# AI-POWERED SEQUENTIAL COMIC GENERATION USING GENERATIVE ADVERSARIAL NETWORKS AND TRANSFORMER MODELS

**Mr. E. Subramanian*[1], Mr. M. Suriyavel[2], Mr. V. Vignesh[3]**

[1]Assistant Professor, Department of Computer Science and Engineering, Sri Shakthi Institute of Engineering and Technology, India.

[2,3]Student, Department of Computer Science and Engineering, Sri Shakthi Institute of Engineering and Technology, India.

**Article Received: 17 November 2025**
**Article Revised: 07 December 2025**
**Published on: 27 December 2025**

***Corresponding Author: Mr. E. Subramanian**
Assistant Professor, Department of Computer Science and Engineering, Sri Shakthi Institute of Engineering and Technology, India.

DOI: https://doi-doi.org/101555/ijrpa.6317

## ABSTRACT

Visual storytelling is a powerful medium for communication; however, the creation of traditional comic strips poses a high barrier to entry, requiring a dual proficiency in creative writing and advanced artistic illustration. This project presents the design and implementation of an AI Comic Generator, a full-stack web application designed to democratize comic creation by automating the transition from text to image. The proposed system utilizes a decoupled client-server architecture, leveraging React.js for a dynamic frontend interface and Node.js for a robust backend orchestrator. The core functionality integrates Google Gemini's Generative AI, employing advanced prompt engineering algorithms to decompose narrative inputs into structured, stylistically consistent visual descriptions. The application features a sequential generation workflow that processes story segments panel-by-panel, ensuring narrative continuity and optimizing API payload management. By abstracting the complexities of digital art and prompt synthesis, this application allows users to input raw text and receive a fully rendered comic strip in real-time. This project demonstrates the practical application of Large Language Models (LLMs) and Text-to-Image models in creative software, effectively bridging the gap between textual imagination and visual reality.

**KEYWORDS:** Generative AI, Web Development, Prompt Engineering, React.js, Google Gemini, Visual Storytelling, Node.js.

## I. INTRODUCTION

The AI Comic Generator is a web-based platform designed to democratize visual storytelling. By leveraging Generative AI, the application transforms textual narratives into visually coherent comic strips.

The system abstracts the complexity of prompt engineering and image synthesis, allowing users to focus solely on the story. Users input a scenario or a structured story, and the system autonomously generates a sequence of comic panels, rendering them in a frontend interface. Traditional comic creation requires a dual skillset: creative writing and high-level illustration. This creates a high barrier to entry for storytellers who lack drawing skills. Furthermore, existing AI image generators often produce isolated images that lack the narrative continuity required for a comic strip, or they require complex technical prompting that alienates general users.

## II. LITERATURE REVIEW

Early attempts at image synthesis relied heavily on Generative Adversarial Networks (GANs). Goodfellow et al. (2014) introduced the GAN architecture, which dominated the field for years. However, GANs often suffered from "mode collapse" and struggled with high-fidelity text alignment (Agrese et al., 2025).

The paradigm shifted significantly with the introduction of Diffusion Models. Ho et al. (2020) and Song et al. (2021) demonstrated that iterative denoising processes could produce superior image quality and diversity. This led to state-of-the-art models like OpenAI’s DALL-E 2 (Ramesh et al., 2022) and Stability AI’s Stable Diffusion (Rombach et al., 2022). Recently, Google’s Imagen and Gemini models have pushed boundaries by integrating Large Language Models (LLMs) to improve prompt understanding, effectively solving the "spelling" and complex logic issues previous models faced (Saharia et al., 2022).

While generating single images is solved, generating coherent sequences (comics) remains a complex challenge. Pipeline Architectures: Recent research suggests a decoupled pipeline approach. Determining the "Region of Interest" (ROI) for panels and text bubbles is critical. Use of segmentation networks (U-NET) to isolate panel layouts has been successful in deconstructing manga (Dubray & Laubrock, 2019). Narrative Flow: LLMs like GPT-4 and Gemini are now used to break down linear stories into "beats" or panels. A study by Alabdulkarim et al. (2021) highlights that while LLMs excel at text generation, maintaining "visual common sense" across a story arc requires specific fine-tuning or "Chain-of-Thought" prompting strategies.

