Project Documentation

ComicCrafter AI is a generative AI based comic generator running locally on edge devices that generates a comic style story based on the input prompts given by the user.

Intel® Unnati Industrial Training 2025

Team Name: COMIC-BUILDER

Prepared By :

Pritam Mondal

Madhusudan Mahatha

Md Usman Ansari

Mentor : Dr. Shivnath Ghosh

---

1. Overview

Comic Builders is a cutting-edge web-based application designed to automatically generate comic strips from textual narratives. Users can input four key components of a short story — Introduction, Storyline, Climax, and Moral — and the application uses LLaMA, an advanced language model, for story generation. Stable Diffusion is then employed to convert these generated stories into corresponding comic panels. These panels are packaged into a downloadable ZIP file, providing an easy way for users to download and share their personalized comic strips.

This application aims to streamline the comic creation process, making it accessible to individuals and organizations without requiring artistic or technical expertise.

2. Objectives

The Comic Builders project aims to democratize comic strip creation using cutting-edge artificial intelligence. Traditionally, creating a comic requires both writing and artistic skills, but Comic Builders removes these barriers by combining the capabilities of LLaMA for story generation and Stable Diffusion for image creation.

The core objectives of the project include:

1. Simplification: Allowing users to generate comic strips quickly and easily from text.

2. Accessibility: Enabling non-artists to create high-quality comic strips based on their creative writing.

3. Creativity: Encouraging users to express their narratives through both written text and visual comics.

4. Scalability: Ensuring that the platform can handle a large number of users simultaneously without performance degradation.

3. Technologies Used

5. Streamlit (Version: 1.44.1): For developing the web-based interface.

6. Llama (llama-cpp-python, Version: 0.3.8): For generating story elements.

7. Stable Diffusion XL (Version: 2.1): For generating comic-style images.

8. Python (Version: 3.11): The primary programming language for backend logic.

9. Pyngrok (Version: 7.2.3): For exposing the local Streamlit app to the internet.

10. Torch (Version: 2.6.0+cu124): For deep learning and AI model processing.

11. Transformers (Version: 4.50.2): For NLP-based tasks in text generation.

12. Diffusers (Version: 0.32.2): For working with diffusion models like Stable Diffusion.

13. OpenCV (Version: 4.11.0.86): For image processing and manipulation.

14. NumPy (Version: 2.0.2): For numerical computations.

15. Pandas (Version: 2.2.2): For handling structured data

5. System Workflow

The Comic Builders application follows a streamlined workflow that combines story generation and image creation to provide users with a comprehensive solution for comic strip production.

4.1 User Input

Users are prompted to input four key sections of their story:

- Introduction: Establishes the scene and introduces characters.

- Storyline: Describes the main events of the narrative.

- Climax: Highlights the turning point or dramatic moment.

- Moral: Concludes the story with a key takeaway or lesson.

4.2 Image Generation with Stable Diffusion

After the story is generated, each section is converted into a text prompt and sent to Stable Diffusion. The model processes these prompts to generate a series of comic panels:

Panel 1: Represents the Introduction.

Panel 2: Depicts the Storyline.

Panel 3: Illustrates the Climax.

Panel 4: Concludes with the Moral of the story.

4.3 Display and Interaction

The generated comic panels are displayed on the user interface, allowing users to view their story in visual form. Each panel corresponds to one part of the story, ensuring clarity and ease of navigation.

4.4 Download and Export

Once the comic panels are generated, they are packaged into a ZIP file using Python's Zipfile library. This makes it convenient for users to download all comic panels at once.

5. Project Roles and Responsibilities

- The development of **Comic Builder** was a collaborative effort involving multiple team members, each with specialized roles to ensure the seamless integration of AI-powered storytelling and image generation into an interactive application. The key roles and responsibilities are outlined below:

- **Image Generation – Pritam Mandal:**

- Pritam Mandal was responsible for integrating **Stable Diffusion XL (SDXL)** to generate high-quality comic-style images corresponding to different story elements. His contributions included:

- Implementing **StableDiffusionXL Pipeline** from the diffusers library to handle image generation.

