INTERNATIONAL JOURNAL OF ECONOMIC SOCIAL SCIENCE AND MANAGEMENT LAW
Peer Reviewed, Referred & Indexed Journal
E-ISSN: 3070-0558
VOL.7, NO.2(1) (2026)

Cloud Kitchen Management & Ordering Platform: Architecture, Challenges, and Emerging Technologies

Soni Kumari Singh
Student, Dept. of CSE
GIFT Autonomous, Bhubaneswar
Odisha, India

B.Kirti
Student, Dept. of CSE
GIFT Autonomous, Bhubaneswar
Odisha, India

Prof.Antaryami Muduli
Professor, Dept. of CSE
GIFT Autonomous, Bhubaneswar, Odisha, India

Abstract—**The rapid growth of online food delivery services has significantly transformed the restaurant and cloud kitchen industry. Although popular food delivery platforms such as Zomato and Swiggy provide large-scale food delivery solutions, many small restaurants and cloud kitchens still face challenges such as high commission charges, dependency on third-party platforms, limited customization options, and lack of operational control. These limitations reduce profit margins and restrict businesses from managing their own customers, delivery workflows, and platform functionalities independently.**

The Cloud Kitchen project is developed to provide an independent, scalable, and real-time food ordering and delivery management system that enables restaurants and cloud kitchens to manage their complete operations without relying entirely on external food delivery services. The system is designed as a full-stack web platform consisting of three separate applications: a Customer Website, an Admin Dashboard, and a Delivery Partner Application, all connected through a centralized backend API server.

The Customer Website allows users to browse menu items, search products, manage carts, place orders, make online or cash-on-delivery payments, and track deliveries in real time. The Admin Dashboard enables administrators to manage menu categories, food items, customers, orders, delivery partners, and operational settings through a centralized interface with real-time monitoring features. The Delivery Partner Application allows delivery personnel to receive delivery requests, update delivery progress, share live GPS location, and manage earnings and incentives efficiently.

The backend system is developed using Express.js with TypeScript and Bun runtime, while PostgreSQL is used as the primary database along with Prisma ORM for database management. Real-time communication is implemented using Socket.IO to provide instant synchronization of order updates, delivery status, and live GPS tracking. Stripe payment gateway integration enables secure online payment processing, while Cloudinary is used for image upload and cloud storage management. The frontend applications are developed using Next.js, React.js, Tailwind CSS, Zustand, and TanStack React Query to provide responsive and interactive user interfaces.

The project demonstrates the practical implementation of modern web technologies, cloud deployment, real-time communication systems, scalable architecture, and full-stack application development in the food delivery domain. The developed platform aims to reduce dependency on third-party services, improve operational efficiency, enhance customer experience, and provide a customizable and cost-effective solution for cloud kitchens and food delivery startups.

Keywords—*Real-time systems; Food delivery; Socket.IO; Geolocation tracking; Order management; Scalability; Route optimization; Progressive Web Apps*

I. INTRODUCTION

The food delivery industry has undergone a dramatic transformation over the past decade. Driven by the ubiquity of smartphones, affordable mobile internet, and the growing demand for convenience, on-demand food delivery platforms have emerged as one of the fastest-growing segments in the global digital economy. Platforms such as Swiggy, Zomato, DoorDash, and Uber Eats have demonstrated that the integration of real-time technologies with logistics management can create scalable, profitable, and customer-centric ecosystems. According to industry estimates, the global online food delivery market is projected to exceed USD 500 billion by 2030, driven primarily by urban consumers demanding speed, variety, and reliability.

At the core of these platforms lies a complex interplay of software systems that must operate in real time. A customer placing an order expects immediate confirmation, live tracking of the delivery partner, and prompt delivery—all within a seamless user experience. Achieving this requires sophisticated architectures that handle concurrent connections, low-latency communication, dynamic route computation, and fault tolerance at scale. Unlike traditional

Received: 20-04-2026 | Accepted: 28-05-2026 | Published: 04-06-2026 | www.ijeml.com

e-commerce platforms, food delivery systems impose strict temporal constraints: a meal prepared too early grows cold, while a delay in order assignment can cascade into customer dissatisfaction and revenue loss.

