ChatGPT

Functional Requirements for an AI-Assisted Novel Writing App

Below is a comprehensive list of functional requirements for a single-user web application like NovelCrafter. Features are grouped by areas (writing, planning, AI, prompts, integration, etc.) and drawn from NovelCrafter's documentation. Each requirement is phrased as a capability of the system and supported by citations from NovelCrafter's docs.

Manuscript Editor (Writing Interface)

- **Rich text editing:** The app shall provide a clean, distraction-free word-processor interface for drafting prose [1]. Users can apply common formatting (bold, italics, lists, headings, alignment, etc.) and adjust font, paragraph spacing, and width via a format menu [2].
- **Scene/Chapter structure:** The novel is organized into scenes and chapters. The writer can view and edit a scene's prose in sequence, add new scenes (via a button or slash command), and collapse/expand sections of the text. A sidebar or panel shows the current chapter and scene selection, and the user can navigate chapters from this sidebar [3].
- **Scene metadata panel:** Each scene includes a detail panel showing metadata such as scene number, word count, point-of-view (POV) character, and chapter summary [4]. From this panel (or its action menu), the user can set or override the scene's POV, add subtitles or time/location notes, duplicate the scene, export its prose, or adjust other scene-specific settings [5].
- **Action menus for scenes/chapters:** Contextual menus allow scene-level actions: toggle automatic numbering on/off, split/merge scenes or chapters, exclude a scene from AI context, add subtitles, and delete or archive a scene. Chapter menus allow copying or exporting all beats or prose in that chapter, deleting empty chapters, and toggling numbering. A "whole novel" menu lets the user copy all prose or summaries to the clipboard and delete unused empty scenes [6] [7].
- **Integrated AI tools in editor:** The writing interface integrates AI tools at the scene level. For any scene, the user can invoke commands to **summarize** the scene or **detect characters** in it, adding the results to the codex, or start a chat "with this scene" for brainstorming/analysis [5] [8]. These AI actions are available from the scene's action menu or via slash commands.
- **Slash-command generation:** Typing "/" in the editor opens a command menu to generate scene content. For example, choosing "Scene Beat" and typing instructions lets the AI produce new prose content under the current scene [9]. After generation, the user can apply (accept) the text, retry (regenerate), discard, or insert it as a collapsible **section** for modular editing [10].
- **Text-replacement prompts:** The editor supports AI-driven text transformations on selected text. When the user highlights text, they can choose from built-in prompts (e.g. **Expand**, **Rephrase**, **Shorten**) or custom prompts. These prompts rewrite the selected text (expanding details, changing POV/tense, adding inner thoughts, summarizing shorter, etc.) according to user inputs [11] [12].
- **Focus and display options:** The interface offers a "Focus Mode" (full-screen, distraction-free writing) and allows theme changes (e.g. dark mode) and font toggles for comfort. (NovelCrafter mentions focus mode and dark mode as features [1] and the ability to adjust fonts and spacing [2]).

Story Planning & Outline Interface

- **Hierarchical story view:** The app shall let the user structure the novel into Acts, Chapters, and Scenes. The user can add new acts/chapters/scenes and reorder them via drag-and-drop in a **Grid** view. By default the plan interface shows a grid of Acts/Chapters/Scenes (with chapter summaries), allowing reordering and quick navigation.
- **Multiple planning views:** In addition to a Grid, the app provides a **Matrix** view to track story elements and a linear **Outline** view. In Matrix view the user can choose a category (character, location, custom tag, etc.) as columns and see which scenes contain which elements. In Outline view the scenes are listed sequentially with their summaries for a compact overview.
- **Scene labels and custom fields:** Users can assign custom labels (e.g. “Draft”, “Final”) or color-coding to scenes to track their status. They can also define custom scene attributes (e.g. POV tags, subplots) that appear in the plan cards. (The features list mentions *Scene Labels* and *Custom POVs*.)
- **Scene subtitles and references:** Each scene can have a subtitle or note (e.g. indicating time skips or location) for reader clarity [13] . In the plan view, manual references can be added to highlight key locations, timelines, or plot points across scenes [14] .
- **Outlines and templates:** The app should allow generating or importing an outline. For example, the user can paste a text outline or use pre-set outline templates (3-act, 5-act, etc.) which the app converts into acts, chapters, and scenes. A preview step shows how the outline will be structured before confirming import.
- **Search & filter:** The plan interface includes a global search box to find scenes by keywords in their titles or summaries, or by codex tags. The user can filter the view by tags (e.g. filtering to one character’s scenes or a subplot).
- **Timeline overview:** A collapsible **story timeline** visually represents the arrangement of scenes/chapters. Each scene is marked on a vertical timeline bar and clicking a marker jumps to that scene in the editor [15] . The current scene is highlighted. This gives a quick view of pacing and distribution.

