Research Report: Blueprint for an Advanced Obsidian Template Generation Plugin

DATE: 2025-09-04

Executive Summary

This report provides a comprehensive research analysis intended to serve as the foundational knowledge base for developing an advanced template generation plugin for Obsidian. The research objective was to cover all necessary aspects for a plugin blueprint focusing on AI integration, dynamic forms, and broad market appeal, including advanced concepts such as Prophetic Context Prompt Engineering (PCPE), hallucination prevention, and multi-domain template support. The findings indicate a significant opportunity to create a market-leading plugin that synthesizes the strengths of existing tools while introducing novel capabilities. Successful plugins within the Obsidian ecosystem, such as Templater and Dataview, thrive by enabling users to automate repetitive tasks, enforce consistency, and build complex, interconnected knowledge systems. The proposed plugin should build upon this foundation by integrating a sophisticated, scriptable templating engine with an intuitive dynamic form builder. This combination will allow users to create interactive and context-aware note-generation workflows, moving beyond the static prompts of current solutions.

Furthermore, the integration of Artificial Intelligence presents a transformative opportunity. Drawing inspiration from the AI-powered “Interpreter” in the Obsidian Web Clipper, the plugin can offer features like content summarization, data extraction, and automated content creation. A critical component of this AI integration must be a robust framework for **hallucination prevention**. Techniques such as Retrieval-Augmented Generation (RAG), which can ground AI responses in the user’s own vault content, and Chain-of-Verification (CoVe) are essential for building user trust and ensuring the reliability of generated information. The report also explores the forward-looking concept of PCPE, envisioning an AI that not only generates text but also acts as an intelligent workflow orchestrator, anticipating user needs and dynamically selecting appropriate tools or sub-templates. To achieve broad appeal, the plugin must support a wide range of use cases, mirroring the versatility of popular productivity platforms like Notion and Airtable, enabling users to create templates for project management, content calendars, personal CRMs, and more. This report concludes with actionable recommendations for the technical architecture, development workflow, and publishing strategy, providing a clear path toward creating a powerful and indispensable tool for the Obsidian community.

1.0 Foundational Analysis: The Obsidian Plugin Ecosystem

A thorough understanding of the Obsidian plugin ecosystem is paramount to designing a new plugin that is both technically sound and resonant with the user base. This ecosystem is characterized by its vibrancy, extensibility, and a strong community focus on personal knowledge management (PKM). The success of any new plugin hinges on its ability to integrate seamlessly into this environment, adhere to established development conventions, and address unmet user needs. This section analyzes the core principles of plugin development, identifies the characteristics of the most successful community plugins, and assesses the market opportunity by looking at trends in the broader productivity software landscape.

1.1 Core Development Principles and Architecture

The technical foundation for Obsidian plugin development is well-documented and accessible, primarily utilizing TypeScript for its type safety and robust integration with the Obsidian API. The official documentation recommends starting with the official sample plugin, which provides a boilerplate structure including essential configuration files like `manifest.json` and a basic TypeScript setup. The development environment requires Node.js and Git, with a standard workflow involving cloning the sample repository, installing dependencies via `npm install`, and compiling TypeScript to JavaScript using a bundler like esbuild. A critical best practice emphasized throughout the official guidance is the use of a separate, dedicated vault for development to prevent accidental data loss or corruption in a user's primary vault.

The architectural core of an Obsidian plugin is the `Plugin` class, which extends a base `Component` class and provides lifecycle methods such as `onload()` and `onunload()`. The `onload()` method is the entry point for initializing the plugin's features, such as registering commands, adding settings tabs, or creating ribbon icons. Key API methods include `addCommand()` for adding actions to the command palette, `addSettingTab()` for creating user-configurable options, and `addRibbonIcon()` for adding quick-access buttons to the main interface. For maintainability and scalability, best practices suggest organizing code by separating concerns into different files—for instance, managing commands in a `commands.ts` file and settings in a `settings.ts` file—and using dependency injection to decouple components. To enhance the development workflow, community-developed tools like `obsidian-plugin-cli` can automate the process of building and reloading the plugin within the test vault, significantly improving iteration speed.

