A System Design for SVG Generation from Text using Large Language Models for the Kaggle ”Drawing with LLMs” Competition

Edward Julian García Gaitán
Systems Engineering Program
Universidad Distrital Francisco José de Caldas
Bogotá, Colombia
ejgarciag@udistrital.edu.co

Jaider Camilo Carvajal Marin
Systems Engineering Program
Universidad Distrital Francisco José de Caldas
Bogotá, Colombia
jccarvajalm@udistrital.edu.co

Nelson David Posso Suárez
Systems Engineering Program
Universidad Distrital Francisco José de Caldas
Bogotá, Colombia
ndpossos@udistrital.edu.co

***Abstract*—The Kaggle competition ”Drawing with LLMs” challenges participants to generate Scalable Vector Graphics (SVG) from textual descriptions using Large Language Models (LLMs). This paper presents a comprehensive systems analysis and design for such a system. We outline system requirements, propose a modular software architecture, and detail strategies to manage the inherent sensitivity and potential chaotic behaviors of LLMs in a generative context. The objective is to develop a robust and efficient methodology for translating textual prompts into valid and semantically coherent SVG images, addressing the complexities of this interdisciplinary task.**

***Index Terms*—Large Language Models, SVG Generation, Systems Engineering, System Design, Kaggle Competition, Chaos Theory, Prompt Engineering.**

I. Introduction

The ”Drawing with LLMs” competition [1], hosted by Kaggle, represents a significant challenge in artificial intelligence: the automated generation of Scalable Vector Graphics (SVG) images directly from natural language prompts using Large Language Models (LLMs) [2]. This task is situated at the confluence of Natural Language Understanding (NLU), automated code generation, and computer graphics synthesis, presenting notable systemic complexity. As an open system, it must process diverse inputs (prompts) and its outputs are subject to external evaluation.

Previous work in this domain highlights challenges in maintaining semantic consistency between textual descriptions and graphical outputs, ensuring the syntactic validity of the generated SVG code, and achieving robustness against ambiguous or novel prompts. The inherent variability and potential for unpredictable (”chaotic”) behavior in LLMs [5] further complicate this task. Early analysis, conducted as part of a Systems Analysis and Design course workshop, identified these critical sensitivities and underscored the need for a design that explicitly addresses them from a systems engineering perspective [4].

This paper details a system design intended to address these challenges. We propose a software architecture and an implementation methodology aimed at efficiently and reliably translating textual prompts into visually coherent and syntactically valid SVGs. The core objective is to develop a system capable of competitive performance by proactively managing the complexities inherent in LLM-based SVG generation.

The proposed solution draws upon established concepts in LLMs [2], SVG standards [3], systems engineering [4], and an understanding of chaos theory [5] to inform design choices. While image quality metrics like SSIM [6] are relevant for raster images, for SVG, the focus is on syntactic validity and semantic fidelity to the prompt.

II. Proposed System Design and Methodology

The methodology for designing and developing the SVG generation system is rooted in systems engineering principles and an iterative development approach. Key architectural principles include modularity, separation of concerns, high cohesion, low coupling, and testability.

A. Functional and Non-Functional Requirements

The system is designed to meet specific functional (FR) and non-functional (NFR) requirements:

- **FRs:** Prompt ingestion and parsing (FR1), advanced semantic interpretation (FR2), coherent and valid SVG code generation (FR3), flexible configuration of generation parameters (FR4), and results persistence (FR5).
- **NFRs:** Key NFRs include performance (generation latency ¡ 1s - an ambitious target, NFR1), reliability and accuracy (100% valid SVGs, high semantic coherence, NFR2), scalability (processing up to 10,000 prompts,

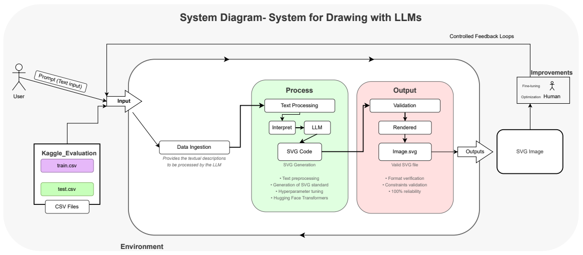

Figure 1. Conceptual System Architecture for LLM-based SVG Generation. The diagram illustrates the flow from user input (prompts) through processing stages (Input, Preprocessing, LLM Core, Postprocessing) to the final SVG output, with a feedback loop for improvements.

NFR3), usability for developers (NFR4), maintainability and modularity (NFR5), and robustness with error handling (NFR6).

B. Strategies for Sensitivity Management and Chaos Mitigation

Given the sensitivity of LLMs to input variations and their potential for non-linear outputs, the following strategies are central to the design:

1) **Robust Prompt Engineering (M2):** Involves prompt normalization, standardization, templating, and dynamic few-shot learning techniques to guide the LLM and reduce spurious variability.
2) **Controlled LLM Generation (M3):** Includes fine-tuning inference parameters, employing multiple sampling strategies with selection based on quality heuristics (using M5), and implementing intelligent retry mechanisms for failed generations. Clear stop sequences and length limits are also enforced.
3) **Defensive SVG Postprocessing (M4):** Implements multi-level validation (XML syntax, SVG profile conformance, attribute validity), heuristic error correction rules for common LLM output patterns, and complexity limits (circuit breakers) to prevent overly complex or invalid SVGs.
4) **Continuous Monitoring and Feedback (System-wide):** Detailed structured logging (e.g., JSON) at each pipeline stage, monitoring of process and quality metrics, and establishing a root cause analysis process for systematic failures are planned to enable iterative improvement. Human-in-the-loop evaluation is considered during development.

