



# FOOD LOG

Hybrid App for Monitoring Food Intake

Final Report

**Name:** Ignas Baranauskas  
**Student No:** 20099691  
**Course:** Higher Diploma in Computer Science

## Declaration

I declare that the work which follows is my own, and that any quotations from any sources (e.g. books, journals, the internet) are clearly identified as such by the use of 'single quotation marks', for shorter excerpt and identified italics for longer quotations. All quotations and paraphrases are accompanied by (author, date) in the text and a fuller citation is the bibliography. I have not submitted the work represented in this report in any other course of study leading to an academic award.

Student Ignas Baranauskas

Date 02/04/2024

## Acknowledgements

I would like to thank my friends and family for their steady support, understanding, and encouragement throughout the project journey. Their patience, support, and belief in my skills have provided the necessary motivation to overcome obstacles and persevere in pursuing my goals.

## Titles

Commercial Title

Food Log

Academic Title

Hybrid app for monitoring food intake (Food Log)

## Preface

The report should be read together with the following resources:

Project planning – GitHub projects:

[Planning · FoodLog \(github.com\)](#)

GitHub Project Repositories: FoodLog (Back End) and FoodLog-Flutter (Flutter Front End), can be found as submodules:

[Ygnas/foodlog-project \(github.com\)](#)

Project page:

[Food Log – Ignas Baranauskas \(ygnas.github.io\)](#)

Built flutter application for the Android device can be found:

[Releases · Ygnas/FoodLog-Flutter \(github.com\)](#)

## Abstract

Food Log, a food tracker application designed specifically for those who prefer simplicity and convenience. This application is perfect for individuals who want to take control of their eating habits without the hassle of manually tracking calories. It allows individuals to monitor their food intake, it's not just about logging meals, it's about making informed choices that align with your health and wellness goals. With Food Log, you can maintain a comprehensive record of all your daily consumptions. This allows you to gain insights into your eating patterns, helping you make healthier choices.

## Table of Contents

Declaration .....	1
Acknowledgements .....	2
Titles .....	3
Commercial Title .....	3
Academic Title .....	3
Preface .....	4
Abstract .....	5
Table of Contents .....	6
Table of Figures .....	8
1. Introduction .....	9
1.1. Background .....	9
1.2. Problem Definition .....	9
1.3. Solution .....	9
1.4. Project Goal .....	9
1.5. Unique Features .....	10
2. Research and Analysis .....	11
2.1. Requirements Analysis .....	11
2.1.1. Mobile Application .....	11
2.1.2. Back End .....	11
2.1.3. CI/CD .....	12
2.2. Market Analysis .....	12
2.3. Feature analysis .....	13
3. Modelling .....	15
3.1. Back End and Database .....	15
3.1.1. Data Modelling .....	16
4. Planning .....	17
4.1. Methodology .....	17
4.2. Planning Tool .....	17
4.3. Getting Started .....	17
5. Implementation .....	19
5.1. Sprint 1 .....	19
5.2. Sprint 2 .....	20
5.3. Sprint 3 .....	21
5.4. Sprint 4: .....	22
5.5. Sprint 5: .....	24

5.6. Sprint 6:.....	26
6. Reflection .....	27
6.1. Problems Encountered.....	27
6.2. What I learned .....	28
6.3. Achievements.....	29
6.4. Future Development .....	29
7. AI Usage in the project.....	30
Bibliography .....	31
Appendix A – GitHub Sprint Report .....	33
Appendix B – Kubernetes Deployment .....	35
Appendix C – Flutter Wireframes.....	37
Appendix D – Ethics Checklist .....	40



## Table of Figures

Figure 1 - Planned Technologies.....	11
Figure 2 - Technologies Used .....	11
Figure 3 - Application Comparison.....	13
Figure 4 - JSON listing.....	15
Figure 5 - ER diagram. ....	16
Figure 6 - GitHub board estimation points.....	17
Figure 7 - System Diagram. ....	19
Figure 8 - Docker build image. ....	20
Figure 9 - CI/CD. ....	21
Figure 10 - Login, Register screen, and their validation. ....	22
Figure 11 - Profile screen, bar chart visualisation.....	23
Figure 12 - Delete Account Confirmation.....	24
Figure 13 - Location, Map Screen.....	24
Figure 14 - Application Notifications.....	25
Figure 15 - Sprint 1.....	33
Figure 16 - Sprint 2.....	33
Figure 17 - Sprint 3.....	33
Figure 18 - Sprint 4.....	33
Figure 19 - Sprint 5.....	34
Figure 20 - Sprint 6.....	34
Figure 21 - Kubernetes config map. ....	35
Figure 22 - Kubernetes make file. ....	35
Figure 23 - Kubernetes deployment.....	35
Figure 24 - Kubernetes service.....	36

# 1. Introduction

## 1.1. Background

I worked in a hotel before becoming a software engineer. This career change was significant for me and those close to me.

The move to the tech industry caused me to lose track of what I was eating, leading to unhealthy habits. I realised my struggle, and it inspired me to create an application – a tool to simplify food tracking, give insights on dietary patterns, and encourage a more informed and balanced approach to eating. I believe that the issue of losing track of time is not unique to me, and I hope this project will not only help me manage dietary needs but may be beneficial to others having similar issues.

## 1.2. Problem Definition

Many individuals need help maintaining a healthy diet due to their hectic lifestyles and the abundance of food choices. Software engineers, among others, regularly find it hard to keep their diet healthy due to the pressures of their profession and the variety of available food options. Meanwhile, others occupied with work or other activities sometimes forget to eat.

The Food Log application will address this common issue by providing a potential solution to the difficulties of managing one's diet. It is not by counting the calories and providing the user with more work, filling the stats of each item they put in their mouth, giving them insights into their eating habits, overeating or even simple reminders for those who do not consume enough.

## 1.3. Solution

Food Log is a mobile application for those who desire a modern and simple solution for managing their daily consumption. The aim is to provide the community with an application that allows them to seamlessly track their food intake and receive valuable notifications about it. The application would contain a complete person record, allowing a deeper understanding of their habits.

The proposed application is a multi-platform Flutter application that has a supporting backend written in Go Lang. It would enable it to be used on a mobile phone or a simple browser.

## 1.4. Project Goal

This project aims to create a multi-platform Flutter mobile application that allows users to gain insights into their eating habits, eventually leading them to become healthier. The application would offer account management, food item management, daily consumption tracking, search and filter capabilities, detailed food item information, and consumption monitoring. It would also provide

reminders and notifications for irregular patterns and have some social features allowing listings to be marked for others to view and rate.

### 1.5. Unique Features

In summary, this application will be simple to use, allow users to control their food logging, and help them navigate a healthier lifestyle. Unlike all the other complex calorie-counting applications where a user must use complicated inputs to allow for calorie-counting, this application will try to be as simple as possible, allowing members to log their meals at their own pace, with as little input as necessary. It will enable users to quickly see their patterns and make meaningful choices.

By making the application simple to use, the aim was to make it accessible to a bigger community, where users may get overwhelmed or think that it is too time-consuming.

## 2. Research and Analysis

### 2.1. Requirements Analysis

When choosing the tools and technologies for this project, decisions were made to support the ongoing learning opportunities and skill progression. Compared to my work placement experiences and coursework, the chosen technologies provide a balanced combination of knowledge gained during my work placement and development that I should get by trying new tools.

*Figure 1 - Planned Technologies*

Mobile application	Back end	CI/CD
<ul style="list-style-type: none"><li>Flutter</li></ul>	<ul style="list-style-type: none"><li>Golang</li><li>Gin or Chi framework</li><li>Firebase</li></ul>	<ul style="list-style-type: none"><li>Docker</li></ul>

*Figure 2 - Technologies Used*

Mobile application	Back end	CI/CD
<ul style="list-style-type: none"><li>Flutter</li></ul>	<ul style="list-style-type: none"><li>Golang</li><li>Chi framework</li><li>Firebase (Realtime Database)</li><li>Firebase (Storage)</li></ul>	<ul style="list-style-type: none"><li>GitHub packages</li><li>Docker</li><li>Kubernetes</li></ul>

#### 2.1.1. Mobile Application

The Food Log mobile application will use Flutter. 'Flutter is an open-source framework by Google for building beautiful, natively compiled, multi-platform applications from a single codebase' (Flutter, 2024). It uses Dart programming language, which is easy to pick up if you know any other programming language, like JavaScript. The main building block of it is widgets; they are reusable and can be seamlessly integrated throughout the application. It should be easy to pick up for anyone who attempted to use React for web applications (Freeborn, 2022).

