ABSTRACT

Food donation and waste management

The Food Donation and Waste Management System (FoodShare) is a dedicated web-based platform designed to efficiently bridge the gap between food donors and receiving organizations, significantly contributing to the reduction of food waste and the mitigation of hunger. The system provides a centralized, user-friendly interface for various stakeholders: **Donors** (restaurants, businesses) can easily post surplus food availability; **Receivers** (charities, NGOs) can browse and request available donations; and **Delivery Partners** facilitate the logistics of safe transport.

The core objective is to create a seamless, transparent, and trackable process for redirecting edible surplus food. Built on modern web technologies (HTML, CSS, JavaScript, and Server-Side Integration), the platform incorporates secure user authentication, role-based access control, and a dynamic interface. This project aims to maximize the societal benefit of food resources by promoting collaboration, efficiency, and sustainability within the community food chain.

ACKNOWLEDGEMENT

I am greatly indebted to our project guide **Mr. M. R. Chennoji** for her able guidance throughout this work. It has been an altogether different experience to work with her and I would like to thank her for her help, suggestions and numerous discussions.

I gladly take this opportunity to thank **Dr. Rajurkar A.M** (Head of CSE DEPT, MGM's College of Engineering, Nanded).

I am heartily thankful to **Dr. Lathkar G. S.** (Director, MGM's College of Engineering, Nanded) for providing facility during the progress of the project; also for her kindly help, guidance and inspiration.

Last but not least I am also thankful to all those who help me directly or indirectly to develop this project and complete it successfully.

With Deep Reverence,

Sayali Dattatrayrao Joshi

Shreya Ganesh Ghogare

Chapter 1

INTRODUCTION & BACKGROUND

The issue of food waste and food insecurity represents a profound global paradox. On one hand, billions of people worldwide suffer from hunger and malnutrition, lacking consistent access to nutritious food. On the other hand, a significant portion of the global food supply—estimated to be one-third—is lost or wasted annually across the supply chain, from production and processing to retail and final consumption. This inefficiency has massive economic, environmental, and social consequences. Economically, it represents a loss of resources; environmentally, wasted food contributes significantly to greenhouse gas emissions in landfills; and socially, it exacerbates the problem of hunger.

Effective food donation and waste management systems are crucial for mitigating this paradox. They serve as a critical bridge, redirecting surplus, edible food from generators (such as restaurants, caterers, and food manufacturers) to charitable organizations and individuals in need. The motivation for developing modern management solutions is driven by a commitment to sustainability, social equity, and compliance with global goals, such as the UN Sustainable Development Goal 12.3, which aims to halve per capita global food waste by 2030.

1.1 Background and Motivation

Food waste and hunger are two persistent global challenges that exist paradoxically alongside one another. A significant amount of edible food is discarded daily by commercial establishments, while vulnerable populations struggle with food insecurity. The inefficiency in the distribution chain, coupled with a lack of organized and timely communication between food surplus generators and charitable organizations, results in a massive loss of resources and a missed opportunity to address social needs.

The motivation behind the Food Donation and Waste Management System is to digitize and streamline the process of food surplus redirection. By creating a centralized, accessible web platform, the system aims to:

- Minimize the economic and environmental impact of food waste.

- Maximize the utilization of surplus food for the benefit of the needy.
- Provide a transparent and trackable mechanism for food donations.

1.2 Problem Statement

Traditional methods of food donation often rely on manual communication, are time-consuming, and suffer from logistical hurdles, leading to food expiring before it can be effectively donated or collected. Specifically, the core problems addressed by this project include:

- **Decentralized Information:** Donors lack an instant, dedicated channel to list food availability to multiple receivers simultaneously.
- **Logistical Inefficiency:** The process of matching available food with the nearest charitable organization is slow and inefficient.
- **Lack of Tracking:** There is no centralized mechanism to track the status of a donation, from initial listing to final pickup and delivery.

1.3 Objectives of the Project

The primary objectives of developing this system are based on the core functionalities identified in the front-end design and back-end logic Secure User Authentication: To implement a robust registration and login system that supports distinct roles for Donor, Receiver, and Admin. Role-Based Access Control (RBAC) To ensure that access and functionality are appropriately segmented, allowing donors to list items and receivers to reserve them. Dynamic Food Listing To allow donors to post surplus food items with necessary details like name, quantity, and expiry date, and track the status of the item (available, reserved, donated).Reservation Management: To enable receivers to place reservations on available food items, and for the system to manage the reservation status (reserved, picked_up, cancelled).

Data Integrity and Security: To securely store user and transaction data using a structured MySQL database schema and protect user credentials through secure session management and database transactions.

1.4 Scope and Report Organization

The Food Donation and Waste Management System is implemented as a web-based application utilizing a Client-Server architecture. The front-end user interface is built with HTML, CSS, and JavaScript, while the server-side logic and database interaction are handled by PHP and MySQL.

LITERATURE REVIEW AND SYSTEM ANALYSIS

Functional and Non-Functional Requirements are two fundamental categories used in systems engineering and software development to define the characteristics and constraints of a software system. They are essential because they articulate exactly what the system must do and how well it must do it, guiding the entire development, testing, and implementation process.

2.1 Functional Requirements

Functional requirements define what the Food Donation and Waste Management System (FoodShare) must do to satisfy the user's needs. The system is designed around three primary user roles: Donor, Receiver, and Admin.

2.1.1 User Management & Authentication

Registration: The system must allow new users to register, selecting their role (donor, receiver, or admin). Donors must provide additional information like Restaurant Name, Contact Number, and Address. The system must present a unified sign-up form allowing users to input a unique Email and a secure Password. The user must explicitly select their Role from a pre-defined list: donor, receiver, or admin. Conditional Field Requirement: If the user selects the donor role, the system must dynamically require and validate additional specific fields: Restaurant Name, Contact Number, and Address. This data is stored in the users table for donor identification and logistics.