C. Technology Stack

The recommended technology stack is centered around Python for its mature AI/ML ecosystem. Key components are detailed in Table I.

D. Implementation Plan Summary

A 8-week iterative phased development approach is envisioned:

Table I
PROPOSED TECHNOLOGY STACK

Category	**Technology/Tool**	**Justification**
Main Language	Python 3.9+	Mature AI/ML/NLP ecosystem [7].
NLP/LLMs	Hugging Face Transformers [2], Sentence Transformers	Access to SOTA LLMs; inference/fine-tuning tools. Embeddings.
SVG Handling	lxml [8], svgwrite [9], (cairosvg)	Efficient XML/SVG parsing, validation, creation. Rasterization for preview.
Data Handling	Pandas	Tabular data manipulation (prompts).
Versioning	Git/GitHub	Collaborative development.
Testing	PyTest	Robust Python testing framework.

- **Week 11 (Phase 1):** Core Prototyping. Setup and minimal end-to-end prototype to validate feasibility.
- **Week 12 (Phase 2):** Central Module Development. Robust implementation of M1-M4.
- **Weeks 13-14 (Phase 3):** Robustness Quality. Implement sensitivity/chaos mitigation strategies, M5.
- **Weeks 15-16 (Phase 4):** Optimization Documentation. Performance tuning, scalability tests.
- **Weeks 17-18 (Phase 5):** Final Testing Packaging. Comprehensive testing, bug fixing, Kaggle submission prep.

This plan is a guideline and subject to adjustments based on development progress.

III. EXPECTED OUTCOMES AND DISCUSSION

The primary outcome of this design effort is a detailed blueprint for a system capable of generating SVGs from textual prompts. The success of the implemented system will be measured by its ability to meet the defined functional and non-functional requirements, particularly NFR1 (Performance: ¡1s/SVG), NFR2 (Reliability: 100% valid SVGs, high semantic coherence), NFR3 (Scalability: 10k prompts), and NFR6 (Robustness).

The proposed modular architecture (Fig. 1) and the explicit strategies for managing LLM sensitivity are key strengths. Modularity enhances maintainability and allows for parallel development, while techniques like advanced prompt engineering and defensive postprocessing aim to improve output predictability and quality. The Python-based technology stack (Table I) provides a robust foundation.

However, several challenges are anticipated. Achieving the sub-second latency target (NFR1) will be demanding and highly dependent on the chosen LLM's efficiency and the complexity of the generated SVGs. Interpreting highly ambiguous or abstract prompts will remain a significant hurdle for NLU components. The system's performance may also be subject to the reliability and versioning of external LLM APIs or models. The iterative implementation plan is designed to address these challenges progressively, allowing for empirical validation and refinement at each phase. Continuous monitoring and feedback loops will be crucial for adapting the system and improving its performance over time.

IV. Conclusion and Future Work

Significant progress has been made in establishing the foundational elements of the project, effectively preparing us to move into the implementation phase. The conceptual framework, system architecture, and strategic approaches are clearly defined. The iterative process, especially through Workshops 1 and 2, was crucial in clarifying both functional and non-functional requirements, as well as in developing robust strategies to manage and mitigate the chaotic behaviors and sensitivities inherent in using Large Language Models (LLMs) for creative generation tasks.

The importance of adopting a systemic perspective to address the complexities of the ”Drawing with LLMs” competition has been emphasized. Systems engineering principles helped structure the problem, design modular components, and anticipate interactions-key aspects for AI-driven projects with nonlinear dynamics. Additionally, understanding chaos theory enhanced our insight into the system's potential behavior, particularly how small variations in input prompts can lead to diverse outputs, a characteristic intrinsic to LLMs. This awareness informed the design of defensive mechanisms and adaptive strategies within the proposed architecture.

The successful development of this comprehensive design document reflects a rich learning environment supported by lectures, academic resources, and the professor's guidance. This collaborative context was essential in shaping the analysis and fostering the robust system design presented. We are now well-positioned to begin practical implementation with confidence in the thoroughness of the preparatory design and analysis.

The key next work or step are implemente the LLM model by Huggin Face.

References

[1] Kaggle, ”Drawing with LLMs,” [Online]. Available: https://www.kaggle.com/competitions/drawing-with-llms/overview. Accessed: May 16, 2025.

[2] Hugging Face, ”Transformers Documentation,” [Online]. Available: https://huggingface.co/docs/transformers/. Accessed: May 16, 2025.

[3] W3C, ”Scalable Vector Graphics (SVG) 1.1 (Second Edition),” W3C Recommendation, Aug. 16, 2011. [Online]. Available: https://www.w3.org/TR/SVG11/

[4] D. H. Meadows, *Thinking in Systems: A Primer*. White River Junction, VT: Chelsea Green Publishing, 2008.

[5] J. Gleick, *Chaos: Making a New Science*. New York, NY: Penguin Books, 1987.

[6] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, ”Image quality assessment: From error visibility to structural similarity,” *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600-612, Apr. 2004.

[7] J. VanderPlas, *Python Data Science Handbook*. Sebastopol, CA: O'Reilly Media, 2016.

[8] lxml.de, ”lxml – Processing XML and HTML with Python,” [Online]. Available: https://lxml.de/. Accessed: May 16, 2025.

[9] The svgwrite Core Developers, ”svgwrite Documentation,” [Online]. Available: https://svgwrite.readthedocs.io/. Accessed: May 16, 2025.