DRAWING

WITH LLMS

Context of the problem

Goal: Generate SVG images from text prompts that describe a scene or object.

Input: Natural language prompt (e.g., "A red circle inside a blue square").

Output: Valid SVG code that visually represents the prompt.

Input/Output constraints

Input:

- A single, natural language prompt describing an image
- 200 characters max

Output:

- A self-contained SVG string (valid XML format)
- 10000 byte limit
- No external dependencies

SVG fidelity score (Evaluation metrics)

- Assesses how closely the generated SVG resembles the intended image described by the text prompt
- Visual accuracy (does the image “look right”)
- Structural correctness of SVG elements (shapes, positions, relationships)
- Semantic alignment with the original text prompt

Sensitivity and chaos

⚠️ Sensitivity Factors

* Small changes in the prompt wording can cause large shifts in SVG layout or shape count

- Model's tokenization may emphasize irrelevant words, altering output

🌪️ Chaos Factors

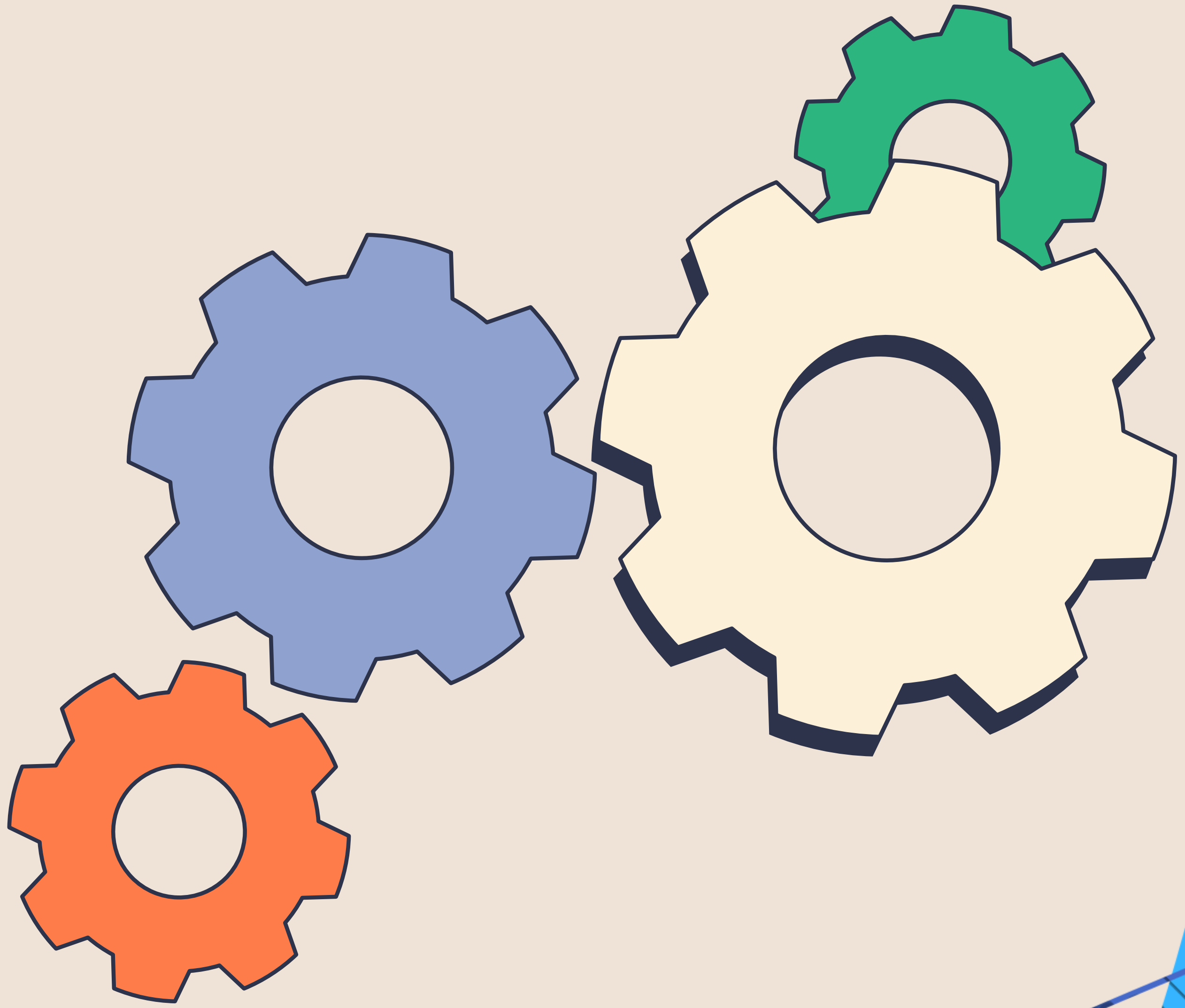
- LLMs may hallucinate SVG tags or structure (e.g., invalid or unrenderable code)
- Overlaping.

