ChatGPT

Quill AI – Competitive Feature Gap Analysis

1. Workflow Gaps: Outlining & Storyboarding (Pantser vs. Plotter)

Quill's current MVP is focused on editing text ("pantser" style), but leading tools support extensive planning for "plotter" writers. For example, **Scrivener** provides a Binder (hierarchical chapters/scenes) and a Corkboard of index cards that sync with the Binder and Outliner [1] [2]. This allows authors to visually storyboard and rearrange scenes, with changes reflected in the manuscript structure [3]. Modern outlining apps go even further: **Plottr** offers a **Timeline** view to drag-and-drop scene cards along plotlines and chapters [4]. Writers can assign characters, tags, or subplots to each scene card and then filter or flip the timeline to spot plot holes and pacing issues [5] [6]. Likewise, **Sudowrite** includes a *Canvas* feature – a virtual corkboard for arranging AI-generated plot cards and character notes, which writers can shuffle into a coherent narrative order [7]. **Novelcrafter** has a full "Plan" interface with multiple views: a **Grid** to organize acts/chapters (and even show which *Codex* entries appear in each scene), a **Matrix** to track characters or subplots across scenes, and a clean Outline list of scene summaries [8] [9].

Gap: Quill AI lacks a robust outlining/storyboarding toolset. Its current `ProjectSidebar/StoryBoard` is rudimentary, meaning plotters have no visual scene board or timeline to develop a story from idea to chapter. To close this gap, Quill should introduce a **Scene Board / Timeline View** that integrates with the manuscript. This could resemble Plottr's timeline of chapters vs. plotlines with index-card scenes, or a Scrivener-style corkboard linked to Quill's "Multiverse" version branches. Such a feature would let users sketch beats and **reorder scenes non-linearly**, then commit changes back to the draft structure [10] [11]. Outlining templates (e.g. 3-Act, Hero's Journey) and scene synopsis fields would further guide planning – Novelcrafter even provides outline templates (Save the Cat, etc.) and an AI "create from outline" tool to flesh out scenes from a structured outline [12] [13]. Implementing a **timeline/corkboard UI with drag-and-drop scenes** is primarily a front-end challenge (Medium complexity) but high priority, as without it Quill will not attract plotters who rely on visual story organization.

2. Data & Sync Gaps: Local-Only vs. "Pro" Features

Serious writers expect their work to be **safe, synced, and exportable**. Currently, Quill stores everything in IndexedDB locally – a single point of failure and inconvenience. By contrast, **Scrivener** (while not cloud-native) mitigates data loss by automatic local backups on every project close [14]. Scrivener users often sync projects via Dropbox or iCloud to work across devices, and Scrivener itself encourages backups to external drives or cloud locations for the 3-2-1 rule [15]. **Obsidian** takes a "local-first" approach similar to Quill's, but offers options like *Obsidian Sync* (end-to-end encrypted, with version history) or community plugins for Git/cloud drive sync [16]. The message is clear: **cloud sync and robust backup** are must-haves for "pro" authors. Quill should implement **cross-device cloud synchronization**, ideally with optional end-to-end encryption (to assuage privacy concerns). This is High complexity (requiring a backend or using a service like Supabase/Firebase), but top priority – without it, Quill cannot serve as a primary writing tool for users who write on multiple devices or need assurance their 100k-word manuscript is safely backed up off their laptop.

Another gap is **export formats for publishing**. Authors need to easily move their manuscripts to publishing or formatting software. Scrivener sets a standard here with its Compile feature (to DOCX, PDF, ePub, etc.), and even Plottr lets users export timelines to Word or Scrivener format [17]. Tools like **Atticus** (an all-in-one writing/formatting app) match Vellum's output quality by exporting to polished EPUB, print-ready PDF, and Word at the click of a button [18]. Atticus also includes a live preview to see how the book will look on Kindle or print devices [19] [20]. Quill currently lacks any export beyond manual copy-paste. To compete, Quill should support **one-click exports to .docx, .pdf, and .epub** (with basic formatting like chapters, scene breaks, italics intact). This is Medium complexity (front-end libraries can generate these formats), but high priority because without standard exports, users cannot easily publish what they create. Additionally, consider **Scrivener/Atticus integration** – Novelcrafter even built direct export to Scrivener and Atticus files [21], acknowledging that many authors use those for final touches. In short, Quill needs to evolve from a local toy to a "pro" application by adding cloud sync/backup and robust export capabilities in its next phase.

