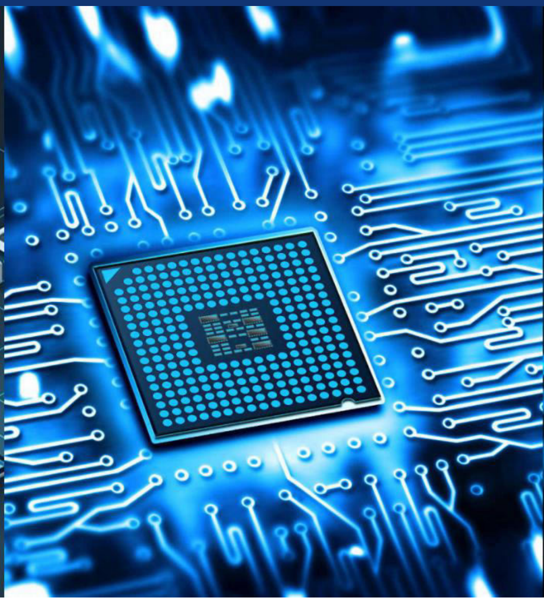
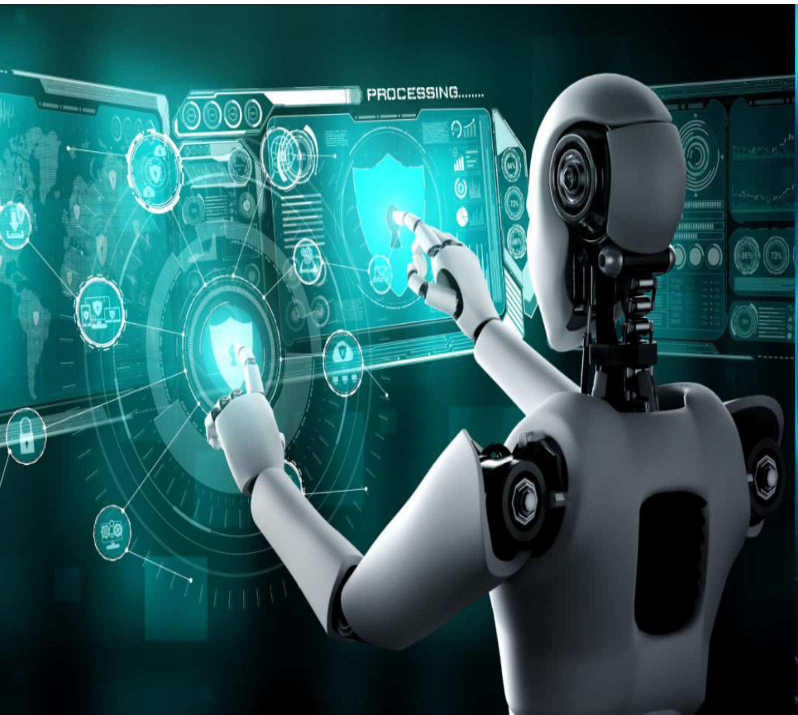
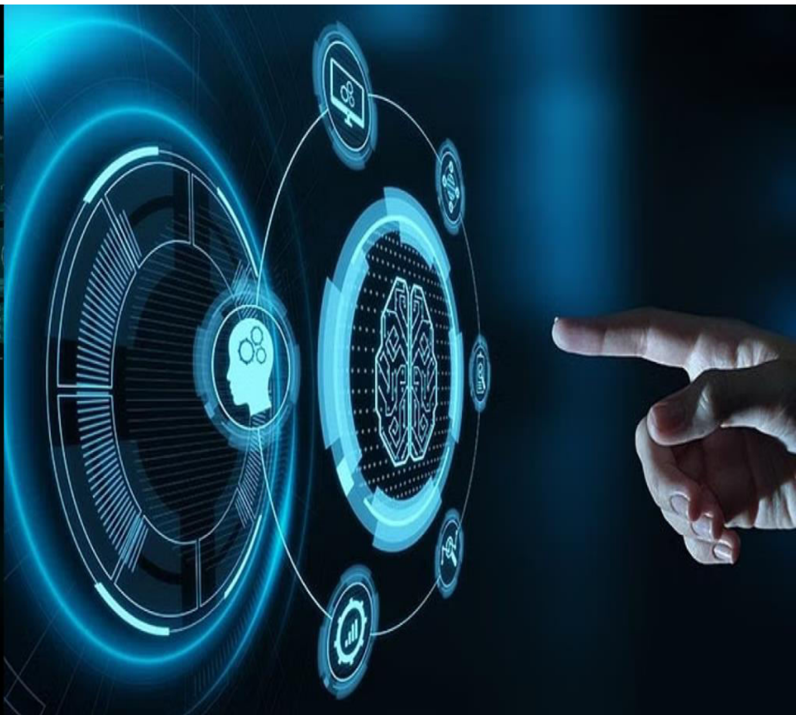
ISSN(O): 2320-9801
ISSN(P): 2320-9798


