SURANA COLLEGE - AUTONOMOUS

Affiliated to Bangalore University

Recognized under 2(f) and 12 B of UGC, Certified by ISO, Re-accredited by NAAC with A+

No 16, South End Road, Bengaluru-560 004

WEB PROGRAMMING REPORT

Project Title: CRUD Operations on Food Donation System

Course : Bachelor of Computer Applications (BCA)

Submitted By –

GAGAN V - U03ME23S0033 (5th Semester BCA 'A'),

VIKAS S- U03ME23S0028(5th Semester BCA 'A')

Submitted To –

SURANA COLLEGE – AUTONOMOUS

In partial fulfilment for the curriculum requirement for

BACHELOR OF COMPUTER APPLICATIONS

Date Of Submission : 4th November 2025

DECLARATION

GAGAN V
NOVEMBER 2025

BCA

I**, GAGAN V** hereby declare that the internship report, entitled **"CRUD OPERATIONS ON FOOD DONATION SYSTEM"** submitted to **SURANA COLLEGE - AUTONOMOUS,** in partial fulfilment for the curriculum requirement for **BACHELOR OF COMPUTER APPLICATIONS** done by me during 2025-2026 under the guidance of **Prof. Vidya. A ,** Head Of the Department , Department of Computer Science, SURANA COLLEGE - AUTONOMOUS, Bengaluru. It has not formed the basis for the award any Degree/Diploma/Associateship/Fellowship or other similar title to any candidate in any College/University.

GAGAN V

DECLARATION

VIKAS S
NOVEMBER 2025

BCA

I**, VIKAS S** hereby declare that the internship report, entitled **"CRUD OPERATIONS ON FOOD DONATION SYSTEM"** submitted to **SURANA COLLEGE - AUTONOMOUS,** in partial fulfilment for the curriculum requirement for **BACHELOR OF COMPUTER APPLICATIONS** done by me during 2025-2026 under the guidance of **Prof. Vidya. A ,** Head Of the Department, Department of Computer Science, SURANA COLLEGE - AUTONOMOUS, Bengaluru. It has not formed the basis for the award any Degree/Diploma/Associateship/Fellowship or other similar title to any candidate in any College/University.

VIKAS S

ACKNOWLEDGEMENT

We are extremely delighted and grateful to thank every single person who helped me in completing my dissertation successfully. I thank the almighty for bestowing me with his blessings. I'm very much indebted and extend my deep sense of gratitude to **Prof. Vidya.A ,** Head Of the Department , Department of Computer Science, internship guide, for her consistent words of motivation, encouragement and filling me completely with insight regarding the topic throughout the period of my study. She was present whenever I needed her the most during the completion of this project.

My sincere thanks are to the faculty members of Department of Computer Science, for their encouragement in my academic endeavors.

I thank my parents and for being very liberal towards me and for having complete trust in me during this project.

GAGAN V

VIKAS S

TABLE OF CONTENTS

CHAPTER 1: ABSTRACT

The project titled **"CRUD Operations on Food Donation System"** aims to transform the complex, real-world challenges of food waste and food insecurity into a secure, efficient, and user-friendly online system. In an era where logistical efficiency is key to sustainability, this project provides a reliable solution by combining modern web technologies with database-driven management. The system allows food donors (Hotels) to manage their surplus food listings , and food collectors (NGOs) to easily log in, view available donations, and claim items in a secure, real-time environment.

The system is developed using HTML, CSS, and JavaScript for the frontend interface, ensuring an intuitive and responsive design. The backend is powered by PHP and MySQL, executed through the XAMPP server environment, which provides a complete development and testing framework. The integration of these technologies enables smooth CRUD (Create, Read, Update, Delete) operations, allowing effective management of user profiles , donation listings , and collection data. The project architecture consists of two primary modules — 'HOTEL' (Donor) and 'NGO' (Collector) — each performing specific roles to ensure organized and efficient functioning.

Security is a key focus of this project. Features such as session-based authentication , password protection , and strict role-based access control are implemented to ensure that only authorized users can perform specific operations. The system also prevents data conflicts, such as an NGO claiming an already "Picked Up" item or a donor editing another user's listing , to safeguard data integrity. Furthermore, the interface has been designed with a clean, responsive, and card-based layout with clear navigation, enhancing usability for both technical and non-technical users.

The expected outcome of the Food Donation System is to provide a complete working model that demonstrates how technology can bridge the gap between food surplus and food scarcity. It showcases the potential of a web application to simplify logistical workloads, minimize food spoilage through time-sensitive listings , and improve the speed and efficiency of getting edible food to those in need. This project not only helps in understanding full-stack web development concepts but also highlights the importance of database connectivity, backend logic, and secure authentication mechanisms in real-world applications. The system can be further enhanced in

the future by integrating an Admin verification panel , real-time notifications for new donations , and blockchain technology for enhanced transparency.

CHAPTER 2: INTRODUCTION

The "CRUD Operations on Food Donation System" is a web-based application designed to digitize and simplify the complex, traditional logistics of surplus food redistribution. The project focuses on creating a reliable and efficient online platform that ensures traceability, speed, and accessibility in connecting food donors with those in need. It eliminates the limitations of manual coordination—such as inefficient phone calls, data fragmentation, and food spoilage due to delays—by introducing an automated system that can be managed easily by donors and used conveniently by collectors.

2.1 Overview of the Project

The Food Donation System is a web-based platform designed to modernize and digitalize the traditional process of food rescue. It provides an automated way to manage surplus food listings, collector details, and donation pickups through an interactive and user-friendly interface. The system aims to replace inefficient, ad-hoc manual coordination with a secure, efficient, and accessible online method, reducing food waste and improving the reliability of food reaching vulnerable populations.

The project is divided into two main modules — the **'HOTEL' (Donor) module** and the **'NGO' (Collector) module**. The 'HOTEL' module allows donors to manage their operations, such as creating, updating, and deleting their own food donation listings. The 'NGO' module enables verified organizations to log in securely, view a real-time list of all available donations, and "Pick Up" (claim) items digitally. Developed using HTML, CSS, and JavaScript for the frontend and PHP with MySQL for the backend , the project integrates all the essential features needed for transparent and timely food rescue management.

2.2 Importance of CRUD Operations in Applications

CRUD operations — **Create, Read, Update, and Delete** — form the foundation of any database-driven application. In the context of the Food Donation System, CRUD operations play a crucial role in maintaining and managing all platform data. For example, the **Create** operation is used when a 'HOTEL' user posts a new donation listing or a new user registers an

account. The **Read** operation retrieves data to display all available items on the public list and on the private dashboards for both donors and collectors. The **Update** operation allows a 'HOTEL' to modify their existing listing details or an 'NGO' to claim a donation, which updates its status to 'Picked Up'. Finally, the **Delete** operation enables a 'HOTEL' to remove an expired or unavailable listing.

