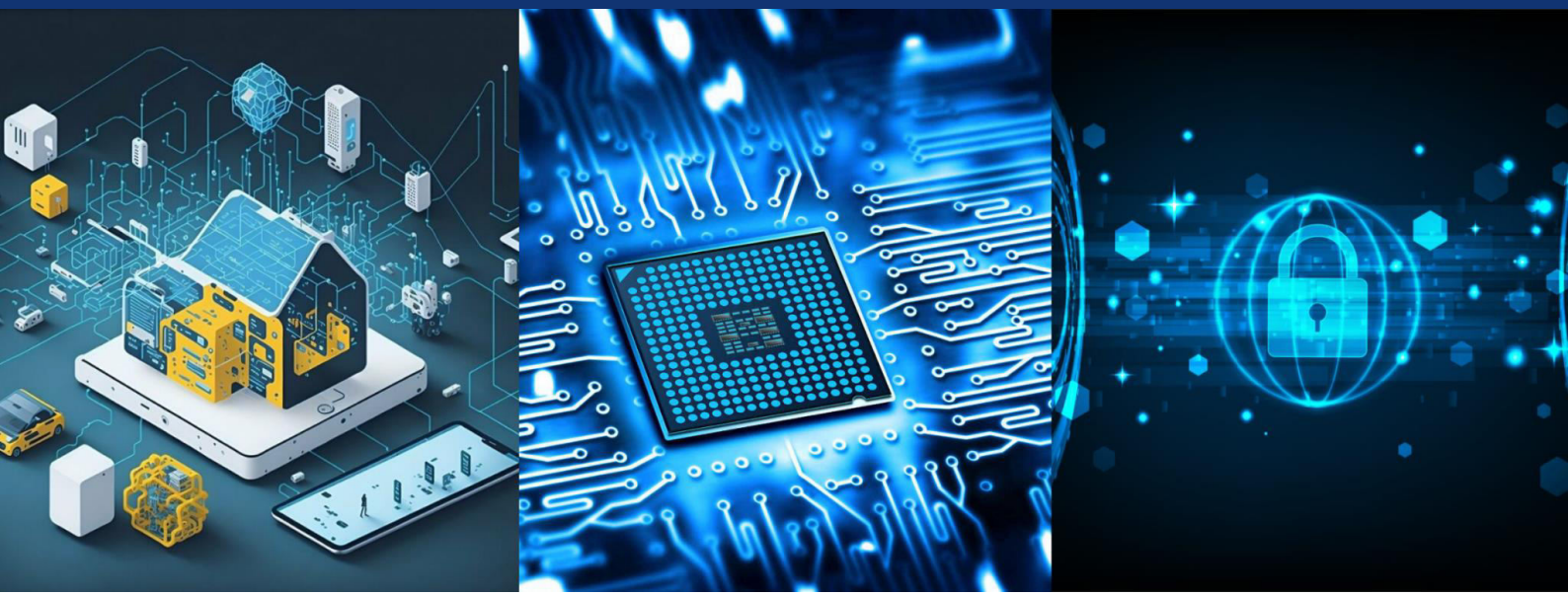
IJIRCCE

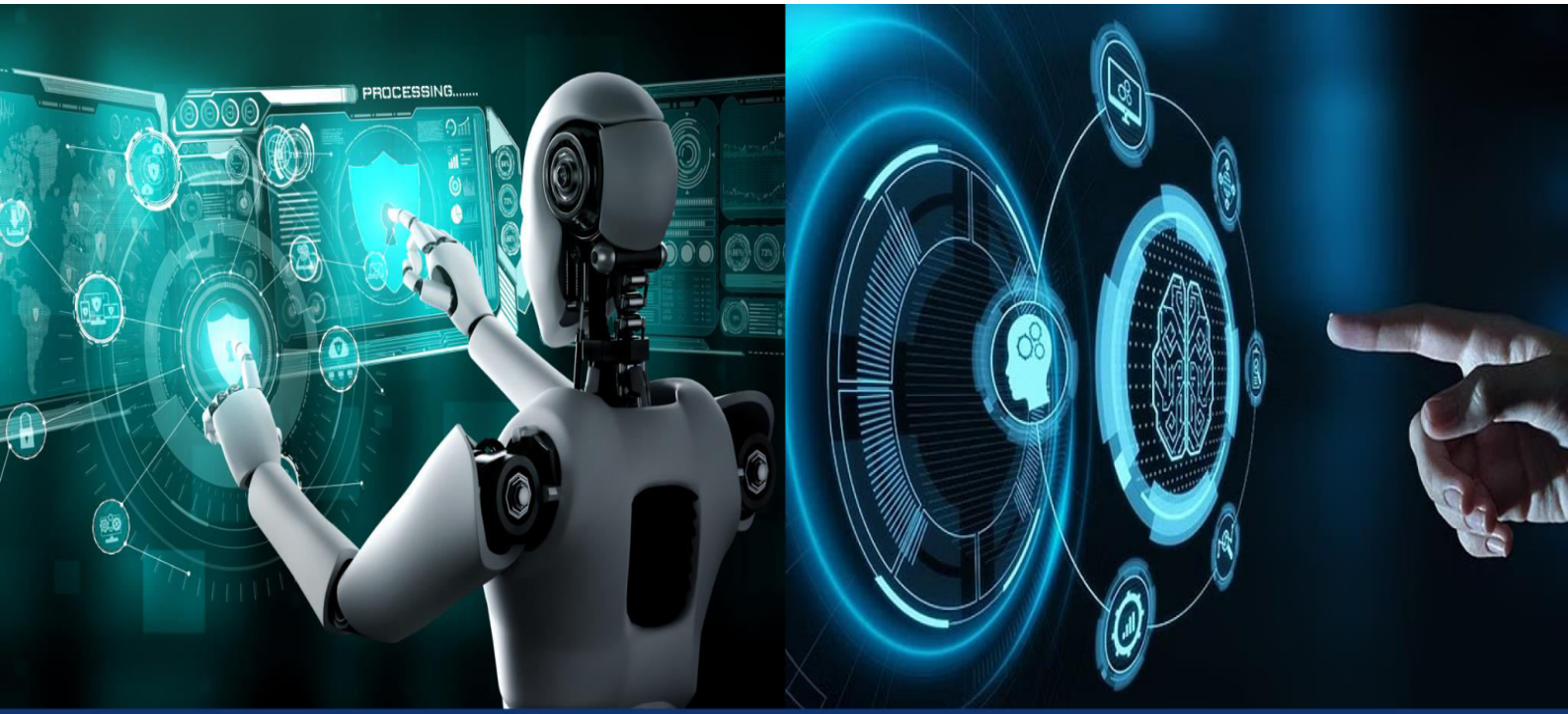
ISSN(O): 2320-9801

ISSN(P): 2320-9798



International Journal of Innovative Research in Computer and Communication Engineering

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)



Impact Factor: 8.771

Volume 14, Issue 4, April 2026

© 2026 IJIRCCE | Volume 14, Issue 4, April 2026 DOI: 10.15680/IJIRCCE.2026.1404011

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Intelligent Workflow Automation System for Organizational Task Management

Mrs. K. Thrilochana Devi[1], K. Bhargav Sai[2], K. Praneeth Rohan[2], M. Kowshik[2]

Assistant Professor, Dept. of Information Technology, Vasireddy Venkatadri Institute of Technology, Nambur, Guntur, Andhra Pradesh, India[1]

B. Tech Students, Dept. of Information Technology, Vasireddy Venkatadri Institute of Technology, Nambur, Guntur, Andhra Pradesh, India[2]

ABSTRACT: Task management in team-based organizational settings becomes increasingly complex as team sizes grow and project interdependencies multiply. Existing platforms such as Jira and Trello function primarily as tracking tools: they record the state of work but offer no intelligent decision support. We present the Intelligent Workflow Automation System for Organizational Task Management, a web-based platform built around three coordinated mechanisms. First, a Role-Based Access Control (RBAC) framework enforces strict boundaries across three user roles — Directors, Managers, and Employees — ensuring each user interacts only with data appropriate to their position. Second, a General AI Assistant answers natural language queries on organizational knowledge, leveraging the Spring AI framework with live database connectivity. Third, an Agentic AI Assistant provides real-time insight into employee workload, active task distribution, and automated deadline monitoring by calling live backend tool functions rather than relying on pre-trained knowledge. The system further incorporates a complete task lifecycle state machine, automated deadline alert notifications, in-platform collaboration through comment threads and file attachments, and team-level reporting. A pilot evaluation demonstrated measurable improvements in workload visibility, reduced manual query overhead, and high user satisfaction across all three roles. The results indicate that combining agentic AI with RBAC and structured workflow management produces a practical and extensible solution for organizational task automation.