The `manifest.json` file serves as the metadata descriptor for the plugin and is crucial for its identification and management by Obsidian. It contains required fields such as a unique `id`, a human-readable `name`, a `version` number following semantic versioning, and the `minAppVersion` of Obsidian the plugin is compatible with. For plugins that utilize Node.js or other desktop-specific APIs, the `isDesktopOnly` flag must be set to `true`. This manifest is not only vital for local development but is a core requirement for submission to the community plugin marketplace, where it informs users about the plugin's purpose, author, and compatibility.

1.2 Hallmarks of Successful Plugins

Analysis of the Obsidian plugin marketplace reveals a clear pattern among the most popular and highly-regarded plugins. Success is not merely a function of technical complexity but of how effectively a plugin empowers users to customize and automate their workflows. Two plugins that consistently emerge as exemplars are **Templater** and **Dataview**. Templater extends Obsidian's native templating capabilities by introducing a powerful JavaScript-based scripting engine. This allows users to create dynamic templates that can automatically insert dates, prompt for user input, and execute complex logic upon note creation. Its success lies in its ability to eliminate repetitive manual tasks and enforce a consistent structure across a user's vault, which is fundamental to effective knowledge management.

Dataview, similarly, transforms how users interact with their notes by allowing them to query metadata embedded in their files (typically in YAML frontmatter) and render the results as dynamic lists, tables, or other visualizations. When used in conjunction, Templater and Dataview create a powerful synergy: Templater automates the creation of structured notes with consistent metadata, and Dataview allows users to aggregate, analyze, and visualize the information contained within those notes. This combination effectively turns an Obsidian vault into a personal, interactive database. The core lesson from their success is that users value tools that provide leverage—plugins that enable them to build systems and workflows far more sophisticated than what the base application offers.

Other successful plugins, such as Kanban, Calendar, and Tasks, follow a similar principle by introducing established productivity paradigms directly into the Obsidian interface, reducing the need for users to switch between different applications. These plugins succeed because they solve tangible problems related to organization, automation, and information retrieval, making the user's knowledge base more active and useful.

1.3 Market Opportunity and Broad Appeal

To ensure the proposed plugin has broad appeal, it is instructive to look beyond the Obsidian ecosystem to wider trends in productivity software, particularly platforms like Notion and Airtable. These tools have gained immense popularity due to their flexibility, allowing users to build custom solutions for a vast array of needs. Trending template types on these platforms provide a clear signal of market demand. Popular categories consistently include project management (especially Agile sprint planning), content calendars for marketing and social media, personal and team task management, and lightweight Customer Relationship Management (CRM) systems. These templates are not just static documents; they are interactive systems featuring relational databases, automated workflows, and collaborative dashboards.

The market opportunity for a new Obsidian plugin lies in bringing this level of structured, workflow-oriented functionality into the local-first, privacy-focused environment of Obsidian. While plugins like Dataview provide querying capabilities, the process of creating and managing the underlying structured data can be cumbersome. A new plugin that simplifies the creation of complex, interconnected notes through dynamic forms and advanced templates could capture this market. For example, a user could invoke a “New Project” template that presents a form to input the project name, deadline, stakeholders, and initial tasks. Upon submission, the plugin would generate not only a main project page but also individual task notes, a project entry in a master timeline, and links to relevant contact notes, all automatically structured with the correct metadata for Dataview queries. By supporting these multi-domain use cases—from a student's assignment tracker to a small business's sales pipeline—the plugin can attract a much broader audience than one focused solely on academic note-taking. The key is to provide the building blocks for users to create their own “Airtable-like” or “Notion-like” systems within the comfort and security of their own vault.

2.0 Core Feature Deep Dive: Advanced Template Generation

The core of the proposed plugin is a template generation engine that surpasses existing solutions in interactivity, flexibility, and power. This requires moving beyond static text replacement and simple scripting to a system that can generate complex, structured content based on dynamic user input and predefined logic. This section details the vision for this advanced engine, focusing on the implementation of dynamic forms for a superior user experience and the necessity of supporting diverse, multi-domain template structures.