Implementing CRUD operations ensures the system remains dynamic, flexible, and easy to maintain. It allows real-time data manipulation while maintaining the accuracy and integrity of stored information. Through the use of PHP and MySQL, these operations are efficiently handled via SQL queries , providing seamless interaction between the web interface and the database. Understanding CRUD functionality also deepens the developer's grasp of core programming concepts and database management techniques essential for modern web applications.

2.3 Aim and Scope of the Project

The primary aim of this project is to develop a secure and scalable Food Donation System that demonstrates how technology can simplify and enhance the food rescue process. It seeks to provide an efficient solution that ensures traceability, prevents food waste by managing time-sensitive donations , and minimizes the logistical friction between donors and collectors. The project also aims to promote trust in the platform by implementing proper authentication mechanisms , role-based access control , and a structured, auditable workflow.

The scope of this project serves as a foundational model for a real-world community service platform. With further development, it can be scaled to a larger implementation by integrating advanced features such as an **Admin verification panel** to vet NGOs , **real-time notifications** for new listings , and **blockchain technology** for immutable donation tracking. The system serves as a learning model for understanding the complete lifecycle of a web application—from front-end design and server-side scripting to database handling and system deployment—making it both educationally valuable and practically relevant.

CHAPTER 3: OBJECTIVES

The "CRUD Operations on Food Donation System" is designed with the primary goal of creating a secure, reliable, and efficient digital platform to bridge the gap between food surplus and food scarcity. The system replaces the conventional, inefficient manual coordination of donations (such as phone calls and fragmented records) with a computer-based solution that ensures data accuracy, speed, and traceability throughout the entire food rescue lifecycle. It focuses on integrating essential CRUD (Create, Read, Update, Delete) functionalities to manage donors (Hotels), collectors (NGOs), and donation data effectively within a well-structured database. This project aims to simplify the donation-listing process for donors while providing collectors with complete, real-time visibility of available food items.

Through this project, the focus is not only on building a functional system but also on understanding the underlying concepts of full-stack web development, database connectivity, and role-based authentication mechanisms. The system is developed using HTML, CSS, and JavaScript for the frontend interface and PHP with MySQL for backend operations. Together, these technologies enable the implementation of role-specific CRUD operations , secure user authentication , and real-time data handling, ensuring both performance and security.

The specific objectives of this project are as follows:

- To design an interactive, user-friendly, and responsive interface that simplifies navigation for both 'HOTEL' (Donors) and 'NGO' (Collectors), ensuring smooth access across various devices.
- To implement Create, Read, Update, and Delete (CRUD) operations effectively, allowing 'HOTEL' users to manage their donation listings and 'NGO' users to manage their profiles and claim available donations.
- To establish secure login and authentication mechanisms that protect user credentials and restrict system access based on specific user roles ('HOTEL' or 'NGO').
- To integrate a robust MySQL database for efficient data storage, retrieval, and management, ensuring consistency and reliability of all user and donation information.

- To automate the donation listing and claiming process, minimizing manual coordination and reducing food waste by effectively managing time-sensitive data (expiry times).
- To develop two distinct, role-based dashboards: a 'HOTEL' dashboard for full CRUD management of their *own* listings , and an 'NGO' dashboard to 'Read' all available donations and 'Update' an item's status to 'Picked Up'.
- To promote traceability and trust in the food rescue process by ensuring every donation is accurately recorded and that the system prevents data conflicts, such as two NGOs claiming the same item.
- To enhance technical awareness by showcasing how modern web technologies can transform traditional logistical challenges into smart, data-driven platforms for social good.
- To gain practical experience in full-stack web development by integrating client-side design , server-side logic , and database operations into a single working model.
- To create a scalable and extendable platform that can be further improved by integrating advanced features like an Admin verification panel , real-time notifications , or blockchain for transparency in future iterations.

In essence, this project not only demonstrates the use of web programming tools but also emphasizes the importance of system design, data integrity, and user experience. It contributes to the understanding of how CRUD-based web applications can streamline real-world logistical operations while ensuring traceability, efficiency, and technological advancement in modern food rescue systems.

CHAPTER 4: SYSTEM REQUIREMENTS

The successful development and execution of the "CRUD Operations on Food Donation System" depend on specific hardware and software configurations. These requirements ensure smooth performance, reliable database connectivity, and proper execution of all web functionalities. Since the system is built using PHP and MySQL within a XAMPP environment , it can be implemented efficiently on any standard computer system with moderate specifications.

4.1 Hardware Requirements

To ensure the system functions smoothly during development, testing, and execution, the following hardware specifications are recommended:

- **Processor:** Intel Core i3 / AMD Ryzen 3 (or equivalent)
- **RAM:** Minimum 8 GB (Recommended for multitasking, running the local server, and browser testing)
- **Storage:** At least 20 GB of free disk space for project files, XAMPP installation, and database storage.
- **Hard Disk:** 256 GB SSD (Solid State Drive) preferred for better performance.
- **Display:** 1024 × 768 resolution or higher for proper interface visualization.
- **Input Devices:** Standard keyboard and mouse for navigation and data entry.
- **Network Connectivity:** Stable internet or local network connection for hosting and accessing the application during testing.

The above configuration is sufficient for developing and running the web-based application in a local or institutional environment. For large-scale implementation, higher specifications and dedicated server systems are recommended.

4.2 Software Requirements

The Food Donation System is developed using open-source technologies , making it cost-effective and accessible to students and institutions. The following software setup is required for its complete functionality:

- **Operating System:** Windows 10 or later / macOS / Linux (Ubuntu preferred for open-source environment).
- **Development Environment:** Visual Studio Code, PhpStorm, or Sublime Text (for editing HTML, CSS, PHP, and JavaScript files).
- **Local Server Package:** XAMPP (includes Apache, PHP, and MySQL) for local hosting and database management.
- **Backend Technology:** PHP 8.0+ for server-side scripting and logic execution.
- **Database:** MySQL (5.7+ or MariaDB 10.0+) for storing and retrieving user, profile, and donation data.
- **Frontend Technologies:** HTML5, CSS3, and JavaScript for user interface design and interactivity.
- **Browser:** Google Chrome, Mozilla Firefox, or Microsoft Edge for running and testing the application.
- **Version Control (Optional):** Git/GitHub for project version tracking and collaboration.

