CONTENT PAGE

1. COMPANY PROFILE

ARDENT (Ardent Computech Pvt. Ltd.), formerly known as Ardent Computech Private Limited, is an ISO

9001:2015 certified Software Development and Training Company based in India. Operating independently since

2003, the organization has recently undergone a strategic merger with ARDENT Technologies, enhancing its global

outreach and service offerings.

ARDENT Technologies

ARDENT Technologies delivers high-end IT services across the UK, USA, Canada, and India. Its core

competencies lie in the development of customized application software, encompassing end-to-end solutions

including system analysis, design, development, implementation, and training. The company also provides expert

consultancy and electronic security solutions. Its clientele spans educational institutions, entertainment companies,

resorts, theme parks, the service industry, telecom operators, media, and diverse business sectors.

ARDENT Collaborations

ARDENT Collaborations, the Research, Training, and Development division of ARDENT (Ardent Computech Pvt.

Ltd.), offers professional IT-enabled services and industrial training programs. These are tailored for freshers and

professionals from B.Tech, M.Tech, MBA, MCA, BCA, and MSc backgrounds. ARDENT (Ardent Computech Pvt.

Ltd.) provides Summer Training, Winter Training, and Industrial Training to eligible candidates. High-performing

students may qualify for stipends, scholarships, and additional benefits based on performance and mentor

recommendations.

Associations and Accreditations

ARDENT (Ardent Computech Pvt. Ltd.) is affiliated with the National Council of Vocational Training (NCVT)

under the Directorate General of Employment & Training (DGET), Ministry of Labour & Employment,

Government of India. The institution upholds strict quality standards under ISO 9001:2015 certification and is

dedicated to bridging the gap between academic knowledge and industry skills through innovative training

programs.

2. INTRODUCTION

In today's fast-paced world, the issue of food wastage has reached alarming levels, even as millions of people struggle with hunger and malnutrition. According to global studies, a significant portion of food produced is discarded due to surplus, improper storage, or distribution inefficiencies. Addressing this imbalance requires an organized, transparent, and technology-driven solution.

The NGOs & Waste Foods Management System aims to bridge the gap between food donors and those in need by creating a centralized, efficient, and reliable platform for food

collection, redistribution, and tracking. By collaborating with NGOs, restaurants, supermarkets, event organizers, and individual donors, the system ensures that excess food is directed to underprivileged communities rather than ending up in landfills.

Powered by the MERN stack, this platform offers a user-friendly interface for donors to list surplus food, volunteers to coordinate pick-up and delivery, and NGOs to manage inventory, beneficiaries, and reporting. Real-time notifications, geolocation-based tracking, and analytics features make the process more transparent and efficient.

By leveraging modern technology, the NGOs & Waste Foods Management System not only reduces food waste but also promotes community engagement, environmental sustainability, and social responsibility. This project stands as a step towards a future where no edible food is wasted, and every meal serves a purpose.

2A. OBJECTIVE

The primary objective of the NGOs & Waste Foods Management System is to minimize food wastage and ensure surplus edible food reaches individuals and communities in need through an efficient, transparent, and technology-driven process. The system is designed to serve the diverse requirements of donors, volunteers, NGOs, and beneficiaries by streamlining the food collection, storage, and distribution workflow.

This platform aims to bridge the gap between excess food availability and food scarcity by providing a centralized network where restaurants, event organizers, grocery stores, and individuals can donate surplus food while NGOs and volunteers coordinate timely delivery to underprivileged recipients.

Ultimately, the NGOs & Waste Foods Management System seeks to promote social responsibility, community welfare, and environmental sustainability by reducing food wastage, fighting hunger, and creating an impact-driven ecosystem that benefits both society and the environment.

2B. SCOPE

Our project focuses on developing a mobile and/or web-based application that facilitates the efficient collection, management, and distribution of surplus edible food through

collaboration with donors, volunteers, and NGOs. The platform is designed to reduce food wastage while ensuring timely delivery to underprivileged communities.

1. User Registration & Login

Secure sign-up/login for donors, volunteers, NGO administrators, and beneficiaries.

2. Food Donation Management

Tools for donors to list surplus food, specify quantity, expiry, and pick-up location.

3. Volunteer Coordination Dashboard

Features for assigning, tracking, and managing volunteer pick-ups and deliveries.

4. NGO Admin Panel

Controls for managing donations, beneficiaries, distribution schedules, reports, and analytics.

5. Multiplatform Access

Available on Android, iOS, and web browsers for ease of use across devices.

6. Payment & Contribution Integration

Support for monetary donations, sponsorships, and crowdfunding to aid operations.

3.SYSTEM ANALYSIS

3A. IDENTIFICATION OF NEED

System analysis is a crucial phase in the development of our NGOs & Waste Foods Management System, involving the creation of an efficient, transparent, and user-friendly platform to connect food donors, volunteers, NGOs, and beneficiaries.

With increasing food wastage and the persistent problem of hunger, there is a growing need for a coordinated, technology-driven solution that can manage food donations efficiently and ensure timely distribution to those in need. Traditional food distribution methods often lack real-time coordination, leading to delays, wastage, and poor tracking of resources. The NGOs & Waste Foods Management System addresses these challenges by providing a scalable, digital platform for food rescue and redistribution**.**

Key Needs Identified:

1. Accessibility to Food for Vulnerable Communities

- Underprivileged individuals in remote or underserved areas need access to surplus edible food.
- NGOs require a centralized platform to match available food with urgent needs efficiently.

2. Transparency and Efficiency in Food Distribution

- Traditional systems often lack tracking, making it difficult to verify if food reaches the intended recipients.
- Real-time updates and GPS tracking can ensure timely deliveries and reduce wastage.

3. Effective Donor and Volunteer Coordination

- Donors need an easy way to register surplus food and schedule pick-ups.
- Volunteers and NGOs require a streamlined process for assigning, tracking, and completing delivery tasks.

3B. FEASIBILITY STUDY

The feasibility study of our NGOs & Waste Foods Management System indicates that the concept is both viable and impactful across several key dimensions.

Technically, the system can be developed using widely available tools such as the MERN stack (MongoDB, Express.js, React.js, and Node.js) for scalability, with integration of GPS tracking and cloud storage solutions to ensure real-time updates and secure data handling. Mobile access via Android and iOS ensures ease of use for donors, volunteers, and NGO staff.

Economically, the platform is cost-effective to operate and can be sustained through funding from donors, grants, sponsorships, and partnerships with corporate social responsibility (CSR) programs. The operational cost is relatively low compared to the social value created.

Operationally, the system is straightforward to use, allowing donors to quickly list surplus food, volunteers to accept delivery tasks, and NGOs to manage distribution efficiently. Automation in scheduling, notifications, and reporting enhances efficiency and reduces human error.

Legally, the project is viable with adherence to food safety standards, hygiene regulations, and local laws regarding food redistribution. Data protection measures will be implemented to safeguard donor and beneficiary information.

The estimated development time for the platform is 6 to 8 months, making it achievable within a reasonable timeframe and budget while delivering significant social and environmental benefits.

3C. WORKFLOW

This document plays a vital role in the Software Development Life Cycle (SDLC) as it describes the complete requirements and processes of the NGOs & Waste Foods Management System. It is intended for use by the development team and will serve as the foundation during the testing phase. Any modifications to the requirements in the future must go through a formal change approval process.

The Waterfall Model was chosen as the primary process model for this project. Known as the linear-sequential life cycle model, it is straightforward to understand and implement. In this model, each phase must be completed before moving to the next, with no overlapping between stages.

The Waterfall Model is particularly suitable for this system because the requirements for food collection, storage, and distribution are well-defined and stable, making it possible to follow a structured development process.

- **Waterfall Model Design**

The Waterfall approach ensures that the entire software development process is divided into separate, clearly defined phases. The output of each phase acts as the input for the next, ensuring smooth progression from requirement gathering to deployment.

For the NGOs & Waste Foods Management System, the phases include:

1. Requirement Analysis – Identifying features like donor registration, volunteer coordination, GPS tracking, and food safety compliance.

2. System Design – Creating database structures, data flow diagrams, and user interface layouts.

3. Implementation – Developing frontend and backend modules for food donation and distribution.

4. Testing – Verifying system reliability, security, and functionality.

5. Deployment – Launching the platform for NGO and donor use.

6. Maintenance – Regular updates and improvements based on feedback.

- **Iterative Waterfall Design**

The Iterative Waterfall Model is a variation of the traditional waterfall method. It retains the linear structure but allows for revisiting and refining earlier phases before moving forward.

For example, if during testing the system shows inefficiencies in volunteer route planning, the process can return to the design phase to make improvements. This approach combines the systematic nature of the Waterfall model with the adaptability of iterative development, ensuring that the final system is both robust and flexible.

➢ Requirement Gathering and Analysis

All functional and non-functional requirements for the NGOs & Waste Foods Management System are identified in this phase. This includes donor registration, food listing, volunteer coordination, GPS-enabled tracking, inventory management, and reporting features. The findings are documented in a requirement specification document for reference throughout the project.

➢ System Design

The requirement specifications from the first phase are analyzed, and the system design is prepared. This includes database schema for food donations, volunteer scheduling workflows, and integration plans for payment gateways (for monetary donations). System design also defines the overall architecture, server requirements, and mobile/web interface layout.

➢ Implementation

Using the system design as input, the platform is developed in smaller modules (units) such as Donor Module, Volunteer Module, NGO Admin Panel, and Beneficiary Management System. Each module undergoes Unit Testing to ensure its functionality works as intended.

➢ Integration and Testing

All modules developed in the implementation phase are integrated into the complete platform. The integrated system is tested for performance, security, and reliability. This includes checking the donation posting process, volunteer assignment automation, and tracking food delivery status in real time.

➢ Deployment of the System

After successful testing, the system is deployed for NGO partners, donors, and volunteers. The platform goes live in a production environment, accessible through mobile and web interfaces.

➢ **Maintenance**

Once deployed, the system may require patches to fix bugs or improvements based on user feedback. Future updates might include AI-based donation-demand matching, expanded language support, or integration with national food bank networks. Maintenance ensures smooth, long-term operation of the system.

Advantages:

1. Flexibility – Iterations allow for adjustments based on NGO and donor feedback.

2. Early Delivery – Certain features, like donor registration or basic volunteer management, can be released before the full system is complete.

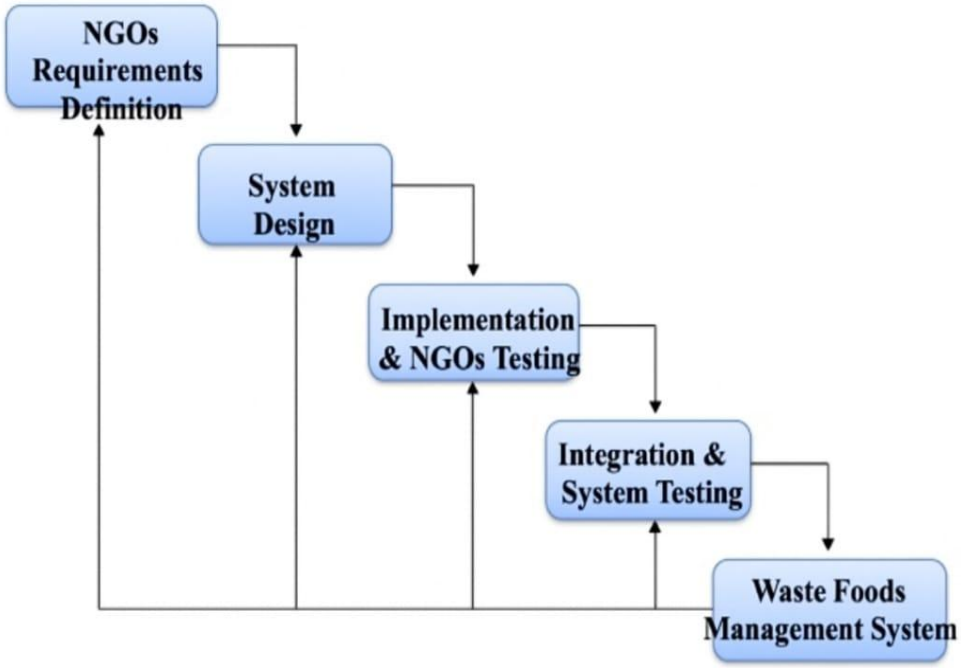
3. Risk Management – Early testing of modules (e.g., donation posting, route optimization) helps detect and fix problems before full deployment.

Disadvantages:

1. Increased Complexity – Revisiting earlier stages can complicate project timelines.

2. Potential for Scope Creep – NGOs may request additional features mid-development, extending the project scope.

3. Resource Intensive – Frequent testing and redesign require more manpower and time.



- **Applications:**

The Iterative Waterfall Model can be highly effective for developing systems that address food waste reduction in collaboration with NGOs. In such projects, requirements often evolve as stakeholders better understand the scale of the issue, logistical challenges, and local community needs.

3D STUDY OF THE SYSTEM

System Modules

The system for managing NGOs and food waste is divided into the following functional modules:

- **Registration:**

1. NGO Registration: NGOs register to access the platform, receive food donations, and manage distribution.

2.Donor Registration: Restaurants, hotels, supermarkets, and individuals register to donate surplus food.

3.Admin Registration: The administrator registers to manage the platform's database, monitor activities, and verify accounts.

- **Account Verification:**

Upon registration, an OTP (One-Time Password) is sent to the registered email/phone number.

This ensures authenticity and prevents fraudulent accounts from being created.

- **Login:**

1.NGO Login: NGOs log in to view available donations, request food, and manage their distribution records.

2.Donor Login: Donors log in to list surplus food items and schedule pickups.

3.Admin Login: Admin logs in to oversee data, approve donations, and coordinate between donors and NGOs.

- **Dashboard:**

1.NGO Dashboard: Displays available donations, request history, and distribution status.

2.Donor Dashboard: Shows listed donations, pickup schedules, and donation history.

3.Admin Dashboard: Provides system-wide control to monitor activities, manage users, and generate reports.

- **Donations:**

1.Donor Interface: Allows donors to post details of surplus food, including quantity, type, and pickup time.

2.NGO Interface: Enables NGOs to claim donations and arrange pickups.

- **Distribution Tracking:**

Tracks the flow of food from donor to NGO to end recipient.

Generates real-time data on quantities distributed, locations served, and time taken.

- **Reports & Analytics:**

1.Impact Reports: Number of meals served, total waste reduced, and beneficiary count.

2.Donation History: Complete history of contributions and distributions.

- **About Section:**

Contains details about the organization behind the platform, its mission, and development team.

- **Account Management:**

1.NGO Profile: View and edit details, track performance, and manage volunteers.

2.Donor Profile: Update contact information, view donation history, and manage pickup preferences.

3.Admin Profile: Access system settings, manage user roles, and handle escalations.

3E. Input and Output

This section outlines the primary inputs, outputs, and major functions of the system.

INPUT

1. Login Credentials – NGOs, donors, and admins enter their username/email and password on the login page.

2. Donation Details – Donors provide information about surplus food, including type, quantity, pickup time, and location.

3. Request Details – NGOs submit requests for available food donations, specifying urgency and delivery preferences.

4. Verification Codes – OTP or email confirmation for account authentication.

OUTPUT

1. For NGOs –

View available donations in real time.

Claim and schedule pickups for requested food.

Track distribution history and impact reports

2. For Donors

View the status of their posted donations.

Access donation history and pickup schedules.

3. For Admin

Access a centralized dashboard with user details, donation history, and distribution tracking.

Approve NGO accounts and verify donor authenticity.

Generate analytics on food waste reduction and beneficiary .

3F. Software Requirement Specifications

The Software Requirements Specification (SRS) defines the scope, functionalities, and technical requirements for the NGOs and Waste Foods Management System. It outlines both functional and non-functional requirements to ensure the system is developed efficiently and meets stakeholder expectations.

This document is prepared after in-depth discussions with NGO representatives, food donors, and the development team.

Developer Responsibilities

The developer is responsible for:

Building the system in alignment with the SRS to address all identified requirements.

Conducting acceptance testing and deploying the system for NGOs and donors after approval.

Providing a comprehensive user manual and technical documentation for smooth operation and maintenance.

Functional Requirements

A. Registration and Authentication

1. NGOs, donors, and admins must be able to register with secure credentials.

2. The system should authenticate all users and maintain secure login sessions.

B. Donation Posting and Requests

1. Donors must be able to post details of surplus food (type, quantity, pickup time).

2. NGOs must be able to request and claim available donations.

C. Search and Browse Donations

1. NGOs can browse all active food listings.

2. Filters (location, food type, urgency) should be available for faster searches.D. Distribution Trackin

1. The system should track the movement of food from donor to NGO to end recipient.

2. Reports on food quantities, distribution timelines, and waste reduction should be generated

.

Hardware Requirements

1. Processor – Intel i3 or higher.

2. RAM – Minimum 8 GB.

3. Storage – SSD with at least 256 GB capacity.

Software Requirements

1. Operating System – Windows 11 or Linux-based OS.

2. Development Tools – Visual Studio Code or similar IDE.

3. Database – MongoDB Atlas for cloud storage and accessibility.

4. Server Environment – Node.js for backend operations.

NGOs and Waste Foods Management System Paradigm Applied

Introduction

Waste Foods Management Systems for NGOs aim to reduce food wastage, improve food redistribution, and ensure resources are delivered efficiently to those in need. These systems rely on structured processes and strategies — known as management paradigms — to design, develop, and maintain an effective platform.

Just as in software engineering, these paradigms can be grouped into categories, where each is interrelated and layered within one another.

Hierarchy of Paradigms

1. Operational Paradigm

Defines the day-to-day processes of food collection, categorization, storage, and distribution.

2. Management Paradigm

Establishes organizational strategies, logistics coordination, and collaboration among NGOs, volunteers, and donors.

