Software Requirements Specification

for

Comic Insights: Extraction & Generation

Version <1.0>

Prepared by

Group Name: Comic Insights team

Ponnam Adithya Sai	**se22ucse209**	**se22ucse209@mahindrauniversity.edu.in**
Ravi Sankar	**se22ucse106**	**se22ucse106@mahindrauniversity.edu.in**
Pidatala Vardhan	**se22ucse207**	**se22ucse207@mahindrauniversity.edu.in**
Dhruva Reddy	**se22ucse082**	**se22ucse082@mahindrauniversity.edu.in**

Instructor: Dr. Vijay Rao

Course: Software engineering

Lab Section: Monday (10:30 AM - 12:30PM)

Teaching Assistant: *Nartakannai*

Date: 10/03/2025

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
1.0	Pidatala Vardhan	Complete document	11/03/25

1 Introduction

1.1. Document Purpose

This Software Requirements Specification (SRS) document outlines the functional and non-functional requirements for the **Comic Insights AI System – Phase 2**, Release Version 2.0. The system is designed to automate the generation of comic book content using artificial intelligence, covering story input, character management, scene generation, and image rendering via a Gradio-based user interface. The project integrates local large language models (e.g., Gemma 12B via Ollama) for plot summarization and scene breakdown, as well as Stable Diffusion-compatible models enhanced with LoRAs, IP Adapters, and VAEs for generating comic-style illustrations.

This document serves as the primary reference for developers, testers, and stakeholders involved in Phase 2, which specifically focuses on the **scene manager module and UI overhaul**, including functionality for scene editing, character-driven prompt injection, iterative image refinement, and finalized comic compilation. It does not cover earlier OCR pipelines or future cloud deployment strategies, which are planned for later phases.

1.2. Product Scope

The **Comic Insights AI System** is an intelligent comic creation platform that leverages local AI models to convert narrative input into fully illustrated comic pages. In its current phase (Phase 2), the system focuses on a modular, interactive interface that guides users from story summarization and character management to scene-wise generation of visuals and dialogue. Key features include: tabbed workflows for story input and editing, integration of character profiles with booru-style visual tags for prompt injection, and an iterative editing pipeline that allows users to refine each scene's text and artwork before confirmation. The final output is a complete comic book compiled from confirmed scenes and images.

The product aims to significantly reduce the time, technical skill, and creative effort required to produce high-quality, stylized comics. By integrating NLP and generative models in a modular UI, the system benefits writers, illustrators, and hobbyists by providing an end-to-end creative assistant. It empowers creators with flexibility and creative control while ensuring visual and narrative consistency across scenes, characters, and plotlines.

1.3. Intended Audience and Document Overview

➔ **Developers:** Implement and refine features.

- ➔ **Project Managers:** Monitor scope and progress.
- ➔ **Testers:** Validate system functionalities.
- ➔ **End Users:** Artists and creators using AI-generated content.

This document includes system architecture, functional requirements, constraints, and use case models.

1.4. Definitions, Acronyms and Abbreviations

TERM	DEFINITION
API	Application Programming Interface
Booru	Tag-based image dataset format used for visual prompting
CRUD	Create, Read, Update, Delete
DB	Database
Gradio	Python library for building web UIs for ML applications
IP Adapter	Image prompt adapter for cross-modal guidance in generation
JSON	JavaScript Object Notation
LLM	Large Language Model
LoRA	Low-Rank Adaptation, used for fine-tuning diffusion models
NLP	Natural Language Processing
SD	Stable Diffusion
UI	User Interface
UX	User Experience
VAE	Variational Autoencoder

1.5. Document Conventions

The formatting conventions used include:

Font Style: Arial

Font Size: 12pt for body text, 14pt for section titles

Text Formatting: Important terms and section headings are **bolded** for emphasis.

Spacing: Single-spaced text with **1-inch margins** on all sides.

Section & Subsection Titles: Follow the provided template structure for consistency.

Italics: Used for comments or notes where applicable.