These software components ensure that the project runs efficiently, allowing the developer to design, test, and deploy all modules—'HOTEL' (Donor) and 'NGO' (Collector) —within a controlled and secure environment.

CHAPTER 5: SYSTEM DESIGN

System design is a crucial stage in the development process that focuses on defining the architecture, data flow, and interaction between various components of the system. It provides a clear blueprint for how the "CRUD Operations on Food Donation System" operates, ensuring logical connections between users (Donors and Collectors), processes (donation and collection), and the database.

The design phase involves creating Entity-Relationship (ER) diagrams, Database Schemas, and Use Case Diagrams to represent the internal structure and functional behavior of the system.

5.1 Entity-Relationship (ER) Diagram

The Entity-Relationship Diagram represents the logical relationships between different entities involved in the Food Donation System. It illustrates how data such as users, donor profiles, collector profiles, and the donations themselves are connected within the system database.

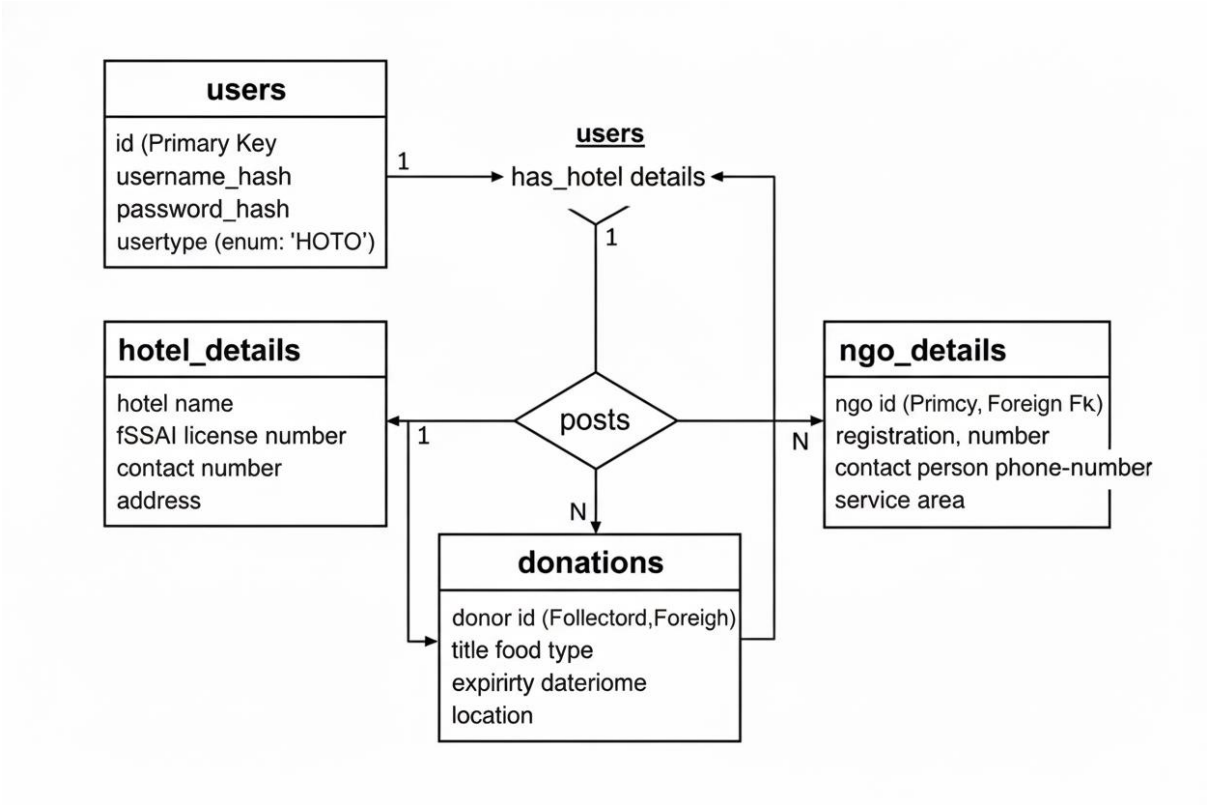

Fig 5.1: ER Diagram

Entities and Attributes

1. **users – id (Primary Key), username, password_hash, user_type (enum: 'HOTEL', 'NGO').**
2. **hotel_details – hotel_id (Primary Key, Foreign Key), hotel_name, fssai_license_number, contact_person, phone_number, address.**
3. **ngo_details – ngo_id (Primary Key, Foreign Key), ngo_name, registration_number, contact_person, phone_number, service_area.**
4. **donations – id (Primary Key), donor_id (Foreign Key), collector_id (Foreign Key), title, food_type, quantity, expiry_datetime, location, status.**

Relationships

- **One users record (where user_type='HOTEL') has exactly one hotel_details record (One-to-One).**
- **One users record (where user_type='NGO') has exactly one ngo_details record (One-to-One).**
- **One users (Donor) can post many donations (One-to-Many).**
- **One users (Collector) can pick up many donations (One-to-Many).**

Explanation

The ER diagram defines the logical flow of information. The **`users`** table is the central entity for authentication. The **`hotel_details`** and **`ngo_details`** tables are extensions (profiles) linked directly to a user ID. The **`donations`** table is the primary transactional entity, linking one donor (via `donor_id`) and one optional collector (via `collector_id`) to a specific food listing

.

5.2 Database Schema

The Database Schema defines the structure of the database, including the names of tables, fields, data types, and relationships. It ensures organized storage and efficient retrieval of data required for the functioning of the system**.**

