GUI-ReRank: Enhancing GUI Retrieval with Multi-Modal LLM-based Reranking

Kristian Kolthoff[†], Felix Kretzer[‡], Alexander Maedche[‡], Simone Paolo Ponzetto[§], and Christian Bartelt[†]
[†]Institute for Software and Systems Engineering, Clausthal University of Technology, Clausthal, Germany
Email: {kristian.kolthoff, christian.bartelt} @tu-clausthal.de
[‡]Human-Centered Systems Lab, Karlsruhe Institute of Technology, Karlsruhe, Germany
Email: {felix.kretzer, alexander.maedche} @kit.edu
[§]Data and Web Science Group, University of Mannheim, Mannheim, Germany
Email: simone@informatik.uni-mannheim.de

Abstract—**Graphical User Interface (GUI) prototyping is a fundamental component in the development of modern interactive systems, which are now ubiquitous across diverse application domains. GUI prototypes play a critical role in requirements elicitation by enabling stakeholders to visualize, assess, and refine system concepts collaboratively. Moreover, prototypes serve as effective tools for early testing, iterative evaluation, and validation of design ideas with both end users and development teams. Despite these advantages, the process of constructing GUI prototypes remains resource-intensive and time-consuming, frequently demanding substantial effort and expertise. Recent research has sought to alleviate this burden through natural language (NL)-based GUI retrieval approaches, which typically rely on embedding-based retrieval or tailored ranking models for specific GUI repositories. However, these methods often suffer from limited retrieval performance and struggle to generalize across arbitrary GUI datasets. In this work, we present *GUI-ReRank*, a novel framework that integrates rapid embedding-based constrained retrieval models with highly effective multi-modal (M)LLM-based reranking techniques. *GUI-ReRank* further introduces a fully customizable GUI repository annotation and embedding pipeline, enabling users to effortlessly make their own GUI repositories searchable, which allows for rapid discovery of relevant GUIs for inspiration or seamless integration into customized LLM-based retrieval-augmented generation (RAG) workflows. We evaluated our approach on an established NL-based GUI retrieval benchmark, demonstrating that *GUI-ReRank* significantly outperforms state-of-the-art (SOTA) tailored Learning-to-Rank (LTR) models in both retrieval accuracy and generalizability. Additionally, we conducted a comprehensive cost and efficiency analysis of employing MLLMs for reranking, providing valuable insights regarding the trade-offs between retrieval effectiveness and computational resources. Video presentation of *GUI-ReRank* available at: https://youtu.be/_7x9UCh82ug**

Index Terms—**Automated GUI Prototyping, Natural-language-based GUI Retrieval, and MLLM-based GUI Reranking**

