University of Glasgow | School of Computing Science

Honours Individual Project Dissertation

Portion Mate

Daily Food Intake Tracker

Inesh Bose

Academic Session 2021-22

Abstract

The Eatwell Guide is a recommendation given by Public Health England on having a balanced diet. Following and maintaining a diet may be difficult for people, and logging can help them keep track of their plan. However, the task of logging itself can also require a lot of commitment which can cause people inconvenience and make them give it up.

This project aims to develop an application that solves these problems of maintaining a diet plan; this was achieved through months of research and development into users, food and logging, in order to create a fully-integrated and convenient system that does not add a burden onto users, but rather motivates them to adhere to their plan. Considering the time-frame, this project also leaves development open-ended with scope for future work that can be passed over to or taken over by developers who can make use of the open-source side projects created and best practices followed by the system without great difficulty.

To confirm the usability of the application, evaluation was carried out through a survey where participants could share their thoughts with the help of System Usability Scale that also in-turn helps in analysing the results to a score that deemed the system as acceptable.

Acknowledgements

My sincere gratitude goes to my supervisor, Dr. Oana Andrei, for being the most understanding and providing guidance to this fourth year student who aimed high and stressed endlessly for this project. Most of the development has also been made possible because of my previous/current employers who have given me incredible amount of precious experience and knowledge in the area of my interest & passion - computing and software development.

The participants who helped evaluate the project and provide important feedback. The communities and developers on platforms like GitHub, StackOverflow, Reddit, Discord, and many-many open-source projects that have given an immeasurable amount of help in providing solutions to problems that would take months. My friends and, most importantly, my family who make me who I am today with all their support and providing me with the opportunity to come to Glasgow to learn from one of the greatest educational institutions in the world.

I would also like to thank **you**, the reader, for taking the valuable time to take a look at this dissertation, which already means the world to me. Thank you.

Educational Use Consent

I hereby grant my permission for this project to be stored, distributed and shown to other University of Glasgow students and staff for educational purposes. **Please note that you are under no obligation to sign this declaration, but doing so would help future students.**

Signature: Inesh Bose Date: 1 April 2022

Contents

List of Figures

List of Tables

1 Introduction

This chapter introduces the dissertation by sharing the motivation and aims of the project of discussion, and providing a high-level summary of the concepts discussed throughout the paper.

1.1 Overview

In this dissertation, we discuss planning and strategies adopted for the development and evaluation of a cross-platform application named "Portion Mate". This project was proposed by Dr. Oana Andrei for the Level 4 Individual Project course [46] at the University of Glasgow during the academic year 2021-22, and undertaken by student Inesh Bose.

An interface is a shared boundary between entities allowing information to be exchanged. The research of designing computer interfaces to be used by humans is called Human Computer Interaction (HCI) and it aims to solve problems and innovate with users in mind. Inspired from that, many factors and considerations went into each decision of the project. These are described in detail over the chapters of this dissertation. Since the application is focused on food and health, you will read and learn about food & culture in the country of development (United Kingdom), and nutrition, along with what similar and competing applications offer. As the development is carried out by a fourth-year university student, the development aspires to use software engineering industry standard agile practices and technologies.

1.2 Motivation

Modern life is very different from life two decades ago. Technology is the obvious, biggest change, but there are many lifestyle changes as well. Food is now available cheap and fast - this does not necessarily mean that food is healthier – the consumption for such items has increased exponentially causing people a lot of health problems [53, 55].

Mobile applications have been on the rise and in demand ever since affordable smartphones were introduced with reliable internet connections all over the globe [75, 114, 22]. They can be found in the pockets of most people of different ages now. Since the devices are portable and small, the applications make accessing a service convenient and easy [13] - this can include healthcare services [113]. A significant example is the usage of "Contact Tracing" and "Vaccine Passport" methods through mobile applications during the COVID-19 pandemic [23, 109].

Fitness and food tracking applications exist, however most have not provided a very suitable way of logging activity therefore becoming a burden on users. Food, despite being a very cultural and social element in many lifestyles and regions, should not be overlooked, but rather be monitored and consumed with diet in mind as it is a major factor for personal development and health.

More motivations are discussed in depth in Chapter 2.

1.3 Aims

As a product-focused software engineering project, our goal is to make a system that would be reliable and efficient for users, acting as a support through their food journey. This would mean developing a mobile application that can be used by users such as patients, nutritionists and general users to monitor their portion intake which should be part of their diet plan. This application must be well-designed and provide convenience to users; this should be judged by conducting evaluation of the product with the appropriate users.

The minimal viable product must be produced and refined within the timeframe of the project. Additionally, this system should be maintainable, robust and well-documented from the code-side making use of best practices, so that it enables other developers to understand and be inspired by the implementation choices of the project, opening doors to future possibilities.

1.4 Outline

This dissertation is structured into seven chapters, along with appendices that are referred between text. These chapters intend to separate aspects and phases of the projects into categories, and are ordered according to the development sequence.

- **Chapter 2** *Background* shares the underlying circumstances and work already done related to the area of the project by talking about food, diet and nutrition;
- **Chapter 3** *Requirements* discusses the expectations, issues and tasks that would outline problems to solve and a plan for the development of the project output;
- **Chapter 4** *Design* conveys solutions for the problems the project wishes to address with heavy focus on users, and any that may come up during the implementation;
- **Chapter 5** *Implementation* is about putting everything planned and discussed from previous chapters into practice to create the final product with technical details;
- **Chapter 6** *Evaluation* evaluates the final implemented product by discussing methods of collecting usability data, and sharing results of the analysis;
- **Chapter 7** *Conclusion* summarises and looks back on the dissertation, sharing lessons that were learnt, with future work and projects that were inspired by this.

2 Background

This chapter shares the background of the dissertation by going deeper into the circumstances and motivations that led to the proposal of the project.

2.1 Eating Habits in the UK

This project was developed in Glasgow, UK and the Eatwell Guide is given by the Healthcare Service of England (NHS). To start with, it is important to learn about the lifestyle around food in the UK which is discussed in this section.

2.1.1 Day Plan

The average height and weight for men in England is 175.3cm and 83.6kg respectively, and for women it is 70.2kg and 161.6cm[103]. This average person in the UK would aim to have three main meals a day.

(a) English breakfast (*from Freaky Fries on Wikimedia*)

(b) Tesco meal deals (*from Rakka on Flickr*)

(c) Roast beef (*from Adam Burt on Flickr*)

Figure 2.1: Popular daily food consumed in the UK

Breakfast

The first meal of the day would be at any time for different persons, but popularly it is targeted to be between 07:00 and 09:00. Like an "American breakfast" is known to be a plate of pancakes, waffles, cereal, juice, bacon and egg, the "tranditional" "full English breakfast"[1] consists of fried eggs, bacon, sausages, baked beans, black pudding and toast. It is a substantial meal that dates back to the 13th century [50]. Given the pan-fried meat involved, some may deem it to be unhealthy, and a study suggests that skipping breakfast might be better than eating unhealthy foods [78]. The NHS, however, suggests to start a day with a light bite, such as fruit or yoghurt, as the morning appetite will naturally increase and snacking will reduce [49].

[1] The quotes here are to indicate labels, as there may be those who have other beliefs.

Lunch

The second meal is expected to be midday (between 12:00 and 14:00), and in the UK, this meal is mostly simple and easy consisting of sandwiches (also known as "*butty*") that can have many different items between the bread. The most common and cheap filling would be ham and cheese, or chips (therefore the popular "chip butty"), or just crushed crisps.

In the current times, the nation also feeds on meal deals (where normally a food and a drink could be bought at a discounted price) offered by chains like Greggs and Tesco.

Dinner

The "formal" meal – Dinner, also called "*Tea*" - is consumed between 18:00 and 20:00. In many cultures, it is supposed to be the largest meal of the day, but this may be discouraged as usually sleep follows dinner (therefore the consumed calories are not utilised). The rule followed for dinner is "meat and two veg" and typically that is a roast dinner. Since this meal is at the end of day when routine & work has ended and temperature has fallen, one would also choose to unwind with a relatively "lavish", enjoyable meal. This could mean cooking cultural food, or simply do a takeaway from a nearby "*chippy*" or stores.

2.1.2 Processed Foods

Over the years, there are options in the market that are easier and cheaper than cooking food yourself (previously discussed in 1.2). Processed foods are consumed daily by most people [10, 52, 92], and they are unavoidable as some processing may be essential, like milk [35].

However, there are also frozen / microwave meals that can be prepared by heating for a few minutes in the microwave or oven, and can also be stored in the freezer for months. For example, pre-assembled pizzas are available in supermarkets, cost as much as a Greggs sausage roll (£1), and just requires baking in the oven for 10 minutes. However, a survey also discovered that one in ten people in the UK have not cooked a meal from scratch in over a year, choosing such alternatives for consumption [6]. Food stores may also use taste enhancing chemicals (such as monosodium glutamate) to addict consumers to their strategically prices meals.

(a) Bowl of cereal with milk (*from PxHere*)

(b) Frozen pizzas (*from Amin on Wikimedia*)

(c) Microwave meal (*from Open Food Facts*)

Figure 2.2: Examples of processed foods

2.1.3 Various Diets

While some cultures restrict meat, some view meat as a social celebration, a Reddit post discussed the consumption of meat in the UK [117] and most answers justify the meat for the protein and adding taste to relatively "boring" dinners that would otherwise consist of vegetables only.

Through the day plan of a person in the UK (discussed in 2.1.1), a person could consume a lot of meat for all meals of the day (full breakfast that revolves around different meats, ham sandwich and roast chicken). Unfortunately, this causes a big impact on the environment since meat production creates a large percentage (60%) of greenhouse gases every year [65].

A new trend has also been on the rise due to this called "veganism" where individuals would choose to cut out all animals product (such as dairy and poultry) from their diet. 3% of the population in the UK are vegans [31]. This has pushed research into finding meat alternative, and many companies to make inclusive meals, like a vegan sausage would use pea protein [57].

(a) Non dairy milk (*from Veganbaking.net from USA on Wikimedia*)

(b) Vegan sauage rolls (*from Open Food Facts*)

Figure 2.3: Examples of vegan food

The student life cannot be ignored. Young[2] adults and children, who require the health and nourishment to learn, often let their diets suffer as it is in relatively lower priority to deadlines. The main factor is time and energy to keep a diet. A day plan for a student may be cereal for breakfast, meal deal for lunch and microwave meal for dinner.

2.2 Eatwell Guide

The National Health Service (NHS) is a unified publicly funded healthcare system in the United Kingdom. Given the eating habits in the country discussed in 2.1, the NHS suggests a guide that encourages people to "eat well" by showing how many portions of each food group should be consumed for a healthy, balanced diet [107, 108]. The guide, previously called "the Eatwell Plate", includes instructions and food recommendations for the main essential food groups.

Fruits & Vegetables

Being excellent sources of vitamins, minerals and fibre, at least 5 portions of fruits and vegetables [1] should be consumed everyday making up over a third of the food eaten in the whole day.

Carbohydrates

Starchy food (such as potatoes, bread, rice and pasta) should also make up a third of the food eaten similar to 2.2. Meals can use this food group as the base – like pasta and meatballs, baked potatoes, sandwiches – therefore being the main source of nutrients and energy in one's diet. [102]

[2]Mostly, but not necessarily.

Dairy

Some dairy (milk or alternatives) is important for calcium, keeping one's bones strong [29]. It should ideally make up about 10% of daily intake, also avoiding products high in saturated fat.

Protein

As discussed in 2.1.3, consumption of meat is high in the UK, however there are alternatives such as beans, peas and lentils, that are low in fat. The NHS recommends to have at least two portions (each of 140g) of fish a week - one of which would be oily [108].

Oils & Fats

Only essential levels of fats should be taken, preferring unsaturated fats (such as vegetable oil instead of butter) as it helps reduce cholesterol in the blood.

Fluids

The instructions for 8 glasses of water a day is popular and should be followed everyday. Juices and smoothies also contribute to fluid consumption, but should be limited to 150ml a day. [116]

Sugars

Foods high in sugar (like cakes & chocolates) are not needed in the diet, but if included, should be in small amounts, since they have high calorie, and also affect parts of the body, such as teeth.

2.3 Seeing a Professional

The downside of the Eatwell Guide is that it is a general guide for everyone and hasn't been tailored for individuals - ones who may be of different age groups needing supplements, and therefore a professional can be approached to get a specific plan.

An expert who can provide guidance and prescriptions on eating habits is a **dietitian**. This may be confused with a nutritionist - the difference is that a dietitian is licensed and receive the title based on a classification by the World Health Organization [72]. When seeking professional guide, there are likely to be follow ups, and accurate information is essential.

2.4 The Client

This project has been proposed by Dr. Oana M. Andrei - a lecturer at the University of Glasgow and the supervisor for this project - who has seen individuals logging their portion intake on paper, wishing to have a digital alternative. These logs are useful for dietitian visits (discussed in 2.3), but difficult to share on occasions.

From the example scan provided by Dr. Andrei (see 2.4), a grid table can be seen with crosses (checkboxes) where each cell is supposed to be one portion. Therefore, in the first row, C (Carbohydrates) has 6 cells i.e. 6 portions. A full portion consumed is denoted by a cross, whereas half would be a slash. If consumption goes over the planned limit, the logs are added anyway (without the cells). In the lower half, there are notes and records maintained about what was eaten during the day as meals, like on Tuesday 20/04, for snacks (between breakfast and lunch) there were grapes, nuts and cheese, and on Wednesday 21/04, bread, cream cheese, butter, honey, strawberries and tea for breakfast. More scans are in the appendix (A.2) along with the initial example displayed at the first meeting (A.1), discussing the idea.

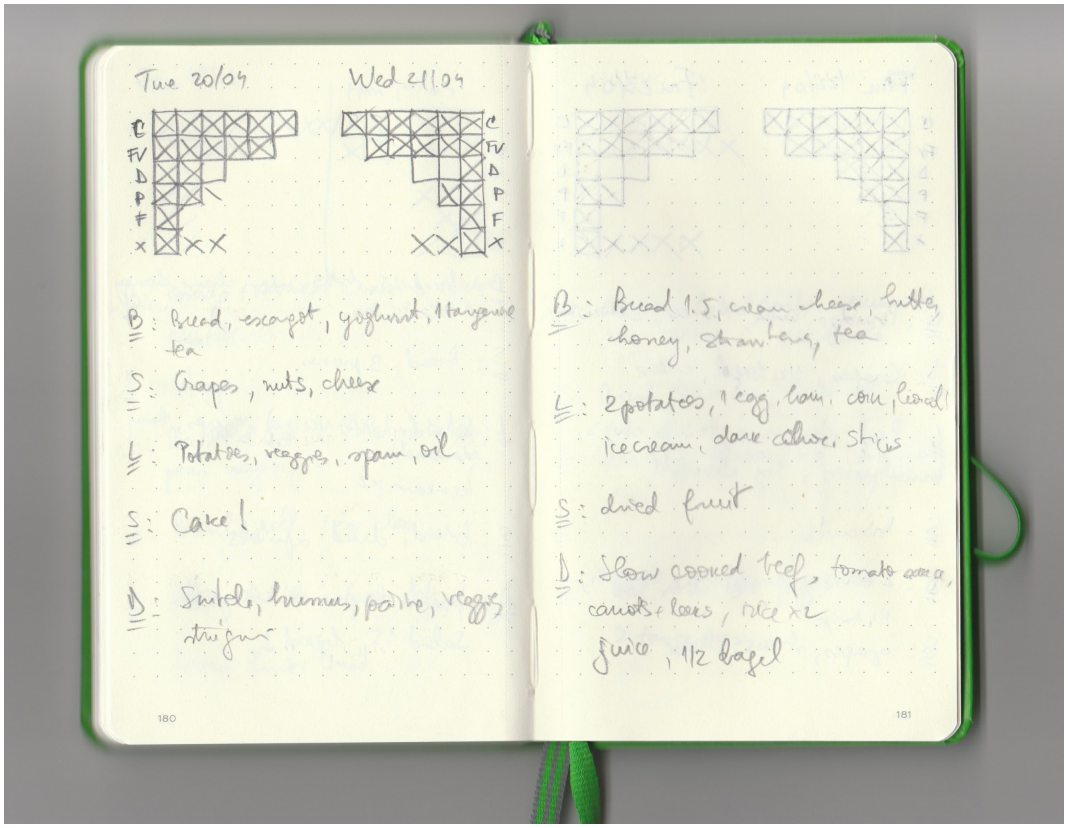


Figure 2.4: A scan of an example paper log

2.5 Existing Apps

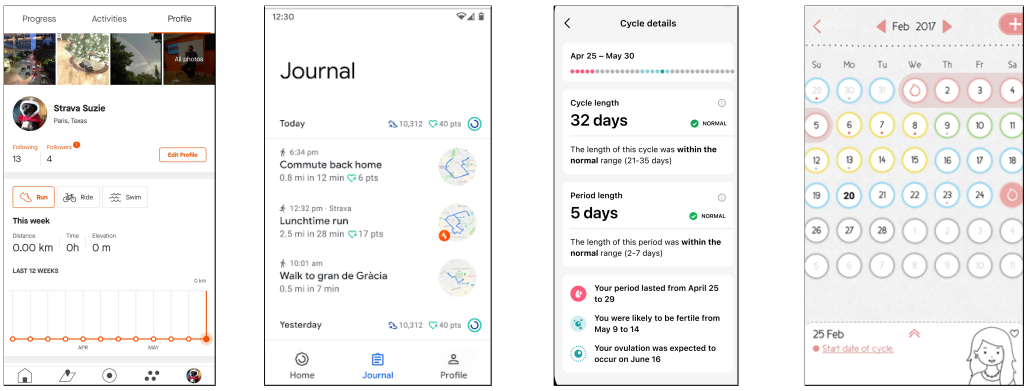
Currently existing applications were briefly discussed and introduced in section 1.2. At the start of the project, there was a study conducted thinking why would one not use the competition instead [147]. This allowed research on features providing inspiration, downsides that would cause user frustration, and it was not limited to portion trackers.

Fitness Trackers

Strava records data for physical exercise activities (mostly cycling and running, that also use GPS data), incorporating social network features like sharing and liking ("*kudos*") an activity. Activities in groups would also be automatically be displayed together when the system notices same place and time.

However, Strava also has an option to add activities manually - through importing from other device/services (like a Garmin), or just the statistic data you would know (like pace for a run), and during this, the form could also adjust for different sports. Strava does not log food intake.

Google Fit is Google's approach of a fitness tracker, released in 2014. It uses the simple, minimal Google interface design and layout which should be aimed for. The application provides its own means of logging and using sensors in a device, but it heavily focuses on providing APIs in order to be the general, overall application that would aggregate data from other device and services.



