

Advances in IT and Digital Security

ISSN 3143-4266

DAC Insight Publishers

<https://journals.dacinsightpublishers.com/AIDY>



A MODULAR EVENT-DRIVEN MESSAGING FRAMEWORK FOR WEB AND MOBILE APPLICATIONS: UNIFYING REST AND WEBSOCKET ACCESS OVER DECLARED CHANNEL SEMANTICS

Harouna Wendpanga Yann Christian Sankara^{1*}, Serif Oyindamola Oyesiji², Emmanuel Eniola Ajala³, Toussida Fatah Tanguy Minoungou⁴.

¹ Independent Researcher, Dallas, TX, USA

² Harrisburg University of Science and Technology, Harrisburg, PA

³ Independent Researcher, Atlanta, Georgia

⁴ Independent Researcher, Dallas, TX, USA

Corresponding Author*: christiansankara@gmail.com

ABSTRACT

Modern web and mobile applications are event-driven at the experience level, feeds refresh, presence changes, collaborative edits land, orders progress, while remaining request-driven at the architecture level, built on REST endpoints with real-time behavior retrofitted through ad hoc WebSocket handlers, duplicated delivery logic, and reconnection code rewritten per feature. The costs are structural: state divergence between what requests return and what events pushed, no uniform semantics for ordering, resumption, or backpressure, and mobile clients whose offline behavior is an afterthought. This paper proposes a modular event-driven messaging framework for web and mobile applications that treats REST and WebSocket as two disciplined faces of one event-oriented core. The framework defines a channel abstraction, named, typed event streams with declared delivery semantics, per-channel ordering keys, and retention windows, produced by application services into an event core and consumed uniformly over three access modes: request-response snapshots over REST, live subscription over WebSocket with sequence numbering and snapshot-plus-delta resumption, and hybrid read-then-subscribe that guarantees gapless transition from snapshot to stream. Modularity is achieved through a layered contract: transport adapters, session and subscription management, channel routing with per-channel policy, and application bindings are separately replaceable, with server-sent events and polling adapters degrading gracefully where full-duplex transport is unavailable. Client libraries specify offline queueing, idempotent command submission, and convergent local caches for mobile conditions. The framework's semantics, delivery guarantees per channel class, reconnection contracts, and backpressure policies, are stated precisely, and an evaluation plan measures latency, resumption correctness under churn, server fan-out capacity, and developer-facing modularity through feature-implementation studies. Applications, limitations, and migration pathways from REST-only systems are analyzed.

Keywords: Event-Driven Architecture, Messaging Framework, WebSocket, REST API, Publish-Subscribe, Real-Time Communication, Channel Semantics, Event Streaming, Backpressure, Session Resumption, Offline-First Computing, Mobile Applications.

Received 1 April 2026, Accepted, Accepted 10 April 2026, Available online 30 April 2026

1. INTRODUCTION

The interaction model of contemporary applications has outgrown the architecture most of them run on. The web's architectural style is request-response by design, and its principled formulation as representational state transfer remains the backbone of application interfaces (Fielding & Taylor, 2002), while the experiences built on top, messaging, collaboration, live dashboards, delivery tracking, are streams of events that clients must learn about as they happen. The mismatch was measured early: polling and long-polling over request channels impose latency floors and overhead ceilings that degrade exactly the interactive behavior they attempt to provide (Gutwin *et al.*, 2011), and the WebSocket protocol standardized the escape, a persistent, full-duplex, message-framed channel negotiated from the web's own infrastructure (Fette & Melnikov, 2011), with measured advantages for streaming updates over the polling patterns it replaced (Pimentel & Nickerson, 2012; Skvorc *et al.*, 2014) and a settled place among the web's real-time options (Loreto & Romano, 2012; Grigorik, 2013). The transport substrate continues to evolve beneath both faces, multiplexed streams and user-space transports with connection migration (Belshe *et al.*, 2015; Langley *et al.*, 2017; Iyengar & Thomson, 2021), and enterprise edge networks evolve alongside it as software-defined overlays whose resilience properties condition what applications can assume about connectivity (Ogunwola, 2026), without changing the application-level problem.

That problem is semantic, not transportation. Publish-subscribe is the right abstraction for event distribution, decoupling producers from consumers in space, time, and synchronization, with a design space of topic addressing, filtering, and quality dimensions mapped by the foundational survey (Eugster *et al.*, 2003) and elaborated by the event-based middleware tradition in content-based routing and distributed notification services (Carzaniga *et al.*, 2001; Pietzuch & Bacon, 2002; Mühl *et al.*, 2006). The enterprise integration literature catalogued the recurring message-flow patterns applications actually need, channels, routers, translators, dead letters (Hohpe & Woolf, 2003), atop the observer foundations of the design-patterns canon (Gamma *et al.*, 1994), and the practitioner strand documents how those patterns are realized in contemporary enterprise service buses and multi-cloud integration layers (Badmus, Jooda, Ozowara, & Anunagba, 2022b) and in the cloud messaging architectures that carry them (Ahmed & Odejebi, 2018a). And the data-infrastructure strand supplied the durable core: the partitioned, replicated log as the system of record for event flow, with replay, per-key ordering, and independent consumption (Kreps *et al.*, 2011; Kleppmann & Kreps, 2015; Kleppmann, 2017), consumed under event-time semantics where processing requires it (Akidau *et al.*, 2015), within microservice architectures whose services communicate increasingly by events rather than calls (Newman, 2015; Richardson, 2018) and whose cloud-native deployments now extend to edge execution contexts (Idika *et al.*, 2021).

Yet at the application edge, where web and mobile clients live, these strands rarely compose into a discipline. Typical codebases expose REST for state and bolt WebSocket handlers beside it per feature, so ordering, resumption, duplicate handling, and backpressure are re-decided, or undecided, feature by feature; the governance literature on application programming interfaces makes the same point from the risk side, that ungoverned interface proliferation produces inconsistent contracts and unpriced operational exposure (Aliliele, Mbonu, & Iwuanyanwu, 2023b). Snapshot reads and event streams race, producing the missed-update and double-apply bugs every real-time feature rediscovers, and the live dashboard and analytics surfaces that consume these streams inherit the inconsistency directly (Basnet, Oghenemaiga, & Anene, 2023; Orise & Niniola, 2022). Mobile clients, for whom disconnection is the common case and radio energy the budget (Carroll & Heiser, 2010), inherit reconnection logic designed for browsers on

wired networks, with offline mutation handled by hope, even though the wireless systems literature has characterized the energy, reliability, and latency trade-offs such clients actually face with some precision (Asiedu & Quainoo, 2024; Quainoo & Ogundapo, 2026a). The distributed-systems literature has principled answers, logical ordering (Lamport, 1978), availability-consistency trade-offs stated plainly (Gilbert & Lynch, 2002; Brewer, 2012), and convergent replication for the offline-first client state problem (Terry *et al.*, 1995; DeCandia *et al.*, 2007; Shapiro *et al.*, 2011), but they arrive in applications piecemeal if at all. Meanwhile the operational literature shows what is at stake at scale, where concurrency behavior and service assurance for large user populations are measured properties rather than assumptions (Odejebi & Ahmed, 2018b; Oteri & Edivri, 2026).

This paper proposes the composition as a framework: a modular event-driven messaging layer in which REST and WebSocket are two access modes over one channel abstraction with declared semantics, and in which the recurring decisions, ordering, resumption, delivery class, backpressure, offline behavior, are made once, per channel, in configuration rather than per feature in code. Section 2 reviews background; Section 3 states requirements and principles; Sections 4 through 7 specify the framework, the channel model, the access modes, and the client libraries; Section 8 details modularity and extension; Section 9 sets out the evaluation plan; Sections 10 and 11 treat applications and limitations; Section 12 concludes.

