FEED MY FOOD BANK APP

Group 26:
Team 15 Minutes

Khalil Greenidge -2239987
kxg087@student.bham.ac.uk
Jan Gryga -2232206
jpg006@student.bham.ac.uk
George Harvey -2237296
gxh096@student.bham.ac.uk
Ting Chun Pan -2147761
txp061@student.bham.ac.uk
Amber Qureshi -1637737
axq037@student.bham.ac.uk
Sai Anirud Rayapeddi -2214626
nxr026@student.bham.ac.uk
Sarina Saqib -2249047
sxs1805@student.bham.ac.uk
Silvester Stephens -1437955
sxs1166@student.bham.ac.uk
Munir Suleman -1348560
mxs486@student.bham.ac.uk
Nicholas Turner -2188863
ngt063@student.bham.ac.uk

UNIVERSITY OF BIRMINGHAM
SEPP/BUS

Abstract

The COVID-19 pandemic has had a profound effect on society leading to record levels of unemployment. Due to this, there has been an unprecedented surge in community reliance on food banks. The Feed My Food Bank app has been designed to provide much needed assistance to communities, working hand in hand with established food banks. The app allows for a contact-free experience both when donating and requesting support from local food banks.

This document outlines the design and use of the app. The Unified Modelling Language (UML) was used to help develop the Feed My Food Bank app. The functional and non-functional requirements of the app are discussed, detailing the services provided and the constraints on the system. Additionally, use cases and scenarios have been detailed followed by the UML diagrams (activity diagrams, class diagrams, sequence diagrams and state diagrams). These diagrams detail different instances of uses of the app, describing the process a user would take for both donating, and requesting a package through to completion of both processes. Alternative flows such as history viewing, and failed login are also outlined. The development process also included the evaluation of different types of architecture for the proposed app, with a comparison between the suitability of both event driven and 3-tier architecture. Furthermore, the software testing plan, which would be used on both the front and back end elements of the app, are detailed in order to evaluate and ensure high functionality and intelligent design. Equivalent Class Partitioning and Boundary Value Analysis were implemented into the Black Box testing by using specific values within a range. Prototype screenshots and an accompanying video highlighting the functionality of the application have been designed. Finally, an ethical appraisal has been conducted. The analysis evaluated the potential ethical issues that could arise from the app including data security issues, in order to verify that the proposed system upholds ethical and GDPR considerations.

Contents

1.0. Introduction

The devastating impact of COVID-19 has been felt across society, creating new and unique problems for different UK communities. For this project, we decided to focus on those relying on food banks to feed their families due to financial difficulties. Research from the Trussell Trust (Trussell Trust, 2020b) highlights that half of all food bank users at the beginning of the pandemic were using them for the first time, the majority of whom are families with children (Butler, 2020a). They also estimate that food banks will be distributing 6 emergency food parcels per minute in the UK during the winter season (Trussell Trust, 2020a). This tremendous rise in demand is not only putting pressure on food banks and their supply, but also increasing crowding at food banks (Butler, 2020b). This leads to further spread of the virus, creating a disproportionate impact of COVID-19 on lower income families.

By increasing food bank supply through community involvement and providing contactless delivery of food packages to vulnerable households, our application, Feed My Food Bank, provides a software solution to this problem.

2.0. Scope

Our proposed system is a platform that connects three different parties: Food Banks, Donors, and Recipients (those who need support). Not only will Feed My Food Bank facilitate the contact-free distribution of resources, but it will also provide a seamless donation interface. The app will improve the distribution of resources through a coloured ranking system indicating the supply level at each food bank. This will help users to decide where is best to both access support and donate. Food banks will be able to confirm or deny each request as it is made, this accounts for walk-in support requests and unforeseen drops in stock. This feature also protects against abuse of the app. There are three main assumptions for this system. The first is that the delivery of the food packages will be handled by the food bank, this can be achieved by utilising their volunteers to help with delivery. The second assumption made is adoption of the app by all food banks, to accurately reflect the number of food banks and the resources in the area. The final assumption is that the app will only be available within the UK.

Food banks must first register with the app. Each food bank is required to input the number of food packages they have available daily which will be used to generate the scale for supply levels in the area. The top figures between the top 66-100% will be assigned green, for high. Those figures between 33- 66% will be assigned yellow, for average, and those 33% and below will be assigned red, for low. Each time a food package is ordered by a user, the supply levels at that food bank and the coloured ranking system will also be updated (if it falls into the lower category) to reflect this change and consistently provide up-to-date information for the user. When setting up their accounts, food banks must also provide banking details to enable them to receive donations.

A user that needs the app to request support is required to create an account containing information such as name, address, phone number and email. This is needed for delivery, and communication with the user to provide both confirmation and updates on their request for support. Users will be asked to input the number of people in their household to assess the level of support they require. Users will then see local food banks with a coloured ranking system, enabling them to request support from a food bank with sufficient stock. Once a food bank is selected, they will be presented with several options for different sized food packages, to account for the differing levels of support required. Users will not be shown packages that contain more resources than what is needed for the size of their household. Once the user's request is confirmed by the food bank, they will receive their package within 24 hours via contact-free delivery.

Those who wish to donate are able to do so directly through the app. This user must also create an account. The user will enter their postcode and be provided with a list of food banks in their local area with a coloured ranking system to show them where their donation is the most needed. After selecting a food bank, they will be taken to an interface that gives three donation options: small package, medium package and large package. Once their option has been selected, they will be taken to a payment subsystem to confirm their transaction and take payment. Donors will also be able to view their accounts donation history.

3.0. Requirements Analysis

The following section aims to detail both the services that the user would require from the system (Functional Requirements) and the constraints under which the system operates and is developed (Non-functional Requirements).

3.1. Functional Requirements

1. User Accounts

1.1. The system shall require the Food Bank to register

1.2. The system shall require the Recipient to register

1.3. The system shall require the Donor to register

1.4. System registration:

1.4.1. The system shall require the Food Bank to enter:

- Food Bank name
- Charity Number
- Address
- Email
- Password
- Phone Number

1.4.2. The system shall require the Recipient to enter:

- Name
- Address
- Email
- Password
- Phone Number
- Number of occupants within their household

1.4.3. The system shall require the Donor to enter:

- Name
- Email
- Password
- Postcode

1.5. The system shall allow the users to login into their account using their registered email and password

1.6. The system shall allow the users to change their information

2. Credentials

2.1. The system shall allow for the charity subsystem to check the charity number supplied by the Food Bank

2.2. The system should check the user login details with those stored in the database

3. Distance

3.1. The system shall provide the Recipient and the Donor with a list of registered food banks within a 10-mile radius of the provided postcode using the map subsystem

4. Packages

4.1. The system allows the Food Bank to input and update the types of packages, the number of packages available and the cost of the packages

4.2. The system shall display to the Donor the type and cost of the packages

4.3. The system shall display to the Recipient the type of packages available to them depending on the number of occupants within their household

4.3.1. Available packages for Recipient depend on the number of people within their household

4.4. The system allows the Recipient to order only from the available packages

4.5. The system should restrict Recipient from ordering another package within 24 hours of an approved previous package

5. Payment

5.1. The system shall allow the Donor to select the amount of money to donate according to the Food Bank and the packages they supply

5.2. The system shall ask what package size they want to pay for, and what quantity of those packages they want to pay for

5.3. The system shall access the payment subsystem to process the donation

5.4. The system shall notify the Food Bank regarding the donation

5.5. The system shall notify the Donor regarding the receival of donation

6. Supply Level

6.1. The system shall display the level of supply for each Food Bank using a colour system (Red = low supply, Yellow = average supply, Green = good supply)

6.2. The system shall calculate the average supply level of all the Food Banks registered within the city

6.3. The system should calculate the supply level percentage and assigns a colour respectively (i.e. 0-33% = Red, 33-66% = Yellow, 66-100% = Green) to the Food Bank

6.4. The system updates the supply level within the database as package quantity changes

6.5. The system displays to the Food Bank the number of packages available according to the system/database

6.6. The system shall restrict the request of food packages that are no longer available due to depleted supply

7. Concurrency

7.1. The system shall prevent multiple users from booking the same package via the accept/deny button from the food bank

7.1.1. The accept button will confirm the Recipient's order and will update the supply level with the database

7.1.2. The deny button will reject the order

8. Notification

8.1. The system shall notify users via the email subsystem

8.1.1. The subsystem shall notify the Food Bank:

- The donations that have been received
- The daily report of the orders

8.1.2. The subsystem shall notify the Donor:

- Confirmation of payment sent

8.1.3. The subsystem shall notify the Recipient:

- Confirmation of approved, denied and/or cancelled orders

8.2. The system shall notify users within the system:

8.2.1. The system shall notify the Food Bank:

- Package requests from Recipient
- Cancellation of orders

8.2.2. The system shall notify the Donor:

- Confirmation of payment sent

8.2.3. The system shall notify the Recipient:

- Confirmation of approved, denied and/or cancelled orders

9. Stored/Displayed Information

9.1. The system shall store the following information within a database:

9.1.1. Registration information (see 1.4)

9.1.2. Package type and cost

9.1.3. Package quantity

9.1.4. Orders: Status (confirmed or pending) and type of package

9.2. The system shall display the following information to the users:

9.2.1. For the food bank:

- History of accepted orders (Recipients name, date)
- History of donations
- Quantity and type of available packages

9.2.2. For the Recipient:

- History of accepted orders (Food Bank name, date)
- Quantity and type of packages ordered
- Status of order (i.e. Pending)

9.2.3. For the Donor:

- History of donations (Food Bank name, data, amount)

10. Miscellaneous

10.1. The system shall allow the Food Bank to accept or deny order requests

10.2. The system shall allow the Food Bank and the Recipient to cancel orders within a given time limit

3.2. Non-Functional Requirements

1. Resources Efficiency

1.1. Performance

1.1.1. The system shall be able to handle multiple user requests without system failure

1.1.2. The system shall be able to access the database within 3 seconds

1.1.3. The user is redirected within 10 seconds to the payment subsystem (PayPal)

1.1.4. The system should populate the map with food banks within 5 seconds

1.1.5. The system shall be able to notify the users within 10 seconds

1.1.6. The system shall be able to display the locations according to the postcode via the map subsystem within 10 seconds

1.1.7. The system shall be able to update the supply level colour every 30 seconds

1.2. Space (capacity)

1.2.1. The system shall be able to store data relating to 1,000,000 users
1.2.2. The system shall be able to record and store the user's registration information up to 10TB
1.2.3. The system shall be able to record and store package information up to 30TB
1.2.4. The system shall be able to record and store donation information up to 10TB
1.2.5. The system shall be able to record and store the order history up to 50TB

2. Security

2.1. The system should ensure that the data stored will be secure from other parties
2.1.1. The database will be located in the UK
2.1.2. The database should be protected from Structured Query Language (SQL) injection attacks
2.2. The system should ensure that the shared data between the systems, subsystems and other parties are encrypted using Secure Sockets Layer (SSL)

3. Reliability

3.1. The system should be online at least 90% of the time, while 2% should be in maintenance
3.2. The system should have safeguards to minimize Distributed Denial of Service (DDoS) attacks
3.3. The system shall ensure to maintain a mean time between failures (MTBF) of 720 hours

4. Usability

4.1. Users should be able to understand the system easily within 3 hours
4.2. The system should have a simple interface with a consistent layout and colour scheme
4.3. The system should use the same language (English) throughout its interface

5. Maintainability

5.1. The system should ensure that the servers will run autonomously with a maximum of 10 hours of maintenance per month
5.2. The system shall ensure to maintain a mean time to recovery (MTTR) of 30 minutes

6. Portability

6.1. The system shall ensure its usability in different mobile devices/environments including Android and iOS platforms
6.2. The system shall ensure its adaptability with the newer versions of mobile devices

7. Scalability

7.1. The system shall be developed such that it can be scaled to allow 1,000 new users per month
7.1.1. The system can expand by integrating multiple servers and backup databases
7.2. The system should be developed to allow for adoption in different countries
7.2.1. The system should integrate with the new databases located in the new locations
7.2.2. The system once scaled should support at least 5 different languages

8. Interoperability

8.1. The system should be able to access the UK charity database
8.2. The system should be able to access the third-party payment subsystem (e.g. PayPal)
8.3. The system should be able to access the map subsystem (e.g. Google Maps)

9. Standards

9.1. Privacy
9.1.1. The system should ensure only the required information and the information users have made public will be displayed to other users
9.1.2. The system should only provide the required information to different subsystems
9.2. The system shall comply with GDPR (General Data Protection Regulation) policy
9.3. The system shall comply the standards provided by the various third parties that the system interacts with (i.e. payment subsystem, charity subsystem, etc)

4.0. Use Case Diagram

This section aims to summarise the interactions of the users (actors) and the system by means of a use case diagram (Figure 1). The Food Bank, Donor and Recipient are the primary actors. The database, charity validation, email, map and payment actors are external systems being used by the system.

