Indonesian Journal of Data and Science

Volume 7 Issue 2
ISSN 2715-9936
https://doi.org/10.56705/ijodas.v7i2.312

Research Article

An LLM-Based AI Task Agent for Academic Task Management with n8n and Telegram

Nabila Widiyanti [1]; Tasrif Hasanuddin[2]; Huzain Azis [3,*]
[1] Universitas Muslim Indonesia, Makassar, Indonesia, nwidiyanti321@gmail.com
[2] Universitas Muslim Indonesia, Makassar, Indonesia, tasrif.hasanuddin@umi.ac.id
[3] Universitas Muslim Indonesia, Makassar, Indonesia, huzain.azis@umi.ac.id

Correspondence should be addressed to Huzain Azis; huzain.azis@umi.ac.id

Received 10 May 2026; Accepted 30 July 2026; Published 31 July 2026

Abstract:
Introduction: Managing multiple academic tasks with overlapping deadlines remains challenging for university students, while conventional task-management applications still require substantial manual organization and prioritization. This study develops an LLM-based AI Task Agent that enables conversational academic task management through Telegram and workflow automation. **Method**: The proposed system integrates Telegram as the interaction interface, n8n for workflow orchestration, an LLM-based AI agent for natural-language interpretation and tool selection, Google Sheets for task-data operations, PostgreSQL for conversational memory, and scheduled workflows for automated reminders. The system supports Create, Read, Update, and Delete operations, contextual priority recommendations based on deadline, urgency, and lecturer strictness, and proactive reminders. Functional performance and response time were evaluated across the primary system functions. **Results and Discussion**: Create, Read, Update, and Delete operations achieved 100% functional accuracy, while priority recommendation and automated reminder functions achieved 95%. Recorded processing times ranged from 3.1 to 3.5 seconds, with an average of approximately 3.32 seconds. The results demonstrate that separating LLM-based interpretation from predefined external tool execution enables reliable conversational task management while maintaining controlled data operations. **Conclusion**: The proposed LLM-based AI Task Agent demonstrates the feasibility of integrating conversational interaction, executable task-management functions, contextual prioritization, memory, and proactive reminders within a unified Telegram-based academic workflow.

Keywords: SQL Injection; Deep Learning; Long Short-Term Memory; Convolutional Neural Network; Embedding.

Dataset link: https://www.kaggle.com/datasets/syedsaqlainhussain/sql-injection-dataset

1. Introduction

Managing multiple academic tasks is a common challenge for university students because assignments, projects, examinations, and other academic activities often have overlapping schedules and different levels of urgency. Conventional task-management applications generally provide features for recording tasks, setting deadlines, managing calendars, and generating reminders. However, users are still required to manually organize task information, monitor approaching deadlines, and determine which activities should be prioritized. This condition creates an opportunity for conversational systems that can provide more flexible assistance through natural-language interaction.

The development of large language models has significantly extended the capabilities of conversational systems beyond conventional question answering. Recent LLM-based agents can combine natural-language understanding, reasoning, memory, and external tool use to perform goal-oriented activities [1]–[4]. ReAct introduced an approach that integrates reasoning with executable actions, enabling language models to interact with external environments during task completion [1]. Toolformer further demonstrated that language models can interact with external tools through API calls to extend their operational capabilities [2]. These developments provide an important foundation

for AI agents that can interpret user requests and subsequently invoke application functions rather than merely generate textual responses.

The integration of LLMs with external tools is particularly relevant for task-oriented and workflow-based applications. Instead of requiring users to issue rigid commands, an AI agent can interpret natural-language instructions and determine which predefined function or workflow should be executed [5]–[7]. Tool-oriented language models have also demonstrated the ability to select and invoke external functions according to contextual requirements [13]. This approach enables conversational interfaces to operate as an interaction layer between users and backend services such as databases, scheduling mechanisms, workflow automation platforms, and other application components.

Conversational technologies have also been widely adopted in educational environments. Chatbots have been investigated for learning assistance, information retrieval, academic support, and communication between students and educational services [8]–[10]. Messaging platforms provide a practical interface for these applications because users can access system functionality without installing or navigating a dedicated information system. Maulana, Purnawansyah, and Azis [11] demonstrated the implementation of a Telegram bot for retrieving information directly from a database. Their study showed that Telegram could function as a lightweight interface for accessing structured information. However, the system primarily focused on data retrieval and did not incorporate contemporary LLM-based capabilities such as dynamic natural-language interpretation, conversational memory, or tool-oriented data manipulation.

The integration of multiple information services is another important consideration in academic environments. Salim et al. [12] proposed a one-gateway system based on microservices architecture for managing campus information services. The study demonstrated the importance of providing a unified access mechanism to coordinate different services within an academic information environment. Although the architectural context differs from the present study, the underlying integration principle is relevant to AI-agent systems, where conversational interfaces, language models, workflow automation, storage services, and notification mechanisms must operate within a coordinated processing pipeline.

