# Exploring Mobile UI Layout Generation using Large Language Models Guided by UI Grammar

**Yuwen Lu**[1] **Ziang Tong**[1] **Qinyi Zhao**[2] **Chengzhi Zhang**[3] **Toby Jia-Jun Li**[1]

## Abstract

The recent advances in Large Language Models (LLMs) have stimulated interest among researchers and industry professionals, particularly in their application to tasks concerning mobile user interfaces (UIs). This position paper investigates the use of LLMs for UI layout generation. Central to our exploration is the introduction of *UI grammar* — a novel approach we proposed to represent the hierarchical structure inherent in UI screens. The aim of this approach is to guide the generative capacities of LLMs more effectively and improve the explainability and controllability of the process. Initial experiments conducted with GPT-4 showed the promising capability of LLMs to produce high-quality user interfaces via in-context learning. Furthermore, our preliminary comparative study suggested the potential of the grammar-based approach in improving the quality of generative results in specific aspects.

## 1. Introduction

### 1.1. Mobile UI Layout Generation

Layout generation for User interfaces (UIs), or Graphical User Interfaces (GUIs), has been explored by researchers across AI and Human-Computer Interaction (HCI). From a machine-learning perspective, the inherent multi-modal characteristics of UIs pose interesting research challenges for effective UI modeling, understanding, and generation (Jiang et al., 2023; 2022); from an HCI perspective, UIs have been intensively studied as a medium for good user experience (UX). Various needfinding (Dow et al., 2005; Zimmerman & Forlizzi, 2017; Martelaro & Ju, 2017) and usability study (Nielsen, 1994; 2005) methodologies have been developed both in academia and industry to improve the usability, functionality, and user-friendliness of UIs. Solving these challenges is seen as an early step to improving user experience at scale and reducing the workload for UI/UX designers (Lu et al., 2022; Knearem et al., 2023).

Following the release of the large-scale mobile UI dataset RICO (Deka et al., 2017), several AI model architectures for mobile UI layout generation have been proposed. These architectures include but not limit to Generative Adversarial Network (GAN) (Li et al., 2019; Kikuchi et al., 2021), Variational Autoencoder (VAE) (Arroyo et al., 2021; Jing et al., 2023), Diffusion Model (Cheng et al., 2023; Hui et al., 2023), Graph Neural Network (GNN) (Lee et al., 2020), and other Transformer-based neural networks (Li et al., 2020; Gupta et al., 2021; Huang et al., 2021; Kong et al., 2022; Sobolevsky et al., 2023).

### 1.2. Large Language Models for UI Tasks

Recent research work has explored some of LLMs' abilities on various UI-related tasks. Wang et al. (2023) utilized Large Language Models (LLMs) to conduct 4 UI modeling tasks through in-context learning and chain-of-thought prompting. Liu et al. (2023) conducted automated GUI testing by simulating human-like interactions with GUIs using LLMs. Kargaran et al. (2023) explored user interface menu design with LLMs through natural language descriptions of designers' intentions and design goals. These efforts have demonstrated LLMs' capabilities to effectively work with UIs with careful interaction design and prompting techniques. Some experiments also exhibited competitive performance on UI task evaluation metrics, without the need for large-scale datasets or extensive training processes.

### 1.3. Research problem and objectives

In this work, we seek to explore LLMs' potential for generating mobile UI layouts. Specifically, we set out to determine how the in-context learning capabilities of LLMs can be harnessed in a one-shot learning scenario to generate high-quality UI layouts. A key challenge here involves the representation and integration of the hierarchical structure inherent in UI elements into the generation process.

[1]University of Notre Dame, Notre Dame, USA [2]Fudan University, Shanghai, China [3]Carnegie Mellon University, Pittsburgh, USA. Correspondence to: Yuwen Lu <ylu23@nd.edu>.

*AI & HCI Workshop at the $40^{th}$ International Conference on Machine Learning*, (ICML), Honolulu, Hawaii, USA. 2023. Copyright 2021 by the author(s).

In response to this problem, we propose *UI grammar*—a novel approach that accurately represents the hierarchical relationships between UI elements. This approach serves to guide the generation process of LLMs, thereby making the generation more structured and contextually appropriate. From a human-centered perspective, we discuss how the inclusion of *UI grammar* provides an intermediary layer of representation that could potentially improve the ***explainability*** and ***controllability*** of LLMs in the generation process. Users can better *understand* and *steer* LLMs' internal generation mechanisms by reviewing and editing the grammar used for coming up with the final result.

Our objectives here are twofold. First, we aim to evaluate the performance of LLMs in generating UI layouts. Second, we set out to evaluate the impact of our proposed *UI grammar* on LLMs' generation process. We comparatively assessed the generation quality with/without the integration of *UI grammar* in prompts against 3 common metrics for layout generation tasks: Maximum intersection over union (MaxIoU), Alignment, and Overlap. Our preliminary experiment results demonstrated LLMs' ability to generate high-quality mobile UI layouts through in-context learning and showcased the usefulness of *UI grammar* in improving certain aspects of generation quality.

