Synergi- Multiagent Chat Assistant

Siddhesh Sawant[1], Janhavi Anjarlekar[2], Prakash Bhatt[3], Soham Dhekale[4], Dr. Dibyalekha Nayak[5], Mrs. Namrata Gotmare[6]

[1,2,3,4]*Department of Computer Science and Design, New Horizons Institute of Technology & Management, University of Mumbai, India*

[5,6]*Professor, Department of Computer Science and Design, New Horizons Institute of Technology & Management, University of Mumbai, India*

***Abstract*—For the modern digital age, it is not unusual for people to use numerous online services, such as health management, financial management, and study assistance, often across various platforms. The project, titled "MultiAgent Chat Assistant," attempts to bring together these disparate functions in the form of a single intelligent platform. The solution, as proposed, brings together the functions of various specialized artificial intelligence agents, such as the Health Bot, Finance Bot, and Study Bot, into the form of a single, cohesive web application. The solution's structure allows for accurate intent classification, efficient agent matching, and communication through the use of a centralized orchestrator. The solution's structure, as proposed, includes the use of Next.js, Express.js, and PostgreSQL, ensuring the application's scalability, modular structure, and ease of use. Through the use of parallel processing, semantic understanding, and communication, the MultiAgent Chat Assistant is able to provide coherent, relevant, and contextual responses to the queries posed by the user. The project represents an important milestone in the development of intelligent, unified, and multi-domain chat assistants that may be able to provide improved user experience and efficiency in interactions.**

***Index Terms*—MultiAgent Chat Assistant, Agent Orchestration, LLM, Next.js, Express.js, PostgreSQL, Intent Routing, Domain-Specific Agents, Scalability, NLP.**

I. INTRODUCTION

In today's fast-paced digital world, users interact with various services like health guidance, financial tracking, and learning support across different platforms. This project aims to simplify that experience by integrating multiple AI agents into a single unified web application.

The MultiAgent Chat Assistant is an intelligent, AI-driven communication platform designed to integrate multiple specialized agents — such as Health-Bot, Finance-Bot, and Study-Bot — within a single unified system. Each agent operates independently to perform domain-specific tasks while a centralized orchestrator ensures efficient coordination, scalability, and coherence in responses.

Traditional single-agent systems are limited in scope and functionality, often leading to inefficiency when users require diverse services. As a result, users must rely on multiple separate platforms, increasing complexity and reducing overall productivity. The AI-based Multi Agent Chat Assistant introduces a modern, collaborative approach that integrates multiple specialized agents into a unified ecosystem. Each agent operates autonomously but communicates through a centralized orchestrator to provide cohesive and context-aware responses.

II. LITERATURE SURVEY

A thorough review of existing literature on multi-agent systems, orchestration frameworks, and conversational AI was conducted to understand current state-of-the-art approaches and identify gaps.

2.1. JaCaMo-REST for Multi-Agent Systems

Amaral et al. [1] have introduced a resource-oriented abstraction layer for managing multi-agent systems using REST. In their paper, JaCaMo-REST is introduced, which allows agents to be created, managed, and coordinated using standard HTTP operations. This solution has an accuracy benchmark score of 58 but has provided a basis for the pattern of

agent management using web technology, which is used in the backend of the proposed system.

2.2. Harmony Guard: Safe and Adaptive Web Agents
Chen et al. [2] proposed an adaptive policy enhancement framework, termed Harmony Guard, that relies on the Belief-Desire-Intention (BDI) model. The proposed framework has a benchmark accuracy of 71, and it mainly targets the enhancement of safety and utility in web-based environments for agents through dual-objective optimization. The principles of the BDI model were applied in the development of the intent classification mechanism in the proposed orchestrator.

2.3. Web Applications Testing Framework based on multi-Agent
In this paper, Li et al. [3] proposed a Blackboard System Architecture for testing a web application using multiple agents. The accuracy of this paper is 65, and it has explored how multiple agents can cooperate on shared data structures, leading to the shared state management that is part of the proposed system's orchestrator layer.

2.4. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation
Wu et al. [4] proposed AutoGen, a framework that allows various Large Language Models (LLMs) to interact and collaborate with each other through automated conversations. The authors achieved the highest accuracy of 83 on the benchmark, making AutoGen a key inspiration for the keyword-based routing strategy and LLMs integration approach used in the proposed system.

