

28. European Commission. Industry 5.0 Community of Practice: Final report of the pilot phase 2023–2024. Brussels, 2024.
29. European Commission. ERA industrial technologies roadmap on human-centric research and innovation for the manufacturing sector. Publications Office of the European Union, 2024. DOI: 10.2777/0266.
30. Матеріали авторської розробки «Спосіб проекційного керування просторовим розміщенням матеріальних об'єктів з адаптивним переплануванням». 2026.
31. European Commission. Data Governance Act. URL: <https://digital-strategy.ec.europa.eu/en/policies/data-governance-act> (дата звернення: 30.07.2026).
32. European Commission. Data Act. URL: <https://digital-strategy.ec.europa.eu/en/policies/data-act> (дата звернення: 30.07.2026).
33. European Commission. AI Act: Regulatory framework for artificial intelligence. URL: <https://digital-strategy.ec.europa.eu/en/policies/regulatory-framework-ai> (дата звернення: 30.07.2026).
34. European Commission. NIS2 Directive: securing network and information systems. URL: <https://digital-strategy.ec.europa.eu/en/policies/nis2-directive> (дата звернення: 30.07.2026).
35. European Commission. Cyber Resilience Act. URL: <https://digital-strategy.ec.europa.eu/en/policies/cyber-resilience-act> (дата звернення: 30.07.2026).
36. ISO/IEC 30173:2023. Digital twin — Concepts and terminology. Geneva: International Organization for Standardization, 2023.

4.2. Unified Digital Platform for Suburban and Intercity Bus Transport Based on the PSO Model with Integrated Anti-Corruption Mechanisms

Anton FESENKO

Maksym DEMYDENKO

National University Yurii Kondratiuk Poltava Polytechnic

The current state of Ukraine's suburban and intercity passenger bus transport

market is characterized by a large shadow economy, fragmented service delivery and low passenger comfort.

The traditional model – where operators run purely on a commercial basis – leads to the degradation of socially critical yet low-margin routes, particularly in rural areas. Direct cash transactions with drivers make it impossible to accurately track passenger flows, ensure full tax compliance or provide transparent state compensation for subsidized passenger categories.

In the context of European integration and post-war infrastructure reconstruction, Ukraine's strategic priority is transitioning to the European PSO (Public Service Obligation) model. Under this model, the state procures transport services via open tenders, paying operators a fixed fee based on transport work completed rather than ticket sales. However, rolling out the PSO model amid severe corruption risks demands strict oversight, as conventional inspection methods reliant on manual field audits fall short due to human collusion.

Addressing this challenge requires developing a comprehensive information system built on the Zero Trust Architecture paradigm (Zero Trust Architecture, 2020), where trip execution verification and passenger counting are automated through the cross-monitoring of digital ticket sales, geospatial data and hardware telemetry.

The aim of this study is to design the architecture and develop a software model for a unified digital suburban and intercity bus transport platform that automates PSO tender management and ticketing while providing algorithmic, anti-corruption trip execution control.

To achieve this aim, the following research objectives were set:

- Design a software architecture that segments business logic across isolated portals: passenger, carrier, state regulator and bus station;

- Develop a hybrid passenger identity verification model (with potential integration into the Diia application) to ensure targeted provision of social fare discounts and eliminate speculation;

- Formulate mathematical anomaly detection algorithms based on the fusion of Automated Passenger Counting (APC) sensor data and route geofencing telemetry.

The contemporary suburban and intercity passenger transport market in Ukraine

operates under heavy fragmentation and technological obsolescence. Historically, cash handed directly to the driver has remained the predominant payment method on regional routes. This paradigm fosters a widespread shadow economy with corrosive consequences across the transport ecosystem: for the state, compliant carriers and passengers alike.

State regulatory bodies lack the tools for objective passenger-flow analytics. Operators self-report passenger numbers, opening substantial leeway for manipulation: companies underreport actual revenues to evade taxes while inflating the number of subsidized passengers carried to extract ungrounded state budget reimbursements. Under these conditions, any infrastructure planning or route network optimization relies on distorted baseline data.

