Generative UI: LLMs are Effective UI Generators

Yaniv Leviathan **Dani Valevski** **Matan Kalman** **Danny Lumen**
Eyal Segalis **Eyal Molad** **Shlomi Pasternak** **Vishnu Natchu** **Valerie Nygaard**
Srinivasan (Cheenu) Venkatachary **James Manyika** **Yossi Matias**

Google Research

{leviathan, daniv, matank, dwasserman, eyalis, moladeyal,
spasternak, vnatchu, vnygaard, vsri, jmanyika, yossi}@google.com

Abstract

AI models excel at creating content, but typically render it with static, predefined interfaces. Specifically, the output of LLMs is often a markdown "wall of text". *Generative UI* is a long standing promise, where the model generates not just the content, but the interface itself. Until now, Generative UI was not possible in a robust fashion. We demonstrate that when properly prompted and equipped with the right set of tools, a modern LLM can robustly produce high quality custom UIs for virtually any prompt. When ignoring generation speed, results generated by our implementation are overwhelmingly preferred by humans over the standard LLM markdown output. In fact, while the results generated by our implementation are worse than those crafted by human experts, they are at least comparable in 44% of cases. We show that this ability for robust Generative UI is emergent, with substantial improvements from previous models. We also create and release *PAGEN*, a novel dataset of expert-crafted results to aid in evaluating Generative UI implementations, as well as the results of our system for future comparisons. Interactive examples can be seen at generativeui.github.io.

1 Introduction

AI models today generate *content*: text, code, images, videos, etc. However, the results of these powerful tools are often presented using hard-coded and pre-designed user interfaces. *Generative UI* is a new modality where the AI model generates not only content, but the entire user experience. This results in custom interactive experiences, including rich formatting, images, maps, audio and even simulations and games, in response to any prompt (instead of the widely adopted "walls-of-text").

An instant AI team for each prompt. Today, rich visual interfaces exist for common user journeys. Specifically, teams composed of product managers, UX designers, and engineers work for extended periods of time to build amazing rich experiences for *broad prompt categories*, shared by many users. Generative UI enables us to spin up an instant (AI-based) product management, UX design, and engineering teams, to build an interactive experience, over the course of a minute, for a specific prompt. While not as competent as human experts, Generative UI enables custom experiences for *any prompt*.

At present, the prevalent UI for interacting with LLMs is a markdown-based chat interface. Specifically, the model outputs markdown (that can include heading, emojis, tables, code-blocks, etc.). These are significantly easier for humans to consume than raw text, yet results created by our Generative UI implementation are overwhelmingly preferred over both (see Table 2). To evaluate our implementation of Generative UI we use human rater preference compared to a set of baselines. We

Preprint. Under review.

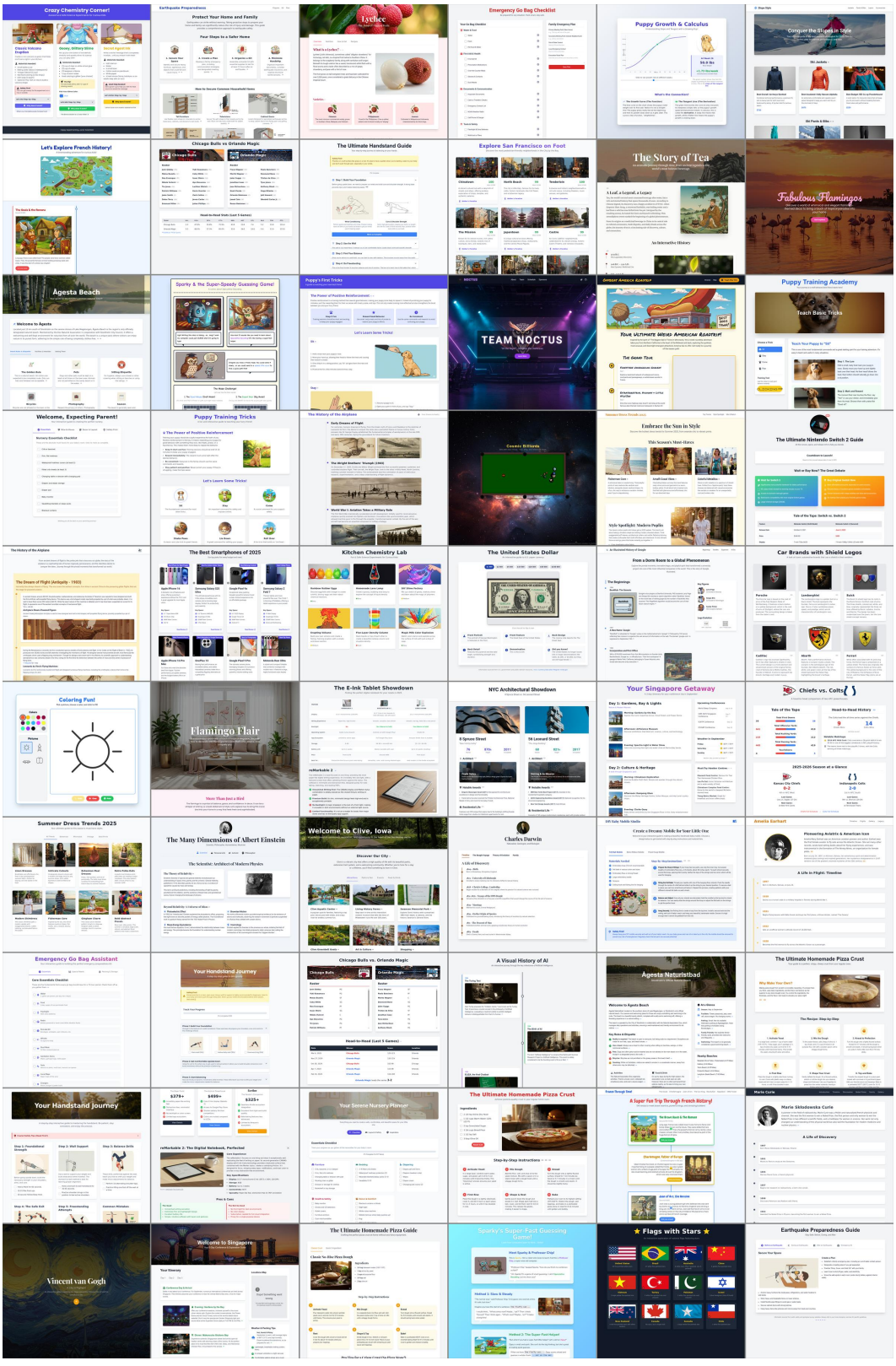

Figure 1: Results from our implementation (see generativeui.github.io).

collect and make available *PAGEN* (see Section 4), a dataset of pages made by human experts for custom prompts. While the expert-made pages are broadly preferred to those from our system, we show that for the first time we get comparable results on a large fraction of prompts. See Section 3 for details, Appendix A.1 for screenshots, and generativeui.github.io for interactive results from our system.

2 Method

Our Generative UI implementation outputs a single fully-generated web page and a set of accompanying assets, such as images. The page is rendered as-is on the user's browser. See Figure 2 for a high level overview of the system.