Flutter competes with React Native, Xamarin Framework and Native Android framework. Each has its own set of advantages. Flutter is recognised for its flexibility, hot-reloading feature for faster development, and multi-platform capabilities, which allow the application to be deployed on multiple platforms – the main reason it was chosen for this project.

#### 2.1.2. Back End

The backend of the Food Log application will use the power of Golang, 'An open-source programming language supported by Google' (Google, 2024).

While I was still on work placement, the primary language used was Golang, which was easy to learn the basics and begin working.

When choosing the framework to use for the backend API, I had two choices: Gin Web Framework or Chi. I chose Chi as this framework allows me to create large REST APIs, as I had tried Gin before this and wanted to gain new skills.

Firebase will be used as a database for backend listings and will manage all the authentication. It was chosen because it is easy to use and can be seamlessly integrated, eliminating the complexity of my application. It allows users to register, log in, and securely store relevant data.

There will be no actual data in my project, I will create everything.

### 2.1.3. CI/CD

Numerous mature Continuous Integration/Continuous Deployment tools and pipelines are available on the market. Like Jenkins, it is an 'open-source tool that helps you automate parts of software development, like building, testing, and deploying' (Wikipedia contributors, 2024). Although I believe it is a great tool, I believe it is too tricky for my needs as I would need to host my own Jenkins server. That is why I choose to utilise GitHub Actions because of its seamless integration with GitHub repositories, which allows me to automate my software development tasks directly within my repository. My pull requests can easily trigger the workflows and are secured by GitHub's secure way of managing my secrets.

Another part of it will be using Docker for my backend application. It was chosen because of easy maintenance and reproducible environments.

## 2.2. Market Analysis

In the area of open-source nutrition tracking projects, a significant gap exists for a user-friendly and actively maintained solution. Several existing open-source food diary projects on platforms such as GitHub exhibit signs of ageing, with outdated technologies and no community involvement. This is a challenge for users seeking modern features and continuing support. There is a lack of solutions that prioritize simplicity and user-friendliness. Many existing projects may be challenging for individuals seeking a straightforward nutrition tracking experience.

The market analysis shows a gap in the availability of a modern, easy-to-use, and continuously maintained open-source nutrition tracking application. The Food Log project aims to fill this gap by providing a fresh alternative that embraces simplicity and encourages community involvement for sustained development.

I have selected two similar projects to compare too, these are:

## OpenNutriTracker

<https://github.com/simonoppowa/OpenNutriTracker>

With my limited time searching for similar applications, this was the only modern one I found. It has a lovely design and is relatively straightforward to understand and use if you only want to count on nutritional value. If you are using the built-in barcode scanner, adding your meals requires you to fill in all the values yourself; that is much work, and most people do not want that.

The app does not have a user system, so all your data can be easily lost.

## Food Diary

[olegnet/food-diary-android \(github.com\)](https://github.com/olegnet/food-diary-android)

Food Diary is only for nutritional values. It does not have a modern user interface and it is not very easy to use, but it provides the ability to export and import your data. In general, it does operate similarly to the previously mentioned application.

All the searches on GitHub were done only for open-source applications, and none were found that would allow you to track the meals or food you consume without nutritional values. It makes it difficult to track simple meals without the scanner feature, and most users do not have time to do so.

## 2.3. Feature analysis

This is the current proposed features of my Food Log application:

- Users can create an account and log in.
- Create, update, and delete food items.
- Like other user's posts.
- Comment on other user's posts.
- Members can track daily food consumption.
- Ability to search and filter your food items.
- Visualise listings by date.
- User can set to get reminders.

Figure 3 - Application Comparison

	User system	Create listings	Update listings	Rate the listings	Nutritional values	Modern User Interface	Visualisations
Food Log	✓	✓	✓	✓		✓	✓
OpenNutriTracker		✓			✓	✓	
Food Diary		✓	✓		✓		

As seen from the compared applications in Figure 3, only one main feature is shared with my application: a type of listing created. Other applications focus on tracking nutritional values, that my application will not have. I will focus more on visualisations and reminders, with, I hope, some social aspects to make it engaging for users to return.

## 3. Modelling

### 3.1. Back End and Database

All the data in my application is stored in Firebase Realtime Database, a NoSQL cloud-hosted database. It stores its data in an easy-to-understand JSON format and synchronises to every connected client in real-time.

Figure 4 - JSON listing.



Firebase was chosen because it is easy to understand, has excellent examples, and removes the need to manage my database service.

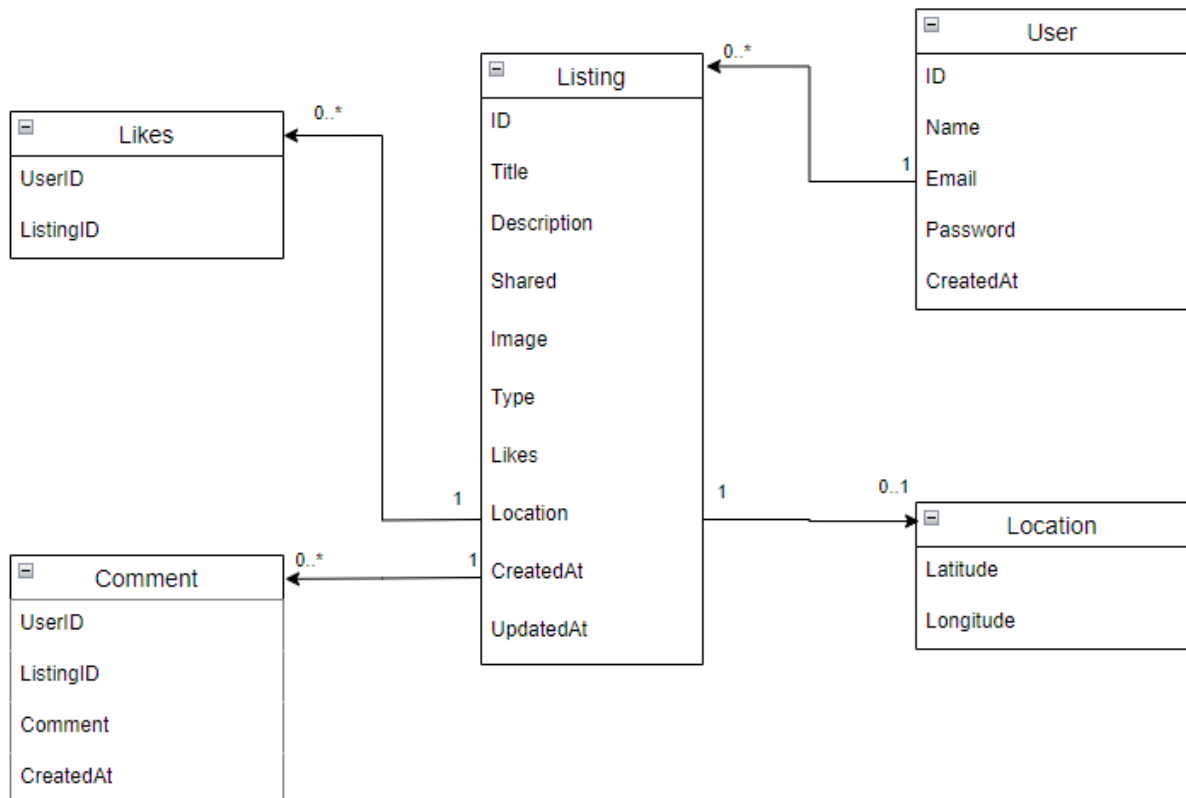
Food Log requires a database to make this application cross-platform so that the data can be retrieved on any client or device, making it easy for the user to switch devices and still be able to use the application.



### 3.1.1. Data Modelling

Even though the database in use is not ‘a relational database, that stores information in tabular form, with rows and columns representing different data attributes and the various relationships between the data values’ (Amazon, 2024). I incorporated an Entity Relationship diagram to better understand the data flow and model of the data.