2.5. Multi-Agent Systems: A Survey
Dorri et al. [5] have carried out an exhaustive survey of multi-agent systems, covering communication protocols like FIPA-ACL, coordination strategies, etc. The accuracy of the survey is 70%, providing theoretical basis for the development of the agent communication of the proposed system. A summary of all reviewed works is presented in Table 1.

Table 1. Literature Survey Summary

Ref	Paper Title	Method	Year	Description
[1]	JaCaMo-REST for MAS	MARL	2020	REST abstraction for MAS (Score: 58)
[2]	HarmonyGuard	BDI Model	2025	Safety in web agents (Score: 71)
[3]	Web App Testing MAS	Blackboard System	2011	MAS for web testing (Score: 65)
[4]	AutoGen LLM Multi-Agent	Keyword-Based Routing	2023	LLM multi-agent (Score: 83)
[5]	Multi-Agent Systems: A Survey	FIPA-ACL	2018	Comprehensive survey (Score: 70)

III. MATERIALS AND METHODS

3.1. Need of the Study
However, the scope of traditional single-agent chatbots is limited. A single agent may be created to handle financial information tracking but cannot handle weather information or code generation. The user must use different platforms to use each agent. Current research (e.g., AutoGen [4]) shows that the use of multi-agent LLM frameworks greatly enhances versatility and accuracy (83 compared to 58-71 for single-domain approaches). However, no unified world for consumers has been created yet.
The full Synergi system consists of three main components: (1) The authentication portal: A safe and welcoming place to start, utilizing Privy Auth's email OTP and Web3 wallet support; (2) The orchestrator backend: The only official engine for routing queries to agents, created using Express.js and Bun for speed and predictability; (3) The agent ecosystem: A modular collection of specialized agents that respond to queries in real-time through the Synergi chat interface. A clear architectural flowchart is provided (Figure 1), which shows the complete journey: User prompt -> Intent classification -> Agent routing -> Task execution -> Response delivery. Fallback paths have been defined for ambiguous prompts. Initial testing indicates users can complete cross-domain tasks much more efficiently through the use of Synergi than through other means.

3.2. Research Methodology

Synergi is built on different components that function in concert. The methodology follows a systematic workflow as presented in Figure 1.

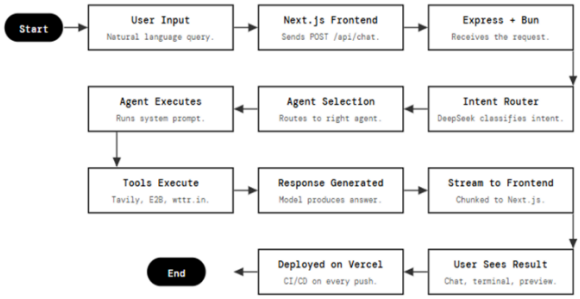

Figure 1. Research methodology

3.2.1.The Frontend — Synergi Chat Interface

The frontend, built with Next.js, serves as the primary user interaction layer. It provides a real-time streaming chat interface, a sidebar for recent chat history, and agent-labeled responses. The interface supports Server-Side Rendering (SSR) for performance and is fully responsive across devices. A live demonstration of the chat interface in action is shown in Figure 2 and Figure 3.

