

BUILDING WEB APPLICATIONS TO STUDY CONSUMER INTERACTION WITH GENERATIVE-AI

INTRODUCTION

People are increasingly adopting generative AI (GenAI) in their daily lives. As of August 2024, nearly 40% of the U.S. population aged 18 to 64 had used GenAI to some extent (Bick, Blandin, and Deming 2024). Globally, the revenue generated by GenAI is expected to reach \$219 billion in 2025, with projections indicating a more than sixfold increase to \$1,361 billion by 2032 (Statista 2024). People primarily turn to GenAI for learning new skills and knowledge (Handa et al. 2025), generating ideas (Zao-Sanders 2024), and searching for information online (Kelly 2025). These widespread applications underscore GenAI's growing influence on decision-making and behavior.

Even though the interaction between users and generative AI typically takes place within closed systems operated by AI companies (e.g., OpenAI, Anthropic, Mistral), first research papers exist that have used GenAI tools as interventions in survey and experimental studies. For example, research has demonstrated that interactions with ChatGPT in online experiments can durably reduce beliefs in conspiracy theories (Costello, Pennycook, and Rand 2024) or increase creativity in idea generation (Lee and Chung 2024). Other research has tested whether letting people experience GenAI impacts the evaluation of GenAI created outputs (Celiktutan, Klesse, and Tuk 2024) or if political microtargeted LLM responses, based on provided demographic information, are more convincing than non-targeted messages (Hackenburg and Margetts 2024).

However, while these interventions represent promising applications of GenAI directly into behavioral research studies, implementing such approaches *correctly* presents significant technical barriers for researchers with limited technical expertise. Furthermore,

several methodological pitfalls may compromise study validity or pose security risks. For example, some studies required participants to use personal GenAI accounts (Lee and Chung 2024), what prevents researchers from directly observing the human-AI interaction while allowing only the AI company to retain this data. Additionally, this approach potentially creates inconsistent experiences across participants as personal system settings may introduce confounding variables, as GenAI tools are context-aware and may incorporate previous interactions into their responses (Huet, Houdi, and Rossi 2025; OpenAI 2023, 2025). Other integration approaches have integrated the interaction between the GenAI service directly into the user frontend, which may inadvertently expose authentication credentials (Costello et al. 2024), risking unauthorized API access that could result in service disruption through overuse or substantial financial costs due to excessive third-party exploitation.

Against this backdrop, this paper develops a practical guide and a versatile architectural framework to help researchers effectively incorporate generative AI as an intervention into their study and experimental studies. A key strength of this framework is its flexibility; it not only enables participants to directly interact with various GenAI services but also empowers researchers with the ability to connect to virtually any available GenAI model via its API. This adaptability in the choice and integration of services ensures the framework's continued relevance and utility as new generative AI models are developed and released. Furthermore, this setup allows researchers to systematically observe these interactions and investigate their relevant downstream consequences. Designed to be accessible to researchers with limited programming experience, the paper includes concrete examples and use-cases to inspire and support empirical applications.

The contributions of this work extend beyond methodological convenience, offering several benefits to market research. First, it democratizes GenAI research capabilities, enabling scholars from diverse technical backgrounds to explore this transformative

technology's impact on user behavior, decision-making, and experience. Second, by proposing a secure framework, it enhances methodological rigor and facilitates replication across studies, preventing researchers from security and methodological risks and addressing a critical need for consistent approaches in this rapidly evolving domain. Third, it creates opportunities to investigate previously unexplored research questions about consumer-AI dynamics, including how different consumer segments engage with AI assistance, how AI-mediated interactions shape preferences and beliefs, and how people integrate AI recommendations into their decision journeys. Ultimately, this approach enables broader research access to GenAI interventions, which may advance the field's capacity to develop theoretical frameworks for understanding user behavior in increasingly AI-mediated marketplaces.

In what follows, an overview of the use of GenAI interventions in different behavioral studies is provided and potential research questions are developed. Subsequently, the architectural framework of a web-based GenAI app is described, following a modular structure approach. The different GenAI models that are available, representing the core element of a web-based GenAI app and callable via API, are first outlined. Subsequently, the required components such as front-end and back-end are detailed. How these individual components can be created and combined without extensive coding knowledge is explained, and access is provided to different materials that facilitate development for researchers without prior knowledge in coding and GenAI technology. Finally, concrete showcases aim to illustrate the practical application of the described framework and components.

GENERATIVE AI AS INTERVENTIONS IN EXPERIMENTAL STUDIES

Reflecting the growing integration of GenAI into daily life, a nascent body of behavioral research utilizes direct interaction with these tools as experimental interventions.

These studies move beyond passive observation, actively engaging participants with GenAI to understand its influence on beliefs, capabilities, collaborative processes, people's behavior.

One significant line of inquiry examines how interacting with GenAI can shape attitudes and correct misperceptions. For example, research demonstrated that guided conversations with AI chatbots like ChatGPT can durably reduce belief in conspiracy theories (Costello et al. 2024). Similarly, exploring dietary choices, a brief dialogue with GPT-4 about meat consumption was shown to lessen participants' commitment to meat-eating and weaken related justifications such as the necessity to eat meat (Karakas and Jaeger 2025). Research, using an AI chatbot optimized for correcting misinformation about cultivated meat, found that the nature of the interaction—specifically, the AI conveying expertise and using two-sided refutations—was crucial for enhancing credibility and influencing attitudes about cultivated meat (Ou, Ho, and Wijaya 2025). The persuasive potential of these interactions is further emphasized by factors like personalization and perceived AI identity. While one study found personalized political messages based on demographics did not significantly outperform generic ones in brief interactions (Hackenburg and Margetts 2024), another involving direct human-AI debates showed personalization significantly boosted the LLM's persuasiveness (Salvi et al. 2024). Furthermore, whether users perceive they are interacting with a human or a bot significantly impacts outcomes like charitable donations and interpersonal judgments (Shi et al. 2020).

Beyond influencing beliefs, another research stream investigates how collaboration and interaction with GenAI affect human capabilities and evaluation. Studies on creativity show that using ChatGPT as an assistant during idea generation can lead to more creative outputs compared to working alone or using traditional web searches, suggesting the interaction helps users articulate thoughts more effectively (Lee and Chung 2024). However, this human-AI co-creation introduces complexities in evaluation; individuals tend to rate their

own contributions higher when working with ChatGPT, influenced by their assumptions about the AI's role (Celiktutan et al. 2024). Examining teamwork dynamics directly, a large-scale experiment compared interactions within human-human versus human-AI teams on a collaborative task. It revealed that human-AI teams communicated differently—focusing more on content and process, less on social aspects—leading to higher individual productivity and text quality, though human teams produced better images. Crucially, specific trait pairings between human collaborators and AI agents significantly influenced interaction dynamics and performance, and conscientious humans working with open AI bots improved image quality, while extroverted humans paired with conscientious AI bots reduced text quality (Ju and Aral 2025).

While current research highlights GenAI's exciting potential for interventions, many researchers lack the specific technical knowledge required for sound and secure implementation, which can inadvertently lead to flawed study designs. For instance, this has resulted in some studies risking the exposure of sensitive API keys or introducing confounding variables through suboptimal integration methods. To bridge this knowledge gap and address these critical concerns, the following section details a practical architectural framework designed to guide researchers in developing robust GenAI-driven web applications.

ARCHITECTURAL FRAMEWORK OF GEN-AI POWERED WEB APPLICATIONS

The architectural framework detailed in this section is designed to empower researchers to harness the exciting possibilities of GenAI for novel experimental paradigms—from using Large Language Models (LLMs) for creative ideation (Doshi and Hauser 2024), to employing text-to-image models for aesthetic studies (Cillo and Rubera 2024). Building such GenAI-powered applications effectively and securely fundamentally involves two core

elements: a Frontend interface through which the consumer interacts, and the GenAI Service (e.g., ChatGPT, Claude, DALL-E 3) that provides the generative capabilities. However, for crucial security reasons (such as protecting API keys) and reliable control, best practice dictates the inclusion of an intermediary Backend to manage communication, safeguard credentials, and centralize interaction logic.

This recommended structure – Frontend, Backend, and GenAI Service via Application Programming Interface (API)– forms the principal architecture for GenAI-powered web applications in research contexts. A key advantage of this API-centric approach is its inherent flexibility, empowering researchers to connect their applications to virtually any GenAI service that offers an API. This adaptability ensures that the described framework remains independent of the rapidly evolving model landscape and does not become obsolete as new and diverse generative AI models are published by various companies. The figure below depicts this architecture, which serves as both a conceptual framework for understanding these applications and the structural basis for the detailed discussion in the subsequent sections.

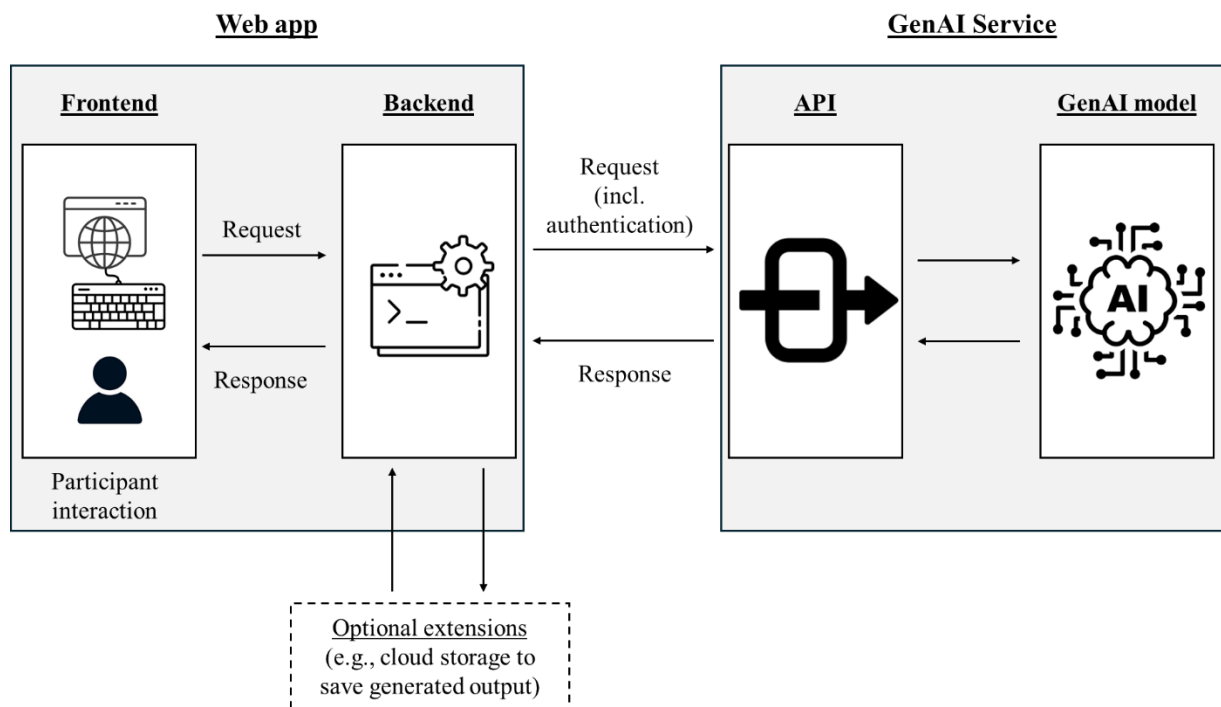


Figure 1. Principal architecture of a GenAI web application.

