Project Report Document

for

Food Donation Application (DON8)

Version 1.0



Prepared by

Group 1 (Monday Batch: 8:30am-11:20am)

Arun Reddy Nalla (Student Id: - 110088379)

Dhruv Vinod Nair (Student Id: - 110088372)

Jaydeep Dharamsey (Student Id: - 110088507)

Kalyan Venkatesh Poludasu(Student Id: - 110089120)

Seyedeh Sara Salamati Taba (Student Id: - 110084098)

Veera Venkata Bharat Kumar Vayitla(Student Id: - 110088432)

31st July 2022

Table of Contents

Table of Contents ii

List of Tables iii

List of Figures………………………………………………………………………………………..iv

1. Introduction 1

1.1 Purpose 1

1.2 Document Conventions 1

1.3 Intended Audience and Reading Suggestions 1

1.4 Product Scope 1

1.5 Market Research 2

2. Overall Description 7

2.1 Product Perspective 7

2.2 Product Functions 7

2.3 User Classes and Characteristics 8

2.4 Design and Implementation Constraints 9

2.5 Assumptions and Dependencies 9

3. External Interface Requirements 10

3.1 User Interfaces 10

3.2 Hardware Interfaces 10

3.3 Software Interfaces 11

3.4 Communications Interfaces 11

3.5 Technology Justification 11

4. System Features 12

4.1 User Entrance 12

4.2 Profile 12

4.3 Product 13

4.4 Homepage 13

5. Other Nonfunctional Requirements 14

5.1 Performance Requirements 14

5.2 Safety Requirements 14

5.3 Security Requirements 14

5.4 Software Quality Attributes 14

6. Quality Assurance Plan 15

6.1 Introduction 15

6.2 Testing Strategy 15

6.3 Code Maintenance Platform 24

6.4 Design Pattern 24

6.5 Project Planning 32

6.6 Risk Analysis 14

6.7 Cost 38

7. User Manual 40

8. Pros and Cons of your solution 41

9. Difficulties faced during the project 41

10.References 42

|  |  |  |
| --- | --- | --- |
|  | **LIST OF TABLES** | **Page No.** |
| Table 1.5.1 | Comparison between all the existing applications | 5 |
| Table 6.5.1 | Sprint 1 planning | 34 |
| Table 6.5.2 | Sprint 2 planning | 34 |
| Table 6.5.3 | Sprint 3 planning | 35 |
| Table 6.5.4 | Sprint 4 planning | 36 |
| Table 6.5.5 | Sprint 5 planning | 37 |
| Table 6.6.1 | Risk Analysis | 37 |
| Table 6.7.1 | Estimated Salary for each role | 38 |
| Table 6.7.2 | Experience and Role in the Project | 38 |
| Table 6.7.3 | Estimated Time Taken for completing the project is 4 Weeks. Cost analysis for each developer for the entire project | 39 |
| Table 6.7.4 | Cost analysis for each Scrum Master for the entire project | 39 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  | **LIST OF FIGURES** | PAGE NO |
| Fig 1.5.1 | Canadian Fast-Food market | 2 |
| Fig 1.5.2 | Canada's fast food industry revenue in 2021 | 3 |
| Fig 2.2.1 | Sequential Diagram (Buyer/Customer) | 7 |
| Fig 2.2.2 | Sequential Diagram (Seller/Donator) | 8 |
| Fig 2.3.1 | Use case diagram | 8 |
| Fig 2.4.1 | ER Diagram | 9 |
| Fig 3.1 | UI design | 10 |
| Fig 6.2.1 | Unit testing | 15 |
| Fig 6.2.2 | Unit testing Results | 16 |
| Fig 6.2.3 | Integration testing Email id | 17 |
| Fig 6.2.4 | Integration testing otp verification | 17 |
| Fig 6.2.5 | Integration testing sign up verification | 18 |
| Fig 6.2.6 | Integration testing Sign in verification | 18 |
| Fig 6.2.7 | Integration testing address verification | 19 |
| Fig 6.2.8 | Integration testing get address by id verification | 19 |
| Fig 6.2.9 | Integration testing add product verification | 20 |
| Fig 6.2.10 | Integration testing Get product by productid verification | 20 |
| Fig 6.2.11 | Integration testing Get all products by user verification | 21 |
| Fig 6.2.12 | Integration testing Forget Password OTP generation verification | 21 |
| Fig 6.2.13 | Performance testing – Thread group test | 22 |
| Fig 6.2.14 | Performance testing - HTTP request in the Thread Group | 22 |
| Fig 6.2.15 | Performance testing - View Result Tree | 23 |
| Fig 6.2.16 | Performance testing - response data | 23 |
| Fig 6.4.1 | Builder pattern -1 | 24 |
| Fig 6.4.2 | Builder pattern -2 | 25 |
| Fig 6.4.3 | Prototype pattern -1 | 25 |
| Fig 6.4.4 | Prototype pattern -2 | 26 |
| Fig 6.4.5 | Composite pattern -1 | 26 |
| Fig 6.4.5 | Composite pattern -2 | 27 |
| Fig 6.4.6 | Decorator pattern -1 | 27 |
| Fig 6.4.7 | Decorator pattern -2 | 28 |
| Fig 6.4.8 | Mediator pattern -1 | 28 |
| Fig 6.4.9 | Mediator pattern -2 | 29 |
| Fig 6.4.10 | Mediator pattern -3 | 29 |
| Fig 6.4.11 | Mediator pattern -4 | 30 |
| Fig 6.4.12 | Chain of Responsibility pattern -1 | 30 |
| Fig 6.4.13 | Chain of Responsibility pattern -2 | 31 |
| Fig 6.4.14 | Chain of Responsibility pattern -3 | 31 |
|  |  |  |
|  |  |  |
|  |  |  |

# Introduction

## Purpose

The purpose of this document is to present a detailed description of the project “**food-sharing that would reduce the wastage of food**”. It will explain the purpose and features of the software, what the software will do and the designing of Software. This document is intended for users of the software and potential developers.

## Document Conventions

This document was created based on the IEEE template for System Requirement Specification Documents. The document is prepared using Microsoft Word and has used the font type 'Times New Roman'. The fixed font size that has been used to type this document is 12pt with 1.5 line spacing. It has used the bold property to set the headings of the document.

## Intended Audience and Reading Suggestions

* Developers who are working on the project for further development or fixing existing bugs.
* Development and operation team of the company, who can use this as a reference to use and integrate these services with future products
* Typical Users, such as Restaurants, Food Bank, Universities and all the Government Offices who want to understand and get knowledge about the services to get the most value out of this service.

## Product Scope

The National Zero Waste Council performed study on household food waste in Canada in 2022, with astounding results.

* 63 percent of the food thrown out in Canada might have been consumed.
* That equates to 140 kilos of wasted food every year for the average Canadian household — at a cost of more than $1,300 per year!
* This equates to about 2.3 million tonnes of edible food wasted each year in Canada, costing Canadians more than $20 billion!
* Food is wasted in many forms, however in Canada, the most discarded foods by weight are:
* 30 percent vegetables, 15 percent fruit, and 13% leftovers
* 9 percent for bakery and bread
* Dairy and eggs account for 7% of the total.

The major purpose of this project is to develop a food-sharing application that would decrease food waste that would otherwise be thrown away. It accomplishes this by bringing together those who have extra food that is about to expire with people who are in desperate need of food.

Surplus food might be donated by people or businesses such as grocery stores, restaurants, and non-profit organizations. People would be able to sell/donate food that was about to be thrown out at a discounted price/for free as part of our effort, preventing it from going to waste.

## Market Research

The foodservice market in Canada is divided into four categories: full-service restaurants, quick-service restaurants, self-service restaurants, cafes, and bars, and 100% home delivery/takeaway stores (Chained Restaurants and Independent Restaurants). For the following segments, the study provides market size and projections in value (USD million).[1]

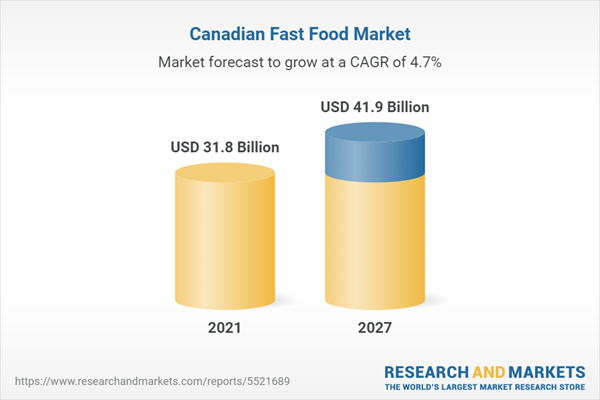


Fig 1.5.1 Canadian Fast-Food Market

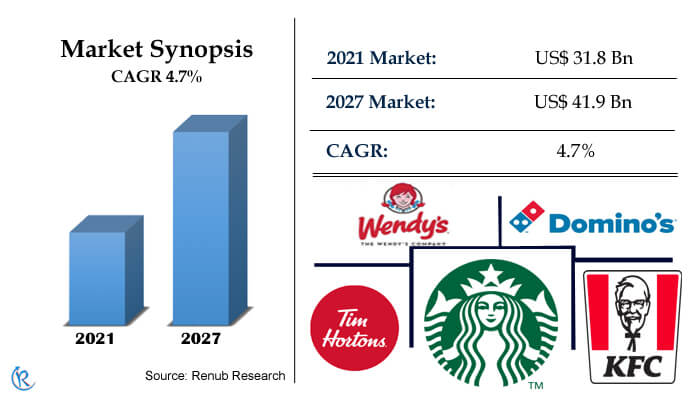
By 2027, the fast-food market in Canada is projected to generate US$41.9 billion. The Canadian food business has evolved and altered throughout the years in response to consumer demands and habits. As a result, fast food has become more and more popular throughout time in both established and emerging countries. This acceptance is mostly driven by its flavour and speedy availability. Food made and served in independent fast-food restaurants, street sellers, or chains like Tim Hortons, Subway, Starbucks, McDonald's, Dairy Queen, KFC, Domino's, Pizza Hut, Wendy's, and Burger King are all considered to provide fast food. It serves as an alternative to homemade food. In 2020, the average Canadian will consume 6.3 percent of his daily calories from fast food, according to BC Food Web.

**Impact of COVID-19 on Canadian Fast-Food Industry**

Due of the increasing instances and concern over catching the virus via food, the COVID-19 pandemic had a negative impact on Canada's fast-food business. In addition, the revenue-generating effects of the COVID-19 epidemic have been seen in the fast-food business owing to operational interruptions in the hotel and food and beverage industries as well as supply chain disruption. The market expansion during the pandemic was influenced by a higher degree of awareness regarding the consumption of healthy foods. In spite of everything, the market will recover its demand after the pandemic due to an increase in fast food franchises and rising interest in online meal delivery.

**Canada's fast-food industry generated $31.8 billion in revenue in 2021.**

Fig 1.5.2 Canada's fast food industry revenue in 2021

The market for plant-based foods is expanding as a result of rising health awareness among consumers and rising demand for wholesome meals. The rise in the proportion of working women, which is caused by their lack of time for meal preparation, the expansion of a number of fast food outlets, which makes it simple to get fast food, as well as technologically advanced ordering methods, are the main drivers of the Canadian fast food market. Additionally, increased exposure to international cuisine through media and travel will accelerate the growth of this sector.[2]

**The most popular fast food in Canada is pizza and burgers/sandwiches.**

Depending on the kind, the market is dominated by pizza, spaghetti, burgers, chicken, seafood, Asian, and Latin American cuisine. Burgers and sandwiches have always been the most popular fast-food item among consumers because of the tastes and ingredients that are used to make the cuisine. The abundance of burger alternatives, including foods with protein options like venison, beef, and many more, can be credited for the rising demand for burgers and sandwiches. Additionally, a number of restaurant owners are experimenting with various meatloaf, cheese, bacon, and onion combinations.

