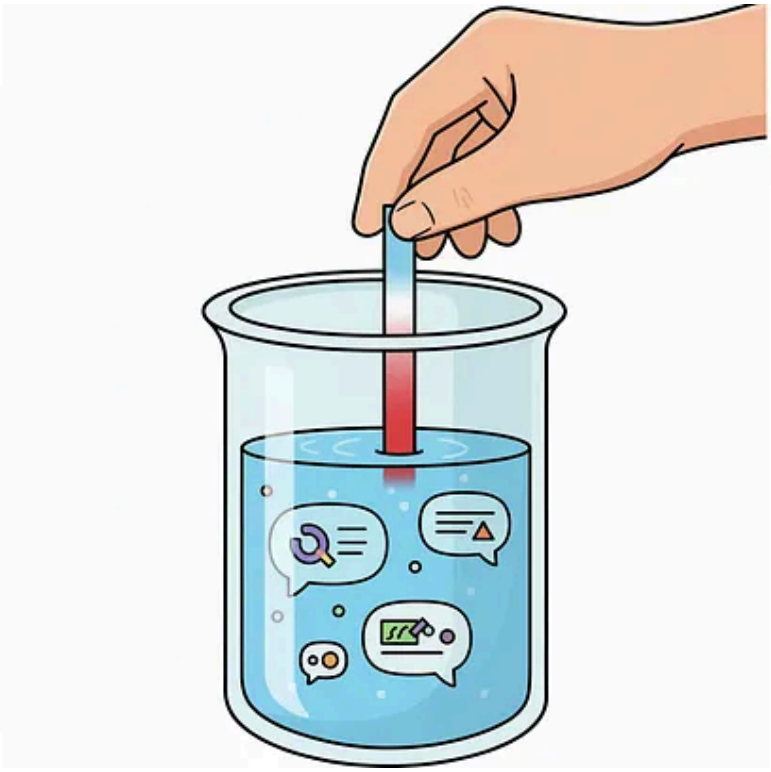
Bootcamp

The 4 characteristics of successful AI chat integrations

Waleed Kadous Follow 10 min read · 4 days ago



TL;DR: We're still learning how to integrate chat interfaces in existing web apps. A *lot* of big companies are getting this badly wrong (including Google and Meta). I posit 4 principles of successful chat integrations: must be able to operate on the core primitives of the app; take advantage of the powerful characteristics of natural language such as flexible application criteria, flexible modifications and multi-step processes; allow the user to fluidly mix and match between the traditional UI interface and the chat interface; and maintain context across the user's entire experience. I discuss each of these with concrete examples of existing products, and also with a simple idea of a chat interface-enabled todo list manager. I also discuss some litmus tests. But if you want the short take-away: imitate Notion, and avoid the mistakes that Google Docs has made.

It is easy to forget that we've been building GUI applications for more than 3 decades and we've learned through experience what works and doesn't. If you're not sure, you can refer to things like Apple's Design Guidelines — there are canonical sources that give you very clear guidance on how to do things to maximize chances of success.