1.6. References and Acknowledgments

Gradio Documentation
https://www.gradio.app

Stable Diffusion & Hugging Face Diffusers
https://huggingface.co/docs/diffusers/index

Ollama – Local LLM Serving
https://ollama.com

Gemma LLM by Google
https://ai.google.dev/gemma

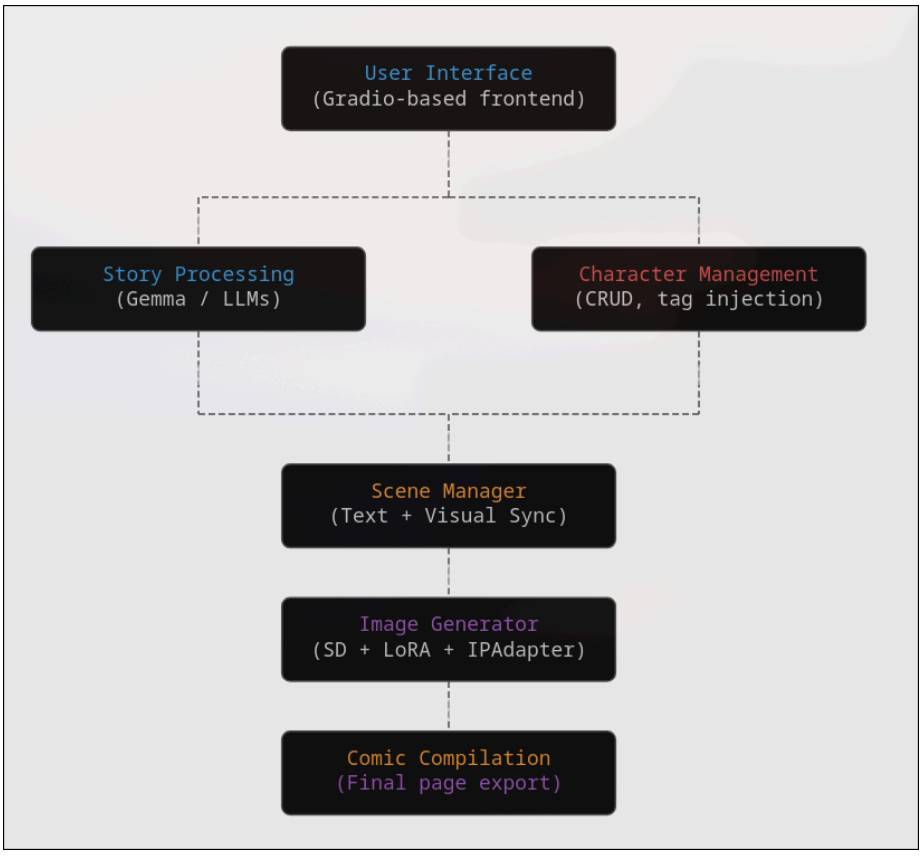
Tesseract OCR Documentation
https://github.com/tesseract-ocr/tesseract

2 Overall Description

2.1. Product Overview

The Comic Insights AI System is a **self-contained, modular application** designed to automate comic creation using AI models for text understanding and image generation. It builds upon internal learnings from an earlier prototype (Phase 1), which focused on OCR and static image rendering. Phase 2 introduces significant upgrades including **plot-to-scene breakdown**, **character-aware prompting**, **iterative editing**, and **final compilation**, structured through a **multi-tabbed Gradio interface**. It uses **local LLMs (e.g., Gemma)** for story processing and **Stable Diffusion-based models enhanced with LoRA, VAE, and IP Adapter** for generating comic-style artwork.

The product functions as a bridge between **human creativity and automated rendering**. The user provides an overarching story, which is broken into scenes and managed via a UI that tracks character involvement and context. Each scene is iteratively refined in both text and visuals before being finalized. Confirmed outputs are compiled to form a cohesive digital comic. While the current version is standalone, future versions may support collaborative workflows, cloud rendering, or web publishing.



2.2. Product Functionality

The Comic Insights AI System will support the following major functions:

- **Story Input & Summarization**
  Users can input a long-form plot which is summarized and split into individual scenes using a local LLM (Gemma).

- **Character Management**
  Users can create, update, and delete characters. Each character includes roles, descriptions, and booru-style tags for visual prompting.

- **Scene Breakdown & Editing**
  The system breaks the story into scene-by-scene segments, allowing users to edit and finalize each scene's text.

- **Prompt Injection for Visuals**
  Character data and scene descriptions are converted into detailed prompts for generating comic-style images.

- **Image Generation (Iterative)**
  Users can preview, accept, or regenerate scene images using SD-compatible models with LoRA, VAE, and IP Adapter support.

- **Confirmation & Compilation**
  Finalized scene texts and images are stored and compiled into a complete digital comic export.