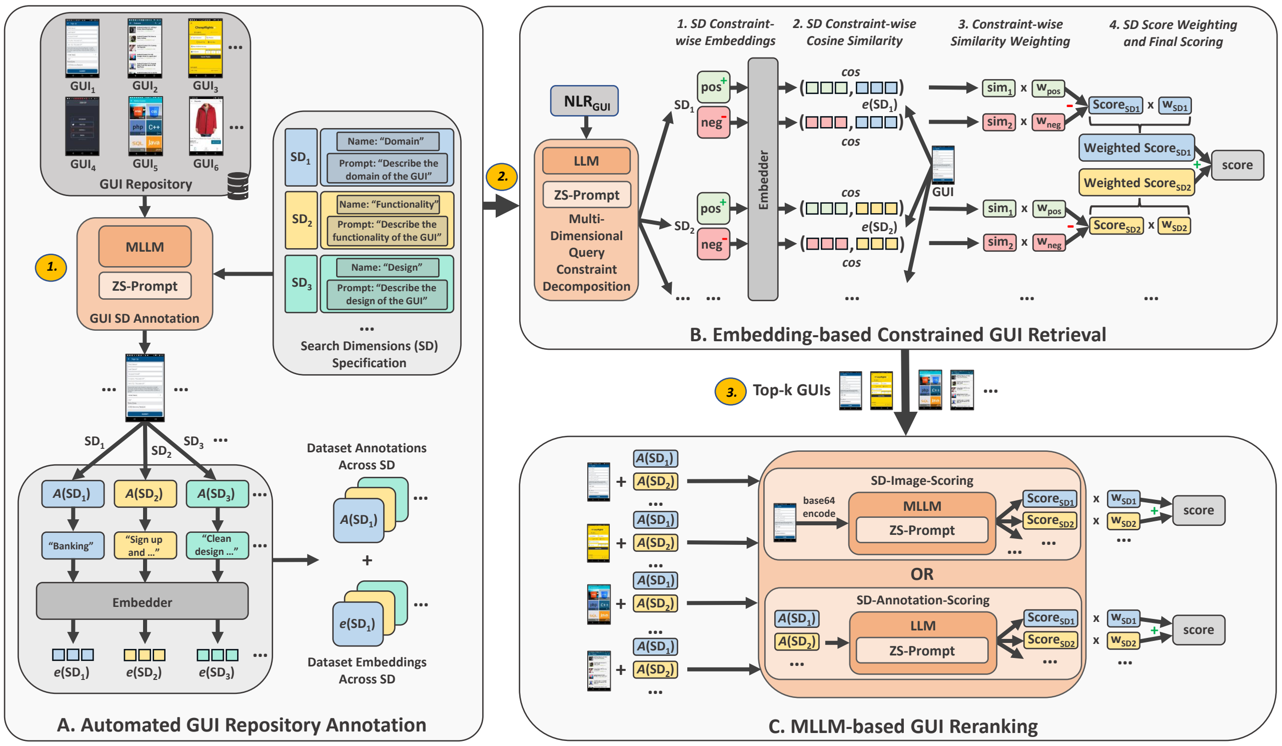
I. Introduction

A wide range of strategies have been explored to support the elicitation of requirements in user-centric software development [1]. One particularly impactful technique is GUI prototyping, which helps analysts convey their interpretation of requirements through visual means, while simultaneously offering stakeholders concrete artifacts for review and feedback. Early exposure to functional prototypes not only encourages stakeholder involvement but also promotes richer dialogue and iterative refinement of requirements [2], [3]. Nevertheless, leveraging GUI prototyping for requirements gathering often necessitates specialized expertise and entails a significant investment of manual labor. The inherently cyclical and time-consuming nature of prototyping demands repeated interactions between analysts and stakeholders, which can lead to requirements that are quickly outdated or misaligned with evolving stakeholder needs [4].

Recent advances in LLM-driven GUI generation [5]–[8] have enabled the automatic creation of user interfaces directly from natural language descriptions, offering remarkable flexibility and the ability to synthesize novel designs. Moreover, these approaches have recently been also integrated into GUI prototyping workflows [9]–[11]. However, these generative approaches can be computationally expensive and often lack mechanisms for reusing existing high-quality designs, which is particularly valuable for custom prototyping styles or specialized domains not well represented in LLM pretraining.

As a complementary direction, many approaches have been proposed for NL-based GUI retrieval, including systems such as *Guigle* [12], *GUI2WiRe* [13], and *RaWi* [14], among others [15]–[17]. While these GUI retrieval methods facilitate prototyping by rapidly retrieving relevant GUIs from large-scale repositories, they excel at enabling practitioners to efficiently search and reuse vast collections of existing GUIs in a cost-effective manner. Retrieval-based methods can also be seamlessly integrated into LLMs as RAG modules, thereby enhancing generative capabilities for custom domains not covered in pretraining data. Despite these advantages, existing retrieval approaches still exhibit limited performance on benchmark evaluations and are typically tailored to specific GUI datasets, most notably *Rico* [18], by exploiting their unique hierarchical structures. This lack of generalizability and suboptimal retrieval effectiveness highlights the need for more robust, adaptable solutions in NL-based GUI retrieval.

