Local Food Wastage Management System

1. Project Overview

Project Title: Local Food Wastage Management System
Project Description: The Local Food Wastage Management System aims to reduce food wastage by connecting food providers (like restaurants and food suppliers) with receivers (such as food banks, shelters, and community organizations). The platform facilitates food donations, tracks claims, and provides data insights to improve food management and reduce waste in the local community.

2. Technologies Used

Technologies Used:
- Python: For backend development and data processing.
- SQL: For database management and queries.
- Streamlit: For creating the web-based user interface.
- Pandas: For data manipulation and analysis.
- CSV: For data storage and management (with the following CSV files used):
  - providers_data.csv
  - receivers_data.csv
  - food_listings_data.csv
  - claims_data.csv

3. System Features

- CRUD Operations: Create, Read, Update, and Delete operations for managing food providers, receivers, food listings, and claims.
- Data Visualization: Graphs and charts displaying trends in food waste, food donations, and claims.
- Food Waste Tracking: Track and manage the quantities of food available for donation and food received by different organizations.
- Claim Management: Allow receivers to claim food from providers, and track the status of claims.
- Query Functionality: SQL-like queries to extract insights from the data.

4. Functional Requirements

1. User Authentication (Optional): Secure login for both providers and receivers.
2. Food Listings: Providers can list available food for donation.
3. Claims System: Receivers can claim food from available listings.

4. Data Analysis: Ability to analyze food waste trends, provider statistics, receiver activities, and more through SQL-like queries.
5. Data Visualization: Generate charts and graphs for food waste management statistics.
6. Search and Filter: Ability to search and filter food listings based on criteria like food type, provider location, or claim status.

5. Non-Functional Requirements

- Scalability: The system should be scalable to handle large datasets of providers, receivers, and food listings.
- Usability: User-friendly interface with intuitive navigation for both providers and receivers.
- Performance: The system should efficiently handle CRUD operations and data visualization without significant delays.

6. Database Schema

providers_data.csv:
- Provider_ID, Name, Location, Contact_Info, Food_Type, Available_Quantity

receivers_data.csv:
- Receiver_ID, Name, Organization_Type, Location, Contact_Info

food_listings_data.csv:
- Listing_ID, Provider_ID, Food_Type, Available_Quantity, Expiration_Date, Status

claims_data.csv:
- Claim_ID, Receiver_ID, Listing_ID, Claimed_Quantity, Claim_Date, Status

7. System Architecture

- Frontend: Built using Streamlit for easy deployment of the interface, allowing users to interact with the system via web pages.
- Backend: The system's backend logic is powered by Python, handling the core functionalities like data manipulation and querying.
- Data Storage: All data is stored in CSV files for easy handling and persistence. This makes the system lightweight and simple to deploy without a complex database setup.

8. User Interface

- Homepage: A dashboard displaying recent food listings, claims, and the overall system status.

- Food Listings Page: Providers can add or update food listings here.
- Claims Page: Receivers can browse available listings and submit claims.
- Data Insights Page: Visualizations and reports based on SQL-like queries (e.g., total food donated, claims by region, etc.).

9. SQL Queries (Example)

1. Query to Get Total Food Donations by Provider:
SQL: SELECT Provider_ID, SUM(Available_Quantity) FROM food_listings_data GROUP BY Provider_ID;

2. Query to Get Pending Claims:
SQL: SELECT * FROM claims_data WHERE Status = 'Pending';

3. Query to Get Food Listings by Type:
SQL: SELECT * FROM food_listings_data WHERE Food_Type = 'Vegetable';

10. Challenges Faced

- Data Integrity: Ensuring the accuracy of food quantities and claims in a real-world scenario could be challenging.
- User Adoption: Encouraging food providers and receivers to actively use the platform.
- Scalability: Handling a large number of records efficiently in CSV files rather than a relational database.

11. Future Enhancements

- User Authentication and Authorization: Adding user login systems with roles (provider, receiver).
- Advanced Reporting: Adding more complex reporting and predictive analysis features.
- Mobile Compatibility: Developing a mobile-friendly version of the application for easier access.
- Database Integration: Migrating from CSV files to a more robust relational database system like MySQL or PostgreSQL.

12. Conclusion

The Local Food Wastage Management System aims to address food waste in the community by enabling seamless interactions between food providers and receivers. Through this platform, we hope to contribute to sustainable food distribution practices and make a positive impact on local communities.