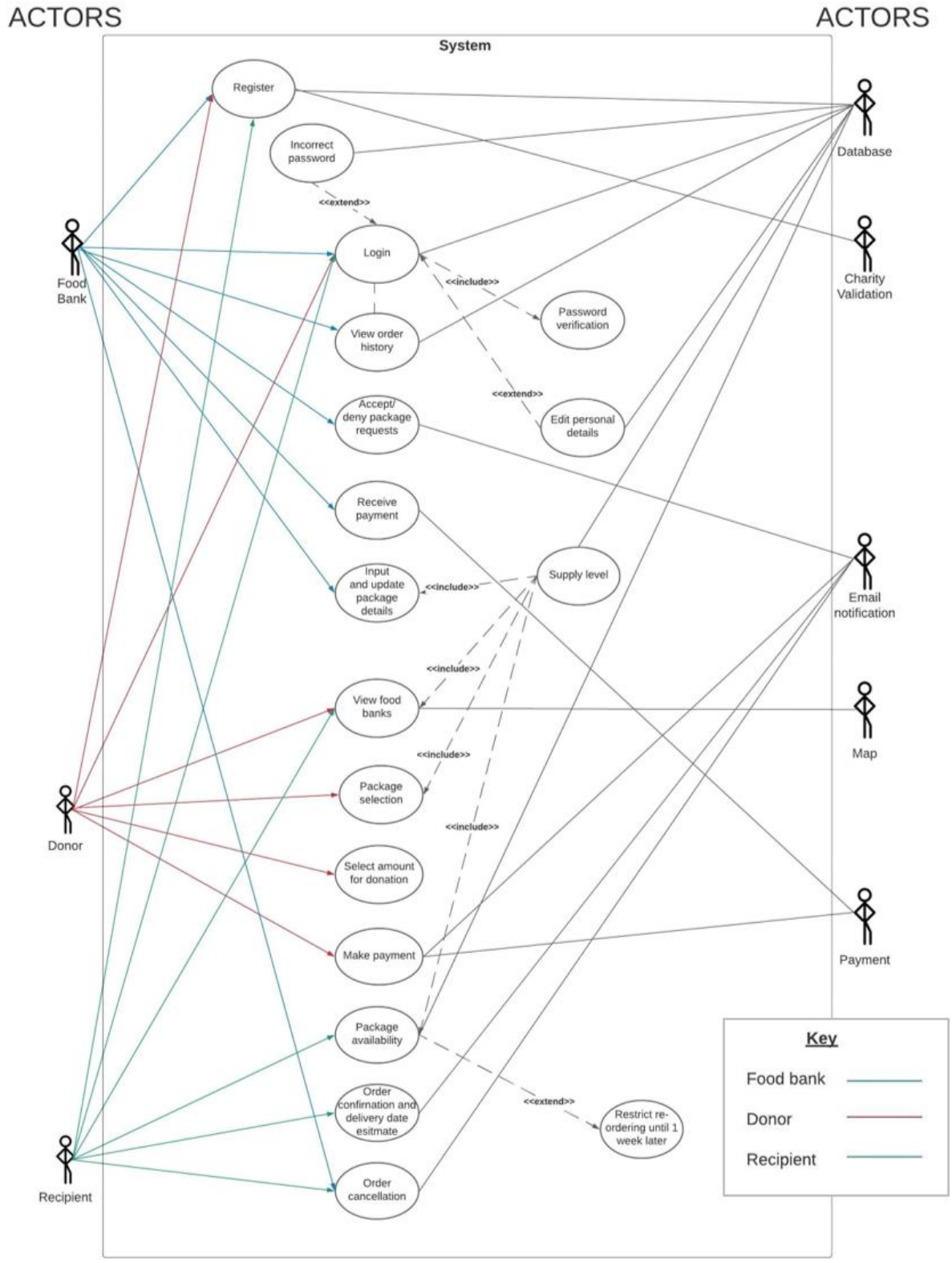


Figure 1: UML use case diagram

5.0. Use Cases

This section details two use cases from start to end, highlighting the interaction between the user and the system to achieve a goal. The two documented use cases are:

1. A Donor donating a package to a food bank
2. A Recipient ordering a food package

5.1. Use Case 1

This is a non-trivial use case that highlights a Donor's journey through the app to donate a package. It highlights the different actors the Donor interacts with, in order to donate a package.

Preconditions

The user is a Donor.

Food banks are registered, and supply exists.

Flow of Events:

1. The use case starts when the Donor launches the app and is asked to either:
    a. Login
    b. Register
2. The Donor chooses to log in and is prompted to enter login details (email and password)
3. The system verifies the login details and that the user is a Donor
4. The system verifies they are a Donor, it displays the right UI (user interface) for Donors
5. The UI displayed to the Donor are as follows:
    a. Map with all local food banks in the area
    b. An option of food banks which they can donate to and their supply levels. (Red = low supply, Yellow = average supply, Green = good supply)
    c. View my (Donor) donation history
    d. View account information
6. In this use case the Donor selects a food bank to donate a package to
7. The system displays the options of predetermined packages that the Donor can possibly donate:
    a. The options are Small package, Medium package and a Large package
    b. The quantity of packages to donate are set at a default of one package
    c. Each package has a cost that is displayed with the package
8. The Donor selects one of the three options to donate:
    a. If the Donor selects a small package further information and cost of the package is displayed to the Donor
    b. If the Donor selects a medium package further information and cost of the package is displayed to the Donor
    c. If the Donor selects a large package further information and cost of the package is displayed to the Donor
9. Once the Donor has chosen a package, the system displays a message to either:
    a. Confirm
    b. Cancel
10. Once the Donor has confirmed, they are sent to a payment subsystem which processes their payment
11. The payment is successful, and the system displays the confirmation message and the donation
12. The subsystem sends a confirmation email to the Donor

13. The system updates inventory of packages in the chosen food bank and updates the Supply Level and Database

Alternative Flow: Donor chooses to register at step 1

1. If the Donor chooses to register, they are asked to enter;
    a. Name, email, password and postcode
2. Email subsystem sends an email to the Donor
3. User confirms account by clicking the verify link in email
4. Return to the primary flow of events step 2

Alternative Flow: Donor chooses to view their history at step 5

1. Donor chooses to view donation history
2. System displays the donation history of the Donor
3. After viewing history Donors choose to return to the view of food banks in the local area
4. Return to primary flow of events step 5

Alternative Flow: Incorrect login information at step 2

1. If the Donor enters incorrect login information an invalid information message is displayed
2. Return to the primary flow of events Step 2

Alternative Flow: Failed payment at step 10

1. If the subsystem has an unsuccessful transaction with the Donor account, payment subsystem returns the Donor to the application
2. Return to step 7

Alternative Flow: Cancellation at step 9

1. If the Donor chooses to cancel an order
2. Return to step 8

Post:

1. The Donor has been charged by the payment subsystem if the transaction is successful
2. The system updates the supply level and database
3. The Donor receives an email confirmation of donation via the email subsystem
4. The food bank is notified of the donation
5. The payment subsystem sends the payment to the food bank

Actors:

The primary actor in this use case is the Donor who initiates the use case by launching the application. The actor involved in the use case is the payment subsystem. The payment subsystem processes the Donor's payment and confirms the payment has been successfully completed.

5.2. Use Case 2

This is a non-trivial use case that highlights a Recipient journey through the app to request a package. It highlights the different actors the Recipient interacts with in order to receive a package.

Precondition:

The user is a package Recipient.

Food bank is registered and has a supply of packages.

Flow of events:

1. The use case starts when the package Recipient launches the app and is prompted to:
    a. Login

   b. Register
2. The Recipient chooses to log in and is prompted to enter login details
3. The system verifies the login details and that the user is a package Recipient
4. The system verifies they are a package recipient, it displays the right UI for package recipients
5. The options displayed for the package recipient to choose from are as follows;
   a. Map with all local food banks in the area
   b. An option of food banks which they can order a package from and their supply levels. (Red = low supply, Yellow = average supply, Green = good supply)
   c. View my (package recipient) order history
   d. View account information
6. The package recipient chooses the closest food bank to them to order a package
7. The system determines which packages can be displayed to the package recipient based on the information on the system for the package recipient
8. The system prompts the package recipient to choose a package size:
   a. Small
   b. Medium
   c. Large (The system only allows large package selection for larger households)
9. The package recipient chooses one of the options displayed by the system
10. The system displays a confirmation of the choice:
    a. Confirm
    b. Cancel
11. Package recipient confirms the package they would like to order
12. The system sends a request to the food bank to either:
    a. accept the order of a package
    b. deny the order of a package
13. The system confirms the order has been successfully accepted by the food bank to the package recipient

Alternative Flow: Recipient chooses to register at step 1

1. If the Recipient is not registered, they are asked to enter:
   a. Name, Email, Password, Postcode, Phone Number and Number of occupants within their household
2. Email subsystem sends a verification email to the Recipient
3. User confirms account by clicking verify link in email
4. Return to the primary flow of events step 2

Alternative Flow: Recipient chooses to view their history at step 5

1. The Recipient chooses to view donation history
2. System displays the order history of the Recipient
3. Return to primary flow of events step 5

Alternative Flow: Incorrect Login information at step 2

1. If the Recipient enters incorrect login information an ‘invalid information’ message is displayed
2. Return to the primary flow of events Step 2

Alternative Flow: Food bank chooses to deny the order of a package at step 12

1. The food bank chooses to deny the order of a package
2. Email subsystem sends an email to the Recipient, updating the Recipient on their request being denied

3. Use case ends

Alternative Flow: Cancel package at step 10

1. If a Recipient chooses to cancel a package
2. Return to step 8

Post:

1. The system notifies the food bank that a package has been ordered
2. The system sends the address and contact information of the package recipient to the food bank
3. The system updates the supply level and database
4. The package recipient receives an email confirmation of the request of the package and estimated delivery date
5. The system updates the information displayed to the package recipient's order history
6. The package recipient is restricted from ordering another package within 24 hours

Actors:

The primary actor in this use case is the package recipient, who initiates the use case by launching the application. The other key actor in this use case is the food bank. The food bank is the final actor in this use case as they need to accept or deny a package request.

6.0. Scenarios

This section details a couple of scenarios of people walking through the Feed My Food Bank app. Each scenario highlights the details of one of the use cases presented in the above section.

6.1. Scenario 1, Use Case 1 – Donate Package to Food Bank

Meredith lives in the centre of London in the midst of a country wide lockdown. Meredith and her husband live together and are fortunate enough to be able to work from home during the pandemic. During one of the weekly grocery deliveries, Meredith talks to the delivery driver who mentions he also volunteers as a driver for food banks around Vauxhall (her local area), where he delivers food packages to people who need a little help due to the current situation. Meredith loves the idea of her local community coming together to support one another and wants to get involved too. After asking the driver how she could help, he directs her to the Feed My Food Bank app.

Meredith downloads the app on her smartphone and opens it. She is prompted by three options: 'I want to donate', 'I am in need' and 'I am a food bank'. Meredith wants to donate to a local food bank so clicks 'I want to donate'. The app then prompts her to either register or log in to an existing account. Meredith does not currently have an account, so fills in the required information (name, postcode, email and password) and accepts the terms and conditions by clicking the check box, before clicking on 'Register'. Meredith receives an account confirmation email, in which she clicks a link to verify her account. After this verification she is able to log in into her account on the Feed My Food Bank app.

Meredith now sees a list of food banks within a 10-mile radius to her, ordered proximity (closest first). She recognises the second food bank on the list (Vauxhall FoodBank) and sees a red box next to it and a label saying, 'Supply Level: Low'. She scrolls through the other food banks listed and the colours next to them and realises a traffic light coloured system is used to indicate the food supply levels at each food bank. She scrolls back and clicks on Vauxhall FoodBank and is presented with a selection of 3 packages: small, medium and large. She sees a cost next to each package type and a little description underneath each one. She reads the description under the small package which reads, "This package can feed a household of up to 3 people for 1 week" and sees its cost at £15. Meredith sees the quantity set by default is 1 package and clicks on this package. The app prompts a

confirmation message, “Would you like to donate a small package of £15 to Vauxhall FoodBank?” with a ‘Donate’ button underneath it. Meredith reads the message underneath the donate button which reads “By clicking this button you will be redirected to a third-party payment service.” Meredith clicks the donate button. The payment subsystem handles the donation and sends this to the specific food bank. After payment, Meredith is automatically redirected back to the app, where a ‘thank you’ message notification pops up. A confirmation of her donation is sent to her email via the email subsystem. Meredith’s donation is displayed in her donation history tab within the app.

6.2. Scenario 2, Use Case 1 – Donate Package to Food bank

Raj, a registered Donor to the Feed My Food Bank app, lives in Birmingham in the Sparkhill area. He prefers using the app over donating in person due to the risks this presents within the current COVID climate. He often donates to his local food banks and would like to see how many packages he has donated in total before donating another package. Raj opens the app which he has not opened for a couple of weeks and is thus required to log in again. Raj types in his email and password and clicks ‘Login’. A loading wheel shows on his screen for a few seconds as the app logs him into his Donor account.

Once Raj is logged in, he can once again see a list of the food banks in his area. At the bottom of the screen he can also see a ‘Donation History’ tab. He clicks on this as he would like to see the food banks he has previously donated to and the amount he has donated. The app opens a list of his donations in chronological order, with the topmost entry being his latest donation he had made two weeks ago. By scrolling through these donations, he realises he has donated most of the packages to Sparkhill Foodbank, which is local to him. Raj clicks back to the home tab which displays the food banks within a 10-mile radius. At the top of the list, Raj sees food banks closest to him, but all of these have a green box with a ‘Supply Level: Good’ label next to them. Raj decides to donate to another food bank which has a shortage of supplies and scrolls down the list until he sees a red box next to Ladywood Foodbank.

Raj clicks on Ladywood Foodbank and is prompted to choose from a list of packages he would like to donate small, medium or large. He sees the cost of each package and would like to donate one medium package to the food bank. He clicks on this and is prompted by the app to confirm his selection by clicking the ‘Donate’ button. Raj changes his mind and would instead like to donate a large package. He clicks the ‘X’ at the top right of the prompted message which brings him back to the select package page. He then clicks on the large package and this time confirms the donation by clicking ‘Donate’. Raj is redirected within 5 seconds to the payment subsystem, where the money is transferred to the food bank. He is automatically redirected back to the app, where a thank you message notification pops up. A confirmation of his donation is sent to his email via the email subsystem. Raj’s donation history tab is updated, and this most recent donation is displayed at the top of the list.

6.3. Scenario 3, Use Case 2 – Ordering a Food Package

Neal and his wife have two children, a 2-year-old boy and a new-born girl. Due to the current COVID situation and lockdown restrictions that were placed in Manchester, Neal is unable to continue working on his event planning business and is thus struggling to support his family. He knows once the restrictions are lifted, he will be able to support himself and his family once again, but for this tough period he needs help. He hears about the Feed My Food Bank app, which allows communities to donate to their local food banks, but also allows for people who need a little help to request some packages from their local food banks. Neal is hesitant to ask for help but is in a desperate situation. He downloads the app.

Neal opens the app and the system presents him with three clickable options: ‘I want to donate’, ‘I am in need’ and ‘I am a food bank’. He clicks on the ‘I am in need’ button. The app requires him to either

register or login to continue with the process. Neal fills in the required information (name, full address, phone number, number of occupants within his household, email and password) and accepts the terms and conditions by clicking the check box, before clicking on 'Register'. Neal checks his email for an account confirmation email wherein he clicks a link to verify his account. Neal returns to the app, enters his email and password and clicks 'Login'.

Once logged in, Neal can see a list of food banks within a 10-mile radius to him, which are also registered to the Feed My Food Bank app and can deliver food packages. Next to each food bank is a coloured block and supply level label, which indicates the supply level of the specific food bank. Neal clicks on the first food bank, which is the one closest to him. The app displays the list of packages available to him from that particular food bank. Neal can only see the small and medium packages. The system restricts Neal from seeing the large package as available, as he does not meet the required number of people within his household to receive the large food package. Neal can see a description of each package type and selects the medium package which is listed as "This package can feed a household of up to 5 people for 1 week". By selecting this package, the app prompts a confirmation message, "You have chosen to receive a medium package from Burnage Foodbank?" with a 'Receive package' button beneath the message. Neal confirms his selection by clicking on the 'Receive Package' button. The app then displays the following message to Neal: "Request sent to food bank. Please wait as the food bank reviews the order".

