Food Wastage Management System in College Hostels

# Problem Statement (Abstract)

Food wastage in college hostels leads to resource loss, environmental impact, and cost inefficiencies. There is no proper system to track, analyze, and reduce daily food wastage. This project aims to provide a data-driven solution to monitor and manage food wastage, helping hostel authorities take informed decisions and promote awareness among students.

# Index

1. Abstract

2. Description

3. Data and Output Overview

4. Solution Plan

5. Design (Diagrams)

6. Implementation

7. Code & Explanation

8. Output Screenshots

9. Closure

10. Bibliography

# Description (Detailed)

This project focuses on developing a Flask-based web application to record daily food details and analyze wastage trends. Students can provide feedback on meals, while the admin dashboard visually shows food utilization, alerts for high wastage, meal-wise wastage, and suggestions for improvement.

# Data and Output Overview

Input Data:

- Date, Day

- Meal Type

- Prepared Quantity

- Consumed Quantity

- Students Served

- Plates Wasted

- Reused or Donated Quantity

Output:

- Graphs (trend, plate waste, pie charts)

- Text-based conclusions

- Alerts and Weekly summaries

Purpose:

- Help reduce food waste, analyze consumption patterns, and plan better

Outcome:

- Visual and textual insights for hostel managers

Benefits:

- Saves food, cost, and improves accountability

# Solution Plan

1. Build Flask backend

2. Store data using SQLite

3. Analyze using Pandas & Matplotlib

4. Generate visuals & reports

5. Display dashboard

# Design (Diagrams)

HTML Form → Flask → SQLite → Analysis (Pandas, Matplotlib) → Charts + Dashboard

# 

# Implementation

Folder Structure:

|- app.py

|-init\_db.py

|- analysis/analysis.py

|- templates/

|-dashboard.html

|-add\_data.html

|-feedback.html

|- static/

|-plots\

|-style.css

|-\*.txt

|- db/food\_data.db

# Code & Explanation

app.py code

from flask import Flask, render\_template, request, redirect

import sqlite3

import pandas as pd

from analysis.analysis import run\_all\_analysis, weekly\_insights

app = Flask(\_\_name\_\_)

# Home Dashboard

@app.route('/')

def dashboard():

    run\_all\_analysis()  # Generate charts and summaries

    # Helper to read text files

    def read\_file(path, default="Not available"):

        try:

            with open(path, 'r', encoding='utf-8') as f:

                return f.read()

        except FileNotFoundError:

            return default

    # Read all summaries

    alerts\_text = read\_file('static/alerts.txt')

    trend\_conclusion = read\_file('static/trend\_summary.txt')

    plate\_message = read\_file('static/plate\_alert.txt')

    reuse\_advice = read\_file('static/reuse\_advice.txt')

    meal\_type\_conclusion = read\_file('static/meal\_advice.txt')

    day\_meal\_conclusion = read\_file('static/day\_meal\_conclusion.txt')  # ✅

    # Database connection

    conn = sqlite3.connect('db/food\_data.db')

    cursor = conn.cursor()

    # Feedback

    cursor.execute("SELECT AVG(rating) FROM feedback")

    avg\_rating = cursor.fetchone()[0]

    avg\_rating = round(avg\_rating, 2) if avg\_rating else "No ratings yet"

    cursor.execute("SELECT date, comment FROM feedback ORDER BY id DESC LIMIT 5")

    recent\_comments = cursor.fetchall()

    # Format feedback dates (remove 00:00:00)

    formatted\_comments = []

    for date, comment in recent\_comments:

        try:

            date\_str = pd.to\_datetime(date).strftime('%Y-%m-%d')

        except:

            date\_str = str(date)

        formatted\_comments.append((date\_str, comment))

