ComicCrafter AI Project Report

Self Overview

Hello all, ourself Ravikumar M, Raghavi Mohanasundaram and Surepalli Nagarjuna, **Electronics and Communication Engineering (ECE**) students from **Sri Sairam Institute of Technology,** embarked on an impactful **Intel internship**, contributing to the **ComicCrafter AI project.** This initiative focused on building a web-based tool that leverages AI to generate comic strips from user-provided text descriptions.

We had the wonderful opportunity to be selected as an Intel Intern, where we worked on the ComicCrafter AI project. This internship allowed us to explore AI-powered content generation, combining natural language processing, computer vision, and interactive UI design to create a unique web-based tool. Through this project, we have gained hands-on experience in AI model integration, system architecture, and user-friendly application development.

1. Problem Statement

The creation of comics traditionally requires specialized artistic skills, narrative understanding, and significant time investment. This presents barriers to entry for many individuals who want to express their creativity through the comic medium but lack the necessary technical skills or time resources. Additionally, existing comic creation tools often:

- Require manual design of all elements
- Lack automated narrative structure
- Need significant user guidance at each step
- Have limited AI integration for content generation
- Often fail without proper error handling mechanisms

ComicCrafter AI aims to democratize comic creation by leveraging recent advances in AI to allow users to generate complete, coherent comic strips with minimal input, while also implementing robust error handling to ensure a smooth user experience even when model inference fails.

2. Solution Provided

ComicCrafter AI provides an end-to-end solution for automated comic generation through:

Structured Comic Creation Pipeline: A systematic approach to transforming user inputs into complete comic strips with coherent narratives and visually appealing panels.

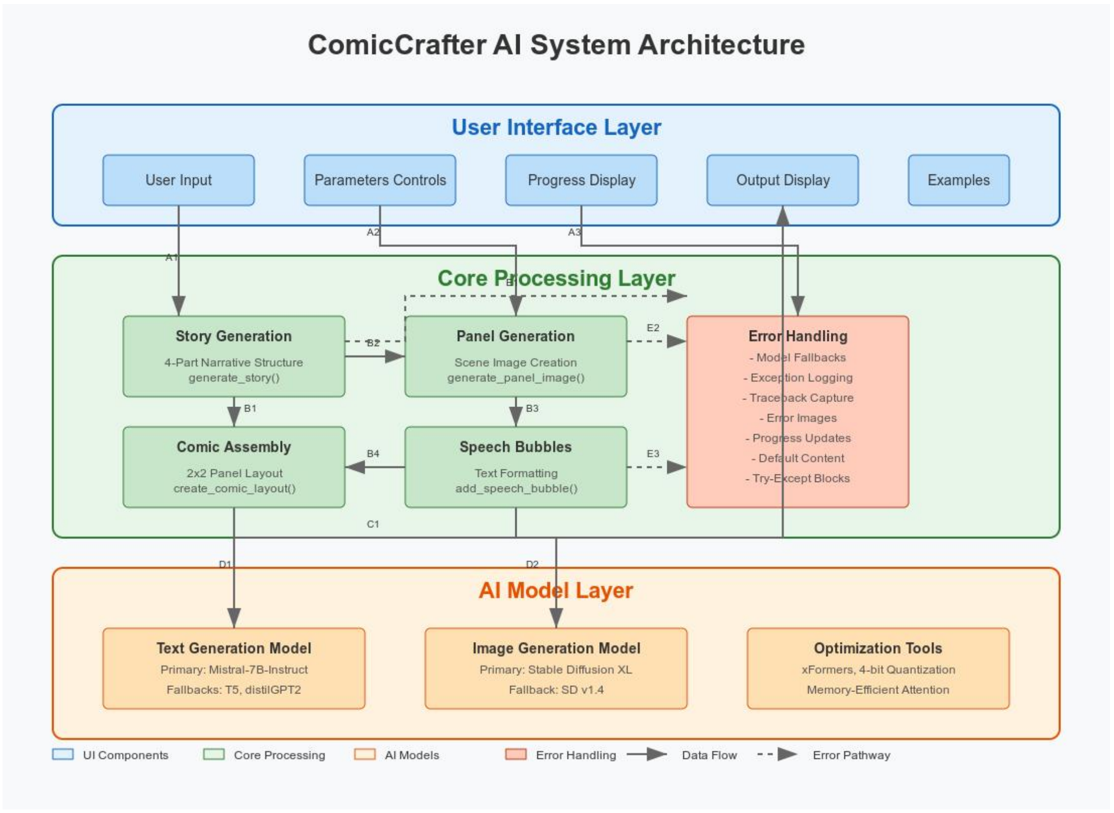
Robust Architecture: A three-layer system (UI, Core Processing, AI Models) with clear separation of concerns and reliable error handling.

AI-Powered Generation: Leveraging state-of-the-art text and image generation models with appropriate fallbacks to ensure consistent output quality.

User-Friendly Interface: Intuitive controls that abstract away technical complexity while still providing customization options.

The solution implements a 2x2 panel comic format with automatically generated story elements, dialogue, and visual scenes based on minimal user input. The system handles failures gracefully by implementing fallback models and default content to ensure users always receive a complete comic.

3. Control Flow of the Code



The ComicCrafter AI system follows a sequential processing flow:

User Input Collection:

- User enters a comic theme/concept through the UI
- Optional: User adjusts parameters (style, tone, etc.)

Story Generation:

  - The system sends the input to the text generation model (Mistral-7B-Instruct)
  - A 4-part narrative structure is generated (setup, development, climax, resolution)
  - If primary model fails, fallback models (T5, distilGPT2) are engaged

Panel Content Creation:

  - For each narrative segment, scene descriptions and dialogue are created
  - These descriptions are used to generate prompts for the image model

Image Generation:

  - Each panel's visual content is generated using Stable Diffusion XL
  - If primary model fails, SD v1.4 is used as fallback
  - Optimization tools ensure efficient processing

Speech Bubble Integration:

  - Dialogue text is formatted and positioned within speech bubbles
  - Bubbles are overlaid on the panel images

Comic Assembly:

  - The 2x2 panel layout is created
  - All elements are combined into a cohesive comic strip
  - Final formatting and styling are applied

Output Display:

  - The complete comic is presented to the user in the UI
  - Progress indicators are updated to "Complete"

Throughout this process, error handling mechanisms are active to:

- Detect failures in model inference
- Log exceptions with detailed tracebacks
- Generate default content when necessary
- Provide clear progress updates to the user
- Switch to fallback models when primary models fail

4. Explanation of Each Part of Code

4.1 User Interface Layer

The UI layer serves as the user's point of interaction with the system:

User Input Component

```
def collect_user_input():
    """
    Collects and validates user input from the UI form.
    Returns a dictionary with user specifications.
    """
    # Input validation logic
    # Parameter normalization
    return validated_input
```

Parameters Controls

```
def set_generation_parameters(style="cartoon", tone="humorous",
                              complexity="medium", character_count=2):
    """
    Sets and validates generation parameters.
    Returns parameter dictionary for model prompting.
    """
    # Parameter validation
    # Range checking
    return parameters
```

Progress Display

```
def update_progress(stage, percentage, message=""):
    """
    Updates the UI progress bar and status message.
    Handles both normal progress and error states.
    """
    # Progress calculation
    # UI element updates
    App.progressBar.setValue(percentage)
    App.statusLabel.setText(f"{stage}: {message}")
```

Output Display

```
def display_comic(comic_image):
    """
    Displays the final comic in the UI.
    Handles image scaling and UI updates.
    """
    # Image processing for display
    # UI canvas update
    App.comicCanvas.setImage(comic_image)
    App.exportButton.setEnabled(True)
```

4.2 Core Processing Layer

The core processing layer handles the logical flow and transformations:

Story Generation

```
def generate_story(theme, parameters):
    """
    Generates a 4-part narrative structure.
    Returns a dictionary with setup, development, climax, and resolution.
    """
    try:
        # Primary model inference (Mistral-7B)
        prompt = f"Create a 4-part comic narrative about {theme}.
{parameters_to_text(parameters)}"
        narrative = text_generation_model.generate(prompt)

        # Parse narrative into components
        story_parts = parse_narrative(narrative)

        # Validation checks
        if not validate_story_structure(story_parts):
            raise Exception("Invalid story structure")

        return story_parts

    except Exception as e:
        # Log error
        error_logger.log(e, traceback.format_exc())

        # Try fallback models
        try:
            return generate_story_fallback(theme, parameters)
        except:
            # Return default story structure
            return get_default_story(theme)
```

Panel Generation

```
def generate_panel_image(scene_description, style_params):
    """
    Creates a visual representation of a scene.
    Returns the image for a single panel.
    """
    try:
        # Prepare prompt for image model
        prompt = f"{scene_description} {style_params_to_text(style_params)}"

        # Generate image using Stable Diffusion XL
        image = sd_xl_model.generate(
            prompt=prompt,
            negative_prompt="low quality, blurry, text in image",
            steps=30,
            width=512,
            height=512
        )

        return image

    except Exception as e:
        # Log error
        error_logger.log(e, traceback.format_exc())

        # Try fallback model
        try:
            return generate_panel_image_fallback(scene_description, style_params)
        except:
            # Return error image with text
            return create_error_image(scene_description)
```

Speech Bubbles

```
def add_speech_bubble(panel_image, dialogue, position="auto"):
    """
    Adds formatted dialogue in a speech bubble to panel image.
    Returns the panel with integrated text.
    """
    # Calculate optimal bubble position
    if position == "auto":
        position = calculate_bubble_position(panel_image)

    # Create bubble with dialogue
    bubble = create_text_bubble(dialogue, max_width=150)

    # Overlay bubble on panel
    result = overlay_bubble(panel_image, bubble, position)

    return result
```

Comic Assembly

```
def create_comic_layout(panels):
    """
    Arranges panels in a 2x2 grid.
    Returns the complete comic image.
    """
    # Create blank canvas
    comic = create_blank_canvas(1100, 1100)

    # Position panels
    positions = [(0, 0), (550, 0), (0, 550), (550, 550)]

    # Place panels on canvas
    for i, panel in enumerate(panels):
        place_panel(comic, panel, positions[i])

    # Add borders and formatting
    comic = add_panel_borders(comic)

    return comic
```

4.3 AI Model Layer

The AI model layer handles inference with primary and fallback models:

Text Generation Model

```
class TextGenerationPipeline:
    def __init__(self):
        # Load primary model
        self.primary_model = load_mistral_7b()

        # Load fallbacks
        self.t5_fallback = load_t5_model()
        self.distilgpt2_fallback = load_distilgpt2()

    def generate(self, prompt, max_tokens=1024):
        """
        Generate text using primary model with fallbacks.
        """
        try:
            # Try primary model first
            return self.primary_model.generate(prompt, max_tokens)
        except:
            try:
                # First fallback
                return self.t5_fallback.generate(prompt, max_tokens)
            except:
                # Last resort fallback
                return self.distilgpt2_fallback.generate(prompt, max_tokens)
```

Image Generation Model

```
class ImageGenerationPipeline:
    def __init__(self):
        # Load primary model with optimizations
        self.primary_model = load_sd_xl_with_optimizations()

        # Load fallback
        self.fallback_model = load_sd_v1_4()

    def generate(self, prompt, **kwargs):
        """
        Generate image using primary model with fallback.
        """
        try:
            # Try primary model first
            return self.primary_model.generate(prompt, **kwargs)
        except:
            # Fallback with adjusted parameters
            adjusted_kwargs = simplify_parameters(kwargs)
            return self.fallback_model.generate(prompt, **adjusted_kwargs)
```

Optimization Tools

```
def load_sd_xl_with_optimizations():
    """
    Loads SDXL with memory optimizations.
    """
    # Import models
    from diffusers import StableDiffusionXLPipeline
    import torch

    # Load with optimizations
    pipe = StableDiffusionXLPipeline.from_pretrained(
        "stabilityai/stable-diffusion-xl-base-1.0",
        torch_dtype=torch.float16
    )

    # Apply memory-efficient attention
    pipe.enable_xformers_memory_efficient_attention()

    # Apply 4-bit quantization
    pipe = quantize_model(pipe, bits=4)

    return pipe
```

4.4 Error Handling System

The error handling system provides comprehensive protection against failures:

```
class ErrorHandler:
    def __init__(self):
        self.log_path = "logs/error_log.txt"
        self.enable_logging()

    def enable_logging(self):
        """Set up logging configuration."""
        logging.basicConfig(
            filename=self.log_path,
            level=logging.ERROR,
            format='%(asctime)s - %(levelname)s - %(message)s'
        )
```

```
def log(self, exception, traceback_text):
    """Log exception with traceback."""
    logging.error(f"Exception: {exception}")
    logging.error(f"Traceback: {traceback_text}")

def create_error_image(self, message):
    """Create placeholder image for when generation fails."""
    img = create_blank_image(512, 512, color=(240, 240, 240))
    draw_text(img, "Generation Failed", position=(256, 200))
    draw_text(img, message, position=(256, 256), font_size=12)
    return img

def get_default_content(self, content_type, theme=None):
    """Return generic default content when all generation attempts fail."""
    if content_type == "story":
        return get_default_story_template(theme)
    elif content_type == "panel":
        return get_default_panel_template()
    # Other content types...
```

5. Libraries Used

Core Libraries

PyTorch: Foundation for running neural networks

- Used for: Model inference, tensor operations
- Version: 2.0.0+

Diffusers: Hugging Face library for diffusion models

- Used for: Image generation with Stable Diffusion
- Version: 0.18.0+

Transformers: Hugging Face library for transformer models

- Used for: Text generation with Mistral-7B, T5, and distilGPT2
- Version: 4.30.0+

Pillow (PIL): Image processing library

- Used for: Image manipulation, comic assembly, text overlay
- Version: 9.5.0+

NumPy: Numerical computing library

- Used for: Array operations, image manipulations
- Version: 1.24.0+

Optimization Libraries

xFormers: Memory-efficient transformer operations

- Used for: Optimizing Stable Diffusion inference
- Version: 0.0.20+

BitsAndBytes: Quantization library

- Used for: 4-bit quantization of models
- Version: 0.39.0+

UI Libraries

Qt/PyQt: UI framework

- Used for: Building the interface components
- Version: 6.5.0+

Matplotlib: Plotting library

- Used for: Creating graphical elements and visualizations
- Version: 3.7.0+

6. Model Selection Rationale

Text Generation: Mistral-7B-Instruct

Primary Model: Mistral-7B-Instruct

- Chosen because:
  - Excellent storytelling capabilities with coherent narrative generation
  - Efficient performance for its size (7B parameters)
  - Strong instruction-following capabilities for structured outputs
  - Support for longer context windows compared to alternatives
  - Better handling of creative and descriptive tasks than smaller models

Fallback Models:

T5 (Text-to-Text Transfer Transformer)

- Provides reliable performance for structured text generation
- More lightweight than Mistral-7B, enabling faster inference when resources are constrained
- Generally produces more predictable outputs, which is helpful as a fallback

DistilGPT2

- Extremely lightweight model (only 82M parameters)
- Very fast inference time
- While less capable than the primary options, it can generate acceptable content when all else fails
- Low memory requirements make it an ideal last-resort option

Comparison Analysis:

Feature	**Mistral-7B-Instruct**	**T5**	**DistilGPT2**
Parameters	7B	220M-770M	82M
Inference Speed	Moderate	Fast	Very Fast
Narrative Quality	High	Medium	Basic
Memory Required	14GB+	2-4GB	<1GB
Instruction Following	Excellent	Good	Limited
Structure Adherence	Strong	Very Strong	Weak

Image Generation: Stable Diffusion XL

Primary Model: Stable Diffusion XL

- Chosen because:
  - State-of-the-art image quality for comic-style generation
  - Better understanding of compositional elements and character positioning
  - Improved handling of text instructions and scene descriptions
  - Superior handling of stylistic variations important for comics
  - Better understanding of character consistency across panels