Codex (World & Character Database)

- **Structured entry editor:** The app shall include a **Codex** wiki where the user can create entries for Characters, Locations, Items, Lore, Organizations, Subplots, etc.. Each entry has a type (chosen at creation) and a name, plus an area to write a descriptive “sheet” [16] [17] .
- **Tags/Labels:** Each codex entry can be tagged or labeled (e.g. “Protagonist”, “Antagonist”, “City”, “Continent”, etc.) for organization [18] . Tags help the user filter and group entries but are not automatically sent to the AI (tags are for human organization).
- **Thumbnails and appearance:** The user can assign a thumbnail image to each entry (e.g. a portrait for characters, a map for a location) [19] . The app should allow uploading and cropping of one image per entry to aid visual recognition.
- **Aliases/Nicknames:** The user can list alternative names (aliases, nicknames, acronyms) for each entry [20] . Aliases are automatically highlighted in the manuscript and used in AI context. For example, adding “Bob” as an alias for “Robert Smith” ensures the AI knows they are the same.
- **Description and custom fields:** Each entry has a free-form description field where the user writes any information the AI should know (appearance, history, role, etc.) [17] . Optionally, the system supports structured “detail fields” where the user can define custom attributes (e.g. Role = “Detective”, Species = “Elf”) [21] [22] . These details can be used to format character/object profiles or for automated lists.

- **Research notes:** A separate “research” or “notes” tab lets the user store additional info or links (e.g. research, images, quizzes) that are **not** sent to the AI when writing [23]. This is for offline notes or development information.
- **Relations (nested entries):** The user can link entries together (e.g. family relationships, organizations). A codex entry can contain relations (nested references) to other entries. For example, linking five members under “Council of Five” means querying the council will also bring up all five members’ sheets [24]. The interface should allow adding, swapping, and removing relations via a searchable dropdown.
- **Mentions tracking:** The Codex tracks where each entry appears in the novel. A “Mentions” tab lists every mention of the entry (by name or alias) across scenes, summaries, chats, and snippets [25]. In the editor, mentions are highlighted (with colors) and a mini-timeline shows where in the novel the entry is mentioned [26].
- **AI context controls:** The user can specify how each entry is sent to the AI. Options include: “Always include” (global entry), “Include when detected” (default), “Exclude from AI” (ignore unless manually added), or “Never include” (private notes) [27]. The plan interface also has a toggle per scene to exclude it from AI context (useful for notes or NSFW content) [28].
- **Progression history:** The app keeps a history of changes to each codex entry (for example, tracking how a character evolves) and shows “codex additions” under the description [29] [30]. (A related *Progressions* feature shows how entries change over time.)
- **Series support:** If the user writes multiple novels, codex entries can be shared across a series so that world data is not re-entered. (The features page notes *Series – Share your Codex* [31].)

Snippets (Notes & Ideas)

- **Quick notes repository:** Provide a *Snippets* section for jotting down quick notes, to-dos, or research that is separate from the novel text [32]. The user can create, edit, and title snippets like simple documents. Snippets can be plain text or even markdown notes.
- **Pinning and access:** Snippets can be “starred” or pinned so they stay visible alongside the manuscript or other views [33]. The sidebar should allow dragging a snippet to re-size or reference while writing (e.g. split-screen with manuscript).
- **Import excerpt:** The user can convert existing text into codex entries or scenes via an “Extract from snippet” function (as mentioned on the features page [34]). For example, selecting text in a snippet and using “Extract to Codex” automatically creates a character/location entry with that name and content.

AI/Chat Interface

- **Chat workspace:** Include a chat interface for free-form AI interaction [5]. Chats have threads and can run GPT or other LLMs. In a chat, the user types messages and receives AI replies. Each chat thread has a name (editable) and may be pinned or split (so multiple chats can be open side-by-side). Users can create multiple chats.
- **Context options:** Before sending a message, the user can select what context to include: e.g. the entire novel, a specific scene/chapter, the current outline, selected codex entries, or a snippet. This context is prepended to the AI prompt. The interface provides a menu for context selection [35].
- **Prompt and model switching:** The user can choose or switch the AI model (OpenAI GPT-3.5/4, Claude, local Llama, etc.) and the active prompt/preset for the chat [36]. There should be a dropdown to change the system prompt or model at any time, even mid-conversation [36].

- **Default and custom prompts:** The chat should use a default conversation prompt, but the user can also apply any prompt from the prompt library on the fly. A prompt preview is available (showing exactly what is sent) and the user can edit prompt inputs (e.g. word count, style) before sending.
- **Thread actions:** For each chat thread, include actions like pin/unpin, duplicate (fork), archive (hide), and delete. The user should be able to export a chat (save transcript to file) or copy text.
- **Transfer to novel:** The user can easily import outputs from chat into the manuscript or codex. For example, a button “Import to scene” or “Add to codex” would take selected text from chat and insert it into the novel. (The docs mention “Extract feature” to pull structured data from chat, though details are limited.)

