Specification Document

Project Name: Food Sharing Application

Team Number 16

Document Information

Project Name:	Food Sharing Application		
Prepared By (list all authors):	Ishan Kalra Ain Fatihah Dan Dingley Maria Papadopoulou Drew Wandless Azam Butt	**Preparation Date:**	10/05/2024
Document Version No:	29	**Document Version Date: 10/05/2024**	

Version History

Ver. No.	Ver. Date	Revised By	Description
1	26/03/2024	Drew Wandless	Draft of Section 1 (Background and Analysis)
2	29/03/2024	Drew Wandless	Draft of Section 2 (Deliverables)
3	30/03/2024	Ain Fatihah	Added to Section 4 (Hardware Specifications and Software)
4	02/04/2024	Dan Dingley	Added to Section 4 (Hardware Specifications and Software)
5	07/04/2024	Drew Wandless	Draft of section 7 (functional and non-functional requirements)
6	12/04/2024	Azam Butt	Added to Section 4 (Hardware Specifications and Software)
7	22/04/2024	Ain Fatihah	Added to Section 7 (Functional Requirements)
8	25/04/2024	Ain Fatihah	Added to Section 6 (Definition of terms)
9	25/04/2024	Azam Butt	Added to Section 8 (Other Considerations)
10	26/04/2024	Drew Wandless	Begun work on testing methodologies section
11	26/04/2024	Ain Fatihah	Begun work on software design section
12	28/04/2024	Drew Wandless	Completed final section (testing methodologies)
13	29/04/2024	Drew Wandless	Added to section 8, constraints dependencies and assumptions
14	1/05/2024	Ain Fatihah	Added to Section 9 (Component diagram) & Added to References
15	01/05/2024	Drew Wandless	Added sequence diagrams to section 9
16	02/05/2024	Drew Wandless	Completed section 1 (added references)
17	02/05/2024	Maria Papadopoulou	Added activity diagram to section 9
18	02/05/2024	Azam Butt	Added to Section 9 (UML use case diagram).
19	03/05/2024	Ishan Kalra	Added to Section 9 (UML use class diagram).
20	03/05/2025	Drew Wandless	Added Table of contents and figures
21	04/05/2024	Ain Fatihah	Added to Section 9 (Block diagram)
22	04/05/2024	Ishan Kalra	Updated section 4 and 7.1
23	05/05/2024	Azam Butt	Added to Section 8 (Other Considerations)
24	06/05/2024	Ain Fatihah	Added to Section 9 (GUI)
25	08/05/2024	Dan Dingley	Revised Section 2 (Deliverables)
26	08/05/2024	Dan Dingley	Revised Section 10 (Testing Methodologies)
27	08/05/2024	Ain Fatihah	Added requirements description to Section 9 (Use case & Sequence diagram, GUI)

Ver. No.	Ver. Date	Revised By	Description
28	09/05/2024	Ain Fatihah	Added requirements description to Section 9 (Class & Activity diagram)
29	10/05/2024	Ishan Kalra	Added specification no.14 to section 8.3

Table of Contents

Introduction 2
- **Document Information** 2
- **Version History** 2
- **Table of Contents** 3
- **Table of Figures** 3

Background and Analysis 4
- **Too Good To Go Review** 4
- **Olio Review** 5
- **Karma Review** 6

Deliverables 7
Project Plan 8
Hardware and Software 9
Definitions and References 10
Solution Requirements 11
- **Functional Requirements** 11
- **Non-functional Requirements** 12

Other Considerations 14
- **Assumptions** 14
- **Constraints** 15
- **Dependencies** 16

Software Design 17
- **Architecture Design** 17
- **High-level overview** 18
- **Class Diagram** 19
- **Sequence Diagram** 20
- **Activity Diagram** 21
- **GUI Design** 23

Testing Methodology 25
Contribution Matrix 26

Table of Figures

Figure 1 4
Figure 2 5
Figure 3 6
Figure 4 8
Figure 5 17
Figure 6 17
Figure 7 18
Figure 8 19
Figure 9 20
Figure 10 20
Figure 11 21
Figure 12 22
Figure 13 23

1. Background & Analysis

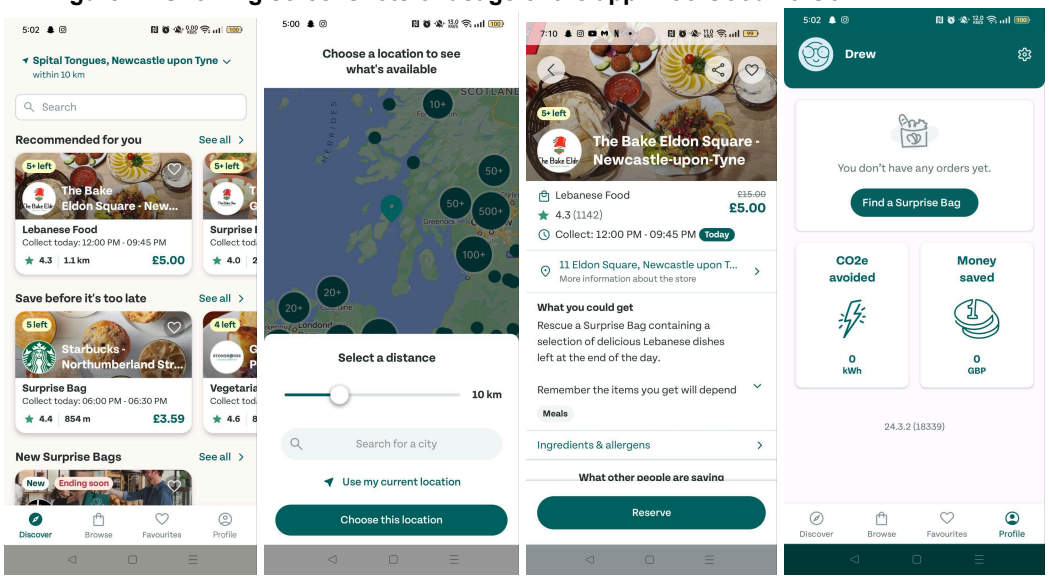
1.1. Analysis

The background to the task. This describes the business environment that the project would typically fit into. Look at 3 similar systems that are available and give feedback on them.

To understand many aspects of our own system that we intend to create, it is often useful to reflect upon similar systems already on the market and to provide unbiased and fair feedback between the systems reviewed. The systems we will be reviewing include: Too Good To Go, Olio, and Karma.

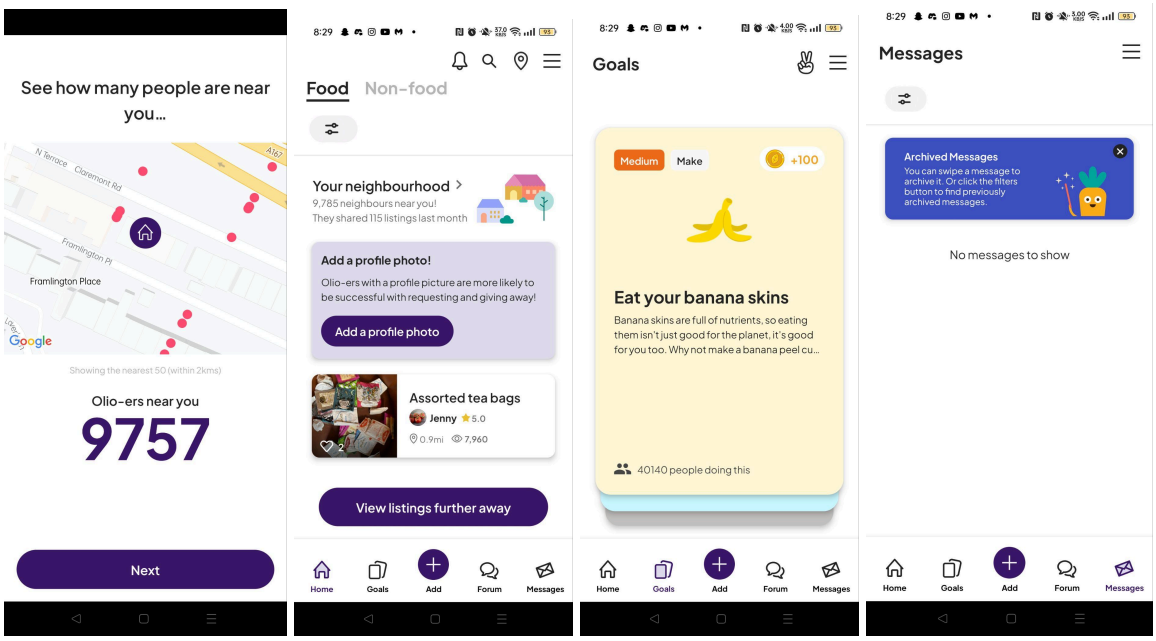
Beginning with a platform with the largest and most dispersed user base of the three, with 75 million registered users (Too Good To Go, 2023), is a system aimed at reducing food waste that companies produce by significantly discounting the unused food at the end of a working day. Accessible via downloading the app, the user is met with a login/signup process, where they can conveniently log in with Google or Facebook. After doing so, users are prompted to enter their location and then taken to a discovery page. From this page, users can access all parts of the app, including the other three pages on the bottom of the navigation bar (browse, favourites, and profile). This discovery page is also where users can access all advertisements for discounted food, which include information such as location, time, price, quantity remaining, review rating, as well as any important allergy information. Additionally, users have the option to add items to favourites, share them, or reserve a slot to pick up the food (as seen in the images below). Another significant (and unique) aspect of the program, highly commended by other users and functionally advertised on their app store (Too Good To Go, n.d.), is the ability to track statistics such as money saved and their effect on the environment, which aids in user retention and engagement.