KEYWORDS: Agentic AI; Role-Based Access Control; General AI Assistant; Spring AI; Workflow Automation; Task Monitoring; Deadline Monitoring; Organizational Productivity.

I. INTRODUCTION

Distributing work effectively across a team appears straightforward in principle but deteriorates quickly at scale. In small teams, a manager can maintain a mental picture of who is doing what and make reasonable decisions. That capacity erodes as team sizes increase, projects multiply, and task dependencies become too numerous to track informally. The consequences are familiar: some employees are chronically overloaded, others are underutilized, and deadlines are missed not because the work exceeded the team's capability but because no one had a reliable view of available capacity.

The dominant project management platforms do not meaningfully address this problem. Tools like Jira, Asana, and Trello are visibility instruments — they record what exists but offer no decision support for what to do about it [10]. As Kamila and Marzuq [10] observe, these platforms excel at tracking but lack mechanisms for intelligent workload balancing or real-time query support.

What is absent from the existing literature is a system that simultaneously integrates: (1) a well-defined Role-Based Access Control framework governing three distinct user roles; (2) a General AI Assistant capable of answering organizational knowledge queries in natural language; and (3) an Agentic AI Assistant that monitors employee workload, tracks deadline status, and retrieves live operational data through tool-calling architecture. This paper presents such a system.

Our contributions are: (1) a three-tier RBAC architecture enforcing organizational boundaries at the API level; (2) a General AI Assistant built on the Spring AI framework for answering general knowledge and organizational queries; (3) an Agentic AI Assistant that calls live backend functions to monitor workload, track task states, and raise deadline alerts

© 2026 IJIRCCE | Volume 14, Issue 4, April 2026 DOI: 10.15680/IJIRCCE.2026.1404011

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

— structurally preventing fabricated responses; (4) a task lifecycle state machine governing full task progression from creation through completion; and (5) integrated deadline monitoring, team-level reporting, and in-platform collaboration.

The remainder of the paper is structured as follows. Section II reviews related work. Section III describes the proposed system. Section IV covers implementation details. Section V presents evaluation results and discussion. Section VI concludes.

II. RELATED WORK

Several lines of prior work are relevant. We examine them in turn.

A. AI-Based Task Prioritization

Musthafa and Muthukumaran [1] developed a task management system using Random Forest supplemented by KNN and Linear Regression. Their key finding is that incorporating urgency signals and historical completion data into feature engineering improves prioritization substantially over deadline-only sorting. The system is designed for individual use, however, and does not address distributing tasks across a team or providing role-based access.

B. NLP-Based Task Categorization and Routing

ProjecTree [4], developed by Farooq et al., applies NLP at the task intake stage to classify incoming descriptions into domain categories and routes each task to employees whose declared skill profiles match the predicted category, reporting classification accuracy of approximately 99%. The limitation is structural: routing is a fixed lookup that cannot adapt based on how prior assignments performed, nor can it answer live workload queries from users.

C. Workforce Management and HR Automation

Suvedhaa et al. [2] built a comprehensive system covering automated appraisal generation using retrieval-augmented generation, project assignment via Skill2Vec embeddings, and performance categorization using DistilBERT. Its orientation, however, is toward annual review cycles rather than daily task workflow and deadline monitoring. There is no mechanism for users to query the system in natural language and receive a response derived from current live data.

D. Hybrid Neural Approaches for Task Classification

Mane et al. [3] describe a hybrid neural model combining TF-IDF text features with numerical metadata to predict whether a task should be flagged as high priority. Five-fold cross-validation yields an average accuracy of 77.2%. The model produces a binary priority label rather than operational insight, placing it one step away from practical workload monitoring.