2. BACKGROUND AND RELATED WORK

Three literatures ground the framework. The event distribution literature defines the abstraction and its options. Publish-subscribe decoupling and its quality dimensions frame what a channel can promise (Eugster *et al.*, 2003); content-based routing systems demonstrated expressive subscription at wide-area scale (Carzaniga *et al.*, 2001); typed event middleware integrated events with application type systems (Pietzuch & Bacon, 2002); and the consolidated treatments organize routing, filtering, and composite event detection as an engineering repertoire (Mühl *et al.*, 2006), with complex event processing extending consumption from delivery to pattern detection where applications need it (Cugola & Margara, 2012). The integration-patterns catalogue names the message-flow structures, point-to-point and publish-subscribe channels, routers, dead-letter handling, that recur regardless of substrate (Hohpe & Woolf, 2003), and the observer pattern anchors the in-process ancestry (Gamma *et al.*, 1994). Applied studies of integration middleware and of extract-transform-load design in enterprise platforms show the same structures reappearing under commercial names, together with the coupling and schema-evolution costs that follow when they are adopted without contracts (Badmus, Dosunmu, & Ozowara, 2019a; Badmus, Jooda, Ozowara, & Anunagba, 2022b).

The web transport literature defines the edge. The REST formulation fixes the request-response face and its cache-friendly, stateless virtues (Fielding & Taylor, 2002); WebSocket fixes the full-duplex face (Fette & Melnikov, 2011), with empirical comparisons quantifying its overhead and latency advantages for push workloads (Pimentel & Nickerson, 2012; Skvorc *et al.*, 2014; Gutwin *et al.*, 2011) and practitioner syntheses situating server-sent events and streaming responses as the unidirectional alternatives (Grigorik, 2013; Loreto & Romano, 2012). Transport evolution, multiplexing in the HTTP/2 framing layer and the QUIC transport's user-space streams with connection migration, changes the floor beneath the adapters without changing channel semantics, and the framework confines it there by design (Belshe *et al.*, 2015; Langley *et al.*, 2017; Iyengar & Thomson, 2021). Beneath the protocol layer, the physical and link layers set the achievable envelope, with throughput degradation under varying channel conditions, impedance and radio-

link characteristics, and system-on-chip transceiver behavior all bearing directly on what latency budgets a mobile client can honor (Quainoo, Ogundapo, & Asiedu, 2024; Quainoo, Ogundapo, & Asiedu, 2025a; Quainoo & Ogundapo, 2026b). Persistent connections also enlarge the abuse surface, as the amplification and denial-of-service literature makes clear for any long-lived or reflectable protocol, which is why the framework treats connection admission and backpressure as security-relevant rather than merely performance-relevant (Jimoh & Ahmed, 2024).

The distributed data literature defines the core and the client. The log-centric position, durable, partitioned, replayable event streams as the integration backbone, supplies the framework's server-side substrate and its retention and per-key ordering vocabulary (Kreps *et al.*, 2011; Kleppmann & Kreps, 2015; Kleppmann, 2017), consumed within microservice architectures whose inter-service communication the pattern literature organizes (Newman, 2015; Richardson, 2018), with event-time processing semantics available where derived views require them (Akidau *et al.*, 2015). The cloud-resources strand supplies the elasticity and placement models that determine whether a core binding can absorb fan-out at all, from constraint-satisfaction formulations of allocation to predictive scaling and query-optimization complexity (Ahmed, Odejebi, & Oshoba, 2019; Ahmed, Odejebi, & Oshoba, 2020; Odejebi, Hammed, & Ahmed, 2019). For clients, the offline-first problem is the classical weakly-connected replication problem: divergence under partition with principled reconciliation (Terry *et al.*, 1995), availability-first stores that made eventual consistency an engineering practice (DeCandia *et al.*, 2007), convergent replicated data types whose merges guarantee convergence without coordination (Shapiro *et al.*, 2011), and the impossibility results that make the trade-offs explicit rather than accidental (Gilbert & Lynch, 2002; Brewer, 2012), all under the logical-time foundations that make ordering claims meaningful (Lamport, 1978). Mobile energy analysis completes the client picture, radio dominance in the power budget arguing for batched, state-aware transmission (Carroll & Heiser, 2010), a finding the intermittent-computing literature pushes further by designing delivery protocols for devices whose power availability is itself unreliable and by modeling coexistence and duty-cycling costs at the system level (Asiedu & Quainoo, 2023a; Asiedu & Quainoo, 2023b; Asiedu & Quainoo, 2025; Asiedu, Quainoo, & Asiedu, 2026; Quainoo & Ogundapo, 2026a). The edge-intelligence strand is relevant to the same client budget from the compute side, where on-device inference and model compression contend for the energy and memory a messaging client must share (Nwakamma, Ojukwu, & Oyesiji, 2025; Oyesiji, Nwakamma, & Ojukwu, 2025), and portable cloud-to-edge runtimes offer a substitution story for adapter and handler code that matches the framework's modularity ambitions (Ojukwu, Oyesiji, & Nwakamma, 2025). The framework's contribution against this background is not a new mechanism in any one literature but a specified, modular composition at the application edge, where the literatures currently meet ad hoc.

3. REQUIREMENTS AND DESIGN PRINCIPLES

Six requirements govern the framework. Semantic unification: snapshot reads and event streams must be views of the same channel state, with a guaranteed gapless read-then-subscribe transition, eliminating the race that ad hoc composites exhibit. Declared semantics: every channel declares its delivery class, ordering key, retention window, and backpressure policy in configuration, so consumers program against contracts, not implementations (Eugster *et al.*, 2003; Hohpe & Woolf, 2003), and so that the declarations become reviewable artifacts in the same sense that interface governance and infrastructure-as-code governance make platform configuration auditable (Aliliele, Mbonu, & Iwuanyanwu, 2023b; Mbonu, Iwuanyanwu, Aliliele, & Uzoka, 2020b). Where channels carry regulated content, the declaration is also the natural attachment point for sensitivity

classification and traceability metadata rather than a separate parallel registry (Aliliele, Mbonu, & Iwuanyanwu, 2024a).

Resumability: sessions reconnect with bounded replay from acknowledged sequence positions and explicit discontinuity signaling beyond the window, on every transport that supports it. Graceful degradation: where full-duplex transport is unavailable, adapters degrade to server-sent events or managed polling with the same channel contracts and honestly weaker latency, never with silently different semantics (Grigorik, 2013), the same posture the intermittent-networking literature adopts when it preserves reliable-broadcast guarantees across power interruptions rather than weakening them silently (Asiedu & Quainoo, 2025). Mobile citizenship: offline queueing of commands with idempotent submission, convergent local caches for subscribed channels, and radio-aware batching are specified client behaviors, not integrator homework (Shapiro *et al.*, 2011; Carroll & Heiser, 2010; Asiedu & Quainoo, 2024; Quainoo & Ogundapo, 2026a). And modularity as replaceability: transports, session management, routing, the durable core binding, and application bindings are layered behind interfaces such that each can be substituted without touching the others, the property Section 8 makes concrete and the evaluation plan tests, with substitution treated as a reviewed change under version control and code-review governance rather than an undocumented rewrite (Badmus, Jooda, Ozowara, & Anunagba, 2022a; Ojukwu, Oyesiji, & Nwakamma, 2025).

4. FRAMEWORK OVERVIEW

The framework is organized as five layers around a channel registry. At the center, the event core is the durable substrate: an embedded store for modest deployments or a partitioned log for scale, bound through a core interface that requires only appended, per-key-ordered, retention-windowed streams (Kreps *et al.*, 2011; Kleppmann, 2017). Which binding is viable at a given scale is a resource question the cloud-allocation literature addresses directly, in placement and constraint-satisfaction terms and in predictive scaling terms, and the framework's core interface is deliberately narrow so that those decisions remain deployment decisions (Ahmed & Odejebi, 2018a; Ahmed, Odejebi, & Oshoba, 2019; Ahmed, Odejebi, & Oshoba, 2020; Odejebi, Hammed, & Ahmed, 2019).

