iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** IV **Month of publication:** April 2026

DOI: https://doi.org/10.22214/ijraset.2026.79343

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 IV Apr 2026- Available at www.ijraset.com

NexusMind: Agentic AI Orchestration Model

Mrs. P. Nandini[1], S. Pradeepthi[2], K. Rahul Babu[3], M. Ram Charan[4]
[1]*Assistant Professor, Department of Computer Science and Engineering, Methodist College of Engineering and Technology, Hyderabad, India*
[2, 3, 4]*B.E. Student, Department of Artificial Intelligence and Data Science, Methodist College of Engineering and Technology, Hyderabad, India*

Abstract: ***Agentic AI has considerably evolved from simple prompt-based language models that would respond to users based exclusively on the input provided. The modern versions of agentic systems are far more ambitious, in that they seek to independently plan tasks, reason out complicated contexts, use memory effectively, and even interact with external tools to execute multistage operations. These increased capabilities make this form of AI an attractive option for deployment in the real world, but each of the better- known frameworks, including the likes of AutoGPT, CrewAI, Puppeteer, and AgentGuard, continues to exhibit various issues that prevent seamless adoption. Common challenges include high computational cost due to repetitive LLM usage, limited scalability when handling large workflows, hallucination risks stemming from unverified outputs, and significant governance gaps that raise safety and compliance concerns. To overcome these limitations, this paper introduces NexusMind: a hybrid agentic AI orchestration model designed to achieve a balance between intelligent reasoning and the utilization of resources in an efficient manner. The proposed NexusMind unifies Large Language Models, Small Language Models, reflective memory, multi-agent collaboration, and policy-driven governance. By routing the tasks dynamically between LLMs and SLMs based on problem complexity and precision requirements, the model attempts to improve processing efficiency, lessen computational overhead, and preserve accuracy. An integrated governance layer that includes policy enforcement, role-based access control, audit trail generation, and risk evaluation ensures that the behavior of agents remains safe, transparent, and conforms to ethical and operational standards. Powered with such capabilities, NexusMind enhances the autonomy, trustworthiness, adaptability, and system reliability for applications in verticals like healthcare decision support, financial analysis, research automation, education technology, and enterprise-level intelligent software. The discussion presents the development of a functional prototype, the introduction of multimodal capabilities for broader perception, reinforcement-learning-based optimization as a means for continuous improvement, and real-time dynamic routing strategies toward making the architecture more flexible for real-world environments.***

Index Terms: ***Agentic AI, Large Language Models (LLM), Small Language Models (SLM), Hybrid Orchestration, Autonomous Agents, NexusMind.***

I. INTRODUCTION

Artificial Intelligence has come a long way from its early days, when most of the systems were built on fixed rules and deterministic logic. However, the rise of LLMs over the last ten years has dramatically altered how AI understands language, performs reasoning, and solves complex problems. These models can take up sophisticated queries and return highly contextual responses. Despite their intelligence, most current AI systems still behave reactively—they react to the prompts provided by the users and do not exhibit continuity, long- term planning, or reflection upon previously executed tasks. Agentic AI represents the next major step in that evolution: instead of waiting for instructions, agentic systems are designed to operate more like autonomous problem solvers. They are able to set goals, break large tasks down into manageable steps, coordinate with other agents, evaluate outcomes, and continuously refine their behavior. As might be expected, this shift of AI from reactive to proactive is making agentic models increasingly popular across a wide range of fields. Today, they are being used increasingly in research and literature review automation, financial forecasting, healthcare support and triage systems, data analytics, personalized education platforms, and enterprise process automation.