Nevertheless, pizza and pasta continue to be the most popular fast food in Canada, particularly for takeout and delivery, which is attributable to increased customer demand for taste and preferences. It is expected to expand at a tremendous rate during the forecast of the Canadian fast-food market.

**Consumers** **in** **Canada** **choose** **quick** **service** **restaurants** **more** **than** **any** **other** **type** **of** **restaurant**.

End-user sectors including Quick Service Restaurants (QSR), street vendors, home delivery, cafés, and bars, among others, frequently employ fast food. Customers in Canada like quick service restaurants the most, which drives up demand for healthy, enticing meals at reasonable prices. As a result, there is a growing need for quick-service restaurants. The home delivery option has seen significant increase throughout the epidemic. Because most people prefer to shop online while indoors. According to the report, from 2021 to 2027, the Canada Fast Food Industry is projected to grow at a CAGR of 4.7 percent.

**Which is the biggest fast food chain in Canada?**

The largest fast-food chain in Canada is Tim Hortons. With 3,148 sites, sandwich company Subway comes in second.

Since McDonald's and Starbucks generate more income than Subway does, having more fast food stores does not necessarily translate to better sales.

**In which Canadian cities can you find the most fast food restaurants?**

https://www.scrapehero.com/top-fast-food-chains-in-canada/

Toronto, Calgary, and Edmonton are the cities with the most fast food restaurants in Canada, each having 758 locations (475).

Compared to other provinces, Ontario has the most fast food establishments. Six of the fifteen cities in the aforementioned chart are in Ontario.

**Competitive** **Landscape**

Starbucks Corp., Domino's Pizza Inc., The Wendy's Co., Kentucky Fried Chicken (KFC), Tim Hortons, Restaurant Brands International Inc., A&W Food Services of Canada Inc., MTY Food Group Inc., Papa John's International Inc., and Performance Food Group Company are the major players in the Canadian fast food market. With well-known brands like KFC and Pizza Hut, Yum! Brands, one of the biggest quick-service providers in Canada, is launching extensive growth plans across the nation to successfully establish its presence in key locations including Calgary, Montreal, Toronto, Vancouver, Halifax, Ottawa, and Edmonton. The market expansion of the quick-service restaurant in Canada is also boosted by keeping the taste and use of healthy, natural components.

**Exciting Application that works in Food-Sharing**

* The main aim of developing this food sharing app is to fight food waste by simply sharing it with those who need food.
* The app is responsible for saving money for merchants and wastage of food, which will contribute for a better tomorrow and fill the hunger for the need.
* The app connects merchants to consumers to sell fresh surplus food that would otherwise end up in the trash. Any individual can easily track, organize, and manage food to be shared.
* Search the map to find a restaurant, cafe, or shop near you that has unsold food.
* Buy the food through the app.
* Collect the food at the present time and enjoy it knowing that you have saved food and filled up your belly too.

**List of Main Features and Elements**

* The chart below shows the comparison of 5 food donation app providers by features.
* When choosing a donation App, the user will make sure whether there are filter preferences options as it is easy for a user if there are filter preferences to sort out the food items and location.
* It should be easy and flexible to use.
* Usually, buyers prefer the app that has a payment gateway to sell their food.
* Whereas sellers tend to prefer apps which offer food for free.
* Usually both types of users prefer to use the donation app which is having contact service.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Application** | **Logo** | **Free Trial** | **Filter preferences** | **Contact us** | **User Friendly** | **Payment** | **Downloads** |
| Too Good To Go |  | Y | Y | Y | Y | Card | 10M+ |
| Olio |  | Y | Y | Y | Y | Free | 5M+ |
| No Waste |  | Y | N | N | N | Free | 10K+ |
| FoodHero |  | Y | Y | Y | Y | Card | 100K+ |
| Karma |  | Y | Y | N | N | Card | 500K+ |

Table 1.5.1: Comparison between all the existing applications

**Too good to go**:

[**https://toogoodtogo.org/en/**](https://toogoodtogo.org/en/)

* The app is well organized, and the functionality is good and easy to use.
* It is exceptionally fair value to get food from local restaurants that would otherwise get thrown out.
* The location service needs to be improved. The sellers should not be allowed to change the time before collection
* Not sure which food items will be there to collect in the bag.

**Olio**:

[**https://olioex.com/**](https://olioex.com/)

* This app contains karma points system where you can receive points for giving or sharing food items and can earn badges.
* The communication window and push notifications need to be fixed as notifications are not received after the order has been placed.

**NoWaste:**

[**https://www.nowasteapp.com/**](https://www.nowasteapp.com/)

* Not user friendly and lacks many of the features and contains bugs and glitches.

**FoodHero:**

[**https://foodhero.com/**](https://foodhero.com/)

* This app is very user friendly and contains most of the functionalities that users need and is easy to navigate.
* Value for money and good discounts by earning points.
* A few small glitches freeze the screen with no response.

**Karma:**

[**https://old.karma.life/**](https://old.karma.life/)

* The user functionality is poor and can include many filter preferences.
* Could not find many stores.
* Not stable and has bugs.

**Market Analysis of Canada**

* The notion of food donation is progressively gaining traction in the Canadian market.
* As of now, applications are only providing these services in major cities such as Toronto and Vancouver, where there are several retailers.
* Due to the pandemic crisis, the number of individuals in need of food and the number of applications offering these services is growing by the day. As a result, consumers are demonstrating interest in these services, and there is a large demand for these sorts of apps since they satisfy their appetite while also allowing them to preserve food rather than waste it.

# Overall Description

## Product Perspective

Sharing Food connects restaurants and neighbors with each other and with local businesses so surplus food can be shared, not thrown away. This could be food nearing its sell-by date in local stores, spare home-grown vegetables, bread from your baker, or the groceries in your fridge when you go away.

This application works so easy, to make an item available, simply open the app, add a photo, description, and when and where the item is available for pick-up.

To access items, simply discover the listings available near you, request whatever takes your fancy and arrange a pick-up via private messaging.

## Product Functions

The function of the System include:

* The mobile application includes an interface where a seller/donator can post a product whenever he/she wishes to sell it at a lower price or to donate it.
* Needy person can view posts by sellers and purchase/get it for free from seller/donator.
* Seller and purchaser can both have a communication over chat in the mobile application related to the product or delivery.
* Filter will be provided according to location, date of expiry, cost/price, and tags provided by user.

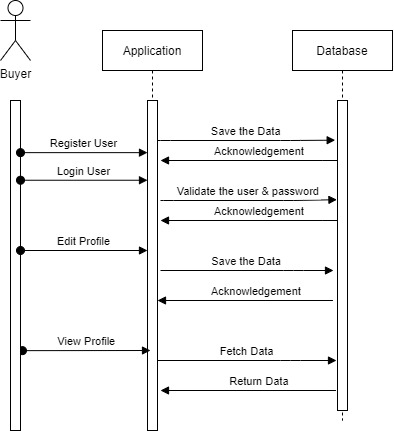


Fig 2.2.1 Sequential Diagram (Buyer/Customer)

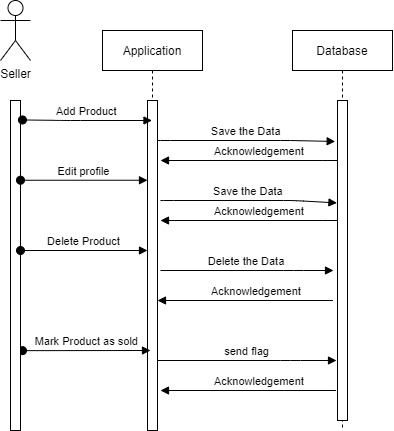


Fig 2.2.2: Sequential Diagram (Seller/Donator)

## User Classes and Characteristics

End-users might be both buyers and sellers, such as students, restaurants, and non-profit organizations.

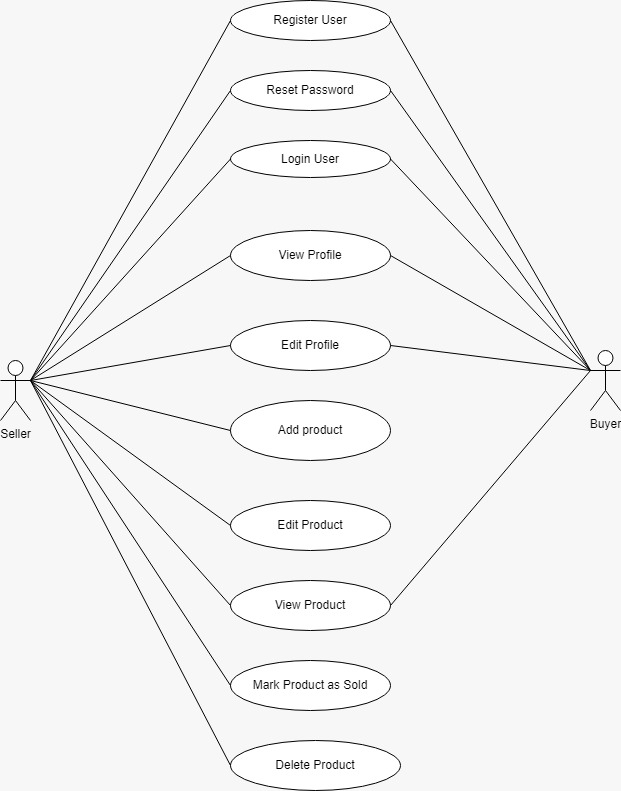


Fig 2.3.1 Use case diagram

**Use Case Scenario:** A baker makes 100 breads expecting to sell all of them but is unable to sell all of them even within 4 days. In this scenario, the actor is the baker, the goal is getting rid of the bread that will go bad. There is likely one outcome of this scenario wherein the baker must throw away his bread or look around for people in need of bread in his locality-- this is the common course use case. However, when the baker uses our app, he can have the people in need reach out to him and even get paid for it. The goal of the application is to connect the people in need of food with those who have an abundance of it.

## Design and Implementation Constraints

* This Application will have a dataset constraint as the people who register in our application cannot be shared as privacy of Users information.