(a) Strava **(b)** Google Fit **(c)** Flo **(d)** Maya

Figure 2.5: Preview of tracking applications

Period Trackers

Flo released in 2015, focusing to track menstruation in women. It supports users through each stage of the cycle by notifying (push-notifications) them about predictions, and helping in any two approaches - tracking the cycle, or tracking fertility. This would be similar to what this project would aim for - providing notifications to remind users, and allowing users to track progress to lose/maintain weight, or increase for bulk/BMI.

Maya also supports women by tracking cycles, but also by providing forums and resources. The application includes a calendar view that can display visualisations & graphs.

Portion Trackers

Fitbit App is very popular for being the companion application for devices from the company, however it aims to provide a lot more functionality. There are resources, including videos, that can help users in fitness. More importantly, the application allows food tracking; the dashboard is highly customisable so the option can be removed if required. However, the tracking allows items to be added easily by scanning any QR codes that a packaged food would have.

Baritastic specialises in the branch of medicine that deals with the causes, prevention and treatment of obesity ("*bariatric*") and therefore the application is tailored for patients of the same. The interface provides a graph with card view, along with an appropriately sized action button in the bottom-middle that can allow certain tasks. It includes reminders and journal logging.

Portion Monitor from Tiny apps appears to be a very simple, basic and possibly outdated (last updated October 2019), but the dashboard includes a main table where the logging can occur. The rows/x-axis of the table are food groups (carbohydrates, protein) and columns/y-axis are the meals (breakfast, snack1, lunch). It can also display graphs for a time period.

Lifesum Health App is a paid application that specialises in food track with a range of fancy functionalities such as a food database, barcode scanner, and meal ratings. An interesting feature here is the list of recipes the application can provide that are in compliance with the diet plan.

Summary

Food has many aspects and backgrounds. In the UK itself, there is history and variety to meals available to people, ranging from all vegan diet to a high meat consumption. Some of it would include food that may be unhealthy for regular consumption as they include ingredients that should be taken in moderation, and therefore the National Health Service (NHS) provide instructions through the Eatwell Guide that layouts the essential food groups and the appropriate amount for a healthy, balanced diet. In specific cases where one may require professional help, dietitians are helping people to have healthier diets by prescribing diets tailored to individuals. Those who wish to aid this, log their intake may be doing it on paper, or other applications such as Google Fit, Fitbit or Lifesum, that vary a lot in features. These differences and features were studied, and along with the motion of promoting healthier diets, an overall application, that *checks all boxes* and tailors to this purpose by helping and motivating users, would be inspired.

3 Requirements

This chapter discusses the problems, tasks and issues of the project that are intended to be solved. The introduction and reasoning of each requirement, mainly classified as functional & non-functional, would justify the expectations and the determine the outcome of the project. For each category, five significant requirements are listed, however, every requirement can be found on the repository, the backlog & boards following the structure discussed in 3.1 with Kanban [7].

3.1 Planning

As instructed by the supervisor, the project timeline was laid out through six blocks that would start and end at certain weeks[1]. While there was recommendation given for these blocks for progress checkpoints, further planning was done by utilising these periods for specific aspects of the development. The general timeline is of ~~26~~ 27 weeks, starting 27 September 2021 (when the project was assigned) and ~~25 March~~[2] 1 April 2022 (when this dissertation is supposed to be handed in). The backlog including issues would be distributed and worked on using these blocks.

Block 1

The *Planning Period* for the project was from weeks 1 (commencing 27 September) and 4 (commencing 18 October). This would include analysis, requirement gathering and design diagrams. Major research of the development strategy was during this block as the repository (further discussed in 5.1) was also being setup.

Block 2

The *Setup Period* was from weeks 5 (commencing 25 October) to 13 (commencing 20 December). The coding & implementation had started specially in the development environment setup that would use configurations for future tasks, and application skeleton was being formed after choosing the frameworks (discussed in 5). All cards on the issue board were linked to concrete tasks related to the repository.

Block 3

The *Development Period* was from weeks 14 (commencing 27 December) to ~~15 (commencing 3 January)~~. This was when the term at university was off, and therefore more hours spent on the project were expected. A lot of backlog and future tasks were laid out and being completed.

Block 4

The *Development Period II* was ~~from weeks 16 (commencing 10 January)~~ to 19 (commencing 31 January). This block is considered to be same as Block 3 and therefore integrated to believe Block 3 ended on week 19. At this point, main development of core functionalities were near completion, and more were being added and refined.

[1] A week commencing Monday.
[2] Date strikeout conveys change of project dates.

Block 5

The *Deployment Period* was from weeks 20 (commencing 7 February) to 23 (commencing 28 February). The application was rigorously tested and deployed onto a server in order to prepare for evaluations. Documentation and survey design (discussed in 6.2) were carried out.

Block 6

The *Writing Period* was from weeks 24 (commencing 7 March) to ~~26 (commencing 21 March)~~ 27 (ending 1 April). The final block wraps up the timeline and project with evaluations running and analysed, and the dissertation coming together.

3.2 Prioritisation

Based on the block an issue is assigned to, further priority planning and timeboxing estimation would be done using the MoSCoW method [59].

Along with this, GitHub provides default issue labels [62] that were used to classify an issue and determine prioritisation; for example a bug would be prioritised before an enhancement usually.

[M]ust have

These requirements are critical for the system, and non-negotiable. The user and system expect and require these functionalities. These requirements are to be worked and managed with the highest priority before any other task in a given timebox. Without these in place, the application would not be a completed output and considered a failure.

[S]hould have

These requirements are important, but not critical. They would add significant functionalities that can add a lot of value, but the system can still work without them (not depending). For example, storing user credentials as a session. Ideally, the system would have these completed, but they do not take priority over critical requirements in the timebox.

[C]ould have

These requirements are desired possibilities that improve user experience and should only be worked on when important requirements are managed first.

[W]~~on't have~~ ould be nice to have

These requirements would not be considered in the development of the project in the case of traditional MoSCoW approach. However, for this project, no requirement is being dismissed and being welcomed with an open mind. These would be listed as issues on the repository, and can remain open. It is understood that users vary with different use cases, backgrounds and purposes, and therefore the development believed to take note of every requirement.

3.3 User Requirements

With end-users first in mind, reasonable (and fictional) scenarios as personas and stories are written [149] which then helps list the user expectations from the application. The aim is to cause zero frustration to users when they are using the app, and so most requirements are considerate of this therefore given relatively higher priority. Similar requirements are discussed in 3.4.2.

[M1] The software must provide a screen to users where they can input their logs

This is the core functionality of the application as users would expect and need to log their portion intake, and it is therefore a core requirement. This screen must be easily accessible being the main functionality and not be a difficult task for users as it would rather cause frustration and encourage them to not use the app. Therefore, this screen should be the home screen / dashboard, and it should display the logs on the same page, instead of dedicating it just for input.

[M2] The software must be accessible and available from different platforms and devices

This ensures that users are not limited from access to the application, and being on mobile is critical as it is much more convenient than PCs (as discussed in 1.2) therefore not creating a burden as users would tend to have their mobile devices with them and then log their portion in a few seconds, at their convenience.

[S1] The software should allow users to visualise and summarise their logs

This means in forms of graphs and statistics, and it should motivate and allow users to be on track with their diet plans. The visualisation can be on a separate page if required, but should not be hidden away from users. The graphs should update immediately after any change in data occurs, like a new log being added or a previous log being deleted.

[S2] The software should provide users with more information and guidance

This provides users with extra details if they require it, allowing them to learn more and have the correct idea about their diet plan. The application would allow users to not require a dietition[3] and enable to pick a plan they desire. This could also explain more about the Eatwell Guide.

[C1] The software could provide an option to add more detail to logs

This is useful as users may want to elaborate and record further about their intakes and use that for the dietitian as well. Since the detail can be vague and different for users, this could be a simple text field and so it can act as diary entries for their meals, and like a diary, it could mention the meal. The system can automatically log based on food mentioned in the entry, but this could be difficult and require expensive parsing, therefore a simple alternative can be allowing logging on the same page so that user would not have to log again on different screens.

3.4 System Requirements

Similar to user expectations, there are expectations from the system generally that should also be considered [146]. Since developer experience is considered as well, the requirements would list positive and negative effects for them. It should be kept in mind that a requirement may also be broken down into sub-requirements therefore there may be glimpses of other tasks, and over time more requirements come up.

3.4.1 Functional

These requirements here outline technical specifications that the system should do in terms of functionality, keeping the system as the main subject. These were decided and adjusted while designing and developing with considerations of what would be feasible with a logical approach using the architecture (discussed in 4.2) and further prioritised according to difficulty.

[3]The app **cannot** replace professional consultation.

[M3] The system must be able to keep a track of logs.

This is essential as the application revolves around logs, and a log depends on the datetime. Users would expect logs to be used in a certain way with the entry. Therefore, the system must be able to use this and track logs according to their date.

[M4] The system must be able to associate data to persons

This is important to associate logs to a user, and also ensure privacy for the user. Authentication would need to be in the form of a register (sign up) and login (sign in) that allows a user to create an account and authenticate into the account respectively. Along with this, the system should ensure authentication is secure and users aid their security. This would include storing sensitive information such as passwords safely (as hashes), and also encouraging users to use strong passwords (by requiring special characters, minimum length and more).

[S3] The system should be compatible for platforms

This is similar, however, on the functional side, it means that the system should use technologies that can run on most platforms and allow development without having to do a lot of platform-specific configuration or code duplication. If the system is incompatible, it restricts access and development for the different circumstance.

[S4] The system should allow customisation of all elements and data

This allows users to change anything at their will and intention. For example, there may be logs entered accidentally or privacy concerns. Along with that, the interface being adjusted or changed according to user preference (light and dark mode). The system should not have any consequences to these customisations however, and only reflect that for users.

[W1] The system would be nice to filter certain data

This ties other specifications with additional functionality to manage and visualise data, such as logs, as they could be analysed to produce graphs. It allows searching through data as well through keywords, or sort based on a field. A lot of features could take advantage of this.

3.4.2 Non-Functional

These requirements are similar to user requirements mentioned in 3.3, but specify how a certain function should be achieved. They would have relatively lower priority unless it strongly affects development, and these were decided mostly during setup (mentioned in 3.1), as the phase relates and inspires on the development process so that it can be guided in the desired direction.

[M5] The system must not perform any loading on logging

This is to ensure latency remains minimal and not cause frustration to user, since logging is essential feature of the application. For this approach, background changes can be made like updating the log locally, but making an asynchronous request to the server. However, fail safes should also be in place, and if loading is required, users should be displayed and notified about the system status (heuristic 1, [67]), so a loading screen can be made visible for the brief moment.

[C2] The system could use modular components and areas

This promotes the system to use modular programming, as everything would be considered to be an object through object-oriented approach anyway, but it allows any area to be swapped out and changed if required without reworking anything else. This would be incredibly useful when overtime features and requirements would increase.

[C3] The system could have endpoints to provide desired data

This extends the system to further modules and enable separation of concerns. Modules can use the endpoints to get data and make use of it, like rendering a fancy interface or graph for users to see. This endpoint would have to be secure, and not require a lot of processing and complexity. Querying and filtering requests would be useful and nice to have.

[C4] The system could not require authentication repeatedly

This makes usage of the application convenient for user, as they could just open the app, login once for the first time, and then continue logging as usual without having to re-authenticate every time. This still means that security must be considered, and use mechanisms to ensure that the account logged-in is the authenticated user.

[W2] The system would be nice to include a setup for easy replication

This ensures that other developers that wish to run the application locally can get started as soon as possible, without spending much time on configuring their environment. The dependency and stack management should be compatible to be deployed anywhere and everyone; this would ideally benefit from automation so that developers would not have to manually start each module.

Summary

The timeline of the project was divided into five appropriate blocks, such as *Setup*, *Development*, and *Evaluation*, to enable focus on the respective requirements over the 27 weeks. These tasks would be established by understanding and relating to users & developers through classification, into User and System requirements (sub-classified into Functional and Non-Functional) defining expectations from the system to provide functions like no-latency easy logging, secure authentication and simple developer experience. Further, other estimations, like complexity of the task and the popular MoSCoW method, would be used to prioritise and allow the development to make progress as planned with minimal obstruction and uncertainty.

4 Design

This chapter builds on the problems established in the previous chapter, and provide an overview of the final solutions for them, in terms of high-level technical and graphical design (architecture and interface respectively), along with justification for each decision.

4.1 Considerations & Principles

As part of developing this project from scratch, there would be considerations to be made therefore establishing principles that all processes would follow. These would be in compliance with the requirements to help create a better output.

4.1.1 Compatibility

The application can be developed to be native, web-based or hybrid. With a native application, the size and performance would be highly optimised for the platform it is written for. However, this means that supporting multiple platforms would result in different codebases with different languages, even though the logic would be same. For example, many organisations write native apps for Android and iOS (such as the NHS COVID-19 App [24, 66]) that use Kotlin/Java and Swift/Objective-C respectively, and therefore there would either be separate teams, or require a lot of time to develop. Furthermore, it would be very difficult to keep the two consistent and free of bugs. A web-based application (commonly called "*web-app*") would only run on a web browser and use HTML, CSS and JavaScript. These are highly compatible as browsers would be available on most platforms, and just need to get web resources. However, access to browser could still be inconvenient and have low performance (especially in mobile). Therefore, a hybrid application is a mixture of both that allows applications to run on a browser, and also run natively. There are technologies and frameworks such as *Xamarin* (C#), *React Native* (JavaScript) and *Flutter* (Dart) that enable development of hybrid apps with one codebase.

4.1.2 Extensibility

Extending this project with new capabilities and features should not require a lot of work other than developing the extension itself. This would also mean that the system would need to be modular i.e. allowing other modules to attach to it and use the system if required. The modules themselves could also allow sub-modules to be added, but they would have to ensure that they are as independent as they can be, and encapsulate logic or concerns into the module itself. Using suitable architectures like REST (4.2.2) and Plug-ins (4.2.3) that would give an API, but also need to organise the codebase directory structure to workspaces.

4.1.3 Durability

To make sure that the system is durable in terms of usability, security and service, the solutions chosen should be future-proof. This means that all methods, utilities and architectures used must not be outdated. Using external dependency would mean that the points of usage of the dependency may introduce vulnerabilities. For example, if a bug is discovered in the dependency, then the application may have the bug as well. So dependencies would need to be on the latest or a long-term-support (LTS) version and the usage must be safeguarded.

4.1.4 Reusability

While the development has the privileged option of open-source that allows usage of existing, pre-developed packages that would be tested and used by a large community, every solution should not depend on these. Keeping a low number of dependencies would cause less problems, like durability as mentioned in section 4.1.3 or a dependency hell. Where possible, they could rather be avoided, and use own constructed implementation, providing lot more flexibility.

4.2 Software Architecture

The approach to write code would be through object-oriented programming where everything would be considered as an object (or module). There would also be some usage of functional programming (through lambda[1] functions and list comprehensions) to make code more concise.

The system architecture, however, acts as a blueprint providing abstraction of logic and complexity between components, and so it is important to consider the patterns that this project could use.

4.2.1 Model View Controller

The system would be required to be separated into three components - Model, View and Controller, each responsible for different aspects therefore enabling separation of concerns. It is important since the system is supposed to be extensible (as discussed in 4.1.2) with different modules and working with a database would mean isolating it from different concerns.

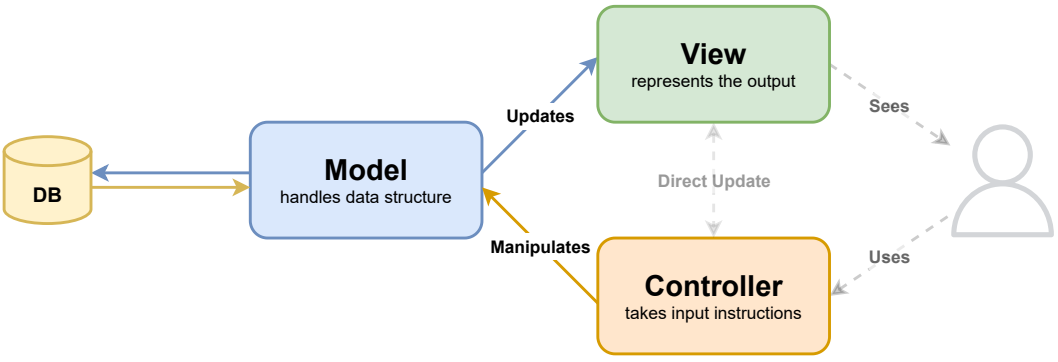


Figure 4.1: Sequence diagram for Model View Controller

[1] Also called anonymous or arrow in JavaScript.

The model, being the most important component of this pattern, requires designing the data structure which would be done with the help of a Entity Relationship diagram. This started with simple schemas of a User, "Nutrient" and their relation (figure A.3). Eventually, this was polished and finalised over development featuring important schemas (see 4.2). The relationships would use the primary keys from other schemas as foreign key. For many-to-one relationships, the primary key from the single schema is used as a foreign key in the "many" side.

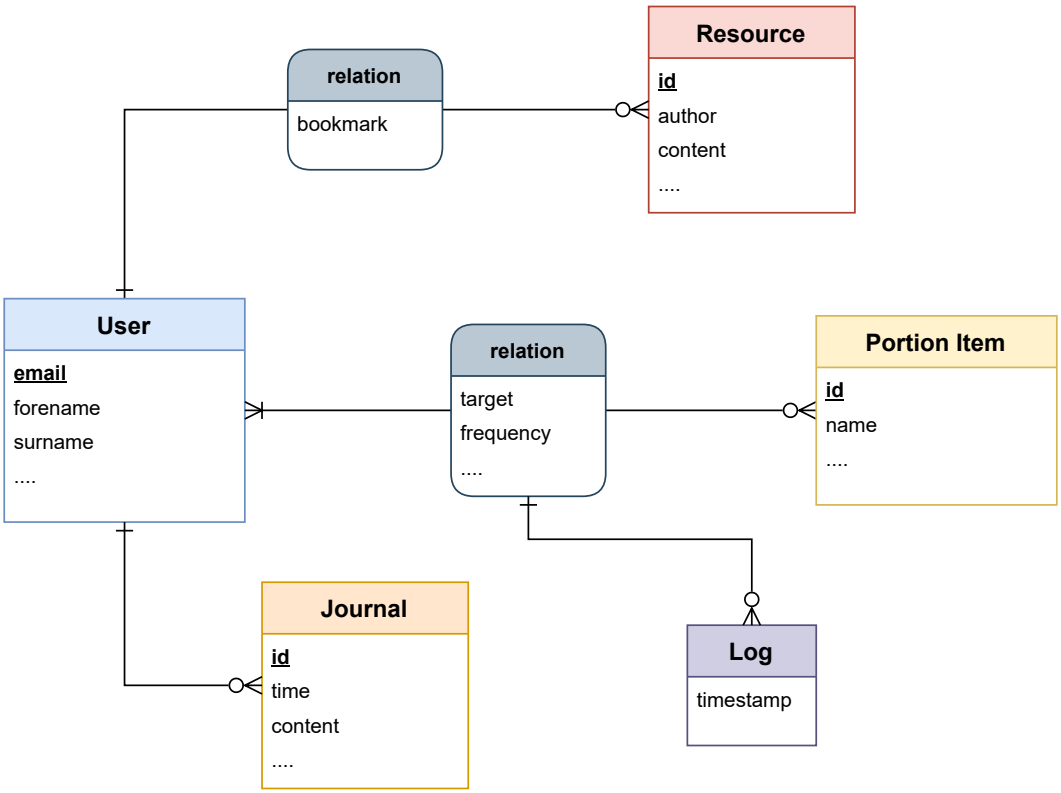


Figure 4.2: Entity Relationship model for the data

4.2.2 Representational State Transfer

It is understood that the server would be separated from client, and the client-side application could be running on multiple, different devices. The frontend interface would not deal with server logic, but would still need to exchange data at a point, like fetching previous logs or providing new ones. This would be achieved through the REST pattern (figure A.4) that allows the data to be sent and received over the World Wide Web[2] in JSON or XML format.

