# **TASK 4:**

SOFTWARE REQUIREMENT SPECIFICATIONS (SRS) & UML DIAGRAMS

Submitted by:

| NDIKINTUM CARL NFON | FE20A073 |
| --- | --- |
| NYENTY EYONG ARREHQUETTE | FE20A094 |
| OBASIARREY M’ONEKE MARY | FE20A095 |
| OROCKTAKANG AGBORBEJA | FE20A097 |
| SALLE-NJUME MERYL EPOTE | FE20A102 |

Supervised by:

Dr. VALERY NKEMENI

# **TABLE OF CONTENT**

[**TASK 4: 0**](#_i72u63pjsnbs)

[**TABLE OF CONTENT 1**](#_vybolm1xc6ej)

[**INTRODUCTION 2**](#_xh2338gda2qe)

[**System Requirement Features 3**](#_2bpc5hju52r3)

[1. User Management 3](#_k34gvjrwpkbr)

[2. Donation Management 3](#_bi37krqquxph)

[3. Tracking 3](#_g7r88ifq7b31)

[4. Location Management 3](#_ycoi4vf4zez8)

[5. User Permissions 3](#_3nakmny6czfh)

[6. Search Functionality 3](#_dwczdws1o0x5)

[7. Reporting and Analytics 4](#_m0w359z40dcr)

[8. Communication 4](#_u25gynzbi0hf)

[9. User Feedback 4](#_efgcewkj31az)

[**Activity Diagram 5**](#_4t3co22w95q)

[**Use Case Diagrams 6**](#_lwufnxob4cik)

[**Class Diagrams 10**](#_bhzrzusspvzh)

# **INTRODUCTION**

In order to efficiently design a system, it is necessary to come up with an SRS - Software Requirement Specification document. This document describes what the software will do and how it will be expected to perform. It will also describe the functionality the platform the system needs to fulfil and the needs of its stakeholders.

# 

# 

# **SYSTEM REQUIREMENT SPECIFICATIONS**

## **1. User Management**

The platform should allow for users to easily create and manage their accounts, including updating their personal information.

## **2. Donation Management**

The platform should have a donation management system that allows donors to donate food items, including details such as number of items, expiration date, type of food, and quantity.

## **3. Tracking**

The platform should have a way to track and manage the inventory of the donated items in real-time, with alerts sent to recipients when certain items are close to expiry.

## **4. Location Management**

The platform should allow for the creation and management of locations such as food banks, shelters, etc.

## **5. User Permissions**

The platform should allow different user roles and permissions, such as donors, recipients, with different levels of access to the system.

## **6. Search Functionality**

The platform should have a powerful search feature that allows users to find the nearest donation locations, users, and items available for donation based on their location and preferences.

## **7. Reporting and Analytics**

Data should be collected and aggregated about donation tracking, inventory management, and user activity to help platform managers make strategic decisions.

## **8. Communication**

The platform should have the ability to send notifications and alerts to users in real-time, such as when a donation is received or when an item is close to expiry.

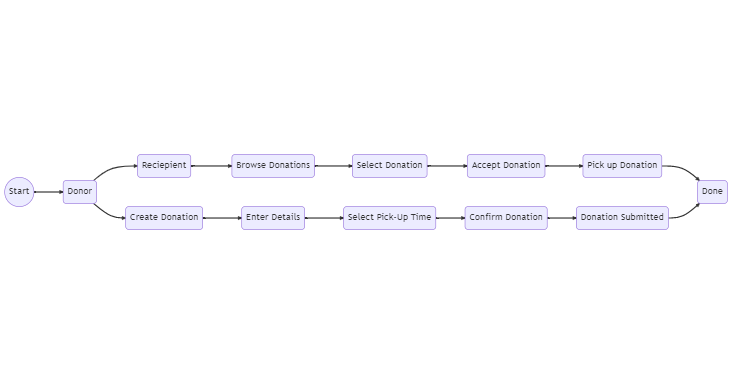
## **9. User Feedback**

The platform should have a feedback mechanism, such as in-app surveys, that allow donors and recipients to provide feedback that can help improve the platform over time.

By having the above system requirements features, a food donation platform can effectively manage donations, communicate with stakeholders, track inventory, improve user experiences, and ensure that donated food items reach those in need.

# 

# **Activity Diagram**



This diagram starts with a Start node and then flows into two paths, one for the Donor and one for the Volunteer. The Donor path includes creating a donation, entering the details, selecting a pick-up time, confirming the donation, and submitting it. The Volunteer path includes browsing donations, selecting a donation, accepting the donation, picking up the donation, and marking it as done. Both paths end at the Done node.