Functional requirements

The system must respect the contest restrictions and generate functional SVG images from text.

Non-functional requirements

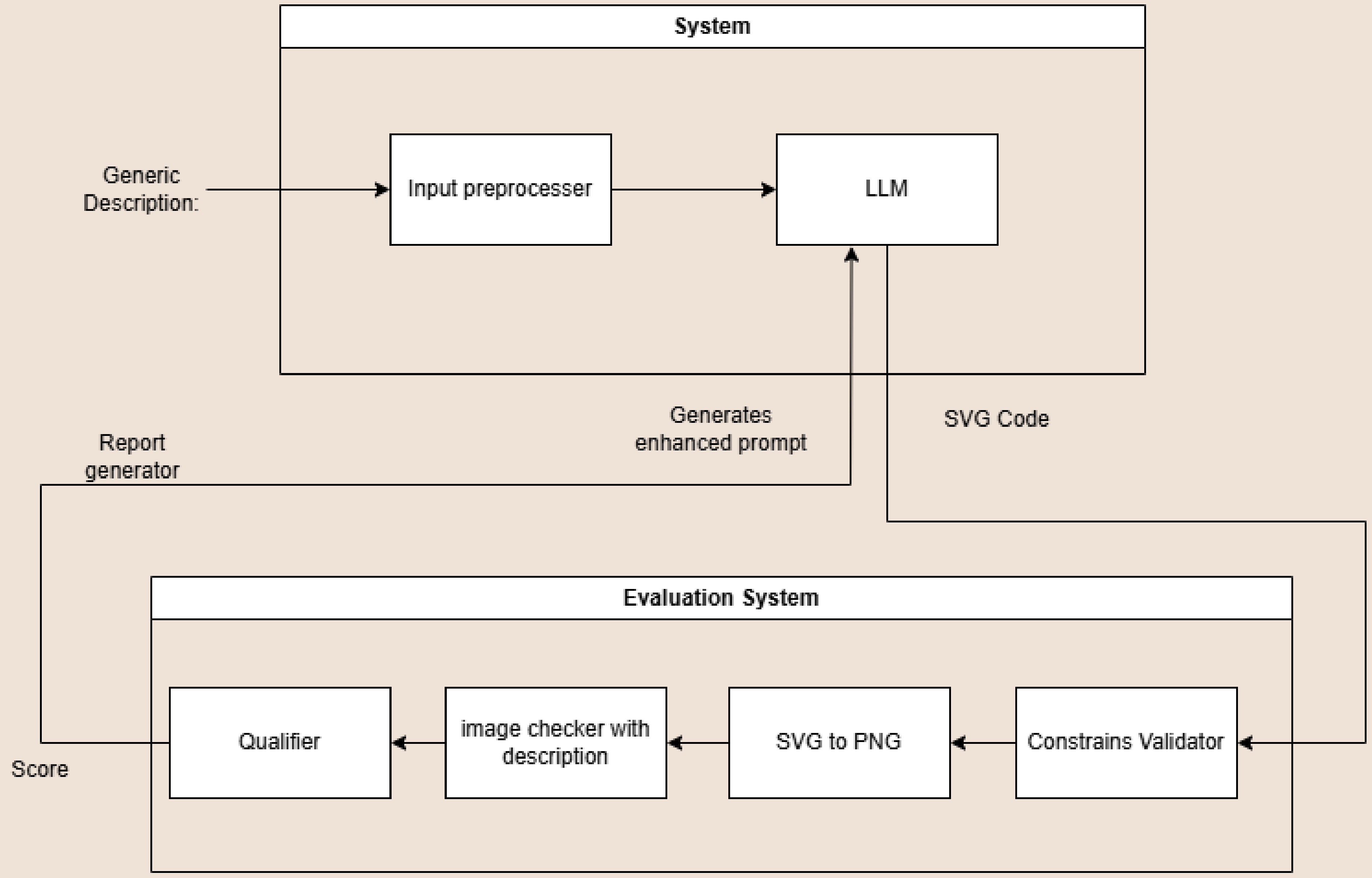
The system must be fast, reliable, efficient and modular.