- Optimizing image quality by adjusting prompts dynamically based on the selected comic style (Manga, Cartoon, Realistic, Vintage).

- Ensuring seamless GPU/CPU compatibility to enhance model efficiency across different hardware configurations.

- Developing functions to structure **collage layouts** (2x2 grid, vertical strip, horizontal strip) to present the generated panels in a visually appealing format.

- Handling image processing and exporting, allowing users to **download** their AI-generated comic as a zip file.

- **Story Generation – Madhusudan:** led the development of the **LLM-powered story generation module**, responsible for dynamically generating structured narratives based on user input. His responsibilities included:

- Integrating a lightweight transformer-based text generation model (GPT-2 in the prototype) to generate four structured story sections: **Introduction, Storyline, Climax, and Moral.**

- Designing **prompt engineering techniques** to ensure meaningful and coherent AI-generated story outputs.

- Evaluating different language models and assessing their performance in producing **creative, engaging, and genre-specific** comic stories.

- Exploring optimizations to replace GPT-2 with **LLaMA 3** for improved story coherence and structure.

Streamlit Development and Testing – Md Usman Ansari:

Md Usman Ansari was responsible for the **front-end development, UI/UX design, and overall system testing** using Streamlit. His contributions included:

- Designing the **interactive user interface** using streamlit with a visually appealing layout and sidebar options for user customization.
- Implementing **custom UI elements**, including **genre selection, comic style options, and AI-powered story generation buttons.**
- Writing **page configuration settings** to ensure a responsive design that adapts to different screen sizes.
- Managing **real-time AI execution** with st.spinner to provide users with visual feedback while the models generate content.
- Testing the entire system, debugging issues, and ensuring seamless **integration between LLaMA (story generation) and Stable Diffusion (image generation).**
- Implementing **ngrok tunneling** to expose the local Streamlit app on a public URL, making it accessible for remote testing.

6. LLM-Based Story Generation Challenges

The AI-powered story generation module in Comic Builder utilizes a large language model (LLM) to create structured narratives based on user input. While this automation significantly enhances creativity and efficiency, it also presents several challenges that must be addressed to ensure a coherent and engaging storytelling experience.

- **Consistency in Story Flow**: One of the primary challenges when using LLMs for multi-panel comics is maintaining a **logical progression of events** across different sections of the story. The AI model generates the **Introduction, Storyline, Climax, and Moral** separately, which can sometimes lead to:
- **Abrupt transitions** between panels if the AI fails to recognize continuity in the plot.
- **Inconsistent themes**, where the mood or tone of the story shifts unexpectedly.
- **Redundant or missing information**, making the story appear incomplete or disjointed.

- **Potential Solutions:**

- Implementing **structured prompts** that explicitly instruct the LLM to maintain a cause-effect relationship between different sections.

- Using **reinforcement learning** to fine-tune the model with a dataset of structured comic book stories.

- Experimenting with **memory-augmented techniques** to allow the model to recall earlier details while generating later sections.

Character Continuity:

For comics, maintaining consistent **character traits, personalities, and dialogues** across panels is crucial to ensuring a natural and immersive experience. However, LLMs often struggle with:

- **Character personality drift**, where a character's behavior or speech style changes unpredictably.

- **Inconsistent naming**, where characters introduced in the introduction might be called something different in later panels.

- **Dialogue mismatch**, where responses in later panels don't align with the character's earlier tone or intent.

Potential Solutions:

- Introducing a **character database** where key attributes (e.g., name, personality traits, catchphrases) are stored and referenced during text generation.

- Using **prompt chaining**, where the AI is reminded of previous character details before generating each new panel.

- Experimenting with **LoRA (Low-Rank Adaptation) fine-tuning** on structured datasets of character-driven comic books.