E. Gap Analysis

Across the reviewed systems, three things are consistently absent. None provides a conversational assistant that retrieves answers from live database records at query time. None implements a complete task lifecycle with explicit state transitions that other system components can act upon. And none combines role-based access control with real-time agentic workload monitoring and a general knowledge assistant within a single integrated platform. This work addresses all three gaps.

III. PROPOSED SYSTEM

The system is a client-server web application organized around three coordinating subsystems: a Role-Based Access Control framework, a General AI Assistant for knowledge queries, and an Agentic AI Assistant for workload and deadline monitoring. These subsystems share a common data layer and operate within a well-defined task lifecycle.

A. System Architecture

The backend is implemented in Java 17 using the Spring Boot framework, organized into a standard layered architecture. Controllers handle incoming HTTP requests and route them to the service layer, which contains business logic. The repository layer manages persistence through Spring Data JPA, connected to a relational database. The frontend is a React application that renders role-specific dashboard views and communicates with the backend through REST APIs.
Both AI assistants are integrated directly within the Spring Boot application using the Spring AI framework. Spring AI provides the infrastructure for registering tool functions, submitting messages and tools to the underlying language model,

© 2026 IJIRCCE | Volume 14, Issue 4, April 2026 DOI: 10.15680/IJIRCCE.2026.1404011

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

and processing tool call responses. This architecture keeps the AI components tightly coupled to the live database, ensuring that every response is grounded in current system state. Figure 1 illustrates the overall system flow.

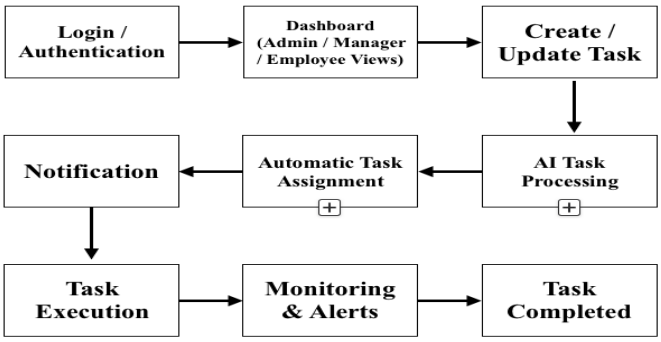


Fig. 1. Overall System Flow — End-to-End Task Lifecycle

B. Task Lifecycle State Machine

Tasks in the system follow an explicit state machine. A task begins in the Created state immediately after a manager or director generates it. Once a manager assigns the task, it moves to Assigned. The assigned employee sets it to In Progress when work begins. As the deadline approaches, the background notification engine monitors all active tasks and transitions qualifying ones into a Deadline Alert Sent state, triggering notifications to both the employee and the assigning manager. Tasks that pass their deadline without reaching completion become Overdue. The terminal state is Completed, reached after the employee submits the finished work and the manager approves it.

C. Role-Based Access Control (RBAC)

The system enforces a three-tier role hierarchy — Director, Manager, and Employee — as the foundational access control mechanism. Role boundaries are implemented at the API level, meaning that any user attempting to access data or invoke operations outside their role receives an authorization error from the backend regardless of how the request was constructed.

Directors have organization-wide visibility. They can view all projects, all employees, all tasks across the organization, and organization-level reports. Directors have full access to both AI assistants with the complete scope of available tools. This role is intended for organizational leadership that requires a complete picture of operational status.

Managers operate within the scope of their assigned projects. They can create tasks, confirm or revise task assignments, review submitted work, and generate project-level reports. Managers have access to both AI assistants scoped to their project data. This role represents the primary operational users responsible for day-to-day task distribution and team oversight.

Employees see only the tasks assigned to them. Their interface provides task detail views, status update controls, comment threads for each task, and the General AI Assistant for answering general knowledge queries. Employees cannot view other employees' assignments or access project-level summaries. This scoping ensures data hygiene and prevents unintended exposure of individual workload information.