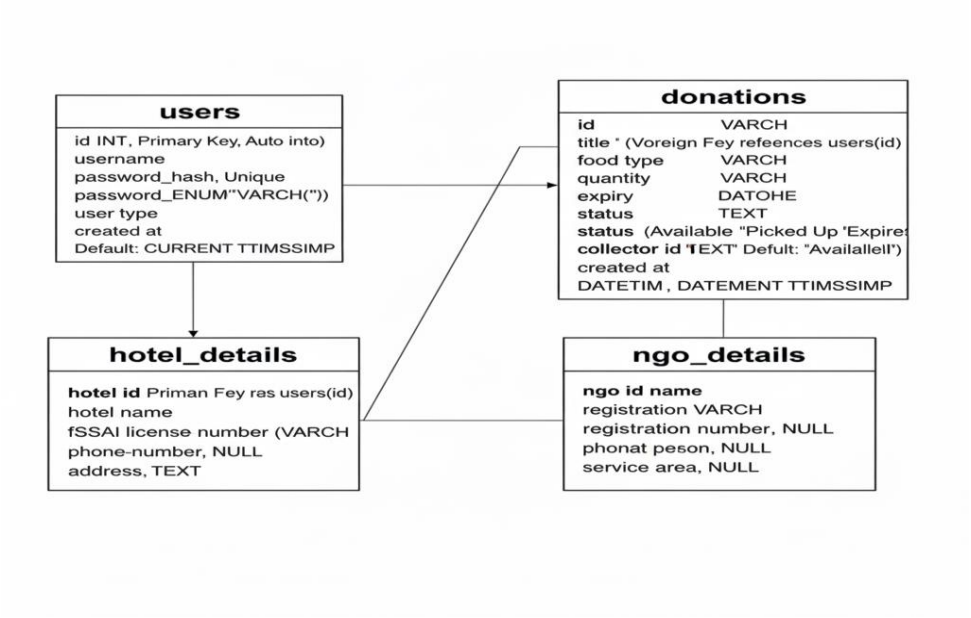

Fig 5.2: Database Schema

Table Descriptions

1. **users**
    - id (INT, Primary Key, Auto Increment)
    - username (VARCHAR, Unique)
    - password_hash (VARCHAR)
    - user_type (ENUM('NGO','HOTEL'))
    - created_at (DATETIME, Default: CURRENT_TIMESTAMP)
2. **donations**
    - id (INT, Primary Key, Auto Increment)
    - donor_id (INT, Foreign Key references users(id))

- title (VARCHAR)
- food_type (VARCHAR)
- quantity (VARCHAR)
- expiry_datetime (DATETIME)
- location (TEXT)
- status (ENUM('Available','Picked Up','Expired'), Default: 'Available')
- collector_id (INT, Foreign Key references users(id), NULL)
- created_at (DATETIME, Default: CURRENT_TIMESTAMP)

3. **hotel_details**
   - hotel_id (INT, Primary Key, Foreign Key references users(id))
   - hotel_name (VARCHAR)
   - fssai_license_number (VARCHAR, NULL)
   - contact_person (VARCHAR)
   - phone_number (VARCHAR, NULL)
   - address (TEXT, NULL)
4. **ngo_details**
   - ngo_id (INT, Primary Key, Foreign Key references users(id))
   - ngo_name (VARCHAR)
   - registration_number (VARCHAR, NULL)
   - contact_person (VARCHAR)
   - phone_number (VARCHAR, NULL)
   - service_area (TEXT, NULL)

Explanation:

The schema is designed for clarity and role segregation. The users table handles authentication. The donations table tracks the complete lifecycle of a food item. The hotel_details and ngo_details tables store specific profile information, which is required to be filled out before the user can access their respective dashboards.

5.3 Use Case Diagrams

1) Use Case Diagram 1

The Use Case Diagram visually represents the interaction between users and the system. It identifies all possible actions that each actor (user type) can perform.

.

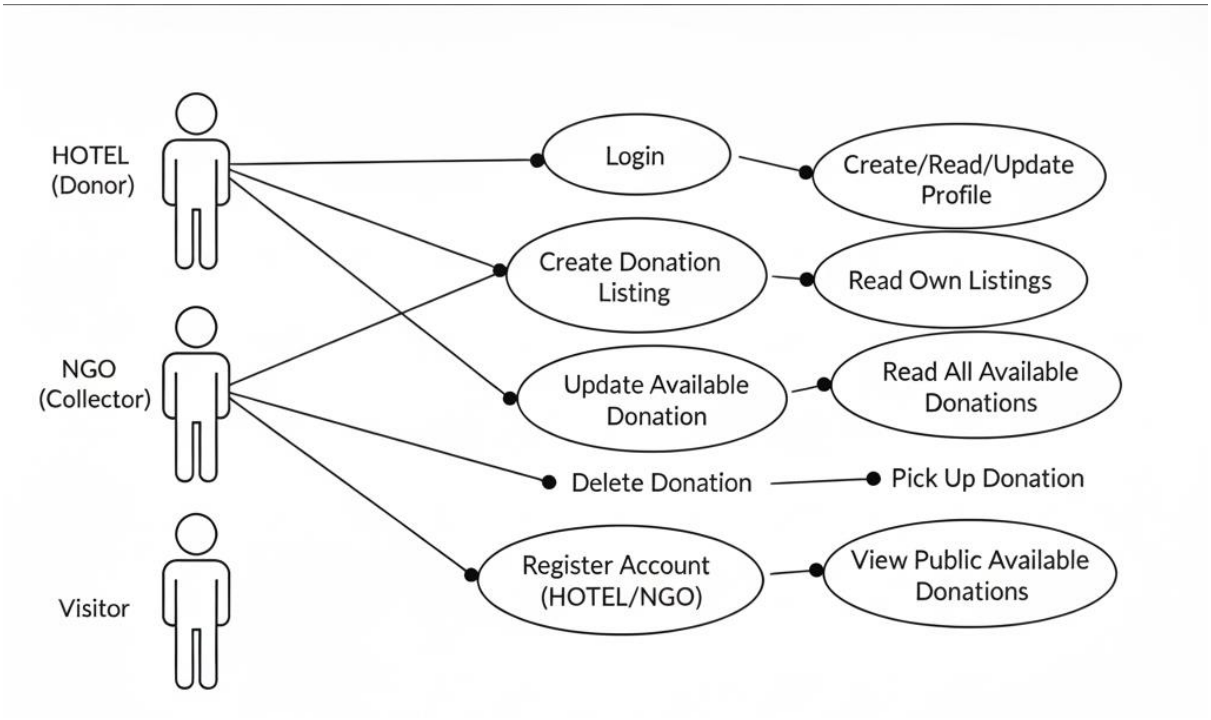

Fig 5.3.1: Use Case Diagram 1

Actors:

- 'HOTEL' (Donor)
- 'NGO' (Collector)
- Visitor (Unauthenticated User)

'HOTEL' (Donor) Use Cases:

- Login securely to the hotel dashboard.
- Create, Read, and Update profile (hotel_details**).**
- Create new donation listing.
- Read own donation listings.
- Update existing 'Available' donation.
- Delete own 'Available' or 'Expired' donation.

'NGO' (Collector) Use Cases:

- Login securely to the NGO dashboard.
- Create, Read, and Update profile (ngo_details).
- Read all 'Available' donations from all donors.
- "Pick Up" Donation (Performs an Update on the donation's status and collector_id).

