A Multi-Agent Approach for Iterative Refinement in Visual Content Generation

Achala Nayak[1*], **Adithya S Kolavi**[1*], **Nigel Teofilo Dias**[1*], **Srinidhi Somayaji P**[1*], **Ramamoorthy Srinath**[1]

[1]PES University

{achala.nayak2, adithyaskolavi, nigeldias27, srinidhisomayaji9113}@gmail.com, srinath@pes.edu

Abstract

Foundational image generation models like Stable Diffusion at producing high-quality images based on textual prompts. Although these models can generate strong visual content, they lack control beyond the initial prompt. The absence of post-editing features, including precise text alignment, layout customization, and image replacement, limits their functionality, sometimes leading to incorrect outcomes. This drawback is relevant to a variety of industries, such as the advertising industry where manual generation of posters, banners, etc. is expensive, laborious and time-consuming, therefore prone to repetition. To address these concerns, we propose a novel multi agent system that generates an initial image from text-based prompts and specified objects, followed by iteratively refining the initial image. The initial process of image generation involves segmenting the key objects from the provided images, crafting an overall narrative that describes the scenario in which these objects will be involved, and ultimately generating a complete image. The refinement process is accomplished using LLM and VLM based agents to look for visual cues that describe potential issues and apply changes to address those problems. This refinement process simulates the editing process humans apply to images while leaving the option open to manually edit any component of the image. Such a system enables people to rapidly generate consistent images with multiple objects or texts added like posters, banners or flyers. In conclusion, our agentic system enhances foundational image generation models by offering iterative refinement and post-editing capabilities. This approach improves control, making it ideal for industries like advertising, where it simplifies and accelerates the creation of visually consistent and customizable content.

Introduction

Text-to-image diffusion models have revolutionized automated image generation, with Stable Diffusion (Podell et al. 2023) leading to widespread adoption in creative workflows. Despite their impressive capabilities, these models primarily rely on text prompts for generation, offering limited user control during the creation process. When modifications are needed, artists must resort to post-processing techniques like inpainting or specialized architectures such as ControlNet (Zhang, Rao, and Agrawala 2023), making it particularly challenging to create content-centric images that require precise control over layout and branding elements.

This limitation significantly impacts the advertising industry, where visual content plays a crucial role in product marketing across social media and streaming platforms. The current process of creating such advertisements is resource-intensive, requiring multiple iterations between designers and stakeholders, often resulting in repetitive content that diminishes user engagement and brand value. Furthermore, existing approaches struggle with maintaining semantic consistency between product features, brand messaging, and visual elements.

Figure 1: Results from current generation models showing inconsistencies in text rendering, object placement, and brand coherence.

Our system addresses these challenges through a narrative-driven approach combining text and image-based prompts for initial generation, followed by an agentic refinement loop. This framework enables users to modify image components throughout the refinement process using an integrated editor application, while maintaining semantic relationships between visual elements. The system leverages compositional control to ensure that generated content aligns with brand guidelines and marketing objectives.

The key contributions of this work are:

- **Multi-Agent Architecture:** A novel system integrating narrative generation and visual analysis through specialized agents, enabling seamless collaboration between automated processes and human designers.
- **Iterative Refinement Loop:** A sophisticated mechanism combining VLMs and LLMs to progressively improve visual content while maintaining brand consistency and design coherence through automated critique and adjustment.

*These authors contributed equally.

- **Efficient Implementation:** A practical system running on modest hardware that demonstrates real-world viability through real-time interaction and significant reduction in content creation time.

Related Work

Our work builds upon several key areas in AI and computer vision, integrating them into a cohesive multi-agent system for visual content generation and editing.

Recent advances in Large Language Models (LLMs) such as LLaMA 2 (Touvron et al. 2023), Qwen (Bai et al. 2023), Mistral (Jiang et al. 2023),have demonstrated remarkable capabilities in reasoning and natural language understanding. These models, particularly when augmented with structured outputs and function calling capabilities through frameworks like OpenAI's Function Calling API and Anthropic's Constitutional AI, serve as effective reasoning engines in multi-agent systems.

Vision-Language Models (VLMs) have revolutionized multimodal understanding through architectures that enable sophisticated visual reasoning and cross-modal alignment. Models like CLIP (Liu et al. 2023), PaLI-X (Beyer et al. 2024), Idefics-2 (Laurençon et al. 2024), MANTIS (Jiang et al. 2024) have achieved remarkable performance in visual comprehension tasks. These models demonstrate sophisticated capabilities in visual content analysis, critique, and aesthetic evaluation through advanced architectures that combine transformer-based vision encoders with large language models. More efficient implementations like CogVLM (Wang et al. 2024a) and MiniGPT (Zhu et al. 2023) have made these capabilities accessible for real-time applications.

Text-to-image generation has evolved significantly with diffusion-based models like Stable Diffusion XL (Rombach et al. 2022) and Flux (Labs 2024). While these models excel at high-quality image synthesis, they typically lack fine-grained control over generation and post-editing capabilities. Recent research has explored controlled generation through attention manipulation (Li et al. 2019), ControlNet (Zhang, Rao, and Agrawala 2023), IP-Adapter (BeFrend 2023), and T2I-Adapter (Balaji et al. 2023). Post-generation editing capabilities have expanded through techniques like inpainting (AI 2024), outpainting, regional prompting , and mask-guided editing. Interactive refinement approaches like InstructPix2Pix and custom LoRA adaptations have further enhanced control over generated content.

In object detection and segmentation, the Segment Anything Model (SAM) (Kirillov et al. 2023), YOLO-World (Cheng et al. 2024), and Florence (Yuan et al. 2021) have revolutionized zero-shot capabilities, enabling robust object isolation without task-specific training. Fast SAM (Ravi et al. 2024) and Mobile SAM(Zhang et al. 2023) have made these capabilities more accessible for real-time applications. These advances support sophisticated image editing and generation workflows through reliable object extraction and manipulation.

Multi-agent systems have emerged as a powerful paradigm for complex AI tasks, with frameworks like ReAct (Yao et al. 2023), AutoGen (Wu et al. 2023), and LangChain(Chase 2022) providing structured approaches for agent collaboration. While these frameworks have demonstrated impressive capabilities in text-based reasoning and task completion, their integration with visual understanding remains limited. Recent work like MultiModal-Agent (Wang et al. 2024b) has begun exploring multi-agent systems with visual capabilities, but significant challenges remain in achieving sophisticated visual reasoning and manipulation comparable to their text-based counterparts.

Previous work in automated advertisement generation has primarily focused on layout generation (Zhu et al. 2024) and style transfer (Liu et al. 2023). While systems like LayoutDM (Chai, Zhuang, and Yan 2023) and LayoutDETR (Yu et al. 2024) use advanced architectures for graphic layouts, they typically lack iterative refinement capabilities and struggle with maintaining brand consistency.

Our work addresses these limitations by introducing a novel multi-agent framework that combines LLM reasoning, VLM understanding, and modern image generation capabilities, enabling iterative refinement through agent collaboration while maintaining visual consistency.

System Architecture

Our system architecture is designed with scalability, efficiency, and user experience as core principles. We employ a distributed microservices architecture that separates computational concerns while maintaining real-time interactive capabilities. Figure3 provides a high-level overview of our system architecture.

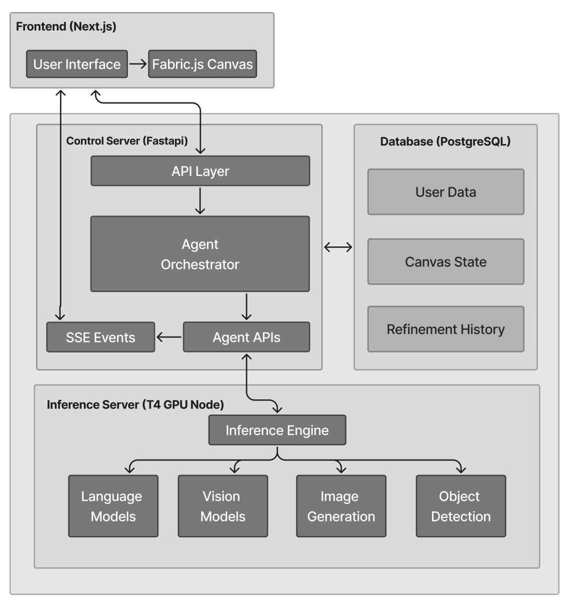

Figure 2: System Architecture

Frontend Application

The frontend is built using Next.js and Fabric.js, providing a Figma / Canva -like interface for image manipulation. Real-time communication between frontend and backend is achieved through RESTful APIs and Server-Sent Events (SSE), enabling users to observe and intervene in the agent's decision-making process.

Backend Services

The backend is structured into three optimized components:

Control Server A lightweight Fastapi server orchestrates the system by managing agent algorithms, user sessions, and event broadcasting. The server coordinates all system components while maintaining minimal resource usage.

Database Layer A serverless PostgreSQL instance stores canvas states, user data, and refinement history in JSON format, optimizing for both cost and scalability.

Model Inference Server Running on a T4 GPU node, this server efficiently hosts all AI models including LLMs for content generation, VLMs for image analysis, and object detection models. The models are optimized to share GPU memory, enabling simultaneous operation on a single GPU. we also tested out system with 3rd partly llm providers to compare peformance of out system

Scalability Considerations

Our architecture is designed for horizontal scalability at every layer:

1. **Frontend Scaling**: The Next.js application can be deployed across multiple edge locations using CDN services, ensuring low-latency access for users worldwide.
2. **Backend Scaling**: The control server can be replicated across multiple instances using Kubernetes, with load balancing handling request distribution. This allows for seamless handling of increased user load.
3. **Database Scaling**: The serverless PostgreSQL instance automatically scales based on demand, with read replicas available for handling increased query loads.
4. **Model Inference Scaling**: Additional GPU nodes can be added to the Kubernetes cluster as demand increases. The system automatically distributes inference requests across available nodes.

Data Flow

Our system's data flow is optimized for real-time interaction and state consistency across components. The process begins when a user initiates an action in the frontend editor, triggering the following sequence:

The frontend sends requests to the Control Server through REST APIs, which initiates the agentic refinement process. As the Agent Loop Orchestrator processes these requests, it performs three parallel operations: updating the canvas state in the database, making inference calls to the GPU server, and broadcasting state updates through SSE to the frontend.

The canvas state, stored as serialized JSON in PostgreSQL, contains all element attributes including position, style, and metadata. This JSON data is rendered into visual form by Fabric.js in the frontend, enabling efficient storage while maintaining element editability.

Real-time visibility into the agentic decision-making process is achieved through SSE connections, which stream agent thoughts, decisions, and state changes to the frontend's state viewer component. This transparency allows users to monitor and intervene in the refinement process while maintaining sub-second response times for user interactions.