3. Technological Paradigm

Focuses on implementing the right software tools and platforms to monitor, track, and optimize waste food management.

Two Levels of Reliability

1. Meeting the Right Needs

Conduct a thorough study of community needs, donor capabilities, and logistics limitations to ensure that the system's objectives align with real-world requirements.

2. Delivering Consistent Results

Ensure that the actual food distribution process matches the planned objectives, maintaining high operational reliability.

4. SYSTEM DESIGN

4A. DATA FLOW DIAGRAM

A Data Flow Diagram (DFD) is a graphical tool used to represent the flow of data within an NGO Waste Foods Management System. It models how information moves through the system, showing how food donations, storage, and distribution are managed.

The DFD provides an overview of the system's structure and is often the first step in designing an efficient waste food management process. It can later be expanded to show detailed workflows.

Purpose of a DFD in Waste Foods Management

Visual Representation: Shows the sources of surplus food (e.g., restaurants, grocery stores, events), the processing steps taken, and the final recipients (e.g., shelters, needy communities).

Data Tracking: Displays where data is collected, processed, stored, and sent, without detailing the exact timing or sequencing of tasks.

Stakeholder Understanding: Enables NGO staff, donors, and volunteers to visualize how the system operates and where they fit in.

Levels of the DFD

1. Context-Level DFD

Shows the entire Waste Foods Management System as one process, with arrows representing incoming food donations and outgoing redistributed goods.

External entities may include Donors, Volunteers, Transport Services, and Recipients.

2. Level 1 DFD

Breaks the system into sub-processes such as:

Food Donation Registration

Quality Check and Categorization

Storage and Inventory Management

Distribution Planning

.

How any system is developed can be determined through a data flow diagram model. In the course of

developing a set of level data flow diagrams, the analyst/designer is forced to address how the system may

be decomposed into component sub-systems and to identify the transaction data in the data model. Data flow

diagrams can be used in both the Analysis and Design phase of the SDLC. There are different notations to

draw data flow diagrams. Defining different visual representations for processes, data stores, data flow, and

external entities.

.

Steps to Construct Data Flow Diagram:

Four Steps are generally used to construct a DFD.

Output

Process should be named and referred for easy reference. Each name should be representative of the reference.

The destination of flow is from top to bottom and from left to right.

When a process is distributed into lower-level details they are numbered.

The names of data stores, sources, and destinations are written in capital letters.

Rules for constructing a Data Flow Diagram:

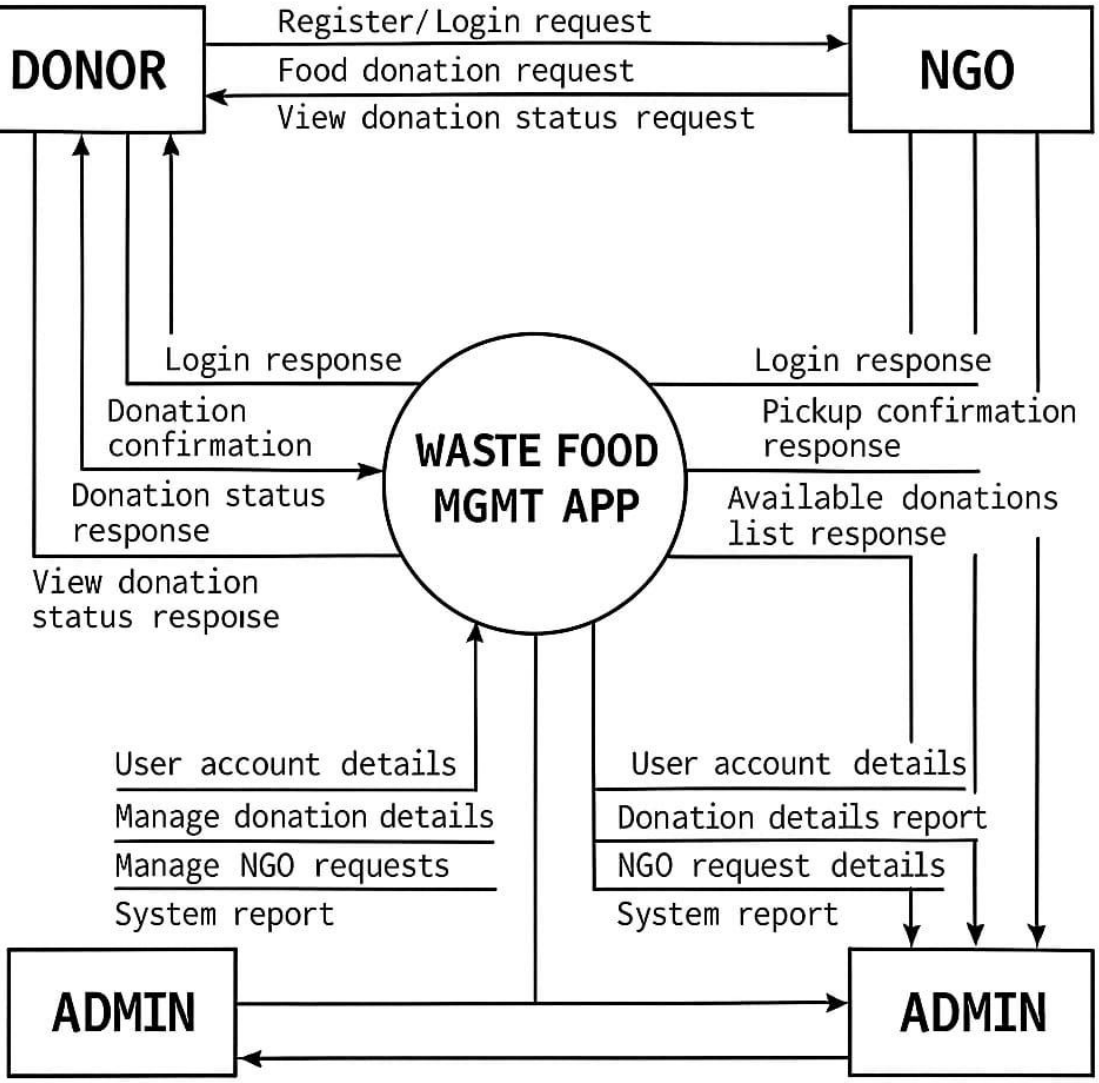
Arrows should not cross each other.

Squares, Circles, and Files must bear a name.

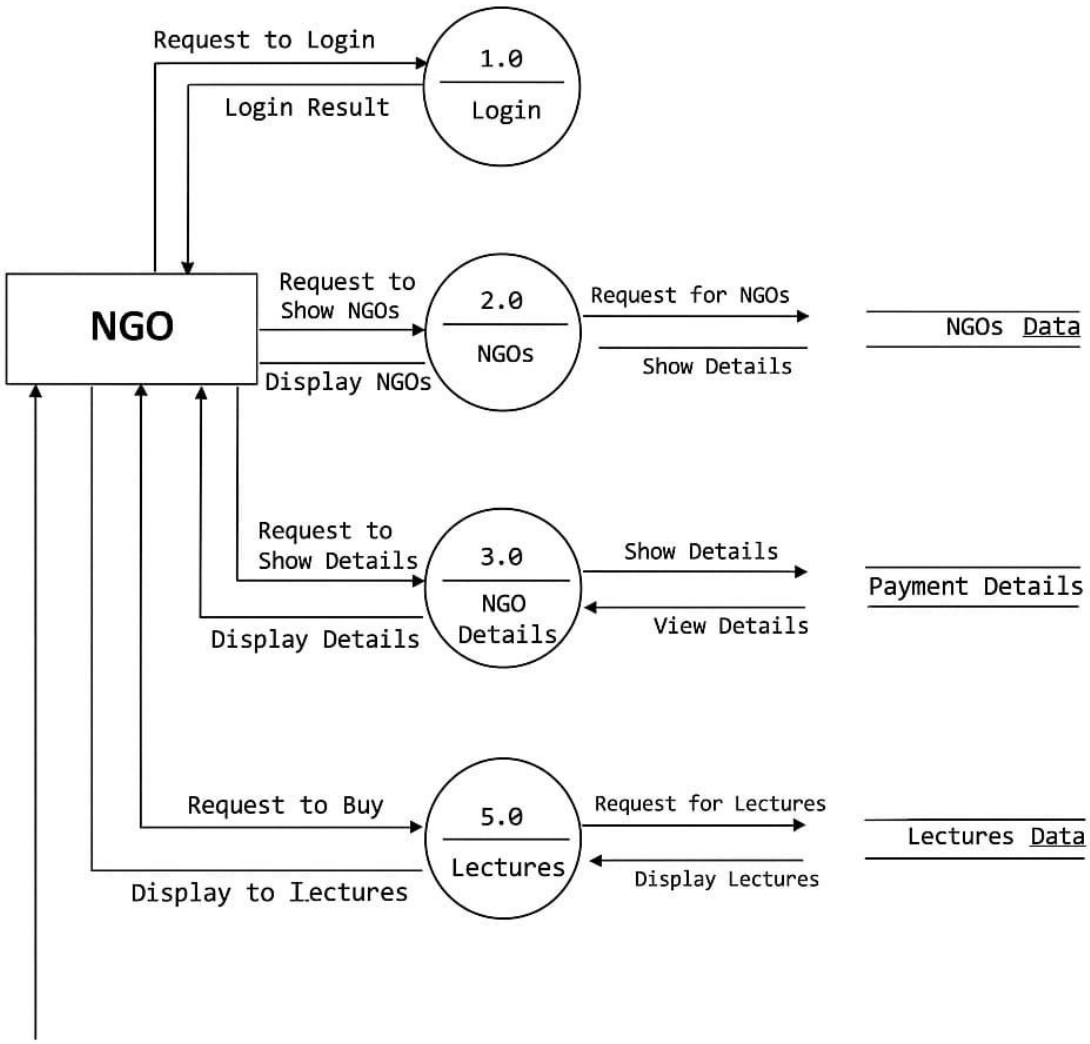
Decomposed data flow squares and circles can have the same names.

Draw all data flow around the outside of the diagram.

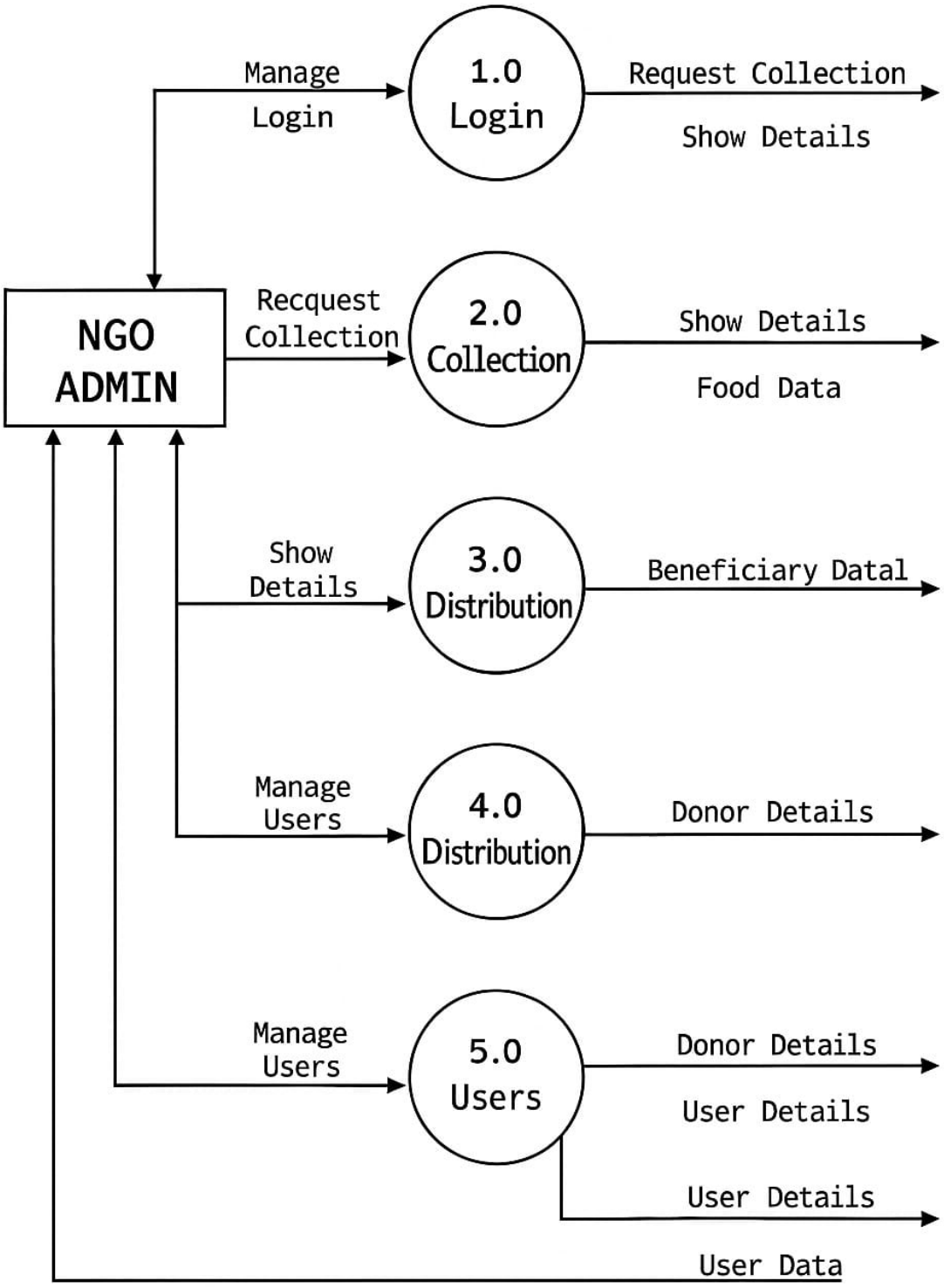
● **LEVEL 0 DFD OR CONTEXT DIAGRAM:**



- **NGO WASTE FOODS MANAGEMENT SYSTEM**



- **LEVEL 1 DFD:**


NGO ADMIN
Manage Login
1.0 Login
Request Collection
Show Details
Recquest Collection
2.0 Collection
Show Details
Food Data
Show Details
3.0 Distribution
Beneficiary Datal
Manage Users
4.0 Distribution
Donor Details
Manage Users
5.0 Users
Donor Details
User Details
User Details
User Data

4B. SEQUENCE DIAGRAM

A Sequence diagram is an interaction diagram that shows how processes operate with one another and

what is their order. It is a construct of a Message Sequence Chart. A sequence diagram shows object

interactions arranged in a time sequence. It depicts the objects and classes involved in the scenario and

the sequence of messages exchanged between the objects needed to carry out the functionality of the

scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of

the system under development.

Sequence diagrams are sometimes called event diagrams or event scenarios.

A sequence diagram shows, as parallel vertical lines (lifelines), different processes

or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in

the order in which they occur. This allows the specification of simple runtime scenarios in a graphical

manner.

A sequence diagram is the most common kind of interaction diagram, which

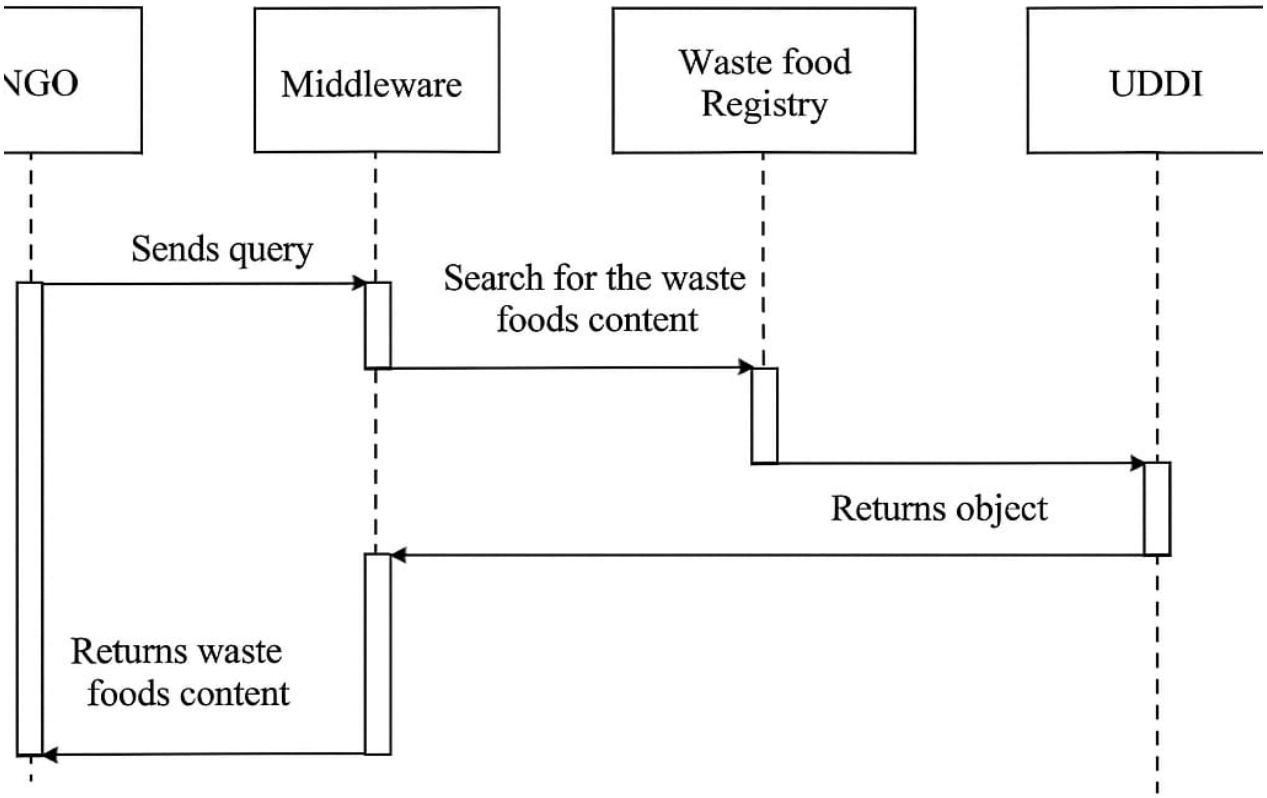
focuses on the message interchange between several life lines .A sequence diagram describes an

interaction by focusing on the sequence of messages that are exchanged, along with their corresponding

occurrence specifications on the lifelines.

