Software Design Specifications

for

<Comic Insights>

Prepared by:

Comic Insights Team

Document Information

Title: Comic Insights – Software Design Specification	
Project Manager: Prof Vijay Rao	**Document Version No:** 1.0
	Document Version Date: 9th April 2025
Prepared By: Comic Insights Team: Ponnam Adithya Sai, Vardhan Pidatala, Ravi Sankar Hoskote, Dhruva G	**Preparation Date:** 10th April 2025

Version History

Ver. No.	Ver. Date	Revised By	Description	Filename : SDD_CI
1.0	10-04-2025	Comics Insight Team	An AI application for image summary and generation	

Table Of Contents

1 Introduction

This Software Design Document (SDD) outlines the architectural and component-level design of Comic Insights, an AI-driven application aimed at automating the comic creation process through text-to-image synthesis and natural language understanding. Following the successful completion of Phase 1—focused on OCR-based extraction and NLP-driven summarization of comic panels—Phase 2 introduces a system for generating comic pages dynamically from user-provided text prompts.

The current implementation leverages a Gradio-based web interface for rapid prototyping and user interaction, supported by modular backend components including OCR (Tesseract), transformer-based summarization models, and diffusion-based image generation models such as Stable Diffusion. The system is optimized for local development environments, with support for GPU acceleration and optional access to college-provided HPC infrastructure for resource-intensive tasks.

While the present focus is on a minimal, page-by-page generation workflow, the underlying design is forward-compatible with potential pivots—such as migrating to a microservices-based architecture or integrating persistent user data and community features in future phases.

1.1 Purpose

The purpose of this Software Design Document (SDD) is to formally outline the architectural and technical design of the Comic Insights application. It serves as a bridge between the Software Requirements Specification (SRS) and the implementation phase by translating functional and non-functional requirements into actionable design components, data flows, and interaction models. This document ensures that developers, designers, testers, and stakeholders have a unified understanding of how the system is structured and how its core features—comic content extraction and comic generation—are technically realized.

The document is organized into sections covering high-level architecture, component design, data handling, interface design, and supporting diagrams, ensuring clarity and traceability throughout the development lifecycle.

This document is intended for the following audiences:

- Developers
- UI/UX Designers
- Testers
- Professors and reviewers

1.2 Scope

This Software Design Document defines the scope of Comic Insights – Phase 2, focusing on the development of a minimal, functional system for AI-driven comic page generation using text prompts.

Core Deliverables for Comic Insights - Phase 2

- A Gradio-based interface for page-by-page comic generation
- Integration of a text-to-image model (e.g., Stable Diffusion) to generate comic-styled visuals
- Support for basic prompt refinement through iterative feedback
- Stateless session management for local deployment
- Export options for generated images in standard formats (e.g., PNG)

Scope

The scope is limited to local, single-user operation without persistent data storage, user authentication, or cloud hosting. Future features like storyline automation, full-stack architecture, or community sharing are acknowledged but out of scope for this phase.

This document serves to define the boundaries of implementation and guide the technical execution of these deliverables.

1.3 Definitions, Acronyms, and Abbreviations

TERM	**Definition**
OCR	Optical Character Recognition — Converts images of text (e.g., comic speech bubbles) into machine-readable text using tools like Tesseract.
NLP	Natural Language Processing — AI techniques for understanding and generating language. Used to extract and summarize comic dialogues.
RAG	Retrieval-Augmented Generation — Enhances generative models by retrieving relevant contextual data before output generation. Useful for plot consistency.
Tesseract	An open-source OCR engine used for extracting text from comic panels. Integrated via pytesseract.