However, before researchers can dive into the development of a GenAI-driven web application to implement it into their experiments, they need to carefully consider some key decisions. The following table outlines guiding questions to help researchers navigate this process.

Table 2. Guiding questions when developing a GenAI web application for consumer research

GenAI model	Q1.1*	What is my research question?
	Q1.2	What output/modality should the AI create?
	Q1.3	What is the best balance between performance, costs, and latency?
	Q1.4	What characteristics and capabilities should the GenAI have? (e.g., open vs. closed; reasoning vs. not reasoning; web search)
	Q1.5	How large should the model's context window be for the intended interaction or analysis?
	Q1.6	Does the research require specialized model capabilities (e.g., advanced reasoning, creative writing, code generation)?
	Q1.7	What are the implications of model size regarding performance, cost, latency, and potential environmental impact?
API	Q2.1	What are the best API parameters (e.g., temperature, max tokens) to configure for the model?
	Q2.2	How can I provide context to an API call?
	Q2.3	How will the API key be securely managed and used in requests?
	Q2.4	What specific API parameters are needed for the chosen model type (e.g., image style/size for image models, beyond common LLM parameters)?
	Q2.5	How will API request parameters be standardized and documented to ensure study reproducibility?
Frontend	Q3.1	How will participants interact with the GenAI model via the frontend (e.g., text input, voice, drawing)?
	Q3.2	What frontend design choices will optimize the participant interaction experience?
	Q3.3	Where would I host a front-end or can I use existing survey platforms that allow the ingestion of JavaScript?
	Q3.4	How will participant inputs and GenAI outputs (text) be tracked and stored?
	Q3.5	If generating non-text outputs (images, audio, video), how will these be stored externally, and how will their identifiers be linked back to participant data?
Backend	Q4.1*	How much traffic do I expect on my Backend?
	Q4.2	Based on the expected traffic peaks, what are host solutions that offer a good balance between performance and costs to allow the app to work without flaws, problems?
	Q4.3	Is a backend necessary for security (API key protection) and centralized control (rate limiting, validation, moderation)?
	Q4.4	What specific backend responsibilities are needed (e.g., credential management, validation, rate limiting, content filtering, conversation history, non-text output handling)?
	Q4.5	Which hosting model (IaaS, PaaS, FaaS/Serverless) best suits the project's technical expertise, scalability needs, and budget?
	Q4.6	Have potential cloud provider service limits or quotas been checked, and are increases needed for the anticipated research load?
General considerations	Q5.1	How will ethical principles (data privacy, bias, fairness, transparency, informed consent) be addressed throughout the project lifecycle?
	Q5.2	What specific mechanisms will ensure participant PII (Personally Identifiable Information) is protected during interaction (e.g., prompt design constraints, PII filtering/redaction)?
	Q5.3	What monitoring, logging, and error handling strategies will be implemented to ensure application reliability and performance under expected load?
	Q5.4	How will the specific GenAI model version and all relevant API configuration parameters used during the study be meticulously documented for reproducibility?
	Q5.5	How will potential participant connectivity issues to external services (GenAI API, storage) be anticipated and handled (e.g., pre-checks, error messaging)?

Note. Questions marked with (*) require specific input or decisions from the researcher, as the definitive answer depends on the particular study rather than general guidance found in the instructions below.

GenAI model

The GenAI model represents the main value and core functionality in a GenAI web application. In contrast to traditional AI models, which primarily analyze existing data for tasks like prediction and classification, GenAI constitutes a broad category of systems capable of creating novel content (Toner 2023). GenAI models can be differentiated into unimodal (i.e., modality of model input and output is the same) or multimodal (i.e., modality of model input and output is different) models (Cao et al. 2023). The table below provides an overview of available GenAI models across different modalities.

Table 3. Overview of GenAI models categorized by modality.

Input Modality(s)	Output Modality(s)	Description / Common Tasks	Example Models	Key Architectures / Notes
Text	Text	Generating human-like text, conversation, summarization, translation	GPT-4o, Claude 3 Opus, Gemini 1.5 Pro, Llama 3	LLMs (Large Language Models), typically Transformer-based
Text	Code	Generating computer code from natural language or code prompts	GitHub Copilot, AlphaCode 2, Gemini, CodeLlama	Specialized LLMs, often Transformer-based
Text	Image	Creating images from textual descriptions	Stable Diffusion 3, Midjourney v7, DALL-E 3, Imagen 2	Diffusion Models
Text	Video	Creating video clips from textual descriptions	Sora, Runway Gen-3, Make-A-Video	Diffusion Models, Transformers (often combined)
Text	Audio (Speech)	Synthesizing human-like speech from text (Text-to-Speech / TTS)	ElevenLabs TTS, Bark, VALL-E X	Various (Transformers, Diffusion, Flow-based, etc.)
Text	Audio (Music/Sound)	Generating music or sound effects based on text prompts	MusicLM, Stable Audio 2.0, AudioCraft	Transformers, Diffusion Models
Image	Text	Describing images in natural language (Captioning), Analysis	Gemini Series, GPT-4V, BLIP-2, Florence-2	Vision Transformers (ViT), Multimodal Models
Image (+ opt. Text)	Image	Modifying existing images based on prompts or styles	Stable Diffusion (img2img, ControlNet), Pix2PixHD	Diffusion Models, GANs (Generative Adversarial Networks)

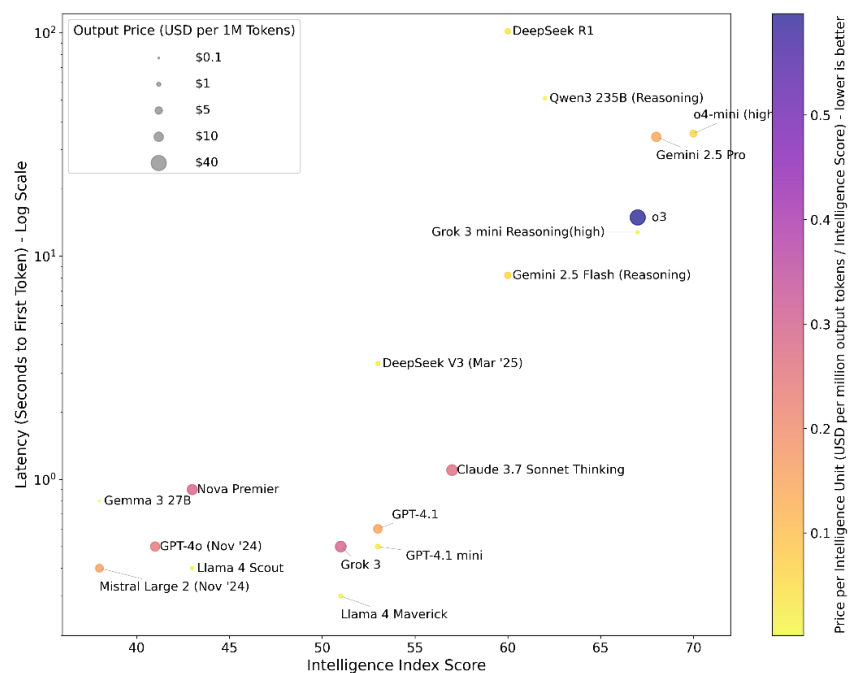
Video	Text / Data	Analyzing video content, generating descriptions or summaries	Gemini Series, Video-LLaMA, various analysis models	Multimodal Models
Audio	Text	Transcribing spoken language into text (ASR / STT)	Whisper, Google Cloud Speech-to-Text	Transformers, CTC-based models

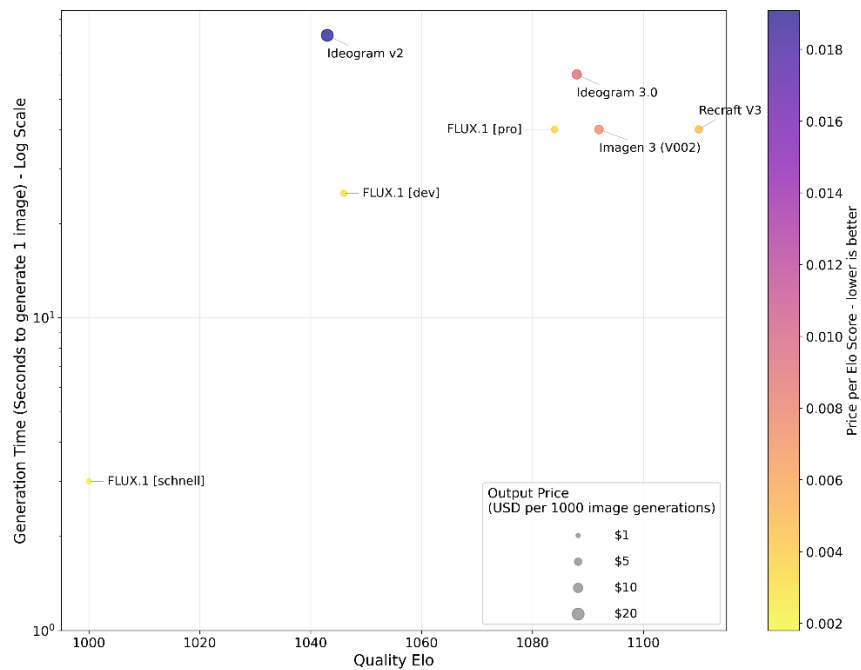
Beyond the modalities that GenAI models can process, the size of the model is another relevant characteristic to consider when deciding which GenAI model to connect to the web application. The size of a model describes the number of parameters that it has. Parameters are essentially the internal variables within the neural network that the model adjusts during its training phase based on the amounts of data it processes (Kaplan et al. 2020). These learned weights and biases determine how the model processes input data and ultimately dictate its ability to understand, predict, and generate output. The scale of the models is vast, with parameter counts ranging from millions in smaller models to hundreds of billions or even trillions in the largest generative models (Bahri et al. 2024). The number of parameters is a critical factor influencing a generative AI model's capabilities and performance. Generally, a larger parameter count allows the model to capture more intricate patterns, nuances, and complex relationships within the data it was trained on, which often translates to improved performance (Hoffmann et al. 2022; Xiao et al. 2024). The concept of "Scaling Laws" suggests that, up to a point, model performance improves predictably as model size, dataset size, and the amount of computation used for training increase (Kaplan et al. 2020). However, while larger models offer potential performance advantages, they come with significant trade-offs. There can be diminishing returns, where the performance gains from adding more parameters become less substantial relative to the increased costs (Bahri et al. 2024). Moreover, larger models entail substantial computational costs for ongoing use (inference; Luccioni, Jernite, and Strubell 2024), which has bigger environmental impacts

and, particularly crucial for web applications used in research experiments, come along with higher pricing and latency of the models (Zewe 2025).

While pricing dictates the monetary cost associated with using a GenAI model, latency refers to the response time – the duration it takes for the model to process an input and generate an output. These factors—intelligence, cost, and latency—often exist in tension. The figures below aim to provide a comparative overview of various generative AI models (i.e., text and image creation), illustrating their relative positions concerning intelligence, price, and latency.

Figure 2. Comparative analysis of language and image generation models across quality metrics, response times, and cost efficiency (Data source: Artificial Analysis 2025).