2.1 Beyond Static Templates: The Need for Dynamic Content

Obsidian's core template functionality and even the popular Templater plugin represent significant steps toward workflow automation. However, their primary interaction model often relies on either pre-defined, non-interactive content insertion or simple text-based prompts. While Templater's JavaScript execution environment is powerful, creating complex, interactive workflows requires significant scripting knowledge, which can be a barrier for many users. The next evolution in template generation is to provide a more accessible and powerful way to handle dynamic content creation. This involves a system where templates are not just passive documents but active agents that can guide the user through the data entry process, validate input, and conditionally generate content based on the information provided.

The proposed plugin will achieve this by treating templates as executable blueprints for note creation. Instead of just inserting today's date, a template could present a calendar widget. Instead of a simple text prompt for "tags," it could offer a multi-select dropdown populated with existing tags from the vault. This approach reduces friction, minimizes errors, and ensures that the data captured is structured, consistent, and immediately usable by other plugins like Dataview. The engine will need to support a rich set of dynamic variables, custom functions, and logical operators that can be easily composed by users without requiring them to be proficient programmers. This bridges the gap between the simplicity of basic templates and the high-ceiling complexity of full-fledged scripting, offering a "low-floor, high-ceiling" experience that appeals to both novice and power users.

2.2 Implementing Dynamic Forms for User Interaction

The cornerstone of this advanced templating system is the concept of **dynamic forms**. When a user invokes a template, instead of the note being created immediately, the plugin will render an interactive form UI within an Obsidian modal or view pane. This form will be defined within the template file itself, likely using a simple, declarative syntax like JSON or a YAML-based schema. This approach separates the form's structure from its presentation and logic, making templates easier to create, share, and maintain. The form builder should support a comprehensive range of input types, including text fields, number inputs, date pickers, toggles, dropdown selectors, and multi-select lists.

Crucially, these forms must be dynamic, meaning they can change in real-time based on user input. This is achieved through **conditional logic** and branching paths. For example, if a user selects "Bug Report" from a "Note Type" dropdown, the form could dynamically display additional fields for "Steps to Reproduce" and "Expected Behavior." If they select "Feature Request," it might instead show fields for "User Story" and "Business Value." This context-aware interaction ensures that users are only presented with relevant fields, streamlining the data entry process and improving data quality. The implementation will also incorporate real-time validation to provide immediate feedback, such as checking if a required field is empty or if an entered URL is in a valid format. By leveraging JavaScript to render these forms from a schema, the plugin can provide a rich, app-like user experience that feels deeply integrated into Obsidian, making the process of creating structured notes both efficient and intuitive.

2.3 Multi-Domain Template Support

To achieve the broad appeal identified in the market analysis, the plugin must be designed to support a wide variety of use cases, or "domains." This means the templating and dynamic form engine must be flexible enough to create structures for vastly different purposes. The goal is to empower users to build templates for anything from a simple daily journal entry to a complex project management dashboard or a CRM for tracking client interactions. This versatility is achieved by ensuring the form and template engine is agnostic to the content it is handling.

For example, a template for a **Content Calendar** might include a form with fields for "Title," "Topic," "Status" (e.g., Idea, Drafting, Published), "Publish Date," and "Target Platform." Upon submission, it would create a new note with this information in the YAML frontmatter and could even be configured to automatically place the note in a specific folder based on its status. A template for a **Book Note** could have fields for "Author," "Rating," "Genre," and a text area for a summary, generating a note that is perfectly formatted for a personal library database. A **Meeting Note** template could prompt for "Attendees," "Date," and "Agenda," and then generate a note with pre-defined sections for "Discussion Points," "Action Items," and "Decisions." By providing robust building blocks—diverse form fields, conditional logic, and the ability to generate multiple files or modify existing ones—the plugin gives users the power to model any workflow they can envision, truly fulfilling the promise of a personalized knowledge management system.

3.0 Advanced Feature Integration: Artificial Intelligence

The integration of Artificial Intelligence (AI) represents the most significant opportunity for the plugin to deliver transformative value. By embedding LLM capabilities directly into the templating workflow, the plugin can move beyond simple automation to intelligent augmentation, assisting users in creating, processing, and refining their knowledge. This section outlines a three-pronged strategy for AI integration, focusing on practical content augmentation, essential risk mitigation through hallucination prevention, and a forward-looking vision for AI as a workflow orchestrator.