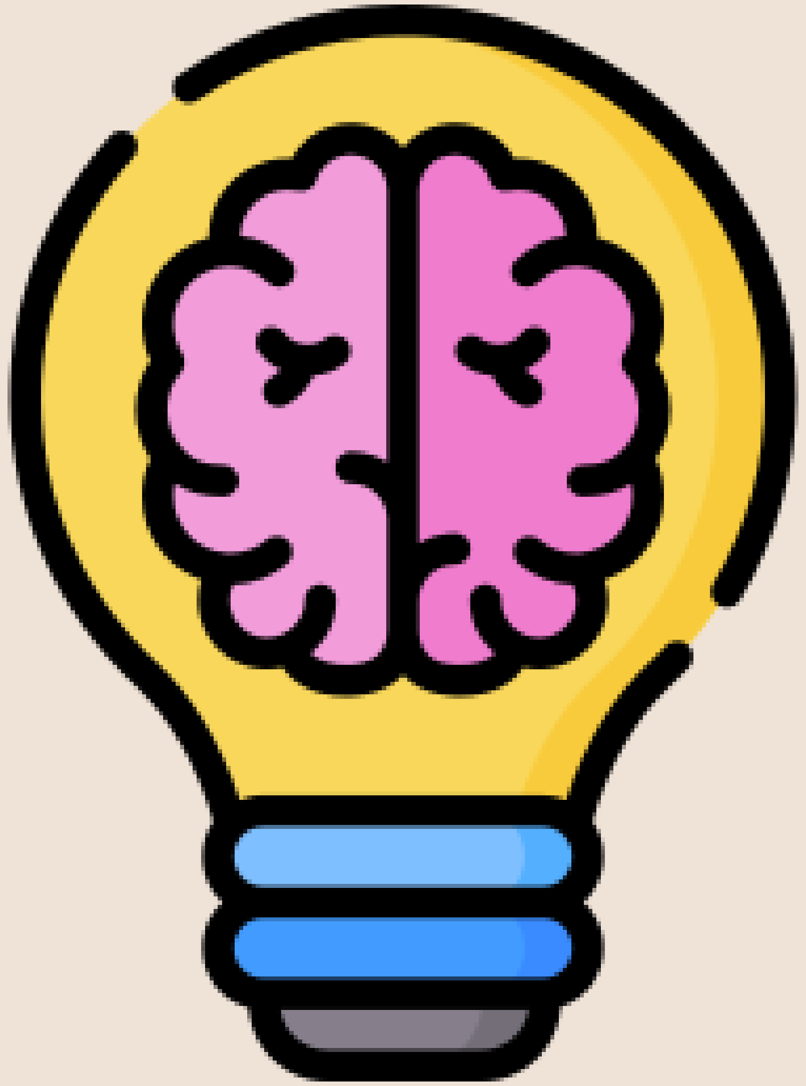
Main components

We divide the system into four parts: Input Handler, Prompt Engineer, LLM Generator and a Feedback Loop. Each performs a key function and communicates with the others.

Architecture diagram and flow



Principles applied



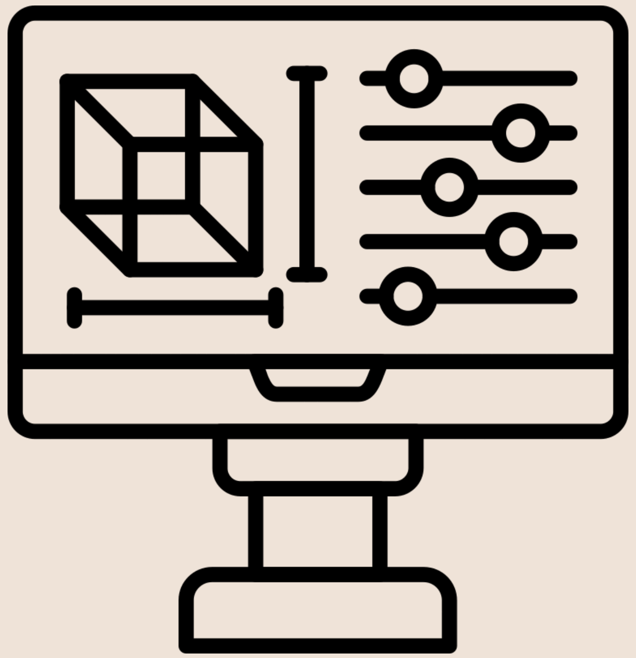
We use systems engineering principles: modularity to interchange parts, separation for functional clarity, and standardization of interfaces to integrate everything.

Strategies against sensitivity

Prompt engineering to avoid ambiguities and improve the understanding of the LLM from the start.