Figure 1 - Showing screenshots of usage of the app "Too Good To Go"



Moving on to a system called Olio, which is designed with communities and individuals in mind compared to the businesses that Too Good To Go supports. The setup process is very similar to the previous systems (including the setup of location), although notably it does not include the ability to log in via Google or Facebook, which is more limiting for users. When taken to the home page after this, it is evident that there are a lot more interactive options than in the other systems. Notably, you can search, view notifications, change location, filter search, and access your profile all from the home screen, not to mention the navigation bar at the bottom. This navigation bar includes goals, add, forum, and messaging pages, specifically the latter two, which are important for such an application. Using the add button, you can upload your own food/items to give away, which is a notable difference from Too Good To Go and an option it does not offer. As for the advertisement of items themselves, once clicked on, you can view the times, location, description, likes, as well as have the ability to request the item or report the listing itself, which is a useful method of detecting malicious actors. Notably, Olio tries to distinguish itself from similar software thanks to its generalisation of recycling and is actually advertised for giving away all types of items and not just specifically food (Olio, n.d.). Overall, Olio is definitely more user-based in the fact that you can upload and create your own listings and acts as an application that you can use both from the point of a receiver and distributor. Additionally, it is more interactive for a user as they can utilise the app to look at anti-waste forums but also be able to directly message other users and members of their community, as seen.

Figure 2 - Screenshots showing using of the app "Olio"

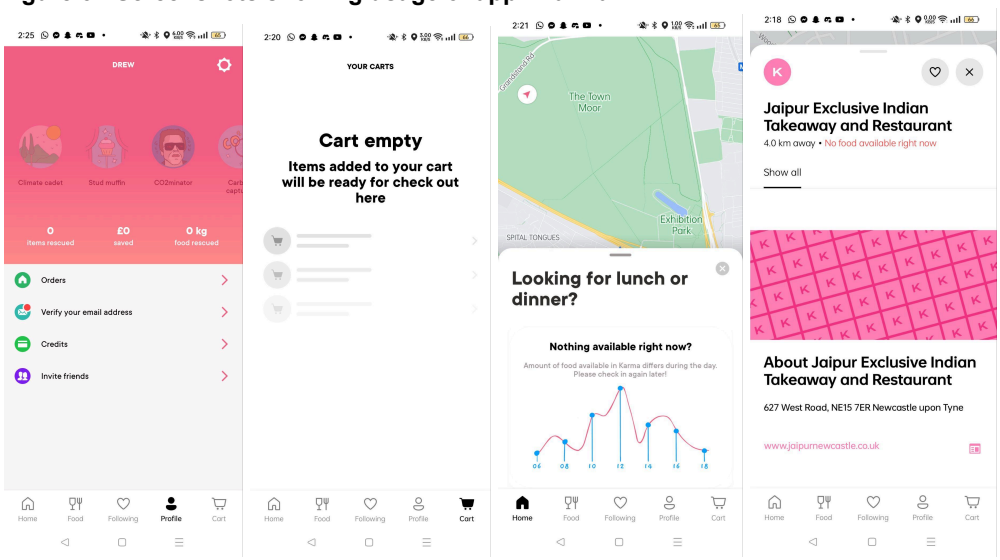


The final and least popular system that we will look at is yet another mobile application known as Karma. Just like the previous two systems, an account is required, and you are provided with the option to log in or sign up (once again without any implementation of Google or Facebook for sign-in). After this, you are prompted to let the app use your location and are then taken to a home screen with an implemented map of the local area with possible listings posted on the map. There is a navigation bar at the bottom with the possible options of food, following, profile, and cart. All of these options pop up over the map once clicked on, compared to the other systems which completely redirect you.

As for the food listings themselves, they are only available to view if they are currently accessible (which differs from Too Good To Go, which permanently advertises even if they are not accessible at the current time). Just like the other applications, it has a bookmark system of “following” specific places/sellers, as well as a shopping cart which enables you to order the food directly from the app itself, compared to reserving the food like the prior systems.

Regarding the profile option, it lets you view orders, change settings, invite friends, and add credits using codes, which is also another unique quirk of the application compared to the other two. Overall, this system is very similar to the other applications but with extra quirks and features that provide it with enough individualism from other competitors, although it clearly has a smaller user base with 500k downloads on the Google Play Store (Karma, n.d.). I think this is partially due to the business model and not the software itself (as distributors are charged a 25% sale fee), which is an obvious disadvantage for most users and hence why the previous two applications are favoured.

Figure 3 - Screenshots showing usage of app “Karma”



2. Deliverables

What you will deliver to the customer.

As the desired outcome of the project is to create a system through which users can donate or receive free food (thereby adhering to the 2nd UN Goal of zero hunger), it is important to understand and reflect upon the deliverables that we must meet for both users and customers. This is illustrated via the following high-level overview.

To ascertain the success of our project and the system itself, there are deliverables that must be set and met. The first of these is crucial to the development and design phase of the project. This is the technical design document itself, along with its sections, specifically a project plan, background analysis, hardware/software platform specifications, requirements, assumptions, constraints, dependencies, as well as an overview of intended testing methodologies and a software design . Additionally, we require a method of managing our team for the project to succeed; therefore, it is vital that we also have a team plan which includes scheduling resources such as a Gantt chart, contribution matrix, and meeting schedule. The deadline for the technical design document is the 10th May.

Using the design document as a guide we can begin development of the source code. All the while we will be tracking our hours via timesheets that give brief descriptions of what was done on which days. We will also be documenting our individual experiences and reflections within our own personal reports. Both will need to be submitted by the 24th May.

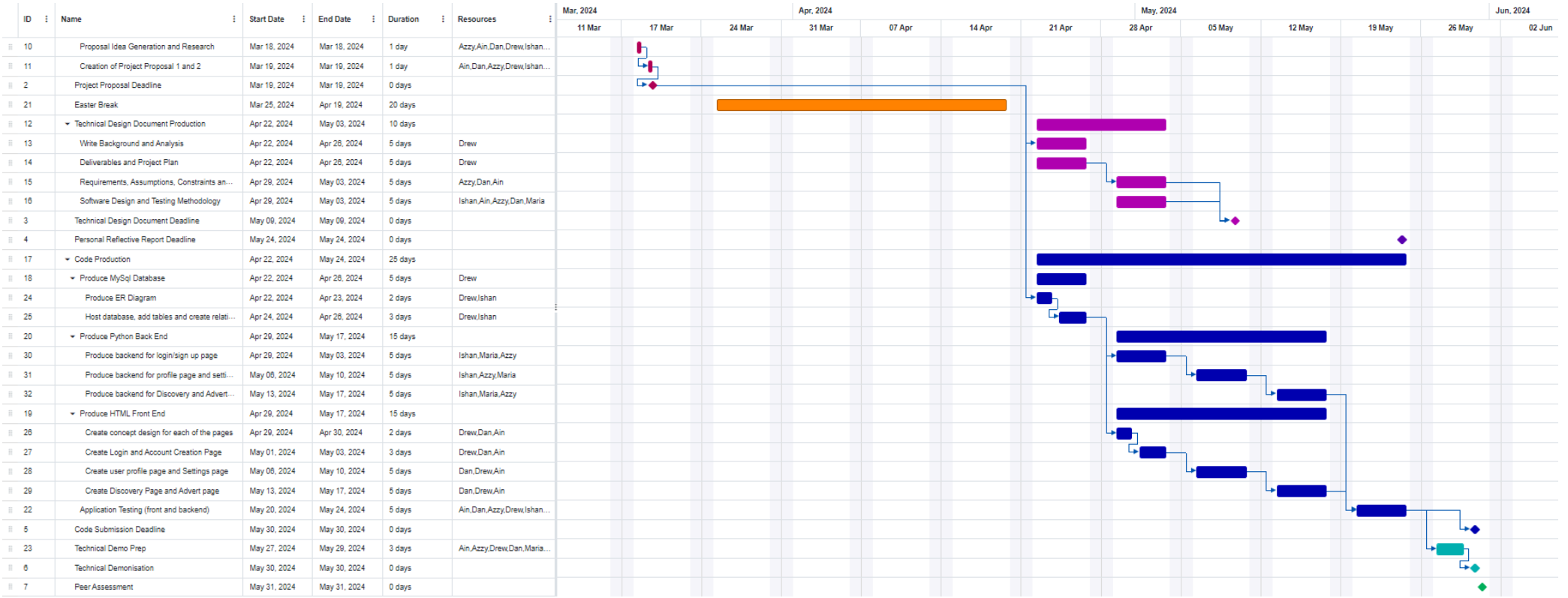
The next is due May 31st for which we need to provide the source code itself, which consists of several components. Firstly, there is the backend code, responsible for making systems work, such as connecting to a database and integrating with APIs. Following that, there is the front end code, responsible for providing our clients with an interactable interface that runs on their mobile devices. Finally, there is the mysql database connected to the rest of the system, responsible for storing and managing the data input into the application. Additionally, all comments, documentation and testing associated with their respective parts of the code-base are included alongside the submitted code. If successful this code will follow cohesive standards throughout, featuring well modularised class designs, accessible, clever GUI and be capable of robustly performing the functionality required of this app.