Login/Logout: Users must be able to securely log in using their email and password, with their session maintained via PHP sessions. Separate login interfaces are provided for different roles (e.g., admin_login.php, donor_login.php). Users must be able to securely log in using their registered email and password. Authentication logic (login.php, not provided but referenced) must verify credentials and the user's role against the users table. PHP sessions must be used to maintain the logged-in state and track the user_id and role. Separate login interfaces (admin_login.php, donor_login.php) must be provided for focused access points.

Role-Based Access Control (RBAC): The system must restrict functionality based on the user's role. For example, only donors can list food items, and only receivers can view and reserve items. Upon successful login, the user must be redirected to a dashboard or area specific to their role. The system must restrict visibility and execution of certain actions based on the user's role (e.g., a Receiver cannot post a new food listing).

2.1.2 Donor Functionality

Food Listing: Donors must be able to create a new donation listing by specifying details such as food name, description, quantity, expiry date, and category. This data is stored in the food_items table. Donors must be able to access a dedicated form to post a new donation. The listing must require mandatory details, including Food Name, Description, Quantity (e.g., number of meals, weight), Category (e.g., baked goods, prepared meals), and a clear Expiry Date.The system must automatically associate the listing with the Donor's user_id and their Restaurant Name. Optionally, the system should allow for an Image URL to be uploaded or linked to provide a visual representation.

Status Management: Donors must be able to view their posted items and track the item's status, which changes from 'available' to 'reserved' and finally to 'donated'. The functionality related to Donation Status Management is crucial for ensuring the smooth and transparent flow of surplus food from the donor to the receiver. When a Donor successfully creates a new listing detailing the surplus food, the item is immediately assigned the status of 'available' within the system's database. Donors are provided with a dedicated interface where they can view all their active listings and monitor the current status of each item. The status is automatically updated to 'reserved' the moment a Receiver (such as a charitable organization) claims the item, which immediately prevents other receivers from attempting to reserve the same food, thus ensuring efficient resource allocation. Finally, after the food has been successfully picked up and the transaction is complete, the Donor is responsible for using a specific function to finalize the transaction, changing the status to 'donated'.

Donation History: Donors must be able to view a log of all their past and active donations. Donors must have a dedicated view listing all past and current donations, including the status, creation date, and—if applicable—the name of the Receiver organization.

2.1.3 Receiver Functionality

Browse/Search: Receivers must be able to view a list of all currently 'available' food items. The list should be searchable and filterable (e.g., by location, category). Receiver Functionality module is designed to provide charitable organizations and individuals with an effective means of discovering and securing available food donations, ensuring timely redistribution. The core of this functionality begins with Browse/Search, where authenticated Receivers are presented with a comprehensive and dynamically updated list of all food items currently marked as **'available'** by Donors.

Reservation: Receivers must be able to reserve an available food item. This action creates an entry in the reservations table and updates the item status in the food_items table to 'reserved'. o help Receivers find the most relevant donations quickly, the platform must offer robust search capabilities and flexible filtering options, allowing them to narrow down the listings based on key criteria such as geographic location (using the Donor's address), food category (e.g., produce, prepared meals), and potentially quantity or expiry date. Once a suitable food item is identified, the Reservation process is initiated. This is a critical step where the Receiver claims the item, which simultaneously creates a new record in the dedicated reservations table, linking the specific food_id and the receiver_id.

Reservation Management: Reservation Management provides a centralized view for Receivers, allowing them to track all their transactions. They can monitor the status of their pending reservations and, after successful pickup, they are expected to confirm the completion of the transaction, which may update the reservation status to 'picked_up' in the system.

2.1.4 Administrator Functionality

Dashboard View: The Admin must have a centralized dashboard to overview system statistics (total users, total donations, total reserved items). The Administrator's primary access point is a centralized Dashboard View, which functions as a system monitoring tool. This dashboard must aggregate and display key performance indicators (KPIs) and summarized statistics in real-time. Crucial metrics include the total number of registered users (segregated by role: Donors, Receivers, and other Admins), the total number of donation listings created to date, and the breakdown of all transactions (e.g., total items currently 'available', 'reserved', and successfully 'donated'). This overview allows the Admin to quickly assess platform activity, growth, and the efficiency of the food redistribution process.

User Management: The Admin must be able to view, modify, or delete any user account (Donor or Receiver). The Admin module provides comprehensive User Management tools essential for maintaining the integrity and security of the community. Administrators must be able to view a list of all registered user accounts, including both Donors and Receivers. From this view, the Admin can perform full CRUD (Create, Read, Update, Delete) operations: they must be able to modify user details (such as correcting contact information or changing a user's role), and critically, they must have the ultimate authority to delete or suspend any user account if there are security concerns or policy violations.

Content Moderation: The Admin must be able to view and manage all food listings and reservations, intervening if necessary. Content Moderation ensures the quality and accuracy of the donation listings and reservations. The Admin must possess the ability to view and manage all food listings from the food_items table and all corresponding reservations from the reservations table. In cases of user disputes, system errors, or food safety concerns, the Administrator must be able to intervene and manually change the status of any food item (e.g., from 'reserved' back to 'available') or modify the details of a reservation. This level of control is necessary for maintaining trust and operational integrity within the food donation network.

SYSTEM DESIGN AND ARCHITECTURE

The system utilizes a traditional Three-Tier Architecture with separate user portals for Donors, Receivers, and an Admin, along with a dedicated logistics/delivery component. The FoodShare Platform is designed to efficiently connect food donors (e.g., restaurants, businesses, and individuals) with receivers (e.g., NGOs, charities, and needy individuals) to reduce food waste and combat food insecurity.The core functionality revolves around three main user groups: Donors (to list surplus food), Receivers (to reserve and collect food), and Administrators (to manage the platform and users). A fourth module manages Delivery/Logistics.