The channel layer projects the core into named channels with declared semantics from the registry. The routing layer matches subscriptions to channels, applies per-channel policy, filtering, fan-out, conflation, and maintains per-subscriber cursors. The session layer owns client identity, authentication context, subscription sets, acknowledgment state, and resumption, and it is here that the identity and access management literature for distributed and federated systems supplies the propagation model the framework needs, since a long-lived subscription must carry an authorization context that can be evaluated continuously rather than once at connect time (Oshoba, Hammed, & Odejebi, 2019; Mbonu, Aliliele, Iwuanyanwu, & Uzoka, 2020; Ogbole, Okoruwa, Fadayomi, *et al.*, 2021). The adapter layer terminates transports, WebSocket foremost, REST for snapshots and commands, server-sent events and polling as degradation paths, translating each into the session layer's uniform operations, with the enterprise network substrate beneath it treated as a variable rather than a constant (Ogunwola, 2026). Application services interact only at the top and bottom edges: they produce events into channels and implement command handlers; everything between is framework. The design's center of gravity is the registry: a channel definition is the single place where a feature's delivery class, ordering, retention, and backpressure

are decided, which is what converts recurring per-feature decisions into reviewed configuration (Hohpe & Woolf, 2003; Aliliele, Mbonu, & Iwuanyanwu, 2024a).

5. THE CHANNEL MODEL

A channel is a named, typed event stream with five declared properties. Type: the event schema, versioned, with compatibility rules for evolution, the same discipline the data-integration literature applies to pipeline contracts when upstream schemas change beneath downstream consumers (Badmus, Dosunmu, & Ozowara, 2019a). Ordering key: the per-key total order the channel guarantees, user, document, order, conversation, mapped onto the core's partitioning so that ordering claims are structural rather than aspirational (Kreps *et al.*, 2011; Lamport, 1978). Delivery class: state channels carry keyed current-state events where the latest per key supersedes predecessors, and are conflatable end to end; event channels carry individually meaningful occurrences, delivered at-least-once with consumer idempotency keys, never conflated; command-result channels carry per-requester outcomes with exactly-once effect achieved through idempotent submission rather than transport magic (Kleppmann, 2017). Retention: the replay window backing resumption, declared per channel from seconds to the full log, and doubling as the evidentiary window where regulated flows require an immutable record of what was delivered and when (Oshoba, Hammed, & Odejebi, 2020). Backpressure policy: per delivery class defaults, conflate state, queue events with bounded buffers and explicit overflow signaling, overridable per channel where consumers justify it.

The snapshot relation is part of the model, not an add-on. Every state channel defines its snapshot as the keyed reduction of its stream, served with the sequence position it reflects, so the read-then-subscribe transition is specified exactly: snapshot at position p , then deltas from p , gapless by construction, with the same contract underpinning REST reads and WebSocket resumption alike. Derived channels, projections, joins, and windowed aggregates over base channels, are declared in the registry and computed within the framework under event-time semantics where lateness matters (Akida *et al.*, 2015), giving dashboards and composite views the same contracts as base features (Cugola & Margara, 2012). This matters most for the operational surfaces that consume aggregates continuously, where real-time monitoring, visualization, and key-performance-indicator tracking currently reimplement their own refresh and reconciliation logic per view (Basnet, Oghenemaiga, & Anene, 2023; Ogbole, Okoruwa, Babatope, & Oyewole, 2021; Tonoyan, Dada, & Ayivi-Donkor, 2024b).

6. ACCESS MODES AND SESSION SEMANTICS

The REST face serves three verbs against the model: snapshot reads of state channels at explicit positions, ranged reads of event channels within retention, and command submission with idempotency keys and asynchronous results on command-result channels, preserving the stateless, cacheable virtues where they apply (Fielding & Taylor, 2002). The idempotency discipline is the same one that payment and settlement systems adopt when a retried instruction must not produce a second effect, and the framework borrows the requirement rather than inventing it (Akomolafe, Olaogun, Adesuyi, Ndukwe, & Sakyi, 2022; Akomolafe, Olaogun, Adesuyi, Ndukwe, & Sakyi, 2024a).

The WebSocket face carries the live protocol: typed frames for subscribe, unsubscribe, event delivery with channel and sequence identifiers, acknowledgment, heartbeat, and discontinuity, over the standard transport (Fette & Melnikov, 2011). Sessions resume by presenting

acknowledged positions per subscription; the session layer replays within retention or signals discontinuity, upon which clients re-snapshot through the same specified transition, so reconnection is a contract, not a strategy per feature (Pimentel & Nickerson, 2012). The hybrid mode composes the faces: subscription requests may ask for snapshot-then-stream in one operation, and the framework guarantees the gapless splice.

Degradation adapters honor the same session semantics at reduced capability. Server-sent events carries downstream delivery with REST-submitted commands upstream; managed polling exposes the cursor model over plain requests with adaptive intervals; both are declared per deployment and negotiated per client, so constrained environments lose latency, never correctness (Grigorik, 2013; Loreto & Romano, 2012; Asiedu & Quainoo, 2025). Backpressure operates in the session layer uniformly: per-subscriber buffers enforce channel policy, conflating state channels to freshest-per-key and bounding event channels with overflow signals that clients must acknowledge, making slow-consumer behavior observable and policy-driven rather than an accident of socket buffers. Because unbounded fan-out is also an availability risk, admission control and per-session quotas are first-class session-layer concerns rather than deployment afterthoughts (Jimoh & Ahmed, 2024; Odejobi & Ahmed, 2018b; Edivri & Oteri, 2022). Transport evolution is absorbed at the adapter boundary alone, with multiplexed and migration-capable transports slotting beneath unchanged frames, and with link-layer variability treated as a property the adapter reports rather than one the application discovers through timeouts (Iyengar & Thomson, 2021; Langley *et al.*, 2017; Ogunwola, 2026; Quainoo, Ogundapo, & Asiedu, 2025a).

7. CLIENT LIBRARIES AND MOBILE DISCIPLINE

The client libraries are part of the framework's specification because edge semantics fail at the edge. Each library maintains, per subscribed channel, a local cache consistent with the channel's class: state channels materialize the keyed reduction, applying deltas idempotently by key and sequence; event channels expose ordered, deduplicated streams to application callbacks. Commands submit through an outbox with idempotency keys, queued across disconnection and replayed on reconnect, so user actions survive tunnels and airplane mode with exactly-once effect at the handler (Kleppmann, 2017), which is the client-side counterpart of the continuity models payment infrastructures adopt for instructions that must survive interruption of the channel that carried them (Adesuyi, Akomolafe, Olaogun, Ndukwe, & Sakyi, 2025b).

For collaborative and offline-mutable state, the libraries support convergent replicated types as the cache representation for channels declared mergeable, importing the convergence guarantee rather than improvising merge logic per application (Shapiro *et al.*, 2011; Terry *et al.*, 1995). Radio awareness is built in: acknowledgments and outbox flushes batch opportunistically, heartbeat cadence adapts to platform power state, and subscription hydration prioritizes visible surfaces, applying the mobile energy findings as library defaults (Carroll & Heiser, 2010) and drawing on the wireless-systems characterization of how radio coexistence, duty cycling, and harvest uncertainty shape the achievable schedule (Asiedu & Quainoo, 2024; Asiedu, Quainoo, & Asiedu, 2026; Quainoo & Ogundapo, 2026a). On devices that also run local inference, the messaging client's energy and memory budget is contested rather than sole, which argues for the same compression and scheduling discipline the on-device model literature applies to its own workloads (Nwakamma, Ojukwu, & Oyesiji, 2025; Oyesiji, Nwakamma, & Ojukwu, 2025).

Local caches are also a data-protection surface, since a subscribed channel materialized on a handset is personal data at rest under most regimes, so the libraries expose per-channel retention

and eviction controls and default to the least persistent option consistent with the declared semantics (Annan, 2022; Badmus, Jooda, & Anunagba, 2024). The availability-consistency stance is stated to developers honestly: caches are available and eventually consistent under partition, with staleness bounded by observable positions, the trade the impossibility results make unavoidable and the framework makes explicit (Gilbert & Lynch, 2002; Brewer, 2012).

8. MODULARITY AND EXTENSION

Modularity is specified as five substitution points with contracts. Core bindings: the event core interface admits embedded stores, partitioned logs, or cloud streaming services, chosen per deployment scale without touching channel definitions, with the placement and elasticity characteristics of the chosen binding governing cost and headroom rather than semantics (Kreps *et al.*, 2011; Ahmed & Odejebi, 2018b; Ahmed, Odejebi, & Oshoba, 2019).

