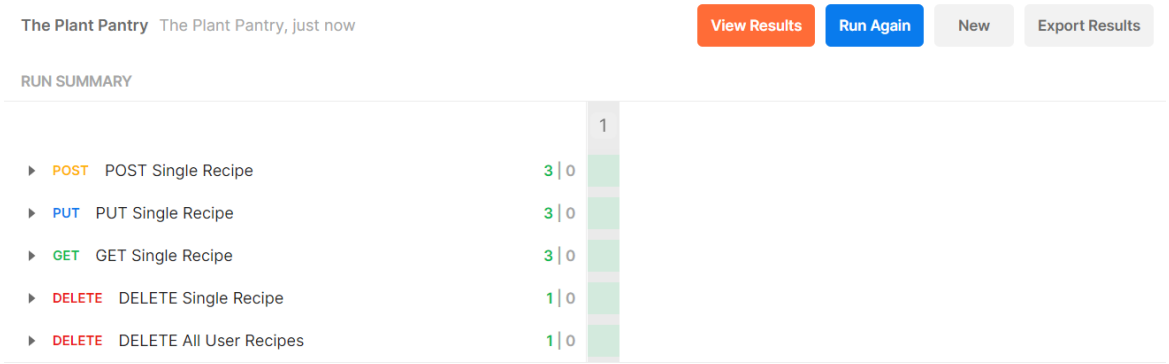
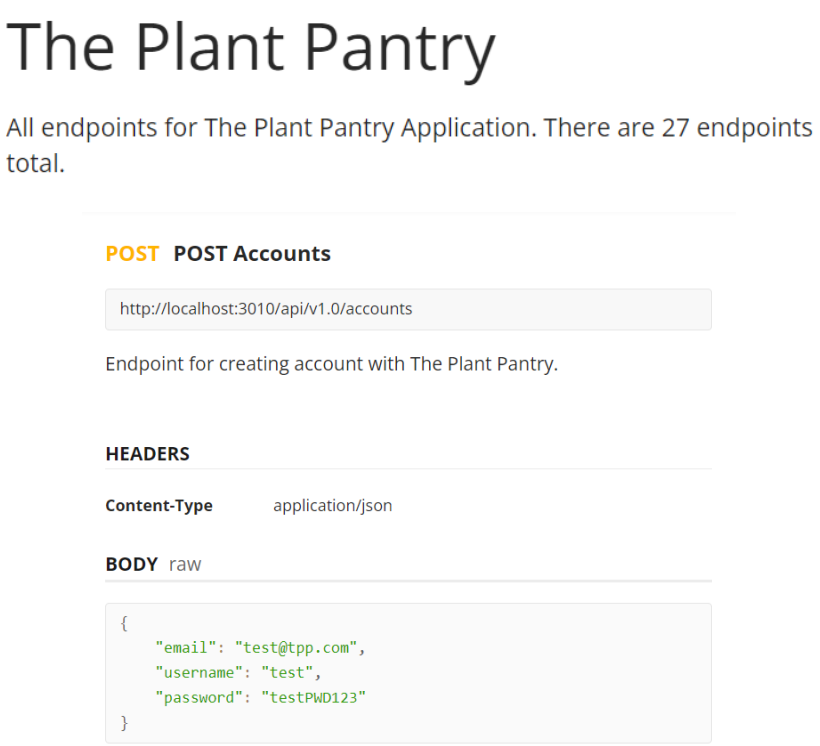


Figure 4.3.1.6: Execution of automated tests for recipe functionality relating to adding and updating recipes, retrieving single recipes, and deleting recipes.

API documentation has additionally been produced in Postman, which helped the developer to keep track of application progress and it beneficial for seeing at a glance the headers, search parameters, and values required to receive success statuses (see Figure 4.3.1.7).



POST POST Login

http://localhost:3010/api/v1.0/login

Endpoint for logging into account for The Plant Pantry.

HEADERS

Content-Type	application/json

BODY raw

```
{
    "email": "test@tpp.com",
    "password": "testPWD123"
}
```

POST POST Login (ADMIN)

http://localhost:3010/api/v1.0/login

Same endpoint as above (used in testing to login as admin).

HEADERS

Content-Type	application/json

BODY raw

```
{
    "email": "admin@tpp.com",
    "password": "adminPWD123"
}
```

POST POST Forgot Password

http://localhost:3010/api/v1.0/forgot-password

Endpoint for making a forgot password request. An email is sent to the user upon success.

HEADERS

Content-Type	application/json

BODY raw

```
{
    "email": "test@tpp.com"
}
```

POST POST Forgot Password Request

```
http://localhost:3010/api/v1.0/forgot-password/request
```

Endpoint to retrieve forgot password request.

HEADERS

Content-Type	application/json

BODY raw

```
{
    "id": "{{tpp_reset_request_id}}"
}
```

DEL DELETE Forgot Password

```
http://localhost:3010/api/v1.0/forgot-password
```

Endpoint for updating account password if forgot password request has been made. Request is then deleted.

HEADERS

Content-Type	application/json

BODY raw

```
{
    "id":"{{tpp_reset_request_id}}",
    "email": "{{tpp_reset_request_email}}",
    "password": "nstPWD123"
}
```

GET GET Account

```
http://localhost:3010/api/v1.0/account
```

Endpoint for retrieving information for a single user through the Authorization token supplied.

HEADERS

Content-Type	application/json
Authorization	Bearer {{tpp_token}}

GET GET Accounts

```
http://localhost:3010/api/v1.0/accounts
```

Endpoint for retrieving a list of all accounts in the application. Only admin users can access this endpoint.

HEADERS

Content-Type	application/json
Authorization	Bearer {{tpp_admin_token}}

GET Get Accounts Stats

```
http://localhost:3010/api/v1.0/accounts/stats
```

Endpoint for retrieving a count of users per account type in the application. Only admin users can access this endpoint.

HEADERS

Content-Type	application/json
Authorization	Bearer {{tpp_admin_token}}

DEL DELETE Account

```
http://localhost:3010/api/v1.0/accounts/{{tpp_user_id}}
```

Endpoint for deleting an application account. Admin users can delete any account while a general user is only able to remove their own.

HEADERS

Content-Type	application/json
Authorization	Bearer {{tpp_token}}

PUT PUT Toggle Permission

```
http://localhost:3010/api/v1.0/accounts/permissions/{{tpp_user_id}}
```

Endpoint for updating user permissions in the application. Only admin users can access this endpoint.

HEADERS

Content-Type	application/json
Authorization	Bearer {{tpp_admin_token}}

PUT PUT Account Update

```
http://localhost:3010/api/v1.0/accounts/{{tpp_user_id}}
```

Endpoint for updating email and password information for an account. The information can only be updated by the user who is the account holder.

HEADERS

Content-Type	application/json
Authorization	Bearer {{tpp_token}}

BODY raw

```
{
    "email": "test@tpp.com",
    "password": "testPWD123"
}
```

GET GET Produce

```
http://localhost:3010/api/v1.0/produce
```

Endpoint for retrieving a list of all produce items in the application.

HEADERS

Authorization	Bearer {{tpp_token}}
Content-Type	application/json

GET GET Produce

```
http://localhost:3010/api/v1.0/produce
```

Endpoint for retrieving a list of all produce items in the application.

HEADERS

Authorization	Bearer {{tpp_token}}
Content-Type	application/json

GET GET Produce Type

```
http://localhost:3010/api/v1.0/produce/fruit
```

Endpoint for retrieving a list of all produce items in the application of a specific type.

HEADERS

Authorization	Bearer {{tpp_token}}
Content-Type	application/json

GET GET Tailored Produce

```
http://localhost:3010/api/v1.0/produce/vegetable?location=indoor&light=part%20sun&notInclude=Carrot&skill_level=novice
```

Endpoint for retrieving a list of all produce items in the application meeting specific criteria.

HEADERS

Authorization	Bearer {{tpp_token}}
Content-Type	application/json

PARAMS

location	indoor
light	part%20sun
notInclude	Carrot
skill_level	novice

GET GET Currently Growing List

```
http://localhost:3010/api/v1.0/currently-growing/{{tpp_user_id}}
```

Endpoint for retrieving the currently growing list of a specific user. A user is only able to view their own list.

HEADERS

Authorization	Bearer {{tpp_token}}
Content-Type	application/json

PUT PUT Currently Growing List

```
http://localhost:3010/api/v1.0/currently-growing/{{tpp_user_id}}
```

Endpoint for editing the currently growing list of a specific user. A user is only able to update their own list.

HEADERS

Authorization	Bearer {{tpp_token}}
Content-Type	application/json

BODY raw

```
{
    "name": "Tomato"
}
```

GET GET Wishlist

```
http://localhost:3010/api/v1.0/wishlist/{{tpp_user_id}}
```

Endpoint for retrieving the wishlist of a specific user. A user is only able to view their own list.

HEADERS

Authorization	Bearer {{tpp_token}}
Content-Type	application/json

PUT PUT Wishlist

```
http://localhost:3010/api/v1.0/wishlist/{{tpp_user_id}}
```

Endpoint for editing the wishlist of a specific user. A user is only able to update their own list.

HEADERS

Authorization	Bearer {{tpp_token}}
Content-Type	application/json

BODY raw

```
{
    "name": "Tomato"
}
```

GET GET Liked Recipes list

```
http://localhost:3010/api/v1.0/liked-recipes/{{tpp_user_id}}
```

Endpoint for retrieving the liked recipes list of a specific user. A user is only able to view their own list.

HEADERS

Authorization	Bearer {{tpp_token}}
Content-Type	application/json

PUT PUT Liked Recipes list

```
http://localhost:3010/api/v1.0/liked-recipes/{{tpp_user_id}}
```

Endpoint for editing the liked recipes of a specific user. A user is only able to update their own list.

HEADERS

Authorization	Bearer {{tpp_token}}
Content-Type	application/json

BODY raw

```
{
    "id": "6259b523d511734930e4e95a"
}
```

GET GET Recipes

```
http://localhost:3010/api/v1.0/recipes
```

Endpoint for retrieving list of all recipes in the application.

HEADERS

Authorization	Bearer {{tpp_token}}
Content-Type	application/json

GET GET Tailored Recipes

```
http://localhost:3010/api/v1.0/recipes?produceInclude=Tomato&dietInclude=Vegan
```

Endpoint for retrieving list of all recipes in the application meeting specified criteria.

HEADERS

Authorization	Bearer {{tpp_token}}
Content-Type	application/json

PARAMS

produceInclude	Tomato
dietInclude	Vegan

GET GET User Recipes

```
http://localhost:3010/api/v1.0/user/recipes
```

Endpoint for retrieving list of all recipes in the application uploaded by a particular user.

HEADERS

Authorization	Bearer {{tpp_token}}
Content-Type	application/json

POST POST Single Recipe

```
http://localhost:3010/api/v1.0/recipes
```

Endpoint for uploading a recipe to the application.

HEADERS

Authorization	Bearer {{tpp_token}}
Content-Type	application/json

BODY raw

```
{
    "title": "title",
    "description": "description",
    "dietInclude": ["diet"],
    "produceInclude": ["produce"],
    "instructions": ["instruction"],
    "ingredients": ["ingredients"],
    "likes": 0,
    "uploaded": "2022-02
    "user": {
```

View More

GET GET Single Recipe

```
http://localhost:3010/api/v1.0/recipes/{{tpp_recipe_id}}
```

Endpoint for retrieving a single recipe in the application.

HEADERS

Authorization	Bearer {{tpp_token}}
Content-Type	application/json

PUT PUT Single Recipe

```
http://localhost:3010/api/v1.0/recipes/{{tpp_recipe_id}}
```

Endpoint for updating a single recipe in the application. Only the user who uploaded the recipe can update it.

HEADERS

Authorization	Bearer {{tpp_token}}
Content-Type	application/json

BODY raw


```
{
    "title": "title",
    "description": "description",
    "dietInclude": ["diet1"],
    "produceInclude": ["produce"],
    "instructions": ["instruction"],
    "ingredients": ["ingredients"],
    "likes": 0,
    "uploaded": "2022-02
    "user": {
```


View More

DEL DELETE Single Recipe

```
http://localhost:3010/api/v1.0/recipes/{{tpp_recipe_id}}
```

Endpoint for deleting a single recipe in the application. A user can delete any recipe they have uploaded and an admin user can remove any recipe uploaded.

HEADERS

Authorization	Bearer {{tpp_token}}
Content-Type	application/json

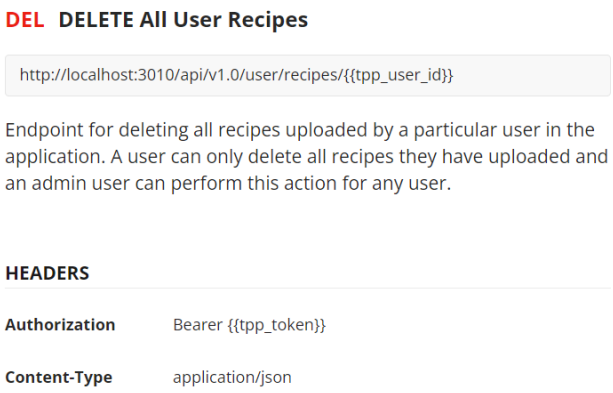


Figure 4.3.1.7: Documentation created for the API developed. Each endpoint outlined has a description discussing its use within the application, alongside information on values required for successful interactions.

4.3.2 Code Review

Code review consisted of reading code files created and modified during each interaction in an attempt to catch errors prior to running the code. In addition to error discovery, code review proved useful in facilitating the refactoring and formatting of code to achieve a professional codebase.

Dedicated time for reviewing code was allocated within every sprint, occurring after the requirement had left the 'In Progress' stage but prior to it entering the 'Testing' stage (see Figure 4.3.2.1). Maintaining a Kanban board enabled requirement focus and a visual representation of application development progress. Multiple boards were created, separated by iteration to support the agile process, and contained a set of requirements for completion within the allotted time. Code review occurred within the period for each sprint and all the requirements listed originated from the product backlog (see Figure 4.3.2.2).

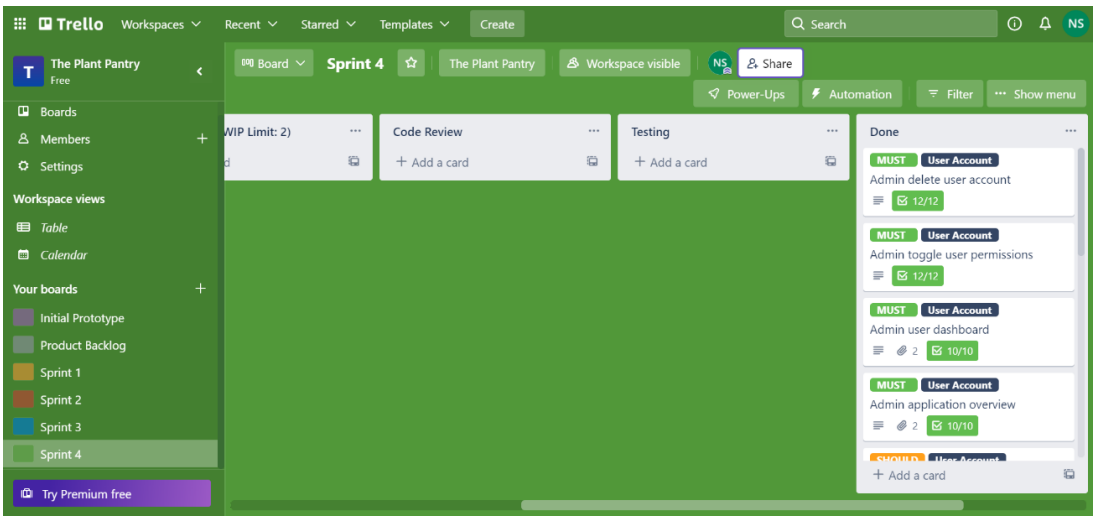


Figure 4.3.2.1: Screenshot of the Trello Kanban board created for the fourth development sprint upon conclusion of the iteration, so all requirements are in the 'Done' column. The 'Code Review' area is visible, and all requirements had to complete this stage before progressing to 'Testing'.

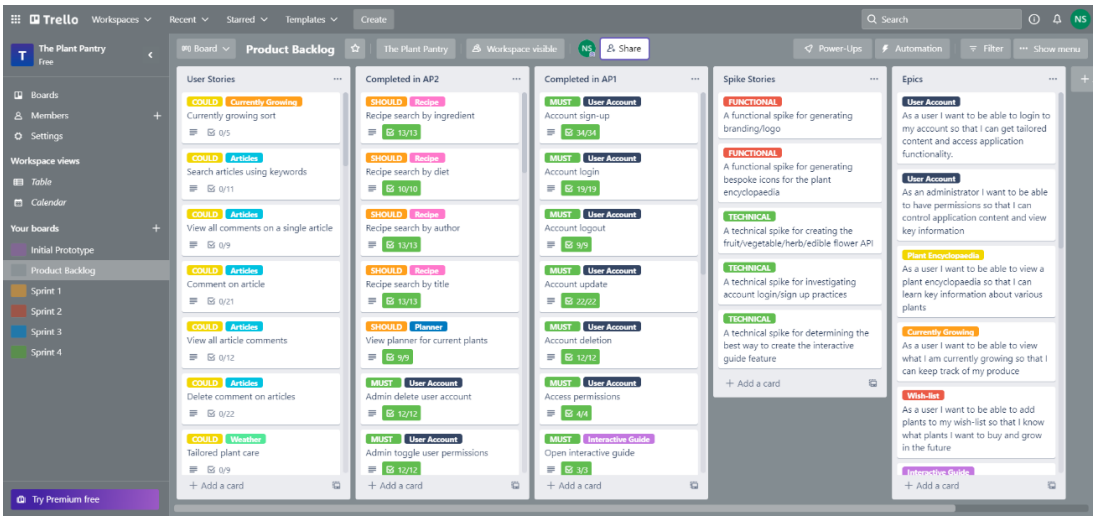


Figure 4.3.2.2: Screenshot of the Trello Kanban board created for the product backlog upon conclusion of all iterations executed. The requirements completed during AP2 have been moved to an allocated column, similarly to those for AP1. Lists containing user stories not yet implemented, spike stories and epics are also present.

4.3.3 Unit Tests

As discussed earlier in the report, unit tests were generated using Jest and Enzyme to facilitate testing of the JavaScript based code. Unit tests are a form of white-box testing, used to ensure all aspects of the developed system work as expected and run without error (Hamilton, 2021; Reeves, 2020).

Numerous test files were created, with a specific focus on the front-end code given the execution of manual and automated testing for all API endpoints using Postman. These files relate to all React components created (see Figure 4.3.3.1), action types (see Figure 4.3.3.2), action creator functions (see Figure 4.3.3.3), Redux reducers (see Figure 4.3.3.4), and other JavaScript (see Figure 4.3.3.5) and React files (see Figure 4.3.3.6). Each file possesses one parent test case, containing multiple smaller cases, a format suiting React components which contain specific functionality.

```
PASS src/client/components/containers/Produce/ProduceHome/ProduceHome.test.js
  ProduceHome
    √ renders (17ms)
    √ renders Button (41ms)
    √ renders ButtonContainer (13ms)
    √ renders TextInput (7ms)
    √ renders Card (8ms)
    √ renders CardContainer (8ms)
    √ componentDidMount (2ms)
    √ onButtonClick - type is all (6ms)
    √ onButtonClick - type is herb (4ms)
    √ onChange (3ms)

Test Suites: 1 passed, 1 total
Tests:       10 passed, 10 total
Snapshots:   0 total
Time:        5.977s, estimated 6s
Ran all test suites matching /ProduceHome/i.

Watch Usage: Press w to show more.
```

Figure 4.3.3.1: Screenshot of the results of the ProduceHome.test.js file when executed in the terminal in VS Code. All test cases have been successfully completed within the test suite, taking an estimated 6 seconds.

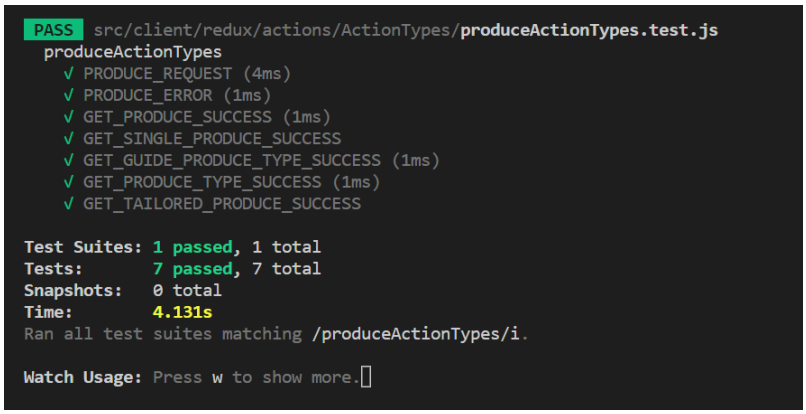


Figure 4.3.3.2: Screenshot of the results of the produceActionTypes.test.js file when executed in the terminal in VS Code. All test cases have been successfully completed within the test suite, taking approximately 4 seconds to execute.

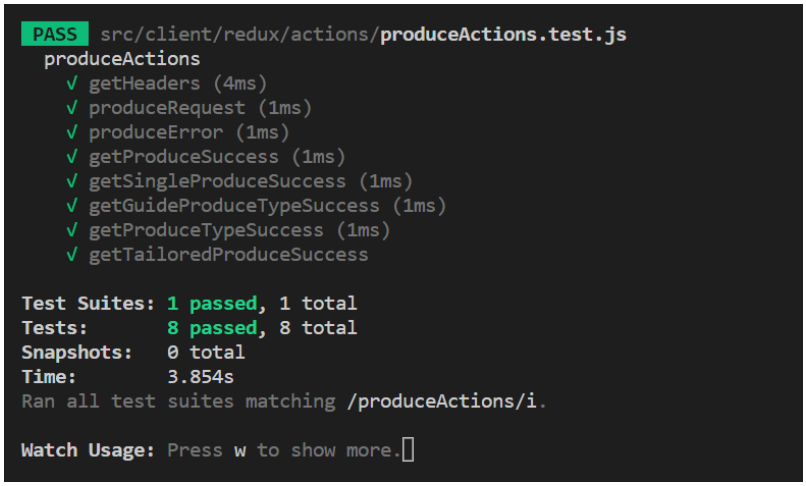


Figure 4.3.3.3: Screenshot of the results of the produceActions.test.js file when executed in the VS Code terminal. All test cases have been successfully completed within the test suite, finishing in under 4 seconds.

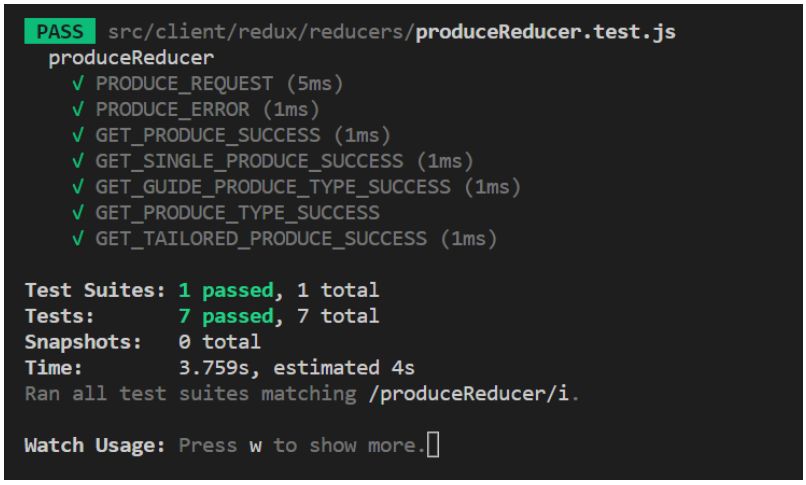


Figure 4.3.3.4: Screenshot of the results of the produceReducer.test.js file when executed in the VS Code terminal. All test cases have been successfully completed within the test suite, finishing in under 4 seconds.

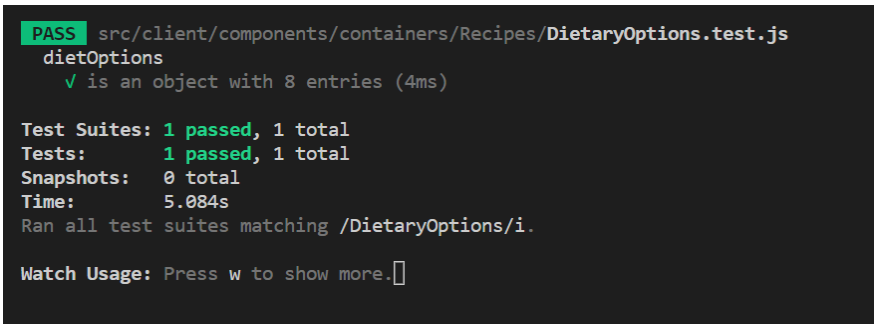


Figure 4.3.3.5: Screenshot of the results of the DietaryOptions.test.js file when executed in the VS Code terminal. The single test case has been successful, taking just over 5 seconds to complete.

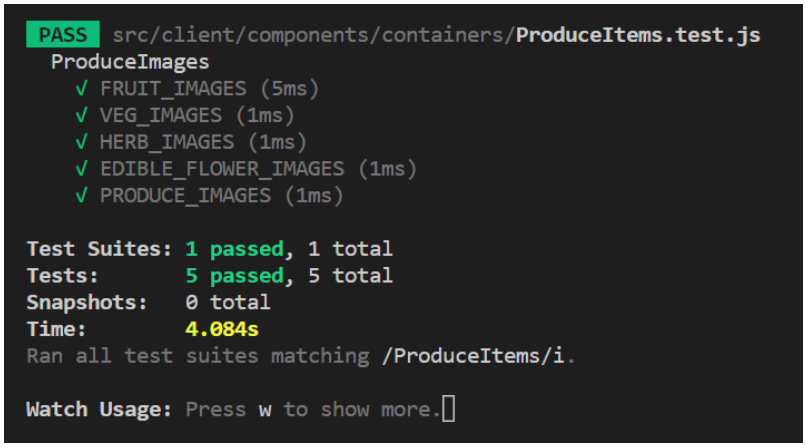


Figure 4.3.3.6: Screenshot of the results of the ProduceItems.test.js file when executed in the terminal in VS Code. All test cases have been successfully completed within the test suite, taking approximately 4 seconds.

4.4 Verification Confirmation Statement

Based on the results of various forms of verification testing, it can be concluded that the application developed meets the requirements specified. Evidence to support this claim has been provided for methods used, including walkthroughs, manual and automated API endpoint testing, code review and unit tests.

Verification success is highlighted in the passing of all tests during the final system walkthrough (see Table 4.2.1.1) and the highly favourable performance of all API endpoint testing conducted. All requirements during each iteration made it past the 'Code Review' stage, conveying effective time management, and unit tests generated successfully evaluate the front-end.

5.0: System Validation

5.1 Reflection on the Validation Plan and Execution

Executing a validation plan guarantees that the developed application meets the expectations of users within the capacity they would use it. A strong user focus has been applied across all other areas of the development process, and enlisting it here ensured the continuity. It is a complementary approach to verification and build upon the evaluation of the final product against defined requirements (Hamilton, 2021).

There are two main areas outlined in section 5.2, which are User Acceptance Testing (UAT) and usability testing. These methods directly interact with end users (Reassemble, 2019), facilitating effective validation. User Acceptance Testing is carried out against the requirements and is foundational to usability testing (Pandit and Tahilani, 2015), which is oriented towards task execution (Reassemble, 2019). Usability testing additionally contributes to the evaluation of WCAG 2.1 accessibility guidelines (W3C, 2018).

It is of utmost importance to evaluate the implementation of non-functional requirements, which are harder to evaluate in some ways as they are more opinion based than their functional counterparts. Due to this interpretation, assessment of the non-functional criteria can be found in section 5.3. Thorough consideration has been given to the aesthetic, performance, and overall nature of the application, as indicated in section 2.2.2 of this report. The system developed was be investigated in relation to the usability standards outlined by Nielsen (1994a) and Krug (2005), alongside Gestalt principles (Chapman, 2022) and colour theory (Corrigan, 2022; Khazanova, 2022).

5.2 System Validation Results

5.2.1 User Acceptance Testing (UAT)

Fundamental to application testing, UAT is a manually performed in agile environments and is a process motivated by “presentation, demonstration, probing, usability and validation” (Pandit and Tahilani, 2015, pp.16). Acceptance criteria is evaluated at this stage, derived from requirements, user stories and epics (Pandit and Tahilani, 2015; Reassemble, 2019).

All requirements have been included in the product backlog created in Trello (see Figure 4.3.2.2). Each card is reflective of a requirement, with coloured tags applied to represent the feature it corresponds to and the MoSCow priority level associated, and a user story has been created within every card to further establish the requirement purpose.

Acceptance criteria has been implemented in the form of user scenarios (see Figures 5.2.1.1 and 5.2.1.2), presented as checklists. During user acceptance testing, which takes place after code review (see Figure 4.3.2.1), each element of the scenario is marked as complete if it is successful during the requirement assessment.

This method has been applied throughout the whole duration of the project (see Figure 5.2.1.1 and 5.2.1.2) and the final set of all requirements implemented have been included in Appendix A, following the VOLERE format.

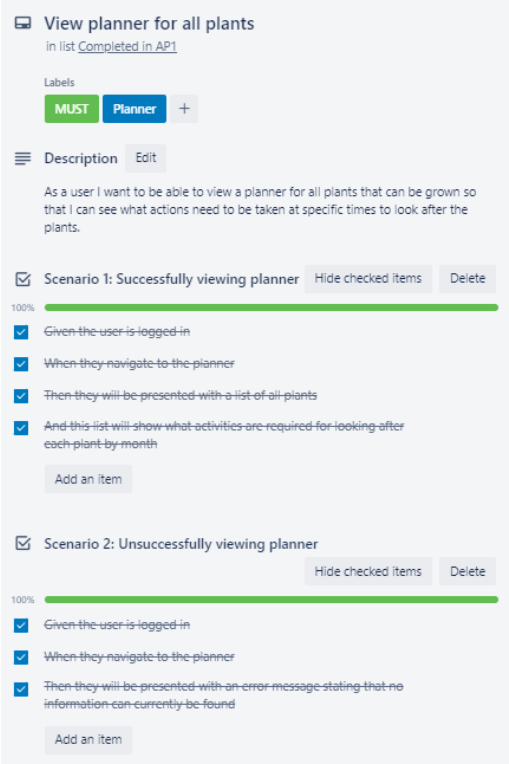


Figure 5.2.1.1: Screenshot of the requirement information for viewing the planner, found in the product backlog on Trello. This was completed during initial prototype implementation and all acceptance criteria has been met during user acceptance testing.

Figure 5.2.1.2: Screenshot of the requirement information for adding a produce item to the wish-list, found in the product backlog on Trello. This was completed during the second sprint and all acceptance criteria has been successfully met during user acceptance testing.

5.2.2 Usability Testing

The complementary component to user acceptance testing, usability testing involves getting end users to complete a set of application tasks to validate whether developer and user requirement interpretation aligns. This user interaction is invaluable in providing "fuzzy" data (Reassemble, 2019), as it is less technical that UAT, which is crucial in determining if user expectations for the system have been met.

During these tests, users were asked to vocalise their reasoning and opinions, following the Think Aloud Protocol (Lewis, 2014; Nielsen, 1994b), enabling deeper evaluation of human performance for each task executed. This information is contained within the notes and observations columns in the results tables for each round of testing, and had been considered in discussing recommendations for fixes required. Issues found during testing were allocated a severity rating (see Figure 5.2.2.1), to convey the scale of the problem discovered, and an ease of fix ranking (see Figure 5.2.2.2), to declare the difficulty of implementing a solution.

Severity Ratings	
Rating	**Definition**
0	Not a usability problem.
1	This is a quality problem, such as a cosmetic issue or spelling error.
2	A usability problem which will frustrate some users but will not affect task completion.
3	A usability problem which will slow down some users during common task completion and is to be fixed as soon as possible.
4	A usability problem which will make some users unable or unwilling to complete a common task and is to be fixed as soon as possible.

Figure 5.2.2.1: Nielsen's (Modified from: 1994b) severity ratings scale for issues discovered during testing.

Ease of Fixing Rankings	
Rating	**Definition**
0	Problem would be extremely easy to fix and could be completed before the next release.
1	Problem would be easy to fix, involving specific interface elements and the solution is clear.
2	Problem would require some effort to fix. It involves multiple aspects of the interface, would require a team of developers to implement changes before the next release, or the solution is unclear.
3	Problem is difficult to fix, requiring concentrated effort to finish before the next release. It involves multiple aspects of the interface and the solutions may not be immediately obvious or may be disrupted.

Figure 5.2.2.2: Dino, D'Amato, and Tennant's (Modified from: 2005) method for determining the ease of fixing issues detected during usability testing.

There are five sets of results displayed below, representative of each round of testing that occurred. Usability testing occurred after development of the initial prototype and completion of sprints 1 to 4, conforming to Krug's (2005) attitude towards testing. There were three participants within this testing process (see Figure 5.2.2.3), possessing varying levels of technical expertise and reasons for application interest.

ID	Age	Gender	Purpose of application use	Technology expertise
001	29	Male	Interested in growing different types of vegetables	Expert
002	27	Female	Wants to learn how to grow produce	Proficient
003	57	Female	Interested to discover what recipes they can make with homegrown produce	Advanced Beginner

Table 5.2.2.3: Overview of each participant involved across all rounds of usability testing.

User acceptance testing has the additional benefit of helping to establish whether the developed application complies with the accessibility principles defined in WCAG 2.1 (W3C, 2018). These standards determine that a system is truly user-friendly if it is perceivable, operable, understandable, and robust (see Figure 2.2.2.5).

Evaluation of user performance during this testing process can assist in verifying this fulfilment, with evidence of this can be found below.

Initial Prototype Usability Testing

User ID	Task ID	Task Description	Elapsed Time	Notes / Observations	Success Criteria	Maximum Time
001	**T1**	Make an account and login to the application	1 min 3 secs	Navigated easily to sign up page, error message for invalid password shown when did not meet requirements.	Successfully creates an application account and signs in using the details supplied.	2 mins
001	**T2**	Find what vegetables can be grown in a greenhouse in part sun	1 min 54 secs	Took a while to figure out what the guide icon was – navigated to produce and planner first.	Successfully opens the interactive guide, supplied the information from the task description and obtains appropriate results.	2 mins
001	**T3**	Find the best time to plant tomatoes from the planner	24 secs	Navigated to produce page, searched for tomato in input bar and clicked through to tomato page.	Successfully navigates to the planner or produce page and finds the information regarding tomato plants.	1 min
001	**T4**	Find the article about different types of soil for growing plants	8 secs	Successfully navigated to articles page from navigation bar and through to 'Dish The Dirt' article.	Successfully navigates to the articles section and locates the article on soil types. Clicks into article.	1 min
001	**T5**	Logout of application	4 secs	Logs out of application successfully.	Successfully logs out of application	1 min
002	**T1**	Login to the application	35 secs	Easily completed form – types quickly.	Successfully creates an application account and signs in using the details supplied.	2 mins
002	**T2**	Find what vegetables can be grown in a greenhouse in part sun	36 secs	Initially went to produce section before finding the guide.	Successfully opens the interactive guide, supplied the information from the task description and obtains appropriate results.	2 mins
002	**T3**	Find the best time to plant tomatoes	38 secs	Filtered the planner by fruit initially which delayed this (tomato is classified as a vegetable in the application).	Successfully navigates to the planner and finds the information regarding tomato plants.	1 min
002	**T4**	Find the article about different types of soil for growing plants	4 secs	Easily navigated to articles section and found relevant option.	Successfully navigates to the articles section and locates the article on soil types. Clicks into article.	1 min
002	**T5**	Logout of application	2 secs	Easily logged out of the application.	Successfully logs out of application	1 min

003	**T1**	Login to the application	56 secs	Likes the layout of the forms. Achieved without error messages shown.	Successfully creates an application account and signs in using the details supplied.	2 mins
003	**T2**	Find what vegetables can be grown in a greenhouse in part sun	68 secs	Initially tried clicking on the produce image on the homepage. Also went back to change value as clicked the wrong option.	Successfully opens the interactive guide, supplied the information from the task description and obtains appropriate results.	2 mins
003	**T3**	Find the best time to plant tomatoes	46 secs	Easily navigated to the planner and scrolled down to find results.	Successfully navigates to the planner and finds the information regarding tomato plants.	1 min
003	**T4**	Find the article about different types of soil for growing plants	45 secs	Easily found the article 'Dish the Dirt'.	Successfully navigates to the articles section and locates the article on soil types. Clicks into article.	1 min
003	**T5**	Logout of application	10 secs	Easily logged out of the application.	Successfully logs out of application	1 min

Table 5.2.2.2: Detailed results from the usability tests carried out for the initial prototype of the application. The five tasks are related to the requirements implemented at this stage and are carried out by all three participants.

Participant	Task 1	Task 2	Task 3	Task 4	Task 5	Total Success by Participant	Completion Rates by Participant
001	**x**	**x**	**x**	**x**	**x**	5	100%
002	**x**	**x**	**x**	**x**	**x**	5	100%
003	**x**	**x**	**x**	**x**	**x**	5	100%
Total Success by Task	3	3	3	3	3	**Total Success**	15
Completion Rates by Task	100%	100%	100%	100%	100%	**Total Average Success**	100%

Table 5.2.2.3: Successful task completion rates by participant (Modified from: Usability.gov, 2022) from the testing on the initial prototype (see Table 5.2.2.2). All tasks were completed successfully by all participants with a total average success rate of 100%.

Participant	Task 1	Task 2	Task 3	Task 4	Task 5	Total Time for Each Participant in secs (for completed tasks)	Average Time for Each Participant in secs (if all tasks completed)
001	63	114	24	8	4	213	43
002	35	36	38	4	2	115	23
003	56	68	46	45	10	225	45
Total Time for Each Task in secs (if completed)	154	218	108	57	16		
Average Time for Each Task in secs (if completed)	51	73	36	19	5		

Table 5.2.2.4: Calculations for surrounding participant and average times taken to complete tasks (Modified from: Usability.gov, 2022). Fields are filled in if participants successfully accomplished tasks within the time frame allocated (see Table 5.2.2.2). Task 2 took the longest on average, probably given the number of steps within the interactive guide.

Participant ID	Task	Task Ratings	Very difficult	Difficult	Neutral	Somewhat Easy	Easy
001	1	Ease of finding information					**X**
001	1	Keep track of location in application					**X**
001	1	Accurate predict application information				**X**	
001	2	Ease of finding information		**X**			
001	2	Keep track of location in application					**X**
001	2	Accurate predict application information				**X**	
001	3	Ease of finding information					**X**
001	3	Keep track of location in application					**X**
001	3	Accurate predict application information				**X**	
001	4	Ease of finding information					**X**
001	4	Keep track of location in application					**X**
001	4	Accurate predict application information					**X**
001	5	Ease of finding information					**X**
001	5	Keep track of location in application					**X**
001	5	Accurate predict application information					**X**

002	1	Ease of finding information					**X**
002	1	Keep track of location in application					**X**
002	1	Accurate predict application information					**X**
002	2	Ease of finding information				**X**	
002	2	Keep track of location in application				**X**	
002	2	Accurate predict application information				**X**	
002	3	Ease of finding information					**X**
002	3	Keep track of location in application					**X**
002	3	Accurate predict application information					**X**
002	4	Ease of finding information					**X**
002	4	Keep track of location in application					**X**
002	4	Accurate predict application information					**X**
002	5	Ease of finding information					**X**
002	5	Keep track of location in application					**X**
002	5	Accurate predict application information					**X**
003	1	Ease of finding information					**X**
003	1	Keep track of location in application					**X**
003	1	Accurate predict application information					**X**
003	2	Ease of finding information					**X**
003	2	Keep track of location in application					**X**
003	2	Accurate predict application information					**X**
003	3	Ease of finding information					**X**
003	3	Keep track of location in application					**X**
003	3	Accurate predict application information					**X**
003	4	Ease of finding information					**X**
003	4	Keep track of location in application					**X**
003	4	Accurate predict application information					**X**
003	5	Ease of finding information					**X**
003	5	Keep track of location in application					**X**
003	5	Accurate predict application information					**X**

Table 5.2.2.5: Ratings allocated by participants to each task upon usability test completion (Modified from: Usability.gov, 2022). Scores are given on a Likert scale, ranging from very difficult to easy, and each task is evaluated against the criteria of the ease of finding information, keeping track of location in the system, and accurately predicting application information.

Task	Ease of finding information	Keep track of location in application	Accurate predict application information	Overall Satisfaction (per task)
1	5(100%)	5(100%)	4.6 (100%)	4.9
2	3.6 (66%)	4.6 (100%)	4.3 (100%)	4.2
3	5 (100%)	4.6 (100%)	5 (100%)	4.9
4	5(100%)	5(100%)	5(100%)	5
5	5(100%)	5(100%)	5(100%)	5
Overall Satisfaction (Per option)	4.7	4.8	4.8	

Table 5.2.2.6: Post-test mean task ratings and percentage agreement based on responses recorded in Table 5.2.2.5 (Modified from: Usability.gov, 2022). Percentage agreements are a combination of agree and strongly agree. Satisfaction ratings are high across all tasks with tasks 4 and 5 being the greatest.

Sprint 1 Usability Testing

User ID	Task ID	Task Description	Elapsed Time	Notes / Observations	Success Criteria	Maximum Time
001	**T1**	Navigate to the produce section	5 secs	Found the Produce link from the navigation bar quickly.	Successfully navigates to the produce section by clicking the 'produce' link within the navigation bar at the top of the application.	1 min
001	**T2**	Filter the produce section to only show all herbs	6 secs	Used button at the top of the page.	Successfully filters the produce section by clicking the 'Herbs' button at the top of the page.	1 min
001	**T3**	Find the herb 'Parsley' within the herb produce list	13 secs	Scrolled down the page to find parsley and clicked into it.	Successfully finds 'Parsley' within the produce list by using the search bar or scrolling down the list.	2 mins
001	**T4**	View the produce information for the herb 'Basil'.	18 secs	Scrolled down the page to find basil and clicked into it.	Successfully finds 'Basil' within the produce list, clicks on the herb and is shown all information held about the produce item.	2 mins
002	**T1**	Navigate to the produce section	5 secs	Easily found produce page.	Successfully navigates to the produce section by clicking the 'produce' link within the navigation bar at the top of the application.	1 min

002	**T2**	Filter the produce section to only show all herbs	3 secs	Used buttons at the top to filter by herbs.	Successfully filters the produce section by clicking the 'Herbs' button at the top of the page.	1 min
002	**T3**	Find the herb 'Parsley' within the herb produce list	2 secs	Used the search bar to find parsley.	Successfully finds 'Parsley' within the produce list by using the search bar or scrolling down the list.	2 mins
002	**T4**	View the produce information for the herb 'Basil'.	10 secs	Filtered by herbs and found basil at the top of the list – found easily.	Successfully finds 'Basil' within the produce list, clicks on the herb and is shown all information held about the produce item.	2 mins
003	**T1**	Navigate to the produce section	3 secs	Easily navigated to produce page from navigation bar.	Successfully navigates to the produce section by clicking the 'produce' link within the navigation bar at the top of the application.	1 min
003	**T2**	Filter the produce section to only show all herbs	2 secs	Used button at the top of the page to filter results.	Successfully filters the produce section by clicking the 'Herbs' button at the top of the page.	1 min
003	**T3**	Find the herb 'Parsley' within the herb produce list	18 secs	Used input bar to filter results.	Successfully finds 'Parsley' within the produce list by using the search bar or scrolling down the list.	2 mins
003	**T4**	View the produce information for the herb 'Basil'.	13 secs	Used input bar to filter results.	Successfully finds 'Basil' within the produce list, clicks on the herb and is shown all information held about the produce item.	2 mins

Table 5.2.2.7: Detailed results from the usability tests carried out for sprint 1. The four tasks are related to the requirements implemented at this stage and are carried out by all three participants.

Participant	Task 1	Task 2	Task 3	Task 4	Total Success by Participant	Completion Rates by Participant
001	**x**	**x**	**x**	**x**	4	100%
002	**x**	**x**	**x**	**x**	4	100%
003	**x**	**x**	**x**	**x**	4	100%
Total Success by Task	3	3	3	3	**Total Success**	12
Completion Rates by Task	100%	100%	100%	100%	**Total Average Success**	100%

Table 5.2.2.8: Successful task completion rates by participant (Modified from: Usability.gov, 2022) from the testing on sprint 1 (see Table 5.2.2.7). All tasks were completed successfully by every participant with a total average success rate of 100%.

Participant	Task 1	Task 2	Task 3	Task 4	Total Time for Each Participant in secs (for completed tasks)	Average Time for Each Participant in secs (if all tasks completed)
001	5	6	13	18	42	11
002	5	2	3	10	20	5
003	3	2	18	13	36	9
Total Time for Each Task in secs (if completed)	13	10	34	41		
Average Time for Each Task In secs (if completed)	4	3	11	14		

Table 5.2.2.9: Calculations for surrounding participant and average times taken to complete tasks (Modified from: Usability.gov, 2022). Fields are filled in if participants successfully accomplished tasks within the time frame allocated (see Table 5.2.2.7). Tasks 1 and 2 took the shortest on average, given that they were simple navigation and filtering tasks.

Participant ID	Task	Task Ratings	Very difficult	Difficult	Neutral	Somewhat Easy	Easy
001	1	Ease of finding information					**X**
001	1	Keep track of location in application					**X**
001	1	Accurate predict application information					**X**
001	2	Ease of finding information					**X**
001	2	Keep track of location in application					**X**
001	2	Accurate predict application information					**X**
001	3	Ease of finding information					**X**
001	3	Keep track of location in application					**X**
001	3	Accurate predict application information					**X**
001	4	Ease of finding information					**X**
001	4	Keep track of location in application					**X**
001	4	Accurate predict application information					**X**
002	1	Ease of finding information					**X**
002	1	Keep track of location in application					**X**
002	1	Accurate predict application information					**X**
002	2	Ease of finding information					**X**
002	2	Keep track of location in application					**X**

002	2	Accurate predict application information					**X**
002	3	Ease of finding information					**X**
002	3	Keep track of location in application					**X**
002	3	Accurate predict application information					**X**
002	4	Ease of finding information					**X**
002	4	Keep track of location in application					**X**
002	4	Accurate predict application information					**X**
003	1	Ease of finding information					**X**
003	1	Keep track of location in application					**X**
003	1	Accurate predict application information					**X**
003	2	Ease of finding information					**X**
003	2	Keep track of location in application					**X**
003	2	Accurate predict application information					**X**
003	3	Ease of finding information					**X**
003	3	Keep track of location in application					**X**
003	3	Accurate predict application information					**X**
003	4	Ease of finding information					**X**
003	4	Keep track of location in application					**X**
003	4	Accurate predict application information					**X**

Table 5.2.2.10: Ratings allocated by participants to each task upon usability test completion for sprint 1 (Modified from: Usability.gov, 2022). Scores are given on a Likert scale, ranging from very difficult to easy, and each task is evaluated against the criteria of the ease of finding information, keeping track of location in the system, and accurately predicting application information.

Task	Ease of finding information	Keep track of location in application	Accurate predict application information	Overall Satisfaction (per task)
1	5 (100%)	5 (100%)	5 (100%)	5
2	5 (100%)	5 (100%)	5 (100%)	5
3	5 (100%)	5 (100%)	5 (100%)	5
4	5 (100%)	5 (100%)	5 (100%)	5
Overall Satisfaction (Per option)	5	5	5	

Table 5.2.2.11: Post-test mean task ratings and percentage agreement based on responses recorded in Table 5.2.2.10 (Modified from: Usability.gov, 2022). Percentage agreements are a combination of agree and strongly agree. Overall satisfaction for all options and tasks received the highest possible score.

