D2S2: Drag 'n' Drop Mobile App Screen Search

Soumik Mohian
soumik.mohian@mavs.uta.edu
University of Texas at Arlington
Arlington, Texas, USA

Tony Tang
ttt9489@mavs.uta.edu
University of Texas at Arlington
Arlington, Texas, USA

Tuan Trinh
tqt5663@mavs.uta.edu
University of Texas at Arlington
Arlington, Texas, USA

Don Dang
dpd5574@mavs.uta.edu
University of Texas at Arlington
Arlington, Texas, USA

Christoph Csallner
csallner@uta.edu
University of Texas at Arlington
Arlington, Texas, USA

ABSTRACT

The lack of diverse UI element representations in publicly available datasets hinders the scalability of sketch-based interactive mobile search. This paper introduces D2S2, a novel approach that addresses this limitation via drag-and-drop mobile screen search, accommodating visual and text-based queries. D2S2 searches 58k Rico screens for relevant UI examples based on UI element attributes, including type, position, shape, and text. In an evaluation with 10 novice software developers D2S2 successfully retrieves target screens within the top-20 search results in 15/19 attempts within a minute. The tool offers interactive and iterative search, updating its search results each time the user modifies the search query. Interested users can freely access D2S2 (http://pixeltoapp.com/D2S2), build on D2S2 or replicate results via D2S2's open-source implementation (https://github.com/toni-tang/D2S2), or watch D2S2's video demonstration (https://youtu.be/fdoYiw8lAn0).

CCS CONCEPTS

• **Human-centered computing** → *Interface design prototyping*; • **Software and its engineering** → **Software prototyping**; **Search-based software engineering**.

KEYWORDS

User interface design, prototyping, information retrieval, design examples, interactive screenshot search

ACM Reference Format:
Soumik Mohian, Tony Tang, Tuan Trinh, Don Dang, and Christoph Csallner. 2023. D2S2: Drag 'n' Drop Mobile App Screen Search. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '23), December 3–9, 2023, San Francisco, CA, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3611643.3613100

1 INTRODUCTION

Iterative app screen search, while an exciting area of recent work [16, 17, 18], still faces many challenges. First, Google image search is fast, searches many images from the open web, and supports text-based search queries. But searching for an app screen via text queries remains clumsy, especially when looking for screens that contain certain UI elements in specific locations. Such searches result in long text queries and produce few relevant results. Recent work has searched the widely-used Rico dataset via a combination of text search [18] and sketched element doodles [16, 17]. These approaches support a few UI element types via deep learning. Expanding their scope would require additional specialized training data, which must be collected and curated.

When designing mobile applications, studying real-world examples aids in gathering requirements, analyzing current trends, and cultivating motivation to develop a compelling mobile app [9, 10]. Given the broad and rapidly expanding market for mobile apps, an efficient mobile app screen search tool becomes valuable.

Designers commonly use drag-and-drop tools (e.g., Figma [2]) to create wireframes. Similarly, software developers utilize drag-and-drop-based visual kits (e.g., the Android Layout Editor [1] or Prototypr [3]) for UI development. The popularity of these techniques is growing due to their user-friendly nature, intuitive interfaces, and since they do not require specialized technical expertise [13]. D2S2 offers an interactive solution via drag and drop for mobile screen search.

D2S2 is for novice users who want help creating a complete UI design during the early software development stages. Users can search for mobile screens by dragging and dropping UI elements on the canvas. The tool's search interface includes basic features such as undo and redo. Users can also add plain text and put text in a text-button. As a user adds, removes, resizes, and moves UI elements, D2S2 searches through 58k Rico [7] screens to fetch UI examples based on UI element type, position, element shape, and texts as shown in Figure 1. D2S2 fetches the top-20 screens and displays them in its website's top-pick screen search results section.

We recruited 10 software developers without prior UI/UX design training to assess D2S2's effectiveness. The participants searched for a given target Rico screen with D2S2 until the screen appeared in D2S2's top-20 search results. In our experiment, D2S2 successfully obtained 15/19 target screens within a minute and 19/19 within four minutes. D2S2 further retrieved more relevant mobile screens than the other closely related competitor, Google image search. In summary, this paper makes the following major contributions.