Real-time food delivery systems are inherently multi-stakeholder platforms. Three primary actors interact continuously: the customer who places and tracks orders, the restaurant or cloud kitchen that prepares food, and the delivery partner who collects and delivers the order. Coordinating these three parties in real time—while managing payments, notifications, geolocation, and status updates—demands a carefully designed technology stack.

This paper surveys the state of real-time food delivery systems by examining their architectural patterns, technology stacks, operational challenges, and future directions. Section II discusses the system architecture and key components. Section III examines the technology stack and real-time communication mechanisms. Section IV explores the major challenges. Section V discusses open research issues. Section VI reviews tools and frameworks. Section VII provides suggestions for future work, and Section VIII concludes the paper.

II. SYSTEM ARCHITECTURE AND KEY COMPONENTS

A real-time food delivery system is typically organized as a multi-tier, microservice-oriented architecture. The system is composed of three distinct frontend applications—the customer website, the administrative dashboard, and the delivery partner application—all served by a unified backend API layer. This separation of concerns allows each interface to be developed, deployed, and scaled independently while sharing a common data model and business logic layer.

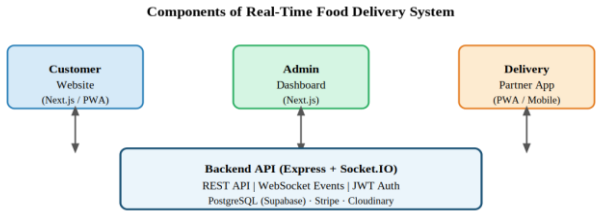

Fig. 1: Components of a Real-Time Food Delivery System

A. Customer-Facing Application

The customer application serves as the primary point of interaction for end users. It provides functionalities for menu browsing, cart management, address selection, order placement, and live delivery tracking. Modern customer applications are built as Progressive Web Applications (PWAs) or native mobile apps, ensuring cross-platform compatibility. Key user flows include phone-based OTP authentication, real-time order status updates via WebSockets, and interactive maps displaying the delivery partner's live location. The application must handle intermittent connectivity gracefully, caching menu data and cart state locally to support offline interactions.

B. Administrative Dashboard

The admin dashboard provides kitchen operators with a comprehensive control panel for managing the entire order lifecycle. It presents real-time order alerts, revenue analytics, delivery partner GPS locations, and menu management interfaces. Critical features include the ability to broadcast ready orders to available delivery partners, assign partners manually, process refunds, and configure kitchen settings such as operating hours and delivery radius. The dashboard consumes real-time events from the backend via a persistent WebSocket connection, rendering live updates without page refreshes.

C. Delivery Partner Application

The delivery partner application is a mobile-first PWA designed for field use. It enables partners to toggle their availability status, accept or reject assigned orders, confirm pickups using OTP verification, and share their GPS coordinates with the backend in real time. The application includes a wallet module for tracking earnings and requesting withdrawals, an incentive system with tiered bonus structures, and a delivery history log. GPS streaming is implemented using the browser Geolocation API, with coordinates transmitted to the server via Socket.IO at configurable intervals.

D. Backend API Layer

The backend serves as the central nervous system of the platform. It exposes a RESTful API for standard CRUD operations and a Socket.IO server for real-time bidirectional communication. Role-based access control differentiates between customer, admin, and delivery partner contexts. The backend integrates with external services for payment processing (Stripe), image hosting (Cloudinary), and SMS delivery for OTP verification. The database layer uses an ORM with connection pooling to handle concurrent requests efficiently.

III. TECHNOLOGY STACK AND REAL-TIME COMMUNICATION

The technology choices in a food delivery platform directly influence its performance, scalability, and developer experience. Modern platforms increasingly adopt JavaScript/TypeScript full-stack approaches, enabling code sharing between frontend and backend while leveraging a rich ecosystem of open-source libraries.

A. Backend Technologies

The backend runtime environment plays a significant role in request throughput and latency. Runtimes such as Bun and Node.js, built on V8, provide high-performance non-

blocking I/O suitable for handling thousands of concurrent connections. Express.js remains the dominant web framework for building RESTful APIs due to its minimalism and extensibility. TypeScript adoption has grown substantially, providing type safety that reduces runtime errors in complex domain models such as order state machines and payment flows. ORM frameworks like Prisma simplify database interactions by generating type-safe query builders, while also providing schema migration tooling essential for evolving database schemas without downtime.

