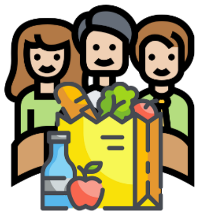
Software Engineering and Professional Practice

Building Useable Systems

University of Birmingham

at Your Step



Mobile App

CWK Group 49

Rosie Platten - 1877897

Doris Riou - 1662829

Gregorius Putra - 1923836

Christopher Rock - 2241822

Yuanhu Tao - 2219479

Ryan Salisbury - 2020085

William Russ - 2231955

Sebastian Gabriel Savu - 2173620

2020 - 2021

Table of Contents

Assumptions

The following are some assumptions we made about the at Your Step application and situation:

- All groceries pulled through the Supermarket's API are available at any time.
- There is at least a decent amount of volunteers out to deliver for the number of customers ordering.

A. Requirements Engineering

1. System Description

Our system is a mobile application that provides a matching service between people self-isolating as a result of a positive COVID test or needing to shield because of a health condition/old age, and volunteers who are willing to provide support by buying groceries and completing a contact free delivery to the self-isolating or vulnerable person. Since supermarket home delivery services have been incredibly busy during the lockdown period and in many cases people have struggled to get delivery slots, this service will allow volunteers to help ensure that vulnerable and self-isolating people are able to get the food they need.

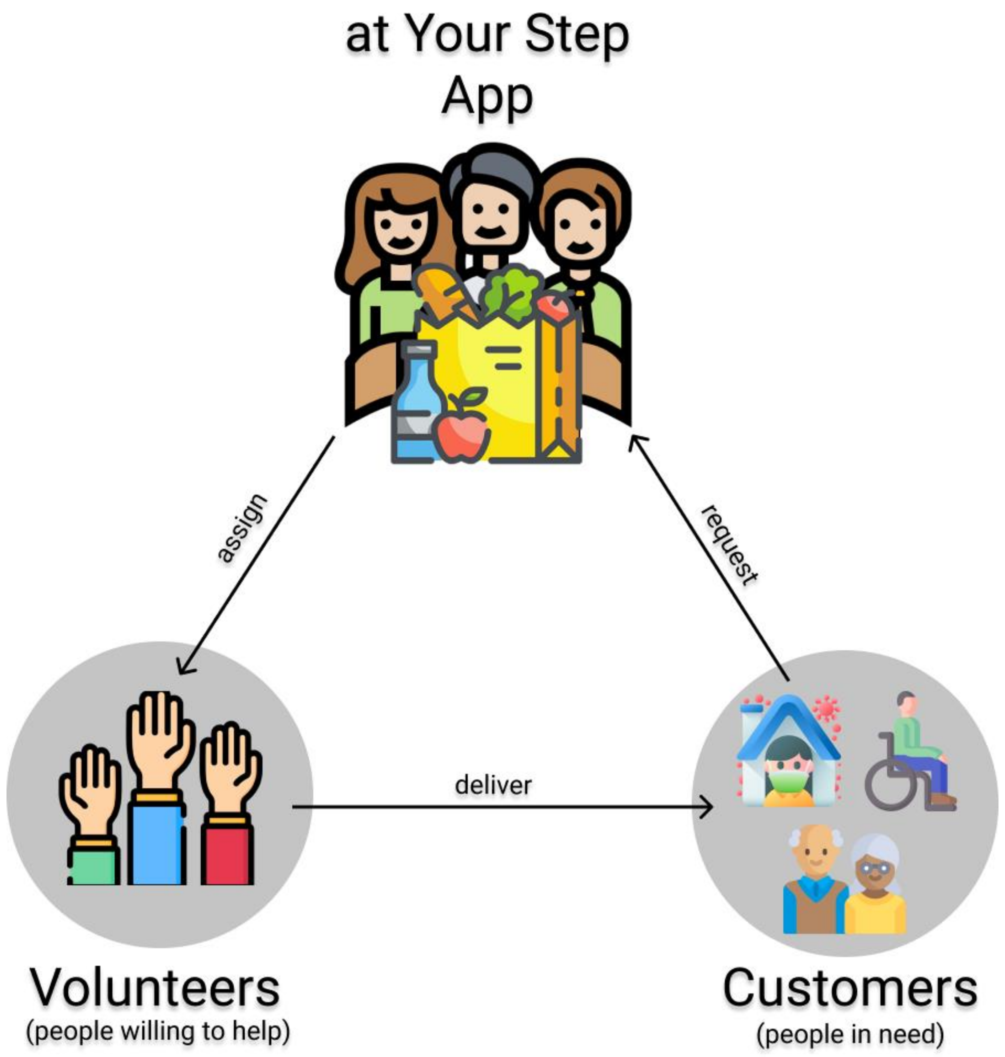
The system will allow the user to create an account holding basic contact details (first name, last name, email address, password, phone number). After creating an account, the system will prompt every user to link a PayPal account for pre-approved payments and to provide their address. The user can choose to either be a customer willing to purchase groceries or a volunteer that wishes to deliver a customer's order. Users will be able to update their account details at any time and swap between the two available account modes: volunteer and customer. The account will only be able to be accessed with a valid email address and password pair.

If the user identifies themselves as a customer, they will be able to create a grocery order by selecting a supermarket chain and a time slot in which they would like their order to be delivered. The user will then be able to select from a list of products available in that respective store and create their desired shopping list. They will then be able to submit the order and the system will display the details of this order (the shopping list, time slot, address for delivery, and supermarket chain). The user will then have the option to confirm or deny the details. If the user denies, they will be directed back to modifying their current order. If the user confirms, the order will be added to a list of active orders within the system from which it will be accepted by a volunteer.

After placing an order, money is put on hold from the user's PayPal account and they will be able to view the current status together with a summary of the order's details. If accepted by a volunteer, the customer will be able to track the active order until it is delivered. After the order has been delivered the customer is prompted to review the volunteer's services through a five-star rating pop-up and a text box describing their experience. If the user skips this interaction, they can comeback at any time and review the volunteer's services by accessing their order history which holds a list of all their past orders.

If the user identifies themselves as a volunteer, they will be able to view a list of active orders in the area, calculated by the system using the volunteer's location and the preferred maximum distance set. They will be able to see a short summary of each order in the list with relevant information which can be sorted to their preference, thus, helping them choose which order they wish to fulfill at free will. If they choose to accept an order, they will be shown a path to the customer via Google Maps together with the shopping list and all available stores (chosen by the customer) in the area. Once they have delivered the groceries, they will submit a photograph of the receipt and the system will transfer the money on hold from the customer to the volunteer. After each completed order the volunteer's review score is calculated and updated by the system (if the customer leaves a review) to reflect an average of their completed services.

The system will send relevant emails to both customers and volunteers when an important event occurs such as: order has been delivered, there is an issue with the order, account is created etc. Both volunteers and customers will be able to open tickets to the system admin regarding any problem they might encounter while using the system.



2. Functional and Non-Functional Requirements

1. Account Registration

1.1. The system must allow a User to register an account.

1.1.1. The system must ensure that the User enters a first name.

1.1.2. The system must ensure that the User enters a last name.

1.1.3. The system must ensure that the User enters a valid email.

1.1.3.1. The system must not allow registration with an email that has been registered with, to be used for the registration of a new account.

1.1.4. The system must ensure that the User enters a valid password.

1.1.4.1. The system must ensure that the User enters between 8 and 50 characters, at least one uppercase letter, at least one lowercase letter, and at least one numerical digit.

1.1.4.2. The system must reject the registration attempt if a password that does not satisfy all the pre-requisites is entered and display a message of what pre-requisite the password is missing.

1.1.5. The system must ensure that the User enters a valid phone number.

1.1.5.1. The system must only accept a UK phone number.

1.1.5.2. The system must reject the registration attempt if a non-UK phone number is entered and display an incorrect phone number message.

1.1.5.3. The system must reject the registration attempt if a UK phone number that has already been associated with an account on the accounts database is entered in this field and used to register a new account.

1.1.6. The system must ensure that the User enters a valid address.

1.1.6.1. The system must verify that the address is a UK-based address through Google's Maps API.

1.1.6.2. The system must reject the registration attempt if a non-UK based address is entered, and display a message saying that the application is only in service within the UK.

1.1.7. The system must ensure that the User enters a valid form of making and receiving payments.

1.1.7.1. The system must transfer the User to the Payflow (PayPal) payment gateway to enter the details.

1.1.7.2. The system must then set the financial details of their account with their PayPal account.

1.2. Upon satisfaction of every pre-requisite, the system must add the new registered account into the accounts database.

1.3. Upon satisfaction of every pre-requisite, the system must display a successful account registration message and direct the User to the sign-in screen.

2. Customer Account

2.1. The system must allow a User to sign-in as a Customer.

2.1.1. The system must verify that the details entered on sign-in matches that of the details stored on the database.

2.1.1.1. If the details do not match, the system must reject the sign-in attempt by the User and display an incorrect details message.

2.1.2. The system must display a prompt and provide a means of changing a password if the User has forgotten their password.

2.1.2.1. If the prompt is selected, the system must direct the User to a secured change password page.

2.1.2.1.1. The system must provide a prompt for the User to enter the account email associated with the forgotten password.

2.1.2.1.2. The system must provide a prompt for the User to enter the account phone number associated with the forgotten password.

2.1.2.1.3. The system must provide a prompt for the User to enter a new password with the same requirements provided in account registration.

2.1.2.1.4. The system must provide a prompt for the User to re-enter the new password.

2.1.2.1.5. The system must display a prompt for the User to confirm/deny that they wish to change their password.

2.1.2.1.5.1. If deny is selected, the system must direct the User to the sign-in page.

2.1.2.1.5.2. If confirm is selected, the system must verify that the new passwords entered in prompts 2.1.2.1.3. and 2.1.2.1.4. are identical and check that there is an account with the provided email and phone number combination in the accounts database.

2.1.2.1.5.2.1. If the passwords entered in prompts 2.1.2.1.3 and 2.1.2.1.4 are not identical and/or there is no account with the provided email and phone number combination in the accounts database, then the system must display an incorrect details message, reject the password change request and return the User back to the change password page with the previously entered prompts.

2.1.2.1.5.2.2. If the passwords entered in prompts 2.1.2.1.3 and 2.1.2.1.4 are identical and there is an account with the provided email and phone number combination in the accounts database, then the system must accomplish the following functionalities.

2.1.2.1.5.2.2.1. The system must direct the User to a secure code verification page.

2.1.2.1.5.2.2.2. The system must clearly display a message to the User that the system has sent the User an email containing a uniquely generated 6-character alphanumeric string to the provided email address.

2.1.2.1.5.2.2.3. The system must provide a prompt for the User to enter a 6-character alphanumeric string.

2.1.2.1.5.2.2.4. The email subsystem must send an email to the provided email address containing a uniquely generated 6-character alphanumeric string

2.1.2.1.5.2.2.5. The system must assign the uniquely generated 6-character alphanumeric string that was sent to the provided email address to the account with the provided email and phone number combination in the accounts database.

2.1.2.1.5.2.2.6. The system must allow the User to exit the application and when returning to the application direct the User to the code verification page.

2.1.2.1.5.2.2.7. The system must display a return to the previous page prompt for the User to exit the code verification page

2.1.2.1.5.2.2.7.1. If this prompt is selected, the system must clearly inform the User that doing so will result in the User being directed back to the sign-in page and cancelling the password change request.

2.1.2.1.5.2.2.7.2. The system must provide a prompt for the User to confirm/cancel this operation.

2.1.2.1.5.2.2.7.3. If confirm is selected, the system must direct the User back to the sign-in page and cancel the password change request.

2.1.2.1.5.2.2.7.4. If cancel is selected, the system must direct the User back to the code verification page.

2.1.2.1.5.2.2.8. The system must display a display a prompt for the User to confirm that they have entered the code in the provided prompt.

2.1.2.1.5.2.2.8.1. If confirm is selected, then the system must verify that the 6-character alphanumeric string entered in the prompt matches that of the 6-character alphanumeric string assigned to the account with the provided email and phone number combination on the accounts database.

2.1.2.1.5.2.2.8.2. If the verification in prompt 2.1.2.1.5.2.2.8.1 returns an unsuccessful result, then the system must return the User back to the code verification page and repeat the functionalities of prompts 2.1.2.1.5.2.2.2. to 2.1.2.1.5.2.2.8. and 2.1.2.1.5.2.2.8.1 until the verification has returned a successful result.

2.1.2.1.5.2.2.8.3. If the verification in prompt 2.1.2.1.5.2.2.8.1 returns a successful result, then the system must set the password in the associated account to the new verified password, display a successful password change message, remove the 6-character alphanumeric string assigned to the account and direct the User to the sign-in page.

2.2. The system must make all the functionalities of the Volunteer account accessible to the Customer account through changing the account type.

2.2.1. The system must provide a prompt to the Customer allowing them to change their account type.

2.2.1.1. The system must provide a prompt to confirm/deny that the Customer wishes to switch their account type.

2.2.1.1.1. If deny is selected, then the system must direct the Customer back to the Customer's home page.

2.2.1.1.2. If confirm is selected, then the system must switch the Customer's account type into a Volunteer account and display the Volunteer's account interface.

2.2.2. The system must not make the functionalities of the Volunteer account available when the User is logged-in as a Customer and has not confirmed to change their account type.

2.3. The system must allow a Customer to access their account details.

2.3.1. The system must allow a Customer to change their account details.

2.3.2. The system must allow a Customer to delete their own account.

2.3.2.1. The system must display a prompt to confirm/deny that the Customer wishes to delete their account.

2.3.2.1.1. If deny is selected, then the system must return the Customer back to their accounts details page.

2.3.2.1.2. If confirm is selected, then the system must delete their account from the database, verify that no trace of their details remains, and return the User to the sign-in screen.

2.4. The system must provide the Customer access to an Information page.

2.4.1. The system must provide the Customer text instructions detailing the use and functionality of the system.

2.4.2. The system must provide the Customer a range of videos detailing the use and functionality of the system.

2.4.3. The system must clearly display an example of how to use the application.

2.5. The system must allow a Customer to create a grocery order.

2.5.1. The system must check that a Customer is signed-in to their account before they are able to create a grocery order.

2.5.2. The system must set the Customer's details to the order when creating an order.

2.5.2.1. The system must set the Customer's first name, last name, email, phone number and address.

2.5.3. The system must allow a Customer to select a supermarket chain.

2.5.3.1. The system must then allow a Customer to select grocery products, add grocery products into a basket, remove grocery products from the basket, change the quantity of a grocery product that has been added to the basket and view the total amount to be paid through the selected supermarket's API.

2.5.3.2. The system must be able to display and provide up-to-date information on the produce/goods sold by the selected supermarket chain through the supermarket's API.

2.5.4. The system must allow a Customer to select a designated time of delivery from a selection of hourly slots.

2.5.5. The system must allow a Customer to review the order details.

2.5.5.1. The system must display a prompt for the Customer to confirm/deny that the order details are correct.

2.5.5.1.1. If deny is selected, the system must direct the Customer back to the order details page.

2.5.5.1.2. If confirm is selected, the system must direct the Customer to a PayPal payment gateway for payment.

2.5.6. The system must allow a Customer to cancel their grocery order.

2.5.6.1. The system must allow a Customer to cancel their grocery order if no Volunteer has accepted the grocery order and no Volunteer has been assigned to the grocery order.

2.5.7. The system must allow a Customer to view the profile of the Volunteer that has accepted and assigned to complete their order.

2.5.7.1. The system must allow a Customer to view the past Customer reviews of the Volunteer that accepted and assigned to complete their order.

2.5.8. The system must only transfer a Customer's order into the open orders queue after payment has been cleared and verified by PayPal.

2.6. The system must allow a Customer to view their order status.

2.6.1. The system must allow a Customer to view their current order status.

2.6.2. The system must allow a Customer to view their previous orders, up to the last 10 orders.

2.7. The system must provide a review prompt for a Customer to review and rate the service and reliability of the Volunteer that has taken their order.

2.7.1. The system must check and verify that the order between a Customer and a Volunteer has been completed before the Customer can review the Volunteer.

2.7.2. The system must allow the Customer to enter the number of stars the Volunteer should receive, from a range of 1 to 5 stars.

2.7.3. The system must allow the Customer to leave a written feedback in the form of a text box.

2.7.4. The system should allow a Customer to review the Volunteer that completed their order after ignoring the initial review prompt.

2.8. The system must allow a Customer to view the profile of the Volunteer that has accepted their order.

2.8.1. The system must allow the Customer to view the ratings and reviews of the Volunteer that accepted their order.

2.8.2. The system must allow the Customer to view the name and phone number of the Volunteer that accepted their order.

2.8.3. The system must not allow the Customer to view the address, email, password, and email of the Volunteer that accepted their order.

2.9. The system must allow a Customer to report an issue with the system.

2.9.1. The system must allow the Customer to type an encountered issue in a text box.

2.9.2. The system must transfer the created issue into an issues database only accessible by the system administrator.

2.9.3. The system must flag the system administrator of an open issue.

2.10. The system must allow a Customer to sign-out of their account.

2.11. The system must provide a means of resolution for any issues that arise, such as cancelling an order, rescheduling an order and payment.

3. Volunteer Account

3.1. The system must allow a User to sign-in as a Volunteer.

3.1.1. The system must verify that the details entered on sign-in matches that of the details stored on the database.

3.1.1.1. If the details do not match, the system must reject the sign-in attempt by the User and display an incorrect details message.

3.1.2. The system must display a prompt and provide a means of changing a password if the User has forgotten their password.

3.1.2.1. If the prompt is selected, the system must direct the User to a secured change password page.

3.1.2.1.1. The system must provide a prompt for the User to enter the account email associated with the forgotten password.

3.1.2.1.2. The system must provide a prompt for the User to enter the account phone number associated with the forgotten password.

3.1.2.1.3. The system must provide a prompt for the User to enter a new password with the same requirements provided in account registration.

3.1.2.1.4. The system must provide a prompt for the User to re-enter the new password.

3.1.2.1.5. The system must display a prompt for the User to confirm/deny that they wish to change their password.

3.1.2.1.5.1. If deny is selected, the system must direct the User to the sign-in page.

3.1.2.1.5.2. If confirm is selected, the system must verify that the new passwords entered in prompts 3.1.2.1.3. and 3.1.2.1.4. are identical and check that there is an account with the provided email and phone number combination in the accounts database.

3.1.2.1.5.2.1. If the passwords entered in prompts 3.1.2.1.3 and 3.1.2.1.4 are not identical and/or there is no account with the provided email and phone number combination in the accounts database, then the system must display an incorrect details message, reject the password change request and return the User back to the change password page with the previously entered prompts.

3.1.2.1.5.2.2. If the passwords entered in prompts 3.1.2.1.3 and 3.1.2.1.4 are identical and there is an account with the provided email and phone number combination in the accounts database, then the system must accomplish the following functionalities.

3.1.2.1.5.2.2.1. The system must direct the User to a secure code verification page.

3.1.2.1.5.2.2.2. The system must clearly display a message to the User that the system has sent the User an email containing a uniquely generated 6-character alphanumeric string to the provided email address.

3.1.2.1.5.2.2.3. The system must provide a prompt for the User to enter a 6-character alphanumeric string.

3.1.2.1.5.2.2.4. The email subsystem must send an email to the provided email address containing a uniquely generated 6-character alphanumeric string.

3.1.2.1.5.2.2.5. The system must assign the uniquely generated 6-character alphanumeric string that was sent to the provided email address to the account with the provided email and phone number combination in the accounts database.

3.1.2.1.5.2.2.6. The system must allow the User to exit the application and when returning to the application direct the User to the code verification page.

3.1.2.1.5.2.2.7. The system must display a return to the previous page prompt for the User to exit the code verification page.

- **3.1.2.1.5.2.2.7.1.** If this prompt is selected, the system must clearly inform the User that doing so will result in the User being directed back to the sign-in page and cancelling the password change request.
- **3.1.2.1.5.2.2.7.2.** The system must provide a prompt for the User to confirm/cancel this operation.
- **3.1.2.1.5.2.2.7.3.** If confirm is selected, the system must direct the User back to the sign-in page and cancel the password change request.
- **3.1.2.1.5.2.2.7.4.** If cancel is selected, the system must direct the User back to the code verification page.

3.1.2.1.5.2.2.8. The system must display a display a prompt for the User to confirm that they have entered the code in the provided prompt.