Ideally the code will be finished before May 31st in order to allow us time to prepare for the final deliverable, the technical demonstration at the trade Fair on that date. At which we will demonstrate the technology and functionality of our app through a confidently delivered presentation that involves everyone in the team and follows a clear, logical structure.

3. Project Plan

This will be based on the hard deadlines set for the project (module webpage/assignments) – however you should also provide any agreed soft deadlines determined by your team. All deadlines should be clearly shown and who is responsible for the activities, on a Gantt chart – can draw this in MSProject or Excel. Please ensure the Gantt chart is legible and fits on one page, preferably landscape view.

Figure 4 - Showing our proposed Gantt Chart along with dependencies, resources and milestones



ID	Name	Start Date	End Date	Duration	Resources
10	Proposal Idea Generation and Research	Mar 18, 2024	Mar 18, 2024	1 day	Azzy,Ain,Dan,Drew,Ishan...
11	Creation of Project Proposal 1 and 2	Mar 19, 2024	Mar 19, 2024	1 day	Ain,Dan,Azzy,Drew,Ishan...
2	Project Proposal Deadline	Mar 19, 2024	Mar 19, 2024	0 days	
21	Easter Break	Mar 25, 2024	Apr 19, 2024	20 days	
12	Technical Design Document Production	Apr 22, 2024	May 03, 2024	10 days	
13	Write Background and Analysis	Apr 22, 2024	Apr 26, 2024	5 days	Drew
14	Deliverables and Project Plan	Apr 22, 2024	Apr 26, 2024	5 days	Drew
15	Requirements, Assumptions, Constraints an...	Apr 29, 2024	May 03, 2024	5 days	Azzy,Dan,Ain
16	Software Design and Testing Methodology	Apr 29, 2024	May 03, 2024	5 days	Ishan,Ain,Azzy,Dan,Maria
3	Technical Design Document Deadline	May 09, 2024	May 09, 2024	0 days	
4	Personal Reflective Report Deadline	May 24, 2024	May 24, 2024	0 days	
17	Code Production	Apr 22, 2024	May 24, 2024	25 days	
18	Produce MySql Database	Apr 22, 2024	Apr 26, 2024	5 days	Drew
24	Produce ER Diagram	Apr 22, 2024	Apr 23, 2024	2 days	Drew,Ishan
25	Host database, add tables and create relati...	Apr 24, 2024	Apr 26, 2024	3 days	Drew,Ishan
20	Produce Python Back End	Apr 29, 2024	May 17, 2024	15 days	
30	Produce backend for login/sign up page	Apr 29, 2024	May 03, 2024	5 days	Ishan,Maria,Azzy
31	Produce backend for profile page and setti...	May 06, 2024	May 10, 2024	5 days	Ishan,Azzy,Maria
32	Produce backend for Discovery and Advert...	May 13, 2024	May 17, 2024	5 days	Ishan,Maria,Azzy
19	Produce HTML Front End	Apr 29, 2024	May 17, 2024	15 days	
26	Create concept design for each of the pages	Apr 29, 2024	Apr 30, 2024	2 days	Drew,Dan,Ain
27	Create Login and Account Creation Page	May 01, 2024	May 03, 2024	3 days	Drew,Dan,Ain
28	Create user profile page and Settings page	May 06, 2024	May 10, 2024	5 days	Dan,Drew,Ain
29	Create Discovery Page and Advert page	May 13, 2024	May 17, 2024	5 days	Dan,Drew,Ain
22	Application Testing (front and backend)	May 20, 2024	May 24, 2024	5 days	Ain,Dan,Azzy,Drew,Ishan...
5	Code Submission Deadline	May 30, 2024	May 30, 2024	0 days	
23	Technical Demo Prep	May 27, 2024	May 29, 2024	3 days	Ain,Azzy,Drew,Dan,Maria...
6	Technical Demonisation	May 30, 2024	May 30, 2024	0 days	
7	Peer Assessment	May 31, 2024	May 31, 2024	0 days	

4. Hardware and software platforms to be used for developing and running your solution

4.1. Hardware Specifications – what devices have you used to test and develop your system. What is the back-end of your system running on?

Device 1:
Asus Vivobook 15.6” Laptop | Operating System: Windows 11 Home | Memory: 16GB | Chipset: Intel® Core™ i5-1135G7 Processor | Display Resolution: Full HD 1920 x 1080p IPS |

Device 2:
Swift SF514-54GT | Operating System: Windows 10 Home | Memory: 8GB | Chipset: Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz 1.19 GHz | Display Resolution: Full HD 1920 x 1080p IPS |

Device 3:
MS-7D43 | Operating System: Windows 11 Home | Memory:16GB | Chipset: Intel® Core™ i3-12100 |

Device 4:
HP Elite Dragonfly Notebook | Operating System: Windows 11 Home | Memory: 16GB | Chipset: Intel® Core™ i5-8265U Display Resolution: Full HD 1920 x 1080p |

Device 5:
HP EliteBook 1040 14 inch G9 Notebook PC | Operating System: Windows 11 Pro | Memory: 16GB | Chipset: 12th Gen Intel(R) Core(TM) i7-1255U 1.70 GHz | Display Resolution: 1920 x 1200 |

Device 6:
HP Envy x360 13 Inch Notebook | Operating System: Windows 11 home | Memory: 8GB | Chipset: AMD Ryzen 5 4500U Display Resolution: Full HD 1920 x 1080p |

4.2. Software – what software have you used to develop your system (with references given to the software)

Software 1: Figma - A very popular UI design tool to help us mock up the front-end.
Figma (Version 116.17.13) [Computer software]. Available at: https://www.figma.com/ (Accessed: 21 March 2024).

Software 2: PyCharm Pro - A Python IDE which all members of the group will be using for the creation of backend and front-end development.
Pycharm (Version 2024.1) [Computer Software]. Available at: https://www.jetbrains.com/pycharm/

Software 3: DBeaver - A free to use and useful DBMS which will be used to create the database.
DBeaver (Version 24.0.3) [Computer software]. Available at: https://dbeaver.io/

Software 4: OnlineGantt - A free to use gantt chart maker.
Online Gantt (2024). *Free Online Gantt Chart Software*. [online] www.onlinegantt.com. Available at: https://www.onlinegantt.com/#/gantt.

Software 5: Kivy - A Python framework specifically designed for creating mobile applications.
Kivy (Version 2.3.0) [Computer Software] Available at: https://kivy.org/doc/stable/gettingstarted/installation.html

Software 6: Git - A versioning tool which helps us manage the system and its versions.
Git (Version 2.45.0) [Computer Software] Available at: https://git-scm.com/download/win

Software 7: Python - A high-level, general purpose programming language.
Python (Version 3.10.11) [Computer Software] Available at: Python Release Python 3.10.11 | Python.org

5. References – to all documents you use e.g. for research and also the initial Project Brief etc.

Too Good To Go (2023). *Too Good To Go Announces New Global Milestone: 200 million meals saved - Too Good To Go*. [online] www.toogoodtogo.com. Available at: https://www.toogoodtogo.com/en-gb/press/200-million.

Too Good To Go (n.d.). *Too Good To Go: End Food Waste - Apps on Google Play*. [online] play.google.com. Available at: https://play.google.com/store/apps/details?id=com.app.tgtg&hl=en_US.

