PromptInfuser: Bringing User Interface Mock-ups to Life with Large Language Models

Savvas Petridis
Google Research
New York, New York, USA
petridis@google.com

Michael Terry
Google Research
Cambridge, Massachusetts, USA
michaelterry@google.com

Carrie J. Cai
Google Research
Mountain View, California, USA
cjcai@google.com

ABSTRACT

Large Language Models have enabled novices without machine learning (ML) experience to quickly prototype ML functionalities with prompt programming. This paper investigates incorporating prompt-based prototyping into designing *functional* user interface (UI) mock-ups. To understand how infusing LLM prompts into UI mock-ups might affect the prototyping process, we conduct a exploratory study with five designers, and find that this capability might significantly speed up creating functional prototypes, inform designers earlier on how their designs will integrate ML, and enable user studies with functional prototypes earlier. From these findings, we built PromptInfuser, a Figma plugin for authoring LLM-infused mock-ups. PromptInfuser introduces two novel LLM-interactions: *input-output*, which makes content interactive and dynamic, and *frame-change*, which directs users to different frames depending on their natural language input. From initial observations, we find that PromptInfuser has the potential to transform the design process by tightly integrating UI and AI prototyping in a single interface.

CCS CONCEPTS

• **Human-centered computing** → **Empirical studies in HCI**; *Interactive systems and tools*; • **Computing methodologies** → *Machine learning*.

KEYWORDS

Prototyping, Large Language Models, Generative AI, Design

ACM Reference Format:
Savvas Petridis, Michael Terry, and Carrie J. Cai. 2023. PromptInfuser: Bringing User Interface Mock-ups to Life with Large Language Models. In *Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems (CHI EA '23), April 23–28, 2023, Hamburg, Germany.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3544549.3585628

1 INTRODUCTION

Prototyping is an essential part of the design process, in which designers create mock-ups of user interfaces (UIs) to evaluate their design. Prototyping artificial intelligence (AI) applications has traditionally been difficult for designers as they might not have a good understanding of the capabilities of AI, often require the assistance of technical experts, and can have trouble thinking of new use-cases of AI [3] [5]. However, recent advances in large language models (LLMs) and the appearance of "prompt programming" have dramatically reduced the barriers of prototyping AI functionality. Novices without any background in programming and machine learning can quickly create custom AI functionalities like translation and summarization through natural language prompts. For example, with an LLM, a user can tailor its functionality to fulfill English-to-French translation by providing a prompt consisting of English and French pairs: (e.g. *"English: Hello. French: Bonjour. English: Where is the bus? French: Où est le bus? English: How are you? French:"*). With this prompt, the LLM will likely output the French translation: *"Comment allez-vous?"*.

Through natural language prompting, designers can use LLMs as a *design material*, for quickly understanding and communicating AI capabilities. In this vein, recent work has introduced "prompt-based prototyping", which explores how LLM prompting might affect the prototyping process [4]. In this prior work, prompt programming enabled individuals without ML expertise to rapidly create AI functionality, understand the capabilities of AI, and communicate their AI ideas to collaborators.

While prompting has been shown to substantially lower the barrier to prototyping new AI functionality, this form of AI prototyping still occurs outside of the context of building an actual prototype with a user interface. Currently, designers are left to prototype their LLM prompts and user interfaces in two separate environments, and as a result, they are likely to have less of an understanding of how the AI functionality might mesh with their UI, and will likely need to spend time iterating over their design when integrating this functionality into their interface. By tightly incorporating LLM prompting into the design of user interfaces, we can (1) enable the creation of fully functional and interactive AI prototypes and (2) help inform designers earlier on how the two technologies will combine. In this space, there are many open questions, including:

- RQ1: What ways might integrating prompt programming and UI design affect the prototyping process?
- RQ2: How should we help designers author UI mock-ups with LLM functionality?

To answer these questions, we conducted a exploratory study with five professional designers and found that the ability to infuse LLM prompts into mock-ups might change prototyping by:

- Helping designers create functional prototypes quickly, without relying on engineers or ML experts.
- Informing designers earlier in the prototyping process on how the AI functionality will specifically integrate into the application UI.