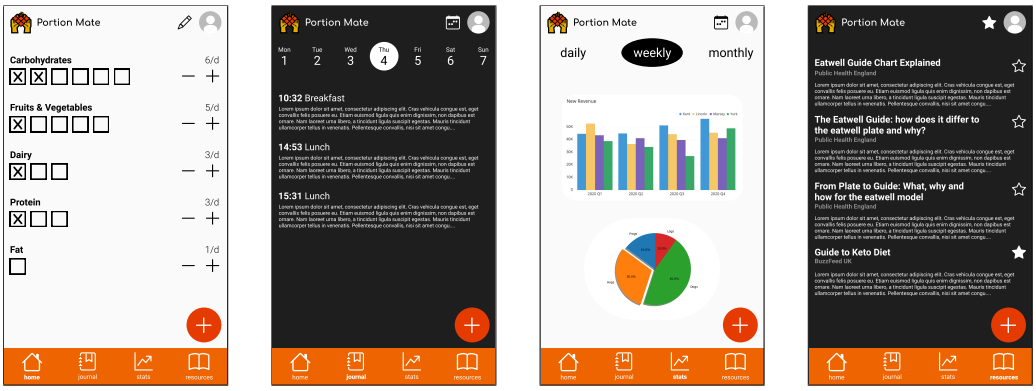
4.2.3 Plug-ins

Since there is expectation with separation of concerns, extensibility and modularity, the system can be developed in a plug-in manner where each functionality would be its own plug-in that can be added or removed from the main module ("*host application*").

[2]Using HTTP methods such as `GET`, `POST`, `PUT` and `DELETE` [39], provided users would have an internet connection.

4.3 Interface Design

The interface of the application is **critical** and **extremely important**. The user experience would majorly be determined by the design of the interface. The interface would be inspired from the paper log analysis in section 2.4, and be designed in Figma [83] first [150]. This wireframe would take two iterations to get on the current stage - the first did not make use of checkboxes and had repetitive text saying "X portions taken" (see A.5). However, on further brainstorming, the right-side buttons were retained, and checkboxes were added instead to provide additional methods of input. The interface is minimal (like material design), and used dark mode by default - this was switched to light mode when the supervisor preferred it. Aside from the ones in section 4.1, there would be considerations made for the interface specifically.



(a) Home Page **(b)** Journal **(c)** Statistics **(d)** Resources

Figure 4.3: Wireframes from `Portion Mate Figma` [83]

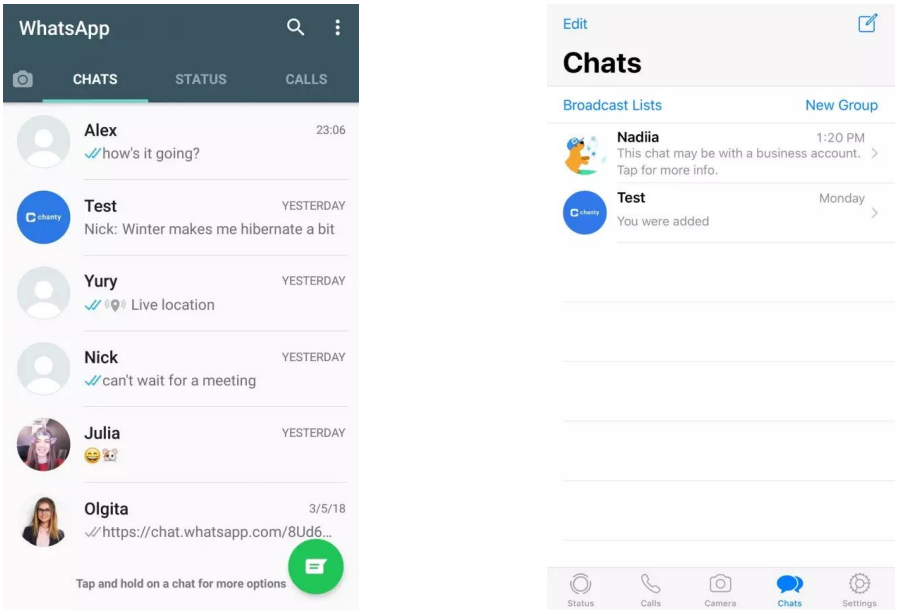
4.3.1 Consistency

The design would need to be consistent throughout the application and introduce no screen that looks very different from any others, else it would frustrate users to learn and understand a new layout, violating Nielsen's Heuristic 4 [67]. An app-wide colour scheme would allow the application give identity and make the screen similar. This would also include font selection, icons, and element sizing - all should be from the same family and use consistent, proportionate numbers. The interface must also be consistent for different screen sizes (responsiveness) and not change interface drastically for a specific breakpoint or platform, like WhatsApp has very different interfaces for Android and iOS[3] (see 4.4). The same goes for toggling between light and dark mode where the interface should not become unfamiliar on changing themes.

4.3.2 Accessibility

The interface would need to be accessible, otherwise it would limit users on their actions and cause frustration. There are guidelines that instruct and define rules to make accessible interfaces, such as W3C [2]. First, the basic elements need to be considerate and inclusive. The images and icons would require `alt` property that can be picked up by screen-readers as they read through each element and parse the HTML code. The font face should be legible and enable users to distinguish between letters; sans-serif fonts, like Arial and Verdana, are preferable. Roboto was

[3] Both operating systems, however, have their own design philosophies.



(a) WhatsApp on Android **(b)** WhatsApp on iOS

Figure 4.4: Very different interfaces for WhatsApp (*from Chanty*)

designed and released by Google in 2011 for Android [54], and has become popular for its clean and modern look. Differentiation should not depend on colours, but can also use other means of displaying hierarchy through text weight, positioning and icons. A responsive design would also enable the interface to be accessible on screens of various sizes[4]. As the application allows switching between light and dark mode, that can help with eye strain, but the ability to toggle should be easily available through the system (and saving the preference).

Summary

Users and developers are human, and since the project focuses on Human Computer Interaction, the design decisions were critical. Making these decisions would lay the foundation for the application as they define principles for the development process. Any written code should be modular so that it could be extended with more functionalities if required. The policy on external dependencies asks to use recommended versions, but avoid usage wherever possible. The architectural patterns that the technology would use include definition for the database model, and interfaces to interact with each component such as the server (from the client) and the database itself. More importantly, the interface design which would be the first element visualised by all users, and providing a method of using the application - so it needs to be inclusive and considerate, meaning that the design should be visually appealing and accessible through devices of different nature, following all guidelines and expectations. This interface would be refined from the wireframes that were constructed and prototyped before the final implementation.

[4]From 8K large displays to palm-sized 360p; watch faces would highly differ.

5 Implementation

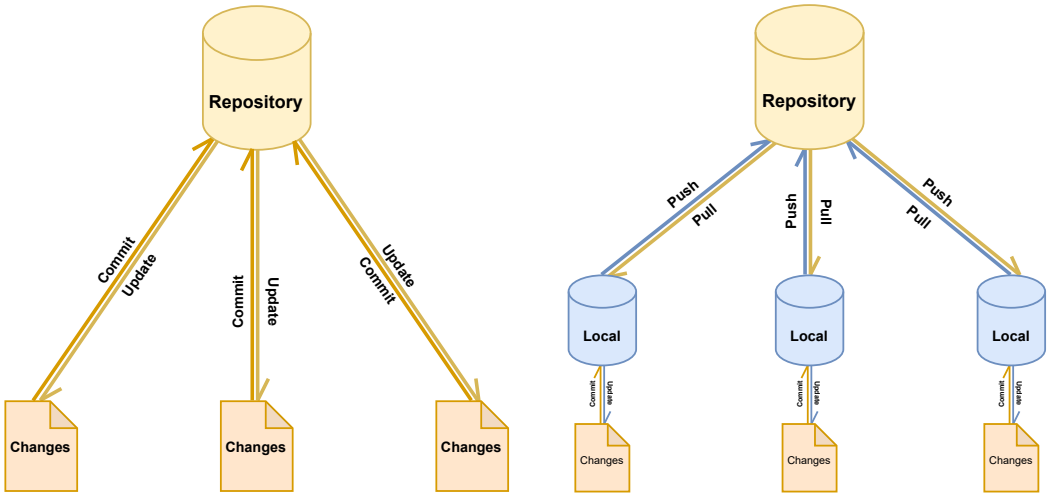
This chapter takes all pieces from the previous chapters and places them on the board to finally bring the system to life. Majority of the project involves implementation and writing elegant, functional code; some snippets are included. The main development environment was using *Visual Studio Code* on Windows 10 with essential clients, such as *Git* and *Node.js*, installed.

5.1 Version Control

Using version control was an important criteria for this dissertation. Despite being required, development would not have gone forward without it since it enables history, rollback, branching, and more, allowing development to be lot more flexible and less scary with many possibilities.

5.1.1 Git & GitHub

There are different version control architectures (such as local, centralized), but Git is the most popular version control system system and uses a distributed repository architecture providing advantages over other systems like resolving conflicts, easier branching and better performance.



(a) Centralised **(b)** Distributed

Figure 5.1: Differences between types of version control

The remote repository is hosted on GitHub, which is equally popular as Git and owned by Microsoft. The service provides hosting the remote repository and displaying it as public to allow other developers to view, fork and star it; in a way it is a social network for Git repositories.

The project was not public until Block 4, however. Additional features that GitHub offers for repositories on the platform include issue tracking, continuous integration and wikis [118]. The project has taken every advantage of all features, and has also developed extensions to make the process extremely streamline (more in side projects (7.3) and automation (5.3.3)).

5.1.2 Issue Branching

As issue tracking systems are industry standard, this project made sure to work with this strategy in the best possible way. It is mentioned earlier that version control (section 5.1) and Git (section 5.1.1) enable branching that allow development to be experimental, and so all issues were worked on separate branches (named after issues, like `issue/19-write_tests_for_api` or `feature/21-enable_reverse_logging`) so that the issue solutions can be experimental or dismissed, and the state of the repository can also be traced back easily. It avoided conflicts, and made review easier as each branch would then create a pull (or merge) request to the development trunk `develop`. This is heavily inspired by Atlassian products - Jira and Bitbucket - that integrate to provide issues and branching mechanism. Exposure to this system was received during an internship at New Verve Consulting in summer 2021. Pipelines were setup to enable the integration and automation (more discussed in 5.3.3). At the time of writing, the repository has over 60 branches (including `develop` and `master`), with 58 issues and 68 PRs closed.

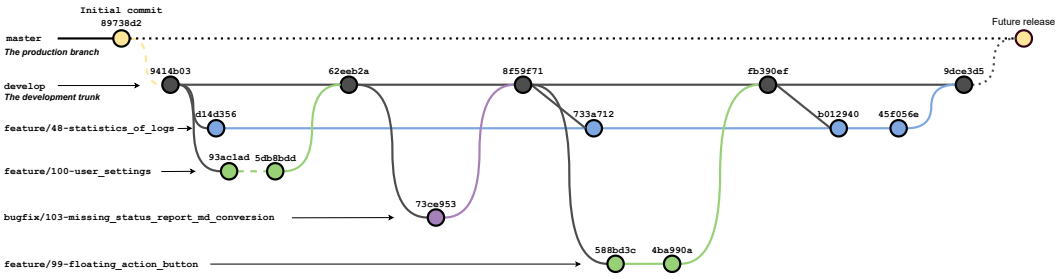
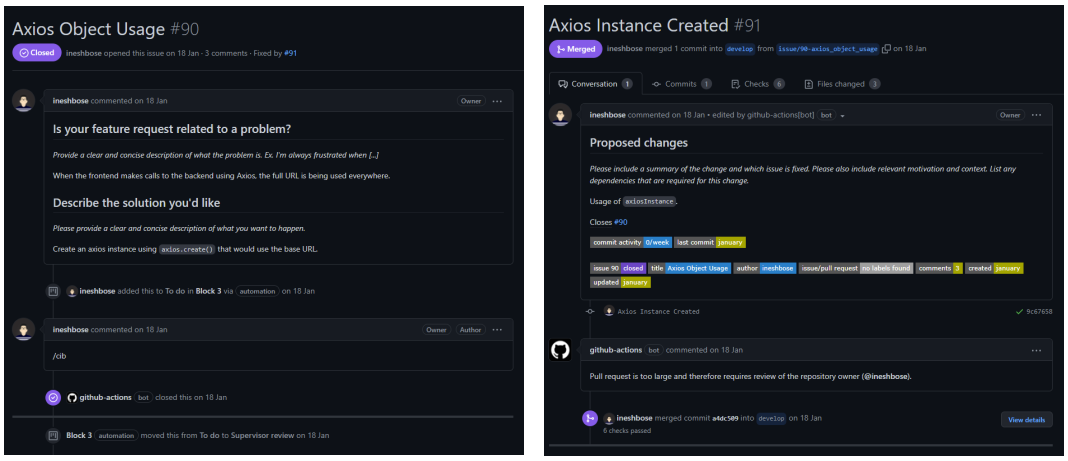


Figure 5.2: Branch network between 23/01/22 and 30/01/22



(a) Issue **(b)** Pull Request

Figure 5.3: Examples of branch management using issues

5.1.3 Hooks

Version control systems usually provide signals (hooks) when specific commands are run. With Git [44], a `pre-commit` hook has been setup that runs the code linters (mentioned in 5.2.1 & 5.2.2) before the changes can be pushed onto the remote repository (see `Issue #52` [137]). The scripts that would run on them could be setup using packages like `pre-commit/pre-commit` [86] and `typicode/husky` [112] - the latter being the most popular. However, husky has drawbacks such as creating additional configuration files and running an install script which may create inconsistencies for developer environments; these were not required in version 4 of the package (currently on v7 at time of writing), but the project strongly prefers packages at their latest versions, an alternate `yyx990803/yorkie` [122] was found (based on version 4 of husky, developed by the creator of Vue framework) that allows configuration to be in the `package.json` and not require installation of the script, but the package itself.

```
"gitHooks": {
  "pre-commit": "npx lint-staged"
}
```

Listing 5.1: `./package.json` [124]

5.2 Environment setup

As the foundation of the application, the development environment had to be configured keeping the requirements and future steps in mind. The major decision was with the languages that the stack will use that would require their own compilers and interpreters.

5.2.1 Python

A very popular programming language that many would have heard of by now. It is dynamically-typed and focuses on code readability. Learning Python is said to be easier than other languages.

Dependency Management

`Python` [90] includes the `pip` [79] package management system that sources packages from the Python Package Index (*PyPI*). This package manager, however, requires configuration that would normally be done manually using `pip`, like maintaing a virtual environment, updating the list of dependencies, and resolving peer-dependency version conflicts.

```
$ pip install -r requirements.txt
$ pip install requests
$ pip freeze > requirements.txt
```

Listing 5.2: example of `pip` usage

These issues have been addressed and resolved by *Poetry* (see `Pull Request #69` [143]) that maintains the dependencies in `pyproject.toml` and provide ways to make the project compatible like switching Python version easily and listing dependencies in a global way. Another similar solution is *Pipenv* that has some different pros and cons, but the development of Poetry is promising with the support of plugins in release version 1.12 [82].

Code formatting

Python offers many different ways to write the same code, and personally, list comprehensions and inline conditions (one-liners) seem impressive, but they could make the one line of code very long and difficult to understand. To ensure that code remains consistent and readable, *Black* is a formatting tool that works in compliance with PEP8 [96] and also produces small diffs to make code reviews faster - useful for the development strategy (discussed in 5.1.2).

5.2.2 TypeScript

A syntactical superset of JavaScript that enables static typing, but then transpile to JavaScript to run on supported platforms. Having type definitions makes the code reliable and predictable. It auto-generates documentation for type-annotated variables and can simply be used like JavaScript with Node.js, by installing the package as a dependency.

```
export type CreateData<
  T extends GenericModel | ModelID = GenericModel,
  R extends keyof T | string = 'name'
> = Partial<Omit<T, 'id' | R>> & {
  [P in R]: R extends keyof T
    ? NonNullable<T[R]> extends GenericModel | ModelID
      ? CreateData<NonNullable<T[R]>> | ModelID
      : T[R]
    : any;
};
```

Listing 5.3: recursive & conditional type definition in `./src/app/types/api.ts` [130]

Dependency Management

The default package manager for `Node.js` [69] is `npm` [70] and it lists and updates dependencies in `package.json` automatically. In 2016, Facebook developed an alternative dubbed "Yet Another Resource Negotiator" popularly known as *Yarn* which provides better speed and stability than `npm` [71]. Because of this, the package manager started to be preferred by many in the community. A few years later, the development for Yarn switched to a different team therefore releasing a rewrite called *Berry*, keeping version 1 as "Classic". Yarn Berry changed the traditional approach of resolving dependencies through a feature called "Plug'n'Play" [68, 91]. At the time of writing, the latest version of Yarn is 3.2.0. Since the approach for modern Yarn is unconventional, the project stuck to classic (see `Issue #44` [136] and `Pull Request #83` [145]). Additionally, Yarn also has a "workspaces" feature that allows the repository to use the monorepo strategy [58, 48, 15, 85] and divide the codebase into modules & packages (see `Issue #71` [139]).