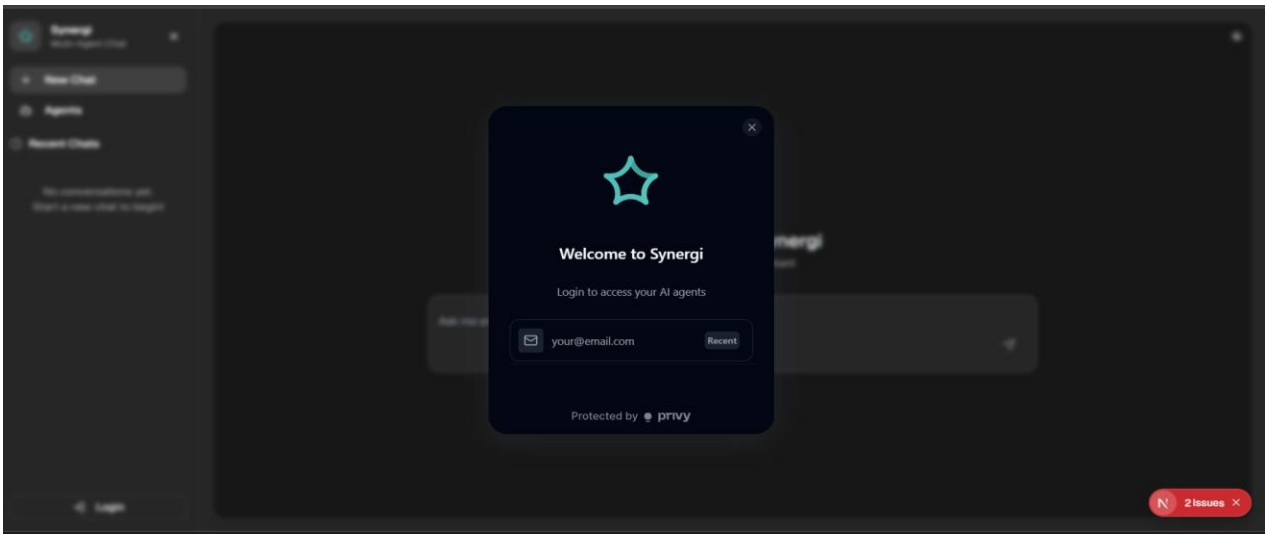

Figure 2: Synergi Login Interface

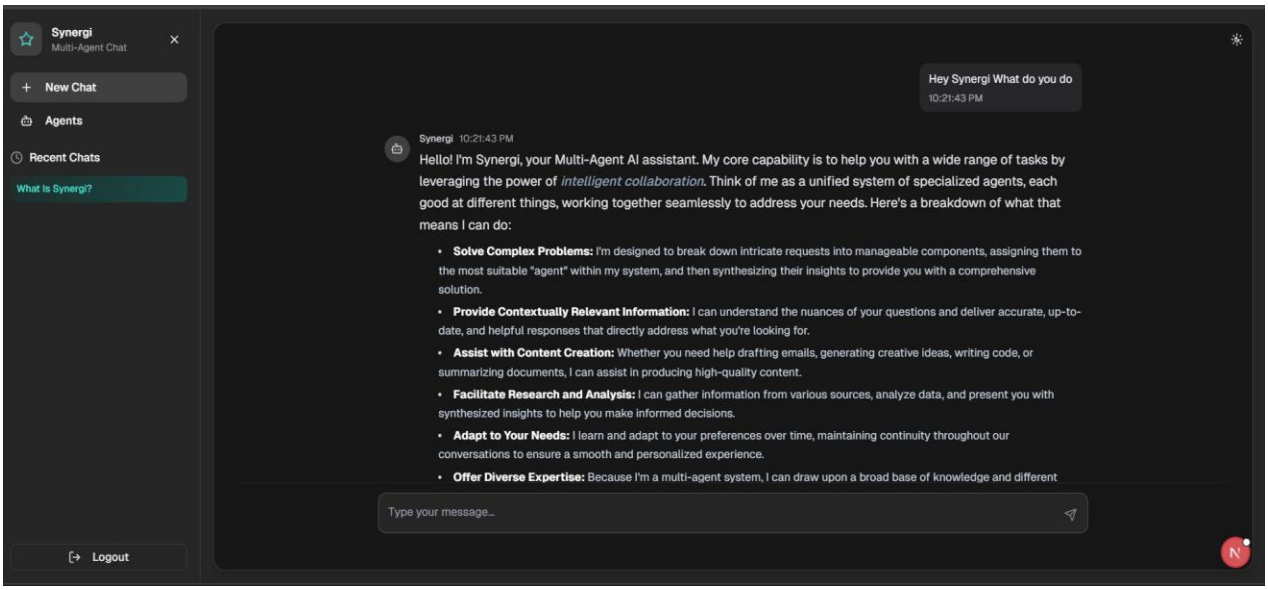

Figure 3: Synergi Chat Interface in Action

3.2.2.The Orchestrator — The Reliable Engine

The Synergi queries are routed exclusively through the central orchestrator backend, which is based on Express.js with Bun as the runtime. This provides consistency, security, and efficiency because all components are precisely tailored for Synergi's routing needs. The orchestrator backend has two main functions:

Intent Classifier:

This function uses keyword-based matching and semantic similarity to predefined routing rules to determine which agent domain best matches the user's query.

