SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Information Systems

From UI Images to Accessible Code: Leveraging LLMs for Automated Frontend Generation

Marco Lutz

SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Information Systems

From UI Images to Accessible Code: Leveraging LLMs for Automated Frontend Generation

From UI Images to Accessible Code: Leveraging LLMs for Automated Frontend Generation

Author:	Marco Lutz
Supervisor:	Sidong Feng
Advisor:	Chunyang Chen
Submission Date:	11.08.2025

I confirm that this bachelor's thesis in information systems is my own work and I have documented all sources and material used.

Munich, 11.08.2025 Marco Lutz

Acknowledgments

Abstract

Contents

1 Introduction

1.1 Section

High quality web user interfaces are the backbone of our modern society. They allow us to present products or services in an interactive way and reach billions of users every day. However, the creation of Websites or user interfaces (UIs) follows a similar and repetitive pattern.
First UI designs are created with the help of special design tools. The UI designs present the foundation for software developers. In a second step, those designs are translated into functional UI code which tries to resemble the intended layout and structure, but also obey to other design aspects.
One essential, but yet frequently underestimated aspect in this process is *accessibility*: According to the official *WCAG* guidelines, code must be perceivable, operable, understandable and robust for people with various disabilities.
Complying with accessibility standards is not only an optional, moral aspect of web development, but it now has to follow regulatory boundaries. Ensuring accessibility is no longer optional. For instance the *European Accessibility Act* comes into effect on June 28, 2025 and obliges any e-commerce or digital service in the EU to comply with those standards.
Current Large-Language Models (LLMs) have shown signifant improvements in automatic code generation. Especially *Image-to-Code* tasks where UI designs are given as input and LLMs output the functional UI code, have been tested by various researchers in the past. Several benchmarks have shown the competitive performance of LLMs on those tasks. However, the capability of modern LLMs to generate accessible Code in an Image-to-Code environment has only started to gain researchers interest quite recently. Existing research in this field have compared human to the generated code and tried to better align with the accessibility standards. Nevertheless, this has never been tested in an Image-to-Code environment. Apart from that, accessibility does not only concern the visual appearance of a web user interace, but also its functionality. Thus, it requires the LLMs to have a deeper understanding than in classical Image-to-Code scenarios where LLMs only reproduce the input images pixel perfectly.

1.1.1 Our Contributions

In order to close this gap, we propose a large scale accessibility evaluation of LLM-based web code generation while also taking the visual similarity into account.

- Therefore, we use a dataset which contains of 53 real-world webpage examples which have been gathered from existing datasets and mutated in order to prevent data leakage. It covers a wide spectrum of layouts, content areas and accessibility features.
- This dataset is then used in a reproducible evaluation pipeline which measures both the visual similarity (pixel/DOM fidelity), as well as the accessibility compliance. Therefore, we propose different benchmarks in order to measure the performance of the LLMs.
- This environment will be tested in an in-depth comparison pipeline of 4 state-of-the-art LLMs across different prompting strategies.

1.1.2 Subsection

Citation test [Lam94]. See Table 1.1, Figure 1.1, Figure 1.2, Figure 1.3.

Table 1.1: An example for a simple table.

A	B	C	D
1	2	1	2
2	3	2	3

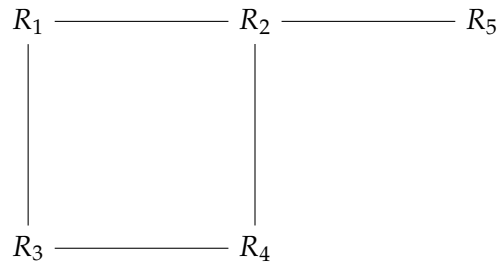

Figure 1.1: An example for a simple drawing.

