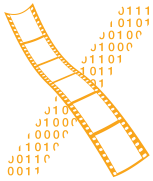

Chair for Computer Science 10 (Media Computing and Human-Computer Interaction)

RWTH AACHEN UNIVERSITY

Optimizing LLMs to Eliminate Deceptive Designs from Websites

Bachelor's Thesis at the
Media Computing Group
Prof. Dr. Jan Borchers
Computer Science Department
RWTH Aachen University

by
Lucia Karl

Thesis advisor:
Prof. Dr. Jan Borchers

Second examiner:
Prof. Dr. Ulrik Schroeder

Registration date: 13.06.2025
Submission date: 30.09.2025

Zentrales Prüfungsamt/Central Examination Office

RWTH AACHEN UNIVERSITY

Eidesstattliche Versicherung
Declaration of Academic Integrity

Karl, Lucia

Name, Vorname/Last Name, First Name

445074

Matrikelnummer (freiwillige Angabe)
Student ID Number (optional)

Ich versichere hiermit an Eides Statt, dass ich die vorliegende ~~Arbeit~~/Bachelorarbeit/ ~~Masterarbeit~~* mit dem Titel
I hereby declare under penalty of perjury that I have completed the present paper/bachelor's thesis/master's thesis* entitled

Optimizing LLMs to Eliminate Deceptive Designs from Websites

selbstständig und ohne unzulässige fremde Hilfe (insbes. akademisches Ghostwriting) erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt; dies umfasst insbesondere auch Software und Dienste zur Sprach-, Text- und Medienproduktion. Ich erkläre, dass für den Fall, dass die Arbeit in unterschiedlichen Formen eingereicht wird (z.B. elektronisch, gedruckt, geplottet, auf einem Datenträger) alle eingereichten Versionen vollständig übereinstimmen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.
independently and without unauthorized assistance from third parties (in particular academic ghostwriting. I have not used any other sources or aids than those indicated; this includes in particular software and services for language, text, and media production. In the event that the work is submitted in different formats (e.g. electronically, printed, plotted, on a data carrier), I declare that all the submitted versions are fully identical. I have not previously submitted this work, either in the same or a similar form to an examination body.

Aachen, 29.9.25

Ort, Datum/City, Date

C. Karl

Unterschrift/Signature

*Nichtzutreffendes bitte streichen/Please delete as appropriate

Belehrung:
Official Notification:

§ 156 StGB: Falsche Versicherung an Eides Statt
Wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.
§ 156 StGB (German Criminal Code): False Unsworn Declarations
Whosoever before a public authority competent to administer unsworn declarations (including Declarations of Academic Integrity) falsely submits such a declaration or falsely testifies while referring to such a declaration shall be liable to imprisonment for a term not exceeding three years or to a fine.
§ 161 StGB: Fahrlässiger Falscheid; fahrlässige falsche Versicherung an Eides Statt
(1) Wenn eine der in den §§ 154 bis 156 bezeichneten Handlungen aus Fahrlässigkeit begangen worden ist, so tritt Freiheitsstrafe bis zu einem Jahr oder Geldstrafe ein.
(2) Straflosigkeit tritt ein, wenn der Täter die falsche Angabe rechtzeitig berichtigt. Die Vorschriften des § 158 Abs. 2 und 3 gelten entsprechend.
§ 161 StGB (German Criminal Code): False Unsworn Declarations Due to Negligence
(1) If an individual commits one of the offenses listed in §§ 154 to 156 due to negligence, they are liable to imprisonment for a term not exceeding one year or to a fine.
(2) The offender shall be exempt from liability if they correct their false testimony in time. The provisions of § 158 (2) and (3) shall apply accordingly.

Die vorstehende Belehrung habe ich zur Kenntnis genommen:
I have read and understood the above official notification:

Aachen, 29.9.25

Ort, Datum/City, Date

C. Karl

Unterschrift/Signature

Contents

List of Figures and Tables

Abstract

Dark or Deceptive Patterns are design strategies meant to steer users towards actions or decisions that might not be in their best interests and have become increasingly prevalent in the online world in recent years. In order to mitigate their further expansion and protect users from their harmful influence, experts call for countermeasures. One possible solution could be the automatic removal of dark patterns from web pages. For this, recent research has started preliminary explorations into the utilization of large language models (LLMs). But while potential has been shown even without prior customizations, no further investigation has been made into adapting LLMs for this specific task.

In this thesis, we explore optimizing large language models to remove deceptive patterns from web pages without compromising the original page's design, layout, context, or functionality. We affected the model's performance by influencing four factors: Pre-training, Hyperparametesr, Prompting, and Fine-Tuning. Specifically, we compared the general-purpose model GPT-4o and reasoning model o4-mini, adjusted their hyperparameters, and prompted them using different prompting techniques. Finally, we used Supervised Fine-Tuning on a GPT-4o model using a dataset of 106 examples. In total, this resulted in 17 different conditions that were evaluated on a set of 25 predominantly real web pages and elements.

Our results show that fine-tuning a model, even on a rather small dataset, already strongly improves the model's performance for deceptive pattern removal. Further, we observed the potential of reasoning models, especially in combination with a customized prompt and high reasoning effort. Lastly, we identified *Few-Shot Chain-of-Thought Prompting* as the most effective out of our investigated prompting techniques.

Überblick

Dark oder Deceptive Patterns sind Designstrategien, die Nutzer zu Taten oder Entscheidungen, drängen, die nicht in ihrem Interesse sein könnten. Diese wurden in den letzten Jahren online immer häufiger. Um der weiteren Verbreitung Einhalt zu bieten und Nutzer vor deren schädlichen Einfluss zu schützen, fordern Experten Gegenmaßnahmen. Eine solche Lösung könnte die automatische Entfernung von Dark Patterns aus Webseiten sein. Kürzlich haben Forschende angefangen, die Nutzung von Large Language Models zu diesem Zweck zu untersuchen. Aber obwohl das Potenzial solcher Modelle sogar ohne vorherige Anpassungen gezeigt wurde, gab es bislang noch keine weiteren Untersuchungen zu der Konfiguration von Modellen für diese spezifische Aufgabe.

In dieser Arbeit erkunden wir die Optimierung von Large Language Models für die Entfernung von Deceptive Patterns aus Webseiten, ohne dass dabei Design, Aufbau, Kontext oder Funktionalität der ursprünglichen Seite beschädigt werden. Wir beeinflussen die Leistung des Modells in vier Aspekten: Vortraining, Hyperparameter, Prompting und Fine-Tuning. Hierfür verglichen wir ein General-Purpose-Modell GPT-4o mit dem Reasoning-Modell o4-mini und befragten diese unter Benutzung verschiedener Prompting-Strategien. Zuletzt nutzten wir noch Supervision-Fine-Tuning mit einem GPT-4o-Modell mit einem Datenset aus 106 Beispielen. Insgesamt ergaben diese Einstellungen 17 verschiedene Konfigurationen, die wir mit 25 primär echten Webseiten evaluiert haben.

Unsere Ergebnisse zeigen, dass das Fine-Tuning eines Modells auch mit einem relativ kleinen Datenset bereits eine starke Verbessung der Leistung bezüglich der Entfernung von Deceptive Patterns aus Webseiten erzeugt. Weiterhin haben wir das Potenzial von Reasoning-Modellen erkannt, vor allem unter Nutzung eines spezialisierten Prompts und mit hohem Reasoning-Aufwand. Zuletzt haben wir *Few-Shot Chain-of-Thought Prompting* als die effektivste Prompting-Strategie für unser Ziel identifiziert.

Acknowledgments

First, I want to thank Prof. Dr. Jan Borchers and Prof. Dr.-Ing. Ulrik Schroeder for examining this thesis.

Further, I'd like to thank René Schäfer for all the guidance and help during the work on this thesis. I really appreciate all the time and effort you spend on valuable advice and feedback.

I also want to thank all my coworkers who sat down with me for an entire afternoon and helped create the dataset and our administrator, Jan Erik, who helped me navigate the fine-tuning process and general model handling.

Finally, I want to thank my friends and family for their emotional support and general help over the last months.

Conventions

Throughout this thesis we use the following conventions:

- The thesis is written in American English.
- The first person is written in plural form.
- Unidentified third persons are described in plural form.

DEFINITION:
Definitions are set off in orange boxes.

EVALUATED PROMPTS:
Prompts we evaluated are set off in green boxes.

Evaluated Prompts

OTHER PROMPTS:
Prompts we otherwise used but did not evaluate are set off in a petrol box.

other Prompts

Names of deceptive patterns relevant for our thesis are written in SMALL CAPS.

Where appropriate, paragraphs are summarized by one or two sentences that are positioned at the margin of the page.

This is a summary of a paragraph.

We will use the terms *deceptive pattern*, *deceptive design* and *dark pattern* interchangeably.

Chapter 1

Introduction

"Deceptive patterns didn't appear overnight. Deception is part of being human – in fact, it's so common in the animal kingdom that we can even think of deception as a feature of life itself..."

—Harry Brignull

The vast majority of popular websites and apps employ manipulative tactics that go against users' best interests.

In 2022, a report commissioned by the European Council found that 97% of the most popular websites and apps in the European Union (EU), contained malicious user interface (UI) design. This is intended to manipulate users into making transactional decisions that might disadvantage them. Examples include people being forced to register an account, unfavorable settings being the default, or information being deliberately hidden or presented in such a way that people feel pressured towards a certain decision or action [Lupiáñez-Villanueva et al., 2022].

Research on these "dark patterns" was started in 2010 and has since featured various different terms and taxonomies.

These tactics are nothing new. In fact, many are based on similar practices that have been common in real-life sales for decades [Narayanan et al., 2020]. However, this digital form first started gaining attention in 2010 [Brignull, 2023; Conti and Sobiesk, 2010] and received the name "dark pattern" by user experience (UX) practitioner Harry Brignull the same year, who also developed an initial taxonomy to categorize 11 different kinds of dark patterns [Brignull, 2023].

We will use the terms *deceptive pattern*, *deceptive design* and *dark pattern* interchangeably.

Recently, some researchers have switched to the term *deceptive patterns* to avoid language with negative connotations [Brignull, 2023]. The field has also expanded upon Brignull's initial taxonomy and has introduced many overlapping but also distinct categorizations [e.g. Bösch et al., 2016; Gray et al., 2018; Mathur et al., 2021]. Last year, Gray et al. [2024] endeavored to combine the different taxonomies into one ontology, organizing these strategies into different levels and categories. In this thesis, we will refer to the following definition by Mathur et al. [2019] and the classification by Gray et al. [2024], which we will further elaborate on in Chapter 3.

Definition: *Deceptive Patterns*

DECEPTIVE PATTERNS:
User interface design choices that benefit an online service by coercing, steering, or deceiving users into making decisions that, if fully informed and capable of selecting alternatives, they might not make [Mathur et al., 2019].

Experts call for further countermeasures to help protect users from deceptive patterns

Apart from such classification research, the impact on and interaction of end-users and deceptive patterns has also been investigated [e.g. Bongard-Blanchy et al., 2021; Gray et al., 2021; Luguri and Strahilevitz, 2021]. As experts uncovered the prevalence of deception and manipulation online, the call for countermeasures and guarding users grew stronger. Especially as research suggests that educating users on the topic does not protect them from the harm of deceptive patterns [Bongard-Blanchy et al., 2021], the complete eradication of such practices might be the most effective solution so that users do not come into contact with them at all. Back in 2010, Brignull had hoped that we could achieve this by shaming the companies employing these patterns and educating on the topic, but the still ever-growing prevalence of deception online demonstrates how this approach is insufficient [Brignull, 2023]. And while legislation has aimed to regulate dark patterns, resulting regulations have been shown to lack in feasibility and enforcement [Krisam et al., 2021; Herman, 2024]. As the eradication of the source of manipulation cannot be expected in the foreseeable future, another area of deceptive pattern research is exploring intercepting at the client side before the

deception reaches the users themselves [e.g. Hasan Mansur et al., 2023; Kollnig et al., 2021; Nayak et al., 2024].

Past research on technical countermeasures focused on deceptive pattern detection and revealed a need for highly adaptable solutions.

Such technical solutions were also an area of intervention already suggested by Bongard-Blanchy et al. in 2021. However, so far these solutions have primarily focused on the detection of deceptive patterns [e.g. Mathur et al., 2019; Chen et al., 2024]. Additionally, research on visual countermeasures found high variance in user preferences and the need for different solutions for different kinds of deceptive patterns [Schäfer et al., 2023]. This, in addition to the steady emergence of new deceptive strategies, calls for a technical countermeasure with high flexibility: On one hand, it needs to adapt to new and unfamiliar deceptive patterns, while on the other hand, it must also be highly customizable regarding what is changed and how the result is presented to a user, matching their individual preferences. One technology capable of such high adaptability is large language models (LLMs) [Singh et al., 2025], which have gained attention in recent years.

LLMs have potential to facilitate DP removal, but are still underexplored in this area.

Some work explores the potential of LLMs for detecting and classifying deceptive patterns automatically [e.g. Babu et al., 2025; Mills and Whittle, 2023; Sazid et al., 2023], but the utilization of LLMs for removing deceptive patterns completely is largely unexplored. Schäfer et al. [2025] began investigating this and found promising potential of using LLMs for deceptive pattern removal. In particular, they showed how adding guardrails to the prompt strongly improved the amount of manipulation removed. Nonetheless, they noted problems of the model hallucinating facts, removing information, or struggling with specific deceptive patterns. But as the addition of the guardrails already improved results, further adjustments of the LLM could remedy those problems as well. Base model choice [Seßler et al., 2024], prompt-engineering [Brown et al., 2020], and dataset fine-tuning [Lomshakov et al., 2023] are examples of factors that can further improve task performance and have been shown to be critical when dealing with deception [Boumber et al., 2024], but were not yet sufficiently addressed in previous research for deceptive pattern removal.

The topic of this thesis will be customizing an LLM for deceptive pattern removal

This thesis will explore the utilization of large language models in the fight against deceptive patterns. Specifically, we will examine how a large language model can be customized for the removal of deceptive patterns so that a user does not have to engage with them at all. We focus on how the model choice, prompting techniques, and fine-tuning influence the results. In particular, the research questions (RQ) that will be answered in this thesis are

RQ 1: Can a customized LLM remove deception from web pages without compromising design, layout, context, or functionality?

RQ 2: Which prompting techniques are most suitable for this task?

RQ 3: How does fine-tuning the model to a task-specific dataset influence the results?

RQ 4: What challenges and advantages does a reasoning model present in comparison to a general-purpose model?

1.1 Outline

In Chapter 2, we will provide necessary background information on large language models and prompt engineering.

Chapter 3 focuses on related research on the topics of deceptive patterns and the usage of large language models with web pages.

The main work of this thesis will be described in Chapter 4, where we will describe the customization of our LLMs in detail, explain the reasoning behind model choice, hyperparameter settings, and prompt engineering, and outline the fine-tuning process.

The results and comparison of different techniques will be presented in Chapter 5 and discussed in Chapter 6.

Finally, Chapter 7 will present a summary of the thesis.

Chapter 2

Background

In this chapter, we will provide the necessary technical background regarding large language models (LLMs) and prompt engineering. As this thesis will utilize OpenAI[1] models accessed through Azure OpenAI[2] we focus on their supported features.

2.1 Large Language Models

Large Language models are computational systems that output natural language based on a prompt using next token prediction.

Language Models are computational systems capable of natural language processing (NLP) that generate natural language output by sequentially predicting the next word or token. They use neural networks and the surrounding context to produce a probability distribution for possible candidates that the next token is sampled from [Jurafsky and Martin, 2025]. Recently, we've seen the emergence of *Large* Language Models that consist of several billion parameters, which are set by pre-training the model on large amounts of textual data [Brown et al., 2020].

[1] `https://openai.com/`, *last accessed September 17, 2025*

[2] `https://azure.microsoft.com/de-de/products/ai-foundry/models/openai`, *last accessed September 17, 2025*

2.1.1 Models offered by OpenAI

OpenAI offers *general-purpose* and *reasoning models*

General-purpose models like the GPT-series of OpenAI are models for a wide range of tasks[3]. They are capable of handling multi-modal inputs and support fine-tuning, as well as a variety of different tools like web search or computer use [OpenAI et al., 2024]. Additionally, OpenAI also offers models that are specialized for more complex tasks: *reasoning models*.

Reasoning models are capable of more logically complex tasks by reasoning over their answers.

In contrast to the general-purpose models, reasoning models like OpenAI's o-series and more recently GPT-5 were trained to reason over their answers. This means that, similar to a human's stream of consciousness, the models break down their thinking process internally in a *chain-of-thought* [OpenAI et al., 2024]. This makes the models better at complex problem-solving tasks, scientific reasoning, and coding[4]. However, these results come at higher costs and lower latency[5]. Further, the internal reasoning is not directly accessible to users due to concerns regarding feasibility and competitive advantage[6]. It is possible to have the model generate a summary of the reasoning process, but this requires a direct OpenAI subscription. Unfortunately, as we accessed our models via AzureOpenAI, this service was unavailable for our work.

2.2 Token Management

LLMs operate on *tokens*.

Large Language models do not handle language like humans, but instead operate on tokens that represent different words or word parts. Due to this, natural lan-

[3] `https://techcommunity.microsoft.com/blog/azure-ai-foundry-blog/general-purpose-vs-reasoning-models-in-azure-openai/4403091`, *last accessed 18 September, 2025*

[4] `https://platform.openai.com/docs/guides/reasoning`, *last accessed September 25, 2025*

[5] `https://techcommunity.microsoft.com/blog/azure-ai-foundry-blog/general-purpose-vs-reasoning-models-in-azure-openai/4403091`, *last accessed September 25, 2025*

[6] `https://openai.com/index/learning-to-reason-with-llms/`, *last accessed September 9, 2025*

guage input first has to be parsed into tokens [Webster and Kit, 1992]. With OpenAI's tokenization technique, one token corresponds to roughly four characters of English language text[7].

Overall, a model can only consider a limited amount of tokens at the same time for the next token prediction [Jurafsky and Martin, 2025]. The size of this *context window* depends on the model itself. GPT-4o has a window size of 128,000 tokens[8], while most reasoning models of the o-series can handle up to 200,000 tokens[9]. Additionally, input and output tokens are not only limited by the available budget, but the latter is also capped at 16,384 for GPT-4o and 100,000 for reasoning models. Reasoning models also produce *reasoning tokens* that are not part of the output but still charged as such and use up space in the context window[10].

The information considered by the models is restricted by the *context window*.

2.2.1 Structured Output

Due to these limits, it might be necessary to restrict the output of an LLM to a desired structure. *Structured Outputs*[11] is a tool capable of this. It uses a constrained decoding approach to determine the validity of next tokens in order to create responses in a valid JSON format. Structured output also facilitates the prompting process, as the structure no longer has to be heavily stressed in the prompt and also allows for automatic detection and handling of cases where the model refuses to answer.

Structured Output restricts the responses format to a desired schema.

[7] `https://platform.openai.com/tokenizer`, *last accessed September 25, 2025*

[8] `https://platform.openai.com/docs/models/gpt-4o`, *last accessed September 25, 2025*

[9] `https://platform.openai.com/docs/models/o4-mini`, *last accessed September 25, 2025*

[10] `https://platform.openai.com/docs/guides/reasoning`, *last accessed 25 September, 2025*

[11] `https://openai.com/index/introducing-structured-outputs-in-the-api/`, *last accessed September 9, 2025*

This desired output format can either be provided directly in a JSON schema or supplied as an object from supported extensions like Pydantic[12] or Zod[13] .

2.3 Hyperparameters

In this section, we refer to information out of prompting guides by Learn Guides[14] and the Prompt Engineering Guide,[15] as well as OpenAI's API reference[16] .

It is possible to set different *hyperparameters* that influence the model's response process and next token prediction.

Repetitions in answers can be penalized with `frequency_penalty` and `presence_penalty`.

The `frequency_penalty` and `presence_penalty` restrict the reoccurrence of tokens, reducing the likelihood of repetitions in the model's answer. This is especially useful for creative writing tasks, but counterproductive for code generation, as languages usually feature important keywords that are often repeated. Both parameters can be set to a value between -2 and 2. In this case, negative values encourage repetitions, 0 is the default and does not affect anything, while positive values apply a penalty.

The parameters `temperature` and `top_p` affect the determinism of the next token prediction.

The parameters `temperature` and `top_p` both influence the sampling of the next tokens. The former affects the probability distribution for the selection from next token candidates. A low `temperature` value widens the gaps between higher and lower probabilities, further decreasing the chance for tokens with lower probabilities to be chosen and thereby increasing the determinism of the response. With a `temperature` of 0, the next token is always guaranteed to be the one with the highest probabilistic value. By setting `top_p` instead of `temperature`, sampling the next

[12] `https://docs.pydantic.dev/`, *last accessed September 25, 2025*
[13] `https://zod.dev/`, *last accessed September 25, 2025*
[14] `https://learnprompting.org/blog/llm-parameters`, *last accessed September 25, 2025*
[15] `https://www.promptingguide.ai/introduction/settings`, *last accessed September 25, 2025*
[16] `https://platform.openai.com/docs/api-reference/`, *last accessed September 25, 2025*

Use Case	Temperature	Top_p	Description
Code Generation	0.2	0.1	Generates code that adheres to established patterns and conventions. Output is more deterministic and focused. Useful for generating syntactically correct code.
Creative Writing	0.7	0.8	Generates creative and diverse text for storytelling. Output is more exploratory and less constrained by patterns.
Chatbot Responses	0.5	0.5	Generates conversational responses that balance coherence and diversity. Output is more natural and engaging.
Code Comment Generation	0.3	0.2	Generates code comments that are more likely to be concise and relevant. Output is more deterministic and adheres to conventions.
Data Analysis Scripting	0.2	0.1	Generates data analysis scripts that are more likely to be correct and efficient. Output is more deterministic and focused.
Exploratory Code Writing	0.6	0.7	Generates code that explores alternative solutions and creative approaches. Output is less constrained by established patterns.

Figure 2.1: Recommendations for `temperature` and `top_p` values based on use cases. Low values are recommended for the task of code generation.

token is randomly selected from a set of candidates. The probability threshold for which tokens are included in this set of candidates is determined by the value of `top_p`. In conclusion, both parameters influence how deterministic and, thereby, how creative the answer is. The main difference is that a low `temperature` only decreases the chance that a token with a low probabilistic value is chosen. Low `top_p` values completely eliminated such tokens from its set of candidates, making them impossible to be chosen. When using LLMs, only one of both parameters should be set. `Temperature` ranges between 0 and 2 and `top_p` requires a value between 0 and 1. Figure 2.1 contains recommended values for `temperature` and `top_p` depending on the use case[17]. For code generation, a low `temperature` of 0.2 and a low `top_p` value of 0.1 is recommended.

For code generation a `temperature` of 0.2 or a `top_p` value of 0.1 is recommended.

[17] `https://community.openai.com/t/cheat-sheet-mastering-temperature-and-top-p-in-chatgpt-api/172683`, *last accessed September 30, 2025*

The reasoning process of reasoning models can be deepened by setting `reasoning_effort`.

`Reasoning_effort` is only supported by reasoning models and sets how much time the model will spend processing the request, resulting in potentially higher amounts of reasoning tokens being used. This parameter can be set to `low`, `medium`, or `high`. Depending on the model, `minimal` reasoning effort is also supported.

Lastly, parameters like `max_tokens` or `stop_sequence` restrict the length of the LLM's answers, either depending on the number of output tokens or based on a specific stop sequence. This can help keep costs low or stop the LLM from digressing.

2.4 Prompt Engineering

Prompt engineering is an iterative process in order to achieve a prompt that gets a model to consistently output desired responses.

Prompt Engineering describes the process of writing a prompt and adjusting it to get the model to consistently return responses with sufficient quality. However, as the model's output is almost always non-deterministic, achieving such a prompt is often a process of trial and error and a "mix of art and science"[18].

Nonetheless, best practices have been established, and research has found some prompting techniques that can assist in writing effective prompts. However, it is important to note that these recommendations differ significantly depending on whether a general-purpose or a reasoning model is being prompted.

[18] `https://platform.openai.com/docs/guides/prompt-engineering`, *last accessed September 28, 2025*

2.4.1 Prompting a General-Purpose Model

Best Practices

The following general best practices are adapted from an article[19] and guide[20] by OpenAI and Digital Ocean.

General-purpose models should receive instructions that are as precise as possible, with the specific context, steps, and expectations broken down in detail.

Firstly, the general-purpose model needs to properly understand the circumstances. For this, the prompt should inform the model about the context of the task and what is expected of the model. The task itself should be described in detail and broken down into steps with specific instructions. For possible errors and edge cases, the LLM should not simply be informed about what not to do without further context, but instead be told what to do instead. Finally, the prompt should also include precise instructions about how the output should be formatted as well as other form details, such as length and style.

Apart from this, the prompt can be further improved by the use of different prompting techniques, which can also be combined.

Prompting Techniques

There have been various techniques for prompting proposed and investigated in related research. We will only explain the most common ones, that are relevant for this thesis.

[19] `https://help.openai.com/en/articles/6654000-best-practices-for-prompt-engineering-with-the-openai-api`, *last accessed September 25, 2025*

[20] `https://cookbook.openai.com/examples/gpt4-1_prompting_guide`, *last accessed September 25, 2025*

Zero-, One- and Few-Shot Prompting

One- or Few-Shot Prompting provides the LLM with example tasks and ideal answers.

Firstly, the performance of a large language model can be improved by providing it with task examples and "ideal" answers. Depending on how many examples were used, this is referred to as *One-, Two-* or *Few-Shot Prompting*, respectively. A model's performance being improved by providing it with examples is also sometimes called *in context learning* [Dong et al., 2024] and has been shown to improve a model's performance significantly [Brown et al., 2020]. In addition to better overall performance, these examples also help define and convey a structure that the LLM should conform to with its output.

Initial explorations or proof of capability efforts often use *Zero-Shot* prompts without any examples.