The following nodes and edges are typically drawn in a UML sequence diagram: lifeline, execution

specification, message, fragment, interaction, state invariant, continuation, and destruction occurrence.



A Use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the use rand the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well.

So only static behavior is not sufficient to model a system rather dynamic behavior is more important than static behavior. In UML there are five diagrams available to model dynamic nature and a use case diagram is one of them. Now as we have to discuss that the use case diagram is dynamic in nature there should be some internal or external factors for making the interaction.

These internal and external agents are known as actors. So, use case diagrams consist of actors, use cases, and their relationships. The diagram is used to model the

System/subsystem of an application. A single-use case diagram captures a particular functionality of a system.

So, to model the entire system numbers of use case diagrams are used. The purpose of a use case diagram is to capture the dynamic aspect of a system. But this definition is too generic to describe the purpose.

Because the other four diagrams (activity, sequence, collaboration, and State chart) are also having the same purpose. So, we will look into some specific purpose that will distinguish it from the other four diagrams.

Use case diagrams are used to gather the requirements of systemincluding internal and wexternal influences. These requirements are mostly design requirements. So, when a system is analyzed to gather its functionalities use cases are prepared and actors are identified. Now when the initial task is complete use case diagrams are modeled to present the outside view. So, in brief, the purposes of use case diagrams can be as follows:

- Used to gather requirements of a system.
- Used to get an outside view of a system.
- Identify external and internal factors influencing the system.

How to draw Use Case Diagram?

Use case diagrams are considered for high level requirement analysis of a system. So, when the requirements of a system are analyzed, the functionalities are captured in use cases.

So, we can say that uses cases are nothing but the system functionalities written in an organized manner. Now the second things which are relevant to the use cases are the actors. Actors can be defined as something that interacts with the system.

The actors can be human user, some internal applications or may be some external applications. So, in a brief when we are planning to draw use case diagram, we should have the following items identified.

1. Functionalities to be represented as a use case

2. Actors

3. Relationships among the use cases and actors.

Use case diagrams are drawn to capture the functional requirements of a system. So, after identifying the above items we have to follow the following guidelines to draw an efficient use case diagram.

1. The name of a use case is very important. So, the name should be chosen in such a way so that it can identify the functionalities performed.

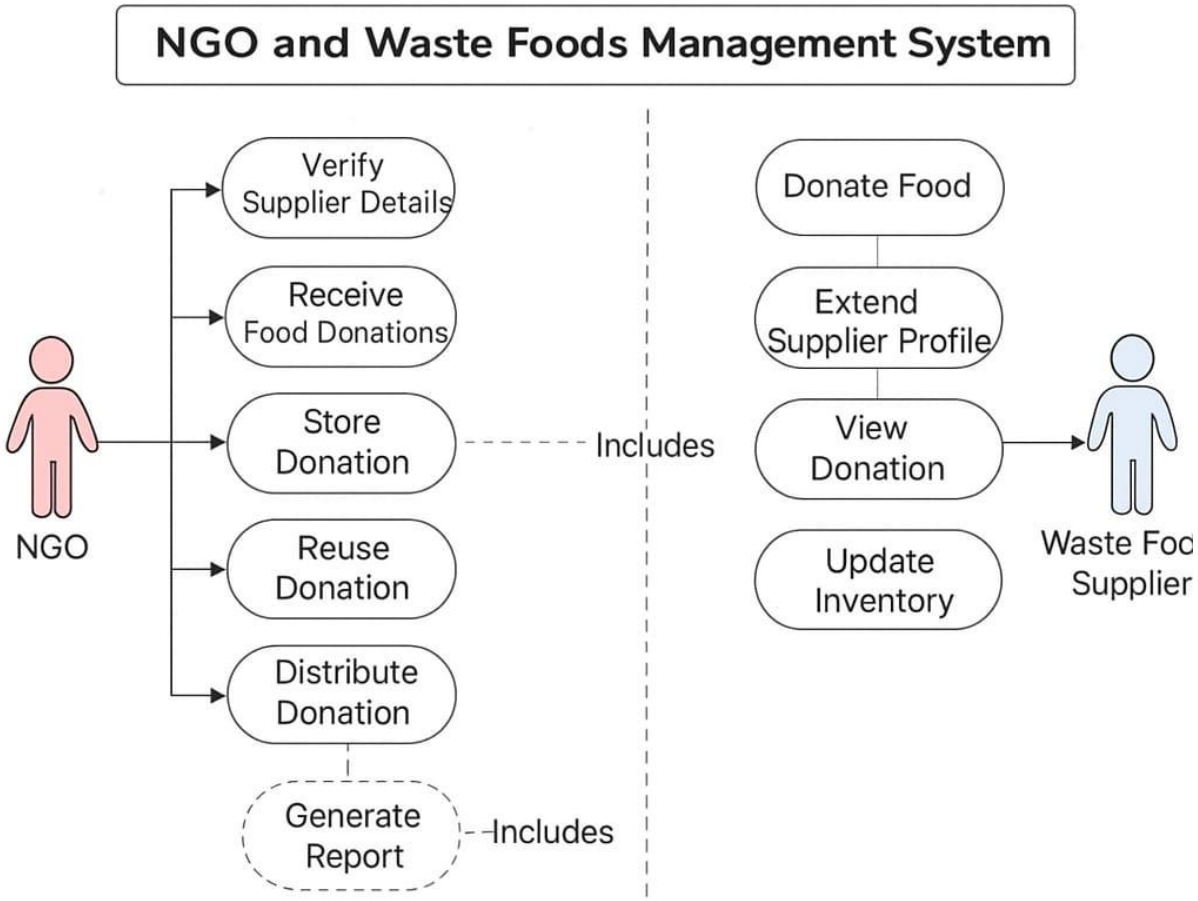
2. Give a suitable name for actors.

3. Show relationships and dependencies clearly in the diagram.

4. Do not try to include all types of relationships. Because the main purpose of the diagram is to identify requirements.

5. Use note whenever required to clarify some important point

- **USE CASE DIAGRAM:**

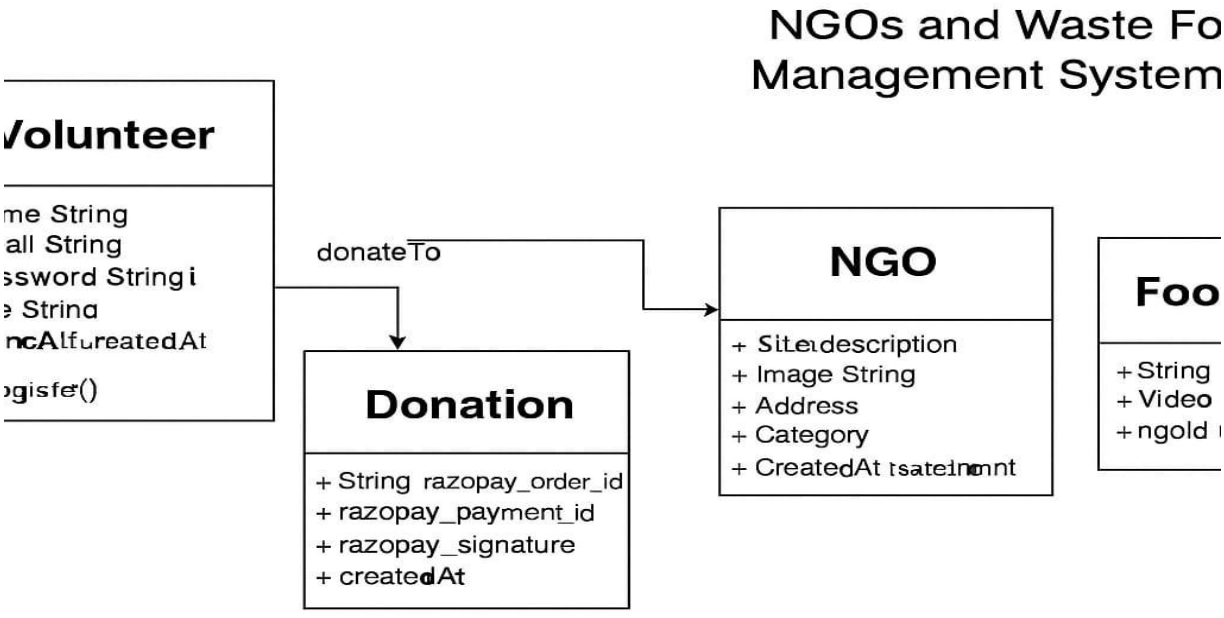


4D. SCHEMA DIAGRAM

The schema is an abstract structure or outline representing the logical view of the database as a whole. Defining categories of data and relationships between those categories, database schema design makes data much easier to retrieve, consume, manipulate, and interpret.

DB schema design organizes data into separate entities, determines how to create relationships between organized entities, and influences the applications of constraints on data. Designers create database schema to give other database users, such as programmers and analysts, a logical understanding of data.

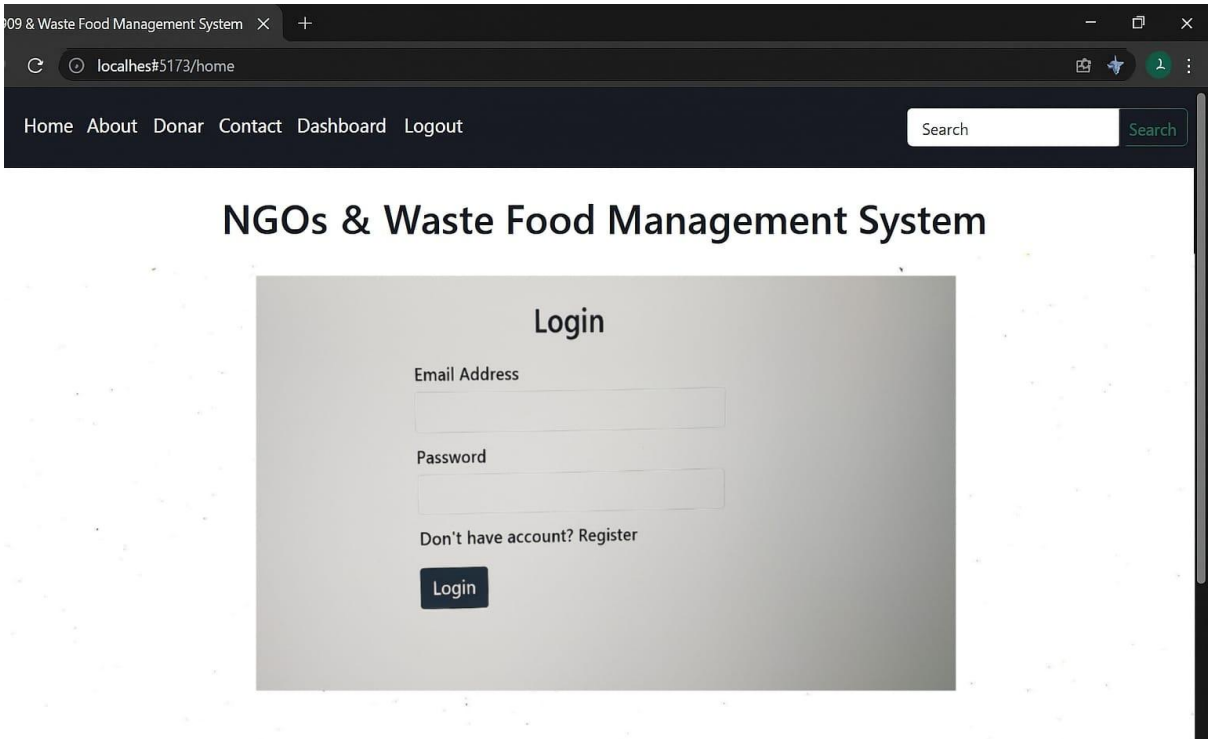
- **SCHEMA DIAGRAM**



5. UI SNAPSHOT

❖ FRONTEND :-

1) Login Page:



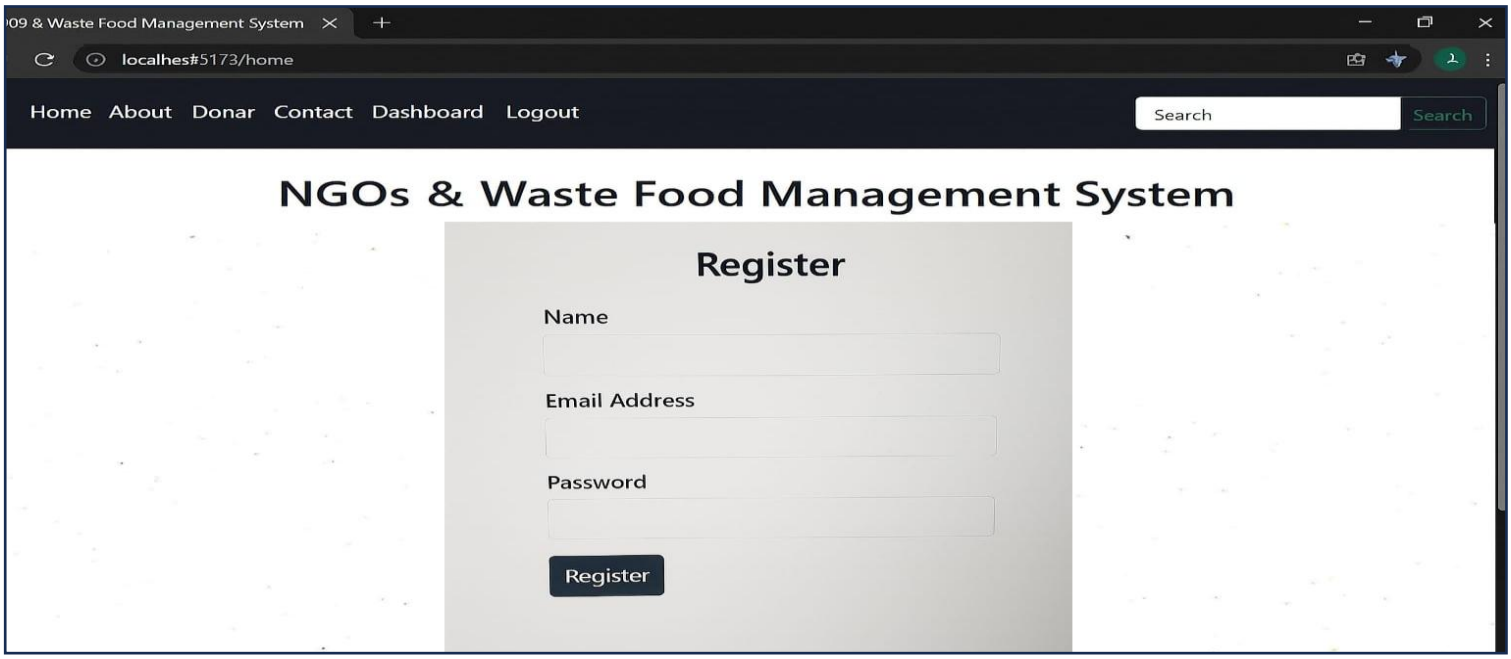
✓ CODE

```
import React, { useState } from "react";

export default function LoginPage() {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");

  const handleSubmit = (e) => {
    e.preventDefault();
```

```
  console.log("Email:", email);
  console.log("Password:", password);
  // Add your login logic here
 };

 return (
  <div className="min-h-screen flex flex-col bg-white">
   {/* Navbar */}
   <nav className="bg-gray-900 text-white px-6 py-3 flex items-center justify-
between">
    <div className="flex gap-6">
     <a href="/home" className="hover:text-gray-300">Home</a>
     <a href="/about" className="hover:text-gray-300">About</a>
     <a href="/donar" className="hover:text-gray-300">Donar</a>
     <a href="/contact" className="hover:text-gray-300">Contact</a>
     <a href="/dashboard" className="hover:text-gray-300">Dashboard</a>
     <a href="/logout" className="hover:text-gray-300">Logout</a>
    </div>
    <div className="flex items-center">
     <input
      type="text"
      placeholder="Search"
      className="px-2 py-1 rounded-l border border-gray-400 focus:outline-none text-
black"
     />
     <button className="bg-green-900 text-white px-3 py-1 rounded-r">
      Search
     </button>
    </div>
   </nav>

   {/* Page Title */}
   <h1 className="text-center text-3xl font-bold mt-8">
    NGOs & Waste Food Management System
   </h1>

   {/* Login Form */}
   <div className="flex justify-center mt-10">
    <form
     onSubmit={handleSubmit}
     className="bg-gray-100 shadow-lg p-8 rounded w-80"
    >
     <h2 className="text-center text-2xl font-semibold mb-6">Login</h2>
```

```
        <label className="block mb-2 font-medium">Email Address</label>
        <input
          type="email"
          value={email}
          onChange={(e) => setEmail(e.target.value)}
          className="w-full px-3 py-2 mb-4 border border-gray-300 rounded focus:outline-
none"
        />

        <label className="block mb-2 font-medium">Password</label>
        <input
          type="password"
          value={password}
          onChange={(e) => setPassword(e.target.value)}
          className="w-full px-3 py-2 mb-4 border border-gray-300 rounded focus:outline-
none"
        />

        <p className="mb-4 text-sm">
          Don’t have account?{" "}
          <a href="/register" className="text-blue-600 hover:underline">
            Register
          </a>
        </p>

        <button
          type="submit"
          className="w-full bg-gray-900 text-white py-2 rounded hover:bg-gray-700"
        >
          Login
        </button>
      </form>
    </div>
  </div>
 );
}
```