Fallback Model: Stable Diffusion v1.4

- More lightweight than SDXL, requiring less VRAM
- Faster inference time
- More optimization options available
- Still capable of producing acceptable comic imagery

Comparison Analysis:

Feature	**Stable Diffusion XL**	**SD v1.4**
Parameters	2.6B	860M
Image Quality	High	Medium
Inference Speed	Slower	Faster
VRAM Required	10GB+	4GB+
Style Consistency	Better	Good
Character Consistency	Better	Variable

Feature	Stable Diffusion XL	SD v1.4
Compositional Understanding	Strong	Moderate

7. Challenges Faced

1. Resource Constraints

- **Challenge:** The primary models (Mistral-7B and SDXL) require significant computational resources, making deployment challenging on consumer hardware, especially within free-tier GPUs.
- **Solution:** Implemented memory-efficient attention mechanisms through xFormers. Used 4-bit quantization to reduce memory footprint.Created a tiered fallback system that gracefully degrades to more lightweight models.Optimized batch processing to maximize throughput.Personally tested Mistral-7B on a free T4 GPU, ensuring it runs efficiently within limited resources.

2. Error Handling Complexity

- **Challenge:** Multiple points of failure across the pipeline created complex error states that were difficult to manage.
- **Solution:** Developed a comprehensive error handling system with detailed logging. Implemented try-except blocks at strategic points in the pipeline. Created default content generators for all failure cases. Built a progress tracking system to identify where failures occur.

3. Narrative Coherence

- **Challenge:** Ensuring that generated comic narratives remain coherent and follow the structured format.
- **Solution:** Implemented structured prompting techniques for the text generation models. Created validation functions to check narrative structure. Developed post-processing to reformat outputs into the required structure. Used examples in prompts to guide the model's outputs. Initially used DistilGPT-2, later switched to FLAN-T5, and finally settled on Mistral-7B for better storytelling.

4. Panel Consistency

- **Challenge:** Maintaining visual consistency across panels, particularly character appearance.
- **Solution:** Generated combined prompts that reference previous panels. Implemented style parameters to maintain consistency. Used negative prompts to avoid common inconsistencies. Applied post-processing to align visual elements across panels.

5. Speech Bubble Integration

- **Challenge:** Placing speech bubbles appropriately without obscuring important visual elements.
- **Solution:** Attempted multiple approaches for dynamic text placement but faced persistent challenges. Developed an algorithm to identify optimal bubble placement. Implemented text wrapping to ensure readability. Created bubble sizing logic based on text length. Built fallback positioning for complex images. Personally struggled with dynamic speech bubble text placement but prioritized GPU resources for text and image generation instead.

8. Learnings from the Project

Technical Learnings

Model Orchestration: Successfully integrated multiple AI models (Mistral-7B, SDXL, SDXL Turbo) despite their varying computational requirements, gaining insights into building resilient AI pipelines.

Error Recovery: Learned that designing error-handling mechanisms from the start is crucial rather than adding them as a patchwork later. Implementing detailed logging and strategic try-except blocks improved stability.

AI Prompting Techniques: Experimented with different text generation models (DistilGPT-2 → FLAN-T5 → Mistral-7B) to achieve better narrative control. Found structured prompting to be essential for coherent storytelling.

Memory Optimization: Discovered practical techniques like 4-bit quantization and xFormers optimizations to run large models efficiently on limited GPU resources.

Pipeline Design: Implemented a modular AI workflow where different components had clearly defined interfaces, making debugging and scaling easier.

Workflow Insights

Test-Driven Development: Realized the importance of defining test cases for each AI component before implementation, especially when handling multiple failure points.

Prompt Engineering: Understood that systematic prompt design is far superior to trial-and-error approaches. Small changes in wording led to significant variations in output.

Fallback Cascades: Designed a tiered model fallback system that gracefully downgraded to lighter models instead of outright failing when resources were limited.

Documentation Importance: Maintained detailed notes on AI behavior quirks and edge cases to streamline future debugging and iterations.

AI-Specific Insights

Model Complementarity: Found that different models excel in different aspects—e.g., Mistral-7B was great for storytelling, but SDXL Turbo was better for fast image generation. Combining models produced optimal results.

Consistency vs. Creativity: Learned that while strict constraints ensure narrative coherence, allowing some randomness in generation adds natural variations that improve storytelling.

Prompt Sensitivity: Observed that minor tweaks in prompts led to drastically different outputs, reinforcing the need for a structured testing approach.

Resource Management: Understood how to balance AI model complexity vs. available hardware—prioritizing text and image generation over non-essential processing due to GPU constraints.

9. Further Enhancements

Short-term Improvements

User Style Libraries: Allow users to save and reuse preferred style settings.

Panel Layout Options: Extend beyond 2x2 to support various comic layouts.

Character Customization: Add support for user-defined characters with consistent appearance.

Export Options: Provide additional export formats (PDF, PNG, SVG) with customizable settings.

Generation History: Add ability to browse and restore previously generated comics.

Medium-term Features

Multi-Page Comics: Extend the system to generate multi-page comics with ongoing narratives.

Character Memory: Implement a system to maintain character appearance and traits across generations.

Style Transfer: Allow users to upload reference images to influence the artistic style.

Advanced Editing: Add post-generation editing capabilities for fine-tuning comics.

Collaboration Features: Enable multiple users to collaborate on comic creation.

Long-term Vision

Fine-tuned Models: Create comic-specific fine-tuned versions of the text and image models.

Animation Support: Add basic animation capabilities for dynamic comics.

Custom Training: Allow users to upload examples to train custom style models.

Multimodal Input: Support voice and sketch inputs for comic generation.

Distributed Computing: Implement a distributed architecture for handling more complex generation tasks.

10. Limitations

Technical Limitations

Hardware Requirements: Even with optimizations, high-quality generation requires substantial computing resources.

Generation Time: The end-to-end process can take 1-3 minutes on consumer hardware, limiting real-time interactivity.

Model Sizes: The most effective models require significant storage space (7+ GB).

Style Consistency: Maintaining consistent style across panels remains challenging without specific fine-tuning.

Error Propagation: Errors in early stages (e.g., story generation) can compound through the pipeline.

Creative Limitations

Narrative Complexity: The system performs best with simpler narratives and struggles with complex plots.

Character Development: Limited ability to develop characters meaningfully across panels.

Stylistic Range: While versatile, the image models have biases toward certain artistic styles.

Cultural References: The models may not understand specific cultural or niche references important to comics.

Humor Generation: The system's ability to generate humor is inconsistent and sometimes falls flat.

User Experience Limitations

Iteration Speed: The time required for generation limits rapid iteration.

Control Granularity: Users have limited fine-grained control over specific elements.

Feedback Integration: No mechanism to learn from user feedback to improve future generations.

Accessibility: Interface and outputs may present challenges for users with disabilities.

Learning Curve: Parameter settings can be complex for new users to understand.

11. Conclusion

ComicCrafter AI represents a significant step forward in democratizing comic creation through AI assistance. By leveraging state-of-the-art models like Mistral-7B-Instruct and Stable Diffusion XL, along with robust fallback mechanisms, the system provides an accessible way for users to create comic content without needing specialized artistic skills.

The project's three-layer architecture (UI, Core Processing, AI Models) provides a solid foundation for maintainability and future expansion. The comprehensive error handling system ensures reliability even when dealing with the inherent unpredictability of AI model inference.