- D2S2 is the first interactive drag-and-drop app screen search tool. After each query change it updates its search results.

ESEC/FSE '23, December 3–9, 2023, San Francisco, CA, USA

ACM ISBN 979-8-4007-0327-0/23/12.
https://doi.org/10.1145/3611643.3613100

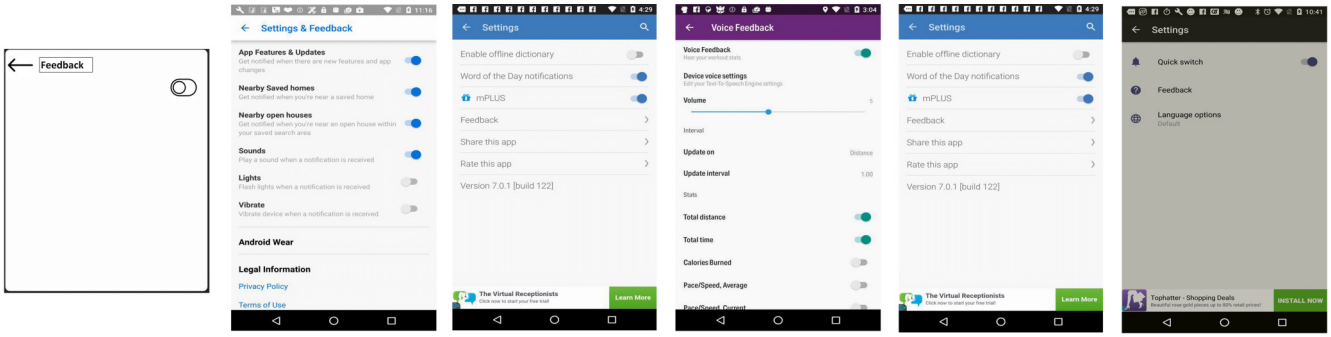

Figure 1: Sample D2S2 search query consisting of 3 UI elements (left); searching through 58k mobile app screens, D2S2's top-five search result screens (right) all contain the query's UI elements at about the locations where they appear in the query screen.

- D2S2 searches 58k Android screens and is freely available (**http://pixeltoapp.com/D2S2**).
- In a preliminary user study, D2S2 performed similarly as the deep-learning based TpD (but without requiring training) and better than Google image search.
- D2S2's implementation (https://github.com/toni-tang/D2S2) is available under a permissive open-source license.

2 BACKGROUND

D2S2 searches 58k mobile Android app screenshots from the Rico dataset by Deka et al. [7]. Each screenshot has a corresponding DOM-tree container hierarchy, where each UI element is described by its Android class name, x/y coordinates, textual information, and on-screen visibility. Liu et al. expanded on this dataset by collecting 73k screen elements, categorizing them into 25 types of UI components, and further dividing text buttons into 197 and icon into 135 sub-classes [15]. D2S2 incorporates several common Android UI elements identified by Liu et al.

Previous studies have explored using sketches and wireframe images to search for relevant mobile screens. However, wireframe-based approaches such as Swire rely on complete wireframe images to identify screens with similar visual characteristics, often not considering UI element type and text within the screen [4, 5]. Dependence on an entire wireframe image does not support the iterative nature of the design process.

Besides Google Image Search, our closest competitors are PSDoodle [16, 17] and TpD [18], which offer an interactive and iterative approach to searching mobile screens. PSDoodle employs a deep neural network to identify sketched UI elements and then computes a ranking score for Rico's screens based on various factors, including UI element type, position, and shape. TpD extends PSDoodle by adding a text-based search that matches a text query with visible text on the mobile screen and UI element descriptions. Notably, TpD allows queries to contain text, UI element sketches, or both.

3 OVERVIEW AND DESIGN

To create a search system that is easy to use, we followed a user-centric approach. Via the Figma [2] graphical design tool, we thus first created a UI prototype (Figure 2), showed the prototype to 11 computer science undergraduate students, and collected their feedback. By incorporating their feedback, we then iteratively enhanced D2S2's user experience, mostly by refining D2S2's UI. All user feedback is in D2S2's repository.