Short Context Window:

Most LLMs have a **limited context window**, meaning they can only consider a fixed amount of text at a time when generating responses. This limitation can result in:

- **Forgetting earlier story elements**, leading to plot holes or inconsistencies in later panels.
- **Repetitive storytelling**, where similar events are reintroduced because the model doesn't remember prior details.
- **Difficulty in handling long-form narratives**, restricting the ability to create extended multi-panel comics with rich world-building.

Potential Solutions:

- Utilizing **long-context models** (e.g., LLaMA 3 with extended context capabilities) to improve information retention.
- Implementing **sliding window attention** to enable the model to reference past text while generating new content.
- Storing previously generated content as **external memory** and feeding key details back into the model when generating subsequent story sections.

2. AI Image Generation Challenges (Stable Diffusion)

In the **Comic Builder** project, the AI-driven image generation system leverages **Stable Diffusion XL (SDXL)** to transform textual descriptions into visually appealing comic panels. This process involves generating high-quality, stylistically consistent illustrations that align with the **genre, character designs, and storytelling elements** of the AI-generated story. However, generating **consistent, coherent, and high-quality comic panels** comes with several challenges. Below are the key obstacles encountered and the solutions implemented to overcome them.

- **Manga/Comic Style Adaptation**: Stable Diffusion is a **general-purpose image generation model**, meaning it is not specifically trained for manga, anime, or Western-style comic illustrations. Adapting SDXL to generate **high-quality and stylistically consistent** comic images is a major challenge.

Issues:

- **Default SDXL outputs a generic style** instead of a distinct comic/manga aesthetic.
- **Lack of control over details like line thickness, shading, and halftone effects.**
- **Variability in artistic style** across different generations, making panels look inconsistent.
- **Difficulty in generating dynamic action scenes** that match comic storytelling conventions.

Proposed Solutions:

- **Fine-Tuning with LoRA (Low-Rank Adaptation):**
    - Training **LoRA models** specifically for different **comic styles** (e.g., manga, Western comics, cartoon).
    - Using specialized **datasets of high-quality comic art** to fine-tune the image generation model.
- **Pre-Trained Comic-Specific Models:**
    - Using fine-tuned versions of SDXL, such as:
        - **Anything V5, DreamBooth models (Anime/Comic-based) for manga styles.**
        - **OpenJourney (Midjourney-like) for Western comic aesthetics.**
- **Stylistic Prompt Engineering:**
    - Incorporating **detailed prompts** that specify art styles, e.g.:
        - “Manga-style, bold outlines, dynamic perspective, vivid shading, highly detailed.”
        - “Vintage comic book, halftone textures, ink shading, realistic lighting.”
- **Post-Processing Enhancements:**

- Using **image enhancement techniques** like:
    - **ESRGAN (Enhanced Super-Resolution GAN)** for better texture resolution.

- **Character Re-identification**:

A critical challenge in comic book creation is ensuring that **the same character appears consistently across multiple panels** without unintended variations in their appearance. Since **Stable Diffusion generates each image independently**, there is a risk of **character drift**, where the same character appears with:

- **Panel Composition**: Arranging images properly to fit comic-style panels.

Issues:

- Slightly different facial features, hair color, or outfits between panels.
- Inconsistent body proportions or poses, affecting continuity.
- Changes in expressions and emotions that do not align with the story.
- Difficulties in generating identical characters from different angles.

Proposed Solutions:

- **Face & Character Embedding Techniques:**
    - Using **DreamBooth** to train Stable Diffusion on a **specific character's face and design** for better consistency.
    - Utilizing **Textual Inversion** to create **personalized embeddings** for recurring characters.
- **Using ControlNet for Character Consistency:**
    - **ControlNet Depth Model** to preserve the depth and proportions of characters.

- **ControlNet OpenPose Model** to maintain the same pose across different panels.
- **ControlNet Scribble Mode** to manually sketch characters for better consistency.