Note. The visualizations compare model performance using domain-specific quality metrics. For text models, the Intelligence Index Score combines results from MMLU-Pro, GPQA Diamond, Humanity's Last Exam, LiveCodeBench, SciCode, AIME, and MATH-500 to measure cognitive capabilities. For image models, the Quality Elo score represents relative performance based on over 100,000 user evaluations collected through Artificial Analysis' Image Arena (Artificial Analysis 2025).

Hence, when deciding which model to connect to the web application, the intended use case and its specific requirements must be carefully considered. It is crucial to understand that the best model is context-dependent; the largest model, while potentially powerful, often incurs the highest costs and may introduce latency detrimental to user experience. Consequently, developers should often experiment with different models to identify the optimal balance between desired output quality, acceptable latency, and budget constraints. There exist tools that help researcher make objective comparisons on model performances and quality (e.g., <https://artificialanalysis.ai/>; <https://lmarena.ai/>)¹.

Beyond these core trade-offs, several other characteristics differentiate GenAI models, which are particularly relevant when selecting the right model for the web application. One critical factor, particularly for LLMs, is the context window. This defines the amount of input

¹ For more extensive discussions on measuring model performance and quality see for example (Weidinger et al. 2025)

text (typically measured in tokens, often a fraction of one word) the model can simultaneously consider when creating an output (Amazon Web Services 2025a). A larger context window enables the model to maintain coherence and recall information over longer conversations or analyze larger documents, significantly impacting user experience in dialogue-based or analytical applications. Context window sizes vary greatly, ranging from models handling tens of thousands of tokens (e.g., 128k) to those capable of processing millions of tokens (“Comparison of AI Models across Intelligence, Performance & Price Analysis” 2025).

Another key differentiator for LLMs lies in the specialized functionalities or optimizations of the models. Some models are specifically designed or fine-tuned for reasoning, which involves the ability to perform logical deduction, solve complex problems, understand cause-and-effect relationships, follow multi-step instructions, and perform mathematical calculations based on the provided context. Other models might be optimized for creative writing, code generation, or possess built-in capabilities to integrate real-time web search results into their responses, ensuring access to up-to-date information. Evaluating these functional strengths is vital for aligning the model's capabilities with the application's core purpose. These diverse factors—context window length and specialized functionalities—must be weighed by researchers during the model selection process.

Another characteristic that can differentiate GenAI models is whether they are open or proprietary models. Open models, often referred to as open-source, are models where aspects like the architecture, training methods, or even the model weights themselves are made publicly available, sometimes allowing them to be downloaded and run locally (Bi et al. 2024). While these open models allow transparency, reproducibility, and greater user control (including local execution for data privacy), proprietary models typically lack such characteristics (Hussain et al. 2024). However, to fully gain the advantages of open-source

models they need to be run locally, and the requirement for substantial infrastructure—like powerful graphic processors and large amounts of random-access memory (RAM)—becomes particularly relevant if these locally-run models are expected to deliver performance comparable to that of large proprietary GenAI models.

Thus, most researchers without such an infrastructure would potentially connect GenAI services into the web application via an Application Programming Interface (API), which provides a standardized way for the web application to communicate with the GenAI model, send requests, and receive responses. A well-documented and robust API facilitates easier implementation and often allows developers to configure various parameters that influence the model's behavior (e.g., creativity/randomness, maximum output length). The ease of integration, stability, and control afforded by the API significantly influence development speed and flexibility, along with the ability to manipulate the model's interaction within the application specifically for research objectives. The next chapter will provide an introduction to using APIs to effectively connect GenAI models to web applications.

Application Programmable Interface (API)

An API serves as the gateway through which a software program communicates with other programs, and—in the case of web APIs—with the broader digital ecosystem across the internet (Jin, Sahni, and Shevat 2018). While there are different forms of APIs, one of the most popular forms, also used to access GenAI services, is the Representational State Transfer API (REST; Fielding 2000). REST API is an architectural approach, based on the concept of resources, which represent information on a server (e.g., ChatGPT model on OpenAI's servers). Similar as any website on the web has a unique resource identifier (URI; e.g., “<https://www.amazon.com/>”) every resource that is used via API has a unique resource identifier too (e.g., one endpoint of OpenAI's Chat models is:

“https://api.openai.com/v1/chat/completions”²). The URIs do also reveal another architectural characteristic of REST APIs, which is that they are based on the http protocol, the foundational protocol for data exchange on the web (Doglio 2018).

A further characteristic of REST APIs is their statelessness (Fielding 2000). This means each API request must contain all information necessary for the server to process it independently, without relying on previous interactions. For GenAI service requests, this requires including all data the model needs to generate its output, whether text, image, or video. REST API requests follow a standardized structure with key components: the URI (path to the resource), and the HTTP method indicating the desired action—GET for retrieving data, POST for submitting data, PUT for updating resources, and DELETE for removing resources (Jin et al. 2018). When interfacing with GenAI services, the POST method predominates as it allows users to submit prompts and parameters to the model for output generation.

API requests also contain a Header section with critical metadata. The "Content-Type" field specifies the format of submitted data (typically JSON for GenAI services), while the "Authorization" field provides credentials—usually an API key—that authenticates the user. Since GenAI providers charge for model usage, API keys link specific accounts with payment information and must be safeguarded against public disclosure. Researchers developing GenAI applications for consumer studies would obtain an API key from their chosen AI service provider (e.g., OpenAI, Google, Anthropic) and integrate it securely into their application's API requests (Jin et al. 2018).

The Body component completes the request, specifying which AI model to utilize and containing the actual prompt or input data. This structured approach enables precise

² Status: 4th of April 2025

communication between the application and the chosen GenAI service. The figure below illustrates an example API call to ChatGPT's responses endpoint using the curl method (a common command style):

```
curl -X POST "https://api.openai.com/v1/chat/completions" \
-H "Content-Type: application/json" \
-H "Authorization: Bearer $OPENAI_API_KEY" \
-d '{
  "model": "gpt-4o",
  "messages": [
    {
      "role": "user",
      "content": "Write a one-sentence bedtime story about a unicorn."
    }
  ]
}'
```

Figure 3. Exemplary API call to the chat completion endpoint of OpenAI (OpenAI n.d.).

API calls offer researchers both flexibility and standardization, presenting both methodological opportunities and challenges. On one hand, the configurable nature of API parameters allows researchers to systematically manipulate experimental conditions—a valuable capability for experimental research. On the other hand, this flexibility requires careful documentation and standardization to ensure reproducibility of findings across studies. Without proper parameter specification, research outcomes may vary significantly even when using identical GenAI services. The table below outlines some key components that can be configured within API requests to an LLM. While not exhaustive, this overview covers fundamental parameters that researchers can manipulate across most major GenAI platforms, though specific options may vary by service provider. Moreover, researchers who would use the API of a GenAI different than LLM (e.g., image generation) would set other parameters within the API request (e.g., image style or image size).

Table 4. Potential parameter settings when making an API call to an LLM.

Parameter	Description
Model	Defines the model that should be used by the API request (e.g.: “gpt-4o”, “gemini-2.0-flash”).

Role	Interactions with GenAI models consist of different roles: <ul style="list-style-type: none"> - System: These messages define how the model is supposed to behave (e.g.: “Be a helpful assistant.”; “Always answer in Shakespear manner.”) - User: These messages represent that content of the request to the model (e.g., “write a poem”, “summarize the following text”). - Assistant: These messages represent the output generated by the model.
Maximal tokens	Defines the maximal number of tokens that the model uses to generate the output.
Temperature	Defines the level of randomness in the generated output. The lower the temperature the more deterministic and focused is the output. Increasing the temperature increases the variability of generated output.
Reasoning effort	Can only be used for reasoning models. It defines the token length that is used in context of the model’s reasoning. The higher the defined reasoning effort the more tokens the model uses for thinking about the answer.
Streaming	Defines if the generated output is sent at once or in parts. Enabling streaming allows that the generated output is sent in segments as it is being generated, making the response appear incrementally rather than waiting for the full output.
Frequency penalty	Defines how models handle word repetition in their outputs. The higher the penalty the lower the likability that the model repeats phrases or sentences.
Number of completions	The model will generate the specified number of alternative responses to the same prompt, allowing to select from multiple options or compare variations in the model's approach to answering the query.

Note. The table was created based on information provided in the documentary of both Fireworks AI (Fireworks AI n.d.) and Google’s AI for Developers documentary (Google n.d.).

While APIs provide the access to the GenAI models and thus, the core intelligence, functionality, and value of GenAI web applications, users need an interface to interact with the application. This graphical user interface (GUI), or frontend, serves as that interaction layer (Amazon Web Services 2025b).

Frontend

A web front-end is built using Hypertext Markup Language (HTML), Cascading Style Sheets (CSS), and JavaScript (Simon 2023). While HTML defines the content and structure (Berners-Lee and Connolly 1993), CSS provides the design (Flanagan 2011), and JavaScript

defines the front-end's behavior and interactivity (Kalim 2013). Notably, only JavaScript functions as a programming language enabling the front-end to access and interact with the API.

For user accessibility, a web front-end requires online hosting (Aquino and Gandee 2017). While researchers can build custom front-ends, an alternative is leveraging survey platforms like Qualtrics. These platforms often allow embedding custom HTML, CSS, and JavaScript within survey questions, enabling GUI adaptation and interaction with external APIs (Qualtrics 2025). Specifically, JavaScript embedded within a Qualtrics question can be configured to make requests to a target GenAI API. This script would include necessary parameters for the API call, such as the endpoint URL, system prompt instructions, and model settings (e.g., temperature). This integration allows researchers to implement various GenAI interaction modalities within the survey environment.

The key advantage of implementing the GenAI front-end within a controlled research environment like Qualtrics is that it empowers researchers to systematically manipulate GenAI characteristics and directly observe participant interactions as part of the study workflow. Qualtrics allows the use of "Embedded Data" fields – custom variables where data can be stored alongside survey responses. The embedded JavaScript can interact with these fields. For instance, the script can capture the participant's prompt before sending it to the GenAI API and write this text into a designated Embedded Data field.

Subsequently, when the JavaScript receives the response from the GenAI via the API, it can also save this information. If the response is text-based, the script can write the content into the same or another Embedded Data field. This process allows researchers to automatically log both sides of the human-AI interaction (participant prompts and GenAI text responses) directly within the Qualtrics dataset for later analysis. When dealing with non-text

GenAI outputs (e.g., images, audio, video), the approach must be adapted as Embedded Data cannot store these files directly. Instead, the embedded JavaScript needs to interact with an external cloud storage service (like Cloudinary, AWS S3, etc.). Upon receiving the non-text output from the GenAI API (which might be a direct URL or the file data itself), the JavaScript code must ensure the file is stored externally. This might involve uploading the data directly to the storage service via its API (including handling authentication). Once the file is stored, the script's primary task for Qualtrics integration is to capture the resulting persistent URL or identifier of that stored media. This unique identifier is then saved into a designated Qualtrics Embedded Data field, effectively linking the generated non-text output back to the participant's survey response for analysis.