The system handles both synchronous operations (like user edits) and asynchronous processes (like model inference) seamlessly, ensuring a responsive user experience while managing computationally intensive AI operations in the background. Error handling and state recovery are built into each step, maintaining system reliability during network issues or processing failures.

Methodology

Our research introduces a novel multi-agent system designed to automate and enhance the visual content creation process, particularly focusing on advertising materials. The system mirrors the iterative nature of human design workflows, where an initial concept is progressively refined through multiple iterations. What distinguishes our approach is the combination of sophisticated AI agents working in concert with human designers, enabling both automated refinements and manual interventions at any stage of the process.

Initial Image Generation

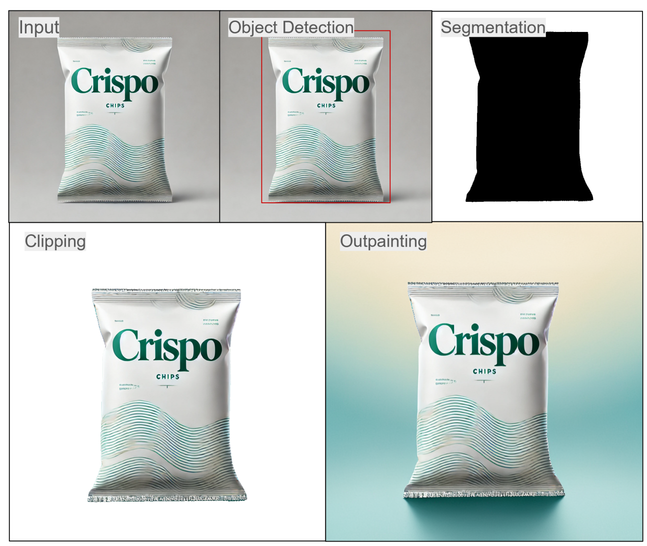

Figure 3: Initial Image Generation

The initial image generation process integrates multiple components to create a cohesive visual composition. This process begins with two key inputs: the objects to be featured (such as product images and logos) and a textual prompt describing the desired composition. Our Content Engine first analyzes these inputs to generate a narrative - a detailed description that serves as a creative blueprint for the image, as illustrated in Figure 3.

This narrative guides three sequential steps:

1. Object Integration: The Vision Engine employs the Segment Anything Model (SAM) (Kirillov et al. 2023) and Florence (Yuan et al. 2021) to segment and extract the input objects. These elements are then strategically positioned on the canvas according to the narrative's compositional guidelines.

2. Background Generation: Using state-of-the-art image generation models, we create a contextually appropriate background based on the narrative's atmospheric and stylistic requirements.

3. Element Composition: Finally, textual elements and additional graphical components are overlaid on the composition, completing the initial image. Each element's placement is determined by the narrative's specifications for visual hierarchy and brand consistency.

This layered approach ensures that the initial image provides a strong foundation for subsequent refinements while maintaining the flexibility needed for adjustments.

Refinement Loop

Algorithm1 shows the procedures called in a single iteration of the refinement loop while Figure 3 shows the entire workflow of the refinement process. A single iteration of the refinement loop, as shown in Algorithm 1, consists of Critic, Planning, Execution and Evaluation Agent. The Critic Agent visualizes the initial image and generates a report on the potential issues in the image. This report is passed onto the Planning Agent which generates a list of changes to be made to the image.

These changes are iterated over and the Execution Agent is called to pick a tool which can apply the change. Currently, the following tools have been implemented:

1. Text Tool: Changes the font style, size, color, stroke and drop shadow of the text
2. Image Tool: Generates a background for the image.
3. Layout Tool: Moves the object's position relatively or absolutely.

The tool picked by the Execution Agent is called to apply the change. The updated image is then sent to the Evaluation Agent to check if the change has been implemented correctly. If it has not been implemented correctly, the tool is again called and given the feedback from the evaluation agent. To prevent infinite execution of this inner 'eval and apply' loop, we set a 'maxiterations' variable which causes the inner loop to terminate once the number of iterations exceeds the value set in this variable.

All changes applied during this process along with the rationale of each of the agents are visible in real-time to humans interacting with the editor application using Server-Sent Events (SSE). Once all the changes generated by the Planning Agent have been iterated over i.e. one iteration of the refinement loop is completed, it waits for human feedback before proceeding to the next iteration. During this period, the user can make edits to any component of the image which they feel has not been addressed adequately by the refinement process. The edited image can be sent for the next iteration refinement loop if necessary or terminated by the user.

Algorithm 1: Agentic Loop

Input: Initial Image
Parameter: Narrative
Output: Refined Image

```
critic ← criticAgent(narrative, image)
changes ← planningAgent(critic, image)
Let i = 0
while i ≤ len(changes) do
    toolPicked←ExecutionAgent.toolPicker(changes[i])
    image←Tools.applyChange(image, changes[i], toolPicked)
    i ← i + 1
end while
applyChange(image, change, toolPicked)
maxIterations ← 3
if toolPicked = 'textTool' then
    i ← 0
    newImage ← Tools.applyTextTool(image, change)
    conclusion ← evaluationAgent(image, newImage)
    while i ≤ maxIterations and 'YES' not in conclusion do
        newImage←Tools.reapplyTextTool(image, change, conclusion)
        conclusion ← evaluationAgent(image, newImage)
        i ← i + 1
    end while
else if toolPicked = 'imageTool' then
    Similar to the text tool
else if toolPicked = 'layoutTool' then
    Same as the text tool
else
    Error: Invalid tool selected
end if
return image
```

Since multiple agents are involved in this process, we have experimented with various approaches of the refinement loop process to optimize the quality of output generated and speed of each iteration:

Approach 1 This was the first approach undertaken and the most rudimentary approach to refining the initial image. Here, the Critic Agent receives the initial image, analyzes the image and generates a list of potential issues along with suggestions on how to fix those issues. This list of suggestions are iterated over. In each iteration the execution agent picks the element that needs to be changed and the appropriate tool to fix the issue using function calling. The tool picked is called to apply the change. The tool's response is made using structured outputs, requiring the complete canvas schema to be generated for each modification. This allows the tool to dynamically edit any attribute of the given element in the image, though at the cost of significant computational overhead.

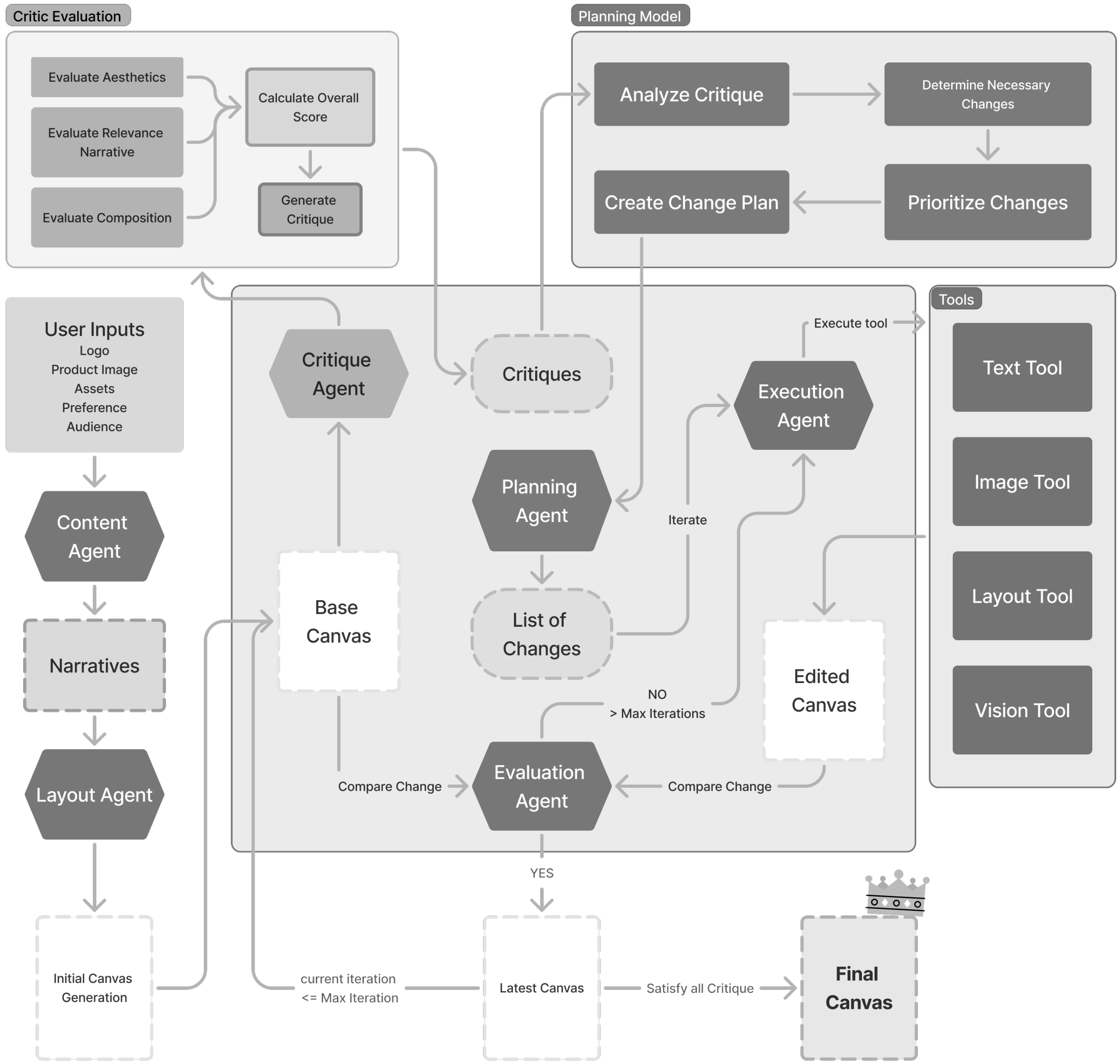

Figure 4: Overall Workflow: This diagram illustrates the complete design refinement process, showing how different agents collaborate in the system. The workflow demonstrates the iterative nature of our approach, where each stage builds upon the previous one to progressively improve the design output. The interconnected components highlight the system's ability to maintain coherence while allowing for both automated refinements and human interventions at key points in the process.

Approach 2 We noticed that in Approach 1, the changes generated by the Critic Agent were local to each issue identified, disregarding the overall outlook of the image. This leads to subsequent changes accidentally overriding previous changes applied. Hence in Approach 2, the Critic Agent focuses on solely generating a comprehensive report on visual flaws and semantic flaws. The Planning Agent is responsible for devising the priority and order in which changes should be executed based on the report created by the Critic Agent.