Transport adapters: new transports implement the frame contract; the degradation set is itself an example, and platform push services can be bound as wake-up adapters that trigger session resumption, a path that also brings notification-quality obligations into scope where the delivered content is safety-relevant or must reach recipients across languages and accessibility needs (Odejide, 2023; Odejide, 2024). Routing extensions: per-channel filters and derived-channel operators register as typed components, giving content-based subscription where features need it within the expressiveness the routing literature maps (Carzaniga *et al.*, 2001; Mühl *et al.*, 2006), and reusing the routing and transformation vocabulary that enterprise integration platforms already expose (Badmus, Jooda, Ozowara, & Anunagba, 2022b).

Policy modules: authentication context propagation, per-channel authorization, and quota enforcement attach at the session layer as middleware, drawing on identity-centric and zero-trust formulations in which authorization is continuous and contextual rather than established once at the perimeter, and on the access-control literature for heterogeneous and bring-your-own-device client populations (Adebayo, Anunagba, & Ozowara, 2025; Ladapo, Jooda, Dosunmu, & Abolaji, 2023b; Mbonu, Iwuanyanwu, Aliliele, & Uzoka, 2022b; Ogbole, Okoruwa, Fadayomi, *et al.*, 2021; Oshoba, Hammed, & Odejebi, 2019).

Application bindings: producer and handler interfaces are thin and framework-agnostic on the service side, so the framework composes with prevailing service architectures rather than prescribing one (Newman, 2015; Richardson, 2018; Idika *et al.*, 2021), with portable component runtimes offering one concrete substitution mechanism for handlers that must run in both cloud and edge contexts (Ojukwu, Oyesiji, & Nwakamma, 2025). Each substitution point is exercised in the reference implementation by at least two bindings, which is the concrete meaning of the modularity claim and the object of the evaluation plan's replaceability study. Because substitution is a deployment event, the framework specifies its bindings to be promotable through ordinary continuous integration and delivery pipelines, with tooling comparisons in adjacent platform-engineering literature indicating how much of the substitution cost is tooling rather than architecture (Badmus, Dosunmu, & Ozowara, 2018; Badmus, Dosunmu, Ozowara, & Anunagba, 2020).

9. EVALUATION PLAN

Evaluation proceeds in three phases. Phase one verifies semantics. A protocol conformance harness drives reference client and server through the specified state machines, subscription,

acknowledgment, resumption, discontinuity, and read-then-subscribe splice, under scripted disconnection, reordering, and duplication at the transport boundary, asserting the channel contracts hold: gapless transitions, per-key order on state channels, no conflation on event channels, idempotent command effect. The harness is built as an automated regression suite in the sense the software-testing literature intends, so that contract violations are caught continuously rather than at integration, and instrument-driven test automation from wireless hardware engineering supplies a useful model for scripting the physical-condition side of the matrix (Borah, Aliliele, Rakshit, & Vajjhala, 2022; Mbonu, Aliliele, Iwuanyanwu, & Uzoka, 2021b; Quainoo, Ogundapo, & Asiedu, 2025b). Fault injection extends the scripts to core failover and adapter degradation, verifying that weaker transports weaken latency, not guarantees, with continuity criteria drawn from the business-continuity literature on infrastructure change under load (Adesuyi, Akomolafe, Olaogun, Ndukwe, & Sakyi, 2025b; Ogunwola & Alozie, 2026).

Phase two measures performance envelopes. Server-side, fan-out capacity is measured as concurrent subscribers per node across channel mixes and payload sizes, with latency distributions, including tails, from publish to client callback, situating the WebSocket face against the polling and server-sent alternatives under identical channel loads, extending the transport comparison literature to the framework's full session semantics (Pimentel & Nickerson, 2012; Skvorc *et al.*, 2014; Gutwin *et al.*, 2011). Concurrency behavior in multi-tenant deployments and service-assurance targets for large user populations provide the reference points against which envelopes are judged adequate or not, and capacity forecasting supplies the method for projecting them forward (Edivri & Oteri, 2022; Odejobi & Ahmed, 2018b; Oteri & Edivri, 2026; Oyeleke, Eze, & Asiedu, 2026). Resumption cost is measured under churn: reconnect storms with varying retention windows, quantifying replay load and time-to-consistency. Client-side, mobile builds are measured for energy per session hour across heartbeat and batching policies on representative devices (Carroll & Heiser, 2010), under channel conditions varied deliberately rather than incidentally so that throughput degradation and coexistence effects are attributed rather than averaged away (Asiedu, Quainoo, & Asiedu, 2025; Quainoo & Ogundapo, 2026a; Quainoo, Ogundapo, & Asiedu, 2025a).

Phase three measures the developer-facing claims. Feature-implementation studies task developers with representative real-time features, presence, collaborative list, live order tracking, against the framework and against a conventional REST-plus-ad-hoc-WebSocket baseline, comparing implementation size, defect classes found by the conformance harness, and time to correct resumption behavior, the modularity and discipline claims rendered as observations rather than assertions. Instrumentation follows the key-performance-indicator frameworks used for service delivery and adoption measurement, so that the developer-facing outcome is reported on the same footing as the runtime envelopes (Edivri & Oteri, 2024; Oteri & Edivri, 2024; Tonoyan, Dada, & Ayivi-Donkor, 2024b). Replaceability is evaluated by substitution exercises at each extension point, measuring the blast radius of each swap in changed lines and affected modules, reviewed under the version-control and code-review governance the platform-engineering literature recommends for exactly this kind of change (Badmus, Jooda, Ozowara, & Anunagba, 2022a). Acceptance for the framework is conjunctive: contracts verified under fault, envelopes adequate for the target application classes, and edge discipline demonstrably cheaper to use than to improvise.

10. ANTICIPATED APPLICATIONS

The framework targets the application classes whose experiences are event-shaped. Collaboration and messaging features consume state and event channels directly, with mergeable caches carrying the offline-editable surfaces, over the cloud messaging substrates such features are already deployed on (Shapiro *et al.*, 2011; Ahmed & Odejebi, 2018a). Commerce and logistics applications map order and delivery lifecycles onto state channels with command-result flows for user actions, gaining gapless tracking views for free, which is directly useful for the routing, fulfilment, and last-mile surfaces where position and status updates are currently polled (Mayo, Ogbale, Okoruwa, & Babatope, 2021; Tonoyan, Dada, & Ayivi-Donkor, 2025; Anene & Clement, 2022). Warehouse and fulfilment automation is a demanding instance of the same shape, since fleet coordination and slotting decisions consume high-rate position and task-state streams whose ordering and freshness requirements are exactly what per-channel declaration is meant to express (Akanbi & Sunday, 2024; Akanbi & Sunday, 2025; Akanbi, Ganiu, & Sunday, 2025).

Dashboards and operations consoles consume derived channels, aggregations and joins declared in the registry rather than hand-built per view (Cugola & Margara, 2012; Akidau *et al.*, 2015), which is the pattern real-time analytics, monitoring, and visualization systems arrive at independently and at some cost (Basnet, Oghenemaiga, & Anene, 2023; Oghenemaiga, Basnet, & Anene, 2024; Ogbale, Okoruwa, Babatope, & Oyewole, 2021; Orise & Niniola, 2022; Tonoyan, Dada, & Ayivi-Donkor, 2024b). Industrial and infrastructure telemetry fits the state-channel model closely, where sensor networks, digital twins, and condition-monitoring surfaces publish keyed current state that operators consume live and analysts replay later (Jimoh, Abolle-Okoyeagu, Ahmed, & Lawal, 2023; Kumuyi, Uzoka, Akeju, & Ozowara, 2024a; Shittu, Shittu, Adeleke, & Adedokun, 2023; Shittu, Shittu, Akintayo, & Opara, 2025; Sunday & Omoegun, 2022). Public-safety and alerting applications are the case where delivery-class declaration carries the most weight, since an alert is an event channel that must never be conflated and whose reach obligations extend to multilingual and accessible delivery (Odejide, 2023; Odejide, 2024).