B. Real-Time Communication with Socket.IO

WebSocket-based communication is the cornerstone of the real-time experience in food delivery systems. Socket.IO, built atop WebSockets with fallback transports, provides a robust abstraction for bidirectional event-driven communication. In a food delivery context, Socket.IO enables several critical real-time flows: new order alerts broadcast to the admin dashboard, order status change events pushed to customers, delivery partner GPS coordinates streamed to both admin and customer views, and order assignment events delivered to specific partner devices.

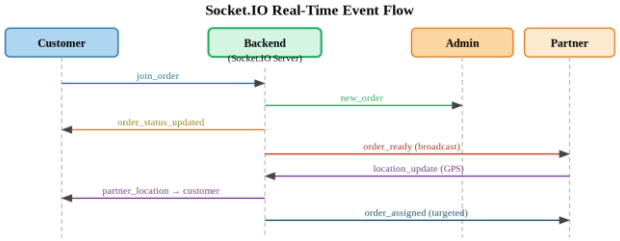

Fig. 2: Socket.IO Real-Time Event Flow Between Actors

C. Frontend Technologies

Next.js has become the de facto framework for building food delivery frontends due to its support for server-side rendering, static site generation, and API routes within a single codebase. React 19's concurrent rendering features improve perceived performance on lower-end devices. State management libraries such as Zustand provide lightweight, persistent client-side state for authentication tokens, cart contents, and user preferences. TanStack React Query handles asynchronous data fetching with built-in caching, background refetching, and optimistic updates—all critical for a responsive order-tracking experience. Mapping libraries like Leaflet with react-leaflet render interactive delivery tracking maps without the licensing costs of proprietary alternatives.

D. Payment Integration

Secure and seamless payment processing is a fundamental requirement. Stripe's PaymentIntent API is widely adopted for its PCI-compliant tokenization, support for multiple payment methods, and webhook-based payment confirmation. The typical flow involves the backend creating a PaymentIntent and returning a client secret to the frontend, which then renders Stripe Elements to collect card details without sensitive data touching the application server. Upon payment confirmation, the backend verifies the payment status via webhook and transitions the order from PAYMENT_PENDING to PLACED.

IV. CHALLENGES IN REAL-TIME FOOD DELIVERY SYSTEMS

Real-time food delivery systems face a distinct set of technical and operational challenges that differentiate them from conventional e-commerce platforms. Unlike static web applications, these platforms must synchronize state across multiple devices simultaneously while maintaining strict latency budgets and high availability. The challenges span multiple dimensions including system scalability, geolocation accuracy, delivery optimization, fault tolerance, and information security. Addressing these challenges requires interdisciplinary expertise spanning distributed systems, operations research, mobile computing, and regulatory compliance.

A. Scalability and Concurrency

During peak hours—typically lunch (12:00–14:00) and dinner (19:00–21:00)—food delivery platforms experience order surges that can be three to five times the average baseline load. The backend must scale horizontally to handle thousands of simultaneous WebSocket connections alongside REST API requests. Stateful WebSocket connections fundamentally complicate horizontal scaling: a client connected to one server instance cannot directly receive events emitted by another instance. This locality problem requires additional infrastructure such as a Redis-based pub/sub adapter for Socket.IO, which broadcasts events across all server nodes through a shared message bus. While effective, this approach introduces network round-trip overhead between the application server and the Redis instance, adding 1–10 ms of latency per event in typical cloud deployments.

Container orchestration platforms like Kubernetes automate horizontal scaling based on CPU and memory metrics. However, food delivery workloads are better predicted by time-of-day and day-of-week patterns rather than instantaneous infrastructure metrics. Proactive, schedule-based autoscaling—spinning up additional pods 15 minutes before anticipated peak periods—can significantly reduce the cold-start latency penalty associated with reactive scaling. Furthermore, connection draining must be handled gracefully during scale-down events to avoid prematurely terminating active WebSocket sessions that carry live order tracking state.