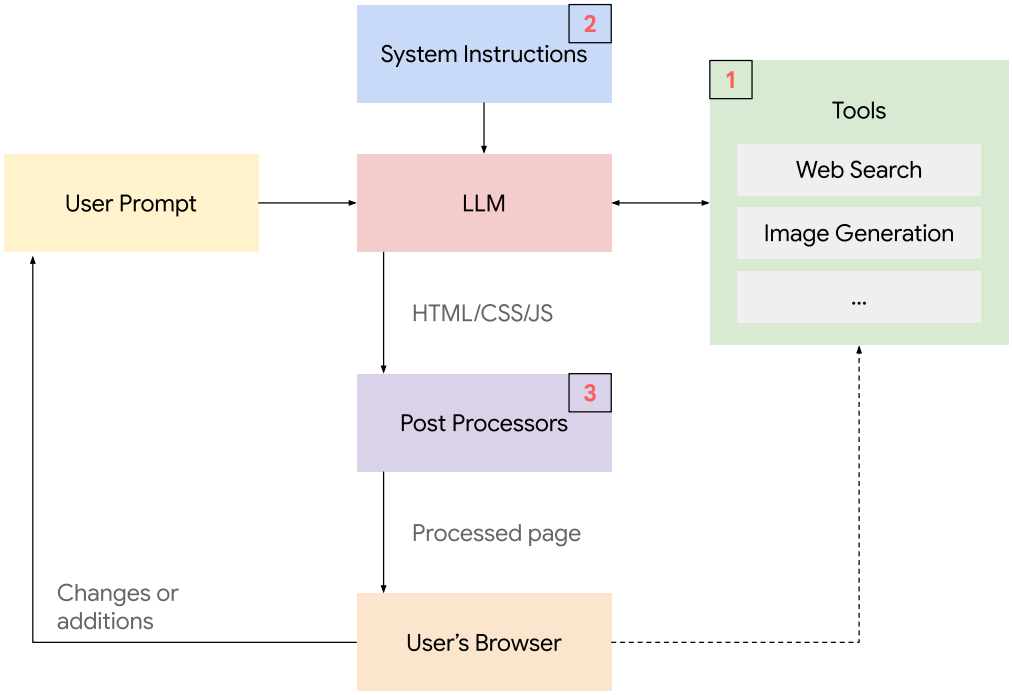

Figure 2: A high level system overview.

As depicted in Figure 2, we employ 3 main components:

1. A server exposes several endpoints enabling access to key tools, such as image generation and search. The results can be made accessible to the model (increasing quality) or sent directly to the user's browser (increasing efficiency).

2. Carefully crafted system instructions. These in turn include: (1) the goal (2) planning and thinking guidelines, (3) examples, and (4) a large set of technical instructions including formatting guidelines, tool endpoints manual, and tips for avoiding common errors. These contribute to the quality of the generated results (see Appendix A.5 for an illustrative prompt from an early research prototype).

3. A set of post-processors. These lightweight components address a set of remaining common issues. Additional post processors deal with error reporting and page analysis. See Appendix A.6.

2.1 Consistent Styling

If desired, our setup allows producing results using a specific style and increased visual consistency across generations. This is done via small changes to the system instructions. Specifically, we experimented with replacing the short "Style" section in our prompt with more detailed variants (which we call "Classic" and "Wizard Green"), specifying colors, fonts, etc. We observe that indeed the generated results follow these styles. Interestingly, the model automatically adapts all elements, including e.g. the generated images and icons to the desired styles. See Figures 3 and 4.

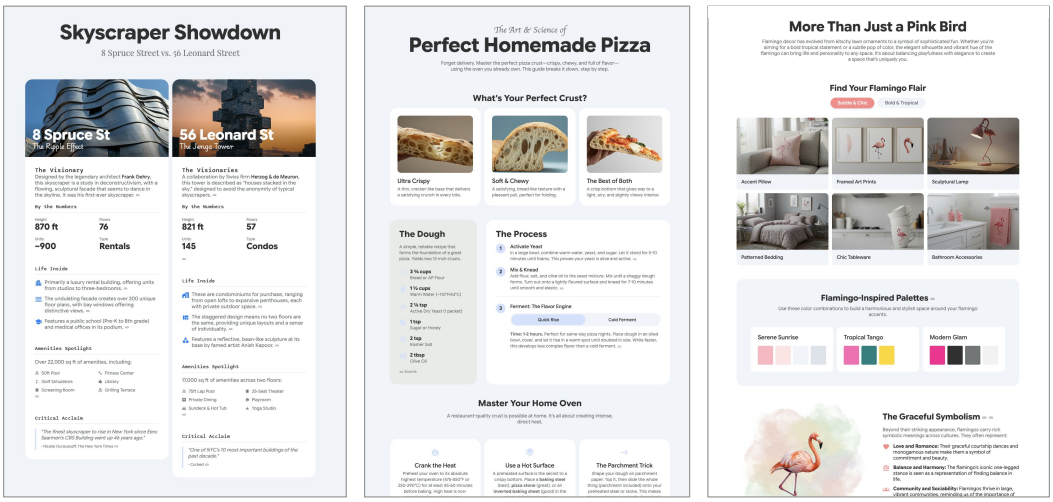

Figure 3: Screenshots of Generative UI results with "Classic" styling.

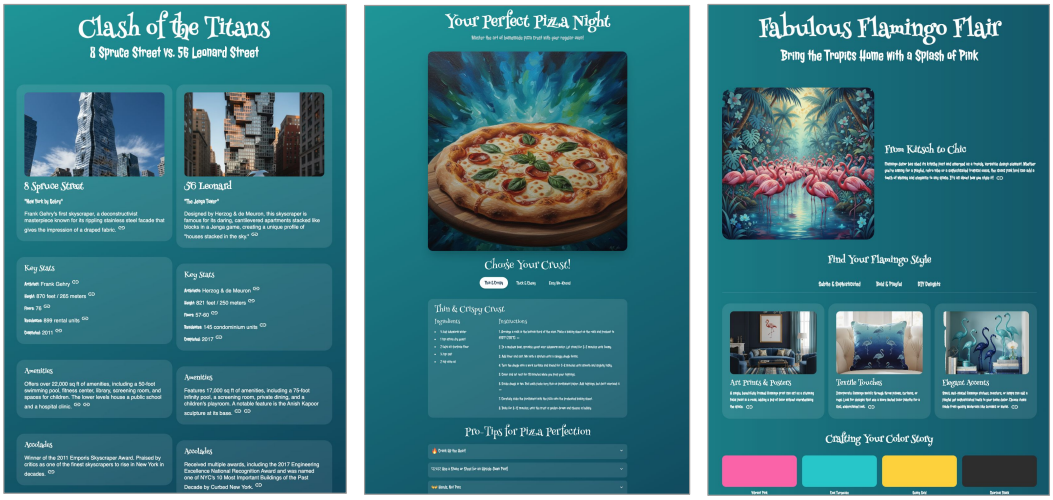

Figure 4: Screenshots of Generative UI results with "Wizard Green" styling.

3 Results

We evaluate user preference across several different result formats: a custom website crafted for the prompt by a human expert (see Section 4), the top Google Search result for the query, text (LLM output without markdown), standard LLM output (in markdown format), and our Generative UI implementation. We randomly sampled 100 prompts from LMArena [Chiang et al., 2024] (and excluded 8 queries, see Appendix A.4) and collected pairwise preferences from human raters, sending each result to 2 raters. Generation time is not a factor in the evaluation and we show raters pre-cached results. We ask the human raters to rate on a 3 point scale: Left Preferred, Neutral, Right Preferred.

In addition to the LMArena prompts, we also created a custom prompt set composed of information seeking prompts (see Appendix A.3). We evaluated on both sets using the same methodology. See Appendix A.1 for selected example generations from our Generative UI implementation.

Tables 1 and 2 show the resulting ELO scores and side-by-side user preference for each of the UI modalities for the prompts from LMArena. Generative UI obtains an ELO score of 1710.7, indicating a strong user preference over all other formats, except human experts. Notably, when compared to Markdown UI - the next best method, Generative UI is preferred 82.8% of the time. See Appendix Tables 6 and 7 for the results when evaluating on the Information Seeking prompt set (92.6% preference for our implementation).

Table 1: ELO scores for user preference (LMArena).

Format	ELO Score
Website (human expert)	**1756.0**
Generative UI	**1710.7**
Generative Markdown	1459.6
Website (top search result)	1355.1
Generative Text	1218.6

Table 2: Pairwise user preferences wins (LMArena). Generative UI strongly preferred except vs. human experts.

Method	Custom Website (human expert)	Generative UI	Markdown	Website (top result)	Text
Website (experts)	-	56.0%	**84.4%**	**89.1%**	**94.0%**
Generative UI	43.0%	-	**82.8%**	**90%**	**97.0%**
Markdown	15.6%	13.9%	-	44.4%	**81.1%**
Website (search)	4.9%	6.7%	52.2%	-	58.9%
Text	2.7%	3.0%	1.1%	38.3%	-

3.1 Emergent Capability

We ablate the importance of the backbone model and show that Generative UI is an emergent capability with newer models. In Tables 3 and Table 4 we see strong user preference and less errors for results with the new Gemini models.