Code styling

The majority of the codebase is expected to be `TypeScript` [111] (which also enforces style rules onto JavaScript for type definitions) that would include React syntax (file extension `.[jt]sx`). All TypeScript code would be analysed using `ESLint` [36] that supports ECMAScript standards to standardise and encourage compatibility [104, 119]. It is the most commonly used JavaScript linter, being downloaded over 14,000,000 times per week. As mentioned, since the code is not vanilla JavaScript, additional configurations had to be added to ESLint.[1]

- `Prettier` [87] is an opinionated formatter (similar to *Black* for Python - discussed in 5.2.1) and extends the formatting rules in ESLint with its style opinions;
- React is unopinionated, however a plugin for ESLint allows parsing of the HTML-like syntax and provide recommended React specific linting rules;
- TypeScript plugin enables ESLint to analyse `.tsx?` files;
- Airbnb defines a JavaScript Style Guide [5] which is preferred by the community and can be setup easily along with ESLint (through `eslint --init`).

5.2.3 Stack Specific

ngrok

Since the application needs to be tested on mobile devices while running locally, the API calls are normally made to `localhost` which would not work on different devices. Therefore, the project uses a very unique, automated setup with *ngrok*[2] making testing incredibly easy.

```
const start = (file = `${__dirname}/.env`) => {
  const env = dotenv.parse(fs.readFileSync(file, 'utf-8'));
  const port = getPort(env.API_BASE);
  if (env.DEBUG.toLowerCase() === 'true' && port) {
    ngrok.connect(port).then((value) => {
      env.API_BASE = `${value}${env.API_BASE.split(`${port}`)[1]}`;
      writeToFile(env);
    });
  } else {
    writeToFile(env);
  }
};
```

Listing 5.4: Automatic ngrok connection in `./src/set-env.js` [132]

5.3 Technologies

5.3.1 Django server

The backend server of Portion Mate was developed using the *Django* framework, written in Python, which follows the MVC architecture pattern [33, 51] described in 4.2.1. The principle

[1] File extensions written are in Regular Expression (RegEx)
[2] Does not use Capitalisation.

of this framework is **don't repeat yourself**. The framework advertises with "Batteries included" which means that it comes out of the box with libraries and utilities that could potentially be required for the development of the server, like authentication system and database connection. This could also be a downside for Django, since it makes the package bigger with modules that may be required; in that case, *Flask* is an alternative. For this project, however, Django was a suitable choice. The framework arguably also follows the plug-in architecture (discussed in 4.2.3) as it treats applications as module plugins that can be simply installed or removed through the configuration in the `settings.py`. Because of this, applications for Django can be distributed easily (through PyPI discussed in 5.2.1), and the project takes advantage of a few.

REST Framework

As discussed in 4.2.2, the project would use a different technology for the frontend application, therefore to build a Web API, *Django REST framework* was adopted. It extends Django and provides with utilities like serializers, authentication policies and a browsable API that would help visualisation JSON data. In addition to the REST architecture, Portion Mate attempted to implement a GraphQL API for more functionality (see `Issue #73` [140]).

OAuth

Open **Auth**orization is an access delegation mechanism used by many major companies for their applications. Users would need to access their information such as logs, these would be provided by this method. Since the project uses API calls between the Django server and the React application, the access occurs through token (but not to be confused with JWT mechanism), which would be given on login and have an expiry.

```
{
   "access_token": "<your_access_token>",
   "token_type": "Bearer",
   "expires_in": 36000,
   "refresh_token": "<your_refresh_token>",
   "scope": "read write groups"
}
```

Listing 5.5: Example of a web access token

In HTTP requests, the token is added in the header as `Authorization`, therefore authorising requests to the user, and on expiry, these tokens can be attempted to be refreshed.

PostgreSQL

Django uses Stuctured Query Language (SQL) for database, and one would not have to write any SQL by themselves, but use functions provided by Django. The default engine for the framework is SQLite that would normally keep data in a local file (`db.sqlite3`), however it may introduce security vulnerabilities and problems in deployment since processes would have different versions of the file. PostgreSQL is an advanced and extensible alternative that runs on a port instead.

5.3.2 React Native app

The principle of this framework is **Learn once, write anywhere** - that translates to *don't repeat yourself* (section 5.3.1) in writing code, programs and applications in different languages / technologies to support native applications. Instead, React Native aims to use one codebase - written in JavaScript/TypeScript, following React syntax - to support multiple platforms.

Expo

While React Native allows development of applications using React, there would be places where native development could be essential (like creating a `build.gradle` to publish on Android). However, *Expo* removes the need for all of that and would automatically build and bundle (along with Metro) applications for cross-platform development.

Axios

Since the frontend application needs to make API requests to the server, axios[3] client was used since it automates many elements that `fetch` (default HTTP client) fails to do (like transforming JSON data). It supports usage of Promises in JavaScript (making usage of `async` and `await` in functions) and more versions of browsers than `fetch` for compatibility.

The application uses and provides a consistent, uniform and easy integrated usage of axios by using a created instance that would keep user's `Authorization` header (along with an attached interceptor for refreshing / revoking token) and methods that reduce code duplication.

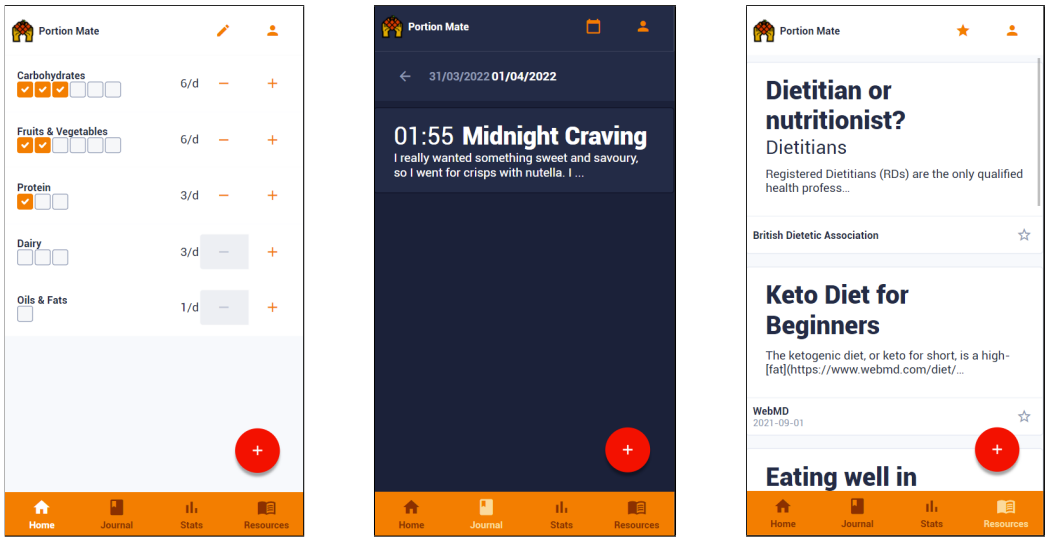
```
export const axiosInstance = axios.create({
  baseURL: API_BASE,
});

// ...

export async function updateData<T extends GenericModel = GenericModel>(
  path: string,
  method: Extract<Methods, 'PATCH' | 'DELETE'>,
  data: UpdateData<T>
) {
  const { id } = data;
  const url = `${path}${id}/`;
  const response = await axiosInstance.request<T>({
    method,
    url,
    data,
  });
  return response.data;
}
```

Listing 5.6: Creation and usage of axios instance in `./src/app/api/index.ts` [127]

[3] Does not require Capitalisation, based on documentation.

UI Kitten

The user interface becomes the most important aspect of the application. To ensure development was easy and not requiring a lot of time and effort on just the interface, a UI component library by Akveo was chosen, and the components in this library adopted the *Eva Design System*, so a theme was followed throughout the application and keeping it consistent (code in B.1). As seen from figure 5.4, it also made the final product similar to the wireframes (4.3).



(a) Home Page **(b)** Journal **(c)** Resources

Figure 5.4: Final implemented interface

5.3.3 Automation

It is highly desirable and preferred to have many processes automated as manual process require a lot of time and accuracy, making it error-prone and expensive for development. As discussed in section 5.1, pipelines and hooks were taken advantage of.

Pipelines

GitHub Actions is an incredibly easy and modular approach to setup continuous integration and delivery. Each configuration for an action is called a workflow, and a workflow can have multiple jobs; these must be defined in YAML (example B.2). The project defined 12 workflows (figure 5.5) at the time of writing, each for different purpose which would include testing code functioning & styling, and the repository issue management (as mentioned in 5.1.2).

Services

Additionally, the project also made use of services that aid software development by providing unique features like static analysis, free hosting, or online compilers.

- ***Code Climate*** judged the maintainability of the whole codebase and give a suitable grade and estimate it would take to fix issues. At the time of writing, the report gives the repository A grade for maintainability with 0 minutes required to fix issues after breaking down 250 files, finding 0 code smells, duplication and other issues.

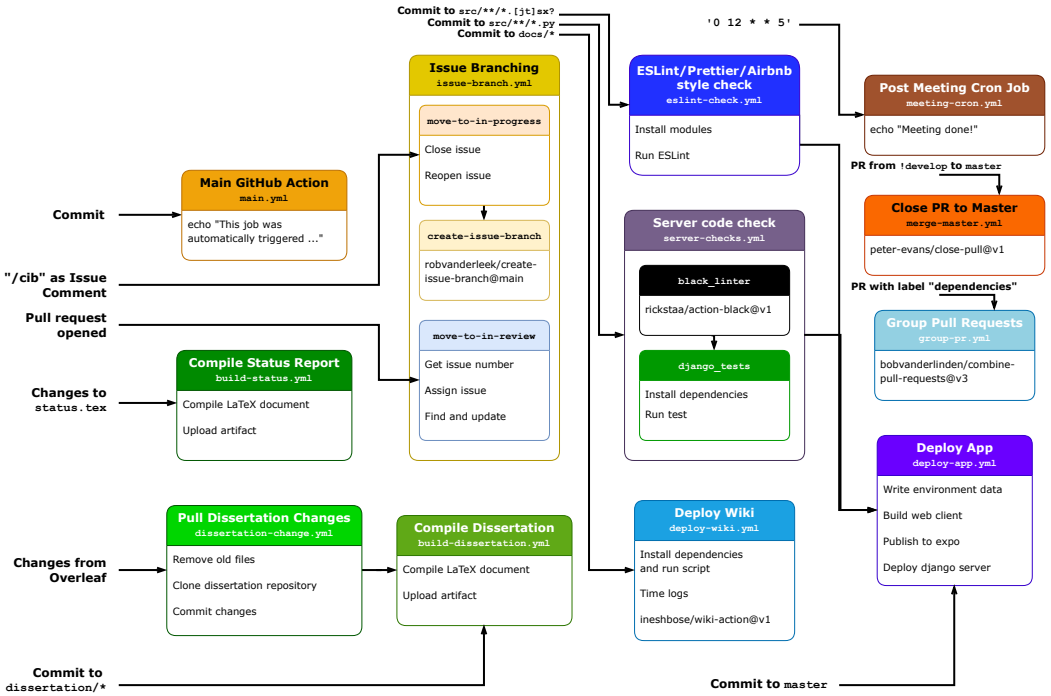


Figure 5.5: High level flowchart for the configured pipelines

- ***Codacy*** also looks for issues in the repository and grades the code quality. At the time of writing, the grade given is an A with four issues indicating three class methods that the analyser feels could be defined as independent functions instead and a super-method being invoked in an unpreferred way.
- ***Read the Docs*** allowed the documentation for the application to be build and hosted on the platform, making use of Mkdocs that converts the Markdown files into HTML.
- ***Overleaf*** enabled development for reports and documents[4] in TEXthrough the online editor environment that manages packages and compiles the source.

Other services include *Snyk*, *Libraries.io*, and *Dependabot* for secure dependency management, along with potential usage of *AppVeyor* and *Travis CI* for further testing.

5.4 Challenges

With such an intense development in place, a lot of challenges were faced. All were listed as issues on the repository, so those that are not yet closed may not have been solved, and some also had solutions that were not as good as desired.

Setup

First, while setting up the repository, the workflows had to be configured and there was not a lot of experience with YAML, and it was the first time using GitHub Actions. Since this feature was also released a few years ago, there are still plenty of solutions in development aside from

[4] Including this dissertation.

the 12,557 actions already listed on the GitHub Marketplace. This includes the full integration of issue branching (from creating a named branch to closing a linked pull-request, all while automating the linked card on the Kanban board). Fortunately, a lot of research, study and determination enabled the setup in time for main development.

Workspaces

At the beginning, the repository had separate directories for the Django server (`src/backend`) and the React app (`src/frontend`). Working on them always required having to change directory, and managing different directories was becoming complex (see `Issue #71` [139]). It was important to have workspaces as the approach for directory organisation was a monorepo (discussed in 5.2.2). The dependency management Poetry (for Python) had limitations of the features than Yarn (for Node.js) offered, like workspaces. This was turned around by reading documentations and having discussions on their support channel, eventually defining sub-packages `"portion-mate-src"` & `"portion-mate-docs"`, and updating the lock file through hooks, therefore restructuring the repository midway through the project (see *Pull Request #79*). Overleaf's editor also had limitations like not supporting symbolic link ("*symlinks*") or resolving conflicts, and therefore the `dissertation` workspace had to be exported to a separate repository, but continue reflecting changes on the monorepo (see `Issue #70` [138]).

```toml
[tool.poetry]
name = "portion-mate"
version = "1.0.0"
description = ""
authors = []

[tool.poetry.dependencies]
python = "^3.8"
portion-mate-src = {path = "./src"}
portion-mate-docs = {path = "./docs"}
```

Listing (5.7) Including the sub-packages as dependencies in `./pyproject.toml` [125]

```json
{
  "name": "portion-mate",
  "version": "1.0.0",
  "main": "src/index.js",
  "private": true,
  "workspaces": [
    "src",
    "docs"
  ], ...
}
```

Listing (5.8) Including the directories as workspaces in `./package.json` [124]

Figure 5.6: examples of Card components

Bootstrapping

For the React Native app, there were greater difficulties. Despite following the traditional TypeScript React approach, there came another learning curve for React Native as it does not offer all features of React, but also requires testing on more devices. Expo (see 5.3.2) has been very helpful, however, a critical functionality was the ability to use CSS in React to style elements.

Most code written deals with components and interface layout. Styling libraries provide methods to make design and development easier, for example *Bootstrap* is one of the most popular frontend frameworks that provides CSS classes like `navbar`, `float-start` and `bg-primary` that would set and style elements accordingly (as the class names would suggest).

```
<div class="card" style="width: 18rem;">
  <img src="..." class="card-img-top">
  <div class="card-body">
    <h5 class="card-title">Card title</h5>
    <p class="card-text">Some quick example text to build on the card title and
        make up the bulk of the card's content.</p>
    <a href="#" class="btn btn-primary">Go somewhere</a>
  </div>
</div>
```

Listing 5.9: Example of a Card element as documented on Bootstrap v5 [16]

For component-based frameworks like React and Vue, such libraries also get extended to provide additional functionality. In this example, *Bootstrap* would have *React-Bootstrap* (for React) and *BootstrapVue* (for Vue) where, instead of using classes, components are predefined with lots of additional configuration options using events (`onClick`, `onHover`) and property parameters.

```
<Card style={{ width: '18rem' }}>
  <Card.Img variant="top" src="..." />
  <Card.Body>
    <Card.Title>Card Title</Card.Title>
    <Card.Text>
      Some quick example text to build
      on the card title and make up the
      bulk of the card's content.
    </Card.Text>
    <Button variant="primary">
       Go somewhere
    </Button>
  </Card.Body>
</Card>
```

Listing (5.10) React-Bootstrap [18]

```
<b-card
  title="Card Title"
  img-src="..."
  img-top
>
  <b-card-text>
    Some quick example text to build on
    the card title and make up the bulk
    of the card's content.
  </b-card-text>
  <b-button href="#" variant="primary">
    Go somewhere
  </b-button>
</b-card>
```

Listing (5.11) BootstrapVue [17]

Figure 5.7: Examples of Card components

Unfortunately, this does not translate for React Native, since most of the utilities have been made possible through CSS classes and mobile development does not use HTML or CSS [106]. There are limited components available for native development, like View, Text and Image, and these can only be styled using certain allowed properties, like margins, border and flex (similar to CSS but not the same). This would pose lots of difficulties in the interface development. The library mentioned in section 5.3.2 helped and aided development, and it was adopted in the middle of development (see `Issue #93` [141]). However, at certain points, extra custom styling had to be added and that also seemed to be repetitive, whereas Bootstrap has a simple class mechanism. Inline styling is not good practice, and for React Native also causes performance issues [106]. Therefore, a React Native equivalent for Bootstrap was developed so that components can use the defined "classes" (see 5.8, `Issue #116` [133] and `Pull Request #120 Files` [142]).

```
const PROVIDED_STYLES = <GlobalStyleSheet>{
  ...spacingStyles,
  ...displayStyles,
  ...flexStyles,
};

export default function createStyle<
  T extends StyleSheet.NamedStyles<T> | StyleSheet.NamedStyles<any>
>(styles: T | StyleSheet.NamedStyles<T>) {
  return StyleSheet.create({ ...PROVIDED_STYLES, ...styles });
}
```

Listing 5.12: Custom style function in `./src/app/constants/Styles/index.ts` [128]

Summary

The project promised and delivered on a system that would be highly maintainable, modular and robust using the best practices throughout the process, even if that meant implementing solutions from scratch, such as automating issue branches in GitHub through pipelines or dynamically generate style properties for components. This went above and beyond the application, which itself was developed using React Native in TypeScript, accompanied with Expo to distribute on different platforms, and the backend database server using Django, along with PostgreSQL, extended by Django REST Framework to provide API to the frontend clients.

```
<div style="display: flex;">
  <div
    style="
      display: flex;
      align-items: 'center';
      justify-content: 'center';
      padding: 2;
    "
  >
    <!-- Spinner CSS is complex -->
    <div style="..."></div>

    <h3>Loading</h3>
  </div>
</div>
```

Listing (5.13) using inline styles

```
<div class="d-flex">
  <div
    class="
      d-flex
      align-items-center
      justify-content-center
      p-2
    "
  >
    <div class="spinner-border"></div>
    <p class="fs-3">
      Loading...
    </p>
  </div>
</div>
```

Listing (5.14) using Bootstrap classes

```
<SafeAreaView style={{ flex: 1 }}>
  <Layout
    style={{
      flex: 1,
      alignItems: 'center',
      justifyContent: 'center',
      padding: 2,
    }}
  >
    <Spinner size="giant" />
    <Text style={styles.title}>
      {'Loading...'}
    </Text>
  </Layout>
</SafeAreaView>
```

Listing (5.15) usage with inline styles (through objects) causing increased complexity (bad practice) [106]

```
<SafeAreaView style={styles.flex1}>
  <Layout
    style={[
      styles.flex1,
      styles.alignItemsCenter,
      styles.justifyContentCenter,
      styles.padding2,
    ]}
  >
    <Spinner size="giant" />
    <Text style={styles.title}>
      {'Loading...'}
    </Text>
  </Layout>
</SafeAreaView>
```

Listing (5.16) example of usage like Bootstrap in `src/app/navigation/RootNavigator.tsx` [148]

Figure 5.8: Effect of Bootstrap, in HTML code (above), and React **Native** code (below; `display:'flex'` is not native)

6 Evaluation

This chapter shares methods on how the final implemented application was judged for being a qualified, established product that the previous chapters discussed and the development aimed for.