Sprint 2 Usability Testing

User ID	Task ID	Task Description	Elapsed Time	Notes / Observations	Success Criteria	Maximum Time
001	**T1**	Find your currently growing list	1 min 58 secs	Navigated across most application areas before looking closer at the navigation bar – was confused at the naming of the 'Account' section.	Successfully navigate to the currently growing list.	2 mins
001	**T2**	Add 'Carrot' and 'Kale' to your currently growing list	20 secs	Searched in planner for produce and then navigated to single produce pages.	Successfully add 'Carrot' and 'Kale' to the currently growing list from the single produce pages.	3 mins
001	**T3**	Remove 'Carrot' from your currently growing list	12 secs	Navigated to planner and to the currently growing section within it to find carrot. Clicked through to single produce page.	Successfully remove 'Carrot' from currently growing through the single produce page or the currently growing list area.	2 mins
001	**T4**	Move 'Kale' to the history section of your currently growing list.	16 secs	Figured out where currently growing list was from earlier task and moved to history.	Successfully move 'Kale' to history section of the currently growing list from the currently growing list area.	2 mins
001	**T5**	Add 'Apple' and 'Dianthus' to your wish-list	17 secs	Searched in planner for produce and then navigated to single produce pages.	Successfully add 'Apple' and 'Dianthus' to the wish-list from the single produce pages.	3 mins
001	**T6**	Remove 'Dianthus' from your wish-list	9 secs	Navigated to planner and to the currently growing section within it to find dianthus plant. Clicked through to single produce page.	Successfully remove 'Dianthus' from the wish-list through the single produce page or the wish-list area.	2 mins
001	**T7**	View your wish-list	6 secs	Successfully navigated to wish-list.	Successfully navigate to the wish-list.	2 mins
001	**T8**	Remove 'Apple' from your wish-list	4 secs	Successfully removed apple from wish-list by navigating to single produce page.	Successfully remove 'Apple' from the wish-list through the single produce page or the wish-list area.	2 mins
002	**T1**	Find your currently growing list	26 secs	Initially went to produce section and then found account area in navigation bar – got confused.	Successfully navigate to the currently growing list.	2 mins

002	**T2**	Add ‘Carrot’ and ‘Kale’ to your currently growing list	14 secs	Used button filter for vegetables and then scrolled.	Successfully add ‘Carrot’ and ‘Kale’ to the currently growing list from the single produce pages.	3 mins
002	**T3**	Remove ‘Carrot’ from your currently growing list	8 secs	Navigated to currently growing list and removed from here.	Successfully remove ‘Carrot’ from currently growing through the single produce page or the currently growing list area.	2 mins
002	**T4**	Move ‘Kale’ to the history section of your currently growing list.	17 secs	Easily carried out task.	Successfully move ‘Kale’ to history section of the currently growing list from the currently growing list area.	2 mins
002	**T5**	Add ‘Apple’ and ‘Dianthus’ to your wish-list	2 secs	Used the input bar to filter results when looking for apple and scrolled for dianthus as was unsure of spelling.	Successfully add ‘Apple’ and ‘Dianthus’ to the wish-list from the single produce pages.	3 mins
002	**T6**	Remove ‘Dianthus’ from your wish-list	3 secs	Removed dianthus from single produce page as was already there.	Successfully remove ‘Dianthus’ from the wish-list through the single produce page or the wish-list area.	2 mins
002	**T7**	View your wish-list	2 secs	Found easily after struggling with currently growing list earlier.	Successfully navigate to the wish-list.	2 mins
002	**T8**	Remove ‘Apple’ from your wish-list	2 secs	Easily carried out task.	Successfully remove ‘Apple’ from the wish-list through the single produce page or the wish-list area.	2 mins
003	**T1**	Find your currently growing list	14 secs	Found easily from the navigation bar.	Successfully navigate to the currently growing list.	2 mins
003	**T2**	Add ‘Carrot’ and ‘Kale’ to your currently growing list	50 secs	Used search input. Liked layout of all information.	Successfully add ‘Carrot’ and ‘Kale’ to the currently growing list from the single produce pages.	3 mins
003	**T3**	Remove ‘Carrot’ from your currently growing list	20 secs	Removed this from the currently growing list.	Successfully remove ‘Carrot’ from currently growing through the single produce page or the currently growing list area.	2 mins
003	**T4**	Move ‘Kale’ to the history section of your currently growing list.	11 secs	Easily completed task.	Successfully move ‘Kale’ to history section of the currently growing list from the currently growing list area.	2 mins
003	**T5**	Add ‘Apple’ and ‘Dianthus’ to your wish-list	77 secs	Used the input bar to filter results in both cases.	Successfully add ‘Apple’ and ‘Dianthus’ to the wish-list from the single produce pages.	3 mins

003	**T6**	Remove 'Dianthus' from your wish-list	4 secs	Task completed easily.	Successfully remove 'Dianthus' from the wish-list through the single produce page or the wish-list area.	2 mins
003	**T7**	View your wish-list	28 secs	Task completed easily.	Successfully navigate to the wish-list.	2 mins
003	**T8**	Remove 'Apple' from your wish-list	4 secs	Task completed easily.	Successfully remove 'Apple' from the wish-list through the single produce page or the wish-list area.	2 mins

Table 5.2.2.12: Detailed results from the usability tests carried out for sprint 2. The eight tasks are related to requirements implemented at this stage and are carried out by all three participants.

Participant	**Task 1**	**Task 2**	**Task 3**	**Task 4**	**Task 5**	**Task 6**	**Task 7**	**Task 8**	**Total Success by Participant**	**Completion Rates by Participant**
001	x	x	x	x	x	x	x	x	8	100%
002	x	x	x	x	x	x	x	x	8	100%
003	x	x	x	x	x	x	x	x	8	100%
Total Success by Task	3	3	3	3	3	3	3	3	**Total Success**	24
Completion Rates by Task	100%	100%	100%	100%	100%	100%	100%	100%	**Total Average Success**	100%

Table 5.2.2.13: Successful task completion rates by participant (Modified from: Usability.gov, 2022) from the testing on sprint 1 (see Table 5.2.2.12). All tasks were completed successfully by every participant with a total average success rate of 100%.

Participant	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7	Task 8	Total Time for Each Participant in secs (for completed tasks)	Average Time for Each Participant in secs (if all tasks completed)
001	118	20	12	16	17	9	6	4	202	25
002	26	14	8	17	2	3	2	2	74	9
003	14	50	20	11	77	4	28	4	208	26
Total Time for Each Task in secs (if completed)	158	84	40	43	96	16	36	10		
Average Time for Each Task in secs (if completed)	53	28	13	14	32	5	12	3		

Table 5.2.2.14: Calculations for surrounding participant and average times taken to complete tasks (Modified from: Usability.gov, 2022). Fields are filled in if participants successfully accomplished tasks within the time frame allocated (see Table 5.2.2.12). Task 1 took the longest due to vocalised lack of familiarity with the application.

Participant ID	Task	Task Ratings	Very difficult	Difficult	Neutral	Somewhat Easy	Easy
001	1	Ease of finding information		**X**			
001	1	Keep track of location in application			**X**		
001	1	Accurate predict application information			**X**		
001	2	Ease of finding information					**X**
001	2	Keep track of location in application					**X**
001	2	Accurate predict application information				**X**	
001	3	Ease of finding information					**X**
001	3	Keep track of location in application					**X**
001	3	Accurate predict application information					**X**
001	4	Ease of finding information					**X**
001	4	Keep track of location in application					**X**
001	4	Accurate predict application information					**X**
001	5	Ease of finding information					**X**

001	5	Keep track of location in application					**X**
001	5	Accurate predict application information					**X**
001	6	Ease of finding information					**X**
001	6	Keep track of location in application					**X**
001	6	Accurate predict application information					**X**
001	7	Ease of finding information					**X**
001	7	Keep track of location in application					**X**
001	7	Accurate predict application information					**X**
001	8	Ease of finding information					**X**
001	8	Keep track of location in application					**X**
001	8	Accurate predict application information					**X**
002	1	Ease of finding information			**X**		
002	1	Keep track of location in application				**X**	
002	1	Accurate predict application information			**X**		
002	2	Ease of finding information					**X**
002	2	Keep track of location in application				**X**	
002	2	Accurate predict application information					**X**
002	3	Ease of finding information					**X**
002	3	Keep track of location in application					**X**
002	3	Accurate predict application information					**X**
002	4	Ease of finding information					**X**
002	4	Keep track of location in application					**X**
002	4	Accurate predict application information					**X**
002	5	Ease of finding information					**X**
002	5	Keep track of location in application					**X**
002	5	Accurate predict application information					**X**
002	6	Ease of finding information					**X**
002	6	Keep track of location in application					**X**
002	6	Accurate predict application information					**X**
002	7	Ease of finding information					**X**
002	7	Keep track of location in application					**X**
002	7	Accurate predict application information					**X**
002	8	Ease of finding information					**X**
002	8	Keep track of location in application					**X**

002	8	Accurate predict application information					**X**
003	1	Ease of finding information				**X**	
003	1	Keep track of location in application				**X**	
003	1	Accurate predict application information			**X**		
003	2	Ease of finding information				**X**	
003	2	Keep track of location in application					**X**
003	2	Accurate predict application information					**X**
003	3	Ease of finding information					**X**
003	3	Keep track of location in application					**X**
003	3	Accurate predict application information					**X**
003	4	Ease of finding information					**X**
003	4	Keep track of location in application					**X**
003	4	Accurate predict application information					**X**
003	5	Ease of finding information					**X**
003	5	Keep track of location in application					**X**
003	5	Accurate predict application information					**X**
003	6	Ease of finding information					**X**
003	6	Keep track of location in application					**X**
003	6	Accurate predict application information					**X**
003	7	Ease of finding information				**X**	
003	7	Keep track of location in application				**X**	
003	7	Accurate predict application information					**X**
003	8	Ease of finding information					**X**
003	8	Keep track of location in application					**X**
003	8	Accurate predict application information					**X**

Table 5.2.2.15: Ratings allocated by participants to each task upon usability test completion for sprint 2 (Modified from: Usability.gov, 2022). Scores are given on a Likert scale, ranging from very difficult to easy, and each task is evaluated against the criteria of the ease of finding information, keeping track of location in the system, and accurately predicting application information.

Task	Ease of finding information	Keep track of location in application	Accurate predict application information	Overall Satisfaction (per task)
1	3 (60%)	3.6 (73%)	3 (60%)	3.2
2	4.6 (93%)	4.6 (93%)	4.6 (93%)	4.6
3	5 (100%)	5 (100%)	5 (100%)	5
4	5 (100%)	5 (100%)	5 (100%)	5
5	5 (100%)	5 (100%)	5 (100%)	5
6	5 (100%)	5 (100%)	5 (100%)	5
7	4.6 (93%)	4.6 (93%)	5 (100%)	4.7
8	5 (100%)	5 (100%)	5 (100%)	5
Overall Satisfaction (Per option)	4.7	4.7	4.7	

Table 5.2.2.16: Post-test mean task ratings and percentage agreement based on responses recorded in Table 5.2.2.15 (Modified from: Usability.gov, 2022). Percentage agreements are a combination of agree and strongly agree. Overall satisfaction for task 8 received the highest possible score, with scores for finding information, keeping track of location, and predicting information accurately received the same score of 4.7.

Sprint 3 Usability Testing

User ID	Task ID	Task Description	Elapsed Time	Notes / Observations	Success Criteria	Maximum Time
001	**T1**	View all Vegan recipes	11 secs	Easily navigated to recipes section and used dropdown to filter.	Successfully uses the dropdown filter to view a list of all Vegan recipes in the application.	2 mins
001	**T2**	View the recipe for Honey-seared Melon	5 secs	Used the input bar to filter recipes based on the title.	Successfully navigates to the recipe for Honey-seared Melon.	2 mins
001	**T3**	Add the Honey-seared Melon recipe to your liked recipes	3 secs	Clicked into the recipe and added easily to liked recipes list	Successfully adds recipe to their liked recipes list by clicking the star at the top of the page.	1 min
001	**T4**	View your liked recipes list	7 secs	Navigated to account area from navigation bar, then to the liked recipes section.	Successfully navigates to profile page and views the liked recipes section.	2 mins
001	**T5**	Delete the Honey-seared recipe from your liked recipes	4 secs	Removed from liked recipes from the liked recipes section within the account area.	Successfully removes recipe from liked recipes list through the section within the profile or from the recipe page.	1 min

001	**T6**	Upload a recipe (contains dummy data supplied by developer)	2 mins 15 secs	Easily added data to the input fields. Impressed by the ability to drag and drop rows.	Successfully navigates to the upload form, supplies information for all fields and uploads the recipe to the application.	3 mins
001	**T7**	Change the title of the recipe uploaded	34 secs	Navigated to update form through the single recipe component.	Successfully navigates to the edit form and changes the title.	2 mins
001	**T8**	Delete the recipe uploaded	9 secs	Performed this action through the single recipe component.	Successfully removes the recipe through the uploads section within the user profile or from the recipe page.	2 mins
002	**T1**	View all Vegan recipes	13 secs	Quickly used dropdown to search for recipes matching this type.	Successfully uses the dropdown filter to view a list of all Vegan recipes in the application.	2 mins
002	**T2**	View the recipe for Honey-seared Melon	22 secs	Used input bar but did not spell recipe properly on initial input which caused delay.	Successfully navigates to the recipe for Honey-seared Melon.	2 mins
002	**T3**	Add the Honey-seared Melon recipe to your liked recipes	3 secs	Easily performed task – likes use of icons for this functionality.	Successfully adds recipe to their liked recipes list by clicking the star at the top of the page.	1 min
002	**T4**	View your liked recipes list	4 secs	Easily completed task.	Successfully navigates to profile page and views the liked recipes section.	2 mins
002	**T5**	Delete the Honey-seared recipe from your liked recipes	5 secs	Performed this from the liked recipes list.	Successfully removes recipe from liked recipes list through the section within the profile or from the recipe page.	1 min
002	**T6**	Upload a recipe (contains dummy data supplied by developer)	83 secs	Filled in all information correctly on first attempt. Liked the different styles of input used in form.	Successfully navigates to the upload form, supplies information for all fields and uploads the recipe to the application.	3 mins
002	**T7**	Change the title of the recipe uploaded	12 secs	Easily executed.	Successfully navigates to the edit form and changes the title.	2 mins
002	**T8**	Delete the recipe uploaded	7 secs	Completed from the liked recipes area.	Successfully removes the recipe through the uploads section within the user profile or from the recipe page.	2 mins
003	**T1**	View all Vegan recipes	34 secs	Initially tried to use input bar to filter but then read description and used dropdown instead.	Successfully uses the dropdown filter to view a list of all Vegan recipes in the application.	2 mins
003	**T2**	View the recipe for Honey-seared Melon	19 secs	Used input filter to find recipe.	Successfully navigates to the recipe for Honey-seared Melon.	2 mins

003	**T3**	Add the Honey-seared Melon recipe to your liked recipes	3 secs	Easily completed task.	Successfully adds recipe to their liked recipes list by clicking the star at the top of the page.	1 min
003	**T4**	View your liked recipes list	9 secs	Found account information from navigation bar and headings.	Successfully navigates to profile page and views the liked recipes section.	2 mins
003	**T5**	Delete the Honey-seared recipe from your liked recipes	5 secs	Easily completed from liked recipes list.	Successfully removes recipe from liked recipes list through the section within the profile or from the recipe page.	1 min
003	**T6**	Upload a recipe (contains dummy data supplied by developer)	94 secs	Accidentally forgot to include description but error message showed to sort this.	Successfully navigates to the upload form, supplies information for all fields and uploads the recipe to the application.	3 mins
003	**T7**	Change the title of the recipe uploaded	13 secs	Updated title easily – misspelt word in initial upload so fixed that.	Successfully navigates to the edit form and changes the title.	2 mins
003	**T8**	Delete the recipe uploaded	4 secs	Easily completed the task. Deleted from single recipe page.	Successfully removes the recipe through the uploads section within the user profile or from the recipe page.	2 mins

Table 5.2.2.17: Detailed results from the usability tests carried out for sprint 3. The eight tasks are related to requirements implemented at this stage and are carried out by all three participants.

Participant	**Task 1**	**Task 2**	**Task 3**	**Task 4**	**Task 5**	**Task 6**	**Task 7**	**Task 8**	**Total Success by Participant**	**Completion Rates by Participant**
001	x	x	x	x	x	x	x	x	8	100%
002	x	x	x	x	x	x	x	x	8	100%
003	x	x	x	x	x	x	x	x	8	100%
Total Success by Task	3	3	3	3	3	3	3	3	**Total Success**	24
Completion Rates by Task	100%	100%	100%	100%	100%	100%	100%	100%	**Total Average Success**	100%

Table 5.2.2.18: Successful task completion rates by participant (Modified from: Usability.gov, 2022) from the testing on sprint 3 (see Table 5.2.2.17). All tasks were completed successfully by every participant with a total average success rate of 100%.

Participant	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7	Task 8	Total Time for Each Participant in secs (for completed tasks)	Average Time for Each Participant in secs (if all tasks completed)
001	11	5	3	7	4	135	34	9	208	26
002	13	22	3	4	5	83	12	7	149	19
003	34	19	3	9	5	94	13	4	181	23
Total Time for Each Task in secs (if completed)	58	46	9	20	14	312	59	20		
Average Time for Each Task in secs (if completed)	19	25	3	7	5	104	20	7		

Table 5.2.2.19: Calculations for surrounding participant and average times taken to complete tasks (Modified from: Usability.gov, 2022). Fields are filled in if participants successfully accomplished tasks within the time frame allocated (see Table 5.2.2.17). Task 6 took the longest due as it required a lot of user input for uploading a new recipe.

Participant ID	Task	Task Ratings	Very difficult	Difficult	Neutral	Somewhat Easy	Easy
001	1	Ease of finding information					**X**
001	1	Keep track of location in application					**X**
001	1	Accurate predict application information					**X**
001	2	Ease of finding information					**X**
001	2	Keep track of location in application					**X**
001	2	Accurate predict application information					**X**
001	3	Ease of finding information				**X**	
001	3	Keep track of location in application					**X**
001	3	Accurate predict application information					**X**
001	4	Ease of finding information				**X**	
001	4	Keep track of location in application					**X**
001	4	Accurate predict application information					**X**
001	5	Ease of finding information					**X**

001	5	Keep track of location in application					**X**
001	5	Accurate predict application information					**X**
001	6	Ease of finding information				**X**	
001	6	Keep track of location in application					**X**
001	6	Accurate predict application information				**X**	
001	7	Ease of finding information					**X**
001	7	Keep track of location in application					**X**
001	7	Accurate predict application information					**X**
001	8	Ease of finding information					**X**
001	8	Keep track of location in application					**X**
001	8	Accurate predict application information					**X**
002	1	Ease of finding information				**X**	
002	1	Keep track of location in application				**X**	
002	1	Accurate predict application information					**X**
002	2	Ease of finding information					**X**
002	2	Keep track of location in application				**X**	
002	2	Accurate predict application information					**X**
002	3	Ease of finding information					**X**
002	3	Keep track of location in application					**X**
002	3	Accurate predict application information					**X**
002	4	Ease of finding information					**X**
002	4	Keep track of location in application					**X**
002	4	Accurate predict application information					**X**
002	5	Ease of finding information					**X**
002	5	Keep track of location in application					**X**
002	5	Accurate predict application information					**X**
002	6	Ease of finding information					**X**
002	6	Keep track of location in application					**X**
002	6	Accurate predict application information					**X**
002	7	Ease of finding information					**X**
002	7	Keep track of location in application					**X**
002	7	Accurate predict application information					**X**
002	8	Ease of finding information					**X**
002	8	Keep track of location in application					**X**

002	8	Accurate predict application information				**X**	
003	1	Ease of finding information			**X**		
003	1	Keep track of location in application				**X**	
003	1	Accurate predict application information					**X**
003	2	Ease of finding information					**X**
003	2	Keep track of location in application				**X**	
003	2	Accurate predict application information					**X**
003	3	Ease of finding information					**X**
003	3	Keep track of location in application					**X**
003	3	Accurate predict application information					**X**
003	4	Ease of finding information					**X**
003	4	Keep track of location in application					**X**
003	4	Accurate predict application information					**X**
003	5	Ease of finding information					**X**
003	5	Keep track of location in application					**X**
003	5	Accurate predict application information					**X**
003	6	Ease of finding information					**X**
003	6	Keep track of location in application					**X**
003	6	Accurate predict application information				**X**	
003	7	Ease of finding information					**X**
003	7	Keep track of location in application					**X**
003	7	Accurate predict application information					**X**
003	8	Ease of finding information					**X**
003	8	Keep track of location in application					**X**
003	8	Accurate predict application information					**X**

Table 5.2.2.20: Ratings allocated by participants to each task upon usability test completion for sprint 3 (Modified from: Usability.gov, 2022). Scores are given on a Likert scale, ranging from very difficult to easy, and each task is evaluated against the criteria of the ease of finding information, keeping track of location in the system, and accurately predicting application information.

Task	Ease of finding information	Keep track of location in application	Accurate predict application information	Overall Satisfaction (per task)
1	4 (80%)	4.3 (86%)	5 (100%)	4.4
2	4.6 (93%)	4.3 (86%)	5 (100%)	4.6
3	4.6 (93%)	5 (100%)	5 (100%)	4.9
4	5 (100%)	5 (100%)	5 (100%)	5
5	5 (100%)	5 (100%)	5 (100%)	5
6	4.6 (93%)	5 (100%)	4.3 (86%)	4.6
7	5 (100%)	5 (100%)	5 (100%)	5
8	5 (100%)	5 (100%)	4.6 (93%)	4.9
Overall Satisfaction (Per option)	4.7	4.8	4.9	

Table 5.2.2.21: Post-test mean task ratings and percentage agreement based on responses recorded in Table 5.2.2.20 (Modified from: Usability.gov, 2022). Percentage agreements are a combination of agree and strongly agree. Overall satisfaction for tasks 3 and 8 were highest, as they were for simpler elements of functionality, with accurately predicting application information scoring 4.9, slightly greater than the other tasks.

Sprint 4 Usability Testing

User ID	Task ID	Task Description	Elapsed Time	Notes / Observations	Success Criteria	Maximum Time
001	**T1**	Navigate to the admin application overview and find the count of vegetarian recipes.	14 secs	Easily found the admin area from the navigation bar and understood the information presented in the charts.	Successfully navigates to the application overview section in the admin area and finds the count of vegetarian recipes.	2 mins
001	**T2**	Navigate to the user dashboard and find all general users	8 secs	Easily navigated to user dashboard and clicked on button to filter the users.	Successfully navigates to the user dashboard and filters the list by general users.	1 min
001	**T3**	Send a password reset request to the account for nstevenson	3 secs	Clicked password reset button within the table for this user.	Successfully sends password reset request to the account for nstevenson with success message showing	1 min
001	**T4**	Update user permissions for the account for nstevenson	3 secs	Clicked toggle permissions button within the table for this user.	Successfully updates user permission for the account for nstevenson, making them an admin user.	1 min

001	**T5**	Delete the account for nstevenson	4 secs	Clicked delete account button within the table for this user and confirmed action through the modal which appears.	Successfully removes account for nstevenson from the application.	1 min
001	**T6**	Navigate to the planner and find the growing information for elderflower	7 secs	Navigated to the planner and used input bar to filter planner results.	Successfully navigates to the planner and either scrolls down the page or uses the input filter to search for elderflower.	2 mins
001	**T7**	View the planner for all information in currently growing list	2 secs	Easily found this section – clicking on the button at the top of the page.	Successfully views planner for plants in currently growing list by clicking button at the top of the page	2 mins
002	**T1**	Navigate to the admin application overview and find the count of vegetarian recipes.	12 secs	Took time reading the charts to find correct value. Felt it was obvious once figured out.	Successfully navigates to the application overview section in the admin area and finds the count of vegetarian recipes.	2 mins
002	**T2**	Navigate to the user dashboard and find all general users	5 secs	Used buttons to filter users.	Successfully navigates to the user dashboard and filters the list by general users.	1 min
002	**T3**	Send a password reset request to the account for nstevenson	4 secs	Easily executed.	Successfully sends password reset request to the account for nstevenson with success message showing	1 min
002	**T4**	Update user permissions for the account for nstevenson	3 secs	Easily completed – understood functionality better once carried out.	Successfully updates user permission for the account for nstevenson, making them an admin user.	1 min
002	**T5**	Delete the account for nstevenson	3 secs	Task easily completed – liked having the modal as a safety net.	Successfully removes account for nstevenson from the application.	1 min
002	**T6**	Navigate to the planner and find the growing information for elderflower	7 secs	Used the input bar to search for elderflower and clicked into it.	Successfully navigates to the planner and either scrolls down the page or uses the input filter to search for elderflower.	2 mins
002	**T7**	View the planner for all information in currently growing list	5 secs	Easily found button to filter the planner to show the list	Successfully views planner for plants in currently growing list by clicking button at the top of the page	2 mins

003	**T1**	Navigate to the admin application overview and find the count of vegetarian recipes.	43 secs	Did not initially understand the charts – felt it made sense once they figured out what each referred to.	Successfully navigates to the application overview section in the admin area and finds the count of vegetarian recipes.	2 mins
003	**T2**	Navigate to the user dashboard and find all general users	20 secs	Easily navigated to the dashboard and then realised about the button filtering.	Successfully navigates to the user dashboard and filters the list by general users.	1 min
003	**T3**	Send a password reset request to the account for nstevenson	3 secs	Easily executed task.	Successfully sends password reset request to the account for nstevenson with success message showing	1 min
003	**T4**	Update user permissions for the account for nstevenson	4 secs	Easily executed task.	Successfully updates user permission for the account for nstevenson, making them an admin user.	1 min
003	**T5**	Delete the account for nstevenson	4 secs	Accidentally clicked wrong account at first – was relieved the modal existed.	Successfully removes account for nstevenson from the application.	1 min
003	**T6**	Navigate to the planner and find the growing information for elderflower	13 secs	Filtered planner results by herbs and then used search bar.	Successfully navigates to the planner and either scrolls down the page or uses the input filter to search for elderflower.	2 mins
003	**T7**	View the planner for all information in currently growing list	6 secs	Used the button at the top to change planner view.	Successfully views planner for plants in currently growing list by clicking button at the top of the page	2 mins

Table 5.2.2.22: Detailed results from the usability tests carried out for sprint 4. The seven tasks are related to requirements implemented at this stage and are carried out by all three participants.

Participant	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7	Total Success by Participant	Completion Rates by Participant
001	x	x	x	x	x	x	x	8	100%
002	x	x	x	x	x	x	x	8	100%
003	x	x	x	x	x	x	x	8	100%
Total Success by Task	3	3	3	3	3	3	3	**Total Success**	24
Completion Rates by Task	100%	100%	100%	100%	100%	100%	100%	**Total Average Success**	100%

Table 5.2.2.23: Successful task completion rates by participant (Modified from: Usability.gov, 2022) from the testing on sprint 4 (see Table 5.2.2.22). All tasks were completed successfully by every participant with a total average success rate of 100%.

Participant	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7	Total Time for Each Participant in secs (for completed tasks)	Average Time for Each Participant in secs (if all tasks completed)
001	14	8	3	3	4	7	2	41	6
002	12	5	4	3	3	7	5	39	6
003	43	20	3	4	4	13	6	93	13
Total Time for Each Task in secs (if completed)	69	33	10	10	11	27	13		
Average Time for Each Task in secs (if completed)	23	11	3	3	4	9	4		

Table 5.2.2.24: Calculations for surrounding participant and average times taken to complete tasks (Modified from: Usability.gov, 2022). Fields are filled in if participants successfully accomplished tasks within the time frame allocated (see Table 5.2.2.2). Task 1 took the longest, not due to navigation but lack of application familiarity.

Participant ID	Task	Task Ratings	Very difficult	Difficult	Neutral	Somewhat Easy	Easy
001	1	Ease of finding information					**X**
001	1	Keep track of location in application					**X**
001	1	Accurate predict application information					**X**
001	2	Ease of finding information					**X**
001	2	Keep track of location in application					**X**
001	2	Accurate predict application information					**X**
001	3	Ease of finding information					**X**
001	3	Keep track of location in application					**X**
001	3	Accurate predict application information					**X**
001	4	Ease of finding information					**X**
001	4	Keep track of location in application					**X**
001	4	Accurate predict application information					**X**
001	5	Ease of finding information					**X**
001	5	Keep track of location in application					**X**
001	5	Accurate predict application information					**X**
001	6	Ease of finding information					**X**
001	6	Keep track of location in application					**X**
001	6	Accurate predict application information					**X**
001	7	Ease of finding information					**X**
001	7	Keep track of location in application					**X**
001	7	Accurate predict application information					**X**
002	1	Ease of finding information			**X**		
002	1	Keep track of location in application				**X**	
002	1	Accurate predict application information				**X**	
002	2	Ease of finding information					**X**
002	2	Keep track of location in application					**X**
002	2	Accurate predict application information					**X**
002	3	Ease of finding information					**X**
002	3	Keep track of location in application					**X**
002	3	Accurate predict application information					**X**
002	4	Ease of finding information					**X**
002	4	Keep track of location in application					**X**

002	4	Accurate predict application information					**X**
002	5	Ease of finding information					**X**
002	5	Keep track of location in application					**X**
002	5	Accurate predict application information					**X**
002	6	Ease of finding information					**X**
002	6	Keep track of location in application					**X**
002	6	Accurate predict application information					**X**
002	7	Ease of finding information					**X**
002	7	Keep track of location in application					**X**
002	7	Accurate predict application information					**X**
003	1	Ease of finding information			**X**		
003	1	Keep track of location in application				**X**	
003	1	Accurate predict application information					**X**
003	2	Ease of finding information				**X**	
003	2	Keep track of location in application				**X**	
003	2	Accurate predict application information					**X**
003	3	Ease of finding information					**X**
003	3	Keep track of location in application					**X**
003	3	Accurate predict application information					**X**
003	4	Ease of finding information					**X**
003	4	Keep track of location in application					**X**
003	4	Accurate predict application information					**X**
003	5	Ease of finding information					**X**
003	5	Keep track of location in application					**X**
003	5	Accurate predict application information					**X**
003	6	Ease of finding information					**X**
003	6	Keep track of location in application					**X**
003	6	Accurate predict application information					**X**
003	7	Ease of finding information					**X**
003	7	Keep track of location in application					**X**
003	7	Accurate predict application information					**X**

Table 5.2.2.25: Ratings allocated by participants to each task upon usability test completion for sprint 4 (Modified from: Usability.gov, 2022). Scores are given on a Likert scale, ranging from very difficult to easy, and each task is evaluated against the criteria of the ease of finding information, keeping track of location in the system, and accurately predicting application information.

Task	Ease of finding information	Keep track of location in application	Accurate predict application information	Overall Satisfaction (per task)
1	4.3 (87%)	4.6 (93%)	5 (100%)	4.6
2	4 (80%)	4.3 (86%)	4.6 (93%)	4.6
3	5 (100%)	5 (100%)	5 (100%)	5
4	5 (100%)	5 (100%)	5 (100%)	5
5	5 (100%)	5 (100%)	5 (100%)	5
6	5 (100%)	5 (100%)	5 (100%)	5
7	5 (100%)	5 (100%)	5 (100%)	5
Overall Satisfaction (Per option)	4.8	4.8	4.9	

Table 5.2.2.26: Post-test mean task ratings and percentage agreement based on responses recorded in Table 5.2.2.25 (Modified from: Usability.gov, 2022). Percentage agreements are a combination of agree and strongly agree. Overall satisfaction for tasks 1 and 2 were lowest, which are related to using the administrative area, with accurately predicting application information scoring 4.9, slightly greater than the other tasks.

Issues Discovered During Usability Testing

Issue #	Issue Type	Issue Description	Location	Round Discovered	Recommendations	Severity Ranking	Ease of Fix Ranking	Solved
1	Filtering	When using interactive guide on produce home page the produce page is affected by filtering necessary for guide functionality.	ProduceHome component	Sprint 2	Add a 'guide_produce' array to redux store so the produce list for the ProduceHome component remains separate from the guide.	2	1	**Yes**
2	Account clarity	Difficult to figure out that the 'Account' link in the navigation bar takes user to all their profile information	Navbar component	Sprint 2	Change the title of this link to the username of the user logged in as this is commonly done on web applications to indicate this sort of functionality	0	0	**Yes**

Table 5.2.2.27: Issues recorded during usability testing, found only in sprint 2 and had low severity scores. The severity ranking and ease of fix ranking were calculated based on the methods in Figure 5.2.2.1 and Figure 5.2.2.1 (Modified from: Usability.gov, 2021).

Comments Regarding Future Development Received During User Testing

Comment #	Comment	Participant ID	Description	Related Round
1	Welcome email upon sign up	**001**	Typically, when a new account is made for a web application, a welcome email is sent to the email address submitted in the registration process. Emails are currently only sent when a password reset request has been made.	Initial Prototype
2	Links to produce pages from the articles	**001**	In some articles, different produce items are discussed and links to the single item pages could be beneficial to navigation as it would introduce shortcuts.	Initial Prototype
3	Table heading in the planner containing month names disappears on scroll	**003**	When the user is presented with a large list of results in the planner, they may need to scroll down the page to find what they are looking for. The table header remains at the top of the results list so the information in each table row becomes less clear once the header disappears from view. Keeping the header in a fixed position could rectify this.	Initial Prototype, Sprint 4
4	Current month could be highlighted in the planner	**002**	Highlighting the table cell in each row containing the plant care action for the current month could improve readability, as users could find relevant information easier.	Initial Prototype, Sprint 4
5	Pictures of plants in real-world could be added	**002**	Current icons are highly effective at representing produce items. Within each produce page it could be beneficial to show the actual physical appearance of these items, so users know what to look for when caring for plants.	Sprint 1
6	Ability to view recipes uploaded by specific author on an author page	**002**	This can be done using the input bar but would be beneficial for users if they are interested in a specific author to view a page containing all information without having to search in the current way.	Sprint 3
7	Ability to sort recipes by upload date	**003**	Sorting recipes by upload date would enable users to view recent activity on the application and quickly access information about current culinary trends.	Sprint 3

Table 5.2.2.28: User feedback from usability testing discussing areas of improvement. This is separate to the information in Table 5.2.2.27 as these are not problematic and so not impede user interaction with the application.

5.3 Additional Evidence

5.3.1 Non-Functional Requirements

Test ID	Related Requirement ID	Requirement	MoSCow Rating	Pass / Accept / Fail	Comments
NFT01	NR01	Accessibility based on personal ability	MUST	**Accept**	Image descriptions have been added to images for screen-readers, created input components allow movement using the tab and enter buttons, and some icons have descriptions regarding the action they possess when hovered over. More should be done in relation to this requirement.
NFT02	NR02	Accessibility based on device	MUST	**Pass**	SCSS media queries have been implemented to accommodate mobile and desktop use. It also works across different browsers.
NFT03	NR03	Security	MUST	**Pass**	JWT has been used as the authentication mechanism for API endpoints, protected routes have been applied in the front-end, and passwords are encrypted.
NFT04	NR04	Privacy	MUST	**Pass**	Users are only asked to supply necessary data when creating an account and the password value is never returned from any API call.
NFT05	NR05	Response time	MUST	**Pass**	Application has a fast load time and performs API calls quickly (see Table 4.2.2.2).
NFT06	NR06	User friendly	MUST	**Pass**	Natural language has been used across all aspects of the system and an easy-to-use interface has been developed. Modals are present to prevent accidental deletions.
NFT07	NR07	Consistency	SHOULD	**Pass**	The application possesses a consistent appearance, largely due to reusable React components employed.
NFT08	NR08	Application aesthetic	SHOULD	**Pass**	Minimalistic design has been implemented, alongside use of a colour palette. Bespoke images have been used throughout the application (see sections 5.3.4 and 5.3.5).
NFT09	NR09	Notification of success	MUST	**Pass**	Successes are indicated to the user in the front-end in the form of view updates and messages supplied on pages or in modals. Success HTTP statuses are returned in API calls.
NFT10	NR10	Notification of failure	MUST	**Pass**	Users are notified of failures through error messages in the front-end, typically shown in red text, and in API requests, which contain information regarding the reason for issues.

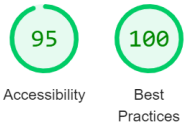
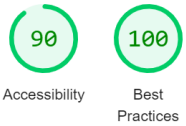
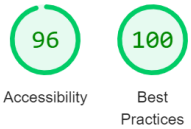
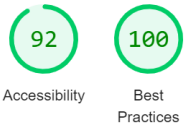
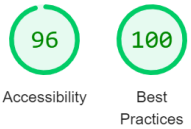
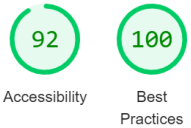
NFT11	NR11	Availability	COULD	**Fail**	The application has not been deployed. The chosen version control system will be able to help facilitate this in the future.
NFT12	NR12	Scalability	SHOULD	**Accept**	MongoDB is a scalable database management system and using React facilitates the application scalability. The system has not been deployed so cannot be fully evidenced.
NFT13	NR13	General information	COULD	**Fail**	General application information has not been included for the end user. API documentation has been established (see Figure 4.3.1.7) but this is for developer use.

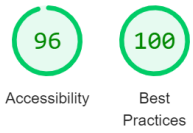
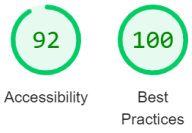
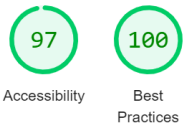
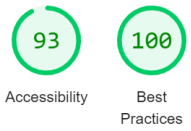
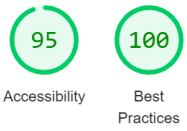
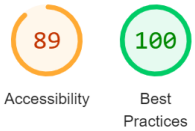
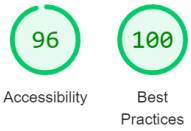
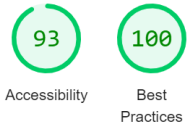
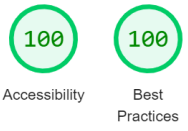
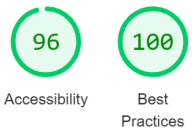
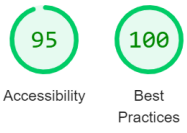
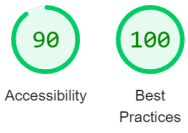
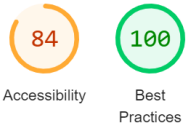
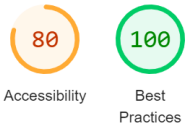
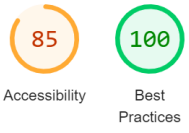
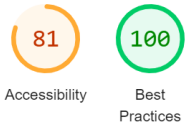
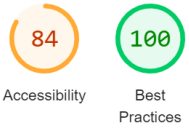
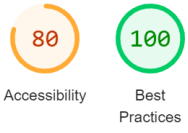
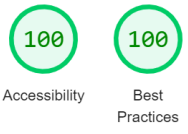
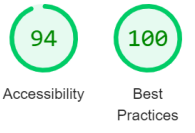
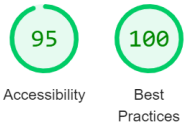
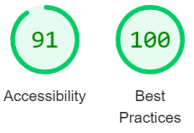
Table 5.3.1.1: Brief discussion of how well the end product meets non-functional requirements defined (see section 1.1.2). NFT11 and NFT13 fail as these areas have not been implemented and were prioritised lower than other requirements. NFT01 is accepted as some progress has been made in this area, with more advancements being included in consideration for future development, as well as NFT as initial work has been done for this requirement.

5.3.2 Lighthouse Performance Testing

Lighthouse is a testing means created by Google which evaluates performance, accessibility, and best practices (Google Developers, 2021). It helps to identify technical issues which can contribute to creating a positive user experience. The developer was unable to test for the performance statistic due to browser restrictions, but detailed descriptions provided in the test results alluded to how performance is slowed due to the wide range of images included in the application and their size. Reducing the file sizes for these could be implemented in the future to combat this issue.

The two key values tested for were accessibility and best practices, and were assessed against desktop and mobile views of the application (see Table 5.3.2.1). Both values consistently scored very high across all routes, with the professional approach of the developer indicative of the best practice scores equalling 100 for all paths. The accessibility results are marginally lower for routes possessing a large number of input fields, which have been tailored for this application and may be the cause of the lower scores.

Test ID	Route	Desktop Results	Mobile Results
LT01	/		
LT02	/login		
LT03	/sign-up		

LT04	/forgot-password/:id		
LT05	/planner		
LT06	/produce/:produceName		
LT07	/produce		
LT08	/admin		
LT09	/recipes/:id		
LT10	/recipes		
LT11	/recipes/recipe-form/:id		
LT12	/recipes/recipe-form		
LT13	/articles		
LT14	/articles/dish-the-dirt		

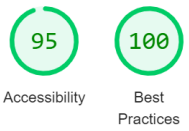
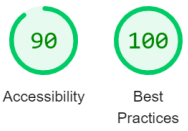
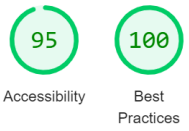
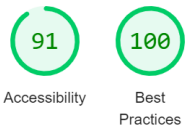
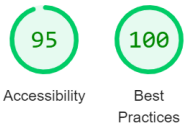
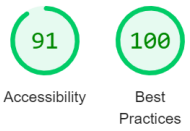
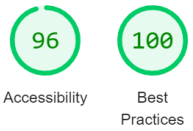
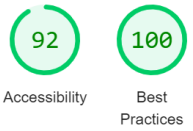
LT15	/articles/plant-problems		
LT16	/articles/seasonal-sustenance		
LT17	/account		
LT18	/account/update		

Table 5.3.2.1: Results for Lighthouse testing on all React routes for the application, evaluating the accessibility and accomplishment of best practices.

5.3.3 Nielsen's Ten Usability Heuristics

After the final round of usability testing, users were asked for their opinions on the generated application in relation to Nielsen's heuristics (1994b), which are discussed in section 2.2.2.

User responses were provided on a Likert scale, ranging from strongly disagree to strongly agree (see Table 5.3.3.1). An overview of these results is provided in Table 5.3.3.2, with comments deriving from user responses when providing ratings for each heuristic.

User ID	Heuristic	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
001	Visibility of system status				**X**	
001	Match between system and real world					**X**
001	User control and freedom					**X**
001	Consistency and standards					**X**
001	Error prevention					**X**
001	Recognition rather than recall					**X**
001	Flexibility and efficiency of use					**X**
001	Aesthetic and minimalist design					**X**

001	Error recognition, diagnosis, and recovery					**X**
001	Help and documentation			**X**		
002	Visibility of system status					**X**
002	Match between system and real world					**X**
002	User control and freedom					**X**
002	Consistency and standards					**X**
002	Error prevention					**X**
002	Recognition rather than recall					**X**
002	Flexibility and efficiency of use					**X**
002	Aesthetic and minimalist design					**X**
002	Error recognition, diagnosis, and recovery					**X**
002	Help and documentation				**X**	
003	Visibility of system status					**X**
003	Match between system and real world					**X**
003	User control and freedom					**X**
003	Consistency and standards					**X**
003	Error prevention					**X**
003	Recognition rather than recall				**X**	
003	Flexibility and efficiency of use					**X**
003	Aesthetic and minimalist design					**X**
003	Error recognition, diagnosis, and recovery					**X**
003	Help and documentation			**X**		

Table 5.3.3.1: User responses when asked to provide scores for how well they believe the developed system meeting Nielsen's usability heuristics. Responses were given on a Likert scale and user IDs match the participants listed in Table 5.2.2.3.

Heuristic	Related Figure	Majority Response	Comments
Visibility of system status	Figure 5.3.3.1	**Strongly Agree**	Feedback on actions is provided through messages, modals, and page redirects upon success. Page titles aid in assisting user understanding of where they are in the system.
Match between system and real world	Figure 5.3.3.2	**Strongly Agree**	Images used are reflective of real-world actions and items. Natural language is used, and information flow follows natural practice, for instance the ingredients appearing before instructions in the single recipe page.
User control and freedom	Figure 5.3.3.3	**Strongly Agree**	Users are able to undo any accidental actions, like adding and removing items to and from lists. Modals are also present before deletion and logout actions to ask for user confirmation and prevent accidental tasks.
Consistency and standards	Figure 5.3.3.4	**Strongly Agree**	Page layout is consistent throughout the application. Elements such as the logo in the navigation bar acting as a link to the homepage, star icons for adding to wish-lists and the account navigation links appearing in the top right of the screen, facilitates conformity to common web application practices.
Error prevention	Figure 5.3.3.5	**Strongly Agree**	Modals appear to the user before deletion and logout actions to ask for task confirmation, valued by the users. Front-end validation assists in preventing erroneous data from being sent to API endpoints.
Recognition rather than recall	Figure 5.3.3.6	**Strongly Agree**	An assortment of icons are used throughout to reflect information, such as varying levels of sunlight, and to signify actions, like adding an item to the currently growing list. Users are able to determine values and potential actions without having to read descriptions.
Flexibility and efficiency of use	Figure 5.3.3.7	**Strongly Agree**	Navigational and task performance shortcuts exist and were used across the application. The single produce pages can be accessed not only from the produce home area but also from the interactive guide, planner, and account area. Recipes can be deleted, and items removed from lists from the account area as well as their respective single item pages.
Aesthetic and minimalist design	Figure 5.3.3.8	**Strongly Agree**	Users feel the application makes use of a consistent and attractive colour palette with a constant layout across all pages and components (see sections 5.3.4 and 5.3.5).
Error recognition, diagnosis, and recovery	Figure 5.3.3.9	**Strongly Agree**	Error messages appear in red text in appropriate, visible places when they occur, for example underneath an input field which is required. Messages are clear and concise with information on how to rectify the encountered problem.
Help and documentation	Figure 5.3.3.10	**Neutral**	Users believe that the application is simplistic enough that user directed documentation is not provided. Tooltips and response messages shown assist in guiding application use.

Table 5.3.3.2: Brief outline of how well the end product meets Nielsen's Ten Usability Heuristics for Interface Design (1994b) based on user feedback. Comments are based on user feedback during the process of obtaining these results.

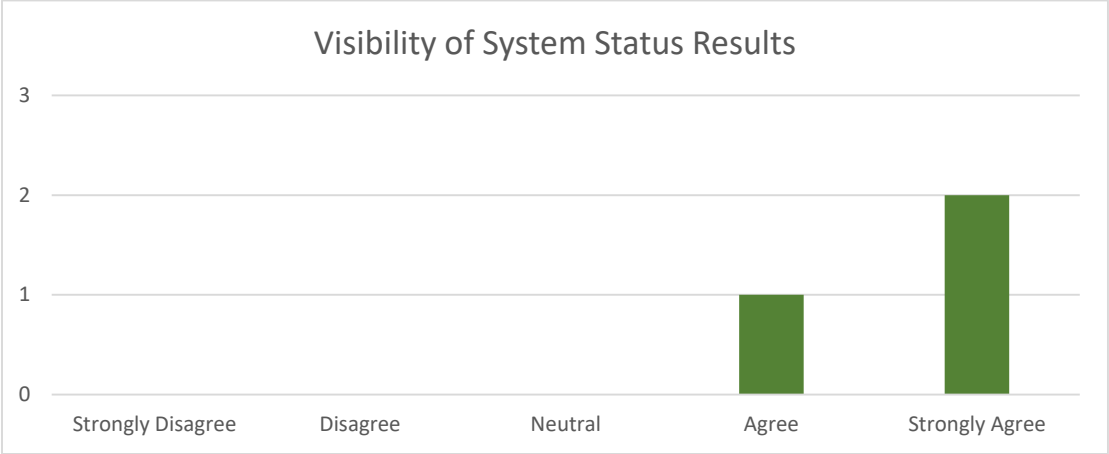


Figure 5.3.3.1: Chart presenting the results gained from user feedback regarding the visibility of system status heuristic.