Register Page :



✓ CODE

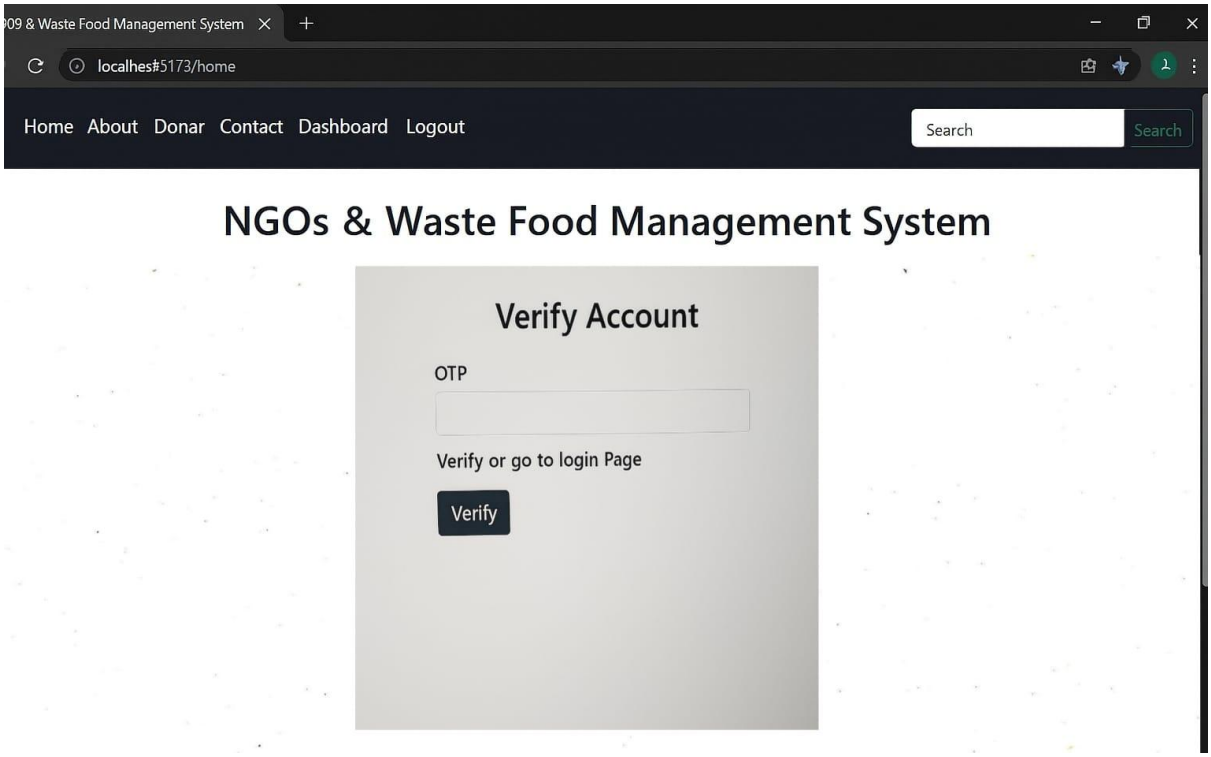
```
import React, { useState } from "react";


export default function Register() {
  const [formData, setFormData] = useState({
    name: "",
    email: "",
    password: "",
  });

  const handleChange = (e) => {
    setFormData({ ...formData, [e.target.name]: e.target.value });
  };

  const handleSubmit = (e) => {
    e.preventDefault();
    console.log("Registration Data:", formData);
    // You can call your API here
```

```
};

return (
  <div className="min-h-screen bg-white">
    {/* Navbar */}
    <nav className="bg-gray-900 text-white px-6 py-3 flex justify-between items-center">
      <div className="flex space-x-6">
        <a href="/home" className="hover:underline">Home</a>
        <a href="/about" className="hover:underline">About</a>
        <a href="/donar" className="hover:underline">Donar</a>
        <a href="/contact" className="hover:underline">Contact</a>
        <a href="/dashboard" className="hover:underline">Dashboard</a>
        <a href="/logout" className="hover:underline">Logout</a>
      </div>
      <div className="flex">
        <input
          type="text"
          placeholder="Search"
          className="px-3 py-1 rounded-l border border-gray-400"
        />
        <button className="bg-green-900 text-white px-4 rounded-r">
          Search
        </button>
      </div>
    </nav>

    {/* Title */}
    <div className="text-center mt-8">
      <h1 className="text-3xl font-bold">NGOs & Waste Food Management System</h1>
```

```
</div>

{/* Register Form */}
<div className="flex justify-center mt-8">
  <form
    onSubmit={handleSubmit}
    className="bg-gray-100 p-8 rounded-lg shadow-md w-96"
  >
    <h2 className="text-xl font-bold text-center mb-6">Register</h2>

    <label className="block mb-2 font-medium">Name</label>
    <input
      type="text"
      name="name"
      value={formData.name}
      onChange={handleChange}
      className="w-full p-2 mb-4 border rounded"
      required
    />

    <label className="block mb-2 font-medium">Email Address</label>
    <input
      type="email"
      name="email"
      value={formData.email}
      onChange={handleChange}
      className="w-full p-2 mb-4 border rounded"
      required
    />
```

```
        <label className="block mb-2 font-medium">Password</label>
        <input
          type="password"
          name="password"
          value={formData.password}
          onChange={handleChange}
          className="w-full p-2 mb-6 border rounded"
          required
        />

        <button
          type="submit"
          className="w-full bg-gray-900 text-white py-2 rounded hover:bg-gray-800"
        >
          Register
        </button>
      </form>
    </div>
  </div>
 );
}
```

Verify Account Page :



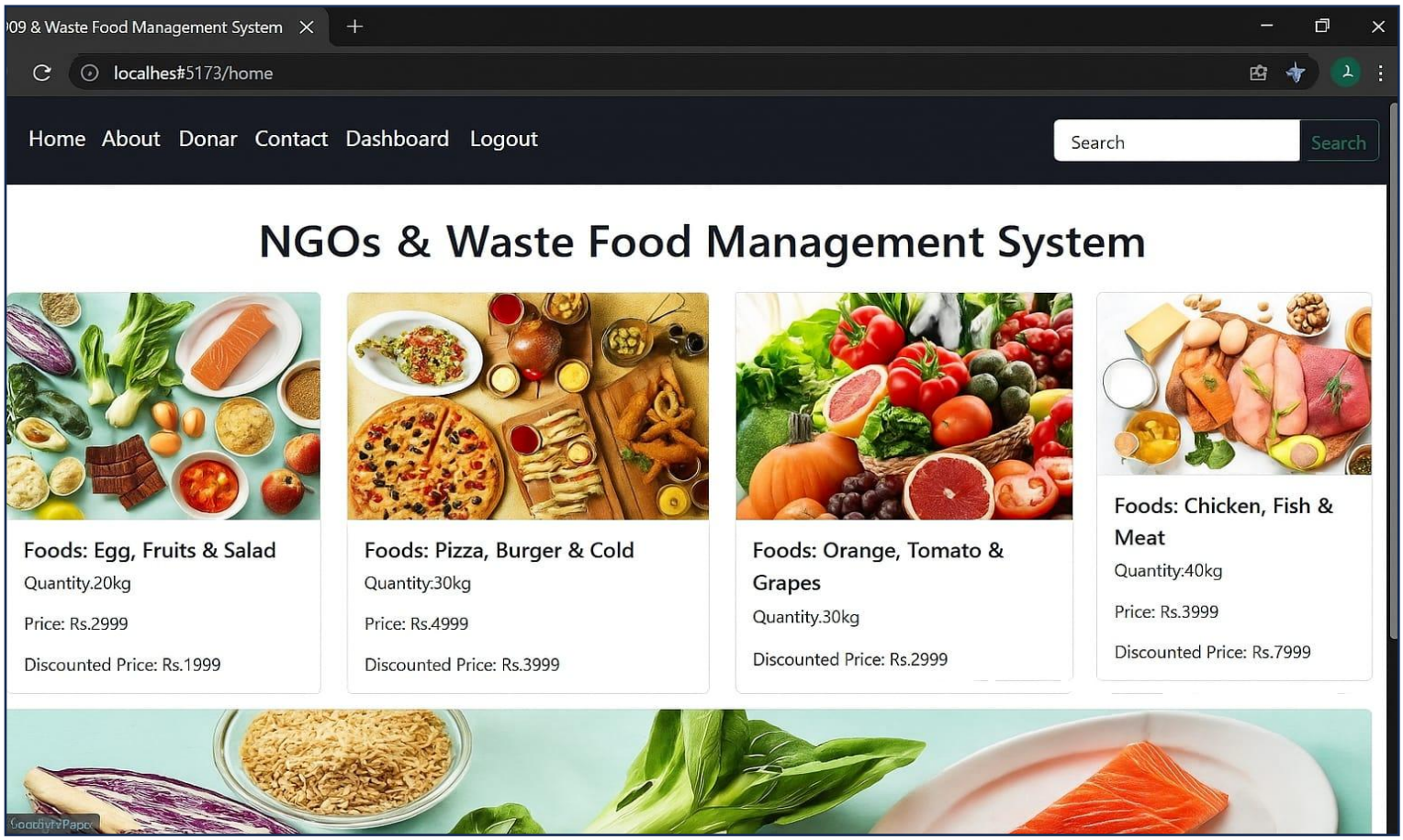
✓ **CODE**

```
import React, { useState } from "react";

function VerifyAccount() {
 const [otp, setOtp] = useState("");

 const handleVerify = () => {
  if (!otp) {
   alert("Please enter OTP");
   return;
  }
  // Call your API here
  console.log("Verifying OTP:", otp);
  alert("OTP Verified (demo)");
 };

 return (
```

```
    <div style={styles.container}>
      <h1 style={styles.title}>NGOs & Waste Food Management System</h1>
      <div style={styles.card}>
        <h2 style={styles.subtitle}>Verify Account</h2>
        <label htmlFor="otp">OTP</label>
        <input
          type="text"
          id="otp"
          value={otp}
          onChange={(e) => setOtp(e.target.value)}
          style={styles.input}
        />
        <p>Verify or go to login Page</p>
        <button style={styles.button} onClick={handleVerify}>
          Verify
        </button>
      </div>
    </div>
  );
}

const styles = {
  container: {
    textAlign: "center",
    padding: "40px",
    fontFamily: "Arial, sans-serif",
  },
  title: {
    fontSize: "24px",
    marginBottom: "20px",
  },
  card: {
    display: "inline-block",
    backgroundColor: "#f5f5f5",
    padding: "30px",
    borderRadius: "10px",
    boxShadow: "0 2px 10px rgba(0,0,0,0.1)",
    textAlign: "left",
    minWidth: "250px",
  },
  subtitle: {
    textAlign: "center",
    marginBottom: "15px",
  },
```

```
  input: {
    width: "100%",
    padding: "8px",
    margin: "8px 0 15px 0",
    borderRadius: "5px",
    border: "1px solid #ccc",
  },
  button: {
    width: "100%",
    padding: "10px",
    backgroundColor: "#14213d",
    color: "#fff",
    border: "none",
    borderRadius: "5px",
    cursor: "pointer",
  },
};

export default VerifyAccount;
```

1) NGOs & Waste Foods Management System PAGE (for admin and user)



- **Header-Page :**

```
import React from 'react'
import {Link,useNavigate} from 'react-router-dom'
const Header = () => {
 const navigate = useNavigate();
 const token = localStorage.getItem('token');


 const hl = ()=>{
  localStorage.removeItem('token');
  navigate('/login');
 }
 return <>
  <nav class="navbar navbar-expand-lg navbar-light bg-dark">
  <Link to="/">Register</Link>|<Link to="/login">Login</Link>|{''}
  {
   token && (
    <>
      <div class="collapse navbar-collapse" id="navbarSupportedContent">
  <ul class="navbar-nav mr-auto">
```

```
    <li class="nav-item active">
      <Link class="nav-link text-light" to="/home">Home <span class="sr-
only">(current)</span></Link>
    </li>
    <li class="nav-item">
      <Link class="nav-link text-light" to="/about">About</Link>
    </li>
    <li class="nav-item dropdown">
      <a class="nav-link dropdown-toggle" href="#" role="button" data-
toggle="dropdown" aria-expanded="false">
        Dropdown
      </a>
      <div class="dropdown-menu">
        <a class="dropdown-item" href="#">Action</a>
        <a class="dropdown-item" href="#">Another action</a>
        <div class="dropdown-divider"></div>
        <a class="dropdown-item" href="#">Something else here</a>
      </div>
    </li>
     <li class="nav-item ">
      <Link class="nav-link text-light" to="/contact">Contact</Link>
    </li>
     <li class="nav-item ">
      <Link class="nav-link text-light" to="/dashboard">Dashboard</Link>|{''}
    </li>
    <button onClick={hl}>Logut</button>

   </ul>
   <form class="form-inline my-2 my-lg-0">
     <input class="form-control mr-sm-2" type="search" placeholder="Search" aria-
label="Search"/>
     <button class="btn btn-outline-success my-2 my-sm-0"
type="submit">Search</button>
   </form>
  </div>
      </>
    )
   }

  <button class="navbar-toggler bg-danger" type="button" data-toggle="collapse"
data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent"
aria-expanded="false" aria-label="Toggle navigation">
   <span class="navbar-toggler-icon"></span>
  </button>
```

```
</nav>
 </>
}

export default Header
```

- **Components-Foodcars.jsx**

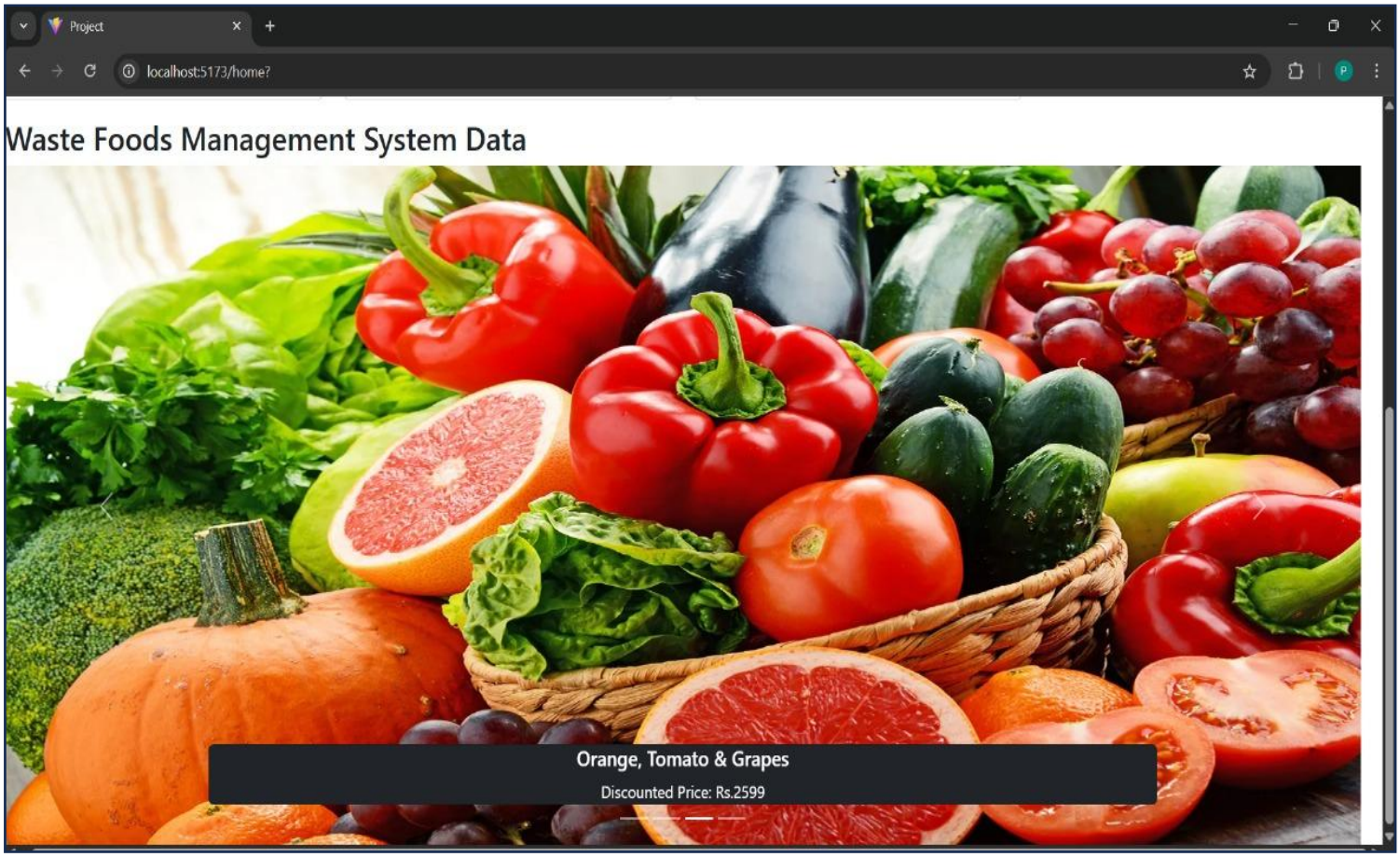
```
import React from 'react'
import { Card } from 'react-bootstrap'

const Foodscard = ({foods}) => {

 return <>
  <Card className='mt-3'>
  <Card.Img src={foods.image} variant='top'/>
  <Card.Body>
      <Card.Title>Foods:{foods.name}</Card.Title>
     <Card.Text>Quantity:{foods.Quantity}</Card.Text>
     <Card.Text>Price:{foods.Price}</Card.Text>
     <Card.Text>Discounted Price:{foods.DiscountedPrice}</Card.Text>
  </Card.Body>

  </Card>
 </>
}

export default Foodscard
```

