Comic Crafter AI: Generative AI-Powered Comic Creation

By

Nandala Nithin (23951A66B8)

S. Pardhi Valli Rani (23951A66B9)

Harshini

Intel Unnati Industrial Training Program 2025

Project Mentor : M . Nagaraju

GitHub Repository Link:

https://github.com/23951A66B8/ComicCrafter-AI

Comic Crafter AI: Generative AI-Powered Comic Creation on Edge Devices

---

Overview:

Comic Crafter AI is a **generative AI-based comic generator** that runs **locally on edge devices**, enabling users to create **comic-style stories** based on **input prompts** without relying on cloud-based processing.

Problem Statement:

Traditional comic creation is time-consuming and requires both writing and artistic skills. With the rise of AI, there's an opportunity to automate comic generation by turning text prompts into fully illustrated, structured comic panels. However, most AI solutions rely on cloud models, which can introduce latency, privacy issues, and the need for continuous internet access.

Goal:

Create a web-based AI system (Comic Crafter AI) that can take a prompt and generate a four-panel illustrated comic strip with both story and images, then allow users to download it—all through local or API-based execution.

Solution Overview:

We built Comic Crafter AI, a full-stack application using:

- **Flask (Python)** for backend logic
- **Jinja2** templating for HTML rendering
- **Hugging Face & Stability AI APIs** for story and image generation
- **JS/HTML/CSS** for frontend interactivity
- **Pillow (PIL)** for image composition and download

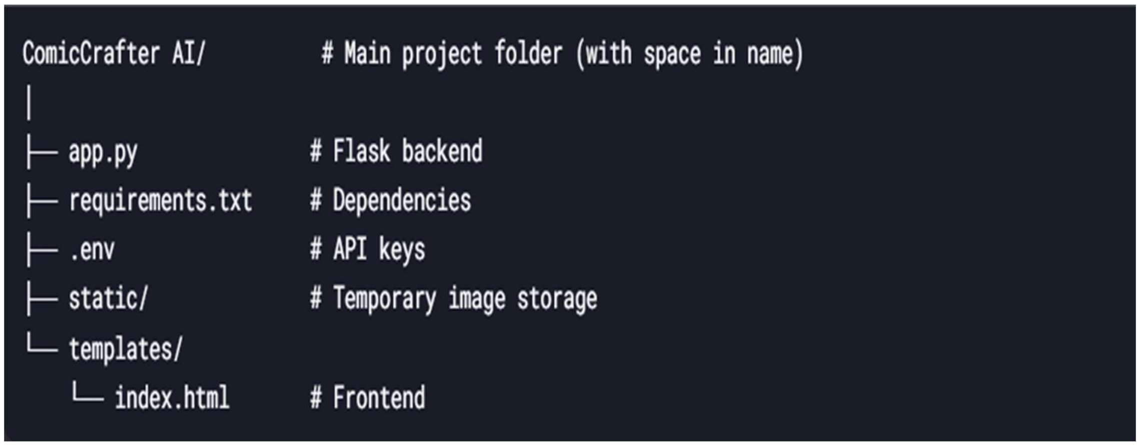
Each comic consists of 4 panels with:

- A short story snippet per panel
- A corresponding AI-generated image
- A final download button that merges all panels into one image

Key Features:

- User prompt-based comic generation
- AI-powered 4-panel story generation
- AI-generated cartoon-style image per panel
- No extra or pre-written panels
- Downloadable combined comic
- Responsive HTML frontend
- Uses APIs (no heavy models needed locally)

Project Directory Structure:



```
ComicCrafter AI/         # Main project folder (with space in name)
|
├── app.py              # Flask backend
├── requirements.txt    # Dependencies
├── .env                # API keys
├── static/             # Temporary image storage
└── templates/
    └── index.html      # Frontend
```

System Architecture:

[User Prompt]

↓

[Flask Backend]

↓

[Text-to-Story Generation (Hugging Face Falcon)]

↓

[Image Generation for each panel (Stable Diffusion XL API)]