Gradio	A Python-based framework for rapidly building web UIs for ML models; used for prototyping the Comic Insights interface.
CLIP Guidance	A mechanism where the CLIP model scores how well generated images match text prompts, steering generation toward semantically aligned outputs.
ControlNet	A conditional extension to diffusion models allowing generation to be guided using edge maps, poses, or depth — useful for structured panel control.
Negative Prompting	A prompt engineering technique where undesirable traits (e.g., 'blurry', 'text overlays') are specified to be excluded from image generation.
Prompt Augmentation	Enhancing base prompts with predefined keywords or tags to improve consistency in mood, style, or structure across comic panels.
Scheduler (Diffusion)	Controls the denoising steps during diffusion; different schedulers (e.g., DDIM, PNDM) influence image speed, detail, and coherence.
LoRA (Low-Rank Adaptation)	A lightweight fine-tuning method that modifies specific layers of a pre-trained model to adapt it to niche domains like comic art styles.
Transcript	Structured representation of extracted text from comic images, used for summaries and archiving.
Prompt Chaining	Feeding outputs (text or image) from previous steps into future prompts to ensure narrative continuity across comic pages.
Session Context	A runtime data structure that tracks the current user’s task, uploaded content, prompt history, and output iterations.
Celery	A distributed task queue used to run long-running operations (e.g., image generation, export) in the background without blocking the UI.
Firebase	A cloud platform used for managing user authentication (Auth), storing user project data (Firestore), and hosting assets (Cloud Storage).
Stable Diffusion	A latent diffusion model that generates images from text prompts with high visual fidelity and control.

2 Use Case View

This section outlines the key use cases that represent the core functionality and design focus of the **Comic Insights** application. Each use case captures a significant user interaction flow and demonstrates how various system components—such as the prompt processor, text generation module, image synthesis pipeline, and session manager—collaborate to deliver an end-to-end experience.

The selected use cases are central to the system's behavior and reflect design-critical paths involving AI model orchestration, user feedback integration, and export mechanisms. They serve as a bridge between functional requirements and technical implementation, ensuring traceability and coherence throughout the development lifecycle.

For each use case, we provide:

- A brief description
- Actor roles and interaction steps
- Edge cases or alternate flows
- Corresponding diagrams where applicable

These use cases inform UI design, backend modularity, error handling strategies, and model interaction patterns, and are integral to both development and testing phases.

2.1 Use Case

Use Case 1: Generate Comic Page from Prompt (Text → Confirm → Image)

Field Details

- Use Case Name: Generate Comic Page from Prompt
- Actor(s): User
- Preconditions:
    - User has launched the Comic Insights interface
    - Required models and configurations are loaded
    - No prior comic session is required (can start fresh)

Main Flow

1. User selects "Start New Comic"

2. System presents a structured prompt form with fields for:
    - Setting
    - Key locations
    - Main characters
    - Timeline
    - Genre
    - Art style (dropdown)
    - Overall vibe
3. Optional: User may enter a "context" block (within token limits) which can be:
    - A full comic summary
    - A central plot event
    - A preceding scene
4. System combines the prompt + context and generates a proposed plot (text) for Page 1
5. User reviews and confirms or edits the plot
6. Upon confirmation, the system proceeds to generate a corresponding image panel
7. Generated image is displayed alongside editing tools for:
    - Prompt
    - Negative prompt
    - CFG scale (creativity/variation slider)
8. User may accept, modify, or regenerate the image as many times as needed
9. All iterations are saved in-session and can be accessed for comparison or reversion

Alternate Flows

- If the prompt is vague or incomplete, dynamic suggestions or auto-fill hints are displayed
- If the user skips the context block, the system proceeds using only the structured prompt

Exceptions

- If text generation fails (e.g., invalid context), fallback messages are shown with edit suggestions
- If image generation fails, the system prompts retry with adjusted inputs

Postconditions

- The user completes Page 1 with a confirmed text plot and a corresponding image
- Both text and visual versions are stored in the active session for further editing, export, or continuation

Use Case 2: Regenerate Panel Using Feedback (Text & Image Iteration)

Field Details

- Use Case Name: Regenerate Comic Panel Using Feedback
- Actor(s): User
- Preconditions:
    - A comic page (with both plot and image) has already been generated
    - The session is active and state is maintained

Main Flow

1. User initiates feedback-based refinement from the UI on an existing comic page

Textual Feedback Stage:

2. The system presents the generated plot in an editable text field
3. The user may either:
    - Directly edit the plot text (e.g., fix tone, pacing, character actions)
    - Provide abstract feedback (e.g., “make this more suspenseful”)
4. System refines the updated or annotated plot to ensure narrative coherence, character consistency, and language clarity

Visual Feedback Stage:

5. Once the revised plot is confirmed, the system proceeds to regenerate the visual panel
6. The UI provides a simple editing GUI including:
    - Prompt
    - Negative Prompt
    - CFG Scale slider (controls adherence to prompt vs. creativity)
7. User modifies the inputs and triggers re-generation
8. The system generates a new image panel using the refined text context and updated visual prompt settings