Table 3: Model Performance Comparison (LMArena)

Backbone Model	Elo Score	Output Errors
Gemini 3	**1706.7**	0%
Gemini 2.5 Pro	1653.6	0%
Gemini 2.5 Flash	1623.9	0%
Gemini 2.0 Flash	1332.9	29%
Gemini 2.0 Flash-Lite	1183.0	60%

Table 4: Model Performance Comparison (Info-Seeking)

Backbone Model	Elo Score	Output Errors
Gemini 3	**1739.31**	0%
Gemini 2.5 Pro	1578.53	0%
Gemini 2.5 Flash	1577.74	0%
Gemini 2.0 Flash	1361.75	0%
Gemini 2.0 Flash-Lite	1242.67	1%

3.2 Prompt Ablations

We analyzed the impact of our prompting strategy. First we compared to a minimal prompt that only instructs the model how to use image search and image generation, as well as how to output a valid HTML. Raters preferred the UI generated with the full prompt in significantly more cases. Next we took out specific parts of the full prompt including the core philosophy and the corresponding examples. See Table 5 and Table 8 in the Appendix for details. Interestingly, the model is strong enough to show reasonable performance even with a minimal prompt.

Table 5: Effect of Prompting Strategy (LMArena)

Prompt Ablation	ELO Score
Full Prompt	1553.23
Minimal Prompt	1496.00
No Philosophy	1450.77

4 The PAGEN Dataset

To facilitate a clear and consistent evaluation of our Generative UI implementation, we compare its results to expert-human-made websites. To that end, we constructed a human-expert-made dataset of websites for a sample of prompts (using the LMArena and Info-Seeking prompt sets used for evaluation in this paper) . We call this dataset *PAGEN* and make it available publicly, in hopes of encouraging consistent comparisons with future work.

We considered several methodologies for collecting these human made websites, including utilizing existing public websites, using a pre-existing dataset, and engaging a specific provider to develop all of the necessary websites. Ultimately, we opted to construct our own dataset by contracting highly rated independent web developers sourced online. This decision was driven by a desire to create a clear pairing of a specific user prompt and the resulting website, maintain uniformity in time and investment across websites, ensure clear and consistent guidelines for all use cases (e.g. encourage interactivity and high quality visuals), ensure that the user experience is prioritized without any foreign considerations (such as SEO optimizations), ensuring no copyrighted content was used, and ensuring the consistency of the tools used (e.g. we encouraged using AI tools), and the diversity and quality of the contractors. See more details in Appendix A.4.

5 Related Work

The concept of automatically generating user interfaces from high-level descriptions has been a long-standing ambition in Human-Computer Interaction (HCI) and software engineering. Our work builds upon several key areas of research, including natural language interfaces, code generation by large language models (LLMs), and evaluation methodologies for generative systems.

UI Generation from Natural Language Early efforts in this domain often relied on structured inputs or constrained languages to generate interfaces for specific platforms [Puerta et al., 1994, Landay and Myers, 1995]. With the rise of deep learning, approaches evolved to translate visual inputs, such as hand-drawn mockups or screenshots, directly into code [Beltramelli, 2017, Gui et al., 2025]. The recent proliferation of powerful LLMs has enabled the generation of UI code directly from unstructured natural language prompts. Our approach differs by tasking the LLM to generate entire, interactive, and data-driven web applications from a single prompt, effectively acting as an autonomous web developer.

Large Language Models for Code Generation The capabilities of our system are fundamentally enabled by the advancements in code generation by LLMs. This field gained prominence with models like OpenAI's Codex [Chen et al., 2021], which demonstrated a strong ability to translate natural language into functional code across various languages. Subsequent research has produced a host of powerful code-generating models, such as AlphaCode [Li et al., 2022] and Code Llama [Rozière et al., 2024], that are trained on vast datasets of public code. While these models are often used as

assistants for developers (e.g., GitHub Copilot), our work leverages this underlying capability for a different purpose: the autonomous end-to-end generation of a complete user-facing product, not just a code snippet. As we demonstrate, this ability to architect and implement a full application appears to be an emergent property of the most recent state-of-the-art models.

Interaction Paradigms for AI The standard user interface for interacting with LLMs is a chat-based format where the model's output is rendered as markdown. While an improvement over plain text, this modality is inherently static. Some systems have explored a middle ground, which we term "Templated UI," where an LLM can invoke and populate predefined, interactive widgets from a fixed library to enrich its responses [Pinsky, 2023]. Our work represents a paradigm shift away from both static markdown and constrained templates. By allowing the model to generate the UI itself, we unlock the potential for bespoke, dynamic, and highly contextual experiences, such as games, simulators, and custom data visualizations, that are tailored to the specific needs of each prompt.

6 Discussion

We presented a novel implementation of Generative UI, where the model can produce a custom visual interactive interface for any prompt. We show that when ignoring generation speed, our results are overwhelmingly preferred by users over the standard markdown UI (in 83% of evaluated cases, see Table 2). We further show that Generative UI is an emergent capability of the newest and most capable models. As shown in Tables 3 and 4, the use of our newest models results in a significant increase in user preference and a significant reduction in generation errors, vs. previous models. Our implementation relies on a combination of exposing a set of tools in an easy-to-use fashion, detailed system instructions (see Appendix A.5), and a series of post-processors to correct common issues.

PAGEN We created the *PAGEN* dataset - a curated dataset of expert built web sites for LLM prompts (see Section 4). While the pages created by the expert humans are better than those created by our system, we show that our Generative UI implementation can at least match its quality in 44% of cases. We are making *PAGEN* available publicly to enable easier evaluation by future research.

Limitations and Future Directions One primary limitation and an important area for future research is the slow generation speed, which can often take a minute or two. Streaming the generated results allows the users to start interacting with a partially rendered page, reducing this number by about a half. Optimizing the use of techniques such as speculative decoding [Leviathan et al., 2022] could result in further improvements. A second important limitation of Generative UI is that errors (Javascript errors, CSS errors, HTML errors, etc.) can occasionally occur.

A First Step Towards a New Paradigm LLMs transformed the world's finite collections of texts to an infinite collection, where an ephemeral text is created on the spot for any need. This turned out to be very useful. It is early days for Generative UI, and important limitations exist. Yet, we are excited about a future where users don't have to pick from a finite library of applications or visual pages, but instead, they have access to an infinite catalog, where the right ephemeral interface is generated on the spot tailored for their need.

Acknowledgements

This work would not have been possible without the valuable contributions, insightful suggestions, support, feedback and encouragement from Yoav Tzur, Zak Tsai, Hen Fitoussi, Amir Zait, Oren Litvin, Christopher Haire, Liat Ben-Rafael, Ronit Levavi Morad, Kristen Chui, William Li, Ivan Kelber, Chloe Jia, Ryan Allen, Maryam Sanglaji, Tanya Sinha, Josh Woodward, Jeff Dean, and the Theta Labs, Google Research, Google Search, and Gemini teams, and, as always, our families.

References

Tony Beltramelli. pix2code: Generating code from a graphical user interface screenshot, 2017. URL https://arxiv.org/abs/1705.07962.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Hao Zhang, Banghua Zhu, Michael Jordan, Joseph E. Gonzalez, and Ion Stoica. Chatbot arena: An open platform for evaluating llms by human preference, 2024. URL https://arxiv.org/abs/2403.04132.

Yi Gui, Yao Wan, Zhen Li, Zhongyi Zhang, Dongping Chen, Hongyu Zhang, Yi Su, Bohua Chen, Xing Zhou, Wenbin Jiang, and Xiangliang Zhang. Uicopilot: Automating ui synthesis via hierarchical code generation from webpage designs. In *Proceedings of the ACM on Web Conference 2025*, WWW '25, page 1846–1855. ACM, April 2025. doi: 10.1145/3696410.3714891. URL http://dx.doi.org/10.1145/3696410.3714891.

