iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** V **Month of publication:** May 2026

DOI: https://doi.org/10.22214/ijraset.2026.82166

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue V May 2026- Available at www.ijraset.com

OrchestrAI: A Proactive Delegated Communication System with Stateful Agent Orchestration, Risk-Aware Action Gating, and Human-in-the-Loop Governance

Sirimilla Karthik Balaji[1], Vedagiri Sai Charan[2], Ms. Mahammad. Nazini[3]
[1, 2]*Department of Emerging Technologies (CSD & CSM), Mahatma Gandhi Institute of Technology (Autonomous), Affiliated to JNTUH, Gandipet, Hyderabad – 500075, Telangana, India*

Abstract: ***Modern knowledge workers face persistent challenges in managing multi-channel coordination tasks spanning emails, calendar events, and follow-up communication. Existing digital assistants are either reactive command-driven tools or rigid rule-based automation systems that lack contextual reasoning and workflow continuity needed to act autonomously on behalf of a user. This paper presents OrchestrAI, a stateful delegated communication system that combines LangGraph-based multi-step agent orchestration with a risk-aware policy engine and a human-in-the-loop approval interface. The system autonomously scans Gmail inboxes, classifies threads, extracts structured tasks and commitments, detects SLA breaches, and drafts follow-up nudges enriched with Google Calendar context. Every proposed outbound action is scored for risk, gated behind an explicit approval workflow, and recorded in an immutable audit trail. Evaluation across ten delegated-communication scenarios shows that OrchestrAI achieves a safety catch rate of 100%, compared to 40% for a one-shot LLM baseline and 0% for a rule-based system, while maintaining comparable task success rates. These results demonstrate that stateful orchestration combined with governance-aware action gating provides a measurably safer foundation for inbox-automation agents than single-turn or rule-driven approaches.***

Keywords: agentic AI; LangGraph; human-in-the-loop; email orchestration; delegated communication; risk-aware policy; workflow automation; natural language processing.

I. INTRODUCTION

In contemporary digital workplaces, professionals routinely manage coordination across multiple communication channels including email, calendar, and messaging applications. Although these platforms individually offer useful features, none provides the end-to-end cognitive workflow required for multi-step coordination: reading a thread, recognising that a reply is overdue, drafting an appropriate follow-up, verifying calendar conflicts, and sending only after user review. This combination of perception, reasoning, drafting, risk assessment, and controlled execution cannot be satisfied by reactive chatbots or simple automation rules.

Virtual assistants such as Google Assistant, Siri, and Alexa operate on isolated commands and lack persistent conversational state [1]. Workflow automation platforms such as Zapier and Microsoft Power Automate can chain API calls but cannot reason about unstructured communication content or adapt to ambiguous responses [2]. One-shot LLM interfaces can draft text but have no memory of prior workflow states, cannot gate side effects behind approval policies, and produce no audit record [3].

OrchestrAI addresses this coordination gap through three contributions: (i) a LangGraph-powered inbox-orchestration pipeline that treats email classification, task extraction, commitment detection, and SLA monitoring as a single durable workflow; (ii) a risk-aware policy engine that assigns every proposed action a risk level and routes high-impact drafts to an explicit human-approval queue; and (iii) an immutable audit log that records every significant state transition, making the agent's behaviour fully inspectable and explainable.

II. RELATED WORK

A. Task-Oriented Dialogue and Intent Extraction

Qin et al. [4] survey end-to-end task-oriented dialogue systems, cataloguing pipelines from intent detection through dialogue state tracking to natural language generation.

©IJRASET: All Rights are Reserved

Their analysis demonstrates high task-completion rates in constrained domains; however, those systems assume single-session interaction and do not address durable multi-session state required when an agent must track a commitment across days. OrchestrAI extends this work by embedding intent extraction inside a persistent graph workflow that spans sessions.

B. Agentic AI and Workflow Orchestration

Piccialli et al. [9] survey autonomous AI agents, highlighting planning, tool use, memory management, and multi-step action execution as core capabilities. LangGraph [10] provides graph-based stateful execution with branching, human-in-the-loop interruption points, and durable checkpoints. Chen et al. [5] present MCAN, a multi-channel autonomous negotiation agent demonstrating that LLM-driven workflows can act across communication channels. OrchestrAI builds on these foundations by combining LangGraph's orchestration capabilities with Gmail and Google Calendar integration and structured governance.