Another issue encountered in Approach 1 was how the system determines if the tool has sufficiently modified the right attribute to apply the change correctly. For instance, consider 'Increase the size of the title' to be the change that must be applied. The Text Tool observes that the font size of the title is 64 and therefore modifies it to 128. Now, we observe that the title is too large since the offset value chosen by the Text Tool was too large. Hence in Approach 2 we

Approach No	Critic Agent	Planning Agent	Execution Agent	Evaluation Agent	Tools
1.	**In:** Image URL **Out:** List of critics and suggestions	-	**In:** Suggestion and JSON image **Out:** Tool picked and index of object	-	**In:** Suggestion and JSON object **Out:** Updated JSON Object
2.	**In:** Image URL and narrative **Out:** Report on visual flaws	**In:** Critic Report **Out:** List of changes to be made	**In:** Change to be made, JSON image and Schema of Tools available **Out:** Tool picked, index of object	**In:** Image URL of old, new image and change **Out:** "YES" if the change is done correctly else returns feedback	**In:** Suggestion and JSON object **Out:** Updated JSON Object
3.	**In:** Image URL and narrative **Out:** Report on visual flaws	**In:** Critic Report and JSON image **Out:** List of changes and index of object	**In:** Change to be made and Schema of Tools available **Out:** Tool picked	**In:** Image URL of old, new image and change **Out:** "YES" if the change is done correctly else returns feedback	**In:** Suggestion, JSON object and Schema of all functions that the tool can execute **Out:** Function picked

Figure 5: Refinement Loop approaches

introduce an inner 'eval and apply' loop where the Evaluation Agent compares the images before and after the tool applies the change. If it does not feel the change applied correctly, it provides feedback to the Tool and the Tool reapplies the change. This process repeats until the Evaluation Agent is satisfied or maximum number of iterations is reached. Therefore, in the previous example, the Evaluation Agent will tell the Tool if the font size is too large or small and the Tool will modify the font size offset accordingly until it reaches the perfect value. This process simulates how humans correct minor errors in creative processes. Similar to Approach 1, this approach still relied on structured outputs for canvas manipulation, requiring the generation of complete canvas states for each modification.

Approach 3 To improve the speed of the refinement loop, we make changes to the Planning Agent, Execution Agent and Tools. We move the responsibility of inferring which element of the image that must be changed from Execution Agent to Planning Agent. Now, the Execution Agent will only be responsible for picking the right tool to apply the change. This removes the need for the image attributes to be given to the Execution Agent, reducing the number of tokens consumed and improving the speed of the Evaluation Agent. A significant improvement in this approach is the transition from structured outputs to pure function calling for canvas manipulation. Instead of generating the entire canvas schema for each modification, the Tools now leverage well-defined functions with specific parameters, dramatically reducing the token overhead and improving latency. This optimization allows for much faster iterations while maintaining precise control over canvas elements.

Results and Observations

In this study, we investigated the use of a novel agentic system to generate and refine visual content such as posters based on text and visual cues. The system addresses the limitations of foundational image generation models by providing iterative refinement and post-editing capabilities. We tested the model's capability with the following inputs:Brand Name: Mango Masti, Product Logo, Product Image: A glass of Mango Smoothie, advertising theme, layout, audience, style, product description, etc.

Figure 6: Brand Logo

Based on the input data, the system created four distinct narratives, each with its own approach to representing the brand and product. One of these narratives was chosen as the best fit to capture Mango Masti's brand identity and product. The image and vision engine then analyzed the given input assets, resulting in the initial version of the advertisement poster Figure 6.

In the initial iteration, it was observed that the title lacked focus due to its font style, which diminished its ability to capture attention. Furthermore, the logo overlapped with the

Figure 7: Product Image

Figure 8: Initial version of the advertisement poster

title, creating a cluttered layout. To address these issues, adjustments were made, resulting in a second version of the poster. While the color of the title was improved in this iteration, the overlap between the logo and the title persisted, requiring further refinement.

After multiple refinements, in the final iteration Figure 7, the system successfully resolved both the color and layout issues. The title's color was adjusted to make it more visually appealing, and the logo was properly positioned, resulting in a well-balanced, visually coherent, and professional advertisement. The final poster had a harmonious alignment of text and logo, as well as increased visual contrast, resulting in a more engaging design.

The refinement loop makes many iterations of changes based on the improvements required and continues until no further adjustments are required. This iterative method guarantees that the design is optimized, with all changes addressed systematically.

The following key observations were made:

1. Generation Speed:The entire poster generation process, including numerous revisions, took only a few minutes. This is a improvement over traditional design workflows, in which human designers generally spend days or weeks to finalise on a result. The system's speed enables rapid iteration, which is especially useful in fast-paced industries like as advertising.
2. Quality of Text and Image Alignment: The refinement loop allowed the system to maintain consistency in the textual narrative and graphic elements throughout multiple iterations. The system successfully addressed layout errors and ensured that the visual material matched the generated narrative.
3. Brand Consistency: Throughout the iterative refinement process, the system maintained the brand's integrity. The approach preserved the brand's visual identity by including the logo and product image consistently between iterations, ensuring that the final advertising remained solid with the brand's identity.
4. Narrative Coherence: One of our system's distinct characteristics was its ability to incorporate narrative coherence across the poster. By creating a story that connected the product to its target audience, it ensured that the final image transmitted the correct message in a visually appealing manner. This is a big improvement over typical AI image production, which frequently results in uneven or disconnected throughout multiple iterations.

Discussion and Limitations

Our multi-agent system demonstrates significant potential in automating and enhancing visual content creation, particularly for advertising materials. The system's ability to generate, refine, and iterate on visual content while maintaining brand consistency represents a substantial advancement in automated design systems. The integration of human oversight at each stage ensures that the final output aligns with creative intent while significantly reducing the time and effort required in traditional design workflows.

Key Strengths

The system's primary strength lies in its iterative refinement approach, which mirrors human design processes. By decomposing the creative process into discrete steps handled by specialized agents, we achieve a level of control and precision that surpasses traditional end-to-end generation approaches. The real-time visualization of agent decision-making through SSE events provides unprecedented transparency into the AI's creative process, enabling effective human-AI collaboration.

Our architecture's efficiency is particularly noteworthy, with all models running on a single T4 GPU node while maintaining interactive response times. This optimization makes the system practically deployable in real-world scenarios where computational resources may be limited. The separation of concerns between the lightweight control server and the GPU-intensive inference server enables efficient scaling as demand increases.

Limitations

Despite these achievements, several limitations warrant discussion:

Aesthetic Consistency: While the system can maintain brand guidelines, it occasionally struggles with subtle aesthetic choices that human designers handle intuitively. The evaluation agent sometimes fails to capture nuanced design principles that go beyond basic visual hierarchy rules.

Creative Boundaries: The system's reliance on predefined tools and modification patterns can limit its creative

Figure 9: Iterative Refinement of Advertisement Poster Design

flexibility. Currently, it cannot invent entirely new design approaches or break established rules in artistically meaningful ways, as human designers often do.

Performance Overhead: The iterative nature of our refinement process, while effective, introduces latency as each change requires multiple agent interactions and evaluations. This can result in longer processing times compared to single-pass generation approaches.

Tool Limitations: The current implementation supports only three primary tools (text, image, and layout). This constrains the system's ability to perform more specialized design modifications that might require additional tools or more complex combinations of existing ones.

Conclusion and Future Work

This paper presents a novel multi-agent approach to visual content generation that bridges the gap between automated systems and human design workflows. Our architecture demonstrates that by breaking down the creative process into specialized agent roles and enabling iterative refinement, we can achieve higher quality and more controllable results than traditional end-to-end approaches.

The system's ability to maintain real-time interaction while performing complex AI operations on modest hardware shows that sophisticated AI-driven design tools can be practically deployed in production environments. The integration of human oversight at key decision points ensures that the system augments rather than replaces human creativity.

Future work will focus on several promising directions:

1. **Enhanced Tool Framework:** Developing a more extensible tool system that allows for dynamic addition of new modification capabilities without requiring architectural changes.
2. **Advanced Aesthetic Learning:** Incorporating more sophisticated aesthetic evaluation metrics through fine-tuned vision-language models trained on design principles and human feedback.
3. **Optimization Techniques:** Investigating methods to reduce the latency of the refinement loop, potentially through parallel processing of compatible changes or more efficient agent communication patterns.
4. **Creative Exploration:** Extending the system to support more experimental and innovative design approaches, possibly through controlled violation of standard design rules or the introduction of style-transfer capabilities.
5. **Multi-Modal Generation:** Expanding the system to handle additional content types such as videos, animations, and interactive elements, enabling more comprehensive marketing material generation.

Through these improvements, we aim to further narrow the gap between AI-generated and human-created content while maintaining the efficiency and scalability advantages of automated systems. Our work demonstrates the potential of multi-agent architectures in creative domains and provides a foundation for future research in AI-assisted design tools.

References

AI, S. 2024. Image Inpainting Using Deep Learning: Advanced Techniques, Architectures, and Applications.

Bai, J.; Bai, S.; Chu, Y.; Cui, Z.; Dang, K.; Deng, X.; Fan, Y.; Ge, W.; Han, Y.; Huang, F.; Hui, B.; Ji, L.; Li, M.; Lin, J.; Lin, R.; Liu, D.; Liu, G.; Lu, C.; Lu, K.; Ma, J.; Men, R.; Ren, X.; Ren, X.; Tan, C.; Tan, S.; Tu, J.; Wang, P.; Wang, S.; Wang, W.; Wu, S.; Xu, B.; Xu, J.; Yang, A.; Yang, H.; Yang, J.; Yang, S.; Yao, Y.; Yu, B.; Yuan, H.; Yuan, Z.; Zhang, J.; Zhang, X.; Zhang, Y.; Zhang, Z.; Zhou, C.; Zhou, J.; Zhou, X.; and Zhu, T. 2023. Qwen Technical Report. arXiv:2309.16609.

Balaji, Y.; Nah, S.; Huang, X.; Vahdat, A.; Song, J.; et al. 2023. T2I-Adapter: A Simple and Small Model for Extra Guidance in Text-to-Image Models. *ArXiv*, abs/2302.08453.

BeFrend. 2023. IP-Adapter: Image Prompt Adapter for Text-to-Image Diffusion Models. https://huggingface.co/BeFrend/IP-Adapter.

Beyer, L.; Steiner, A.; Pinto, A. S.; Kolesnikov, A.; Wang, X.; Salz, D.; Neumann, M.; Alabdulmohsin, I.; Tschannen, M.; Bugliarello, E.; Unterthiner, T.; Keysers, D.; Koppula, S.; Liu, F.; Grycner, A.; Gritsenko, A.; Houlsby, N.; Kumar, M.; Rong, K.; Eisenschlos, J.; Kabra, R.; Bauer, M.; Bošnjak, M.; Chen, X.; Minderer, M.; Voigtlaender, P.; Bica, I.; Balazevic, I.; Puigcerver, J.; Papalampidi, P.; Henaff, O.; Xiong, X.; Soricut, R.; Harmsen, J.; and Zhai, X. 2024. PaliGemma: A versatile 3B VLM for transfer. arXiv:2407.07726.