A. Overview of Agentic AI Orchestration

Agentic AI orchestration primarily concerns how multiple intelligent components-models, tools, and agents- can be orchestrated such that the system is autonomous in executing complex workflows. The AutoGPT, CrewAI, LangGraph, and Puppeteer frameworks are early attempts to construct this kind of orchestrated system. They allow tasks to be decomposed into subtasks, automate workflow management, interact with external tools or APIs, and enable multiple agents to collaborate on a single

objective. These frameworks still use LLMs as the foundation for most operations. While powerful, the use of LLMs is quite computationally expensive and adds delays for tasks that might be simpler in nature. Activities like summarizing small chunks of text, extracting keywords, or performing structured classification can easily be run by SLMs with greater efficiency. Since existing systems do not discriminate between such activities, they remain prone to unnecessary latency and increased cloud or infrastructure costs. Real-world adoption requires more than autonomy; indeed, it requires safety. Systems must have mechanisms for reducing hallucinations, enforcing policy constraints, maintaining data confidentiality, and preventing misuse in sensitive sectors such as healthcare or legal analysis. These are just some of the needs that reinforce hybrid orchestration, along with reliable governance, a key fundamental principle leading toward the architecture presented in this paper.

B. Need for the Study

Despite the advances of existing agentic frameworks, a number of key gaps still remain: Firstly, most systems cannot intelligently make decisions about which model would be appropriate to apply for any given task, leading to inefficient routing and poor resource utilization. Secondly, robust memory mechanisms- whereby the system is able to recall, reuse, and build upon previous interactions- are sorely lacking, thereby reducing continuity and personalization of agent behaviors. Thirdly, in the absence of any policy-driven governance structure, agents may return unsafe responses, violate constraints, or generally behave in an unpredictable manner without any mechanism by which to assess or log their decisions. These gaps become particularly problematic in industries where accuracy, trust, and compliance are paramount: for instance, AI systems used in health cannot afford to hallucinate clinical advice, AI in cybersecurity must follow protocol strictly, and legal or financial analysis tools must remain transparent and traceable. Addressing these challenges, this work introduces NexusMind, a governance-aware hybrid agentic orchestration model.

By integrating dynamic model routing, reflective memory, structured agent collaboration, and strong safety validation, NexusMind offers a more reliable and scalable way of achieving autonomy in AI. The main objective is the enhancement of efficiency, accuracy, adaptability, and safety to confidently deploy agentic systems in real-world environments where reliability cannot be compromised.

To address these challenges, this work introduces NexusMind, a hybrid agentic AI orchestration framework designed to improve both efficiency and reliability in autonomous systems. The proposed architecture combines Large Language Models and Small Language Models with a structured multi-agent workflow consisting of planner, executor, and reviewer agents. Tasks are dynamically routed based on their complexity so that lightweight operations are handled by efficient models while complex reasoning is processed by more powerful models. In addition, NexusMind integrates reflective memory and policy-based governance mechanisms to ensure transparency, safety, and compliance. By coordinating intelligent model selection, agent collaboration, and validation processes, the framework aims to provide a scalable solution for deploying agentic AI systems in real-world applications.

Furthermore, the NexusMind framework emphasizes scalability and adaptability for real-world environments. The architecture is designed in a modular manner so that additional agents, models, or external tools can be integrated without major system modifications. This flexibility allows the system to support diverse applications such as intelligent research assistance, enterprise workflow automation, and decision- support platforms. By combining efficient model utilization, structured task coordination, and governance-based validation, NexusMind provides a reliable foundation for building next- generation autonomous AI systems capable of operating safely and efficiently in complex environments.

II. EXISTING SYSTEM

The existing agentic AI frameworks that are being utilized today are AutoGPT, CrewAI, AgentGuard, Reflexion, LangChain, and Puppeteer. They are designed to incorporate concepts of autonomous reasoning, tool integration, iterative feedback, and multi-agent communication.

However, there are significant limitations to these frameworks that are being utilized today. They include excessive token consumption, poor performance, poor governance, insufficient hybrid model routing, insufficient memory reflection, and insufficient compliance-focused security.

AutoGPT's heavy reliance on iterative calls to the LLM causes excessive token consumption. CrewAI's structured agent functions are lacking enforcement for safety. AgentGuard provides better safety but does not utilize hybrid model decisions. Reflection provides more adaptive learning capabilities but is also expensive and computationally heavy. LangChain provides better construction of workflows but lacks governance capabilities.

