RAJALAKSHMI ENGINEERING COLLEGE

RAJALAKSHMI NAGAR, THANDALAM – 602 105

RAJALAKSHMI
ENGINEERING COLLEGE

CB23332
SOFTWARE ENGINEERING LAB

Laboratory Record Note Book

Name : . . . . . .

Year / Branch / Section : . . . . . .

Register No. : . . . . . .

Semester : . . . . . .

Academic Year : . . . . . .

RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)
RAJALAKSHMI NAGAR, THANDALAM – 602-105

BONAFIDE CERTIFICATE

NAME:_____REGISTER NO.: _____

ACADEMIC YEAR: 2024-25 **SEMESTER:** III **BRANCH:_____**B.E/B.Tech

This Certification is the bonafide record of work done by the above student in the

CB23332-SOFTWARE ENGINEERING - Laboratory during the year 2024 – 2025.

Signature of Faculty -in – Charge

Submitted for the Practical Examination held on _____

Internal Examiner External Examiner

INDEX

S.No.	Name of the Experiment	Expt. Date	Faculty Sign
1.	Preparing Problem Statement		
2.	Software Requirement Specification (SRS)		
3.	Entity-Relational Diagram		
4.	Data Flow Diagram		
5.	Use Case Diagram		
6.	Activity Diagram		
7.	State Chart Diagram		
8.	Sequence Diagram		
9.	Collaboration Diagramt		
10.	Class Diagram		

EX NO:1	WRITE THE COMPLETE PROBLEM STATEMENT
DATE:	

AIM:

To prepare PROBLEM STATEMENT for Food donation management.

ALGORITHM:

1. Start.
2. Donor Registration: Input donor details (name, email, phone, address).Store details in the donor database.
3. Food Donation: Donor submits food details (type, quantity, expiration date).Validate food details for safety and quality checks.Store food information in the donation database.
4. Recipient Registration: Input recipient details (name, email, phone, address).Store recipient information in the database.
5. Food Request by Recipient: Recipient searches for available food donations.Match recipient's request with available donations based on proximity and requirements
6. Distribution Center Allocation: Assign the nearest distribution center to handle the delivery or pickup.Update delivery status in the database.
7. Delivery Management: Schedule and track delivery status (in transit, delivered). Update recipient's confirmation of food received.
8. End.

INPUT:

- Donor Data:

Name, contact information, address.

Food details: type, quantity, expiration date.

- Recipient Data:

Name, contact information, address.

Food request: type and quantity required.

- Distribution Center Data:

Name, location, and available logistics for delivery.

Problem:

Despite widespread hunger and food insecurity, a significant portion of food is wasted daily. Restaurants, households, and organizations often discard surplus edible food due to a lack of streamlined systems for donation. The problem is further exacerbated by logistical challenges, food safety concerns, and the absence of effective connections between donors and recipients. This project aims to solve this issue by building a **food waste donation system** that efficiently bridges the gap between food donors and recipients.

Background:

Food Wastage Problem: According to studies, approximately one-third of food produced globally is wasted annually, while millions face hunger and malnutrition.

Need for Digital Platforms: Existing manual donation processes lack proper tracking and logistical support, leading to inefficiencies. A digital platform can provide a real-time, systematic approach to managing donations.

Safety and Quality Concerns: Ensuring donated food is safe and suitable for consumption is vital to prevent health issues. The system incorporates validation steps to address this.

Relevance:

The food waste donation system is critical in addressing global food security challenges:

1. Reduces food wastage and environmental impact.
2. Provides a platform to connect donors and recipients efficiently.
3. Enhances community support and collaboration to combat hunger.
4. Promotes social responsibility among donors.

Objectives:

1. Efficient Food Donation Platform:
    a. Develop a user-friendly web and mobile application for registering donors, recipients, and distribution centers.
2. Minimizing Food Waste:
    a. Reduce the amount of edible food being discarded through real-time donation tracking.
3. Ensuring Food Safety:
    a. Incorporate safety validation processes for donated food, including expiration dates and conditions.
4. Streamlined Logistics:
    a. Integrate logistics for efficient food distribution, including delivery scheduling and status tracking.
5. Data Analysis and Insights:
    a. Provide insights and reports on donation patterns, food wastage reduction, and the number of people benefitted
6. Scalability and Accessibility:
    a. Build a scalable platform that supports multiple donors, recipients, and locations, accessible across devices.

RESULT:

EX NO:2	WRITE THE SOFTWARE REQUIREMENT SPECIFICATION DOCUMENT
DATE:	

AIM:

To do requirement analysis and develop Software Requirement Specification Sheet(SRS) for food waste donation.

ALGORITHM:

Requirement Gathering

1. Identify stakeholders: donors, recipients, administrators, and distribution centers.
2. Gather requirements through surveys, interviews, and questionnaires to understand user needs.

Define System Functionality

Document what the system is supposed to do:

1. Allow donors to register and log food donations.
2. Provide recipients access to available food donations.
3. Match donations with recipients based on proximity and needs.
4. Facilitate safe and efficient logistics for delivery or pickup.

Address External Interfaces

a) User Interface Requirements:

1. Design intuitive interfaces for donors, recipients, and administrators.
2. Ensure accessibility on web and mobile devices.

b) Hardware Interfaces:

1. Define required hardware (e.g., servers, client devices).
2. Specify compatibility with devices (desktops, tablets, and smartphones).

c) Software Interfaces:

1. Specify integration with databases, payment gateways (if needed), and external APIs for maps or logistics.

Performance Requirements

Define metrics for:

2. Speed: Real-time updates for donations and requests.
3. Availability: Ensure the system operates 24/7 with minimal downtime.
4. Response Time: Quick processing of user queries and actions.
5. Recovery Time: Outline recovery methods in case of system failure.

Specify Software Attributes

a) Portability: Ensure the system is compatible across devices and platforms.
b) Correctness: Validate data entry and ensure accuracy in matching donors and recipients.
c) Maintainability: Enable regular updates, bug fixes, and feature enhancements.
d) Security: Implement authentication, authorization, and encryption for secure transactions.

Identify Design Constraints

a) Standards:

Adhere to food safety regulations and local laws for food donations.

b) Implementation Language:

Specify programming languages (e.g., Python, JavaScript) and frameworks.

c) Database Policies:

Ensure data integrity, normalization, and backups.

d) Resource Limits:

Consider storage, server capacity, and bandwidth.

e) Operating Environment:

Define operating systems (e.g., Windows, Linux) and cloud platforms (AWS, Azure).

Review and Validate

1. Review the SRS document with stakeholders to ensure alignment with expectations.
2. Validate the feasibility of requirements with technical and financial constraints.

Finalize SRS

1. Document all the requirements comprehensively.
2. Finalize and circulate the SRS for development and implementation.

1. Introduction

1.1 PURPOSE

The purpose of this document is to build an online system that facilitates the donation of surplus food from donors (individuals, restaurants, and businesses) to food banks, shelters, and other receivers. The goal of the project is to reduce food waste by creating a streamlined process for connecting donors with organizations that can make use of the donated food. This platform will also track donations and provide logistical support for pickups and deliveries.

1.2 DOCUMENT CONVENTIONS

This document follows these conventions:

- **DB**: Database
- **API**: Application Programming Interface
- **UI**: User Interface
- **GPS**: Global Positioning System
- **SSL**: Secure Socket Layer
- **ER**: Entity Relationship

1.3 INTENDED AUDIENCE AND READING SUGGESTIONS

This project is intended for stakeholders, developers, project managers, and food donation organizations that are interested in understanding the functional and non-functional requirements of the system. The system will be used by donors, food banks, NGOs, and logistics partners. Developers should focus on sections detailing system architecture and functional specifications, while stakeholders may find the scope and overall features useful in understanding the platform's impact.

1.4 PROJECT SCOPE

The purpose of the food waste donation website is to provide a simple, user-friendly system for donors to easily schedule food pickups, and for receivers to manage incoming food donations. The system is web-based and will eventually be expanded to cover various cities and regions. The platform will allow users to:

- Register as a donor or receiver
- Schedule donations with real-time updates and tracking
- Match donors with nearby food banks or charities based on their location
- Receive notifications on donation status, pickup, and delivery The platform aims to ensure food donations are efficiently managed, reducing food waste while addressing hunger in local communities.

2. Overall Description

2.1 PRODUCT PERSPECTIVE

The food waste donation platform is a web-based system designed to connect food donors with food receivers. It stores and manages the following information:

- **Donation Details**:
  - Information about the food being donated, including type (e.g., fruits, vegetables, packaged goods), quantity, expiration dates, and condition.
  - Location of the donor, pickup address, and delivery address for food receivers.
  - Status of each donation, including whether it is pending pickup, in transit, or delivered.
- **Donor Description**:
  - Donor profiles contain essential details such as donor type (individual, business, or restaurant), contact name, address, phone number, and email. This information is used to schedule pickups and provide notifications for updates.
- **Receiver Description**:
  - Food receiver profiles, including organization name (food bank, shelter, NGO), location, type of food accepted, and contact information. This helps ensure that donations meet the needs of each receiving organization.
- **Pickup and Delivery Details**:
  - Scheduled pickup time, location, and delivery destination for the food donations.
  - Tracking information to monitor the status of food donations as they move from the donor to the receiver.

- Logistics information, including third-party transport services used for pickup and delivery.

2.2 PRODUCT FEATURES

The major features of the food waste donation system include:

- **Donor and Receiver Registration**:
  - Donors and receivers can register through the platform, providing necessary information for verification.
  - User-friendly registration forms with simple steps for onboarding.
- **Donation Scheduling and Management**:
  - Donors can create new donation entries by specifying the type and quantity of food and scheduling a pickup.
  - The system will automatically match the donor with nearby food banks or receivers based on geolocation.
  - Real-time updates on donation status (pending, picked up, in transit, delivered).
- **Tracking and Notifications**:
  - Both donors and receivers can track donations through the platform's dashboard.
  - Email and SMS notifications will be sent to donors and receivers for major events, including pickup confirmation and delivery.
- **Search and Filter Functionality**:
  - Donors can search for nearby food banks or shelters based on their location.
  - Receivers can post specific requests for food types they need, which will be shown to nearby donors.
- **Logistics Integration**:
  - The system integrates with third-party logistics providers to facilitate pickup and delivery services.
  - Pickup agents can scan QR codes for food items to log the donations as picked up and delivered.