    # Weekly insights

    df = pd.read\_sql\_query("SELECT \* FROM food\_log", conn)

    conn.close()

    insights = weekly\_insights(df)

    return render\_template('dashboard.html',

                           alerts=alerts\_text,

                           avg\_rating=avg\_rating,

                           comments=formatted\_comments,

                           weekly\_waste=insights['weekly\_waste'],

                           weekly\_attendance=insights['weekly\_attendance'],

                           top\_meals=insights['top\_meals'],

                           trend\_conclusion=trend\_conclusion,

                           plate\_message=plate\_message,

                           reuse\_advice=reuse\_advice,

                           meal\_type\_conclusion=meal\_type\_conclusion,

                           day\_meal\_conclusion=day\_meal\_conclusion)  # ✅

# Add Data Form

@app.route('/add')

def add\_data():

    return render\_template('add\_data.html')

# Submit Food Data

@app.route('/submit', methods=['POST'])

def submit\_data():

    data = (

        request.form['date'],

        request.form['day'],

        request.form['meal\_type'],

        float(request.form['prepared']),

        float(request.form['consumed']),

        int(request.form['students']),

        int(request.form['plates']),

        float(request.form['donated'])

    )

    conn = sqlite3.connect('db/food\_data.db')

    c = conn.cursor()

    c.execute('INSERT INTO food\_log VALUES (NULL,?,?,?,?,?,?,?,?)', data)

    conn.commit()

    conn.close()

    return redirect('/')

# Feedback Form

@app.route('/feedback')

def feedback\_form():

    return render\_template('feedback.html')

# Submit Feedback

@app.route('/submit\_feedback', methods=['POST'])

def submit\_feedback():

    date = request.form['date']

    rating = int(request.form['rating'])

    comment = request.form['comment']

    conn = sqlite3.connect('db/food\_data.db')

    c = conn.cursor()

    c.execute('INSERT INTO feedback (date, rating, comment) VALUES (?, ?, ?)', (date, rating, comment))

    conn.commit()

    conn.close()

    return redirect('/')

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True)

import matplotlib

matplotlib.use('Agg')

import sqlite3

import pandas as pd

import matplotlib.pyplot as plt

import os

DB\_PATH = 'db/food\_data.db'

PLOT\_DIR = 'static/plots'

# 1. Daily Food Wastage Trend

def food\_wastage\_trend(df):

    df['date'] = pd.to\_datetime(df['date'])

    df['wasted'] = df['prepared\_quantity'] - df['consumed\_quantity']

    last\_month\_df = df[df['date'] >= df['date'].max() - pd.Timedelta(days=30)]

    trend = last\_month\_df.groupby('date')['wasted'].sum()

    plt.figure(figsize=(8, 4))

    trend.index = trend.index.strftime('%Y-%m-%d')

    trend.plot(kind='line', marker='o', color='red')

    plt.title("Last 30 Days Food Wastage Trend")

    plt.xlabel("Date")

    plt.ylabel("Wasted (kg)")

    plt.xticks(rotation=45)

    plt.tight\_layout()

    plt.savefig(f"{PLOT\_DIR}/trend.png")

    plt.close()

    if not trend.empty:

        min\_date = trend.idxmin()

        max\_date = trend.idxmax()

        min\_val = trend.min()

        max\_val = trend.max()

        conclusion = f"📅 MINIMUM WASTE: {min\_date} → {min\_val:.2f} kg\n" \

                     f"📅 MAXIMUM WASTE: {max\_date} → {max\_val:.2f} kg\n"

        with open('static/trend\_summary.txt', 'w', encoding='utf-8') as f:

            f.write(conclusion)

# 2. Meal-wise & Day-wise Wastage

def meal\_day\_wastage(df):

    df['wasted'] = df['prepared\_quantity'] - df['consumed\_quantity']

    df['date'] = pd.to\_datetime(df['date'])

    last\_week = df[df['date'] >= df['date'].max() - pd.Timedelta(days=7)]

    pivot = last\_week.groupby(['day', 'meal\_type'])['wasted'].mean().unstack()

    pivot.plot(kind='bar', figsize=(8,5))

    plt.title("Avg Wastage by Day & Meal (Last 7 Days)")

    plt.ylabel("Wasted (kg)")

    plt.xticks(rotation=45)

    plt.tight\_layout()

    plt.savefig(f"{PLOT\_DIR}/day\_meal.png")

    plt.close()

    summary = last\_week.groupby(['day', 'meal\_type'])['wasted'].mean()

    if not summary.empty:

        max\_combination = summary.idxmax()

        max\_value = summary.max()

        conclusion = f"🔍 Highest avg wastage: {max\_combination[0]} - {max\_combination[1]} → {max\_value:.2f} kg."