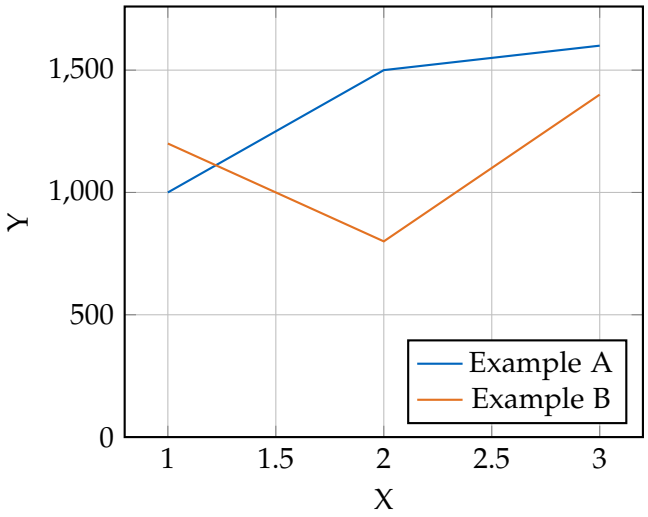

Figure 1.2: An example for a simple plot.

```
SELECT * FROM tbl WHERE tbl.str = "str"
```

Figure 1.3: An example for a source code listing.

2 Related Work

2.1 Web Accessibility

asf

2.2 Image-to-Code

sided

3 Dataset

3.1 Construction

The main goal is to gather a diverse and high-quality dataset which represents static real-world HTML/CSS webpages. This includes different layouts, components and contents. In the past, there have been different attempts to collect a dataset fullfilling exactly those requirements.
Two promising examples in this field are *Design2Code* and *Webcode2m*. Both have used existing, large datasets and applied different processing steps to filter bad examples and remove noise or redundancy from the code. Based on their dataset curation, both serve as a good base for this thesis.
Therefore, we decided to use both datasets and manually select 53 high-quality data entries. Those 53 data entries consist of 28 entries from Design2Code and 25 entries from Webcode2m. In order to compare them on a fair basis, we only collect webpages that have english as their primary language.

3.1.1 Content Distribution

By using data entries of various domains and different layouts, we make sure to get a fair representation of the distribution of webpages in the real world. Based on our manual selection, we present the domain distribution in a pie chart in Figure 1.

3.2 Dataset Mutation

Due to the fact that Design2Code and Webcode2m use different strategies to purify their data, it is necessary to align both datasets in order to get a fair comparison. This includes removing all external dependencies such as multimedia files (e.g. images , audio, videos, ...) from the datasets. Furhtermore, this means adding placeholders like src=p̈laceholder.jpgf̈or images or href=#f̈or <a> Tags. Lastly, we remove all of the non-visible content (advertisement-related, hidden) of the webpages, because it is not necessary in an Image-to-Code environment and could only add negatively to the accessibility score.
In a last step we try to minimize the risk of data leakage. Both datasets have been