After a short period of time (1 hour), a confirmation message is sent within the app to Neal: "Your request has been accepted by the food bank. Your food package shall be delivered to you within the next 24 hours. Thank You!". The email subsystem also sends a confirmation email to Neal regarding his accepted food package order.

6.4. Scenario 4, Use Case 2 – Ordering a Food Package

Ronda is currently living with her grandparents, husband and 3 children. Due to COVID lockdown restrictions and the current situation with her job, she requires a little help to support her whole family. Having previously donated to food banks, she realises she could get the little help she needs from her local food bank. However, Ronda is wary of going outside too often due to both her grandparents being vulnerable in relation to the COVID virus. So, she logs in to the Feed My Food Bank app and has previously ordered 1 food package.

Ronda now needs to order another food package for her family, but the previous package was too large for her needs and so she decides to order a smaller package this time. She opens the app and clicks the 'Login' button. She types in her email and password and presses 'Login' but within a few seconds the app returns an error message: "Invalid Credentials. Your email and password do not match those stored in our system. Please try again". Ronda tries again realising she typed her password wrong the first time and is now successfully logged into the system within a few seconds.

She clicks on her 'Package History' tab at the bottom of the screen to see the size of the package in her previous order. Having seen it was a large package, which would feed more than 5 people for one week, Ronda decides she will order a medium sized package instead. Ronda returns to the 'Food Banks' tab and finds the same food bank that she previously ordered from. This food bank has a yellow box next to it and a 'Supply Level: Average' label. She clicks on the food bank name which prompts her to the next screen showing the available packages that can be delivered by the food bank. Unfortunately, the medium sized package is unavailable at this food bank and thus displays "Out of stock". Ronda could order the large package again as she meets the required number of people within her household to make this package available, but instead Ronda would rather select another food

bank nearby that has a medium package. Ronda clicks back on the top left of the screen and is back on the home tab which displays the list of food banks.

She clicks on the first food bank on the list which has a green box next to it. Ronda then selects the medium package from this food bank and the app prompts a confirmation message, “You have selected to receive a medium package from Town Foodbank?” followed by a ‘Receive package’ button beneath it. Ronda confirms her selection by clicking the button. The app notifies her via a pop-up message that the request has been sent to the food bank and to wait for confirmation of the order. Once the food bank has accepted the order, Ronda receives a confirmation email and an in-app notification of her package order confirmation.

7.0. Activity Diagram

This activity diagram in Figure 2 models’ scenario 1, which is based on use case 1. As such, it follows how a Donor would donate a package to a food bank. We already assume that the user is a Donor, so we do not select what type of user they are. Since it is only a Donor, the swim-lanes do not need to represent charity validation or any other user types.

The user firstly has the option to log in or register. If they register, they will get an email for verification, after which they will log in. Then the Donor sees the food bank map screen, where they could go into donation history. If not, they can choose their food bank, and the package they want to donate. The Donor can reselect their package at this point, by cancelling the confirmation. After confirming a package, they will process their payment. If there is not a successful payment, the Donor returns to select a package. Once they do have a successful payment, the system updates its database and sends the Donor a confirmation email.

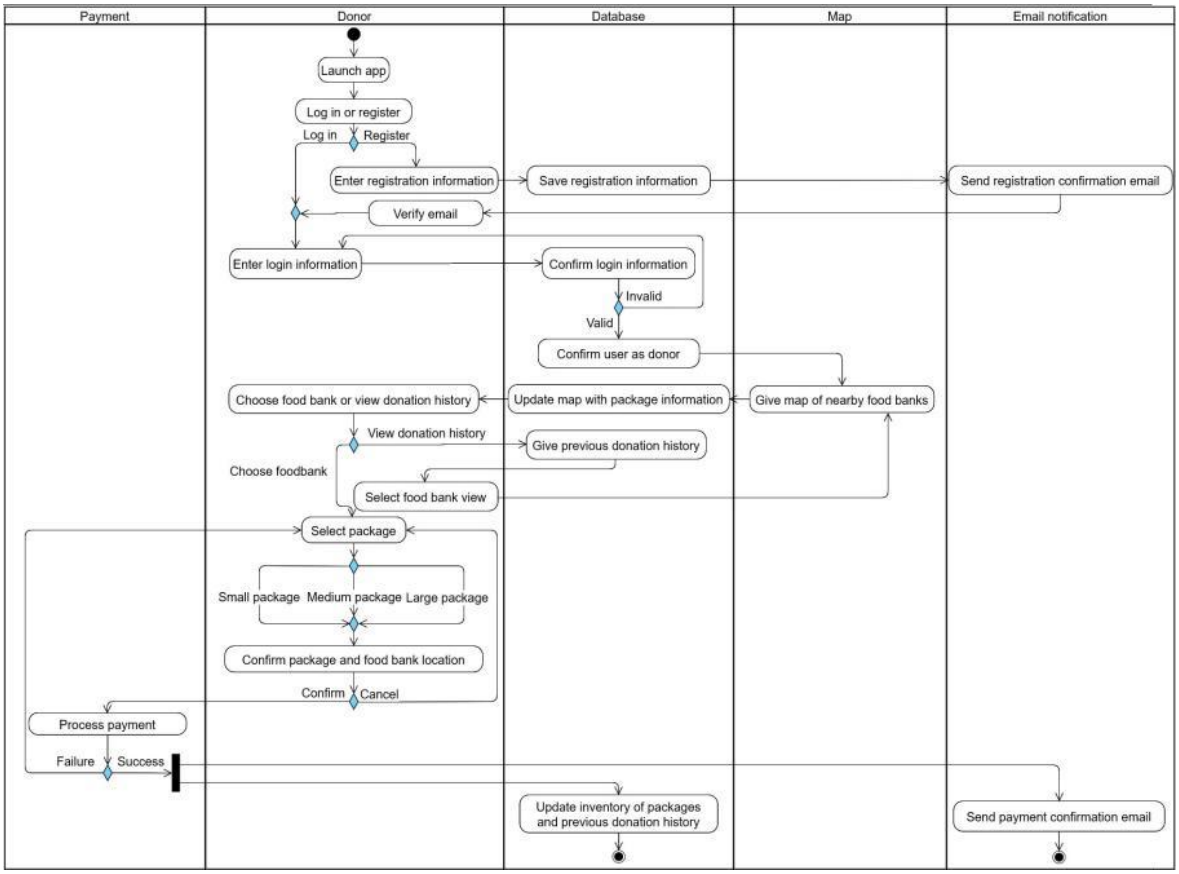


Figure 2: Activity Diagram of scenario 1

8.0. Class Analysis

This section undergoes a detailed noun-verb analysis and displays the class diagram for the system.

8.1. Noun-Verb Analysis

Below is the noun-verb analysis for the Feed My Food Bank system. While it doesn't provide an exhaustive list of classes and methods associated with the system, noun-verb analysis offers a solid starting point for translating the specification into implementable code. The requirements, use cases and scenarios previously detailed have been copied below, and significant words have been emboldened in accordance with the following table. What follows is a table showing the word/word-phrase of interest, the corresponding class/method name if it has been accepted, and the reason for this decision.

Part of speech	**Model Component**
Proper Noun	Instance (object)
Common Noun	Class (or attribute)
Doing Verb	Operation
Being Verb	Inheritance
Having Verb	Aggregation/Composition
Modal Verb	Constraint
Adjective	Helps identify an attribute

Results

Word Phrase	**Accepted**	**Reason**
Food Bank	FoodBank	Class
register	register()	Method of Account
input	updateSupplyLevel()	Method of FoodBank
packages	FoodPackage	Class
Supply levels	availableFoodPackages	Attribute of FoodBank
ordered	requestFoodPackage()	Method of Recipient class
user	Recipient	Class
updated	no	Duplicate of updateSupplyLevel()

Banking detail	no	Subsystem responsibility
Receive donations	no	Outside scope
information	PersonalDetails	Class
name	name	Attribute of PersonalDetails
address	address	Attribute of RecipientPersonalDetails
email	email	Attribute of Account
confirmation/updates	Confirmation	Class
number of people	numInHousehold	Attribute of RecipientPersonalDetails
Shown a list	displayLocalFoodBanks()	Method of PersonalAccount
Coloured ranking system	no	Abstraction of FoodBank attribute
sized	Size	Enumeration
request	FoodRequest	Class
confirm	confirmFoodRequest()	Method of FoodBank
donate	makeDonation()	Method of Donor
postcode	postcode	Attribute of DonorPersonalDetails
local area	no	Calculated by Location class
donation	Donation	Class
donation options	DonationOption	Class
selected	makeDonation()	Method of Donor
Taken to payment	makePayment()	Method of Donor
see how many packages he has donated	displayPreviousDonations()	Method of Donor class
has a shortage of supplies	availableFoodPackages	Aggregation - FoodPackage comprises FoodBank
list of packages	donationOptions	Attribute of FoodBank
notification pops us	displayConfirmation()	Method of Confirmation

options	no	Too general
login	login()	Method of Account class
personal information	PersonalDetails	Class
phone number	phoneNumber	Attribute of PersonalDetails
password	password	Attribute of Account
account	Account	Class
See a list	no	duplicate
deliver	no	Too general
coloured block	no	Abstraction of supply level
Supply level	displaySupplyLevel()	Method of FoodBank
Displays the list	no	duplicate
confirms	confirmRequest()	Method of FoodRequest
Request	FoodRequest	Class
Message is sent	sendConfirmationEmail()	Method of Confirmation

Using this noun/verb analysis, CRC cards were created to model potential classes. On these cards are the class name, an overview of their responsibilities and how they would collaborate with other classes in the system.

Account	
Holds basic functionality for accounts within the application such as login and logout.	

PersonalDetails	
Holds personal details for personal accounts.	

PersonalAccount	
Inherits from account, represents individual users i.e.Donor or Recipient, as opposed to FoodBank. Can display nearby food banks.	**PersonalDetails**

Donor	
Represents users that have registered as a donor. Can get a list of nearby food banks, make a donation and view previous donations.	**DonorPersonalDetails** **FoodBank** **Donation**

Recipient	
Represents users who are registered to receive food. Class can make new food requests and view previous activity.	**RecipientPersonalDetails** **FoodPackage** **FoodRequest**

FoodBank	
Represents a food bank that can accept or deny incoming requests for food and maintain stock level information.	**BankingInformation** **FoodPackage** **FoodRequest** **FoodBankDetails** **DonationOption**

DonorPersonalDetails	
Holds personal details for donors, specifically the postcode.	

FoodBankDetails	
Holds details for a food bank such as charity number, address and whether the bank has been confirmed by the charity subsystem.	**FoodBank** **BankingDetails** **Address**

RecipientPersonalDetails	
Holds personal details for recipients, specifically number of occupants in household and address.	**Address**

BankingInformation	
Holds banking details for a food bank.	

FoodRequest	
Represents an individual request for food. Sent to the food bank to be accepted/denied.	**Recipient** **FoodBank** **FoodPackage**

Address	
Holds an address	**Location**

Location	
Holds geographic coordinates. Can calculate distance between two locations.	

FoodPackage	
This class represents a food package and contains information about its size.	**FoodItem**

Donation	
Represents a donation. Holds functionality to make payment which generates a confirmation once processed.	**Donor** **Payment** **DonationConfirmation** **FoodBank**

DonationOption	
This class represents a donation package. Can be small, medium or large. Amounts set by food bank	**Size** **FoodBank** **Donation**

Confirmation	
This class represents a confirmation message that is sent to the owner of a personal account	

DonationConfirmation	
Inherits from Confirmation. Sent to the donor after donation has been confirmed.	**Donation** **Donor**

RequestConfirmation	
Inherits from Confirmation. Sent to the recipient after the food package has been confirmed.	**FoodPackage** **Recipient**

Size	
Enumeration that is used by both FoodPackage and DonationOption	

8.2. First-cut Class Diagram

These CRC cards in turn helped to produce the first cut class diagram (Figure 3), which offers a visual representation of the relationships between classes.

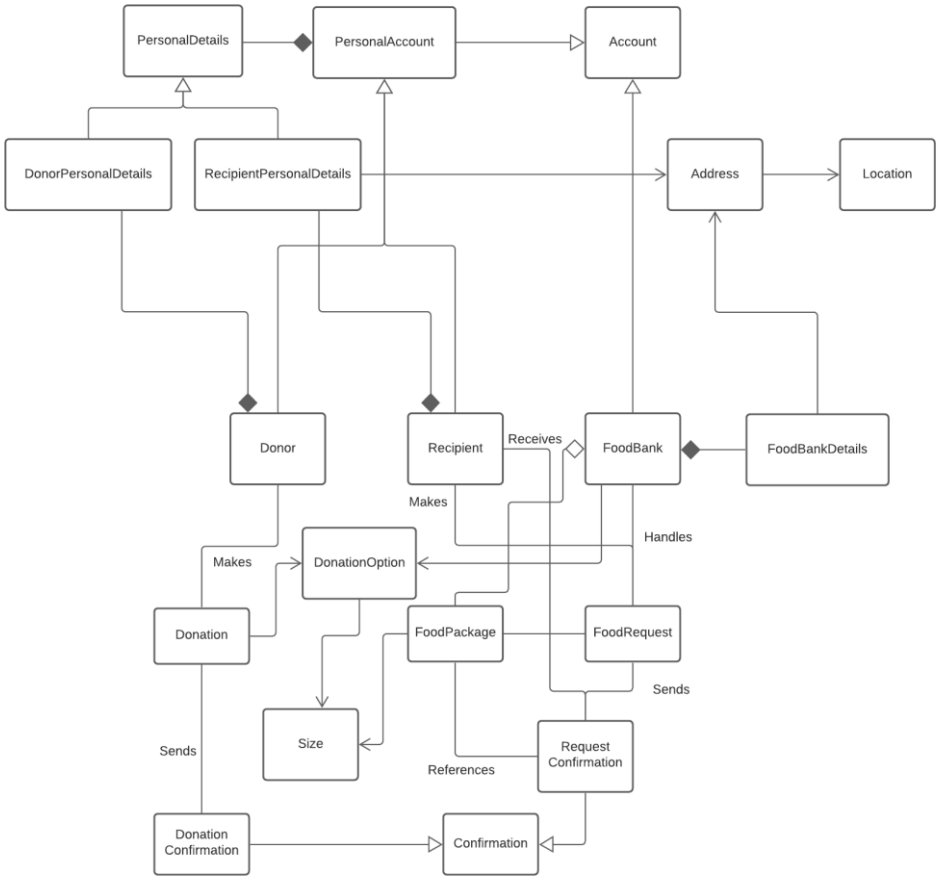


Figure 3: First cut class diagram

8.3. Detailed Class Diagram

The final step was to detail the contents of each class, showing the associated methods and attributes. This can be found on the next page. Standard getter and setter methods have been omitted to improve readability of the diagram (Figure 4).

