Adaptive AI Agents: Design and Implementation of a Multi-Agent Framework for Automation

Mandala Bhargavi[1], Tadikonda Ramya Sree[2], A. Thanusree[3], Dr. B. V. Ramana Murthy [4]
[1,2,3] *Department of Computer Science and Engineering Stanley College of Engineering & Technology for Women Hyderabad, India*
[4]*Vice Principal, Department of Computer Science and Engineering Stanley College of Engineering & Technology for Women Hyderabad, India*

Abstract-The increased need for intelligent automation and informed decision has emphasized the need for intelligent task orchestration and automation, which has traditionally required programming skills. To mitigate these issues, this project proposes a No-Code Multi-Agent Framework for Intelligent Task Orchestration and Automation. This framework provides an opportunity for users to create and implement intelligent multi-agent workflows without programming, using a YAML-based agent definition schema. It represents intelligent agents in terms of modular tasks, arranged in a directed acyclic graph, enabling sequential and parallel execution. It has integrated essential features, including task execution, embedding, vector databases, and logging, making it suitable for intelligent tasks like retrieval-augmented generation, multi-step automation, and knowledge-based tasks. It has also integrated features that enable parallel task execution, output merging, and scalability, making it suitable for cloud and real-time deployment. This framework bridges the gap between research models and practical automation, making it an adaptable, scalable, and user-friendly platform for intelligent automation and orchestration.

Keywords: No-code automation; Multi-agent systems; Task orchestration; Directed Acyclic Graph (DAG); YAML schema; Parallel task execution; Retrieval-Augmented Generation (RAG); AI workflow automation; Scalable architectures; Intelligent systems; Workflow automation framework.

I. INTRODUCTION

The recent rapid advancement of artificial intelligence (AI) technology has dramatically changed the digital automation landscape and has enabled machines to perform increasingly complex tasks without human intervention. Modern enterprises and research environments increasingly use AI-based systems for decision support, data processing, workflow management, and intelligent reasoning. With the increasing complexity of AI applications, the need for effective and efficient AI workflow systems that can manage multiple tasks and intelligent agents is also increasing. However, the development of AI workflows using conventional AI technology is heavily dependent on code and requires significant technical expertise in programming and various AI frameworks.

For these challenges to be met, this project proposes a No-Code Multi-Agent Framework for Intelligent Task Orchestration and Automation that is aimed at democratizing intelligent automation workflows. The proposed framework is based on the idea of abstracting away programming complexities through its intuitive and structured YAML-based configuration schema.

At the heart of the system is the Directed Acyclic Graph (DAG)-based orchestration engine that enables reliable execution of sequential, conditional, and parallel task execution. The use of the DAG data structure allows for the efficient execution of complex task pipelines. This provides the ability to execute sophisticated task pipelines that meet the needs of advanced use cases such as Retrieval Augmented Generation (RAG), multi-step reasoning pipelines, automated information extraction.

The project aims to bridge the gap between research-oriented multi-agent systems and practical applications. With the proposed framework providing the ability for users of both technical and non-technical expertise to develop intelligent automation pipelines without the necessity of writing any code,

the project contributes to the growing movement of accessible and adaptable AI. The proposed framework is a powerful and accessible platform that enhances the capabilities of modern automation systems.

About the project

The project "Adaptive AI Agents: Design and Implementation of a Multi-Agent Framework for Automation" primarily aims to develop a powerful and user-friendly system that facilitates intelligent automation of tasks without requiring programming skills. The system is based on the concept of adaptive AI agents that are capable of performing specific tasks, interacting with each other, and making decisions based on the environment. A simple YAML configuration system is used, which does not require any programming skills, allowing users to define agents, workflows, and execution strategies, which are further converted into a Directed Acyclic Graph (DAG) for sequential, conditional, and parallel task execution. This makes the system flexible, adaptable, and useful for real-world automation scenarios.

The system is integrated with essential components for task execution, embeddings, vector database, logging, and output handling, facilitating complex AI-based tasks like multi-step reasoning, information retrieval, and Retrieval-Augmented Generation (RAG). This project aims to bridge the gap between complex AI technologies and automation tools by providing a powerful and user-friendly system that can be used by organizations and users across various domains.

Objectives of project

To design a multi-agent framework that enables multiple AI agents to collaborate and perform complex automation tasks efficiently.

To implement a modular architecture where tasks are organized using a Directed Acyclic Graph (DAG) for sequential and parallel execution.