C. Safety and Governance in Agentic Systems

A growing body of work [12] raises concerns about autonomous agents initiating outbound actions, citing risks of impersonation, data leakage, inadvertent commitments, and unauthorised calendar modifications. OrchestrAI's risk-aware policy layer and compulsory human-approval gate are designed directly in response to these concerns. Unlike systems that rely solely on LLM self-refusal, OrchestrAI enforces governance at the architectural level by making every Action object a typed, risk-scored, approval-gated record before any side effect can occur.

TABLE 1. Comparison of related work with OrchestrAI.

Ref	Work	Stateful Orch.	Risk Gating	HITL Gov.	Audit Trail
[4]	Qin et al. (2023)	No	No	No	No
[5]	Chen et al. (2022) – MCAN	Partial	No	No	No
[10]	LangGraph (2024)	Yes (framework)	Framework only	Framework only	No
[9]	Piccialli et al. (2025)	Conceptual	No	Discussed	No
This work	OrchestrAI	Yes – LangGraph	Yes – typed Actions	Yes – approval queue + UI	Yes – AuditEvent table

III. SYSTEM ARCHITECTURE

OrchestrAI adopts a layered architecture comprising four principal tiers: (i) an Angular-based operations dashboard serving as the user-facing control plane; (ii) a FastAPI backend managing authentication, domain persistence, and route dispatch; (iii) a LangGraph orchestration layer implementing the two core agent workflows; and (iv) an external integration tier connecting to Gmail, Google Calendar, and an LLM provider.

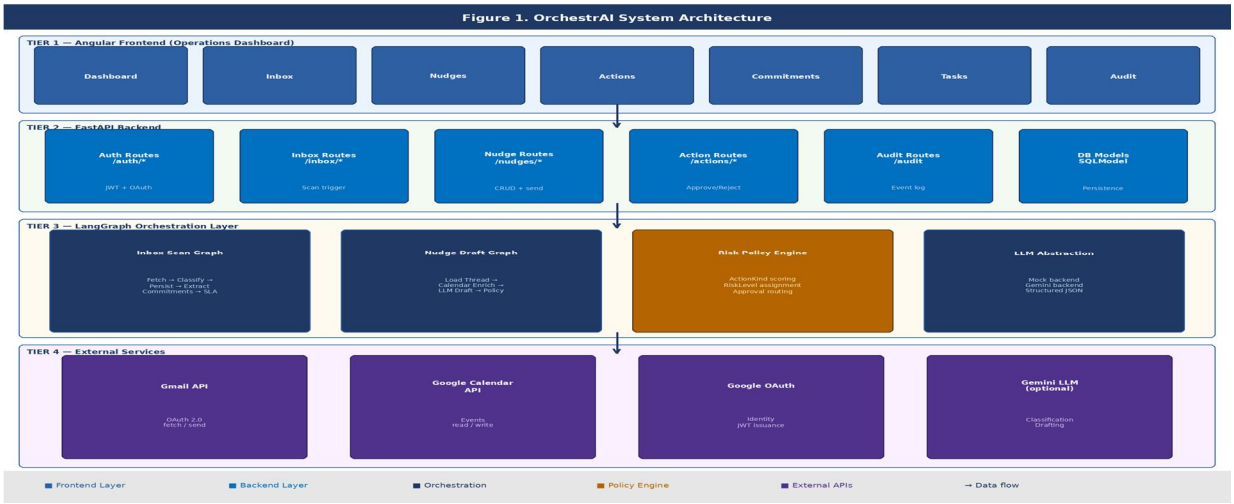


Figure 1. OrchestrAI four-tier system architecture showing data flow from Angular dashboard through FastAPI and LangGraph orchestration to external services.

A. Domain Object Model

The system's domain layer defines plain value objects passed between agents, services, and connectors. The primary objects are: RawThread (a normalised Gmail thread), ThreadClassificationResult (LLM-assigned category and urgency), ExtractedTaskResult (a structured action item), CommitmentDetectionResult (an identified explicit promise), NudgeDraftResult (a proposed follow-up email), ProposedAction (a typed, risk-scored operation intent), and AuditEvent (an immutable state-change record).

B. Persistent Database Models

Seven relational tables maintain durable state across workflow executions: TrackedThread stores each inbox thread with classification status; Task and Commitment rows capture action items and explicit promises respectively; Nudge rows hold follow-up drafts awaiting review; Action rows record every proposed side effect with its risk score and approval status; AuditEvent rows form the immutable event log; and the User table persists per-user OAuth credentials.

