


AI VIDEO GEN

AI in daily life

 The Only Trait for Success in the AI Era—How to Build It | Carnegie Mellon University Po-Shen ...


[Rocket: App Templates to Build Anything in Minutes](#)

<https://medium.com/@rathod9/rocket-new-the-one-platform-to-build-a-1m-application-in-a-single-prompt-70bf69ff064f>

Business AI

<https://www.salesforce.com/agentforce>

AI Glasses

 Googles New AI Glasses Are The Future Better Than META!

<https://arvr.google.com>

Personal AI assistant

<https://clawd.bot>

<https://chat.lindy.ai>

<https://questflow.ai>

<https://www.minimax.io/models/text/m27>

Google Notebook LM

<https://notebooklm.google>

Voice to notes

<https://www.writevoice.io>

<https://sona.wtf>

Notes/Docs summary

<https://www.chatpdf.com>

<https://www.recall.it>

[Notion](#)

<https://obsidian.md>

Chat/Meeting AI transcription companion

<https://sona.wtf>

<https://www.amurex.ai> - <https://github.com/the-personal-ai-company/amurex>

Note -> Visual

<https://www.napkin.ai>

Mindmap/Note Graph

<https://www.averi.ai>

<https://obsidian.md>

Translator

<https://www.deepl.com>

Web Browsing

<https://blog.google/products-and-platforms/products/chrome/skills-in-chrome>

AI Interactive 3D world

World Labs

<https://www.worldlabs.ai/blog/marble-world-model>

<https://marble.worldlabs.ai>

<https://www.worldlabs.ai/blog/announcing-the-world-api>

<https://platform.worldlabs.ai>

Google Genie

<https://deepmind.google/discover/blog/genie-3-a-new-frontier-for-world-models>

<https://labs.google/projectgenie>

<https://deepmind.google/models/genie>

 **Genie 3: Creating dynamic worlds that you can navigate in real-time**

Tencent Hunyuan

<https://3d-models.hunyuan.tencent.com>

PhysTwin

<https://github.com/Jianghanxiao/PhysTwin>

AI app dev

<https://github.com/openai/skills>

<https://v0.app>

<https://rork.com>

<https://huggingface.co/spaces>

<https://www.gradio.app/guides/developing-faster-with-reload-mode#vibe-mode>

<https://ag2.ai>


<https://mistral.ai/news/le-chat-mcp-connectors-memories>


<https://huyenchip.com>

<https://www.amazon.com/Designing-Machine-Learning-Systems-Production-Ready/dp/1098107969>

- <https://stanford-cs329s.github.io>

https://www.youtube.com/playlist?list=PLqGT7Mfu5Dly8foorUr-z_QQbzTP6TJPj

 [The Matthew Berman Vibe Coding Playbook.pdf](#)

 [Humanity's Last Prompt Engineering Guide.pdf](#)

Google AI Studio

<https://www.skills.google> Free training + Certificates with Gemini

<https://aistudio.google.com/apps> AI model app development for developers

<https://gemini.google.com> AI conversational agent for normal users

<https://github.com/google-gemini/cookbook>

Google ADK

<https://github.com/google/adk-python>

<https://github.com/google/adk-samples>

OpenAI Codex

<https://developers.openai.com/codex/use-cases>

<https://developers.openai.com/showcase>

<https://github.com/openai/skills>

https://developers.openai.com/cookbook/examples/gpt-5/codex_prompting_guide

LLM app integration

<https://en.wikipedia.org/wiki/LangChain>

<https://www.langchain.com>

<https://github.com/facebookresearch/SONAR>

<https://github.com/Alibaba-NLP/DeepResearch>

Microsoft Phi

[Phi Open Models - Small Language Models](#)

NVIDIA Nemo

<https://github.com/NVIDIA-NeMo>

<https://developer.nvidia.com/blog/build-with-kimi-k2-5-multimodal-vlm-using-nvidia-gpu-accelerated-endpoints>

NVIDIA AI RAG

<https://developer.nvidia.com/ai-models>

<https://developer.nvidia.com/blog/build-a-rag-agent-with-nvidia-nemotron>

<https://developer.nvidia.com/blog/traditional-rag-vs-agentic-rag-why-ai-agents-need-dynamic-knowledge-to-get-smarter>