Figure 5 - ER diagram.



## 4. Planning

### 4.1. Methodology

The Agile methodology was chosen for this project so I could work on it in smaller parts and add new features as time went on instead of doing so in one go. Working in smaller parts ensures I can concentrate on a specific feature or functionality with greater focus. This is especially valuable in environments where priorities may change, or new ideas emerge during the development. This allows me to focus on the requirements more efficiently and change parts of it in a way that makes sense and will enable me to evolve it quickly.

### 4.2. Planning Tool

For this project, I have chosen to use 'GitHub Projects, an adaptable, flexible tool to plan and track my work' (GitHub, 2024). Rather than enforcing a single methodology on the user, it provides various features, allowing me to customise it to my needs and processes. It lets me create multiple views, filter, or sort them, and add custom fields to track the required metadata like 'Estimate' story points shown in Figure 6.

Other project management tools I considered were Trello and Jira by Atlassian. My experience with Trello has been very positive. It offers a simple, easy-to-understand user interface and a Kanban-based system allowing for easy task organisation. On the other hand, Jira is a more robust and complex tool with a lot of advanced features and systems. Although my experience with it is limited, it was used in my workplace for managing customer-facing issues and projects.

In the end, the tool chosen was influenced by my past experiences.

### 4.3. Getting Started

The work will be completed in two-week iterations, with an issue for each ticket. Within each sprint, I focus on specific topics and work on them until they are finished. That allows me to assess and adjust my priorities as needed quickly. Each story has its story points, which help me predict how long it should take me to deliver a specific feature of the backlog. I assign story points in relation to the issue's complexity and the amount of work it needs (Atlassian, 2024).

*Figure 6 - GitHub board estimation points.*

	Title	...	Status	...	Iteration	...	Estimate	...
1	🕒 Set up Git repo's #2		Done	▼	Iteration 1	▼	3	
2	🕒 Write introduction section of report #3		Done	▼	Iteration 1	▼	8	
3	🕒 Background #4		Done	▼	Iteration 1	▼	3	
4	🕒 Problem Definition #5		Done	▼	Iteration 1	▼	3	
5	🕒 Project Goal #7		Done	▼	Iteration 1	▼	3	
6	🕒 Solution #6		Done	▼	Iteration 1	▼	3	
7	🕒 Research using Gin or Chi framework #8		Done	▼	Iteration 1	▼	8	
8	🕒 Write research and analysis section of report #9		Done	▼	Iteration 1	▼	5	
9	🕒 Requirements analysis #10		Done	▼	Iteration 1	▼	3	
10	🕒 Create git repo's #1		Done	▼	Iteration 1	▼	3	
11	🕒 Market analysis #11		Done	▼	Iteration 1	▼	3	
12	🕒 Feature analysis #12		Done	▼	Iteration 1	▼	3	

## 5. Implementation

Figure 7 - System Diagram.

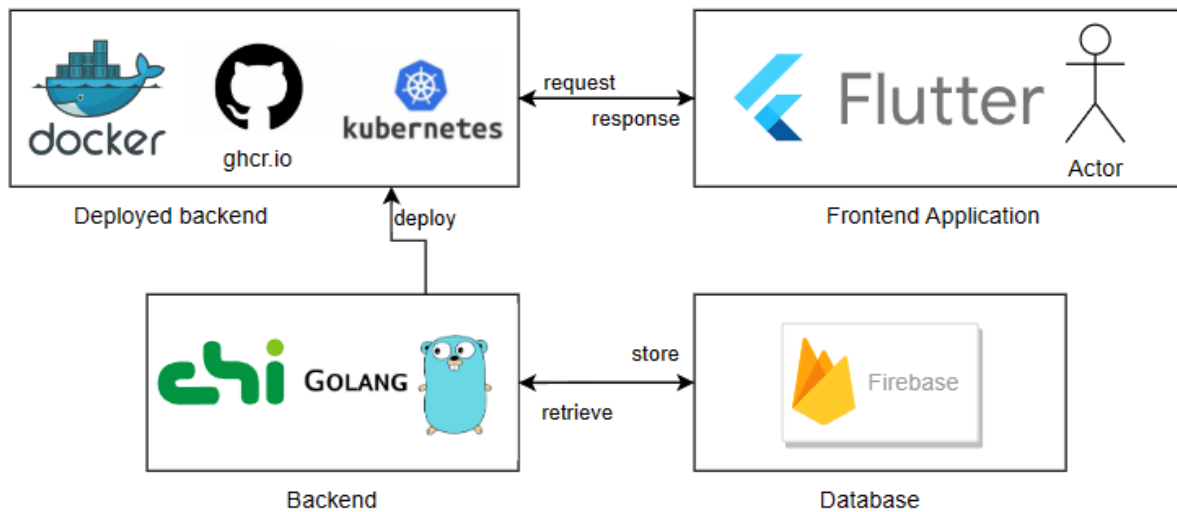


Figure 7 Shows the architecture of the system diagram. The Flutter frontend application is where the user interacts with the system in a single application. All the interactions happen in a Kubernetes-hosted backend that gets its image from the GitHub packages built using Docker. The application sends its request and gets the response from that hosted image written in Golang, which interacts with the Firebase API to store and retrieve the necessary data.

### 5.1. Sprint 1

During the initial sprint, I spent time setting up all the essential elements necessary for the project's development. I also created the project version control repositories, which allowed me to keep track of the changes made to the code.

Research and analysis were conducted to explore the relevant frameworks, such as Gin and Chi, which helped me determine whether they would suit and meet my project requirements.

At the same time, I did a market analysis to find similar projects and get ideas I could incorporate into my final application. This step helped me see a lack of modern open-source finished projects about this subject, mainly any project that would be of any interest, as all of them lacked any feature that could keep the community engaged and wanting to return.

In summary, the first sprint established all the infrastructure and set the main project goals by conducting research and analysis. This set me up for the following sprints, providing a clear path for the project.

## 5.2. Sprint 2

I created application wireframes and model diagrams during the second sprint for my Food Log project. These models were used in the application to better understand the database architecture and data flow. Afterwards, storyboards were created to guide development and app design.

This sprint was used for the initial code for the backend, where I created the base for a backend application and connected it to the firebase. For the initial prototype I have used the following as an example (Vubon, 2023). The first thing I achieved was implementing the register and login routes for my backend API. For authentication, I did not choose the easy route and just used the existing Firebase Authentication for simplicity. However, I wanted to challenge myself and build my own auth system. It was done by only keeping the user's hashed and salted password. This method ensures the security of user data, as the actual password is never directly stored. Upon successful verification of the user credentials, a JWT token is issued; it serves as proof of authentication, allowing the user to interact with the application securely.

After implementing the authentication system, I implemented simple CRUD functionality for the application's listings. This feature is one of the main parts of this dynamic application, allowing users to interact with it. All operations can only be used with a valid JWT token, ensuring that only the authenticated can change their data. Firebase Realtime Database is used for storing listings and all user data.

The end of the sprint was used to implement Continuous Integration (CI) and Continuous Delivery (CD) to enhance my development process. The CI performs the backend tests I wrote for every route to ensure it functions as intended. These tests were automated using GitHub Actions, triggered by every pull request, which is set up to prevent any code from being merged without them passing.