2.4 OPERATING ENVIRONMENT

The operating environment for the food waste donation system is as listed below:

- **Distributed Database**: The system operates on a distributed database architecture to handle donations, user information, and tracking data across different locations.
- **Client/Server System**: The platform is based on a client-server architecture. Users interact with the system through a web browser (client), while the back-end server manages the database, business logic, and API integrations.
- **Operating System**:
  - The system is hosted on cloud servers, which can run on multiple operating systems such as Linux (Ubuntu, CentOS) or Windows Server, depending on the hosting environment.
  - For development and local testing, developers may use environments like Windows, macOS, or Linux.
- **Database**:
  - The system will use a relational database (e.g., MySQL or PostgreSQL) to store donor/receiver information, food donation details, and tracking information.
  - The database should support real-time queries to enable efficient matching of donors with receivers based on location.
- **Web Technologies**:
  - **Frontend**: The client-side interface will be built using standard web technologies such as HTML, CSS, JavaScript, and potentially a framework like React.js or Angular.
  - **Backend**: The server-side logic will be implemented using a technology stack such as Node.js, Django (Python), or Ruby on Rails, depending on development choices.
- **Security**: The platform will use **SSL/TLS** encryption for secure communication between clients and the server.

2.5 DESIGN AND IMPLEMENTATION CONSTRAINTS

1. **Database Schema**:
    - The system will operate using a distributed database architecture. The **global schema** will cover the overall data model, including entities like Donor, Receiver, Donation, Pickup, and Logistics.
    - **Fragmentation schema**: The data will be fragmented based on geographical locations to ensure efficient data access and processing. For instance, food donations and users will be stored in regional databases (e.g., North, South, East, West regions).
    - **Allocation schema**: Each regional database will handle requests for donations or receivers within its geographical area. Queries that require cross-region data (e.g., when logistics partners need to coordinate across regions) will be handled by combining data from multiple fragments.
2. **SQL Queries**:
    - The system will use SQL commands to handle standard queries such as:
        - Finding nearby food banks for a particular donor.
        - Matching a donation with a food receiver based on location, type of food, and available logistics.
        - Querying historical donations to track patterns or performance over time.
3. **Global Queries**:
    - For global queries, such as determining the total number of donations made across all regions, the system will need to combine fragments from multiple regional databases. This can be done by federating queries across the distributed database and aggregating the results.
4. **Centralized DBMS**:
    - While the system is designed to be distributed for scalability and performance, a centralized database can also be implemented for smaller-scale operations. In this case, all data will reside in a single database, and queries will not need to be federated across multiple regions.

2.6 ASSUMPTIONS AND DEPENDENCIES

The food waste donation system assumes the following:

- **Assumption 1**:
    - The system is distributed across multiple cities, each managing donations and food receivers in its region. Users will primarily interact with the system based on their location, though cross-region coordination (e.g., transporting food from one city to another) is possible.
- **Assumption 2**:
    - Donors can submit requests for food donations at any time. The system will then match the request with a nearby receiver. If no local receivers are available, it will suggest the nearest available food banks in other regions.
- **Assumption 3**:
    - Logistics partners can access the system to manage pickups and deliveries across regions. Real-time tracking will depend on integrations with third-party delivery services.
- **Assumption 4**:
    - Users (both donors and receivers) depend on the system for real-time updates on donation status. The system assumes that donors and receivers will keep their contact information (email, phone) updated for notifications.

3. SYSTEM FEATURES

DESCRIPTION AND PRIORITY

The food waste donation system tracks food donations, food receivers, and the logistics involved in getting the food from the donor to the receiver. The system is high-priority as it helps reduce food waste and provide meals to those in need by facilitating the donation process efficiently.

STIMULUS/RESPONSE SEQUENCES

- **Donor Action: Scheduling a Donation**
  - **Stimulus**: A donor creates a donation request, providing details of the food and scheduling a pickup time.
  - **Response**: The system matches the donor with the nearest food bank or receiver and provides the donor with an estimated pickup time. The system sends a confirmation message to the donor and the receiver.
- **Receiver Action: Receiving a Donation**
  - **Stimulus**: A food bank receives a notification of an incoming donation, including the type and quantity of food.
  - **Response**: The system tracks the progress of the donation, from pickup to delivery. The receiver confirms the receipt of the donation once it has been delivered.
- **Logistics Integration**
  - **Stimulus**: A logistics partner is assigned to pick up a donation.
  - **Response**: The system tracks the pickup in real-time and updates both the donor and receiver when the donation is on its way.

3.1 FUNCTIONAL REQUIREMENTS

The primary functional requirements for the food waste donation system are as follows:

Distributed Database:

The system operates using a distributed database model, enabling the management of food donations across multiple cities. This ensures that donation and receiver data is localized for efficient processing but can be accessed across the network as needed. Each region, such as North, South, East, and West, will have its own database, as shown in the diagram (similar to the one you provided).

Details stored include:

- **Donor Information**: Name, address, contact details, and donation history.
- **Receiver Information**: Name, type of organization, contact details, and food needs.
- **Logistics**: Transportation data to manage food pickups and deliveries.

Client/Server System:

The system will function in a client/server architecture where:

- **Client**: The user interface for donors, receivers, and logistics partners (front-end).
- **Server**: The backend database that processes donation requests, matches donors with receivers, and manages logistics (DBMS).

This setup allows users to make donation requests, track pickups, and receive real-time updates on the status of their donations.

4. EXTERNAL INTERFACE REQUIREMENTS

4.1 USER INTERFACES

- **Front-end software**:
  The system's front-end will be built using **Vb.net version**, allowing for an interactive and user-friendly interface for both donors and receivers to manage their activities.
- **Back-end software**:
  The back-end will use **SQL+**, which will store donation details, donor/receiver information, and logistics data securely.

4.2 HARDWARE INTERFACES

- **Operating System**: The system is compatible with **Windows** platforms, providing a stable environment for users.
- **Browser Compatibility**: Any web browser that supports **CGI, HTML**, and **JavaScript** will be used for accessing the system. This allows for flexible use across devices that support these standards.

4.3 SOFTWARE INTERFACES

The following software components are used in the food waste donation system:

Software Used	Description
Operating System	We have chosen **Windows OS** for its ease of use and broad compatibility with necessary development tools.
Database	To store records of food donations, donors, and receivers, the system uses **SQL+** for its reliability in handling structured data efficiently.
VB.Net	The **Vb.Net** language has been chosen to implement the project due to its strong support for developing interactive and robust user interfaces.

4.4 COMMUNICATION INTERFACES

The food waste donation system supports all modern web browsers, enabling users to access the platform from any device that can connect to the internet. The communication between the client (user) and the server is facilitated through **electronic forms** that handle the following tasks:

- **Donation forms**: Donors can easily submit food donation details.
- **Request forms**: Receivers can request specific types of food donations based on their needs.
- **Tracking forms**: Both donors and receivers can track the status of ongoing donations and logistics.

5. NONFUNCTIONAL REQUIREMENTS

5.1 PERFORMANCE REQUIREMENTS

The food waste donation system is designed to meet performance expectations in the following ways:

- **Data Representation (E-R Diagram)**:
  An **E-R (Entity-Relationship) Diagram** represents the logical structure of the system's database.

- This diagram helps in organizing data into relations (tables), which are then normalized to ensure database efficiency.

  Entities:

  - **Donor**: Represents individuals or organizations contributing food.
  - **Receiver**: Represents organizations receiving food donations.
  - **Logistics**: Represents transport entities managing the pick-up and delivery of donations.

  Properties/Attributes:
  Each entity has associated attributes, such as:

  - **Donor**: Name, contact information, donation history.
  - **Receiver**: Organization name, type, food requirements.
  - **Logistics**: Transportation details, pick-up and delivery schedules.

  Relationships:
  Relationships represent meaningful connections between entities, such as:

  - **Donor-Donation**: Links the donor to the food donations made.
  - **Receiver-Request**: Links the receiver to the food requested.
  - **Logistics-Delivery**: Links logistics providers to the transport of donations.

The ER diagram for the system would visually depict how each of these entities connects, ensuring a smooth flow of data and efficient processing of donation transactions.

5.2 SAFETY REQUIREMENTS:
In the event of a major failure, such as a server crash, the food waste donation system must have a robust recovery plan. The recovery process should restore the most recent version of the database using regular backups, stored in archival storage. The system should also have transaction logs that can be used to restore the most recent data, ensuring no data is lost during a crash or emergency.

5.3 SECURITY REQUIREMENTS:
Given that the food waste donation system stores sensitive data, such as personal information of donors, recipients, and other stakeholders, robust security mechanisms must be implemented. Access to the database should be restricted to authorized personnel only, and encryption should be used to protect sensitive data. Regular audits and monitoring can also ensure that any unauthorized access or anomalies are detected promptly.