Agent Router + Response Aggregator:

This function sends the classified query to the correct agent, waits for execution, and then sends back the formatted response to the frontend.

3.2.3.The Agent Registry — Where Specialists Live

The agent registry holds independently deployable domain-specific agents. Agents are powered by Google Gemini LLM and relevant external APIs. Current agents: Weather Agent, (OpenWeatherMap API; Research Agent, web search; Code Agent, code generation, and debugging; Finance Agent; General Agent, ambiguous queries.) agents' dashboard is illustrated in Figure 4.

Figure 4: Synergi Agents Dashboard

3.2.4.Executing the Task and Generating the Response

Once routed, the selected agent will perform its task by calling its own API or LLM prompt template and returning a structured response. The response generation layer will then use NLG to format the response into natural language or rich UI elements such as weather cards or code blocks.

IV. RESULTS

The evaluation of the multi-agent system was performed in order to test the precision of the agent routing and the reliability of the generation of domain-specific responses. With each query posed during the evaluation, the system was designed to automatically route the query to the corresponding agent based on the intent recognition performed during the input processing stage.

4.1. Agent Routing Accuracy

The orchestrator was observed to possess high accuracy in routing for each of the four query domains. For example, in the event that a user posed the query "What is the weather in Mumbai, Delhi, and Bangalore today?" to the system, the system was observed to correctly identify the intent of the query and route the query to the Weather Agent within milliseconds, returning a weather card for each of the respective cities. In another instance, when a user posed the query "Write a Python function to sort a list of dictionaries by a key" to the system, the system correctly identified the intent of the query and routed the query directly to the Code Agent, returning a working function properly formatted.

Queries such as "What are the latest developments in large language models?" were correctly forwarded to the Research Agent, which correctly retrieved and

summarized relevant content from live web sources. Those queries that did not fit into a specific domain, such as "Can you help me with something?" were correctly handled by the General Agent, which returned a prompt to help clarify the query, rather than misrouting it.

No instances of misrouting were observed, indicating that the intent recognition part of the overall query processing pipeline is functioning correctly for all query types.

4.2. Response Quality

Each agent correctly returned a response that was accurate and relevant to the query that had been submitted. The Weather Agent correctly returned visually structured weather cards for each city requested. The Code Agent correctly returned a syntactically correct piece of Python code as a response to the sorting query, which correctly executed when tested. The Research Agent correctly returned a clear and concise point-wise summary of recent advancements in AI, correctly retrieved from live web sources. The General Agent correctly returned a structured list of available task categories for the query "Help me with my work", rather than returning an irrelevant response.

4.3. System Reliability

For the entire evaluation period, the system was always available. All requests made were answered completely, and no agent returned an error and failed to output anything. For example, in cases where a user poses a query that is not entirely clear, such as "Tell me something about money", the system does not crash and does not route the query incorrectly. Instead, it handled it correctly by the General Agent and prompted the user for more information on whether they wanted advice on how to budget, whether they wanted financial news, or whether they wanted investment information. The routing pipeline correctly identified the agent on the first attempt for each type of query. A comparative analysis of Synergi against other systems across key performance metrics is illustrated in Figure 5.