But while multiple examples improve the results, they also require effort to construct and take up input tokens and thereby space in the context window. *Zero-Shot Prompting* without any examples can also often be found in related work for comparison with *Few-Shot Prompting* or to demonstrate the general capability of LLMs even without further specifications [e.g. Kojima et al., 2022; Schäfer et al., 2025].

Chain-of-Thought Prompting

Chain-of-Thought Prompting enables the LLM to perform reasoning for complex tasks, leading to fewer mistakes

Wei et al. [2023] found that getting an LLM to generate a *chain-of-thought* increases performance in complex tasks. This is similar to human thinking, where complicated tasks are broken down into smaller intermediate steps and then handled sequentially. A simple example they investigated was the following math problem: *"The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?"* Instead of the usual answer of a model, that would only contain the answer, in *Chain-of-Thought* prompted models, the answers include their thought process: *"The cafeteria had 23 apples originally. They used 20 to make lunch. So they had 23 - 20 = 3. They bought 6 more apples, so they have 3 + 6 = 9. The answer is 9."*

Chain-of-Thought Prompting allows insight into where a model went wrong.

Apart from a model making fewer mistakes, with chain-of-thought output, another advantage is offered by the insight into the "thoughts" and behavior of the model. This makes

the output more interpretable and helps understand how the model reached its outcome and, if necessary, where it went wrong and why.

Wei et al. [2023] elicited with chain-of-thought by prompting with examples that also include this. However, this behavior can also be reached in *Zero-Shot* Prompts by adding simple instructions like "Let's think step by step"[Kojima et al., 2022].

Chain-of-Thought behavior can be triggered by *Few-Shot* examples or *Zero-Shot* instructions.

The idea of having models produce a chain-of-thought for better performance is the motivation behind reasoning models.

Persona Prompts

Persona or *Role Prompting* assigns a specific persona to the model. This helps with getting the model's response to adhere to a specific style and tone, but also increases performance [Kong et al., 2024]. Despite this, this technique is rather controversial, with some research even observing worse results when using personas [Zheng et al., 2024]. However, Shin et al. [2025] found *Persona Prompting* to be a useful technique for UX evaluation, especially in combination with *Chain-of-Thought Prompting*.

Persona Prompting assigns a persona or role to the LLM. Results of research are split on whether this improves performance.

Prompt Chaining

Prompt Chaining or sometimes also called *Least-To-Most Prompting* [Zhou et al., 2023] describes the approach of splitting the overarching task into separate smaller subtasks and then solving them sequentially with individual prompts, where the answer of the previous step is used as input for the next one.

Prompt Chaining splits a task into steps, that are each addressed with their own prompt.

2.4.2 Prompting a Reasoning Model

Reasoning models work best with prompts specifying the desired outcome without specific instructions.

The approach to prompting a reasoning model differs substantially from the best practices of prompting general-purpose models. The latter profit from very precise instructions and breaking the task down into the specific steps, while reasoning models work best if the prompt specifies the desired result and lets the model decide on the approach.

The previous prompting techniques are not recommended for reasoning models.

For this specific reason, specifying and splitting the individual steps with the *Prompt Chaining* technique is not helpful. *Chain-of-Thought Prompting* is also not necessary as this idea is already integrated into the model itself. However, Nori et al. [2024] have shown that explicitly instructing the model to extend the reasoning process led to higher performance. On the other hand, they also showed the opposite effect when examples were included for *Few-Shot Prompting*.

2.4.3 Meta Prompting

Meta Prompting uses one LLM to create an optimized prompt for another model based on a task description.

Meta Prompting is a technique in which the prompt for a task is written by another model. For this, a developer gives task context and information to an LLM, which then generates an effective prompt that could be used by another model to solve the original task [Suzgun and Kalai, 2024]. It might be especially useful to use a different model, like a reasoning model, for creating the meta prompt, than the one that will receive the more task prompt [21]. OpenAI have published a meta prompt[22] that can be used to create and edit optimized prompts for various objectives by simply adding the task description.

[21] `https://cookbook.openai.com/examples/enhance_your_prompts_with_meta_prompting`, *last accessed September 28, 2025*

[22] `https://platform.openai.com/docs/guides/prompt-generation?context=text-out`, *last accessed September 28, 2025*

2.5 Dataset Fine-Tuning

Fine-Tuning adjusts the internal parameters of an LLM based on a labeled dataset.

Dataset Fine-Tuning or simply *Fine-tuning* describes the process of adjusting the weights and biases of a large language model based on a task-specific dataset. For this process, labeled data is required, the form of which depends on the used technique. This dataset is first split into different subsets: One for training and one for evaluating the model's performance after the fine-tuning process is complete. Optionally, one can also split off a validation set to monitor the performance development during the fine-tuning process.

OpenAI supports three different fine-tuning processes[23]: *Supervised Fine-Tuning*, *Direct Preference Optimization*, and *Reinforcement Fine-Tuning*.

Supervised Fine-tuning and *Direct Preferences Optimization* adjust based on provided target output.

Supervised Fine-tuning uses a labeled dataset consisting of input and the matching ideal output, a "ground truth". The parameters of the models are then adjusted so that the model returns answers matching the ground truth of the given input. In *Direct Preferences Optimization* the dataset does not only contain a desired target output, but also an example of a non-preferred output. The model is then trained not only to produce responses similar to the target output, but also to avoid ones like the non-preferred output. This helps the model adjust to subjective preferences and helps in guiding its focus in specific tasks.

Reinforcement Fine-Tuning uses a rewarding systems with a grading script or another LLM acting as a judge.

Lastly, *Reinforcement Fine-Tuning* uses a grading system to iteratively reward the model based on its performance, which allows for optimization of more complex tasks. It requires an "expert" scorer, which could be a script for simple tasks or another LLM acting as a judge, that evaluates the performance.

The different fine-tuning processes are supported by different model types. *Supervision Fine-Tuning* and *Direct Preferences Optimization* are only supported by general-purpose models, while *Reinforcement Fine-Tuning* is meant for reasoning models.

[23] `https://learn.microsoft.com/en-us/azure/ai-foundry/concepts/fine-tuning-overview`, *last accessed 13 September, 2025*

Chapter 3

Related Work

In this chapter, we will address related work on dark or deceptive patterns and research on large language models and their interaction with websites or deceptive patterns.

3.1 Deceptive Patterns

Since the emergence of the term in 2010, dark patterns have gained increasing attention from researchers in various areas and disciplines [Mathur et al., 2021].

3.1.1 Prevalence and Harm

Deceptive Patterns are prevalent on web pages, mobile apps, and are also further distributed by third-party services.

Deceptive Patterns have been shown to be prevalent across web pages and apps. A study by Lupiáñez-Villanueva et al. [2022] was commissioned by the European Council to investigate unfair commercial practices online. They found that 97% of the most popular websites and apps in the EU contained at least one dark pattern. Moreover, they noted that this did not depend on their size, but was also true for smaller businesses. The most commonly found deceptive patterns included HIDDEN INFORMATION/FALSE HIERARCHY, PRESELECTION, NAGGING, DIFFICULT CANCELLATION and FORCED REGISTRATION, but they also ob-

served how deceptive patterns most commonly occurred in combination. Mathur et al. [2019] and Di Geronimo et al. [2020] reinforced these findings by reporting on similar observations for shopping web pages and mobile apps, respectively. Mathur et al. [2019] also discovered 22 third-party services that use deceptive patterns in their products. These are then included in other websites, further extending their prevalence. In addition to these contexts, other studies by Zagal et al. [2013] and Sousa and Oliveira [2023] have discovered various deceptive patterns in games and even in those designed for young children.

Being exposed to deceptive patterns is accompanied by users experiencing negative consequences like negative emotions or financial harm.

Various studies have investigated the consequences of users being exposed to deceptive patterns. Those could include negative emotions like annoyance and frustration, loss of autonomy and control or financial and privacy harm [Gunawan et al., 2022; Lupiáñez-Villanueva et al., 2022]. They also have negative consequences for the services employing deceptive patterns, like the users losing trust in them [Gray et al., 2021]. Although regarding the latter, Lupiáñez-Villanueva et al. [2022] found that users were generally more forgiving towards larger companies employing such practices if the service they provided had a generally higher usefulness.

Users are unaware of the issue of deceptive patterns and struggle with identifying them and resisting their influence.

Despite experiencing negative backlash from being confronted with deceptive patterns, users are generally unaware of the issue [e.g. Bongard-Blanchy et al., 2021; Lupiáñez-Villanueva et al., 2022] and unable to recognize deceptive patterns. This finding has been observed by both Lupiáñez-Villanueva et al. [2022] and Di Geronimo et al. [2020], though the latter discovered that the performance can be improved by educating users. However, Bongard-Blanchy et al. [2021] observed how users recognizing deception does not necessarily shield them from its influence. There is also the additional challenge that deceptive patterns frequently function in a "gray area" between legitimate persuasion and illegitimate manipulation, making them harder to recognize or outlaw [Lupiáñez-Villanueva et al., 2022].

3.1.2 Classification

As deceptive patterns appear in different forms, many researchers have undertaken the effort of identifying and classifying them, resulting in various taxonomies that have also been based on different approaches.

Various different taxonomies have been proposed to define different deceptive pattern categories.

Similar to Brignull's initial taxonomy[1], various researchers have undertaken defining different categories of deceptive patterns [e.g. Bösch et al., 2016; Gray et al., 2018]. Some, however, have taken approaches different from defining specific instances of deceptive patterns, but instead focused on other viewpoints or characteristics. Mathur et al. [2021] have based their categorization on six different attributes describing deceptive patterns: Asymmetric, Restrictive, Disparate Treatment, Covert, Deceptive, and Information Hiding. Luguri and Strahilevitz [2021] have made their distinction between *mild* and *aggressive* dark patterns.

Context is an important factor to distinguish between deceptive patterns and design patterns that benefit users.

With their ongoing work, Lewis and Vassileva [2024] strive to establish a taxonomy-independent evaluation process for the identification and description of deceptive patterns by analyzing previously identified patterns for their properties, consequences, and contexts of application. They highlighted how it is important to take the purpose of an interface into account, which outcome it seeks, and which resources of the users the interface tries to acquire. This is demonstrated by the example of OBSTRUCTION: Used as a friction design, obstructing the user's interaction might protect them from making grave errors like accidental deletions. However, OBSTRUCTION applied to processes like cancellations or privacy setting management is harmful to users and is, therefore, a deceptive pattern.

Dark patterns can also function over time and multiple pages.

Gray et al. [2025] have found that deceptive patterns also occur overtime and over multiple pages and steps. Due to this they have created a temporal distinction between *intra-page*, *inter-page* and *system temporality* deceptive patterns.

[1] `https://old.deceptive.design/`, *last accessed September 27, 2025*

In this thesis, we refer to the ontology by Gray et al. which defines and organizes 65 types of deceptive patterns over 3 levels.

The most important classification for this thesis, however, is the ontology by Gray et al. [2024], which structures deceptive patterns from other established taxonomies into three levels. *High-level* patterns describe the most abstract level, which represents the general underlying strategies. This includes OBSTRUCTION, SNEAKING, INTERFACE INTERFERENCE, FORCED ACTION and SOCIAL ENGINEERING. These five categories each feature several other deceptive patterns on the *Meso-* and *Low-levels*. The former of which describes an angle of attack, while the lowest level specifies the means of execution. In total, the ontology includes definitions for 65 types of dark patterns. We will use this ontology to refer to deceptive pattern types in our work. The definitions of patterns that we encountered can be found in Appendix C.

3.1.3 Countermeasures

Bongard-Blanchy et al. have proposed four areas of intervention: Education, Design, Technology, and Regulation.

Another area of research regarding deceptive patterns is concerned with countering them. To this end, Bongard-Blanchy et al. [2021] proposed four different areas of intervention: educational, design, technical, and regulatory measures. These can help counter deceptive patterns through awareness, detection, resistance, and elimination.

Strengthening user awareness does not necessarily help them resist deceptive patterns.

While multiple studies found that users struggle with identifying deceptive patterns [Lupiáñez-Villanueva et al., 2022], Di Geronimo et al. [2020] found that educating users on deceptive patterns can increase their detection performance. On the other side, Bongard-Blanchy et al. [2021] found that users being aware of deceptive patterns does not necessarily enable them to resist their influence.

Regulatory interventions exist, but are insufficiently enforced.

While regulatory efforts have been made to outlaw these practices that are harmful to consumers like the General Data Protection Regulation (GDPR) in the EU [Gray et al., 2021], several studies have investigated their realization and have found insufficient compliance with these laws combined with a lack of enforcement [Krisam et al., 2021; Leiser and Santos, 2024; Nouwens et al., 2020].

Design Interventions include efforts towards better design practices that would counter dark patterns. Potel-Saville and Da Rocha presented a shift to a problem-solving approach with a taxonomy of *fair* patterns, which do not manipulate the user in any way and comply with legal and regulatory frameworks. More specifically, they propose a taxonomy of seven fair category counterparts to deceptive patterns. This taxonomy is more robust against future deceptive patterns as it is based on the cognitive biases exploited by deceptive patterns. Another step further utilizes *bright patterns*, persuasive design that utilizes similar approaches as dark patterns, but instead nudge user towards actions or decisions in their interests [Graßl et al., 2021].

Fair patterns establish an alternative practice to counter deceptive patterns.

Bright patterns also manipulate users, but towards user-friendly actions or decisions.

The last area of interventions contains technical countermeasures that aim to help users automatically. However, regarding such measures, it has to be considered whether they are actually feasible and, if so, what they should look like.

Regarding the former, Stavrakakis et al. [2021] have established a framework of deceptive patterns based on whether and how they could be automatically detected. For this, Stavrakakis et al. conducted brainstorming sessions using images from over 100 websites to categorize different deceptive patterns. The categorization was based on whether they are fully detectable, partially detectable, or undetectable and whether this would be achievable automatically or would have to be done manually. In case of the latter, a system supports the process, but ultimately a human would have to judge whether a certain design is deceptive. As a result, they determined that MISDIRECTION, CONFIRMSHAMING, FORCED CONTINUITY, PRIVACY ZUCKERING and BAIT AND SWITCH could not be automatically detected as their implementation varies too much. Overall, they also noted how automatic detection might be hindered by some websites blocking web scraping. Additionally, some patterns generally require human judgment to determine whether they are deceptive, for instance, the PRESELECTION of an option.

Certain deceptive pattern types are impossible to detect automatically.

User preferences on visual countermeasures vary between individuals.

Schäfer et al. [2023] further explored the presentation of countermeasures by investigating user preferences for six different visual countermeasures against the dark patterns CONFIRMSHAMING, LOW-STOCK MESSAGE and VISUAL INTERFERENCE. In their work, they found a strong variance of preferences depending on individuals and on the dark pattern being countered. Users also expressed two possibly conflicting preferences: On one hand, they did not want the interface to be cluttered with additional elements. On the other hand, they do not want silent changes to be applied without their supervision.

GreaseDroid is a community-driven countermeasure approach where users create and share neutralizations of deceptive patterns as app add-ons.

Looking into explicit realizations of countermeasures, Kollnig et al. [2021] proposed *GreaseDroid*, a community-managed app modification framework for automatic deceptive pattern removal. For this, experts would develop patches for interfaces. Those patches could either remove deceptive patterns on the interface like INTERFACE INTERFERENCE or modify the control flow for deceptive patterns like NAGGING, FORCED ACTION, OBSTRUCTION or SNEAKING. Users would then be able to activate such patches for specific apps. In a case study, they demonstrated with examples from the app *Twitter* that adapting UI components is rather straightforward, but changing the control flow proved to be more challenging due to the source code being hard to navigate.

When using AI for technical interventions, the choice of correct representation of the deceptive interface to the model is a challenge.

As another form of technical intervention, Soe et al. [2022] investigated the use of supervised machine learning for dark pattern detection and classification in cookie banners. Their approach was rather naive and mainly used to identify challenges for this task, like the challenge of representing deceptive patterns to an AI. This could, for example, be done through images, but this would cause problems for deceptive patterns that occur as events, such as NAGGING. Textual input could easily detect deceptive patterns based on linguistic choices such as CONFIRMSHAMING, but it would not be possible to detect visual cues, like the use of different colors. The last option would be to pass a set of characteristic interface features to the model and collecting values of those features when deceptive patterns are included. However, as the latter is a highly time-consuming process Soe et al. propose that future work could identify a

set of features that can be used to distinguish between non-problematic and deceptive design but also be extracted automatically.

3.2 Large Language Models

Research regarding large language models has seen a rapid rise in the past years [Naveed et al., 2025]. We will only introduce relevant research on LLM and website interaction, deception, and dark patterns.

3.2.1 Interaction with Websites

Various studies have investigated LLMs' capabilities to write and comment code in various languages [e.g. Ahmed and Devanbu, 2023; **?**] according to natural language instructions, and, albeit more rare, there have also been some studies where LLMs interacted with HTML code or websites.

Pre-Training on natural language data helps with HTML understanding.

Gur et al. [2023] came up with 3 different HTML-related baseline tasks (Autonomous Web Navigation, Semantic Classification, and Description Generation) and compared the performance of different LLMs varying in architecture, size, and training. They found that LLMs that were pre-trained on natural language data could transfer this into better performance on HTML understanding tasks, even with relatively little further expert fine-tuning. However, a big restriction is imposed by the size of the context window, which acts as a bottleneck for LLMs' performance for HTML understanding.

LLMs are capable of UX evaluation of websites similar to humans, especially when using Persona and Chain-of-Thought Prompting

In their study Shin et al. [2025] tested the capability of LLMs to evaluate UX design. For this, they prompted ChatGPT-4 with five different prompting techniques (*Zero-Shot Prompting*, *Role Prompting*, *(Zero-Shot) Chain-of-Thought Prompting*, *Self-Refine Prompting*, and *Least-to-Most Prompting*) to evaluate the UX of a shopping platform based on screenshots. For the prompting techniques, they found that

Persona Prompting and *Zero-shot Chain-of-Thought Prompting* were highly effective and achieved the best results when combined. They also noted the strength of LLMs in analyzing large data in detail compared to human evaluators, noting how the LLMs were able to identify tiny details humans overlooked. On the other hand, the LLM failed to properly comprehend the context appropriately, which is intuitive to humans. Instead, the LLM was overly critical of tiny changes that human evaluators would consider good enough.

LLMs can actively change the HTML code of existing web pages based on user requests, but hallucinate components, attributes, or functions to the detriment of page layout and functionality

Li et al. [2023] used LLMs to actively modify the User Interface (UI) design of a web page based on natural language input. For this, they compared Text Completion, Code Completion, and Code Edit endpoints to change the source code. Due to token limits, they had to split the existing source code and prompt the model with it sequentially. They found that even the text completion endpoint was capable of handling the HTML code, despite not being optimized for code. Overall, they observed that the LLM performed best when prompted with simple requests regarding color and size. However, they noted how it often hallucinated variable names or made assumptions about existing elements or functionalities, especially when it needed to manipulate components that are not included in the current chunk. They also noted problems with unclear requests: For simple websites, this resulted in little to no visible changes, but with complicated pages, the LLM hallucinated nonsensical additional components, cluttering the layout. If the model was unsure what to do, it might output nonsensical components. Additionally, with complex components, the LLM sometimes returned visually similar copies, but did not replicate the original functionality, making the component unusable.

3.2.2 LLMs and Deception

Research has also concerned itself with large language models and deception.

Boumber et al. [2024] investigated the ability of LLMs to recognize deception in text from different domains such as political polls, job listings, fishing mails, or SMS. For this, they used a model in combination with *Chain-of-Thought Prompting* and a Retrieval-Augmented Generation (RAG) framework, that allowed the LLM to access further context and information for in-context learning. Overall, their approach showed improvement over previous work, which affirmed the importance of model selection and adaptation. They also stressed the significant influence of the provided definition of what characterizes deception on the model's performance. However, they found that the models' understanding struggles with more complex forms of communication like irony or sarcasm.

LLMs are capable of deception detection, but the performance is strongly influenced by the provided definition of deception.

On the other hand, Kran et al. [2024] and Wolfe and Hiniker [2024] have found deception being caused by the LLMs themselves. Kran et al. [2024] established six categories of deceptive patterns found in the answers of LLMs: BRAND BIAS, USER RETENTION, SYCOPHANCY, ANTHROPOMORPHISM, HARMFUL GENERATION, and SNEAKING. They found that some models are explicitly designed to favor their developers' products and exhibit untruthful communication, among other manipulative behaviors. Further, Wolfe and Hiniker [2024] investigated a new deceptive pattern they called EXPERTISE FOG, which describes how an interface presents itself as an expert without any evidence of such expertise. They encountered this deceptive design in GPT models on OpenAI's GPT store.

LLMs can be deceptive in their answers or presentation.

3.2.3 LLMs and Dark Patterns

LLMs have also been shown to employ deceptive patterns when prompted to generate web elements.

Krauß et al. [2025] examined GPT-4 and whether a user can unintentionally create deceptive designs when using a model for front-end development. For this, they had participants ask ChatGPT to generate web elements and modify them according to neutral business goals, such as increasing the sales of a specific featured product or the number

LLMs generate web elements that contain deceptive patterns, even when they are not specifically prompted to do so.

of sign-ups for a newsletter. They found that all 20 generated web pages contained at least one deceptive pattern, with a maximum of nine and a mean of five patterns per page. They especially found that the generated designs violate EU law and even OpenAI's usage policies.
In a similar study Chen et al. [2025] used 4 well-known LLMs to generate web components commonly found on e-commerce websites and found deceptive patterns in over one-third of the generated components. They found that the number of employed dark patterns varied depending on which model was used, which component was generated, and whose interests were instructed to be prioritized: The company's, the user's, or none. Similar to the results by Krauß et al., dark patterns are also more frequently produced in components that are related to company interests and less frequently generated when users' interests were prioritized. The most commonly used deceptive patterns hid information or restricted actions.

Despite this, LLMs can also be helpful in the fight against deceptive patterns.

Providing an LLM with screenshots is the most promising approach of representing web pages for simulating user behavior in deceptive interfaces.

Mills and Whittle [2023] investigated using generative AI to simulate the behavior of users with differing digital skills in order to detect deceptive patterns. For this, they came up with three different approaches where the LLM is assigned a persona with different digital literacy levels and asked to navigate a web page depending on either user description, an image of the interface, or by providing the model with the HTML and JavaScript code. They noted problems of these approaches, namely the dependence of the first approach on the description by a user. The second approach of using AI vision and images was seen as the most promising method for identifying dark patterns and simulating the behavior of a person resulting from deception and the interface design. Finally, while the last approach is the most objective, as the input is directly dependent on the website, the model struggled with interpreting the code as a rendered web page, and overly focused on links over visuals.

LLMs can be used to enforce user preferences, mitigating deception.

Porcelli et al. [2024] used LLMs to navigate deceptive cookie banners and automatically enforce user preferences. They proposed a *User Privacy Management System (UPPMs)* that helps users create a personal privacy policy, which is then automatically applied to cookie banners on visited websites by using an LLM. Experiments showed how analyzing the HTML code enabled the model to understand cookie banners and apply even complicated privacy policies.

LLMs can help detect deceptive patterns, but struggle with SNEAKING.

Further, multiple studies have investigated the use of LLMs for the automatic identification and classification of deceptive patterns. Sazid et al. [2023] explored the identification of texts containing deceptive patterns. They achieved an overall accuracy of 83.57% by using zero-, one-, and few-shot prompting for each deceptive pattern category. While this worked well for deceptive patterns like FORCED ACTION, SCARCITY, SOCIAL PROOF, or non-deceptive designs, the model performed poorly for patterns like MISDIRECTION and OBSTRUCTION. Most notably, SNEAKING was never identified. There have also been further studies that used multi-modal LLMs in combination with images for automatic detection of deceptive patterns [Chen et al., 2023; Kodmurgi et al., 2024; Nayak et al., 2024].

GPT-4o is generally capable of deceptive pattern removal, even without further adjustments.

Finally, Schäfer et al. [2025] already investigated the use of LLMs for deceptive pattern removal. For this, they used a GPT-4o model without any further model adjustments and iteratively prompted it to remove dark patterns from web elements and pages. Initially, they started with a simple prompt *"Make that less manipulative"* and iterated 10 times over the results. After the third iteration, manipulation was fully removed in 45% of cases and partially removed in 24% more. Based on problems they found with this basic prompt, they came up with guardrails consisting of 12 rules, added them to the prompt, and repeated the process from the beginning for 10 iterations. With this improved prompt, 72% of manipulations were fully removed with an additional 19% being partially removed after the third iteration. During this whole process, they observed problems like the LLM hallucinating facts or functions, graying out buttons and thereby adding additional deception or deleting the buttons fully.

Chapter 4

Method

In this chapter, we will detail the approach we used for customizing and evaluating the LLM. Namely, we will explain our customization approach, how the conversations with the LLM took place, and how the results were evaluated.

4.1 Process

The process of our investigation consisted of 3 stages: The customization of the LLM, the actual conversation with it, and the evaluation of the results.

The underlying process of our investigation of deceptive pattern removal is visualized in Figure 4.1 and consists of the following stages: First, we influenced the LLM's responses based on four different factors: *Pre-Training*, *Hyperparameters*, *Prompting*, and *Fine-Tuning*. The conversations with the customized LLM were structured using *Structured Output* and a replacement approach, which only required a short list of changes from the model that were then combined to reconstruct the now more neutral web pages. Finally, we evaluated the results using a metric consisting of our main score for *Manipulation* and two others for *Design, Layout, and Context* and *Functionality*. The evaluation also featured an analysis of instances where the returned changes were not applicable.

