Journal of Computers Vol. 37 No. 4, August 2026, pp. 75-102
https://doi.org/10.63367/19911599202608370400 6

A Visualized Chain of Thought System Based on the Symmetry Concept: Cracking the Dilemma of Complex Reasoning in Large Language Models

Ai Wu[1], Shun-Nian Luo[2], Jungmin Li[3*], and Yicheng Jin[2]

[1] School of Overseas College (College of Foreign Languages), Sanming University, Sanming, China
[2] School of Information Science and Technology, Sanda University, Shanghai, China
[3] College of Marine Culture and Law, Jimei University, Xiamen, China
vajranam@foxmail.com

Received 22 October 2025; Revised 13 January 2026; Accepted 15 January 2026

Abstract. Since 2023, large language models (LLMs) have achieved remarkable progress in natural language processing, but they still face dilemmas of unclear reasoning processes and low accuracy when tackling complex tasks such as logical reasoning and mathematical computation. The Chain of Thought (CoT) strategy has emerged as a viable solution to this problem, yet traditional CoT construction and management tools (e.g., LangChain) suffer from high abstraction and cumbersome operations. To address these issues, this study proposes a visualized CoT system integrated with the symmetry concept. The symmetry principle ensures consistency and coordination across all operational links, while the intuitive graphical interface and dynamic interactive functions simplify the entire process of CoT construction, adjustment, and management—effectively lowering the technical threshold for users. By presenting complex thinking structures in a visually and logically symmetrical form, the system enhances the orderliness and operability of complex reasoning processes. This work contributes to simplifying LLM reasoning workflows and promoting the application of CoT technology among non-technical users.

Keywords: LLMs, question-answering system, CoT, visual editing, low code

1 Introduction

1.1 Background and Significance

Since 2023, large language models (LLMs) have excelled in text generation and language understanding, but their reasoning processes remain opaque and inaccurate when handling logic-intensive tasks (e.g., logical reasoning, mathematical computation). The Chain of Thought (CoT) strategy—guiding models to present intermediate reasoning steps—effectively improves complex problem-solving capabilities, yet its implementation is hindered by high abstraction and cumbersome operations. While tools like LangChain simplify CoT construction through modular design, they still require programming skills and lack intuitive visualization, creating barriers for non-technical users. The development of a visualized CoT system is therefore critical: intuitive graphical interfaces and dynamic interactions can reduce operational complexity, provide real-time feedback, and enhance the efficiency of LLMs in handling complex tasks—unlocking great potential in education, scientific research, and management.

The introduction of a visual chain of thought system effectively solves this problem. The graphical interface and dynamic interaction simplified the construction, management and coordination of the chain of thought. The whole process reduces the operational difficulty and improves the efficiency of the model in handling complex tasks. This innovative tool provides users with an efficient operation platform also creates conditions for optimizing and exploring model reasoning paths, which brings great development potential in diverse application scenarios such as education, scientific research and management.

* Corresponding Author

1.2 Chain of Thought (CoT) Overview

Chain of Thought, or CoT for short, has built up a multi-dimensional and cross-field research system since it was proposed, and in this content, the breakthroughs made in the core theory, the continuous evolution of technical tools, and the industry applications will be sorted out from the three dimensions of the relevant research progress.

Progress in Core Theoretical Research. Wei et al. in 2022 in "Chain of Thought Prompting Elicits Reasoning in Large Language Models" [1] made the first systematic demonstration of the feasibility of the CoT strategy, with the help of a leap from 58% to 70% accuracy on the GSM8K mathematical inference dataset. Which confirms that The Google Research team in 2022 integrated CoT into a 540B-parameter PaLM model, achieving 74% SOTA performance in complex logic tasks, and revealed a nonlinear positive correlation between model size and reasoning power. implicit CoT, which can maintain 95% inference accuracy in zero-sample scenarios and significantly reduce the generation cost of explicit inference chains [2]. The pathway diagram for innovation and application of the Big Model tool chain is shown as Fig. 1.

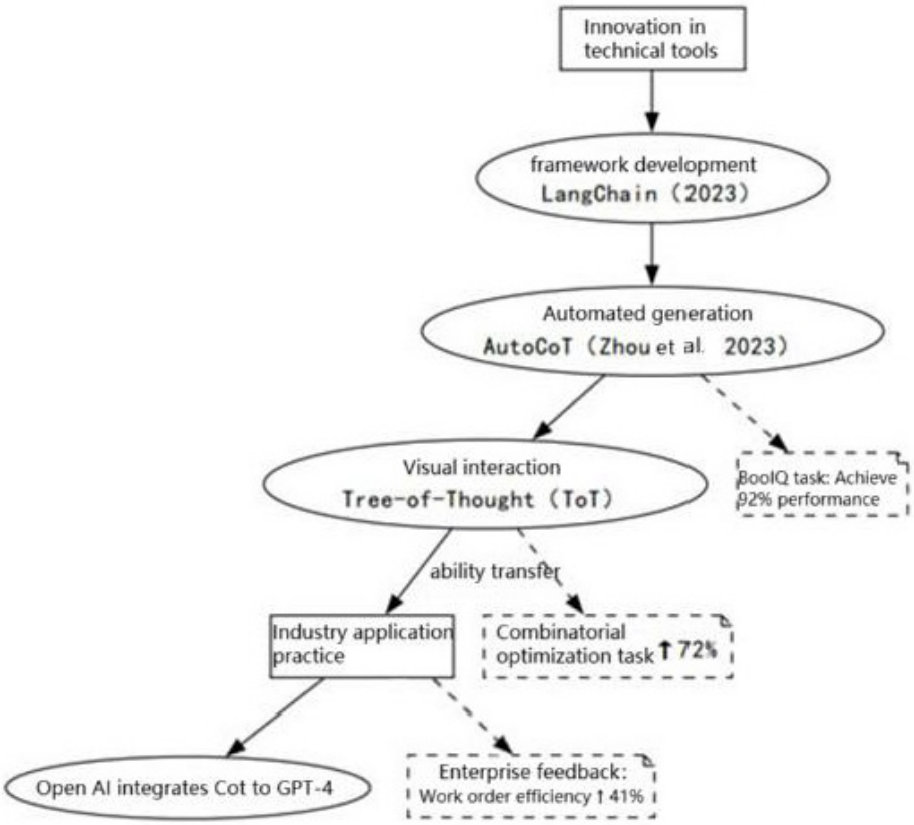

Fig. 1. Pathway diagram for innovation and application of the Big Model tool chain

Innovation in Technological Tools. The development path of the technology tool is gradually shifting from "framework development" to "automated construction" and "visual interaction". In 2023, the Langchain framework standardizes the structure of chain of thought with a modular design. Its open source ecosystem integrates more than 200 predefined reasoning templates. From the perspective of automatic generation, Zhou et al. proposed an automatic generation system system. It introduces an augmented learning mechanism to automatically generate chain of thought prompts based on specific task requirements. The system achieves 92% of the performance of manual prompts on the BoolQ dataset [3]. At the structural level, the chain of thought

proposed by Yao et al. combines multipath inference with a dynamic backtracking strategy. When dealing with optimization tasks (e.g., combinatorial 24-point games), the accuracy will be improved to 72% [4], Table-as-Thought paradigm provides a structured alternative for tabular reasoning tasks [5].

With each repeated iteration, application solutions from various industries show significant differentiation. After GPT-4 deepened the chain-of-thinking mechanism through the API service system, empirical data from company users showed a 41% increase in the efficiency of complex work order processing. Systems with chain-of-thinking reasoning show greater performance benefits for many tasks (e.g., logical inference, numerical computation, and program code generation) than traditional mass-trained models.

Current orchestration tools face several key challenges that need to be addressed, and comprehensive comparisons of prompt engineering paradigms can be found in existing survey work [6].

1. Editorial chain of thought are relatively complex and costly to maintain. Existing solutions such as Langchain are opaque in the face of complex process design, increasing the burden on developers to understand and debug, and lack harmonized data standards between modules.
2. Inconsistent data structures and incompatible interfaces limit scalability across components and hinder collaboration between system generalizations.
3. From the perspective of batch task processing, it is clear that existing visual orchestration tools lack concurrency. In large application scenarios, this technical limitation severely restricts the practical use of the tools.
4. The system architecture design suffers from redundancy issues and uses very specific communication protocols. This technical solution leads to a strong coupling between the development framework and specific interfaces and protocols, which significantly increases the complexity of the system's second-level development and creates major obstacles to achieving rapid feature customization and iterative optimization of the implementation.
5. The complexity of the inference paths of the TOT structure shows nonlinear growth characteristics, and its growth rate follows the mathematical law of N(n-1)/2. [7] The scalability of this dendritic structure greatly increases the difficulty of system maintenance and the complexity of routing optimization [8].
6. Implicit COT reduces the interpretability of reasoning. The computational power requirements have been reduced, but the transparency of reasoning is greatly reduced. Deng et al. state that it reduces interpretability by 37% [9].

Aiming at the problems of high construction complexity, insufficient visualization support and lack of systematic tools in the practical application of chain of thought technology, this research designs and implements a Web-based visualized chain of thought management system. Core innovations and contributions include:

Visual Interaction Innovations hopes to develop a graphical editor that supports visual node arrangement and dynamic linking, and relies on a combination with LeaderLine.js to enable the intuitive construction of multi-level chains of thought, reducing the difficulty of operation for non-technical users. This allows the user to create a chain of thoughts and then visualize the logic of its operation from the graphical interface.

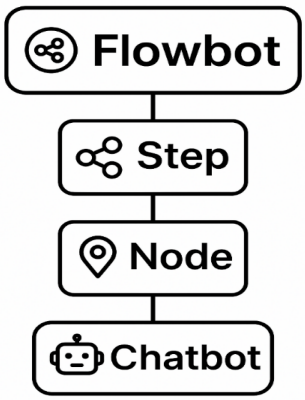

Fig. 2. Flowbot chain of thought engine and intelligent task modeling framework

In Fig. 2. The elements that make up a chain of thought or chain of thought are generally more complex, although the system presented in this paper redefines the core structure of process modeling at the beginning of the design.

A new set of conceptual systems is given to more clearly support the visual construction of intelligent tasks and the management of their execution, encompassing the following key concepts:

- Flowbot: represents a complete intelligent task flow and is the top-level carrier of the flow;
- Step: Within Flowbot there is a logical grouping unit called Step that organizes the order of execution of the nodes and allows for the division of stages;
- Node (node): the smallest execution unit in the process, carrying the jump logic, attribute configuration and intelligent call;
- Chatbot: bound to Node, responsible for interacting with external dialog models to realize semantic processing and intelligent decision making.

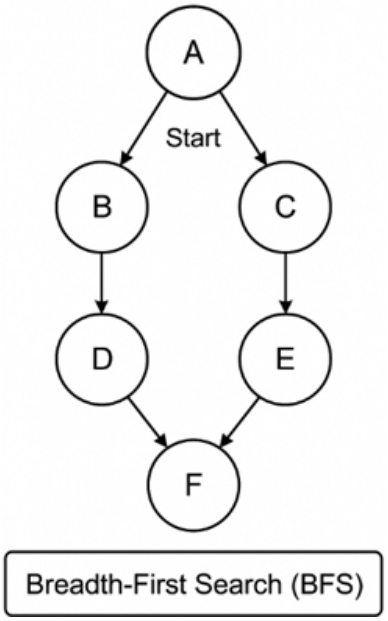

Fig. 3. Example of BFS execution flow

The example of BFS execution flow is shown as Fig. 3. This design has a modularized chain of thought, based on the breadth-first search (BFS) execution framework. Integrated regular expression rule engine and external service invocation mechanism, support for dynamic jumping and fault-tolerant control of complex logic paths; in terms of execution, in the chain of thought construction process, there will exist a full process of empirical verification to avoid logical errors. Demonstrate that the project can greatly reduce the time consumption of orchestrating the chain of thought. In this paper, the system introduces the symmetry concept in architecture, logic control and user interaction, emphasizes the consistency of visual intuition and execution process, and significantly improves the constructability, visibility and controllability of the complex chain of thoughts.

2 Theory and Techniques

2.1 The Origin and Development of the Chain-of-Consciousness Strategy

Wei et al, first systematically proposed the Chain of Thought (CoT) strategy, or Chain of Thought for short. Its core point is to guide a large-scale language model to present the intermediate reasoning process step by step, so as to improve the model's performance in logical reasoning and complex tasks [1]. This strategy is proposed

because large-scale language models have limitations in complex reasoning tasks: despite their powerful language generation capabilities, they still have outstanding deficiencies in the transparency and accuracy of the reasoning process. the CoT strategy uses a step-by-step steering mechanism to allow the model to present the complete chain of reasoning before generating the final answer, which improves the model's reasoning ability and the interpretability of the results. subsequent studies have shown that CoT effectively mitigates the long-range dependency defect of the Transformer model with the help of decomposing complex problems into interpretable intermediate states [10].

Recent research has shown that the application of chain of thought has greatly accelerated the development of the field. Kojima and his colleagues found empirically that in a sample of zero-learning conditions, where only simple cues, such as "stage," were needed, and the model could automatically generate effective reasoning processes. The strategy has been shown to be effective in a strategy QA benchmark. This improved the accuracy of the model by 21 percentage points [11]. Role-play oriented prompting can further boost the zero-shot reasoning capacity of LLMs [12]. Further theoretical studies have confirmed that the COT mechanism can help in cross-structure modeling through traditional sequencing. The algorithmic constraints have been successfully applied to solve backpacking problems that require dynamic programming capabilities [13]. These findings fully demonstrate that COT techniques are constantly evolving and improving [14]. The adaptability and performance in various task scenarios have been systematically improved.

Application of the Chain of thought to Different Tasks. The mind-chaining strategy has been shown to be valuable in a variety of logic-intensive tasks, such as mathematical reasoning, general knowledge quizzing, and programming code generation, and extended research has confirmed its generalizability to legal reasoning and medical diagnostic professional scenarios.

In mathematical reasoning tasks, models often need to complete multi-step computations and rigorous logical derivations. In further work, Shao et al. proposed the Math Shepherd framework, which combines process supervision and reward mechanism, and achieved an accuracy rate of 82.3% in dynamic planning-type problems [15], which fully proves the effectiveness of chain of thought in complex structural problems. Further performance gains on mathematical tasks can be achieved via progressive-hint prompting [16].

For general knowledge Q&A tasks, the model not only needs to understand the question, but also needs to construct a reasonable chain of contextual reasoning. The research study by Wei et al. showed that CoT strategy can significantly improve the contextual understanding of the model and the accuracy of the answers [1]. On this basis, Yao et al. further proposed an enhanced CoT reasoning method based on the ReAct framework, which improved the accuracy to 82% on the StrategyQA dataset [17]. This shows that the strategy performs particularly well on multi-round conditional problems.

In the field of code generation, the challenges faced by models mainly stem from the complexity of syntactic structure and logic control. By applying the chain-of-thinking strategy to the code generation process, the model can be modularized to disassemble the generation logic and effectively reduce syntactic errors and logic loopholes, as illustrated by the empirical study of LangChain, which reduced syntactic errors by about 35% in Python tasks and improved the executability and quality of generated code. LangChain's empirical research shows that it reduces the syntax error rate by about 35% in Python tasks, which improves the executability and quality of the generated code [14].

Based on the above research that already exists, it is not difficult to see that the chain-of-thinking strategy has gradually become an important path to improve the reasoning ability of large language models, which is not only applicable to abstract computational tasks, but also shows extensive adaptability to scenarios such as open-domain question-and-answer and structured text generation.

Current Status of Research on Visualized Chain of thought Systems. The construction and management of effective chain of thought strategies has been widely attracted as a central issue in current scholarship, with research progress reflected in four main areas [18]. First, we optimize generative mechanisms and improve task adaptation. Second, we develop an effective visual interaction management tool for the reasoning process. Third, we promote automated generation techniques and reduce labor costs. Fourth, we improve evaluation and optimization methods to enhance the performance of complex tasks. The coordinated development of these research directions has not only facilitated theoretical breakthroughs in chain-of-thinking strategies, but also laid a solid foundation for their practical application. It is important to note that while innovative tools such as Langchain can significantly improve the efficiency of chain construction through modular design, visual frameworks supporting standard BPMN 2.0 editors and node editors (e.g., ReactFlow) can handle more than

200 stages of complex processes. However, existing tools face challenges such as high abstraction and tedious manipulation of when dealing with multi-step thinking steps. In particular, the lack of user interaction experience and real-time optimization features severely limit the breadth of applications. To cope with this situation, Prof. Yang's research team proposed the concept of thought buffers. The response time of less than 200 ms is optimized by dynamically updating the inference template library [19]. This provides an important reference for the development of a new visualization system that combines a graphical interface with dynamic interactive features.

2.2 Visualization Construction Techniques

In this study, we hope to design and implement a Web-based visualization of the chain of thought system, relying on an intuitive graphical interface and dynamic interactive features to help users smoothly build, manage and optimize the chain of thought, the next will be a detailed description of the system's architecture, key technologies and their innovations.

System Architecture Overview. This system adopts the architecture design of front and back end separation; the specific structure is as Fig. 4:

1. Frontend: responsive design using Bootstrap 4, combined with LeaderLine.js to achieve dynamic connections between nodes and provide a user-friendly visualization interface.
2. Backend: built with Python language combined with Flask framework.
3. Database: FAISS vector database is used for efficient vector retrieval and Pandas for data processing and management.

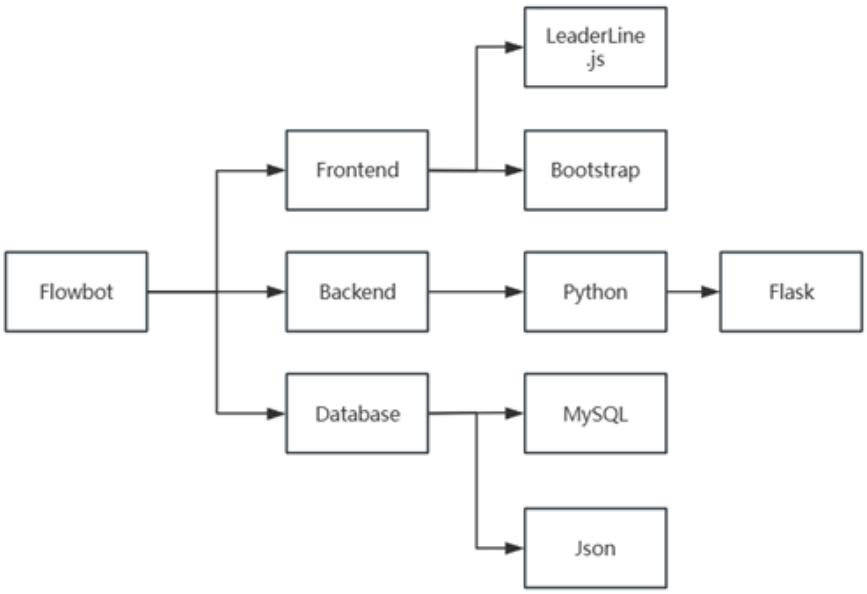

Fig. 4. Project system architecture diagram (Technology Stack)

Key Technologies and Implementation Details. Frontend technology: Bootstrap 4 framework was used for the Frontend to ensure responsive interface and cross-platform compatibility. LeadingLine.js is used to implement dynamic connecting lines between nodes, drag-and-drop support and real-time updates, and to improve the interactive user experience in building chain of thought.

Backend Technology: The backend uses the Python language and builds web servers based on the Flask framework to handle Frontend requests and back-end logic. FAISS vector database is used to implement efficient vector searches and can support fast querying of big data.Pandas library performs core data cleansing, processing and analysis tasks within the system. Powerful data processing functions ensure efficiency and maximum

utilization of data management. From the perspective of interactive features, the system provides intuitive operations, such as node editing, dynamic generation and connecting lines, to provide users with a convenient chain of thought building and optimization experience. The system integrates real-time feedback and intelligent optimization proposal functions, and dynamically adjusts the chain of thought structure based on user operations and model reasoning results to improve the accuracy and execution efficiency of the reasoning process.

Calculation Formula. Since chain of thought often invoke large language models multiple times during operation, this study additionally counts the total frequency of model invocations and the corresponding token consumption. The chain of thoughts is abstracted as a directed graph structure G=(V,E). Here, V denotes the set of nodes and E denotes the set of edges. In order to comprehensively analyze the execution process of the chain of thoughts, the system employs a breadth-first search algorithm to achieve a systematic exploration of the chain of thoughts graph, with the starting node v_0 V as the traversal starting point.

In the process, step S denotes the depth level of the BFS. l_s denotes a set of S-level nodes. m(v) indicates whether node V should invoke a large model or not. Based on this, the total number of large models actually triggered in the chain of thought can be expressed as follows:

$$C = \sum_{s=0}^{d} \sum_{v \in L_s} M(v) \tag{1}$$

A hierarchical statistical approach is used to analyze large model calls at each level node in the chain-of-thought graph. First, we define the maximum depth D of the chain-of-thought graph as the largest level of the first search (BFS). Second, for each level step, all nodes that satisfy the condition M (V) = 1 are selected for cumulative computation. It accurately calculates the total number of tokens in the fast input section, focusing on the count of nodes that need to make large model calls at each level. Given that current large model services often employ a billing model based on the number of tokens, this metric can be used as an important reference for evaluating computational resource consumption. In order to eliminate the effect of output length uncertainty in large language models, we focus on the "minimum token consumption", i.e., the cumulative token value of the fast input portion of all model-calling nodes in a single run.

$$\bar{T}_{min} = \frac{T_{min}}{C} \tag{2}$$

Among them:

t_{in} (v) denotes the token consumption of each model that calls a node on the input.

T_{min} is the total input token cost of the model calls throughout the chain of thought process during the entire operation.

C is the total number of model calls.

$\bar{T}_{min}$ is the average minimum input token consumption per unit model call.

Innovation Points and Advantages. Visualization construction: The system carries out innovative design work in the visualization construction of the chain of thought, providing users with an intuitive graphical interface, which enables users to easily understand and edit the relationship between the nodes of the chain of thought and effectively reduces the technical difficulty of constructing the chain of thought.

User-friendliness: The user-friendliness of the system has been enhanced through clear and concise operation flow and interaction design. Even non-technical users can complete complex chain of thought construction and management through simple operations such as drag and drop and click.

2.3 The Existing Methods

Inside the current research, the optimization as well as application of chain of thought strategies is mainly based on the concepts related to the following types of methods:

Frameworks such as LangChain. Functions and features: LangChain, as an open source framework launched in 2023 specifically for chained reasoning construction and management, relies on modular design. Giving users a flexible way to combine functional components, this framework integrates core modules such as hint templates, callback functions and model interfaces, simplifying the construction process of the chain of thought. It is worth noting that LangChain has extensive support for multi-language models and diversified task types, has strong scalability and task adaptability. Cross-lingual Tree-of-Thought designs further extend multilingual reasoning capacity [20].

LangChain greatly reduces the technical threshold of chain of thought construction by providing rich functional modules. However, it still appears to be abstract and operationally complex when dealing with highly complex and multi-layered logical reasoning, and users need to have certain programming and model tuning skills, which limits its popularity among non-technical users.

Automated Prompt Generation Techniques. Automated Prompt Engineering, which is also known as Automated Prompt Engineering, relies on algorithms to automatically generate chain of thought prompts that fit the needs of a particular task, thus reducing the workload of manual design. Although automated methods are effective in reducing manual intervention, there are still areas that can be optimized when dealing with highly complex and specialized tasks [21]. These methods have to improve the reasoning chain generating mechanism to ensure that the logic chain is reasonable and effective, to strengthen the depth of understanding of the task. The optimization of the reasoning path and the integration of domain knowledge, etc., to improve the performance and reliability of the system in professional scenarios, and this optimization can improve the practicality of the automated methods, and also expand the scope of their application in more professional fields [22].

Other Visualization Tools and Systems. Dify and Coze integrate a series of functional modules and interfaces that enable users to design and optimize efficient cueing templates. These tools are generally embedded in existing frameworks and support rapid iteration and multi-strategy testing, which greatly improves the efficiency of cueing engineering.

In the field of visual chain of thought systems, several related tools are already available on the market, in addition to the solutions proposed in this study. These tools provide users with intuitive chain of thought building and management capabilities with the help of graphical interfaces and interaction design. Some tools also support the import and export of chain of thought that effectively facilitate integration and collaboration between systems. However, these tools still have shortcomings in user interaction experience and real-time optimization suggestions that need to be improved.

From a technical implementation perspective, the various tools exhibit different characteristics of architectural design. Some tools use browser-based Frontend technologies for immediate visualization, while others rely on back-end services focused on complex logic and data management. Existing tools have unique features in terms of functionality and performance, but require continuous optimization and improvement in terms of system consistency and user experience consistency.

2.4 Comparison of These Methods

The visualized chain of thought system constructed in this study will be compared with existing methods in terms of functionality, performance, and user experience, so as to highlight the advantages and unique innovations of the system.

Comparison of Functions and Characteristics. Comparison of system functions: Comparing this system with existing frameworks such as LangChain, we can find that in the visual construction and management of the chain of thought, this system has a more intuitive graphical interface, with the help of which the user can complete the construction of complex chains of thought even without programming, and the system integrates a feedback function, which improves the quality of the chain of thought and the efficiency of its execution. The system incorporates a feedback function that improves the quality of the reasoning chain and the efficiency of its execution. This innovative function dynamically analyzes the results of user operations and model inference, and automatically generates optimization suggestions to effectively guide the user in adjusting the inference path [23]. With continuous quality monitoring and intelligent optimization, the system ensures the accuracy of the inference process, and greatly improves the efficiency of solving complex tasks, thus providing the user with

more reliable and efficient inference support.

Comparison of features: The system supports a multi-level chain of thought structure, which can cope with highly complex logical paths. The modular architecture design of the system provides users with highly flexible functional expansion and customization capabilities. With standardized interfaces and pluggable component design, users can quickly integrate new functions or adjust existing modules according to specific needs, which enhances the flexibility of the system and improves its adaptability to diverse application scenarios, and lays a solid technological foundation for the subsequent iteration and optimization of functions. This design concept enhances the flexibility of the system and improves its adaptability in diverse application scenarios, laying a solid technical foundation for subsequent iteration and optimization, which shown as Table 1:

Table 1. Multi-dimensional comparison table between this system and the framework represented by LangChain

Comparative dimension	This system	Existing frameworks such as LangChain
Visualization capability	Provides an intuitive graphical interface that supports visual construction and management without the need for programming.	Code-based, lack of visualization support
User operating threshold	Low, graphical interface operation-oriented, suitable for non-technical users	High, need to write cues or codes
Intelligent optimization support	Supports real-time feedback and automatic adjustment of inference paths	No built-in optimization suggestion mechanism
Reasoning efficiency and quality control	Provide continuous quality monitoring and dynamic optimization	Relies heavily on manual debugging and feedback
Chain of thought structure support	Supports multi-level, complex logic path construction	Mostly linear or weakly structured paths
System architecture design	Modular architecture, support for feature expansion and component pluggable	More fixed architecture with limited scalability
Interface and adaptability	Standardized interfaces for flexible integration of multiple functional modules	More reliance on specific languages and plug-ins

Effectiveness evaluation: conduct editing tests on logic-intensive datasets to observe their intermediate outputs as well as the overall chain of thought orchestration as a way of demonstrating all the process transparency and intuitive advantages of a visual chain of thought editing system on complex tasks.

Comparison of Ease of Use and User Experience. User operation process: This system simplifies the construction process of chain of thought with the help of intuitive arrangement and observation and clicking operation. Compared with the complex code writing process and parameter configuration process in the traditional method, this system simplifies the construction and optimization process of chain of thought, and with the intuitive graphical interface and intelligent interaction design, the user can quickly complete the construction and adjustment of chain of thought, which greatly reduces the technical threshold and the operation difficulty. This reduces the technical threshold and operation difficulty to a great extent. The efficient operation experience of enhances the working efficiency of users, and also provides a convenient reasoning tool for users without technical background, which expands the application scope of the system which shown as Fig. 5.

From the perspective of interface design, this research adopts the dynamic connection technology based on Leaderline.js and combines it with the latest graphical interface design concepts to build an effective chain of thought visual management task system through color coding system, layout optimization strategy, and interactive element design. This design paradigm combines human computer interaction theories with visualization techniques, which not only improves user efficiency to understand node relationships, but also provides a unified collaboration platform for interdisciplinary researchers through intuitive manipulation mechanisms. The ergonomic design improves the efficiency of building and managing chain of thought.

The Web-based visualized chain of thought system proposed in this study presents comprehensive advantages at multiple levels. The system excels in user interaction design, introduces an intuitive graphical interface and a dynamic interaction mechanism, improves the visual expressiveness of the construction of chain of thought, and reduces the threshold of use by presenting the complex reasoning process in a more intuitive and vivid form, and improves the user experience through the concise and expressive interface design. The simple and expressive interface design improves the user experience, and the smoothness of operation and the immediacy of system

feedback are improved for non-professional users. The method of this study has synergistic advantages in terms of ease of use, versatility and efficiency, and provides strong support for the visualization of complex chain of thought.

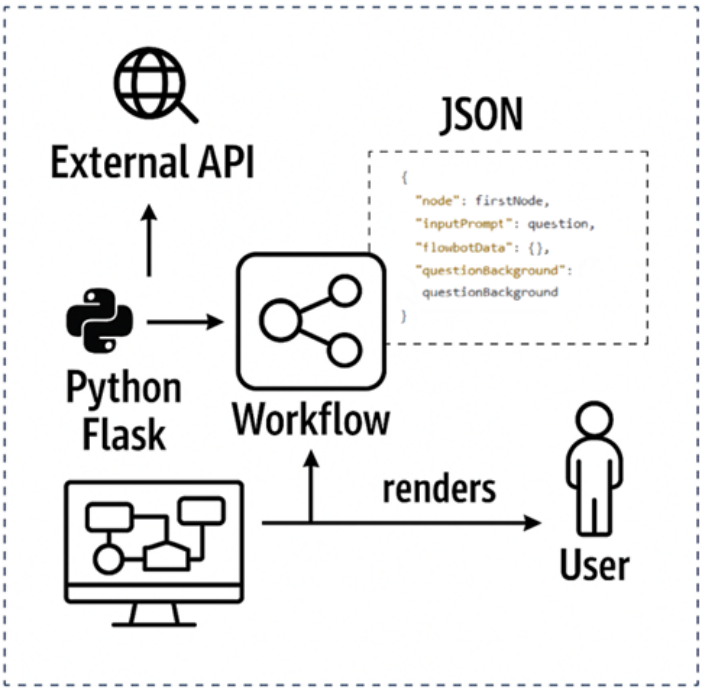

Fig. 5. System working model

Compared with the existing LangChain framework and automated hint generation technology, this system presents advantages in visualization and user interaction experience. While traditional methods mostly rely on programming and complex configuration, this system simplifies the operation process with the help of a graphical interface, which makes it easy for non-technical users to build and manage the chain of thoughts.

This research system builds early chain of thought cues by integrating automated just-in-time generation techniques and uses a visual interface for iterative chain of thought optimization. This not only maintains the efficiency of the automated approach, but also leverages the intuition and flexibility of the visually interactive interface, thereby greatly improving the system's inferential performance and user interaction experience.

3 System Analysis

3.1 Context of the Demand Analysis

The system uses Flowbot as its core and a hierarchical structure (Step) to create an intuitive and simple visual chain of thought management model. Most modern work management systems rely on writing complex code to configure and maintain processes. However, this approach typically faces two major problems in terms of practical use. First, the lack of visual support in building and modifying processes makes it difficult for developers to intuitively understand and manipulate the process. Second, it is difficult to maintain the code itself. As business requirements change or processes become more complex, the chain of thought becomes more complex and error-prone.

The chain of thought management system designed in this study employs the concept of graphical interface design and greatly simplifies the chain of thought process configuration and management process through intuitive clicks. The system supports the flexibility of adding, deleting, modifying, modifying and querying process nodes, and innovatively combines a regular expression matching mechanism with an external service

interface for conditional process control and jumping functionality, which effectively reduces the complexity of system configuration, operation and maintenance costs. This design approach not only improves the visualization of the chain of thought, but also enhances the flexibility and extensibility of the system through a modular architecture, thus allowing users to customize it for various business scenarios.

The goal of the system is to create an automated tool that is efficient, flexible, and easy to maintain, and that deals with some of the common problems in the traditional code-writing chain of thought, such as visualization deficiencies, complexity, and difficulty in maintenance.

3.2 Non-functional Requirements

Availability and Maintenance. From the interface layout and interaction point of view, the system uses bootstrappers and JavaScript in its UI design to ensure a clear and easy-to-use interface with support for interactive operations such as hovering the mouse, zooming in and out and hiding, separating its Frontend and back-end. The RESTFUL API is used for data exchange and the Frontend and back-end modules are independent, making the system easy to extend and maintain. In addition, the system provides logging and error handling. When the process is running, the system outputs detailed logging information, including input and output data, exceptions, etc. to facilitate user monitoring and debugging.

Reliability and Data Consistency. The system uses a MySQL database to provide persistent storage for core data (e.g., nodes and steps) which shown as Fig. 6. The integrity and consistency of the system data is ensured by a strict constraint mechanism that effectively avoids invalid reference problems. A failure-resistant batch processing mechanism is designed into the system for large-scale data processing scenarios. This mechanism not only integrates automatic re-entry, but also uses detailed exception logs to effectively respond to network anomalies, external API failures, and other emergencies, thus ensuring business process continuity. In addition, the system automatically generates data backup files when changing chatbot JSON settings or performing batch operations. This feature not only ensures the security of the original data, but also provides reliable support for version backtracking.

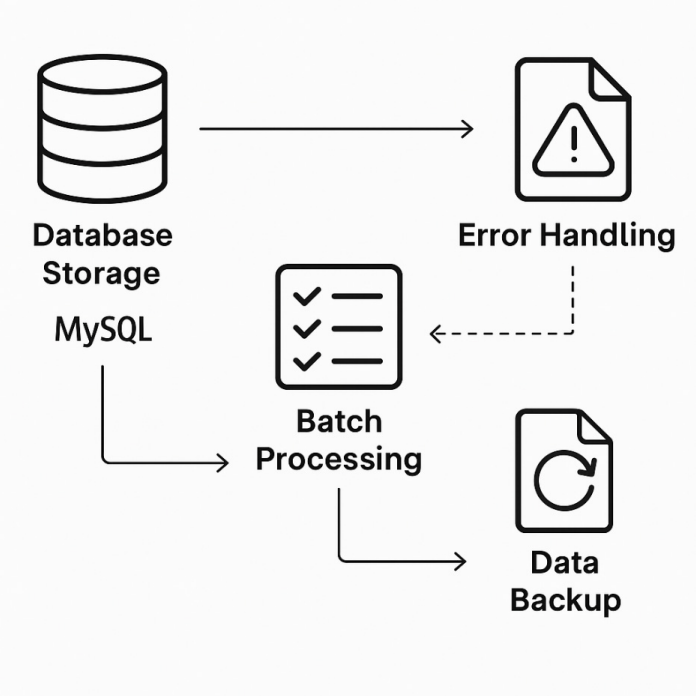

Fig. 6. Storage & Data backup & Batch example

Performance and Scalability. In terms of system performance and scalability design, the system applies a series of strategies to ensure that it still has good responsiveness and maintainability under high load and future functional expansion scenarios. At the interface processing level, the current system relies on synchronized APIs to carry out request and response operations, which can effectively control the process and simplify the logic of small and medium-sized applications. Nevertheless, with the increase in the scale of users and the pressure of concurrent requests, the system can introduce an asynchronous task queue mechanism to achieve asynchronous processing of requests to improve the overall throughput capacity and response efficiency. However, as the user scale increases and the pressure of concurrent requests increases, the system can introduce an asynchronous task queue mechanism to achieve asynchronous processing of requests to improve the overall throughput capacity and response efficiency.

The system uses modular design concepts at the architectural level, maintaining low coupling and clear boundaries between each functional module, whether it is the process engine, visualization Frontend or data storage interface, can be independently upgraded as well as functionality expansion, which improves the scalability of the system, but also beneficial to the long-term maintenance and technological evolution. Such as the integration of new models in the future to invoke the mechanism or to replace the underlying. For example, in the future integration of new model calling mechanisms or replacement of the underlying database, developers do not need to reconfigure the entire system, but only need to directional transformation of the relevant modules on the line.

3.3 User Role Requirements Analysis

The system divides its functionality into three core user groups. Process designers are the main architects of the system, whose responsibilities include planning and configuring the business process logic (including, in particular, node attribute settings and jump rule definitions) to ensure that the process design perfectly meets the business requirements. Typical business users are end users. Emphasis is placed on operational convenience and conducting business activities according to pre-configured processes, mainly without hindering the adaptation of the underlying logic. The chatbot development and maintenance team is responsible for implementing the intelligent interaction features of the system. These tasks include interfacing with external dialogs, managing chatbot parameters and configurations, and continuing to optimize the interactive experience.

Database Overview. A MySQL relational database management system is used as a data storage solution. Its main feature is the structured storage and effective management of process handling data data. The system architecture is centralized in the chain of thought data table. By recording important fields such as flowbotid, step information, node configurations and related attributes, we achieve two design goals: a highly configurable process engine and a maintainable system. The meaning of some fields (column) in the corresponding node execution are shown as Fig. 7.

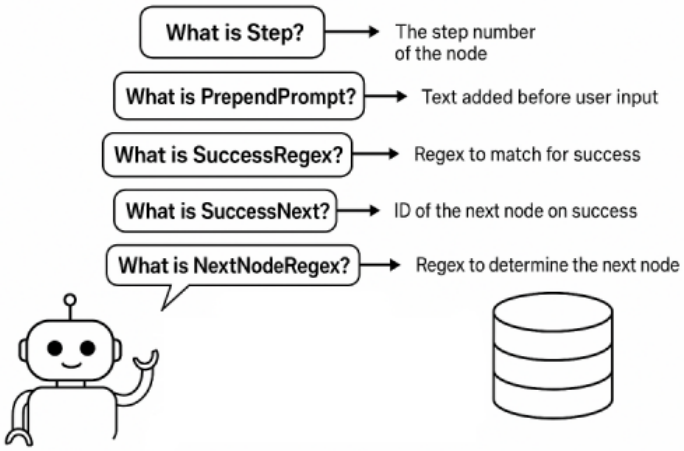

Fig. 7. Meaning of some fields (column) in the corresponding node execution

Data Sheet Design. To support definition, storage, and processing, the system was designed with several core data tables. These include the important metadata table for recording basic information about each chain-thinking linkage, which is structured as Table 2:

Table 2. Workflow table fields and their descriptions

Serial number	Field name	Data type	Austerity	Default value	Clarification
1	FlowbotID	VARCHAR(255)	NOT NULL	not have	Process identifier, unique ID for each process
2	ID (primary key)	INT	primary key, auto_ increment	not have	Uniquely identifies each node
3	Step	INT	NOT NULL	not have	Step number to which the node belongs
4	SuccessRegex	VARCHAR(255)	NULL	NULL	Successful regular expressions
5	FailRegex	VARCHAR(255)	NULL	NULL	Regular expressions that fail to match
6	SuccessNext	TEXT	NULL	NULL	Next node ID after a successful match
7	FailNext	TEXT	NULL	NULL	Next node ID after a failed match
8	ChatbotID	VARCHAR(255)	NOT NULL	not have	Corresponding Dialog Bot ID
9	Retry	INT	NULL	3	Maximum number of retries
10	user_id	INT	NULL	1	User ID
11	public	INT	NULL	0	Whether it is a public process (1=yes, 0=no)
12	activate	INT	NULL	1	Whether the process is enabled (1=yes, 0=no)
13	isEnd	TINYINT(1)	NULL	0	Whether it is an end node (1=yes, 0=no)
14	multiple_run	TINYINT(1)	NULL	0	Whether multiple runs are allowed (1=yes, 0=no)
15	Note	VARCHAR(255)	NULL	NULL	Remarks
16	NextNodeRegex	TEXT	NULL	NULL	Dynamic jump rules stored in JSON format
17	PrependPrompt	VARCHAR(255)	NULL	NULL	Append prompt text before user input
18	AppendAfterAnswer	TEXT	NULL	NULL	Append text after AI output

SQL generation statements are attached as “workflow.sql”.

Data Sheet Design Elements. In the current system, the design of the data table is centered around the flexibility and control of the process nodes, focusing on a number of different dimensions, such as primary key policy, process attribution, logical jumps, execution control, and user permissions, etc. Each node is identified with a self-incrementing primary key ID, which ensures its uniqueness and facilitates subsequent references, and in order to achieve the management of process attribution, the nodes are associated with the corresponding processes by means of a logical foreign key FlowbotID. In order to achieve the management of

process attribution, the nodes are associated with the corresponding processes by means of a logical foreign key FlowbotID. Although foreign key constraints are not mandatory at the database level, this structure provides a clear process organization at the logical level.

This research innovatively introduces Step as a basis for node categorization in process control mechanisms. The domain is dynamic and tunable and can effectively support the customization requirements of complex business processes. The multimodal jump system is constructed at the system architecture level. For a particular implementation, two regular expression modules, SuccessNext and FailRegex, are used to determine the execution state and associate the SuccessNext and FailNext fields to realize the state-driven node jumping functionality.

In order to improve runtime stability and scalability, the table introduces a Retry field that defines the number of retry failures, effectively improving the fault tolerance of the node. The multiple_run parameter supports batch processing of JSON arrays, which is useful for parallel task execution scenarios with structured inputs. In terms of user rights management, the user_id field ensures the identity of the process owner, public controls the public state of the process, and activate marks whether the process is enabled or not, which allows flexible control of the process online or offline without destroying the data.

The system is specially designed to improve the dialogue fields, in which PrependPrompt can automatically append contextual prompts before user input, and AppendAfterAnswer is used to supplement the content of the explanation after the AI response, so as to reach a more complete and humanized interactive experience. The table structure takes into account the flexibility and security, and provides a solid data support for the intelligent and dynamic operation of the process engine. The table structure balances flexibility and security, giving solid data support for the intelligent and dynamic operation of the process engine.

Chatbot Document Management. Chatbot management allows users to manage json files in the Chatbot directory and query, modify and update Chatbot parameters through the backend API. External service invocation and message logging section, the system provides /forward_chat and /api/proxy interfaces to realize interaction with external conversation APIs and record request logs for problem troubleshooting which shown as Fig. 8.

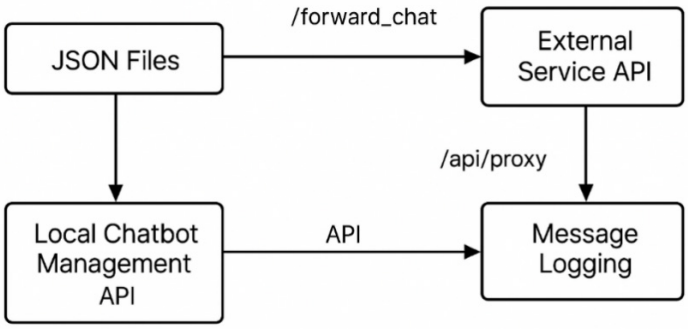

Fig. 8. Log module concept demonstration

Use Case Diagrams. This system is centered around the construction and execution of “Flowbot”, which allows users to select an existing process or create a new process, and add, delete, rearrange and configure the process steps (Steps) with the help of visualization, and each node of the process can be operated directly in the Frontend interface. Each node of the process can be operated directly from the Frontend interface. Users can set the logic of node jumping, change the type of nodes, and choose the execution method, such as single run or batch run, without writing code manually. The system adopts a modular architecture to organize various functions, mainly including process management, step management and node permission editing of the three major categories of

operations. And finally converge to the process editing module, the user with the help of the graphical interface to complete all the interactive behavior, the system will be in the back-end in accordance with the logic of the preset process analysis and execution, for those who need to build a complex chain of logic or multiple rounds of reasoning application scenarios which shown as Fig. 9.

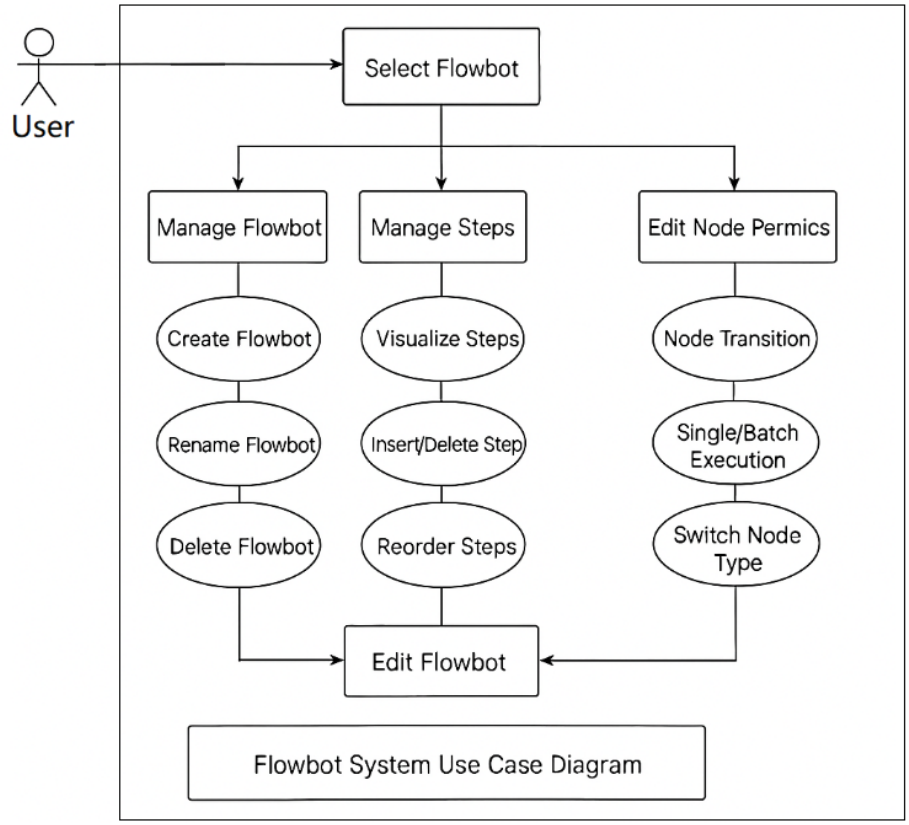

Fig. 9. System use case diagram

4 System Implementation and Validation

4.1 System Architecture and Module Design

Overall System Architecture. The system uses a clear layered architecture, taking the modularity concept as the core, to create a highly maintainable and scalable visualization platform for the chain of thought. In the overall architecture, the Frontend interaction layer acts as an intuitive bridge between the user and the system, with the help of responsive design and dynamic linking mechanism, the user can drag-and-drop the way to define all kinds of process nodes. Set the execution steps and flexibly configure the logic of the jump, which This greatly reduces the difficulty of use for non-program developers. The logic control layer, as the execution core of the system, is responsible for analyzing the flow chart constructed by the user, scheduling the execution of each node according to a fixed order. It also supporting the failure retry strategy and the dynamic invocation of external services, so as to ensure the continuity of the process and robustness of the process operation. The layer decouples the control logic from the front and back ends with the help of the standard RESTful interface, which improves the flexibility of the system.

The data storage layer serves as the basic supporting architecture for system operations, organizing and maintaining different types of information through structured data management mechanisms. The architecture

not only centrally stores and manages core elements such as process metadata, operation logs, user and process rights associations, but also integrates a high-performance vector search module based on the FAISS framework.

In order to meet the diversified needs of functions, the system integrates external service layer and realizes seamless connection with third-party systems with the help of unified proxy interface /api/proxy. So that the chain of thought is not limited to local computation and rule judgments, but also can dynamically access dialogue models, external databases and other capability modules, which expands the boundary of the system's practical applications. The layered design clearly defines the responsibilities of each module and leaves a good expansion channel for future system evolutions. The layered design clearly defines the responsibilities of each module and reserves a good expansion channel for the future evolution of the system. The overall system structure is shown as Fig. 10.

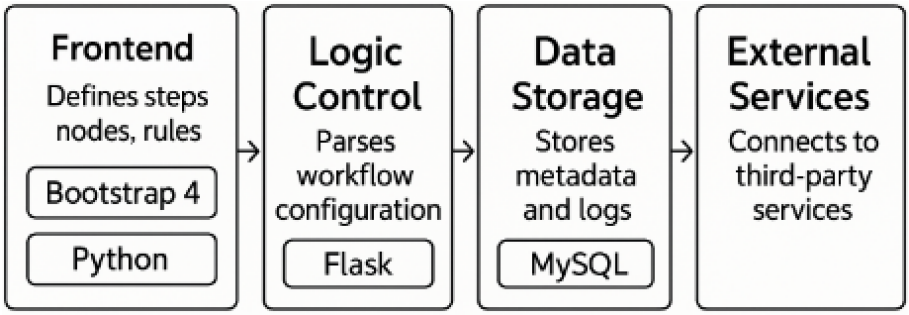

Fig. 10. Overall system structure

Module Interaction Flow. The module interaction process of the system is centered around the entire life cycle of the user's construction and execution of the chain of thought, which can reflect the front and back-end synergy and closed-loop management of data such as the design of a mechanism. Than the user will initially complete the construction of the chain of thought with the help of a visual editor in the Frontend interface, which includes a series of node drag and drop layout, the execution of the rules of the set and the configuration of the jump logic operations. In this process does not require programming background, which largely improves the ease of use and flexibility of the system, when the configuration is completed. The Frontend will submit the entire process structure to the back-end by virtue of the REST interface, and the back-end in the receipt of the configuration information, the node relationship of the chain of thought and control parameters will be analyzed, and will be stored in the database, in order to ensure that the process traceability and persistent management. and persistence management. In the system execution phase, the back-end will gradually traverse the nodes in the flowchart according to the breadth-first search (BFS) strategy, judge the jump paths according to the set regular matching rules, and dynamically call external service interfaces to realize the specific task functions at each step.

By providing real-time feedback of the execution results of each node to the Frontend interface, it can intuitively monitor the operation of the chain of thought. The system also synchronizes and stores the relevant operation data in the database to provide data support for subsequent audit trails, problem troubleshooting, data analysis, and so on. The design of this interaction mechanism fully embodies the clarity of module role division of labor, the timeliness of data synchronization, and the smoothness of user operation, which effectively ensures the high efficiency and stability of system operation.

4.2 Visualization of Chain of thought Construction Implementation

Visual Editor Design. The visual editor is the interactive core part of the system, which is designed to bring users an intuitive, efficient and rule-bound process building experience. The editor presents the process structure graphically, and users can drag nodes to the canvas area at will, and different types of nodes, such as the start node, the end node, the action node, and the conditional judgement node, rely on the icons and the color scheme to distinguish them, which can improve the different types of nodes. Such as start nodes, end nodes, action nodes and conditional judgment nodes, are distinguished by icons and color schemes, which improves recognition and

makes the operation more intuitive. Relying on LeaderLine.js, users can easily establish logical jumps between nodes, and refine the flow direction with the help of binding rule expressions such as regular matching.

In terms of node interaction design, the system supports clicking on a node to expand the attribute panel, and the user can specify external call interfaces for the node like ChatbotID. Than set the number of failures to retry, and edit the input and output parameters, so as to ensure that each node has a clear definition of behavior at the execution level.

To ensure the structural integrity and logical correctness of the process, the system adds various validation mechanisms to the editing process. First, the loop detection feature automatically identifies potential loop-dependent paths to prevent special structures that cause process termination. The system enforces the presence of the initiating node and the presence of the terminating node and checks for consistency in the jump path configuration. This requires all nodes to clearly declare jump rules and set default paths. This gives the build process a closed loop execution function. These designs allow the visual editor to be efficient as the user builds complex processes, reduces the chance of misconfiguration and achieves the right balance between low obstacles and high system reliability.

Example of a Chain-of-Consciousness Configuration. In the system application practice, the chain of thought configuration can be flexibly set according to the specific task type, take the LogiQA-ZH logical reasoning task as an example. The user can quickly construct a reasoning process with semantic understanding and conditional jumping ability by means of the visual editor, which takes the Start node as the entry point, and then connects Question, Parse, Background and End nodes in turn to complete a complete logical judgment task, Background and End nodes to complete a complete logical judgment task.

The process control mechanism is designed to use regular expressions to implement smart jump logic configuration. Specifically, when the user sets the NextNodereGex parameter on a parsing node, the system automatically jumps to the background node if it detects a logical keyword related to “inconsistency” during text parsing. At this point, the system calls on external knowledge-based resources to assist in decision-making. Otherwise, the system continues to run in the preconfigured default process path until the process is terminated. The design not only fully demonstrates the conditional branching capabilities of the system’s processes, but also emphasizes flexibility and adaptability in the processing of complex natural language tasks.

This example illustrates the system’s ability to support dynamic path adjustment specific to task semantics. Combining graph manipulation with potential regular expression mechanisms can greatly improve the efficiency and accuracy of constructing logical reasoning processes.

4.3 Chain of thought Execution Engine Implementation Details

Process Analysis and Execution. The process resolution and execution mechanism is a key part of the system operation, and its design focuses on the organic integration of orderly execution, response flexibility and abnormality tolerance. In terms of the execution logic, the system utilizes the breadth-first search (BFS) algorithm [24] to traverse the nodes in the chain-of-consciousness graph from the Start node, layer by layer, to ensure that the process can be structurally advanced in an orderly manner. When each node is scheduled for execution, it will call the bound logic function or external service, and analyze the returned result in real time, and take this result as the decision basis for process jumping.

The system supports a dynamic jump mechanism based on regular expressions. Each node can be configured with a NextNodeRegex, which is used to do pattern matching on the response content; if the match is successful, then it will jump to the target node according to the corresponding rules; if the match is unsuccessful, then it will continue to be executed according to the default path. This mechanism gives the process a stronger ability to adapt to the context, so that the process can dynamically adjust the execution path according to the specific content.

In order to ensure system robustness in complex application scenarios, a fully fault-tolerant mechanism is designed in this study. Each running node can be configured with a retry parameter for the number of failed retry times. If the retry time runs out and the execution fails, the system automatically marks the node state on failure and logs detailed runtime errors for troubleshooting. The thought rollback mechanism offers a more flexible solution for abnormal reasoning recovery [25]. The system achieves real-time persistence in the running state. By synchronizing the writing of important data, such as the current execution progress and the context of the database, it not only ensures process recovery after abnormal interruptions, but also provides technical support

for breakpoint interruptions and full traceability, which greatly improves system reliability and maintainability. At the framework design level, the system adopts an innovative symmetric design concept. The visualization based on the BFS traversal algorithm ensures the consistency of the visual representation and the process in the actual execution. This chain of thought design adapts to human cognitive habits and supports prospective execution and backtracking, making it useful for debugging the system.

External Service Integration. An intelligent chain of thought execution mechanism based on the integration of external services. Through a standardized interface invocation methodology, the system greatly enhances the reasoning and interaction capabilities of complex scenarios, making it particularly suitable for tasks such as natural language processing, knowledge search and numerical computation. At the technical implementation level, the system uses Chatbotid as its core service identification mechanism. Each process node is functionally bound by configuring a specific chatbotid. When the chain of thought engine is scheduled to the node, the system automatically completes the required encapsulation and sends the incoming data, thus ensuring a seamless connection with the pre-configured language model or inference engine API, which shown as Fig. 11.

The response result returned by the service will become the output content of the node, which will be recorded and used for the determination of the subsequent process jump rules. For example, when a node needs to generate an answer, extract keywords or determine the semantic conflict, the system can call the large language model with relevant capabilities, and the large language model will generate the processing results and feedback to the chain of thought environment. Multi-hop reasoning accuracy can be further enhanced by integrating external knowledge graphs into the inference flow [26]. This mechanism not only supports mainstream model services such as OpenAI, Wisdom Spectrum, Deepseek, etc., but also can be docked with privately deployed models or third-party inference engines by virtue of a unified/api/proxy interface, thus realizing pluggable access to model capabilities.

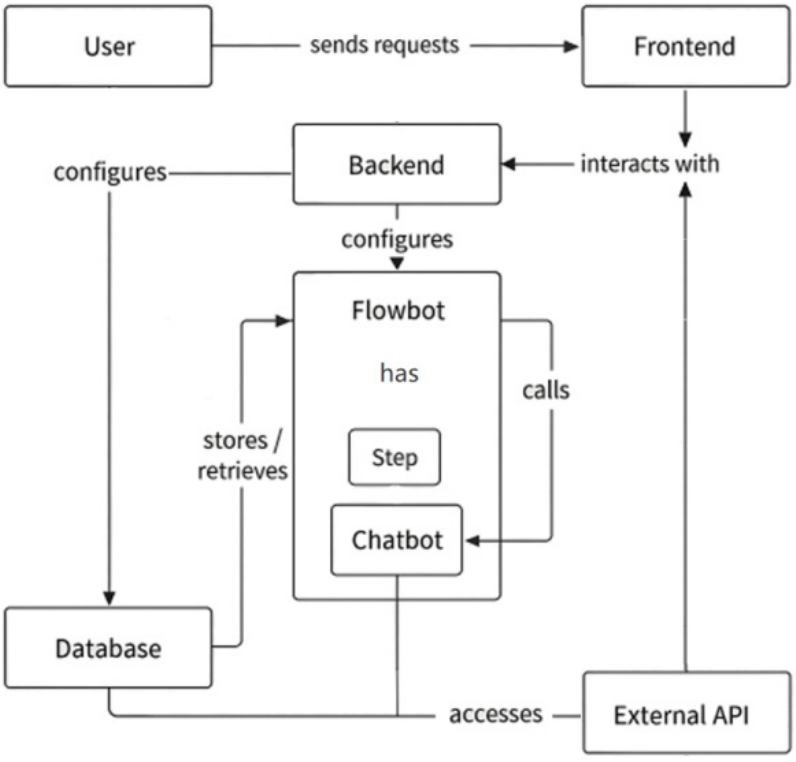

Fig. 11. Calls to the API in the complete chain of thought process

Prompt Splice Example. In order to systematically present the behavioral mechanisms of the rapid clipping mechanism during the execution phase of the process, this study selects classical analytical examples and describes in particular the important domains involved in the construction of dynamic prompts in a specific context, such as prependPrompt, inputPrompt, background, and key node outputs. The aim of this work is to demonstrate how optimized prompts can be created through a collaborative mechanism of configuration parameters and user inputs by tagging the data source and semantic properties of each component. In the case of the writing assistance bot, when the large language model is invoked at the end of the Flowbot chain of thought process. The system integrates contextual information into the knowledge base, fully prompting the character with preset prompts, historical dialog outputs as well as the current user inputs and other multi-source data, which is ultimately sent to the model.

[PrependPrompt] (fixed configuration from the current Node to set roles and tasks for the Chatbot)

You are a high school language teacher with many years of teaching experience. Please help the student analyze the essay topic given below from various perspectives, such as writing structure, depth of intention, language expression and presentation of theme, and provide as specific and actionable suggestions as possible, so that the student can write a high-scoring essay in the exam.

[Previous Node Output] (from the previous Node execution result, automatically injected by the system as a contextual reference)

The previous analysis pointed out that the essay topic emphasizes the philosophical theme of "thinking differently", and suggested the use of a parallel structure, starting from the three perspectives of "understanding others", "putting oneself in their shoes" and "empathy and sympathy". It is recommended to use the parallel structure from the three perspectives of "understanding others", "putting yourself in their shoes", "empathy and sympathy", and to use concrete examples from real life to enhance the persuasive power. At the same time, attention should be paid to the truthfulness and infectiousness of the language to reflect the depth of emotion.

[inputPrompt] (from this question entered by the user on the front end)

The essay topic is: "Putting Yourself in Someone Else's Perspective." Please help me analyze what optional writing directions are available for this topic and what styles are appropriate? How should the beginning and ending be designed? How should the content of the middle paragraphs be organized?

[questionBackground] (additional context from the user to improve personalization and precision of the response)

I'm a senior in high school, my language foundation is okay, and my essays are usually in the upper-middle level of my class. The goal is to participate in a provincial writing competition, and I hope to write a thoughtful and compelling argumentative essay. The topic is not limited to the style of writing, but I prefer the argumentative style and want to make a breakthrough in terms of depth of thought and logical rigor.

4.4 Visualization of the Chain of thought System Architecture Design

The core functions of the system can be summarized in the following Fig. 12 categories:

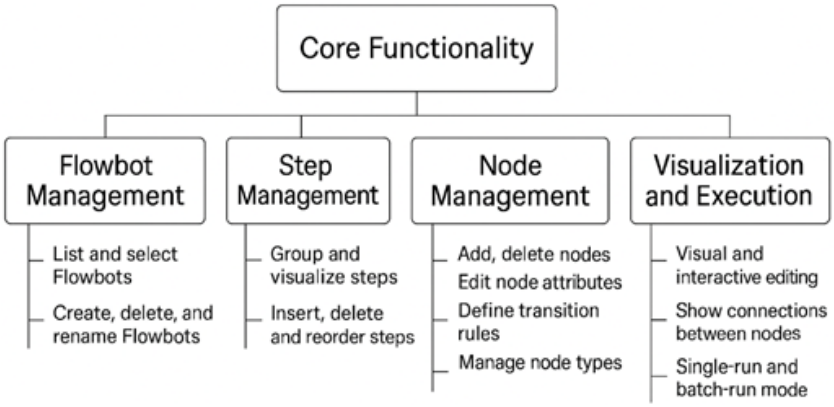

Fig. 12. System architecture diagram

Flowbot Management Function Description. Flowbot list and selection function, the system provides an interface to query all FlowbotIDs in the database, and the user can select the flow that needs to be operated from the drop-down menu for subsequent management.

Create and Delete Flowbot function, allows users to create a new named Flowbot, the system will automatically create the initial node in the database. If the Flowbot is no longer needed, the user can delete the process and all its associated data to ensure the tidiness of the database.

Rename Flowbot function, through the back-end interface support for naming adjustments to the created Flowbot, to optimize process management, to avoid name confusion.

Step Management Description. In the step grouping and visualization of this link, the system will be based on the step number for all the nodes to carry out the sorting operation, and then in the front end of the visual form to give the presentation. So as to ensure that the user can rely on the intuitive way to view and manage the process of the various steps which shown as Fig. 13.

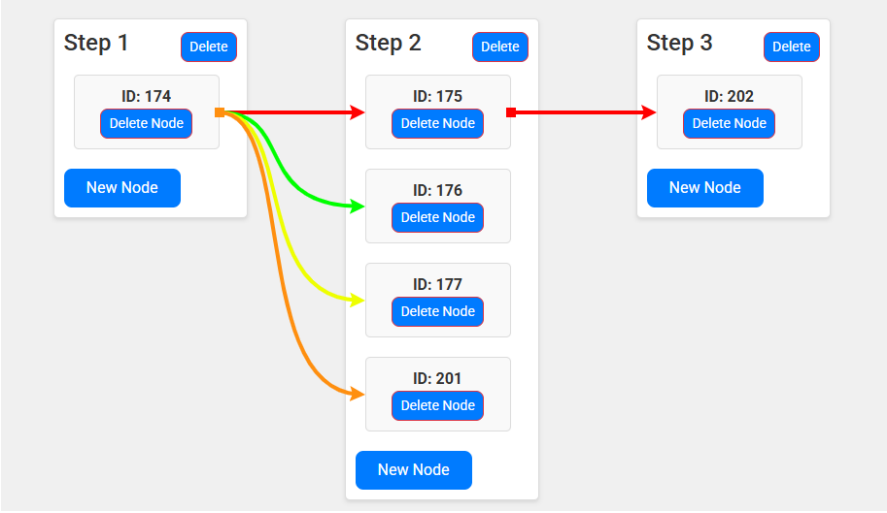

Fig. 13. Step management runtime screenshot

As for the Insert and Remove Steps section, it allows users to add or remove steps in a given Flowbot, and automatically updates the step numbers to ensure the integrity and logical consistency of the process which shown as Fig. 14.

In step rearrangement, when a step is deleted, the system can automatically adjust the references to the related steps to avoid invalid nodes or logical errors, and ensure that the process can be executed correctly.

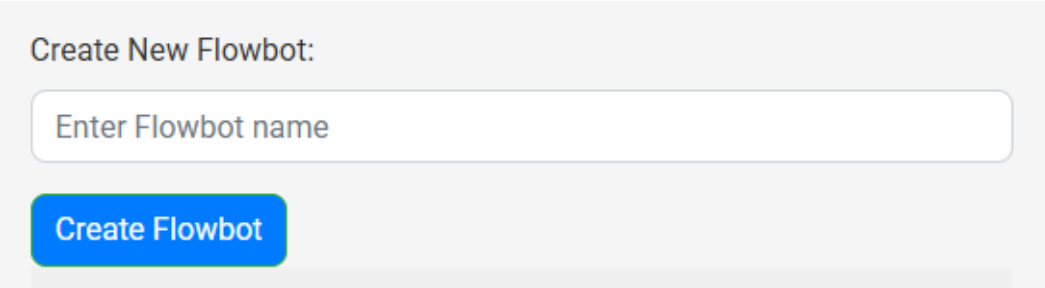

Fig. 14. Screenshot of the Create Chain of thought runtime

Node Management. During process design, users can dynamically add new nodes to specific steps and configure important parameters for these nodes (e.g., Chatbotid, retry, Successregex, Failregex, etc.) to dynamically control the node's business logic. When you perform a node deletion operation, the system automatically handles the associated reference relationships to ensure that the entire process is topologically complete. Each node is bound to a unique chatbot instance. Essentially, it represents an independent, intelligent conversational agent with a base layer that is flexible enough to plug into a variety of large-scale language modeling architectures, such as the DeepSeek family and ChatGPT. in addition to basic language processing functionality, each chatbot instance can also be associated with multiple custom knowledge bases. The design not only preserves general interaction features, but also significantly improves the response performance in specific scenarios by integrating domain knowledge.

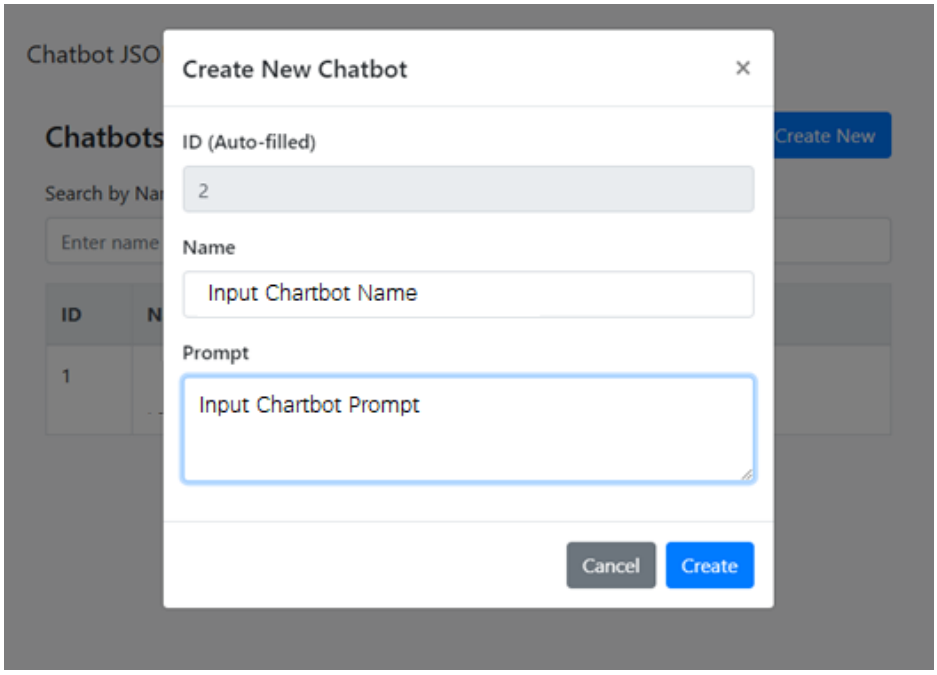

Fig. 15. Chatbot edit page

Node editing and attribute configuration allows users to modify node attributes with the help of the Frontend interface. These attributes cover Note, PrependPrompt and NextNodeRegex, etc., which can be adapted to different application scenarios, and the jump rule is the NextNodeRegex, which can be used to define the regular matching rules, and to set the corresponding target for each branch. In the process of system execution, it will decide the jump path based on the matching result. The system supports a variety of node types, such as None, Start, Pass, End, Background, Parse, and Question, to adapt to different task requirements, and can also connect to external dialog interfaces with the ChatbotID. (Shown as Fig. 15.)

Flowbot Editing Functions. In practice, flowbot contains several Nodes, and users can edit each Node in an intuitive way to fit their needs. This includes setting judgment conditions to determine which subsequent node the flow will go to. The visual way to define the connection relationship since makes the whole process design more intuitive and easy to maintain.

The system provides rapid content customization and knowledge-based association capabilities for each node, which significantly improves the level of intelligent response for situational awareness. Based on this visually interactive interface, developers can efficiently build multi-layered interaction logic and business process architecture. This not only optimizes development efficiency, but also significantly improves system management and operational control. (Shown as Fig. 16.)

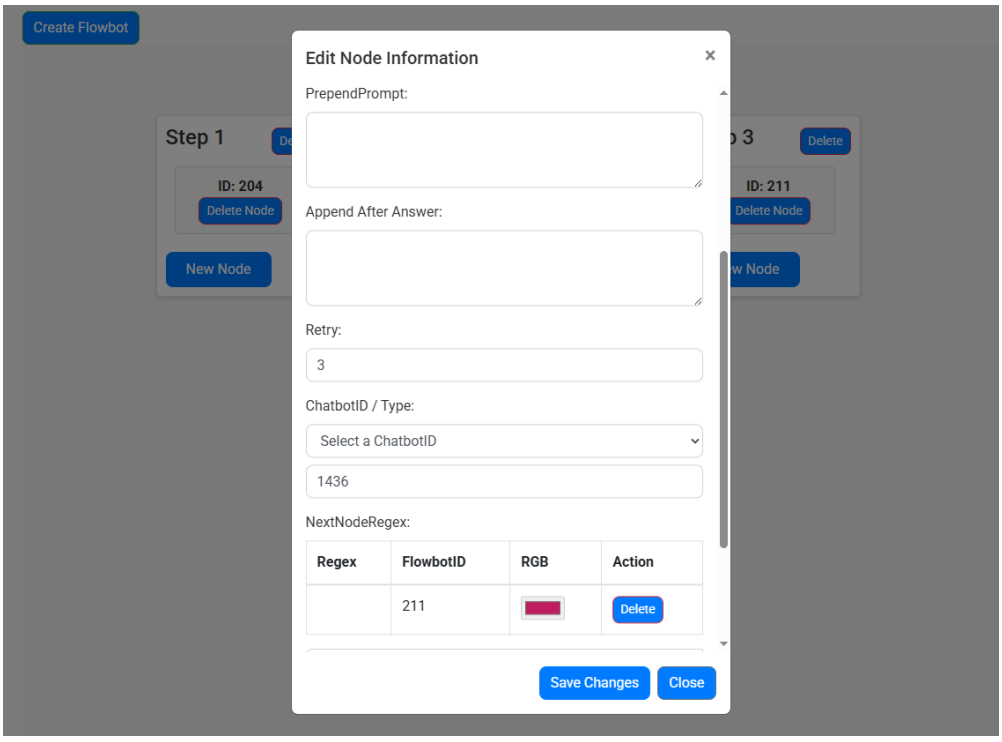

Fig. 16. Node edit page

Frontend Visualization and Operation. This system is a visual reasoning platform for the construction and execution of complex task processes, with the following core features:

1. With visual orchestration and zoom features, users can intuitively manage steps and nodes in the process, improve ease of operation, view node connections, dynamically draw connections between nodes using lines, and differentiate different jump paths using colors for easy readability.
2. Visual orchestration and zoom features allow users to intuitively manage steps and nodes in the process for improved ease of operation.
3. Node connection displays, which use readers to dynamically draw connections between nodes, thus distinguishing different jump paths and thus making them easier to read.

4. Interactive Editing. The Frontend interface allows users to directly change, delete or adjust jump settings to nodes without manually editing the code.
5. During a single run, as the user enters questions and background information, the system sequentially executes the process according to the BFS logic and continues until the end node is reached.
6. Batch operation: supports batch processing of .xlsx files. The system runs the process according to the Excel data line, and outputs the result to a new file. Suitable for large-scale data processing programs.

5 System Testing

5.1 Testing Methodology and Environment

This chapter focuses on system stability validation, from the setting of test objectives, the determination of execution methods to the evaluation of the results, we have carefully constructed a set of complete and suitable for the real use of the test process. The main purpose of the test work is to clarify the stability and reliability of the system in regular use, in the high-frequency scenarios. Such as large-model interactions, complex process orchestration, and batch processing, whether it can always maintain normal functionality, timely response, and no abnormalities in operation. In high frequency scenarios, such as large model interaction and complex process orchestration and batch processing. The system can always maintain normal function, timely response and no abnormal operation.

The entire testing process was carried out in a standardized hardware and software environment, based on a Windows 10 system and typical mid-range and high-end configurations. So as to ensure that the evaluation results are representative and reproducible, and the core functions of the system were tested in real conditions, including the creation, editing, saving and execution of the chain of thought, and the interface requests were initiated by means of Python scripts, and key data and status feedback were recorded in real time by means of the logging module, so as to achieve a complete observation of the system operation process and behavioral reproduction. With Python scripts to initiate interface requests and the logging module to record key data and status feedback, we can achieve a complete observation of the system's operation process and restore its behavior.

5.2 Test Program

The basic functionality test focuses on the operation processes that are most frequently used by users in their daily life, including the creation, editing, saving and deletion of chain of thought to ensure that the system will not be blocked or logically incorrect in the course of regular use. Subsequently, by running a variety of different types of chain of thought instances, the system is verified in terms of coordination and accuracy at the level of the execution nodes, the jump logic, and the call of external services. Subsequently, by running many different types of chain of thought instances, we verify the coordination and accuracy of the system in terms of execution nodes, jump logic, and external service invocation, etc., to ensure that the system has the ability to adapt to many different tasks.

Exception handling tests are conducted by actively constructing error scenarios, simulating typical misoperation situations such as users missing parameters, processes not configured with jump logic. Or the existence of broken nodes, etc., and observing whether the system can recognize the problem at the first time and make accurate prompts, verifying its robustness and friendliness to the user guidance, and the stability test adopts a long time and continuous operation observation to monitor whether the system may crash or lag without intervention. Stability testing uses long-time continuous operation to monitor whether the system would likely to crash, memory leakage, or task lag under the state of no intervention.

Some of the typical test scenarios are: trying to run the process without filling in the Prompt or without uploading the pending file, clicking directly run without selecting any chain of thought. Than running the process of the logical loopholes, the system in these scenarios should show a clear rejection mechanism as well as the error feedback, to prevent irregularities in the operation due to irregularities in the input.

In order to detect the performance of the system under high load, a special test for model response accuracy and server stress tolerance is also designed. Where the number of concurrent requests is continuously increased on the LogiQA-ZH dataset, which is used to evaluate whether the response accuracy of the language model will fluctuate significantly due to the rise in stress.

The concurrency test script is designed to stress the back-end interface with a request frequency of 10 requests/sec to 100 requests/sec. Testing and comprehensively evaluates the system load performance from multiple metrics dimensions such as response latency, timeout rate and system resource consumption. The test plan comprehensively examined the system's functional integrity, operational stability, user interaction experience, basic performance metrics, and employed a multi-layered testing strategy to establish a reliable quality assurance system for formal deployment and online operation. Detailed records of each test case, expected outcomes and actual performance are summarized in Table 3.

5.3 Test Results

The experimental data show that the large-scale visual orchestration system for modeled chain of thought has excellent operational stability and logical processing capabilities, and the exception handling mechanism performs well, maintaining reliable system performance even under high load conditions.

Table 3. Test results

Test No.	Test item	Test scenario	Description of the test operation	Expected results	Actual result
TC-001	Parameter calibration	Prompt or Excel file not entered	Running without a Prompt or without uploading a file	Decline to run	Pass (a bill or inspection etc)
TC-002	Process selection calibration	Unchecked chain of thought	Click Run directly without selecting a process	Decline to run	Pass (a bill or inspection etc)
TC-003	Logical structure checking	There is a logical error in the chain of thought	Running a process that contains a logic error [27]	Logic error detected, execution denied	Pass (a bill or inspection etc)
TC-004	Model accuracy testing	Impact of high concurrency on model accuracy	Improve concurrency test model accuracy	Stable accuracy with no significant decline	Pass (a bill or inspection etc)
TC-005	Interface concurrency testing	Server API access stress testing	High concurrency testing API response performance	Stable system with no timeout crashes	High concurrency with occasional delays

In such basic exception scenarios as missing input or incomplete process selection, the system can respond in time. If no Prompt is entered or no pending file is uploaded, the system can accurately reject the process execution and rely on the interface to clearly prompt the user to prevent null values from triggering runtime errors, and when the user clicks on "When the user clicks "Run" without selecting any chain of thought, the system will also trigger the interception mechanism, and at the same time, it will give a guided prompt to ensure that the user's operation process has a clear direction.

In the face of more complex logic error scenarios, such as the existence of broken links or undefined jump path nodes in the flowchart, the system can accurately identify the error points before execution and refuse to run, but also with detailed feedback to help the user locate the problem, so as to facilitate the subsequent repair and debugging, this mechanism fully demonstrates the system's sensitivity to the correctness of the chain of thought structure and the ability to make intelligent judgments.

In order to evaluate the performance of the system in a highly concurrent scenario, a real application load environment is simulated with a request rate that gradually increases from 10 requests per second for the stress test requests to 100 requests per second. As Fig. 17: The experimental results show that even if the number of concurrent requests continues to increase, the system scheduling mechanism can maintain a stable working state by monitoring the response performance of the model in the Logiqa-ZH test set. The accuracy of the model output varies slightly under high concurrency. Boosting-of-Thought trial-and-error frameworks can strengthen the robustness of inference under heavy request loads [28]. This phenomenon fully demonstrates that the system has excellent stable performance and balanced scheduling characteristics in the dynamic allocation of computational resources and model call management.

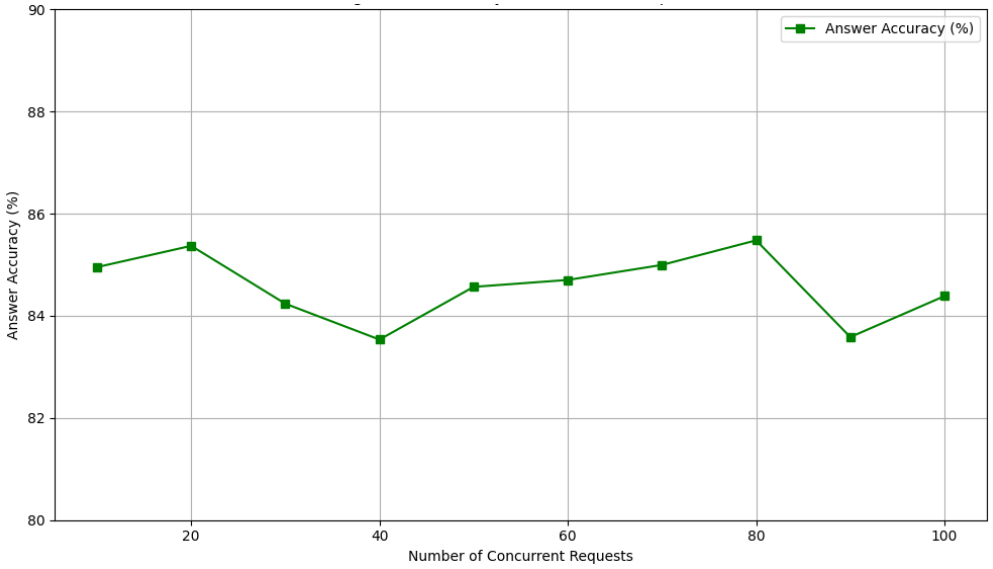

Fig. 17. Model Q&A question set accuracy as the number of concurrent requests rises

Tests of server response performance confirmed the system's ability to withstand stress. The test was conducted for different request densities of API calls, and throughout the test, there were no timeouts, interruptions, or service crashes, and the system's response speed remained smooth overall, with only slight delays in the face of extremely high concurrency transient peaks, but this phenomenon did not have an impact on the overall usability of the system. However, this phenomenon did not affect the overall availability of the system.

A comprehensive analysis of the experimental data shows that the system exhibits system robustness and internal consistency characteristics when dealing with various operational scenarios. The accuracy of the basic interaction mechanism was not only verified, but also fully reflected the controllability of complex business processes. When concurrent accesses were high, the responsiveness and accuracy of the model interface results met the requirements of real applications. These performance metrics provide a solid foundation for commercial deployment and long-term stable operation of the system at scale.

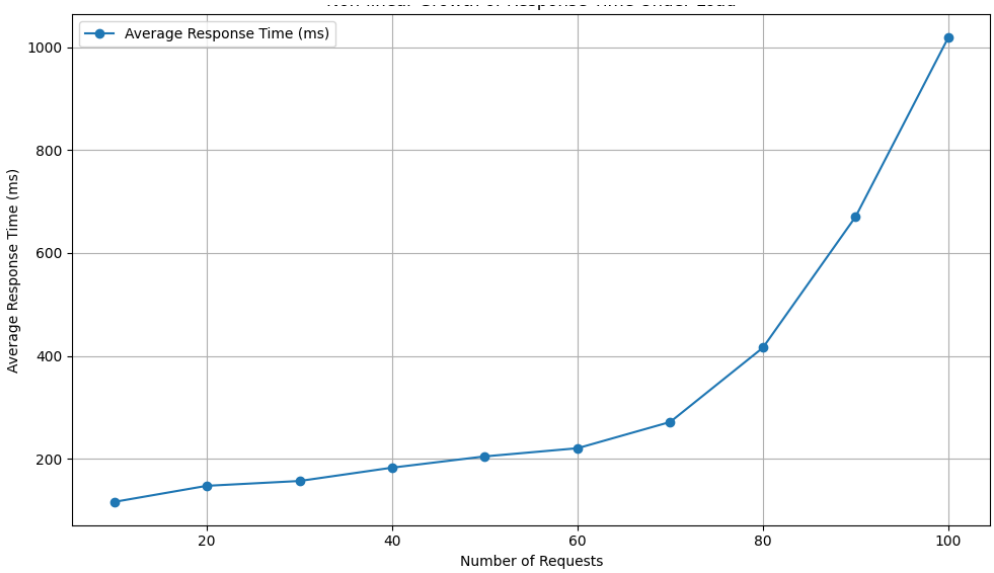

Fig. 18. Average response time per request when the number of concurrent requests rises

As shown in Fig. 18, when the number of concurrent requests exceeds 70, the average response time of the system shows a significant increase. This test was conducted in a Tesla V100 SMX 16G hardware environment, using Ollama as the local inference framework. With the sharp increase in the number of requests, too many concurrent responses take up a large amount of video memory and computational resources, making it difficult for the local runtime environment to adequately schedule the processing within a short period of time, which leads to a certain degree of decline in response speed. This performance degradation is not due to system architectural flaws, but rather to local resource bottlenecks.

Debugging Class Functions. During operation, the Flowbot system continuously records logs of prompt-related operations. These log data contain all important information requested by the user, including core parameters such as the input prompt text, response delay time and model output content. Based on these detailed log records, developers can perfectly replicate the entire process of human computer interaction. (Shown as Fig. 19.)

Prompt: .
Your output should contain only one option letter and should not contain any redundant parts.
Sample output:
A
B
C
D[Answer the question] passage: There are five teams participating in the competition. Regarding the results of the competition, the audience has the following opinions: (1) The champion is either Shannan or Jiangbei. (2) The champion is neither Shanbei nor Jiangnan. (3) The champion is Jiangnan. (4) The champion is not Shannan.|question: The results of the competition show that only one opinion is correct. Which team won the championship?|options: ['(A) Shannan', '(B) Jiangnan', '(C) Shanbei', '(D) Jiangbei']

ChatbotID: 1
Prompt: .
Your output should contain only one option letter and should not contain any redundant parts.
Sample output:
A
B
C
D[Answer the question] passage: Compared with small and medium-sized cities, especially small towns, large cities have higher living costs, which inevitably restricts the entry of rural population. Therefore, urbanization cannot be achieved by developing large cities alone|question: Which of the following conclusions must be assumed|options: ['(A) Urbanization is the only way for my country's development', '(B) Simply developing large cities is not conducive to the promotion of urbanization', '(C) To achieve urbanization, cities must fully absorb rural population', '(D) Large cities are obviously less attractive to rural population than small and medium-sized cities']

Fig. 19. Partial log content

When a user uploads an Excel file as input, Flowbot automatically recognizes and extracts each piece of data in the file as a separate input entry in a batch process. The system executes the configuration flow logic for each row in turn, summarizes and outputs all the results of the execution, and then outputs them to a new Excel file. (Shown as Fig. 20.)

	A	B	C	D
1	passage	question	options	Answer1
2	A certain township is planning a new di	Based on the above statements, which	['(A) The Citizen Par	A
3	The company sent three young staff mem	So who are the three young people or	['(A) 20-year-old acc	A
4	In a Chinese medicine preparation, at l	Based on the above statements, which	['(A) No Codonopsis',	C
5	In recent years, the postgraduate entra	Which of the following best strengthe	['(A) If you take an	D
6	A certain unit conducted an annual eval	Based on the above statements, we ca	['(A) Jia. ', '(B) Yi	C
7	Zhang Ming, Li Ying, Wang Jia and Chen	Based on the above statements, which	['(A) Chen Rui can't	A
8	The head of the relevant department sai	Which of the following, if true, wou	['(A) The new houses	D
9	There are five teams participating in t	The results of the competition show	['(A) Shannan', '(B)	D
10	Compared with small and medium-sized ci	Which of the following is an assumpt	['(A) Urbanization is	C

Fig. 20. Example of batch output content

Verification by virtue of testing reveals that the various types of operation paths covered are completed successfully.

The system performance of key links such as task schedules, process jumps and external service calls meets the expected standards. Meanwhile, the logging mechanism works well, and the log data generated throughout the process is complete and contains no error messages, providing traceability support for the system's operational processes, further proving the system's reliability in terms of stability and observability.

6 Conclusion

6.1 Summary of the Study

This research focuses on the task orchestration and process scheduling requirements of large-scale model application scenarios. A Web-based visual Workflow management system focuses on addressing the shortcomings of traditional process construction methods in terms of visibility, flexibility and intelligent integration. The tight combination of graphical manipulation and modular architecture effectively reduces the difficulty of understanding and configuring complex processes. The key findings of this research are as follows.

The study develops a Web-based visualized CoT management system to address the visibility, flexibility, and intelligent integration shortcomings of traditional processes. Core achievements include: 1) A drag-and-drop interactive editor (powered by LeaderLine.js) enabling code-free process organization, with loop detection and node verification ensuring logical integrity; 2) A BFS-based execution engine supporting complex conditional branching and dynamic path adjustment, enhanced by node-level failure retry and real-time state persistence for robust operation; 3) Integration of the symmetry concept into architecture, logic control, and user interaction—aligning visual presentation with execution processes to improve usability. Application verification confirms the system's ability to decompose complex tasks into controllable steps, enhancing transparency and traceability, while stably supporting batch processing in real-world scenarios.

6.2 Main Contributions

This research has gained substantial results in system design and practical application, and its main contributions can be summarized into the following two aspects. In terms of technological innovation, this system is the first one to integrate the regular expression-based jump rule engine, that is, NextNodeRegex, with the visual process editor, which breaks through the limitations of the traditional static configuration of the chain of thought, and allows users to flexibly define complex conditional branches to achieve the configurable execution of logic-intensive tasks in the graphical interface. It allows users to flexibly define complex conditional branches in a graphical interface and achieve configurable execution of logic-intensive tasks. The system utilizes a modularized architecture design with decoupled front and back ends to improve the scalability of the system. Users can quickly access various external services, such as large-scale language models, structured database query interfaces, etc., based on unified API interfaces, which meets the dynamic integration needs of diversified intelligent tasks.

From an application-value perspective, the platform provides a set of low-threshold and flexible thought-chain process construction and execution tools for a non-technical user group. Users can accomplish graphical configuration of task processes and set up execution logic without learning programming skills. These can be widely used in many fields such as educational counseling, medical counseling, and financial quizzing. In addition, the system enables excellent engineering reusability, and the complete technical evolution from basic CoT reasoning to autonomous language agents is elaborated in [29].

References

[1] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, D. Zhou, Chain-of-thought prompting elicits reasoning in large language models, in: Proc. NIPS'22: Proceedings of the 36th International Conference on Neural Information Processing Systems, 2022.
https://openreview.net/forum?id=_VjQlMeSB_J

[2] Z. Bi, D. Hajialigol, Z. Sun, J. Hao, X. Wang, STOC-TOT: Stochastic Tree-of-Thought with Constrained Decoding for Complex Reasoning in Multi-Hop Question Answering. <https://arxiv.org/abs/2407.03687>, 2024.
[3] Z. Zhang, A. Zhang, M. Li, A. Smola, Automatic chain of thought prompting in Large Language Models. <https://arxiv.org/abs/2210.03493>, 2022.
[4] S. Yao, D. Yu, J. Zhao, I. Shafran, T.L. Griffiths, Y. Cao, K. Narasimhan, Tree of Thoughts: Deliberate Problem Solving with Large Language Models. <https://arxiv.org/abs/2305.10601>, 2023.
[5] Z. Sun, N. Deng, H. Yu, J. You, Table as Thought: Exploring Structured Thoughts in LLM Reasoning. <https://arxiv.org/abs/2501.02152>, 2025.
[6] S. Vatsal, H. Dubey, A survey of prompt engineering methods in large language models for different NLP tasks. <https://arxiv.org/abs/2407.12994>, 2024.
[7] S. Mo, M. Xin, Tree of Uncertain Thoughts Reasoning for Large Language Models. <https://arxiv.org/abs/2309.07694>, 2023.
[8] J. Long, Large Language Model Guided Tree-of-Thought. < https://arxiv.org/abs/2305.08291>, 2023.
[9] Y. Deng, Y. Choi, S. Shieber, From Explicit CoT to Implicit CoT: Learning to Internalize CoT Step by Step. <https://arxiv.org/abs/2405.14838>, 2024.
[10] X. Zhang, D. Ding, Supervised chain of thought. <https://arxiv.org/abs/2410.14198>, 2024.
[11] T. Kojima, S.S. Gu, M. Reid, Y. Matsuo, Y. Iwasawa, Large language models are zero-shot reasoners. <https://arxiv.org/abs/2205.11916>, 2023.
[12] A. Kong, S. Zhao, H. Chen, Q. Li, Y. Qin, R. Sun, X. Zhou, E. Wang, X. Dong, Better zero-shot reasoning with role-play prompting. <https://arxiv.org/abs/2308.07702>, 2024.
[13] Z. Li, H. Liu, D. Zhou, T. Ma, Chain of thought empowers transformers to solve inherently serial problems. <https://arxiv.org/abs/2402.12875>, 2024.
[14] Z. Yu, L. He, Z. Wu, X. Dai, J. Chen, Towards better chain-of-thought prompting strategies: a survey. <https://arxiv.org/abs/2310.04959>, 2023.
[15] H. Lightman, V. Kosaraju, Y. Burda, H. Edwards, B. Baker, T. Lee, J. Leike, J. Schulman, I. Sutskever, K. Cobbe, Let's Verify Step by Step. <https://arxiv.org/abs/2305.20050>, 2023.
[16] C. Zheng, Z. Liu, E. Xie, Z. Li, Y. Li, Progressive-hint prompting improves reasoning in large language models. <https://arxiv.org/abs/2304.09797>, 2024.
[17] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, Y. Cao, ReAct: Synergizing reasoning and acting in language models. <https://arxiv.org/abs/2210.03629>, 2023.
[18] Z. Chu, J. Chen, Q. Chen, W. Yu, T. He, H. Wang, W. Peng, M. Liu, B. Qin, T. Liu, Navigate through enigmatic labyrinth: a survey of chain of thought reasoning. <https://arxiv.org/abs/2309.15402>, 2024.
[19] L. Yang, Z. Yu, T. Zhang, S. Cao, M. Xu, W. Zhang, J.E. Gonzalez, B. Cui, Buffer of thoughts: Thought-augmented reasoning with large language models. <https://arxiv.org/abs/2406.04271>, 2024.
[20] L. Ranaldi, G. Pucci, F. Ranaldi, E.S. Ruzzetti, F.M. Zanzotto, Empowering Multi-step Reasoning across Languages via Tree-of-Thoughts. <https://arxiv.org/abs/2311.08097>, 2024.
[21] X. Chen, Z. Sun, W. Guo, M. Zhang, Y. Chen, Y. Sun, H. Su, Y. Pan, D. Klakow, W. Li, X. Shen, Unveiling the key factors for distilling chain-of-thought reasoning. <https://arxiv.org/abs/2502.18001>, 2025.
[22] S. Qiao, Y. Ou, N. Zhang, X. Chen, Y. Yao, S. Deng, C. Tan, F. Huang, H. Chen, Reasoning with language model prompting: a survey. <https://arxiv.org/abs/2212.09597>, 2023.
[23] A. Madaan, N. Tandon, P. Gupta, S. Hallinan, L. Gao, S. Wiegreffe, U. Alon, N. Dziri, S. Prabhumoye, Y. Yang, S. Gupta, B.P. Majumder, K. Hermann, S. Welleck, A. Yazdanbakhsh, P. Clark, Self-Refine: Iterative Refinement with Self-Feedback. <https://arxiv.org/abs/2303.17651>, 2023.
[24] A. Chari, A. Tiwari, R. Lian, S. Reddy, B. Zhou, Pheromone-based Learning of Optimal Reasoning Paths. <https://arxiv.org/abs/2501.19278>, 2025.
[25] S. Chen, B. Li, Toward Adaptive Reasoning in Large Language Models with Thought Rollback. <https://arxiv.org/abs/2412.19707>, 2024.
[26] X. Tan, X. Wang, Q. Liu, X. Xu, X. Yuan, W. Zhang, Paths-over-graph: Knowledge graph empowered large language model reasoning. <https://arxiv.org/abs/2410.14211>, 2025.
[27] T. Liu, W. Xu, W. Huang, Y. Zeng, J. Wang, X. Wang, H. Yang, J. Li, Logic-of-Thought: Injecting Logic into Contexts for Full Reasoning in Large Language Models. <https://arxiv.org/abs/2409.17539>, 2025.
[28] S. Chen, B. Li, D. Niu, Boosting of Thoughts: Trial-and-Error Problem Solving with Large Language Models. <https://arxiv.org/abs/2402.11140>, 2025.
[29] Z. Zhang, Y. Yao, A. Zhang, X. Tang, X. Ma, Z. He, Y. Wang, M. Gerstein, R. Wang, G. Liu, H. Zhao, Igniting language intelligence: The hitchhiker's guide from chain-of-thought reasoning to language agents. <https://arxiv.org/abs/2311.11797>, 2023.