Social and content feeds use event channels with bounded retention and conflated presence beside them. A wide class of ordinary line-of-business applications benefits without any real-time ambition at all, since bidding, approval, tracking, and attendance systems all contain state whose staleness is currently managed by refresh buttons (Aniji, Chizoba, & Ahmed, 2026a; Aniji, Chizoba, Ahmed, & Chukwu, 2026b; Ayari, Chizoba, & Ahmed, 2026; Ussher-Eke, Okoh, & Ijiga, 2025; Yusuf, Akintunde, Shittu, & Quadri, 2023). Beyond product features, the framework serves as migration infrastructure: REST-only systems adopt it by declaring their first state channels over existing stores, serving snapshots immediately and adding live subscription without re-architecting services, the incremental path that determines whether edge disciplines get adopted at all, and the path the cloud-migration and service-management literature identifies as the one organizations can actually execute (Newman, 2015; Kleppmann, 2017; Ladapo, Dosunmu, Jooda, & Abolaji, 2023b; Ladapo, Dosunmu, Jooda, & Abolaji, 2025; Ladapo, Jooda, Dosunmu, & Abolaji, 2022).

11. LIMITATIONS

The framework's boundaries are explicit. It specifies application-edge messaging, not inter-service architecture: services keep their own integration substrate, and the framework binds to it rather than replacing it (Richardson, 2018). Its guarantees are per-channel and per-key: cross-channel transactions and global ordering are out of scope by design, and features needing them must model

them explicitly, the honest consequence of the availability-first stance (Gilbert & Lynch, 2002; Brewer, 2012). Mergeable caches cover the state shapes convergent types express well, sets, maps, counters, sequences with known trade-offs, and do not absolve applications of semantic merge decisions beyond them (Shapiro *et al.*, 2011). Degradation adapters preserve contracts at real latency cost, and applications whose value collapses without full-duplex transport should treat the degraded modes as availability signaling, not equivalence.

The framework standardizes delivery semantics, not compliance: channels carrying regulated or cross-border data still require classification, retention justification, and jurisdictional analysis that the registry can record but cannot decide (Aliliele, Mbonu, & Iwuanyanwu, 2024a; Annan, 2022; Mbonu, Aliliele, Iwuanyanwu, & Oluoha, 2018). Nor does it establish a security posture on its own; long-lived authenticated sessions enlarge the attack surface, and the framework inherits rather than resolves the threat-detection and secure-engineering obligations that come with that (Odejobi, Okonkwo, Ahiaekwe Patrick, Okeke, & Mayo, 2025). The evaluation's developer studies carry the usual external-validity bounds of such designs, tasks and participants sample a space, and the performance envelopes are per reference implementation, with the core-binding modularity meaning envelopes must be re-established per binding rather than inherited (Oteri & Edivri, 2026). Finally, the framework standardizes semantics, not judgment: channel declarations concentrate the decisions, and mis-declared channels, an event stream marked conflatable, fail as efficiently as they run, which makes the human and organizational factors around who declares channels and under what review a live risk rather than an implementation detail (Onche, Adegbite, & Ogbonna, 2026).

12. CONCLUSION

This paper has proposed a modular event-driven messaging framework for web and mobile applications in which REST and WebSocket are disciplined access modes over one channel abstraction. The framework contributes a channel model with declared types, ordering keys, delivery classes, retention, and backpressure policy; a specified snapshot relation that makes read-then-subscribe gapless by construction; session semantics with sequence-numbered delivery, bounded replay resumption, and honest discontinuity; client libraries that make offline queueing, idempotent commands, convergent caches, and radio-aware batching default behavior; and a five-point modularity contract exercised by construction in the reference design. Its evaluation plan verifies the contracts under fault, measures the envelopes against the alternatives the transport literature quantified, and tests the developer-facing claims empirically.

The position beneath the design is that real-time correctness at the application edge is a semantics problem that each feature team currently re-solves, and that the remedy is the same one software architecture has always applied to recurring decisions: name the abstraction, declare the contract, and make the disciplined path the easy one. Executed, the framework turns the recurring bugs of ad hoc real-time, races between reads and streams, resumption gaps, silent conflation, lost offline actions, into contract violations a conformance harness catches, which is what it means for event-driven application messaging to become engineering rather than folklore.

REFERENCES

1. Adebayo, A., Anunagba, C. O., & Ozowara, D. E. (2025). A review of zero trust security models and cost effectiveness in healthcare infrastructure. *International Journal of Advanced Multidisciplinary Research and Studies*, 5(6), 2269-2283. <https://doi.org/10.62225/2583049X.2025.5.6.6051>
2. Adesuyi, M. O., Akomolafe, O., Olaogun, B. O., Ndukwe, V. U., & Sakyi, J. K. (2025b). Resilience and continuity model for global payment infrastructure under geopolitical risks. *International Journal of Multidisciplinary Research and Growth Evaluation*, 6(4), 1467-1482. <https://doi.org/10.54660/IJMRGE.2025.6.4.1467-1482>
3. Ahmed, K. S., & Odejebi, O. D. (2018a). Conceptual framework for scalable and secure cloud architectures for enterprise messaging. *IRE Journals*, 2(1), 1-15.
4. Ahmed, K. S., & Odejebi, O. D. (2018b). Resource allocation model for energy-efficient virtual machine placement in data centers. *IRE Journals*, 2(3), 1-10.
5. Ahmed, K. S., Odejebi, O. D., & Oshoba, T. O. (2019). Algorithmic model for constraint satisfaction in cloud network resource allocation. *IRE Journals*, 2(12), 516-532.
6. Ahmed, K. S., Odejebi, O. D., & Oshoba, T. O. (2020). Predictive model for cloud resource scaling using machine learning techniques. *Journal of Frontiers in Multidisciplinary Research*, 1(1), 173-183.
7. Akanbi, O., & Sunday, E. A. (2024). An integrated path-planning and slotting optimization model for AMR-enabled high-density warehousing. *International Journal of Advanced Multidisciplinary Research and Studies*, 4(6), 3226-3243. <https://doi.org/10.62225/2583049X.2024.4.6.6181>
8. Akanbi, O., & Sunday, E. A. (2025). A systematic review of AI-driven autonomous mobile robots (AMR) in scalable micro-fulfillment centers. *International Journal of Advanced Multidisciplinary Research and Studies*, 5(6), 2363-2379. <https://doi.org/10.62225/2583049X.2025.5.6.6182>
9. Akanbi, O., Ganiu, O. S., & Sunday, E. A. (2025). A multi-agent AI framework for swarm intelligence in autonomous mobile robot (AMR) fleet coordination. *International Journal of Scientific Research in Humanities and Social Sciences*, 2(1), 81-109.
10. Akidau, T., Bradshaw, R., Chambers, C., *et al.* (2015). The dataflow model: A practical approach to balancing correctness, latency, and cost in massive-scale, unbounded, out-of-order data processing. *Proceedings of the VLDB Endowment*, 8(12), 1792-1803.
11. Akomolafe, O., Olaogun, B. O., Adesuyi, M. O., Ndukwe, V. U., & Sakyi, J. K. (2022). Smart contract automation model for supplier payment systems and performance benchmarking. *International Journal of Multidisciplinary Research and Growth Evaluation*, 3(6), 827-836. <https://doi.org/10.54660/IJMRGE.2022.3.6.827-836>
12. Akomolafe, O., Olaogun, B. O., Adesuyi, M. O., Ndukwe, V. U., & Sakyi, J. K. (2024a). Scalable blockchain payment gateway architecture model for enterprise-grade adoption. *International Journal of Advanced Multidisciplinary Research and Studies*, 4(1), 1552-1568. <https://doi.org/10.62225/2583049X.2024.4.1.5280>
13. Aliliele, C., Mbonu, I. S., & Iwuanyanwu, U. (2023b). A review of API governance and risk prioritization frameworks in modern financial institutions. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 9(10), 395-433. <https://doi.org/10.32628/CSEIT2361072>
14. Aliliele, C., Mbonu, I. S., & Iwuanyanwu, U. (2024a). A conceptual framework for enterprise data sensitivity classification and regulatory traceability mechanisms. *International Journal of Advanced Multidisciplinary Research and Studies*, 4(6), 3103-3124. <https://doi.org/10.62225/2583049X.2024.4.6.5991>
15. Anene, U. N., & Clement, T. (2022). A resilient logistics framework for humanitarian supply chains: Integrating predictive analytics, IoT, and localized distribution to strengthen emergency response systems. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 8(5), 398-424.