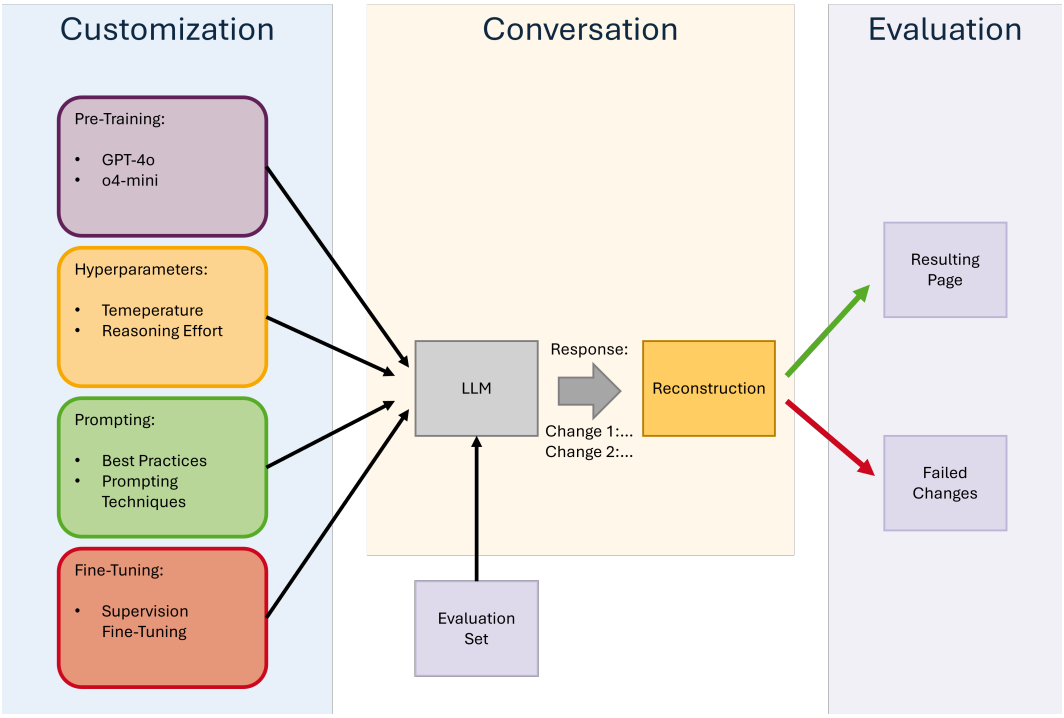

Figure 4.1: The overall process of our investigation into optimizing LLMs for deceptive pattern removal. We customized the LLM based on 4 factors, made it return the neutralized version as a list of changes that are then used to reconstruct the web page and finally, evaluated the restulting page and analyzed failed changes.

4.2 Collecting Examples

We needed examples of deceptive and neutral web pages not only for the evaluation, but also for initial tests, some prompting techniques, and fine-tuning. Therefore, we first gathered a large collection of HTML files with and without deceptive elements.

We gathered a collection of HTML files containing different deceptive or fair designs from related work, real websites, or created by ourselves.

The collection contained web elements and full pages with deceptive patterns provided by Schäfer et al. [2025] and Krauß et al. [2025]. We also recreated examples from Brignull's website[1] and a Hall of Shame[2] either manually

[1] `https://www.deceptive.design/`, *last accessed September 23, 2025*
[2] `https://hallofshame.design/`, *last accessed September 23, 2025*

or using a specialized LLM[3] to turn screenshots into HTML code. Additionally, we extracted the source code from real websites that contained deceptive patterns. Unfortunately, many of the real pages were dynamic and relied on server calls for general functionality and user interactivity. As we only had access to the code on the client side, such functionality was often lost during the extraction. To compensate for this and to allow for the inclusion of deceptive patterns like NAGGING, that rely on dynamic changes of the web page, we wrote fully functional web pages containing such patterns ourselves. We also ensured that our collection included more complicated edge cases, like unremovable deceptive patterns and friction or fair designs. From this collection we put aside an *Evaluation Set* of 25 examples and used the remaining collection with 58 examples as a *Practice Set* for initial tests. While the *Practice Set* included designs similar to our *Evaluation Set* it was important to keep both sets separate, so that the improvements of our prompts, that we used the *Practice Set* for did not overfit to our *Evaluation Set*. We will further detail the contents of the *Evaluation Set* in Chapter 4.5.1.

4.3 Customization

In the customization stage we influenced the responses of the LLM with four different factors: *Pre-Training*, *Hyperparameters*, *Prompting*, and *Fine-Tuning*.

4.3.1 Pre-Training

We investigated one general-purpose and one reasoning model

We decided not to concentrate our efforts on pre-training a model ourselves, but instead decided to use an off-the-shelf model and adapt it to our task. For the choice of this base model, we were interested in whether a *general-purpose model* or a *reasoning model* would perform better in our task. Due to this, we decided to select one model of each type and compare the results.

[3] `https://chatgpt.com/g/g-0fiJrSSdG-screen-shot-to-code`, *last accessed 23. September, 2025*

As our general-purpose model we chose GPT-4o[4]

We decided on GPT-4o[5] as our general-purpose model due to several reasons. For one, GPT-4 and the models based on it were the current state of the art at the time we started working on this thesis and were heavily featured in related work featuring LLMs and deceptive patterns [e.g. Chen et al., 2025; Mills and Whittle, 2023; Schäfer et al., 2025]. GPT-4o also had the additional advantage of a larger context window of 128,000 tokens compared to GPT-4.

We used o4-mini[6] as our reasoning model

For our reasoning model, we wanted to allow the model to spend tokens on the reasoning process without any budgetary restrictions. Due to this, we decided on o4-mini[7] , as it is capable of advanced reasoning, but is significantly cheaper than the full o-series models, reducing our concerns about financial limitations.

Both models also supported various tools that we considered using for this thesis. These include *File Search*, *Dataset Fine-Tuning*, and *Structured Output*. It also allowed for image input. In the end, we did not use all of those, but still kept our initial choice.

4.3.2 Hyperparameters

We investigated temperatures 0.1 and 0.2 for our general -gurpose model and low and high reasoning effort for our reasoning model

As mentioned in Chapter 2.3, the parameters `presence_penalty` and `frequency_penalty` were detrimental to coding tasks and, therefore, also to our objective. Due to this, we did not investigate them further.

As `top_p` and `temperature` both have a similar effect and should not be adjusted simultaneously, we had to decide on one. We chose `temperature` as we wanted to keep the probabilistic distribution while still allowing less likely candidates. We also found more references to `temperature` values in related work [e.g. Li et al., 2024; Sazid et al., 2023; Zhang et al., 2025]. As seen in Chapter 2.3 and Figure **??**, `temperature` 0.2 is recom-

[5] `https://openai.com/index/hello-gpt-4o/`, *last accessed September 23, 2025*

[7] `https://openai.com/index/introducing-o3-and-o4-mini/`, *last accessed September 23, 2025*

mended for code generation, and the next higher value 0.3 is assigned to the task of comment generation. We consider our task as considerably less creative and in need of more deterministic responses than commenting code. Due to this, we decided not to explore temperatures higher than 0.2 and investigate 0.2 as well as 0.1. To this end, we evaluated the performance of both temperatures on the *Basic* and *Advanced Prompts* that we will describe in the following section.

Since `temperature` is not supported by reasoning models, we only investigated the `reasoning_effort` for our o4-mini model and evaluated `low` and `high` reasoning effort.

4.3.3 Prompting

The following section covered how we created the different prompts. We created a minimal *Basic Prompt* as a baseline similar to Schäfer et al. [2025]. For better comparability of the prompting techniques, we wanted to base them all on the same prompt: Our *Advanced Prompt*. This prompt should adhere to recommended best practices for prompting and therefore include all necessary and specific instructions for deceptive pattern removal. In the following section we will explain the process of how these prompts were created. It is important to note, that we specifically excluded the *Evaluation Set* from the prompt generation process, as not to bias our evaluation results. Consequently, once we started using those examples, we did not allow for any further modification of the prompts.

We used the *Practice Set* to create an *Advanced Prompt* as a basis for our prompting techniques. Once we started evaluating, no further adjustments of the prompts were allowed.

Basic and Advanced Prompt

Starting out, we decided on a simple *Basic Prompt* as a baseline without any further prompting strategies applied. We kept this prompt minimal, and notably did not add any instructions on the expected output form while still using *Structured Output* with our standard response format (see Chapter 4.4 for details).

We came up with *Basic Prompt*, a minimal baseline prompt.

Basic Prompt

BASIC PROMPT:
`Neutralize all manipulative elements.`

Following this, we used our *Practice Set* in order to build an *Advanced Prompt*.

Using the *Meta Prompting Technique* we constructed an initial prompt that we adjusted further. For this we prompted o4-mini with an adapted form of the Meta Prompt recommended by OpenAI[8].

Meta Prompt

META PROMPT:
See Appendix A.9.

We iteratively constructed our *Advanced Prompt* using our practice set and adjusting the prompt manually as well as by using the *Meta Prompting Technique*.

Starting with this as a basis, we iteratively adapted the prompt based on problems we found with our practice set and added a rules section where we specified output related instructions. In addition to manual improvements, we also used the *Meta Prompting Technique* for fixing mistakes. For this improvement prompt, we specifically instructed the o4-mini model to add general rules, so as not to overfit to our practice set.

Improvement Meta Prompt

IMPROVEMENT META PROMPT:
See Appendix A.10.

We adapted the results of the meta prompts to remove prompting strategies and references to deceptive pattern terminology.

When using the Meta Prompts, we often had to remove specific prompting techniques like personas, instructions to "think aloud" or examples as these were techniques we wanted to evaluate separately. We also removed specific references to deceptive pattern categories as the goal was for the LLM to detect and neutralize any form of deception. We did not want it to overfit to known deceptive pattern taxonomies as it should also be able to keep up with future forms of deception.

[8] `https://platform.openai.com/docs/guides/prompt-generation`, *last accessed September 22, 2025*

Once we noticed no more crucial shortcomings on our practice set, we stopped further adjustments. While at that point the model did not fully neutralize every example in the practice set, we were satisfied with the overall performance and wanted to leave room for improvement with the different prompting techniques as well as avoid overfitting our prompt to the practice set. Ultimately, this lead to the following prompt:

ADVANCED PROMPT:
See Appendix A.2.

Advanced Prompt

In order to isolate and compare the effect of the different techniques on the model's performance, we decided to base their prompts on the *Advanced Prompt*.

(Zero-Shot) Chain-of-Thought Prompt

To cause the model to output the thought process behind its changes, we simply added a specific instruction to do so at the end of the *Advanced Prompt*. We also had to change our response structure and instruct the model accordingly (see Chapter 4.4.2 for details.)

(ZERO-SHOT) CHAIN-OF-THOUGHT PROMPT:
`Advanced Prompt + ‘‘Think step by step and always describe your approach for each change in the ’explanation’ field. If no changes were possible, add your explanation to the ’comments’ field’’`

(Zero-Shot) Chain-of-Thought Prompt

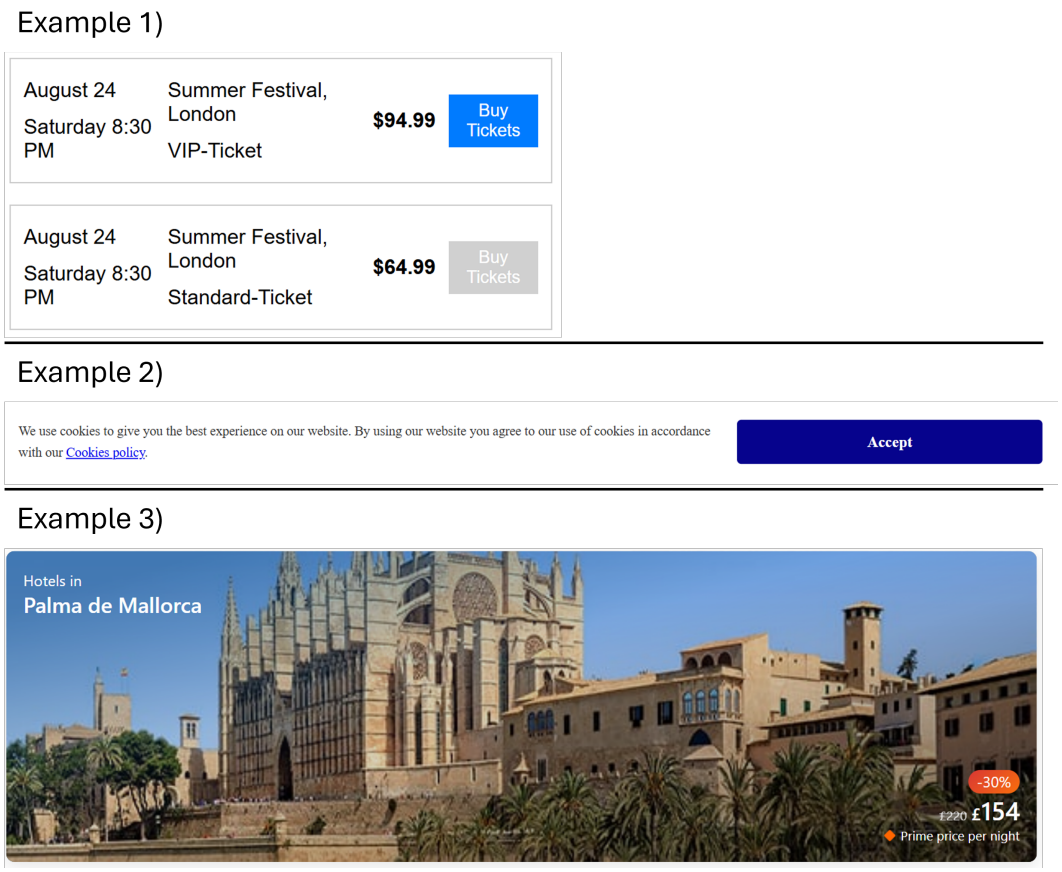

Figure 4.2: The examples provided for the *Few-Shot*, *Few-Shot Chain-of-Thought* and *Combination Prompt*.

Few-Shot Advanced and Few-Shot Chain-of-Thought Prompt

We added three different examples for *Few-Shot* and *Few-Shot Chain-of-Thought Prompting*.

We decided on three different examples for prompts using *Few-Shot Prompting* (see Figure 4.2). In order to maintain a low token count in order to remain within limits and reduce costs, the examples chosen were short and simple. With the latter, we intended to allow for an easier transfer to other web elements employing similar patterns. It is important to note. that we did not intend to cover all the different kinds of deceptive patterns. This would require too many examples, and it would also risk overfitting to the current taxonomies. Instead, we used examples with high transferability (Example 1) or to cover problems that have per-

sisted in the training phase, like the model not recognizing unremovable deceptive patterns (Example 2) and rephrasing deceptive elements like REFERENCE PRICING instead of removing them (Example 3). With the last example we also wanted to include a design that requires several changes.

FEW-SHOT PROMPT:
`Advanced Prompt + Examples`
(See Appendix A.3)

Few-Shot Prompt

For the *Few-Shot Chain-of-Thought Prompt* we also added an explanation for why and how the change neutralizes deception for each example.

FEW-SHOT CHAIN-OF-THOUGHT:
`Chain-of-Thought Prompt + Examples`
(See Appendix A.5)

Few-Shot Chain-of-Thought

Persona Prompts

Despite *Persona Prompting* being controversial, we decided to try out different personas as this has been shown to be effective for the evaluation of UX design [Shin et al., 2025] and the removal of deceptive patterns also requires the model to properly evaluate the design of the web page. We decided to compare three different expert personas. Our first persona reflects the task itself exactly: An expert in the removal of deceptive designs. For the other two, we wanted to try different viewpoints regarding deceptive patterns: First, people with expertise in deceptive patterns. Secondly, the people employing those tactics, who are experts in marketing and sales, that know how to influence people for their profit.

We tested three different expert roles as personas.

All these personas precede the *Advanced Prompt*.

Persona "Deception Removal Expert"

PERSONA "DECEPTION REMOVAL EXPERT":
`''You are an expert in the neutralization of manipulative design and Dark or Deceptive Patterns in particular.'' + Advanced Prompt`

Persona "Deceptive Pattern Expert"

PERSONA "DECEPTIVE PATTERN EXPERT":
`''You are an expert in manipulative design and Dark or Deceptive Patterns in particular.'' + Advanced Prompt`

Persona "Marketing and Sales Expert"

PERSONA "MARKETING AND SALES EXPERT":
`''You are an expert in manipulative design and Dark or Deceptive Patterns in particular.'' + Advanced Prompt`

Prompt Chaining

With *Prompt Chaining* we split the task into an *Analysis* and a *Removal Prompt*

For the *prompt chaining technique*, we split our *Advanced Prompt* into its two main components: Firstly, the analysis of the code for potentially manipulative elements, and secondly, their removal. Each step was designated its own prompt and model response.

Analysis Prompt

ANALYSIS PROMPT:
See Appendix A.6.1.

The result of the first prompt was directly passed to the model for the next prompt using the first response's ID. This allowed the model to access the results of and the context surrounding the analysis step.

Removal Prompt

REMOVAL PROMPT:
See Appendix A.6.2.

Combination Prompt

We constructed a *Combination Prompt* that used all previous prompting techniques combined.

Finally, we tested a combination of the previous techniques. We split our prompt, added examples, an instruction to explain the thought process, and a persona to the individual prompts. For the latter, we chose the *Removal Expert* as it previously performed best out of all three personas (see Chapter 5.1.2 for details). We also adapted the examples and the *Chain-of-Thought* instruction to fit the form of the separate steps.

Combination Analysis Prompt

COMBINATION ANALYSIS PROMPT:
`Removal Expert Persona + ‘‘Think step by step and always describe your approach for each change in the ’explanation’ field. If no important elements were found, add your explanation to the ’comments’ field.’’ + Chaining Analysis Prompt + Examples`
(See Appendix A.7.1.)

Combination Removal Prompt

COMBINATION REMOVAL PROMPT:
`Removal Expert Persona + ‘‘Think step by step and always describe your approach for each change in the ’explanation’ field. If no important elements were found, add your explanation to the ’comments’ field.’’ + Chaining Removal Prompt + Examples`
(See Appendix A.7.2.)

Reasoning Prompt

Our *Reasoning Prompt* is an adaptation of the *Advanced Prompt* to best practices for reasoning models.

Since the concepts of the techniques used for the GPT-4o model are not recommended for reasoning models (see Chapter 2.4.2), we only prompted the o4-mini model with the *Basic Prompt* and adapted the *Advanced Prompt* to the best practices for reasoning models addressed in Chapter 2.4.2. For this, we removed the specific instructions and only kept the initial task context and rules section. We also

added the specific instruction “Take your time and think as carefully and methodically about the task as you need to." from Nori et al. [2024].

Reasoning Prompt

REASONING PROMPT:
See Appendix A.8

4.3.4 Fine-Tuning

We used *Supervised Fine-Tuning* on a GPT-4o model.

As presented in Chapter 2.5, OpenAI supports *Supervised Fine-Tuning*, *Direct Preferences Optimization*, and *Reinforcement Fine-Tuning*. *Direct Preferences Optimization* is a technique for Alignment Fine-Tuning and is used to adjust the style of the model's output to human preferences. As we only care about the performance and not further alignments, we did not consider this process. *Reinforcement Fine-Tuning* requires a specific grader, which could either have been a specific script or an LLM Judge. Due to the nature of our task, a script grading would not work. Instead, an LLM judge would be required, which is another entire field of research and out of the scope of this thesis. Due to this, we decided to instead use *Supervised Fine-Tuning* and used a GPT-4o model `gpt-4o-2024-08-06` as the basis to allow for comparability with the other techniques and results. We did not fine-tune the o4-mini model as reasoning models do not support *Supervised Fine-Tuning*.

Unfortunately, there are no publicly accessible datasets available that we could use for the fine-tuning process. The existing datasets on deceptive patterns are either based on images [Chen et al., 2024] or text [Mathur et al., 2019; Yada et al., 2022], while we would specifically require HTML code.

We gathered a dataset of 106 designs from a workshop conducted at our chair, related work, and our remaining initial collection

Due to this, we created a dataset ourselves for which we conducted a one-afternoon workshop session with eight coworkers from our chair. In this workshop, participants constructed pairs of deceptive and neutral HTML files by either creating them manually, using an LLM, or extracting designs from real web pages and neutralizing them.

One participant was not familiar with HTML and instead paired examples together from the published data of Krauß et al. [2025]. After filtering out designs that were either already in the *Evaluation Set* or not usable, we ended up with 52 designs resulting from this workshop. On top of this, we neutralized some remaining designs of our *Practice Set*, that also included examples of Schäfer et al. [2025], selected and neutralized further designs by Krauß et al. [2025], and added non-removable and fair examples. In total, this culminated in a dataset with 106 examples.

OpenAI's fine-tuning requires the data to be presented in the format of a conversation. We decided to use the *Advanced Prompt* in the dataset for comparability to the other techniques and formatted our deceptive and neutral files into the response format we use with our models (see Chapter 4.4.2).

We completed a fine-tuning iteration with an 80:20 split into training and validation sets and another without splitting the dataset.

After our dataset was prepared, we used an 80:20 ratio to split it into a training and validation set and started an initial fine-tuning iteration with both sets. The data returned for this run revealed decreasing full validation loss, which indicates improving performance of the model on the validation set. However, the other data provided by the integrated evaluation on the validation set was not useful for our task. The generally positive result encouraged us to train for another iteration without adjusting the default parameters, but as the feedback was otherwise unhelpful, we decided not to split the dataset for this iteration, as it was already rather small. Additionally, the information gained by providing the validation set was not worth reducing the examples in the training set.

After the process was complete, we prompted the fine-tuned model with the *Advanced Prompt* and evaluated the results.

4.4 Conversation

In this section, we will describe how we communicated with the LLM.

4.4.1 Setup

We access our models via Azure Open AI[9] and the Responses API[10]

We use Azure Open AI[11] to access OpenAI's Application Programming Interface (APIs) and models. OpenAI offers two different APIs: The *Chat Completions API*[12] and *Responses API*[13] . While *Chat Completions* is still supported, OpenAI calls for migration to *Responses* for new projects. Using *Responses* also improves model performance, reduces costs, has the additional advantage that future models will be supported, and supports additional tools and functionalities like file search, code interpretation, and the creation of reasoning summaries for reasoning models[14]. Due to these benefits, we decided to access our models via *Responses*.

In order to keep track of our budget, we had a token limit and allowed for up to 100,000 tokens for the GPT-4o Model and 500,000 tokens for the o4-mini model.

[11] `https://azure.microsoft.com/en-us/products/ai-foundry/models/openai`, *last accessed September 23, 2025*

[12] `https://platform.openai.com/docs/api-reference/chat`, *last accessed September 23, 2025*

[13] `https://platform.openai.com/docs/api-reference/responses`, *last accessed September 23, 2025*

[14] `https://platform.openai.com/docs/guides/migrate-to-responses`, *last accessed September 22, 2025*

4.4.2 Messages

Our conversation with the LLM consists of a list of messages with different types:

1. A *system* or *developer message* that gives the LLM general instructions of what to do. This is the prompt we aim to optimize through prompting techniques.
2. A *user message* consisting of the HTML code for a web page or element.
3. An *assistant message* which contains the answer of the LLM.

We used our previously presented prompts as *system* or *developer messages*.

System or *developer messages* provide general instructions that are applied to the *user message* and as such have higher priority. We started out using *system messages* for the general-purpose model but switched to *developer messages* for the reasoning models as this was explicitly recommended for these models[15]. However, this affects the hierarchy of the different messages while being functionally identical[16], and as we always only used either a *developer* or a *system message*, there is no indication that this affected anything.

The model's response is returned as individual changes that are then applied to the HTML file.

The *user message* contains only the HTML code. We aimed to minimize the amount of tokens this requires by deleting unused CSS rules and removing whitespace, comments, and unused attributes in the HTML code. We use *Structured Output* for the *assistant message* in order to receive a specific answer structure. In this structure, the model only returns different changes that are to be applied to the original HTML file in order to neutralize it, instead of returning the full neutralized file. This crucially reduces the amount of spent output tokens, decreasing costs and also helping us adhere to token limits. We defined our formats as Pydantic

[15] `https://platform.openai.com/docs/guides/reasoning-best-practices`, *last accessed September 28, 2025*

[16] `https://learn.microsoft.com/en-us/azure/ai-foundry/openai/how-to/reasonings`, *last accessed September 30, 2025*

Base Models[17] . All structures can be found in Appendix B and are based on the following standard response format.

The standard format of the model's response.

```
class Response(BaseModel):
   changes: list[Change] = Field(description="All the
       changes that are to be made")
   comments: str = Field(description="If changes were
       unnecessary or impossible")


class Change(BaseModel):
   original: str = Field(description="The exact
       original HTML code that should be removed")
   replacement: str = Field(description="The new HTML
       string that replaces the manipulation")
```

In each response, the model returned a list of `Changes` and a field for `comments`.
Each `Change` consists of a part of `original` HTML code that gets replaced with the `replacement` code. In case an element should be deleted, `replacement` was left empty. The `comments` field was used for edge cases in which changes were either unnecessary or not possible.

This standard format had to be adapted for certain prompting techniques. For Prompts featuring chain-of-thought output, an `explanation` field was added to each `change`. In case of split prompts, the analysis step needed an individual response structure: It returned a list of `Important Elements` that contain the HTML code of the element and a short description of it. We added the description part, as the identification of relevant aspects is part of the analysis step. The *Removal Prompt* should only concern itself with whether the influence counts as manipulative and how it could be neutralized. Therefore, we wanted to have the influence already described in the input for the *Removal Prompt*. The output format also allowed for a field with comments in case no elements of interest were identified.

[17] `https://docs.pydantic.dev/latest/api/base_model/`, *last accessed September 23, 2025*

In the *Combination Prompt* the `Important Elements` also received an `explanation`.

4.4.3 Reconstruction

After receiving the model's response, we iterated through all `changes` and applied them to the HTML file. However, since we did this using `string.replace()`, the `original` code returned in each change had to match the element in the original file exactly. If this was not the case, the change failed, which we recorded and analyzed afterwards.

If the code returned in the `change` did not match the original file exactly, the change fails.

4.5 Evaluation

In order to compare the influence of the different customization approaches, we evaluated their performance on an *Evaluation Set* consisting of 25 web pages or elements using a metric we derived from related work. Lastly, we also kept track of instances where the changes returned by the model failed.

