ChatGPT

Empowering Non-Developers with AI: From *Vibe Coding* to Visual Documentation

Authors: Dinis Cruz; ChatGPT Deep Research

Introduction

In recent years, a new wave of non-developers has begun creating software applications by *"vibe coding"* – using AI assistants to write code based on natural language descriptions rather than manual programming [1]. This approach, popularized by AI expert Andrej Karpathy, allows people with ideas but little coding experience to **build prototypes by simply describing their vision** and letting the AI handle the heavy lifting of writing code [1] [2]. Thanks to this democratization of coding, what once took an expert programmer days or weeks can sometimes be achieved in just hours of iterative dialogue with an AI [2]. Entrepreneurs and product thinkers are seizing this opportunity to **rapidly bring their ideas to life**, identifying market gaps and using AI-driven tools to fill them.

However, as these *vibe coders* turn rough prototypes into more mature solutions, they encounter new challenges. Many *non-technical* creators eventually face the realities of software development: implementing non-functional requirements (performance, security, maintainability), managing source control and testing, and clearly communicating the system's design. As one observer noted, vibe coding (like no-code approaches) is fantastic for quick prototypes, but *"the moment you want more control over your app, you need to… clearly and unambiguously specify what you want it to do. The best way to do that is coding"* [3]. In other words, to evolve a prototype into a robust product, collaboration with professional developers and **clear technical communication** become essential. Yet many of these new creators lack a background in technical documentation or architecture communication.

This white paper introduces a workflow to bridge that gap by leveraging two cutting-edge Generative AI tools: **large language models (LLMs) for automated documentation**, and **Google's NotebookLM for instantaneous visualization of that documentation**. In simple terms, we show how a vibe coder can *ask their AI assistant to generate a technical brief* of what they've built, and then with a few clicks, *transform that brief into a polished infographic and slide deck* using NotebookLM. This approach empowers non-developers to better understand their own creations and to communicate them professionally – whether to potential investors, collaborators, or software engineers who will help take the project further.

Authorship note: The document you are reading is itself a product of this AI-augmented workflow. It was co-created by Dinis Cruz and an advanced AI assistant (ChatGPT Deep Research), derived from an initial "brain dump" voice memo. This demonstrates how modern AI can turn unstructured ideas into a structured, formal paper – the very process we advocate here.

Vibe Coding and the Rise of AI-Assisted Development

Vibe coding refers to coding in an intuitive, almost conversational manner with the help of AI. Instead of writing every line of code, the creator describes what they want and iteratively guides an AI to produce the code [1]. Karpathy humorously remarked that *"the hottest new programming language is*

English," meaning that describing your program's intent in plain English can be as effective as writing the code yourself [4]. In vibe coding, you're not dealing with syntax or debugging in the traditional sense – you're *"steering the thing until it feels right,"* as one early practitioner put it [5]. A vibe coder might say, *"I need a simple web page that lets users upload a photo and then applies a filter,"* and the AI will generate the necessary HTML/CSS/JavaScript for that functionality. They then refine the result by telling the AI what to tweak or fix, in a tight human-AI loop of feedback [6] [7].

This paradigm has made software creation accessible to a whole new class of innovators. People with domain expertise or business savvy but no formal programming training can now develop prototypes by conversing with AI. Industry experts note that this **lowers the barrier to entry** – individuals can *"bring their ideas to life"* without months of learning to code [2]. Tools like Cursor, Windsurf, Replit, and others provide specialized environments for vibe coding, where the user interfaces focus on natural language prompts and AI-generated suggestions rather than manual code editing [5]. The result is an explosion of creativity: critical thinkers and entrepreneurs are tackling inefficiencies in various industries with custom apps and solutions developed via AI assistance.