GeeksforGeeks (2017). *Types of Software Testing - GeeksforGeeks*. [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/types-software-testing/.

United Nations (n.d.). *The 17 Goals*. [online] The Global Goals. Available at: https://www.globalgoals.org/goals/.

Guru99.com. (2024). *Static Testing vs Dynamic Testing: What's the Difference?* [online] Available at: https://www.guru99.com/static-dynamic-testing.html.

Olio (n.d.). *Olio - Share more, waste less - Apps on Google Play*. [online] play.google.com. Available at: https://play.google.com/store/apps/details?id=com.olioex.android&hl=en&gl=US.

Karma (n.d.). *Karma - Save food with a tap - Apps on Google Play*. [online] play.google.com. Available at: https://play.google.com/store/apps/details?id=com.karma.life&hl=en&gl=US.

6. Definition of terms – all terms used within the document e.g. Carbon Footprint – what is it? Brief description. This section can also be put at the end of the document if you think it best.

IDE - An Integrated Development Environment (IDE) is a software application that combines commonly used developer tools into a simple interface for writing and testing other applications.

DBMS - Database Management System (DBMS) is a software system that acts as an interface between users and databases, allowing them to create, retrieve, conduct queries on data, and control database access.

HD - HD stands for High Definition, which refers to higher-quality video than standard resolution. The amount of pixels on the display determines its resolution. HD videos typically have a minimum resolution of 1280 x 720 pixels.

UN Goals - The United Nations has established 17 objectives to reduce poverty by addressing social concerns such as education, gender equality, and health, while also taking into consideration environmental preservation and climate change.

API - Application Programming Interface, this is an interface which acts as a point of contact between two services or different pieces of technology.

HTML - HyperText markup language, this is a markup language which is commonly used for building core functionality of websites alongside common languages such as javascript and CSS

User - A member of the general public who has registered for the application.

FR(Number) - Referencing a specific functional requirement within our Functional requirement table

Admin - A type of user with a range of extended privileges compared to a conventional user, including the ability to manage other users.

7. Solution requirements

For each requirement indicate its Priority e.g. High, Medium, Low. As you (your team) are the suppliers of the system you need to provide your customer with information on what requirements you will complete within the development of the system. In the supplier compliance section your responses must be Yes, No or Partial coverage of the requirement by the solution you are proposing. If partial, you need to specify why you will only deliver part of the requirement in the Supplier Comment column e.g., if a requirement is high priority and you will not provide this you must give a short-written rationale.

7.1. Functional Requirements

ID	Description	Priority (H, M, L)	Supplier Compliance (Full, partial or will not be delivered)	Supplier Comment
FR1	Unregistered users can create an account using an email , password, postcode and phone number.	H	Full	The details will be validated.
FR2	Users can sign into an account using their email and password.	H	Full	
FR3	Users can manage their profile once logged in (change account details, etc.) which is then updated in the database.	H	Full	
FR4	Users must confirm that they are 18+ when creating an account.	H	Full	The age restriction applies to donation and request for certain items.
FR5	User data should be stored in a database and be retrievable via queries.	H	Full	
FR6	Users will be able to query and store data (in the database) through a front-end interface.	H	Full	Logic and algorithm to be completed.
FR7	Users will be able to post advertisements. This will include adding a name, description, location, phone number, time, image and status to the advertisement.	H	Full	
FR8	Users must be verified to be able to post advertisements.	L	Partial	Users could upload a form of ID, for example, photo ID, a letter with their details, etc.
FR9	There will be a response system to contact users who post advertisements as well as the ability to reserve food.	L	Full	
FR10	There should be a way of sharing user advertisements as well as receiving notification of nearby adverts.	L	Full	
FR11	Users should be able to see active advertisements on the ‘Homepage’.	L	Full	Homepage logic not presented with backend Class UML.
FR12	Users should be able to bookmark other advertisements to their ‘Favourites’ page.	L	Full	Favourites page logic not presented

				with backend Class UML.
FR13	Users should be able to see the hidden details on the advertisements after following the advertiser account.	M	Will not be delivered.	
FR14	Admins will have access to an admin panel (UI).	H	Full	
FR15	Admins will be able to review database logs.	H	Full	
FR16	Admins will be able to modify/remove users.	H	Partial	Not all users (for example, other admins)
FR17	Admins will be able to modify/remove adverts.	M	Full	
FR18	Admins will be able to modify/remove advert responses.	M	Full	
FR19	Admins will be able to review flagged posts.	H	Full	
FR20	Users should be able to search for specific advertisements or search for them in a specific area.	H	Full	Search bar logic not presented in the backend Class UML.
FR21	A notification system is required to inform users whenever a new advertisement is posted.	M	Partial	This may be in-app notifications
FR22	Users should be able to select their notification preferences.	M	Partial	This is managed outside the app
FR23	There should be a way of reporting advertisements.	L	Full	
FR24	There should be a review system for users.	L	Partial	This will be indirectly through flags/ flagged accounts
FR25	There should be a review system for advertisements.	L	Partial	This will be for users instead
FR26	Flagged posts are added to a ‘Review’ section to be reviewed by an admin.	M	Full	
FR27	It should be available as an iOS and Android app.	H	Full	
FR28	There should be an embedded Google Maps API which allows users to view all advertisements in the local area on their ‘Discovery’ page.	H	Full	
FR29	App should remember the user's log in details until they log out.	H	Full	

7.2 Non-Functional Requirements

ID	Description	Priority (H, M, L)	Supplier Compliance (Full, partial or will not be delivered)	Supplier Comment
NFR1	The application should load and respond to user interactions within a max delay of 1 second.	H	Partial	This is not guaranteed.
NFR2	The web application should perform as normal with common web browsers: Google Chrome, Microsoft Edge, Firefox, and Safari.	H	Full	
NFR3	The system should be able to operate as normal with at least 1000 users.	M	Partial	If storage space limit permits
NFR4	The user data should be secure. This means encryption should be used for the database and even SQL injection prevention.	H	Full	
NFR5	The database should be backed up every 5 days so that it can be recovered.	L	Partial	This will be done from the DB provider's end.
NFR6	The system should be well-documented to be easily maintained by developers (markdown files)	L	Full	
NFR7	All application functionality outside of mobile unique features/optimization should share the same code.	H	Full	
NFR8	The application should be installable on android as an apk that could be published to the playstore.	H	Full	
NFR9	The program should include documentation/tutorials on how to use it for new users to reinforce its usability.	L	Partial	This may be external to the program.
NFR10	The advert feed should update with new adverts posted within 10 seconds of posting.	L	Full	
NFR11	The app should safeguard against malicious users via reporting, post flagging,	H	Full	
NFR12	The system will have multiple language options as well as a visual impairment/colourblind mode, and be made inclusive with larger icon and text options.	M	Full	

NB: Compliance –Comments can relate to priority level and the reason for it e.g. we will only deliver part of this functionality at this time as it is low priority. Generally, the team should aim to deliver all requirements that are noted as High (H) and Medium priority (M).

8. Other considerations

8.1 Assumptions

ID	Description	Comments
1	Users must have a device with compatible hardware and software to run the application.	
2	Users must have an Internet connection to use the application.	
3	Users must be appropriately proficient with their device to interact with the application effectively.	A user-friendly and accessible design will mitigate the amount of documentation required.
4	Users intend on using input methods with non-malicious intent.	We will still be sanitising and validating input methods.
5	Users will primarily be non-profits/individuals looking to donate food or seek donations (the system will be used by its intended target audience).	We assume that all users make use of the application for its intended purpose.
6	Posting and transacting food donations via the app complies with local food safety and handling regulations.	Legislation of such is out of our hands.
7	The database is assumed to be scalable as needed, despite current limitations on storage capacity.	Currently the database storage capacity is limited by the host.
8	Admins are actively reviewing flagged posts and users.	This will mitigate troll posts.
9	Basic security measures are in place such as a secure WiFi network, a firewall, and an antivirus.	This will help prevent malicious attacks.
10	The default language of the user’s device is assumed to be English (UK).	This can be modified in settings.
11	Users will input UK postcodes.	
12	Documentation, tutorials, and support will be widely accessible.	This reduces the need to incorporate them into the app itself.
13	Third-party software, APIs, frameworks, and the database will remain maintained and supported.	
14	The software will meet its performance benchmark and will respond quickly under varying loads.	This also adheres to performance under pressure and scalability

8.2 Constraints

ID	Description	Comments
1	The app must be compatible with iOS and Android platforms, requiring resources to develop both versions.	Limits the amount of possible users based on their device.
2	Strong requirements for protecting user data which could limit some social media features due to privacy concerns.	
3	Development is limited by our small team size (6 developers), affecting the breadth and speed of project tasks.	If we had more developers we could get more work done in a faster time
4	The scalability of our platform is constrained by current database storage limits.	The database currently has a set storage space
5	The scope of the project must adhere to at least one of the UN goals.	We have to fit the goal of our project into one of the design goals
6	We are constrained by the available technologies that we have as well as specifically what prior technologies we have used and are confident with using.	Some developers will be less experienced in specific software and programming languages that we will use.
7	We are constrained by the requirements of the end users (as well as the previously stated requirements)	
8	We must meet regulatory guide lines such as data protection as well as software standards.	
9	Team members are not required to work outside scheduled practicals, which may affect project continuity and development speed.	
10	Constrained to the interoperability between Python and the MySQL database.	Our software is only as functional as the relationship between Python and MySQL.
11	Economically constrained by a zero-budget project, limiting us to free-to-use software.	This significantly limits what software we can use.
12	The app must cater to users with varying levels of technical proficiency, necessitating a simplified user interface.	We need to keep it as nontechnical as possible.