Despite these developments, a gap remains between conventional academic chatbots, task-oriented conversational systems, and modern LLM-based agents. Existing educational chatbots commonly emphasize information retrieval, question answering, or learning support [8]–[10], while research on LLM-based agents largely focuses on general reasoning, external tool use, and autonomous task execution [1]–[7], [13]. Relatively limited attention has been given to the integration of conversational task creation, retrieval, modification, deletion, contextual priority recommendation, conversational memory, and proactive reminders within a single academic task-management workflow.

To address this gap, this study develops an AI Task Agent for academic task management by integrating Telegram, n8n workflow automation, an LLM-based AI agent, Google Sheets-based task-management tools, PostgreSQL conversational memory, and an automated scheduling mechanism. Incoming Telegram messages are processed by the AI Agent through the n8n workflow, after which the agent determines the appropriate task-management operation and invokes the corresponding external tool. The system supports task creation, retrieval, modification, deletion, priority recommendation, and automated reminders. Task attributes such as deadline, urgency, and lecturer strictness are used as contextual information when generating priority recommendations.

The proposed system differs from conventional Telegram-based chatbots because the conversational layer is connected directly to executable task-management functions rather than being limited to information retrieval. The main contributions of this study are threefold. First, it presents an integrated architecture that combines Telegram, n8n, an LLM-based agent, external task-management tools, and conversational memory. Second, it enables tool-oriented task management through natural-language interaction, including Create, Read, Update, and Delete operations. Third, it combines contextual task-priority recommendations with scheduled reminders to provide both reactive task management and limited proactive assistance. Through this approach, the study demonstrates the practical application of LLM-based agents and workflow automation for conversational academic task management.

2. Method

This study employed a system-development approach to design and evaluate an AI-based conversational agent for academic task management. The proposed system integrates Telegram as the user interaction interface, n8n as the workflow orchestration platform, an LLM-based AI agent for interpreting natural-language requests and selecting appropriate tools, Google Sheets for task-data management, PostgreSQL-based conversational memory, and a scheduled workflow for automated reminders. This architecture follows the general concept of LLM-based agents in which language models interact with memory, external tools, and application environments to perform goal-oriented actions [3], [14], [15].

The development process consisted of five main stages: problem identification, requirement analysis, system design, workflow implementation, and functional evaluation. The functional requirements were derived from academic task-management activities, including creating tasks, retrieving stored information, updating task records, deleting tasks, generating priority recommendations, and delivering automated reminders.

Research Workflow

The overall research procedure is summarized as follows:

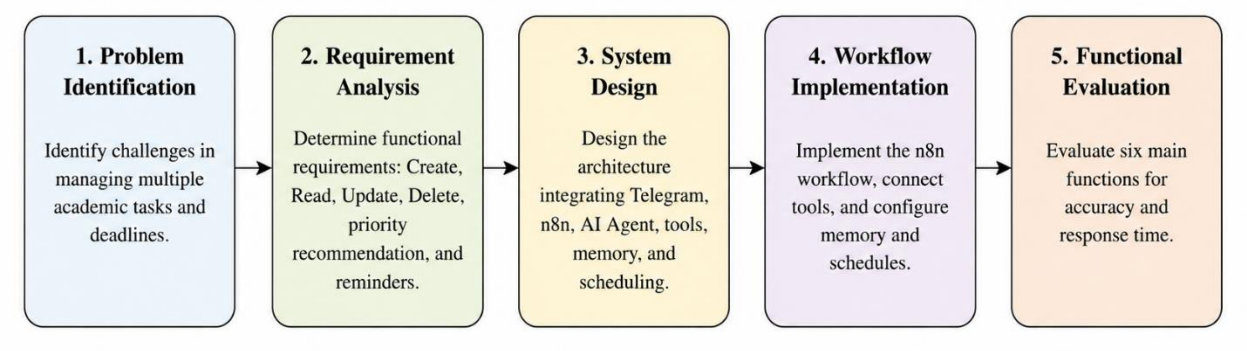

Figure 1. Research Flow

Problem identification focused on the difficulties associated with managing multiple academic tasks and deadlines. Requirement analysis determined the functions required to support conversational task management. System design defined the integration between Telegram, n8n, the AI Agent, task-management tools, conversational memory, and reminder services. Workflow implementation translated the proposed architecture into executable n8n workflows, while functional evaluation assessed the ability of the resulting system to execute its primary task-management functions.

The research workflow can be presented as **Figure 1**, while the complete system architecture can be presented separately as **Figure 2.** The original n8n workflow screenshot can subsequently be retained as an implementation-level illustration to demonstrate the actual integration of the AI Agent, Telegram Trigger, Google Sheets tools, PostgreSQL Chat Memory, and Schedule Trigger.

System Architecture

The proposed architecture consists of four main layers: the interaction layer, workflow orchestration layer, AI-agent layer, and data and execution layer. Telegram serves as the interaction layer through which users submit task-related requests and receive system responses. Incoming messages are captured by the Telegram Trigger and processed through n8n, which acts as the workflow orchestration layer.

The AI-agent layer contains an LLM-based agent connected to an OpenAI Chat Model. The agent interprets user messages, identifies the intended operation, and selects an appropriate external tool. The implementation provides Google Sheets tools for retrieving, adding, editing, and deleting task records. PostgreSQL Chat Memory is connected

to the agent to maintain conversational context across multiple interactions. In addition, a Schedule Trigger is used to initiate reminder-related workflows independently of direct user requests.