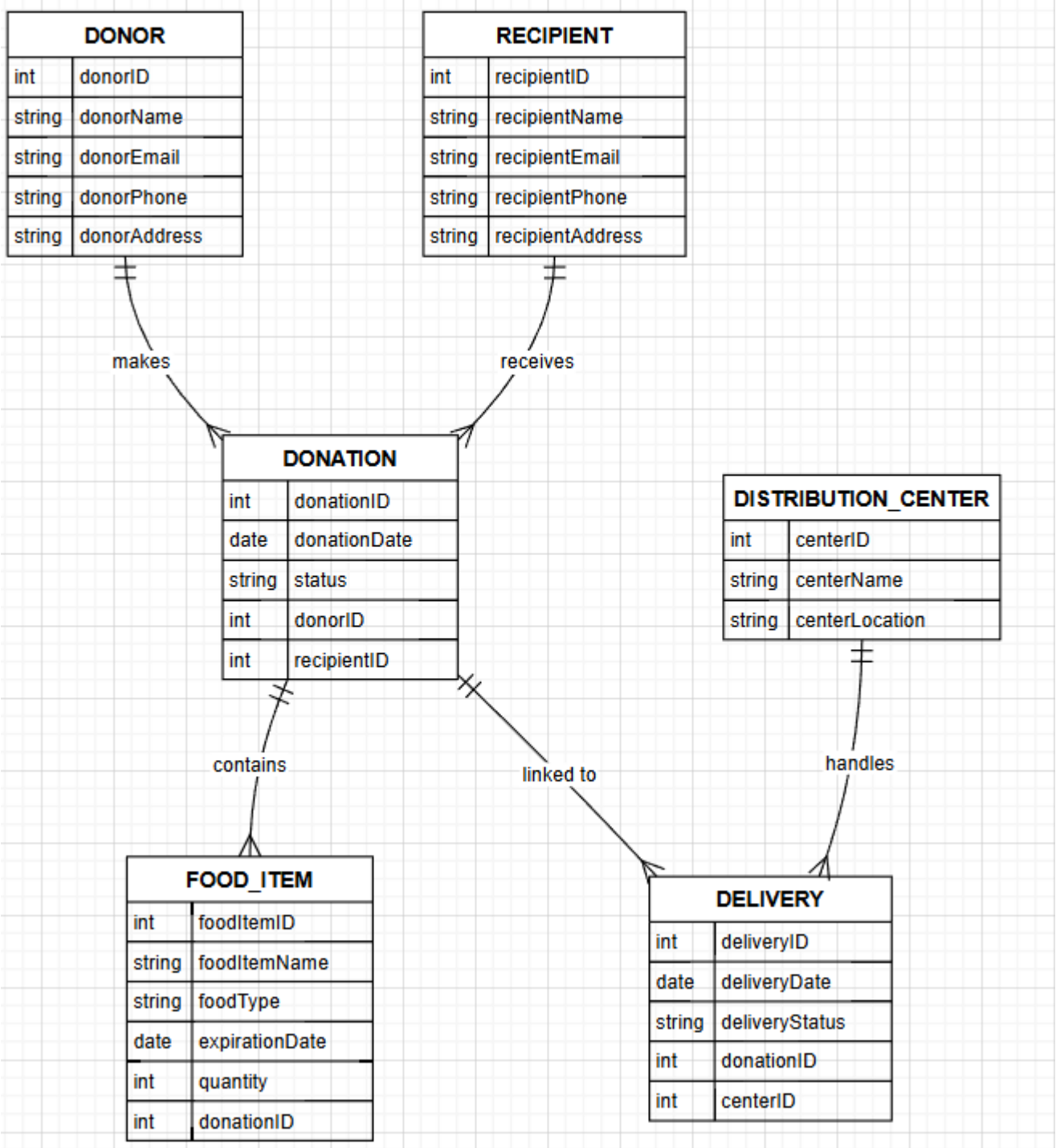
5.4 SOFTWARE QUALITY ATTRIBUTES:

- **Availability:** The system must be available 24/7 to ensure that donors can input data and recipients can access it at any time. Given the time-sensitive nature of food donations, the system should have minimal downtime.
- **Correctness:** The information about donations, such as quantity, type of food, and location of collection, must be accurate to ensure that the right food reaches the right people in a timely manner. Miscommunication could lead to food wastage or delivery issues.

MAINTAINABILITY:
The food waste donation system should be designed to support regular updates for compatibility with the latest technologies. This includes updates to the web application for improved performance, bug fixes, and the application of security patches. The architecture should allow for future scalability, ensuring that new features, such as integration with other platforms (e.g., delivery services or food safety organizations), can be added easily without major disruptions to the existing system.

USABILITY:
The food waste donation system should cater to a wide range of users, from individual donors to organizations and recipients. The application should be accessible across multiple devices, including desktops, tablets, and mobile phones, to accommodate users' varying preferences. The interface must be intuitive, enabling users to quickly input donation information, locate food distribution points, and track donations. Furthermore, the system should allow for clear navigation and provide features like donation scheduling and progress tracking to enhance the user experience.

RESULT:


DONOR
int donorID
string donorName
string donorEmail
string donorPhone
string donorAddress
RECIPIENT
int recipientID
string recipientName
string recipientEmail
string recipientPhone
string recipientAddress
makes
receives
DONATION
int donationID
date donationDate
string status
int donorID
int recipientID
DISTRIBUTION_CENTER
int centerID
string centerName
string centerLocation
contains
linked to
handles
FOOD_ITEM
int foodItemID
string foodItemName
string foodType
date expirationDate
int quantity
int donationID
DELIVERY
int deliveryID
date deliveryDate
string deliveryStatus
int donationID
int centerID

EX NO:3	DRAW THE ENTITY RELATIONSHIP DIAGRAM
DATE:	

AIM:

To Draw the Entity Relationship Diagram for Food Donation System.

ALGORITHM:

Step 1: Mapping of Regular Entity Types

Step 2: Mapping of Weak Entity Types

Step 3: Mapping of Binary 1:1 Relation Types

Step 4: Mapping of Binary 1:N Relationship Types.

Step 5: Mapping of Binary M:N Relationship Types.

Step 6: Mapping of Multivalued attributes.

INPUT:

Entities

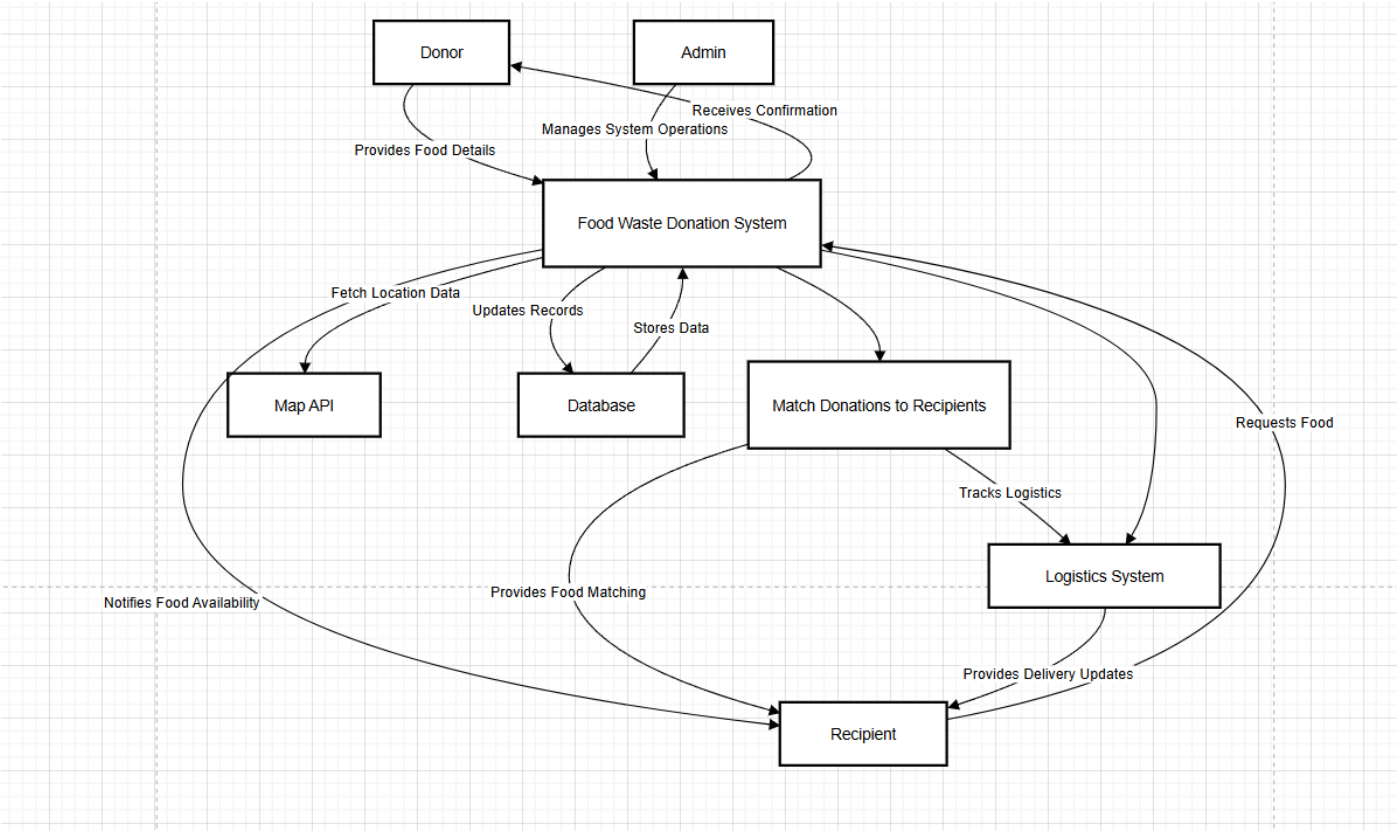
Entity Relationship Matrix

Primary Keys

Attributes

Mapping of Attributes with Entities

RESULT:


Donor
Admin
Receives Confirmation
Manages System Operations
Provides Food Details
Food Waste Donation System
Fetch Location Data
Updates Records
Stores Data
Map API
Database
Match Donations to Recipients
Requests Food
Tracks Logistics
Logistics System
Notifies Food Availability
Provides Food Matching
Provides Delivery Updates
Recipient

EX NO:4	DRAW THE DATA FLOW DIAGRAMS AT LEVEL 0 AND LEVEL 1
DATE:	

AIM:

To Draw the Data Flow Diagram for Food Donation System and List the Modules in the Application.