Visitor Use Cases:

- Register a new 'HOTEL' or 'NGO' account.
- View public list of 'Available' donations.

Explanation:

The Use Case Diagram demonstrates system behavior from a user's perspective, ensuring all functionalities are mapped properly. It also highlights the strict role-based access structure—Donors manage their own content , while Collectors can view all public content and claim it.

2) Use Case Diagram 2

The Use Case Diagram for the Food Donation System illustrates how the two main actors — 'HOTEL' (Donor) and 'NGO' (Collector) — interact with the system's core functionalities. It visually represents the various CRUD operations provided by the system to each actor, ensuring a clear understanding of their roles and the flow of activities.

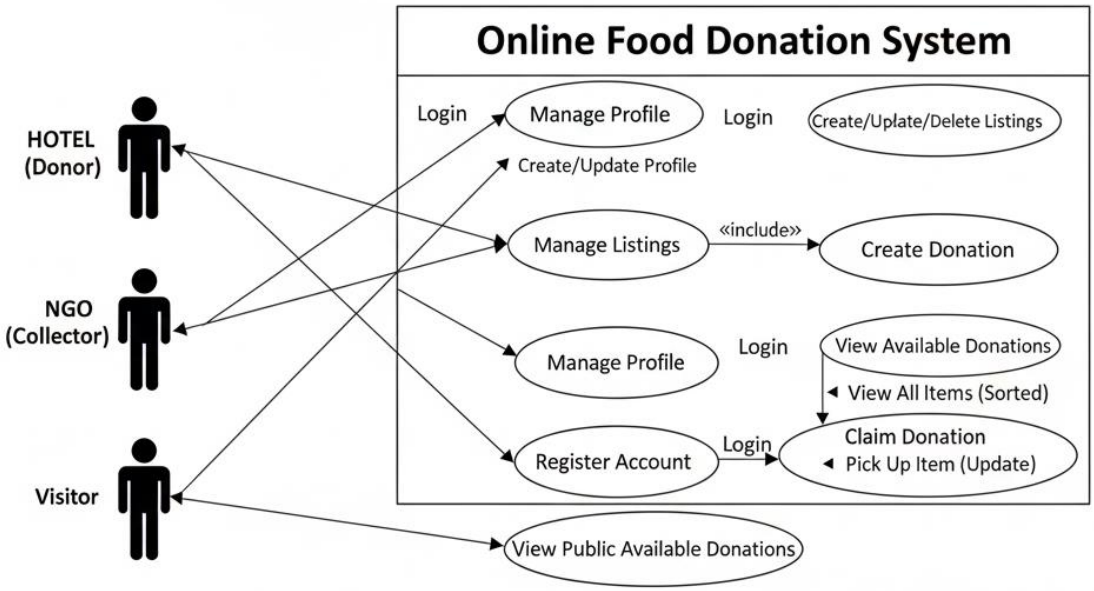

Fig 5.3.2: Use Case Diagram 2

Actors Involved

1. 'HOTEL' (Donor): The donor is responsible for managing their profile and posting surplus food listings. The donor performs the full range of CRUD operations on the donations they own.
2. 'NGO' (Collector): The collector is the verified end user who interacts with the system to claim donations. The collector primarily performs 'Read' and 'Update' operations on the public pool of donations**.**

'HOTEL' (Donor) Use Cases

- Login: The donor logs in to the system using valid credentials to gain access to their dashboard.
- Manage Profile: This includes Create and Update operations on their hotel_details record.
- Create Donation: The donor adds a new record to the donations table, including title, quantity, and expiry time.

- Manage Listings: This includes Read (view their own listings), Update (edit 'Available' listings) , and Delete (remove 'Available' or 'Expired' listings).

'NGO' (Collector) Use Cases

- Register: The collector creates a new 'NGO' account.
- Login: After registration, the collector logs in to access the donation dashboard.
- Manage Profile: This includes Create and Update operations on their ngo_details record.
- View Available Donations: The collector performs a Read operation to see all 'Available' donations, sorted by urgency.
- Claim Donation: Once authenticated, the collector can "Pick Up" a donation. This is an Update operation that changes the donation's status to 'Picked Up' and assigns their user_id to the collector_id field**.**

Use Case Flow Explanation

The interaction begins when a 'HOTEL' or 'NGO' Registers and Logs In. Both must complete their profiles. The 'HOTEL' user then Creates, Updates, or Deletes donation listings from their private dashboard. The 'NGO' user logs in, Reads the list of all available items , and performs an Update ("Pick Up") to claim one. This structure ensures that all functionalities are divided logically between both actors, maintaining data integrity, security, and role-based access throughout the system.

.

CHAPTER 6: IMPLEMENTATION

The implementation phase is the most crucial part of the "CRUD Operations on Food Donation System," where all the system designs and plans are transformed into a fully functional web application. The system has been implemented using HTML, CSS, and JavaScript for frontend development and PHP with MySQL for backend integration. The development environment used is XAMPP, which provides an Apache server and MySQL database support for local hosting and testing.

The system is modularized into two main sections: the **'HOTEL' (Donor) Module** and the **'NGO' (Collector) Module** , each performing specific tasks as designed in the earlier system design phase. Both modules interact with the database to perform essential CRUD (Create, Read, Update, Delete) operations, ensuring that data is dynamically managed and updated in real time.

6.1 Create Operation:

The **Create** operation is used to insert new data into the database. It allows a 'HOTEL' user to add a new food donation listing and allows both 'HOTEL' and 'NGO' users to register for a new account. This process involves input forms where the user enters details. Once submitted, PHP scripts validate and store the data into the respective MySQL tables.

Example Process:

- 'HOTEL' user enters donation details (title, quantity, expiry time, etc.) → Submits form → PHP script executes INSERT INTO query → Data saved in donations table.
- Similarly, when a new user registers, data is added to the users table.

Sample SQL Query:

```
INSERT INTO donations (donor_id, title, food_type, quantity, expiry_datetime, location, status)VALUES (?,?,?,?,?,?, 'Available');
```

This ensures that new records are dynamically created in the database whenever an authorized user performs a valid creation action.

6.2 Read Operation:

The Read operation retrieves and displays data from the database for both role-based dashboards and public-facing pages.

- The 'NGO' Module reads information such as the list of all 'Available' donations from all hotels.
- The 'HOTEL' Module reads a list of *only* the donations they have personally posted.
- The Public Page (view_donations.php) reads all 'Available' items for any visitor to see.
- Example Process:
  - 'NGO' user logs in and opens their dashboard → PHP executes a SELECT query on the donations table → Data is fetched and displayed in a tabular format, ordered by the soonest expiry time.

