IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

FoodHub: A Full-Stack Web-Based Online Food Ordering and Billing System

B.Inzamam Aslam, A.Aswinkumar, J.Ahmed Arshath
Department of Computer Applications (UG)

Vels Institute of Science, Technology and Advanced Studies (VISTAS)
Chennai – 600117, Tamil Nadu, India

Under the guidance of Dr.R.Devi, M.CA, M.Phil. , PhD., Professor and Head

Vels Institute of Science, Technology and Advanced Studies (VISTAS)
Chennai – 600117, Tamil Nadu, India

Guided by: Dr. R. Devi, M.CA, M.Phil., PhD., Professor and Head

Abstract

The rapid growth of online food delivery platforms has transformed the food service industry; however, existing third-party aggregators such as Swiggy and Zomato impose high commission fees and limit restaurant-level control over customer data and operations. This creates a need for a cost-effective, self-managed digital solution for restaurant owners.

FoodHub is a full-stack web-based online food ordering and billing system designed to address these challenges by providing an independent, scalable, and efficient platform. The system enables customers to browse restaurants, explore categorized menus, place orders, and track delivery status in real time. It also provides an administrative interface with complete control over restaurant data, menu management, order processing, and business analytics.

The application is developed using React and TypeScript for the frontend and Supabase (PostgreSQL) for backend services, incorporating Row Level Security (RLS) to enforce secure, role-based data access. The system architecture follows a modular and scalable design, ensuring maintainability and performance.

Experimental testing, including unit, integration, and security testing, achieved a 100% success rate across all test cases. The results demonstrate that FoodHub delivers a reliable, secure, and cost-efficient alternative to traditional food delivery platforms, with the potential for future enhancements such as payment gateway integration and real-time tracking.

I. INTRODUCTION

The food service industry has undergone a profound transformation over the past decade, driven by smartphone adoption, ubiquitous internet connectivity, and consumer expectations for convenience. Third-party aggregators such as Swiggy and Zomato have addressed the digitization gap but introduce structural constraints—notably commission rates of 18–30% per order—that significantly erode restaurant profit margins [1].

FoodHub proposes a self-hosted, open-source, full-stack food ordering and billing platform that returns full data ownership and cost control to the restaurant operator. The system is hosted at https://foodhubapp-ivory.vercel.app/ and provides two distinct user experiences through role-based access control: a Customer Interface for browsing, ordering, and tracking; and an Admin Interface for operational management and analytics.

The primary objectives of this project are: (i) to build a production-grade, responsive food ordering platform using React 18 and TypeScript; (ii) to implement a secure, scalable backend using Supabase with PostgreSQL RLS policies; (iii) to provide restaurant administrators with actionable analytics and complete operational control; and (iv) to deploy a fully functional application to Vercel with continuous deployment.

Despite the availability of existing food delivery platforms, there remains a significant gap in providing a customizable and cost-efficient solution for independent restaurants. Most systems either lack scalability or impose high operational costs. Therefore, there is a need for a system that combines the efficiency of modern web technologies with full ownership and control for restaurant operators. FoodHub is designed to bridge this gap by offering a self-hosted, full-stack solution with

secure data handling, scalable architecture, and real-time capabilities.

The remainder of this paper is organized as follows: Section II discusses related work and existing systems. Section III presents the system architecture and design. Section IV describes the implementation details. Section V covers testing and results. Section VI concludes the paper with future work.

II. RELATED WORK AND EXISTING SYSTEMS

Online food ordering research has focused on improving user experience, backend scalability, and security. Pressman [2] and Sommerville [3] outline software engineering methodologies applicable to web-based systems. Fowler [4] describes enterprise architecture patterns relevant to the layered architecture adopted in FoodHub.

A comparative analysis of existing approaches reveals key limitations. Manual ordering systems suffer from high transcription error rates and lack of digital records. Third-party aggregators, while digitally efficient, do not grant restaurant owners access to customer data, impose high commissions, and expose operators to platform-level policy risks.