4.5.1 Evaluation Set

We put aside 25 examples from our original collection (see Chapter 4.2) in order to use them for the evaluation. Due to this, these designs were not included in the *Practice Set* or the dataset used for fine-tuning. Therefore, they were unknown to the LLM, allowing for unbiased evaluation. Table 4.3 gives an overview of the designs featured in the *Evaluation Set* and the different deceptive patterns that were included based on the ontology by Gray et al. [2024].

During the selection, we prioritized examples from real websites and only added manually created examples for deceptive patterns that were not extractable but of certain interest. This included patterns that resulted from elements being created dynamically, such as SNEAK INTO BASKET

Our *Evaluation Set* included deceptive designs from all high-level categories by Gray et al. [2024].

ID	Deceptive Patterns	Source
E00	False Hierarchy, Pressured Selling, Positive or Negative Framing, Visual Prominence, Drip Pricing, Hidden Costs, or Partitioned Pricing	Ryanair
E01	False Hierarchy, Low Stock, Emotional or Sensory Manipulation, Reference Pricing	Wizzair
E02	Countdown Timer, Visual Prominence, Emotional and Sensory Manipulation, Scarcity and Popularity Claims, Reference Pricing, False Hierarchy	AsapTickets
E03	Scarcity and Popularity Claims, Emotional or Sensory Manipulation, Countdown Timer, Limited Time Message	Amazon
E04	Low Stock, High Demand	Viagogo
E05	Bad Defaults, Trick Question (not removable)	Sky
E06	Visual Prominence, Emotional and Sensory Manipulation, Low Stock	Booking
E07	False Hierarchy, Emotional and Sensory Manipulation	Wizzair
E08	Privacy Zuckering (not removable)	Opodo
E09	Fair	European Union
E10	Fair	British Museum
E11	High Demand, Low Stock	Viagogo
E12	Fair	MasterCubeStore
E13	Countdown Timer, Reference Pricing, Visual Prominence	self made
E14	Sneak into Basket, Bad Defaults	self made
E15	Nagging, Confirmshaming	self made
E16	Price Comparison Prevention	self made
E17	Fair (Friction)	self made
E18	Visual Prominence, Limited Time Message, Disguised Ad, Reference Pricing	Wish
E19	False Hierarchy, Pressured Selling, Emotional and Sensory Manipulation	Ryanair
E20	Emotional and Sensory Manipulation	Ryanair
E21	Drip Pricing, Hidden Costs, or Partitioned Pricing	AirBnB
E22	Bad Defaults, Activity Messages, Visual Prominence, Parasocial Pressure	WinRed
E23	Confirmshaming, Visual Prominence	Ryanair
E24	False Hierarchy	self made

Table 4.3: The examples in the *Evaluation Set* and the deceptive patterns they include after the ontology by Gray et al. [2024]. We included instances from all high-level deceptive pattern categories as well as fair and friction designs.

(E14), NAGGING (E15), or PRICE COMPARISON PREVENTION (E16). We specifically added these in order to include patterns from all high-level categories from the ontology by Gray et al. [2024].

We added fair, friction, and unremovable deceptive designs as edge cases to our *Evaluation Set*.

Lewis and Vassileva [2024] highlighted the importance of context for the distinction between deceptive patterns and designs that are beneficial to the user by contrasting OBSTRUCTION as a manipulative technique with OBSTRUCTION used as a friction design for deleting a branch in GitLab. To address this edge case, we also wrote a similar profile deletion page and included it in our evaluation page (E17). Furthermore, we added fair versions of web elements that commonly contain deceptive patterns, like a cookie banner (E09), an online shop (E10), and a sign-up form (E12). With the friction and the fair designs, we aimed to test whether the LLM can use the context of a page to determine the lack of manipulation for the friction case, but also disregard the context of the chosen fair designs, often associated with deception. As not all deceptive patterns are removable, we added such cases with two patterns: an opt-out design (E05) and a cookie banner with no option to decline (E08). For all these conditions, we intended the LLM to return no changes and instead state that no changes were possible or necessary.

The examples encompass different sizes and atmospheres.

Lastly, we used pages with different sizes and settings to assess if the LLM interferes with other elements on the page. For example, E02 contained a full web page with many additional, irrelevant elements. E22 is a political donation web page and thus uses deceptive patterns in an overall highly emotional atmosphere.

We used a strict interpretation of Gray et al.'s definitions.

Appendix C specifies the different definitions of the patterns, but it has to be noted that for many patterns Gray et al. specifies a need for the claims pressuring the user to be misleading or false. However, as we cannot verify such sources and also intended to make our LLM strict with elements that might already be potentially manipulative, we always assumed the origin of the coercive elements to be deceptive and, as such, covered by Gray et al.'s ontology.

4.5.2 Metric

The metric for our evaluation was based on problems we identified in initial tests and that occurred in related work. It covered 3 different factors: *Design, Layout, and Context*, *Manipulation* and *Functionality*.

Design, Layout, and Context

We evaluated whether the design, layout, and context remained consistent with the original page.

In their work Li et al. [2023] noted problems of LLMs hallucinating elements, resulting in a cluttered layout. As cluttered layouts are considerably unpopular in countermeasures for deceptive patterns [Schäfer et al., 2023], we wanted the model to only neutralize the manipulative elements and leave the rest of the page unchanged.

For this, we came up with a score to measure whether the resulting page stayed consistent with the original regarding design, layout, and context, ranging from 0 to 4:

4: No neutral elements were changed, and the changes of the manipulative elements matched the design, layout, and context of the original page.

3: There were changes to neutral elements, but the results were consistent with the design, layout, and context of the original page.

2: There were changes to the design, layout, and context, but they only deteriorate aesthetics.

1: Changes to the design, layout, or context were either substantial or resulted in a confusing user experience.

0: The page was broken.

In case no changes took place, the score defaulted to 4.

Manipulation

The *Manipulation Score* is probably the most crucial for our task and describes how the LLM handled deceptive patterns. It is inspired by the score of Schäfer et al. [2025] and also ranges from 0 to 4:

We measured how much of the manipulation was removed.

- 4: All manipulative elements were neutralized or it was correctly identified that no changes were necessary or possible.
- 3: The web page is less manipulative with at least half of the manipulative elements neutralized.
- 2: The web page is less manipulative with less than half of the manipulative elements neutralized.
- 1: The web page is no more or less manipulative than before
- 0: Existing manipulative effects were enhanced or new manipulative elements were added.

For this metric, the default value in case the model returned no successful changes is 1.

It is important to note that the score 0 does not necessarily mean that the page is now more manipulative overall. Instead, we intended to penalize the LLM as soon as it is the cause of deception. In cases of non-removable or fair designs, we also required the model to return "No changes necessary" or "No changes possible" in order to receive the full points. For the fair designs, we wanted this in order to reward the model correctly, recognizing that no manipulative elements were featured. Without this rule, a model returning no changes and one mistaking and changing other elements for being manipulative would receive the same *Manipulation Score* despite the latter misunderstanding manipulation. For unremovable designs, the model identifying that removal does not work on a deceptive design would enable us to activate other countermeasures as fallback options.

Functionality

Lastly, we kept track of whether the functionality of the page was still intact.

As this was also a problem occurring in related work [Li et al., 2023], we also wanted to track if the changes of the LLM broke the functionality of the web page or element. Since the extraction of the pages sometimes removed it as well, not all our evaluated pages were functional. In such cases, all results for this design received a *Functionality score* of 0 regardless of the LLM's response. Otherwise, a score of -1 was assigned if the originally intended actions a user could take on the original page were not possible after the changes. If this was not the case or no changes took place, the value remained at 1.

4.5.3 Success Rate

Success Rate measures how many of the returned changes were successful.

Due to hallucinations of the LLM, the code returned in the `original` fields of the `changes` was occasionally slightly different from the initial code, causing the planned change to fail. To keep track of this, we recorded the number of planned and successful changes. The ratio of successful to planned changes is represented by the *Success Rate*.

Chapter 5

Results

This chapter presents the results of the evaluation. For this, we will present all the different conditions tested step by step and compare them to the *Advanced Prompt*. As an overview, Appendix D contains a table which includes mean and mode for the *Design, Layout, and Context* and *Manipulation Score* over all 17 conditions as well as the amount of assigned scores of 0 and 4. In all tables and figures we will use the following abbreviations for the most important values: The average *Design, Layout, and Context Score* (∅ DLC), the average *Manipulation Score* (∅ M), the average *Functionality Score* (∅ F), *Success Rate* (SR) and the number of changes the LLM returned (PC), which includes both successful and failed changes.

5.1 General-Purpose Model

For our general-purpose model, we evaluated the temperatures 0.1 and 0.2 on the *Basic* and *Advanced Prompt*, our 8 prompts that featured prompting techniques, and the fine-tuned GPT-model. Figure 5.1 gives an overview of all prompts used with the non-fine-tuned general-purpose model. In those violin plots the scores of the *Advanced Prompt* is grayed out in the background to allow for easier comparison.

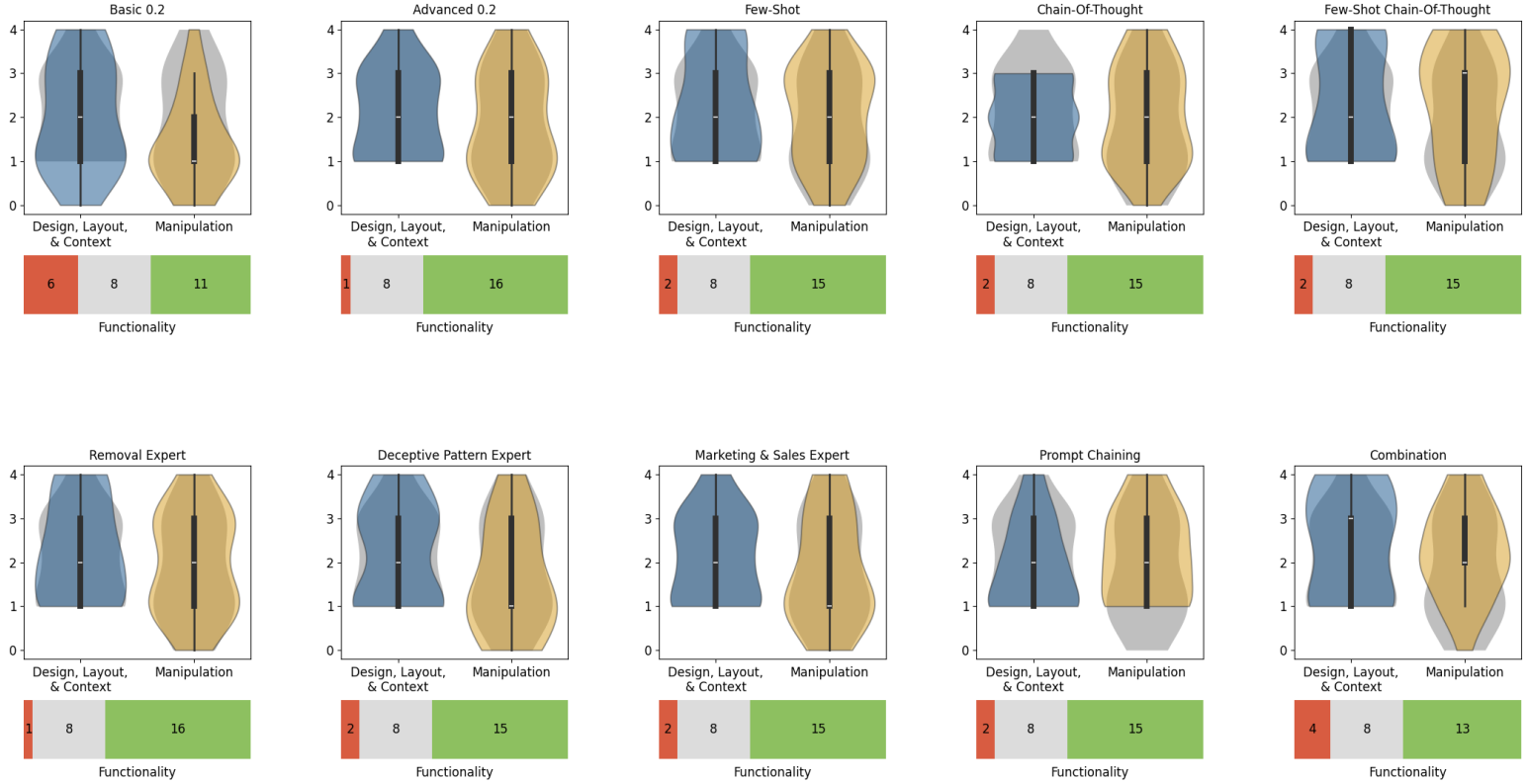

Figure 5.1: An overview of the results from the general-purpose model. The score of the *Advanced Prompt* is shown grayed out in the background to allow for easier comparison. Most prompting techniques showed improvement over the *Advanced Prompt*.

5.1.1 Basic and Advanced Prompt

Prompt (`temperature`)	∅ DLC	∅ M	∅ F	SR	PC
Basic (`0.1`)	1.92	1	0.12	0.8969	165
Basic (`0.2`)	2.04	1.4	0.2	0.8922	167
Advanced (`0.1`)	2.08	1.84	0.6	0.9804	153
Advanced (`0.12`)	2.12	1.84	0.6	0.9470	132

Table 5.2: The average *Design, Layout, and Context* (∅ DLC), average *Manipulation* (∅ M), *Functionality* (∅ F), *Success Rate* (SR), and total amount of intended changes of the *Basic* and *Advanced* Prompt for the temperatures 0.1 and 0.2. Temperature 0.2 outscored 0.1 for the *Basic Prompt* for all values and for DLC in the *Advanced Prompt*.

Table 5.2 shows the results for the *Basic* and *Advanced Prompt* for the temperatures 0.1 and 0.2.

Our baseline *Basic Prompt* received the worst scores overall as it was overly critical and misjudged common design and functionality as manipulative.

The *Basic Prompt* received poor results across all scores, regardless of the temperature used. It struggled to properly estimate what is manipulative and should be removed. For instance, it often removed entire `script`-tags or function calls, thereby breaking the functionality of seven and six pages in total for temperature 0.1 and 0.2, respectively. The *Design, Layout, and Context* of the pages was also frequently misjudged as manipulative, as the model disapproved of the use of colors or shadow effects in general, regardless of whether they caused INTERFACE INTERFERENCE, VISUAL PROMINENCE, or FALSE HIERARCHY. All of this did not actually improve the average *Manipulation*, which remained at the default 1 for temperature 0.1 and only slightly improved to 1.4 for temperature 0.2. However, most problematically, with these changes, the model sometimes increased the manipulative effect by removing indicators of deceptive patterns, but not the pattern itself. This occurred, for example, in E18 where the model removed the already barely noticeable label marking a DISGUISED AD, but not the actual ad. The prompt also resulted in the highest amount of planned changes (165 and 167), indicating an overeagerness for altering the page.

The *Advanced Prompt* performed notably better than our baseline across all scores, but still left room for improvement.

In contrast, the *Advanced Prompt* received better results. Here, functionality was only broken for one page in both temperatures. Its main struggle was the *Design, Layout, and Context* as it made changes that left the remaining page confusing or changed some trivial design traits or phrasings. But while the average *Design, Layout, and Context* was rather low with 2.08 and 2.12, the page or element was never broken, so it never received a *Design, Layout, and Context Score* of 0. Overall, *Average Prompt* made slightly fewer changes than *Basic Prompt* but had a noticeably better *Success Rate*.

Temperature

The differences between temperature 0.1 and 0.2 were minimal, but 0.2 received slightly better average scores.

Comparing the results of temperatures 0.1 and 0.2 only yields slight differences. Using the *Average Prompt* with a higher temperature only slightly improved the average *Design, Layout, and Context* by 0.4 and improved the *Success Rate* by 3.34%, but had no effect on the *Functionality*. But while, the average *Manipulation Score* also remained the same, with temperature 0.2 the LLM received one more score of 0, meaning that deception was enhanced or added for one design more.

On the other hand, we observed the reverse for the *Basic Prompt*: Here, the temperature 0.1 received one more *Manipulation Score* of 0 compared to 0.2. For this prompt, increasing the temperature also improved all average scores, albeit rather minimally. For *Design, Layout, and Context*, it is also noteworthy that temperature 0.1 broke one more page completely.

We used temperature 0.2 for all other prompts.

In summary, while the differences were minimal, temperature 0.2 performed slightly better than 0.1. Due to this, we decided to use temperature 0.2 for all other prompts that followed. In the following, when speaking of the *Basic* or *Advanced Prompt*, we will always refer to the results of temperature 0.2 unless stated otherwise.

5.1.2 Prompting Techniques

As we based all prompts for the different techniques on the *Advanced Prompt*, we will present its average scores in the tables as well to allow for easier comparison.

Few-Shot Prompting

Prompt	∅ DLC	∅ M	∅ F	SR	PC
Advanced	2.12	1.84	0.6	0.9470	132
Few-Shot	2.32	2.08	0.52	0.9739	153

Table 5.3: The average *Design, Layout, and Context* (∅ DLC), average *Manipulation* (∅ M), *Functionality* (∅ F), *Success Rate* (SR), and total amount of intended changes of the *Few-Shot* and *Advanced Prompt*. *Few-Shot Prompting* improved the average *Design, Layout, and Context* and *Manipulation Scores*, while deteriorating *Functionality*

Few-Shot Prompting improved the average *Manipulation* by 0.24 and the average *Design, Layout, and Context* by 0.2, but broke the functionality of one page more compared to the *Average Prompt*.

Adding simple examples to the *Average Prompt* improved the average *Design, Layout, and Context* by 0.2 and the average *Manipulation* by 0.24 (see Table 5.3). Notably, the model still did not properly recognize the cookie banner with only an accept button (E08) as an unremovable deceptive design, even though a similar case was included in the examples. This means that it was not able to transfer this same case of deceptive pattern. But the overall better performance regarding *Manipulation* still indicates that it was able to transfer some aspects of the examples to other deceptive patterns.

Interestingly, with 149 out of 153, the *Few-Shot Prompt* produced considerably more successful changes than the other prompting techniques or the reasoning model. This was only matched by the *Basic Prompt* or the *Advanced Prompt* with temperature `0.1`. The success rate of 0.974 is also rather high.

Zero- and Few-Shot Chain-of-Thought

Prompt	∅ DLC	∅ M	∅ F	SR	PC
Advanced	2.12	1.84	0.6	0.9470	152
Zero-Shot Chain of Thought	2	2.08	0.52	0.9444	108
Few-Shot Chain of Thought	2.36	2.28	0.52	1	128

Table 5.4: The average *Design, Layout, and Context* (∅ DLC), average *Manipulation* (∅ M), *Functionality* (∅ F), *Success Rate* (SR), and total amount of intended changes of the *Zero-* and *Few-Shot Chain-of-Thought Prompts* and the *Advanced Prompt*. *Few-Shot Chain-of-Thought* achieved the highest improvement over the *Advanced Prompt* out of all prompting techniques.

Zero- and *Few-Shot Chain-of-Thought Prompting* increased the *Manipulation Score* by 0.24 and an additional 0.2, but broke the functionality of one page more.

Zero-Shot Chain-of-Thought deteriorated the *Design, Layout, and Context* of the examples by 0.12 on average and broke the *Functionality* of one page more. However, it increased the average *Manipulation Score* by 0.24 and an additional 0.2 once examples were added in the *Few-Shot Chain-of-Thought Prompt*. The latter also increased the average *Design, Layout, and Context Score* again, although the lower *Functionality Score* remained the same (see Table 5.4).

Unlike simple *Few-Shot Prompting*, the *Few-Shot Chain-of-Thought Prompt* correctly recognized the PRIVACY ZUCKERING cookie banner with only an accept button (E08) as a deceptive, but unremovable design. It was the only prompt that did so in this thesis. However, it nonetheless changed the phrasing of the accept button minimally instead of returning no changes as it was instructed to do.

Few-Shot Chain-of-Thought was the only condition with a *Success Rate* of 1, meaning that the LLM never hallucinated any of the `original` code returned in the changes.

Persona Prompts

Prompt	∅DLC	∅ M	∅ F	SR	PC
Advanced	2.12	1.84	0.6	0.9470	152
Expert Persona "Deception Removal"	2.28	1.96	0.6	0.9030	134
Expert Persona "Deceptive Pattern"	2.24	1.6	0.52	0.8811	143
Expert Persona "Marketing & Sales"	2.12	1.8	0.52	0.9638	138

Table 5.5: The average *Design, Layout, and Context* (∅ DLC), average *Manipulation* (∅ M), *Functionality* (∅ F), *Success Rate* (SR), and total amount of intended changes of the *Advanced* and different *Persona Prompts*. Only the *Removal Expert* persona resulted in improved scores.

Person Prompting resulted in overall weak performanced. Only the *Removal Expert Persona* showed an improvement over the *Advanced Prompt*.

Out of the three Personas, only the *Removal Expert Persona* showed an improvement to the *Advanced Prompt* and outperformed the other two in all metrics. The other two actually deteriorated the *Manipulation* and *Functionality Scores*. Especially the *Deceptive Pattern Expert* got unsatisfactory *Manipulation* scores: It enhanced existing or introduced new deception for five examples. A score of 0 being assigned so often was only also the case for the *Basic Prompt* with temperature `0.1`. The average *Manipulation* is also the lowest of all prompts except for the *Basic Prompt*.

Prompt Chaining and Combination Prompt

Despite never breaking the page and thus never receiving a score of 0, *Prompt Chaining* received the worst results on average in the *Design, Layout, and Context Score* out of all techniques and also had the lowest *Success Rate* out of all general-purpose model prompts (76.79%). This could have been caused by the split prompts requiring two responses and thereby giving double the opportunity for hallucinations. But surprisingly, the *Combination Prompt*, which was also split, resulted in the highest average *Design, Layout, and*

Prompt	∅ DLC	∅ M	∅ F	SR	PC
Advanced	2.12	1.84	0.6	0.9470	132
Prompt Chaining	1.92	2.20	0.52	0.7679	112
Combination	2.44	2.24	0.36	0.9041	73

Table 5.6: The average *Design, Layout, and Context* (∅ DLC), average *Manipulation* (∅ M), *Functionality* (∅ F), *Success Rate* (SR), and total amount of intended changes of the *Advanced*, *Prompt Chaining*, and *Combination Prompts*. While the *Chaining Prompt* received the worst overall *Design, Layout, and Context Score*, *Combination* received the best.

Context Score out of all prompting techniques and achieved a much higher *Success Rate* of 90.41%.

The *Manipulation Score* was an improvement for both Prompts, but the overall *Functionality* suffered: *Prompt Chaining* removed functionality from two pages in total which is one more than than the *Average Prompt* did. Worryingly, *Combination* broke the functionality of 4 pages.

5.1.3 Fine-Tuning

Fine-Tuning achieved the best performance overall.

With a median of 3 and a mode of 4 for both *Design, Layout, and Context* and *Manipulation*, the fine-tuned model achieved the overall best performance of all conditions. It was also the most reliable condition for dealing with fair or unremovable designs and the only one that recognized it would not be able to remove the deception resulting from the opt-out checkbox in E05. Despite this, it did not properly recognize the same for the other unchangeable cookie banner (E08) and instead hallucinated a reject button. This behavior we otherwise only observed in the reasoning model.

However, the *Functionality Score* was worse than the one of the *Advanced Prompt*, and the fine-tuned model was notably also the only instance of a general-purpose model condition, apart from the *Basic Prompt*, that broke a page com-

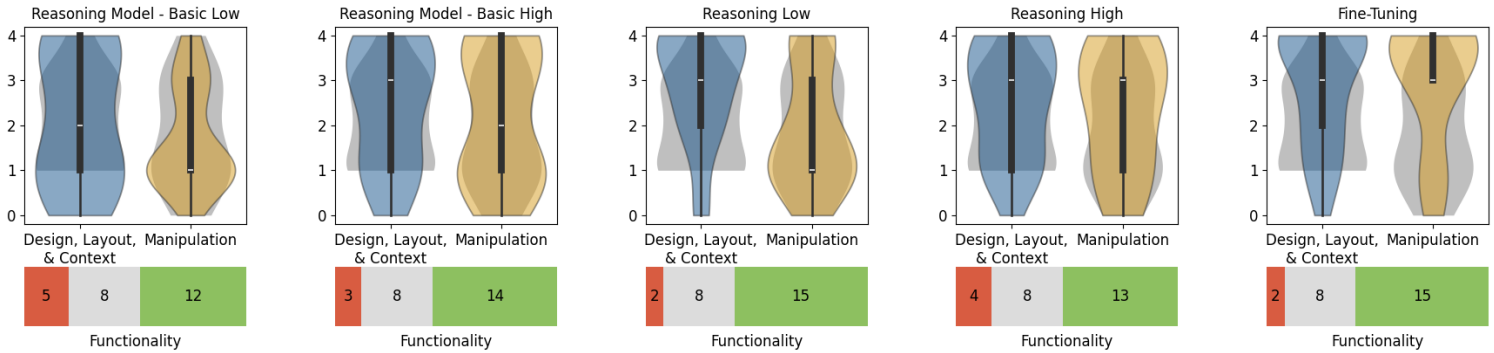

Figure 5.7: An overview of the results from the reasoning and fine-tuned model. The score of the *Advanced Prompt* is shown grayed out in the background to allow for easier comparison. The fine-tuned model showed the best performance overall, followed by the *Reasoning Prompt* with high reasoning effort.

Prompt	∅ DLC	∅ M	∅ F	SR	PC
Advanced	2.12	1.84	0.6	0.9470	132
Fine-Tuning	2.88	2.92	0.52	0.8523	149

Table 5.8: The average *Design, Layout, and Context* (∅ DLC), average *Manipulation* (∅ M), *Functionality* (∅ F), *Success Rate* (SR), and total amount of intended changes of the *Fine-Tuned* model and *Advanced Prompt*. Fine-tuning resulted in the best scores overall.

pletely and received a 0 as the *Design, Layout, and Context Score* for one example

Another interesting observation was that for E03, a design that featured multiple product cards that each contained the same deceptive patterns, the fine-tuned model removed almost all deceptive patterns consistently in each card, except for the "claimed" bars, which were never removed. Notably, the training set the model was fine-tuned on featured a similar design that also contained claimed bars that were removed.

