ENGINEERING

Community Food Rescue System: A Web-Based Platform for Surplus Food Redistribution and Community Welfare

Savita G. Gangani, Sneha, Sadhana, S. Ranjitha

Department of Computer Science and Engineering, City Engineering College (affiliated to Visvesvaraya Technological University), Bengaluru, Karnataka, India

Published November 2026

Abstract

Edible food is discarded by households, restaurants, hotels, caterers and event organisers, while many people nearby go without enough to eat. Part of this gap comes from poor coordination: surplus food is usually offered through phone calls and messaging groups, which are slow and leave no shared record. This paper presents the Community Food Rescue System, a Flask-based web application that connects food donors with registered NGOs and volunteers. Donors publish listings with the food name, description, quantity in kilograms, pickup address and an expiry or collection deadline. NGOs and volunteers browse the listings and submit requests, and the first valid request for a listing receives priority. The donor is informed of the request, pickup is coordinated, and the volunteer either delivers the food or asks an optional porter for a transport quotation, which the volunteer can accept or reject. The system uses HTML, CSS and JavaScript on the front end, Python with Flask on the back end, SQLite with SQLAlchemy for storage, Flask-Login for authentication, and Render for deployment. We review related work, describe the design and implementation, and discuss limitations. No field evaluation has been carried out yet, so the paper reports no measured food savings and lists the data that a proper evaluation would need.

Keywords: Flask; food redistribution; surplus food management; volunteer coordination; web application

1 Introduction

Food is lost or wasted at every stage between production and consumption. An FAO study estimated that roughly one third of the food produced for human consumption is lost or wasted worldwide [1], and the more recent UNEP Food Waste Index Report highlights that more than a billion meals are wasted every day, most of it in households, food services and retail, while many people remain hungry [2]. These are global figures, but the pattern is familiar in any Indian city: a wedding hall with trays of cooked food left at night, a restaurant with unsold meals, and an NGO a few kilometres away trying to arrange meals for people who have none.

In many such cases the food is wasted not because nobody wants it, but because the donor and the receiver cannot find each other in time. Cooked food stays safe for only a limited period, so a donor with surplus food late in the evening cannot spend an hour phoning around. Informal arrangements work when people already know each other, but they give no shared view of what is available, who has responded, or whether the food reached anyone.

This paper describes the Community Food Rescue System, a web platform built as our final-year project to address this coordination problem. It gives donors, NGOs, volunteers, porters and an administrator one place to list food, request it, arrange pickup and transport, and follow progress. The paper makes three contributions. First, it documents a complete, deployed workflow that includes an optional porter quotation step. Second, it explains a simple first-valid-request rule for assigning a donation and states clearly what the rule does and does not do. Third, it reports honestly what has and has not been evaluated.

The paper first reviews related work and states the problem and objectives. It then presents the proposed system and its architecture and database design, followed by the methodology, implementation and testing, and results. Limitations and future work are discussed before the conclusion.

2 Literature review

We searched for work on surplus food redistribution, digital donation platforms, volunteer coordination and food banks. The search was not a formal systematic review, and the studies below are those we could read and verify. They fall into three groups.

2.1 Organisational and social studies

Spring and Biddulph studied two UK surplus food redistribution initiatives through ethnography, comparing how autonomous, how fast-growing and how centrally governed each was [3]. They found that one initiative worked closely with corporate food actors and used standard protocols and strong central control, while the other emphasised autonomy and decentralised governance. Their work explains how such initiatives are organised, but it does not examine software design. Damavandi et al. studied barriers to adopting blockchain in food bank supply chains using expert judgments from one Italian food bank network and academics, and identified lack of access to technology and lack of organisational policy as leading barriers [6]. Their sample was small and drawn from a single network, but the result is a useful reminder that access to technology limits what food charities can adopt, which supports keeping a platform simple.

2.2 Matching and volunteer coordination models

Sanyal et al. proposed an app-based approach in which donors, receivers and volunteers who transport food are matched through a mechanism called FDRM-CA, which gives priority by time of acceptance and handles perishable food first [4]. The authors claim properties such as strategyproofness and evaluate the mechanism by simulation on 5,000 randomly generated requests. The evaluation uses synthetic data, and the proofs are described as informal in the paper. Lo et al. analysed nearly two years of data from Food Rescue US and built a matching model to study how recurring volunteer commitment affects growth and engagement [5]. They found that donors and unmatched volunteers were more likely to drop out after failed matches. This finding matters for our design because failed or unclear matches appear to cost a platform its users.