3.1 AI-Powered Content Augmentation

The initial application of AI within the plugin should focus on tangible, high-value tasks that augment the user’s content creation process. A direct model for this functionality is the “Interpreter” feature found in the official Obsidian Web Clipper. This feature allows users to apply natural language prompts to process captured web content, performing tasks like generating summaries, extracting key takeaways, or translating text. The proposed plugin will integrate a similar capability, but with a much broader scope, applying it to any content within the user’s vault or generated through a dynamic form.

This AI functionality would be exposed as a set of functions or modules available within the templating engine. For instance, a user creating a template for meeting notes could include an AI-powered step that takes the raw transcript of the meeting (pasted into a form field) and automatically generates a concise summary and a list of action items. Another use case could be a research note template where a user provides a link to an academic paper, and the AI fetches the abstract, extracts key concepts, and suggests relevant tags based on the content. This feature would be powered by user-provided API keys for various LLM providers (like OpenAI or Anthropic), ensuring that the user remains in control of their data and service usage. The AI’s role here is to act as an intelligent assistant, reducing manual effort and surfacing insights that might otherwise be missed.

3.2 Mitigating Risk: Hallucination Prevention Strategies

While the potential of AI is immense, its primary risk in a knowledge management context is **hallucination**—the generation of plausible but factually incorrect or nonsensical information. For a tool designed to be a “second brain,” such unreliability is unacceptable. Therefore, a core design principle of the plugin’s AI integration must be the implementation of robust hallucination prevention techniques. The goal is to create a trustworthy AI assistant that grounds its outputs in verifiable facts.

The most promising technique for this is **Retrieval-Augmented Generation (RAG)**. Instead of relying solely on its pre-trained knowledge, the LLM would be provided with relevant context retrieved directly from the user’s Obsidian vault. When a user asks the AI to “summarize my notes on quantum computing,” the plugin would first perform a search within the vault for relevant notes and then feed that specific content to the LLM as the basis for its summary. This anchors the AI’s response in the user’s own knowledge base, dramatically reducing the likelihood of it inventing information.

Other techniques will be employed to further enhance reliability. **Chain-of-Verification (CoVe)** prompting can be used for complex queries, forcing the model to break down its response into a series of verifiable steps and to double-check its own reasoning before presenting a final answer. Simple but effective methods like **“According to…” prompting** can be used to instruct the model to base its response on a specific source provided by the user. By building these safety mechanisms into the core of the AI feature, the plugin can foster user trust and ensure that it serves as a reliable tool for thought, not a source of misinformation.

3.3 The Next Frontier: Prophetic Context Prompt Engineering (PCPE)

Looking beyond immediate content augmentation, the plugin can pioneer a more advanced form of AI interaction described as **Prophetic Context Prompt Engineering (PCPE)**. This concept, which appears to be an evolution of ideas like Model Context Protocol (MCP), envisions the AI not as a passive text generator but as an active, decision-making “brain” for user workflows. Instead of the user explicitly telling the AI what to do at every step, the AI would anticipate the user’s needs based on context and proactively orchestrate a series of actions.

In this paradigm, the AI acts as a central dispatcher. For example, a user might start with a simple prompt like “Draft a project plan for the Q4 product launch.” A traditional AI would generate a block of text. An AI powered by PCPE would recognize the intent and initiate a workflow. It might first invoke a dynamic form to ask clarifying questions about the project scope and key deadlines. Based on the answers, it could then select and execute a series of specialized sub-templates: one to create the main project hub page, several others to generate initial task notes for team members, and another to create an entry in a master project timeline. It could even query the user’s vault to find notes from previous product launches and suggest incorporating relevant lessons learned. This “prophetic” capability—the ability to predict and automate the optimal sequence of actions—transforms the AI from a simple tool into a true workflow partner, enabling a level of automation and intelligence far beyond what is currently possible in personal knowledge management tools.

4.0 Blueprint Recommendations and Conclusion

The successful development and launch of this advanced templating plugin require a clear technical strategy, a well-defined development process, and a thoughtful approach to community engagement. This final section synthesizes the research findings into a set of actionable recommendations for the plugin’s blueprint, covering its architecture, the development workflow, and the process for publishing it to the Obsidian community.