Simulation

objectives



- Validate our prompt generation strategies with actual LLM responses
- Test SVG constraint compliance with real outputs
- Analyze system behavior across different prompt templates
- Identify bottlenecks in our generation pipeline

Limits

- Google Gemini 2.5 Flash API for cost-effectiveness
- 60 total simulations across 4 categories
- 3 distinct prompt templates for comparative analysis
- Real-time SVG constraint validation

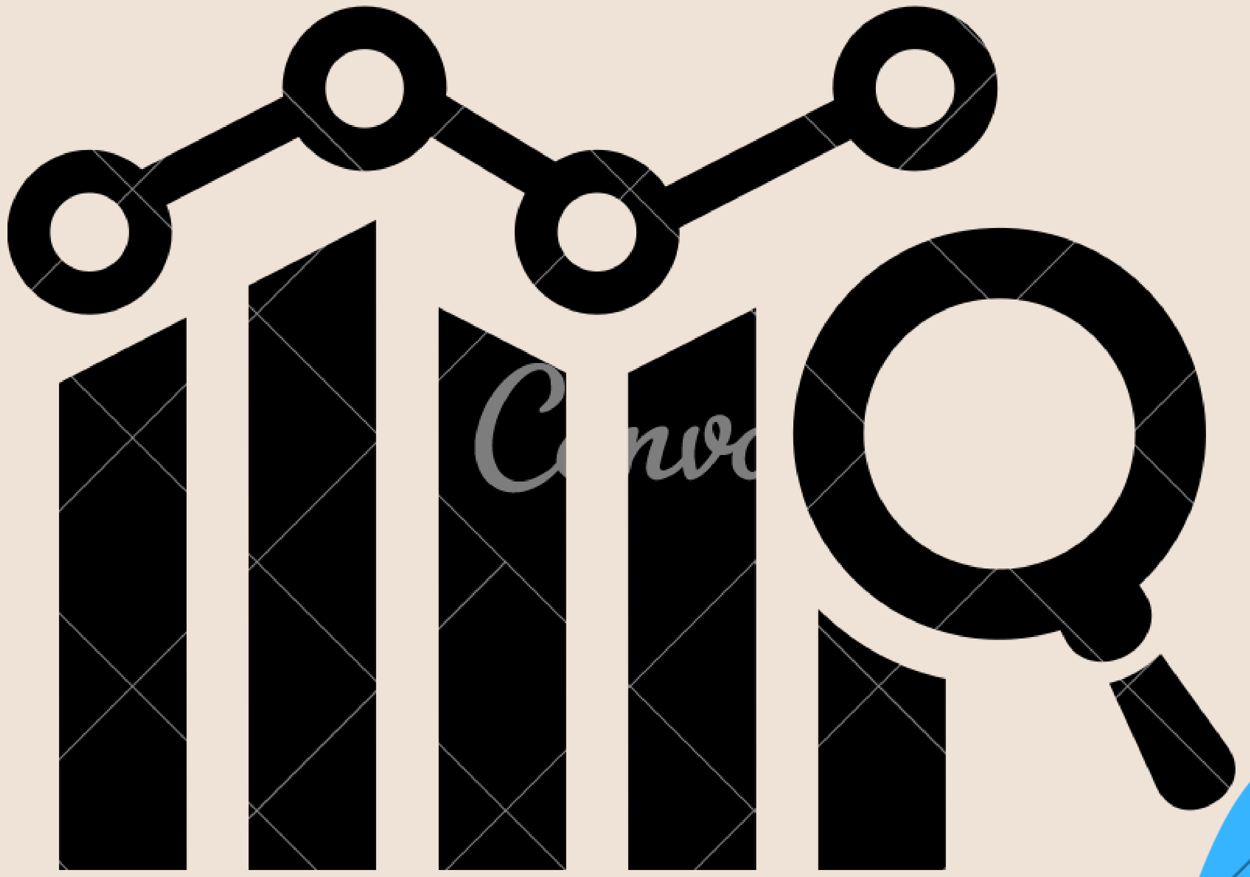
Unfortunately, we couldn't implement the feedback loop due to API costs and time limitations - each iteration would require additional expensive API calls.

Results

1. 76.67% compliance rate across all generations
2. Average generation time: 135ms - fast enough for real-time use
3. Average SVG size: 1,992 bytes - well within the 10,000-byte limit

Analysis

- Our almost all examples respected the 10,000-byte limit
- Visual coherence – images appropriately represented input descriptions
- Template influence – more specific templates produced better illustration quality



Implementing in Kaggle

The core implementation was structured into separate modules: prompt generation and LLM integration into a system called Model

Conclusion

- Comprehensive problem decomposition and requirements analysis
- Modular, extensible architecture addressing system sensitivity
- Validated simulation framework with quantitative performance metrics
- Practical implementation strategy for resource-constrained environments

Thanks for your attention