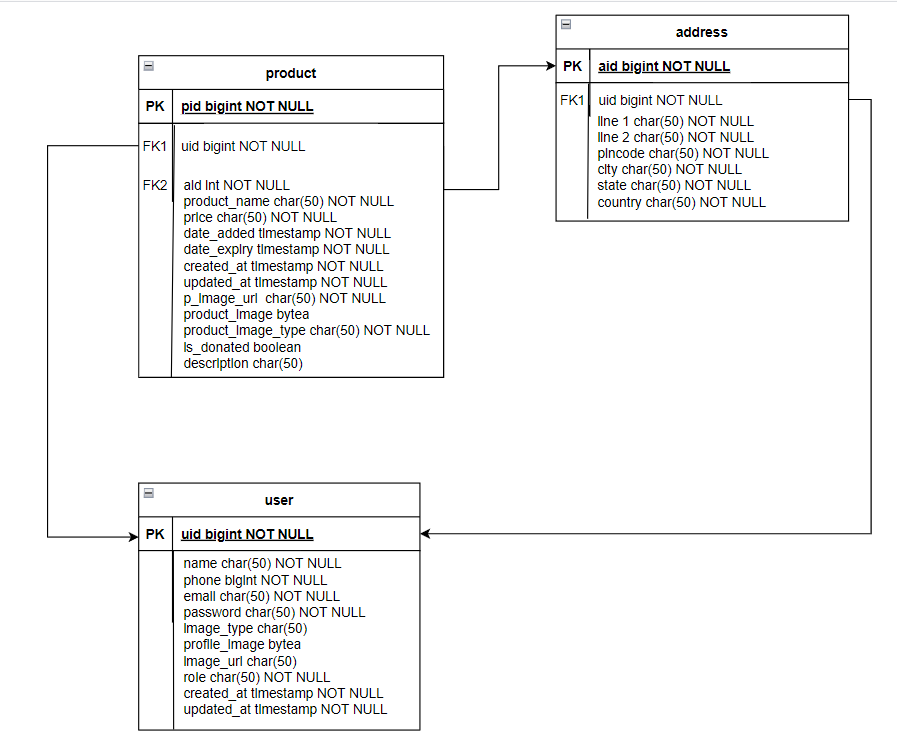


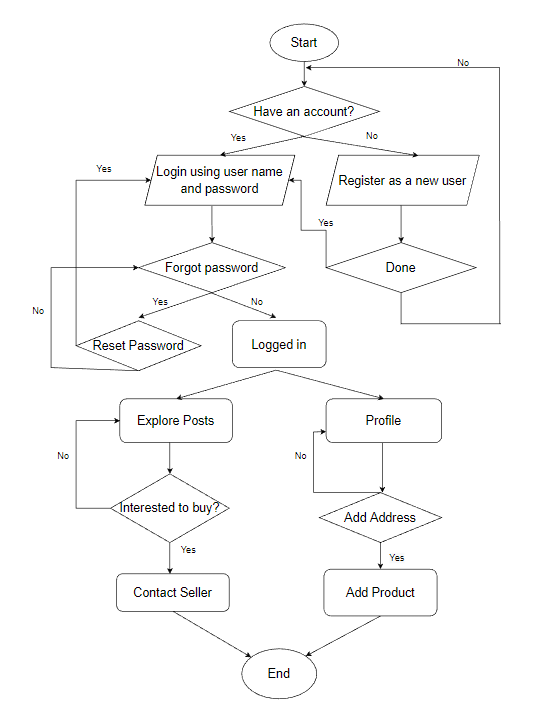
Fig 2.4.1 ER Diagram

## 2.5 Assumptions and Dependencies

As this platform is for everyone and it is available free of cost, there are chances that scammers can trap the sellers/donators which is why he/she should be aware before sharing confidential details like address or phone number.

**2.6 Workflow**

This section provides the basic workflow of Don8 application

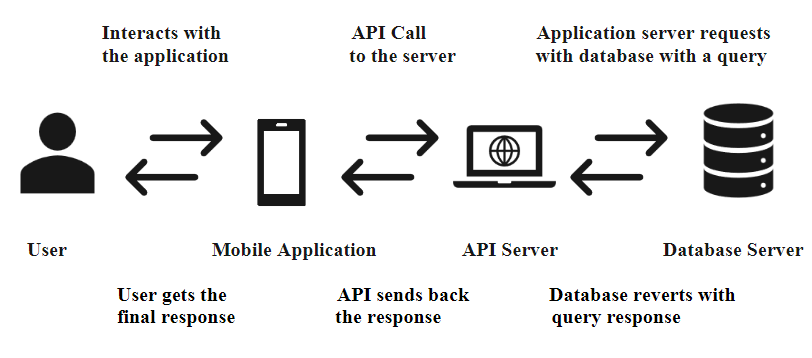


2.6 Workflow of Don8 Application

* Firstly, the user will be logging into their account if he/she is already an existing user or else user needs to register for a new account and needs to enter relevant details and password. After entering details and creating an account, the user can log in using his username and password.
* If an existing user forgets their password user can reset their password as well.
* Once the user is logged in, he/she might explore food products willing to buy and contact the seller or if the user wants to sell a product, he/she can post the product by first entering the address details and then adding the remaining details about the product.

**2.7 Architecture Diagram**

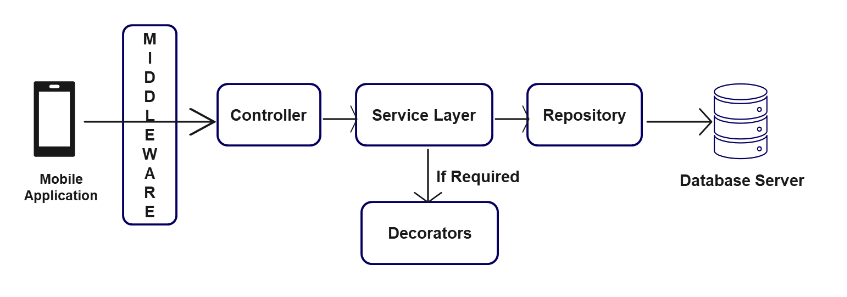
This section provides a basic architecture diagram on how user requests are processed.



2.7 Architecture Diagram for processing user request

* Firstly, the user interacts with the mobile application and requests a service. The mobile application interacts with the API Server which will process the request by connecting with database through query and in return receives the query response from the database. The API server then sends back the processed response back to mobile application where user will be getting final response for the service requested.

**2.7.1 Internal Architecture for Communication between Mobile Application and the API Server**



2.7.1 Internal Architecture Diagram

* This section describes the internal architecture on how mobile application communicates with the API server. Mobile application interacts with the API using middleware and requests controller which directs the job to service and then the service layer is responsible for requesting repository, which calls the database server. Service layer gets methods from decorators when required.

# External Interface Requirements

**Phase - 1 UX Design**

This design was created during the planning stage. The design shown below is simple. It consists of the basic layout and architecture of the application. Please note that these mockups are just for the approximation of how and where the user interface elements will be positioned. The result will be updated in upcoming designs