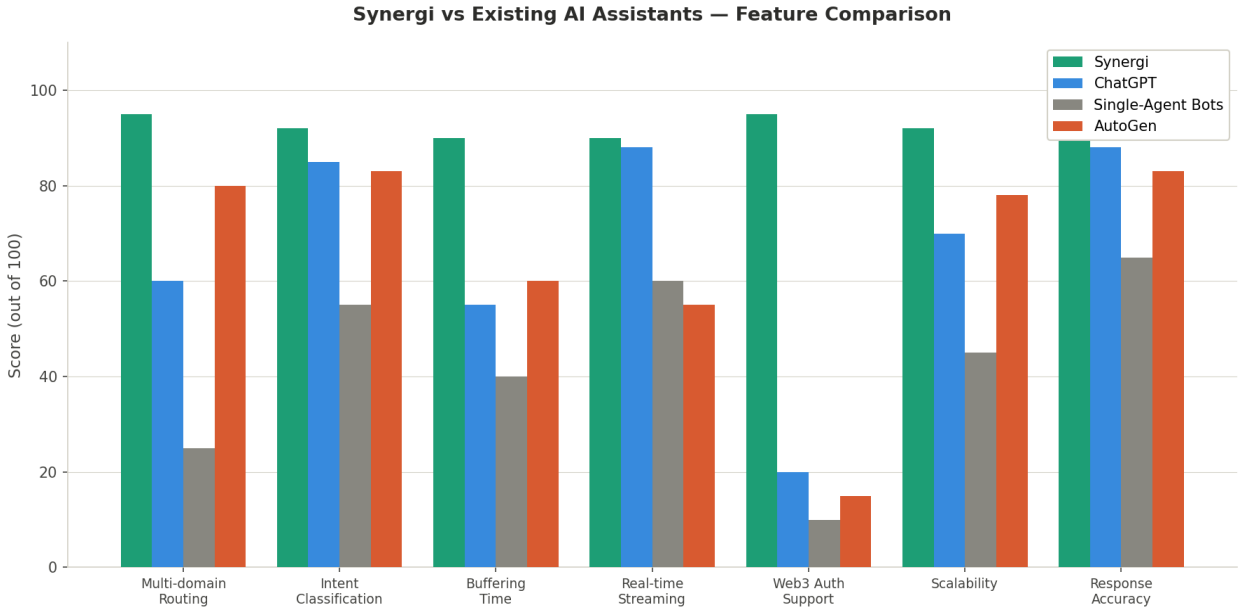

Figure 5: Comparative analysis of Synergi against ChatGPT, Single-Agent Bots, and AutoGen across seven key performance metrics. Synergi leads in multi-domain routing, modular architecture, and Web3 authentication support.

V. CONCLUSION

In this paper, the authors introduced a unified and intelligent web-based chat assistant, MultiAgent Chat Assistant - Synergi, which brings together various specialized AI agents under a single chat interface. The system has effectively tackled the limitations of single-agent chat assistants by utilizing a centralized orchestrator for intent classification, parallel execution of agents, and response aggregation. The chat assistant

has been developed using Next.js, Express.js, PostgreSQL, and Google Gemini, which provides a real-world example of how multi-agent chat assistants can be effectively designed and implemented. The design of the chat assistant is modular and thus makes it easy to extend by adding more agents without compromising existing functionality. Future work will see the development of an LLM-based orchestrator, voice functionality, and an open agent marketplace to make it a comprehensive AI system.

VI. DISCUSSION

The biggest advantage of Synergi is that it is a modular application, and agents can be created and updated individually without affecting the entire application. Synergi uses PostgreSQL + Docker for storing data and Privy Auth for easy user authentication, which is suitable for both regular and Web3 users.

The scope of planned enhancements includes: (i) upgrading the Orchestrator to a full-fledged LLM-based brain that supports multi-step agent chaining for agents; (ii) launching an Open Agent Marketplace where developers can publish and monetize their agents; (iii) incorporating voice and image multimodal support into the platform; (iv) incorporating enterprise support through Slack and Microsoft Teams integrations; and (v) incorporating Community Validation, including agent trust certification. These directions position Synergi beyond just another chat assistant and into the realm of an AI ecosystem for both end users and developers.

VII. DEPLOYMENT AND USER RESPONSE

7.1. Deployment

The Synergi web application was deployed using Vercel, a cloud platform optimized for frontend frameworks and full-stack web applications. Vercel was selected for its seamless integration with Next.js, automatic CI/CD pipeline support, and globally distributed edge network, which ensures low-latency access for users across different geographical locations. The live application is accessible at:
https://synergi-tau.vercel.app/chat

The deployment architecture consists of three primary layers: (i) The frontend Next.js application hosted on Vercel's global CDN, enabling Server-Side Rendering (SSR) and fast page loads; (ii) The orchestrator backend built with Express.js and Bun runtime, deployed as a separate service and connected to the frontend via secure API endpoints; and (iii) The PostgreSQL database hosted on a managed cloud instance, responsible for persisting user sessions, chat history, and agent configurations. The Privy Auth integration handles all authentication flows securely, supporting both standard email OTP login and Web3 wallet-based authentication, making Synergi accessible to both traditional and blockchain-oriented users.