To close this gap, we propose *GUI-ReRank*, a novel framework that significantly advances NL-based GUI retrieval and enables customized search workflows. *GUI-ReRank* employs a two-stage ranking approach, integrating rapid embedding-based retrieval models with powerful multi-modal (M)LLM-based reranking techniques to deliver highly accurate and



GUI Repository
MLLM
ZS-Prompt
GUI SD Annotation
Name: "Domain"
Prompt: "Describe the domain of the GUI"
Name: "Functionality"
Prompt: "Describe the functionality of the GUI"
Name: "Design"
Prompt: "Describe the design of the GUI"
Search Dimensions (SD) Specification
"Banking"
"Sign up and ..."
"Clean design ..."
Embedder
Dataset Annotations Across SD
Dataset Embeddings Across SD
A. Automated GUI Repository Annotation
1. SD Constraint-wise Embeddings
2. SD Constraint-wise Cosine Similarity
3. Constraint-wise Similarity Weighting
4. SD Score Weighting and Final Scoring
LLM
ZS-Prompt
Multi-Dimensional Query Constraint Decomposition
pos
neg
Embedder
score
B. Embedding-based Constrained GUI Retrieval
Top-k GUIs
SD-Image-Scoring
base64 encode
MLLM
ZS-Prompt
OR
SD-Annotation-Scoring
LLM
ZS-Prompt
score
C. MLLM-based GUI Reranking

Fig. 1: Overview of the *GUI-ReRank* approach with *(A)* the automated GUI repository annotation and embedding pipeline, *(B)* the embedding-based constrained GUI retrieval approach and *(C)* the MLLM-based GUI reranking via images or annotations

contextually relevant results. Our approach is designed to support complex constrained natural language requirements (NLR) for search, including negation and multiple customizable dimensions such as *domain*, *functionality*, *design* and *GUI components* already at the embedding-based retrieval stage, enabling more expressive and precise GUI searches than previously possible. Furthermore, *GUI-ReRank* features a fully customizable LLM-based GUI annotation and embedding pipeline, allowing practitioners to seamlessly incorporate custom GUI datasets. *GUI-ReRank* ships with the well-known GUI dataset *Rico* [18] pre-annotated and embedded, to provide access to a large-scale and diverse GUI dataset. *GUI-ReRank* empowers developers, designers and analysts (and MLLMs via RAG pipelines) to effectively discover and reuse high-quality GUIs in diverse development contexts. Our source code, datasets and prototype are available at repository [19].

II. Approach: GUI-ReRank

GUI-ReRank consists of three main components: *(A)* an automated GUI repository annotation module that prepares custom GUI datasets via LLM-based annotation and embedding generation, *(B)* an embedding-based constrained GUI retrieval component, which uses these embeddings combined with LLM-driven query decomposition to extract and utilize positive and negative constraints across multiple search dimensions and computes an initial GUI ranking, and *(C)* an MLLM-based GUI reranking model that performs image- or text-based reranking using zero-shot prompting and computes a weighted score over multiple search dimensions. Fig. 1 shows an overview of the three components and the overall workflow.

A. Automated GUI Repository Annotation

Given a GUI repository as images, *GUI-ReRank* initiates the process by employing a zero-shot prompted MLLM to generate annotations for each GUI based on a customizable set of search dimensions (SD). The framework provides a default set of SD, including *domain*, *functionality*, *design*, *GUI components*, and *displayed text*, which are broadly applicable for many GUI retrieval scenarios. However, users can readily extend or modify these dimensions to address specific requirements, e.g., *platform* or *accessibility*, by specifying a name and an NL description for each SD. Subsequently, each annotation is processed by an embedding model to produce SD-specific embeddings, which, together with the annotations, form the foundation for efficient and flexible search across the dataset.

B. Embedding-based Constrained GUI Retrieval

To support complex semantic searches with both positive and negative constraints across multiple search dimensions, *GUI-ReRank* first decomposes the natural language requirements (NLR) using a zero-shot prompted LLM, extracting positive and negative constraints for each SD (e.g., SD_{Design}:

TABLE I: Gold standard results of previous SOTA GUI reranking approaches in comparison to MLLM-based GUI reranking

