International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 13 Issue: 04 | Apr 2026 www.irjet.net p-ISSN: 2395-0072

Where2Purchase: A Real-Time O2O Inventory Synchronization and Hyper-Local Product Discovery System with Live Availability Verification

K. Vindhya Anuragh[1], Lekkala Koshik[2], Macha Poorna Prasad[3], Bandi Manoj Reddy[4], Degala Pradeep[5]

[1] Assistant Professor, Dept. of Computer Science and Engineering,
Sri Venkateswara University College of Engineering, Tirupati, Andhra Pradesh, India
[2345] Student, Dept. of Computer Science and Engineering,
Sri Venkateswara University College of Engineering, Tirupati, Andhra Pradesh, India

---

Abstract - *The growing expectation for real-time local product availability information has exposed a critical gap in existing Online-to-Offline (O2O) commerce systems. Current solutions, predominantly reliant on Point-of-Sale (POS) synchronization, suffer from inventory inaccuracy resulting in the widely observed phenomenon of 'ghost stock'—a state where digital records indicate availability for items physically absent from the shelf. This paper presents Where2Purchase, a novel hyper-local product discovery and real-time inventory synchronization system designed to address these deficiencies through three primary innovations: (1) a Live Availability Ping mechanism enabling real-time physical verification by shopkeepers upon customer query, (2) a product-context 1:1 chat system facilitating direct customer-shopkeeper communication scoped to specific product queries, and (3) PostGIS-powered geospatial search utilizing ST_DWithin indexing to replace conventional radius-based approaches with accurate proximity-aware discovery. The system is implemented using React.js, Node.js/Express, PostgreSQL with PostGIS, and Firebase. Functional testing confirms the viability of the proposed architecture in eliminating ghost-stock friction and improving hyper-local retail discovery.*

Key Words: O2O Commerce, Hyper-Local Discovery, Inventory Synchronization, Ghost Stock, PostGIS, Real-Time Systems, WebSockets, Geospatial Query

1. INTRODUCTION

The rapid digitization of consumer behavior has created an expectation for immediate, accurate product availability information before making the decision to visit a physical store. Research indicates that a significant majority of modern consumers utilize digital search tools—including Google Search, Google Maps, and specialized local discovery applications—to verify in-store availability prior to committing to a physical visit. Despite this behavioral shift, a critical and persistent gap exists between what is digitally displayed and what is physically present on the shelf.

Existing Online-to-Offline (O2O) solutions, such as Google's Local Inventory Ads (LIA) and Pointy within Google Business Profiles, have made substantial progress in bridging the discovery gap for local retail. However, these systems are fundamentally constrained by their reliance on Point-of-Sale (POS) synchronization as the sole mechanism for inventory updates. POS-based synchronization is inherently reactive: records update only upon transaction completion, creating a temporal gap during which displayed availability may be entirely inaccurate. This gives rise to 'ghost stock'—where a product appears available online but is physically absent from the shelf.

Additionally, current hyper-local discovery systems employ simplistic geospatial models—typically static circular radii—that fail to account for road-network topology, user velocity, or retail point-of-interest density, resulting in an imprecise discovery experience.

This paper presents Where2Purchase, a real-time O2O inventory synchronization and hyper-local product discovery system addressing these limitations through three core innovations: a Live Availability Ping mechanism, a Product-Context Chat system, and PostGIS ST_DWithin-powered geospatial discovery. The remainder of this paper is organized as follows: Section 2 reviews related work, Section 3 details the system architecture, Section 4 describes the methodology, Section 5 presents functional testing results, Section 6 discusses implications, and Section 7 concludes the paper.

2. LITERATURE REVIEW

2.1 Evolution of Local Product Discovery Systems

The trajectory of local product discovery has evolved from static business directory listings to dynamic, inventory-integrated digital storefronts. Early solutions such as Storemapper and Bullseye addressed the 'where' of retail discovery but provided no insight into product availability, relying on manual updates or scheduled batch CSV imports [1][4].