LLM Knowledge Bases

<https://venturebeat.com/data/karpathy-shares-llm-knowledge-base-architecture-that-bypasses-rag-with-an>

ForwardFuture AI

<https://tools.forwardfuture.ai>

Vibe coding

<https://opal.google>

<https://www.rocket.new>

<https://cursor.directory>

<https://trymagically.com>

<https://bolt.new>

Tips

<https://claude.ai/public/artifacts/a2e1f3a1-747e-4e86-89f0-23bf3f1f2014>

<https://medium.com/@amyabafor013/i-tried-vibe-coding-with-bolt-new-heres-my-honest-verdict-ff03384cdece>

https://www.reddit.com/r/replit/comments/1kpuudj/replit_learnings_best_practices_after_a_month_of

https://www.reddit.com/r/ClaudeAI/comments/1kivv0w/the_ultimate_vibe_coding_guide

[The Ultimate Vibe Coding Guide : r/ClaudeAI](#)

Productivity

So I have been using Cursor for more than 6 months now and I find it a very helpful and very strong tool if used correctly and thoughtfully. Through these 6 months and with a lot of fun projects personal and some production-level projects and after more than 2500+ prompts, I learned a lot of tips and tricks that make the development process much easier and faster and makes and help you vibe without so much pain when the codebase gets bigger and I wanted to make a guide for anyone who is new to this and want literally everything in one post and refer to it whenever need any guidance on what to do!:

1. Define Your Vision Clearly

Start with a strong, detailed vision of what you want to build and how it should work. If your input is vague or messy, the output will be too. Remember: garbage in, garbage out. Take time to think through your idea from both a product and user perspective. Use tools like Gemini 2.5 Pro in Google AI Studio to help structure your thoughts, outline the product goals, and map out how to bring your vision to life. The clearer your plan, the smoother the execution.

2. Plan Your UI/UX First

Before you start building, take time to carefully plan your UI. Use tools like v0 to help you visualize and experiment with layouts early. Consistency is key. Decide on your design system upfront and stick with it. Create reusable components such as buttons, loading indicators, and other common UI elements right from the start. This will save you tons of time and effort later on You can also use `**https://21st.dev/**`; it has a ton of components with their AI prompts, you just copy-paste the prompt, it is great!

3. Master Git & GitHub

Git is your best friend. You must know GitHub and Git; it will save you a lot if AI messed things up, you could easily return to an older version. If you did not use Git, your codebase could be destroyed with some wrong changes. You must use it; it makes everything much easier and organized. After finishing a big feature, you must make sure to commit your code. Trust me, this will save you from a lot of disasters in the future!

4. Choose a Popular Tech Stack

Stick to widely-used, well-documented technologies. AI models are trained on public data. The more common the stack, the better the AI can help you write high-quality code.

I personally recommend:

Next.js (for frontend and APIs) + Supabase (for database and authentication) + Tailwind CSS (for styling) + Vercel (for hosting).

This combo is beginner-friendly, fast to develop with, and removes a lot of boilerplate and manual setup.

5. Utilize Cursor Rules

Cursor Rules is your friend. I am still using it and I think it is still the best solution to start solid. You must have very good Cursor Rules with all the tech stack you are using, instructions to the AI model, best practices, patterns, and some things to avoid. You can find a lot of templates here:

<https://cursor.directory>

6. Maintain an Instructions Folder

Always have an instructions folder. It should have markdown files. It should be full of docs-example components to provide to the AI to guide it better or use (or context7 mcp, it has a tons of documentation).

7. Craft Detailed Prompts

Now the building phase starts. You open Cursor and start giving it your prompts. Again, garbage in, garbage out. You must give very good prompts. If you cannot, just go plan with Gemini 2.5 Pro on Google AI Studio; make it make a very good intricate version of your prompt. It should be as detailed as possible; do not leave any room for the AI to guess, you must tell it everything.