CHI EA '23, April 23–28, 2023, Hamburg, Germany

ACM ISBN 978-1-4503-9422-2/23/04.
https://doi.org/10.1145/3544549.3585628

- Enabling user testing on functional prototypes earlier in the design process.

Inspired by these findings, we built PromptInfuser, an interactive tool for adding AI functionality to UI mock-ups with LLM prompts. PromptInfuser is a plugin for Figma, a popular prototyping tool that designers commonly use to create mock-ups of user interfaces [1]. To bring UI mock-ups in Figma to life, PromptInfuser introduces two novel LLM interactions that users can author. The first interaction, *input-output*, makes text content in mock-ups dynamic and interactive; users can hook up a text element as input into an LLM prompt and then display the prompt's completion to another text element, enabling extraction, classification, generation, and a multitude of other AI functionalities within the mock-up. The second interaction, *frame-change*, adds additional expressivity by enabling authors to direct users to different frames (or screen designs) of content, depending on their natural language input. From initial observations, we find that these two interactions enable the creation of expressive AI mock-ups and that they transform the design process by tightly integrating both AI and UI prototyping. We also consider how a user's prompt decisions affect their resulting design and how to make this process smoother and accessible to prompt programming novices via additional support and scaffolding.

Overall, this paper contributes (1) a set of findings from an exploratory design probe, detailing how tightly integrating UI and LLM prompt design can affect the prototyping process and (2) PromptInfuser, our prototype that enables users to infuse LLM prompts into UIs with two novel interaction types: *input-output* and *frame-change*.

2 EXPLORATORY STUDY

We conducted a exploratory study to answer our two research questions and understand: (1) how integrating prompt programming with UI design might affect the prototyping process and (2) how we should support designers in authoring LLM-infused UI mock-ups.

2.1 Procedure

We recruited 5 professional designers at a large technology company (3 male, 2 female). We identified designers that already had experience writing LLM prompts. At the start of the study, each designer was asked questions that probed how integrating LLM functionality into prototypes they have made in the past might have helped them. Then to provide further context, each designer was shown a simple demo of an LLM-infused translation application prototype, made in Google Slides. This demo was made prior to PromptInfuser and involved creating a Google Slide extension that sends a translation prompt to an LLM and outputs the translation on a slide. Within the demo, users could alter the input-language text (e.g. "Hello") on the slide and then see a real translation appear in the output text-box (e.g. "Bonjour," generated with a translation LLM prompt). After viewing this demo, the designers were given 10 minutes to put together a mock-up of an application where the user inputs a natural language description of the weather and the interface outputs a few suggested outfits to wear. The designers made this mock-up in Google Slides, and as they drew the interface, designers were also asked to write out the LLM prompts and show how the prompt's inputs (e.g. weather) and outputs (e.g. clothing) connect to the interface (by drawn arrows indicating inputs and outputs). That is, the designers could not author actual LLM functionality. This was a design probe to explore how prompting might be done within the context of creating a UI-mock-up; thus, participants were asked to *imagine* their prompts were functional as they connected them to their design. Afterwards, they were interviewed on how they might imagine authoring these interactions. The study took approximately 30 minutes of each participant's time.

[1] https://www.figma.com/

2.2 Findings

2.2.1 RQ1. How does integrating prompt programming and UI design affect the prototyping process? Many participants noted that a tool enabling easy integration of LLM prompts into lower fidelity prototypes, such as those made in Figma or Google Slides, would **greatly reduce the time required to create a functional prototype**. Currently, to incorporate the outputs of an LLM prompt, the designers need to have a functional web application that is connected to the model, which requires significant time to set up, and often additional help from an engineer. P3 explained, "Now I've got a functional mock-up, which is sort of self-evidently useful... the advantage of this is that I did this in ten minutes. I can do these Lo-Fi mock-ups really quickly, either here [Google Slides] or in Figma". With a way to build functional prototypes quickly, P3 continued to explain that he could spend less time convincing engineers to help him build an application and instead quickly experiment with different iterations of the Lo-Fi prototype. At the same time, P5 explained that having a functional mock-up is sometimes more useful than a static mock-up as they enable clearer communication for an idea and ensure "that we're all understanding the concept in the same way". Overall, including LLM functionality into mock-ups would enable designers to rapidly construct functional mock-ups, helping them (1) allocate more of their time to designing and experimenting with layouts, and (2) clearly communicate their AI application idea.