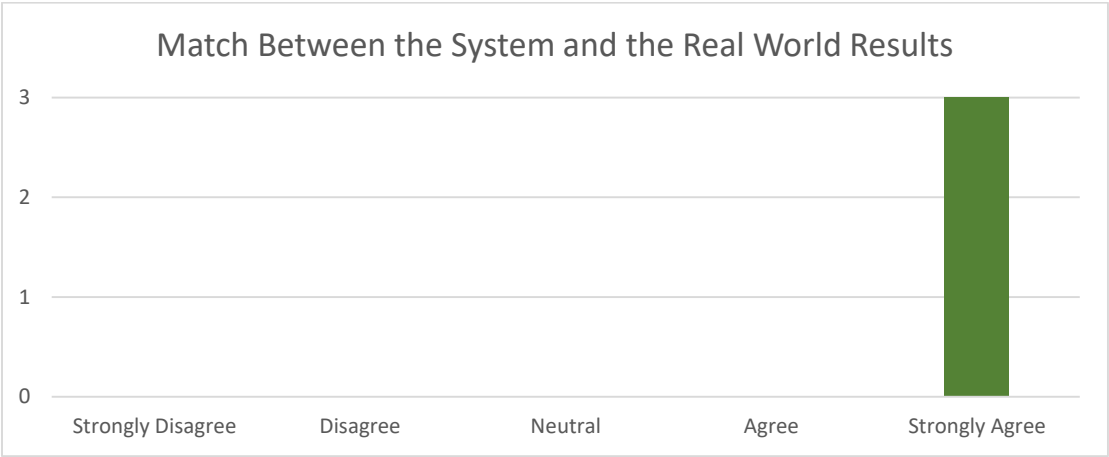


Figure 5.3.3.2: Chart presenting the results gained from user feedback regarding the match between the system and real-world heuristic.

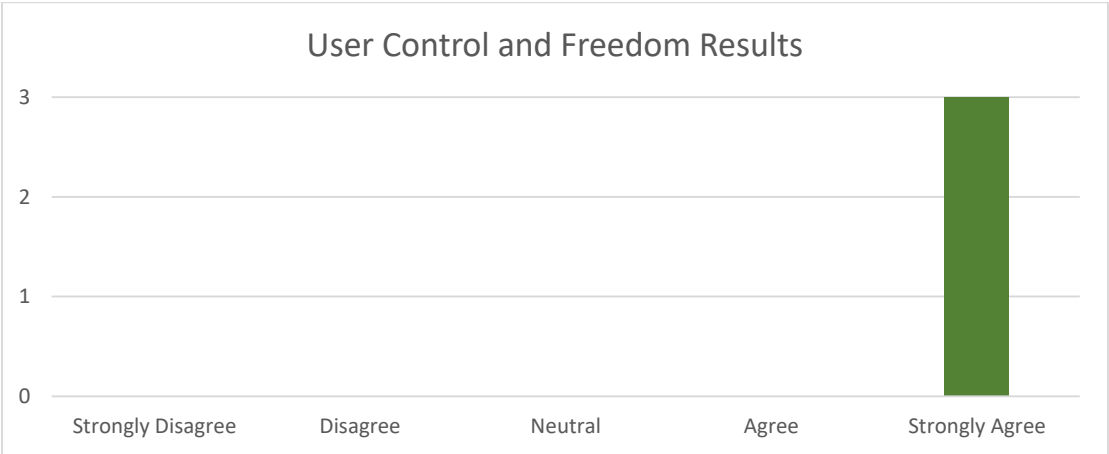


Figure 5.3.3.3: Chart presenting the results gained from user feedback regarding the user control and freedom heuristic.

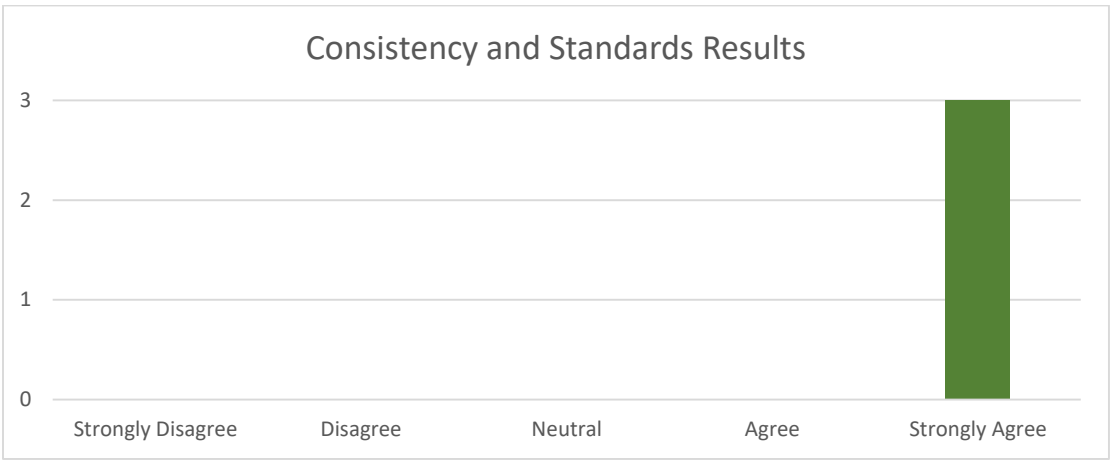


Figure 5.3.3.4: Chart presenting the results gained from user feedback regarding the consistency and standards heuristic.

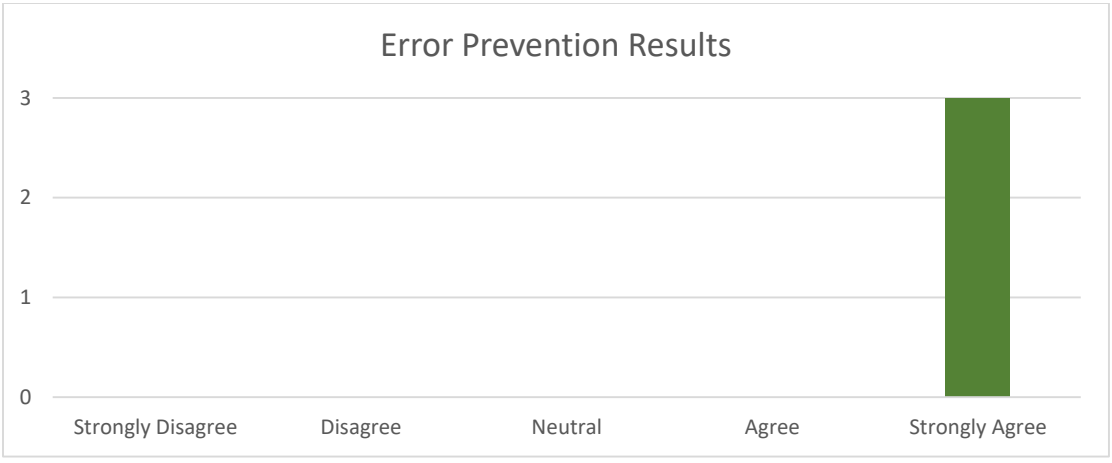


Figure 5.3.3.5: Chart presenting the results gained from user feedback regarding the error prevention heuristic.

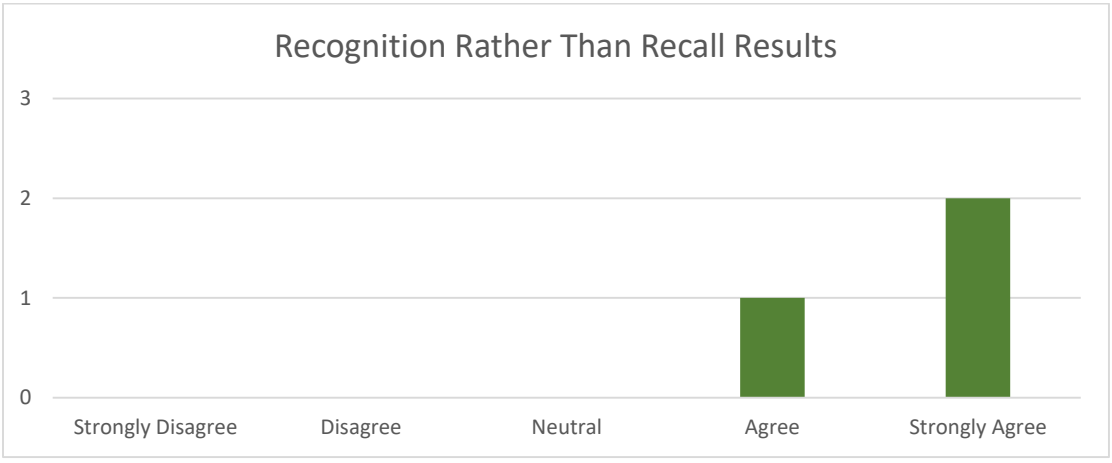


Figure 5.3.3.6: Chart presenting the results gained from user feedback regarding the recognition rather than recall heuristic.

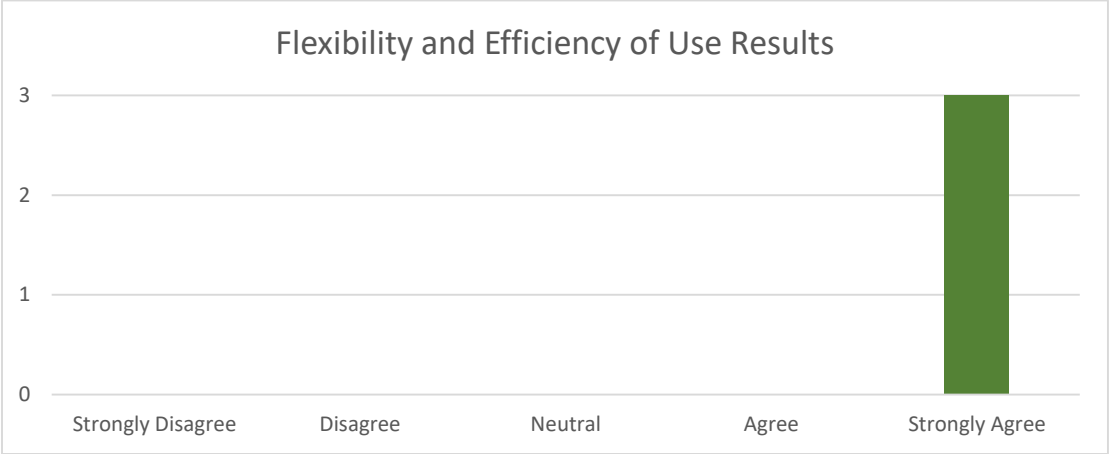


Figure 5.3.3.7: Chart presenting the results gained from user feedback regarding the flexibility and efficiency of use heuristic.

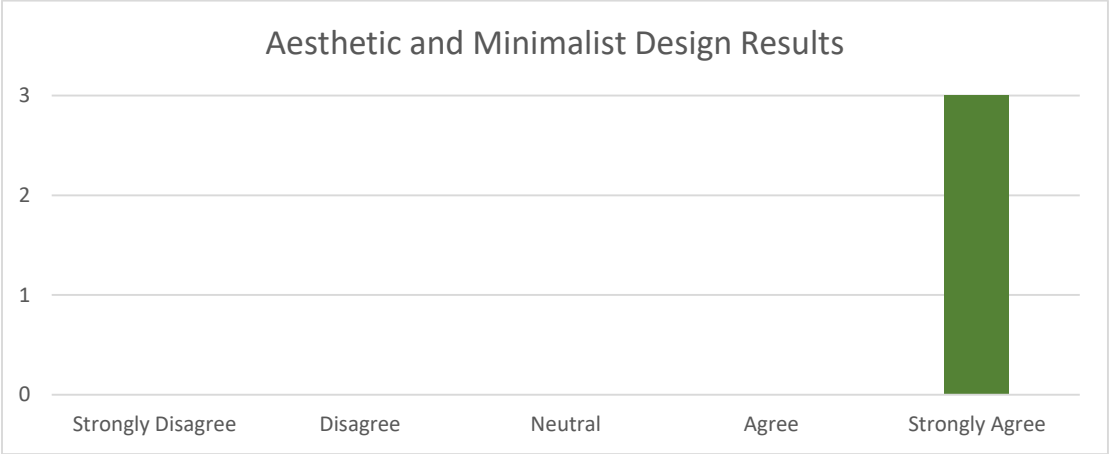


Figure 5.3.3.8: Chart presenting the results gained from user feedback regarding the aesthetic and minimalist design heuristic.

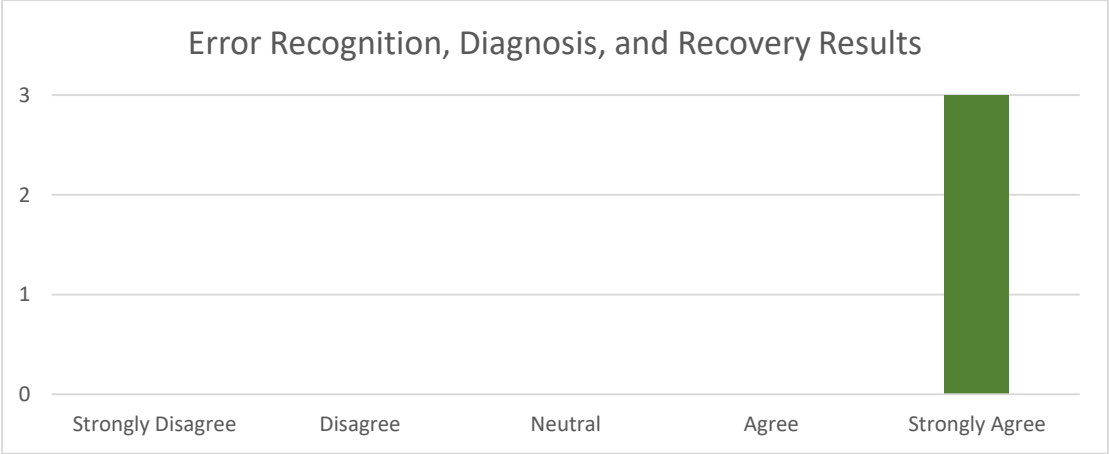


Figure 5.3.3.9: Chart presenting the results gained from user feedback regarding the error recognition, diagnosis, and recovery heuristic.

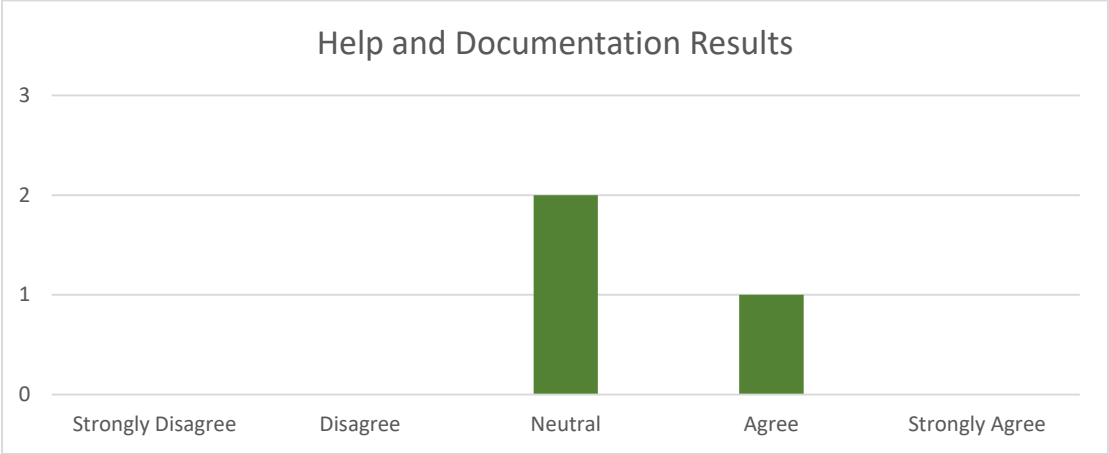


Figure 5.3.3.10: Chart presenting the results gained from user feedback regarding the help and documentation heuristic.

5.3.4 Gestalt Principles

The Gestalt principles of similarity, continuation, proximity, figure ground, symmetry, and common fate (Chapman, 2022; Wertheimer, 2012) have been intentionally applied across all areas of the project. They contribute to the overall user experience of the application as aesthetically pleasing design created using these techniques leads to positive interactions.

The concept of similarity is reflective of items that are not identical, but can be recognised as associated due to their comparable appearance. Components enabling result filtering to enforce this, with the buttons at the top of the produce and planner pages possessing different colours (see Figure 5.3.4.1) and the various input fields on the recipes page having a similar form (see Figure 5.3.4.2).

Figure 5.3.4.1: Screenshot of the buttons used for filtering produce items in the planner by type, or to show those contained in the currently growing list.

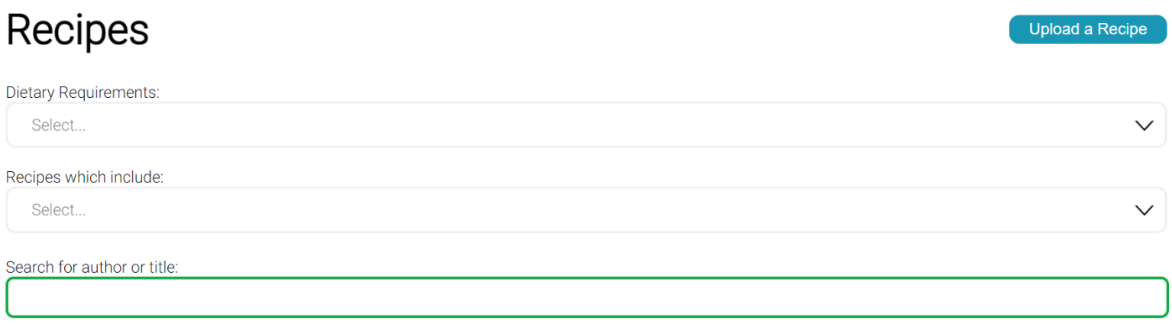


Figure 5.3.4.2: Screenshot of the top of the recipes page showing the input fields used to filter results.

Tags additionally showcase this principle, with dietary requirement tags shown for recipes having different background colours (see Figure 5.3.4.3) and tags outlining different produce varieties (see Figure 5.4.3.4) possessing different colours when hovered over (see Figure 5.3.4.5). Card components also follow this principle as the information contained within each one is different, but their appearance is the same (see Figures 5.3.4.7, 5.3.4.8 and 5.3.4.9).

Figure 5.3.4.3: Screenshot of the dietary requirement tags shown in the single recipe page for 'Potato and Roasted Garlic Broth'.

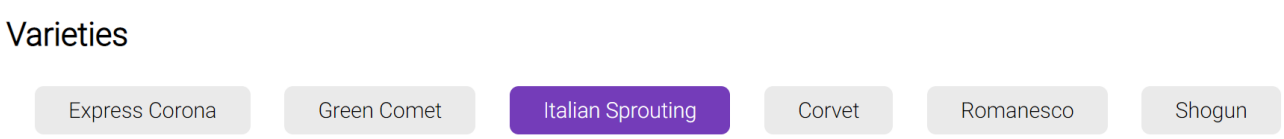


Figure 5.3.4.4: Screenshot of the variety tags shown in the single produce page for 'Calabrese'. The background of each tag changes to a random colour when hovered over, in this instance 'Italian Sprouting' is being hovered on.

As for continuation, items not clearly contained in a section by a border can still be grouped together based on this principle. This can be evidenced in all the filter buttons and the dietary requirement tags outlined above (see Figures 5.3.4.1, 5.3.4.2 and 5.3.4.3). Content within tables is also a strong example of this as single entries of information can still be identified given that the items are within a row (see Figure 5.3.4.5). Items can also be associated to each other based on proximity as keeping elements close together ensure they are identified as related, whether they are grouped by a title or not, such as elements of single item pages (see Figure 5.3.4.6) and filter components.

Figure 5.3.4.5: Screenshot of the uploaded recipes section within the account page for the 'admin' user.

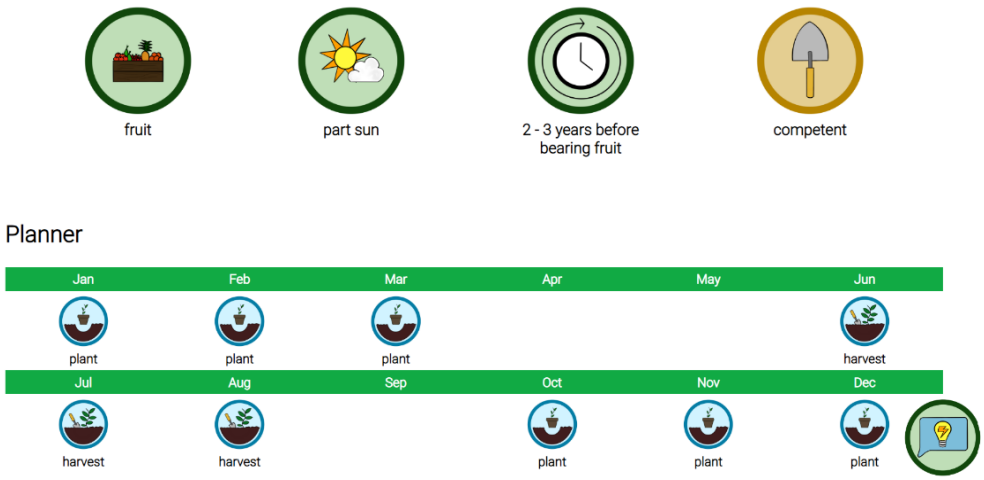


Figure 5.3.4.6: Screenshot of part of the single produce page for 'Acid Cherry' general information is shown at the top with the planner beneath.

The figure ground principle is executed through modals, which appear as a white body over a translucent black background, and also in the event that a card is hovered on as a blurred border appears, giving the illusion that it is hovering (see Figure 5.3.4.7). Similar to modals, when in mobile view the navigation bar for the application is opened through clicking the logo and the navigation links appear as an overlay (see Figure 5.3.4.8) to the current page.

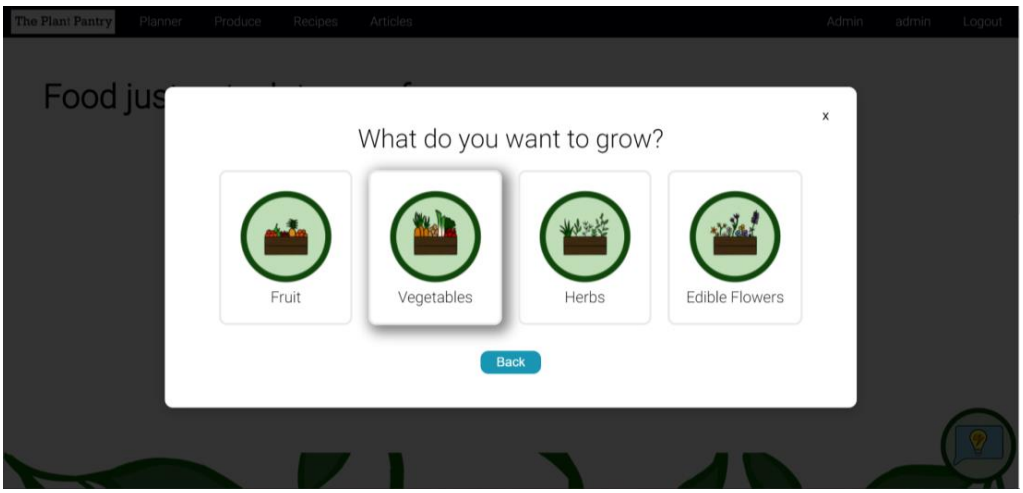


Figure 5.3.4.7: Screenshot of the modal section where the user chooses the produce type they are interested in. They are currently hovering on the 'Vegetables' card.

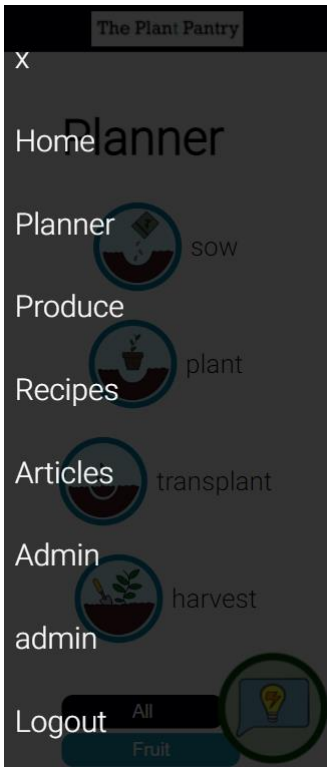


Figure 5.3.4.7: The navigation pane when the application is in mobile view and contains the same links as the horizontal bar when in desktop view. It is closed in this case using the 'x' icon.

All pages across the application have a somewhat symmetrical layout, while some pages are more clearly balanced (see Figures 5.3.4.7 and 5.3.4.8), others use a range of features to create an evenly weighted appearance (see Figure 5.3.4.9).

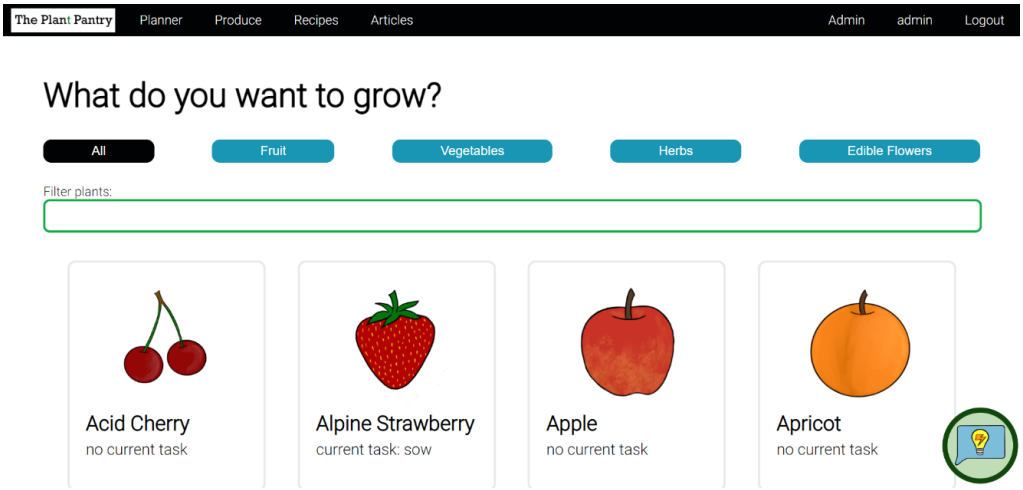


Figure 5.3.4.8: Screenshot of the produce page when all results are shown, with the cards organised in alphabetical order.

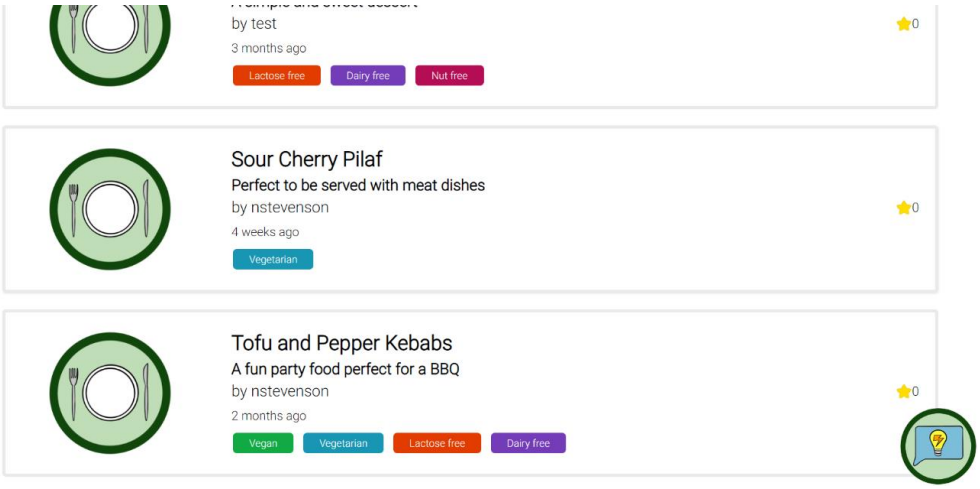


Figure 5.3.4.9: Screenshot of results shown on the recipes page, each card is split into three areas; the image, the information, and the likes count.

The principle of common fate is exhibited through movement, with groups of elements connected through the ability to scroll. Option lists for dropdown fields contain many entries (see Figure 5.3.4.10) and are restricted to a certain height so the scroll bar indicates there are more choices that those initially shown. Modal results are also shown in a contained area, therefore also uses a scrollbar (see Figure 5.3.4.11).

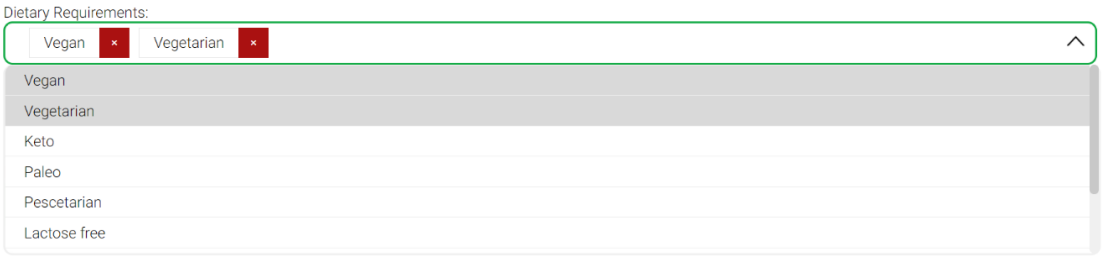


Figure 5.3.4.10: Screenshot of results shown on the recipes page, each card is split into three areas; the image, the information, and the likes count.

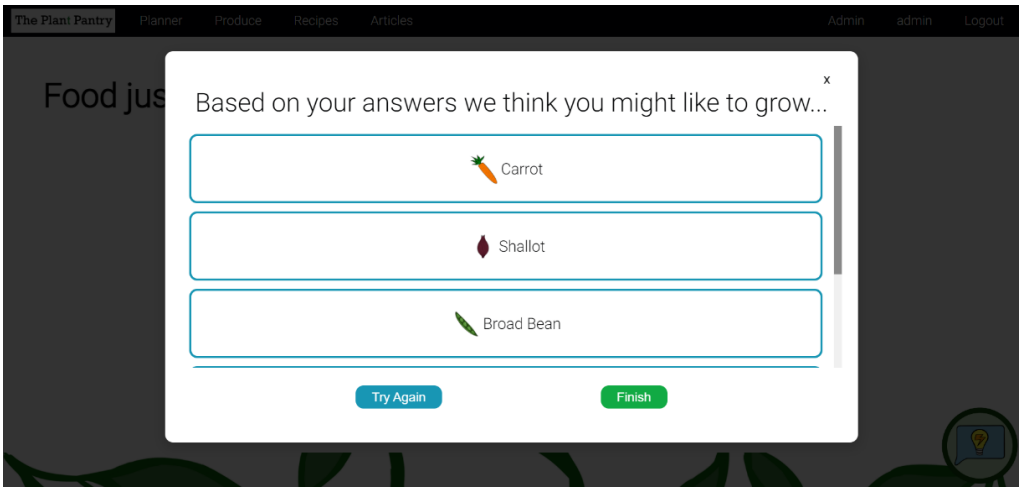


Figure 5.3.4.11: Screenshot of results shown from the interactive guide when a user searches for vegetables which can be grown indoors in a location with partial sun, and indicate they do not have much experience.

5.3.5 Use of Colour

An effective colour palette has the ability to transform web applications, attracting users and creating a positive experience. An assortment of colours have been used across all facets of this application. In Figure 5.3.5.1, the top three colours defined are central to all pages and are largely reflective of the primary colours. Within the scientific arena, the primary colours are red, blue, and green, whereas within art they are red, blue, and yellow. In this system green is used as opposed to yellow, to reflect the natural world, the fundamental theme of the application, and due to the poor contrast of yellow on a white background. Blue instils a sense of professionalism and organisation, used largely for buttons for action execution, and red is representative of danger, therefore is used for error messages and deletion tasks (Khazanova, 2022). Conversely to this, green is reflective of success.

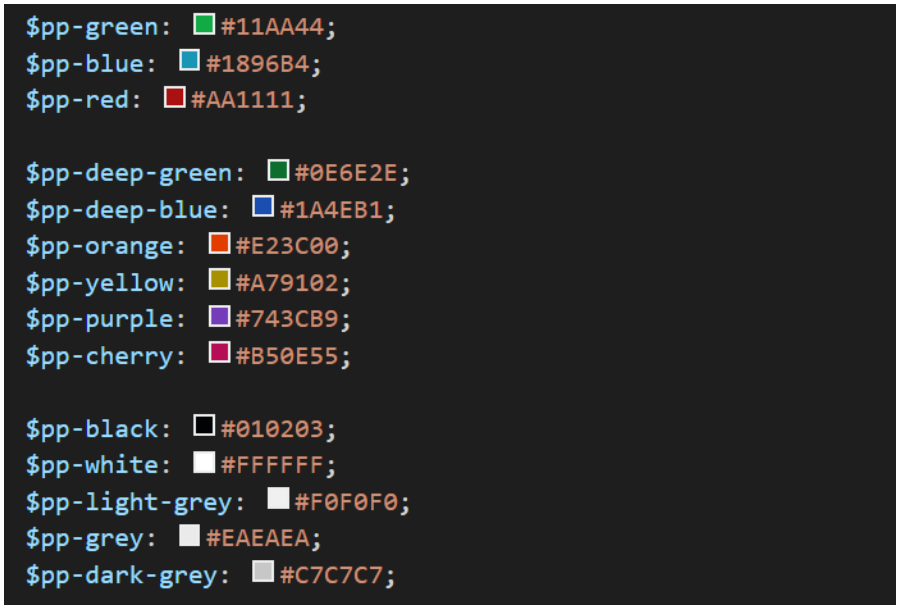


Figure 5.3.5.1: Screenshot of _variables.scss file contained within the client directory for the system. It contains colour values used throughout the stylesheets imported into the application front-end. They are grouped into three sections: the main colours, supplementary colours, and grayscale variables.

Supplementary colours ranging from a dark green to cherry (see Figure 5.3.5.1) are used for smaller, less significant elements, such as tags (see Figures 5.4.4.3 and 5.4.4.4) to provide additional vibrancy and colour associations, for instance the 'Lactose free' diet is classified by the colour orange. The colour palette implemented intentionally makes use of bright, warm colours that are complimentary (Chapman, 2022; Khazanova, 2022), with shades of grey utilised to reflect hovering and option selection (see Figure 5.3.4.10).

Most importantly, the colour palette has been determined with accessibility in mind. Sufficient contrast levels are vital for the perception of information, one of the WCAG 2.1 principles (W3C, 2018), to ensure that end users with varying levels of vision can use this application. Contract levels were checked using Accessible Web's WCAG colour contrast checker (2022) and these tests were sectioned into three areas; checking the main colours against black and white backgrounds, checking supplementary colours against black and white backgrounds, and checking that white text can be viewed on backgrounds using the supplementary colours.

Contrast between black and white was also examined (see Figure 5.3.5.2) within the checks for the main colours on black and white backgrounds. The only combination which fails is the colour grey on a white background, which is used as borders for cards (see Figures 5.3.4.8 and 5.3.4.9) and the input dropdown component (see Figure 5.3.4.10).

The Gestalt principle of proximity (Chapman, 2022; Wertheimer, 2012) aids in the visual organisation of cards, however the potential issue of a user not seeing the border for the dropdown field should be rectified in the future. All tests for checking the contrast ratios of supplementary colours pass, demonstrating visual accessibility (see Figure 5.3.5.3).

		Background colours		
Hex. Colour Code	**Application Variable Name**	**Black (#000000 / $pp-black)**	**White (#FFFFFF / $pp-white)**	**Light Grey (#EAEAEA / $pp-grey)**
#11AA44	$pp-green	Pass	Pass	-
#1896B4	$pp-blue	Pass	Pass	-
#AA1111	$pp-red	Pass	Pass	-
#000000	$pp-black	-	Pass	Pass
#EAEAEA	$pp-grey	Pass	**Fail**	-

Figure 5.3.5.2: Results obtained from compliance checks executed on Accessible Web's WCAG colour contrast checker (2022) for the main colours on black, white, and grey backgrounds. The pass/fail result was taken from the UI component score.

		Background colours		Colour as background
Hex. Colour Code	**Application Variable Name**	**Black (#000000 / $pp-black)**	**White (#FFFFFF / $pp-white)**	**White (#FFFFFF / $pp-white) text on background**
#0E6E2E	$pp-deep-green	Pass	Pass	Pass
#1A4EB1	$pp-deep-blue	Pass	Pass	Pass
#E23C00	$pp-orange	Pass	Pass	Pass
#A79102	$pp-yellow	Pass	Pass	Pass
#743CB9	$pp-purple	Pass	Pass	Pass
#B50E55	$pp-cherry	Pass	Pass	Pass

Figure 5.3.5.3: Results obtained from compliance checks executed on Accessible Web's WCAG colour contrast checker (2022) for the secondary colours on black, white, and grey backgrounds. Using white text on backgrounds of these colours has also been tested. The pass/fail result was taken from the UI component score.

Overall, all colour contrast tests pass, aside from one, highlighting that the use of colour has been largely effective in terms of accessibility. User feedback subsequent to usability testing regarding Nielsen's aesthetic and minimalistic design heuristic (see Table 5.3.3.2) provided glowing reviews of the colour palette employed.

5.4 Consideration for Future Work

Development of this project does not have to finish at this stage, and there are many ways for this to continue. This potential falls into three categories of origin; the developer, the requirements list generated, and the testing results.

Developer belief is that pagination could be interested across some pages and components to aid in site readability. This opinion was not offered during validation testing (see Tables 5.2.2.27, 5.2.2.28, 5.3.1.1 and 5.3.3.2), probably due to the small sample size of information that had been seeded into the application at this stage in some areas. This feature would be especially useful in the recipes section, as it is not uncommon for long lists consisting of short entries to appear unpaginated, as is the case on the planner and produce pages. The recipes page contains cards possessing an abundance of important information (see Figure 5.3.4.9) and without pagination would run the risk of a cluttered appearance. The articles page could additionally benefit from this in the event more articles are added to the application.

Deploying the application would assist in successfully meeting the availability and scalability non-functional requirements, listed as NF11 and NF12 in Table 5.3.1.1. As discussed earlier in the report, Azure DevOps Repos, the version control system enlisted, facilitates CI/CD pipelines which can be used for deployment and can be used to accommodate both front and back-end areas of the application.

Eighty-one functional requirements were identified during the initial requirements gathering process (see section 1.1.1) and in some cases their priorities were updated to reflect what could reasonably be achieved by the end of the fourth sprint executed. Priorities, most visibly shown in the MoSCoW ratings allocated, indicate what progress could be made during development up to this point and all requirements are deemed worthy of inclusion at any stage, meaning that they should be heavily considered in future development as they still matter to end users.

Results obtained from the testing processes employed are arguably the most important considerations, as they embody end user opinion, and are necessary to address for maintaining the strong end user focus. User opinion is clearly outlined in Table 5.2.2.28, which contains feedback obtained during usability testing, and also in Tables 5.3.1.1 and 5.3.3.2. Acting upon this feedback would require varying levels of effort, for example highlighting the current month in the planner would be more straightforward than accumulating and including real-world pictures of produce items in pages. A common thread across these results is the lack of application documentation supplied to end users, a non-functional requirement listed by the developer and in Nielsen's heuristics (1994b). While users state that the application is largely self-explanatory and help is given in the form of system feedback, tooltips and action descriptions on hover, the requirement has not explicitly been met and therefore needs addressed.

Accessibility is necessary to ensure the inclusion of all potential end users. Strides have been made in this area regarding the ease of using mobile devices and the careful consideration of colour (see section 5.3.5), but there is always room for improvement, which are key elements of the application that were prioritised (Krug, 2005). During Lighthouse testing (see Figure 5.3.2.1), result descriptions indicated that performance could be improved with smaller image file sizes, which should be heavily considered in the future. On a more practical level, the ability to increase font size throughout the entirety of the application and dark mode could be added to improve accessibility for end users with different levels of sight. There has been dispute over the actual benefits of dark mode, with popular opinion believing it reduces eyestrain. Recent studies have mixed opinions on this, with results showing stronger correlation between dark mode usage and improved sleep rhythms (Watson, 2021), and that user performance and concentration is improved without it (Clarke, 2019). Regardless of merit given to the health and accessibility benefits of dark mode, it remains a commonly desirable feature of web applications that should be considered for implementation.

6.0 Conclusion and Reflection

6.1 Critical Appraisal of the Project

This application has been developed to a high standard, makes use of a modern technology stack, and meets all requirements prioritised for inclusion in the development up to this stage.

End users are successfully able to perform a wide range of actions across the application. They are able to register for an account which, once logged into, provides access to features including areas showing an annual planner for all plants; a produce section containing information for produce items; a recipes area where users can view different recipes they can make with their homegrown produce; and an articles section containing top tips. Users are able to upload their own recipes to the application, which they can monitor in the account area.

From the single produce pages, users can manage their currently growing list and wish list, while recipes can be added and removed from their liked recipes list from the single recipe page. Functionality in the account section enables users to update their personal information and manage all lists associated to their account. Admin users can also perform all these actions, and have access to an admin area that offers application insights through charts, and maintenance options for all user accounts.

The security and privacy of the application have been heavily considered, with implementations of JWT authorisation tokens, back-end variable encryption, and user password encryption successfully added. Additional privacy measures, such as protected routes in the front-end and only returning necessary information in API responses, have bolstered user trust in the system and facilitated the perception of a professional application.

Version control has been successfully utilised, following best practices for branch management, and has been consistently used from the outset. Implementation of this has enabled confident development and professional practice to be demonstrated.

Results acquired from outcomes of verification and validation testing revealed the perspective of end users and requirements in relation to application appraisal. In terms of verification, favourable outcomes were evidenced in walkthroughs, manual and automated API endpoint testing, code review and unit tests. Regarding validation, all requirement acceptance criteria were met in user acceptance testing (see Appendix A) and usability testing enabled direct interaction with end users, who provided informed feedback. Carrying out these testing practices after the conclusion of each iteration enabled issues uncovered to be fixed, such as fixing how produce data was returned to the interactive guide feature (see Table 5.2.2.27).

Feedback also enabled quick changes to minor elements, such as the text shown in the navigation bar for the link to the account area and the ability to filter the planner and produce areas by more than one type: elements not accommodated in the requirements list or initial prototypes. User responses provided insight into areas of the project which require further consideration in the future, such as accessibility, which are outlined in section 5.4 of this report.

There is always potential for this application to continue development outside of the boundary of this project, given its superior performance within user testing and professional approach.

6.2 Reflection on the Project Plan

The project plan devised was effective in covering all aspects of the development process and was executed to a high standard.

Requirement gathering occurred prior to the initial protype and occurred in the forms of individual interviews, focus groups and a survey (see Figures 1.2.1 and 1.2.2) to establish a core set of functional and non-functional requirements from which the project was generated from. Upon completion of the initial prototype, some additional functional requirements were added (see section 1.1.1). All were executed and were largely allocated a high priority. Prioritisation using MoSCoW and relative weighting aided in streamlining the application and focusing on end user goals for usage.

The selection of technologies implemented proved to be effective and did not require any changes between the completion of the initial prototype and the commencement of following iterations. Creating the front-end and back-end of the system using JavaScript based methods, benefitted the programming performance of the developer as often skills grown in one area of the application could be utilised in the other. All technologies selected for inclusion were implemented across the project to a high standard and enabled the technical skills of the developer to be demonstrated, improved, and expanded.

Knowledge of different software elements has been increased in addition to this, with extensions being utilised in VS Code, illustrative techniques used within Procreate, and automated testing in Postman. This understanding mutually benefitted the developer and the application. Version control was used effectively across the whole development process and in the way it was intended.

There were some changes to the project plan from what was initially planned before the initial prototype was developed. Most namely, the number of sprints to be executed within the timeframe changed. The greater number was rejected in favour of ensuring that there would be enough time to manage user and developer expectation. There were five iterations of new requirement development (see section 1.3) with an additional iteration added to allow for changes to be made based on user feedback from previous sprints. This was visually represented through the Gantt Chart in section 1.3.3.

Testing was also implemented and embodied both verification and validation. The methodologies for these determined prior to the start of development were executed, in addition to some evaluations not specified at the inception of the project. The verification process initially consisted of walkthroughs, code review and unit testing, but was expanded to include API endpoint testing which enabled thorough examination of the effectiveness of the back end of the system. As for validation, user acceptance testing (UAT) and usability testing were carried out, alongside evaluation of compliance to standards like Nielsen's heuristics (1994b), Krug's principles (2005), colour theory, and WCAG 2.1 standards (W3C, 2018).

Combining both verification and validation was a successful way of testing the entire system and how effective the project plan was, as a poorly designed plan would have seriously impinged the conclusions drawn from testing. The results gathered were highly insightful and favourable towards the application generated, evidencing the success of the plan.

6.3 Reflection on the Appropriateness of Initial Time/Effort Estimations

The overall intended project timeline is depicted in Table 1.3.2.1 and embodies checkpoints the developer believed could be met throughout the development of the project, which was largely accurate.

The initial prototype of the application was developed on schedule, which provided insight into how the rest of the development process would pan out. Insight into how to balance development and documentational aspects was ascertained at this stage and had a large influence on how the project plan was modified to feature fewer iterations.

In relation to the execution of the development plan outlined in Table 1.3.2.1, it was entirely successful, with the exception of the final round of usability testing. Aligning schedules to meet with users was easier in early stages of the project, so when the time approached for the final round of testing it was difficult to arrange these meetings and meant that the fixes based on feedback occurred at a later stage than anticipated. Evidence of this is shown in the contribution graph in Figure 3.3.4, as commits to the project repository on the version control system extended past the start of April, the predicted completion of the system.

This was a factor beyond developer control, and the intention behind estimating an earlier completion date was to allow a buffer to be created which accommodated any potential issues that would affect the overall project time frame, such as this. Elements of the project that hinged on the developer, such as documentation generation and all programming aspects were carried out according to plan, except for the last fixes due to the delay of final testing.

Involvement in projects within a personal and professional capacity enabled reasonably accurate estimations to be provided, and the knowledge of the importance of having slack within the project.

6.4 Reflection on the Appropriateness of Software Methodology Used

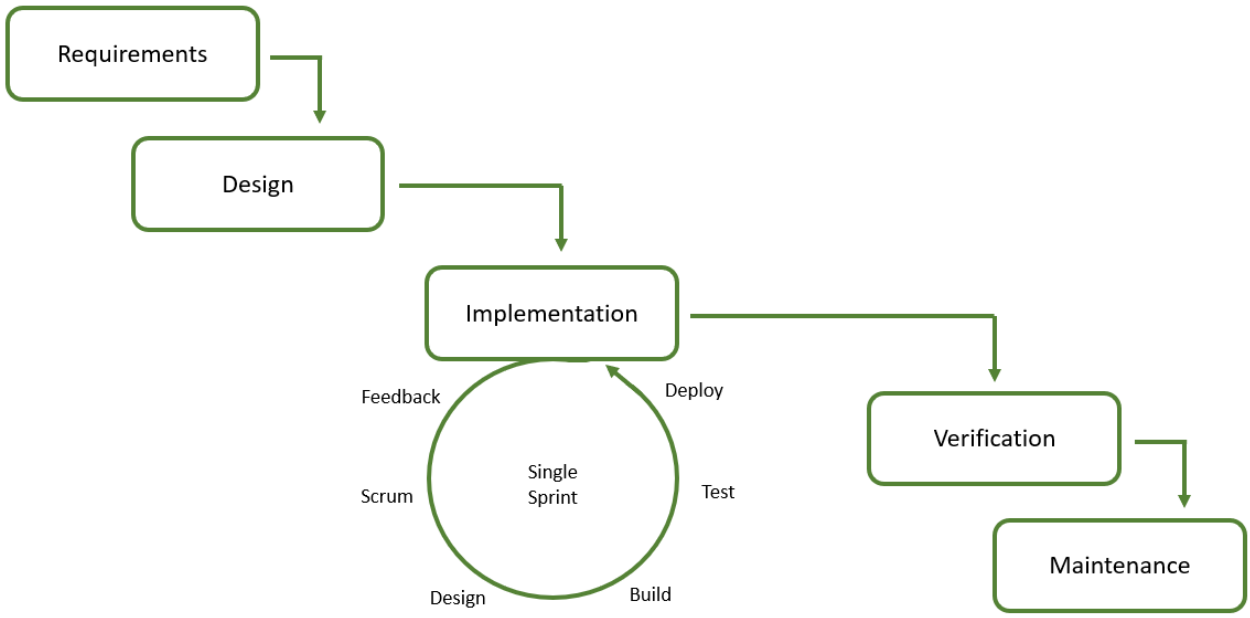


Figure 6.4.1: The Agile-Waterfall Hybrid methodology implemented (Modified from: Tan, 2019).