Challenges emerge as projects grow: The ease of vibe coding a prototype can belie the complexity that lurks in scaling that prototype. Once a basic application is up and running, creators often find themselves at a crossroads. On one hand, they have achieved a functional MVP (Minimum Viable Product) quickly; on the other hand, they may not fully understand the code that was generated or how to improve it further. Important software engineering practices – such as writing tests, handling edge cases, optimizing performance, ensuring security, using version control, and planning an architecture that can evolve – become critical for continued success. These *non-functional requirements* and engineering best practices are areas where non-developers can easily get stuck.

Equally important is the challenge of **communication and documentation**. A vibe-coded project might lack clear documentation of its features and inner workings, since it was built through conversational interactions rather than carefully crafted by a developer writing design docs. When the creator needs to bring in others – e.g. to get feedback, to on-board a software engineer, or to pitch the concept to stakeholders – they need a way to explain *what has been built so far* and *what is planned next*. For someone without a technical writing background, producing a coherent technical brief or architecture diagram can be daunting. This is where AI can again assist: by *auto-generating documentation and visuals* that capture the essence of the project.

Leveraging LLMs for Automatic Documentation

Modern LLMs like ChatGPT (especially in advanced "deep research" modes) can be used as powerful writing assistants. They can digest raw information – code, error logs, or even plain-language notes – and produce organized explanations. To help our non-developer creators, the first step is to use their coding assistant (or another AI like ChatGPT) to **generate a technical brief/debrief of the project**. This brief is essentially a summary and explanation of what the app does, how it works (at a high level), and any notable technical details.

For example, after a coding session, the creator might prompt the AI with something like: *"Explain what we just built. Provide a technical overview of the application, including its key features, the tech stack or components used, how the main functionalities are implemented, and any important assumptions or future work."* From a few paragraphs of such prompting (and the context of the code it just helped write), the LLM can produce a structured description of the app. It might outline the **features**, describe the **architecture** (e.g. "frontend in React, backend using an API endpoint, data stored in X, etc."), and list what was achieved in the session versus what remains TODO.

Crucially, this does not require the non-developer to fully understand every line of code – the AI will infer the purpose of the code it wrote and articulate it in layman or semi-technical terms. The *vibe coder* thus obtains a *"debrief"* of their own project, which can greatly enhance their understanding and highlight gaps or next steps. This brief should be saved as a Markdown or PDF document (for easy handling) and ideally checked into version control alongside the code (establishing a habit of documentation).

Tip: This technical brief could even originate from a voice note. Many innovators find it easier to *speak* their ideas or retrospectives. One effective workflow is to record a voice memo describing what the app does or what was accomplished in a session, then use a transcription service (e.g. Otter.ai) to convert it to text, and finally have an AI like ChatGPT clean up and structure it. In fact, *"by leveraging ChatGPT to transform voice notes into structured documents, you can capture ideas effortlessly while speaking and have them automatically converted into organized, formatted text ready for use in reports or presentations"* [8]. The AI can remove filler words, correct transcription errors, and organize the content into coherent paragraphs. For instance, one suggested prompt for ChatGPT is to *"correct obvious transcription errors, fix grammar and sentence structure while preserving meaning, remove filler words and verbal tics, and organize related thoughts into coherent paragraphs"* [9]. In practice, this means you can pour out your thoughts freely in a voice memo and let the AI turn it into a polished write-up.

Using an LLM for documentation in this way turns a *messy brain-dump* into a structured deliverable. This document will serve as the input for the next stage of our process. It's important to ensure the brief is as clear and technical as possible about the project – focusing on *what was built, how it works, and why certain decisions were made*. The richer the brief, the better the output from NotebookLM will be.

NotebookLM: AI-Powered Visualization of Your Project Brief

With a solid technical brief in hand, we turn to **Google NotebookLM** – a tool that represents the next evolution of AI-assisted productivity. NotebookLM is an AI notebook environment that can ingest your documents (notes, PDFs, etc.), analyze them, and *generate various outputs including summaries, reports, infographics, slide decks, and even audio/video overviews*. In essence, it collapses the content creation process from research to design into one tool [10]. *NotebookLM's twin superpowers allow you to first find and analyze information in your documents, and then instantly visualize that material as a presentation or graphic* [11]. Unlike some generative AI that might hallucinate details, NotebookLM grounds its output in your provided sources – it *only uses the content you give it, not general web data* [12]. This means the slides or infographics it produces will reflect the specifics of your project brief accurately.