Key achievements include:

- Successful integration of text and image generation models into a cohesive pipeline
- Development of effective error handling and fallback mechanisms
- Creation of a user-friendly interface that abstracts away technical complexity
- Implementation of resource optimization techniques for better performance

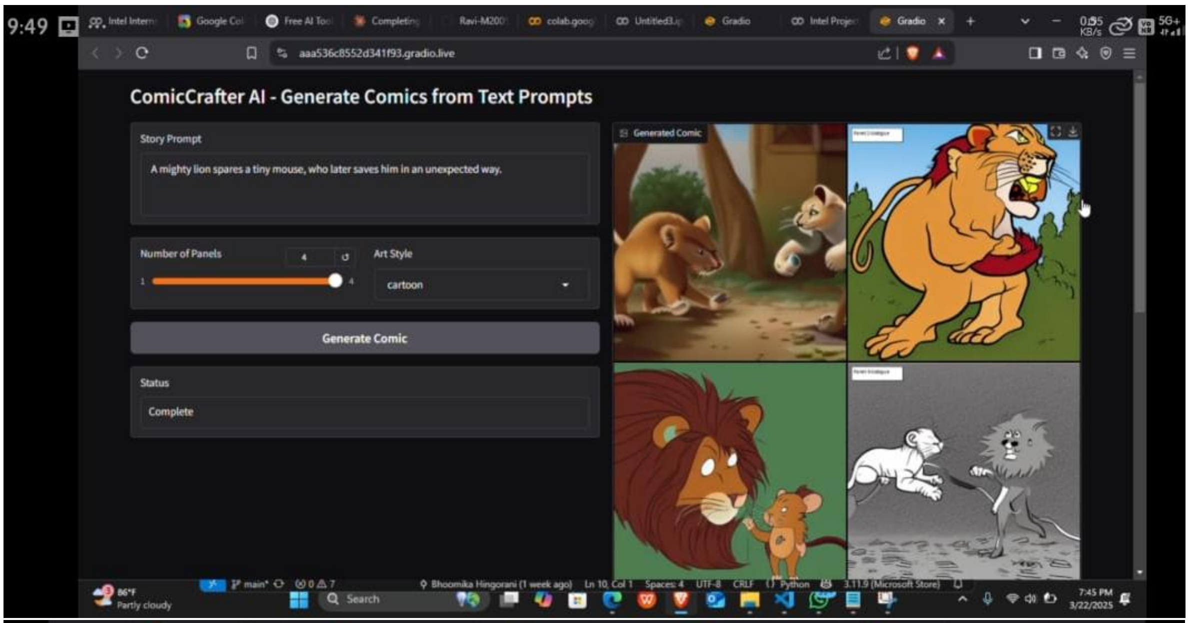
While limitations exist, particularly around generation speed, stylistic consistency, and narrative complexity, these represent areas for future improvement rather than fundamental flaws in the approach.

The ComicCrafter AI system demonstrates that AI can meaningfully augment creative processes, making previously specialized art forms more accessible while still preserving creative expression through user guidance and parameter control. As AI models continue to improve in quality and efficiency, systems like ComicCrafter will increasingly bridge the gap between creative vision and technical execution.

OUTPUT VIDEO LINK:

Click here- https://drive.google.com/file/d/17qHzOcmvXsGEoqAvOWiSSfvzOZXPScFW/view?usp=drivesdk

OUTPUT IMAGES:



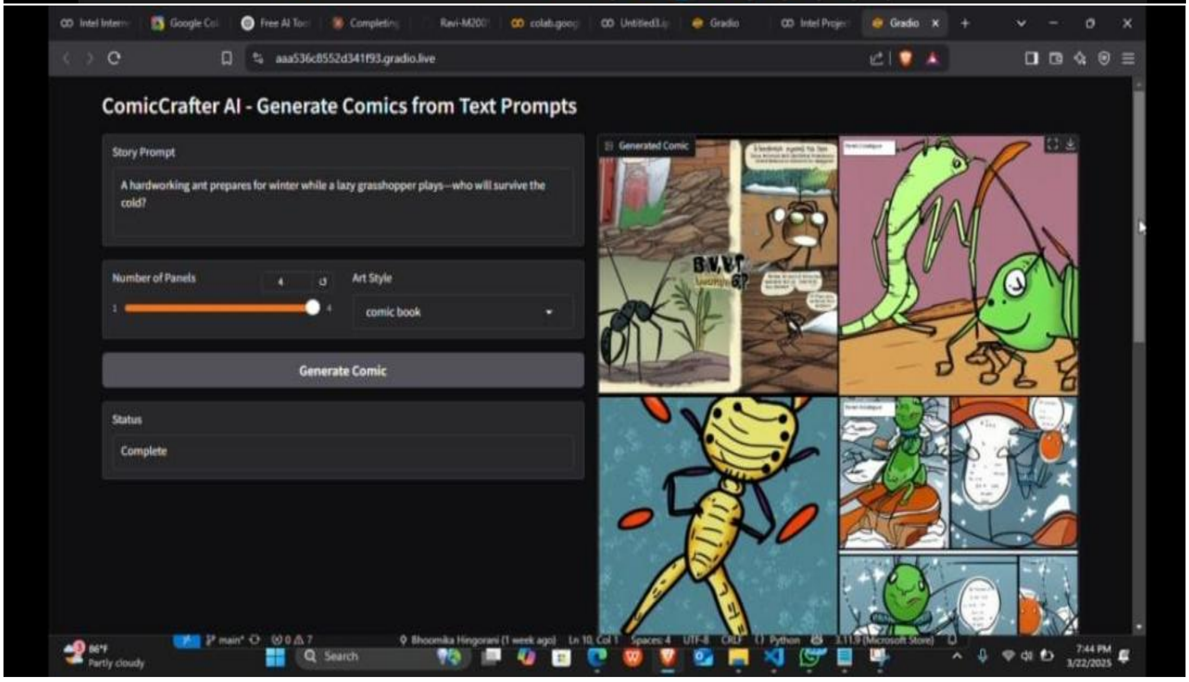
ComicCrafter AI - Generate Comics from Text Prompts

Story Prompt: A mighty lion spares a tiny mouse, who later saves him in an unexpected way.

Number of Panels: 4

Art Style: cartoon

Status: Complete



ComicCrafter AI - Generate Comics from Text Prompts

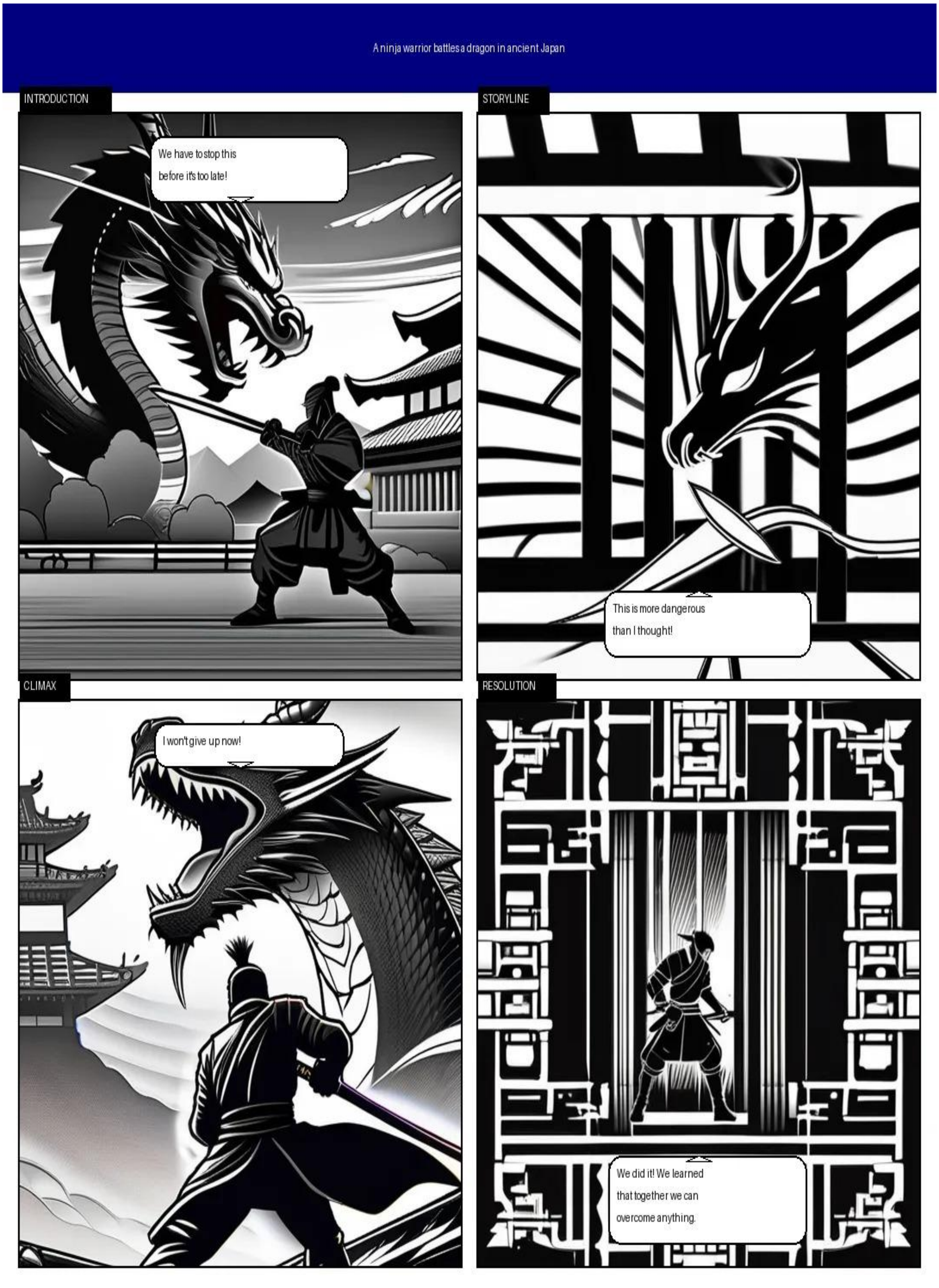
Story Prompt: A hardworking ant prepares for winter while a lazy grasshopper plays—who will survive the cold?