Google's Pointy, embedded as the Local Inventory feature within Google Business Profiles, enabled small retailers to surface real-time product availability on Google Search and Maps by connecting directly to POS terminals [1]. Facebook's Local Inventory Ads extended this model through behavioral and location-based targeting [7]. Both systems, however, remain confined to a one-way broadcast model with no mechanism for user-initiated verification or contextual communication.

2.2 Limitations of POS-Based Synchronization

The fundamental constraint of POS-reliant systems is their reactive nature. Inventory records update only upon transaction completion, leaving synchronization lags ranging from minutes to hours. During high-traffic retail events, this lag significantly contributes to phantom availability, directing customers to stores for items no longer present [2][3]. These systems also fail to account for the 'item-in-cart' phenomenon, where goods physically held by shoppers have not yet been processed at checkout, leading to overselling risks for low-stock items.

2.3 Geospatial Search Techniques in Mobile Applications

Location-based retail services traditionally rely on Euclidean distance or the Haversine formula for proximity determination. While mathematically sound, Haversine requires full table scans as it computes distance for every database record, scaling poorly with data volume [5]. PostGIS provides superior spatial indexing through Generalized Search Tree (GiST) structures, enabling ST_DWithin to discard irrelevant spatial clusters in logarithmic time, providing substantial query performance improvements [6].

2.4 Real-Time Communication in Commerce Platforms

Existing O2O platforms expose a 'communication gap' that forces consumers to resort to telephone calls for nuanced product queries not captured in standard catalog listings. Research supports real-time chat integration as a means to bridge this gap and reduce the 'convenience surrender' phenomenon—where consumers default to large e-commerce platforms due to reduced digital friction [2]. WebSocket-based bidirectional communication is established as the appropriate protocol for low-latency, stateful chat within microservice architectures.

3. SYSTEM ARCHITECTURE

3.1 Overview

Where2Purchase is architected as a set of decoupled functional services ensuring that high-volume discovery requests do not impact the transactional consistency of the inventory ledger. The primary services are:

- Discovery Service: Manages geospatial proximity queries and search result ranking, optimized for high read throughput using PostGIS spatial indexing.
- Inventory Service: Authoritative source of truth for stock levels with ACID-compliant transactional updates and soft-hold reservation logic.
- Messaging Service: Facilitates real-time product-context communication between customers and shopkeepers via WebSocket-based bidirectional transmission.
- Notification Engine: Manages asynchronous event-driven alerts for Live Ping requests and low-stock threshold triggers.

3.2 Technology Stack

Table -1: System Technology Stack

Layer	Technology	Justification
Frontend	React.js / Next.js	Component-based UI, SSR for performance
Backend API	Node.js / Express.js	Non-blocking I/O for concurrent requests
Primary Database	PostgreSQL + PostGIS	ACID compliance + spatial indexing
Real-Time Messaging	Firebase Realtime DB	Low-latency bidirectional sync
Geospatial Queries	PostGIS ST_DWithin	GiST index, O(log n) query time

3.3 Database Schema

The database design employs a hybrid normalization strategy. Core transactional entities such as Users, Shops, and Inventory follow Third Normal Form (3NF) for referential integrity. The product catalog utilizes JSONB columns for attribute metadata, accommodating heterogeneity of product attributes across diverse retail categories without requiring schema modifications for new product types.

Table -2: Primary Entity Definitions

Entity	Primary Key	Key Attributes
Users	user_id	username, email, role (Customer/Shopkeeper)
Shops	shop_id	owner_id, shop_name, location (GEOGRAPHY Point)
Products	product_id	sku (Unique), product_name, attributes (JSONB)
Inventory	inv_id	shop_id, product_id, qty_available, qty_reserved
Chat Threads	thread_id	customer_id, shop_id, product_context_id, status
Messages	message_id	thread_id, sender_id, content, timestamp, is_read

