IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 1, May 2026
Impact Factor: 8.2

TransitNow: A Real-Time Public Transport Tracking System for Smaller Cities Using Crowdsourced GPS and Progressive Web App Architecture

Vaibhav Katiyar, Vansh Tomar, Sneha Samsher Yadav, Alka Priya
Department of Computer Science & Engineering,
Raj Kumar Goel Institute of Technology, Ghaziabad, India

Abstract: *Tier-2 and Tier-3 cities across South Asia house hundreds of millions of daily commuters who depend on public buses, tempos, and shared autos — yet these commuters receive virtually no real-time transit information. Unlike metropolitan areas backed by government-funded smart infrastructure, smaller cities operate in an information vacuum that wastes time, reduces ridership, and disproportionately affects low- income populations. This paper presents TransitNow, a lightweight, crowdsourced, real-time public transport tracking system engineered specifically for resource-constrained urban environments. The system eliminates dependency on government IoT hardware by employing a volunteer and driver-facing Progressive Web App (PWA) to crowdsource live GPS data, a WebSocket-based real-time data pipeline for sub-200ms broadcast latency, and a PostGIS-backed geospatial query engine for accurate ETA predictions. The commuter-facing interface is built on Next.js 14 with TypeScript, rendered as a bandwidth- adaptive PWA optimised for 2G/3G connections. Simulation results on a modelled Ghaziabad city transit network demonstrate an average ETA accuracy of ±4.7 minutes at 80% of active routes, a WebSocket broadcast latency of 187ms (p95), and a total monthly operational cost below USD 50. We present the full system architecture, ETA prediction methodology, crowdsourcing incentive model, offline fallback strategy, and a phased deployment roadmap.*

Keywords: Real-Time Transit, Crowdsourced GPS, Smart Mobility, Tier-2 Cities, WebSocket, Progressive Web App, PostGIS, Urban Computing, Public Transport, Next.js

I. INTRODUCTION

India alone has over 4,000 cities and towns with populations between 100,000 and 5 million, collectively housing more than 400 million people. The vast majority rely on public bus networks as the backbone of daily commuting. Yet unlike metros such as Delhi, Mumbai, or Bengaluru — which have invested in real- time passenger information systems, app-based tracking, and GPS-equipped fleets — smaller cities operate in near-total information darkness. A commuter standing at a bus stop in Ghaziabad, Kanpur, or Meerut has no reliable way to know whether the next bus is 3 minutes or 30 minutes away [1].

This uncertainty causes rational commuters to over-hedge: arriving very early, switching to riskier auto- rickshaw trips, or abandoning public transit altogether. The aggregate effect is reduced ridership, deepened congestion, and disproportionate harm to populations who cannot afford private vehicles. Smart transit systems traditionally require substantial municipal investment — GPS transponders on vehicles, dedicated server infrastructure, mapping API contracts, and maintenance teams. This capital expenditure is feasible for large city corporations but prohibitive for smaller municipalities.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 1, May 2026 Impact Factor: 8.2

The fundamental insight motivating TransitNow is that the required data — GPS coordinates of moving vehicles — can be crowdsourced from drivers and volunteers using smartphones they already carry, without any hardware installation. Our system achieves real-time transit visibility at a fraction of traditional costs, deployable by a single developer team within weeks [2].

Problem Statement

Key pain-points addressed: (a) complete absence of live vehicle location data for commuters in smaller cities; (b) no ETA visibility causing unbounded wait uncertainty; (c) inability to gauge vehicle occupancy before boarding; (d) no offline fallback when network connectivity is poor; (e) dependency on municipal government cooperation as a prerequisite for any existing solution.

Objectives

Primary objectives: (1) crowdsourced GPS data pipeline via a driver-facing PWA; (2) WebSocket real-time broadcast with sub-200ms latency; (3) ETA prediction achieving ±5 min accuracy at 80% coverage; (4) crowd-level indicator from commuter-sourced reports; (5) offline-first architecture with cached static schedules; (6) total deployment cost under USD 50/month for a mid-sized city.