To integrate Large Language Models (LLMs) to allow agents to understand instructions, generate responses, and automate decision-making processes.

To develop a configurable system using YAML files so that users can define and manage AI agents without extensive coding.

To enable intelligent task orchestration where different agents coordinate and share information to complete workflows effectively.

To improve automation efficiency and scalability by allowing the system to handle multiple tasks simultaneously.

To build a user-friendly framework that simplifies the creation, deployment, and management of AI agents.

To demonstrate the effectiveness of adaptive AI agents in real-world automation scenarios such as data processing, information retrieval, and workflow management.

Algorithm

Input: User Request/Task Specification

Output: Automated Workflow Output

Step 1: Start the system.

Step 2: Get user input via the interface.

Step 3: Get agent and task configurations from the YAML file.

Step 4: Create a directed acyclic graph (DAG) that represents task dependencies.

Step 5: Initialize the workflow orchestrator that controls agent execution.

Step 6: Find tasks that have no dependencies and add them to the execution queue.

Step 7: Assign tasks to AI agents that match the tasks.

Step 8: Agents execute tasks using LLM, APIs, and internal tools.

Step 9: Store results and update task status.

Step 10: Execute dependent tasks after prerequisite tasks finish.

Step 11: Continue executing tasks until all tasks in the DAG are complete.

Step 12: Collect all output results from all AI agents.

Step 13: Return results to the user.

Step 14: Stop the system.

Figure 1 represents the system architecture of the Adaptive AI Agents: Design and Implementation of a Multi-Agent Framework for Automation.

User (Web Interface UI)

The process begins with the User Interface (UI).

Flask Server (API Backend)

The Flask server is the backend API server of the system.

YAML Agent Engine (services.py)

The YAML Agent Engine is the main component of

the system. The functions of the YAML Agent Engine are as follows: interpret YAML agent definitions ,create workflow pipelines, manage task execution, coordinate multiple AI agents this allows for the creation of agents without the need for coding.

Task Nodes (Node Helpers)
Task Nodes are the steps of the workflow.
This improves the efficiency of the workflow.

Knowledge Base (Vector Store)
The Knowledge Base stores the processed information as vector embeddings.

MongoDB (Document Database)
MongoDB is utilized in this system as a document database.

7. LLM Engine
The LLM engine provides the necessary intelligence in the AI agents.

8. Final Agent Output
The output is sent back through the Flask server and retrieved in the user interface.

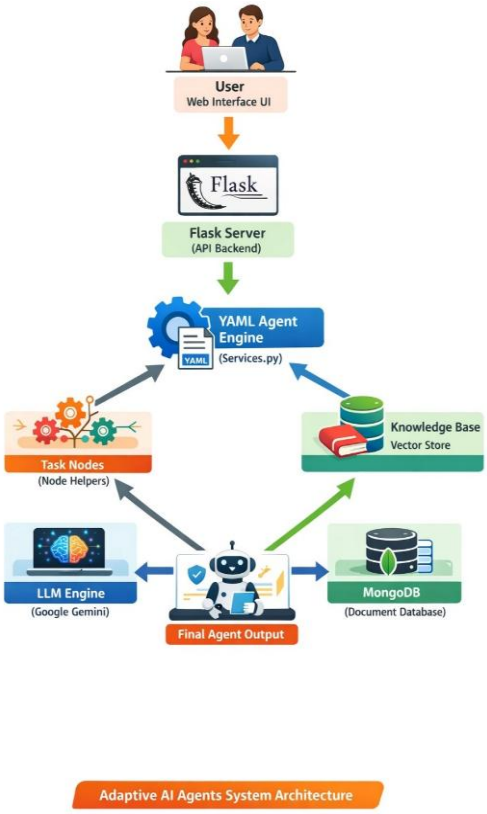


Figure 1. System Architecture

II. LITERATURE REVIEW

Recent advancements in the field of multi-agent systems and artificial intelligence have enabled the development of intelligent automation frameworks that can efficiently address complex distributed problems. In contrast, conventional single-agent systems are associated with various limitations and inefficiencies in terms of scalability, flexibility, and coordination when dealing with complex and large-scale tasks. In this context, a number of service composition models have been proposed using multiple agents to efficiently address these challenges. For instance, Seongsoo Cho et al. proposed a cloud-based service composition using a hybrid Integrated Particle-Ant Algorithm (IPAA), which is a combination of both Particle Swarm Optimization and Ant Colony Optimization. This approach is used to improve service selection and Quality of Service optimization in a cloud environment. However, various challenges and inefficiencies have been associated with the model in terms of re-optimization in a dynamic environment [1].