Iteration Management:

9. Each version (text + image) is saved with a timestamp and version ID
10. Users can browse a gallery view of all previous iterations, revisit, or restore any version at any time

Alternate Flows

- If user feedback is ambiguous or conflicting, the system may prompt for clarification or suggest prompt reformulations
- If text editing is skipped, the system assumes the last confirmed plot version

Exceptions

- If the text-to-image model fails to render an image, the UI displays a fallback image and invites the user to retry with modified CFG or prompts
- If plot refinement fails to parse or clean the user's edits, suggestions are provided for better formatting

Postconditions

- A new panel (text + image) is generated and logged
- The session grows into a versioned storyline that supports creative experimentation without data loss

Use Case 3: Export Comic Page

Field Details

- Use Case Name: Export Comic Page
- Actor(s): User
- Preconditions:
    - At least one comic page (plot + image) has been finalized in the current session

Main Flow

1. User selects the “Export” option from the session toolbar or end-of-page panel
2. System displays export options including:
    - Image (PNG)
    - Plot Text (TXT / JSON)
    - Combined (ZIP: image + text)
3. User selects desired format and filename (default auto-named with timestamp)
4. System packages the file(s) and triggers browser download or saves to Firebase Storage (if

cloud mode is enabled)

Alternate Flows

- If the user selects “Export All,” the system compiles all confirmed panels into a downloadable ZIP archive or a PDF (if implemented)
- If the session contains incomplete panels, the UI notifies the user to review or skip

Exceptions

- If file packaging fails, the UI prompts the user to retry and logs the error
- For unsupported file formats, fallback to default export (PNG/TXT) is triggered

Postconditions

- The user receives a downloadable file containing one or more finalized comic pages with both visual and narrative content

UI Notes

- Export button should be visible only after a page is confirmed
- Download trigger should work offline (local mode) and via Firebase (hosted mode)

3. Design Overview

The Comic Insights system is designed as a modular, AI-powered platform that enables both comic understanding (Phase 1) and comic generation (Phase 2). Initially focused on summarizing comic panels using OCR and NLP, the project has evolved to support dynamic, page-by-page comic creation using natural language prompts and diffusion-based image synthesis.

The system follows a clean, beginner-friendly Model-View-Controller (MVC) inspired architecture, with loosely coupled modules for user interaction, text processing, image generation, and session tracking. While early development prioritizes local execution and simplicity, the architecture allows for future enhancements including persistent storage, cloud-based deployment, and advanced creative tools.3.1 Design Goals and ConstraintsDesign Goals

- **User-Centric Simplicity:** Minimal, intuitive UI suitable for non-technical users to summarize existing comics or create new ones from scratch.
- **Iterative Interactivity:** Support for continuous refinement through user feedback—on

both text and image—enabling a flexible, storytelling-first workflow.

- **Coherent Summarization:** NLP modules generate accurate and context-aware summaries from extracted text in Phase 1.
- **Comic-Styled Image Generation:** Visual output reflects the comic medium with consistent characters, stylistic fidelity, and outlined art.

3.1 Design Goals and Constraints

Design Goals

- **User-Centric Simplicity**: Minimal, intuitive UI suitable for non-technical users to summarize existing comics or create new ones from scratch.

- **Iterative Interactivity**: Support for continuous refinement through user feedback—on both text and image—enabling a flexible, storytelling-first workflow.

- **Coherent Summarization**: NLP modules generate accurate and context-aware summaries from extracted text in Phase 1.

- **Comic-Styled Image Generation**: Visual output reflects the comic medium with consistent characters, stylistic fidelity, and outlined art.

Design Constraints

- **Development Tools**:
    - Python for all backend services
    - Gradio for initial UI prototyping
    - Tesseract for OCR (Phase 1)
    - Hugging Face Transformers or DeepSeek for summarization
    - Stable Diffusion or similar models for image generation
- **Model Execution Environment**:
    - Local hardware for development and lightweight testing

    - GPU acceleration and DGX supercomputer access for model fine-tuning and evaluation

- **Team Structure**:

    - 2–4 developers with varied experience across NLP, computer vision, and web systems

    - Collaborative development using GitHub

3.2 Design AssumptionsInput Image Assumptions (Phase 1)

- Users upload clear JPG/PNG comic panels
- Speech bubbles are cleanly aligned and mostly in English