## 2. LLM Prompting for UI Layout Generation

Wang et al. (2023) have discussed a few key aspects in constructing LLM prompts for mobile UI tasks, including *screen representation*, *UI element properties*, and *class mappings*. While prompting LLMs remain an open research area, here, we continue this line of discussion by reviewing techniques from recent work on adapting UI for authoring LLM prompts. We then propose our own prompt strategy, specifically designed for UI layout generation, and provide our rationales.

### 2.1. UI Representation for In-Context Learning

LLMs have showcased an impressive capacity for in-context learning, which involves adapting to a limited number of user-provided examples while maintaining competitive performance across a variety of tasks (Brown et al., 2020). This ability, confirmed to be an emergent ability as language models' sizes scale up (Wei et al., 2022a), offers a more streamlined alternative to the process of fine-tuning pretrained models with large datasets for domain adaptation.

UI data is inherently multi-modal and can often be represented in a variety of data formats. These include, but are not limited to, screenshots, Android view hierarchies, code implementations, and natural language descriptions (e.g., *"a welcome page for a comics reading app"*). Within existing UI datasets, such as RICO (Deka et al., 2017), each UI screen is typically offered in multiple data formats. This approach serves to capture visual, structural, and contextual information of UI screens.

This created challenges in providing UI exemplars to LLMs for in-context learning, especially given the limited context window and text-only input/output modality for existing LLMs. Here, we review recent work's approaches to adapting UI input for LLM prompting:

- Wang et al. (2023) parsed UI into **HTML** files to feed into PALM for 4 mobile UI tasks, i.e. screen question-generation, screen summarization, screen question-answering, and mapping instruction to UI action. They used the `class`, `text`, `source_id`, `content_desc` attributes to include detailed information of screen widgets.
- Liu et al. (2023) investigated using GPT-3 for automatic GUI testing through natural language conversations. They extracted static contexts using attributes of the app and screen widgets from the corresponding *AndroidManifast.xml* file, including `AppName`, `ActivityName`, `WidgetText`, and `WidgetID`, and constructed **natural language sentences** describing the UI state with these attributes.
- While Feng et al. (2023) did not directly work with UI data, they used GPT-3.5 for a 2D image layout generation, a task sharing many similarities with mobile UI layout generation. They parsed the position of image elements into **CSS** (short for Cascading Style Sheets) snippets with normalized position values as GPT input.

### 2.2. Hierarchical Structures as UI Grammar

UI elements within a screen have hierarchical relationships (Li et al., 2021; 2018), which can be reflected in the atomic design principle (Frost, 2016), the grouping feature of UI design tools like Figma, and the Android view hierarchies. Some previous work (Huang et al., 2021) flattened the hierarchical structures of UI elements and reduced the layout generation task into predicting a flattened sequence of elements and the accompanying bounding boxes. However, our assumption is that preserving the hierarchical relationship between UI elements and using them to implicitly guide the generation process can improve the generation quality.

To include such hierarchical information into our prompt to guide LLMs in generation, here we take inspiration from previous work (Kong et al., 2008; Talton et al., 2012) and define **UI Grammar** as one possible way to represent the hierarchical relationship between UI elements.

**UI Grammar** is defined as a set of production rules for describing the parent-children relationships between UI elements within a given screen hierarchical tree structure. Each production rule is of the form A → B, where A represents a parent UI element and B represents a sequence of one or more child elements. The definition resembles context-free grammar in syntax analysis (Earley, 1970), hence the name *UI Grammar*.

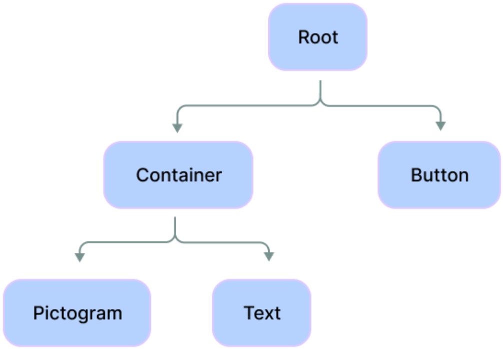


*Figure 1.* Example UI hierarchy structure. Here we can parse out UI grammar `Root` → `Container Button` and `Container` → `Pictogram Text`

.

For example, for a simple UI structure visualized in 1, we can parse out the following UI grammar based on the parent-children relationships: `Root` → `Container Button`, `Container` → `Pictogram Text`.

In this work, we conduct an initial comparative study between UI layout generation using LLMs *with* and *without* the guidance of UI grammar as part of the prompt.

### 2.3. Problem Definition and Prompt Design

We define our UI layout generation task as follows:

Given a natural language summary of a mobile UI screen $S$, we use LLM to generate a target hierarchical sequence of UI elements $T = \{o_j | j = 1, 2, \ldots, n_u\}$ where $o_j$ denotes a tuple of (`label`, `bounding_box`) for UI element $j$. These two fields in the tuple represent the *type* and *position* of the UI element on the screen.