Some other researchers have also worked on the optimization of workload distribution and intelligent decision-making in distributed computing environments. Juan Li et al. proposed a multi-agent rollout optimization approach to optimize workload distribution in edge-cloud continuum. The proposed approach is mainly focused on optimizing resource allocation and reducing delays in cloud-native applications. In addition, Ahmed H. et al. proposed a multi-agent deep reinforcement learning (MADRL) model to effectively manage cloud-based digital twins in real-time. In this approach, agents can effectively learn optimal control policies for complex cloud infrastructures. However, these approaches are computationally complex and may impact system performance. These approaches are also computationally complex and may impact system performance in large-scale systems [2], [3].

Recent research has also emphasized the development of autonomy, reasoning, and collaboration in AI agents by utilizing LLMs. Lei Wang et al. have proposed a comprehensive survey on the development of autonomous agents utilizing LLMs, where the emphasis is given to the

architectures of the agents that utilize memory, planning, and reasoning mechanisms for intelligent agents. Xiang Li et al. have proposed a study on the development of collaborative multi-agent systems utilizing LLMs, including role-based and debate-based agent architectures for the development of intelligent agents. Such intelligent agents require high computational resources, leading to hallucinations and reliability problems during practical implementation [4], [7].

Another significant research area is adaptive and learning-based multi-agent systems that can function properly in a changing environment. Navid Nezamoddini discussed adaptive multi-agent networks for smart city environments, which apply rule-based, learning-based, or hybrid methods to adapt agent behavior according to changes in the environment. Zhiguo Ning et al. also explored various multi-agent reinforcement learning (MARL) methods, including centralized, decentralized, or hybrid coordination methods for intelligent distributed systems. These models also improve agent coordination and learning to a great extent but require significant computing power or struggle to apply these models inreal-time environments [5], [6].

Research has also been proposed to automatically generate agents to cooperate on tasks. Chen et al. proposed a framework called AutoAgents, which automatically generates agents to solve specific tasks using a two-stage process. The process involves a drafting stage and an execution stage. Another framework, AutoGen, was proposed by Wu et al. The framework is a conversational multi-agent framework. The agents cooperate to solve tasks by having a dialogue. The framework has been proven to have good collaboration and automation capabilities. However, it is considered to require a high level of coding skills

III. PROPOSED METHODOLOGY

Requirement Analysis & Problem Definition

The first step in the project is to identify automation needs, user constraints, and functional requirements. This includes analyzing existing multi-agent systems, user constraints on coding-based automation tools, and the requirement for a no-code configurable workflow system. Key scenarios such as RAG workflows, data processing, and multi-step decision processes are documented.

System Architecture Design

The next step in designing an automation tool is to create a high-level architecture. This includes agent modules, orchestration engine, YAML configuration parser, task execution manager, logging unit, and database/embedding support. Agents are conceptualized as functional units that can be used together to create workflows in the form of a directed acyclic graph. This architecture includes communication, execution, parallel task execution, and error handling.

YAML Schema Development for No-Code Agent Definition

A user-friendly YAML schema is developed that enables users to define agents, tasks, dependencies, inputs, outputs, and conditions. The YAML schema eliminates programming complexities by transforming YAML descriptions into executable workflows. The schema defines parameters for sequential as well as parallel execution paths.

Implementation of Multi-Agent Task Orchestration Engine

The orchestration engine is developed that reads YAML configurations and generates a DAG executed plan. The engine handles task scheduling, concurrent execution, passing messages between agents, propagating context, and aggregating output. The engine also incorporates retry logic along with fallbacks for fault tolerance.

Development of Core Agent Modules Various agent modules are developed that are modular in nature. These are:

o Task Execution Agent: API calls, computations

o RAG/Knowledge Agent: Embedding Retrieval & Context Building Agent

o Automation Agent: Workflow Triggers Agent, Monitoring Agent

Utility Agents

o Logging Agent

o Data Storage Agent

Integration of AI/ML Components Embedding models, LLM calls, and retrieval mechanisms are integrated to support intelligent decision-making. The system is optimized for real-time responses by caching embeddings and 11 enabling dynamic context generation during workflows.

Parallel Task Execution & Output Merging A concurrency layer is implemented to allow parallel

execution of agents. The framework merges outputs using defined rules and ensures synchronization before proceeding to dependent tasks.