In addition to reducing the time to create functional prototypes, **integrating LLM functionality directly into Lo-Fi mock-ups helps designers understand their requirements of these models earlier in the design process**. From the task portion of the exploratory study, we observed that the processes of authoring prompts and mocking up a UI design inform each other. For example, P1 started designing his weather-to-outfit application by writing an LLM prompt that takes as input a description of the weather and outputs a description of a suggested wardrobe (Figure 1A). After writing this prompt, he began his UI design, which consisted of a voice-input mobile application, which upon receiving the user's weather description, directs them to a list of suggested wardrobe items (Figure 1B). After he finished his design, P1 realized that the output of his LLM prompt should be structured as well, to map each suggested item in the generation to a list element in the UI. He then revised his original prompt to output a list of wardrobe items (Figure 1C). Thus, as he constructed his LoFi mock-up for an application, P1 gained a clearer understanding of how the prompt would need to output its generations to fit his current design. In this way, integrating LLM functionality into mock-ups could help

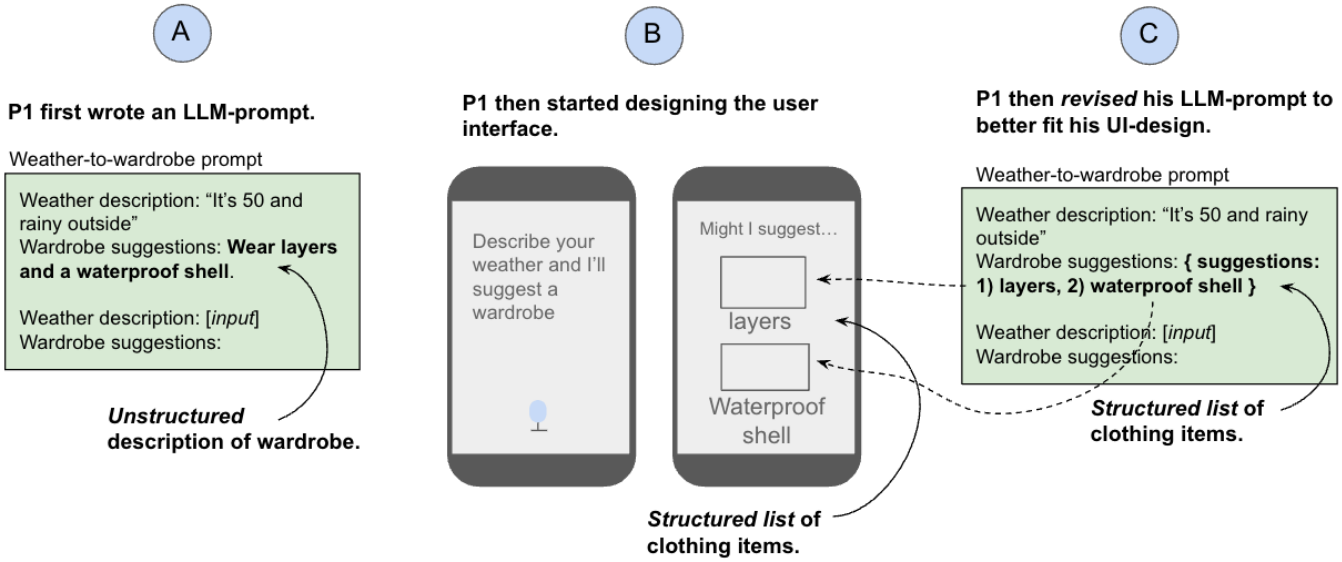

Figure 1: P1's prototyping process in the exploratory study, illustrating how the processes of LLM prompt writing and UI design influenced each other. They started by first writing a weather-to-wardrobe prompt that outputs an unstructured description of a wardrobe suggestion (A). After writing this prompt, they then start mocking up the user-interface, which he imagines as a voice-input system, which directs the user to a suggested list of wardrobe items (B). Realizing that the output of the LLM prompt should easily map to a list of wardrobe items (instead of an unstructured wardrobe description), P1 then revises his original prompt to output a structured list of items (C).