3. AI Memory Gaps: Maintaining Long-Form Consistency (the "Series" problem)

Quill's AI presently loads only the current chapter (or a limited window) for context. This makes it challenging to maintain consistency in a long novel or series – facts from 20 chapters ago can be forgotten, leading to AI **hallucinations or plot holes**. Competing AI writing tools tackle this with larger context windows and structured memory aids. **Sudowrite** and **Novelcrafter** both employ *structured RAG* (Retrieval Augmented Generation) approaches [22]. They require authors to fill in **knowledge buckets** – e.g. Sudowrite's **Story Bible** of characters, world facts, and synopsis, or Novelcrafter's **Codex** entries – which the AI then references during generation. This structured data ensures the AI has a "source of truth" for the story [23] [24]. In practice, Sudowrite's Story Bible is a set of predefined fields (character goals, world info, etc.) that the AI will automatically pull in to keep details consistent [25]. Novelcrafter similarly lets users tag characters/locations in scenes and maintains a Codex; notably, its system prompts automatically include the **summaries of prior scenes** to give the AI context of everything that has happened [26]. This means if a red ball was mentioned in Chapter 2, the AI is reminded of it in Chapter 5's prompt and won't randomly change it to blue [27].

For series or very long works, the approach scales by sharing context across books. Sudowrite introduced **Series Folders** that share the Story Bible info (characters, world, outline) across multiple projects, keeping a series "bible" consistent for all volumes [28] [29]. Changes to a character in Book 1 (e.g. hair color or backstory) propagate to Book 2, and crucially, Sudowrite's AI has access to all those shared entries to avoid contradictions [29]. Novelcrafter likewise has a **Series Codex**: entries marked as "series" are visible in every book of that series and are meant to store key information from previous books for easy AI reference [30] [31]. Beyond structured data, model improvements are extending raw context length – Sudowrite now leverages models like Claude 3 with up to 100k token windows, and even hints at Google Gemini's potential 1 million token context for future continuity [32] [33].

Gap: Quill needs a strategy for **global story memory**. A two-pronged solution is suggested: (a) Implement a **"Lore Bible" or World Wiki** feature where authors can input character profiles, setting details, and plot summaries. This should be accessible to the AI (Quill's multi-persona agents) whenever they generate text, serving as a factual reference (Medium complexity to build UI and prompt logic). (b) Add a **retrieval system for long context** – for example, automatically summarize each chapter and store these embeddings so that

when working on Chapter 20, Quill can retrieve relevant facts from Chapter 1-19. This might require a vector database or at least a clever local indexing of scenes (High complexity, likely needs backend). Without such features, Quill's AI will struggle with series consistency, whereas competitors like Sudowrite explicitly *"take care of hallucinations and context window issues"* by structuring how context is fed to the AI [22]. Closing this gap is high priority for attracting novelists writing series or epics. Quill's advantage is being local-first; it could potentially perform AI inference or at least RAG lookups locally for privacy, but a cloud-based vector store may be more practical once sync is in place. Regardless, introducing a **Story Bible/Codex** that the AI consults (perhaps via Quill's agent tools reading the knowledge graph) will significantly enhance continuity for long-form projects.

4. Market Differentiators: "Delighter" Features from Niche Tools

Beyond parity with major competitors, Quill can differentiate by offering features that many AI writing apps overlook – often the "quality of life" tools that traditional writing software provide:

- **Goal Tracking & Productivity Analytics:** Writing apps like **Ulysses** and **Atticus** include features to set daily word count goals, track writing streaks, and visualize progress. Atticus, for instance, "lets you track your goals and writing habits to gamify the process" [34]. This kind of feature delights users by making the long slog of writing a novel feel like a rewarding game. Quill could implement a goal-setting dashboard (e.g. daily word target, or milestones per chapter) with gentle reminders or celebratory animations upon hitting goals. This is relatively Low complexity (mostly front-end counters), but Low priority compared to core functionality – still, it's a notable differentiator to keep writers engaged and motivated, which current AI-centric tools (focused on text generation) usually ignore.

- **Built-in Advanced Book Formatting:** While AI tools help generate content, they often ignore the final stages of making a manuscript print- or ebook-ready. **Atticus** has turned this into a selling point – it merges a word processor with Vellum-like formatting. Users can customize fonts, chapter themes, section breaks, and then export to polished EPUB, DOCX, or print PDF with one click [18]. Atticus even provides a live device previewer for various screen sizes [19]. Quill could carve a niche by offering **one-stop formatting** as well: for example, allowing authors to select a book theme/template (chapter heading style, ornamental scene breaks, etc.) and preview the styled output. This "what you see is what you get" approach is a delight because it saves authors from purchasing a separate tool like Vellum or Atticus for formatting. Complexity is Medium (it would involve a rendering engine for PDF/EPUB and template management). Priority is Medium – not as immediate as sync or AI features – but implementing even basic **export themes** (e.g. a clean novel format vs. a manuscript format) would set Quill apart from Sudowrite/NovelCrafter, which require exporting text and then using another program for formatting.