International Journal of Innovative Research in Computer and Communication Engineering

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)





Impact Factor: 8.771

Volume 14, Issue 3, March 2026

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Development of Food Redistribution Platform Using Supabase and PostgreSQL

Amit Atri[1], Deepika Kataria[2], Venu Madhava Reddy Y[2], Mohan Boya[2] and Harsha Vardhan A[2]

Department of Electrical and Electronics Engineering, Manav Rachna International Institute of Research and Studies, Faridabad, India[1]

Department of Computer Science and Engineering, Manav Rachna International Institute of Research and Studies, Faridabad, India[2]

ABSTRACT: Food donation through Web Application (FoodShare) addresses the critical paradox of food wastage and persistent hunger in India. Despite adequate national food production, a large amount of food is wasted leaving millions of lives in food scarcity. This study presents the design and implementation of a technology-driven web platform that bridges the gap between food donors and recipients through a streamlined digital interface. The application is developed using HTML, CSS, and JavaScript for the frontend, with Supabase and PostgreSQL supporting backend services and relational database management. The system enables user registration, real-time food listing, browsing, and coordinated pickups through integrated communication and geolocation features. Security mechanisms such as user authentication, encrypted data transmission, and role-based access control ensure data privacy and system integrity. Comprehensive testing including unit, integration, system, and user acceptance validates reliability and performance. The proposed platform demonstrates scalability, maintainability, and user-centric design while contributing to Sustainable Development Goals, particularly Zero Hunger (SDG 2), Responsible Consumption and Production (SDG 12), and Reduced Inequalities (SDG 10).

KEYWORDS: Food Donation, Web Application, Supabase, Real-time Notification, Location Services, Food Waste Reduction,

I. INTRODUCTION

In a country as vast and populous as India, a profound paradox exists at the heart of its food system. While India produces sufficient food to feed its entire population, millions of citizens face hunger and malnutrition on a daily basis. Simultaneously, an alarming quantity of edible food is wasted throughout the country's supply chains and consumption systems. This wastage occurs across multiple sectors including households (generating about 55 kg per capita annually), restaurants, hotels, catering services, retail establishments, and large-scale events. The sources of this food waste are multifaceted. In restaurants and hotels, food waste results from preparation errors, unsold inventory, and customer plate waste. In households, it stems from overpurchasing, miscalculations in meal preparation, spoilage due to inadequate storage facilities, and changing food preferences. Large events such as weddings, corporate functions, and parties often generate massive quantities of surplus food that finds its way to landfills rather than tables of those in need.

According to the UNEP Food Waste Index Report 2024, Indian households generate about 55 kg of food waste per person per year, resulting in roughly 78.2 million tonnes of food wasted annually [1]. Earlier assessments (UNEP 2021) estimated household food waste at ~50 kg per person annually, amounting to about 68.8 million tonnes per year in India [2]. Estimates indicate 10–15 % of food grains produced are lost due to poor storage, pest damage, and logistical inefficiencies in India [3]. According to FAO estimates, up to 30–40 % of fruits and vegetables in India are lost before reaching consumers, due to spoilage and lack of cold chain infrastructure [4]. Globally, around 1.05 billion tonnes of food were wasted in 2022, with food waste contributing a significant share to emissions and lost nutrition — underscoring the scale of the problem India also faces [1]. Some estimates suggest food waste in India could represent loss of ₹92,000 crore annually due to inefficiencies across the value chain [5]. Food waste can also serve as a resource of energy on subjecting it to anaerobic digestion [6].