Sample SQL Query:

```
SELECT * FROM donations;
```

This allows both users and visitors to view stored data efficiently, promoting the system's goal of connecting surplus food with those who need it..

6.3 Update Operation:

The Update operation enables users to modify existing data. This functionality is implemented in two distinct ways:

1. Data Editing: A 'HOTEL' user can modify the details of their *own* 'Available' donations (e.g., change the quantity or location).
2. Status Change: An 'NGO' user can "Pick Up" a donation, which executes an UPDATE query to change the item's status from 'Available' to 'Picked Up' and assigns their user_id to the collector_id field.

- Example Process:
    - 'NGO' user clicks "Pick Up" on an item → PHP executes an UPDATE query → The status and collector_id fields are changed in the donations table, removing the item from the available list**.**

Sample SQL Query:

```
UPDATE donations
SET status = 'Picked Up', collector_id =?
WHERE id =? AND status = 'Available';
```

Through this feature, the system maintains data integrity and allows for real-time tracking of the donation lifecycle.

6.4 Delete Operation:

The **Delete** operation allows for the removal of data from the database. This action is rightly restricted to 'HOTEL' users, who can only delete their *own* donation listings. This operation ensures a clean and efficient database, allowing donors to remove items that are no longer valid.

- **Example Process:**
    - 'HOTEL' user selects their own donation record → Confirms deletion → PHP script verifies ownership and status ('Available' or 'Expired') → PHP executes a `DELETE FROM` query → Record removed from the `donations` table

.

Sample SQL Query:

```
DELETE FROM donations WHERE id =? AND donor_id =?;
```

Deletion is handled carefully with server-side authorization to prevent users from deleting data that does not belong to them or that has already been 'Picked Up'..

6.5 Integration and Workflow:

Once the CRUD functionalities are implemented, all modules are integrated into a cohesive workflow:

1. User Registration: A 'HOTEL' or 'NGO' user is created in the users table.
2. Profile Management: The user logs in and must Create or Update their profile in hotel_details or ngo_details before proceeding.
3. Donation Process: A 'HOTEL' user Creates a new listing in the donations table.
4. Collection Process: An 'NGO' user Reads the list of available donations and uses the Update operation to claim one, changing its status.
5. Lifecycle Management: The 'HOTEL' user can Update or Delete their listings as needed.

All database interactions are executed using PHP's mysqli with prepared statements to prevent SQL injection. Proper validation and session management are implemented to ensure security and enforce role-based permissions.

6.6 Summary:

The implementation phase successfully converts the system design into a functioning web application. CRUD operations form the backbone of data management, while the combination of frontend and backend technologies ensures a seamless and interactive user experience. This stage demonstrates the real-world application of web development concepts and database handling, resulting in a traceable, reliable, and efficient Food Donation System that fulfills its intended objectives**.**