# 

# 

# 

# **Use Case Diagrams**

**Actors**: Donor, Platform, Recipient

**Scenario**: A donor wants to donate excess food items to a recipient in need.

**Description**

1. The donor signs in to the food donation platform and selects "Donate Food" from the main menu.

2. The platform displays a list of nearby recipients and their locations.

3. The donor selects the recipient to donate to and adds the food items to the platform, including details such as quantity, type, and expiration date.

4. The platform confirms the donation and generates a unique donation ID for tracking purposes.

5. The recipient receives a notification from the platform that a donation is available and visits the donor's donation location.

6. The recipient shows the donation ID to the donor and the donor provides the donated food items to the recipient.

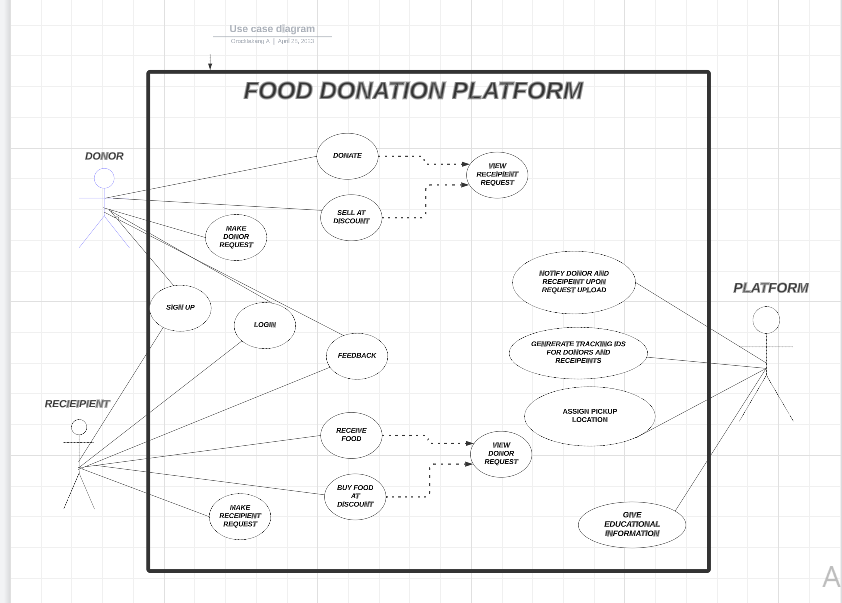
7. The recipient confirms the donation through the platform, indicating the reception of the food and the quality of the items received.

8. The platform updates the inventory levels for both the donor and recipient accordingly.

9. The donor receives a notification that the donation has been received and marked as complete by the recipient.

This use case demonstrates how a food donation platform can facilitate the process of donating excess food items to those in need. The platform provides transparency and accountability through the use of donation IDs and confirmation of receipt, as well as real-time inventory tracking to ensure that the donations are managed efficiently.

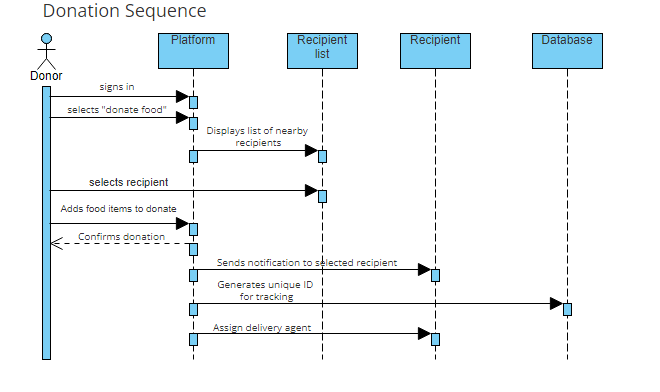
The use case diagram is as seen below:



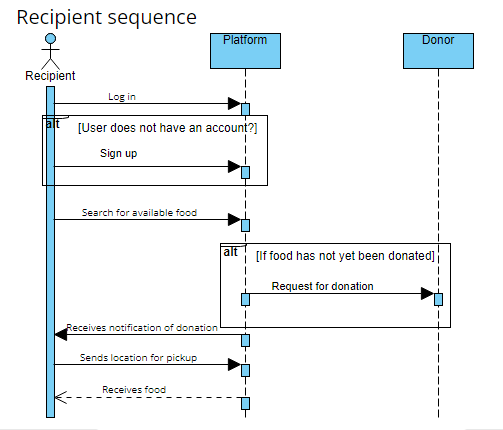
**Sequence diagrams**

The sequence diagrams have been drawn in sequences according to the stages constituting the platform.

