Food Waste Management System - Complete Documentation

Table of Contents

1. Project Overview
2. System Architecture
3. Features & Modules
4. Technology Stack
5. Database Schema
6. User Flow
7. Module Details

---

1. Project Overview

Purpose

The Food Waste Management System is a comprehensive web application designed to reduce food waste through multiple approaches: donation sharing, expiry tracking, recipe suggestions, and composting guidance.

Objectives

- Reduce Food Waste: Help users manage food before it expires
- Facilitate Food Donation: Connect food donors with receivers
- Promote Sustainability: Provide composting and recipe alternatives
- Track Food Expiry: Alert users about expiring food items

Target Users

- Individual households
- Community organizations
- NGOs and food banks
- Environmentally conscious individuals

---

2. System Architecture

Architecture Type

Hybrid Architecture with Django Backend and React Frontend (Landing Page) + Static HTML pages for feature modules

Components

```
┌─────────────────────────────────────────────────────────────┐
            │           User Interface Layer                │
│ ┌──────────────────┐  ┌────────────────────────────┐  │
      │  │  React Frontend │ │  Static HTML/CSS/JS     │   │
      │  │  (Landing Page) │ │  (Feature Modules)      │   │
      │  │  Port: 5173     │ │  - Recipe Finder        │   │
   │  └─────────┬─────────┘ │  - Compost Ideas        │   │
      │      │      └────────────────────────────┘   │
└────────────┼────────────────────────────────────────────────┘
                              │
┌────────────▼────────────────────────────────────────────────┐
      │     Application Layer (Django Backend)          │
      │  - Authentication Views (Port: 8000)            │
        │  - Food Donation Views                       │
       │  - Food Expiry Tracker Views                   │
└────────────┬────────────────────────────────────────────────┘
                              │
┌────────────▼────────────────────────────────────────────────┐
            │      Business Logic Layer                     │
       │  (Models, Forms, Email Services)                 │
└────────────┬────────────────────────────────────────────────┘
                              │
┌────────────▼────────────────────────────────────────────────┐
            │      Data Access Layer                        │
         │      (Django ORM + Database)                   │
```

3. Features & Modules

3.1 Authentication System

- User Registration
- User Login
- User Logout
- Session Management

3.2 Food Donation Module

Use Cases:

1. Create Donation
2. View Available Listings
3. Request Food Donation
4. View My Donations
5. Accept/Reject Requests
6. View My Requests

Key Features:

- Donors can list available food items
- Receivers can browse and request donations
- Email notifications for all actions
- Status tracking (available, requested, accepted, completed)
- Soft delete for donations

3.3 Food Expiry Tracker Module

Use Cases:

1. Add Food Item
2. Edit Food Item
3. View Food Items
4. Delete Food Item

5. **Receive Expiry Notification**

Key Features:

- **Track food items with expiry dates**
- **Automatic status calculation (fresh, expiring soon, expired)**
- **Visual dashboard with statistics**
- **Categorization of food items**
- **Expiry notifications (6 days, 2 days, expired)**

3.4 Recipe Finder Module

Key Features:

- **Search recipes by available ingredients**
- **Reduce waste by using leftover ingredients**
- **Uses local recipes.json database with Indian recipes**
- **User-friendly search interface**
- **Comma-separated ingredient input**
- **Filters recipes based on available ingredients**
- **Displays recipe image, cooking time, cuisine type**
- **Shows complete ingredient list and instructions**

Recipe Database:

- **Over 1500+ Indian recipes**
- **Includes recipe name, ingredients, instructions**
- **Cooking time and cuisine information**
- **Recipe images and URLs for reference**

3.5 Compost Ideas Module

Key Features:

- **AI-powered composting suggestions using Google Gemini**
- **Get composting advice for any food or waste item**
- **Detailed, step-by-step instructions**
- **Practical tips for beginners**

- **Educational content about composting methods**
- **Information cards about different composting techniques:**
    - **Kitchen Compost**
    - **Yard Waste Compost**
    - **Vermicomposting**
    - **Bokashi Compost**
    - **Green Manure**
    - **Compost Tea**

Technical Implementation:

- **Express.js microservice (Port 3000)**
- **Google Gemini API integration**
- **Real-time AI-generated responses**
- **CORS-enabled for frontend communication**