    else:

        conclusion = "No sufficient data to evaluate meal-wise wastage trend."

    with open("static/day\_meal\_conclusion.txt", "w", encoding="utf-8") as f:

        f.write(conclusion)

# 3. Plate Waste Ratio

def plate\_waste\_ratio(df):

    df['date'] = pd.to\_datetime(df['date'])

    df['waste\_ratio'] = df['plates\_wasted'] / df['students\_served']

    recent\_df = df[df['date'] >= df['date'].max() - pd.Timedelta(days=7)]

    avg\_ratio = recent\_df.groupby('date')['waste\_ratio'].mean()

    plt.figure(figsize=(8,4))

    avg\_ratio.index = avg\_ratio.index.strftime('%Y-%m-%d')

    avg\_ratio.plot(kind='bar', color='orange')

    plt.title("Plate Waste Ratio (Plates Wasted / Students Served)")

    plt.ylabel("Waste Ratio")

    plt.xticks(rotation=45)

    plt.tight\_layout()

    plt.savefig(f"{PLOT\_DIR}/plate\_waste.png")

    plt.close()

    warning\_text = ""

    if any(avg\_ratio > 0.2):

        warning\_text = "⚠️ ALERT: Plate waste ratio exceeded 0.2 on some days! Please create awareness about not wasting food.\n\n"

        for date, ratio in avg\_ratio.items():

            if ratio > 0.2:

                warning\_text += f"{date}: Ratio = {ratio:.2f}\n"

    else:

        warning\_text = "✅ Plate waste ratio is under control for all recent days. Keep up the good work!"

    with open("static/plate\_alert.txt", "w", encoding="utf-8") as f:

        f.write(warning\_text)

# 4. Reused vs Wasted Food

def reused\_vs\_wasted(df):

    df['date'] = pd.to\_datetime(df['date'])

    df['wasted'] = df['prepared\_quantity'] - df['consumed\_quantity']

    recent\_df = df[df['date'] >= df['date'].max() - pd.Timedelta(days=30)]

    grouped = recent\_df.groupby('date').agg({

        'wasted': 'sum',

        'reused\_or\_donated': 'sum'

    }).reset\_index()

    plt.figure(figsize=(8, 4))

    bar\_width = 0.35

    index = range(len(grouped))

    plt.bar(index, grouped['wasted'], bar\_width, label='Wasted', color='red')

    plt.bar([i + bar\_width for i in index], grouped['reused\_or\_donated'], bar\_width, label='Reused/Donated', color='blue')

    plt.xlabel('Date')

    plt.ylabel('Quantity (kg)')

    plt.title('Reused vs Wasted Food (Last 30 Days)')

    plt.xticks([i + bar\_width / 2 for i in index], grouped['date'].dt.strftime('%Y-%m-%d'), rotation=45)

    plt.legend()

    plt.tight\_layout()

    plt.savefig(f"{PLOT\_DIR}/reuse\_vs\_waste.png")

    plt.close()

    advice = ""

    high\_waste\_days = grouped[grouped['wasted'] > 250]

    if not high\_waste\_days.empty:

        advice += "Conclusion: The following days had food wastage over 250 kg. It's better to donate more food on these days:\n"

        for \_, row in high\_waste\_days.iterrows():

            advice += f"{row['date'].strftime('%Y-%m-%d')} - Wasted: {row['wasted']} kg\n"

    with open("static/reuse\_advice.txt", "w", encoding="utf-8") as f:

        f.write(advice)

# 5. Predict Next Day Consumption

def predict\_consumption(df):

    df['date'] = pd.to\_datetime(df['date'])

    df = df.sort\_values('date')

    df['predicted'] = df['consumed\_quantity'].rolling(window=3).mean()

    plt.figure(figsize=(8, 4))

    plt.plot(df['date'].dt.strftime('%Y-%m-%d'), df['consumed\_quantity'], label='Predicted', marker='o')

    plt.plot(df['date'].dt.strftime('%Y-%m-%d'), df['predicted'], label='Actual', linestyle='--', color='green')

    plt.title("Food Consumption Prediction")

    plt.legend()

    plt.xticks(rotation=45)

    plt.tight\_layout()

    plt.savefig(f"{PLOT\_DIR}/prediction.png")

    plt.close()