3.1 System Architecture Overview

3.1.1 Architecture Model: The system employs a conventional Three-Tier Architecture, which logically separates the application into a Presentation, Application, and Data Layer.

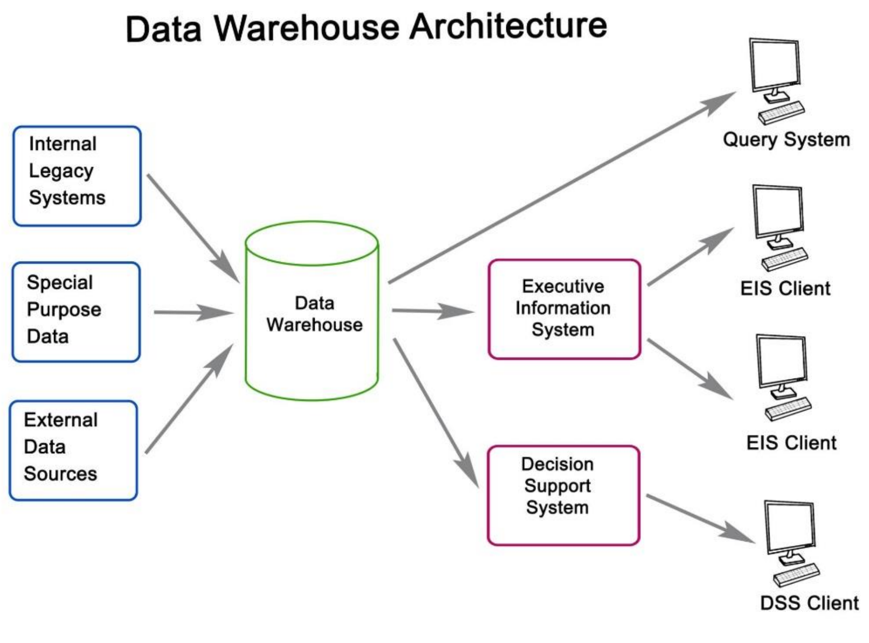


Fig 2.1 Overall Architecture

The FoodShare platform adopts a robust, scalable, and modular Three-Tier Architecture to separate user presentation, business logic, and data management. This approach enhances development, security, and maintenance by isolating concerns into distinct layers.

The Presentation Tier is the user interface, developed using HTML, CSS, and JavaScript, with server-side rendering handled by PHP for user-specific entry points like donor_login.php, receiver_login.php, and admin_login.php. It is responsible solely for displaying information and collecting user input. The Application Tier, often hosted on a Web Server environment like XAMPP, executes the core business logic using PHP. This is where crucial operations like authentication, validation, food item creation, and reservation processing are managed, acting as the intermediary between the user interface and the database. Finally, the Data Tier leverages a MySQL Relational Database Management System (RDBMS) to ensure data integrity and persistence. This tier is comprised of the core foodshare_db and a dedicated foodshare_delivery_db for logistics data, ensuring that all application data is stored reliably and can be retrieved efficiently based on business requests.

3.2 Modular Breakdown and Business Logic

The entire system functionality is partitioned into four primary logical modules, each corresponding to a specific user role or process flow. The Authentication Module serves as the gateway, processing login requests from login_page.php and role-specific pages, validating credentials against the users table, and enforcing Role-Based Access Control (RBAC) to redirect users to their respective dashboards (Donor, Receiver, or Admin). The Donor Management Module is the core tool for organizations or individuals listing surplus food. Donors input details like food_name, quantity, and expiry_date via their portal, creating a new record in the food_items table with the initial status of 'available'. The Receiver/Reservation Module enables registered charity organizations or individuals to browse available food listings. Upon selecting an item, a reservation is created in the reservations table, which simultaneously triggers an update of the corresponding food_items.status to 'reserved', preventing over-allocation. The Administrator Module, accessed via admin_login.php, is the central command for platform oversight, managing user approvals, moderating content, and crucially, reviewing applications from potential delivery personnel stored in the separate delivery_applications table within foodshare_delivery_db.

3.3 Data Design (Database Schema)

The persistent data storage is implemented across two structured MySQL databases, ensuring a clear separation of core application data from administrative logistics. The main foodshare_db contains three critical entities. The users table is the master record for all platform members, defining their unique email, secure password, and vital role (donor, receiver, or admin). The food_items table is central to the donation process, holding all details of the food available, including a user_id foreign key linking it back to the donating organization. The reservations table models the transactional relationship, using foreign keys to link a specific food_items record (food_id) to the reserving users record (receiver_id), tracking the process through status states like 'reserved', 'picked_up', and 'cancelled'. A secondary database, foodshare_delivery_db, manages the independent application workflow for delivery personnel via the delivery_applications table, allowing administrators to track and manage the status of these applicants without cluttering the main transaction database.

SYSTEM IMPLEMENTATION

The implementation phase began with the creation of the Data Tier, where the two relational databases—foodshare_db and foodshare_delivery_db—were initialized using their respective SQL schema scripts. This process established the key tables (users, food_items, reservations, and delivery_applications) and enforced Foreign Key Constraints to ensure data integrity and reliable relationships between entities. Simultaneously, the Application Tier was configured on a Web Server (e.g., XAMPP), ready to execute the PHP scripts that house the business logic.

4.1 Technology Stack and Environment Setup (Detailed)

This section details the specific technologies chosen for the FoodShare platform and the necessary environment setup to ensure the application runs correctly, supporting the outlined three-tier architecture.

4.1.1 Programming Language: PHP (Application Tier)

PHP is the primary language used to implement the Application Tier (Business Logic) of the system. It handles all server-side operations, which are critical for platform functionality and security. Specifically, PHP scripts are responsible for managing user sessions (as seen by the session_start() and session destruction logic in the login files), executing database queries (connecting to MySQL and performing CRUD operations), and enforcing business rules (such as validating donation data and managing the status of food items and reservations). The integration of PHP allows for dynamic content generation, essential for rendering user-specific dashboards for Donors, Receivers, and Admins.

