POLITECNICO DI TORINO
Repository ISTITUZIONALE

Bridging Web and Figma: Automating Large-Scale UI Dataset Generation for AI-Enhanced Design

Original
Bridging Web and Figma: Automating Large-Scale UI Dataset Generation for AI-Enhanced Design / Russo, F., Calò, T., De Russis, L.. - ELETTRONICO. - (2025), pp. 13-20. (ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS '25) Trier (DEU) 23-27 June 2025) [10.1145/3731406.3734974].

Availability:
This version is available at: 11583/3000447 since: 2025-08-04T16:00:57Z

Publisher:
Association for Computing Machinery

Published
DOI:10.1145/3731406.3734974

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

25 November 2026



Bridging Web and Figma: Automating Large-Scale UI Dataset Generation for AI-Enhanced Design

Francesca Russo
Politecnico di Torino
Dipartimento di Automatica e Informatica
Torino, Torino, Italy
francesca.russo@polito.it

Tommaso Calò
Politecnico Di Torino
Dipartimento di Automatica e Informatica
Torino, Torino, Italy
tommaso.calo@polito.it

Luigi De Russis
Politecnico di Torino
Dipartimento di Automatica e Informatica
Torino, Torino, Italy
luigi.derussis@polito.it

Abstract

Large-scale User Interface (UI) data is essential for developing Artificial Intelligence (AI)-driven tools that can support designers in creating interfaces. However, many publicly available datasets are either manually annotated, a time-consuming and costly process that limits their scale or lack crucial structural information, such as semantic labels and hierarchical relationships, necessary for effective design assistance. Moreover, no existing dataset offers a standard format designed for seamless integration of AI models into real-world design tools. In this work, we introduce a pipeline that automatically converts any HTML content into structured, Figma-compatible representations. To validate our pipeline, we apply it to WebUI, a large-scale HTML-based dataset, and conduct a comparative evaluation by training five state-of-the-art layout generation models on our data and the manually annotated Rico dataset. Experimental results demonstrate that the models achieve comparable performance across both datasets and suggest that our pipeline can effectively produce high-quality data suitable for training AI models integrable into design workflows.

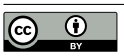
CCS Concepts

• **Human-centered computing** → **User interface design**; • **Computing methodologies** → *Machine learning*; • **Information systems** → **Web mining**.

Keywords

User Interface, Layout Generation, Web, Pipeline

ACM Reference Format:
Francesca Russo, Tommaso Calò, and Luigi De Russis. 2025. Bridging Web and Figma: Automating Large-Scale UI Dataset Generation for AI-Enhanced Design. In *Companion Proceedings of the 17th ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS Companion '25), June 23–27, 2025, Trier, Germany*. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3731406.3734974



This work is licensed under a Creative Commons Attribution 4.0 International License.

EICS Companion '25, Trier, Germany

© 2025 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-1866-3/25/06
https://doi.org/10.1145/3731406.3734974

1 Introduction

User Interfaces (UIs) are fundamental in shaping user interaction across web, mobile, and desktop applications. Achieving consistently high-quality UI designs across these diverse platforms, however, presents a significant challenge due to the inherent complexity involved in balancing functionality, usability, and aesthetics [5, 26].

To navigate these complexities, designers rely heavily on specialized tools that streamline workflows and facilitate collaboration. Among these, Figma has gained considerable popularity owing to its real-time collaborative environment and extensive plugin ecosystem [10]. In Figma, designers can efficiently manipulate UI components, dynamically adjusting layouts to swiftly iterate design concepts. These capabilities position Figma as an ideal platform for integrating Artificial Intelligence (AI)-assisted design tools into existing professional workflows.

Recent advancements in AI [17] offer promising opportunities for designers by automating repetitive tasks [27] and generating intelligent, data-driven layout suggestions [16, 31]. However, current AI-driven Figma plugins primarily leverage general-purpose Large Language Models (LLMs) that typically lack specialized, fine-grained knowledge of UI design principles and domain-specific conventions, limiting their practical utility for detailed design assistance [9].

Building specialized generative AI models for UI-assisted design entails a multi-step methodology: (1) automatically gathering extensive, diverse datasets of UI designs; (2) converting these datasets into structured formats that retain essential hierarchical and semantic information necessary for practical design tasks; (3) training specialized generative AI models tailored explicitly for UI design tasks, optionally fine-tuned on curated, high-quality samples; and (4) embedding these AI capabilities into Figma through dedicated plugins, seamlessly accessible within a designer's workflow.

Currently, these goals face substantial limitations due to existing datasets' constraints. Widely used UI datasets such as Rico [7] rely on supervised, manually annotated approaches, inherently limiting their scale and generalizability. More recently, large-scale web-based datasets such as WebUI [29], while extensive, primarily focus on tasks such as UI element detection. Critically, they lack essential structural and semantic information—including hierarchical relationships, visual element properties, and detailed component metadata—fundamental for both designers and AI models to effectively understand and manipulate UI layouts.

This paper directly addresses these limitations through two primary contributions. First, we introduce an automated pipeline capable of converting any HTML-based user interface into a structured,

Figma-compatible JSON representation. We then apply it to WebUI to explicitly capture critical structural and semantic details, such as hierarchical relationships among UI elements and visual properties (e.g., size, position), which are indispensable for UI designers. Additionally, the pipeline accommodates multiple viewport dimensions, enhancing its applicability across diverse platforms.