↓

[Return panel data to frontend]

↓

[Display 4 panels: story + image]

↓

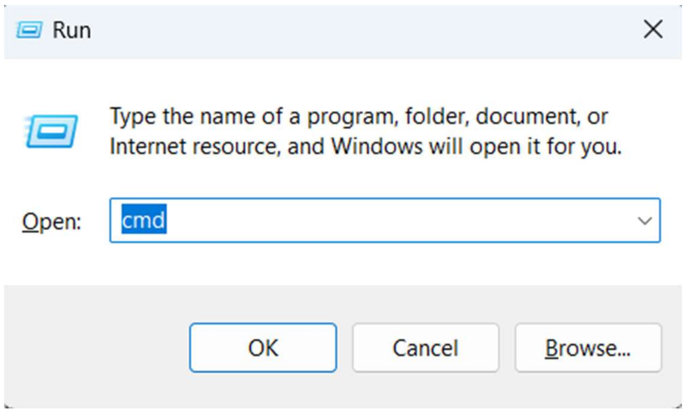
[Combine panels on “Download” → Return final image]

Technology Stack:

Component	-	Technology Used
Backend	-	Flask (Python), Requests, PIL
Frontend	-	HTML, CSS, JavaScript (Jinja2 templates)
Story Generation	-	Hugging Face API (Falcon-7B)
Image Generation	-	Stability AI API (SDXL)
Image Handling	-	Pillow (PIL), Base64
Template Engine	-	Jinja2

Steps to Run:

1. **Open CMD**



Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.

Open: cmd

2. **Navigate to the project folder**

```
cd path_to_your_folder/COMICCRAFTER AI
```

3. **Install Python dependencies**

```
pip install -r requirements.txt
```

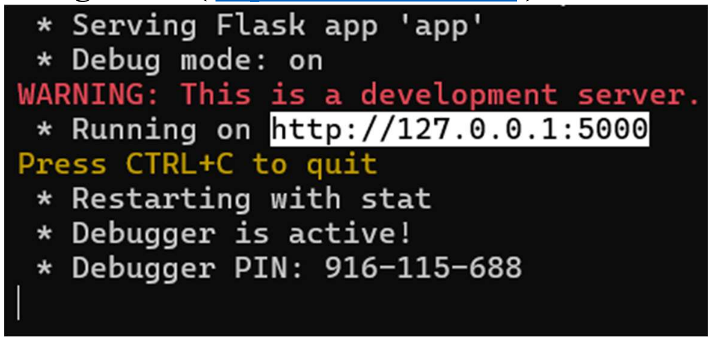
4. **Set Your API Keys (Mandatory)**
   Before running, set your API keys in your terminal:

```
set HF_API_KEY=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
set SD_API_KEY=sk-2RjZdARvjqiyr3qkUXacKCTGKKYnfzq1vawCA8NjrAHFoC5j
```

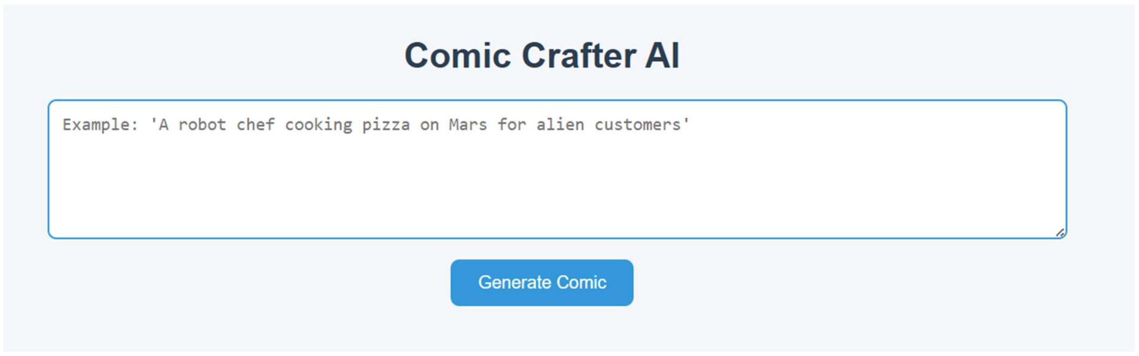
5. **Run the app:**

```
python app.py
```