4.1.2 Database Management: MySQL (Data Tier)

MySQL is the chosen Relational Database Management System (RDBMS), serving as the persistent store for the Data Tier. The system utilizes two separate database instances to logically organize data: the core application data is stored in foodshare_db (containing users, food_items, and reservations), while logistics-related data is housed in foodshare_delivery_db (containing delivery_applications). The use of MySQL ensures data consistency, reliability, and the enforcement of Referential Integrity through Foreign Keys. The structure and constraints are defined by the provided SQL scripts (foodshare_db.sql and delivery_applications_db.sql).

5.1.3 Web Server Environment (Application/Presentation Tier Bridge)

The PHP scripts and HTML pages must be hosted on a compatible web server environment. A lightweight local server setup, such as XAMPP (or WAMP/MAMP), is typically used during development, as it conveniently bundles the Apache HTTP Server, MySQL, and PHP. For production deployment, a robust server like Apache or Nginx is required, either on dedicated hardware or a cloud platform. This server environment is responsible for:

1. Handling all incoming HTTP requests (e.g., from login_page.php).
2. Interpreting the PHP scripts to execute the server-side logic.
3. Serving the resulting static and dynamic HTML pages back to the client browser.

4.1.4 Frontend Technologies: HTML, CSS, and JavaScript (Presentation Tier)

The Presentation Tier is constructed using fundamental web technologies:

- HTML is used for structuring the content of all web pages, including the login forms (donor_login.php, receiver_login.php, etc.) and the dashboard layouts.
- CSS is implemented for styling and visual presentation, ensuring a consistent and user-friendly interface. The inline styles observed in the provided files confirm its use for layout, colors, and responsive design elements.
- JavaScript is used for client-side interactivity, form validation (prior to server submission), and enhancing the user experience without requiring constant server round-trips. It would also be essential for integrating third-party APIs like maps (for location tracking and matching).

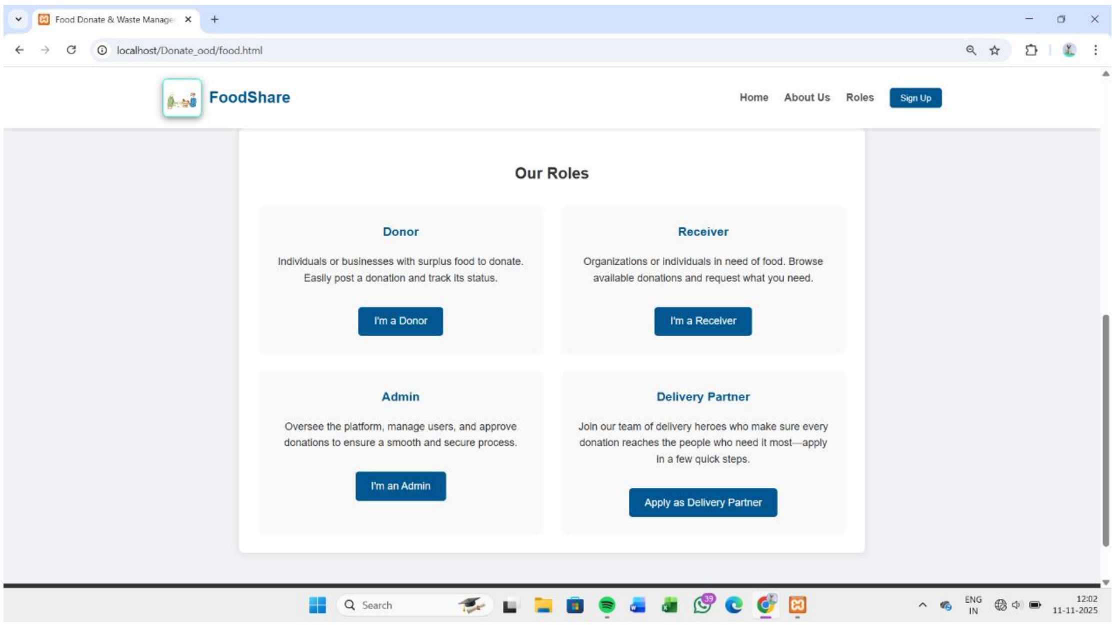
4.2 Implementation of Core Modules (Function-Point Implementation)

This section details the actual code-level logic and database interaction implemented for the primary functionalities of the FoodShare platform.

4.2.1 Authentication and Session Management

A. Login Page Implementation (login_page.php, donor_login.php, etc.):

The implementation of all login entry points (e.g., donor_login.php, receiver_login.php, admin_login.php) begins with the mandatory PHP function session_start(). For security, the first block of code on each login page includes a conditional check to see if a session already exists (isset($_SESSION['user_id'])). If a user is already authenticated, the script executes a session destruction sequence ($_SESSION = array(), session_destroy()) to enforce a fresh, explicit login, mitigating the risk of session fixation or unauthorized access if a user leaves the page open.


Food Donate & Waste Manage
localhost/Donate_ood/food.html
FoodShare
Home
About Us
Roles
Sign Up
I'm a Donor
I'm a Receiver
I'm an Admin
Apply as Delivery Partner

Our Roles

Donor

Individuals or businesses with surplus food to donate. Easily post a donation and track its status.

Receiver

Organizations or individuals in need of food. Browse available donations and request what you need.

Admin

Oversee the platform, manage users, and approve donations to ensure a smooth and secure process.

