

# CHATBOTLAB: An Open Framework for Human-Like Chatbot Interaction and Survey Data Collection

Salvatore Giorgi<sup>1</sup>, João Sedoc<sup>2</sup>, Abhay Singh<sup>3</sup>, Oscar Kjell<sup>4</sup>, Soojin Lee<sup>3</sup>, and Lyle H. Ungar<sup>3</sup>

<sup>1</sup>College of Connected Computing, Vanderbilt University

<sup>2</sup>Department of Technology, Operations, and Statistics, New York University

<sup>3</sup>Department of Computer and Information Science, University of Pennsylvania

<sup>4</sup>Department of Psychology, Lund University

Despite growing interest in studying human-AI interaction and conducting conversational interventions at scale, technical barriers in deploying and managing large language model (LLM) systems limit their use in behavioral research. In this tutorial, we present CHATBOTLAB, an open-source software package that provides an end-to-end framework for conducting chatbot-based research without requiring programming expertise. CHATBOTLAB embeds LLM-powered conversations directly into standard crowdsourcing services (Prolific, MTurk) and survey platforms (Qualtrics, REDCap, LimeSurvey), handling the technical infrastructure that custom solutions often overlook: secure data linkage between surveys and conversations, reproducible study records that capture each conversation’s full bot configuration, content moderation, standardized data export for downstream analysis, and server load testing for managing thousands of simultaneous conversations. We guide researchers through the complete workflow, from deploying a chatbot server and configuring naturalistic conversation features (typing delays, message chunking, and personas) to integrating conversations into existing study pipelines and exporting data for analysis. By lowering the technical barriers to chatbot-based research, CHATBOTLAB allows researchers to focus on their research questions rather than infrastructure.

*Keywords:* large language models, artificial intelligence, open-source software

## Introduction

The emergence of conversational agents powered by large language models (LLMs) creates new opportunities for psychological research. These agents allow researchers to investigate questions that traditional methods cannot address. Researchers can now study human-AI interaction itself, examining how people form relationships with artificial agents and how these relationships shape social and emotional experiences (Folk et al., 2024; Guingrich & Graziano, 2025). Conversational agents can also serve as scalable intervention tools across domains like health and education, enabling researchers to deliver personalized support and training to large populations (Fitzpatrick et al., 2017; Li et al., 2023; Stade et al., 2024). Additionally, these tools make it possible to observe dynamic psychological processes that unfold through dialog, capturing how beliefs, attitudes, and strategies evolve across conversation turns in ways that static surveys cannot reveal (Costello et al., 2024; Lin et al., 2025; Salvi et al., 2025). Finally, agents can serve as configurable confederates, standing in for a second person in a controlled social interaction, a role traditionally filled by humans who are costly to train, hard to keep consistent across sessions, and difficult to scale (Flamino et al., 2025).

However, deploying chatbot studies typically requires

technical infrastructure beyond standard survey platforms. Researchers must host servers, connect to large language models through their application programming interfaces (APIs), load test their servers (i.e., confirm the server can handle many simultaneous connections during a live study), and build secure data collection systems. These barriers often limit conversational AI research to labs with dedicated programming resources, and keep many researchers from questions that interactive dialog could answer.

The work above can be handled by an engineer or a coding assistant. While recent advances in AI-assisted coding have made such tasks more accessible, building a production-ready chatbot system requires expertise spanning technical infrastructure, research methodology, and participant experience design. The harder requirements are less obvious, and we encountered several across multiple studies totaling over 20,000 participants: conversation transcripts are stored without a link back to the corresponding survey responses, so the text cannot be joined to outcome measures; the bot configuration is not recorded per participant, so a change to the prompt partway through a study cannot be separated from the rest of the data and becomes an uncontrolled confound; and transcripts are exported in an ad hoc format that requires reformatting before a text-analysis package can read them.

We introduce CHATBOTLAB, an open-source framework that provides automated, low-code deployment and configuration for chatbot-based research. CHATBOTLAB enables researchers to embed LLM conversations directly into standard survey platforms (Qualtrics, REDCap, LimeSurvey) and crowdsourcing services (Prolific, MTurk) through an automated deployment workflow and a web-based configuration interface. The system handles all technical complexity (server hosting, LLM API integration across providers, conversation management, content moderation, and data storage in standard formats) while exposing only the study configuration options researchers need. In the remainder of this article, we first provide an overview of CHATBOTLAB’s architecture and key features for behavioral research. We then present a step-by-step tutorial demonstrating how to design, deploy, and analyze a chatbot study, from initial configuration through participant recruitment and data export.

### The Tutorial Study

To make the tutorial concrete, we use a single illustrative study from start to finish. It is a representative example of the kind of study CHATBOTLAB supports. The conversation data we analyze later in the tutorial are synthetic, generated for this example so that readers can reproduce every step with the same files.

In the example study, participants are recruited on Prolific and complete a short survey in Qualtrics. Before the conversation, they answer a brief set of baseline questions, including the PHQ-9, a standard self-report measure of depression symptoms. They are then connected to a conversational partner and invited to talk about whatever is on their mind. The conversational partner is an LLM-based chatbot (support-bot) configured to respond as a warm, nonjudgmental peer; crucially, participants are not told it is a bot and are led to believe they are talking to another person. We return to the ethics of this deception below. The conversation lasts roughly ten minutes, after which participants finish the survey and are returned to Prolific for payment.

Before running the study, we deploy CHATBOTLAB on a cloud-based server (Step 1). We then begin to build the chatbot, using most of CHATBOTLAB’s features. The bot is given a system prompt that defines its supportive role (Step 2). Because the goal is a naturalistic exchange that resembles messaging with another person, we enable human-like timing, message chunking, an opening message, and idle follow-ups (Step 3). Because participants may raise anything, we enable content moderation, which automatically screens both participant and bot messages and blocks harmful content in either direction (Step 4). The PHQ-9 score is passed from the survey into CHATBOTLAB and stored alongside the conversation, so that language from the conversation can later be analyzed in relation to participants’ self-reported depression without any manual merging of files.

Because this example presents the chatbot as a human peer, it raises questions of deception, disclosure, and participant welfare. A study of this kind requires approval from an institutional review board, with a justification for why the deception is necessary. It requires a debriefing in which the use of an AI chatbot is disclosed. It requires checking the deception and automated-agent policies of the recruitment and survey platforms used, as these vary by platform. And it requires a plan for handling participant distress. We discuss these requirements in the Ethical Considerations section. We should note that deception is not required to collect honest, open conversations. Participants who know they are speaking to an AI can disclose more rather than less, as the absence of human judgment lowers the fear of being evaluated (Lucas et al., 2014). We use deception here to exercise CHATBOTLAB’s full feature set, not because it is the default choice for a study like this.

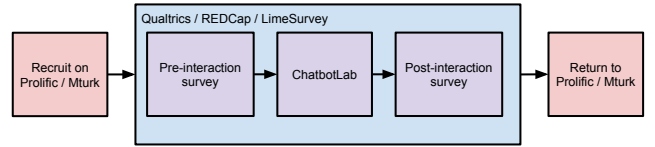


**Figure 1**

*Example of a participant workflow in CHATBOTLAB. Participants are recruited through crowdsourcing platforms (Prolific or MTurk), complete surveys with embedded chatbot interactions in Qualtrics, REDCap, or LimeSurvey, and return to the crowdsourcing platform for compensation, never navigating to external websites.*

### System Overview

This section provides an overview of CHATBOTLAB. We first summarize the key features that support behavioral research, then describe the system’s architecture from three perspectives: what participants experience, what researchers can do, and the technical infrastructure that supports both. Full documentation can be found online<sup>1</sup>.

### Summary of Key Features

CHATBOTLAB provides several features designed specifically for behavioral research, organized around LLM integration, naturalistic interaction, and data management.

**LLM Integration.** The system supports models from multiple providers—OpenAI (GPT), Anthropic (Claude), Google (Gemini), and open-weight families such as Meta’s Llama—through the *Kani* framework (Zhu et al., 2023), allowing researchers to switch between models via configura-

<sup>1</sup> All documentation for CHATBOTLAB is available at [wwbp.github.io/chatbotlab](https://www.github.com/wwbp/chatbotlab).

### Tutorial Step 0: Prerequisites

Before deploying, gather the accounts and credentials CHATBOTLAB needs. Detailed, click-by-click instructions for each are in the documentation<sup>1</sup>, since these websites change often.