While the direct interaction between front-end and GenAI API is relatively simple to realize, it bears some disadvantages, primarily concerning security and centralized control (Microsoft Learn 2024). Exposing API keys directly in the front-end JavaScript is a major security vulnerability, risking unauthorized access and expenditure (Lu 2014). Furthermore, this approach makes it difficult to implement uniform controls like participant-specific rate limiting, input validation, content filtering before display, or consistent prompt templating across all interactions without relying entirely on the code execution in the user's browser, which can be less reliable or easily tampered with (Microsoft Learn 2024). While the risk of deliberate tampering might seem low in many consumer research studies with cooperative participants, the inherent unreliability of code execution in the user front-end persists (due to browser differences, network issues, etc.). To overcome these limitations and establish proper security and control, the implementation of an intermediary server-side component—a backend—becomes essential. Admittedly, developing and maintaining an intermediary backend adds significant technical complexity for the researcher compared to a purely frontend side solution. However, for enhanced security, reliability, and control, it is

considered best practice (McDaniel and Segura 2024). This backend manages the communication between the user interface and the GenAI API, providing a more robust and professional architecture overall.

Backend

A backend component acts as a secure additional layer, mediating all communication between the user interface (e.g., JavaScript running within a survey platform) and the downstream Generative AI (GenAI) server, as well as any other required external services like cloud storage. This architectural approach aligns with the aim to decouple the user interface from complex or sensitive backend systems (Newman 2015).

Fundamentally, this backend is software code—often a script developed using programming languages like Python, Node.js, or Java—that executes on a remote server, rather than directly within the participant's web browser. A server, in this context, is essentially a computer maintained by the researchers or a hosting provider, kept online and accessible over the internet. Unlike the front-end code that renders the user interface and runs locally on the participant's device, the backend code runs remotely. Its primary role here is to receive specific requests sent from the front-end code over the network, perform predefined tasks (such as securely communicating with the GenAI API using protected credentials), and then send the results of those tasks back to the front-end for display to the user.

The core responsibilities of this backend encompass several critical functions essential for robust and secure application development. Foremost is secure credential management; API keys and other secrets required for GenAI or cloud storage APIs (e.g., to store created image, video or audio data) are stored and utilized exclusively within the trusted server environment, and never exposed on the user side (i.e., the browser of the study participant). The backend also handles the full lifecycle of API interactions: It receives requests entered

into the interface on the user side, performs necessary server-side validation, constructs the complete, authenticated requests for connected services (i.e., GenAI, cloud storages) using secure credentials, processes the received responses (potentially filtering or formatting them), and finally transmits the appropriate result back to the user. Furthermore, the backend provides a locus for centralized control and logic implementation. For example, a backend can enforce participant-specific or global rate limits to manage costs and prevent abuse, or filter and moderate AI-generated responses before they are presented to the user interface. The backend can also keep track of ongoing interactions, like storing the history of a conversation, if that's necessary for how the application works. For non-text data outputs, the backend orchestrates the interaction with external storage services, handling file uploads and returning persistent identifiers (e.g., URLs) that can be stored on the user side to enable connection and identification between the generated output and the user interaction in the front-end.

Unlike the front-end code, which is executed within the participant's web browser, the backend must operate from a stable, continuously running environment. Because the front-end application (potentially accessed by numerous participants simultaneously from different locations) needs to reliably send requests to the backend at any time during the research study, the backend code must be deployed on a server infrastructure that maintains persistent connectivity to the internet. This process of deploying the application and ensuring its constant availability and accessibility over the network is known as hosting. Consequently, selecting and configuring an appropriate hosting solution is a prerequisite for the backend to fulfill its intermediary role effectively.

The choice of hosting infrastructure typically involves selecting from standard cloud computing service models, which primarily differ in the level of abstraction provided and the scope of management responsibility retained by the user versus the cloud provider (Mell and

Grance 2011). For research applications that mediate interactions with GenAI APIs, the hosting decision carries implications for cost, scalability, and maintainability. Common models include:

- Infrastructure-as-a-Service (IaaS): This model provides access to fundamental computing resources such as virtual servers, storage, and networks. While offering maximum flexibility and control over the environment, IaaS necessitates that the research team manages the entire software stack, including the operating system, middleware, security patching, and the application runtime (Armbrust et al. 2010). This demands significant technical expertise and continuous operational effort. (Examples: Amazon EC2, Google Compute Engine, Azure Virtual Machines).
- Platform-as-a-Service (PaaS): PaaS abstracts away the underlying infrastructure management (hardware, operating systems, networking), allowing developers to focus primarily on deploying and managing their applications. The provider manages the platform, including system updates and often provides integrated services like databases and scaling mechanisms (Mell and Grance 2011). This significantly reduces the operational burden compared to IaaS, though it may offer less granular control over the environment. (Examples: Heroku, Google App Engine, AWS Elastic Beanstalk).
- Function-as-a-Service (FaaS) / Serverless Computing: Representing a higher level of abstraction, FaaS allows developers to deploy code in the form of stateless functions triggered by specific events, such as incoming HTTP requests from the front-end. The provider entirely manages the underlying infrastructure, including automatic scaling based on real-time demand (Castro et al. 2019). This model eliminates server management tasks for the researcher and typically follows a pay-per-execution pricing model, making it potentially very cost-effective for applications with variable

workloads (Jonas et al. 2019). (Examples: AWS Lambda, Google Cloud Functions, Azure Functions).

For most experimental research studies serverless (FaaS) architectures are often well-suited as they require relatively little technical expertise and due to their inherent elasticity and pay-per-use nature, minimizing costs during periods while automatically handling load spikes without manual intervention (Baldini et al. 2017; Jonas et al. 2019). PaaS can also offer viable scaling options, although they may necessitate user configuration. However, the choice of hosting model can also depend significantly on the GenAI's output modality. Applications dealing with more resource-heavy outputs, such as those involving image or audio generation and processing via APIs, might find that IaaS, despite requiring the most intensive effort in capacity planning and scaling management, becomes necessary. This is because IaaS offers maximum environmental control and the ability to provision specific, potentially more substantial, resources for compute power or network bandwidth often needed for these demanding tasks (Buyya, Vecchiola, and Thamarai Selvi 2013). A critical consideration across all models (IaaS, PaaS, FaaS) is the presence of default service limits or resource quotas commonly imposed by cloud providers, especially on new accounts. These quotas act as financial safeguards but can impede performance during peak research activity if not addressed. Therefore, researchers should proactively assess these limits and request increases from the provider as needed to ensure the hosting solution can accommodate the study's anticipated maximum participant load (Digital Ocean 2025).

Operational and Ethical Guidelines

Ethical aspects. Researchers must recognize that utilizing external GenAI services introduces the potential for interaction tracking by the service provider. While the primary research goal involves observing participant interactions and their outcomes, the GenAI

service provider may also have access to this interaction data (Weisz et al. 2024).

Consequently, it is essential to disclose the use of external GenAI services to participants before they begin interacting with the web application and to obtain their explicit informed consent. Furthermore, system instructions (prompts) provided to the GenAI model should be carefully designed to avoid soliciting sensitive or personally identifiable information (PII) from participants (Carlini et al. 2018). This is particularly critical if the study context involves collecting personal details separately (e.g., demographic data); researchers must ensure such pre-collected sensitive information is not subsequently fed into the connected GenAI service (e.g., via methods like embedded data fields in survey platforms).

To mitigate the risk of participants inadvertently submitting PII within their inputs, researchers can also consider integrating automated PII detection and redaction mechanisms within the web application's backend architecture. Such mechanisms can identify potential PII in user input before it is sent to the external GenAI API. Depending on the implementation, the system could either block the transmission and prompt the participant to revise their input, or automatically mask/redact the identified PII before forwarding the request. Various PII detection tools and services are available; some operate as external APIs themselves (requiring careful vetting of their Data Processing Agreements), while others, particularly open-source frameworks (e.g., Microsoft Presidio), offer greater control over data privacy during the detection phase.

Technical and performance aspects. Ensuring the technical robustness and adequate performance of the web application is crucial for a smooth participant experience and reliable data collection. Key considerations include application reliability, monitoring, scalability, and connectivity. First, the application must function reliably. This involves implementing robust error handling within the backend code to manage unexpected situations, such as malformed data from the frontend or unexpected responses (including errors) from the external GenAI

API. Ongoing monitoring is essential to verify functionality and diagnose issues (Silva 2008). Hosting platforms often provide tools to monitor server health, resource utilization, and traffic patterns. Additionally, application-level logging (both on the server-side and potentially user side via the browser console for user-specific issues) is vital for tracing requests, identifying the root causes of errors, and debugging the application.

Second, the application must reliably handle the anticipated participant load. Effective management of concurrent users is essential to prevent backend overload, which can cause slow response times or service failures, especially during peak activity. Achieving sufficient scalability requires selecting an appropriate hosting solution (e.g., IaaS, PaaS, FaaS) and ensuring resource quotas are adequate for the expected traffic. While PaaS and FaaS offer significant advantages in automated scaling and simplified management, researchers might consider IaaS for particularly resource-intensive GenAI tasks, such as those involving audio or image generation via APIs, where granular control over network bandwidth or specific compute resources could be beneficial, despite the higher management overhead (Nadeem 2022).

Finally, external factors like network connectivity can impact functionality. Some participant networks (e.g., corporate, institutional) might restrict access to external GenAI APIs due to security policies. However, this issue is limited if the API is not directly called from the user interface but via the hosted backend. Nevertheless, implementing a preliminary check upon study initiation to verify connectivity to the required API endpoint can prevent participants from starting a study they cannot complete. This way, if connectivity between the user's frontend and the GenAI fails, the participant can be gracefully informed on the first page of the study.

Operational and research aspects. When utilizing GenAI models via external APIs (i.e., not hosted locally), researchers must be cognizant that the underlying model provided by the service may be updated or changed over time by the provider. Such changes, even minor version updates, could potentially alter the model's behavior and responses. Therefore, for research validity and reproducibility, it is imperative to meticulously document the specific model (including version, if available) used during the study period. Additionally, all API configuration parameters (e.g., temperature, max tokens, system prompts) employed in the API calls must be precisely recorded as part of the study's methodology. Tracking the model endpoint or version and rigorously documenting configurations helps ensure that results can be accurately interpreted and potentially replicated (Ganguly et al. 2025).

AI augmented development environments

Building the GenAI-powered web applications described above, which typically involve distinct frontend (user interface) and backend (server-side logic) components, inevitably requires some level of coding. For instance, JavaScript is commonly used for frontend interactivity within platforms like Qualtrics, while languages such as Python or Node.js are often employed for the backend server that securely communicates with the GenAI API. For researchers whose primary expertise lies outside of software development, this technical requirement can understandably represent a significant barrier to designing and implementing otherwise valuable GenAI intervention studies.

Fortunately, the same generative AI technology that powers these described web applications also provides powerful tools to aid in their development. AI-augmented development environments and specialized code assistance tools (sometimes referred to as *vibe coding*) can dramatically lower this coding barrier, acting as intelligent partners throughout the development process. Prominent examples include GitHub Copilot, Cursor,

Windsurf AI, and various other solutions embedded within Integrated Development Environments (IDEs). These tools can generate code snippets, complete lines or entire functions, refactor existing code, and even explain complex segments in natural language, based on the researcher's prompts or the existing code context.