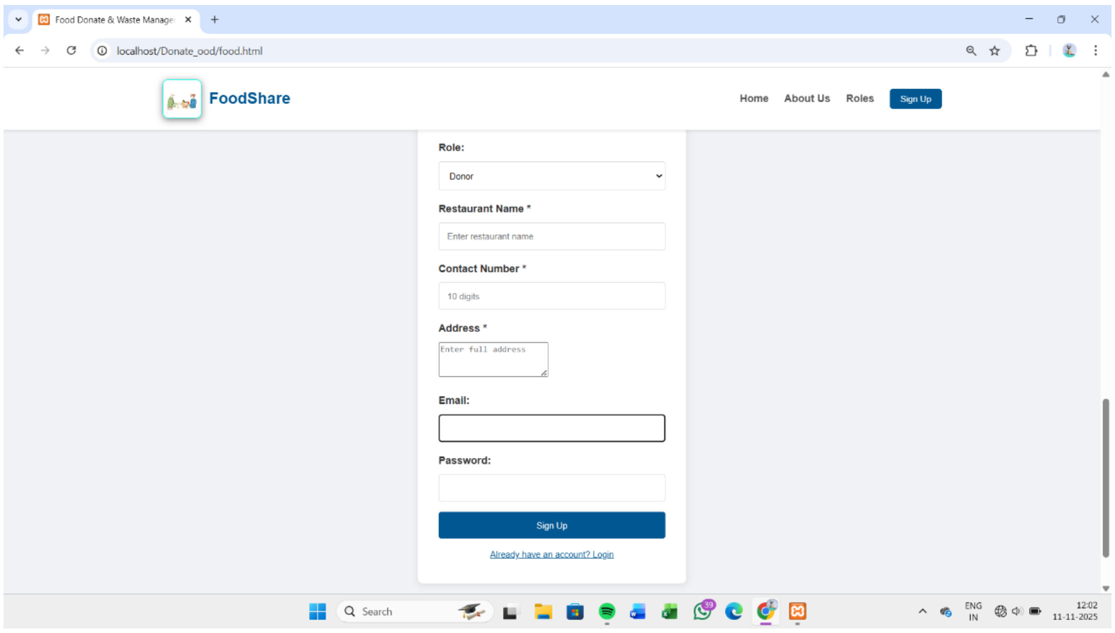
Delivery Partner

Join our team of delivery heroes who make sure every donation reaches the people who need it most—apply in a few quick steps.

4.1 Home Page

The implementation of all login entry points (e.g., donor_login.php, receiver_login.php, admin_login.php) begins with the mandatory PHP function session_start(). For security, the first block of code on each login page includes a conditional check to see if a session already exists (isset($_SESSION['user_id'])). If a user is already authenticated, the script executes a session destruction sequence ($_SESSION = array(), session_destroy()) to enforce a fresh, explicit login, mitigating the risk of session fixation or unauthorized access if a user leaves the page open. The HTML form on each page includes a hidden input field (<input type="hidden" name="redirect" value="..." />) which tells the processing script the user's intended role, a key component for Role-Based Access Control (RBAC).

B. Login Processing (login.php - Inferred):



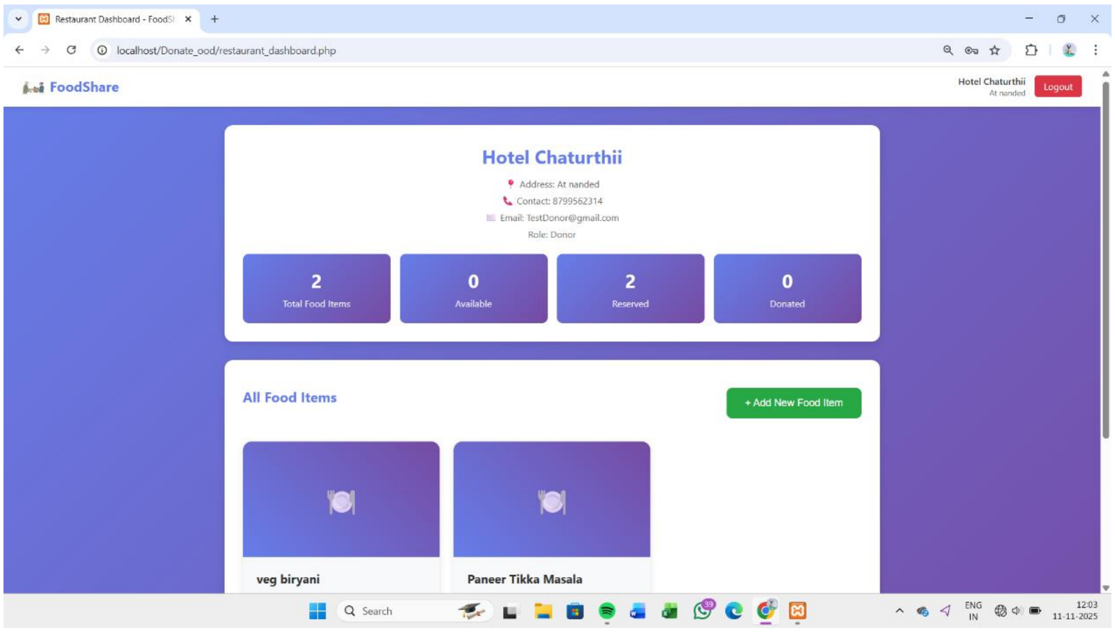
4.2 (a)Login page

All login forms submit their credentials to a central PHP script, typically named login.php. This script first retrieves the submitted user inputs (email and password) using the $_POST superglobal. It then connects to the MySQL database and executes a SELECT query on the users table to locate the record matching the submitted email. For security, the script does not simply compare passwords; instead, it uses functions like password_verify() to securely check the submitted plaintext password against the hashed password retrieved from the database. Upon successful authentication, the user's essential data, specifically their user_id and role, are immediately stored in the $_SESSION array. Finally, the script uses the value from the hidden redirect field to programmatically redirect the user to the correct, role-specific dashboard (e.g., /donor/dashboard.php), thereby completing the authentication and authorization workflow.