The Agile-Waterfall Hybrid methodology (see Figure 6.4.1) was selected at the start of the development process as it works well for the size of project developed, can be implemented by the developer in the way that suits the project best, and enables speed of implementation (Intland Software, 2021).

There were no issues observed from the use of this methodology, with the advantages of it being of the utmost benefit. These strengths include how it is not a strict practice, enhances overall collaboration and the end goal is central throughout development (Intland Software, 2021; Lucid Content Team, 2022; Tan, 2019). Supposed drawbacks of the Agile-Waterfall approach were not disadvantageous to the implementation of the application, with recognition of these potential issues strengthening the techniques used in the development process. Such issues included not using appropriate tools for management and the necessity to trust the development team (Intland Software, 2021; Lucid Content Team, 2022; Tan, 2019).

The freedom facilitated by this methodology meant that it worked well with the evolving nature of requirements (see section 1.1), and responded well in the face of user feedback from testing. These changes occurred during the implementation stage shown in Figure 6.4.1. Communication between the developer and end users was enhanced by the occurrence of regular meetings for user testing. This consistency enabled empathy to be built for the users (Faller, 2019) and therefore improved the user experience of the application. The constant testing executed helped maintain a strong focus on the development of a well-rounded application.

Awareness of potential issues encouraged active engagement in the areas where problems arise with this methodology. Consideration was given to the tools used to manage the project, with the two main ones being Trello and Azure DevOps Repos (see sections 3.2 and 3.3). Trello enabled the creation of a range of Kanban boards used for all iterations of the project, including a product backlog (see Figures 4.3.2.1 and 4.3.2.2), while Azure DevOps Repos established a secure mechanism for managing application code in a professional capacity. As there was only one developer, accountability rested on the singular entity and choices made by them. Successful time estimations and development practices, alongside planning for verification and validation testing, are reflective of the dedication, work ethic and organisational skills possessed by the developer.

The Agile-Waterfall methodology provided an approach which merged the successful, monolithic waterfall approach with the more modern and innovative agile method. The compiling of these techniques ensured a well-established and contemporary style of development was applied to the implementation of the system created.

References

Accessible Web (2022) *WCAG Color Contrast Checker.* https://accessibleweb.com/color-contrast-checker/ [Accessed 23 March 2022].

Aggarwal, S. and Verma, J. (2018) Comparative analysis of MEAN stack and MERN stack. *International Journal of Recent Research Aspects*, *5*(1), pp.127-32.

Ahmad, K.S., Ahmad, N., Tahir, H. and Khan, S. (2017) Fuzzy_MoSCoW: A fuzzy based MoSCoW method for the prioritization of software requirements. In *2017 International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)* (pp. 433-437). IEEE.

Allaway, Z. (2017) Indoor Edible Garden. London: Dorling Kindersley Limited.

Atlantic Systems Guild Limited (2004) Volere Requirements Specification Template Edition 10.1. Available from: www.inf.ed.ac.uk/teaching/courses/seoc1/2005_2006/resources/volere-template.pdf [Accessed 22 April 2022].

Atlassian (2022) *Gitflow Workflow.* Available from: www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow [Accessed 20 April 2022].

AWS (2022) *What is A Document Database?* Available from: https://aws.amazon.com/nosql/document/ [Accessed 18 April 2022].

Babich, N. (2019) *The 4 Golden Rules of UI Design.* Available from: https://xd.adobe.com/ideas/process/ui-design/4-golden-rules-ui-design/ [Accessed 30 March 2022].

Beytes, C. (2013) *A College Class Asks: Why Don't People Garden?* Available from: www.growertalks.com/Article/?articleid=20101 [Accessed 14 September 2021].

Carroll, J.M. (1997) Human-computer interaction: psychology as a science of design. *Annual review of psychology*, *48*(1), pp.61-83.

Carroll, J.M. (2012) 2. Human Computer Interaction - brief intro. *In:* Soegaard, M. and Dam, R.F. The encyclopedia of human-computer interaction.

Casey, K. (2021) *Angular Vs. React Vs. Vue: A 2021 Comparison.* Available from: https://codersera.com/blog/angular-vs-react-vs-vue/ [18 April 2022].

Chapman, C. (2022) *Exploring the Gestalt Principles of Design.* Available from: www.toptal.com/designers/ui/gestalt-principles-of-design [Accessed 30 March 2022].

Clarke, L. (2019) *Dark mode isn't as good for your eyes as you believe.* Available from: www.wired.co.uk/article/dark-mode-chrome-android-ios-science [Accessed 22 April 2022].

Corrigan, S. (2022) *How to strategically use color in website design.* Available from: www.flux-academy.com/blog/how-to-strategically-use-color-in-website-design [Accessed 31 March 2022].

Costa, C. (2021) *Top Programming Languages and Their Uses.* Available from: www.kdnuggets.com/2021/05/top-programming-languages.html [Accessed 18 April 2022].

Creately (2021) *Use Case Diagram Relationships Explained with Examples.* Available from: https://creately.com/blog/diagrams/use-case-diagram-relationships/ [Accessed 17 April 2022].

D'Amato, C., Dino, A. and Tennent, E. (2005) iTunes User Testing Report. Available from: www.dinoanastasia.com/portfolio/622/622-UsabilityReport.pdf [Accessed 17 April 2022].

Doglio, F. (2015) *Pro REST API Development with Node. js*. Apress.

Enzyme (2022) *Introduction | Enzyme.* Available from: https://enzymejs.github.io/enzyme/ [Accessed 18 April 2022].

Faller, P. (2019) Putting Personas to Work in UX Design: What They Are and Why They're Important. Available from: https://xd.adobe.com/ideas/process/user-research/putting-personas-to-work-in-ux-design/ [Accessed 21 December 2021].

Ganguly, S. (2019) Most Popular Technology Stack To Choose From Full Stack Vs. MEAN Stack Vs. MERN Stack In 2020. Available from: https://medium.datadriveninvestor.com/most-popular-technology-stack-to-choose-from-full-stack-vs-mean-stack-vs-mern-stack-in-2019-d12c0a17439a [Accessed 24 October 2021].

Gibbons, S. (2018) *Journey Mapping 101.* Available from: www.nngroup.com/articles/journey-mapping-101/ [Accessed 17 April 2022].

Girvan, L. and Paul, D. (2017) *Agile and Business Analysis: Practical guidance for IT professionals*. BCS, The Chartered Institute for IT.

Goel, A. (2021) *10 Best Web Development Frameworks.* Available from: https://hackr.io/blog/web-development-frameworks [Accessed 18 April 2022].

Goltz, S. (2014) *A Closer Look At Personas: What They Are And How They Work | 1.* Available from: www.smashingmagazine.com/2014/08/a-closer-look-at-personas-part-1/ [Accessed 18 April 2022].

Google Developers (2022) Lighthouse. Available from: https://developers.google.com/web/tools/lighthouse [Accessed 18 April 2022].

Hamilton, T. (2022) Difference Between Verification and Validation in Software Testing with Example. Available from: www.guru99.com/verification-v-s-validation-in-a-software-testing.html [Accessed 19 April 2022].

HASpod (2020) *How To Use (And Understand) A 5x5 Risk Matrix.* Available from: www.haspod.com/blog/paperwork/5x5-risk-matrix [Accessed 26 September 2021].

Hessayon, D. G. (1995) The Fruit Expert. 2nd ed. London: Expert Books.

Hessayon, D. G. (1997) The Vegetable and Herb Expert. London: Expert Books.

Intland Software (2021) *When, Why, and How to use the Agile-Waterfall Hybrid Model.* Available from: https://content.intland.com/blog/agile/when-why-how-to-use-the-hybrid-model [Accessed 25 September 2021].

Jest (2022) *Delightful JavaScript Testing.* Available from: https://jestjs.io/ [Accessed 18 April 2022].

JWT (2022) *JSON Web Tokens*. Available from: https://jwt.io/ [Accessed 18 April 2022].

Khazanova, A. (2022) *Color Theory Fundamentals Every Web Designer Should Know.* Available from: https://elementor.com/blog/color-theory-web-design/ [Accessed 31 March 2022].

Krug, S. (2005) Don't Make Me Think! A Common Sense Approach to Web Usability. 2nd ed. New Riders Publishing: Berkeley, California USA.

Lewis, C. H. (1982) Using the "Thinking Aloud" Method In Cognitive Interface Design. Technical Report IBM RC-9265.

Lucidchart (2022) *UML Activity Diagram Tutorial.* Available from: www.lucidchart.com/pages/uml-activity-diagram [Accessed 17 April 2022].

Lucid Content Team (2022) Agile-Waterfall Hybrid: Is It Right for Your Team? Available from: www.lucidchart.com/blog/is-agile-waterfall-hybrid-right-for-your-team [Accessed 25 September 2021].

lyzerk (2020) *Lines of Code (LOC)*. Available from: https://marketplace.visualstudio.com/items?itemName=lyzerk.linecounter [Accessed 21 April 2022].

Masse, M. (2011) *REST API design rulebook: designing consistent RESTful web service interfaces*. O'Reilly Media, Inc.

Meyerovich, L.A. and Rabkin, A.S. (2013) Empirical analysis of programming language adoption. In *Proceedings of the 2013 ACM SIGPLAN international conference on Object oriented programming systems languages & applications* (pp. 1-18).

Middleton, K. (2018) *12 Reasons Why Gardening Is Good for You.* Available from: www.gardenbuildingsdirect.co.uk/blog/why-gardening-is-good-for-you/ [Accessed 27 September 2021].

MongoDB (2022a) *MERN Stack Explained.* Available from: www.mongodb.com/mern-stack [Accessed 03 March 2022].

MongoDB (2022b) *MongoDB Atlas The multi-cloud application data platform.* Available from: www.mongodb.com/atlas [Accessed 18 April 2022].

Mozilla (2022) *Same-origin policy - Web security | MDN*. Available from: https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy [Accessed 15 April 2022].

Nielsen, J. (1994a) *10 Usability Heuristics for User Interface Design*. Available from: www.nngroup.com/articles/ten-usability-heuristics [Accessed 30 March 2022].

Nielsen, J. (1994b) Usability engineering. Elsevier.

npm (2022a) *secure-env*. Available from: www.npmjs.com/package/secure-env [Accessed 15 April 2022].

npm (2022b) *bcryptjs.* Available from: www.npmjs.com/package/bcryptjs [Accessed 15 April 2022].

Pandit, P. and Tahiliani, S. (2015) AgileUAT: A framework for user acceptance testing based on user stories and acceptance criteria. *International Journal of Computer Applications*, *120*(10).

Postman (2022) *What is Postman?* Available from: www.postman.com/product/what-is-postman/ [Accessed 18 April 2022].

Prause, C.R., Scholten, M., Zimmermann, A., Reiners, R. and Eisenhauer, M. (2008) Managing the iterative requirements process in a multi-national project using an issue tracker. In *2008 IEEE International Conference on Global Software Engineering* (pp. 151-159). IEEE.

React (2022) *Introducing JSX – React.* Available at: https://reactjs.org/docs/introducing-jsx.html [Accessed 18 April 2022].

Reassemble (2019) User Acceptance Testing vs. Usability Testing: What is the Difference? Available from: https://medium.com/reassemble/user-acceptance-testing-vs-usability-testing-what-is-the-difference-8c4150e163e8 [Accessed 16 April 2022].

Redux (2022) *Getting Started with Redux | Redux*. Available at: https://redux.js.org/introduction/getting-started [Accessed 18 April 2022].

Reeves, S. (2020) Black Box And White Box Testing: Everything You Need To Know. Available from: www.goodcore.co.uk/blog/black-box-and-white-box-testing/ [Accessed 07 April 2022].

Rendle, R. (2016) *BEM 101.* Available from: https://css-tricks.com/bem-101/ [Accessed 22 April 2022].

RHS (2021) Annual Report and Consolidated Financial Statements 1 February 2020 – 31 January 2021. Available from: www.rhs.org.uk/about-the-rhs/pdfs/about-the-rhs/mission-and-strategy/past-annual-reports/rhs-annual-report-2020%E2%80%9321.pdf [Accessed 14 September 2021].

Schneiderman, B. (2016) Guidelines, Principles and Theories. *In:* Shneiderman, B., Plaisant, C., Cohen, M.S., Jacobs, S., Elmqvist, N. and Diakopoulos, N. 6th ed. *Designing the user interface: strategies for effective human-computer interaction*. Pearson.

Tan, B.Z.Y. (2019) Effective Agile + Waterfall Hybrid Project Management. Available from: https://medium.com/2359media/effective-agile-waterfall-hybrid-project-management-25af260aaa6d [Accessed 25 September 2021]

Usability.gov (2023) *Report Template: Usability Test*. Available from: www.usability.gov/how-to-and-tools/resources/templates/report-template-usability-test.html [Accessed 23 March 2022].

Vinugayathri (2020) *MVC vs Flux vs Redux – The Real Differences.* Available from: www.clariontech.com/blog/mvc-vs-flux-vs-redux-the-real-differences [Accessed 21 March 2022].

W3C (2018) *Web Content Accessibility Guidelines (WCAG) 2.1.* Available from: www.w3.org/TR/WCAG21/ [Accessed 31 March 2022].

Watson, K. (2021) *Is Dark Mode Better for Your Eyes?* Available from: www.healthline.com/health/is-dark-mode-better-for-your-eyes [Accessed 22 April 2022].

Webpack (2021) *Concepts.* Available from: https://webpack.js.org/concepts/ [Accessed 18 April 2022].

White, J. (2012) *500 Four Ingredient Recipes*. Leicestershire: Anness Publishing.

Wiegers, K. (1999) First things first: prioritizing requirements. *Software Development*, *7*(9), pp.48-53.

Zhou, B. (2016) *Risk Management Plan for Training Project of Green life Corporation.* Available from: www.belannazhou.com/risk-management-project/risk-management/ [Accessed 26 September 2021].

Bibliography

Alshehri, S. and Benedicenti, L. (2013) Using the analytical hierarchy process as a ranking tool for user story prioritization techniques. In *Proceedings of the 8th International Conference on Software Engineering Advances* (pp. 329-335).

Bevan, N. (2001) International standards for HCI and usability. *International journal of human-computer studies*, *55*(4), pp.533-552.

Gillis, A. S. (2020) Performance Testing. Available from: https://searchsoftwarequality.techtarget.com/definition/performance-testing [Accessed 18 April 2022].

Hudaib, A., Masadeh, R., Qasem, M.H. and Alzaqebah, A. (2018) Requirements prioritization techniques comparison. *Modern Applied Science*, *12*(2), pp.62.

Hujainah, F., Bakar, R.B.A., Abdulgabber, M.A. and Zamli, K.Z. (2018) Software requirements prioritisation: a systematic literature review on significance, stakeholders, techniques, and challenges. *IEEE Access*, *6*, pp.71497-71523.

Shaleynikov, A (2017) Top 10 Programming Languages in 2017. Available from: https://dzone.com/articles/top-10-programming-languages-in-2017 [Accessed 18 April 2022].

Appendix A: Final Requirements Final Format

The list of requirements selected for development within the project, shown largely through the prioritisation mechanisms in section 1.1 of this report, are described in detail below using the VOLERE format (Atlantic Systems Guild Limited, 2004; Prause et. al., 2018).

Customer satisfaction and dissatisfaction ratings are allocated on a Likert scale from 1 to 5, with 5 reflective of great happiness in terms of satisfaction and of great frustration for dissatisfaction (Atlantic Systems Guild Limited, 2004). Requirement numbers equate to values given in section 1.1 and supporting materials reference wireframes and prototypes developed, which are contained within Appendix B.

Requirement #:	FR01	**Requirement Type:**	Functional	**Use Case:**	N/A
Description:	Account sign-up				
Rationale:	Users require an account to access the application and its functionality.				
Source:	Naomi Stevenson				
Fit Criterion:	User successfully creates an account.				
Customer Satisfaction Rating:	3	**Customer Dissatisfaction Rating:**	5		
Dependencies:	N/A	**Conflicts:**	N/A		
History:	Created October 2021	**Supporting Materials:**	See Appendix B.2.2 and B.4.2		

Figure A.1: Requirement information for FR01, written in the VOLERE format.

Requirement #:	FR02	**Requirement Type:**	Functional	**Use Case:**	N/A
Description:	Account login				
Rationale:	Signing into application enables access to application functionality.				
Source:	Naomi Stevenson				
Fit Criterion:	User successfully signs into their account.				
Customer Satisfaction Rating:	3	**Customer Dissatisfaction Rating:**	5		
Dependencies:	FR01	**Conflicts:**	N/A		
History:	Created October 2021	**Supporting Materials:**	See Appendix B.2.3 and B.4.3		

Figure A.2: Requirement information for FR02, written in the VOLERE format.

Requirement #:	FR03	**Requirement Type:**	Functional	**Use Case:**	N/A
Description:	Account logout				
Rationale:	Logging out of account maintains application security and privacy of user information.				
Source:	Naomi Stevenson				
Fit Criterion:	User successfully logs out of their account.				
Customer Satisfaction Rating:	2	**Customer Dissatisfaction Rating:**	3		
Dependencies:	FR02	**Conflicts:**	N/A		
History:	Created October 2021	**Supporting Materials:**	See Appendix B.2.7 and B.4.7		

Figure A.3: Requirement information for FR03, written in the VOLERE format.

Requirement #:	FR04	**Requirement Type:**	Functional	**Use Case:**	7
Description:	Account update				
Rationale:	In the event users forget their password, want to change their password, or update their email, their account information can be altered.				
Source:	Naomi Stevenson				
Fit Criterion:	User successfully updated information from their account page, or the password reset feature.				
Customer Satisfaction Rating:	5	**Customer Dissatisfaction Rating:**	5		
Dependencies:	FR02	**Conflicts:**	N/A		
History:	Created October 2021	**Supporting Materials:**	See Appendix B.2.4, B.2.5, B.2.29, B.4.4, B.4.5 and B.4.29		

Figure A.4: Requirement information for FR04, written in the VOLERE format.

Requirement #:	FR05	**Requirement Type:**	Functional	**Use Case:**	7
Description:	Account deletion				
Rationale:	In the event a user no longer wishes to have an account with the application they should be able to delete it themselves.				
Source:	Naomi Stevenson				
Fit Criterion:	User successfully deletes account from the account page.				
Customer Satisfaction Rating:	4	**Customer Dissatisfaction Rating:**	5		
Dependencies:	FR02	**Conflicts:**	N/A		
History:	Created October 2021	**Supporting Materials:**	See Appendix B.2.6 and B.4.6		

Figure A.5: Requirement information for FR05, written in the VOLERE format.

Requirement #:	FR06	**Requirement Type:**	Functional	**Use Case:**	1, 2, 3, 4
Description:	Plant information				
Rationale:	Information on fruit, vegetables, herbs, and edible flowers should be available to be read in various application components.				
Source:	Naomi Stevenson				
Fit Criterion:	Produce information successfully loads in the interactive guide, planner, and produce areas.				
Customer Satisfaction Rating:	4	**Customer Dissatisfaction Rating:**	5		
Dependencies:	N/A	**Conflicts:**	N/A		
History:	Created October 2021	**Supporting Materials:**	See section 2.3.2 and Appendix B.3		

Figure A.6: Requirement information for FR06, written in the VOLERE format.

Requirement #:	FR07	**Requirement Type:**	Functional	**Use Case:**	2
Description:	View plant encyclopaedia				
Rationale:	User should be able to view a list of all fruit, vegetables, herbs, and edible flowers they can grow.				
Source:	Naomi Stevenson				
Fit Criterion:	User successfully views all produce from the produce page.				
Customer Satisfaction Rating:	5	**Customer Dissatisfaction Rating:**	5		
Dependencies:	FR02, FR06	**Conflicts:**	N/A		
History:	Created October 2021	**Supporting Materials:**	See Appendix B.2.16 and B.4.16		

Figure A.7: Requirement information for FR07, written in the VOLERE format.

Requirement #:	FR08	**Requirement Type:**	Functional	**Use Case:**	2
Description:	Plant encyclopaedia type filter				
Rationale:	User should be able to filter the plant list by type to find items easier.				
Source:	Naomi Stevenson				
Fit Criterion:	User successfully filters the items in the produce page by type.				
Customer Satisfaction Rating:	5	**Customer Dissatisfaction Rating:**	4		
Dependencies:	FR07	**Conflicts:**	N/A		
History:	Created October 2021	**Supporting Materials:**	See Appendix B.2.16 and B.4.16		

Figure A.8: Requirement information for FR08, written in the VOLERE format.

Requirement #:	FR09	**Requirement Type:**	Functional	**Use Case:**	2
Description:	Plant encyclopaedia search				
Rationale:	User should be able to filter the plant list by name to find items easier.				
Source:	Naomi Stevenson				
Fit Criterion:	User successfully filters the items in the produce page by name.				
Customer Satisfaction Rating:	5	**Customer Dissatisfaction Rating:**	3		
Dependencies:	FR07	**Conflicts:**	N/A		
History:	Created October 2021	**Supporting Materials:**	See Appendix B.2.16 and B.4.16		

Figure A.9: Requirement information for FR09, written in the VOLERE format.

Requirement #:	FR10	**Requirement Type:**	Functional	**Use Case:**	2
Description:	View single plant				
Rationale:	User should be able to view information for each specific plant to gain more insight into the produce item.				
Source:	Naomi Stevenson				
Fit Criterion:	User successfully views a single produce item from the single produce page.				
Customer Satisfaction Rating:	4	**Customer Dissatisfaction Rating:**	5		
Dependencies:	FR07	**Conflicts:**	N/A		
History:	Created October 2021	**Supporting Materials:**	See Appendix B.2.17 and B.4.17		

Figure A.10: Requirement information for FR10, written in the VOLERE format.

Requirement #:	FR11	**Requirement Type:**	Functional	**Use Case:**	2
Description:	Add plant to currently growing				
Rationale:	User should be able to add plant to area to record what they are currently growing.				
Source:	Naomi Stevenson				
Fit Criterion:	User successfully adds a produce item to their currently growing list from the single produce page.				
Customer Satisfaction Rating:	5	**Customer Dissatisfaction Rating:**	4		
Dependencies:	FR10	**Conflicts:**	N/A		
History:	Created October 2021	**Supporting Materials:**	See Appendix B.2.17 and B.4.17		

Figure A.11: Requirement information for FR11, written in the VOLERE format.

Requirement #:	FR12	**Requirement Type:**	Functional	**Use Case:**	2, 7
Description:	View currently growing				
Rationale:	User should be able to view all plants they are currently growing.				
Source:	Naomi Stevenson				
Fit Criterion:	User successfully views currently growing list from their account page.				
Customer Satisfaction Rating:	5	**Customer Dissatisfaction Rating:**	4		
Dependencies:	FR01, FR11	**Conflicts:**	N/A		
History:	Created October 2021	**Supporting Materials:**	See Appendix B.2.17, B.2.30, B.4.17 and B.4.30		

Figure A.12: Requirement information for FR12, written in the VOLERE format.

Requirement #:	FR13	**Requirement Type:**	Functional	**Use Case:**	2, 7
Description:	Delete plant from currently growing				
Rationale:	User should be able to remove plant from list of what they are currently growing.				
Source:	Naomi Stevenson				
Fit Criterion:	User successfully removes a produce item from their currently growing list.				
Customer Satisfaction Rating:	5	**Customer Dissatisfaction Rating:**	4		
Dependencies:	FR11, FR12	**Conflicts:**	N/A		
History:	Created October 2021	**Supporting Materials:**	See Appendix B.2.17, B.2.30, B.4.17 and B.4.30		

Figure A.13: Requirement information for FR13, written in the VOLERE format.

Requirement #:	FR15	**Requirement Type:**	Functional	**Use Case:**	2
Description:	Add plant to wish-list				
Rationale:	User should be able to save plant to their wish-list to record what they want to grow in the future.				
Source:	Naomi Stevenson				
Fit Criterion:	User successfully adds a produce item to their wish-list from the single produce page.				
Customer Satisfaction Rating:	5	**Customer Dissatisfaction Rating:**	4		
Dependencies:	FR10	**Conflicts:**	N/A		
History:	Created October 2021	**Supporting Materials:**	See Appendix B.2.17 and B.4.17		

Figure A.14: Requirement information for FR15, written in the VOLERE format.

Requirement #:	FR16	**Requirement Type:**	Functional	**Use Case:**	2, 7
Description:	View wish-list				
Rationale:	User should be able to view all plants in their wish-list.				
Source:	Naomi Stevenson				
Fit Criterion:	User successfully views wish-list from their account page.				
Customer Satisfaction Rating:	5	**Customer Dissatisfaction Rating:**	4		
Dependencies:	FR01, FR15	**Conflicts:**	N/A		
History:	Created October 2021	**Supporting Materials:**	See Appendix B.2.17, B.2.31, B.4.17 and B.4.31		

Figure A.15: Requirement information for FR16, written in the VOLERE format.

Requirement #:	FR17	**Requirement Type:**	Functional	**Use Case:**	2, 7
Description:	Remove plant from wish-list				
Rationale:	User should be able to remove plant from their wish-list.				
Source:	Naomi Stevenson				
Fit Criterion:	User successfully removes a produce item from their wish-list.				
Customer Satisfaction Rating:	5	**Customer Dissatisfaction Rating:**	4		
Dependencies:	FR15, FR16	**Conflicts:**	N/A		
History:	Created October 2021	**Supporting Materials:**	See Appendix B.2.17, B.2.31, B.4.17 and B.4.31		

Figure A.17: Requirement information for FR17, written in the VOLERE format.

Requirement #:	FR18	**Requirement Type:**	Functional	**Use Case:**	3
Description:	Open interactive guide				
Rationale:	User should be able to open the interactive guide to use the feature.				
Source:	Naomi Stevenson				
Fit Criterion:	User successfully opens the interactive guide.				
Customer Satisfaction Rating:	4	**Customer Dissatisfaction Rating:**	5		
Dependencies:	FR02	**Conflicts:**	N/A		
History:	Created October 2021	**Supporting Materials:**	See Appendix B.2.1 and B.4.1		

Figure A.17: Requirement information for FR18, written in the VOLERE format.

Requirement #:	FR19	**Requirement Type:**	Functional	**Use Case:**	3
Description:	Close interactive guide				
Rationale:	User should be able to close the interactive guide to use other areas of the application.				
Source:	Naomi Stevenson				
Fit Criterion:	User successfully closes the interactive guide.				
Customer Satisfaction Rating:	4	**Customer Dissatisfaction Rating:**	5		
Dependencies:	FR18	**Conflicts:**	N/A		
History:	Created October 2021	**Supporting Materials:**	See Appendix B.2.8, B.2.9, B.2.10, B.2.11, B.2.12, B.2.13, B.2.14, B.4.8, B.4.9, B.4.10, B.4.11, B.4.12, B.4.13 and B.4.14		

Figure A.18: Requirement information for FR19, written in the VOLERE format.

Requirement #:	FR20	**Requirement Type:**	Functional	**Use Case:**	3
Description:	Interactive guide parameters				
Rationale:	User should be able to interact with a guide that can help determine what the best plants to start growing at that particular time based on criteria, such as the season and available space.				
Source:	Naomi Stevenson				
Fit Criterion:	User successfully reaches the results section of the interactive guide and are presented with a list of produce.				
Customer Satisfaction Rating:	5	**Customer Dissatisfaction Rating:**	5		
Dependencies:	FR06, FR18	**Conflicts:**	N/A		
History:	Created October 2021	**Supporting Materials:**	See Appendix B.2.8, B.2.9, B.2.10, B.2.11, B.2.12, B.2.13, B.2.14, B.4.8, B.4.9, B.4.10, B.4.11, B.4.12, B.4.13 and B.4.14		

Figure A.19: Requirement information for FR20, written in the VOLERE format.

Requirement #:	FR21	**Requirement Type:**	Functional	**Use Case:**	3
Description:	Interactive guide response changes				
Rationale:	User should be able to go back and change their responses to questions asked during use of the interactive guide.				
Source:	Naomi Stevenson				
Fit Criterion:	User is able to navigate to a previous section of the guide and change their answer.				
Customer Satisfaction Rating:	4	**Customer Dissatisfaction Rating:**	3		
Dependencies:	FR18	**Conflicts:**	N/A		
History:	Created October 2021	**Supporting Materials:**	See Appendix B.2.8, B.2.9, B.2.10, B.2.11, B.2.12, B.2.13, B.2.14, B.4.8, B.4.9, B.4.10, B.4.11, B.4.12, B.4.13 and B.4.14		

Figure A.20: Requirement information for FR21, written in the VOLERE format.

Requirement #:	FR22	**Requirement Type:**	Functional	**Use Case:**	3
Description:	Interactive guide page redirect				
Rationale:	During use of the interactive guide the user should be redirected to the appropriate pages within the application based on their contact with the interface.				
Source:	Naomi Stevenson				
Fit Criterion:	User successfully clicks a produce item shown in the guide results and is redirected to the single produce page for the item.				
Customer Satisfaction Rating:	5	**Customer Dissatisfaction Rating:**	4		
Dependencies:	FR18	**Conflicts:**	N/A		
History:	Created October 2021	**Supporting Materials:**	See Appendix B.2.14 and B.4.14		

Figure A.21: Requirement information for FR22, written in the VOLERE format.

Requirement #:	FR23	**Requirement Type:**	Functional	**Use Case:**	1
Description:	View planner for all plants				
Rationale:	User should be able to view a planner detailing when to sow and harvest all produce based on the current month.				
Source:	Naomi Stevenson				
Fit Criterion:	User successfully views the planner page.				
Customer Satisfaction Rating:	4	**Customer Dissatisfaction Rating:**	4		
Dependencies:	FR06	**Conflicts:**	N/A		
History:	Created October 2021	**Supporting Materials:**	See Appendix B.2.15 and B.4.15		

Figure A.22: Requirement information for FR23, written in the VOLERE format.

Requirement #:	FR24	**Requirement Type:**	Functional	**Use Case:**	1
Description:	View planner for current plants				
Rationale:	User should be able to view a tailored version of the planner based on what they are currently growing.				
Source:	Naomi Stevenson				
Fit Criterion:	User successfully filters the items in the planner to show what is in their currently growing list.				
Customer Satisfaction Rating:	5	**Customer Dissatisfaction Rating:**	3		
Dependencies:	FR06, FR11	**Conflicts:**	N/A		
History:	Created October 2021	**Supporting Materials:**	See Appendix B.2.15 and B.4.15		

Figure A.23: Requirement information for FR24, written in the VOLERE format.

Requirement #:	FR25	**Requirement Type:**	Functional	**Use Case:**	1
Description:	Planner type filter				
Rationale:	User should be able to view a planner of each type of produce to find items easier.				
Source:	Naomi Stevenson				
Fit Criterion:	User successfully filters the planner by type.				
Customer Satisfaction Rating:	5	**Customer Dissatisfaction Rating:**	3		
Dependencies:	FR06	**Conflicts:**	N/A		
History:	Created March 2022	**Supporting Materials:**	See Appendix B.2.15 and B.4.15		

Figure A.24: Requirement information for FR25, written in the VOLERE format.

Requirement #:	FR26	**Requirement Type:**	Functional	**Use Case:**	1
Description:	Planner search by name				
Rationale:	User should be able to search the planner for produce by name to find items easier.				
Source:	Naomi Stevenson				
Fit Criterion:	User successfully filters the planner by name.				
Customer Satisfaction Rating:	5	**Customer Dissatisfaction Rating:**	4		
Dependencies:	FR06	**Conflicts:**	N/A		
History:	Created March 2022	**Supporting Materials:**	See Appendix B.2.15 and B.4.15		

Figure A.25: Requirement information for FR26, written in the VOLERE format.

Requirement #:	FR27	**Requirement Type:**	Functional	**Use Case:**	5
Description:	Access to articles				
Rationale:	User should be able to view a list of helpful advice articles containing top tips.				
Source:	Naomi Stevenson				
Fit Criterion:	User successfully views the articles page.				
Customer Satisfaction Rating:	3	**Customer Dissatisfaction Rating:**	3		
Dependencies:	FR02	**Conflicts:**	N/A		
History:	Created October 2021	**Supporting Materials:**	See Appendix B.2.22 and B.4.22		

Figure A.26: Requirement information for FR27, written in the VOLERE format.

Requirement #:	FR28	**Requirement Type:**	Functional	**Use Case:**	5
Description:	View a single article				
Rationale:	User should be able to view a single article.				
Source:	Naomi Stevenson				
Fit Criterion:	User successfully views a single article.				
Customer Satisfaction Rating:	3	**Customer Dissatisfaction Rating:**	3		
Dependencies:	FR27	**Conflicts:**	N/A		
History:	Created October 2021	**Supporting Materials:**	See Appendix B.2.23 and B.4.23		

Figure A.27: Requirement information for FR28, written in the VOLERE format.

Requirement #:	FR38	**Requirement Type:**	Functional	**Use Case:**	4
Description:	Recipe section				
Rationale:	User should be able to view a list of recipes for dishes which can be made using produce grown.				
Source:	Naomi Stevenson				
Fit Criterion:	User successfully views all recipes from the recipes page.				
Customer Satisfaction Rating:	4	**Customer Dissatisfaction Rating:**	5		
Dependencies:	FR02	**Conflicts:**	N/A		
History:	Created October 2021	**Supporting Materials:**	See Appendix B.2.18 and B.4.18		

Figure A.28: Requirement information for FR38, written in the VOLERE format.

Requirement #:	FR39	**Requirement Type:**	Functional	**Use Case:**	4
Description:	View single recipe				
Rationale:	User should be able to view all information for an individual recipe.				
Source:	Naomi Stevenson				
Fit Criterion:	User successfully views a single recipe.				
Customer Satisfaction Rating:	4	**Customer Dissatisfaction Rating:**	5		
Dependencies:	FR38	**Conflicts:**	N/A		
History:	Created October 2021	**Supporting Materials:**	See Appendix B.2.19 and B.4.19		

Figure A.29: Requirement information for FR39, written in the VOLERE format.

Requirement #:	FR45	**Requirement Type:**	Functional	**Use Case:**	4
Description:	Recipe search by title				
Rationale:	User should be able to search for recipes by title to find recipes easier.				
Source:	Naomi Stevenson				
Fit Criterion:	User successfully filters the items in the recipes page by title.				
Customer Satisfaction Rating:	5	**Customer Dissatisfaction Rating:**	5		
Dependencies:	FR38	**Conflicts:**	N/A		
History:	Created October 2021	**Supporting Materials:**	See Appendix B.2.18 and B.4.18		

Figure A.30: Requirement information for FR45, written in the VOLERE format.

Requirement #:	FR46	**Requirement Type:**	Functional	**Use Case:**	4
Description:	Recipe search by author				
Rationale:	User should be able to search for recipes by author to find recipes easier.				
Source:	Naomi Stevenson				
Fit Criterion:	User successfully filters the items in the recipes page by author.				
Customer Satisfaction Rating:	5	**Customer Dissatisfaction Rating:**	5		
Dependencies:	FR38	**Conflicts:**	N/A		
History:	Created October 2021	**Supporting Materials:**	See Appendix B.2.18 and B.4.18		

Figure A.31: Requirement information for FR46, written in the VOLERE format.

Requirement #:	FR47	**Requirement Type:**	Functional	**Use Case:**	4
Description:	Recipe search by ingredient				
Rationale:	User should be able to search for recipe by ingredient they can grow to find recipes easier.				
Source:	Naomi Stevenson				
Fit Criterion:	User successfully filters the items in the recipes page by ingredient from produce list.				
Customer Satisfaction Rating:	5	**Customer Dissatisfaction Rating:**	4		
Dependencies:	FR38	**Conflicts:**	N/A		
History:	Created October 2021	**Supporting Materials:**	See Appendix B.2.18 and B.4.18		

Figure A.32: Requirement information for FR47, written in the VOLERE format.

Requirement #:	FR48	**Requirement Type:**	Functional	**Use Case:**	4
Description:	Recipe search by diet				
Rationale:	User should be able to search for recipe by dietary requirement to find recipes easier.				
Source:	Naomi Stevenson				
Fit Criterion:	User successfully filters the items in the recipes page by dietary requirement.				
Customer Satisfaction Rating:	5	**Customer Dissatisfaction Rating:**	5		
Dependencies:	FR38	**Conflicts:**	N/A		
History:	Created October 2021	**Supporting Materials:**	See Appendix B.2.18 and B.4.18		

Figure A.33: Requirement information for FR48, written in the VOLERE format.

Requirement #:	FR49	**Requirement Type:**	Functional	**Use Case:**	4
Description:	Add recipe to favourites				
Rationale:	User should be able to favourite an individual recipe so they can record what they want to make in the future or what they enjoyed making.				
Source:	Naomi Stevenson				
Fit Criterion:	User successfully adds a recipe to their liked recipes list from the single recipe page.				
Customer Satisfaction Rating:	5	**Customer Dissatisfaction Rating:**	5		
Dependencies:	FR39	**Conflicts:**	N/A		
History:	Created October 2021	**Supporting Materials:**	See Appendix B.2.19 and B.4.19		

Figure A.34: Requirement information for FR49, written in the VOLERE format.

Requirement #:	FR50	**Requirement Type:**	Functional	**Use Case:**	4, 7
Description:	View recipe favourites				
Rationale:	User should be able to view favourites list.				
Source:	Naomi Stevenson				
Fit Criterion:	User successfully views their liked recipes list from their account page.				
Customer Satisfaction Rating:	5	**Customer Dissatisfaction Rating:**	5		
Dependencies:	FR01, FR49	**Conflicts:**	N/A		
History:	Created October 2021	**Supporting Materials:**	See Appendix B.2.19, B.2.32, B.4.19 and B.4.32		

Figure A.35: Requirement information for FR50, written in the VOLERE format.

Requirement #:	FR51	**Requirement Type:**	Functional	**Use Case:**	4, 7
Description:	Remove recipe from favourites				
Rationale:	User should be able to remove a recipe from their favourites list.				
Source:	Naomi Stevenson				
Fit Criterion:	User successfully removes a recipe from their liked recipes list.				
Customer Satisfaction Rating:	5	**Customer Dissatisfaction Rating:**	5		
Dependencies:	FR49, FR50	**Conflicts:**	N/A		
History:	Created October 2021	**Supporting Materials:**	See Appendix B.2.19, B.2.32, B.4.19 and B.4.32		

Figure A.36: Requirement information for FR51, written in the VOLERE format.

Requirement #:	FR52	**Requirement Type:**	Functional	**Use Case:**	4, 7
Description:	Recipe upload				
Rationale:	User should be able to upload their own recipe.				
Source:	Naomi Stevenson				
Fit Criterion:	User successfully uploads a new recipe to the application using the recipe form.				
Customer Satisfaction Rating:	4	**Customer Dissatisfaction Rating:**	4		
Dependencies:	FR02	**Conflicts:**	N/A		
History:	Created October 2021	**Supporting Materials:**	See Appendix B.2.20 and B.4.20		

Figure A.37: Requirement information for FR52, written in the VOLERE format.

Requirement #:	FR53	**Requirement Type:**	Functional	**Use Case:**	4, 7
Description:	View all recipe uploads				
Rationale:	User should be able to view all recipes they have uploaded.				
Source:	Naomi Stevenson				
Fit Criterion:	User successfully views their uploaded recipes list from their account page.				
Customer Satisfaction Rating:	5	**Customer Dissatisfaction Rating:**	5		
Dependencies:	FR52	**Conflicts:**	N/A		
History:	Created October 2021	**Supporting Materials:**	See Appendix B.2.18, B.2.33, B.4.18 and B.4.33		

Figure A.38: Requirement information for FR53, written in the VOLERE format.

Requirement #:	FR54	**Requirement Type:**	Functional	**Use Case:**	4, 7
Description:	Edit recipe uploaded				
Rationale:	User should be able to edit a recipe they have uploaded.				
Source:	Naomi Stevenson				
Fit Criterion:	User successfully edits a recipe they uploaded using the recipe form.				
Customer Satisfaction Rating:	5	**Customer Dissatisfaction Rating:**	5		
Dependencies:	FR52	**Conflicts:**	N/A		
History:	Created October 2021	**Supporting Materials:**	See Appendix B.2.19, B.2.20, B.2.33, B.4.19, B.4.20 and B.4.33		

Figure A.39: Requirement information for FR54, written in the VOLERE format.

Requirement #:	FR55	**Requirement Type:**	Functional	**Use Case:**	4, 7
Description:	Remove recipe uploaded				
Rationale:	User should be able to delete a recipe they have uploaded.				
Source:	Naomi Stevenson				
Fit Criterion:	User successfully deletes a recipe they uploaded.				
Customer Satisfaction Rating:	5	**Customer Dissatisfaction Rating:**	5		
Dependencies:	FR53	**Conflicts:**	N/A		
History:	Created October 2021	**Supporting Materials:**	See Appendix B.2.19, B.2.33, B.4.19 and B.4.33		

Figure A.40: Requirement information for FR55, written in the VOLERE format.

Requirement #:	FR75	**Requirement Type:**	Functional	**Use Case:**	N/A
Description:	Access permissions				
Rationale:	Users should be allocated an account type to enable access permissions.				
Source:	Naomi Stevenson				
Fit Criterion:	User successfully navigates to the admin page if they have admin permissions.				
Customer Satisfaction Rating:	5	**Customer Dissatisfaction Rating:**	4		
Dependencies:	FR01	**Conflicts:**	N/A		
History:	Created October 2021	**Supporting Materials:**	See Appendix B.2.24, B.2.25, B.4.24 and B.4.25		

Figure A.41: Requirement information for FR75, written in the VOLERE format.

Requirement #:	FR76	**Requirement Type:**	Functional	**Use Case:**	6
Description:	Admin application overview				
Rationale:	Admin users should be able to access a dashboard to monitor application activity.				
Source:	Naomi Stevenson				
Fit Criterion:	User successfully views the application overview on the admin page if they have admin permissions.				
Customer Satisfaction Rating:	5	**Customer Dissatisfaction Rating:**	3		
Dependencies:	FR75	**Conflicts:**	N/A		
History:	Created March 2022	**Supporting Materials:**	See Appendix B.2.24 and B.4.24		

Figure A.42: Requirement information for FR76, written in the VOLERE format.

Requirement #:	FR77	**Requirement Type:**	Functional	**Use Case:**	6
Description:	View admin user dashboard				
Rationale:	Admin users should be able to access a dashboard to view and modify basic user information.				
Source:	Naomi Stevenson				
Fit Criterion:	User successfully views the user dashboard on the admin page if they have admin permissions.				
Customer Satisfaction Rating:	5	**Customer Dissatisfaction Rating:**	5		
Dependencies:	FR75	**Conflicts:**	N/A		
History:	Created October 2021	**Supporting Materials:**	See Appendix B.2.25 and B.4.25		

Figure A.42: Requirement information for FR77, written in the VOLERE format.

Requirement #:	FR78	**Requirement Type:**	Functional	**Use Case:**	6
Description:	Admin toggle user permissions				
Rationale:	Admin users should be able to update user permissions, to give or revoke admin access.				
Source:	Naomi Stevenson				
Fit Criterion:	User successfully updates permission for a user on the admin page if they have admin permissions.				
Customer Satisfaction Rating:	5	**Customer Dissatisfaction Rating:**	5		
Dependencies:	FR75	**Conflicts:**	N/A		
History:	Created March 2022	**Supporting Materials:**	See Appendix B.2.25, B.2.27, B.4.25 and B.4.27		

Figure A.43: Requirement information for FR78, written in the VOLERE format.

Requirement #:	FR79	**Requirement Type:**	Functional	**Use Case:**	6
Description:	Admin send user password reset email				
Rationale:	Admin users should be able to send users a password reset email if they cannot remember their password.				
Source:	Naomi Stevenson				
Fit Criterion:	User successfully creates a password reset request for a user on the admin page if they have admin permissions.				
Customer Satisfaction Rating:	5	**Customer Dissatisfaction Rating:**	3		
Dependencies:	FR75	**Conflicts:**	N/A		
History:	Created March 2022	**Supporting Materials:**	See Appendix B.2.25, B.2.26, B.4.25 and B.4.26		

Figure A.44: Requirement information for FR79, written in the VOLERE format.

Requirement #:	FR80	**Requirement Type:**	Functional	**Use Case:**	6
Description:	Admin delete user account				
Rationale:	Admin users should be able to delete any user account.				
Source:	Naomi Stevenson				
Fit Criterion:	User successfully deletes a user account from the admin page if they have admin permissions.				
Customer Satisfaction Rating:	5	**Customer Dissatisfaction Rating:**	5		
Dependencies:	FR75	**Conflicts:**	N/A		
History:	Created March 2022	**Supporting Materials:**	See Appendix B.2.25, B.2.29, B.4.25 and B.4.29		

Figure A.45: Requirement information for FR80, written in the VOLERE format.

Requirement #:	FR81	**Requirement Type:**	Functional	**Use Case:**	4, 6
Description:	Admin content management				
Rationale:	Admin users should be able to remove user generated content if it violates community guidelines.				
Source:	Naomi Stevenson				
Fit Criterion:	User successfully deletes any recipe from the single recipe page if they have admin permissions.				
Customer Satisfaction Rating:	5	**Customer Dissatisfaction Rating:**	5		
Dependencies:	FR75	**Conflicts:**	N/A		
History:	Created October 2021	**Supporting Materials:**	See Appendix B.2.19 and B.4.19		

Figure A.46: Requirement information for FR81, written in the VOLERE format.

Requirement #:	NR01	**Requirement Type:**	Non-Functional	**Use Case:**	N/A
Description:	Accessibility based on personal ability				
Rationale:	The application should be able to be used by people with visual, auditory, physical speech, cognitive and neurological disabilities.				
Source:	Naomi Stevenson				
Fit Criterion:	Application is successfully used by people with visual, auditory, physical speech, cognitive and neurological disabilities				
Customer Satisfaction Rating:	5	**Customer Dissatisfaction Rating:**	5		
Dependencies:	N/A	**Conflicts:**	N/A		
History:	Created October 2021	**Supporting Materials:**	N/A		

Figure A.47: Requirement information for NR01, written in the VOLERE format.

Requirement #:	NR02	**Requirement Type:**	Non-Functional	**Use Case:**	N/A
Description:	Accessibility based on device				
Rationale:	The application should be able to be used across various device screen sizes.				
Source:	Naomi Stevenson				
Fit Criterion:	The application is successfully used in desktop and mobile view.				
Customer Satisfaction Rating:	5	**Customer Dissatisfaction Rating:**	5		
Dependencies:	N/A	**Conflicts:**	N/A		
History:	Created October 2021	**Supporting Materials:**	See Appendix B.2 and B.4		

Figure A.48: Requirement information for NR02, written in the VOLERE format.

Requirement #:	NR03	**Requirement Type:**	Non-Functional	**Use Case:**	N/A
Description:	Security				
Rationale:	The application should store user information in a secure manner.				
Source:	Naomi Stevenson				
Fit Criterion:	Passwords and connection strings to databases are encrypted and endpoints require authorisation tokens for access.				
Customer Satisfaction Rating:	5	**Customer Dissatisfaction Rating:**	5		
Dependencies:	N/A	**Conflicts:**	N/A		
History:	Created October 2021	**Supporting Materials:**	N/A		

Figure A.49: Requirement information for NR03, written in the VOLERE format.