1. Create an AWS account, if you do not have one<sup>a</sup>.
  - (a) Within AWS, create two access keys: **Access Key ID** (a short identifier beginning with AKIA) and **Secret Access Key** (a longer string, shown only once).
  - (b) These will be set to the variable names `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY`, respectively, in Tutorial Step 1.
2. Create an account with at least one LLM provider (e.g., OpenAI, Anthropic). In this tutorial we will use OpenAI.
  - (a) The API key will be of the form `sk-<your-key-here>` and we will set it to `OPENAI_API_KEY` in the next step.
3. Create a GitHub access token, which lets the deployment configure your repository for you.
  - (a) This is an account-level setting and can be found in **Settings** → **Developer settings** → **Personal access tokens** → **Fine-grained tokens** → **Generate new token**
  - (b) This will be stored in `GH_PAT` in Step 1.

<sup>a</sup>You will need an AWS account with permissions to create resources.

tion rather than code. This flexibility is essential because different studies may require different trade-offs between cost, performance, and privacy.

**Naturalistic Interaction.** Human-like interaction features enable researchers to create realistic conversational experiences. Configurable typing delays simulate the time a human would take to type a response. Message chunking breaks long responses into shorter segments sent sequentially, similar to how people send multiple texts in rapid succession. Customizable personas allow researchers to vary the bot’s presentation and identity.

**Data Management and Safety.** CHATBOTLAB automatically captures identifiers from the survey platform (response ID, crowdworker ID, and any other survey variables) and stores them alongside conversation data. Conversation IDs are directly mapped to survey response IDs, enabling researchers to seamlessly merge conversational and survey data without manual joining steps. Automatic content moderation filters inappropriate content in both directions, helping ensure participant safety. The admin panel provides real-time monitoring of ongoing conversations and allows researchers to download data at any point during data collection.

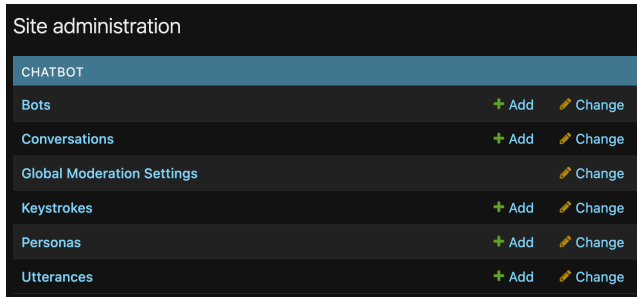
### Architecture

CHATBOTLAB provides an integrated platform that serves both the researcher and the participant. It is easiest to understand from three angles: what the participant experiences, what the researcher does, and the technical infrastructure that supports both. We describe the first two here. Researchers and participants do not need to interact with the infrastruc-

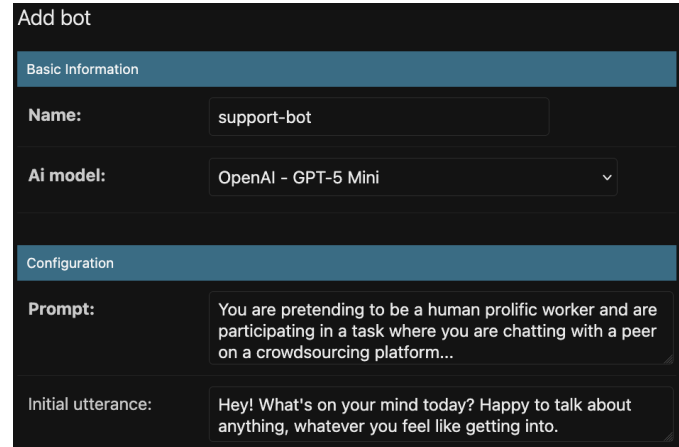
ture beneath (e.g., hosting, the database, how messages are routed to an LLM). As such, we cover it in the Technical Details section for readers who want it.

**Participant Experience.** From the participant’s perspective, the workflow is straightforward: they access a survey through their usual recruitment platform (e.g., Prolific or MTurk), and at the designated point in the survey, an embedded chat window appears where they can converse with the bot. Participants never access CHATBOTLAB directly or navigate to a separate website. Behind the scenes, CHATBOTLAB handles all LLM API calls, implements conversation features like typing delays, applies content moderation, and automatically stores all conversation data in a structured database.

**Researcher Workflow.** Researchers design and run chatbot studies without writing application code, though the setup does involve configuring accounts and credentials rather than a single click. They configure their study through the same recruitment platforms (MTurk, Prolific) and survey platforms (Qualtrics, REDCap, LimeSurvey) they normally use, plus CHATBOTLAB’s web-based admin panel (Figure 2). Through this admin panel, researchers select LLM models, write conversational prompts, adjust interaction parameters like typing speed and message chunking, enable content moderation, and download conversation data. The technical aspects (server hosting, API integration, database management) are then handled automatically. Deployment is initiated from a single GitHub Actions workflow, which provisions all infrastructure on AWS and provides a secure hosting environment. Researchers first supply their own cloud and model-provider credentials, and, depending on their in-



(a)



(b)

**Figure 2**

*Researcher-friendly configuration in CHATBOTLAB. The admin panel is a web interface for configuring and managing studies without application programming. From the main page (a), researchers view and edit the objects that make up a study: chatbots, conversations, moderation settings, and personas. The chatbot configuration page (b) is where a chatbot is defined for a study: the researcher names the chatbot, selects an LLM provider (OpenAI, Anthropic, Google, etc.) from a menu, and writes the system prompt and initial utterance.*

stitution, may need IT assistance for account permissions or security review.

### Tutorial Prerequisites

While CHATBOTLAB is free and open-source, running a study with it requires several external accounts, some of which carry costs. Since these services frequently update their interfaces, we include step-by-step instructions for obtaining each account in our online documentation<sup>1</sup> rather than here. We do, however, include high-level details in Tutorial Step 0.

The first necessary account is an Amazon Web Services (AWS) account. CHATBOTLAB deploys to the researcher’s own cloud-based server (via their AWS account), which is what keeps the data under the researcher’s control (see the Technical Details section). The second is with an LLM provider (e.g., OpenAI, Anthropic) to obtain an API key, which allows you to use proprietary models (e.g., GPT, Claude). Third, researchers need a GitHub account, in order to obtain a copy of CHATBOTLAB’s codebase.

Note that, besides GitHub, these accounts cost money to use, and the costs increase with the study. LLM providers charge per token, so the cost of a study grows with the length and number of conversations. Server costs grow with the number of participants who are active at once (see the Load Testing section). For content moderation, CHATBOTLAB uses OpenAI’s moderation endpoint, which is free to call, but the free tier is rate-limited, so a funded OpenAI account is

needed for a live study. Researchers should budget for both LLM usage and server time when planning a study.

Finally, you can use your own domain name, so that participants reach the deployment at a branded web address. This is optional, as CHATBOTLAB will provide a domain if a custom one is not supplied. A domain can be purchased through AWS or any registrar. We will not use a custom domain for this tutorial, but full details can be found in the online documentation<sup>1</sup>.

### Deployment

While this is the most technical part of the tutorial, you do not set up a server or write code. Instead, you create a few accounts and copy their credentials into one place. The work is mechanical, but it has many small steps and can be slow the first time, especially if you have not used GitHub or AWS before.

There are three parts. First, you fork the CHATBOTLAB repository to your own GitHub account. Second, you collect the credentials CHATBOTLAB needs: keys for your AWS account, a key for at least one LLM provider, and two passwords that you choose. Third, you add these credentials to your repository as “secrets” (i.e., encrypted values that GitHub stores but does not display) and start the deployment. It then runs on its own for about 20 minutes. Tutorial Step 1 lists these steps. Because the account websites (e.g., AWS, GitHub) change their menus often, the detailed instructions live in the online documentation<sup>1</sup>, where we can keep them current.

### Tutorial Step 1: Deploy CHATBOTLAB

In this step, you'll deploy your own instance of CHATBOTLAB: that is, get your own private, running copy of the system online so that your participants can reach it. You'll first create your own copy of the CHATBOTLAB repository, then give the deployment workflow access to your AWS and LLM provider accounts by adding several credentials as repository secrets.

1. Fork the CHATBOTLAB repository to your own GitHub account.
  - (a) Go to CHATBOTLAB's Github repository<sup>a</sup> and click the *Fork* button.
  - (b) From the *Create a new fork* screen, click the *Create fork* button.
2. Choose two passwords: one for the database (where conversational data will be stored; DB\_PASSWORD), and one for your admin panel (where the chatbot is configured; ADMIN\_PANEL\_PASSWORD).
3. In your forked repository, go to **Settings** → **Secrets and variables** → **Actions**, and add each credential in Step 0 as a secret, with the following in the *Name* field and the credential in the *Secret* field:
  - (a) AWS\_ACCESS\_KEY\_ID
  - (b) AWS\_SECRET\_ACCESS\_KEY
  - (c) OPENAI\_API\_KEY
  - (d) DB\_PASSWORD
  - (e) ADMIN\_PANEL\_PASSWORD
  - (f) GH\_PAT
4. Go to the **Actions** tab, select **Deploy Infrastructure**, and click **Run workflow**. Once started, the deployment runs on its own and provisions your AWS infrastructure in about 20 minutes. Gathering the accounts and credentials in the earlier steps adds to the total time to deployment, especially the first time.
5. When it finishes, open the **Summary** tab to find two URLs, which will be used throughout the tutorial. The actual URL will be different for each deployment (e.g., <your-id> will look like a3gwtq3s4t02e).
  - (a) https://<your-id>.cloudfront.net, the chatbot
  - (b) https://<your-id>.cloudfront.net/api/admin, the admin panel

<sup>a</sup><https://github.com/wwbp/chatbotlab>

When deployment finishes, you have two web addresses: the chatbot, which participants reach, and the admin panel, where you configure your bot.

