

An AI- and IoT-Enabled Smart Food Donation Management System with Explainable Food-Risk Prediction, Safe-Time Forecasting and Automated NGO Assignment

S. Mathavencie Sudhakar*, Dr. Manoj Singh**

*(Department of Computer Science, SIES College of Arts, Science and Commerce, Mumbai, India

Email: vencie2003@gmail.com)

** (Department of Computer Science, SIES College of Arts, Science and Commerce, Mumbai, India)

Abstract:

Surplus cooked food from restaurants, hotels and cafés is often discarded because donors cannot verify its safety or reach an NGO in time. This paper presents a Smart Food Donation Management System that combines low-cost IoT sensing, explainable machine learning and automated NGO allocation in one web platform. An ESP32 node with DHT22 and MQ-135 sensors streams storage conditions, and high gas readings flag or block spoiled food. Six algorithms were compared with 5-fold cross-validation; the selected Gradient Boosting model classifies each donation as low, medium or high risk from temperature, humidity, food age and food type. A safe-time forecast estimates when food will become high risk, and exact Shapley values explain each prediction. An allocation module offers each donation to one NGO at a time, filtering by remaining capacity and by whether the NGO can arrive before the food becomes unsafe. On 600 held-out records the model achieved 97.0% accuracy, a macro F1-score of 0.957 and 98.7% high-risk recall, with no high-risk record classified as low risk. In a 50-day simulation the allocation method delivered 87.2% of meals safely, against 83.3% for nearest-NGO allocation, and all 45 functional tests passed.

Keywords — ESP32, explainable AI, Flask, food donation, food safety, food waste, gradient boosting, Internet of Things, NGO matching, Shapley values, shelf-life prediction.

I. INTRODUCTION

India wastes about 78.1 million tonnes of food a year, second only to China, even as many people remain food-insecure [1], [2]. Globally, 1.05 billion tonnes of food were wasted in 2022, roughly 19% of the food available to consumers, and food services accounted for 28% of that waste [1].

Restaurants, hotels, caterers and cafés routinely hold surplus cooked food at the end of service. Two practical barriers limit its donation. First, donors and NGOs have no quick, objective way to judge whether cooked food is still safe, so food is either discarded out of caution or redistributed with unknown risk. Second, coordination is manual: phone calls and messaging groups are slow, and perishable food can pass its safe-holding window

while waiting for an NGO that may not have the capacity to accept it.

Low-cost IoT prototypes built on ESP32 or Arduino boards with DHT22 and MQ-series gas sensors can detect spoilage trends [8]–[10], and web platforms can match donors with NGOs [5]–[7]. However, few systems connect sensed evidence, an explainable risk score, a remaining-time estimate and capacity-aware assignment in one platform that a small donor can afford. The objectives of this work are to:

- 1) build a low-cost IoT unit that measures storage temperature, humidity and spoilage-related gases at the point of donation;
- 2) predict the safety risk of each cooked-food donation and estimate its remaining safe time;
- 3) explain each prediction in terms donors and NGOs can understand;

- 4) assign pickups automatically to suitable NGOs based on location, capacity and urgency; and
- 5) provide separate, role-specific portals for donors, NGOs and an administrator.

The main contributions are: (i) an integrated architecture linking an ESP32 sensor node, a machine-learning risk classifier and a Flask–SQLite donation platform, with a cross-validated comparison of six algorithms; (ii) a five-class food-type taxonomy for Indian cooked food that encodes relative perishability; (iii) a safe-time forecast and SHAP-based explanations shown with every prediction; (iv) a safety-aware NGO allocation method that offers donations sequentially and skips NGOs that cannot reach the food before it becomes unsafe; and (v) an evaluation covering model comparison, feature ablation, robustness to sensor error, data requirements and a simulation study of NGO allocation.

Section II reviews related work. Section III presents the architecture, Section IV the methodology and Section V the implementation. Section VI describes the experimental setup and results, Section VII discusses limitations, and Section VIII concludes.

II. RELATED WORK

Prior systems either sense spoilage or coordinate donations, but rarely combine both with explainable decisions.

Donation and redistribution platforms: AI-FEED prototypes an AI-powered platform for the food charity ecosystem, in which a donor module alerts nearby charities to surplus food [5], and recent work examines how digital solutions can address food donation challenges [6]. Game-theoretic matching schemes have been proposed that prioritise perishable items when pairing donors with receivers [7]. Field studies stress that food-safety standards are essential for redistribution to scale [11].

IoT spoilage monitoring: ESP32 and Arduino prototypes using DHT22 with MQ-135, MQ-137 or MQ-4 gas sensors can track freshness in storage [8]–[10]. One framework combined CO₂, temperature, humidity and methane sensing with Random Forest classification to separate fresh from spoiled food [8], and ammonia readings rise far more slowly under refrigeration, confirming the value of temperature

data [9]. These studies target storage monitoring rather than a donation workflow.