		AP	**MRR**	**Precision@k**				**HITS@k**				**NDCG@k**			
		AP	MRR	P@3	P@5	P@7	P@10	H@1	H@3	H@5	H@10	N@3	N@5	N@10	N@15
BL	**BERT-LTR-1**	0.486	0.618	0.377	0.350	0.307	0.269	0.460	0.710	0.860	0.980	0.530	0.560	0.634	0.697
	BERT-LTR-2	0.501	0.631	0.400	0.340	0.304	0.281	0.440	0.750	0.910	0.980	0.543	0.556	0.636	0.701
	BERT-LTR-3	0.499	0.626	0.363	0.354	0.317	0.287	0.450	0.730	0.860	1.000	0.517	0.554	0.646	0.694
Text	**GPT-4.1**	0.813	0.927	0.720	0.552	0.451	0.349	**0.870**	**1.000**	**1.000**	**1.000**	0.841	0.823	0.866	0.891
	GPT-4.1 Mini	0.797	0.923	0.673	0.546	0.446	0.344	0.870	0.980	1.000	1.000	0.817	0.810	0.844	0.880
	GPT-4.1 Nano	0.662	0.842	0.550	0.460	0.390	0.304	0.750	0.910	0.960	1.000	0.714	0.712	0.760	0.810
Image	**GPT-4.1**	**0.840**	0.928	**0.750**	**0.566**	**0.463**	**0.354**	0.870	0.990	1.000	1.000	**0.877**	**0.852**	**0.884**	**0.908**
	GPT-4.1 Mini	0.811	**0.930**	0.717	0.540	0.441	0.351	0.880	0.980	1.000	1.000	0.850	0.820	0.868	0.895
	GPT-4.1 Nano	0.582	0.790	0.490	0.408	0.334	0.277	0.660	0.910	0.990	1.000	0.621	0.622	0.667	0.724

		#Tokens (k=1)		**Cost (k)**		**Time (k)**	
		Input	Output	100	500	100	500
Text	**GPT-4.1**	179.77	6.00	\$.041	\$.204	2.4s	12s
	GPT-4.1 Mini	179.77	6.00	\$.008	\$.041	2.6s	13s
	GPT-4.1 Nano	179.77	7.11	\$.002	\$.010	2s	10s
Image	**GPT-4.1**	1089.02	6.00	\$.223	\$1.113	12.4s	62s
	GPT-4.1 Mini	2154.96	6.00	\$.087	\$.436	9s	45s
	GPT-4.1 Nano	3242.41	5.95	\$.033	\$.163	8s	40s

TABLE II: Token usage, cost and runtime for *GPT-4.1* models (denotes means computed over the gold standard runs, *k*=number of retrieved GUIs to rerank, #*Celery* workers = 10

$pos^{+}(\text{"}modern\text{"})$, $neg^{-}(\text{"}dark\text{"})$). For each dimension, we compute the cosine similarity between the user-specified constraints and the corresponding GUI annotations, quantifying the degree of match for both positive and negative aspects. These similarity scores are then weighted and the negative similarity is subtracted from the positive, thus a higher negative match proportionally reduces the overall positive score for each dimension. The final retrieval score for each GUI is computed as the normalized sum of the weighted similarities across all dimensions, allowing for user-defined emphasis on specific SD. This approach enables nuanced, constraint-based retrieval such as explicitly excluding certain features directly during initial and cost-effective embedding-based retrieval, overcoming the limitations of standard embedding models and thus supporting flexible, user-driven NL-based GUI retrieval.

C. Multi-Modal (M)LLM-based GUI Reranking

Building on the top-*k* GUIs retrieved in the previous stage, the multi-modal LLM-based GUI reranking component refines the results by evaluating either the GUI images or their textual SD annotations using zero-shot prompted MLLMs. For each search dimension, the MLLM assigns a score between 0 (*no match*) and 100 (*perfect match*), reflecting the relevance of each GUI to the NLR of the user. *GUI-ReRank* offers support for both text-based and image-based reranking: text-based reranking offers greater speed and cost-efficiency by leveraging abstracted SD annotations, while image-based reranking provides the most detailed assessment at the expense of higher computational cost and latency. Similar to the previous stage, the dimension-specific scores are weighted according to user preferences, summed and normalized to produce the reranking.

D. Prototype Implementation

The *GUI-ReRank* prototype is implemented as a *Django* web application with a *HTML/CSS* and *JavaScript* frontend combined with *MySQL* for data storage. Background processing tasks, such as large-scale GUI annotations, embedding generation and parallelized batch-wise reranking of top-*k* GUIs are managed using *Celery* distributed across multiple worker nodes, with *Redis* serving as the message broker to ensure efficient and scalable task execution. Our current prototype supports a range of widely employed LLMs, including *GPT* from *OpenAI*, *Google Gemini* and *Anthropic Claude Sonnet*, allowing users to select the most suitable model for their needs.