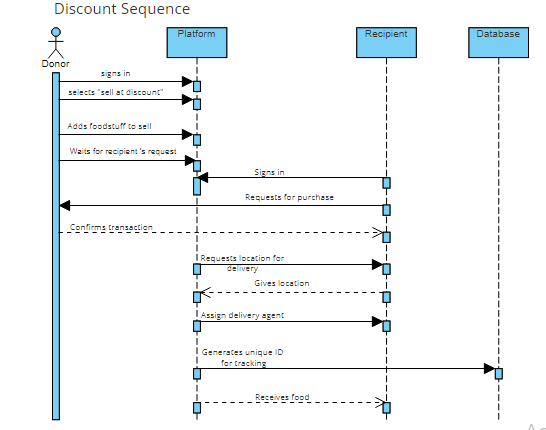
1. Donation sequence



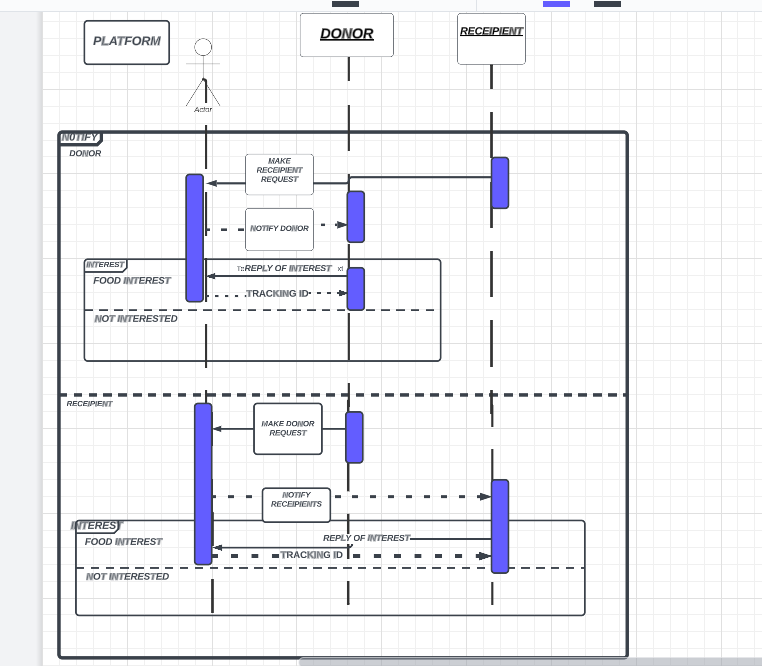
1. Recipient Sequence



1. Discount Sequence

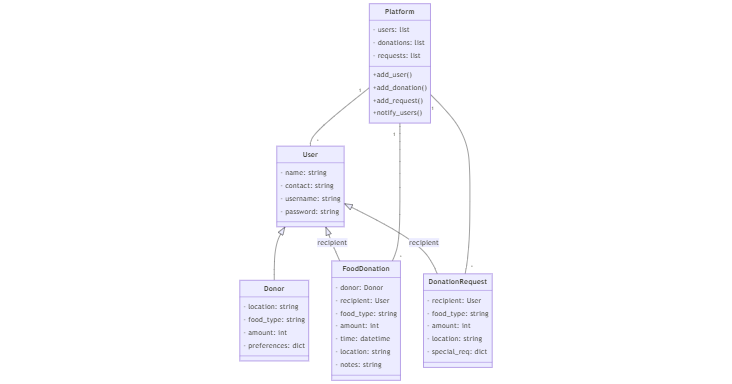


1. Platform notification sequence



# 

# **Class Diagrams**



The main classes in this system could include:

**User**: This class would represent any person or organisation that is using the platform to either donate or receive food. It could contain attributes such as name, contact information, and login credentials.

**Donor**: This class would be a subclass of User and would contain additional attributes specific to donors, such as their location, the type and amount of food they are donating, and any preferences they have for who should receive their donation.

**Recipient**: This class would also be a subclass of User and would contain attributes specific to recipients, such as their location, the type and amount of food they need, and any dietary restrictions or allergies they have.

**Food Donation**: This class would represent a single donation of food from a donor to a recipient. It could contain attributes such as the quantity and type of food being donated, the time and location of the donation, and any notes or comments from the donor or recipient.

**Donation Request**: This class would represent a request from a recipient for food donations. It could contain attributes such as the type and amount of food needed, the recipient's location, and any special requests or instructions.

**Platform**: This class would represent the online platform itself and could contain methods for managing users, donations, and donation requests, as well as for handling any communication or notifications between users.

These classes are connected through various relationships, such as a one-to-many relationship between Donor and Food Donation (since a single donor could make multiple donations), or a many-to-one relationship between Donation Request and Recipient (since multiple recipients could make requests for the same type of food).