ALGORITHM:

1. Open the Visual Paradigm to draw DFD (Ex.Lucidchart)
2. Select a data flow diagram template
3. Name the data flow diagram
4. Add an external entity that starts the process
5. Add a Process to the DFD
6. Add a data store to the diagram
7. Continue to add items to the DFD
8. Add data flow to the DFD
9. Name the data flow
10. Customize the DFD with colours and fonts
11. Add a title and share your data flow diagram

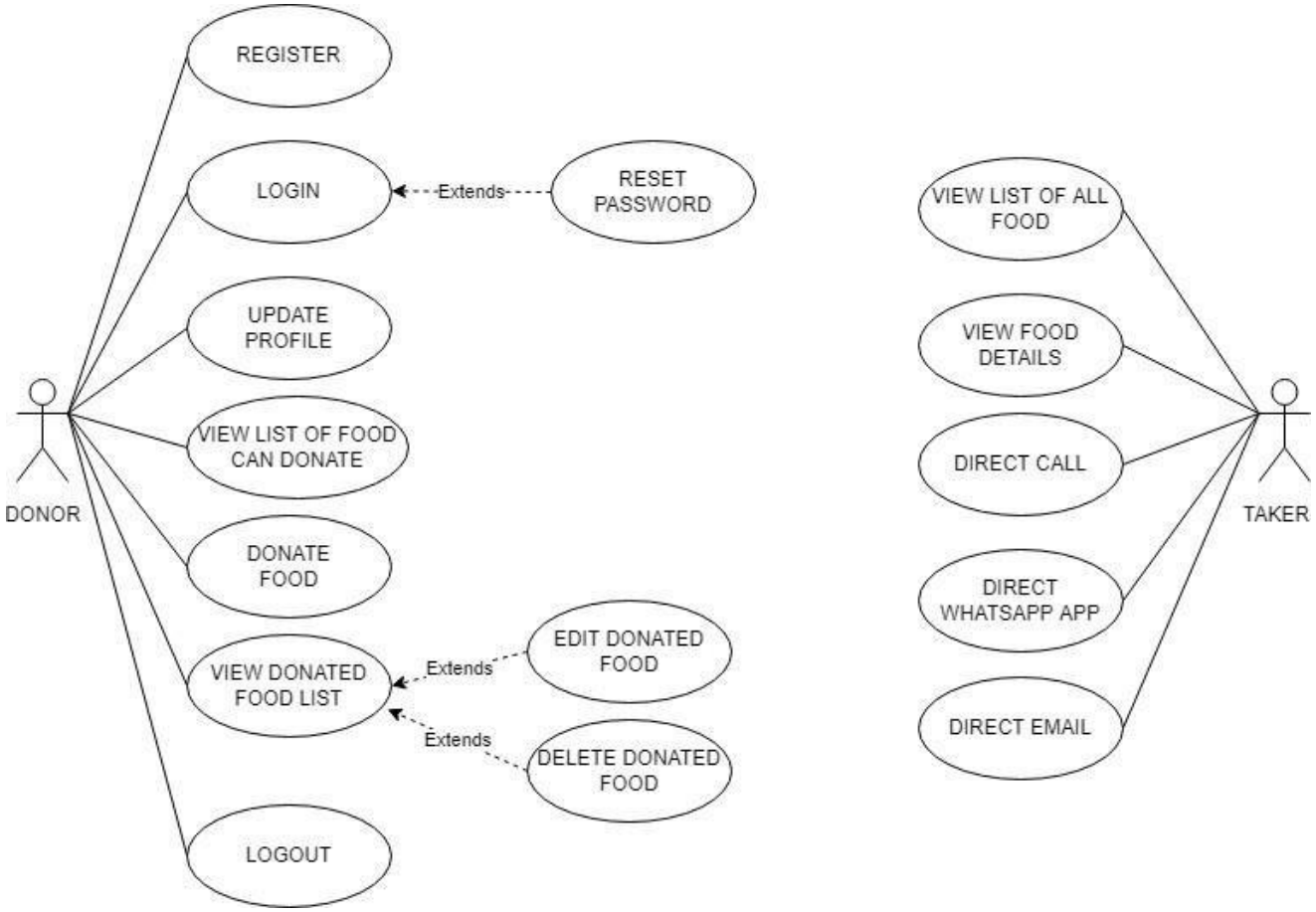
INPUT:

Processes

Datastores

External Entities

RESULT:


REGISTER
LOGIN
Extends
RESET PASSWORD
UPDATE PROFILE
VIEW LIST OF FOOD CAN DONATE
DONATE FOOD
VIEW DONATED FOOD LIST
Extends
EDIT DONATED FOOD
Extends
DELETE DONATED FOOD
LOGOUT
DONOR
VIEW LIST OF ALL FOOD
VIEW FOOD DETAILS
DIRECT CALL
DIRECT WHATSAPP APP
DIRECT EMAIL
TAKER

EX NO:5	DRAW USE CASE DIAGRAM
DATE:	

AIM:

To Draw the Use Case Diagram for Food Donation System.

ALGORITHM:

Step 1: Identify Actors

Step 2: Identify Use Cases

Step 3: Connect Actors and Use Cases

Step 4: Add System Boundary

Step 5: Define Relationships

Step 6: Review and Refine

Step 7: Validate

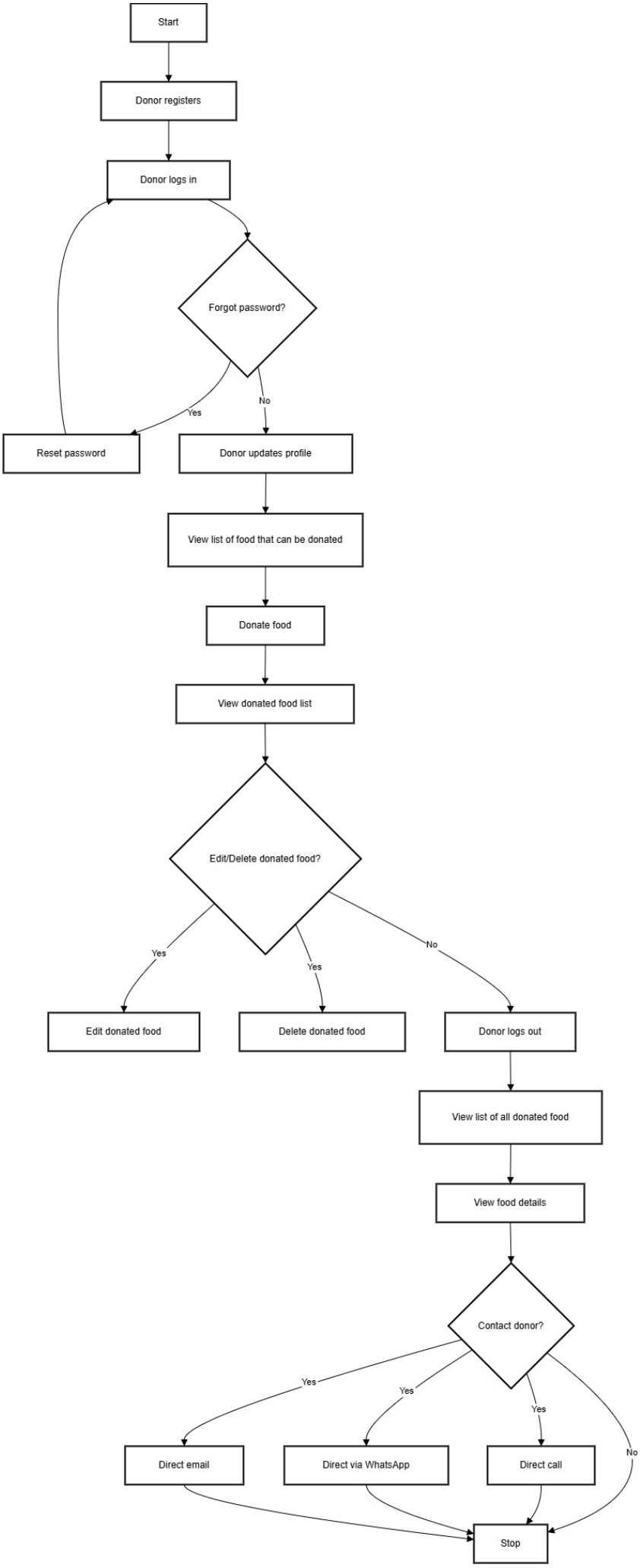
INPUTS:

Actors

Use Cases

Relations

RESULT:


Start
Donor registers
Donor logs in
Forgot password?
Yes
No
Reset password
Donor updates profile
View list of food that can be donated
Donate food
View donated food list
Edit/Delete donated food?
Yes
Yes
No
Edit donated food
Delete donated food
Donor logs out
View list of all donated food
View food details
Contact donor?
Yes
Yes
Yes
No
Direct email
Direct via WhatsApp
Direct call
Stop

EX NO:6	DRAW ACTIVITY DIAGRAM OF ALL USE CASES.
DATE:	

AIM:

To Draw the activity Diagram for Food Donation System.

ALGORITHM:

Step 1: Identify the Initial State and Final States

Step 2: Identify the Intermediate Activities Needed

Step 3: Identify the Conditions or Constraints

Step 4: Draw the Diagram with Appropriate Notations

INPUTS:

Activities

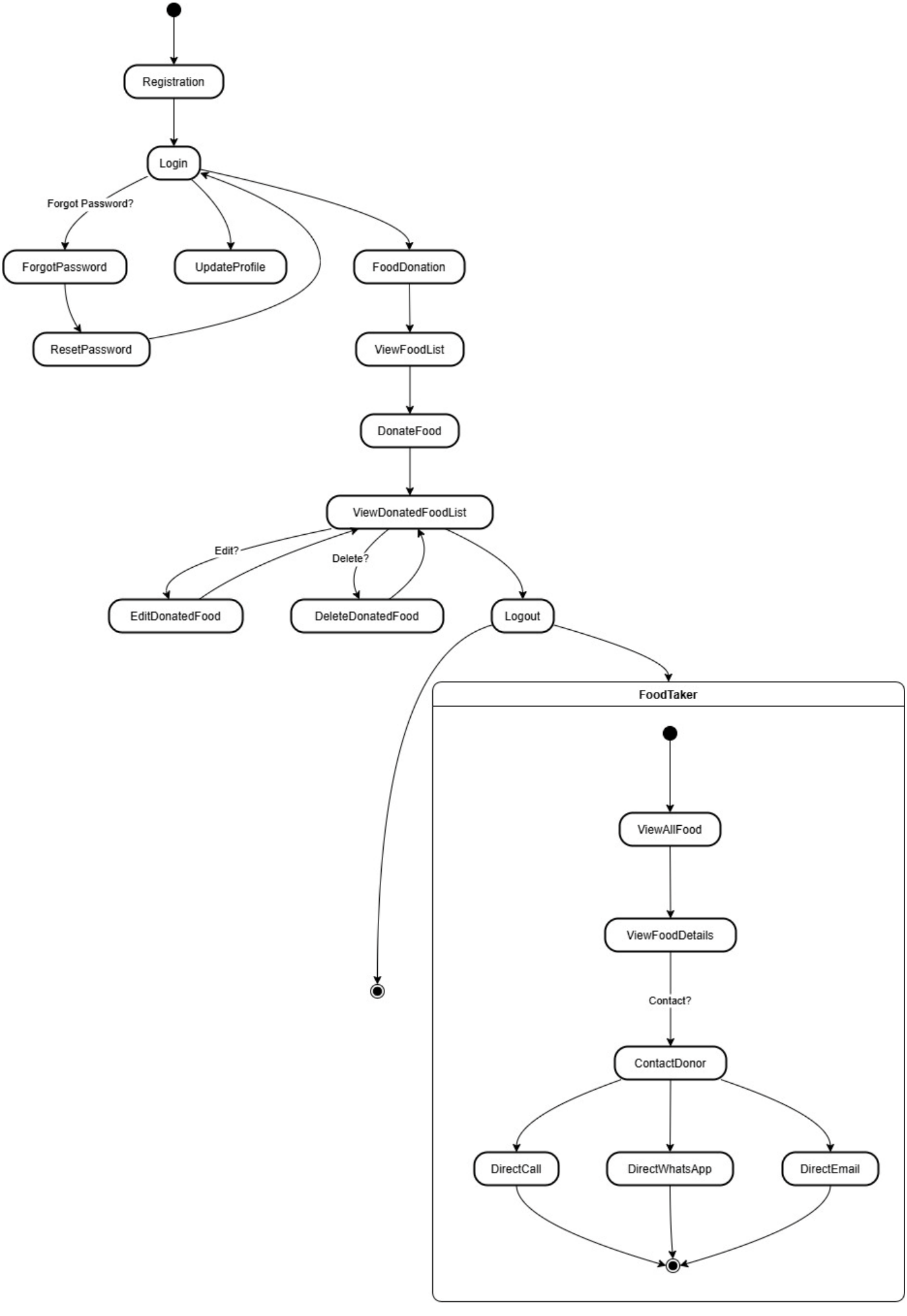
Decision Points

Guards

Parallel Activities

Conditions

RESULT:


Registration
Login
Forgot Password?
ForgotPassword
UpdateProfile
FoodDonation
ResetPassword
ViewFoodList
DonateFood
ViewDonatedFoodList
Edit?
Delete?
EditDonatedFood
DeleteDonatedFood
Logout
FoodTaker
ViewAllFood
ViewFoodDetails
Contact?
ContactDonor
DirectCall
DirectWhatsApp
DirectEmail

EX NO:7	DRAW STATE CHART DIAGRAM OF ALL USE CASES.
DATE:	

AIM:

To Draw the State Chart Diagram for Food Donation System.

ALGORITHM:

STEP-1: Identify the important objects to be analysed.

STEP-2: Identify the states.

STEP-3: Identify the events.

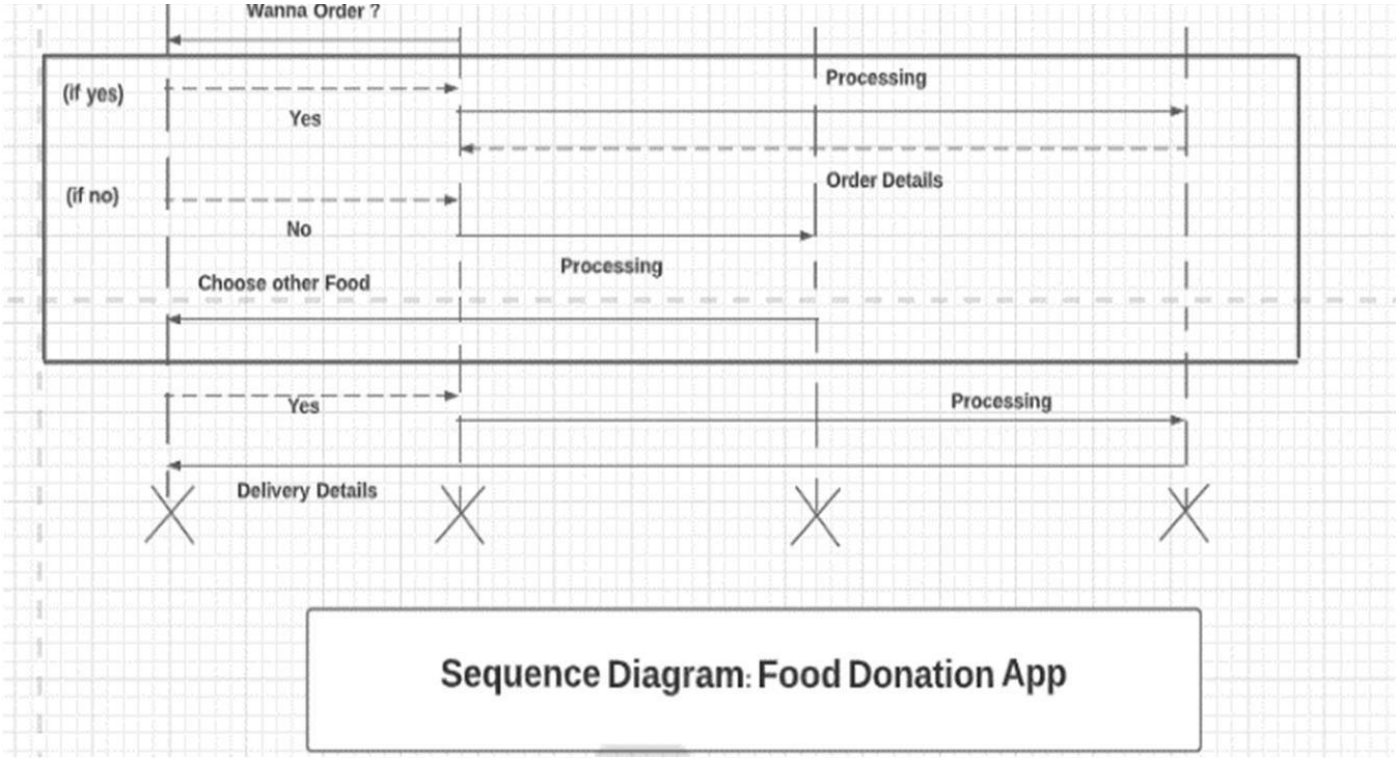
INPUTS:

Objects

States

Events

Result:



Sequence Diagram: Food Donation App

EX NO:8	DRAW SEQUENCE DIAGRAM OF ALL USE CASES.
DATE:	

AIM:

To Draw the Sequence Diagram for Food Donation System **.**

ALGORITHM:

1. Identify the Scenario
2. List the Participants
3. Define Lifelines
4. Arrange Lifelines
5. Add Activation Bars
6. Draw Messages
7. Include Return Messages
8. Indicate Timing and Order
9. Include Conditions and Loops
10. Consider Parallel Execution
11. Review and Refine
12. Add Annotations and Comments
13. Document Assumptions and Constraints
14. Use a Tool to create a neat sequence diagram

INPUTS:

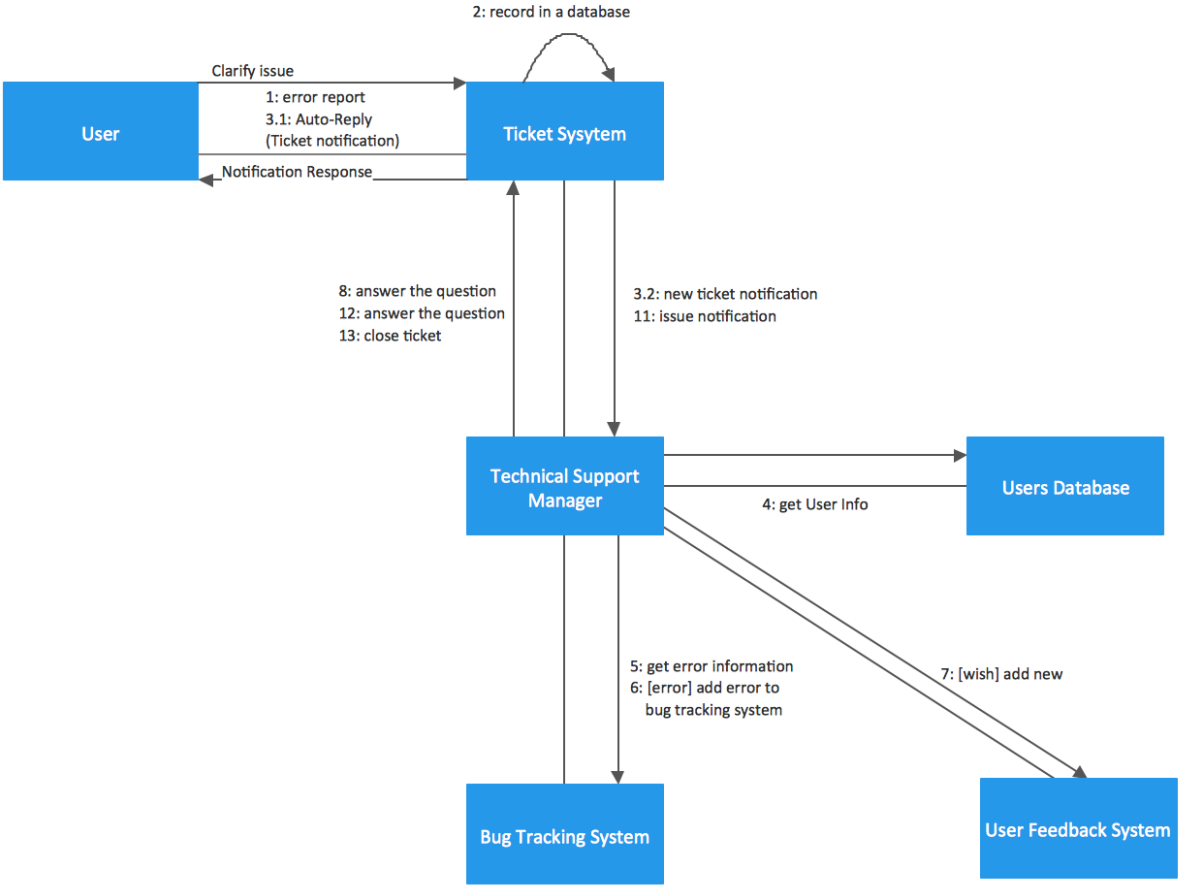
Objects taking part in the interaction.

Message flows among the objects.