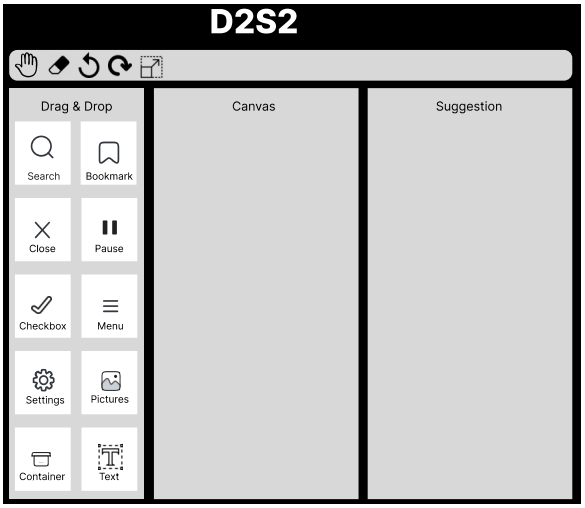

Figure 2: Initial D2S2 UI mock-up.

3.1 User Interface & Query Language

In D2S2, a search query consists of a set of UI elements arranged on a canvas that models the screenshot of a mobile app. Starting with an empty canvas, the user interactively refines this canvas, adding and adjusting UI elements as they should appear on the desired app screens. Each time the user modifies this search query, D2S2 retrieves matching app screens that have the query's UI elements at about the location the user placed them on the canvas. A part of the search is matching any texts the user added to the search query with screens' text contents and descriptions of their UI elements.

Figure 3 shows D2S2's current UI. Besides moving the app bar to the bottom, the biggest change is allowing users to search D2S2's library of 52 built-in UI elements by the UI element's name and various synonyms. Figure 4 lists these 52 UI elements in the order D2S2' UI presents them. The order is TpD's UI elements first, then

UI elements identified by Liu et al., ordered by how common they appear in Rico screens [15].

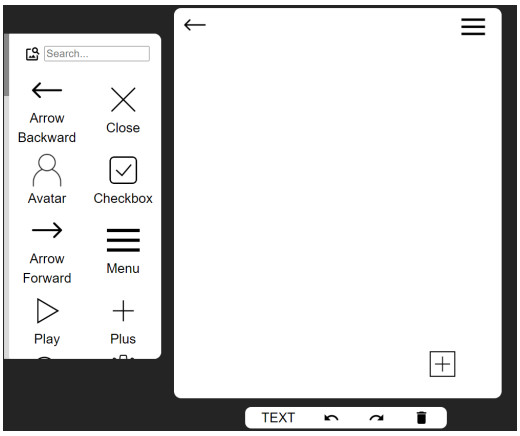

Figure 3: D2S2's webpage: Searchable UI elements (left), canvas with 3 user-placed UI elements (right), app bar (bottom). Cropped screenshot (from http://pixeltoapp.com/D2S2), not shown: D2S2's top-20 search result screen gallery.

Liu et al. classified the UI elements of Rico's screens into 25 categories. 11/25 categories are various container types, which D2S2 represents via a single general container. D2S2 directly supports 11/14 of the remaining categories, plus 44/135 icon types. Combining 3/56 of these UI elements due to their similar visual representation with another UI element (e.g., slider vs. slider icon) yields D2S2's 52 UI element types plus text.

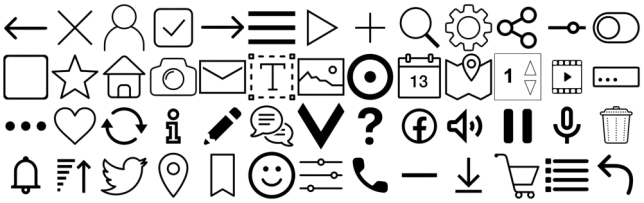

Figure 4: A user builds an app screen search query on D2S2's canvas by dragging and adjusting text or these 52 UI elements.