Chai, S.; Zhuang, L.; and Yan, F. 2023. LayoutDM: Transformer-based Diffusion Model for Layout Generation. arXiv:2305.02567.

Chase, H. 2022. LangChain. If you use this software, please cite it as below.

Cheng, T.; Song, L.; Ge, Y.; Liu, W.; Wang, X.; and Shan, Y. 2024. YOLO-World: Real-Time Open-Vocabulary Object Detection. arXiv:2401.17270.

Jiang, A. Q.; Sablayrolles, A.; Mensch, A.; Bamford, C.; Chaplot, D. S.; de las Casas, D.; Bressand, F.; Lengyel, G.; Lample, G.; Saulnier, L.; Lavaud, L. R.; Lachaux, M.-A.; Stock, P.; Scao, T. L.; Lavril, T.; Wang, T.; Lacroix, T.; and Sayed, W. E. 2023. Mistral 7B. arXiv:2310.06825.

Jiang, D.; He, X.; Zeng, H.; Wei, C.; Ku, M.; Liu, Q.; and Chen, W. 2024. MANTIS: Interleaved Multi-Image Instruction Tuning. arXiv:2405.01483.

Kirillov, A.; Mintun, E.; Ravi, N.; Mao, H.; Rolland, C.; Gustafson, L.; Xiao, T.; Whitehead, S.; Berg, A. C.; Lo, W.-Y.; Dollár, P.; and Girshick, R. 2023. Segment Anything. arXiv:2304.02643.

Labs, B. F. 2024. FLUX.1 [dev]: A 12 Billion Parameter Rectified Flow Transformer. https://huggingface.co/black-forest-labs/FLUX.1-dev. Accessed: 2024-11-29.

Laurençon, H.; Tronchon, L.; Cord, M.; and Sanh, V. 2024. What matters when building vision-language models? arXiv:2405.02246.

Li, B.; Qi, X.; Lukasiewicz, T.; and Torr, P. H. S. 2019. Controllable Text-to-Image Generation. arXiv:1909.07083.

Liu, H.; Li, C.; Wu, Q.; and Lee, Y. J. 2023. Visual Instruction Tuning. arXiv:2304.08485.

Podell, D.; English, Z.; Lacey, K.; Blattmann, A.; Dockhorn, T.; Müller, J.; Penna, J.; and Rombach, R. 2023. SDXL: Improving Latent Diffusion Models for High-Resolution Image Synthesis. arXiv:2307.01952.

Ravi, N.; Gabeur, V.; Hu, Y.-T.; Hu, R.; Ryali, C.; Ma, T.; Khedr, H.; Rädle, R.; Rolland, C.; Gustafson, L.; Mintun, E.; Pan, J.; Alwala, K. V.; Carion, N.; Wu, C.-Y.; Girshick, R.; Dollár, P.; and Feichtenhofer, C. 2024. SAM 2: Segment Anything in Images and Videos. arXiv:2408.00714.

Rombach, R.; Blattmann, A.; Lorenz, D.; Esser, P.; and Ommer, B. 2022. High-Resolution Image Synthesis with Latent Diffusion Models. arXiv:2112.10752.

Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; Bikel, D.; Blecher, L.; Ferrer, C. C.; Chen, M.; Cucurull, G.; Esiobu, D.; Fernandes, J.; Fu, J.; Fu, W.; Fuller, B.; Gao, C.; Goswami, V.; Goyal, N.; Hartshorn, A.; Hosseini, S.; Hou, R.; Inan, H.; Kardas, M.; Kerkez, V.; Khabsa, M.; Kloumann, I.; Korenev, A.; Koura, P. S.; Lachaux, M.-A.; Lavril, T.; Lee, J.; Liskovich, D.; Lu, Y.; Mao, Y.; Martinet, X.; Mihaylov, T.; Mishra, P.; Molybog, I.; Nie, Y.; Poulton, A.; Reizenstein, J.; Rungta, R.; Saladi, K.; Schelten, A.; Silva, R.; Smith, E. M.; Subramanian, R.; Tan, X. E.; Tang, B.; Taylor, R.; Williams, A.; Kuan, J. X.; Xu, P.; Yan, Z.; Zarov, I.; Zhang, Y.; Fan, A.; Kambadur, M.; Narang, S.; Rodriguez, A.; Stojnic, R.; Edunov, S.; and Scialom, T. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. arXiv:2307.09288.

Wang, W.; Lv, Q.; Yu, W.; Hong, W.; Qi, J.; Wang, Y.; Ji, J.; Yang, Z.; Zhao, L.; Song, X.; Xu, J.; Xu, B.; Li, J.; Dong, Y.; Ding, M.; and Tang, J. 2024a. CogVLM: Visual Expert for Pretrained Language Models. arXiv:2311.03079.

Wang, Z.; Li, A.; Li, Z.; and Liu, X. 2024b. GenArtist: Multimodal LLM as an Agent for Unified Image Generation and Editing. arXiv:2407.05600.

Wu, Q.; Bansal, G.; Zhang, J.; Wu, Y.; Li, B.; Zhu, E.; Jiang, L.; Zhang, X.; Zhang, S.; Liu, J.; Awadallah, A. H.; White, R. W.; Burger, D.; and Wang, C. 2023. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation. arXiv:2308.08155.

Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K.; and Cao, Y. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. arXiv:2210.03629.

Yu, N.; Chen, C.-C.; Chen, Z.; Meng, R.; Wu, G.; Josel, P.; Niebles, J. C.; Xiong, C.; and Xu, R. 2024. LayoutDETR: Detection Transformer Is a Good Multimodal Layout Designer. arXiv:2212.09877.

Yuan, L.; Chen, D.; Chen, Y.-L.; Codella, N.; Dai, X.; Gao, J.; Hu, H.; Huang, X.; Li, B.; Li, C.; Liu, C.; Liu, M.; Liu, Z.; Lu, Y.; Shi, Y.; Wang, L.; Wang, J.; Xiao, B.; Xiao, Z.; Yang, J.; Zeng, M.; Zhou, L.; and Zhang, P. 2021. Florence: A New Foundation Model for Computer Vision. arXiv:2111.11432.

Zhang, C.; Han, D.; Zheng, S.; Choi, J.; Kim, T.-H.; and Hong, C. S. 2023. MobileSAMv2: Faster Segment Anything to Everything. arXiv:2312.09579.

Zhang, L.; Rao, A.; and Agrawala, M. 2023. Adding Conditional Control to Text-to-Image Diffusion Models. arXiv:2302.05543.

Zhu, D.; Chen, J.; Shen, X.; Li, X.; and Elhoseiny, M. 2023. MiniGPT-4: Enhancing Vision-Language Understanding with Advanced Large Language Models. arXiv:2304.10592.

Zhu, W.; Healey, J.; Zhang, R.; Wang, W. Y.; and Sun, T. 2024. Automatic Layout Planning for Visually-Rich Documents with Instruction-Following Models. arXiv:2404.15271.

APPENDIX

Implementation Details

Model Specifications

Our system architecture integrates multiple cutting-edge artificial intelligence models, each carefully selected and optimized for specific tasks within the design workflow. The implementation leverages the following state-of-the-art models:

- **Language Models:** We utilize LLaMA 2 70B as our primary language model, chosen for its exceptional performance in content generation and planning tasks. This model demonstrates superior capabilities in understanding design context, generating appropriate content suggestions, and planning design modifications. Its architecture allows for efficient processing of complex design briefs while maintaining contextual awareness throughout the design process.
- **Vision-Language Models:** The system incorporates Pixtral , GPT-4o-mini , etc for sophisticated visual analysis and evaluation tasks. These models excels in understanding visual elements and their relationships, providing detailed feedback on design compositions, and evaluating aesthetic qualities. Its multimodal capabilities enable seamless integration of visual and textual information in the design workflow.
- **Image Generation:** Stable Diffusion XL serves as our core image generation model, selected for its outstanding ability to create high-quality, contextually appropriate visual elements. The model demonstrates exceptional performance in generating backgrounds, textures, and visual assets that align with specific design requirements.
- **Object Detection:** We implement a dual-model approach using SAM (Segment Anything Model) and Florence for precise object segmentation and detection. This combination provides robust capabilities in identifying and manipulating individual design elements while maintaining spatial relationships and visual hierarchy.

Hardware Requirements

Our system has been engineered to deliver optimal performance while maintaining resource efficiency. The infrastructure requirements have been carefully balanced to ensure accessibility without compromising functionality:

- **GPU Server Configuration:** The system operates effectively on a single NVIDIA T4 GPU with 16GB VRAM. This configuration provides sufficient computational power for real-time model inference while maintaining cost-effectiveness. The T4's architecture is particularly well-suited for running multiple AI models simultaneously, enabling smooth interaction between different system components.
- **Control Server Specifications:** The control layer runs on a modest yet efficient setup of 4 vCPUs with 16GB RAM. This configuration has been optimized to handle concurrent user sessions, manage model coordination, and process design operations with minimal latency. The server architecture includes load balancing capabilities to ensure consistent performance during peak usage.
- **Database Infrastructure:** We employ a serverless PostgreSQL instance through Neon, chosen for its combination of scalability and minimal maintenance requirements. This configuration provides robust data persistence while automatically handling scaling and backup operations. The serverless architecture ensures optimal resource utilization and cost efficiency.
- **Storage Solution:** AWS S3 serves as our primary storage solution for image assets, selected for its high availability, durability, and seamless integration capabilities. The system implements intelligent caching mechanisms to optimize asset delivery and reduce latency in accessing frequently used design elements.

Interactive Editor Interface

Primary Dashboard Interface

The primary dashboard functions as the central command center for all design operations, offering an intuitive and comprehensive overview of the available tools and resources. The interface is strategically organized to minimize cognitive load while maximizing productivity, featuring distinct zones for different functionalities. The workspace presents a clean, organized layout that allows designers to focus on their creative process while maintaining easy access to essential tools and features.

Content Engine Interface

The content engine interface represents a sophisticated approach to AI-assisted content generation and management, serving as a bridge between human creativity and artificial intelligence. This interface seamlessly integrates with the main dashboard while providing specialized tools for content creation and refinement. The system incorporates advanced parameter controls that allow for fine-tuned content generation while maintaining brand consistency and design principles.