Database scalability presents an additional challenge. PostgreSQL, while robust for transactional workloads, does not natively support horizontal write scaling. Connection pooling via PgBouncer or built-in pooling in managed

Received: 20-04-2026 | Accepted: 28-05-2026 | Published: 04-06-2026 | www.ijeml.com

services like Supabase is essential to prevent connection exhaustion under high concurrency. For read-heavy endpoints such as menu listings and order history, read replicas offload query load from the primary instance, though replication lag must be carefully managed to avoid serving stale order status data.

B. Geolocation Accuracy and GPS Tracking

Accurate real-time geolocation is foundational to the live delivery tracking feature. The browser Geolocation API, standardized in the W3C Geolocation specification, provides latitude, longitude, and accuracy estimates sourced from GPS hardware, Wi-Fi triangulation, or cellular network positioning depending on device capability and environment. In dense urban areas, GPS signals suffer from multipath interference caused by reflections off buildings, leading to positional errors of 10–50 meters. Inside building lobbies or underground parking facilities, GPS signals may be entirely unavailable, causing tracking to appear frozen on the customer's map.

Continuous GPS polling at high frequencies (e.g., every 2 seconds) improves tracking smoothness but significantly accelerates battery drain on delivery partner devices. Adaptive polling strategies—reducing update frequency when the partner is stationary and increasing it during active movement detected via the device accelerometer—provide a practical trade-off. On the server side, received GPS coordinates are stored in a dedicated delivery location table and simultaneously emitted to the order room via Socket.IO, where the customer's map component re-renders the partner marker. Kalman filtering applied to the incoming coordinate stream can smooth erratic positional jumps caused by signal noise, providing a more natural tracking experience.

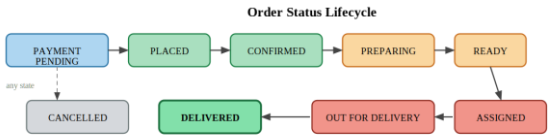

Fig. 3: Order Status Lifecycle in a Food Delivery Platform

C. Order Assignment and Route Optimization

Efficient order assignment constitutes a combinatorial optimization problem with real-time constraints. At any given moment, the system must select, from a pool of available delivery partners, the optimal partner for each newly ready order—balancing proximity to the restaurant, current workload, and estimated delivery time to the customer. Greedy nearest-partner assignment is computationally inexpensive and easy to implement but is demonstrably suboptimal when multiple orders become ready simultaneously, as it may assign several orders to a single proximate partner while ignoring more distant but idle partners.

The Vehicle Routing Problem (VRP) and its variants—such as the Capacitated VRP (CVRP) and the Time-Windowed VRP (VRPTW)—provide formal frameworks for optimal multi-order assignment. However, exact VRP solvers have exponential worst-case complexity, rendering them impractical for real-time operation at scale. Heuristic approaches including the Clarke-Wright savings algorithm, simulated annealing, and genetic algorithms offer near-optimal solutions within bounded computation times. Reinforcement learning (RL) agents trained on historical dispatch data have also demonstrated competitive performance in dynamic routing scenarios, learning assignment policies that implicitly account for traffic patterns and demand distribution.

D. Fault Tolerance and High Availability

Food delivery platforms must maintain high availability even during partial infrastructure failures. The stateful nature of active order deliveries means that a backend server crash during an ongoing delivery could leave both the customer and delivery partner without real-time updates. Implementing graceful degradation—where clients automatically reconnect and resynchronize state upon connection loss—is therefore critical. Socket.IO's built-in reconnection logic with exponential backoff provides a baseline, but the server must also persist the last-known delivery state so that reconnecting clients can receive a full state snapshot rather than relying solely on live event streams. Health checks, circuit breakers, and automatic instance replacement via container orchestration reduce mean time to recovery (MTTR) for individual server failures. Multi-availability-zone deployments ensure that a data center-level outage does not take down the entire platform. Database failover using PostgreSQL streaming replication with automatic promotion of standby replicas provides recovery point objectives (RPO) of near-zero for transactional data.

