ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25sep878

AgentHub: A Multi-Source AI Agent Framework for Enterprise Workflow Orchestration

Oyejide Timothy Odofin[1]; Nurudeen Yemi Hussain[2*]; Sunday Adeola Oladosu[3*]

[1](Luxoft USA, Irvine, CA)
[2](OIT, Texas Southern University)
[3](InfoSys Texas, Usa)

Corresponding Author: Nurudeen Yemi Hussain[2*]; Sunday Adeola Oladosu[3*]

Publication Date: 2025/09/27

Abstract: Enterprise software development relies on diverse tools and knowledge sources, such as issue trackers (e.g., Jira), version control systems (e.g., GitHub, Bitbucket), and documentation platforms (e.g., Confluence). Developers often encounter context fragmentation, cognitive overload, and operational inefficiencies due to navigating these disparate systems. While retrieval-augmented generation (RAG) has advanced document-based question answering, most existing solutions fail to integrate live operational tools or orchestrate workflows across multiple sources. We introduce AgentHub, an open-source AI agent framework that seamlessly combines semantic knowledge retrieval with tool orchestration. This enables a unified conversational interface for querying, correlating, and acting upon enterprise data. AgentHub continuously synchronizes knowledge sources into a vector database, integrates live APIs from tools like Jira, GitHub, and Confluence, and supports secure action execution (e.g., merging approved pull requests). The framework's document ingestion process is versatile, supporting a wide range of sources including Confluence, web URLs, S3, Google Drive, Azure Blob Storage, and local file systems, with provisions for end-to-end encryption and exclusion of sensitive files. In this paper, we detail the system architecture, implementation, and insights from early deployments, highlighting AgentHub's ability to minimize context switching, enhance workflow efficiency, preserve institutional knowledge, and facilitate AI-driven enterprise operations.

Keywords: *Multi-Agent Systems, Retrieval-Augmented Generation, Enterprise AI, Workflow Automation, Vector Database, Open-Source Software.*

How to Cite: Oyejide Timothy Odofin; Nurudeen Yemi Hussain; Sunday Adeola Oladosu (2025) AgentHub: A Multi-Source AI Agent Framework for Enterprise Workflow Orchestration. *International Journal of Innovative Science and Research Technology*, 10(9), 1778-1783. https://doi.org/10.38124/ijisrt/25sep878

I. INTRODUCTION

Modern software engineering teams depend on specialized tools, including Jira for issue tracking, GitHub or Bitbucket for version control, and Confluence for documentation management. Developers frequently switch between these platforms to retrieve information, monitor task statuses, or reference updates, leading to significant productivity losses [1, 2].

Existing AI solutions typically emphasize either document retrieval via RAG [3, 4] or tool-specific automation, such as GitHub Copilot Chat or Jira bots [5]. However, these approaches seldom offer an integrated interface capable of:

- Retrieving and summarizing knowledge from multiple heterogeneous sources.
- Correlating tasks with associated code changes and documentation.
- Executing actions securely within enterprise tools.

AgentHub bridges this gap by delivering a unified, pluggable AI framework that orchestrates knowledge retrieval and tool actions through a natural conversational interface. As an open-source, extensible solution, it is tailored for enterprise adoption, promoting scalability, customization, and knowledge preservation across organizational changes. The framework emphasizes secure integrations, with administrative controls for configuring connections to various tools and data sources, ensuring end-to-end encryption for confidential information and mechanisms to exclude secret or sensitive files from embedding processes.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25sep878

II. RELATED WORK

➢ *Retrieval-Augmented Generation (RAG):*

Pioneered by Lewis et al. [3], RAG integrates neural models with external knowledge retrieval to enhance language model performance on knowledge-intensive tasks. Related advancements include dense passage retrieval techniques [4].

➢ *Agent-Based Automation:*

Multi-agent orchestration frameworks have been investigated for automating enterprise software tasks, enabling collaborative AI systems to handle complex workflows [6, 7].