3.4 Inventory Ledger Implementation

The inventory table bridges the global product catalog and localized shop presence. All updates execute within ACID-compliant transactions to prevent race conditions during simultaneous digital and physical purchase scenarios. A qty_reserved column enables soft-hold logic, temporarily reserving items during active customer queries to prevent overselling:

```
CREATE TABLE inventory (
  inv_id       SERIAL PRIMARY KEY,
  shop_id      INT NOT NULL REFERENCES shops(shop_id),
  product_id   INT NOT NULL REFERENCES products(product_id),
  qty_available INT DEFAULT 0,
  qty_reserved  INT DEFAULT 0,
  reorder_level INT DEFAULT 5,
  last_updated  TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  UNIQUE(shop_id, product_id)
);
```

4. METHODOLOGY AND CORE INNOVATIONS

4.1 Live Availability Ping Mechanism

The Live Availability Ping is the primary differentiating feature of Where2Purchase. Existing O2O platforms operate as passive broadcast systems—inventory data updates at the system level via POS sync, and users receive potentially stale information without any active verification pathway.

When a customer identifies a product of interest, they may initiate a Live Ping to the associated shop. This triggers a real-time WebSocket notification on the shopkeeper's merchant dashboard, requesting rapid physical shelf verification. The shopkeeper responds with one of three states: Confirmed Available, Unavailable, or Unable to Verify. This response is reflected on the customer's interface within seconds, providing a human-verified confidence layer that purely digital systems cannot replicate.

This dual-verification logic—combining POS records as a baseline with the Live Ping as active confirmation—directly addresses ghost stock at its operational root. Ping requests impose no mandatory obligations on shopkeepers, reducing adoption friction for small retail operators.

4.2 Product-Context Chat System

Where2Purchase integrates a 1:1 real-time chat system scoped to a specific product context. Each conversation thread is initialized with the product SKU as a foreign key reference, ensuring permanent association between conversation history and the queried item. This allows shopkeepers to quickly retrieve prior customer inquiries for the same product and enables contextual continuity across sessions.

Firebase Realtime Database provides sub-second message delivery through persistent WebSocket connections. The message schema includes an is_read boolean field and a GiST-indexed timestamp column for efficient unread message counting and paginated history retrieval. This directly addresses the conversion friction observed in existing O2O platforms, where the absence of contextual communication channels forces consumers toward telephone calls or e-commerce alternatives.

4.3 PostGIS-Powered Hyper-Local Discovery

Standard radius-based proximity queries using the Haversine formula require full sequential table scans, scaling poorly with dataset size. Where2Purchase employs PostGIS ST_DWithin with a GiST spatial index on the shop location column, reducing effective query complexity from O(n) to O(log n). Shop locations are stored as GEOGRAPHY(Point, 4326) for accurate WGS84 spherical distance calculations. The core proximity query implementation is as follows:

```
SELECT s.shop_name, i.qty_available
FROM shops s
JOIN inventory i ON s.shop_id = i.shop_id
WHERE ST_DWithin(
  s.location,
  ST_SetSRID(ST_MakePoint(:lon, :lat), 4326)::geography,
  5000  -- radius in meters
)
AND i.product_id = :target_product_id
AND i.qty_available > 0
ORDER BY s.location <->
  ST_SetSRID(ST_MakePoint(:lon, :lat), 4326)::geography;
```

This query simultaneously filters by proximity and availability, returning results ordered by geographic proximity using the KNN operator. Only actionable results—nearby shops with confirmed positive stock—are surfaced to the user.

5. RESULTS AND DISCUSSION

5.1 Functional Testing Overview

Where2Purchase was evaluated through structured functional testing across its three primary innovation areas. Functional correctness and operational viability were verified across representative test scenarios. Due to the current prototype development stage, formal quantitative benchmarking against production-scale datasets was not conducted; however, all core features were verified for correctness.