6.1 Testing

The first step of testing the project was by using TypeScript. The languages chosen are runtime, and therefore all bugs and errors would not be detected and caught during compilation and main execution. TypeScript, however, has helped a lot by eliminating the problem of dynamic types in JavaScript, and wherever functions are called or values are updated, it is made sure that the desired datatype is checked and used.

```
const itemCheckBox = (item: TrackItem, idx: number) => (
  <CheckBox
    style={styles.checkbox}
    key={idx}
    checked={idx < (item.logs as UserLogs).length} // prop is of type UserLogs
    onChange={(checked) => updateItemLogs(checked ? 'add' : 'remove', item)}
  />
);
```

Listing 6.1: Type assertions in `./src/app/screens/BottomTab/HomePage.tsx` [129]

Testing would also happen manually aided with the desired interface and the browsable API that Django REST Framework offers. This was to establish that all requirements of the project were fulfilled and the application was established as desired. Since automation has been preferred throughout the project, popular testing frameworks for Python and TypeScript are *Pytest* and *Jest* respectively. The application would further be tested with Django. To setup Jest for the interface, however, the API (axios discussed in 5.3.2) needs to be mocked which would require a lot of work. Browser activity could also be mocked using frameworks like *Cypress*, but that as well requires hours to setup. Since the server is tested, interface testing is not as critical.

```
def test_user_creation(self):
    """
    Ensure we can create new users as desired through API.
    """
    url = reverse("user-list")
```

```
    response = self.client.post(url, data={"email": "test"})
    self.assertEqual(response.status_code, status.HTTP_400_BAD_REQUEST)
    self.assertDictContainsSubset(
       {
          "email": ["Enter a valid email address."],
          "password": ["This field is required."],
       },
       response.data,
    )
```

Listing 6.2: Testing registration API for security in `./src/server/rest/tests.py` [131]

6.2 Survey Design

Finally, the further evaluation of the application would be done by users themselves, and so a questionnaire would also follow this to help understand the usability. There were a few considerations while designing the questionnaire, like gathering sensitive information, and not taking too much time from participants. All questions are in compliance with GDPR & the Ethics Checklist from the School of Computing Science, University of Glasgow [101].

6.2.1 Demographics

Information about the participants is gathered. None of these are personal, identifying questions that a participant may not be comfortable with. The reason for these questions is to understand why a participant may choose a specific response. It gives an idea if a person would be good with technology (which could introduce bias) and therefore not having great difficulty with operating the interface. More importantly, it helps understand a person's dietary habits - as food is the topic of this application - answering questions if they would have the need and motivation for the application. This is pre-activity and entirely optional to participants.

6.2.2 Effect

Aside from the usability, the survey also aims to gather the effect the application made on users post-activity. This would mean if they were able to learn more about the Eatwell Guide, and if they would be motivated about following a diet plan using the app as a means of support.

6.2.3 Method

User Experience (UX) measurement could be difficult, as experiences are intangible and highly subjective. Metrics can be introduced by measuring task completion time/rate - where users would perform specified tasks - and monitoring page analytics - where user sessions would be recorded to learn about page jumps and retention over time with practical usage, however they may not convey experience accurately and not suitable for the project (given the timeline). There are standardised methods of UX Measurement [77], such as the User Experience Questionnaire (UEQ) that requires 26 linear ratings to measure attractiveness and efficiency of the system. To ensure that participants do not lose focus of the task and their time is respected, the target was 15

questions, requiring 30 minutes in total at most. Therefore, the *System Usability Scale (SUS)* was chosen. Being used in over 1300 articles and easy to administer to participants [4], it consists of 10 statements that one can respond easily to using a five point range from Strongly Agree (1) to Strongly disagree (5) with 3 being neutral. The statements vary in the describing the system as positive (denoted by $^+$) or negative (denoted by $^-$) as well, so inverting the meaning ensures that participants do not lose focus by expecting the same answer format.

1. I think that I would like to use this system frequently.$^+$
2. I found the system unnecessarily complex.$^-$
3. I thought the system was easy to use.$^+$
4. I think that I would need the support of a technical person to be able to use this system.$^-$
5. I found the various functions in this system were well integrated.$^+$
6. I thought there was too much inconsistency in this system.$^-$
7. I would imagine that most people would learn to use this system very quickly.$^+$
8. ~~I found the system very cumbersome to use.$^-$~~
9. I felt very confident using the system.$^+$
10. I needed to learn a lot of things before I could get going with this system.$^-$

6.3 Results

Picking suitable statements from the ones provided from the System Usability Survey[1] (discussed in 6.2.3) and deploying the application onto GitHub Pages (for the frontend discussed in 5.3.2) and Heroku (for the backend discussed in 5.3.1), evaluations started on 28 February 2022 (Monday) through a Google Form and ran for one week, getting responses from 10 participants. To get the usability score, the number mapped to the response for each statement (depending if the statement is negative$^-$, then the number is subtracted from 5 i.e. $5 - x$ or if it is positive$^+$, then 1 is subtracted from the number i.e. $x - 1$) added together for each participant, then multiplied by 2.5 and since the survey used 9 questions instead of 10, divided by 0.9. The mean (or average) score would be 80.0 which makes the application "Acceptable" with a grade of A- [76, 99]. Given the early stage of the application, the score is impressive and with more time, can be improved.

Summary

The application was tested manually to verify that initial requirements are met, and automatically to verify functionalities are functioning as intended. With survey planned and designed to gather market demographics and the usability of the app more importantly, the evaluation used the System Usability Survey (SUS) scale, with 10 participants testing the application between 28 February 2022 and 7 March 2022, giving the result of an acceptable system (score 80.0), meaning it is effective, efficient and easy to use.

[1] Only statement 8 was redundant.

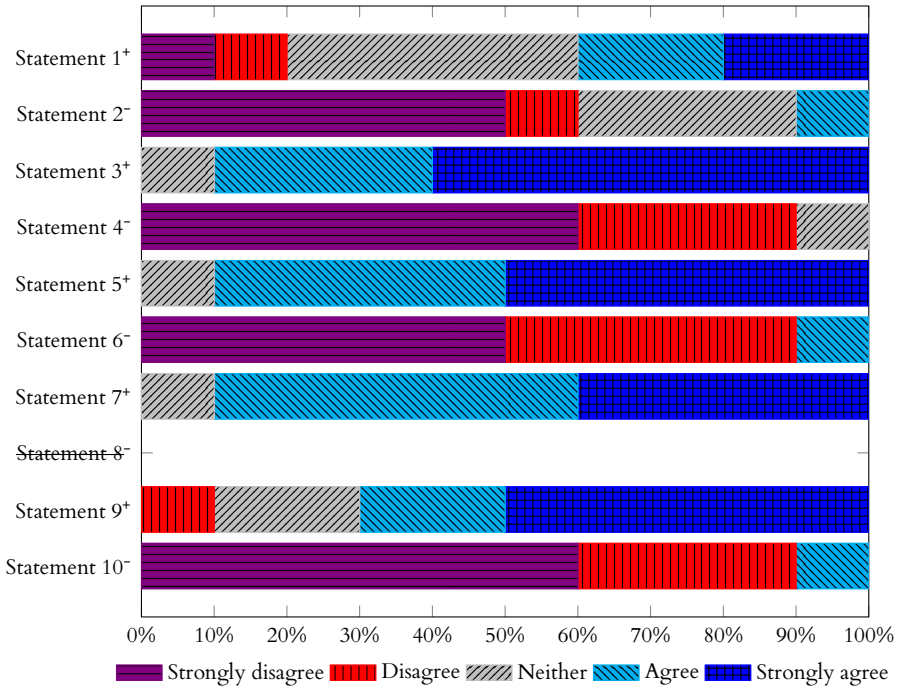


Figure 6.1: Responses for each statement

	Score	
Participant 1	61.1	D
Participant 2	52.8	D
Participant 3	58.3	D
Participant 4	94.4	A+
Participant 5	91.7	A+
Participant 6	75.0	B
Participant 7	88.9	A+
Participant 8	97.2	A+
Participant 9	88.9	A+
Participant 10	91.7	A+
System Score (Mean)	**80.0**	**A–**

Table 6.1: SUS Score for each participant

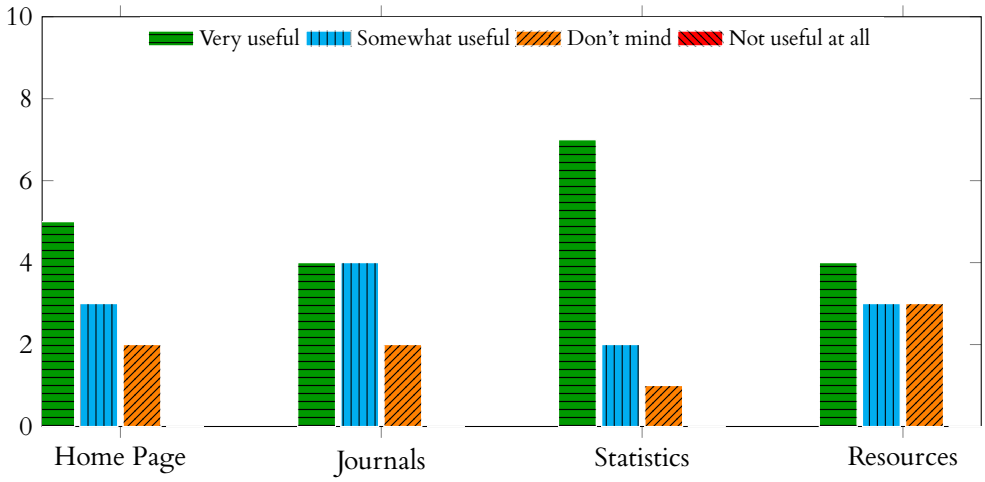


Figure 6.2: Feedback for each screen of the app

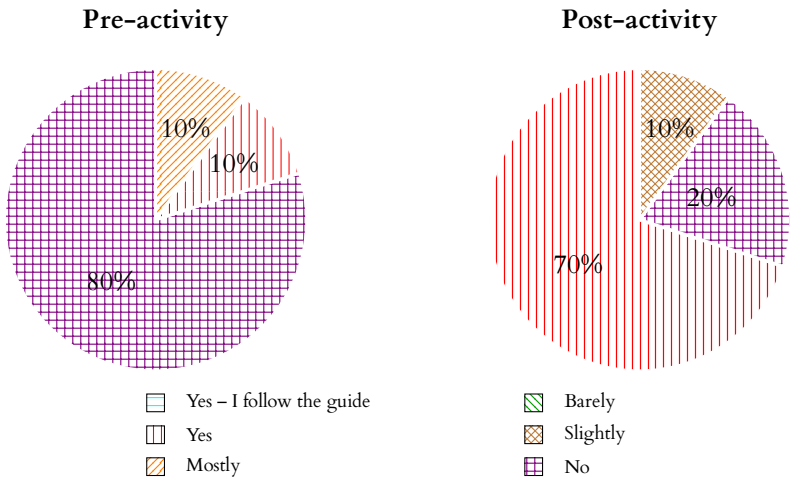


Figure 6.3: Response to knowing about the Eatwell Guide

7 Conclusion

This chapter addresses the problem and their solutions introduced in the previous chapters, along with summarising and closing off the dissertation with final thoughts about the entire project in terms of doing things differently, personal reflection and potential for future projects.

7.1 Summary

Portion Mate was originally an idea for an application to make logging food portions easier, and it was inspired by the NHS and dietitians who help people maintain healthier lifestyles through following simple suggestions such as the Eatwell Guide. However, this became more than that. Portion Mate became a story about learning and developing technologies that considers and dignifies all people, from users to developers, treating them as humans since the decisions were complying the study of Human Computer Interaction; the requirements and design work together to create and solve user expectations that developers could also help improve. The development used specific practices strictly, implemented from scratch if they seemed unfeasible, in a block-structured timeline enabling flexible implementation with rigid progress. With the help of existing technology and packages, the construction of the application went ahead, but also brought ideas to develop more for similar use cases. This enabled creation of a fully-integrated robust and modular system, as evaluated through the usability study with participant-users to get the Acceptable score, showing promising potential for more, that can make a difference in software development and the medical health industry.

7.2 Side Projects

During the early stages of development, there was need for different solutions already. As mentioned throughout the dissertation, issue branching was used, inspired by other code repository hosting services that take advantage of issue tracking (Bitbucket and Jira specifically); this had to be implemented for GitHub, where the remote repository for this project is/was hosted, as, at the time of development, it did not provide this functionality, but instead provide an innovative CI/CD solution called GitHub Actions, and this was used to automate issue branching with project boards on GitHub. Since the code was long and complex, it was best to export it to another repository to only keep appropriate logic of the application in the main monorepo. The new repository was therefore named "Project Card Action" and distributed on GitHub Marketplace for others to use in their GitHub Actions workflow.

In a similar scenario of automating through GitHub and its different features, the platform offers a section to host documentation related to the repository. The wiki is a separate repository itself and therefore changes would need to be committed. Maintaining the main repository as the single source of truth, the documentation would be stored there, and any changes would be reflected to the wiki by automatically committing. However, there are also limitations of configuring a navigation sidebar for wikis on GitHub, therefore a solution was developed to

automatically generate this, and commit onto the wiki repository. This package was named "Wiki Action", and also distributed on the marketplace for others.

Following this, some documentation, such as the meeting minutes, would follow a certain format, and maintaining it was difficult at times. Therefore, a possible solution came to be generating the compatible Markdown file by reading data from JSON, which can also then be used for APIs if required. At the time of writing, this is in use in the repository, however, more research and development would be required to establish this as a package.

Before running evaluations, the Ethics Checklist had to be signed. Since the project development was during the COVID-19 pandemic, where work would be remote and online, along with the repository being the central source to hold documents and code, and also being public, there were difficulties to ensure that signatures for both - the student and the supervisor - would be secure, as print-name could be done by anyone, and adding scans of signatures as image assets can allow people to download it. Taking inspiration from metadata badges such as Shields.io and public key encryption, an authenticating API was developed called "Authentica".

The styling problem in React Native was solved in 5.4 by implementing a Bootstrap-classes equivalent module. Being in TypeScript and aspiring to provide all classes in Bootstrap, more development would be required and then exporting as a separate package. Many more configurations aim to be provided as plugins to developers so that focus can be spent on improving software. More discussion in detail about side projects is in Appendix C.2.

7.3 Future Work

Projects mentioned in 7.2 require work to refine development over time and help other developers in the community. As for Portion Mate, development can continue since issues have been listed on the repository with a few open at the time of writing. With no time constraint, the application can be perfected and perhaps development could appreciate a team, given the nature of the environment setup and practices used. The evaluation score was promising, and based on feedback from the participants, there would be elements to improve. Since this was the first time working with such a project using such technologies, more experience would help polish all corners, including the interface that required the most time in this project and still needs some, given it is an essential part, as discussed in 4.3 and the principles of Human Computer Interaction.

7.4 Reflection

This experience, for me, has been phenomenal. As someone who takes on projects with passion and vision, I aimed for the limitless for Portion Mate since it is the first individual project I would be graded on. I feel blessed with the opportunity to work on this, under the supervision of Dr. Andrei, and over the weeks, I have been nothing but inspired to strive for better. Having pushed myself to learn while I'm a student at a brilliant university, I feel happy to go beyond my comfort zone to use tools I would not know about before this project, even though they were not asked for. My love-hate relationship with being specific about things carried over to this project, as I would focus on nothing but the minor details in order to perfect it, but that consumes a lot of time - an asset that cannot be reversed or bought. The bigger picture can be seen differently from everyone, and it is essential to not lose touch from it. Working on each detail helped, but perfecting them may have led to an incomplete system or sub-system optimisation, whereas the goal was to develop a working application. Having achieved that, with additional features, I am extremely proud of my work and I aspire to be a qualified, professional software developer in the industry.