- **Tabbed Modular Interface**
  A clean, intuitive Gradio UI allows navigation between different functional modules: Story, Characters, Scenes, Images, and Final Output.

2.3. Design and Implementation Constraints

- **COMET Method & UML Modeli**ng
  The software design must follow the COMET (Collaborative Object Modeling and Architectural Design Method) for structuring components and interactions. UML diagrams will be used to represent use cases, class structures, and workflows.

    - Reference (COMET): Hassan Gomaa, Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures,

Addison-Wesley, 2004.

- Reference (UML): Object Management Group (OMG), Unified Modeling Language (UML) Specification, Version 2.5.1.

- **Hardware Constraints**
  The system must be able to run on a local machine with a GPU (preferably with ≥12GB VRAM), as all AI models (LLM and SD) are executed without cloud APIs. RAM usage should also remain under 16 GB for basic operations.

- **Technology Stack**

  ➔ Frontend: Gradio-based tabbed UI

  ➔ Backend: Python 3.10+

  ➔ AI Models: Local LLMs (Gemma via Ollama), Stable Diffusion (with LoRA, IP Adapter, VAE support)

  ➔ Prompt Format: Booru-style prompts for image generation

  ➔ Data Format: JSON for character and scene storage

- **Modularity**
  All components should be modular and independently testable. For example, character management must operate separately from image generation.

- **Security Considerations**
  As the system deals with user-generated content, appropriate file sanitization and session isolation should be implemented to prevent unwanted access or data leakage.

- **Maintainability**
  Codebase must follow PEP8 standards for Python and be annotated with docstrings. External dependencies should be documented in a `requirements.txt` file.

- **Offline Operation**
  The system must function entirely offline without dependence on external APIs (e.g., OpenAI, Replicate), ensuring local privacy and control.

2.4. Assumptions and Dependencies

- It is **assumed that the end-user has access to a local machine** with sufficient compute power (e.g., an NVIDIA GPU with ≥12GB VRAM) to run local LLMs and Stable Diffusion models effectively.

- The project **depends on pre-trained models** (e.g., Gemma for LLM tasks, and SD XL checkpoints with LoRA/IPAdapter support) being available and compatible with the chosen inference tools.

- It is assumed that **Ollama or an equivalent local LLM runtime** is properly set up and operational during development and deployment.

- The system assumes that **all user inputs (e.g., text, character descriptions) are safe and valid**, and no malicious or malformed data will be intentionally introduced.

- The application **relies on Gradio** as the UI framework, and it is assumed that future updates to Gradio will not break current tabbed layout functionalities.

- The system assumes **no internet connectivity** during generation tasks, meaning all inference must work fully offline using local resources.

- Dependencies include **Python libraries** such as Hugging Face Transformers, Diffusers, Gradio, and custom scripts built during Phase 1, which must remain stable and accessible.

3 Specific Requirements

3.1. External Interface Requirements

3.1.1. User Interfaces

The Comic Insights system uses a **multi-tabbed Gradio web interface** that runs locally in a browser. Users interact with the system through **buttons, dropdowns, text fields, image previews, and confirmation actions**. The interface is designed to be **modular**, with each tab corresponding to a major stage of the workflow.

UI Tabs and Interactions:

- **Tab 1: Story Input** – Users enter or upload a long-form narrative. On clicking "Summarize," the system extracts a scene-wise structure using the local LLM.
- **Tab 2: Character Management** – Users can create, update, or delete characters. Each character includes a name, role, description, and visual tags.
- **Tab 3: Scene Editor** – Displays generated scene text. Users can edit, regenerate, or approve each scene description.
- **Tab 4: Image Generator** – Displays scene-based image generated using Stable Diffusion with prompt injection. Users can regenerate or confirm the image.
- **Tab 5: Final Output** – Shows confirmed scenes and images in sequence. Allows export as a comic PDF or image set.