- **3.1.2.1.5.2.2.8.1.** If confirm is selected, then the system must verify that the 6-character alphanumeric string entered in the prompt matches that of the 6-character alphanumeric string assigned to the account with the provided email and phone number combination on the accounts database.
- **3.1.2.1.5.2.2.8.2.** If the verification in prompt 3.1.2.1.5.2.2.8.1 returns an unsuccessful result, then the system must return the User back to the code verification page and repeat the functionalities of prompts 3.1.2.1.5.2.2.2. to 3.1.2.1.5.2.2.8. and 3.1.2.1.5.2.2.8.1 until the verification has returned a successful result.
- **3.1.2.1.5.2.2.8.3.** If the verification in prompt 3.1.2.1.5.2.2.8.1 returns a successful result, then the system must set the password in the associated account to the new verified password, display a successful password change

message, remove the 6-character alphanumeric string assigned to the account and direct the User to the sign-in page.

3.2. The system must make all the functionalities of the Customer account accessible to the Volunteer account through changing the account type.

3.2.1. The system must provide a prompt to the Volunteer allowing them to change their account type.

3.2.1.1. The system must provide a prompt to confirm/deny that the Volunteer wishes to switch their account type.

3.2.1.1.1. If deny is selected, then the system must direct the Volunteer back to the Volunteer's home page.

3.2.1.1.2. If confirm is selected, then the system must switch the Volunteer's account type into a Customer account and display the Customer account interface.

3.2.2. The system must not make the functionalities of the Customer account available when the User is logged-in as a Volunteer and has not confirmed to change their account type.

3.3. The system must allow a Volunteer to access their account details.

3.3.1. The system must allow a Volunteer to change their account details.

3.3.2. The system must allow a Volunteer to delete their own account.

3.3.2.1. The system must display a prompt to confirm/deny that the Volunteer wishes to delete their account.

3.3.2.1.1. If deny is selected, then the system must return the Volunteer back to their accounts details page.

3.3.2.1.2. If confirm is selected, then the system must delete their account from the database, verify that no trace of their details remains, and return the User to the sign-in screen.

3.4. The system must provide the Volunteer access to an Information page.

3.4.1. The system must provide the Volunteer text instructions detailing the use and functionality of the system.

3.4.2. The system must provide the Volunteer a range of videos detailing the use and functionality of the system.

3.4.3. The system must clearly display an example of how to use the application.

3.5. The system must allow a Volunteer to access their jobs availability.

3.5.1. The system must display a prompt of a Volunteer's current job availability.

3.5.2. The system must allow a Volunteer to change their job availability.

3.6. The system must allow a Volunteer to view all Customer orders, restricted to a 20km radius of the postal code they would like to serve.

3.6.1. The system must display the Customer orders in a queue, with the least recently added Customer order displayed first.

3.6.2. The system must allow a Volunteer to view the basic details of the Customer for a given Customer order.

3.6.2.1. The system must not allow a Volunteer to view the Customer's full details, except for address, before accepting an order.

3.7. The system must allow a Volunteer to accept a Customer order.

3.7.1. The system must check that a Volunteer is signed-in to their Volunteer account before they are able to accept a Customer order.

3.7.2. The system must display a prompt to confirm that the Volunteer wishes to accept a Customer order.

3.7.3. The system must allow a Volunteer to cancel an accepted Customer order.

3.7.3.1. The system must only allow a Volunteer to cancel an accepted Customer order within 3 minutes of the Volunteer accepting the order.

3.7.4. The system must allow a Volunteer to view the Customer's full details, except for email, password and financial details, after accepting their order.

3.7.5. The system must display the directions to the address of the Customer using Google's Maps and Directions API.

3.7.6. The system must display nearby locations of the Customer order's supermarket chain using Google's Maps and Directions API.

3.8. The system must allow a Volunteer to request an order closure.

3.8.1. The system must require the Volunteer to provide proof of completion.

3.8.1.1. The system must allow the Volunteer to take and upload a picture for proof of completion.

3.8.2. The system must not be able to close an order if proof of completion is not provided by the Volunteer.

3.9. The system must allow a Volunteer to view the reviews associated with their Volunteer account and the reviews of every completed Customer order.

3.9.1. The system must be able to calculate and display an aggregate review rating, from 1 to 5 stars, from the reviews of every completed Customer order.

3.9.2. The system must allow a Volunteer to view the written feedback left in a Customer review.

3.9.3. The system must allow a Volunteer to view the rating, from 1 to 5 stars, left in a Customer review.

3.10. The system must allow a Volunteer to report an issue with the system.

3.10.1. The system must allow the Volunteer to type an encountered issue in a text box.

3.10.2. The system must transfer the created issue into an issues database only accessible by the system administrator.

3.10.3. The system must flag the system administrator of an open issue.

3.11. The system must allow a Volunteer to sign-out of their account.

3.12. The system must provide a means of resolution for any issues that arise, such as, inability to accept a Customer order, and payment forwarding.

4. Email and Updates

4.1. The system must enable update emails to be sent.

4.1.1. The system must be able to send an email to a Customer informing them that their order has been successfully created and display their order details.

4.1.2. The system must be able to send an email to a Customer informing them that payment has been completed and display an invoice of the transaction.

- **4.1.3.** The system must be able to send an email to a Customer informing them that their order has been delivered, and display proof of completion as uploaded by the Volunteer.
- **4.1.4.** The system must be able to send an email to a Volunteer informing them that they have accepted an order and display the order details.
- **4.1.5.** The system must be able to send an email to a Volunteer informing them that payment has been successfully forwarded into their account.
- **4.1.6.** The system must be able to send an email to a Volunteer informing them that they have successfully completed an order.
- **4.1.7.** The system must be able to send an email to a User informing them of a uniquely generated 6-character alphanumeric string for changing their password.

4.2. The system must enable registration emails to be sent.

- **4.2.1.** The system must be able to send an email to a User informing them of an account creation associated with their email.

5. Payment

5.1. The system must be able to automatically make and receive transactions from and to the at Your Step PayPal account.

5.2. The system must transfer the User to the PayPal payment gateway upon the confirmation of an order's detail to complete payment.

5.3. The system must be able to deduct payment for a Customer order from the Customer's financial details and transfer the payment into the at Your Step PayPal account.

- **5.3.1.** The system must verify that the payment for a Customer order has been forwarded to at Your Step's PayPal account before adding the Customer order into the open orders queue.

5.4. The system must be able to deduct payment for a cancelled Customer order from the at your Step's PayPal account and transfer the payment into the Customer's financial details.

- **5.4.1.** The system must check and verify that the transferred payment amount matches that of the cancelled Customer order amount before deducting the payment from the at Your Step's PayPal account.

5.5. The system must be able to forward the total amount of a completed Customer order from at Your Step's PayPal account into the Volunteer that completed the Customer order's financial account details.

5.6. The system must be able to receive any disputes related to application-side payment either from Customers or Volunteers.

- **5.6.1.** The system must be able to log these disputes and inform the system administrator of any open disputes.
- **5.6.2.** The system must provide a means of resolution for any disputes that arise, such as a late payment transfer to a Volunteer's financial account or incorrect deduction of funds from a Customer's financial account.

5.7. The system must be able to receive any disputes related to PayPal either from Customers or Volunteers.

- **5.7.1.** The system must direct the Customer or Volunteer of any disputes related to PayPal to PayPal's dispute system.

6. Accounts Database / Orders Database / Backend

6.1. The system must be able to add an account into the accounts database.

6.2. The system must be able to remove an account from the accounts database.

6.2.1. The system must only be able to remove an account from the accounts database after confirmation from the account owner.

6.3. The system must be able to accept Customer orders.

6.3.1. The system must be able to add a Customer's order into the queue of open orders.

6.3.1.1. The system must only be able to add a Customer's order into the queue of open orders after payment for the Customer's order has been verified.

6.3.1.2. The system must not be able to add multiple instances of a Customer order into the queue of open orders.

6.3.2. The system must be able to remove a Customer order from the queue of open orders.

6.3.2.1. The system must be able to remove a Customer order from the queue of open orders after a Volunteer has accepted and been assigned to the Customer order.

6.3.2.2. The system must be able to remove a Customer order from the queue of open orders after the Customer who has made the order has requested the order be cancelled.

6.3.3. The system must be able to add a Customer order back into the same position it held in the queue of open orders after a Volunteer has cancelled their acceptance of the Customer order.

6.4. The system must be able to reject a Customer order.

6.4.1. If payment is unable to be completed and verified, the system must not add the Customer order into the queue of open orders and inform the Customer that the order cannot be processed due to a payment error.

6.5. The system must be able to update a Customer order's status.

6.6. The system must be able to handle rescheduling of Customer orders.

6.6.1. The system must be able to handle the possibility of a Customer order going unaccepted by a Volunteer.

6.7. The system must be able to handle the cancellation of a Customer order, as requested by the Customer.

6.8. The system must be able to handle the cancellation of an accepted Customer order, as requested by the Volunteer.

6.8.1. The system must unassign the Volunteer from the Customer order and allow the Volunteer to view the queue of open orders.

6.9. The system must be able to assign a Volunteer onto a Customer order after the Volunteer has chosen to accept the Customer order.

6.9.1. The system must not be able to assign multiple Volunteers to a Customer order.

6.10. The system must be able to close a Customer order.

6.10.1. The system must not be able to close a Customer order with no assigned Volunteer.

6.10.2. The system must not be able to close a Customer order if there is no proof of completion.

6.10.3. The system must not be able to close a Customer order if payment has not been forwarded into the Volunteer's financial account details.

6.11. The system must be able to generate a unique 6-character alphanumeric string and correctly assign it to an account on the accounts database with only a provided email and phone number combination when that account has requested for its password to be changed.

6.11.1. The system must be able to remove the uniquely generated 6-character alphanumeric string from an account after the account has completed its password change request.

6.12. The system must be able to inform the systems administrator for any issues that arise.

Non-Functional Requirements

1. Usability

1.1. (Customer Account – Order) The system must allow a Volunteer to cancel an accepted Customer order, up to 3 minutes after an order has been accepted.

1.2. (Customer Account – Reviews) The system should allow a Customer to review the Volunteer that completed their order, up to 3 days of ignoring the initial review prompt.

1.3. (Email and Updates – Security Code for Password Change) The system must send the email containing the uniquely generated 6-character alphanumeric string to the provided email for a password change request, within 25 seconds of the User clicking confirm and passing the verification checks when completing the password change page.

1.4. (Accounts Database / Orders Database / Backend – Order Assignment) The system must remove an order from the open orders queue and be unable to accept other Volunteers onto the order within 1 second of a Volunteer accepting the Customer order.

1.5. (Accounts Database / Orders Database / Backend – Issues) The system should be able to log every issue encountered during operation.

1.6. (Accounts Database / Orders Database / Backend – Information) The system should display an appropriate loading screen to all Users during times when the application could be thought as unresponsive.

1.7. (Payment - Volunteer) The system should forward the total payment of a completed Customer order from the at Your Step's PayPal account into the Volunteer that completed the Customer order's financial account details within 20 seconds of the order being closed.

1.8. (Payment – Customer) The system should forward the total payment for a Customer's order from the Customer's financial account into the at Your Step's PayPal account after 5 seconds of a Customer 's order cancellation window.

2. Efficiency (Performance)

2.1. After a User successfully creates an account, the system should be able to process and add the new account into the accounts database within 4 seconds.

2.2. The system should be able to authenticate the email and password combination entered in the sign-in screen and access the account of the associated email within 5 seconds.

2.3. The system must be able to process a Customer's created order and add it into the queue within 10 seconds.

2.4. The system must be able to remove a Customer's created order, upon the Customer's request, from the queue if no Volunteer has accepted the order or been assigned to the order within 5 seconds.

2.5. The system must be able to process a Volunteer accepting a Customer order and display the order details within 10 seconds.

2.6. The system must be able to add a Customer order that was accepted and then cancelled by a Volunteer back into the same place it held in the queue within 5 seconds after the Volunteer has confirmed cancellation.

2.7. The system must be able to process a proof of order completion uploaded by the Volunteer and close the order within 20 seconds.

2.8. The system should be able to create an account and send a registration email to the User within 2 minutes.

2.9. The system must be able to create a Customer order and send an update email to the Customer within 2 minutes.

2.10. The system must be able to update a Customer's order status and send an update email to the Customer within 2 minutes of a Volunteer accepting their order.

3. Efficiency (Space)

3.1. The system must be able to process, record and save both text and images.

3.2. The system should be able to record and save the details of every order made by Customers

3.3. The system should be able to record and save the details of every account registered on the application.

3.4. The system should be able to record and save the details of every financial transaction occurring on the application

4. Security

4.1. The system must be able to direct any User, Customer or Volunteer to the PayPal gateway in a secured and encrypted form.

4.2. Debit/credit card transactions must be handled through the PayPal payment gateway.

4.3. The system must transfer a Customer's personal data from the accounts database to the PayPal payment server in an encrypted form.

4.4. The system must record and store a User's personal data in an encrypted form.

4.5. Volunteers must receive an order recipient's personal information e.g., name, phone number, address) through an encrypted service.

5. Scalability

5.1. The system must be able to support 10,000 concurrent orders.

5.2. The system should be able to store 25,000,000 unique accounts

5.3. The system could integrate a load-balancer in order to increase orders capacity and the reliability of the system.

6. **Reliability**
   6.1. The system must be available for 99.9% of the time on a 24/7 basis.
   6.2. The system should only be down for maintenance/repair for a combined 44 minutes in a month.
   6.3. The system should aim to do maintenance/repair at times of minimum User count on the application.
   6.4. The system should not exceed 30 minutes of downtime after a critical failure.
   6.5. The system should have a duplicate accounts and orders database that is automatically synced every 15 minutes.

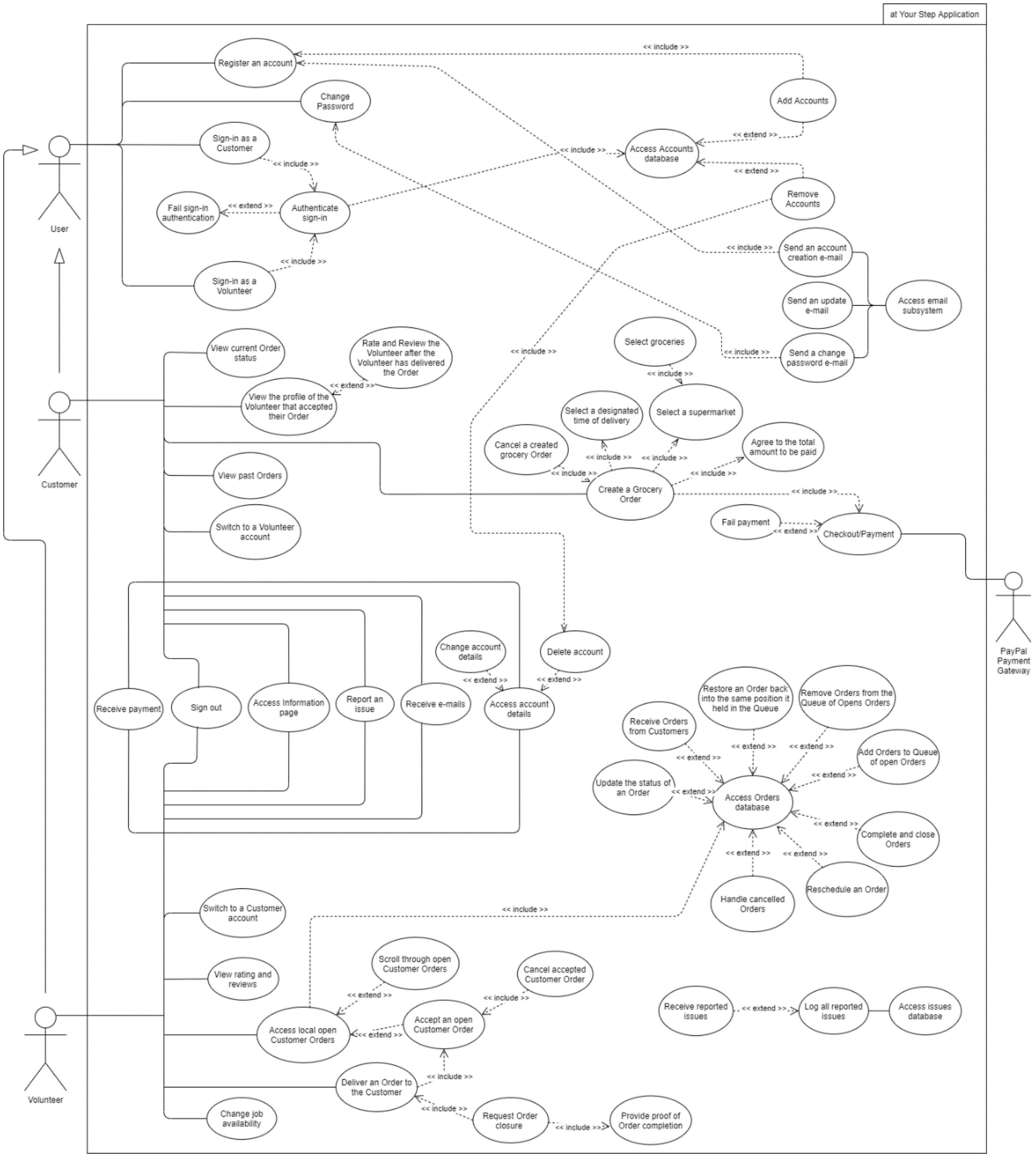
7. **Interoperability**
   7.1. The system must correctly integrate with the Payflow (PayPal) payment gateway.
   7.2. The system must correctly integrate with all the selectable supermarket chain's API.
   7.3. The system must correctly integrate with Google's Maps and Directions API.
   7.4. The system must correctly integrate with the at Your Step orders database.
   7.5. The system must correctly integrate with the at Your Step accounts database.
   7.6. The system must correctly integrate with the at Your Step issues database.

B. Software Design with UML

1. Use Case Diagram

The Use Case diagram for the at Your Step application is presented below. The actors that have been identified are depicted outside the system boundary, which is represented by the rectangle that surrounds the diagram. Within the boundaries are the primary use cases along with their relationships to the main actors and between other use cases

.


at Your Step Application
User
Customer
Volunteer
PayPal Payment Gateway
Register an account
Change Password
Sign-in as a Customer
Authenticate sign-in
Fail sign-in authentication
Sign-in as a Volunteer
Add Accounts
Access Accounts database
Remove Accounts
Send an account creation e-mail
Send an update e-mail
Send a change password e-mail
Access email subsystem
View current Order status
Rate and Review the Volunteer after the Volunteer has delivered the Order
View the profile of the Volunteer that accepted their Order
View past Orders
Switch to a Volunteer account
Select groceries
Select a designated time of delivery
Select a supermarket
Cancel a created grocery Order
Agree to the total amount to be paid
Create a Grocery Order
Fail payment
Checkout/Payment
Change account details
Delete account
Receive payment
Sign out
Access Information page
Report an issue
Receive e-mails
Access account details
Receive Orders from Customers
Restore an Order back into the same position it held in the Queue
Remove Orders from the Queue of Opens Orders
Add Orders to Queue of open Orders
Update the status of an Order
Access Orders database
Complete and close Orders
Handle cancelled Orders
Reschedule an Order
Switch to a Customer account
View rating and reviews
Scroll through open Customer Orders
Cancel accepted Customer Order
Access local open Customer Orders
Accept an open Customer Order
Deliver an Order to the Customer
Change job availability
Request Order closure
Provide proof of Order completion
Receive reported issues
Log all reported issues
Access issues database
<< include >>
<< extend >>

2. Actors Documentation

The following use-cases have been selected for further documentation. These are:

1. Sign-in as a Customer account and create a grocery order.
2. Sign-in as a Volunteer and accept an open Customer order.

The use-cases are presented below, showing the corresponding actors, pre/post conditions and flow of events.

Use Case: Sign-in as a Customer account and create a grocery order	
Actors	User, Customer, at Your Step system, accounts database, orders database, email sub-system, payment sub-system
Preconditions	1. The User has installed the at Your Step application on their smartphone. 2. The User has previously registered an account on the at Your Step application. 3. The User's financial details has been previously set-up on the account during account registration.
Flow of Events	1. The use-case starts when a User selects the at Your Step application on their smartphone. 2. User is directed to the at Your Step front page by the system. 3. User selects to sign-in as a Customer. 4. User is directed to the sign-in page, where they correspondingly enter their email and password and then selects Sign-In. 5. On clicking Sign-In, the system checks the accounts database for an account with the entered email and password combination. a. If the system finds an account with the corresponding email and password combination on the accounts database, the system verifies that the details match, the User is signed-in to the application as a Customer and directed to the Customer home page. b. If the system does not find an account with the corresponding email and password combination on the accounts database, the system rejects the sign-in attempt and displays an incorrect details message on the front page with the details of the previously entered email and password in their respective prompts. 6. User is directed to the Customer home page by the system

   a. After this stage, the User is now referred to as Customer as they have signed-in as a Customer.