A Figures & Graphics

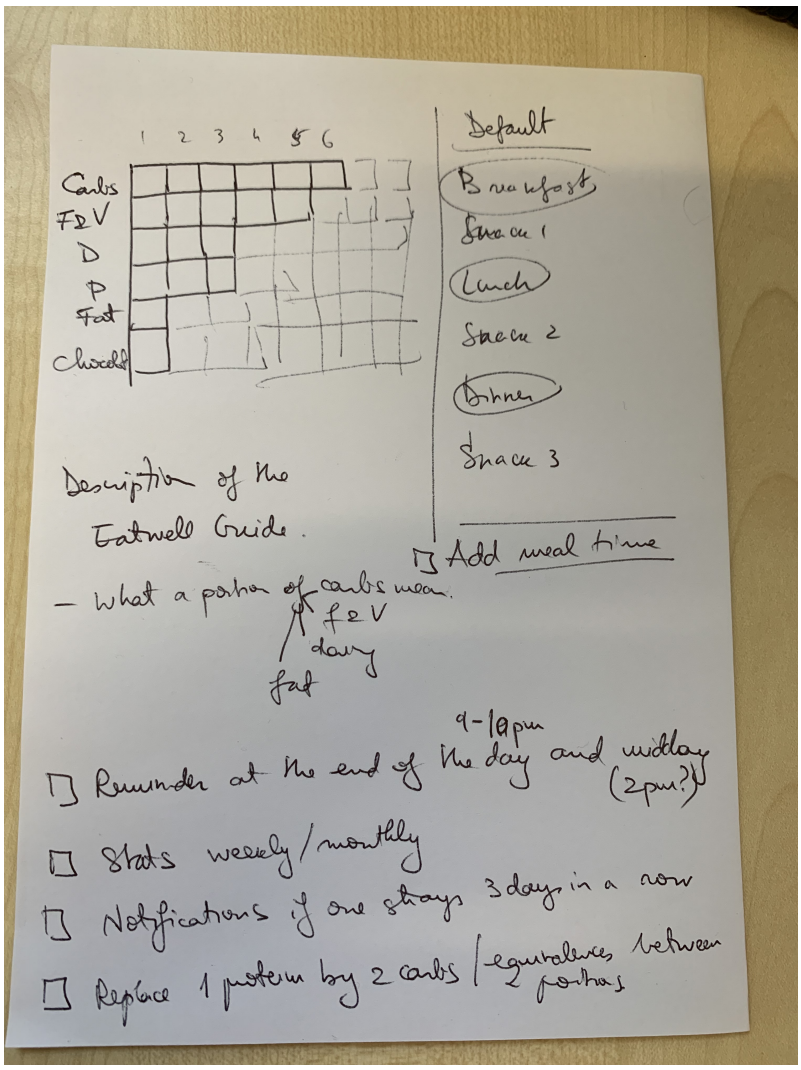


Figure A.1: Example paper log displayed at first meet

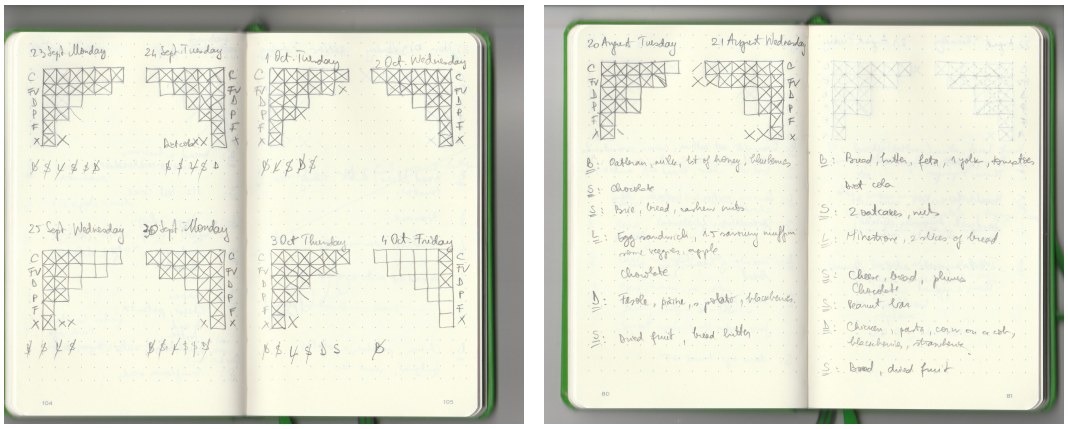


Figure A.2: More scans of practical usage of paper logs

Figure A.3: Initial Entity Relationship model

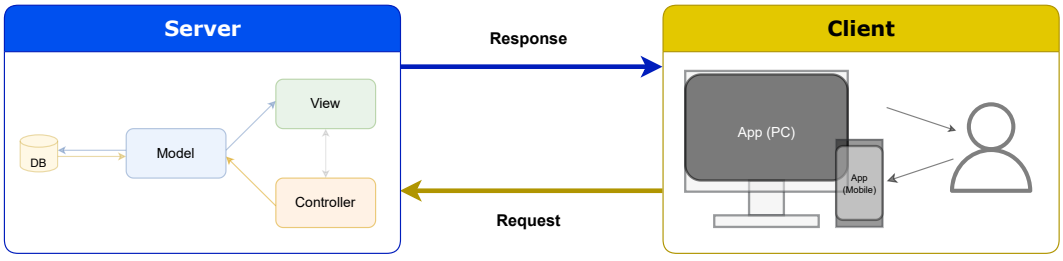
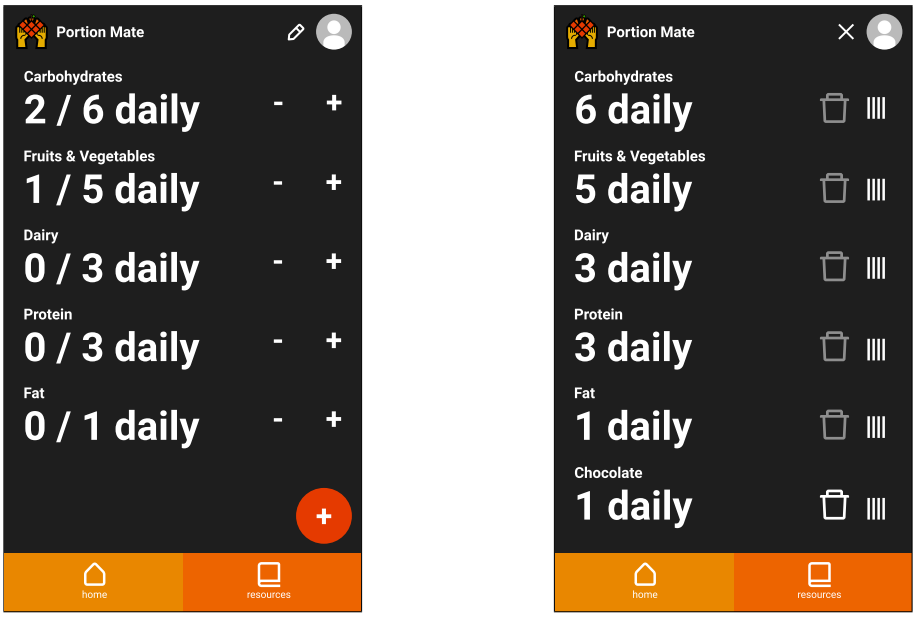


Figure A.4: Overview of REST Architecture

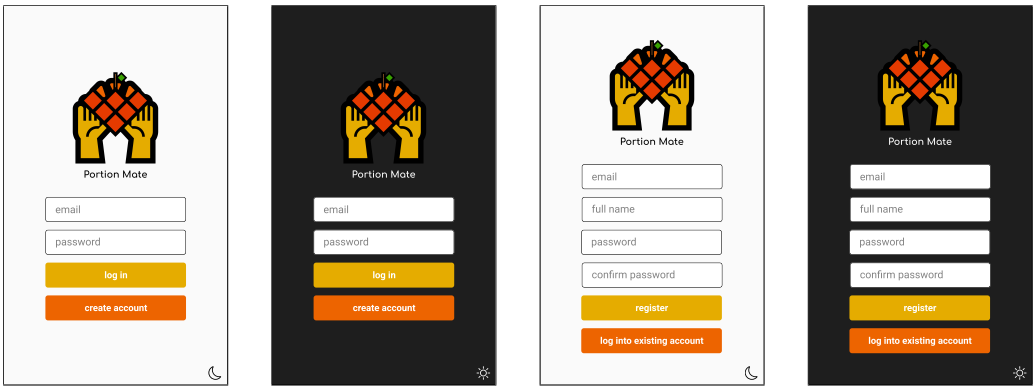


(a) Home page **(b)** Edit view of the home page

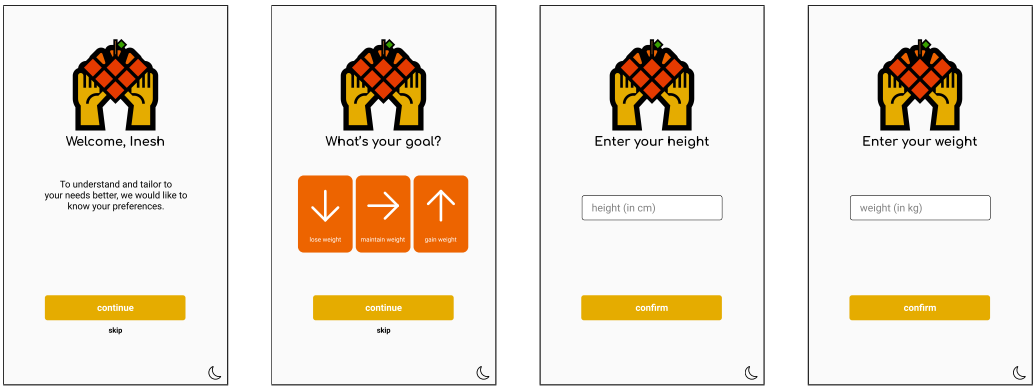
Figure A.5: Homepage design from initial wireframes

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
$S1^+$	1	4	3	5	3	3	3	5	4	2
$S2^-$	3	3	4	1	1	1	2	1	3	1
$S3^+$	4	4	3	5	5	4	5	5	5	5
$S4^-$	2	3	2	1	1	2	1	1	1	1
$S5^+$	4	4	3	5	5	4	5	4	5	5
$S6^-$	2	4	2	2	1	2	1	1	1	1
$S7^+$	4	3	4	4	5	4	5	5	4	5
$S8^-$										
$S9^+$	3	3	2	5	5	4	4	5	5	5
$S10^-$	2	4	1	1	2	2	1	1	1	1

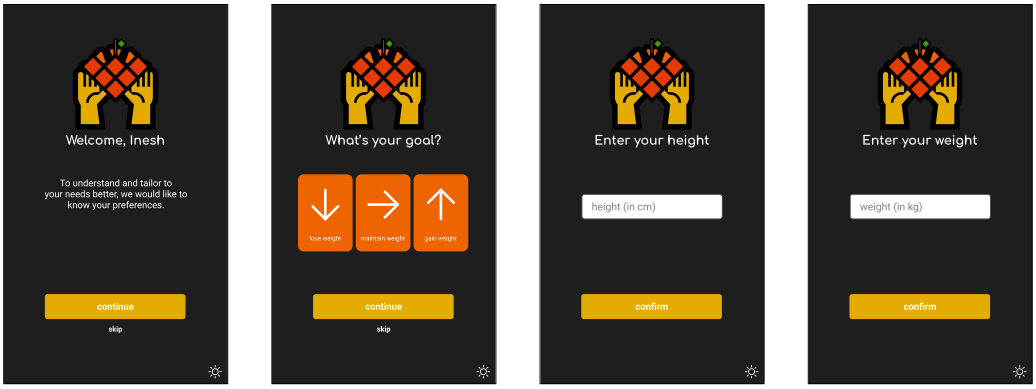
Table A.1: Participant responses for each statement



(a) Login form (light) **(b)** Login form (dark) **(c)** Register form (light) **(d)** Register form (dark)

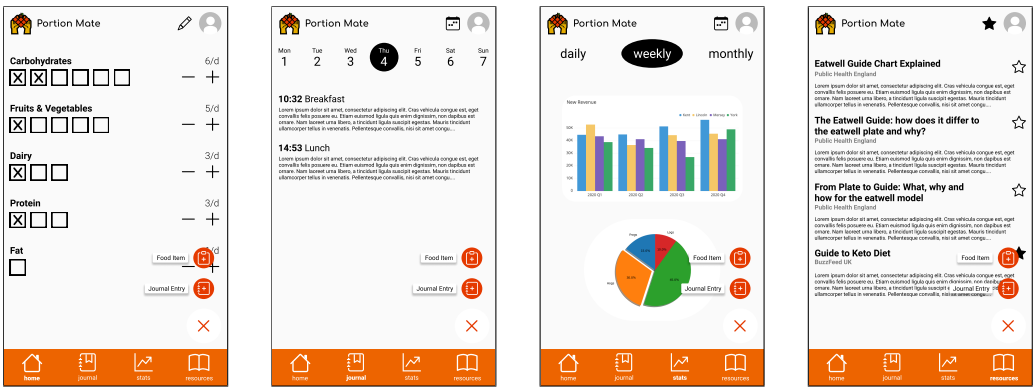


(e) Welcome (light) **(f)** Goal inquiry (light) **(g)** Height inquiry (light) **(h)** Weight inquiry (light)

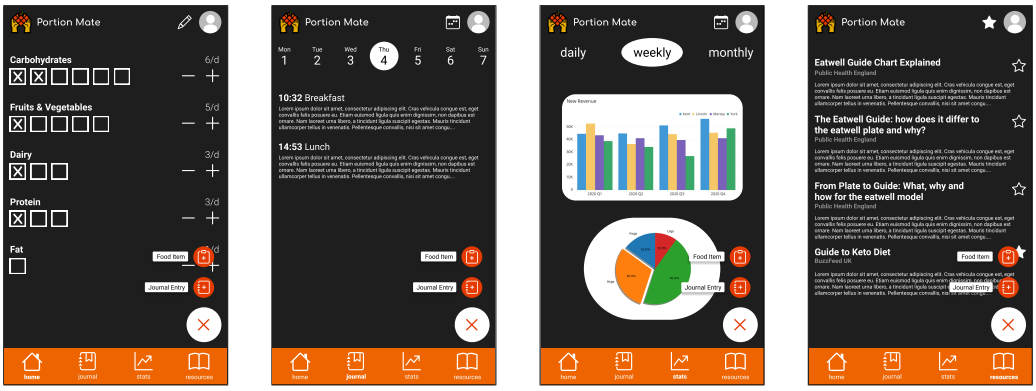


(i) Welcome (dark) **(j)** Goal inquiry (dark) **(k)** Height inquiry (dark) **(l)** Weight inquiry (dark)

Figure A.6: Wireframes for login and registration

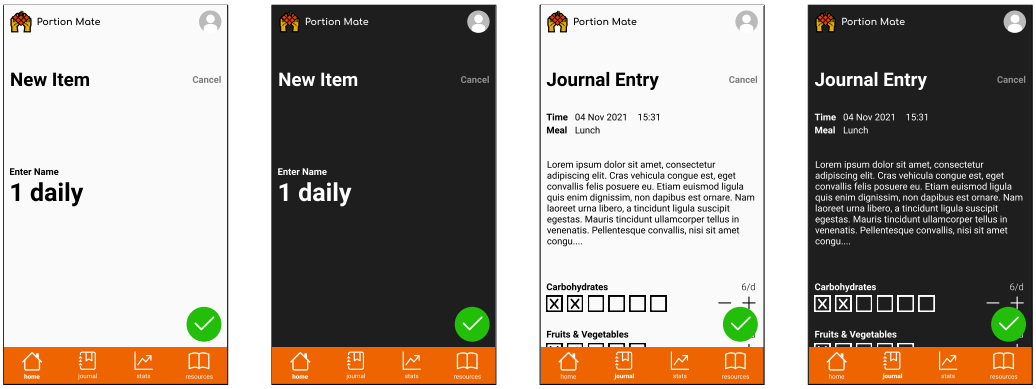


(a) Home (light) **(b)** Journal (light) **(c)** Stats (light) **(d)** Resources (light)



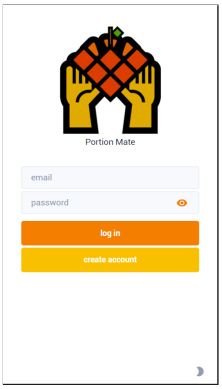
(e) Home (dark) **(f)** Journal (dark) **(g)** Stats (dark) **(h)** Resources (dark)

Figure A.7: Floating Action Button preview on different screens (wireframes)

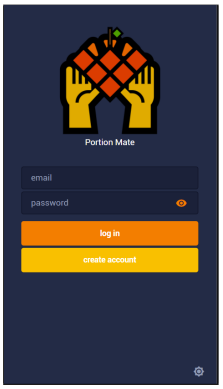


(a) Add portion item (light) **(b)** Add portion item (dark) **(c)** Add diary entry (light) **(d)** Add diary entry (dark)

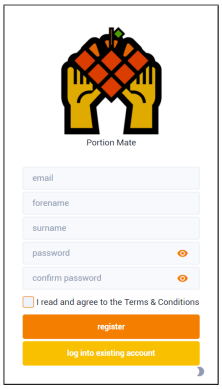
Figure A.8: Wireframes for adding logs and entries



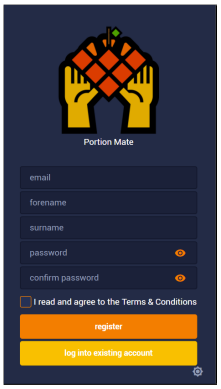
(a) Login form (light)



(b) Login form (dark)

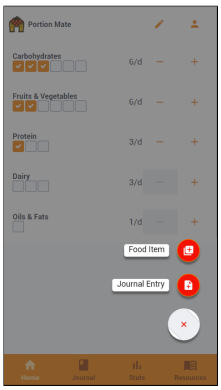


(c) Register form (light)

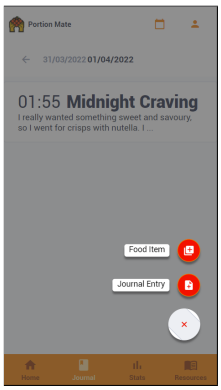


(d) Register form (dark)

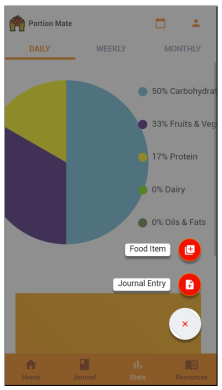
Figure A.9: Final interface for login and registration



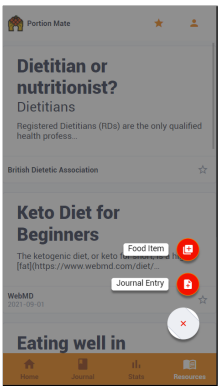
(a) Home (light)



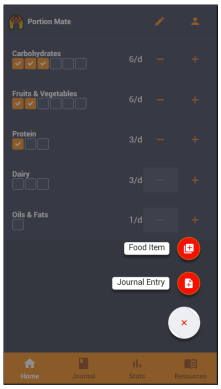
(b) Journal (light)



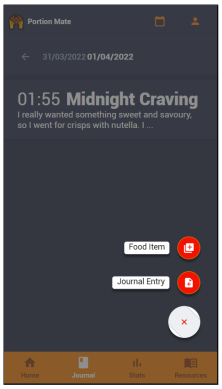
(c) Stats (light)



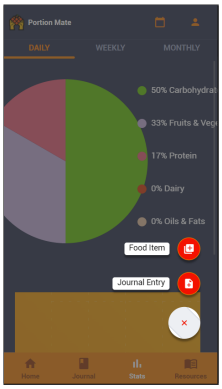
(d) Resources (light)



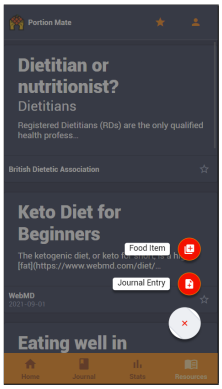
(e) Home (dark)



(f) Journal (dark)

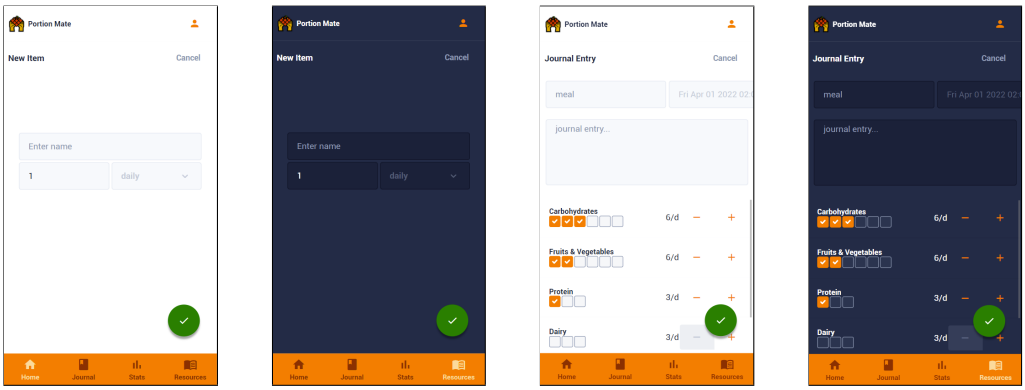


(g) Stats (dark)