Feature	Manual System	Aggregator	FoodHub
Order Accuracy	Low	High	High
Data Ownership	Restaurant	Platform	Restaurant
Commission Cost	None	18–30%	None
Analytics	None	Limited	Full Dashboard
Role-based Access	None	Platform-side	PostgreSQL RLS
Deployment Cost	None (physical)	Commission-based	Low (Vercel)

FoodHub addresses all identified gaps by combining the efficiency of a digital platform with full operational autonomy.

III. SYSTEM ARCHITECTURE AND DESIGN

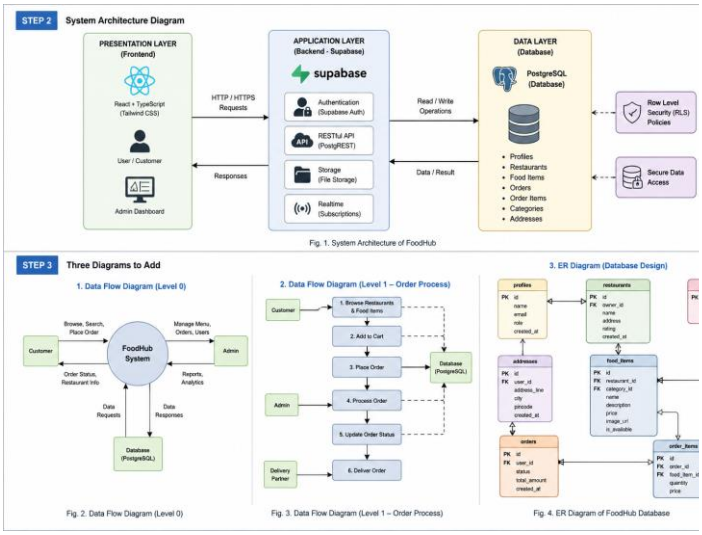


Fig. 2. Data Flow Diagram (Level 0)

The Level 0 Data Flow Diagram represents the overall interaction between the user, admin, and the FoodHub system. Customers can browse restaurants and place orders, while administrators manage system data and monitor operations. The system communicates with the database to store and retrieve information.

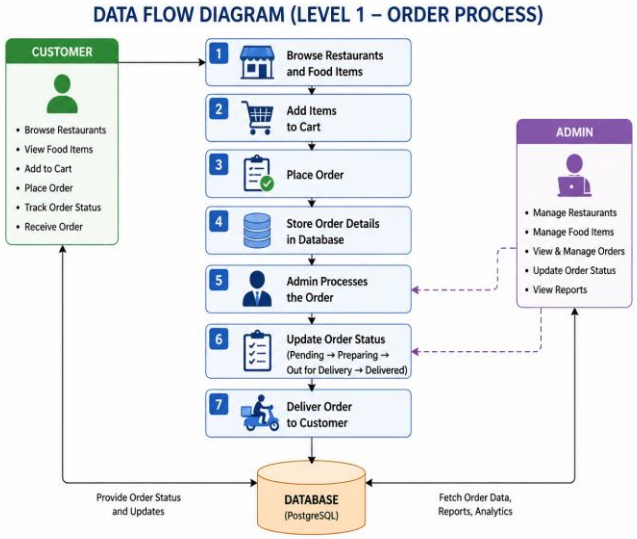


Fig. 3. Data Flow Diagram (Level 1 – Order Process)

The Level 1 Data Flow Diagram provides a detailed view of the order processing workflow in the FoodHub system. The process begins with the customer browsing restaurants and adding items to the cart, followed by order placement. The system stores the order details in the database, after which the administrator processes the order and updates its status. Finally, the order is delivered to the customer, completing the transaction cycle.