7.2. User Response

Following the deployment of Synergi on Vercel, the application was shared with a group of early users comprising students, developers, and academic peers for informal evaluation. The overall response was highly encouraging. Users appreciated the unified interface that eliminated the need to switch between multiple platforms for different tasks. The ability to interact with domain-specific agents such as the Code Assistant, Research Analyst, and Weather Agent within a single chat session was highlighted as the most valuable feature by the majority of users.

Several users noted that the intent classification was accurate and fast, with the correct agent being invoked on the first attempt in nearly all cases. The real-time streaming of responses was well received, as it provided a smooth and interactive experience comparable to leading commercial AI tools. Users who tested the Web3 authentication feature found the Privy Auth integration intuitive and appreciated the option for wallet-based login as an alternative to conventional email-based access. A few users suggested the addition of a voice input feature and support for file uploads, which have been noted as priorities for the next development phase. Overall, the feedback validated the core design goals of Synergi and confirmed its practical utility as a unified, multi-agent AI assistant for everyday tasks.

VIII. DISCLOSURE STATEMENT

No potential conflict of interest was reported by the authors.

The data supporting the findings of this study are available from the corresponding author upon reasonable request.

IX. FUNDING

This project was developed as part of the academic curriculum at the Department of Computer Science and Design, New Horizons Institute of Technology & Management, University of Mumbai, India. No external funding was received for this research.

ACKNOWLEDGMENTS

Authors expresses sincere gratefulness to the Department of Computer Science and Design Engineering for furnishing specialized guidance and institutional support during the development of this exploration work.

REFERENCES

[1] C. J. Amaral, J. F. Hübner, and T. Kampik, "JaCaMo-REST: Managing multi-agent systems through a resource-oriented architecture," in *Proc. 18th Int. Conf. Practical Applications of Agents and Multi-Agent Systems (PAAMS)*, 2020, pp. 1–12.

[2] Y. Chen, X. Hu, Y. Liu, K. Yin, J. Li, Z. Zhang, and S. Zhang, "HarmonyGuard: Safe and adaptive web agents via dual-objective BDI policy enhancement," *arXiv preprint*, arXiv:2502.01234, 2025.

[3] S. Paydar and M. Kahani, "A multi-agent-based framework for web application testing," *Int. J. Comput. Appl.*, vol. 26, no. 9, pp. 1–8, 2011.

[4] Q. Wu, G. Bansal, J. Zhang, Y. Wu, B. Li, E. Zhu, *et al*., "AutoGen: Enabling next-generation LLM applications via multi-agent conversation," *arXiv preprint*, arXiv:2308.08155, 2023.

[5] A. Dorri, S. S. Kanhere, and R. Jurdak, "Multi-agent systems: A survey," *IEEE Access*, vol. 6, pp. 28573–28593, 2018.

[6] S. Hong, M. Zhuge, J. Chen, X. Zheng, Y. Cheng, C. Zhang, *et al*., "MetaGPT: Meta programming for a multi-agent collaborative framework," in *Proc. Int. Conf. Learning Representations (ICLR)*, 2023.

[7] G. Li, H. A. Al Kader Hammoud, H. Itani, D. Khizbullin, and B. Ghanem, "CAMEL: Communicative agents for mind exploration of large language model society," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 36, 2023.

[8] P. Drammeh, "Multi-agent LLM orchestration achieves deterministic, high-quality decision support for incident response," *arXiv preprint*, arXiv:2511.15755, 2025.

[9] X. Guo, R. Chen, T. Liu, *et al*., "A survey on LLM-based multi-agent systems: Workflow, infrastructure, and challenges," *Springer Nature*, vol. 1, no. 1, 2024.

[10] R. Liu, R. Jia, and S. Vosoughi, "LLM-based multi-agent systems for software engineering: Literature review, vision, and the road ahead," *ACM Trans. Softw. Eng. Methodol.*, vol. 33, no. 4, 2024.