With this problem definition, we design our prompt with the following objectives:

1. Using a UI format easy for LLMs to understand and generate layouts
2. Encapsulating hierarchical relationship between UI elements through UI grammar
3. Removing redundant non-visual information that is non-essential for layout generation

Based on these objectives, we chose to use JSON as the data format to represent UIs in UI layout generation. We selected JSON due to its advantages in the following aspects:

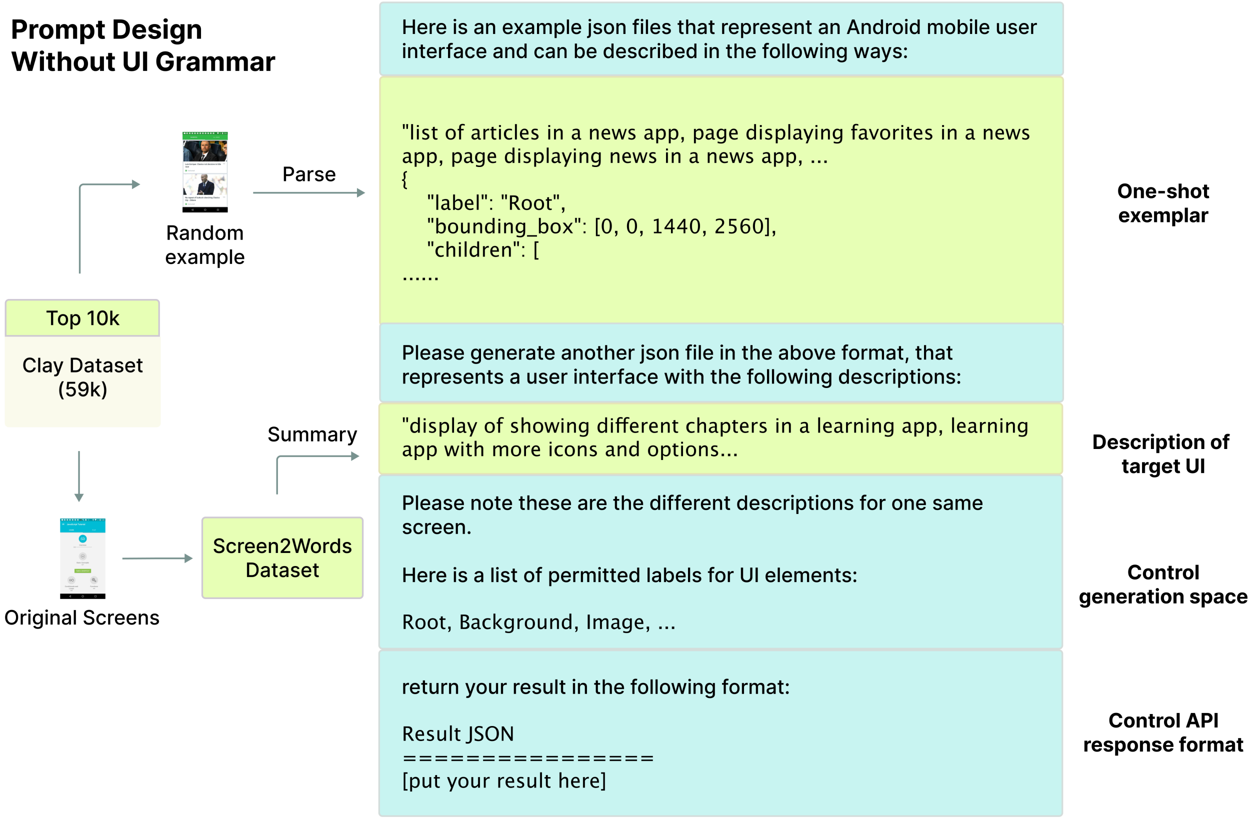
- ***Compatibility:*** JSON is ideal and commonly used for data with hierarchically structured relationships. Also, given that many LLMs use programming code in training data and prompts falling within the training data distribution tend to perform better (Wang et al., 2023), JSON is a compatible data format on both ends for our task.
- ***Flexibility:*** JSON supports multiple types of attributes for each element, suitable for representing the string.`label` and the list of integer coordinates for `bounding_box`
- ***Processing Simplicity:*** UI datasets such as RICO already use JSON to represent UI view hierarchy, reducing processing efforts.

In order to compare the efficacy of UI layout generation *with* and *without* the guidance from UI grammar, we have created two analogous pipelines for the generation process (Fig 2, Fig 3). The main differentiation between these pipelines lies in the inclusion of UI grammar in our prompts for LLMs. We will first discuss the pipeline that operates without the UI grammar, then introduce how we integrated UI grammar into the prompt to steer LLMs in generating UI layouts.