The user searches (or scrolls) the UI element list, selects a UI element and drags and drops it on the canvas. The user can interact with the UI element, i.e., to move or resize it there. The app bar at the bottom of the canvas allows undoing and redoing the last element modifications and clearing the screen. The user can add text either via a text-button from the UI element collection or via the app bar's "TEXT" feature. The latter adds a text element to the canvas the user can manipulate like any other UI element. Clicking on such a text element enables modifying its text content.

As in the earlier PSDoodle and TpD, UI elements may be nested, i.e., to support UI elements grouped in a container element. D2S2 encodes the canvas's current state as a set of 6-tuples of the form (x, y, w, h, c, t), one tuple per UI element on the canvas. The tuple lists an element's left-top corner's location in pixel-space, the UI element's width, height, category, and text content (for text and text buttons). The D2S2 webpage is written in React, as it provides client-side rendering [14] and efficiently manages various events such as drag-start, drag-end, and the undo/redo functionality.

3.2 D2S2's Back-end

Figure 5 illustrates D2S2's overall architecture, which consists of D2S2's webpage front-end and its AWS-hosted back-end. Each time the user modifies the query, D2S2 sends the updated query's tuple encoding via HTTP post request to AWS EC2. For the current query, the D2S2 back-end ranks its 58k Rico screens and sends the IDs of the top-20 ranked screens back to the front end. The front-end then retrieves a lower-resolution version of the 20 corresponding screen images from D2S2's AWS S3 bucket and displays them in the top-pick gallery. When clicking on a result screen, D2S2 fetches and displays a higher-resolution version of the result screen.

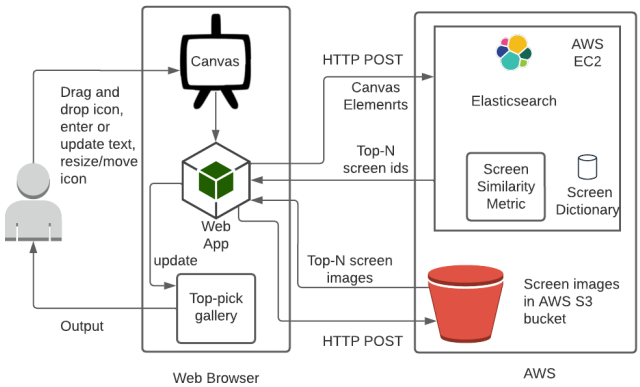

Figure 5: D2S2's architecture: A user drags or adjusts Figure 4 UI elements on the Figure 3 D2S2 front-end webpage (left), which communicates with its AWS-hosted back-end (right).

To rank its 58k screens, D2S2 uses TpD's infrastructure, which in turn builds on PSDoodle's. For non-text UI elements, D2S2 uses PSDoodle's screen scoring scheme (which TpD similarly reused). Specifically, D2S2 divides a mobile app screen into 24 equally sized tiles (6 along the width and 4 along the height) and maintains TpD's tile configuration. The main change is in more than doubling TpD's 23 UI element classes to D2S2's 52. In the back-end, this is straightforward by adding one screen ID index for each of the additional UI element classes to allow fast screen lookup.

For text elements, D2S2 reuses TpD's pipeline [18], which preprocesses the Rico screens' text contents and UI element descriptions (remove stop words, identify names, lemmatization, adding synonyms via contextual analysis, and tagging text content with on-screen location). As for text contents TpD only supports four different screen areas (top-left, top-right, bottom-left, and bottom-right), D2S2 first maps the location of a text element to one of these four TpD screen areas. As TpD, D2S2 then uses ElasticSearch with Levenshtein edit distance one, to heuristically also match slightly mis-typed user-provided text to screen contents.

4 D2S2 USAGE

To compare with its closest competitors TpD and Google Image Search, we enlisted 10 computer science students who did not have formal UI/UX design training. While the participants differ, we recruited them using the same criteria the TpD study used. To ensure diversity, we selected five individuals with and five without previous mobile app development experience. All participants were early-stage undergraduates aged 20–25. As a token of appreciation, each participant received USD 10 compensation. Specifically, we are interested in the following research questions.

RQ1 How does D2S2 compare with TpD, in terms of total time of the interactive search, final queries' UI element counts, and final queries' top-k screen retrieval accuracy?