Explainability: SHAP assigns each input feature a contribution to an individual prediction, based on Shapley values from cooperative game theory [4]. It is widely used for tree models but rarely applied to food-donation decisions, where trust in the risk score is essential. Table I compares this work with related systems.

TABLE I
COMPARISON WITH RELATED SYSTEMS

Work	Sensing	Risk model	Explanation	Safe-time	NGO assignment
AI-FEED charity platform [5]	No	No	No	No	Partial
Game-theoretic matching [7]	No	No	No	No	Yes
Multi-sensor spoilage [8]	Yes	Yes	No	No	No
Ammonia-based detector [9]	Yes	Threshold	No	No	No
This work	Yes	Yes	SHAP	Yes	Yes

III. SYSTEM ARCHITECTURE

The system has four layers: an IoT sensing layer, a Flask application layer hosting the AI engine, an allocation layer, and role-based portals (Fig. 1). Sensor readings and donor-entered details reach the same application, the AI engine attaches a risk level, safe time and explanation, and the allocation module offers the donation to a suitable NGO. A housekeeping routine runs at most every 30 seconds while the site is in use: it re-scores waiting donations with their current age, expires unanswered offers and makes new ones.

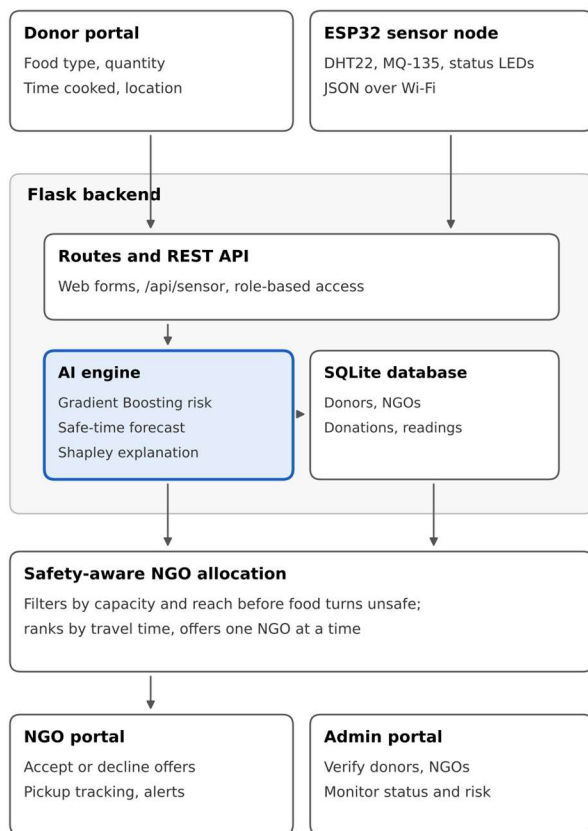


Fig. 1. System architecture and data flow.

The donation lifecycle proceeds as follows:

- 1) the donor posts the food with its type, quantity, cooking time and storage readings, and can attach a sensor node to the donation;
- 2) the ESP32 posts temperature, humidity and gas readings to the sensor API, which re-scores the donation with its current age;
- 3) the AI engine predicts the risk, forecasts the safe time and computes the Shapley explanation shown to the donor;
- 4) low- and medium-risk donations are offered to the best eligible NGO, high-risk donations are held for a manual check by the administrator, and spoiled food is blocked; and
- 5) the NGO accepts or declines the offer and then moves the pickup from Assigned to Scheduled, Picked Up and Delivered, with notifications at each step.

IV. METHODOLOGY

A. IoT Sensing Unit

The sensing node uses an ESP32-WROOM-32 DevKit (30-pin) with a DHT22 temperature–

humidity sensor and an MQ-135 gas sensor (Table II). A green LED indicates safe storage, and a red LED indicates an unsafe reading. The ESP32 sends readings over Wi-Fi to a Flask REST endpoint as JSON, authenticated with a device API key; the administrator attaches the device to a donation on the website, so every reading updates that donation. A gas reading above an alert level forces a high-risk label regardless of the model, and a reading above a spoilage level blocks the donation and cancels any pickup. The MQ-135 is powered from 5 V and its analog output is scaled by a resistive divider before the 12-bit analog-to-digital converter of the ESP32 (0–4095); each transmitted value is the mean of 10 samples. The alert and spoilage levels are set at 1.5 and 2.0 times the clean-air level measured after warm-up (Section VI-H).