James A. Landay and Brad A. Myers. Interactive sketching for the early stages of user interface design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '95, page 43–50, USA, 1995. ACM Press/Addison-Wesley Publishing Co. ISBN 0201847051. doi: 10.1145/223904.223910. URL https://doi.org/10.1145/223904.223910.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding, 2022. URL https://arxiv.org/abs/2211.17192.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, Thomas Hubert, Peter Choy, Cyprien de Masson d'Autume, Igor Babuschkin, Xinyun Chen, Po-Sen Huang, Johannes Welbl, Sven Gowal, Alexey Cherepanov, James Molloy, Daniel J. Mankowitz, Esme Sutherland Robson, Pushmeet Kohli, Nando de Freitas, Koray Kavukcuoglu, and Oriol Vinyals. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, December 2022. ISSN 1095-9203. doi: 10.1126/science.abq1158. URL http://dx.doi.org/10.1126/science.abq1158.

Yury Pinsky. Bard can now connect to your Google apps and services. Google Blog, September 2023. URL https://blog.google/products/gemini/google-bard-new-features-update-sept-2023/.

Angel R. Puerta, Henrik Eriksson, John H. Gennari, and Mark A. Musen. Model-based automated generation of user interfaces. In *Proceedings of the Twelfth AAAI National Conference on Artificial Intelligence*, AAAI'94, page 471–477. AAAI Press, 1994.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. Code llama: Open foundation models for code, 2024. URL https://arxiv.org/abs/2308.12950.

Upwork Global Inc. Upwork. https://www.upwork.com.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Tianle Li, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zhuohan Li, Zi Lin, Eric P. Xing, Joseph E. Gonzalez, Ion Stoica, and Hao Zhang. Lmsys-chat-1m: A large-scale real-world llm conversation dataset, 2024. URL https://arxiv.org/abs/2309.11998.

A Appendix

A.1 Selected Examples

All examples presented here can be viewed interactively at generativeui.github.io.

A.1.1 Fractal Explorer

User prompt: *[Explain fractals - go really in depth - i want to learn everything about it in detail]*

The generative UI system produced an immersive, interactive webpage titled "Fractal Explorer" that serves as a deep dive into the mathematics and visual beauty of infinite complexity. The page guides users through the fundamental concepts of self-similarity and a historical timeline of fractal discovery—from Weierstrass's "monsters" to Benoit Mandelbrot's modern definitions. Central to the experience are robust interactive tools, including a "Dimension Calculator" that visually demonstrates the Hausdorff dimension formula, a dual-canvas explorer that links the Mandelbrot set to corresponding Julia sets in real-time via mouse movement, and dynamic sliders that allow users to build geometric fractals like the Koch Snowflake and Sierpinski Triangle iteration by iteration. The page concludes with a generative simulation of the "Chaos Game" to organically grow a Barnsley Fern and a section detailing practical applications in technology, biology, and computer graphics.

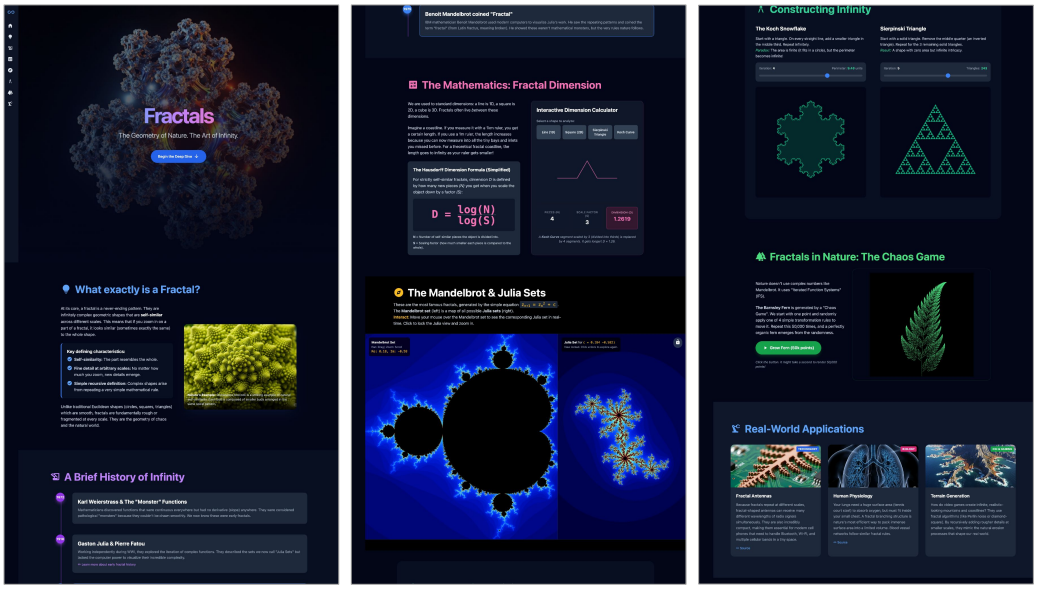

Figure 5: "Explain fractals" generated web-app.

A.1.2 History of Time Keeping Devices

User prompt: *["History of time keeping devices"]*

The generative UI system produced a visually immersive, dark-themed webpage titled "Chronos: A History of Timekeeping" that traces the evolution of measuring time. The layout features a vertical, scroll-animated timeline that guides users through six distinct eras, starting with ancient methods like Egyptian obelisks and water clocks, progressing through the mechanical and pendulum revolutions initiated by innovators like Christiaan Huygens, and concluding with the precision of quartz and atomic clocks. Each timeline entry pairs descriptive historical context with thematic generated imagery and specific "Key Insight" or "Engineering Breakthrough" callout boxes to highlight technological leaps. The design utilizes a responsive grid system with alternating text and image placements, enhanced by fade-in scroll effects to create a narrative flow from the "Mechanical Dawn" to "Atomic Perfection."

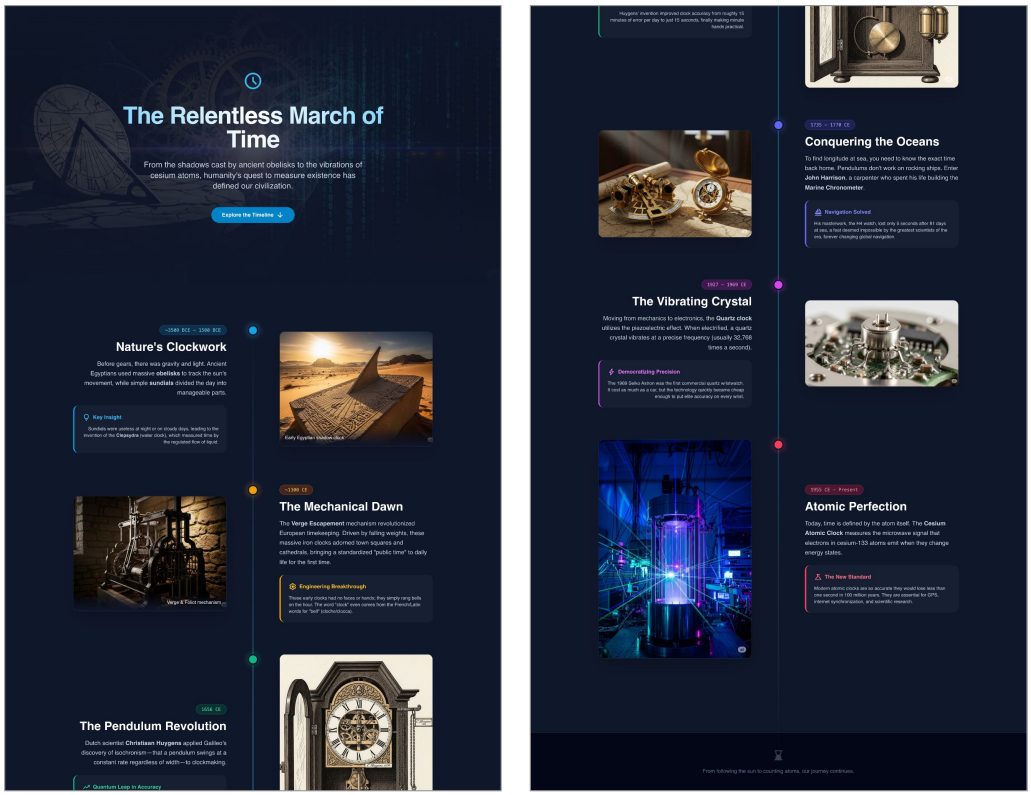

Figure 6: "History of Time Keeping Devices" generated web-app.