---

4. Technology Stack

Backend

- **Framework: Django 4.x**
- **Language: Python 3.x**
- **ORM: Django ORM**
- **Authentication: Django Auth**
- **Port: 8000 (default)**

Compost Ideas Microservice:

- **Framework: Express.js (Node.js)**
- **Port: 3000**
- **AI Integration: Google Gemini API**
- **Purpose: Generate AI-powered composting suggestions**

Frontend

- **Landing Page: React.js (Port: 5173)**

- **Feature Modules: HTML5, CSS3, JavaScript (Static files)**
- **Icons: Font Awesome**
- **Styling: Custom CSS**

Data Storage

- **Backend Database: SQLite (development) / PostgreSQL (production)**
- **Recipe Data: recipes.json (local JSON file with Indian recipes)**
- **Static Assets: CSS, JavaScript, images**

External Services

- **Google Gemini API: AI-powered compost suggestions via Express.js microservice**

Communication

- **Backend → Frontend: After successful login, Django redirects to React app (http://localhost:5173/index.html)**
- **Frontend → Backend: React landing page provides navigation links to Django-served feature pages**
- **Recipe Finder: JavaScript fetches and filters data from local recipes.json file**
- **Compost Ideas: Frontend sends POST request to Express server (localhost:3000) → Express calls Gemini API → Returns AI-generated suggestions**

Database

- **Development: SQLite (default)**
- **Production: PostgreSQL/MySQL (recommended)**

Email Service

- **Django Email Backend: SMTP configuration**
- **HTML Email Templates: Custom styled emails**

Additional Tools

- **Static Files: CSS, JavaScript**
- **Media Files: User uploads (if any)**

---

5. Database Schema

5.1 User Model (Django Built-in)

```
User
 ├── id (PK)
 ├── username (email)
 ├── first_name
 ├── last_name
 ├── email
 ├── password (hashed)
 ├── is_active
└── date_joined
```

5.2 Donation Model

```
Donation
 ├── id (PK)
 ├── donor (FK → User)
 ├── full_name
 ├── contact
 ├── address
 ├── item_name
 ├── food_type (vegetables/fruits/cooked/others)
 ├── quantity
 ├── instructions
 ├── pickup_datetime
 ├── drop_location
 ├── consent
 ├── status (available/requested/accepted/completed)
 ├── is_deleted_by_donor
└── created_at
```

5.3 DonationRequest Model

DonationRequest

├── id (PK)

├── donation (FK → Donation)

├── receiver (FK → User)

├── receiver_name

├── receiver_contact

├── message

├── status (pending/accepted/rejected)

└── created_at

5.4 FoodItem Model

FoodItem

├── id (PK)

├── user (FK → User)

├── item_name

├── expiry_date

├── quantity

├── category

├── notes

├── created_at

├── updated_at

├── status (fresh/expiring_soon/very_close_to_expiry/expired)

├── notified_6 (boolean)

├── notified_2 (boolean)

└── notified_expired (boolean)

Relationships

- **User ↔ Donation: One-to-Many (One user can create many donations)**
- **User ↔ DonationRequest: One-to-Many (One user can make many requests)**

- **Donation ↔ DonationRequest: One-to-Many (One donation can have many requests)**
- **User ↔ FoodItem: One-to-Many (One user can track many food items)**

---

6. User Flow

Main Application Flow

1. User accesses Django Login page (localhost:8000/login/)

↓

2. User enters credentials

↓

3. Django authenticates and redirects to React Landing Page

→ http://localhost:5173/index.html

↓

4. React Landing Page displays 4 feature cards:

```
 ├── Food Donation (Django view)
 ├── Food Expiry Tracker (Django view)
 ├── Recipe Finder (Static HTML - localhost:5173)
 └── Compost Ideas (Static HTML - localhost:5173)
```

↓

5. User clicks a feature card

↓

6. Navigates to respective module:

- Food Donation/Tracker → Django backend (localhost:8000)

- Recipe Finder/Compost → Static pages (localhost:5173)

Cross-Origin Setup

- **Django backend runs on localhost:8000**
- **React frontend runs on localhost:5173**
- **After login: Django redirects to http://localhost:5173/index.html**

- Feature modules navigate back to Django or stay on static pages

Food Donation Flow

Donor Flow:

1. Create Donation → Fill form → Submit

2. View My Donations → See status

3. View Requests → Accept/Reject

Receiver Flow:

1. View Available Listings → Browse donations

2. Request Donation → Submit request

3. View My Requests → Track status

4. Receive acceptance email → Get donor details

Food Expiry Tracker Flow

1. Access Dashboard → View all items with status

2. Add Item → Enter details with expiry date

3. System auto-calculates status

4. View statistics (fresh/expiring/expired counts)

5. Edit/Delete items as needed

6. Receive email notifications (automated)

---

7. Module Details

7.1 Food Donation Module

Donor Functionalities

Create Donation:

- **Form fields: Name, contact, address, item name, food type, quantity, pickup time, drop location**
- **Validation for required fields**
- **Consent checkbox for terms**

- **Initial status: "available"**

Manage Donations:

- **View all created donations**
- **See request status**
- **Accept/reject incoming requests**
- **Delete donations (soft delete)**

Accept Request:

- **Marks request as accepted**
- **Updates donation status to "accepted"**
- **Auto-rejects other pending requests**
- **Sends emails to:**
  - **Accepted receiver (with donor details)**
  - **Donor (with receiver details)**
  - **Rejected receivers (notification)**

Reject Request:

- **Marks request as rejected**
- **If no accepted requests exist, donation becomes "available" again**
- **Sends rejection email to receiver**

Receiver Functionalities

Browse Listings:

- **View all available donations**
- **Filter excludes own donations**
- **See donation details**

Request Donation:

- **Submit request with contact details**
- **Validation: Cannot request own donation**
- **Validation: Cannot request same donation twice**
- **Email sent to donor about new request**

- **Donation status changes to "requested"**

Track Requests:

- **View pending requests**
- **View accepted requests (with donor details)**
- **View rejected requests**
- **Delete own requests**

Email Notifications

1. **New Request: Sent to donor when receiver requests**
2. **Request Accepted: Sent to receiver with donor contact**
3. **Request Rejected: Sent to receiver**
4. **Receiver Details: Sent to donor after acceptance**
5. **Donation Deleted: Sent to pending receivers**

7.2 Food Expiry Tracker Module

Dashboard

- **Displays all food items ordered by expiry date**
- **Shows statistics:**
  - **Total items count**
  - **Fresh items count**
  - **Expiring soon count**
  - **Expired items count**
- **Color-coded status indicators**

Add Food Item

- **Form fields: Item name, category, expiry date, quantity, notes**
- **Automatic status calculation on save**
- **Status determined by days until expiry:**
  - **Fresh: > 6 days**
  - **Expiring Soon: 4-6 days**

    - **Very Close to Expiry: 1-2 days**
    - **Expired: < 0 days**

Edit Food Item

- **Update any field including expiry date**
- **Automatic status recalculation**
- **Maintains notification flags**

Delete Food Item

- **Permanent deletion from database**
- **Confirmation required**

Notification System (Designed)

Note: Model supports notifications, but cron job needs implementation

- **6-day notification: First warning**
- **2-day notification: Urgent warning**
- **Expiry notification: Item has expired**
- **Flags prevent duplicate notifications (notified_6, notified_2, notified_expired)**

Implementation Required:

- **Django management command or Celery task**
- **Scheduled to run daily**
- **Query items and send emails based on expiry thresholds**

7.3 Recipe Finder Module

Functionality

- **Input: Comma-separated ingredients (e.g., "tomato, onion, rice")**
- **Process: Search local recipes.json database for matching recipes**
- **Output: Recipe cards with images, cooking time, and complete instructions**
- **Purpose: Use leftover ingredients before they expire**

Data Source

recipes.json Structure:

{

"TranslatedRecipeName": "Coriander and Coconut Chutney Recipe",

"TranslatedIngredients": "1 green chili, salt, 1 tablespoon lemon juice...",

"TotalTimeInMins": 20,

"Cuisine": "Indian",

"TranslatedInstructions": "To make coriander and coconut chutney...",

"URL": "https://www.archanaskitchen.com/...",

"Cleaned-Ingredients": "ginger,green chilli,coconut,saltlemon,coriander",

"image-url": "https://www.archanaskitchen.com/images/...",

"Ingredient-count": 6

}

Database Information:

- **Source: Archana's Kitchen (Indian recipes) [Dataset from Kaggle]**
- **Size: 1500+ recipes**
- **Cuisine Focus: Primarily Indian cuisine**
- **Data Fields: Recipe name, ingredients, instructions, cooking time, cuisine type, image, URL**

Search Logic

- **Takes user input ingredients (comma-separated)**
- **Searches "Cleaned-Ingredients" field in JSON**
- **Matches recipes containing any of the input ingredients**
- **Returns recipe cards with:**
  - **Recipe name**
  - **Recipe image**
  - **Total cooking time**
  - **Cuisine type**
  - **Complete ingredient list**
  - **Step-by-step instructions**

   - **Link to original recipe**

User Interface

- **Hero section with "Get Started" call-to-action**
- **Search bar for ingredient input (comma-separated)**
- **Results display as recipe cards with images**
- **"Back Home" button for navigation to React landing page**
- **Responsive grid layout for recipe cards**

Benefits

- **Offline functionality (no API dependency)**
- **Fast search performance**
- **Extensive Indian recipe database**
- **Visual recipe cards with images**
- **No rate limits or API costs**
- **Direct links to original recipes for more details**

7.4 Compost Ideas Module

Functionality

- **Input: Any food waste item (e.g., "banana peels", "expired oats", "cooked rice")**
- **Process: Send request to Express.js server → Call Google Gemini API**
- **Output: AI-generated, practical composting advice**
- **Purpose: Help users properly compost different types of waste**

Technical Architecture

```
Frontend (HTML/JS)
  ↓ POST /compost-ideas
Express.js Server (Port 3000)
  ↓ API Call
Google Gemini API
  ↓ AI Response
```

Express.js Server

↓ JSON Response

Frontend Display

Express.js Microservice (server.js)

Endpoint: POST http://localhost:3000/compost-ideas

Request Body:

```
{
  "item": "banana peels"
}
```

Response Processing:

1. **Validates item input**
2. **Constructs detailed prompt for Gemini AI**
3. **Calls Gemini API with environment variable API key**
4. **Returns structured composting advice**

Prompt Engineering:

- **Requests detailed, step-by-step advice**
- **Asks for method name, 2-4 simple steps, and 1 tip**
- **Optimized for beginner-friendly, actionable content**
- **Avoids long paragraphs for better readability**

AI Response Format

Gemini provides:

- **Composting Method: Name of the technique**
- **Steps: 2-4 simple, actionable steps**
- **Tip: One practical tip for success**
- **Beginner-friendly language**
- **Specific to the input item**

Educational Content

Six static information cards about composting methods:

1. **Kitchen Compost**
    - **Use fruit/vegetable scraps, coffee grounds, eggshells**
    - **Link: EPA Composting Guide**
2. **Yard Waste Compost**
    - **Leaves, grass clippings, small branches**
    - **Link: NC State Extension Guide**
3. **Vermicomposting**
    - **Worm-based composting for premium castings**
    - **Link: Microbiology Notes**
4. **Bokashi Compost**
    - **Fermentation method for all kitchen waste**
    - **Link: Planet Natural Guide**
5. **Green Manure**
    - **Cover crops for soil enrichment**
    - **Link: Kisan Sabha Blog**
6. **Compost Tea**
    - **Liquid extract for plant nutrition**
    - **Link: Old World Garden Farms**

Benefits Highlighted

- **Reduce Waste: Cut household waste by up to 50%**
- **Nourish Soil: Natural nutrient enrichment**
- **Save Money: Less fertilizer and waste bags needed**
- **Fight Climate Change: Reduce methane emissions from landfills**

User Interface

- **Hero section with nature-themed design**
- **Input field for food/waste item**
- **"Get Compost Ideas" button**
- **Output area displaying AI-generated advice**

- **Educational cards section below**
- **"Back Home" button for navigation**

Error Handling

- **Validates item input (required field)**
- **Handles API failures gracefully**
- **CORS configuration for cross-origin requests**
- **Console error logging for debugging**

Environment Configuration

Required: .env file with:

```
GEMINI_API_KEY=your_gemini_api_key_here
```

Dependencies:

- **express**
- **cors**
- **node-fetch**
- **dotenv**