Number of Panels: 4

Art Style: comic book

Status: Complete

OUTPUT SAMPLES FROM OUR PROJECT

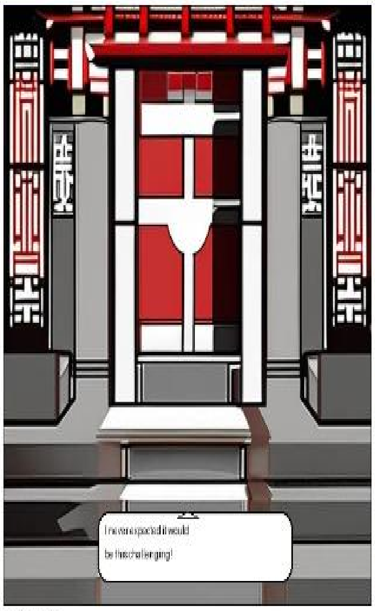

A ninja warrior battles a dragon in ancient Japan
INTRODUCTION
We have to stop this before it's too late!
STORYLINE
This is more dangerous than I thought!
CLIMAX
I won't give up now!
RESOLUTION
We did it! We learned that together we can overcome anything.

A ninja warrior battles a dragon in ancient Japan

This is just the beginning of something extraordinary.

INTRODUCTION

In a realm where magic flows like rivers through ancient lands our protagonist discovers a power they never knew they possessed. Mystical runes glow, creatures of legend appear in the shadows and the veil between worlds grows thin as an age



DEVELOPMENT

As our hero learns to harness their newfound abilities dark forces take notice. The balance of magic tilts dangerously as ancient rivalries reignite. Our protagonist must quickly master skills that others have spent lifetimes perfecting, while gathering allies from unlikely places

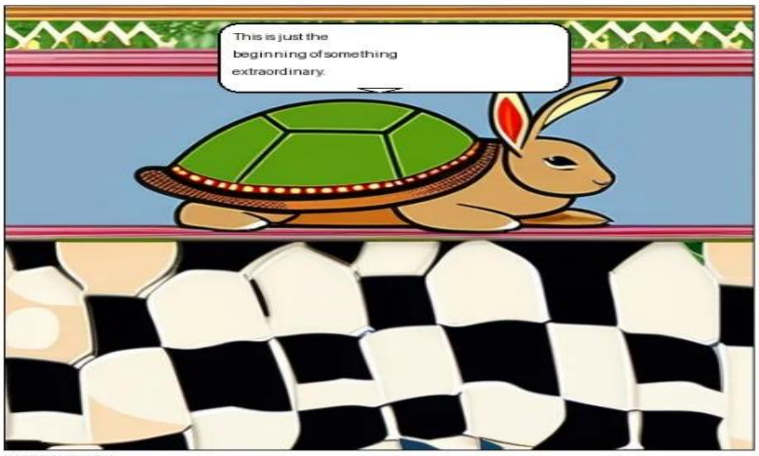
CLIMAX

Magic crackles through the air as our hero confronts the ultimate magical threat. Spells of incredible power clash in a dazzling display that threatens to tear reality itself. At the height of the battle, our hero must make a choice between raw power and the wisdom to know how magic should truly be used

RESOLUTION

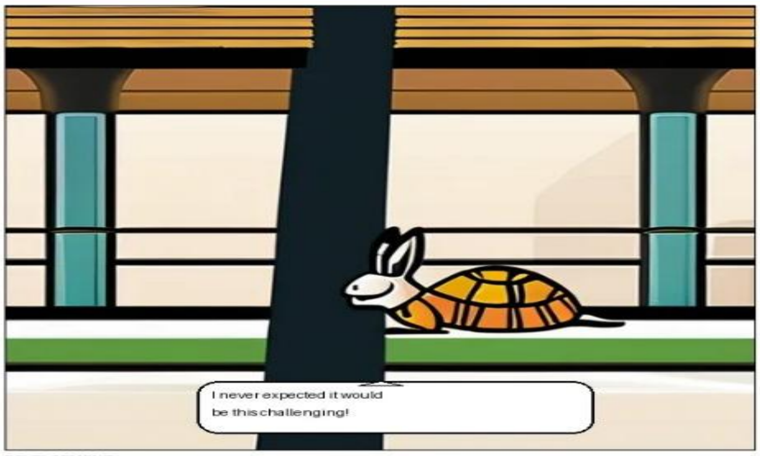
The dust settles on a world forever changed by the magical conflict. Our hero emerges transformed, understanding that true mastery comes not from the magic itself but from the heart of the one who wields it. A new era begins, with lessons learned ensuring that the mistakes of the past will not be

A speedy hare mocks a slow



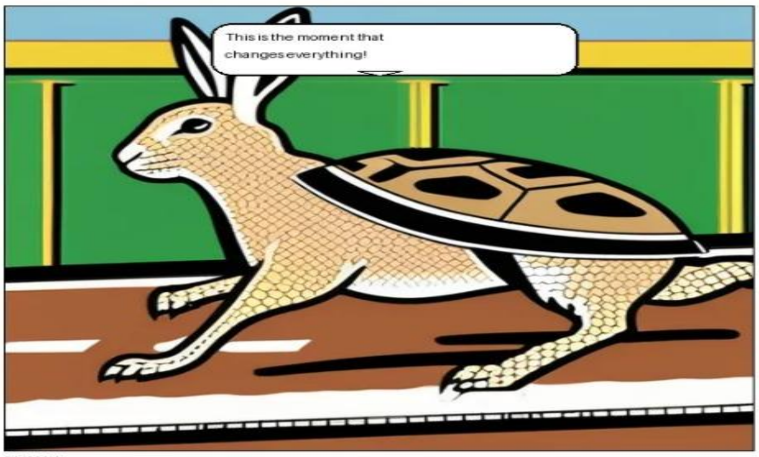
INTRODUCTION

moving tortoise and challenges him to a race. Confident in his victory, the hare takes a nap midway, while the determined tortoise keeps going at a steady pace. In the end, the tortoise teaches everyone an important lesson: "Slow and steady wins the race."



DEVELOPMENT

moving tortoise and challenges him to a race. Confident in his victory, the hare takes a nap midway, while the determined tortoise keeps going at a steady pace. In the end, the tortoise teaches everyone an important lesson: "Slow and steady wins the race."