Figure 5.7 shows the performance of the fine-tuned model next to the reasoning models. Again, the scores of the *Advanced Prompt* are grayed out in the background.

5.2 Reasoning Model

With the Reasoning Model, we evaluated a total of 4 conditions with the *Basic* and *Reasoning Prompts* being evaluated with low and high reasoning effort. As for the fine-tuned model, an overview of the results can be found in Figure 5.7.

5.2.1 Basic Prompt

Using the *Basic Prompt* with the reasoning model achieved better results than using it with the general-purpose model.

Notably, despite using the exact same prompt, using the *Basic Prompt* with the reasoning model achieved much better

Prompt (`reasoning_effort`)	∅ **DLC**	∅ **M**	∅ **F**	**SR**	**PC**
Advanced	2.12	1.84	0.6	0.9470	132
Basic (`low`)	2.12	1.72	0.28	0.7049	122
Basic (`high`)	2.44	2.16	0.44	0.8772	114

Table 5.9: The average *Design, Layout, and Context* (∅ DLC), average *Manipulation* (∅ M), *Functionality* (∅ F), *Success Rate* (SR), and total amount of intended changes of the *Basic* (`low`), *Basic* (`high`), and *Advanced* Prompt. While the scores were still rather low, especially for `low` reasoning effort, the results were overall better than with the general-purpose model.

results than it did with the general-purpose model. Still, the scores were generally low when using low reasoning effort. This was also accompanied by a weak *Success Rate* of only 70.49%. This was, however, considerably improved by allowing the model to spend higher effort on the reasoning process, where the model even achieved higher scores than some prompting techniques in the *Manipulation* and *Design, Layout, and Context Scores*.

5.2.2 Reasoning Prompt

Prompt (`reasoning_effort`**)**	∅ **DLC**	∅ **M**	∅ **F**	**SR**	**PC**
Advanced	2.12	1.84	0.6	0.9470	132
Reasoning (`low`)	2.92	1.96	0.52	0.7013	77
Reasoning (`high`)	2.52	2.32	0.36	0.8241	108

Table 5.10: The average *Design, Layout, and Context* (∅ DLC), average *Manipulation* (∅ M), *Functionality* (∅ F), *Success Rate* (SR), and total amount of intended changes of the *Reasoning* (`low`), *Reasoning* (`high`), and *Advanced* Prompt. Increasing the effort resulted in overall remarkable scores.

The *Reasoning Prompt* with low reasoning effort achieved quite contrasting results: It achieved a rather high *Design, Layout, and Context score* of 2.92 on average, but low results for *Manipulation* with only 1.72. Such a high difference between the two scores was not observed in any other condition, as commonly the results either both around roughly the same values, or only one score performed especially low or high, while the other achieved an unremarkable result.

The *Reasoning Prompt* with low reasoning effort received contrasting results for the *Design, Layout, and Context* and *Manipulation Score*.

However, increasing the reasoning effort improved the *Manipulation*, resulting in an overall positive performance.

Reasoning Effort

Increasing reasoning effort lead to better results.

Unsurprisingly, increasing the reasoning effort strongly improved the overall results. However, we noted that the reasoning effort also had a high influence on the *Success Rate*, with low effort only resulting in around 70% of changes being successful.

5.3 Results of Specific Designs

5.3.1 Fair and unremovable Designs

Overall, the models were hesitant to not change anything. Even fair designs predominantly resulted in minor design changes and, for the unremovable designs, neutralization was attempted. This often resulted in a visual neutralization of the pattern that was not actually supported by the page's functionality, essentially resulting in a BAIT AND SWITCH deceptive pattern.

E08: A fair cookie banner

In the fair cookie banner colors and phrasings were frequently changed.

Only the fine-tuned model correctly identified that no changes were necessary for the fair cookie banner, while all other prompts changed the phrasing of the buttons or changed the colors. This resulted in generally low *Design, Layout, and Context Scores*. *Prompt Chaining* received a full score of 4, but this was only because all of the intended changes failed, and the score, therefore, used the default value. *Combination Prompt* actually turned the fair into a *Bright Pattern*, arguing that *"The button to accept all cookies is visually identical to the button for accepting only essential cookies. This can lead users to choose 'Accept all cookies' without*

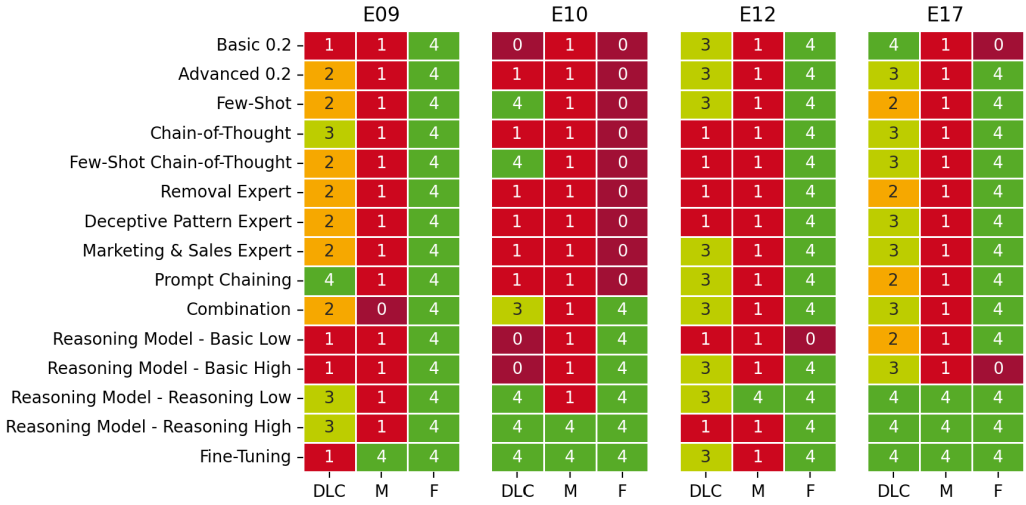

Figure 5.11: The *Design, Layout, and Context Score* (DLC), *Manipulation Score* (M) and *Functionality Score* (F) of the different prompts for the fair designs. For easier readability, we normalized the *Functionality Score* to the range 0 to 4, meaning that a broken functionality with *Functionality Score* -1 would be displayed as a 0 in this figure, while an intact functionality is shown as a 4.

realizing they have a less intrusive option. Differentiating the buttons visually can help users make a more informed choice"

E10: A fair web shop

The wishlist feature of the fair web shop was often considered non-essential or even manipulative and consequently removed.

The fair online shop in E10 had a wishlist feature, which allowed for items to be saved for later purchases. This feature in itself was frequently removed. *Zero-Shot Chain-of-Thought* justified this with *"The 'Add to Wishlist' button is non-essential and can be removed to prevent steering users towards unnecessary actions."* Removing the wishlist feature and buttons often also slightly broke the layout of the page, resulting in lower *Design, Layout, and Context scores*. The design was, however, correctly identified as fair by the *Reasoning Prompt* with high reasoning effort and the *Fine-Tuned* model.

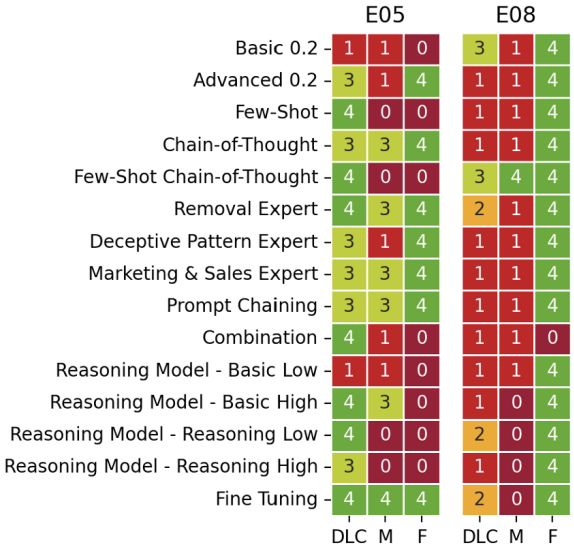

Figure 5.12: The *Design, Layout, and Context Score* (DLC), *Manipulation Score* (M) and *Functionality Score* (F) of the different prompts for the unremovable designs. For easier readability, we normalized the *Functionality Score* to the range 0 to 4, meaning that a broken functionality with *Functionality Score* -1 would be displayed as a 0 in this figure, while an intact functionality is shown as a 4.

E12: A fair sign up form

Only the *Basic Prompt* with low reasoning effort identified this design as fair with no necessary changes. All the other prompts changed the phrasing of a checkbox, which offered emails about discounts when clicked.

E17: A profile deletion page with Friction Design

Friction design was not seen as problematic.

Our friction example was correctly identified as non-manipulative by the *Reasoning Prompt* and the *Fine-Tuned* model. Encouragingly, the friction aspect was never seen as problematic by the models despite closely resembling OBSTRUCTION patterns. Instead, the models sometimes changed different stylistic choices, like the use of icons or simple phrasing which lead to a lower *Design, Layout, and Context*.

E05: An unremovable opt-out

Only the *Fine-Tuned* model correctly identified the opt-out feature as unremovable. Other prompts often attempted to change it into an opt-in situation, by changing the phrasing in the label and sometimes also changing the `name`-attribute in the HTML code. This, however, would break the functionality of the page and essentially introduce a BAIT AND SWITCH pattern: The user would leave the box unchecked, expecting nothing to happen due to the label stating that the box has to be checked. But as only the phrasing of the label was changed and the underlying functionality remained the same or broke, not checking the box would cause the website to behave as if the user agreed.

E08: A forced accept cookie banner

The reasoning and fine-tuned models hallucinated a reject button, which in some cases included FALSE HIERARCHY.

The forced accept cookie banner was never properly recognized by any model. The standard GPT-4o model instead only changed its design, often graying out the button, while the fine-tuned and reasoning model hallucinated a reject option. As this button would not be functional due to the website not offering the option to reject cookies, this, again, would cause a BAIT AND SWITCH situation: A user would click reject and expect cookies not to be saved, only for the opposite to happen. It was also noticeable that the reject buttons of the *Basic Prompt* with high reasoning effort and of the *Reasoning Prompt* with low reasoning effort were grayed out, despite no other element in the HTML file having such a design. This also caused a FALSE HIERARCHY between the accept and reject buttons, which is a new deceptive pattern added by the model.

5.3.2 Unchanged Deceptive Patterns

Some instances of PRESSURED SELLING were never removed.

Some deceptive patterns have been shown to be a challenge across all or most prompts. Examples taken from Ryanair[1] often included a very subtle PRESSURED SELL-

[1] `https://www.ryanair.com/`, *last accessed September 27, 2025*

ING by slightly graying out the background of a cheaper option. This was never addressed or changed in any way by the LLMs.

Patterns that required restructuring, like SNEAK INTO BASKET (E14), or referred to other elements further away, like DRIP PRICING (E21), or some cases of FALSE HIERARCHY (E02, E24) also caused the LLMs to struggle. Those cases were often only neutralized by the reasoning or fine-tuned model.

DISGUISED AD was also challenging.

The DISGUISED AD in E18 was only changed by the *Chaining* and *Combination* prompt, which only changed "ad" to the full word "advertisement" making it slightly more visually prominent. The *Basic Prompt* with the general-purpose model also removed the little "ad" label completely, but not the ad itself, despite claiming "Advertisements have been removed" in the comments. This essentially made the pattern more deceptive as now the ad was not only disguised by appearing visually very similar to non-sponsored products, but not being visually completely indistinguishable.

Several other Designs were never fully neutralized, but in those cases, this was not because of issues with specific patterns, but instead due to a higher amount of deceptive patterns that were simply never all removed at once.

5.4 Fails

Fails occurred when the LLM returned `original` code for the changes that did not match the actual code in the HTML file. Usually, this was due to minor differences like spaces, single classes, or attributes missing.

Fails seem to occur based on the design, instead of the prompt.

When looking further into why specific LLM changes failed, we found that frequently the exact same fail happened for a design across different prompts. For example, for E03, in total 50 changes failed because a specific attribute was always left out.

The fails of the reasoning model were often fundamentally different from those of the general-purpose model.

The reasoning models also often struggled with HTML character entities, instead replacing them with their Unicode equivalents. We observed this once with the *Prompt Chaining* technique, but otherwise it never happened for the general-purpose model, despite it being a problem for all 4 reasoning model conditions. We also had a higher failure rate for Reasoning Models due to them replacing inner elements or attributes with Regular Expressions like `[\s\S]*?`. Interestingly, although this was common in all other prompts of the Reasoning Model, it never occurred for the *Reasoning prompt* with high reasoning effort. Instead, here we had a design where it replaced all inner parts that were not relevant with "[...]", which also caused those changes to fail. Apart from this case, the prompt also behaved rather curiously regarding other fails. While, usually, fails occurred due to minor changes like missing spaces or singular classes or attributes being off, in this condition, entire class lists were missing or strongly off. In one case, it even skipped an entire other nested HTML element, even without any attempt to replace it with a placeholder.

In some cases, however, the fails were not the result of hallucinations but occurred because a previous change modified the full HTML page so that the `original` code part in the change was no longer featured in the HTML file.

This happened, for example, when multiple identical deceptive elements occurred on a web page. In such cases, the LLM returned the same exact change multiple times. But our reconstruction process replaced all instances occurring in the HTML file already on the first change, causing all upcoming identical changes to fail. In cases where a deceptive element was a child element of another deceptive element, the change of the child element sometimes failed because it was already removed or modified when the parent element was changed. In one case of the *Basic Prompt* in the reasoning model and low reasoning effort changes also failed because a previous change replaced all closing `div`-tags. This broke the entire page and made any following changes impossible.

Chapter 6

Discussion

6.1 Best Strategies - Research Questions

Our results revealed the strengths and weaknesses of the different approaches. In order to properly consider all the findings affecting RQ 1, we will first consider RQs 2 to 4.

6.1.1 RQ 2: Which prompting techniques are most suitable for this task?

Few-Shot Chain-Of-Thought Prompting was the overall best performing pormpting strategy.

Firstly, we found that out of the prompting techniques we used on GPT-4o, *Few-Shot Chain-Of-Thought Prompting* resulted in the best performance of the model for removing manipulation. This technique was also the only one with a sucess rate of 1 and only broke the functionality of one page, which was the overall best received functionality score, albeit having been achieved by multiple different prompts.

Few-Shot Chain-of-Thought Prompting was only outscored by another prompting technique in the *Design, Layout, and Context*, where *combination* received an average score that was 0.08 higher than that of *Few-Shot Chain-of-Thought* Prompting. And while all prompting techniques outscored the *Advanced Prompt* in the *Design, Layout, and Context Score*,

Persona Prompts yielded the worst results of the techniques. Two of three personas even scored worse than the *Average Prompt* in both the *Manipulation* and *Functionality*. *Prompt Chaining* scored remarkably well in the *Manipulation* but severely compromised the design, layout, and context of the original page.

Overall, we conclude that, except for certain personas, all explored prompting techniques achieved an improvement over the *Advanced Prompt* for the removal of manipulation, but some with varying performance regarding the maintenance of design, layout, context, or functionality of the page. The most suitable out of our investigated techniques was *Few-Shot Chain-of-Thought Prompting*.

6.1.2 RQ 3: How does fine-tuning the model to a task-specific dataset influence the results?

Fine-Tuning even with a small dataset, already strongly improves the model's performance, leading to the best results overall.

Fine-Tuning the GPT-4o model, even on our small dataset of only 106 examples and with only two iterations, already massively improved the performance on the evaluation set as well, leading to the highest *Manipulation* of all conditions by a margin of 0.6. The *Design, Layout, and Context* was also the second highest overall.

Another big advantage of the fine-tuned model was that it correctly identified when changes were not necessary or impossible, which almost never happened with only prompting techniques. In these cases, no manipulative elements can get removed, but if the model reports that the element is deceptive, but removal does not work, it would be possible to fall back to an alternative countermeasure. This could be highlighting and informing the user about this deception as investigated by Schäfer et al. [2023].

6.1.3 RQ 4: What challenges and advantages does a reasoning model present in comparison to a general-purpose model?

Reasoning models returned more effective and efficient changes. With high reasoning effort and a prompt specialized for reasoning models, the model outperformed all GPT-4o prompting techniques.

Reasoning models allow for a generally better performance compared to the regular GPT-4o model, even if the latter is used with prompting techniques. Even the simpler *Basic Prompt* with the reasoning model outperformed the *Advanced Prompt* if the effort assigned to the reasoning process is high enough. Our specialized *Reasoning Prompt* with high reasoning effort reached the overall second-best performance. We also observed that the reasoning model tended to change the CSS style more, while the general-purpose model instead rather removed elements or changed classes to adjust the style. The latter, however, bears the risk of also affecting functionality. The reasoning model's approach of adding rules was more efficient as it changed all instances of a deceptive pattern with one change.

Reasoning models struggled with unremovable deception and tried to remove them, for which they hallucinated alternative options.

But the reasoning models and especially the latter combination produced more random hallucinations or failures that would be hard to systematically counter. They also did not work at all with unremovable deceptive patterns and instead hallucinated solutions which would essentially be BAIT AND SWITCH deception. In one instance, the *Basic Prompt* with low reasoning effort also did not return changes but instead asked for further clarifications on what is considered manipulative. When using a reasoning model, a technical countermeasure would have to be able to handle such requests.

6.1.4 RQ 1: Can a customized LLM remove deception from web pages without compromising design, layout, context or functionality?

A customized LLM can not yet reliably remove all deception from web pages without compromising design, layout, context, or functionality on the first try, but our techniques have shown strong improvements.

In conclusion, we did not find an approach that already consistently and reliably removed all manipulative elements without compromising design, layout, context, or functionality for all evaluated pages or elements on the first try. However, we found immense potential in the fine-tuned model. Additionally, a reasoning model with

high reasoning effort and our *Reasoning Prompt*, as well as the *Few-Shot Chain-of-Thought Prompt* for our base GPT-4o model, showed promising results.

6.2 Challenges

6.2.1 Challenges Encountered in Related Work

Unlike in related work, our model did not accidentally destroy page functionality. If this was the case, it was done deliberately or a byproduct of the whole element being broken.

Li et al. [2023] described several struggles they had with their LLMs. For one, they described how the LLM sometimes returned visually identical elements, but without the functionality. This was a reason why we included our *Functionality Score*. But in our examples, the overall functionality remained intact. Even in the cases in which it was broken, this was frequently done deliberately. This happened because the model misjudged certain features or even the mere existence of JavaScript in the code as deceptive, or it resulted as a byproduct of the LLM breaking the whole page. We suspect two reasons why we did not encounter the problem encountered by Li et al.: On one hand, the general functionality of our pages was already restricted due to it not being extractable. Elements like dropdowns, which Li et al. [2023] exemplify their functionality problem on, were usually not functional on our pages. But that even our manually created designs with intact functionality did not encounter these problems could be due to another suspected factor: Using structured output and having the model only return the changes minimized the opportunity for hallucinations and mistakes like accidentally leaving certain features out, as such cases were filtered out by the change failing. In fact, we quite often observed changes fail as the LLM "forgot" about attributes in the HTML elements that would have been essential for functionality. But as these changes automatically failed, such elements remained untouched.

Our models rarely hallucinated entirely new elements.

Similarly, we also did not encounter nonsensical new components that were also described by Li et al. [2023]. However, especially the GPT model was generally reluctant to create new elements, instead focusing on changing or delet-

ing existing elements. Again, we attribute this behavior to the structured output, which enforced this approach of only neutralizing by changing existing code to something else.

Similar to findings by Shin et al. [2025] we observed the models being overly critical of design choices that a user would probably not notice or dismiss given the context.

We recognized some traces of the *Hyper-Accuracy Distortion* discovered by Shin et al. [2025], where the LLM overly focused on little aspects that were not really noticeable to the users. In our case, the models changed trivial design choices that a user would not have noticed. An example of this would be a change in E09, which featured a blue hyperlink. Instead of the default blue associated with links on websites, the color in the original design was slightly changed to a blue hue that is more consistent with the overall design of the element. This difference was minimal and not noticeable to us at first, but was still seen by the LLM as problematic and changed. This example shows how the models lacked the ability to properly judge the design through its context. This was also frequently the case for E21, which featured a button that had a gradient background. The models that returned explanations criticized this as "visual emphasis" that draws attention, removed all colors, and, interestingly, even smoothed the buttons' rounded edges. Again, similar to Shin et al.'s findings, here the LLM overly focused on the individual features without taking the context into account. However, while we consider the colors of the button and especially the rounded corners as non-problematic, this might be subjective.

We found the same problem with an LLM flipping the meaning of TRICK QUESTION as Schäfer et al. [2025].

Lastly, we encountered the same problem as Schäfer et al. [2025] where a model intended to fix the TRICK QUESTION of our unremovable opt-out design (E05) and instead flipped the meaning in the displayed label, while the user would have to act exactly the opposite way than claimed for the desired results. We also had some instances of FALSE HIERARCHY, where the LLM hallucinated further options.

6.2.2 Challenges Encountered by Us

Functionality

LLMs struggled with changes that would require the page to be functional.

One big challenge in using LLMs with real-world pages is

their access to page functionality. In the majority of our examples, the page functionality was not extractable from the website, as we only had access to the code on the client side, and most of it was handled on external servers. Due to this, our models often had to understand how the web page works based on the HTML and CSS parts alone. This resulted in changes that would need to change the page functionality as well to properly work, but the LLM only changed what was displayed to the user. For example, E04 featured a list of seating options for concert tickets. This list was sorted "by recommended" as indicated by a label stating so at the top. The LLM identified this default sorting as problematic, and instead frequently changed the label to "Sort by Price", but kept the order of the elements and did not actually change the sorting.

Due to the LLM not having access to the page functionality, it might unknowingly change classes that are needed for it.

Another problem caused by the missing access to functionality is that an LLM might miss its dependencies. Both models, but especially the reasoning model, changed the style of elements by modifying the `class`-attribute and removing classes associated with that style. However, class names can also be used in `script`-elements to select components for further changing them dynamically. Due to this, removing the class attributes not only cuts the connection of the element to certain styles but also to possible functionality. As our models usually did not have access to functionality in cases where they removed classes, we are not certain whether they would still do so when they are otherwise required.

Full Websites

The LLMs' handling of full web pages is severely restricted by the size of their context window.

When preparing our *Evaluation Set*, we normally had to massively reduce the page content to fit the token limit of our models. GPT-4o has a maximum context window of 128,000 context window and allows 16,384 max output tokens. The latter makes it impossible for most neutralized web pages to be returned in full. This is remedied by the usage of *structured output*, but the issue of the limited context window still persists. The problem of the restricted con-

text window was also identified as the bottleneck of HTML understanding by Gur et al. [2023].

Splitting a page to fit a smaller context window introduces problems of deceptive patterns not being properly identified or neutralized.

To remedy this, a page could be split into subparts and passed to the model sequentially. But this raises the problem of how this split should occur. However, it might be necessary for the LLM to "see" the rest of the web page to properly recognize deceptive patterns. Deceptive patterns like FALSE HIERARCHY occur when one element is presented in a particular way relative to another one. If the split occurred between those two elements, an LLM would see one element first and then another one afterwards in an entirely new context, which could make it hard to properly recognize the deceptive pattern. Additionally, it might be necessary to access other web elements for handling the removal. Li et al. [2023] separated their web pages into chunks, but then had problems with hallucination if content that needed to be manipulated was not included in the current one. We had a similar concern regarding this problem when there were different prompts for deceptive pattern analysis and removal in the *Chaining* and *Combination* prompts, and also observed a similar case with the latter. E15 features a pop-up asking to allow notifications. Here, the decline button features CONFIRMSHAMING. However, while the explanation correctly included the pressure of the decline button, the analysis step only returned the accept button element. This led the removal step to change the phrasing of that button instead of the CONFIRMSHAMING.

LLMs might also struggle with the amount of deceptive and irrelevant elements in full web pages.

On top of that, our models also had problems with recognizing deceptive patterns on full web pages that have a lot going on. E02 was one of our more complex pages. It contained a whole web page, with multiple deceptive patterns in its main content, but also a cookie banner with FALSE HIERARCHY. The *Combination Prompt* was the only condition that changed the banner, and even here, the deceptive pattern was not correctly neutralized: Instead of making the existing reject button visually equivalent to the accept button, it hallucinated a new one and kept the existing button the same. Due to the high complexity of the web page, we suspect that the poor performance was caused by the abundance of other "distracting" elements in the page.

Failed Changes

The repeated instances of failing changes might be due to how LLMs function in general and, thus, would not be affected by prompting techniques.

Our discovery of the exact same changes failing for the same design across different prompts is understandable when considering how the responses are generated. As explained in Chapter 2.1, the probability distribution for the next token prediction is calculated using the context window. This would mean that if certain patterns are repeated across the context window, like certain class names appearing often, the likelihood of the LLM finishing this pattern when the class name starts similarly is rather high. This would be especially problematic for real web pages, which often use the same naming pattern, but only slightly changed. This can most likely not be fixed by prompting techniques, which could be why we observed this behavior across different prompt, but it might be affected by using higher `temperature` values, making the next token prediction less deterministic.

Some fails are fixable or in themselves non-problematic.

However, while this example reduces our performance, other reasons for changes failing are not as problematic. For example, some changes failed because the model returned placeholders or regular expressions in order to skip the code of irrelevant elements. This caused our reconstruction process to fail due to our reconstruction method using `string.replace()`, which requires exact matches. Instead, one could use an approach to utilize the regular expressions or skip the placeholders, allowing the changes to pass. In other cases, the changes failed because the element was already modified due to a previous change. As in this case, the manipulation was usually already neutralized, the fail was not an issue.

Eliminating hallucinations is most likely not possible.

Overall, we can most likely not achieve a customized model that does not hallucinate. While LLMs are usually tested on hallucination benchmarks prior to their release [OpenAI et al., 2024,], recent research has criticized the effectiveness of this. Kalai et al. [2025] propose that a general restructuring of the training process is necessary to eliminate hallucinations instead, as they are the consequences of the LLMs being trained to be good at test-taking, which achieves better results when guessing and penalizes uncertainty.