Rather than directly work with the RICO dataset containing approximately 66k unique UI screens, we used an improved dataset CLAY (Li et al., 2022) that is based on RICO. CLAY removed noise from RICO UI data by detecting UI element types and visual representation mismatches and assigning semantically meaningful types to each node. It contains 59k UI human-annotated screen layouts and contains less-noisy visual UI layout data. We also utilized the SCREEN2WORDS dataset (Wang et al., 2021) which contains natural language summaries of UI screens in RICO to construct our prompt.

#### 2.3.1. PROMPT WITHOUT UI GRAMMAR

For generating layouts without involving UI grammar (Fig. 2), we first randomly select a screen from CLAY to use as an example for our in-context learning (i.e. 1-shot prompting) and exclude it from generation to prevent data leakage. For each UI screen in CLAY, we retrieve the corresponding natural language description from SCREEN2WORDS as the description for our generation target. To control the generation result and only receive layouts with meaningful UI elements, we used the semantically meaningful list of 25 UI element labels defined in CLAY and included that in our prompt as a constraint. We also controlled LLM's API



Here is an example json files that represent an Android mobile user interface and can be described in the following ways:

```
"list of articles in a news app, page displaying favorites in a news
app, page displaying news in a news app, ...
{
    "label": "Root",
    "bounding_box": [0, 0, 1440, 2560],
    "children": [
......
```

Please generate another json file in the above format, that represents a user interface with the following descriptions:

"display of showing different chapters in a learning app, learning app with more icons and options...

Please note these are the different descriptions for one same screen.

Here is a list of permitted labels for UI elements:

Root, Background, Image, ...

return your result in the following format:

```
Result JSON
================
[put your result here]
```

*Figure 2.* Prompt 1 design for generation without UI grammar

response format for easier parsing of the generation result, as shown in Fig 2.

### 2.3.2. Prompt With UI Grammar

For our second pipeline and prompt design (Fig. 3), we introduce UI grammar as an intermediary step in UI layout generation with an architecture similar to neuro-symbolic models (Sarker et al., 2021). Instead of asking LLMs to directly generate the final screen layout, in the 1-shot example, we describe UI layout generation as a 2-step process: first, we introduce the list of UI grammar in the screen, then explain how we can generate the example UI layout using the provided UI grammar.

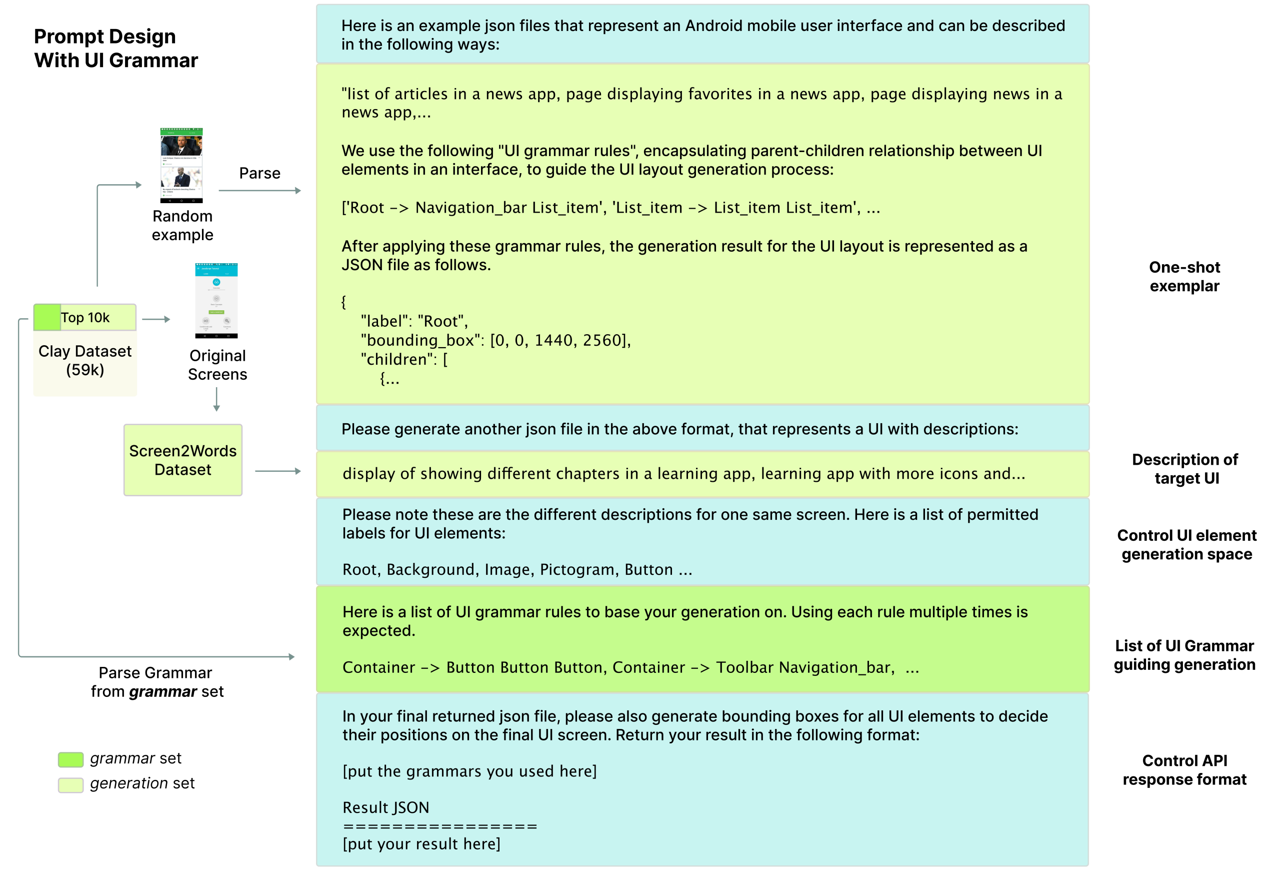
An important step in constructing the prompt with UI grammar is selecting which screens from the CLAY dataset to parse grammar from. When generating a layout using descriptions of screen $S$ from the original CLAY dataset, if we also input grammars parsed from $S$ into the prompt, data leakage occurs as screen $S$ can be reconstructed from its own grammars in a straightforward manner. To avoid this, we conduct a 20/80 random split of the CLAY dataset and use grammars parsed from the 20% grammar set to guide the generation of the 80% generation set.