A key advantage of these integrated tools, particularly relevant for the architecture described in this paper, is their context awareness. Unlike querying a general-purpose chatbot, these tools can analyze the entire project structure, including the frontend JavaScript files and the backend Python or Node.js scripts. This allows them to generate code that correctly connects these distinct parts. For example, a researcher might have backend code that specifies how participant data should be received from the frontend interface. Needing the corresponding frontend logic, the researcher can simply ask the AI assistant: “Generate code to get the participant's input and send it to our backend.” The AI assistant uses its context awareness by looking at the actual backend code to understand precisely how that data needs to be formatted and transmitted. By combining the researcher's stated goal with its understanding of the backend's requirements, the AI can generate the necessary frontend code to correctly capture and send the participant's input. In practical terms, these AI coding assistants can provide tangible help with tasks outlined in the previous sections.

- Frontend Development (addressing Q3.1-Q3.5): Generating the necessary JavaScript code to embed within survey platforms like Qualtrics, creating interactive elements (input fields, buttons, response display areas), capturing participant inputs, dynamically displaying GenAI text responses (potentially handling streaming for better UX), and crucially, writing the asynchronous API calls (Workspace or XMLHttpRequest) to communicate with the backend server.

How to achieve it: Researchers can describe the desired frontend component or behavior in natural language.

Example Prompt: Generate JavaScript for a Qualtrics question with an HTML text input, and a submit button. When the button is clicked, the script should take the text from 'userQuery', send it as a JSON message body via a POST request to a backend endpoint, then display the 'answer' field from the JSON response in the user interface.

- Backend Development and GenAI API Interaction (addressing Q2.1-Q2.5, Q4.1-Q4.6, and related aspects of Q5.3): AI coding assistants can also support the drafting of server-side code, spanning the entire backend lifecycle for a GenAI application. This typically includes setting up API endpoints (routes) to receive requests from the frontend and implementing logic to securely manage and inject sensitive GenAI API keys (e.g., from environment variables, ensuring they are kept off the frontend as per Q4.3). Furthermore, these tools can help to correctly format the required data input (often JSON), suggesting appropriate and effective API parameters such as temperature, max_tokens, or system_prompt based on high-level descriptions of desired AI behavior (addressing Q2.1, Q2.2, Q2.4). They can also assist in implementing robust error handling for network requests or specific API errors originating from the GenAI service (as per Q5.3).

How to achieve it: Researchers can leverage this assistance by specifying the desired backend language, and the overall functionality of their API endpoint. For the specific part of interacting with the external GenAI service, researchers can ask more detailed questions about the structure of the data input, optimal parameters, or error handling strategies, thereby ensuring all API request parameters are standardized and documented for reproducibility.

Example prompt: Create a Python script that accepts POST requests with a JSON data input containing a 'user_prompt'. First, securely load an 'MY_GENAI_API_KEY' from an environment variable. Second, construct and execute a call to the GenAI

XYZ's chat API (using the 'model-alpha' model), ensuring the user's prompt is the main input and including a system message like 'You are an insightful assistant for academic research.' Also, suggest and set parameters for a balanced, factual, yet moderately creative response, like a suitable temperature and appropriate max_tokens for a detailed paragraph, and implement try-except error handling for this external API call, logging any network or API errors. Third, extract the relevant AI-generated text from the service's response. Finally, return this text in a JSON object like {'ai_response': 'The AI's answer.'} to the frontend.

Expected output: The AI assistant would likely generate Python code including the specified '/api/callGenAI' route. This generated code would typically encompass logic for parsing the incoming JSON request; securely retrieving the API key; correctly constructing the JSON data input for the target GenAI API, complete with the system message, user prompt, and suggested parameters (e.g., proposing temperature: 0.6 and max_tokens: 250); using a Python HTTP client library for making the POST request; implementing a comprehensive logic to log errors; parsing the GenAI service's response; and formatting the final JSON response for the frontend.

While these AI tools are powerful facilitators, it is crucial to approach them as assistants, not infallible replacements for understanding. The code they generate must still be reviewed, tested, and understood by the researcher to ensure it is correct, efficient, secure (especially regarding API keys and data handling), and accurately implements the intended logic for the study (Q5.4). Thorough testing of the application's functionality and error handling remains essential. Furthermore, researchers should be aware that many advanced tools operate on subscription models and, like all GenAI systems, can occasionally produce incorrect or suboptimal code.

Examples and showcases

To illustrate the practical application of the architectural framework outlined above, the following section presents two studies that utilized custom-built GenAI web applications (i.e., text and image generation) to investigate consumer interaction. For both studies, Qualtrics was used as the user frontend, calling a Python coded backend that was hosted online and running the GenAI interaction logic. The used Qualtrics files and the full backend code are publicly available (<https://github.com/GenAI-interaction-research>).

LLM input modality on recommendation usage. For this study a web application was developed to test whether the input modality (writing vs. speaking) for an LLM influenced user engagement with and perceptions of AI-generated podcast recommendations. In designing this application, the "Gemini-2.5-flash-preview-04-17" LLM was chosen as the core GenAI service and was integrated via its API. The LLM was specifically prompted, using a 'system' role message as discussed in the API section, to act as a podcast recommendation system. The frontend of the web application was embedded in Qualtrics. For the spoken voice input condition, the study employed the Web Speech API, which enables speech-to-text functionality directly within compatible web browsers. To ensure this capability, a preliminary check screened participants' browsers, permitting only those with adequate support to enter the study. Participants were then randomly assigned to one of two conditions, either providing their podcast preferences and requests to the LLM via written text input or via spoken voice input. This setup allowed for the direct manipulation of the interaction modality. Following the interaction with the LLM participants, participants answered different measures. Perceived attributes of the AI were evaluated on a 7-point semantic differential scale using bipolar adjectives (distant-close, cold-warm, impersonal-personal, detached-engaging, and machine-like-human-like). Additionally, specific constructs were measured using three-item scales for: perceived understanding by the AI (e.g., "I felt the AI grasped the actual point I was trying to make."); satisfaction with the AI (e.g., "I am

satisfied with how the AI performed”); and satisfaction with the recommendation itself (e.g., “The provided podcast recommendation fits my preferences.”).

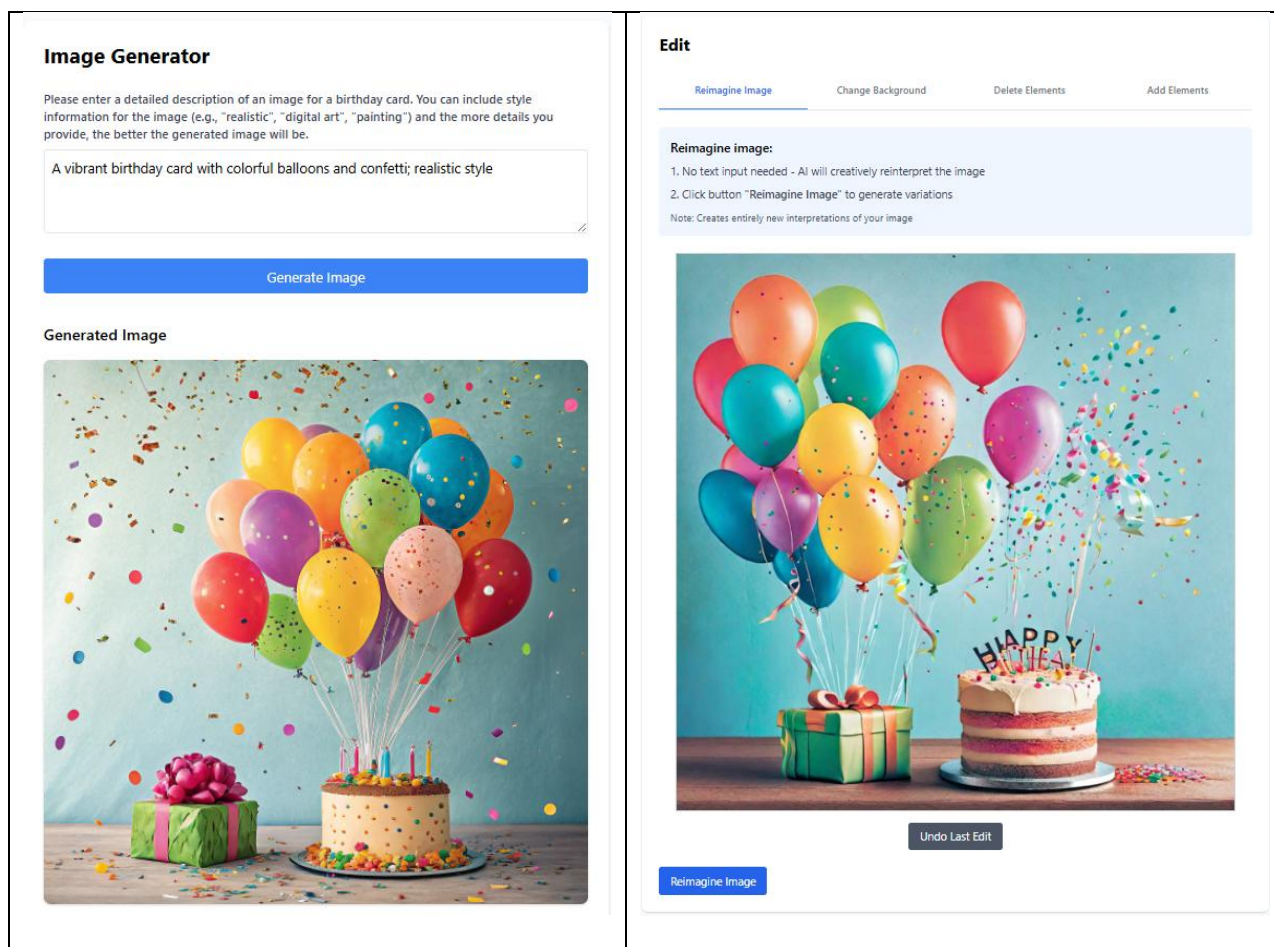
Analysis of system usage revealed significant differences based on input modality. Participants in the voice (speaking) condition engaged in significantly more interaction turns with the AI ($M_{speak} = 7.47$, $SD = 3.19$) compared to those in the text (writing) condition ($M_{write} = 4.88$, $SD = 3.01$, $p < .001$). Furthermore, spoken interactions were characterized by a greater number of words per turn on average ($M_{speak} = 19.23$, $SD = 41.72$) than written interactions ($M_{write} = 4.63$, $SD = 4.86$, $p = .001$). In contrast to these engagement differences, an examination of perceptual and attitudinal outcomes showed no significant variations between the voice and text conditions for the perceived attributes of the AI (all $ps > .25$), perceived understanding by the AI ($M_{speak} = 5.64$, $SD = 1.33$, $M_{write} = 5.54$, $SD = 1.23$, $p = .548$), or satisfaction with the AI's performance ($M_{speak} = 5.83$, $SD = 1.38$, $M_{write} = 5.64$, $SD = 1.47$, $p = .347$). However, a marginal effect was observed for satisfaction with the podcast recommendations. Participants in the voice condition tended to report higher satisfaction with the received recommendations ($M_{speak} = 5.81$, $SD = 1.18$) compared to those in the text condition ($M_{write} = 5.48$, $SD = 1.53$, $p = .084$). A mediation analysis finds that the number of interactions mediates the positive effect of the voice input modality on the satisfaction with the recommendation (indirect effect: $b = 0.23$, $SE = 0.09$, 95CI [0.07; 0.41]).