2.3 Web and mobile donation platforms

Several recent papers describe donation platforms similar to ours. Karanth et al. describe a MERN-stack platform with maps, status tracking and a spoilage-risk prediction model, but the paper reports no empirical evaluation [7]. Sandhiya et al. present a Node.js and Express web application with a map view, notifications, offline support and an AI assistant, and report no measured results [8]. Shivani

and Usha propose a centralised web platform with location tracking, notifications and an administrative module for donor verification, and report improvements from simulated scenarios rather than real deployment [9]. These papers share a common pattern: they describe features well, but the evidence for effectiveness is limited or simulated.

The reviewed studies and the present work are summarised in Table 1.

Table 1. Summary of reviewed studies and this work.

Ref.	Approach	Evidence reported	Limitation relevant here
[3]	Ethnography of two UK redistribution initiatives	Qualitative comparison	No software design
[4]	Matching mechanism with time-based priority	Simulation, synthetic data	No real deployment
[5]	Model and data analysis of volunteer platform	Food Rescue US data; model	Coordination only; not a build
[6]	Barriers to blockchain in food banks	Expert ratings, one network	Small sample
[7]	MERN web platform, maps, spoilage prediction	Design description	No evaluation reported
[8]	Node.js web platform, maps, offline support	Design description	No measured results
[9]	Web platform with location tracking	Simulated scenarios	Not deployed
This work	Flask app, first-valid-request rule, optional porter quotation	Working deployment; no field data yet	Needs pilot evaluation

2.4 Research gap

From the studies we could verify, the gap is practical rather than conceptual. Matching research tends to be evaluated on synthetic data, and platform papers tend to describe features without field evidence. We did not find, among these papers, a description of a simple platform that treats transport as an explicit and optional step, in which a porter quotes a price and the volunteer decides. Because our search was limited, we do not claim that no such system exists. Our contribution is a documented, deployed implementation of this workflow, with a clear statement of its assignment rule and of what remains to be evaluated.

3 Problem statement and research objectives

3.1 Problem statement

Surplus edible food and food insecurity exist side by side, and the link between them is weak. Donors often do not know which NGOs or volunteers are nearby and active, and NGOs cannot see what is available. Coordination through calls and messages is slow, offers no common record, and can produce duplicate responses, so a donor may be contacted repeatedly or two parties may arrive for the same food. Time is critical, because delays reduce the chance that food is still suitable to eat. Volunteers may also lack transport, with no easy way to arrange it. The research problem is therefore how to build a simple, accessible web system that makes available food visible, assigns each donation to one requester without confusion, and supports pickup, optional transport and progress tracking.

3.2 Objectives

- Develop a centralised web-based food donation platform and connect donors with NGOs and volunteers.
- Let donors publish surplus food information, and let NGOs and volunteers view and request available food.
- Implement a first-valid-request priority mechanism and notify donors of submitted requests.
- Support volunteer-based collection and distribution, with optional porter transport assistance.
- Enable donation status monitoring and delivery tracking, and improve coordination between participants.
- Support the redistribution of surplus food to people in need and contribute to responsible consumption.

4 Proposed system

4.1 Users

The system has five roles: donor, NGO, volunteer, porter and administrator. Donors are households, restaurants, hotels, caterers and event organisers who register, publish surplus food details with quantity, pickup location and expiry information, receive request notifications and follow progress. NGOs register, view available donations with quantity and collection details, submit requests and coordinate distribution. Volunteers view and request donations, coordinate collection, deliver food directly to people in need and update delivery progress. Porters provide optional transport: when a volunteer cannot manage transport alone, the volunteer requests a porter, who gives a quotation that the volunteer accepts or rejects. Porter involvement is optional and not needed for every donation. The administrator manages the platform and its users through the available administrative functions.

4.2 Workflow

Figure 1 shows the workflow. A donor registers, logs in and creates a listing containing the food name, description, quantity in kilograms, expiry time or collection deadline and pickup address. Registered NGOs and volunteers can view the listing and submit requests. The first valid request receives priority for that donation. The donor receives information about the request, and the assigned user coordinates collection. The volunteer either manages transport independently or requests a porter; the porter quotes a price and the volunteer accepts or rejects it. The food is then delivered directly to people in need, and progress can be followed through the status information shown by the system.

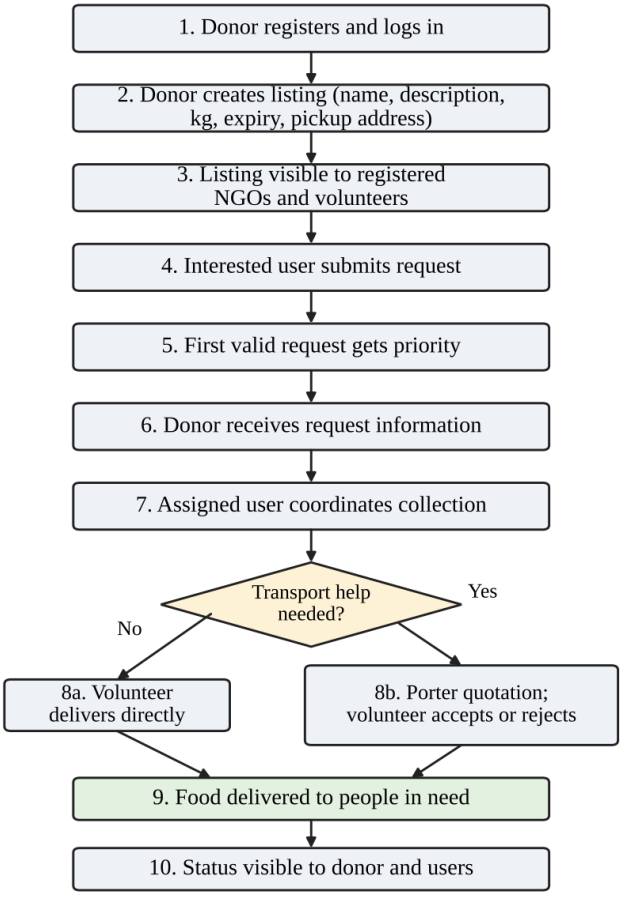


Figure 1. Complete system workflow.

4.3 First-valid-request priority

When several users want the same donation, the system must decide who receives it. The rule is simple: the first request that is valid for a listing receives priority, and the listing is then no longer open to independent claims by others. Figure 2 shows the logic. The rule is deterministic and based on order of submission and validity. It is not machine learning, and it does not rank requesters by need, distance or past behaviour. We chose it because it is easy to explain and every requester faces the same condition. Its weakness is that a distant or poorly equipped requester may hold a donation ahead of a better-suited one.

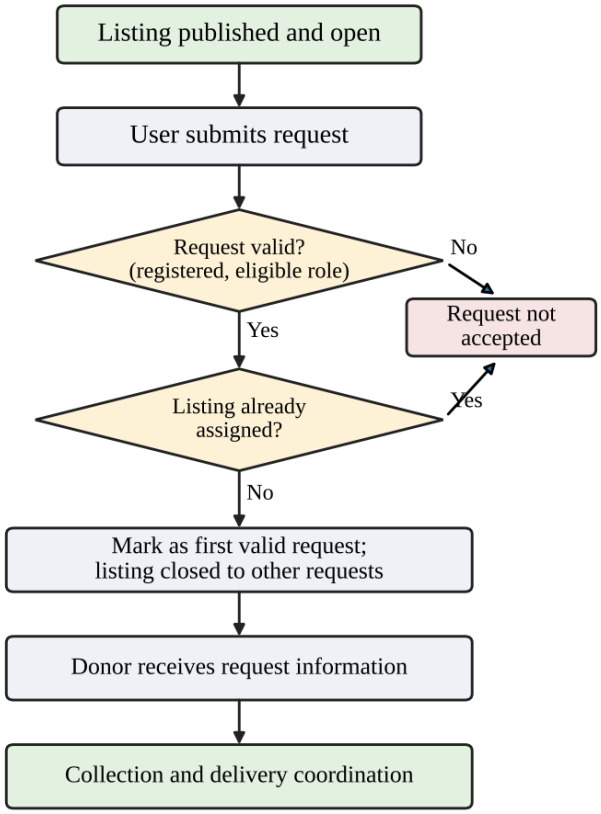


Figure 2. Food donation request processing flowchart.