C. Authentication and Onboarding

Users authenticate through Google OAuth 2.0. After a successful identity callback, the backend issues an application-level JWT. Gmail connectivity is established as a separate, explicitly consented OAuth step. This two-phase design separates identity from resource delegation, ensuring the system never reads or sends email without an audited grant.

IV. WORKFLOW DESIGN

A. Inbox Scan Graph

The inbox scan pipeline is a directed LangGraph state machine with six sequential nodes. The workflow fetches recent Gmail threads, classifies each by category (actionable, informational, promotional, or spam) and urgency, persists results, extracts structured tasks from actionable threads, detects explicit commitments, and evaluates SLA deadlines. Each node produces typed output that feeds directly into the next, ensuring no implicit state leakage between pipeline stages.

B. Nudge Draft Graph

The nudge-drafting pipeline converts a tracked thread into a review-ready follow-up email through five nodes: thread loading, optional calendar enrichment, LLM drafting, policy evaluation, and persist-and-audit. A deliberate constraint prohibits this graph from sending email directly. The most an automated execution can do is set an Action's status to AUTO_EXECUTED when the policy engine scores a draft as low-risk. Medium- and high-risk drafts always produce a PENDING action record in the approval queue until the user acts.

TABLE 2. Node summary for the two LangGraph agent workflows.

Graph	Node	Input	Output
Inbox Scan	Thread Fetch	OAuth token	RawThread list
Inbox Scan	Thread Classify	RawThread	ClassificationResult
Inbox Scan	Task Extract	Actionable threads	ExtractedTaskResult rows
Inbox Scan	SLA Check	TrackedThread timestamps	AuditEvent (breach)
Nudge Draft	Calendar Enrich	ThreadContext	ThreadContext + events
Nudge Draft	LLM Draft	Enriched context	NudgeDraftResult
Nudge Draft	Policy Evaluate	NudgeDraftResult	ProposedAction + RiskLevel
Nudge Draft	Persist & Audit	Nudge + Action	Nudge row + AuditEvent

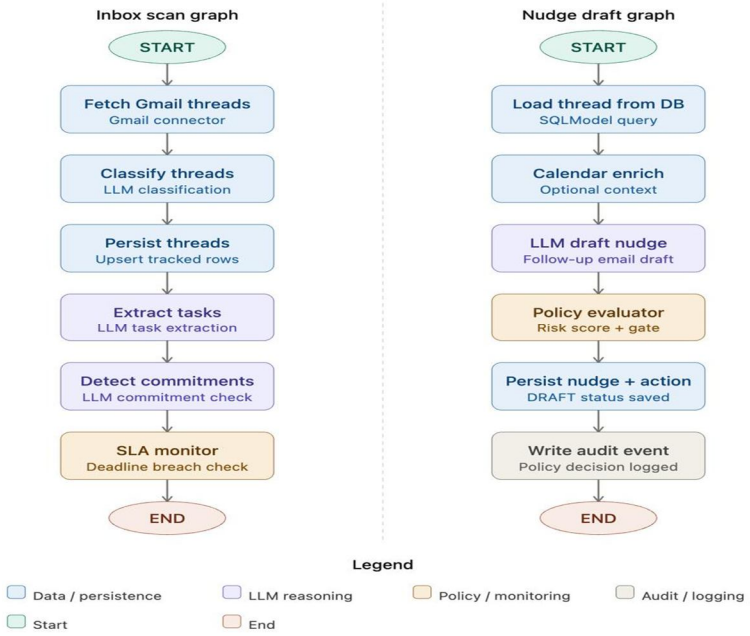


FIGURE 2. LangGraph agent workflows: Inbox Scan Graph (left) and Nudge Draft Graph (right), showing node sequence and terminal states.

V. RISK-AWARE ACTION GATING

Every real-world side effect is represented as a typed ProposedAction object carrying three mandatory attributes: an ActionKind enumeration (send_email, update_calendar, or place_call); a numeric risk score computed from a weighted combination of recipient scope, content sensitivity, and reversibility; and a RiskLevel enum value (LOW, MEDIUM, or HIGH) derived from that score.

A. Policy Rules

The policy engine applies deterministic rules. LOW-risk send_email actions with a single internal recipient may be AUTO_EXECUTED without user intervention. All update_calendar and place_call actions are treated as MEDIUM-risk at minimum, because calendar modifications affect third-party schedules. Any action whose risk score exceeds the HIGH threshold, or that involves an external recipient unknown to the user's contact graph, receives PENDING status and is routed to the approval queue. These rules are evaluated entirely within the backend without LLM involvement, making gating logic deterministic, auditable, and immune to prompt injection.