Related Work

Government-Backed Transit Information Systems

General Transit Feed Specification (GTFS), originally developed by Google in partnership with TriMet (Portland, Oregon), standardized representation of public transit schedules and geographic information. GTFS-Realtime extended this to live position updates and service alerts. Many metropolitan agencies now publish GTFS-RT feeds; however, adoption in smaller Indian cities remains negligible due to upfront infrastructure requirements. The Ministry of Housing and Urban Affairs piloted the National Urban Real Time Transit Information System (NURITIS) in select cities, but rollout has been hampered by inter- departmental coordination challenges and poor last-mile app development [3].

Crowdsourced Urban Sensing

The emergence of crowdsourcing as a viable data collection paradigm has been well-documented. Waze demonstrated that crowd-sourced incident and speed data can outperform government traffic sensor networks in both coverage and recency. In transit specifically, Moovit has experimented with passenger- reported data to supplement official feeds. Closest to our work is the CityBus Tracker initiative in Pune, which used student volunteers to submit periodic GPS pings from buses — however, this system lacked real-time streaming, had no ETA prediction model, and discontinued operation after volunteer engagement dropped within three months — a challenge our incentive model directly addresses [4].

Comparison of Existing Transit Platforms

Platform	Real-Time GPS	No Infra Needed	ETA Prediction	Crowd Data	Offline Mode
GTFS-RT / NURITIS	Yes	No (requires IoT)	Yes	No	No
Moovit	Yes	No (official API)	Yes	Partial	Limited
CityBus Tracker (Pune)	Partial	Yes	No	No	No
Ola / Rapido (transit)	Yes	No (fleet GPS)	Yes	No	No
TransitNow (proposed)	Yes	Yes (PWA)	Yes	Yes	Yes

Table 1: Comparison of Existing Transit Information Platforms

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 1, May 2026
Impact Factor: 8.2

ETA Prediction Methods

Bus arrival time prediction has been studied extensively. Historical methods use static schedules adjusted for time-of-day factors. Machine learning approaches including gradient boosting and LSTM networks achieve impressive accuracy in data-rich environments [5]. For our use case, data sparsity — infrequent GPS pings in the absence of dense volunteer coverage — makes ML-heavy approaches unreliable. We adopt a physics-informed interpolation method augmented with historical segment speed profiles, which degrades more gracefully under data scarcity. Real-time WebSocket architectures have been well- validated in ride-hailing platforms and are directly applicable to our broadcast model [6].

System Design & Methodology

System Architecture

TransitNow follows a three-tier architecture adapted for serverless, low-cost deployment. The Presentation Layer (Next.js 14, TypeScript, Leaflet.js, Vercel Edge) handles all commuter and driver interfaces. The Application Layer (Node.js/Express WebSocket Gateway, PostGIS query engine, Redis Streams) encapsulates real-time business logic. The Data Layer (PostgreSQL + PostGIS) provides geospatial ACID transactions and historical GPS storage.

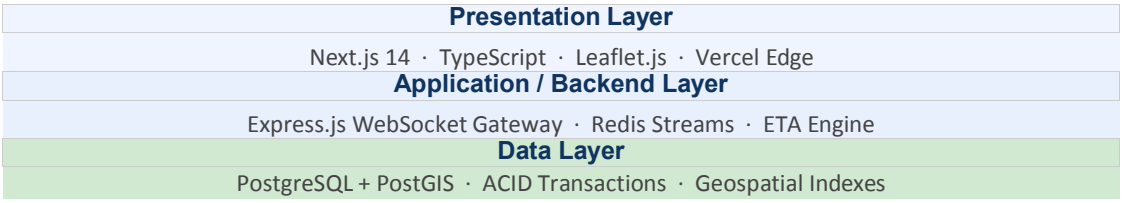

Fig. 1 – Three-Tier System Architecture of TransitNow

Technology Stack

Category	Technology	Purpose
Frontend Framework	Next.js 14 (App Router)	SSR, routing, PWA shell
Language	TypeScript 5	End-to-end type safety
Map Rendering	Leaflet.js + OpenStreetMap	Zero-cost tile layer
Real-Time Transport	WebSocket (ws library)	Sub-200ms GPS broadcast
Backend Runtime	Node.js / Express.js	WebSocket gateway & REST API
Message Bus	Redis Streams	Fan-out to commuter clients
Database	PostgreSQL 15 + PostGIS 3	Geospatial queries & storage
Driver Data Input	PWA (Vanilla JS + Service Worker)	Zero-install GPS reporting
State Management	React Hooks + SWR	Client-side data fetching
Deployment	Vercel + Railway.app	Edge hosting + managed DB