Limitation	Affected Systems
No Hybrid LLM–SLM Model Routing	AutoGPT, LangChain, CrewAI
High Token Cost and Slow Performance	AutoGPT, Reflexion, CrewAI
No Governance or Policy Enforcement	AutoGPT, LangChain, Puppeteer
Limited Memory Reflection	AutoGPT, LangChain
No Risk Scoring or Compliance	CrewAI, Puppeteer, Reflexion
Domain-Restricted Security	HIPAA Framework

III. PROPOSED SYSTEM

The proposed system based on NexusMind provides a new way to structure agentic AI systems by utilizing a balanced blend of several model types and special tool agents. Instead of relying on any single type of model, NexusMind explicitly combines Large Language Models (LLMs), Small Language Models (SLMs), and a heterogeneous set of external tools. This model provides an advantage to the system in mapping tasks to the most appropriate processing components. This way, the entire process can be done in a more efficient, secure, and resourceful manner.

The proposed system is based on a task analysis and routing engine. When a task is input into the system, it evaluates the nature of the task based on several factors, including but not limited to reasoning depth, linguistic complexity, input structure, and potential risks. Routine tasks, including keyword extraction, data sorting, summary creation, and queries in a database, are intentionally mapped to SLMs. SLMs are more efficient in terms of processing because they can perform tasks quickly and utilize fewer resources. That is why they are more suitable for repetitive tasks with low complexity.

On the other hand, tasks that require deeper reasoning abilities, such as interpreting ambiguous instructions, long- form analysis, generating creative solutions, or understanding context-rich scenarios, are routed to the LLMs. They are more resource-intensive models; however, they are more appropriate for nuanced, more cognitively challenging tasks. This reduces unnecessary computation through this selective mechanism significantly.

In addition to the above-mentioned agents, tool-based agents are also integrated into NexusMind. They are not considered to be models; rather, they are units that can be utilized to carry out tasks such as:

1) Executing code snippets or debugging scripts
2) Searching large datasets or document repositories
3) Information extraction from structured data
4) Interacting with APIs
5) Mathematical/logical computation

The above-mentioned agents avoid unnecessary calls to the LLMs and speed up execution by offloading tasks to these agents when model execution is not required.

The framework of the system is the planner-executor-reviewer framework that ensures that the execution of every task is carried out through a structured pipeline rather than a single- pass execution.

It interprets the user request, breaks it down into steps, and executes each step through a model or tool depending upon the suitability. The actual execution of tasks is carried out through the Executor Agent.

The Reviewer Agent examines these outputs for correctness, coherence, policy adherence, and evidence of hallucination before accepting the response.

This multi-stage workflow emulates the way teams of human specialists collaborate, thereby improving reliability and quality control.

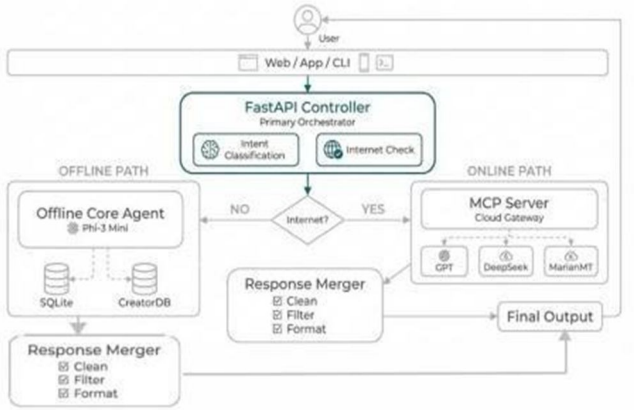
NexusMind also embeds a reflective memory component that makes it different from other prompt-based models.

It does not consider each query as an isolated event, but the system keeps the structured memories composed of relevant user contexts, choices, and task histories.

This enables agents to enjoy continuity across dialogue interactions, avoid repetitive computation, and generate behavior that is incrementally more personalized and context- sensitive. Long-term adaptations supported by the reflective memory allow performance improvement of the system in repeated usage.

NexusMind features a strong layer of governance and control to prevent autonomous behavior from becoming unsafe or unpredictable. This layer involves enforcing deep policy sets that include access control, safety constraints, and compliance; executing risk scoring of outputs; providing monitoring of critical data handling; and tracking decisions that have been made throughout the system via detailed audit trails. These audit trails track all actions taken by agents within the system, thus providing traceability and accountability, which are two of the most important features for applications within a regulated domain such as healthcare, finance, or cybersecurity. NexusMind uses dynamic model routing, reflective memory, multi-agent systems, and other techniques.