7. Customer selects Order from the Customer home page and is directed to the Create a New Order page.
   a. If the Customer selects return, the Customer is directed to the Customer home page.
8. Customer selects the supermarket chain they wish to buy groceries from and clicks on next.
   a. If the Customer selects return, the Customer is directed to the Customer home page.
9. Customer selects a designated time slot for delivery and clicks on next.
   a. If the Customer selects return, the Customer is directed to the select a supermarket chain page.
10. Customer is then directed to the pages of available groceries through the selected supermarket's chain API.
11. Customer then selects groceries, and the system displays a modifiable quantity amount that the Customer can increase and decrease.
12. Customer then selects all their required groceries along with the corresponding quantity.
13. Customer then clicks on the shopping cart icon where the system directs them to their order's details page and the Customer reviews all the details of their order.
    a. If the Customer selects return, the Customer is directed back to the pages of available groceries with the already selected groceries and their corresponding amounts saved.
14. Customer clicks on submit and the system displays a prompt for the Customer to confirm that they wish to submit the order
    a. If the Customer selects No, then the Customer is directed back to their order's details page.
    b. If the Customer selects Yes, then the following functionalities shall occur.
15. The payment sub-system deducts the total payment amount for the order from the Customer's financial account and transfers the payment amount to the at Your Step PayPal account.
    a. If the Customer does not have sufficient funds for their order, then the system rejects the order and informs the Customer that the order cannot be processed due to a payment error.
    b. If no payment related error occurs, then the system validates that payment has successfully been transferred and the following functionalities shall occur.

	16. The at Your Step system creates a page for the Customer’s order accessible to the Customer and Volunteers. 17. The Customer's order is then submitted to the orders database. 18. The email sub-system sends an email to the Customer's email address notifying that their order has been successfully created and displays their order’s details. 19. The email sub-system sends an email to the Customer's email address notifying that payment has been completed. 20. The orders database then moves the Customer’s order to the open orders queue and displays the order to the Volunteers.
Post-Conditions	1. The Customer’s order has been successfully created and is now available to be accepted by a Volunteer. 2. An order page has been successfully created and is now accessible by the Customer in order to view the order’s status and details. 3. Payment has successfully been processed and transferred to at Your Step's PayPal account. 4. The at Your Step system has updated its records as a result of the transaction. 5. The email subsystem has sent an email informing the Customer that the order has been created and payment has been completed.

Use Case: Sign-in as a Volunteer and accept an open Customer order	
Actors	User, Volunteer, at Your Step system, accounts database, orders database, email sub-system
Preconditions	1. The User has installed the at Your Step application on their smartphone. 2. The User has previously registered an account on the at Your Step application.
Flow of Events	1. The use-case starts when a User selects the at Your Step application on their smartphone. 2. User is directed to the at Your Step front page by the system. 3. User selects to sign-in as a Volunteer. 4. User is directed to the sign-in page, where they correspondingly enter their email and password and then selects Sign-In. 5. On clicking Sign-In, the system checks the accounts database for an account with the entered email and password combination.

	a. If the system finds an account with the corresponding email and password combination on the accounts database, the system verifies that the details match, the User is signed-in to the application as a Volunteer and directed to the Volunteer home page. b. If the system does not find an account with the corresponding email and password combination on the accounts database, the system rejects the sign-in attempt and displays an incorrect details message on the front page with the details of the previously entered email and password in their respective prompts. 6. User is directed to the Volunteer home page by the system. a. After this stage, the User is now referred to as Volunteer as they have signed-in as a Volunteer. 7. On the Volunteer home page, the at Your Step system displays a scrollable list of open Customer orders. 8. Volunteer scrolls through the list and clicks on an order. 9. Volunteer is then directed to that order's page and the system displays the order's details and the basic details of the Customer who created the order to the Volunteer. a. If the Volunteer wishes to not accept this order, the Volunteer clicks on return, the system directs the Volunteer back to the Volunteer Home Page and displays an updated scrollable list of open Customer orders. b. If the Volunteer wishes to accept this order, the Volunteer clicks on Accept and the following functionalities shall occur. 10. The system displays a prompt for the Volunteer to confirm that they wish to fulfill this order. a. If the Volunteer selects No, then the Volunteer is directed back to the selected order's page. b. If the Volunteer selects Yes, then the following functionalities shall occur. 11. The at Your Step system assigns the Volunteer to the order and promptly removes the accepted order from the queue of open Customer orders viewable by all Volunteers. 12. The email sub-system sends an email to the Volunteer's email address informing them that they have accepted an order and displays the order's details. 13. Volunteer is then directed to a page for the Customer's order and displayed the nearest located supermarket of

	choice for the order and the full details of Customer who created the order. 14. Volunteer then goes on to fulfill the order.
Post-Conditions	1. The system has assigned the Volunteer to the Customer order. 2. The system has removed the Customer order from the queue of all open Customer orders. 3. The at Your Step system has updated its records as a result of the actions taken by the Volunteer. 4. The Volunteer will go on to fulfill the Customer order. 5. The email subsystem has sent an email informing the Volunteer that they have accepted an order.

3. Scenarios

Scenario 1

<u>User ordering groceries:</u>

- **INITIAL ASSUMPTION**: A user who is self- isolating or must shield themselves from the public due to COVID wishes to place an order using the at Your Step application.
- **NORMAL**: The user is prompted to login or create a new account, once either of these have been selected and the user has entered required details, they will be asked to select a supermarket chain that the groceries will be purchased from. Once the user has selected all the groceries they want, the list of groceries can be checked through by the user and the payment can be confirmed. On confirmation of order the grocery list one of the registered volunteers can accept it and the user can accept the volunteer to carry out the order (depending on past reviews of the volunteer) - the order can then be carried out.
- **WHAT CAN GO WRONG:**
    - The user inputs the wrong details when signing into their account (Maybe forgotten details)- They will be prompted with a message that their details are incorrect and will be asked to login again or create a new account.
    - There may not be a volunteer nearby for the job to be accepted. If so, then the order will be put on standby until there is a volunteer available.
- **OTHER ACTIVITIES:**
    - Volunteer will have to have their status set to available for them to be selected as an option for delivery
    - Once order is confirmed an email confirming the grocery list will be sent to the users specified email address.
    - Once order is conformed an email confirming the payment for the groceries will be sent to the email specified by the user.
    - Volunteer may cancel the accepted order due to personal reasons etc., if so, the order will be passed to another volunteer.

- **SYSTEM STATE ON COMPLETION:** The user will still be logged in, and the order will be in the midst of being delivered by the volunteer.

Scenario 2

<u>Volunteer accepting order from user:</u>

- **INITIAL ASSUMPTION:** Volunteer for the ***at Your Step APP*** during COVID pandemic checks through orders and accepts one of them to be completed and delivered.
- **NORMAL:** The volunteer signs in with the details they provided when first signed up as a volunteer and sets their job availability accordingly. Volunteer can check all user orders sent out within a customizable radius (default 20km) from their position. The volunteer can accept the most recent order sent by a user for example. If the user accepts that volunteer to carry out the order the delivery can be carried out by the volunteer.
- **WHAT CAN GO WRONG:**
    - o A first-time volunteer may be confused of the set up and functionality of the system when they have selected to be a volunteer - an information page (and information videos) will be provided on the system to aid volunteers in understanding the functionality of the system and feedback can be left by volunteers to allow for further editing of the system depending on feedback.
    - o Volunteer may cancel the accepted order from the user due to valid personal reasons. If so, the order will be passed onto another volunteer who has their status set as available.
- **OTHER ACTIVITIES:** User must accept the volunteer to be able to carry out the delivery based on past reviews from other orders. If first time volunteer, there will be no reviews for user to see.
- **SYSTEM STATE ON COMPLETION:** After the order has been accepted, it is removed from the pending orders list where it can be seen by other volunteers and its status is changed to in progress, awaiting delivery by the volunteer. Both the volunteer and customer will then have access to the order's summary through which they can either track or visualize anything relevant to that order.

Scenario 3

<u>Volunteer delivers the groceries:</u>

- **INITIAL ASSUMPTION:** Volunteer delivers groceries and uploads the receipt as proof of delivery through the app.
- **NORMAL:** Volunteer arrives at the customer's address and delivers the groceries. He/she then opens the app and submits a picture of the receipt in the order's proof of delivery section which then can be seen by both the volunteer and the customer at any time.
- **WHAT CAN GO WRONG:**
    - o The picture quality from the volunteer's phone might be too low, making the receipt unintelligible for both users and system admins.
    - o The size of the picture might be too big in which case the system will downscale the image to fit within certain parameters making it readable to the customer and also to fit within the database.

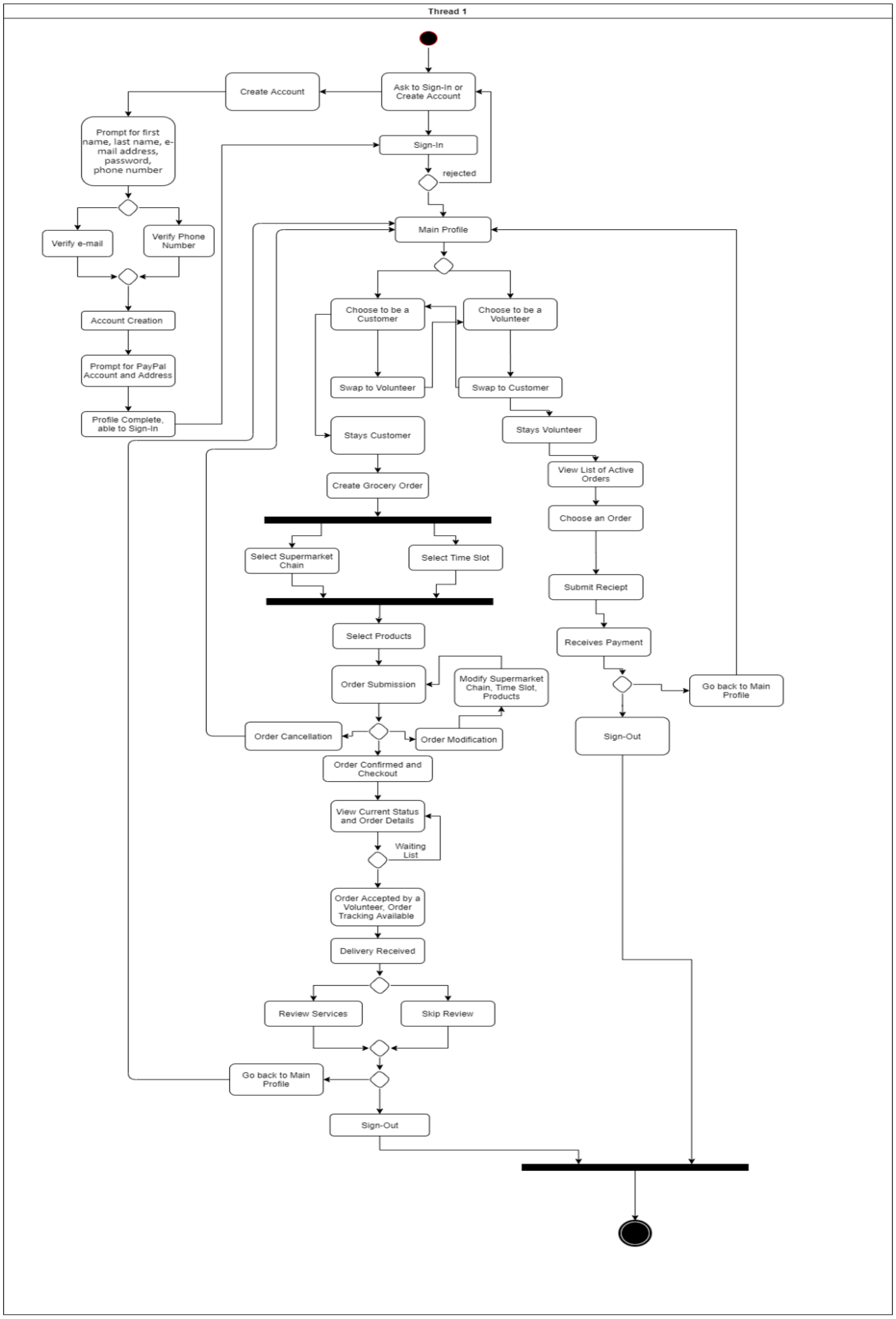
- **OTHER ACTIVITIES:**
  - The volunteer will handle the groceries respecting all government guidelines regarding health safety and social distancing.
  - The customer can appeal this proof of delivery if they feel something is wrong about the order within the next couple of hours.
- **SYSTEM STATE ON COMPLETION:** After the proof of delivery has been uploaded the order transitions from being an active order to a completed order, thus, it is removed from the home screen of both users and placed in their history. Internally the order just changes status and joins a more compressed database. After the order is delivered the customer will be able to access a track record of where the volunteer has been. The address of the customer is removed from the volunteer's order history for safety reasons.

Scenario 4

<u>Customer leaves feedback to the volunteer:</u>

- **INITIAL ASSUMPTION:** The customer is prompted to leave a review regarding their recent order.
- **NORMAL:** After the order has been delivered and proof of delivery is uploaded to the database, the customer receives a pop-up asking them to write feedback to the volunteer that has delivered their order. The customer selects the amount of stars that he/she wants to give to the volunteer and may or may not write a short description to their feedback. Pressing submit finishes this interaction.
- **WHAT CAN GO WRONG:**
  - The text might be too long to be submitted, forcing the customer to reduce the size of his feedback.
  - The customer might upload an empty review by mistake however the system will prompt them that they are unable to do so.
- **OTHER ACTIVITIES:**
  - The customer may skip this pop-up so no feedback will be displayed to the volunteer.
  - The customer will be able to come back to his/her review at any time and edit it.
- **SYSTEM STATE ON COMPLETION:** The review is checked for any profanity and then uploaded to the database. It will be displayed on the volunteer's profile together with an updated aggregation score moments after everything is completed on the back-end regarding processing.

4. Activity Diagram



5. Class Analysis

i. Noun-verb analysis

In order to identify the key classes of the system we perform noun-verb analysis on the system's specification. Nouns have been highlighted using bold text style and verbs using italics text style. This information will be used together with the Responsibility Driven Analysis to create the system's Class Diagram.

Our **system** is a **mobile application** that *provides* a matching **service** between **people self-isolating** as a result of a positive **COVID test** or needing *to shield* because of a **health condition**/old age, and **volunteers** who are *willing to provide support* by *buying* **groceries** and *completing* a contact free **delivery** to the self-isolating or vulnerable **person**. Since **supermarket** home delivery **service**s have been incredibly busy during the lockdown period and in many cases **people** have struggled to get delivery slots, this **service** will allow **volunteers** to *help* ensure that vulnerable and self-isolating **people** are able to get the **food** they need.

The **system** will allow the **user** to create an **account** *holding* basic contact details (first name, last name, email address, password, phone number). After creating an **account**, the **system** will prompt every **user** to *link* a PayPal account for pre-approved payments and to provide their address. The **user** can choose to either be a **customer** willing to *purchase* **groceries** or a **volunteer** that wishes to *deliver* a **customer**'s **order**. **Users** will be able to *update* their **account details** at any time and swap between the two available account modes: **volunteer** and **customer**. The account will only be able to be accessed with a valid **email address** and **password** pair.

If the user *identifies* themselves as a **customer**, they will be able to *create* a **grocery order** by *selecting* a supermarket chain and a time slot in which they would like their order to be delivered. The user will then be able to *select* from a **list of products** available in that respective **store** and *create* their desired **shopping list**. They will then be able to *submit* the **order** and the **system** will *display* the **details of this order** (the shopping list, time slot, address for delivery, and supermarket chain). The **user** will then have the option to confirm or deny the details. If the **user** denies, they will be directed back to *modifying* their current **order**. If the **user** confirms, the **order** will be added to a **list of active orders** within the **system** from which it will be accepted by a **volunteer**.

After *placing* an **order**, money is put on hold from the **user**'s **PayPal account** and they will be able to *view the current status* together with a **summary of the order's details**. If *accepted* by a **volunteer**, the **customer** will be able to *track* the **active order** until it is delivered. After the **order** has been *delivered* the **customer** is prompted to *review* the **volunteer**'s **services** through a **five-star rating pop-up** and a text box *describing* their experience. If the user *skips* this interaction, they can come back at any time and *review* the **volunteer**'s **service**s by *accessing* their order history which holds a list of all their past orders.

If the **user** identifies themselves as a **volunteer**, they will be able to *view* a **list of active orders** in the **area**, calculated by the **system** using the **volunteer**'s **location** and the preferred **maximum distance** *set*. They will be able to *see* a short **summary** of each **order** in the **list** with relevant information which can be *sorted* to their preference, thus, helping them choose which order they wish to fulfill at free will. If they choose to *accept an order,* they will be shown a path to the **customer** via **Google Maps** together with the **shopping list** and all available **stores** (chosen by the **customer**) in the area. Once they *have delivered* the **groceries**, they will *submit* **a photograph of the receipt** and the **system** will *transfer* the **money** on hold from the **customer** to the **volunteer**. After each **completed order** the **volunteer**'s **review score** is *calculated and updated* by the **system** (if the **customer** *leaves* a **review**) to reflect an **average** of their **completed services**.

The **system** will *send* relevant emails to both **customer**s and **volunteer**s when an important event occurs such as: order has been delivered, there is an issue with the order, account is created etc. Both **volunteers** and **customers** will be able to *open* **tickets** to the **system admin** regarding any problem they might encounter while using the **system**.

Word/Phrase	**Accepted**	**Reason**
system	No	Refers to the app itself.
mobile application	No	Refers to the type of our app.
provides matching service	No	Refers to what the system does in general but not specific enough.
service	No	Far too general and not related to the system.
people self-isolating (customer)	Yes	Also encountered as “customers”. Refers to the customers of the app, a type of user . Definitely deserves a class of its own.
COVID test	No	Reason for which the app is made but not part of the system.
to shield	No	Not related to the system.
health condition	No	Not related to the system.
volunteer(s)	Yes	This is one of the users of the system, so it deserves a class of its own.
user	No	Too general.
groceries	No	Not used by the system. But implied by it.
order (groceryOrder)	Yes	It is one of the core functionalities of the system. Both customers and volunteers interact with this.
account	Yes	One of the core components of the app. Allows users to utilize the app.
area	No	Too general and not withing the scope of the system.
volunteer’s services	No	Too general. Refers to the user’s actions. Not related to the app.
List of orders	Yes	Required for customers to store their order and for the

		volunteer’s to view available orders.
rating/review	Yes	Used by the customers to offer feedback on the volunteer’s services. Should have a class of its own.
List of products		
ticket	Yes	Used by users to raise issues regarding the app.
event	No	Too general
create and submit order	Yes	It represents an action of the customer. Can be used as a method in the customer class
add/remove item to shopping list	Yes	Represents an action of the customer through the grocery order class. Will be used as a method.
modify order	Yes	Action of the customer. Can be used as a method.
placing order	No	Duplicate to create order.
track order	Yes	Action by the customer. Will be added as a method.
write review	Yes	Method used by the customer to write a review.
skip review	Yes	Action of the customer. Will be added as a method.
view list of active orders	Yes	This is one of the main functionalities of the volunteer. Must be added as a method.
sort list of active orders	Yes	Method for the volunteer class.
accept order	Yes	Main functionality of the volunteer. Must be added as a method.
deliver order	No	Not really related to the system’s design however it is how the volunteer acts outside of the system.
submit photograph	Yes	This is how we verify the above mentioned action outside of the system and turning an inside action. Must be added as a method.
transfer the money	No	Action outside of the system.
open ticket	Yes	Really important action done by any user account. Must be used as a method.

link PayPal account	Yes	Core method used by Account for transactions within the app.
update account details	Yes	Method which provides functionality for users to keep info up to date.
describe their experience	No	Far too general to be a method.
summary of the order's details	No	Not specific enough to be a class.
system will send emails	No	Talks about the system's actions in general rather than of a specific class.
holding basic information	No	Too general to be a method. Every class holds information.
Purchase groceries and deliver order	No	Actions outside of the system.
access order history	Yes	Action of both the customer and the volunteer. Add as a method.
shopping list	No	Important part of the system, however, can be integrated within the order without requiring a class. Will become part of the order itself.
stores	Yes	Will be added as an API class used by the grocery orders.
Paypal	Yes	Important external class that manages anything related to payments and transactions.
Google Maps	Yes	External class used by both customers and volunteers.
Server	Yes	External class used by both customers and volunteers. Needed to stack data.
Email	Yes	External subsystem used for email confirmations throughout the app.