CHAPTER 7: RESULTS AND OUTPUT

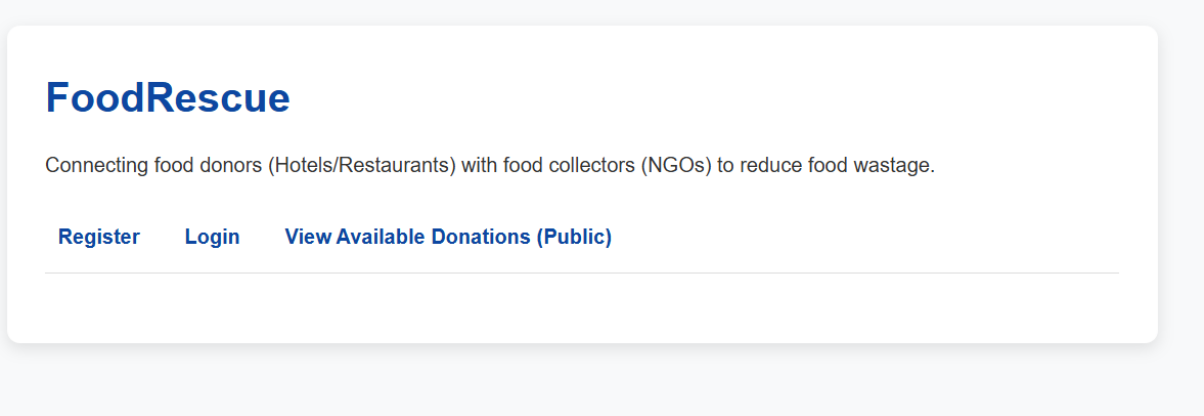

Fig 7.1: Home Page

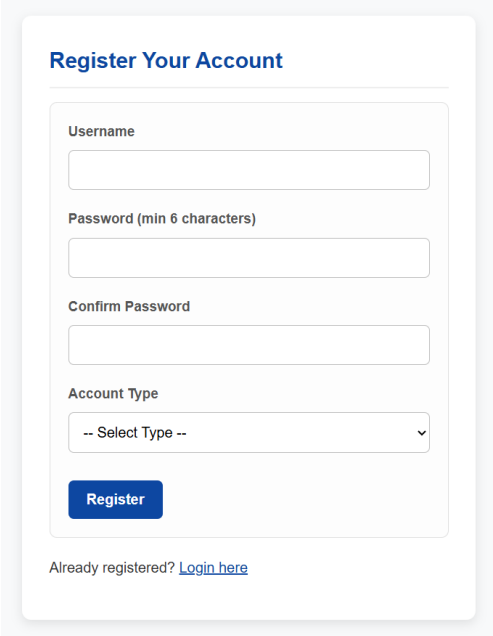

Fig 7.2: Registration Page

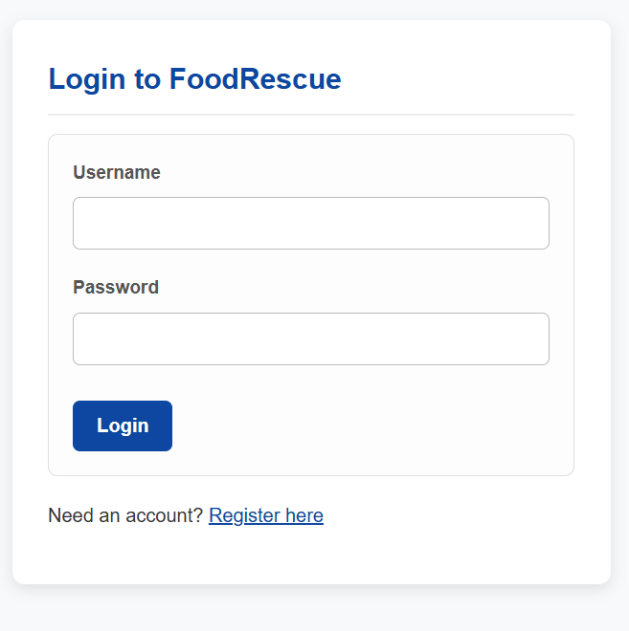

Fig 7.3: Login Page

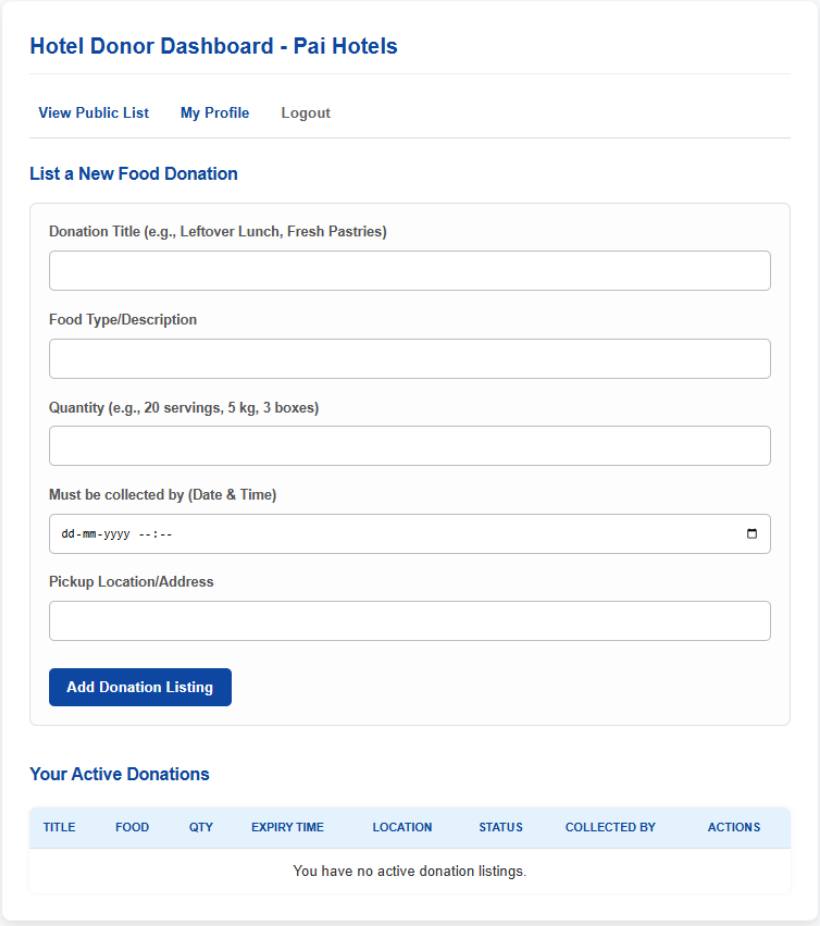

Fig 7.4: Hotel (Donor) Dashboard

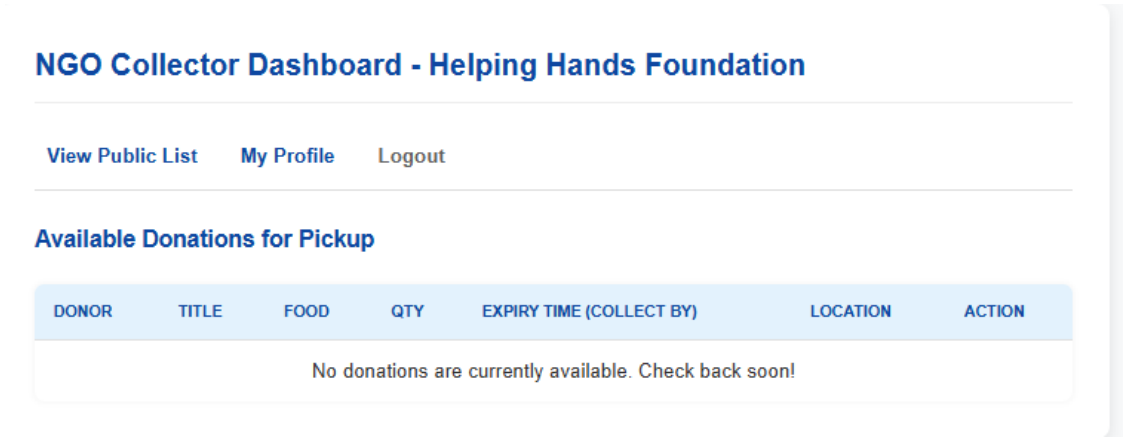

Fig 7.5: NGO (Collector) Dashboard

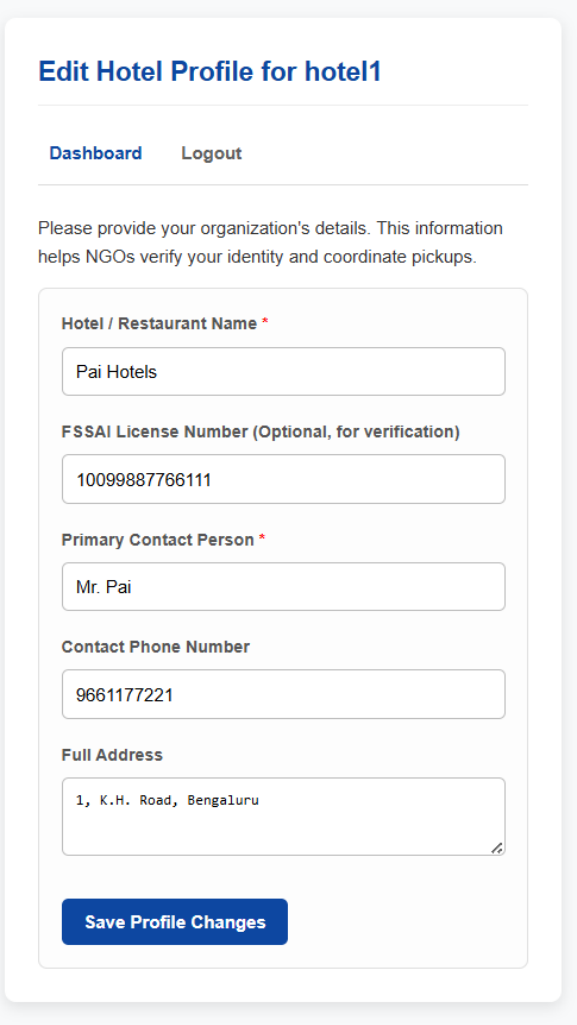

Fig 7.6: Edit Donation Page

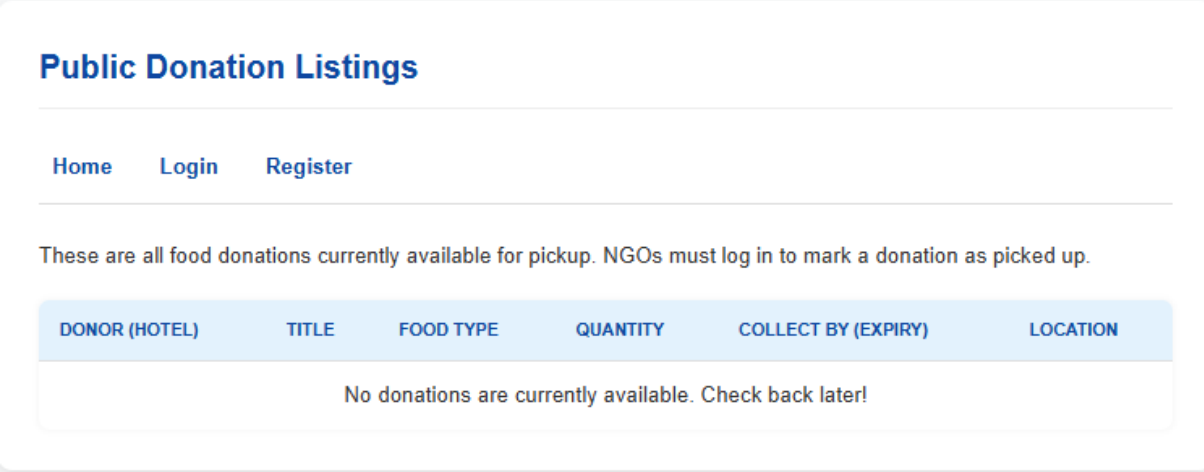

Fig 7.8: Public Donation List

The "Food Donation System" works by connecting users with two distinct roles—'HOTEL' (Donor) and 'NGO' (Collector)—through a secure web interface linked to a centralized database. The system operates in two main modules, each performing specific functions:

1. **User Registration:** New users (both Hotels and NGOs) register by entering a username, password, and selecting their account type. This information is stored in the users table through the **Create** operation.
2. **Profile Management:** Before accessing their dashboards, all users must complete their profiles (hotel_details or ngo_details). This is a one-time **Create** or **Update** operation.
3. **Donation Listing & Management:** A 'HOTEL' user logs in and accesses their dashboard to post surplus food. They use the **Create** operation to add new listings, the **Read** operation to view their own listings, and the **Update** or **Delete** operations to manage them.
4. **Donation Collection:** A registered 'NGO' user logs in and **Reads** the central list of all 'Available' donations. When they claim an item, they execute an **Update** operation that changes the donation's status to 'Picked Up' and assigns them as the collector.

In short, the system ensures secure login , accurate data handling, role-based permissions , and real-time tracking of food donations , making the entire food rescue process digital, efficient, and traceable.

CHAPTER 8: CONCLUSION

The "CRUD Operations on Food Donation System" successfully demonstrates how technology can be used to simplify, secure, and modernize the traditional logistics of food rescue. The project integrates core web development concepts such as front-end design , backend scripting , and database connectivity to create a fully functional web-based donation platform. Through the implementation of CRUD operations, the system ensures smooth handling of data—from user registration to donation claiming—while maintaining traceability and speed throughout the process.

During the development of this project, various technologies such as HTML, CSS , JavaScript, PHP , and MySQL were utilized to design both the user interface and the backend logic. The use of XAMPP as the local server environment provided a stable testing platform for integrating the frontend and database components. This project offered a practical understanding of how web-based systems operate and how data is processed through Create, Read, Update, and Delete operations in real-world applications.