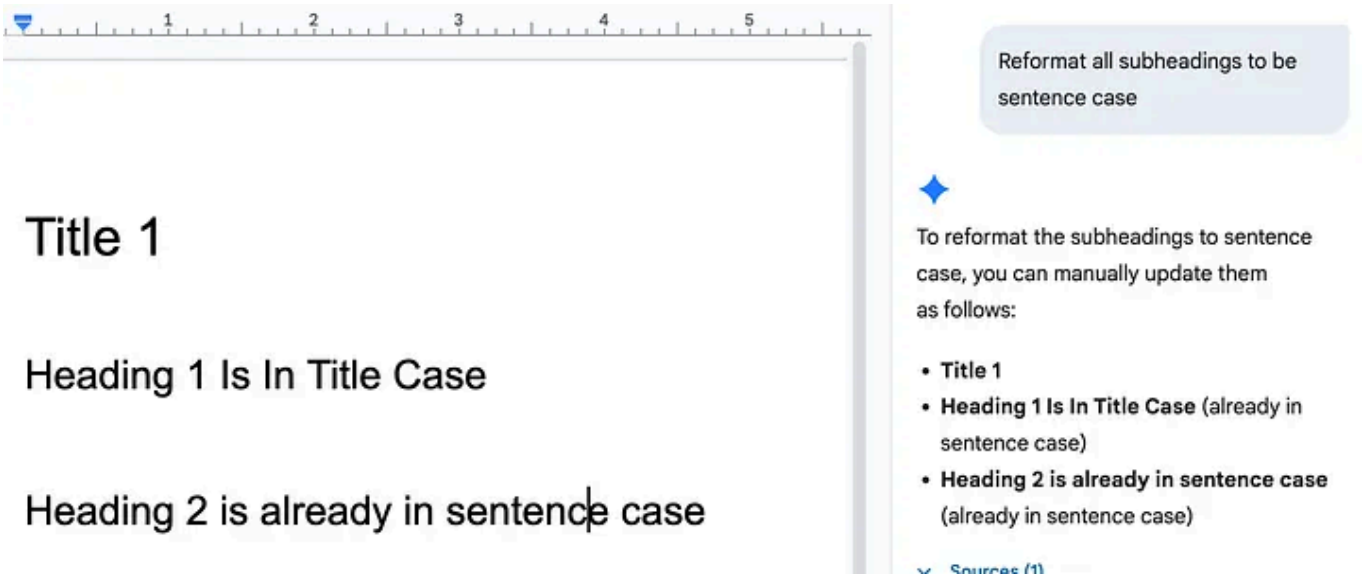
But we've only had a few years of building powerful chat integrations into those existing apps. The playbook is still being written. What I'd like to do in this article is start to sketch out what the characteristics of successful chat integrations are and what part of that playbook might look like. In the rest of this article I will discuss four areas:

Can operate on the core primitives of the web app

This is the most basic and yet for some reason companies are repeatedly getting it wrong.

If you're a messaging app, and you add a chat interface, you better be able to send messages, and search contacts. And yet WhatsApp's Llama interface in WhatsApp can't actually send messages.

If you're a word processing app, you better be able to fluidly add, remove and modify the content in the document. And yet Google's Doc AI can't do any of these things. Asking it to "Reformat all subheadings to be sentence case" fails, as it can only generate text, not directly edit the document structure. The experience with Google Docs AI is particularly frustrating because of this limitation.



Google Docs Chat can't really operate on documents. This is a really bad experience. I know I can manually fix the case, but I asked you to. Why can't you?

This brings us to our first litmus test, which I call "**The Copy-Paste Test**": if you're not the best chat interface to your own application, you have failed. My preferred way of building/editing Google docs is to open up Gemini and then export the end product to Google Docs after I've used Gemini to help me do it. Are your users copying content out of your app into ChatGPT or Gemini, and then pasting the results back in? If people would prefer to take content from your app, import it into another AI tool, work on it and feed it

back into your own app, you've clearly missed the mark; and your first goal is to fix that above all else.

Let's consider a todo list manager. If you're going to add a chat interface to a todo list manager, it better be *very* good at all the CRUD operations on todos. It's an insult to me if I ask you to add a todo and you come back and say "I can't actually add todos, but if you paste text, I can extract the todos for you and tell you how to manually add them …" Yet this is pretty much what some of the above tools do.

This is so mind-blowingly obvious that you have to ask why is it that these companies are struggling with these basic issues. My best guess is at these big companies, the chat interface is being caught up in turf wars between the gatekeepers of the traditional interface and the chat interface builders. It may also have something to do with legal departments that are being overly conservative.

It's not universal, and one interesting example is the Google Sheets chat integration. That one does many things right. Unlike its Docs counterpart, the Sheets AI can directly manipulate the application's core primitives. A prompt like "Create a new column that calculates the percentage change between Q1 and Q2 sales" results in a correctly formatted formula being inserted directly into the sheet, demonstrating a true understanding and manipulation of the application's objects.