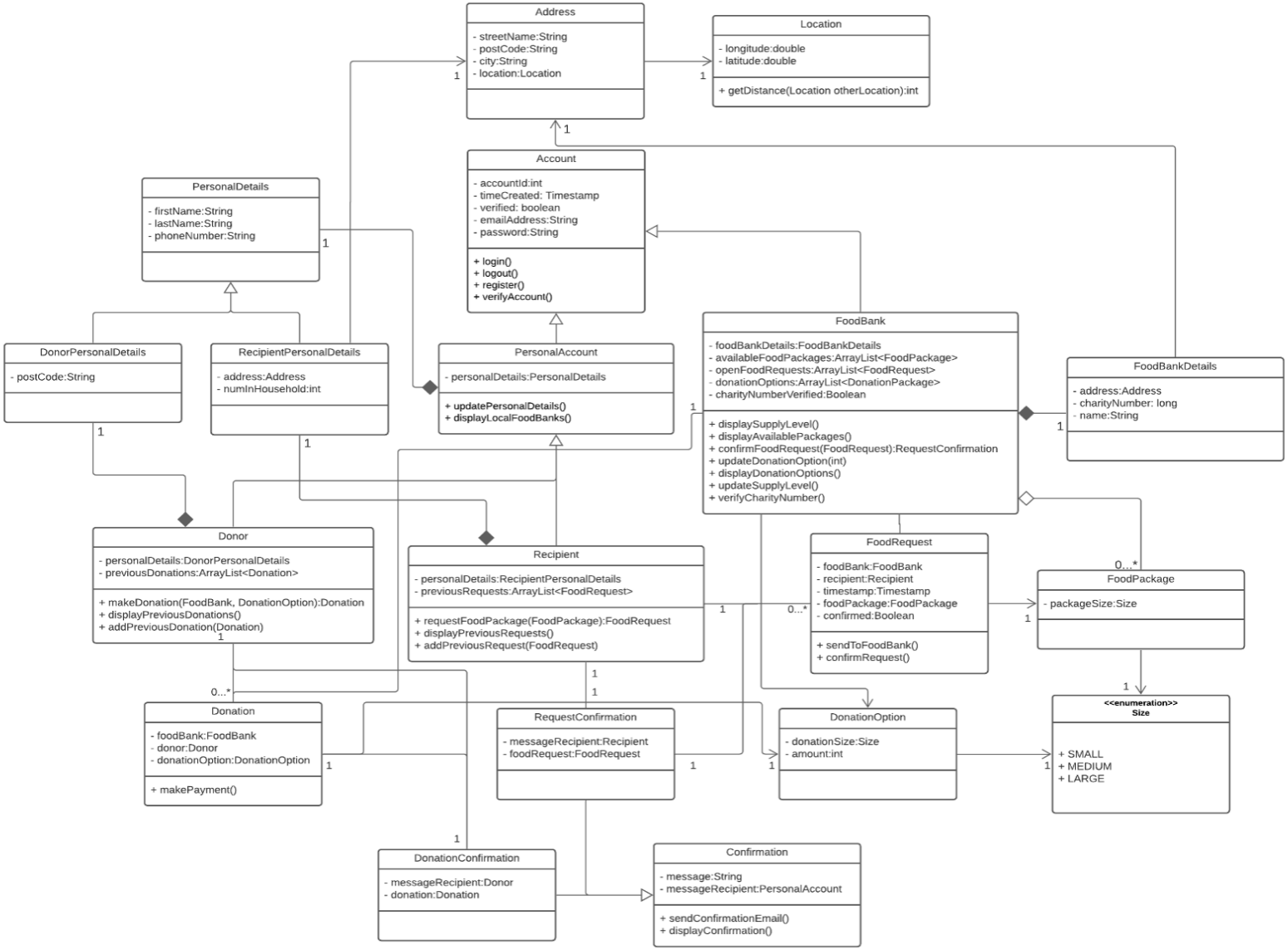


Figure 4: Detailed class diagram of the system

9.0. Object Diagram

This section shows the object diagram (Figure 5) of the system, considering the classes and objects that are used while going through a scenario of requesting a package.

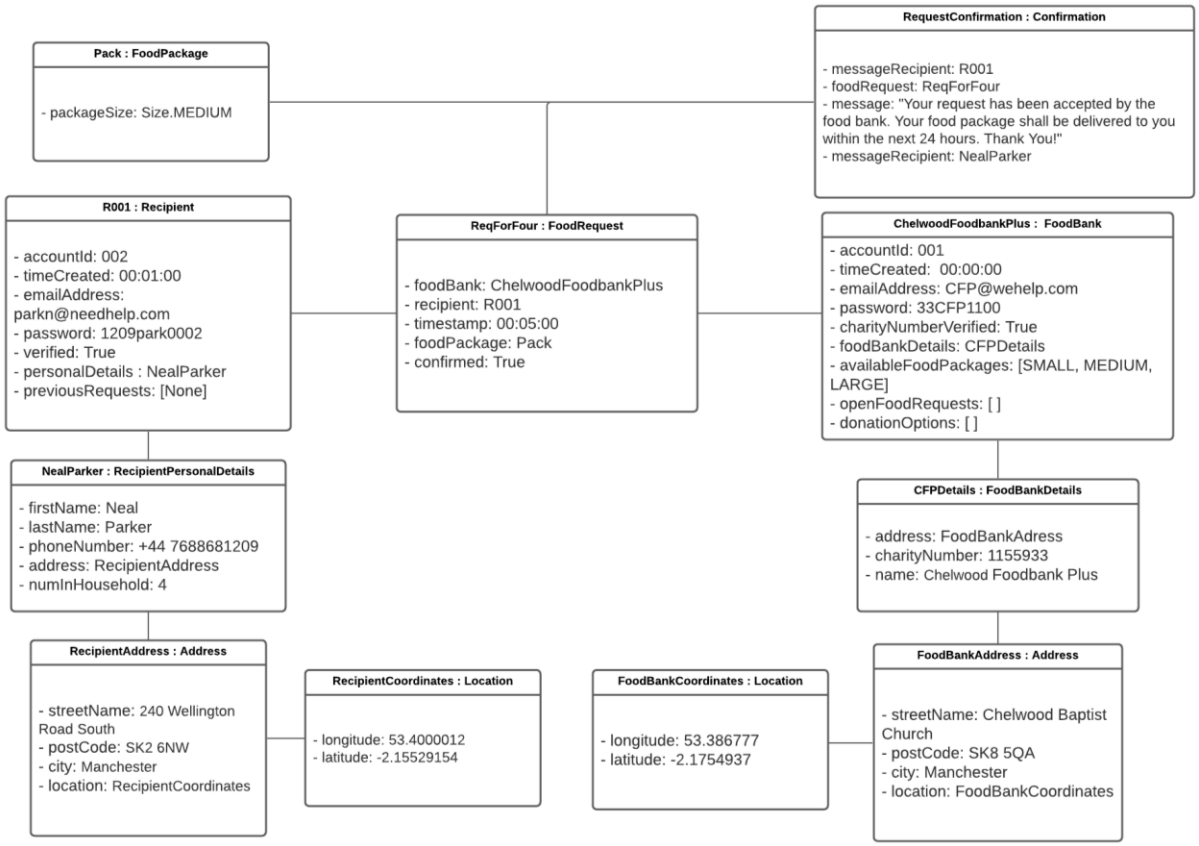


Figure 5: Object diagram of the system

This object diagram follows the class diagram closely representing all the objects and the methods that are used for scenario. The methods involved for:

a. Registering an account:
   - register()
   - verifyAccount()

b. Food Bank details:
   - displayLocalFoodBanks()
   - displaySupplyLevel()

c. Package details:
   - displayAvailablePackages()

d. Requesting a package:
   - requestFoodPackage(ReqForFour)
   - confirmFoodRequest(ReqForFour)

e. Confirmation:
   - sendConfirmationEmail()
   - displayConfirmation()

10.0. Sequence Diagram

This section details the interaction between objects to carry out an operation in a time sequence via a sequence diagram. It depicts the objects involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

10.1. Sequence Diagram for Scenario 1

The sequence diagram of scenario 1 (Figure 6) is for donating packages to the food bank. The diagram starts from the first time when the user enters the Feed My Food Bank app. The user is prompted to choose a food bank highlighting the level of supply, and then is presented with a selection of packages to donate. After payment, the Donor would receive a thank you message notification within the app and confirmation of the donation via an email.

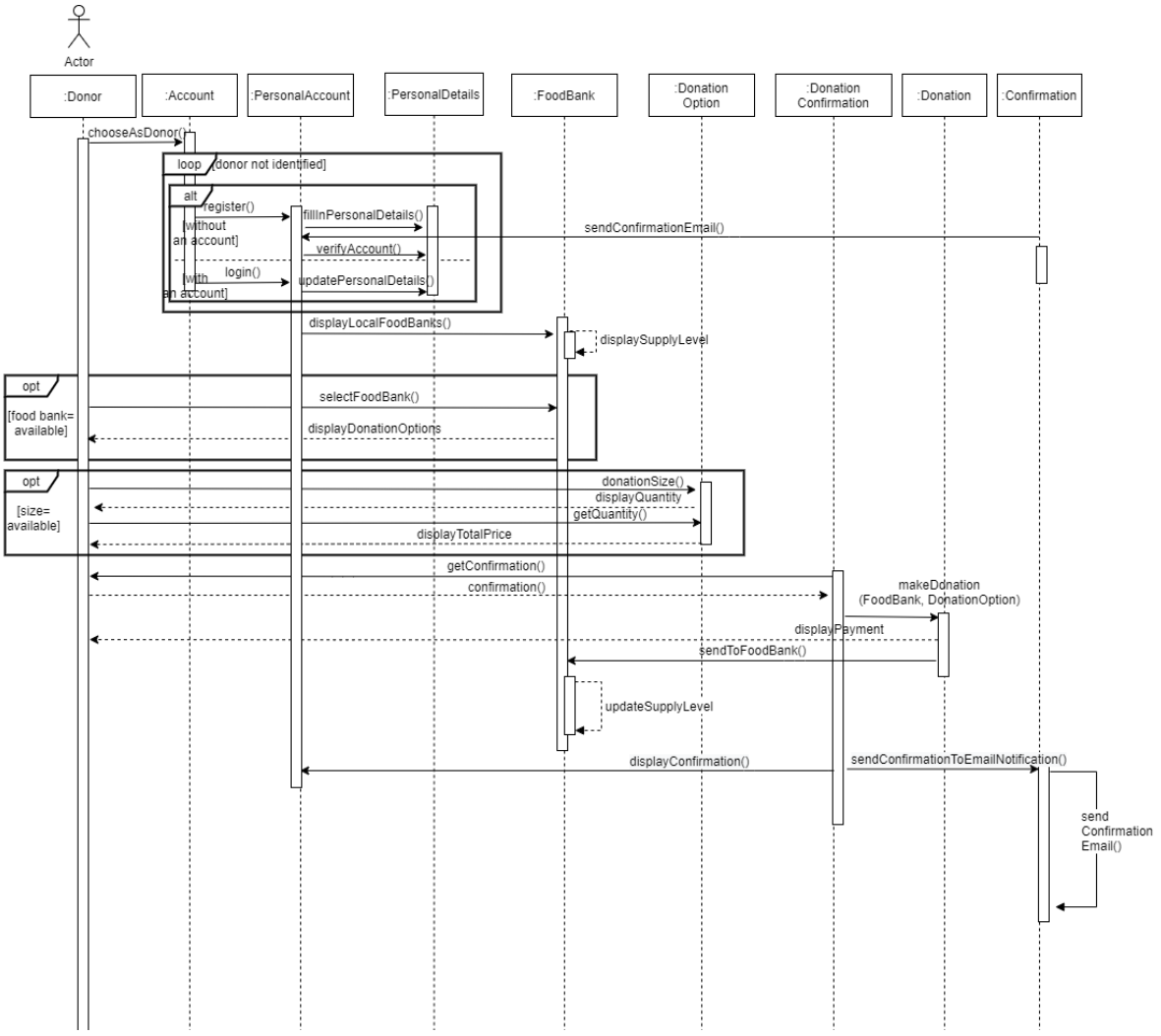


Figure 6: Sequence diagram of scenario 1

10.2. Sequence Diagram for Scenario 3

The sequence diagram of scenario 3 (Figure 7) is for a user ordering a package from a food bank. The diagram starts with the register for personal information. The interface then prompts the user to select from one of the nearby food banks and to choose from one of the available options of packages to select. App displays confirmation of selected package for the user. The order is then sent to the food bank to be confirmed. After the food bank accepts the user's request, a confirmation message is sent within the app as well as via email.