8.3 Dependencies

ID	Description	Comments
1	UI design adjustments depend on team feedback and approval.	The team must be aware before further changes are made.
2	The system itself will be dependant upon user interaction and having an active/interactive user-base who regularly upload adverts and respond to them	If we have no users then there will be no adverts and responses.
3	Backend functionality depends on the successful setup of the MySQL database.	We first need a database to be able to manipulate it with the backend.
4	Testing phase can only commence once the frontend, backend, and database are fully implemented.	We need to have a functioning system to test it.
5	To move onto implementing the backend and frontend, and initiating the testing phase both depend on the completion of the technical design document.	Specifically, deliverables, requirements, and software design must be completed.
6	Ideally the software itself will be dependent upon either iOS or Android operating systems.	
7	The system is dependent upon Python, the software we use to build it.	
8	Mobile app development depends on using Kivy, a Python framework.	
9	The system will be dependent upon external APIs such as the google maps API.	
10	The entire system will be dependent upon a MySQL database.	
11	Hardware-wise as the system is built for iOS or Android it will be dependent upon being run on a mobile device.	
12	The project’s version control will be dependent upon Git.	
13	The project must adhere to privacy laws ensuring user data is secure and encrypted.	
14	The system is dependent on the correct software version, virtual environment, and dynamic shift libraries.	High level volatility - requiring constant system maintenance and

		active server to function.

9. Software Design

9.1 System Architecture – describe design decisions (Architecture diagram)

Figure 5 - A representation of our overall architecture described in a layer diagram

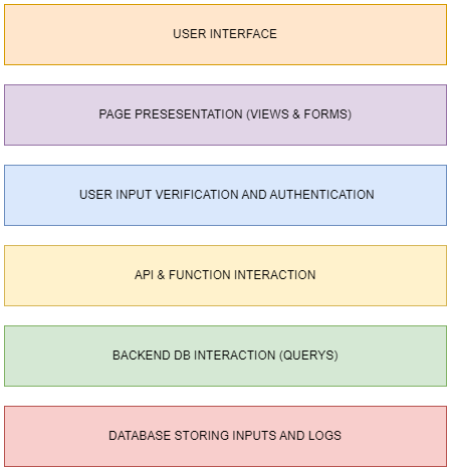
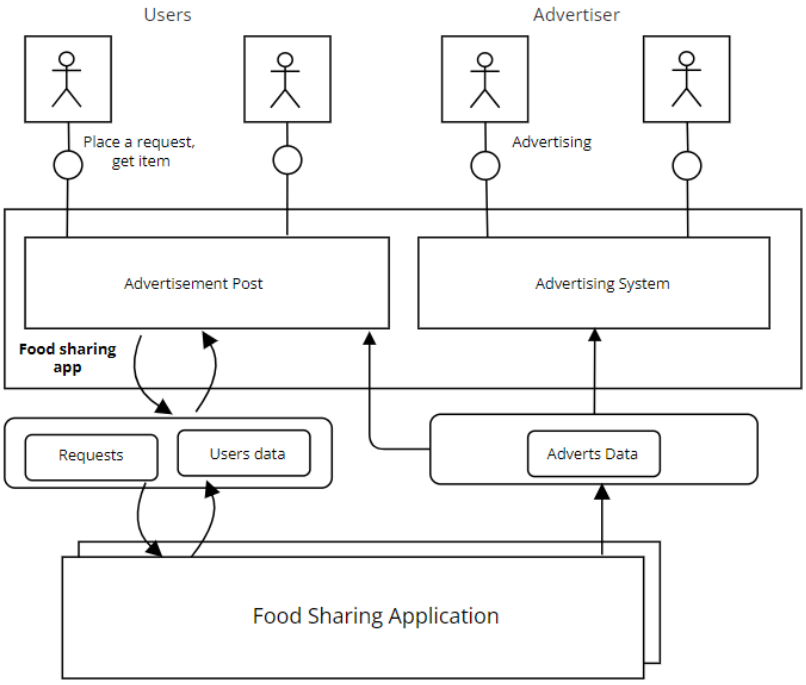


Figure 6 - Detailed Design: Block Diagram



9.2 High level overview of how the functionality and responsibilities of the system are partitioned and assigned to components (deployment diagram, component diagram)

Figure 7 - Showing a use case diagram of users and admins in our system and its intended interactions.


Post advertisement to donate food.
<<extend>>
Mark donation as complete.
When users successfully logged in, they can share all the posts (FR10) by clicking the share icon on the post.
<<extend>>
After being authenticated as user when login, they will have the access to make advertisement posts by filling in the form (FR7).
<<extend>>
Share post.
Mark donation as received.
Modify post.
Delete post.
Create post.
<<extend>>
<<extend>>
<<extend>>
Post request for food.
Moderate posts.
Search for post.
After being authenticated as admin during login process, , admins will have the access to review the database logs (FR15), and they can review the flagged posts (FR19).
Manage users.
Admin
User
Bookmark post.
When users successfully logged in, they can view all the posts (FR11) and saves the posts (FR12) to favourite pages by clicking the 'save' icon.
Review flagged users.
Search for user.
When user successfully logged in, they can search for specific users or adverts (FR20), by clicking the standardised 'search' button.
View audit logs.
Follow user.
Manage profile
Register for application.
Block user.
By entering username, password, and other required details in the registration form, users' data will be saved into the database, and their accountS are created (FR1).
Then, users can login by using the registered username and password (FR2).
Users can modify their details in the profile pages by clicking the pen icon (FR3).
<<extend>>
Log into application.
Report user.
Users can view the adverts, and report the advertisements(FR23) by clicking the 'flag' icon on top of the posts.

Figure 8 - Showing use case specifications for our system relating to some of the use cases of Figure 7