4.1 Technical Blueprint and Development Workflow

The recommended technical stack for the plugin is TypeScript, leveraging its strong typing to ensure code quality and maintainability, which is the standard for modern Obsidian plugin development. The architecture should be modular, with clear separation of concerns. The core templating engine, the dynamic form rendering module, and the AI integration service should be developed as distinct components to facilitate testing and future enhancements. The dynamic forms should be rendered from a declarative JSON or YAML schema defined within the template files, allowing users to easily create and share complex UIs without writing rendering code.

The development process should begin by cloning the official Obsidian sample plugin to establish the basic project structure. A dedicated, non-critical vault must be used for all testing purposes. To accelerate development, the **Hot-Reload plugin** should be utilized to enable automatic reloading of the plugin upon code changes, bypassing the need for manual restarts of the Obsidian application. The development workflow will involve iterative cycles of coding in TypeScript, compiling to JavaScript using a modern bundler like esbuild, and testing the functionality within the dedicated vault. Adherence to the architectural best practice of separating UI, logic, and data management will be critical for managing the complexity of a feature-rich plugin like this.

4.2 Publishing and Community Engagement Strategy

Once the plugin reaches a stable, feature-complete state, it can be submitted to the official Obsidian community plugin marketplace. The submission process is managed via GitHub. It requires creating a public GitHub repository for the plugin's source code. A GitHub release must be created with a tag that exactly matches the `version` number specified in the `manifest.json` file. This release must include the compiled `main.js` file, the `manifest.json`, and any necessary `styles.css` files as release assets. Following this, a pull request is made to the official `obsidian-releases` repository, adding the plugin's manifest to the community list.

Prior to submission, it is imperative to thoroughly review and adhere to all of Obsidian's plugin guidelines. This includes removing all boilerplate code from the sample plugin, writing a clear and concise description for the `manifest.json` that is action-oriented, and ensuring the plugin does not use deprecated APIs or insecure practices like using `innerHTML`. A comprehensive `README.md` file is also essential, providing clear instructions on how to install, configure, and use the plugin, complete with examples of the template syntax and dynamic form schemas. Engaging with the community early, perhaps through beta testing with a small group of users via the Obsidian forums or Discord, can provide invaluable feedback for refining features and fixing bugs before the official launch.

4.3 Concluding Analysis

The research conducted for this report overwhelmingly indicates a powerful opportunity to create a next-generation templating plugin for Obsidian. By combining a highly flexible, scriptable templating engine with an intuitive, schema-driven dynamic form builder, the plugin can empower users to automate the creation of structured data in a way that is currently complex and inaccessible to many. This alone would constitute a significant contribution to the ecosystem. However, the integration of AI, particularly when implemented with a steadfast commitment to reliability through robust hallucination prevention techniques like RAG, elevates the plugin from a useful utility to an indispensable tool for thought. The forward-looking vision of PCPE provides a roadmap for future development, promising an intelligent assistant that can orchestrate entire workflows with minimal user intervention. By addressing a clear user need for more sophisticated automation and providing a trustworthy AI partner, this plugin is well-positioned to become an essential, high-impact component of the modern personal knowledge management toolkit, with the potential for broad appeal both within and beyond the existing Obsidian community.

References

Build a plugin - Obsidian Hub (https://docs.obsidian.md/Plugins/Getting+started/Build+a+plugin)
Manifest - Obsidian Hub (https://docs.obsidian.md/Reference/Manifest)
Obsidian TypeScript API Reference - Obsidian Hub (https://docs.obsidian.md/Reference/TypeScript+API/Plugin)
Obsidian Web Clipper – Introduction - Obsidian Hub (https://obsidian.md/clipper)
Journey developing an Obsidian plugin - Part 2: Improving the architecture, basic error handling and... - DEV Community (https://dev.to/bjarnerentz/journey-developing-an-obsidian-plugin-part-2-improving-the-architecture-basic-error-handling-and-5aa6)
How to create your own Obsidian plugin - Medium (https://phibr0.medium.com/how-to-create-your-own-obsidian-plugin-53f2d5d44046)
obsidian-plugin-tutorial - GitHub (https://github.com/flolu/obsidian-plugin-tutorial)
A more streamlined development workflow for Obsidian plugins - DEV Community (https://dev.to/lukasbach/a-more-streamlined-development-workflow-for-obsidian-plugins-5hm5)
How to get started with developing a custom plugin - Obsidian Forum (https://forum.obsidian.md/t/how-to-get-started-with-developing-a-custom-plugin/8157)
How to create an Obsidian Plugin in 30 minutes - YouTube (https://www.youtube.com/watch?v=AgXa03ZxJ88)
Submission requirements for plugins - Obsidian Hub (https://docs.obsidian.md/Plugins/Releasing/Sub-

mission+requirements+for+plugins)