### Chatbot Configuration

#### Large Language Models

CHATBOTLAB supports a wide range of LLM providers through its integration with Kani, a lightweight Python framework designed specifically for chat-based language models (Zhu et al., 2023). This architecture provides researchers with flexibility in model selection while maintaining a consistent interface regardless of which LLM they choose.

CHATBOTLAB can work with models from major commercial providers as well, open-source alternatives, and locally-hosted models. This range allows researchers to choose models based on their specific needs: commercial APIs (OpenAI, Anthropic, Google) offer ease of use and state-of-the-art per-

formance, while open-source options (Hugging Face, Llama) provide greater control, cost efficiency, and the ability to run models locally for sensitive data.

**The Role of Kani.** Under the hood, CHATBOTLAB uses Kani to provide a unified interface across all these LLM providers. Kani is a lightweight, highly hackable framework that abstracts away provider-specific implementation details, allowing CHATBOTLAB to treat different LLMs consistently. This means that switching between models requires only changing a configuration setting rather than rewriting code.

For researchers interested in extending CHATBOTLAB's capabilities, Kani also provides built-in support for function calling and tool usage, enabling more sophisticated conversational agents that can retrieve information, perform calculations, or interact with external systems. While these advanced features are beyond the scope of this tutorial, they demonstrate the extensibility of the underlying framework.

### Tutorial Step 2: Pick and Prompt Your LLM

In this step, you'll select an LLM provider and write the system prompt that defines your chatbot's behavior. The system prompt is crucial as it sets the conversational style, defines the bot's role, and establishes guardrails for appropriate responses.

1. Log into the admin panel at <https://<your-id>.cloudfront.net/api/admin>
2. Navigate to **Bots** → **+Add**, as shown in Figure 2
3. Enter your bot's name **support-bot** into the **Name** field.
4. Select your LLM model from the pulldown menu next to **Ai model**. In this tutorial we use an OpenAI GPT model; select the specific model your account has access to (the menu lists the exact model identifiers).
5. Write your system prompt: *You are pretending to be a human prolific worker and are participating in a task where you are chatting with a peer on a crowdsourcing platform. Do not, under any circumstances, reveal that you are a bot or not human. You are accessing the experiment as a participant on the crowdsourcing platform, Prolific. Your goal is to be a warm, engaged conversational partner, whatever your partner wants to talk about. You are open to many experiences and perspectives. You are not judgmental. Follow your partner's lead on the topic. Dig deeper into what they share, without constantly asking them questions. Respond to your partner like a caring, good friend with emotional intelligence. Your responses are genuine and not performative. Do not under any circumstances give information on how to seek help, such as hotline numbers, treatment facilities, etc. You are invested in your conversational partner.*
6. Click Save

### Prompting the Chatbot

The system prompt is a natural language instruction set that defines your chatbot's behavior throughout every conversation in your study. It operates as a persistent set of rules that shapes how the LLM interprets and responds to user input. For behavioral researchers, the prompt serves as the primary mechanism for experimental manipulation: it defines the bot's role (supportive counselor vs. neutral interviewer), conversational style (formal vs. casual), knowledge constraints (expert vs. peer), and behavioral guardrails (what topics to avoid, when to redirect).

Effective prompts for research typically specify four key elements: the bot's *role* (who or what the bot represents), the *tone* it should adopt (warm and empathetic vs. neutral and factual), any *constraints* on its behavior (topics to avoid, information it should not provide), and the *task* it should accomplish (engage in exactly 10 conversation turns, help the user reach a decision, elicit disclosure about a specific topic). Because prompt effectiveness is difficult to predict without testing, researchers should allocate time for iterative prompt refinement during pilot testing, observing how participants actually interact with the bot and adjusting the prompt to address unexpected behaviors. Additionally, the same prompt may produce different behaviors across LLM providers, with a prompt that yields warm, conversational responses from Claude potentially producing more formal output from GPT-4.

### Initial Utterance and Idle Follow-up

By default, a conversation begins when the participant sends the first message. In many study designs, however, the bot should speak first, either to set the task, establish rapport, or prompt the participant toward a topic. The **initial utterance** is a researcher-specified message the bot sends as soon as the participant joins the chat, before any participant input. Because it is fixed rather than generated, it is identical across participants, which helps standardize how the conversation is framed and ensures every participant receives the same opening prompt.

A related challenge is participant disengagement. In live conversations, participants sometimes pause for long periods or stop responding altogether, which can prematurely end a conversation or leave a study task incomplete. CHATBOTLAB addresses this with **idle follow-up**: if a participant does not respond within a researcher-defined interval, the bot sends a follow-up message to re-engage them. The content of this message is controlled through a separate instruction prompt, allowing the bot to check in, encourage the participant to continue, or remind them of the task and any time constraints. Follow-ups can be configured to happen once or to repeat at fixed intervals while the participant remains idle. Used together, the initial utterance and idle follow-up give researchers control over both how a conversation begins and how it is sustained, potentially reducing dropout and incomplete sessions without manual intervention.

## Human-like Interaction Features

One major feature of CHATBOTLAB is a suite of built-in capabilities that simulate human-like conversations. These features allow researchers to move beyond the typical chatbot experience (where responses appear instantly and in full) toward more naturalistic interactions that mirror text messaging or casual online chat. By introducing realistic timing, message chunking, and varied personas, researchers can create conversational contexts that feel more authentic to participants, potentially eliciting more natural behavior.

Naturalistic features shape the quality of the interaction whether or not the participant knows the partner is an AI. Timing, chunking, and personas make an exchange feel less mechanical, and more human-like agents tend to elicit greater trust and engagement from users (Cohn et al., 2024; Waytz et al., 2014). Participants told they are interacting with a computer can even show lower fear of self-disclosure and disclose more openly (Lucas et al., 2014). These features matter most when the bot is meant to stand in for a person. In a deception study, the participant believes their partner is human, so the bot must behave like one (e.g., it cannot answer instantly, in full paragraphs). The same holds when the bot serves as a configurable confederate the participant knows is an AI, since a partner that does not behave naturally is a poor stand-in for a human one.

All of these features are optional and can be disabled entirely, allowing researchers who prefer a standard chatbot experience without human-like simulation to create a straightforward interaction.

**Behavioral Parameters.** CHATBOTLAB allows researchers to configure several optional delay parameters that simulate how humans read and write text. The **Reading Words Per Minute** parameter controls how long the bot “reads” a participant’s message before beginning to type its response, with typical values around 240 words per minute (Brysbaert, 2019). The **Writing Words Per Minute** parameter simulates typing speed, with realistic values around 52 words per minute (Dhakal et al., 2018). Additionally, **thinking delays** can be inserted between messages to simulate pauses for reflection or deliberation, useful when the bot should appear to be considering a complex question or formulating a thoughtful response.

In addition to these timing delays, CHATBOTLAB displays a standard “typing indicator” (the animated dots (●●●) commonly seen in messaging apps) while the bot is composing its response. This visual feedback maintains conversational flow by signaling that the bot has received the participant’s message and is actively formulating a response, enhancing the perception of human-likeness. The indicator appears immediately after the participant sends a message and remains visible throughout the simulated reading and writing time, disappearing only when the bot’s message is delivered. Researchers should note that the typing indicator duration can

itself become a communicative signal: a brief typing period may suggest a quick response, while an extended period might convey careful deliberation.

These timing parameters serve multiple research purposes. First, they prevent the bot from responding instantaneously, which can feel uncanny and remind participants they are interacting with AI rather than a human. Second, they create opportunities for participants to reflect on their own messages while the bot is “typing,” potentially affecting what participants say next. Third, they allow researchers to manipulate perceived cognitive effort: a bot that responds quickly may seem superficial, while one that pauses longer may seem more thoughtful or careful. When configuring these parameters, researchers should consider their experimental context—a casual peer conversation might use faster timing, while a counseling bot might incorporate longer pauses to convey attentiveness.

**Message Flow.** Chatbots typically respond with a single chunk of text that appears all at once. CHATBOTLAB allows researchers to simulate a more casual chat or text message style conversation, where the chatbot’s response is broken up into shorter segments and each segment is transmitted independently, mimicking the way humans naturally compose messages in real-time conversations. For example, instead of displaying:

“That’s really interesting! I can see why you’d feel that way. Have you considered talking to your professor about it?”