Account	
Responsibilities	*Collaborators*
Holds all of the user's information and enables them to utilize all of the app's features.	

Volunteer	
Responsibilities	*Collaborators*

A type of Account. Enables the user to access the list of orders created by customers and accept them, thus being able to deliver.	Account

Customer	
Responsibilities	*Collaborators*
A type of Account. Enables the user to be able to create orders which will be eventually picked up by volunteers.	Account

groceryOrder	
Responsibilities	*Collaborators*
Holds all of the information regarding the user’s order. Selected supermarket, timeslot, item list, etc.	Customer

orderQueue	
Responsibilities	*Collaborators*
Holds all of the orders created by the customers. After an order is accepted by a volunteer it adds a couple extra detail fields to the order making it an activeOrder	groceryOrder activeOrder

Review	
Responsibilities	*Collaborators*
Holds all of the information of a review from the customer towards the volunteer.	Customer Volunteer

Ticket	
Responsibilities	*Collaborators*
Holds all of the information of a ticket created by any user.	Account

ticketQueue	
Responsibilities	*Collaborators*
Holds all of the tickets created by users allowing system moderators to view and solve current issues regarding the system.	Ticket

Supermarket API

Responsibilities	*Collaborators*
Fetches all of the products from a supermarket for the user to select within the groceryOrder	groceryOrder

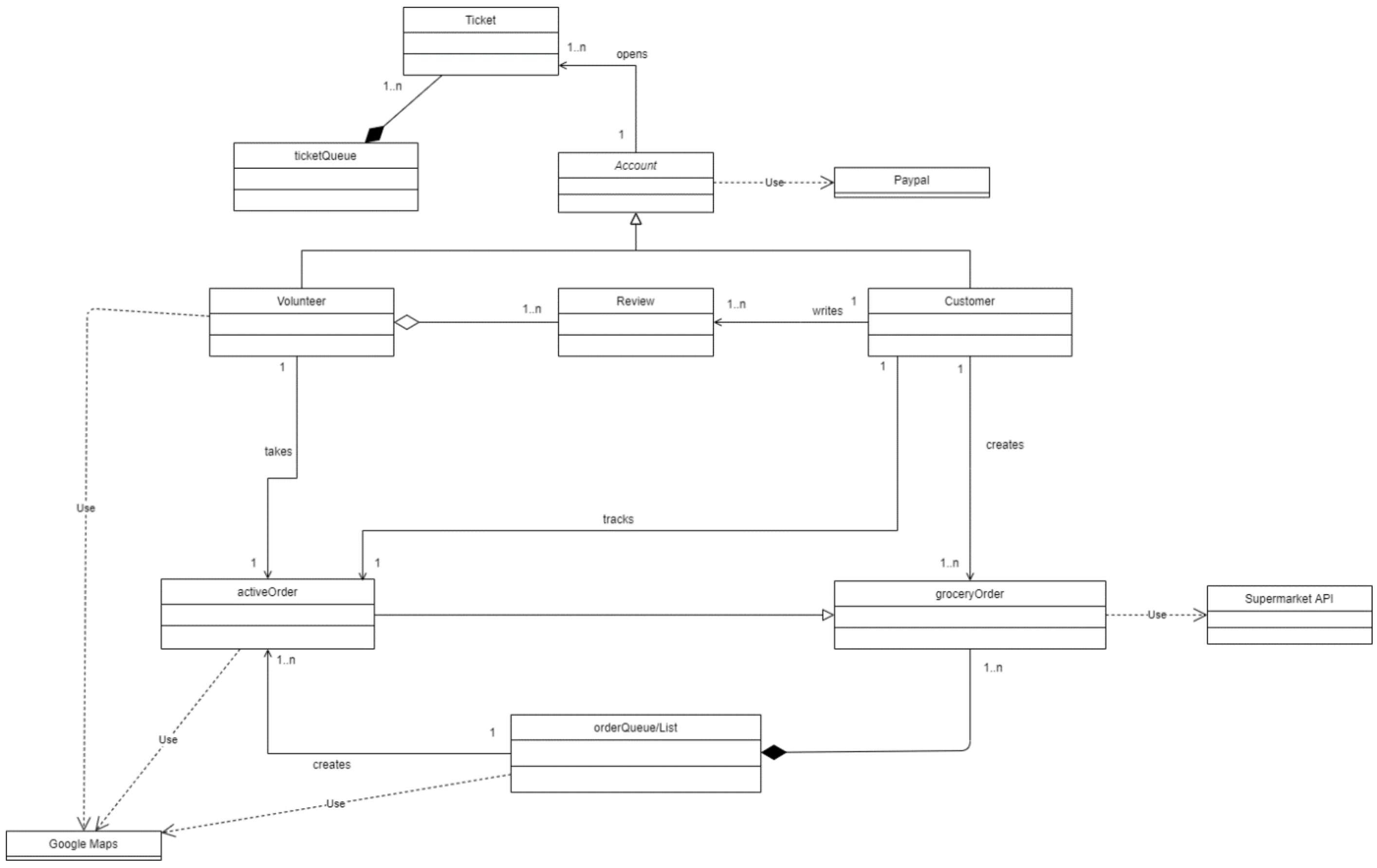
Paypal	
Responsibilities	*Collaborators*
Handles all transactions and payments within the app.	Account

Google Maps	
Responsibilities	*Collaborators*
Provides accurate tracking, pathfinding and location for multiple classes within the system.	Volunteer, activeOrder, orderQueue

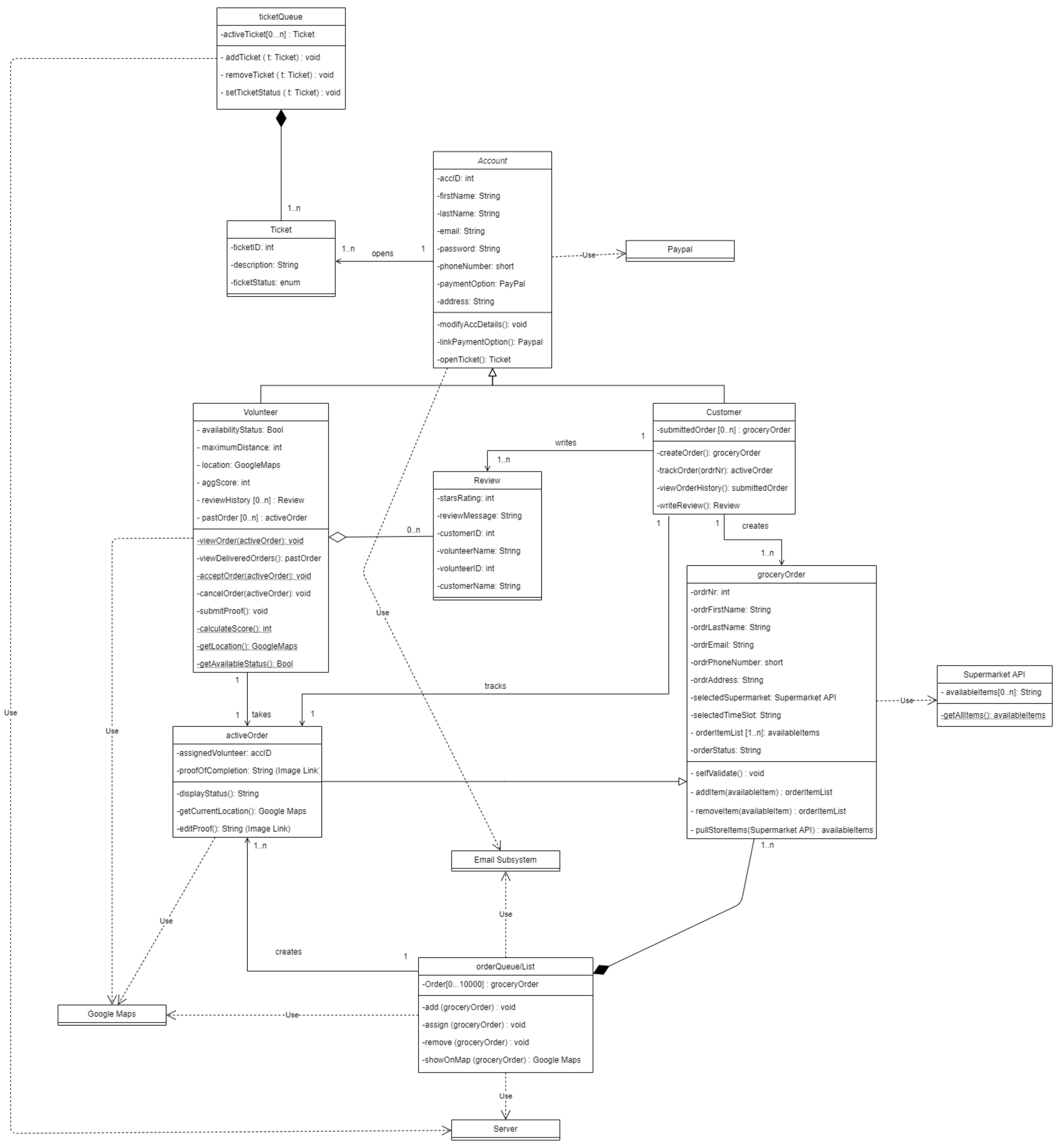
Email Subsystem	
Responsibilities	*Collaborators*
Sends confirmation and status emails to users regarding key events in the app.	Account, orderQueue/List

Server	
Responsibilities	*Collaborators*
Stores the orders and tickets queue for users and admins to access/manage	ticketQueue, orderQueue/List

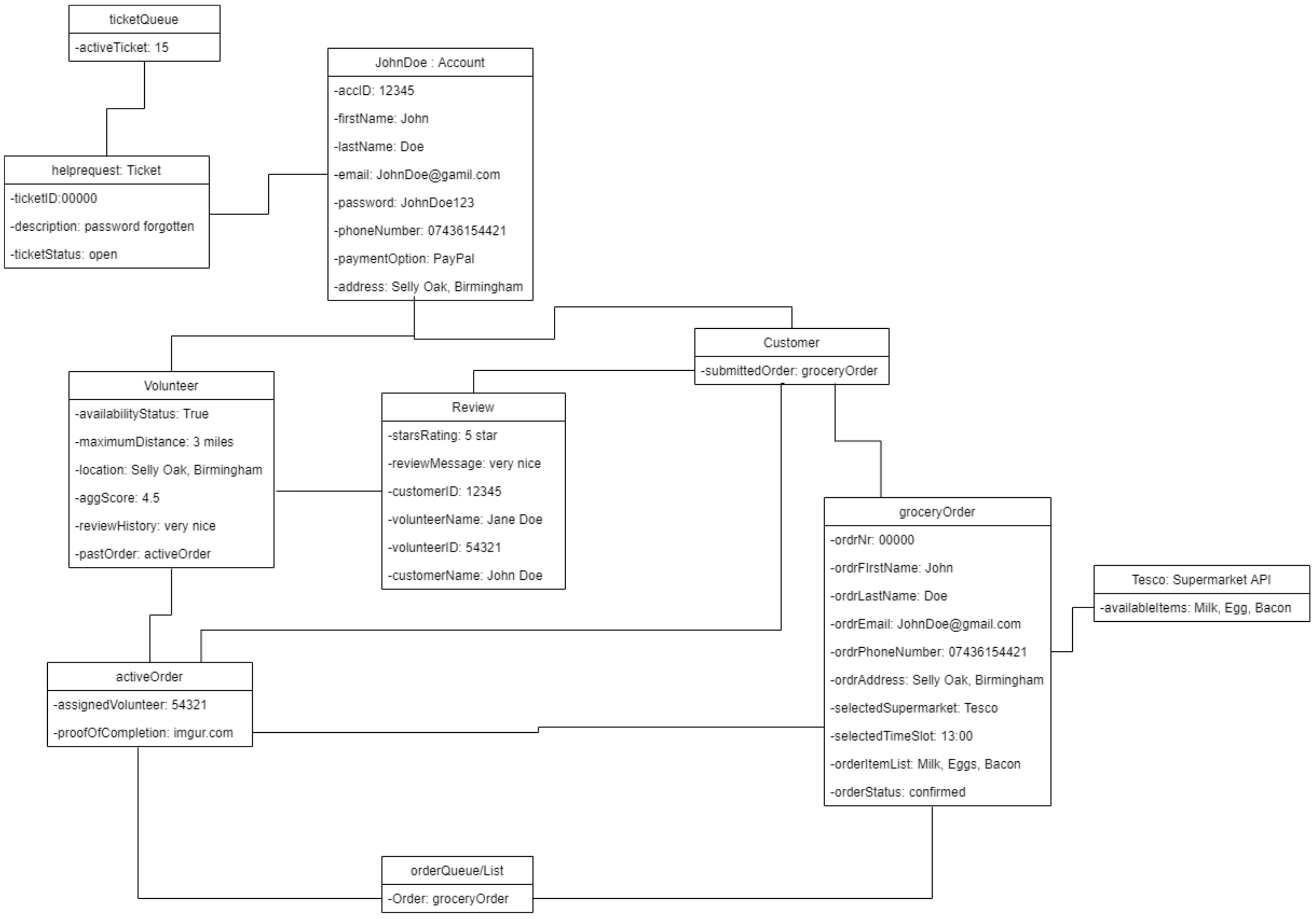
ii. First Cut Class diagram



iii. Final Class Diagram

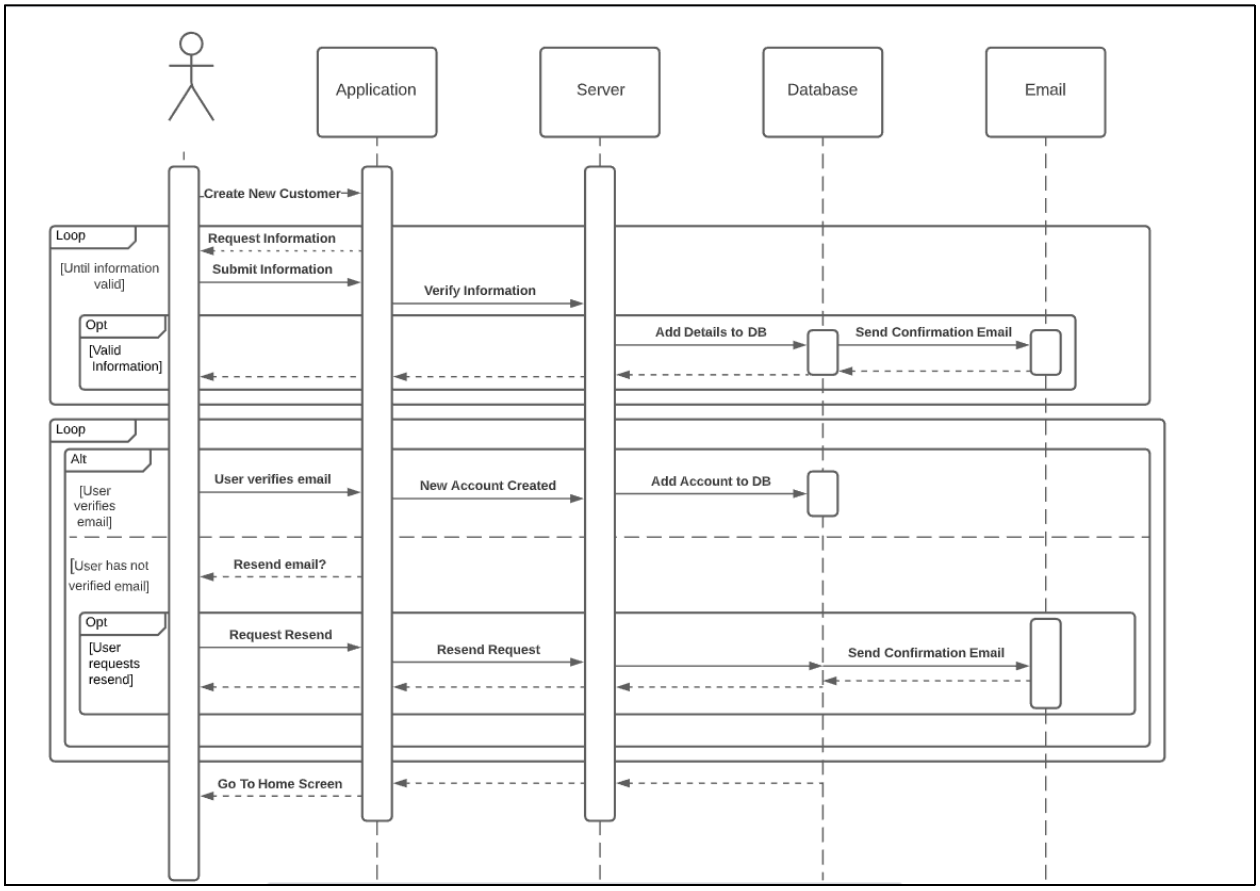

ticketQueue
-activeTicket[0...n] : Ticket
- addTicket (t: Ticket) : void
- removeTicket (t: Ticket) : void
- setTicketStatus (t: Ticket) : void
1..n
Ticket
-ticketID: int
-description: String
-ticketStatus: enum
1..n
opens
1
Account
-accID: int
-firstName: String
-lastName: String
-email: String
-password: String
-phoneNumber: short
-paymentOption: PayPal
-address: String
-modifyAccDetails(): void
-linkPaymentOption(): Paypal
-openTicket(): Ticket
Use
Paypal
Volunteer
- availabilityStatus: Bool
- maximumDistance: int
- location: GoogleMaps
- aggScore: int
- reviewHistory [0..n] : Review
- pastOrder [0..n] : activeOrder
-viewOrder(activeOrder): void
-viewDeliveredOrders(): pastOrder
-acceptOrder(activeOrder): void
-cancelOrder(activeOrder): void
-submitProof(): void
-calculateScore(): int
-getLocation(): GoogleMaps
-getAvailableStatus(): Bool
Customer
-submittedOrder [0..n] : groceryOrder
-createOrder(): groceryOrder
-trackOrder(ordrNr): activeOrder
-viewOrderHistory(): submittedOrder
-writeReview(): Review
writes
1
1..n
Review
-starsRating: int
-reviewMessage: String
-customerID: int
-volunteerName: String
-volunteerID: int
-customerName: String
0..n
Use
1
1
creates
1..n
groceryOrder
-ordrNr: int
-ordrFirstName: String
-ordrLastName: String
-ordrEmail: String
-ordrPhoneNumber: short
-ordrAddress: String
-selectedSupermarket: Supermarket API
-selectedTimeSlot: String
- orderItemList [1..n]: availableItems
-orderStatus: String
- selfValidate() : void
- addItem(availableItem) : orderItemList
- removeItem(availableItem) : orderItemList
- pullStoreItems(Supermarket API) : availableItems
Use
Supermarket API
- availableItems[0..n]: String
-getAllItems(): availableItems
Use
Use
1
1
takes
tracks
1
activeOrder
-assignedVolunteer: accID
-proofOfCompletion: String (Image Link)
-displayStatus(): String
-getCurrentLocation(): Google Maps
-editProof(): String (Image Link)
1..n
1..n
Email Subsystem
Use
Use
creates
1
orderQueue/List
-Order[0...10000] : groceryOrder
-add (groceryOrder) : void
-assign (groceryOrder) : void
-remove (groceryOrder) : void
-showOnMap (groceryOrder) : Google Maps
Google Maps
Use
Use
Server