Figure 8 - Docker build image.

```
FROM golang:latest as builder

WORKDIR /app

COPY . .

RUN go mod download && go build -o api .

FROM alpine:latest

WORKDIR /app

COPY --from=builder /app/api .

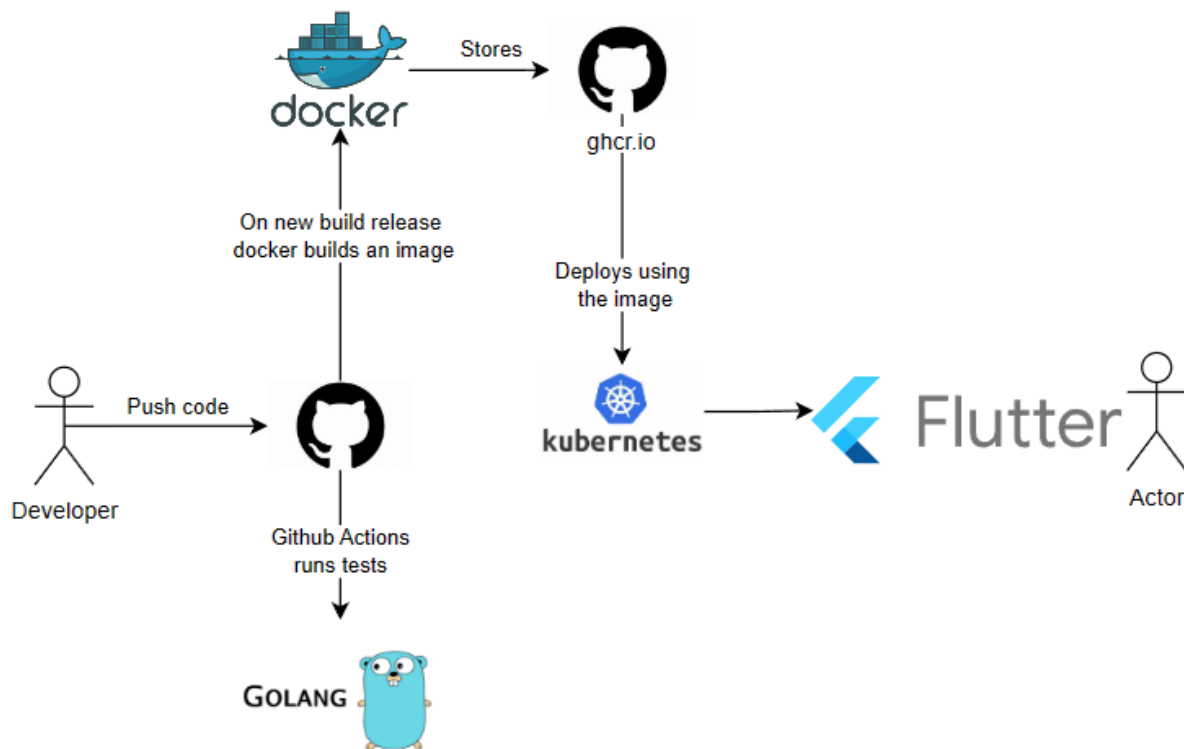
RUN apk add --no-cache libc6-compat

EXPOSE 3000

CMD ["/api"]
```

At the same time, the CD workflow is initiated, which tries to build the Go Lang code and make a Docker image out of it that is used to start the backend. The Docker image is built using the multi-stage builds feature, which first makes the image itself, moves it to the second stage with only the necessary libraries, and runs it. This feature optimises Docker images and makes them easy to read and maintain (Docker, 2024). This early process helped me in future sprints and made my code more reliable by identifying issues early.

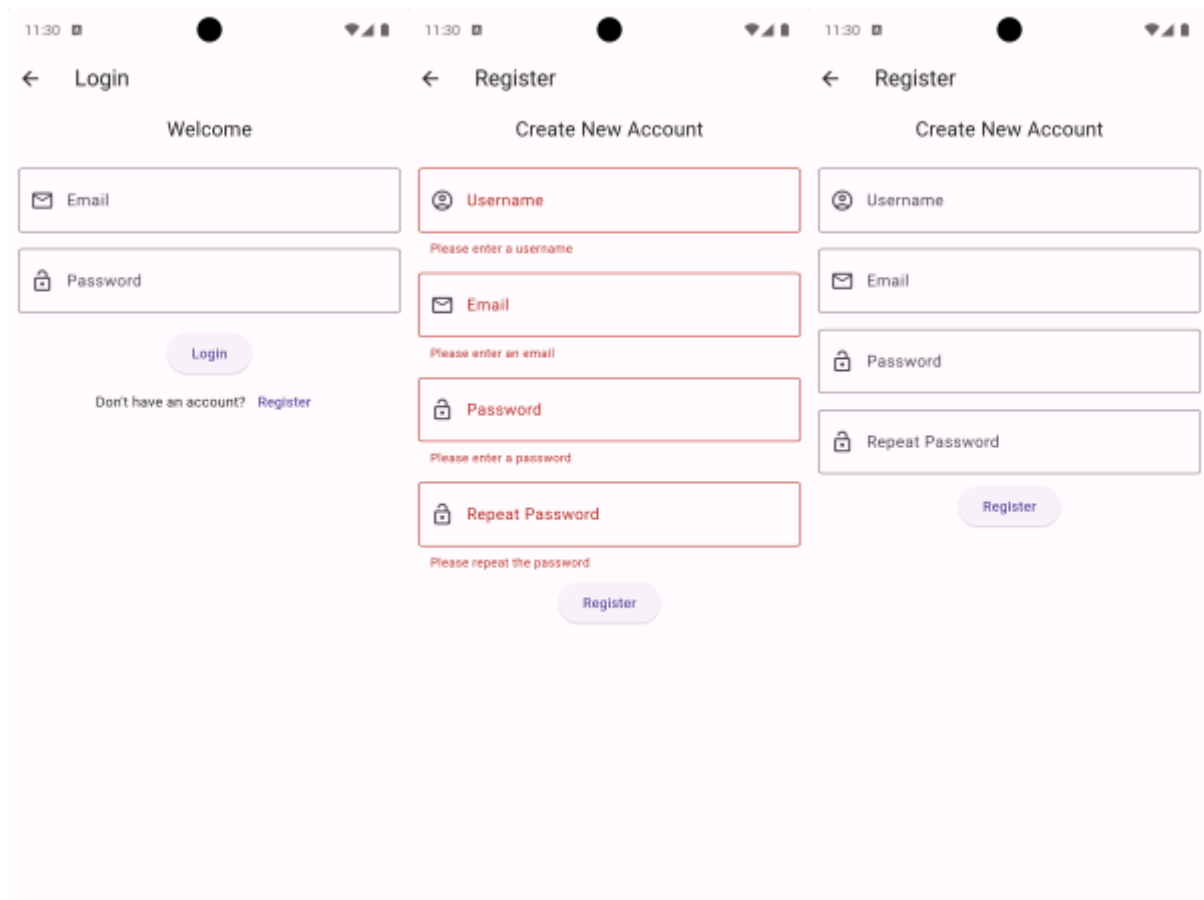
Figure 9 - CI/CD.



### 5.3. Sprint 3

In the third sprint, I focused on implementing the CRUD functionality in the Flutter mobile application. It started with implementing the register and login by using the backend API. I quickly realised that it would not be as easy to do. So, I have started looking into the recommended Flutter state management plugins to allow me to have a simpler state of user data. After testing a few of those, I decided to work with one named **Provider**; it provides a simplified allocation of resources and dramatically reduces boilerplate (dash-overflow.net, 2024). It allowed me to create simple provider classes for users and listings to view that data quickly on different screens. After a successful login, the only data the application saves is the JWT token, which is used for the subsequent request as proof of authentication. I have used the following example while implementing the authentication (Chi contributors, n.d.).

Figure 10 - Login, Register screen, and their validation.



Both login and register screens have data validation that prevents the user from submitting form data to the API if validation rules do not comply with the settings. That can be seen in Figure 10.

After I got authentication working, I started implementing the rest of the CRUD functionality so that users could better control their listings. I used an asynchronous approach to ensure smooth interaction with my API, which allowed me to make my app responsive even during long network requests. It was all done using the Flutter Future class, which, instead of blocking the application until results are calculated, returns a Future, which will eventually complete and get the result (Flutter, 2024).

Lastly, the sprint was finished by refactoring the backend to allow passing all the credentials files for Firebase using environment variables. I also rate-limited the backend API to prevent future spam of its endpoints by nefarious users and make it more responsive for the rest.

#### 5.4. Sprint 4:

The beginning of sprint four was spent implementing the community screen in the mobile application, which allows community members to mark their posted listings as shared, and that will make it show up on that screen. Each shared listing can be liked, or any users can leave a comment. Like functionality was done to prevent a person from liking the same listing many times and only removes a like if it was pressed again. On the other hand, there can be many comments from the

same person. All this is done so that every functional item is represented as a separate widget and can be easily reused anywhere.

The community screen was implemented to make it more entertaining for the users and give them a reason to interact with and engage with others making it more interesting to return. In the end, it is implemented in a way that only filters all user listings by 'shared' condition, and if that one is true, it gets displayed there.