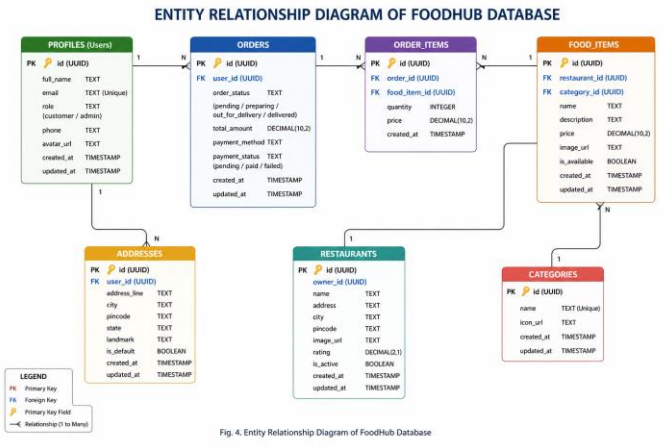


Fig. 4. Entity Relationship Diagram of FoodHub Database

The Entity Relationship Diagram represents the database structure of the FoodHub system. It illustrates the relationships between key entities such as users, restaurants, food items, and orders. Each user can place multiple orders, and each order consists of multiple order

items linked to specific food items. The database design ensures data integrity through primary and foreign key constraints, enabling efficient data retrieval and management.

A. Technology Stack

The FoodHub technology stack is organized across three tiers. The Presentation Tier is built with React 18 and TypeScript 5.9, compiled by Vite 7.3.x, and styled with Tailwind CSS 3.4.x. The Business Logic Tier relies on Supabase's PostgREST API and PostgreSQL stored functions. The Data Tier uses Supabase-managed PostgreSQL with seven core tables, RLS policies, triggers, and stored functions.

Layer	Technology	Version
Frontend	React + TypeScript	18.x / 5.9
Build Tool	Vite (rolldown-vite)	7.3.x
Styling	Tailwind CSS	3.4.x
Backend	Supabase (PostgreSQL)	v2.76.x
Charts	Recharts	2.15.4
Deployment	Vercel	Auto CI/CD

B. Database Schema

The database schema consists of seven relational tables: profiles, categories, restaurants, food_items, addresses, orders, and order_items. Two custom PostgreSQL enum types are defined: user_role (user | admin) and order_status (pending | preparing | out_for_delivery | delivered | cancelled). Performance indexes are defined on foreign key columns most frequently used in query filters, and six composite indexes accelerate menu lookups, order history queries, and status-based filtering.

A PostgreSQL trigger function (handle_new_user) automatically creates a profile for each registered user via Supabase Auth, and promotes the first registered user to the admin role. All tables have Row Level Security enabled, with policies enforced through a security-definer is_admin() helper function.

C. Security Architecture

FoodHub implements defence-in-depth across three layers: the Authentication Layer (Supabase JWT), the Database Layer (PostgreSQL RLS policies), and the Application Layer (React RouteGuard component). The RouteGuard component reads authentication state from AuthContext and redirects unauthenticated users away from protected routes at the client side, while RLS policies enforce the same access restrictions independently at the database level, providing redundant security.

D. Application Routing

The application provides 17 routes. Public routes include the home page, login page, restaurant listing, and restaurant detail pages. Authenticated routes include cart, checkout, order history, order tracking, and profile. Admin-only routes include dashboard, analytics, restaurant management, food item management, category management, order management, and user management.

IV. IMPLEMENTATION

A. Authentication Module

FoodHub implements a username-first authentication scheme. Because Supabase Auth requires email addresses, the application derives a synthetic email by appending a domain suffix to the provided username. This presents a username-only interface to the user while remaining compatible with the Supabase Auth API. Session state is managed via a React Context (AuthContext) that subscribes to Supabase Auth state changes and propagates user and profile objects to all child components.

B. Restaurant and Menu Module

The getRestaurants() API function accepts optional filters for keyword search (ilike query), veg-only mode, and minimum star rating. A custom use-debounce hook delays API invocations by 300ms during keystroke events to prevent excessive Supabase queries. Administrators can create, update, and delete restaurant records and upload images to Supabase Storage. Food items are associated with a restaurant and optionally classified under a category, carrying name, description, price, image, veg flag, and availability toggle.