6. Object Diagram

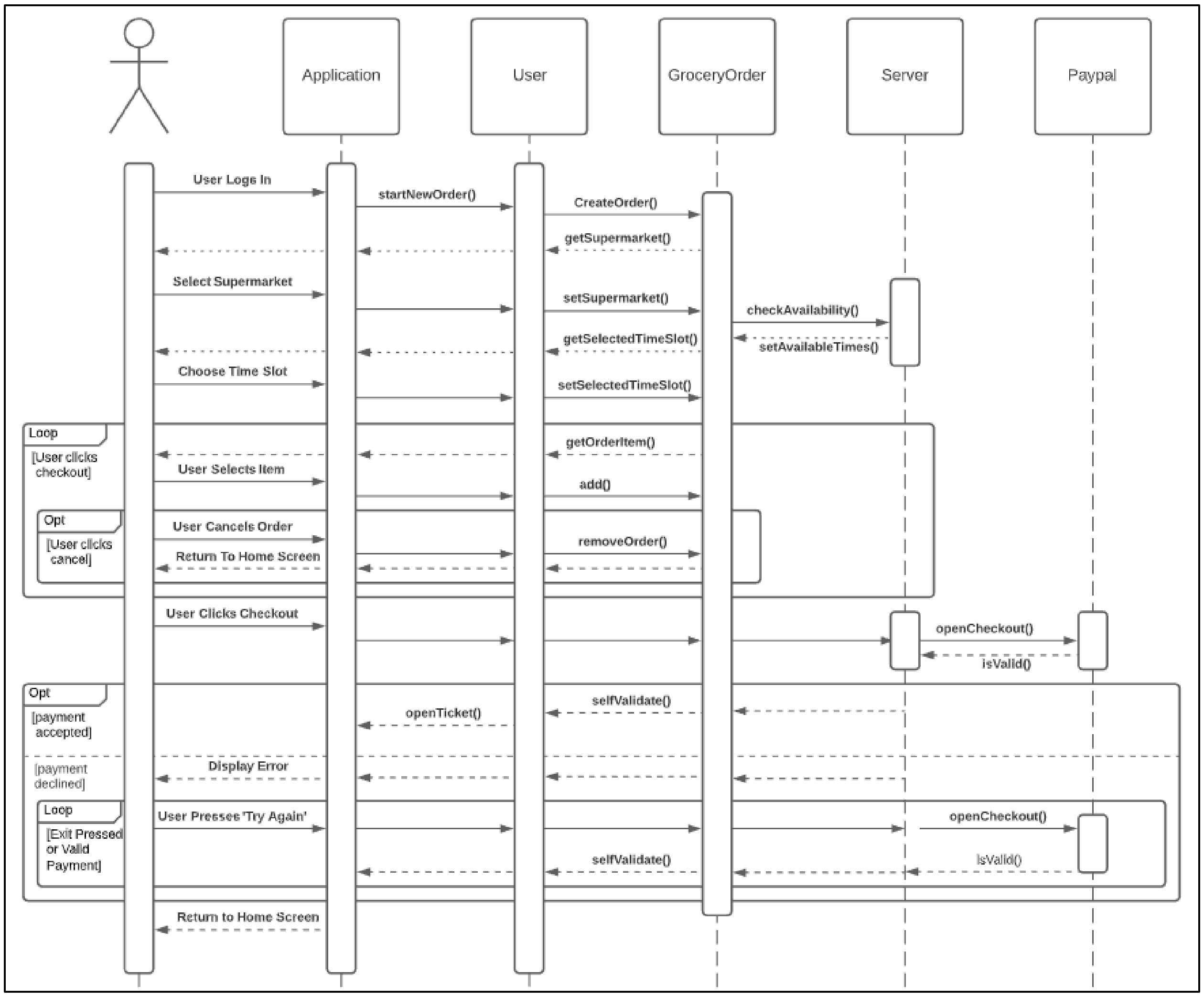


7. Sequence Diagrams

Sequence Diagram 1: Customer Creates a New Account

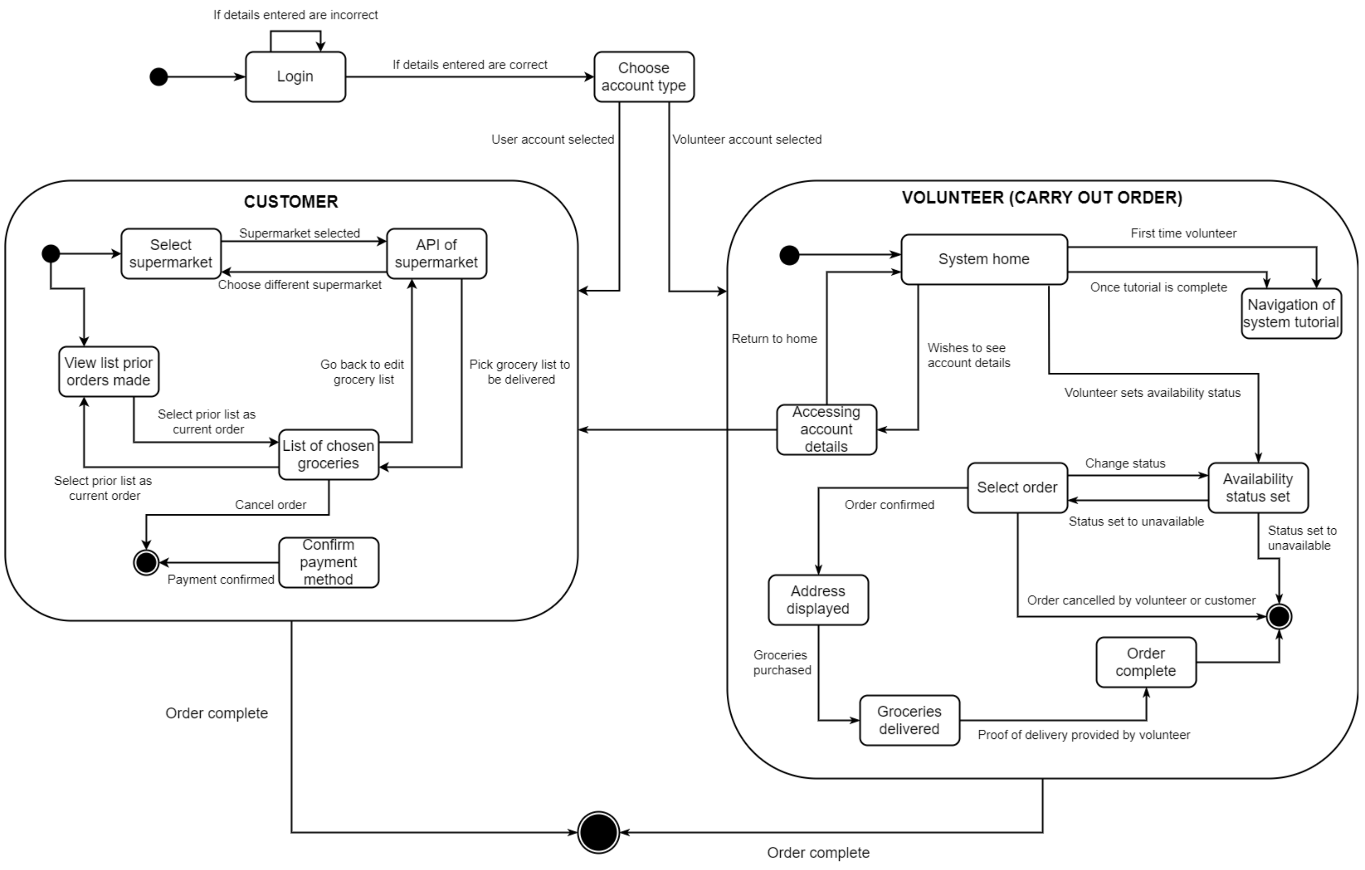


Sequence Diagram 2: Volunteer Accepts an Order

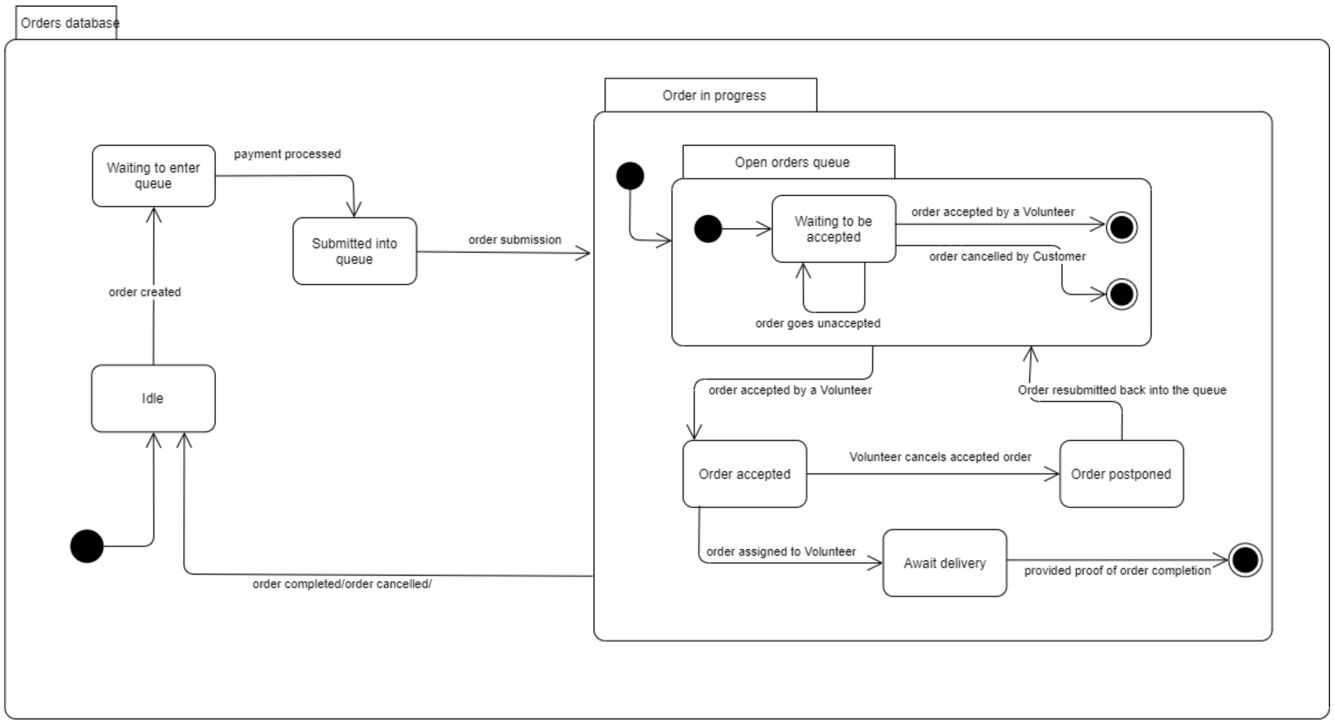


8. State Diagrams

State Diagram 1


If details entered are incorrect
Login
If details entered are correct
Choose account type
User account selected
Volunteer account selected
CUSTOMER
Select supermarket
Supermarket selected
API of supermarket
Choose different supermarket
View list prior orders made
Go back to edit grocery list
Pick grocery list to be delivered
Select prior list as current order
List of chosen groceries
Select prior list as current order
Cancel order
Confirm payment method
Payment confirmed
VOLUNTEER (CARRY OUT ORDER)
System home
First time volunteer
Once tutorial is complete
Navigation of system tutorial
Return to home
Wishes to see account details
Volunteer sets availability status
Accessing account details
Select order
Change status
Availability status set
Order confirmed
Status set to unavailable
Status set to unavailable
Address displayed
Order cancelled by volunteer or customer
Groceries purchased
Order complete
Groceries delivered
Proof of delivery provided by volunteer
Order complete
Order complete

State Diagram 2

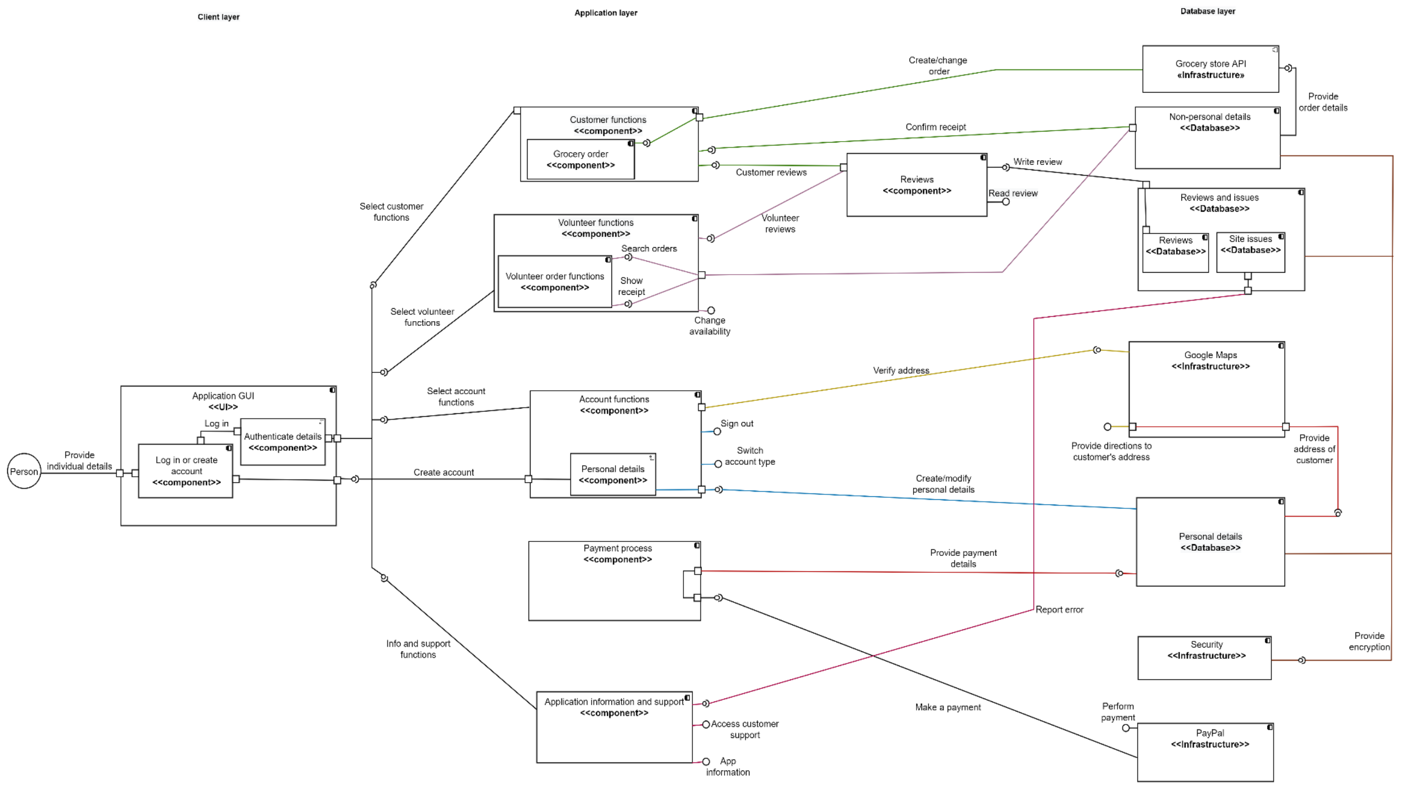


Orders database
Waiting to enter queue
payment processed
Submitted into queue
order submission
order created
Idle
order completed/order cancelled/
Order in progress
Open orders queue
Waiting to be accepted
order accepted by a Volunteer
order cancelled by Customer
order goes unaccepted
order accepted by a Volunteer
Order resubmitted back into the queue
Order accepted
Volunteer cancels accepted order
Order postponed
order assigned to Volunteer
Await delivery
provided proof of order completion

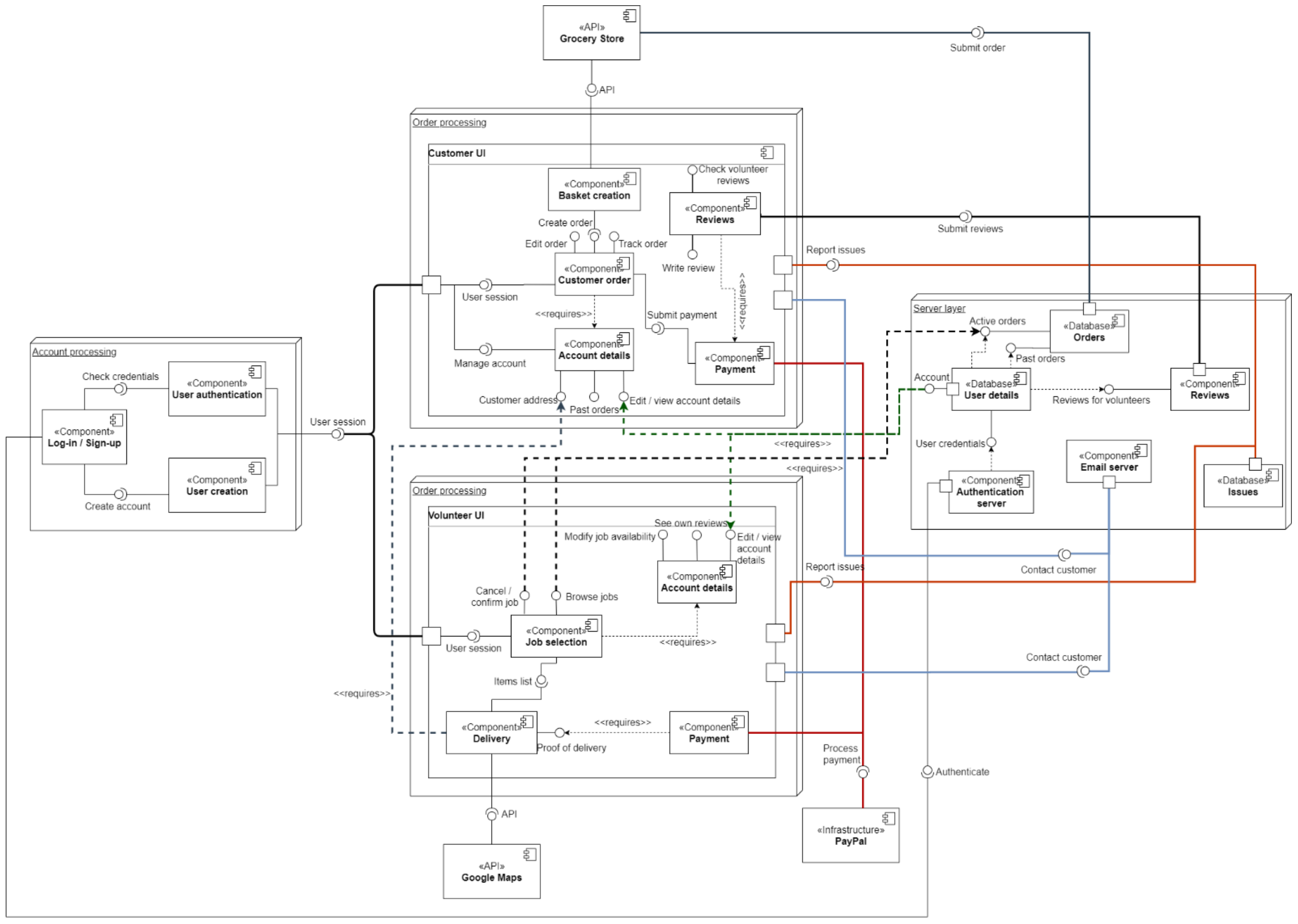
C. Software Architecture Style, Modelling and Evaluation