Prompt and Context Assumptions (Phase 2)

- Prompts are paragraph-length, in English, with basic scene/character descriptions
- Optional "context" block provided for narrative anchoring (e.g., summary or key event)
- Users iteratively refine prompts and feedback using GUI controls (e.g., CFG scale, negative prompt)

Model Assumptions

- Tesseract OCR is sufficiently accurate without fine-tuning
- Summarization uses pre-trained models like DeepSeek or Pegasus
- Stable Diffusion is used in its public or fine-tuned variant for image generation
- All models are accessed via Python APIs or diffusers library

Infrastructure Assumptions:

- Development occurs on local systems with GPU support
- No persistent backend/database unless explicitly enabled in future phases
- Firebase (Auth, Firestore, Cloud Storage) considered for cloud expansion

Developer Assumptions

- Comfortable with Python, REST APIs, PyTorch, and Hugging Face tools
- Development workflows are version-controlled and modular

3.3 Significant Design Packages

Package	Responsibilities
UI Layer	Presents structured prompt forms, manages user inputs, displays plot and images
Prompt Processor	Collects prompt data and optional context, handles augmentation and chaining
Text Generation Engine	Generates and refines comic plots using LLMs (e.g., DeepSeek)
Image Generator	Interfaces with Stable Diffusion models; allows CFG control, negative prompts, iterative generation
Feedback Manager	Accepts user modifications to plot/image, manages regeneration workflows
Session Handler	Tracks state across pages, manages prompt history, versioned outputs, and iteration gallery
Exporter	Packages finalized pages as PNG, TXT, or ZIP files for download
Queue Manager	(Planned) Handles async tasks via Celery and Redis for non-blocking image generation
Cloud Connector	(Planned) Manages user authentication, persistent storage, and exports using Firebase

3.4 Dependent External Interfaces

External Application and Interface Name	Module Using the Interface	Functionality / Description
Tesseract OCR Engine (pytesseract)	OCR Module (ocr_processor)	Extracts raw dialogue and narration from uploaded comic images for summarization or context-aware generation.
Hugging Face Transformers API	Text Generation Engine (story_engine)	Loads pretrained NLP models (e.g., DeepSeek, Pegasus) for plot generation and summarization.
Diffusers Library (Stable Diffusion)	Image Generator (visual_synthesis)	Generates comic-style images from confirmed prompts using latent diffusion models.
Gradio Web Interface	UI Layer (gradio_interface)	Presents input forms, displays outputs, and manages user interaction with minimal configuration.
Python Standard Libraries (e.g., json, os)	Session Handler, Export Module	Handles file operations, data export, session storage, and response formatting.
(Planned) Firebase SDK (Auth, Firestore, Storage)	Session Manager, Export Module	Will be used to authenticate users, store export history, and host assets like images and PDFs (future integration).
(Planned) Celery + Redis	Task Queue / Async Job Dispatcher	Will handle background tasks such as image generation or large batch export without blocking the UI (for future scalability).

3.5 Implemented Application External Interfaces (and SOA Web Services)

This section outlines the interfaces **owned and implemented** by Comic Insights. These are primarily internal for now, exposed through a local Gradio interface. While no external-facing APIs or services are available in the current phase, the architecture is designed to allow clean expansion toward REST APIs and hosted web applications in future iterations.

Interface Name	Module Implementing the Interface	Functionality / Description
Comic Generation UI (Local)	gradio_interface	Provides an interactive Gradio-based UI where users input story prompts, confirm plot text, and generate image panels. Enables full session-level control, including prompt editing, CFG tuning, and visual feedback.
Session Iteration Viewer	session_manager	Displays all plot/image generations from the current session with timestamped versions. Users can reuse, modify, or export any iteration directly.
Export Interface (Download Panel)	export_module	Embedded in the Gradio UI, this component allows users to download individual or grouped comic pages in PNG, TXT, or ZIP format. Future updates may expose this as an API.
(Planned) REST API for Plot/Image Generation	story_engine, visual_synthesis	Planned interface for exposing backend functionality (e.g., plot generation, image synthesis) over REST. Will be implemented once the project migrates beyond local-first architecture.
(Planned) Web-Based Interactive UI	*(To replace **gradio_interface**)*	In a future deployment phase, the Gradio UI may be replaced by a custom-built web interface hosted on the project's own domain, with improved responsiveness and UX.