- **Component-Home.jsx**

```
import React from 'react'

import Foodsdata from '../data/data'

import Foodscard from './Foodscard'

import Foodslider from './Foodslider'


const Home = () => {
```

```
 return <>
 <div className='row'>
  {
    Foodsdata.map(x=>(
      <div className='col-md-3'>
       <Foodscard foods={x}/>
      </div>
    ))
  }
 </div>
 <div className='container-fluid mt-3'>
  <h2>Waste Foods Management System Data</h2>
  <Foodslider foods={Foodsdata}/>
 </div>

 </>
}

export default Home
```

4) FOODS DESCRIPTION PAGE AND SYSTEM (for user):

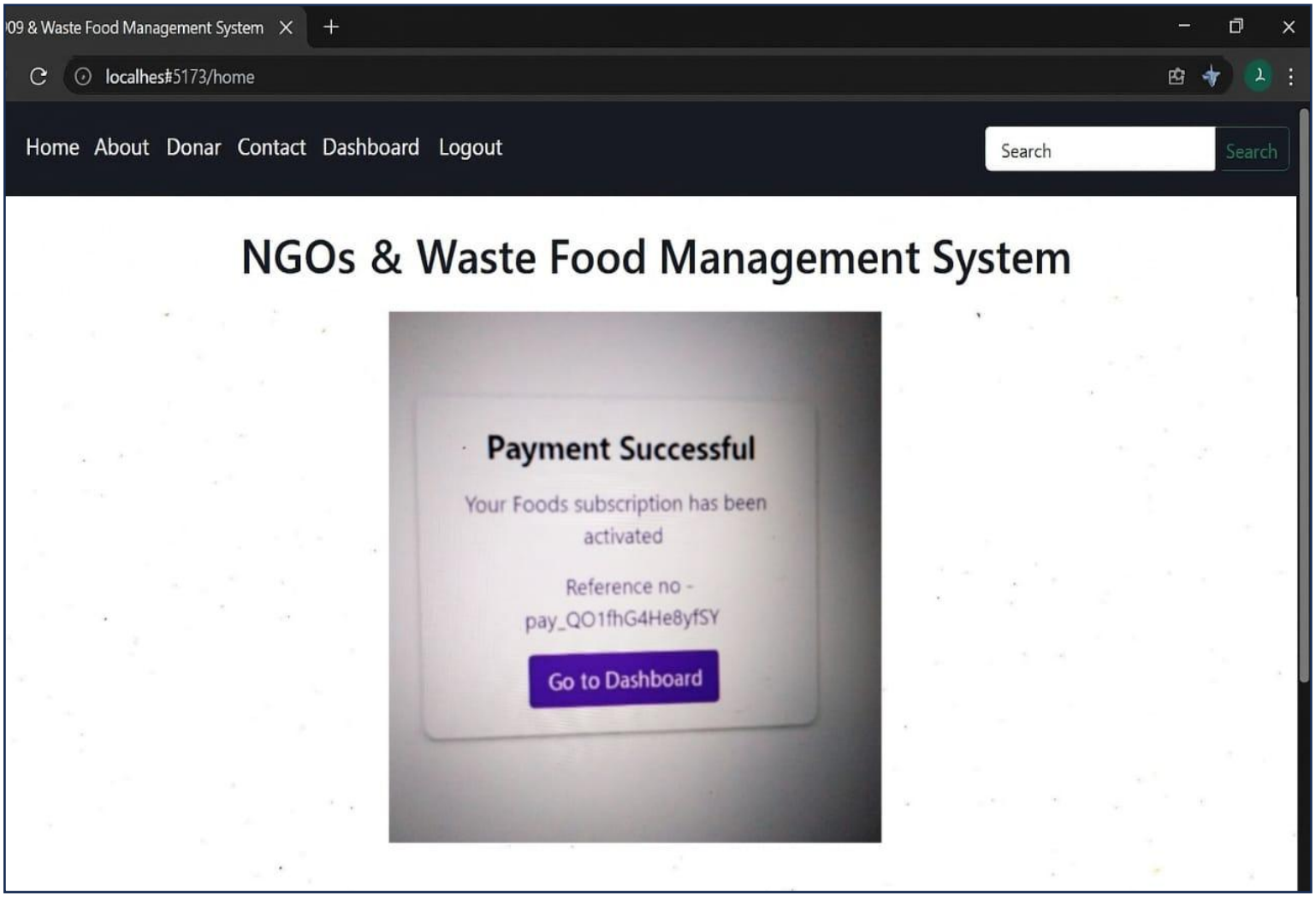


✓ **CODE**

```
import React from "react";

export default function WasteFoodCard() {
 return (
  <div className="min-h-screen bg-white">
   {/* Title */}
   <h1 className="text-3xl font-bold p-4">
    Waste Foods Management System Data
   </h1>

   {/* Image Container */}
   <div className="relative w-full">
    <img
```

```
        src="https://images.unsplash.com/photo-1604908177225-
6ecf3d8b3c5c" // Replace with your image URL
        alt="Fruits and Vegetables"
        className="w-full h-[500px] object-cover"
       />

       {/* Overlay */}
       <div className="absolute bottom-0 w-full bg-black bg-opacity-
80 text-white py-3 text-center">
        <p className="text-lg font-semibold">Orange, Tomato &
Grapes</p>
        <p className="text-sm">Discounted Price: Rs.2599</p>
       </div>
     </div>
   </div>
  );
}
```

5) PAYMENT SUCCESSFFULLY PAGE:

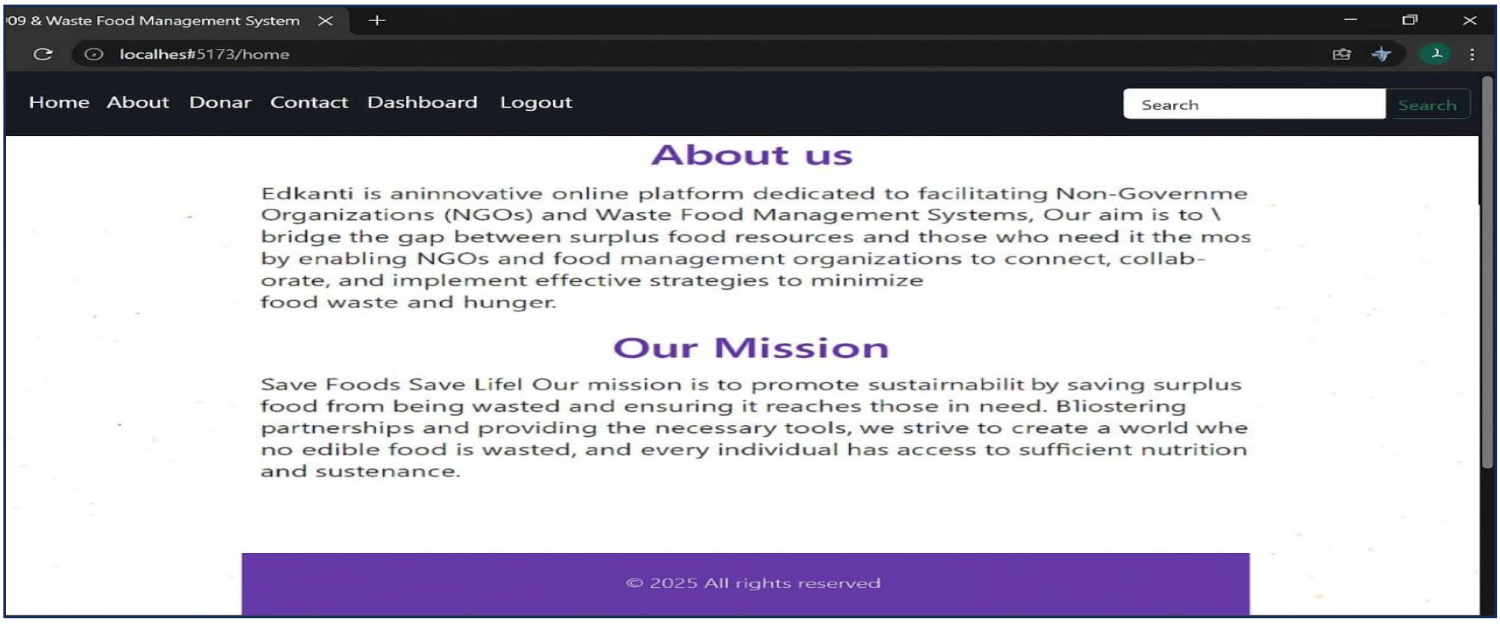


✓ **CODE**

```
import React from "react";

export default function PaymentSuccess() {
 return (
  <div className="min-h-screen bg-white flex flex-col">
   {/* Navigation Bar */}
   <nav className="bg-gray-900 text-white flex justify-between
items-center px-6 py-3">
    <div className="flex space-x-6">
     <a href="/home" className="hover:text-gray-300">Home</a>
     <a href="/about" className="hover:text-gray-300">About</a>
     <a href="/donar" className="hover:text-gray-300">Donar</a>
     <a href="/contact" className="hover:text-gray-
300">Contact</a>
     <a href="/dashboard" className="hover:text-gray-
300">Dashboard</a>
     <a href="/logout" className="hover:text-gray-
300">Logout</a>
    </div>
    <div className="flex">
     <input
      type="text"
      placeholder="Search"
      className="px-3 py-1 text-black rounded-l border"
     />
     <button className="bg-gray-700 px-3 rounded-r hover:bg-gray-
600">
      Search
     </button>
    </div>
   </nav>

   {/* Page Content */}
   <main className="flex flex-col items-center py-10">
    <h1 className="text-3xl font-bold mb-8">
```

```
          NGOs & Waste Food Management System
        </h1>

        {/* Payment Card */}
        <div className="bg-white shadow-md rounded-lg p-8 text-
center border">
          <h2 className="text-xl font-bold mb-2">Payment
Successful</h2>
          <p className="text-gray-600 mb-4">
            Your Foods subscription has been activated
          </p>
          <p className="text-gray-500 mb-6">
            Reference no - <span className="font-
mono">pay_QO1fhG4He8yfSY</span>
          </p>
          <a
            href="/dashboard"
            className="bg-purple-700 text-white px-5 py-2 rounded
hover:bg-purple-800"
          >
            Go to Dashboard
          </a>
        </div>
      </main>
    </div>
  );
}
```

6) About Page:



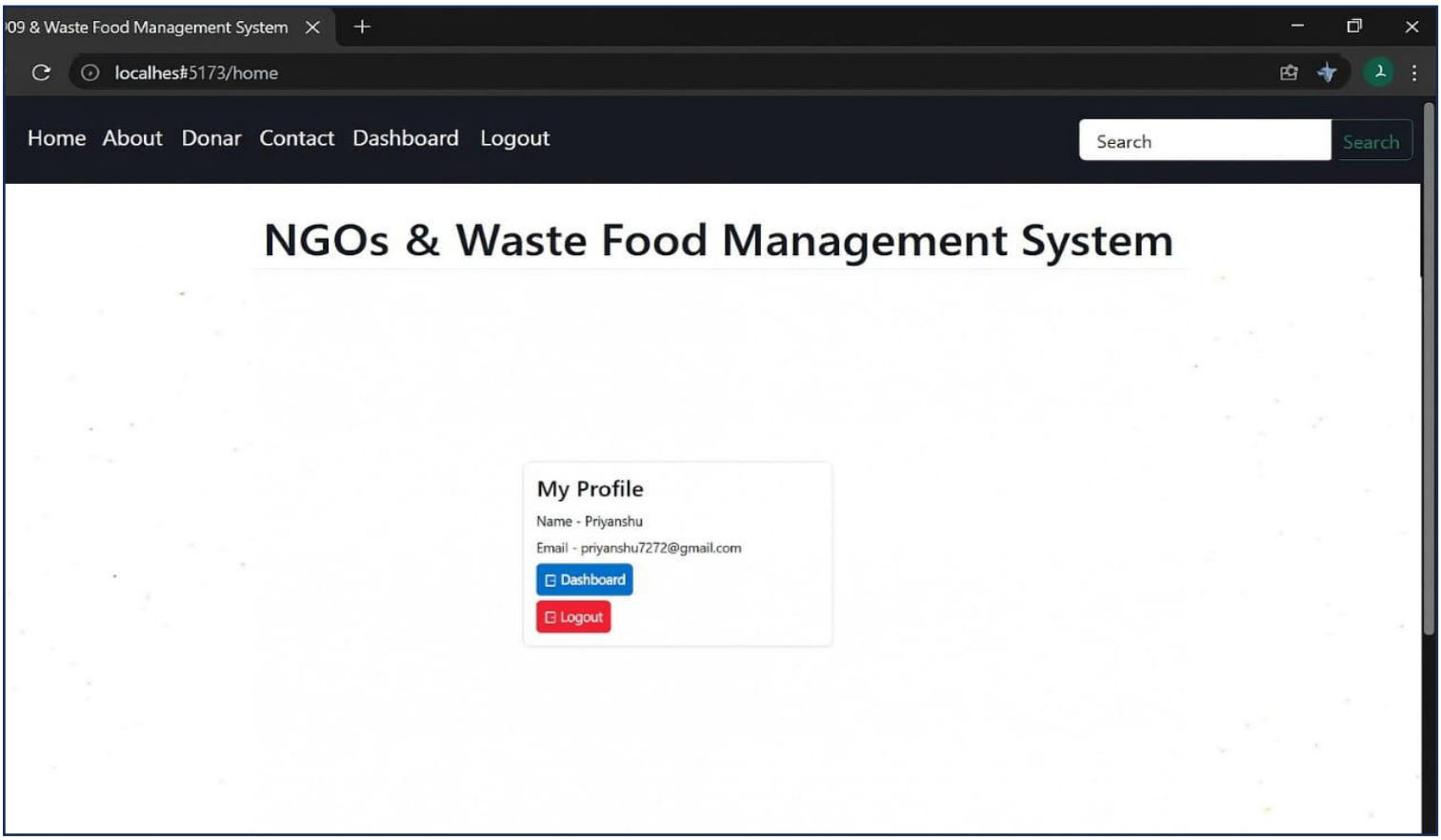
✓ **CODE:**

```
import React from "react";

export default function AboutPage() {
  return (
    <div className="min-h-screen bg-white text-gray-800">
      {/* Navbar */}
      <nav className="bg-gray-900 text-white px-6 py-4 flex items-
center justify-between">
        <div className="flex gap-6">
          <a href="/home" className="hover:text-purple-
400">Home</a>
          <a href="/about" className="hover:text-purple-
400">About</a>
          <a href="/donor" className="hover:text-purple-
400">Donor</a>
          <a href="/contact" className="hover:text-purple-
400">Contact</a>
          <a href="/dashboard" className="hover:text-purple-
400">Dashboard</a>
          <a href="/logout" className="hover:text-purple-
400">Logout</a>
        </div>
        <div className="flex">
          <input
            type="text"
            placeholder="Search"
            className="px-2 py-1 rounded-l-md border border-gray-400
text-black"
          />
          <button className="bg-purple-500 px-4 py-1 rounded-r-md
hover:bg-purple-600">
            Search
          </button>
        </div>
      </nav>
```

```
    {/* About Us */}
    <section className="max-w-4xl mx-auto py-12 px-4 text-center">
      <h2 className="text-3xl font-bold text-purple-600 mb-4">About
us</h2>
      <p className="text-lg leading-relaxed">
        Edkanti is an innovative online platform dedicated to facilitating
        Non-Government Organizations (NGOs) and Waste Food
Management Systems.
        Our aim is to bridge the gap between surplus food resources
and those
        who need it the most by enabling NGOs and food management
organizations
        to connect, collaborate, and implement effective strategies to
minimize
        food waste and hunger.
      </p>
    </section>

    {/* Mission */}
    <section className="max-w-4xl mx-auto py-8 px-4 text-center">
      <h2 className="text-3xl font-bold text-purple-600 mb-4">Our
Mission</h2>
      <p className="text-lg leading-relaxed">
        Save Foods Save Life! Our mission is to promote sustainability by
saving
        surplus food from being wasted and ensuring it reaches those in
need.
        By fostering partnerships and providing the necessary tools, we
strive
        to create a world where no edible food is wasted, and every
individual
        has access to sufficient nutrition and sustenance.
      </p>
    </section>

    {/* Footer */}
```

```
    <footer className="bg-purple-700 text-white py-4 text-center mt-
8">
      © 2025 All rights reserved
    </footer>
   </div>
  );
}
```

7) COUNT PAGE (for admin and user):