Second, we validate the effectiveness and quality of our automatically created dataset by training and evaluating five state-of-the-art layout generative models. By quantitatively comparing layout generation performance on our dataset with that on the manually annotated Rico dataset, we evaluate the ability of our dataset to support models in learning detailed spatial and semantic UI structures. Our results demonstrate that models trained on our unsupervised dataset achieve comparable performance to those trained on curated datasets, validating the quality and practical usability of our automatically generated, large-scale dataset. This pipeline thus significantly reduces reliance on costly manual annotations, enabling scalable and effective integration of AI-driven design tools within Figma workflows.

2 Background

Recent advancements in AI and their applications to UI design have attracted growing interest. Researchers have examined current trends, professionals' perceptions of AI-driven tools, and outlined future directions for intelligent assistants [20, 24, 25]. These studies suggest that AI has the potential to meaningfully support various stages of the UI design workflow, particularly in areas such as design inspiration search, exploration of alternative layouts, and validation against established design guidelines [25].

In response, both academic and industrial research efforts have increasingly focused on developing tools that enhance creativity by offering layout suggestions [16] or by enabling interface generation [31].

Nevertheless, while specialized techniques tailored for UI design have emerged, a significant portion of existing intelligent design assistants rely on natural language description as input to prompt-engineered general-purpose LLMs to either generate UI prototypes [1, 2] or modify existing interfaces by operating on the individual components [3, 4]. Although powerful and versatile, these models are not tailored for precise and deep understanding of UI-specific patterns, guidelines, and recommendations. As a result, their ability to support designers in complex or highly contextualized design tasks remains limited [28].

Conversely, when developing specialized AI models for AI-assisted UI design, the Rico dataset [7] stands out as the most widely adopted resource. It comprises more than 72k screens from only Android applications, sourced from almost 10k apps, and includes annotations for component hierarchies, screen metadata, and element properties. Rico was mostly built with a crowdsourcing approach involving manual data annotation, making the process time-consuming, cost-intensive, and constrained in scale. Leiva et al. [22] have conducted a manual verification of Rico UIs and have identified issues such as labeling errors and inconsistencies between UI hierarchies and their corresponding screenshots. Recent efforts have turned to automatic UI dataset generation to reduce reliance on manual labeling and to enable large-scale training, especially leveraging the Web as a rich and publicly accessible source of UI data. Webzeitgeist [21], for instance, is an automatic crawler used to collect more than 100k web UI interfaces. A similar method is also used for the creation of the WebUI dataset [29], which includes 400k rendered web pages and focuses on their static properties.

Despite their extensive size, these datasets lack semantic and hierarchical structures, providing only screenshots and limited metadata without the explicit semantic and structural information—including hierarchical relationships and visual properties—crucial for UI designers to effectively understand and manipulate layouts.

To explicitly retrieve the semantic and spatial information (e.g., component hierarchies, visual elements) from HTML files, where these information are hidden within complex and unnecessarily nested structures, we propose a pipeline for converting any HTML page into a JSON representation. This representation extracts the essential spatial and semantic information needed by both designers and AI design support tools, enabling more effective layout manipulation and understanding. Additionally, to our knowledge, no existing dataset offers a data representation directly suitable for real-world design tools, creating a barrier to developing specialized AI-assisted design systems. Finally, unlike Rico, which is limited to Android apps, our approach leverages HTML as a platform-independent source that supports rendering and extracting interfaces across multiple viewports.

As a result, the proposed pipeline enables the construction of high-scale datasets for training specialized UI design assistants. To validate the quality of our automatically generated dataset, we adopt a generative model evaluation approach that has emerged as a robust method for comparing data distributions in related domains [13, 18]. By training and evaluating five state-of-the-art layout generative models on both our dataset and the manually annotated Rico dataset, we quantitatively assess whether our unsupervised pipeline produces training data of comparable quality.

In summary, while previous works have significantly advanced the collection and utilization of UI datasets, substantial gaps remain in scalability, dataset annotation quality, and practical integration into design workflows. Our work extends beyond current limitations by introducing an automated pipeline capable of transforming HTML-based interfaces into richly structured, Figma-compatible data, thus enabling designers and researchers to leverage vast web-based resources effectively in AI-driven UI design.

3 From HTML to Figma: Data Pipeline and Generative Evaluation

In this section, we present our pipeline for extracting a Figma-compatible representation from web interfaces and discuss how we evaluate its output using state-of-the-art generative models.

3.1 Data Extraction Pipeline

Figure 1 provides an overview of our pipeline. Four sequential stages can be identified: (i) data collection, (ii) data cleaning, (iii) application of heuristics, and (iv) data saving.

Data collection. We deploy Puppeteer [8], a headless browser that autonomously accesses and navigates each page, while supporting flexible viewport configurations (i.e., any desired screen