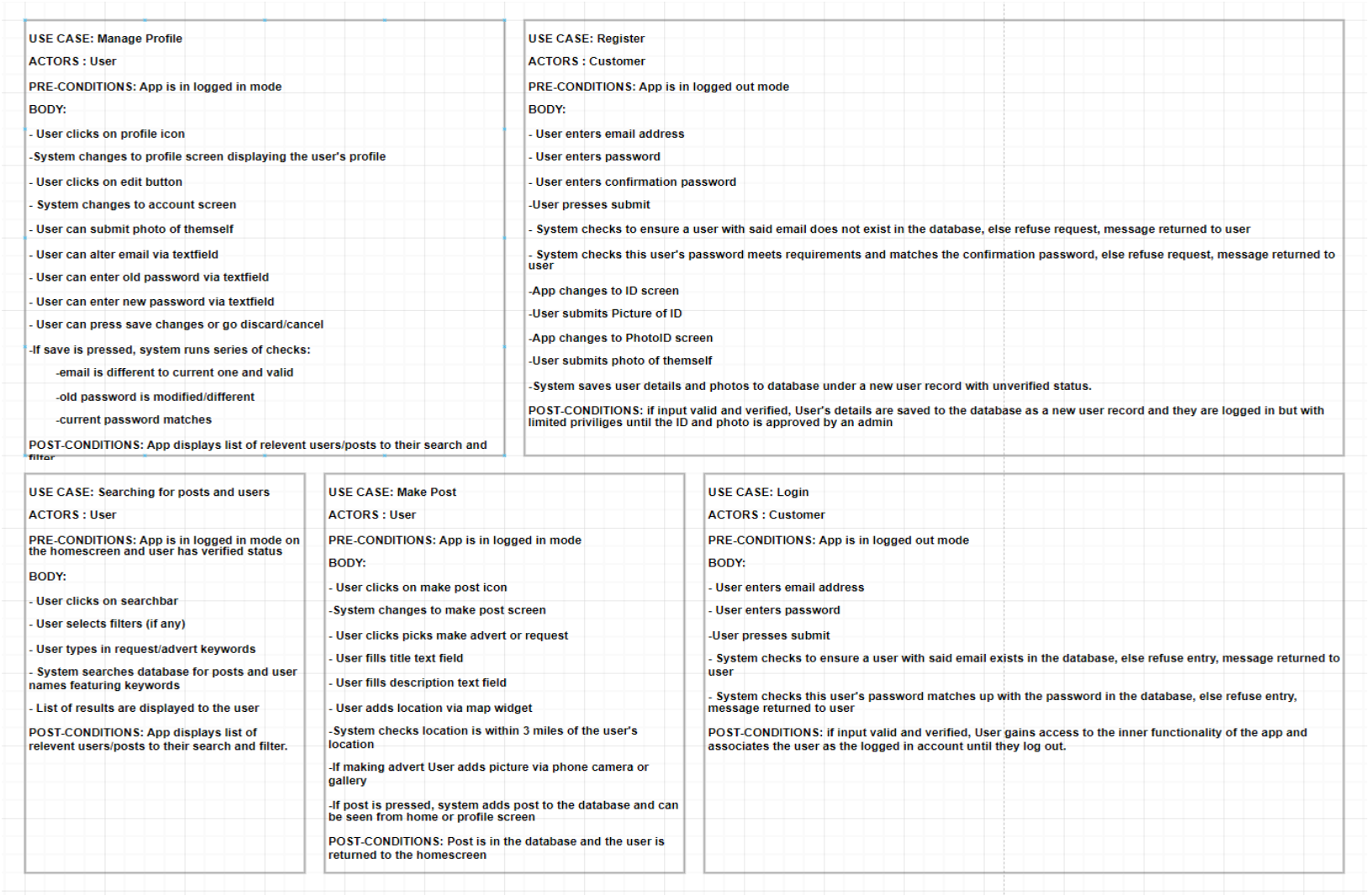
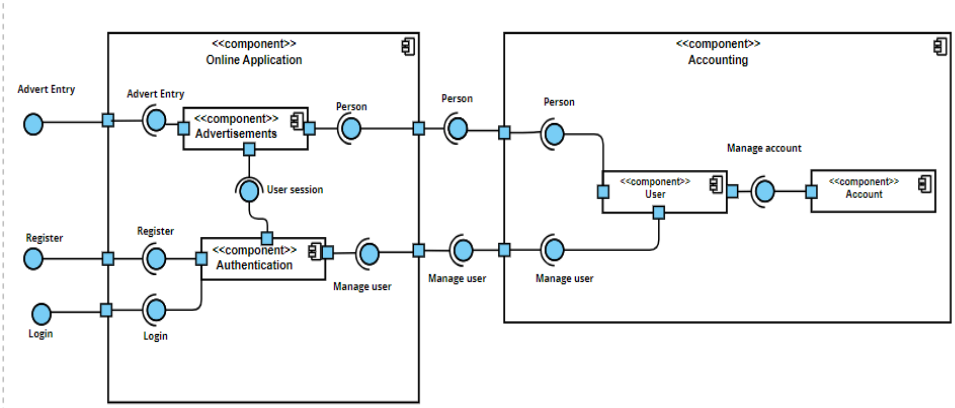
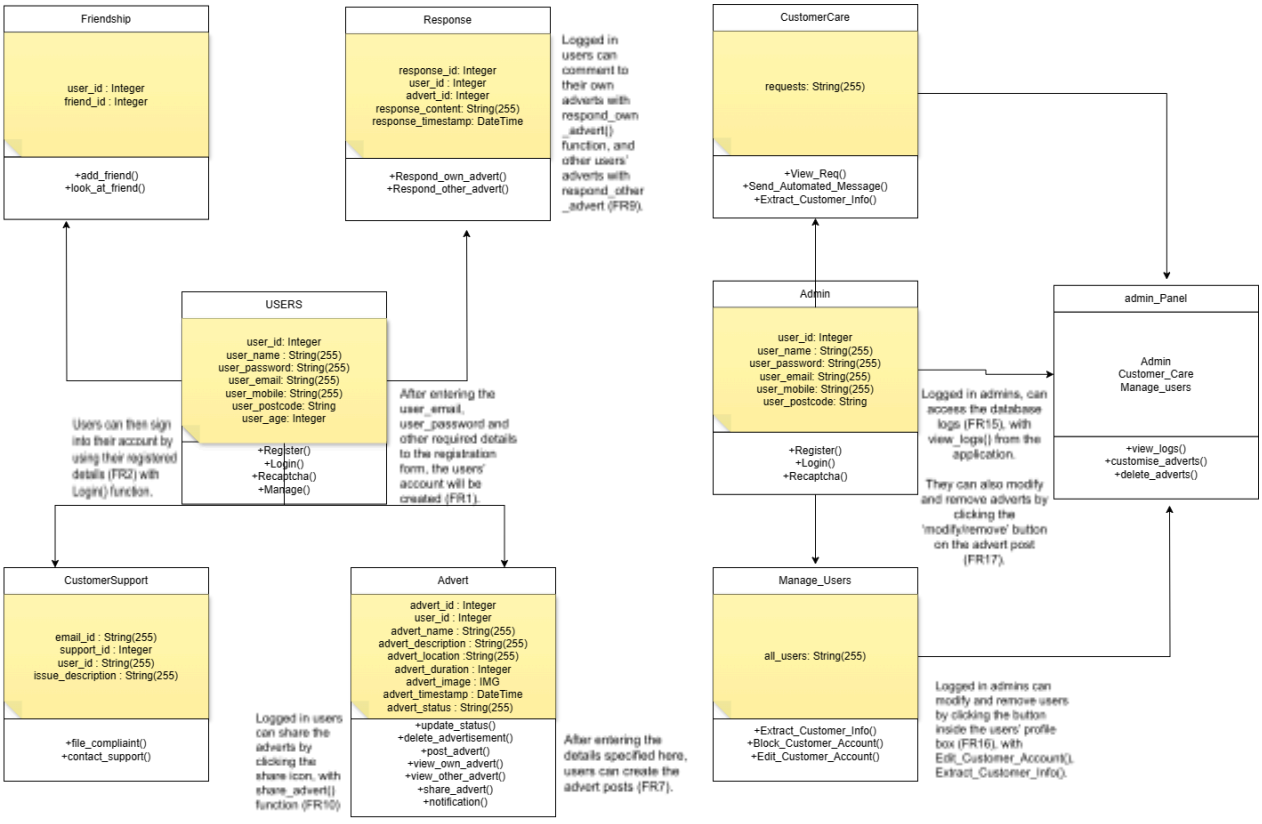


Figure 9 - Showing a component diagram of our system -
This diagram shows the components for FR7. After the authentication process, users are given user sessions that enable them to access the application, and make a post.



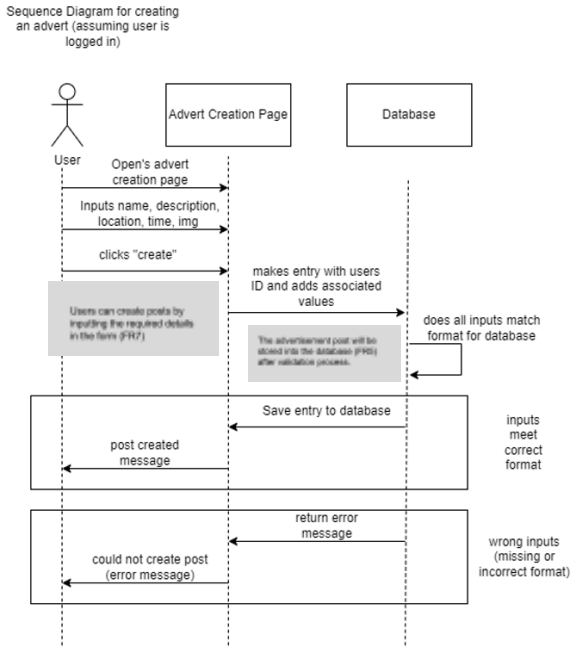
9.3 Package and Class diagrams which show dependencies between components

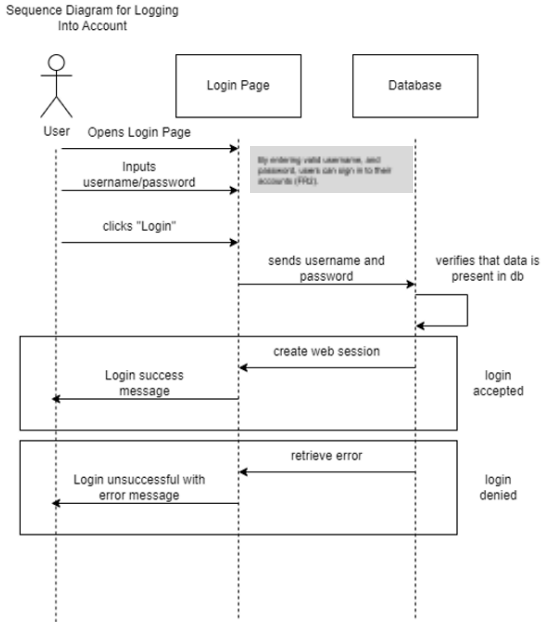
Figure 10 - A class diagram to represent the relationships between different classes of our system