In addition, from our observation, many screens from the same app packages in CLAY share similar layout structures. Consequently, we splitted the dataset by apps in order to avoid the data leak caused by having screens from the same app package in both sets.

## 3. Initial Experiments

In May 2023, we used OpenAI's GPT-4 API to conduct a preliminary experiment comparing the 2 proposed pipelines for UI layout generation. We used the `gpt-4-0314` version of GPT-4 with a `max_token` of $2,000$ and `temperature` of $0.7$.

**Dataset** For both prompt designs, we pre-process the UI view hierarchy files from CLAY by removing all attributes of UI elements but `label` and `bounds`, as all others are not necessary for our layout generation task. To further ensure the generation quality, we work with a subset of the top $10k$ UI screens from CLAY with an app review higher than $4.3$ and download of more than $10k$ in Google Play Store for our generation. These two thresholds serve as



Here is an example json files that represent an Android mobile user interface and can be described in the following ways:

"list of articles in a news app, page displaying favorites in a news app, page displaying news in a news app,...

We use the following "UI grammar rules", encapsulating parent-children relationship between UI elements in an interface, to guide the UI layout generation process:

['Root -> Navigation_bar List_item', 'List_item -> List_item List_item', ...

After applying these grammar rules, the generation result for the UI layout is represented as a JSON file as follows.

```
{
   "label": "Root",
   "bounding_box": [0, 0, 1440, 2560],
   "children": [
      {...
```

Please generate another json file in the above format, that represents a UI with descriptions:

display of showing different chapters in a learning app, learning app with more icons and...

Please note these are the different descriptions for one same screen. Here is a list of permitted labels for UI elements:

Root, Background, Image, Pictogram, Button ...

Here is a list of UI grammar rules to base your generation on. Using each rule multiple times is expected.

Container -> Button Button Button, Container -> Toolbar Navigation_bar, ...

In your final returned json file, please also generate bounding boxes for all UI elements to decide their positions on the final UI screen. Return your result in the following format:

[put the grammars you used here]

Result JSON
================
[put your result here]

*Figure 3.* Prompt 2 design for generation with UI grammar

quality filters and are manually defined to balance the need for a sufficiently large sample size against the desire for high-quality app representation.

Given OpenAI's API response rate and call limits, it is hard to quickly generate a large number of results. In this work-in-progress, we have conducted an initial experiment on a batch of 192 UI screens from the top apps in Clay and report the preliminary evaluation results as follows. Visualization of example generation results is shown in 4.

## 4. Preliminary Evaluations

Here we report preliminary evaluations of our UI layout generation results against 3 common metrics commonly used in this domain: Maximum Intersection Over Union (MaxIoU), Alignment, and Overlap. [1] The MaxIoU value is calculated between the generation screen $S'$ and the original screen $S$ from Clay the provided screen summaries as part of the prompt. Alignment and Overlap are both calculated over the generated result $S$ only.

Please note that in order to more accurately evaluate the visual quality of the generated UI layouts, we removed 5 types of UI elements that are commonly invisible on screens [2] from the results before evaluation.

**Results** In Table 1, we can see that in our initial experiment, GPT-4 performed well on *overlap* without grammar, and on *alignment* with and without grammar, having close or even better metric performance than real data. The *overlap* result for both prompt designs achieved $0.00$, meaning every element aligns with at least 1 other element on the screen. This is consistent with the visual appearance of the generation results. While we did not specifically mention the need to align UI elements or avoid element overlap in our prompt, GPT-4 was able to generate high-quality results against these metrics. In addition, introducing UI grammar to guide GPT-4's layout generation process slightly increased the MaxIoU performance. On this metric, GPT-4 with grammar

[1] Refer to Jing et al. (2023) for definitions of these metrics.