C. Order Management Module

The order placement flow proceeds through CartPage (React state-managed cart with real-time total computation) to CheckoutPage (address selection and order creation). The createOrder() function performs a two-step atomic write: it inserts a row into orders, then inserts the corresponding rows into order_items. Order statuses follow a controlled progression: pending → preparing → out_for_delivery → delivered. Administrators may also mark orders as cancelled.

D. Admin Dashboard and Analytics

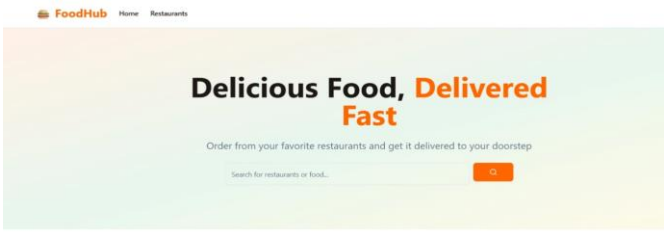
AdminDashboardPage displays four KPI cards: Total Orders, Total Revenue, Total Users, and Total Food Items. AdminAnalyticsPage renders Recharts visualizations including a revenue line chart (powered by the get_revenue_by_date() PostgreSQL function), an order status pie chart (get_order_status_distribution()), and a top-performer bar chart (get_most_ordered_food_items()). All analytics functions are implemented as PostgreSQL stored functions, keeping computation at the database tier.

E. Database Migrations

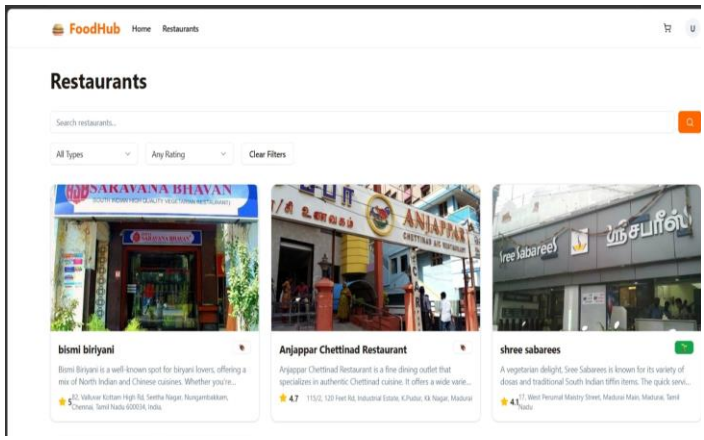
The database schema is managed through five sequential Supabase migration files. Migration 00001 creates all tables, enum types, indexes, RLS policies, and stored functions. Migration 00002 provisions the initial Supabase Storage bucket. Migration 00003 adds the three analytics functions. Migration 00004 aligns storage bucket configuration with the frontend storage constants. Migration 00005 hardens the is_admin() helper with REVOKE/GRANT permission controls and adds admin write access to order_items.