designers better understand their needs of these models earlier on, before spending the time to create a high-fidelity prototype.

Finally, with the ability to easily make functional LoFi mock-ups, **designers felt they would be able to conduct user studies with functional prototypes earlier in the design process**. Conducting a user study on a functional prototype is very valuable and allows designers to see how a user might *actually* interact with their proposed design. Normally a functional prototype is quite costly to produce, involving quite a bit of development and involvement from engineers. But with a tool to produce these functional mock-ups quickly, P3 explained that he could conduct more informed user studies: "Now with a functional prototype, it's not 'Hey user, talk to me like as if I was the model - now we can just have them try it". P3 continued to explain that being able to alter the functional mock-up quickly would also enable conducting studies with multiple different versions of the mock-up to quickly converge on the most viable option. Thus, by making the process of creating functional prototypes faster, we can help designers quickly test multiple ideas and converge on final designs faster.

2.2.2 RQ2. How can we help designers author LLM-infused prototypes? Designers easily adopted the idea of mapping UI elements to the inputs and outputs of LLM prompts. As they authored their mock-ups, designers would often write LLM prompts, revise them after updating their UI design, and then reassign different UI elements to the LLM prompt. Based on this, a tool for helping users author LLM-infused mock-ups should help users easily assign and reassign UI elements to inputs and outputs of an LLM prompt. This assignment should be as easy as selecting UI elements and clicking on the relevant LLM prompt. Finally, to ensure that designers can rapidly construct these UI elements, this tool should be built atop prototyping tools they already know how to use, such as Figma or Google Slides. Therefore, a tool to create LLM-infused mock-ups should tightly integrate a common prototyping tool with an LLM, and enable designers to easily connect elements of their designs to this LLM.

3 PROMPTINFUSER

Informed by the exploratory study, we created PromptInfuser, our system that enables designers to infuse UI mock-ups with LLM functionality (Figure 2). PromptInfuser is implemented as a Figma plugin so that it is built atop a tool designers already commonly use to mock-up interfaces. PromptInfuser also incorporates a version [2] of the LLM detailed in [1], which is a large language model that is promptable in the same way as GPT-3, OpenAI's LLM [2]. When an LLM prompt is run in PromptInfuser, the prompt text and input is sent to a server containing the LLM, and the completion is sent back to PromptInfuser.

3.1 Two Infusable LLM-interactions

3.1.1 Input-output. PromptInfuser implements two primary LLM-based interactions to help designers make their mock-ups functional: *input-output* (Figure 2) and *frame-change* (Figure 3). The *input-output* interaction consists of taking a text element as input