Prompt and Template Management

- **Prompt library:** The system must include a **Prompt Library** where users can create, save, and organize custom AI prompts [37] [38]. The library shows a list of prompts (with search and filter) and a detail pane for editing. Each prompt has a name, description, category, and configuration tabs. The user can create new prompts, clone existing ones, or import shared prompts.
- **Prompt structure:** Prompts can include text and dynamic parts. The interface should support prompt *inputs* or variables (e.g. a placeholder for a word count or subject) that are filled in when the prompt is used [39]. Complex prompts can be built from modular components or sub-prompts.
- **Prompt types:** The app provides several prompt types, such as a general “scene completion” prompt for prose generation, built-in text-transformation prompts (expand/rephrase/shorten) [11] [12], and system/user message templates for chat. Users can define defaults (e.g. default chat or text prompts for certain contexts) [9].
- **Preview and testing:** Before sending to the AI, the user can preview the final prompt text (with variables filled) and optionally copy it. The interface may allow “tweak and generate” to adjust inputs (like temperature, word count) on the fly [40].
- **Model settings:** Prompts are sent to a selected model. The user can configure model parameters (temperature, top-p, max tokens, etc.) per prompt or chat. The settings UI should allow adjusting these parameters for each generation request [41].

Model/AI Connections

- **Connect to multiple LLM providers:** The app shall support integrating both cloud and local LLMs. For cloud, provide connectors for OpenAI (GPT-3/GPT-4), Anthropic/Claude, Google Gemini (via Google Cloud API), and any OpenAI-compatible endpoints (Groq, xAI, etc.) [42] [43]. For on-premises, support local APIs like LM Studio or Ollama running on the user’s machine [44] [45].
- **API key management:** In a Settings or Profile page, allow the user to enter API keys or OAuth tokens. For example, connecting to OpenAI requires pasting the secret key from platform.openai.com [46] [47]. For OpenRouter (Anthropic, Claude), allow OAuth login. For other vendors, allow specifying a base URL and key (OpenAI-API compatible mode) [48] [43].
- **Local model connections:** For LM Studio or Ollama, provide a form to enter the local server URL/port. The app should verify connectivity. For Ollama specifically, instruct the user to set the `OLLAMA_ORIGINS` environment variable so the browser-based app can access it [49] [45]. These local models are listed in the model selection menu after connecting.
- **Model selection UI:** In the writing and chat interfaces, the user can pick which model to use for each generation. The system should display available models from all connected providers (e.g.

`GPT-4o (OpenAI)`, `Llama2-13B (LM Studio)`, etc.). It should show disabled models (e.g. GPT-4 locked until top-up) and hide unsupported ones (e.g. filtered NSFW models) [50].
- **Cost & limits:** The app should warn or disable certain models if not accessible (e.g. GPT-4 until account has $5 spend) [50]. It should also inform the user of usage or credit limits (e.g. OpenRouter credits).
- **Fallback & offline:** If no cloud connection is available, the app can still use local models. (NovelCrafter requires at least one AI connection for AI features, but basic writing should still work without it.)

Content Import/Export & Data Management

- **Import existing work:** The user can import a novel from Word (.docx) or Markdown files. The importer should parse headings: e.g. Heading 1 = Acts or Chapters, Heading 2 = Chapters (if acts absent), and `***` or blank lines to split scenes [51]. The user chooses whether to import text as full prose or as chapter summaries. A preview shows the parsed acts/chapters/scenes before confirming [52]. The import function should accept `.docx` and `.md`.
- **Export to standard formats:** The app provides an **Export** function to save the project. Users can export the entire novel (prose and/or summaries) as Word (.docx) or Markdown. They can toggle options: include/exclude act titles (for Vellum/Atticus compatibility) and scene subtitles [53]. Users can also choose to export only selected parts (specific chapters/scenes) via checkboxes [54]. For example, "Export to Scrivener" yields a Markdown file with headings for scenes [55].
- **Export metadata:** The export dialog allows including supplementary data. For example, the user can choose to include the Codex contents, chat histories, and snippets in the export package [56]. For novel text, the user can include or omit scene summaries, subtitles, and act labels via toggles [54].
- **Archiving scenes:** Scenes removed from the main story can be archived rather than deleted. Archived scenes go into a safe repository (accessible via an "archive" icon in the plan view), and can be restored later [57]. The app should allow the user to archive a scene to free up space, and later browse archived scenes to restore or delete them.
- **Revision history:** The system maintains an automatic version history for key content. For example, every edit to a scene's summary or content, or a codex description, is saved so the user can revert to any previous version [58]. The user interface should offer "history" buttons near editable fields; clicking it opens a timeline of past versions to review and restore [59] [60]. This also supports undo/redo functionality.
- **Novel covers:** The user can upload or select a cover image for the novel. (The features page shows *Novel Covers* [61].) The app will display the cover in the project overview and on export (embed in docs).
- **Localization (optional):** The app should be designed to support multiple languages. UI elements and system messages should be localizable. (While not explicitly in documentation, supporting multilingual UI or exporting in different languages is desirable.)