IV. SYSTEM ARCHITECTURE



The NexusMind architecture is designed to be a hybrid agentic AI system that combines several components to effectively process user queries. The system starts at the User Interface layer, where users interact with the system through web applications, mobile applications, or command line interfaces. User queries are passed to the FastAPI controller from the user interface. The FastAPI controller is the central orchestrator of the system. The FastAPI controller carries out initial processing of user queries, such as intent identification and connectivity checks.

After processing the request, the controller decides whether the system should run in offline mode or online mode. In the absence of internet connectivity, the request will follow the offline processing flow, which utilizes a Small Language Model like Phi-3 Mini to process the request. The offline agent will fetch the required information from local knowledge sources like SQLite databases, which will help the system respond without depending on the internet.

When an internet connection is available, the online process path will be enabled. In this case, the query would be sent to the Model Control Plane (MCP) server), which acts as an entry point to advanced cloud-based models (such as GPT) (and); furthermore, advanced cloud-based models (such as GPT and DeepSeek) are used for data processing and reasoning productivity for users with large/infinity-to-the-users number of complex queries. The response generated by the agents is sent to the response merging module to perform all cleansing, filtering, and formatting functions before sending the user the final response.

V. IMPLEMENTATION

NexusMind was built using modular architecture that allows for efficient interaction between components of the system. User Interface (UI), Back End Services, AI Agents, and Knowledge Retrieval Mechanism comprise the key modules that will work together to support users' requests; working together allows for an efficient workflow. The modules perform an appropriate function in the system and can work either individually or in combination to for continued scalability and maintainability of the overall NexusMind ecosystem.

A. Data Collection and Knowledge Sources

The primary source for the collection of data and the development of knowledge sources is through a combination of local and internet-based sources. The primary knowledge source being an artificial intelligence and machine learning textbook represents an offline structured dataset that enables the system to produce reliable results even if no connection to the internet exists.

The additional sources of information are connected to the NexusMind system via APIs allowing users access to external AI resources and large-scale globally available datasets; therefore, if a user makes a broad query, the system can communicate with external AI Models and produce additional answers based on the data within the provided external resources.

B. Data Preprocessing

Since the offline knowledge base is stored in PDF format, preprocessing is required to convert the content into a machine-readable format. The preprocessing pipeline includes text extraction, cleaning, and segmentation of the document into smaller text chunks. Each chunk is then transformed into a numerical embedding using a sentence-embedding model. These embeddings capture the semantic meaning of the text and are stored in a vector database. This process enables efficient similarity search when a user query is submitted.

C. User Interface and Backend Integration

We built the user interface with Next.js. It's pretty straightforward—people type in their questions and see the answers right on the web page. The frontend talks to the backend over HTTP. For the backend, we went with FastAPI. That's where all the heavy lifting happens: it grabs user queries, gets the different parts of the system working together, and sends back the final response.

D. Agent Routing Mechanism

Think of the agent routing component as the decision-maker for processing queries. It starts by checking if there's an internet connection. If it finds one, it sends the request to the online AI agent. If not, it moves things over to the offline knowledge retrieval system. It's a quick switch, all based on whether the network is up or down

.

E. Online and Offline Agents

In order to produce answers for complicated or general knowledge queries, the online agent communicates with external language models via APIs. Within the local vector database, the offline agent conducts a semantic search. It transforms the user query into an embedding vector, finds text segments that are similar, and then uses the information it has retrieved to create a response.

F. Memory and Response Processing

The system saves interaction history in a SQLite database to preserve conversational continuity. Context-aware interactions are made possible by this memory module, which logs user questions, answers, and timestamps. A response processing module cleans, filters, and formats the output before sending it to the user interface after the agents have generated a response.

VI. RESULTS

The NexusMind system was put into use and tested to see how well it could handle user inquiries using both offline knowledge retrieval techniques and online AI services. The findings show that the hybrid agentic architecture effectively combines several modules, such as the offline vector-based knowledge retrieval, online AI models, agent routing mechanism, backend API, and user interface.