RQ2 How does D2S2 compare with Google Image Search on a free user query, for producing relevant top-20 search results?

For each participant, we had one video conference of about 30 minutes that started with us explaining D2S2's objectives. We then demonstrated the search process for an icon, dragging the icon to the canvas, resizing and adjusting the icon's position on the canvas, the functionality of the undo/redo/clear-screen buttons, and how to add text using the text and text-button features. Each participant accessed D2S2 over the internet via a web browser on their personal machine. We used D2S2's standard setup as a website hosted on an Amazon AWS EC2 general-purpose instance (t2.large), featuring two virtual CPUs and 8GB RAM. D2S2's repository contains all experimental results.

4.1 Similar Screen Search Performance as TpD

For this second part of a participant meeting, we used the 26 randomly selected Rico target screens used by TpD's evaluation. For each participant, we randomly selected from this pool one target screen per search session. We instructed the participant to create a query that would retrieve the target screen and refine the query until the target screens appeared in D2S2's top-20 results.

Table 1: Participants' search sessions for target screens via D2S2 (left) and free search via D2S2 and Google Images (right): t = search session's total time; n = final query's UI elements (including texts); r = target screen's rank for final query; G/D = top-20 Google/D2S2 results participant judged relevant.

	Target 1			Target 2			Free 1		Free 2	
	t[s]	n	r	t[s]	n	r	G	D	G	D
1	40	4	2	27	3	1	2	18	18	0
2	120	9	1	52	3	2	1	3	3	4
3	37	3	4	48	4	3	5	20	0	20
4	50	3	10	23	2	4	8	16	10	20
5	70	4	1	240	2	1	0	7	4	13
6	59	3	12	196	15	14	3	16	4	20
7	63	3	18	51	3	2	7	11	1	3
8	42	4	16	51	3	7	7	10	0	7
9	39	3	9	42	7	17	1	10	1	19
10	50	4	8	-	-	-	2	5	-	-

Nine participants used D2S2 twice, and one used it once, yielding 19 D2S2 search sessions. For each such search session, we recorded the total time, the number of UI elements and texts in the participant's final query, and the target screen's rank in D2S2's results for that final query. D2S2's top-k retrieval accuracy is the number of search sessions in which D2S2 ranks the target screen in its answer to the participant's final query in the top-k. We use top-k retrieval accuracy, as the metric is widely used to evaluate related work [6, 8, 11] and correlates with user satisfaction [12].

Table 1 summarizes the results. Comparing with TpD's results is a little tricky as TpD's participants were instructed to search until the target screen appears in TpD's top-10 search results or the search exceeds 3 minutes. So TpD participants were encouraged to spend a bit of additional time to refine a query. With this caveat, the overall results for D2S2 and TpD are similar. D2S2's top-20 screen retrieval accuracy is 100% (19/19) vs. TpD's 97% (29/30).

D2S2's total search session time was at least 23 seconds, 240s maximum, 68s average, and 50s median. This compares to a 5s minimum, 156s maximum, average 45s, and 35s median for TpD. Contributing to TpD shorter search sessions are TpD's experimental setup (which allowed participants to practice using TpD for some 10 minutes before collecting results) and D2S2 having more than twice the number of UI elements to choose from for a search query. We observed participants using significant time browsing the UI elements available in D2S2 and selecting the correct UI element.

4.2 More Targeted Than Google Image Search

In this final meeting part, we instructed each participant to formulate a Google-style search query and perform a corresponding search using both D2S2 and Google image search (an example of a participant's query is "mobile screen menu icon top left and search icon top right"). We then asked the participant to rate each result in both tools' top-20 results as relevant or non-relevant to the query.

Participants judged 20% (77/380) of Google image search's results as relevant and 58% (222/380) of D2S2's result screens. D2S2's 58% relevance here is largely in line with TpD's 52% reported for searches for a given target screen [18]. Given D2S2's and TpD's slightly different experimental setups, it is hard to draw conclusions about their relative performance. For the search scenario over 58k Rico screens, both tools clearly perform better than Google Image Search.