4.2.2 Donor Functionality

A. New Donation Listing:

When a donor submits the "Add Donation" form, a dedicated PHP script handles the POST request. The script immediately performs rigorous server-side validation on all inputs: for instance, verifying that the quantity is a non-zero integer, that the expiry_date is a valid date that has not already passed, and ensuring required fields are not empty. Once validation passes, the script executes an INSERT query on the food_items table. This query populates fields such as restaurant_name, food_name, and quantity, automatically linking the item to the logged-in donor by recording the session’s user_id as a foreign key. Critically, the status field for the new record is defaulted to the ENUM value ‘available’.


Restaurant Dashboard - FoodSh
localhost/Donate_ood/restaurant_dashboard.php
FoodShare
Hotel Chaturthii
At nanded
Logout

Hotel Chaturthii

Address: At nanded
Contact: 8799562314
Email: TestDonor@gmail.com
Role: Donor

2	0	2	0
Total Food Items	Available	Reserved	Donated

All Food Items

+ Add New Food Item

veg biryani

Paneer Tikka Masala

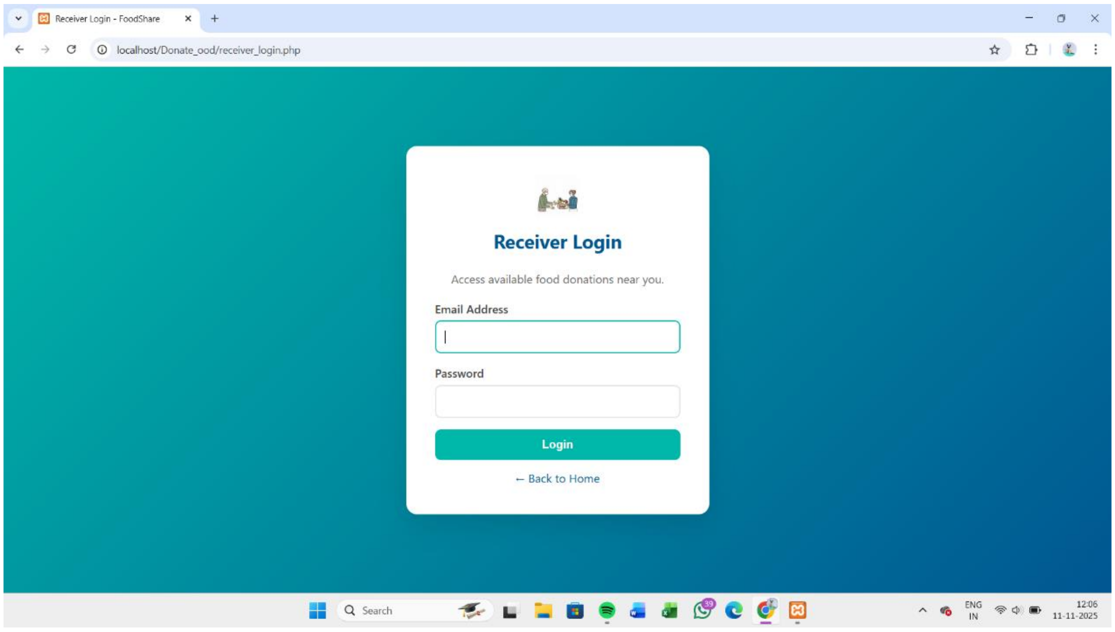
4.2.(b)Login Functionality

B. Donation Status Tracking:

The Donor Dashboard utilizes a SELECT query on the food_items table, filtered by the logged-in user's user_id, to retrieve all of their donation history. The implementation dynamically displays the item's current status (e.g., 'available', 'reserved', or 'donated'), which is determined by querying and joining data from both the food_items and reservations tables to reflect the item's life cycle accurately.

4.2.3 Receiver Functionality

The Receiver Dashboard implements the primary search function by executing a SELECT query on the food_items table, using a crucial WHERE clause to filter results only for records where the status is equal to 'available'. The results are then rendered on the dashboard, typically including data inferred to be essential for logistics, such as the donor's location (address/city from the users table) and the food's quantity. The Receiver Dashboard implements the primary search function by executing a SELECT query on the food_items table, using a crucial WHERE clause to filter results only for records where the status is equal to 'available'. The results are then rendered on the dashboard, typically including data inferred to be essential for logistics, such as the donor's location (address/city from the users table) and the food's quantit


Receiver Login - FoodShare
localhost/Donate_ood/receiver_login.php
Login
← Back to Home

Receiver Login

Access available food donations near you.