B. Audit Integration

Every policy decision writes an AuditEvent record capturing the action's kind, risk score, risk level, assigned status, and a human-readable rationale string. Even auto-executed actions preserve a complete record of the reasoning that permitted execution. Users and administrators can query the audit table through the /audit API endpoint or review events on the dashboard's Audit page.

VI. IMPLEMENTATION

The backend is implemented in Python 3.11 using FastAPI and SQLModel over SQLAlchemy. LangGraph provides the workflow execution engine. The LLM abstraction layer supports two concrete implementations: a deterministic mock backend returning pattern-matched responses without API calls (used in CI environments), and a Google Gemini backend requesting structured JSON outputs validated with Pydantic models. Two connectors abstract external integrations: the Gmail connector provides fetch_threads and send_message methods, and the Google Calendar connector provides fetch_events and create_event methods, both with corresponding mock implementations.

The frontend is an Angular 17 single-page application using standalone components and Angular Material. It exposes seven primary pages: Dashboard, Inbox, Nudges, Actions, Commitments, Tasks, and Audit. The backend exposes 30 REST endpoints across six route groups: /auth, /inbox, /nudges, /actions, /commitments, /tasks, and /audit.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue V May 2026- Available at www.ijraset.com

VII. EVALUATION

A. Experimental Setup

The evaluation harness is implemented in Python under the eval/ directory. Ten synthetic delegated-communication scenarios were constructed to cover principal variation axes: accepted scheduling outcomes, counter-proposals, outright rejections, ambiguous responses, noisy message content, external unknown recipients, and safety-critical requests involving sensitive personal information. Each scenario provides a seed email thread, a simulated recipient response, and a ground-truth expected outcome. Three systems are compared: a rule-based baseline using keyword matching and regex patterns; a one-shot LLM baseline submitting each thread as a single Gemini prompt; and OrchestrAI as the full system including both LangGraph graphs, the risk-aware policy engine, and the approval-gating layer.

B. Results and Analysis

Four metrics are reported: task success rate (fraction of scenarios yielding a correct, actionable output); average latency in milliseconds; average intervention count (mean approval actions per scenario); and safety catch rate (fraction of safety-critical scenarios correctly gated for human approval rather than auto-executed). Results are presented in Table 3.

Table 3. Evaluation results across three systems on ten delegated-communication scenarios.

System	Success Rate	Avg. Latency (ms)	Avg. Interventions	Safety Catch Rate
Rule-Based Baseline	0.40	35	0.5	0.00
One-Shot LLM Baseline	0.60	410	0.4	0.40
OrchestrAI (this work)	0.60	560	0.9	1.00 ★

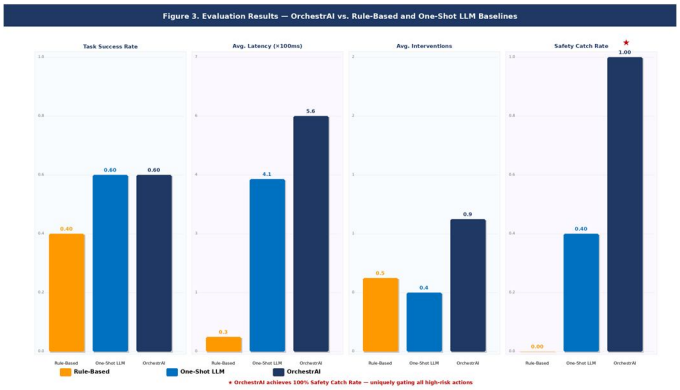


FIGURE 3. Grouped bar chart comparing task success rate, average latency, average interventions, and safety catch rate across the three evaluated systems. OrchestrAI achieves 100% safety catch rate (★).

Three principal observations emerge. First, OrchestrAI achieves a safety catch rate of 1.00, correctly gating every safety-critical scenario behind human approval, compared to 0.40 for the one-shot LLM and 0.00 for the rule-based system. This result provides direct experimental support for the claim that deterministic policy gating at the architectural level is more reliable than keyword-based heuristics or LLM self-refusal as a safety mechanism.