The next step was to create a search function that allows a user to search its created listings by any data inside them, even by the date they were created, for the implementation this was used as an example (Sharan, 2023). That allows for an easy way to find the data you are looking for. For example, a user would like to know what he had on a specific date or dates.

Lastly, a bar chart visualisation was created in the user profile screen so that a person can see his daily consumption, separated by 'Snacks' and 'Other', at a glance.

Figure 11 - Profile screen, bar chart visualisation.



The bar chart provides users with a clear and concise overview of their weekly patterns. It allows them to quickly see differences between days and check how they are keeping up with their goals.

The ability to add or update an image for a given listing offers its users a more visual history and enhanced listings. By allowing images in the listings, users can showcase their items and capture memorable moments for that single entry.

This feature allows users to capture a picture with their phone camera and directly upload it to Firebase Storage. The listing is updated with a web link to an image from the storage. Additionally,

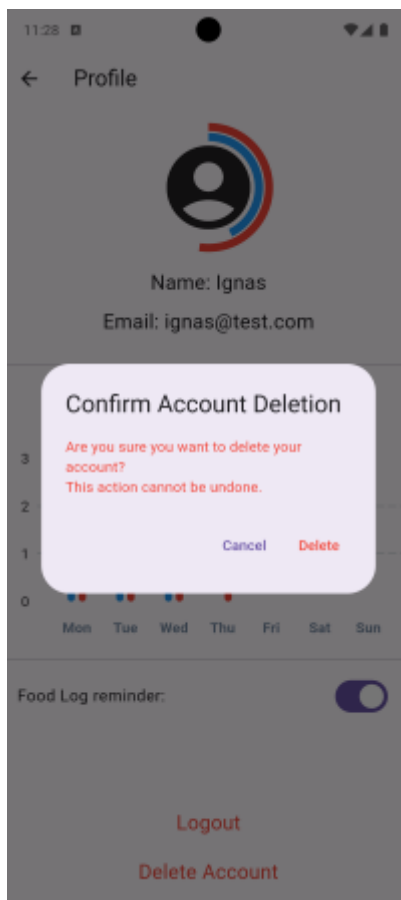


the inclusion of images made the community page interesting, encouraging interaction with other community members and giving the users more reasons to come back and share more content.

### 5.5. Sprint 5:

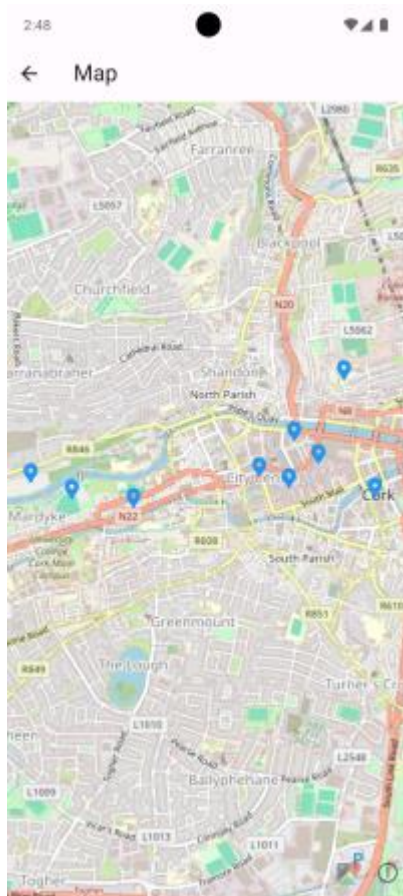
During the fifth sprint, my focus was on implementing some key features and further enhancing the application's user experience. One of those features implemented was allowing a user to delete their account. This aligns well with data protection regulations, where a user must have a right to be forgotten. This feature allows users to better control their data and privacy.

*Figure 12 - Delete Account Confirmation.*



Geolocation functionality allows for the location to be saved for a listing if the permission is granted. Furthermore, saved listings with location data can now be displayed on a map. This provides users with their activity represented on a map, serving as a reminder of where the food was consumed or even helping to keep track of places the user has been.

*Figure 13 - Location, Map Screen.*



Local notifications were implemented to keep users more engaged and serve as a reminder to keep using the application and track their daily consumption. They were integrated into the user profile, where users can choose to enable “Food Log reminders” or disable it if they do not want that functionality.

*Figure 14 - Application Notifications.*



As seen in Figure 14 - Application Notifications. the Application, it does not need to be running for the notification to arrive; it even works if the phone is locked.

### 5.6. Sprint 6:

During the sixth sprint, I focused on automating backend deployment into Kubernetes to ensure it could be deployed quickly. This was chosen to make the backend as reliable as possible, allowing it to have a few load-balanced replicas if the load increases. More about the deployment can be seen here [Appendix B – Kubernetes Deployment](#).

The second part was writing a few tests for the Flutter main classes. I did it to show that if more time was given, I could have done more testing, but it's more than nothing. Its tests are running in GitHub actions in the same way as the backend.

Lastly, I did some refactoring, a few user experience changes, and the last workflow in GitHub to automate the Android executable build. It works by waiting for a new release to be done and then running an action to build an executable and upload it into the newly created release. This saves a lot of time and ensures that new releases are readily available for users without requiring any manual work from me.

## 6. Reflection

### 6.1. Problems Encountered

#### **Android Notification**

From the beginning, I intended to have a simple notification reminder where users could choose when to be notified and even a condition that would have to be met. Unfortunately, when I started implementing the notifications, it became clear that it would be challenging to do correctly. The main reason is that I did not use cloud notification providers and tried to implement only local phone reminders, where the phone must work in the background. The thing is, my phone manufacturer blocks most of the work in the background, and that makes the notifications late. Moreover, that made any testing with accurate notifications very difficult and unreliable, as they were usually late for even a few days and or could arrive in bulk.

To do any notifications, I scaled down what they will do, only leaving simple reminders that come every single day. Furthermore, ignore them being late, even though forcing the application to send a notification using the Android debug bridge always works.

#### **Testing**

Testing my code was part of my plans, and I wrote tests for every single route in the backend. I kept updating it when I added any new features. I did have the same plans for the Flutter application to have some tests. After checking the proper way of doing them, I see a gap in my development process, mainly in time management, as I would have run out of time writing all the tests for every widget. The backend testing also covers the main parts of the front end because, in most cases, the actual calculations happen in the backend, and the Flutter application only uses it.

The only testing done in the Flutter application is for the listing and user class; all the tests run in the GitHub actions on every pull request. They work in the same way as backend ones, not allowing any merging of code if any of the tests fail. At the same time, another action is running to build an Android application that makes sure that even if my unit tests pass, there must be no mistakes or syntax errors in the code base.

I still see it as a massive problem because tests ensure the application's reliability and stability. As the application becomes more complex, this becomes a more significant issue, making further development difficult.

In the future, I will try to follow the Test-Driven Development model, writing tests as I go and making them a priority in my sprint planning.

## 6.2. What I learned

### Flutter

It was a massive learning curve, making it more complex than anticipated. The only similar thing I have tried before is React, as I think it has similar state management. Even though at the start of the project, I spent too much time trying to get some things working and had a hard time understanding why it worked like that, in the end, I improved a lot and could understand and solve all the issues quickly.

### Backend Development

Writing the backend in Golang was way more straightforward than writing the Flutter application, and I enjoyed working on the backend code more. In addition to using the same language while on the work placement, I found it more straightforward. Nonetheless, I still learned a lot. I explored new frameworks like Chi and improved my code management skills. I look forward to expanding my skills further in the future.

### Planning

I learned a lot about planning during the development of my project, and I now understand how important it is to the whole cycle of writing software. In the beginning, I had a well-laid-out plan for the upcoming sprints, but in the later ones, when I had run out of the main simple features, I was struggling to keep up with adding new planned things to do. Furthermore, I made new ones while working on something and seeing what was needed next. Ultimately, I still prefer planning as I work, which feels more natural.

### Wireframes

I learned how to make a wireframe for my application using Figma, and it was a new experience for me. I did not enjoy it because working on the user interface is not my cup of tea. Another reason would be that my feature set changed from what I envisioned initially, so I had to make on-the-fly design changes. This experience made me realise that wireframes are more like a guide than a final design. In the future, I would like to try working on and doing wireframe design to make the process smoother.

