

Chapter 7. Chatbots and Virtual Assistants

Chatbots have been a staple of digital customer service and automation for over a decade. Initially built as simple rule-based programs, traditional chatbots followed predefined logic trees to handle repetitive tasks, such as answering frequently asked questions, providing automated responses in customer support, or collecting user information in structured workflows. These chatbots weren't very powerful compared to today's LLM-based bots, but they became a very popular way to automate high-volume, low-complexity interactions. However, their technical implementations struggled with unstructured conversations, ambiguity, or user inputs that deviated from expected patterns. The experience usually felt rigid and unnatural—people clearly understood they were talking to a machine, and those user interactions would frequently end with frustrating statements like, “I don't understand. Please try again.”

Now that people are used to chatting with LLM-based tools like ChatGPT, the rule-based approach to chatbots seems like a distant memory. Today's AI-powered chatbots no longer rely solely on predefined scripts and rules. Instead, they leverage sophisticated natural language processing (NLP), contextual understanding, and generative AI to deliver dynamic, humanlike interactions. These advancements mean the role of chatbots has significantly expanded to include tasks like:

- Understanding complex queries and responding with relevant, well-formed answers
- Maintaining memory across conversations, providing continuity and context
- Taking actions on behalf of users, such as booking appointments or retrieving personalized data
- Learning and adapting over time, improving based on real-world interactions
- Integrating with external APIs and databases to fetch real-time information

This transformation means that software engineers now have a broad spectrum of tools and frameworks available to help us build intelligent chatbots, ranging from no-code workflow builders to fully customizable AI-powered assistants that leverage agentic reasoning and training data. Whether you want to build a simple automated helpdesk bot or a sophisticated AI agent capable of complex decision-making, the technology is now more accessible than ever.

Types of Chatbot Implementations

There are three main options for implementing a chatbot:

No-code AI chatbots

These tools promise that users without any programming knowledge will be able to deploy AI-powered assistants. They usually have easy-to-use interfaces where users can upload datasets, define workflows, and select the underlying AI models. While these tools are promoted for a nontechnical audience, I'm including them in the book because, in my experience, certain technical knowledge is often required to integrate these tools with other parts of a company's workflow, and this ends up being part of software engineers' scope of work.

These no-code chatbots are a natural evolution of the rule-based chatbots mentioned previously and cater to the same use cases: customer support, ecommerce, and FAQ automation.

Drag-and-drop chatbot builders

These tools aim to bridge the gap between simplicity and flexibility to adapt to the needs of different clients. They are usually easy to set up and allow users to build conversation flows visually by connecting boxes with predefined functionalities, while integrating AI-powered features for dynamic interactions. Developers can automate workflows and use more complex logic in the backend for the chatbot.

These chatbot builders cater to use cases such as customer service, lead generation, and internal workflow automation.

Code-based AI frameworks

These are usually SDKs, APIs, or open source repositories that allow software engineers to fully control the chatbot's behavior, which they'll then use as boilerplate to implement customized chatbot experiences. They tend to have higher standards for training data. Use cases include agentic reasoning, integration with other tools and workflows, and even data security (since these chatbots can be self-hosted on a company's cloud infrastructure).

Evaluation Process

To compare chatbot-building tools, I created a chatbot that interacts with this [Online Retail Dataset](#), created by Lucas Soares, a fellow O'Reilly author in the field of machine learning. It consists of transactional data from an ecommerce shop; I've used it to test chatbots' capabilities to retrieve factual data and answer complex user queries.

Each chatbot was tested using a set of predefined queries, evaluating its:

- Ease of setup and deployment
- Ability to retrieve factual information
- Context retention across multiple interactions
- Ability to handle complex queries

I tested multiple tools, but I have included one from each of the categories outlined in the previous section.

Let's begin with a no-code option: Chatbase.