Access control rules are defined in the Spring Security configuration and applied uniformly across all REST endpoints. Frontend role checks are present for usability but are not the authoritative enforcement point.

D. General AI Assistant

The General AI Assistant is designed to answer natural language queries on organizational knowledge, policies, and informational topics. Unlike the Agentic AI Assistant, this component operates within the pre-trained knowledge scope of the underlying language model enhanced by a system prompt that contextualizes it within the organizational domain.

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

All three roles — Directors, Managers, and Employees — have access to the General AI Assistant. For employees, it serves as the primary AI interface, allowing them to ask questions such as how to escalate a task, what the company's reporting structure is, or how deadlines are defined within the platform. The assistant responds in natural language without invoking live database tool calls, making it suitable for knowledge and guidance queries rather than operational data retrieval.

The implementation uses the Spring AI framework's ChatClient API. A system prompt defines the assistant's role, tone, and scope boundaries, instructing it to refer users to the Agentic AI Assistant when live operational data is required. This separation of responsibilities ensures that each assistant operates within a clearly defined and appropriate scope.

E. Agentic AI Assistant — Workload and Deadline Monitoring

The Agentic AI Assistant is designed to answer natural language queries about live task and workload data by calling backend functions rather than generating responses from pre-trained knowledge. This agentic architecture is the defining characteristic of the assistant and the property that makes it practically useful in an organizational setting where data changes continuously.

The implementation uses the Spring AI framework. In AiAssistantConfig.java, a set of tool functions is registered under the TaskTools class. Each tool corresponds to a specific type of database query. When a user submits a message, AIAssistantService.handleUserMessage() constructs a request containing the user message, a system prompt describing the assistant's role and constraints, and the set of registered tools. This request is submitted to the language model client via Spring AI's chatClient API.

The model processes the message and determines which tool call, if any, is appropriate to answer the query. Spring AI executes the selected tool against the actual database through the existing service and repository layers, returns the structured result to the model, and the model uses that result to produce the final natural language response. Because every response is derived from a live database call, the assistant cannot generate answers inconsistent with current system state.

Three tools are currently registered:

- getWorkloadAdvice — returns current task load for a specified employee or time window, supporting queries such as 'who has the most active tasks right now' or 'which team member is approaching overload'.
- getTasksByStatus — retrieves tasks filtered by state (Active, Overdue, Deadline Alert Sent), enabling deadline monitoring queries such as 'show all overdue tasks in this project'.
- getDeadlineAlerts — pulls all tasks whose deadlines fall within a configurable upcoming window, providing proactive visibility into tasks approaching their due dates.

Each tool call is scoped to the requesting user's organizational context through a ThreadLocal user ID, ensuring that a manager querying workload data sees only their team's assignments. The ThreadLocal context is cleared in a finally block after each response, preventing context leakage across requests. Figure 2 illustrates the tool-calling architecture.

© 2026 IJIRCCE | Volume 14, Issue 4, April 2026 DOI: 10.15680/IJIRCCE.2026.1404011