16. Aniji, I. F., Chizoba, C. E., & Ahmed, M. O. (2026a). Digitalizing procurement processes: An online automated auction system: Design, implementation and evaluation. *Middle East Journal of Applied Science and Technology*, 9(1), 11-22. <https://doi.org/10.46431/mejast.2026.9102>
17. Aniji, I. F., Chizoba, C. E., Ahmed, M. O., & Chukwu, J. C. (2026b). Development of an automated employee performance appraisal system to boost organizational productivity. *Asian Journal of Applied Science and Technology*, 10(1), 1-12. <https://doi.org/10.38177/ajast/2026.10101>
18. Annan, A. O. (2022). Data privacy governance models for cross-border digital platforms. *International Journal of Multidisciplinary Research and Growth Evaluation*, 3(6), 949-964.
19. Asiedu, W., & Quainoo, R. (2023a). How far can energy harvesting take us? A systematic review of radio frequency strategies for energy autonomous sensing. *Shodhshauryam, International Scientific Refereed Research Journal*, 6(1), 448-466.
20. Asiedu, W., & Quainoo, R. (2023b). Toward maintenance free wireless infrastructure: Simulating performance and reliability in large scale intermittently powered IoT networks. *Gyanshauryam, International Scientific Refereed Research Journal*, 6(1), 489-510.
21. Asiedu, W., & Quainoo, R. (2024). Rethinking energy, reliability, and latency trade-offs in green communication for next generation IoT. *International Journal of Multidisciplinary Research and Growth Evaluation*, 5(6), 1987-1994.
22. Asiedu, W., & Quainoo, R. (2025). GleanCast: Epoch-aligned reliable broadcast in batteryless intermittent sensor networks. *International Journal of Engineering and Modern Technology*, 11(12), 205-218. <https://doi.org/10.56201/ijemt.vol.11.no12.2025.pg205.218>
23. Asiedu, W., Quainoo, R., & Asiedu, A. (2025). A conceptual framework for characterizing fundamental energy, reliability, and latency trade-offs in green communication paradigms for next-generation IoT. *International Journal of Advanced Multidisciplinary Research and Studies*, 5(6), 2447-2453. <https://doi.org/10.62225/2583049X.2025.5.6.6479>
24. Asiedu, W., Quainoo, R., & Asiedu, A. (2026). Predictive power management for intermittent IoT devices using lightweight machine learning under uncertain energy harvest. *Gulf Journal of Engineering and Technology*, 2(4), 108-118.
25. Ayari, O. N., Chizoba, C. E., & Ahmed, M. O. (2026). A digital solution for student internship: Tracking and enhancing student's experience. *Asian Journal of Basic Science and Research*, 8(1), 23-36. <https://doi.org/10.38177/ajbsr.2026.8103>
26. Badmus, O., Dosunmu, A. A., & Ozowara, D. E. (2018). A systematic review of CI/CD pipeline strategies in Salesforce DevOps: Tools, practices, and deployment outcomes. *Iconic Research and Engineering Journals*, 2(6).
27. Badmus, O., Dosunmu, A. A., & Ozowara, D. E. (2019a). A conceptual model for ETL design and data integration in Salesforce-centric enterprise architectures. *Iconic Research and Engineering Journals*, 3(5).
28. Badmus, O., Dosunmu, A. A., Ozowara, D. E., & Anunagba, C. O. (2020). A comparative framework for Salesforce DevOps tooling: Evaluating Copado, Flosum, and Salesforce DX across enterprise deployment contexts. *International Journal of Multidisciplinary Research and Growth Evaluation*, 1(5), 1021-1031. <https://doi.org/10.54660/IJMRGE.2020.1.5.1021-1031>
29. Badmus, O., Jooda, D., & Anunagba, C. O. (2024). A privacy-by-design framework for role-based security architecture in Salesforce environments handling sensitive personal data. *International Journal of Advanced Multidisciplinary Research and Studies*, 4(6), 3244-3254. <https://doi.org/10.62225/2583049X.2024.4.6.6243>
30. Badmus, O., Jooda, D., Ozowara, D. E., & Anunagba, C. O. (2022a). A framework for Git-based version control and code review governance in Salesforce development teams. *Shodhshauryam, International Scientific Refereed Research Journal*, 5(2), 473-493.
31. Badmus, O., Jooda, D., Ozowara, D. E., & Anunagba, C. O. (2022b). A systematic review of MuleSoft ESB integration patterns in multi-cloud Salesforce architectures. *Gyanshauryam, International Scientific Refereed Research Journal*, 5(3), 462-481.

32. Basnet, A., Oghenemaiga, E., & Anene, U. N. (2023). Real time analytics and monitoring systems for livestream media performance using Google platforms. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 10(1).
33. Belshe, M., Peon, R., & Thomson, M. (2015). Hypertext transfer protocol version 2 (HTTP/2). RFC 7540, Internet Engineering Task Force.
34. Borah, S., Aliliele, K. C., Rakshit, S., & Vajjhala, N. R. (2022). Applications of artificial intelligence in software testing. In P. K. Mallick *et al.* (Eds.), *Cognitive informatics and soft computing* (Lecture Notes in Networks and Systems, Vol. 375, pp. 727-736). Springer. https://doi.org/10.1007/978-981-16-8763-1_60
35. Brewer, E. (2012). CAP twelve years later: How the rules have changed. *Computer*, 45(2), 23-29.
36. Carroll, A., & Heiser, G. (2010). An analysis of power consumption in a smartphone. *Proceedings of the USENIX Annual Technical Conference*, 271-284.
37. Carzaniga, A., Rosenblum, D. S., & Wolf, A. L. (2001). Design and evaluation of a wide-area event notification service. *ACM Transactions on Computer Systems*, 19(3), 332-383.
38. Cugola, G., & Margara, A. (2012). Processing flows of information: From data stream to complex event processing. *ACM Computing Surveys*, 44(3), 15.
39. DeCandia, G., Hastorun, D., Jampani, M., *et al.* (2007). Dynamo: Amazon's highly available key-value store. *Proceedings of the ACM Symposium on Operating Systems Principles*, 205-220.
40. Edivri, J., & Oteri, O. (2022). Predictive capacity planning and resource utilization forecasting models for multi-program and multi-stakeholder IT portfolios. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 8(4), 826-845.
41. Edivri, J., & Oteri, O. (2024). A conceptual KPI-driven decision and optimization framework for IT service delivery, portfolio performance, and adoption. *Gyanshauryam, International Scientific Refereed Research Journal*, 7(1), 167-187. <https://doi.org/10.32628/GISRRJ246643>
42. Eugster, P. T., Felber, P. A., Guerraoui, R., & Kermarrec, A.-M. (2003). The many faces of publish/subscribe. *ACM Computing Surveys*, 35(2), 114-131.
43. Fette, I., & Melnikov, A. (2011). The WebSocket protocol. RFC 6455, Internet Engineering Task Force.
44. Fielding, R. T., & Taylor, R. N. (2002). Principled design of the modern web architecture. *ACM Transactions on Internet Technology*, 2(2), 115-150.
45. Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design patterns: Elements of reusable object-oriented software*. Addison-Wesley.
46. Gilbert, S., & Lynch, N. (2002). Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *ACM SIGACT News*, 33(2), 51-59.
47. Grigorik, I. (2013). *High performance browser networking*. O'Reilly Media.
48. Gutwin, C. A., Lippold, M., & Graham, T. C. N. (2011). Real-time groupware in the browser: Testing the performance of web-based networking. *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, 167-176.
49. Hohpe, G., & Woolf, B. (2003). *Enterprise integration patterns: Designing, building, and deploying messaging solutions*. Addison-Wesley.
50. Idika, C. N., Salami, E. O., Ijiga, O. M., & Enyejo, L. A. (2021). Deep learning driven malware classification for cloud-native microservices in edge computing architectures. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 7(4).
51. Iyengar, J., & Thomson, M. (2021). QUIC: A UDP-based multiplexed and secure transport. RFC 9000, Internet Engineering Task Force.