Chatbase

[Chatbase](#) was launched in 2022 as a no-code AI chatbot builder that allows users to create and train chatbots by uploading documents or datasets. It gained popularity due to its simple interface and integration with OpenAI's GPT models, which made it an accessible solution for businesses looking to automate their support and customer interactions without the need for technical expertise. While the product is still relatively young, Chatbase has seen adoption among small businesses, solopreneurs, and startups, particularly for handling customer support and FAQ automation. The tool offers a free-tier option and a paid plan for scaling your chatbot's capabilities.

For this test, I created a free account on the Chatbase website and looked for a way to create my chatbot. I quickly found the place to add the training data, my spreadsheet with the ecommerce product catalog (see [Figure 7-1](#)). In a matter of two or three minutes, I was interacting with my chatbot.

Sources

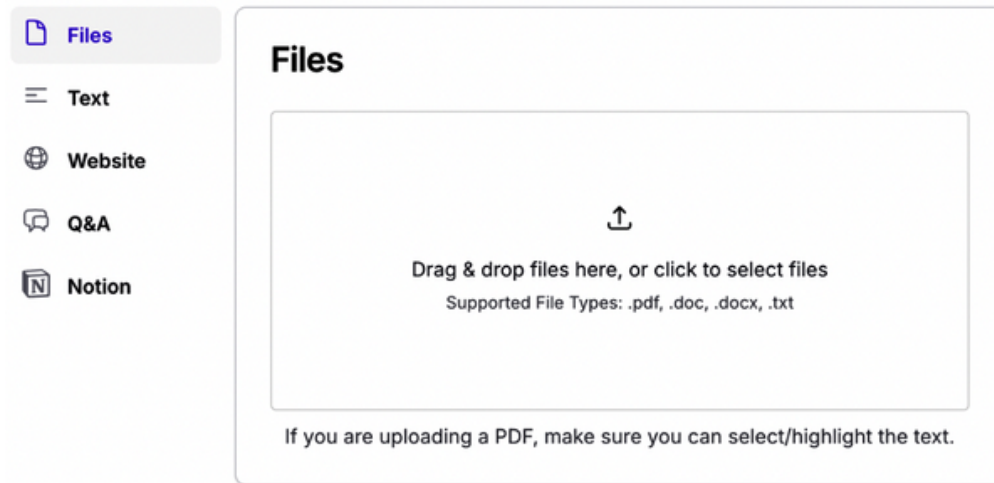


Figure 7-1. Chatbase's Sources UI, where users can add training data

Once the chatbot was ready, I started asking questions, as if I were a real client:

I see there are some Android phones in your catalog. Can you list their features and prices, and tell me the best budget Android phone?



It did well with the first question, fetching correct results and displaying them in a well-formatted response, as shown in [Figure 7-2](#).

I see there are some Android phones in your catalog. Compare the features and prices, and tell me the best budget Android phones?

Sure! Here are a few budget Android phones from the catalog along with their features and prices:

1. **Samsung Galaxy A3 (2016)**

- **Display:** 4.7 inches
- **Storage:** 16GB
- **Connectivity:** 4G
- **Price:** £149.99
- **Rating:** 2.4

2. **Huawei Ascend P2**

- **Display:** 4.7 inches
- **Processor:** 1.2 GHz dual-core
- **Camera:** 8 MP rear camera
- **Price:** £299.99
- **Rating:** 3.5

3. **Motorola Moto G4 Plus**

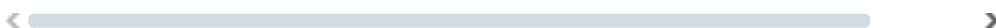
- **Display:** 5.5 inches



Figure 7-2. First question to my Chatbase chatbot (truncated to the visible area)

In the second question, I purposely asked about a product that doesn't exist:

I'm looking for a laptop below \$10. What's the best opt



This is a common source of hallucinations, and once again, my Chatbase bot did well, stating that no product matches my query (see [Figure 7-3](#)).

I'm looking for a laptop below \$10. What's the best option?

Unfortunately, there are no laptops listed in the catalog for under \$10. In fact, laptops typically start at a higher price point. If you're looking for budget-friendly options, I recommend considering a slightly higher budget to find a suitable laptop.

If you have any other questions or need assistance with something else, feel free to let me know!