The purely commercial orientation of the market drives operators to concentrate resources strictly on high-yield corridors, such as connections between major urban centers. Conversely, routes serving remote rural communities are operated on a residual basis – relying on obsolete rolling stock that fails environmental and accessibility standards, paired with critically low service frequencies. This systemic issue undermines citizens' fundamental right to mobility.

While several disparate commercial ticketing platforms exist on the market, they do not resolve the sector's fundamental problem. Existing systems are built on the premise of complete trust in the operator, assuming that drivers or conductors will diligently use mobile validators or handheld terminals to register every passenger. However, as transit experience in transition economies demonstrates, whenever a financial incentive exists – namely, the opportunity to pocket cash – the human factor invariably becomes the weakest link. Mere ticketing digitization without hardware-level oversight and a Zero Trust paradigm simply automates chaos without eliminating underlying corruption risks.

Resolving these structural failures requires overhauling the very model of engagement between the state and transport operators: specifically, transitioning to the European public service framework.

To overcome this regional transit crisis, the strategic vector for Ukraine's transport infrastructure is the transposition of European public passenger transport

service rules grounded in the PSO framework under Regulation (EC) No 1370/2007 (Public Service Obligation) (Regulation (EC) No 1370/2007).

The core premise of PSO is that ensuring population mobility is recognized as a social obligation of the state rather than a purely commercial activity. Instead of an operator generating revenue solely through ticket sales – a model that renders rural routes unprofitable – the state or local transport authority acts as the sole contracting authority. The transport authority conducts a competitive tender and awards a contract under which the carrier receives a fixed fee for completed "transport work" (actual vehicle-kilometers operated according to the published timetable).

In this paradigm, all ticket revenue is consolidated into a single digital platform (deposited directly into the organizing authority's accounts), stripping the carrier of any financial incentive to conceal passenger volumes or collect unrecorded cash. However, the viability of such a framework hinges entirely on the state's capacity to audit contract compliance accurately: verifying whether the scheduled trip was actually executed, adhered to the timetable, and serviced designated stops.

Yet, despite adopting a Zero Trust paradigm at the vehicle level (deploying hardware sensors to monitor onboard crew behavior), the information system operates within a broader socio-political context. This exposes the platform to insider threat vectors originating from executive leadership, tender evaluation committees and system administrators.

According to secure system design methodologies, software architecture alone cannot eliminate the corrupt intent of administrative personnel. Its function, however, is to minimize the attack surface and enforce an immutable digital audit trail.

The primary administrative threat vectors and their architectural mitigations include:

1. Tender specification tampering. Attack scenario: an official configures tender requirements within the system (e.g., passenger capacity thresholds, environmental standards) tailored specifically so only a predetermined operator can qualify. System mitigation: operating deterministically, the platform cannot evaluate subjective corrupt intent when parameters are defined. However, the system maintains an immutable, cryptographically verifiable log of tender creation and

revision history, enabling anti-corruption oversight bodies (such as NABU and SAPO) to perform retrospective audits on discriminatory criteria.

2. Unauthorized penalty waivers. Attack scenario: hardware sensors log a non-compliance event (e.g., unticketed passengers), triggering the platform to issue an automated fine. A corrupt official with administrative privileges manually voids the penalty in the system. Architectural mitigation: the platform utilizes the Apache Kafka event broker as an immutable event-sourced store. Administrators are structurally prevented from deleting violation records from the database; they can only append a new compensation event documenting the cancellation rationale, permanently bound to their digital signature. This pulls the corrupt act out of the shadows and into an auditable, transparent plane.

Engineering this information system demands a fine balance between high throughput, operational resilience and deployment simplicity. For a regional Minimum Viable Product (MVP), adopting a traditional microservices architecture often introduces excessive infrastructure overhead, network latency penalties and complex distributed transaction integrity challenges.

To implement the platform's backend, the "Modular Monolith" architectural pattern (Architect Modern Web Applications with ASP.NET Core and Azure) based on the .NET 10 platform was selected. This approach involves deploying the system physically as a single server process with a unified PostgreSQL database, while enforcing strict logical separation of code into independent bounded contexts according to Domain-Driven Design (DDD) principles.