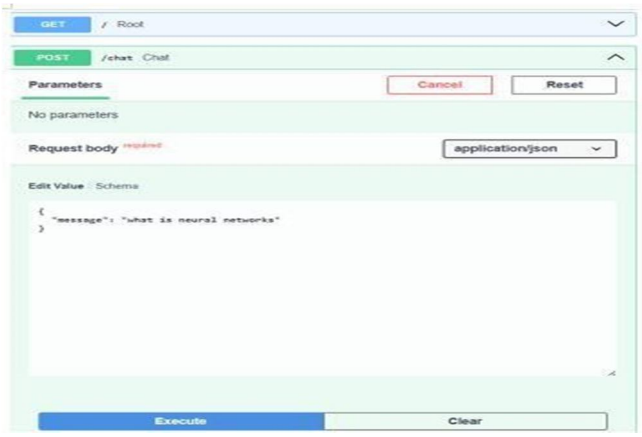


Fig. 1: sending of query message

Fig. 1 shows the steps in sending a query message to the API interface. The user sends a query by means of the request body, which is sent to the backend service by means of the specified API endpoint. The request is processed by the controller, which identifies the agent to be used in generating the response. When internet connectivity is available, the online processing mode is activated. In this mode, the query is sent to AI models by means of API communication.

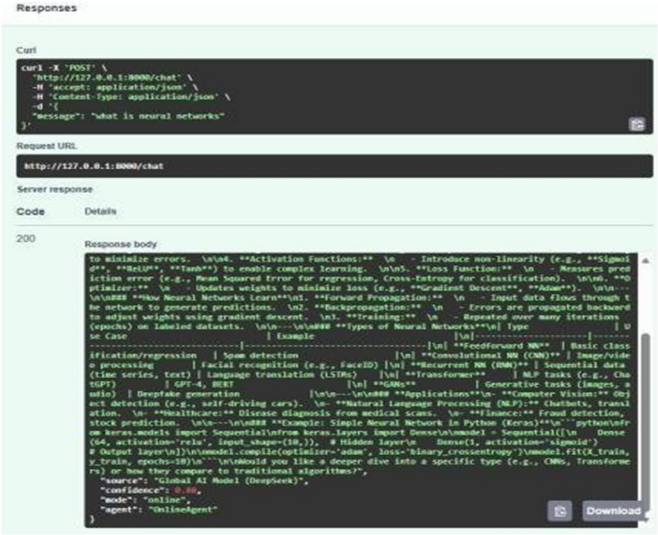


Fig. 2: response/answer online mode

As can be seen in Figure 2, in online mode, the system is able to fetch data from the connected AI service and return a data structure with the response and metadata.

The system also provides an offline processing capability to ensure that it can function continuously without access to the internet. In this case, the offline agent performs a semantic similarity search in the local vector database that was created based on the project data set. As can be seen in Figure 3, the system returns a data structure with the response and metadata.

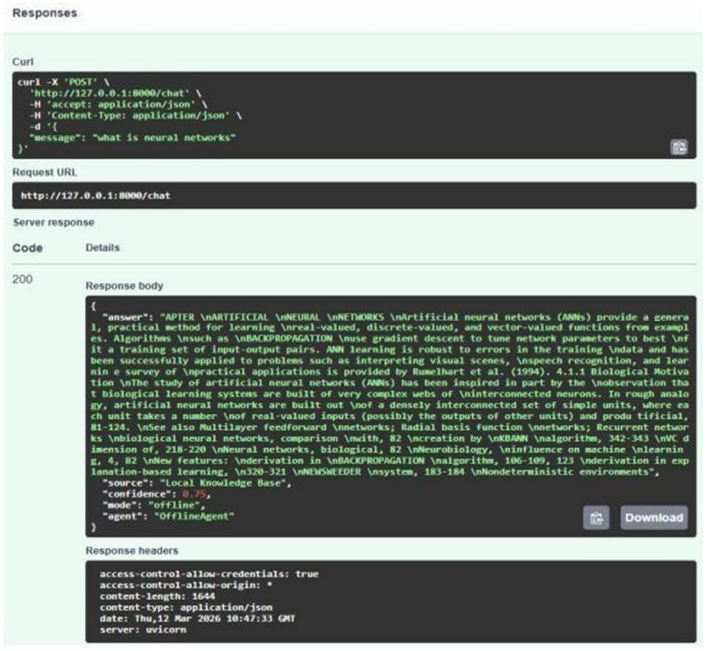


Fig. 3: response/answer offline mode

In other words, Fig. 3 shows that the system receives relevant knowledge from local storage of the embedding and produces a response without any use of external AI services