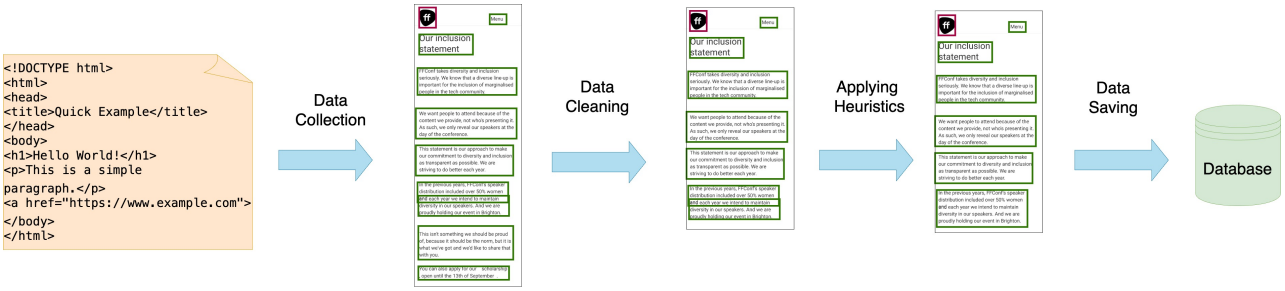


Figure 1: Overview of our pipeline, with its four stages: (i) data collection in which a headless browser is deployed in a given viewport and the UI is converted into a Figma-compatible JSON format, (ii) data cleaning in which non-visible elements and irrelevant attributes are removed, (iii) application of heuristics in which we apply domain-specific rules to filter duplicate, overlapping, or small elements to enhance dataset quality, and (iv) saving. For illustrative purposes, each stage is graphically represented by the picture obtained by applying the JSON content of the stage on the reference web-page screenshot.

size and orientation), enabling us to capture interfaces as they would appear across different devices or layouts. We then use the parser of the *Builder.io for Figma*[1] module into the page to extract a Figma-compatible JSON representation. The module parses HTML code and inline CSS, identifies key tags (e.g., `<div>`, `<img>`, `<p>`), and translates them into Figma-equivalent objects, such as **frames**, **text layers**, **image layers**, and **vector layers**. In addition, the pipeline captures a screenshot of the rendered viewport to serve as a reference image.

Data cleaning. In the second stage, the pipeline retains only visible elements within the chosen viewport. We then remove stylistic attributes empirically found to have minimal impact on the visual fidelity of the UI[2]. Such attributes were identified through various visual examination during the pipeline design phase.

Application of heuristics. While the first two stages constitute a best-effort, unsupervised procedure, they do not guarantee perfect results. Therefore, we refine the data with four domain-specific heuristics, derived from the literature [32]:

(1) Remove duplicate elements based on Intersection over Union (IoU) when their overlap exceeds $T = 0.9$, retaining only the largest bounding box.
(2) Merge bounding boxes sharing the same label when they are within a distance $D = 5$ pixels (based on the Euclidean distance of their edges).
(3) Filter out elements with an area smaller than 100 pixels to exclude minor artifacts.
(4) Discard UIs with fewer than 3 elements.

Data saving. Finally, the pipeline stores the result in structured JSON files that Figma can natively render. These files serve as training data for AI models, bridging raw HTML-based UIs and real-world design tools. Figure 2 shows two random WebUI interfaces and the corresponding output of our pipeline imported in Figma. Finally, Appendix A reports a short example of a Figma-compatible JSON file, extracted from the application of the pipeline.

[1]https://github.com/BuilderIO/figma-html, last visited on March 5, 2025
[2]The complete list of discarded attributes includes: `imageHash`, `name`, `blendMode`, `textCase`, `strokeWeight`, `radius`, `visible`, `horizontal`, `vertical`, `heightType`, `widthType`, `position`, `topLeftRadius`, `topRightRadius`, `bottomRightRadius`, `bottomLeftRadius`, `data`, `constraints`, `lineHeight`, `textAlignHorizontal`, `opacity`, `backgrounds`.

3.2 Generative Evaluation

To assess the suitability of the dataset produced by our pipeline, we train five modern generative models (described in Section 3.2.1) capable of layout synthesis and reconstruction. Unlike simpler discriminative tasks (e.g., classification), generative modeling demands a rich understanding of complex spatial structures and element relationships. By observing how well these models learn to reproduce realistic designs under established quantitative metrics, we gain direct insight into whether the dataset effectively captures the diversity and complexity of real-world interfaces. In particular, we compare performance on our dataset to those of the same models trained on Rico [7], allowing us to evaluate whether our pipeline's automatically generated data achieves comparable learnability and design fidelity.

We apply our pipeline to the HTML interfaces of WebUI-70k [29], setting the viewport to 390×844 pixel (i.e., iPhone 12 Pro) for consistency with the mobile-focused Rico dataset [7]. We then evaluate the resulting dataset by training five state-of-the-art layout generation models — BART [23], LayoutTransformer [12], LayoutDM [15], MaskGIT [6], and VQ-Diffusion [11] — with publicly available implementations and accessible via the official LayoutDM repository[3].

3.2.1 Models Description. **BART** [23] is a Transformer-based model pre-trained as a denoising autoencoder strategy. It corrupts a sequence with a noising function and then learns to reconstruct the original sequence, supporting both generation and comprehension tasks.

LayoutTransformer [12] uses a Transformer backbone to learn contextual relationships in layout elements, generating new layouts conditioned on existing context.

LayoutDM [15] is a discrete diffusion model for layout generation, gradually refining a noisy initial layout into a coherent structure through iterative denoising steps.

MaskGIT [6] is a non-autoregressive model that iteratively predicts the content of masked sequences, fostering efficient and high-quality visual content generation.