6.3 Complications with Removal as a Countermeasure

Complete removal of deceptive patterns as a countermeasure poses some questions and problems about what should and can be removed.

6.3.1 What needs to be removed?

The models sometimes changed design choices that were not intended as manipulative.

First of all, opinions might differ on what qualifies as manipulative and should be removed. This problem was especially apparent in the results of the basic prompt of the GPT-4o model, where the LLM often modified elements that were not intended by us.

This could lead to broken pages or compromise the aesthetics and integrity of the page.

Here, the LLM often classified the existence of `script`-tags or function calls in the HTML code as manipulative and consequently removed all functionality from the web page. This failure in judgment of the LLM is quite severe and has lasting consequences. It makes the entire web page unusable, which would be critical in an in-the-field application. A less severe but also problematic misjudgment was that often the use of color was regarded as "visual emphasis" and as such considered manipulative. It is true that color is often utilized for deceptive patterns like VISUAL PROMINENCE or FALSE HIERARCHY to emphasize them. But color in itself is not automatically manipulative when used consistently, and changing it not only deteriorates the aesthetics but also compromises the integrity of the page.

This frequently happened for fair designs.

This behavior was especially prominent in fair designs. This might be due to the model still "searching" for any form of manipulation. Since it could not find anything actually manipulative, it focused on these minor design choices, or useful features like the wishlist in the fair shopping page choices, interpreting them as deceptive.

Differing opinions on what constitutes deception can also be seen in related work.

But the differing opinions of what is manipulative are not only a problem of large language models. Soe et al. [2022] also pointed out how the automatic detection of deceptive

patterns is challenged by the lack of a consensual definition of dark patterns across different disciplines. Additionally, Lupiáñez-Villanueva et al. [2022] noted how distinguishing between manipulation by deceptive patterns and legitimate persuasion attempts is also problematic.

6.3.2 What should we remove?

It is questionable how far an LLM should interfere with the original web page and its intent.

Page integrity also opens up the question of what *should* get removed. On one hand, it is already risky to trust the LLM to make changes to the web page without the user noticing. Distrust in such technologies can also be seen in Schäfer et al. [2023], where users specifically did not want silent changes to be made to their pages. But often the web pages were deliberately designed like this and at some point it is debatable how far the page should be changed afterwards. One example of this was E22 in our evaluation set. This was extracted from a political donation page. It features various straightforward deceptive patterns like e.g.ACTIVITY MESSAGES, BAD DEFAULTS, and VISUAL PROMINENCE. But it also features a message, supposedly by the U.S. American president himself, that was often recognized by the LLM as manipulative and changed. This judgment is absolutely fair as it constitutes PARASOCIAL PRESSURE and we also recognize the coercive tone of language used. On the other hand, this was the text chosen to influence and persuade people to donate, and it could be problematic to change the words that are supposedly a direct quote. Or in some cases, complete features in themselves are manipulative. For example, the donation page also features a "moneybomb" where they ask for an additional donation at a set date in the future. This was also recognized as manipulative by different conditions and removed completely. However, one might argue that, although it is manipulative, it is an integral feature of this web page and should not be removed. Additionally, the tone of the web page and the prevalence of deceptive patterns also leave a certain impression, and weakening this can cause a user to misjudge the page. This problem was also already addressed by Schäfer et al. [2023], who identified the problem that people might consider shady web pages that employ var-

ious dark patterns as untrustworthy, while removing these "red flags" can lead to them being tricked.

6.3.3 What can we remove?

Not all deceptive patterns are removable.

Lastly, removal as a countermeasure also poses problems of feasibility as not all deceptive patterns are removable. Encouragingly, our models were able to recognize and remove CONFIRMSHAMING, which was classified as undetectable by Stavrakakis et al. [2021], but they also struggled with PRIVACY ZUCKERING like Stavrakakis et al. [2021] predicted and with TRICK QUESTION similar to Schäfer et al. [2025]. Overall, many of the issues of impossible removal stem from the lack of access to specific functionalities as discussed in Chapter 6.2.2. Additionally, as Gray et al. [2025] recently addressed, the effect of deceptive patterns can also stem from the deception being spread throughout multiple pages. This would need the model to navigate through several pages. While models have been shown to be capable of this [Gu et al., 2025], it would probably require a much larger context window or other solutions to keep track of deception on previous pages.

6.4 Dangers

6.4.1 Integrated Deception

LLMs might be trained to accidentally add elements, including deceptive patterns, in their pre-training.

One thing that stood out to us was that two of the reject buttons that were added by the o4-mini model were grayed out in comparison to the accept button, creating a FALSE HIERARCHY. The page featured no other similarly styled element, meaning the LLM could not have taken such a design as a pattern from its context window. Instead, given that the button itself was entirely hallucinated, the LLM most likely gained this design from its own understanding of what a cookie banner should look like based on its pre-training. This reflects findings of Krauß et al. [2025] and Chen et al. [2025], who investigated web elements gen-

erated by LLMs and found that they frequently contained deceptive patterns even when not explicitly prompted to include such practices.

6.4.2 Deliberate Deception

Optimizing an LLM for deceptive pattern removal could simultaneously optimize them for deceptive pattern *addition*,

There might however, be cases in which the LLM deliberately adds deception. The *Waluigi effect* describes instances in which LLMs explicitly trained to achieve one thing are also more likely to achieve the exact opposite.

Definition: *Waluigi Effect*

WALUIGI EFFECT:
After you train an LLM to satisfy a desirable property P, then it's easier to elicit the chatbot into satisfying the exact opposite of property P[1].

In our case, this would mean that an LLM optimized for deceptive pattern removal would also be somewhat optimized for *adding* manipulation.

The uploaded data of Schäfer et al. [2025] already featured the results of prompting the unmodified to make a page *more* manipulative, which resulted in the addition of various malicious deceptive patterns. Since this can already be achieved with a model not customized for deceptive pattern removal, according to the *Waluigi effect*, our fine-tuned model would be even more effective at adding manipulation.

An extreme CONFIRMSHAMING pattern caused the model to refuse an answer.

Before publication, LLMs go through the process of *Alignment Fine-Tuning*, which consists of prompting them to produce undesired responses and then updating their internal parameters so that such responses are no longer returned [Ouyang et al., 2022]. Additionally, model responses have to pass a content filter that causes the model to refuse certain requests. We have seen its effect, when the model refused its answer for an extreme CONFIRMSHAMING case (see Figure 6.1). We managed to overcome this by reducing the content filter for our model, but it shows an issue with our structured output approach: It needs the model

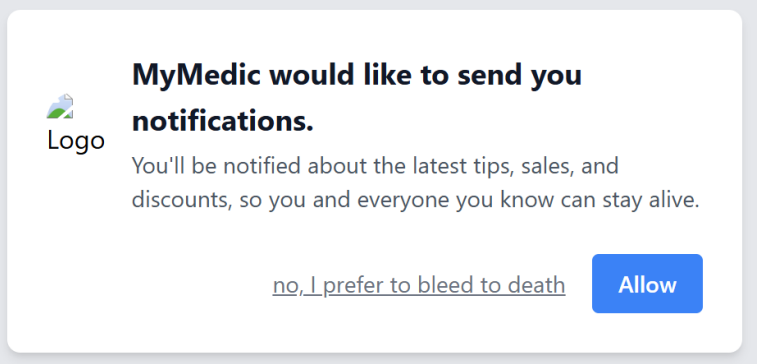

Figure 6.1: An extreme case of CONFIRMSHAMING. The model refused to answer for this design due to its content filter.

to return the code for deceptive patterns in the `original` field, in order for the replacement to work. For this, we need the LLM to be able to output even extreme cases of deceptive patterns. Now, this only happened to us in one design due to the rather radical, violent phrasing of the CONFIRMSHAMING. But in an ideal world, LLMs would be restricted by other measures in order to prevent the output of any deceptive patterns at all, which would cause a model to refuse all requests with our approach that would return changes.

Our approach could be adapted so that it would not trigger the content filter.

However, it would be quite easy to adapt our approach into a form that does not require it to output the entire deceptive element. We decided on this as outputting the whole web page predominantly exceeded the output limit. But before settling on our approach, we also initially tried having the LLM reference the position in the HTML file where the code should be changed. However, this did not work as the position values were often incorrect. Another idea we had was assigning IDs to the different elements during preprocessing and then referencing those IDs for the changes. While we did not test this out and instead moved on with our approach, we expect this approach to work and overcome content filters.

6.5 Our Recommendations

Overall, we conclude the following recommendations for using an LLM for deceptive pattern removal:

1. **Use a fine-tuned model**. Even if the training set is rather small and the fine-tuned process is not often iterated, the performance might already improve immensely.
2. **Get the model to use reasoning** either by using the *Chain-of-Thought* technique, ideally combined with *Few-Shot* examples or using a reasoning model.
3. If costs and time requirements allow for it, **let the reasoning model use high reasoning effort**.
4. Do not bother with Persona Prompting or Prompt Chaining on their own; instead, **combine different prompting techniques**.
5. Have the model only answer as parts of the original page and **use *structured output* to reduce the necessary output tokens** and facilitate further processing of the model's answer. This might, however, make it necessary to lower the content filter.

6.6 Limitations and Future Work

Our approach was limited by the token limit and missing functionality.

The biggest limitations were connected with the usage of real websites. For one, we had a limited selection of web pages we could use, due to our token limits. Even so, we had to further reduce the size of the pages by deleting some of their elements. Additionally, as we could only access client-side source code, we had no access to the functionality behind most pages.

Further, for the fine-tuning process, we used a rather small dataset and only iterated on the process twice without specific adjustments of fine-tuning parameters. While we

still achieved promising results, by further expanding the dataset, the model can be further optimized for our task. Additionally, future work could explore LLM-as-a-Judge and use it for *Reinforcement Fine-Tuning* a reasoning model. This approach could be especially useful for our task as it is more complex with different ways to solve it, making it ideal for reinforcement fine-tuning. We could also combine our approach with further reinforcement fine-tuning and LLM-as-a-Judge, as well as with the iterative approach of Schäfer et al. [2025] to further improve the overall results. Another aspect for potential improvement would be to provide the model with more information. Mills and Whittle [2023] found using images to be the most effective way for providing an LLM with web page design regarding deceptive patterns. As many models support image input, it would be possible to add web page screenshots to our prompts. Retrieval-Augmented Generation, similar to Boumber et al. [2024], could also be used to provide the model with more examples and definitions of deceptive designs. It could be especially interesting to explore the use of different taxonomies for this. Boumber et al. [2024] has shown the impact the definition of deception had on the performance of an LLM. Since the taxonomy landscape is so diverse for deceptive patterns, it would be interesting to see whether an LLM would perform better with an attribute [Mathur et al., 2021], definition [Gray et al., 2024], or solution-based [Potel-Saville and Da Rocha, 2024] taxonomy. Especially the latter viewpoint could be beneficial to explore with a reasoning model. This is because those perform best when told what the result should look like, instead, which was specifically the intent behind the Potel-Saville and Da Rocha's fair pattern taxonomy.

Future work could expand on fine-tuning models with different fine-tuning processes and improve results by providing more information.

Lastly, future work could concern itself with what our findings on the challenges and advantages of using a reasoning model or general-purpose model mean for GPT-5[2], which combines both types in a unified model. It could harness the potential of prompting techniques with better performance through high-effort reasoning. GPT-5 also has a much higher context window of 400.000, which is almost double the size we had available for our models. With this,

GPT-5 could also be explored for our task in future work.

[2] `https://openai.com/index/introducing-gpt-5/`, *last accessed September 27, 2025*

the model could handle substantially larger web pages and take more of or even the entire page into account for the output generation.

Chapter 7

Summary

In the following chapter we conclude this thesis by providing a summary of our thesis and its contributions

7.1 Summary and Contributions

We explored the optimization of large language models for deceptive pattern removal.

In this thesis, we explored the optimization of large language models for deceptive pattern removal from websites. Especially, we focused on whether an LLM is capable of removing deceptive patterns from websites without compromising design, layout, context, or functionality of the original page, and how model choice, prompting, and fine-tuning influence those results. For this, we used OpenAIs GPT-4o and o4-mini models and affected their performance using prompts adhering to recommended best practices and prompting techniques. We also adjusted the temperature of the general-purpose model and the reasoning effort for the reasoning model. Finally, we used supervision fine-tuning on a GPT-4o model with a dataset of 106 examples. In total, this resulted in 17 different conditions that we evaluated across 25 selected web pages. We based our evaluation on the consistency with the original page's design, layout, or context, on whether the manipulation was fully or partially removed or enhanced, or a new manipulation added, and whether the page was still functional.

Lastly, we analyzed instances where the changes proposed by the model failed due to hallucinations and investigated their cause.

Fine-tuning a model achieved the overall best results.

We found that despite our dataset being rather small and only fine-tuning for two iterations, our fine-tuned model achieved the best performance for deceptive pattern removal. This was followed by the usage of a reasoning model with a specialized prompt that allowed us to use high reasoning effort for this task. Finally, we observed that out of the investigated prompting techniques, *Few-Shot Chain-of-Thought Prompting* performed best, while *Persona Prompting* and *Prompt Chaining* achieved noticeably low scores.

We contribute insights into the advantages of the different models and prompting techniques, address challenges we and related work faced with utilizing LLMs for this task, and discuss general problems of removal as a countermeasure and LLMs adding deception accidentally or when being prompted to do so. Our findings resulted in 5 recommendations for customizing LLMs for deceptive pattern removal. Finally, we discuss further steps that could be taken to improve the LLM's performance.

Appendix A

Prompts

The following appendix includes all prompts used for this thesis.

A.1 Basic Prompt

```
Neutralize all manipulative elements.
```

A.2 Advanced Prompt

```
You are given the HTML and CSS for a web page or
    component. Your task is to identify and eliminate
    every element, style rule, label, layout choice,
    or interactive behavior that is intended to
    steer, pressure, or mislead usersespecially
    so-called dark patterns or deceptive design. If
    the element is non-essential remove it. Otherwise
    replace it with a non-manipulative version of it.
    All remaining information must be presented
    transparently and unambiguously.

Instructions:
```

```
1. Parse the HTML/CSS and identify every element that
   could manipulate the user. Focus on these groups:
   A. Textual Cues
         Labels, badges, messages, hints,
         placeholders, microcopy
         Claims of scarcity, urgency or exclusivity
         Framing that understates costs or overstates
         benefits
         Phrasing that pressures emotionally
   B. Visual Emphasis
         Color, contrast, size, font weight, icons or
         graphics used to draw undue attention
         Layout deviations, z-index stacking, or
         other styling that creates a false hierarchy
   C. Number Formatting
         Strikethroughs of original prices,
         highlighting of partial or starting at
         prices
         Any presentation that hides total costs
   D. Interactive Controls
         Pre-selected defaults, opt-outs, or obscured
         controls
         Button labels or affordances that obscure
         true outcomes
   E. Presentation of Information
         Additional mandatory fees shown late or in a
         less prominent way
         Critical details hidden behind extra steps,
         small text or low-contrast presentation

2. For each identified manipulative element:
      If it is not essential for site functionality
      and does not provide information critical to
      an informed decision: Remove the element
      entirely
      Otherwise:
    - Neutralize any steering or pressure tactics
        (e.g. remove urgency markers, normalize
        styling).
        Ensure full transparency: all critical
        information (especially pricing) must be
        immediately and clearly visible.
        Identify the true total cost (including all
        mandatory fees and charges) and update the
        page so that this total is displayed
        immediately and most prominently (e.g.,
        replace any smaller base price at the top
        with the full-cost figure). Relegate
```

```
        individual fees to a clearly labeled
        breakdown elsewhere.

3. If you encounter a manipulative element that
    cannot be altered without inventing new
    functionality or content, return exactly:
    Changes  not  possible

4. If you find no manipulative or deceptive elements,
    return exactly:
    No  changes necessary

Rules:
- Do not obscure or downplay manipulative facts:
    either remove them or neutralize their impact.
- Only keep elements that are essential for
    functionality or to allow users an informed
    opinion.
- After edits, all remaining information must be
    fully transparent and unambiguous.
- If manipulative elements were identified and all
    required changes were made, leave comments empty.
- For each change, return the original HTML parts in
    full and exact as they were before
```

A.3 Few-Shot Prompt

In order to keep the prompt readable, we replaced the actual HTML code input of the examples with placeholders.

```
    You are given the HTML and CSS for a web page or
        component. Your task is to identify and
        eliminate every element, style rule, label,
        layout choice, or interactive behavior that
        is intended to steer, pressure, or mislead
        usersespecially so-called dark patterns or
        deceptive design. If the element is
        non-essential remove it. Otherwise replace it
        with a non-manipulative version of it. All
        remaining information must be presented
        transparently and unambiguously.

Instructions:
```

```
1. Parse the HTML/CSS and identify every element that
   could manipulate the user. Focus on these groups:
   A. Textual Cues
         Labels, badges, messages, hints,
         placeholders, microcopy
         Claims of scarcity, urgency or exclusivity
         Framing that understates costs or overstates
         benefits
         Phrasing that pressures emotionally
   B. Visual Emphasis
         Color, contrast, size, font weight, icons or
         graphics used to draw undue attention
         Layout deviations, z-index stacking, or
         other styling that creates a false hierarchy
   C. Number Formatting
         Strikethroughs of original prices,
         highlighting of partial or starting at
         prices
         Any presentation that hides total costs
   D. Interactive Controls
         Pre-selected defaults, opt-outs, or obscured
         controls
         Button labels or affordances that obscure
         true outcomes
   E. Presentation of Information
         Additional mandatory fees shown late or in a
         less prominent way
         Critical details hidden behind extra steps,
         small text or low-contrast presentation

2. For each identified manipulative element:
      If it is not essential for site functionality
      and does not provide information critical to
      an informed decision: Remove the element
      entirely
      Otherwise:
    - Neutralize any steering or pressure tactics
        (e.g. remove urgency markers, normalize
        styling).
        Ensure full transparency: all critical
        information (especially pricing) must be
        immediately and clearly visible.
        Identify the true total cost (including all
        mandatory fees and charges) and update the
        page so that this total is displayed
        immediately and most prominently (e.g.,
        replace any smaller base price at the top
        with the full-cost figure). Relegate
```

```
        individual fees to a clearly labeled
        breakdown elsewhere.

3. If you encounter a manipulative element that
    cannot be altered without inventing new
    functionality or content, return exactly:
    Changes  not  possible

4. If you find no manipulative or deceptive elements,
    return exactly:
    No  changes necessary

Rules:
- Do not obscure or downplay manipulative facts:
    either remove them or neutralize their impact.
- Only keep elements that are essential for
    functionality or to allow users an informed
    opinion.
- After edits, all remaining information must be
    fully transparent and unambiguous.
- If manipulative elements were identified and all
    required changes were made, leave comments empty.
- For each change, return the original HTML parts in
    full and exact as they were before

Example 1:
   Input:
   [HTML code 1]

   Output:
      Changes:
         Original:
         <button class="buy-button-standard">Buy
             Tickets</button>
         Replacement:
         <button class="buy-button">Buy
             Tickets</button>


Example 2:
   Input:
   [HTML code 2]

   Output:
      Comments:
         Changes not possible

Example 3:
   Input:
```

```
[HTML code 3]

Output:
   Changes:
      Original:
      <span data-testid="discount-pill"
          class="css-1nt6bqz e1cq2ihg2" >-30%</span
          >
      Replacement:

      Original:
      <span dir="auto" role="group"
          aria-label="220" class="css-92woig
          e139ay0z0" ><span type="none"
          class="money-currency css-49gx66
          e139ay0z1" aria-hidden="true" ></span
          ><span type="none" class="money-integer
          css-2h7jcq e139ay0z2" aria-hidden="true"
          >220</span ></span >
      Replacement:
```

A.4 Chain-of-Thought Prompt

```
    You are given the HTML and CSS for a web page or
        component. Your task is to identify and
        eliminate every element, style rule, label,
        layout choice, or interactive behavior that
        is intended to steer, pressure, or mislead
        usersespecially so-called dark patterns or
        deceptive design. If the element is
        non-essential remove it. Otherwise replace it
        with a non-manipulative version of it. All
        remaining information must be presented
        transparently and unambiguously.

Instructions:

1. Parse the HTML/CSS and identify every element that
    could manipulate the user. Focus on these groups:
   A. Textual Cues
         Labels, badges, messages, hints,
         placeholders, microcopy
         Claims of scarcity, urgency or exclusivity
         Framing that understates costs or overstates
         benefits
```

```
        Phrasing that pressures emotionally
   B. Visual Emphasis
        Color, contrast, size, font weight, icons or
        graphics used to draw undue attention
        Layout deviations, z-index stacking, or
        other styling that creates a false hierarchy
   C. Number Formatting
        Strikethroughs of original prices,
        highlighting of partial or starting at
        prices
        Any presentation that hides total costs
   D. Interactive Controls
        Pre-selected defaults, opt-outs, or obscured
        controls
        Button labels or affordances that obscure
        true outcomes
   E. Presentation of Information
        Additional mandatory fees shown late or in a
        less prominent way
        Critical details hidden behind extra steps,
        small text or low-contrast presentation

2. For each identified manipulative element:
      If it is not essential for site functionality
      and does not provide information critical to
      an informed decision: Remove the element
      entirely
      Otherwise:
    - Neutralize any steering or pressure tactics
        (e.g. remove urgency markers, normalize
        styling).
        Ensure full transparency: all critical
        information (especially pricing) must be
        immediately and clearly visible.
        Identify the true total cost (including all
        mandatory fees and charges) and update the
        page so that this total is displayed
        immediately and most prominently (e.g.,
        replace any smaller base price at the top
        with the full-cost figure). Relegate
        individual fees to a clearly labeled
        breakdown elsewhere.

3. If you encounter a manipulative element that
    cannot be altered without inventing new
    functionality or content, return exactly:
    Changes  not  possible
```

```
4. If you find no manipulative or deceptive elements,
    return exactly:
    No  changes necessary

Rules:
- Do not obscure or downplay manipulative facts:
    either remove them or neutralize their impact.
- Only keep elements that are essential for
    functionality or to allow users an informed
    opinion.
- After edits, all remaining information must be
    fully transparent and unambiguous.
- If manipulative elements were identified and all
    required changes were made, leave comments empty.
- For each change, return the original HTML parts in
    full and exact as they were beforeThink step by
    step and always describe your approach for each
    change in the "explanation" field. If no changes
    were possible, add your explanation to the
    'comments' field
```

A.5 Few-Shot Chain-of-Thought Prompt

In order to keep the prompt readable, we replaced the actual HTML code input of the examples with placeholders.

```
    You are given the HTML and CSS for a web page or
        component. Your task is to identify and
        eliminate every element, style rule, label,
        layout choice, or interactive behavior that
        is intended to steer, pressure, or mislead
        usersespecially so-called dark patterns or
        deceptive design. If the element is
        non-essential remove it. Otherwise replace it
        with a non-manipulative version of it. All
        remaining information must be presented
        transparently and unambiguously.

Instructions:

1. Parse the HTML/CSS and identify every element that
    could manipulate the user. Focus on these groups:
   A. Textual Cues
```

```
        Labels, badges, messages, hints,
        placeholders, microcopy
        Claims of scarcity, urgency or exclusivity
        Framing that understates costs or overstates
        benefits
        Phrasing that pressures emotionally
  B. Visual Emphasis
        Color, contrast, size, font weight, icons or
        graphics used to draw undue attention
        Layout deviations, z-index stacking, or
        other styling that creates a false hierarchy
  C. Number Formatting
        Strikethroughs of original prices,
        highlighting of partial or starting at
        prices
        Any presentation that hides total costs
  D. Interactive Controls
        Pre-selected defaults, opt-outs, or obscured
        controls
        Button labels or affordances that obscure
        true outcomes
  E. Presentation of Information
        Additional mandatory fees shown late or in a
        less prominent way
        Critical details hidden behind extra steps,
        small text or low-contrast presentation

2. For each identified manipulative element:
      If it is not essential for site functionality
      and does not provide information critical to
      an informed decision: Remove the element
      entirely
      Otherwise:
    - Neutralize any steering or pressure tactics
        (e.g. remove urgency markers, normalize
        styling).
        Ensure full transparency: all critical
        information (especially pricing) must be
        immediately and clearly visible.
        Identify the true total cost (including all
        mandatory fees and charges) and update the
        page so that this total is displayed
        immediately and most prominently (e.g.,
        replace any smaller base price at the top
        with the full-cost figure). Relegate
        individual fees to a clearly labeled
        breakdown elsewhere.
```

```
3. If you encounter a manipulative element that
    cannot be altered without inventing new
    functionality or content, return exactly:
    Changes  not  possible

4. If you find no manipulative or deceptive elements,
    return exactly:
    No  changes  necessary

Rules:
- Do not obscure or downplay manipulative facts:
    either remove them or neutralize their impact.
- Only keep elements that are essential for
    functionality or to allow users an informed
    opinion.
- After edits, all remaining information must be
    fully transparent and unambiguous.
- If manipulative elements were identified and all
    required changes were made, leave comments empty.
- For each change, return the original HTML parts in
    full and exact as they were beforeThink step by
    step and always describe your approach for each
    change in the "explanation" field. If no changes
    were possible, add your explanation to the
    'comments' field

Example 1:
   Input:
   [HTML code 1]

   Output:
      Changes:
         Original:
         <button class="buy-button-standard">Buy
             Tickets</button>
         Replacement:
         <button class="buy-button">Buy
             Tickets</button>
         explanation: There are two types of tickets
             available. The button for the VIP ticket
             is colorful and stands out. The button
             for the standard ticket is gray. The
             color difference creates a false
             hierarchy: The VIP option appears more
             prominent compared to the standard
             ticket, even though they are both options
             a user can choose. This manipulates the
             user in their choice, nudging them
             towards selecting the VIP ticket.
```

```
            Instead, both buttons should look the
            same. The style of the buttons is
            connected to the classes
            "buy-button-standard" and "buy-button".
            By changing the class of the button for
            the standard ticket to "buy-button" the
            button gets the same appearance as the
            button for the VIP ticket.


Example 2:
   Input:
   [HTML code 2]

   Output:
      Comments:
         Changes not possible. Explanation: In this
             cookie banner, the user only has the
             choice to accept the cookies. Giving the
             user no choice is manipulative. In order
             to remove it, I would need to provide an
             option for the user to reject. But with
             the given code, I cannot access and alter
             how the website stores cookies. So it is
             not possible to provide an option to
             reject the cookies.

Example 3:
   Input:
   [HTML code 3]

   Output:
      Changes:
         Original:
         <span data-testid="discount-pill"
             class="css-1nt6bqz e1cq2ihg2" >-30%</span
             >
         Replacement:
         explanation: This dicount badge is very
             visually prominent and attracts the
             user's attention. Additionally,
             displaying percentage discounts can
             exaggerate savings and steer decisions.
             Removing the discount percentage
             completely prevents manipulation.

         Original:
         <span dir="auto" role="group"
             aria-label="220" class="css-92woig
```

```
            e139ay0z0" ><span type="none"
            class="money-currency css-49gx66
            e139ay0z1" aria-hidden="true" ></span
            ><span type="none" class="money-integer
            css-2h7jcq e139ay0z2" aria-hidden="true"
            >220</span ></span >
        Replacement:
        explanation: Displaying discounts can
            exaggerate savings and steer decisions.
            Removing the discount completely prevents
            manipulation.
```