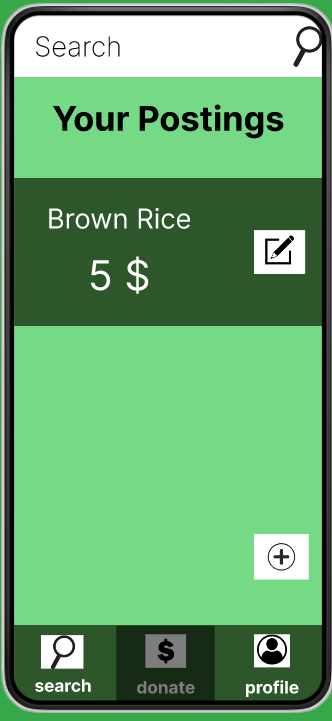
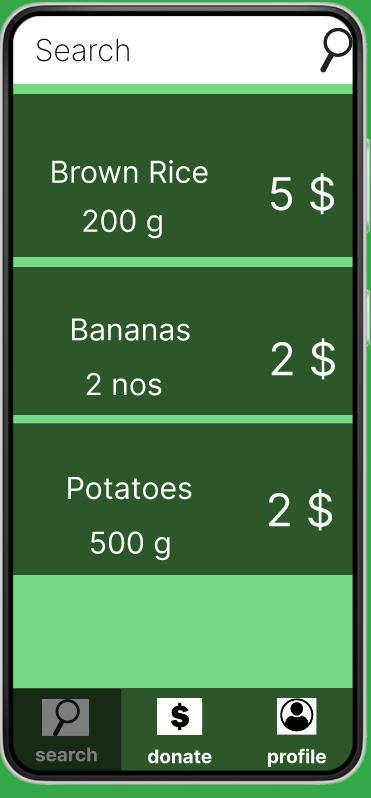
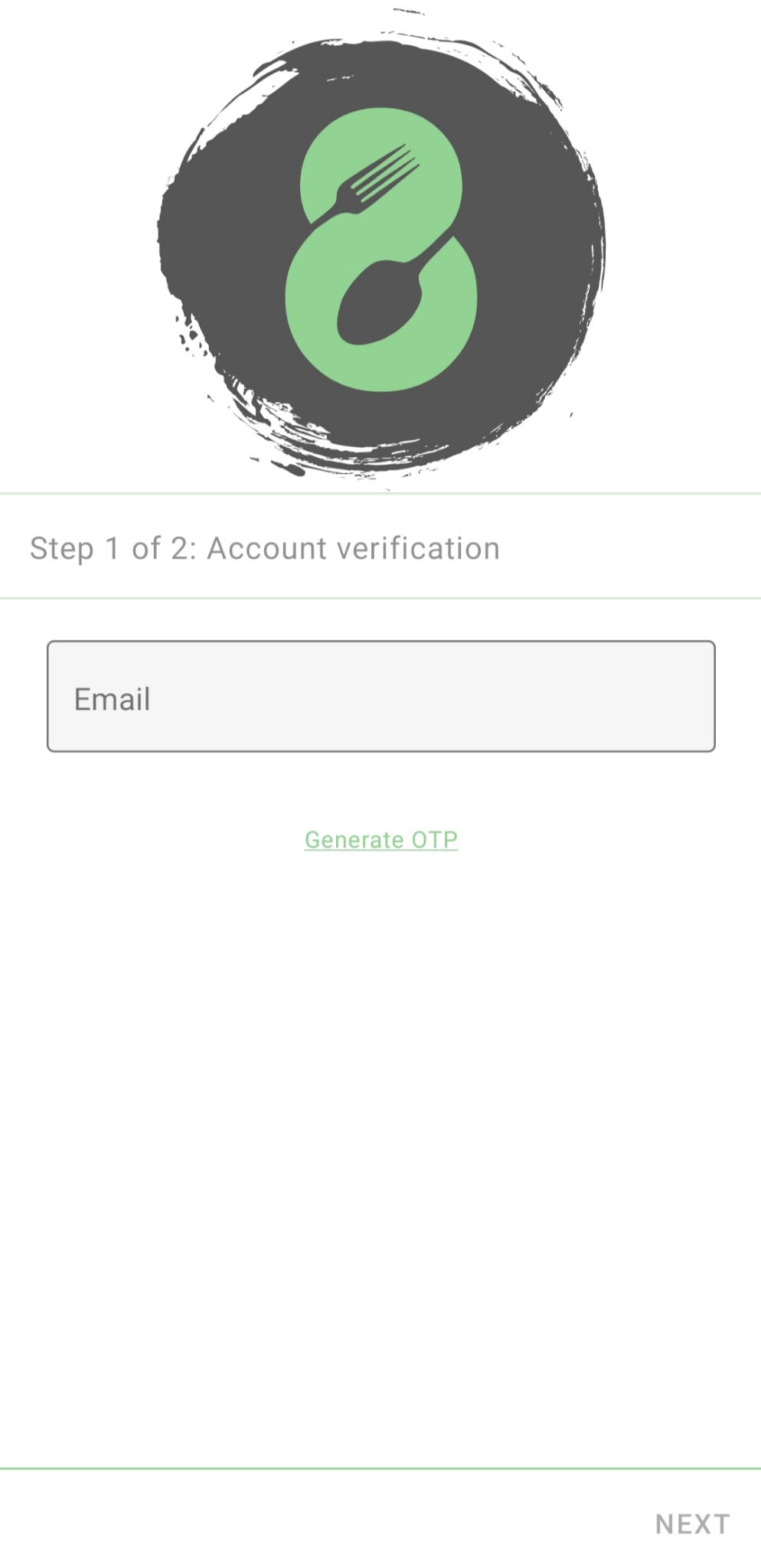
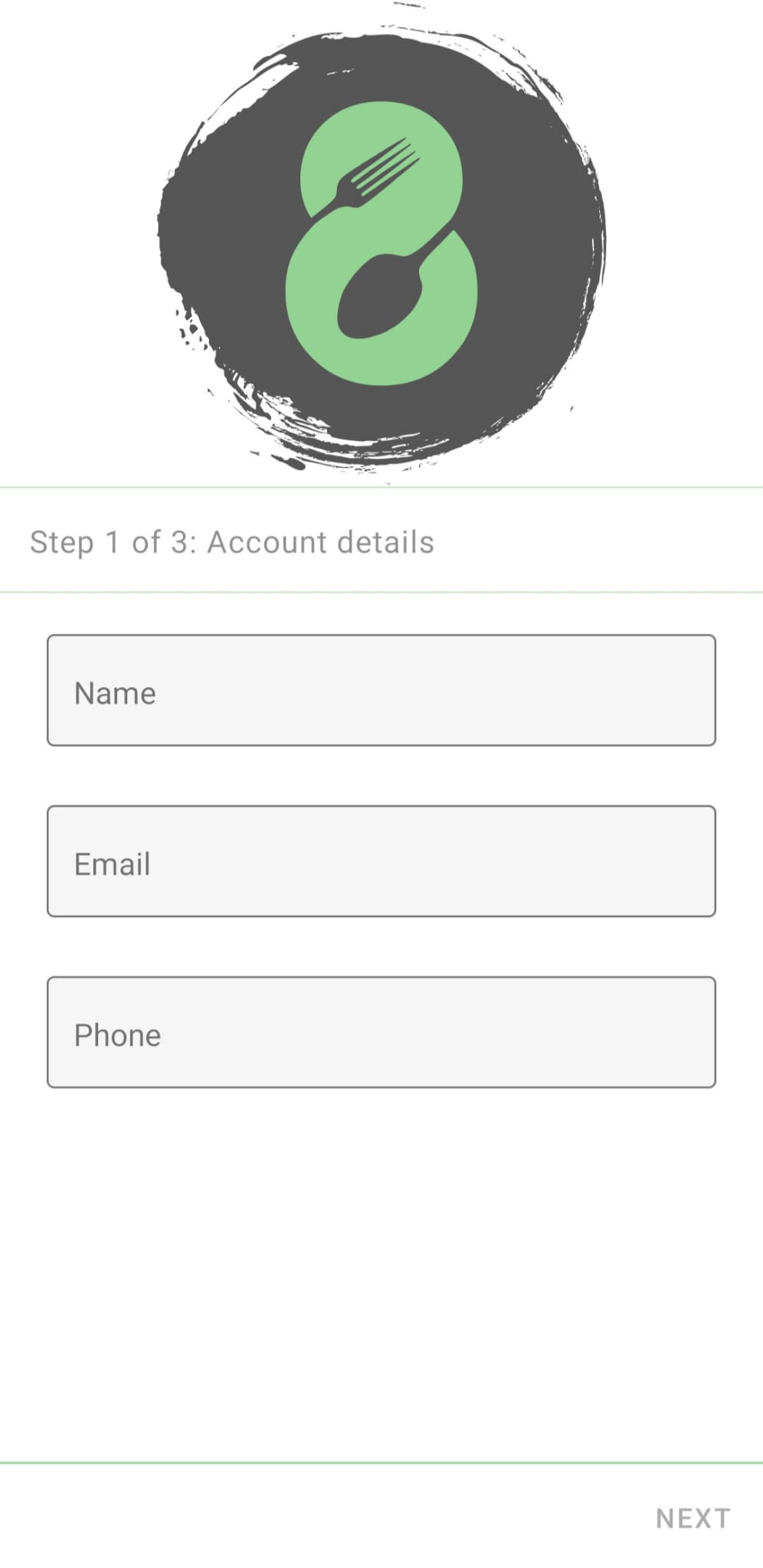
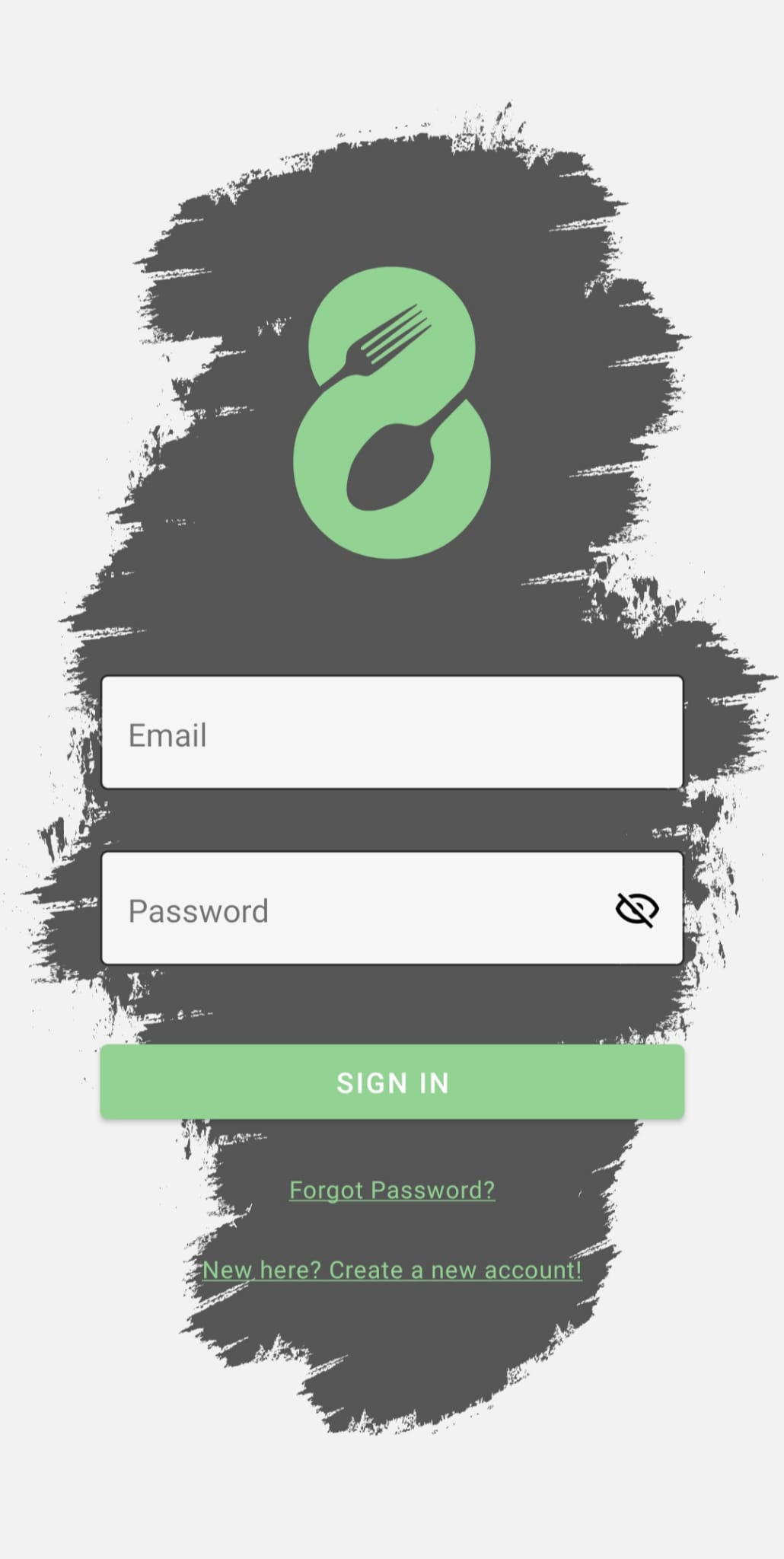


Fig 3.1 Phase 1 UX Screens

**Phase - 2 UX Design**

We have worked on the authentication modules and designed UX Screens for login page, forgot password and register page



. Fig 3.2 Phase 2 UX Screens

**Phase - 3 UX Design**

We have designed the profile section along with the posting and address UX screens.

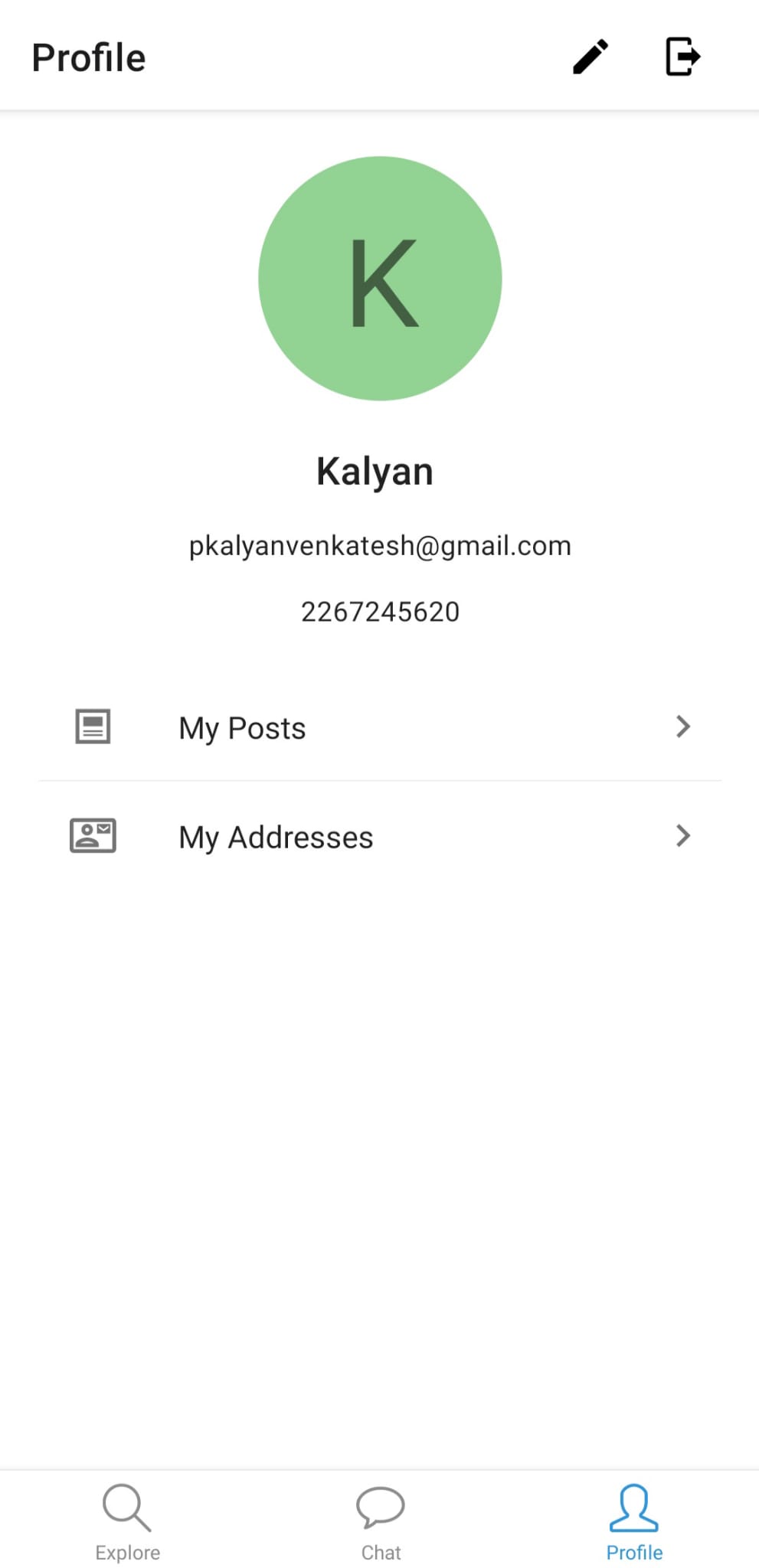


Fig 3.3 Phase 3 UX Screens

**Phase - 4 UX Design**

We have designed the explore UX page to display the product listings and to display details (image, description, price, expiry date, address) of the respective product and seller information.