User Input Method:

- **Interaction Type**: Point-and-click via mouse or touchpad
- **Input Fields**: Textboxes (for story, characters), Dropdowns (for selecting roles), Buttons (for generate, regenerate, confirm)
- **Output Views**: Scene text viewer, Image preview component, Final comic compilation window

3.1.2. Hardware Interfaces

The Comic Insights system is designed for **local deployment on personal computers or workstations** and does not interact directly with external hardware sensors. However, it has the following hardware-related interfaces:

- **GPU Interface**
  The software communicates with the local **NVIDIA GPU** (e.g., via CUDA or ROCm) to accelerate inference for image generation (Stable Diffusion) and LLM tasks (Gemma via Ollama).
  - *Function*: Enables fast image synthesis and natural language processing.
  - *Requirement*: Minimum 12GB VRAM recommended.
- **CPU Interface**
  Handles lightweight operations such as session control, UI rendering (via Gradio), and file I/O.
- **Storage Device**
  Used to store character JSON files, generated images, user stories, and final comic compilations.
- **Monitor & Input Devices**
  Standard hardware for interaction — **mouse/trackpad, keyboard, and monitor** are required for navigating the UI and inputting data.

3.1.3. Software Interfaces

- **Local LLM Runtime (Ollama)**
  The application sends prompts and receives responses via a **local API** from Ollama running the Gemma 12B model.
  - *Interface Type*: Localhost-based API
  - *Function*: Processes story summarization and scene generation.
- **Stable Diffusion Model Interface**
  The backend interacts with **Diffusers or custom inference scripts** for generating images based on booru-style prompts. LoRA, VAE, and IP Adapter modules are integrated during runtime.
- **Gradio Interface Layer**
  The Gradio web server is the main interaction point between the user and all backend components. Each tab uses Python backend logic tied to specific AI tasks.

- **Mobile/Web Client** *(Future Phase)*
  The current version does **not include** a mobile interface, but future iterations may expose REST endpoints or WebSockets for remote interaction.

3.2 Functional Requirements

FR1: Story Input and Summarization

- **FR1.1**: The system shall allow the user to input a long-form story as plain text.
- **FR1.2**: The system shall summarize the story using a local LLM (Gemma via Ollama).
- **FR1.3**: The system shall divide the story into scene-wise segments.

FR2: Character Management

- **FR2.1**: The system shall allow users to create new characters with fields for name, role, description, and booru-style tags.
- **FR2.2**: The system shall support editing and deletion of existing characters.
- **FR2.3**: The system shall inject character-specific visual tags into image generation prompts.

FR3: Scene Editing and Confirmation

- **FR3.1**: The system shall display auto-generated scene text to the user.
- **FR3.2**: The system shall allow users to edit the scene text before proceeding.
- **FR3.3**: The system shall mark a scene as “confirmed” once finalized.

FR4: Image Generation

- **FR4.1**: The system shall generate an image using Stable Diffusion based on the scene description and character tags.
- **FR4.2**: The system shall allow users to regenerate the image if unsatisfied.
- **FR4.3**: The system shall allow the user to confirm the final image.

FR5: Comic Compilation and Export

- **FR5.1**: The system shall compile all confirmed scenes and images in sequential order.

- **FR5.2**: The system shall allow the export of the final comic as a PDF or image bundle