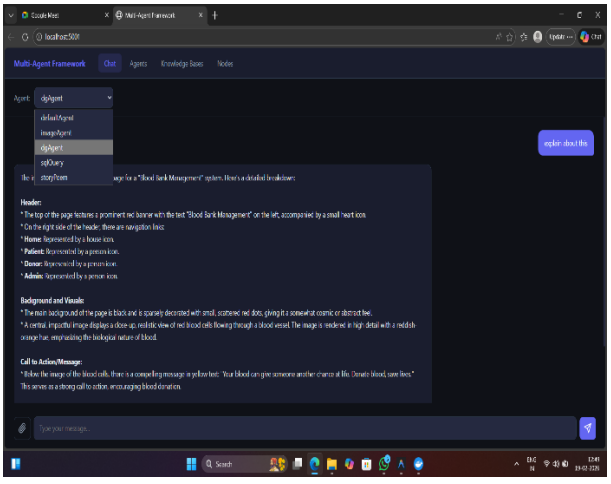
Testing, Validation, and Performance Evaluation Unit tests, workflow tests, and stress tests are conducted to verify reliability. Test cases include both simple and complex multi-agent pipelines. The framework's scalability, execution speed, fault recovery, and accuracy are evaluated against benchmark automation systems.

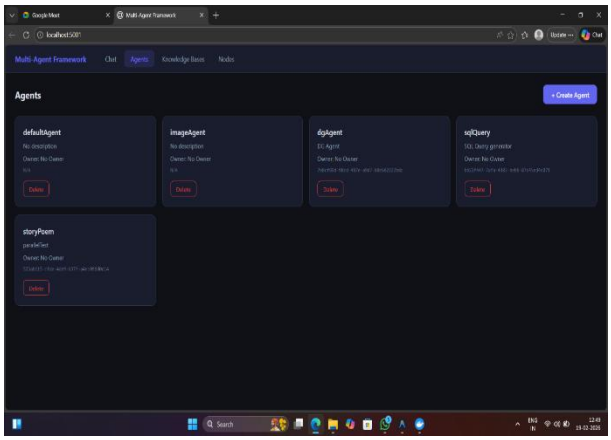
Deployment & User Interface Integration The system is deployed in a cloud environment to support distributed execution. Optionally, a simple UI/dashboard can be connected to allow users to upload YAML files, view workflow status, and monitor logs. Deployment ensures cross-platform accessibility and real-time process tracking.

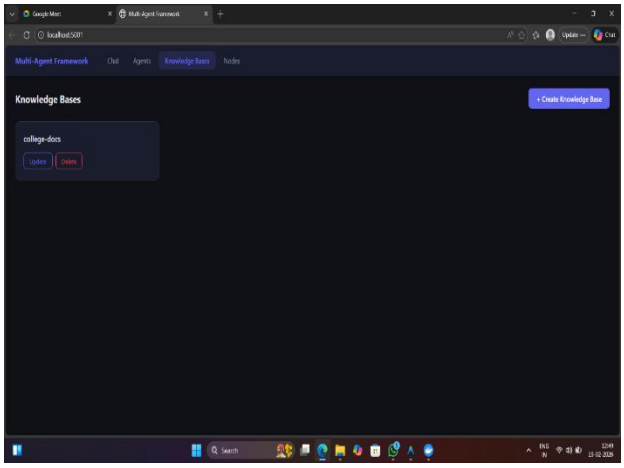
Documentation & Future Enhancements Complete technical documentation, user guide, and YAML reference manual are prepared. Recommendations for future improvements—like advanced agent types, adaptive learning, or auto generation of workflows—are documented.

RESULT

The proposed framework, named Adaptive AI Multi-Agent Framework for Automation, was successfully implemented and tested through a web interface. The framework combines various artificial intelligence agents, task nodes, and knowledge bases to automate processes and provide intelligent responses to user queries based on their input. The implementation shows the power of agent collaboration in solving complex automation problems.







System Interface and Agent Management The system provides a web interface through which users can interact with various artificial intelligence agents. As depicted in the figure below, various artificial intelligence agents such as defaultAgent, imageAgent, dgAgent, sqlQuery, storyPoem, etc., are available in the system. Each agent is created to perform a specific task, such as query handling, image processing, database queries, etc.

The integration of the knowledge base in the system facilitates Retrieval Augmented Generation (RAG), whereby the AI agents can retrieve documents to generate more accurate and context-aware output. This enhances the quality and reliability of the output generated by the agents.