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

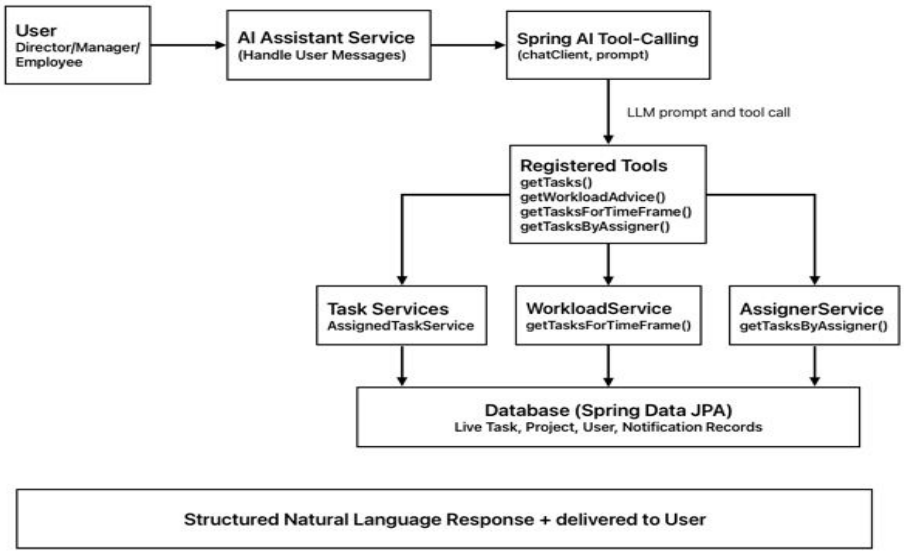


Fig. 2. Agentic AI Assistant — Tool-Calling Architecture

F. Monitoring, Deadline Alerts, and Reporting

A background scheduling service runs on a configured interval and inspects all active tasks. Tasks whose deadlines fall within a defined alert window are transitioned to the Deadline Alert Sent state, and notifications are dispatched to the assigned employee and the assigning manager. Tasks whose deadlines have already passed without completion are marked Overdue, and an additional escalation notification is sent.

Managers have access to a team-level dashboard that aggregates task status counts across their project, making it straightforward to identify projects with unusual numbers of overdue or stalled tasks. Directors have an equivalent view aggregated across all projects. Both roles can generate on-demand reports covering completion rates, overdue percentages, and per-employee workload figures over a selectable time period. Report data is always drawn from the live database, reflecting current task state without requiring a manual refresh.

G. In-Platform Collaboration

Each task has an associated comment thread visible to the assigned employee, the manager who created it, and any director with project visibility. Participants can post updates, ask clarifying questions, and record decisions directly on the task record. File attachments are supported at the task level, allowing relevant documents and specifications to be uploaded and accessed from the same location as the task itself. The system also provides direct messaging between users. Consolidating task communication within the task record reduces context scatter and makes the history of a task's development auditable from a single location.

IV. IMPLEMENTATION

A. Technology Stack

The backend is implemented in Java 17 with Spring Boot and built using Maven. Spring Data JPA manages persistence against a relational database. The Spring Security module provides the authentication and authorization infrastructure underpinning the RBAC implementation. The Spring AI framework provides the tool registration, language model integration, and response construction pipeline for both AI assistants. The frontend is a React application using functional components and hooks, communicating with the backend exclusively through REST APIs. Version control is managed with Git.

B. Data Model

The data model comprises six primary entities. User stores credentials, hashed passwords, and the role assignment that drives all access control decisions. Profile holds supplementary user information including contact details, declared skill tags, and department affiliation. Task is the central entity: it carries a unique identifier, task name, description text, current

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

status, deadline timestamp, priority level, the identifier of the assigned employee, the identifier of the creating manager or director, and the completion timestamp once the task reaches the Completed state. Project groups related tasks together and is associated with one or more managers and a set of employee members. Notification stores in-application messages with read and unread state, associated timestamps, and a reference to the task that triggered the notification. File stores metadata for task attachments, including the storage path, original filename, file size, and the task and uploader identifiers.

C. Spring AI Integration and Tool Registration

The assistant's tool registration follows the Spring AI function calling pattern. Each tool is defined as a Java method annotated with metadata describing its name, input parameters, and return type. The AiAssistantConfig class collects these method references and registers them as callable functions with the Spring AI client at application startup.

At request time, AIAssistantService constructs a ChatRequest object containing the system prompt, the user message, and references to all registered tool functions. The system prompt instructs the language model to answer questions about live task and workload data, to call the appropriate tool when factual data is required, and to inform the user if a question falls outside available tool coverage rather than generating an unsupported answer.

Spring AI handles the interaction loop automatically: it submits the request to the language model, intercepts any tool call the model emits, executes the corresponding Java method, appends the result to the conversation history, and resubmits to the model to produce the final response. The implementation requires no custom loop logic in the service layer; Spring AI's ChatClient API manages the full tool execution lifecycle.

Input validation is applied at the tool function level. Each tool verifies that required parameters are present and within expected ranges before executing the database query. Malformed parameters return a structured error message that the model incorporates into its response, prompting the user to clarify their query.