[2] https://openai.com/api/

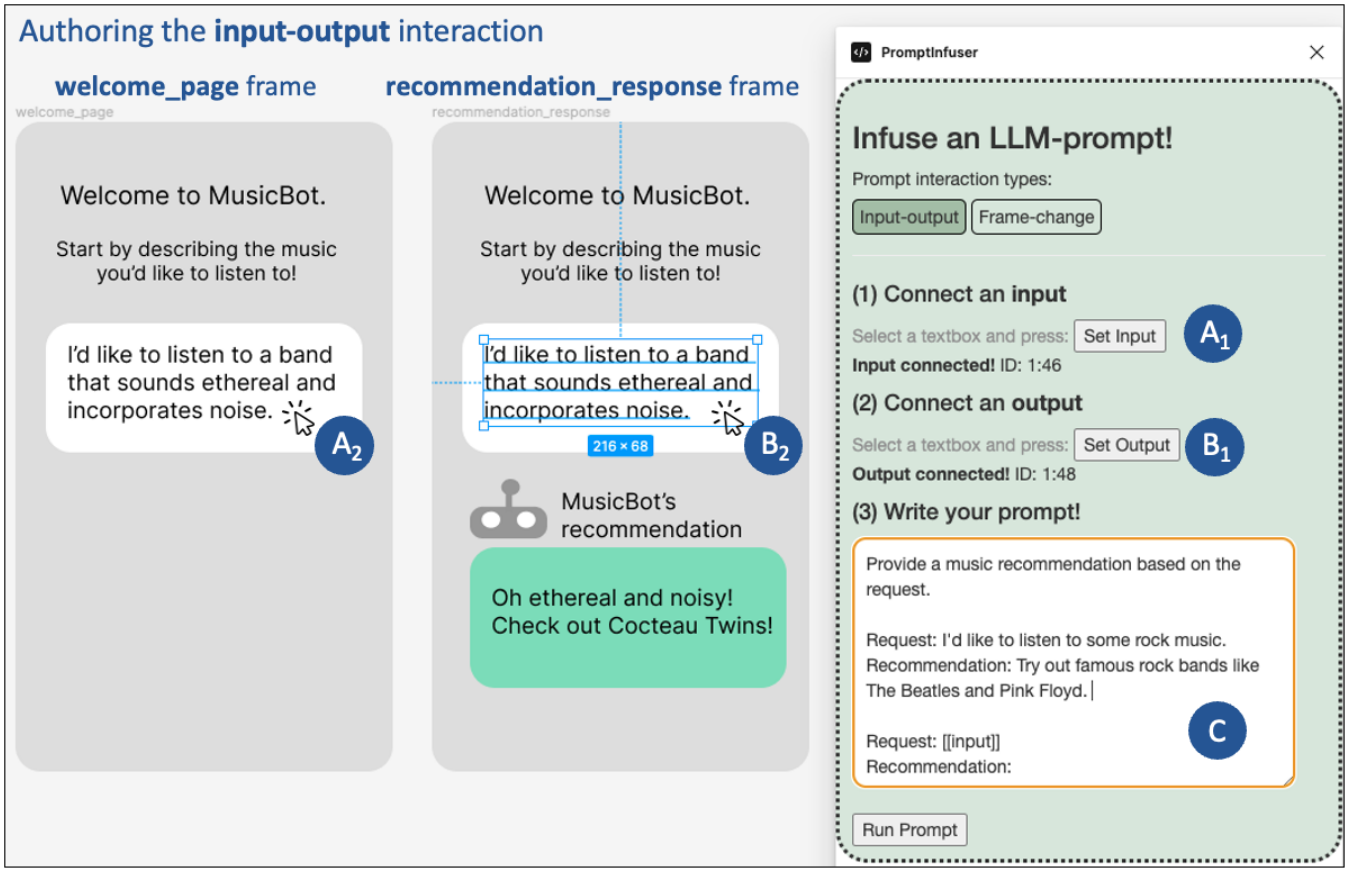

Figure 2: The *input-output* authoring interface in PromptInfuser. The *input-output* interaction consists of an LLM prompt that takes input from a text element and outputs its completion to another text element. Here, the user is trying to provide a music recommendation given a natural language request (A_2) and output this recommendation to another text element (B_2). To author this interaction, they selected (A_2) as the input text element and set it in the PromptInfuser sidebar (A_1). They also selected the output text element (B_2) and set it as the output in (B_1). Finally, they wrote the prompt which generates music recommendations based on a request in (C); [[input]] denotes where the text from (A_2) will be inserted. They can then press "Run Prompt" to execute this prompt and update the output text element (B_2).