1. TWO Candidate Architecture styles – Component Diagrams

Component Diagram Model A

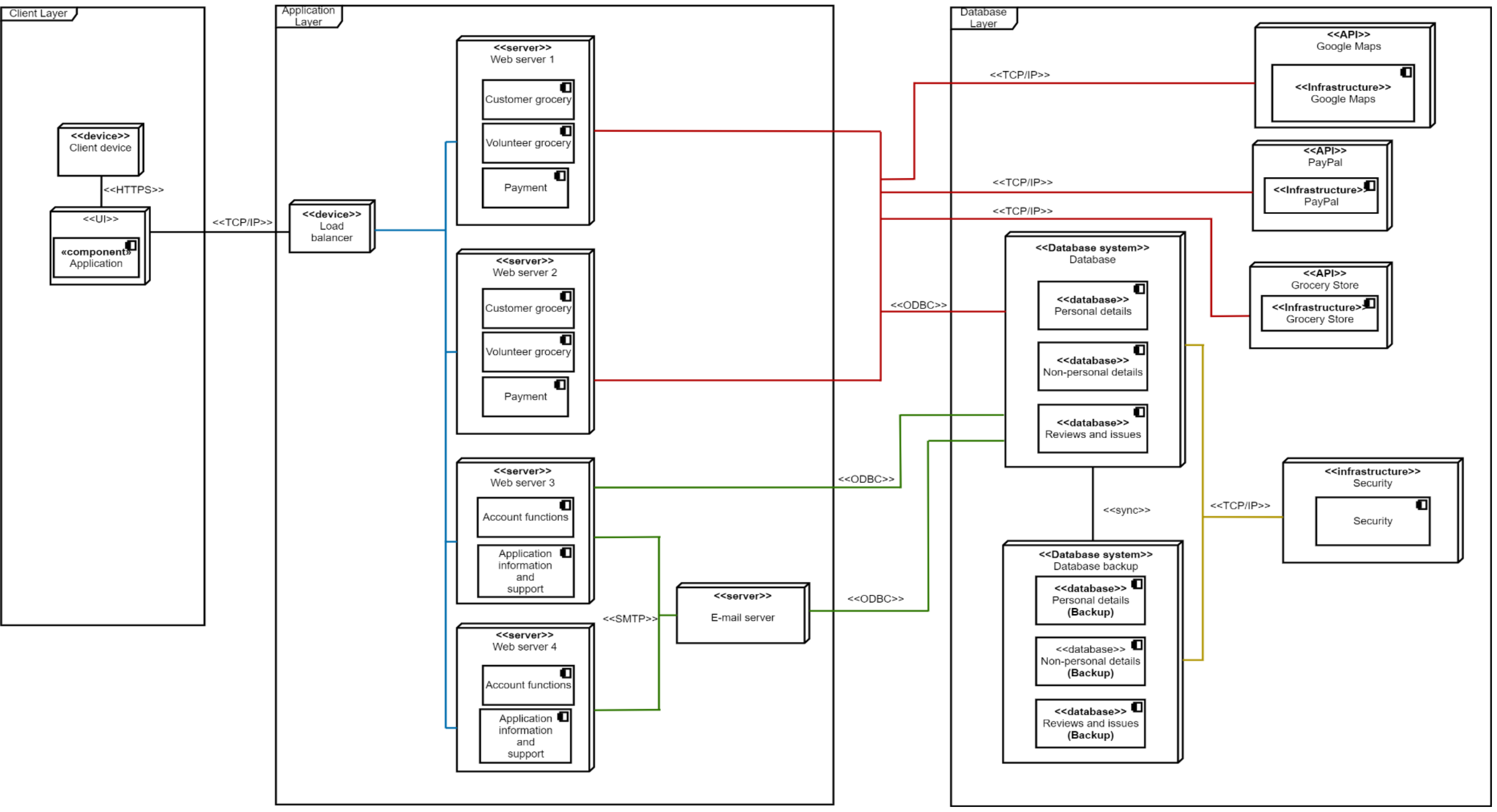

Client layer
Application layer
Database layer
Person
Provide individual details
Application GUI <<UI>>
Log in
Authenticate details <<component>>
Log in or create account <<component>>
Select customer functions
Select volunteer functions
Select account functions
Create account
Info and support functions
Customer functions <<component>>
Grocery order <<component>>
Volunteer functions <<component>>
Volunteer order functions <<component>>
Search orders
Show receipt
Change availability
Volunteer reviews
Customer reviews
Create/change order
Confirm receipt
Reviews <<component>>
Write review
Read review
Grocery store API «Infrastructure»
Provide order details
Non-personal details <<Database>>
Reviews and issues <<Database>>
Reviews <<Database>>
Site issues <<Database>>
Account functions <<component>>
Personal details <<component>>
Sign out
Switch account type
Verify address
Google Maps <<Infrastructure>>
Provide directions to customer's address
Provide address of customer
Create/modify personal details
Personal details <<Database>>
Payment process <<component>>
Provide payment details
Report error
Make a payment
Security <<Infrastructure>>
Provide encryption
Perform payment
PayPal <<Infrastructure>>
Application information and support <<component>>
Access customer support
App information

Component Diagram Model B

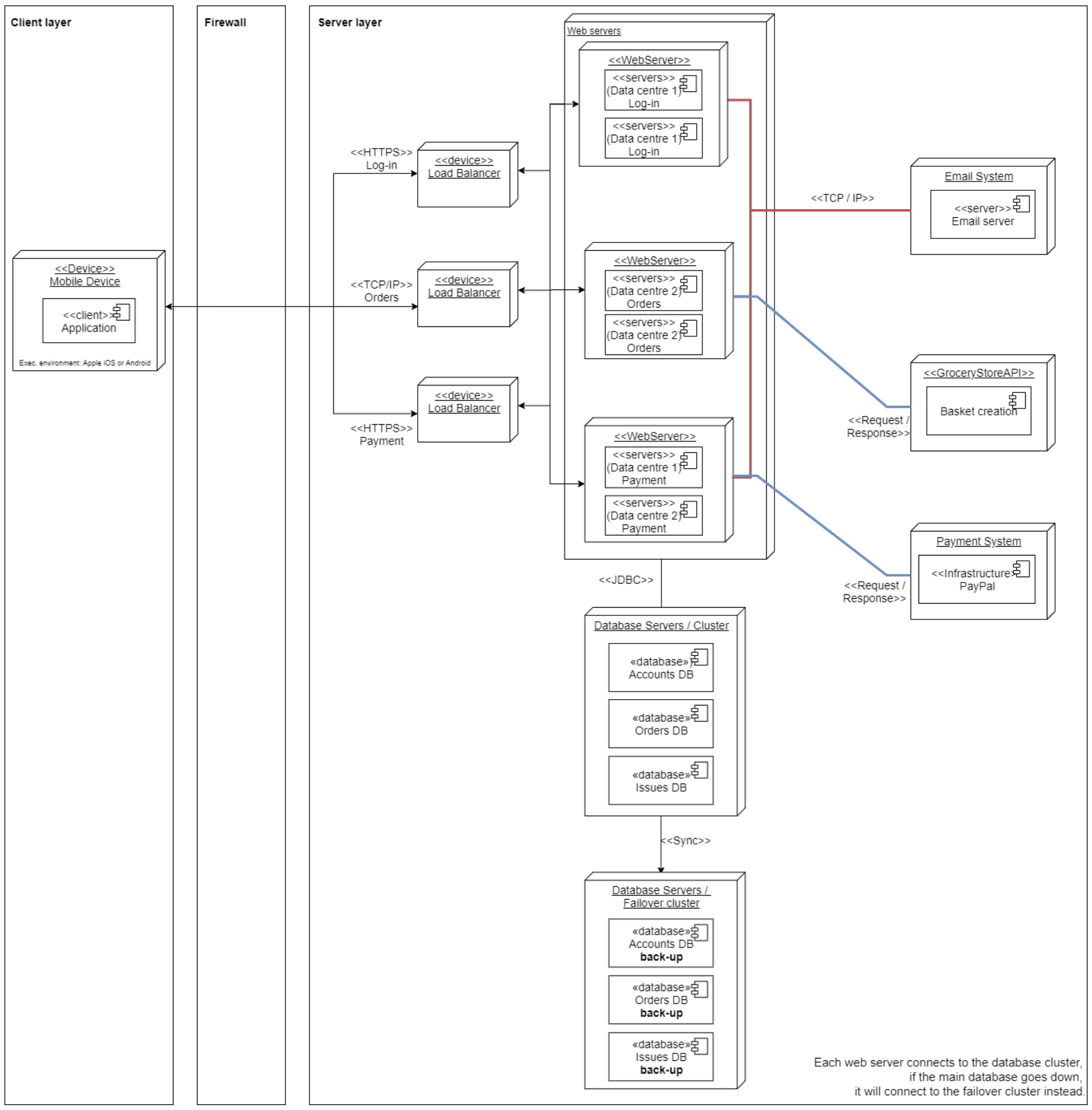


2. Deployment Diagrams

Deployment Diagram Model A



Deployment Diagram Model B



3. Comparison between the two

Model A. 3-tiered architecture

Our first approach would be a system using the 3-tier architecture. The key point for using this architecture is represented by its potential for scalability enabling distributed deployment of future servers. Nonetheless data integrity and security are also key aspects for considering this design as corrupted data and unauthorized access are removed and cut off in the middle tier.

Although this design benefits from many advantages, the complexity for implementing such an architecture is very high and might require significant work to be fully developed and deployed.

Model B. Client-server architecture

The second approach is to have the client user interfaces (customer and volunteer) directly communicating with the server layer.

Direct communication between the two allows faster communication which allows us to achieve performance as non-functional requirement. This set-up can increase speed of order processing and account processing as the web servers communicate directly with the databases via JDBC or ODBC API.

However, a trade-off is that should the number of users increase, and should the database servers have to service multiple concurrent requests, data integrity will become a concern on top of degraded performance. For example, two volunteers accepting an order within seconds of each other may start causing issues and corrupt data.

Conclusion

Considering that scalability is a non-functional requirement of our software, a 3-tiered architecture would be more appropriate. The latter offers separation of three layers which in turn makes scaling the application or data layers easier. For example, if the number of users increases, adding a User database server becomes a straight-forward process and will have no effect on the client layer.

A 2-tiered architecture does not prevent direct communication of the client with servers which can cause security issues as well (another of our non-functional requirements). As our software handles customers' personal and payment details, we want to be able to protect this information at all cost. To conclude, a 3-tiered architecture style is more suitable here.

D. Software Testing

Test Plan

At Your Step Food Delivery Application

Introduction:

The system aims to provide an interface for customers and volunteers to connect with one another. The volunteers in this context act as surrogate shoppers i.e., they take a grocery order from a self-isolating customer, go to the supermarket and purchase it, then deliver it to the customer's house. The end goal of the system is to enhance the support networks available to the general public during the COVID-19 pandemic and facilitate social distancing to prevent its spread.

Testing Objectives:

The objective of this testing plan is to assess whether or not the software has met its previously described objectives. This will be assessed through the testing of its core functionality on both the front and back end of the application to ensure it can be rolled out as soon as possible during the pandemic. This will involve testing the placing of orders by users, the ability to accept orders for volunteers, and so on. Tests will take place on up-to-date versions of iOS and Android to ensure a high degree of device compatibility.

Overview of Features to be Tested:

Functional:

1. User registering a new account.
2. User signing in.
3. User requesting a password reset.
4. Customer leaves a review of a volunteer.
5. User creating a new grocery order.
6. Volunteer accepts a user order.
7. Volunteer delivers an order.

Non-Functional:

1. Time window a volunteer can cancel an accepted order in.
2. Time it takes for a user to receive a password reset code via email.
3. Time for user created order to be added to the queue.

Features That Will Not Be Tested:

Features that are outside the scope of the system e.g. paying through the PayPal payment gateway will not be tested. Certain features that will not see as high frequency use in typical scenarios will not be tested due to the limited size of this testing program and in order to improve time to market. Examples of fringe features like this include cancelling of orders (except in vital cases), swapping of orders among volunteers, changing of account information, and more.

This testing block consists mainly of system level testing, therefore various aspects of software quality will not be tested such as user friendliness.

Approach:

Generally speaking, the front-end features that users will take advantage of will be tested using black-box testing. When testing some of the non-functional requirements of the system however, white-box unit testing will have to be employed to test things such as the time it takes for an order to be added to the queue, as the database is not something which is accessible to the end user. The testers will handle the functional requirement testing and the developers will handle the non-functional requirement testing.

The core functionality of the application can be said to be working if what was set out in the requirements as the most important parts of the system are able to execute without error. Therefore, items on the specification will be tested specifically.

Features to be Tested (Detailed):

1. <u>User registering a new account:</u>

The accounts database and back-end must only add a new account if all the requirements listed in section 1.1 of the requirements are met.

Requirements:

1.1. The system must allow an individual to register an account.

- 1.1.1. The system must ensure that the individual enters a first name.
- 1.1.2. The system must ensure that the individual enters a last name.
- 1.1.3. The system must ensure that the individual enters a valid email.
  - 1.1.3.1. The system must not allow registration with an email that has previously been registered to register a new account.
- 1.1.4. The system must ensure that an individual enters a valid password.
  - 1.1.4.1. The system must ensure that the individual enters between 8 and 50 characters, at least one uppercase letter, at least one lowercase letter, and at least one numerical digit.
  - 1.1.4.2. The system must reject the registration attempt if a password that does not satisfy all the pre-requisites is entered and display a message of what pre-requisite the password is missing.
- 1.1.5. The system must ensure that the individual enters a valid phone number.

1.1.5.1. The system must only accept a UK phone number.

1.1.5.2. The system must reject the registration attempt if a non-UK phone number is entered and display an incorrect phone number message

1.1.5.3. The system must not allow the same phone number to be used for two accounts.

1.1.6. The system must ensure that the individual enters a valid address.

1.1.6.1. The system must verify that the address is a UK-based address through Google's Maps API.

1.1.6.2. The system must reject the registration attempt if a non-UK based address is entered, and display a message saying that the application is only in service within the UK.

1.1.7. The system must ensure that the individual enters a valid form of making and receiving payments.

1.1.7.1. The system must transfer the individual to the Payflow (PayPal) payment gateway to enter the details.

1.1.7.2. The system must then set the financial details of their account with their PayPal account.

1.2. Upon satisfaction of every pre-requisite, the system must add the new registered account into the accounts database.

1.3. Upon satisfaction of every pre-requisite, the system must display a successful account registration message and direct the individual to the sign-in screen.

Testing Method:

The testing approach used to test whether or not this requirement has been met involve black-box system level testing. Testing of the system in the form that the end user sees it will help ensure a more reliable system at launch time. In the following tests it is assumed that the rest of the inputs will be valid, so we are testing only one section of the requirements at a time. For instance, if the test involves a user trying to sign up with an invalid email, all the other data fields will contain valid entries.

Test Case ID:	Test Description:	Test Steps:	Expected Result:	Actual Result:	Pass/Fail:	Test Comments:

1	Only allow registration with a valid name.	Go to registration page, enter a name and attempt to register.	Successful registration.			
2	Only allow registration with a valid and not in use email address.	Go to registration page, enter an email address and attempt to register.	Successful registration.			
3	Only allow registration with a password that meets the requirements.	Go to registration page, enter a password and attempt to register.	Successful registration.			
4	Only allow registration with a valid and not in use phone number.	Go to registration page, enter a phone number and attempt to register.	Successful registration.			
5	Only allow registration with a valid UK address that is not in use by another account.	Go to registration page, enter an address and attempt to register.	Successful registration.			
6	Only allow a user to link a valid PayPal account as a payment method.	Go to registration page and link PayPal account as payment method, then attempt to register.	Successful registration.			

7	Once an account has been successfully registered, the application should display a success message and redirect the individual to the home screen.	Register an account with valid account details.	Application redirects to homepage.			

2. <u>User signing in:</u>

The user should be able to sign into the application as a user if all of the requirements in section 2.1.1 are met.

Requirements:

2.1. The system should allow an individual to sign-in as a user (customer or volunteer).

2.1.1. The system must verify that the details entered on sign-in matches that of the details stored on the database.

2.1.1.1. If the details do not match, the system must reject the sign-in attempt by the individual and display an incorrect details message.

Testing Method:

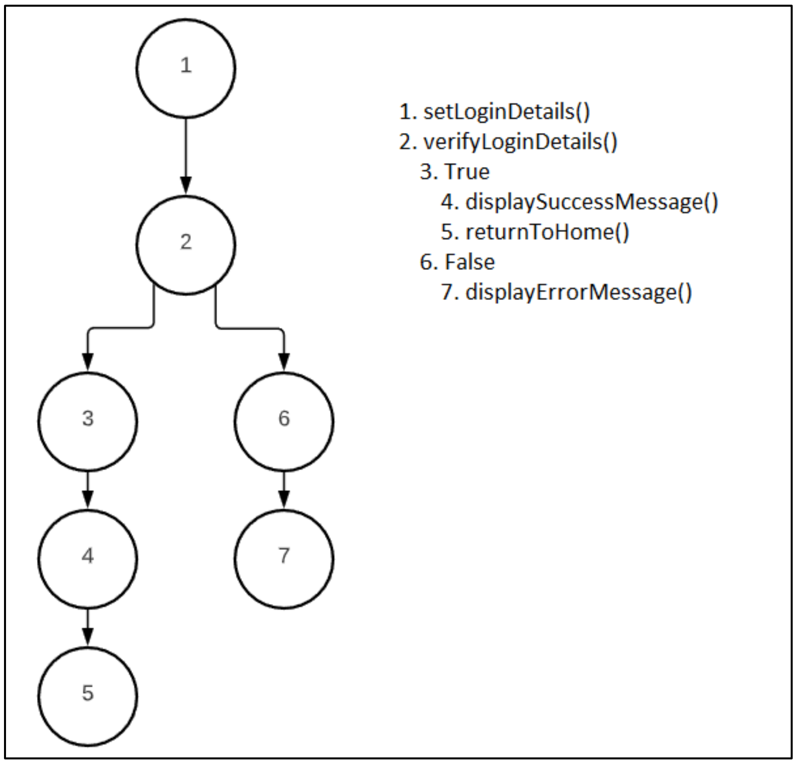
To test the requirements in section 2.1. a combination of black and white-box testing must be used. Whilst it is important to see how the system functions for the end user with regards to these requirements, some aspects are simply not testable in this format. To test 2.1.1. a white-box test must be conducted as it is not possible to see what the contents of the sign-in details on the database are through the app. As 2.1.1.1 details application level behaviour i.e. the application displays a message when incorrect details are entered, a test account will be set up with a username and password given to the testers once it has been verified in the first tests that the back-end section of the login is working as intended.

Test - Verify Details Entered on Sign-in Matches Those on Database:

In order to test whether or not the verifyLoginDetails() works, a white box-test will be constructed. The user details on the database will be known by the programmers that will perform this test therefore they will be able to verify when the details are a match or not.

Test Cases:

Test cases to ensure every branch of the control flow tree is tested.



Test Case ID:	Test Description:	Test Steps:	Expected Result:	Actual Result:	Pass/Fail:	Test Comments:
1	Login details match database.	Set login details to be the same as those on the database and then use verification method.	Calls method to display a success message, then returns use to the home screen.			
2	Login details do not match database.	Set log-in details to a value different from those on the database.	Calls method to display an unsuccessful message.			

Test - Front-End Sign-In Feature:

Once it has been established that the login system does in fact require a match with the database, it can be given to the testers to see whether it works successfully at the application level. The testers are provided with a set of login details for testing that are provided to them by the engineers and the tests will be run on an otherwise empty database. Application level testing will take place on both iOS and Android devices.

Test Case ID:	Test Description:	Test Steps:	Expected Result:	Actual Result:	Pass/Fail:	Test Comments:
1	System should only allow login with valid details i.e. they match those on record on the database.	Enter the details provided by engineers.	Successful login message, return to home screen, else return to login screen.			

3. The System Sends a Password Reset Code When Requested by a User:

The user should be able to request a password reset if section 2.1.2 has been met.

Requirements:

2. The system must display a prompt and provide a means of recovering a password if the individual has forgotten their password.

2.1.2.1. If the prompt is selected, the system must direct the individual to a secured change password page.

2.1.2.1.1. The system must provide a prompt from the individual to enter the account email associated with the forgotten password.

2.1.2.1.2. The system must provide a prompt for the individual to enter the account phone number associated with the forgotten password.

2.1.2.1.3 If the user has provided a valid email address or phone number, then an email/text containing a code to reset their password with will be sent to the specified destination.

Testing Method:

The password reset code test will be performed by the testers as part of a black-box system level test. The testers will have access to the test account set up by the developers for the previous tests. They will also have an email address registered for the account name on the company servers and a mobile phone number and phone provided too to check whether the code is sent. The following tests are devised to ensure that the email / text message is sent to the correct account when requested, and that nothing is sent if the email / text specified is not valid.

Test Cases:

Test Case ID:	Test Description:	Test Steps:	Expected Result:	Actual Result:	Pass/Fail:	Test Comments:
1	System should only send reset code to a valid email address.	Press forgotten password button on sign in screen, enter test account email as send destination.	Reset code is sent to the testing email account.			
2	System should only send reset code to a valid phone number.	Press forgotten password button on sign in screen, enter test account phone number as send destination.	Reset code is sent to the testing email account.			

4. <u>User Changes Password Using Reset Code:</u>

With a valid password reset code the user should be able to change the password of their account without being signed in as per section 2.1.2.1.5 of the requirements specification.