[3]https://github.com/CyberAgentAILab/layout-dm, last visited on March 4, 2025

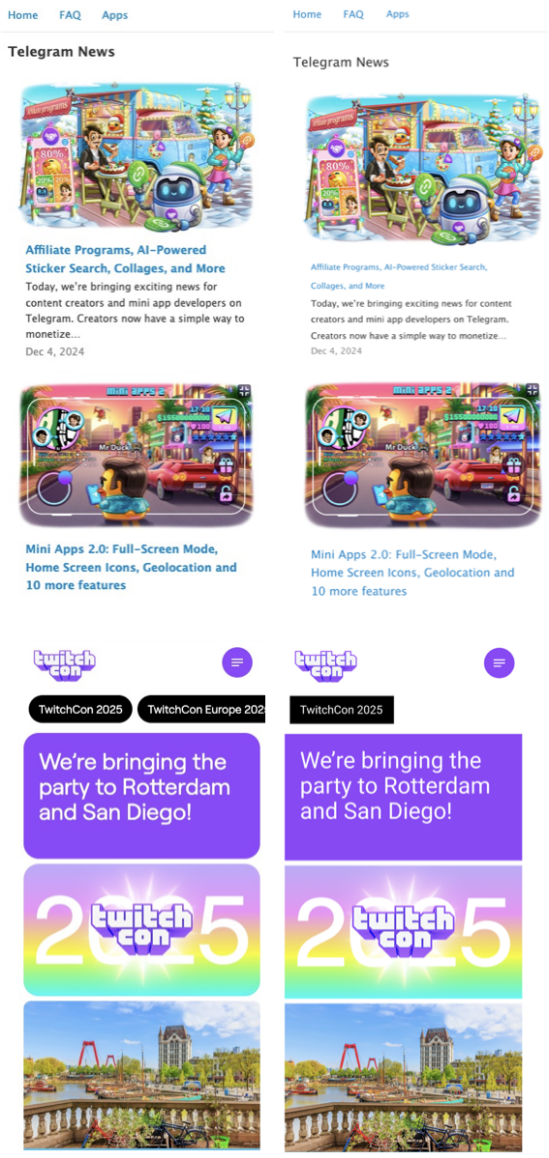


Figure 2: Comparison between two random WebUI interfaces and the corresponding output of our pipeline imported in Figma. Left: Original WebUI interfaces. Right: Outputs of our pipeline imported in Figma. The output of our pipeline preserves the original design fidelity.

VQ-Diffusion [11] combines vector quantization with diffusion processes, discretizing data into latent variables and progressively denoising them to generate high-quality samples.

3.2.2 Experimental Setup. For reproducibility, we detail the experimental setup used for training and evaluation. We use the AdamW optimizer with a learning rate of 5.0×10^{-4}, $\beta_1 = 0.9$, and $\beta_2 = 0.98$. Models are trained from scratch for 75 epochs. Both datasets are split into training, validation, and test sets using a 70%-15%-15% split. Preprocessing includes K-Means clustering for quantization of bounding box attributes, and a maximum layout size of 25 elements is used, with [PAD] tokens added for variable-length layouts. The models are trained with diffusion timesteps $T = 100$ and a loss weight $\lambda = 0.1$. In models that employ a Transformer-based backbone, we use a configuration consisting of 4 layers, 8 attention heads, 512 embedding dimensions, 2048 hidden dimensions, and a 0.1 dropout rate. Only labels that overlap between our dataset and Rico are considered (i.e., *TEXT*, *IMAGE*, *SVG*), mapped to *Text*, *Image*, and *Icon* categories in Rico. We evaluate three conditioned layout generation tasks:

(1) **Category → Size + Position**: Predict the bounding box given an element's category.
(2) **Category + Size → Position**: Predict the bounding box position given category and size.
(3) **Completion**: Given a partial layout, generate additional elements to fill the design.

We use FID [14] to measure distributional similarity between generated and real layouts, and MaxIoU [19] to evaluate how accurately predicted bounding boxes align with ground-truth. Together, these metrics capture both global realism and local geometric correctness, thereby gauging the effectiveness of our dataset in training layout generation models.

4 Results

Our pipeline was applied to the WebUI-70k [29] dataset, from which we successfully extracted 43k samples. Due to robust security protocols on some websites, it was not possible to inject our web plugin code, necessitating the exclusion of these samples. Interfaces with fewer than three UI elements were omitted from our analysis, according to the heuristics stage of the pipeline. This selection criterion was essential for maintaining the integrity and relevance of the data produced by the pipeline, ensuring that the results are both robust and representative of well-structured interfaces. The entire application process of our pipeline spanned approximately one day. Notably, this process was fully automated and did not require human intervention, which significantly improved the efficiency and scalability of our data transformation.

The quantitative results for the five layout generation models on both our dataset and Rico, for the three selected conditioned layout generation tasks, are reported in Table 1 in terms of FID and MaxIoU.

While MaxIoU consistently exhibits slightly lower values on our dataset compared to Rico, the overall similarity, as measured by FID scores, indicates a balanced situation, with models trained on Rico performing slightly better in some cases, and our dataset in others. Notably, models trained on our dataset achieve the lowest average FID score for two generation tasks out of three, namely *Category + Size → Position* (i.e., 4.15 for our dataset vs. 4.24 for Rico) and *Completion* (i.e., 29.59 vs. 46.35). Conversely, on average, MaxIoU scores are generally lower across all tasks when models are trained with our dataset, particularly in the *Category + Size → Position* task. Figure 3 shows the original layout, the condition and the layout outputs of the five layout generation models for the three conditioned tasks, on both our dataset and Rico.