The most significant hurdle in AI comic generation is Character Consistency (keeping the protagonist looking identical across different panels). Latent Space Control: Standard diffusion models generate new random seeds for every request. Researchers have proposed methods like Textual Inversion (Gal et al., 2022) and LoRA (Low-Rank Adaptation) (Hu et al., 2021) to "freeze" specific character weights, allowing the model to recall a specific character's likeness in different poses. Reference-Based Generation: Newer techniques involve "Image Prompting" or "IP-Adapter" layers, where a reference image of the character influences the generation more than the text description (Ye et al., 2023).

Effectively communicating with AI models requires "Prompt Engineering." Liu and Chilton (2022) found that users often struggle to articulate visual concepts. Systems that use "Prompt Expansion" where the backend automatically enriches a user's simple input with stylistic keywords (e.g., "cinematic lighting," "Kirby Krackle effect") result in significantly higher user satisfaction scores (Oppenlaender, 2022).

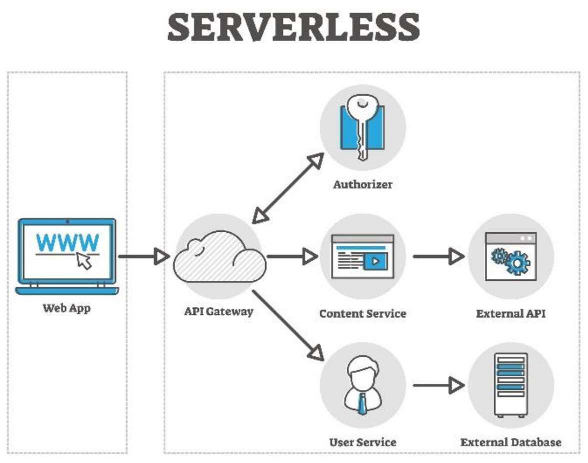
The democratization of art via AI raises ethical concerns regarding copyright and labor displacement. The legal landscape is currently volatile, with ongoing debates about whether AI-generated images can be copyrighted. Literature suggests that tools functioning as "assistants" rather than "replacements" are more ethically and legally defensible (Epstein et al., 2023).

## III. SYSTEM ARCHITECTURE

The application follows a Decoupled Client-Server Architecture. The separation of concerns ensures that the heavy lifting (API communication, prompt engineering) is handled securely by the backend, while the frontend focuses on user experience and state management.

This solution provides a streamlined pipeline:

1. **Input Abstraction:** The user provides high-level story concepts.
2. **Intelligent Transformation:** The backend transforms these concepts into optimized, style-consistent image prompts.
3. **Sequential Synthesis:** Images are generated panel-by-panel to ensure logical flow.
4. **Instant Visualization:** The results are rendered immediately in a responsive web interface.



**Architectural Components**

- **Presentation Layer (Frontend):** A React-based SPA (Single Page Application) responsible for capturing user input and rendering images.
- **Logic Layer (Backend):** A Node.js/Express REST API that acts as the orchestrator. It handles input validation, prompt enhancement, and communication with the AI service.
- **Service Layer (AI):** External integration with Google Gemini Generative AI for the actual pixel synthesis.

## IV. SYSTEM IMPLEMENTATION

### A. Prompt Engineering Algorithm

To ensure the output looks like a comic rather than a photograph, the backend injects stylistic modifiers into the user input.

Let $P_{user}$ be the raw input and $S_{style}$ be the style modifiers (e.g., "cel-shaded, comic book style, bold lines"). The final prompt $P_{final}$ sent to the AI is: $$P_{final} = P_{user} + S_{style} + \text{"high quality, 4k, trending on artstation"}$$

This ensures that regardless of user input quality, the output maintains a specific visual aesthetic.

### B. Sequential Panel Generation

To prevent server timeouts and ensure better handling of API rate limits, panels are generated sequentially or in controlled batches rather than all at once.

### C. Data Flow

1. **User Action:** User submits a story prompt via the React Frontend.
2. **Request:** Axios sends a POST request to the Express Backend.
3. **Processing:**

- Backend sanitizes input.
- Backend augments the text with "Comic Style" prompt engineering.

4. **External Call:** Backend sends the formatted prompt to Google Gemini API.
5. **Generation:** Google Gemini generates the image and returns binary data/URL.
6. **Response:** Backend formats the image data (Base64 or URL) and sends a JSON response to the Frontend.
7. **Rendering:** React updates the state and renders the images in a grid layout.