4.4 Quantity management

The system records quantity in kilograms. For an NGO, this shows whether a donation fits the number of people it expects to serve and whether it has enough containers and transport. For a volunteer, it indicates whether the load can be carried alone. As an illustration only, if 50 kg is donated and an approximate serving of 500 g is assumed, the donation could provide around 100 servings. The actual number of beneficiaries depends on the food type, serving size and distribution conditions. The system stores and displays quantity; it does not automatically calculate beneficiaries.

4.5 Comparison with informal donation

The traditional approach and the proposed system are compared in Table 2.

Table 2. Traditional approach versus the proposed system.

Aspect	Traditional approach	Proposed system
Communication	Manual calls and messages	Centralised digital coordination
Food information	Informal, scattered	Structured listings
Requests	Made by phone or message	Submitted online
Allocation	Unstructured coordination	First-valid-request priority
Visibility	Limited	Donation progress tracking
Transport	Arranged manually	Optional porter quotation

5 System architecture and design

5.1 Architecture

The system is a single Flask-based web application [11] organised in layers, not a microservices architecture. Figure 3 shows the layers, and Figure 4 shows the use cases of the five roles. The User Interface Layer consists of HTML, CSS and JavaScript pages used by all five roles; it collects input and displays listings, requests and status. The browser sends requests to Flask, and Flask-CORS controls which origins may call the endpoints. The Authentication and Authorization Layer uses Flask-Login [13] to identify the logged-in user and applies role-based access, so that, for example, only donors can create listings. The Application Logic Layer is the Flask application, which receives requests, applies the rules of the system and returns responses. Five functional modules sit within it: food donation management, food request management, notification, volunteer and porter coordination, and delivery tracking. The Database Layer uses SQLAlchemy [12] to map Python classes to tables in a SQLite file, community_food_rescue.db, and is the only layer that reads or writes stored data.

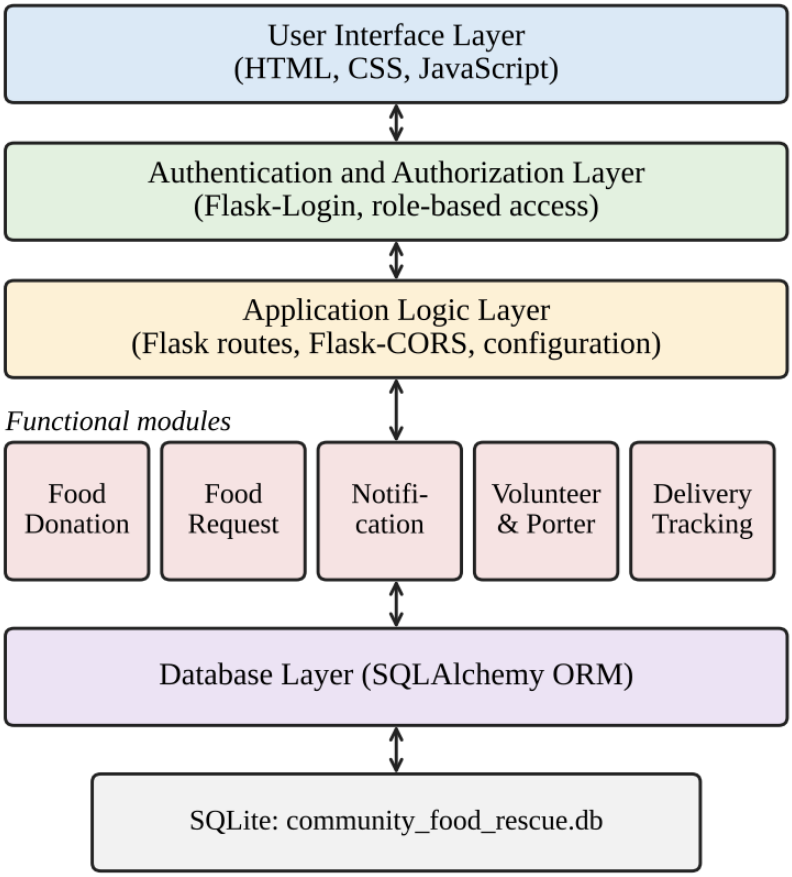


Figure 3. Overall system architecture.

The modules cooperate through shared records. A listing created by the donation module is read by the request module when a request arrives. A valid request produces information for the donor through the notification module. If transport help is needed, the porter module links a quotation to that request, and the tracking module reads the combined state to show progress.