The system's modular design—consisting of 'HOTEL' and 'NGO' modules—helps ensure a clear separation of responsibilities and secure interaction between users and the system. Features such as login authentication , role-based dashboards , and time-sensitive status updates contribute to maintaining data integrity and preventing food waste. Moreover, the project helped in developing essential skills such as database design , session management , and secure backend programming, which are critical for modern web developers.

In conclusion, the Food Donation System successfully fulfills its primary objective of providing a digital alternative to inefficient manual coordination. It enhances efficiency, traceability, and accessibility while reducing food waste and administrative workload. The system's functionality demonstrates how automation can improve logistical operations and promote a sustainable solution to a critical social problem.

For future enhancement, the project can be expanded by integrating advanced security and trust features such as an **Admin verification panel** to vet new NGOs , **real-time notifications** to alert NGOs of new donations in their service area , and **blockchain technology** to ensure end-to-end, immutable traceability for all donations. Overall, this project serves as a solid foundation for understanding full-stack development and real-world data-driven application design in the context of sustainable food management

CHAPTER 9: REFERENCES

The following resources were referred to during the design, development, and documentation of the "CRUD Operations on Food Donation System." These references provided guidance in understanding programming concepts, database management, and web development techniques essential for implementing the project.

1. **W3Schools – https://www.w3schools.com** → Used for learning HTML, CSS, JavaScript, and PHP syntax and examples.
2. **MySQL Official Documentation – https://dev.mysql.com/doc** → Provided information on database queries, table creation , and CRUD operations.
3. **PHP Official Website – https://www.php.net** → Referred to for understanding PHP functions , password hashing , and database connectivity.
4. **XAMPP Apache Friends – https://www.apachefriends.org** → Used to install and configure the XAMPP environment for local web hosting.
5. **MDN Web Docs – https://developer.mozilla.org** → Helped in understanding web standards, HTML5, CSS3 , and JavaScript features.
6. **U.S. Environmental Protection Agency (EPA) – https://www.epa.gov** → Referenced for understanding food waste statistics and the importance of food donation.
7. **GeeksforGeeks – https://www.geeksforgeeks.org** → Referenced for logical concepts in web development, data validation, and CRUD operations.
8. **Stack Overflow – https://stackoverflow.com** → Used to troubleshoot coding issues and learn implementation techniques from developer discussions.
9. **ResearchGate – [suspicious link removed]** → Consulted for research papers on food donation systems, including the need for Admin panels and future enhancements.
10. **College Project Guidelines – Provided by Surana College (Autonomous)** → Used as the primary structure and formatting reference for preparing this project report.