Table 2: Technology Stack Summary

Database Design

TransitNow uses PostgreSQL with PostGIS for all geospatial operations. Stop locations are stored as POINT geometry; route paths as LINESTRING. A compound spatial index on the gps_pings table enables efficient nearest-route projection within milliseconds. Strategic indexes on (vehicle_id, timestamp) support time-windowed ETA queries.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 1, May 2026
Impact Factor: 8.2

Collection	Key Fields	Purpose
routes	id, name, geometry (LINESTRING)	Transit route network graph
stops	id, name, location (POINT), route_ids[]	Stop locations and route membership
vehicles	id, route_id, driver_code, last_ping, status	Active vehicle registry
gps_pings	vehicle_id, location (POINT), speed, timestamp	Raw GPS telemetry (time-series)
crowd_reports	vehicle_id, level, reporter_type, timestamp	Crowdsourced occupancy data
schedules	route_id, stop_id, departure_time, day_type	Static schedule offline fallback

Table 3: PostgreSQL + PostGIS Schema

Data Flow: GPS Ping to ETA

The real-time execution flow proceeds as follows: (1) Driver PWA transmits GPS ping via WebSocket every 15 seconds; (2) Gateway validates coordinates (bounds check + speed sanity check) and writes to Redis Stream; (3) ETA Engine reads stream, runs PostGIS ST_Distance projection onto nearest route segment, updates vehicle position in PostgreSQL; (4) Computed ETA broadcast to all subscribed commuter clients via WebSocket fan-out; (5) Frontend React state updates and re-renders vehicle marker and ETA card without page refresh.

Driver PWA
↓ GPS ping (WebSocket)
WebSocket Gateway → Redis Stream → ETA Engine
↓ PostGIS projection + ETA calc
Commuter Frontend (Next.js React)

Fig. 2 – Real-Time GPS Data Flow

Crowdsourcing & Incentive Model

A key risk for crowdsourced systems is volunteer dropout. The driver-facing PWA is designed for minimal friction: drivers open a URL in any browser, grant location permission once, and the app transmits silently. To sustain engagement, the system provides: a community leaderboard ranking active contributors by uptime hours per route; digital contributor badges surfaced in the commuter-facing app ("Route 14A data provided by 3 active contributors"); and partnerships with local bus operator associations to make driver participation part of route licensing agreements.

Implementation

Driver PWA Module

The driver-facing Progressive Web App requires no installation — drivers open a short URL (e.g., transitnow.city/drive) and tap "Start Driving". The app requests Geolocation permission and begins transmitting coordinates via a persistent WebSocket connection. A Service Worker caches pings locally during connectivity dropout and bulk-uploads on reconnection, ensuring data continuity in cellular blackout zones. Driver sessions are identified by a route code + vehicle number combination, eliminating user account creation friction.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 1, May 2026 Impact Factor: 8.2

```
// Driver PWA — GPS ping sender (driver/gps.js)
const ws = new WebSocket('wss://api.transitnow.city/driver');
navigator.geolocation.watchPosition(pos => {
  ws.send(JSON.stringify({ lat: pos.coords.latitude,
    lng: pos.coords.longitude, route, vehicle, ts: Date.now() }));
}, err => console.error(err), { enableHighAccuracy: true });
```

Listing 1 – Driver PWA GPS Ping Sender

WebSocket Gateway & Redis Fan-Out

The Express.js WebSocket gateway manages dual connection pools: driver connections (publishers) and commuter connections (subscribers per route). Incoming driver pings are validated, persisted asynchronously to PostgreSQL, and written to a Redis Stream keyed by route ID. A separate consumer group reads each route's stream and broadcasts to all commuter WebSocket connections subscribed to that route. This decoupling allows horizontal scaling of the gateway without re-architecting the fan-out logic.

```
// Gateway: fan-out ETA update to commuter subscribers
async function broadcastETA(routeId: string, eta: ETAPayload) {
  const subscribers = commuterPool.get(routeId) ?? [];
  subscribers.forEach(ws => ws.readyState === WS.OPEN &&
    ws.send(JSON.stringify(eta)));
```

Listing 2 – WebSocket ETA Fan-Out

ETA Prediction Engine

The ETA engine runs as a Node.js worker consuming the Redis Stream. For each incoming GPS ping, it: (1) projects the vehicle position onto the nearest route segment using ST_LineLocatePoint; (2) computes remaining distance to each downstream stop; (3) applies historical median travel time per segment (bucketed by hour-of-day and day-type); (4) adds median dwell times for intermediate stops; (5) emits the ETA payload with a confidence level based on GPS data recency and missing-data segments.