III. Experimental Evaluation

In the following, we present a comprehensive evaluation of the *GUI-ReRank* approach, comparing its effectiveness and efficiency against SOTA NL-based GUI ranking approaches.

A. Experimental Setup

We evaluated our GUI reranking models with MLLMs possessing different cost/performance trade-offs (*GPT-4.1*, *GPT-4.1 Mini* and *GPT-4.1 Nano* (accessed in *July 2025*, *temperature=0.05*) using an established gold standard for NL-based GUI ranking [14]. This benchmark comprises 100 NL queries, each paired with 20 GUIs from *Rico* [18] and enables direct comparison with state-of-the-art *BERT-LTR* models [14]. We report *Average Precision (AP)*, *Mean Reciprocal Rank (MRR)*, *Precision@k*, *HITS@k* and *NDCG@k*, similar to previous work. To assess efficiency, we report *input/output token consumption* (per reranked GUI or *k=1*), *runtime* and API *costs* (both for *k=100* and *k=500*) for each reranking configuration.

B. Evaluation Results

Table I presents the ranking performance of our MLLM-based GUI reranking models compared to previous SOTA *BERT-LTR* approaches. Both text- and image-based reranking models substantially outperform *BERT-LTR* baselines across all metrics. Comparing image-based and text-based models, we observe a higher effectiveness for image-based models. Moreover, the observed effectiveness of the models generally aligns with their performance profiles, with *GPT-4.1* achieving the highest scores, closely followed by the *Mini* model, which, however, is substantially more cost-effective. Table II summarizes the efficiency analysis. Text-based reranking is significantly more cost- and time-efficient in contrast to the image-based GUI reranking, with token usage and inference cost for *k=100* and *k=500* GUIs significantly reduced. Furthermore, when comparing text-based *GPT-4.1* against the next best model *GPT-4.1 Mini*, only 1.9% of *Average Precision* performance is lost, while reducing the cost by 79.9%. These results provide practical guidance for selecting the appropriate reranking mode and model based on resource constraints and application requirements. Overall, *GUI-ReRank* delivers SOTA GUI reranking performance with flexible efficiency trade-offs.

IV. Related Work

A variety of methods have been proposed for NL-based GUI retrieval to facilitate rapid prototyping and reuse of interface designs. *Guigle* [12] introduced one of the first search engines for GUI screenshots, leveraging traditional information retrieval techniques such as *TF-IDF* and *BM25* over multiple facets of the GUI hierarchy. Other approaches have utilized pooled *BERT* embeddings and multi-modal embedding spaces to enhance text-based GUI retrieval [17]. Advanced retrieval methods and *BERT-based Learning-to-Rank* (*LTR*) models have also been adopted in approaches like *GUI2WiRe* [13], [16] and *RaWi* [14], which exploit the hierarchical structure of datasets such as *Rico* [18] to improve retrieval accuracy. Beyond general GUI retrieval, *d.tour* [20] focuses on stylistic similarity and keywords to find websites with comparable design aesthetics, while *Gallery D.C.* [21] extracts GUI components from real-world applications and provides a multi-modal search engine supporting queries across dimensions such as *size*, *color* and *text*. Although these retrieval methods enable efficient search and reuse of large-scale GUI repositories, they often show limited performance on benchmarks and are typically tailored to specific datasets such as *Rico* [18], restricting their generalizability. In contrast, *GUI-ReRank* integrates rapid embedding-based constrained retrieval methods with advanced multi-modal LLM-based reranking and a flexible GUI dataset annotation and embedding pipeline. This combination achieves substantially higher retrieval accuracy and robustness as demonstrated in our evaluation, while also supporting customizable search dimensions and seamless adaptation to proprietary or domain-specific GUI repositories.

V. Conclusion

In this work, we introduced *GUI-ReRank*, a novel framework that combines embedding-based constrained retrieval over multiple customizable search dimensions with MLLM-based reranking to achieve state-of-the-art performance in NL-based GUI ranking. Our experiments show substantial gains over previous GUI retrieval methods and offer valuable insights into efficiency trade-offs between the different models.

Acknowledgment

This work is supported by the *Federal Ministry of Research, Technology and Space (BMFTR)* of Germany and by the *Federal Ministry for Economic Affairs and Energy (BMWE)*.

