1 System Components

The Food Waste Redistribution system manages the following core entities:

- **Users:** Donors, NGOs, volunteers, and admins with various roles.
- **Food Listings:** Food items offered for donation by donors.
- **Requests:** NGOs/volunteers requesting available food listings.
- **Assignments:** Tracking volunteer/NGO assignments for pickup and delivery.
- **Pickup Locations:** Donor pickup points in urban areas.
- **Food Categories:** Standardized food type classifications.
- **Feedback:** Rating system between users (donors, NGOs, volunteers).

2 Unnormalized Form (0NF)

In the Unnormalized Form, all food redistribution data is stored in a single table (or "flat file") with repeating groups and non-atomic attributes. This results in significant redundancy and data inconsistency.

2.1 0NF Table Structure Issues

Attribute	**Issue**
Core Listing Information	
listing_id	Atomic
donor_name	Non-Atomic / Redundant (Repeats for every listing)
role	Non-Atomic (ENUM stored as text, repeated)
status	Non-Atomic (ENUM stored as text, repeated)
Repeating Groups	
food_type	Repeating ["rice", "dal", "bread"...]
request_list	Repeating [R01, R02, R03...]
pickup_locations	Repeating [Loc1, Loc2...]

Table 1: UNF Table – Issues with Repeating Groups

2.2 Key Violations in UNF

Violation Type	Example	Impact
Non-Atomic Attributes	role stored as text ENUM (donor, ngo, volunteer, admin)	Data retrieval complexity
Repeating Groups	Multiple requests per single listing row	Difficult to query specific requests
Redundancy	Donor info repeated for every listing	Storage waste
Anomalies	Deleting a listing deletes donor info	Data inconsistency

Table 2: Violations Identified in UNF

3 First Normal Form (1NF)

To achieve 1NF, we eliminate repeating groups by splitting multi-valued fields into separate records and ensure all attributes are atomic.

3.1 1NF: Users Table (Atomic)

Column Name	Data Type	Constraint
user_id	INT	Primary Key
name	VARCHAR(100)	NOT NULL
phone	VARCHAR(20)	NOT NULL
email	VARCHAR(100)	UNIQUE
role	VARCHAR(50)	NOT NULL (donor/ngo/volunteer/admin)
address	TEXT	NOT NULL
city	VARCHAR(100)	NOT NULL

Table 3: 1NF: Users Table

3.2 1NF: Food_Listings Table (Atomic)

Column Name	Data Type	Constraint
listing_id	INT	Primary Key
user_id	INT	FK → Users
food_type	VARCHAR(100)	NOT NULL
quantity	INT	NOT NULL
prepared_time	DATETIME	NOT NULL
expiry_time	DATETIME	NOT NULL
status	VARCHAR(50)	NOT NULL (pending/assigned/completed/expired)
location	TEXT	NOT NULL

Table 4: 1NF: Food_Listings Table

3.3 1NF: Requests Table (Decomposed)

Column Name	Data Type	Constraint
request_id	INT	Primary Key
listing_id	INT	FK → Food_Listings
requested_by	INT	FK → Users
request_time	DATETIME	NOT NULL
approval_status	VARCHAR(50)	NOT NULL (pending/approved/rejected)

Table 5: 1NF: Requests Table

4 Second Normal Form (2NF)

2NF eliminates partial dependencies where non-key attributes depend only on part of a composite primary key. In our 1NF tables, there are no composite primary keys causing partial dependencies, but we identify redundancy in ENUM-like text fields.

4.1 2NF: Partial Dependency Analysis

In the *Users* table, the role field stores repeated text values.

Column	Table	Issue	Status
role	Users	Text repeated for each user	**Redundancy**
food_type	Food_Listings	Non-standardized text values	**Redundancy**
status	Food_Listings	ENUM as repeated text	**Redundancy**
approval_status	Requests	ENUM as repeated text	**Redundancy**

Table 6: 2NF: Redundancy Analysis

4.2 2NF Resolution: Creating Lookup Tables

We create lookup tables for standardized values.

Column Name	Type	Constraint	Purpose
role_id	INT	PK	Unique identifier
role_name	VARCHAR(50)	UNIQUE, NOT NULL	donor, ngo, volunteer, admin

Table 7: 2NF: User_Roles Table

Column Name	Type	Constraint	Purpose
category_id	INT	PK	Unique identifier
category_name	VARCHAR(100)	UNIQUE, NOT NULL	Standardized food types

Table 8: 2NF: Food_Categories Table

Column Name	Type	Constraint	Purpose
status_id	INT	PK	Unique identifier
status_name	VARCHAR(50)	UNIQUE, NOT NULL	pending, assigned, completed, expired

Table 9: 2NF: Listing_Status Table

5 Third Normal Form (3NF)

3NF removes transitive dependencies where non-key attributes depend on other non-key attributes.