Requirements:

- The system must allow the user to change their password if they have a valid password reset code and the password matches the requirements set out in the account registration phase (i.e. password contains between 8 and 50 characters, at least one uppercase letter, at least one lowercase letter, and at least one numerical digit).
  - A reset code is considered valid when it matches the reset code stored on the database for the account having its password changed.
  - The system must display an error message to the user when the user tries to reset their password with an invalid reset code.
- The system must update the password variable on the accounts database for the user when a password reset is successful.

  (More details can be found in section 2.1.2.1.5 of the functional requirements but for the readers convenience the above points present a summary of them.)

Testing Method:

The password reset functionality will be tested with a combination of black and white-box testing as with the account creation tests in order to verify that it is updating the password variable in the database. There will be two main tests conducted – the first test will use a black-box test to check whether the password reset code works as intended and whether the application does what it is supposed to after a successful or unsuccessful password reset. The second test is a simple unit test that checks whether or not a successful password reset updates the database. The testers will be provided with a valid reset code send to the test email account and phone number.

Test - User Attempts to Change Password Using Reset Code:

Test Case ID:	Test Description:	Test Steps:	Expected Result:	Actual Result:	Pass/Fail:	Test Comments:
1	Password should only be changeable with a valid reset code.	Go to password reset page and enter the valid reset code with a valid password.	Change password on server, display success message, and direct individual to sign in page.			

Test – User Successfully Changes Password:

Test Case ID:	Test Description:	Test Steps:	Expected Result:	Actual Result:	Pass/Fail:	Test Comments:
1	System should update the database value of the password when a successful password change is made.	Change password of the test account, check database value for match.	Passwords are identical.			

5. User Changes Password Using Reset Code:

As outlined in the requirements specifications (2.7. & 2.8.) customers should be able to leave a review of the volunteer filling their delivery after its conclusion.

Requirements:

2.7. The system must provide a review prompt for a Customer to review and rate the service/reliability of the Volunteer that has taken their order.

> 2.7.1. The system must check and verify that the order between a Customer and a Volunteer has been completed before the Customer can review the Volunteer.
>
> 2.7.2. The system must allow the Customer to enter the number of stars the Volunteer should receive, from a range of 1 to 5 stars.
>
> 2.7.3. The system must allow the Customer to leave written feedback in the form of a text box.
>
> 2.7.4. The system should allow a Customer to review the Volunteer that completed their order after ignoring the initial review prompt.

2.8. The system must allow a Customer to view the profile of the Volunteer that has accepted their order.

> 2.8.1. The system must allow the Customer to view the ratings and reviews of the Volunteer that accepted their order.
>
> 2.8.2. The system must allow the customer to view the name and phone number of the Volunteer that accepted their order.
>
> 2.8.3. The system must not allow the customer to view the address, email, password, and email of the Volunteer that accepted their order.

Testing Method:

System level black-box testing will be used in order to ensure that the review functionality works from the end-user's perspective. Section 2.7. will be tested first, where the test account will be used to leave feedback of a test volunteer. After the first review has been left, section 2.8. will be tested by the programmers adding a number of test reviews to the Volunteer's profile which the testers should be able to view within the app.

Test 1 – Customer Leaves a Review of a Volunteer:

Test Case ID:	Test Description:	Test Steps:	Expected Result:	Actual Result:	Pass/Fail:	Test Comments:
1	System should allow the Customer to leave a text-based review of a Volunteer who delivered to them.	Go to Volunteer's profile, leave a text review.	Review is posted.			
2	System should allow the Customer to leave a star rating of a Volunteer who delivered to them.	Go to Volunteer's profile, leave a star rating.	Rating is added to the volunteers' profile.			

Test 2 – Customer Checks Reviews of a Volunteer:

Test Case ID:	Test Description:	Test Steps:	Expected Result:	Actual Result:	Pass/Fail:	Test Comments:
1	System should allow the volunteer to view their own and other user's reviews of a volunteer.	Go to test Volunteer's profile, look at reviews.	Customer can see their own and other users reviews of the Volunteer.			

2	System should allow the Customer to view Customer's star ratings of a Volunteer.	Go to test Volunteer's profile, look at star ratings.	Customer can view ratings of the Volunteer.			

6. Customer Creates a Shopping Cart:

As outlined in section 2.5.3. of the requirements, the customer should be able to place a grocery order through the application using the supermarket of their choice's (assuming it is supported by the application) API.

Requirements:

2.5.3. The system must allow a Customer to select a supermarket chain:

2.5.3.1. The system must then allow a Customer to select grocery products, add and remove them from the basket and view the total amount to be paid.

Testing Plan:

The testers will create a basic grocery list within the application from each of the supported supermarkets through a system-level black-box test.

Test 1 – Customer tests basket functionality.

Test Case ID:	Test Description:	Test Steps:	Expected Result:	Actual Result:	Pass/Fail:	Test Comments:
1	Customer can add products from the supermarket to their basket.	Go into application, start new order, select supermarket, add items to basket.	Items are added to basket.			
2	Customer can remove products from the supermarket to their basket.	Go to basket, remove items from basket.	Items are removed from basket.			
3	Application displays correct total price.	Add items to cart, manually calculate price, then check the difference.	Manually calculated price and price displayed in application are equal.			

7. <u>Volunteer Completes an Order:</u>

When a volunteer completes an order, they should be able to mark the order as completed via the application as detailed in section 3.8. of the requirements section.

Requirements:

The system must allow a Volunteer to request an order closure.

3.8.1. The system must require the Volunteer to provide proof of completion.

3.8.1.1. The system must allow the Volunteer to take and upload a picture for proof of completion.

3.8.2. The system must not be able to close an order if proof of competition is not provided by the Volunteer.

Testing Plan:

A test order will be set up by the testers using two test accounts, then the testers will mark the order as completed and upload proof of completion. The proof can be any picture (typically proof consists of a picture of the receipt, but if a junk image is uploaded then this will be used as evidence if any conflict resolution is needed). Once a picture is uploaded and the order is requested to be closed then the application should store this and wait until the customer marks the order as closed too, before changing the ticket status of the order to closed. The tests will just check that this order closing and proof uploading functionality works.

Test 1 – Volunteer Closes Order:

Test Case ID:	Test Description:	Test Steps:	Expected Result:	Actual Result:	Pass/Fail:	Test Comments:
1	Volunteer selects close order.	Select close order.	Application prompts Volunteer to upload proof.			
2	Volunteer uploads proof.	Upload image from camera roll when prompted.	Application shows that file is ready to be uploaded.			

3	Volunteer clicks complete order.	Press the complete order button.	Success message is shown, and application tells them that order will be completed when Customer confirms its status.			

Non-Functional Requirements:

1. Order Cancelation Time:

As per section 1.1. of the non-functional requirements specification, the user should be able to cancel their order up to 3 minutes after their order has been created.

1. **Usability**
    1. (**Customer Account – Order)** The system must allow a Volunteer to cancel an accepted Customer order, up to 3 minutes after an order has been accepted.

Testing Plan:

This functionality is easily tested through a simple black-box system level test. The volunteers can place an order using the testing account setup for them, then they simply have to cancel the order within 3 minutes. There will also be a test to see if the order can be cancelled after 3 minutes because this would not be intended.

Test - Order Cancellation Time:

Test Case ID:	Test Description:	Test Steps:	Expected Result:	Actual Result:	Pass/Fail:	Test Comments:
1	Cancel an order within 3 minutes of placing it.	Place an order on the app using the test Customer account, then cancel the order before 3 minutes has passed.	Order should be terminated, and ticket status changed to closed.			
2	Cancel an order after 3 minutes of placing it.	Place an order on the app using the test Customer account, then cancel the order after 3 minutes has passed.	The application should display an error message and not cancel the order.			

2. Password Reset Code Email Delivery Time:

As per section 1.4. of the non-functional requirements specification, the system must send an email containing a password reset code within 25 seconds of a user requesting one.

1. **Usability**
    1.1. (**Email and Updates – Security Code for Password Change)** The system must send the email containing the uniquely generated 6-character alphanumeric string to the provided email for a password change request, within 25 seconds of the User clocking confirm and passing the verification checks when completing the password change page.

Testing Plan:

As with the previous test this test can be done via black-box system level testing. This is ideal as it allows the testers to get an idea of the applications performance in real life usage. To test this the testers simply need to request a password reset, set a timer, and then stop the timer when the test email account receives the reset code.

Test – Password Reset Code Delivery Time:

Test Case ID:	Test Description:	Test Steps:	Expected Result:	Actual Result:	Pass/Fail:	Test Comments:
1	Tests whether the user receives a password reset code in <= 25 seconds of requesting one.	Request a password reset code within the application, start a timer, and wait until the email is delivered to the test email account.	email should be delivered in <= 25 seconds.			

3. Order Placement Time:

As stated in section 2.3. the system must place an order into the order's queue within 10 seconds of the user placing it.

2. **Efficiency (Performance)**

   2.3. The system must be able to process a Customer's created order and add it into the queue within 10 seconds.

Testing Plan:

As the order's queue is not visible to end-users, a mixed testing approach using black-box testing by testers at the system level to place the order must be used, whilst the time it takes to reach the order queue must be checked with white-box unit testing at the back end.

Test – Order Processing and Adding to Queue Time:

Test Case ID:	Test Description:	Test Steps:	Expected Result:	Actual Result:	Pass/Fail:	Test Comments:
1	Tests whether the order reaches the queue within <= 10 seconds of it being placed.	Testers use test account to place an order via the application, then programmers check how long it takes to be added to the queue.	Order should be processed and added to the queue in <= 10 seconds.			

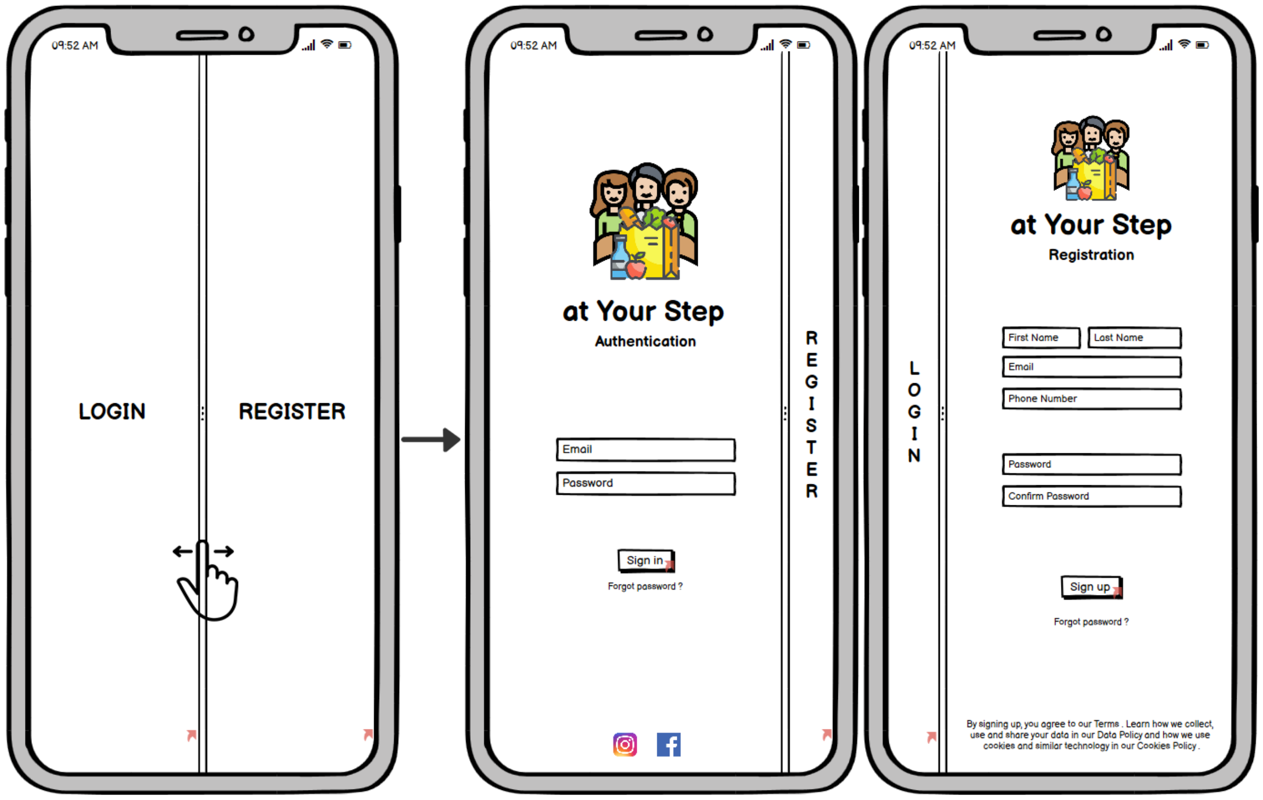
E. Usability and Prototyping

1. Prototype

Full Prototype: https://drive.google.com/file/d/1P3dI_ZWGuyY6wds0kib7ZbNANQm4Cies/view

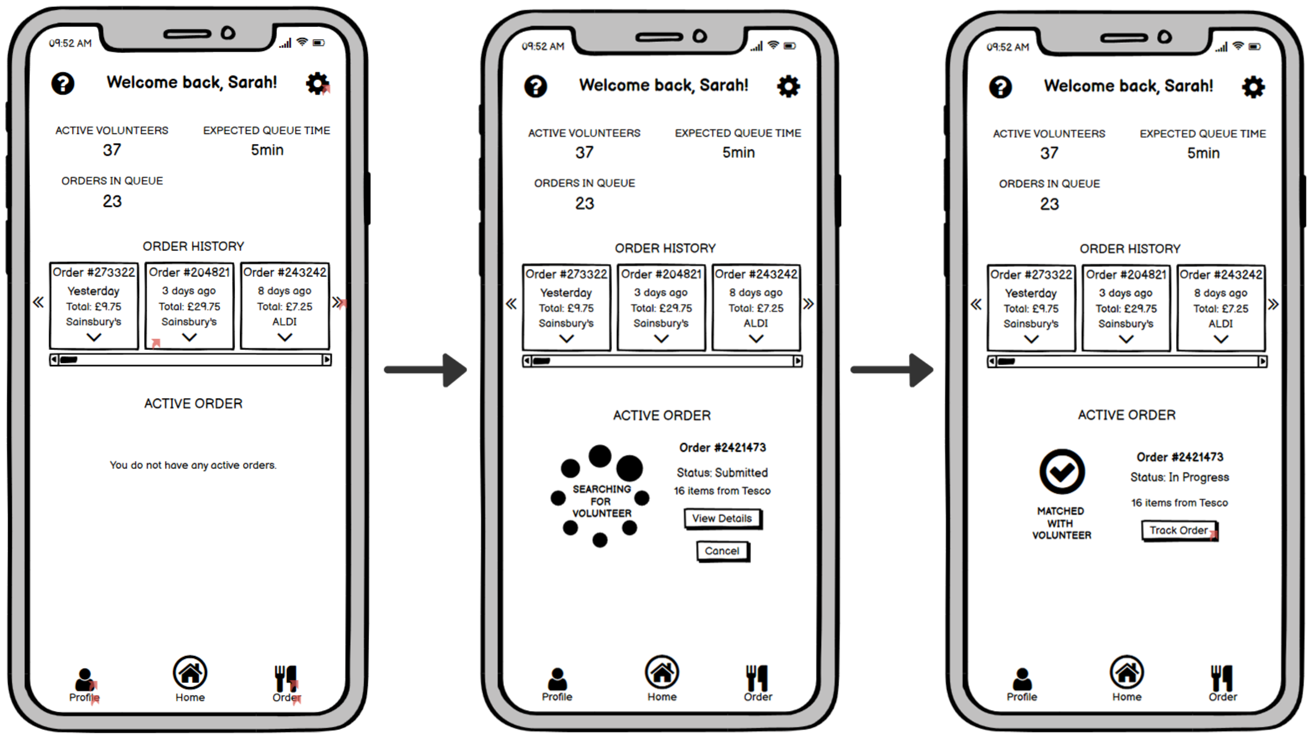
Landing Page:

When first initializing the app the user can choose to either log in or register a new account. This choice is made simple and visible by the middle splitting slider. Sliding right we reveal the login section. Swiping left we reveal the register page.



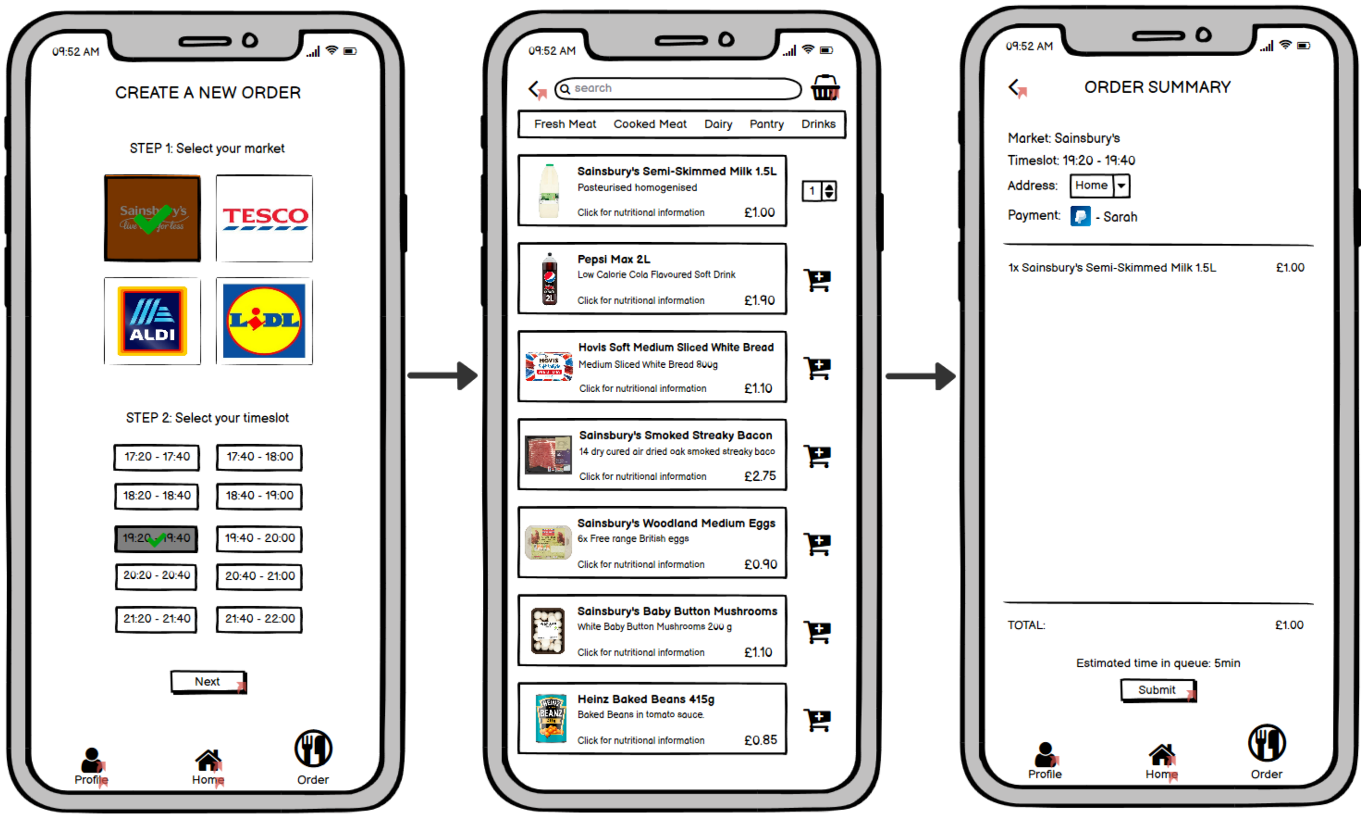
Home Screen:

The home screen was designed with the idea of offering the tools users would most of the time be looking for when booting up the app. Stats such as the expected queue time allow customers to plan their orders ahead of time or postpone ordering at a busier time. Displaying active orders on the home page is also a decision taken for ease of use and accessibility.



Order Screen

After selecting the desired market and timeslot the user can then browse every single item available in the respective shop and add it to the cart. The user can either search for an item by name or sort it through categories. A summary page is displayed before submitting an order to the system showing key details regarding the customer's order and the expected wait time in the queue.