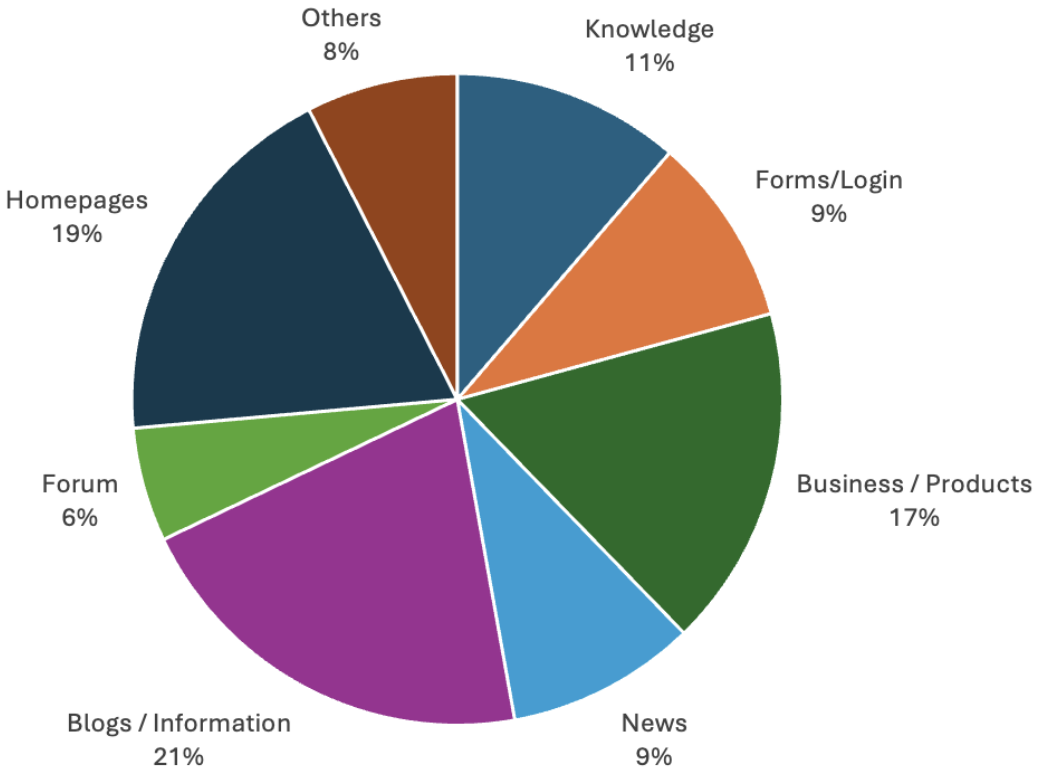

Figure 3.1: Distribution of Topics in Dataset

uploaded a few months before the official knowledge-cutoff of some of the LLMs, which we use, to Huggingface and theoretically, they were publicly available at that time. While Design2Code has uploaded its data 3 months before the knowledge-cutoff, in the case of Webcode2m, it was only 2 months. To tackle this issue, we mutate the dataset in the following way without adding or removing (max ±10 %) many accessibility violations:

- **Text:** The entire text has been rewritten by a LLM. While the meaning and the length (max ±20 %) remains roughly the same, the wording changes completely, in order to avoid memorization based on text snippets.
- **Text Font:** To change the visual appearance of the data entries, we define a set of 5 commonly used fonts in webpages. Based on a random sample, we change the text font for each data entry.
- **Colors:** The *HUE* color code of each element is slightly changed based on a random shift (±20 degrees). Apart from that, the saturation and lightness of the color is changed by a maximum of ±20 %.

Those changes combined allow us to alter the dataset in such a way to minimize the risk of data leakage. At this point, it is important to mention, that the mutations described above, caused some new, artificial accessibility violations. Nevertheless, the amount of new violations remains small ($\approx$ 1 new violation / file) and thus is seen as negligible.

4 Benchmarks

4.1 Visual and Structural

As the main instruction for LLMs for this study remains an Image-to-Code task, it is necessary to evaluate the generated HTML/CSS code based on visual and structural similarity. One approach which seems very promising has been presented in the paper Design2Code. This approach is more fine-grained than former ideas, since it compares the input and the output on a component-level rather than in its entirety. The authors describe a sophisticated matching algorithm that combines the HTML elements in the ground truth with those in the generated code. Based on this matching, it is then possible to run several metrics, such as text-similarity, position-similarity, color-difference, clip score and area sum score.

4.2 Accessibility

In order to measure and compare the accessibility improvements in terms of quantity and severity of the violations we use two different metrics.
The first metric is the *Inaccessibility Rate*(IR) which has been used in previous researchs in this field. This metric divides the amount of nodes with accessibility violations by the amount of nodes which are susceptible for violations. The interpretation of this metric is straight-forward, since it allows us to get the percentage of nodes with violations compared to the total amount of nodes.

$$\text{IR} = \frac{N_{\text{violations}}}{N_{\text{total}}} \tag{4.1}$$

However, the Inaccessibility Rate does not take the severity of issues into account. Thus, we have created the *Impact-Weighted Inaccessibility Rate*(IWIR). This metric uses the impacts of Accessibility Violations found (minor, moderate, serious, critical) and assigns them to a value (1, 3, 6, 10). Our scoring reflects the non-linear increase in impact for people with disablities if a violation with a higher impact takes place within the code. Finally, the Impact-Weighted Inaccessibility Rate is calculated by creating the sum over all Violations found multiplied by the corresponding value for the severity. This sum is then divided by the amount of violations found multiplied by the highest impact

score. By dividing the sum above through the worst possible outcome, a situation where every violation is critical, allows to get an understanding of the severity of the violations found.

$$\text{IWIR} = \frac{\sum_{i=1}^{k} v_i \, w_i}{\sum_{i=1}^{k} v_i \, w_{\max}} \tag{4.2}$$

The combination of both metrics allows us to understand wether LLMs can not only decrease the amount of accessibility violations, but also its severity.

5 Experiment

5.1 Section

5.1.1 Subsection

6 Accessibility

6.1 Section

6.1.1 Subsection

7 Conclusion

7.1 Section

7.1.1 Subsection

List of Figures

List of Tables

Bibliography

[Lam94] L. Lamport. *LaTeX : A Documentation Preparation System User's Guide and Reference Manual*. Addison-Wesley Professional, 1994.