from the UI, running an LLM prompt on this input text, and outputting the prompt's completion into another text element. This type of interaction was shown in the exploratory study (Section 2.1) to designers in the form of a translation application. To author this interaction, users simply assign input and output text elements in PromptInfuser and write the corresponding LLM prompt. For example, consider a PromptInfuser user designing an application where its users describe music requests and the system provides a music recommendation (Figure 2). Currently in Figma, the PromptInfuser user has two frames created, a *welcome_page* frame, where the user inputs their request, and a *recommendation_response* frame, where the system presents its recommendation based on the user's request. They would like to have an LLM prompt generate a music recommendation from the request inputted in Figure 2A_2 and then present this recommendation in a text element in the recommendation_response frame (Figure 2B_2). To do so, they select the input text element in Figure 2A_2, press "Set Input", and then complete the same process for the output text element. Finally, they write their prompt in Figure 2C, and include "[[input]]" to denote where the text from the input text element will go in the prompt. They press "Run Prompt" to execute this input-output interaction and change the output text element's content. With this interaction, users can infuse AI functions that dynamically update the text within their mock-ups.

3.1.2 Frame-change. While *input-output* interactions can make the content within mock-ups dynamic and interactive, they do not cover the wide range of interactivity required for a fully functional prototype. Prototypes often consist of multiple frames, or screens of content, which users are directed to depending on their input. This type of interaction can also be authored in PromptInfuser, with the *frame-change* interaction. Consider the same music recommendation scenario as before, but this time the PromptInfuser user would like to direct users of their application to a separate