Sainsbury's Semi-Skimmed Milk 1.5L
Pasteurised homogenised
Click for nutritional information £1.00

Pepsi Max 2L
Low Calorie Cola Flavoured Soft Drink
Click for nutritional information £1.90

Hovis Soft Medium Sliced White Bread
Medium Sliced White Bread 800g
Click for nutritional information £1.10

Sainsbury's Smoked Streaky Bacon
14 dry cured air dried oak smoked streaky baco
Click for nutritional information £2.75

Sainsbury's Woodland Medium Eggs
6x Free range British eggs
Click for nutritional information £0.90

Sainsbury's Baby Button Mushrooms
White Baby Button Mushrooms 200 g
Click for nutritional information £1.10

Heinz Baked Beans 415g
Baked Beans in tomato sauce.
Click for nutritional information £0.85

Market: Sainsbury's
Timeslot: 19:20 - 19:40
Address: Home
Payment: - Sarah

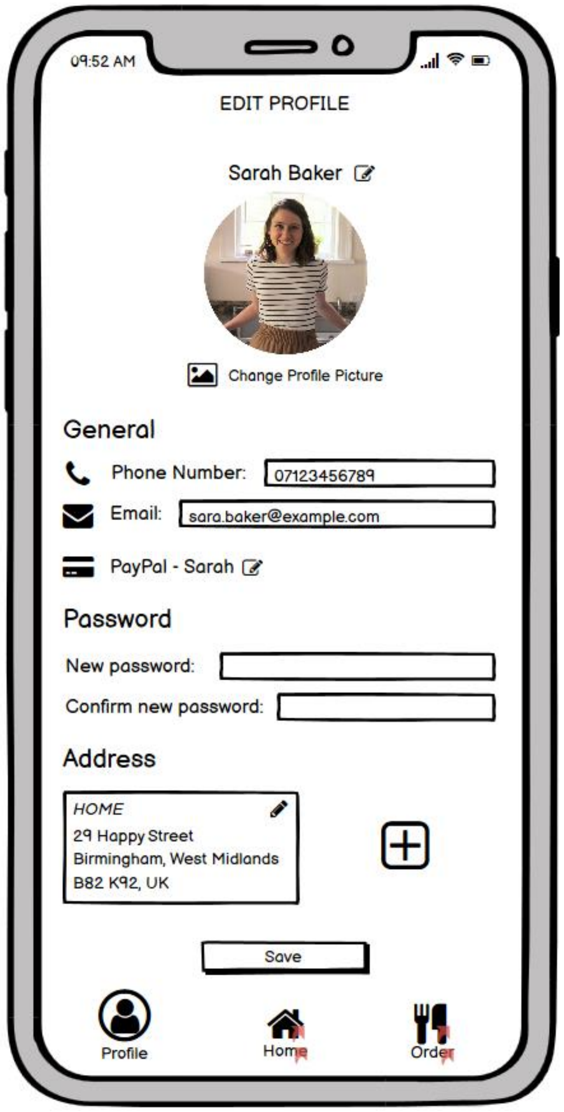
1x Sainsbury's Semi-Skimmed Milk 1.5L £1.00

TOTAL: £1.00

Estimated time in queue: 5min

Profile Screen

Designed with the scope of allowing users to edit and update their details. The noticeable difference here is that volunteers also have their star rating displayed at the top. In general, it allows users to rapidly modify key details such as their name, phone number, email etc. It also allows changing your preferred address, adding a new address or changing your payment method.



Sarah Baker

General

Phone Number: 07123456789

Email: sara.baker@example.com

PayPal - Sarah

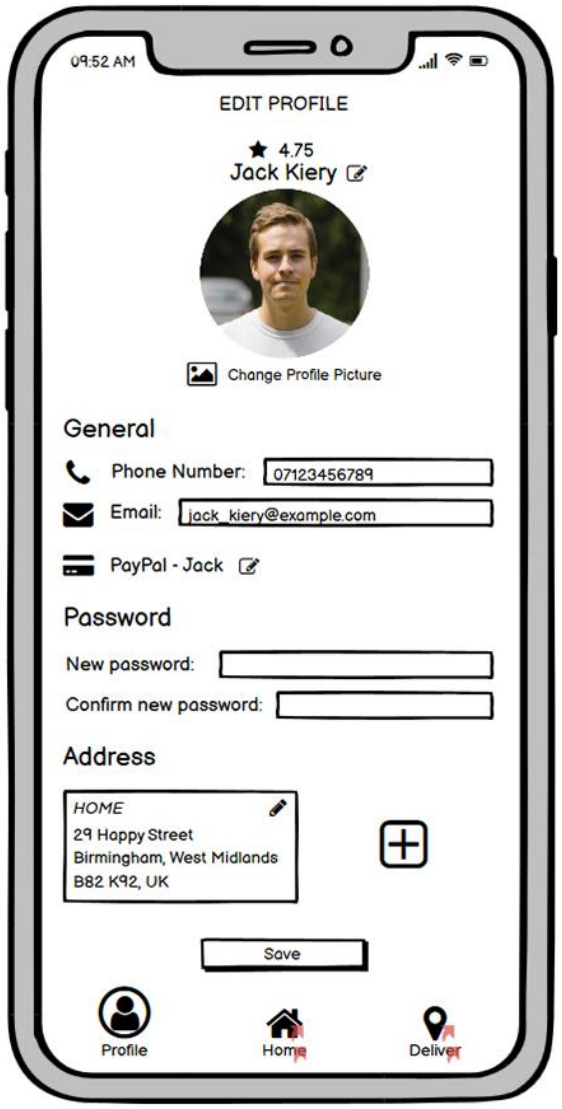
Password

New password:

Confirm new password:

Address

HOME
29 Happy Street
Birmingham, West Midlands
B82 K92, UK



★ 4.75
Jack Kiery

General

Phone Number: 07123456789

Email: jack_kiery@example.com

PayPal - Jack

Password

New password:

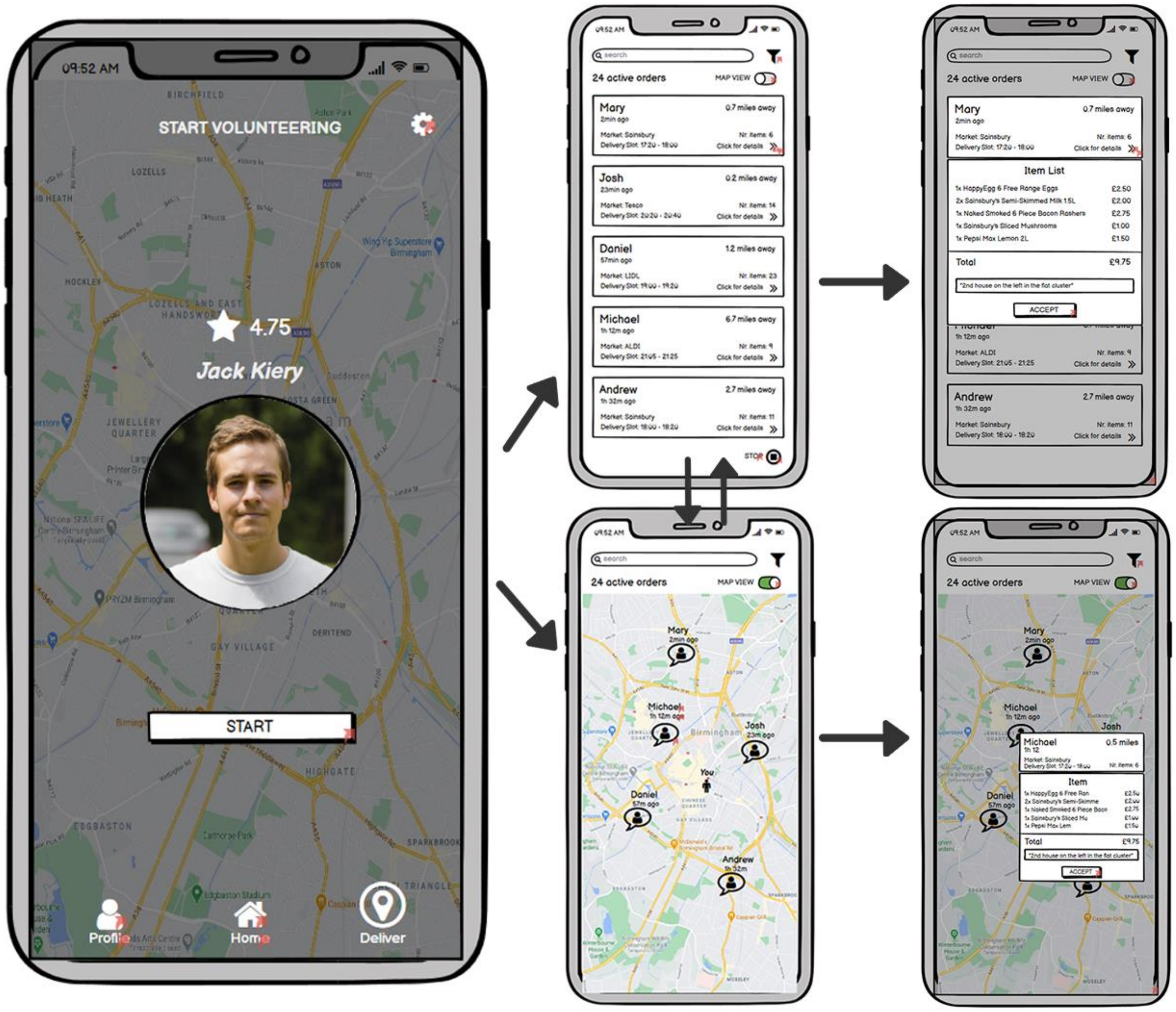
Confirm new password:

Address

HOME
29 Happy Street
Birmingham, West Midlands
B82 K92, UK

Deliver Page

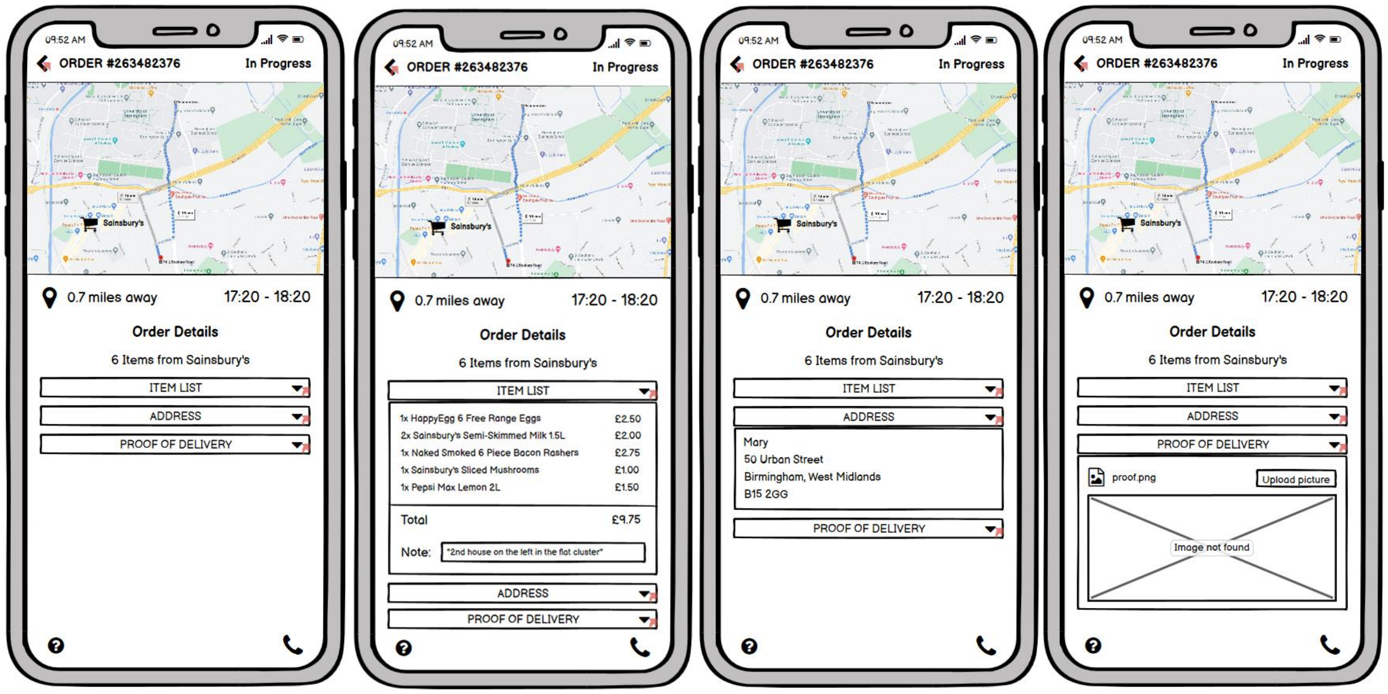
Used by volunteers to find orders in their area. A simple on/off button allows the volunteer to switch between a normal list view and a map view which would offer a better perspective at how far the orders are from the volunteer's current position. Each card in the list offers a quick summary of each order. Tapping the card offers more details regarding the items that the user wishes to purchase.



09:52 AM
START VOLUNTEERING
4.75
Jack Kiery
START
Profile
Home
Deliver
search
24 active orders
MAP VIEW
Mary 2min ago 0.7 miles away
Market: Sainsbury Nr. items: 6
Delivery Slot: 17:20 - 18:00 Click for details
Josh 23min ago 0.2 miles away
Market: Tesco Nr. items: 14
Delivery Slot: 20:20 - 20:40 Click for details
Daniel 57min ago 1.2 miles away
Market: LIDL Nr. items: 23
Delivery Slot: 19:00 - 19:20 Click for details
Michael 1h 12m ago 6.7 miles away
Market: ALDI Nr. items: 9
Delivery Slot: 21:05 - 21:25 Click for details
Andrew 1h 32m ago 2.7 miles away
Market: Sainsbury Nr. items: 11
Delivery Slot: 18:00 - 18:20 Click for details
STOP
Item List
1x HappyEgg 6 Free Range Eggs £2.50
2x Sainsbury's Semi-Skimmed Milk 1.5L £2.00
1x Naked Smoked 6 Piece Bacon Rashers £2.75
1x Sainsbury's Sliced Mushrooms £1.00
1x Pepsi Max Lemon 2L £1.50
Total £9.75
"2nd house on the left in the flat cluster"
ACCEPT
You
Birmingham

Active Order – Volunteer:

This is the page used by the volunteers to find their way to the customer and review key details regarding the order. The map should also display the nearest store chosen by the customer. The proof of delivery section allows the volunteer to upload the receipt after delivering the groceries. The information is kept in little dropdown sections in order to maximize space and keep everything tidy.



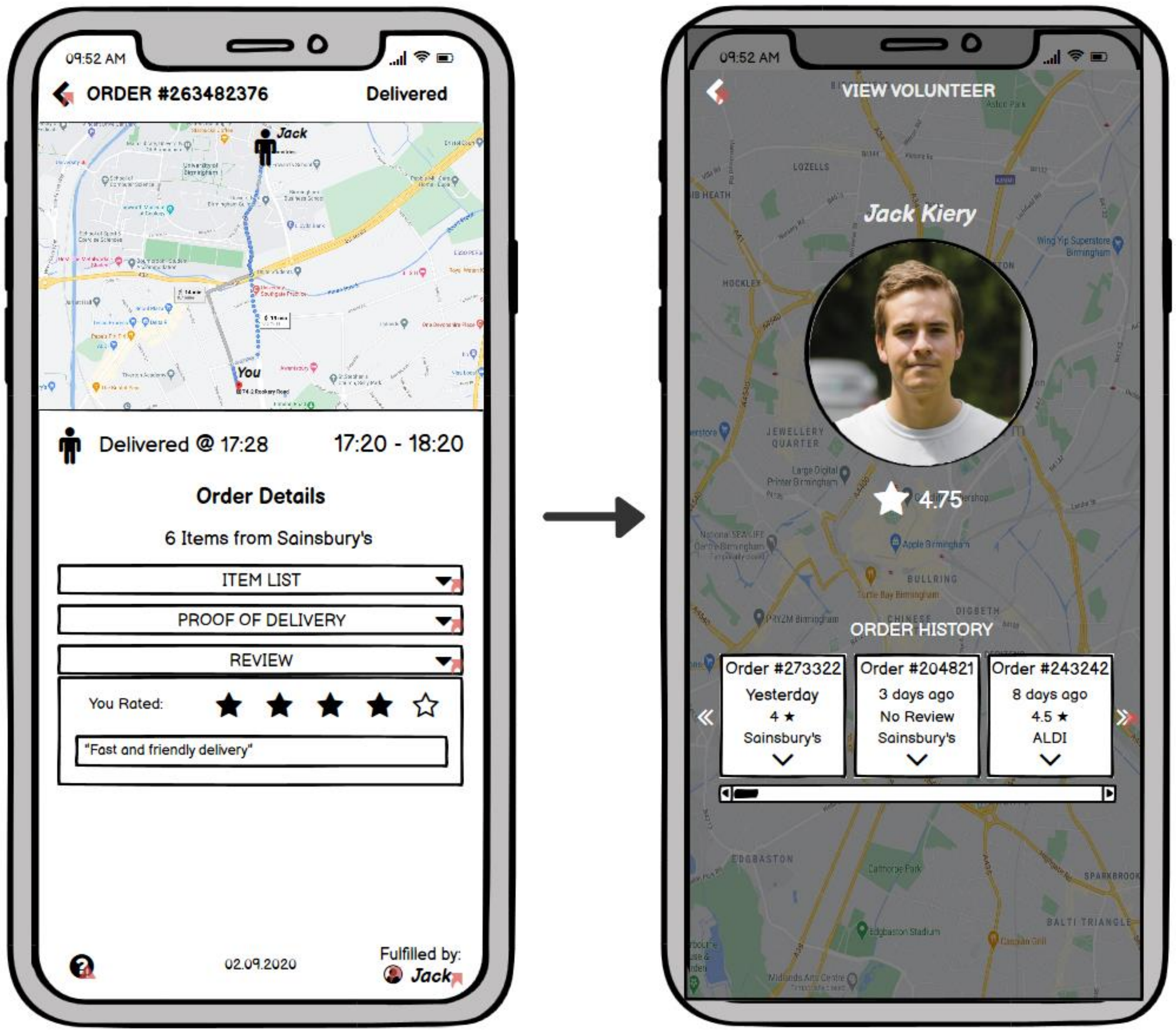
1x HappyEgg 6 Free Range Eggs	£2.50
2x Sainsbury's Semi-Skimmed Milk 1.5L	£2.00
1x Naked Smoked 6 Piece Bacon Rashers	£2.75
1x Sainsbury's Sliced Mushrooms	£1.00
1x Pepsi Max Lemon 2L	£1.50
Total	£9.75

Note: "2nd house on the left in the flat cluster"

Mary
50 Urban Street
Birmingham, West Midlands
B15 2GG

Active/Finished Order – Customer:

The customer track order page is similar to the volunteer one above, however, there is the option to view the volunteer's profile with their past orders and review score. The option of writing/editing a review for the volunteer also becomes available when the order has been delivered.



2. Video recording

Google Drive Link:

https://drive.google.com/file/d/198zMhb4sFwmhz20Zllm4IQ35r-TfExLg/view?usp=sharing

F. Ethics and Professional Practice

In making decisions about our application, we need to consider the impact on our stakeholders, both the service users who are likely to be elderly or disabled and may be especially vulnerable to being scammed and manipulated and the volunteers who are helping out of kindness and may be at risk of being taken advantage of.

Under principle 1.2 of the ACM Code of Ethics, we would need to mitigate all risks of potential harm. This may come from accidental exposure to COVID to either party, and we would need to ensure both parties are fully educated on proper procedure for handing over shopping, to avoid exposure.

Under principle 1.4 of the code of ethics, we also need to design our application with inclusiveness and accessibility in mind. This is especially relevant since it is likely that our service users may be elderly or disabled and have specific accessibility requirements or be unfamiliar with technology. We may want to consider adding accessibility features such as a large text option, text to speech functionality, or compatibility with alternative input devices. We would also need to be careful that our app is laid out clearly so those who are not comfortable with technology can understand how to use it easily.

We also want to protect our service users from discrimination by the volunteers, since volunteers have a choice of whether to accept orders, there is potential for them to not accept orders on the basis of race etc. To avoid this, we may want to avoid displaying information that may enable this (e.g. a photo or the users name may allow racial discrimination by the volunteer)

We will also be holding personal data about volunteers, and so under principle 1.6 we should be careful to collect only the minimum amount of data and use it only for the purposes stated when it was collected. We should also ensure compliance with the laws of data protection.