# 6. Food Sharing Alerts + Worst Days

def food\_alerts(df):

    df['wasted'] = df['prepared\_quantity'] - df['consumed\_quantity']

    df['waste\_percent'] = (df['wasted'] / df['prepared\_quantity']) \* 100

    alerts = df[df['waste\_percent'] > 30].drop\_duplicates(subset=['date', 'meal\_type'])

    alert\_text = "FOOD SHARING ALERTS (Wastage > 30%):\n"

    for \_, row in alerts.iterrows():

        alert\_text += f"{row['date'].strftime('%Y-%m-%d')} ({row['meal\_type']}): Wasted {row['waste\_percent']:.1f}%\n"

    worst\_days = df.groupby('date')['waste\_percent'].mean().sort\_values(ascending=False).head(3)

    alert\_text += "\nTOP 3 WORST WASTAGE DAYS:\n"

    for date, percent in worst\_days.items():

        alert\_text += f"{date.strftime('%Y-%m-%d')}: Avg Wastage = {percent:.1f}%\n"

    with open('static/alerts.txt', 'w', encoding='utf-8') as f:

        f.write(alert\_text)

# 7. Pie Chart: Food Utilization

def food\_utilization\_pie(df):

    df['wasted'] = df['prepared\_quantity'] - df['consumed\_quantity']

    total\_consumed = df['consumed\_quantity'].sum()

    total\_reused = df['reused\_or\_donated'].sum()

    total\_wasted = df['wasted'].sum()

    plt.figure(figsize=(6,6))

    plt.pie([total\_consumed, total\_wasted, total\_reused],

            labels=['Consumed', 'Wasted', 'Reused'],

            autopct='%1.1f%%',

            colors=['#8BC34A', '#F44336', '#03A9F4'])

    plt.title("Total Food Utilization")

    plt.savefig(f"{PLOT\_DIR}/utilization\_pie.png")

    plt.close()

# 8. Pie Chart: Meal Type Wastage

def meal\_type\_wastage\_pie(df):

    df['wasted'] = df['prepared\_quantity'] - df['consumed\_quantity']

    totals = df.groupby('meal\_type')['wasted'].sum()

    plt.figure(figsize=(6,6))

    plt.pie(totals, labels=totals.index, autopct='%1.1f%%', startangle=140)

    plt.title("Meal Type Contribution to Wastage")

    plt.savefig(f"{PLOT\_DIR}/meal\_wastage\_pie.png")

    plt.close()

    most\_wasted\_meal = totals.idxmax()

    wasted\_kg = round(totals.max(), 2)

    advice = f"Conclusion: '{most\_wasted\_meal}' contributes the highest to food wastage with approximately {wasted\_kg} kg wasted. Special care should be taken to reduce wastage during this meal."

    with open("static/meal\_advice.txt", "w", encoding="utf-8") as f:

        f.write(advice)

# 9. Weekly Insights

def weekly\_insights(df):

    df['wasted'] = df['prepared\_quantity'] - df['consumed\_quantity']

    df['date'] = pd.to\_datetime(df['date'])

    recent\_df = df[df['date'] >= df['date'].max() - pd.Timedelta(days=7)]

    weekly\_waste = round(recent\_df['wasted'].sum(), 2)

    weekly\_attendance = round(recent\_df['students\_served'].mean(), 2)

    meal\_waste = recent\_df.groupby('meal\_type')['wasted'].sum()

    top\_meal = meal\_waste.idxmax() if not meal\_waste.empty else "N/A"

    return {

        'weekly\_waste': weekly\_waste,

        'weekly\_attendance': weekly\_attendance,

        'top\_meals': top\_meal

    }

# Master function to run all analysis

def run\_all\_analysis():

    if not os.path.exists(DB\_PATH):

        print("Database not found.")