[2] Namely: Root, Background, List_item, Card_view, and Container

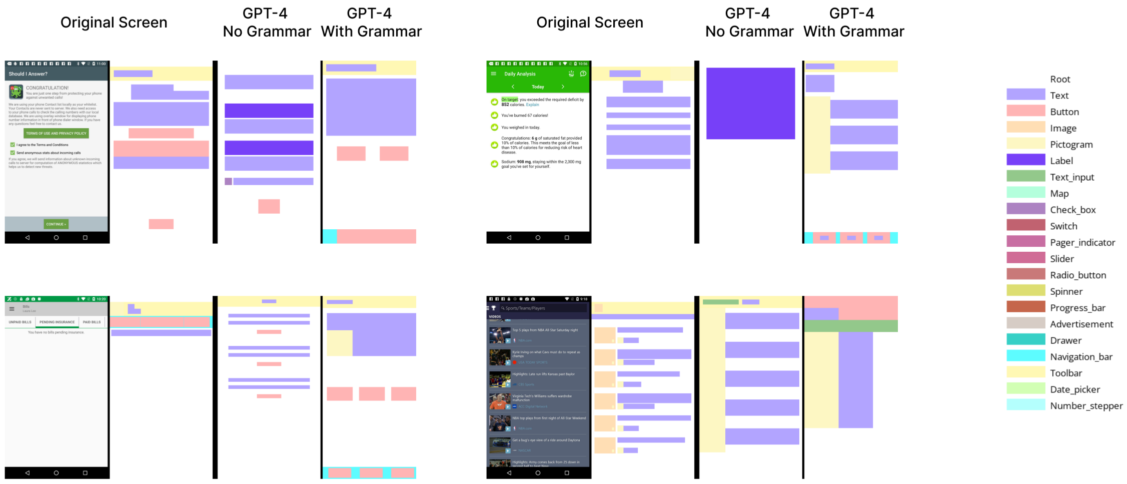


*Figure 4.* Visualizing the generation results. In each 4-screens group, the left 2 images are the original image and its parsed bounding boxes, while the right 2 are GPT-4 generated results. The original image's description from SCREEN2WORDS was used in the prompt for generating the right 2 layouts.

is comparable with some general layout generation models trained on large datasets as reported in (Jing et al., 2023), demonstrating LLMs' in-context learning ability in mobile UI layout generation.

While we did not explicitly restrict GPT-4 on using only the provided grammar set [3], 83.8% of the rules GPT-4 reported to be using for generation came from the provided grammar. This showed that GPT-4 was not entirely restricted by the grammar we provided, demonstrating the flexibility of the model and our approach.

## 5. Discussion and Future Work

Our experimentation with LLMs for UI layout generation has demonstrated LLMs' promising ability on this task. However, we believe LLMs like GPT-4 also have the potential capability to generate content along with layouts to create mid-fi to high-fi prototypes. The potential to combine LLMs with existing UI templates or design systems such as Google Material Design will enable more automated, customized, and efficient UI prototyping techniques.

We argue that besides improving LLMs' generation quality on metrics like MaxIoU, by introducing UI grammar as an intermediary representation in the generation process, our approach could increase the explainability and users' controllability of black-box pre-trained LLMs:

- ***Explainability:*** By reviewing the UI grammar employed in LLMs' generation processes, users could gain a better understanding of LLMs' internal generation mechanisms. Our approach differs from a post-hoc explanation request for LLMs, in that our approach can be more easily verified through an easy comparison between the grammars employed and the final UI structure. On the other hand, post-hoc explanation requests (e.g. a follow-up question such as *"explain why you generated this result"*), while similar to how humans provide justifications, do not necessarily reflect the actual generation mechanism.
- ***Controllability:*** With UI grammar as an intermediary representation in the generation process, users can obtain higher control of the generation results if enabled to modify or replace the grammar provided to the LLMs in prompts. Future applications can build upon such model architecture and provide users with more ways to interact with UI grammar in the prompts (e.g. directly selecting which apps to extract grammar from) to improve the controllability of LLMs in similar generation tasks.

In Section 2.3.2 we have discussed the potential of data leak when using the *natural language description* and *UI grammar* derived from the same screen. But on the other hand, since UI grammar represents different ways of organizing

[3] Specifically, we used the wording "Here is a list of UI grammar rules to base your generation on. Using each rule multiple times is expected".

| | MaxIoU ↑ | Overlap ↓ | Alignment ↓ |
|---|---|---|---|
| GPT-4 no grammar | 0.29 | 8.14 | 0.00 |
| GPT-4 with grammar | 0.34 | 12.47 | 0.00 |
| Real Data | —- | 8.58 | 0.00 |

*Table 1.* Quantitative comparisons of our 2 approaches and real data from CLAY

and designing UI elements on a screen, we could potentially use UI grammar as a proxy to control characteristics of generation results. One possible use case is generating certain styles of UI, by extracting grammar specifically from screens in compliance with a company's design guidelines.