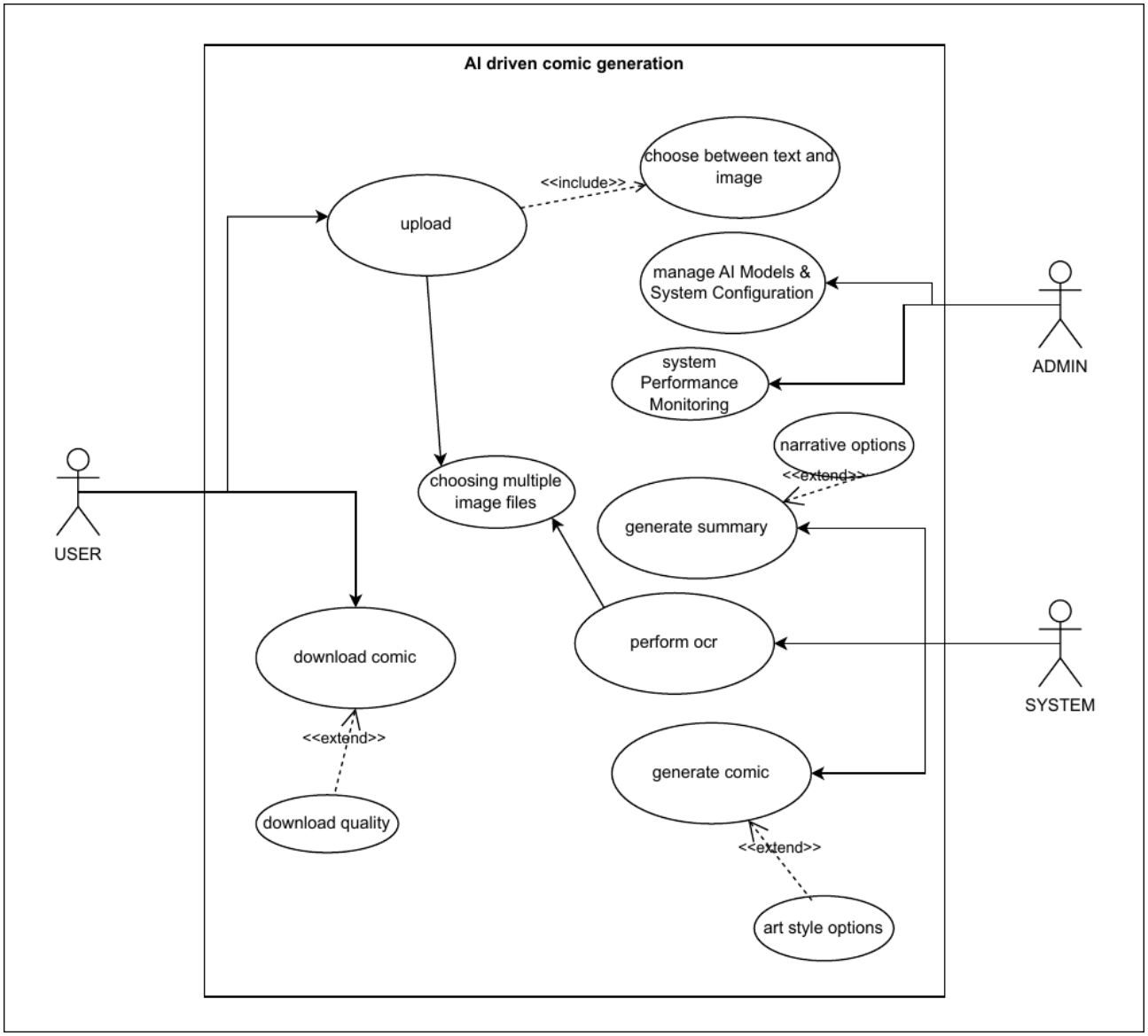
FR6: User Interface Functions

- **FR6.1**: The system shall provide a tabbed interface with dedicated sections for each workflow stage.

- **FR6.2**: The interface shall allow navigation between tabs without losing session data.

- **FR6.3**: The interface shall show visual feedback (e.g., loading bars, image previews) for user actions

3.3. Use Case Model



Use Case #1

Author – Pidatala Vardhan

Purpose - Abstraction of the functions and users that encapsulate the project.

Requirements Traceability -

FR1.1 – User can upload story/image

FR1.2 – System summarizes story

FR4.1 – System generates comic images

FR5.1–5.2 – Comic is compiled and downloaded

FR3.1–FR3.3 – Scene and narrative editing

FR2.1–FR2.3 – Character and tag configuration

FR6.1–6.3 – UI tab interactions

Non-functional: Offline AI model inference, system monitoring

Priority - High

This use case represents the **core functionality** of the system. It is essential for user interaction, comic generation, and final output. Failure would render the platform non-functional.

Preconditions -

- User must have access to the application (locally or via local server).
- Required AI models (LLM and SD) must be installed and operational.
- Admin must have preconfigured system and model parame

Actors – User, Admin, System

Extends –

- `generate comic` → **extends** `art style options`
- `download comic` → **extends** `download quality`
- `generate summary` → **extends** `narrative option`

Flow of Events

1. Basic Flow

- User uploads content (text or image).
- System performs OCR if needed.
- System summarizes text into scenes.
- System generates images for scenes using AI.
- User views and downloads final comic.

2. Alternative Flow

- User chooses to input text manually instead of uploading.
- Admin adjusts model parameters before generation.
- User customizes art style before image generation

3. Exceptions

- Model inference fails due to hardware or configuration issues.
- Uploaded file is invalid or corrupted.
- System times out or runs out of memory during generation.

4 Other Non-functional Requirements

4.1. Performance Requirements

P1. Scene summarization using the local LLM (Gemma) should complete within 15 seconds for an average story input of 500–1000 words.

Rationale: Ensures responsiveness and smooth user experience in the first step of the workflow.