Key features of NotebookLM we leverage:

- **Infographic Generation:** With one click, NotebookLM can create a polished **infographic** that visually summarizes the key points of your document. You can choose the format (landscape, portrait, etc.) and even provide an optional prompt to guide the style (e.g. "use a modern tech theme with my brand colors"). The AI (powered by Google's latest Gemini model for images, code-named *Nano Banana Pro*) will produce an image highlighting the core information from your text in a creative visual form [13] [14]. This is great for conveying high-level concepts or data in a single glance.

- **Slide Deck Generation:** Even more impressively, NotebookLM can generate an entire **slide deck** (a multi-slide presentation) based on your source document. This is essentially an automated way to create a PowerPoint/Google Slides-style presentation without manual design work. The slides typically include titles, bullet points, diagrams or illustrations created by the AI, and they

follow a narrative extracted from your document. It's as if a skilled designer read your brief and made a visual presentation out of it, in seconds. This feature turns dense text into a storytelling format which is easier to present and discuss with others [10] [15].

Using NotebookLM for our scenario will allow the vibe coder to **instantly visualize their project's details**. Instead of laboring to draw architecture diagrams or write slides, they get them auto-generated. This not only saves time but can reveal new ways to frame the information – the AI might draw metaphors or connections that aid understanding [14]. It's worth noting that NotebookLM works best when the input is a well-written narrative or descriptive text, rather than a list of instructions. Users have found that it excels with *stories, debriefs or focused topic summaries*, whereas giving it terse bullet lists or raw instructions is less effective. In our context, the technical brief (especially if written as a cohesive description of the project) is an ideal input.

Current limitations: As of today (December 29, 2025), NotebookLM is a relatively new tool and still evolving. Free-tier users may encounter some usage limits – for example, heavy use of the slide/infographic generator might be capped. Google has, at times, restricted these features for free users due to high demand, offering higher quotas on paid plans [16] [17]. On the free plan you might hit limits when creating multimedia materials like slides or infographics [17], whereas Pro subscribers (e.g. Google AI Pro at ~$20/month) get higher allowances. Additionally, as of now, **the generated slides are not directly editable** in NotebookLM's interface. The tool creates them autonomously, and you cannot yet tweak the slide content or design within NotebookLM after generation (you would have to export them to Google Slides or PowerPoint for editing). Also, the tool does not currently show the intermediate text or prompts it used to create the visuals – it's a one-shot generation. These limitations are widely expected to be temporary. The pace of development in AI tools is rapid, and it's reasonable to predict that soon NotebookLM (or competing services, perhaps even from OpenAI or others) will allow more interactivity, such as editing generated slides or revealing the underlying generation process. In short, **today's NotebookLM is the weakest it will ever be** – future versions will surely be more powerful and flexible. Even in its current form, it offers tremendous value, as we will see.

Step-by-Step Workflow: From Code to Slides in Minutes

Now let's walk through the recommended workflow step by step, integrating the use of LLMs and NotebookLM. This process assumes you have already built some portion of your app using vibe coding techniques and are ready to document and visualize it:

1. **Build (Vibe Coding Session):** As you develop your application idea, you use an AI coding assistant (such as Cursor, Windsurf, ChatGPT, etc.) to generate the code for your features. You guide the AI with natural language prompts and refine the output in iterations until your prototype's functionality is working. Keep notes of what features you implemented or any architectural decisions made (if not, you will recall them in the next step).