AI Edit Mode Interface

The AI Edit Mode interface introduces a revolutionary approach to design refinement by establishing a clear distinction between human-driven and AI-assisted editing sessions. Through a dedicated toggle system, users can explicitly switch between human editing and AI-assisted editing

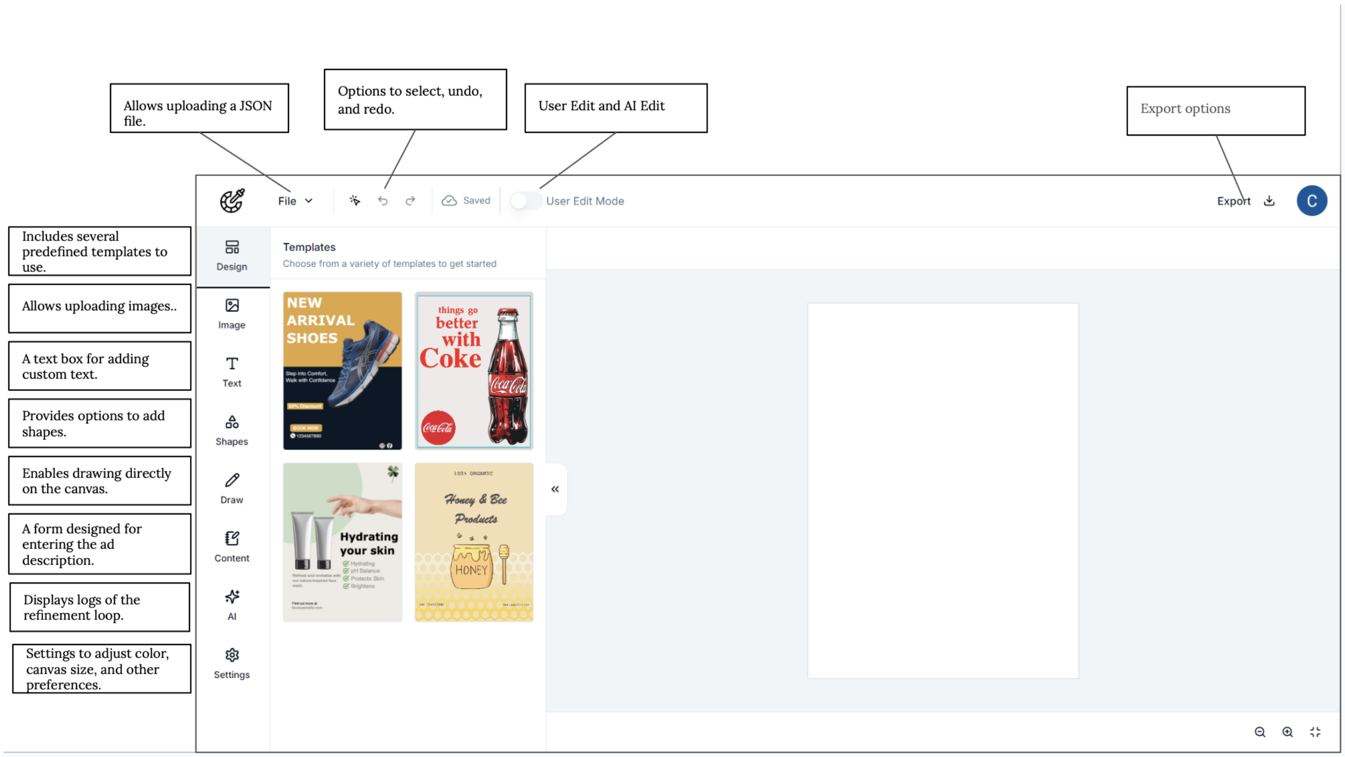

Figure 10: Main Dashboard View 1: The primary interface showcases an integrated workspace environment with well-organized tool panels, layer management systems, and intuitive navigation controls. The layout emphasizes efficiency in workflow management while maintaining quick access to frequently used tools

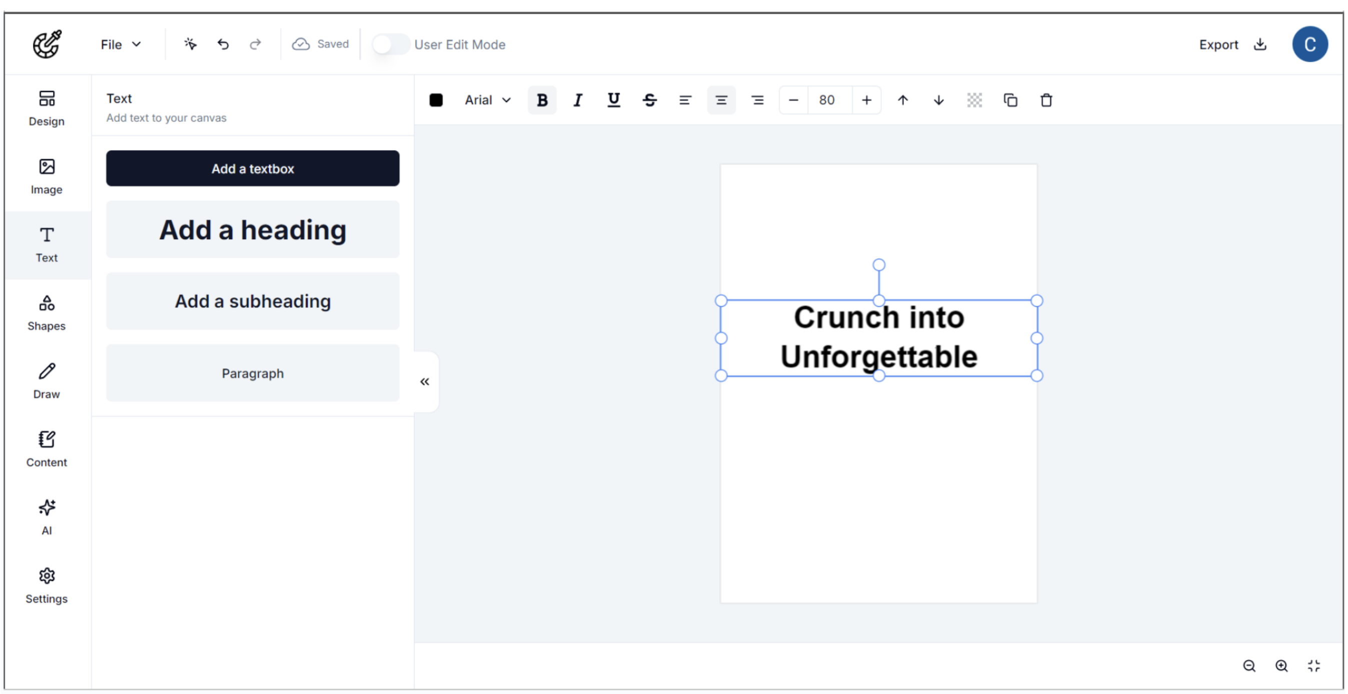

Figure 11: Main Dashboard View 2: Advanced features include sophisticated layer controls, smart grouping tools, and automated alignment systems. The interface demonstrates how complex operations are simplified through intelligent workspace organization

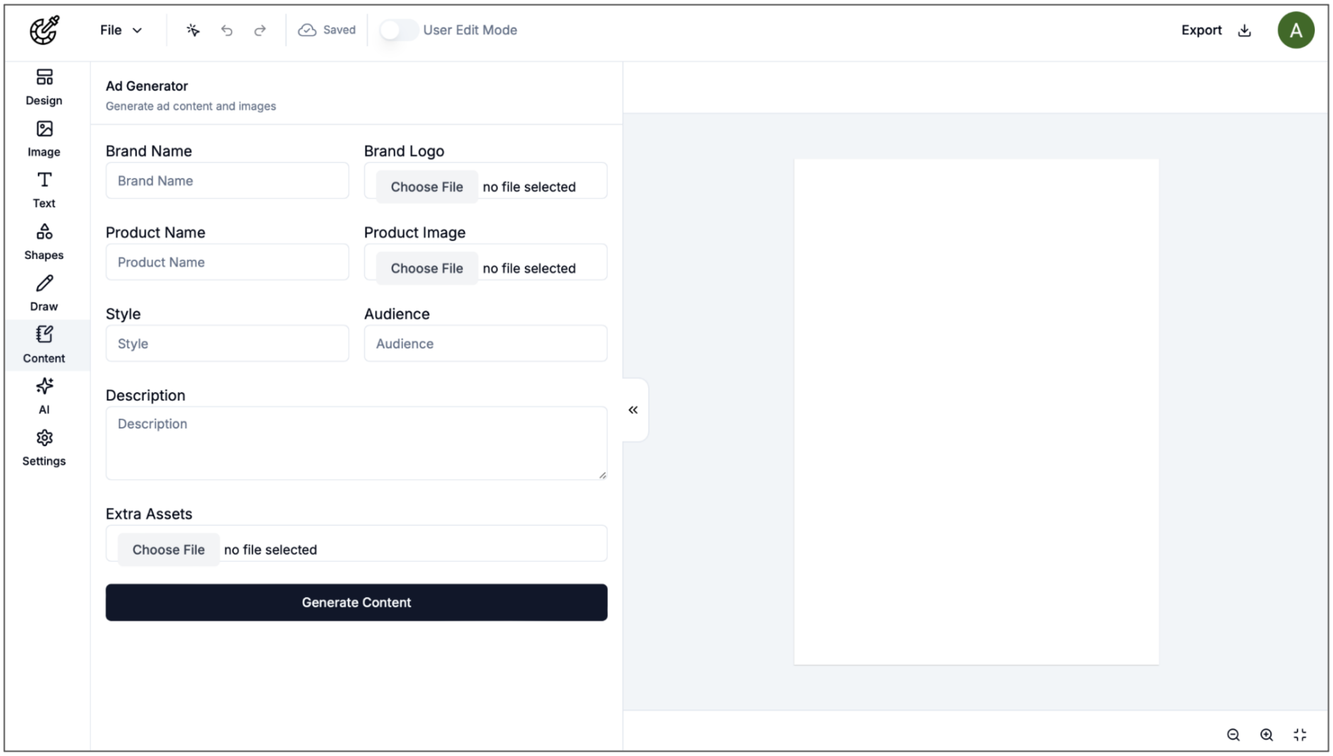

Figure 12: Content Engine Dashboard: The input interface presents sophisticated content generation controls and parameters, allowing users to specify detailed requirements for automated content creation. The system includes brand guideline integration and style preference settings

modes, ensuring transparency in the design process. When AI Edit Mode is activated, the system provides real-time visualization of AI-driven improvements and modifications while maintaining a complete history of changes for review and potential rollback.

Refinement Loop Visualization

The refinement loop interface provides detailed insights into the iterative improvement process, offering a transparent view of the collaboration between various AI agents and human input. This visualization system tracks and displays the progression of design changes, showing how different elements evolve through successive iterations. The interface includes detailed metrics and progress indicators that help users understand the impact of each refinement cycle.