The system architecture encapsulates the following isolated modules:

- Ticketing Module: responsible for seat reservations, ticket sales and multimodal route calculation;
- Tender Management Module: handles PSO contract administration, route publication and the registry of carriers' assigned fleets;
- IoT Engine: a high-throughput module designed for ingesting and analyzing geospatial telemetry alongside hardware sensor data.

Inter-module communication occurs exclusively via strictly defined in-code

interfaces or through an event broker.

Upon successful completion of the pilot project in Poltava Oblast and subsequent nationwide rollout, the modular monolith architecture enables seamless extraction of any domain (such as the IoT engine) into a standalone microservice.

The user interface architecture is tailored to the distinct needs of system actors while minimizing overall development time. Rather than engineering numerous fragmented front-end applications, the system is streamlined into two core portals developed using Blazor technology.

The first application is a universal passenger portal and self-service terminal. Built on a responsive design pattern, it serves a dual role by addressing the requirements of both online users and passengers purchasing tickets on-site.

Its mobile mode functions as a standard web application in smartphone browsers, allowing passengers to select routes and generate digital QR tickets. The kiosk mode – running off the identical codebase – operates on hardware self-service terminals deployed across bus stations. This entirely removes the overhead of developing and maintaining a dedicated ticket clerk workstation interface.

To implement the MVP and ensure maximum accessibility (particularly for elderly passengers), the authorization flow was radically streamlined. Rather than introducing complex integrations with state registries or the Diia platform, a user or terminal operator needs only to enter a full name and phone number to complete a ticket purchase.

The second application is the state regulator control dashboard. This portal is designed exclusively for dispatchers, auditors and transport authority officials. It serves as the primary monitoring hub, consolidating all platform telemetry and analytics.

Because the dashboard demands real-time rendering of large, dynamic datasets, it was built on the Blazor Server architecture. In this model, all UI processing executes on the server, while the client browser receives instantaneous DOM-tree updates over a persistent binary connection. This allows the system to push critical alerts directly to an inspector's screen the exact millisecond a background worker detects passenger-count discrepancies, completely bypassing the need for complex client-side polling

mechanisms.

A major hurdle in ensuring rural mobility is the economic unfeasibility of running direct routes between every settlement. Traditional commercial models neglect this constraint, forcing travelers to coordinate their own transfers. The proposed system resolves this algorithmically by adopting a hub-and-spoke network topology.

In this topology, major bus stations (such as the central bus station in Poltava) act as transit hubs, while feeder routes serving surrounding rural communities function as spokes. The integrated ticketing module automatically constructs multimodal route itineraries. When a passenger searches for a route from "Village A to Village B," the platform traverses the graph of available PSO services to produce an itinerary featuring a guaranteed transfer in Poltava, factored against transfer buffer times.

Instead of navigating fragmented transactions, the passenger receives a single unified digital ticket. From the perspective of the state as the contracting authority under PSO agreements, this architecture maximizes cost-effectiveness: it eliminates the need to subsidize underutilized direct routes between rural settlements while securing consistent patronage flows across central transit hubs.

Graph-traversal and route-planning algorithms rely on complex, aggregation-heavy SQL queries executed against the PostgreSQL database, continuously cross-referencing thousands of timetables, intermediate stops and real-time seat inventories. During peak load windows – such as early morning hours when ticket purchasing spikes – this generates acute database contention.

To maintain high availability and low read latency, the system implements a multi-tier caching strategy powered by the hybrid caching framework built into .NET 10, backed by Redis as the distributed tier.

This pattern establishes a two-level (L1/L2) data access architecture. The L1 tier (in-memory cache) serves ultra-low-latency lookups for high-frequency routes directly from local server RAM. The L2 tier (distributed Redis cache) keeps cache states coherent across backend application instances, offloading timetable datasets from the primary PostgreSQL instance.