5 CONCLUSIONS

Current sketch-based iterative mobile screen search has limitations in supporting many UI elements. Drag-and-drop provides a flexible alternative. D2S2' provides an interactive, drag-and-drop search that displays results interactively. The tool is freely available and has undergone user testing, demonstrating its effectiveness. D2S2 is a promising solution for novice users who require assistance creating a comprehensive UI design in the initial development phases.

ACKNOWLEDGEMENTS

Christoph Csallner has a potential research conflict of interest due to a financial interest with Microsoft and The Trade Desk. A management plan has been created to preserve objectivity in research in accordance with UTA policy. This material is based upon work supported by the National Science Foundation (NSF) under Grant No. 1911017.

REFERENCES

[1] 2009. Android Developers. https://developer.android.com/studio/write/layout-editor. Accessed: Aug 2023.

[2] 2019. Figma. https://www.figma.com. Accessed: Aug 2023.

[3] 2022. Prototypr. https://prototypr.io. Accessed: Aug 2023.

[4] Sara Bunian, Kai Li, Chaima Jemmali, Casper Harteveld, Yun Fu, and Magy Seif Seif El-Nasr. 2021. VINS: Visual Search for Mobile User Interface Design. In *Proc. CHI Conference on Human Factors in Computing Systems*. 1–14.

[5] Jieshan Chen, Chunyang Chen, Zhenchang Xing, Xin Xia, Liming Zhu, John Grundy, and Jinshui Wang. 2020. Wireframe-based UI design search through image autoencoder. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 29, 3 (2020), 1–31.

[6] John Collomosse, Tu Bui, and Hailin Jin. 2019. Livesketch: Query perturbations for guided sketch-based visual search. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2879–2887.

[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 *Proc. 30th Annual ACM Symposium on User Interface Software and Technology (UIST)*. ACM, 845–854.

[8] Sounak Dey, Pau Riba, Anjan Dutta, Josep Llados, and Yi-Zhe Song. 2019. Doodle to search: Practical zero-shot sketch-based image retrieval. In *Proc. IEEE/CVF conference on computer vision and pattern recognition*. 2179–2188.

[9] Claudia Eckert and Martin Stacey. 2000. Sources of inspiration: a language of design. *Design studies* 21, 5 (2000), 523–538.

[10] Scarlett R Herring, Chia-Chen Chang, Jesse Krantzler, and Brian P Bailey. 2009. Getting inspired! Understanding how and why examples are used in creative design practice. In *Proc. SIGCHI conference on human factors in computing systems*. 87–96.

[11] Forrest Huang, John F. Canny, and Jeffrey Nichols. 2019. Swire: Sketch-based user interface retrieval. In *Proc. CHI Conference on Human Factors in Computing Systems*. ACM.

[12] Scott B Huffman and Michael Hochster. 2007. How well does result relevance predict session satisfaction?. In *Proc. 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 567–574.

[13] Johnsymol Joy. 2018. Review on different types of drag and drop mobile app development platforms. *International Journal of Computer Sciences and Engineering* 6, 11 (Nov. 2018), 864–866.

[14] Mochammad Fariz Syah Lazuardy and Dyah Anggraini. 2022. Modern Front End Web Architectures with React.Js and Next.Js. *International Research Journal of Advanced Engineering and Science* 7, 1 (2022), 132–141.

[15] Thomas F Liu, Mark Craft, Jason Situ, Ersin Yumer, Radomir Mech, and Ranjitha Kumar. 2018. Learning design semantics for mobile apps. In *Proc. 31st Annual ACM Symposium on User Interface Software and Technology (UIST)*. 569–579.

[16] Soumik Mohian and Christoph Csallner. 2022. PSDoodle: Fast app screen search via partial screen doodle. In *Proc. 9th IEEE/ACM International Conference on Mobile Software Engineering and Systems*. 89–99.

[17] Soumik Mohian and Christoph Csallner. 2022. PSDoodle: Searching for app screens via interactive sketching. In *Proc. 9th IEEE/ACM International Conference on Mobile Software Engineering and Systems*. 84–88.

[18] Soumik Mohian and Christoph Csallner. 2023. Searching Mobile App Screens via Text + Doodle. arXiv:2305.06165 [cs.IR]

Received 2023-05-11; accepted 2023-07-20