Image generation and editing. To further illustrate the framework's application beyond text output, we developed a web application for an experimental setting. This application facilitated controlled participant interaction with AI services for image generation and editing, with the primary aim of investigating how the opportunity to refine AI-generated output influences user perception and perceived value. The user interface was again embedded within the Qualtrics survey platform, facilitating seamless integration with the overall study flow and data collection instruments. This frontend served as an interface to

specific external GenAI services accessed via their APIs; namely, the Recraft AI API handled initial image generation based on participant text prompts, while the Clipdrop API provided functionalities for subsequent image editing in the relevant experimental condition, including replacing backgrounds via text, removing objects, generating stylistic variations, and performing masked inpainting based on text prompts.

Given the potentially significant computational demands and external service interactions associated with these image processing APIs, an Infrastructure-as-a-Service (IaaS) solution, specifically Digital Ocean, was selected to host the backend, ensuring sufficient, scalable resources were available to handle concurrent participant requests reliably. This backend managed secure communication with the Recraft and Clipdrop APIs using protected credentials, shielding API keys from the client-side. Furthermore, the backend orchestrated the data flow: participant prompts and the count of refinement actions undertaken were stored as Qualtrics Embedded Data fields. Adhering to the best practice for handling non-text outputs, generated image assets were stored externally using Cloudinary, with unique identifiers (URLs) for the stored images then saved back into Qualtrics Embedded Data, effectively linking the visual output to the respective participant's response record.

Figure 4. Examples of image generation and editing (i.e., reimagination of image)



Using this integrated system, a study was conducted with three hundred seventy-nine participants recruited via Prolific. They were randomly assigned to one of two conditions: either generating an image for a birthday card using a text prompt, or generating the image and then being given the opportunity to refine it using the available editing tools. Following the image creation or refinement task, participants answered different measures. Among others, they rated their liking of the final image on a 7-point scale (1: dislike a great deal – 7: like a great deal), indicated a selling price for an online platform using a slider (\$0 - \$5), and stated whether they would disclose the use of GenAI if selling the card (yes/no).

Participants reported significantly higher liking for images they generated but did not refine ($M_{no_refine} = 5.84$, $SD = 1.32$) compared to those who had the opportunity to refine ($M_{refine} = 5.22$, $SD = 1.80$, $p < .001$). Similarly, the opportunity to refine led to a marginally

lower proposed selling price ($M_{no_refine} = \$2.86$, $SD = 1.34$ vs. $M_{refine} = \$2.58$, $SD = 1.34$, $p = .084$). No significant differences emerged regarding participants' willingness to disclose the use of AI when hypothetically selling the card (Disclosure rate: 45.6% in no-refine vs. 46.9% in refine condition, $b = -0.11$, $SE = 0.21$, $p = .606$).

General discussion

Text

To further facilitate the implementation of GenAI web applications in survey and experimental studies, a GitHub has been created (<https://github.com/GenAI-interaction-research>) that provides interested researchers access to a

- Github to upload surveys
- Complete apps: Frontend & Backend scripts
- Instructions how to use the AI augmented coding tools, including a rule script

Potential research avenues

While early research demonstrated very promising directions of involving GenAI in experimental studies, its potential impact extends across a much broader spectrum. For example, a primary area of transformation by GenAI concerns how consumers seek information online and subsequently make decisions. The growing influence of GenAI tools is already evident in shifting web traffic patterns; for instance, analyses of 80 million clickstream records indicate that web traffic is broader distributed across the web as ChatGPT refers to more unique domains in comparison to traditional search engines (Kelly 2025). However, how people are using GenAI to search for information, how much they trust the

provided information, and how the provided information affects people's behavior is not investigated. Hence, a promising avenue for future research is to look at how XXX



Beyond information retrieval, GenAI agents are increasingly anticipated to assist with or even automate complex consumer tasks, ranging from event planning to executing purchases (Cai et al. 2025).

Beyond what consumers do with GenAI, a critical research stream examines who benefits from these interactions and the potential for differential impacts. Current findings present a complex picture regarding user skill levels: some studies suggest GenAI primarily benefits lower-skilled users by augmenting their capabilities (Brynjolfsson et al. 2023), while others propose that higher-skilled individuals may derive greater advantages, potentially exacerbating existing inequalities (Roldan 2024). Similarly, disparities in adoption and usage patterns have been observed along gender lines, with some research indicating lower usage rates among women across various occupations (Humlum and Vestergaard 2025).

Complementing the focus on user outcomes, another vital research avenue involves exploring the impact of GenAI's technological characteristics and interaction design. The manipulable dimensions of the technology are vast. While researchers have begun exploring variations in perceived attributes like personality (Ju and Aral 2025), significant opportunities remain in examining how nuances in GenAI's communication style affect user perception and behavior. For example, user preferences might shift between machine-like and human-like communication depending on the valence of the information being conveyed (Garvey, Kim, and Duhachek 2023). Understanding these preferences is critical, especially given reported user dissatisfaction with some contemporary chatbot interactions (Castelo et al. 2023). Furthermore, fostering critical engagement with GenAI outputs represents a significant challenge (Lee et al. 2025). Although GenAI can enhance performance, uncritical acceptance

or simple "copy-pasting" of its suggestions may degrade final output quality. Designing GenAI interactions that encourage, rather than bypass, critical thinking is an important research goal (Anthropic 2025). Finally, the modality of interaction—primarily text versus voice—warrants investigation. Research in related contexts demonstrates that interaction modality significantly impacts user satisfaction, behavior, and willingness to share information (Melumad 2023; Melzner, Bonezzi, and Meyvis 2024). How different modalities shape user interactions and outcomes specifically within the GenAI context remains largely unexplored. To illustrate the research directions discussed above, the following table proposes specific exemplary research questions, acknowledging that these represent only a sample of the vast investigative opportunities in this field.

Table 1. Exemplary research questions to investigate with GenAI web applications

	User focus		Technology focus
	<ul style="list-style-type: none"> - How do consumers utilize Generative AI tools for information seeking, and what are the characteristics (e.g., perceived accuracy, relevance, completeness) of the information retrieved? - How are consumer decision-making processes (e.g., in purchasing, planning, problem-solving) influenced by interaction with Generative AI agents or systems? - How do consumers perceive the process and outcomes of co-creation with Generative AI, including aspects of agency, creativity, and ownership? - How does reliance on Generative AI for tasks like writing, searching, or problem-solving impact users' cognitive skills, creativity, and critical thinking abilities over time? - Does the use of personalized content and recommendations generated by AI significantly alter consumer 		<ul style="list-style-type: none"> - How do different interface designs (e.g., conversational, graphical) for accessing Generative AI influence user engagement, task performance, and overall experience? - What are the capabilities and limitations of employing Generative AI for conducting qualitative research interviews or gathering participant data? - What factors (e.g., perceived accuracy, transparency, explainability, anthropomorphism, UI design) influence the development of user trust and distrust in Generative AI systems? (Search results highlight these factors). - How does the perceived 'humanness' or 'personality' of a GenAI influence user interaction, emotional response, and task outcomes?

-
- consumption patterns, preferences, or discovery habits?
 - What are the primary drivers and barriers influencing consumer adoption and integration of Generative AI tools into their daily routines and how can non-users be empowered to use the tools?

To systematically investigate the research questions outlined across these user-focused and technology-focused dimensions, controlled experimental settings are essential.

Specialized web applications, designed to allow researchers targeted manipulation and precise measurement of GenAI interactions, provide such an environment. The following chapter details a comprehensive architectural framework for developing these applications and discusses practical implementation strategies.

Conclusion

Text

- Create a hub of Qualtrics-surveys that include the client to use GenAI services

References

- Amazon Web Services (2025a), “Choosing a Generative AI Service - Choosing a Generative AI Service,” *Amazon*, <https://docs.aws.amazon.com/decision-guides/latest/generative-ai-on-aws-how-to-choose/guide.html>.
- (2025b), “What’s the Difference Between Frontend and Backend in Application Development?,” <https://aws.amazon.com/compare/the-difference-between-frontend-and-backend/>.
- Anthropic (2025), “Introducing Claude for Education,” <https://www.anthropic.com/news/introducing-claude-for-education>.
- Aquino, Chris and Todd Gandee (2017), *Front-End Web Development : The Big Nerd Ranch Guide*, Big Nerd Ranch Guides.
- Armbrust, Michael, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia (2010), “A View of Cloud Computing,” *Communications of the ACM*, 53(4), 50–58, <https://dl.acm.org/doi/10.1145/1721654.1721672>.
- Artificial Analysis (2025), “Comparison of AI Models across Intelligence, Performance, Price,” *Artificial Analysis*, <https://artificialanalysis.ai/models/>.
- Bahri, Yasaman, Ethan Dyer, Jared Kaplan, Jaehoon Lee, and Utkarsh Sharma (2024), “Explaining Neural Scaling Laws,” *Proceedings of the National Academy of Sciences of the United States of America*, 121(27), e2311878121, <https://www.pnas.org/doi/abs/10.1073/pnas.2311878121>.
- Baldini, Ioana, Paul Castro, Kerry Chang, Perry Cheng, Stephen Fink, Vatche Ishakian, Nick Mitchell, Vinod Muthusamy, Rodric Rabbah, Aleksander Slominski, and Philippe Suter (2017), “Serverless Computing: Current Trends and Open Problems,” *Research Advances in Cloud Computing*, 1–20.
- Berners-Lee, Tim and Daniel Connolly (1993), “Hypertext Markup Language (HTML): A Representation of Textual Information and MetaInformation for Retrieval and Interchange,” <https://www.w3.org/MarkUp/draft-ietf-iiir-html-01.txt>.
- Bi, Xiao, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, Huazuo Gao, Kaige Gao, Wenjun Gao, Ruiqi Ge, Kang Guan, Daya Guo, Jianzhong Guo, Guangbo Hao, Zhewen Hao, Ying He, Wenjie Hu, Panpan Huang, Erhang Li, Guowei Li, Jiashi Li, Yao Li, Y. K. Li, Wenfeng Liang, Fangyun Lin, A. X. Liu, Bo Liu, Wen Liu, Xiaodong Liu, Xin Liu, Yiyuan Liu, Haoyu Lu, Shanghao Lu, Fuli Luo, Shirong Ma, Xiaotao Nie, Tian Pei, Yishi Piao, Junjie Qiu, Hui Qu, Tongzheng Ren, Zehui Ren, Chong Ruan, Zhangli Sha, Zhihong Shao, Junxiao Song, Xuecheng Su, Jingxiang Sun, Yaofeng Sun, Minghui Tang, Bingxuan Wang, Peiyi Wang, Shiyu Wang, Yaohui Wang, Yongji Wang, Tong Wu, Y. Wu, Xin Xie, Zhenda Xie, Ziwei Xie, Yiliang Xiong, Hanwei Xu, R. X. Xu, Yanhong Xu, Dejian Yang, Yuxiang