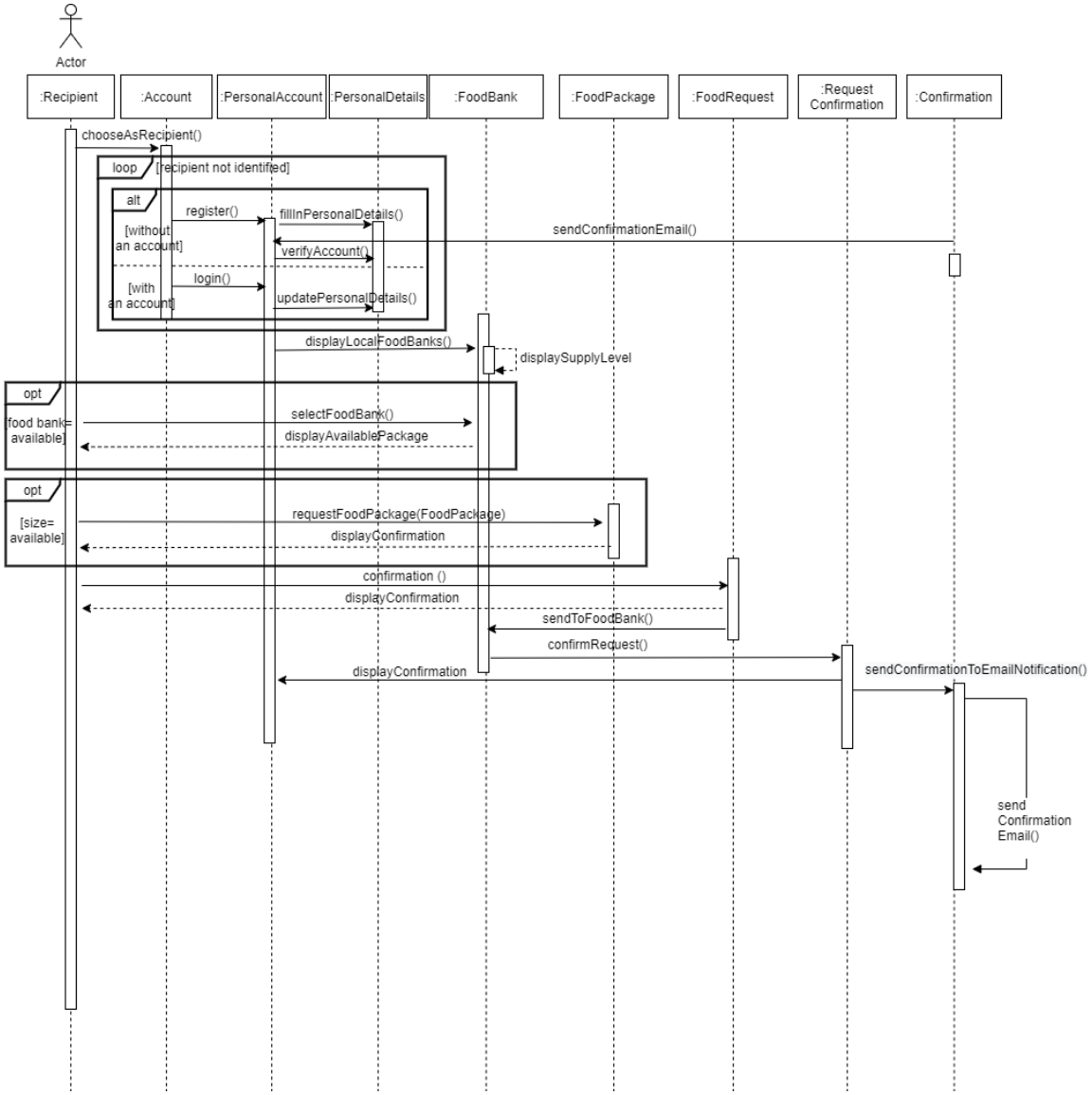


Figure 7: Sequence diagram of scenario 3

11.0. State Diagrams

The state diagram represents the state or the condition of the system at any given moment. Each diagram goes through a specific scenario and shows the state of the system and its transitions, following closely with the class diagram.

11.1. State Diagram 1

The system begins from the idle state until an event/action is caused. As the user is a Donor and has interacted with the application, the system will then lead to registration/login. In this case the user has yet to register, which lets the system transition from waiting to getting the user details. The user registers themselves and proceeds to verify their email.

After the registration process is successful the user then logs in to donate towards a package. The system waits for the credentials from the user and verifies it. Logging in, the user can now select the food bank of their choice (until which the system will be waiting for the user selection). Once a food bank is selected, the system will be waiting for the user to select the package. If a package is selected, the system then confirms with the user and then transits to the payment system. The system looks for the response from the payment system and then updates the database and halts.

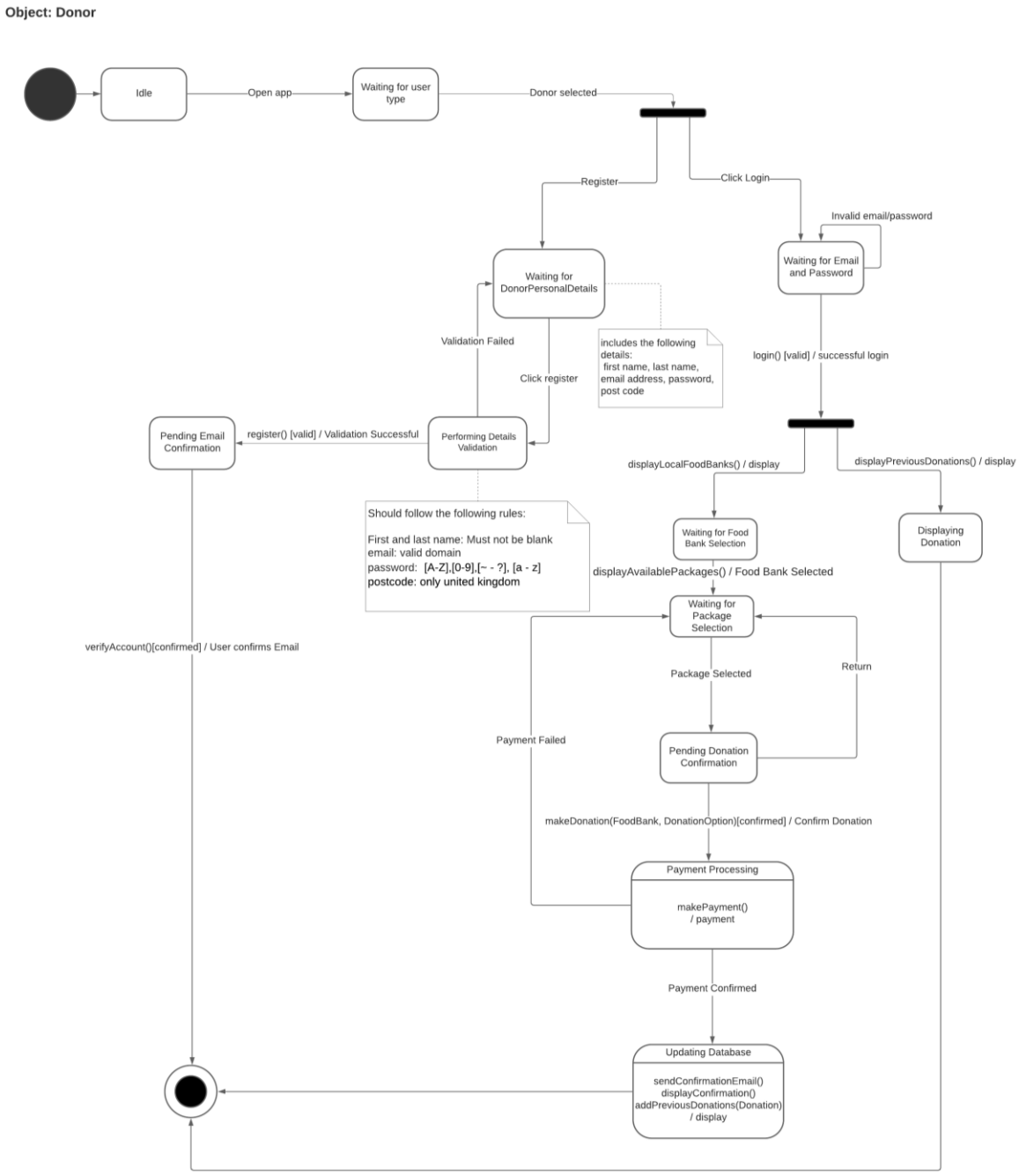


Figure 8: State diagram of scenario 1

11.2. State Diagram 2

The system follows the same process for the registering and logging for all the users with varying information. After the user has gone through registration and login themselves. The system will be waiting for the user to select the Food Bank, then transits to get the package from the user selection. Getting the package from the user, the system sends the requests to the Food Bank and looks for the response. If the package request has been accepted, the system then updates the database accordingly and halts.

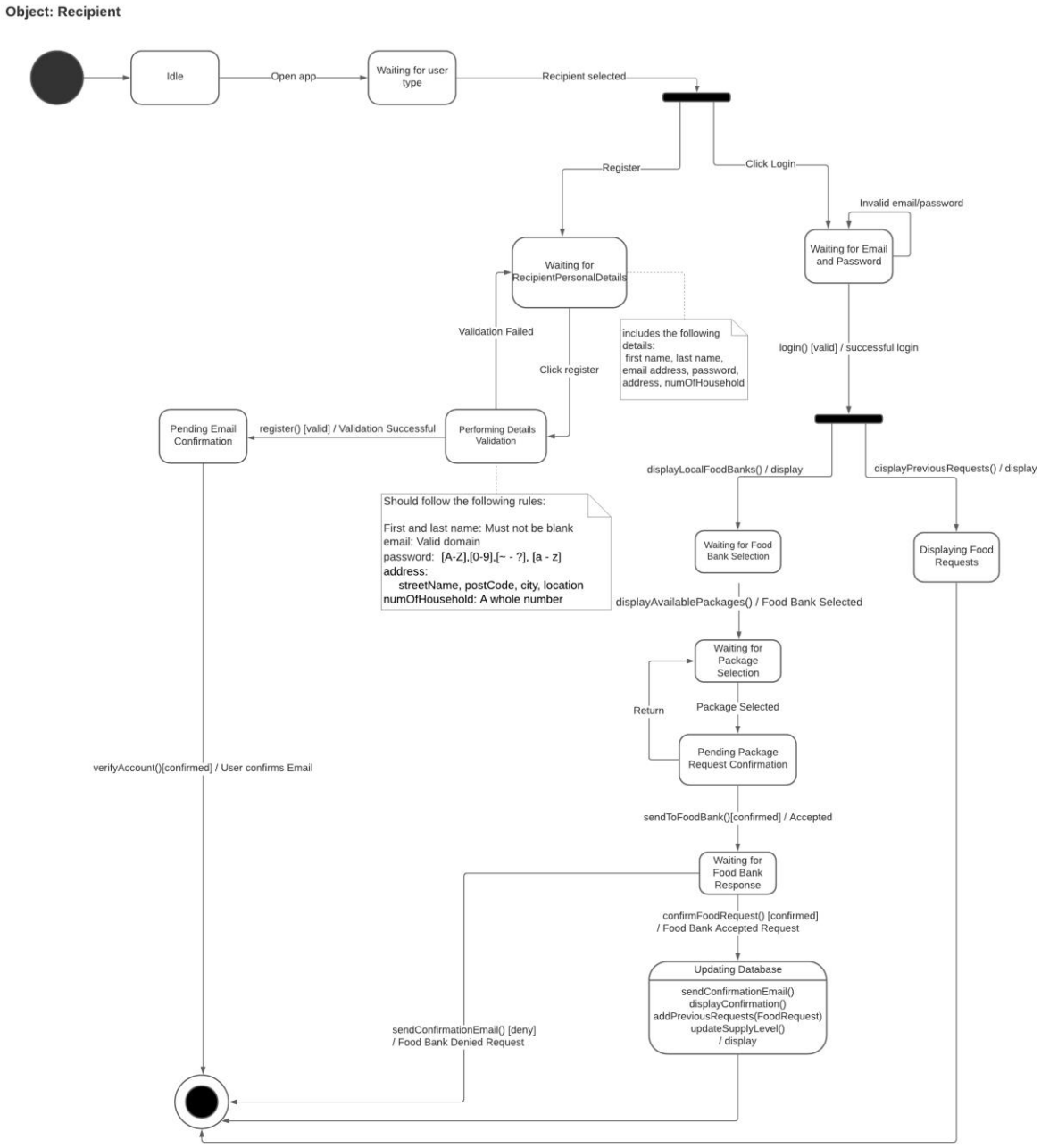


Figure 9: State diagram for scenario 3

12.0. Software Architecture

This section aims to detail the structure of the application with considerations to two architecture types: 3-Tier architecture and event driven architecture. Component and deployment diagrams are drawn for each architecture type to highlight the interactions between components within the system.

12.1. Component Diagram: 3-Tier Architecture

This a 3-tier layered architecture component diagram (Figure 10) shows how the different components would interact if split between a presentation layer, logic tier and a database tier. The presentation tier is the topmost level of the architecture, which displays information to the user. Logic tier coordinates the application and makes logical decisions. Database tier is where all the information is stored and retrieved. In the logic tier we use 3rd party components to handle some functionalities of the application. The 3rd party components are coloured in blue.