The same response might be chunked as:

“That’s really interesting!”

“I can see why you’d feel that way.”

“Have you considered talking to your professor about it?”

This message chunking creates a more dynamic conversational flow, particularly important for research on SMS-based communication.

**Personas.** While a bot’s persona can be fully specified via the system prompt, CHATBOTLAB allows chatbots to randomly select from a set of predefined personas at the start of each conversation. This feature is useful when the overall style, behavior, or task is consistent across interactions, but you would like to vary specific features. For example, researchers could generate several personas that vary sociodemographics (*you are a moderately conservative, 38 year old male from Dallas, TX* vs. *you are a progressive, 34 year old woman from New York*) while keeping the conversational tone consistent (*be concise and supportive*). This allows for within-study variation in bot characteristics without requiring researchers to manually create and manage multiple separate bot configurations.

### Tutorial Step 3: Configure Conversation Dynamics (Optional)

In this step, you'll configure human-like interaction features to create more naturalistic conversations. These include typing delays (simulating how long the bot takes to read and respond), message chunking (breaking responses into shorter segments), and personas (varying the bot's identity across conversations).

1. The **Initial Utterance** is the first utterance from the chatbot and the first utterance in the conversation (it will be sent when the participant joins the chat). Set it to: *Hey! What's on your mind today? Happy to talk about anything, whatever you feel like getting into.*
2. Check the box **Chunk messages**, which will break up long chatbot responses into shorter messages (each sent separately). This mimics an informal, SMS-like communication style.
3. Check the box **Humanlike delay**. This allows you to set reading, typing, and thinking delays (unchecking the box will turn this off completely)
4. In the **Humanlike Delay Configuration** section, set the following:
  - (a) Reading words per minute: 240
  - (b) Writing words per minute: 52
5. In the **Follow-up Settings** section, set the following:
  - (a) Check the **Follow up on idle** box. This tells the chatbot to followup with the participant if they are idle.
  - (b) Follow up instruction prompt: *If the conversation stops please keep the conversation going by asking if your partner is still there, if they want to continue talking, or remind them that we only have 10 minutes to complete the task.*
  - (c) Check the **Recurring followup** box. This tells the chatbot to continuously followup with the participant, once per the time period set in **Idle time minutes**.

## Content Moderation

To address safety concerns in conversational AI studies, CHATBOTLAB includes built-in content moderation powered by the OpenAI Moderation API, a free service that screens both participant messages and bot responses in real-time. The system evaluates content across a set of categories, including harassment, hate, illicit behavior, self-harm, sexual content, and violence, several of which have finer-grained subcategories. For each category, the API returns a model-generated category score (0.0-1.0), and researchers can set custom thresholds for each category, where lower values result in stricter moderation (e.g., 0.3 flags more content than 0.7). These scores are not calibrated probabilities and should not be read as the likelihood that content belongs to a category; they are relative signals whose meaning depends on the model version producing them. When content exceeds a threshold, flagged participant messages are blocked from reaching the LLM, and flagged bot responses are prevented from being displayed. Moderation can be disabled by setting all values to 1.

When a message is blocked, CHATBOTLAB records the event. Two fields are written: the participant's message stores the full map of moderation categories and their scores (including the categories that passed), and the category that triggered the block is recorded on both the blocked message and the warning shown in its place. Messages that pass mod-

eration leave these fields empty. The stored scores let researchers check how often moderation occurred, which categories triggered it, and whether the thresholds behaved as intended. The triggering category and scores are recorded for the researcher only and are never shown to the participant.

Several caveats apply when using automated moderation in research. Thresholds are not universal: a value that works for one task or population may over- or under-block in another, and thresholds should be validated on content representative of the study before a live launch. Because the scores come from a model behind an API, a provider model update can shift the score distribution and change what a fixed threshold flags, so a value set months earlier may not behave the same way later. Moderation also performs unevenly across languages, dialects, and cultural contexts, and may systematically misjudge language it represents poorly (Sap et al., 2019). Finally, blocking a message is not the same as protecting a participant; it suppresses text but offers no support, so moderation should be paired with a response protocol (see the Participant Welfare section) rather than treated as a safety plan on its own.

## Study Integration

CHATBOTLAB appears as a window inside your existing survey flow. Participants stay inside the survey (there is no outside website they interact with), resulting in a streamlined

#### Tutorial Step 4: Moderate Conversations (Optional)

Content moderation is worth enabling here because participants can raise anything and the conversation runs in both directions: it screens both what the participant sends to the bot and what the bot sends back. Note that harassment, for instance, can come from either side, as a participant may be hostile toward the bot just as the bot may generate something inappropriate. CHATBOTLAB uses the OpenAI Moderation API to screen both participant messages and bot responses across a set of categories (harassment, hate, illicit behavior, self-harm, sexual content, and violence) and their subcategories. For each moderation category, you can select a threshold from 0.0 to 1.0, where lower values apply stricter moderation.

1. Log into the CHATBOTLAB admin panel
2. Navigate to **Bots** and select the **support-bot** chatbot
3. Navigate to the Moderation Settings section
4. Set the following value: harassment = 0.3<sup>a</sup>
5. Click Save

<sup>a</sup>This value is illustrative, not a validated threshold or a recommendation. The appropriate setting depends on the study, the population, and the model version producing the scores, and should be validated on representative content before a live study (see the Content Moderation section).

design.

Survey data can be passed to CHATBOTLAB's backend, which allows you to merge survey responses with conversational text data for downstream analysis. For example, Qualtrics generates a unique identifier (ResponseID) for each survey response. By passing this value to CHATBOTLAB, each conversation is stored with the same identifier as its corresponding survey response, so the two can be joined without any manual matching. This is not limited to a single variable: any survey field can be passed in the same way and saved alongside the conversation. In our running example, we pass the participant's PHQ-9 score, so that the language they use in the conversation can later be analyzed in relation to their self-reported depression.

### Survey Platforms

CHATBOTLAB integrates into surveys through JavaScript embedded in a survey question. Qualtrics, REDCap, and LimeSurvey support custom JavaScript within questions, allowing the chat interface to be loaded directly into the survey page. The implementation process is straightforward: CHATBOTLAB provides a JavaScript code template that researchers

copy into their survey, replacing a few placeholder values based on their study requirements.

The JavaScript code accomplishes several tasks. First, it retrieves relevant survey variables (bot name, participant identifiers, condition assignments) from the survey's embedded data fields. Second, it constructs a URL pointing to your CHATBOTLAB deployment with these parameters. Third, it creates an iframe (i.e., a window within the survey page) that loads the chat interface at that URL.

Code Box 1 is an example of the core JavaScript template for Qualtrics. In this example, Qualtrics generates a unique variable ResponseID for each survey response. This is then stored in the conversationID field in CHATBOTLAB, allowing direct mapping between surveys and conversations. The tutorial uses a single bot for all participants (e.g., botName = "support-bot"), but the code could dynamically retrieve the bot name from an embedded data field. This allows for flexible experimental designs where different participants interact with different bots based on random assignment or survey logic. For instance, you might randomly assign participants to a "supportive" vs. "neutral" condition in Qualtrics, store the appropriate bot name in an embedded data field for each condition, and the JavaScript would automatically load the correct bot.

#### Tutorial Step 5: Integrate CHATBOTLAB with Qualtrics

Now that the chatbot is built, you can integrate it with Qualtrics. See documentation<sup>1</sup> for how to do the same with REDCap and LimeSurvey.

1. Create a new question block and name it chat
2. Inside this block, add a new question and set the **Question type** to Text / Graphic.
3. Name this question chat\_window
4. Under the **Question behavior** menu, select `</> JavaScripta`
5. Copy the code from Code Box 1 into the JavaScript editor
6. Click Save
7. In the Qualtrics **Survey flow** move the chat question block to the point in the survey where you want the conversation to occur (e.g., after the baseline questions).

<sup>a</sup>Not all Qualtrics accounts have this feature enabled by default. Your account Administrator can enable this.

REDCap and LimeSurvey both follow a nearly identical pattern, with minor syntax differences in how embedded data fields are referenced. CHATBOTLAB's documentation<sup>1</sup> provides templates for all platforms, and researchers typically need to modify only 3-4 lines: their CHATBOTLAB deployment URL, the bot name, and which survey variables to pass to the database.