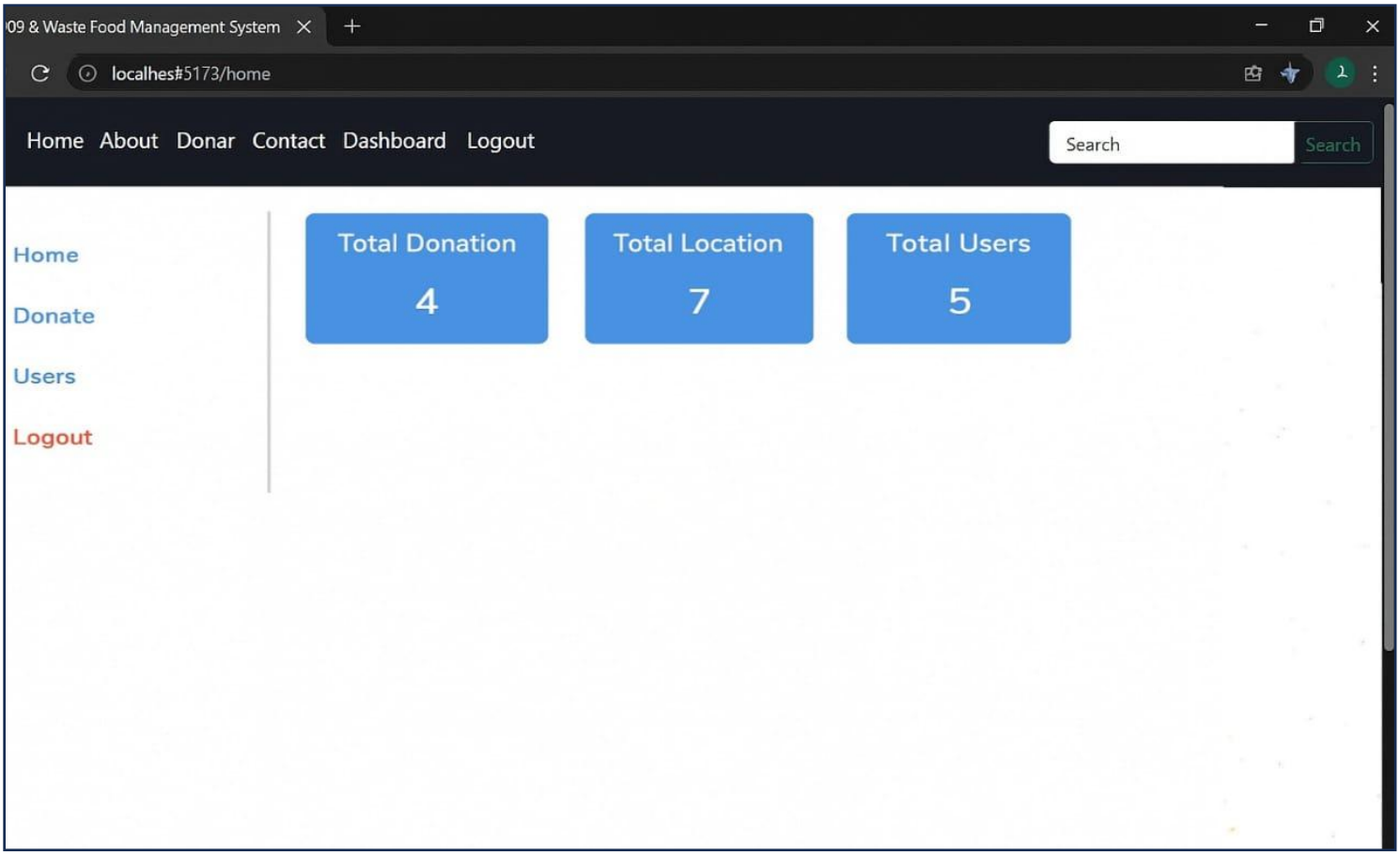
✓ CODE:

```
// App.js
import React from "react";
import "./App.css";

function Navbar() {
```

```
  return (
    <nav className="navbar">
      <div className="nav-left">
        <a href="/home" className="nav-link">Home</a>
        <a href="/about" className="nav-link">About</a>
        <a href="/donar" className="nav-link">Donar</a>
        <a href="/contact" className="nav-link">Contact</a>
        <a href="/dashboard" className="nav-link">Dashboard</a>
        <a href="/logout" className="nav-link">Logout</a>
      </div>
      <div className="nav-right">
        <input type="text" placeholder="Search" className="search-
input" />
        <button className="search-btn">Search</button>
      </div>
    </nav>
  );
}

function ProfileCard() {
  return (
    <div className="profile-card">
      <h3>My Profile</h3>
      <p><strong>Name -</strong> Priyanshu</p>
      <p><strong>Email -</strong> priyanshu7272@gmail.com</p>
      <div className="button-group">
        <button className="btn dashboard-btn">📊
Dashboard</button>
        <button className="btn logout-btn">🔴 Logout</button>
      </div>
    </div>
  );
}

export default function App() {
  return (
    <div>
```

```
      <Navbar />
      <main>
        <h1 className="title">NGOs & Waste Food Management
System</h1>
        <ProfileCard />
      </main>
    </div>
  );
}
```

8) ADMIN DASHBOARD PAGE:



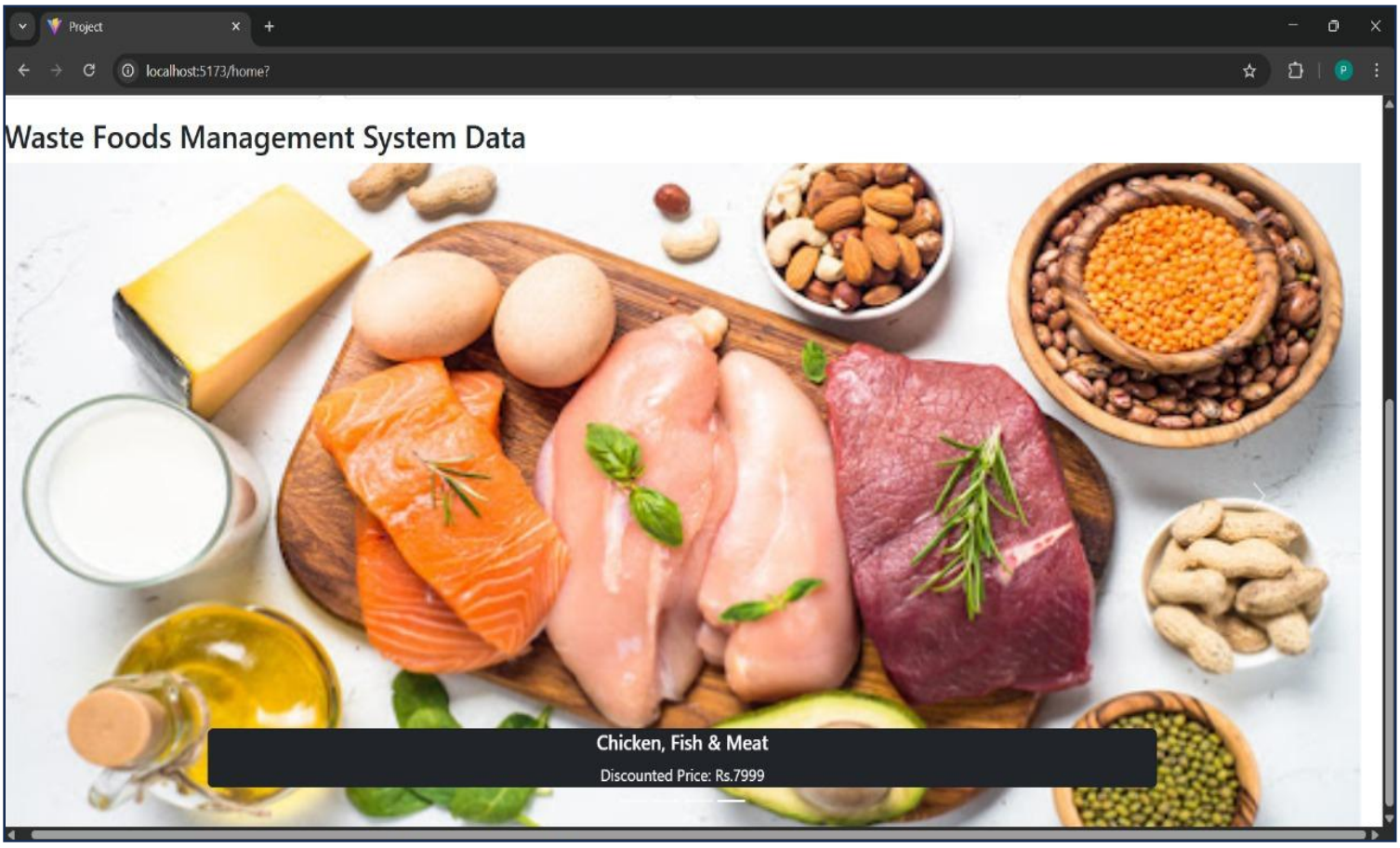
✓ CODE:

```
import React from "react";
import "./Dashboard.css";
```

```
export default function Dashboard() {
  return (
    <div className="dashboard">
      {/* Top Navbar */}
      <nav className="navbar">
        <div className="nav-links">
          <a href="#home">Home</a>
          <a href="#about">About</a>
          <a href="#donar">Donar</a>
          <a href="#contact">Contact</a>
          <a href="#dashboard">Dashboard</a>
          <a href="#logout">Logout</a>
        </div>
        <div className="search-bar">
          <input type="text" placeholder="Search" />
          <button>Search</button>
        </div>
      </nav>

      <div className="content-wrapper">
        {/* Sidebar */}
        <aside className="sidebar">
          <a href="#home">Home</a>
          <a href="#donate">Donate</a>
          <a href="#users">Users</a>
          <a href="#logout" className="logout">Logout</a>
        </aside>

        {/* Main Content */}
        <main className="main-content">
          <div className="card">
            <h3>Total Donation</h3>
            <p>4</p>
          </div>
          <div className="card">
            <h3>Total Location</h3>
```

```
          <p>7</p>
        </div>
        <div className="card">
          <h3>Total Users</h3>
          <p>5</p>
        </div>
      </main>
    </div>
  </div>
 );
}
```

9) ADMIN FOOD PAGE:

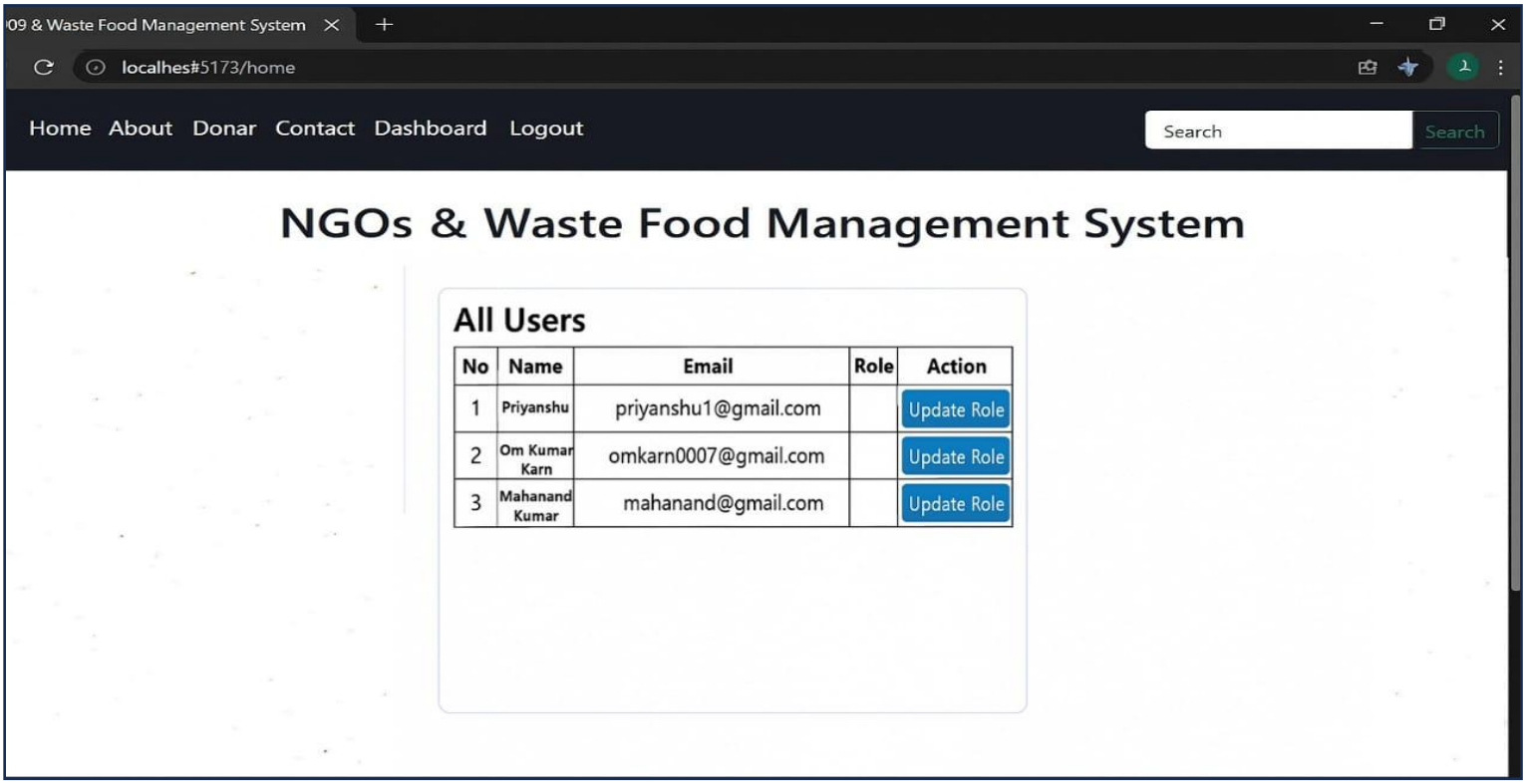


- ✓ **CODE:**

```
import React from "react";

export default function FoodCard() {
```

```
 return (
  <div style={{ fontFamily: "Arial, sans-serif" }}>
   {/* Page Heading */}
   <h1 style={{ margin: "20px", fontWeight: "bold" }}>
    Waste Foods Management System Data
   </h1>

   {/* Image Section */}
   <div
    style={{
     position: "relative",
     width: "100%",
     maxWidth: "900px",
     margin: "0 auto",
    }}
   >
    <img
     src="https://images.unsplash.com/photo-1605478261103-8d56e8f02a0b" //
replace with your image URL or import
     alt="Chicken, Fish & Meat"
     style={{
      width: "100%",
      height: "auto",
      borderRadius: "5px",
     }}
    />

    {/* Overlay Text */}
    <div
     style={{
      position: "absolute",
      bottom: "0",
      width: "100%",
      background: "rgba(0, 0, 0, 0.8)",
      color: "#fff",
      textAlign: "center",
      padding: "10px 0",
     }}
    >
     <h3 style={{ margin: 0 }}>Chicken, Fish & Meat</h3>
     <p style={{ margin: "5px 0 0" }}>Discounted Price: Rs. 7999</p>
    </div>
   </div>
  </div>
 );
}
```

10) ADMIN USERS PAGE:



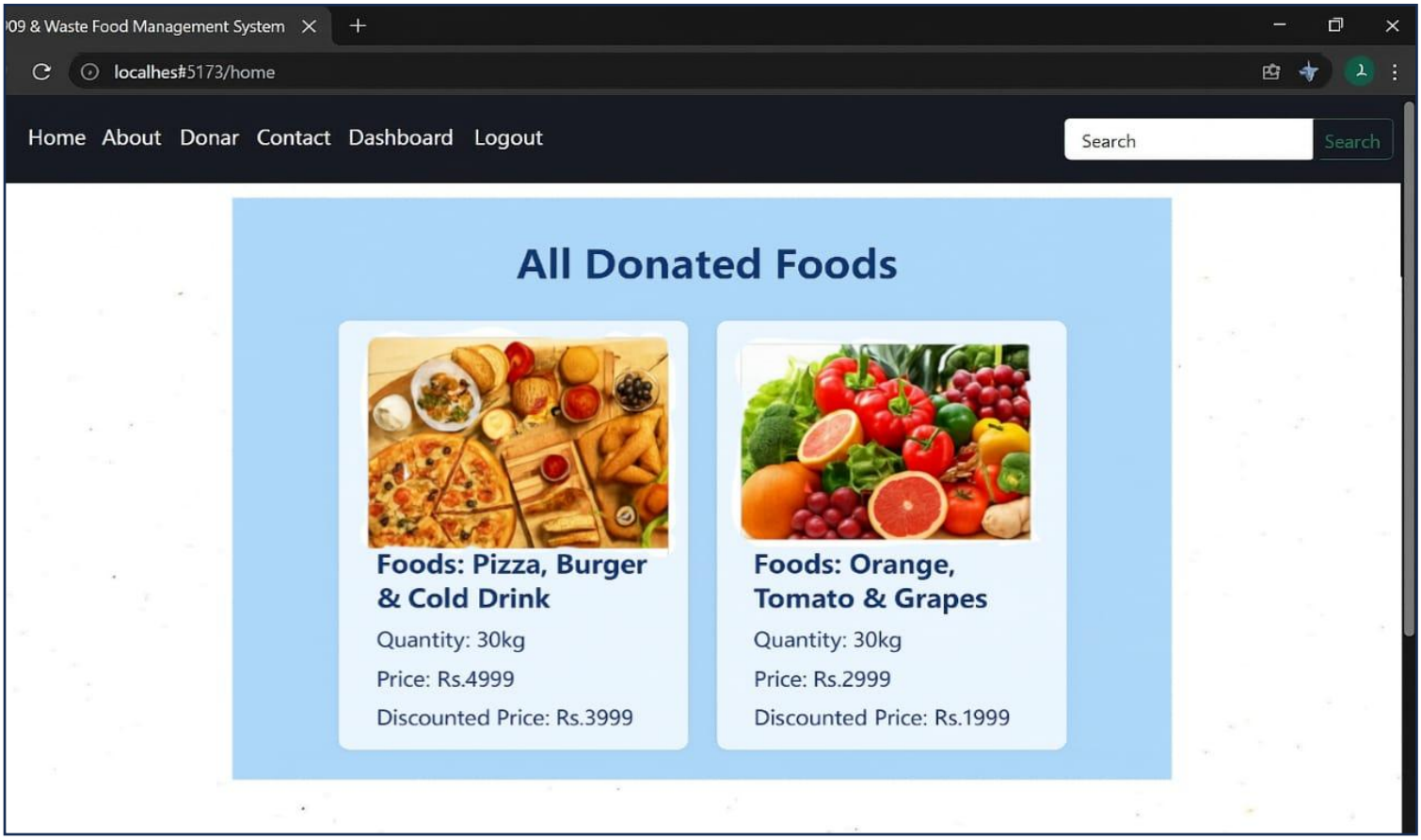
✓ **CODE:**

```
import React, { useState } from "react";

export default function UserManagement() {
 const [users, setUsers] = useState([
  { id: 1, name: "Priyanshu", email: "priyanshu1@gmail.com", role: "" },
  { id: 2, name: "Om Kumar Karn", email: "omkarn0007@gmail.com", role: "" },
  { id: 3, name: "Mahanand Kumar", email: "mahanand@gmail.com", role: "" },
 ]);

 const handleUpdateRole = (id) => {
  const newRole = prompt("Enter new role:");
  if (newRole !== null) {
   setUsers(users.map(user =>
    user.id === id ? { ...user, role: newRole } : user
   ));
  }
 };

 return (
  <div className="min-h-screen bg-white text-black">
   {/* Navbar */}
```

```
    <nav className="bg-[#0d1b2a] text-white px-6 py-4 flex justify-between items-
center">
      <ul className="flex space-x-6">
        <li><a href="#" className="hover:underline">Home</a></li>
        <li><a href="#" className="hover:underline">About</a></li>
        <li><a href="#" className="hover:underline">Donar</a></li>
        <li><a href="#" className="hover:underline">Contact</a></li>
        <li><a href="#" className="hover:underline">Dashboard</a></li>
        <li><a href="#" className="hover:underline">Logout</a></li>
      </ul>
      <div className="flex">
        <input
          type="text"
          placeholder="Search"
          className="px-2 py-1 rounded-l border border-gray-300 text-black"
        />
        <button className="bg-green-700 px-3 rounded-r">Search</button>
      </div>
    </nav>

    {/* Title */}
    <div className="text-center mt-8">
      <h1 className="text-3xl font-bold">NGOs & Waste Food Management
System</h1>
    </div>

    {/* Table */}
    <div className="max-w-3xl mx-auto mt-6 bg-white shadow rounded-lg p-4">
      <h2 className="text-lg font-bold mb-3">All Users</h2>
      <table className="w-full border border-black">
        <thead>
          <tr className="bg-gray-100">
            <th className="border border-black px-4 py-2">No</th>
            <th className="border border-black px-4 py-2">Name</th>
            <th className="border border-black px-4 py-2">Email</th>
            <th className="border border-black px-4 py-2">Role</th>
            <th className="border border-black px-4 py-2">Action</th>
          </tr>
        </thead>
        <tbody>
          {users.map((user, index) => (
            <tr key={user.id}>
              <td className="border border-black px-4 py-2 text-center">{index + 1}</td>
              <td className="border border-black px-4 py-2">{user.name}</td>
              <td className="border border-black px-4 py-2">{user.email}</td>
              <td className="border border-black px-4 py-2">{user.role}</td>
              <td className="border border-black px-4 py-2 text-center">
                <button
```

```
                  className="bg-blue-500 hover:bg-blue-600 text-white px-3 py-1 rounded"
                  onClick={() => handleUpdateRole(user.id)}
                >
                  Update Role
                </button>
              </td>
            </tr>
          ))}
        </tbody>
      </table>
    </div>
  </div>
 );
}
```