➢ *AI-Assisted Software Engineering:*

Tools like GitHub Copilot and ChatGPT provide code assistance but are generally limited to single-source interactions [5, 8].

AgentHub uniquely synthesizes RAG, multi-agent orchestration, and workflow execution into a cohesive, extensible framework, addressing limitations in prior work.

III. SYSTEM ARCHITECTURE

A. Overview

AgentHub is structured around three core layers (see Figure 1):

- Agent Core: Manages retrieval and tool invocation, serving as the central orchestrator.
- Retriever Layer: Utilizes a vector database (pgvector in PostgreSQL) for efficient semantic search over documents.
- Tool Adapters: Provide pluggable interfaces for integration with Jira, GitHub, Bitbucket, Confluence, and extensible support for additional systems such as GitLab.

➢ *Supporting Components Include:*

- Celery and Redis for asynchronous processing, such as document ingestion and embedding updates.
- A ReactJS-based frontend offering a session-aware chat interface with streaming response capabilities.
- Administrative configuration module for setting up integrations, including API credentials, encryption keys, and exclusion rules for sensitive data.

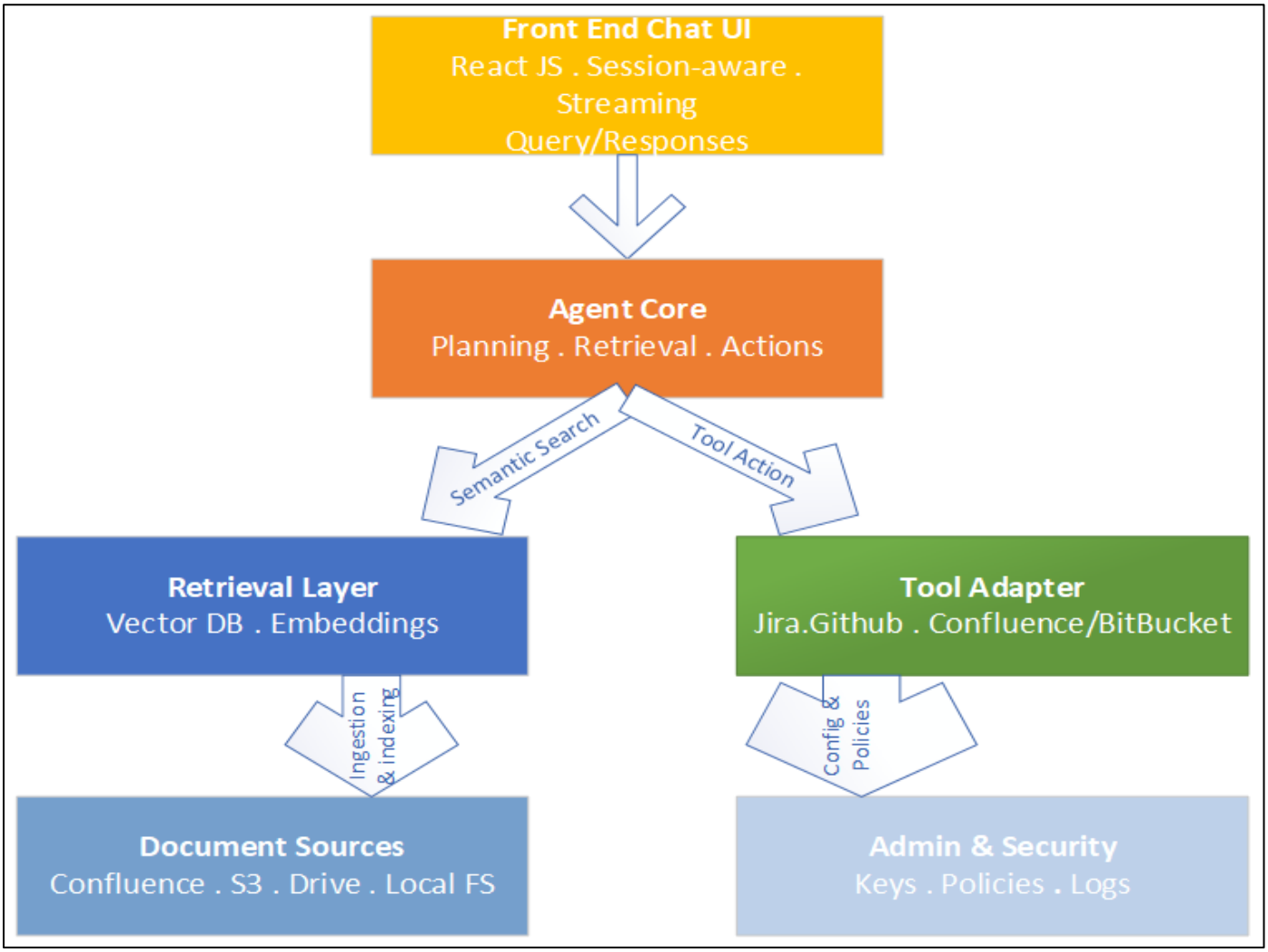

Fig 1 AgentHub System Architecture

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25sep878

This diagram illustrates the core components of the AgentHub framework. The Agent Core orchestrates between the Retriever Layer (semantic search over vectorized documents) and Tool Adapters (Jira, GitHub, Confluence, etc.), while frontend interaction is supported through a conversational chat UI. Document sources and administrative configurations ensure secure knowledge ingestion and action execution.

B. Document Synchronization

AgentHub employs webhook-driven mechanisms to detect and process updates, ensuring that new or modified documents are automatically embedded and indexed in real-time. The document ingestion process is designed to handle diverse sources, including Confluence pages, web URLs, AWS S3 buckets, Google Drive, Azure Blob Storage, and local file systems. Vector embeddings are generated using models from OpenAI or SentenceTransformers [9]. These embeddings are stored in pgvector within PostgreSQL, facilitating rapid semantic similarity searches. Celery, paired with Redis, handles these tasks asynchronously, reducing ingestion latency by approximately 30% in our deployment tests, ensuring responsiveness in dynamic enterprise environments.