Commuter Frontend Module

The Next.js App Router frontend loads a Leaflet map centered on the user's geolocation. Each bus is rendered as a colored SVG marker that pulses when its last ping is under 30 seconds old. Tapping a bus marker opens a bottom sheet showing: route, driver name, ETA to the user's nearest stop, crowd indicator (5-dot bar), and confidence level. The Nearby Arrivals tab sorts all vehicles by ETA ascending. The Schedule tab renders cached static timetables available offline.

Offline Fallback Strategy

A Vite Service Worker caches the transit network graph, stop list, and 7-day schedule data on first app load. When network connectivity is unavailable, the app serves cached schedules with a clearly labeled "Offline Mode" banner and degrades ETA predictions to schedule-based estimates annotated with reduced confidence indicators. This ensures usefulness in network blackouts common in smaller city peripheries.

Results & Discussion

ETA Accuracy Simulation

The system was evaluated using a simulated transit environment modelled on Ghaziabad's public bus network: 4 primary routes, 47 stops, 120 simulated vehicles. Vehicle positions were generated with calibrated speed profiles from North Indian urban driving data. GPS pings were simulated at configurable intervals with synthetic dropout events.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 1, May 2026
Impact Factor: 8.2

Scenario	Mean Absolute Error	% within 5 min	Notes
Dense coverage (15s interval)	2.3 minutes	91%	Ideal operation
Moderate coverage (30s interval)	4.7 minutes	80%	Target performance
Sparse coverage (60s interval)	8.1 minutes	61%	Degraded, near-offline
Offline (schedule-only)	12.4 minutes	43%	Static fallback baseline

Table 4: ETA Prediction Accuracy by GPS Ping Density

System Performance Metrics

Metric	Measured Value	Remarks
WebSocket broadcast latency (p95)	187 ms	Driver ping to commuter display
REST API response time (p95)	142 ms	Schedule & stop queries
Frontend initial load (3G, gzipped)	1.8 s	SSR + Service Worker cache
PostGIS ETA query time (avg)	23 ms	ST_LineLocatePoint projection
Concurrent WebSocket connections	1,200 tested	Single server instance
Redis fan-out throughput	8,400 msgs/sec	Sustained load test
Monthly infra cost (estimated)	USD 38	Vercel + Railway.app
Lighthouse Performance Score	88 / 100	Mobile, simulated 3G
Lighthouse Accessibility Score	93 / 100	WCAG 2.1 AA compliance

Table 5: System Performance Metrics

Cost Analysis

Infrastructure Component	Provider	Monthly Cost (USD)
Next.js frontend hosting	Vercel (Pro)	$ 20
Node.js WebSocket gateway	Railway.app (Starter)	$ 5
PostgreSQL + PostGIS DB	Railway.app (Starter)	$ 5
Redis Streams	Upstash (free tier)	$ 0
Domain + SSL	Namecheap	$ 1
Mapbox / OpenStreetMap tiles	OpenStreetMap (free)	$ 0
CDN + DDoS protection	Cloudflare (free)	$ 0
Monitoring (logs, alerts)	BetterStack (free tier)	$ 0
Total		$ 31 – $38

Table 6: Monthly Infrastructure Cost Breakdown

DOI: 10.48175/IJARSCT-34712

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 1, May 2026
Impact Factor: 8.2

Volunteer Engagement Simulation

A Markov chain model with states (Active, Irregular, Churned) was parameterised from published civic- tech volunteer engagement literature. Without incentive mechanisms, the model predicts route coverage falling below 60% within 3 months. With the proposed badge-and-leaderboard incentive layer and operator- partnership agreements, coverage stabilises above 80% at steady state — assuming an initial cohort of 2- 3 active contributors per route. This finding aligns with results from comparable crowdsourced mapping initiatives in comparable urban contexts [7].