Multi-Agent Chat Interaction : The framework also supports a chat interaction facility through which a user can interact with various agents. A user can select a specific agent from the dropdown list and send a query to the agent. For instance, the dgAgent

was used to produce an explanation of a webpage regarding a Blood Bank Management system.

Multi

The system processes the request and generates an explanation accordingly. This showcases the intelligent text generation and reasoning capability of the framework through the use of AI models. The chat interaction facility is a simple and convenient way of interacting with the agents.

The framework includes several task nodes that define the workflow execution process. The nodes implemented in the system include:

- retrieve_chunks
- select_prompt
- retrieve_uploaded_content
- call_llm
- parallel_execution
- merge

These nodes represent individual processing steps within the system. The nodes work together in a workflow structure where data is retrieved, processed, and passed to the language model for response generation. The parallel execution node allows tasks to be executed simultaneously, improving system efficiency and reducing response time.

Workflow Execution and Automation

The system successfully demonstrates automated workflow execution through the interaction of multiple agents and nodes. When a user submits a query, the following steps occur:

1. The user selects an agent and sends a request.
2. The system retrieves relevant information from the knowledge base.
3. Task nodes process the data and generate prompts.
4. The language model processes the request.
5. The system combines results and returns the final response to the user.

This workflow shows how the framework coordinates different components to perform complex automation tasks efficiently.

Performance and Observations

The experimental results show that the proposed system successfully integrates AI agents, knowledge bases, and workflow nodes into a unified automation framework. The system provides several advantages:

- Modular architecture for easy expansion
- Efficient task orchestration through node-based workflows
- Improved response quality using knowledge base retrieval
- User-friendly interface for interacting with AI agents

However, some limitations were also observed. The system performance may depend on the response time of the language model and database queries, and the addition of more agents or complex workflows may increase computational requirements.

IV. CONCLUSION

The project "Adaptive AI Agents: Design and Implementation of a Multi-Agent Framework for Automation" successfully demonstrates the potential of multiple AI agents working collaboratively to execute complex tasks in an efficient way. The proposed system employs the multi-agent concept in which every AI agent is assigned a specific task. This enhances intelligent automation and workflow management.

The proposed framework employs various tools and technologies, including Flask, MongoDB, YAML, and the Google Gemini LLM engine. The tools assist the proposed system in processing user requests and delivering intelligent responses. The proposed system also employs the Directed Acyclic Graph (DAG) concept and facilitates the communication of AI agents. This enhances the efficient execution of tasks and workflow management.

The proposed project demonstrates the potential of adaptive multi-agent systems in developing intelligent automation systems. The proposed framework can be employed in various domains, including data processing, information retrieval, and AI-based decision support systems.

The proposed project demonstrates the potential of adaptive multi-agent systems in developing intelligent automation systems. The proposed framework can be employed in various domains. The proposed project also demonstrates the potential of the multi-agent concept and the intelligent automation of tasks.

REFERENCES

[1] S. Cho, Y. Lee, and H. Choi, "Multi-Agent-Based Service Composition Using Integrated Particle-Ant Algorithm in the Cloud," Applied Sciences (MDPI), 2025.

[2] J. Li, M. Chen, et al., "Performance Optimization Across the Edge–Cloud Continuum: A Multi-Agent Rollout Approach for Cloud-Native Application Workload Placement," SN Computer Science, 2024.

[3] A. H. Ahmed et al., "Multi-Agent Deep Reinforcement Learning for Cloud-Based Digital Twins," Journal of Cloud Computing, 2024.

[4] Y. Liu et al., "No-Code AI Workflow Automation," ACM Conference Proceedings, 2023.

[5] N. Nezamoddini, "A Survey of Adaptive Multi-Agent Networks and Their Applications in Smart Cities," Sensors (MDPI), 2022.

[6] Z. Ning et al., "A Survey on Multi-Agent Reinforcement Learning: Foundations, Algorithms, and Applications," IEEE Transactions on Neural Networks and Learning Systems (TNNLS), 2024.

[7] X. Li et al., "A Survey on LLM-Based Multi-Agent Systems," arXiv, 2024.

[8] M. Torres et al., "AI-Based Workflow Automation with Real-Time Decision Making," IEEE/ACM Conference Proceedings, 2023.

[9] A. Gupta et al., "Multi-Agent Reinforcement Learning for Task Planning," IEEE Conference Proceedings, 2022.

[10] Q. Wu et al., "AutoGen: Enabling Next-Gen Multi-Agent Systems," arXiv / Microsoft Research, 2025