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

This dichotomy between abundance and scarcity, waste and want, is the core problem that this paper aims to address. The fundamental challenge is the inefficiency and informality of the current food donation ecosystem. Food donation in India predominantly operates through informal channels, including personal networks, social media, word-of-mouth recommendations, and non-governmental organizations (NGOs). While several platforms like Robin Hood Army, Feeding India, and No Food Waste have made commendable efforts in this space, they operate with significant limitations: restricted geographic coverage, lack of real -time tracking, non -intuitive user interfaces, dependence on manual coordination, and inability to handle the time - sensitive nature of food (which is perishable and requires quick redistribution).

Proposed web application "FoodShare" is a full-stack, three-tier web application designed to enable efficient and transparent food redistribution between donors and recipients. The system comprises a responsive frontend built with HTML5, CSS3, and Vanilla JavaScript, supported by Bootstrap/Tailwind CSS for cross-device compatibility. The backend is implemented using Supabase, integrating PostgreSQL for relational data management, RESTful APIs for communication, and built-in authentication for secure access control. Real-time data synchronization supports instant updates for donations and reservations. The normalized database schema ensures data integrity and optimized queries. Deployment is managed via Netlify and Supabase Cloud, with Git-based version control supporting maintainable and scalable development.

II. TECHNOLOGICAL FRAMEWORK

The integration of digital technologies to address societal challenges has gained significant scholarly attention under the domains of social entrepreneurship and Technology for Development (Tech4Dev). Effective technology-driven social solutions are grounded in user-centric design, ensuring that systems are tailored to the real needs and constraints of target beneficiaries rather than being technology-imposed interventions [7]. Accessibility and affordability are equally critical, particularly in developing economies where disparities in device ownership, digital literacy, and internet affordability persist [8]. Scalability and sustainability further determine long-term impact, requiring solutions to incorporate viable business models and modular architectures from inception [9]. Interoperability with existing digital ecosystems enhances adoption and reduces systemic friction.

Given that smartphone penetration in India has reached approximately 65% [10], mobile-first and web-first approaches are essential. Progressive Web Apps (PWAs), responsive design, offline capability, and data-efficient architectures ensure inclusivity across heterogeneous device environments [11]. Emerging technologies such as blockchain offer transparency and traceability in supply chains, enhancing accountability [12], while Artificial Intelligence and Machine Learning enable predictive analytics, recommendation systems, and demand forecasting in food redistribution networks [13].

The backend architectural decision for FoodShare involved evaluating Firebase and Supabase. Firebase's NoSQL Firestore provides real-time synchronization and scalability but limits complex relational queries and introduces vendor lock-in concerns [14]. Supabase, built on PostgreSQL, supports relational schemas, SQL joins, and Row-Level Security (RLS), enabling fine-grained access control and predictable pricing [15]. Considering the relational complexity of food donation workflows and the need for structured data integrity, Supabase was selected.

Development adhered to modern web best practices, including responsive design, WCAG-compliant accessibility, RESTful API principles, secure authentication, input validation, and rate limiting to ensure performance, security, and maintainability [16].

III. SYSTEM REQUIREMENTS AND DESIGN

The system requirements analysis establishes the functional capabilities and performance constraints necessary for the effective operation of the FoodShare platform. These requirements provide the foundation for system design, implementation, and evaluation.

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

3.1 Functional Requirements

The functional requirements define the core operational capabilities of the FoodShare system. These requirements were prioritized based on system criticality and implementation feasibility. Critical functionalities include user authentication (email-based login and password reset), food donation form submission, quantity and category selection, pickup location specification, expiry time input, real-time donation listing, and donation reservation mechanisms. High-priority features comprise user profile management, food image upload, time slot selection, contact information display, filtering and location-based search, messaging system, notification system, and donation history tracking. Certain features such as the rating and review system and administrative dashboard have been identified as future enhancements to improve system governance and trust management.

3.2 Non-Functional Requirements

Non-functional requirements specify performance, reliability, scalability, and security constraints. The system targets 99% uptime availability and response times below two seconds. Scalability has been designed to support over 1000 concurrent users. Security measures include AES-256 encryption, OAuth 2.0 authentication, backup and disaster recovery mechanisms, and partial compliance with data privacy regulations.