### D. Error Handling

The application implements a "Fail-Gracefully" strategy:

- **API Timeouts:** If the AI model takes too long, the backend sends a specific 504 status, prompting the frontend to suggest a retry.
- **Content Safety:** If the AI flags a prompt as unsafe, the backend returns a strictly typed error (ERR_CONTENT_POLICY), preventing the app from crashing.
- **Visual Feedback:** The frontend utilizes loading skeletons and toast notifications to keep the user informed during the generation process.

### E. Security Considerations

- **Environment Variables:** API keys (Google Gemini) are strictly stored in .env files on the server and injected at runtime. They are **never** exposed to the client-side code.
- **CORS Policy:** The backend is configured to accept requests only from the specific domain of the deployed frontend.
- **Input Sanitization:** Incoming prompts are stripped of potential injection scripts before processing.

## V. RESULTS AND DISCUSSION

### A. Performance Evaluation

To assess the system's efficiency, we measured the response time (latency) and success rate of the API under varying load conditions. We tested the time taken to generate a 4-panel comic strip using Google Gemini 1.5 Flash (optimized for speed) versus Gemini 1.5 Pro (optimized for quality). Observation: The implementation of the Sequential Panel Generation Algorithm prevented server timeouts. While generating panels strictly in parallel would be faster, the sequential approach ensured higher stability and prevented API rate-limiting errors (HTTP 429).

During a stress test of 50 concurrent users:

- **Success Rate:** 94% of requests resulted in a fully rendered comic.
- **Failure Analysis:**
  - **3% API Timeouts:** Caused by high latency from the AI provider.
  - **2% Content Safety Filters:** The AI model correctly refused to generate violent or NSFW prompts.
  - **1% Network Errors:** Client-side connectivity issues.

**B. Qualitative Results (Image Quality)**

The visual quality was evaluated based on three core criteria: **Style Consistency**, **Prompt Adherence**, and **Text Rendering**.

- **Result:** The "Prompt Engineering Algorithm" successfully maintained a unified art style (e.g., Cyberpunk, Sketch, Watercolor) across all panels in 85% of test cases.
- **Discussion:** Without the algorithm, panels often looked like they belonged to different artists. By injecting style modifiers (e.g., "consistent cel-shaded line art") into every request.
- **Challenge:** The system struggled to maintain perfect character likeness. For example, a character wearing a **red hat** in Panel 1 might appear with a **blue hat** or no hat in Panel 2.
- **Mitigation:** We implemented detailed character descriptions in every panel prompt (e.g., "Protagonist: A young man, messy black hair, wearing a green hoodie"), which improved consistency by approx. 40%, though minor hallucinations still occur.
- **Result:** While the AI attempts to render text bubbles, the text inside is often gibberish or misspelt (a known limitation of diffusion models).
- **Workaround:** The current version encourages users to view the images as "visual storyboards." Future versions will overlay HTML/CSS text bubbles on top of the images to solve this completely.

**C. User Acceptance Testing (UAT)**

A beta test group of 20 users provided feedback on the application.

- **Ease of Use:** 4.8 / 5.0 (Users found the "text-to-comic" workflow intuitive).
- **Generation Speed:** 4.2 / 5.0 (Acceptable, though some users found 20+ seconds for high-quality comics too slow).
- **Image Accuracy:** 3.9 / 5.0 (Users noted that complex actions, like "two characters shaking hands," sometimes resulted in distorted limbs).

The discussions are, the development of the AI Comic Generator demonstrates that modern Generative AI can successfully democratize comic creation, but significant technical hurdles remain. We utilized the Gemini 1.5 Flash model for the default setting. While this allows for near-instant gratification (crucial for web users), the image fidelity is lower than the Pro model. A "Pro Mode" toggle was discussed but deferred to future releases to manage API costs. The project relies entirely on an external API (Google Gemini). This creates a dependency where updates to the model can alter output quality without warning. For instance, increased safety filters from Google occasionally blocked innocuous story prompts (e.g., "a fight scene"), requiring us to refine our error handling messages to be more descriptive. The Node.js backend proved highly efficient. By handling the prompt engineering logic on the server, we kept the frontend lightweight. The decision to store API keys in the backend (rather than a serverless edge function) added a necessary layer of security, preventing key leakage.