System Integration and Workflow

Component Interaction

The system implements a sophisticated interaction model between various components, ensuring smooth data flow and real-time updates across all interfaces. Each component maintains state awareness while operating independently, allowing for robust error handling and graceful degradation when necessary.

Performance Optimization

The interface components are optimized for performance through:

- Intelligent component loading with dynamic resource allocation
- Efficient state management using advanced caching mechanisms
- Optimized render cycles for complex visualizations
- Streamlined data flow between interface components

Future Developments

The interface system is designed for continuous evolution, with planned enhancements including:

- Advanced collaborative features for team-based design workflows
- Enhanced visualization tools for agent decision-making processes
- Expanded analytics capabilities for design performance metrics
- Integration of emerging AI technologies for improved design assistance

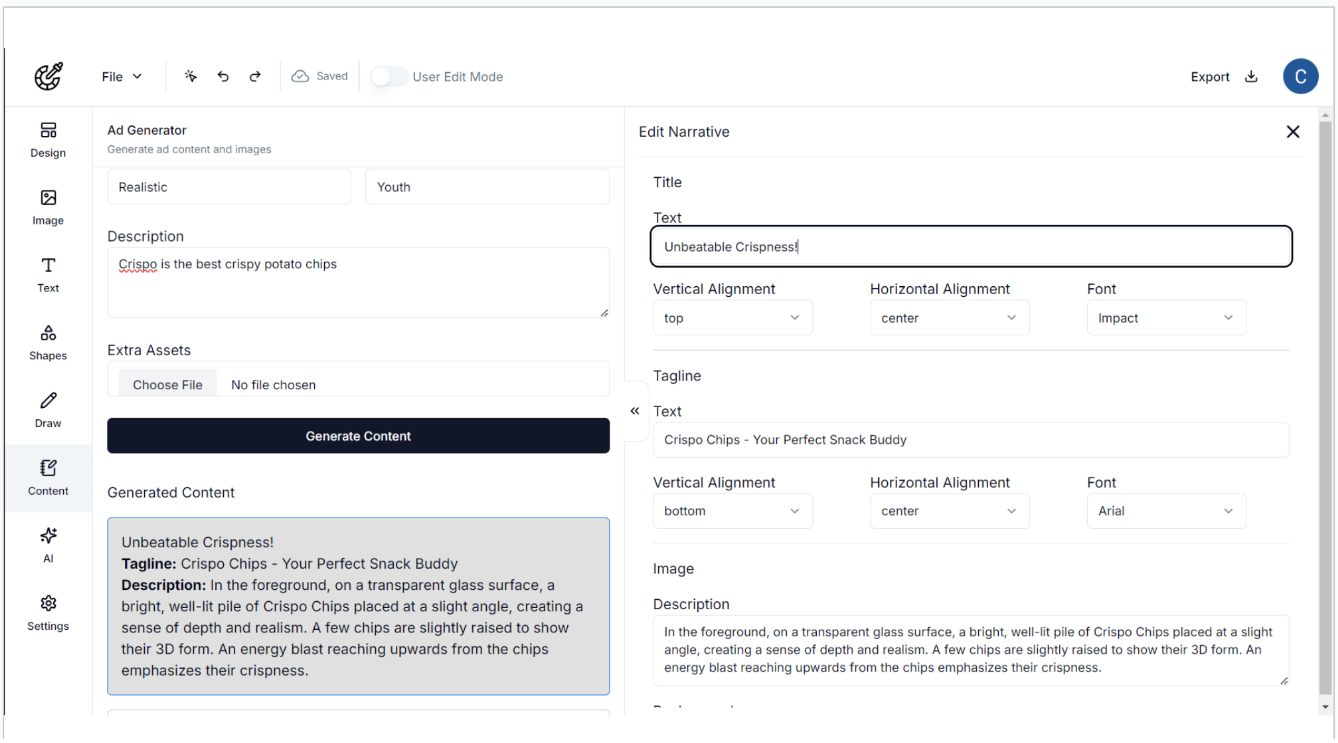

Figure 13: Content Engine Output View: The results interface displays generated content alongside comprehensive analytics and refinement options. Users can evaluate generated content through various metrics and access tools for immediate modifications

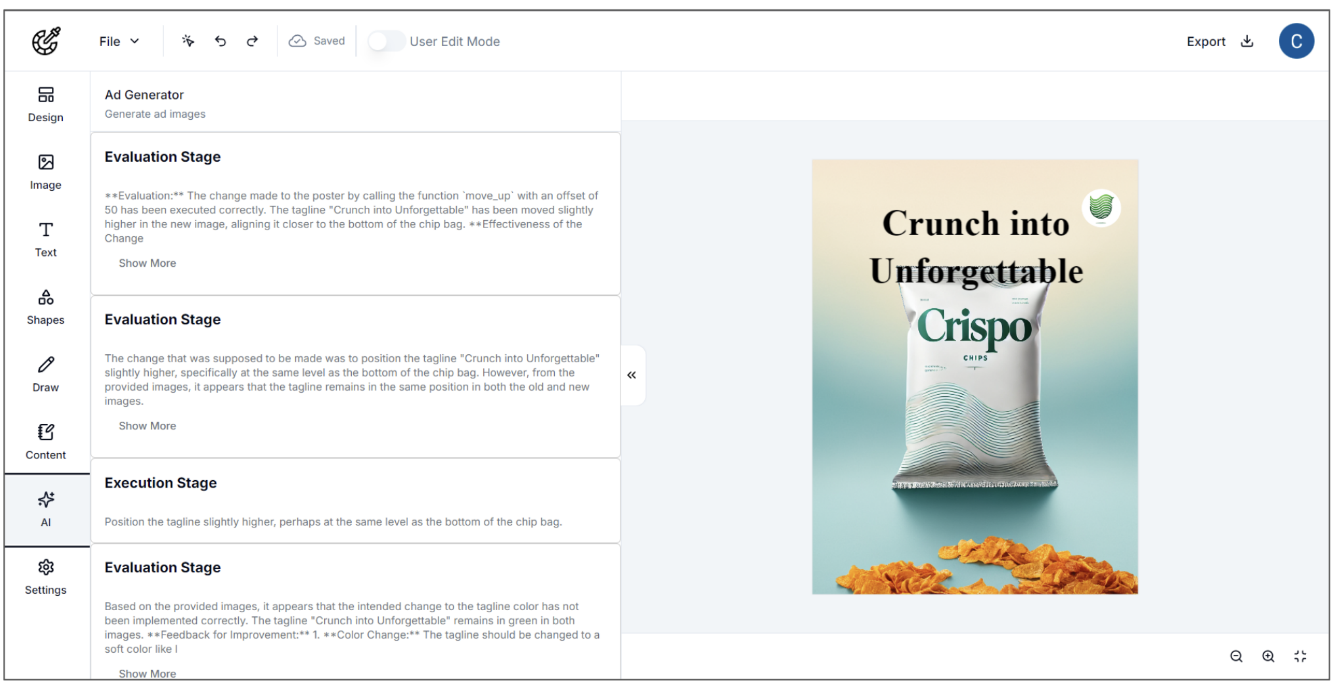

Figure 14: AI Edit Mode Interface: The comprehensive view shows the intelligent design modification system in action, featuring clear indicators of AI engagement, intervention points, and automated suggestion implementations. The interface maintains transparency by explicitly highlighting AI-driven changes versus human modifications

Figure 15: Refinement Loop Interface: A comprehensive visual representation of the iterative design improvement process, showcasing the interplay between AI agents and human decisions. The interface tracks modification history, decision points, and improvement metrics across multiple refinement cycles

Tool Documentation

Text Manipulation Tools

Text manipulation tools provide precise control over typographic elements and text styling within the design environment. Each tool is specifically engineered to handle distinct aspects of text modification while maintaining visual coherence.

Text Size Tool **Function:** changeTextSize
Input Parameters:

- textObject: The target text element
- size: Integer value for font size

Description: This tool provides precise control over text dimensions, allowing for dynamic size adjustments while maintaining proportional relationships within the design. It automatically handles size scaling while preserving text clarity and legibility across different display contexts.

Figure 16: Text size adjustment demonstration

Color Modification Tool **Function:** changeColor
Input Parameters:

- textObject: The target text element
- color: String representation of color value

Description: The Color Modification Tool enables seamless color adjustments for text elements. It supports various color formats including hex codes and named colors, ensuring consistent color application across the design system.

Figure 17: Color modification showcasing different text color applications

Shadow Effect Tool **Function:** addShadow
Input Parameters:

- textObject: The target text element
- color: Shadow color value
- blur: Integer value for shadow blur
- offsetX: Horizontal offset
- offsetY: Vertical offset

Description: This tool applies customizable shadow effects to text elements, enhancing depth and visual hierarchy. It includes intelligent contrast checking to ensure shadow effects enhance rather than diminish text visibility.

Figure 18: Shadow effect application showing depth and dimensionality

Stroke Effect Tool **Function:** addStroke
Input Parameters:

- textObject: The target text element
- color: Stroke color value
- width: Stroke width in pixels

Description: The Stroke Effect Tool adds outline effects to text elements, allowing for enhanced visibility and visual impact. It maintains text legibility while providing flexible stroke customization options.

Figure 19: Stroke effect demonstration on text elements

Text Alignment Tool **Function:** changeAlignment
Input Parameters:

- textObject: The target text element
- alignment: String ('left', 'center', 'right')

Description: This tool manages text alignment with precision, supporting various alignment options while maintaining consistent spacing and layout integrity across different text blocks.

Figure 20: Text alignment options demonstration

Text Content Tool **Function:** changeTextContent
Input Parameters:

- textObject: The target text element
- text: New text content string

Description: The Text Content Tool enables dynamic text content updates while preserving all styling and formatting attributes. It handles text replacement operations with automatic adjustment of container dimensions.

Font Style Tool **Function:** changeTextStyle
Input Parameters:

- textObject: The target text element
- style: Font family name
- fontWeight: Integer weight value

Description: This comprehensive typography tool manages font family and weight adjustments, ensuring consistent rendering across different platforms while maintaining design integrity.

Figure 21: Text content modification example

Figure 22: Font style modifications showing different typographic treatments

Text Decoration Tool **Function:** applyLine
Input Parameters:

- textObject: The target text element
- underline: Boolean for underline effect
- overline: Boolean for overline effect
- linethrough: Boolean for strikethrough effect

Description: The Text Decoration Tool applies various line decorations to text elements, supporting multiple decoration types simultaneously while maintaining visual harmony.

Figure 23: Text decoration effects demonstration

Layout Manipulation Tools

Layout tools provide precise control over element positioning and arrangement within the design canvas. These tools ensure consistent spacing and alignment across different design elements.

Vertical Movement Tool (Down) **Function:** move_down
Input Parameters:

- object: The target element
- offset: Integer value for vertical movement

Description: This tool enables precise downward movement of design elements, maintaining relative positioning with surrounding elements while ensuring smooth transitions.