P2. Each image generation operation must complete within 20–25 seconds on a machine with a GPU (12GB+ VRAM).

Rationale: Comic generation is an iterative process—low wait times improve user productivity and system usability.

P3. The UI must respond to user inputs (clicks, text entry, tab changes) within 1 second under normal system load.

Rationale: Maintains smooth interaction in a Gradio-based local web interface.

P4. The full comic compilation and export (PDF or image set) must complete within 10 seconds after the user confirms the final scene.

Rationale: Ensures end-to-end efficiency for batch export of finalized scenes and images.

P5. The system should remain operational and stable for input projects up to 50 scenes or 100MB in total assets, without crashes or degradation.

Rationale: Supports realistic comic book volumes without requiring external scaling or cloud infrastructure

4.2. Safety and Security Requirements

S1. Local-only Execution Requirement

All AI models and user data must remain on the local machine; the system must **not transmit any data to external servers or APIs**.
Rationale: Ensures privacy and protects intellectual property of user-generated content.

S2. Secure File Handling
The system must sanitize all uploaded files (text or image) to prevent **code injection, path traversal, or file corruption attacks**.
Safeguard: File types must be restricted to `.txt`, `.json`, and common image formats (`.png`, `.jpg`).

S3. Session Isolation
Each user session must be independent, with **temporary caches or intermediary files cleared after session ends**.
Rationale: Prevents data leakage between comic generation sessions.

S4. User Input Validation
Inputs must be validated to prevent **prompt injection attacks** on the LLM or malformed prompt crashes during image generation.
Safeguard: Escape sequences, excessive repetition, or harmful content must be filtered.

S5. Admin-only Configuration Access
All access to AI model configuration, performance tuning, and advanced debug controls must be **restricted to admin users only**.

S6. Mobile/Web Access Security *(For future versions)*
If a mobile or remote interface is added, it must include:

- **OAuth2-based user authentication**
- **End-to-end encryption (HTTPS)**
- **Rate-limiting to prevent misuse of image generation APIs**

S7. Compliance and Policy
The system should align with basic **data protection principles under GDPR**, ensuring that:

- No personal data is stored without consent
- Users can delete their generated data

- Logs, if any, are anonymize

4.3. Software Quality Attributes

4.3.1 Maintainability

To ensure long-term maintainability, the system has been designed with:

- **Modular architecture**: Backend components like story processing, character management, and image generation are isolated Python modules with clearly defined interfaces.
- **Separation of Concerns**: Each Gradio tab handles a distinct function. UI logic is separated from AI model logic.
- **Consistent coding standards** (PEP8): Code is well-commented, with docstrings and inline explanations for future developers.
- **Loose coupling**: Components interact through shared JSON structures or API-like function calls, minimizing interdependencies.

Goal*:* New modules (e.g., different image models or text processors) should be integrable with minimal changes to the rest of the codebase.

4.3.2 Extensibility (Adaptability/Design for Change)

To accommodate evolving requirements:

- The system supports **plug-and-play model replacement**, such as swapping Gemma with another local LLM or changing the Stable Diffusion variant used for image generation.
- **Prompting logic and character metadata** are driven by configurable JSON, allowing fast customization.
- Scene chunking and comic export are being developed as **separate modules**, which can be extended or replaced independently.

Example: In future versions, a new "Web Comic Exporter" can be added without altering the rest of the pipeline.

4.3.3 Usability

The UI is designed with **simplicity and clarity** in mind:

- A **multi-tabbed Gradio interface** breaks complex workflows into manageable steps.
- Users can **confirm, edit, or regenerate** outputs at each stage, maintaining control.
- Real-time feedback is provided during generation and summarization, with status indicators or previews.
- Default values and tooltips are included for non-technical users.

Goal: Enable users to create a basic comic page with no prior experience in prompting or model configuration within 10–15 minutes.

4.3.4 Reliability

Reliability is achieved through:

- **Local-first design**: All LLM and image generation happens offline, eliminating network-related failures.
- **Error handling and input validation**: Incorrect prompts, corrupt files, or misconfigured JSONs are flagged with descriptive messages.
- **Session management**: Intermediate data is auto-saved across tabs, reducing loss from accidental page refreshes or failures.

Metric: System must handle at least 20 consecutive scene generations without crashing or memory overload on a standard 16GB RAM system with a 12GB VRAM GPU.