Traditional dispatch and fare-collection monitoring systems rely on a presumption of trust. The state or contracting authority trusts the carrier to operate the service according to the timetable, while the carrier trusts the driver or conductor to ensure every passenger holds a valid ticket. As established in previous sections, the persistence of cash transactions renders this reliance on trust an inevitable driver of corruption and state budget leakages during PSO subsidy disbursements.

To address this fundamental flaw, the architecture of the proposed platform is anchored in the Zero Trust paradigm. In cybersecurity, this principle translates to "never trust, always verify." In the domain of passenger transport, it manifests as the following operational axiom: the information system rejects any human-generated or manually entered data unless it is corroborated by hardware telemetry.

The backbone of this architecture is the platform's integration with onboard Internet of Things (IoT) hardware (Buyya et al., 2016), specifically Automated Passenger Counting (APC) systems.

Rather than relying on ticket validation counts from onboard card readers – which a driver can easily disable – system deploys optical sensors mounted above every vehicle doorway. Modern APC sensors leverage 3D stereoscopic cameras or Time-of-Flight (ToF) technology. These devices reliably register doorway threshold crossings, determine directional movement (boarding vs. alighting) and distinguish adult passengers from children or bulky luggage by analyzing thermal and volumetric profiles.

Passenger volume metrics alone hold little value without spatial and temporal context. Consequently, the onboard hardware architecture aggregates three independent data streams via a unified IoT gateway:

1. APC metrics: the precise count of passengers boarding and alighting.
2. Door status: an open/closed binary signal pulled directly from the bus telemetry.
3. GPS positioning: high-precision vehicle coordinates and instantaneous velocity.

The onboard gateway continuously ingests these data points, packages them into lightweight JSON payloads and transmits them at high frequency (e.g., every 5-10

seconds or triggered by a "door opened" event) across a 4G/LTE cellular network to the central platform API.

A critical architectural consideration is the full autonomy of the onboard hardware. The driver lacks physical access to disconnect or power down sensors without breaking tamper-evident seals or de-energizing the entire system – an action instantly flagged by the backend as a critical incident, automatically freezing payment disbursement for the service.

The platform's mathematical engine executes as a background worker, continuously cross-referencing incoming telemetry against geospatial coordinates. Any logical discrepancy across these data streams is classified as an attempted fraud vector. Below are three primary algorithmic "traps" embedded within the system architecture.

1. Fare Evasion Trap (Unticketed Passenger Detection). This algorithm identifies the classic scenario where a driver accepts cash directly from a passenger without requiring them to scan a ticket. The system treats each bus stop as a distinct episode defined by the door-open interval.

The algorithm cross-references the number of individuals physically boarding the cabin against the volume of digital tickets validated within that exact time window. Only a minimal tolerance threshold is permitted (e.g., 1-2 passengers) to account for infants carried in arms or edge cases in optical sensor performance. If physical boardings exceed scanned tickets beyond this allowable threshold, the background worker instantly triggers a fare-evasion incident.

An additional layer of validation monitors residual onboard capacity: if the calculation indicates that passengers remain onboard at the final terminus after all passengers should have alighted, it serves as an indicator that the driver intentionally obstructed the sensor during egress to conceal unticketed riders.

2. Equipment Sabotage/Tampering Trap. Under stringent hardware oversight, drivers may attempt to physically incapacitate the APC sensor – for example, by taping over the camera lens or obstructing its field of view. This algorithm detects sabotage through a logical paradox, using the transaction log as definitive proof.

The system triggers when the optical sensor reports zero physical boardings, yet the central server records successful ticket validations from the onboard validator. This yields a physical impossibility: tickets are being scanned by passengers who, by sensor records, never boarded the vehicle. Detecting this pattern automatically flags the vehicle as hardware-compromised, establishing formal grounds to immediately suspend PSO subsidy disbursements for that route pending a technical audit.

3. Off-Route/Unauthorized Stop Trap. To collect unrecorded cash, drivers frequently make unauthorized stops outside designated bus stations and stops – at road intersections, shoulders or commercial plazas. Route compliance monitoring is enforced by constructing virtual geofences and calculating proximity based on GPS coordinates.