This architectural design is consistent with contemporary agent-oriented systems that separate natural-language reasoning from external tool execution [14]–[16]. Rather than allowing the language model to manipulate persistent data directly, the LLM determines the required action while predefined workflow components perform the actual operation.

The general processing architecture can be represented as

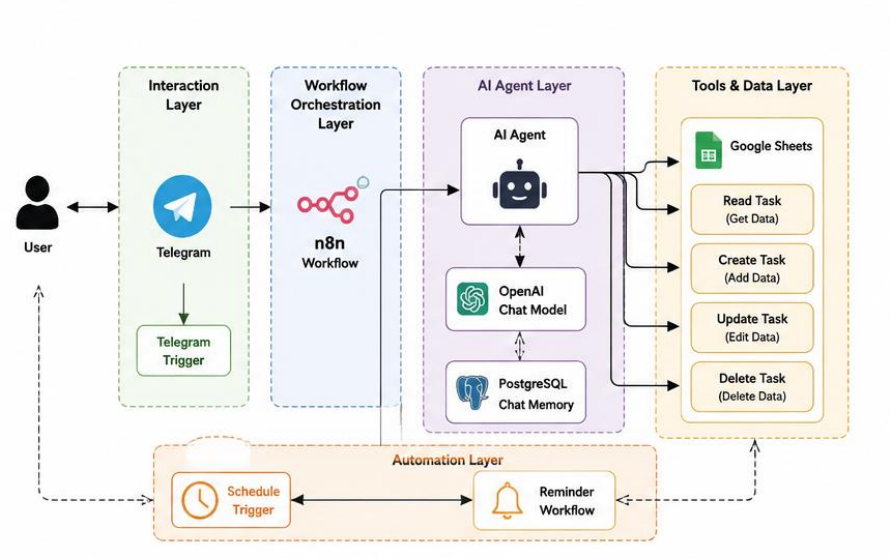

Figure 2. System Architecture of the AI Task Agent

The scheduled reminder mechanism operates in parallel with the user-initiated interaction workflow.

AI Agent and Task Management Workflow

The AI Task Agent serves as the main decision component of the proposed system. When a user sends a message through Telegram, the message is forwarded to the agent through n8n. The agent interprets the semantic meaning of the request and determines whether the user intends to create, retrieve, update, or delete task information.

The available task-management operations can be represented as

$$T = \{T_{create}, T_{read}, T_{update}, T_{delete}\} \quad (1)$$

For example, a request such as "*Add the Machine Learning assignment with a deadline tomorrow*" requires the agent to select the task-creation operation. In contrast, a request such as "*What assignments do I have tomorrow?*" requires the retrieval operation.

The implemented n8n workflow provides four Google Sheets tools corresponding to these operations: data retrieval, data insertion, data modification, and data deletion. After a selected tool has completed its operation, the result is returned to the AI Agent, which generates a natural-language response and delivers it to the user through Telegram.

The implementation of the proposed AI Task Agent workflow in n8n is illustrated in **Figure 3**, showing the integration of the Telegram Trigger, AI Agent, OpenAI Chat Model, PostgreSQL Chat Memory, task-management tools, and Telegram response mechanism.

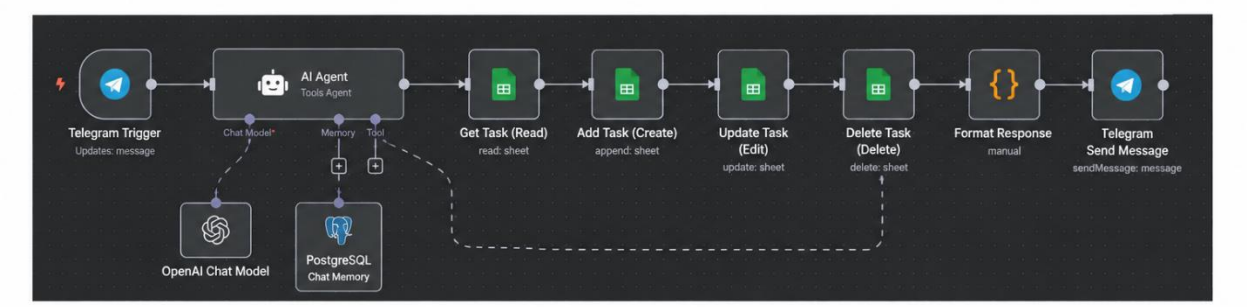

Figure 3. n8n Workflow Implementation of the AI Task Agent

This workflow demonstrates how the AI Agent acts as the central orchestration component between natural-language interaction and executable task-management functions. The language model supports request interpretation, while predefined tools perform the corresponding operations on the task repository.

This mechanism follows the principles of tool-augmented and task-oriented LLM systems, in which conversational input is translated into structured actions that can be executed by external applications [5], [7], [15], [17]. Separating language interpretation from application execution provides greater flexibility for natural-language interaction while maintaining controlled access to task data.

Conversational context is maintained using PostgreSQL Chat Memory. This component allows the agent to use information from previous exchanges when interpreting subsequent requests. Such memory is useful for multi-turn interactions in which a user may refer to previously mentioned tasks without repeating all task attributes. Persistent or contextual memory has become an important component of modern LLM-agent architectures because it supports continuity across sequential interactions [14], [16].

Task Representation, Prioritization, and Reminder

Each academic task is represented using several contextual attributes. Based on the system requirements, these attributes include task name, task type, description, execution time, deadline, urgency, and lecturer strictness.

Table 1. Task Information Used by the AI Task Agent

Attribute	Description
Task name	Name or title of the academic task
Task type	Category of the task
Description	Additional information related to the task
Execution time	Planned time associated with the task
Deadline	Final completion time
Urgency	User-defined level of task urgency
Lecturer strictness	Contextual information regarding deadline enforcement

In addition to managing task records, the system provides contextual priority recommendations. Three principal attributes are considered in the prioritization process:

$$X_p = \{D, U, S\}, \tag{1}$$

where D represents deadline information, U represents task urgency, and S represents lecturer strictness.

The available system documentation does not contain a deterministic weighting equation or validated numerical scoring model for these variables. Therefore, the implemented mechanism is described as LLM-assisted contextual prioritization. The agent evaluates the available task attributes and uses them as contextual information when recommending which task should receive greater attention.

This approach allows multiple contextual factors to be considered simultaneously, but it also introduces potential variability because the recommendation depends on the interpretation of the language model. Studies of LLM-based autonomous and tool-learning agents similarly indicate that context interpretation and decision consistency remain important challenges in agent-based applications [14], [15].

The system also incorporates an automated reminder mechanism using the Schedule Trigger in n8n. Unlike the Telegram Trigger, which is activated by user interaction, the Schedule Trigger initiates the workflow automatically according to the configured schedule. Relevant task information is retrieved and used to generate a reminder that is subsequently delivered through Telegram.

The system therefore supports two complementary interaction modes: reactive task management, initiated by user requests, and proactive task assistance, initiated through scheduled reminders.

Function Evaluation

The system was evaluated based on six functional categories: Create, Read, Update, Delete, priority recommendation, and automated reminder. These categories represent the main capabilities of the implemented AI Task Agent.

Functional performance was expressed using accuracy:

$$Accuracy = \frac{N_{correct}}{N_{tested}} \times 100\%, \quad (1)$$

where $N_{correct}$ represents the number of successfully executed responses and N_{tested} represents the total number of evaluated operations.

The evaluation examined whether each user request produced the expected system operation. Create testing evaluated whether new task information was stored correctly. Read testing examined whether relevant task records could be retrieved. Update testing evaluated modification of existing task information, while Delete testing examined whether the requested task could be removed correctly. Priority recommendation testing evaluated whether the system generated recommendations based on available task context, whereas reminder testing examined the execution of scheduled task notifications.

In addition to functional correctness, response time was recorded for each evaluated operation. Response-time evaluation was used to examine whether the workflow could provide sufficiently responsive interaction through Telegram. Agent-based systems should be evaluated not only on language-generation quality but also on their ability to select and execute appropriate actions in the external environment [14], [15].

The original evaluation records provide functional accuracy and response-time results for the six operations. However, the detailed raw test logs and exact number of individual test cases are no longer available. Therefore, the revised study retains the documented evaluation results without reconstructing unsupported experimental quantities such as confidence intervals, standard deviations, or statistical significance tests

3. Result and Discussion

System Implementation

The AI Task Agent was successfully implemented as an integrated conversational task-management system combining Telegram, n8n, an LLM-based AI agent, external task-management tools, conversational memory, and a scheduled reminder mechanism. The implemented workflow enables users to interact with the system using natural-language messages rather than accessing a separate task-management interface.

Incoming messages are received through the Telegram Trigger and forwarded to the AI Agent within n8n. The agent interprets the user's request and selects the appropriate operation. Task-management functions are executed through external tools for retrieving, creating, updating, and deleting task records. PostgreSQL Chat Memory maintains conversational context, while the Schedule Trigger supports automated reminder execution.

This implementation reflects the development of conversational systems from simple information-retrieval interfaces toward systems capable of interacting with backend applications. Generative AI-based conversational systems increasingly combine natural-language interaction with personalization and task execution [18], while workflow automation allows conversational interfaces to be connected with operational processes and external services [19]. Such integration enables the AI component to function as an interaction and decision layer rather than merely as a text-generation mechanism.

In the proposed system, this architecture allows users to perform task-management activities through Telegram, including adding a new assignment, retrieving task information, modifying existing task data, deleting a task, requesting a priority recommendation, and receiving scheduled reminders. This extends earlier Telegram-based academic information systems, such as the database retrieval approach developed by Maulana, Purnawansyah, and Azis [11], by enabling multiple executable operations instead of retrieval alone.

Functional Performance

Functional testing was conducted on six primary system functions: Create, Read, Update, Delete, priority recommendation, and automated reminder. The results are presented in **Table 2**.

Table 2. Functional Performance of the AI Task Agent

No.	Function	Accuracy	Recorded Processing Time	Telegram Response Time	Status
1	Create Task	100%	3.2 s	2.8 s	Successful
2	Read Task	100%	3.5 s	3.0 s	Successful
3	Update Task	100%	3.1 s	2.7 s	Successful
4	Delete Task	100%	3.4 s	3.0 s	Successful
5	Priority Recommendation	95%	3.3 s	2.9 s	Successful

The results indicate that the four CRUD operations achieved 100% functional accuracy, while priority recommendation and automated reminder achieved 95%. These results show that the system was able to perform its core task-management operations successfully within the evaluated scenarios.

The difference between CRUD operations and priority recommendation can be explained by the characteristics of each process. Create, Read, Update, and Delete functions are based on clearly defined external operations. Once the agent correctly determines the intended action, execution is performed by a predefined workflow tool. In contrast, priority recommendation requires interpretation of contextual information, including deadline, urgency, and lecturer strictness. Consequently, this function involves a greater degree of semantic interpretation by the AI Agent.

This distinction is consistent with current LLM-agent research, which identifies tool selection and context interpretation as more challenging than the deterministic execution of an already selected tool [14], [15]. The results therefore support the architectural decision to separate conversational reasoning from actual data manipulation. The LLM interprets the request, while predefined external tools perform persistent data operations.

Response-Time Analysis

The recorded processing times ranged from 3.1 to 3.5 seconds across the six evaluated functions. The second response-time measurement associated with Telegram ranged from 2.7 to 3.0 seconds. Based on the recorded values, the average processing time was approximately 3.32 seconds, while the average Telegram response time was approximately 2.90 seconds.

The relatively narrow variation across operations indicates that adding different task-management functions did not result in substantial differences in the recorded response duration. Create, Read, Update, and Delete operations exhibited response times comparable to those of priority recommendation and automated reminders.

Response speed is an important aspect of conversational interaction because excessive delays may reduce the perceived continuity of a dialogue. Conversational AI systems are expected to combine functional capability with accessible and responsive interaction [20]. The use of Telegram also reduces the need for users to navigate multiple interfaces because requests and responses occur within the same conversational environment.

However, the response-time results should be interpreted cautiously. The original experimental records do not provide detailed instrumentation separating LLM inference time, workflow processing time, data-access latency, Telegram transmission time, and complete end-to-end latency. Therefore, the values reported in **Table 2** represent implementation-level measurements rather than a controlled latency benchmark. A future evaluation should independently measure each processing component and report repeated measurements using statistics such as mean, standard deviation, and percentile latency.

Natural-Language Task Management and CRUD Operations

One of the primary advantages of the proposed system is the ability to connect natural-language interaction with executable task operations. The agent is not limited to providing information about stored tasks but can also invoke functions that change the underlying task records.

This capability represents an important progression from conventional command-based bots. Previous research has demonstrated that educational chatbots can provide accessible interaction for learning and academic services [21], [22]. Meta-analytic evidence also suggests that chatbot-supported interaction can contribute positively to educational activities when the chatbot is appropriately designed for its intended task [23].

The present system focuses specifically on task-oriented interaction. For example, semantically different messages may refer to the same underlying operation. A student may request that the system “add an assignment for tomorrow” or “record my machine learning task due tomorrow.” Both requests conceptually correspond to a Create operation, even though their linguistic forms differ.

Task-oriented LLM approaches similarly seek to reduce dependence on fixed intent commands by allowing language models to interpret user requests and connect them with executable application functions [17]. The 100% functional results reported for CRUD operations indicate successful execution in the documented test conditions. However, these results should not be interpreted as evidence that the system can correctly process every possible linguistic variation because the original records do not document the number, complexity, or linguistic diversity of test commands.

Future evaluations should therefore include paraphrased requests, incomplete instructions, ambiguous references, spelling variations, multi-turn commands, and requests containing multiple task attributes. Such testing would provide stronger evidence regarding the natural-language robustness of the AI Agent.

Contextual Priority Recommendation

The priority recommendation function is one of the principal features that distinguishes the proposed system from a basic task-recording chatbot. The system considers three contextual attributes when assisting users in identifying tasks that should receive greater attention: deadline, urgency, and lecturer strictness.

The priority recommendation achieved a reported functional accuracy of 95%. Although slightly lower than the CRUD results, this performance indicates that the agent was generally capable of using task context to generate priority recommendations within the documented evaluation scenarios.

Context-sensitive assistance is increasingly relevant in conversational systems because user requirements cannot always be addressed using a single fixed variable. Personalized educational chatbots, for example, increasingly incorporate contextual information to adapt interaction and support to individual conditions [18], [21]. Similarly, research examining AI in project and task management highlights the potential use of AI for scheduling, decision support, prioritization, and automation [25], [26].

Nevertheless, the proposed priority mechanism should be considered an initial contextual recommendation approach rather than a validated decision model. The available documentation does not provide a deterministic weighting formula for deadline, urgency, and lecturer strictness. Therefore, it cannot be concluded that the current mechanism generates an objectively optimal task ranking.

A stronger future implementation could define a transparent priority score, for example by combining normalized deadline proximity, urgency, and additional contextual factors. The resulting ranking could then be compared with expert-defined or user-defined priority labels. Such an evaluation would allow the prioritization component to be assessed independently from the conversational capabilities of the LLM.

Automated Reminder

The automated reminder function achieved a reported accuracy of 95%. Unlike CRUD operations that are initiated directly by user messages, the reminder mechanism can be activated through the Schedule Trigger. This enables the system to initiate a task-related interaction without requiring a new request from the user.

The combination of reactive and proactive functions is important for academic task management. Reactive interaction allows students to request information or execute operations when needed, while proactive interaction enables the system to provide support when a relevant deadline approach. Similar integration between conversational interfaces and scheduling functions has been investigated in other task-oriented domains, including appointment scheduling and automated notification systems [24].

This functionality moves the AI Task Agent beyond a purely reactive chatbot. However, the available implementation records do not provide sufficient information concerning reminder intervals, notification frequency, deadline thresholds, or duplicate-reminder prevention. These parameters should be explicitly defined in future implementations to improve reproducibility and provide users with greater control over notification behavior.

Comparison with Related Studies

The contribution of the proposed system does not depend on a single new technology. Telegram bots, educational chatbots, LLM-based agents, external tool invocation, and scheduling mechanisms have all been investigated independently. The contribution of this study lies primarily in combining these components within a single workflow for academic task management.

Table 3. Comparison with Representative Related Studies

Study	Main Application	Conversational Interaction	Backend/Tool Execution	Task Management	Automated Reminder
Maulana et al. [11]	Database information retrieval	Telegram	Database retrieval	Limited	No
Sánchez Cuadrado et al. [17]	Task-oriented LLM chatbot development	LLM-based	Application functions	Application dependent	Not primary focus
Chen et al. [18]	Personalized educational chatbot	Generative AI	Educational services	Limited	Not primary focus
Tabassum et al. [24]	Appointment scheduling	Chatbot	Scheduling operations	Scheduling oriented	Yes
Lee et al. [30]	University learning support	Multiplatform chatbot	Educational interaction	No	No
Proposed System	Academic task management	Telegram + LLM Agent	CRUD tools and workflow automation	Yes	Yes

Compared with the Telegram database retrieval approach in [11], the proposed system extends conversational access from Read operations to complete CRUD functionality. Compared with general task-oriented LLM approaches [17], the present study applies agent-based tool execution specifically to academic task management and integrates a scheduled reminder mechanism. Educational chatbot studies [18], [21]–[23], [27], [29], [30] demonstrate the growing

role of conversational systems in educational environments, but many primarily emphasize information provision, learning support, or interaction rather than executable task management.

The proposed architecture therefore combines three functional levels: conversation, execution, and proactive assistance. Conversation enables users to express task requirements naturally; execution connects those requirements to external task-management tools; and proactive assistance enables scheduled reminders without requiring an immediate user request.

This direction is also consistent with the increasing adoption of AI in workflow and project-management environments. AI-supported project management has been investigated for automation, scheduling, decision support, and improved coordination [19], [25], [26]. The present study adapts these broader concepts to a lightweight academic setting using a messaging interface and low-code workflow automation.

Limitations and Research Implications

The results demonstrate the feasibility of integrating an LLM-based conversational agent with workflow automation for academic task management. The use of external tools provides a practical separation between flexible natural-language interpretation and controlled data operations. In addition, Telegram provides a familiar communication environment that can reduce dependence on a separate dedicated task-management application.

These findings are consistent with previous reviews showing that conversational agents can provide accessible and immediate support in educational environments [21], [22], [27], [29]. However, previous studies also emphasize that the effectiveness of educational conversational systems depends strongly on reliability, usability, interaction design, and evaluation quality [21], [22].

Several limitations should therefore be considered when interpreting the present results. The original raw experimental logs are unavailable, preventing reconstruction of the exact number and linguistic diversity of test cases. The exact LLM model configuration, prompting strategy, and inference parameters are also unavailable. In addition, the priority recommendation mechanism does not have a documented deterministic ranking algorithm. The original manuscript mentioned an evaluation involving 30 respondents, but the questionnaire, sampling information, and raw responses could not be recovered; consequently, those data were not used to support quantitative usability claims in the revised study.

Future research should conduct controlled evaluations using a documented collection of natural-language task commands, including paraphrased, ambiguous, and multi-turn instructions. The system should also be evaluated using different LLM configurations to determine the effect of the underlying model on intent understanding and tool selection. A transparent priority-ranking algorithm, systematic latency measurements, configurable reminder policies, and a structured usability evaluation should also be incorporated. These improvements would enable a more rigorous assessment of both the technical reliability and practical usefulness of AI-assisted academic task management.

4. Conclusion

This study developed an LLM-based AI Task Agent for academic task management by integrating Telegram, n8n workflow automation, an AI agent, Google Sheets-based task operations, PostgreSQL conversational memory, and an automated reminder mechanism. The proposed system enables users to manage academic tasks through natural-language interaction, including creating, retrieving, updating, and deleting task records, requesting contextual priority recommendations, and receiving scheduled reminders.

The functional evaluation showed that the Create, Read, Update, and Delete operations achieved 100% accuracy, while the priority recommendation and automated reminder functions achieved 95% accuracy. The recorded response times were approximately three seconds across the evaluated functions, indicating that the implemented workflow was capable of providing responsive interaction within the tested environment. The integration of natural-language interpretation with predefined external tools also allowed the system to separate conversational reasoning from actual data manipulation, providing a practical architecture for task-oriented AI applications.

The main contribution of this study lies in combining conversational interaction, executable task-management operations, contextual prioritization, conversational memory, and proactive reminders within a single Telegram-based workflow. Unlike conventional academic chatbots that primarily provide information retrieval or question-answering functions, the proposed system extends conversational interaction toward direct task execution and limited proactive assistance.

Several limitations remain. The available experimental records do not contain the complete raw test logs, detailed LLM configuration, prompting strategy, or a deterministic priority-ranking model. Therefore, the reported results should be interpreted within the scope of the documented implementation. Future work should evaluate the system using a larger and explicitly documented set of natural-language commands, compare different LLM configurations, introduce a transparent and validated task-priority ranking mechanism, perform controlled end-to-end latency measurements, and conduct a structured usability study involving target users. These improvements could provide stronger evidence regarding the reliability, scalability, and practical effectiveness of AI-assisted academic task management.

References

[1] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "ReAct: Synergizing reasoning and acting in language models," in *Proc. 11th Int. Conf. Learn. Representations (ICLR)*, 2023, doi: https://doi.org/10.48550/arXiv.2210.03629.

[2] T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli, E. Hambro, L. Zettlemoyer, N. Cancedda, and T. Scialom, "Toolformer: Language models can teach themselves to use tools," in *Advances in Neural Information Processing Systems*, vol. 36, pp. 68539–68551, 2023, doi: https://doi.org/10.52202/075280-2997.

[3] Q. Wu et al., "AutoGen: Enabling next-gen LLM applications via multi-agent conversation," in *Proc. Conf. Language Modeling (COLM)*, 2024, doi: https://doi.org/10.48550/arXiv.2308.08155.

[4] J. S. Park, J. C. O'Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein, "Generative agents: Interactive simulacra of human behavior," in *Proc. 36th Annu. ACM Symp. User Interface Software and Technology (UIST)*, 2023, Art. no. 2, pp. 1–22, doi: https://doi.org/10.1145/3586183.3606763.

[5] S. Gallo, F. Paternò, and A. Malizia, "A conversational agent for creating automations exploiting large language models," *Personal and Ubiquitous Computing*, vol. 28, pp. 931–946, 2024, doi: https://doi.org/10.1007/s00779-024-01825-5.

[6] S. Kusal, S. Patil, J. Choudrie, K. Kotecha, S. Mishra, and A. Abraham, "AI-based conversational agents: A scoping review from technologies to future directions," *IEEE Access*, vol. 10, pp. 92337–92356, 2022, doi: https://doi.org/10.1109/ACCESS.2022.3201144.

[7] H.-D. Xu, X.-L. Mao, F. Sun, T.-Y. Che, C. Xu, and H. Huang, "AgentTOD: A task-oriented dialogue agent with a flexible and adaptive API calling paradigm," *ACM Transactions on Information Systems*, vol. 43, no. 5, Art. no. 136, pp. 1–32, 2025, doi: https://doi.org/10.1145/3745021.

[8] J. A. Kumar, "Educational chatbots for project-based learning: Investigating learning outcomes for a team-based design course," *International Journal of Educational Technology in Higher Education*, vol. 18, Art. no. 65, 2021, doi: https://doi.org/10.1186/s41239-021-00302-w.

[9] C. W. Okonkwo and A. Ade-Ibijola, "Chatbots applications in education: A systematic review," *Computers and Education: Artificial Intelligence*, vol. 2, Art. no. 100033, 2021, doi: https://doi.org/10.1016/j.caeai.2021.100033.

[10] S. Wollny, J. Schneider, D. Di Mitri, J. Weidlich, M. Rittberger, and H. Drachsler, "Are we there yet? A systematic literature review on chatbots in education," *Frontiers in Artificial Intelligence*, vol. 4, Art. no. 654924, 2021, doi: https://doi.org/10.3389/frai.2021.654924.

[11] M. I. Maulana, Purnawansyah, and H. Azis, "Implementasi Bot Telegram pada proses retrieval data dalam database," *Buletin Sistem Informasi dan Teknologi Islam*, vol. 1, no. 3, pp. 150–157, 2020, doi: https://doi.org/10.33096/busiti.v1i3.835.

[12] Y. Salim, I. Muis, L. Syafie, H. Azis, and A. R. Manga, "One-gateway system in managing campus information system using microservices architecture," *Bulletin of Social Informatics Theory and Application*, vol. 7, no. 2, pp. 83–91, 2023, doi: https://doi.org/10.31763/businta.v7i2.635.

[13] S. Hao, T. Liu, Z. Wang, and Z. Hu, "ToolkenGPT: Augmenting frozen language models with massive tools via tool embeddings," in *Advances in Neural Information Processing Systems*, vol. 36, pp. 45870–45894, 2023, doi: https://doi.org/10.52202/075280-1988.

[14] L. Wang et al., "A survey on large language model based autonomous agents," *Frontiers of Computer Science*, vol. 18, Art. no. 186345, 2024, doi: https://doi.org/10.1007/s11704-024-40231-1.