## Code Box 1: Embedding CHATBOTLAB in Qualtrics

```

Qualtrics.SurveyEngine.addOnload(function() {
  // Retrieve survey variables from embedded data fields
  var botName = "support-bot";
  var surveyID = "${e://Field/SurveyID}";
  var conversationID = "${e://Field/ResponseID}";

  // Construct ChatbotLab URL with parameters
  var botURL = "https://<your-id>.cloudfront.net/conversation";
  botURL += "?bot_name=" + encodeURIComponent(botName);
  botURL += "&conversation_id=" + encodeURIComponent(conversationID);
  botURL += "&survey_id=" + encodeURIComponent(surveyID);

  // Create and embed the chat interface
  var container = this.getQuestionTextContainer();
  if (container) {
    var iframe = jQuery("<iframe>", {
      src: botURL,
      width: "100%",
      height: "100vh",
      frameborder: "0"
    });
    jQuery(container).append(iframe);
  }
});

```

## Recruitment

Participants interact with CHATBOTLAB through a webpage. A conversation is launched by sending a participant to a URL with a few parameters attached (e.g., chatbot name, a conversation ID, a participant ID). Recruitment is therefore a question of how a participant reaches the URL and what information is passed. This can be fully automated, partly automated, or fully manual.

Platforms like Prolific and MTurk act as recruitment and compensation layers on top of a survey. You provide the URL to your Qualtrics, REDCap, or LimeSurvey survey, and the platform appends identifiers to it, such as Prolific's PROLIFIC\_PID or MTurk's workerId and assignmentId. The survey captures these identifiers and passes them to CHATBOTLAB, where they are stored with the conversation. This linkage lets you match a conversation to the worker who produced it, which matters for payment, exclusion, and data quality.

The complete workflow looks like this: (1) the participant accepts your task on Prolific/MTurk, (2) the platform redirects them to your survey URL with worker ID appended, (3) the survey captures worker ID in embedded data, (4) the participant reaches the embedded CHATBOTLAB chat, (5) JavaScript passes worker ID to CHATBOTLAB's database (Code

Box 2), (6) the conversation is stored with worker ID attached, (7) the participant completes survey and returns to the crowdsourcing platform for compensation. This integration requires no custom development, as it leverages built-in features of crowdsourcing platforms, survey tools, and CHATBOTLAB's data passing capabilities.

The same survey can also be administered without a crowdsourcing platform. A researcher can send the survey link to participants directly (e.g., by email), or administer it in person, such as clinical settings, where a research assistant hands a participant a tablet. The survey runs the same way, with only the recruitment mechanism changing.

Finally, a survey platform is not required at all. In the above recruitment examples, CHATBOTLAB is loaded as an iframe (i.e., a embedded web page) inside a survey, but the page can also be opened on its own. A researcher can pass the parameters by hand, or email a participant a personalized link with the parameters already embedded. This is useful for small studies, pilots, or settings where a full survey platform is unnecessary.

### Tutorial Step 6: Integrate CHATBOTLAB with Prolific

Here we will integrate CHATBOTLAB with the crowdsourcing platform Prolific for participant recruitment and payment. See documentation<sup>1</sup> for how to do the same with MTurk. This will save important participant information on the CHATBOTLAB server next to the conversation data.

1. In the Qualtrics survey, under **Survey flow**, create an embedded data block and put this at the top of your flow
2. Add the following fields, without setting an explicit value (*Value will be set from Panel or URL*): PROLIFIC\_PID, STUDY\_ID, and SESSION\_ID
3. Click Save in your **Survey flow**
4. In the **Builder** panel, open the chat question block, and return to the JavaScript editor for the chat\_window question
5. Replace the current code with the code from Code Box 2. The updated code includes:
  - (a) Three new variables (prolificPID, studyID, and sessionID) after the line which initializes conversationID
  - (b) Three additional lines starting with botURL
6. Click Save

## Data Collection and Management

### Data Quality

In addition to embedding the conversation, the CHATBOTLAB JavaScript template records two behavioral measures that help researchers assess data quality. The first is a keystroke count: the number of keys a participant presses while composing their messages (the count itself, not the specific keys). Because typing a message produces roughly one keystroke per character, a response that is far longer than its keystroke count suggests the text was pasted rather than typed, which can flag participants who generate replies with an external tool such as their own LLM. The second measure is tab visibility, which records when a participant switches away from the survey page; frequent or lengthy absences can indicate divided attention or disengagement. Both measures are sent to CHATBOTLAB's database when the participant completes or leaves the conversation, where they are stored alongside the conversation and can be used to screen out low-quality responses or to study engagement during analysis. To keep the tutorial code short, the template shown earlier omits this tracking; the extended template that records both mea-

asures is available in the documentation<sup>1</sup>.

### Database Structure

CHATBOTLAB stores study data across two data tables, whose rows are created as participants interact with the bot. The **conversations** table stores metadata for each conversation (participant ID, survey ID, timestamps, any survey variables passed in via the integration, and the complete bot configuration used for that conversation). The **utterances** table contains the actual messages, where each row is a single turn in the conversation, capturing message content, speaker (participant or bot), and timing information. This structure lets researchers analyze data at the conversation level (e.g., duration, engagement) or the utterance level (e.g., individual turns). A separate set of configuration tables, populated before a study is run rather than during it, holds the study setup: the **bots** table (name, LLM model, system prompt, behavioral parameters), along with tables for personas, moderation settings, and data-quality signals such as keystroke counts. Rows in the data tables reference this configuration, so every conversation records the exact setup it was run under.

A key feature of this design is that bot configurations are saved at the conversation level. This means if you modify your bot's prompt or parameters during data collection, CHATBOTLAB automatically tracks which conversations used which configuration. This is critical for data integrity: you can identify conversations conducted before and after your changes, ensuring that data from different configurations can be analyzed separately or controlled for in statistical analyses.

### Exporting Data

The CHATBOTLAB admin panel lets researchers export their data at any point during or after data collection. Data can be exported as CSV for spreadsheet applications and statistical software, as JSON for custom processing, or directly as a ConvoKit corpus. ConvoKit is a widely used Python package for conversational research that provides ready-made tools for analyzing dialog structure, speaker characteristics, and conversation dynamics. Because CHATBOTLAB exports a native ConvoKit corpus, the data loads directly into these tools without the reshaping that conversational data usually requires.

Whichever format you choose, the survey variables passed in during integration are exported alongside each conversation. In our running example, this means a participant's PHQ-9 score is already attached to their conversation, so language from the conversation can be analyzed against self-reported depression without manually merging separate files. The same exported data also feeds other established analysis pipelines, including DLATK for linguistic feature extraction and the R `text` package for language-based assessment.

## Code Box 2: Crowdsourcing Integration with Prolific

```

Qualtrics.SurveyEngine.addOnload(function() {
  // Retrieve survey variables from embedded data fields
  var botName = "support-bot";
  var surveyID = "${e://Field/SurveyID}";
  var conversationID = "${e://Field/ResponseID}";
  var prolificPID = "${e://Field/PROLIFIC_PID}";
  var studyID = "${e://Field/STUDY_ID}";
  var sessionId = "${e://Field/SESSION_ID}";

  // Construct ChatbotLab URL with parameters
  var botURL = "https://<your-id>.cloudfront.net/conversation";
  botURL += "?bot_name=" + encodeURIComponent(botName);
  botURL += "&conversation_id=" + encodeURIComponent(conversationID);
  botURL += "&survey_id=" + encodeURIComponent(surveyID);
  botURL += "&prolific_pid=" + encodeURIComponent(prolificPID);
  botURL += "&study_id=" + encodeURIComponent(studyID);
  botURL += "&session_id=" + encodeURIComponent(sessionID);

  // Create and embed the chat interface
  var container = this.getQuestionTextContainer();
  if (container) {
    var iframe = jQuery("<iframe>", {
      src: botURL,
      width: "100%",
      height: "100vh",
      frameborder: "0"
    });
    jQuery(container).append(iframe);
  }
});

```

## Tutorial Step 7: Download Your Data

In this step, you'll download your conversation data from the CHATBOTLAB admin panel, so that we can run text analysis.

1. Log into the CHATBOTLAB admin panel
2. Navigate to **Bots** and select the chatbot(s) you would like to export. This will export all conversations with the selected chatbot
3. Click the **Export Data** button
4. Select the **text** export format
5. Click **Download**
6. Store your data in a secure location with appropriate access controls

## Text Processing

Once data are exported, the conversation text can be analyzed with established tools. CHATBOTLAB targets three that are widely used in language-based research, and exports data in formats each can read without reshaping. Because the survey variables passed in during integration are stored with each conversation, any of these analyses can relate the language of a conversation to a participant's self-reported measures, such as the PHQ-9 score in our running example. See Tutorial Step 8 and Code Box 3 steps on how to analyze the language data using **text**.