E. Data Privacy and Security

Food delivery platforms collect and continuously process sensitive personal data: home and work addresses, phone numbers linked to real identities, payment credentials, and fine-grained movement trajectories of delivery partners. Compliance with India's Digital Personal Data Protection Act (DPDPA 2023) and, for internationally operating platforms, the EU's General Data Protection Regulation (GDPR), requires that personal data be collected minimally, stored securely, and deleted upon request. Phone OTP authentication reduces password-related attack vectors but introduces SMS interception risks via SIM-swapping attacks. Implementing OTP rate limiting, device fingerprinting, and anomaly detection on login patterns mitigates these risks.

Delivery partner GPS data presents particular privacy implications, as continuous location tracking at fine granularity over long periods can reveal sensitive behavioral

patterns including home locations, medical facility visits, and places of worship. Data minimization principles suggest retaining only aggregated or anonymized location data beyond the active delivery window. Payment data must never be stored on platform servers; tokenization via Stripe Elements ensures that raw card numbers are processed exclusively within PCI-DSS compliant infrastructure. JWT tokens used for API authentication should implement short expiry times (15–60 minutes) with secure refresh token rotation to limit the blast radius of token theft.

V. OPEN RESEARCH ISSUES

Despite significant progress, several research issues remain open in the domain of real-time food delivery systems. These represent opportunities for academic and industry collaboration to advance the state of the art.

A. AI-Driven Demand Forecasting

Accurate prediction of order volumes at the restaurant and area level enables proactive delivery partner positioning, reducing assignment latency. Current approaches use historical order data with time-of-day and day-of-week features. However, external factors such as weather conditions, local events, sports matches, and public holidays introduce significant variance that simple statistical models fail to capture. Deep learning models, particularly Long Short-Term Memory (LSTM) networks and Transformer-based architectures, show promise for multi-variate time series forecasting in this domain.

B. Autonomous and Semi-Autonomous Delivery

The deployment of autonomous delivery robots and drones represents a frontier research area with significant regulatory, technical, and social dimensions. Last-mile delivery automation could dramatically reduce operational costs and carbon emissions. However, navigating unstructured urban environments, handling multi-story apartment buildings, and ensuring package security in the absence of a human partner present unsolved engineering challenges. Hybrid approaches, where autonomous vehicles handle the primary route while human partners manage the final building entry, are being piloted but require seamless integration with existing dispatch systems.

C. Sustainability and Green Logistics

The environmental impact of food delivery—particularly single-use packaging and fossil fuel consumption by delivery vehicles—is an emerging area of concern. Research into route batching algorithms that consolidate multiple orders into single delivery trips can significantly reduce per-order carbon emissions. Electric vehicle integration with charging station-aware routing and incentive structures that reward eco-friendly delivery behavior are active research directions.

D. Fairness and Algorithmic Transparency

Order assignment algorithms and incentive structures have significant implications for delivery partner income and working conditions. Research into algorithmic fairness examines whether automated assignment systems inadvertently discriminate based on partner location, gender, or other demographic factors. Incentive tier structures must be designed to motivate performance without inducing unsafe driving behaviors. Transparent explainability of algorithmic decisions is both a research challenge and an emerging regulatory requirement in gig economy legislation.

VI. TOOLS AND FRAMEWORKS FOR FOOD DELIVERY SYSTEMS

A broad ecosystem of open-source and commercial tools supports the development and operation of real-time food delivery platforms. We categorize these by their functional role.

A. Real-Time Communication

Socket.IO is the most widely adopted library for real-time bidirectional event-based communication in Node.js environments. Built atop the WebSocket protocol with automatic fallback to HTTP long-polling, it provides reliable message delivery even in environments with restrictive network proxies. Key features include room-based subscription management, namespace isolation for multi-tenant deployments, and middleware support for authentication integration. The @socket.io/redis-adapter package enables horizontal scaling by routing events across multiple server instances through a Redis pub/sub channel, making Socket.IO suitable for production deployments at scale.

For scenarios requiring simpler unidirectional streaming, Server-Sent Events (SSE) offer a lightweight alternative. SSE connections are standard HTTP connections that remain open, allowing the server to push text events to the client without the overhead of WebSocket handshake negotiation. This makes SSE suitable for order status update feeds where the client does not need to send real-time data upstream. However, SSE lacks the bidirectional capability required for GPS coordinate sharing from delivery partner devices, making Socket.IO the preferred choice for full-featured delivery applications.

