Demonstration of MultiVis-Agent: Interactively Generating Reliable Visualizations via Logic-Rule Agents

Jinwei Lu*
The Hong Kong Polytechnic University
Hong Kong, China
jwlu18@gmail.com

Jiawei Lu*
The Hong Kong Polytechnic University
Hong Kong, China
1749010569ljw@gmail.com

Chen Zhang
The Hong Kong Polytechnic University
Hong Kong, China
jason-c.zhang@polyu.edu.hk

Yuanfeng Song
ByteDance
Shanghai, China
songyf@outlook.com

Raymond Chi-Wing Wong
The Hong Kong University of Science and Technology
Hong Kong, China
raywong@cse.ust.hk

Abstract

Recent advances in LLM-based agent systems enable natural language–driven, cross-modal data visualization. Yet, existing systems remain unreliable for complex, multi-modal, or iterative tasks. This demonstration introduces **MultiVis-Agent** [3], an LLM agent-based system enhanced with logic rules for reliable visualization generation. It integrates specialized agents for data querying, visualization, and validation under a central coordinator governed by a four-layer logic rule framework ensuring parameter safety, error recovery, and guaranteed termination. Built upon the **MultiVis** task—covering four scenarios from basic generation to iterative refinement—MultiVis-Agent produces executable visualization code from natural language, databases, and optional references. We will show that MultiVis-Agent achieves real-time generation, execution, and refinement with 99% task completion and strong cross-modal reasoning. In particular, we emphasize an end-to-end and interactive demonstration where users can upload databases, provide multi-modal references (images/code), and iteratively refine executable visualization code within the same interface. The demo video and code are available at https://github.com/Jinwei-Lu/MultiVis.

CCS Concepts

• **Computing methodologies** → **Artificial intelligence**.

Keywords

Multi-Agent System, Data Visualization, Database Applications

ACM Reference Format:
Jinwei Lu, Jiawei Lu, Chen Zhang, Yuanfeng Song, and Raymond Chi-Wing Wong. 2026. Demonstration of MultiVis-Agent: Interactively Generating Reliable Visualizations via Logic-Rule Agents. In *Companion of the International Conference on Management of Data (SIGMOD Companion '26), May 31-June 05, 2026, Bengaluru, India.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3788853.3801584

*Both authors contributed equally to this research.

SIGMOD Companion '26, Bengaluru, India

ACM ISBN 979-8-4007-2450-3/2026/05
https://doi.org/10.1145/3788853.3801584

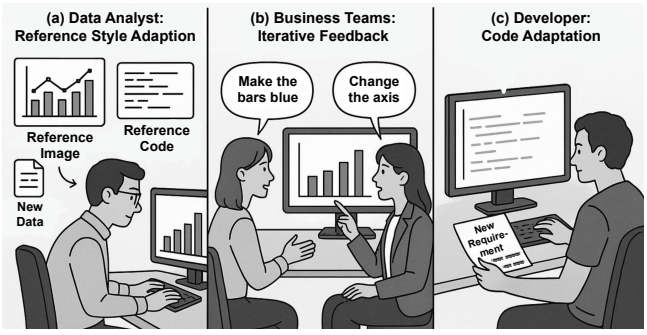


Figure 1: Complex Requirement of Real-world visualization tasks.

1 Introduction

Automated visualization generation plays a vital role in data analysis and decision-making processes. However, existing systems often fall short when confronted with the complexity and dynamic nature of real-world workflows [2, 6]. Current Text-to-Visualization (Text-to-Vis) approaches typically accept only single-modal natural language input [5] and produce one-shot outputs [2], which restricts their adaptability in scenarios requiring reference style transfer, iterative feedback incorporation, or fine-grained code adaptation (Figure 1).

Recent systems built upon large language models (LLMs) have improved flexibility, yet they continue to face significant reliability challenges, including catastrophic failures, infinite reasoning loops, and logically inconsistent reasoning [1, 4, 6]. These limitations underscore the urgent need for a more robust, interpretable and controllable system for visualization generation.

MultiVis-Agent. To address these challenges, we propose MultiVis-Agent, an automatic visualization generation system designed for reliable multi-modal data visualization. At its core, MultiVis-Agent is a logic rule–enhanced multi-agent framework that combines the reasoning capability of LLMs with the rigor of formal logic rules.

MultiVis-Agent is built upon a newly defined task, MultiVis, described in Section 2.1, which generalizes traditional Text-to-Vis paradigms into four practical scenarios: (1) Basic Generation, (2)

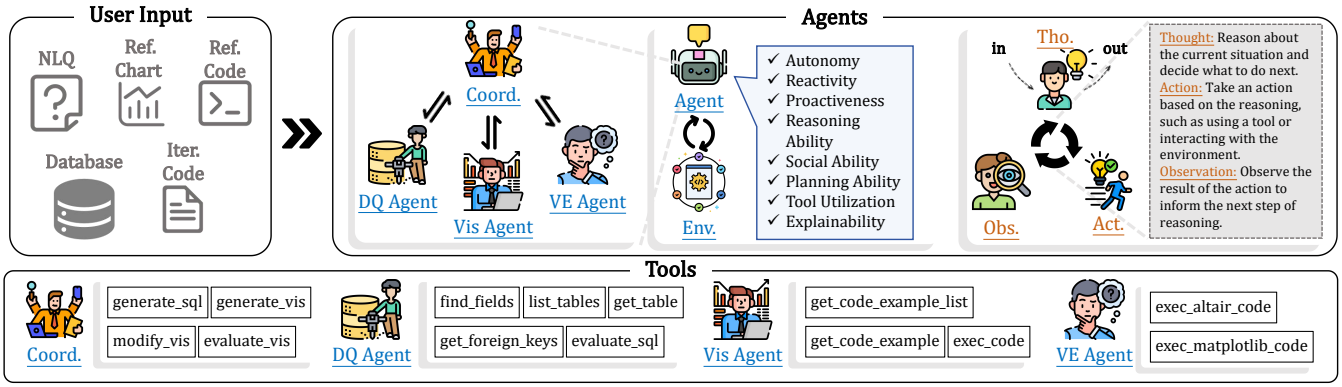


Figure 2: Overview of the MultiVis-Agent architecture. The system comprises a central rule-guided coordinating agent that manages the task lifecycle by governing three domain-specific agents (i.e., DA, Vis, and VE agents) for data visualization.

Image-Refere-nced Generation, (3) Code-Referenced Generation, and (4) Iterative Refinement. In this task, given a database, a natural language query (NLQ), and optional reference images or reference code, the system aims to automatically produce executable visualization code that accurately captures both the intended analytical content and visual style.

As a comprehensive solution framework for this task, MultiVis-Agent systematically decomposes the complex visualization generation process into data querying, visualization implementation, validation and refinement, each handled by specialized agents. The underlying logic rule–enhanced LLM architecture ensures parameter safety, systematic error recovery, and guaranteed termination. By combining the creativity of LLM reasoning with mathematically grounded decision-making, MultiVis-Agent achieves both adaptability and reliability in automated visualization generation.

In this demo paper, we focus on the user-facing system and interaction workflow, and present user-study evidence that the proposed design improves both usability and reliability across multi-modal scenarios. Compared with our full research paper [3], which describes the underlying algorithm and mathematical framework of the agent, this demonstration emphasizes three new perspectives: i) **End-to-end Workflow Experience:** The demo showcases the end-to-end user journey, illustrating how users seamlessly navigate from raw cross-modal data to final visualizations. ii) **Real-Time Interaction:** The demo extends the research framework with a front-end designed for practical usability. It introduces an interactive result interface where users can engage with the generated visualizations (e.g., zooming, highlighting). iii) **Empirical Human-Centric Validation:** The demo conducts user study, proving the system's practical utility, reduced cognitive load, and enhanced user trust compared to baseline automated method.

2 System Architecture

The architecture of MultiVis-Agent is designed to achieve reliable and cross-modal visualization generation under complex user requirements. As depicted in Figure 2, the framework processes multi-modal user inputs (NLQ, Reference File, Database) through a central Coordinator Agent that dynamically orchestrates three specialized agents: the Database & Query Agent, the Visualization Implementation Agent, and the Validation & Evaluation Agent. The system is governed by a four-layer logic rule framework that ensures reliable execution and error handling, enabling task-adaptive visualization generation and iterative refinement.

2.1 MultiVis Task

The MultiVis task extends the traditional Text-to-Vis paradigm into a multi-modal and iterative setting. Given base inputs (a database and an NLQ), and optionally a reference image or reference code, MultiVis formalizes four scenarios of visualization generation: i) **Basic Generation:** Automatically generates executable visualization code directly from natural language instructions and uploaded data. ii) **Image-Referenced Generation:** Inherits the visual style and layout patterns of an additionally provided reference image. iii) **Code-Referenced Generation:** Allows stylistic adaptation or code transfer from an unrelated reference visualization code. iv) **Iterative Refinement:** Refines or modifies an existing visualization to meet new analytical or stylistic requirements. These scenarios reflect real-world visualization needs where users often alternate between reference-guided and feedback-driven design processes.

In all scenarios, the objective is to automatically generate executable Python visualization code (Altair) that is semantically aligned with the user intent and consistent with the provided references, supporting both initial synthesis and iterative refinement.

2.2 Logic Rule–Enhanced Agent Architecture

A core innovation of MultiVis-Agent lies in its integration of LLM reasoning with a formally defined logic rule system to ensure reliability. While LLM-based visualization agents have demonstrated strong generalization in code generation, they often suffer from hallucination or infinite reasoning loops due to the absence of deterministic constraints and procedural safeguards. To address these challenges, MultiVis-Agent introduces a four-layer logic rule framework that constrains the agent's reasoning process.

The logic rule framework comprises Coordination Rules (CR), Tool Execution (TE) Rules, Error Handling (EH) Rules, and ReAct Control (RC) Rules:

- **Coordination Rules** operate at the system level to govern task classification, sub-agent coordination, and tool invocation, ensuring deterministic routing and consistent decision flows.

- **Tool Execution Rules** validate tool parameters and standardize the execution environment, enforcing boundary constraints and reproducible runs to reduce runtime failures.
- **Error Handling Rules** classify execution and validation errors and trigger targeted recovery strategies (e.g., output re-parsing and code correction) for reliable self-correction.
- **ReAct Control Rules** enforce bounded iterations and termination conditions to prevent infinite loops and redundant refinements in interactive workflows.

These rules jointly improve parameter safety, systematic recovery, and bounded convergence, which are particularly important for end-to-end and iterative visualization generation in the live demo.

2.3 Multi-Agent Coordination Framework

The MultiVis-Agent framework follows a centralized coordination paradigm that decomposes the visualization generation process into collaborative sub-tasks handled by specialized agents. The architecture consists of four main components: the Coordinator Agent, the Database and Query Agent, the Visualization Implementation Agent, and the Validation and Evaluation Agent.

The Coordinator Agent governs the overall workflow by identifying task types, dispatching responsibilities, and managing inter-agent communication. The Database and Query Agent interprets NLQ, constructs executable queries, and retrieves relevant data. The Visualization Implementation Agent transforms the data into executable visualization code, supporting both direct generation and reference-based adaptation. The Validation and Evaluation Agent assesses the correctness and quality of the generated outputs, providing structured feedback for refinement. This coordination framework enables robust and interpretable multi-modal visualization generation, supporting real-time interaction, iterative refinement, and consistent task completion.

3 Demonstration

In this section, we describe the website interface of MultiVis-Agent where users can easily generate visualization code and interactive chart.The online website will be continuously updated.

3.1 End-to-End Experience

Figure 3a illustrates the front-end interface of MultiVis-Agent, which provides an intuitive, web-based environment for automatic and cross-modal visualization generation. Users can interact with the system through three major steps:

(1) **Upload Database:** By clicking the "Upload Database" button, as shown in Figure 3a, the system automatically analyzes the uploaded dataset and displays its schema and a portion of its contents for reference, enabling users to inspect data tables and column structures when formulating their NLQ. This interactive preview helps users better align their NLQ with the database semantics and data attributes.
(2) **Select AI Model:** The interface includes an AI model selection panel, supporting mainstream APIs from OpenAI, Anthropic, Google, and DeepSeek, thereby reflecting the system's model-agnostic and extensible design. This design ensures that MultiVis-Agent can flexibly adapt to different foundation models according to user access and preference.
(3) **(Optional) Upload Reference Materials:** When users wish to guide the visualization generation with stylistic cues, they may upload reference materials. The "Upload Image" button allows users to submit a reference image, enabling the system to infer color schemes, chart types, or layout styles. Alternatively, by clicking the "Add Code" button, users can either manually input or upload a .py file containing reference visualization code. These references provide contextual guidance that enhances the generated visualization's stylistic or semantic fidelity.

After completing the above steps and a short processing period, the system returns a comprehensive result page containing: i) **Reference Preview:** displays the reference image or the visualization generated from the reference code when provided. ii) **Visualization Code:** the executable visualization code. iii) **Visualization Chart:** the corresponding visualization chart, which is rendered using the Altair library and supports interactive exploration such as data point highlighting and zooming.

Figure 3b shows an example. The NLQ in this example is "I like this chart showing daily values with a hazardous threshold line..."; the database is about journal sales. The task involved adapting a reference bar chart with a threshold line to visualize journal sales by theme, highlighting themes exceeding 2,000 sales in red while keeping blue for others. Our system accurately executed it, demonstrating superior comprehension of multi-modal instructions.

This integrated interface design allows users to directly compare generated results with references, supporting interactive inspection and iterative refinement within a single workspace.

3.2 Live Demonstration Scenarios

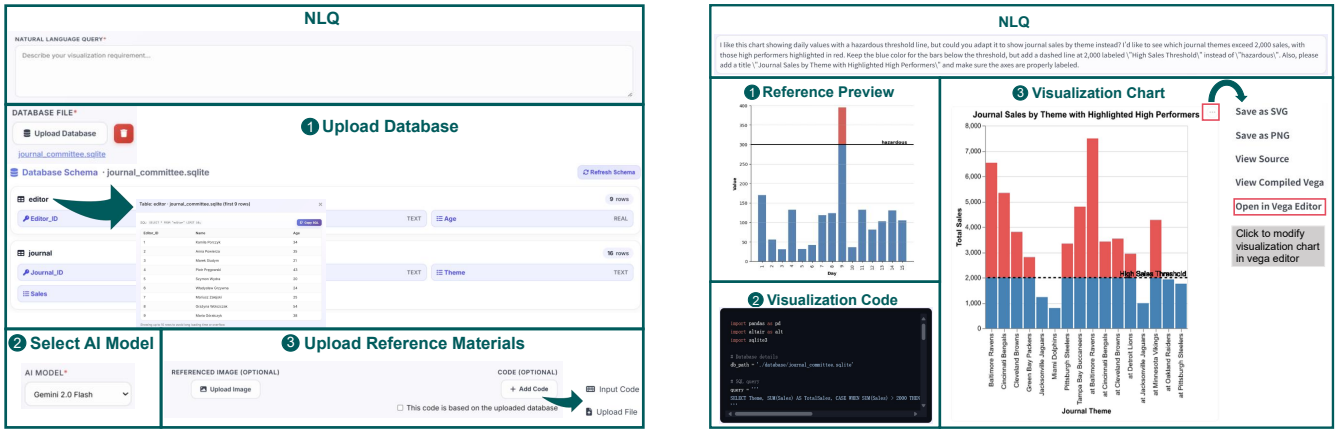
To better illustrate the practical utility of MultiVis-Agent beyond a static interface walk-through, we organize the live demo into four representative scenarios aligned with MultiVis. The on-site demonstration highlights the end-to-end workflow, where the system jointly performs data querying, code generation, execution, and validation in a closed loop. Figure 3 provides a compact storyboard view of the user input methods and result pages for the four scenarios presented during the demo session.

Scenario A (Basic Generation). Given a database and an NLQ, the system first retrieves relevant data through the Database & Query Agent and then generates executable Altair code through the Visualization Implementation Agent. The generated chart is rendered on the result page for interactive inspection.

Scenario B (Image-Referenced Generation). When a reference chart image is provided, MultiVis-Agent additionally extracts visual cues (e.g., chart type, and layout) and transfers them to the new visualization. The demo emphasizes style-consistent generation under cross-modal constraints.

Scenario C (Code-Referenced Generation). When a reference code snippet is provided (possibly unrelated to the uploaded database), the system adapts reusable design patterns and implementation idioms to the current task. The demo illustrates how code references can improve controllability and reduce trial-and-error in complex chart construction.

Scenario D (Iterative Refinement). Starting from an existing visualization code, users provide follow-up natural language instructions (e.g., "Change the bar color to blue") to modify analytical



(a) User Interface of MultiVis-Agent

(b) Result Page of MultiVis-Agent

Figure 3: Front-end of MultiVis-Agent. (a) shows the user interface, (b) is the result page.

Table 1: User satisfaction scores (0-10) and task success rates (%) across methods. Format: Score / Success.