- **Real-Time Collaboration & Review:** Many AI writing tools are single-player – they lack the ability for an author to easily invite an editor or co-author to work on the manuscript. Yet collaboration is a delighter for those writing with a partner or seeking professional edits. Atticus recently introduced a collaboration feature where authors can invite others and avoid emailing Word docs back and forth [35]. Google Docs remains popular in writing circles precisely for its real-time co-writing and commenting. Quill, being web-based, is well-positioned to offer **live collaboration** (multiple cursors, comments, suggestions mode). This is High complexity (requires multi-user presence, conflict resolution, and a cloud backend), and likely a lower priority in the short term. However, even a

simpler implementation like **shareable web links for read-only or comments** could delight users who currently have to copy AI-generated text into Google Docs for feedback. Major AI competitors haven't focused on this yet – Sudowrite and others are primarily single-user experiences. Making Quill a platform for *collaborative* creative writing (imagine an editor persona plus a human editor reviewing the same draft) could be a unique selling point.

Other minor delights could include: **integrations with grammar/style tools** (Atticus integrates ProWritingAid for advanced copy editing [36]), **distraction-free enhancements** (Quill already has Zen Mode; perhaps add typewriter sound effects or focus highlights), and **mobile writing support** (Ulysses thrives by letting users jot down ideas on the go with seamless iCloud sync – Quill could aim for a mobile app once cloud sync is in place). Each of these features addresses the holistic writing experience, ensuring Quill isn't just an "AI text generator" but a writer's daily driver.

Using the above findings, here's a prioritized feature roadmap for Quill AI, with references to which competitor highlights the importance of each and an estimate of technical complexity:

Feature	Priority	Competitor Reference	Technical Complexity Estimate
Visual Scene Board & Timeline View – Index-card style scene planning with drag-drop reordering, synced to manuscript outline. *Closes the pantser/ plotter gap.*	**High** – Essential for plot outlining.	**Scrivener** (Corkboard & Outliner) [1]; **Plottr** (Timeline & Scene Cards) [4]; **Sudowrite** (Canvas corkboard) [7]	**Medium** – Frontend UI/UX heavy (React + Tiptap), uses local DB to store scene metadata. Little new backend needed.
Hierarchical Binder & Outline Integration – A robust sidebar for chapters, scenes, acts (with nesting, drag to reorder), plus an Outline mode to view/edit all scene summaries.	**High** – Core structural tool for writers.	**Scrivener** (Binder organizes texts) [2]; **Novelcrafter** (Grid/ Outline views for acts→scenes) [37] [9]	**Medium** – Frontend and data-model work. Can extend current Dexie schema for nested documents; no server required.
Cloud Sync & Automatic Backups – Cloud-based project sync across devices with offline support, and versioned backups (with potential E2E encryption).	**High** – Protects data, multi-device use.	**Obsidian** (Vault sync with encryption) [16]; **Scrivener** (auto local backups + Dropbox sync) [14] [15]; **Atticus** (cloud-first, instant backup) [38]	**High** – Requires backend (database + auth + storage). Likely need a service (e.g. Supabase) or custom server for real-time sync and conflict handling. Significant engineering effort.

Feature	Priority	Competitor Reference	Technical Complexity Estimate
Universal Export (DOCX, EPUB, PDF) – Export manuscripts in standard formats with basic formatting and metadata for self-publishing.	**High** – Needed for publishing workflow.	**Atticus** (one-click export to EPUB, PDF, DOCX) [18] ; **Plottr** (export outlines to Word/Scrivener) [17] ; **Novelcrafter** (Export to Scrivener/Atticus) [21]	**Medium** – Frontend-driven file generation using libraries (HTML/CSS to PDF, Markdown to DOCX). Some complexity for formatting consistency. No persistent server needed (generate on client).
Integrated Story Bible (World & Character Lore) – A persistent lore database (characters, settings, items) that both the writer and AI assistants can reference to maintain consistency. Includes per-entry fields and links (e.g. character arcs, locations).	**High** – Prevents AI continuity errors in long projects.	**Sudowrite** (Story Bible for core story facts) [23] ; **NovelAI** (Lorebook with world info injection) [39] ; **Novelcrafter** (Codex with aliases/ mentions) [40]	**Medium** – Frontend form/ UI for entries and local storage. AI integration via prompt construction (could be done client-side). Could later tie into a vector DB, but initial implementation can be local/structured.
Series-Wide Memory & RAG – Support for multi-book projects: a Series Bible shared across books, and AI retrieval of details from earlier chapters/ books (via summaries or vector search) to avoid contradictions.	**Medium** – Important for series authors (after core Bible exists).	**Sudowrite** (Series Folders share chars/ world across books) [28] [29] ; **Novelcrafter** (Series Codex for recurring info) [30] ; **Reddit user** ("structured RAG" prevents forgotten facts) [22]	**High** – Likely needs backend: e.g. host a vector index of chapter embeddings for similarity search. Also requires logic to merge context from multiple files. Significant backend and some AI prompt engineering needed.
Pro Formatting & Theming – WYSIWYG book formatting options (chapter title styles, scene break symbols, etc.) with a live preview, plus output templates resembling print-ready layout (a la Vellum/Atticus).	**Medium** – Differentiator for polished output.	**Atticus** (custom chapter theme builder with fonts, images, ornamental breaks) [41] [42] ; **Vellum** (industry standard for print layout – via Atticus parity) [41]	**Medium** – Mostly frontend. Involves creating style templates and applying them in export (or in a preview viewer). Could use a JS PDF generator with styling. Moderate complexity, no heavy server component.