And there are companies that get this right: two that are especially good. The first is Notion. Notion is about pages and tables. When you use Notion's Chat Integration, it's *really* good about being able to create and edit pages and tables. For example, you can ask Notion to "Summarize the key points from this page and create a new page with action items assigned to @Mike and

@Sarah," and it executes perfectly. The other one is Gamma's new Agent. It really does give you arbitrary control over how to modify the slides.

The contrast is stark. Notion can edit pages and tables seamlessly. Google Docs can't manipulate the text. WhatsApp can't send messages on your behalf or extract context from your conversations. If you can't perform the core operations of your application, you're asking users to work around your chat interface rather than with it.

But simply being able to manipulate your app's primitives is just the baseline. The real value of chat interfaces comes from what they can do that GUIs can't.

Take advantage of the powers of natural language

For many things, text isn't the natural interface. If you have a todo list, "Add a new todo to buy a gift for Sarah with due date next Friday" isn't any more useful than using the GUI to do the same thing.

But there are a few areas where natural language is … well, more natural than GUI. There are five specific areas where chat interfaces can provide capabilities that are difficult or impossible to achieve through traditional GUIs:

Selection: If you have a list, or you have a long document, selection can be clumsy. These are the things that you might use a filter for, or you might use a find + replace for. All the paragraphs in a document that refer to a particular person. All the animated objects in a slide deck. This is much easier to do with language than with GUIs.

Intelligent application: Now that you have your selection, you want to do something with it. AI lets you do that in an extremely intelligent way. Sometimes this is straightforward: like "slow down all the animations in this presentation by 50%". But sometimes it's far more nuanced: make sure all the sentences that mentioned that person are respectful and deferential. As a lark, for my todo list manager, I once told it there weren't enough emojis in my todo list and it should add more emojis. That's something that can't be done with a GUI.

Combination: If you have two lists, doing something that combines them is just much easier in natural language. Something like "For each of my customers based on what I know about them, pick the best small gift from this list of potential gifts" is much easier to say than to express visually. A special case is combining with external sources: images, imports, etc etc.

Compression and expansion: If I have a small list, I might want to add more examples to it. If I have a long document, I may want to summarize it.

Learning: It is much easier to learn how to do something with words than it is with a GUI. AI can help you work out how to solve problems in the tool. Even if all the AI says is "I can't do that right now, but here are the steps to take in the GUI" that is sometimes all it takes.

Thus the second litmus test is: how strong are you on these 5 things? What fraction of practical use cases of each of the above can you do? You can use the above list in a very practical way (of course, using AI to accelerate the process): generate examples that are specific to your application, run them and score the output (again, using AI in the form of LLM as a judge works pretty well in the use cases above).

This is where we come back to the leaders: Notion, Gamma and Google Sheets. Each one of them scores high on most of them. Perhaps one that still needs work across all of them is combination. Even so, there are gems here. I wanted to move my todo list from Slack to Notion. The number was small, so I just took a screen shot in Slack, and asked Notion to analyze the screenshot and combine it with the existing list.

Gamma's ability to take a wordy presentation and make it less wordy, or more recently to take a 40 minute presentation and compress it into a 20 minute one targeted at a particular audience are great examples of how chat interfaces make that interaction much better.

These natural language superpowers are impressive, but they shouldn't come at the cost of traditional GUI interactions. The best implementations let users move fluidly between both modalities.

Fluid mixing of GUI and Chat

The best integrations don't lock you in GUI or chat mode. You can select objects using the GUI and then give instructions in chat about what you've just selected ("I need you to brighten up the 3 images I just selected"). You can use the chat to select items by describing what you want and then use the GUI to modify them: e.g. "Select all the rectangles on this slide" and then use an opacity slider in the GUI. These are things where fluid transitions between GUI and chat allows the user to actively optimize what works better.

This bidirectionality is crucial. A designer might use chat to ask "Find all images that don't have alt text," then manually add the alt text through the GUI for each one. Or they might drag-and-drop to rearrange elements visually, then ask the chat to "Apply the same styling to all of these." The

power comes from letting users choose the best modality for each micro-task within a larger workflow.