The sequence in which the messages are flowing.

Object organization.

Result:


2: record in a database
Clarify issue
User
1: error report
3.1: Auto-Reply
(Ticket notification)
Ticket Sysytem
Notification Response
8: answer the question
12: answer the question
13: close ticket
3.2: new ticket notification
11: issue notification
Technical Support Manager
Users Database
4: get User Info
5: get error information
6: [error] add error to bug tracking system
7: [wish] add new
Bug Tracking System
User Feedback System

EX NO:9	DRAW COLLABORATION DIAGRAM OF ALL USE CASES
DATE:	

AIM:

To Draw the Collaboration Diagram for any project

ALGORITHM:

Step 1: Identify Objects/Participants

Step 2: Define Interactions

Step 3: Add Messages

Step 4: Consider Relationships

Step 5: Document the collaboration diagram along with any relevant explanations or annotations.

INPUTS:

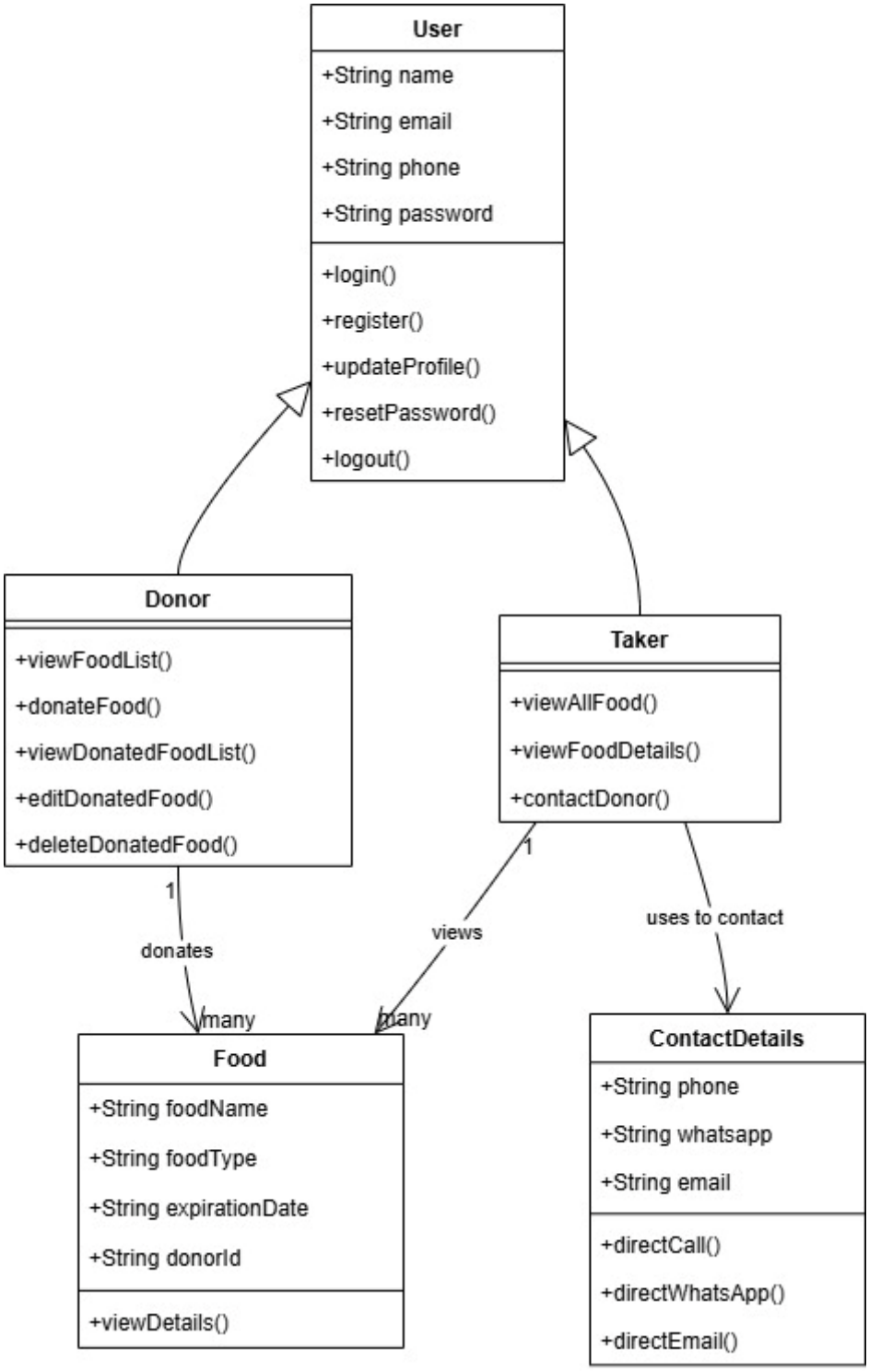
Objects taking part in the interaction.

Message flows among the objects.

The sequence in which the messages are flowing.

Object organization.

RESULT:


User
+String name
+String email
+String phone
+String password
+login()
+register()
+updateProfile()
+resetPassword()
+logout()
Donor
+viewFoodList()
+donateFood()
+viewDonatedFoodList()
+editDonatedFood()
+deleteDonatedFood()
Taker
+viewAllFood()
+viewFoodDetails()
+contactDonor()
1
donates
many
1
views
many
uses to contact
Food
+String foodName
+String foodType
+String expirationDate
+String donorId
+viewDetails()
ContactDetails
+String phone
+String whatsapp
+String email
+directCall()
+directWhatsApp()
+directEmail()

EX NO:10	ASSIGN OBJECTS IN SEQUENCE DIAGRAM TO CLASSES AND MAKE CLASS DIAGRAM.
DATE:	

AIM:

To Draw the Class Diagram for Food Donation System.

ALGORITHM:

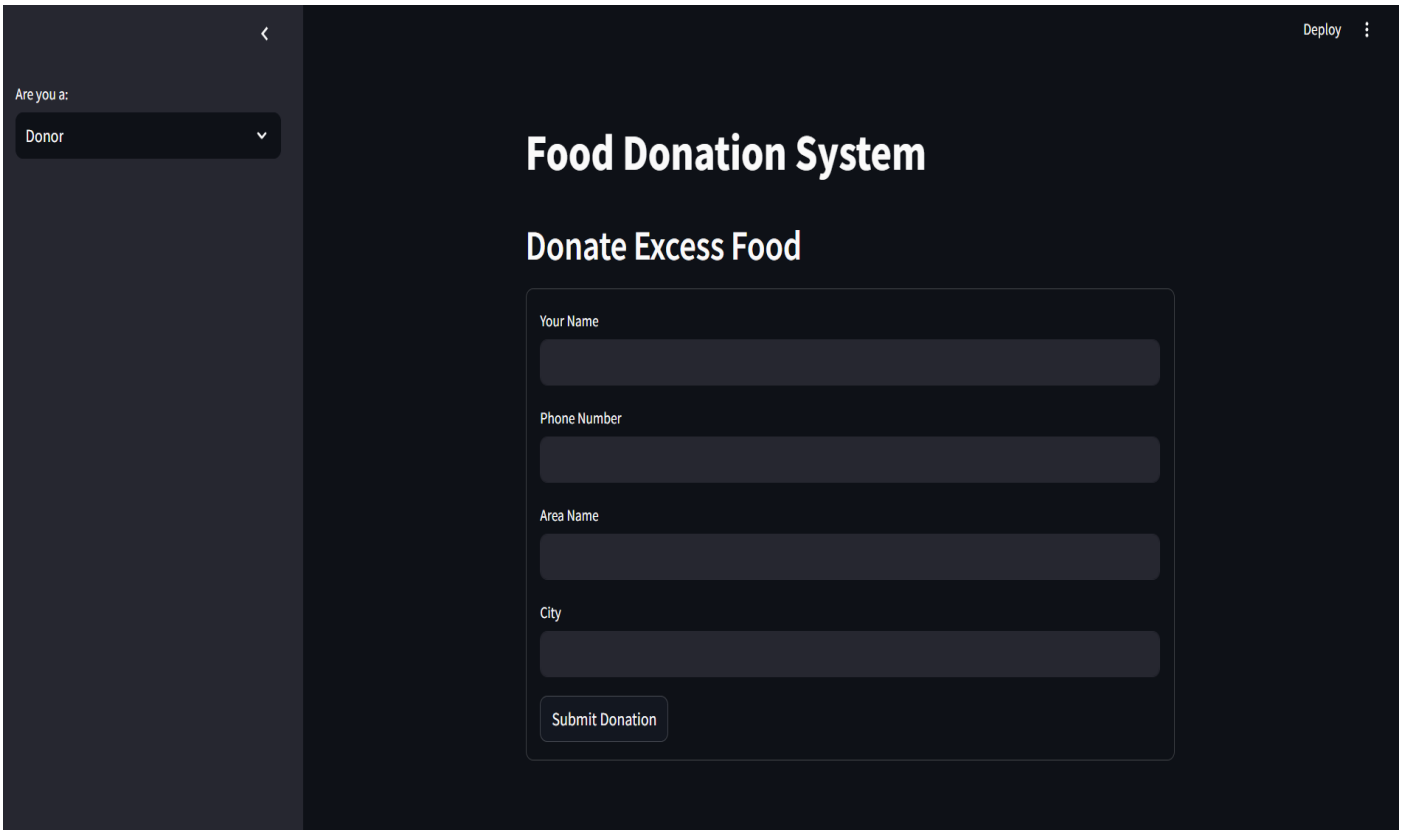
1. Identify Classes
2. List Attributes and Methods
3. Identify Relationships
4. Create Class Boxes
5. Add Attributes and Methods
6. Draw Relationships
7. Label Relationships
8. Review and Refine
9. Use Tools for Digital Drawing

INPUTS:

1. Class Name
2. Attributes
3. Methods
4. Visibility Notation

RESULT:

DONOR INTERFACE:



EX NO:11	MINI PROJECT- FOOD DONATION APP
DATE	

AIM:

The aim of the provided code is to create a simple web application for a Food Donation System using Streamlit. This application allows users to either donate excess food (as donors) or find available food donations (as receivers). The application facilitates the connection between donors and receivers, promoting food sharing and minimizing waste.

