These Premium Leaked Startup Prompts Became My Secret Weapon

The Real Way Billion Dollar AI Tools Think

Alex Dunlop • 6 min read • 2025-07-09

https://medium.com/vibe-coding/these-premium-leaked-startup-prompts-became-my-secret-weapon-92aa4e9cde2f

Image I made with Midjourney and then edited with Figma

I was building AI applications with prompts that felt amateur. My LLM responses were giving generic responses, lacked personality, and customers constantly complained about inconsistent behavior.

This issue was worst when I was building a prompt for disability access, the client kept saying it felt not personal for users that really needed it. Having these problems compared to UX of Cursor, made me feel like I had the access to a Ferrari but couldn't get it out of first gear.

Some questions would get basic responses, some would get great results, the worst thing being the inconsistency.

Everything changed when I discovered a GitHub repository, it changed my approach to prompting. Someone leaked system prompts for billion dollar AI companies and I had no idea.

Not a Medium member? Keep reading for free by clicking here.

The Main Problem With AI Prompts

Whenever you build a system prompt, you quickly realise how little documentation there is to writing prompts. So what's the best solution, learning by example.

What most developers start with:

• Saying *“You are a helpful assistant”* prompts.

• One prompt to rule them all.

• Personality lacking prompts.

• Prompts that break when users ask unexpected questions.

• Tool calling that feels very robotic.

• Tool calling gets over fitted and makes unnecessary tool calls.

Users will want to stop using AI chats after disappointing interactions. When their expectations aren’t met, it is hard to understand what they are even able to do.

I have experienced this so many times firsthand. While using the AI feature with Gmail for example, I will ask something simple for it to say *“I’m unable to perform that action”*, no further help given to me.

The Discovery That Changed My Prompts Forever

While viewing the prompt engineering reddit, I stumbled across a GitHub repository with **65,000 stars,** with a very bold claim.

GitHub Repository: system-prompts-and-models-of-ai-tools

Reading the README, I thought no way this is real.

```
Full v0, Cursor, Manus, Same.dev, Lovable, Devin, Replit Agent,
Windsurf Agent, VSCode Agent system prompts and models.
```

These aren’t just some random examples that have been shared, these are the current most successful use cases from the most successful AI startups.

• **v0 by Vercel**

• **Cursor**

• **Devin AI**

• **Lovable**

This gave me a complete upgrade on my basic system prompts, these prompts built these unbelievably advanced AI products. It’s not just about the content of the prompt but the structure of these prompts.

What I Found Inside Changed Everything

I first checked my personal favourite product of the list, Cursor. This is also my favourite prompt to refer back to. I was struggling with an agent I had just built, the problem was the agent would always call the tools even when it didn’t need to. I had searched for examples of the case but couldn’t find something I liked.

That’s when I found the Cursor prompt and how they deal with the tool problem is absolutely masterclass.

Now I follow all of the examples as a template that I modify, I will always define the **AI role, the tone, provide examples, specify data, describe the task**. I will always do this when applicable. The structure in the leaked prompts is incredible and I use them often, it's amazing they are free.

```
<tool_calling>
You have tools at your disposal to solve the coding task. Follow these rules
regarding tool calls:
1. ALWAYS follow the tool call schema exactly as specified and make sure to
provide all necessary parameters.
2. The conversation may reference tools that are no longer available. NEVER call
tools that are not explicitly provided.
3. **NEVER refer to tool names when speaking to the USER.** For example, instead
of saying 'I need to use the edit_file tool to edit your file', just say 'I will
edit your file'.
4. Only calls tools when they are necessary. If the USER's task is general or
you already know the answer, just respond without calling tools.
5. Before calling each tool, first explain to the USER why you are calling it.
</tool_calling>
```

I use this tool calling prompt every time I have an agentic flow now, the issue wasn't the tools the AI had access to but instead how it thought about using them.

I have even tried updating my Claude prompts with great success, here is an example of one I tried:

```
You are a Senior Backend API Architect, specialising in secure and scalable
microservices.
Maintain a precise and analytical tone.

Here is the problematic endpoint code, relevant database schema, and recent
error logs:
<code_snippet>...</code_snippet>
<db_schema>...</db_schema>
<error_logs>...</error_logs>

Here's an example of a good, detailed debug response:
<example>

  <response>...</response>
</example>

Your task is to identify the root cause of the bug, propose a robust solution,
and provide refactored code. Before providing a fix, ask clarifying questions to
ensure you understand the entire system context and potential downstream
impacts.
Think step by step before giving your answer.
Put your final response in <solution> tags.
```

This difference is amazing, the AI didn't just give me a fix, it also explained it to me in a format that worked how I think, not how it thinks I think. This would have taken me multiple prompts and back and forths.

Your Cheat Code of Prompts

Use this GitHub repository like me, use it as a practical guide to mastering AI prompts.

Don’t just copy-paste. Read through these prompts (Devin AI, Cursor, or VSCode Agent). Notice how they systematically define various aspects of the AI’s behavior:

- `TASK_CONTEXT` - What is the overarching goal and the AI’s persona *(coding agent, senior partner).* This helps with the entire AI interaction.
- `TONE_CONTEXT` - I noticed this at work but it’s backed up here, is there a specific tone needed *(friendly, analytical, authoritative)*. This helps the AI communication style.
- `DETAILED_TASK_DESCRIPTION` - Notice how they break down complex tasks. This one is very hard, I have found looking at multiple prompt examples helps with this.
- `EXAMPLES` - I was already doing this one, as an ex colleague shared this with me, this is currently one of the most powerful ways of explaining what you are expecting. If you aren’t expecting something, examples also help for this, provide the example with the response you wish you had.
- `INPUT_DATA`, `OUTPUT_FORMATTING` - I personally find these less relevant as I no longer use straight prompting and now use a library, but if you use straight prompting then I recommend this for you! I only really use this with the likes of Claude.
- `TOOL_CALLING` - This is massive for advanced prompting, the key to preventing over fitting *(the AI making unneeded tool calls)*. Things like *“only call tools when necessary”, “explain why you’re calling a tool”, and “never refer to tool names directly to the user”.* Will completely change everything for you.

These insights will help you build **more sophisticated and effective prompts**.

Improving Your AI Prompting

Applying these prompting techniques completely changes how I interact with AI. If you’re interested in improving the way you interact with Claude I wrote an article helping with that.

Stop Building Amateur Prompts

By using x1xhlol’s system-prompts-and-models-of-ai-tools repo, you are able to gain production high grade prompts for extremely sophisticated AI systems, all for free.

I would recommend finding more repo’s like this, I’ll share ones I find useful here but another resource I check often is reddit, when it comes to AI prompting this has surprisingly been my most useful way of finding new AI prompt techniques.

If you have any tips I missed I would love to hear about them. If you have had any AI prompting disasters I would also love to hear about them.

I am not affiliated with Claude or other tools mentioned. All opinions and experiences shared are my own.

Other mentions by Author

- medium.com | 99% of Developers are Using Claude Wrong (How to Be The 1%)
- medium.com | Published in Vibe Coding
- medium.com | Written by Alex Dunlop