(h) Resources (dark)

Figure A.10: Floating Action Button preview on different screens (final)



(a) Add portion item (light) **(b)** Add portion item (dark) **(c)** Add diary entry (light) **(d)** Add diary entry (dark)

Figure A.11: Wireframes for adding logs and entries

	Score	Mean	SD
Statement 1^+	57.5	3.3	1.18
Statement 2^-	75.0	2.0	1.09
Statement 3^+	87.5	4.5	0.67
Statement 4^-	87.5	4.5	0.67
Statement 5^+	85.0	4.4	0.66
Statement 6^-	82.5	1.7	0.90
Statement 7^+	82.5	4.3	0.64
~~Statement 8^-~~			
Statement 9^+	77.5	4.1	1.04
Statement 10^-	85.0	1.6	0.91

Table A.2: SUS Score, with mean and standard deviation, for each statement

B Code Listings

```
import { mapping, light, dark } from '@eva-design/eva';
import {
  ApplicationProvider,
  Button,
  ButtonProps,
  Icon,
  IconRegistry,
} from '@ui-kitten/components';
import { MaterialIconsPack } from './app/components/AppIcons';
import { ColorScheme } from './app/types';
import { ThemeContext } from './app/contexts/ThemeContext';
import Navigation from './app/navigation';
import { default as colors } from './app/assets/theme.json';
import { default as customMapping } from './app/assets/mapping.json';

export default function App() {

  // ...

  return (
    <>
      <IconRegistry icons={MaterialIconsPack} />
      <ThemeContext.Provider
        value={{ theme, setTheme, switchTheme, ThemeToggle }}
      >
        <ApplicationProvider
          mapping={mapping}
          customMapping={customMapping}
          theme={{ ...(isLightTheme ? light : dark), ...colors }}
        >
          <Navigation />
        </ApplicationProvider>
      </ThemeContext.Provider>
    </>
  );
}
```

Listing B.1: Root definition of the App component, using Eva Design and UI Kitten components in `./src/App.tsx` [126]

```yaml
jobs:
  black_linter:
    name: Black formatter
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - uses: rickstaa/action-black@v1
        with:
          black_args: ". --check"
  django_tests:
    name: Django tests
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - uses: actions/setup-python@v3
        with:
          python-version: 3.9
      - name: Install dependencies
        run: |
          python -m pip install --upgrade pip
          pip install ./src
      - name: Run test
        run: |
          python run.py migrate
          python run.py test main rest
```

Listing B.2: job definitions for checking Python code and style in `./.github/workflows/server-checks.yml` [123]

C Extra Discussion

C.1 Usability Survey

These subsections include the instruction text from each section of the survey.

C.1.1 Introduction

The aim of this experiment is to investigate the scale of usability for an application that is designed for people, such as yourself, to log their food intake. As you could understand, this may require commitment, but this experiment should take no more than 30 minutes. Before you decide whether you want to take part, it is important for you to understand why the research is being done and what your participation will involve. You should only participate if you want to; choosing not to take part will not disadvantage you in any way, and you are welcome to withdraw at any time. If you do so, then it will not be possible for you to be debriefed about the purposes of the experiment. Please take time to read the following information carefully and ask us if there is anything that is not clear or if you would like more information. For questions, you can send an email to Inesh Bose: 2504266b@student.gla.ac.uk

The application "Portion Mate" was created as a Level 4 Dissertation Project with the target to help users keep track of their daily portion count from each food group based on the NHS Eatwell Guide (you can read more on https://portion-mate.readthedocs.io/). You will see the app in action during this experiment, and in order to evaluate its usability, you will be required to answer this survey. There are three sections in this survey which will also provide further information about the relevance of the questions. You are free to not answer questions that are not required, but the data will help. All responses are anonymous and in compliance with GDPR and the University of Glasgow SoCS Project Ethics.

C.1.2 Pre-Activity

The questions in this section focus on your demographics. These questions are designed in order to understand you, as a user, and to mitigate any bias. This section has no required responses.

C.1.3 Interface

This section would be answered after using the app and focuses on the usability of the application through your experience and journey with the app. The questions here are statements, based on the System Usability Scale, which you can answer on a scale of 1-5 (3 is neutral) going from Strongly disagree and Strongly agree depending on how you feel and agree with the statement. All responses in this section are required.

You are now asked to navigate to the app on your browser (https://inesh.xyz/portion-mate) and follow the tasks given below which are not timed or judging your performance, but helping you go through the app itself. You may choose to evaluate the application without these tasks.

- Create an account, or login if you already have one
- Log your portion items taken so far in the day
- Visualise the graphs for the items logged
- Go to Resources, read about The Eatwell Guide and bookmark it if needed
- Additionally, add a journal entry about your first meal of the day

C.1.4 Post-Activity

This final section is focused on hearing your thoughts and feelings after using the app. This can help open doors to future work that can create a better product for more users. There are no required responses.

C.1.5 Debrief

With this, the survey ends and you will be debriefed about this experiment.

The aim of the experiment was to evaluate the usability of a daily food portion count tracker application (called Portion Mate). This meant navigating through the pages, interacting with elements, visualisation and appeal. Your responses would help point out and understand any frustrations or problems that might have occurred during your experience.

If you would like to know and see more about this project, everything (including the source code) can be found on this GitHub repository: https://github.com/ineshbose/portion-mate. This repository acts as the single source of truth for all information regarding Portion Mate. This would also mean participant feedback and data analysis, therefore if you would like to aid this practice, you can confirm your responses being held in this public repository anonymously. By default, your data will not be public and on the repository if you do not agree. It will help more developers gather insight for making better applications for the world in the future.

For more questions and concerns, you can send an email to Inesh Bose: 2504266b@student.gla.ac.uk

C.2 Side Projects

These subsections discuss background and development of the side projects born out of this.

C.2.1 Project Card Action

This was the first side project developed, in the early weeks of setup [134]. Before exporting to a separate repository, the script was implemented within a workflow YAML, but using `actions/github-script` [3] to execute JavaScript within the job and interact with GitHub API. As mentioned, the code was complex and long, therefore it moved to a separate repository, along with the purpose to distribute it. It is a JavaScript action [27], but written in TypeScript, that compiles to JavaScript in the `dist` directory. The repository is on https://github.com/ineshbose/project-card-action and open-sourced with the MIT License. However, at the time of writing, GitHub Projects (Beta) has been made public (migration from classic pending) which may already implement such a functionality.

C.2.2 Wiki Action

The reason why deploying on Wiki was important was because on the repository, it is difficult to navigate between documentation and view Markdown. The project knew that the difference between the documentation on Read the Docs would be more public, whereas on the wiki, it would be development related, such as background and meeting notes. GitHub requires a `_Sidebar.md` [26] with the desired layout and pages listed. If a new file would be added, another change would be made to the sidebar, and as humans, developers may forget this change. Therefore this would be automated through a shell script, hence creating a composite action [25]. There have been very similar solutions, however, some would specialise in one aspect or have limitations like making changes before committing onto the wiki repository. Wiki Action expects the repository to be cloned using `Checkout V3` [19] as most other actions on GitHub, and changes can be made in between. The sidebar generation can have custom elements, or be disabled entirely. Another limitation this solves is not requiring any special token stored in the repository secrets, and rather use the default `GITHUB_TOKEN`. There are still a lot of bugfixes and features implementations that need to be carried out. The repository, https://github.com/ineshbose/wiki-action, is open-sourced with the MIT License.

C.2.3 Authentica

The idea for this repository came when the Ethics Checklist had to be signed before starting evaluations [135]. It requires the student's name and university ID - none of these are confidential or require any security. Mainly, it required signatures from the student and the supervisor. Similar to how metadata badges from *Shields.io* function with other services to display status on elements, a signature method could be implemented, since signatures could be boolean - `true` if the signature is supposed to be there, or `false` if it hasn't been added by the person. Shields.io provides an endpoint where JSON data can be given to render a badge, but loading this data would pose difficulties. If a server had to be created and hosted, the system would not really innovate. For the Ethics Checklist, the signature was verified by hosting a JSON file onto a personal domain owned by the signee. Dr. Andrei verified her signature by making a commit that produces her print-name as the diff, but wondered with curiosity if she would be able to do this as well. This gave the feeling that such a project could be useful for everyone. Keep in mind that this is similar to services like Eversign, but different - more tailored to the open-source community who aim to keep everything transparent and write documents in Markdown. However, no time could be invested in the implementation of this - until the coursework for Human Centred Security (M), taken in the same year of this project, required students to develop a security solutions in groups. Coincidentally, the developers of How Green? united again for this project to use Next.js hosted on Vercel and make the system possible. The source code is on https://github.com/ineshbose/authentica, with the service on https://authentica-io.vercel.app/.

C.2.4 Template Markdown

Markdown provides an incredible method to document elements, and could be rendered to HTML or LaTeX. `mdx-js/mdx` [64] extends this to use it in JavaScript libraries such as React to make user interfaces. The supervisor meetings for Portion Mate were regular and documented using a fancy format with tables. Ensuring the tables are error-free and the code style is good becomes really tiring. So a script was developed that would create Markdown documents using JSON data and a template defined as `.t.md`. This template would include placeholders in the form of template literal strings for JavaScript to parse and replace values. There are still limitations, like restrictions on functional approaches and nested template strings. More research is required.

7 Bibliography

[1] *5 A Day: what counts?* en. Section: livewell. Apr. 2018. URL: https://www.nhs.uk/live-well/eat-well/5-a-day-what-counts/ (visited on 03/21/2022).

[2] *Accessibility - W3C*. URL: https://www.w3.org/standards/webdesign/accessibility (visited on 04/01/2022).

[3] *actions/github-script*. original-date: 2019-08-29T22:46:51Z. Mar. 2022. URL: https://github.com/actions/github-script (visited on 04/01/2022).

[4] Assistant Secretary for Public Affairs. *System Usability Scale (SUS)*. en-us. Publisher: Department of Health and Human Services. Sept. 2013. URL: https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html (visited on 04/01/2022).

[5] *airbnb/javascript: JavaScript Style Guide*. URL: https://github.com/airbnb/javascript (visited on 03/20/2022).

[6] Stian Alexander. *One in ten people haven't cooked for a YEAR as they rely on takeaways*. Section: News. Nov. 2019. URL: https://www.dailymail.co.uk/news/article-7718917/One-ten-people-havent-cooked-YEAR-rely-takeaways-microwave-meals.html (visited on 03/21/2022).

[7] D.J. Anderson and A. Carmichael. *Essential kanban condensed*. [Essential kanban]. Lean-Kanban University, 2015. ISBN: 978-0-9845214-2-5. URL: https://books.google.co.uk/books?id=fHMCDQEACAAJ.

[8] *AppVeyor*. en. URL: https://www.appveyor.com/ (visited on 04/01/2022).

[9] *Baritastic*. URL: https://www.baritastic.com/ (visited on 03/29/2022).

[10] BBC. *The shocking transformation of the UK household diet since 1980*. June 2021. URL: https://www.youtube.com/watch?v=A1qbkL-Iff8 (visited on 03/21/2022).

[11] *Berry*. original-date: 2018-09-10T22:49:22Z. Mar. 2022. URL: https://github.com/yarnpkg/berry (visited on 03/21/2022).

[12] *Black*. original-date: 2018-03-14T19:54:45Z. Mar. 2022. URL: https://github.com/psf/black (visited on 03/21/2022).

[13] Matthias Böhmer et al. "Falling asleep with Angry Birds, Facebook and Kindle: a large scale study on mobile application usage". en. In: *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services - MobileHCI '11*. Stockholm, Sweden: ACM Press, 2011, p. 47. ISBN: 978-1-4503-0541-9. DOI: 10.1145/2037373.2037383. URL: http://dl.acm.org/citation.cfm?doid=2037373.2037383 (visited on 03/11/2022).

[14] *BootstrapVue*. en. URL: https://bootstrap-vue.org (visited on 03/20/2022).

[15] Nicolas Brousse. "The issue of monorepo and polyrepo in large enterprises". en. In: *Proceedings of the Conference Companion of the 3rd International Conference on Art, Science, and Engineering of Programming*. Genova Italy: ACM, Apr. 2019, pp. 1–4. ISBN: 978-1-4503-6257-3. DOI: 10.1145/3328433.3328435. URL: https://dl.acm.org/doi/10.1145/3328433.3328435 (visited on 03/21/2022).

[16] *Cards*. en. URL: https://getbootstrap.com/docs/5.1/components/card/#example (visited on 03/20/2022).

[17] *Cards. BootstrapVue*. en. URL: https://bootstrap-vue.org/docs/components/card/#overview (visited on 03/20/2022).

[18] *Cards. React-Bootstrap*. URL: https://react-bootstrap.github.io/components/cards/#basic-example (visited on 03/20/2022).

[19] *Checkout V3*. original-date: 2019-07-19T17:15:33Z. Mar. 2022. URL: https://github.com/actions/checkout (visited on 04/01/2022).

[20] *Codacy*. en. URL: https://www.codacy.com/ (visited on 04/01/2022).

[21] *Code Climate*. en. URL: https://codeclimate.com/ (visited on 04/01/2022).

[22] comScore. *In U.S. Mobile Market, Samsung, Android Top The Charts; Apps Overtake Web Browsing*. en-US. URL: https://social.techcrunch.com/2012/07/02/comscore-in-u-s-mobile-market-samsung-android-top-the-charts-apps-overtake-web-browsing/ (visited on 03/11/2022).

[23] *COVID-19 App*. original-date: 2020-05-03T09:36:44Z. Feb. 2022. URL: https://github.com/nihp-public/COVID-19-app-Documentation-BETA (visited on 03/28/2022).

[24] *Covid-19 exposure app*. original-date: 2020-07-30T15:07:22Z. Mar. 2022. URL: https://github.com/nihp-public/covid-19-app-android-ag-public (visited on 04/01/2022).

[25] *Creating a composite action*. en. URL: http://ghdocs-prod.azurewebsites.net:80/en/actions/creating-actions/creating-a-composite-action (visited on 04/01/2022).

[26] *Creating a footer or sidebar for your wiki*. en. URL: https://docs.github.com/en/communities/documenting-your-project-with-wikis/creating-a-footer-or-sidebar-for-your-wiki (visited on 04/01/2022).

[27] *Creating a JavaScript action*. en. URL: http://ghdocs-prod.azurewebsites.net:80/en/actions/creating-actions/creating-a-javascript-action (visited on 04/01/2022).

[28] *Cypress*. original-date: 2015-03-04T00:46:28Z. Apr. 2022. URL: https://github.com/cypress-io/cypress (visited on 04/01/2022).

[29] *Dairy and alternatives in your diet*. en. Section: livewell. Apr. 2018. URL: https://www.nhs.uk/live-well/eat-well/milk-and-dairy-nutrition/ (visited on 03/21/2022).

[30] *Dependabot*. original-date: 2017-06-02T12:23:31Z. Apr. 2022. URL: https://github.com/dependabot/dependabot-core (visited on 04/01/2022).

[31] *Dietary choices of Brits (e.g. vegeterian, flexitarian, meat-eater etc)?* en-gb. URL: https://yougov.co.uk/topics/lifestyle/trackers/dietery-choices-of-brits-eg-vegeterian-flexitarian-meat-eater-etc (visited on 03/21/2022).

[32] *Django*. original-date: 2012-04-28T02:47:18Z. Mar. 2022. URL: https://github.com/django/django (visited on 03/21/2022).

[33] *Django appears to be a MVC framework, but you call the Controller the "view", and the View the "template". How come you don't use the standard names?* URL: https://docs.djangoproject.com/en/dev/faq/general/#django-appears-to-be-a-mvc-framework-but-you-call-the-controller-the-view-and-the-view-the-template-how-come-you-don-t-use-the-standard-names (visited on 03/21/2022).

[34] *Django REST framework*. original-date: 2011-03-02T17:13:56Z. Mar. 2022. URL: https://github.com/encode/django-rest-framework (visited on 03/21/2022).

[35] *Eating processed foods*. en. Feb. 2022. URL: https://www.nhs.uk/live-well/eat-well/how-to-eat-a-balanced-diet/what-are-processed-foods/ (visited on 04/01/2022).

[36] *ESLint*. original-date: 2013-06-29T23:59:48Z. Mar. 2022. URL: https://github.com/eslint/eslint (visited on 03/21/2022).

[37] *Eva Design System*. original-date: 2019-05-29T08:16:12Z. Mar. 2022. URL: https://github.com/eva-design/eva (visited on 03/24/2022).

[38] *Expo*. original-date: 2016-08-15T17:14:25Z. Mar. 2022. URL: https://github.com/expo/expo (visited on 03/24/2022).

[39] Roy T. Fielding and Julian Reschke. *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content*. Request for Comments RFC 7231. Num Pages: 101. Internet Engineering Task Force, June 2014. DOI: 10.17487/RFC7231. URL: https://datatracker.ietf.org/doc/rfc7231 (visited on 04/01/2022).

[40] *Fitbit App*. URL: https://www.fitbit.com/gb/app (visited on 03/29/2022).

[41] *Flask*. original-date: 2010-04-06T11:11:59Z. Mar. 2022. URL: https://github.com/pallets/flask (visited on 03/21/2022).

[42] *Flo*. en. URL: https://flo.health/ (visited on 03/29/2022).

[43] *Flutter*. original-date: 2015-03-06T22:54:58Z. Apr. 2022. URL: https://github.com/flutter/flutter (visited on 04/01/2022).

[44] Git. *Git Hooks*. URL: https://git-scm.com/book/en/v2/Customizing-Git-Git-Hooks (visited on 03/20/2022).

[45] *GitHub Actions*. en. URL: https://github.com/features/actions (visited on 04/01/2022).

[46] University of Glasgow. *INDIVIDUAL PROJECT (H) (SINGLE) COMPSCI4025P*. URL: https://www.gla.ac.uk/coursecatalogue/course/?code=COMPSCI4025P (visited on 03/20/2022).

[47] *Google Fit*. en-GB. URL: https://www.google.com/fit/ (visited on 03/29/2022).

[48] Paul Hammant. *Trunk Based Development*. URL: https://trunkbaseddevelopment.com/monorepos/ (visited on 03/21/2022).

[49] *Healthy breakfasts (for people who hate breakfast)*. en. Section: livewell. Apr. 2018. URL: https://www.nhs.uk/live-well/eat-well/healthy-breakfasts-recipes/ (visited on 03/28/2022).