Requirement #:	NR04	**Requirement Type:**	Non-Functional	**Use Case:**	N/A
Description:	Privacy				
Rationale:	The application should only utilise user information for necessary tasks to carry out specific functionality.				
Source:	Naomi Stevenson				
Fit Criterion:	The application only utilises user information for necessary tasks to carry out specific functionality and only returns required information in API requests.				
Customer Satisfaction Rating:	5	**Customer Dissatisfaction Rating:**	5		
Dependencies:	N/A	**Conflicts:**	N/A		
History:	Created October 2021	**Supporting Materials:**	N/A		

Figure A.50: Requirement information for NR04, written in the VOLERE format.

Requirement #:	NR05	**Requirement Type:**	Non-Functional	**Use Case:**	N/A
Description:	Response time				
Rationale:	The application should have a fast response time.				
Source:	Naomi Stevenson				
Fit Criterion:	The application successfully loads within a short space of time.				
Customer Satisfaction Rating:	5	**Customer Dissatisfaction Rating:**	4		
Dependencies:	N/A	**Conflicts:**	N/A		
History:	Created October 2021	**Supporting Materials:**	N/A		

Figure A.51: Requirement information for NR05, written in the VOLERE format.

Requirement #:	NR06	**Requirement Type:**	Non-Functional	**Use Case:**	N/A
Description:	User friendly				
Rationale:	The application should use natural language.				
Source:	Naomi Stevenson				
Fit Criterion:	The application successfully uses natural language and information can be understood by all end users.				
Customer Satisfaction Rating:	4	**Customer Dissatisfaction Rating:**	3		
Dependencies:	N/A	**Conflicts:**	N/A		
History:	Created October 2021	**Supporting Materials:**	N/A		

Figure A.52: Requirement information for NR06, written in the VOLERE format.

Requirement #:	NR07	**Requirement Type:**	Non-Functional	**Use Case:**	N/A
Description:	Consistency				
Rationale:	The application should have a consistent user interface.				
Source:	Naomi Stevenson				
Fit Criterion:	The application has a consistent user interface across all elements of the application.				
Customer Satisfaction Rating:	4	**Customer Dissatisfaction Rating:**	3		
Dependencies:	N/A	**Conflicts:**	N/A		
History:	Created October 2021	**Supporting Materials:**	See Appendix B.2 and B.4		

Figure A.53: Requirement information for NR07, written in the VOLERE format.

Requirement #:	NR08	**Requirement Type:**	Non-Functional	**Use Case:**	N/A
Description:	Application aesthetic				
Rationale:	The user interface should be aesthetically pleasing.				
Source:	Naomi Stevenson				
Fit Criterion:	The user interface is aesthetically pleasing, with an appropriate colour palette and layout.				
Customer Satisfaction Rating:	5	**Customer Dissatisfaction Rating:**	4		
Dependencies:	N/A	**Conflicts:**	N/A		
History:	Created October 2021	**Supporting Materials:**	See Appendix B.2 and B.4		

Figure A.54: Requirement information for NR08, written in the VOLERE format.

Requirement #:	NR09	**Requirement Type:**	Non-Functional	**Use Case:**	N/A
Description:	Notification of success				
Rationale:	The application should notify the user of successful actions.				
Source:	Naomi Stevenson				
Fit Criterion:	The application notifies the user of successful actions through success messages.				
Customer Satisfaction Rating:	5	**Customer Dissatisfaction Rating:**	5		
Dependencies:	N/A	**Conflicts:**	N/A		
History:	Created October 2021	**Supporting Materials:**	N/A		

Figure A.55: Requirement information for NR09, written in the VOLERE format.

Requirement #:	NR10	**Requirement Type:**	Non-Functional	**Use Case:**	N/A
Description:	Notification of failure				
Rationale:	The application should notify the user of unsuccessful actions.				
Source:	Naomi Stevenson				
Fit Criterion:	The application notifies the user of unsuccessful actions through error messages.				
Customer Satisfaction Rating:	5	**Customer Dissatisfaction Rating:**	5		
Dependencies:	N/A	**Conflicts:**	N/A		
History:	Created October 2021	**Supporting Materials:**	N/A		

Figure A.56: Requirement information for NR10, written in the VOLERE format.

Requirement #:	NR11	**Requirement Type:**	Non-Functional	**Use Case:**	N/A
Description:	Availability				
Rationale:	The application should be available for use 24/7.				
Source:	Naomi Stevenson				
Fit Criterion:	The application is available for use 24/7.				
Customer Satisfaction Rating:	4	**Customer Dissatisfaction Rating:**	5		
Dependencies:	N/A	**Conflicts:**	N/A		
History:	Created October 2021	**Supporting Materials:**	N/A		

Figure A.57: Requirement information for NR11, written in the VOLERE format.

Requirement #:	NR12	**Requirement Type:**	Non-Functional	**Use Case:**	N/A
Description:	Scalability				
Rationale:	The application should be able to adapt to usage requirements, such as number of people using the application at the same time.				
Source:	Naomi Stevenson				
Fit Criterion:	The application can successfully adapt to usage requirements, such as number of people using the application at the same time.				
Customer Satisfaction Rating:	3	**Customer Dissatisfaction Rating:**	5		
Dependencies:	N/A	**Conflicts:**	N/A		
History:	Created October 2021	**Supporting Materials:**	N/A		

Figure A.58: Requirement information for NR12, written in the VOLERE format.

Requirement #:	NR13	**Requirement Type:**	Non-Functional	**Use Case:**	N/A
Description:	General information				
Rationale:	Basic information about the application should be available, such as developer and version information.				
Source:	Naomi Stevenson				
Fit Criterion:	Basic information about the application is available on an about page.				
Customer Satisfaction Rating:	2	**Customer Dissatisfaction Rating:**	4		
Dependencies:	N/A	**Conflicts:**	N/A		
History:	Created October 2021	**Supporting Materials:**	N/A		

Figure A.59: Requirement information for NR13, written in the VOLERE format.

Appendix B: Additional Artefacts required for the Project

B.1: Relative Weighting Prioritisation Calculations and Results

Requirement Prioritization Model (Functional)

User instructions:

1. The input columns (starting at **Requirement #1**) are **Relative Importance(C), Relative Penalty(D), Relative Cost(G) and Relative Risk (I). Lowest-1, highest-9**.
2. The **Relative Weight** of each input can be adjusted (**Row 10, Columns C, D, G, and I**). The higher the number, the more weigh.
3. After entering the inputs, sort on the **Column K** starting from **Row 14** and the **"Requirement"** column in descending order to see the risk priorities shown in **Column K**.
4. Requirements with the same calculated priority are ordered by ascending **"Item #"**.

	Relative Weights:	2.0	1.0			1.0		0.5		
		Relative Importance	**Relative Penalty**	**Total Value**	**Value %**	**Relative Cost**	**Cost %**	**Relative Risk**	**Risk %**	**Priority**
	Totals	451	468	1370	100	379	100.000	348	100.000	
Item #	**Requirement**									
5	**Account deletion**	8	8	24	1.7	2	0.524	2	0.563	2.168
3	**Account logout**	9	7	25	1.8	3	0.785	2	0.563	1.704
13	**Delete plant from currently growing**	5	6	16	1.2	2	0.524	2	0.563	1.445
17	**Remove plant from wish-list**	5	5	15	1.1	2	0.524	2	0.563	1.355
19	**Close interactive guide**	9	8	26	1.9	3	0.785	5	1.408	1.269
51	**Remove recipe from favourites**	4	5	13	0.9	2	0.524	2	0.563	1.174
2	**Account login**	9	9	27	2.0	5	1.309	3	0.845	1.134
60	**View all forum posts**	8	6	22	1.6	4	1.047	3	0.845	1.089
16	**Add plant to wish-list**	6	6	18	1.3	3	0.785	3	0.845	1.084
1	**Account sign-up**	9	9	27	2.0	5	1.309	4	1.127	1.049
73	**View results from single quiz**	6	7	19	1.4	3	0.785	4	1.127	1.025
50	**View favourite recipes**	5	5	15	1.1	3	0.785	2	0.563	1.022
18	**Open interactive guide**	9	8	26	1.9	4	1.047	6	1.690	0.999
75	**Access permissions**	9	9	27	2.0	5	1.309	5	1.408	0.975

55	**Remove recipe uploaded**	5	6	16	1.2	3	0.785	3	0.845	0.963
34	**Delete comment on articles**	4	6	14	1.0	3	0.785	2	0.563	0.954
68	**Forum post search by category**	6	7	19	1.4	4	1.047	3	0.845	0.940
12	**View currently growing**	6	5	17	1.2	3	0.785	4	1.127	0.917
59	**Add post to forum**	8	7	23	1.7	5	1.309	4	1.127	0.893
38	**Recipe section**	8	8	24	1.7	6	1.571	3	0.845	0.876
27	**Access to articles**	6	7	19	1.4	4	1.047	4	1.127	0.858
46	**Recipe search by author**	4	6	14	1.0	3	0.785	3	0.845	0.843
45	**Recipe search by title**	5	7	17	1.2	4	1.047	3	0.845	0.841
25	**Planner type filter**	6	5	17	1.2	3	0.785	5	1.408	0.830
26	**Planner search by name**	4	4	12	0.9	2	0.524	4	1.127	0.803
57	**View single forum post**	7	5	19	1.4	4	1.047	5	1.408	0.789
32	**View all article comments**	4	5	13	0.9	3	0.785	3	0.845	0.783
62	**Remove forum post**	4	5	13	0.9	3	0.785	3	0.845	0.783
80	**Admin delete user account**	6	7	19	1.4	3	0.785	7	1.972	0.780
28	**View a single article**	7	6	20	1.5	5	1.309	4	1.127	0.777
21	**Interactive guide response changes**	8	9	25	1.8	6	1.571	6	1.690	0.753
35	**Location entry**	6	7	19	1.4	6	1.571	2	0.563	0.746
66	**Remove comment on forum post**	4	4	12	0.9	3	0.785	3	0.845	0.723
47	**Recipe search by ingredient**	7	8	22	1.6	6	1.571	5	1.408	0.703
14	**Currently growing sort**	4	5	13	0.9	3	0.785	4	1.127	0.701
79	**Admin send user password reset email**	5	3	13	0.9	3	0.785	4	1.127	0.701
49	**Add recipe to favourites**	5	4	14	1.0	4	1.047	3	0.845	0.693
74	**View results from all quizzes taken**	4	6	14	1.0	4	1.047	3	0.845	0.693
30	**Search articles using keywords**	6	7	19	1.4	5	1.309	5	1.408	0.686
44	**Delete recipe comment**	3	4	10	0.7	3	0.785	2	0.563	0.682
56	**Access to all forum content**	7	6	20	1.5	6	1.571	4	1.127	0.682
81	**Admin content management**	8	9	25	1.8	6	1.571	8	2.254	0.674
11	**Add plant to currently growing**	5	6	16	1.2	4	1.047	5	1.408	0.664
63	**Add comment to forum post**	5	6	16	1.2	4	1.047	5	1.408	0.664
78	**Admin toggle user permissions**	7	8	22	1.6	6	1.571	6	1.690	0.662
53	**View all recipe uploads**	6	6	18	1.3	5	1.309	5	1.408	0.650
23	**View planner for all plants**	8	9	25	1.8	8	2.094	5	1.408	0.650
31	**Comment on article**	5	4	14	1.0	4	1.047	4	1.127	0.632

39	**View single recipe**	7	8	22	1.6	7	1.832	5	1.408	0.631
15	**View wish-list**	5	6	16	1.2	5	1.309	4	1.127	0.622
6	**Plant information**	9	9	27	2.0	8	2.094	8	2.254	0.610
48	**Recipe search by diet**	6	8	20	1.5	7	1.832	4	1.127	0.607
24	**View planner for current plants**	7	9	23	1.7	7	1.832	7	1.972	0.594
9	**Plant encyclopaedia search**	6	2	14	1.0	5	1.309	3	0.845	0.588
69	**Forum post search by title**	4	6	14	1.0	5	1.309	3	0.845	0.588
61	**Edit forum post**	6	4	16	1.2	5	1.309	5	1.408	0.578
4	**Account update**	6	6	18	1.3	6	1.571	5	1.408	0.575
10	**View single plant**	6	7	19	1.4	6	1.571	6	1.690	0.572
29	**View all article comments**	4	5	13	0.9	5	1.309	3	0.845	0.546
58	**View all comments on single post**	4	4	12	0.9	4	1.047	4	1.127	0.542
20	**Interactive guide parameters**	8	9	25	1.8	8	2.094	9	2.535	0.541
7	**Plant encyclopaedia**	7	8	22	1.6	7	1.832	8	2.254	0.541
70	**Links to purchase seeds**	3	3	9	0.7	2	0.524	5	1.408	0.533
77	**Admin user dashboard**	8	7	23	1.7	8	2.094	8	2.254	0.519
54	**Edit recipe uploaded**	6	5	17	1.2	6	1.571	6	1.690	0.512
8	**Plant encyclopaedia type filter**	6	3	15	1.1	6	1.571	4	1.127	0.511
76	**Admin application overview**	6	3	15	1.1	6	1.571	4	1.127	0.511
41	**View all comments on a single recipe**	4	5	13	0.9	5	1.309	4	1.127	0.505
52	**Recipe upload**	7	4	18	1.3	7	1.832	6	1.690	0.489
22	**Interactive guide page redirect**	4	5	13	0.9	5	1.309	6	1.690	0.439
40	**Comment on recipe**	3	5	11	0.8	5	1.309	4	1.127	0.427
71	**View quizzes**	5	4	14	1.0	6	1.571	6	1.690	0.421
36	**Tailored plant care**	5	7	17	1.2	8	2.094	7	1.972	0.401
42	**View all recipe comments**	2	4	8	0.6	4	1.047	3	0.845	0.396
72	**Take a single quiz**	5	5	15	1.1	7	1.832	7	1.972	0.387
33	**Edit comment on articles**	3	2	8	0.6	4	1.047	4	1.127	0.361
37	**Notifications for plant care**	5	6	16	1.2	8	2.094	8	2.254	0.361
65	**Edit comment on forum post**	3	4	10	0.7	6	1.571	5	1.408	0.320
64	**View all comments made on forum**	3	4	10	0.7	7	1.832	4	1.127	0.304
43	**Edit recipe comment**	2	3	7	0.5	5	1.309	4	1.127	0.272
67	**Forum post search by author**	2	2	6	0.4	4	1.047	4	1.127	0.271

Requirements Prioritization Model (Non-functional)

User instructions:

1. The input columns (starting at **Requirement #1**) are **Relative Importance(C), Relative Penalty(D), Relative Cost(G) and Relative Risk (I). Lowest-1, highest-9**.
2. The **Relative Weight** of each input can be adjusted (**Row 10, Columns C, D, G and I**). The higher the number, the more weigh.
3. After entering the inputs, sort on the **Column K** starting from **Row 14** and the **"Requirement"** column in descending order to see the risk priorities shown in **Column K**.

	Relative Weights:	2.0	1.0			1.0		0.5		
		Relative Importance	**Relative Penalty**	**Total Value**	**Value %**	**Relative Cost**	**Cost %**	**Relative Risk**	**Risk %**	**Priority**
	Totals	90	104	284	100	48	100.000	53	100.000	
Item #	**Requirement**									
13	**General information**	4	5	13	4.6	1	2.083	2	3.774	1.153
9	**Notification of success**	7	7	21	7.4	3	6.250	3	5.660	0.814
4	**Privacy**	9	9	27	9.5	3	6.250	6	11.321	0.798
1	**Accessibility based on personal ability**	9	9	27	9.5	4	8.333	4	7.547	0.785
2	**Accessibility based on device**	7	8	22	7.7	3	6.250	4	7.547	0.773
11	**Availability**	8	9	25	8.8	3	6.250	6	11.321	0.739
8	**Application aesthetic**	4	8	16	5.6	3	6.250	2	3.774	0.692
5	**Response time**	6	8	20	7.0	4	8.333	3	5.660	0.631
12	**Scalability**	7	8	22	7.7	3	6.250	7	13.208	0.603
10	**Notification of failure**	7	8	22	7.7	4	8.333	5	9.434	0.594
3	**Security**	9	9	27	9.5	5	10.417	7	13.208	0.559
6	**User friendly**	8	9	25	8.8	7	14.583	2	3.774	0.534
7	**Consistency**	5	7	17	6.0	5	10.417	2	3.774	0.487

B.2: Application Wireframes

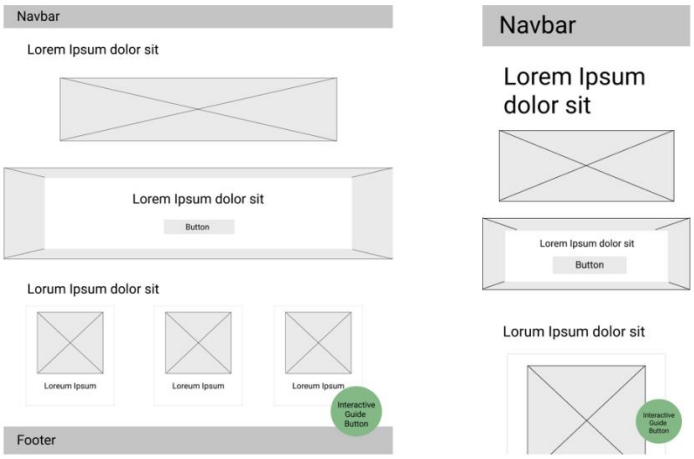


Figure B.2.1: Wireframes for desktop and mobile views of the homepage to be included in the application.

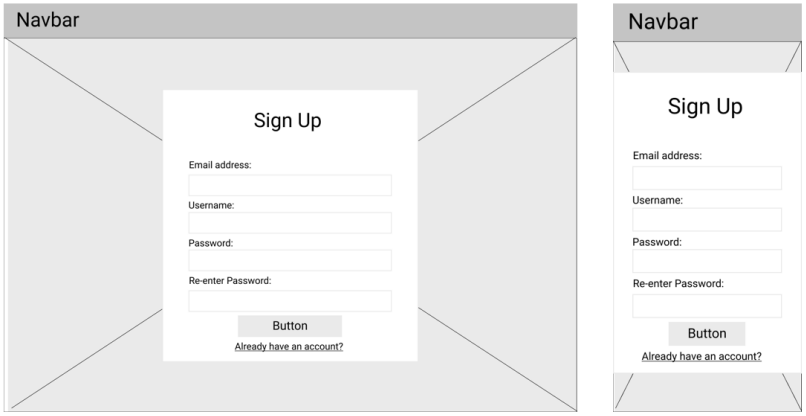


Figure B.2.2: Wireframes for desktop and mobile views of the sign-up page to be included in the application.

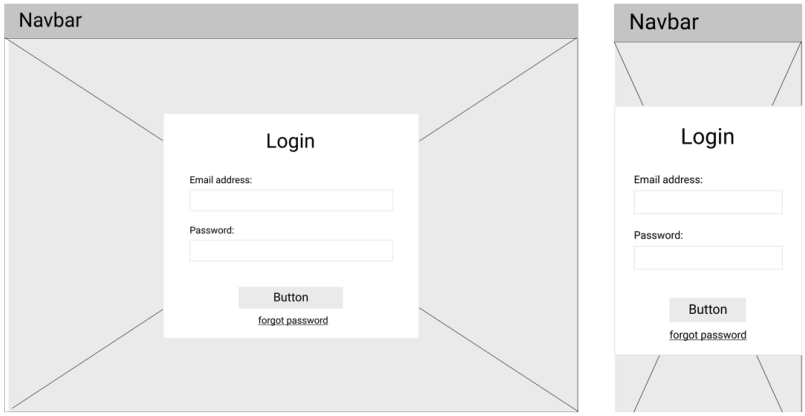


Figure B.2.3: Wireframes for desktop and mobile views of the login page to be included in the application.

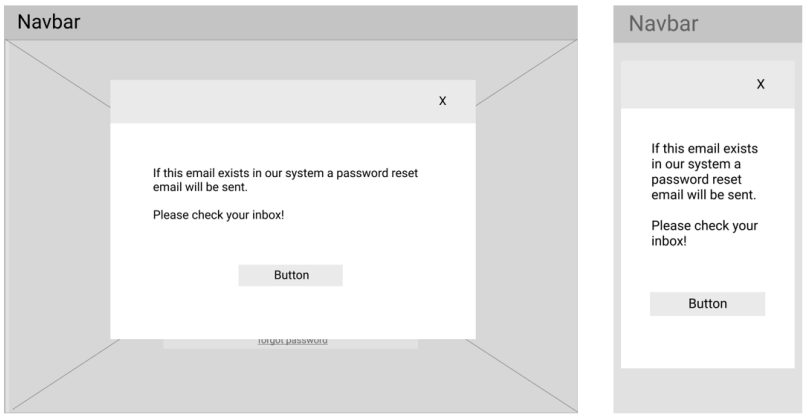


Figure B.2.4: Wireframes for desktop and mobile views of the forgot password modal to be included in the application.

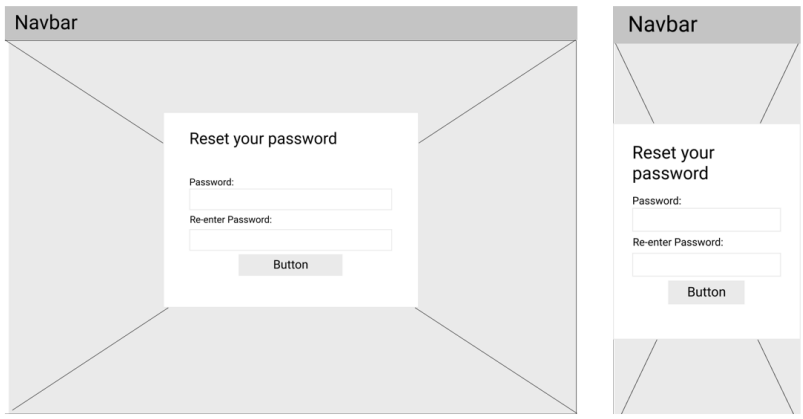


Figure B.2.5: Wireframes for desktop and mobile views of the reset password page to be included in the application.

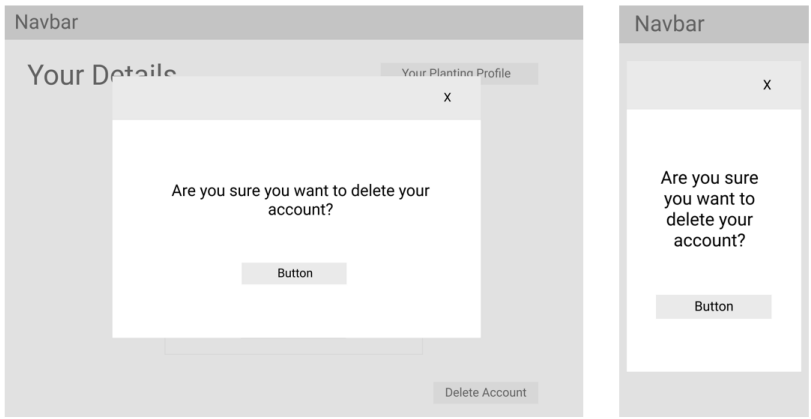


Figure B.2.6: Wireframes for desktop and mobile views of the delete account modal to be included in the application.

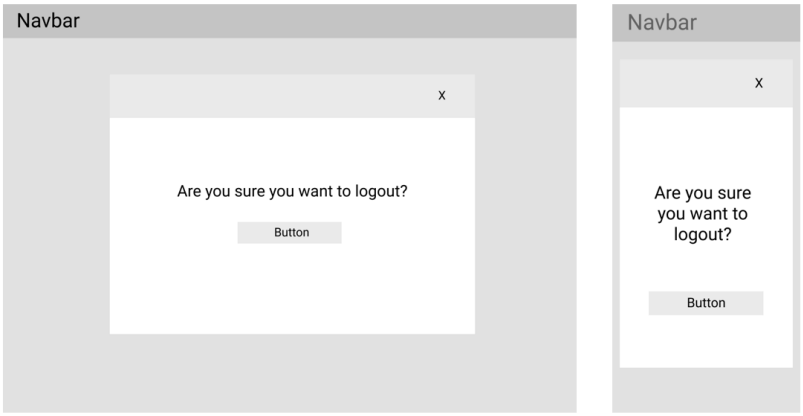


Figure B.2.7: Wireframes for desktop and mobile views of the logout modal to be included in the application.

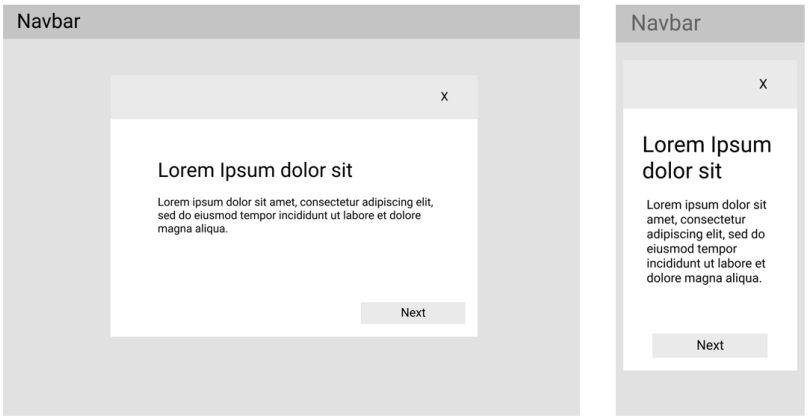


Figure B.2.8: Wireframes for desktop and mobile views of the first section of the interactive guide to be included in the application, introducing the user to the guide.

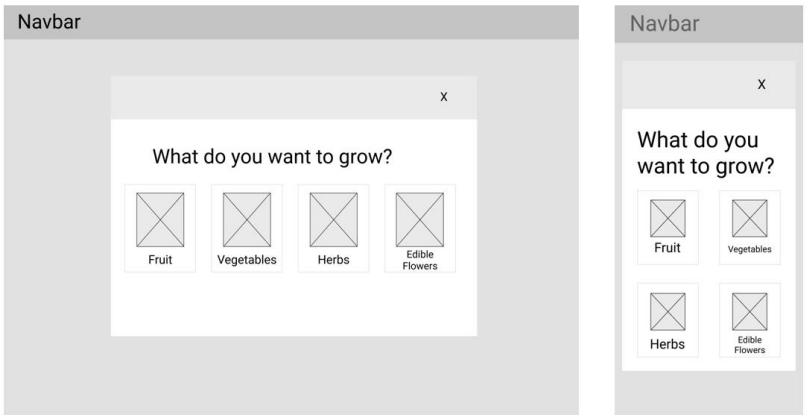


Figure B.2.9: Wireframes for desktop and mobile views of the second section of the interactive guide to be included in the application, asking the user what they want to grow.

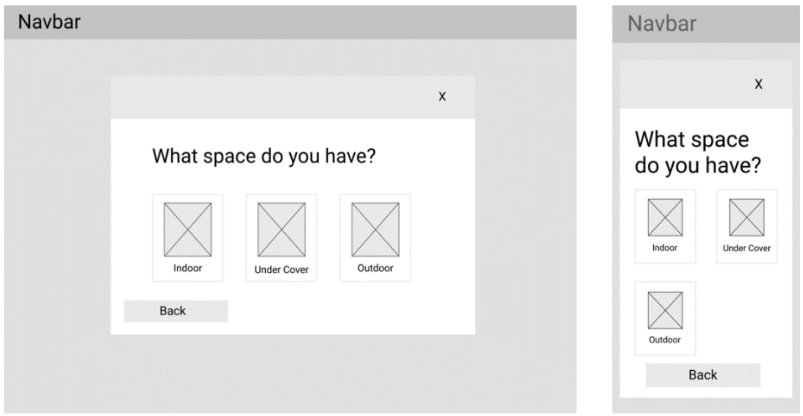


Figure B.2.10: Wireframes for desktop and mobile views of the third section of the interactive guide to be included in the application, asking the user what space they have.

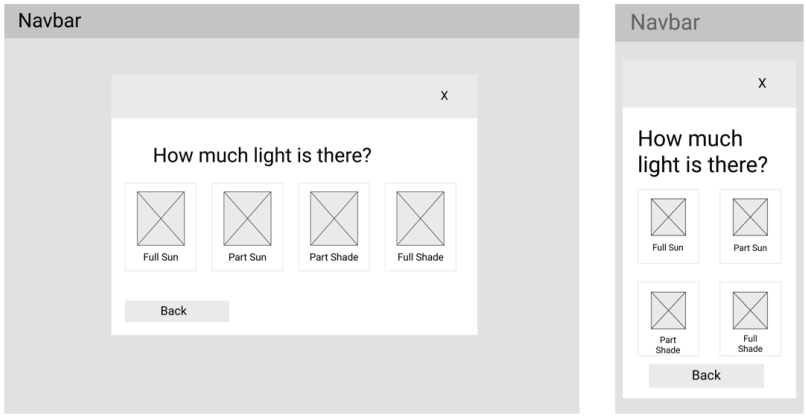


Figure B.2.11: Wireframes for desktop and mobile views of the fourth section of the interactive guide to be included in the application, asking the user how much light there is in this space.

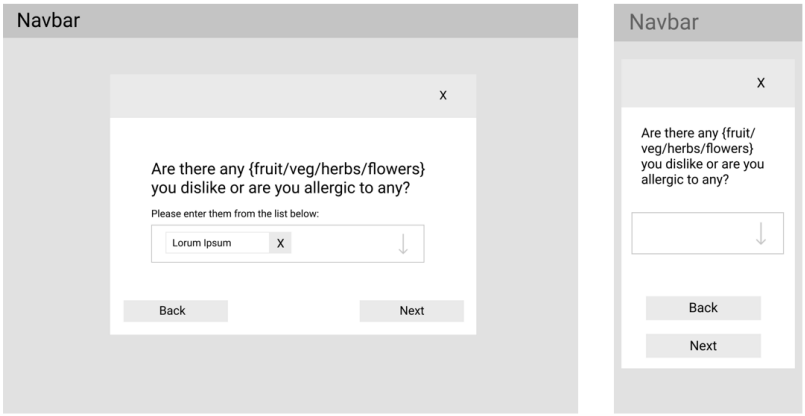


Figure B.2.12: Wireframes for desktop and mobile views of the fifth section of the interactive guide to be included in the application, asking the user if they have any allergies or if there is any produce they do not like.

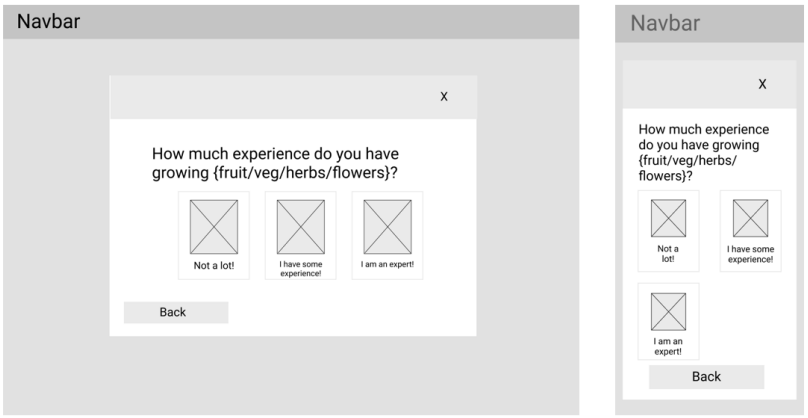


Figure B.2.13: Wireframes for desktop and mobile views of the sixth section of the interactive guide to be included in the application, asking the user about their gardening experience.

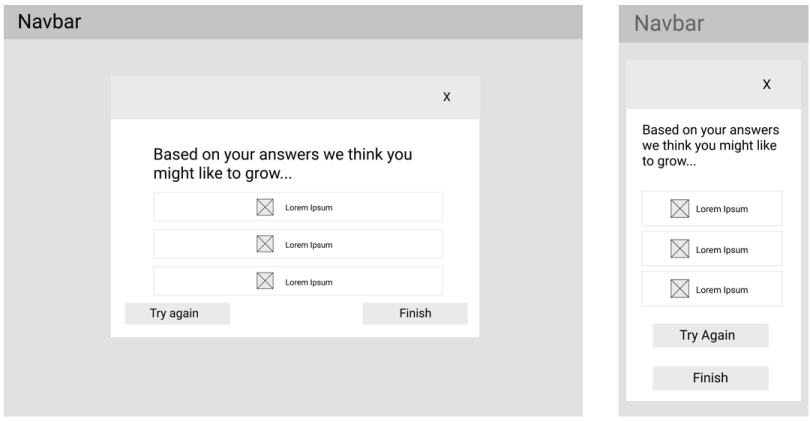


Figure B.2.14: Wireframes for desktop and mobile views of the seventh section of the interactive guide to be included in the application, asking the user what they want to grow.

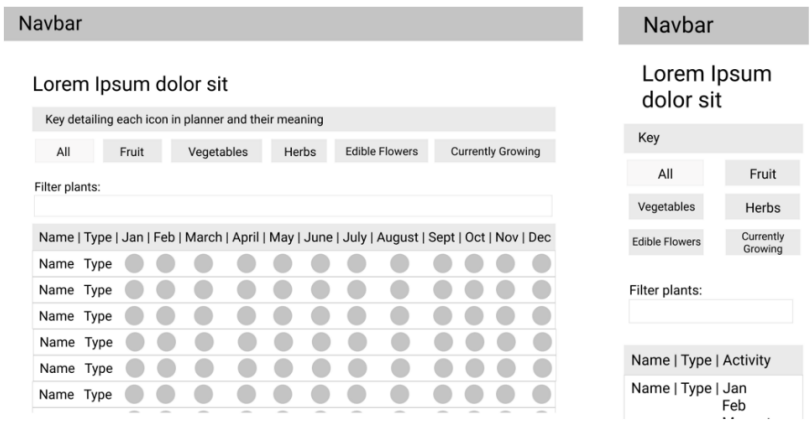


Figure B.2.15: Wireframes for desktop and mobile views of the planner to be included in the application, showing an overall view of the plant activities required for each plant per month.

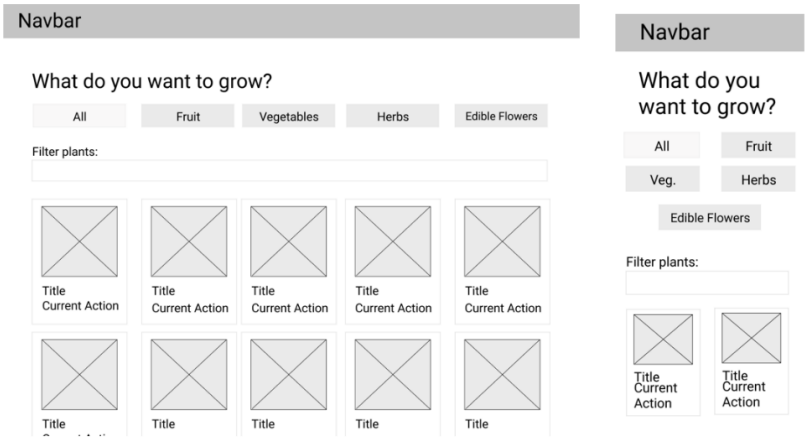


Figure B.2.16: Wireframes for desktop and mobile views of the produce page to be included in the application.

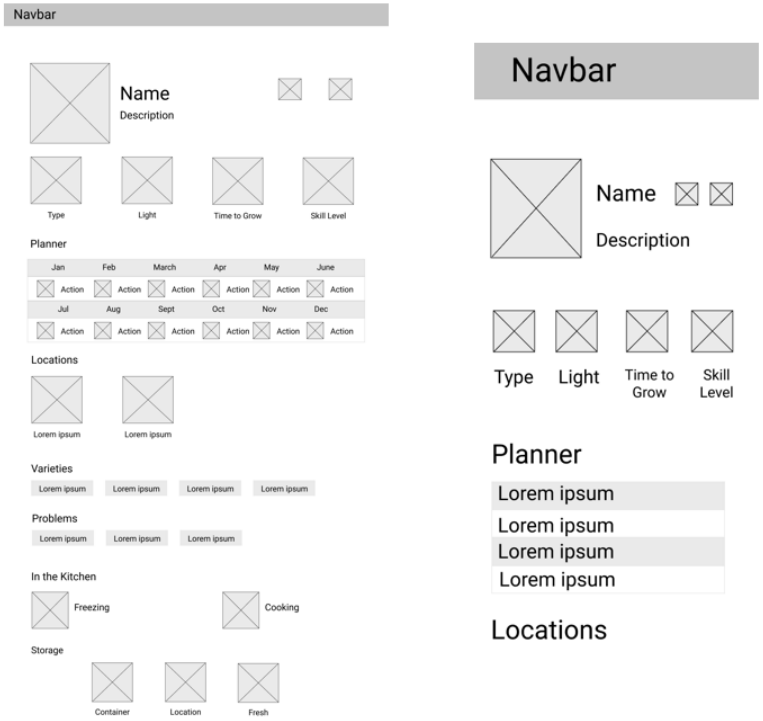


Figure B.2.17: Wireframes for desktop and mobile views of the produce page for a single produce item to be included in the application.

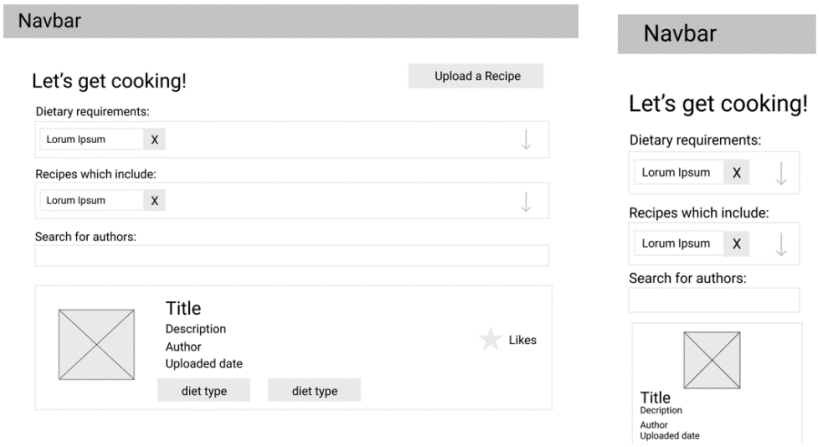


Figure B.2.18: Wireframes for desktop and mobile views of the recipes page to be included in the application.

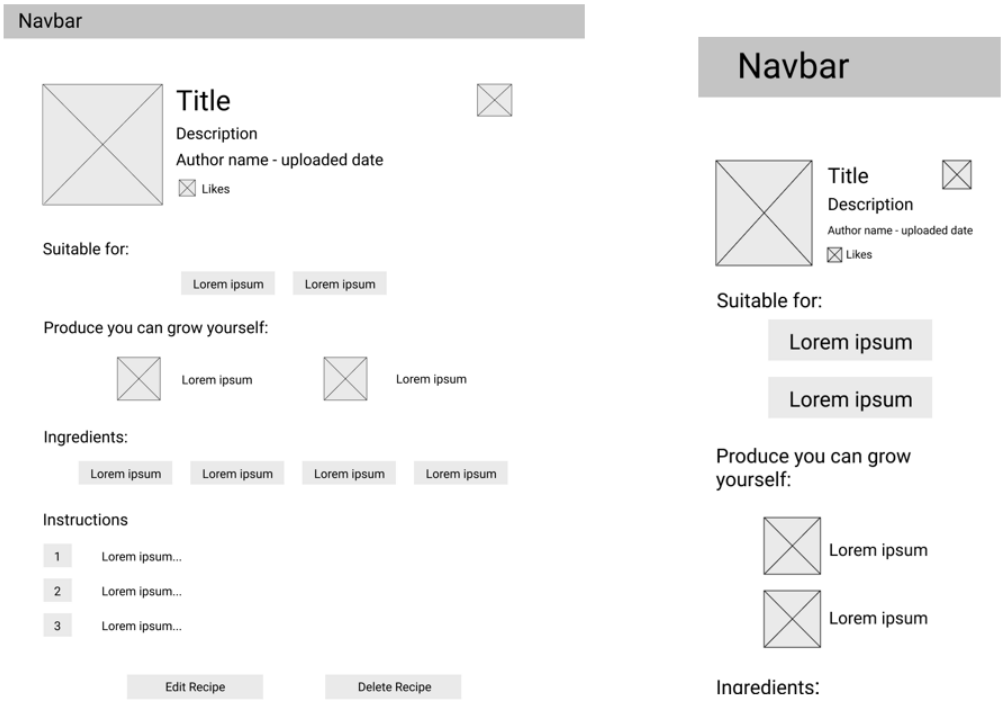


Figure B.2.19: Wireframes for desktop and mobile views of the recipe page for a single recipe to be included in the application. The edit button at the bottom only appears if the user is the author and the delete button is only present if the user is the author or has administrator permissions.

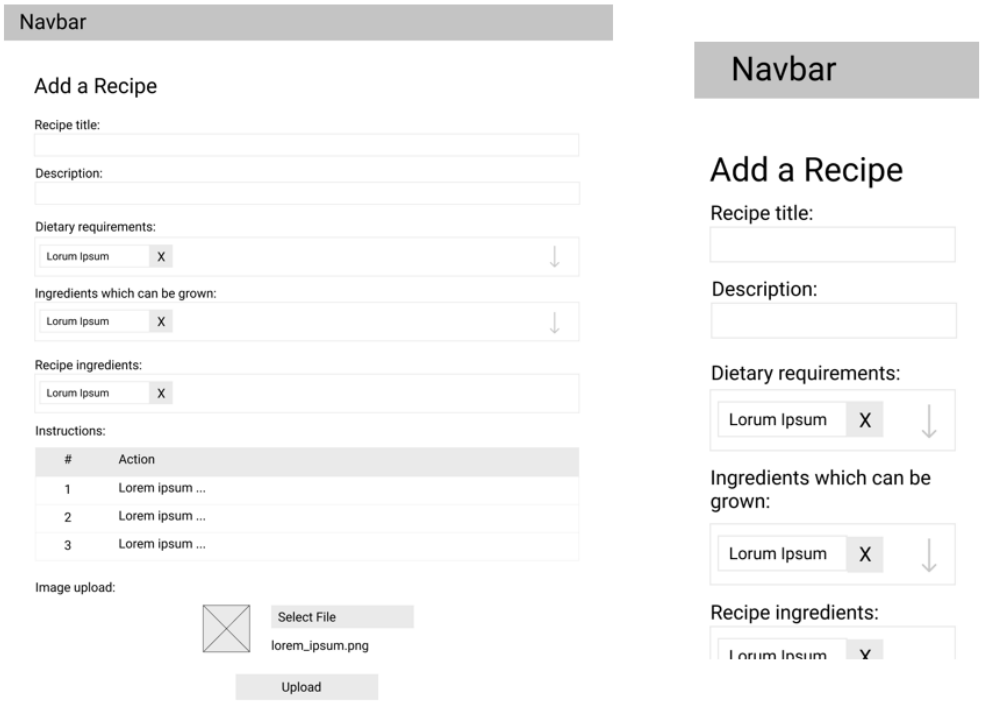


Figure B.2.20: Wireframes for desktop and mobile views of the upload recipe page to be included in the application.

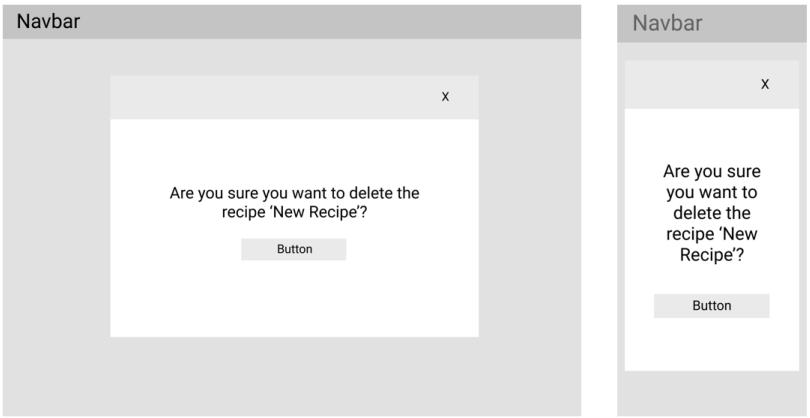


Figure B.2.21: Wireframes for desktop and mobile views of the delete recipe modal to be included in the application. The modal can be accessed from the Planting Profile and the single recipe page.

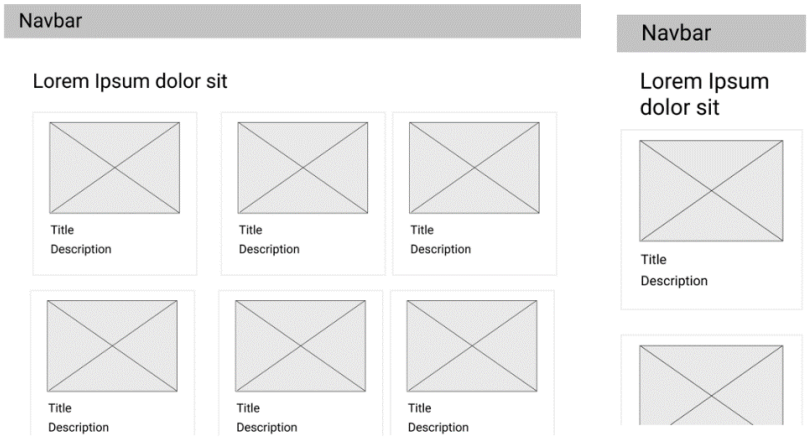


Figure B.2.22: Wireframes for desktop and mobile views of the articles page to be included in the application.

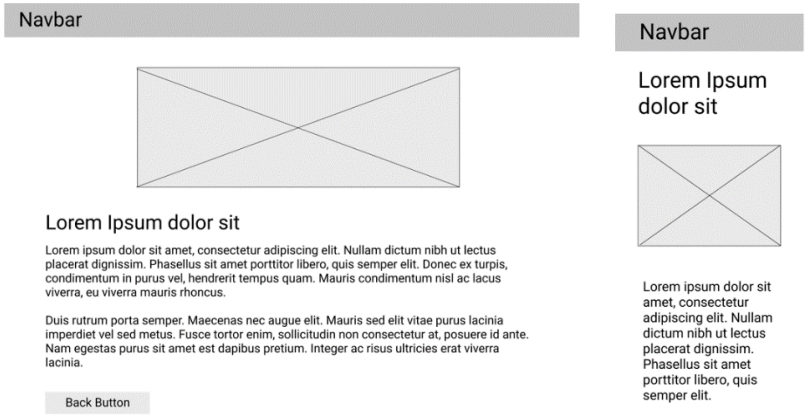


Figure B.2.23: Wireframes for desktop and mobile views of the article page for a single article to be included in the application.

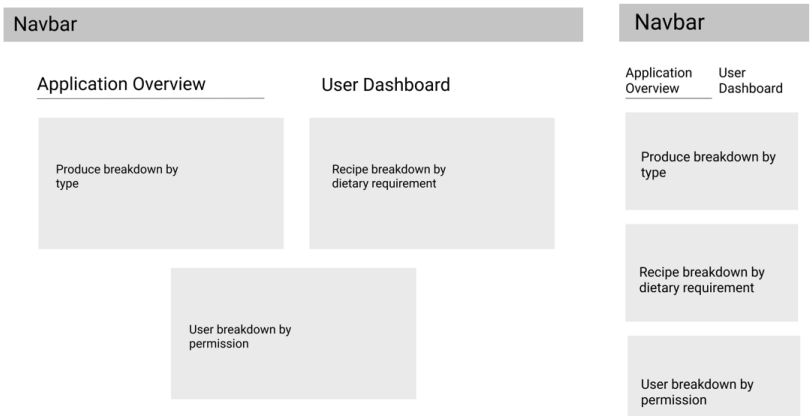


Figure B.2.24: Wireframes for desktop and mobile views of the application overview within the admin section to be included in the application.