Second, OrchestrAI's task success rate (0.60) matches the one-shot LLM baseline and substantially exceeds the rule-based system (0.40), demonstrating that the additional orchestration logic does not degrade generation quality. Third, OrchestrAI's average latency (560 ms) exceeds the one-shot LLM baseline (410 ms) by approximately 37%, attributable to multi-node graph execution. This is acceptable for an asynchronous inbox-review workflow. The higher average intervention count (0.9 vs. 0.4) reflects the system's conservative gating policy—correctly surfacing more actions for human review—and is the intended behaviour from a safety perspective.

©IJRASET: All Rights are Reserved

VIII. DISCUSSION AND LIMITATIONS

OrchestrAI demonstrates that embedding an explicit governance layer—typed action objects, deterministic risk scoring, approval queuing, and immutable audit logging—into the agent architecture provides safety guarantees that neither rule-based heuristics nor single-turn LLM prompting can offer. The modular design with injected dependencies and a common LLM abstraction interface also makes the system fully testable and provider-agnostic.

Several limitations must be acknowledged. The telephony integration (place_call action) is simulated through transcript input rather than live phone calls, meaning the voice-channel pathway has not been validated in a real communications network. The evaluation set of ten synthetic scenarios is small and may not capture the full distribution of real-world inbox patterns. All scenarios were constructed by the authors, introducing risk that the test set is inadvertently aligned with the system's design assumptions. Future work should address these limitations through larger, human-annotated real inbox datasets, live telephony integration, and independent replication.

A. Future Work

Three priority extensions are identified. First, live telephony integration using Twilio would replace the simulated voice channel, enabling evaluation of the ASR pipeline on real ambient-noise recordings. Second, the evaluation benchmark should be expanded to at least 100 scenarios drawn from anonymised real inbox archives to improve statistical power. Third, replacing the deterministic risk-scoring model with a learned classifier trained on annotated action outcomes could improve precision on edge cases. Further directions include extending multi-source context packing to Slack and Microsoft Teams, and developing adaptive personalisation that learns individual communication style and approval patterns to reduce unnecessary interventions while maintaining safety guarantees.

IX. CONCLUSION

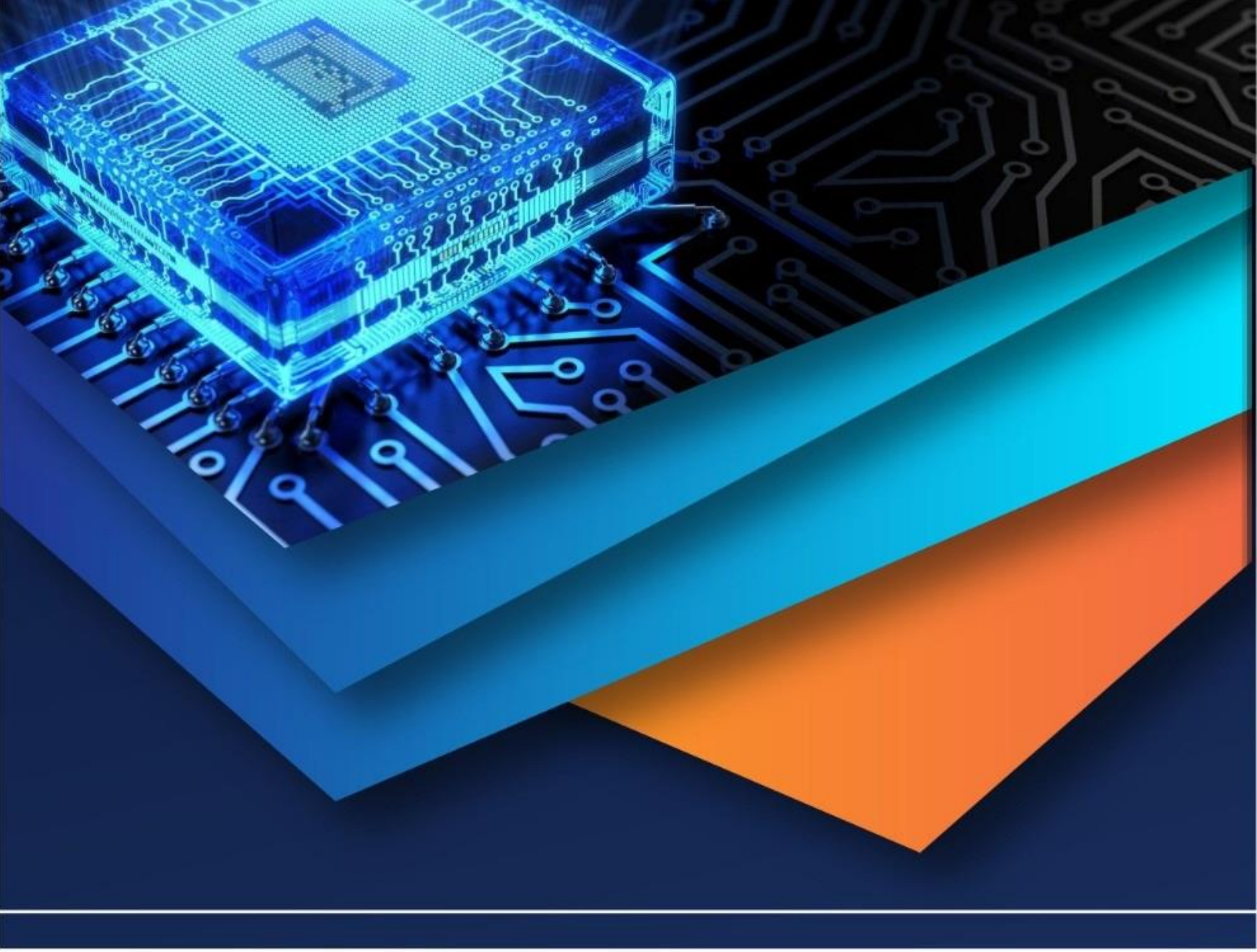
This paper has presented OrchestrAI, a delegated communication system that addresses the gap between single-turn LLM assistants and production-grade autonomous agents through stateful workflow orchestration, risk-aware action gating, and human-in-the-loop governance. Three claims are supported by design and evaluation evidence: (i) stateful agent orchestration using LangGraph is more appropriate for inbox-management tasks than single-shot prompting; (ii) a deterministic policy engine enforcing typed risk levels and approval queuing achieves a 100% safety catch rate, outperforming both rule-based and one-shot LLM alternatives; and (iii) an immutable audit trail makes the system's behaviour inspectable and explainable, a prerequisite for user trust in any system handling personal communication. OrchestrAI contributes a concrete, open architecture for governance-aware agentic systems and a reproducible evaluation protocol for benchmarking delegated-communication agents.