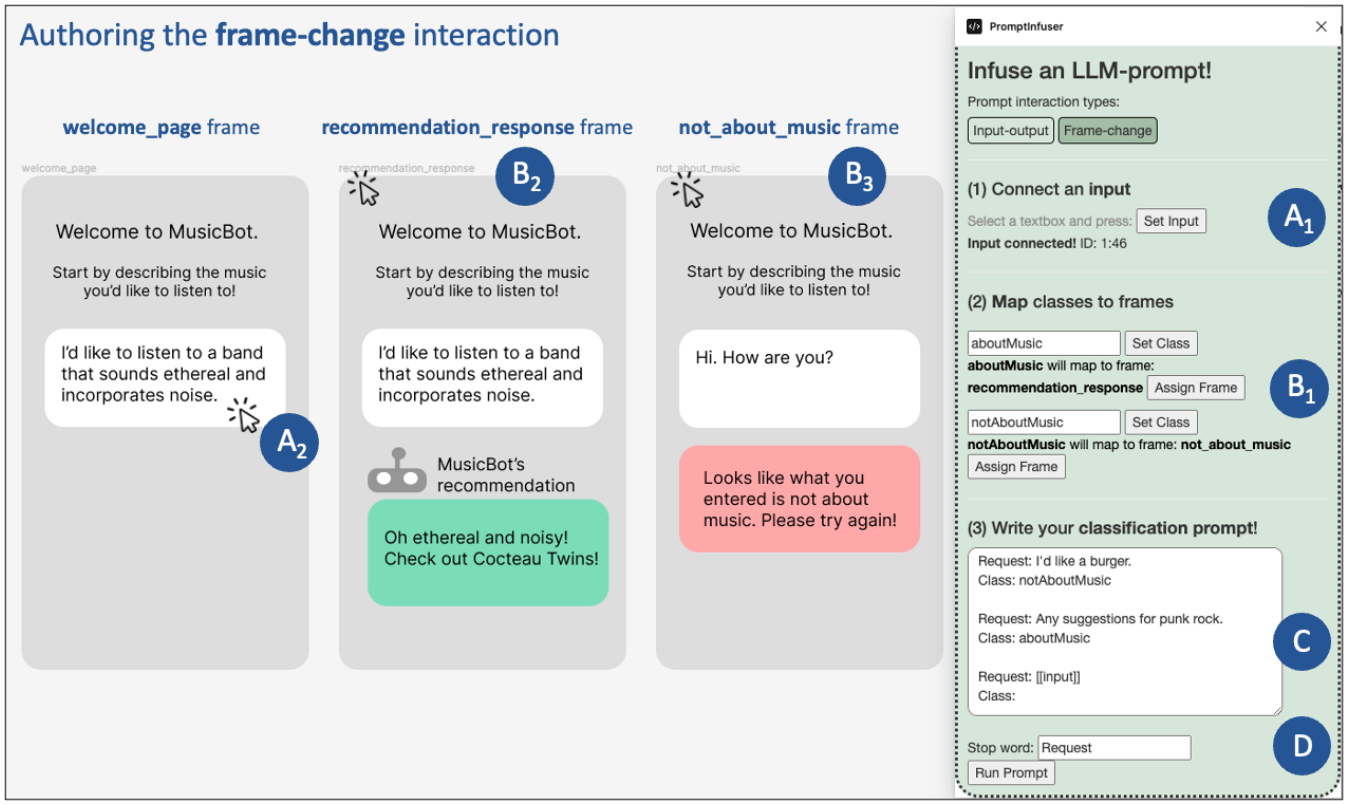

Figure 3: The *frame-change* authoring interface in PromptInfuser. The *frame-change* interaction consists of a classification prompt that takes user input and redirects them to a new frame, depending on how that user's input is classified. In this example, the goal is to classify the user's input request as either "aboutMusic" or "notAboutMusic", and to redirect them to the corresponding frames: *recommendation_response* and *not_about_music*. Here the user selected the input text element which contains the input request (A_2) and set that as the input (A_1). They then defined the mapping of the two classes: "aboutMusic" and "notAboutMusic" by first setting the classes in (B_1), selecting its corresponding frame (B_2 for "aboutMusic" and B_3 for "notAboutMusic"), and then pressing "Assign frame" in B_1. Finally, they wrote the classification prompt in (C) which determines if the request is about music; [[input]] is used to denote where the text from A_2 will be inserted in the prompt. By setting the stop word in (D), users can shorten the LLM completion to just the output class. Finally, they can press "Run Prompt" to execute the prompt, and depending on the output class, PromptInfuser alters its view port to place the correct corresponding frame in the center of the screen.

screen when their input request is not about music (Figure 3). Concretely, if the input on the *welcome_page* frame (Figure 3A_2) is classified as about music, users should be directed to the *recommendation_response* frame (Figure 3B_2), otherwise users should be directed to the *not_about_music* frame (Figure 3B_3). To author this interactivity, the user selects an input text element (Figure 3A_2) and sets it as the input in the PromptInfuser sidebar (Figure 3A_1). Next, the user defines the two classes that the input text could be classified as: *aboutMusic* or *notAboutMusic* (Figure 3B_1), and then selects each class's corresponding frame to map to in Figma. After defining the classes and their mappings, the user then writes a few-shot classification prompt, which determines if the given text is *aboutMusic* or *notAboutMusic* (Figure 3C). Finally, the user includes a stop word "Request" to truncate the LLM's output to only the output classification (Figure 3D). When "Run Prompt" is clicked, PromptInfuser will classify the input text and alter Figma's viewport to place the corresponding frame at the center of the screen. Overall, the *frame-change* interaction enables richer interactivity, where users are directed to frames with entirely different content based on their natural language input.

3.2 Initial Observations

The authors conducted their own informal experiments making LLM infused mock-ups with PromptInfuser. From initial observations, we found that PromptInfuser encourages a new form of design process, where prototyping of the UI and AI are tightly integrated. Users are constantly switching contexts between probing the LLM and designing frames and flows, and these two processes affect and inform each other. While this process can lead to exciting and dynamic UIs, it is a bit brittle, as users can get derailed troubleshooting

defective prompts, indicating a need for more structured prompt writing or error handling in future versions of PromptInfuser.

3.2.1 With PromptInfuser, UI and AI prototyping are tightly integrated. The performance of an LLM prompt greatly affects the design of the user interface. While mocking up an application, one author initially tried to write a prompt that when given input text, extracted a list of items from the text. However, the prompt they authored would often fail, by either hallucinating items that did not appear in the original input text or extracting the items imperfectly. From their prior experience prompt programming, they realized that this task might involve too many steps for a single LLM prompt and decided to decompose the task into (1) first identifying if the text contains items to extract and then (2) extracting these items. With this new breakdown of the AI functionality, they then needed to update their UI. To account for inputs without items to extract, they added a new frame for users to re-input new text. Overall, a user's knowledge of prompt programming ultimately affects how well they author prompts and how they decompose AI operations, which in turn affects the design of the user interface.