The litmus test for this is what I call "**The Round Trip Test"*:** Can a user start an action in the GUI, refine it with chat, and then finalize it back in the GUI without losing context? For example, selecting a group of images with the mouse, asking the chat to "make these images 25% smaller and apply a grayscale filter," and then manually adjusting the brightness of one of them using a GUI slider, all in one seamless flow.

But even with great primitives, powerful natural language capabilities, and fluid modality switching, there's one more critical requirement that many integrations fail to meet.

***Maintain context**

Chat users frequently have some high level goal they are trying to accomplish that can take dozens of steps. One of the most effective ways to make them productive is to let the context flow between these tasks, often the information from earlier tasks can simplify subsequent tasks. If you know the content of the slide deck and you want to use an AI image generator, the chat has a pretty good idea of what that image needs to match in terms of the surrounding text.

This sounds so obvious, but so many chat integrations break this. In several of these applications there is one chat integration for the image generation which is *completely independent* of the chat that the user was having about the words on the slide. The right way is that the user has one conversation about everything in front of them.

The worst thing you can do is ask the user to carry the context between chat integrations in the same application. You are forcing the user to treat your product like multiple products rather than a single integrated whole. If that's the approach you take, don't be surprised if the user turns to other tools: if the user has to manually craft the prompt for an image in your slides, hey, they might as well paste it into someone else's image generator. You're basically daring them to leave your application.

Why does this happen? It is because AI is *extremely* challenging organizationally. Many organizations were set up with different components maintained by different teams. This made sense when there was a GUI and you could literally cut up the screen and say this tool belongs to team X. AI forces teams to work together in the user's best interests in a way that doesn't easily match the "swim lanes" that have been set up in the past.

As a result, the cardinal product sin emerges: shipping the org chart (also known as Conway's Law). The image gen team builds its chat integration for generating images. The text editing team builds a chat integration for editing text. And the customer experience team builds a chat integration for user interaction. And you end up with 3 chat integrations that are barely aware of one another.

Again, tools like Notion and Gamma have worked out how to overcome this. Gamma has the advantage of being AI native: it came into existence after the AI revolution. Notion had strong support for great unified user-facing design and that helped it get through this.

The litmus test for this is simple: how many chat integrations does the user have to navigate through to accomplish a complex task? Ideally the answer is 1. If it's more than one, then there should be very good reasons for it.

Conclusion

We are just at the beginning of the era of chat integration into existing products — we still have a long way to go. However, there are definitely characteristics that we are seeing arise from the people who do this best; and unfortunately counterexamples of how to do it wrong too.

Those characteristics sometimes seem obvious: can actually do useful operations on the key objects in the app, take advantage of the power of natural language, allow fluid mixing between GUI and chat based interaction and finally maintain a consistent context across the user's entire experience. If a chat integration can nail these four things, its chances of success are greatly increased.

Ultimately the challenges of chat integrations aren't just technical but also organizational. The products that succeed will be those built by teams who break down their internal silos to mirror the unified, context-aware experience they want to deliver to their users. "Shipping the org chart" has never been more problematic.

Artificial Intelligence | Product Management | UX

Published in Bootcamp

95K followers · Last published 7 hours ago

Follow

From idea to product, one lesson at a time. To submit your story: https://tinyurl.com/bootspub1

Written by Waleed Kadous

460 followers · 4 following

Follow

Agent whisperer; AI for Good self-appointed missionary; fmr Chief Scientist @ StockApp & Anyscale; ex-Principal Engineer++ @Canva, Uber, Google; PhD in AI

Responses (8)

Bgerby

What are your thoughts?

mohamad shahkhajeh
3 days ago

It's surprising how often chat integrations fail to support the core actions users expect from the app itself.

1 1 reply Reply

Rinfo
13 hours ago

Great insights on AI chat integrations! On a related note, Textideo(https://textideo.com/model/sora-2?utm_source=info12138&utm_medium=medium&utm_campaign=1010c) is revolutionizing AI video generation with its Sora 2 model, offering powerful video creation tools. Worth checking out!