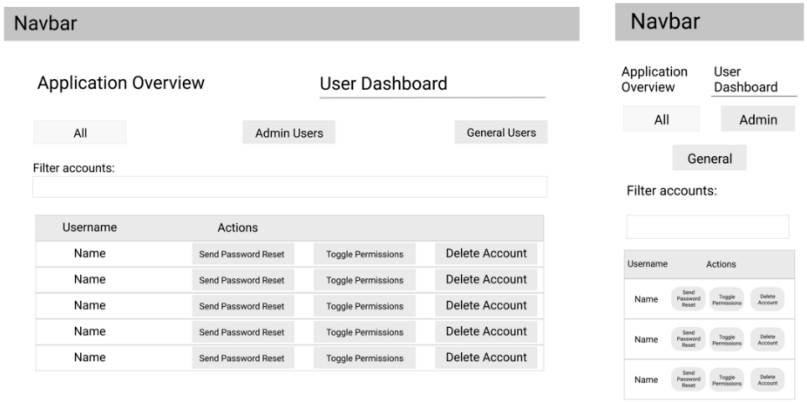


Figure B.2.25: Wireframes for desktop and mobile views of the user dashboard within the admin section to be included in the application.

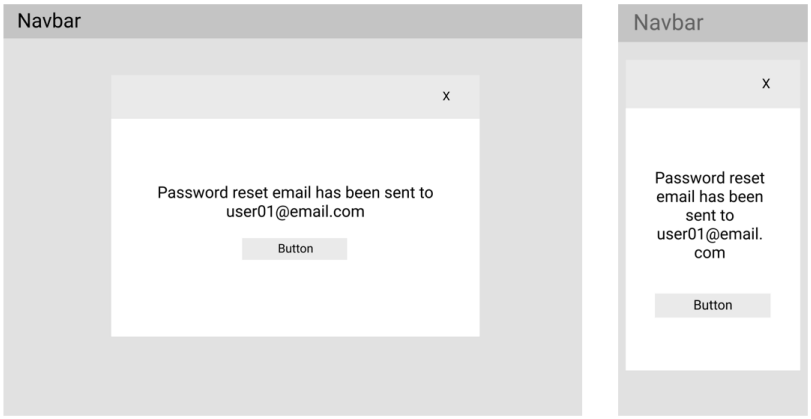


Figure B.2.26: Wireframes for desktop and mobile views of the admin user password reset modal to be included in the application.

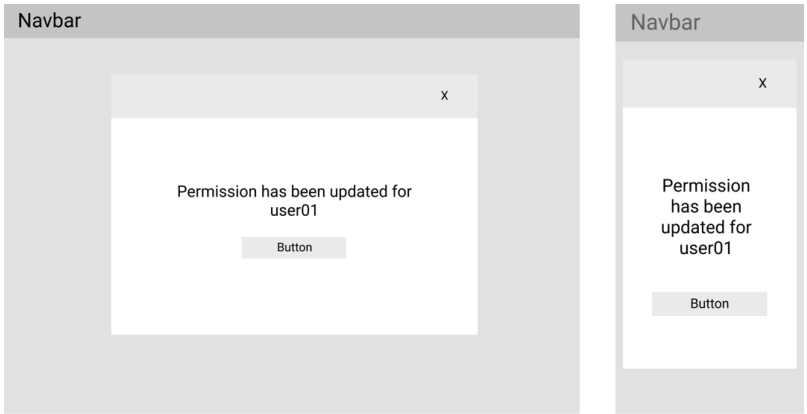


Figure B.2.27: Wireframes for desktop and mobile views of the admin toggle user permission modal to be included in the application.

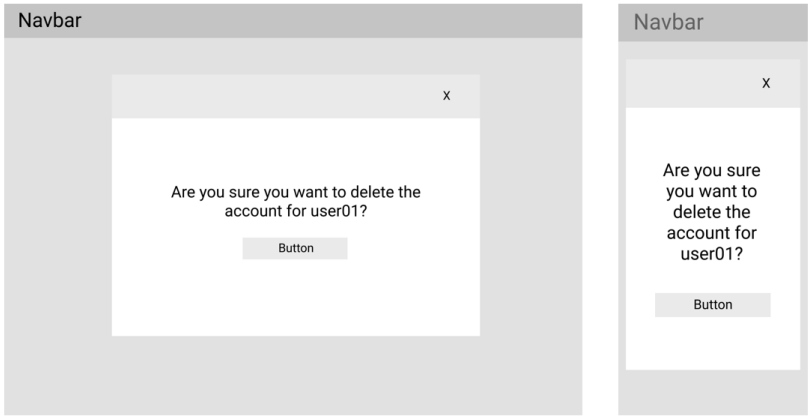


Figure B.2.28: Wireframes for desktop and mobile views of the admin delete user account modal to be included in the application.

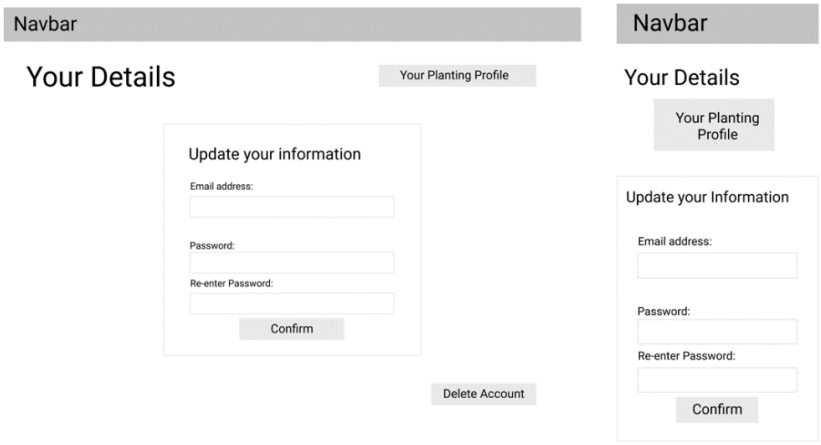


Figure B.2.29: Wireframes for desktop and mobile views of the user details container within the account section to be included in the application.

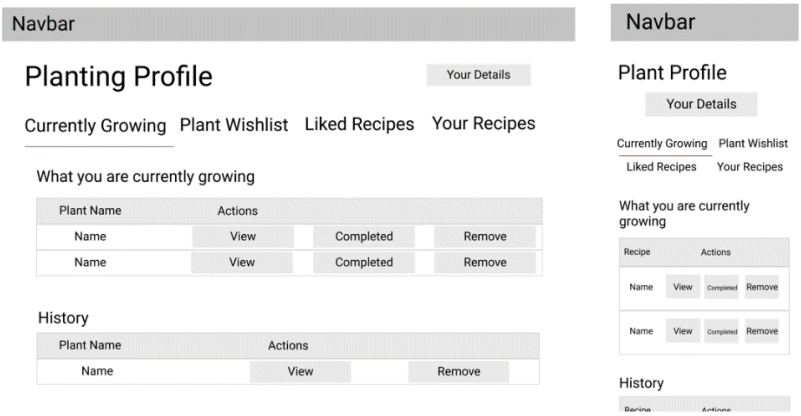


Figure B.2.30: Wireframes for desktop and mobile views of the currently growing list within the account section to be included in the application.

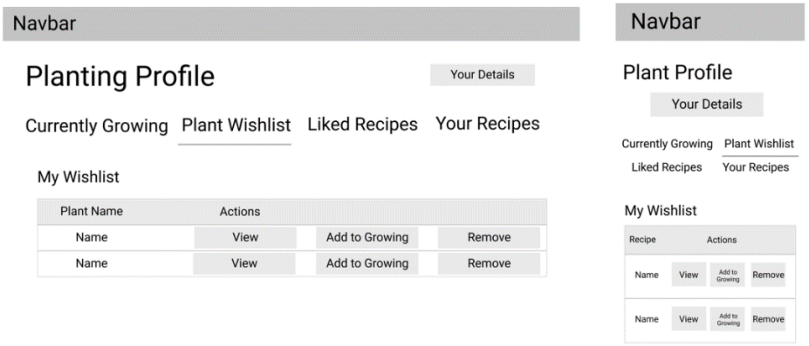


Figure B.2.31: Wireframes for desktop and mobile views of the plant wish-list within the account section to be included in the application.

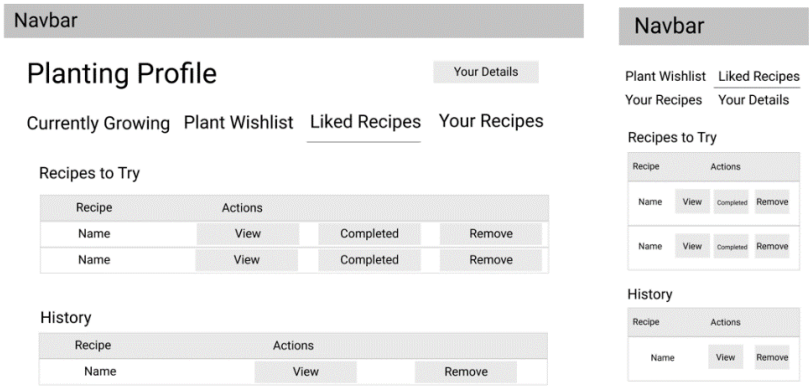


Figure B.2.32: Wireframes for desktop and mobile views of the liked recipes container within the account section to be included in the application.

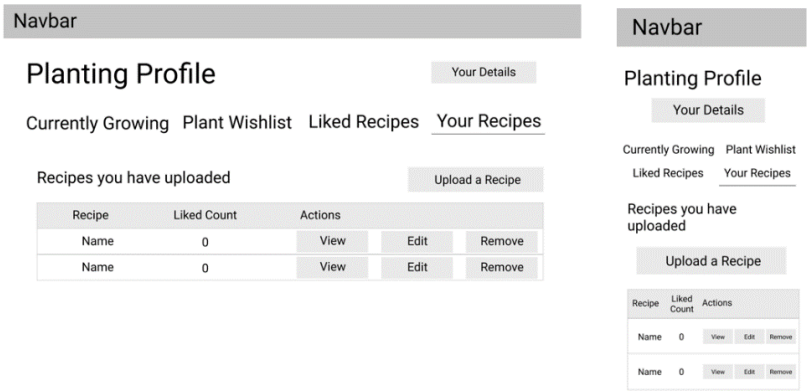


Figure B.2.33: Wireframes for desktop and mobile views of the uploaded recipes container within the account section to be included in the application.

B.3: Application Images

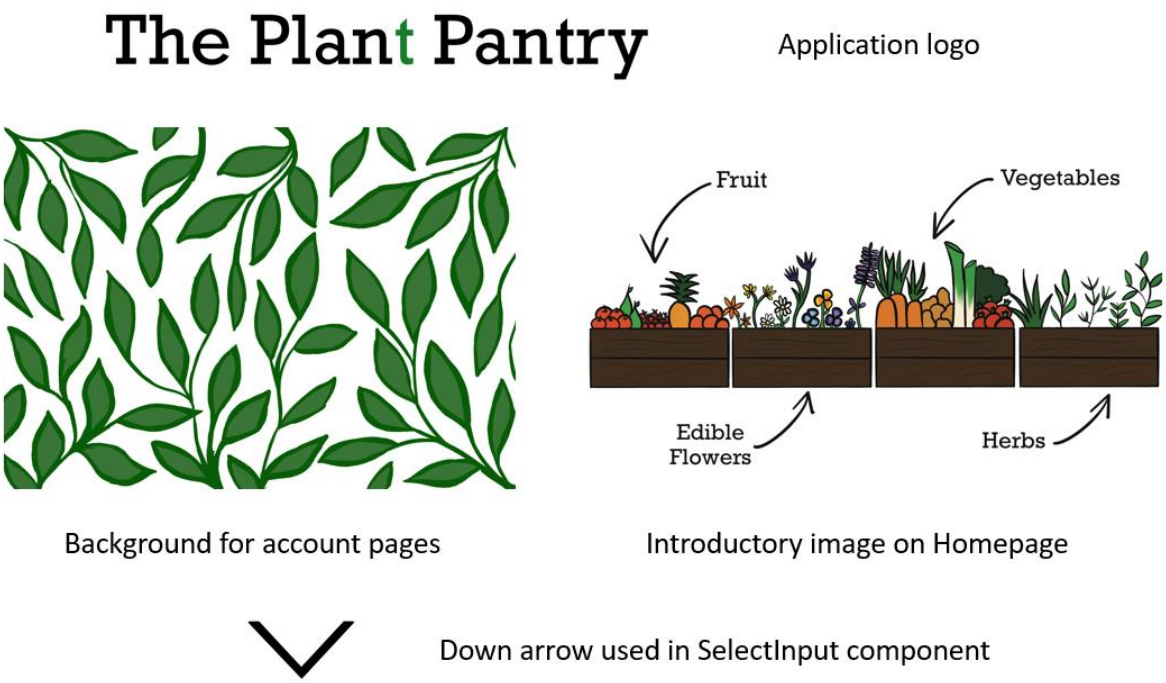


Figure B.3.1: High fidelity icons and images used within the application, created using Procreate. Includes the application icon included in the navigation bar, the background for account elements and an image for the homepage. The down arrow used in the SelectInput component is also present.

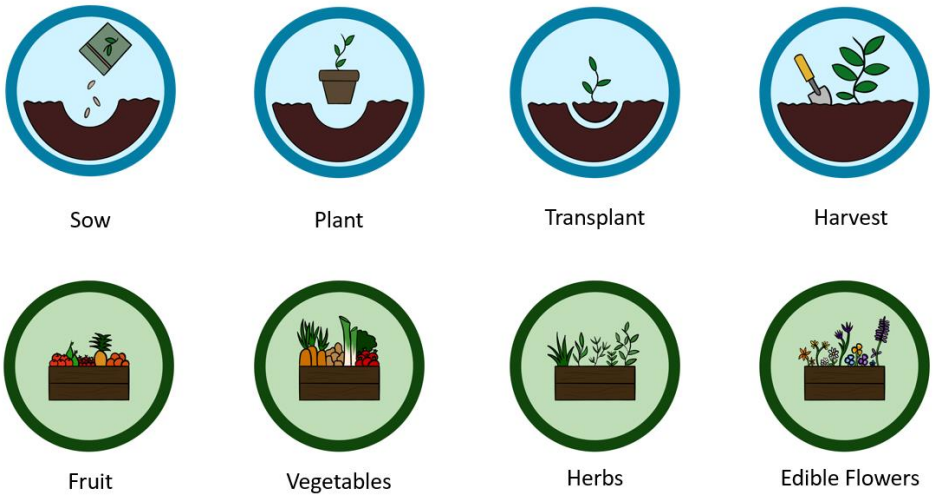


Figure B.3.2: High fidelity icons used throughout application features, created using Procreate. Includes icons for the planner and different types of produce.

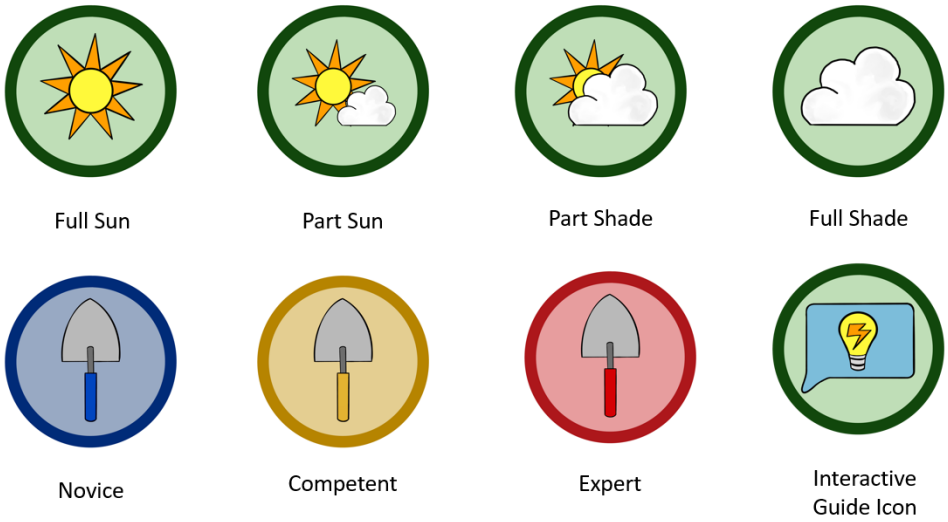


Figure B.3.3: High fidelity icons used throughout various application features, created using Procreate. Includes icons for varying light levels, varying competencies, and the icon for the interactive guide.

Figure B.3.4: High fidelity icons used throughout various application features, created using Procreate. Includes icons for different growing locations alongside the default recipe upload image. The icons representing item inclusion in currently growing lists, wish-lists, and liked recipes lists are also shown.

Figure B.3.5: High fidelity icons used within the SingleProduce component, created using Procreate. Includes icons for storage containers, freshness, time, storage locations, cooking, and freezing actions.

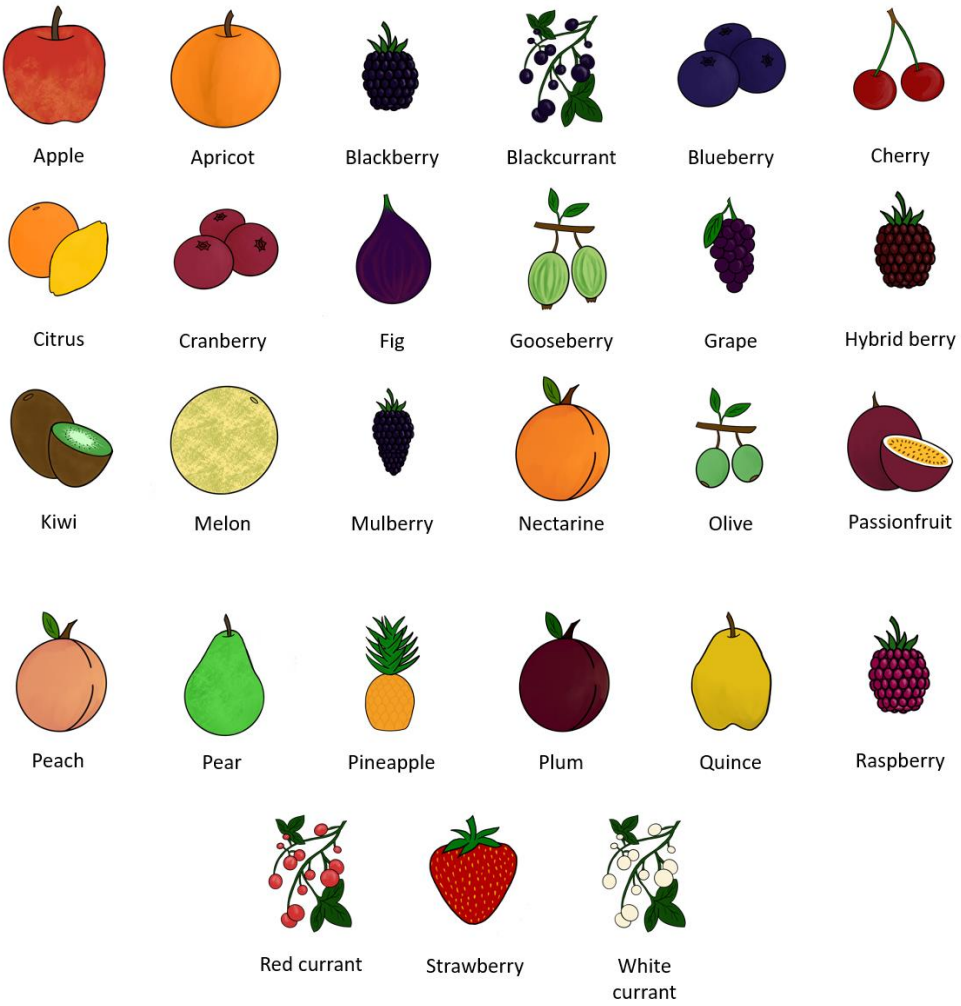


Figure B.3.6: High fidelity icons representing fruit produce items, created using Procreate. These icons are used in the interactive guide, produce homepage and single produce pages.

Figure B.3.7: High fidelity icons representing vegetable produce items, created using Procreate. These icons are used in the interactive guide, produce homepage and single produce pages.

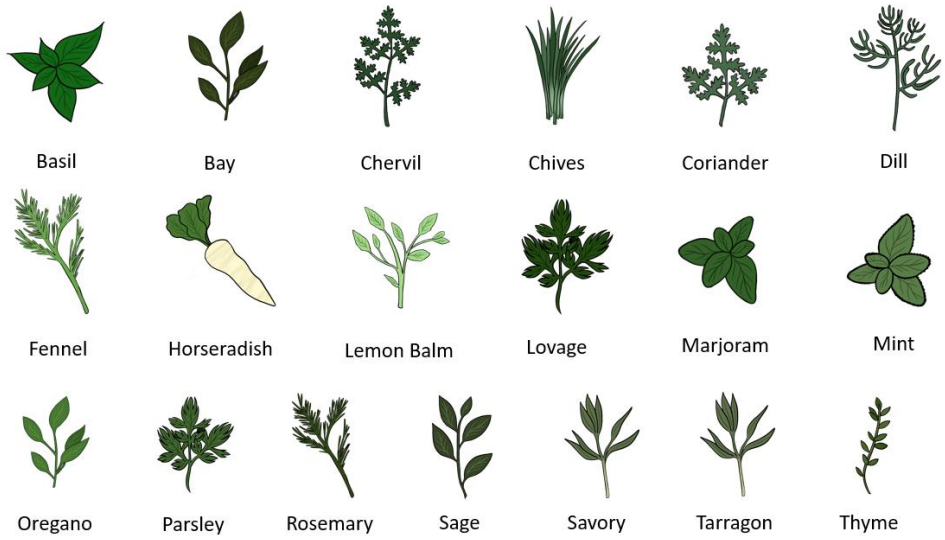


Figure B.3.8: High fidelity icons representing herb produce items, created using Procreate. These icons are used in the interactive guide, produce homepage and single produce pages.

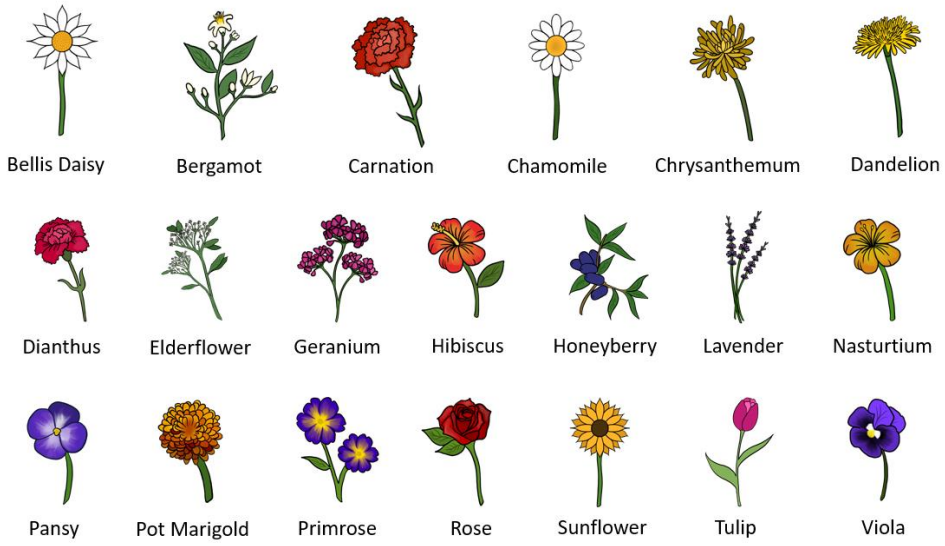


Figure B.3.9: High fidelity icons representing edible flower produce items, created using Procreate. These icons are used in the interactive guide, produce homepage and single produce pages.

B.4: Application Prototypes

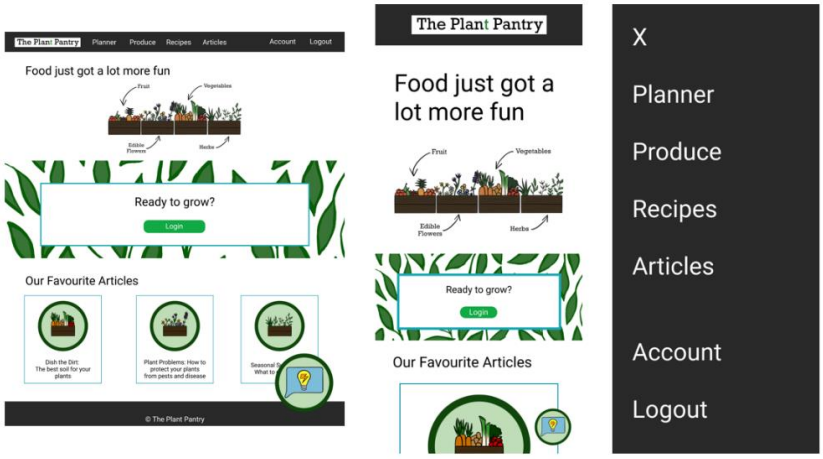


Figure B.4.1: Prototypes for desktop and mobile views of the homepage to be included in the application.

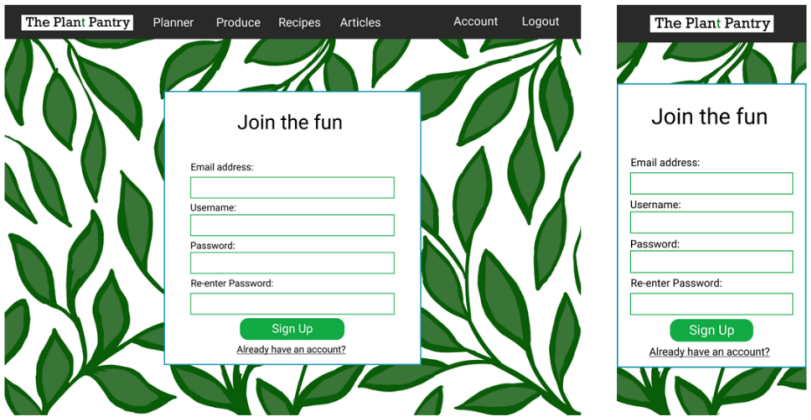


Figure B.4.2: Prototypes for desktop and mobile views of the sign-up page to be included in the application.

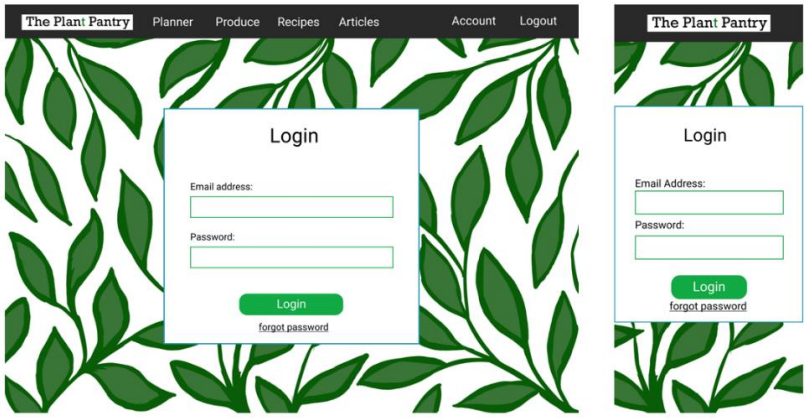


Figure B.4.3: Prototypes for desktop and mobile views of the login page to be included in the application.

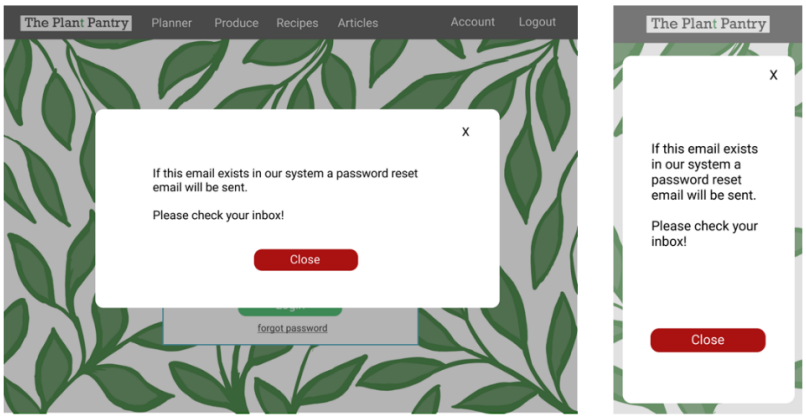


Figure B.4.4: Prototypes for desktop and mobile views of the forgot password modal to be included in the application.

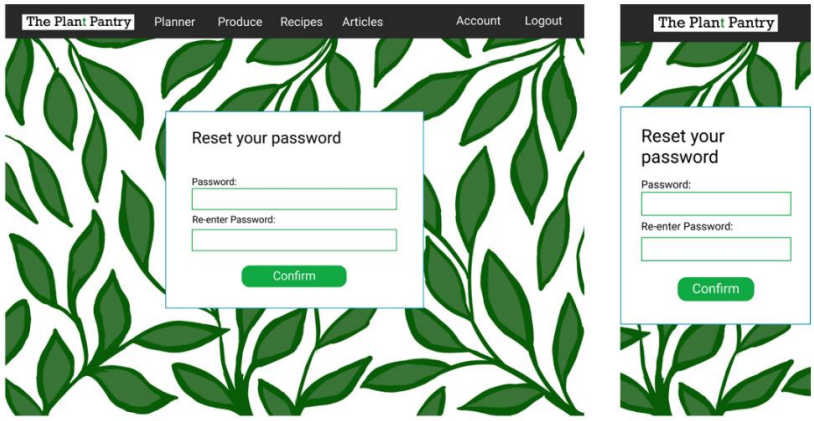


Figure B.4.5: Prototypes for desktop and mobile views of the reset password page to be included in the application.

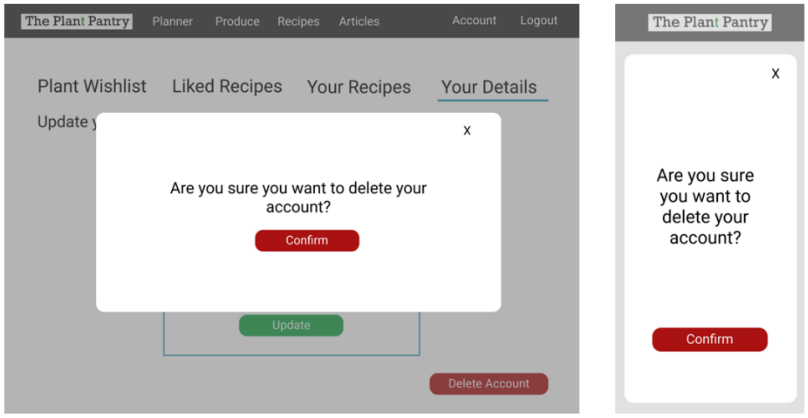


Figure B.4.6: Prototypes for desktop and mobile views of the delete account modal to be included in the application.

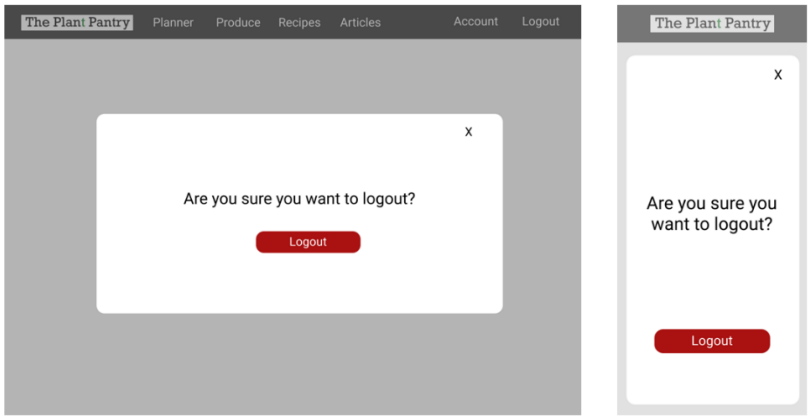


Figure B.4.7: Prototypes for desktop and mobile views of the logout modal to be included in the application.

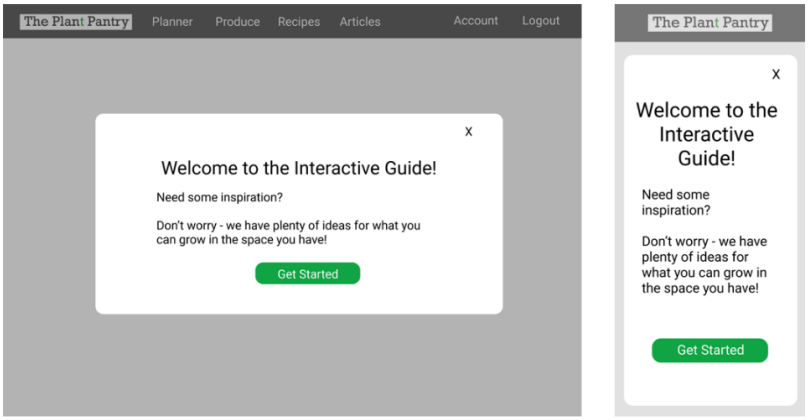


Figure B.4.8: Prototypes for desktop and mobile views of the first section of the interactive guide to be included in the application, introducing the user to the guide.

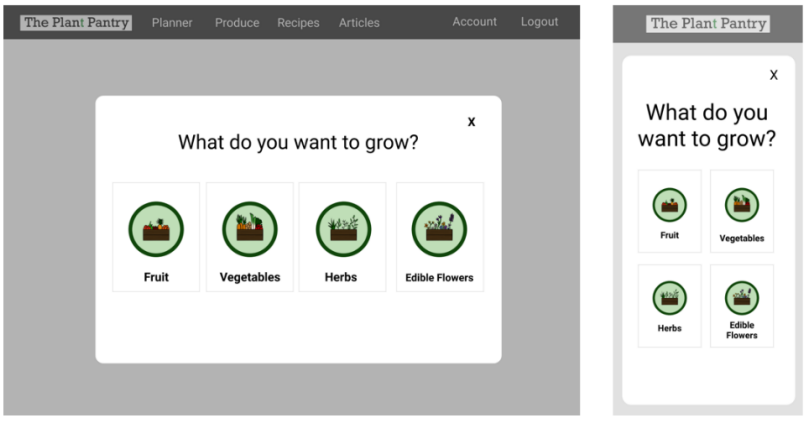


Figure B.4.9: Prototypes for desktop and mobile views of the second section of the interactive guide to be included in the application, asking the user what they want to grow.

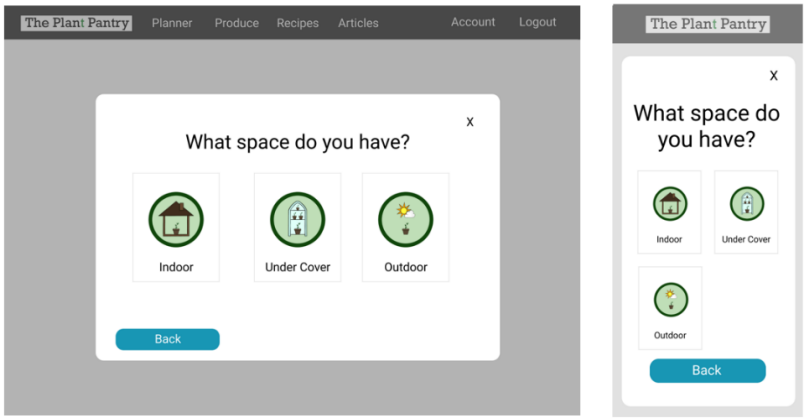


Figure B.4.10: Prototypes for desktop and mobile views of the third section of the interactive guide to be included in the application, asking the user what space they have.

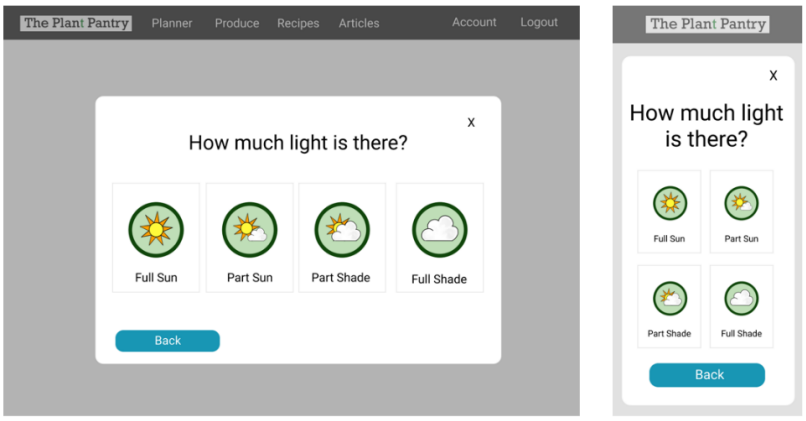


Figure B.4.11: Prototypes for desktop and mobile views of the fourth section of the interactive guide to be included in the application, asking the user how much light there is in this space.

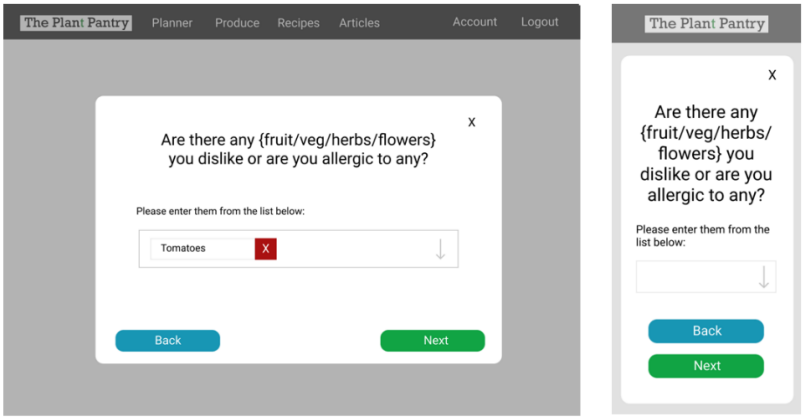


Figure B.4.12: Prototypes for desktop and mobile views of the fifth section of the interactive guide to be included in the application, asking the user if they have any allergies or if there is any produce they do not like.

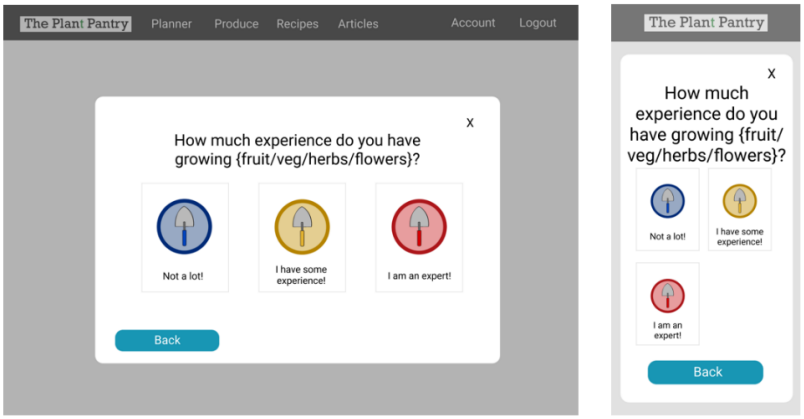


Figure B.4.13: Prototypes for desktop and mobile views of the sixth section of the interactive guide to be included in the application, asking the user about their gardening experience.

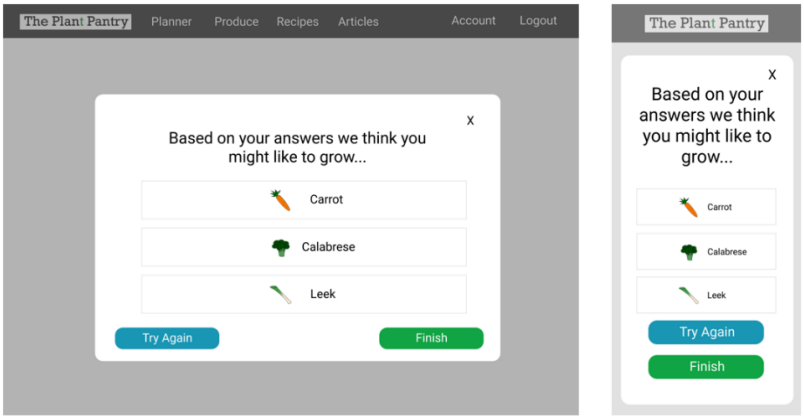


Figure B.4.14: Prototypes for desktop and mobile views of the seventh section of the interactive guide to be included in the application, asking the user what they want to grow.

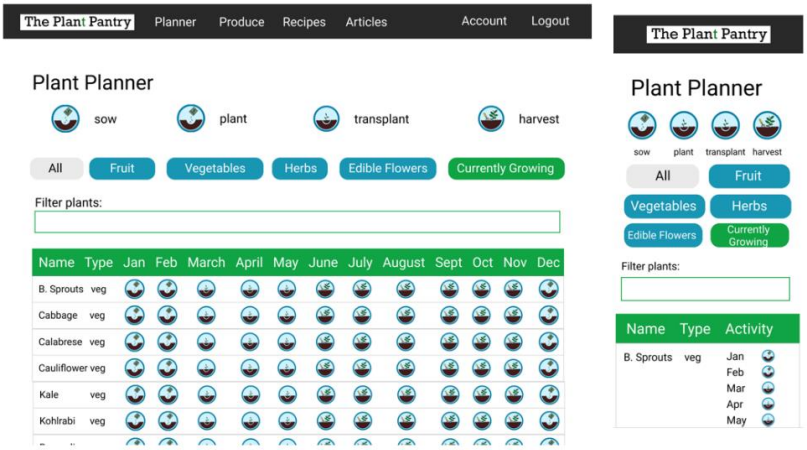


Figure B.4.15: Prototypes for desktop and mobile views of the planner to be included in the application, showing an overall view of the plant activities required for each plant per month.

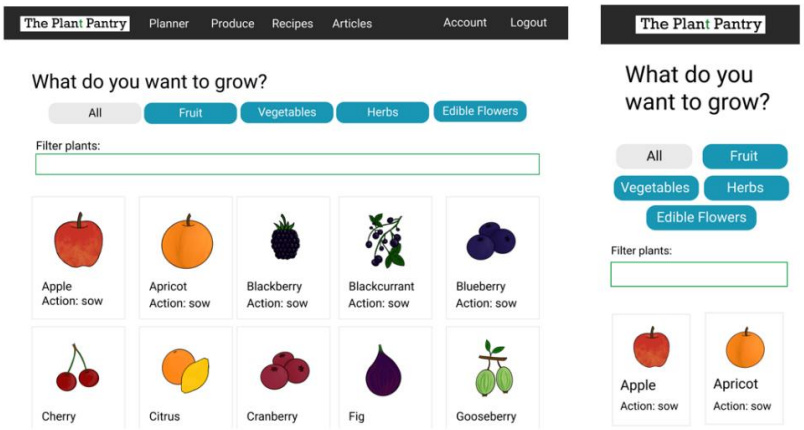


Figure B.4.16: Prototypes for desktop and mobile views of the produce page to be included in the application.

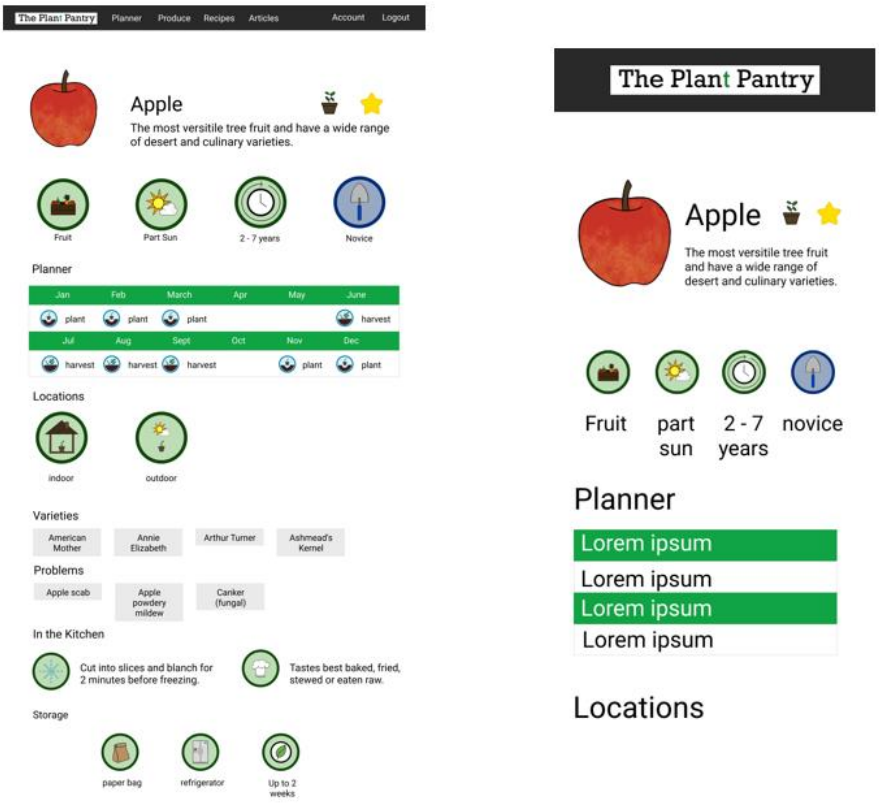


Figure B.4.17: Prototypes for desktop and mobile views of the produce page for a single produce item to be included in the application.

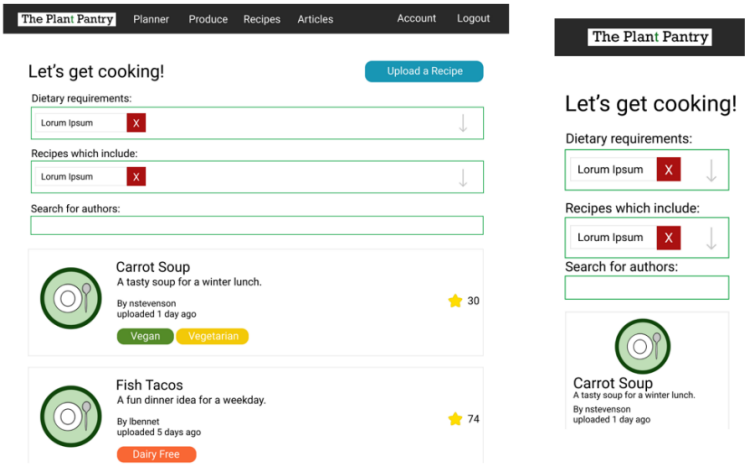


Figure B.4.18: Prototypes for desktop and mobile views of the recipes page to be included in the application.