2. **Summarize (LLM Debrief):** At a suitable milestone (e.g., a feature is completed or a prototype MVP is ready), ask your LLM assistant to **create a technical brief** of what has been done. For example: *"Summarize the current state of the app we built. Describe its purpose, main features, how the code is structured, and any remaining TODOs or issues."* The AI will produce a write-up; review it and ensure it's accurate. You might need to prompt for more detail in areas that are important (e.g., *"Explain how the user authentication works in this app"*). The goal is a clear **document** that an outside reader (with some technical background) can understand.

3. **Refine & Save the Brief:** Edit the AI-generated brief as needed (this might involve clarifying certain points or adding context you know but the AI didn't mention). Keep the tone professional and descriptive. Then save this document as a Markdown (`.md`) or PDF file. Give it a version or date in the filename (e.g., *ProjectX-TechBrief-Dec29.md*). If you use a repository or cloud drive, add it there for safekeeping. This is now the source of truth for the next steps.

4. **Upload to NotebookLM:** Open **NotebookLM** (notebooklm.google.com) and create a new notebook or choose an existing one for your project. Use the "Add sources" feature to upload the technical brief document you just saved. Once uploaded, **ensure that this is the only source selected** for NotebookLM to work with. (By default, NotebookLM might have multiple sources or notebooks active; you should deselect any others. This focuses the AI on *only your content* to avoid confusion.)

5. **Generate an Infographic:** In NotebookLM's interface, switch to the *Studio* or *Visualize* tab (as applicable) and click the option to **Generate Infographic**. You may be prompted to choose an orientation (landscape, portrait, square) and a level of detail (simple or detailed). For a start, you can accept defaults. Optionally, you can provide a short additional prompt to guide the style – for instance, *"Use a modern tech infographic style with blue tones, suitable for a business pitch."* Then, initiate the generation. In a matter of seconds, NotebookLM will produce a single-image infographic that encapsulates key points from your document. Review this output. It might highlight the problem your app solves, the main features, and perhaps a diagram or graphic metaphor of the solution. If it's not to your liking, you can regenerate (possibly with a tweaked prompt). Once satisfied, **export or save the infographic** (NotebookLM typically allows downloading the image). This visual can be a powerful quick summary of your project.

6. **Generate a Slide Deck:** Next, click the **Generate Slide Deck** button. NotebookLM will now create a multi-slide presentation based on your brief. This may take a bit longer (several tens of seconds, up to a minute) as it is creating multiple slides and graphics. The result will be a set of slides – often including a title slide, agenda or overview, a few slides expanding on different aspects of the document (e.g. "Features", "Architecture", "Roadmap"), and a conclusion. **Note:** As mentioned, you cannot edit the content in NotebookLM at this stage. If a slide isn't ideal, one workaround is to regenerate a new slide deck (the content will be grounded in the same brief but the wording or graphics might change). Each generation is slightly different, so you can repeat this and perhaps get 2-3 variants. You might find one version explains a certain feature better, while another version has a nicer diagram – you can later combine these externally if needed. Download the slide deck (usually it will export to Google Slides or a PDF/PPTX format).

7. **Iterate and Combine:** If time allows, consider iterating. For example, if your brief was very lengthy, NotebookLM might have focused on certain parts and skipped others. You could create a shorter brief on a specific aspect and run a generation just for that, to get a focused set of slides on that sub-topic. Or if the style wasn't what you wanted, adjust the custom prompt (e.g., "make the slides investor-friendly with minimal text and more graphics"). Over a few runs, you will learn how to **best communicate with NotebookLM** – often, providing a narrative with clear section headings in your brief yields better-structured decks, whereas a flat narrative might cause the AI to guess slide segmentation. *NotebookLM currently thrives on well-structured, story-like input rather than direct imperative instructions*, so phrasing your document in a storytelling format (e.g., "Introduction, Problem, Solution, Features, Next Steps") can lead to a correspondingly well-structured presentation.