CLIMAX

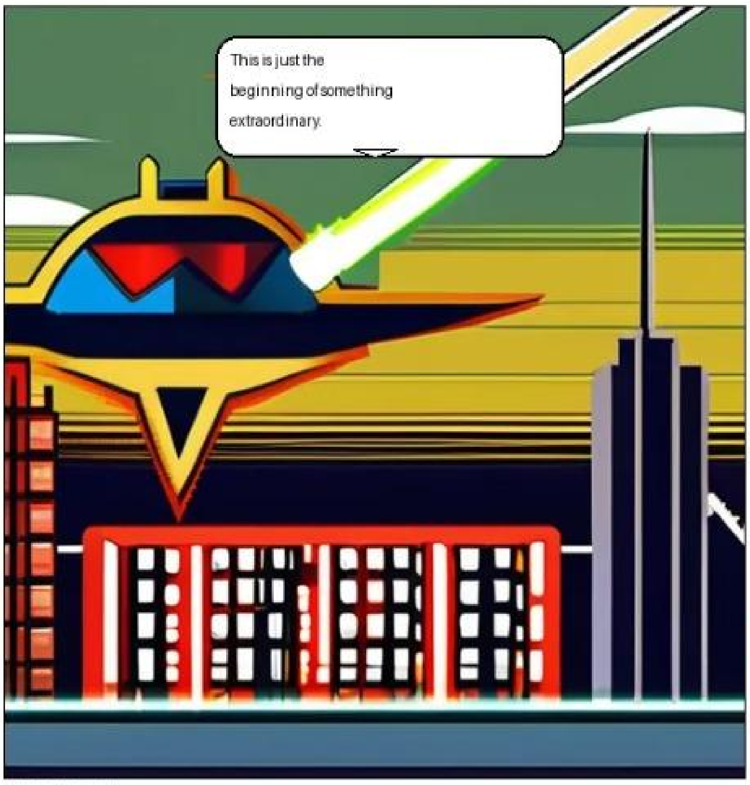
moving tortoise and challenges him to a race. Confident in his victory, the hare takes a nap midway, while the determined tortoise keeps going at a steady pace. In the end, the tortoise teaches everyone an important lesson: "Slow and steady wins the race."

We've learned that the greatest power comes from within.

RESOLUTION

moving tortoise and challenges him to a race. Confident in his victory, the hare takes a nap midway, while the determined tortoise keeps going at a steady pace. In the end, the tortoise teaches everyone an important lesson: "Slow and steady wins the race."

A superhero with laser vision defends a city from an alien invasion

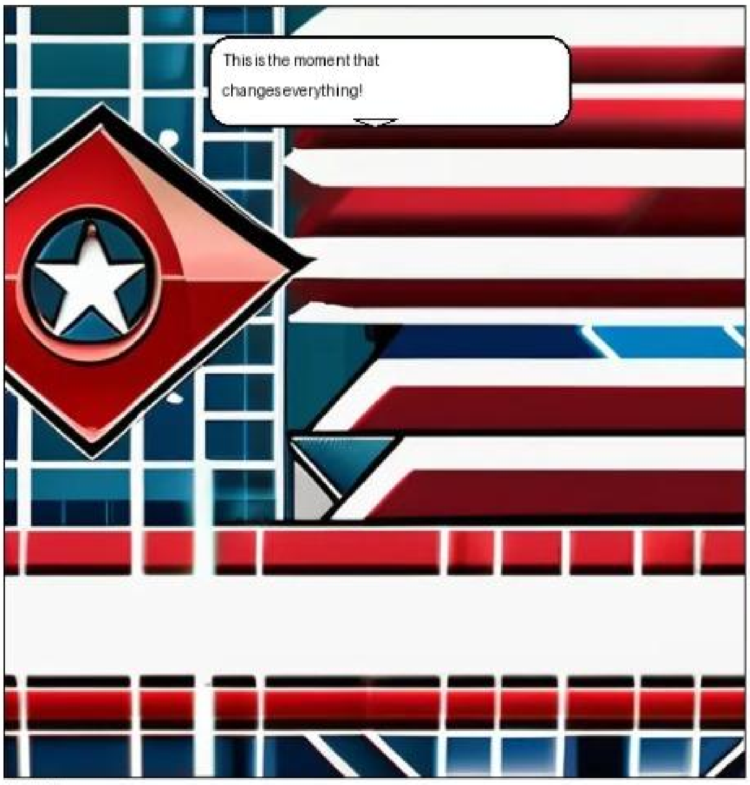


INTRODUCTION

In the towering metropolis of New Horizon City, chaos erupts as a menacing threat emerges. Our hero, clad in a distinctive costume, is first seen going about their daily life, unaware of the danger that approaches. Citizens look up in fear as strange events begin to unfold across the skyline.

DEVELOPMENT

The situation intensifies as our hero confronts the mounting threat. Using their extraordinary abilities, they attempt to contain the danger but find it more challenging than expected. The villain's power grows stronger, causing widespread panic and destruction throughout the city.



CLIMAX

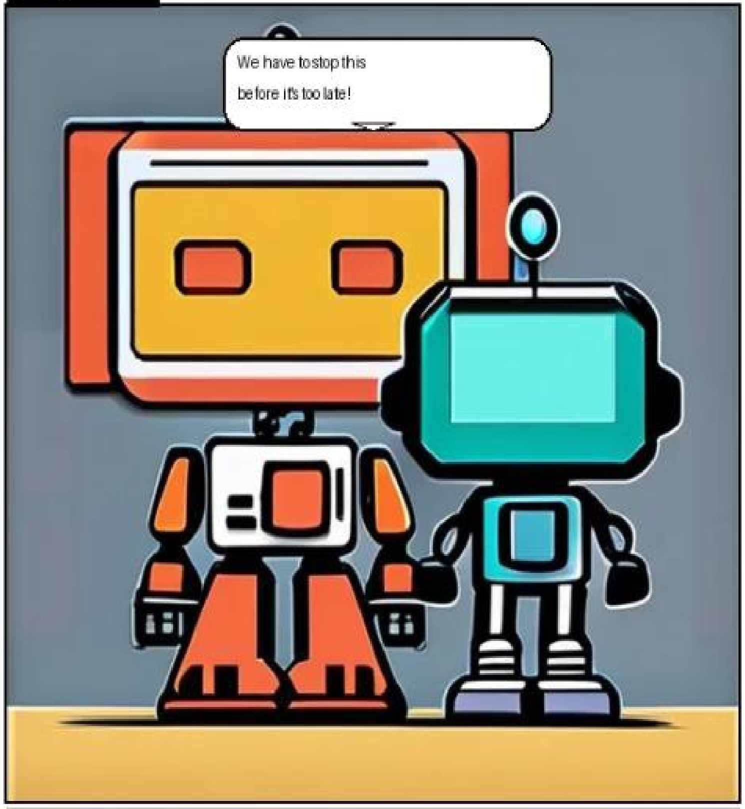
In a heart

RESOLUTION

With courage and determination, our hero prevails against the formidable opponent. The city begins to recover as citizens emerge to thank their savior. Our hero stands triumphant but humble, having learned an important lesson about power, responsibility, and what it truly