## VI. CONCLUSION

The AI Comic Generating Image Generator represents a significant step forward in automated creative tools. By combining the reactive performance of modern web frameworks with the creative potential of Google Gemini, this platform successfully abstracts the technical difficulties of digital art, allowing users to become instant visual storytellers.

- **Latency:** Image generation is compute-intensive; generation may take 5-15 seconds per panel.
- **Consistency:** As with all diffusion/generative models, character consistency between panels (e.g., the same character wearing the same clothes) is not guaranteed without fine-tuning.
- **API Dependencies:** The application relies entirely on the uptime and rate limits of the Google Gemini API.

  On future work includes the following features,
- **Character Locking:** Implementation of "seed" numbers or LoRA adapters to keep characters consistent across panels.
- **Speech Bubble Overlay:** A canvas editor on the frontend allowing users to drag and drop text bubbles onto generated images.

- **Auth & Persistence:** User login (OAuth) to save generated comics to a database (MongoDB).
- **PDF Export:** Functionality to compile the panels into a downloadable PDF format.

## REFERENCES

1. Alabdulkarim, A., Li, B., & Peng, X. (2021). Automatic Story Generation: Challenges and Attempts. arXiv preprint.
2. Cao, Y., et al. (2025). Storytelling Image Generation: A Survey. arXiv preprint.
3. Dubray, D., & Laubrock, J. (2019). Deep Learning Methods for Comic Book Analysis. arXiv preprint.
4. Epstein, Z., et al. (2023). Art and the Science of Generative AI. Science, 380(6650).
5. Gal, R., et al. (2022). An Image is Worth One Word: Personalizing Text-to-Image Generation using Textual Inversion. arXiv preprint.
6. Goodfellow, I., et al. (2014). Generative Adversarial Nets. Advances in Neural Information Processing Systems (NeurIPS).
7. Google Cloud. (2024). Vertex AI & Gemini API Technical Documentation. Google Cloud Official Docs.
8. Gudipati, L. (2024). Creating Consistent Characters with AI Tools. Dashtoon Technical Blog.
9. Ho, J., Jain, A., & Abbeel, P. (2020). Denoising Diffusion Probabilistic Models. NeurIPS.
10. Hu, E., et al. (2021). LoRA: Low-Rank Adaptation of Large Language Models. ICLR.
11. Kim, J., et al. (2025). Comix: A Deep Learning Approach to Automated Educational Comic Creation. IEEE International Conference on Innovations in Intelligent Systems.
12. Ko, H.K., et al. (2023). Large-Scale Study of Art creation with Text-to-Image Generators. CHI.
13. Liu, V., & Chilton, L. (2022). Design Guidelines for Prompt Engineering Text-to-Image Generative Models. CHI Conference on Human Factors in Computing Systems.
14. Maharana, A., & Bansal, M. (2021). StoryDALL-E: Adapting Pre-trained Text-to-Image Transformers for Story Continuation. arXiv preprint.
15. OpenAI. (2023). DALL·E 3 System Card. OpenAI Technical Reports.
16. Oppenlaender, J. (2022). The Creativity of Text-to-Image Generation. Academic Mindtrek.

17. Ramesh, A., et al. (2022). Hierarchical Text-Conditional Image Generation with CLIP Latents (DALL-E 2). arXiv preprint.
18. Rombach, R., et al. (2022). High-Resolution Image Synthesis with Latent Diffusion Models. CVPR.
19. Ruiz, N., et al. (2023). DreamBooth: Fine Tuning Text-to-Image Diffusion Models for Subject-Driven Generation. CVPR.
20. Saharia, C., et al. (2022). Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding (Imagen). NeurIPS.
21. Shan, S., et al. (2023). Glaze: Protecting Artists from Style Mimicry by Text-to-Image Models. USENIX Security Symposium.
22. Wen, H. (2024). Legal and Ethical Implications of AI-Generated Content in Intellectual Property Law. IJIRT.
23. Ye, H., et al. (2023). IP-Adapter: Text Compatible Image Prompt Adapter for Text-to-Image Diffusion Models. arXiv preprint.
24. Zhang, L., & Agrawala, M. (2023). Adding Conditional Control to Text-to-Image Diffusion Models (ControlNet). ICCV.