Cross-browser compatibility and mobile responsiveness have been achieved to ensure accessibility across heterogeneous devices. Performance under load conditions has been tested to validate operational stability.

3.3 Use Case Analysis

Three primary use cases were modeled: Donor Registration, Food Donation Creation, and Browse and Reserve Food.
The Donor Registration use case involves role selection, credential creation, email verification, and dashboard access.
The Food Donation Creation use case allows donors to submit structured information including food title, category, quantity, images, pickup location, time slots, and expiry details. Upon submission, the system validates inputs, stores records, and notifies nearby receivers within a defined radius.
The Browse and Reserve Food use case enables receivers to filter donations by category, distance, and availability, review detailed listings, reserve available items, and communicate with donors via in-app messaging.

3.4 System Architecture

FoodShare follows a three-tier distributed architecture to ensure modularity and scalability.
Presentation Layer (Frontend): A responsive web interface developed using HTML5, CSS3, and JavaScript operates within the client’s browser, handling user interaction and form validation while communicating with backend services via REST APIs.
Application Layer (Backend): Supabase provides RESTful API endpoints, business logic processing, authentication, authorization, and real-time data synchronization.
Data Layer (Database): A PostgreSQL relational database ensures structured data storage, referential integrity, and row-level security enforcement across users, donations, reservations, and communication records.
This layered architecture enhances maintainability, scalability, and secure data management while supporting real-time interaction between stakeholders. Communication flow is shown as in fig 1 and deployment architecture in fig 2

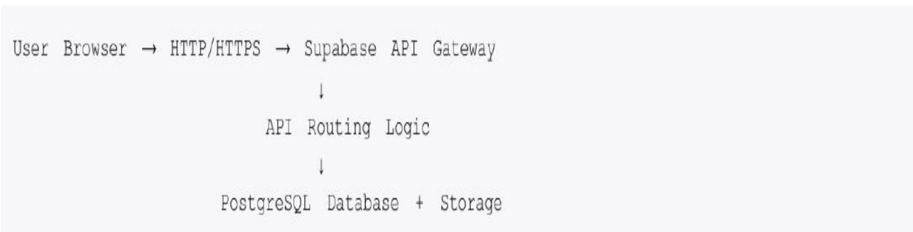


Fig 1. Communication flow

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

```
Frontend: Netlify CDN
    ↓
    ├→ HTML/CSS/JS Assets
    └→ API Calls

Backend: Supabase Cloud
    ├→ PostgreSQL Database (Managed)
    ├→ Authentication Service (GoTrue)

    ├→ REST API (PostgREST)
    ├→ File Storage (S3-compatible)
    └→ Real-time Subscriptions

Services:
    ├→ Google Maps API (Location Services)
    └→ Supabase Auth (User Management)
```

Fig 2. Deployment architecture

3.4 Database Design

The FoodShare database is structured using a relational schema comprising seven primary tables: users, donations, reservations, messages, ratings, notifications, and associated metadata tables. The design enforces well-defined relationships and integrity constraints to ensure consistency and optimized query performance. One-to-many relationships exist between users and donations, donations and reservations (with only one active reservation permitted per donation), and users and messages. A many-to-many association between users and donations is implemented implicitly through the reservations table. Referential integrity is maintained through foreign key constraints, while unique constraints prevent duplicate reservations for the same donation-receiver pair. An indexing strategy is applied to frequently queried fields to enhance retrieval efficiency. Additionally, Row-Level Security (RLS) policies enforce fine-grained access control, ensuring users can access only authorized records, thereby strengthening data security and privacy compliance.