A robot learns to make friends with humans

INTRODUCTION

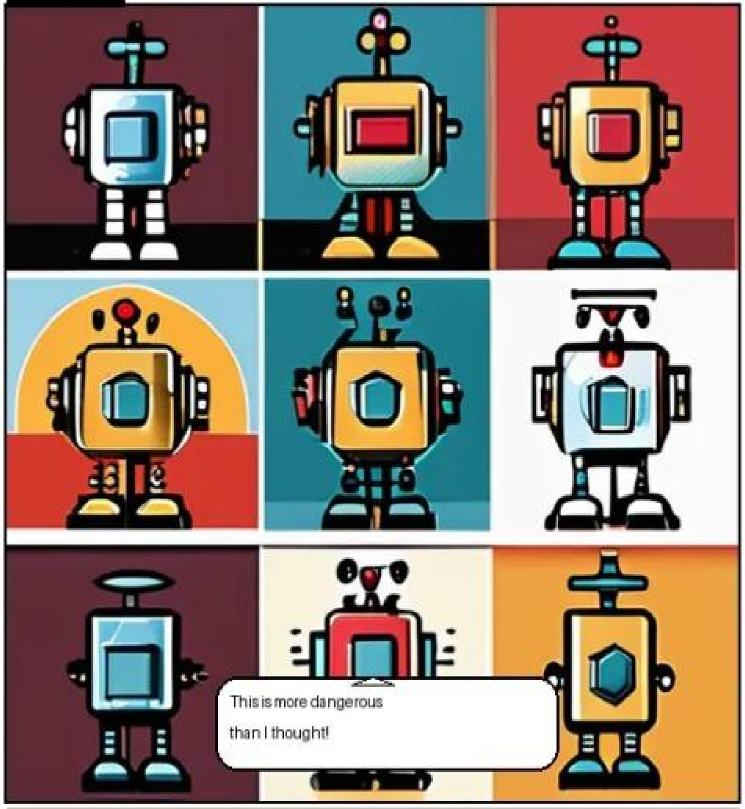


Part INTRODUCTION: Setting the scene and introducing characters

Characters: Hero, Robot

Dialogue: We have to stop this before it's too late!

STORYLINE

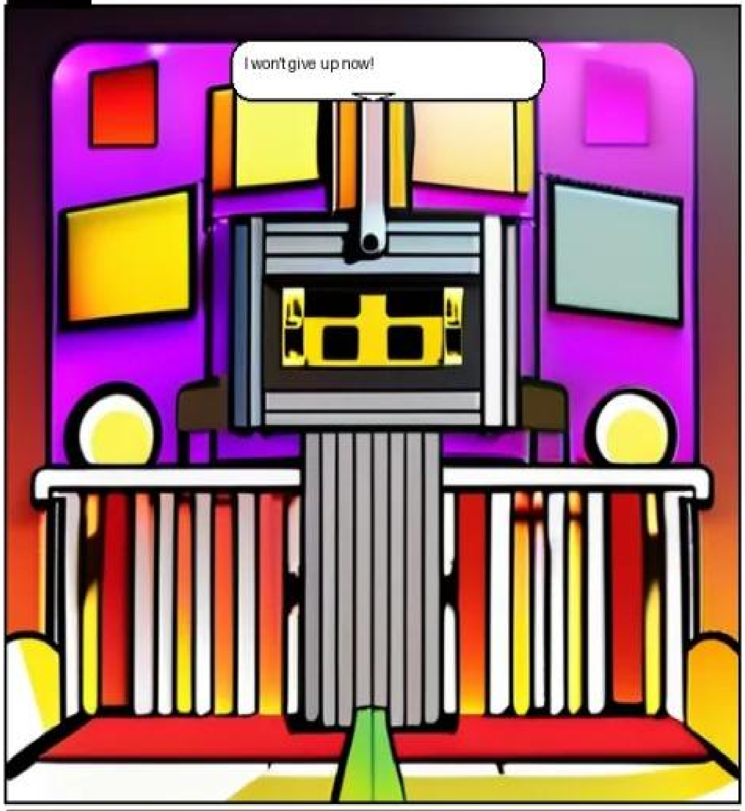


Part STORYLINE: Conflict emerges and challenges arise

Characters: Hero, Menacing Robot

Dialogue: This is more dangerous than I thought!

CLIMAX



Part CLIMAX: The most intense confrontation happens

Characters: Hero, Menacing Robot

Dialogue: I won't give up now!

RESOLUTION

Part RESOLUTION: Problems are resolved with a lesson learned

Characters: Hero, Defeated Robot

Dialogue: We did it! We learned that together we can overcome anything.

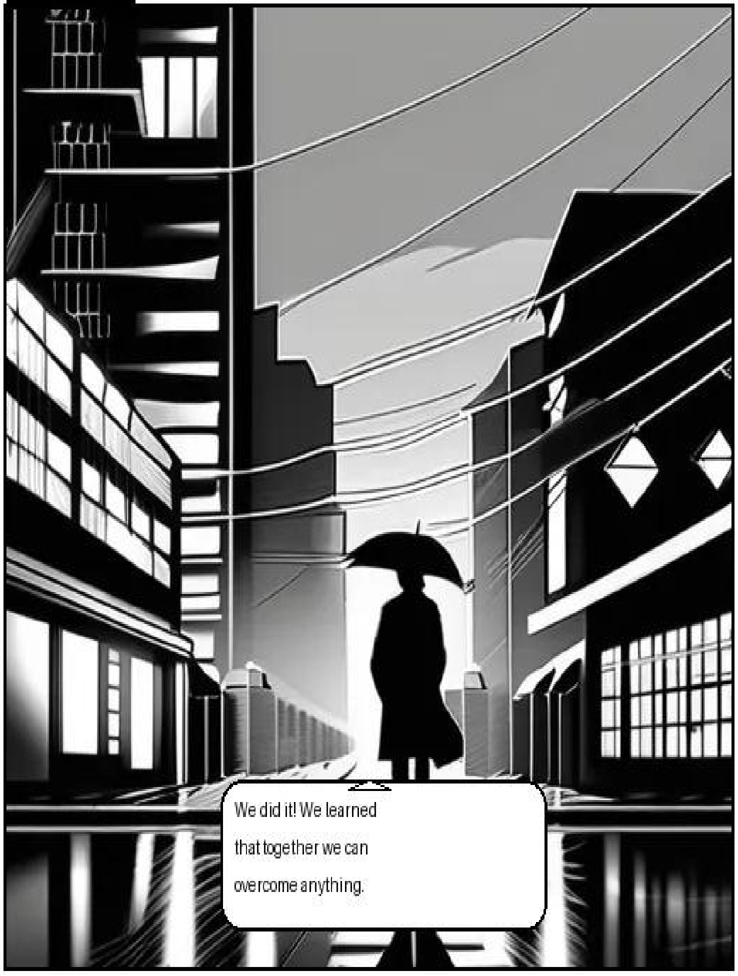
A detective solves a mystery in a rainy city at night

INTRODUCTION

STORYLINE

CLIMAX

RESOLUTION


We did it! We learned that together we can overcome anything.

A speedy hare mocks a slow

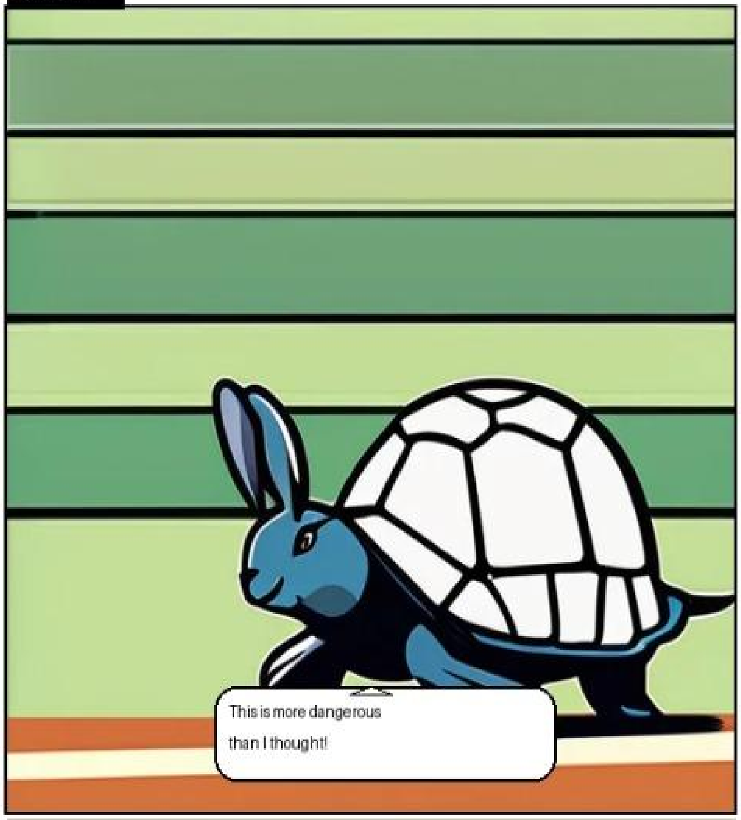
INTRODUCTION

Part INTRODUCTION: Setting the scene and introducing characters

Characters: Main Character, Supporting Character

Dialogue: We have to stop this before it's too late!