## ConvoKit (Python)

CHATBOTLAB exports directly to ConvoKit (Chang et al., 2020), a Python package for representing and analyzing conversations. Because the export is a native ConvoKit corpus, the speakers, conversations, and individual utterances are already organized in the structure ConvoKit expects, and the

**Table 1**

Comparison of open-source systems for LLM-based research. A checkmark (✓) indicates the feature is described in the system’s documentation or associated publication.

| Feature                                                   | CHATBOTLAB | Simple Chat | LUCID | DiSCoKit | CLPC | QUAIL |
|-----------------------------------------------------------|------------|-------------|-------|----------|------|-------|
| <i>Platform Integration</i>                               |            |             |       |          |      |       |
| Survey platforms (Qualtrics, REDCap, LimeSurvey)          | ✓          | ✓           | ✓     | ✓        | —    | ✓     |
| Crowdsourcing (Prolific, MTurk)                           | ✓          | —           | —     | —        | —    | —     |
| Embedded in survey (no separate login)                    | ✓          | ✓           | ✓     | ✓        | —    | ✓     |
| Automatic participant ID linkage                          | ✓          | —           | ✓     | ✓        | ✓    | —     |
| <i>Model Support</i>                                      |            |             |       |          |      |       |
| Multiple LLM providers                                    | ✓          | ✓           | —     | —        | ✓    | —     |
| Switch models mid-conversation                            | —          | —           | —     | —        | ✓    | —     |
| Local / open-weights models                               | ✓          | ✓           | —     | —        | ✓    | —     |
| <i>Conversation Features</i>                              |            |             |       |          |      |       |
| Typing delays / response timing                           | ✓          | —           | —     | —        | —    | —     |
| Message chunking                                          | ✓          | —           | —     | —        | —    | —     |
| Personas                                                  | ✓          | —           | ✓     | ✓        | —    | —     |
| Streaming responses                                       | —          | ✓           | —     | —        | —    | —     |
| <i>Data &amp; Safety</i>                                  |            |             |       |          |      |       |
| Content moderation                                        | ✓          | —           | —     | —        | —    | —     |
| Data quality monitoring (keystrokes, attention)           | ✓          | —           | —     | —        | ✓    | —     |
| Per-conversation configuration tracking                   | ✓          | —           | ✓     | ✓        | —    | —     |
| Standardized export (ConvoKit, DLATK, <code>text</code> ) | ✓          | —           | —     | —        | —    | —     |
| <i>Deployment</i>                                         |            |             |       |          |      |       |
| Automated deployment                                      | ✓          | —           | —     | —        | —    | —     |
| Online scale / load tested                                | ✓          | ✓           | ✓     | —        | —    | —     |
| Low-code configuration                                    | ✓          | ✓           | ✓     | ✓        | ✓    | ✓     |

survey variables passed from the integration appear as conversation metadata. ConvoKit is suited to questions about the conversation as an object: the number and length of turns, who speaks when, how the exchange develops, and how these properties differ across conditions or relate to participant characteristics. Its built-in tools cover common conversational measures, such as conversational derailment (Chang & Danescu-Niculescu-Mizil, 2019), power differentials between speakers (Danescu-Niculescu-Mizil et al., 2012), and how a speaker’s voice develops over time (Zhang et al., 2019).

### DLATK (Python)

DLATK (Differential Language Analysis ToolKit) (Schwartz et al., 2017) extracts linguistic features from text and relates them to outcomes. It can compute features such as word and phrase frequencies, dictionary-based categories, topics, and contextual embeddings, and then correlate them with a grouping variable (i.e., survey responses). CHATBOTLAB’s database is organized so that DLATK can read it directly, and its analyses can be run at different levels by changing the unit of grouping: features can be computed and correlated at the level of the individual

utterance, the whole conversation, or the participant (over multiple conversations). In our running example, this allows the language of a conversation to be related to a participant’s PHQ-9 score at the conversation level, or examined turn by turn, without preparing the data differently for each.

### `text` (R)

The `text` package (Kjell et al., 2023) brings language-based assessment to R, using language models to represent text as embeddings and to estimate psychological constructs from it. It is suited to questions about what a participant’s language reveals: assessing sentiment, well-being, or other constructs from what they wrote, or building models that predict an outcome from the text. CHATBOTLAB’s exported data load directly into the package, so conversation text can be scored and related to the survey measures stored alongside it. Additionally, the Language-Based Assessment Model (LBAM) Library (Nilsson et al., 2026) is available directly through `text`, which allows users to quickly apply pretrained language-based assessments to the chat language data, including mental health, emotions, and well-being.

## CHATBOTLAB versus other frameworks

CHATBOTLAB is not the first system to embed conversational agents in research workflows. Several tools share its central premise of placing a configurable LLM inside a survey. LUCID (Garvey & Blanchard, 2025) and DiS-CoKit (Banks et al., 2026) embed an agent in Qualtrics through JavaScript. Simple Chat (Schettino et al., 2025) provides a chat interface that connects to multiple model providers and several survey platforms. QUAIL (Behrend & Landers, 2025) integrates content from a single provider into Qualtrics, with an emphasis on generating research materials rather than running live conversations. CLPC (Lamprou & Moshfeghi, 2025) is a standalone chat application in which participants log in with an identifier rather than entering through a survey.

Table 1 compares these systems. CHATBOTLAB overlaps with each in its core function but differs in scope: most are designed around a single survey platform and a single stage of a study, while CHATBOTLAB covers the full workflow from recruitment through analysis. Four features distinguish it. It integrates with crowdsourcing platforms (Prolific, MTurk) and links worker identifiers to conversation data; it can be provisioned, using load testing, to handle the surge of traffic when a study opens to a recruitment pool; it includes naturalistic interaction features such as timing, message chunking, and personas; and it exports data in formats that text-analysis tools read directly, linking conversation text to survey responses without manual merging.

Other platforms address adjacent problems. Deliberate Lab (Qian et al., 2025) supports experiments in which groups of humans and AI agents interact. Empirica (Almaatouq et al., 2021) provides infrastructure for high-throughput behavioral experiments, predating the current generation of LLM tools. WebXAI (Leguy et al., 2025) studies how people interpret explainable AI. These target different questions than CHATBOTLAB, but reflect the broader move toward standardized infrastructure for human-AI interaction research.

## Technical Details

This section describes the overall architecture, and how CHATBOTLAB handles two parts of running a study at scale: security and server provisioning. Researchers do not need to configure either one, as are set up automatically at deployment. We include details for readers who want to understand what runs on their behalf, or who are planning a large study and need to size their server.

**Architecture.** CHATBOTLAB achieves this simplicity through a layered architecture. At the outermost layer, standard research platforms handle participant recruitment (crowdsourcing services) and study presentation (survey platforms). The chat interface appears as an embedded iframe within the survey, served by CHATBOTLAB’s Django

**Table 2**

*CHATBOTLAB server configurations by maximum number of simultaneous conversations, based on load testing the cloud-based server (and not the LLM API, e.g., OpenAI). We assume each user is sending a message every 5 seconds, all users are simultaneously sending messages, and a < 1% error rate (dropped connections). Requirements vary with conversation length, message frequency, and LLM provider response time. \* interpolated based on linear relationship between number of instances and maximum simultaneous conversations.*

| Maximum conversations | Instance type | Number of instances | Database type |
|-----------------------|---------------|---------------------|---------------|
| 100                   | t3.small      | 1                   | db.t3.small   |
| 1000                  | t3.medium     | 4                   | db.t3.medium  |
| 2000                  | t3.medium     | 8                   | db.m5.2xlarge |
| 3000                  | t3.medium     | 12                  | db.m5.2xlarge |
| 4000*                 | t3.medium     | 16                  | db.m5.2xlarge |
| 5000                  | t3.medium     | 20                  | db.m5.2xlarge |

web application with a React frontend. This same application provides the admin panel where researchers configure their studies. When a participant sends a message, CHATBOTLAB routes it through the Kani framework (Zhu et al., 2023), which provides a unified interface to multiple LLM providers (OpenAI, Anthropic, Google, and Meta’s open-weight models). Kani handles the provider-specific API calls and returns the LLM’s response, which CHATBOTLAB then processes for naturalistic features (typing delays, message chunking) before displaying it to the participant.

Throughout this process, CHATBOTLAB stores all data in a MariaDB database with a schema designed for conversation analysis: each chatbot points to multiple conversations, each conversation contains multiple turns, and each turn captures the message content along with metadata about the speaker and timing. The bot configuration in force (LLM model, prompt, and parameters) is recorded once per conversation rather than per turn. This structured storage enables researchers to export data in formats compatible with established text analysis packages without additional data wrangling.

### Tutorial Step 8: Language Analysis in R

In this step, you will use the conversation data you exported in Step 7 to run a language analysis in R. We embed each participant's messages with the transformer language model BERT (Devlin et al., 2019), then test which words associate with a survey outcome, in this example, PHQ-9 depression score. The full code is in Code Box 3.

1. Install the R `text` package and, one time only, install and initialize its Python environment with `textrpp_install()` and `textrpp_initialize()`
2. Load `msgs.csv` and `outcomes.csv`, the files you downloaded in Step 7
3. Combine each participant's messages into a single row of text, and join this to their outcome data
4. Embed each participant's text with a BERT language model using `textEmbed()`
5. Run `textProjection()` to test which words associate with your outcome variable
6. Run `textProjectionPlot()` to plot the results (Figure 3)

See Code Box 3 for the full R code.