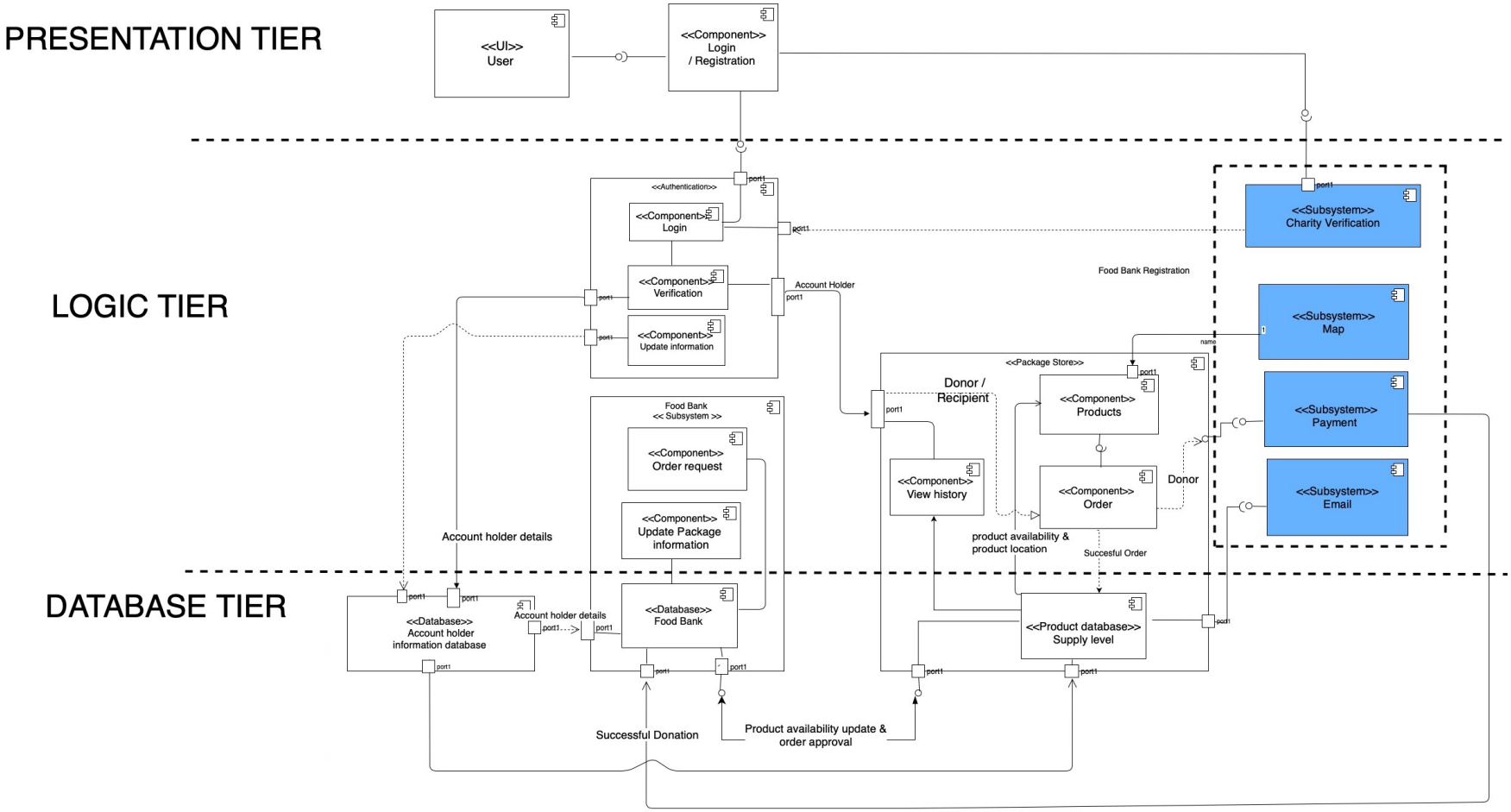


Figure 10: 3-tier architecture component diagram

12.2. Deployment Diagram: 3-Tier Architecture

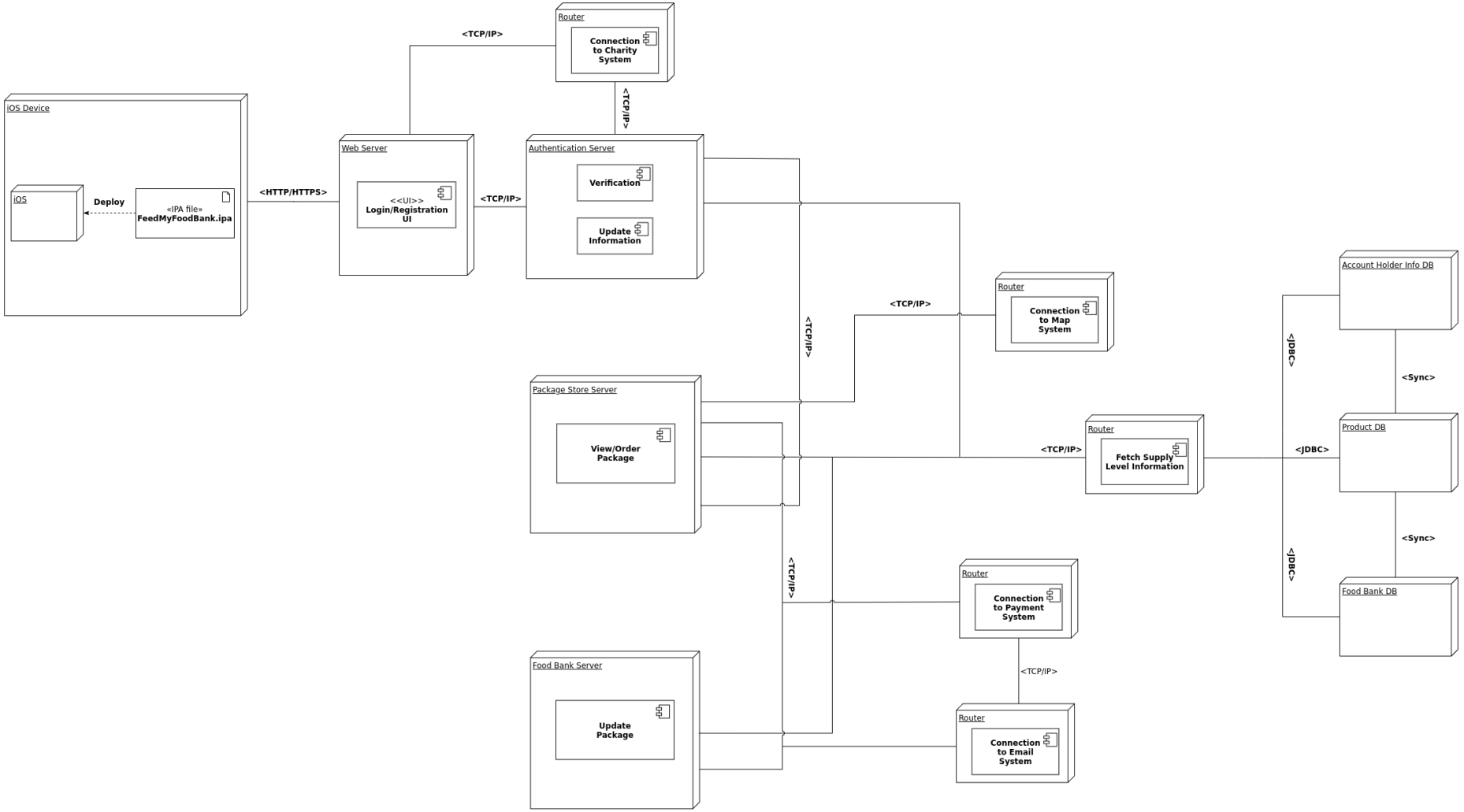


Figure 11: 3-tier architecture deployment diagram

This deployment diagram (Figure 11) is based on the 3-Tier Architecture (Component Diagram 2). This divides the system into three different tiers:

1. Presentation Tier
   - Application (in this case an iOS IPA app)
   - A User Interface
2. Logic Tier
   - This tier consists of the servers that coordinate the requests from the application, process the commands, and make necessary decisions.
3. Data Tier
   - This tier holds the Account Holder Information, Product Information, Food Bank Information in their respective databases.

For this system, as the user information has higher priority the data storage has been divided into different databases respectively. Having different databases may decrease the chance of system failure, protect the database from a compromised system/database and increase the stability of the system. The databases communicate with each other to sync the modifications, which will be helpful in having consistent flow of information. Only the required database will be accessed when specific information is required, which will not expose the other databases to the network, decreasing the attack surface.

12.3. Component Diagram: Event Driven Architecture

The event driven architecture relies on 3 main layers: event, event channel and event processing. This system utilises the mediator topology. The event is generated by the user through the user interface (UI). The event is passed through the queue to the event mediator (in this case the monitor component (Clark and Barn, 2011)) wherein it is passed onto the appropriate component to process the event. The process itself can also generate an event, such as updating the database, and this is channelled through to the database via the monitor. The charity verification is conducted within the security component. Each component conducts a single process. The components coloured in blue are 3rd party systems. Figure 12 below is a component diagram of the event driven architecture.

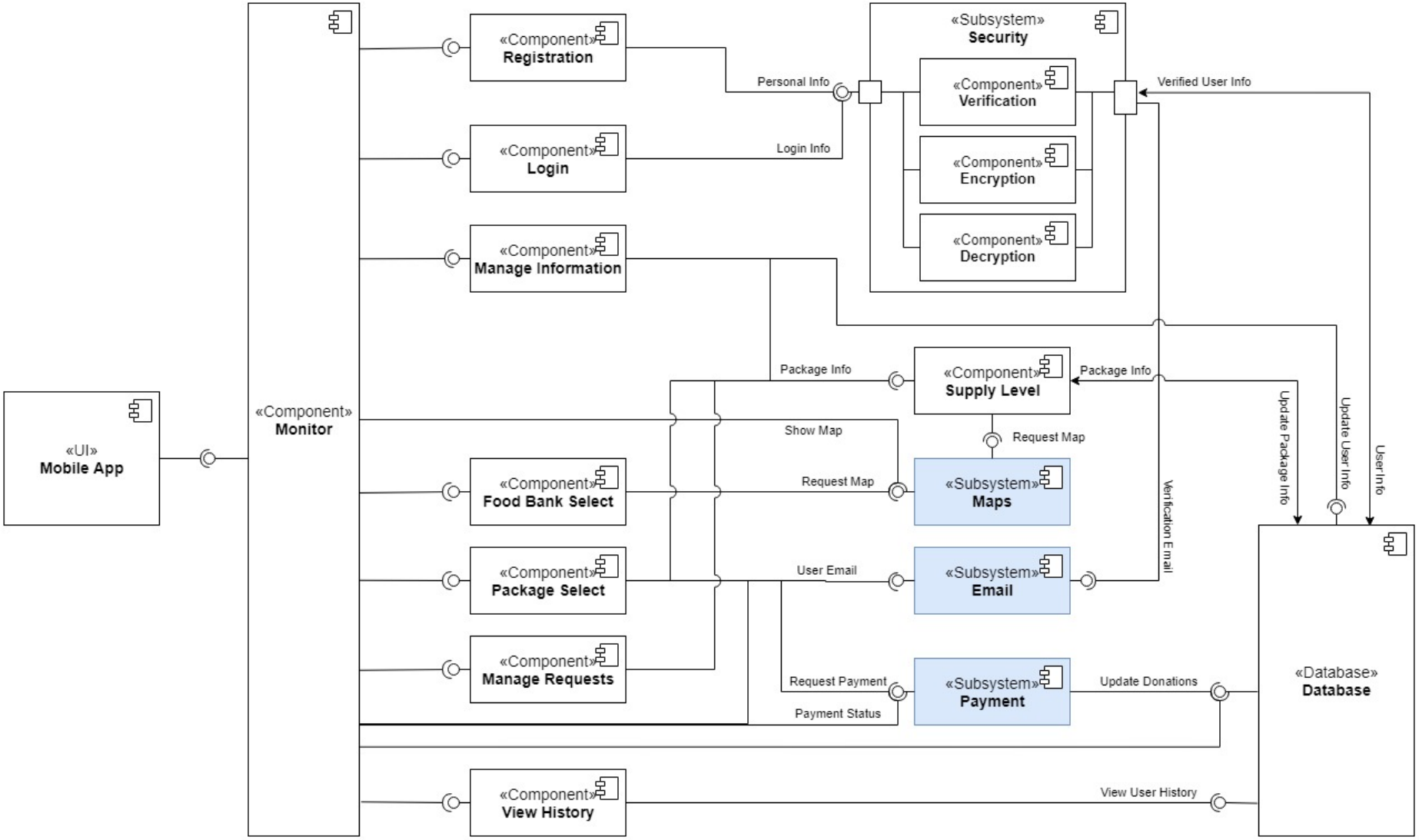


Figure 12: Event driven architecture component diagram

12.4. Deployment Diagram: Event Driven Architecture

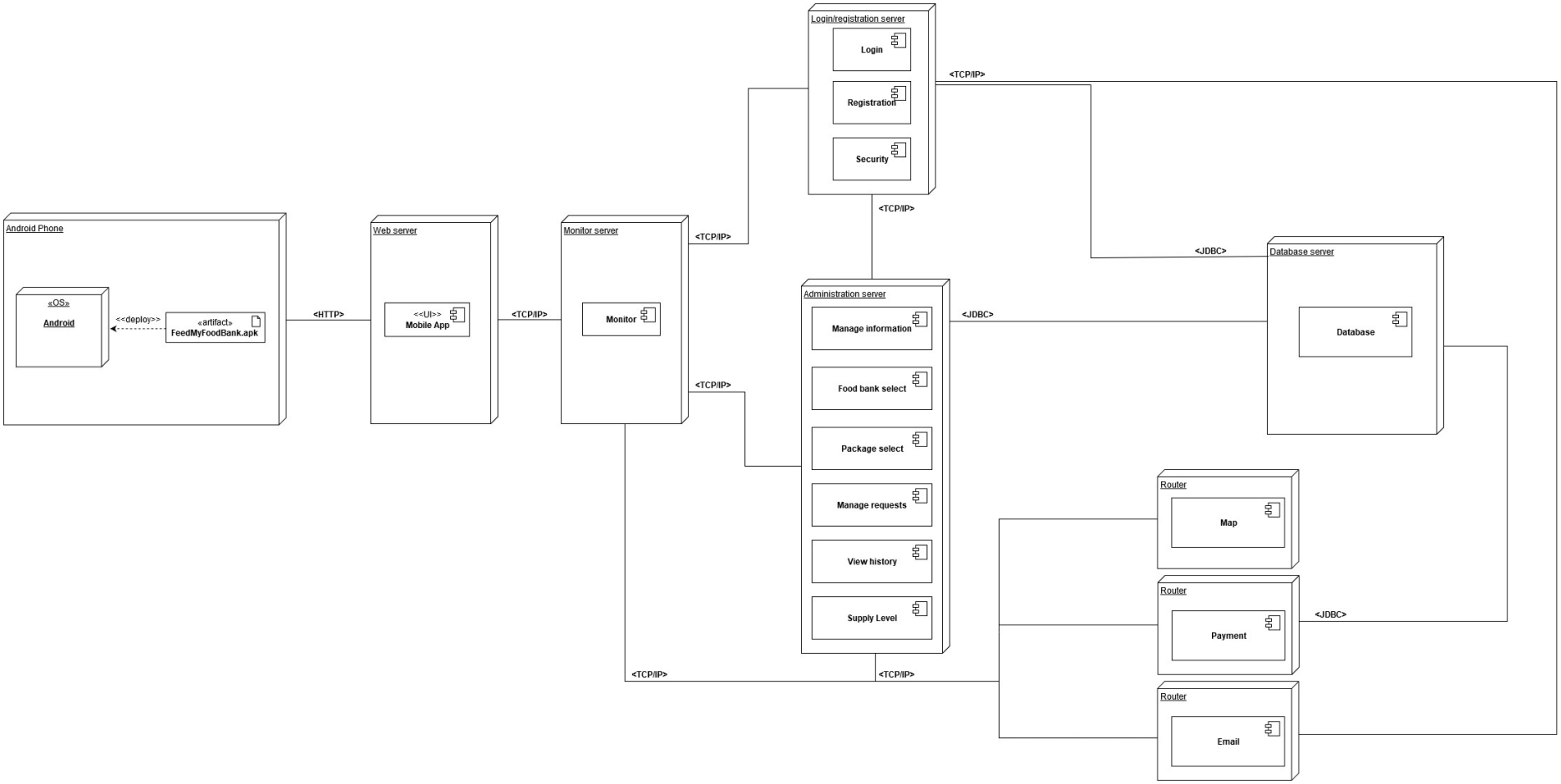


Figure 13: Event driven architecture deployment diagram

This deployment diagram (Figure 13) follows the first architecture, being event driven. We display this specific deployment on an Android phone. It divides the main architecture as follows:

- The monitor, forwarding events to servers
- The administration server, helping manage data and deal with the main functionalities of the app.
- The login/registration server, dealing with users logging into or registering for the app.
- The database holds all user, package and donation information
- Third-party applications

The Mobile App component supplies the events, sending them based on user behaviour on the app, which is passed to the monitor component in the monitor server. The monitor component acts as both the event controller and event bus. It organises and distributes the events to the correct nodes, as necessary. This includes the administration server, the login/registration server, and the third-party components.