### Tokens

I learned how to use JSON Web tokens to implement secure authentication for my application. Initially, I misunderstood how they should work and refresh themselves. My application provides a long-lived access token for the user. However, the proper implementation should issue a short-lived access token and a very long-lived refresh token so the application can ask to refresh the access token for a long time without requesting the user to log in again. This would improve the security of the application by having the access token expire quickly.

### 6.3. Achievements

Despite encountering challenges during my project's development, I achieved my goals. While some of the features could still be improved, I am happy with the project I managed to make.

I demonstrated the ability to plan and organise my project tasks effectively, adapting to the challenges and solving problems by finding creative solutions to reach my goals. I gained some experience in designing user interfaces and better understanding them, improving the overall design, and enhancing the user experience.

I have continuously learned and made gradual improvements throughout the project development process, improving my front and backend skills.

### 6.4. Future Development

In the future, or if the time would allow me, the following improvements could be made to my project:

- Refactor most of the backend and frontend to use the DRY principle (Do not Repeat Yourself) and ensure some of the logic appears only once. This would make it easier to update the application in the future and require fewer changes.
- Implement unit, integration, and widget tests for the Flutter application.
- Add dark/light mode for the application to allow a user to choose.
- The community screen in the future could allow people to start sharing the recipes of the food they share. This could attract even more people to come back and use the application.
- Expand on the notification system. It could be done using the background worker, so it could be done the way I envisioned. Some calculations would happen in the background to check if the notification should be sent.
- Test the application on the iPhone and make sure it works as intended. It should be straightforward, as Flutter is cross-platform and should work. If I would have to guess, the application should work as is or need some configuration for the libraries.
- Implement proper JWT token refreshing, with two tokens—a short-lived access token and a long-lived refresh token.
- Implement a scroll pagination, so when the app launches, it does not download all the listings from the user but a set amount of them. When the user reaches the bottom of the list, it downloads the next part of the data, and so on. This would remove the overhead of downloading all the available listings and only get the ones the user can see.

## 7. Generative AI Usage

Generative AI was not used to write any sections of the report. It was used for brainstorming purposes only and to give me ideas. It did serve as a valuable tool for ideas, and I can see that it can be a great tool when used properly.

Other tools used were Microsoft Editor and Grammarly. They checked grammar and spelling mistakes and sometimes offered to rephrase my sentences for correctness, clarity, engagement, or delivery. However, those tools did not generate sentences.

## Bibliography

Amazon, 2024. *What is SQL*. [Online]

Available at: <https://aws.amazon.com/what-is/sql/>

[Accessed 22 January 2024].

Atlassian, 2024. *Estimation*. [Online]

Available at: <https://www.atlassian.com/agile/project-management/estimation>

[Accessed 23 January 2024].

Chi contributors, n.d. *go-chi Docs*. [Online]

Available at: <https://go-chi.io/#/pages/middleware?id=jwt-authentication>

[Accessed 7 February 2024].

dash-overflow.net, 2024. *Provider - Flutter Package*. [Online]

Available at: <https://pub.dev/packages/provider>

[Accessed 10 February 2024].

Docker, 2024. *Multi-Stage builds*. [Online]

Available at: <https://docs.docker.com/build/building/multi-stage/>

[Accessed 03 March 2024].

Flutter, 2024. *Flutter - Build apps for any screen*. [Online]

Available at: <https://flutter.dev/>

[Accessed 17 January 2024].

Flutter, 2024. *Future Class*. [Online]

Available at: <https://api.flutter.dev/flutter/dart-async/Future-class.html>

[Accessed 19 February 2024].

Freeborn, C., 2022. *Pros and cons of Flutter app development*. [Online]

Available at: <https://blog.logrocket.com/pros-cons-flutter-app-development/#what-is-flutter>

[Accessed January 2024].

GitHub, 2024. *Learning about Projects*. [Online]

Available at: <https://docs.github.com/en/issues/planning-and-tracking-with-projects/learning-about-projects/about-projects>

[Accessed 23 January 2024].

Google, 2024. *The Go Programming Language*. [Online]

Available at: <https://go.dev/>

[Accessed 16 January 2024].

Sharan, S., 2023. *Implementing Flutter SearchDelegate: a Step-by-Step Guide*. [Online]

Available at: <https://medium.com/@sharansukesh2000/implementing-flutter-searchdelegate-a-step-by-step-guide-b8c4550627d1>

[Accessed 1 March 2024].

Vubon, R., 2023. *Let's Integrate the Firebase Realtime Database with Golang*. [Online]

Available at: <https://medium.com/@vubon.roy/lets-integrate-the-firebase-realtime-database-with->



golang-7c065a7b7313

[Accessed 25 January 2024].

Wikipedia contributors, 2., 2024. *Jenkins (software)*. [Online]  
Available at: [https://en.wikipedia.org/wiki/Jenkins\\_\(software\)](https://en.wikipedia.org/wiki/Jenkins_(software))

[Accessed 01 February 2024].

## Appendix A – GitHub Sprint Report

Figure 15 - Sprint 1.

1	🕒 Set up Git repo's #2	Ygnas	Done	Iteration 1	3
2	🕒 Write introduction section of report #3	Ygnas	Done	Iteration 1	8
3	🕒 Background #4	Ygnas	Done	Iteration 1	3
4	🕒 Problem Definition #5	Ygnas	Done	Iteration 1	3
5	🕒 Project Goal #7	Ygnas	Done	Iteration 1	3
6	🕒 Solution #6	Ygnas	Done	Iteration 1	3
7	🕒 Research using Gin or Chi framework #8	Ygnas	Done	Iteration 1	8
8	🕒 Write research and analysis section of report #9	Ygnas	Done	Iteration 1	5
9	🕒 Requirements analysis #10	Ygnas	Done	Iteration 1	3
10	🕒 Create git repo's #1	Ygnas	Done	Iteration 1	3
11	🕒 Market analysis #11	Ygnas	Done	Iteration 1	3
12	🕒 Feature analysis #12	Ygnas	Done	Iteration 1	3

Figure 16 - Sprint 2.

13	🕒 Write modelling section of report #13	Ygnas	Done	Iteration 2	5
14	🕒 Create wireframes and Diagrams #15	Ygnas	Done	Iteration 2	5
15	🕒 As a user I want to register #36	Ygnas	Done	Iteration 2	5
16	🕒 As a user I want to login #37	Ygnas	Done	Iteration 2	5
17	🕒 As a User I want to update/edit listing #30	Ygnas	Done	Iteration 2	3
18	🕒 Add github actions for the backend #19	Ygnas	Done	Iteration 2	1
19	🕒 Write planning section of report #16	Ygnas	Done	Iteration 2	5
20	🕒 Initialize new repo's #14	Ygnas	Done	Iteration 2	1
21	🕒 As a user I want to add a listing #18	Ygnas	Done	Iteration 2	5
22	🕒 As a user I want to delete listing #21	Ygnas	Done	Iteration 2	5
23	🕒 Write tests for backend #31	Ygnas	Done	Iteration 2	5
24	🕒 Add github action for tests #33	Ygnas	Done	Iteration 2	3
25	🕒 As a user I want to get a listing #23	Ygnas	Done	Iteration 2	3
26	🕒 Add protection to github branch, to prevent merges before passing actions #25	Ygnas	Done	Iteration 2	1
27	🕒 Add logger middleware to backend #26	Ygnas	Done	Iteration 2	1
28	🕒 As a User I want to be able to get all the listings #27	Ygnas	Done	Iteration 2	3
29	🕒 Write CI/CD section of the report #39	Ygnas	Done	Iteration 2	5
30	🕒 Build a container Image for backend #42	Ygnas	Done	Iteration 2	5
31	🕒 Feature/update listing and tests #35	Ygnas	Done	Iteration 2	3

Figure 17 - Sprint 3.