8. Break Down Complex Features

Do not give huge prompts like "build me this whole feature." The AI will start to hallucinate and produce shit. You must break down any feature you want to add into phases, especially when you are building a complex feature. Instead of one huge prompt, it should be broken down into 3-5 requests or even more based on your use case.

9. Manage Chat Context Wisely

When the chat gets very big, just open a new one. Trust me, this is the best. The AI context window is limited; if the chat is very big, it will forget everything earlier, it will forget any patterns, design and will start to produce bad outputs. Just start a new chat window then. When you open the new window, just give the AI a brief description about the feature you were working on and mention the files you were working on. Context is very important (more on that is coming..)!

10. Don't Hesitate to Restart/Refine Prompts

When the AI gets it wrong and goes in the wrong way or adding things that you do not want, returning back, changing the prompt, and sending the AI again would be just much better than completing on this shit code because AI will try to save its mistakes and will probably introduce new ones. So just return, refine the prompt, and send it again!

11. Provide Precise Context

Providing the right context is the most important thing, especially when your codebase gets bigger. Mentioning the right files that you know the changes will be made to will save a lot of requests and too much time for you and the AI. But you must make sure these files are relevant because too much context can overwhelm the AI too. You must always make sure to mention the right components that will provide the AI with the context it needs.

12. Leverage Existing Components for Consistency

A good trick is that you can mention previously made components to the AI when building new ones. The AI will pick up your patterns fast and will use the same in the new component without so much effort!

13. Iteratively Review Code with AI

After building each feature, you can take the code of the whole feature, copy-paste it to Gemini 2.5 Pro (in Google AI Studio) to check for any security vulnerabilities or bad coding patterns; it has a huge context window. Hence, it actually gives very good insights where you can then input into Claude in Cursor and tell it to fix these flaws. (Tell Gemini to act as a security expert and spot any flaws. In another chat, tell it so you are an expert (in the tech stack at your tech stack), ask it for any performance issues or bad coding patterns). Yeah, it is very good at spotting them! After getting the insights from Gemini, just copy-paste it into Claude to fix any of them, then send it Gemini again until it tells you everything is 100% ok.

14. Prioritize Security Best Practices

Regarding security, because it causes a lot of backlash, here are security patterns that you must follow to ensure your website is good and has no very bad security flaws (though it won't be 100% because there will be always flaws in any website by anyone!):

Trusting Client Data: Using form/URL input directly.

Fix: Always validate & sanitize on server; escape output.

Secrets in Frontend: API keys/creds in React/Next.js client code.

Fix: Keep secrets server-side only (env vars, ensure .env is in .gitignore).

Weak Authorization: Only checking if logged in, not if allowed to do/see something.

Fix: Server must verify permissions for every action & resource.

Leaky Errors: Showing detailed stack traces/DB errors to users.

Fix: Generic error messages for users; detailed logs for devs.

No Ownership Checks (IDOR): Letting user X access/edit user Y's data via predictable IDs.

Fix: Server must confirm current user owns/can access the specific resource ID.

Ignoring DB-Level Security: Bypassing database features like RLS for fine-grained access.

Fix: Define data access rules directly in your database (e.g., RLS).

Unprotected APIs & Sensitive Data: Missing rate limits; sensitive data unencrypted.

Fix: Rate limit APIs (middleware); encrypt sensitive data at rest; always use HTTPS.

15. Handle Errors Effectively

When you face an error, you have two options:

Either return back and make the AI do what you asked for again, and yeah this actually works sometimes.

If you want to continue, just copy-paste the error from the console and tell the AI to solve it. But if it took more than three requests without solving it, the best thing to do is returning back again, tweaking your prompt, and providing the correct context as I said before. Correct prompt and right context can save sooo much effort and requests.

16. Debug Stubborn Errors Systematically

If there is an error that the AI took so much on and seems never to get it or solve it and started to go on rabbit holes (usually after 3 requests and still did not get it right), just tell Claude to take an overview of the components the error is coming from and list top suspects it thinks are causing the error. And also tell it to add logs and then provide the output of them to it again. This will significantly help it find the problem and it works correctly most of the times!