        return

    conn = sqlite3.connect(DB\_PATH)

    df = pd.read\_sql\_query("SELECT \* FROM food\_log", conn)

    conn.close()

    if df.empty:

        print("No data available.")

        return

    food\_wastage\_trend(df)

    meal\_day\_wastage(df)

    plate\_waste\_ratio(df)

    reused\_vs\_wasted(df)

    predict\_consumption(df)

    food\_alerts(df)

    food\_utilization\_pie(df)

    meal\_type\_wastage\_pie(df)

    print(" All analysis graphs saved.")

dashboard.html

<!DOCTYPE html>

<html>

<head>

    <title>Food Wastage Dashboard</title>

    <link rel="stylesheet" href="/static/style.css">

</head>

<body>

    <h1>🍽️ Food Wastage Analysis Dashboard</h1>

    <p>

        <a href="/add">➕ Add Food Entry</a> |

        <a href="/feedback">🗣️ Give Feedback</a>

    </p>

    <hr>

    <!-- 🔔 Food Sharing Alerts -->

    <h2>📢 Food Sharing Alerts (Wastage > 30%) & Worst Days</h2>

    <pre style="background:#f8f8f8; padding:10px; border:1px solid #ccc;">

{{ alerts }}

    </pre>

    <hr>

    <!-- 📈 Graphical Insights -->

    <div class="chart-section">

        <h2>📊 Daily Food Wastage Trend</h2>

        <img src="{{ url\_for('static', filename='plots/trend.png') }}" width="600">

        <p><strong>Conclusion:</strong><br>{{ trend\_conclusion }}</p>

        <h2>📊 Avg Wastage by Day & Meal</h2>

        <img src="{{ url\_for('static', filename='plots/day\_meal.png') }}" width="600">

        <p><strong>Conclusion:</strong><br>{{ day\_meal\_conclusion }}</p>

        <h2>📊 Plate Waste Ratio</h2>

        <img src="{{ url\_for('static', filename='plots/plate\_waste.png') }}" width="600">

        <p><strong>Message:</strong><br>{{ plate\_message }}</p>

        <h2>📊 Reused vs Wasted Food</h2>

        <img src="{{ url\_for('static', filename='plots/reuse\_vs\_waste.png') }}" width="600">

        <p><strong>Advice:</strong><br>{{ reuse\_advice }}</p>

        <h2>📈 Predicted Consumption Trend</h2>

        <img src="{{ url\_for('static', filename='plots/prediction.png') }}" width="600">

    </div>

    <hr>

    <!-- 🥧 Pie Charts -->

    <h2>🥗 Total Food Utilization</h2>

    <img src="{{ url\_for('static', filename='plots/utilization\_pie.png') }}" width="400">

    <h2>🍱 Wastage by Meal Type</h2>

    <img src="{{ url\_for('static', filename='plots/meal\_wastage\_pie.png') }}" width="400">

    <p><strong>Observation:</strong><br>{{ meal\_type\_conclusion }}</p>

    <hr>

    <!-- 📆 Weekly Insights -->

    <h2>🗓️ Weekly Insights</h2>

    <p><strong>Total Weekly Wastage:</strong> {{ weekly\_waste }} kg</p>

    <p><strong>Average Weekly Attendance:</strong> {{ weekly\_attendance }} students</p>

    <p><strong>Most Wasted Meal:</strong> {{ top\_meals }}</p>

    <hr>

    <!-- 📝 Student Feedback -->

    <h2>📝 Student Feedback Summary</h2>

    <p><strong>Average Rating:</strong> {{ avg\_rating }}</p>

    <ul>

        {% for date, comment in comments %}

            {% if date and comment %}

                <li><strong>{{ date }}</strong>: {{ comment }}</li>

            {% endif %}

        {% endfor %}

    </ul>

</body>

</html>

# Output Screenshots

# 

# 

# 

# 

# 

# 

# 

# 

# 

# Closure

This project provides a working tool to monitor and reduce food wastage in hostels. It enables data entry, analysis, and action. The system is simple, scalable, and impactful.

# Bibliography

- Pandas Documentation

- Flask Docs

- Matplotlib Docs

- SQLite Basics

- OpenAI Prompting Assistance