3.5 API Design and Specifications

The FoodShare system follows a RESTful API architecture with a base URL (shown in fig 3) hosted on Supabase and secured using Bearer token authentication in the Authorization header. All API responses are formatted in JSON to ensure standardized data exchange between frontend and backend services. Core endpoints are structured into Authentication (fig 4), Donation (fig 5), Reservation (fig 6), and Messaging modules (fig 7), enabling secure user management and real-time operational workflows.

https://xyzcompany.supabase.co/rest/v1

Fig 3. URL

```
POST /auth/v1/signup
- Create new user account
- Request: { email, password, full_name, phone, user_role, location }
- Response: { user, session }

POST /auth/v1/signin
- User login
- Request: { email, password }
- Response: { user, session }

POST /auth/v1/logout
- User logout
- Request: (authenticated)
- Response: { success }

POST /auth/v1/reset-password
- Request password reset
- Request: { email }
- Response: { message }
```

Fig 4. Authentication Endpoints

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

```
GET /donations
- List all active donations
- Query params: ?category=cooked&amp;max_distance=5000&amp;limit=20&amp;offset=0
- Response: { donations: [...], total_count, has_more }

GET /donations/{donation_id}
- Get donation details
- Response: { donation }

POST /donations
- Create new donation (requires auth, role=donor)
- Request: { food_title, category, description, quantity, images, pickup_location, ... }
- Response: { donation_id, created_at }

PUT /donations/{donation_id}
- Update donation (only owner can edit)
- Request: { food_title, description, ... }
- Response: { updated_donation }

DELETE /donations/{donation_id}
- Cancel donation (only owner)
- Response: { success, message }
```

Fig 5. Donation Endpoints

```
POST /reservations
- Create reservation (requires auth, role=receiver)
- Request: { donation_id, pickup_date_time }
- Response: { reservation_id }

GET /reservations/my-reservations
- Get user's reservations (auth required)
- Response: { reservations: [...] }

PUT /reservations/{reservation_id}
- Update reservation status
- Request: { status: 'confirmed' | 'completed' | 'cancelled' }
- Response: { updated_reservation }
```

Fig 6. Reservation Endpoints

```
GET /messages/{reservation_id}
- Get messages for a reservation
- Response: { messages: [...] }

POST /messages
- Send message
- Request: { reservation_id, message_content }
- Response: { message_id, created_at }
```

Fig 7. Message Endpoints

IV. IMPLEMENTATION DETAILS

The implementation phase translates the system design and architectural specifications into a functional and deployable application. This section outlines the technical realization of frontend, backend, security mechanisms, and deployment strategies adopted in the FoodShare platform.

4.1 Frontend Implementation

The frontend of FoodShare is developed using a lightweight yet robust technology stack comprising HTML5, CSS3, Tailwind CSS, and Vanilla JavaScript (ES6+). HTML5 ensures semantic structuring of web content, improving accessibility and maintainability. CSS3 combined with Tailwind CSS follows a utility-first approach, enabling rapid UI

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

development with responsive design principles. Vanilla JavaScript provides dynamic interactivity without introducing additional framework overhead, ensuring performance efficiency.

The key frontend components include the Authentication Module, Donation Form Component, Real-Time Donation Listing, and In-App Messaging System. The authentication module manages user registration, login, password reset, and email verification workflows. The donation form allows donors to upload food details, images, pickup location, time slots, and expiry information, with client-side validation to enhance data accuracy. Real-time listing dynamically updates available donations using backend subscriptions. The messaging module enables direct communication between donors and receivers after reservation confirmation. Additionally, location-based map integration enhances discoverability of nearby food donations.

4.2 Backend Implementation

The backend is implemented using Supabase, built on PostgreSQL 13+, with RESTful and GraphQL APIs auto-generated through PostgREST and PostGraphQL. Authentication is configured using email/password login, email verification, JWT-based session management, and password reset functionality.

The database configuration enables Row-Level Security (RLS) on all tables to enforce fine-grained access control. Automated daily backups with 30-day retention ensure data resilience. Supabase Storage is configured with a dedicated "donations" bucket for food images, limiting uploads to 5 MB per image and implementing automated deletion of unused files after 90 days.

Custom backend logic is implemented using PostgreSQL functions and triggers. For example, reservation confirmation functions automatically update donation status and generate notifications, ensuring transactional consistency and workflow automation.

4.3 Security Implementation

Security mechanisms are integrated at multiple layers. Passwords are hashed using bcrypt with 12-round salting and enforced complexity requirements. Data in transit is secured using HTTPS/TLS 1.3, while sensitive fields are encrypted at rest using AES-256 encryption.