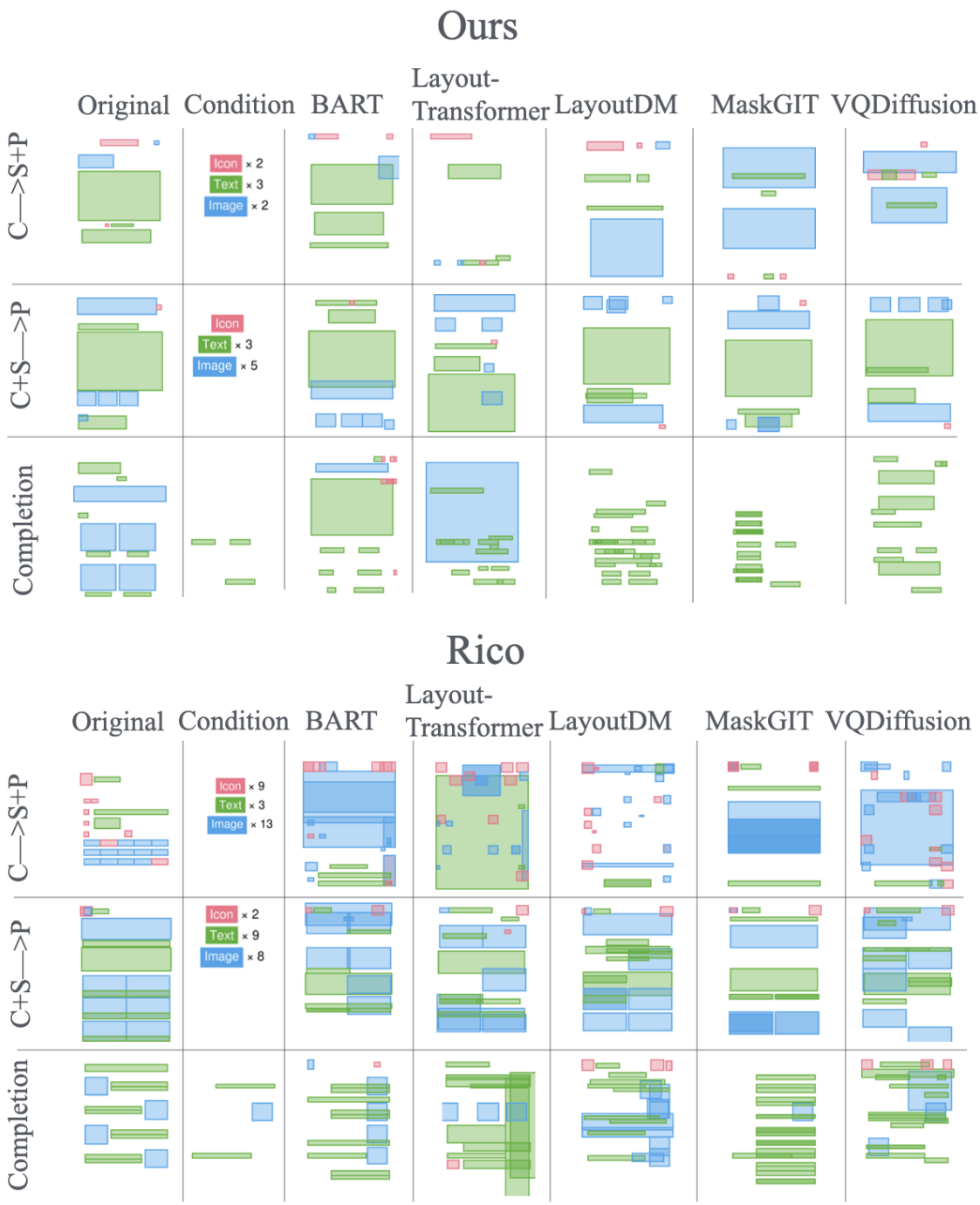


Figure 3: Original layout, condition and layout outputs of the five layout generation models for the three conditioned tasks, with pink bounding boxes for icon elements, green bounding boxes for text elements, and blue bounding boxes for image elements. For *Category → Size + Position* and *Category + Size → Position*, the numbers near the labels under the Condition column represent the number of elements of that category whose properties need to be predicted. Top: our dataset. Bottom: Rico.

Table 1: Experimental FID and MaxIoU results for the five layout generation models on both our dataset and Rico, for three conditioned layout generation tasks (*Category → Size + Position*, *Category + Size → Position*, *Completion*). Values in bold represent the best scores for each metric and task across models column.

Model	Category → Size + Position				Category + Size → Position				Completion			
	Ours		Rico		Ours		Rico		Ours		Rico	
	FID ↓	MaxIoU ↑	FID ↓	MaxIoU ↑	FID ↓	MaxIoU ↑	FID ↓	MaxIoU ↑	FID ↓	MaxIoU ↑	FID ↓	MaxIoU ↑
BART [23]	**2.10**	**0.17**	2.10	0.20	**1.79**	**0.22**	1.85	0.26	**2.25**	0.24	12.08	0.23
LayoutTransformer [15]	10.04	**0.17**	6.99	0.18	5.78	0.21	4.63	0.22	8.84	0.25	**5.21**	**0.29**
LayoutDM [15]	2.75	**0.17**	**1.95**	**0.23**	2.02	0.20	**1.40**	**0.30**	5.03	0.21	11.67	0.28
MaskGIT [6]	30.80	0.14	30.67	0.21	8.26	0.22	11.08	0.26	125.54	**0.13**	189.82	0.14
VQ-Diffusion [11]	3.43	**0.17**	2.61	0.20	2.92	0.20	2.26	0.24	6.29	0.21	12.96	0.23

For the *Category → Size + Position* and *Category + Size→ Position* tasks, the models are able to learn spatial relationships between UI elements on both datasets. Conversely, the *Completion* task is inherently more complex to solve, leading to the generation of confusing layouts for both datasets.