32	🕒 Implement register in flutter app #2	Ygnas	Done	Iteration 3	5
33	🕒 Implement login in flutter app #1	Ygnas	Done	Iteration 3	5
34	🕒 Refactor firebase controller so that credentials file can be pass in using ENV vars #44	Ygnas	Done	Iteration 3	3
35	🕒 Change so the listing controller, so it makes listings for a user #45	Ygnas	Done	Iteration 3	3
36	🕒 Rate limit backend endpoints #46	Ygnas	Done	Iteration 3	1
37	🕒 Flutter add new listing #12	Ygnas	Done	Iteration 3	5
38	🕒 Change GitHub actions to not double test, and make docker image only on release #49	Ygnas	Done	Iteration 3	1
39	🕒 Flutter application get user listings #4	Ygnas	Done	Iteration 3	5
40	🕒 Refactor flutter user auth and it's state #6	Ygnas	Done	Iteration 3	3
41	🕒 Flutter application see listing in details #8	Ygnas	Done	Iteration 3	5
42	🕒 Flutter application make listing model same as in backend #11	Ygnas	Done	Iteration 3	3
43	🕒 Flutter delete a listing #15	Ygnas	Done	Iteration 3	3
44	🕒 Flutter profile page #20	Ygnas	Done	Iteration 3	5
45	🕒 Flutter edit a listing #17	Ygnas	Done	Iteration 3	5
46	🕒 Sort returned listings from backend #51	Ygnas	Done	Iteration 3	1

Figure 18 - Sprint 4

47	🕒 As a user I want to search listing #38	Ygnas	Done	Iteration 4	3
48	🕒 Ability to delete a user #57	Ygnas	Done	Iteration 4	3
49	🕒 As a user I want to comment on listings #65	Ygnas	Done	Iteration 4	8
50	🕒 Flutter app write a comment #30	Ygnas	Done	Iteration 4	8
51	🕒 Flutter community page for shared listings #22	Ygnas	Done	Iteration 4	5
52	🕒 Backend get all listings from all users #53	Ygnas	Done	Iteration 4	3
53	🕒 Move Firebase url to env var #54	Ygnas	Done	Iteration 4	3
54	🕒 Update flutter community page to show listings from all users #24	Ygnas	Done	Iteration 4	3
55	🕒 Flutter show comment content #26	Ygnas	Done	Iteration 4	5
56	🕒 Backend like a listing #61	Ygnas	Done	Iteration 4	8
57	🕒 Flutter ability to like a listing #27	Ygnas	Done	Iteration 4	8
58	🕒 Flutter display week consumption visualization #33	Ygnas	Done	Iteration 4	8
59	🕒 Flutter take a picture of food #34	Ygnas	Done	Iteration 4	5
60	🕒 Backend upload and compress the image from app #68	Ygnas	Done	Iteration 4	5

Figure 19 - Sprint 5.

61	🕒 As a user I want to filter listings #56	Ygnas	Done	Iteration 5	3
62	🕒 Implementation section sprint 2 #73	Ygnas	Done	Iteration 5	5
63	🕒 Implementation section sprint 1 #72	Ygnas	Done	Iteration 5	5
64	🕒 Ability for user to delete it's account #39	Ygnas	Done	Iteration 5	5
65	🕒 Flutter app local notifications #41	Ygnas	Done	Iteration 5	8
66	🕒 Notification settings in the app #46	Ygnas	Done	Iteration 5	3
67	🕒 Flutter save a listing location if permission is given #42	Ygnas	Done	Iteration 5	5
68	🕒 Flutter display listing on the map #45	Ygnas	Done	Iteration 5	5
69	🕒 Flutter fix the image uploading on new listings #47	Ygnas	Done	Iteration 5	3
70	🕒 Backend do not create new listing ID if frontend gave one #79	Ygnas	Done	Iteration 5	1
71	🕒 Local notification logic #50	Ygnas	Done	Iteration 5	5
72	🕒 Flutter filtering by date with date picker #53	Ygnas	Done	Iteration 5	5
73	🕒 Refactor search button as separate widget #55	Ygnas	Done	Iteration 5	1
74	🕒 Profile progress circle for the week #57	Ygnas	Done	Iteration 5	3

Figure 20 - Sprint 6.

75	🕒 Write implementation section of report #71	Ygnas	Done	Iteration 6	15
76	🕒 Implementation section sprint 5 #76	Ygnas	Done	Iteration 6	5
77	🕒 Implementation section sprint 6 #77	Ygnas	Done	Iteration 6	5
78	🕒 Implementation section sprint 3 #74	Ygnas	Done	Iteration 6	5
79	🕒 Implementation section sprint 4 #75	Ygnas	Done	Iteration 6	5
80	🕒 Deploy backend to kubernetes #82	Ygnas	Done	Iteration 6	8
81	🕒 Listing, user class tests for flutter app #61	Ygnas	Done	Iteration 6	5
82	🕒 Write up reflection section of the doc #84	Ygnas	Done	Iteration 6	8
83	🕒 Github action to build the android apk #63	Ygnas	Done	Iteration 6	1
84	🕒 Community screen no more dismissable #64	Ygnas	Done	Iteration 6	1
85	🕒 Flutter app auto build and upload apk on new release #71	Ygnas	Done	Iteration 6	3

## Appendix B – Kubernetes Deployment

Kubernetes deployment is done after the new release in GitHub, and the docker images are built in the GitHub packages repository. At first, I thought of using ArgoCD to automate the application's lifecycle and make it easy to understand. That would have allowed me to use the GitOps pattern to have my version control repository in the desired application state. I only used Kubernetes hosting to make the deployment less complicated for a simple backend.

Deploying the backend only needs the 'foodlog-credentials.json'. An example can be found in GitHub, or the file can be downloaded from your Firebase project website. As my credentials file would allow anyone to control my project, I will not share that here. The second change must be made is setting the JSON web token secret and a URL to your Firebase database.

Figure 21 - Kubernetes config map.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: foodlog-config
data:
  JWT_SECRET: #Fill this for the JWT token secret generation
  DATABASE_URL: https://foodlog-9c3fd-default-rtdb.europe-west1.firebaseio.com/ #Change this to point to your firebase
```

After that, deployment can be done using one command in the terminal:

make

Figure 22 - Kubernetes make file.

```
.PHONY: apply
apply:
  kubectl create secret generic foodlog-credentials --from-file=foodlog-credentials.json | kubectl apply -k .
```

This will run the make file, creating the secrets inside Kubernetes out of the provided credentials file. Then, apply the kustomization file, which will make the config map, deployment, and create a service that will allow the application to be reachable from the outside.

Figure 23 - Kubernetes deployment.

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: foodlog-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: foodlog
  template:
    metadata:
      labels:
        app: foodlog
    spec:
      containers:
        - name: foodlog
          image: ghcr.io/ygnas/foodlog:latest
          envFrom:
            - configMapRef:
                name: foodlog-config
          ports:
            - containerPort: 3000
          volumeMounts:
            - name: credentials
              mountPath: /app/foodlog-credentials.json
              subPath: foodlog-credentials.json
      volumes:
        - name: credentials
          secret:
            secretName: foodlog-credentials

```

Figure 24 - Kubernetes service.

```

apiVersion: v1
kind: Service
metadata:
  name: foodlog-service
spec:
  selector:
    app: foodlog
  ports:
    - protocol: TCP
      port: 80
      targetPort: 3000
  type: LoadBalancer


```

## Appendix C – Flutter Wireframes


**Login.**

Login

Email address

 Enter Email Address

Password

 Enter Password

Login

## Home screen.

Listing details.

↑ 15

💬 0 Comments

3

Label

Label

Label

Label

Add new listing.

←

Add new

Type

Snack

Take photo

Location

Save listing

3

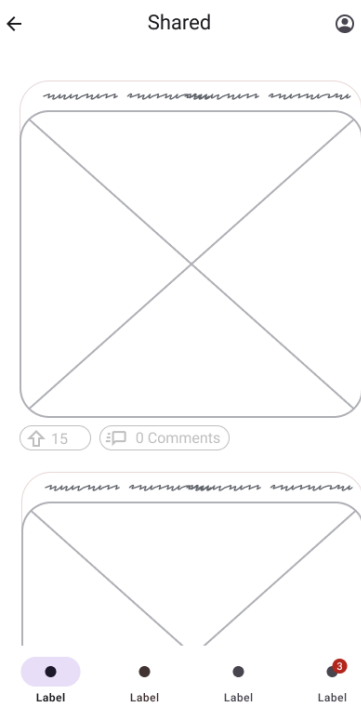
Label

Label

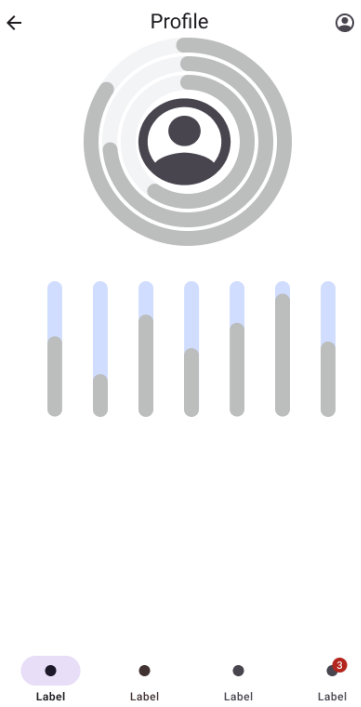
Label

Label

Community screen



Profile screen.