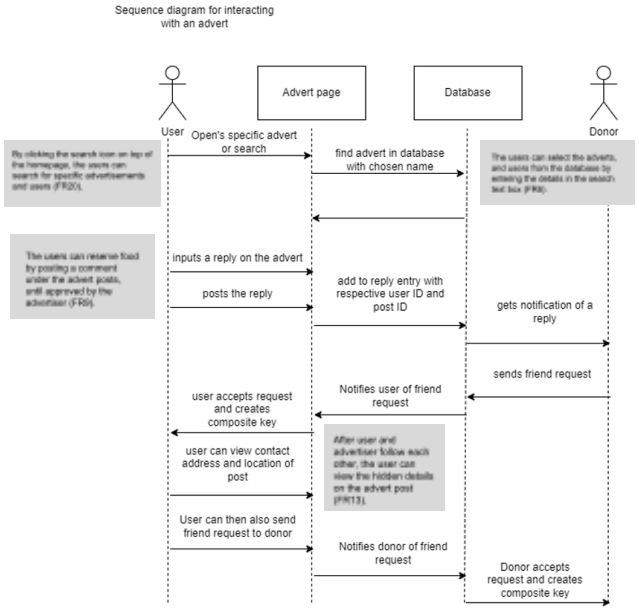


9.4 Show the dynamic behaviour of the system (sequence diagrams that show communication between objects. Activity diagrams that show the state of objects).

Figure 11 - Showing 4 different sequence diagrams for a variety of system interactions, These 4 diagrams also represent our goals of meeting functional requirements and represent some of the core functionalities for our software to meet.







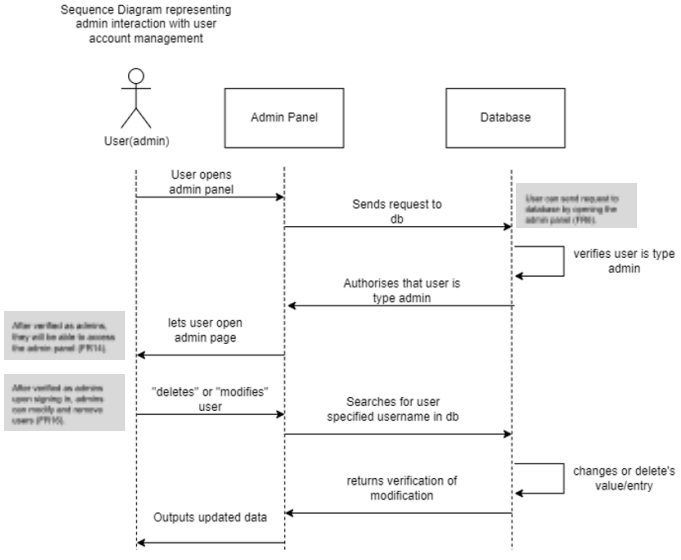
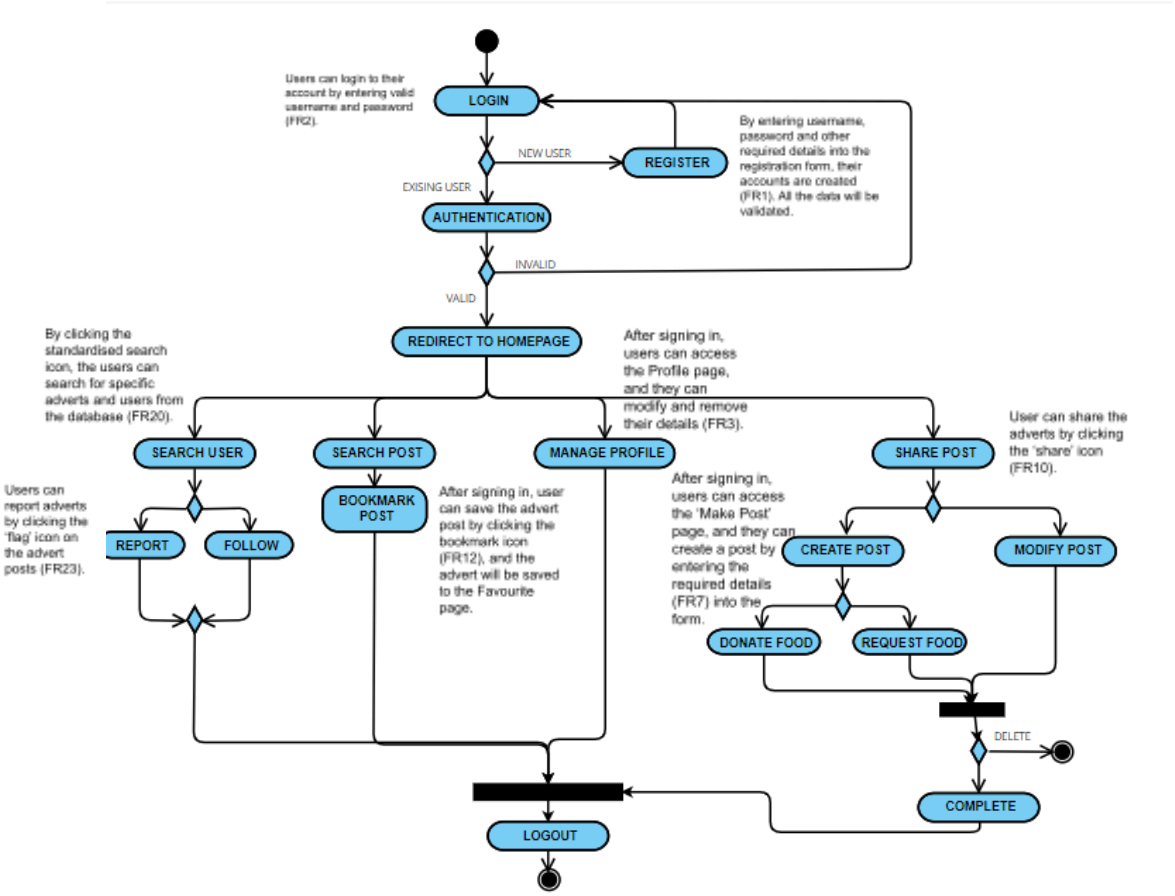


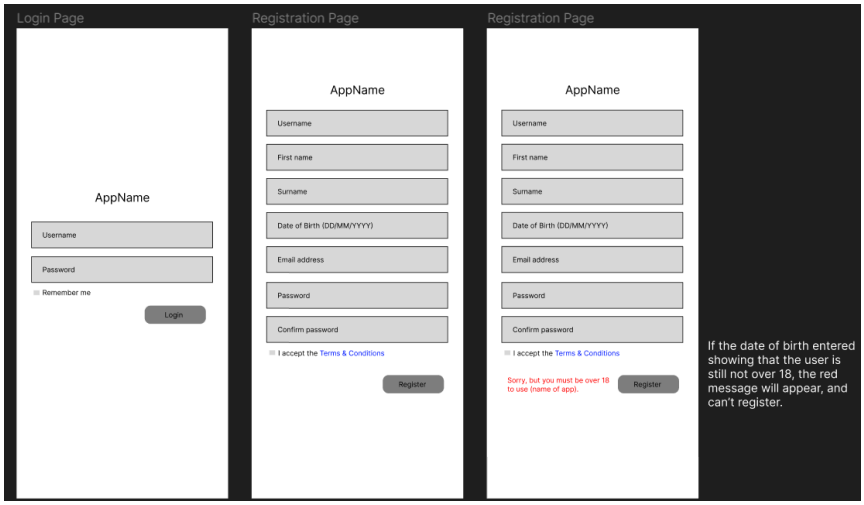
Figure 12 - Showing an activity diagram of the intended flow of our system. An advanced version of flow chart that modelling the flow from one activity to another activity.