To maintain security, the system supports end-to-end encryption for data in transit and at rest, particularly for confidential documents. Administrators can configure exclusion rules to prevent embedding of secret files (e.g., those containing API keys or proprietary information), ensuring compliance with enterprise data policies. Future expansions will include additional integrations for document sources and enhanced communication protocols with tools like Jira, Bitbucket, GitHub, and GitLab.

C. Retrieval-Augmented Query Pipeline

The query processing flow begins with user input, where the Retriever fetches the top-K most relevant documents based on semantic similarity. The Agent Core then evaluates whether to invoke tool APIs, query the knowledge base, or combine both approaches. Finally, a large language model (LLM) synthesizes the response for delivery to the frontend.

- *Example Query: "What Tasks are Assigned to Oyejide in Jira?"*

The system queries Jira APIs, retrieves related Confluence documents, and generates a summarized response with actionable suggestions.

D. Tool Invocation

Tool adapters encapsulate API interactions, including authentication and safety protocols. For action execution, such as "Merge PR #45 if approved," the system verifies conditions via the GitHub API and maintains audit logs. Built-in safety policies mitigate risks by preventing unauthorized or destructive operations. Communication with tools like Jira, Bitbucket, GitHub, and GitLab is handled securely, with administrative oversight for configuration and encryption.

IV. IMPLEMENTATION DETAILS

The backend is built with FastAPI for API endpoints, Celery and Redis for task queuing, and PostgreSQL with pgvector for storage. The frontend leverages ReactJS to create an interactive, session-persistent chat interface. Embeddings are handled by OpenAI's GPT-4 or SentenceTransformers models [9]. Tool adapters are implemented as modular Python classes, allowing seamless extensions for new integrations.