Obsidian Plugins - Obsidian (https://obsidian.md/plugins)

These Obsidian Plugins Changed My Workflow - Medium (https://medium.com/@theo-james/these-obsidian-plugins-changed-my-workflow-e90f9f4a15d6)

A guide to plugin development for Obsidian - Obsidian Forum (https://forum.obsidian.md/t/a-guide-to-plugin-development-for-obsidian/26290)

How I Built Two Obsidian Plugins While Kiro AI Did Most of the Work - Stephan Miller (https://www.stephanmiller.com/how-i-built-two-obsidian-plugins-while-kiro-ai-did-most-of-the-work/)

Shaping Obsidian’s Tomorrow: How Obsidian Can Embrace AI and Stay True to Its Roots - Obsidian Forum (https://forum.obsidian.md/t/shaping-obsidian-s-tomorrow-how-obsidian-can-embrace-ai-and-stay-true-to-its-roots/92220)

How I Built My Second Brain with Obsidian - Medium (https://medium.com/@kevin.tsyen/how-i-built-my-second-brain-with-obsidian-54edad2ecc44)

Obsidian build plugin bundle implementation workflow advice - Obsidian Forum (https://forum.obsidian.md/t/obsidian-build-plugin-bundle-implementation-workflow-advice/25973)

Obsidian Web Clipper - Obsidian Hub (https://help.obsidian.md/web-clipper)

Supercharge your knowledge capture workflow with the Obsidian Web Clipper - Sébastien Dubois (https://www.dsebastien.net/supercharge-your-knowledge-capture-workflow-with-the-obsidian-web-clipper/)

Obsidian Web Clipper - Obsidian Forum (https://forum.obsidian.md/t/obsidian-web-clipper/53123)

New Obsidian Web Clipper is a Game Changer - Stephan Miller (https://www.stephanmiller.com/new-obsidian-web-clipper-is-a-game-changer/)

Obsidian Web Clipper browser extension - Geeky Gadgets (https://www.geeky-gadgets.com/obsidian-web-clipper-browser-extension/)

Obsidian Web Clipper is better than I what I thought - Medium (https://beingpax.medium.com/obsidian-web-clipper-is-better-than-i-what-i-thought-cc7d36ad3936)

Templater: How to add information to YAML frontmatter - Obsidian Forum (https://forum.obsidian.md/t/templater-how-to-add-information-to-yaml-frontmatter/38009)

Input the creation date of a note on creation in YAML frontmatter - Obsidian Forum (https://forum.obsidian.md/t/input-the-creation-date-of-a-note-on-creation-in-yaml-frontmatter/49530)

Format Obsidian Publish URLs with Templater and YAML - Medium (https://medium.com/@mtzfox/format-obsidian-publish-urls-with-templater-and-yaml-3eca397a5629)

Use Obsidian Templater to Automate Our Note-Taking System (Beginners) - Medium (https://amyjuanli.medium.com/use-obsidian-templater-to-automate-our-note-taking-system-beginners-f6c507258427)

Templater with new Properties feature - GitHub (https://github.com/SilentVoid13/Templater/issues/1191)

Unlock the secret to effortless note-taking with YAML in Obsidian - Geeky Gadgets (https://www.geeky-gadgets.com/unlock-the-secret-to-effortless-note-taking-with-yaml-in-obsidian/)

YAML front-matter - Assemble (https://assemble.io/docs/YAML-front-matter.html)

Submit your plugin - Obsidian Hub (https://docs.obsidian.md/Plugins/Releasing/Submit+your+plugin)

Plugin guidelines - Obsidian Hub (https://docs.obsidian.md/Plugins/Releasing/Plugin+guidelines)