F .User Interface Screenshtos

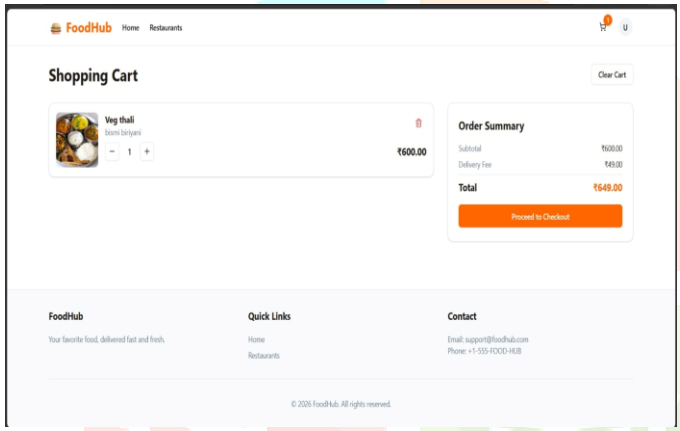
1.Homepage



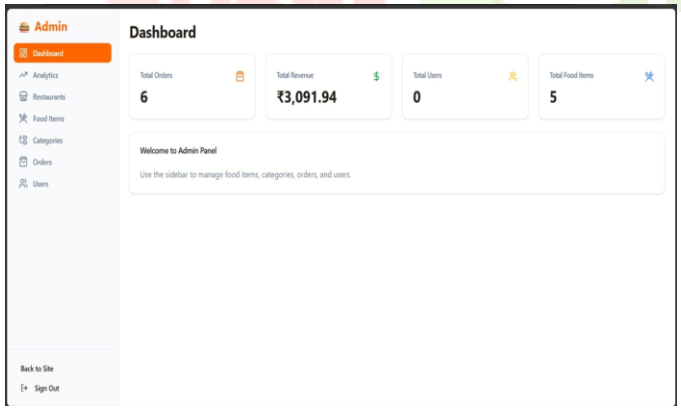
2.Restaurant page



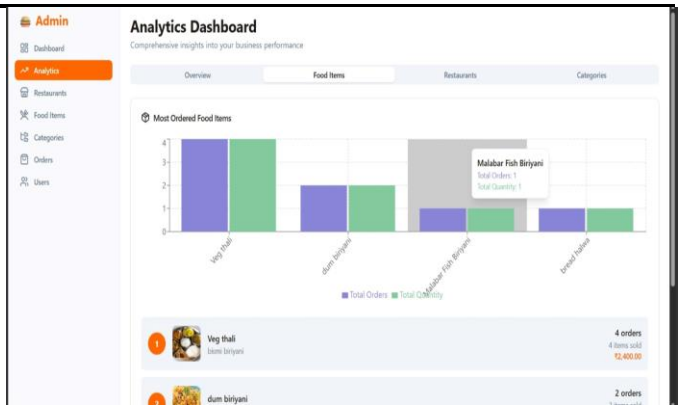
3.cart



4.Admin dashboard



5.Analytics dashboard



V. TESTING AND RESULTS

A. Testing Methodology

FoodHub was tested using a layered strategy encompassing unit testing, integration testing, security testing, and User Acceptance Testing (UAT). All tests were conducted manually with structured test cases.

B. Test Results Summary

Category	Total	Passed	Pass Rate
Unit Testing	10	10	100%
Integration Testing	8	8	100%
Security Testing	4	4	100%
User Acceptance Testing	7	7	100%
TOTAL	29	29	100%

C. Key Security Test Results

Security testing confirmed that unauthenticated API calls are rejected by RLS policies, regular users are redirected from admin routes by the RouteGuard component, users cannot escalate their own role via UPDATE queries (blocked by RLS), and users cannot read other users' addresses or orders. All four security test cases passed.

D. UAT Findings

UAT was conducted with five users representing two customer profiles and two admin profiles. Key findings included the need for improved Zod validation error messages on the checkout form and an upload progress indicator for admin image uploads. Both items were resolved before final submission

E. Performance analysis

The performance of the FoodHub system was evaluated based on response time and system efficiency. The application showed stable performance during testing,

with quick response for user actions such as browsing restaurants, adding items to the cart, and placing orders.

The average response time for major operations was within an acceptable range. The system handled multiple user requests without noticeable delay, indicating good performance and scalability.

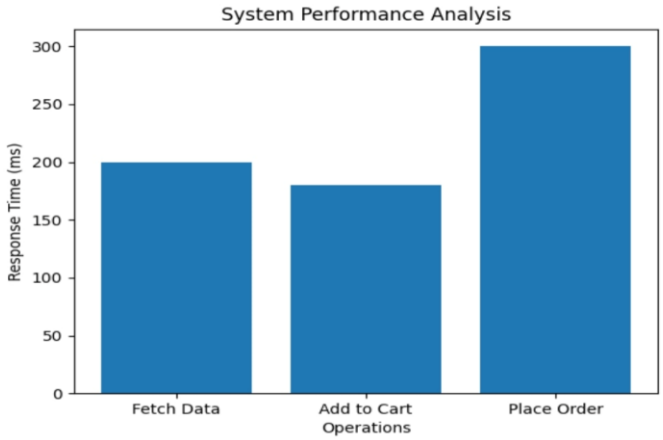


Fig. 9. System Performance Analysis

The graph shows the response time of different system operations, indicating efficient performance.