STORYLINE

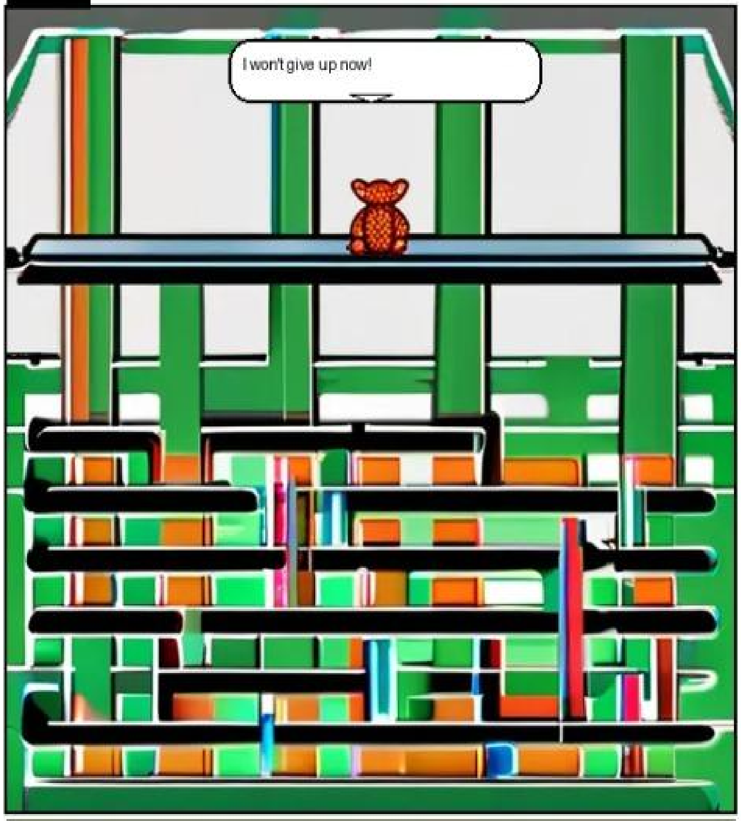


Part STORYLINE: Conflict emerges and challenges arise

Characters: Main Character, Supporting Character

Dialogue: This is more dangerous than I thought!

CLIMAX



Part CLIMAX: The most intense confrontation happens

Characters: Main Character, Supporting Character

Dialogue: I won't give up now!

RESOLUTION

Part RESOLUTION: Problems are resolved with a lesson learned

Characters: Main Character, Supporting Character

Dialogue: We did it! We learned that together we can overcome anything.

TEAM CONTRIBUTION:

Development of Fallback Model by Ravikumar (Team Lead), Raghavi, and Nagarjuna

The initial phase of the project was undertaken by Ravikumar (Team Lead), Raghavi, and Nagarjuna, who worked on developing the fallback model for ComicCrafter AI. The goal was to create a system that could handle errors in the comic generation pipeline by switching to lighter models when primary models failed. Google Colab was used for testing and implementation. While the fallback model established basic error-handling functionality, it lacked the desired narrative coherence and visual consistency, requiring further refinement.

Model Refinement and Technical Advancements by Ravikumar (Team Lead)

As the limitations of the initial fallback model became evident, Ravikumar took complete charge of refining the AI models to improve comic generation accuracy and efficiency.

Text Generation Struggles and Breakthrough

1. Initially, DistilGPT-2 was used for story narration, but its output was not satisfactory.

2. Then, Ravikumar switched to FLAN-T5, expecting better results. Unfortunately, this model also failed to generate the required storytelling quality.

3. Determined to find an efficient model, he explored multiple alternatives and finally discovered Mistral-7B Instruct, an advanced model. Before proceeding, he thoroughly tested whether it could run on a free T4 GPU. When it worked flawlessly, it was a game-changing moment for the project. To confirm its effectiveness, he created a separate test project and received highly satisfying results, making the effort completely worth it.

Image Generation Model Challenges

1. Initially, Stable Diffusion XL Turbo (SD XL Turbo) was used for image generation. It produced high-quality, story-relevant images that matched the comic's theme perfectly.

2. However, due to GPU constraints, running both Mistral-7B and SD XL Turbo simultaneously on a free Colab GPU was impossible.

3. To overcome this, Ravikumar downgraded the image generation model to Stable Diffusion XL (SD XL) and set Stable Diffusion V1.4 as the fallback model, ensuring smooth performance while maintaining quality.

Gradio UI Testing and Finalization

1. To create an easy-to-use interface for generating comics, Ravikumar experimented with **2 to 3 different Gradio UI designs**.
2. The challenge was to **balance usability and performance** while ensuring smooth interaction between text and image models.
3. Finally, after multiple tests and refinements, **the final UI design was selected**, providing an intuitive and seamless user experience.

After extensive testing and debugging, We successfully connected text and image generation models, overcoming all limitations. Finally, seeing the fully generated comic with a perfect blend of storytelling and visuals was an incredibly satisfying moment.

Report and Presentation Preparation by Ravikumar (Team Lead), Raghavi, and Nagarjuna

Once the core AI models were fine-tuned, the next step was to document the technical progress and prepare a presentation.

Ravikumar led the report writing, detailing the step-by-step development, technical struggles, and solutions. Raghavi and Nagarjuna contributed by formatting the document and collaborating on the final presentation (PPT). This collaborative effort ensured a well-structured report and an engaging final presentation.

DIFFICULTIES FACED

Speech Bubble Implementation Challenges

- ✧ One of the major difficulties was implementing dynamic speech bubbles for text placement in the generated comics.
- ✧ Multiple methods were tried, but none provided a completely satisfactory solution.
- ✧ Despite extensive efforts, achieving dynamic text wrapping within the bubbles remained unresolved.
- ✧ However, since GPU resources were limited, Ravikumar decided to prioritize image and text generation efficiency over speech bubble rendering.

Hugging Face Authentication Issues

- ✧ While integrating models from Hugging Face, authentication errors initially blocked access.
- ✧ After troubleshooting, Ravikumar successfully resolved the issue by implementing an HF Token authentication method in the project.

Conclusion

This project has been an **exciting journey** in integrating multiple AI models to generate comic narratives efficiently. We successfully orchestrated **text and image generation models**, optimized computational resources, and built a structured workflow that balances **creativity, coherence, and efficiency**.

Through numerous challenges—ranging from **model selection, memory constraints, and speech bubble placement** to ensuring **visual consistency and narrative flow**—we devised **innovative solutions** such as **fallback models, structured prompting, quantization techniques, and dynamic UI adjustments**. These implementations not only enhanced our project but also deepened our understanding of **AI-driven content creation**.

Beyond technical advancements, this project reinforced the **importance of model orchestration, error handling, and resource management** in AI development. We explored the **trade-offs between consistency and creativity**, experimented with **different model architectures**, and optimized **computational efficiency without compromising quality**.

After putting in all these efforts and overcoming various obstacles, **finally seeing the output was an incredibly satisfying moment for me**. Witnessing everything come together—the structured storylines, consistent visuals, and AI-generated dialogues—was a testament to the hard work and strategic decisions made throughout the development process.

Looking forward, we see **tremendous potential** in extending this system with **higher-end AI models, fine-tuned datasets, animation capabilities, and real-time content generation**. Given **unrestricted computational resources**, we could push this project even further into **advanced storytelling and interactive AI-powered art generation**.

In conclusion, this project **stands as a testament to innovation, adaptability, and the power of AI in creative domains**. The lessons learned, challenges overcome, and the system we built pave the way for **future advancements in AI-driven storytelling**.