8. **Export and Share:** Organize your outputs. You now have a technical brief (text), an infographic image, and a slide deck file. It's good practice to **name these with version numbers or dates**

(for example, attach the date to filenames or inside the title slides) to keep track of progress over time. You might create a folder like “ProjectX Briefings” and store **Brief_v1.md**, **Brief_v1_Infographic.png**, **Brief_v1_Slides.pptx** together. These can then be shared with your friends, team, or stakeholders. By examining the slides, your non-technical colleagues might grasp the project better, and by reading the brief, a developer you bring on board can quickly get up to speed. The infographic can be a quick **teaser or summary** in an email or chat – a nice visual to spark interest.

Following this workflow, you essentially create a feedback loop where each development iteration is accompanied by updated documentation and visuals. This habit will pay off in clarity and communication. Below is an ASCII diagram summarizing the flow:

```
Non-Dev "Vibe" Coder
    |
    |-- uses AI to build app (prototype)
    v
LLM Coding Assistant
    |
    |-- generates technical brief (markdown/PDF)
    v
Project Brief Document
    |
    |-- uploaded into NotebookLM as source
    v
NotebookLM (AI Studio)
    |
    |-- creates Infographic & Slide Deck
    v
Visual Outputs (image & slides)
    |
    |-- exported and shared with team
    v
Team / Stakeholders (review and feedback)
```

Through this process, a once messy or opaque development journey becomes a well-documented project with attractive visuals.

Benefits of AI-Assisted Documentation and Visualization

Adopting this AI-augmented workflow provides several clear benefits for non-developer innovators:

- **Clarity and Understanding:** By asking an LLM to articulate the technical details, you force an explanation of the project in plain language. This often helps *you* as the creator to understand the structure and nuances of your own app better. It can highlight aspects you hadn’t considered (e.g., “Oh, the AI says we don’t have error handling for X feature yet”) and solidify your grasp on what has been built.

- **Professional Communication:** The outputs (the brief, infographic, and slides) allow you to communicate about the project in a professional format. You can approach a conversation with a

potential partner or advisor not just with enthusiasm, but with a one-pager and a deck in hand. This lends credibility and helps others engage with your idea more seriously. The slide deck, especially, provides a narrative flow that can be used in meetings or pitch sessions, while the technical brief can be shared for offline reading or as an appendix for those who want details.

- **Time and Cost Efficiency:** Traditionally, making a polished slide deck or graphic can take many hours of design work or the hiring of a designer. Here, the AI does it in seconds, and for free or minimal cost. You avoid the need for a dedicated technical writer or graphic designer in early stages. This speed lets you update documentation continuously as the project evolves, without significant overhead. As a result, documentation is no longer a burdensome task but a near-automatic byproduct of your development process.

- **Bridge to Collaboration:** If and when you bring developers on board to enhance the project (to address those non-functional requirements or scale it up), you will have a much easier handover. Instead of dumping a pile of uncommented code on them, you're providing a clear brief and even visual aids to illustrate the system. This can significantly reduce onboarding time. Developers will appreciate having an architecture overview or feature list in slides – it gives them a starting mental model. In essence, you're using AI to create *technical onboarding materials* for your own project.

- **Documenting for Posterity and Learning:** Each iteration's documents and slides form a chronicle of your project's progression. Looking back at previous briefs can reveal how your idea expanded or changed. It's also a learning tool for you – by seeing how the AI explains each version, you pick up more technical literacy over time. Over multiple projects, you'll find your own ability to describe systems improves, influenced by the AI's phrasing and structure.

- **Enhanced Creativity through Visualization:** The infographic and slide generation can spark new ideas. Sometimes the way NotebookLM presents your concept – perhaps using a clever metaphor in an image or a particular phrasing on a slide – can give you a fresh perspective. It might surface connections you hadn't verbalized. For example, an auto-generated graphic might categorize your features in a novel way, prompting you to think "Actually, we could group our features into these three themes, as the graphic suggests." Thus, the AI isn't just regurgitating your input; it's *synthesizing and reframing* it in potentially insightful ways [15].