The algorithm continuously correlates the bus's real-time geolocation with its door-status telemetry. For each authorized stop in the route manifest, a designated buffer radius is defined (e.g., a 50-meter geofence centered on the boarding platform). If the telemetry registers door actuation outside all authorized geofences, the platform classifies the event as an unauthorized passenger boarding or alighting maneuver.

Processing continuous telemetry streams (GPS coordinates, door statuses, APC sensor metrics, validator transactions) across hundreds of buses concurrently places extreme load on the server infrastructure. To ensure platform resilience and scalability, the backend architecture incorporates the Apache Kafka distributed event streaming platform. Within this system, Kafka performs two mission-critical functions: it guarantees the mathematical precision of analytical algorithms and produces a tamper-proof digital audit trail.

The core challenge of ingesting IoT telemetry within multithreaded systems is the risk of out-of-order event delivery. Due to cellular network latency variations, signals dispatched from onboard hardware may reach the server with minor timing skews and be consumed by background workers at differing rates. For instance, if the system processes a "5 passengers boarded" event prior to processing an out-of-sequence "doors opened" event, the mathematical model would generate a false-positive fraud alert.

Traditional message brokers (such as RabbitMQ) do not guarantee strict ordering semantics under parallel consumer pool workloads. Apache Kafka resolves this structural constraint at the architectural layer via partition key routing.

When streaming any telemetry event, the system designates the bus unique identifier as the partition key. This guarantees that all events originating from a given vehicle (movement, stop dwell, ticket scans, door status transitions) are consistently routed to the same partition and processed sequentially by consumers in the exact chronological order of their physical occurrence. Consequently, the anomaly detection algorithms described in the previous section receive a fully coherent, chronologically validated context for real-time analysis.

The second – and from a public administration standpoint, paramount – function of Apache Kafka is the mitigation of the administrative threat vectors outlined in the opening section (such as unauthorized penalty voids and corrupt influence on regulatory officials).

Conventional relational databases remain vulnerable to tampering by privileged accounts or database administrators. A corrupt official could technically execute an UPDATE or DELETE query to expunge records of carrier infractions or void automatically calculated penalties without leaving a trace.

Apache Kafka is grounded in the Event Sourcing paradigm and operates as an append-only transaction log [5]. Data written to Kafka is inherently immutable.

Within the developed platform, every critical state transition – such as an algorithm flagging unticketed passengers and issuing a fine, or an administrator manually altering the status of that fine – is concurrently published as an independent event to a dedicated, isolated topic. If an administrator attempts to void a sanction, the system does not delete the infraction entry; instead, it appends a new compensating event cryptographically bound to the administrator's digital signature and timestamped down to the exact millisecond.

REFERENCES FOR SUBSECTION 4.2

1. Regulation (EC) No 1370/2007 of the European Parliament and of the Council of 23 October 2007 on public passenger transport services by rail and by road and repealing Council Regulations (EEC) Nos 1191/69 and 1107/70. Official Journal of the European Union. [Электронный ресурс]. Режим доступа: <https://eur-lex.europa.eu/eli/reg/2007/1370/oj/eng>
2. Architect Modern Web Applications with ASP.NET Core and Azure. Microsoft Docs [Электронный ресурс]. Режим доступа: <https://learn.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/>
3. Zero Trust Architecture (NIST Special Publication 800-207). Rose, S., Borchert, O., Mitchell, S., & Connelly, S. (2020). National Institute of Standards and Technology [Электронный ресурс]. Режим доступа: <https://www.nist.gov/publications/zero-trust-architecture>
4. Internet of Things: Principles and Paradigms. Buyya, R., Dastjerdi, A. V. (Eds.). (2016).
5. The Log: What every software engineer should know about real-time data's unifying abstraction. Kreps, J. (2013) [Электронный ресурс]. Режим доступа: <https://www.scribd.com/document/276130136/The-Log-What-Every-Software-Engineer-Should-Know-About-Real-time-Data-s-Unifying-Abstraction>