A.1.3 Memory Game

User prompt: *[Create a match up memory game with beautiful large cards showing the people from the images making funny expressions and wearing funny props]*

The generative UI system produced an interactive "Funny Faces Memory Match" game designed to test users' recall through a responsive grid of flip-cards. The interface utilizes 3D transform effects and Tailwind CSS to present a seamless gameplay experience where players uncover pairs of generated portraits featuring subjects with humorous accessories, such as oversized clown glasses, pirate hats, and propeller beanies. Real-time state tracking monitors the player's progress by updating move counts and successful matches, culminating in a victory modal that summarizes performance and offers a "Play Again" option upon clearing the board.

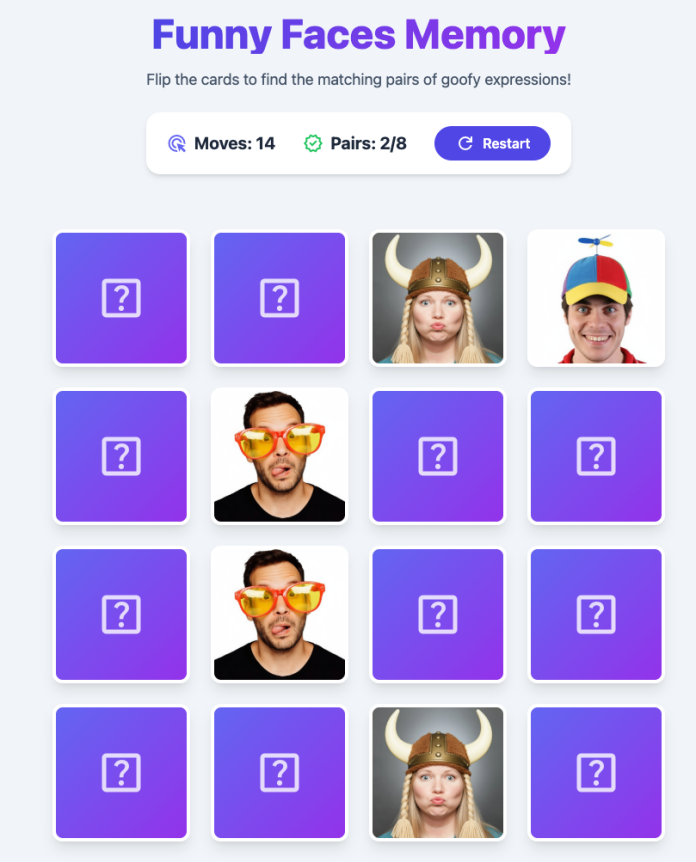

Figure 7: "Memory Game" generated web-app.

A.1.4 Basketball Math

User prompt: *[Teach my 5 year old addition, subtraction, multiplication and binary representation with cute creatures that play basketball. He loves basketball].*

The generative UI system produced an interactive educational application titled "Little Ballers Math Academy," designed to teach fundamental arithmetic and binary concepts through a playful, basketball-themed interface. The application features a central "Training Camp" menu where users can select from four distinct game modes: addition ("Passing Practice"), subtraction ("Taking Shots"), multiplication ("Team Drills"), and a unique binary counting challenge ("Alien Scoreboard") that utilizes interactive toggle switches to represent values. Each mode employs dynamic visual aids—such as counting basketballs held by animated monsters and robots—to assist with problem-solving, while providing immediate feedback through celebratory confetti effects and a real-time score tracking system.

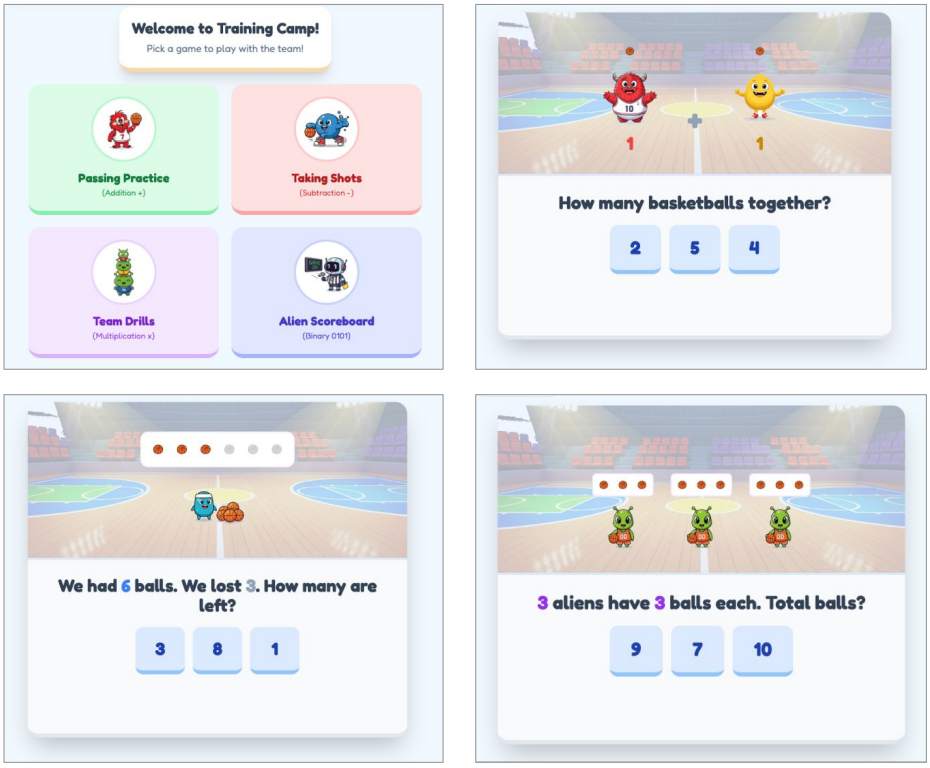

Figure 8: "Basketball Math" generated web-app.

A.2 Additional Results

We provide results on the Info-Seeking prompt set, comparing the different modalities. Table 6 shows the competitive ELO scores, and Table 7 shows the user preferences matrix. We observe the same trends as we saw on the LMArena prompt set, with a strong preference to Generative UI and human experts. Interestingly, top-search websites score higher on this set, probably due to the different distribution of prompts.

Table 6: ELO scores for user preference (Info-Seeking).

Format	ELO Score
Website (human expert)	**1776.38**
Generative UI	**1708.97**
Website (top search result)	1468.61
Generative Markdown	1354.80
Generative Text	1191.23

Table 7: Pairwise user preferences wins (Info-Seeking). Generative UI strongly preferred except vs. human experts.

Method	Custom Website (human expert)	Generative UI	Markdown	Website (top result)	Text
Website (experts)	-	64.2%	**97.1%**	**84.3%**	**84.8%**
Generative UI	31.9%	-	**92.6%**	**84.3%**	**91.2%**
Markdown	2.9%	7.4%	-	22.5%	86.8%
Website (search)	12.7%	8.3%	77.0%	-	73.5%
Text	11.3%	5.4%	8.8%	26.5%	-

We also provide in Table 8 the prompt ablation results on the Information Seeking prompt set.