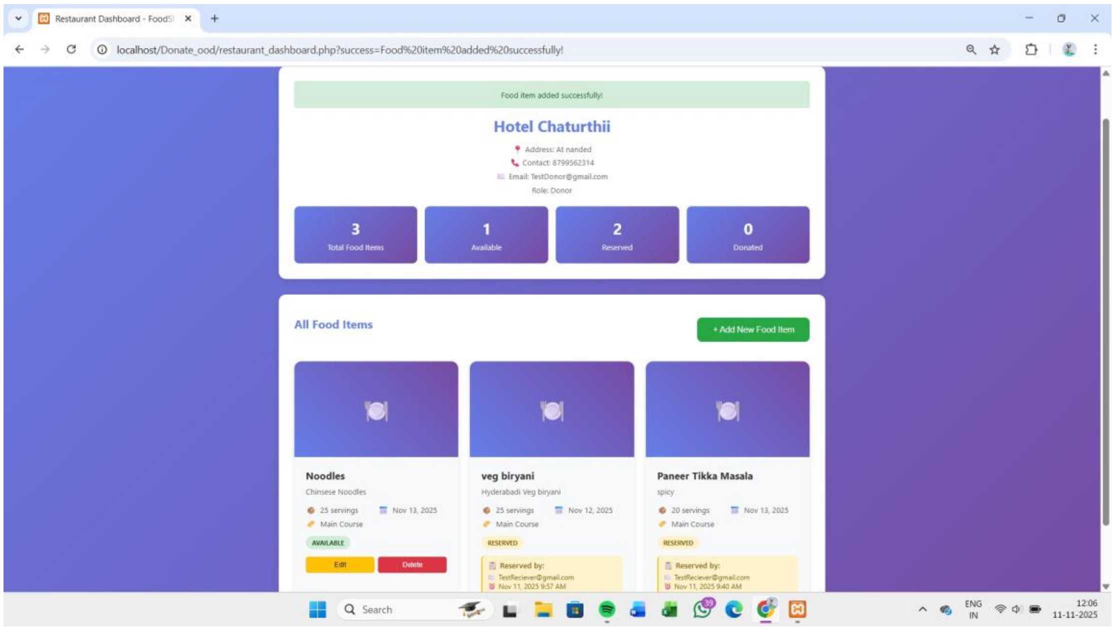
Email Address

Password

4.2(c) Reciver Login

A. Browsing Available Food

The Receiver Dashboard implements the primary search function by executing a SELECT query on the food_items table, using a crucial WHERE clause to filter results only for records where the status is equal to 'available'. The results are then rendered on the dashboard, typically including data inferred to be essential for logistics, such as the donor's location (address/city from the users table) and the food's quantity.


Restaurant Dashboard - FoodS
localhost/Donate_ood/restaurant_dashboard.php?success=Food%20item%20added%20successfully!
+ Add New Food Item
Edit
Delete

Food item added successfully!

Hotel Chaturthii

Address: At nanded
Contact: 8799562314
Email: TestDonor@gmail.com
Role: Donor

3 Total Food Items | 1 Available | 2 Reserved | 0 Donated

All Food Items

Noodles
Chinsese Noodles
25 servings Nov 13, 2025
Main Course
AVAILABLE

veg biryani
Hyderabadi Veg biryani
25 servings Nov 12, 2025
Main Course
RESERVED
Reserved by:
TestReciever@gmail.com
Nov 11, 2025 9:57 AM

Paneer Tikka Masala
spicy
20 servings Nov 13, 2025
Main Course
RESERVED
Reserved by:
TestReciever@gmail.com
Nov 11, 2025 9:40 AM

B. Reservation Logic:

The reservation process is implemented as a critical database transaction to ensure system reliability and prevent resource conflict. When a receiver clicks to reserve an item, the script initiates this transaction, which consists of two sequential and dependent database operations:

1. An INSERT query is executed on the reservations table, recording the food_id and the current session's receiver_id, and setting the reservation status to 'reserved'.
2. An UPDATE query is immediately executed on the food_items table, changing the corresponding item's status from 'available' to 'reserved'. This atomic, two-step operation ensures the item is instantly locked for the reserving receiver, making it unavailable to other users and guaranteeing that the data integrity between the two tables is maintained.

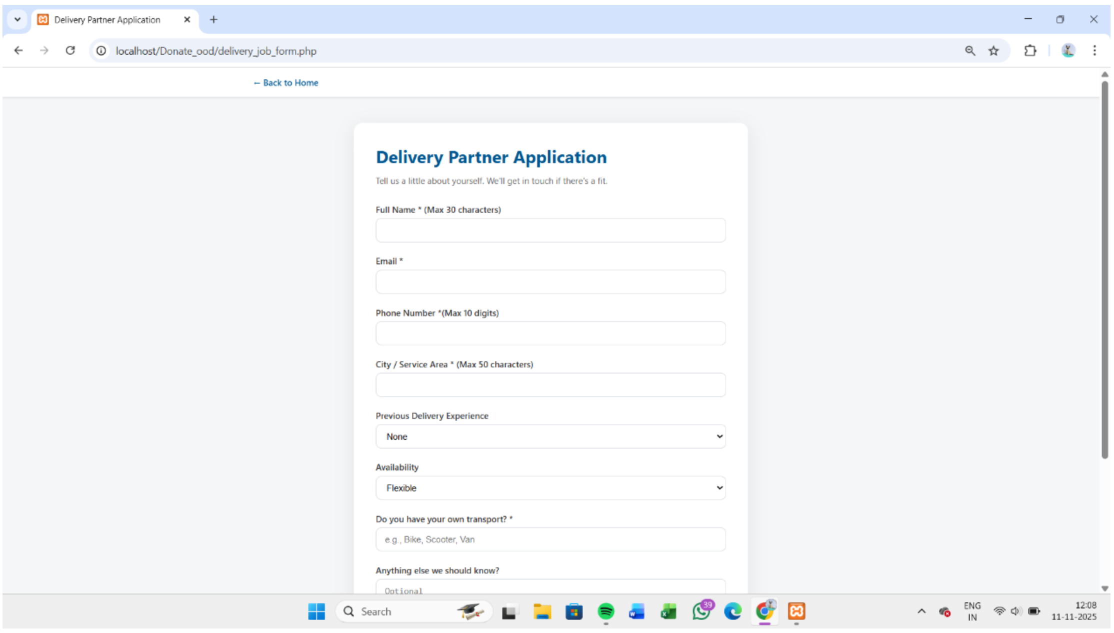
4.2.4 Administrator Functionality

A. User and Donation Management:

The Admin Panel, accessed via admin_login.php, is implemented to provide comprehensive platform oversight. It runs complex SELECT queries involving JOINs across the users, food_items, and reservations tables to generate detailed reports and overall system statistics. The platform utilizes UPDATE and DELETE queries to enable the administrator to perform moderation actions, such as removing inappropriate food listings from food_items or managing user accounts (e.g., suspending or deleting records from the users table).