At the infrastructure level, Apache Kafka is increasingly adopted as a distributed event streaming platform to decouple real-time event producers from consumers. In a food delivery context, Kafka topics can buffer high-volume events such as GPS location updates, decoupling the Socket.IO broadcast layer from the database persistence layer. This architecture allows the system to absorb bursts of incoming GPS data without creating database write bottlenecks, while consumer groups independently process events for persistence, analytics, and real-time broadcasting.

Received: 20-04-2026 | Accepted: 28-05-2026 | Published: 04-06-2026 | www.ijeml.com

B. Mapping and Geolocation

Leaflet is an open-source JavaScript mapping library that powers interactive delivery tracking maps in customer-facing applications. Its lightweight core (approximately 39 KB gzipped) and extensive plugin ecosystem make it ideal for mobile web deployments where bandwidth is constrained. The react-leaflet package provides a declarative React component wrapper, enabling the live partner location marker to be updated reactively as new GPS coordinates arrive via Socket.IO. OpenStreetMap tiles, freely available under the Open Database License, eliminate the per-request costs associated with proprietary tile providers.

For server-side route calculation, OSRM (Open Source Routing Machine) and GraphHopper are self-hostable routing engines that compute turn-by-turn directions and estimated travel times using OpenStreetMap road network data. These engines can be deployed within the same infrastructure as the application backend, avoiding the per-request pricing of Google Maps Directions API and eliminating latency from external API calls. Google Maps Platform remains the industry benchmark for geocoding accuracy and real-time traffic integration, but its usage-based pricing model can become prohibitive at millions of route requests per day. PostGIS, a spatial extension for PostgreSQL, enables server-side geospatial queries such as finding all delivery partners within a given radius of a restaurant, supporting the order broadcast and assignment workflows.

C. Payment Processing

Stripe is the dominant payment gateway for food delivery platforms due to its developer-friendly API, robust webhook infrastructure, and PCI-DSS Level 1 certification. The PaymentIntent API handles the complete payment lifecycle including authentication via 3D Secure (3DS2), authorization, capture, and refund. Stripe Elements, a set of pre-built UI components, renders the card input form within an iframe hosted on Stripe's servers, ensuring that raw card numbers never pass through the platform's application layer. Webhook events such as payment_intent.succeeded and payment_intent.payment_failed enable the backend to reliably update order status in response to payment outcomes, even in cases where the customer's browser tab closes during payment.

For markets where card payments are less prevalent, Razorpay (India) and PhonePe provide UPI payment integration, allowing customers to pay directly from their bank accounts via the Unified Payments Interface. Cash on Delivery (COD) remains the most widely used payment method in tier-2 and tier-3 Indian cities, requiring no 8payment gateway integration but introducing cash reconciliation workflows for delivery partners and increased fraud risk for high-value orders.

D. Deployment, Containerization, and CI/CD

Docker enables consistent, reproducible application environments across development, staging, and production deployments. Multi-stage Dockerfiles reduce final image sizes by separating the build environment (with full development dependencies) from the runtime environment (containing only compiled artifacts and production dependencies). A typical three-stage build for a TypeScript backend produces images in the 150–250 MB range, significantly smaller than single-stage builds. GitHub Container Registry (GHCR) provides free private container image hosting for open-source and team repositories, making it a cost-effective alternative to Amazon ECR for independent projects.

GitHub Actions automates the CI/CD pipeline: on each push to the main branch, workflows build and push the Docker image to GHCR, then SSH into the production EC2 instance to pull the new image and restart the container. NGINX acts as the reverse proxy and SSL termination layer on the EC2 host, routing HTTPS traffic to the containerized backend via HTTP on the internal network. Certbot with the Let's Encrypt certificate authority automates TLS certificate provisioning and 90-day renewal, eliminating manual certificate management. Vercel's Git integration automatically deploys Next.js frontend applications on every push, providing instant preview deployments for pull requests and atomic production promotions.