12.5. Architecture Evaluation

The first architectural style that was considered was a 3-tier architecture (Richards, 2015). As the layered architecture is among the most widely known patterns, it was well suited for implementing this system. The greatest benefit of this pattern is its ease of implementation. Due to its popularity, there are many libraries and frameworks that can be used for the layer interfaces. Furthermore, owing to the high separation of concerns, code becomes easier to maintain and test.

On the other hand, if the layers are not clearly defined, the source code can become unintelligible. Furthermore, the fact that the layers are all strongly bound to one another means that the application can only be deployed as a whole. Even small changes will require the application to be redeployed in full

Secondly, we looked into using an event driven architecture (Clark and Barn, 2011). Event driven architectures are suitable for highly asynchronous applications, for example one that is sending and receiving a lot of data at variable time intervals. They work well with chaotic environments as the system is not flooded with irrelevant data; only the necessary components ever see data that needs to be processed. Moreover, the architecture is adaptable. It is easy to extend the system and add a new component or function without disrupting the rest of the application.

This being said, one concern associated with an event-driven architecture is the difficulty of testing if the components are tightly coupled. While individual components can be tested in isolation, the way they interact can only be tested when the system is up and running.

Ultimately, the decision to implement an event-driven architecture was taken for the Feed My Food Bank system. The nature of our application is highly asynchronous; the application centres around the Recipient requesting food, the Donor donating to a food bank or the Food Bank supplying packages to those in need. These events are very different in nature, so it is beneficial to make sure components are only handling data that is relevant to them. This contrasts with a layered architecture, where every layer would handle these events, requiring complicated code to process the data depending on the nature of the event. The system also has preference for the increased flexibility that an event-driven architecture offers. For example, if the components handling donations needed to be updated, only those components would need to be redeployed. Given the same situation for the layered architecture, the whole layer would need to be adjusted and the whole application redeployed, potentially resulting in downtime for the users.

13.0. Software Testing

The aim of this section is to outline the testing plan to be undertaken on the Feed My Food Bank app in order to meet the requirements of the client. This test plan highlights the potential bugs or issues that may arise. The system would be tested using seven functional requirements and three non-functional requirements. The objectives of these tests are to ensure our system conforms strictly to these requirements.

Test Items

Our system will be testing both the front-end of the application and the backend of the application. The system will be tested using both android and iOS platforms including Nougat, Oreo and Pie (android) and iOS 12, 13, and 14.

Features to be tested

Functional requirements to be tested:

1. Users should be able to register as one of the three following user types: Food Bank account, Recipient account, and Donor account. All users must enter a password with a minimum of 6 characters, and at least one uppercase, lowercase and special character.
2. Users should be able to login using their email and password.
3. Users can change their user information.
4. Users (Recipients) should be able to see available food banks within a 10-mile range from their location.
5. Users (food banks) should be able to provide and update their stock levels (packages) by changing the following parameters:
    a. Types of packages and corresponding availability
    b. Cost of package types
6. Users (Donors) should be able to select the food bank they want to donate to, the package they want to pay for, and should be then processed to the payment subsystem.
7. Users (food banks) should be able to see and then answer (deny/approve) requests made by Recipients.

Non-functional requirements to be tested:

8. The system must be able to access the database within 3 seconds.
9. The user is redirected within 10 seconds to the payment subsystem (PayPal).
10. The system should populate the map with food banks within 5 seconds.

Features not to be tested

Testing of the credit or debit cards will not be tested, due to the fact that we will need sensitive data which we do not have access to at this current moment. In addition to this, we do not have a sandbox account to test the data at the current moment. We will not test the operating systems on actual mobile phones but will be using emulators instead.

Approach

Our system engineers and quality testing team have both decided that it was best to approach our testing using both Black Box and White Box testing. Equally importantly, we believe it is incumbent that equivalent class partitioning and Boundary Value Analysis be implemented into our Black Box testing by using specific values within a range. For e.g. testing passwords outside of the allowed range of values [A-Z], [0-9], [~ - ?], [a - z]. With regards to White Box testing, we will create Control Flow Graphs to test the internal operations of the system. Even though the source code is necessary to write all the specific tests, we have set out to accomplish 100% branch coverage. Moreover, for each

class in the Detailed Class Diagram, we will create corresponding test classes that will later test the original classes and whether they accomplish the goals we've set out for them. For example the Account class will have a corresponding TestAccount class that will verify the methods (login(), logout(), register(), verifyAccount()), as well as the attributes (accountId, timeCreated, verified) that an object of this class should hold.

Test case ID	Test description	Test steps	Test data	Expected results	Actual result	Pass / fail	Test comms
T-1-1	Verify registration process: - try invalid password, - try valid password, - email confirmation	- Open the mobile app - Select user type - Create account with valid user information	User type: Donor, Name: Jon Doe, Email: jondoe@gmail.com, Password: CorrectPassword123! Postcode: SW1A 2AA	User should be able to create an account with a valid password and valid email, The user should receive an email with an account confirmation link. The email subsystem should return a success response.			
T-1-2	Verify registration process: -try valid name - try valid password - try invalid email -try valid street address, city and post code. -try valid number of household occupant	- Open the mobile app - Select user type - Create account with valid user information	User type: Recipient Name: John Doe Email: jondoe@g.com Password: password123 Street Address: 30 Castle Boulevard City: Bridgetown Postcode: SW1A 2AA Number of household occupants: 3	The user should not be able to create an account with a password that does not include upper- and lowercase alphanumeric characters, and special characters. Email subsystem should return error for invalid domain. The system should return invalid city for the address - must only contain Birmingham.			
T-1-3	Verify registration process: -try valid name - try valid email -try invalid password	- Open the mobile app - Select user type - Create account with valid user information	User type: Food Bank Name: Salvation Army Email: jondoe@g.com,	The user should not be able to create an account with a password that does not include upper- and lowercase alphanumeric characters, and special characters.			

	-try valid street address, city and post code. -try valid phone number -try valid charity number		Passwords: password123 Street Address: 30 Castle Boulevard City: Birmingham Postcode: SW1A 2AA Phone Number: 01233333 Charity number: 299	Email subsystem should return error for invalid domain. The system should return an error saying invalid phone number.			
T-2-1	Verify login functionality	-Open the mobile application and go to login page - Enter valid email and password	Email: testuser123@gmail.com , Password: TestPass21!	The user should be logged in with correct account details.			
T-2-2	Verify login functionality	-Open the mobile application and go to login page - Enter valid email and password	Email: testuser123@gmail.com , Password: TestPass21! *Device not connected to internet*	The user should be prompted to connect the device to the internet in order to check the database inside the web server for credentials.			
T-3-1	Test functionality allowing user to change their account information	-Open the mobile application -Log into an existing account - Go to user account - Change user name and save	Email: testuser123@gmail.com Password: TestPass21! New name: “Jon Doex”	User should see their name changed in the application and the detail should update in the database.			
T-3-2	Test if system recognizes incorrect details	-Open the mobile application -Log into an existing account	Email: testuser123@gmail.com Password: TestPass21!	User’s request to update details should be denied for invalid values entered in name field.			

		- Go to user account - Change user name and save	New names: “432“, “Joh/n”, “!John”				
T-4-1	Verify whether user can see available food banks	-Open mobile application and login as a Recipient - Go to food bank selection	User location: (3223, 522) food bank Location: (31, 53)	Users should see available food banks ONLY in their district. E.g. a user in Birmingham should not see food banks in Australia.			
T-4-2	Verify whether user gets correctly notified if their location services are switched off	-Open mobile application and login as a Recipient - Go to food bank selection	User email: validemail@gmail.com User: CorrectPassword123 User location: permission not granted / internet connection is lost	**no permission** Users should be notified that they need to change the settings accordingly. **lost connection** Users session should expire and they should be redirected to the login page.			
T-5-1	Test whether food banks are able to update the cost of their package, and supply levels.	-Open mobile application and login as a food bank. -Go to food bank information screen - Increase the cost of a small package by £4. - Increase supply level	Email: foodbankexample@gmail.com Password: TestPass21! Price increase: £4	The cost of the small package should now display a change in price. The price should update for all users and also in the database. The level of the food bank should also be reflected in the database.			
T-5-2	Verify if the system appropriately handles incorrect details in T-5 test	-Open mobile application and login as a food bank -Go to food bank information screen -Input price incorrectly -Save changes	Email: foodbankexample@gmail.com Password: TestPass21! Price: “ !@”, “ “, “412312343434234123132131231235443212 3”	An error message should appear for invalid values entered in price.			

T-6	Test Donor ability to make donation	-Open mobile application and login as a Donor -Select food bank from a list -Select package -Confirm you want to make the donation	Email: validdonoraccount@gamil.com Password: TestPass21!	User should be redirected to the payment subsystem			
T-7	Test if food banks can approve / deny requests made by Recipients	-Open mobile application and login as a food bank -Go to the section with pending requests -Approve request	Email: validfoodbank@gmail.com Password: TestPass21! Registered Donor request that is fetched from the database	Food banks should be able to approve the request from the screen. The request should change its status in the database from “pending” to “approved”. The Recipient should receive a confirmation email stating that their request has been approved.			
T-8	Check if the database sends a result back within 3 seconds from making the request	-Open mobile application -Input valid user details and click on login button	Email: validuser@gmail.com Password: TestPass21!	User’s status in the database should update to logged in within 3 seconds			
T-9	Test whether the Donor is redirected to the payment subsystem within 10 seconds from confirming	-Open mobile application and login as a Donor -Choose a food bank and what donation you want to make -Click on confirm	Email: validonor@gmail.com Password: TestPass21! Package: small Total donation: £10	The server should redirect the user to the payment subsystem within 10 seconds.			
T-10	Test whether the food bank map populates within 5 seconds	-Open mobile application - login	User Location: (31, 52)	User should see food banks in their area within 5 seconds from loading the page with food banks.			

		- Go to the page with nearby food banks					
T-10-2	Test whether the food bank map populates within 5 seconds	-Open mobile application - login - Go to the page with nearby food banks	User Location: permission denied	User should be prompted to allow GPS tracking in order to view food banks.			

Exit Criteria

The senior leaders of the team have decided that the testing would be concluded when 97% of the test cases have passed.

Assumptions

One of the assumptions is that all the test cases have been executed. Another assumption is that some test cases have extended parts. For example, T-1-2 means test case 1 part 2.

14.0. Prototypes

A prototype of screen captures (Figure 14) was developed in order to represent the Donor's view when using the Feed My Food Bank app. An interactive prototype was also developed using these images on Balsamiq and is shown in the Appendix. A video was also developed introducing the application.

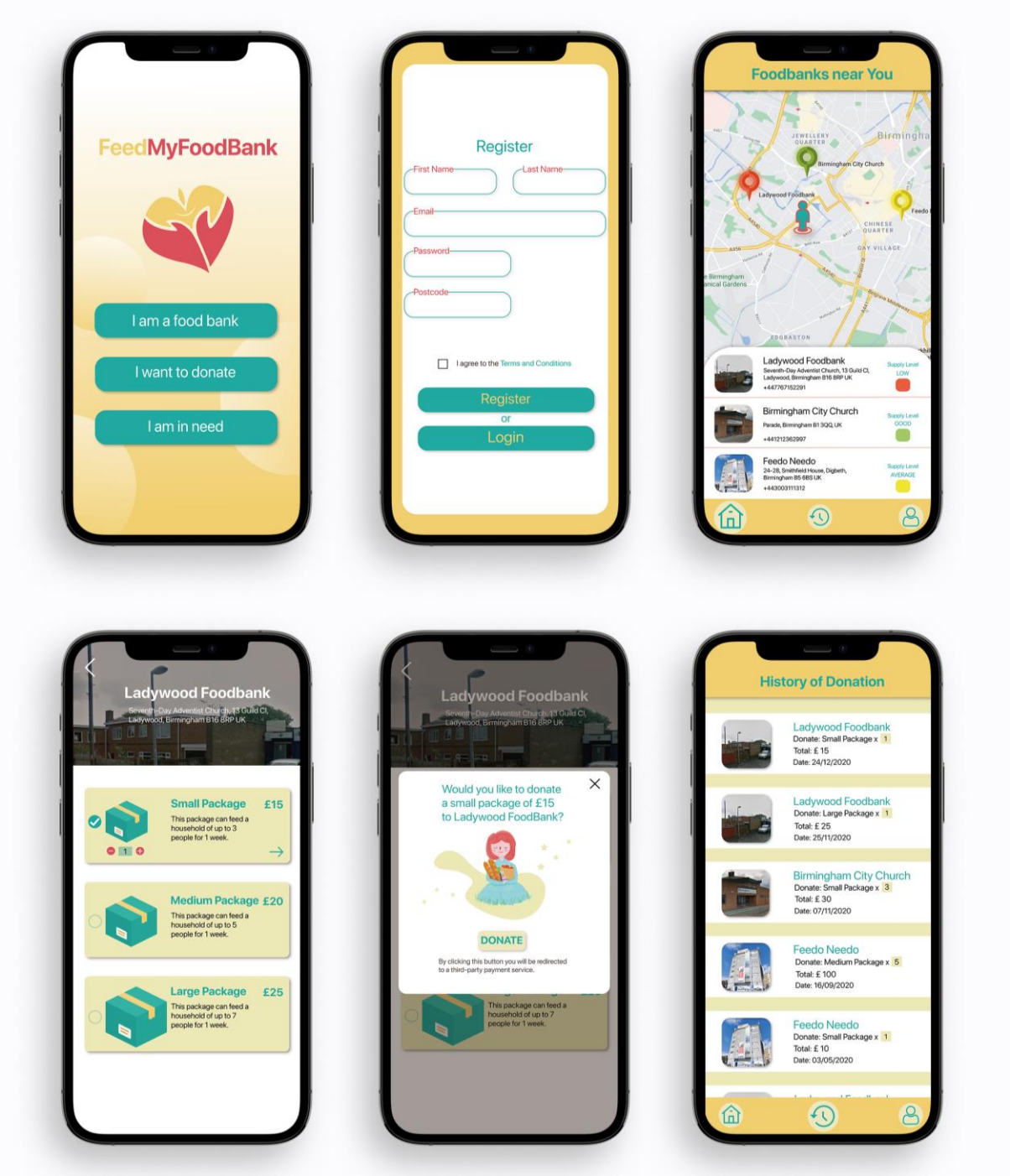


Figure 14: Prototype of the Feed My Food Bank app

15.0. Ethics and Professional Practice

Regular communication and open discussion created a smooth project workflow throughout this process. By being receptive and respectful towards diverse opinions, we created a working environment where people felt comfortable voicing their thoughts and had the freedom to work in areas suited to their individual skill sets. Throughout the development of Feed My Food Bank, all aspects have been evaluated with user needs at their core to ensure ease of use and functionality above all, in accordance with ACM section 2.1, Association for Computing Machinery (Gotterbarn *et al.*, 2018).