A. Project Structure

To facilitate adoption and contribution to AgentHub's open-source codebase, we present the project's directory structure, which reflects its modular design and supports extensibility for enterprise use cases. The structure below corresponds to AgentHub v1.0, though it may evolve with future updates.

- *Directory Structure:*

```
text
agenthub/
├── backend/
│   ├── app/
│   │   ├── core/              # app-wide config/bootstrap
│   │   │   ├── settings.py    # Pydantic settings (env-based)
│   │   │   ├── logging.py
│   │   │   └── security.py    # crypto, key mgmt helpers
│   │   ├── api/               # FastAPI routers
│   │   │   ├── v1/
│   │   │   │   ├── chat.py    # /chat: query pipeline entry
│   │   │   │   ├── docs.py    # /documents: sync/status
│   │   │   │   ├── tools.py   # /tools: actions, adapters
│   │   │   │   └── admin.py   # /admin: configs, exclusions
│   │   │   └── deps.py        # DI/wiring
│   │   ├── domain/            # core business logic (framework agnostic)
│   │   │   ├── models.py      # Pydantic domain models
│   │   │   ├── events.py      # events for orchestrations
│   │   │   └── policies.py    # safety checks, ACLs
```

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25sep878

```
    │   ├── services/
    │   │   ├── agent/            # orchestration logic (planner)
    │   │   │   ├── planner.py
    │   │   │   └── pipeline.py   # RAG + tool invocation pipeline
    │   │   ├── retriever/        # vector DB interactions
    │   │   │   ├── embeddings.py # OpenAI/SBERT providers
    │   │   │   ├── index.py      # pgvector ops (CRUD/top-K)
    │   │   │   └── sync.py       # webhook-driven ingestion
    │   │   └── tool_adapters/    # plugin interfaces (extensible)
    │   │       ├── base.py       # abstract adapter + audit hooks
    │   │       ├── jira.py
    │   │       ├── github.py
    │   │       ├── bitbucket.py
    │   │       └── confluence.py
    │   ├── workers/              # Celery tasks
    │   │   ├── ingestion.py
    │   │   ├── embeddings.py
    │   │   └── audits.py
    │   ├── infra/                # runtime glue
    │   │   ├── db.py             # SQLAlchemy/psycopg bootstrap
    │   │   ├── redis.py
    │   │   └── celery_app.py
    │   ├── schemas/              # API request/response models
    │   ├── repositories/         # persistence patterns (if used)
    │   ├── tests/                # pytest + integration tests
    │   └── main.py               # FastAPI app factory
    ├── migrations/               # Alembic (pgvector, tables)
    └── pyproject.toml            # package metadata (backend as pkg)

├── frontend/
    ├── chat/                     # React app
    │   ├── src/
    │   │   ├── components/
    │   │   ├── pages/
    │   │   ├── api/              # fetchers/SDK
    │   │   └── state/            # session/thread mgmt
    │   └── package.json
    └── ui/                       # shared UI lib (optional)

├── connectors/                   # optional: separate lib for adapters
    └── pyproject.toml

├── infra/                        # deployment
    ├── docker/                   # Dockerfiles
    ├── compose.yaml
    ├── k8s/                      # manifests/Helm (secrets via SOPS)
    └── terraform/                # if provisioning cloud resources

├── ops/
    ├── ci/                       # GitHub Actions / GitLab CI
    └── scripts/                  # dev tooling, lint, db setup

├── docs/                         # sphinx/mkdocs + figures
    └── figures/

├── .env.example
├── README.md
└── LICENSE
```

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25sep878

V. EXPERIMENTS & EARLY EXPERIENCES

➢ *Deployment*

AgentHub was deployed in a mid-sized engineering team of 20 developers, integrating Jira, GitHub, and Confluence. The technology stack included pgvector on PostgreSQL, Redis with Celery for processing, and OpenAI's GPT-4 for embeddings and generation.