A.6 Prompt Chaining

A.6.1 Analysis Prompt

```
    You are given the HTML and CSS for a web page or
        component. Your task is to identify and
        describe every element, style rule, label,
        layout choice, or interactive behavior that
        is intended to steer, pressure, or mislead
        usersespecially so-called dark patterns or
        deceptive design.

Instructions:

1. Parse the HTML/CSS and identify and describe every
    element that could manipulate the user. Focus on
    these groups:
  A. Textual Cues
        Labels, badges, messages, hints,
        placeholders, microcopy
        Claims of scarcity, urgency or exclusivity
        Framing that understates costs or overstates
        benefits
    - Phrasing that pressures emotionally
  B. Visual Emphasis
        Color, contrast, size, font weight, icons or
        graphics used to draw undue attention
        Layout deviations, z-index stacking, or
        other styling that creates a false hierarchy
  C. Number Formatting
```

```
        Strikethroughs of original prices,
        highlighting of partial or starting at
        prices
        Any presentation that hides total costs
  D. Interactive Controls
        Pre-selected defaults, opt-outs, or obscured
        controls
        Button labels or affordances that obscure
        true outcomes
  E. Presentation of Information
        Additional mandatory fees shown late or in a
        less prominent way
        Critical details hidden behind extra steps,
        small text or low-contrast presentation

2. If you find no manipulative or deceptive elements,
    return exactly:
       No  changes  necessary
```

A.6.2 Removal Prompt

```
    You are given the HTML and CSS and descriptions
        of web components that are intended to steer,
        pressure, or mislead users. Your task is to
        neutralize the deceptive effect of these
        elements. If the element is non-essential
        remove it. Otherwise replace it with a
        non-manipulative version of it. All remaining
        information must be presented transparently
        and unambiguously.

Instructions:
   1. If the element is not essential for site
       functionality and does not provide information
       critical to an informed decision: Remove the
       element entirely

   2. Otherwise:
     - Neutralize any steering or pressure tactics
         (e.g. remove urgency markers, normalize
         styling).
         Ensure full transparency: all critical
         information (especially pricing) must be
         immediately and clearly visible.
         Identify the true total cost (including all
         mandatory fees and charges) and update the
```

```
        page so that this total is displayed
        immediately and most prominently (e.g.,
        replace any smaller base price at the top
        with the full-cost figure). Relegate
        individual fees to a clearly labeled
        breakdown elsewhere.

   3. If you encounter a manipulative element that
       cannot be altered without inventing new
       functionality or content, return exactly:
       Changes  not  possible
   4. If you receive no manipulative elements, return
       exactly:
      "Changes not necessary"


Rules:
- Do not obscure or downplay manipulative facts:
    either remove them or neutralize their impact.
- Only keep elements that are essential for
    functionality or to allow users an informed
    opinion.
- After edits, all remaining information must be
    fully transparent and unambiguous.
- If all required changes were made, leave comments
    empty.
- For each change, return the original HTML parts in
    full and exact as they were before
```

A.7 Combination

In order to keep the prompt readable, we replaced the actual HTML code input of the examples with placeholders.

A.7.1 Combination Analysis Prompt

```
    You are an expert in the neutralization of
        manipulative design and Dark or Deceptive
        Patterns in particular. Think step by step
        and always describe your approach for each
        change in the "explanation" field. If no
```

```
       important elements were found, add your
       explanation to the 'comments' field.
You are given the HTML and CSS for a web page or
    component. Your task is to identify and describe
    every element, style rule, label, layout choice,
    or interactive behavior that is intended to
    steer, pressure, or mislead usersespecially
    so-called dark patterns or deceptive design.

Instructions:

1. Parse the HTML/CSS and identify and describe every
    element that could manipulate the user. Focus on
    these groups:
   A. Textual Cues
          Labels, badges, messages, hints,
          placeholders, microcopy
          Claims of scarcity, urgency or exclusivity
          Framing that understates costs or overstates
          benefits
      - Phrasing that pressures emotionally
   B. Visual Emphasis
          Color, contrast, size, font weight, icons or
          graphics used to draw undue attention
          Layout deviations, z-index stacking, or
          other styling that creates a false hierarchy
   C. Number Formatting
          Strikethroughs of original prices,
          highlighting of partial or starting at
          prices
          Any presentation that hides total costs
   D. Interactive Controls
          Pre-selected defaults, opt-outs, or obscured
          controls
          Button labels or affordances that obscure
          true outcomes
   E. Presentation of Information
          Additional mandatory fees shown late or in a
          less prominent way
          Critical details hidden behind extra steps,
          small text or low-contrast presentation

2. If you find no manipulative or deceptive elements,
    return exactly:
       No  changes  necessary

Example 1:
   Input:
   [HTML code 1]
```

```
   Output:
      ImportantElements:
         elementCode: <button
             class="buy-button-standard">Buy
             Tickets</button>
         elementDescription: A gray button for buying
             the standard ticket.
         explanation: There are two types of tickets
             available. The button for the VIP ticket
             is colorful and stands out. The button
             for the standard ticket is gray. The
             color difference creates a false
             hierarchy: The VIP option appears more
             prominent compared to the standard
             ticket, even though they are both options
             a user can choose. This manipulates the
             user in their choice, nudging them
             towards selecting the VIP ticket.
             Instead, both buttons should look the
             same. The style of the buttons is
             connected to the classes
             "buy-button-standard" and "buy-button".
             By changing the class of the button for
             the standard ticket to "buy-button" the
             button gets the same appearance as the
             button for the VIP ticket.


Example 2:
   Input:
   [HTML code 2]

   Output:
      ImportantElements:
         elementCode: <div
             class="GTXCookieBarstyle__Content-sc-67say9-1
             fQTZzc"><p
             class="GTXTypographystyle__Type-sc-1kk8ybz-0
             PEwXm"><span
             class="GTXCookieBarstyle__TextContainer-sc-67say9-2
             jtyuvy">We use cookies to give you the
             best experience on our website. By using
             our website you agree to our use of
             cookies in accordance with our <a
             href="/en/cookies-policy"
             target="_blank">Cookies
             policy</a>.</span></p><div
             class="GTXCookieBarstyle__ButtonContainer-sc-67say9-3
```

```
            hdpzmm"><button
            class="GTXButtonstyle__Button-sc-1af6gn5-0
            kvLhRv" aria-label="button"
            background="#06038D" border="2px solid
            transparent" padding="0 0.5rem"
            margin="0px 8px" color="#FFFFFF"
            data-testid="cookieAccept"
            type="submit"><span
            class="GTXTypographystyle__Type-sc-1kk8ybz-0
            gnLmnZ">Accept</span></button></div></div>
         elementDescription: A cookie banner with a
            button to accept the cookies.
         explanation: In this cookie banner, the user
            only has the choice to accept the
            cookies. Giving the user no choice is
            manipulative.


Example 3:
   Input:
   [HTML code 3]

   Output:
      ImportantElements:
         elementCode:
         <span data-testid="discount-pill"
            class="css-1nt6bqz e1cq2ihg2" >-30%</span
            >
         elementDescription: A red discount badge
         explanation: This dicount badge is very
            visually prominent and attracts the
            user's attention. Additionally,
            displaying percentage discounts can
            exaggerate savings and steer decisions

         elementCode:
         <span dir="auto" role="group"
            aria-label="220" class="css-92woig
            e139ay0z0" ><span type="none"
            class="money-currency css-49gx66
            e139ay0z1" aria-hidden="true" ></span
            ><span type="none" class="money-integer
            css-2h7jcq e139ay0z2" aria-hidden="true"
            >220</span ></span >
         elementDescription: A price that was striked
            through
         explanation: Displaying discounts can
            exaggerate savings and steer decisions.
```

A.7.2 Combination Removal Prompt

```
    You are an expert in the neutralization of
        manipulative design and Dark or Deceptive
        Patterns in particular. Think step by step
        and always describe your approach for each
        change in the "explanation" field. If no
        changes were possible, add your explanation
        to the 'comments' field.
You are given the HTML and CSS and descriptions of
    web components that are intended to steer,
    pressure, or mislead users. Your task is to
    neutralize the deceptive effect of these
    elements. If the element is non-essential remove
    it. Otherwise replace it with a non-manipulative
    version of it. All remaining information must be
    presented transparently and unambiguously.

Instructions:
   1. If the element is not essential for site
       functionality and does not provide information
       critical to an informed decision: Remove the
       element entirely

   2. Otherwise:
     - Neutralize any steering or pressure tactics
         (e.g. remove urgency markers, normalize
         styling).
         Ensure full transparency: all critical
         information (especially pricing) must be
         immediately and clearly visible.
         Identify the true total cost (including all
         mandatory fees and charges) and update the
         page so that this total is displayed
         immediately and most prominently (e.g.,
         replace any smaller base price at the top
         with the full-cost figure). Relegate
         individual fees to a clearly labeled
         breakdown elsewhere.

   3. If you encounter a manipulative element that
       cannot be altered without inventing new
       functionality or content, return exactly:
       Changes  not  possible
   4. If you receive no manipulative elements, return
       exactly:
      "Changes not necessary"
```

```
Rules:
- Do not obscure or downplay manipulative facts:
    either remove them or neutralize their impact.
- Only keep elements that are essential for
    functionality or to allow users an informed
    opinion.
- After edits, all remaining information must be
    fully transparent and unambiguous.
- If all required changes were made, leave comments
    empty.
- For each change, return the original HTML parts in
    full and exact as they were before

Example 1:
   Input:
      ImportantElements:
           elementCode: <button
               class="buy-button-standard">Buy
               Tickets</button>
           elementDescription: A gray button for
               buying the standard ticket.
           explanation: There are two types of
               tickets available. The button for the
               VIP ticket is colorful and stands out.
               The button for the standard ticket is
               gray. The color difference creates a
               false hierarchy: The VIP option
               appears more prominent compared to the
               standard ticket, even though they are
               both options a user can choose. This
               manipulates the user in their choice,
               nudging them towards selecting the VIP
               ticket. Instead, both buttons should
               look the same. The style of the
               buttons is connected to the classes
               "buy-button-standard" and
               "buy-button". By changing the class of
               the button for the standard ticket to
               "buy-button" the button gets the same
               appearance as the button for the VIP
               ticket.

   Output:
      Changes:
         Original:
         <button class="buy-button-standard">Buy
             Tickets</button>
         Replacement:
```

```
                <button class="buy-button">Buy
                    Tickets</button>
                explanation: There are two types of tickets
                    available. The button for the VIP ticket
                    is colorful and stands out. The button
                    for the standard ticket is gray. The
                    color difference creates a false
                    hierarchy: The VIP option appears more
                    prominent compared to the standard
                    ticket, even though they are both options
                    a user can choose. This manipulates the
                    user in their choice, nudging them
                    towards selecting the VIP ticket.
                    Instead, both buttons should look the
                    same. The style of the buttons is
                    connected to the classes
                    "buy-button-standard" and "buy-button".
                    By changing the class of the button for
                    the standard ticket to "buy-button" the
                    button gets the same appearance as the
                    button for the VIP ticket.


Example 2:
   Input:
     ImportantElements:
         elementCode: <div
             class="GTXCookieBarstyle__Content-sc-67say9-1
             fQTZzc"><p
             class="GTXTypographystyle__Type-sc-1kk8ybz-0
             PEwXm"><span
             class="GTXCookieBarstyle__TextContainer-sc-67say9-2
             jtyuvy">We use cookies to give you the
             best experience on our website. By using
             our website you agree to our use of
             cookies in accordance with our <a
             href="/en/cookies-policy"
             target="_blank">Cookies
             policy</a>.</span></p><div
             class="GTXCookieBarstyle__ButtonContainer-sc-67say9-3
             hdpzmm"><button
             class="GTXButtonstyle__Button-sc-1af6gn5-0
             kvLhRv" aria-label="button"
             background="#06038D" border="2px solid
             transparent" padding="0 0.5rem"
             margin="0px 8px" color="#FFFFFF"
             data-testid="cookieAccept"
             type="submit"><span
```

```
            class="GTXTypographystyle__Type-sc-1kk8ybz-0
            gnLmnZ">Accept</span></button></div></div>
         elementDescription: A cookie banner with a
            button to accept the cookies.
         explanation: In this cookie banner, the user
            only has the choice to accept the
            cookies. Giving the user no choice is
            manipulative.

   Output:
      Comments:
         Changes not possible. Explanation: In order
            to remove the manipulation of the cookie
            banner, I would need to provide an option
            for the user to reject cookies. But with
            the given code, I cannot access and alter
            how the website stores cookies. So it is
            not possible to provide an option to
            reject the cookies.


Example 3:
   Input:
      ImportantElements:
         elementCode:
         <span data-testid="discount-pill"
            class="css-1nt6bqz e1cq2ihg2" >-30%</span
            >
         elementDescription: A red discount badge
         explanation: This dicount badge is very
            visually prominent and attracts the
            user's attention. Additionally,
            displaying percentage discounts can
            exaggerate savings and steer decisions

         elementCode:
         <span dir="auto" role="group"
            aria-label="220" class="css-92woig
            e139ay0z0" ><span type="none"
            class="money-currency css-49gx66
            e139ay0z1" aria-hidden="true" ></span
            ><span type="none" class="money-integer
            css-2h7jcq e139ay0z2" aria-hidden="true"
            >220</span ></span >
         elementDescription: A price that was striked
            through
         explanation: Displaying discounts can
            exaggerate savings and steer decisions.
```

```
Output:
   Changes:
      Original:
      <span data-testid="discount-pill"
          class="css-1nt6bqz e1cq2ihg2" >-30%</span
          >
      Replacement:
      explanation: Removing the discount percentage
          completely prevents manipulation.

      Original:
      <span dir="auto" role="group"
          aria-label="220" class="css-92woig
          e139ay0z0" ><span type="none"
          class="money-currency css-49gx66
          e139ay0z1" aria-hidden="true" ></span
          ><span type="none" class="money-integer
          css-2h7jcq e139ay0z2" aria-hidden="true"
          >220</span ></span >
      Replacement:
      explanation: Removing the discount completely
          prevents manipulation.
```

A.8 Reasoning Prompt

```
    You are given the HTML and CSS for a web page or
        component. Your task is to identify and
        eliminate every element, style rule, label,
        layout choice, or interactive behavior that
        is intended to steer, pressure, or mislead
        usersespecially so-called dark patterns or
        deceptive design. If the element is
        non-essential remove it. Otherwise replace it
        with a non-manipulative version of it. All
        remaining information must be presented
        transparently and unambiguously.
Take your time and think as carefully and
    methodically about the task as you need to.

Rules:
- Do not obscure or downplay manipulative facts:
    either remove them or neutralize their impact.
- Only keep elements that are essential for
    functionality or to allow users an informed
    opinion.
```

```
- After edits, all remaining information must be
    fully transparent and unambiguous.
- If changes cannot or do not have to be made specify
    this in "comments"
- If manipulative elements were identified and all
    required changes were made, leave comments empty.
- For each change, return the original HTML parts in
    full and exact as they were before
```

A.9 Meta Prompt

```
Given a task description or existing prompt, produce
    a detailed system prompt to guide a language
    model in completing the task effectively.

# Guidelines

- Understand the Task: Grasp the main objective,
    goals, requirements, constraints, and expected
    output.
- Minimal Changes: If an existing prompt is provided,
    improve it only if it's simple. For complex
    prompts, enhance clarity and add missing elements
    without altering the original structure.
- Reasoning Before Conclusions**: Encourage reasoning
    steps before any conclusions are reached.
    ATTENTION! If the user provides examples where
    the reasoning happens afterward, REVERSE the
    order! NEVER START EXAMPLES WITH CONCLUSIONS!
    - Reasoning Order: Call out reasoning portions of
        the prompt and conclusion parts (specific
        fields by name). For each, determine the
        ORDER in which this is done, and whether it
        needs to be reversed.
    - Conclusion, classifications, or results should
        ALWAYS appear last.
- Examples: Include high-quality examples if helpful,
    using placeholders [in brackets] for complex
    elements.
   - What kinds of examples may need to be included,
       how many, and whether they are complex enough
       to benefit from placeholders.
- Clarity and Conciseness: Use clear, specific
    language. Avoid unnecessary instructions or bland
    statements.
```

```
- Formatting: Use markdown features for readability.
    DO NOT USE ‘‘‘ CODE BLOCKS UNLESS SPECIFICALLY
    REQUESTED.
- Preserve User Content: If the input task or prompt
    includes extensive guidelines or examples,
    preserve them entirely, or as closely as
    possible. If they are vague, consider breaking
    down into sub-steps. Keep any details,
    guidelines, examples, variables, or placeholders
    provided by the user.
- Constants: DO include constants in the prompt, as
    they are not susceptible to prompt injection.
    Such as guides, rubrics, and examples.
- Output Format: Explicitly the most appropriate
    output format, in detail. This should include
    length and syntax (e.g. short sentence,
    paragraph, JSON, etc.)
    - For tasks outputting well-defined or structured
        data (classification, JSON, etc.) bias toward
        outputting a JSON.
    - JSON should never be wrapped in code blocks
        (‘‘‘) unless explicitly requested.

The final prompt you output should adhere to the
    following structure below. Do not include any
    additional commentary, only output the completed
    system prompt. SPECIFICALLY, do not include any
    additional messages at the start or end of the
    prompt. (e.g. no "---")

The goal is to remove deceptive or dark patterns from
    websites. For this, the assistant will be given
    HTML Code of Web-Elements or whole Websites that
    may or may not include parts that nudge users
    into decicions or actions that might not be in
    their best interests. The assitant should
    identify these elements and change the HTML code
    so that they are not manipulative anymore. ONLY
    those manipulations should get changed, nothing
    else. If nothing is manipulative, the assistant
    should return "No changes". I want the output to
    be using the structured Output response_format.

# Output Format

The response should only include the modified HTML
    code
```

A.10 Improvement Meta Prompt

```
Your task is to improve a given system prompt to
    guide a language model in completing the
    following task effectively.

The Model for which you should produce the prompt
    will be given HTML Code of a website. It's task
    is to remove all elements that influence the user
    in the actions or decisions they make. Especially
    in regards to "Dark Patterns" or "Deceptive
    Design".

You will be given the Code the Model should use for
    it's task, the current prompt and a list of any
    problems it still has.

Do not add an own specific instruction for each
    problem. Instead add or adapt previous general
    instructions that cover the problem without only
    applying to this example. You can add or change
    rules or steps given in the original prompt, but
    do not change it fundamentally.

Please return an improved version of the prompt.
```

Appendix B

Response Formats

B.1 Standard Response

This is the format used for the *Basic, Advanced, Few-Shot, Removal* and *Reasoning Prompt*.

```
class Response(BaseModel):
   changes: list[Change] = Field(description="All the
       changes that are to be made")
   comments: str = Field(description="If changes were
       unnecessary or impossible")


class Change(BaseModel):
   original: str = Field(description="The exact
       original HTML code that should be removed")
   replacement: str = Field(description="The new HTML
       string that replaces the manipulation")
```

B.2 Explanation Response

This format was used for the *Zero-* and *Few-Shot Chain-of-Thought Prompt* as well as the *Combination Removal Prompt*.

```
class Response_Explanation(BaseModel):
```

```
    changes: list[Change_Explanation] =
        Field(description="All the changes that are
        to be made")
    comments: str = Field(description="If changes
        were unnecessary or impossible")

class Change_Explanation(BaseModel):
   original: str = Field(description="The exact
       original HTML code that should be removed")
   replacement: str = Field(description="The new HTML
       string that replaces the manipulation")
   explanation: str = Field(description="The
       reasoning as to why something is manipulative
       and how it was neutralized")
```

B.3 Analysis Response

This format was used by the *Analysis Prompt* for the *prompt chaining* technique.

```
class Response_Analysis(BaseModel):
   importantElements: list[Elements]
   comments: str = Field(description="If changes were
       unnecessary or impossible")

class Elements(BaseModel):
   elementCode: str = Field(description="The HTML
       Code of the element")
   elementDescription: str =
       Field(description="Description of the element")
```

B.4 Combination Analysis Response

This format was only used by the analysis part of the *Combination Prompt*

```
class Response_Analysis_Explanation(BaseModel):
   importantElements: list[Elements_Explanation]
```

```
   comments: str = Field(description="If changes were
       unnecessary or impossible")

class Elements_Explanation(BaseModel):
   elementCode: str = Field(description="The HTML
       Code of the element")
   elementDescription: str =
       Field(description="Description of the element")
   explanation: str = Field(description="The
       reasoning as to why something is manipulative")
```

Appendix C

Deceptive Pattern Ontology

The following contains the definitions by Gray et al. [2024] of the deceptive patterns, that are relevant in this thesis. We adapted them verbatim.

- SNEAKING is a strategy which hides, disguises, or delays the disclosure of important information that, if made available to users, would cause a user to unintentionally take an action they would likely object to.
  - BAIT AND SWITCH subverts the user's expectation that their choice will result in a desired action, instead leading to an unexpected, undesirable outcome.
    - DISGUISED ADS Bait and Switch and use Sneaking to style interface elements so they are not clearly marked as an advertisement or other biased source. As a result, users are induced into clicking on the interface element because they assume that it is a relevant and salient interaction, leading to unwitting interaction with advertising content.
  - HIDING INFORMATION subverts the user's expectation that all relevant information to make

an informed choice will be available to them, instead hiding information or delaying the disclosure of information until later in the user journey that may have led to them making another choice.

  * SNEAK INTO BASKET Hides Information and uses Sneaking to add unwanted items to a user's shopping cart without their consent. As a result, a user assumes that only the items they explicitly added to their cart will be purchased, leading to the unintentional purchase of additional items.
  * DRIP PRICING, HIDDEN COSTS, OR PARTITIONED PRICING Hides Information and uses Sneaking to reveal new charges or costs, present only partial price components, or otherwise delay revealing the full price of a product or service through late or incomplete disclosure. As a result, the user is misled about the total or complete price of the product or service, leading them to make a purchase decision after they have expended effort on false pretenses.
  * REFERENCE PRICING Hides Information and uses Sneaking to include a misleading or inaccurate price for a product or service that makes a discounted price appear more attractive. As a result, the user is misled into believing that the price they pay is discounted, leading them to make a decision to purchase a product or service on false pretenses.

- OBSTRUCTION is a strategy which impedes a user's task flow, making an interaction more difficult than it inherently needs to be, dissuading a user from taking an action.

  – CREATING BARRIERS subverts the user's expectation that relevant user tasks will be supported by the interface, instead preventing, abstracting, or otherwise complicating a user task to disincentivize user action.

    * PRICE COMPARISON PREVENTION Creates Barriers and uses Obstruction by excluding relevant information, limiting the ability of a user to copy/paste, or otherwise inhibiting a user from comparing prices across two or more vendors. As a result, the user cannot make an informed decision about where to buy a product or service.

- INTERFACE INTERFERENCE is a strategy which privileges specific actions over others through manipulation of the user interface, thereby confusing the user or limiting discoverability of relevant action possibilities.
  - MANIPULATING CHOICE ARCHITECTURE subverts the user's expectation that the options presented will support their desired goal, instead including an order or structure of options that makes another outcome more likely.
    * FALSE HIERARCHY Manipulates the Choice Architecture, using Interface Interference to give one or more options visual or interactive prominence over others, particularly where items should be in parallel rather than hierarchical. As a result, the user may misunderstand or be unable to accurately compare their options, making a selection based on a false or incomplete choice architecture.
    * VISUAL PROMINENCE Manipulates the Choice Architecture, using Interface Interference to place an element relevant to user goals in visual competition with a more distracting and prominent element. As a result, the user may forget about or be distracted from their original goal, even if that goal was their primary intent.
    * PRESSURED SELLING Manipulates the Choice Architecture, using Interface Interference to preselect or use visual prominence to focus user attention on more expensive product options. As a result, the user may be unaware that a lower price is

available or even desirable for their needs, steering the user into making a more expensive product selection than they otherwise would have.

- BAD DEFAULTS subverts the user's expectation that default settings will be in their best interest, instead requiring users to take active steps to change settings that may cause harm or unintentional disclosure of information.
- EMOTIONAL OR SENSORY MANIPULATION subverts the user's expectation that the design of the site will allow them to achieve their goal without manipulation, instead altering the language, style, color, or other design elements to evoke an emotion or manipulate the senses in order to persuade the user into a particular action.
  - POSITIVE OR NEGATIVE FRAMING uses Emotional or Sensory Manipulation and Interface Interference to visually obscure, distract, or persuade a user from important information they need to achieve their goal. As a result, the user may assume that the system is providing equal access to relevant information, leading the user to be distracted by positive or negative aesthetic cues that distract them from important information or action possibilities or otherwise convince them to pursue a different goal.
- TRICK QUESTIONS subvert the user's expectation that prompts will be written in a straightforward and intelligible manner, instead using confusing wording, double negatives, or otherwise leading language or interface cues to manipulate a user's choice.