While experimental results vary by model, overall performance on our dataset is comparable to that on Rico, leading to the following considerations:

(1) It is possible to leverage the abundance and diversity of HTML-based data to create, without human supervision, large and structured datasets that can be valid alternatives to currently available datasets (e.g., Rico) for training state-of-the-art AI models.
(2) The pipeline could also be used to transform N HTML-based data into N uniformly-formatted JSON data representations. This approach supports the creation of extensive UI datasets with a standard structure, while preserving and leveraging the unique features of the original datasets.
(3) The compatibility with Figma facilitates experimentation with AI-driven applications directly usable by end-users (e.g., UI designers), enabling the development of accessible and practical UI design assistants.

These findings underscore the potential of this work to advance research and development of AI-based tools that can assist in UI design tasks.

5 Conclusions and Future Work

In this work, we presented an innovative HTML-to-Figma pipeline that addresses the challenges of large-scale UI dataset generation for training AI models. We have applied this pipeline to the WebUI dataset, generating a Figma-compatible comprehensive representation from HTML-based samples. By evaluating five state-of-the-art layout generation models on the WebUI dataset transformed using our pipeline, we demonstrate its effectiveness in automating the extraction of UI elements from HTML content into Figma-compatible JSON files, aiming to facilitate the deployment of AI-based design assistants in Figma.

The empirical results across three tasks reveal that models trained on our automatically-generated dataset achieve comparable performance to those trained on the manually-annotated Rico dataset. This finding is particularly significant as it suggests the viability of leveraging vast available HTML content for creating training data without the need for manual annotation, minimizing the time and costs for the creation of the dataset. Our approach allows to avoid errors caused by human annotators and maintains high data quality for training advanced AI models. The potential applications of these models in engineering interactive systems and UI design workflows include automated layout generation, style-conditioned design variations, structural design exploration, layout completion, and iterative layout refinement.

Despite the effectiveness and scalability of our unsupervised pipeline, there are limitations to address in future research. First, we rely on empirically selected filtering heuristics (e.g., bounding-box overlap and minimum size), which might merge or discard important components. A more adaptive approach that learns these thresholds from data would likely yield more nuanced results. Second, the pipeline flattens semantic classes (e.g., TextButton, Images) to a limited set of Figma-equivalent objects. This might prevent AI systems to learn fine-grained semantic associations. Third, we have not conducted a direct evaluation of the accuracy of the generated annotations as Leiva et al. did in Enrico [22], which included a systematic assessment of dataset quality to validate the reliability and consistency of the extracted UI elements.

Additionally, while our experiments focused on evaluating layout generation, our structured representation opens the door to other designer-related tasks such as design classification, retrieval, and further downstream applications. For instance, the integration of a vision-language model (VLM) fine-tuned on UI data (e.g., FerretUI [30]) could enable automatic identification of more fine-grained element categories, improving both dataset quality and tasks like sketch-to-code directly within Figma. Moreover, designing a domain-specific language (DSL) tailored for Figma could streamline autoregressive generation of user interfaces in Figma's native format, bridging the gap between high-level design specifications and production-ready layouts. We plan to explore these avenues in future work, aiming to make our pipeline even more robust, flexible, and widely applicable to real-world UI design scenarios.

Acknowledgments

We acknowledge the CINECA award under the IscrC UIML initiative, for the availability of high-performance computing resources and support.

References

[1] 2017. TeleportHQ: Low-code Front-end Design & Development. Retrieved 2025-03-21 from https://teleporthq.io/

[2] 2018. Uizard: App, Web and Ul Design Made Easy. Retrieved 2025-03-21 from https://uizard.io/

[3] 2022. Visily: Al-Powered Wireframing and Design. Retrieved 2025-03-21 from https://www.visily.ai/

[4] 2023. v0: Generative UI. Retrieved 2025-03-21 from https://v0.dev/

[5] Eren Akça and Ömer Özgür Tanriöver. 2021. A comprehensive appraisal of perceptual visual complexity analysis methods in GUI design. *Displays* 69 (2021), 102031. doi:10.1016/j.displa.2021.102031

[6] Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. 2022. Maskgit: Masked generative image transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11315–11325.

[7] Biplab Deka, Zifeng Huang, Chad Franzen, Joshua Hibschman, Daniel Afergan, Yang Li, Jeffrey Nichols, and Ranjitha Kumar. 2017. Rico: A Mobile App Dataset for Building Data-Driven Design Applications. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology* (Québec City, QC, Canada) *(UIST '17)*. Association for Computing Machinery, New York, NY, USA, 845–854. doi:10.1145/3126594.3126651

[8] Chrome DevTools. 2017. Puppeteer - Chrome. Retrieved 2025-01-18 from https://developer.chrome.com/docs/puppeteer/

[9] Peitong Duan, Chin-Yi Cheng, Gang Li, Bjoern Hartmann, and Yang Li. 2024. UICrit: Enhancing Automated Design Evaluation with a UI Critique Dataset. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology* (Pittsburgh, PA, USA) *(UIST '24)*. Association for Computing Machinery, New York, NY, USA, Article 46, 17 pages. doi:10.1145/3654777.3676381

[10] Figma, Inc. 2016. Figma: the collaborative interface design tool. Retrieved 2025-01-21 from https://www.figma.com/

[11] Shuyang Gu, Dong Chen, Jianmin Bao, Fang Wen, Bo Zhang, Dongdong Chen, Lu Yuan, and Baining Guo. 2022. Vector Quantized Diffusion Model for Text-to-Image Synthesis. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 10686–10696. doi:10.1109/CVPR52688.2022.01043

[12] Kamal Gupta, Justin Lazarow, Alessandro Achille, Larry S Davis, Vijay Mahadevan, and Abhinav Shrivastava. 2021. Layouttransformer: Layout generation and completion with self-attention. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 1004–1014.