11) USER DASHBOARD PAGE:



✓ **CODE:**

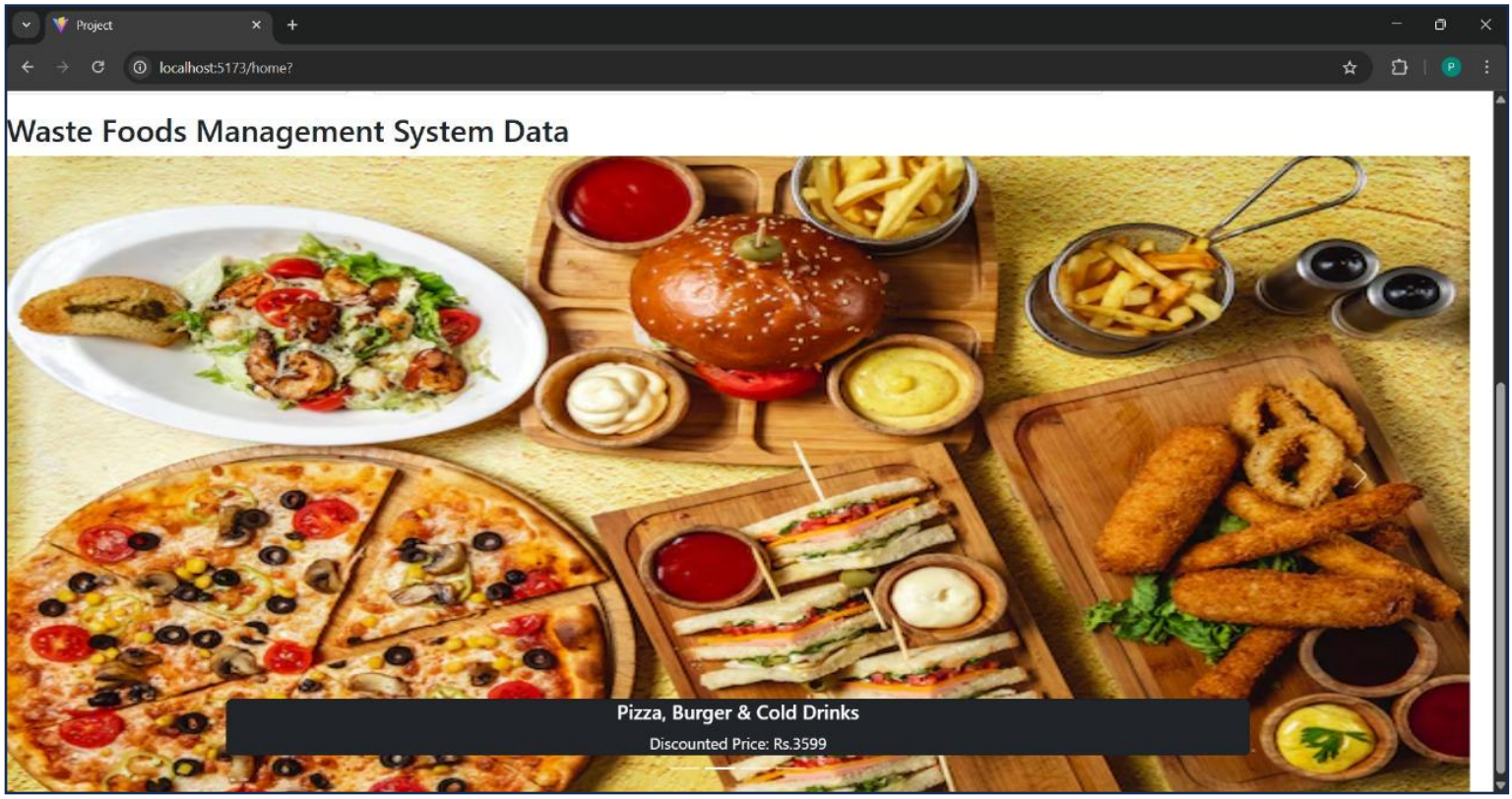
```
import React from "react";
```

```
const donatedFoods = [
 {
  id: 1,
  name: "Pizza, Burger & Cold Drink",
  quantity: "30kg",
  price: 4999,
  discountedPrice: 3999,
  image:
"https://via.placeholder.com/300x200?text=Pizza+Burger+Drink", //
Replace with real image URL
 },
 {
  id: 2,
  name: "Orange, Tomato & Grapes",
  quantity: "30kg",
  price: 2999,
  discountedPrice: 1999,
  image:
"https://via.placeholder.com/300x200?text=Fruits+Veggies", //
Replace with real image URL
 },
];

export default function DonatedFoods() {
 return (
  <div className="min-h-screen bg-white">
   {/* Navbar */}
   <nav className="bg-[#0d1b2a] text-white px-6 py-4 flex justify-
between items-center">
    <div className="flex gap-6">
     <a href="/" className="hover:text-gray-300">Home</a>
     <a href="/" className="hover:text-gray-300">About</a>
     <a href="/" className="hover:text-gray-300">Donar</a>
     <a href="/" className="hover:text-gray-300">Contact</a>
     <a href="/" className="hover:text-gray-300">Dashboard</a>
     <a href="/" className="hover:text-gray-300">Logout</a>
    </div>
```

```
      <div className="flex">
        <input
          type="text"
          placeholder="Search"
          className="px-3 py-1 rounded-l border border-gray-400"
        />
        <button className="bg-[#0d1b2a] border border-gray-400 px-3
rounded-r">
          Search
        </button>
      </div>
    </nav>

    {/* Heading */}
    <div className="bg-blue-200 py-10 px-6">
      <h1 className="text-center text-2xl font-bold text-[#0d1b2a]
mb-8">
        All Donated Foods
      </h1>

      {/* Food Cards */}
      <div className="flex justify-center gap-8 flex-wrap">
        {donatedFoods.map((food) => (
          <div
            key={food.id}
            className="bg-white shadow-lg rounded-lg p-4 w-80"
          >
            <img
              src={food.image}
              alt={food.name}
              className="rounded-lg mb-4 w-full h-48 object-cover"
            />
            <h2 className="text-lg font-bold mb-2">
              Foods: {food.name}
            </h2>
            <p>Quantity: {food.quantity}</p>
            <p>Price: Rs.{food.price}</p>
```

```
            <p className="font-semibold text-green-600">
              Discounted Price: Rs.{food.discountedPrice}
            </p>
          </div>
        ))}
      </div>
    </div>
  </div>
 );
}
```

12) FOODS DONAR PAGE:



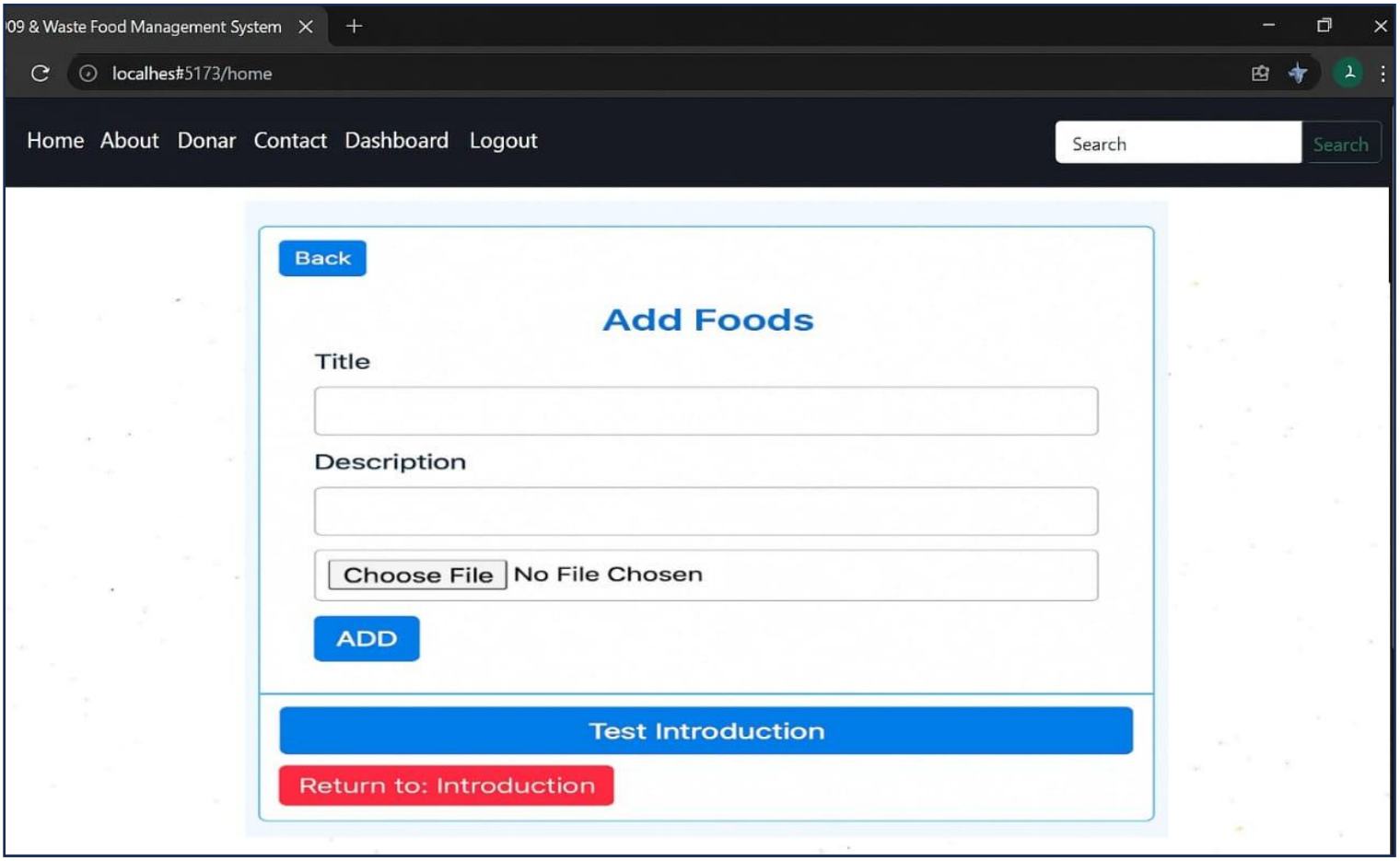
✓ **CODE:**

```
import React from "react";

export default function FoodCard() {
 return (
  <div style={{ fontFamily: "Arial, sans-serif" }}>
   {/* Page Title */}
```

```
      <h2 style={{ padding: "10px 0", textAlign: "left" }}>
        Waste Foods Management System Data
      </h2>

      {/* Food Image */}
      <div style={{ position: "relative", textAlign: "center" }}>
        <img
          src="https://your-image-link-here.jpg" // Replace with your
image path
          alt="Pizza, Burger & Cold Drinks"
          style={{
            width: "100%",
            maxHeight: "500px",
            objectFit: "cover",
            borderRadius: "8px"
          }}
        />

        {/* Caption Overlay */}
        <div
          style={{
            position: "absolute",
            bottom: "20px",
            left: "50%",
            transform: "translateX(-50%)",
            backgroundColor: "rgba(0, 0, 0, 0.85)",
            color: "white",
            padding: "10px 20px",
            borderRadius: "4px",
            textAlign: "center"
          }}
        >
          <div style={{ fontSize: "18px", fontWeight: "bold" }}>
            Pizza, Burger & Cold Drinks
          </div>
          <div style={{ fontSize: "14px" }}>Discounted Price: Rs.3599</div>
        </div>
```

```
    </div>
   </div>
  );
}
```

13) DONATIONS PAGE (for users):



✓ **CODE:**

```
import React, { useState } from "react";

export default function AddFoods() {
  const [title, setTitle] = useState("");
  const [description, setDescription] = useState("");
  const [file, setFile] = useState(null);

  const handleSubmit = (e) => {
```

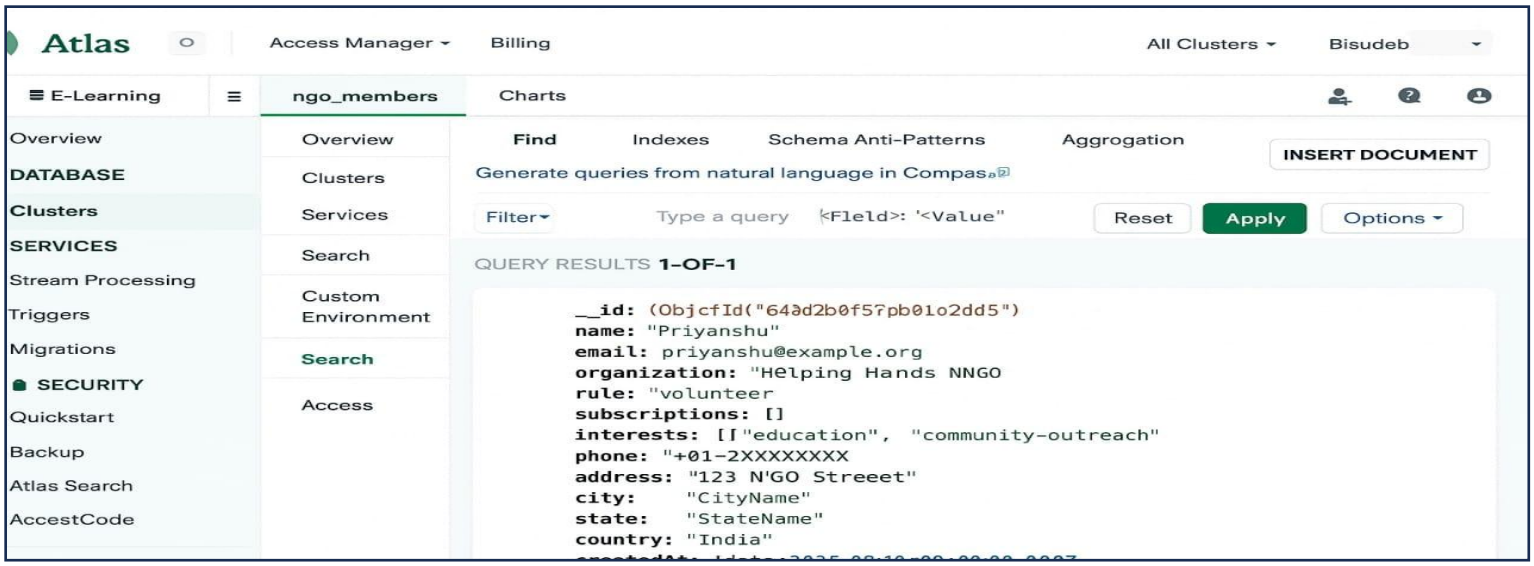
```
    e.preventDefault();
    // Handle form submission logic
    console.log({
      title,
      description,
      file
    });
  };

  return (
    <div className="min-h-screen bg-gray-100 flex flex-col items-
center">
      {/* Top Navigation */}
      <nav className="w-full bg-black text-white flex justify-between
items-center px-6 py-3">
        <ul className="flex gap-6">
          <li className="cursor-pointer hover:text-gray-300">Home</li>
          <li className="cursor-pointer hover:text-gray-300">About</li>
          <li className="cursor-pointer hover:text-gray-300">Donar</li>
          <li className="cursor-pointer hover:text-gray-
300">Contact</li>
          <li className="cursor-pointer hover:text-gray-
300">Dashboard</li>
          <li className="cursor-pointer hover:text-gray-300">Logout</li>
        </ul>
        <div className="flex">
          <input
            type="text"
            placeholder="Search"
            className="px-2 py-1 border rounded-l"
          />
          <button className="bg-green-600 px-3 rounded-
r">Search</button>
        </div>
      </nav>

      {/* Form Container */}
```

```
    <div className="bg-white border border-blue-300 rounded-lg p-6
mt-8 w-full max-w-lg shadow-md">
      <button className="bg-blue-500 text-white px-4 py-1 rounded
mb-4">
        Back
      </button>
      <h2 className="text-center text-2xl font-bold text-blue-500 mb-
4">
        Add Foods
      </h2>

      <form onSubmit={handleSubmit} className="space-y-4">
        <div>
          <label className="block font-medium">Title</label>
          <input
            type="text"
            value={title}
            onChange={(e) => setTitle(e.target.value)}
            className="border w-full p-2 rounded"
            required
          />
        </div>

        <div>
          <label className="block font-medium">Description</label>
          <input
            type="text"
            value={description}
            onChange={(e) => setDescription(e.target.value)}
            className="border w-full p-2 rounded"
            required
          />
        </div>

        <div>
          <input
            type="file"
```

```
            onChange={(e) => setFile(e.target.files[0])}
            className="block"
           />
          </div>

          <button
           type="submit"
           className="bg-blue-500 text-white px-6 py-2 rounded"
          >
           ADD
          </button>
         </form>

         <div className="mt-6 flex flex-col gap-2">
          <button className="bg-blue-500 text-white px-6 py-2
rounded">
           Test Introduction
          </button>
          <button className="bg-red-500 text-white px-6 py-2 rounded">
           Return to: Introduction
          </button>
         </div>
        </div>
       </div>
      );
     }
```