- FORCED ACTION is a strategy which requires users to knowingly or unknowingly perform an additional and/or tangential action or information to access (or continue to access) specific functionality, preventing them from continuing their interaction with a system without performing that action.

  - NAGGING subverts the user's expectation that they have rational control over the interaction they make with a system, instead distracting the user from a desired task the user is focusing on to induce an action or make a decision the user does not want to make by repeatedly interrupting the user during normal interaction.
  - FORCED COMMUNICATION OR DISCLOSURE subverts the user's expectation that a system will only request information needed to complete their desired goals, instead tricking them into sharing more information about themselves or using their information for purposes that they do not desire.
    * *Privacy Zuckering* uses Forced Communication or Disclosure as a type of Forced Action to trick users into sharing more information about themselves than they intend to or would agree to if fully informed. As a result, the user assumes that information they are requested to provide is vital for the use of the service, even while this information is used or sold for other purposes.

- SOCIAL ENGINEERING is a strategy which presents options or information that causes a user to be more likely to perform a specific action based on their individual and/or social cognitive biases, thereby leveraging a user's desire to follow expected or imposed social norms.
  - Scarcity or Popularity Claims subverts the user's expectation that information provided about a product's availability or desirability is accurate, instead pressuring the user to purchase a product without additional reflection or verification.
    * HIGH DEMAND uses Scarcity and Popularity Claims as a type of Social Engineering to indicate that a product is in high demand or likely to sell out soon, even though that claim is misleading or false. As a result, the user may assume that demand is high when it is not, leading to their uninformed purchase of a product or service.

- SOCIAL PROOF subverts the user's expectation that the indicated behavior of others in a specific situation is correct or desirable, instead accelerating user decision-making and encouraging the user to trust flawed implications through provided information.
  - LOW STOCK uses Social Proof as a type of Social Engineering to indicate that a product is limited in quantity, even though that claim is misleading or false. As a result, the user may assume that a product is desirable due to demand, leading to undue or uninformed pressure to buy the product immediately.
  - PARASOCIAL PRESSURE uses Social Proof as a type of Social Engineering to indicate that a product or service has been endorsed by a celebrity, infuencer, or other entity that the user trusts, even though the source of that endorsement is biased, misleading, incomplete, or false. As a result, the user may assume that the endorsement is accurate and unbiased, leading to their uninformed purchase of a product or service.
- URGENCY subverts the user's expectation that information provided about discounts or a limited-time deal for a product is accurate, instead accelerating the user's decision-making process by demanding immediate or timely action.
  - Activity Messages use Urgency as a type of Social Engineering to describe other user activity on the site or service, even though the data presented about other users' purchases, views, visits, or contributions are misleading or false. As a result, the user may falsely feel a sense of urgency, assuming that other users are purchasing or otherwise interested product or service, leading to their uninformed purchase of a product or service.
  - COUNTDOWN TIMERS use Urgency as a type of Social Engineering to indicate that

a deal or discount will expire by displaying a countdown clock or timer, even though the clock or timer is completely fake, disappears, or resets automatically. As a result, the user may feel undue urgency and purchasing pressure, leading to their uninformed purchase of a product or service.

  * LIMITED TIME MESSAGES use Urgency as a type of Social Engineering to indicate that a deal or discount will expire soon or be available only for a limited time, but without specifying a specific deadline. As a result, the user may feel undue urgency and purchasing pressure, leading to their uninformed purchase of a product or service.

- PERSONALIZATION subverts the user's expectation that products or service features are offered to all users in similar ways, instead using personal data to shape elements of the user experience that manipulate the user's goals while hiding other alternatives.

  * CONFIRMSHAMING uses Personalization as a type of Social Engineering to frame a choice to opt-in or opt-out of a decision through emotional language or imagery that relies upon shame or guilt. As a result, the user may be convinced to change their goal due to the emotionally manipulative tactics, resulting in being steered away from making a choice that matched their initial goal.

Appendix D

Results

Prompt	DLC				M			
	Average	# 0	# 4	Mode	Average	# 0	# 4	Mode
Basic 0.1	1.92	3	0	1	1	5	1	1
Basic 0.2	2.04	2	4	1	1.4	4	1	1
Advanced 0.1	2.08	0	3	1	1.84	2	2	1
Advanced 0.2	2.12	0	2	1	1.84	3	2	1
Chain-of-Thought	2	0	0	2	2.08	1	3	1
Few-Shot	2.32	0	6	1	2.08	2	2	3
Few-Shot Chain-of-Thought	2.36	0	7	1	2.28	3	5	3
Removal Expert	2.28	0	5	1	1.96	2	2	1
Deceptive Pattern Expert	2.24	0	4	1	1.6	5	2	1
Marketing & Sales Expert	2.12	0	2	1	1.8	3	3	1
Prompt Chaining	1.92	0	2	1	2.2	0	3	1
Combination	2.44	0	6	1	2.24	1	3	2
Reasoning Model: Basic Low	2.12	4	7	4	1.72	1	2	1
Reasoning Model: Basic High	2.44	2	7	4	2.16	3	7	1
Reasoning Low	2.92	2	11	4	1.96	2	6	1
Reasoning Low	2.52	1	7	4	2.32	3	6	3
Fine-Tuning	2.88	1	12	4	2.92	2	12	4

Table D.1: Relevant values for the *Design, Layout, and Context* (DLC) and *Manipulation Score* (M) for all 17 conditions. “Average" contains the mean score over all designs. “#0” and “#4” contain the amount of times a score of 0 or 4 was assigned. “Mode” is the overall most assigned score for the condition. The fine-tuned and reasoning model achieved overall better results.

Bibliography

[1] Toufique Ahmed and Premkumar Devanbu. Few-shot training LLMs for project-specific code-summarization. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, ASE '22, New York, NY, USA, 2023. Association for Computing Machinery. doi.org/10.1145/3551349.3559555.

[2] K. Srinivas Babu, Bangari Sai Nithin, E. Raghuram, and A. Bharath. Dark Surfer-A Dark Pattern Analyzer & Classifier. *ITM Web of Conferences*, 74:01018, 2025. doi.org/10.1051/itmconf/20257401018.

[3] Kerstin Bongard-Blanchy, Arianna Rossi, Salvador Rivas, Sophie Doublet, Vincent Koenig, and Gabriele Lenzini. "I Am Definitely Manipulated, Even When I Am Aware of It. It's Ridiculous!" - Dark Patterns from the End-User Perspective. In *Proceedings of the 2021 ACM Designing Interactive Systems Conference*, DIS '21, pages 763–776, New York, NY, USA, June 2021. Association for Computing Machinery. doi.org/10.1145/3461778.3462086.

[4] Christoph Bösch, Benjamin Erb, Frank Kargl, Henning Kopp, and Stefan Pfattheicher. Tales from the Dark Side: Privacy Dark Strategies and Privacy Dark Patterns. *Proceedings on Privacy Enhancing Technologies*, page 237–254, 2016. doi.org/10.1515/popets-2016-0038.

[5] Dainis Boumber, Bryan E. Tuck, Rakesh M. Verma, and Fatima Zahra Qachfar. LLMs for Explainable Few-shot Deception Detection. In *Proceedings of the 10th ACM International Workshop on Security and Privacy Analytics*, IWSPA '24, pages 37–47, New York, NY, USA, June 2024. Association for Computing Machinery. doi.org/10.1145/3643651.3659898.

[6] Harry Brignull. *Deceptive patterns: Exposing the tricks tech companies use to control you*. Testimonium Ltd, 2023. ISBN 978-1739454401.

[7] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger,

Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc.

[8] Jieshan Chen, Jiamou Sun, Sidong Feng, Zhenchang Xing, Qinghua Lu, Xiwei Xu, and Chunyang Chen. Unveiling the Tricks: Automated Detection of Dark Patterns in Mobile Applications. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pages 1–20, San Francisco CA USA, October 2023. ACM. doi.org/10.1145/3586183.3606783.

[9] Jieshan Chen, Zhen Wang, Jiamou Sun, Wenbo Zou, Zhenchang Xing, Qinghua Lu, Qing Huang, and Xiwei Xu. From Exploration to Revelation: Detecting Dark Patterns in Mobile Apps. November 2024. doi.org/10.48550/arXiv.2411.18084.

[10] Ziwei Chen, Jiawen Shen, Luna, and Kristen Vaccaro. Hidden Darkness in LLM-Generated Designs: Exploring Dark Patterns in Ecommerce Web Components Generated by LLMs. February 2025. doi.org/10.48550/arXiv.2502.13499.

[11] Gregory Conti and Edward Sobiesk. Malicious Interface Design: Exploiting the User. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 271–280, New York, NY, USA, 2010. Association for Computing Machinery. doi.org/10.1145/1772690.1772719.

[12] Linda Di Geronimo, Larissa Braz, Enrico Fregnan, Fabio Palomba, and Alberto Bacchelli. UI Dark Patterns and Where to Find Them: A Study on Mobile Applications and User Perception. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, pages 1–14, New York, NY, USA, April 2020. Association for Computing Machinery. doi.org/10.1145/3313831.3376600.

[13] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Tianyu Liu, Baobao Chang, Xu Sun, Lei Li, and Zhifang Sui. A Survey on In-context Learning. October 2024. doi.org/10.48550/arXiv.2301.00234.

[14] Paul Graßl, Hanna Schraffenberger, Frederik Zuiderveen Borgesius, and Moniek Buijzen. Dark and Bright Patterns in Cookie Consent Requests. *Journal of Digital Social Research*, 3(1):1–38, February 2021. doi.org/10.33621/jdsr.v3i1.54.

[15] Colin M. Gray, Yubo Kou, Bryan Battles, Joseph Hoggatt, and Austin L. Toombs. The Dark (Patterns) Side of UX Design. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, pages 1–14, New York, NY, USA, April 2018. Association for Computing Machinery. doi.org/10.1145/3173574.3174108.

[16] Colin M. Gray, Jingle Chen, Shruthi Sai Chivukula, and Liyang Qu. End User Accounts of Dark Patterns as Felt Manipulation. *Proc. ACM Hum.-Comput. Interact.*, 5(CSCW2):372:1–372:25, October 2021. doi.org/10.1145/3479516.

[17] Colin M. Gray, Cristiana Santos, Nataliia Bielova, Michael Toth, and Damian Clifford. Dark Patterns and the Legal Requirements of Consent Banners: An Interaction Criticism Perspective. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI '21, pages 1–18, New York, NY, USA, May 2021. Association for Computing Machinery. doi.org/10.1145/3411764.3445779.

[18] Colin M. Gray, Cristiana Teixeira Santos, Nataliia Bielova, and Thomas Mildner. An Ontology of Dark Patterns Knowledge: Foundations, Definitions, and a Pathway for Shared Knowledge-Building. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pages 1–22, Honolulu HI USA, May 2024. ACM. doi.org/10.1145/3613904.3642436.

[19] Colin M. Gray, Thomas Mildner, and Ritika Gairola. Getting Trapped in Amazon's "Iliad Flow": A Foundation for the Temporal Analysis of Dark Patterns. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, CHI '25, pages 1–10, New York, NY, USA, April 2025. Association for Computing Machinery. doi.org/10.1145/3706598.3713828.

[20] Yu Gu, Kai Zhang, Yuting Ning, Boyuan Zheng, Boyu Gou, Tianci Xue, Cheng Chang, Sanjari Srivastava, Yanan Xie, Peng Qi, Huan Sun, and Yu Su. Is Your LLM Secretly a World Model of the Internet? Model-Based Planning for Web Agents. April 2025. doi.org/10.48550/arXiv.2411.06559.

[21] Johanna Gunawan, Cristiana Santos, and Irene Kamara. Redress for Dark Patterns Privacy Harms? A Case Study on Consent Interactions. In *Proceedings of the 2022 Symposium on Computer Science and Law*, pages 181–194, Washington DC USA, November 2022. ACM. doi.org/10.1145/3511265.3550448.

[22] Izzeddin Gur, Ofir Nachum, Yingjie Miao, Mustafa Safdari, Austin Huang, Aakanksha Chowdhery, Sharan Narang, Noah Fiedel, and Aleksandra Faust. Understanding HTML with Large Language Models. May 2023. doi.org/10.48550/arXiv.2210.03945.

[23] S. M. Hasan Mansur, Sabiha Salma, Damilola Awofisayo, and Kevin Moran. AidUI: Toward Automated Recognition of Dark Patterns in User Interfaces.

In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, pages 1958–1970, 2023. doi.org/10.1109/ICSE48619.2023.00166.

[24] Johanna Herman. Dark Patterns: EU's Regulatory Efforts. *SECURITY AND PRIVACY*, 7(6):e441, 2024. doi.org/10.1002/spy2.441.

[25] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, with Language Models*. 3rd edition, 2025. Online manuscript released August 24, 2025.

[26] Adam Tauman Kalai, Ofir Nachum, Santosh S. Vempala, and Edwin Zhang. Why Language Models Hallucinate. September 2025. doi.org/10.48550/arXiv.2509.04664.

[27] Pruthvi S. Kodmurgi, Srihari Adiga, Srikrshna Parthasarthy, R Thanushree, Varun Vishwanatha Avabratha, Preet Kanwal, and Prasad B Honnavalli. ScrapeAI: A Multi-Modal Approach to Detect Dark Patterns. In *2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, pages 1–6, June 2024. doi.org/10.1109/ICCCNT61001.2024.10723319.

[28] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc.

[29] Konrad Kollnig, Siddhartha Datta, and Max Van Kleek. I Want My App That Way: Reclaiming Sovereignty Over Personal Devices. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–8, Yokohama Japan, May 2021. ACM. doi.org/10.1145/3411763.3451632.

[30] Aobo Kong, Shiwan Zhao, Hao Chen, Qicheng Li, Yong Qin, Ruiqi Sun, Xin Zhou, Enzhi Wang, and Xiaohang Dong. Better Zero-Shot Reasoning with Role-Play Prompting. March 2024. doi.org/10.48550/arXiv.2308.07702.

[31] Esben Kran, Hieu Minh Nguyen, Akash Kundu, Sami Jawhar, Jinsuk Park, and Mateusz Maria Jurewicz. DarkBench: Benchmarking Dark Patterns in Large Language Models. In *The Thirteenth International Conference on Learning Representations*, October 2024. doi.org/10.48550/arXiv.2503.10728.

[32] Veronika Krauß, Mark McGill, Thomas Kosch, Yolanda Maira Thiel, Dominik Schön, and Jan Gugenheimer. "Create a Fear of Missing Out" - ChatGPT Implements Unsolicited Deceptive Designs in Generated Websites Without Warning. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, pages 1–20, Yokohama Japan, April 2025. ACM. doi.org/10.1145/3706598.3713083.

[33] Chiara Krisam, Heike Dietmann, Melanie Volkamer, and Oksana Kulyk. Dark Patterns in the Wild: Review of Cookie Disclaimer Designs on Top 500 German Websites. In *Proceedings of the 2021 European Symposium on Usable Security*, EuroUSEC '21, pages 1–8, New York, NY, USA, December 2021. Association for Computing Machinery. doi.org/10.1145/3481357.3481516.

[34] Mark Leiser and Cristiana Santos. Dark Patterns, Enforcement, and the Emerging Digital Design Acquis: Manipulation beneath the Interface. *European Journal of Law and Technology*, 15(1), May 2024.

[35] Frank Lewis and Julita Vassileva. Seeing in the Dark: Revealing the Relationships, Goals, and Harms of Dark Patterns. In Colin M. Gray, Johanna Gunawan, René Schäfer, Nataliia Bielova, Lorena Sánchez Chamorro, Katie Seaborn, Thomas Mildner, and Hauke Sandhaus, editors, *Proceedings of the Workshop Mobilizing Research and Regulatory Action on Dark Patterns and Deceptive Design Practices (DDPCHI 2024)*, volume 3720 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2024.

[36] Amanda Li, Jason Wu, and Jeffrey P Bigham. Using LLMs to Customize the UI of Webpages. In *Adjunct Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, UIST '23 Adjunct, pages 1–3, New York, NY, USA, October 2023. Association for Computing Machinery. doi.org/10.1145/3586182.3616671.

[37] Chenxi Li, Yuanhe Tian, Zhaxi Zerong, Yan Song, and Fei Xia. Challenging Large Language Models with New Tasks: A Study on Their Adaptability and Robustness. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 8140–8162, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi.org/10.18653/v1/2024.findings-acl.485.

[38] Vadim Lomshakov, Sergey Kovalchuk, Maxim Omelchenko, Sergey Nikolenko, and Artem Aliev. Fine-Tuning Large Language Models for Answering Programming Questions with Code Snippets. In *Computational Science – ICCS 2023: 23rd International Conference, Prague, Czech Republic, July 3–5, 2023, Proceedings, Part II*, pages 171–179, Berlin, Heidelberg, July 2023. Springer-Verlag. doi.org/10.1007/978-3-031-36021-3_15.

[39] Jamie Luguri and Lior Jacob Strahilevitz. Shining a Light on Dark Patterns. *Journal of Legal Analysis*, 13(1):43–109, January 2021. doi.org/10.1093/jla/laaa006.

[40] Francisco Lupiáñez-Villanueva, Alba Boluda, Francesco Bogliacino, Giovanni Liva, Lucie Lechardoy, and Teresa Rodríguez de las Heras Ballell. *Behavioural Study on Unfair Commercial Practices in the Digital Environment: Dark Patterns*

and Manipulative Personalisation. Publications Office of the European Union, Luxembourg, Luxembourg, 2022. doi.org/10.2838/859030.

[41] Arunesh Mathur, Gunes Acar, Michael J. Friedman, Eli Lucherini, Jonathan Mayer, Marshini Chetty, and Arvind Narayanan. Dark Patterns at Scale: Findings from a Crawl of 11K Shopping Websites. *Proc. ACM Hum.-Comput. Interact.*, 3(CSCW):81:1–81:32, November 2019. doi.org/10.1145/3359183.

[42] Arunesh Mathur, Mihir Kshirsagar, and Jonathan Mayer. What Makes a Dark Pattern... Dark? Design Attributes, Normative Considerations, and Measurement Methods. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI '21, pages 1–18, New York, NY, USA, May 2021. Association for Computing Machinery. doi.org/10.1145/3411764.3445610.

[43] Stuart Mills and Richard Whittle. Detecting Dark Patterns Using Generative AI: Some Preliminary Results. *SSRN*, (4614907), October 2023. doi.org/10.2139/ssrn.4614907.

[44] Arvind Narayanan, Arunesh Mathur, Marshini Chetty, and Mihir Kshirsagar. Dark Patterns: Past, Present, and Future: The Evolution of Tricky User Interfaces. *Queue*, 18(2):Pages 10:67–Pages 10:92, May 2020. doi.org/10.1145/3400899.3400901.

[45] Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. A Comprehensive Overview of Large Language Models. *ACM Transactions on Intelligent Systems and Technology*, 16(5):1–72, August 2025. doi.org/10.1145/3744746.

[46] Asmit Nayak, Shirley Zhang, Yash Wani, Rishabh Khandelwal, and Kassem Fawaz. Automatically Detecting Online Deceptive Patterns in Real-time. November 2024. doi.org/10.48550/arXiv.2411.07441.

[47] Harsha Nori, Naoto Usuyama, Nicholas King, Scott Mayer McKinney, Xavier Fernandes, Sheng Zhang, and Eric Horvitz. From Medprompt to O1: Exploration of Run-Time Strategies for Medical Challenge Problems and Beyond. November 2024. doi.org/10.48550/arXiv.2411.03590.

[48] Midas Nouwens, Ilaria Liccardi, Michael Veale, David Karger, and Lalana Kagal. Dark Patterns after the GDPR: Scraping Consent Pop-ups and Demonstrating Their Influence. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, pages 1–13, New York, NY, USA, April 2020. Association for Computing Machinery. doi.org/10.1145/3313831.3376321.

[49] OpenAI, Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, A. J. Ostrow, Akila Welihinda, Alan Hayes, and

others. GPT-4o System Card. October 2024. doi.org/10.48550/arXiv.2410.21276.

[50] OpenAI, Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, and others. OpenAI O1 System Card. December 2024. doi.org/10.48550/arXiv.2412.16720.

[51] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc.

[52] Lorenzo Porcelli, Michele Mastroianni, Massimo Ficco, and Francesco Palmieri. A User-Centered Privacy Policy Management System for Automatic Consent on Cookie Banners. *Computers*, 13(2), 2024. doi.org/10.3390/computers13020043.

[53] Marie Potel-Saville and Mathilde Da Rocha. From Dark Patterns to Fair Patterns? Usable Taxonomy to Contribute Solving the Issue with Countermeasures. In Kai Rannenberg, Prokopios Drogkaris, and Cédric Lauradoux, editors, *Privacy Technologies and Policy*, volume 13888, pages 145–165, Cham, 2024. Springer Nature Switzerland. doi.org/10.1007/978-3-031-61089-9_7.

[54] Yasin Sazid, Mridha Md. Nafis Fuad, and Kazi Sakib. Automated Detection of Dark Patterns Using In-Context Learning Capabilities of GPT-3. In *2023 30th Asia-Pacific Software Engineering Conference (APSEC)*, pages 569–573, December 2023. doi.org/10.1109/APSEC60848.2023.00072.

[55] René Schäfer, Paul Miles Preuschoff, Rene Niewianda, Sophie Hahn, Kevin Fiedler, and Jan Borchers. Don't Detect, Just Correct: Can LLMs Defuse Deceptive Patterns Directly? In *Proceedings of the Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, pages 1–11, Yokohama Japan, April 2025. ACM. doi.org/10.1145/3706599.3719683.

[56] René Schäfer, Paul Miles Preuschoff, and Jan Borchers. Investigating Visual Countermeasures Against Dark Patterns in User Interfaces. In *Proceedings of Mensch Und Computer 2023*, MuC '23, page 161–172, New York, NY, USA, 2023. Association for Computing Machinery. doi.org/10.1145/3603555.3603563.

[57] Kathrin Seßler, Yao Rong, Emek Gözlüklü, and Enkelejda Kasneci. Benchmarking Large Language Models for Math Reasoning Tasks. December 2024. doi.org/10.48550/arXiv.2408.10839.

[58] Subin Shin, Jeesun Oh, and Sangwon Lee. Can LLMs See What I See? A Study on Five Prompt Engineering Techniques for Evaluating UX on a Shopping Site. In *Proceedings of the Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, pages 1–7, Yokohama Japan, April 2025. ACM. doi.org/10.1145/3706599.3720079.

[59] Anikait Singh, Sheryl Hsu, Kyle Hsu, Eric Mitchell, Stefano Ermon, Tatsunori Hashimoto, Archit Sharma, and Chelsea Finn. FSPO: Few-Shot Preference Optimization of Synthetic Preference Data in LLMs Elicits Effective Personalization to Real Users. February 2025. doi.org/10.48550/arXiv.2502.19312.

[60] Than Htut Soe, Cristiana Teixeira Santos, and Marija Slavkovik. Automated Detection of Dark Patterns in Cookie Banners: How to Do It Poorly and Why It Is Hard to Do It Any Other Way. April 2022. doi.org/10.48550/arXiv.2204.11836.

[61] Carla Sousa and Ana Oliveira. The Dark Side of Fun: Understanding Dark Patterns and Literacy Needs in Early Childhood Mobile Gaming. *European Conference on Games Based Learning*, 17(1):599–610, September 2023. doi.org/10.34190/ecgbl.17.1.1656.

[62] Ioannis Stavrakakis, Andrea Curley, Dympna O'Sullivan, Damian Gordon, and Brendan Tierney. A Framework of Web-Based Dark Patterns That Can Be Detected Manually or Automatically. 2021.

[63] Mirac Suzgun and Adam Tauman Kalai. Meta-Prompting: Enhancing Language Models with Task-Agnostic Scaffolding. January 2024. doi.org/10.48550/arXiv.2401.12954.

[64] Jonathan J. Webster and Chunyu Kit. Tokenization as the initial phase in NLP. In *Proceedings of the 14th Conference on Computational Linguistics - Volume 4*, COLING '92, page 1106–1110, USA, 1992. Association for Computational Linguistics. doi.org/10.3115/992424.992434.

[65] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. January 2023. doi.org/10.48550/arXiv.2201.11903.

[66] Robert Wolfe and Alexis Hiniker. Expertise Fog on the GPT Store: Deceptive Design Patterns in User-Facing Generative AI. In *Mobilizing Research and Regulatory Action on Dark Patterns and Deceptive Design Practices* Workshop at CHI '24, Honolulu, 2024. URL https://ceur-ws.org/Vol-3720/paper15.pdf.

[67] Yuki Yada, Jiaying Feng, Tsuneo Matsumoto, Nao Fukushima, Fuyuko Kido, and Hayato Yamana. Dark Patterns in E-Commerce: A Dataset and Its Baseline Evaluations. November 2022. doi.org/10.48550/arXiv.2211.06543.

[68] José P. Zagal, Staffan Björk, and Chris Lewis. Dark Patterns in the Design of Games. In *Foundations of Digital Games 2013*, 2013.

[69] Ziyao Zhang, Chong Wang, Yanlin Wang, Ensheng Shi, Yuchi Ma, Wanjun Zhong, Jiachi Chen, Mingzhi Mao, and Zibin Zheng. LLM Hallucinations in Practical Code Generation: Phenomena, Mechanism, and Mitigation. *Proc. ACM Softw. Eng.*, 2(ISSTA):ISSTA022:481–ISSTA022:503, June 2025. doi.org/10.1145/3728894.

[70] Mingqian Zheng, Jiaxin Pei, Lajanugen Logeswaran, Moontae Lee, and David Jurgens. When "A Helpful Assistant" Is Not Really Helpful: Personas in System Prompts Do Not Improve Performances of Large Language Models. October 2024. doi.org/10.48550/arXiv.2311.10054.

[71] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and Ed Chi. Least-to-Most Prompting Enables Complex Reasoning in Large Language Models, April 2023.

Index

Typeset September 30, 2025