B. Delivery Application Processing:

For the logistics workflow, the implementation requires a separate database connection to the foodshare_delivery_db. The Admin views a list of new applicants by executing a SELECT query on the delivery_applications table, filtered by the status of 'pending'. The core action here is the UPDATE query, which the Admin uses to change the applicant's status field to either the 'approved' or 'rejected' ENUM value, completing the application processing lifecycle.


Delivery Partner Application

localhost/Donate_ood/delivery_job_form.php

← Back to Home

Delivery Partner Application

Tell us a little about yourself. We'll get in touch if there's a fit.

Full Name * (Max 30 characters)

Email *

Phone Number *(Max 10 digits)

City / Service Area * (Max 50 characters)

Previous Delivery Experience: None

Availability: Flexible

Do you have your own transport? *

e.g., Bike, Scooter, Van

Anything else we should know?

Optional

For the logistics workflow, the implementation requires a separate database connection to the foodshare_delivery_db. The Admin views a list of new applicants by executing a SELECT query on the delivery_applications table, filtered by the status of 'pending'. The core action here is the UPDATE query, which the Admin uses to change the applicant's status field to either the 'approved' or 'rejected' ENUM value, completing the application processing lifecycle.

WEB DESIGN AND DEVELOPMENT

5.1 Web Design Principles and User Experience (UX)

The system's design adheres to user-centric principles, ensuring that the interface is intuitive and efficient for each of the three distinct user roles.

5.1.1 User-Centric Design and Role Differentiation:

The primary design goal is clarity and efficiency, minimizing the steps required for core tasks. The application is segregated into distinct dashboards for the Donor, Receiver, and Administrator.

The Donor Dashboard is optimized for data input (adding new listings) and tracking (viewing status updates for reservations).

The Receiver Dashboard is focused on a browsing interface for instantly locating available food items and a streamlined process for initiating a reservation.

The Admin Dashboard is designed for data aggregation and moderation, displaying overall system statistics, user lists, and pending applications.

5.1.2 Visual Consistency and Branding:

Visual appeal is established through a consistent theme across all pages, which reinforces professionalism.

A strategic Color Palette is implemented to help users quickly differentiate entry points: for example, the login pages display distinct gradient backgrounds (Donor: Purple/Blue; Receiver: Teal/Dark Blue; Admin: Red/Pink) to aid role identification.

The platform utilizes a clear logo (e.g., OIP (4).jpg as seen in the files) and a clean, readable Typography (e.g., 'Segoe UI', Tahoma) to maintain consistency and clarity across all textual content, alerts, and navigation.

5.1.3 Responsive Design (Inferred):

While the provided snippets focus on desktop styles, the platform must adopt responsive design principles using CSS to ensure usability across all device sizes (desktops, tablets, and smartphones). This involves using fluid layouts and implementing media queries to dynamically adjust element sizing, navigation menus, and form layouts, ensuring that forms remain easy to fill out regardless of screen size.

5.2 Web Development Implementation

The development process involves both frontend markup and styling, and backend integration using PHP to manage dynamic content.

5.2.1 Frontend Development (HTML/CSS/JavaScript)

This is the development of the Presentation Tier, focused on user interaction.

Structural Markup (HTML): Semantic HTML is used to structure all forms and content. Forms utilize appropriate input types (e.g., type="email", type="password") and the required attribute for basic client-side input constraints.

Styling (CSS): Styling is applied using both embedded <style> blocks and potentially external CSS files. Key CSS techniques utilized include:

Layout: Use of display: flex and justify-content: center (as seen in the login files) for precise alignment and centering of form containers.

Aesthetics: Implementation of modern effects such as box shadows (box-shadow: 0 15px 35px rgba(0,0,0,0.1...) and CSS transitions for interactive elements like the login buttons (transition: transform .2s;, transform: translateY(-1px); on hover).

Client-Side Scripting (JavaScript): JavaScript is used for enhancing user experience by implementing client-side form validation before submission. This reduces unnecessary server load and provides immediate feedback to the user on input errors (e.g., confirming password match during registration).

5.2.2 Backend Integration (PHP)

PHP is the bridge that connects the frontend interface to the business logic and database.

Form Handling and Routing: All forms use the POST method and target the central login.php script. PHP then processes the form data using the $_POST superglobal and uses the hidden redirect field to route the user appropriately after authentication.

Dynamic Content Generation: PHP scripts are responsible for fetching data from MySQL and embedding it directly into the HTML markup. This is critical for:

Displaying real-time information, such as the list of available food items or the donor's tracked donation statuses.

Rendering System Messages (success, warning, or error messages) fetched via URL parameters (<?php if (isset($_GET['error'])) { ... } ?>), using the htmlspecialchars() function to sanitize output and prevent Cross-Site Scripting (XSS) vulnerabilities.

Security Features: Beyond session management (session_start(), session_destroy()), the backend implementation is responsible for ensuring security by performing input sanitization and utilizing prepared statements (inferred) when interacting with MySQL to defend against common vulnerabilities like SQL Injection.

CONCLUSION

The development of the Food Donation and Waste Management System (FoodShare) successfully addressed the stated problem of logistical inefficiency and decentralized information within the local food distribution chain. By leveraging modern web technologies, the project delivered a robust, secure, and transparent platform that effectively bridges the gap between surplus food donors and organizations in need.

REFERENCES

[1] Sanchez L. (2023). *Web Programming with HTML, CSS, Bootstrap, JavaScript, React.JS, PHP, and MySQL*, (Fourth Edition). IngramSpark. ISBN-13: 978-1088239872.

[2] Kogent Learning Solutions Inc. (2009). *Web Technologies: HTML, JAVASCRIPT, PHP, JAVA, JSP, ASP.NET, XML and Ajax, Black Book*. Dreamtech Press. ISBN-13: 978-8177229974.

[3] Benaloh, J. (2006). *Simple verifiable elections*. In *USENIX/ACCURATE Electronic Voting Technology Workshop (EVT)*.