Table -3: Functional Testing Results

Feature	Test Scenario	Outcome
Live Availability Ping	Customer initiates ping; shopkeeper responds	Notification delivered; status updated within 2–3 seconds
Product-Context Chat	Customer queries product variant via chat	Message delivered bidirectionally; thread linked to SKU
PostGIS Discovery	Proximity query for product within 5km radius	Correct shops returned using GiST spatial index
Ghost Stock Mitigation	Product marked available but physically absent	Live Ping correctly surfaces unavailability
Inventory Soft-Hold	Concurrent queries for same low-stock item	qty_reserved incremented; overselling prevented

5.2 Comparative Analysis

Table -4: Comparison of O2O Discovery Systems

Feature	Static Locators	POS-Based (Pointy/LIA)	Where2Purchase
Data Source	Manual / CSV	Automated POS Sync	POS + Human Verification
Update Latency	Days / Weeks	Minutes / Hours	Seconds (Real-Time)
Interaction Model	Passive Browsing	Passive Intent Capture	Active Query + 1:1 Chat
Verification Logic	None	Systemic (POS Only)	Dual: POS + Live Ping
Spatial Model	Static Points	Radial Proximity	PostGIS ST_DWithin (GiST)
Ghost Stock Handling	None	None	Live Ping Verification

5.3 Discussion

Functional testing confirms the core architectural objectives of Where2Purchase are operationally viable. The Live Availability Ping demonstrates that the temporal gap between POS data and physical shelf reality can be effectively bridged through a lightweight human-in-the-loop verification mechanism without significant latency impact on the user experience.

The PostGIS-based discovery layer demonstrates clear advantages over Haversine-based approaches in query efficiency and spatial accuracy. For production deployments at scale, the GiST spatial index is expected to provide increasing performance benefits relative to full-scan methods as the number of registered shops grows.

The primary limitation of the current implementation is the dependency on shopkeeper responsiveness for the Live Ping mechanism. Future iterations should incorporate automated fallback responses and response time analytics to address this dependency.

6. CONCLUSIONS

This paper presented Where2Purchase, a real-time O2O inventory synchronization and hyper-local product discovery system designed to address the fundamental limitations of existing broadcast-based local inventory platforms. The three core innovations—Live Availability Ping, product-context 1:1 chat, and PostGIS ST_DWithin spatial querying—collectively represent a transition from passive, broadcast-oriented O2O discovery to an interactive, verification-first model.

Functional testing confirms the viability of the proposed architecture. Future work will focus on AI-driven semantic search for natural-language product queries, automated inventory correction based on ping response patterns, and large-scale performance benchmarking under simulated production load. The Where2Purchase framework represents a meaningful step toward a more accurate, interactive, and human-centered model of hyper-local retail discovery.

ACKNOWLEDGEMENT

The authors would like to thank the Department of Computer Science and Engineering, Sri Venkateswara University College of Engineering, Tirupati, for providing the resources and support necessary to carry out this research.

REFERENCES

[1] Google, "Showcase in-store products on Google Search & Maps," Google Business Help, 2023. [Online]. Available: https://support.google.com/business/answer/9934993

[2] J. Smith and R. Lee, "Ghost Stock and the Synchronicity Problem in O2O Commerce," Journal of Retail Technology, vol. 14, no. 2, pp. 45–58, 2022.

[3] 42Signals, "Stock Availability: The Silent Driver of E-commerce Conversions & Loyalty," 42Signals Blog, 2023.

[4] Storemapper, "Store Locator App & Software," 2023. [Online]. Available: https://www.storemapper.com/

[5] Stack Overflow, "PostGIS/Haversine accuracy: which one is best?," 2021. [Online]. Available: https://stackoverflow.com/questions/63569432

[6] Leapcell, "Empowering Web Applications with Geographic Awareness Using PostGIS," Leapcell Blog, 2023.

[7] DataFeedWatch, "Local Inventory Ads: Facebook vs. Google," DataFeedWatch Blog, 2023.

[8] CockroachLabs, "How to build an inventory management system that scales," CockroachLabs Blog, 2023.

[9] DEV Community, "Designing a Scalable and Real-Time Messaging System," DEV Community, 2023.

[10] Redis Documentation, "Geospatial queries," Redis Docs, 2023. [Online]. Available: https://redis.io/docs/latest/

[11] TiDB, "CAP Theorem Explained: Balancing Consistency, Availability & Partition Tolerance," PingCAP, 2023.

[12] Android Developers, "Create and monitor geofences," Android Developer Documentation, 2023.