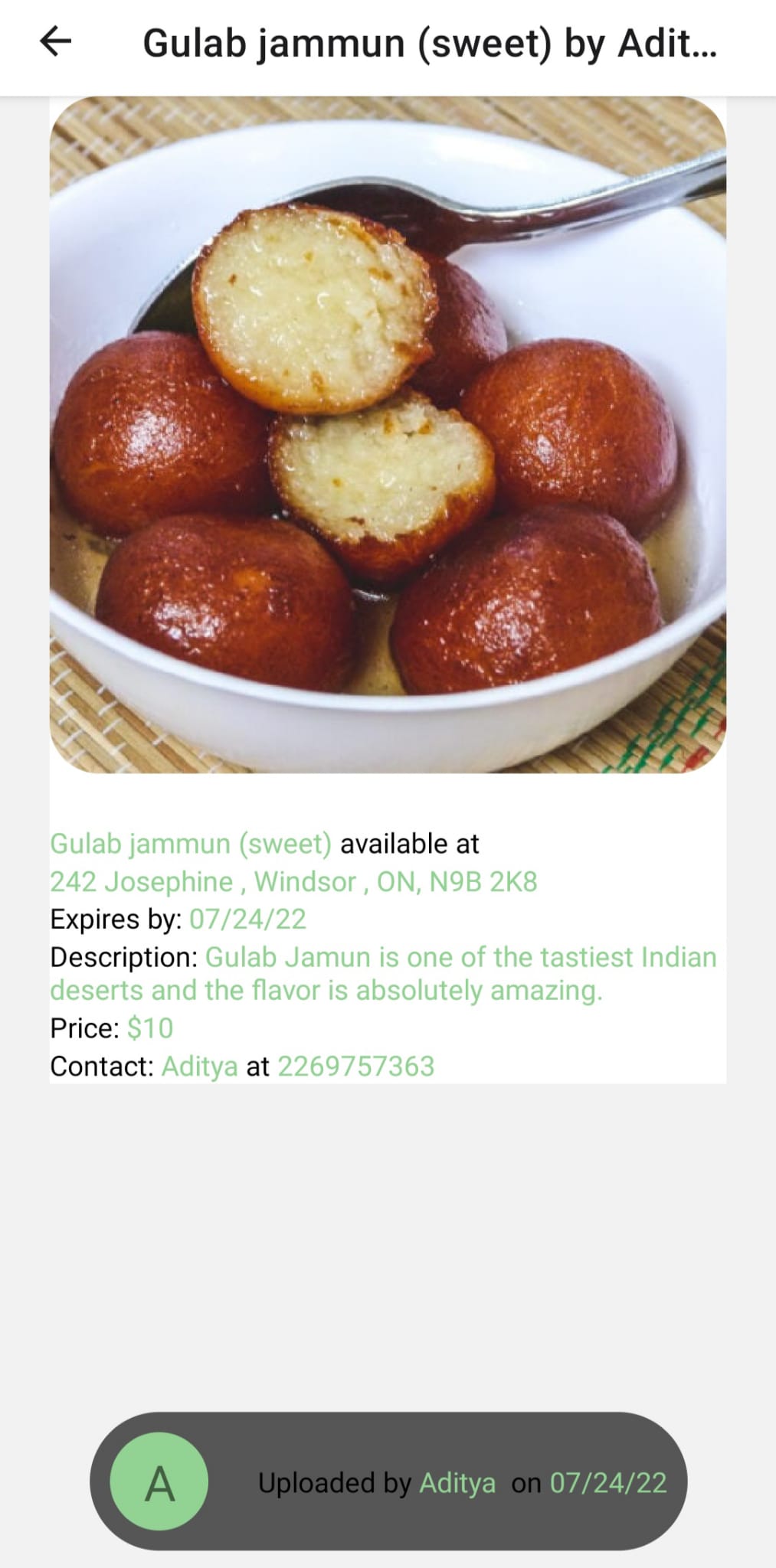
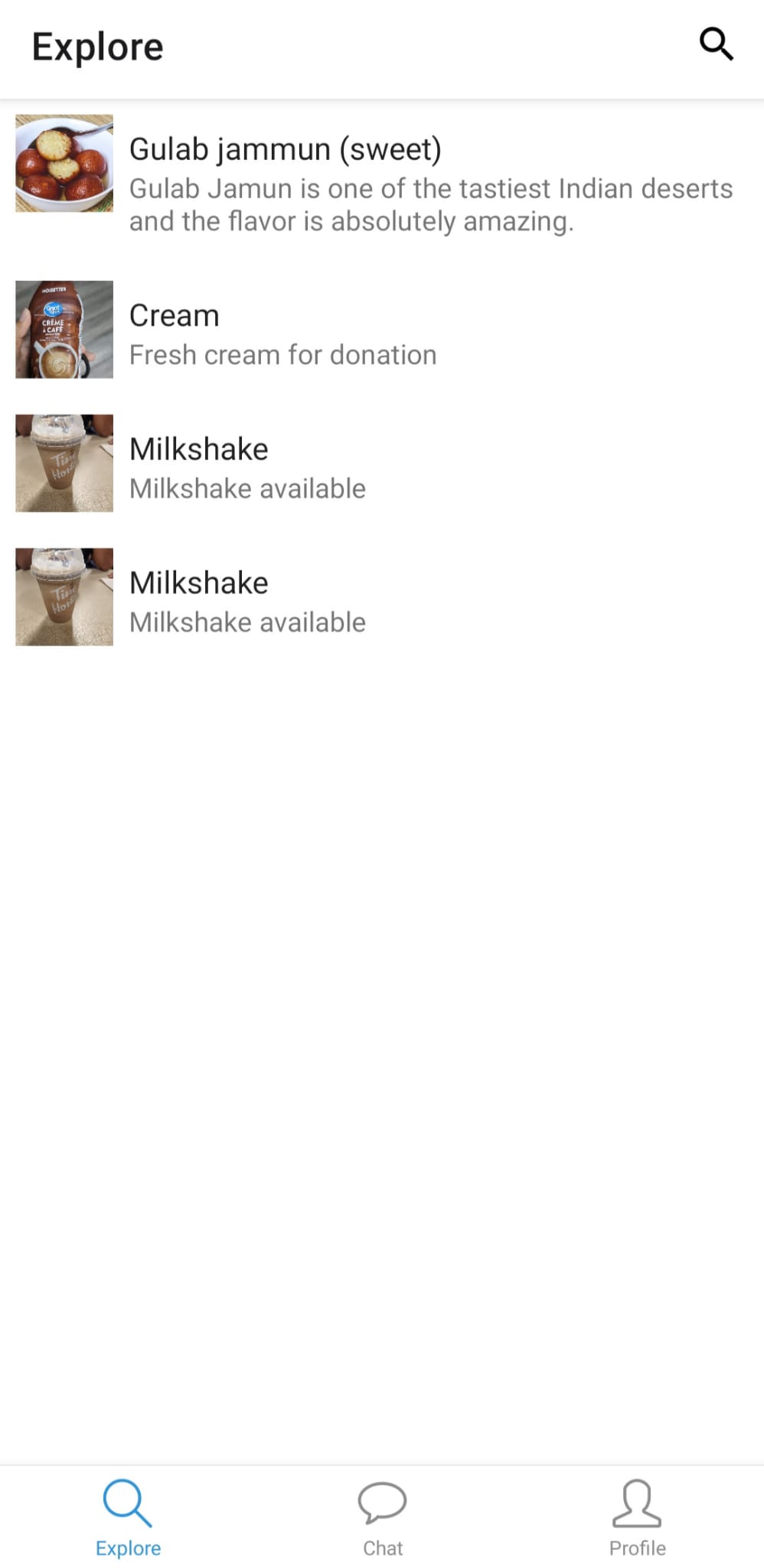


Fig 3.4 Phase 4 UX Screens

## Hardware Interfaces

* Every hardware running iOS 11.0 and Android 5.0 (API 21) or newer can run the mobile application.

**Production Environment**

* Minimum 8GB of RAM
* Minimum 6 CPUs
* Minimum 20 GB of storage

**Development environment**:

* Windows 8.1/10/11
* Intel Core i5-8259U, or AMD Ryzen 7 2700X or higher
* 8GB RAM
* 20 GB available storage

## Software Interfaces

* VSCode, Eclipse, IntelliJ (IDE)
* Database server (PostgreSQL)
  + - Spring Boot (Backend API)
    - React Native (Mobile Application)
    - Jira/Trello (Managing and Tacking)
    - Git/GitHub (Version Control)
    - Docker (To Store Application Image)

## Communications Interfaces

* Email SMTP server is required to verify user’s email address.
* All the APIs will be having a secured HTTPS connection to enable a safe and secured connection.

## Technology Justification

**React Native:**

It is an open-source framework that allows us to create cross-platform mobile applications, allowing us to utilize a single codebase for both Android and iOS, making cross-platform development much easier.

**Java (Spring Boot):**

Java Spring Boot (Spring Boot) is a solution that uses three basic characteristics to help create web applications and microservices using Spring Framework faster and easier: Autoconfiguration. Configuration with a strong point of view. The capacity to develop stand-alone programs.

**PostgreSQL Server:**

A free and open-source relational database management system that emphasises extensibility and SQL compliance is PostgreSQL, also referred to as Postgres.

# System Features

## Sign in/ Signup Pages

* **Login user**
  + If the user provides the correct username and password combination, then he/she will be redirected to the home page.
  + When the user provides an incorrect username or password then he/she will get an error message on the same page.
* **Register user**
  + When the user clicks on the register on the entrance page, then he navigates to a new page which asks the user the basic details to register.
  + In this page all the mandatory fields to be filled beifore clicking on register else he will get error message that please fill mandatory fields. Mandatory fields Once all fields are filled correctly and clicked on register then in the background, we will send an OTP to the given email which they have to enter in the next page to complete registration.
  + A new password is to be added after entering correct OTP in case if the OTP is correct user gets 3 attempts after which the OTP will be invalid, and email is to be corrected.
* **Reset Password**
  + If a user forgets their password, then they can click on the reset password in the entrance page to reset their password
  + This page will ask for user’s email id if the user is registered then the OTP will be sent to that email id else it will show error stating the user does not exist.

## Profile

* **View Profile**
  + The View Profile page shows user details like name, phone number, role, email, addresses, profile photo, etc.
* **Edit Profile**
  + Edit profile enables users to edit name, profile photo and phone number whereas email and role are immutable.

## Product

* **Posting a product** 
  + The seller can provide information about the product, such as its name, description, amount, unit, date of expiry, and image, as well as the location where it can be picked up.
* **Edit a post**
  + The seller has the access to edit their pending product details like name, description, amount, unit, date of expiry, and image, as well as the location where it can be picked up.
* **Delete a post**
  + Incase when seller wants to remove/delete a posting they are free to do it before the product is marked as sold/donated.
* **Mark post as donated**
  + After the product has been successfully provided, the donor may input the purchaser's email address to verify the transaction via OTP and designate the post as complete.
* **View a Product**
  + When a user clicks on any specific product on the home page, he should get complete product details like its name, description, amount, unit, date of expiry, and image of the product, as well as the location where it can be picked up.