52. Jimoh, H. O., & Ahmed, M. O. (2024). Analyzing Network Time Protocol (NTP) based amplification DDoS attack and its mitigation techniques. *Journal of Digital Innovations and Contemporary Research in Science, Engineering and Technology*, 12(2), 17-24. <https://doi.org/10.22624/AIMS/DIGITAL/V11N2P2x>
53. Jimoh, H. O., Abolle-Okoyeagu, C. J., Ahmed, M. O., & Lawal, N. O. (2023). Advancing security in IoT-driven critical infrastructure: A focus on smart transportation system. *American Journal of Engineering Research*, 12(12), 33-46.
54. Kleppmann, M. (2017). *Designing data-intensive applications: The big ideas behind reliable, scalable, and maintainable systems*. O'Reilly Media.
55. Kleppmann, M., & Kreps, J. (2015). Kafka, Samza and the Unix philosophy of distributed data. *IEEE Data Engineering Bulletin*, 38(4), 4-14.
56. Kreps, J., Narkhede, N., & Rao, J. (2011). Kafka: A distributed messaging system for log processing. *Proceedings of the NetDB Workshop*, 1-7.
57. Kumuyi, O., Uzoka, E., Akeju, B., & Ozowara, D. E. (2024a). Architecture for machine learning-enabled predictive energy management using IoT sensor networks. *International Journal of Advanced Multidisciplinary Research and Studies*, 4(6), 3138-3149. <https://doi.org/10.62225/2583049X.2024.4.6.6004>
58. Ladapo, O. O., Dosunmu, A. A., Jooda, D., & Abolaji, T. O. (2023b). An IT service management literature review: Challenges, benefits, opportunities, and implementation practices. *International Journal of Advanced Multidisciplinary Research and Studies*, 3(6), 2875-2889. <https://doi.org/10.62225/2583049X.2023.3.6.6205>
59. Ladapo, O. O., Dosunmu, A. A., Jooda, D., & Abolaji, T. O. (2025). Migration of applications and information systems to cloud computing infrastructure: Lessons from a South African retail bank. *International Journal of Multidisciplinary Research and Growth Evaluation*, 6(6), 1361-1375. <https://doi.org/10.54660/IJMRGE.2025.6.6.1361-1375>
60. Ladapo, O. O., Jooda, D., Dosunmu, A. A., & Abolaji, T. O. (2022). Navigating digital transformation: Best practices for cloud migration strategies in the enterprise. *Journal of Frontiers in Multidisciplinary Research*, 3(1), 643-655. <https://doi.org/10.54660/JFMR.2022.3.1.643-655>
61. Ladapo, O. O., Jooda, D., Dosunmu, A. A., & Abolaji, T. O. (2023b). Systematic literature review on security access control policies and techniques based on privacy requirements in a BYOD environment. *International Journal of Advanced Multidisciplinary Research and Studies*, 3(6), 2890-2904. <https://doi.org/10.62225/2583049X.2023.3.6.6206>
62. Lamport, L. (1978). Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7), 558-565.
63. Langley, A., Riddoch, A., Wilk, A., et al. (2017). The QUIC transport protocol: Design and internet-scale deployment. *Proceedings of the ACM SIGCOMM Conference*, 183-196.
64. Loreto, S., & Romano, S. P. (2012). Real-time communications in the web: Issues, achievements, and ongoing research. *IEEE Internet Computing*, 16(5), 68-73.
65. Mayo, W., Ogbale, J. I., Okoruwa, P. O., & Babatope, O. M. (2021). Designing an AI-predictive maintenance model for e-commerce systems using machine learning and cloud analytics. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 7(5), 416-440. <https://doi.org/10.32628/IJSRCSEIT>
66. Mbonu, I. S., Aliliele, C., Iwuanyanwu, U., & Oluoha, O. M. (2018). A conceptual framework for legal and ethical risk modeling in enterprise data protection governance systems. *Iconic Research and Engineering Journals*, 2(2), 207-226. <https://doi.org/10.64388/IREV2I2-1714911>
67. Mbonu, I. S., Aliliele, C., Iwuanyanwu, U., & Uzoka, E. (2020). A review of identity and access management integration strategies in hybrid and multi cloud environments. *International Journal of Multidisciplinary Research and Growth Evaluation*, 1(5), 795-810. <https://doi.org/10.54660/IJMRGE.2020.1.5.795-810>
68. Mbonu, I. S., Aliliele, C., Iwuanyanwu, U., & Uzoka, E. (2021b). Advances in artificial intelligence techniques for secure software testing and automated regression control mechanisms. *International Journal*

- of Scientific Research in Computer Science, Engineering and Information Technology*, 7(5), 468-496. <https://doi.org/10.32628/CSEIT217565>
69. Mbonu, I. S., Iwuanyanwu, U., Aliliele, C., & Uzoka, E. (2020b). Advances in infrastructure as code governance for secure terraform based enterprise cloud deployments. *International Journal of Multidisciplinary Research and Growth Evaluation*, 1(5), 811-828. <https://doi.org/10.54660/IJMRGE.2020.1.5.811-828>
70. Mbonu, I. S., Iwuanyanwu, U., Aliliele, C., & Uzoka, E. (2022b). Advances in cloud identity and access governance optimization in large scale AWS enterprise environments. *Shodhshauryam, International Scientific Refereed Research Journal*, 5(3), 403-438. <https://doi.org/10.32628/SHISRRJ25490>
71. Mühl, G., Fiege, L., & Pietzuch, P. (2006). *Distributed event-based systems*. Springer.
72. Newman, S. (2015). *Building microservices: Designing fine-grained systems*. O'Reilly Media.
73. Nwakamma, S., Ojukwu, J., & Oyesiji, S. O. (2025). On-device multimodal small language models for privacy-preserving edge intelligence through quantization, pruning, distillation, and energy-efficient inference. *World Journal of Innovation and Modern Technology*, 9(12), 271-302. <https://doi.org/10.56201/wjimt.v9.no12.2025.pg271.302>
74. Odejide, G. S. (2023). Artificial intelligence in multilingual and accessible emergency alerting: A scoping review of evidence, applications, and gaps in the United States. *International Journal of Scientific Research in Science, Engineering and Technology*, 10(6), 499-512. <https://doi.org/10.32628/IJSRSET26132152>
75. Odejide, G. S. (2024). A trust-centered conceptual model for AI-driven multilingual emergency communication. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 10(6), 2781-2811. <https://doi.org/10.32628/CSEIT2410794>
76. Odejobi, O. D., & Ahmed, K. S. (2018b). Performance evaluation model for multi-tenant Microsoft 365 deployments under high concurrency. *IRE Journals*, 1(11), 92-107.
77. Odejobi, O. D., Hammed, N. I., & Ahmed, K. S. (2019). Approximation complexity model for cloud-based database optimization problems. *IRE Journals*, 2(9), 1-10.
78. Odejobi, O. D., Okonkwo, C. S., Ahiaekwe Patrick, M. C., Okeke, O. T., & Mayo, W. (2025). AI-augmented secure software engineering: Leveraging deep learning for autonomous threat detection and mitigation. *International Journal of Engineering and Modern Technology*, 11(12), 101-121. <https://doi.org/10.56201/ijemt.vol.11.no12.2025.pg101.121>
79. Ogbole, J. I., Okoruwa, P. O., Babatope, O. M., & Oyewole, T. (2021). Developing an integrated data visualization model for continuous business performance monitoring and optimization. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 7(5), 441-467. <https://doi.org/10.32628/IJSRCSEIT>
80. Ogbole, J. I., Okoruwa, P. O., Fadayomi, O., Abolaji, T. O., Edivri, J., & Akeju, B. (2021). Conceptual model for identity-centric zero trust architecture in enterprise security governance. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 7(5), 393-415. <https://doi.org/10.32628/IJSRCSEIT217562>
81. Oghenemaiga, E., Basnet, A., & Anene, U. N. (2024). Predictive analytics applications for forecasting traffic patterns across owned and operated media platforms. *International Journal of Advanced Multidisciplinary Research and Studies*, 4(6). <https://doi.org/10.62225/2583049X.2024.4.6.5668>
82. Ogunwola, T. A. (2026). Security and resilience considerations for software-defined wide area network deployments in multi-site enterprise environments. *International Advanced Research Journal in Science, Engineering and Technology*, 12(1). <https://doi.org/10.17148/iarjset.2025.12150>
83. Ogunwola, T. A., & Alozie, C. (2026). Enterprise firewall modernization for business continuity: A framework for secure migration and operational stability. *Computer Science & IT Research Journal*, 7(5), 347-361. <https://doi.org/10.51594/csitj.v7i5.2287>
84. Ojukwu, J., Oyesiji, S. O., & Nwakamma, S. (2025). WebAssembly component model and WASI for portable cloud-to-edge software: Language interoperability, security, performance, and container comparisons.