- **Prompt Weighting & Negative Prompts:**
  - Using **prompt weighting** to emphasize character features, e.g.:
    - ((black hair, red eyes, golden armor, battle pose))
  - Implementing **negative prompts** to prevent unwanted variations, e.g.:
    - ((no face distortion, no outfit change, no extra accessories))

- **Reference-Based Image Generation:**
  - Using **IP-Adapter** (Image Prompt Adapter) to condition Stable Diffusion on **a reference image** to generate consistent characters.
  - Implementing **multi-frame latent consistency models** for improved coherence in sequential panels.

3. Panel Composition:

A comic book's effectiveness depends on how images are arranged into panels to create a visually engaging, readable, and structured narrative. Stable Diffusion generates individual images, but arranging them into a proper comic layout is an additional challenge.

Issues:

- **Image sizes and aspect ratios may vary, leading to misaligned panels.**
- **Fitting text within speech bubbles** without obstructing important visual elements.

- **Lack of control over panel framing** (e.g., some scenes might need close-ups while others need wide shots).

- **Maintaining visual storytelling flow across multiple images.**

Proposed Solutions:

- **Pre-Defined Comic Layout Templates:**
    - Implementing a **set of predefined panel layouts** (e.g., 2x2 grid, vertical strip, horizontal strip).
    - Using **automated image placement** functions to arrange images into structured panels.
- **Scene Composition Control with ControlNet:**
    - Using **ControlNet Depth Maps** to maintain consistent framing of objects in the scene.
    - Implementing **Perspective Control** to ensure a balanced mix of close-ups, mid-range shots, and wide shots.
- **Text-Aware Image Generation:**
    - Using **depth-to-image** models to leave blank spaces in the image for speech bubbles.
    - **Manually overlaying speech bubbles** post-generation for better readability.
- **Dynamic Panel Styling:**

- Allowing users to **choose between different comic styles** (e.g., dynamic action-based panels vs. structured grids).

- Applying **post-processing effects** (such as halftone patterns or ink outlines) to enhance comic authenticity.

5. Integration of Story & Images into a Web App

The Comic Builder project integrates LLM-based story generation and Stable Diffusion image synthesis into a seamless, interactive web application using Streamlit. The goal is to provide users with an intuitive and engaging platform that enables them to generate comic stories with AI-generated illustrations effortlessly.

However, integrating story generation, image synthesis, and layout design into a real-time web app introduces multiple challenges. Below is a detailed breakdown of the key aspects of Streamlit UI optimization and efficient workflow integration in the project.

5.1 Streamlit UI Optimization

Streamlit serves as the **frontend framework** for the Comic Builder app, providing an **interactive user interface (UI)** where users can input their story ideas, configure comic styles, and generate comic panels. Designing an **optimized UI** is crucial for a smooth user experience.

Challenges:

- **Real-time responsiveness:** Ensuring that the interface remains responsive while running **compute-heavy** LLM and Stable Diffusion processes.

- **User-friendly design:** Creating an intuitive layout for users with minimal AI knowledge.

- **Efficient sidebar controls:** Organizing settings like **genre selection, comic style, and layout format** efficiently.

- **Adaptive layouts:** Adjusting panel arrangements dynamically based on **user selections** (e.g., 2x2 grid vs. vertical strip).

Implementation Strategies:

- **Optimized Page Layout:**
    - Used st.set_page_config(layout="wide") to maximize screen space.
    - Structured the UI into **distinct sections**:
        - **Sidebar:** Genre, style, and format selection.
        - **Main Panel:** Story input and AI-generated outputs.
- **Real-Time Interactivity:**
    - Implemented **buttons and loading indicators** (`st.spinner`) to inform users about processing states.
    - Ensured **non-blocking execution** using Streamlit's **session caching (`st.cache_resource`)** to prevent redundant model loading.
    - Provided **text input fields** for users to **edit AI-generated stories manually.**

7. Training & Fine-Tuning Models

Training and fine-tuning AI models is a crucial aspect of improving **story generation (LLM) and image synthesis (Stable Diffusion)** for the **Comic Builder** project. This process involves **dataset curation, model optimization, and computational resource management** to ensure high-quality, stylistically accurate comic panels.