## Appendix D – Ethics Checklist

This Ethics Checklist must be completed for all final year undergraduate, taught postgraduate and research projects in the School of Science and Computing.

### View your response(s)

 Respondent: **Ignas Baranauskas** (Group: CM-HDIPCS) Submitted on: Saturday, 25 November 2023, 3:28 PM

### Ethics Checklist for Undergraduate, Taught Postgraduate and Research Projects in the School of Science and Computing

All students in the School of Science and Computing who are either (1) in the final year of an undergraduate/BSc degree, or (2) on a taught postgraduate/MSc programme **must complete this Ethics Checklist before conducting their project** regardless of the project type or discipline. The Checklist should also be completed by anyone (whether staff member or student) conducting a **research project** (whether programmatic or not) within the School.

The purpose of this Ethics Checklist is to **identify projects that will require formal ethical approval** from the School Research Ethics Committee, or the SETU Research Ethics Committee, before they can proceed.

Students/applicants should note that this Ethics Checklist is a **formal declaration**, and great care must be taken to **answer all questions accurately**. Students should consult with their project supervisors/advisors regarding any aspects or questions that they are unsure of before completing and submitting the Ethics Checklist.

Students/applicants must **answer all questions** presented to them until the Checklist questionnaire is completed.

### Feedback Report

No human experimentation issues (UG).

No animal experimentation issues (N/A).

No issues regarding the use of human tissues.

No animal tissue or biological fluids issues.

No ionising radiation issues.

No primary data collection issues (N/A).

No underage/vulnerable people issues (UG).

?

No issues regarding existing/secondary data use (N/A).

No controversial data issues.

No issues related to the collection of rare or protected plants.

No issues regarding the use of genetically modified (GM) plant material.

**Instructions:**

1. If the above feedback is **entirely green** then, based on your answers, there is **no need to apply for ethical approval** for your project.
2. If **any** part of the above feedback is **yellow/amber**, then there is at least one issue with your project that needs to be reviewed and **you must apply for ethical approval** to continue your project.
3. If **any** part of the above feedback is **red** then there is a serious ethical issue and **you cannot continue your project** as currently planned.

*It is recommended that you print this Feedback Report to a PDF file for your records. You should also forward and discuss this Feedback Report PDF with your project supervisor. They will be able to advise if you have any further questions or if you need to apply for ethical approval.*

- 1 \* Are you a student on a **final year undergraduate** programme, a **taught postgraduate** programme, or are you conducting a **research project**?
  - ☒ Final Year Undergraduate
  - ☐ Taught Postgraduate
  - ☐ Postgraduate Research Project
  - ☐ Other Research Project
- 2 \* What is the **working title** of your project?

FoodLog
- 3 \* Who are the project **supervisors/advisors/principal investigators**?

Colm Dunphy
- 4 \* Does your project involve **human experimentation**?
  - ☐ Yes ☒ No
- 5 \* Does your project involve **live animal experimentation**?
  - ☐ Yes ☒ No
- (6) \* Is the planned animal experimentation limited to **non-invasive procedures only** (such as feeding, weighing, or taking naturally voided faecal or hair samples), and does **not** involve any invasive procedures (such as taking rectal faecal samples or blood) from live animals?
  - ☐ Yes ☐ No
- 7 \* Does your project involve the use of **human remains/cadavers/tissues/cells/biological fluids/embryos/foetuses**?
  - ☐ Yes ☒ No
- (8) \* Do you intend to only use established **commercial human cell lines**, and no other **human** remains/cadavers/tissues/cells/biological

fluids/embryos/foetuses in your project?

☐ Yes ☐ No

**9 \*** Does your project involve the use of **animal cells, tissues or biological fluids**?

☐ Yes ☒ No

**(10) \*** Do you intend to only use (1) **established commercial animal cell lines**, or (2) **slaughterhouse-derived tissues/fluids**, or (3) **fluids collected as part of routine animal husbandry** (e.g. milk) and no other animal tissues or biological fluids in your project?

☐ Yes ☐ No

**11 \*** Does your project involve the **collection of rare or protected plants**?

☐ Yes ☒ No

**12 \*** Does your project involve the generation or use of **genetically modified (GM) plant material**?

☐ Yes ☒ No

**(13) \*** Do you agree to (1) only use **established genetically modified (GM) plant cell lines, seeds, or plant products** in your project, (2) **not generate new plant mutations** using chemical or other means, and (3) follow specified SETU **containment and use protocols** for GM plant materials at all times?

☐ Yes ☐ No

**14 \*** Does your project involve the use of **ionising radiation**? (e.g. use of gamma ray spectrometry)

☐ Yes ☒ No

**(15) \*** Do you agree to carefully **follow the instructions** of the SETU designated **Radiation Protection Officer (RPO)**, and **adhere to all legal requirements** as set out in the Radiological Protection Act 1991 (Ionising Radiation) Regulations ([2019](#)), regarding the use of ionising radiation materials and equipment?

☐ Yes ☐ No

**16 \*** Does your project involve the **collection of any new (or primary) data** from **individual people or groups**?

☐ Yes ☒ No

**(17) \*** Does your project involve the **collection of any new (or primary) individual or group data** that is **personally or uniquely identifying**? (e.g. data about people or organisations/companies/groups that could be used to identify those individuals or groups; data collection might take any form, including internet and social media data, etc.)

☐ Yes ☐ No

**(18) \*** Will you ensure that participants who you are collecting data from are provided with **fair warning** and must provide **explicit informed consent** for any data collected?

☐ Yes ☐ No

**(19) \*** Will you ensure that any project-related data collection, data storage, and data use is in **full compliance** with the **EU General Data Protection Regulation (GDPR)** and the **Data Protection Act (2018)**?

☐ Yes ☐ No

**(20) \*** Does any of the data that you intend to collect include **sensitive or private personal information** about individuals, or **commercially sensitive information** about organisations/companies/groups?

☐ Yes ☐ No

**21 \*** Does your project involve **persons under the age of 18 years** (i.e. minors), or **any vulnerable groups**? (e.g. prisoners, refugees, those in care, addiction service users, etc.)

☐ Yes ☒ No

**22 \*** Does your project involve the use of **existing (or secondary) human data?** (i.e. data originally collected for another purpose)

☐ Yes ☒ No

**(23) \*** Is the existing or secondary human data you intend to use either (1) **anonymous/non-personally identifying** and in the **public domain**, or (2) available with **explicit and specific informed consent or permission** for the data to be **legally** reused in the way you intend?

☐ Yes ☐ No

**24 \*** Are any aspects of the primary/secondary data you intend to use for the project **controversial** in nature?

☐ Yes ☒ No

**25 \*** Before you submit the Ethics Checklist, you must **confirm all of the following**:

- ☒ I understand that the Ethics Checklist is a formal declaration.
- ☒ I have answered all questions on the Ethics Checklist carefully and truthfully.
- ☒ The supervisor/advisor (or principal investigator) for the project is present as the Ethics Checklist is being submitted, or they have given me explicit permission to submit it in their absence.
- ☒ I have had adequate ethics training and/or instruction prior to completing the Ethics Checklist.
- ☒ I understand, and agree to abide by, the general ethical principle of "do no harm" for this project.
- ☒ I will follow the instructions given in the Feedback Report.

**26 \*** Authentication Code (ask your project supervisor/advisor for this code)

Enter Student Number:

Enter the Authentication Code below and click "Verify Code"

**Note: If an INVALID authentication code is used then this submission is NULL and VOID**

4773