## Home Page

* **List of Products**
  + A list of all the posts will be displayed initially with details like name, cost, and image of the product.
* **Filter Products** 
  + In the home page users can filter products by below mentioned criteria.
* Category
* Location
* Price
* **Sort Products**
  + In home page users can also sort products by the following criteria:
* Price
* Posted date

# Other Nonfunctional Requirements

## Performance Requirements

* It will be easy for the users to interact with the application if the response time for the filter preferences to sort out is less.
* It should contain most of the functionalities so that the user can easily navigate through the application, and it should be user friendly.
* Able to process all work correctly and completely without being aborted and should be flexible to use.
* Server should be able to process a considerable number of requests from users.

## Safety Requirements

* If a consumer consumes the expired food, he/she may suffer illness. So, we need to make sure that expired food should not be sold on the application.
* We need to display the ingredients regarding the specific food item because some of them are allergenic to a few of the food items.

## Security Requirements

* Personal details of either buyer or seller are confidential and should be displayed only if needed.
* Able to remove bugs and glitches from time to time and should give regular security updates.

## Software Quality Attributes

* Portability: application to be available on multiple OS’s, development frameworks, or databases
* Recoverability: The application should be able to restore function and data in the event of a failure.
* Maintainability: Application should be monitored and maintained to keep the system performing, secure, and running smoothly.

# Quality Assurance Plan

**6.1 Introduction**

The effort a software engineer makes to guarantee that the end-user receives the best product and experience is known as quality assurance (QA). As a mobile application developer, you ultimately want to make sure that everything functions as intended, at the proper speed, and on the user's device.

This section discusses metrics that will validate the product's quality. Additionally, it will guarantee that quality is checked at every stage of development.

**6.2 Testing Strategy**

This subsection mentions the various types and strategies used for testing the “Don8” application at various stages of development.

**6.2.1 Unit Testing**

The process of testing individual software parts or components is known as unit testing. The goal is to confirm that each piece of software code operates as intended. Developers perform unit testing while creating an application (the coding phase).

**Junit Testing**

JUnit is a unit testing open-source framework for the Java programming language. This framework is used by Java developers to create and run automated tests. Every time new code is added, certain test cases in Java need to be run again. To make sure that the code is not broken, this is done.

 Fig 6.2.1 Unit testing

In this test case, we are testing  user module which contains name, phone number, password, email address.

Each of these fields, are tested and verified using JUnit and below is the output for valid user.

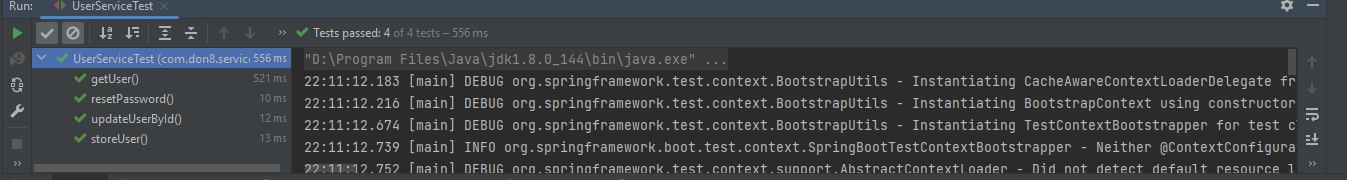


Fig 6.2.2 Unit testing results

**7.2.2 Integration Testing**

Integration testing is used to examine how various components work together, how they interact, how separate systems connect to form a larger system, and how closely the code complies to the specifications.

**Integration Testing:**

Testing various components together by verifying and validating the interaction between them is called Integration testing. This testing mainly focuses on the flow between set of modules.

In our project we have performed testing different scenarios of the communication between different components.

Example:

Scenario:

* Pass unregistered mail id in the email verification endpoint.
* Sign up with an unregistered mail id.
* Now sign in with the mail id which registered in the previous step.
* Add an address for this user.
* Get address by user id to know the address id.
* Now add product.
* Get product by product id and perform get the product by the user(token).

Expected:

* Success response for all endpoints and valid values.

Results:

Integration Testing Results

**6.2.3 Performance Testing**

Performance testing is a testing technique where we see how much speed, how responsive, how stable, and how scalable the application gives under a certain workload. In our application for few endpoints, we have done the performance test to check how responsive is our application.

Example:

Below picture shows the performance testing of the ‘get all products’ endpoint where I am giving the parallel users value as 10 and hitting the endpoint, now we will get the load time, connection time, latency. Based on our requirement we can set these values to certain limit and accordingly we can evaluate the performance of the application.

Graphical user interface, text, application, email

Description automatically generated

Fig 6.2.3 Performance testing - View Result Tree

**6.2.4 Compatibility Testing**

Compatibility testing is non-functional testing. It is done to see if a program or application can function properly across a range of hardware, browsers, operating systems, and networks.

DON8 is a mobile application that runs in hardware running on Android 12.0 or newer

1. Motorola G60 (Android 12)
2. Samsung Galaxy S22 (Android 12)

**6.2.5 Beta Testing**

The last round of testing before making a product widely available is called beta testing. In this controlled environment, the goal is to find numerous flaws or usability problems as possible.

The "actual" users that perform beta testing do it in a production setting using the same hardware, networks, etc. as the final product.

**6.3 Code Maintenance Platform**

The source code was regularly maintained using the version control tool Git on Github. The entire team was a collaborator on GitHub and had access to the code as well as the ability to publish, access, create, and integrate pull requests.

From May 2022 to July 2022, there have been 145 code commits to the github repository.

Github project links: <https://github.com/DhruvNair/Don8-UI>

<https://github.com/Jaydeep21/don8>

**6.4 Design Pattern**

**6.4.1 Creational Design Pattern**

**Builder pattern**

Builder is a creational design pattern that enables the step-by-step construction of complicated items.

The builder doesn't require products to have a standard interface, in contrast to other creational patterns. This enables the production of many goods using the same construction method.

Text

Description automatically generated

Fig 6.4.1 Builder pattern -1

Text

Description automatically generated Fig 6.4.2 Builder pattern -2

Explanation: We are Using Lombok library to create builder class. As we can see in the above Fig 6.4.2 Builder pattern -2 GenericResponse is build according to our preference.

**Prototype**

A creational design pattern called prototype enables the duplication of objects—even complicated ones—without attaching them to particular classes.

Text

Description automatically generated Fig 6.4.3 Prototype pattern -1

Text

Description automatically generated Fig 6.4.4 Prototype pattern -1

AuditModel is a prototype which is having fields “createdAt” and “updatedAt” which will be used by all the other objects one of them is Product as shown in above Fig 6.4.4 Prototype pattern -1.

**6.4.2 Structural Design Pattern**

**Composite**

A structural design pattern called Composite enables you to group objects into a tree-like structure and interact with them as if they were all one object.

Text

Description automatically generated Fig 6.4.5 composite pattern -1

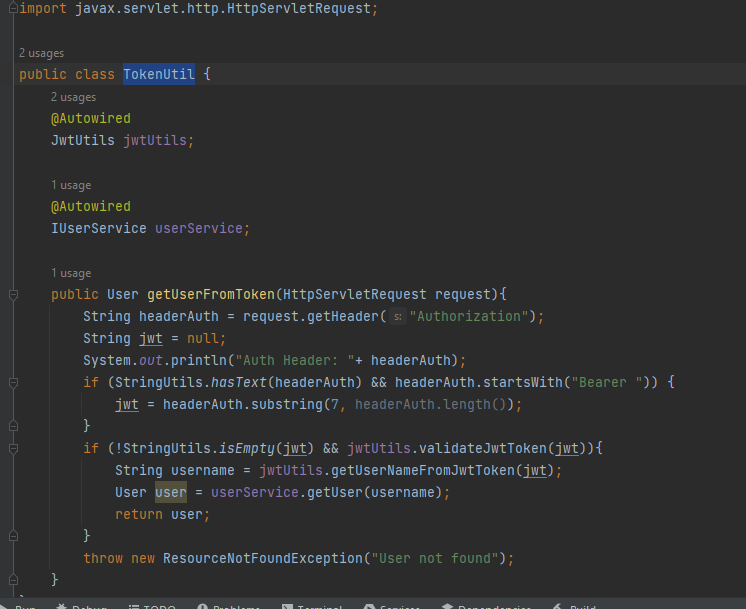
Text

Description automatically generated Fig 6.4.5 composite pattern -2

Explanation: As we can see in above image() AuditModel is extending Serializable and Product extends AuditModel this is how multiple hierarchy is formed to build a family. In this case all the methods coming from Serializable and AuditModel are inherited by Product class.

**decorator**

A structural design known as decorator enables dynamically introducing new behaviours to objects by enclosing them in specialised wrapper objects.

 Fig 6.4.6 decorator pattern -1

Text

Description automatically generated  Fig 6.4.7 decorator pattern -2

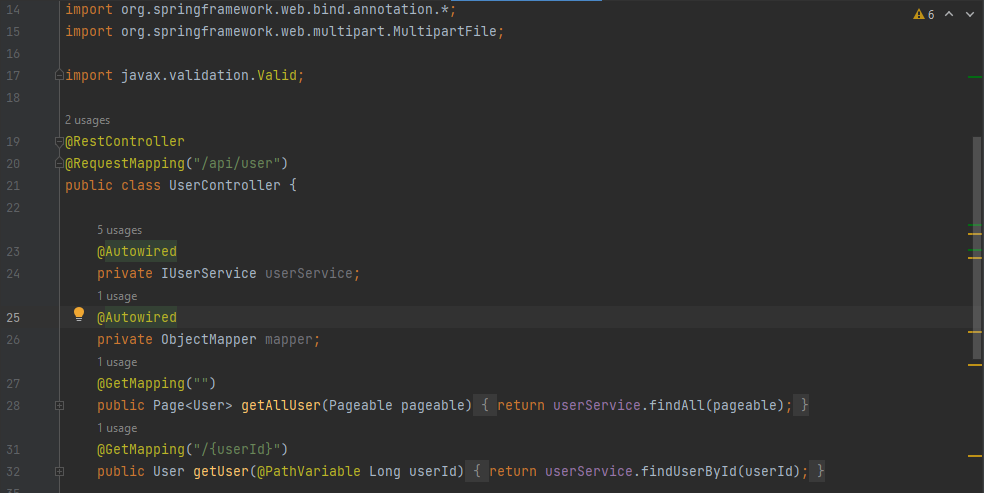
Explanation: As we can see in above image there is one decorator created which will always fetch User object from the auth token. Service class can call this decorator anytime when needed. This avoids repeating same code everywhere.