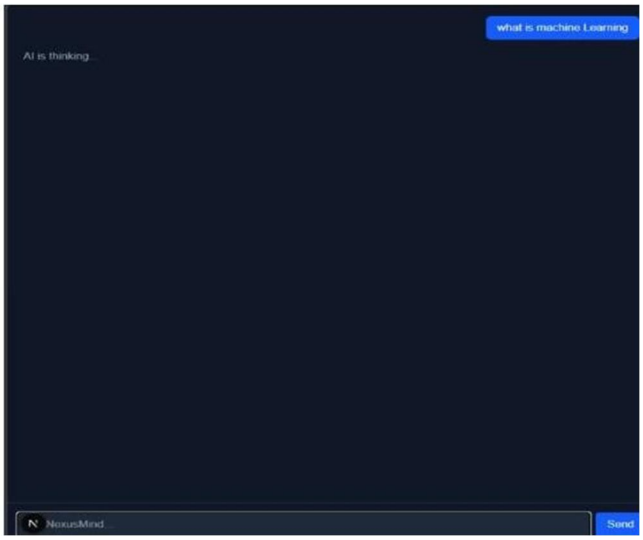


Fig. 4: Web Page

Finally, Fig. 4 shows the user interface that was designed to be used via the web for the NexusMind system. This interface offers a chat environment where users can type their questions and read the response produced by the system. This interface uses the API to enable smooth communication with the AI processing modules.

The experimental results demonstrate the effectiveness of the NexusMind framework in providing dynamic routing of agents, efficient knowledge retrieval, and response generation. The hybrid approach guarantees the system can perform in both online and offline settings.

VII. CONCLUSION

In this paper, NexusMind was introduced as a hybrid agentic AI orchestration framework. The system aims to promote efficiency in the development and utilization of intelligent systems. The system's architecture was also presented. This includes online AI models, offline knowledge retrieval methods, and a routing component that determines the best route to process queries based on system conditions. This system guarantees online and offline functionality and efficiency in handling queries.