TABLE II
SENSING UNIT COMPONENTS

Component	Measures	Range / accuracy	Role
ESP32-WROOM-32	—	Dual-core, 2.4 GHz Wi-Fi	Acquisition, upload
DHT22 (AM2302)	Temperature, humidity	−40–80 °C ±0.5 °C; 0–100% RH ±2–5%	Storage condition
MQ-135	NH ₃ , VOCs, CO ₂ -equivalent	Analog; 24–48 h burn-in, calibration	Spoilage-gas index
Green and red LEDs	—	—	Local status indicator

B. Input Features and Food-Type Taxonomy

The model uses temperature (°C), relative humidity (%), food age (hours since cooking) and food type (Table III). Food type is an ordinal category reflecting perishability; mixed dishes take the category of their riskiest ingredient. The MQ-135 gas index is logged with every reading and will be added as a fifth feature once enough sensor data is collected.

TABLE III
FOOD-TYPE CATEGORIES

Code	Category	Examples
1	Breads and dry items	Roti, chapati, dry snacks
2	Cooked rice and grains	Plain rice, pulao, upma
3	Vegetable curries and dals	Sabzi, dal, sambar
4	Dairy-based	Paneer dishes, kheer, curd items

Code	Category	Examples
5	Meat, fish and egg	Chicken curry, fish fry, egg curry

C. Dataset and Labelling

The dataset contains 3,000 records generated from food-safety rules, covering temperatures of 0–76 °C, relative humidity of 25–100% and food ages of 0–24 h across all five food types. Its class distribution is 1,337 low, 486 medium and 1,177 high risk. Labels follow guidance that cooked food held between 5 °C and 60 °C should not stay out longer than about 2 hours and should be discarded after 4 hours [12], [13]. Food kept hot (above 60 °C) or chilled (below 5 °C), and recently cooked dry items, are labelled low risk; food approaching the safe limit, or perishable food (codes 3–5) held at high humidity, is medium risk; and food past the limit is high risk. The gas reading is not a model feature; it is applied as a separate rule on top of the model output. The data is split 80:20 with stratification by class into 2,400 training and 600 test records; the test set is used only once, for the final evaluation.

D. Risk Classification and Model Selection

Six algorithms were compared: Logistic Regression, k-Nearest Neighbours, Support Vector Machine, Decision Tree, Random Forest [3] and Gradient Boosting [16], all implemented in scikit-learn [14]. Class weights were balanced where supported, because the medium class is smallest. Each model was scored by 5-fold stratified cross-validation on the training set, and the model with the highest mean macro F1-score was selected. Gradient Boosting was selected; it builds an additive ensemble of M shallow regression trees, each fitted to the errors of the ensemble so far:

$$F_m(x) = F_{m-1}(x) + v h_m(x), \quad m = 1, \dots, M \quad (1)$$

where h_m is the tree added at stage m and v is the learning rate. After selection, the model is evaluated once on the test set and then retrained on all 3,000 records for deployment in the web application.

E. Safe-Time Forecast

The safe-time forecast estimates how long a donation can wait before it becomes high risk. Assuming the storage conditions stay the same, the model is queried for the same food at increasing ages

in 15-minute steps up to 24 hours, and the first step predicted as high risk is returned:

$$T_{\text{safe}} = \min \{ \tau : f(\theta, h, a + \tau, c) = \text{High} \} \quad (2)$$

where τ runs from 0 to 24 h in 15-minute steps, θ is temperature, h humidity, a the current food age in hours and c the food type. If no step is high risk, the food is reported as safe for more than 24 hours. Because the forecast uses the trained model itself, it stays consistent with the risk label shown to the user.

F. Explainable Predictions with Shapley Values

Each prediction is explained with exact Shapley values [4]. The value of a set of features S is the model's expected risk score, a 0–100 score computed as $0 \cdot P(\text{low}) + 50 \cdot P(\text{medium}) + 100 \cdot P(\text{high})$, averaged over 50 background records sampled from the training data, with the features in S set to the donation's values. The contribution of feature i is

$$\phi_i = \sum_{S \subseteq N \setminus \{i\}} \frac{[|S|!(n-|S|-1)!/n!]}{[|S|!(n-|S|-1)!/n!]} (v(S \cup \{i\}) - v(S)) \quad (3)$$

where N is the set of $n = 4$ features. With only four features, all 16 subsets are evaluated exactly, so the method works with any classifier, including multi-class Gradient Boosting. The donation page shows each feature's contribution in risk points, for example how much the food's age pushed the score above that of a typical donation.

G. Safety-Aware NGO Allocation

Each NGO profile stores its Mumbai area, service area and daily capacity in meals. For a waiting donation, an NGO is eligible only if it is approved, has not already been offered this donation, has enough capacity left today after subtracting meals already offered to it, and can reach the donor before the food becomes high risk, i.e. its estimated travel time is shorter than T_{safe} . Eligible NGOs are ranked by travel time and then by their number of active pickups.