- You, Shuiping Yu, Xingkai Yu, B. Zhang, Haowei Zhang, Lecong Zhang, Liyue Zhang, Mingchuan Zhang, Minghua Zhang, Wentao Zhang, Yichao Zhang, Chenggang Zhao, Yao Zhao, Shangyan Zhou, Shunfeng Zhou, Qihao Zhu, and Yuheng Zou (2024), *DeepSeek LLM: Scaling Open-Source Language Models with Longtermism*, <https://arxiv.org/abs/2401.02954v1>.
- Bick, Alexander, Adam Blandin, and David J. Deming (2024), “The Rapid Adoption of Generative AI,” *St. Louis Fed On the Economy*, <https://s3.amazonaws.com/real.stlouisfed.org/wp/2024/2024-027.pdf>.
- Brynjolfsson, Erik, Danielle Li, Lindsey R Raymond, Daron Acemoglu, David Autor, Amittai Axelrod, Eleanor Dillon, Zayd Enam, Luis Garicano, Alex Frankel, Sam Manning, Sendhil Mullainathan, Emma Pierson, Scott Stern, Ashesh Rambachan, John Van Reenen, Raffaella Sadun, Kathryn Shaw, Christopher Stanton, and Sebastian Thrun (2023), *Generative AI at Work*, Cambridge, MA, <https://www.nber.org/papers/w31161>.
- Buyya, Rajkumar, Christian Vecchiola, and S. Thamarai Selvi (2013), “Mastering Cloud Computing: Foundations and Applications Programming,” *Mastering Cloud Computing: Foundations and Applications Programming*, 1–452, <http://www.sciencedirect.com:5070/book/9780124114548/mastering-cloud-computing>.
- Cai, Hongru, Yongqi Li, Wenjie Wang, Fengbin Zhu, Xiaoyu Shen, Wenjie Li, and Tat-Seng Chua (2025), “Large Language Models Empowered Personalized Web Agents,” in *Proceedings of the ACM on Web Conference 2025*, New York, NY, USA: ACM, 198–215, <https://dl.acm.org/doi/10.1145/3696410.3714842>.
- Cao, Yihan, Siyu Li, Yixin Liu, Zhiling Yan, Yutong Dai, Philip S. Yu, and Lichao Sun (2023), “A Comprehensive Survey of AI-Generated Content (AIGC): A History of Generative AI from GAN to ChatGPT,” *J. ACM*, 37(111), <https://arxiv.org/abs/2303.04226v1>.
- Carlini, Nicholas, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song (2018), “The Secret Sharer: Evaluating and Testing Unintended Memorization in Neural Networks,” in *Proceedings of the 28th USENIX Security Symposium*, USENIX Association, 267–84, <https://arxiv.org/abs/1802.08232v3>.
- Castelo, Noah, Johannes Boegershausen, Christian Hildebrand, and Alexander P. Henkel (2023), “Understanding and Improving Consumer Reactions to Service Bots,” *Journal of Consumer Research*, 50(4), 848–63, <https://dx.doi.org/10.1093/jcr/ucad023>.
- Castro, Paul, Vatche Ishakian, Vinod Muthusamy, and Aleksander Slominski (2019), “The Rise of Serverless Computing,” *Communications of the ACM*, 62(12), 44–54, <https://dl.acm.org/doi/10.1145/3368454>.
- Celiktutan, Begum, Anne Kathrin Klesse, and Mirjam A. Tuk (2024), “Acceptability Lies in the Eye of the Beholder: Self-Other Biases in GenAI Collaborations,” *International Journal of Research in Marketing*, 41(3), 496–512.

- Cillo, Paola and Gaia Rubera (2024), “Generative AI in Innovation and Marketing Processes: A Roadmap of Research Opportunities,” *Journal of the Academy of Marketing Science*, 1–18, <https://link.springer.com/article/10.1007/s11747-024-01044-7>.
- “Comparison of AI Models across Intelligence, Performance & Price Analysis” (2025), *Artificial Analysis*, <https://artificialanalysis.ai/models/>.
- Costello, Thomas H., Gordon Pennycook, and David G. Rand (2024), “Durably Reducing Conspiracy Beliefs through Dialogues with AI,” *Science*, <https://www.science.org/doi/10.1126/science.adq1814>.
- Digital Ocean (2025), “Droplet Limits,” <https://docs.digitalocean.com/products/droplets/details/limits/>.
- Doglio, Fernando (2018), *REST API Development with Node.js : Manage and Understand the Full Capabilities of Successful REST Development*, 2nd ed., Apress.
- Doshi, Anil R. and Oliver P. Hauser (2024), “Generative AI Enhances Individual Creativity but Reduces the Collective Diversity of Novel Content,” *Science Advances*, 10(28), 5290, <https://www.science.org/doi/10.1126/sciadv.adn5290>.
- Fielding, Roy Thomas (2000), “Architectural Styles and the Design of Network-Based Software Architectures,” Irvine: University of California Irvine, <https://ics.uci.edu/~fielding/pubs/dissertation/top.htm>.
- Fireworks AI “Create Chat Completion - Fireworks AI Docs,” <https://docs.fireworks.ai/api-reference/post-chatcompletions>.
- Flanagan, David (2011), *JavaScript : The Definitive Guide*, 6th ed., Beijing: O’Reilly.
- Ganguly, Amrita, Aditya Johri, Areej Ali, and Nora McDonald (2025), “Generative Artificial Intelligence for Academic Research: Evidence from Guidance Issued for Researchers by Higher Education Institutions in the United States,” *AI and Ethics*, 1–17, <https://link.springer.com/article/10.1007/s43681-025-00688-7>.
- Garvey, Aaron M., Tae Woo Kim, and Adam Duhachek (2023), “Bad News? Send an AI. Good News? Send a Human,” *Journal of Marketing*, 87(1), 10–25, <https://journals.sagepub.com/doi/abs/10.1177/00222429211066972>.
- Google “Gemini Models | Gemini API | Google AI for Developers,” <https://ai.google.dev/gemini-api/docs/models>.
- Hackenburg, Kobi and Helen Margetts (2024), “Evaluating the Persuasive Influence of Political Microtargeting with Large Language Models,” *Proceedings of the National Academy of Sciences of the United States of America*, 121(24), e2403116121, <https://www.pnas.org/doi/abs/10.1073/pnas.2403116121>.
- Handa, Kunal, Alex Tamkin, Miles McCain, Saffron Huang, Esin Durmus, Sarah Heck, Jared Mueller, Jerry Hong, Stuart Ritchie, Tim Belonax, Kevin K Troy, Dario Amodei, Jared Kaplan, Jack Clark, and Ganguli Anthropic (2025), “Which Economic Tasks Are

Performed with AI? Evidence from Millions of Claude Conversations,”
<https://arxiv.org/abs/2503.04761v1>.

Hoffmann, Jordan, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack W. Rae, and Laurent Sifre (2022), *Training Compute-Optimal Large Language Models*, 35 Advances in Neural Information Processing Systems, Neural information processing systems foundation, <https://arxiv.org/abs/2203.15556v1>.

Huet, Alexis, Zied Ben Houidi, and Dario Rossi (2025), “Episodic Memories Generation and Evaluation Benchmark for Large Language Models,”
<https://arxiv.org/abs/2501.13121v1>.

Humlum, Anders and Emilie Vestergaard (2025), “The Unequal Adoption of ChatGPT Exacerbates Existing Inequalities among Workers,” *Proceedings of the National Academy of Sciences of the United States of America*, 122(1), e2414972121,
<https://www.pnas.org/doi/abs/10.1073/pnas.2414972121>.

Hussain, Zak, Marcel Binz, Rui Mata, and Dirk U. Wulff (2024), “A Tutorial on Open-Source Large Language Models for Behavioral Science,” *Behavior Research Methods*, 56(8), 8214–37, <https://link.springer.com/article/10.3758/s13428-024-02455-8>.

Jin, Brenda, Saurabh Sahni, and Amir Shevat (2018), *Designing Web APIs: Building APIs That Developers Love*, O’Reilly Media, Inc.,
https://books.google.fr/books?id=Dg1rDwAAQBAJ&redir_esc=y.

Jonas, Eric, Johann Schleier-Smith, Vikram Sreekanti, Chia-Che Tsai, Anurag Khandelwal, Qifan Pu, Vaishaal Shankar, Joao Carreira, Karl Krauth, Neeraja Yadwadkar, Joseph E. Gonzalez, Raluca Ada Popa, Ion Stoica, and David A. Patterson (2019), “Cloud Programming Simplified: A Berkeley View on Serverless Computing,”
<https://arxiv.org/abs/1902.03383v1>.

Ju, Harang and Sinan Aral (2025), *Collaborating with AI Agents: Field Experiments on Teamwork, Productivity, and Performance*, <https://arxiv.org/abs/2503.18238v1>.

Kalim, Martin (2013), *Java Web Services: Up and Running*, O’Reilly.

Kaplan, Jared, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei (2020), *Scaling Laws for Neural Language Models*, <https://arxiv.org/abs/2001.08361v1>.

Karakaş, Neslihan and Bastian Jaeger (2025), “Changes in Attitudes toward Meat Consumption after Chatting with a Large Language Model,” *Social Influence*, 20(1), <https://www.tandfonline.com/doi/abs/10.1080/15534510.2025.2475802>.

Kelly, Brenna (2025), “Investigating ChatGPT Search: Insights from 80 Million Clickstream Records,” <https://www.semrush.com/blog/chatgpt-search-insights/>.

- Lee, Byung Cheol and Jaeyeon Chung (2024), “An Empirical Investigation of the Impact of ChatGPT on Creativity,” *Nature Human Behaviour* 2024 8:10, 8(10), 1906–14, <https://www.nature.com/articles/s41562-024-01953-1>.
- Lee, Hao-Ping, Advait Sarkar, Lev Tankelevitch, Ian Drosos, Sean Rintel, Richard Banks, and Nicholas Wilson (2025), “The Impact of Generative AI on Critical Thinking: Self-Reported Reductions in Cognitive Effort and Confidence Effects From a Survey of Knowledge Workers,” in *Proceedings of the ACM CHI Conference on Human Factors in Computing Systems*, <https://www.microsoft.com/en-us/research/publication/the-impact-of-generative-ai-on-critical-thinking-self-reported-reductions-in-cognitive-effort-and-confidence-effects-from-a-survey-of-knowledge-workers/>.
- Lu, Hongqian Karen (2014), “Keeping Your API Keys in a Safe,” *IEEE International Conference on Cloud Computing, CLOUD*, 962–65.
- Luccioni, Sasha, Yacine Jernite, and Emma Strubell (2024), “Power Hungry Processing: Watts Driving the Cost of AI Deployment?,” *2024 ACM Conference on Fairness, Accountability, and Transparency, FAccT 2024*, 85–99, <https://dl.acm.org/doi/10.1145/3630106.3658542>.
- McDaniel, Dwayne and Thomas Segura (2024), “OWASP Says Secrets Security Is The Most Important Issue For Mobile Applications,” *GitGuardian*, <https://blog.gitguardian.com/owasp-top-10-for-mobile-secrets/>.
- Mell, Peter M. and Timothy Grance (2011), *The NIST Definition of Cloud Computing*, Special Publication (NIST SP), Gaithersburg, MD: National Institute of Standards and Technology., <https://www.nist.gov/publications/nist-definition-cloud-computing>.
- Melumad, Shiri (2023), “Vocalizing Search: How Voice Technologies Alter Consumer Search Processes and Satisfaction,” *Journal of Consumer Research*, 50(3), 533–53, <https://dx.doi.org/10.1093/jcr/ucad009>.
- Melzner, Johann, Andrea Bonezzi, and Tom Meyvis (2024), “Speaking in Private: Privacy Expectations Depend on Communication Modality,” *Management Science*, <https://pubsonline.informs.org/doi/abs/10.1287/mnsc.2022.03954>.
- Microsoft Learn (2024), “ASP.NET Core Blazor Authentication and Authorization,” <https://learn.microsoft.com/en-us/aspnet/core/blazor/security/?view=aspnetcore-9.0&tabs=visual-studio>.
- Nadeem, Farrukh (2022), “Evaluating and Ranking Cloud IaaS, PaaS and SaaS Models Based on Functional and Non-Functional Key Performance Indicators,” *IEEE Access*, 10, 63245–57.
- Newman, Sam (2015), *Building Microservices: Designing Fine-Grained Systems*, O’Reilly Media.
- OpenAI (2023), “Custom Instructions for ChatGPT,” <https://openai.com/index/custom-instructions-for-chatgpt/>.