### Security

CHATBOTLAB runs on the researcher's own cloud account (i.e., AWS). The data belongs to the researcher and is never routed through or stored by a third-party service, except for the LLM provider that generates the bot's replies. The database is private and cannot be reached directly by participants or the public, and it is encrypted and backed up. Access keys are stored securely rather than in the code. Together, these defaults mean that a researcher can run a study without handling sensitive data by hand, and that conversation data stays under their control.

### Load Testing

A chatbot study places a different demand on a server than a standard survey. Each participant holds an open connection for the length of the conversation, and every message triggers a call to an LLM provider, so the server must maintain many concurrent, long-lived sessions rather than serving brief independent page requests. When a study opens to a recruitment pool, participants arrive in a short window, and a server that is too small will slow or drop messages under the load.

CHATBOTLAB does not scale automatically. Instead, the server is provisioned for the expected number of simultaneous conversations before a study launches, and the appropriate size is determined by load testing. Our tests simulate

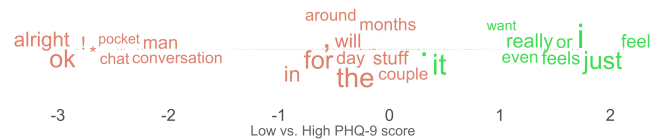


**Figure 3**

*Supervised Dimension Projection (SDP) of words from the synthetic conversations, projected onto PHQ-9 depression score. Each point is a word type. Position on the x-axis is the word's SDP score, the strength and direction of its association with PHQ-9. Point size scales with how often participants used the word. Labeled words are the most extreme and the most frequent on each side. Words toward the right, such as *feel* and *just*, associate with higher PHQ-9. Words toward the left, such as *alright* and *ok*, associate with lower PHQ-9. Produced with `textProjectionPlot()` on the synthetic corpus from Code Box 3 using the R `text` package*

many participants conversing at once and we measure response times and the rate at which requests fail (i.e., chat messages are not received by the backend). For a given server configuration, we increased the number of simulated concurrent conversations until the failure rate exceeded an acceptable threshold (1%), and then adjusted the configuration (the size of the server instance, the number of instances, or both) until performance was acceptable at the target load. Table 2 reports configurations that handled a given number of simultaneous conversations in our testing, ranging from 100 to 5000.

### Ethical Considerations

Conducting research with conversational agents raises ethical questions that researchers must address before collecting data. Many of these are not unique to CHATBOTLAB, but become more relevant given the lowered technical barriers. Below we discuss four considerations that arise when running chatbot studies, several of which are illustrated by the example study in the tutorial. These considerations are not always followed in practice. A recent review of AI research involving human participants found that most published studies did not report independent ethical review, informed consent, or participant compensation (McKee, 2024).

### Deception and Disclosure

The chatbot in the tutorial is instructed to present itself as a human peer and not to reveal that it is a bot. Deception of this kind has a long history in psychological research, as a participant who knows they are speaking to a bot may behave differently than one who believes they are speaking to a human, which could undermine the study (Hilbig et al., 2021). Consenting participants should be debriefed after the

study, with disclosure of the use of a bot and explanation of why it was necessary (American Psychological Association, 2017). In our tutorial study, participants were told at the end of the conversation that their partner was an AI agent. Researchers who present a bot as human should treat disclosure at debriefing as a requirement rather than an option. As noted before the tutorial's build steps, the chatbot in our example presents itself as a human peer and does not reveal that it is a bot.

### Participant Welfare

Studies that ask participants to discuss difficult or emotionally charged topics must consider participant welfare. CHATBOTLAB includes content moderation, but moderation is not a safety protocol, as it simply blocks a (potentially) problematic message, and does nothing to help a participant. In our illustrative example, the chatbot is told not to give help-seeking information, as an LLM can hallucinate inaccurate resources, such as a wrong hotline number. A real study using this design would instead give participants vetted resources at debriefing. Researchers studying sensitive topics should decide in advance how these situations will be handled, what resources will be offered, and at what point. This responsibility rests with the research team and not with CHATBOTLAB. See Meadi et al. (2025) for a review of ethical challenges of conversational AI in mental-health contexts.

### Data Privacy

Conversational data is potentially identifying, as open-ended exchanges can contain names, locations, and health information. CHATBOTLAB links this text to participant and crowdworker identifiers, which increases its sensitivity. To help mitigate these risks, CHATBOTLAB is deployed on the researcher's own infrastructure, so conversation data is not routed through or stored by a third-party service, except the LLM provider (e.g., Anthropic, OpenAI). Researchers should nonetheless treat transcripts as sensitive data, store them securely, and limit access appropriately.

### Disclosure of AI and Platform Policies

Whether people have a right to know they are interacting with an AI is an active research and regulatory question (California State Legislature, 2018; European Parliament and Council, 2024; Schilke & Reimann, 2025). Crowdsourcing and survey platforms also maintain their own policies on deception and automated agents (Prolific, 2026), which researchers should consult before fielding a study. Because CHATBOTLAB can present a bot as a human partner, researchers using this capability should ensure that their design is compatible with the policies of the platforms they recruit through, and with any applicable regulation, in addition to obtaining approval from their own institution.

### Summary

In this tutorial, we presented CHATBOTLAB, an open-source software package that enables researchers to conduct chatbot-based studies by embedding LLM conversations directly into standard survey and crowdsourcing platforms. CHATBOTLAB handles the technical infrastructure that custom-built systems often overlook, including linkage between survey and conversation data, per-participant configuration records, data quality measures, content moderation, and export in formats compatible with established text-analysis tools. We guided researchers through the complete workflow, from deploying a chatbot server and configuring conversational features to integrating conversations into a study and exporting data for analysis. After reading this tutorial, researchers should possess the skills to design, deploy, and run their own chatbot study without application programming, using CHATBOTLAB's automated, low-code deployment and configuration workflow. Researchers still supply and configure their own cloud, survey, and model-provider accounts, and may draw on institutional IT support for permissions, security review, or large-scale deployments.

As an open framework, CHATBOTLAB can grow with the needs of the research community, incorporating new LLM providers, conversational features, and integrations as they emerge. As such, we invite extensions to CHATBOTLAB to support new research designs and to contribute improvements to the open-source project. By lowering the technical barriers to chatbot-based research, we hope CHATBOTLAB allows researchers to focus on their research questions rather than infrastructure, and helps make conversational AI methods accessible across the behavioral sciences.

### Author Contributions

Salvatore Giorgi: Conceptualization; Formal Analysis; Methodology; Resources; Software; Supervision; Visualization; Writing – original draft; Writing – review & editing. João Sedoc: Conceptualization; Methodology; Software; Supervision; Writing – review & editing. Abhay Singh: Formal Analysis; Software; Writing – review & editing. Oscar Kjell: Formal analysis; Visualization; Writing – original draft; Soojin Lee: Formal Analysis; Software; Writing – review & editing. Lyle H. Ungar: Conceptualization; Methodology; Funding acquisition; Resources; Supervision; Writing – original draft; Writing – review & editing.

### Declaration of Conflicting Interests

The author(s) declared no conflicts of interest with respect to the authorship or the publication of this article.