5.1 3NF: Transitive Dependency Analysis

In the *Food_Listings* table:

- **Dependency:** listing_id → user_id → donor_address
- **Issue:** *location* in Food_Listings may duplicate donor's address from Users table.

In the *Requests* table:

- **Dependency:** request_id → approval_status (text) → status_meaning
- **Issue:** Status text values should reference a lookup table.

5.2 3NF Resolution: Creating Status Lookup Tables

Column Name	Data Type	Constraint	Note
request_status_id	INT	PK	Unique identifier
status_name	VARCHAR(50)	UNIQUE, NOT NULL	pending, approved, rejected

Table 10: 3NF: Request_Status Table

Column Name	Data Type	Constraint	Note
assignment_status_id	INT	PK	Unique identifier
status_name	VARCHAR(50)	UNIQUE, NOT NULL	assigned, picked_up, delivered, c

Table 11: 3NF: Assignment_Status Table

5.3 3NF Resolution: Pickup_Locations Table

Separating pickup locations from food listings eliminates transitive dependency on donor address.

Column Name	Data Type	Constraint	Note
location_id	INT	PK	Unique identifier
donor_id	INT	FK → Users	Donor reference
address	TEXT	NOT NULL	Pickup address
landmark	VARCHAR(200)	—	Nearby landmark
latitude	DECIMAL(10,8)	—	GPS coordinate
longitude	DECIMAL(11,8)	—	GPS coordinate

Table 12: 3NF: Pickup_Locations Table

6 Final Normalized Database Schema

The complete schema after applying 1NF, 2NF, and 3NF.

6.1 Lookup/Reference Tables

Table	PK	Purpose
User_Roles	role_id	Stores role types (donor, ngo, volunteer, admin)
Food_Categories	category_id	Standardized food type classifications
Listing_Status	status_id	Listing statuses (pending, assigned, completed, expired)
Request_Status	request_status_id	Request statuses (pending, approved, rejected)
Assignment_Status	assignment_status_id	Assignment statuses (assigned, picked_up, delivered, cancelled)

Table 13: Final Schema: Lookup Tables

6.2 Core Entity Tables

Table	PK	Foreign Keys
Users	user_id	role_id → User_Roles
Pickup_Locations	location_id	donor_id → Users
Food_Listings	listing_id	user_id → Users category_id → Food_Categories status_id → Listing_Status pickup_location_id → Pickup_Locations

Table 14: Final Schema: Master Tables

6.3 Transactional Tables

Table	PK	Foreign Keys
Requests	request_id	listing_id → Food_Listings requested_by → Users request_status_id → Request_Status
Assignments	assignment_id	request_id → Requests assigned_to → Users assignment_status_id → Assignment_Status
Feedback	feedback_id	from_user → Users to_user → Users listing_id → Food_Listings

Table 15: Final Schema: Transactional Tables

6.4 Complete Table Definitions

6.4.1 Users Table

```
Users (user_id PK, name, phone, email, role_id FK, address, city)
```

6.4.2 Food_Listings Table

```
Food_Listings (listing_id PK, user_id FK, category_id FK, quantity,
               prepared_time, expiry_time, status_id FK,
               pickup_location_id FK)
```

6.4.3 Requests Table

```
Requests (request_id PK, listing_id FK, requested_by FK,
          request_time, request_status_id FK)
```

6.4.4 Assignments Table

```
Assignments (assignment_id PK, request_id FK, assigned_to FK,
             assigned_time, pickup_time, delivery_time,
             assignment_status_id FK)
```

6.4.5 Feedback Table

```
Feedback (feedback_id PK, from_user FK, to_user FK, listing_id FK,
          rating, comments, date)
```

7 Relationships Overview

Relationship	Cardinality	Description
Users → Food_Listings	1 : M	One donor creates many food listings
Food_Listings → Requests	1 : M	One listing can receive many requests
Requests → Assignments	1 : 1	One approved request leads to one assignment
Users → Assignments	1 : M	One volunteer/NGO handles many assignments
Users → Pickup_Locations	1 : M	Donors may have multiple pickup points
Food_Categories → Food_Listings	1 : M	Each listing belongs to one category
Users ↔ Users (Feedback)	M : M	Users rate each other via Feedback

Table 16: Entity Relationships Summary