Protecting user information from breach should be a priority for every service, including MITM, Denial-of-Service (DoS), Injections and more. We have considered multiple data protection measures for our system. Data is encrypted and can only be decrypted with a specific key (Public key systems). Our database is divided to increase protection so that in a situation of compromise, not all data is at risk. We want to ensure that communication between the system and third-party entities is done through a secure channel by using protocols like “HTTPS''. In case of a breach or user information damage, the user would be notified immediately, and the situation rectified immediately, as stated in ACM section 1.2.

The ability of the system to guarantee the safety of Donor funds and personal assets is particularly important. Payment is processed by a third-party payment system; this data would not be accessible by our system or any food banks. This way, the theft of credit card information caused by multiple public transmission of credit card information on the Internet can be avoided, thereby protecting users and adhering to ACM section 1.6. Additionally, the third-party payment would use SMS password authentication as a double guarantee for each transaction. As for the email system, the user is required to enter their email during registration. An email is sent to confirm registration and is also used as a means of confirmation after completing a donation or package request procedure. Email address data is not used for any other commercial purpose. During the food bank validation process, an external ‘gov.uk’ database will be consulted to verify that the food bank is a legally registered charity. In accordance with ACM section 2.8, this is to ensure that the app is being used by legitimate parties for the safety of our users.

Furthermore, the Feed My Food Bank app operates in accordance with both GDPR and ethical guidelines surrounding location privacy. According to GDPR guidelines by ICO (ICO, 2019), issues arise around location data when the service both tracks and stores a user’s information without their consent or does so unnecessarily. To maintain the safety and privacy of users, our map system does not use GPS to track user location. Instead, the system uses the postcode entered by the user to display food banks within the user’s local area. With regards to package recipients, their address is stored on a database to enable home delivery, which employs the use of encryption to ensure the security of their personal data. Based on upholding ethical code 1.3, the data will not be shared with any other third-party entities. Any protection method/principle that may cause inappropriate action will be not implemented (sections 2.8, 2.9).

References

Butler, P. (2020a) *Extreme Poverty 'Will Double By Christmas' In UK Because Of Covid-19*, *The Guardian*.

Butler, P. (2020b) *UK Food Banks Face Record Demand In Coronavirus Crisis*, *The Guardian*. Available at: https://www.theguardian.com/society/2020/may/01/uk-food-banks-face-record-demand-in-coronavirus-crisis (Accessed: 17 December 2020).

Clark, T. and Barn, B. S. (2011) 'Event driven architecture modelling and simulation', in *Proceedings of 2011 IEEE 6th International Symposium on Service Oriented System (SOSE)*, pp. 43–54. doi: 10.1109/SOSE.2011.6139091.

Gotterbarn, D. *et al.* (2018) *ACM Code of Ethics and Professional Conduct*. Available at: http://hdl.handle.net/2086/16422%0A.

ICO (2019) *Guide to the General Data Protection Regulation (GDPR)*, *Guide to the General Data Protection Regulation*. Available at: https://ico.org.uk/for-organisations/guide-to-the-general-data-protection-regulation-gdpr/ (Accessed: 17 December 2020).

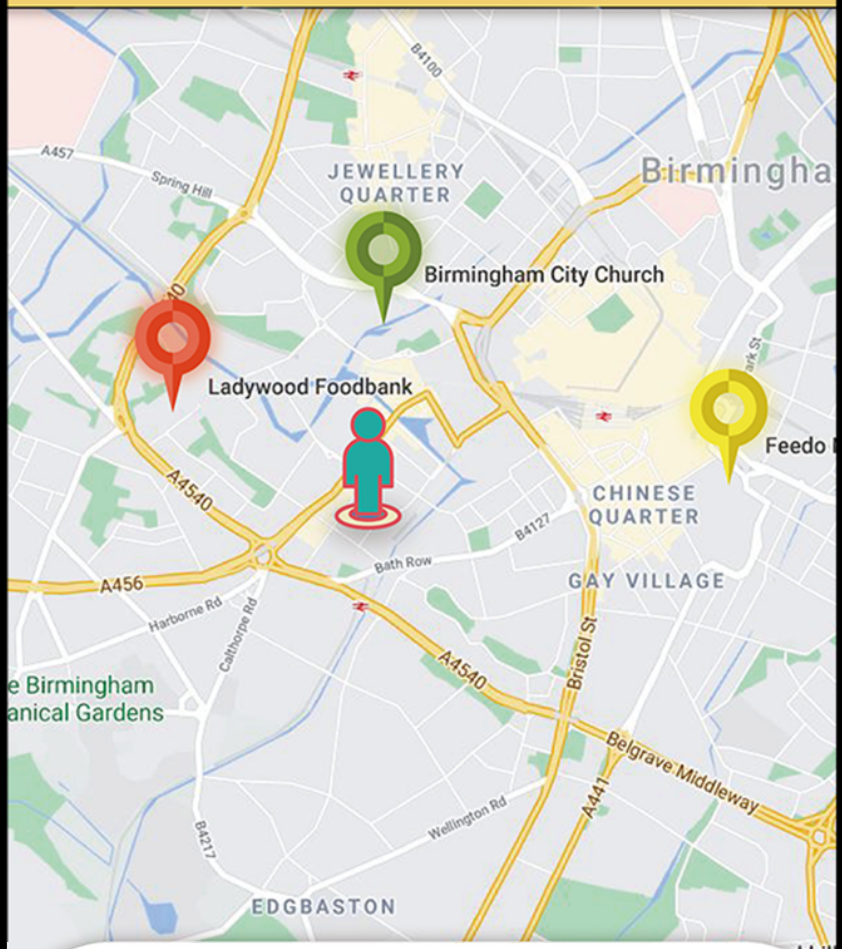
Richards, M. (2015) *Software Architecture Patterns*. O'Reilly Media, Inc.

Trust, T. T. (2020a) *LOCKDOWN, LIFELINES AND THE LONG HAUL AHEAD: The Impact Of Covid- 19 On Food Banks In The Trussell Trust Network*. Available at: https://www.trusselltrust.org/wp-content/uploads/sites/2/2020/09/the-impact-of-covid-19-on-food-banks-report.pdf (Accessed: 17 December 2020).

Trust, T. T. (2020b) *New Report Reveals How Coronavirus Has Affected Food Bank Use - The Trussell Trust.* Available at: https://www.tusselltrust.org/2020/09/14/new-report-reveals-how-coronavirus-has-affected-food-bank-use/ (Accessed: 17 December 2020).

Appendix: Interactive Prototype

Foodbanks near You



Ladywood Foodbank
Seventh-Day Adventist Church, 13 Guild Cl, Ladywood, Birmingham B16 8RP UK
+447767152291

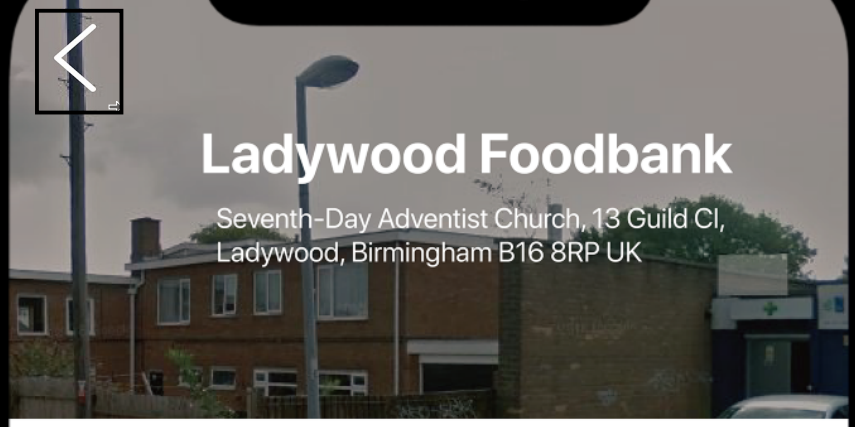
Supply Level
LOW

Birmingham City Church
Parade, Birmingham B1 3QQ, UK
+441212362997

Supply Level
GOOD

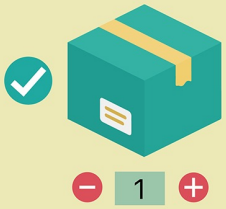
Feedo Needo
24-28, Smithfield House, Digbeth, Birmingham B5 6BS UK
+443003111312

Supply Level
AVERAGE



Ladywood Foodbank

Seventh-Day Adventist Church, 13 Guild Cl,
Ladywood, Birmingham B16 8RP UK



Small Package £15

This package can feed a household of up to 3 people for 1 week.

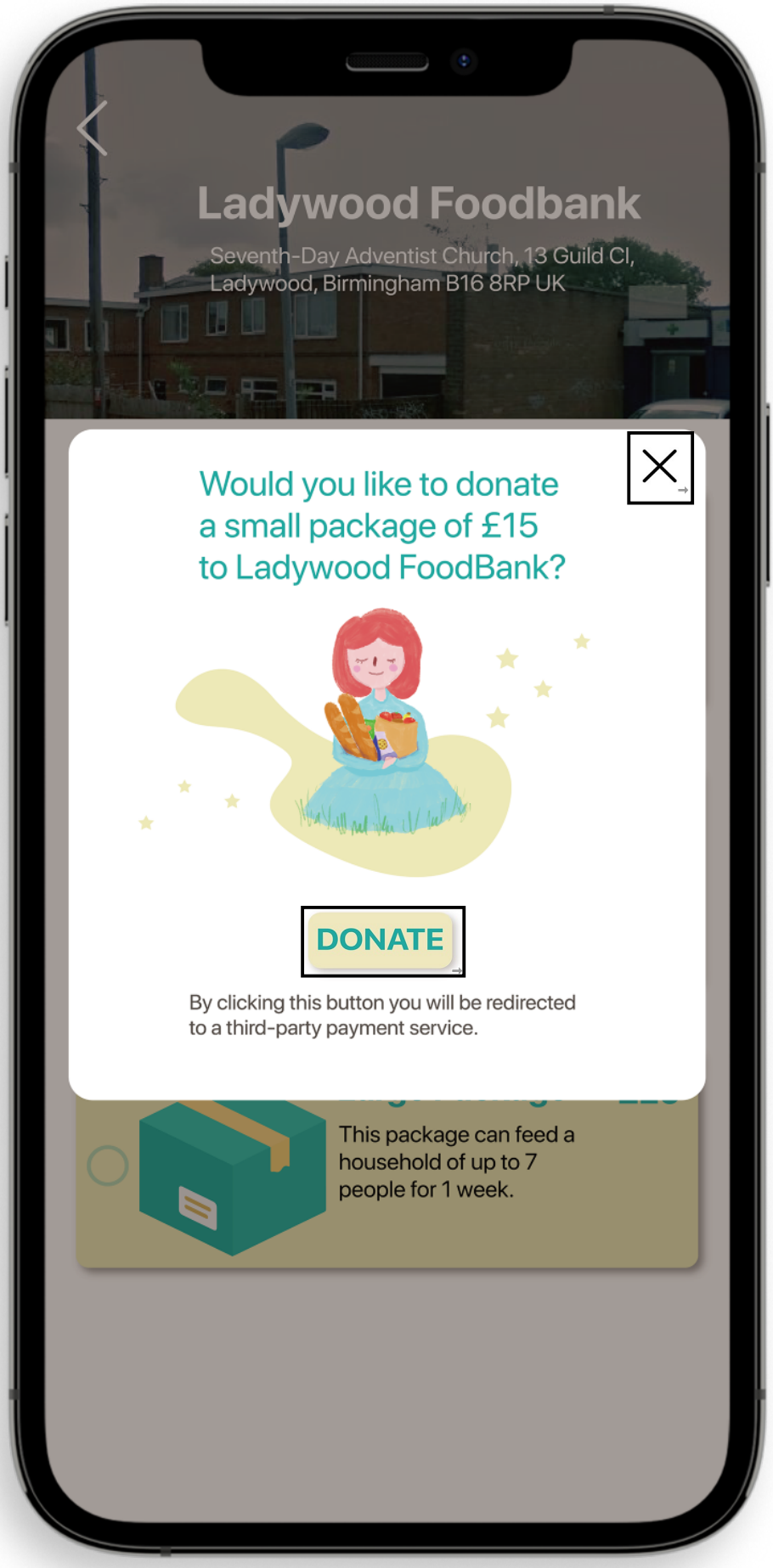
1

Medium Package £20

This package can feed a household of up to 5 people for 1 week.

Large Package £25

This package can feed a household of up to 7 people for 1 week.


Ladywood Foodbank
Seventh-Day Adventist Church, 13 Guild Cl,
Ladywood, Birmingham B16 8RP UK
Would you like to donate
a small package of £15
to Ladywood FoodBank?
DONATE
By clicking this button you will be redirected
to a third-party payment service.
This package can feed a
household of up to 7
people for 1 week.

History of Donation

Ladywood Foodbank

Donate: Small Package x 1

Total: £ 15

Date: 24/12/2020

Ladywood Foodbank

Donate: Large Package x 1

Total: £ 25

Date: 25/11/2020

Birmingham City Church

Donate: Small Package x 3

Total: £ 30

Date: 07/11/2020

Feedo Needo

Donate: Medium Package x 5

Total: £ 100

Date: 16/09/2020

Feedo Needo

Donate: Small Package x 1

Total: £ 10

Date: 03/05/2020