Reply

Ankit Upadhyay
1 day ago

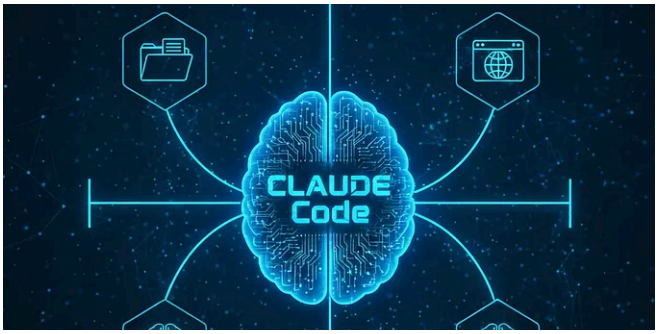
chat integrations fail when they can't do the app's core actions like adding todos, editing docs, or sending messages. good ones use natural language to handle complex tasks, combine data, or summarize content.

they let you switch between gui and... more

Reply

See all responses

More from Waleed Kadous and Bootcamp

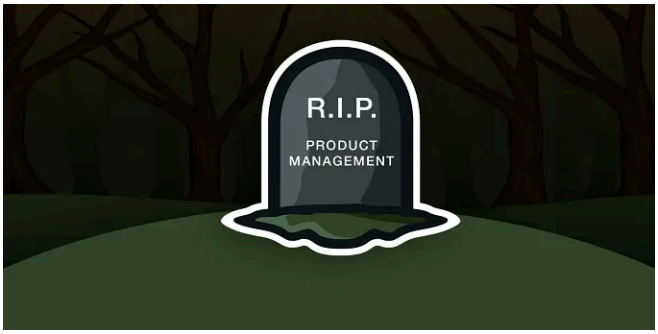


Waleed Kadous

Beyond Code with Claude Code: Why My Daily Driver AI Isn't What...

TL;DR: Advanced users should consider Claude Code as their daily driver AI, not just...

Sep 15 67 1



In Bootcamp by Joe Smiley

The Death of Product Management

The Product Manager (PM) role has gained prominence in recent years as companies...

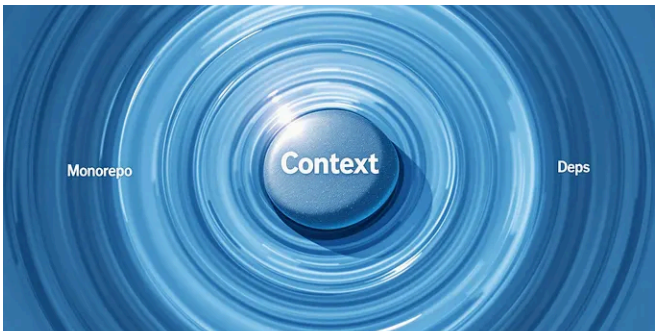
Aug 26 540 35

In Bootcamp by Kartscrut

Game theory is the cheat code to life

Game theory can explain humanity's biggest problem

Aug 26 2.1K 65



Waleed Kadous

Give Claude Code Context: One Principle, Many Implications

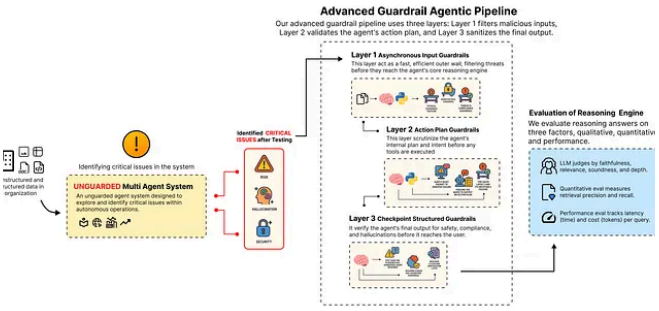
TL;DR: Providing Claude Code with relevant context significantly improves its output...