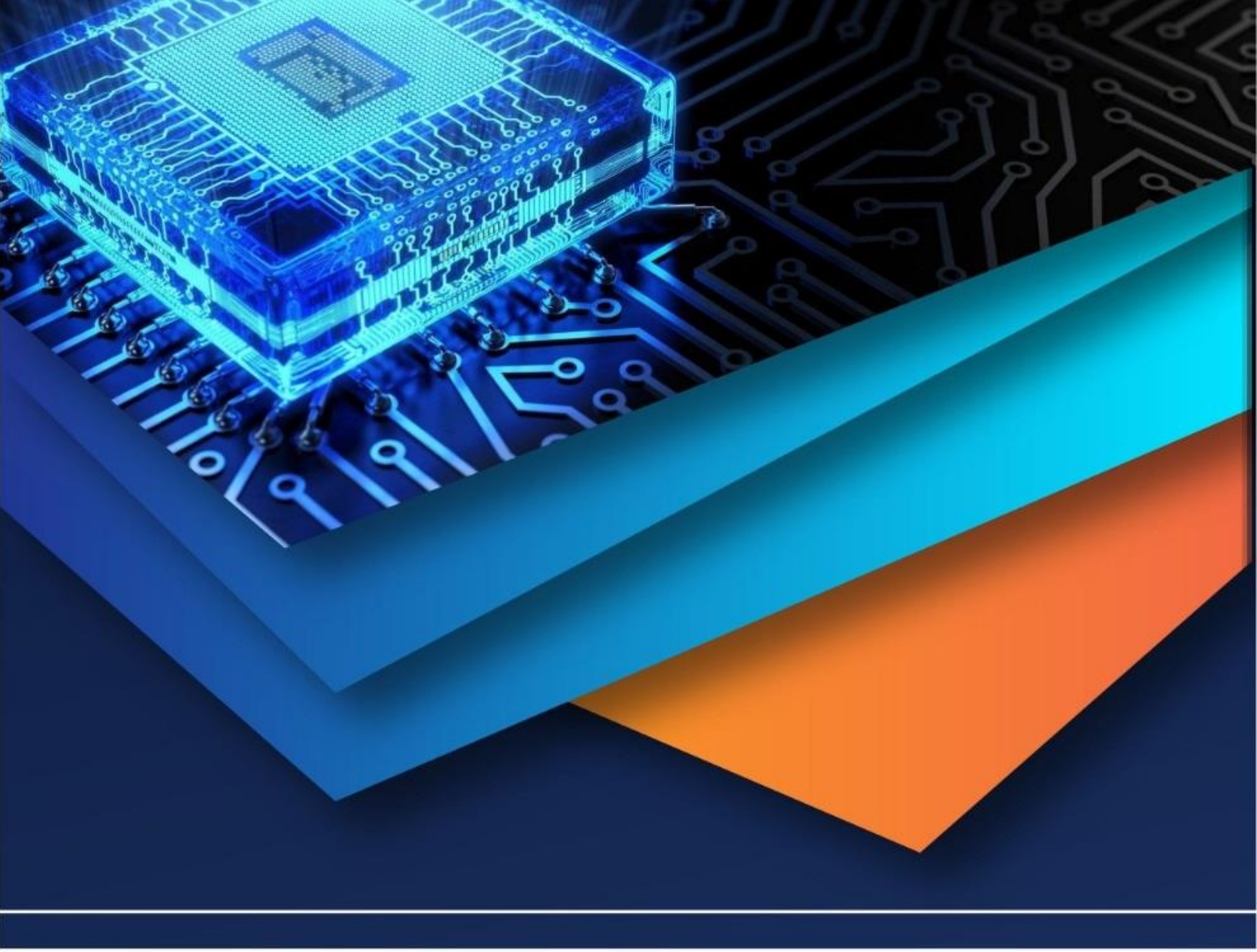
The system's implementation was also discussed. This includes how queries can be processed using a modular system composed of components such as the user interface, API, agent routing module, memory storage, and response processing. Experimental results show that the system can generate accurate responses using online AI services when the internet is available. Additionally, the system can also perform semantic knowledge retrieval using a vector database when the internet is unavailable.

In general, the NexusMind framework offers a scalable and flexible method for developing the agentic AI systems that can function in different environments with reliability, efficiency, and better utilization of resources.

Future work will focus on integrating real-time adaptive learning and improving multimodal capabilities to improve system performance.

REFERENCES

[1] T. Richards, "AutoGPT: An Experimental Open-Source Attempt to Make GPT-4 Fully Autonomous," GitHub Repository, 2023.
[2] A. Vaswani et al., "Attention Is All You Need," Advances in Neural Information Processing Systems (NeurIPS), vol. 30, 2017.
[3] Z. Chen et al., "AgentGuard: A Safety Framework for Autonomous AI Agents," arXiv preprint arXiv:2401.xxxxx, 2024.
[4] CrewAI, "CrewAI: Multi-Agent Collaboration Framework for AI Applications," GitHub Repository, 2024.
[5] Y. Ma et al., "Puppeteer: A Reinforcement Learning Framework for Agent Orchestration," Proceedings of the AAAI Conference on Artificial Intelligence, 2024.
[6] S. Gupta et al., "A HIPAA-Compliant Framework for Deploying Autonomous Medical Agents," Journal of Healthcare Informatics, 2023.
[7] N. Shinn et al., "Reflexion: Language Agents with Verbal Reinforcement Learning," Advances in Neural Information Processing Systems, 2023.
[8] H. Chase, "LangChain: Building Applications with Large Language Models," GitHub Repository, 2022.
[9] S. Mukherjee et al., "Orca: Progressive Learning from Complex Explanation Traces of GPT-4," Microsoft Research, 2023.
[10] M. Belcak et al., "Small Language Models Are the Future of Agentic AI," Journal of Artificial Intelligence Research, 2025.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY