# LLM4VA: A Multi-Agent LLM Framework for Automated Prototyping of Visual Analytics Systems

Junjie Xiong* ShanghaiTech University
Longfei Chen* ShanghaiTech University
Xiaofeng Dou ShanghaiTech University
Shiwei Wang Huolala Technology Co., Ltd
Ken Lin Huolala Technology Co., Ltd
Sibei Yang Sun Yat-sen University
Quan Li* ShanghaiTech University

**ABSTRACT**

Visual analytics (VA) systems empower data exploration through interactive visualization techniques, yet their development has traditionally relied on significant human expertise in data analysis, requirement engineering, programming, and visualization design. While existing visualization recommendation methods have advanced the automated generation of basic visualizations, they fall short in addressing the complex, iterative demands of end-to-end VA system development. Recent advancements in large language model (LLM)-powered multi-agent architectures have demonstrated considerable potential in automating intricate tasks across domains such as software engineering. Inspired by this, we propose *LLM4VA*, a novel LLM-driven multi-agent framework designed to automate VA system design and prototyping. The framework decomposes the development pipeline into core tasks—data interpretation, requirement exploration, knowledge retrieval, visualization design, and prototype implementation—and assigns them to specialized, collaborative agents. This approach facilitates dynamic human-AI collaboration, enabling developers to iteratively refine agent-generated outputs and steer system behavior. To operationalize this framework, we implement a hybrid user interface that supports real-time progress monitoring, human-in-the-loop validation, and interactive adjustments. A usage scenario and user study ($N = 11$) demonstrate the framework's effectiveness in streamlining VA workflows and its extensibility for full system development.

**Index Terms:** Visual Analytics System, Large Language Model, Multi-Agent

## 1 INTRODUCTION

Visual analytics (VA) integrates automated analysis techniques with interactive visualizations to help users comprehend data, reason systematically, and make informed decisions [25]. Empirical studies have validated the effectiveness of VA approaches across diverse domains, including urban planning [5], game analytics [29], and social network analysis [8]. A typical VA system integrates multiple interrelated views to facilitate data exploration and analysis. However, designing effective views requires balancing data characteristics with user needs to ensure clear insights—a complex process demanding visualization expertise and tailored encodings.

To streamline visualization design, researchers have proposed various automated methods. Rule-based approaches [40, 33] recommend visualizations by applying predefined mapping rules based on data types. Machine learning (ML)-based approaches [32, 36, 24, 13] alleviate human efforts by learning design patterns from large-scale historical data. Natural language interfaces (NLIs) further enhance interaction by translating user queries into visualizations [43]. More recently, Large Language Models (LLMs) have been explored for visualization generation. For example, *Chat2VIS* [34] evaluates the visualization generation capabilities of different LLMs, *LLM4Vis* [47] introduces a prompting strategy for visualization generation and explanation, and *ChartGPT* [46] finetunes an open-source LLM using a dataset of abstract utterances and corresponding charts.

Despite advances, existing automated methods have two critical limitations. First, current approaches prioritize data characteristics over specific analytical tasks that VA systems support—such as pattern recognition, causal inference, and decision-making [2, 39, 41]. For example, time-series data might require trend forecasting [54] in one scenario and anomaly detection [31] in another. However, existing approaches lack mechanisms to align visualizations with such needs. Second, automated visualization generation is typically limited to simple chart types (e.g., bar and line charts), inadequate for complex VA tasks. Real-world VA applications [4, 12] often require integrating multiple visualization techniques to manage large-scale, diverse, and dynamic data. For example, *FSLens*[6] overlays maps, heatmaps, and radar-like glyphs within a single view to represent fire incident frequency and fire station distribution. The composition relies heavily on developer expertise and established design principles[44, 26, 37, 2, 38, 42]. Therefore, while existing approaches made strides in generating simple visualizations, they remain inadequate for comprehensive VA system development.

To address these challenges, we propose *LLM4VA*, a multi-agent framework leveraging LLMs to automate the design and prototyping of VA system. The choice of an LLM-based approach is motivated by LLMs' strong semantic understanding and contextual reasoning capabilities, which enable them to flexibly accommodate diverse data types and analytical requirements [55]. Additionally, LLMs can generate visualization code [28] (e.g., *Vega-Lite*, *D3.js*), reducing implementation effort for developers and improving efficiency. Since VA development typically involves multiple tasks [11], such as data insights, requirement exploration, and visualization design, our framework adopts a multi-agent architecture to handle these processes. By assigning tasks to independent yet collaborative agents, each agent focuses on its specific domain, improving overall efficiency and decision quality [20]. Meanwhile, users actively participate in decision-making through interactive engagement, ensuring the final solution aligns with actual needs.

*LLM4VA* consists of five stages, each handled by a dedicated agent: **(1) Data Interpretation:** A *data analyst* agent processes raw data, extracts key features, and generates structured reports; **(2) Requirement Exploration:** A *domain expert* agent reviews the data report, incorporates user intent, and identifies analytical requirements; **(3) Visualization Knowledge Retrieval:** A *searcher* agent retrieves relevant case studies and design principles from a VA knowledge base; **(4) System Design:** A *designer* agent synthesizes data characteristics, analytical requirements, and domain knowledge to formulate a VA system design; **(5) Prototype Implementation:** A *coder* agent translates the design into executable visualization code. These agents work collaboratively to streamline the VA development, while human users act as supervisors, provid-

*liquan@shanghaitech.edu.cn

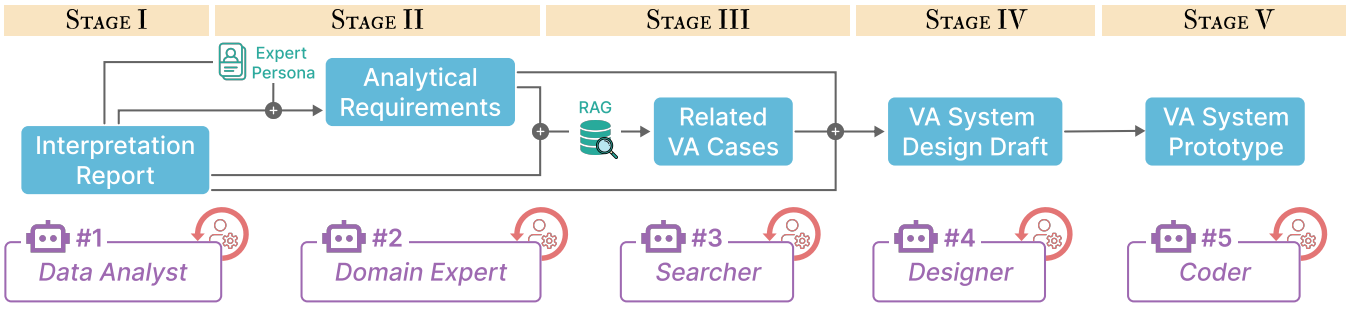


Figure 1: The *LLM4VA* framework with five stages: (I) Data Interpretation, (II) Requirement Exploration, (III) Visualization Knowledge Retrieval, (IV) System Design, (V) Prototype Implementation.

ing feedback and making adjustments as needed. To enhance interaction, we developed a visualization system as shown in Fig. 3: **(A) Work Process View:** Displays the workflow, enabling real-time progress tracking; **(B) Dialogue History View:** Supports natural language communication between users and agents; **(C) Agent Output View:** Presents intermediate outputs facilitating user inspection and refinement.

## 2 LLM4VA Framework

As illustrated in Fig. 1, the *LLM4VA* framework is structured into five key stages. The following subsections provide a detailed breakdown of each stage's workflow and technical implementation. Notably, the framework offers high flexibility in selecting the LLM for its agents, allowing users to integrate any preferred model.

### 2.1 Stage I: Data Interpretation

The goal of this stage is to build a comprehensive understanding of the input data and produce a structured interpretation report as the foundation for later processing. A dedicated *data analyst* agent handles datasets in CSV, XLSX, or JSON format, using the *pandas* toolkit to extract core characteristics such as dataset size, field descriptions, and missing value distribution. The resulting report includes: 1) **Size:** number of records and fields; 2) **Fields:** names and descriptions; 3) **Missing Values:** counts per field. If multiple data files are uploaded, the agent generates individual reports and integrates them into a unified summary, while also identifying potential relationships based on shared or similar fields.

**Human-in-the-loop Interaction.** Users are required to upload data files and may provide optional explanations. After the report is generated, users can review and edit its contents for accuracy.

### 2.2 Stage II: Requirement Exploration

This stage focuses on extracting specific analytical requirements from the data. Drawing inspiration from formative methods in visualization research, a *domain expert* agent analyzes the interpretation report to infer analytical goals based on context. A key aspect is selecting a suitable expert identity aligned with the dataset's domain—for example, economic data requires a different perspective than literary texts. To support this, we adopt the *ExpertPrompting* [53] approach, leveraging in-context learning [3] to dynamically generate domain-relevant expert personas via a large language model. Operating within this contextual frame, the agent formulates structured analytical requirements comprising: 1) **Task:** the analytical objective. 2) **Insights:** the targeted knowledge. 3) **Related Data:** associated files and fields relevant to the task.

**Human-in-the-loop Interaction.** Users may customize the expert persona to better fit the data domain and are encouraged to review, refine, and expand the generated analytical requirements, ensuring alignment with practical goals.

### 2.3 Stage III: Visualization Knowledge Retrieval

After determining the data and analytical requirements, the next step is to design the VA system. Traditionally, this process depends on the designer's expertise in visualization techniques and prior research. However, general-purpose LLMs often lack such domain-specific knowledge, which limits their ability to produce high-quality designs. To address this, we adopt retrieval-augmented generation (RAG) [27], which improves LLM performance by incorporating relevant information retrieved from an external knowledge base.

We construct a dedicated visualization knowledge base using academic publications to provide essential context for system design. We collect papers describing VA system designs from leading conferences and journals in the visualization domain, extracting key information about the data and design requirements involved. A *searcher* agent retrieves relevant papers based on the data summaries from Stage I and the analytical requirements from Stage II. The full construction process is described in Appendix B.

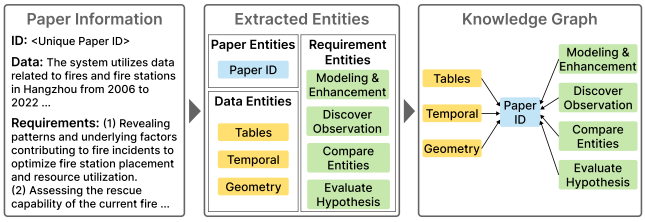


Figure 2: Showcasing the paper-to-KG transformation.

To improve retrieval precision, we adopt a hybrid mechanism inspired by *GraphRAG*[15] and *LightRAG*[21], combining semantic embedding and structured conceptual knowledge. Given a query $Q = \langle data, requirement \rangle$, the *searcher* agent executes a two-step process to identify relevant papers.

First, the *searcher* employs a pretrained text embedding model to project both $Q$ and VA knowledge base content into the same semantic space. Relevant papers are identified by computing cosine similarity between query and paper embeddings.

Next, the *searcher* abstracts data and requirement descriptions in $Q$ into high-level conceptual types and performs structured retrieval based on a knowledge graph (KG). The KG consists of three entity types: **Paper Entities** ($E_p$) represent individual papers in the knowledge base; **Data Entities** ($E_d$) represent abstract data types used in papers, such as "*temporal*" or "*geometry*" data; and **Requirement Entities** ($E_r$) represent abstract analytical task types.

As shown in Fig. 2, we use LLMs to extract data and requirement types based on Wu et al.'s framework [50], defining 9 data types and 17 requirement types. These entities are linked to corresponding papers in the KG. The *searcher* merges results from both retrieval channels and produces structured outputs for each selected paper, including title, data type, requirement type, VA design, and selection rationale—providing contextual knowledge for system design.

**Human-in-the-loop Interaction.** Users can review the retrieved papers and apply additional filters. They can also adjust the number of recommended papers or manually add additional references.

### 2.4 Stage IV: System Design

A *designer* agent creates design drafts by integrating data, analytical requirements, and insights from previous VA research. The pro-

cess starts with view segmentation, where the designer determines the number of primary views for the VA system. For each view, the *designer* defines the data to display and analytical requirements to address. Next, it creates a detailed visualization plan for each view, including selecting visual encodings, configuring visual variables (color, shape, size), and mapping these variables to data attributes.

**Human-in-the-loop Interaction.** Users can review proposed design drafts and provide feedback to optimize the system. They can suggest changes to view segmentation, visual encodings, and other aspects to better align with practical needs.

### 2.5 Stage V: Prototype Implementation

In the final stage, a *coder* agent transforms the design drafts into an executable VA prototype using Vue and D3.js. Each view is implemented as a separate Vue component, with D3.js used for flexible visualization rendering. This component-based architecture ensures modularity, maintainability, and scalability, supporting future enhancements.

**Human-in-the-loop Interaction.** Users can review the generated code and provide feedback, suggesting changes to functionality, layout, or appearance. They can also modify the code manually to better suit specific requirements or to refine certain details.

## 3 LLM4VA System

### 3.1 System Overview

To enhance user interaction with the *LLM4VA* framework, we designed a system interface comprising three integrated components, as shown in Fig. 3. These components – **(A) Work Process View**, **(B) Dialogue History View**, and **(C) Agent Output View** – collectively support workflow tracking, interactive communication, and iterative content refinement.

**Work Process View (Fig. 3 (A)):** This component visualizes the workflow as a directed graph, where nodes represent LLM agents corresponding to distinct development stages: *data analyst*, *domain expert*, *searcher*, *designer*, and *coder*. Users can configure each agent's LLM (e.g., *GPT* series) by clicking the button. Agent states update dynamically throughout the workflow: initially inactive, transitioning to active during execution, and finally completed. Users can navigate between stages by clicking nodes, facilitating adjustments at any development phase.

**Dialogue History View (Fig. 3 (B)):** This component employs a chat-based interface for natural language interactions with LLM agents. Users can send messages to the current agent using the button or advance the workflow to the next agent via the button. The view maintains complete conversation records between users and agents throughout the workflow, including manual modifications, ensuring contextual continuity.

**Agent Output View (Fig. 3 (C)):** This component displays structured outputs from LLM agents, including data interpretation reports, analytical requirements, related papers, design drafts, and code implementations. Outputs are formatted for readability with tailored visualizations to enhance interpretability. Users can directly review and edit results, enabling iterative refinements.

### 3.2 System Walkthrough

To demonstrate a typical use case, consider Emma, an analyst at an educational institution aiming to visualize student learning progress for tracking academic performance, identifying learning difficulties, and optimizing instructional strategies. She uses the *LLM4VA* framework for this prototyping task. Emma's analysis leverages three tabular data files: StudentInfo.csv containing student metadata, TitleInfo.csv detailing assessment items, and SubmitRecord.csv capturing students' answer records.

**Interpreting the Data.** Upon uploading files, the *data analyst* agent automatically generates a structured interpretation report. The interface visualizes data files and relationships through a node-link graph (Fig. 3 (1-1)), where orange nodes represent individual files. Clicking a node reveals contents in tabular view (Fig. 3 (1-2)), with links signifying inferred relationships—for instance, the shared `student_ID` field between StudentInfo.csv and SubmitRecord.csv. The report (Fig. 3 (1-3)) summarizes key metrics including data volume, field descriptions, and completeness indicators. Emma identifies a misinterpretation of the `method` field, erroneously labeled as "*submission type*" rather than "*programming language used for submission*". She corrects it via the button.

**Exploring the Requirements.** After refining the data report, Emma articulates her analysis goal: "*I would like to assess students' learning progress.*" She submits this objective with the data interpretation to the *domain expert* agent via the button. The agent proposes two analytical requirements: **R1** focuses on evaluating student score distribution and performance trends across demographic segments, while **R2** tracks iterative submission behavior over time to understand improvement trends. The system visually maps relationships between analytical requirements and dataset elements using a node-link graph (Fig. 3 (2-1)), with a dedicated panel displaying granular requirement details (Fig. 3 (2-2)). Emma reviews these outputs, noting: *These insights are incredibly precise and align perfectly with my intent. I hadn't even considered some of these perspectives!*" Inspired by **R1**, she expands the analysis scope via the chat interface: "*I would also like some insights into question design efficacy.*" The *domain expert* generates an additional requirement **R3**, which evaluates specific question performance to determine curriculum alignment with expected learning outcomes.(Fig. 3 (2-2)) Satisfied with the augmented analytical requirements, Emma proceeds to the next workflow phase.

**Retrieving Relevant Cases.** Guided by the dataset and analytical requirements, the *searcher* agent retrieves 13 candidate papers from the knowledge base, including [51, 30, 10, 9, 49, 16, 7, 22, 17, 48, 14, 19, 52]. These recommendations are structured as a KG (Fig. 3 (3-1)), where blue rectangles denote paper entities, orange circles represent data-type entities, and green squares indicate requirement-type entities. Edges encode relationships such as *the paper uses (type) of data*" or *the paper addresses (type) of requirement*". Edge bundling mitigates visual clutter by aggregating bijective edge sets between entity groups [45]. Emma evaluates recommendations via the expandable list (Fig. 3 (3-2)), inspecting each paper's data sources, requirements, visualization methods, and algorithmic justification. While most papers align with her objectives, she identifies two mismatches. *Socrates* [48] focuses on machine-guided adaptive data stories, which the *searcher* justifies as helping educators extract insights from learning data. However, Emma finds it unconvincing, as *Socrates* does not align with her research objectives. Similarly, [17] emphasizes self-regulated learning for AI beginners, diverging from her performance trend focus. Emma removes both papers using the button, refining the selection to 11 relevant studies.

**Designing and Implementing the Prototype.** Based on previous agents' outputs, the *designer* agent plans a three-view VA prototype targeting each analytical requirement (Fig. 3 (4)). Finding textual design drafts lacking intuitiveness, Emma invokes the *coder* agent to generate an executable prototype within a code sandbox (Fig. 3 (5)). The sandbox displays auto-generated code in the upper pane and renders the VA interface below. Using the Show Description button, Emma compares textual drafts with live visualizations side-by-side. Though largely satisfied with the prototype's alignment to analytical goals, Emma identifies minor flaws—missing view headers and overlapping visual elements. She iteratively refines the prototype by directing the *coder* agent to add headers and manually adjusting aesthetics. The final prototype is detailed in Appendix C.

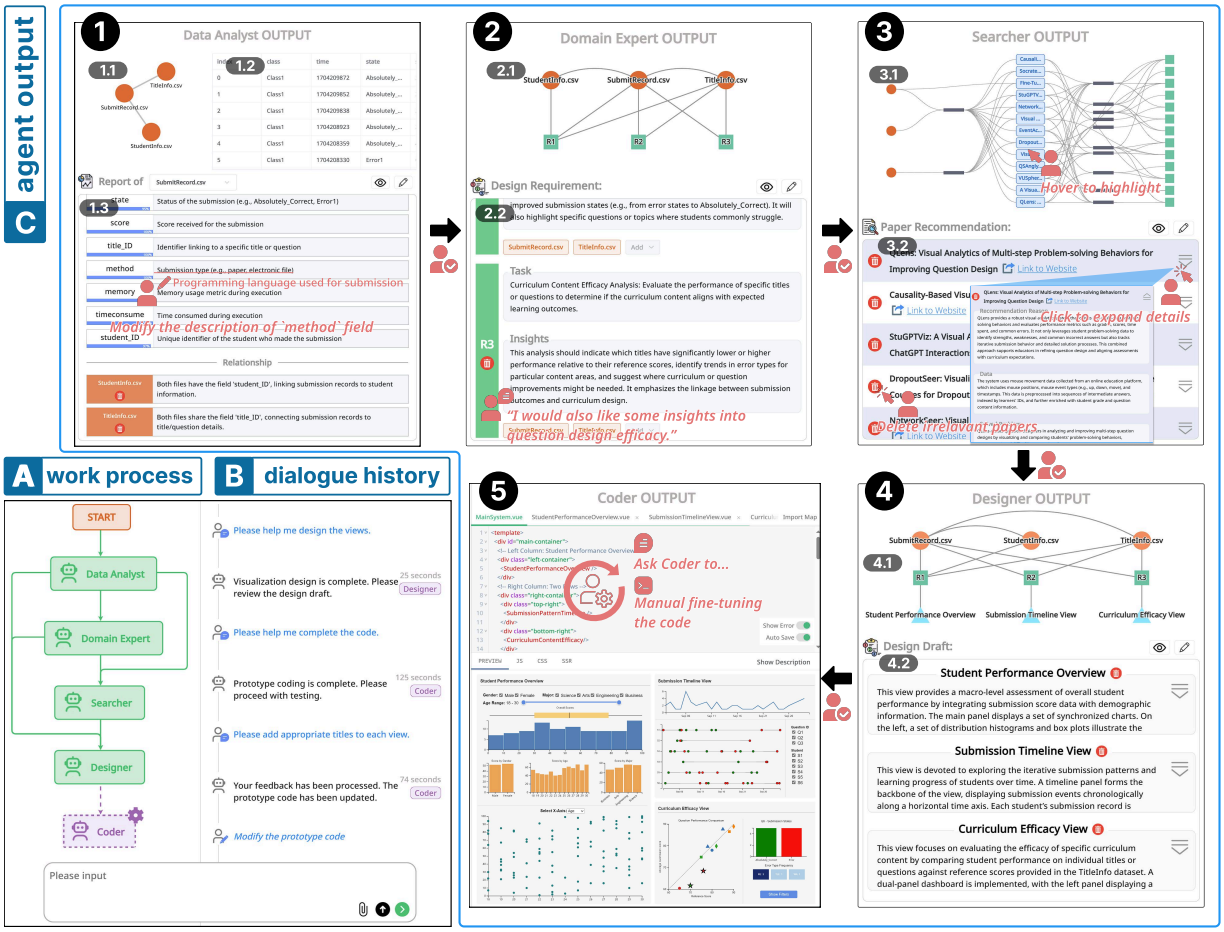


Figure 3: The user interface and system walkthrough of *LLM4VA*.

## 4 Discussion and Limitation

**"Quick and Dirty" Prototyping.** The primary goal of *LLM4VA* is to streamline VA prototype creation, minimizing development overhead while transforming conceptual ideas into functional systems. User studies confirm that *LLM4VA* rapidly generates foundational, operational prototypes, enabling users to test diverse visualization strategies and explore multi-perspective data insights. This aligns with the quick and dirty" prototyping paradigm [23], which prioritizes cost-effective, iterative validation using readily available tools. Prior research demonstrates that this approach facilitates brainstorming [18] and early evaluation [1] while providing invaluable feedback for design iterations. Built on these principles, *LLM4VA* leverages a multi-agent framework to simplify prototype creation through LLM-assisted automation, allowing users to swiftly visualize ideas, identify shortcomings, and refine designs.

**LLM-Driven Assistance Across the VA Pipeline.** The *LLM4VA* framework structures five distinct stages, each augmented by specialized LLM agents providing comprehensive development assistance. First, LLMs automatically generate data processing and visualization code, freeing users from tedious programming tasks. The *coder* agent translates design drafts into visualization code, reducing manual programming burdens. While generated code may contain minor flaws, users can easily debug outputs, accelerating development. Second, LLMs offer intelligent suggestions that inspire deeper exploration. The *designer* agent analyzes existing data patterns and successful cases to recommend appropriate visualization types, layouts, and interaction designs. Finally, LLMs facilitate iterative refinement by dynamically optimizing content based on user feedback. Users simply describe desired modifications, and the system swiftly adjusts designs to produce improved solutions, ensuring continuous alignment with evolving requirements.

**Limitations.** Despite the promising results demonstrated by our approach, several limitations remain that warrant further exploration. First, the current framework only considers metadata of data attributes as input. As a result, the generated VA prototypes tend to directly map raw data, lacking advanced data transformation and processing capabilities. One possible improvement is to enhance the data analysis stage within the framework, enabling it to better handle data transformations and support more sophisticated visualization needs. Second, while *LLM4VA* performs well in generating static visualizations or those with basic interactive functions, it still faces challenges in more complex interaction designs, such as dynamic filtering and multi-view coordination. Future improvements will focus on strengthening the framework's capabilities in interactive design to enhance the flexibility and scalability of VA prototypes. Finally, users may become overly reliant on LLM-generated recommendations, neglecting critical evaluation of the suggested solutions. This dependence could lead to suboptimal design choices and reinforce potential biases present in the LLM's training data. To mitigate this issue, future versions of *LLM4VA* could incorporate more transparent design explanations, making users maintain critical thinking in their decision-making process.

## 5 Conclusion

In this study, we propose *LLM4VA*, a multi-agent framework that leverages LLMs to facilitate the design and development of VA prototypes. By seamlessly integrating LLMs into the development process, our approach enables users to efficiently explore data and requirements, retrieve relevant knowledge, design visualizations, and rapidly implement VA prototypes. A usage scenario and a user study (Appendix A) demonstrate that *LLM4VA* enhances the efficiency of VA design and prototyping, helping users transform their ideas into functional prototypes and laying a solid foundation for the development of mature systems.

## REFERENCES


[1] L. Baillie and L. Morton. Designing quick & dirty applications for mobiles: Making the case for the utility of hci principles. Journal of computing and information technology, 18(2):103–107, 2010. 4

[2] M. Brehmer and T. Munzner. A multi-level typology of abstract visualization tasks. IEEE transactions on visualization and computer graphics, 19(12):2376–2385, 2013. 1

[3] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. Advances in neural information processing systems, 33:1877–1901, 2020. 2

[4] J. Chen, M. Ling, R. Li, P. Isenberg, T. Isenberg, M. Sedlmair, T. Möller, R. S. Laramee, H.-W. Shen, K. Wünsche, et al. Vis30k: A collection of figures and tables from ieee visualization conference publications. IEEE Transactions on Visualization and Computer Graphics, 27(9):3826–3833, 2021. 1

[5] L. Chen, Y. Ouyang, H. Zhang, S. Hong, and Q. Li. Riseer: Inspecting the status and dynamics of regional industrial structure via visual analytics. IEEE transactions on visualization and computer graphics, 29(1):1070–1080, 2022. 1

[6] L. Chen, H. Wang, Y. Ouyang, Y. Zhou, N. Wang, and Q. Li. Fslens: A visual analytics approach to evaluating and optimizing the spatial layout of fire stations. IEEE Transactions on Visualization and Computer Graphics, 2023. 1

[7] N.-C. Chen and B. Kim. Qsanglyzer: Visual analytics for prismatic analysis of question answering system evaluations. In 2017 IEEE Conference on Visual Analytics Science and Technology (VAST), pp. 48–58. IEEE, 2017. 3

[8] S. Chen, S. Chen, Z. Wang, J. Liang, X. Yuan, N. Cao, and Y. Wu. D-map: Visual analysis of ego-centric information diffusion patterns in social media. In 2016 IEEE conference on visual analytics science and technology (VAST), pp. 41–50. IEEE, 2016. 1

[9] Y. Chen, Q. Chen, M. Zhao, S. Boyer, K. Veeramachaneni, and H. Qu. Dropoutseer: Visualizing learning patterns in massive open online courses for dropout reasoning and prediction. In 2016 IEEE Conference on Visual Analytics Science and Technology (VAST), pp. 111–120. IEEE, 2016. 3

[10] Z. Chen, J. Wang, M. Xia, K. Shigyo, D. Liu, R. Zhang, and H. Qu. Stugptviz: A visual analytics approach to understand student-chatgpt interactions. IEEE Transactions on Visualization and Computer Graphics, 2024. 3

[11] W. Cui. Visual analytics: A comprehensive overview. IEEE access, 7:81555–81573, 2019. 1

[12] D. Deng, Y. Wu, X. Shu, J. Wu, S. Fu, W. Cui, and Y. Wu. Visimages: A fine-grained expert-annotated visualization dataset. IEEE Transactions on Visualization and Computer Graphics, 29(7):3298–3311, 2022. 1

[13] V. Dibia and Ç. Demiralp. Data2vis: Automatic generation of data visualizations using sequence-to-sequence recurrent neural networks. IEEE computer graphics and applications, 39(5):33–46, 2019. 1

[14] F. Du, C. Plaisant, N. Spring, and B. Shneiderman. Eventaction: Visual analytics for temporal event sequence recommendation. In 2016 IEEE Conference on Visual Analytics Science and Technology (VAST), pp. 61–70. IEEE, 2016. 3

[15] D. Edge, H. Trinh, N. Cheng, J. Bradley, A. Chao, A. Mody, S. Truitt, and J. Larson. From local to global: A graph rag approach to query-focused summarization. arXiv preprint arXiv:2404.16130, 2024. 2

[16] S. Fu, J. Zhao, W. Cui, and H. Qu. Visual analysis of mooc forums with iforum. IEEE Transactions on Visualization and Computer Graphics, 23(1):201–210, 2016. 3

[17] L. Gao, J. Lu, Z. Shao, Z. Lin, S. Yue, C. Ieong, Y. Sun, R. J. Zauner, Z. Wei, and S. Chen. Fine-tuned large language model for visualization system: A study on self-regulated learning in education. IEEE Transactions on Visualization and Computer Graphics, 2024. 3

[18] G. Gómez and R. Lopez-Leon. Impossible design: fostering creativity by quick and dirty prototyping. 2019. 4

[19] S. Guo, Z. Jin, D. Gotz, F. Du, H. Zha, and N. Cao. Visual progression analysis of event sequence data. IEEE transactions on visualization and computer graphics, 25(1):417–426, 2018. 3

[20] T. Guo, X. Chen, Y. Wang, R. Chang, S. Pei, N. V. Chawla, O. Wiest, and X. Zhang. Large language model based multi-agents: A survey of progress and challenges. arXiv preprint arXiv:2402.01680, 2024. 1

[21] Z. Guo, L. Xia, Y. Yu, T. Ao, and C. Huang. Lightrag: Simple and fast retrieval-augmented generation. arXiv preprint arXiv:2410.05779, 2024. 2

[22] H. He, O. Zheng, and B. Dong. Vusphere: Visual analysis of video utilization in online distance education. In 2018 IEEE conference on visual analytics science and technology (VAST), pp. 25–35. IEEE, 2018. 3

[23] N. Hollender, C. Hofmann, M. Deneke, and B. Schmitz. Integrating cognitive load theory and concepts of human–computer interaction. Computers in human behavior, 26(6):1278–1288, 2010. 4

[24] K. Hu, M. A. Bakker, S. Li, T. Kraska, and C. Hidalgo. Vizml: A machine learning approach to visualization recommendation. In Proceedings of the 2019 CHI conference on human factors in computing systems, pp. 1–12, 2019. 1

[25] D. Keim, G. Andrienko, J.-D. Fekete, C. Görg, J. Kohlhammer, and G. Melançon. Visual analytics: Definition, process, and challenges. Springer, 2008. 1

[26] D. A. Keim, F. Mansmann, J. Schneidewind, J. Thomas, and H. Ziegler. Visual analytics: Scope and challenges. Springer, 2008. 1

[27] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. Advances in neural information processing systems, 33:9459–9474, 2020. 2

[28] G. Li, X. Wang, G. Aodeng, S. Zheng, Y. Zhang, C. Ou, S. Wang, and C. H. Liu. Visualization generation with large language models: An evaluation. arXiv preprint arXiv:2401.11255, 2024. 1

[29] Q. Li, P. Xu, Y. Y. Chan, Y. Wang, Z. Wang, H. Qu, and X. Ma. A visual analytics approach for understanding reasons behind snowballing and comeback in moba games. IEEE transactions on visualization and computer graphics, 23(1):211–220, 2016. 1

[30] R. Li, W. Cui, T. Song, X. Xie, R. Ding, Y. Wang, H. Zhang, H. Zhou, and Y. Wu. Causality-based visual analysis of questionnaire responses. IEEE Transactions on Visualization and Computer Graphics, 30(1):638–648, 2023. 3

[31] D. Liu, S. Alnegheimish, A. Zytek, and K. Veeramachaneni. Mtv: Visual analytics for detecting, investigating, and annotating anomalies in multivariate time series. Proceedings of the ACM on Human-Computer Interaction, 6(CSCW1):1–30, 2022. 1

[32] Y. Luo, X. Qin, N. Tang, and G. Li. Deepeye: Towards automatic data visualization. In 2018 IEEE 34th international conference on data engineering (ICDE), pp. 101–112. IEEE, 2018. 1

[33] J. Mackinlay, P. Hanrahan, and C. Stolte. Show me: Automatic presentation for visual analysis. IEEE transactions on visualization and computer graphics, 13(6):1137–1144, 2007. 1

[34] P. Maddigan and T. Susnjak. Chat2vis: generating data visualizations via natural language using chatgpt, codex and gpt-3 large language models. Ieee Access, 11:45181–45193, 2023. 1

[35] L. McNabb and R. S. Laramee. Survey of surveys (sos)-mapping the landscape of survey papers in information visualization. In computer graphics forum, vol. 36, pp. 589–617. Wiley Online Library, 2017. 7

[36] D. Moritz, C. Wang, G. L. Nelson, H. Lin, A. M. Smith, B. Howe, and J. Heer. Formalizing visualization design knowledge as constraints: Actionable and extensible models in draco. IEEE transactions on visualization and computer graphics, 25(1):438–448, 2018. 1

[37] T. Munzner. A nested model for visualization design and validation. IEEE transactions on visualization and computer graphics, 15(6):921–928, 2009. 1

[38] T. Munzner. Visualization analysis and design. CRC press, 2014. 1

[39] A. Rind, W. Aigner, M. Wagner, S. Miksch, and T. Lammarsch. Task cube: A three-dimensional conceptual space of user tasks in visualization design and evaluation. Information Visualization, 15(4):288–300, 2016. 1

[40] S. F. Roth, J. Kolojejchick, J. Mattis, and J. Goldstein. Interactive graphic design using automatic presentation knowledge. In Proceedings of the SIGCHI conference on Human factors in computing systems, pp. 112–117, 1994. 1

[41] B. Saket, A. Endert, and Ç. Demiralp. Task-based effectiveness of

basic visualizations. IEEE transactions on visualization and computer graphics, 25(7):2505–2512, 2018. 1
[42] M. Sedlmair, M. Meyer, and T. Munzner. Design study methodology: Reflections from the trenches and the stacks. IEEE transactions on visualization and computer graphics, 18(12):2431–2440, 2012. 1
[43] L. Shen, E. Shen, Y. Luo, X. Yang, X. Hu, X. Zhang, Z. Tai, and J. Wang. Towards natural language interfaces for data visualization: A survey. IEEE transactions on visualization and computer graphics, 29(6):3121–3144, 2022. 1
[44] B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In The craft of information visualization, pp. 364–371. Elsevier, 2003. 1
[45] M. Sun, P. Mi, C. North, and N. Ramakrishnan. Biset: Semantic edge bundling with biclusters for sensemaking. IEEE transactions on visualization and computer graphics, 22(1):310–319, 2015. 3
[46] Y. Tian, W. Cui, D. Deng, X. Yi, Y. Yang, H. Zhang, and Y. Wu. Chartgpt: Leveraging llms to generate charts from abstract natural language. IEEE Transactions on Visualization and Computer Graphics, 2024. 1
[47] L. Wang, S. Zhang, Y. Wang, E.-P. Lim, and Y. Wang. Llm4vis: Explainable visualization recommendation using chatgpt. arXiv preprint arXiv:2310.07652, 2023. 1
[48] G. Wu, S. Guo, J. Hoffswell, G. Y.-Y. Chan, R. A. Rossi, and E. Koh. Socrates: Data story generation via adaptive machine-guided elicitation of user feedback. IEEE Transactions on Visualization and Computer Graphics, 30(1):131–141, 2023. 3
[49] T. Wu, Y. Yao, Y. Duan, X. Fan, and H. Qu. Networkseer: Visual analysis for social network in moocs. In 2016 IEEE Pacific visualization symposium (pacificvis), pp. 194–198. IEEE, 2016. 3
[50] Y. Wu, S. Gao, S. Zhang, X. Dou, X. Wang, and Q. Li. From requirement to solution: Unveiling problem-driven design patterns in visual analytics. IEEE Transactions on Visualization and Computer Graphics, 2025. 2
[51] M. Xia, R. P. Velumani, Y. Wang, H. Qu, and X. Ma. Qlens: Visual analytics of multi-step problem-solving behaviors for improving question design. IEEE Transactions on Visualization and Computer Graphics, 27(2):870–880, 2020. 3
[52] X. Xie, F. Du, and Y. Wu. A visual analytics approach for exploratory causal analysis: Exploration, validation, and applications. IEEE Transactions on Visualization and Computer Graphics, 27(2):1448–1458, 2020. 3
[53] B. Xu, A. Yang, J. Lin, Q. Wang, C. Zhou, Y. Zhang, and Z. Mao. Expertprompting: Instructing large language models to be distinguished experts. arXiv preprint arXiv:2305.14688, 2023. 2
[54] K. Xu, J. Yuan, Y. Wang, C. Silva, and E. Bertini. Mtseer: Interactive visual exploration of models on multivariate time-series forecast. In Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems, pp. 1–15, 2021. 1
[55] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, et al. A survey of large language models. arXiv preprint arXiv:2303.18223, 2023. 1

# A User Study

We conducted a controlled lab user study to evaluate *LLM4VA*, focusing on assessing the usability and effectiveness of the system in supporting the development of VA prototypes. The study aimed to determine how well *LLM4VA* facilitates the creation and iteration of voice assistant prototypes, examining both the practical utility of the tool and users' ability to successfully accomplish their development goals through the interface.

## A.1 Participants

We recruited 11 participants (**P1–P11**) via social media and word-of-mouth, comprising individuals aged 21–28 years (M = 23.09, SD = 1.87), with a gender distribution of 8 male, 2 female, and 1 participant who preferred not to disclose. All participants were experienced developers with 1–5 years of prior work in VA system development (M = 2.41, SD = 1.36) and demonstrated proficiency in D3.js syntax.

## A.2 Dataset

We provided participants with U.S. patent data spanning 1989 to 2020, sourced from a visualization course final project at our partner university.

## A.3 Procedure

The study began by collecting participant demographics and introducing the study's purpose. Participants then viewed a 15-minute demonstration video, with opportunities to ask clarifying questions from support staff. After confirming comprehension, they engaged in a 10-minute unstructured exploration of the dataset, followed by a 10-minute session to articulate their analytical goals. Subsequently, participants used the *LLM4VA* system to complete a VA prototype development task. Upon task completion, participants provided feedback via a 5-point Likert scale questionnaire (1: strongly disagree, 5: strongly agree; see Fig. 4) and a semi-structured interview. A coordinator conducted in-person, one-on-one sessions, observing and documenting participant interactions. With consent, sessions were recorded for post-study analysis. Participants received $15 compensation upon completion.

## A.4 Result

We collected participants' feedback on the *LLM4VA* system through questionnaires (Fig. 4) and semi-structured interviews. Participants expressed predominantly positive perceptions of the system, with many acknowledging its effectiveness in streamlining VA prototype development.

**On the *Usefulness* (Q1–6).** All participants (11/11) agreed that *LLM4VA* meets VA development needs and improves development efficiency. **P2** stated, "*[The system] effectively refines vague analytical directions into concrete tasks, which is very helpful for me.*" **P9** shared a similar sentiment: "*I can't believe I got a code-based VA demo in less than an hour, and I can easily iterate on it.*" Additionally, all participants (11/11) acknowledged the system's well-structured workflow and the rationality of its multi-agent design. **P10** commented, "*[The workflow] covers everything from data analysis and requirement extraction to design and implementation, providing great support throughout the process.*" **P3** particularly appreciated the multi-agent framework, stating, "*The multi-agent setup makes the workflow clearer, helping me understand what each part is doing. This ultimately improves the quality of the generated VA prototype. If I were only interacting with a single LLM, I doubt I could achieve the same results.*"

**On the *Ease of Learning & Use* (Q7–9).** All participants (11/11) found the system intuitive and easy to use. **P1** noted, "*The interface is clean, the operations are simple, and it is easy to get started.*" **P3** and **P6** pointed out that the system clearly displays the output of each agent. **P6** mentioned, "*[The system] uses a node-link diagram to illustrate the relationships between data, requirements, and views, making it very intuitive.*" Additionally, **P4**, **P10**, and **P11** highlighted that the system allows them to flexibly adjust agent outputs, whether through interactive dialogues or manual modifications. **P11** particularly praised the interface and interaction design as "*There are rich interaction options at every stage, allowing me to actively engage in the development process rather than just passively accepting the LLM's output.*"

**On the *Satisfaction* (Q10–12).** All participants (11/11) expressed willingness to frequently using *LLM4VA* for VA prototype development and were satisfied with its interactive experience. The majority of participants (9/11) were also content with the final VA prototype generated by the system, with the exception of **P3** and **P5**, who provided some suggestions for improvement. **P3** noted that the readability of the final prototype could be enhanced: "*The legends are not presented intuitively, which increases the cognitive effort needed to explore the system.*" And **P5** commented, "*The final output works well as an initial prototype, but its representation*

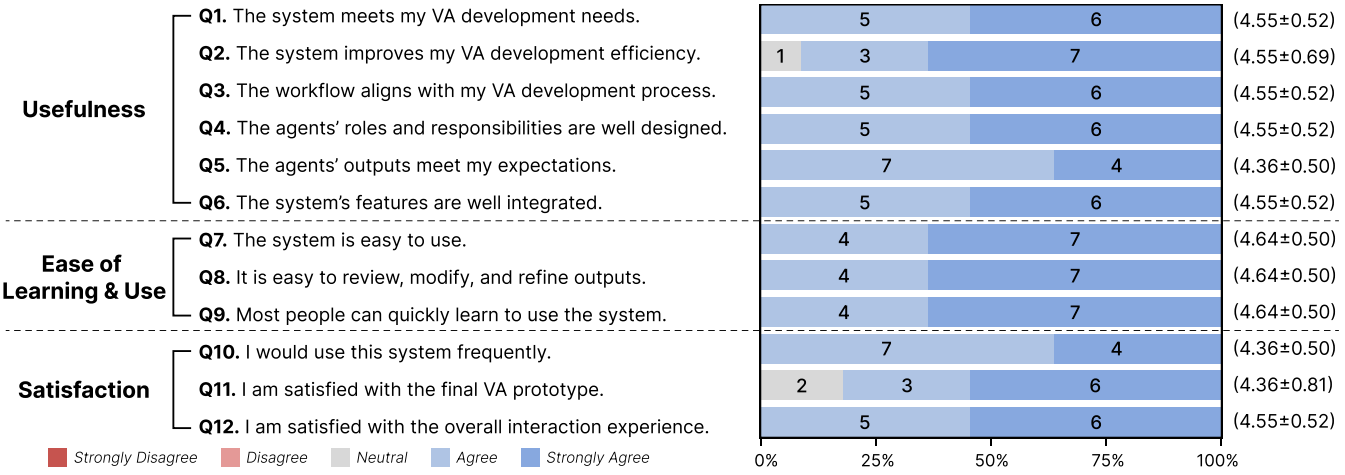


Figure 4: Assessment of our system in terms of *Usefulness* (**Q1–6**), *Ease of Learning & Use* (**Q7–9**), and *Satisfaction* (**Q10–12**). Note the right column displays the (average ± standard deviation).

*of relationships between views could be improved, possibly due to current LLM limitations.*"

**Opportunities for Improvement.** Despite the benefits mentioned, participants also provided some suggestions for improving the *LLM4VA* system. Some participants (**P2**, **P3**, **P6**, **P8**, **P11**) suggested that the final VA prototype should pay more attention to labeling legends and view titles to reduce the cognitive effort required for users to understand the prototype. **P6** noted, "*Although I can further interact with the system to refine the VA prototype, I would prefer if the initial version already had a well-labeled and easy-to-understand structure.*" We believe this issue could be directly addressed by refining the prompt instructions for the *coder* agent. Additionally, **P1**, **P7** and **P9** encountered interruptions in the LLM's responses during the code generation phases, where the LLM agent needed to response a large number of tokens. **P7** remarked, "*The response was cut off in the middle of an incomplete code snippet, and I had to manually prompt it to continue, which was a bit frustrating.*" To address this issue, we propose implementing an automatic retry mechanism that detects when a response is interrupted and prompts the LLM to complete it, reducing the need for manual intervention. Participants also suggested other potential improvements, such as providing multiple design alternatives when generating VA prototypes (**P10**) and enhancing the system's adaptability to user preferences (**P8**).

Despite these areas for improvement, participants remained optimistic about the system's potential. They acknowledged that while certain refinements could enhance the experience, *LLM4VA* already represents a significant step forward in leveraging LLMs for VA prototype development, with great potential for future advancements.

## B VA Knowledge Base Construction

To construct the VA knowledge base, we collect research papers from leading conferences and journals in the visualization domain, extracting key knowledge from these sources.

**Paper Collection and Filtering.** We collect papers published up until 2025 in IEEE VIS, VAST, InfoVis, PacificVis, and TVCG. A keyword-based search strategy [35] is employed, using terms such as "*visual analytics*", "*visual analysis*" and "*visualization*". We then apply several filtering criteria to ensure the relevance and quality of the collected papers:

- ... must utilize real (i.e., non-synthetic) data.
- ... must clearly define VA design requirements.
- ... must include at least one system interface.
- ... must provide explicit textual descriptions of the data, analytical tasks (requirements), and visualization design.

After a thorough review by two authors, we finalize the selection, retaining 573 papers. An overview of the collected papers and their sources is provided in Tab. 1.

Table 1: The number of collected papers from various sources.

| VIS | VAST | InfoVis | PacificVis | TVCG | **Total** |
|---|---|---|---|---|---|
| 151 | 215 | 39 | 80 | 88 | 573 |

**Knowledge Extraction.** To enhance retrieval accuracy and minimize the impact of irrelevant information, we extract three key elements from each paper: 1) **Data:** A description of the datasets used in the paper. 2) **Requirement:** A description of the specific analytical tasks or problems the paper addresses. 3) **Design:** A description of the system design proposed in the paper. The extraction process is facilitated by LLMs, specifically *GPT-4o*, which analyzes the full text of each paper and automatically identifies and structures relevant information. To mitigate potential misinterpretations, we explicitly instruct the model to prioritize "excerpt" over "summary", preserving as much of the original wording as possible. After extraction, two researchers manually review and refine the LLM-generated results to ensure accuracy and consistency. Finally, the verified structured information is encoded into a document database for efficient retrieval.

## C VA PROTOTYPES IN SYSTEM WALKTHROUGH

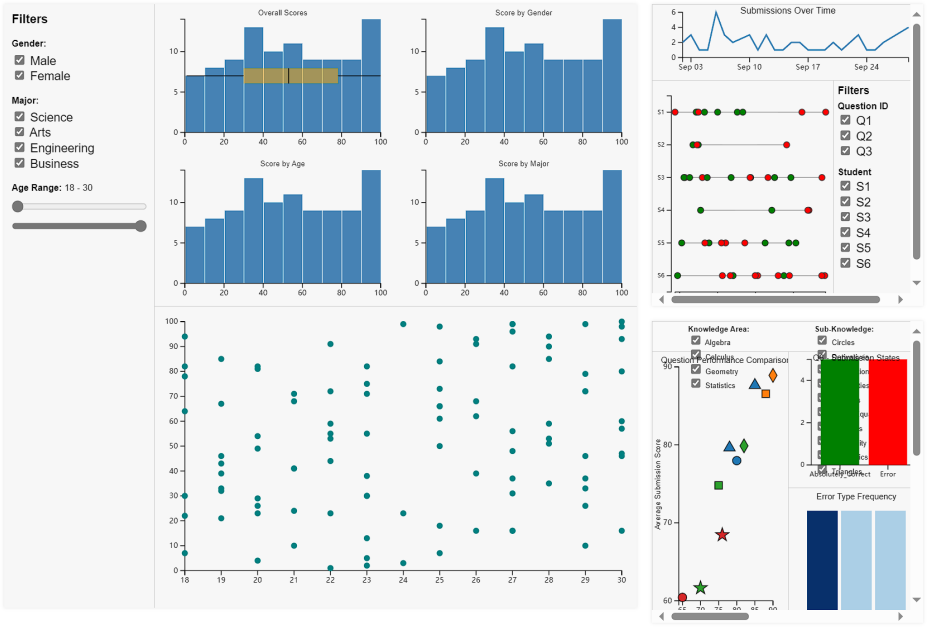


Figure 5: Initial VA prototype in system walkthrough

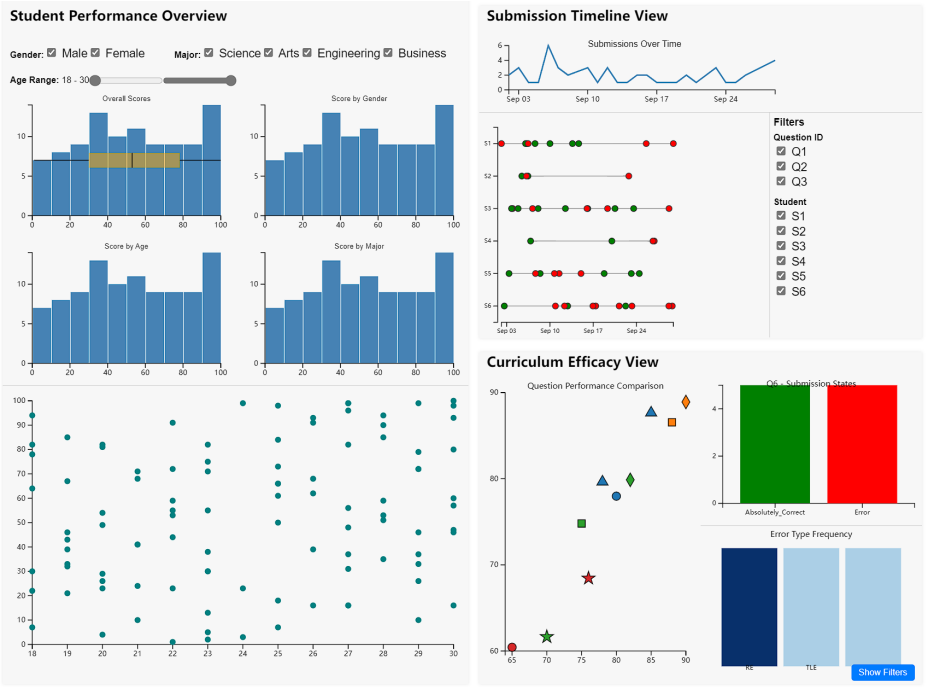


Figure 6: VA Prototype after setting layout

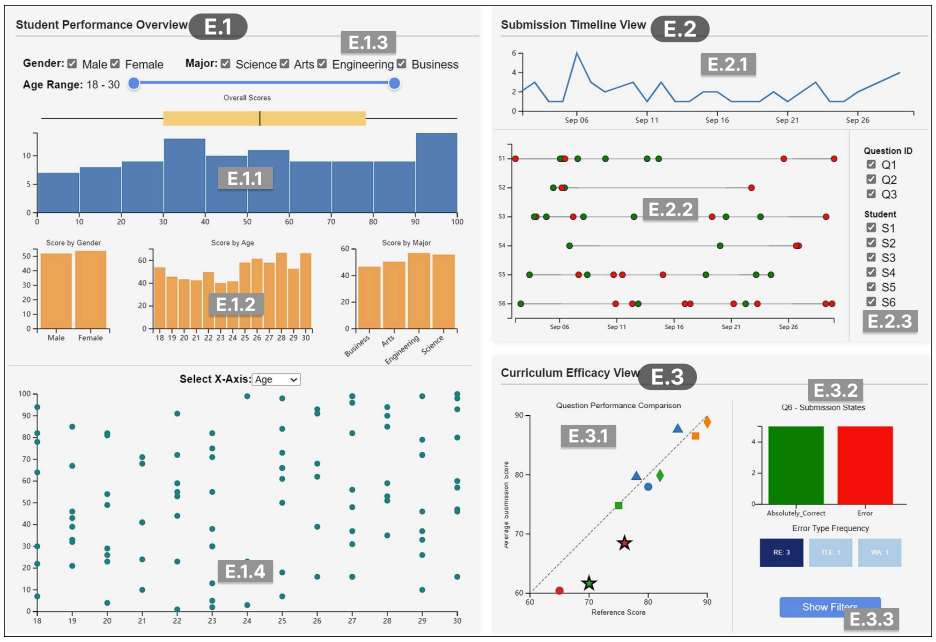


Figure 7: VA Prototype after manual adjustment

◇ *Student Performance Overview*(Fig. 7 (E.1)) provides a macro-level assessment of students' overall performance by integrating submitted grade data with demographic information. This view includes a comprehensive statistical chart (E.1.1), where a histogram is overlaid with a box plot, visually presenting the overall distribution of students' average scores, along with key statistical indicators such as mean, median, and score range. Additionally, three separate histograms (E.1.2) display score distributions categorized by gender, age, and major, allowing users to compare academic performance across different demographic groups. To enhance interactivity, the view features a control panel (E.1.3) that includes checkboxes and sliders, enabling users to dynamically filter data and focus on specific groups. Furthermore, a scatter plot (E.1.4) is provided, where each point represents an individual student. The x-axis encodes the selected demographic attribute, while the y-axis represents the student's average score, allowing users to intuitively identify performance trends and detect potential outliers.

◇ *Submission Timeline View* (Fig. 7 (E.2)) is designed to explore students' iterative submission patterns and track their learning progress over time. A line chart (E.2.1) visualizes the number of submissions per time unit (e.g., daily or weekly), revealing trends in students' submission behaviors over time. At the core of this view is the timeline (E.2.2), which sequentially displays submission events along a horizontal axis. Each student's submission is represented by a circular icon, color-coded based on submission status (e.g., incorrect attempts vs. absolutely correct submissions). When users click or hover over a circular icon, a tooltip appears, providing detailed metrics such as submission score, time spent, and memory usage. Additionally, users can interactively refine the view by brushing a specific time range on the line chart (E.2.1), dynamically updating the timeline to focus on the selected period. A filter panel on the side (E.2.3) allows users to focus on specific problems or students, ensuring a more tailored and insightful exploration of submission behaviors.

◇ *Curriculum Efficacy View* (Fig. 7 (E.3)) evaluates instructional effectiveness of course content by comparing students' actual performance on specific questions with reference scores. Its interactive dual-panel design includes: the left panel (E.3.1) displays a scatter plot where each dot represents a question, plotted by reference score (x-axis) versus students' average score (y-axis). Questions are color-coded by knowledge domain and marked by shape for subcategories, with outliers visually highlighted via bordered markers. When users hover over any data point, the right panel (E.3.2) dynamically updates to display diagnostic details (e.g., submission status distributions, frequent error patterns), helping instructors identify learning gaps. A multi-dimensional filter (E.3.3) allows to refine the analysis by selecting specific knowledge domains or subcategories for targeted analysis.