4. Logical View

The logical view presents the internal design of Comic Insights, structured around modular, functional layers that reflect the core use cases and system goals. The focus remains on building a working, local-first prototype with clean separation of responsibilities, minimal overhead, and modular design — while keeping future extensibility in mind.

The architecture follows a simplified Model-View-Controller (MVC) paradigm:
- UI Layer: Captures input and displays output via Gradio
- Application Logic Layer: Manages prompt processing, feedback, and orchestration
- AI Core Layer: Handles model-based NLP and image generation
- Persistence Layer (planned): Future integration of lightweight database or Firebase backend

4.1 Design Model

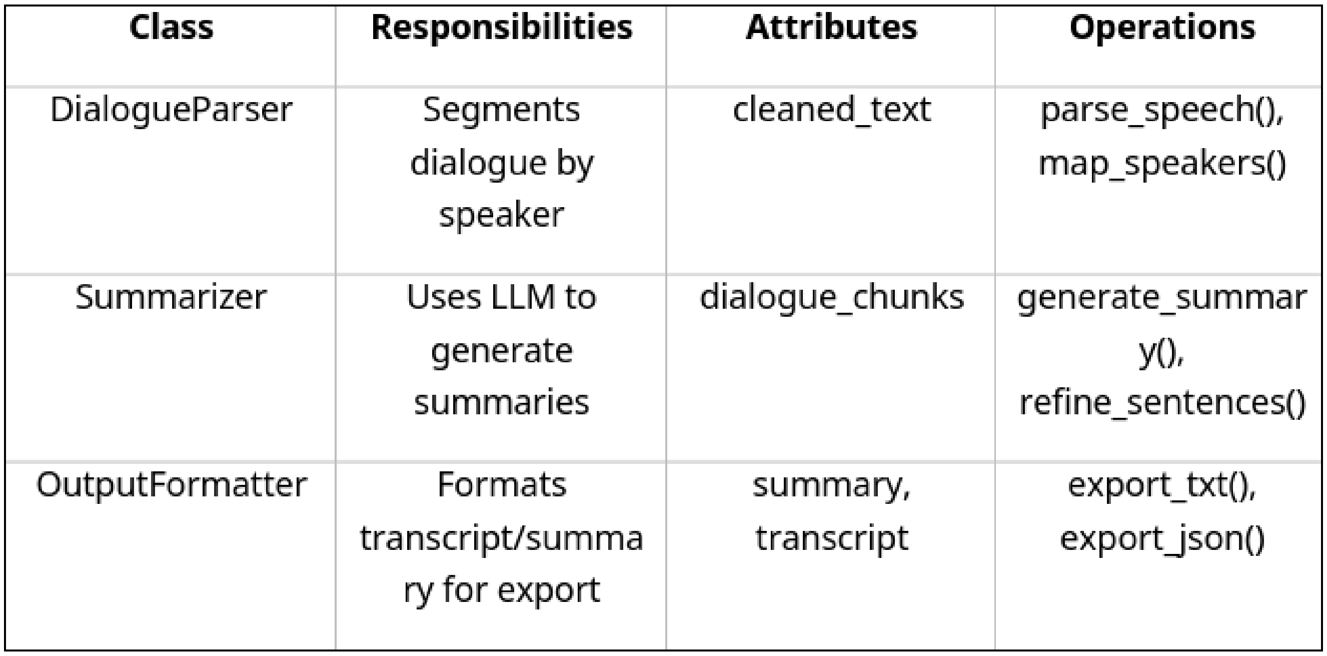
UI Module (gradio_interface)

Class	Responsibilities	Attributes	Operations
UIController	Handles user input/output, task routing	current_task, prompt_input, feedback_input	render_ui(), get_user_input(), display_output()
SessionManager	Tracks live session data, manages iterations	session_id, iteration_history, active_panel	start_session(), log_iteration(), get_gallery()

OCR Module (ocr_processor) (Phase 1)

Class	Responsibilities	Attributes	Operations
TextExtractor	Extracts text using Tesseract	preprocessed_image	run_ocr(), extract_text()
ImagePreprocessor	Prepares comic images for OCR	image_data	resize(), denoise(), enhance_contrast()
TextCleaner	Formats and normalizes extracted text	raw_text	clean_text(), detect_speakers()

Summarization Module (summary_engine)