3.2.2 Helping designers author functional LLM prompts while prototyping. If a designer has trouble writing a functional LLM prompt or decomposing the prompt into easier steps, their prototyping process could get disrupted. Currently, PromptInfuser offers little support for designers to understand either how well their prompts are doing or how to quickly improve them. Even seasoned prompt programmers can have trouble getting a prompt to work for a variety of reasons, from finding the right way to phrase a prompt, coming up with examples, etc. [4]. From our early observations, we noticed that these same problems occur within PromptInfuser, and currently only those with ample experience in prompt programming can quickly prototype a functional interface. In the future, we can support more novice users in writing prompts by providing prior examples to build from, or updating the interface so that constructing a prompt is more scaffolded (e.g., users can be asked to input example pairs instead of writing a classification prompt from scratch). By supporting the prompt writing process, we can make developing AI-infused mock-ups smoother and accessible to more designers.

4 CONCLUSION AND FUTURE WORK

This paper has conducted an initial investigation into incorporating the power of prompt programming into the context of creating functional UI mock-ups. To understand how this capability might affect the prototyping process, we conducted an exploratory study and found that by enabling LLM infusion into mock-ups, we can potentially reduce the time needed to create a functional prototype, give designers an earlier understanding of how to integrate AI functionality into their design, and enable designers to conduct user studies on functional prototypes earlier. Inspired by this study, we created PromptInfuser, our Figma Plugin that enables designers to author LLM-infused mock-ups. With PromptInfuser, designers can author two novel LLM-based interactions. The first interaction, *input-output*, makes content in Figma interactive and dynamic; a text element can be inputted into an LLM prompt and the model's completion is displayed in another text element. The second interaction, *frame-change*, adds additional expressivity by directing users to different frames within Figma based on their natural language input. From initial observations, we found that PromptInfuser transforms the design process by tightly linking UI and AI prototyping, and could further support prompt writing to ensure a smoother prototyping experience.

Finally, there are many opportunities for future work, including identifying and implementing new forms of LLM-based interactions, on top of *input-output* and *frame-change*, incorporating support for designers to write functional prompts with PromptInfuser, investigating how large language models or text-to-image models could also help with generating frames and user interfaces for designers, and finally, conducting a formal evaluation of PromptInfuser and an in-depth analysis on how it affects the prototyping process.

REFERENCES

[1] Daniel Adiwardana, Minh-Thang Luong, David R. So, Jamie Hall, Noah Fiedel, Romal Thoppilan, Zi Yang, Apoorv Kulshreshtha, Gaurav Nemade, Yifeng Lu, and Quoc V. Le. 2020. Towards a Human-like Open-Domain Chatbot. https://doi.org/10.48550/ARXIV.2001.09977

[2] Eli Collins and Zoubin Ghahramani. 2021. LaMDA: our breakthrough conversation technology. https://blog.google/technology/ai/lamda/ Accessed: 2023-01-11.

[3] Matthew K. Hong, Adam Fourney, Derek DeBellis, and Saleema Amershi. 2021. Planning for Natural Language Failures with the AI Playbook. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 386, 11 pages. https://doi.org/10.1145/3411764.3445735

[4] Ellen Jiang, Kristen Olson, Edwin Toh, Alejandra Molina, Aaron Donsbach, Michael Terry, and Carrie J Cai. 2022. PromptMaker: Prompt-Based Prototyping with Large Language Models. In *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI EA '22)*. Association for Computing Machinery, New York, NY, USA, Article 35, 8 pages. https://doi.org/10.1145/3491101.3503564

[5] Qian Yang, Aaron Steinfeld, Carolyn Rosé, and John Zimmerman. 2020. Re-Examining Whether, Why, and How Human-AI Interaction Is Uniquely Difficult to Design. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3313831.3376301