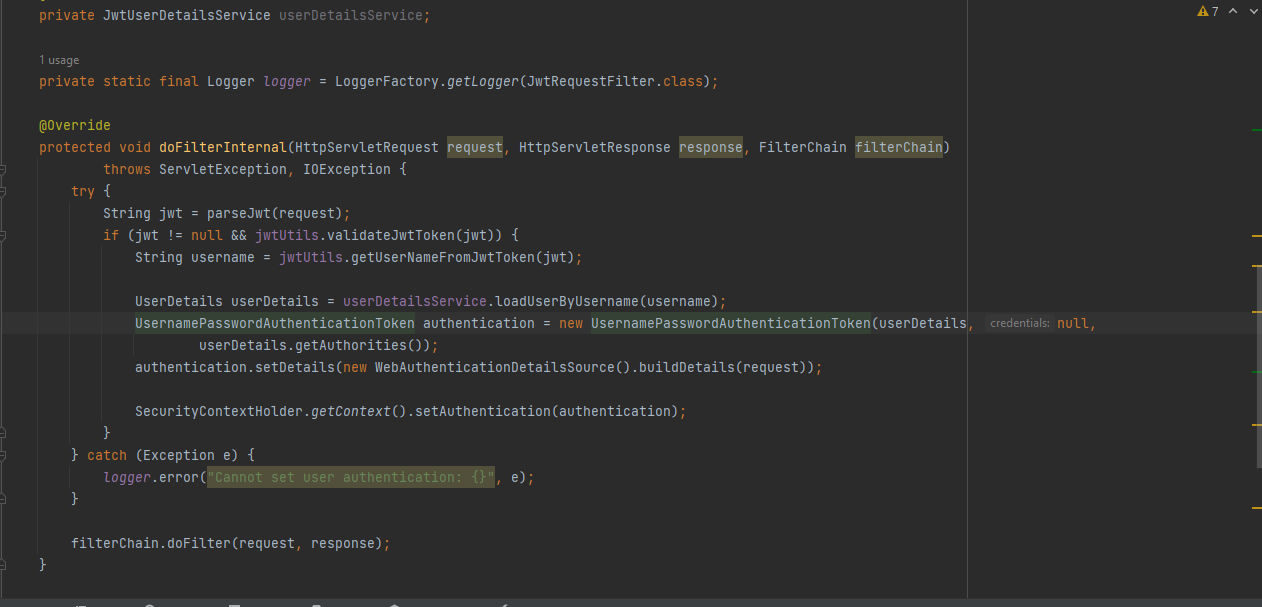
**6.4.3 Behavioral Design pattern**

**Mediator**

A behavioural design pattern called mediator makes program components communicate with one another indirectly via a unique mediator object, hence reducing coupling between them.

Individual components are simple to edit, expand, and reuse thanks to the Mediator because they are no longer dependant on the numerous other classes.

 Fig 6.4.8 Mediator pattern -1

 Fig 6.4.9 Mediator pattern -2

Text

Description automatically generated Fig 6.4.10 Mediator pattern -3

Text

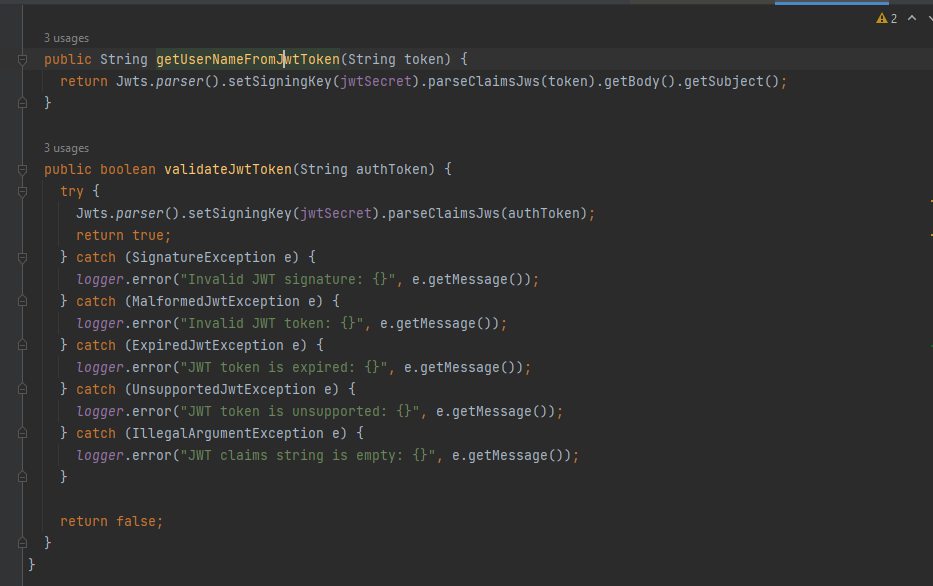
Description automatically generated Fig 6.4.11 Mediator pattern -4

Explanation: - In the above scenario UserController class is the mediator which connects request from User Interface to correct service class and accordingly service class makes repository call and so on.

**Chain of Responsibility**

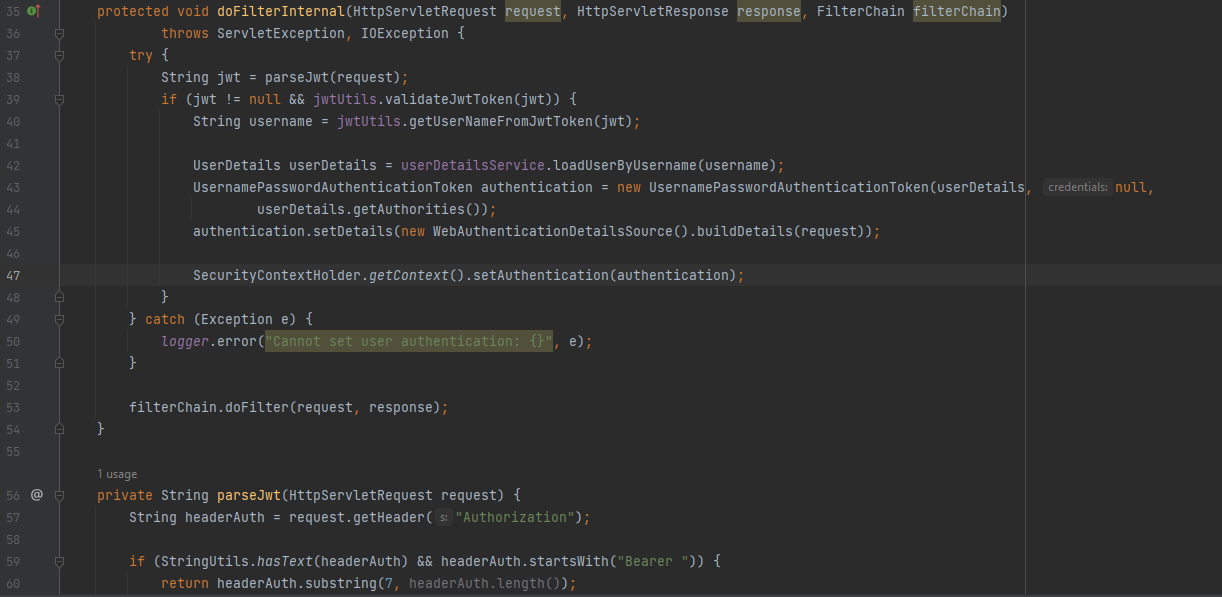
Until one of them takes on the request, requests can be sent along a chain of potential handlers using the behavioral design pattern known as Chain of Responsibility.

The approach enables several objects to respond to the request without the sender class becoming dependent on the receivers' concrete classes. Any handler that adheres to a common handler protocol can be used to dynamically assemble the chain at runtime.

 Fig 6.4.12 Chain of Responsibility pattern -1

Text

Description automatically generated Fig 6.4.13 Chain of Responsibility pattern -2

 Fig 6.4.14 Chain of Responsibility pattern -3

Explanation: For any request there is always a middleware layer whose job is to validate the request for factors like

1. If the token is present or not.

2. If the token present is legitimate.

3. If the token is valid or expired.

4. If the user role having access to make this request.

All such scenarios are automatically called, and every method has a chain of responsibility if one fails it is passed to the other one.

## 6.5 PROJECT PLANNING

Scrum is a popular agile approach for product development, particularly software development. Scrum is a project management methodology that may be used on any project with tight deadlines, complicated requirements, or a high level of uniqueness. Projects in Scrum are progressed through a series of iterations known as sprints. Each sprint usually lasts two to four weeks.

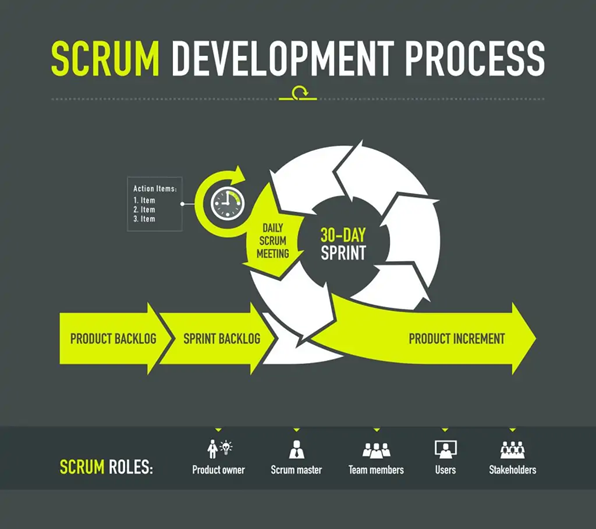


Figure 6.5.1 Scrum Development Process

**The Role of Scrum**

The Scrum team focuses on producing high-quality software. The Scrum project owner concentrates on identifying what qualities the product must have to construct (what to build, what not to build, and in what sequence) and overcoming any obstacles that may obstruct the development team's job.

**Scrum master:** The person in charge of leading the team and ensuring that they follow the methodology's rules and practises. Scrum masters handle the project's obstacles and collaborate with the Product Owner to maximise ROI. The Scrum Master is responsible for maintaining Scrum up to date, as well as coaching, mentoring, and training to the teams in the Scrum framework.

**Teams**: A group of specialists with the requisite technical competence who work together to build the project and complete the stories they commit to at the beginning of each sprint.

**6.5.1 Provisional Planning**

The DON8 project includes distinct goals and deliverables. To achieve the intended outcome, careful project planning and management are required. Given that we are using the Scrum Agile Model, each sprint is supposed to last two weeks in our action plan. For each sprint, the team would choose a few things from the product backlog and work to fulfill them.

Here is the detailed tentative planning for the Don8 Project. Between May 15 and July 24, 2022, there are 5 sprints total, plus 2 necessary milestone meetings. The tentative planning has provided a clear roadmap for the project.

Iteration 1 (15th May 2022 - 29th May 2022)

* Functional: -
* Write a business model.
* Design markup UI according to requirements.
* Setup Database, GitHub, Jira
* Non Functional: -
* Security: - GitHub repository should be private; all the members should be connected in Jira and database should be robust.
* Scalability: - Database should be capable of handling multiple requests at a time.
* Reliability: - Database should be reliable so that application does not crash in between.

Iteration 2 (30th May 2022 - 12th June 2022)

* UI and database design
* Setting up DevOps tools and creating a database
* Database and UI Designing
* Setting up DevOps tools and creating a database
* Starting the Development: Create a simple project and begin by developing the authentication APIs (login, register, and reset password). Start the front-end development of the login, register, and reset password UI at the same time.

Iteration 3 (Every functionality applies for backend and frontend) (13th June - 26th June)

* CRUD operations for product
* Search Module for buyer
* Exploring Module
* Posting and Display buyer's product listing.
* Display user profile.
* Edit profile (frontend and backend).

Iteration 4 (27th June 2022 - 10th July 2022)

* Profile Module
* Creating CI/CD
* Socket Server Setup
* Deployment of backend APIs

Iteration 5 (Testing) (11th July 2022 - 24th July 2022)

* Deployment, integration testing and documentation.

**6.5.2 Effective Planning**

Tasks were planned appropriately for each sprint to match the criteria and the scenario provided to us based on the provisional planning for DON8. The following details how each sprint was efficiently planned using TRELLO software.