17. Be Explicit: Prevent Unwanted AI Changes

Claude has this trait of adding, removing, or modifying things you did not ask for. We all hate it and it sucks. Just a simple sentence under every prompt like (Do not fuckin change anything I did not ask for Just do only what I fuckin told you) works very well and it is really effective!

18. Keep a "Common AI Mistakes" File

Always have a file of mistakes that you find Claude doing a lot. Add them all to that file and when adding any new feature, just mention that file. This will prevent it from doing any frustrating repeated mistakes and you from repeating yourself!

I know it does not sound as "vibe coding" anymore and does not sound as easy as all of others describe, but this is actually what you need to do in order to pull off a good project that is useful and usable for a large number of users. These are the most important tips that I learned after using Cursor for more than 6 months and building some projects using it! I hope you found it helpful and if you have any other questions I am happy to help!

Also, if you made it to here you are a legend and serious about this, so congrats bro!

Happy vibing!

GenAI Toolkit

<https://genai.works>

Webinars

▶ The GenAI Toolkit: Frameworks, Libraries, and Open-Source Resources

▶ Webinar: Automate Your Data Labeling With GenAI

Codes

<https://github.com/genai-works-org>

Courses

<https://genai.works/genai-courses>

<https://newsletter.genai.works/p/vibe-coding-and-the-entropy-of-modern-software>

2D -> 3D

<https://aidemos.meta.com/segment-anything>

<https://3d-models.hunyuan.tencent.com>

<https://github.com/Microsoft/TRELLIS>

<https://microsoft.github.io/TRELLIS>

<https://trellis3d.net>

<https://build.nvidia.com/microsoft/trellis>

<https://studio.tripo3d.ai>

<https://www.meshy.ai>

<https://replicate.com/camenduru/instantmesh/examples>

<https://animateimage.net>

Image generation

<https://bytedance.github.io/InfiniteYou>

<https://bfl.ai> - <https://github.com/black-forest-labs/flux>

<https://www.recraft.ai>

Google

<https://ai.google.dev/gemini-api/docs/imagen>

<https://deepmind.google/models/gemini-image>

Whisk

<https://labs.google/fx/projectgenie/tools/whisk/project>

Microsoft

<https://playground.microsoft.ai>

OpenAI

<https://developers.openai.com/codex/use-cases/browser-games>

<https://github.com/openai/skills/blob/main/skills/.curated/imagegen/SKILL.md>

Audio/Music generation

suno.com

<https://deepmind.google/models/lyria/prompt-guide>

Video generation

<https://huggingface.co/spaces?category=video-generation>

<https://animateimage.net>

<https://www.artificialstudio.ai>

https://www.linkedin.com/posts/genaiworks_womeninaitech-genaiworks-generativeai-activity-7304901220014424064-kjLd

<https://medium.com/@thierryjmoreau/build-your-own-genai-video-generation-pipeline-cdc1515d1db9>

Video gen ai tools

<https://higgsfield.ai>

<https://www.promeai.pro/image-to-video>

<https://captions.ai/overview>

[Vidu AI](#)

<https://github.com/zai-org/CogVideo>

<https://github.com/THUDM/CogKit>

<https://github.com/aigc-apps/VideoX-Fun>

<https://github.com/WeChatCV/UnderEraser>

<https://github.com/ModelTC/LightX2V>

▶ Which Cinematic AI Video Tool is Best?

▶ This FREE AI Video Generator is Unbelievable!

<https://www.v7labs.com/darwin/video-annotation>

<https://www.labellerr.com/blog/best-data-annotation-platform-with-auto-labeling-feature>