Class	Responsibilities	Attributes	Operations
DialogueParser	Segments dialogue by speaker	cleaned_text	parse_speech(), map_speakers()
Summarizer	Uses LLM to generate summaries	dialogue_chunks	generate_summary(), refine_sentences()
OutputFormatter	Formats transcript/summary for export	summary, transcript	export_txt(), export_json()

Comic Generation Module (comic_creator)

Class	Responsibilities	Attributes	Operations
PromptProcessor	Prepares user input + context for models	raw_prompt, context_block	tokenize(), augment()
ImageGenerator	Interfaces with diffusion models	prompt, cfg_scale, neg_prompt	generate_image(), apply_style()
FeedbackManager	Parses feedback and adjusts prompts	feedback_log, iteration_state	revise_prompt(), regenerate()

Persistence Module (Planned)

Class	Responsibilities	Attributes	Operations
LocalDataStore (current)	Saves session data locally as JSON	session_data, output_dir	save_local(), load_session()
FirebaseConnector (planned)	Syncs with Firestore and Cloud Storage	user_id, firebase_auth	upload_export(), load_remote_project()
ExportManager	Handles all output formats	export_type, file_map	generate_zip(), download()

Relationships and Interactions:

- UIController interacts with both OCR and Comic Generation modules based on the task selected.
- PromptProcessor, ImageGenerator, and FeedbackManager form a feedback loop for iterative comic generation.
- SessionManager stores all plot/image iterations and passes data to ExportManager.
- Logger (shared utility) handles event tracking and error logging across modules.

4.2 Use Case Realization

Use Case 1: Generate Comic Page from Prompt

Modules Involved: UIController, PromptProcessor, Summarizer, ImageGenerator, SessionManager

Flow:

- 1. UIController collects prompt and context
- 2. PromptProcessor prepares prompt for LLM
- 3. Summarizer generates plot, user confirms
- 4. ImageGenerator renders image
- 5. SessionManager logs output

Use Case 2: Regenerate Panel Using Feedback

Modules Involved: UIController, FeedbackManager, PromptProcessor, ImageGenerator, SessionManager

Flow:

- 1. User edits plot or submits feedback
- 2. FeedbackManager adjusts prompt
- 3. ImageGenerator regenerates panel
- 4. SessionManager stores new version

Use Case 3: Export Comic Page

Modules Involved: SessionManager, ExportManager, LocalDataStore

Flow:

- 1. User selects page(s) to export
- 2. ExportManager bundles output

- 3. LocalDataStore saves to disk

5 Data View

Comic Insights manages a set of structured data entities during each user session. While the system currently operates without a persistent database, it maintains an in-memory session context that temporarily stores all inputs, generated outputs, and revision history. This transient model forms the basis of the system's domain layer.

In Phase 1, this model supported comic understanding through OCR and summarization. In Phase 2, it has expanded to include prompt-driven image generation, feedback-based refinement, and session-based version tracking.

A lightweight export layer enables users to download these entities as .png, .txt, .json, or .zip files. In the future, these data entities will be mapped to a persistent backend (e.g., Firebase Firestore and Cloud Storage) to support project saving, version control, and collaborative creation.

5.1 Domain Model

Entity	Applies to	Current Scope	Description
ComicImage	Phase 1 & 2	✅ Yes	Stores user-uploaded panels (Phase 1) or AI-generated visuals (Phase 2).
OCRTranscript	Phase 1	✅ Yes	Contains raw and cleaned dialogue extracted from comic panels.
ComicSummary	Phase 1	✅ Yes	Generated summaries of the dialogue, used for context or understanding.
PromptInput	Phase 2	✅ Yes	User-structured prompt form + optional context block for story/scene input.

GeneratedImage	Phase 2	✅ Yes	Resulting image panel based on prompt; includes style settings and generation metadata.
FeedbackIteration	Phase 2	✅ Yes	Stores each re-generation cycle, tracking feedback, adjusted prompts, and outputs.
SessionContext	Phase 1 & 2	✅ Yes	Holds the current session's complete history of inputs, prompts, outputs, and metadata.
UserProject (Planned)	Future Phase	🚫 No (Planned)	Represents a saved comic project across sessions, with and

5.2 Data Model (Persistent Data View)

Current Implementation:

- All data is stored in runtime memory only during a session.
- Users can optionally export data in common formats:
- .png — Generated or uploaded images
- .txt / .json — Plot summaries, transcript, prompt metadata
- .zip — Bundled download of a full session or comic page set

Planned Future Extension:

- Firebase Firestore (NoSQL) will store user sessions, project data, and revision history.
- Firebase Cloud Storage will host user-generated images and exported artifacts.
- User Authentication Layer (e.g., Google OAuth) will be added to manage access and ownership.