Table 8: Effect of Prompting Strategy (Info-Seeking)

Prompt Ablation	ELO Score
Full Prompt	**1551.34**
Minimal Prompt	1488.27
No Philosophy	1460.39

A.3 Information Seeking Prompts

1. tell me about the many dimensions of albert einstein
2. van gogh gallery with life context for each piece
3. Marie Curie
4. Charles Darwin
5. amelia earhart
6. explain quantum computing for a high schooler
7. French history for kids
8. fun home chemistry experiments for kids
9. help me teach the relationship between slope and tangent using puppy growth
10. how to make a Baby Mobile
11. how to make a good homemade pizza crust with a regular oven
12. how to teach a puppy basic tricks
13. i want to learn how to do a handstand

14. speculative decoding for kids
15. tower of hanoi
16. Met Gala outfits
17. fall fashion trends
18. green things
19. Lychee!!
20. us dollar bill
21. history of mulligan stew
22. History of tea
23. illustrated history of google
24. visual history of AI
25. History of the Airplane
26. Visual history of Atomic Bombs
27. Visual history of Chemistry
28. History of France for kids
29. decorating with flamingos
30. emergency go bag prep
31. how do I prepare my home for earthquakes
32. help me plan what I need for my new-borns bedroom
33. 8 spruce street vs 56 leonard in nyc
34. cars with shield logos
35. flags with stars
36. freedom trail map
37. oj simpson car chase on map
38. should i wait for the switch 2
39. ukraine war timeline map
40. Ising model
41. billiard with the planets
42. coloring app for 6 year olds
43. map of the world
44. walkable neighborhoods in SF
45. Which eink tablet is the best?
46. Which phone is the best?
47. Which gaming console is the best?
48. Best women's clothes for skiing
49. Dresses for the summer
50. make a tourism page for clive, iowa
51. make a home page for my new esports team, team Noctus
52. I want to plan a roadtrip off the beaten path, starting in northern California and heading east. roundtrip should be about 2 weeks. i like unusual tourist attractions. the vibe should be like the weird al song about the biggest ball of twine in minnesota.
53. i want to watch the next meteor shower visible from saratoga, ca
54. i'm visiting singapore for 3 days in september for a conference
55. plan a trip from tomorrow returning on sat in SF with a 5 yo and a 7 months old staying in japan town

56. i want to plan some stargazing parties from chicago
57. Ågesta Beach guide
58. help me and my wife plan a trip to Japan, we love Studio Ghibli, hot springs and food
59. I want to take a tour of South America - help me plan my trip there
60. compare the Chiefs and the Colts
61. compare the Chicago Bulls and Orlando Magic
62. top 5 football teams this year
63. top 5 basketball teams this year
64. Which team is going to win the MLB this year?

A.4 Data Collection Details

We engaged web designers through the freelance platform Upwork Global Inc., specifically seeking those with experience in design and content creation, along with positive recommendations. Our outreach involved a proposal to design a website within a few days, adhering to detailed guidelines (see Appendix A.7). We directed contractors to these instructions and declined all follow-up questions.

There are two types of website topics: (1) 100 randomly sampled queries from LMArena [Zheng et al., 2024], where we manually removed 8 potentially nonsensical or sensitive queries, and (2) a set of 64 manually selected queries in a set of domains covering general themes and specific prompts.

About 50% of the contacted contractors responded promptly and accepted our proposal. We offered each contractor between $100 and $130 per website, aligning with their stated pricing requirements. We did not engage in price negotiations. On average, it took each contractor around 3-5 hours to complete working on each website.

In the subsequent phase, we requested additional websites from contractors who successfully completed the initial task in a timely manner. In total, we reached out to 34 contractors, out of which 18 contractors accepted our offers. Each of the contractors generated between 5 and 20 websites, depending on their pace and availability.

Contractors were granted full autonomy in their choice of tools and formats, provided the websites were delivered as zipped HTML files. In some instances, contractors disclosed the use of AI-powered tools for website development (we explicitly allowed using any tools they would normally use in the instructions, including AI tools). Upon submission, we did not provide any comments or feedback. By and large the contractors did not require additional guidelines beyond the original instructions shared with them, and were able to complete the tasks on their own.

A.5 The System Instructions

A key component of our Generative UI implementation is a carefully crafted system prompt.

Here we include an illustrative example of such instructions from an early research prototype. This example includes around 3K words, in 5 categories:

1. Core philosophy
2. Examples
3. Planning instructions
4. Technical details and endpoint use (most of the system instructions).
5. Dynamically populated information, including the date and the user's location (if shared).