The donation is offered to the top-ranked NGO only. The offer expires after 15 minutes if the food becomes high risk within 3 hours, and after 60 minutes otherwise; an expired or declined offer passes to the next NGO. High-risk donations are not offered automatically but held for a manual check by the administrator, and if no NGO is eligible the reason is recorded and the administrator is alerted.

V. IMPLEMENTATION

The prototype is a Python Flask web application backed by SQLite, with the trained model and the Shapley background sample loaded at start-up and ESP32 firmware written in Arduino C++. Tables IV–VI summarise the technology stack, the functional modules and the database schema.

TABLE IV
TECHNOLOGY STACK

Layer	Technology
Sensor firmware	Arduino C++ on ESP32; DHT library; WiFi and HTTPClient
Backend	Python, Flask, Jinja2 templates, REST API
Database	SQLite
Machine learning	scikit-learn, pandas, NumPy, joblib
Front end	HTML, CSS, JavaScript

TABLE V
FUNCTIONAL MODULES BY PORTAL

Portal	Module	Function
Donor	Donation posting	Food name, type, quantity, cooking time and storage readings
Donor	Risk result	Risk level, confidence, safe-time forecast and Shapley explanation
Donor	History	Donations with status and assigned NGO
NGO	Profile	Area, service area, contact details, daily meal capacity
NGO	Offers	Accept or decline offers before they expire
NGO	Pickups	Move pickups from Assigned to Delivered
Admin	Dashboard	Donations by status and risk; alerts and notifications
Admin	Management	Approve and edit donors and NGOs; attach sensors; assign or unblock donations
Admin	Reports	Period reports and data export
System	Sensor API	Key-protected readings; live re-scoring; gas-based blocking
System	Housekeeping	Re-scores ageing food, expires offers, makes new offers

TABLE VI
MAIN DATABASE TABLES

Table	Contents
donors, ngos	Profiles, Mumbai area, approval; NGO service area and daily capacity
donations	Food details, food type and age, storage readings, AI risk and confidence, pickup status, assigned NGO, allocation note

Table	Contents
sensor_data	Donation, device, temperature, humidity, gas level, time
devices	Device ID, attached donation, last seen
offers	Donation, NGO, distance, travel time, expiry, status
notifications	Alerts for the administrator and portals

VI. EXPERIMENTAL SETUP AND RESULTS

All experiments were run on a laptop with an AMD Ryzen 5000-series processor running Windows 11, using Python and scikit-learn with a fixed random seed of 42. Accuracy, precision, recall and F1-score are reported, with macro averages so that the smaller medium class counts equally. Recall on the high-risk class is the most important metric, because a high-risk donation predicted as safe could reach people.

$$\text{Precision} = TP/(TP + FP), \text{ Recall} = TP/(TP + FN) \quad (4)$$

A. Model Comparison

Table VII lists the cross-validated performance of the six algorithms on the training set. The tree ensembles clearly outperform the other models, and Gradient Boosting and Random Forest are almost tied (macro F1 0.932 and 0.931). Logistic Regression performs poorly because risk depends non-linearly on the interaction of temperature and food age, which a linear boundary cannot capture.

TABLE VII
5-FOLD CROSS-VALIDATION ON THE TRAINING SET (MEAN ± SD)

Model	Accuracy	Macro precision	Macro recall	Macro F1
Gradient Boosting	0.953 ± 0.010	0.935 ± 0.018	0.929 ± 0.012	0.932 ± 0.015
Random Forest	0.950 ± 0.010	0.926 ± 0.015	0.938 ± 0.016	0.931 ± 0.014
Decision Tree	0.916 ± 0.008	0.880 ± 0.010	0.901 ± 0.018	0.888 ± 0.012
Support Vector Machine	0.905 ± 0.006	0.871 ± 0.005	0.906 ± 0.003	0.880 ± 0.005
k-Nearest Neighbours	0.883 ± 0.007	0.840 ± 0.012	0.837 ± 0.012	0.838 ± 0.012
Logistic Regression	0.487 ± 0.022	0.480 ± 0.021	0.522 ± 0.029	0.465 ± 0.022

B. Test-Set Performance of the Selected Model

On the 600 held-out test records, Gradient Boosting achieved 97.0% accuracy and a macro F1-score of 0.957 (Table VIII). High-risk recall was

98.7%: 232 of 235 high-risk records were identified, the other three were labelled medium, and none was labelled low risk (Fig. 2). The medium class is hardest to separate (F1 0.906) because it lies on the boundary between the other two.

TABLE VIII
TEST-SET RESULTS OF GRADIENT BOOSTING (600 RECORDS)

Class	Precision	Recall	F1-score	Support
Low	0.985	0.981	0.983	268
Medium	0.916	0.897	0.906	97
High	0.975	0.987	0.981	235
Macro average	0.959	0.955	0.957	600
Accuracy			0.970	600

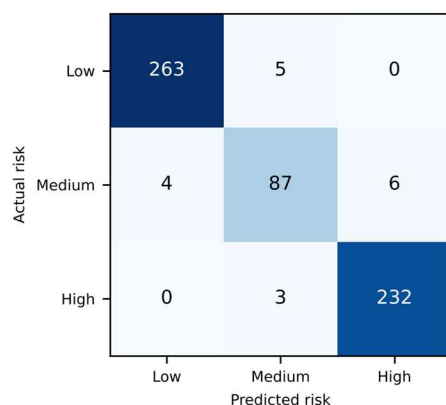


Fig. 2. Confusion matrix of Gradient Boosting on the test set.

C. Feature Importance

Permutation importance measures how much the macro F1-score drops when one feature is randomly shuffled (Fig. 3). Temperature (0.433) and food age (0.400) dominate, followed by food type (0.179), while humidity contributes little (0.026). This agrees with food-safety guidance, in which holding temperature and time are the main hazards and perishability depends on food category.

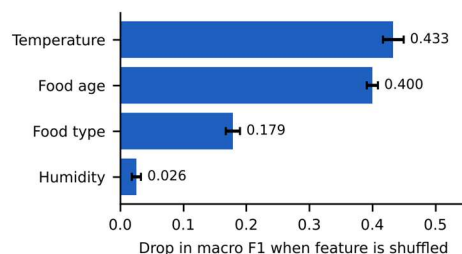


Fig. 3. Permutation feature importance of the selected model (10 repeats).

D. Feature Ablation

To confirm the role of each input, the model was retrained with one feature removed at a time (Table IX). Removing temperature or food age causes the largest losses, cutting accuracy to 64.5% and 72.7% and producing 13 and 7 high-risk records classified as low risk. Removing food type lowers accuracy to 86.8%, while removing humidity costs only 2 points, matching the importance ranking in Fig. 3.

TABLE IX
FEATURE ABLATION ON THE TEST SET (GRADIENT BOOSTING)

Feature removed	Accuracy (%)	Macro F1	High-risk recall	High → low errors
None (all four)	97.0	0.957	0.987	0
Humidity	95.0	0.927	0.983	0
Food type	86.8	0.822	0.962	0
Food age	72.7	0.580	0.911	7
Temperature	64.5	0.578	0.872	13

E. Robustness to Sensor Error

Real sensors and donor estimates are imperfect. Uniform random error was added to the test inputs, matching the DHT22 datasheet accuracy and harsher cases, and each setting was repeated 20 times (Table X). Within the DHT22 specification, accuracy is unchanged at 97.0% and no high-risk record is classified as low risk. Performance falls only gradually as the error grows: with a cheaper sensor ($\pm 2^\circ\text{C}$) accuracy is 96.4%, and even with $\pm 3^\circ\text{C}$, $\pm 10\%$ humidity and a one-hour error in food age it remains 89.2%.

TABLE X
TEST-SET PERFORMANCE UNDER SIMULATED INPUT ERROR (20 RUNS)

Added error	Accuracy (%)	Macro F1	High-risk recall	High → low (mean)
None	97.0	0.957	0.987	0
$\pm 0.5^\circ\text{C}$, $\pm 2\%$ RH (DHT22 spec)	97.0 ± 0.2	0.957	0.987	0
$\pm 0.5^\circ\text{C}$, $\pm 5\%$ RH	97.0 ± 0.3	0.957	0.988	0
$\pm 2^\circ\text{C}$, $\pm 5\%$ RH	96.4 ± 0.6	0.949	0.986	0.25
$\pm 0.5^\circ\text{C}$, $\pm 5\%$ RH, age ± 0.5 h	94.9 ± 0.7	0.925	0.983	0
$\pm 3^\circ\text{C}$, $\pm 10\%$ RH, age ± 1 h	89.2 ± 1.0	0.841	0.949	0.95

F. Training-Data Requirements

A learning curve was obtained by training on random subsets of the training set and evaluating on the same test set, averaging five subsets per size (Fig. 4). Macro F1 rises from 0.783 with 120 records to 0.916 with 600 and 0.932 with 1,200, after which gains are small. Errors of the dangerous high-to-low kind disappear from 1,200 training records upward, suggesting that a few thousand real readings would be enough to retrain the model on field data.

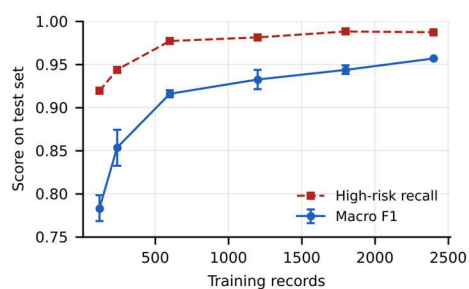


Fig. 4. Learning curve of Gradient Boosting (mean of five subsets per size).

G. Allocation Simulation

The allocation method was evaluated with a discrete-event simulation that reproduces the application's ranking, offer and expiry logic minute by minute. Each simulated day has 12 NGOs with daily capacities of 100–300 meals in random Mumbai areas and 50 donations posted between 10:00 and 23:00 from random areas, with 10–200 meals, mixed food types and storage at room temperature, hot or chilled. Travel times use the same Haversine distances and 20 km/h city speed as the application, and the safe time of each donation comes from the deployed model. An NGO accepts an offer with probability 0.75 after an exponentially distributed response time (mean 12 min), and declines if it is already full; collection takes 10 minutes plus travel. A pickup is counted as safe if it arrives before the forecast safe time. Five policies were compared over the same 50 days (Table XI, Fig. 5).

TABLE XI
ALLOCATION POLICIES OVER 50 SIMULATED DAYS (MEAN $\pm$ SD)

Policy	Meals delivered safely (%)	Late pickups (%)	Unallocated (%)	Wait to acceptance (min)
Proposed (app)	87.2 $\pm$ 6.4	4.3 $\pm$ 3.0	7.2 $\pm$ 3.7	15.9 $\pm$ 2.1

Policy	Meals delivered safely (%)	Late pickups (%)	Unallocated (%)	Wait to acceptance (min)
Proposed + 30-min buffer	85.2 $\pm$ 6.8	0.0 $\pm$ 0.0	13.6 $\pm$ 5.2	15.8 $\pm$ 2.2
Without reach filter	86.4 $\pm$ 6.5	9.0 $\pm$ 4.0	3.0 $\pm$ 2.9	16.5 $\pm$ 1.9
Nearest NGO only	83.3 $\pm$ 7.8	9.5 $\pm$ 4.3	4.6 $\pm$ 2.9	23.7 $\pm$ 4.7
Random NGO	78.0 $\pm$ 7.7	15.4 $\pm$ 6.1	4.6 $\pm$ 3.4	20.7 $\pm$ 2.9



Fig. 5. Simulated allocation outcomes (mean and standard deviation over 50 days).

The proposed method delivered the largest share of meals safely (87.2%) and cut late pickups to 4.3%, against 9.5% for nearest-NGO and 15.4% for random allocation; capacity filtering also shortened the wait for an acceptance from 23.7 to 15.9 minutes compared with offering to the nearest NGO regardless of capacity. Removing the reach filter roughly doubled late pickups (9.0%), confirming that using the safe-time forecast in allocation prevents unsafe collections. The remaining late pickups arise because the filter compares only travel time with the safe time, while the NGO's response and dispatch also take time. Adding a 30-minute buffer removed late pickups entirely, but left more donations unallocated (13.6%), a trade-off between safety and coverage. Ranking was stable when the acceptance probability was varied: at 0.5 and 0.9 the proposed method delivered 82.3% and 88.0% of meals safely, against 79.1% and 85.3% for nearest-NGO allocation.

H. Sensor Node Characterisation and Expired-Food Test

The assembled sensor node was operated indoors in Mumbai and logged every 2 s in two sessions (Table XII). In the first session the DHT22 was highly stable: temperature stayed at 32.5 °C in every reading and relative humidity varied only between

62.3% and 63.3% (standard deviation 0.28 percentage points), well within the sensor's ± 2 –5% specification. The freshly installed MQ-135 read 125 ADC counts in clean air (standard deviation 10), and a single noise excursion to 1.35 times this level triggered a false alert with an initial fixed threshold of 150 counts.

In the second session, an expired bread sample (past its best-before date, with no visible mould) was placed directly on the MQ-135. In the 30 s before the sample was placed, the sensor read 246 counts on average (standard deviation 8). After placement the reading rose within about 25 s to a mean of 393 counts (1.6 times the preceding level) and a peak of 478 counts (1.9 times). The sensor therefore responded clearly even to food that was only past its date, without visible spoilage.

TABLE XII
SENSOR NODE READINGS

Condition	Mean	SD	Range
Session 1: DHT22 temperature (°C)	32.5	0.0	32.5–32.5
Session 1: DHT22 humidity (%)	62.9	0.28	62.3–63.3
Session 1: MQ-135, clean air (counts)	125	10.1	112–169
Session 2: MQ-135, before sample (counts)	246	8.2	234–262
Session 2: MQ-135, expired bread (counts)	393	—	peak 478

The clean-air level roughly doubled between sessions, which is typical of a new MQ-135 during its 24–48 h burn-in, so fixed thresholds in raw counts are unreliable. The gas levels are therefore defined relative to the clean-air level measured after warm-up: an alert at 1.5 times (about 370 counts for the second session), matching the red-LED rule on the node, and blocking at 2.0 times (about 490 counts). With these levels the expired bread was flagged for checking (mean 393) but not blocked (peak 478), which is appropriate for bread without visible mould. This is a preliminary single-sample test; validation on more food types, including clearly spoiled cooked food, is planned. The ESP32-to-server latency over Wi-Fi was not measured in this study.

I. Functional Testing

The application was tested with an automated suite of 45 test cases that drives the real Flask application through its own routes, using the production templates and a temporary copy of the database. The suite covers registration and approval, sign-in for all three roles and protection of role-specific pages, donation posting and input validation, risk prediction, the safe-time forecast and explanations, automatic offers and capacity limits, decline and expiry hand-over, the key-protected sensor API with gas-based alerting and blocking, and the pickup workflow with notifications. All 45 cases passed (Table XIII).

TABLE XIII
FUNCTIONAL TEST SUMMARY

Test area	Cases	Passed
Registration, login and role access	11	11
Donation posting and validation	4	4
Risk, safe time and explanation	8	8
NGO eligibility and capacity limits	5	5
Offer decline, expiry and hand-over	3	3
Sensor API and gas-based blocking	6	6
Pickup status and notifications	8	8
Total	45	45

The tests also confirmed the expected behaviour of the AI features on concrete cases: cold bread and hot-held rice were rated low risk and chicken curry held for 8 hours at 32 °C high risk; the forecast for dal at 30 °C fell from 3.5 h when freshly cooked to 2.0 h after 2 hours, consistent with ageing; and the Shapley contributions summed exactly to the difference between the donation's risk score and the base score (44.1 points), as the efficiency property requires. The 5,000-meal donation exceeding every NGO's capacity was correctly left unallocated with its reason recorded, and a high-risk donation was held for the administrator instead of being offered.

J. Response Times

Table XIV reports server-side response times on the test laptop. Every AI step completes in under 20 ms; the Shapley explanation is the slowest because it evaluates 16 feature subsets against 50 background records, yet it still takes only 16 ms. Processing a sensor reading, including saving it, re-scoring the donation and checking the gas thresholds, takes

about 20 ms, and the portal pages render in 5–16 ms, so the software adds negligible delay compared with Wi-Fi transmission and travel times.

TABLE XIV
SERVER-SIDE RESPONSE TIMES (MEAN $\pm$ SD)

Operation	Time (ms)
Risk prediction per donation (200 runs)	5.7 $\pm$ 1.1
Safe-time forecast per donation (100 runs)	4.7 $\pm$ 0.8
Shapley explanation per donation (50 runs)	16.3 $\pm$ 2.2
Sensor API request, server processing (50 runs)	20.1 $\pm$ 4.0
Admin dashboard render (20 runs)	15.9 $\pm$ 0.9
Donor portal render (20 runs)	9.8 $\pm$ 0.8
NGO portal render (20 runs)	4.6 $\pm$ 0.3

VII. DISCUSSION AND LIMITATIONS

The system turns a subjective “is this still okay?” decision into a consistent, explainable risk level, and shifts safety screening from the NGO at pickup to the donor at posting, when there is still time to act. On the test set the model never classified high-risk food as low risk, the most dangerous possible error. The robustness test shows that the accuracy of a low-cost DHT22 is sufficient for the model, while the ablation shows that reliable temperature and cooking-time data matter most; this supports measuring temperature with a sensor rather than relying on donor estimates. Because offers skip NGOs that cannot arrive before the forecast safe time, the allocation module uses the AI output directly rather than only for display, and the simulation shows that this halves late pickups. Including an expected response and dispatch time in the reach check would remove the remaining late pickups, at the cost of leaving more donations unallocated.

Indirect sensing: the MQ-135 is a broad-spectrum gas sensor, not a pathogen detector, and spoilage gases may appear only after unsafe bacterial growth has begun.

Provisional gas thresholds: the alert and spoilage levels rest on a single expired-bread sample and a sensor still in burn-in, so gas-based blocking should be treated as a prototype feature until validated on more foods.

Sensor drift: MQ-series sensors are sensitive to ambient temperature, humidity and ageing, and need periodic recalibration.

Dataset: the 3,000 training records are generated from food-safety rules, so the high accuracy mainly shows that the model learns those rules reliably. Validation on real readings from donors is needed before the results can be generalised.