E-Learning

<https://elai.io/e-learning>

RunwayML

<https://runwayml.com>

<https://runwayml.com/research/creativity-as-search-mapping-latent-space>

LumaLabs

<https://lumalabs.ai/dream-machine>

Google

<https://github.com/google-gemini/cookbook>

<https://github.com/googleapis/python-genai>

<https://github.com/google-gemini/gemini-api-quickstart>

Veo

<https://gemini.google/overview/video-generation>

<https://deepmind.google/models/veo>

<https://ai.google.dev/gemini-api/docs/video>

<https://veo3.ai>

<https://blog.google/technology/ai/google-flow-veo-ai-filmmaking-tool>

<https://labs.google/flow>

<https://workspace.google.com/products/vids>

Flow

<https://labs.google/flow>

Vertex AI

<https://cloud.google.com/vertex-ai>

<https://cloud.google.com/generative-ai-studio>

Meta

<https://makeavideo.studio>

Auto object labelling in videos

<https://ai.meta.com/sam2>

<https://ai.meta.com/sam3>

<https://segment-anything.com>

<https://labelbox.com/solutions/object-detection>

<https://www.innovatiana.com/en/post/top-10-annotation-tools>

[BakuFlow: A Streamlining Semi-Automatic Label Generation Tool](#)

[SAM2Auto: Auto Annotation Using FLASH](#)

[GitHub - NVlabs/ODISE: Official PyTorch implementation of ODISE: Open-Vocabulary Panoptic Segmentation with Text-to-Image Diffusion Models \[CVPR 2023 Highlight\]](#)

[GitHub - MCG-NKU/E2FGVI: Official code for "Towards An End-to-End Framework for Flow-Guided Video Inpainting" \(CVPR2022\)](#)

Roboflow

<https://roboflow.com/workflows/templates>

<https://github.com/roboflow/notebooks>

<https://www.youtube.com/c/Roboflow>

NVIDIA-Cosmos

<https://www.nvidia.com/en-us/ai/cosmos>

<https://developer.nvidia.com/blog/develop-custom-physical-ai-foundation-models-with-nvidia-cosmos-predict-2>

Alibaba - CausVid

<https://causvid.github.io>

<https://github.com/alibaba-damo-academy/Inferix/tree/main/example/causvid>

Audio2Face

<https://developer.nvidia.com/blog/nvidia-open-sources-audio2face-animation-model>

<https://developer.nvidia.com/ace-for-games>

Video Analytics

<https://cloud.google.com/vision>

<https://learn.microsoft.com/en-us/azure/ai-services/computer-vision/concept-describe-images-40>

<https://www.nvidia.com/en-us/ai/cosmos>

<https://www.nvidia.com/en-us/use-cases/video-analytics-ai-agents>

<https://developer.nvidia.com/blog/advance-video-analytics-ai-agents-using-the-nvidia-ai-blueprint-for-video-search-and-summarization>

<https://allenai.org/molmo>

Photo/Image Analysis

<https://en.wikipedia.org/wiki/Photoanalysis>

https://en.wikipedia.org/wiki/Image_analysis

Image Captioning/Annotation

https://en.wikipedia.org/wiki/Automatic_image_annotation

[Surveying the Landscape of Image Captioning Evaluation: A Comprehensive Taxonomy, Trends and Metrics Analysis](#)

<https://research.google/blog/a-picture-is-worth-a-thousand-coherent-words-building-a-natural-description-of-images>

<https://research.google/blog/show-and-tell-image-captioning-open-sourced-in-tensorflow>

<https://github.com/karpathy/neuraltalk2>

<https://cs.stanford.edu/people/karpathy/neuraltalk2/demo.html>

<https://vimeo.com/146492001>

<https://github.com/salesforce/LAVIS?tab=readme-ov-file#image-captioning>

<https://huggingface.co/Salesforce/blip-image-captioning-base>

<https://openai.com/index/clip>

Visual reasoning

<https://github.com/facebookresearch/clevr-dataset-gen>

<https://github.com/nvidia-cosmos/cosmos-reason1>

<https://developer.nvidia.com/blog/build-with-kimi-k2-5-multimodal-vlm-using-nvidia-gpu-accelerated-endpoints>

AI Database

<https://supabase.com/docs/guides/getting-started/features>

AI Data Analytics

<https://www.thoughtspot.com>

<https://www.knime.com>

RESEARCH

<https://github.com/ALEEEHU/World-Simulator>

<https://github.com/Saiyan-World/goku>

Benchmark

<https://huggingface.co/datasets/saiyan-world/Goku-MovieGenBench>

<https://huggingface.co/datasets/heyuanyu/LV-Bench>