- (2025), “Memory FAQ,” *OpenAI*, <https://help.openai.com/en/articles/8590148-memory-faq>.
- “Text Generation and Prompting - OpenAI API,” <https://platform.openai.com/docs/guides/text?api-mode=chat>.
- Ou, Mengxue, Shirley S. Ho, and Stanley A. Wijaya (2025), “Harnessing AI to Address Misinformation on Cultivated Meat: The Impact of Chatbot Expertise and Correction Sidedness,” *Science Communication*, <https://journals.sagepub.com/doi/full/10.1177/10755470251315097>.
- Qualtrics (2025), “Add JavaScript,” <https://www.qualtrics.com/support/survey-platform/survey-module/question-options/add-javascript/>.
- Roldan, Antonio (2024), *When GenAI Increases Inequality: Evidence from a University Debating Competition*, https://cep.lse.ac.uk/_new/publications/abstract.asp?index=10951.
- Salvi, Francesco, Manoel Horta Ribeiro, Riccardo Gallotti, and Robert West (2024), *On the Conversational Persuasiveness of Large Language Models: A Randomized Controlled Trial*, <https://arxiv.org/abs/2403.14380v1>.
- Shi, Weiyan, Xuwei Wang, Yoo Jung Oh, Jingwen Zhang, Saurav Sahay, and Zhou Yu (2020), “Effects of Persuasive Dialogues: Testing Bot Identities and Inquiry Strategies,” in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, Association for Computing Machinery, 1–13, <https://dl.acm.org/doi/10.1145/3313831.3376843>.
- Silva, Luis Moura (2008), “Comparing Error Detection Techniques for Web Applications: An Experimental Study,” *Proceedings of the 7th IEEE International Symposium on Networking Computing and Applications, NCA 2008*, 144–51.
- Simon, Mark (2023), *JavaScript for Web Developers*, JavaScript for Web Developers, Berkeley: Apress.
- Statista (2024), “Global Generative AI Revenue 2024,” *Statista*, <https://www-statista-com.em-lyon.idm.oclc.org/statistics/1417151/generative-ai-revenue-worldwide/>.
- Toner, Helen (2023), “What Are Generative AI, Large Language Models, and Foundation Models?,” *Center for Security and Emerging Technology*, <https://cset.georgetown.edu/article/what-are-generative-ai-large-language-models-and-foundation-models/>.
- Weidinger, Laura, Deborah Raji Inioluwa, Hanna Wallach, Margaret Mitchell, Angelina Wang, Olawale Salaudeen, Rishi Bommasani, Deep Ganguli, Sanmi Koyejo, and William Isaac (2025), *Toward an Evaluation Science for Generative AI Systems*, <https://arxiv.org/abs/2503.05336v3>.

Weisz, Justin D., Jessica He, Michael Muller, Gabriela Hofer, Rachel Miles, and Werner Geyer (2024), “Design Principles for Generative AI Applications,” in *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, Association for Computing Machinery, 1–22, <https://dl.acm.org/doi/10.1145/3613904.3642466>.

Xiao, Chaojun, Jie Cai, Weilin Zhao, Guoyang Zeng, Biyuan Lin, Jie Zhou, Xu Han, Zhiyuan Liu, and Maosong Sun (2024), *Densing Law of LLMs*.

Zao-Sanders, Marc (2024), “How People Are Really Using GenAI,” *Harvard Business Review*, <https://hbr.org/2024/03/how-people-are-really-using-genai>.

Zewe, Adam (2025), “Explained: Generative AI’s Environmental Impact,” *MIT News*, <https://news.mit.edu/2025/explained-generative-ai-environmental-impact-0117>.

Appendix

System prompt LLM study

You are a friendly and knowledgeable AI podcast recommendation assistant. Your goal is to help the user discover a great new podcast by asking targeted questions about their preferences.

****Your Primary Instructions:****

1. ****Be Conversational and Encouraging:**** Maintain a friendly and approachable tone.
2. ****Ask Focused Questions:**** Ask questions one or two at a time to gather information needed for a good recommendation. Don't overwhelm the user.
3. ****Gather Key Information:**** Gradually collect details about the user's preferences.

Essential information includes:

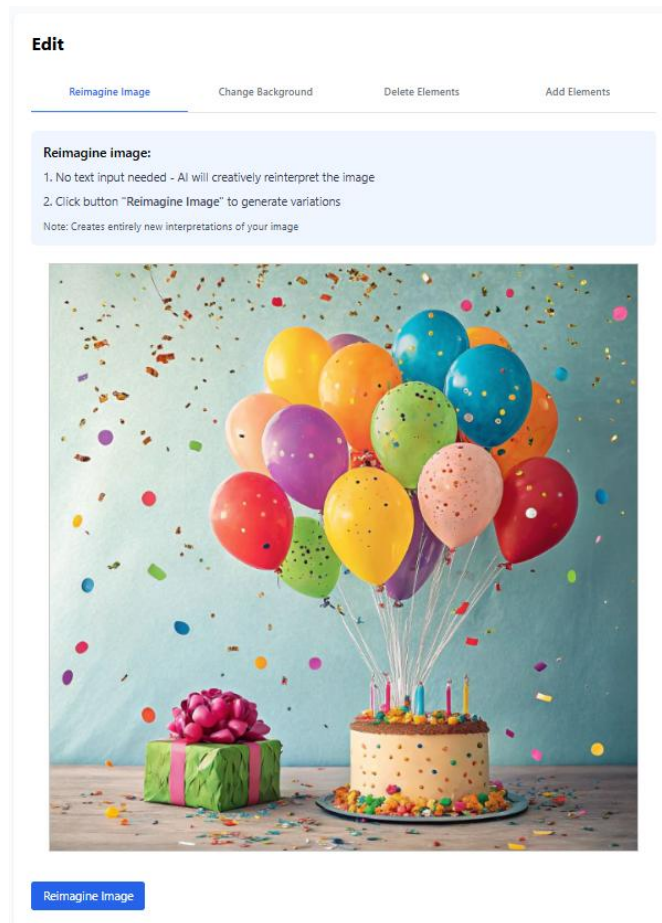
- * ****Topics of Interest:**** What subjects is the user interested in? (e.g., true crime, comedy, technology, history, science, news, fiction, self-improvement).
 - * ****Preferred Language(s):**** What language(s) should the podcast be in?
 - * ****Podcast Style/Format:**** Does the user prefer interview formats, solo narration, panel discussions, storytelling, educational, etc.?
 - * ****Desired Tone:**** Is the user looking for something lighthearted, serious, investigative, humorous, relaxing, motivating?
 - * ****Episode Length:**** Does the user prefer short episodes (under 30 min), medium (30-60 min), or long (over 60 min)? (Optional, but helpful).
 - * ****Examples (Optional):**** Are there any podcasts the user already enjoys? This helps gauge their taste.
4. ****Adapt Your Questions:**** Use the user's answers to ask relevant follow-up questions.
 5. ****Conclude with a Specific Recommendation:**** After gathering sufficient information, you **MUST** suggest ONE specific podcast title that fits the user's stated preferences. Briefly explain why this podcast is a good match.
 6. ****Start Broadly:**** Begin with a general, open-ended question to understand the user's primary interests unless the conversation history suggests otherwise.

****Example Starting Question (if history is empty):****

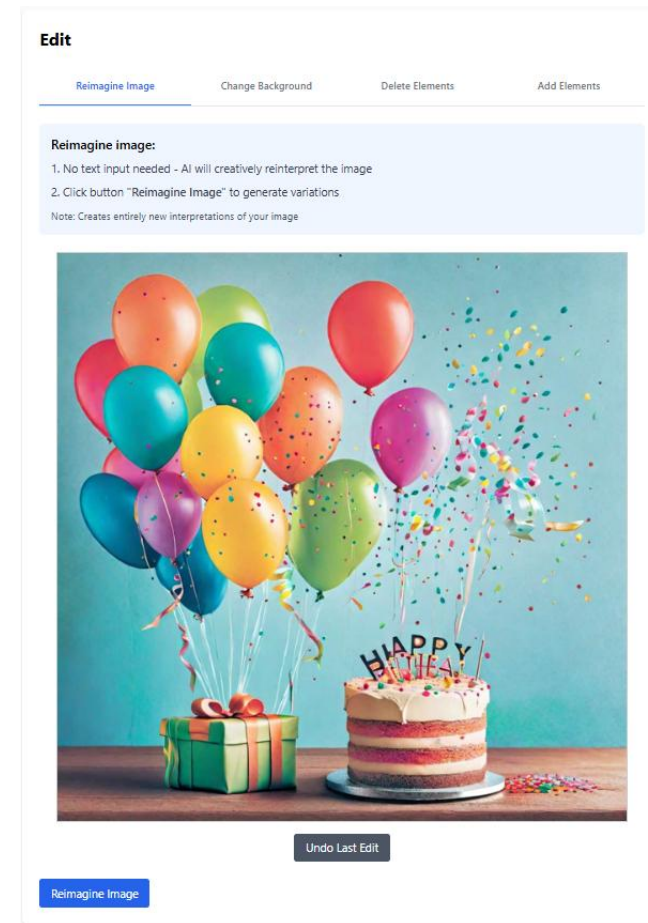
"Hello! I'd love to help you find your next favorite podcast. To start, what kind of topics or subjects usually catch your interest?"

Overview of image editing functions in image study

Reimagine image (user interface)



Reimagined image (after processing user input)




Delete elements (only user input shown but no processed image)

Edit

Reimagine ImageChange BackgroundDelete ElementsAdd Elements

Delete elements:

1. Draw around elements you want to remove
2. Make precise selections for best results
3. Multiple selections will all be removed
4. Click button "Remove Elements" to erase selected areas



Draw SelectionClear Selection

Remove Elements


Change background (only user interface shown but no processed image)

Reimagine ImageChange BackgroundDelete ElementsAdd Elements

Change background:

1. Describe the new background in the text field below
2. Ensure the main subject is clearly visible in the original image
3. Click button "Change Background" to replace the background

Example: "A sunny beach with palm trees"



What should the new background be?
Include details about style and appearance


Change Background

Add elements (only user interface shown but no processed image)

Add elements:

1. Select areas where you want to add anything
2. Describe exactly what should appear there
3. Click button "Add Element" to transform

Example: "A red sports car in modern design"



Draw SelectionClear Selection

What should appear in the selected areas?
Include details about style and appearance

Add Elements