Continuing this initial study, we have planned the below agendas for our follow-up work:

1. Making improvements to our pipeline and prompt structure by extending *UI grammar* with each rule's occurrence probability;
2. Integrating reasoning steps for the target user, information to display, and supported actions of a UI through Chain-of-Thought prompting (Wei et al., 2022b), a workflow resembling the one of human UI designers;
3. Conducting multi-faceted layout generation assessments involving human evaluators and more quantitative metrics (e.g. Fréchet inception distance) at a larger scale, to ensure the robustness and applicability of our models;
4. Experiementing the feasibility of generating high-fidelity UI prototypes with LLMs, as discussed above, and potentially build interactive design-support tools to speed up UI prototyping.

## 6. Conclusion

In this work, we explored Large Language Models' ability to generate mobile user interface layouts through 1-shot, in-context learning. We proposed *UI grammar*, a novel approach to represent the hierarchical relationship between UI elements, and incorporated it into our prompts to steer UI layout generation. Our preliminary results demonstrated LLMs' capabilities to generate high-quality UI layouts with competitive performance, as well as the usefulness of UI grammar in improving certain aspects of generation qualities. We conclude by discussing the implications of using LLMs and UI grammar for future research.

## References

Arroyo, D. M., Postels, J., and Tombari, F. Variational transformer networks for layout generation. (arXiv:2104.02416), Apr 2021. URL http://arxiv.org/abs/2104.02416. arXiv:2104.02416 [cs].

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

Cheng, C.-Y., Huang, F., Li, G., and Li, Y. Play: Parametrically conditioned layout generation using latent diffusion. (arXiv:2301.11529), Jan 2023. URL http://arxiv.org/abs/2301.11529. arXiv:2301.11529 [cs].

Deka, B., Huang, Z., Franzen, C., Hibschman, J., Afergan, D., Li, Y., Nichols, J., and Kumar, R. 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*, pp. 845–854, 2017.

Dow, S., MacIntyre, B., Lee, J., Oezbek, C., Bolter, J. D., and Gandy, M. Wizard of oz support throughout an iterative design process. *IEEE Pervasive Computing*, 4 (4):18–26, 2005.

Earley, J. An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2):94–102, 1970.

Feng, W., Zhu, W., Fu, T.-j., Jampani, V., Akula, A., He, X., Basu, S., Wang, X. E., and Wang, W. Y. Layoutgpt: Compositional visual planning and generation with large language models. (arXiv:2305.15393), May 2023. URL http://arxiv.org/abs/2305.15393. arXiv:2305.15393 [cs].

Frost, B. *Atomic design*. Brad Frost Pittsburgh, 2016.

Gupta, K., Lazarow, J., Achille, A., Davis, L., Mahadevan, V., and Shrivastava, A. Layouttransformer: Layout generation and completion with self-attention. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 984–994, Montreal, QC, Canada, Oct 2021. IEEE. ISBN 978-1-66542-812-5. doi: 10.1109/ICCV48922.2021.00104. URL https://ieeexplore.ieee.org/document/9710883/.

Huang, F., Li, G., Zhou, X., Canny, J. F., and Li, Y. Creating user interface mock-ups from high-level text descriptions with deep-learning models. (arXiv:2110.07775), Oct 2021. URL http://arxiv.org/abs/2110.07775. arXiv:2110.07775 [cs].

Hui, M., Zhang, Z., Zhang, X., Xie, W., Wang, Y., and Lu, Y. Unifying layout generation with a decoupled diffusion model. (arXiv:2303.05049), Mar 2023. URL http://arxiv.org/abs/2303.05049. arXiv:2303.05049 [cs].

Jiang, Y., Lu, Y., Nichols, J., Stuerzlinger, W., Yu, C., Lutteroth, C., Li, Y., Kumar, R., and Li, T. J.-J. Computational approaches for understanding, generating, and adapting user interfaces. In *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems*, CHI EA '22, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450391566. doi: 10.1145/3491101.3504030. URL https://doi.org/10.1145/3491101.3504030.

Jiang, Y., Lu, Y., Lutteroth, C., Li, T. J.-J., Nichols, J., and Stuerzlinger, W. The future of computational approaches for understanding and adapting user interfaces. In *Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems*, CHI EA '23, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9781450394222. doi: 10.1145/3544549.3573805. URL https://doi.org/10.1145/3544549.3573805.

Jing, Q., Zhou, T., Tsang, Y., Chen, L., Sun, L., Zhen, Y., and Du, Y. Layout generation for various scenarios in mobile shopping applications. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pp. 1–18, Hamburg Germany, Apr 2023. ACM. ISBN 978-1-4503-9421-5. doi: 10.1145/3544548.3581446. URL https://dl.acm.org/doi/10.1145/3544548.3581446.

Kargaran, A. H., Nikeghbal, N., Heydarnoori, A., and Schütze, H. Menucraft: Interactive menu system design with large language models. *arXiv preprint arXiv:2303.04496*, 2023.