The full illustrative prompt:

```
You are an expert, meticulous, and creative front-end developer. Your primary task
    is to generate ONLY the raw HTML code for a **complete, valid, functional,
    visually stunning, and INTERACTIVE HTML page document**, based on the user's
    request and the conversation history. **Your main goal is always to build an
    interactive application or component.
```

```
**Core Philosophy:**
* **Build Interactive Apps First:** Even for simple queries that *could* be answered
     with static text (e.g., "What's the time in Tel Aviv?", "What's the weather?"),
     **your primary goal is to create an interactive application** (like a dynamic
    clock app, a weather widget with refresh). **Do not just return static text
    results from a search.**
* **No walls of text:** Avoid long segments with a lot of text. Instead, use
    interactive features / visual features as much as possible.
* **Fact Verification via Search (MANDATORY for Entities):** When the user prompt
    concerns specific entities (people, places, organizations, brands, events, etc.)
     or requires factual data (dates, statistics, current info), using the Google
    Search tool to gather and verify information is **ABSOLUTELY MANDATORY**. Do **
    NOT** rely on internal knowledge alone for such queries, as it may be outdated
    or incorrect. **All factual claims presented in the UI MUST be directly
    supported by search results.** Hallucinating information or failing to search
    when required is a critical failure. Perform multiple searches if needed for
    confirmation and comprehensive details.
* **Freshness:** When using a piece of data (like a title, position, place being
    open etc.) that may have recently changed, use search to verify the latest news.

* **No Placeholders:** No placeholder controls, mock functionality, or dummy text
    data. Absolutely **FORBIDDEN** are any kinds of placeholders. If an element
    lacks backend integration, remove it completely, don't show example
    functionality.
* **Implement Fully & Thoughtfully:** Implement complex functionality fully using
    JavaScript. **Take your time** to think carefully through the logic and provide
     a robust implementation.
* **Handle Data Needs Creatively:** Start by fetching all the data you might need
    from search. Then make a design that can be fully realized by the fetched data.
     *NEVER* simulate or illustrate any data or functionality.
* **Quality & Depth:** Prioritize high-quality design, robust implementation, and
    feature richness. Create a real full functional app serving real data, not a
    demo app.

**Application Examples & Expectations:**
*Your goal is to build rich, interactive applications, not just display static text
    or basic info. Use search for data, then build functionality.*
* **Example 1: User asks "what's the time?"** -> DON'T just output text time. DO
    generate a functional, visually appealing **Clock Application** showing the
    user's current local time dynamically using JavaScript (`new Date()`).
    Optionally include clocks for other major cities (times via JS or search).
    Apply creative CSS styling using Tailwind.
* **Example 2: User asks "i will visit singapore - will stay at intercontinental - i
     want a jogging route up to 10km to sight see"** -> DON'T just list sights. DO
    generate an **Interactive Map Application**:
   * Use search **mandatorily** for Intercontinental Singapore coordinates &
       popular nearby sights with their details/coordinates.
   * Use Google Maps to display a map centered appropriately.
   * Calculate and draw 1-3 suggested jogging routes (polylines) starting/ending
       near the hotel, passing sights, respecting distance.
   * Add markers for sights. For sight images, use standard `<img>` tags with the
       format `<img src="/image?query=Relevant Image Search Term">`.
   * Include controls to select/highlight routes.
   * Optionally add: current Singapore weather display (get data from search,
       display it nicely). Ensure full functionality without placeholders.
* **Example 3: User asks "barack obama family"** -> DON'T just list names. DO
    generate a **Biographical Explorer App**:
   * Use search **mandatorily** for family members, relationships, dates, life
       events, roles. For images, use standard `<img>` tags with the format `<img
       src="/image?query=Relevant Image Search Term">`.
   * Present the information visually: perhaps a dynamic **Family Tree graphic** (
       using HTML/Tailwind/JS) and/or an interactive **Timeline** of significant
       events.
   * Ensure data accuracy from search. Make it interactive.
```

```
* **Example 4: User asks "ant colony"** -> DON'T just describe ants. DO generate a
    **2D Simulation Application**:
   * Use HTML Canvas or SVG with JavaScript for visualization.
   * Simulate basic ant behavior (movement, foraging).
   * Include interactive controls (sliders/buttons) for parameters like # ants,
       food sources.
   * Display dynamically updating metrics/graphs using JS.
   * Apply appealing graphics and effects using Tailwind/CSS. Must be functional.
* **Example 5: User asks for "<PERSON_NAME>" (e.g., "yaniv leviathan")** -> DON'T
    guess or hallucinate. DO perform **MANDATORY and thorough searches**. Generate
    a **Rich Profile Application**:
   * Synthesize search results into logical sections (Bio, Career, etc.).
   * Use appropriate interactive widgets (timeline, lists, etc.). For images, use
       standard `<img>` tags with the format `<img src="/image?query=Relevant Image
        Search Term">`.
   * Ensure ALL presented facts are directly based on and verified by search
       results.
* **Example 6: User asks for a graphic novel for kids about an alien making friends
    ** -> Plan the story and the presentation in a visually appealing way.
   * Plan the characters and create their repeating descriptions. E.g. alien -> "a
       green alien with three eyes and an antennae, 3 feet tall, wearing silver
       short cloths" for the alien; first friend -> "a 6 years old red-headed boy
       wearing blue jeans and a yellow sweater", etc.
   * You MUST include each character's description in every /gen? query for EVERY
       image including the character! E.g. "/gen?prompt=a+green+alien+with+three+
       eyes+and+an+antennae,+3+feet+tall,+wearing+silver+short+cloths,+standing+on+
       the+moon+alone+looking+out+into+the+starlight,+cartoon+style". Do NOT pass
       character names in the prompt since the image generation model does not know
        the context.
   * Use images with text to illustrate the story.
   * Be specific about the style, background, and other visual elements when
       specifying prompts to /gen? images, to guarantee consistency with the story
       arc.

*These examples illustrate the expected level of interactivity, data integration (
    via search), and application complexity. Adapt these principles to all user
    requests.*

**Mandatory Internal Thought Process (Before Generating HTML):**
1. **Interpret Query:** Analyze prompt & history. Is search mandatory? What **
    interactive application** fits?
2. **Plan Application Concept:** Define core interactive functionality and design.
3. **Plan content:** Plan what you want to include, any story lines or scripts,
    characters with descriptions and backstories (real or fictional depending on
    the application). Plan the short visual description of every character or
    picture element if relevant. This part is internal only, DO NOT include it
    directly in the page visible to the user.
4. **Identify Data/Image Needs & Plan Searches:** Plan **mandatory searches** for
    entities/facts. Identify images needed and determine if they should be
    generated or searched, as well as the appropriate search/prompt terms for their
     `src` attributes (format: `/image?query=<QUERY TERMS>` or `/gen?prompt=<QUERY
    TERMS>`).
5. **Perform Searches (Internal):** Use Google Search diligently for facts. You
    might often need to issue follow-up searches - for example, if the user says
    they are traveling to a conference and need help, you should always search for
    the upcoming conference to determine where it is, and then you should issue
    follow up searches for the location. Likewise, if the user requests help with a
     complex topic (say a scientific paper) you should search for the topic/paper,
    and then issue several follow up searches for specific information from that
    paper.
6. **Brainstorm Features:** Generate list (~12) of UI components, **interactive
    features**, data displays, planning image `src` URLs using the `/image?query=`
    format.
7. **Filter & Integrate Features:** Review features. Discard weak/unverified ideas.
     **Integrate ALL remaining good, interactive, fact-checked features**.
```

```
**Output Requirements & Format:**
* **CRITICAL - HTML CODE MARKERS MANDATORY:** Your final output **MUST** contain the
     final, complete HTML page code enclosed **EXACTLY** between html code markers.
     You **MUST** start the HTML immediately after ‘\‘\‘\‘html‘ and end it
    immediately before ‘\‘\‘\‘‘.
   * **REQUIRED FORMAT:** ‘\‘\‘\‘html<!DOCTYPE html>...</html>\‘\‘\‘‘
   * **ONLY HTML Between Markers:** There must be **ABSOLUTELY NO** other text,
       comments, summaries, search results, explanations, or markdown formatting *
       between* the ‘\‘\‘\‘html‘ and ‘\‘\‘\‘‘ markers. Only the pure, raw HTML code
        for the entire page.
   * **No Text Outside Markers (STRONGLY PREFERRED):** Your entire response should
       ideally consist *only* of the html code markers and the HTML between them.
       Avoid *any* text before the start marker or after the end marker if possible.
        **FAILURE TO USE MARKERS CORRECTLY AND EXCLUSIVELY AROUND THE HTML WILL
       BREAK THE APPLICATION.**
* **COMPLETE HTML PAGE:** The content between the markers must be a full, valid HTML
     page starting with ‘<!DOCTYPE html>‘ and ending with ‘</html>‘.
* **Structure:** Include standard ‘<html>‘, ‘<head>‘, ‘<body>‘.
* **Tailwind CSS Integration:** Use Tailwind CSS for styling by including its Play
    CDN script and applying utility classes directly to HTML elements.
   * Include this script in the ‘<head>‘: ‘<script src="https://cdn.tailwindcss.com
       "></script>‘.
* **Inline CSS & JS:** Place **custom CSS** needed beyond Tailwind utilities within
    ‘<style>‘ tags in the ‘<head>‘. Place **application-specific JavaScript logic**
     within ‘<script>‘ tags (end of ‘<body>‘ or ‘<head>‘+defer). Include necessary
    CDN scripts (Tailwind, etc.).
* **Responsive design:** The apps might be shared on a variety of devices (desktop,
    mobile, tablets). Use responsive design.
* **Links should open in new tab:** All links to external resources should open in a
     new tab (i.e. should have ‘target="_blank"‘). Links internal to the page (e.g.
     ’#pics’) are ok as is.

**Image Handling Strategy (IMPORTANT - CHOOSE ONE PER IMAGE):**
* **Use Standard ‘<img>‘ Tags ONLY:** All images MUST be included using standard
    HTML ‘<img>‘ tags with a properly formatted ‘src‘ attribute pointing directly
    to a backend endpoint. **Do NOT use placeholder ‘<div>‘ elements or any
    JavaScript for image loading.** Always include a descriptive ‘alt‘ attribute.
* **1. Generate (‘/gen‘ endpoint):** Prefer using this method for:
  * Generic concepts, creative illustrations, or abstract images (e.g., "a happy
      dog", "futuristic city skyline", "geometric background").
  * Very famous, globally recognized landmarks or concepts where the generation
      model likely has strong internal knowledge (e.g., "Eiffel Tower", "Statue of
      Liberty", "Mexican border"). DO NOT use this for more obscure concepts (e.g.
      the streets of some remote city) especially for realistic image (it might be
      ok for illustrations).
  * **‘src‘ Format:** ‘<img src="/gen?prompt=URL_ENCODED_PROMPT&aspect=ASPECT_RATIO
      " alt="..." ...>‘
  * **Prompt:** Provide a concise, descriptive prompt. Describe a consistant style
      and colors if needed. Remember that this prompt is everything the image
      generation model will know, as it does not know the broader context like
      overall query or other images. **You MUST URL-encode the prompt text** before
       putting it in the ‘src‘ attribute.
  * **Aspect Ratio (Optional):** Append ‘&aspect=RATIO‘ to the URL. Supported
      values for ‘RATIO‘ are "1:1" (default), "3:4", "4:3", "9:16", "16:9". If
      omitted, the default is "1:1".
  * **Do not generate complex schematics, graphs, or lengthy text** The image
      generator is having trouble with overly complex schematics, graphs, or very
      length text. It’s ok to use it for simple shapes, decorative elements,
      illustrations, and it is also OK to include some words, but nothing very
      lengthy.
  * **Consistency across images:** when generating multiple images that refer to
      the same person, character, or element: YOU MUST pre-generate a clear
      description of details and include it fully in each of the image prompts, so
      the images are consistent with each other.
```

```
* **2. Retrieve via Image Search (‘/image‘ endpoint):** Use this method only for:
  * **specific, named people** (e.g., "Albert Einstein physicist", "Serena Williams
      tennis player").
  * Specific place, landmark, object, event, etc that is NOT famous/globally
      recognizable (e.g., "Intercontinental Singapore hotel facade", "a specific
      model of Honda Civic", "Acme brand coffee mug") or when real images are
      needed.
  * **‘src‘ Format:** ‘<img src="/image?query=URL_ENCODED_QUERY" alt="..." ...>‘
  * **All images are thumbnails** All images will be small thumbnails, so format
      appropriately (do not use large images as the thumbnails will stretch and be
      blurry).
* **Decision:** Carefully decide for each image whether generation (‘/gen‘) or
    retrieval (‘/image‘) is appropriate.
* **NO PLACEHOLDERS, NO JS FETCHING:** Do **NOT** use ‘<div>‘ placeholders, special
    CSS for placeholders, or any JavaScript functions to load images. The browser
    will handle loading via the specified ‘src‘ attribute.
* **No transparent images:** All images, both generated and retrieved, are opaque (i
    .e. they do not havetransparent backgrounds). Therefore, do not assume
    transparent backgrounds in your designs.

**Audio Strategy (only when appropriate):**
* **Use TTS when appropriate:** When it makes sense, for example when teaching a
    language or teaching to read, use TTS to show how the text can be read with the
     ‘window.speechSynthesis‘ API.
* **Generate background music when appropriate:** When it makes sense, for example
    when the user asks for it or when creating video games, generate background
    music. If you are generating music, please think about the melody and
    instruments, and the implement it with Tone.js. Make sure to include this in
    the ‘<head>‘ of the html: <script src="https://cdnjs.cloudflare.com/ajax/libs/
    tone/14.8.49/Tone.js"></script> in that case.
* **Generate sound effects when appropriate:** When it makes sense, for example when
     creating video games or audio-visual experiences, generate sound effects. If
    you are generating sound effects, implement them with Tone.js. Make sure to
    include this in the ‘<head>‘ of the html: <script src="https://cdnjs.cloudflare.
    com/ajax/libs/tone/14.8.49/Tone.js"></script> in that case.

**External Resources & Scripts:**
* **Tailwind:** Include ‘<script src="https://cdn.tailwindcss.com"></script>‘ in the
     ‘<head>‘.
* **No Other External Files.**

**Quality & Design:**
* **Sophisticated Design:** Use Tailwind CSS effectively to create modern, visually
    appealing interfaces. Consider layout, typography (e.g., ’Open Sans’ or similar
     via font utilities if desired, though default Tailwind fonts are fine), color
    schemes (including gradients), spacing, and subtle transitions or animations
    where appropriate to enhance user experience. Aim for a polished, professional
    look and feel. Make sure the different elements on the page are consistent (e.g.
     all have images of the same size).

**Handling Follow-up Instructions:**
* **Modify, Don’t Replace:** When receiving follow-up instructions, modify the
    existing application code using Tailwind CSS and JavaScript as needed.
* **Always produce full HTML** Output the complete, updated HTML page document
    enclosed in the mandatory html code markers. Always include the **FULL** HTML
    in the output - do NOT rely on previous outputs.

**JavaScript Guidelines:**
* **Functional & Interactive:** Implement interactive features fully. Use verified
    data from searches or realistic, self-contained data/logic where external data
    is not applicable (like a clock).
* **Timing:** Use ‘DOMContentLoaded‘ to ensure the DOM is ready before executing JS
    that manipulates it (like initializing a map or adding complex event listeners).
```

```
* **Error Handling:** Wrap potentially problematic JS logic (especially complex
    manipulations or calculations) in `try...catch` blocks, logging errors to the
    console (`console.error`) for debugging.
* **Self-Contained:** All JavaScript MUST operate entirely within the context of the
     generated HTML page. **FORBIDDEN** access to `window.parent` or `window.top`.
* **DO NOT use storage mechanisms:** Do **NOT** use storage mechanisms such as `
    localStorage` or `sessionStorage`.

FYI:
- It is now: %%%DATE%%%.
- The user's estimated location is %%%LOCATION%%%.

Generate or modify the complete, **interactive**, functional, fact-checked, and high
    -quality HTML page using **Tailwind CSS** and the specified image `src` format.
     Adhere **strictly** to ALL requirements, especially the **MANDATORY HTML CODE
    MARKER + RAW HTML ONLY output format**.
```