ALGORITHM:

Session State Initialization:

1. Check if the session state for donations exists. If not, initialize it as an empty list.

Define Functions:

1.add_donation(name, phone, area, city): This function appends a new donation record to the session state.

2. remove_donation(index): This function removes a donation from the session state based on the provided index.

User Interface Setup:

1. Display the title of the application.
2. Use a sidebar to allow users to select their type (Donor or Receiver).

Donor Functionality:

If the user selects "Donor":

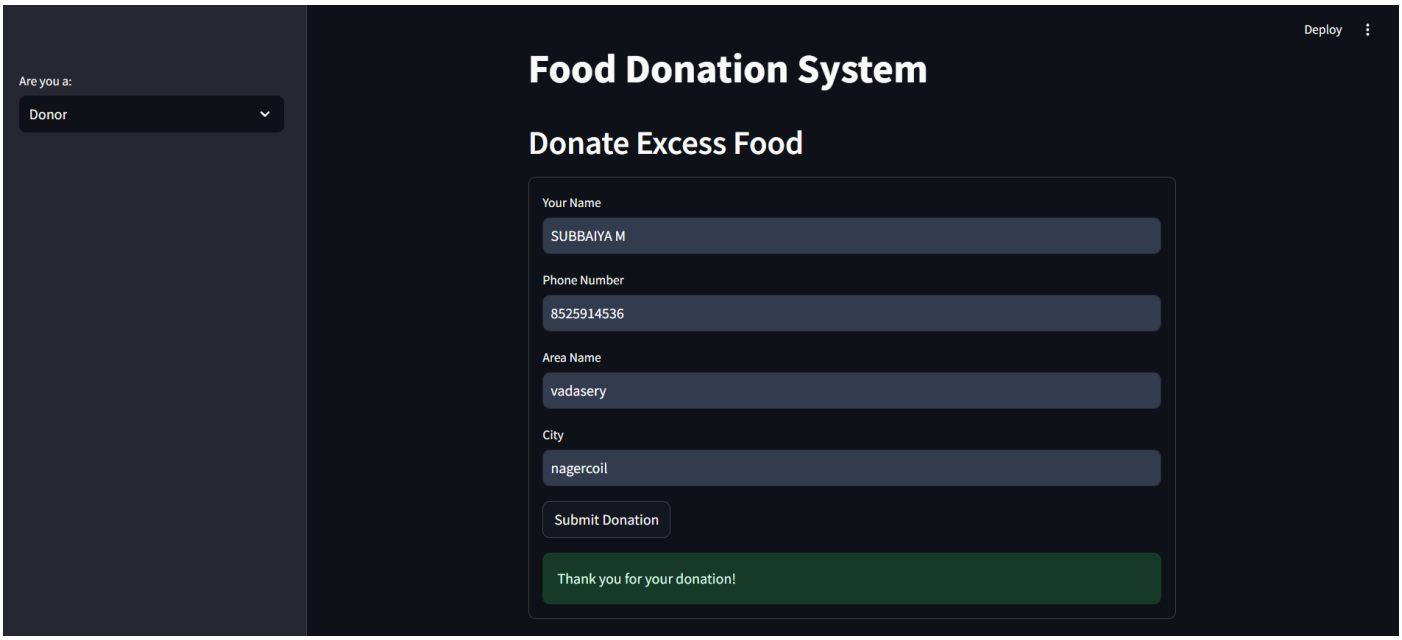
1. Present a form to collect the donor's name, phone number, area, and city.
2. On form submission, validate the inputs and add the donation to the session state if all fields are filled.
3. Provide feedback to the user (success or error message).

Receiver Functionality:

If the user selects "Receiver":

1. Present input fields to filter donations by area and city.
2. Filter the donations stored in the session state based on the provided area and city.
3. Display the filtered donations with donor details.
4. Provide a button to mark a donation as received, which will remove the donation from the list.


Are you a:
Donor
Deploy
Rerun R
Settings
Print
Record a screencast
About
Developer options
Clear cache C
Food Donation System
Donate Excess Food
Your Name
Phone Number
Area Name
City
Submit Donation

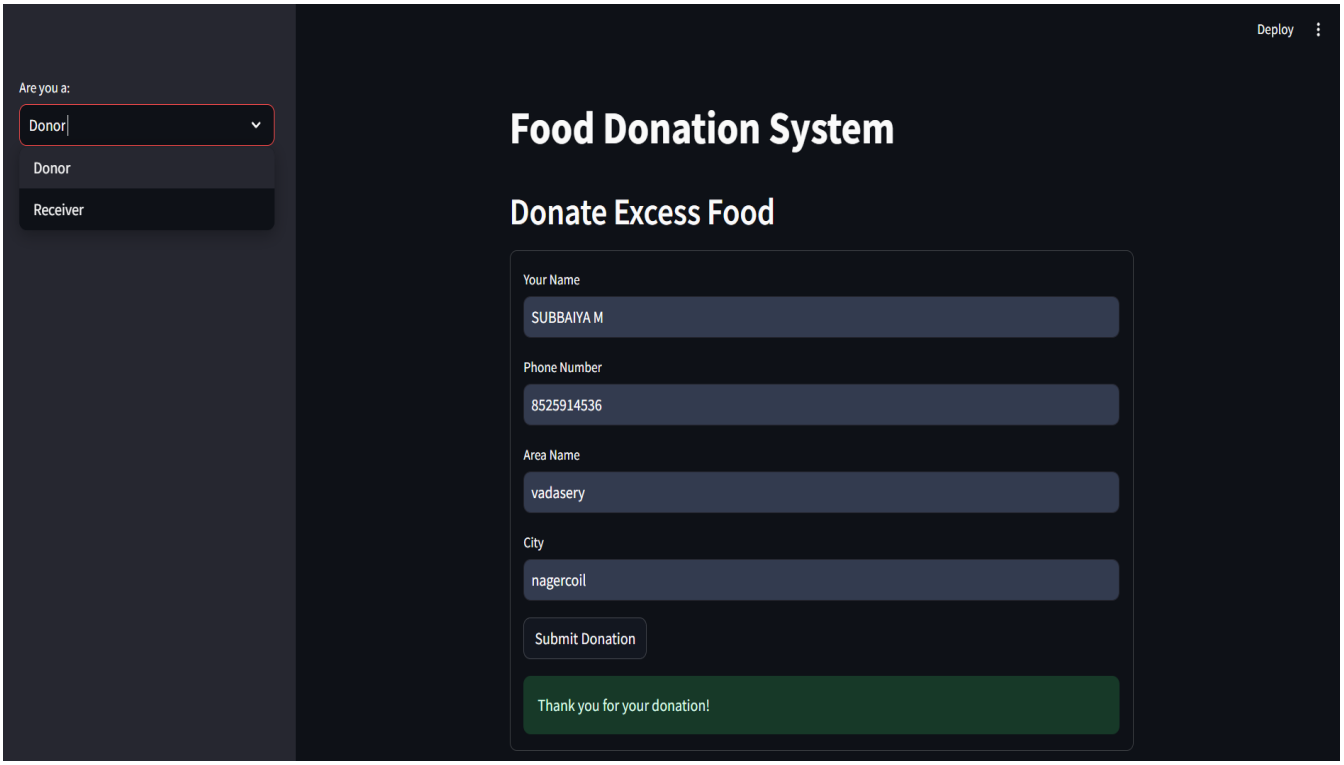

Are you a:
Donor
Deploy
Food Donation System
Donate Excess Food
Your Name
SUBBAIYA M
Phone Number
8525914536
Area Name
vadasery
City
nagercoil
Submit Donation
Thank you for your donation!

User Feedback:

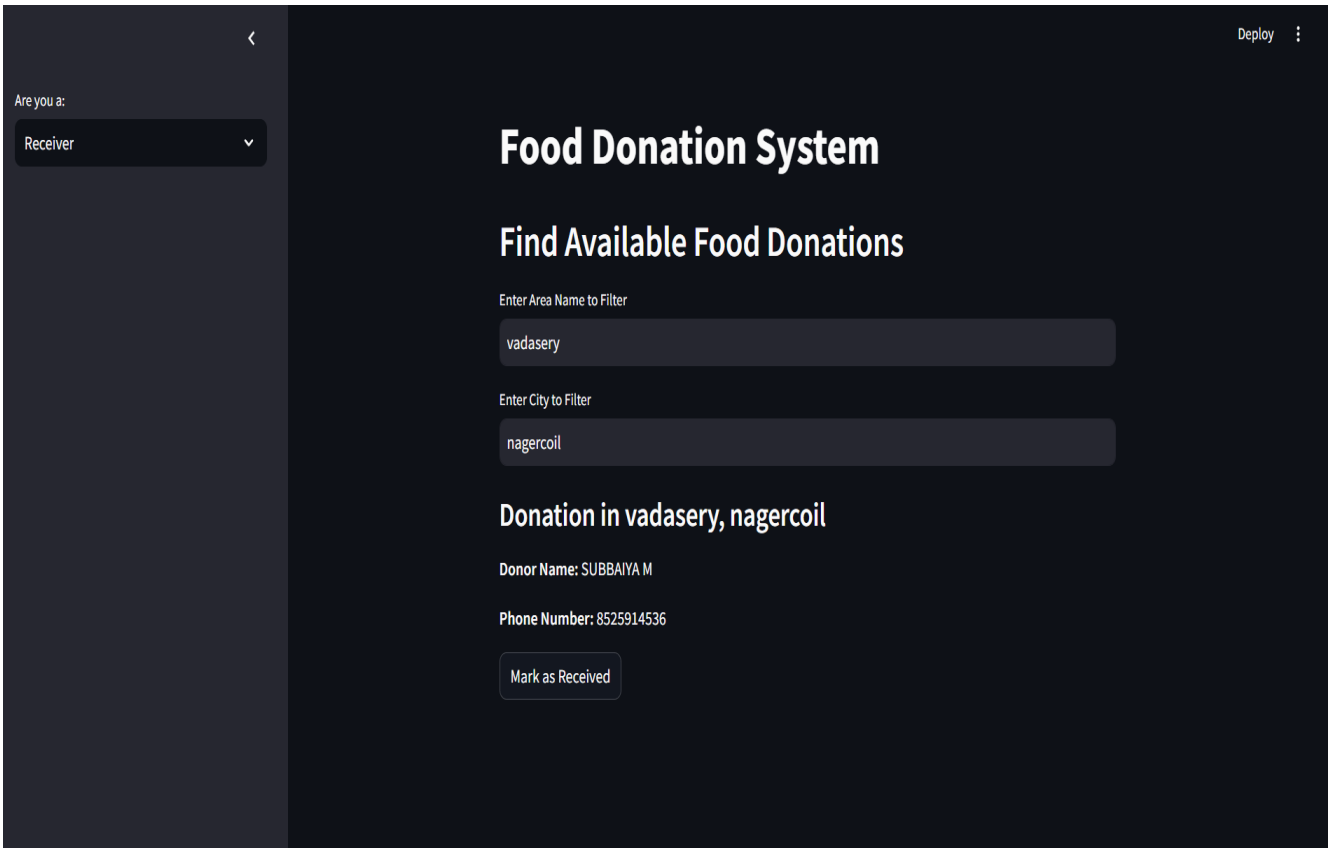
1. Provide success messages when donations are added or marked as received.
2. Display appropriate messages when no donations are available based on the filter criteria.