Access control is implemented through Row-Level Security policies and role-based authorization. API keys and secrets are stored securely in environment variables. Additional protections include CORS configuration, parameterized queries to prevent SQL injection, HTML escaping for XSS prevention, CSRF token validation, and dual-layer input validation at both frontend and backend levels.

4.4 Deployment Strategy

The frontend is deployed using Netlify, providing automated builds, SSL certification, and global CDN distribution. The backend is hosted on Supabase Cloud, where database migrations are applied automatically and API endpoints are provisioned instantly. Real-time subscriptions, RLS enforcement, automated SSL, and built-in DDoS protection ensure secure and scalable production deployment.

This implementation strategy ensures high availability, performance optimization, and secure operations while maintaining architectural modularity and scalability.

V. TESTING AND VALIDATION

The testing and validation phase was conducted to ensure functional correctness, system reliability, performance efficiency, and security compliance of the FoodShare platform. A structured multi-level testing methodology was adopted to evaluate the system comprehensively under both normal and stress conditions.

5.1 Testing Methodology

FoodShare employed a layered testing strategy consisting of Unit Testing, Integration Testing, System Testing, User Acceptance Testing (UAT), and Performance Testing. Unit testing validated individual components in isolation. Integration testing verified interactions between interconnected modules. System testing assessed complete end-to-end

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

workflows. User Acceptance Testing evaluated real-world usability scenarios, while performance testing measured system behavior under simulated load conditions.

5.2 Unit Testing

Unit testing was performed on both frontend and backend components to verify functional accuracy. Frontend unit tests validated input handling mechanisms such as email format verification and password strength validation. Functions responsible for form validation were tested against valid and invalid inputs to ensure robustness. Backend unit testing focused on database operations, authentication workflows, reservation status updates, and notification triggers. Isolated testing of PostgreSQL functions and API endpoints ensured transactional consistency and data integrity.

5.3 Integration Testing

Integration testing verified seamless communication between system modules. Authentication and database integration testing ensured that user credentials were correctly validated and stored. Donation and notification integration confirmed that upon donation submission, real-time notifications were triggered for eligible receivers. Messaging integration validated bidirectional communication between donors and receivers after reservation confirmation.

5.4 System Testing

System testing evaluated complete user workflows from donor login to food pickup completion. A comprehensive test case was executed for the entire donation lifecycle, including login, donation creation, image upload, real-time notification delivery, reservation confirmation, in-app messaging, and final completion marking. All steps successfully passed validation, confirming operational integrity of the end-to-end system.

5.5 Performance Testing

Performance testing assessed system responsiveness and stability under load. Load testing simulated concurrent users performing donation, browsing, and reservation operations. Results indicated acceptable response times within the target threshold (<2 seconds) and stable performance under designed concurrency limits. Performance testing and testing results are shown in fig 8a and 8b).

The comprehensive testing strategy ensured that FoodShare meets functional, performance, and reliability standards required for deployment in real-world environments.