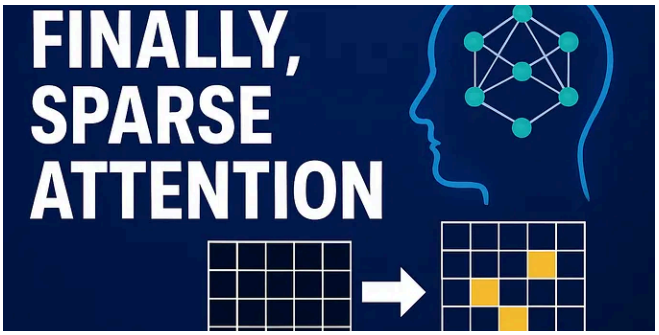
Mar 18 54 2

See all from Waleed Kadous

See all from Bootcamp

Recommended from Medium





In Level Up Coding by Fareed Khan

Building a Multi-Layered Agentic Guardrail Pipeline to Reduce...

Layer 1 for Input, layer 2 for Planning, layer 3 for Output

4d ago 464 3

Ignacio de Gregorio

DeepSeek is Finally Back, Solving Sparse Attention.

A Years-old Mystery, Solved

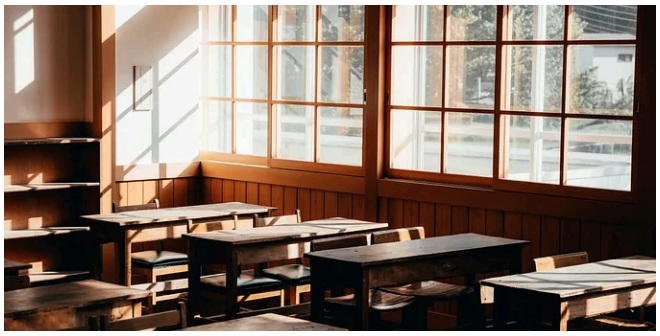
4d ago 960 18

In The Generator by Jim the AI Whisperer

The words “blah blah blah” increase AI accuracy

Who needs Chain of Thought when “blah blah blah” works?

6d ago 3.8K 79

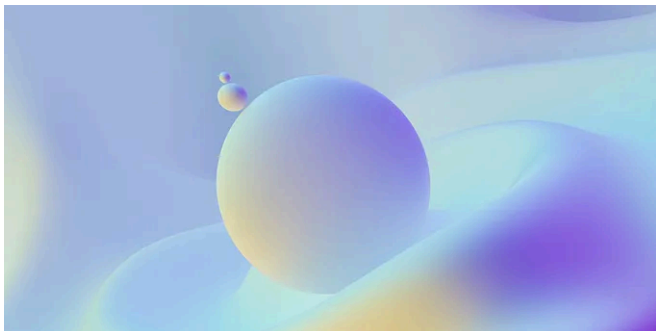


In Versent Tech Blog by Mathew Hemphill

Training an LLM with Hugging Face

Beginners Guide to fine-tuning an LLM

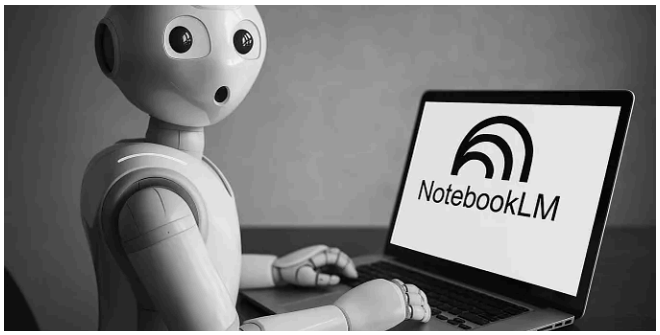
Sep 30 242 8



Will Lockett

Is The AI Bubble About To Pop?

The $100 billion red flag.



In Data Science Collect... by Amanda Iglesias Mor...

NotebookLM Just Got a Serious Upgrade—Exploring...

What’s New and Why It Matters

5d ago 2.1K 63

Sep 29 620 15

See more recommendations