6. **Navigate to: (http://127.0.0.1:5000)**



```
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 916-115-688
```



Comic Crafter AI
Example: 'A robot chef cooking pizza on Mars for alien customers'
Generate Comic

Sample Prompt:

"A robot joins a human cricket team and surprises everyone with its skills."

Result:

- **Panel 1**: Robot enters cricket ground (with image)
- **Panel 2**: Robot bowls to human batsman (with image)
- **Panel 3**: Robot fields and throws to wicketkeeper (with image)
- **Panel 4**: Robot celebrates with team (with image)

A download button combines these into one image.

Core Code Snippets:

Flask Route: Story and Image Generation

```python
@app.route('/generate', methods=['POST'])
def generate():
    ...
    # Get story panels via HuggingFace Falcon
    story = generate_story(prompt)

    # Get images via Stability API
    panel_data = []
    for i, panel_text in enumerate(story):
        image_url = generate_image(panel_text, panel=i+1)
        panel_data.append({"story": panel_text, "image_url": image_url})

    return jsonify({"panels": panel_data})
```

Frontend Template (index.html)

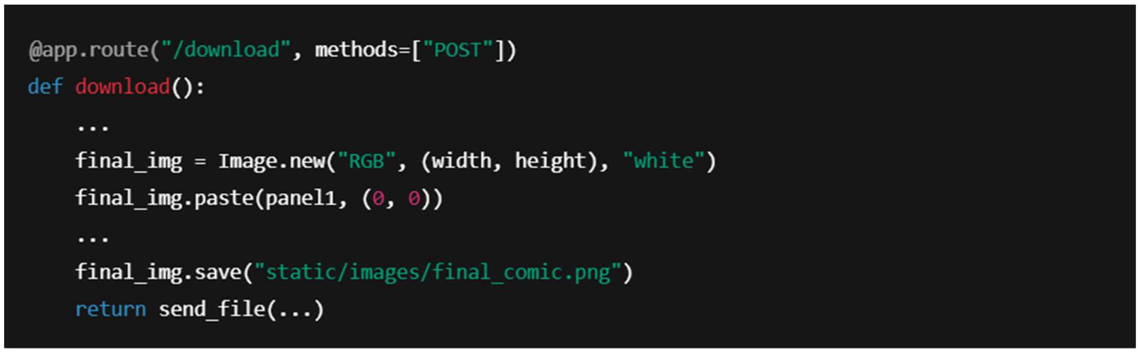
```html
<form id="generate-form">
    <input type="text" id="prompt" required>
    <button type="submit">Generate</button>
</form>

<div id="comic-panels">
    <!-- Dynamically loads 4 story panels with images -->
</div>

<button id="download-btn">Download Comic</button>
```

Comic Download Function

Uses Pillow to stitch the 4 panels (story + image) into one tall image and returns it as a downloadable PNG.



```
@app.route("/download", methods=["POST"])
def download():
    ...
    final_img = Image.new("RGB", (width, height), "white")
    final_img.paste(panel1, (0, 0))
    ...
    final_img.save("static/images/final_comic.png")
    return send_file(...)
```

Potential Future Improvements:

- Add multi-style comic rendering (manga, Disney, etc.)
- Enable character consistency across panels (LoRA training)
- Add speech bubble overlays via NLP+OpenCV
- Deploy as PWA (mobile app)
- Enable comic saving as PDF
- Add comic title/author fields

Conclusion:

Comic Crafter AI transforms your prompt into a 4-panel comic using real-time API-based AI models. The application is lightweight, privacy-friendly, and doesn't rely on local model downloads. It gives users a smooth and fun experience creating unique, illustrated comics—all in one click.