❖ **BACKEND:-**

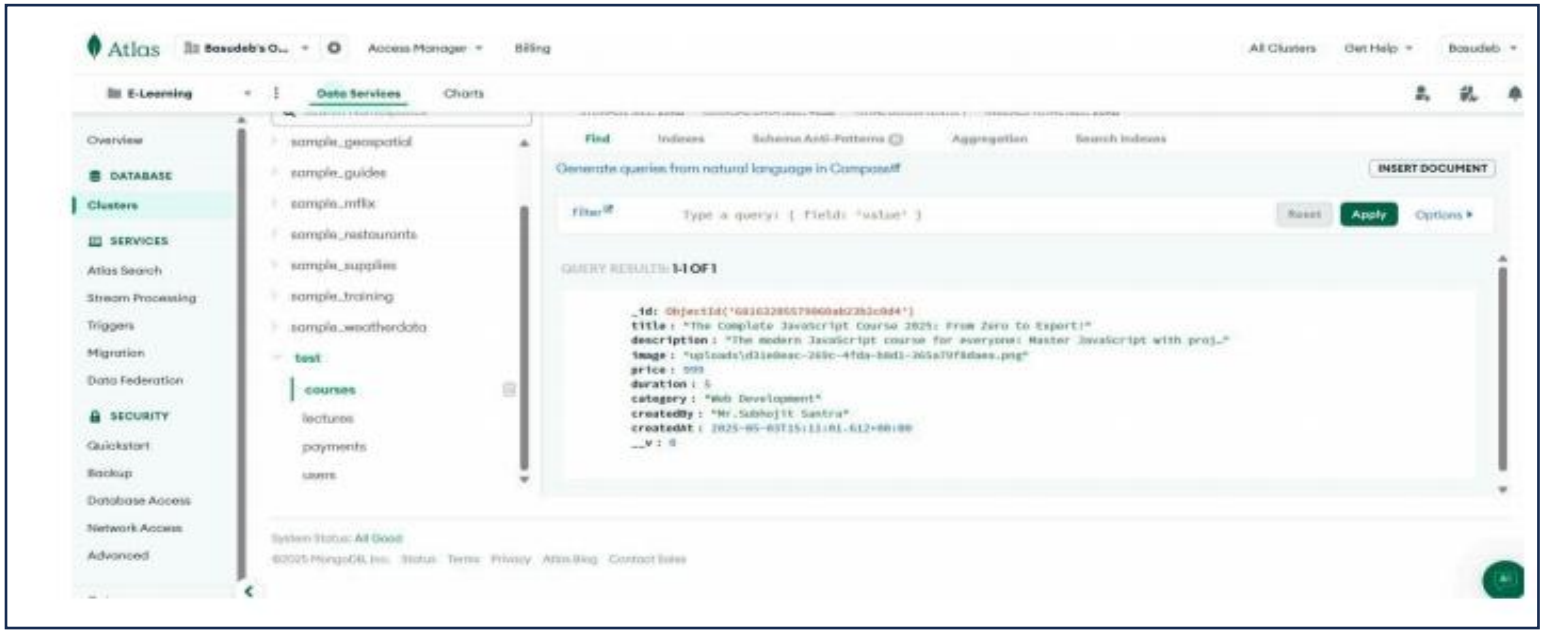
1) USER AND ADMIN DATA:



● Database - db.js:

```
import mongoose from "mongoose";
export const connectDB = async () => { try
{
    await mongoose.connect(process.env.DB);
    console.log("Database Connected");
  } catch (error) {
    console.log(error);
}
};
```

2) FOODS DATA:



● model – Foods.js :-

```
import mongoose from "mongoose";
const schema = new mongoose.Schema({
title: {
```

type: String, required:

true,

},

description: { type:

String, required:

true,

},

image: {

type: String, required:

true,

},

price: {

type: Number, required:

true,

},

duration: { type:

Number, required:

true,

},

category: { type:

String, required:

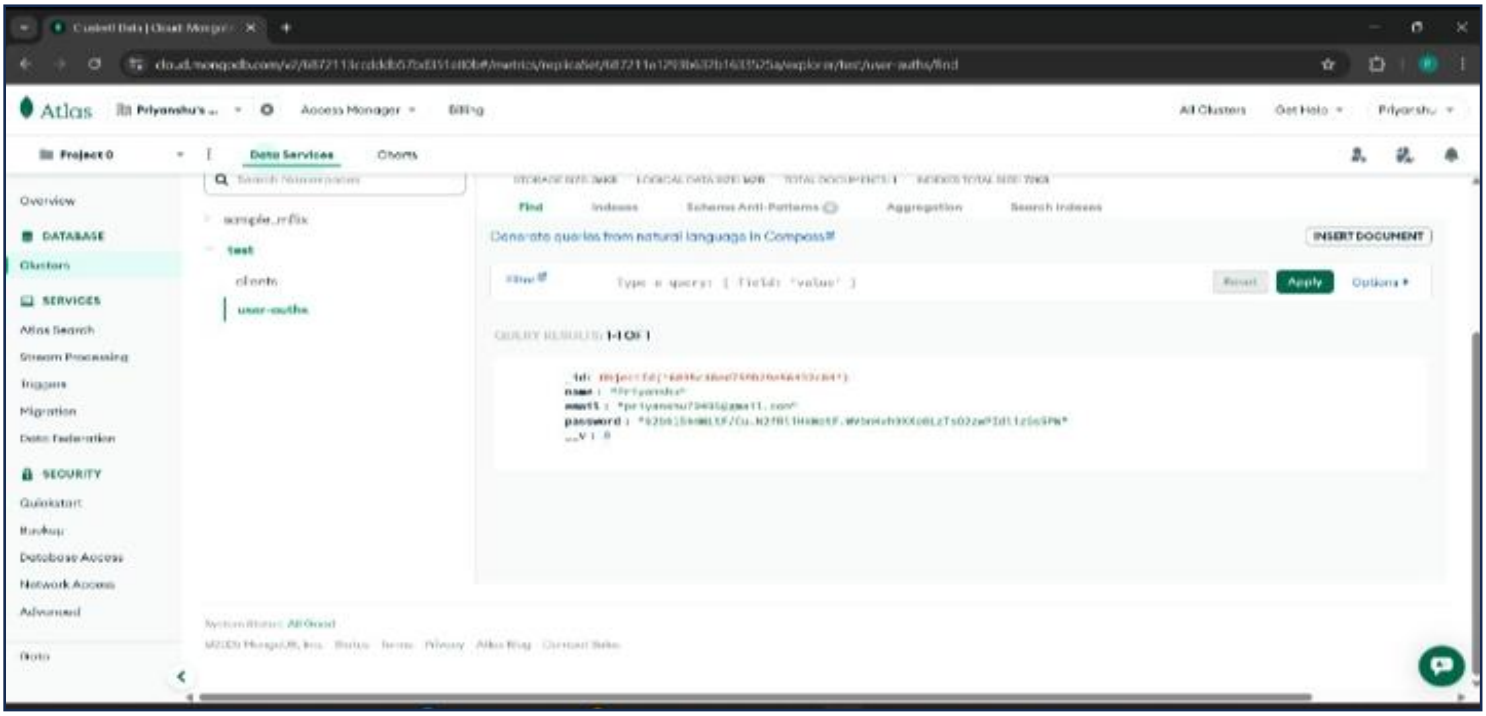
true,

},

createdBy: { type:

String, required:

3) PAYMENT DATA:



● model – Payment.js :-

```
import mongoose from "mongoose";
const schema = new mongoose.Schema({
razorpay_order_id: {
type: String, required:
true,
},
razorpay_payment_id: {
type: String, required:
true,
},
razorpay_signature: {
type: String, required:
true,
```

},

createdAt: { type:

Date,

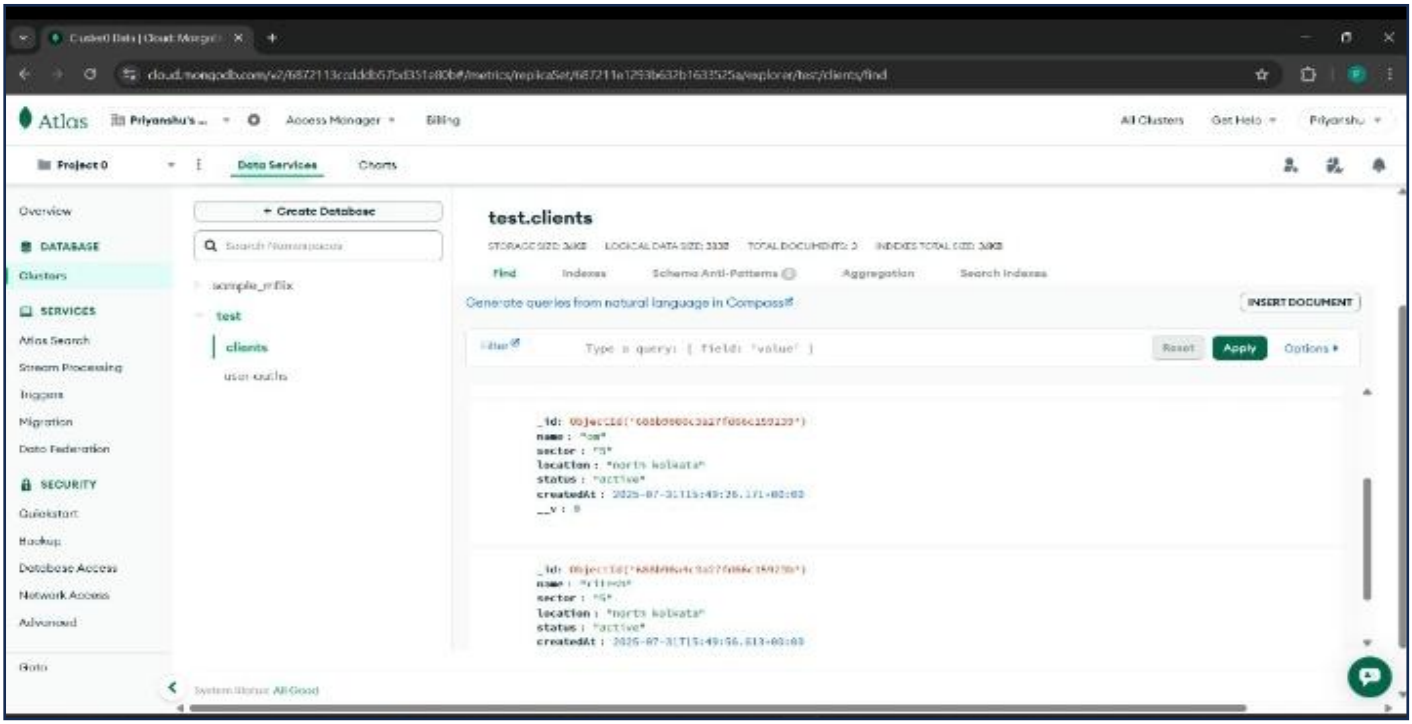
default: Date.now,

},

});

export const Payment = mongoose.model("Payment", schema);

- **DONATION DATA:**



● **model – Donate.js :-**

import mongoose from "mongoose";

const schema = new mongoose.Schema({

title: {

type: String,

required: true,

```
},
description: { type:
String, required:
true,
},
video: {
type: String,
required: true,
},
course: {
type: mongoose.Schema.Types.ObjectId,
ref: "Courses",
required: true,
},
createAt: { type:
Date,
default: Date.now,
},});
export const Lecture = mongoose.model("Lecture", schema);
```

6) CONCLUSION:

The NGO and Waste Food Management System serves as an innovative platform for addressing food wastage while promoting social welfare. By efficiently connecting food donors, NGOs, and recipients, it ensures that surplus food is redistributed to those in need rather than being wasted. Leveraging technology, the system streamlines food collection, storage, and distribution processes, making operations more transparent, efficient, and scalable. Features such as real-time tracking, donor-recipient matching, and reporting tools enhance accountability and trust among stakeholders. As global concerns over hunger and waste intensify, this system emerges as a practical and impactful solution, fostering community engagement and contributing toward a sustainable, hunger-free future.

7. FUTURE SCOPE & FURTHER ENHANCEMENTS

❖ Future scope:

The future of the NGO and Waste Food Management System holds significant potential to expand its reach, impact, and efficiency. Key areas for development include:

● **AI-Driven Demand Forecasting:** Implementing artificial intelligence to predict food demand patterns in specific regions, enabling better allocation and reducing wastage.

● **Blockchain-Based Transparency:** Using blockchain technology to record and track every stage of the food collection and distribution process, ensuring transparency, accountability, and donor trust.

● **IoT-Enabled Monitoring:** Integrating Internet of Things (IoT) sensors to monitor food quality, storage conditions, and expiry timelines in real time.

● **Geo-Location Optimization:** Enhancing route planning and delivery efficiency using GPS-based systems to minimize delays and ensure timely distribution.

● **Partnership Expansion:** Building networks with more NGOs, restaurants, supermarkets, and local communities to increase the volume of food rescued and redistributed.

● **Volunteer Engagement Platform:** Creating an in-app system for volunteer registration, task assignments, and tracking contributions to boost participation and coordination.

● **Awareness & Education Campaigns:** Adding features to promote food waste awareness, sustainable consumption habits, and community involvement through articles, videos, and workshops.

❖ Further Enhancements:

1. User Experience Enhancements:

● **Personalized Dashboards:** Display donation history, impact metrics (meals served, people helped), and upcoming volunteering opportunities.

● **Gamification:** Introduce badges, leaderboards, or recognition levels for donors, volunteers, and partner organizations to encourage engagement.

● **Dark Mode & Accessibility Features:** Improve usability for users with visual impairments or those accessing the platform in different lighting conditions.

2. Food Management & Logistics Tools:

● **Real-Time Food Quality Tracking:** Integrate IoT-based sensors to monitor storage temperature, expiry dates, and freshness of food.

● **Automated Matching System:** Match surplus food sources with the nearest NGOs or communities in need.

● **Expiry Alerts:** Notify donors and storage managers when food items are nearing expiry to prioritize distribution.

3. Community & Volunteer Features:

● **Volunteer Coordination System:** Assign roles, track activities, and schedule shifts through the app.

● **Awareness Campaigns:** Share educational materials, videos, and infographics about food waste reduction and sustainable practices.

● **Donor-Recipient Communication:** Provide a secure chat system for coordination between donors, NGOs, and delivery teams.

4. Analytics & Impact Tracking:

● **Impact Reports:** Generate detailed analytics showing total food saved, meals distributed, and environmental benefits (e.g., CO_2 reduction).

● **Predictive Demand Analysis:** Use AI to forecast upcoming needs in specific regions based on historical data and seasonal trends.

● **Route Optimization:** Track and optimize delivery routes for efficiency and reduced fuel consumption.

5. Tech Integrations:

● **Cloud Sync & Offline Mode:** Ensure uninterrupted access for NGOs operating in areas with poor connectivity.

● **Multilingual Support:** Enable operation in diverse communities by supporting multiple languages.

● **Integration with Donation Platforms:** Connect with existing crowdfunding and charity platforms to boost support and funding.

8. BIBLIOGRAPHY

1) www.w3schools.com

2)www.youtube.com

3)www.pexels.com

4)www.codepen.io

5)www.google.com

6)www.googlefont.com

7)www.react.our

8)www.codingworld.com