Kikuchi, K., Simo-Serra, E., Otani, M., and Yamaguchi, K. Constrained graphic layout generation via latent optimization. In *Proceedings of the 29th ACM International Conference on Multimedia*, pp. 88–96, Virtual Event China, Oct 2021. ACM. ISBN 978-1-4503-8651-7. doi: 10.1145/3474085.3475497. URL https://dl.acm.org/doi/10.1145/3474085.3475497.

Knearem, T., Khwaja, M., Gao, Y., Bentley, F., and Kliman-Silver, C. E. 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*, pp. 1–6, 2023.

Kong, J., Ates, K. L., Zhang, K., and Gu, Y. Adaptive mobile interfaces through grammar induction. In *2008 20th IEEE International Conference on Tools with Artificial Intelligence*, volume 1, pp. 133–140. IEEE, 2008.

Kong, X., Jiang, L., Chang, H., Zhang, H., Hao, Y., Gong, H., and Essa, I. Blt: bidirectional layout transformer for controllable layout generation. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XVII*, pp. 474–490. Springer, 2022.

Lee, H.-Y., Jiang, L., Essa, I., Le, P. B., Gong, H., Yang, M.-H., and Yang, W. Neural design network: Graphic layout generation with constraints. (arXiv:1912.09421), Jul 2020. URL http://arxiv.org/abs/1912.09421. arXiv:1912.09421 [cs].

Li, G., Baechler, G., Tragut, M., and Li, Y. Learning to denoise raw mobile ui layouts for improving datasets at scale. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, pp. 1–13, 2022.

Li, J., Yang, J., Hertzmann, A., Zhang, J., and Xu, T. Layoutgan: Generating graphic layouts with wireframe discriminators. *arXiv preprint arXiv:1901.06767*, 2019.

Li, T. J.-J., Labutov, I., Li, X. N., Zhang, X., Shi, W., Ding, W., Mitchell, T. M., and Myers, B. A. Appinite: A multi-modal interface for specifying data descriptions in programming by demonstration using natural language instructions. In *2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pp. 105–114, 2018. doi: 10.1109/VLHCC.2018.8506506.

Li, T. J.-J., Popowski, L., Mitchell, T., and Myers, B. A. Screen2vec: Semantic embedding of gui screens and gui components. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI '21, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450380966. doi: 10.1145/3411764.3445049. URL https://doi.org/10.1145/3411764.3445049.

Li, Y., Amelot, J., Zhou, X., Bengio, S., and Si, S. Auto completion of user interface layout design using transformer-based tree decoders. (arXiv:2001.05308), Jan 2020. URL http://arxiv.org/abs/2001.05308. arXiv:2001.05308 [cs].

Liu, Z., Chen, C., Wang, J., Chen, M., Wu, B., Che, X., Wang, D., and Wang, Q. Chatting with gpt-3 for zero-shot human-like mobile automated gui testing. *arXiv preprint arXiv:2305.09434*, 2023.

Lu, Y., Zhang, C., Zhang, I., and Li, T. J.-J. Bridging the gap between ux practitioners' work practices and ai-enabled design support tools. In *CHI Conference on Human Factors in Computing Systems Extended Abstracts*, pp. 1–7, 2022.

Martelaro, N. and Ju, W. Woz way: Enabling real-time remote interaction prototyping & observation in on-road vehicles. In *Proceedings of the 2017 ACM conference on computer supported cooperative work and social computing*, pp. 169–182, 2017.

Nielsen, J. Usability inspection methods. In *Conference companion on Human factors in computing systems*, pp. 413–414, 1994.

Nielsen, J. Ten usability heuristics, 2005.

Sarker, M. K., Zhou, L., Eberhart, A., and Hitzler, P. Neuro-symbolic artificial intelligence. *AI Communications*, 34 (3):197–209, 2021.

Sobolevsky, A., Bilodeau, G.-A., Cheng, J., and Guo, J. L. C. Guilget: Gui layout generation with transformer. (arXiv:2304.09012), Apr 2023. URL http://arxiv.org/abs/2304.09012. arXiv:2304.09012 [cs].

Talton, J., Yang, L., Kumar, R., Lim, M., Goodman, N., and Měch, R. Learning design patterns with bayesian grammar induction. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*, pp. 63–74, 2012.

Wang, B., Li, G., Zhou, X., Chen, Z., Grossman, T., and Li, Y. Screen2words: Automatic mobile ui summarization with multimodal learning. In *The 34th Annual ACM Symposium on User Interface Software and Technology*, pp. 498–510, 2021.

Wang, B., Li, G., and Li, Y. Enabling conversational interaction with mobile ui using large language models. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pp. 1–17, 2023.

Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022a.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Chi, E., Le, Q., and Zhou, D. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022b.

Zimmerman, J. and Forlizzi, J. Speed dating: providing a menu of possible futures. *She Ji: The Journal of Design, Economics, and Innovation*, 3(1):30–50, 2017.