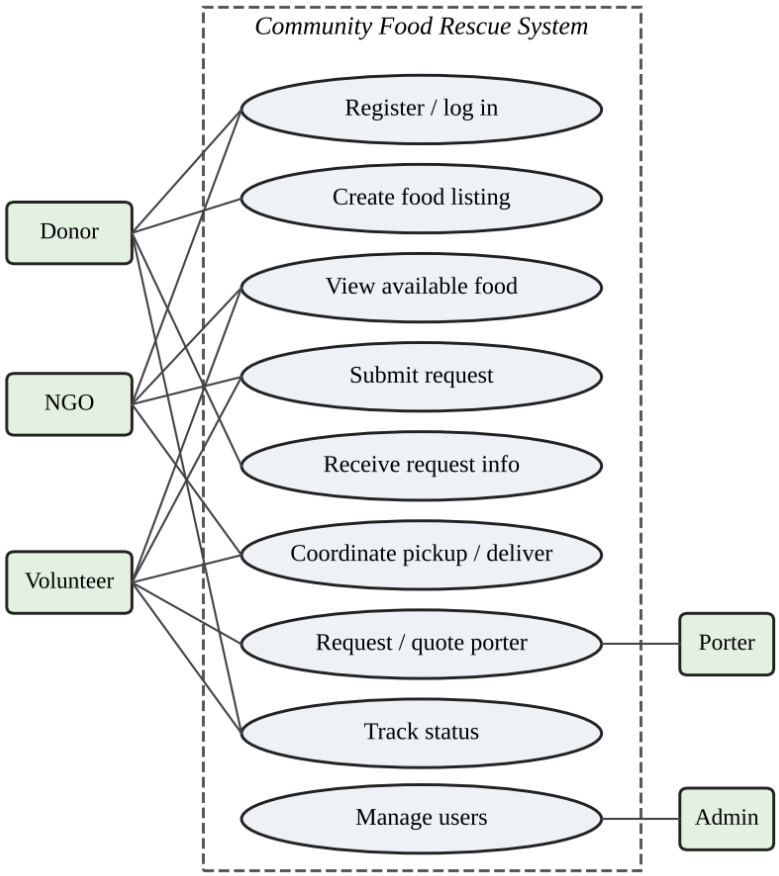


Figure 4. Use-case diagram.

5.2 Database design

The logical entities are users with roles, food donations, food requests, donation status, volunteer assignments, and porter requests with quotations. Figure 5 shows a proposed logical design in which a user may own many donations, a donation may receive requests, a request may have a porter request and quotation, and status updates record progress. This is a proposed logical design, not a verified description of the implemented schema.