Single-NGO allocation: a donation larger than any one NGO’s remaining daily capacity cannot currently be allocated; splitting it across several NGOs is planned.

Simulation assumptions: NGO behaviour, donation patterns and travel speeds in the simulation are assumed rather than measured, so the absolute percentages are indicative; the comparison between policies, run on identical simulated days, is the main result.

Model-level explanations: Shapley values explain the model’s reasoning, not the true microbial state of the food. Predictions support, and do not replace, human food-safety judgement.

Ethics and compliance: food donation in India is governed by FSSAI’s surplus-food regulations [15]. The system keeps an audit trail of readings and risk decisions and stores only the personal data needed for pickups.

VIII. CONCLUSION AND FUTURE WORK

This paper presented a Smart Food Donation Management System that combines an ESP32 sensor node, an explainable Gradient Boosting risk model (97.0% test accuracy, 98.7% high-risk recall), a safe-time forecast and safety-aware NGO allocation in a Flask–SQLite platform with separate portals for donors, NGOs and an administrator. The system makes food-safety screening practical for small donors and helps NGOs prioritise pickups before food becomes unsafe.

Future work will validate the gas thresholds on spoiled cooked food, measure end-to-end latency over Wi-Fi, add the response-time buffer to the reach check, split large donations across several NGOs, collect a larger field dataset from partner restaurants and NGOs in Mumbai, add an ammonia-specific or pH sensor and time-series models on continuous readings, introduce SMS or WhatsApp alerts and volunteer routing, and run a pilot deployment to

measure meals recovered and food diverted from landfill.

REFERENCES

- [1] United Nations Environment Programme, *Food Waste Index Report 2024*. Nairobi, Kenya: UNEP, 2024.
- [2] Statista, “Total annual household food waste in selected countries,” based on UNEP Food Waste Index Report 2024.
- [3] L. Breiman, “Random forests,” *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.
- [4] S. M. Lundberg and S.-I. Lee, “A unified approach to interpreting model predictions,” in *Advances in Neural Information Processing Systems 30*, 2017.
- [5] M. Sammer, K. Seong, N. Olvera, S. L. Gronseth, E. Anderson-Fletcher, J. Jiao, A. Reese *et al.*, “AI-FEED: Prototyping an AI-powered platform for the food charity ecosystem,” *Int. J. Comput. Intell. Syst.*, vol. 17, Art. no. 259, 2024, doi: 10.1007/s44196-024-00656-9.
- [6] J. Nurgazina and G. Reiner, “Opportunities of digital solutions for addressing food donation challenges,” *J. Responsible Technol.*, vol. 27, 2026.
- [7] S. Sanyal, V. K. Singh, F. Xhafa, B. Sanyal and S. Mukhopadhyay, “A game theoretic framework for surplus food distribution in smart cities and beyond,” arXiv:2102.04929, 2021.
- [8] K. Valarmathi, M. Mageshwari, N. Prabakaran and S. Rajendran, “Food wastage analysis framework using comprehensive IoT-based multi-sensor,” *Global NEST J.*, vol. 27, no. 10, Art. no. 07826, 2025, doi: 10.30955/gnj.07826.
- [9] T. Mumbre, T. Salunkhe and R. Kadu, “IoT-based food spoilage detection system using ammonia sensing and environmental monitoring,” *IJERT*, vol. 15, no. 4, 2026.
- [10] S. Senthamarai Selvi and P. Devi, “Food quality monitoring system using IoT-based pH, gas, and temperature sensors,” *Int. Res. J. Mod. Eng. Technol. Sci.*, vol. 7, no. 6, pp. 2518–2522, 2025, doi: 10.56726/IRJMETS79599.
- [11] N. Aloysius and J. Ananda, “A circular economy approach to food security and poverty: A case study in food rescue in Sri Lanka,” *Circ. Econ. Sustain.*, vol. 3, no. 4, pp. 1919–1940, 2023, doi: 10.1007/s43615-023-00255-4.
- [12] U.S. Department of Agriculture, Food Safety and Inspection Service, “Danger zone (40 °F–140 °F),” food safety guidance.
- [13] World Health Organization, *Five Keys to Safer Food Manual*. Geneva: WHO, 2006.
- [14] F. Pedregosa *et al.*, “Scikit-learn: Machine learning in Python,” *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.
- [15] Food Safety and Standards Authority of India, *Food Safety and Standards (Recovery and Distribution of Surplus Food) Regulations*, 2019.
- [16] J. H. Friedman, “Greedy function approximation: A gradient boosting machine,” *Ann. Statist.*, vol. 29, no. 5, pp. 1189–1232, 2001.