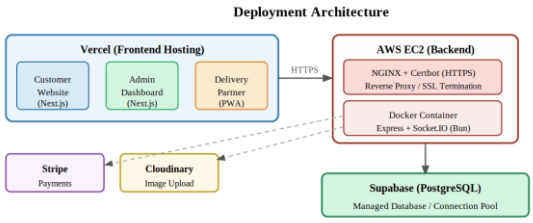

Fig. 4: Deployment Architecture of a Cloud Kitchen Platform

Tool / Library	Category	Primary Use	License	Scalability
Socket.IO	Real-time Comm.	Bidirectional events, GPS streaming	MIT	High (Redis adapter)
Apache Kafka	Message Broker	Event buffering, analytics pipelines	Apache 2.0	Very High
Leaflet + OSM	Mapping	Delivery tracking map	BSD / ODbL	High
OSRM	Route Engine	ETA, route optimization	BSD	Medium
Stripe	Payments	Card/UPI payments, refunds	Commercial	Very High
Docker + GHCR	Containerization	Consistent deployment environments	Apache 2.0	High

Prisma ORM	Database Access	Type-safe queries, migrations	Apache 2.0	Medium
Sentry	Observability	Error tracking, APM	Commercial	High

Fig. 5: Comparison of Key Tools Used in Real-Time Food Delivery Systems

E. Monitoring and Observability

Production food delivery systems require comprehensive observability to detect and diagnose issues before they impact customers. Structured logging with libraries such as Pino (for Node.js/Bun) emits JSON-formatted log entries that can be ingested by log aggregation platforms such as AWS CloudWatch Logs, Datadog, or the ELK Stack (Elasticsearch, Logstash, Kibana). Application Performance Monitoring (APM) tools such as Sentry capture unhandled exceptions and performance traces, enabling developers to identify slow database queries, high-latency API endpoints, and frontend JavaScript errors in real time. Uptime monitoring via services such as Better Uptime or UptimeRobot provides alerting when the API or frontend becomes unreachable, enabling rapid incident response.

VII. SUGGESTIONS FOR FUTURE WORK

Real-time food delivery systems are at an inflection point where advances in artificial intelligence, edge computing, and autonomous systems are poised to reshape operational paradigms. Several promising directions warrant focused research attention.

First, the integration of large language models (LLMs) into customer service workflows could automate complaint resolution, order modification requests, and personalized menu recommendations. Conversational AI agents capable of handling ambiguous customer requests—such as dietary substitutions or address clarifications—could significantly reduce support costs while improving satisfaction.

Second, federated learning approaches could enable platforms to train demand forecasting and fraud detection models collaboratively across restaurant partners without sharing raw transaction data, addressing privacy concerns while improving model accuracy through larger effective datasets.

Third, the development of standardized benchmarks for evaluating food delivery platform performance—encompassing metrics such as end-to-end order latency, assignment efficiency, GPS tracking accuracy, and system availability—would enable rigorous comparison of architectural approaches and drive adoption of best practices across the industry.

Fourth, research into multi-modal delivery networks that seamlessly coordinate motorcycles, bicycles, autonomous robots, and drones within a unified dispatch system represents a rich area for optimization algorithm development and field experimentation.

VIII. CONCLUSION

This paper has presented a comprehensive survey of real-time food delivery systems, examining their multi-tier architectures, enabling technologies, operational challenges, and open research directions. We have shown that the real-time nature of food delivery imposes stringent requirements on system design—from WebSocket-based event propagation and GPS tracking to payment processing and dynamic order assignment. Key challenges including scalability, geolocation accuracy, route optimization, and data privacy demand continued research attention as these platforms scale to serve hundreds of millions of users globally.

The technology landscape for food delivery continues to evolve rapidly, with AI-driven forecasting, autonomous delivery vehicles, and sustainability-aware logistics emerging as transformative forces. We believe that interdisciplinary collaboration between systems engineers, data scientists, operations researchers, and policy makers will be essential to address the complex sociotechnical dimensions of next-generation food delivery platforms. This survey aims to serve as a structured reference for researchers and practitioners entering this domain, providing a foundation for advancing the science and engineering of real-time food delivery systems.

REFERENCES

[1] S. Johari, S. Rajendran and P. Srinivasan, "Real-time delivery management systems: A review of architectures and challenges," International Journal of Computer Applications, vol. 182, no. 5, pp. 1–12, 2019.