However, **fine-tuning models for comic-style generation presents multiple challenges**, including **gathering high-quality datasets, optimizing LoRA (Low-Rank Adaptation) models, and managing GPU resources efficiently.** Below is a detailed breakdown of the **training and fine-tuning process** for AI Comic Crafter.

Dataset Curation:

- To generate **consistent, high-quality comic images**, the AI model needs to learn from a **well-structured, high-resolution dataset** that captures:
- **Manga and Comic Styles** (e.g., Japanese manga, Western comics, digital illustrations)

- **Character Expressions & Poses** (to maintain continuity across multiple panels)
- **Background Elements** (to create immersive scenes)
- **Text and Speech Bubbles** (to integrate AI-generated dialogue with images)

Computational Cost:

Fine-tuning deep learning models for **comic generation** requires significant **computational resources**.

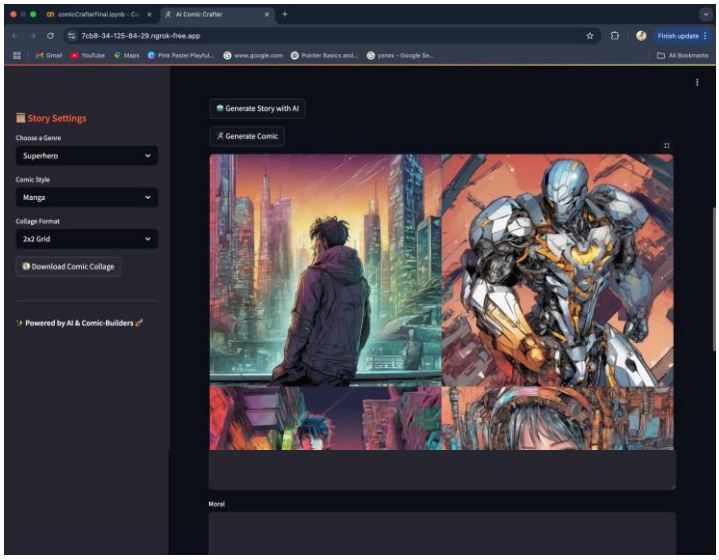
Challenges in Managing Computational Costs:

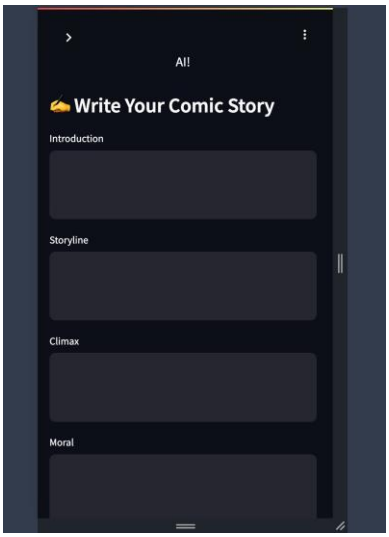
1. **High GPU Requirements:**
    - **Stable Diffusion LoRA fine-tuning requires GPUs with 24GB+ VRAM.**
    - Limited local resources make fine-tuning **costly and time-consuming.**
2. **Cloud Training Costs:**
    - Running **GPU-based cloud instances** (AWS, Google Colab Pro) can be expensive.

6. Results

The Comic Builders application has successfully automated the comic creation process. By combining LLaMA for story generation and Stable Diffusion for image generation, the platform provides an efficient and effective tool for transforming written stories into visually appealing comic strips.

Outputs :


AI Comic Crafter
7cb8-34-125-84-29.ngrok-free.app
Finish update
Story Settings
Choose a Genre
Superhero
Comic Style
Manga
Collage Format
2x2 Grid
Download Comic Collage
Powered by AI & Comic-Builders
Generate Story with AI
Generate Comic
Moral


AI!
Write Your Comic Story
Introduction
Storyline
Climax
Moral