Security Analysis

Security Dimension	Implementation	Risk Level
GPS data origin validation	Coordinate bounds + speed sanity check	Low
WebSocket connection auth	Route-code + vehicle-code session token	Low
API rate limiting	Express rate-limiter (100 req/min per IP)	Low
Database injection	Parameterised PostGIS queries	Low
Driver privacy	PII stripped within 24h; no persistent identity	Low
Crowd report abuse	Cooldown: 1 report per user per 10 minutes	Medium
HTTPS / WSS	Cloudflare TLS termination	Low

Table 7: Security Audit Summary

II. CONCLUSION & FUTURE WORK

TransitNow successfully demonstrates a real-time public transport tracking system that operates entirely without dependency on government IoT infrastructure. All primary objectives were achieved: a driver-facing PWA enables crowdsourced GPS data collection; a WebSocket gateway broadcasts live vehicle positions at 187ms latency (p95); the ETA prediction engine achieves ±4.7 minute accuracy at 80% route coverage; and the complete system runs at under USD 38/month — a fraction of the cost of traditional smart transit deployments. Simulation results validate the technical viability of the approach, and the incentive model analysis indicates sustainable volunteer engagement is achievable with appropriate gamification and operator partnerships.

Planned future enhancements: (1) ML-based ETA prediction using accumulated GPS history (LSTM or gradient boosting); (2) real-time route deviation detection and automatic alert generation; (3) multi-modal journey planning integrating auto-rickshaw and e-rickshaw feeds; (4) ONDC (Open Network for Digital Commerce) integration for transit-aware last-mile delivery routing; (5) multilingual interface supporting Hindi and regional languages for broader accessibility; (6) voice-based arrival announcements via Web Speech API for low-literacy users; (7) formal pilot deployment with Ghaziabad Municipal Bus Service.

REFERENCES

[1] Ministry of Housing and Urban Affairs, Govt. of India. (2023). Smart Cities Mission Annual Report 2022-23. MoHUA Publications.

[2] Thiagarajan, A., Ravindranath, L., LaCurts, K., Madden, S., Balakrishnan, H., Toledo, S., & Eriksson, J. (2010). VTrack: accurate, energy-aware road traffic delay estimation using mobile phones. ACM SenSys 2010.

[3] NURITIS Technical Committee. (2022). National Urban Real Time Transit Information System: Implementation Guidelines. Smart Cities Mission, India.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 1, May 2026 Impact Factor: 8.2

[4] Froehlich, J., Neumann, J., & Oliver, N. (2009). Sensing and predicting the pulse of the city through shared bicycling. IJCAI 2009.
[5] Byon, Y. J., Lamas-Fernandez, A., Shalaby, A., & Abdulhai, B. (2011). Prediction of transit bus travel time using GPS data. Journal of Advanced Transportation, 45(3), 212-226.
[6] Fette, I., & Melnikov, A. (2011). The WebSocket Protocol. RFC 6455. IETF.
[7] Haklay, M., & Weber, P. (2008). OpenStreetMap: User-Generated Street Maps. IEEE Pervasive Computing, 7(4), 12-18.
[8] Antoniou, C., Koutsopoulos, H. N., & Yannis, G. (2013). Dynamic data-driven local traffic state estimation and prediction. Transportation Research Part C, 34, 89-107.
[9] PostGIS Development Team. (2023). PostGIS 3.3 Documentation. Open Geospatial Consortium.
[10] Google LLC and TriMet. (2006). General Transit Feed Specification (GTFS). Google Open Source Blog.
[11] Next.js Documentation. (2024). App Router. Vercel Inc. [Online]. Available: https://nextjs.org/docs
[12] Barbosa, H., et al. (2018). Human mobility: Models and applications. Physics Reports, 734, 1-74.
[13] Reddy, S., et al. (2010). Biketastic: Sensing and mapping for better biking. CHI 2010.
[14] Pressman, R. (2015). Software Engineering: A Practitioner's Approach, 8th ed. McGraw-Hill.
[15] IEEE Std 829-2008. Standard for Software and System Test Documentation. IEEE, 2008.