PROGRAM:

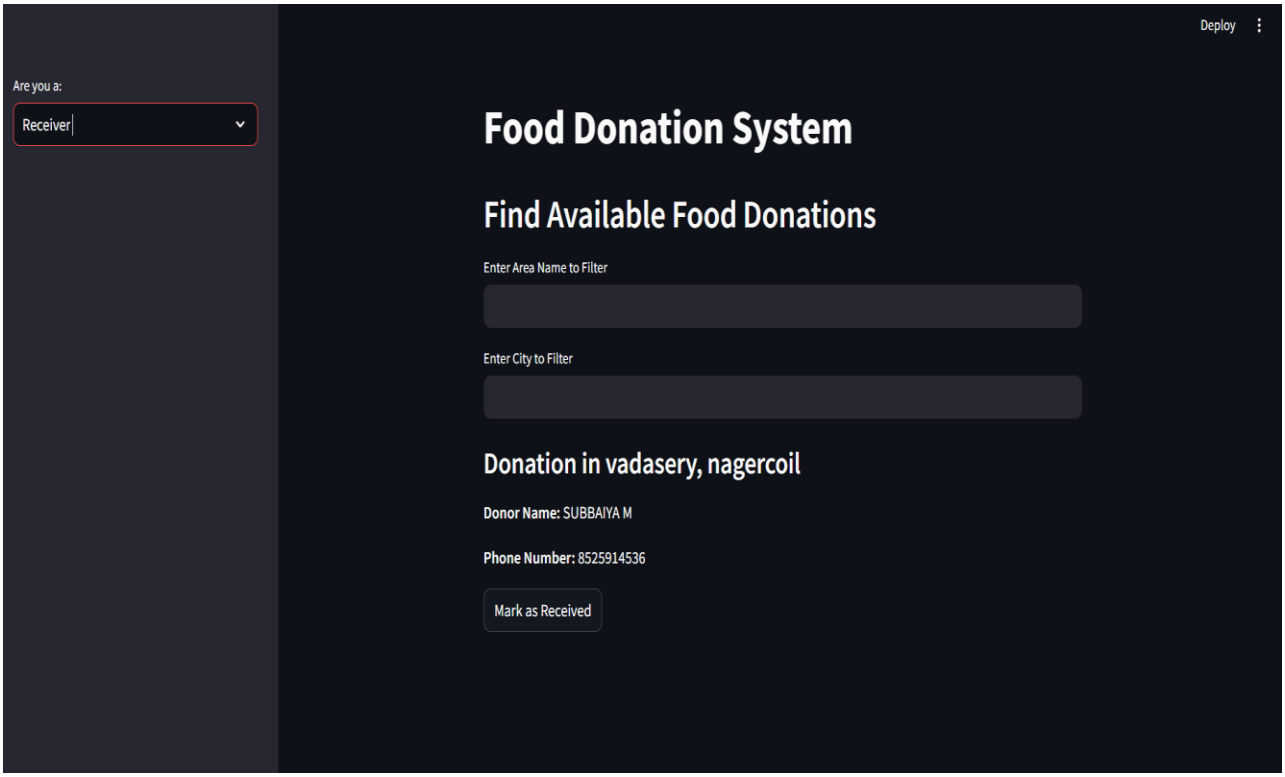
```
import streamlit as st

import pandas as pd


# Initialize data storage in session state

if 'donations' not in st.session_state:

    st.session_state['donations'] = []


# Function to add a new donation

def add_donation(name, phone, area, city):

    st.session_state['donations'].append({

        'name': name,

        'phone': phone,

        'area': area,

        'city': city

    })

# Function to remove donation once exchanged

def remove_donation(index):

    del st.session_state['donations'][index]


# Main app layout

st.title("Food Donation System")
```



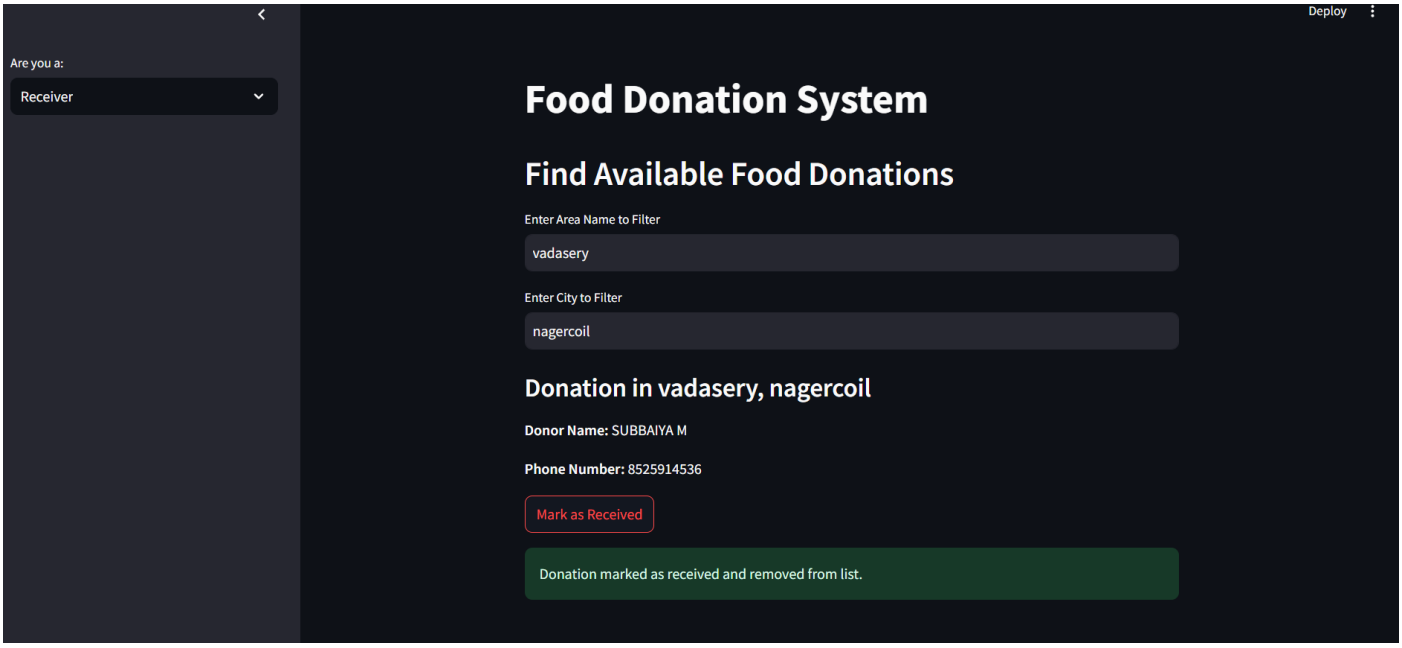
RECIEVER INTERFACE:



```
    # Select user type
    user_type = st.sidebar.selectbox("Are you a:", ["Donor", "Receiver"])
    if user_type == "Donor":
        st.header("Donate Excess Food")
        with st.form("donor_form"):
            name = st.text_input("Your Name")
            phone = st.text_input("Phone Number")
            area = st.text_input("Area Name")
            city = st.text_input("City")

            submit = st.form_submit_button("Submit Donation")
            if submit:
                if name and phone and area and city:
                    add_donation(name, phone, area, city)
                    st.success("Thank you for your donation!")
                else:
                    st.error("Please fill all fields.")
    elif user_type == "Receiver":
        st.header("Find Available Food Donations")
filter_area = st.text_input("Enter Area Name to Filter")
        filter_city = st.text_input("Enter City to Filter")

        # Filter donations based on area and city
        filtered_donations = [
            (index, donation) for index, donation in enumerate(st.session_state['donations'])
            if (not filter_area or donation['area'].lower() == filter_area.lower()) and
         (not filter_city or donation['city'].lower() == filter_city.lower())
        ]
```


Deploy
Are you a:
Receiver
Food Donation System
Find Available Food Donations
Enter Area Name to Filter
Enter City to Filter
Donation in vadasery, nagercoil
Donor Name: SUBBAIYA M
Phone Number: 8525914536
Mark as Received

```
if filtered_donations:
    for index, donation in filtered_donations:
        st.subheader(f"Donation in {donation['area']}, {donation['city']}")
        st.write("**Donor Name:**", donation['name'])
        st.write("**Phone Number:**", donation['phone'])
        if st.button(f"Mark as Received", key=f"remove_{index}"):
            remove_donation(index)
            st.success("Donation marked as received and removed from list.")
else:
    st.info("No donations available for the specified area and city.")
```


Deploy
Are you a:
Receiver
Food Donation System
Find Available Food Donations
Enter Area Name to Filter
vadasery
Enter City to Filter
nagercoil
Donation in vadasery, nagercoil
Donor Name: SUBBAIYA M
Phone Number: 8525914536
Mark as Received
Donation marked as received and removed from list.

CONCLUSION

The Food Donation System implemented in this code effectively connects donors with receivers, enabling the sharing of excess food. By utilizing Streamlit's session state, the application maintains a simple and user-friendly interface that allows users to easily donate or seek food donations. The filtering mechanism enhances usability, allowing receivers to find relevant donations based on their location. Overall, this application can contribute to reducing food waste and promoting community support through food sharing.