References

[1] K. Pohl, *Requirements engineering: fundamentals, principles, and techniques*. Springer Publishing Company, Incorporated, 2010.

[2] A. Ravid and D. M. Berry, “A method for extracting and stating software requirements that a user interface prototype contains,” *Requirements Engineering*, vol. 5, no. 4, pp. 225–241, 2000.

[3] M. Beaudouin-Lafon and W. Mackay, “Prototyping development and tools,” *Handbook of Human-Computer Interaction*, pp. 1006–1031, 2002.

[4] K. Schneider, “Generating fast feedback in requirements elicitation,” in *International working conference on requirements engineering: Foundation for software quality*. Springer, 2007, pp. 160–174.

[5] M. Chen *et al.*, “Evaluating large language models trained on code,” *arXiv preprint arXiv:2107.03374*, 2021.

[6] P. Brie, N. Burny, A. Sluÿters, and J. Vanderdonckt, “Evaluating a large language model on searching for gui layouts,” *Proceedings of the ACM on Human-Computer Interaction*, vol. 7, no. EICS, pp. 1–37, 2023.

[7] K. Kolthoff, F. Kretzer, L. Fiebig, C. Bartelt, A. Maedche, and S. P. Ponzetto, “Zero-shot prompting approaches for llm-based graphical user interface generation,” *arXiv preprint arXiv:2412.11328*, 2024.

[8] L. Fiebig *et al.*, “Effective gui generation: Leveraging large language models for automated gui prototyping,” 2025.

[9] K. Kolthoff *et al.*, “Interlinking user stories and gui prototyping: A semi-automatic llm-based approach,” in *2024 IEEE 32nd International Requirements Engineering Conference (RE)*. IEEE, 2024, pp. 380–388.

[10] F. Kretzer *et al.*, “Closing the loop between user stories and gui prototypes: an llm-based assistant for cross-functional integration in software development,” in *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, 2025, pp. 1–19.

[11] K. Kolthoff *et al.*, “Guide: Llm-driven gui generation decomposition for automated prototyping,” in *2025 IEEE/ACM 47th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*. IEEE, 2025, pp. 1–4.

[12] C. Bernal-Cárdenas, K. Moran, M. Tufano, Z. Liu, L. Nan, Z. Shi, and D. Poshyvanyk, “Guigle: a gui search engine for android apps,” in *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*. IEEE, 2019, pp. 71–74.

[13] K. Kolthoff, C. Bartelt, and S. P. Ponzetto, “Gui2wire: Rapid wireframing with a mined and large-scale gui repository using natural language requirements,” in *35th IEEE/ACM International Conference on Automated Software Engineering (ASE ’20)*. ACM, 2020.

[14] ——, “Data-driven prototyping via natural-language-based gui retrieval,” *Automated Software Engineering*, vol. 30, no. 1, p. 13, 2023.

[15] K. Kolthoff, “Automatic generation of graphical user interface prototypes from unrestricted natural language requirements,” in *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2019, pp. 1234–1237.

[16] K. Kolthoff, C. Bartelt, and S. P. Ponzetto, “Automated retrieval of graphical user interface prototypes from natural language requirements,” in *International Conference on Applications of Natural Language to Information Systems*. Springer, 2021, pp. 376–384.

[17] F. Huang, G. Li, X. Zhou, J. F. Canny, and Y. Li, “Creating user interface mock-ups from high-level text descriptions with deep-learning models,” *arXiv preprint arXiv:2110.07775*, 2021.

[18] B. Deka *et al.*, “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*, 2017, pp. 845–854.

[19] “Gui-rerank - github repository,” https://github.com/kristiankolthoff/GUI-ReRank, accessed: 2025-07-23.

[20] D. Ritchie, A. A. Kejriwal, and S. R. Klemmer, “d. tour: Style-based exploration of design example galleries,” in *Proceedings of the 24th annual ACM symposium on User interface software and technology*, 2011, pp. 165–174.

[21] C. Chen, S. Feng, Z. Xing, L. Liu, S. Zhao, and J. Wang, “Gallery dc: Design search and knowledge discovery through auto-created gui component gallery,” *Proceedings of the ACM on Human-Computer Interaction*, vol. 3, no. CSCW, pp. 1–22, 2019.