[13] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. 2017. GANs trained by a two time-scale update rule converge to a local nash equilibrium. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 6629–6640.

[14] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. 2017. GANs trained by a two time-scale update rule converge to a local nash equilibrium. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 6629–6640.

[15] Naoto Inoue, Kotaro Kikuchi, Edgar Simo-Serra, Mayu Otani, and Kota Yamaguchi. 2023. LayoutDM: Discrete Diffusion Model for Controllable Layout Generation. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 10167–10176. doi:10.1109/CVPR52729.2023.00980

[16] Yue Jiang, Changkong Zhou, Vikas Garg, and Antti Oulasvirta. 2024. Graph4GUI: Graph Neural Networks for Representing Graphical User Interfaces. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '24)*. Association for Computing Machinery, New York, NY, USA, Article 988, 18 pages. doi:10.1145/3613904.3642822

[17] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling Laws for Neural Language Models. arXiv:2001.08361 [cs.LG] https://arxiv.org/abs/2001.08361

[18] Kotaro Kikuchi, Edgar Simo-Serra, Mayu Otani, and Kota Yamaguchi. 2021. Constrained Graphic Layout Generation via Latent Optimization. In *Proceedings of the 29th ACM International Conference on Multimedia*. 88–96.

[19] Kotaro Kikuchi, Edgar Simo-Serra, Mayu Otani, and Kota Yamaguchi. 2021. Constrained Graphic Layout Generation via Latent Optimization. In *Proceedings of the 29th ACM International Conference on Multimedia* (Virtual Event, China) *(MM '21)*. Association for Computing Machinery, New York, NY, USA, 88–96. doi:10.1145/3474085.3475497

[20] Tiffany Knearem, Mohammed Khwaja, Yuling Gao, Frank Bentley, and Clara E Kliman-Silver. 2023. Exploring the future of design tooling: The role of artificial intelligence in tools for user experience professionals. In *Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI EA '23)*. Association for Computing Machinery, New York, NY, USA, Article 384, 6 pages. doi:10.1145/3544549.3573874

[21] Ranjitha Kumar, Arvind Satyanarayan, Cesar Torres, Maxine Lim, Salman Ahmad, Scott R. Klemmer, and Jerry O. Talton. 2013. Webzeitgeist: design mining the web. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Paris, France) *(CHI '13)*. Association for Computing Machinery, New York, NY, USA, 3083–3092. doi:10.1145/2470654.2466420

[22] Luis A. Leiva, Asutosh Hota, and Antti Oulasvirta. 2021. Enrico: A Dataset for Topic Modeling of Mobile UI Designs. In *22nd International Conference on Human-Computer Interaction with Mobile Devices and Services* (Oldenburg, Germany) *(MobileHCI '20)*. Association for Computing Machinery, New York, NY, USA, Article 9, 4 pages. doi:10.1145/3406324.3410710

[23] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (Eds.). Association for Computational Linguistics, Online, 7871–7880. doi:10.18653/v1/2020.acl-main.703

[24] Jie Li, Hancheng Cao, Laura Lin, Youyang Hou, Ruihao Zhu, and Abdallah El Ali. 2024. User Experience Design Professionals' Perceptions of Generative Artificial Intelligence. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '24)*. Association for Computing Machinery, New York, NY, USA, Article 381, 18 pages. doi:10.1145/3613904.3642114

[25] Yuwen Lu, Chengzhi Zhang, Iris Zhang, and Toby Jia-Jun Li. 2022. Bridging the Gap Between UX Practitioners' Work Practices and AI-Enabled Design Support Tools. In *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI EA '22)*. Association for Computing Machinery, New York, NY, USA, Article 268, 7 pages. doi:10.1145/3491101.3519809

[26] Aliaksei Miniukovich, Simone Sulpizio, and Antonella De Angeli. 2018. Visual complexity of graphical user interfaces. In *Proceedings of the 2018 International Conference on Advanced Visual Interfaces* (Castiglione della Pescaia, Grosseto, Italy) *(AVI '18)*. Association for Computing Machinery, New York, NY, USA, Article 20, 9 pages. doi:10.1145/3206505.3206549

[27] Vinoth Pandian Sermuga Pandian, Sarah Suleri, Christian Beecks, and Matthias Jarke. 2021. MetaMorph: AI Assistance to Transform Lo-Fi Sketches to Higher Fidelities. In *Proceedings of the 32nd Australian Conference on Human-Computer Interaction* (Sydney, NSW, Australia) *(OzCHI '20)*. Association for Computing Machinery, New York, NY, USA, 403–412. doi:10.1145/3441000.3441030

[28] Chenglei Si, Yanzhe Zhang, Ryan Li, Zhengyuan Yang, Ruibo Liu, and Diyi Yang. 2025. Design2Code: Benchmarking Multimodal Code Generation for Automated Front-End Engineering. arXiv:2403.03163 [cs.CL] https://arxiv.org/abs/2403.03163

[29] Jason Wu, Siyan Wang, Siman Shen, Yi-Hao Peng, Jeffrey Nichols, and Jeffrey P Bigham. 2023. WebUI: A Dataset for Enhancing Visual UI Understanding with Web Semantics. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI '23)*. Association for Computing Machinery, New York, NY, USA, Article 286, 14 pages. doi:10.1145/3544548.3581158