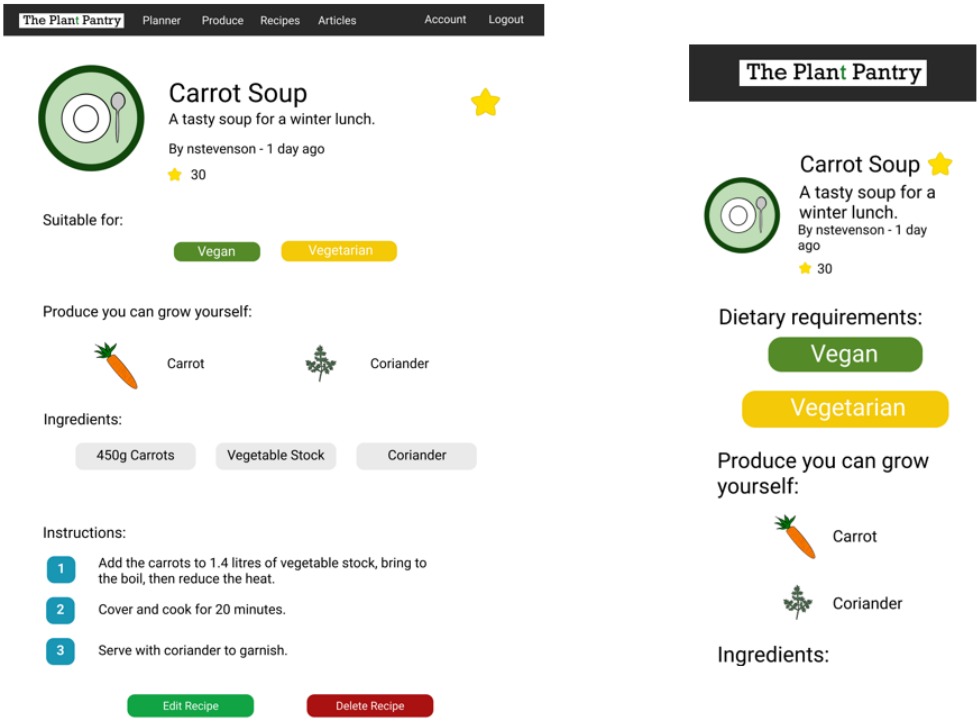


Figure B.4.19: Prototypes for desktop and mobile views of the recipe page for a single recipe to be included in the application. The edit button at the bottom only appears if the user is the author and the delete button is only present if the user is the author or has administrator permissions.

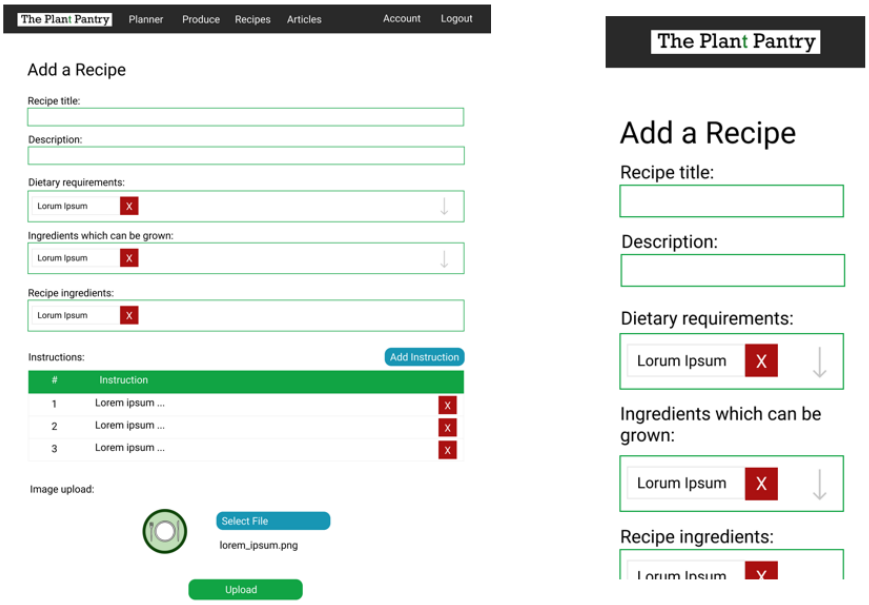


Figure B.4.20: Prototypes for desktop and mobile views of the upload recipe page to be included in the application.

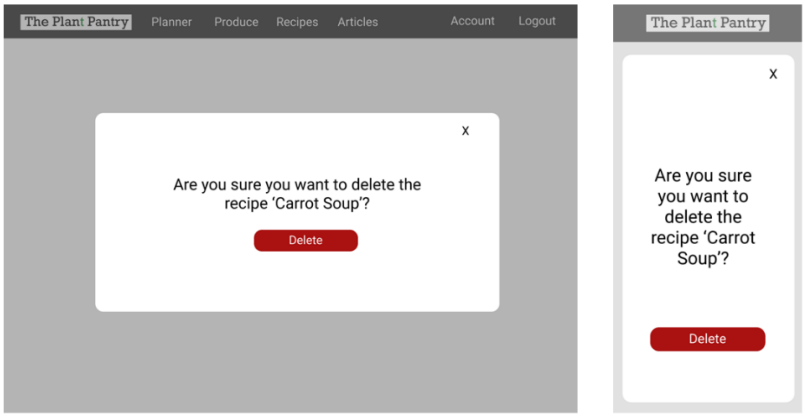


Figure B.4.21: Prototypes for desktop and mobile views of the delete recipe modal to be included in the application. The modal can be accessed from the Planting Profile and the single recipe page.

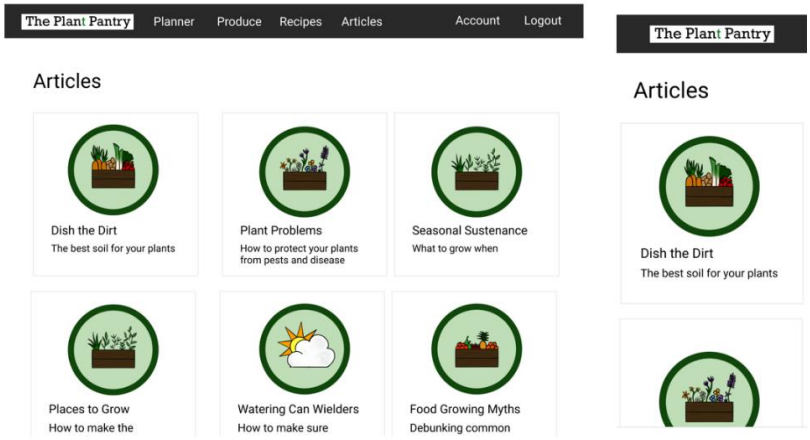


Figure B.4.22: Prototypes for desktop and mobile views of the articles page to be included in the application.

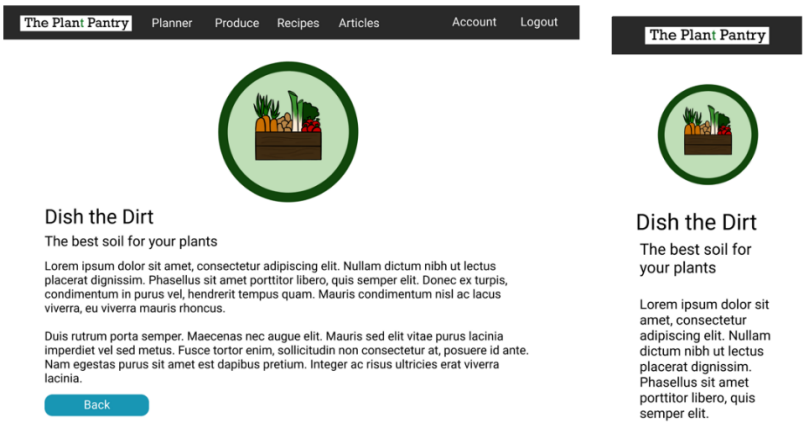


Figure B.4.23: Prototypes for desktop and mobile views of the article page for a single article to be included in the application.

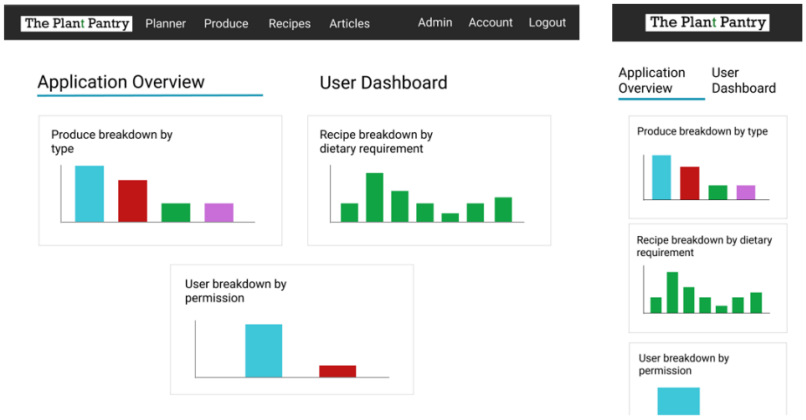


Figure B.4.24: Prototypes for desktop and mobile views of the application overview within the admin section to be included in the application.

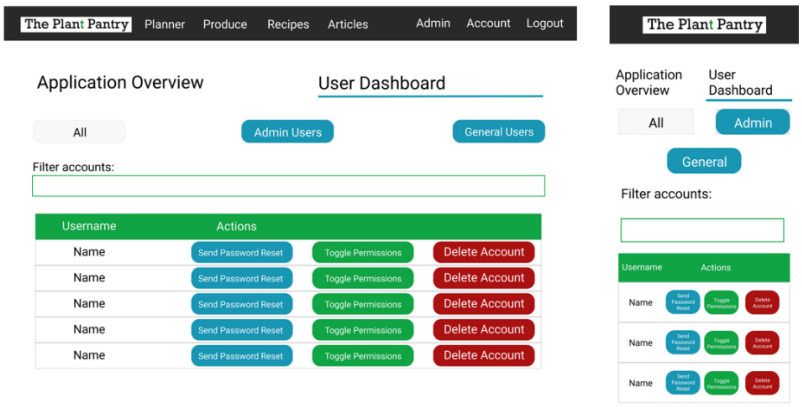


Figure B.4.25: Prototypes for desktop and mobile views of the user dashboard within the admin section to be included in the application.

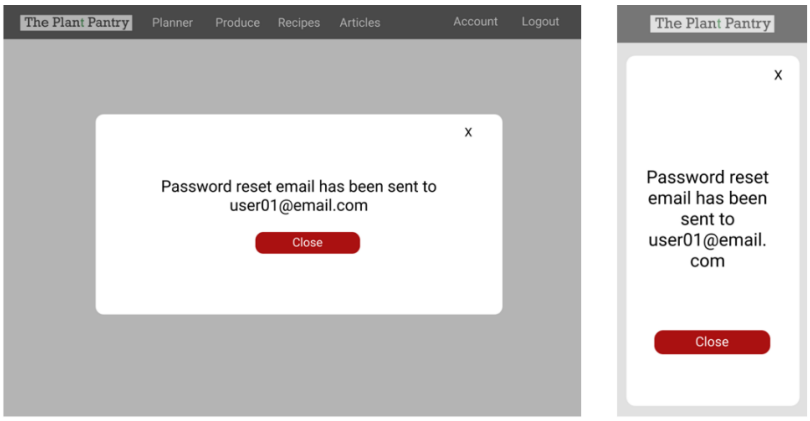


Figure B.4.26: Prototypes for desktop and mobile views of the admin user password reset modal to be included in the application.

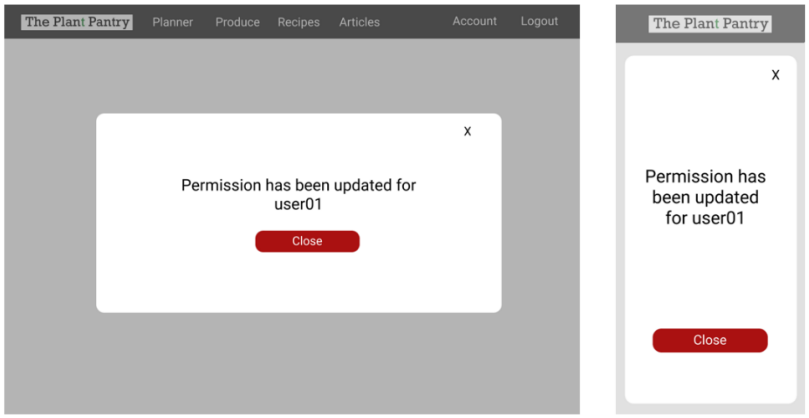


Figure B.4.27: Prototypes for desktop and mobile views of the admin toggle user permission modal to be included in the application.

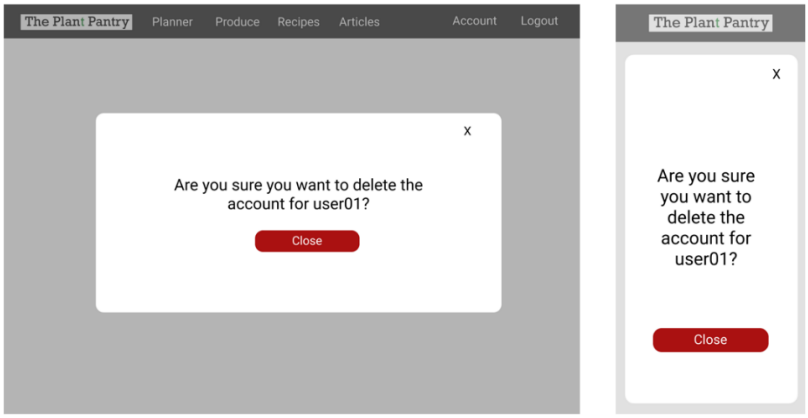


Figure B.4.28: Wireframes for desktop and mobile views of the admin delete user account modal to be included in the application.

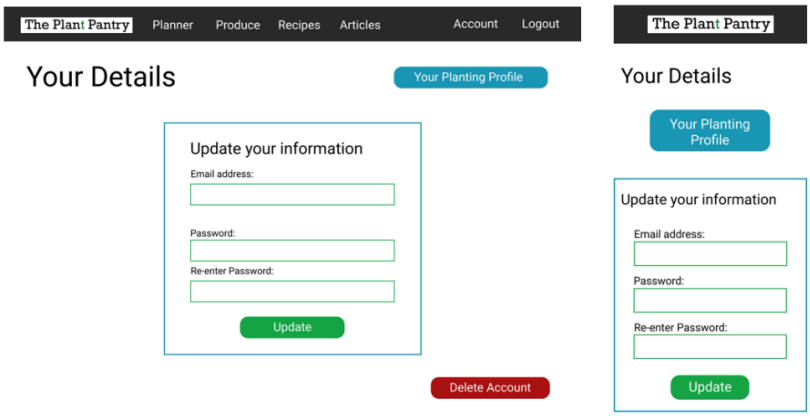


Figure B.4.29: Prototypes for desktop and mobile views of the user details container within the account section to be included in the application.

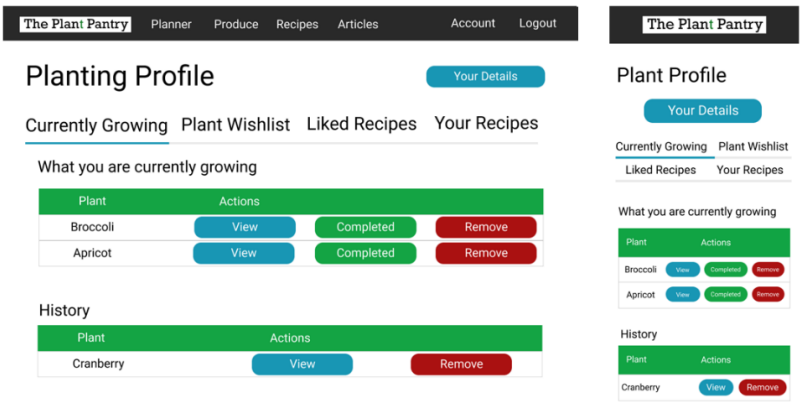


Figure B.4.30: Prototypes for desktop and mobile views of the currently growing list within the account section to be included in the application.

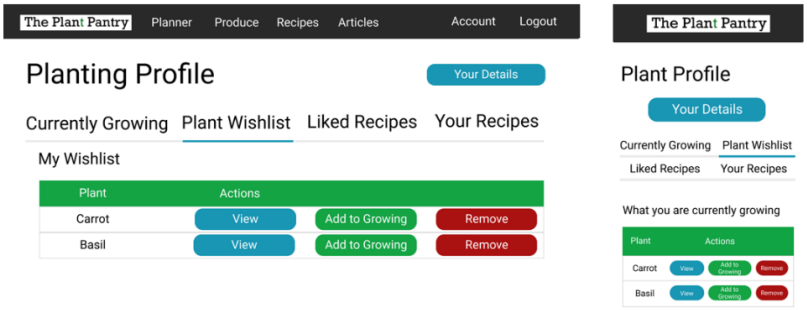


Figure B.4.31: Prototypes for desktop and mobile views of the plant wish-list within the account section to be included in the application.

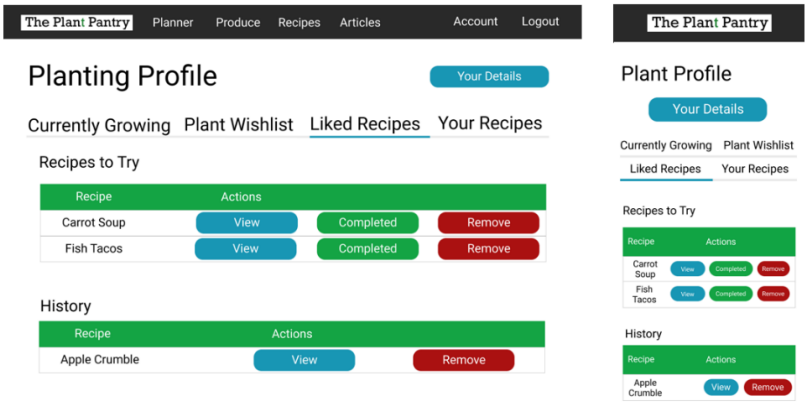


Figure B.4.32: Prototypes for desktop and mobile views of the liked recipes container within the account section to be included in the application.

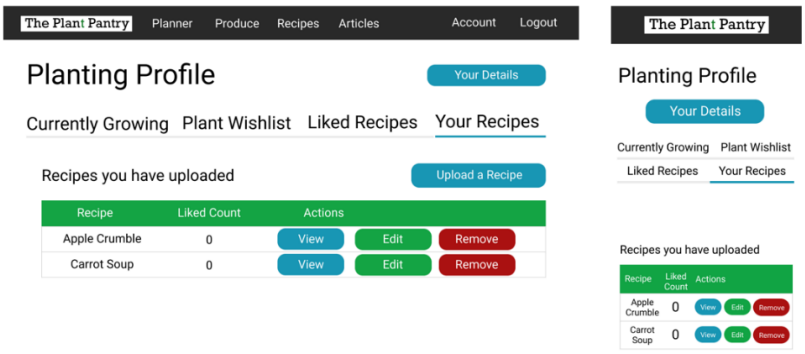


Figure B.4.33: Prototypes for desktop and mobile views of the uploaded recipes container within the account section to be included in the application.

Appendix C: Code Manifest

There are two elements fundamental to the application which have not been submitted: the node_modules folder and the .env file. The node_modules folder has not been included as it contains NPM packages which have been installed not created by the developer. Details on these packages are within the package.json and the package-lock.json files.

The .env file contains three environment variables, the database connection string, the JWT secret string, and the password to the application email account. The database connection string contains account information necessary for connecting to MongoDB, the JWT secret is used to manage user tokens, and the password is used to connect to the Gmail account used for sending forgotten password links. The two strings must be stored securely and therefore have not been included in the code submission, but the encrypted version of the file has been added and this is required to run the code.

Ref. #	File Name	Front-end / Back-end	Purpose	Created / Modified	Description
F001	src/client/components/ AccountContainer/ AccountContainer.react.js	Front-end	To provide the page formatting for the login and sign-up components.	Created	A React arrow functional component creating the appearance of the login and sign-up components. Props can be passed to the component, allowing it to be reused across the project.
F002	src/client/components/ AccountContainer/ AccountContainer.test.js	Front-end	To unit test the AccountContainer component.	Created	A test file using Jest and Enzyme to test the component renders correctly.
F003	src/client/components/ Button/Button.react.js	Front-end	To create a reusable button component.	Created	A React arrow functional component creating a button with appropriate and consistent styles. Props can be passed to the component, allowing it to be reused across the project.
F004	src/client/components/ Button/Button.test.js	Front-end	To unit test the Button component.	Created	A test file using Jest and Enzyme to test the component renders correctly.
F005	src/client/components/ Card/Card.react.js	Front-end	To create a reusable card component.	Created	Two React arrow functional components creating a card and card container with appropriate and consistent styles. Props can be passed to the component, allowing it to be reused across the project.

F006	src/client/components/ Card/Card.test.js	Front-end	To unit test the Card component.	Created	A test file using Jest and Enzyme to test the component renders correctly.
F007	src/client/components/containers/ Account/ForgotPassword/ ForgotPassword.react.js	Front-end	To create a component for end user usage, allowing a user to change their password if forgotten and a request has been made to do so.	Created	A React class component which can be accessed by a route allocated in the App.js file. Makes use of the AccountContainer, Button and TextInput components. Enables interaction with actions for checking the request id matches one in the 'password_resets' MongoDB collection and to delete the request and update the password upon successful user input.
F008	src/client/components/containers/ Account/ForgotPassword/ ForgotPassword.test.js	Front-end	To unit test the ForgotPassword component.	Created	A test file using Jest and Enzyme to test the component renders correctly, all imported components render properly, and all functions work as intended.
F009	src/client/components/containers/ Account/Login/Login.react.js	Front-end	To create a login component for end user usage, enabling application access.	Created	A React class component which can be accessed by a route allocated in the App.js file. Makes use of the AccountContainer, Button, TextInput and Modal components. Enables interaction with actions for checking information provided by the user matches existing data held in the 'users' MongoDB collection.
F010	src/client/components/containers/ Account/Login/Login.test.js	Front-end	To unit test the Login component.	Created	A test file using Jest and Enzyme to test the component renders correctly, all imported components render properly, and all functions work as intended.
F011	src/client/components/containers/ Account/Profile/Profile.react.js	Front-end	To create a profile component for end user usage, for viewing personal contributions and lists they have on the application.	Created	A React class component which can be accessed by a route allocated in the App.js file. Makes use of the CurrentlyGrowing, Wishlist, LikedRecipes, YourRecipes and Button components.
F012	src/client/components/containers/ Account/Profile/Profile.test.js	Front-end	To unit test the Profile component.	Created	A test file using Jest and Enzyme to test the component renders correctly, all imported

					components render properly, and all functions work as intended.
F013	src/client/components/containers/ Account/Profile/CurrentlyGrowing/ CurrentlyGrowing.react.js	Front-end	To create a component for end user usage, a subsection within the Profile component, for viewing and updating their currently growing list.	Created	A React class component which can be accessed through the Profile component. Makes use of the Table, Button, ButtonContainer and EmptyListMessage components. Enables interaction with actions for checking user information, retrieving the user's currently growing list, and updating the user's currently growing list in the MongoDB 'users' collection.
F014	src/client/components/containers/ Account/Profile/CurrentlyGrowing/ CurrentlyGrowing.test.js	Front-end	To unit test the CurrentlyGrowing component.	Created	A test file using Jest and Enzyme to test the component renders correctly, all imported components render properly, and all functions work as intended.
F015	src/client/components/containers/ Account/Profile/LikedRecipes/ LikedRecipes.react.js	Front-end	To create a component for end user usage, a subsection within the Profile component, for viewing and updating their liked recipes list.	Created	A React class component which can be accessed through the Profile component. Makes use of the Table, Button, ButtonContainer and EmptyListMessage components. Enables interaction with actions for retrieving the user's liked recipes list and updating the user's liked recipes list in the MongoDB 'users' collection.
F016	src/client/components/containers/ Account/Profile/LikedRecipes/ LikedRecipes.test.js	Front-end	To unit test the LikedRecipes component.	Created	A test file using Jest and Enzyme to test the component renders correctly, all imported components render properly, and all functions work as intended.
F017	src/client/components/containers/ Account/Profile/Wishlist/ Wishlist.react.js	Front-end	To create a component for end user usage, a subsection within the Profile component, for viewing and updating their produce wish-list.	Created	A React class component which can be accessed through the Profile component. Makes use of the Table, Button, ButtonContainer and EmptyListMessage components. Enables interaction with actions for retrieving the user's wishlist and updating the user's wishlist in the MongoDB users collection.

F018	src/client/components/containers/ Account/Profile/Wishlist/ Wishlist.test.js	Front-end	To unit test the Wishlist component.	Created	A test file using Jest and Enzyme to test the component renders correctly, all imported components render properly, and all functions work as intended.
F019	src/client/components/containers/ Account/Profile/YourRecipes/ YourRecipes.react.js	Front-end	To create a component for end user usage, a subsection within the Profile component, for viewing and updating the recipes they have uploaded.	Created	A React class component which can be accessed through the Profile component. Makes use of the Table, Button, ButtonContainer, EmptyListMessage, and DeleteRecipeModal components. Enables interaction with actions for retrieving the recipes uploaded by the user and deleting a specific recipe created by the specific user in the MongoDB 'users' and 'recipes' collections.
F020	src/client/components/containers/ Account/Profile/YourRecipes/ YourRecipes.test.js	Front-end	To unit test the YourRecipes component.	Created	A test file using Jest and Enzyme to test the component renders correctly, all imported components render properly, and all functions work as intended.
F023	src/client/components/containers/ Account/SignUp/SignUp.react.js	Front-end	To create a sign-up component for end user usage, required for enabling application access through the Login component.	Created	A React class component which can be accessed by a route allocated in the App.js file. Makes use of the AccountContainer, Button and TextInput components. Enables interaction with actions for adding new user to the 'users' MongoDB collection.
F024	src/client/components/containers/ Account/SignUp/SignUp.test.js	Front-end	To unit test the SignUp component.	Created	A test file using Jest and Enzyme to test the component renders correctly, all imported components render properly, and all functions work as intended.
F025	src/client/components/containers/ Account/UpdateAccount/ UpdateAccount.react.js	Front-end	To create a component for end user usage which can be accessed after successful login. Allows the user to update their email and password, and to delete their account.	Created	A React class component which can be accessed by a route allocated in the App.js file. Makes use of the Button and TextInput components. Enables interaction with actions for getting information for a single account, updating an account, and deleting an account.

F026	src/client/components/containers/ Account/UpdateAccount/ UpdateAccount.test.js	Front-end	To unit test the UpdateAccount component.	Created	A test file using Jest and Enzyme to test the component renders correctly, all imported components render properly, and all functions work as intended.
F027	src/client/components/containers/ Admin/ApplicationOverview/ ApplicationOverview.react.js	Front-end	To create a component for admin user usage, a subsection within the Admin component, for viewing an overview of the information held in the application.	Created	A React class component which can be accessed through the Admin component. Makes use of React Chart.js and Chart.js npm modules. Enables interaction with actions for getting produce, recipe and user statistic information.
F028	src/client/components/containers/ Admin/ApplicationOverview/ ApplicationOverview.test.js	Front-end	To unit test the ApplicationOverview component.	Created	A test file using Jest and Enzyme to test the component renders correctly, all imported components render properly, and all functions work as intended.
F029	src/client/components/containers/ Admin/UserDashboard/ UserDashboard.react.js	Front-end	To create a component for admin user usage, a subsection within the Admin component, for viewing and updating user information held in the application.	Created	A React class component which can be accessed by a route allocated in the App.js file. Makes use of the Button, ButtonContainer, Table, TextInput, AdminSendUserPasswordReset, AdminToggleUserPermission and AdminDeleteUserAccount components. Enables interaction with actions for getting information for getting a list of all users, creating a password reset request, and updating a user's admin permission.
F030	src/client/components/containers/ Admin/UserDashboard/ UserDashboard.test.js	Front-end	To unit test the UserDashboard component.	Created	A test file using Jest and Enzyme to test the component renders correctly, all imported components render properly, and all functions work as intended.
F031	src/client/components/containers/ Admin/Admin.react.js	Front-end	To create an admin component for end user usage, to allow admin users to view and update key application information.	Created	A React class component which can be accessed by a route allocated in the App.js file. Makes use of the ApplicationOverview and UserDashboard components.

F032	src/client/components/containers/ Admin/Admin.test.js	Front-end	To unit test the Admin component.	Created	A test file using Jest and Enzyme to test the component renders correctly and all functions work as intended.
F033	src/client/components/containers/ Articles/ArticlesHome/ ArticlesHome.react.js	Front-end	To create a component for end user usage which can be accessed after successful login. Allows the user to view a list of all existing articles within the application.	Created	A React class component which can be accessed by a route allocated in the App.js file. Makes use of the ArticleCard component and various images.
F034	src/client/components/containers/ Articles/ArticlesHome/ ArticlesHome.test.js	Front-end	To unit test the ArticlesHome component.	Created	A test file using Jest and Enzyme to test the component renders correctly.
F035	src/client/components/containers/ Articles/SingleArticles/ ArticlesTemplate.react.js	Front-end	To create a reusable ArticlesTemplate component to be used across the application.	Created	A React compound component, multiple React components are attached to a default component exported from the file. It enables a consistent Article appearance throughout the application, and allows props to be passed to it to aid in reusability.
F036	src/client/components/containers/ Articles/SingleArticles/ ArticlesTemplate.test.js	Front-end	To unit test the ArticlesTemplate, Image, Title, Body, CardContainer, InformationCard, BackButton	Created	A test file using Jest and Enzyme to test the components render correctly.
F037	src/client/components/containers/ Articles/SingleArticles/ DishTheDirt.react.js	Front-end	To allow users to view an article called ‘Dish the Dirt’, after successful login.	Created	A React functional arrow component which can be accessed by a route allocated in the App.js file. Makes use of an image and information shown is hard-coded.
F038	src/client/components/containers/ Articles/SingleArticles/ DishTheDirt.test.js	Front-end	To unit test the DishTheDirt component.	Created	A test file using Jest and Enzyme to test the component renders correctly.
F039	src/client/components/containers/ Articles/SingleArticles/ PlantProblems.react.js	Front-end	To allow users to view an article called ‘Plant Problems’, after successful login.	Created	A React functional arrow component which can be accessed by a route allocated in the App.js file. Makes use of an image and information shown is hard-coded.
F040	src/client/components/containers/ Articles/SingleArticles/	Front-end	To unit test the PlantProblems component.	Created	A test file using Jest and Enzyme to test the component renders correctly.

	PlantProblems.test.js				
F041	src/client/components/containers/ Articles/SingleArticles/ SeasonalSustenance.react.js	Front-end	To allow users to view an article called 'Seasonal Sustenance', after successful login.	Created	A React functional arrow component which can be accessed by a route allocated in the App.js file. Makes use of an image and information shown is hard-coded.
F042	src/client/components/containers/ Articles/SingleArticles/ SeasonalSustenance.test.js	Front-end	To unit test the SeasonalSustenance component.	Created	A test file using Jest and Enzyme to test the component renders correctly.
F043	src/client/components/containers/ Homepage/Homepage.react.js	Front-end	To create a homepage component the user can view.	Created	A React class component which can be accessed by a route allocated in the App.js file. Makes use of the ArticleCard, Button and Footer components. Various images are used and there are links to other application components.
F044	src/client/components/containers/ Homepage/Homepage.test.js	Front-end	To unit test the Homepage component.	Created	A test file using Jest and Enzyme to test the component renders correctly and all imported components render properly.
F045	src/client/components/containers/ InteractiveGuide/InputOptions/ ImageCard.react.js	Front-end	To create an ImageCard component for use in the guide.	Created	A React arrow functional component creating a card containing an image with appropriate and consistent styles. Props can be passed to the component, allowing it to be reused across the project.
F046	src/client/components/containers/ InteractiveGuide/InputOptions/ ImageCard.test.js	Front-end	To unit test the ImageCard component.	Created	A test file using Jest and Enzyme to test the component renders correctly.
F047	src/client/components/containers/ InteractiveGuide/InputOptions/ PlainCard.react.js	Front-end	To create an PlainCard component for use in the guide.	Created	A React arrow functional component creating a card with appropriate and consistent styles. Props can be passed to the component, allowing it to be reused across the project.
F048	src/client/components/containers/ InteractiveGuide/InputOptions/ PlainCard.test.js	Front-end	To unit test the PlainCard component.	Created	A test file using Jest and Enzyme to test the component renders correctly.

F049	src/client/components/containers/ InteractiveGuide/Questions/ Compentency.react.js	Front-end	To create a component for the InteractiveGuide for the user to input gardening experience, with appropriate styles. This is the sixth section of the guide.	Created	A React class component creating a component for the user to enter information required for using the guide, with appropriate and consistent styles. Props can be passed to the component and makes use of the Modal, ImageCard and Button components. Images from the 'images/skillLevel' folder are used.
F050	src/client/components/containers/ InteractiveGuide/Questions/ Compentency.test.js	Front-end	To unit test the Competency component.	Created	A test file using Jest and Enzyme to test the component renders correctly and all imported components render properly.
F051	src/client/components/containers/ InteractiveGuide/Questions/ DislikesAllergies.react.js	Front-end	To create a component for the InteractiveGuide for the user to input dislikes and allergy information, with appropriate styles. This is the fifth section of the guide.	Created	A React class component creating a component for the user to enter information required for using the guide, with appropriate and consistent styles. Props can be passed to the component and makes use of the Modal, SelectInput and Button components.
F052	src/client/components/containers/ InteractiveGuide/Questions/ DislikesAllergies.test.js	Front-end	To unit test the DislikesAllergies component.	Created	A test file using Jest and Enzyme to test the component renders correctly, all imported components render properly, and all functions work as intended.
F053	src/client/components/containers/ InteractiveGuide/Questions/ GetStarted.react.js	Front-end	To create a welcoming the user to the InteractiveGuide component, with appropriate styles. This is the first section of the guide.	Created	A React class component creating a component welcoming the user to the guide, with appropriate and consistent styles. Props can be passed to the component and makes use of the Modal and Button components.
F054	src/client/components/containers/ InteractiveGuide/Questions/ GetStarted.test.js	Front-end	To unit test the GetStarted component.	Created	A test file using Jest and Enzyme to test the component renders correctly and all imported components render properly.
F055	src/client/components/containers/ InteractiveGuide/Questions/ GuideResults.react.js	Front-end	To create a component for the InteractiveGuide for the user to view results based on their	Created	A React class component creating a component for the user to view their tailored results after using the guide, with

			input, with appropriate styles. This is the last section of the guide.		appropriate and consistent styles. Props can be passed to the component and makes use of the Modal, PlainCard and Button components.
F056	src/client/components/containers/ InteractiveGuide/Questions/ GuideResults.test.js	Front-end	To unit test the GuideResults component.	Created	A test file using Jest and Enzyme to test the component renders correctly and all imported components render properly.
F057	src/client/components/containers/ InteractiveGuide/Questions/ LightConditions.react.js	Front-end	To create a component for the InteractiveGuide for the user to input the available light they have, with appropriate styles. This is the fourth section of the guide.	Created	A React class component creating a component for the user to enter information required for using the guide, with appropriate and consistent styles. Props can be passed to the component and makes use of the Modal, ImageCard and Button components. Images from the 'images/lightConditions' folder are used.
F058	src/client/components/containers/ InteractiveGuide/Questions/ LightConditions.test.js	Front-end	To unit test the LightConditions component.	Created	A test file using Jest and Enzyme to test the component renders correctly and all imported components render properly.
F059	src/client/components/containers/ InteractiveGuide/Questions/ ProduceType.react.js	Front-end	To create a component for the InteractiveGuide for the user to input the type of produce they wish to grow, with appropriate styles. This is the second section of the guide.	Created	A React class component creating a component for the user to enter information required for using the guide, with appropriate and consistent styles. Props can be passed to the component and makes use of the Modal, ImageCard and Button components. Images from the 'images/produce' folder are used.
F060	src/client/components/containers/ InteractiveGuide/Questions/ ProduceType.test.js	Front-end	To unit test the ProduceType component.	Created	A test file using Jest and Enzyme to test the component renders correctly and all imported components render properly.
F061	src/client/components/containers/ InteractiveGuide/Questions/ SpaceType.react.js	Front-end	To create a component for the InteractiveGuide for the user to input the available space	Created	A React class component creating a component for the user to enter information required for using the guide, with

			they have, with appropriate styles. This is the third section of the guide.		appropriate and consistent styles. Props can be passed to the component and makes use of the Modal, ImageCard and Button components. Images from the 'images/space' folder are used.
F062	src/client/components/containers/ InteractiveGuide/Questions/ SpaceType.test.js	Front-end	To unit test the SpaceType component.	Created	A test file using Jest and Enzyme to test the component renders correctly and all imported components render properly.
F063	src/client/components/containers/ InteractiveGuide/ InteractiveGuide.react.js	Front-end	To create a guide component to allow users to receive guidance on what to grow based on information they provide such as light conditions and expertise.	Created	A React class component creating a component for the interactive guide feature, with appropriate and consistent styles. It is accessed through the InteractiveGuideIcon component. It makes use of the Modal, Competency, DislikesAllergies, GetStarted, GuideResults, LightConditions, ProduceType and SpaceType components. Enables interaction with actions for getting information for a particular produce type and for getting tailored results from the 'produce' MongoDB collection.
F064	src/client/components/containers/ InteractiveGuide/ InteractiveGuide.test.js	Front-end	To unit test the InteractiveGuide component.	Created	A test file using Jest and Enzyme to test the component renders correctly, all imported components render properly, and all functions work as intended.
F065	src/client/components/containers/ InteractiveGuide/ InteractiveGuideIcon.react.js	Front-end	To create a component to allow users to open the InteractiveGuide from anywhere in the application once logged in.	Created	A React class component creating a component for the user to open the interactive guide, with appropriate styles. It makes use of the InteractiveGuide component and uses the guide icon from the images folder.
F066	src/client/components/containers/ InteractiveGuide/ InteractiveGuideIcon.test.js	Front-end	To unit test the InteractiveGuideIcon component.	Created	A test file using Jest and Enzyme to test the component renders correctly, all imported components render properly, and all functions work as intended.

F067	src/client/components/containers/ Planner/Planner.react.js	Front-end	To create a component to allow users to view a plant planner containing information on whether to sow, plant, transplant or harvest different types of produce for each month once logged in.	Created	A React class component creating a component for the planner feature, with appropriate styles. It can be accessed by a route allocated in the App.js file. It makes use of the Table component and uses images from the 'images/planner' folder. Enables interaction with action for getting information for all produce held in the 'produce' MongoDB collection.
F068	src/client/components/containers/ Planner/Planner.test.js	Front-end	To unit test the Planner component.	Created	A test file using Jest and Enzyme to test the component renders correctly, all imported components render properly, and all functions work as intended.
F069	src/client/components/containers/ Produce/ProduceHome/ ProduceHome.react.js	Front-end	To create a component allowing users to view a complete list of all produce they can grow, alongside an image, the name of the item and the current action required for caring for it.	Created	A React class component creating a component for the produce section, with appropriate styles. It can be accessed by a route allocated in the App.js file. It makes use of the Button, ButtonContainer, TextInput, Card and CardContainer components and uses images from the ProduceItems file. Enables interaction with actions for getting information for all produce and getting all produce information by type from the 'produce' MongoDB collection.
F070	src/client/components/containers/ Produce/ProduceHome/ ProduceHome.test.js	Front-end	To unit test the ProduceHome component.	Created	A test file using Jest and Enzyme to test the component renders correctly, all imported components render properly, and all functions work as intended.
F071	src/client/components/containers/ Produce/ProduceHome/ SingleProduce.react.js	Front-end	To create a component allowing users to view all information for a single produce item, alongside appropriate images, and styling.	Created	A React class component creating a component for the produce section, with appropriate styles. It can be accessed by a route allocated in the App.js file. It makes use of the IconContainer, IconSection and Table components and uses images from the 'images/produce/details' directory. Enables

					interaction with actions for getting account information, getting single produce information, retrieving the user currently growing list and wish-list, and updating the user's currently growing list and wish-list in the 'produce' and 'users' MongoDB collections.
F072	src/client/components/containers/ Produce/ProduceHome/ SingleProduce.test.js	Front-end	To unit test the SingleProduce component.	Created	A test file using Jest and Enzyme to test the component renders correctly, all imported components render properly, and all functions work as intended.
F073	src/client/components/containers/ Recipes/RecipeForm/ RecipeForm.react.js	Front-end	To create a component allowing users to upload and edit recipes on the application, alongside appropriate styling.	Created	A React class component creating a component for the produce section, with appropriate styles. It can be accessed by a route allocated in the App.js file. It makes use of the EmptyListMessage, Button, SelectInput, TextInput, MultiTextInput, and Table components. Enables interaction with actions for getting account information, getting produce information, getting the recipes list, finding a single recipe, creating a recipe, and updating a recipe in the 'users' and 'recipes' MongoDB collections.
F074	src/client/components/containers/ Recipes/RecipeForm/ RecipeForm.test.js	Front-end	To unit test the RecipeForm component.	Created	A test file using Jest and Enzyme to test the component renders correctly, all imported components render properly, and all functions work as intended.
F075	src/client/components/containers/ Recipes/RecipesHome/ RecipeCard/RecipeCard.react.js	Front-end	To create a reusable card used to contain brief information for a single recipe in the RecipeHome component.	Created	A React arrow functional component creating a card which can be added to other components. It enables a consistent RecipeCard appearance throughout the application, and allows props to be passed to it to aid in reusability.
F076	src/client/components/containers/ Recipes/RecipesHome/	Front-end	To unit test the RecipeForm component.	Created	A test file using Jest and Enzyme to test the component renders correctly.

	RecipeCard/RecipeCard.react.js				
F077	src/client/components/containers/ Recipes/RecipesHome/ RecipesHome.react.js	Front-end	To create a component allowing users to view a complete list of all recipes held in the application.	Created	A React class component creating a component for the produce section, with appropriate styles. It can be accessed by a route allocated in the App.js file. It makes use of the RecipeCard, Button, SelectInput and TextInput components and uses the plate image from the 'images/recipes' directory. It also uses the dietOptions object and formatDate function. Enables interaction with actions for getting produce information, a list of all the recipes and a tailored recipe list based on user input.
F078	src/client/components/containers/ Recipes/RecipesHome/ RecipesHome.test.js	Front-end	To unit test the RecipesHome component.	Created	A test file using Jest and Enzyme to test the component renders correctly, all imported components render properly, and all functions work as intended.
F079	src/client/components/containers/ Recipes/SingleRecipe/ SingleRecipe.react.js	Front-end	To create a component allowing users to view all information for a single recipe, alongside styling.	Created	A React class component creating a component for the produce section, with appropriate styles. It can be accessed by a route allocated in the App.js file. It makes use of the IconContainer, IconSection, Button, ButtonSection and DeleteRecipeModal components and uses images from the 'images' directory. Enables interaction with actions for getting account information, getting single recipe information, deleting a recipe, retrieving the user liked recipes list and updating the user's liked recipes list 'recipes' and 'users' MongoDB collections.
F080	src/client/components/containers/ Recipes/SingleRecipe / SingleRecipe.test.js	Front-end	To unit test the SingleRecipe component.	Created	A test file using Jest and Enzyme to test the component renders correctly, all imported components render properly, and all functions work as intended.

F081	src/client/components/containers/ Recipes/DietaryOptions.js	Front-end	To contain a list of dietary types accommodated in the recipe section.	Created	A JavaScript file containing a single variable, an array of objects. This performs as a list of nine dietary types which can be searched in the RecipesHome component and tagged in the RecipeForm component to aid in user interaction with the application to help them find information.
F082	src/client/components/containers/ Recipes/DietaryOptions.test.js	Front-end	To unit test the DietaryOptions object.	Created	A test file using Jest and Enzyme to test the object renders correctly.
F083	src/client/components/containers/ generalFunctions.js	Front-end	To contain generic functions used across multiple components within the application.	Created	A JavaScript file containing one function, formatDate, which is used across multiple components to format dates and return a readable string.
F084	src/client/components/containers/ generalFunctions.test.js	Front-end	To unit test the functions in the generalFunctions file.	Created	A test file using Jest and Enzyme to test the functions execute correctly.
F085	src/client/components/containers/ ProduceItems.js	Front-end	To act as a reference point for all produce images within the application to make it easier to dynamically import images into components.	Created	A React file containing variables which map produce images to the produce names they correspond with. All produce images are imported into this file, which is used across the ProduceHome and SingleProduce components. This simplifies the development and bundling process executed by Webpack.
F086	src/client/components/containers/ ProduceItems.test.js	Front-end	To unit test the objects in the ProduceItems file.	Created	A test file using Jest and Enzyme to test the objects render correctly.
F087	src/client/components/ EmptyListMessage/ EmptyListMessage.react.js	Front-end	To create a reusable EmptyListMessage component to be used across the application.	Created	A React compound component, multiple React components are attached to a default component exported from the file. It enables a consistent EmptyListMessage appearance throughout the application, and allows props to be passed to it to aid in reusability.
F088	src/client/components/ EmptyListMessage/ EmptyListMessage.test.js	Front-end	To unit test the EmptyListMessage, Text and LinkText components.	Created	A test file using Jest and Enzyme to test the components render correctly.

F089	src/client/components/Footer/ Footer.react.js	Front-end	To create a footer which can be seen at the bottom of the Homepage component.	Created	A React arrow functional component creating a footer which can be added to other components.
F090	src/client/components/Footer/ Footer.test.js	Front-end	To unit test the Footer component.	Created	A test file using Jest and Enzyme to test the component renders correctly.
F091	src/client/components/ IconContainer/ IconContainer.react.js	Front-end	To create a reusable IconContainer component to be used across the application.	Created	A React compound component, multiple React components are attached to a default component exported from the file. It enables a consistent IconContainer appearance throughout the application, and allows props to be passed to it to aid in reusability.
F092	src/client/components/ IconContainer/ IconContainer.test.js	Front-end	To unit test the IconContainer and IconSection components.	Created	A test file using Jest and Enzyme to test the components render correctly.
F093	src/client/components/Input/ MultiTag/MultiTag.react.js	Front-end	To create a component to allow users to view a single input value for an input field for array values. The user can remove the entry from the array from this component.	Created	A React arrow functional component creating a container for displaying a single value and a button for deletion, used in the SelectInput component. Props can be passed to the component.
F094	src/client/components/Input/ MultiTag/MultiTag.test.js	Front-end	To unit test the MultiTag component.	Created	A test file using Jest and Enzyme to test the component renders correctly and all functions work as intended.
F095	src/client/components/Input/ MultiTextInput/ MultiTextInput.react.js	Front-end	To create a reusable component to allow users to input multiple values into a text input field, with appropriate styles.	Created	A React functional component creating a component allowing for multiple values to be entered into and removed from an input field accommodating array values. It makes use of the MultiTag component. Props can be passed to the component, allowing it to be reused across the project.
F096	src/client/components/Input/ MultiTextInput/ MultiTextInput.test.js	Front-end	To unit test the MultiTextInput component.	Created	A test file using Jest and Enzyme to test the component renders correctly.
F097	src/client/components/Input/ SelectInput/	Front-end	To create a reusable component to allow users to	Created	A React functional component creating a component allowing for multiple values from

	SelectInput.react.js		enter multiple values into an input field from a list of options, with appropriate styles.		a list to be entered into and removed from an input field accommodating array values. It makes use of the MultiTag component. Props can be passed to the component, allowing it to be reused across the project.
F098	src/client/components/Input/ SelectInput/ SelectInput.test.js	Front-end	To unit test the SelectInput component.	Created	A test file using Jest and Enzyme to test the component renders correctly, all imported components render properly, and all functions work as intended.
F099	src/client/components/Input/ TextInput/TextInput.react.js	Front-end	To create a reusable component to allow users to view a single input value for an input field for array values, with appropriate styles. The user can remove the entry from the array from this component.	Created	A React arrow functional component creating a component containing an input field accommodating string, numeric and password types. It enables a consistent text field appearance to be used across the application. Props can be passed to the component, allowing it to be reused across the project.
F100	src/client/components/Input/ TextInput/TextInput.test.js	Front-end	To unit test the TextInput component.	Created	A test file using Jest and Enzyme to test the component renders correctly.
F101	src/client/components/Modal/ AdminDeleteUserAccount/ AdminDeleteUserAccount.react.js	Front-end	To create a modal component to allow admin users to delete a specific user's account, acting as a cautionary check.	Created	A React class component creating a modal used in the UserDashboard component in the Admin section of the application. It makes use of the Modal and Button components, alongside the deleteAccount action which interacts with the MongoDB 'users' collection. Props can be passed to the component, allowing it to be reused across the project.
F102	src/client/components/Modal/ AdminDeleteUserAccount/ AdminDeleteUserAccount.test.js	Front-end	To unit test the AdminDeleteUserAccount component.	Created	A test file using Jest and Enzyme to test the component renders correctly.
F103	src/client/components/Modal/ AdminSendUserPasswordReset/ AdminSendUserPasswordReset.react.js	Front-end	To create a modal component to allow admin users to create a password reset request for a	Created	A React arrow functional component creating a modal used in the UserDashboard component in the Admin section of the application. It makes use of the Modal and

			user's account, acting as a confirmation message.		Button components. Props can be passed to the component, allowing it to be reused across the project.
F104	src/client/components/Modal/ AdminSendUserPasswordReset/ AdminSendUserPasswordReset.test.js	Front-end	To unit test the AdminSendUserPasswordReset component.	Created	A test file using Jest and Enzyme to test the component renders correctly.
F105	src/client/components/Modal/ AdminToggleUserPermission/ AdminToggleUserPermission.test.js	Front-end	To create a modal component to allow admin users to user permissions for a particular account, acting as a confirmation message.	Created	A React arrow functional component creating a modal used in the UserDashboard component in the Admin section of the application. It makes use of the Modal and Button components. Props can be passed to the component, allowing it to be reused across the project.
F106	src/client/components/Modal/ AdminToggleUserPermission/ AdminToggleUserPermission.react.js	Front-end	To unit test the AdminToggleUserPermission component.	Created	A test file using Jest and Enzyme to test the component renders correctly.
F107	src/client/components/Modal/ DeleteAccountModal/ DeleteAccountModal.react.js	Front-end	To create a modal component to allow a user to delete their account with the application, acting as a cautionary check.	Created	A React arrow functional component creating a modal used in the UpdateAccount component. It makes use of the Modal and Button components. Props can be passed to the component, allowing it to be reused across the project.
F108	src/client/components/Modal/ DeleteAccountModal/ DeleteAccountModal.test.js	Front-end	To unit test the DeleteAccountModal component.	Created	A test file using Jest and Enzyme to test the component renders correctly.
F109	src/client/components/Modal/ DeleteRecipeModal/ DeleteRecipeModal.react.js	Front-end	To create a modal component to allow a user to delete a recipe they have uploaded or an admin user to delete any recipe, acting as a cautionary check.	Created	A React arrow functional component creating a modal used in the YourRecipes and SingleRecipe components. It makes use of the Modal and Button components. Props can be passed to the component, allowing it to be reused across the project.
F110	src/client/components/Modal/ DeleteRecipeModal/ DeleteRecipeModal.test.js	Front-end	To unit test the DeleteRecipeModal component.	Created	A test file using Jest and Enzyme to test the component renders correctly.