Figure 7-3. Second question to my Chatbase chatbot

For the third question, I asked something more complex, not only inquiring about specific products in the catalog, but also asking Chatbase to compare their attributes and ultimately provide a recommendation:

You have mobile phones and digital cameras. I'm interested in photography, should I buy the Sony Xperia XA2 Ultra or K-1 camera? Why?



Again, my Chatbase bot did well, stating the attributes of each option and making recommendations about when to buy one or the other (see [Figure 7-4](#)).

Chatbase also makes it very easy to publish a chatbot using its Connect section. With a couple of clicks, I get some code to embed my chatbot in a website via a chat bubble or iframe. I was impressed by how easy it was to upload training data, test the bot, and publish it.

Chatbase did very well in the test, and it's probably one of the easiest ways to integrate a chatbot into a website, if you have specific training data and make some changes to the prompt to get the desired functionality, tone, and so on. As such, I rate it a 9/10. I didn't rate it 10/10 because the end product's behavior significantly resembles the default behavior of the underlying LLM (gpt-4o-mini, in this test). In other words, if a user were to upload the same training data to ChatGPT and ask these questions, the answers would be very

similar. Still, the ease of doing all this and publishing the bot with a few clicks is impressive.

Figure 7-4. Third question to my Chatbase chatbot

Botpress

Drag-and-drop option [Botpress](#) was founded as an open source chatbot-development platform back in 2017, when that meant rule-based chatbots. The current version of the product offers a visual builder that enables users to create complex conversational logic and to integrate the resulting bot with third-party APIs and other applications. In recent years, Botpress's adoption has grown, especially among midsize enterprises and organizations looking for an alternative to closed source chatbot platforms. With a modular architecture and self-hosting options, it appeals to businesses that need to comply with data privacy regulations while maintaining scalability.

To test Botpress, I followed the same playbook I used for Chatbase. I created a free account on the Botpress website and went through its widget to build an agent (that's what it calls a chatbot). It took me some time to figure out how to add the training data, since I first needed to create a table (its name for training data in CSV format) on a separate page, then add that as a knowledge base to my bot (see [Figure 7-5](#)).

Figure 7-5. Botpress’s Agent Builder UI, where users can add training data and configure their workflow

Botpress offers more options beyond a simple chatbot feature. Users can add different agents and tools and compose a workflow with them that serves as the backend reasoning under the bot’s hood (see [Figure 7-6](#)). It’s more powerful than Chatbase; however, getting a simple bot up and running took me 30 to 40 minutes of trial and error.

Figure 7-6. Botpress’s workflow builder—I selected a very simple flow for my test

With the chatbot finally ready, I started interacting with it, asking the exact same questions I asked the Chatbase bot (see [Figure 7-7](#)).

Figure 7-7. First question to my Botpress chatbot

For the first question, the bot correctly fetched three budget Android phones; however, it also offered an option priced at \$0, which is clearly a data error that it didn't flag to me. I'd be a bit frustrated with this suggestion if I were a real client.

Again, in the second question I purposefully asked about a product that doesn't exist, and my Botpress bot did well, correctly stating that no product matched my query (see [Figure 7-8](#)).

Figure 7-8. Second question to my Botpress chatbot

Third, I once again asked a complex question about comparing the phone with the camera. My Botpress bot did well, stating the attributes of each option and when it recommended buying one or the other. The format wasn't great, however, since each product's attributes were packed into a single bullet point (see [Figure 7-9](#)).

Figure 7-9. Third question to my Botpress chatbot

Botpress also features a one-click Publish feature that creates a standalone URL for your chatbot and provides simple guidelines for the most popular use cases (implementing chat bubbles, embedding into a website, and so on).

Botpress did well in the test. The lack of polish in some answers might be due to its use of an older model by default on the date of this test (gpt-4o-2024-08-06). It has much more comprehensive workflow features than Chatbase does;

however, the UI is quite challenging for a new user to navigate. I rate it an 8/10.