How to Submit Your First Obsidian Plugin - YouTube (https://www.youtube.com/watch?v=QcZNu1TY11w)

Obsidian Plugin: Image Upload Toolkit Came to 1.0 - Medium (https://addozhang.medium.com/obsidian-plugin-image-upload-toolkit-came-to-1-0-7921c68ef22f)

Community plugins - Obsidian Hub (https://help.obsidian.md/community-plugins)

Practically Paperless with Obsidian, Episode 25: Five Use Cases for Managing My Writing in Obsidian - Jamie Rubin (https://jamierubin.net/2022/04/05/practically-paperless-with-obsidian-episode-25-five-use-cases-for-managing-my-writing-in-obsidian/)

obsidian-sample-plugin - GitHub (https://github.com/obsidianmd/obsidian-sample-plugin)

Publish Plugins - Obsidian Stats (https://www.obsidianstats.com/posts/2025-04-16-publish-plugins)
Obsidian and Task Management: Best Practices - Medium (https://medium.com/@theo-james/obsidian-and-task-management-best-practices-e7abc777f0a5)
A Roundup of the Best Obsidian Plugins in 2024 - The Sweet Setup (https://thesweetsetup.com/a-roundup-of-the-best-obsidian-plugin-in-2024/)
Obsidian for Worldbuilding - Obsidian Tavern (https://obsidiantavern.com/obsidian-worldbuilding/)
The Must-Have Obsidian Plugins - Sébastien Dubois (https://www.dsebastien.net/2022-10-19-the-must-have-obsidian-plugins/)
5 Essential Obsidian Plugins for Supercharging Your Productivity - Mixa Blog (https://blog.mixa.site/blog/5-essential-obsidian-plugins-for-supercharging-your-productivity)
Workflows & Automations Notion Templates - Notion (https://www.notion.com/templates/category/workflows-automations?srsltid=AfmBOoqRDWyrxvU92iHZ-aaNJYkAm5lxgtGSPpYQuxPTMf5AtziKe7iS)
Project Template Libraries Streamline Workflow - ones.com (https://ones.com/blog/project-template-libraries-streamline-workflow/)
Notion Templates - The Knowledge Academy (https://www.theknowledgeacademy.com/blog/notion-templates/)
Notion template that works like Airtable? - Reddit (https://www.reddit.com/r/Notion/comments/mhmueh/notion_template_that_works_like_airtable/)
Create a Productivity Workflow That Works for You - Harvard Business Review (https://hbr.org/2020/01/create-a-productivity-workflow-that-works-for-you)
Airtable vs Notion - Slam Media Lab (https://www.slammedialab.com/post/airtable-vs-notion)
Top 10 free personal productivity templates in Notion - Notion (https://www.notion.com/templates/collections/top-10-free-personal-productivity-templates-in-notion?srsltid=AfmBOoqz3F5jOD_rVXzGTFIOhZv3jZ2it9wtW0Tn0_AcU-2mcxt6rqn5)
Airtable Templates - PMP Certification Course (https://www.pmpcertificationcourse.com/blog/airtable-templates)
Airtable Templates for a Better Workflow Management - Shift (https://shift.com/guides/productivity/airtable-templates-for-a-better-workflow-management/)
Notion Project Management - NoteForms (https://noteforms.com/resources/notion-project-management)
Beginner’s Guide to Creating Dynamic Forms with JavaScript - Medium (https://medium.com/@AlexanderObregon/beginners-guide-to-creating-dynamic-forms-with-javascript-10aef6a8843d)
Creating Dynamic Form Fields with HTML, CSS, and JavaScript - DEV Community (https://dev.to/getsmartwebsite/creating-dynamic-form-fields-with-html-css-and-javascript-a-comprehensive-guide-46bn)
Dynamic Forms with Stimulus - thoughtbot (https://thoughtbot.com/blog/dynamic-forms-with-stimulus)
Dynamic forms - Angular (https://angular.dev/guide/forms/dynamic-forms)
Dynamically create UI components from user uploaded templates - Stack Overflow (https://stackoverflow.com/questions/62808737/dynamically-create-ui-components-from-user-uploaded-templates)
Main Concepts of Dynamic Forms Modeling - Progress Blogs (https://www.progress.com/blogs/main-concepts-dynamic-forms-modeling)
Form Design for UX/UI Designers - Formsort (https://formsort.com/article/form-design-for-ux-ui-designers/)
Dynamically Create Forms from JSON Schema - SurveyJS (https://surveyjs.io/stay-updated/blog/dynamically-create-forms-from-json-schema)
Open Source Form Builder - Budibase (https://budibase.com/blog/open-source-form-builder/)
How to Build a Dynamic Form App - Stackademic (https://blog.stackademic.com/how-to-build-a-dynamic-form-app-4a9f61c30440)
Advanced Prompt Engineering for Reducing Hallucination - Medium (https://medium.com/@bijit211987/advanced-prompt-engineering-for-reducing-hallucination-bb2c8ce62fc6)
9 Prompt Engineering Methods to Reduce Hallucinations - God of Prompt (https://www.godofprompt.ai/