* **ITERATION 1 -** [15th May 2022 - 29th May 2022]

|  |  |
| --- | --- |
| Tasks | Assignee |
| Project Proposal | Team |
| Analysis Of the need | Team |
| Setup Trello | Sara |
| Market Study of Product | Kalyan |
| R & D AWS | Bharat |
| R & D Azure | Jaydeep |
| GitHub Repository | Jaydeep |
| A Market Analysis | Sara |

Table 6.5.1: Sprint 1 planning

Iteration 2 (30th May 2022 - 12th June 2022)

|  |  |
| --- | --- |
| Task | Assignee |
| Initial Setup: Server and DB | Jaydeep |
| Initial Setup for Front | Dhruv |
| R & D for Junit | Sara |
| Auth Module - Sign In, Sign Up and Email Verification API, Forget Password | Dhruv |
| Auth Module - Sign In, Sign Up and Email Verification API, Forget Password | Jaydeep |
| Create update, read, and delete operations for product | Bharat |
| Address Module- Address Screens | Dhruv |
| Address Module - Address | Jaydeep |
| Address List Component | Arun |
| API Integration | Dhruv |
| Integration testing for API’s | Bharat |
| Unit testing for user | Jaydeep |

Table 6.5.2: Sprint 2 planning

Iteration 3 (Every functionality applies for backend and frontend) (13th June - 26th June)

|  |  |
| --- | --- |
| Task | Assignee |
| Search Module - Search Screen | Dhruv |
| Add image for product class | Bharat |
| API Integration | Dhruv |
| Unit testing for API's | Jaydeep |
| Integration testing for API’s | Bharat |
| Gant chart | Sara |
| Explore Module | Dhruv |
| Posting List Component | Kalyan |
| R & D For Testing Models | Sara |
| Posting Module | Dhruv |

Table 6.5.3: Sprint 3 planning

Iteration 4 (27th June 2022 - 10th July 2022)

|  |  |
| --- | --- |
| task | assignee |
| Profile Module- Edit, Screen, create posting,Edit | Dhruv |
| Profile Module- Edit, Screen, Create posting,Edit | Jaydeep |
| Beta Testing | Arun |
| Compatibality Testing | Kalyan |
| Creating CI/CD | Jaydeep |
| API Integration | Dhruv |
| Unit testing for API's | Sara |
| Product Users Types Id | Bharat |
| Product API modifications | Bharat |
| Unit Testing for Api’s | Jaydeep |
| R & D Jmeter | Sara |

Table 6.5.4: Sprint 4 planning

Iteration 5 (11 July 2022 - 24 July 2022)

|  |  |
| --- | --- |
| task | assignee |
| Deployment, integration testing and documentation | Team |
| Testing and Bug Fixes - Backend | Jaydeep |
| Testing and Bug Fixes - Front | Dhruv |

Table 6.5.5: Sprint 5 planning

## 6.6 Risk Analysis

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sr. No | Risk | Owner | Probability of Occurrence | Loss Size | Risk Exposure |
| 1 | There is insufficient time for research and development of the chat feature | Dhruv | 0.6 | 15 | 9 |
| 2 | The changes pushed to the backend server cause a fatal crash | Jaydeep/  Bharat | 0.5 | 5 | 2.5 |
| 3 | The salt of the password hash gets leaked | Jaydeep | 0.1 | 1 | 0.1 |
| 4 | Insufficient QA team to validate on all phone vendors. | Arun/Kalyan | 0.5 | 2 | 1 |
| 5 | Following end-user testing, more effort on the user interface may be necessary. | Dhruv | 0.3 | 5 | 1.5 |
| 6 | Support for one of the project dependencies is stopped abruptly | Dhruv/Jaydeep | 0.4 | 10 | 4 |
| 7 | Client wants new features implemented without prototype or specifications | Dhruv/Jaydeep | 0.3 | 30 | 9 |

Table 6.6.1: Risk Analysis

Impact

|  |
| --- |
| Likelihood |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Low | Medium | High | Critical |
| Low | 3 |  |  |  |
| Medium |  |  | 5 | 6,7 |
| High |  | 4 | 2 |  |
| Critical |  | 1 |  |  |

Table 6.3.1 Risk Analysis

## 6.7 Cost

|  |  |
| --- | --- |
| **ROLES** | **Estimated salary per hour** |
| Developer/tester | 30 $/hr. |
| Scrum Master | 30 $/hr. |

Table 6.7.1 Estimated Salary for each role

|  |  |  |
| --- | --- | --- |
| TEAM MEMBER | EXPERIENCE | PROJECT ROLE |
| Arun Reddy Nalla | 4 Months Internship + 8 Months + .NET Developer | Front-end Developer |
| Dhruv Vinod Nair | 1 Year- React/React Native + WebRTC developer | Front-end Developer |
| Jaydeep Pritesh Dharamsey | 2 years - Microservices developer, Java | Back-end Developer |
| Kalyan Venkatesh Poludasu | 5 Months – Analyst Trainee | Front-end Developer |
| Seyedeh Sara Salamati Taba | 4 Years – Product Manager | SCRUM Master |
| Veera Venkata Bharat Kumar Vayitla | 3 years - developer/automation QA | Back-end Developer |

Table 6.7.2: Experience and Role in the Project

|  |  |
| --- | --- |
| Fee Components | Estimated salary for 4 weeks |
| Basic Salary | 2400$ |
| Basket of Allowances | 600$ |
| Health Insurance | 400$ |
| Total Gross Salary | 3400$/- |

Table 6.7.3: Estimated Time Taken for completing the project is 4 Weeks. Cost analysis for each developer for the entire project

|  |  |
| --- | --- |
| Fee Components | Estimated salary for 4 weeks |
| Basic Salary | 2880$ |
| Basket of Allowances | 600$ |
| Health Insurance | 400$ |
| Total Gross Salary | 3880$/- |

Table 6.7.4: Cost analysis for each Scrum Master for the entire project

Total number of Students: 6

Total number of hours worked per week: 20 hours.

# USER MANUAL

**How to run your application (Backend)?**

Our backend application is deployed on Heroku server which is why we can directly hit API and test it. If we want to run on a local server here are the following steps:

1. Download tools/software Required:

Open JDK version >=1.8

Maven >= 3

1. Clone project from GitHub using either of the ways:
   1. Using command line interface: - git clone <https://github.com/Jaydeep21/don8.git>)
   2. Download zip: - <https://github.com/Jaydeep21/don8>
2. Clean and install all libraries using maven (mvn clean install)
3. Run the main spring boot application file inside {main\_dir}/ src/main/java/com/don8/

**How to run your application (Frontend)?**

1. Download tools/software Required:
   1. React native ([React Native Documentation](https://reactnative.dev/docs/environment-setup)) Using the Expo CLI QuickStart section
   2. Yarn ([Yarn Documentation](https://classic.yarnpkg.com/lang/en/docs/install/#windows-stable))
2. Clone project from GitHub using either of the ways:
   1. Using command line interface: - git clone <https://github.com/DhruvNair/Don8-UI.git>)
   2. Download zip: - <https://github.com/DhruvNair/Don8-UI/archive/refs/heads/main.zip>
3. Clean and install all libraries using yarn (run “yarn” in the project directory)
4. Run the application using expo (run “expo run:android” in project directory)

# Pros and Cons of your solution

**Pros:**

1. We are reducing food wastage
2. We’ve used many design patterns which makes many functionalities reusable and easy to maintain.
3. Sharing food can contribute to more effective use of our natural resources, which is a substantial advantage for environment.
4. Our global population is expanding quickly, and if we continue to use our natural resources inefficiently, there is a good probability that many more people may go hungry and starve in the future. Therefore, it is essential that we utilize our natural resources in the most effective way in order to feed as many people as possible.

**Cons:**

1. Potential health issues may occur because seller might unknowingly donate expired food.
2. Some user might add unwanted posts and create disturabance.

# Difficulties faced during the project

1. Deployment: - Our aim was to find a server which did not charge initially so that development cost would be minimum. We tried using Azure and AWS, but both were having some issues for free version which is why we jumped to Heroku. This complete R&D and testing took 1-1.5 weeks.
2. Testing: - As we wanted testing to be mandatory before deployment to the main branch, we had to learn how to automate this process so that code in production is secured and also a previous version of it is stored in Docker as an image file as a backup.
3. Connection Limit: - As we were using a free version of database from Heroku Server which had a restriction of 20 connections at a time which is why we made production and development environment to tackle this issue and completed development smoothly.

**10. Future Scope**

* In future we can add a chat option which will be a reliable way to communicate with the buyer/seller.
* We can add geo location integration so that it becomes easy for the buyer to reach the location where buyer is offering the product.
* We can expand our application scope to other platforms such as IOS
* Adding a rewards and referrals section that motivates the user to use our application.
* Improving user experience by attractive UI screens and improving performance.

**Differences between the initial and current SRS**

* **Market Research**

The market research section of the report contains in-depth information and insight into the Canadian food market, the diseases impacted by Covid disease, and the study of major and well-known Canadian businesses. Eventually, we reached the competition arena.

* **Architecture Design**

Two-tier architectures separate applications into Client systems handle Presentation and Application layers, while server systems handle Database layers, we used this approach for our product.  **Our application is designed on the basis that we can use the advantages of this method, which include faster development, improved scalability, and improved reliability.**

* **Quality Assurance**

Since we tested the software in this part, quality assurance testing is the act of determining whether or not a system or one of its components satisfies predetermined requirements. Simply put, software quality assurance is the process of carrying out a series of operations to find discrepancies, mistakes, and faults that conflict with the specifications.

* **Project Planning**

**The don8 application had 5 iterations at the start of the software development process, and all of the backend, frontend, and testing items were completed successfully. At the completion of the job, all of the things were clearly outlined in Trello and the Gantt chart.t**

## 11. References

## https://www.researchandmarkets.com/reports/5521689/canada-fast-food-market-share-insight-forecast

https://www.renub.com/canada-fast-food-market-p.php

[1] B. R. M. Roger S. Pressman, Software Engineering: A Practitioner’s Approach, Eighth Edition, McGraw Hill.

[2] C. Larman, Applying UML and Patterns: An Introduction to Object-oriented Analysis and Design and the Unified Process, Prentice Hall, Second edition.

[3] B. Eisenman, Learn React Native, Building Native Mobile Apps with JavaScript, O’Reilly Media, 2015.

[4] S. P. Reddy.K, Beginning Spring Boot 2, Applications and Microservices with the Spring Framework, Hyderabad: Apress.

[5] H. Bentahar, "Al Akhawayn University: capstone-repository.html," 2019. [Online]. Available: <http://www.aui.ma/sse-capstone-repository/pdf/spring-2019/FOODCOURT%20AN%20ONLINE%20FOOD%20ORDERING%20PLATFORM.pdf>.

[6] K. E. Khroufi, "Al Akhawayn University: capstone-repository.html," 2017. [Online]. Available: <http://www.aui.ma/sse-capstone-repository/pdf/spring-2017/First_Step_Mobile_Application_Final_Report.pdf>

[7] I. Ibrisevic, "Donor Box Blog," 26 April 2022. [Online]. Available: <https://donorbox.org/nonprofit-blog/food-bank-fundraising-ideas>.

[8] M. A. R.Adline Freeda, "Mobile Application for Excess Food Donation," IJIRSET, 2018.

[9] R. Dr.T.Sankar, "Review in Food Wastage Reduction Through Donation Application," IJIREEICE, 2020.

[10] <https://www.atlassian.com/software/jira>

[11] <https://www.guru99.com/risk-analysis-project-management.html>

[12] <https://lovefoodhatewaste.ca/about/food-waste/>