Feature	Priority	Competitor Reference	Technical Complexity Estimate
Collaboration & Sharing – Real-time co-authoring or editor review mode. Allow multiple users (or an invited editor) to comment and edit the manuscript concurrently, with change tracking or suggestion mode.	**Low/Med** – Strong nice-to-have, but after solo features.	**Atticus** (collaborative writing and editing in-app) [35] ; **Google Docs** (de facto standard for co-writing)	**High** – Requires significant backend (real-time sync engine, user roles, conflict resolution). Possibly build on top of Cloud Sync once that exists (e.g. operational transform or CRDT approach). Consider third-party frameworks to reduce complexity.
Goal Tracking & Analytics Dashboard – Writing targets (daily word count, chapter word goals) and statistics (reading time, frequency of certain words, etc.), with visual progress charts and streak tracking.	**Low** – Adds writer motivation value.	**Ulysses** (writing goals and progress bars); **Atticus** (gamified goal tracking) [34]	**Low** – Frontend-only feature. Compute word counts from manuscript, store goals in local DB. Uses simple charts or progress bars. Minimal technical risk, can be done once core features are stable.

1 2 3 10 11 Organize Your Scrivener Project with the Corkboard - Literature & Latte
https://www.literatureandlatte.com/blog/organize-your-scrivener-project-with-the-corkboard

4 5 6 17 Timeline - Overview - Plottr Knowledge Base
https://docs.plottr.com/article/54-timeline-overview

7 24 25 39 Sudowrite vs. NovelAI: The No-BS Guide for Fiction Writers
https://sudowrite.com/blog/sudowrite-vs-novelai-the-no-bs-guide-for-fiction-writers/

8 9 37 40 Plan Views - Plan - Novelcrafter Help
https://www.novelcrafter.com/help/docs/plan/plan-views

12 13 21 Create from Outline - Plan - Novelcrafter Help
https://www.novelcrafter.com/help/docs/plan/create-from-outline

14 15 How to Back Up Your Scrivener Projects - Literature & Latte
https://www.literatureandlatte.com/blog/how-to-back-up-your-scrivener-projects

16 Syncing Obsidian Notes across devices using Git and GitLab
https://andrewwegner.com/obsidian-gitlab-setup.html

18 19 20 34 35 36 38 41 42 Atticus Review: Is This Book Writing Software Your Writing and Formatting Solution?
https://thewritepractice.com/book-writing-software-atticus-review/

22 Memory and Consistency : r/WritingWithAI
https://www.reddit.com/r/WritingWithAI/comments/1m1dfis/memory_and_consistency/

23 What is Story Bible – Sudowrite | Documentation
https://docs.sudowrite.com/using-sudowrite/1ow1qkGqof9rtcyGnrWUBS/what-is-story-bible/jmWepHcQdJetNrE991fjJC

26 27 What does the summary/outline do to the AI generation? - Plan - Novelcrafter Help
https://www.novelcrafter.com/help/faq/plan/outline-impact

28 29 Series Support – Sudowrite | Documentation
https://docs.sudowrite.com/using-sudowrite/1ow1qkGqof9rtcyGnrWUBS/series-support/3vfbZPCB1ANLm75FXmJf28

30 31 Series Codex - Codex - Novelcrafter Help
https://www.novelcrafter.com/help/docs/codex/series-codex

32 Which AI model should I use? - Sudowrite | Documentation
https://docs.sudowrite.com/using-sudowrite/1ow1qkGqof9rtcyGnrWUBS/which-ai-model-should-i-use/veMq9xRH6KLCQPFm5XkQx7

33 [AINews] Sora pushes SOTA - Buttondown
https://buttondown.com/ainews/archive/ainews-sora-pushes-sota/