[50] *History Of The Traditional English Breakfast*. en. Section: Culinary Culture. URL: https://englishbreakfastsociety.com/full-english-breakfast.html (visited on 03/28/2022).

[51] Adrian Holovaty and Jacob Kaplan-Moss. *Django follows this MVC pattern closely enough that it can be called an MVC framework*. Mar. 2013.

[52] *How Brits' eating habits have changed in three decades*. en-US. Mar. 2020. URL: https://www.wcrf-uk.org/about-us/press-releases/how-brits-eating-habits-have-changed-in-three-decades/ (visited on 03/21/2022).

[53] Timothy Huzar. *Fast food effects: Short-term, long-term, physical, mental, and more*. en. Dec. 2021. URL: https://www.medicalnewstoday.com/articles/324847 (visited on 04/01/2022).

[54] Mike Isaac. *Google Unwraps Ice Cream Sandwich, the Next-Generation Android OS | WIRED*. Oct. 2011. URL: https://www.wired.com/2011/10/android-ice-cream-sandwich-3/ (visited on 04/01/2022).

[55] Agnieszka Jaworowska et al. “Nutritional challenges and health implications of takeaway and fast food”. en. In: *Nutrition Reviews* 71.5 (Jan. 2013), pp. 310–318. ISSN: 00296643. DOI: 10.1111/nure.12031. URL: https://academic.oup.com/nutritionreviews/article-lookup/doi/10.1111/nure.12031 (visited on 03/28/2022).

[56] *Jest*. original-date: 2013-12-10T00:18:04Z. Apr. 2022. URL: https://github.com/facebook/jest (visited on 04/01/2022).

[57] Aaron Kassraie. *How Healthy Is Vegan Sausage, and What Brand Tastes Best?* Mar. 2021. URL: https://www.aarp.org/health/healthy-living/info-2021/vegan-sausage.html (visited on 04/01/2022).

[58] Maciej Kocik. *Yarn workspaces — monorepo beginner's guide*. en. May 2020. URL: https://medium.com/swlh/yarn-workspaces-monorepo-beginners-guide-ed89de47aa25 (visited on 03/21/2022).

[59] Janet Kuhn. “Decrypting the MoSCoW analysis”. In: *The workable, practical guide to Do IT Yourself* 5 (2009).

[60] *Libraries.io*. en. URL: https://libraries.io (visited on 04/01/2022).

[61] *Lifesum Health App*. URL: https://lifesum.com/ (visited on 03/29/2022).

[62] *Managing labels*. en. URL: https://docs.github.com/en/issues/using-labels-and-milestones-to-track-work/managing-labels#about-default-labels (visited on 03/21/2022).

[63] *Maya*. URL: https://www.maya.live (visited on 03/29/2022).

[64] *mdx-js/mdx*. URL: https://github.com/mdx-js/mdx (visited on 04/01/2022).

[65] Oliver Milman. “Meat accounts for nearly 60% of all greenhouse gases from food production, study finds”. en-GB. In: *The Guardian* (Sept. 2021). ISSN: 0261-3077. URL: https://www.theguardian.com/environment/2021/sep/13/meat-greenhouses-gases-food-production-study (visited on 03/21/2022).

[66] *NHS COVID-19 AG*. original-date: 2020-07-30T14:32:34Z. Mar. 2022. URL: https://github.com/nihp-public/covid-19-app-ios-ag-public (visited on 04/01/2022).

[67] Jakob Nielsen. *10 Usability Heuristics for User Interface Design*. en. Apr. 1994. URL: https://www.nngroup.com/articles/ten-usability-heuristics/ (visited on 04/01/2022).

[68] Maël Nison. *Introducing Yarn 2*. en. URL: https://dev.to/arcanis/introducing-yarn-2-4eh1 (visited on 03/21/2022).

[69] *Node.js*. original-date: 2014-11-26T19:57:11Z. Mar. 2022. URL: https://github.com/nodejs/node (visited on 03/21/2022).

[70] *npm*. original-date: 2018-07-05T23:26:52Z. Mar. 2022. URL: https://github.com/npm/cli (visited on 03/21/2022).

[71] *npm*. en. URL: https://www.npmjs.com/ (visited on 03/21/2022).

[72] World Health Organization. *Classifying health workers*. 2010.

[73] Mark Otto and Jacob Thornton. *Bootstrap*. en. URL: https://getbootstrap.com/ (visited on 03/20/2022).

[74] *Overleaf*. URL: https://overleaf.com.

[75] Xuan Lam Pham, Thi Huyen Nguyen, and Gwo Dong Chen. “Research Through the App Store: Understanding Participant Behavior on a Mobile English Learning App”. en. In: *Journal of Educational Computing Research* 56.7 (Dec. 2018), pp. 1076–1098. ISSN: 0735-6331, 1541-4140. DOI: 10.1177/0735633117727599. URL: http://journals.sagepub.com/doi/10.1177/0735633117727599 (visited on 03/11/2022).

[76] Jeff Sauro PhD. *5 Ways to Interpret a SUS Score*. en-US. Sept. 2018. URL: https://measuringu.com/interpret-sus-score/ (visited on 03/28/2022).

[77] Jim Lewis PhD and Jeff Sauro PhD. *Three Branches of Standardized UX Measurement – MeasuringU*. en-US. URL: https://measuringu.com/three-branches-ux/ (visited on 03/24/2022).

[78] Fabien Pifferi and Fabienne Aujard. "Caloric restriction, longevity and aging: Recent contributions from human and non-human primate studies". en. In: *Progress in Neuro-Psychopharmacology and Biological Psychiatry* 95 (Dec. 2019), p. 109702. ISSN: 02785846. DOI: 10.1016/j.pnpbp.2019.109702. URL: https://linkinghub.elsevier.com/retrieve/pii/S0278584619302702 (visited on 03/28/2022).

[79] *pip*. original-date: 2011-03-06T14:30:46Z. Mar. 2022. URL: https://github.com/pypa/pip (visited on 03/21/2022).

[80] *Pipenv*. original-date: 2017-01-20T00:44:02Z. Mar. 2022. URL: https://github.com/pypa/pipenv (visited on 03/21/2022).

[81] *Poetry*. original-date: 2018-02-28T15:23:47Z. Mar. 2022. URL: https://github.com/python-poetry/poetry (visited on 03/21/2022).

[82] *Poetry to provide a decent hook-based plugin system*. URL: https://github.com/python-poetry/poetry/issues/693.

[83] *Portion Mate Figma*. en. Section: Design. URL: https://www.figma.com/file/da6CNmv1PMIYAQaLGfSzNM/PortionMate (visited on 03/23/2022).

[84] *Portion Monitor*. URL: https://play.google.com/store/apps/details?id=com.portionMonitor.app (visited on 03/29/2022).

[85] Rachel Potvin and Josh Levenberg. "Why Google stores billions of lines of code in a single repository". en. In: *Communications of the ACM* 59.7 (June 2016), pp. 78–87. ISSN: 0001-0782, 1557-7317. DOI: 10.1145/2854146. URL: https://dl.acm.org/doi/10.1145/2854146 (visited on 03/21/2022).

[86] *pre-commit/pre-commit*. original-date: 2014-03-13T00:39:38Z. Mar. 2022. URL: https://github.com/pre-commit/pre-commit (visited on 03/21/2022).

[87] *Prettier*. original-date: 2016-11-29T17:13:37Z. Mar. 2022. URL: https://github.com/prettier/prettier (visited on 03/21/2022).

[88] *PyPI*. en. URL: https://pypi.org/ (visited on 03/21/2022).

[89] *Pytest*. original-date: 2015-06-15T20:28:27Z. Apr. 2022. URL: https://github.com/pytest-dev/pytest (visited on 04/01/2022).

[90] *Python*. original-date: 2017-02-10T19:23:51Z. Mar. 2022. URL: https://github.com/python/cpython (visited on 03/21/2022).

[91] Alcides Queiroz. *Getting rid of node_modules with Yarn Plug'n'Play*. en. Feb. 2019. URL: https://medium.com/free-code-camp/getting-rid-of-node-modules-with-yarn-plugn-play-a490e5e747d7 (visited on 03/21/2022).

[92] Fernanda Rauber et al. "Ultra-processed food consumption and indicators of obesity in the United Kingdom population (2008-2016)". en. In: *PLOS ONE* 15.5 (May 2020). Ed. by David Meyre, e0232676. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0232676. URL: https://dx.plos.org/10.1371/journal.pone.0232676 (visited on 03/11/2022).

[93] *React Native*. original-date: 2015-01-09T18:10:16Z. Mar. 2022. URL: https://github.com/facebook/react-native (visited on 03/24/2022).

[94] *React-Bootstrap*. URL: https://react-bootstrap.github.io/ (visited on 03/20/2022).

[95] *Read the Docs*. original-date: 2010-08-16T19:18:06Z. Mar. 2022. URL: https://github.com/readthedocs/readthedocs.org (visited on 04/01/2022).

[96] Guido van Rossum, Barry Warsaw, and Nick Coghlan. *PEP 8 – Style Guide for Python Code*. May 2001. URL: https://peps.python.org/pep-0008/ (visited on 03/21/2022).

[97] *Shields.io*. URL: https://shields.io/ (visited on 04/01/2022).

[98] Alan Shreve. *ngrok*. original-date: 2013-03-20T09:37:43Z. Mar. 2022. URL: https://github.com/inconshreveable/ngrok (visited on 03/21/2022).

[99] Andrew Smyk. *The System Usability Scale & How it's Used in UX*. en-US. Section: User Testing. Mar. 2020. URL: https://xd.adobe.com/ideas/process/user-testing/sus-system-usability-scale-ux/ (visited on 03/27/2022).

[100] *Snyk*. en-US. Oct. 2020. URL: https://snyk.io/ (visited on 04/01/2022).

[101] University of Glasgow SOCS. "Ethics procedures". In: URL: https://www.gla.ac.uk/schools/computing/informationforstudents/#ethicsprocedures.

[102] *Starchy foods and carbohydrates*. en. Section: livewell. Apr. 2018. URL: https://www.nhs.uk/live-well/eat-well/starchy-foods-and-carbohydrates/ (visited on 03/21/2022).

[103] "Statistics reveal Britain's 'Mr and Mrs Average'". en-GB. In: *BBC News* (Oct. 2010). URL: https://www.bbc.com/news/uk-11534042 (visited on 04/01/2022).

[104] S. Stefanov. *JavaScript patterns: Build better applications with coding and design patterns*. O'Reilly Media, 2010. ISBN: 978-1-4493-9694-7. URL: https://books.google.co.uk/books?id=WTZqecc9olUC.

[105] *Strava*. URL: https://www.strava.com/ (visited on 03/29/2022).

[106] *Style · React Native*. URL: https://reactnative.dev/docs/style (visited on 03/20/2022).

[107] *The Eatwell Guide*. en. Mar. 2016. URL: https://www.gov.uk/government/publications/the-eatwell-guide (visited on 03/21/2022).

[108] *The Eatwell Guide*. en. Section: livewell. Apr. 2018. URL: https://www.nhs.uk/live-well/eat-well/the-eatwell-guide/ (visited on 03/21/2022).

[109] *The NHS COVID-19 app*. en. URL: https://www.covid19.nhs.uk/ (visited on 04/01/2022).

[110] *Travis CI*. en-US. URL: https://www.travis-ci.com/ (visited on 04/01/2022).

[111] *TypeScript*. original-date: 2014-06-17T15:28:39Z. Mar. 2022. URL: https://github.com/microsoft/TypeScript (visited on 03/21/2022).

[112] typicode. *typicode/husky*. original-date: 2014-06-23T12:14:21Z. Mar. 2022. URL: https://github.com/typicode/husky (visited on 03/21/2022).

[113] C. Lee Ventola. "Mobile Devices and Apps for Health Care Professionals: Uses and Benefits". In: *Pharmacy and Therapeutics* 39.5 (May 2014), pp. 356–364. ISSN: 1052-1372. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4029126/ (visited on 03/11/2022).

[114] VentureBeat. *Study: Mobile app usage grows 35%, TV & web not so much*. en-US. Dec. 2012. URL: https://venturebeat.com/2012/12/05/mobile-app-usage-tv-web-2012/ (visited on 03/11/2022).

[115] *Visual Studio Code*. original-date: 2015-09-03T20:23:38Z. Mar. 2022. URL: https://github.com/microsoft/vscode (visited on 03/21/2022).

[116] *Water, drinks and your health*. en. Section: livewell. June 2018. URL: https://www.nhs.uk/live-well/eat-well/water-drinks-nutrition/ (visited on 03/21/2022).

[117] *Who feels every meal MUST have meat? If so, why?* Reddit Post. Jan. 2022. URL: www.reddit.com/r/AskUK/comments/s35u3s/who_feels_every_meal_must_have_meat_if_so_why/ (visited on 03/21/2022).

[118] Alex Williams. *GitHub Pours Energies into Enterprise – Raises $100 Million From Power VC Andreessen Horowitz*. en-US. Sept. 2012. URL: https://social.techcrunch.com/2012/07/09/github-pours-energies-into-enterprise-raises-100-million-from-power-vc-andreesen-horowitz/ (visited on 03/28/2022).

[119] Allen Wirfs-Brock and Brendan Eich. "JavaScript: the first 20 years". en. In: *Proceedings of the ACM on Programming Languages* 4.HOPL (June 2020), pp. 1–189. ISSN: 2475-1421. DOI: 10.1145/3386327. URL: https://dl.acm.org/doi/10.1145/3386327 (visited on 03/21/2022).

[120] *Xamarin*. en-US. URL: https://dotnet.microsoft.com/en-us/apps/xamarin (visited on 04/01/2022).

[121] *Yarn*. original-date: 2016-01-19T17:39:16Z. Mar. 2022. URL: https://github.com/yarnpkg/yarn (visited on 03/21/2022).

[122] Evan You. *yyx990803/yorkie*. original-date: 2018-01-06T04:26:07Z. Mar. 2022. URL: https://github.com/yyx990803/yorkie (visited on 03/21/2022).

Repository

[123] Portion Mate. GitHub. ./.github/workflows/server-checks.yml. URL: https://github.com/ineshbose/portion-mate/blob/cb05d5b7505df7d7e65550ef01d22ef2ada0a47c/.github/workflows/server-checks.yml#L15.

[124] Portion Mate. GitHub. ./package.json. URL: https://github.com/ineshbose/portion-mate/blob/c51ee3f32d05df64115746716 9e9659732202b7b/package.json#L22.

[125] Portion Mate. GitHub. ./pyproject.toml. URL: https://github.com/ineshbose/portion-mate/blob/cb05d5b7505df7d7e65550ef01d22ef2ada0a47c/pyproject.toml.

[126] Portion Mate. GitHub. ./src/App.tsx. URL: https://github.com/ineshbose/portion-mate/blob/c51ee3f32d05df64115746716 9e9659732202b7b/src/app/App.tsx#L28.

[127] Portion Mate. GitHub. ./src/app/api/index.ts. URL: https://github.com/ineshbose/portion-mate/blob/c51ee3f32d05df64115746716 9e9659732202b7b/src/app/api/index.ts#L20.

[128] Portion Mate. GitHub. ./src/app/constants/Styles/index.ts. URL: https://github.com/ineshbose/portion-mate/blob/c51ee3f32d05df641157467169e9659732202b7b/src/app/constants/Styles/index.ts#L7.

[129] Portion Mate. GitHub. ./src/app/screens/BottomTab/HomePage.tsx. URL: https://github.com/ineshbose/portion-mate/blob/82a726e58c501ab2b4fe9c5f18ac399fb29b0422/src/app/screens/BottomTab/HomePage.tsx#L123.

[130] Portion Mate. GitHub. ./src/app/types/api.ts. URL: https://github.com/ineshbose/portion-mate/blob/c51ee3f32d05df64115746716 9e9659732202b7b/src/app/types/api.ts#L28.

[131] Portion Mate. GitHub. ./src/server/rest/tests.py. URL: https://github.com/ineshbose/portion-mate/blob/cb05d5b7505df7d7e65550ef01d22ef2ada0a47c/src/server/rest/tests.py#L41.

[132] Portion Mate. GitHub. ./src/set-env.js. URL: https://github.com/ineshbose/portion-mate/blob/c51ee3f32d05df6411574671 69e9659732202b7b/src/set-env.js#L26.

[133] Portion Mate. GitHub. Issue #116. URL: https://github.com/ineshbose/portion-mate/issues/116.

[134] Portion Mate. GitHub. Issue #28. URL: https://github.com/ineshbose/portion-mate/issues/28.

[135] Portion Mate. GitHub. Issue #41. URL: https://github.com/ineshbose/portion-mate/issues/41.

[136] Portion Mate. GitHub. Issue #44. URL: https://github.com/ineshbose/portion-mate/issues/44.

[137] Portion Mate. GitHub. Issue #52. URL: https://github.com/ineshbose/portion-mate/issues/52.

[138] Portion Mate. GitHub. Issue #70. URL: https://github.com/ineshbose/portion-mate/issues/70.

[139] Portion Mate. GitHub. Issue #71. URL: https://github.com/ineshbose/portion-mate/issues/71.

[140] Portion Mate. GitHub. Issue #73. URL: https://github.com/ineshbose/portion-mate/issues/73.

[141] Portion Mate. GitHub. Issue #93. URL: https://github.com/ineshbose/portion-mate/issues/93.

[142] Portion Mate. GitHub. Pull Request #120 Files. URL: https://github.com/ineshbose/portion-mate/pull/120/files.

[143] Portion Mate. GitHub. Pull Request #69. URL: https://github.com/ineshbose/portion-mate/pull/69.

[144] Portion Mate. GitHub. Pull Request #79. URL: https://github.com/ineshbose/portion-mate/pull/79.

[145] Portion Mate. GitHub. Pull Request #83. URL: https://github.com/ineshbose/portion-mate/pull/83.

[146] Portion Mate. GitHub. Requirements Wiki. URL: https://github.com/ineshbose/portion-mate/wiki/Requirements.

[147] Portion Mate. GitHub. Research Wiki. URL: https://github.com/ineshbose/portion-mate/wiki/Research.

[148] Portion Mate. GitHub. src/app/navigation/RootNavigator.tsx. URL: https://github.com/ineshbose/portion-mate/blob/82a726e58c501ab2b4fe9c5f18ac399fb29b0422/src/app/navigation/RootNavigator.tsx#L13.

[149] Portion Mate. GitHub. Users Wiki. URL: https://github.com/ineshbose/portion-mate/wiki/Users.

[150] Portion Mate. GitHub. Wireframes Wiki. URL: https://github.com/ineshbose/portion-mate/wiki/Wireframes.