

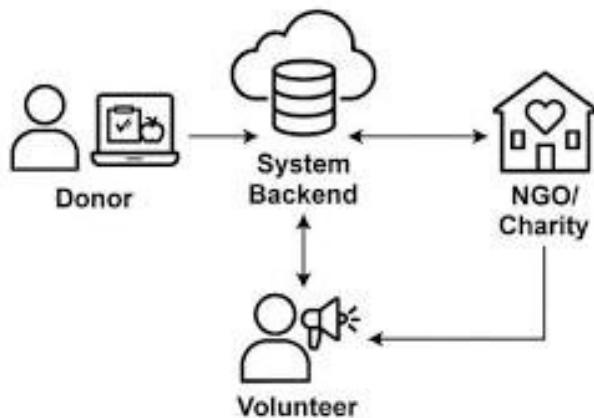
## PROJECT DESIGN PHASE-II

### TECHNOLOGY STACK (ARCHITECTURE & STACK)

|                       |                                 |
|-----------------------|---------------------------------|
| <b>Date:</b>          | 1 November 2025                 |
| <b>Team ID:</b>       | NM2025TMID01438                 |
| <b>Project Name:</b>  | To Supply Leftover Food to Poor |
| <b>Maximum Marks:</b> | 4 Marks                         |

#### **Technical Architecture:**

The system helps collect leftover food from restaurants, events, and households, then redistributes it to needy people using a web platform and mobile interface. The architecture consists of user registration, food donation requests, real-time location tracking, and delivery management.



#### **Guidelines Followed:**

- Includes all processes (as application logic/technology block)
- Provides infrastructural demarcation (Local/Cloud)
- Indicates external interfaces (third-party APIs like Maps, Payment Gateway, etc.)
- Indicates data storage components/services
- Indicates optional interfaces for AI-based demand prediction (if applicable)

**Table-1: Components & Technologies**

| S.No | Component                                | Description   | Technology                         |
|------|--|---|------------------------------------|
| 1.   | <b>User Interface</b>                    | Donors, volunteers, and admins interact via a web and mobile dashboard. | ReactJS (Web),<br>Flutter (Mobile) |
| 2.   | <b>Application Logic-1</b>               | Manages donor registration, food listings, and volunteer assignments.   | Node.js / Express.js               |
| 3.   | <b>Application Logic-2</b>               | Validates location, food expiry time, and nearby volunteers.            | REST API, Google Maps API          |
| 4.   | <b>Application Logic-3</b>               | Sends real-time notifications to volunteers and NGOs.                   | Firebase Cloud Messaging           |
| 5.   | <b>Database</b>                          | Stores donor info, food details, volunteer data, and delivery records.  | MongoDB / MySQL                    |
| 6.   | <b>Cloud Database</b>                    | Centralized cloud-hosted backend for all data operations.               | AWS / Firebase Cloud               |
| 7.   | <b>File Storage</b>                      | Stores uploaded food images, reports, and donor verification proofs.    | AWS S3 / Firebase Storage          |
| 8.   | <b>External API-1</b>                    | Google Maps API for location tracking and navigation.                   | Google Cloud Services              |
| 9.   | <b>External API-2 (Optional)</b>         | NGO verification via government portal API.                             | REST API Integration               |
| 10.  | <b>Machine Learning Model (Optional)</b> | Predicts high-demand areas for food donation.                           | Python (TensorFlow / scikit-learn) |
| 11.  | <b>Infrastructure (Server / Cloud)</b>   | Hosted on scalable cloud architecture for reliability and uptime.       | AWS EC2 / Firebase Hosting         |

**Table-2: Application Characteristics**

| S.No | Characteristics                 | Description  | Technology                      |
|------|---------------------------------|--|---------------------------------|
| 1.   | <b>Open-Source Frameworks</b>   | Built using open-source front-end and back-end technologies.   | ReactJS, Node.js, MongoDB       |
| 2.   | <b>Security Implementations</b> | Includes role-based access, data encryption, and secure login. | JWT Authentication, HTTPS       |
| 3.   | <b>Scalable Architecture</b>    | Supports increasing numbers of donors, NGOs, and volunteers.   | Microservices Architecture      |
| 4.   | <b>Availability</b>             | Cloud hosting ensures continuous system availability.          | AWS / Firebase Cloud            |
| 5.   | <b>Performance</b>              | Fast response via optimized queries and caching.               | Redis Cache, Indexed DB Queries |