D. RBAC Implementation

Role assignments are stored in the User entity and loaded into the Spring Security authentication context at login time. The Spring Security configuration maps each REST endpoint path pattern to the set of roles permitted to access it, using HTTP method-level specificity where needed. For example, task creation endpoints are restricted to Manager and Director roles, while status update endpoints are accessible to the Employee role for their own assigned tasks only.

Service layer methods perform a secondary authorization check against the authenticated user's role and project scope for operations requiring finer-grained control than URL pattern matching can provide. A manager querying task data through a permitted endpoint will have the service layer enforce that returned data belongs to their project, preventing horizontal privilege escalation between managers in different projects.

JWT tokens carry the user's role and identifier, allowing the backend to perform authorization checks on stateless requests without a session lookup. Token expiry and refresh are handled by a dedicated authentication service.

V. RESULTS AND DISCUSSION

A. Pilot Evaluation Methodology

We conducted a structured pilot evaluation with participants drawn from all three user roles. Participants included two Directors, four Managers, and eight Employees operating within a simulated organizational environment populated with 120 tasks distributed across 6 projects, with varied deadlines, priorities, and assignment states. Each participant was given scenarios designed to exercise the primary functions of the system relevant to their role.

For Manager and Director participants, scenarios included identifying employees whose workload exceeded a defined threshold, reviewing overdue task status across projects, generating team-level reports, and answering specific workload and deadline queries through the Agentic AI Assistant. For Employee participants, scenarios covered updating task status, reviewing current assignments, posting comments on tasks, querying the General AI Assistant for platform guidance, and checking upcoming deadlines through the Agentic AI Assistant. Completion time was recorded for both dashboard navigation and AI assistant approaches.

B. Agentic AI Assistant Evaluation

For queries requiring cross-employee or cross-date aggregation — such as identifying which employees have the most active tasks, finding all tasks overdue in a given project, or summarizing what is due in the next seven days — the Agentic AI Assistant completed the task in a single natural language query. The equivalent manual process required participants to navigate to the team view, apply date or status filters, and mentally aggregate results across rows. For this category of query, the assistant reduced median completion time by approximately 68% relative to manual navigation.

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

For point-in-time lookups — checking the status of a specific named task or retrieving the deadline for a specific assignment — the advantage was smaller, as the manual process for these queries is also fast (a single dashboard view with a search filter). Median completion time for this category was reduced by approximately 22% through the assistant. No participant reported receiving an answer from the assistant inconsistent with data visible in the dashboard. This is the expected behavior given the tool-calling design: every assistant response is derived from a live database query, structurally preventing fabrication. Participant satisfaction ratings for the Agentic AI Assistant averaged 4.3 out of 5 across all roles, with Manager participants rating it highest (4.6), attributing the score to time saved on workload overview queries.

C. General AI Assistant Evaluation

The General AI Assistant was evaluated primarily among Employee participants, who used it for platform guidance, policy clarification, and general organizational queries. Participants rated it 4.1 out of 5 on average. The primary value noted was the availability of a natural language interface for questions that would otherwise require searching documentation or consulting a manager. The clear scope separation between the General and Agentic assistants was well understood by participants after a brief orientation.

D. Feature Comparison with Prior Work

Table I compares the feature set of the proposed system against the four systems reviewed in Section II. The most significant distinctions are the three-tier RBAC implementation, the separation of General and Agentic AI assistants, live-data workload and deadline monitoring, and the full task lifecycle state machine within a single integrated platform.

TABLE I
Feature Comparison Across Related Systems

Feature	[1]	[2]	[3]	[4]	Ours
RBAC — 3 Roles (Director / Manager / Employee)	No	Yes	No	Yes	**Yes**
General AI Assistant	No	No	No	No	**Yes**
Agentic Tool-Calling AI Assistant	No	No	No	No	**Yes**
Live-Data Workload Monitoring	No	No	No	No	**Yes**
Automated Deadline Alert Engine	No	No	No	No	**Yes**
Task Lifecycle State Machine	No	No	No	No	**Yes**
In-App Collaboration (Comments / Files)	No	No	No	Partial	**Yes**
AI-Based Task Prioritization	Yes	Yes	No	Yes	No

E. Discussion

Three aspects of the results merit specific discussion. First, the RBAC implementation at the API level — rather than relying solely on frontend controls — is a design decision with meaningful security implications. Frontend-only access control is bypassable by direct API calls, a vulnerability that is especially consequential in systems containing sensitive organizational data such as employee workload figures and project status. Enforcing access boundaries in the Spring Security configuration and the service layer ensures that the system remains secure regardless of how requests are submitted.