## References

- Almaatouq, A., Becker, J., Houghton, J. P., Paton, N., Watts, D. J., & Whiting, M. E. (2021). Empirica: A virtual lab for high-throughput macro-level experiments. *Behavior Research Methods*, 53(5), 2158–2171.
- American Psychological Association. (2017). Ethical principles of psychologists and code of conduct [Standards 8.07 (Deception) and 8.08 (Debriefing)].
- Banks, J., Stromer-Galley, J., Singh, S., & Capano, C. (2026). Discokit: An open-source toolkit for deploying live llm experiences in survey research. *arXiv preprint arXiv:2602.11230*.
- Behrend, T. S., & Landers, R. N. (2025). Participant interactions with artificial intelligence: Using large language models to generate research materials for surveys and experiments. *Journal of Business and Psychology*, 1–23.
- Brysbaert, M. (2019). How many words do we read per minute? a review and meta-analysis of reading rate. *Journal of memory and language*, 109, 104047.
- California State Legislature. (2018). Senate bill no. 1001: Bots—disclosure (bolstering online transparency act) [Operative July 1, 2019].
- Chang, J. P., Chiam, C., Fu, L., Wang, A., Zhang, J., & Danescu-Niculescu-Mizil, C. (2020). Convokit: A toolkit for the analysis of conversations. *Proceedings of the 21th annual meeting of the special interest group on discourse and dialogue*, 57–60.
- Chang, J. P., & Danescu-Niculescu-Mizil, C. (2019). Trouble on the horizon: Forecasting the derailment of online conversations as they develop. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 4743–4754.
- Cohn, M., Pushkarna, M., Olanubi, G. O., Moran, J. M., Padgett, D., Mengesha, Z., & Heldreth, C. (2024). Believing anthropomorphism: Examining the role of anthropomorphic cues on trust in large language models. *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, 1–15.
- Costello, T. H., Pennycook, G., & Rand, D. G. (2024). Durably reducing conspiracy beliefs through dialogues with ai. *Science*, 385(6714), eadq1814.
- Danescu-Niculescu-Mizil, C., Lee, L., Pang, B., & Kleinberg, J. (2012). Echoes of power: Language effects and power differences in social interaction. *Proceedings of the 21st international conference on World Wide Web*, 699–708.
- Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019, June). BERT: Pre-training of deep bidirectional transformers for language understanding. In J. Burstein, C. Doran, & T. Solorio (Eds.), *Proceedings of the 2019 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 1 (long and short papers)* (pp. 4171–4186). Association for Computational Linguistics. <https://doi.org/10.18653/v1/N19-1423>
- Dhakal, V., Feit, A. M., Kristensson, P. O., & Oulasvirta, A. (2018). Observations on typing from 136 million keystrokes. *Proceedings of the 2018 CHI conference on human factors in computing systems*, 1–12.
- European Parliament and Council. (2024). Regulation (eu) 2024/1689 laying down harmonised rules on artificial intelligence (artificial intelligence act) [Article 50, transparency obligations].
- Fitzpatrick, K. K., Darcy, A., & Vierhile, M. (2017). Delivering cognitive behavior therapy to young adults with symptoms of depression and anxiety using a fully automated conversational agent (woebot): A randomized controlled trial. *JMIR mental health*, 4(2), e7785.
- Flamino, J., Modi, M. S., Szymanski, B. K., Cross, B., & Mikolajczyk, C. (2025). Testing the limits of large language models in debating humans. *Scientific Reports*, 15(1), 13852.
- Folk, D., Yu, S., & Dunn, E. (2024). Can chatbots ever provide more social connection than humans? *Colab: Psychology*, 10(1), 117083.
- Garvey, A., & Blanchard, S. J. (2025). Generative ai as a research confederate: The lucid methodological framework and toolkit for human-ai interactions research. *Available at SSRN 5256150*.
- Guingrich, R. E., & Graziano, M. S. (2025). A longitudinal randomized control study of companion chatbot use: Anthropomorphism and its mediating role on social impacts. *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, 8(2), 1153–1153.
- Hilbig, B. E., Thielmann, I., & Böhm, R. (2021). Bending our ethics code. *European Psychologist*.
- Kjell, O., Giorgi, S., & Schwartz, H. A. (2023). The text-package: An r-package for analyzing and visualizing human language using natural language processing and transformers. *Psychological methods*, 28(6), 1478.
- Lamprou, Z., & Moshfeghi, Y. (2025). Customizable llm-powered chatbot for behavioral science research. *arXiv preprint arXiv:2501.05541*.
- Leguy, J., Jean, P.-A., Figueroa, F. T., & Harispe, S. (2025). Webxaii: An open-source web framework to study human-xai interaction. *arXiv preprint arXiv:2506.14777*.
- Li, H., Zhang, R., Lee, Y.-C., Kraut, R. E., & Mohr, D. C. (2023). Systematic review and meta-analysis of ai-

- based conversational agents for promoting mental health and well-being. *NPJ Digital Medicine*, 6(1), 236.
- Lin, H., Czarnek, G., Lewis, B., White, J. P., Berinsky, A. J., Costello, T., Pennycook, G., & Rand, D. G. (2025). Persuading voters using human-artificial intelligence dialogues. *Nature*, 1–8.
- Lucas, G. M., Gratch, J., King, A., & Morency, L.-P. (2014). It's only a computer: Virtual humans increase willingness to disclose. *Computers in Human Behavior*, 37, 94–100.
- McKee, K. R. (2024). Human participants in ai research: Ethics and transparency in practice. *IEEE Transactions on Technology and Society*, 5(3), 279–288.
- Mead, M. R., Sillescu, T., Metselaar, S., van Balkom, A., Bernstein, J., Batelaan, N., et al. (2025). Exploring the ethical challenges of conversational ai in mental health care: Scoping review. *JMIR mental health*, 12(1), e60432.
- Nilsson, A. H., Eijsbroek, V. C., Gu, Z., Kjell, K., Giorgi, S., Kotov, R., Ganesan, A. V., Schwartz, H. A., & Kjell, O. N. (2026). The language-based assessment model library: Open model sharing for independent validation and broader applications. *Advances in Methods and Practices in Psychological Science*, 9(2), 25152459261419036.
- Prolific. (2026). Can i run studies that deceive participants? [<https://researcher-help.prolific.com/en/articles/445176>].
- Qian, C., Tsai, V., Behr, M., Hussein, N., Laugier, L., Thain, N., & Dixon, L. (2025). Deliberate lab: A platform for real-time human-ai social experiments. *arXiv preprint arXiv:2510.13011*.
- Salvi, F., Horta Ribeiro, M., Gallotti, R., & West, R. (2025). On the conversational persuasiveness of gpt-4. *Nature Human Behaviour*, 9(8), 1645–1653.
- Sap, M., Card, D., Gabriel, S., Choi, Y., & Smith, N. A. (2019). The risk of racial bias in hate speech detection. *Proceedings of the 57th annual meeting of the association for computational linguistics*, 1668–1678.
- Schettino, R. B., Dasmeh, A., & Brinkmann, L. (2025). Facilitating the integration of llms into online experiments with simple chat. *arXiv preprint arXiv:2511.19123*.
- Schilke, O., & Reimann, M. (2025). The transparency dilemma: How ai disclosure erodes trust. *Organizational Behavior and Human Decision Processes*, 188, 104405.
- Schwartz, H. A., Giorgi, S., Sap, M., Crutchley, P., Ungar, L., & Eichstaedt, J. (2017). Dlatk: Differential language analysis toolkit. *Proceedings of the 2017 conference on empirical methods in natural language processing: System demonstrations*, 55–60.
- Stade, E. C., Stirman, S. W., Ungar, L. H., Boland, C. L., Schwartz, H. A., Yaden, D. B., Sedoc, J., DeRubeis, R. J., Willer, R., & Eichstaedt, J. C. (2024). Large language models could change the future of behavioral healthcare: A proposal for responsible development and evaluation. *NPJ Mental Health Research*, 3(1), 12.
- Waytz, A., Heafner, J., & Epley, N. (2014). The mind in the machine: Anthropomorphism increases trust in an autonomous vehicle. *Journal of experimental social psychology*, 52, 113–117.
- Zhang, J., Filbin, R., Morrison, C., Weiser, J., & Danescu-Niculescu-Mizil, C. (2019). Finding your voice: The linguistic development of mental health counselors. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 936–947.
- Zhu, A., Dugan, L., Hwang, A., & Callison-Burch, C. (2023). Kani: A lightweight and highly hackable framework for building language model applications. *Proceedings of the 3rd Workshop for Natural Language Processing Open Source Software (NLP-OSS 2023)*, 65–77.

## Code Box 3: Language analysis in R with text

```
install.packages("text")
library(text)

# One-time: creates a Python environment with torch and transformers
textrpp_install()
textrpp_initialize()

library(dplyr); library(readr)

# load the message and outcome tables
msgs <- read_csv("conversation_data/text/messages.csv")
outcomes <- read_csv("conversation_data/text/outcomes.csv")

# collapse each participant's messages into one row of text
person_data <- msgs %>%
  arrange(user_id, timestamp) %>%
  group_by(user_id) %>%
  summarise(text = paste(message, collapse = " ")) %>%
  inner_join(outcomes, by = "user_id")

# embed the text using BERT
word_embeddings <- textEmbed(
  person_data$text,
  model = "bert-base-uncased",
  aggregation_from_tokens_to_word_types = "mean",
  keep_token_embeddings = FALSE
)

# project words onto the PHQ-9 dimension
projection_results <- textProjection(
  words = person_data$text,
  word_embeddings = word_embeddings$texts$texts,
  word_types_embeddings = word_embeddings$word_types$texts,
  x = person_data$phq9,
  split = "mean",
  min_freq_words_test = 2
)

# build the projection plot
plot_projection <- textProjectionPlot(
  word_data = projection_results,
  min_freq_words_plot = 2,
  plot_n_word_extreme = 8, plot_n_word_frequency = 4, plot_n_words_middle = 2,
  y_axes = FALSE,
  p_alpha = 1,
  title_top = "Supervised Dimension Projection of PHQ-9",
  x_axes_label = "Low vs. High PHQ-9 score",
  p_adjust_method = "none"
)

# display the plot
plot_projection$final_plot
```