9.5 GUI, Human Interface Views – Describe how the user accesses functionality

Figure 13 - Showing a prototype user interface of our application also available HERE

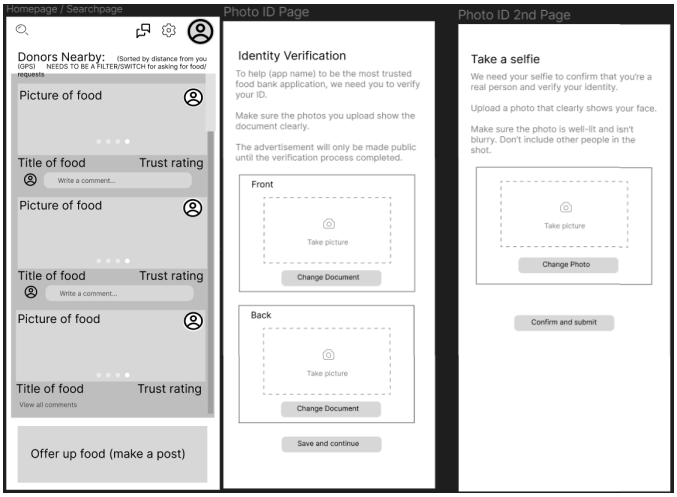
NB: Remember to associate aspects of design to the requirement number that they fulfil – communicate how your design meets the customer requirements.



The login page allows the user to login using email address, and password (FR1). Before signing in, unregistered users must create an account providing the email address, password (FR2), and other required details. The 'date of birth' text box in registration form will verify the age of the user, which must not exceed 18 years old to use the application (FR4).

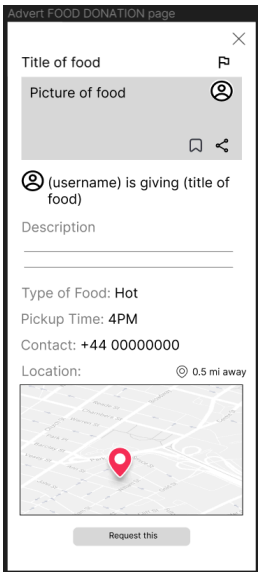
The homepage allows the user to search for any advertisements from specific users (FR20) using the standardised search icon. When users open the homepage, it will automatically list down the advertisements in the local area based on users' current location (FR28).

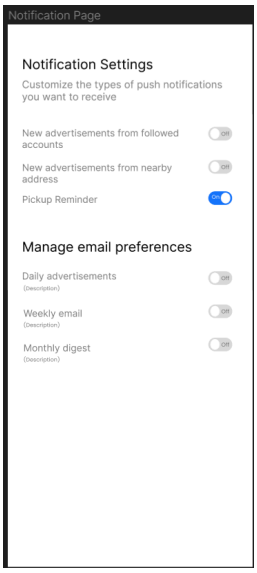
The second image, 'Photo ID Pages' showing the form for users to upload their ID, and selfie for verification purposes (FR8). The camera will be opened when the users click the camera icon in the boxes, allowing the users to take pictures.



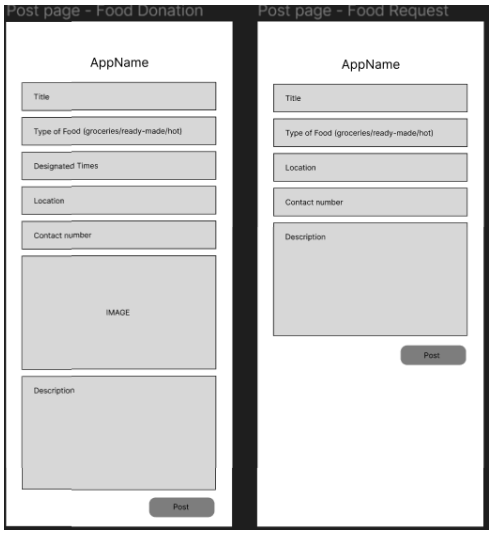
There will be icons allowing users to share, save, and report the advertisements posted (FR10, FR12, FR23). The icons mentioned will be familiar for users and large (width: 24, height: 24) to ensure complete accessibility for persons with visual disabilities (NFR12).

The second image ‘Notification Page’ provides the toggle bars for users to turn on and off the notifications easily based on their update preferences, such as receiving notifications for any adverts with nearby location (FR10) and recently added advertisements (FR21). The users can also set reminders to pick up items (FR22) by using the toggle bar.





The users will be able to post the form after filling in all the required details in the text boxes (FR7). The second image, ‘Profile Page’ allows users to make any changes to their details by clicking the ‘pen’ icon after they logged into their accounts (FR3). The changes will then be updated and stored in the database (FR6). The ‘pen’ icons are designed large (width: 34, height: 34) to ensure complete accessibility for persons with visual disabilities (NFR12).



10. Testing Methodology

As shown in the Gantt chart, we have dedicated one week to the testing phase of our project. Therefore, it is critical that this short amount of time is used effectively, hence why a testing methodology reflecting the nature of this project is required.

We plan to incorporate a mixture of Static and Dynamic testing since they both have their advantages and disadvantages (Guru99, 2024). Control flow testing can be applied by navigating through our program and comparing its status along our sequence/block diagrams to ensure the front end functionality and navigation sequences match our intended design (e.g FR11). We will frequently review source code in detail by explaining our code to the rest of the team as we write it, systematically examining each of our commits to ensure each change iteration matches the requirements of our specification and is compatible with the rest of our system.

Functional testing will be performed using the python library unittest. We plan to create individual test drivers for our backend files which automate tests of each function for various possible inputs such as data validation and retrieval/input to the database e.g FR5. Domain testing can be applied throughout by defining the minimum set of inputs required to adequately test a function with a stub. This would include valid, extreme and erroneous data as inputs and ensuring the correct output is given e.g: returning an error message as a result of validating a phone number with non numeric characters (FR1).

Each team member will be assigned a specific set of test cases in order to cover all possibilities through a divide and conquer technique. This inevitably leads to team members testing parts of the program that they have not designed, where the "internal structures are only partially known" (GeeksforGeeks, 2017). This will result in a style of black box testing in which we focus on validating our requirements and analysing internal parts of the program that they have respectively developed, referred to as grey box testing.

To aid the success of the project, it is vital that both non-functional and functional requirements retaining a priority of high or medium (as seen in the chart) should be tested for at a bare minimum. Although non-functional requirements measure the degree to which a functional requirement is met, it is important that we address the functional requirements first and use a staggered approach to testing followed by the non-functional requirements. This will hopefully result in a post-tested program that is bug-free, well-performing, and meets the criteria set by all our relevant requirements, not to mention the time-saving and quality control incurred from using these predefined requirements.

Furthermore, during this testing phase, we will cover a wide range of specific types of tests, each dedicated to a team member (as mentioned prior). This includes integration/compatibility tests, system tests, performance tests, stress tests, security tests, and finally usability tests to cover all aspects within our stated functional and non-functional requirements. Typically, all of these will be done with regression testing in mind. For example, if we run a performance test and upgrade the program's execution speed, we should then run regression tests to prevent any oversights or regression bugs from occurring. Generally speaking, a combination of all these types of tests, as well as the structure of manually testing the application when compared to the requirements and distributing this workload throughout the entire team, should provide us with an overall testing methodology that applies directly to our situation and given time/resource constraints.

Contribution Matrix

Task	Ishan (*student ID*)	Ain (*220319650*)	Dan (*student ID*)	Maria (*220145152*)	Drew (*220346715*)	Azam (*200549389*)
1. Background & Analysis	R,M	R, M	R	R	C, M	R
2. Deliverables	R	R	M,R	R	C, M	R
3. Project Plan	R	R	R	R	C, M	R
4. Hardware and Software platforms to be used	R	C	R	R	M	C, M
4.1 Hardware Specifications	R	C, R, M	R,M	R	M	C, M
4.2 Software	R	C, M	R	R	M	C, M
5. References	R	C	M,R	R	M, R	C
6. Definition of terms	R	C, R	R	R	M, R	C, R
7. Solution requirements	R,M	C, M	R	R	M	C, M, R
7.1 Functional Requirements	R	C, R, M	C,R,M	R	C, M, R	C, M
7.2 Non-Functional Requirements	R	R	C,R,M	R	C, M, R	M, R
8. Other considerations	R	R, M	R	R	M, R	C, M, R
8.1 Assumptions	R	R	R	R	M, R	C, M, R
8.2 Constraints	R	R	R	R	M, R	C, M, R
8.3 Dependencies	R,M	R	R	R	M, R	C, M, R
9. Software Design	C.M	C, M	R,C	C, R	C, M, R	C, M

9.1 System Architecture	C,M	C	R	R	R	R
9.2 High level overview	R,M	C, R, M	R,C	R	C, M	C, M
9.3 Package and Class diagrams	C,M,R	C, M	R	R	R	R
9.4 Dynamic behaviour	R	C, M	R	C, R	C, M	R
9.5 GUI, Human Interface Views	R	C, R, M	R	R	R	C, R
10. Testing Methodology	R	R	M,R	R	C, M	R