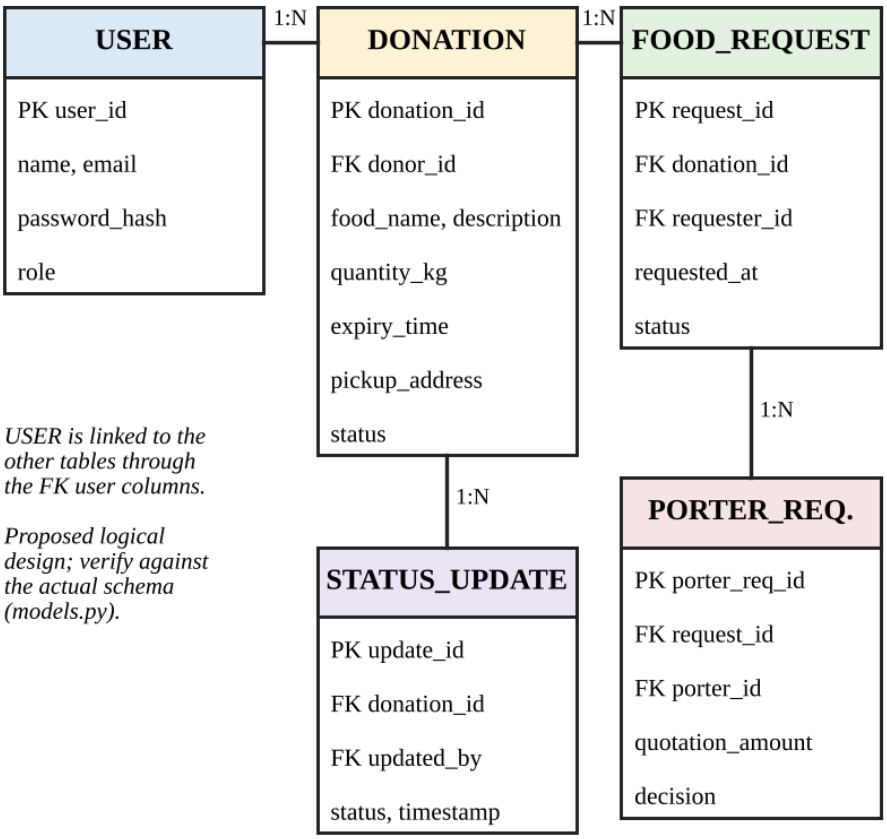


Figure 5. Proposed logical entity relationship diagram.

6 Research methodology

The work followed the stages below, in the order in which we carried them out or planned them.

- **Problem identification:** observing how surplus food is shared informally and identifying delays, duplicate responses and transport gaps.
- **Literature review:** reading work on redistribution, matching and donation platforms.
- **Requirement analysis:** defining the five roles, the listing fields and the workflow.
- **Architecture and database design:** choosing a layered Flask application and a logical data model.
- **Frontend development:** building HTML, CSS and JavaScript pages for each role.
- **Backend development:** implementing routes, role checks and request rules in Flask with SQLAlchemy.
- **Module integration:** connecting donation, request, notification, porter and tracking modules.
- **Functional and integration testing:** checking each role's workflow end to end.
- **Cloud deployment:** hosting the application on Render.
- **Evaluation:** planned as a pilot with real donors, NGOs and volunteers; not yet done.

7 Implementation and testing

7.1 Implementation

The backend is written in Python with Flask. Users are stored with a role (donor, NGO, volunteer, porter or admin), and Flask-Login handles registration, login and sessions, with role checks before sensitive actions. Donors create listings through a form; listings appear to registered NGOs and volunteers. Request submission applies the first-valid-request rule. The donor receives information about requests, volunteers can ask for a porter, the porter provides a quotation, and the volunteer accepts or rejects it. Status information is updated as the donation progresses. The configuration is kept in a Python configuration module, development was done in Visual Studio Code, and the application is deployed on Render [14] at https://community-food-rescue.onrender.com.

7.2 Testing

We do not report test results because no test records have yet been compiled. Table 3 lists the scenarios that should be tested and recorded, covering functional, integration, database and usability aspects.

Table 3. Test scenarios to be recorded.

Type	Scenario	Result
Functional	Register and log in as each of the five roles	Not yet recorded
Functional	Create a listing with valid and invalid inputs	Not yet recorded
Integration	Two users request the same listing; check priority	Not yet recorded
Integration	Porter quotation accepted and rejected	Not yet recorded
Access	A role opens a page meant for another role	Not yet recorded
Database	Records persist after restart of the application	Not yet recorded
Usability	A new user completes a donation and a request	Not yet recorded

8 Results and discussion

8.1 Implemented capabilities

The implemented platform supports the full workflow described above: registration and login for five roles, structured donation listings, request submission with first-valid-request priority, donor information about requests, optional porter quotation, delivery to people in need and status monitoring. These are descriptions of what the system is designed and built to do. We have no measured results such as kilograms of food rescued, response times, user counts or beneficiary numbers, and we do not estimate any.

8.2 Observations and evidence still needed

Observations from testing, expected social benefits and future possibilities should be kept separate. At present only the first of these, the implemented capabilities, is supported. A publishable evaluation would need data from a real pilot, such as the number of listings posted, the share of listings that were requested, the time from posting to request and to pickup, the share of food collected before

expiry, the number of porter requests and how many quotations were accepted, and feedback from users. Such data will be reported once the pilot has been carried out.

8.3 Social contribution and sustainable development goals

By making available food visible to organisations that can use it, the platform aims to reduce edible food waste and improve redistribution, to involve volunteers and NGOs in the community, and to support economically disadvantaged people. Its relation to the Sustainable Development Goals [10] is as follows. For SDG 2 (Zero Hunger), it aims to help available food reach people who need it. For SDG 12 (Responsible Consumption and Production), it aims to reduce the waste of edible food. For SDG 17 (Partnerships for the Goals), it brings together donors, NGOs, volunteers and porters. These are intended contributions; we have no evidence of measured impact.