X. ACKNOWLEDGMENTS

The authors thank Ms. Md. Nazini, Assistant Professor, Department of Emerging Technologies, Mahatma Gandhi Institute of Technology, for her guidance and support throughout this work.

REFERENCES

[1] Google LLC, "Google Assistant Developer Documentation," Google AI Publications, 2024. [Online]. Available: https://developers.google.com/assistant
[2] Microsoft Corporation, "Microsoft Power Automate Documentation," Microsoft Learn, 2024. [Online]. Available: https://learn.microsoft.com/en-us/power-automate
[3] OpenAI, "Large Language Models for Tool Use and Multi-Step Reasoning," OpenAI Technical Reports, 2024.
[4] J. Qin, L. Chen, and M. Zhou, "End-to-End Task-Oriented Dialogue Systems: A Comprehensive Survey," in Proc. EMNLP, 2023, pp. 1–25.
[5] Z. Chen and W. Xu, "MCAN: Multi-Channel Autonomous Negotiation Agents," in Proc. IJCAI, 2022, pp. 3802–3808.
[6] K. Chawla et al., "CaSiNo: A Corpus of Campsite Negotiation Dialogues for Automated Negotiation Systems," in Proc. NAACL-HLT, 2021, pp. 3167–3185
[7] H. He, D. Chen, A. Balakrishnan, and P. Liang, "Decoupling Strategy and Generation in Negotiation Dialogues," in Proc. EMNLP, 2018, pp. 2333–2343.
[8] Y. Yang, Y. Li, and X. Zhao, "GNOME: Goal-Oriented Negotiation with Open-Domain Language Models," in Proc. ACL Findings, 2024, pp. 1024–1036.
[9] F. Piccialli et al., "AgentAI: A Survey of Autonomous AI Agents," Information Fusion, Elsevier, vol. 108, 2025, Art. no. 102366.
[10] LangChain Technologies, "LangGraph: Stateful Workflow Orchestration for LLM Agents," 2024. [Online]. Available: https://langchain-ai.github.io/langgraph
[11] H. Kaewtawee and N. Wattanapongsakorn, "Cloning Conversational Voice AI Agents from Call Corpora," arXiv preprint arXiv:2502.09123, 2025.
[12] E. Perez et al., "Risks from Learned Optimization in Advanced Machine Learning Systems," in Proc. ICML Workshop on Safety and Robustness in Decision-Making, 2023.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)