Figure 24: Downward movement demonstration

Vertical Movement Tool (Up) **Function:** move_up
Input Parameters:

- object: The target element
- offset: Integer value for vertical movement

Description: The upward movement tool provides controlled vertical positioning, supporting precise element placement while maintaining design hierarchy.

Figure 25: Upward movement demonstration

Horizontal Movement Tool (Right) **Function:** move_right
Input Parameters:

- object: The target element
- offset: Integer value for horizontal movement

Description: This tool manages rightward movement of elements with precision, maintaining proper spacing and alignment with adjacent elements.

Figure 26: Rightward movement demonstration

Horizontal Movement Tool (Left) **Function:** move_left
Input Parameters:

- object: The target element
- offset: Integer value for horizontal movement

Description: The leftward movement tool enables precise horizontal positioning, supporting fluid element arrangement while maintaining design balance.

Figure 27: Leftward movement demonstration

Top Position Tool **Function:** position_top
Input Parameters:

- object: The target element

Description: This tool automatically positions elements at the top of the canvas while maintaining proper alignment and spacing with other elements.

Bottom Position Tool **Function:** position_down
Input Parameters:

- object: The target element

Description: The bottom positioning tool places elements at the bottom of the canvas while ensuring proper spacing and alignment with existing elements.

Left Position Tool **Function:** position_left
Input Parameters:

- object: The target element

Description: This tool manages left-edge positioning of elements, handling both text and image elements with appropriate origin point adjustments.

Figure 28: Top positioning demonstration

Figure 29: Bottom positioning demonstration

Figure 30: Left positioning demonstration

Right Position Tool **Function:** position_right
Input Parameters:

- object: The target element

Description: The right positioning tool handles element placement at the canvas's right edge, automatically adjusting origin points based on element type.

Horizontal Center Tool **Function:** position_x_to_center
Input Parameters:

- object: The target element

Description: This tool provides precise horizontal centering of elements, handling both text and image elements with appropriate origin point and alignment adjustments.

Figure 31: Right positioning demonstration

Figure 32: Horizontal centering demonstration

Vertical Center Tool **Function:** position_y_to_center
Input Parameters:

- object: The target element

Description: The vertical centering tool ensures precise middle alignment of elements along the vertical axis while maintaining proper spacing relationships.

Figure 33: Vertical centering demonstration

Image Manipulation Tools

These tools provide specialized functionality for handling image assets within the design environment.

Image Clipping Tool **Function:** clip_asset
Input Parameters:

- image: The target image element
- clipPath: Path coordinates for clipping

Description: This tool enables precise image cropping and masking operations, supporting various clipping paths while maintaining image quality and performance.

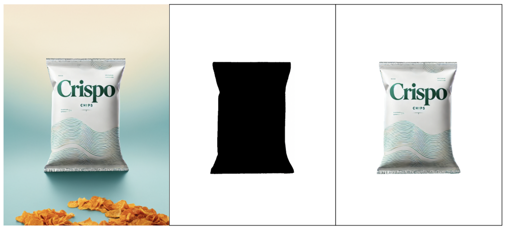

Figure 34: Image clipping demonstration

Image Movement Tool **Function:** move_asset
Input Parameters:

- image: The target image element
- position: New coordinates for image placement

Description: The image movement tool provides precise control over image positioning while maintaining aspect ratios and relative scaling.

Figure 35: Image movement demonstration

Background Modification Tool **Function:** change_background
Input Parameters:

- image: The target image element
- background: New background properties

Description: This tool manages background modifications for image elements, supporting various background types and ensuring proper rendering across different contexts. **Suggestion:** Change the background to a beach scene.

Extended Results

Refinement Loop Examples

Case Study 1: Potato Chip Design This case study demonstrates our system's ability to handle complex text-heavy designs while maintaining visual hierarchy and brand consistency. Figure 37 shows the initial generated design, which underwent several refinement iterations to improve readability, spacing, and visual appeal.

Initial Input:

Figure 36: Image movement demonstration

- Brand: Crispo
- Style: Modern, minimalist

Refinement Process:

Stage 1: Initial Image Generation As shown in Figure 37, the system generated an initial design incorporating the brand elements and key messaging.

Figure 37: Initial generated design for Crispo chips packaging, showing base layout and typography

Stage 2: Typography Refinement
Critic Agent:

1. Can we improve the title 'Your Healthy Crunch' font style to match the theme of the poster better? Yes, the current sans-serif font appears too basic for a snack product. A more modern, rounded font would better match the wave elements and friendly nature of the product. Consider using fonts like "Quicksand" or "Comfortaa" that have softer edges while maintaining readability.

2. Is the title 'Your Healthy Crunch' size too large or too small? The title size is appropriate as it creates a clear hierarchy, but it could be reduced by about 10-15

3. Would adding some extra effects like a stroke or drop shadow to the title 'Your Healthy Crunch' improve the image? Yes, adding a subtle drop shadow would help separate the text from the light green background and potato chip elements. A very light shadow with 20-30

4. Can we improve the color of the title 'Your Healthy Crunch' to stand out with the theme? The black color could be replaced with the same teal green used in the Crispo logo to create better brand consistency and a more cohesive color scheme while maintaining readability against the light background.

5. Should the title's position be changed to improve the poster? The current position works well but could be moved slightly higher to create more breathing room between the title and the product package, allowing for better visual flow.

6. Can we improve the tagline 'The Snack You'll Love to Munch!' font style to match the theme of the poster better? Yes, the current casual script font looks somewhat amateur. A playful sans-serif font with rounded edges would maintain the friendly feel while looking more professional and matching the brand's modern aesthetic.

7. Is the tagline 'The Snack You'll Love to Munch!' size too large or too small? The tagline size is too small relative to the overall composition. Increasing it by approximately 25

8. Would adding some extra effects like a stroke or drop shadow to the tagline 'The Snack You'll Love to Munch!' improve the image? Similar to the title, a subtle drop shadow would help the tagline stand out from the background and chip elements, improving readability while maintaining the clean design.

9. Can we improve the color of the tagline 'The Snack You'll Love to Munch!' to stand out with the theme? Yes, changing the black to either the teal green from the logo or a slightly darker shade of it would create better harmony while ensuring readability.

10. Should the tagline's position be changed to improve the poster? The tagline could be moved slightly higher from the bottom edge to create more margin space. Additionally, centering it horizontally would create better alignment with the centered product package and title above.

Planning Agent

"changes": [

"index": 2, "change": "Change font family to 'Quicksand' for a more modern and rounded appearance that matches the theme" ,

"index": 2, "change": "Reduce font size from 100 to 85 to better balance with the product package" ,

"index": 2, "change": "Add drop shadow effect with 25,

"index": 2, "change": "Change fill color from black to teal green (matching Crispo logo) for brand consistency" ,

"index": 2, "change": "Move position slightly higher to create more space between title and product package" ,

"index": 3, "change": "Change font family from 'Comic Sans MS' to a playful rounded sans-serif font" ,

"index": 3, "change": "Increase font size from 55 to 69 for better readability and balance" ,

"index": 3, "change": "Add subtle drop shadow effect with 25,

"index": 3, "change": "Change fill color from black to teal green (matching brand color) for harmony" ,

"index": 3, "change": "Adjust position: move slightly higher from bottom edge and center horizontally"]

Change 1

Text Tool:

User Prompt: You are given a change to be made to a textbox and the properties of the textbox by the user. You have access to the given tools/functions. Properties of text:"type": "textbox", "version": "5.3.0", "originX": "center", "originY": "top", "left": 450, "top": 100, "width": 900, "height": 244.08, "fill": "rgba(0,0,0,1)", "stroke": null, "strokeWidth": 1, "strokeDashArray": null, "strokeLineCap": "butt", "strokeDashOffset": 0, "strokeLineJoin": "miter", "strokeUniform": false, "strokeMiterLimit": 4, "scaleX": 1, "scaleY": 1, "angle": 0, "flipX": false, "flipY": false, "opacity": 1, "shadow": null, "visible": true, "backgroundColor": "", "fillRule": "nonzero", "paintFirst": "fill", "globalCompositeOperation": "source-over", "skewX": 0, "skewY": 0, "fontFamily": "Helvetica", "fontWeight": 700, "fontSize": 75, "text": "Your Healthy Crunch", "underline": false, "overline": false, "linethrough": false, "textAlign": "center", "fontStyle": "normal", "lineHeight": 1.16, "textBackgroundColor": "", "charSpacing": 0, "styles": [], "direction": "ltr", "path": null, "pathStartOffset": 0, "pathSide": "left", "pathAlign": "baseline", "minWidth": 20, "splitByGrapheme": false, "selectable": true, "hasControls": true, "editable": true Change to be made: Add drop shadow effect with 25% opacity to create depth

Assistant: I'll help add a drop shadow to create depth. For 25% opacity, we'll use rgba color with 0.25 alpha value. The shadow should be subtle but noticeable, so I'll use moderate offset and blur values. addShadow("color": "rgba(0, 0, 0, 0.25)", "offsetX": 3, "offsetY": 3, "blur": 5)

Evaluation Agent:

User: You are a user interface evaluator. You are given an old image of a poster and a new image of poster after a change has been made by calling a function. You must eval-

Figure 38: Change 1: Adding a drop shadow

uate whether the change has been put into effect correctly and if the change improves the image. If it has been put into effect correctly and improves the image, output 'YES'. Otherwise, Provide feedback on how to make the change more effective . Change that should be made: Add drop shadow effect with 25 % opacity to create depth. Function called to make the change: "function": "addShadow", "arguments": "color": "rgba(0, 0, 0, 0.25)", "offsetX": 3, "offsetY": 3, "blur": 5

Assistant: YES. The drop shadow effect has been correctly applied to the chips package in the second image, creating a subtle but effective depth that makes the package appear to float above the green background. The shadow parameters (25 % opacity, 3px offset, and 5px blur) create a natural-looking shadow that enhances the overall design without being too dramatic or distracting. This change successfully improves the visual hierarchy and makes the product stand out more prominently against the scattered chip elements in the background. *Change 2*

Case Study 2: University Advertisement This case study showcases the system's ability to handle image-centric designs with strong brand messaging.

Initial Input:

- Product: ABC University
- Target Audience: Prospective students
- Key Message: "Focus on Your Future"

Refinement Process:
Stage 1: Initial Image Generation

In this step, a background is blended into the university building, and text components are added to create a preliminary poster.

Stage 2: Typography and Layout Refinement The Critic Agent identified issues with text hierarchy and readability.

Critic Agent:

1. Can we improve the title font style to match the theme of the poster better? The current bold sans-serif font feels heavy and lacks sophistication. A more modern, sleek font like Futura or Helvetica Neue would better reflect the contemporary architecture and forward-thinking message. Consider using a lighter weight font that maintains readability while appearing more elegant.