Organizational & Miscellaneous Features

- **Dashboard/project list:** On login, the user sees a list of novels (projects). Each project has metadata (title, pen name, cover, POV, tense, etc.). The user can create a new novel from scratch or from a template [62]. Pen names allow grouping multiple novels under an identity [63].
- **Theme and font settings:** The user can switch between light/dark mode [64] and adjust font sizes or enable dyslexia-friendly fonts for accessibility. These preferences apply across the app.

- **Collapsable UI panels:** The sidebar (with codex/chat/nav) can be collapsed or split. Users can "pin" a chat beside the manuscript to view both simultaneously [65]. The interface should allow resizing and docking panels.
- **Search within project:** A global search should find text across the manuscript, summaries, codex, and snippets. The help docs mention searching plan by keyword; similarly, free-text search across the project is expected.
- **Backup/restore:** The user should be able to download a full backup of the project (including text, codex, chat, etc.). Conversely, the app should autosave continuously to prevent data loss. If writing vanishes (e.g. internet glitch), there should be recovery options (as hinted by docs on revision history [58] and quick-start guides).
- **User account & profile:** Though single-user only, the app will have a user account system (sign-up/login). The user's API keys and settings are stored in their profile. (Settings for AI connections must be exportable and importable if the user uses multiple devices [66].)
- **Context-help and documentation:** Provide inline help tips and tooltips (e.g. explain what "Exclude from AI" does) and link to documentation for advanced features (as seen in help pages).

All features above are drawn from NovelCrafter's official documentation (see citations). Together, they define a complete, structured set of functional requirements for a one-person AI-enhanced novel-writing web application.

Sources: Feature descriptions and how they work are documented in NovelCrafter's help docs [1] [17] [6] [11] [10] [46] [67] [32], as cited above.

---

[1] [2] [3] [4] [5] [15] The Manuscript - Write - Novelcrafter Help
https://www.novelcrafter.com/help/docs/write/the-write-interface

[6] [7] [8] [13] [57] Actions Menu (Plan) - Plan - Novelcrafter Help
https://www.novelcrafter.com/help/docs/plan/plan-actions-menu

[9] [10] [40] Generating Prose - Write - Novelcrafter Help
https://www.novelcrafter.com/help/docs/write/generating-prose

[11] [12] [39] Text Replacement Prompts - Write - Novelcrafter Help
https://www.novelcrafter.com/help/docs/write/text-replacement-prompts

[14] [22] [26] [28] [30] [31] [34] [35] [36] [41] [61] [62] [63] [64] [65] Discover all the features - Novelcrafter
https://www.novelcrafter.com/features

[16] [17] [18] [19] [20] [21] [23] [24] [25] [29] Anatomy of a Codex Entry - Codex - Novelcrafter Help
https://www.novelcrafter.com/help/docs/codex/anatomy-codex-entry

[27] [45] [49] Ollama - AI Connections - Novelcrafter Help
https://www.novelcrafter.com/help/docs/ai-connections/ollama

[32] [33] Creating a snippet - Snippets - Novelcrafter Help
https://www.novelcrafter.com/help/docs/snippets/snippets

[37] [38] Prompt Library - Prompts - Novelcrafter Help
https://www.novelcrafter.com/help/docs/prompts/prompt-library

[42] [43] [46] [47] [48] [50] [66] OpenAI - AI Connections - Novelcrafter Help
https://www.novelcrafter.com/help/docs/ai-connections/openai

[44] [67] LM Studio - AI Connections - Novelcrafter Help
https://www.novelcrafter.com/help/docs/ai-connections/lm-studio

[51] [52] Word (.docx) - Import - Novelcrafter Help
https://www.novelcrafter.com/help/docs/import/word

[53] [54] [56] Novel - Export - Novelcrafter Help
https://www.novelcrafter.com/help/docs/export/novel

[55] Export to Scrivener - Export - Novelcrafter Help
https://www.novelcrafter.com/help/docs/export/to-scrivener

[58] [59] [60] Revision History - Organization - Novelcrafter Help
https://www.novelcrafter.com/help/docs/organization/revision-history