9 Limitations and future enhancements

9.1 Limitations

Several limits apply. The system needs internet access. It cannot verify food quality and relies on donors to describe food honestly and set sensible deadlines. Transport depends on volunteers and porters being available. Collection is time-sensitive, and delays can still make food unusable. The first-valid-request rule is simple and ignores need and distance. SQLite suits a prototype but not many concurrent users, and deployment so far is geographically limited. Most importantly, the system has not had a larger real-world evaluation.

9.2 Future enhancements

The following are possible future developments and are not part of the current system: location-based donor and NGO matching, route optimisation, a mobile application, real-time notifications, improved expiry alerts, advanced analytics, multilingual support, automated beneficiary estimation, integration with more NGOs and community organisations, and AI-based prediction of surplus food and demand. Matching studies such as [4] could guide a fairer allocation rule than first-come priority.

10 Conclusion

This paper addressed the coordination gap between people who have surplus food and people who can use it. We built a layered Flask web application with five user roles, structured listings, a first-valid-request rule, an optional porter quotation step and status tracking, and deployed it on Render. Our review suggests that existing platform papers describe features but report little field evidence, and our own work shares that limit: we present a working design, not a measured result. The practical value of the system now depends on a pilot with real donors, NGOs and volunteers, careful verification of the database and request rules, and an honest report of what the data show.

Acknowledgment

The authors thank City Engineering College, the Department of Computer Science and Engineering, their project guide and the faculty members for their guidance and support.

References

[1] Gustavsson J, Cederberg C, Sonesson U, van Otterdijk R, Meybeck A. *Global food losses and food waste: extent, causes and prevention*. Rome: Food and Agriculture Organization of the United Nations; 2011.

[2] United Nations Environment Programme. *Food Waste Index Report 2024*. Think Eat Save: tracking progress to halve global food waste. Nairobi: UNEP; 2024.

[3] Spring CA, Biddulph R. Capturing waste or capturing innovation? Comparing self-organising potentials of surplus food redistribution initiatives to prevent food waste. *Sustainability*. 2020;12(10):4252. https://doi.org/10.3390/su12104252

[4] Sanyal S, Singh VK, Xhafa F, Sanyal B, Mukhopadhyay S. A game theoretic framework for surplus food distribution in smart cities and beyond. *arXiv* [Preprint]. 2021. arXiv:2102.04929.

[5] Lo I, Manshadi V, Rodilitz S, Shameli A. Commitment on volunteer crowdsourcing platforms: implications for growth and engagement. *arXiv* [Preprint]. 2020. arXiv:2005.10731.

[6] Damavandi S, Berardi L, Abbasi S. Blockchain technology adoption in food bank supply chains: a rough DEMATEL-based approach. *arXiv* [Preprint]. 2025. arXiv:2503.05811.

[7] Karanth A, Hemareddy AK, Chethana GT, Gowtham N, Magadum HV. A web-based food donation and redistribution platform for efficient surplus food management. *International Journal of Computer Sciences and Engineering*. 2025;13(12):27–32. https://doi.org/10.26438/ijcse/v13i12.2732

[8] Sandhiya C, Vishal E, Prakash A, Vikashini G, Kiruthika S. Smart food waste reduction system using a web-based donation platform. *International Journal of Engineering Research and Technology*. 2026;15(3). https://doi.org/10.5281/zenodo.19221250

[9] Shivani SS, Usha M. *A web-based food donation and redistribution system to minimize food waste and support NGOs*. International Journal of Advanced Research in Computer and Communication Engineering. 2026;15(1). https://doi.org/10.17148/IJARCCE.2026.151148

[10] United Nations. *Transforming our world: the 2030 Agenda for Sustainable Development*. General Assembly Resolution A/RES/70/1. New York: United Nations; 2015.

[11] Pallets Projects. *Flask documentation* [Internet]. Available from: https://flask.palletsprojects.com

[12] SQLAlchemy. *SQLAlchemy documentation* [Internet]. Available from: https://docs.sqlalchemy.org

[13] Flask-Login. *Flask-Login documentation* [Internet]. Available from: https://flask-login.readthedocs.io

[14] Render. *Render documentation* [Internet]. Available from: https://render.com/docs