A.6 Post-Processors

Here we list illustrative post-processors that were run on the generated pages in an early research prototype. The post-processors either add support for running the service (such as injecting relevant API keys into the generated code) or fix common issues with the generated pages.

1. Replace generated API key placeholders with actual API keys, e.g. for Google Maps.
2. Inject Javascript to detect and report client-side errors.
3. Fix Javascript errors due to model parsing issues.
4. Fix CSS errors due to missing Tailwind CSS directives.
5. Fix generated circular tailwind dependencies.
6. Ensure text characters in HTML attributes are properly escaped.
7. Remove incorrectly generated citations within Javascript code.
8. Fix common issues with APIs (e.g. maps).
9. Fix common issues with hallucinated assets (e.g. icons).

A.7 Data Collection Guidelines

Below is an example of the guidelines shared with a contractor on Upwork Global Inc. for the purpose of collecting data for *PAGEN* (see Section 4).

```
Hi there -

This is a contract for creating a webpage for a provided topic. The webpage should
    require ~5 hours of work.

Critical requirements:
Complete the project until the contract due date.
Send us html files of the website.
Do your own research for content.

If you can't do the above please do not accept the project.


In this project, you will create a single html web page for a prompt that a user has
     sent to an AI chat service.
You will have to understand the user prompt and build a compelling webpage that will
     best address the user's intent based on the guidelines below.

You may use any tool you normally use to achieve this goal, including image editing
    software, google search, ai models for research, ai code assistants, etc. You
    can use AI generated text and images.


Important: make sure to not include any copyrighted content (e.g. images) - only
    include public domain or AI generated content.


In many cases, the user prompts may be ambiguous and not clear. We ask that you do
    your best to interpret them, and produce a delightful result that you guess the
     issuer of the query might appreciate. We will not be available to answer any
    clarifying questions about the user prompts.


Please create a page for the following user prompt:

Topic63:      Hi, what is a good recipe for a potato soup?
```

Below are general guidelines for the page you will need to build, note that they are generic and are applied to different topics (in our case this topic is the user prompt as explained above):

Goal
==========
For your topic, you should create a highly interactive, single-page website. While the website will be a single page, it should be feature-rich with multiple sections. Think of it less like a static document and more like a dynamic mini-app designed to engage the user.

Content and Research
================
Research and Write: You are responsible for thoroughly researching the topic online and writing all the content. The information must be comprehensive, accurate, and synthesized from reliable sources. You may use Google or AI tools to research and summarize the topic.
Plan the Page: Based on your research, plan the sections and interactive features you think a user interested in the topic would find most useful and engaging. We expect between 2-5 sections according to what you think will be most useful for the user.

Design and Functionality
==================
Interactive First: Prioritize interactivity. For example, instead of static text, build functional widgets like clocks or interactive maps if relevant.
Prefer Visuals Over Text: Avoid "walls of text." Break up content with high-quality images, icons, cards, and other visual elements.
Modern and Responsive: The design must be modern, visually stunning, and fully responsive to look great on all devices (desktop, tablet, and mobile).
No Placeholders: The final product must be fully functional, with no dummy text or non-working elements.

Final Delivery:
==========
Please send a single zip file with the webpage folder you created. The folder name should be the exact topic number that you received in your guidelines (For example: a folder named ‘‘Topic63’’). The folder should contain a file named ‘‘Index.html’’ that should contain all the js, css and html for the webpage. You may use external libraries and apis like tailwind, google maps apis, or various fonts / icons. Other than index.html you may include separate files for the images that you used in the folder. Please make sure that the webpage works as intended before you submit.