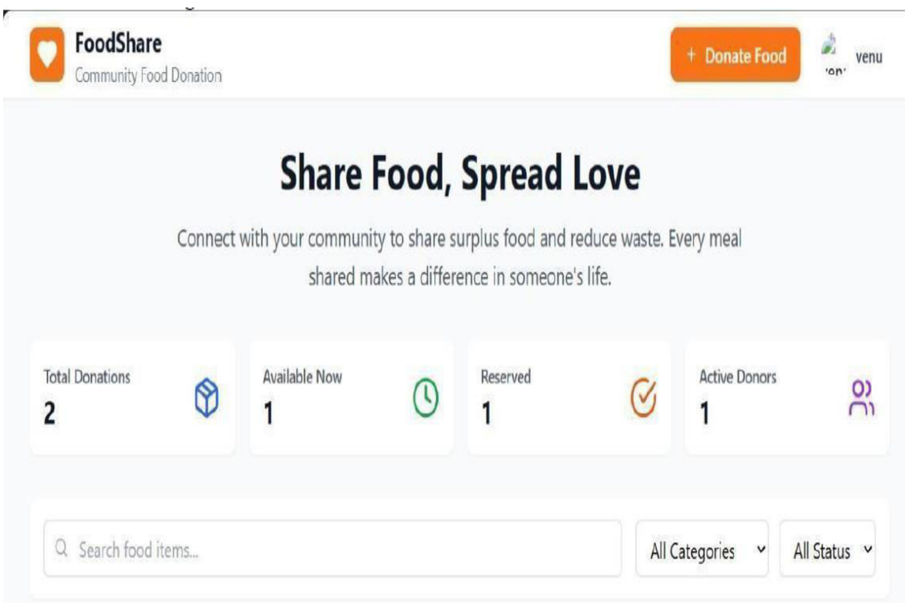


Fig 8a. Performance testing

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

```
Test Configuration:
- Virtual Users: 100 concurrent users
- Ramp-up Time: 5 minutes
- Test Duration: 15 minutes
- Request Rate: 10 requests/second

Results:
- Average Response Time: 1.2 seconds
- 95th Percentile Response Time: 2.1 seconds
- 99th Percentile Response Time: 3.5 seconds
- Error Rate: 0.02% (acceptable)
- Throughput: 150 requests/second
- Database Query Time: 89ms average
- API Response Time: 189ms average
- Network/Browser Time: 312ms average (acceptable for web)

Conclusion: System handles 100 concurrent users effectively.
Recommendation: Implement read replicas before scaling to 500+ users.
```

Fig 8b. Testing results

VI. CONCLUSION

The FoodShare application demonstrates the effective use of digital technology to address the dual challenge of food waste and food insecurity in India. By creating a real-time, user-friendly platform that connects food donors with recipients, the system provides a scalable and practical solution for surplus food redistribution. The application successfully meets its primary objectives of reducing food waste, improving access to food for vulnerable populations, ensuring accessibility through intuitive design, and supporting scalable growth through a robust three-tier architecture.

From a technical perspective, the project achieved full-stack implementation using modern web technologies, a scalable PostgreSQL database, real-time integration via Supabase, and comprehensive security mechanisms. All core functional requirements were implemented and validated through extensive testing. Beyond technical accomplishments, FoodShare presents a deployable social impact model with the potential to redirect surplus meals efficiently and serve as a replicable framework for other regions seeking sustainable food redistribution solutions.

REFERENCES

[1] United Nations Environment Programme (UNEP), Food Waste Index Report 2024. Pp. 41
[2] United Nations Environment Programme (UNEP), Food Waste Index Report 2021. Pp. 40
[3] Press Information Bureau (PIB), Government of India, "Measures to reduce post-harvest losses of food grains," Ministry of Consumer Affairs, Food & Public Distribution, New Delhi, India, 2023
[4] Food and Agriculture Organization (FAO), The State of Food and Agriculture 2019: Moving Forward on Food Loss and Waste Reduction. Rome, Italy: FAO, 2019
[5] NITI Aayog, Strategy for New India @75. New Delhi, India: Government of India, 2018
[6] Atri, A., Khosla, A., Gaur, S. (2025). Power Generation from Food and Garden Waste in Educational Institute. In: Lecture Notes in Electrical Engineering, vol 1303
[7] A. Smith and R. Anderson, "Technology for Social Good: A Review of Emerging Practices," Journal of Social Entrepreneurship, vol. 11, no. 2, pp. 145–160, 2020
[8] World Bank, World Development Report 2021: Data for Better Lives. Washington, DC, USA: World Bank, 2021
[9] OECD, Digital Innovation for Social Impact. Paris, France: OECD Publishing, 2020
[10] Internet and Mobile Association of India (IAMAI), Digital in India Report 2024. Mumbai, India: IAMAI, 2024
[11] A. Russell, Progressive Web Apps. Sebastopol, CA, USA: O'Reilly Media, 2016
[12] M. Casino, T. K. Dasaklis, and C. Patsakis, "A systematic literature review of blockchain-based applications," Telematics and Informatics, vol. 36, pp. 55–81, 2019

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

[13] FAO, The State of Food and Agriculture 2019: Moving Forward on Food Loss and Waste Reduction. Rome, Italy: FAO, 2019
[14] Google Cloud, “Cloud Firestore Documentation,” 2023
[15] Supabase Inc., “Supabase Documentation,” 2024
[16] M. Fowler, Patterns of Enterprise Application Architecture. Boston, MA, USA: Addison-Wesley, 2002