Best Practices and Tips for Success

To get the most out of this workflow, consider the following best practices:

- **Write Detailed Briefs:** The quality of NotebookLM's output depends on the richness of your source document. Include not just what the app does, but *why* (the problem it solves) and *how* it's implemented at a high level. If your brief has sections (e.g., Introduction, Features, Architecture, Future Work), the slide deck is more likely to mirror those sections in a logical sequence. Avoid overly prompt-like language in the brief; write it as if explaining to a colleague or future team member.

- **Use Narrative Style for NotebookLM:** As noted, NotebookLM currently excels with narrative or story-like input. Instead of a bullet list of facts, write a brief story of the project. For instance, rather than: "Feature 1: user login. Feature 2: data export. Next: add notifications," you might write: "*Our application allows users to log in securely to access personalized data. It currently supports exporting this data as files, which was a key milestone achieved this week. The next step in*

our roadmap is to implement a notification system, so users are alerted to updates…". This reads more like a report or story, which NotebookLM can more naturally turn into slides with titles like "User Login & Security", "Data Export Feature", "Planned: Notification System", etc.

- **Double-Check AI-Generated Facts:** While NotebookLM sticks to your source content, there is always a possibility of subtle misinterpretation. Review the infographic text and each slide's bullets to ensure they accurately reflect your project. If something is off, you may need to clarify that point in your brief and re-run the generation. Remember that generative AI is experimental – it's very powerful, but you are the final editor.

- **Regenerate and Remix:** Don't hesitate to run multiple generations. Each run is somewhat unique. If you have the time, you might generate two infographics and three slide decks, then cherry-pick the best one or even splice slides from different decks. For example, if deck A had a great architecture diagram on slide 3, you can export both decks and copy that slide into deck B's file. This is a manual step, but far quicker than crafting everything yourself. Since direct editing in NotebookLM isn't available (yet), this regenerate-and-mix approach is a useful technique.

- **Manage Your Notebooks and Usage:** Keep your NotebookLM workspace organized. You might create one notebook per project (or even per major version of a project) and only load relevant sources into it. This avoids any cross-talk between unrelated documents. Also, be mindful of usage limits if on the free tier – if you plan to do many runs, consider upgrading to Pro or using a student access program if available [17]. If you hit a generation limit, you might have to wait or trim your content (e.g., generate slides for half the document at a time).

- **Stay Updated on Tool Improvements:** As mentioned, this field is evolving fast. New features may roll out – for instance, Google might re-introduce editing capabilities, or other tools might offer similar slide-generation from text. Keep an eye on updates (NotebookLM's team posts new features often). Being early adopters, you and your fellow vibe coders can quickly integrate these improvements into your workflow, maintaining a competitive edge in speed and presentation.

Future Outlook

What we are doing today with a combination of LLM assistants and NotebookLM is likely just the beginning of AI-augmented development workflows. It's safe to say that *today's limitations will be gone tomorrow*. The inability to edit AI-generated slides or see how exactly they were created is likely to be solved by upcoming releases or competitive products. We might soon see end-to-end integration: imagine coding an app with an AI assistant, and simultaneously the system generates updated documentation and a dashboard of project health (tests, performance metrics) – all driven by AI observing your progress. NotebookLM itself may integrate more deeply with development tools, so that after each Git commit, a new slide deck can be waiting for the team's review, summarizing the changes.

For now, we recommend viewing NotebookLM as a *design partner* that can take your raw text and turn it into something visually appealing. Its current state is already impressive, but improvements (possibly including multi-modal outputs like videos or interactive diagrams) are on the horizon [13] [10]. OpenAI and others are undoubtedly experimenting in this space too – we might see integrations where ChatGPT or similar can produce slide decks directly from a conversation (some users already compare NotebookLM's output to what GPT-4 with plugins can do, but with the advantage of being grounded in user-provided data).