Scenario	Ours	w/o. Rule	Inst. LLM	Workflow
Basic Gen.	**9.1 / 100**	7.2 / **100**	7.9 / **100**	7.4 / **100**
Image-Ref. Gen.	**9.3 / 96.7**	6.9 / 90	7.6 / 87	8.1 / 93
Code-Ref. Gen.	**9.5 / 100**	7.7 / **100**	8.4 / 97	8.2 / 90
Iter. Refine.	9.0 / **100**	8.5 / 93.3	8.8 / 93	**9.1 / 100**
Average	**9.225 / 99.2**	7.575 / 95.8	8.175 / 94.3	8.2 / 95.8

content or visual presentation. Users also have the flexibility to manually tweak the generated code. The system iteratively executes, validates, and refines the code.

3.3 User Study

To evaluate the usability and reliability of MultiVis-Agent as an interactive system (which is also the focus of this demonstration), we conducted a controlled user study with thirty volunteers experienced in programming and data visualization. Each participant completed four visualization tasks aligned with the four MultiVis scenarios: Basic Generation, Image-Referenced Generation, Code-Referenced Generation, and Iterative Refinement.

Since our task formulation is newly proposed, there are no existing methods available for comparison.Therefore, to evaluate the effectiveness of our proposed MultiVis-Agent system, we designed two baseline models that represent common paradigms for LLM-based visualization generation: i) **Instructing LLM:** A direct prompting approach that combines all the user's inputs into a single comprehensive prompt for one-step visualization. ii) **LLM Workflow:** A sequential pipeline approach that breaks visualization generation into distinct stages (SQL generation followed by visualization code generation), without the dynamic coordination mechanism present in our agent-based approach.

For each scenario, volunteers used four methods—MultiVis-Agent, MultiVis-Agent without logic rules, instructing LLM, and LLM workflow. All experiments used the Gemini model as the underlying LLM to ensure consistent comparisons.

We recorded whether each task was successfully completed, and participants rated the generated outputs on a 10-point Likert scale according to three criteria: accuracy, visual expressiveness, and semantic alignment. To ensure unbiased evaluation, participants were unaware of the method used in each trial.

The results indicated that MultiVis-Agent consistently achieved the highest average task success rate and user satisfaction across the baselines, which aligns with the qualitative experience showcased in the live demo (i.e., end-to-end generation, execution, and refinement under multi-modal inputs). Detailed task success rates and user ratings are presented in Table 1, demonstrateing the system's effectiveness and user-friendliness in handling complex cross-modal visualization tasks.

Acknowledgments

Yuanfeng Song is the corresponding author. The research of Chen Zhang is supported by grants from: (1) the NSFC/RGC Joint Research Scheme sponsored by the Research Grants Council of Hong Kong and the National Natural Science Foundation of China (Project No. N_PolyU5179/25); (2) the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. PolyU25600624); (3) the Innovation Technology Fund (Project No. ITS/052/23MX and PRP/009/22FX). The research of Raymond Chi-Wing Wong is supported by PRP/004/25FX.

References

[1] Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, et al. 2024. MetaGPT: Meta programming for a multi-agent collaborative framework. In *ICLR*.

[2] Shuaimin Li, Xuanang Chen, Yuanfeng Song, Yunze Song, Chen Jason Zhang, Fei Hao, and Lei Chen. 2025. prompt4vis: prompting large language models with example mining for tabular data visualization. *VLDBJ* (2025).

[3] Jinwei Lu, Yuanfeng Song, Chen Zhang, and Raymond Chi-Wing Wong. 2026. MultiVis-Agent: A Multi-Agent Framework with Logic Rules for Reliable and Comprehensive Cross-Modal Data Visualization. *Proc. ACM Manag. Data* 4, 1, Article 56 (feb 2026). doi:10.1145/3786670

[4] Yuyu Luo, Guoliang Li, Ju Fan, and Nan Tang. 2026. Data Agents: Levels, State of the Art, and Open Problems. *SIGMOD* (2026).

[5] Paula Maddigan and Teo Susnjak. 2023. Chat2VIS: Generating Data Visualizations via Natural Language Using ChatGPT, Codex and GPT-3 Large Language Models. *IEEE Access* (2023).

[6] Geliang Ouyang, Jingyao Chen, Zhihe Nie, Yi Gui, Yao Wan, Hongyu Zhang, and Dongping Chen. 2025. nvAgent: Automated Data Visualization from Natural Language via Collaborative Agent Workflow. In *ACL*.