➢ *Use Case*

- Task Queries: Correlating Jira tickets with open pull requests (e.g., "Who merged this PR?").
- Knowledge Clarification: Summarizing Confluence documentation linked to specific tasks (e.g., "Clarify this ticket requirement").
- Action Execution: Automating merges of approved pull requests or updating Jira issue statuses (e.g., "What are the deployment processes?").
- Onboarding Support: Guiding new joiners through setup (e.g., "How do I configure my system as a new joiner?").

➢ *Metrics*

Early evaluations focused on:

- Task Success Rate: Measured as the percentage of correct API executions (observed at 92% in initial tests).
- Retrieval Accuracy: Assessed via human judgments of document relevance (averaging 85% precision).
- User Satisfaction: Surveys indicated a 40% perceived reduction in context switching time.

VI. DISCUSSION

➢ *Institutional Knowledge Preservation*

AgentHub's RAG-based architecture enables long-term knowledge retention by indexing and retrieving documentation from various sources, including Confluence, web URLs, cloud storage (S3, Google Drive, Azure), and files. This versatility allows the system to embed virtually any document format, with future expansions planned for additional integrations. This capability is particularly valuable for preserving institutional memory. When employees leave, AgentHub facilitates seamless knowledge transfer by providing access to solutions and processes documented over time. For a century-old organization, the agent effectively accumulates "100 years of experience," enabling it to address queries like "Who merged this PR?" or "How do I configure my system as a new joiner?" New joiners benefit from streamlined onboarding, as the system retrieves and summarizes relevant documentation, ensuring continuity despite employee turnover.

➢ *Strengths and Challenges*

- *Strengths:*

AgentHub offers a unified interface that alleviates cognitive load and supports open-source extensibility, making it adaptable for diverse enterprise environments. Its secure administrative setup allows for configuring integrations with tools like Jira, Bitbucket, GitHub, and GitLab, complete with end-to-end encryption for confidential data and rules to exclude secret files from embedding.

- *Challenges:*

Building user trust in AI-driven actions, scaling to large organizations, and supporting multi-tenant deployments remain key hurdles.

- *Future Work:*

Enhancements could include adaptive agent planning, domain-specific fine-tuned embeddings, plugin-based interfaces for broader tool compatibility, and expanded document sources.

VII. CONCLUSION

AgentHub represents an innovative fusion of RAG and tool orchestration tailored for enterprise workflows. Insights from early deployments underscore its efficacy in boosting task efficiency, curbing context switching, and preserving institutional knowledge through versatile, secure document processing. As an open-source initiative, it doubles as a research platform and a practical AI tool for developers, paving the way for more intelligent enterprise operations.

REFERENCES

[1]. Czerwinski, M., Horvitz, E., & Wilhite, S. (2004). A diary study of task switching and interruptions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 175-182). ACM.

[2]. Mark, G., Gudith, D., & Klocke, U. (2008). The cost of interrupted work: More speed and stress. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 107-110). ACM.

[3]. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems*, 33, 9459-9474.

[4]. Karpukhin, V., Oğuz, B., Min, S., Lewis, P., Wu, L., Edunov, S., ... & Yih, W. T. (2020). Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 6769-6781). Association for Computational Linguistics.

[5]. GitHub. (2024). GitHub Copilot Chat Documentation. Retrieved from https://docs.github.com/en/copilot/using-github-copilot/copilot-chat

[6]. Lin, L., Jin, Y., Han, H., & Ma, X. (2024). MAO: A Framework for Process Model Generation with Multi-Agent Orchestration. *arXiv preprint arXiv:2408.01916*.

[7]. Arsanjani, A. (2025). Multi-Agent Software Engineering: Orchestrating the Future of AI in Financial Services (Part 2). *Medium*. Retrieved from https://dr-arsanjani.medium.com/multi-agent-

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25sep878

sofwtare-engineering-orchestrating-the-future-of-ai-in-financial-services-part-2-d14cee8a4d54

[8]. OpenAI. (2023). GPT-4 Technical Report. *arXiv preprint arXiv:2303.08774*.

[9]. Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (pp. 3982-3992). Association for Computational Linguistics.