The broader implication is that **AI is leveling the playing field** in software development and communication. Non-engineers can not only create functional software with AI help, but also all the collateral that traditionally required a team (designers for slides, technical writers for docs). The role of professional developers and designers will evolve to higher-level guidance, review, and refinement roles – focusing on the creative and complex tasks that AI alone can't handle yet, while AI handles the boilerplate and initial drafts.

By embracing these tools early, you equip yourself (and your friends, in the case of this paper's intended audience) to ride this wave of change. The excitement and passion we see in vibe coders today can be sustained into later stages of projects because the frustrating chores are minimized. You don't get bogged down as quickly by documentation or communication issues; AI helps carry that load.

Conclusion

Generative AI has opened the door for a new class of creators to turn ideas into reality without the traditional barriers of learning to code or hiring big teams. *Vibe coding* with LLMs empowers business-savvy individuals to build prototypes at the speed of thought. Yet, turning those prototypes into successful products requires clear understanding and communication of the technology being built. In this paper, we presented a method to leverage AI not just in writing code, but in **writing about the code and visualizing it**. By using an LLM to generate technical briefs and Google's NotebookLM to convert those briefs into infographics and slide decks, even a lone non-developer can produce **professional-grade documentation and presentations** with minimal effort.

This workflow – **think, build, summarize, visualize, share** – can dramatically accelerate the journey from a rough idea to a polished presentation. It enables better collaboration with others and provides a level of polish that can be crucial for gaining support (be it convincing a partner, onboarding a developer, or impressing an investor). And as evidenced by the creation of this very document, tools like ChatGPT (in Deep Research mode) are capable of turning a stream-of-consciousness memo into a well-structured white paper, ready for NotebookLM to work its magic on visualizing it.

We encourage you and our fellow innovators to experiment with this approach. Use your LLM of choice to narrate what you've built, feed that knowledge into NotebookLM, and behold the slides and graphics that come out. It's a thrilling experience the first time you see an AI-generated diagram of your idea – it's like the concept in your head has taken shape on screen without a human designer involved. And it's equally thrilling to realize that you, as a non-coder, can produce a technical write-up that reads well, courtesy of your AI writing partner.

In a sense, **you are never working alone** – you have co-pilots for coding (to build the app), for writing (to create the brief), and for designing (to generate visuals). This collaboration between human creativity and AI capability is the hallmark of our current technological moment. Embrace it fully: the tools will only get better from here [13], and those who integrate them into their workflow will pioneer new possibilities in software development and knowledge sharing.

Credit: This document was co-authored by *Dinis Cruz* and *ChatGPT Deep Research*, exemplifying the human-AI partnership in content creation. All source materials and inspirations have been cited, and the process described herein was used in real-time to produce the guidance we now share with you. We look forward to seeing how you leverage these AI techniques to amplify your own projects. Happy vibe coding, and happy visualizing!

---

1 2 4 6 7 Vibe Coding. AI-Assisted Coding for Non-Developers | by Niall McNulty | Medium
https://medium.com/@niall.mcnulty/vibe-coding-b79a6d3f0caa

3 5 Vibe coding: the pros and cons of no-code development | Emmanuel Maggiori posted on the topic | LinkedIn
https://www.linkedin.com/posts/emaggiori_vibe-coding-is-the-new-no-code-it-has-activity-7362813338994499584-IrVY

8 9 Turn Voice Notes into Structured Documents Automatically | AI Business Success Mentoring
https://ai-bsm.com/turn-voice-notes-into-structured-documents-automatically

10 13 14 15 8 ways to make the most out of Slide Decks in NotebookLM
https://blog.google/technology/google-labs/8-ways-to-make-the-most-out-of-slide-decks-in-notebooklm/

11 12 17 NotebookLM: The Most Useful Free AI Tool of 2025
https://wondertools.substack.com/p/notebooklm-the-complete-guide

16 Google rolls back NotebookLM's new features! : r/notebooklm
https://www.reddit.com/r/notebooklm/comments/1p7xwh8/google_rolls_back_notebooklms_new_features/