[15] W. Xu, C. Huang, S. Gao, and S. Shang, "LLM-based agents for tool learning: A survey," *Data Science and Engineering*, vol. 10, pp. 533–563, 2025, doi: https://doi.org/10.1007/s41019-025-00296-9.

[16] X. Li, S. Wang, S. Zeng, Y. Wu, and Y. Yang, "A survey on LLM-based multi-agent systems: Workflow, infrastructure, and challenges," *Vicinagearth*, vol. 1, Art. no. 9, 2024, doi: https://doi.org/10.1007/s44336-024-00009-2.

[17] J. Sánchez Cuadrado, S. Pérez-Soler, E. Guerra, and J. de Lara, "Automating the development of task-oriented LLM-based chatbots," in *Proc. 6th ACM Conf. Conversational User Interfaces (CUI '24)*, Luxembourg, 2024, Art. no. 1, 11 pp., doi: https://doi.org/10.1145/3640794.3665538.

[18] D.-L. Chen, K. Aaltonen, H. Lampela, and J. Kujala, "The design and implementation of an educational chatbot with personalized adaptive learning features for project management training," *Technology, Knowledge and Learning*, vol. 30, pp. 1047–1072, 2025, doi: https://doi.org/10.1007/s10758-024-09807-5.

[19] M. Ben Hassen and A. Bellaaj, "Optimizing agile project management with a Copilot extension for Jira: Integrating AI, automation, and business intelligence," *Procedia Computer Science*, vol. 278, pp. 2028–2038, 2026, doi: https://doi.org/10.1016/j.procs.2026.03.200.

[20] L. Ramaul, P. Ritala, and M. Ruokonen, "Creational and conversational AI affordances: How the new breed of chatbots is revolutionizing knowledge industries," *Business Horizons*, vol. 67, no. 5, pp. 615–627, 2024, doi: https://doi.org/10.1016/j.bushor.2024.05.006.

[21] M. A. Kuhail, N. Alturki, S. Alramlawi, and K. Alhejori, "Interacting with educational chatbots: A systematic review," *Education and Information Technologies*, vol. 28, no. 1, pp. 973–1018, 2023, doi: https://doi.org/10.1007/s10639-022-11177-3.

[22] L. Labadze, M. Grigolia, and L. Machaidze, "Role of AI chatbots in education: Systematic literature review," *International Journal of Educational Technology in Higher Education*, vol. 20, Art. no. 56, 2023, doi: https://doi.org/10.1186/s41239-023-00426-1.

[23] E. Alemdag, "The effect of chatbots on learning: A meta-analysis of empirical research," *Journal of Research on Technology in Education*, vol. 57, no. 2, pp. 459–481, 2025, doi: https://doi.org/10.1080/15391523.2023.2255698.

[24] A. Tabassum, A. Sultana, A. Noor, M. S. Alam, and F. Tasnim, "Intelligent healthcare scheduling: A multilingual Rasa chatbot and Flutter-based doctor appointment framework," in *Proc. 2025 IEEE International Women in Engineering Conference on Electrical and Computer Engineering (WIECON-ECE)*, Dhaka, Bangladesh, 2025, doi: https://doi.org/10.1109/WIECON-ECE69386.2025.11526351.

[25] T. V. Fridgeirsson, H. T. Ingason, H. I. Jonasson, and H. Jonsdottir, "An authoritative study on the near future effect of artificial intelligence on project management knowledge areas," *Sustainability*, vol. 13, no. 4, Art. no. 2345, 2021, doi: https://doi.org/10.3390/su13042345.

[26] V. Holzmann, D. Zitter, and S. Peshkess, "The expectations of project managers from artificial intelligence: A Delphi study," *Project Management Journal*, vol. 53, no. 5, pp. 438–455, 2022, doi: https://doi.org/10.1177/87569728211061779.

[27] R. Wu and Z. Yu, "Do AI chatbots improve students' learning outcomes? Evidence from a meta-analysis," *British Journal of Educational Technology*, vol. 55, no. 1, pp. 10–33, 2024, doi: https://doi.org/10.1111/bjet.13334.

[28] A. Ganguly, N. Mehjabin, A. Malik, and A. Johri, "Conversational AI agents in education: An umbrella review of current utilization, challenges, and future directions for ethical and responsible use," *AI and Ethics*, vol. 6, Art. no. 72, 2026, doi: https://doi.org/10.1007/s43681-025-00916-0.

[29] J. Quiroga Pérez, T. Daradoumis, and J. M. Marquès Puig, "Rediscovering the use of chatbots in education: A systematic literature review," *Computer Applications in Engineering Education*, vol. 28, no. 6, pp. 1549–1565, 2020, doi: https://doi.org/10.1002/cae.22326.

[30] L. K. Lee, Y. C. Fung, Y. W. Pun, K. K. Wong, M. T. Y. Yu, and N. I. Wu, "Using a multiplatform chatbot as an online tutor in a university course," in *Proc. 2020 International Symposium on Educational Technology (ISET)*, Bangkok, Thailand, 2020, pp. 53–56, doi: https://doi.org/10.1109/ISET49818.2020.00021.