[2] A. Gandomi and M. Haider, "Beyond the hype: Big data concepts, methods, and analytics," International Journal of Information Management, vol. 35, no. 2, pp. 137–144, 2015.

[3] T. Sakaki, M. Okazaki and Y. Matsuo, "Earthquake shakes Twitter users: Real-time event detection by social sensors," in Proc. 19th Int. Conf. World Wide Web (WWW), 2010, pp. 851–860.

[4] M. Aazam, S. Zeadally and K. A. Harras, "Offloading in fog computing for IoT: Review, enabling technologies, and research opportunities," Future Generation Computer Systems, vol. 87, pp. 278–289, 2018.

[5] P. Garg and A. Varma, "Geolocation and GPS tracking in mobile applications: Challenges and solutions," in Proc. IEEE Int. Conf. Computing, Communication and Automation (ICCCA), 2016, pp. 847–852.

[6] N. Agatz, A. Fleischmann and M. Schmidt, "Optimization for online food delivery platforms," European Journal of Operational Research, vol. 259, no. 2, pp. 432–446, 2017.

[7] D. Toth and M. Vigo, Eds., The Vehicle Routing Problem: Latest Advances and New Challenges. New York: Springer, 2014.

[8] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.

[9] A. Vaswani et al., "Attention is all you need," in Advances in Neural Information Processing Systems (NeurIPS), 2017, vol. 30.

[10] H. B. McMahan et al., "Communication-efficient learning of deep networks from decentralized data," in Proc. 20th Int. Conf. Artificial Intelligence and Statistics (AISTATS), 2017, pp. 1273–1282.

[11] R. Buyya, C. S. Yeo and S. Venugopal, "Market-oriented cloud computing: Vision, hype, and reality," in Proc. 10th IEEE Int. Conf. High Performance Computing and Communications (HPCC), 2008, pp. 5–13.

[12] I. Grigorik, High Performance Browser Networking. Sebastopol, CA: O'Reilly Media, 2013.

[13] C. L. Philip, Q. Chen and C. Y. Zhang, "Data-intensive applications, challenges, techniques and technologies: A survey on big data," Information Sciences, vol. 275, pp. 314–347, 2014.

[14] K. Kambatla, G. Kollias, V. Kumar and A. Gram, "Trends in big data analytics," Journal of Parallel and Distributed Computing, vol. 74, no. 7, pp. 2561–2573, 2014.

[15] D. P. Acharjya and K. Ahmed, "A survey on big data analytics: Challenges, open research issues and tools," International Journal of Advanced Computer Science and Applications, vol. 7, no. 2, pp. 511–518, 2016.

[16] Sreenivasulu Gajula. (2024). Adaptive Zero Trust Architecture for Securing Financial Microservices. Computer Fraud and Security, 643–655. https://doi.org/10.52710/cfs.845

[17] Gummadi, V. P. K., Chilamkurthi, L. S., & Kavuri, S. (2026). Service Level Objective (SLO) Observability with Splunk and Dynatrace in Microservices. 2026 International Conference on Artificial Intelligence, Systems, and Emerging Technologies (ICAISET), 1–4. https://doi.org/10.1109/icaiset66439.2026.11541542

[18] Kavuri, S. (Ed.). (2024). Shift-left and shift-right testing approaches: A practical roadmap for continuous quality in agile and DevOps. Journal of Information Systems Engineering and Management, 9(4). https://doi.org/10.52783/jisem.v9i4.127

[19] Pavan Kumar Adabala. (2026). IoT-Driven Digital Twins for Manufacturing Optimization: Hybrid Modelling, Reinforcement Learning and Sustainable Operations. International Journal of Computational and Experimental Science and Engineering, 12(1). https://doi.org/10.22399/ijcesen.5050

[20] Doragacharla, V. R. (2026). Building Real-Time Pricing Systems for Modern Retail. Available at SSRN 6451760.

[21] Maturi, S. Y. (2024). Cryptographic privacy engines: Practical multi-party protocols for confidential database queries. Nanotechnology Perceptions, 20(S13), 2770–2785