[30] Keen You, Haotian Zhang, Eldon Schoop, Floris Weers, Amanda Swearngin, Jeffrey Nichols, Yinfei Yang, and Zhe Gan. 2024. Ferret-UI: Grounded Mobile UI Understanding with Multimodal LLMs. In *Computer Vision – ECCV 2024: 18th European Conference, Milan, Italy, September 29–October 4, 2024, Proceedings, Part LXIV* (Milan, Italy). Springer-Verlag, Berlin, Heidelberg, 240–255. doi:10.1007/978-3-031-73039-9_14

[31] Ning Yu, Chia-Chih Chen, Zeyuan Chen, Rui Meng, Gang Wu, Paul Josel, Juan Carlos Niebles, Caiming Xiong, and Ran Xu. 2025. LayoutDETR: Detection Transformer Is a Good Multimodal Layout Designer. In *Computer Vision – ECCV 2024*, Aleš Leonardis, Elisa Ricci, Stefan Roth, Olga Russakovsky, Torsten Sattler, and Gül Varol (Eds.). Springer Nature Switzerland, Cham, 169–187.

[32] Xiaoyi Zhang, Lilian de Greef, Amanda Swearngin, Samuel White, Kyle Murray, Lisa Yu, Qi Shan, Jeffrey Nichols, Jason Wu, Chris Fleizach, Aaron Everitt, and Jeffrey P Bigham. 2021. Screen Recognition: Creating Accessibility Metadata for Mobile Applications from Pixels. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 275, 15 pages. doi:10.1145/3411764.3445186

A Extract of a Figma-compatible JSON file

```
{
  "layers": [
    {
    "type": "FRAME",
    "width": 390,
    "height": 844,
    "x": 0,
    "y": 0,
    "children": [
      {
        "type": "FRAME",
          "clipsContent": false,
          "x": 0,
          "y": 0,
          "width": 390,
```

```
        "height": 734,
        "children": [
          {
            "type": "FRAME",
            "clipsContent": false,
            "x": 0,
            "y": 0,
            "width": 390,
            "height": 70,
            "children": [
              {
                "type": "SVG",
                "x": 20,
                "y": 13,
                "width": 60,
                "height": 60,
                "svg": "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n<svg width
                       =\"192px\" height=\"191px\" viewBox=\"0 0 192 191\"
                       version=\"1.1\" xmlns=\"http://www.w3.org/2000/svg\"
                       xmlns:xlink=\"http://www.w3.org/1999/xlink\">\n  <!--
                       Generator: Sketch 52.6 (67491) - http://www.
                       bohemiancoding.com/sketch -->\n  <title>logo</title>\n
                        <desc>Created with Sketch.</desc>\n  <g id=\"logo\"
                       stroke=\"none\" stroke-width=\"1\" fill=\"none\" fill-
                       rule=\"evenodd\">\n    <path d=\"M84.5,189.6 C97.5,
                       186.5 114.9,177.3 138.5,161.2 C153.2,151.1 157.1,148
                       166.3,139.1 C183.1,122.9 190.4,107.1 191.7,84 C193,60.1
                        184.7,39.1 168.9,26.3 C148.8,10 108.3,-0.5 66,-0.4 C50
                       .3,-0.4 46.1,-0.1 38.6,1.8 C13,8.2 2.1,22.1 0.3,50.5 C
                       -1.6,81.7 13.3,134.3 32.1,163 C34.5,166.6 39.6,172.7
                       43.4,176.6 C56.3,189.5 68.4,193.3 84.5,189.6 Z\" id=\"
                       Path\" fill=\"#000000\" fill-rule=\"nonzero\"/>\n    <
                       path d=\"M62,109 L62,88 L55,88 L48,88 L48,81.5 L48,75
                       L55,75 L62,75 L62,71.3 C62,58.3 71.6,49 85,49 C92.5,49
                       93.3,49.7 92.5,55.2 C91.6,61.3 91.2,61.6 85.5,62.3 C79
                       .5,62.9 77,65.5 77,71.1 L77,75 L87.5,75 L98,75 L98,69.7
                        C98,62.2 101.3,56.4 107.5,52.5 C111.8,49.8 113.2,49.5
                       120.1,49.5 C124.3,49.5 128,49.7 128.3,50 C128.6,50.3
                       128.4,53.1 127.8,56.3 L126.8,62 L122.5,62 C116.6,62
                       113.1,65 112.3,70.8 L111.7,75 L120.4,75 L129,75 L129,
                       81.5 L129,88 L120.5,88 L112,88 L112,109 L112,130 L105,
                       130 L98,130 L98,109 L98,88 L87.5,88 L77,88 L76.8,108.8
                       L76.5,129.5 L69.3,129.8 L62,130.1 L62,109 Z\" id=\"Path
                       \" fill=\"#FFFFFF\"/>\n  </g>\n</svg>"
```



```
              },
              {
                "x": 323,
                "y": 36,
                "width": 47,
                "height": 18,
                "type": "TEXT",
                "characters": "Menu",
                "fills": [
                  {
                    "type": "SOLID",
                    "color": {
                      "r": 0.2,
                      "g": 0.2,
                      "b": 0.2
                    }
                  }
                ],
                "fontSize": 18,
                "fontFamily": "BasierSquare"
              }
            ]
          },
          ...
        ]
      }
    ]
  }
}
```