- International Journal of Computer Science and Mathematical Theory*, 11(12), 232-263. <https://doi.org/10.56201/ijcsmt.vol.11.no12.2025.pg232.263>
85. Onche, V. O., Adegbite, M. P., & Ogbonna, C. S. (2026). Human factors in information security: A capability framework for administrative professionals in the digital workplace. *World Journal of Innovation and Modern Technology*, 10(5), 154-179. <https://doi.org/10.56201/wjimt.v10.no5.2026.pg154.179>
 86. Orise, C., & Niniola, S. (2022). Real-time analytics for educational excellence: A conceptual framework for performance dashboards in health professional training. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 8(2), 802-815.
 87. Oshoba, T. O., Hammed, N. I., & Odejobi, O. D. (2019). Secure identity and access management model for distributed and federated systems. *IRE Journals*, 3(4), 550-567.
 88. Oshoba, T. O., Hammed, N. I., & Odejobi, O. D. (2020). Blockchain-enabled compliance and audit trail model for cloud configuration management. *Journal of Frontiers in Multidisciplinary Research*, 1(1), 193-201.
 89. Oteri, O., & Edivri, J. (2024). Applied performance optimization frameworks for managing high-demand enterprise technology programs and system rollouts. *Gyanshauryam, International Scientific Refereed Research Journal*, 7(1), 188-210.
 90. Oteri, O., & Edivri, J. (2026). An operational reliability and service assurance framework for enterprise IT systems supporting large user populations. *Gulf Journal of Engineering and Technology*, 2(1), 1-19. <https://doi.org/10.51594/gjet.v2i1.205>
 91. Oyeleke, A. V., Eze, F. C., & Asiedu, W. (2026). Reliability-centered maintenance strategies for minimizing downtime and maximizing performance in high-density GPU cluster environments. *International Journal of Engineering Technology Research & Management*, 10(7), 18-38.
 92. Oyesiji, S. O., Nwakamma, S., & Ojukwu, J. (2025). Compressing recommender and ranking models for edge deployment: A survey. *International Journal of Engineering and Modern Technology*, 11(12), 205-237. <https://doi.org/10.56201/ijemt.vol.11.no12.2025.pg205.237>
 93. Pietzuch, P. R., & Bacon, J. M. (2002). Hermes: A distributed event-based middleware architecture. *Proceedings of the International Conference on Distributed Computing Systems Workshops*, 611-618.
 94. Pimentel, V., & Nickerson, B. G. (2012). Communicating and displaying real-time data with WebSocket. *IEEE Internet Computing*, 16(4), 45-53.
 95. Quainoo, R., & Ogundapo, O. (2026a). A system-level power behavior model for Bluetooth and Wi-Fi coexistence in dual-mode wireless devices. *International Journal of Computer Science and Mathematical Theory*, 12(2), 298-358. <https://doi.org/10.56201/ijcsmt.vol.12.no2.2026.pg298.358>
 96. Quainoo, R., & Ogundapo, O. (2026b). Wireless system-on-chip performance in next-generation Internet of Things devices: A systematic review of integrated transceiver testing methodologies. *World Journal of Innovation and Modern Technology*, 10(5), 76-126. <https://doi.org/10.56201/wjimt.v10.no5.2026.pg76.126>
 97. Quainoo, R., Ogundapo, O., & Asiedu, W. A. (2024). Modeling the impact of impedance mismatch on wireless link performance: A framework for prototype development and system optimization. *International Journal of Computer Science and Mathematical Theory*, 10(2), 62-116. <https://doi.org/10.56201/ijcsmt.v10.no2.2024.pg62.116>
 98. Quainoo, R., Ogundapo, O., & Asiedu, W. A. (2025a). Predicting throughput degradation in Wi-Fi 6 networks under varying channel conditions: A physical layer performance model. *International Journal of Engineering and Modern Technology*, 11(12), 205-264. <https://doi.org/10.56201/ijemt.vol.11.no12.2025.pg205.264>
 99. Quainoo, R., Ogundapo, O., & Asiedu, W. A. (2025b). Test automation in wireless hardware engineering: A comprehensive review of scripting frameworks and instrument control strategies. *International Journal of Engineering and Modern Technology*, 11(10), 405-464. <https://doi.org/10.56201/ijemt.vol.11.no10.2025.pg405.464>
 100. Richardson, C. (2018). *Microservices patterns: With examples in Java*. Manning Publications.

101. Shapiro, M., Preguiça, N., Baquero, C., & Zawirski, M. (2011). Conflict-free replicated data types. *Proceedings of the International Symposium on Stabilization, Safety, and Security of Distributed Systems*, 386-400.
102. Shittu, M. A., Shittu, H. A., Adeleke, O. J., & Adedokun, O. J. (2023). Digital twin modeling for real time monitoring and fault detection in smart substations. *International Journal of Industrial Engineering Research and Development*, 14(2), 25-44. https://doi.org/10.34218/IJIERD_14_02_003
103. Shittu, M. A., Shittu, H., Akintayo, T. A., & Opara, I. S. (2025). IoT enabled microgrid and BESS integration in ADMS: A framework for climate resilient grid operation. In *2025 IEEE Symposium on Wireless Technology & Applications (ISWTA)* (pp. 1-9). IEEE. <https://doi.org/10.1109/ISWTA68114.2025.11330023>
104. Skvorc, D., Horvat, M., & Srbljic, S. (2014). Performance evaluation of WebSocket protocol for implementation of full-duplex web streams. *Proceedings of the International Convention on Information and Communication Technology, Electronics and Microelectronics*, 1003-1008.
105. Sunday, E. A., & Omoegun, G. O. (2022). Smart fault detection in HVAC systems using sensor-based monitoring. *International Journal of Scientific Research in Science, Engineering and Technology*, 7(4), 380-403.
106. Terry, D. B., Theimer, M. M., Petersen, K., Demers, A. J., Spreitzer, M. J., & Hauser, C. H. (1995). Managing update conflicts in Bayou, a weakly connected replicated storage system. *Proceedings of the ACM Symposium on Operating Systems Principles*, 172-182.
107. Tonoyan, A., Dada, O., & Ayivi-Donkor, S. S. (2024b). Real-time KPI tracking systems: A review of automated performance monitoring and data-driven decision making. *World Journal of Innovation and Modern Technology*, 8(6), 247-281. <https://doi.org/10.56201/wjimt.v8.no6.2024.pg247.281>
108. Tonoyan, A., Dada, O., & Ayivi-Donkor, S. S. (2025). A conceptual model for real-time route optimization: Algorithmic pathways to delivery profitability and logistics cost reduction. *World Journal of Innovation and Modern Technology*, 9(12), 308-364. <https://doi.org/10.56201/wjimt.v9.no12.2025.pg308.364>
109. Ussher-Eke, D., Okoh, O. F., & Ijiga, O. M. (2025). The role of biometric and IoT-based attendance systems in streamlining HR administrative functions, enhancing workforce accountability, and reducing labor inefficiencies. *International Journal for Multidisciplinary Research*, 7(4).
110. Yusuf, M., Akintunde, A., Shittu, H., & Quadri, A. (2023). Radio frequency identification (RFID) based library management system. *International Journal of Technology and Systems*, 8(2), 21-35. <https://doi.org/10.47604/ijts.2041>