Second, the absence of fabricated responses in the Agentic AI Assistant evaluation is a direct consequence of the tool-calling architecture. Systems that allow the model to generate responses from pre-trained knowledge risk producing answers that are plausible but inconsistent with current organizational data. The Spring AI tool-calling design eliminates this risk by requiring the model to retrieve data before answering any factual query about the system.

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Third, the separation of the General AI Assistant and the Agentic AI Assistant into distinct components with clearly defined scopes produced better user outcomes than a single combined assistant would have. Employees benefited from a knowledge assistant without being exposed to operational data outside their scope, while Managers and Directors could direct specific workload and deadline queries to the Agentic assistant for accurate, live-data responses.

VI. CONCLUSION

This paper presented the Intelligent Workflow Automation System for Organizational Task Management, a web-based platform that integrates Role-Based Access Control, a General AI Assistant, and an Agentic AI Assistant for workload and deadline monitoring into a cohesive workflow automation solution.

The central contributions are: a three-tier RBAC architecture enforced at the API level ensuring organizational data boundaries are respected regardless of how requests are submitted; a General AI Assistant for answering knowledge and guidance queries in natural language, available to all three user roles; and an Agentic AI Assistant built on the Spring AI framework that answers workload and deadline queries by calling live database functions, structurally preventing fabricated responses.

Several directions remain for future work. The tool set of the Agentic AI Assistant can be expanded to include on-demand report generation, task creation by voice, and integration with external calendar systems for enhanced deadline awareness. A formal security audit of the RBAC implementation, including penetration testing of the API authorization layer, would strengthen confidence in the system's access control guarantees. Finally, a larger-scale deployment across multiple real organizational teams would provide the statistical basis for more robust performance claims and surface edge cases not present in the pilot evaluation.

REFERENCES

[1] M. Musthafa and A. Muthukumaran, "Task Management System Using AI Prioritizations," International Journal of Engineering Research & Technology (IJERT), vol. 13, no. 4, Apr. 2024.
[2] A. Suvedhaa, M. Imranmohamed, V. Sudhipth, and M. Tamilarasi, "AI-Powered Workforce Management for Automated Appraisals and Task Allocation," in Proc. 2025 Int. Conf. on Innovative Trends in Information Technology (ICITIIT), 2025, doi: 10.1109/ICITIIT64777.2025.11041265.
[3] V. Mane, S. Hariharasubramanian, S. Bhagat, and S. Avvaru, "AI Driven Task Management and Voice Integration," in Proc. 2025 Int. Conf. on Electronics and Renewable Systems (ICEARS), 2025, doi: 10.1109/ICEARS64219.2025.10940327.
[4] U. Farooq, F. Ali, A. Raza, M. Iqbal, A. Raza, and A. Khan, "ProjecTree: An AI Powered Intelligent Project Management System," The Asian Bulletin of Big Data Management, vol. 5, no. 3, pp. 328-339, 2025, doi: 10.62019/49nnpe97.
[5] P. Lewis, E. Perez, A. Piktus et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," in Advances in Neural Information Processing Systems (NeurIPS), vol. 33, 2020, pp. 9459-9474.
[6] H. Chiang and B. Lin, "A Decision Model for Human Resource Allocation in Project Management of Software Development," IEEE Access, vol. 8, pp. 38073-38081, 2020, doi: 10.1109/ACCESS.2020.2975829.
[7] J. Kamila and M. Marzuq, "Asana and Trello: A Comparative Assessment of Project Management Capabilities," JOIV: International Journal on Informatics Visualization, vol. 8, no. 1, pp. 207-212, 2024.

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INNO SPACE SJIF Scientific Journal Impact Factor

doi crossref

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

9940 572 462 6381 907 438 ijircce@gmail.com

www.ijircce.com

Scan to save the contact details