LangChain

The code-based framework [LangChain](#) emerged in 2022 in response to the increasing demand for developer-friendly frameworks to build AI-driven applications. It's designed to integrate with popular LLMs like GPT-4, Claude, and open source alternatives, and to allow software engineers to create custom AI-powered assistants with advanced reasoning capabilities, memory, and dynamic API integrations. The framework has gained rapid adoption among AI engineers and startups looking to build agent-based systems that go beyond basic chatbot interactions, and it has become the go-to framework for LLM-powered applications and AI agents.

For this test, I coded my own chatbot using the LangChain framework. It's a very well-documented framework, so it was rather easy to get started. It took me about 30 to 40 minutes to get from an empty repository to the chatbot in the following images (with help from Cursor for boilerplate, the verbose chunks of the code, and some formatting adjustments at the end). The resulting code is available in the book's [GitHub repository](#). I ran it on my local machine and used the gpt-4o-mini model for this test.

For the first question, the bot correctly fetched three Android phones, suggested one that's actually a good budget phone, and explained its reasoning, which was also good (see [Figure 7-10](#)).

Figure 7-10. The beginning and end of my first question and the response from the chatbot I coded with LangChain (the verbose middle section is not displayed)

For the second question, about the product that doesn't exist, the bot indeed realized that no such product existed (see [Figure 7-11](#)). The reply "I don't know" results from a literal interpretation of the section of the prompt that says, "If you don't know the answer, just say you don't know." This is quirky, but actually correct.

Figure 7-11. Second question to my LangChain chatbot

On the complex question comparing the phone with the camera, my LangChain chatbot did well, stating the attributes of each option and the trade-offs of choosing one or the other (see [Figure 7-12](#)). Its answer is very much in line with those of the bots I created with the other tools, which illustrates how much of the output can be attributed to the underlying models.

Figure 7-12. Beginning and end of my third question and my LangChain chatbot's response (verbose middle section not displayed)

LangChain doesn't offer a one-click publish option. I ran this code on my local machine, but to make the bot publicly accessible, I would need to deploy it on a cloud server. Also, LangChain offers no off-the-shelf guidelines for integrating my chatbot as a bubble chat or embedding it in other people's websites; that would need to be developed as desired.

I've implemented several real-world chatbot solutions with LangChain, and I like it a lot. It allows you to build simple chatbots or more complex ones, as well as multiagent workflows using LangGraph. It's a powerful solution, but it certainly requires software development skills. It's also very well documented, with a great developer community. With assistance from other AI tools, like Cursor and GitHub Copilot, these implementations take a fraction of the time they'd have taken just a few years ago. As such, I rate LangChain a 10/10.

Tool Comparison

All of the tools in this test performed very well, which is why they're included here (see [Table 7-1](#)). However, LangChain will continue to be my go-to tool for building any kind of chatbot.

Table 7-1. AI chatbot tools overview

Tool	UX	Test performance
Chatbase	No-code AI chatbot	9/10
Botpress	Drag/drop chatbot builder	8/10
LangChain	AI code framework	10/10

Conclusion

As a CTO, I’ve built many chatbots over the years, mostly before the recent generative AI wave, with functionality limited to predefined logic trees and very niche use cases.

The tests for this chapter highlight that it’s now possible to build a highly capable chatbot with minimal effort. These chatbots are capable both in terms of being trained on specific data, and also in their tone, range of inference, and ability to follow the specifics of the desired use case. It should be no surprise that thousands of businesses are integrating these capabilities into their products. It’s also interesting to observe the massive shift in UI patterns, away from the traditional list→search→detail approach and toward chatbot UIs that allow the user to “talk with the data” in a less structured way.

It’s worth noting that this shift toward chatbot UIs creates room for hallucinations, partial responses, and other well-known quirks of LLM technology. Indeed, my main takeaway from writing this chapter is that the outputs from all the tools in this test were highly dependent on the underlying LLMs they used—for better and for worse. It’s great that, with minimal action, a user can set up an impressive chatbot that fetches information, structures it correctly, and draws complex conclusions. But the chatbot’s output is still very much subject to the limitations of the underlying LLM, such as hallucinations.