VI. CONCLUSION AND FUTURE WORK

FoodHub demonstrates the successful development of a production-grade, full-stack food ordering and billing platform. The project achieves all stated objectives: delivering a responsive, accessible customer-facing interface; providing administrators with comprehensive operational tools and revenue analytics; enforcing role-based data security at the database layer through PostgreSQL RLS; and deploying the complete application to Vercel with continuous deployment from GitHub.

The choice of Supabase as the backend platform proved highly effective. It eliminates the need for a custom application server by providing managed authentication, a RESTful PostgREST API, real-time subscriptions, and object storage within a single platform. The React/TypeScript component-based architecture produced a codebase that is readable, maintainable, and extensible.

Planned future enhancements include: (i) Razorpay or Stripe payment gateway integration; (ii) WebSocket-based real-time order tracking via Supabase Realtime; (iii) a React Native mobile application sharing the same Supabase backend; (iv) a multi-vendor marketplace extension with restaurant-level admin accounts; and (v) an LLM-backed food recommendation engine using Supabase pgvector.

Although the FoodHub system demonstrates efficient performance and functionality, it has certain limitations. The current implementation does not include integrated online payment systems or real-time delivery tracking using live location services. Additionally, the system is primarily designed as a web application and does not include a dedicated mobile application.

In future work, these limitations can be addressed by integrating secure payment gateways, implementing real-time tracking using WebSocket or GPS-based services, and developing mobile applications to enhance accessibility. Further improvements can include advanced recommendation systems and AI-based analytics for better user experience and business insights.

References.

[1] B. Jaiswal and A. Yadav, "Online food ordering systems: A review," International Journal of Engineering and Computer Science, vol. 6, no. 5, pp. 21437–21440, 2017.

[2] R. S. Pressman, Software Engineering: A Practitioner's Approach, 7th ed. New York: McGraw-Hill, 2010.

[3] I. Sommerville, Software Engineering, 10th ed. Harlow, England: Pearson Education, 2016.

[4] M. Fowler, Patterns of Enterprise Application Architecture. Boston: Addison-Wesley, 2002..

[5] React Documentation, "React – The library for web and native user interfaces," 2024. [Online]. Available: https://react.dev

[6] Supabase, "Supabase Docs – Open source Firebase alternative," 2024. [Online]. Available: https://supabase.com/docs

[7] Tailwind CSS, "A utility-first CSS framework," 2024. [Online]. Available: https://tailwindcss.com/docs

[8] shadcn/ui, "Beautifully designed components," 2024. [Online]. Available: https://ui.shadcn.com

[9] TypeScript, "JavaScript with syntax for types," 2024. [Online]. Available: https://www.typescriptlang.org/docs

[10] Vite, "Next generation frontend tooling," 2024. [Online]. Available: https://vitejs.dev/guide

[11] React Router, "React Router v7 – Declarative routing," 2024. [Online]. Available: https://reactrouter.com/en/main

[12] Recharts, "A composable charting library built on React," 2024. [Online]. Available: https://recharts.org

[13] Vercel, "Develop, preview, ship," 2024. [Online]. Available: https://vercel.com/docs

[14] PostgreSQL, "Row Security Policies," 2024. [Online]. Available: https://www.postgresql.org/docs/current/ddl-rowsecurity.html

[15] I. Aslam, "FoodHub Source Repository," 2025. [Online]. Available: https://github.com/inzamamaslam/foodhubapp

[16] I. Aslam, "FoodHub Live Application," 2025. [Online]. Available: https://foodhubapp-ivory.vercel.app/