2. Is the title size too large or too small? The title size is appropriate for grabbing attention, but could be reduced by about 15-20

3. Would adding extra effects like stroke or drop shadow to the title improve the image? The current drop shadow is

Figure 39: All iterations

too heavy and dated-looking. Instead of removing effects completely, use a subtle, soft shadow (30-40

4. Can we improve the color of the title to stand out with the theme? The black color appears too stark against the bright sky. Consider using a deep navy blue or charcoal gray (like 2C3E50) to maintain contrast while better complementing the architectural elements and sky tones in the image.

5. Should the title's position be changed to improve the poster? The title could be moved slightly higher to create more space between it and the building, allowing the architecture to breathe. This would also create better visual hierarchy with the tagline below.

6. Can we improve the tagline font style to match the theme better? Yes, the tagline should use a complementary font to the header, perhaps the same font family but in a lighter weight. This would create better visual harmony while maintaining hierarchy.

7. Is the tagline size too large or too small? The tagline size is too large relative to the main title. Reducing it by about 25

8. Would adding extra effects to the tagline improve the image? The tagline has the same heavy shadow effect as the title. It should match whatever subtle effect is applied to the title, but at an even lighter opacity (around 25

9. Can we improve the color of the tagline to stand out with the theme? The tagline should be in the same color family as the title but slightly lighter to create depth. If the title uses deep navy, the tagline could use a slightly lighter blue-gray tone.

10. Should the tagline's position be changed to improve the poster? The tagline could be moved slightly higher from the bottom edge to prevent it from feeling cramped. Additionally, consider breaking it into two lines with better line spacing to improve readability and overall composition.

Planning Agent

”changes”: [

”index”: 2, ”change”: ”Update title font family to ’Futura’ and reduce font size to 60px” ,

”index”: 2, ”change”: ”Change title color to 2C3E50 (deep navy blue)” ,

”index”: 2, ”change”: ”Modify title shadow to color: ’rgba(0,0,0,0.35)’, blur: 3, offsetX: 1, offsetY: 1” ,

”index”: 2, ”change”: ”Move title position 50 pixels higher on the canvas” ,

”index”: 3, ”change”: ”Update tagline font family to ’Futura Light’ and reduce font size to 41px” ,

”index”: 3, ”change”: ”Change tagline color to 405D78 (lighter blue-gray)” ,

”index”: 3, ”change”: ”Modify tagline shadow to color: ’rgba(0,0,0,0.25)’, blur: 3, offsetX: 1, offsetY: 1” ,

”index”: 3, ”change”: ”Move tagline position 40 pix-

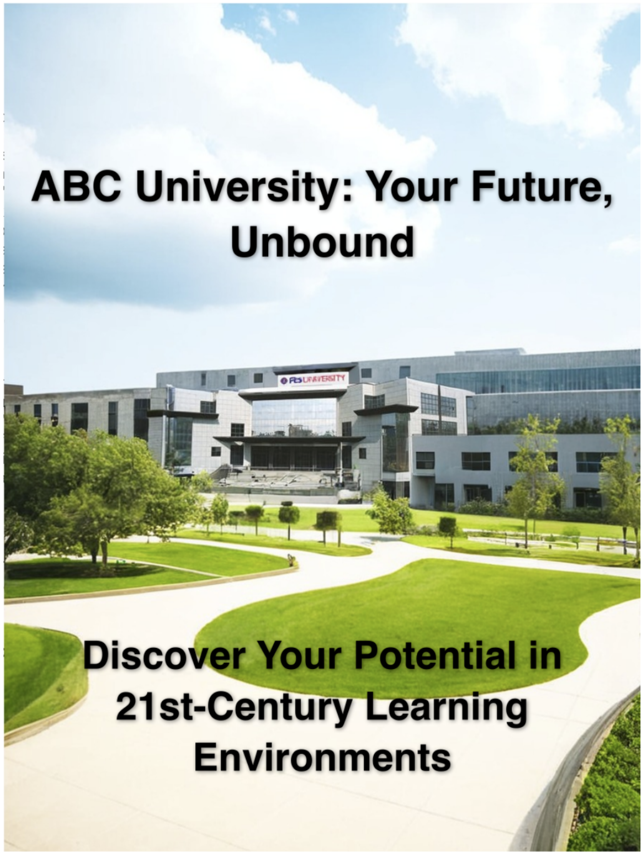

Figure 40: Initial image of university poster

els up from bottom edge and split into two lines with line height of 1.3”])

Change 1

Figure 41: Change 1: Adding a drop shadow

Text Tool User: You are given a change to be made to a textbox and the properties of the textbox by the user. You have access to the given tools/functions. Properties of text:”type”: ”textbox”, ”version”: ”5.3.0”, ”originX”: ”center”, ”originY”: ”center”, ”left”: 450, ”top”: 300.68, ”width”: 900, ”height”: 180.62, ”fill”: ”rgba(0,0,0,1)”, ”stroke”: null, ”strokeWidth”: 1, ”strokeDashArray”: null, ”strokeLineCap”: ”butt”, ”strokeDashOffset”: 0, ”strokeLineJoin”: ”miter”, ”strokeUniform”: false, ”strokeMiterLimit”: 4, ”scaleX”: 1, ”scaleY”: 1, ”angle”: 0, ”flipX”: false, ”flipY”: false, ”opacity”: 1, ”shadow”: ”color”: ”rgba(0,0,0,0.5)”, ”blur”: 5, ”offsetX”: 2, ”offsetY”: 2, ”affectStroke”: false, ”nonScaling”: false, ”visible”: true, ”backgroundColor”: ””, ”fillRule”: ”nonzero”, ”paintFirst”: ”fill”, ”globalCompositeOperation”: ”source-over”, ”skewX”: 0, ”skewY”: 0, ”fontFamily”: ”Helvetica”, ”fontWeight”: 700, ”fontSize”: 60, ”text”: ”ABC University: Your Future, Unbound”, ”underline”: false, ”overline”: false, ”linethrough”: false, ”textAlign”: ”center”, ”fontStyle”: ”normal”, ”lineHeight”: 1.16, ”textBackgroundColor”: ””, ”charSpacing”: 0, ”styles”: [], ”direction”: ”ltr”, ”path”: null, ”pathStartOffset”: 0, ”pathSide”: ”left”, ”pathAlign”: ”baseline”, ”minWidth”: 20, ”splitByGrapheme”: false, ”selectable”: true, ”hasControls”: true, ”editable”: true Change to be made: Change title color to 2C3E50 (deep navy blue)

Assistant: I’ll help change the text color to the specified hex color 2C3E50 (deep navy blue). I’ll need to convert the hex color to rgba format as required by the changeColor function.

The hex color 2C3E50 converts to rgba(44, 62, 80, 1). changeColor(”color”: ”rgba(44, 62, 80, 1)”)

Evaluation Agent:

User prompt: You are a user interface evaluator. You are given an old image of a poster and a new image of poster after a change has been made by calling a function. You must evaluate whether the change has been put into effect correctly and if the change improves the image. If it has been put into effect correctly and improves the image, output ’YES’. Otherwise, Provide feedback on how to make the change more effective. In case the executor has any issue with applying the change, it will provide feedback and it is your job to resolve any potential conflicts. Change that should be made: Change title color to 2C3E50 (deep navy blue). Function called to make the change/Feedback from Tool: ”function”: ”changeColor”, ”arguments”: ”color”: ”rgba(44, 62, 80, 1)”

Assistant: YES - The color change function has been called correctly with the right RGB values for 2C3E50 but I don’t see the title color actually changing to deep navy blue clearly in the second image

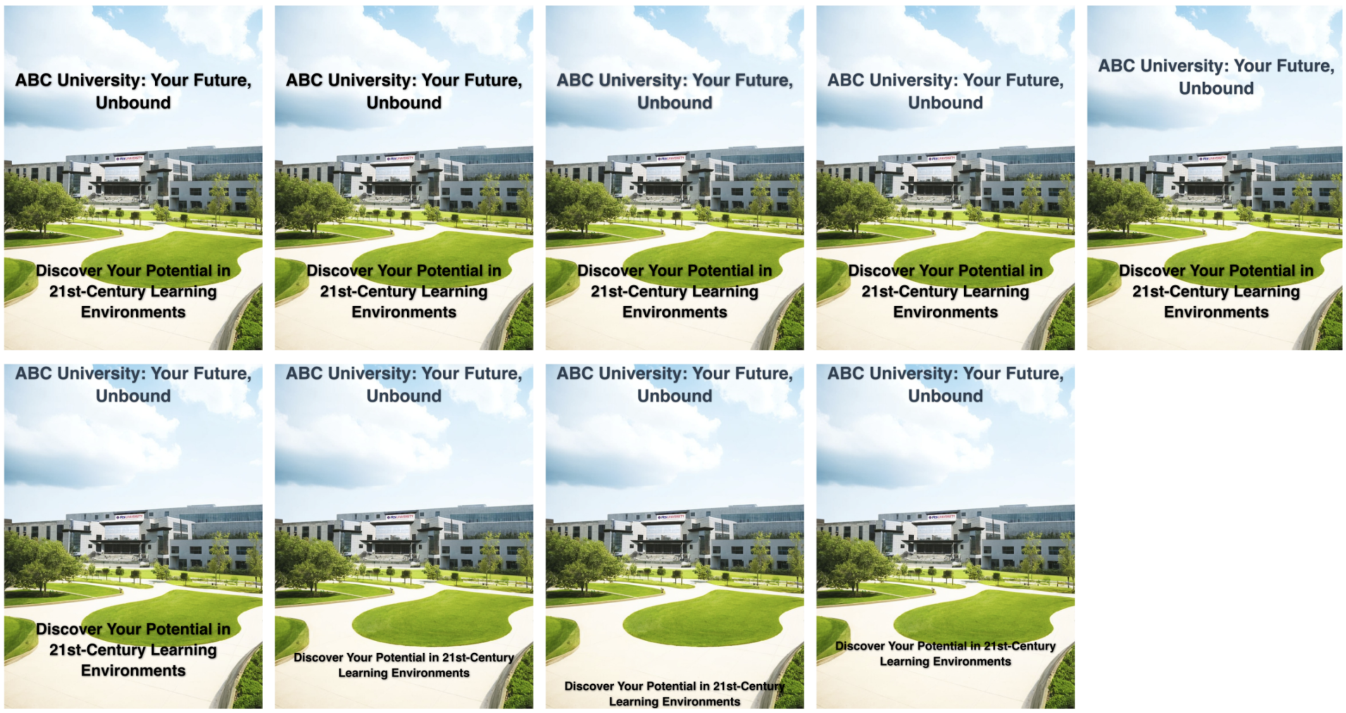

Figure 42: All iterations