5.2.1 Data Dictionary

Field	**Entity**	**Type**	**Phase**	**Current Scope**	**Description**
image_id	ComicImage	String	1 & 2	✅ Yes	Unique identifier for image (filename or hash).
raw_text	OCRTranscript	String	1	✅ Yes	Raw OCR output from speech bubbles.
cleaned_text	OCRTranscript	String	1	✅ Yes	Cleaned and structured version of the text.
summary_text	ComicSummary	String	1	✅ Yes	Summarized narrative generated via NLP.
prompt_text	PromptInput	String	2	✅ Yes	User's full natural language prompt.
context_block	PromptInput	String	2	✅ Yes	Optional contextual narrative or backstory.
generated_image_path	GeneratedImage	String	2	✅ Yes	Local file path or remote URL of the generated panel.

cfg_scale	GeneratedImage	Float	2	✅ Yes	Configuration value controlling prompt adherence.
style_tag	GeneratedImage	String	2	✅ Yes	User-selected art style (e.g., “manga”, “noir”).
user_feedback	FeedbackIteration	String	2	✅ Yes	Feedback used for refining the image or plot.
iteration_number	FeedbackIteration	Integer	2	✅ Yes	Index of the feedback loop in session history.
timestamp	SessionContext	Datetime	1 & 2	✅ Yes	Time when input or generation occurred.
session_id	SessionContext	String	1 & 2	✅ Yes	ID of the current
user_id	UserProject (Planned)	String	Future	🚫 No	Logged-in user identifier linked to saved projects.
project_id	UserProject (Planned)	String	Future	🚫 No	Project-level grouping for multi-page comics.

6 Exception Handling

Exception handling in Comic Insights is designed to ensure that user-facing errors are communicated clearly and constructively, while technical issues are logged in detail for developers. The system uses custom exception classes tied to specific modules and workflows, with friendly UI messages, contextual logging, and safe fallback mechanisms where possible.

All exceptions follow this core strategy:
• Catch → Log → Notify User → Offer Action (Retry/Edit/Fallback)

Exception Name	**When It Is Thrown**	**Handled By**	**User Message / Follow-Up Action**
FileFormatException	When the uploaded file is not a supported image format (e.g., .txt, .docx).	UIController	"Invalid file format. Please upload a JPG or PNG comic image."
OCRFailureException	If OCR fails to extract any text from the comic image (e.g., blurry or empty).	TextExtractor	"Text could not be extracted. Try using a clearer image."
SummarizationException	If the summarization model crashes or produces empty/invalid output.	Summarizer	"Unable to summarize content. Please try again or check the input."
PromptProcessingException	If the user's prompt is missing, malformed, or exceeds model input limits.	PromptProcessor	"Your prompt seems incomplete or too long. Please revise it."
ImageGenerationException	When the image generation model fails due to internal error or corrupted input.	ImageGenerator	"Image generation failed. Try again with a different prompt."
FeedbackLoopException	If the feedback-based re-generation fails due to input conflict or memory errors.	FeedbackManager	"Something went wrong while refining the image. Restarting iteration."
ModelLoadException	If an AI model (OCR/NLP/Image) fails to load properly during startup.	ModelLoader	"Required model could not be loaded. Some features may be unavailable."

Logging Strategy

All exceptions are logged using the Logger utility. Each log entry includes:

- Timestamp
- Module name
- Exception type and message

- Stack trace (if available)
- Input context (e.g., filename or prompt snippet)

User-Facing Behavior

Where possible, technical errors are caught before reaching the UI. The system provides friendly error messages like:

- “We couldn't extract any text from this image.”
- “The prompt seems empty or unclear. Please revise and try again.”
- “Something went wrong while generating the image. Let's try again.”

7. Configurable Parameters

Comic Insights supports a small set of tunable parameters that give users real-time control over both the **plot generation** and **visual output**. These parameters are implemented in a way that aligns with the project’s goal of delivering a smooth, intuitive, and iterative comic creation experience.