F111	src/client/components/Modal/ LogoutModal/LogoutModal.react.js	Front-end	To create a modal component to allow a user to logout of their account, acting as a cautionary check.	Created	A React arrow functional component creating a modal accessed from the Navbar component. It makes use of the Modal and Button components. Props can be passed to the component, allowing it to be reused across the project.
F112	src/client/components/Modal/ LogoutModal/LogoutModal.test.js	Front-end	To unit test the LogoutModal component.	Created	A test file using Jest and Enzyme to test the component renders correctly.
F113	src/client/components/Modal/ PasswordResetModal/ PasswordResetModal.react.js	Front-end	To create a modal component to allow a user to reset their password for the application, acting as a confirmation message.	Created	A React arrow functional component creating a modal used in the UserDashboard component in the Admin section and the Login component. It makes use of the Modal and Button components. Props can be passed to the component, allowing it to be reused across the project.
F114	src/client/components/Modal/ PasswordResetModal/ PasswordResetModal.test.js	Front-end	To unit test the PasswordResetModal component.	Created	A test file using Jest and Enzyme to test the component renders correctly.
F115	src/client/components/Modal/ Modal.react.js	Front-end	To create a reusable Modal component to be used across the application.	Created	A React compound component, multiple React components are attached to a default component exported from the file. It enables a consistent Modal appearance throughout the application, and allows props to be passed to it to aid in reusability.
F116	src/client/components/Modal/ Modal.test.js	Front-end	To unit test the Modal, Background, Content, CloseIcon, Title, SubTitle and ButtonContainer components.	Created	A test file using Jest and Enzyme to test the components render correctly.
F117	src/client/components/Navbar/ Navbar.react.js	Front-end	To create a navigation bar which can be seen throughout the whole application and adapts to whether the user is logged in or not.	Created	A React arrow functional component creating a navigation bar containing links to components ad defined in the App.js file. It makes use of the logo image from the images folder. This component shows and hides links and functionality based on whether the user is logged in to their account.

F118	src/client/components/Navbar/ Navbar.test.js	Front-end	To unit test the Navbar component.	Created	A test file using Jest and Enzyme to test the component renders correctly.
F119	src/client/components/ ProtectedRoute/ ProtectedRoute.react.js	Front-end	To create a component that limits access to particular application paths.	Created	A React arrow functional component enabling checks to be ran before a user is routed to a path, particularly ensuring that they are logged in and have a valid token. Props can be passed to the component, allowing it to be reused multiple times.
F120	src/client/components/ ProtectedRoute/ ProtectedRoute.test.js	Front-end	To unit test the ProtectedRoute component.	Created	A test file using Jest and Enzyme to test the component renders correctly.
F121	src/client/components/Table/ Table.react.js	Front-end	To create a reusable Table component to be used across the application.	Created	A React compound component, multiple React components are attached to a default component exported from the file. It enables a consistent Table appearance throughout the application, and allows props to be passed to it to aid in reusability.
F122	src/client/components/Table/ Table.test.js	Front-end	To unit test the Table, Header, HeaderCell, Body, Row, and Cell components.	Created	A test file using Jest and Enzyme to test the components render correctly.
F123	src/client/components/ App.js	Front-end	To create a component allowing users to access other components in the application.	Created	A React functional component establishing routes to application components that act as front-facing pages. It is foundational to the running of the project. It makes use of the Navbar, ProtectedRoute, and InteractiveGuideIcon components. It provides connections to the Homepage, Planner, Login, SignUp, UpdateAccount, ForgotPassword, ArticlesHome, DisheTheDirt, PlantProblems, and SeasonalSustenance components. It also runs checks on whether the user is logged in to either show or hide the guide icon.

F124	src/client/components/ App.test.js	Front-end	To unit test the Table, Header, HeaderCell, Body, Row, and Cell components.	Created	A test file using Jest and Enzyme to test the component renders correctly.
F125	src/client/images/actions/ growing.PNG	Front-end	To represent that the particular produce item is in the user's currently growing list in the SingleProduce component.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a brown plant pot with a small green plant growing from it.
F126	src/client/images/actions/ not_growing.PNG	Front-end	To represent that the particular produce item is not in the user's currently growing list in the SingleProduce component.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as the outline of growing.PNG with no colour filling in the image.
F127	src/client/images/actions/ not_wishlist.PNG	Front-end	To represent that the particular produce item is not in the user's wish-list in the SingleProduce component and is not in the user's liked recipes list in the SingleRecipe component.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as the outline of wishlist.PNG with no colour filling in the image.
F128	src/client/images/actions/ wishlist.PNG	Front-end	To represent that the particular produce item is in the user's wish-list in the SingleProduce component, is in the user's liked recipes list in the SingleRecipe component, and the number of likes a recipe has in the SingleRecipe and RecipesHome components	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a yellow star with rounded edges.
F129	src/client/images/lightConditions/ full_shade.PNG	Front-end	To represent the 'Full Shade' option in the LightConditions component within the InteractiveGuide.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a cloud on a green circular background with a dark green border.
F130	src/client/images/lightConditions/ full_sun.PNG	Front-end	To represent the 'Full Sun' option in the LightConditions	Created	An image file created by the developer, using Procreate, possessing a cartoon style.

			component within the InteractiveGuide.		Appears as a sun on a green circular background with a dark green border.
F131	src/client/images/lightConditions/ part_shade.PNG	Front-end	To represent the 'Part Shade' option in the LightConditions component within the InteractiveGuide.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a sun behind a small cloud on a green circular background with a dark green border.
F132	src/client/images/lightConditions/ full_shade.PNG	Front-end	To represent the 'Full Shade' option in the LightConditions component within the InteractiveGuide.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a sun behind a large cloud on a green circular background with a dark green border.
F133	src/client/images/planner/ harvest.PNG	Front-end	To represent the 'Harvest' action in the Planner component.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a plant in the ground with a trowel beside it. It has a blue circular background with a dark blue border.
F134	src/client/images/planner/ plant.PNG	Front-end	To represent the 'Plant' action in the Planner component.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a plant in a pot hovering above the ground. It has a blue circular background with a dark blue border.
F135	src/client/images/planner/ sow.PNG	Front-end	To represent the 'Sow' action in the Planner component.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a bag of seeds being poured onto the ground. It has a blue circular background with a dark blue border.
F136	src/client/images/planner/ transplant.PNG	Front-end	To represent the 'Transplant' action in the Planner component.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a plant without a pot hovering above the ground. It has a blue circular background with a dark blue border.
F137	src/client/images/produce/details/ cooking.PNG	Front-end	To represent the 'Cooking' section in the SingleProduce component.	Created	An image file created by the developer, using Procreate, possessing a cartoon style.

					Appears as a white chef's hat on a green circular background with a dark green border.
F138	src/client/images/produce/details/ cupboard.PNG	Front-end	To represent the 'cool, dry place' storage location value in the SingleProduce component.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a brown cupboard on a green circular background with a dark green border.
F139	src/client/images/produce/details/ freezing.PNG	Front-end	To represent the 'Freezing' section in the SingleProduce component.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a light blue snowflake on a green circular background with a dark green border.
F140	src/client/images/produce/details/ fresh.PNG	Front-end	To accompany the value for how long the produce item is 'fresh' for in the SingleProduce component.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a green leaf in a white circle. It is on a green circular background with a dark green border.
F141	src/client/images/produce/details/ paper_bag.PNG	Front-end	To represent the 'paper bag' storage container value in the SingleProduce component.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a brown paper bag on a green circular background with a dark green border.
F142	src/client/images/produce/details/ perforated_bag.PNG	Front-end	To represent the 'perforated bag' storage container value in the SingleProduce component.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a light blue bag with small holes in it on a green circular background with a dark green border.
F143	src/client/images/produce/details/ plastic_bag.PNG	Front-end	To represent the 'plastic bag' storage container value in the SingleProduce component.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a light blue bag on a green circular background with a dark green border.
F144	src/client/images/produce/details/ refrigerator.PNG	Front-end	To represent the 'refrigerator' storage location value in the SingleProduce component.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a grey refrigerator on a green circular background with a dark green border.
F145	src/client/images/produce/details/ time.PNG	Front-end	To accompany the value for the length of 'time' it takes for	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a clock with a clockwise arrow

			the produce item to grow in the SingleProduce component.		around the outside of it on a green circular background with a dark green border.
F146	src/client/images/produce/items/edibleFlowers/bellis_daisy.PNG	Front-end	To represent the 'Bellis Daisy' edible flower.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a yellow centred, white bellis daisy flower.
F147	src/client/images/produce/items/edibleFlowers/bergamot.PNG	Front-end	To represent the 'Bergamot' edible flower.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a cluster of small, yellow centred, white bergamot flowers.
F148	src/client/images/produce/items/edibleFlowers/carnation.PNG	Front-end	To represent the 'Carnation' edible flower.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a red petaled carnation flower.
F149	src/client/images/produce/items/edibleFlowers/chamomile.PNG	Front-end	To represent the 'Chamomile' edible flower.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a yellow centred, white chamomile flower.
F150	src/client/images/produce/items/edibleFlowers/chrysanthemum.PNG	Front-end	To represent the 'Chrysanthemum' edible flower.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a yellow petaled chrysanthemum flower.
F151	src/client/images/produce/items/edibleFlowers/dandelion.PNG	Front-end	To represent the 'Dandelion' edible flower.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a yellow petaled dandelion flower.
F152	src/client/images/produce/items/edibleFlowers/dianthus.PNG	Front-end	To represent the 'Dianthus' edible flower.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a pink petaled dianthus flower.
F153	src/client/images/produce/items/edibleFlowers/elderflower.PNG	Front-end	To represent the 'Elderflower' edible flower.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a bunch of small-petaled white elderflowers.
F154	src/client/images/produce/items/edibleFlowers/geranium.PNG	Front-end	To represent the 'Geranium' edible flower.	Created	An image file created by the developer, using Procreate, possessing a cartoon style.

					Appears as a bunch of small-petaled pink geraniums.
F155	src/client/images/produce/items/ edibleFlowers/hibiscus.PNG	Front-end	To represent the 'Hibiscus' edible flower.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as an orange petaled hibiscus flower.
F156	src/client/images/produce/items/ edibleFlowers/honeyberry.PNG	Front-end	To represent the 'Honeyberry' edible flower.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as green leafed flower with blue berries.
F157	src/client/images/produce/items/ edibleFlowers/lavender.PNG	Front-end	To represent the 'Lavender' edible flower.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a bunch of tall lilac lavender flowers.
F158	src/client/images/produce/items/ edibleFlowers/nasturtium.PNG	Front-end	To represent the 'Nasturtium' edible flower.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as an orange petaled nasturtium flower.
F159	src/client/images/produce/items/ edibleFlowers/pansy.PNG	Front-end	To represent the 'Pansy' edible flower.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a purple petaled pansy flower.
F160	src/client/images/produce/items/ edibleFlowers/pot_marigold.PNG	Front-end	To represent the 'Pot Marigold' edible flower.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as an orange and yellow petaled pot marigold flower.
F161	src/client/images/produce/items/ edibleFlowers/primrose.PNG	Front-end	To represent the 'Primrose' edible flower.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a purple petaled primrose flower.
F162	src/client/images/produce/items/ edibleFlowers/rose.PNG	Front-end	To represent the 'Rose' edible flower.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a red petaled rose flower.
F163	src/client/images/produce/items/ edibleFlowers/sunflower.PNG	Front-end	To represent the 'Sunflower' edible flower.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a yellow petaled sunflower flower with a brown centre.

F164	src/client/images/produce/items/ edibleFlowers/tulip.PNG	Front-end	To represent the ‘Tulip’ edible flower.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a pink petaled tulip flower.
F165	src/client/images/produce/items/ edibleFlowers/viola.PNG	Front-end	To represent the ‘Viola’ edible flower.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a purple petaled pansy flower.
F166	src/client/images/produce/items/ fruit/apple.PNG	Front-end	To represent the ‘Apple’ fruit.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a red apple fruit.
F167	src/client/images/produce/items/ fruit/apricot.PNG	Front-end	To represent the ‘Apricot’ fruit.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as an orange apricot fruit.
F168	src/client/images/produce/items/ fruit/blackberry.PNG	Front-end	To represent the ‘Blackberry’ fruit.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a dark purple blackberry fruit.
F169	src/client/images/produce/items/ fruit/blueberry.PNG	Front-end	To represent the ‘Blueberry’ fruit.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a group of deep blue blueberry fruits.
F170	src/client/images/produce/items/ fruit/cherry.PNG	Front-end	To represent the ‘Acid Cherry’ and ‘Sweet Cherry’ fruits.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a group of two deep red cherries.
F171	src/client/images/produce/items/ fruit/citrus.PNG	Front-end	To represent the ‘Citrus’ fruit.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as an orange and lemon.
F172	src/client/images/produce/items/ fruit/cranberry.PNG	Front-end	To represent the ‘Cranberry’ fruit.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as deep pink cranberry fruits.
F173	src/client/images/produce/items/ fruit/fig.PNG	Front-end	To represent the ‘Fig’ fruit.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a purple fig fruit.
F174	src/client/images/produce/items/ fruit/gooseberry.PNG	Front-end	To represent the ‘Gooseberry’ fruit.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a group of two green gooseberries.

F175	src/client/images/produce/items/fruit/grape.PNG	Front-end	To represent the ‘Grape’ fruit.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a bunch of deep purple grapes.
F176	src/client/images/produce/items/fruit/hybridberry.PNG	Front-end	To represent the ‘Hybrid Berry’ fruit.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a deep red hybrid berry fruit.
F177	src/client/images/produce/items/fruit/kiwi.PNG	Front-end	To represent the ‘Kiwi’ fruit.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a whole brown kiwi fruit, alongside a halved one, showing its green centre.
F178	src/client/images/produce/items/fruit/melon.PNG	Front-end	To represent the ‘Melon’ fruit.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a pale lemon and green melon fruit.
F179	src/client/images/produce/items/fruit/mulberry.PNG	Front-end	To represent the ‘Mulberry’ fruit.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a dark purple mulberry fruit.
F180	src/client/images/produce/items/fruit/nectarine.PNG	Front-end	To represent the ‘Nectarine’ fruit.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as an orange nectarine fruit.
F181	src/client/images/produce/items/fruit/olive.PNG	Front-end	To represent the ‘Olive’ fruit.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a group of two green olives.
F182	src/client/images/produce/items/fruit/passionfruit.PNG	Front-end	To represent the ‘Passionfruit’ fruit.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a whole plum-coloured passionfruit, alongside a halved one, showing its yellow centre.
F183	src/client/images/produce/items/fruit/peach.PNG	Front-end	To represent the ‘Peach’ fruit.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a pale orange peach fruit.
F184	src/client/images/produce/items/fruit/pear.PNG	Front-end	To represent the ‘Pear’ fruit.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a green pear fruit.

F185	src/client/images/produce/items/fruit/pineapple.PNG	Front-end	To represent the 'Pineapple' fruit.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a golden yellow pineapple fruit with green leaves.
F186	src/client/images/produce/items/fruit/plum.PNG	Front-end	To represent the 'Plum' fruit.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a deep burgundy plum fruit.
F187	src/client/images/produce/items/fruit/quince.PNG	Front-end	To represent the 'Quince' fruit.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a golden yellow quince fruit.
F188	src/client/images/produce/items/fruit/raspberry.PNG	Front-end	To represent the 'Raspberry' fruit.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a dark pink raspberry fruit.
F189	src/client/images/produce/items/fruit/redcurrant.PNG	Front-end	To represent the 'Red Currant' fruit.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a bunch of dark red-coloured, red currant berries.
F190	src/client/images/produce/items/fruit/strawberry.PNG	Front-end	To represent the 'Summer Fruiting Strawberry', 'Perpetual Strawberry' and 'Alpine Strawberry' fruit.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a red strawberry fruit.
F191	src/client/images/produce/items/fruit/whitecurrant.PNG	Front-end	To represent the 'White Currant' fruit.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a bunch of white-coloured, white currant berries.
F192	src/client/images/produce/items/herbs/basil.PNG	Front-end	To represent the 'Basil' herb.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a group of green leaved basil.
F193	src/client/images/produce/items/herbs/bay.PNG	Front-end	To represent the 'Bay' herb.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a group of green leaved bay.
F194	src/client/images/produce/items/herbs/chervil.PNG	Front-end	To represent the 'Chervil' herb.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a group of green leaved chervil.

F195	src/client/images/produce/items/herbs/chives.PNG	Front-end	To represent the 'Chives' herb.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a group of green leaved chives.
F196	src/client/images/produce/items/herbs/coriander.PNG	Front-end	To represent the 'Coriander' herb.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a group of green leaved coriander.
F197	src/client/images/produce/items/herbs/dill.PNG	Front-end	To represent the 'Dill' herb.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a group of green leaved dill.
F198	src/client/images/produce/items/herbs/fennel.PNG	Front-end	To represent the 'Fennel' herb.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a group of green leaved fennel.
F199	src/client/images/produce/items/herbs/horseradish.PNG	Front-end	To represent the 'Horseradish' herb.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a green leaved horseradish with a white root.
F200	src/client/images/produce/items/herbs/lemon_balm.PNG	Front-end	To represent the 'Lemon Balm' herb.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a group of green leaved lemon balm.
F201	src/client/images/produce/items/herbs/lovage.PNG	Front-end	To represent the 'Lovage' herb.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a group of green leaved lovage.
F202	src/client/images/produce/items/herbs/marjoram.PNG	Front-end	To represent the 'Marjoram' herb.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a group of green leaved marjoram.
F203	src/client/images/produce/items/herbs/mint.PNG	Front-end	To represent the 'Mint' herb.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a group of green leaved mint.
F204	src/client/images/produce/items/herbs/oregano.PNG	Front-end	To represent the 'Oregano' herb.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a group of green leaved oregano.

F205	src/client/images/produce/items/herbs/parsley.PNG	Front-end	To represent the 'Parsley' herb.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a group of green leaved parsley.
F206	src/client/images/produce/items/herbs/rosemary.PNG	Front-end	To represent the 'Rosemary' herb.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a group of green leaved rosemary.
F207	src/client/images/produce/items/herbs/sage.PNG	Front-end	To represent the 'Sage' herb.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a group of green leaved sage.
F208	src/client/images/produce/items/herbs/savory.PNG	Front-end	To represent the 'Savory' herb.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a group of green leaved savory.
F209	src/client/images/produce/items/herbs/tarragon.PNG	Front-end	To represent the 'Tarragon' herb.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a group of green leaved tarragon.
F210	src/client/images/produce/items/herbs/thyme.PNG	Front-end	To represent the 'Thyme' herb.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a group of green leaved thyme.
F211	src/client/images/produce/items/vegetables/asparagus_pea.PNG	Front-end	To represent the 'Asparagus Pea' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a green asparagus pea vegetable.
F212	src/client/images/produce/items/vegetables/asparagus.PNG	Front-end	To represent the 'Asparagus' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a green asparagus vegetable.
F213	src/client/images/produce/items/vegetables/aubergine.PNG	Front-end	To represent the 'Aubergine' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a deep purple aubergine vegetable.
F214	src/client/images/produce/items/vegetables/beetroot.PNG	Front-end	To represent the 'Beetroot' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a deep burgundy beetroot vegetable.
F215	src/client/images/produce/items/vegetables/broad_bean.PNG	Front-end	To represent the 'Broad Bean' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a green broad bean vegetable.

F216	src/client/images/produce/items/vegetables/brussel_sprouts.PNG	Front-end	To represent the 'Brussel Sprouts' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a group of green brussel sprouts.
F217	src/client/images/produce/items/vegetables/cabbage.PNG	Front-end	To represent the 'Cabbage' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a green broad bean vegetable.
F218	src/client/images/produce/items/vegetables/calabrese.PNG	Front-end	To represent the 'Calabrese' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a green calabrese vegetable.
F219	src/client/images/produce/items/vegetables/carrot.PNG	Front-end	To represent the 'Carrot' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as an orange rooted carrot vegetable.
F220	src/client/images/produce/items/vegetables/cauliflower.PNG	Front-end	To represent the 'Cauliflower' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a green and white cauliflower vegetable.
F221	src/client/images/produce/items/vegetables/celeriac.PNG	Front-end	To represent the 'Celeriac' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a white rooted celeriac vegetable.
F222	src/client/images/produce/items/vegetables/celery.PNG	Front-end	To represent the 'Celery' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a green celery vegetable.
F223	src/client/images/produce/items/vegetables/chicory.PNG	Front-end	To represent the 'Chicory' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a light yellow rooted chicory vegetable.
F224	src/client/images/produce/items/vegetables/chilli_pepper.PNG	Front-end	To represent the 'Chilli Pepper' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a red chilli pepper vegetable.
F225	src/client/images/produce/items/vegetables/chinese_artichoke.PNG	Front-end	To represent the 'Chinese Artichoke' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a white-coloured Chinese artichoke vegetable.

F226	src/client/images/produce/items/vegetables/chinese_cabbage.PNG	Front-end	To represent the 'Chinese Cabbage' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a green Chinese cabbage vegetable.
F227	src/client/images/produce/items/vegetables/corn_salad.PNG	Front-end	To represent the 'Corn Salad' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a green leaved corn salad vegetable.
F228	src/client/images/produce/items/vegetables/courgette.PNG	Front-end	To represent the 'Courgette' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a green courgette vegetable.
F229	src/client/images/produce/items/vegetables/cucumber.PNG	Front-end	To represent the 'Cucumber' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a green cucumber vegetable.
F230	src/client/images/produce/items/vegetables/endive.PNG	Front-end	To represent the 'Endive' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a pale green endive vegetable.
F231	src/client/images/produce/items/vegetables/florence_fennel.PNG	Front-end	To represent the 'Florence Fennel' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a white rooted Florence fennel vegetable.
F232	src/client/images/produce/items/vegetables/french_bean.PNG	Front-end	To represent the 'French Bean' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a green French bean vegetable.
F233	src/client/images/produce/items/vegetables/garlic.PNG	Front-end	To represent the 'Garlic' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a cream-coloured garlic vegetable.
F234	src/client/images/produce/items/vegetables/gherkin.PNG	Front-end	To represent the 'Gherkin' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a green gherkin vegetable.
F235	src/client/images/produce/items/vegetables/globe_artichoke.PNG	Front-end	To represent the 'Globe Artichoke' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a green globe artichoke vegetable.

F236	src/client/images/produce/items/vegetables/hamburg_parsley.PNG	Front-end	To represent the 'Hamburg Parsley' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a white-rooted Hamburg parsley vegetable.
F237	src/client/images/produce/items/vegetables/jerusalem_artichoke.PNG	Front-end	To represent the 'Jerusalem Artichoke' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a cream-coloured Jerusalem artichoke vegetable.
F238	src/client/images/produce/items/vegetables/kale.PNG	Front-end	To represent the 'Kale' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a group of leafy green kale vegetables.
F239	src/client/images/produce/items/vegetables/kohlrabi.PNG	Front-end	To represent the 'Kohlrabi' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a green kohlrabi vegetable.
F240	src/client/images/produce/items/vegetables/land_cress.PNG	Front-end	To represent the 'Land Cress' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a green leaved land cress vegetable.
F241	src/client/images/produce/items/vegetables/leek.PNG	Front-end	To represent the 'Leek' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a green and cream leek vegetable.
F242	src/client/images/produce/items/vegetables/lettuce.PNG	Front-end	To represent the 'Lettuce' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a green lettuce vegetable.
F243	src/client/images/produce/items/vegetables/mushroom.PNG	Front-end	To represent the 'Mushroom' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a brown and cream mushroom vegetable.
F244	src/client/images/produce/items/vegetables/onion.PNG	Front-end	To represent the 'Onion' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a light brown onion vegetable.
F245	src/client/images/produce/items/vegetables/pak_choi.PNG	Front-end	To represent the 'Pak Choi' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a green pak choi vegetable.

F246	src/client/images/produce/items/vegetables/parsnip.PNG	Front-end	To represent the 'Parsnip' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a white rooted parsnip vegetable.
F247	src/client/images/produce/items/vegetables/pea.PNG	Front-end	To represent the 'Pea' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a green pea vegetable.
F248	src/client/images/produce/items/vegetables/potato.PNG	Front-end	To represent the 'Potato' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a light brown potato vegetable.
F249	src/client/images/produce/items/vegetables/pumpkin.PNG	Front-end	To represent the 'Pumpkin' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as an orange pumpkin vegetable.
F250	src/client/images/produce/items/vegetables/radish.PNG	Front-end	To represent the 'Radish' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a deep pink radish vegetable.
F251	src/client/images/produce/items/vegetables/red_cabbage.PNG	Front-end	To represent the 'Red Cabbage' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a deep red cabbage vegetable.
F252	src/client/images/produce/items/vegetables/rhubarb.PNG	Front-end	To represent the 'Rhubarb' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a green leaved rhubarb vegetable with deep pink stalks.
F253	src/client/images/produce/items/vegetables/rocket.PNG	Front-end	To represent the 'Rocket' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a green leafy rocket vegetable.
F254	src/client/images/produce/items/vegetables/runner_bean.PNG	Front-end	To represent the 'Runner Bean' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a green runner bean vegetable.
F255	src/client/images/produce/items/vegetables/salsify.PNG	Front-end	To represent the 'Salsify' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a cream-coloured salsify vegetable.
F256	src/client/images/produce/items/vegetables/scorzonera.PNG	Front-end	To represent the 'Scorzonera' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style.

					Appears as a brown-coloured scorzonera vegetable.
F257	src/client/images/produce/items/vegetables/shallot.PNG	Front-end	To represent the 'Shallot' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a burgundy shallot vegetable.
F258	src/client/images/produce/items/vegetables/spinach.PNG	Front-end	To represent the 'Spinach' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a group of green leafy spinach.
F259	src/client/images/produce/items/vegetables/sprouting_broccoli.PNG	Front-end	To represent the 'Sprouting Broccoli' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a purple sprouting broccoli vegetable.
F260	src/client/images/produce/items/vegetables/summer_squash.PNG	Front-end	To represent the 'Summer Squash' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a golden yellow summer squash vegetable.
F261	src/client/images/produce/items/vegetables/swede.PNG	Front-end	To represent the 'Swede' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a purple and cream swede vegetable.
F262	src/client/images/produce/items/vegetables/sweet_pepper.PNG	Front-end	To represent the 'Sweet Pepper' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a red sweet pepper vegetable.
F263	src/client/images/produce/items/vegetables/sweet_potato.PNG	Front-end	To represent the 'Sweet Potato' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a light brown sweet potato vegetable.
F264	src/client/images/produce/items/vegetables/sweetcorn.PNG	Front-end	To represent the 'Sweetcorn' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a pale-yellow sweetcorn vegetable.
F265	src/client/images/produce/items/vegetables/swiss_chard.PNG	Front-end	To represent the 'Swiss Chard' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a green leafy Swiss chard vegetable, with deep pink stalks.

F266	src/client/images/produce/items/vegetables/tomato.PNG	Front-end	To represent the 'Tomato' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a red tomato vegetable.
F267	src/client/images/produce/items/vegetables/turnip.PNG	Front-end	To represent the 'Turnip' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a purple and cream turnip vegetable.
F268	src/client/images/produce/items/vegetables/winter_squash.PNG	Front-end	To represent the 'Winter Squash' vegetable.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a light brown winter squash vegetable.
F269	src/client/images/produce/edible_flowers.PNG	Front-end	To represent the 'Edible Flowers' option in the ProduceType component within the InteractiveGuide.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a variety of edible flowers in a wooden box. It has a green circular background with a dark green border.
F270	src/client/images/produce/fruit.PNG	Front-end	To represent the 'Fruit' option in the ProduceType component within the InteractiveGuide.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a variety fruit in a wooden box. It has a green circular background with a dark green border.
F271	src/client/images/produce/herbs.PNG	Front-end	To represent the 'Herbs' option in the ProduceType component within the InteractiveGuide.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a variety of herbs in a wooden box. It has a green circular background with a dark green border.
F272	src/client/images/produce/vegetables.PNG	Front-end	To represent the 'Vegetables' option in the ProduceType component within the InteractiveGuide.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a variety of vegetables in a wooden box. It has a green circular background with a dark green border.
F273	src/client/images/recipes/plate.PNG	Front-end	To represent a plate in the RecipesHome and SingleRecipe component if the user has not	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a white plate in the middle of a silver-coloured knife and fork set. It has a

			uploaded an image with their recipe.		green circular background with a dark green border.
F274	src/client/images/skillLevel/ level_beginner.PNG	Front-end	To represent the novice option in the Competency component within the InteractiveGuide.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a trowel with a blue handle. It has a blue circular background with a dark blue border.
F275	src/client/images/skillLevel/ level_expert.PNG	Front-end	To represent the expert option in the Competency component within the InteractiveGuide.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a trowel with a red handle. It has a red circular background with a dark red border.
F276	src/client/images/skillLevel/ level_moderate.PNG	Front-end	To represent the competent option in the Competency component within the InteractiveGuide.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a trowel with a green handle. It has a green circular background with a dark green border.
F277	src/client/images/space/ indoor.PNG	Front-end	To represent the 'Indoor' option in the SpaceType component within the InteractiveGuide.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a plant in a pot within a house. It has a green circular background with a dark green border.
F278	src/client/images/space/ outdoor.PNG	Front-end	To represent the 'Outdoor' option in the SpaceType component within the InteractiveGuide.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a plant in a pot underneath a sun and cloud. It has a green circular background with a dark green border.
F279	src/client/images/space/ under_cover.PNG	Front-end	To represent the 'Under Cover' option in the SpaceType component within the InteractiveGuide.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a plant in a pot underneath a sun and cloud. It has a green circular background with a dark green border.
F280	src/client/images/ cover_image.PNG	Front-end	To be used as a general image for the Homepage component showing all types of produce	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as four boxes in a row with a

			the application holds information on.		different group of produce in each: fruit, vegetables, herbs, and edible flowers. Arrows with text are present to label each produce box.
F281	src/client/images/ down_arrow.PNG	Front-end	To be used in the SelectInput component as the arrow representing that there is a drop-down.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a black arrow pointing down, which can be flipped to be an upward facing arrow.
F282	src/client/images/ interactive_guide.PNG	Front-end	To represent the interactive guide feature.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a blue speech bubble with a lightbulb inside it. It has a green circular background with a dark green border.
F283	src/client/images/ leaves_bg.PNG	Front-end	To act as a background to break up page content, predominantly for account sections.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a white background with green leaves woven across the entirety of the image.
F284	src/client/images/logo.PNG	Front-end	To act as the application logo on the navigation bar, which will also represent a link to the Homepage.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as black text containing the application name “The Plant Pantry” on a white background. The “t” at the end of the word “Plant” is green.
F285	src/client/redux/actions/ ActionTypes/ accountActionTypes.js	Front-end	To contain variables representing string values used in the account actions and reducers, ensuring there are no spelling mistakes during development.	Created	A JavaScript file containing variables which are dispatched from account actions to the account reducer as an action type. All string variables are capitalised and use underscores between words instead of spaces.
F286	src/client/redux/actions/ ActionTypes/ accountActionTypes.test.js	Front-end	To unit test the account action types.	Created	A test file using Jest and Enzyme to test the account action types return the correct string value.

F287	src/client/redux/actions/ ActionTypes/ produceActionTypes.js	Front-end	To contain variables representing string values used in the produce actions and reducers, ensuring there are no spelling mistakes during development.	Created	A JavaScript file containing variables which are dispatched from produce actions to the produce reducer as an action type. All string variables are capitalised and use underscores between words instead of spaces.
F288	src/client/redux/actions/ ActionTypes/ produceActionTypes.test.js	Front-end	To unit test the produce action types.	Created	A test file using Jest and Enzyme to test the account produce types return the correct string value.
F289	src/client/redux/actions/ ActionTypes/ recipeActionTypes.js	Front-end	To contain variables representing string values used in the recipe actions and reducers, ensuring there are no spelling mistakes during development.	Created	A JavaScript file containing variables which are dispatched from recipe actions to the recipe reducer as an action type. All string variables are capitalised and use underscores between words instead of spaces.
F290	src/client/redux/actions/ ActionTypes/ recipeActionTypes.test.js	Front-end	To unit test the recipe action types.	Created	A test file using Jest and Enzyme to test the recipe action types return the correct string value.
F291	src/client/redux/actions/ accountActions.js	Front-end	To contain all the actions which make API calls for account functionality.	Created	A JavaScript file containing all the functions required for application functionality for account features, making requests to API endpoints and dispatching actions based on the response given. It contains nine functions: getLogin, getSignUp, getLogout, getAccount, deleteAccount, updateAccount, createForgotPassword, getForgotPassword, and deleteForgotPassword.
F292	src/client/redux/actions/ accountActions.test.js	Front-end	To unit test the account action creator functions.	Created	A test file using Jest and Enzyme to test the account action creators return the correct object values.
F293	src/client/redux/actions/ produceActions.js	Front-end	To contain all the actions which make API calls for produce functionality.	Created	A JavaScript file containing all the functions required for application functionality for produce features, making requests to API endpoints and dispatching actions based on the response given. It contains three

					functions: getProduce, getProduceType, and getTailoredProduceList.
F294	src/client/redux/actions/ produceActions.test.js	Front-end	To unit test the produce action creator functions.	Created	A test file using Jest and Enzyme to test the produce action creators return the correct object values.
F295	src/client/redux/actions/ recipeActions.js	Front-end	To contain all the actions which make API calls for recipe functionality.	Created	A JavaScript file containing all the functions required for application functionality for recipe features, making requests to API endpoints and dispatching actions based on the response given. It contains eight functions: getRecipes, getUserRecipes, getSingleRecipe, createRecipe, updateRecipe, deleteRecipe, deleteUserRecipes, getTailoredRecipes.
F296	src/client/redux/actions/ recipeActions.test.js	Front-end	To unit test the recipe action creator functions.	Created	A test file using Jest and Enzyme to test the recipe action creators return the correct object values.
F297	src/client/redux/reducers/ accountReducer.js	Front-end	To update the account information held in the application store based on the action dispatched.	Created	A JavaScript file containing a function acting as the account reducer. Within the function there is a switch statement which determines how to update the store based on the type of the action dispatched.
F298	src/client/redux/reducers/ accountReducer.test.js	Front-end	To test the accountReducer.	Created	A test file using Jest and Enzyme to test the accountReducer returns the correct object to the application store.
F299	src/client/redux/reducers/ index.js	Front-end	To combine all of the reducers together.	Created	A JavaScript file which combines the account and produce reducers together so that when an action is dispatched both reducers are checked for a case in the switch statements which matches the action type.
F300	src/client/redux/reducers/ initialState.js	Front-end	To establish the initial application state provided to the store, which is manipulated by actions	Created	A JavaScript file containing an object representing the initial value of the application store. It currently contains two objects representing the account and

			dispatched during application usage.		produce sections, which aids in the reducer functionality.
F301	src/client/redux/reducers/ produceReducer.js	Front-end	To update the produce information held in the application store based on the action dispatched.	Created	A JavaScript file containing a function acting as the produce reducer. Within the function there is a switch statement which determines how to update the store based on the type of the action dispatched.
F302	src/client/redux/reducers/ produceReducer.test.js	Front-end	To test the produceReducer.	Created	A test file using Jest and Enzyme to test the produceReducer returns the correct object to the application store.
F303	src/client/redux/reducers/ recipeReducer.js	Front-end	To update the recipe information held in the application store based on the action dispatched.	Created	A JavaScript file containing a function acting as the recipe reducer. Within the function there is a switch statement which determines how to update the store based on the type of the action dispatched.
F304	src/client/redux/reducers/ recipeReducer.test.js	Front-end	To test the recipeReducer.	Created	A test file using Jest and Enzyme to test the recipeReducer returns the correct object to the application store.
F305	src/client/styles/components/ containers/Account.scss	Front-end	To create the styles for the account components.	Created	A SCSS file containing the styles for the account components, through the use of classes.
F306	src/client/styles/components/ containers/Article.scss	Front-end	To create the styles for the article components.	Created	A SCSS file containing the styles for the article components, through the use of classes.
F307	src/client/styles/components/ containers/Basic.scss	Front-end	To create the styles which are used across various container components.	Created	A SCSS file containing the general styles used across various container components in the application, through the use of classes. For instance, page title styling.
F308	src/client/styles/components/ containers/Homepage.scss	Front-end	To create the styles for the Homepage component.	Created	A SCSS file containing the styles for the homepage component, through the use of classes.
F309	src/client/styles/components/ containers/ InteractiveGuide.scss	Front-end	To create the styles for the InteractiveGuide component.	Created	A SCSS file containing the styles for the guide component, through the use of classes.

F310	src/client/styles/components/containers/Planner.scss	Front-end	To create the styles for the Planner component.	Created	A SCSS file containing the styles for the planner component, through the use of classes.
F311	src/client/styles/components/containers/Produce.scss	Front-end	To create the styles for the Produce components.	Created	A SCSS file containing the styles for the produce components, through the use of classes.
F312	src/client/styles/components/containers/Profile.scss	Front-end	To create the styles for the Profile components.	Created	A SCSS file containing the styles for the profile components, through the use of classes.
F313	src/client/styles/components/containers/Recipe.scss	Front-end	To create the styles for the Recipe components.	Created	A SCSS file containing the styles for the recipe components, through the use of classes.
F314	src/client/styles/components/containers/SingleItemPage.scss	Front-end	To create the styles for the SingleItemPage component.	Created	A SCSS file containing the styles for the single item display components, through the use of classes.
F315	src/client/styles/components/Button.scss	Front-end	To create the styles for the Button component.	Created	A SCSS file containing the styles for the button component, through the use of classes.
F316	src/client/styles/components/Card.scss	Front-end	To create the styles for the Card component.	Created	A SCSS file containing the styles for the card component, through the use of classes.
F137	src/client/styles/components/EmptyListMessage.scss	Front-end	To create the styles for the EmptyListMessage component.	Created	A SCSS file containing the styles for the empty list message, through the use of classes.
F318	src/client/styles/components/Input.scss	Front-end	To create the styles for the TextInput component.	Created	A SCSS file containing the styles for the TextInput component, through the use of classes.
F319	src/client/styles/components/Modal.scss	Front-end	To create the styles for the Modal component.	Created	A SCSS file containing the styles for the modal component, through the use of classes.
F320	src/client/styles/components/containers/MultiTag.scss	Front-end	To create the styles for the MultiTag component.	Created	A SCSS file containing the styles for the MultiTag component, through the use of classes.
F321	src/client/styles/components/Navbar.scss	Front-end	To create the styles for the Navbar component.	Created	A SCSS file containing the styles for the navigation bar component, through the use of classes.

F322	src/client/styles/components/ SelectInput.scss	Front-end	To create the styles for the SelectInput component.	Created	A SCSS file containing the styles for the multiple select input component, through the use of classes.
F323	src/client/styles/components/ Table.scss	Front-end	To create the styles for the Table component.	Created	A SCSS file containing the styles for the table component, through the use of classes.
F324	src/client/styles/_variables.scss	Front-end	To establish variables representing values that are used across the stylesheets.	Created	A SCSS file containing variables representing colours with hex codes, ensuring that the scheme remains the same throughout the application.
F325	src/client/styles/main.scss	Front-end	To import all the stylesheets into the one file, so that only this file is required to be imported into the application	Created	A SCSS file importing all of the other SCSS files in the 'styles' directory. It also imports a font and assigns it to the application body.
F326	src/client/styles/mixins.scss	Front-end	To create styles that can be used across all stylesheets in the application.	Created	A SCSS file containing the mixins for the applications, for instance, a scrollbar.
F327	src/client/index.html	Front-end	To establish the HTML template the React application code is injected into.	Created	A HTML file which acts as the application entry point. It contains minimal information, only a div which the application code is injected into and the title which appears on the browser tab.
F328	src/client/index.js	Front-end	To establish the application code which is injected into the HTML file.	Created	A JavaScript file which determines the code to be injected into the div in the HTML file. It imports the App.js file, wrapping it in a Provider which establishes the application store. It also enables Redux DevTools to be used if the Chrome browser is being used during development.
F329	src/client/logo.PNG	Front-end	To act as the icon in the browser tab when the application is running.	Created	An image file created by the developer, using Procreate, possessing a cartoon style. Appears as a blue speech bubble with a lightbulb inside it. It has a green circular background with a dark green border. It is the same as the icon for the interactive guide at this time.

F330	src/client/reset.scss	Front-end	To reset the application styling before applying other styles.	Created	An SCSS file which contains a reset style for the body, removing all margins.
F331	src/server/data/produce/ edibleFlowers.json	Back-end	To define information for all edible flowers to be added to the produce collection in MongoDB.	Created	A JSON file containing and array of data for twenty types of edible flowers. Information for each object in the array contains values such as the name, type, description, basic instructions, best locations, varieties, planner information, and instructions for storage and cooking.
F332	src/server/data/produce/ fruit.json	Back-end	To define information for all fruit to be added to the produce collection in MongoDB.	Created	A JSON file containing and array of data for twenty-nine types of fruit. Information for each object in the array contains values such as the name, type, description, basic instructions, best locations, varieties, planner information, and instructions for storage and cooking.
F333	src/server/data/produce/ herbs.json	Back-end	To define information for all herbs to be added to the produce collection in MongoDB.	Created	A JSON file containing and array of data for twenty types of herbs. Information for each object in the array contains values such as the name, type, description, basic instructions, best locations, varieties, planner information, and instructions for storage and cooking.
F334	src/server/data/produce/ vegetables.json	Back-end	To define information for all vegetables to be added to the produce collection in MongoDB.	Created	A JSON file containing and array of data for sixty-four types of vegetables. Information for each object in the array contains values such as the name, type, description, basic instructions, best locations, varieties, planner information, and instructions for storage and cooking.
F335	src/server/data/seed-db.js	Back-end	To add all the data held in the JSON files in the data directory to their corresponding MongoDB collections.	Created	A JavaScript file which connects to the MongoDB cluster and contains a function, which when called adds a specified array of objects into a specified collection. This information is passed into the function as

					parameters and is how the JSON files for produce and the users is added to the MongoDB cluster.
F336	src/server/data/users.json	Back-end	To define information for test users to be added to the users collection in MongoDB.	Created	A JSON file containing and array of data for two dummy users. Information for each object in the array contains the username, email, admin permission, and password.
F337	src/server/api-router.js	Back-end	Define the functionality of each endpoint used to retrieve and perform actions on information held in the application MongoDB collections.	Created	A JavaScript file which defines the endpoints which can be accessed by the application to perform create, read, update, and delete functionality. There are eleven routes defined, four of these do not require a valid JWT token as they are called when the user is not logged into the application so a token will not have been allocated yet. Error handling is thoroughly handled.
F338	src/server/create-express-app.js	Back-end	To create the Express API.	Created	A JavaScript file which contains a function which sets up the Express API, returning the routes defines in the api-router.js file and utilising CORS.
F339	src/server/index.js	Back-end	To connect to the MongoDB cluster.	Created	A JavaScript file which runs the function setting up the API and makes the connection to the MongoDB cluster, effectively running the back end of the application.
F340	src/tools/fileMock.js	Front-end	To define functionality regarding file handling that needs mocked for Jest and Enzyme unit tests for React components.	Created	A JavaScript file, which is currently empty, but has been set up. This is so that in the future if file handling is required in the application, functions that needs mocked regarding this can be added here for use across all unit tests.
F341	src/tools/styleMock.js	Front-end	To define styles that need mocked for Jest and Enzyme unit tests for React components.	Created	A JavaScript file, which is currently empty, but has been set up. This is so that in the future if unit tests testing application styles are needed, mocks can be established here.

F342	src/tools/testSetup.js	Front-end	To define general functionality that needs mocked for Jest and Enzyme unit tests for React components.	Created	A JavaScript file to mock functionality that is used across the application that is necessary for unit testing. Currently a localStorage mock is present.
F343	.env.enc	Back-end	To define environment variables required for connecting to MongoDB and managing user tokens.	Created	An encrypted version of the .env file not included in the submission. It contains two encrypted strings, one with the secret required for creating and checking user tokens and another containing the connection string for connecting to the MongoDB Atlas cluster. This file is encrypted for security purposes, to protect the user and database information.
F344	.gitignore	Front-end / Back-end	To define files which are not to be pushed to the repository storing the code.	Modified	A file containing a list of files which are not included in the code repository when a new commit is made. Most notably this includes the node_modules folder, as this is local to the developer, and the .env file, as it contains information which must be managed securely.
F345	package-lock.json	Front-end / Back-end	To define the versions of each package installed in the node_modules folder for the application.	Modified	A JSON file which contains the current versions of all packages which have been installed in the node_modules folder in addition to required packages. It is a good way to keep track of development.
F346	package.json	Front-end / Back-end	To define application information vital to the project.	Modified	A JSON file which contains core information fundamental to the operation of the application. It contains scripts, dependencies, rules, and other configurations.
F347	README.md	Front-end / Back-end	To provide an application description to be visibly shown in the code repository.	Modified	A file containing a simple message about the application, this will be built upon as the application grows.
F348	webpack.dev.config.js	Front-end / Back-end	To create a development version of the front-end of the	Created	A JavaScript file containing Webpack configuration information for the development environment, such as

			code which is ran using the ‘npm start:dev’ command.		templates and how to manage file types. This is the main configuration file for the front end of the application.
F349	webpack.prod.config.js	Front-end / Back-end	To create a production version of the front-end of the code which is ran using the ‘npm build’ command.	Created	A JavaScript file containing Webpack configuration information, such as templates and how to manage file types, for a production environment. This is not used currently as the application is not deployed anywhere and this file does not combine with the server script, but may be used in the future.