blog/9-prompt-engineering-methods-to-reduce-hallucinations-proven-tips?srsltid=AfmBOoqkKEw45-0kNrTN1UHbFSfm3WoIbQrScDwGWAD6c7UXOiWSDS8c)

The Importance of Prompt Engineering in Preventing AI Hallucinations - AlfaPeople (https://alfapeople.com/importance-of-prompt-engineering-preventing-ai-hallucinations/)

AI Strategies Series: 7 Ways to Overcome Hallucinations - FactSet (https://insight.factset.com/ai-strategies-series-7-ways-to-overcome-hallucinations)

Three Prompt Engineering Methods to Reduce Hallucinations - PromptHub (https://www.prompthub.us/blog/three-prompt-engineering-methods-to-reduce-hallucinations)

AI hallucinations: Why they happen and how to prevent them - Zapier (https://zapier.com/blog/ai-hallucinations/)

What is AI Hallucination? - DigitalOcean (https://www.digitalocean.com/resources/articles/ai-hallucination)

How to Prevent AI Hallucinations in 2025 - Enkrypt AI (https://www.enkryptai.com/blog/how-to-prevent-ai-hallucinations)

Prompt Engineering Techniques to Improve LLMs Performance - Level Up Coding (https://levelup.gitconnected.com/prompt-engineering-techniques-to-improve-llms-performance-2f06cbbc78f5)

Anthropic just revealed their internal prompt engineering guide - Reddit (https://www.reddit.com/r/PromptEngineering/comments/1n08dpp/anthropic_just_revealed_their_internal_prompt/)

Evolving Prompt Engineering for Smarter Automation with MCP and n8n - Medium (https://medium.com/@tam.tamanna18/evolving-prompt-engineering-for-smarter-automation-with-mcp-and-n8n-ace4a840e75d)

Prompt Engineering is Dead and Context Engineering is Already Obsolete - OpenAI Community (https://community.openai.com/t/prompt-engineering-is-dead-and-context-engineering-is-already-obsolete-why-the-future-is-automated-workflow-architecture-with-llms/1314011?page=3)

What is the most user-friendly way to create workflows with prompt engineering? - Reddit (https://www.reddit.com/r/n8n/comments/1hb5wli/what_is_the_most_userfriendly_way_to_create/)

Context Engineering vs. Prompt Engineering - Medium (https://medium.com/data-science-in-your-pocket/context-engineering-vs-prompt-engineering-379e9622e19d)

Transforming Prompts into Production Code - YouTube (https://www.youtube.com/watch?v=XwWCFINXIoU)

A Developer’s Guide to Prompt Engineering - Stack AI (https://www.stack-ai.com/blog/guide-to-prompt-engineering)

Prompt Engineering is Dead and Context Engineering is Already Obsolete - OpenAI Community (https://community.openai.com/t/prompt-engineering-is-dead-and-context-engineering-is-already-obsolete-why-the-future-is-automated-workflow-architecture-with-llms/1314011)

A Developer’s Guide to Prompt Engineering - Accelirate (https://www.accelirate.com/prompt-engineering-guide-for-developers/)

From Templates to Toolchains: Prompt Engineering Trends 2025 Explained - Refonte Learning (https://www.refontelearning.com/blog/from-templates-to-toolchains-prompt-engineering-trends-2025-explained)