All current parameters are configurable directly from the Gradio UI and do **not require restarting the application**. Most are passed into the backend modules as function arguments and affect output generation dynamically.

Key Parameters and Implementation

- **CFG Scale**
  Controls the adherence of generated images to the input prompt. Implemented as a slider in the UI; passed into the Stable Diffusion pipeline as `cfg_scale`.

- **Style Tag**
  Defines the visual art style (e.g., "manga", "noir"). Selected from a dropdown; injected into the prompt string by `PromptProcessor`.

- **Prompt Length Limit**
  Enforced in `PromptProcessor` to prevent overflow. Prompts exceeding the limit trigger a validation warning in the UI.

- **Negative Prompt**
  Allows users to specify what should *not* appear in the image. Set via a text field and passed to the image generation model as `negative_prompt`.

- **Seed** *(Planned)*
  Enables reproducible image outputs. Will be added as an optional numeric field and used to seed the

diffusion model.

This configuration system supports the **user-centric and exploratory nature** of Comic Insights while keeping the implementation minimal and beginner-friendly. Parameters are modular and can be easily extended in future UI or API layers.

8. Quality of Service

This section outlines the reliability, security, performance, and monitoring considerations in the design of Comic Insights. As a local-first, minimal-dependency prototype, the system focuses on ease of use, fast iteration, and self-contained deployment. However, the design also anticipates future expansion into cloud deployment, user authentication, and scalable project storage using services like Firebase. Additional GUI components are also being introduced progressively to improve user accessibility and creative control.

8.1 Availability

• **Local-first architecture**: The system runs on local machines, avoiding dependency on external APIs or services.
• **Stateless sessions**: No persistent database is required, simplifying restarts and enabling zero-downtime recovery.
• **Low-maintenance deployment**: Restarting the system or updating configurations requires minimal effort.
• **Downtime considerations**: Sessions reset on demand. No background processes or maintenance cycles required.
• **Planned**: Future server-based deployment will allow real-time collaboration, saving, and background queue processing.

8.2 Security and Authorization

A. Current Security Scope

• No personal data storage, reducing privacy risks.
• Strict file type restrictions (only .jpg/.png).
• Prompt and feedback inputs are sanitized before model processing.

B. Authorization and Access Control

• No authentication in current version.
• Assumes a trusted local environment (e.g., personal laptop or lab PC).
• All backend access controlled through UI components.

C. Security Features

- Temporary files are deleted post-session.
- Logs contain no personal identifiers.
- Backend APIs and model internals are not exposed to the user.

D. Future Enhancements

- Firebase Auth will be used to introduce secure login for saving and sharing projects.
- Role-based access and user-level storage management will be supported.

8.3 Load and Performance Implications

A. Expected Load Characteristics

Component	Expected Load	Usage Notes
Comic Upload (OCR)	5–10 images/session	Avg. 200–800 KB per file; sequential processing
Text Summarization	~100–1000 words/transcript	Token limits enforced; <10s response target
Prompt-based Generation	1–2 prompts/page, 5–10 feedback loops	15–30s with GPU
Session Duration	5–15 minutes	Auto-reset after inactivity
Concurrent Users	1 (solo), 2–3 (lab)	No queueing or session isolation yet

B. Performance Goals

Feature	Target
OCR Response Time	≤ 5 seconds/image
Summarization Time	≤ 10 seconds/transcript
Image Generation Time	≤ 30 seconds (w/ GPU)
Feedback Re-iteration	≤ 25 seconds

UI Response Time	< 1 second
Max RAM Usage	≤ 8 GB peak

C. Performance Test Scenarios

Test Case	**Goal**
Uploading 5 comic panels	Measure OCR + summary performance
Prompt >100 words	Validate input handling, token management
10 feedback iterations	Test memory stability and regeneration time
Concurrent use on shared PC	Measure performance under light load
GPU vs CPU fallback	Compare generation speed across hardware

8.4 Monitoring and Control

• Central logger handles error/warning capture and system events.
• Logs saved in 'logs/error_log.txt'; console logging is enabled in dev mode.
• User-facing fallback messages improve recovery and awareness.
• No autonomous daemons or async handlers implemented yet — all interactions are user-triggered.
• Planned: Lightweight admin dashboard for usage tracking, error review, and future deployment health check